EPFL

Thèse n° 8121

Robot skills learning with Riemannian manifolds: Leveraging geometry-awareness in robot learning, optimization and control

Présentée le 31 juillet 2020

à la Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Noémie Laure Gwendoline JAQUIER

Acceptée sur proposition du jury

Prof. A. Ijspeert, président du jury
Prof. H. Bourlard, Dr S. Calinon, directeurs de thèse
Prof. F. Park, rapporteur
Dr J.-B. Mouret, rapporteur
Dr N. Flammarion, rapporteur

École
polytechnique
fédérale
de Lausanne

2020

Dreams can come true, but there is a secret. They're realized through the magic of persistence, determination, commitment, passion, practice, focus and hard work. They happen a step at a time, manifested over years, not weeks.
— Elbert Hubbard

To my parents,
for their unconditional support.

Acknowledgements

A PhD thesis is a long, sometimes tortuous, but also precious and joyful journey that one does not travel alone. Therefore, I owe the beginning of this thesis to the people around me who made that journey shine by their presence, support, and affection throughout these years.

First of all, I would like to thank my advisors Prof. Hervé Bourlard and Dr. Sylvain Calinon for giving me the opportunity of pursuing a thesis jointly with the Idiap Research Institute and EPFL and for supporting my research path. I am especially grateful to Sylvain for his permanent availability and for the great opportunities that he provided me with during the last four years. I also would like to thank my friends at Idiap for their valuable friendship, which always accompanied me during this journey. A special thanks goes to Sandrine, who started the PhD the same day as I did, for many shared times around a tea, for her understanding, and for her open mind. Another thanks goes to Rémy for his positiveness, for valuable conversations, and for his attentive ear. I would also like to thank the RLI group members, particularly Hakan, Emmanuel, Teguh, Antonio, Andras, Martin, Thibaut, and Clément for various and numerous interesting discussions in the office.

I am also thankful to Mathias Bürger, who hosted my six-months PhD sabbatical in the Bosch Center for Artificial Intelligence. I really appreciated this novel and valuable experience, both from the professional and the personal side. I also thank the R31 team for the welcoming and pleasant atmosphere that I enjoyed while working in BCAI.

I would like to extend my gratitude to the members of my thesis committee who evaluated this work: Prof. Frank Park, Dr. Jean-Baptiste Mouret, Prof. Nicolas Flammarion, and Prof. Auke Ijspeert. Their insightful comments, suggestions and compliments were highly valuable, encouraging, and greatly contributed to refine the final version of this thesis.

I am especially grateful to my family: Without them, I would not be the person that I became. A special thanks goes to my Dad for sharing with me his deep interest for science since very early on. I particularly remember building and controlling a lego robot with him during a Christmas of my childhood — maybe an early indication on the carrier that I chose. I am deeply grateful for his infinite patience and for always providing help without counting. I am especially thankful to my Mom for always encouraging my choices and for her endless support in all the aspects of my life. It is precious to know that someone cares for me as she always does. I would also like to thank my two first teammates, my sisters Noelle and Karine. I profoundly value our complicity, along with

our similarities and differences, and I deeply admire their ability and commitment to helping people every single day. My gratitude extends to my grandparents: to Rosemarie for her calm and gentle presence, and to Maria and Pierre for their wise advices and for many enriching conversations over a dinner.

I would like to thank Fjolla for her true and simple friendship. I deeply appreciate our mutual understanding and companionable conversations during which any topic may appear. My thanks also go to Sara for her unfailing friendship, her positiveness, and her ability to call me at the right time even though we did not talk for weeks.

I also thank my two hairy flatmates, my cats Darwin and Murphy. I always appreciate their friendly company and their commitment to lying on my desk during my work time at home.

Finally, my gratitude goes to my correlated variable, Leonel. This thesis would not look the same without our joint work and his mentorship. I am deeply grateful for all that I have learned from him, both at the professional and at the personal level. I am equally grateful for his unwavering support and love. His presence by my side makes bright days even brighter and dispels the clouds of harder moments.

Vevey, July 27, 2020
Noémie Jaquier

Abstract

Humans exhibit outstanding learning and adaptation capabilities while performing various types of manipulation tasks. When learning new skills, humans are able to extract important information by observing examples of a task and efficiently refine a priori knowledge to master new tasks. However, learning does not happen in isolation, and the planning and control abilities of humans also play a crucial role to properly execute manipulation tasks of varying complexity. In the last years, a lot of attention has been devoted to the problem of providing robots with close-to-human-level abilities. In this context, this thesis proposes to enhance robot learning and control capabilities by introducing domain knowledge into the corresponding models. Our approaches, built on Riemannian manifolds, exploit the geometry of non-Euclidean spaces, which are ubiquitous in robotics to represent rigid-body orientations, inertia matrices, manipulability ellipsoids, or controller gain matrices.

We initially consider the problem of transferring skills to a robot and propose a probabilistic framework to learn symmetric positive definite (SPD) matrices from demonstrations. Given a learned reference trajectory in the form of SPD matrices, the goal of the robot is to reproduce the task by tracking this sequence using appropriate controllers. This challenge is tackled in the second part of this thesis. We focus on a specific application and propose a complete geometry-aware framework to learn, control and transfer posture-dependent task requirements from humans to robots via manipulability profiles. The third part of this thesis focuses on refining the skills learned by the robot and on adapting them to new situations by introducing a geometry-aware Bayesian optimization framework. Overall, this thesis proves that geometry-awareness is crucial for successfully learning, controlling and refining non-Euclidean parameters in addition to providing a proper mathematical treatment of the different problems. Finally, the last part of this work shows that domain knowledge can be introduced not only through geometry-awareness, but also through structure-awareness, which may be considered either as prior models or as intrinsically present in the data. With its final part, this thesis emphasizes that domain knowledge, that can be introduced into robot learning algorithms through various means, may be significant for providing robots with close-to-humans abilities.

Keywords: Robot learning, Riemannian manifolds, Domain knowledge, Learning from demonstrations, Bayesian optimization, Manipulability ellipsoids, Tensor methods.

Résumé

Les humains exhibent d'impressionnantes capacités d'apprentissage et d'adaptation lorsqu'ils effectuent différentes tâches de manipulation. Lorsqu'ils apprennent de nouvelles compétences, les humains sont capables d'extraire les informations importantes en observant des exemples d'une tâche et d'affiner efficacement leurs connaissances a priori pour maîtriser de nouvelles tâches. Cependant, l'apprentissage ne se passe pas de manière isolée et les capacités de planification et de contrôle des humains jouent également un rôle crucial pour exécuter correctement des tâches de manipulation de complexité variable. Ces dernières années, une attention particulière a été dévolue au problème de pourvoir les robots de capacités proches de celles des humains. Dans ce contexte, cette thèse propose d'améliorer les capacités d'apprentissage et de contrôle des robots en introduisant des informations de domaine dans les modèles correspondants. Nos approches, construites à l'aide de variétés riemanniennes, exploitent la géométrie des espaces non euclidiens, omniprésents en robotique pour représenter l'orientation de corps rigides, les matrices d'inerties, les ellipses de manipulabilité ou les matrices de gains de contrôleurs.

Nous considérons initialement le problème consistant à transférer des compétences à un robot et proposons une approche probabiliste pour apprendre des matrices symétriques positive définies (SPD) par démonstrations. Étant donné une trajectoire de référence apprise sous la forme de matrices SPD, le but du robot est de reproduire la tâche en suivant cette séquence avec des contrôleurs appropriés. Ce défi est relevé dans la deuxième partie de cette thèse. Nous nous concentrons sur une application spécifique et proposons une approche complète tenant compte de la géométrie pour apprendre, contrôler et transférer des contraintes de tâches qui dépendent de la posture d'humains à des robots via des profils de manipulabilité. La troisième partie de cette thèse se concentre sur l'affinage de compétences apprises par le robot et sur l'adaptation de ces compétences pour de nouvelles situations en introduisant une méthode d'optimisation bayésienne tenant compte de la géometrie. En résumé, cette thèse montre que les méthodes tenant compte de la géométrie sont cruciales pour apprendre, contrôler et affiner fructueusement les paramètres non euclidiens, en plus de traiter les différents problèmes avec des outils mathématiques appropriés. Finalement, la dernière partie de ce travail montre que des informations de domaine peuvent être introduites non seulement à travers la géométrie, mais aussi à travers la structure, qui peut être considérée sous la forme de modèles a priori ou comme intrinsèquement présente dans les données. Avec cette dernière partie, cette thèse souligne que les informations de domaine, pouvant être introduites dans les algorithmes d'apprentissage de différentes manières, pourraient être significatives pour pourvoir les robots de capacités proches de celles des humains.

Mots-clés : Apprentissage robotique, Variétés riemanniennes, Informations de domaine, Apprentissage par démonstrations, Optimisation bayésienne, Ellipses de manipulabilité, Méthodes tensorielles.

Contents

Acknowledgements **i**

Abstract **iii**

Résumé **v**

List of Figures **xi**

List of Tables **xv**

1 Introduction **1**
- 1.1 Motivation 2
- 1.2 Thesis Objectives and Outline 5

2 Background **9**
- 2.1 Introduction 10
- 2.2 Riemannian Manifolds 10
  - 2.2.1 Riemannian Manifolds in Robotics 11
  - 2.2.2 Riemannian Manifolds Theory 12
  - 2.2.3 The Sphere Manifold 15
  - 2.2.4 The Symmetric Positive Definite Matrix Manifold 17
- 2.3 Tensor Operations 19
  - 2.3.1 Notations 20
  - 2.3.2 Product Operations 20
  - 2.3.3 Distribution of Symmetric Matrices and Related Operations . . . 23
  - 2.3.4 Matrix Derivatives 24
- 2.4 Learning Algorithms 26
  - 2.4.1 Gaussian Mixture Model in Learning from Demonstrations . . . 26
  - 2.4.2 Gaussian Processes 29
- 2.5 Conclusion 33

I Geometry-aware Robot Learning from Demonstrations **35**

3 Gaussian Mixture Model on the SPD Manifold **37**

3.1 Introduction 38
3.2 Background: Statistics on SPD Manifolds 39
3.2.1 Mean and Covariance of SPD Matrices 39
3.2.2 The Normal Distribution for SPD Matrices 40
3.3 Gaussian Mixture Model on SPD Manifolds 43
3.4 Gaussian Mixture Regression on SPD Manifolds 44
3.5 Importance of Geometry-awareness for Learning SPD Matrices 49
3.5.1 Importance of Geometry-awareness for Encoding SPD Matrices . 49
3.5.2 Importance of Geometry-awareness for Retrieving SPD Matrices 51
3.6 Conclusion 55

4 Learning Task-dependent Requirements via Manipulability Analysis 57
4.1 Introduction 58
4.2 Background: Manipulability Ellipsoids 60
4.3 Manipulability Analysis 61
4.3.1 Data Description 62
4.3.2 Human Manipulability Computation 62
4.3.3 Analysis 64
4.4 Learning Manipulability Ellipsoids 70
4.5 Conclusion 73

II Designing Geometry-aware Controllers 75

5 Tracking Manipulability Ellipsoids 77
5.1 Introduction 78
5.2 Tracking Manipulability Ellipsoids 79
5.2.1 Manipulability Jacobian 80
5.2.2 Geometry-aware Manipulability Tracking Formulation 81
5.2.3 Actuators Contribution 86
5.2.4 Exploiting 4th-order Precision Matrix as Controller Gain 88
5.2.5 Nullspace of the Manipulability Jacobian 91
5.3 Importance of Geometry-awareness for Manipulability Tracking 92
5.3.1 Comparisons with Euclidean Tracking 92
5.3.2 Comparisons with Manipulability-based Optimization 100
5.4 Experiments 102
5.4.1 Manipulability Tracking for a Robotic Arm 102
5.4.2 Manipulability Tracking for a Robotic Hand 103
5.4.3 Manipulability Tracking for a Humanoid Center of Mass 105
5.5 Conclusion 108

6 A Human-to-robot Manipulability Transfer Framework 111
6.1 Introduction 112

6.2 Manipulability Transfer . . . 113
6.3 Experiments . . . 114
    6.3.1 Human-to-robot Manipulability Transfer . . . 114
    6.3.2 Manipulability Transfer Between Robots for a Bimanual Task . . 117
6.4 Discussion . . . 123
6.5 Conclusion . . . 125

III Geometry-aware Policy Optimization 127

7 GaBO: Geometry-aware Bayesian Optimization 129
7.1 Introduction . . . 130
7.2 Related Work . . . 131
7.3 Background: Bayesian Optimization . . . 133
7.4 Bayesian Optimization on Riemannian Manifolds . . . 134
    7.4.1 Geometry-aware Kernels . . . 134
    7.4.2 Optimization of Acquisition Functions . . . 137
7.5 Experiments . . . 146
    7.5.1 Benchmark Functions . . . 146
    7.5.2 Simulated Robotic Experiments . . . 150
7.6 Conclusion . . . 154

8 Handling High-dimensional Problems with GaBO 155
8.1 Introduction . . . 156
8.2 High-dimensional Geometry-aware Bayesian Optimization . . . 158
    8.2.1 HD-GaBO Surrogate Model . . . 159
    8.2.2 Input Reconstruction from the Latent Embedding to the Original Space . . . 160
    8.2.3 Nested Manifolds Mappings . . . 161
8.3 Experiments . . . 165
    8.3.1 Optimization of High-dimensional Objectives with GaBO . . . 165
    8.3.2 Optimization of High-dimensional Objectives with HD-GaBO . . 169
8.4 Conclusion . . . 171

IV Leveraging Structure-awareness in Robot Learning 175

9 A Model-based Gaussian Process 177
9.1 Introduction . . . 178
9.2 Background: Multi-output Gaussian Processes . . . 179
9.3 A Model-based Gaussian Process . . . 180
    9.3.1 GMR-based GPs Formulation . . . 181
    9.3.2 GMR-based GPs Properties . . . 182
9.4 Experiments . . . 185

9.5 Discussion 189
9.6 Conclusion 191

10 Exploiting Tensor Structures in a Mixture of Experts 193
10.1 Introduction 194
10.2 Background: Linear Models For Tensors 196
  10.2.1 Tensor Rank and CANDECOMP/PARAFAC Decomposition . . 196
  10.2.2 Generalized Linear Model for Tensors 197
  10.2.3 Tensor Ridge Regression (TRR) 198
  10.2.4 Tensor Logistic Regression (TLR) 200
10.3 Tensor-variate Mixture of Experts 201
  10.3.1 TME Model 201
  10.3.2 Training of TME 202
  10.3.3 Model Selection and Initialization 204
10.4 Experiments 204
  10.4.1 2D Shape Example 204
  10.4.2 Shape Example of Higher Dimensions 207
  10.4.3 Detection of Hand Movements from Tactile Myography 209
  10.4.4 Real-time Teleoperation with Tactile Myography 212
10.5 Discussion 216
10.6 Conclusion 217

11 Conclusions and Future Work 219
11.1 Conclusions 220
11.2 Future Work 222

A Appendices to Chapter 4 225
A.1 Identification Method for the Kinematics of an Anthropomorphic Arm . 225

B Appendices to Chapter 5 229
B.1 Symbolic Manipulability Jacobian for a Serial Kinematic Chain 229
B.2 Symbolic Dynamic Manipulability Jacobian for a Serial Kinematic Chain 230
B.3 Symbolic Derivative of the Manipulability Jacobian for a Serial Kinematic Chain 230

C Appendices to Chapter 8 233
C.1 Distances between Points on Nested Spheres 233

Bibliography 235

Curriculum Vitae 253

List of Figures

1.1 Illustrations of two non-Euclidean parameter spaces arising in robotics 4
1.2 Examples of ill-conditioned behaviors arising while ignoring the intrinsic geometry of non-Euclidean parameter spaces 4
1.3 Thesis outline 6

2.1 Sphere manifold $\mathcal{S}^2$ (incl. e.g., orientations) 16
2.2 SPD manifold $\mathcal{S}^2_{++}$ (incl. e.g., stiffness ellipsoids) 18
2.3 Illustration of the tensor matricization operation 21
2.4 Illustration of GMM and GMR on a two-dimensional time-driven trajectory 30
2.5 Illustration of GPR on a two-dimensional time-driven trajectory 32

3.1 Illustration of the concepts of mean and covariance on the SPD manifold $\mathcal{S}^2_{++}$ 41
3.2 Illustration of the M-step of one iteration of the EM algorithm to fit a GMM model on the SPD manifold $\mathcal{S}^2_{++}$ 45
3.3 Illustration of GMR on the SPD manifold $\mathcal{S}^2_{++}$ 48
3.4 Importance of geometry in GMM formulation 51
3.5 Importance of geometry in the computation of the responsabilities of GMM components 52
3.6 Importance of geometry to retrieve SPD matrices via GMR 54

4.1 Illustration of pushing and pulling tasks for which the posture of the humans significantly influences their ability to carry out the task. . . . 58
4.2 Human arm triangle model 63
4.3 Posture of a participant during screwing actions at 3 different heights . 65
4.4 Temporal evolution of the single-arm velocity manipulability ellipsoid for the screwing tasks 66
4.5 Temporal evolution of the dual-arm force manipulability ellipsoid for the carrying task `C5` 68
4.6 Temporal evolution of the dual-arm force manipulability ellipsoid for the carrying task `C10` 69
4.7 Four demonstrations (Cartesian position and manipulability) of a 3-DoF planar robot tracking a C-shape trajectory 71
4.8 GMM encoding the four demonstrations of a C-shape trajectory 72

4.9 Desired execution (Cartesian position and manipulability) of a C-shape tracking task 72

5.1 Manipulability tracking as main task 82
5.2 Manipulability-based redundancy resolution with Cartesian position control 83
5.3 Illustration of actuators contribution 88
5.4 Manipulability tracking as main task with diagonal gain matrices defined from different precision tensors 90
5.5 Reproductions of a learned C-shape tracking task with desired manipulability ellipsoids 91
5.6 Use of the nullspace of the manipulability Jacobian 92
5.7 Performance of different manipulability tracking formulations 95
5.8 Comparison of the performance of different manipulability tracking formulations for different gains 96
5.9 Performance comparison of the different manipulability-based redundancy resolution formulations 98
5.10 Comparison of our manipulability tracking controller with manipulability-based optimization 101
5.11 Cartesian velocities achieved with our manipulability tracking controller and the volume maximization approach as secondary objective for a Cartesian velocity controller 102
5.12 Manipulability tracking for a pushing task with the Baxter robot . . . 104
5.13 Manipulability tracking for an insertion task with the Baxter robot . . 104
5.14 Manipulability tracking for grasping tasks with the Allegro hand in simulation 106
5.15 Tracking of the COM manipulability with the Centauro robot in simulation 107
5.16 Tracking of the COM manipulability with NAO in simulation 108

6.1 Overview of the proposed manipulability transfer framework 113
6.2 Partial reproduction of a screwing task using human-robot manipulability transfer 116
6.3 Snapshots of the robot executing a partial reproduction of a screwing task using human-robot manipulability transfer 117
6.4 Partial reproduction of a carrying task using human-robot manipulability transfer 118
6.5 Snapshots of the robot executing a partial reproduction of a carrying task using human-robot manipulability transfer 119
6.6 Snapshots of the bimanial unplugging task 120
6.7 Demonstrations and GMM encoding the bimanual unplugging task . . 121
6.8 Reproduction of the bimanual unplugging task with Baxter 122
6.9 Reproduction of the bimanual unplugging task with the two Panda robots 122

7.1 Experimental selection of $\beta_{\min}$ 136
7.2 Illustration of the effect of β on GaBO 137
7.3 Illustration of one iteration of the conjugate gradient algorithm on the sphere 140
7.4 Application of Algorithm 4 on $\mathcal{S}^1$ 142
7.5 Surrogate GP of GaBO for the minimization of the Ackley function in $\mathcal{S}^2$ 148
7.6 Logarithm of the simple regret in function of the BO iterations for the Ackley function in $\mathcal{S}^d$ 149
7.7 Logarithm of the simple regret for benchmark test functions in the Riemannian manifold of SPD matrices $\mathcal{S}^2_{++}$ 149
7.8 Illustrations of the end-effector orientation learning 151
7.9 Total cost for end-effector orientation learning in $\mathcal{S}^3$ 151
7.10 Illustration of the direct policy search experiment 152
7.11 Total cost for stiffness learning of a Cartesian impedance controller using direct policy search 153

8.1 Illustration of the low-dimensional assumption on Riemannian manifolds 158
8.2 Illustration of the nested sphere projection mapping 163
8.3 Logarithm of the simple regret for high-dimensional benchmark test functions in $\mathcal{S}^D$ 167
8.4 Logarithm of the simple regret for high-dimensional benchmark test functions in $\mathcal{S}^D_{++}$ 168
8.5 Logarithm of the simple regret for benchmark test functions in the nested sphere manifold $\mathcal{S}^5$ of $\mathcal{S}^50$ 170
8.6 Logarithm of the simple regret for benchmark test functions in the nested SPD manifold $\mathcal{S}^3_{++}$ of $\mathcal{S}^{10}_{++}$ 172

9.1 Illustration of GMR-based GP on a two-dimensional time-driven trajectory 182
9.2 Illustration of the properties of GMR-based GP on time-driven 1D-trajectories 184
9.3 Demonstrations of a peg-in-hole task are provided via kinesthetic teaching 185
9.4 Demonstrations and reproductions of the peg-in-hole task with GMR-based GP, MOGP and GMR 187
9.5 Snapshots of the robot reproducing the peg-in-hole task using GMR-based GP and MOGP 188
9.6 3D representation of the trajectories followed by the robot in the reproduction of the peg-in-hole task 189
9.7 Comparison of the predicted two-dimensional trajectories generated by ProMP, KMP and GMR-based GP with three via-points 191
9.8 Example of time-driven 1D-trajectory predicted by KMP with three via-points 191

10.1 TMG sensor used for the experiments 196

10.2 Illustrations of generalized linear models (GLM) 199
10.3 Proposed TME model 203
10.4 True and recovered weight matrices of the gate and experts functions by ME and TME 206
10.5 Recovered weight matrix by RR and MRR 206
10.6 Evolution of the RMSE of the estimation of TME weight matrices and recovered weight matrices in function of the sample size for different noise levels 207
10.7 Evolution of the RMSE in function of the dimension of the data for the estimation of the weight tensors by ME and TME 208
10.8 Data collection setup for the detection of hand movements from TMG 210
10.9 Original and recovered activations of the different fingers and wrist movements over time 211
10.10 Setup of the teleoperation experiment 214
10.11 Snapshots of the teleoperation experiment for different tasks 215

List of Tables

2.1 Principal operations on the sphere $\mathcal{S}^d$ and SPD manifold $\mathcal{S}^d_{++}$ 15
2.2 Tensor notations 20
2.3 Tensor-related product operations 21

5.1 Initial and final Riemannian affine-invariant distances $d_{\mathcal{S}^d_{++}}(\hat{\boldsymbol{M}}, \boldsymbol{M}_t)$ between the current and desired manipulability for the experiments illustrated in Fig. 5.1 and 5.2. 83
5.2 Final Riemannian affine-invariant distances $d_{\mathcal{S}^d_{++}}(\hat{\boldsymbol{M}}, \boldsymbol{M}_t)$ for different manipulability tracking formulations 94
5.3 Final Riemannian affine-invariant distances $d_{\mathcal{S}^d_{++}}(\hat{\boldsymbol{M}}, \boldsymbol{M}_t)$ for different manipulability-based redundancy resolution formulations 99

7.1 Selected values of $\beta_{\min}$ for different Riemannian manifolds used in this chapter 135
7.2 Computation time for one iteration of the different BO frameworks when optimizing the Ackley function on several manifolds 150
7.3 Cost parameter values for the orientation sampling experiment 151
7.4 Cost parameter values of f_1 in the optimal stiffness learning experiment 153
7.5 Cost parameter values of f_2 in the optimal stiffness learning experiment 153

9.1 Computation time of GMR, MOGP and GMR-based GP for one test data in the real robot experiment 189

10.1 Performance comparison in terms of RMSE between different regression methods to predict fingers and wrist movements from TMG data. . . . 211
10.2 Average testing computation time for the different regression methods 212
10.3 Success rates over all the tasks and for each object in the case where the participants teleoperated both the robot arm and hand 216
10.4 Proportion of the failed tasks due to running out of time during the grasp, the displacement and the release of the object 216
10.5 Success rates over all the tasks and for each object, for the case in which the commands of the arm had to be disabled 217

1 Introduction

1.1 Motivation

When we learn to execute a novel manipulation task, we naturally shape our behavior in function of our a-priori knowledge and experience of the given task. In this process, we shape our gestures to mimic these of people that we may have seen executing — successfully or not — similar tasks and integrate it to our own experience in manipulating different objects. As a matter of fact, we might first fail at executing the given task; in this case, we are able to update our understanding of the problem and our gestures in order to achieve our goal, usually after only few trials. However, successful learning would not be possible without efficient perception, planning and control abilities. Indeed, at each instant of a manipulation task, we naturally adapt the motion and strength of our arms and body to make them compatible with both the available information about the task and our instantaneous perception of the environment. To do so, our brain is able to deal with various external parameters, such as rigid-body positions and orientations, forces or objects inertias, and to trigger an appropriate reaction from our body through limb motions and force exertions. In that sense, our brain may be seen as a geometric machine, transforming sensory information into motor commands [Pellionisz and Llinás, 1985; Habas et al., 2020].

One of the main requirements for robots to be endowed with close-to-human abilities is the capability to learn new skills. To do so, inspiration has been taken from the ways human learn and various robot learning methods have been developed in the past years. In this context, the two main research directions in robot learning are *learning from demonstrations* (LfD), where the robot learns by observing and generalizing expert demonstrations [Billard et al., 2008; Ravichandar et al., 2020] and *reinforcement learning* (RL), where the robot learns through trial and error [Deisenroth et al., 2013; Sutton and Barto, 2018]. Similarly to humans, when learning a new skill, robots should be able to cope with a profusion of external information and learn to execute the appropriate motions to fulfill the task. As an example, for manipulation tasks, the robot learns how to position and orient its end-effector in order to manipulate different objects. Moreover, it may need to learn to apply certain forces while interacting with the environment and to adapt the stiffness of its joints to be resistant to perturbations or, in the contrary, to be compliant to guarantee safe interactions with the user [Rozo et al., 2016; Silvério et al., 2018]. In other cases, the robot should be able to place not only its end-effector, but also its other joints, to avoid collisions in a cluttered environment [Rana et al., 2017a], or to adapt the optimal directions in which to generate a motion or apply a specific force, e.g., in a planning process [Lee and Oh, 2016; Gu et al., 2015]. Therefore, a variety of manifolds arise during the process of a robot learning new manipulation skills; positions, forces and torques belong to the Euclidean manifold $\mathbb{R}^d$, stiffness, inertia, manipulability and controller gain matrices lie in the manifold of symmetric positive definite (SPD) matrices $\mathcal{S}^d_{++}$, the special orthogonal group $SO(3)$ or the unit sphere $\mathcal{S}^3$ are used to represent orientations, and the special Euclidean group $SE(3)$ describes robot poses.

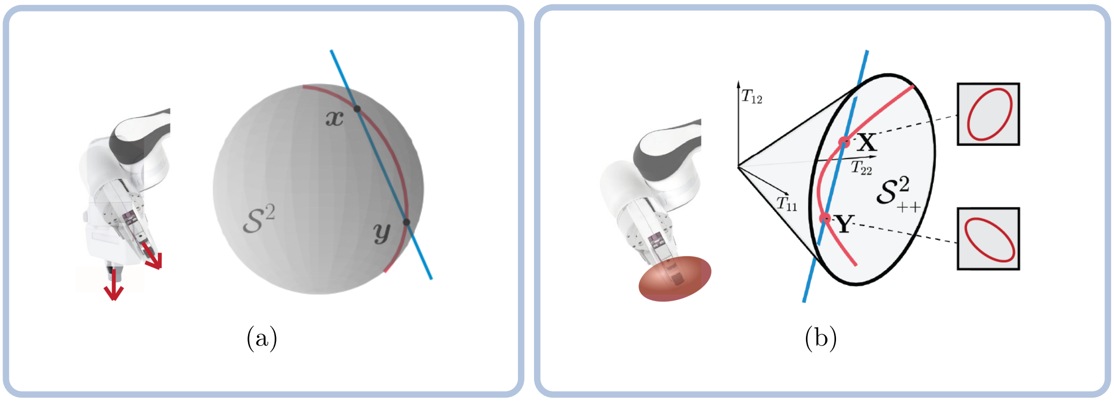
In this context, training data — collected during the demonstration phase in LfD and arising from previous trials in RL — may belong to one or several of these manifolds. While Euclidean data are automatically encoded and retrieved by classical learning algorithms, special care must be taken with non-Euclidean variables. Indeed:

1. The shortest paths between two points does not usually follow a straight line in curved spaces, therefore modifying the relationships between data compared to these in Euclidean spaces — for example, the distance between two points on a ball is naturally measured along its surface instead of through the ball;
2. The outputs retrieved by a learning algorithm must be restricted to the set of points belonging to the corresponding manifold.

Figure 1.1 illustrates these concepts for the unit-sphere and SPD manifolds. For both manifolds, the Euclidean path linking two points $\boldsymbol{x}$ and $\boldsymbol{y}$ is represented with a blue line. In the case of the sphere, this path intersects with the sphere surface only at $\boldsymbol{x}$ and $\boldsymbol{y}$ and is therefore not suited to measure the distance between or retrieve data based on these two points, e.g., via interpolation or extrapolation. The situation is slightly different in the case of the SPD manifold, as the Euclidean blue line remains in the manifold from $\boldsymbol{X}$ to $\boldsymbol{Y}$. Therefore, it could potentially be used to measure distances between elements on the manifold. However, as illustrated in Figure 1.1, straight lines are generally not embedded in the manifold, therefore leading to invalid outputs if these paths are used to extrapolate new points from known data. In contrast, the paths obtained by taking the intrinsic *geometry* of these non-Euclidean spaces into account, depicted as red curves, are suited both to measure distances and to retrieve data.

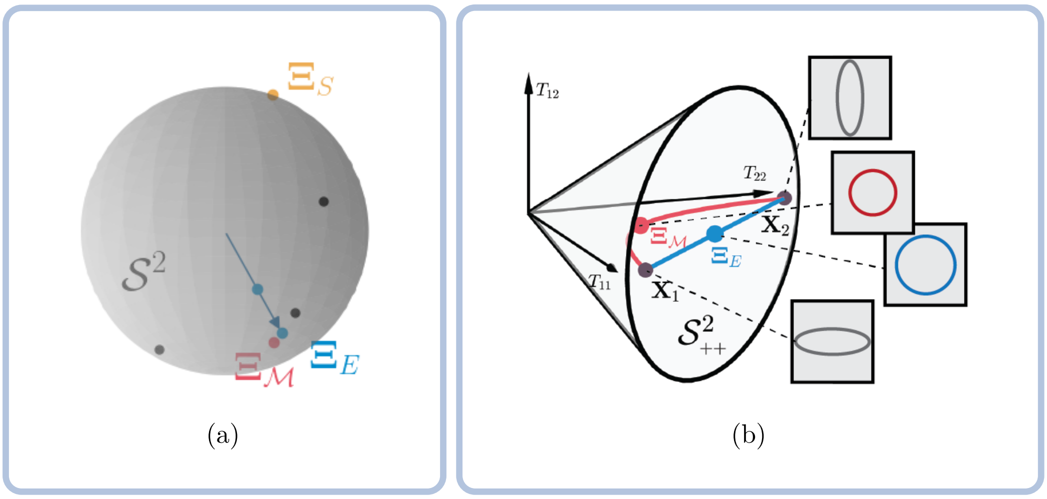
Treating data that do not lie on a vector space with Euclidean methods leads not only to incorrect paths that may not belong to the manifold, but also to ill-conditioned behaviors in many situations. As a first simple example, we consider the computation of the mean of three points on the unit sphere, as illustrated in Figure 1.2a. The Euclidean mean, represented by a blue dot, does not lie on the unit-sphere and therefore must be projected to the manifold as a post-processing step. Importantly, we observe that the resulting projection does not correspond to the intrinsic mean $\Xi_{\mathcal{M}}$, depicted by a red dot, obtained by considering the intrinsic geometry of the data. Moreover, the obtained Euclidean mean depends on the coordinate system used to represent the data: the Cartesian mean Ξ_{E} is not equivalent to the mean Ξ_{S} obtained with spherical coordinates (yellow dot), which lies outside the support of the data. In order to avoid this problem, we aim at developing *coordinate-invariant* algorithms, which may naturally be obtained by considering the intrinsic geometry of non-Euclidean variables [Biess et al., 2011; Lee and Park, 2018; Park et al., 2018]. As a second example, we consider the mean of two SPD matrices

$$\boldsymbol{X}_1 = \begin{pmatrix} 18 & 0 \\ 0 & 2 \end{pmatrix} \text{ and } \boldsymbol{X}_2 = \begin{pmatrix} 2 & 0 \\ 0 & 18 \end{pmatrix},$$



(a) (b)

Figure 1.1 – Illustrations of two non-Euclidean parameter spaces arising in robotics, namely (*a*) the unit sphere, used to represent orientations, and (*b*) the SPD manifold, used to represent, among others, stiffness matrices. (*a*) Unit vectors correspond to points in the surface of the sphere. (*b*) Each point in the graph represents a symmetric matrix $\begin{pmatrix} T_{11} & T_{12} \\ T_{12} & T_{22} \end{pmatrix}$. SPD matrices corresponds to the points inside the cone. For both spaces, the Euclidean shortest path passing through $\boldsymbol{x}$ and $\boldsymbol{y}$ is represented as a blue line. Note that, unlike the geometry-aware path (depicted in red), this line is not embedded in the manifold.



(a) (b)

Figure 1.2 – Examples of ill-conditioned behaviors arising while ignoring the intrinsic geometry of non-Euclidean parameter spaces. (*a*) Computation of the mean of three points (black dots) on the unit sphere $\mathcal{S}^2$. The Euclidean mean depends on the choice of coordinate system: the Euclidean mean based on Cartesian coordinates Ξ_{E} (blue dot) does not correspond to the Euclidean mean in spherical coordinates Ξ_{S} (yellow dot). In contrast, the intrinsic mean $\Xi_{\mathcal{M}}$, depicted with a red dot, takes the geometry of the data into account and therefore is coordinate invariant. (*b*) Interpolation between two SPD matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, whose volume is identical. The geometry-aware path, depicted by a red curve, consistently preserves the volume of the ellipsoids. In contrast, this volume is considerably increased while following the Euclidean path, represented by a blue line.

whose volume is identical, i.e., $\det(\boldsymbol{X}_1) = \det(\boldsymbol{X}_2) = 36$. As shown in Figure 1.2b, the intrinsic mean $\Xi_{\mathcal{M}} = \left(\begin{smallmatrix} 6 & 0 \\ 0 & 6 \end{smallmatrix}\right)$ differs from the Euclidean mean $\Xi_{\text{E}} = \left(\begin{smallmatrix} 10 & 0 \\ 0 & 10 \end{smallmatrix}\right)$. Moreover, while the geometry-aware path, including the intrinsic mean, consistently preserves the determinant, this is not conserved along the Euclidean path. Consequently, the volume of the Euclidean mean, proportional to $\det(\Xi_{\text{E}}) = 100$, is considerably varied compared to that of the data. In robotics, it is particularly important that the volume of SPD data evolves in a consistent manner, e.g., when designing stable controllers. Therefore, in order to properly learn and control non-Euclidean variables independently of the choice of coordinates, the intrinsic *geometry* of these non-Euclidean spaces must be taken into account.

Many robotics parameters lie on non-Euclidean spaces and the particular geometry of these spaces is often known. The special orthogonal group SO(3) and the special Euclidean group SE(3) are well-studied instances of *Lie groups* [Park, 1995; Selig, 2005; Solà et al., 2018], while the unit-sphere and SPD manifolds can be endowed with a Riemannian metric to form instances of *Riemannian manifolds* [do Carmo, 1992; Lee, 2012; Pennec et al., 2019]. In this thesis, we focus on robotic parameters lying on Riemannian manifolds and propose to take their geometry into account via the introduction of *domain knowledge* into robot learning and control algorithms. To do so, we exploit Riemannian manifolds theory and tensor representations to build geometry-aware learning, control and optimization schemes. The main contributions of this thesis are the followings: *(i)* we provide a detailed analysis of human arm motion patterns in industry-like activities encoded via manipulability ellipsoids; *(ii)* we propose a geometry-aware probabilistic framework to learn SPD matrices from demonstrations; *(iii)* we consider the problem of tracking manipulability ellipsoids from a control perspective and develop geometry-aware manipulability tracking controllers; *(iv)* we combine the aforementioned learning and tracking approaches into a complete human-to-robot manipulability transfer framework; *(v)* we introduce a novel geometry-aware Bayesian optimization (GaBO) framework for refining and optimizing parameters lying on Riemannian manifolds; *(vi)* we scale GaBO towards high-dimensional structured parameter spaces; *(vii)* we introduce a novel model-based Gaussian process, where domain knowledge is incorporated in the form of a prior model; *(viii)* we propose a tensor-variate-mixture-of-experts model for regression, which takes the intrinsic structure of the data into account. Finally, we demonstrate throughout the thesis that levering geometry-awareness into robot learning, control and optimization may not only bring mathematical correctness but greatly benefit the performances of different approaches in terms of accuracy, stability, convergence and scalability.

1.2 Thesis Objectives and Outline

This thesis is organized in four main parts. Each part is composed of two chapters aiming at solving a specific challenge. The objective of each part is described below, along with a

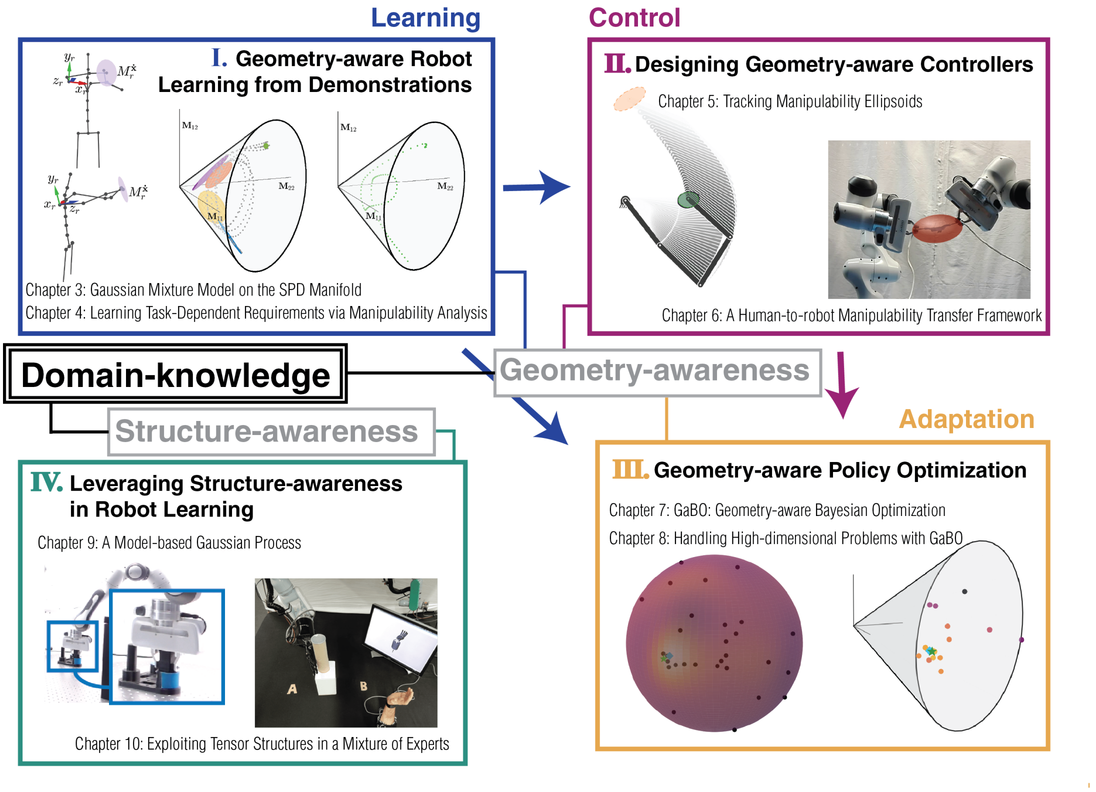

Learning
Control
I. Geometry-aware Robot Learning from Demonstrations
y_r
z_r
x_r
$M_r^{\dot{x}}$
M12
M22
M11
Chapter 3: Gaussian Mixture Model on the SPD Manifold
Chapter 4: Learning Task-Dependent Requirements via Manipulability Analysis
II. Designing Geometry-aware Controllers
Chapter 5: Tracking Manipulability Ellipsoids
Chapter 6: A Human-to-robot Manipulability Transfer Framework
Domain-knowledge
Geometry-awareness
Structure-awareness
Adaptation
IV. Leveraging Structure-awareness in Robot Learning
Chapter 9: A Model-based Gaussian Process
A
B
Chapter 10: Exploiting Tensor Structures in a Mixture of Experts
III. Geometry-aware Policy Optimization
Chapter 7: GaBO: Geometry-aware Bayesian Optimization
Chapter 8: Handling High-dimensional Problems with GaBO

Figure 1.3 – Thesis outline.

brief overview of the content of each chapter. The outline of the thesis is also highlighted in Figure 1.3.

Part I: Geometry-aware Learning from Demonstrations

After introducing the main mathematical tools, methods and algorithms that will be used in the core of this thesis (Chapter 2), we first consider the problem of transferring skills to a robot based on experts demonstrations. In particular, we aim at complementing existing approaches to encode position, force and orientation data, and propose a probabilistic framework to learn SPD matrices from demonstrations in Chapter 3. To do so, we exploit statistical tools adapted to Riemannian manifolds and tensor methods and encode demonstrated sequences of SPD matrices with a Gaussian mixture model (GMM). A distribution of SPD trajectories can then be retrieved via conditioning of the geometry-aware GMM. We then focus on a specific application in Chapter 4. More precisely, based on the observation that humans are able to plan their limbs motion in anticipation of the execution of specific skills, we propose to analyze human arms motion patterns in industrial activities. To do so, we build our analysis on the so-called manipulability ellipsoid, which captures a posture-dependent ability to perform motion and exert forces along different task directions. Finally, we show how manipulability patterns in the form of SPD matrices can be learned by robots based on the probabilistic model presented in Chapter 3.

Part II: Designing Geometry-aware Controllers

Given a learned reference trajectory in the form of SPD matrices, the goal of the robot is to reproduce the task by tracking this sequence using appropriate controllers. Therefore, the objective of the second part of this thesis is to endow robots with controllers adapted to the tracking of SPD matrices. Chapter 5 presents a geometry-aware tracking control scheme in which the robot is requested to follow a desired profile of manipulability ellipsoids, either as its main task or as a secondary objective. By coupling the aforementioned controllers with the learning scheme presented in the first part of this thesis, a complete geometry-aware framework to learn, control and transfer posture-dependent task requirements from human to robots via manipulability profiles is introduced in Chapter 6.

Part III: Geometry-aware Policy Optimization

After learning a skill from demonstrations, it is often desirable that the robot is able to refine or adapt this skill to a new situation. In the third part of this thesis, we aim at developing an approach to refine, adapt and optimize parameters lying on Riemannian manifolds. To do so, we propose to utilize Bayesian optimization (BO), an approach that recently became popular in robotics to optimize control parameters and parametric policies in direct reinforcement learning due to its data efficiency and gradient-free approach. However, its performance may be seriously compromised when the parameter space is of high dimension. In Chapter 7, we hypothesize that bringing geometry-awareness into BO may improve its performance and scalability. We introduce the geometry-aware BO (GaBO) framework, that incorporates domain knowledge into BO via geometry-awareness. Following this line, we further extend GaBO to handle high-dimensional Riemannian manifold spaces in Chapter 8.

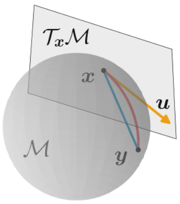
Part IV: Leveraging Structure-awareness in Robot Learning

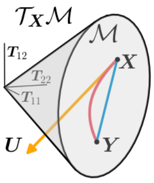
The objective of the last part of this thesis is to investigate the concept of bringing domain knowledge into learning algorithms from a broader perspective. We hypothesize here that domain knowledge can benefit robot learning by introducing not only geometry-awareness, but structure-awareness into the algorithms. In Chapter 9, structure-awareness is viewed through the incorporation of prior models into a non-parametric learning approach. More specifically, we propose a novel multi-output Gaussian process (GP) based on a GMM model and show that structure-awareness allows the proposed algorithm to outperform its individual components. Finally, Chapter 10 considers problems where an intrinsic structure exists within the data. Namely, we introduce a mixture-of-experts model handling high-dimensional data in the form of matrices and tensors. The proposed algorithm is applied in the context of prosthetics hands to recover hand motions from myographic signals.

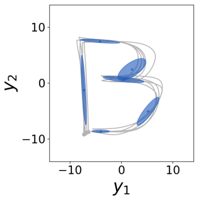
The importance and benefits of geometry-awareness in our approaches are highlighted throughout the three first part of this thesis. We prove that geometry-awareness is crucial for successfully learning, controlling and refining non-Euclidean parameters in addition to providing a proper mathematical treatment of the different problems tackled in this thesis. Overall, this thesis shows that including domain knowledge via geometry-awareness and structure-awareness contribute to enhance robot learning capabilities in various applications. The publications, supplementary material and source codes related to the content presented in this thesis are emphasized at the beginning of each chapter.

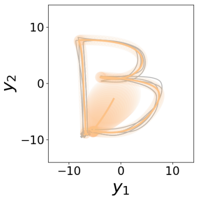
2 Background

The aim of this chapter is to introduce the mathematical tools, methods and algorithms that will be used to form the models and solutions to the problems tackled in this thesis. Riemannian manifolds, tensor operations and two learning algorithms are introduced. Following this background chapter, the Chapters 3-10, constituting the core of this thesis, will present approaches for robots to learn and refine skills while exploiting domain knowledge via geometry- and structure-awareness.









2.1 Introduction

We start this thesis by introducing the required mathematical tools for enhancing learning, control and optimization algorithms with geometry- and structure-awareness. As we aim at designing approaches that can efficiently handle non-Euclidean parameters such as rigid-body orientations, inertia matrices, manipulability ellipsoids, or controller gain matrices, we first introduce the basic theory on *Riemannian manifolds*. In order to handle multidimensional data, a brief introduction on *tensor* representations and operations is presented next. Finally, the two main learning algorithms used throughout this thesis are described. Each of these topics has been widely covered in the literature. Only brief overviews are provided in this chapter, as we aim here at providing the intuition and the necessary mathematical tools for the applications covered in this thesis. More in-depth references are provided for each topic in the related sections.

For the reader interested in specific chapters of the thesis, the background sections related to each chapter are highlighted below.

Chapter	**Background sections**
3: Gaussian Mixture Model on the SPD manifold	2.2, 2.3.2, 2.3.3, 2.4.1
4: Learning Task-dependent Requirements via Manipulability Analysis	2.2, 2.3.2, 2.3.3, 2.4.1
5: Tracking Manipulability Ellipsoids	2.2, 2.3.2, 2.3.4
6: A Human-to-robot Manipulability Transfer Framework	2.2
7: GaBO: Geometry-aware Bayesian Optimization	2.2, 2.4.2
8: Handling High-dimensional Problems with GaBO	2.2, 2.4.2
9: A Model-Based Gaussian Process	2.4.1, 2.4.2
10: Exploiting Tensor Structures in a Mixture of Experts	2.3.2

2.2 Riemannian Manifolds

In robotics, diverse types of data do not belong to a vector space and thus the use of classical Euclidean space methods for treating and analyzing these variables is inadequate. A common example is the unit quaternion, widely used to represent orientations. The quaternion has unit norm and therefore can be represented as a point on the surface of a 3-sphere. More generally, many data are normalized in a preprocessing step to discard superfluous scaling and hence are better explained through spherical representations [Fisher et al., 1987]. Symmetric positive definite (SPD) matrices are also widely used in robotics in the form of stiffness and inertia matrices, manipulability ellipsoids, or controller gain matrices. In general, SPD matrices are also widely encountered in data science: They coincide with the covariance matrices of multivariate distributions and are used as descriptors in many applications, such as computer vision [Tuzel et al., 2006] and brain-computer interface classification [Barachant et al., 2012]. For a mathematically consistent treatment of their particular geometry, both the sphere and the space of SPD matrices can be endowed with a Riemannian metric to form Riemannian manifolds.

A very simplistic way to deal with manifold-valued data is to consider them as if they were Euclidean and to apply classical algorithms augmented with a post-processing step to ensure the validity of the outcomes. However, this may lead to inconsistent behaviors, and to poor performance in practice. After a brief overview of the use of manifolds in robotics, the mathematical tools allowing a sound treatment of data lying on Riemannian manifolds are introduced in this chapter. The benefits of considering the intrinsic geometry of these data will be further demonstrated throughout this thesis. For thorough introductions to differential geometry and Riemannian manifolds, we refer the interested reader to the following books [do Carmo, 1992; Jost, 2008; Lee, 2012; Pennec et al., 2019].

2.2.1 Riemannian Manifolds in Robotics

Lie group and Riemannian manifold theories have been successfully applied to robotics from early on, for example in robot design [Park, 1995] or robot kinematics and dynamics analysis [Selig, 2005]. However, they only gained interest recently in robot learning, control and optimization. In particular, the usage of Lie group theory for motion or state estimation in robotics significantly increased in the last years. In this context, geometric methods improved the performance of estimation algorithms in terms of precision, stability and consistency, e.g., in the fields of visual odometry [Forster et al., 2017], Kalman filtering and simultaneous localization and mapping (SLAM) [Chirikjian and Kobilarov, 2014; Barfoot and Furgale, 2014], among others.

Riemannian manifold theory has also been exploited in robotic applications for motion and path planning. Notably, Zucker et al. [2013] introduced the covariant Hamiltonian optimization for motion planning (CHOMP) algorithm, which exploits the invariance properties of Riemannian manifolds to efficiently plan high-quality trajectories in cluttered environments. Along a similar line, Ratliff [2013] proposed to leverage the underlying geometry of the environment for planning and control applications by learning the Riemannian metric characterizing the configuration space. As another example, Bhattacharya et al. [2014] introduced a control law for the multi-robot coverage problem on Riemannian manifolds with boundaries, which can also be applied to coverage problems in unknown environments.

In the context of robot motion generation, Ratliff et al. [2018] introduced Riemannian motion policies (RMPs), where a robot motion policy is associated with a Riemannian metric that defines its local geometry. RMPs describe a virtual dynamical system, whose inertia correspond to the metric tensor, under the influence of a damped virtual potential field. Following this work, Rana et al. [2019] proposed an approach to learn RMPs from human demonstrations, where the potential field and Riemannian metric are estimated from the demonstrated robot motions. Notice that this approach is deterministic, so that the learned trajectories do not reflect neither the aleatoric, nor the epistemic uncertainty of the demonstrations.

Another line of research consists of exploiting the intrinsic geometry of parameters lying on Riemannian manifolds, whose geometry is known *a priori*. In this thesis, we follow this research path and propose to take into account the geometry of robotics parameters lying on Riemannian manifolds in robot learning, optimization and control. In particular, the work of Zeestraten [2018] is closely related to this line, as the author proposed a learning-from-demonstrations framework based on Riemannian manifold theory to encode and retrieve the orientation of the robot end-effector, represented as a unit quaternion.

2.2.2 Riemannian Manifolds Theory

A d-dimensional manifold $\mathcal{M}$ is a topological space which is locally Euclidean, which means that each point in $\mathcal{M}$ has a neighborhood which is homeomorphic to an open subset of the d-dimensional Euclidean space $\mathbb{R}^d$. A Riemannian manifold $\mathcal{M}$ is a differentiable manifold equipped with a Riemannian metric. For each point $\boldsymbol{x} \in \mathcal{M}$, there exists a tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ which is formed by the tangent vectors to all 1-dimensional curves on $\mathcal{M}$ passing through $\boldsymbol{x}$. The origin of the tangent space coincides with $\boldsymbol{x}$. The Riemannian metric is a smoothly-varying positive-definite inner product $\langle \cdot, \cdot \rangle_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \rightarrow \mathcal{M}$ acting on $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. The norm of a vector $\boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ is denoted by $\|\boldsymbol{u}\|_{\boldsymbol{x}}^2 = \langle \boldsymbol{u}, \boldsymbol{u} \rangle_{\boldsymbol{x}}$. Given a choice of local coordinates, a metric is represented by a SPD matrix $\boldsymbol{G}(\boldsymbol{x})$, called a metric tensor, whose coefficients depend smoothly on $\boldsymbol{x}$. Therefore, the inner product of two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ can equivalently be written as

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{x}} = \langle \boldsymbol{u}, \boldsymbol{G}(\boldsymbol{x})\boldsymbol{v} \rangle. \tag{2.1}$$

Curve length and geodesics

The Riemannian metric leads to local expressions of distances and angles between curves. Therefore, this characterization of the manifold can further be exploited to compute the length of curves on $\mathcal{M}$. Specifically, the length l of a curve $s(t)$ on a Riemannian manifold is given, following the classical arc-length formulation, as

$$l\left(s(t)\right) = \int_0^1 \|\dot{s}(t)\|_{s(t)} dt. \tag{2.2}$$

The Riemannian distance between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$ is defined as the minimum length over all possible smooth curves on the manifold between $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e.,

$$d_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y}) = \min_{s(t)} l\left(s(t)\right). \tag{2.3}$$

The corresponding curve is called a *geodesic*. Geodesics are the generalization of straight lines to Riemannian manifolds, as they are locally length-minimizing curves with constant speed in $\mathcal{M}$. These definitions are illustrated in Figure 2.1b, where the sphere manifold $\mathcal{S}^2 = \{\boldsymbol{x} \in \mathbb{R}^3 : \boldsymbol{x}^\mathsf{T}\boldsymbol{x} = 1\}$ is represented in its embedding space $\mathbb{R}^3$. The manifold $\mathcal{S}^2$ is composed of all the points lying on the surface of the 2-sphere. The tangent space $\mathcal{T}_x\mathcal{S}^2$ is the 2-plane tangent to the sphere at $\boldsymbol{x}$. The shortest path between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ on the sphere is the geodesic represented as a red curve. Note that it differs from the Euclidean shortest path, depicted in blue, that does not lie on the manifold.

Coordinate invariance

Importantly, the geometric constructions defined via the Riemannian metric on a Riemannian manifold — e.g., the length of a curve, the angle between a pair of curves, the area of a piece of surface, or the geodesics — do not depend on the choice of local coordinates and thus can be written in a coordinate-invariant form. This can be explained by the transformation behavior of the Riemannian metric under coordinate changes. Namely, the Riemannian metric encompasses the transformation rules between different sets of coordinates. To illustrate this point, we compute the coordinate expression for the Euclidean metric on $\mathbb{R}^2$ in polar coordinates. Given a curve $(x_1(t), x_2(t))$, the incremental arc length corresponds to $dx_1^2 + dx_2^2$. By substituting $x_1 = r\cos(\theta)$ and $x_2 = r\sin(\theta)$, we obtain

$$\begin{aligned}
dx_1^2 + dx_2^2 &= d(r\cos(\theta))^2 + d(r\sin(\theta))^2 \\
&= (\cos(\theta)dr - r\sin(\theta)d\theta)^2 + (\sin(\theta)dr + r\cos(\theta)d\theta)^2 \\
&= dr^2 + r^2 d\theta^2.
\end{aligned}$$

So, the Riemannian metric tensor is expressed as $\boldsymbol{G}(\boldsymbol{x}) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$ in Euclidean coordinates and as $\boldsymbol{G}(\boldsymbol{x}) = \left(\begin{smallmatrix} 1 & 0 \\ 0 & r^2 \end{smallmatrix}\right)$ in polar coordinates. Therefore, the Riemannian formulation allows us to develop coordinate-invariant algorithms whose performance is not influenced by a specific choice of coordinates.

Exponential map, Logarithm map and parallel transport

A Riemannian manifold is not a vector space since it is not closed under addition and scalar product [Pennec et al., 2006], and thus the use of classical Euclidean space methods for treating and analyzing data lying on this manifold is inadequate. However, due to their Euclidean geometry, linear algebra operations can be performed on the elements of each tangent space of a manifold. Therefore, these tangent spaces can be used not only to perform operations and interpolate between manifold-valued data, but to compute statistics on Riemannian manifolds. They are thus essential to generalize algorithms working on Euclidean space to Riemannian manifolds. To utilize these Euclidean tangent spaces, we need mappings back and forth between $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ and $\mathcal{M}$, which are known as exponential and logarithmic maps. The exponential map $\text{Exp}_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$ maps a point $\boldsymbol{u}$ in the tangent space of $\boldsymbol{x}$ to a point $\boldsymbol{y}$ on the manifold, so that it is reached at time 1 by the geodesic starting at $\boldsymbol{x}$ in the direction $\boldsymbol{u}$, i.e., $d_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{u}\|_{\boldsymbol{x}}$. The inverse operation is called the logarithmic map $\text{Log}_{\boldsymbol{x}} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. The exponential and logarithmic maps are illustrated for the manifold $\mathcal{S}^2$ in Figure 2.2b.

When working with data on Riemannian manifolds, we may need to compare or combine vectors lying on different tangent spaces. For example, in the conjugate gradient descent algorithm on Riemannian manifold exploited in Chapter 7, the search direction is updated by adding the gradient of the function to the previous search direction. However, these two vectors lie on two different tangent spaces, namely on the tangent spaces of the current and of the previous minimum estimates, while they must belong to the same space to be properly combined. Therefore, another useful operation over manifolds is the parallel transport $\Gamma_{\boldsymbol{x}\to\boldsymbol{y}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{T}_{\boldsymbol{y}}\mathcal{M}$, which moves elements between tangent spaces such that the inner product between two elements in the tangent space remains constant, i.e., $\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle_{\boldsymbol{x}} = \langle \Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}_1), \Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}_2) \rangle_{\boldsymbol{y}}$. This implies that the angle between a vector in the tangent space and the direction of the geodesic connecting $\boldsymbol{x}$ to $\boldsymbol{y}$ is also conserved. The notion of parallel transport is illustrated in Figure 2.2c.

Product of Riemannian manifolds

Importantly, the Cartesian product of two Riemannian manifolds corresponds to a Riemannian manifold. Specifically, the product $\mathcal{M}_1 \times \mathcal{M}_2$ of the manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$ is composed of the set of pairs $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, where $\boldsymbol{x}_1 \in \mathcal{M}_1$ and $\boldsymbol{x}_2 \in \mathcal{M}_2$. This property allows us to jointly treat data belonging to different Riemannian manifolds. In particular, it can be exploited to define joint distributions on any combination of manifolds. For example, the pose of a robot is often represented by a position and a quaternion. Therefore, we may define joint distribution of poses on the product of manifolds $\mathbb{R}^3 \times \mathcal{S}^3$. Notice that the Riemannian metric of a product of manifolds is built from the Riemannian metrics of the individual manifolds, so that the metric tensor corresponds to the block diagonal matrix

$$\boldsymbol{G}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \begin{pmatrix} \boldsymbol{G}(\boldsymbol{x}_1) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}(\boldsymbol{x}_2) \end{pmatrix}. \tag{2.4}$$

Table 2.1 – Principal operations on the sphere $\mathcal{S}^d$ and SPD manifold $\mathcal{S}_{++}^d$ (see [Absil et al., 2007b; Pennec et al., 2006; Sra and Hosseini, 2015] for details).

$\mathcal{M}$	$d_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y})$	$\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u})$	$\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y})$
$\mathcal{S}^d$	$\arccos(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y})$	$\boldsymbol{x}\cos(\lVert\boldsymbol{u}\rVert) + \frac{\boldsymbol{u}}{\lVert\boldsymbol{u}\rVert}\sin(\lVert\boldsymbol{u}\rVert)$	$d_{\mathcal{S}^d}(\boldsymbol{x}, \boldsymbol{y})\,\frac{\boldsymbol{y}-\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}\,\boldsymbol{x}}{\lVert\boldsymbol{y}-\boldsymbol{x}^{\mathsf{T}}\boldsymbol{y}\,\boldsymbol{x}\rVert}$
$\mathcal{S}_{++}^d$	$\lVert \log(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{Y}\boldsymbol{X}^{-\frac{1}{2}})\rVert_{\mathrm{F}}$	$\boldsymbol{X}^{\frac{1}{2}}\exp(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{U}\boldsymbol{X}^{-\frac{1}{2}})\boldsymbol{X}^{\frac{1}{2}}$	$\boldsymbol{X}^{\frac{1}{2}}\log(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{Y}\boldsymbol{X}^{-\frac{1}{2}})\boldsymbol{X}^{\frac{1}{2}}$
	Parallel transport $\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v})$		
$\mathcal{S}^d$	$\Big(-\boldsymbol{x}\sin(\lVert\boldsymbol{u}\rVert)\overline{\boldsymbol{u}}^{\mathsf{T}} + \overline{\boldsymbol{u}}\cos(\lVert\boldsymbol{u}\rVert)\overline{\boldsymbol{u}}^{\mathsf{T}} + (\boldsymbol{I} - \overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}^{\mathsf{T}})\Big)\boldsymbol{v}$ with $\overline{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\lVert\boldsymbol{u}\rVert}$		
$\mathcal{S}_{++}^d$	$\boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}}\ \boldsymbol{V}\ \boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}}^{\mathsf{T}}$ with $\boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}} = \boldsymbol{Y}^{\frac{1}{2}}\boldsymbol{X}^{-\frac{1}{2}}$		

Therefore, the exponential map, logarithmic map and parallel transport of a product of manifolds are obtained by combining the corresponding functions of the individual manifolds, e.g.,

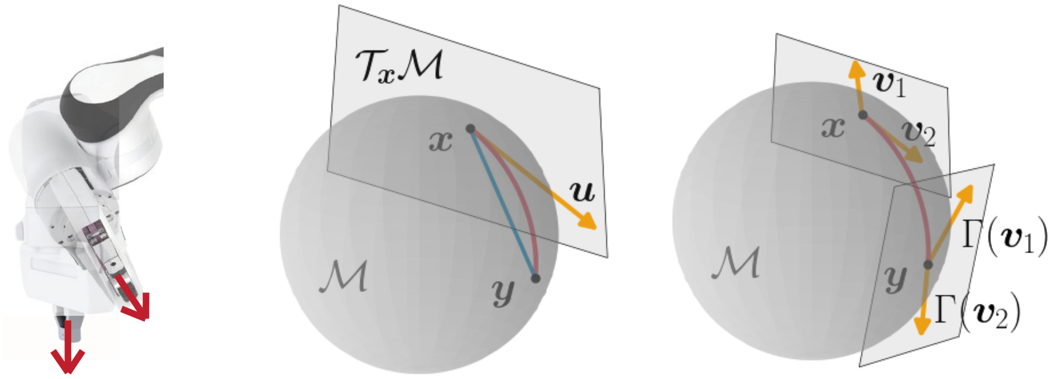
$$\mathrm{Log}_{(\boldsymbol{x}_1,\boldsymbol{x}_2)}\Big((\boldsymbol{y}_1, \boldsymbol{y}_2)\Big) = \Big(\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{y}_1), \mathrm{Log}_{\boldsymbol{x}_2}(\boldsymbol{y}_2)\Big), \tag{2.5}$$

with $\boldsymbol{x}_1, \boldsymbol{y}_1 \in \mathcal{M}_1$ and $\boldsymbol{x}_2, \boldsymbol{y}_2 \in \mathcal{M}_2$.

In this thesis, we consider problems involving two manifolds widely used in robotics, namely the sphere (Chapters 7-8) and SPD (Chapters 3-8) manifolds. These manifolds are described hereafter and the foregoing operations are summarized in Table 2.1.

2.2.3 The Sphere Manifold

The simplest and widely used example of a Riemannian manifold is the sphere $\mathcal{S}^2$ embedded in $\mathbb{R}^3$. This manifold can be easily represented and intuitively understood as it constitutes an idealized model of the surface of the earth. In general, directional data are commonly represented by unit vectors and are thus naturally analyzed as data belonging to hyperspheres. Moreover, for various machine learning problems, the data are normalized in a preprocessing step. For example, in image processing, shapes are often analyzed through the Kendall shape space, which defines an equivalent class of all translations, rotations, and scalings of the set of points defining a shape [Kendall, 1984]. After a preshape process, consisting in removing the translation and scaling from the data, each shape is represented as a point on a hypersphere [Dryden, 2005]. As an other example, spherical representations have been recently exploited to design variational autoencoders with application to natural language processing [Xu and Durrett, 2018; Davidson et al., 2018].



(a) (b) (c)

Figure 2.1 – Sphere manifold $\mathcal{S}^2$, including e.g., (*a*) orientations. (*b*)-(*c*) Points on the surface of the sphere, such as $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the manifold. (*b*) The shortest path between $\boldsymbol{x}$ and $\boldsymbol{y}$ is the geodesic represented as a red curve, which differs from the Euclidean path depicted in blue. The vector $\boldsymbol{u}$ lies on the tangent space of $\boldsymbol{x}$ such that $\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y})$. (*c*) $\Gamma(\boldsymbol{v}_1)$ and $\Gamma(\boldsymbol{v}_2)$ are the parallel transported vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ from $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ to $\mathcal{T}_{\boldsymbol{y}}\mathcal{M}$. The inner product, i.e. the angle, between vectors is conserved by this operation.

The unit sphere $\mathcal{S}^d$, defined as

$$\mathcal{S}^d = \{\boldsymbol{x} \in \mathbb{R}^{d+1} : \boldsymbol{x}^\mathsf{T}\boldsymbol{x} = 1\}, \tag{2.6}$$

is a d-dimensional manifold embedded in $\mathbb{R}^{d+1}$. The tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{S}^d$ is the hyperplane tangent to the sphere at $\boldsymbol{x}$. The unit sphere is endowed with the canonical Riemannian metric

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{x}} = \boldsymbol{u}^\mathsf{T}\boldsymbol{v}, \tag{2.7}$$

with $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{S}^d$. The geodesic distance is defined as

$$d_{\mathcal{S}^d}(\boldsymbol{x}, \boldsymbol{y}) = \arccos(\boldsymbol{x}^\mathsf{T}\boldsymbol{y}), \tag{2.8}$$

with $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}^d$ and the corresponding exponential and logarithmic maps can be computed as (see e.g., [Absil et al., 2007b])

$$\boldsymbol{y} = \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) = \boldsymbol{x}\cos(\|\boldsymbol{u}\|) + \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}\sin(\boldsymbol{u}), \tag{2.9}$$

$$\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) = d_{\mathcal{S}^d}(\boldsymbol{x}, \boldsymbol{y})\,\frac{\boldsymbol{y} - \boldsymbol{x}^\mathsf{T}\boldsymbol{y}\,\boldsymbol{x}}{\|\boldsymbol{y} - \boldsymbol{x}^\mathsf{T}\boldsymbol{y}\,\boldsymbol{x}\|}. \tag{2.10}$$

The parallel transport of $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{S}^d$ to $\mathcal{T}_{\boldsymbol{y}}\mathcal{S}^d$ is given by

$$\tilde{\boldsymbol{v}} = \Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}) = \boldsymbol{A}_{\boldsymbol{x}\to\boldsymbol{y}}\boldsymbol{v}, \tag{2.11}$$

$$\text{with } \boldsymbol{A}_{\boldsymbol{x}\to\boldsymbol{y}} = -\boldsymbol{x}\sin(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^{\mathsf{T}} + \overline{\boldsymbol{u}}\cos(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^{\mathsf{T}} + (\boldsymbol{I} - \overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}^{\mathsf{T}}), \tag{2.12}$$

where $\boldsymbol{u} = \text{Log}_{\boldsymbol{x}}(\boldsymbol{y})$ and $\overline{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$. The different aforementioned operations are illustrated on $\mathcal{S}^2$ in Figure 2.1.

In this thesis, the sphere manifold $\mathcal{S}^3$, coinciding with unit quaternions, is used as a representation of robot orientations in the geometry-aware Bayesian optimization framework presented in Chapter 7, see also Figure 2.1a. Bayesian optimization on high-dimensional sphere manifolds is then studied in Chapter 8, with possible applications in shape space representations [Dryden, 2005] or various machine learning problems where data are normalized (e.g., in a preprocessing step).

2.2.4 The Symmetric Positive Definite Matrix Manifold

A $d\times d$ symmetric real matrix $\boldsymbol{X}$ is positive definite if $\boldsymbol{u}^{\mathsf{T}}\boldsymbol{X}\boldsymbol{u} > 0$ for all non-zero $\boldsymbol{u} \in \mathbb{R}^d$. Equivalently, the matrix $\boldsymbol{X}$ is positive definite if all its eigenvalues are positive (see e.g., [Bhatia, 2007] for a thorough documentation on positive definite matrices). Therefore, the set of $d\times d$ symmetric positive definite (SPD) matrices is denoted as $\mathcal{S}^d_{++}$ with

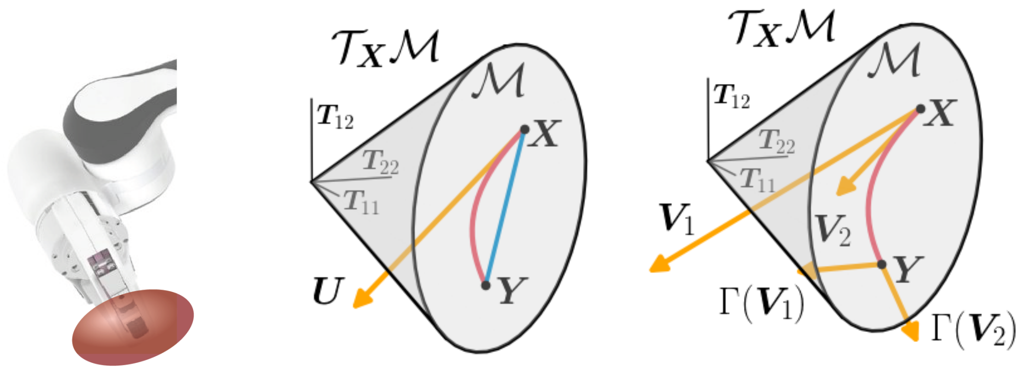
$$\mathcal{S}^d_{++} = \{\boldsymbol{X} \in \mathbb{R}^{d\times d} : \boldsymbol{X}_{ij} = \boldsymbol{X}_{ji}\ \forall i,j = 1\dots n \text{ and } \boldsymbol{u}^{\mathsf{T}}\boldsymbol{X}\boldsymbol{u} > 0\ \forall \boldsymbol{u} \in \mathbb{R}^d \backslash \boldsymbol{0}\}. \tag{2.13}$$

As the set $\mathcal{S}^d_{++}$ is not a vector space, the use of classical Euclidean space methods for treating and analyzing these matrices is inadequate. Similarly to spheres, a compelling solution is to endow these matrices with a Riemannian metric so that these form a Riemannian manifold. To do so, several Riemannian metrics have been proposed in the literature, notably the affine-invariant [Pennec et al., 2006] and Log-Euclidean [Arsigny et al., 2006] metrics[1], which both set matrices with null eigenvalues at an infinite distance of any SPD matrix. While the former provides excellent theoretical properties, it is computationally expensive in practice, therefore leading to a need for simpler metrics. However, as relatively low-dimensional SPD matrices are, in general, considered in this thesis, the computational cost of the operations derived from affine-invariant metric, i.e., the geodesic distance, exponential and logarithmic maps and parallel transport, remains reasonable for our algorithms and applications.

The manifold of $d\times d$ SPD matrices $\mathcal{S}^d_{++}$ can be represented as the interior of a convex cone embedded in the space of symmetric matrices Sym^d, as illustrated in Figure 2.2.

---

[1] Although they do not provide geodesic distances, several other metrics have also been proposed to measure distances on the SPD manifold, see e.g., [Sra, 2012; Wang and Vemuri, 2004].



(a) (b) (c)

Figure 2.2 – SPD manifold $\mathcal{S}^2_{++}$, including e.g., (a) stiffness ellipsoids. (b)-(c) One point in the graph corresponds to a matrix $\begin{pmatrix} T_{11} & T_{12} \\ T_{12} & T_{22} \end{pmatrix} \in \text{Sym}^2$ in which the manifold is embedded. Points inside the cone, such as $\boldsymbol{X}$ and $\boldsymbol{Y}$ belong to the manifold. (b) The shortest path between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is the geodesic represented as a red curve, which differs from the Euclidean path depicted in blue. $\boldsymbol{U}$ lies on the tangent space of $\boldsymbol{X}$ such that $\boldsymbol{U} = \text{Log}_{\boldsymbol{X}}(\boldsymbol{Y})$. (c) $\Gamma(\boldsymbol{V}_1)$ and $\Gamma(\boldsymbol{V}_2)$ are the parallel transported vectors $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ from $\mathcal{T}_{\boldsymbol{X}}\mathcal{M}$ to $\mathcal{T}_{\boldsymbol{Y}}\mathcal{M}$. The inner product between vectors is conserved by this operation.

Moreover, the tangent space at any point $\boldsymbol{X} \in \mathcal{S}^d_{++}$ is identified by the space of symmetric matrices Sym^d whose origin is located at $\boldsymbol{X}$. The affine-invariant metric transforms the original cone of SPD matrices into a regular and complete (but curved) manifold with an infinite development in each of its $d(d+1)/2$ directions [Pennec et al., 2006]. Therefore, matrices with null or negative eigenvalues are both at an infinite distance of any SPD matrix. The corresponding inner product between two matrices $\boldsymbol{U}$, $\boldsymbol{V} \in \mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++}$ is

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} \quad = \quad \text{tr}(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{U}\boldsymbol{X}^{-1}\boldsymbol{V}\boldsymbol{X}^{-\frac{1}{2}}). \tag{2.14}$$

Specifically, the exponential and logarithmic maps on the SPD manifold corresponding to the affine-invariant distance

$$d_{\mathcal{S}^d_{++}}(\boldsymbol{X}, \boldsymbol{Y}) = \| \log(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{Y}\boldsymbol{X}^{-\frac{1}{2}}) \|_{\text{F}}, \tag{2.15}$$

are computed as (see [Pennec et al., 2006] for details)

$$\boldsymbol{Y} = \text{Exp}_{\boldsymbol{X}}(\boldsymbol{U}) = \boldsymbol{X}^{\frac{1}{2}} \exp(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{U}\boldsymbol{X}^{-\frac{1}{2}})\boldsymbol{X}^{\frac{1}{2}}, \tag{2.16}$$

$$\boldsymbol{U} = \text{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = \boldsymbol{X}^{\frac{1}{2}} \log(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{Y}\boldsymbol{X}^{-\frac{1}{2}})\boldsymbol{X}^{\frac{1}{2}}, \tag{2.17}$$

where $\exp(\cdot)$ and $\log(\cdot)$ denote the matrix exponential and logarithm functions. Figure 2.2b illustrates the exponential and logarithmic maps on $\mathcal{S}^2_{++}$. The parallel transport of $\boldsymbol{V} \in \mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++}$ to $\mathcal{T}_{\boldsymbol{Y}}\mathcal{S}^d_{++}$ is given by

$$\tilde{\boldsymbol{V}} = \Gamma_{\boldsymbol{X}\to\boldsymbol{Y}}(\boldsymbol{T}) = \boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}}\ \boldsymbol{V}\ \boldsymbol{A}^{\mathsf{T}}_{\boldsymbol{X}\to\boldsymbol{Y}}, \tag{2.18}$$

with $\boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}} = \boldsymbol{Y}^{\frac{1}{2}}\boldsymbol{X}^{-\frac{1}{2}}$ (see [Sra and Hosseini, 2015] for details). Figure 2.2c illustrates the parallel transport operation on $\mathcal{S}^2_{++}$.

Note that, for a more efficient computation, the following equivalent formulations of operations on the SPD manifold can be implemented (see Chapter 6 of [Bhatia, 2007], and Chapter 3 of [Pennec et al., 2019] for details):

$$\begin{aligned} d_{\mathcal{S}^d_{++}}(\boldsymbol{X},\boldsymbol{Y}) &= \left(\sum_{i=1}^{d} \log^2 \lambda_i(\boldsymbol{X}^{-1}\boldsymbol{Y})\right)^{\frac{1}{2}}, \\ \mathrm{Exp}_{\boldsymbol{X}}(\boldsymbol{U}) &= \boldsymbol{X}\exp(\boldsymbol{X}^{-1}\boldsymbol{U}), \\ \mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) &= \boldsymbol{X}\log(\boldsymbol{X}^{-1}\boldsymbol{Y}), \\ \boldsymbol{A}_{\boldsymbol{X}\to\boldsymbol{Y}} &= (\boldsymbol{Y}\boldsymbol{X}^{-1})^{\frac{1}{2}}, \end{aligned}$$

where $\lambda_i(\boldsymbol{X})$ are the eigenvalues of the matrix $\boldsymbol{X}$.

In this thesis, we propose a probabilistic approach to learn sequences of SPD matrices from demonstrations in Chapter 3. As the SPD manifold coincides with the space of manipulability ellipsoids, it is then used in the learning, tracking and transfer frameworks for human and/or robot manipulability proposed in Chapters 4, 5 and 6. Finally, the manifold structure is further exploited for representing robot stiffness, inertia matrices and manipulability ellipsoids in the geometry-aware Bayesian optimization framework presented in Chapters 7 and 8.

2.3 Tensor Operations

Tensors are generalization of matrices to arrays of higher dimensions [Kolda and Bader, 2009], where vectors and matrices correspond to 1st and 2nd-order tensors. Tensor representation permits to represent and exploit data structure of multidimensional arrays. In this thesis, such representation is first used in the learning process to encode a distribution of SPD matrices (as explained in Chapter 3), which is exploited to learn a sequence of manipulability ellipsoids (see Chapter 4). Then, tensor representation is also exploited in the proposed manipulability tracking formulation to find the first-order differential relationship between the robot joints and the robot manipulability ellipsoid (1st- and 2nd-order tensors, respectively), which results in a 3rd-order tensor (see Chapter 5). Finally, we exploit tensor representations to take into account the structure of matrix-variate and tensor-variate data in a mixture of expert learning framework (as explained in Chapter 10). This section introduces the tensor operations needed for our mathematical treatment.

2.3.1 Notations

Table 2.2 introduces the notations and dimensionality of the variables related to the methods presented in Section 2.3. Vectors are denoted by lowercase bold letters, e.g., $\boldsymbol{x}$, matrices by uppercase bold letters, e.g., $\boldsymbol{X}$, and tensors by uppercase bold calligraphic letters, e.g., $\boldsymbol{\mathcal{X}}$. In particular, the matrix $\boldsymbol{X}_{(m)}$ results from the mode-m matricization of a tensor $\boldsymbol{\mathcal{X}}$. The matricization operation consists in reordering the elements of a tensor into a matrix. By defining fibers as generalization of matrix rows and columns to tensors (rows and columns are mode-1 and mode-2 fibers), the mode-m matricization turns the components of the mode-m fiber of $\boldsymbol{\mathcal{X}}$ into the columns of a matrix $\boldsymbol{X}_{(m)}$, as illustrated in Figure 2.3.

Table 2.2 – Tensor notations

Variable / operation	Description
M	Number of dimensions or modes of a tensor
$\boldsymbol{x} \in \mathbb{R}^{I_1}$	Vector variable
$\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$	Matrix variable
$\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$	Tensor variable
$\boldsymbol{X}_{(m)} \in \mathbb{R}^{I_m \times (I_1 \ldots I_{m-1} I_{m+1} \ldots I_M)}$	m-mode matricization or unfolding of a tensor

2.3.2 Product Operations

This section introduces the product operations defined in multilinear algebra for the mathematical treatment of tensors. These operations and corresponding symbols are listed in Table 2.3.

Outer product

The outer product between two vectors $\boldsymbol{x} \in \mathbb{R}^I$ and $\boldsymbol{y} \in \mathbb{R}^J$ results in a $I \times J$ matrix, so that

$$\boldsymbol{x} \circ \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^{\mathsf{T}}. \tag{2.19}$$

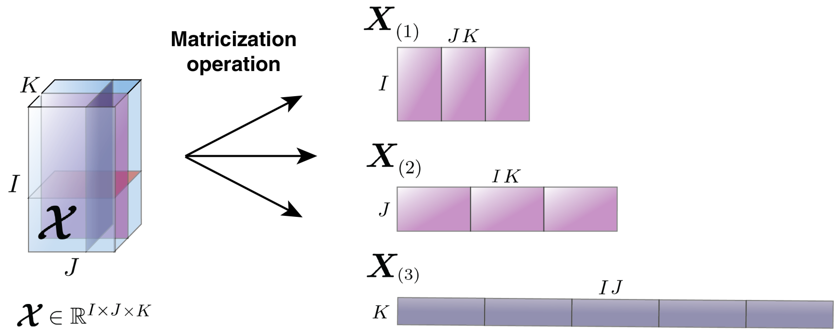


Figure 2.3 – Illustration of the tensor matricization operation on a third order tensor. The matrices $\boldsymbol{X}_{(1)} \in \mathbb{R}^{I\times JK}$, $\boldsymbol{X}_{(2)} \in \mathbb{R}^{J\times IK}$ and $\boldsymbol{X}_{(3)} \in \mathbb{R}^{K\times IJ}$ are three different matricizations of the tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$.

Table 2.3 – Tensor-related product operations.

Product operation	Description
$\boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \ldots \circ \boldsymbol{x}^{(M)}$	Outer product
$\boldsymbol{\mathcal{X}} \otimes \boldsymbol{\mathcal{Y}}$	Tensor product
$\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A}$	n-mode product
$\boldsymbol{X} \otimes_{\mathrm{K}} \boldsymbol{Y}$	Kronecker product
$\boldsymbol{X} \odot \boldsymbol{Y}$	Khatri-Rao product
$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle$	Inner product

In general, the outer product between M vectors $\boldsymbol{x}^{(m)}$, $m = 1 \ldots M$, results in a Mth-order tensor whose elements are defined as

$$\left(\boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \ldots \circ \boldsymbol{x}^{(M)}\right)_{i_1,i_2,\ldots,i_M} = x^{(1)}_{i_1} x^{(2)}_{i_2} \ldots x^{(M)}_{i_M}.$$

Tensor product

The tensor product is a multilinear generalization of the outer product of two vectors. The tensor product of two tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times\ldots\times I_M}$, $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{J_1\times\ldots\times J_N}$ is given by $\boldsymbol{\mathcal{X}} \otimes \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1\times\ldots\times I_M\times J_1\times\ldots\times J_N}$ whose elements are

$$(\boldsymbol{\mathcal{X}} \otimes \boldsymbol{\mathcal{Y}})_{i_1,\ldots,i_M,j_1,\ldots,j_N} = x_{i_1,\ldots,i_M}\, y_{j_1,\ldots,j_N}. \tag{2.20}$$

The tensor product is notably used to compute the covariance of tensor data, as explained in Section 2.3.3.

n-mode product

The multiplication of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \ldots \times I_n \times \ldots \times I_N}$ by a matrix $\boldsymbol{A} \in \mathbb{R}^{J \times I_n}$, known as the n-mode product, results in a tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \ldots \times J \times \ldots \times I_N}$, so that

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_n \boldsymbol{A} \iff \boldsymbol{Y}_{(n)} = \boldsymbol{A}\boldsymbol{X}_{(n)}, \tag{2.21}$$

where $\boldsymbol{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \ldots I_N}$ is the n-mode matricization or unfolding of tensor $\boldsymbol{\mathcal{X}}$. Element-wise, this n-mode product can be written as

$$(\boldsymbol{\mathcal{X}} \times_n \boldsymbol{A})_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{j_n i_n} x_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N}.$$

Kronecker product

The Kronecker product of two matrices $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ and $\boldsymbol{Y} \in \mathbb{R}^{P \times Q}$ is the $IP \times JQ$ matrix

$$\boldsymbol{X} \otimes_{\mathrm{K}} \boldsymbol{Y} = \begin{bmatrix} x_{11}\boldsymbol{Y} & \ldots & x_{1J}\boldsymbol{Y} \\ \vdots & \ddots & \vdots \\ x_{J1}\boldsymbol{Y} & \ldots & x_{IJ}\boldsymbol{Y} \end{bmatrix}. \tag{2.22}$$

Khatri-Rao product

The Khatri-Rao product of two matrices $\boldsymbol{X} \in \mathbb{R}^{I \times L}$ and $\boldsymbol{Y} \in \mathbb{R}^{J \times L}$ results in a matrix $\boldsymbol{X} \odot \boldsymbol{Y} \in \mathbb{R}^{IJ \times L}$ whose columns are equal to the Kronecker product of the columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$, i.e.,

$$\boldsymbol{X} \odot \boldsymbol{Y} = [\boldsymbol{x}_1 \otimes_{\mathrm{K}} \boldsymbol{y}_1 \quad \ldots \quad \boldsymbol{x}_K \otimes_{\mathrm{K}} \boldsymbol{y}_L]. \tag{2.23}$$

Inner product

The inner product of two tensors $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ is defined as the sum of the products of their entries, so that

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_M=1}^{I_M} x_{i_1, i_2, \ldots i_M}\, y_{i_1, i_2, \ldots i_M}. \tag{2.24}$$

Note that the inner product of two tensors is equivalent to the Frobenius inner product of their m-mode matricization or unfolding $\boldsymbol{X}_{(m)}, \boldsymbol{Y}_{(m)} \in \mathbb{R}^{I_m \times (I_1 \ldots I_{m-1} I_{m+1} \ldots I_M)}$ and to the inner product of their vectorization $\mathrm{vec}(\boldsymbol{\mathcal{X}})$, $\mathrm{vec}(\boldsymbol{\mathcal{Y}}) \in \mathbb{R}^{I_1 \dots I_M}$, namely

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \langle \boldsymbol{X}_{(m)}, \boldsymbol{Y}_{(m)} \rangle = \langle \mathrm{vec}(\boldsymbol{\mathcal{X}}), \mathrm{vec}(\boldsymbol{\mathcal{Y}}) \rangle. \tag{2.25}$$

2.3.3 Distribution of Symmetric Matrices and Related Operations

In Chapter 3, we propose to adapt the formulations of Gaussian mixture model (GMM) and Gaussian mixture regression (GMR) to encode and retrieve sequences of SPD matrices. To do so, we need to formulate a normal distribution of SPD matrices, for which we first require the formulation of a normal distribution for symmetric matrices. The corresponding distribution and related operations are defined by exploiting the previously-introduced tensor operations, as described below.

Tensor covariance

Similarly to the covariance of vectors, the $2M$th-order covariance tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{I_1 \times \dots \times I_M \times I_1 \times \dots \times I_M}$ of centered tensors $\overline{\boldsymbol{\mathcal{X}}}_n \in \mathbb{R}^{I_1 \times \dots \times I_M}$ is given by

$$\boldsymbol{\mathcal{S}} = \frac{1}{N-1} \sum_{n=1}^{N} \overline{\boldsymbol{\mathcal{X}}}_n \otimes \overline{\boldsymbol{\mathcal{X}}}_n, \tag{2.26}$$

where N is the total number of datapoints. This definition is used in the formulation of tensor-variate normal distributions.

Tensor contraction

As described in [Tyagi and Davis, 2008], we denote the element (i, j, k, l) of a 4th-order tensor $\boldsymbol{\mathcal{S}}$ by s_{ij}^{kl} with two covariant indices i, j and two contravariant indices k, l. The element (k,l) of a matrix $\boldsymbol{X}$ is denoted by x_{kl} with two covariant indices k, l. A tensor contraction between two tensors is performed when one or more contravariant and covariant indices are identical. For example, the tensor contraction of $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d \times d \times d \times d}$ and $\boldsymbol{X} \in \mathbb{R}^{d \times d}$ is written as

$$\boldsymbol{\mathcal{S}} \boldsymbol{X} = \sum_{k=1}^{D} \sum_{l=1}^{D} s_{ij}^{kl} x_{kl}. \tag{2.27}$$

Normal distribution of symmetric matrices

The tensor-variate normal distribution of a random 2nd-order symmetric matrix $\boldsymbol{X} \in \mathrm{Sym}^d$ with mean $\boldsymbol{\Xi} \in \mathrm{Sym}^d$ and covariance $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d \times d \times d \times d}$ is defined as [Basser and Pajevic, 2007]

$$\mathcal{N}(\boldsymbol{X}|\boldsymbol{\Xi},\boldsymbol{\mathcal{S}}) = \frac{1}{\sqrt{(2\pi)^{\tilde{d}}|\boldsymbol{\mathcal{S}}|}}\, e^{-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\Xi})\boldsymbol{\mathcal{S}}^{-1}(\boldsymbol{X}-\boldsymbol{\Xi})}, \tag{2.28}$$

with $\tilde{d} = d + d(d-1)/2$.

Mandel Notation

Mandel notation [Mandel, 1965] represents a symmetric matrix $\boldsymbol{X} \in \mathrm{Sym}^d$ in the form a $\tilde{d}$-dimensional vector $\mathrm{mvec}(\boldsymbol{X})$ containing the $\tilde{d} = d + d(d-1)/2$ distinct elements of the matrix. In Mandel notation, the off-diagonal elements of the matrix are scaled by $\sqrt{2}$. As an example, the expression of a 3×3 symmetric matrix $\boldsymbol{X}$ in Mandel notation is

$$\mathrm{mvec}(\boldsymbol{X}) = \begin{pmatrix} x_{11} \\ x_{22} \\ x_{33} \\ \sqrt{2}x_{23} \\ \sqrt{2}x_{13} \\ \sqrt{2}x_{12} \end{pmatrix} \quad \text{with } \boldsymbol{X} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{12} & x_{22} & x_{23} \\ x_{13} & x_{23} & x_{33} \end{pmatrix} \tag{2.29}$$

The advantage of Mandel notation over other representation of symmetric matrices, such as the Voigt notation, is that it conserves the inner product thanks to the off-diagonal element scaling, i.e., for two matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathrm{Sym}^d$

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \langle\ \mathrm{mvec}(\boldsymbol{X}), \mathrm{mvec}(\boldsymbol{Y})\ \rangle. \tag{2.30}$$

2.3.4 Matrix Derivatives

The geometry-aware manipulability tracking formulation introduced in Chapter 5 takes inspiration from the computation of the robot Jacobian, which is computed from the 1st-order time derivative of the robot forward kinematics. We use the tensor representation to similarly compute the 1st-order derivative of the function that describes the relationship between the robot joint configuration $\boldsymbol{q}$ and the manipulability ellipsoid $\boldsymbol{M}$. The related matrix derivatives identities are introduced in this section. In the following identities, the matrix $\boldsymbol{Y} \in \mathbb{R}^{I \times J}$ is a function of $\boldsymbol{x} \in \mathbb{R}^K$, while $\boldsymbol{A} \in \mathbb{R}^{L \times I}$ and $\boldsymbol{B} \in \mathbb{R}^{J \times L}$ are constant matrices.

Derivative of a matrix w.r.t a vector

The derivative of a matrix function $\boldsymbol{Y}$ with respect to a vector $\boldsymbol{x}$ is a 3rd-order tensor $\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}} \in \mathbb{R}^{I \times J \times K}$ such that

$$\left(\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}}\right)_{ijk} = \frac{\partial y_{ij}}{\partial x_k}. \tag{2.31}$$

Left multiplication by a constant matrix

When the matrix function $\boldsymbol{Y}$ is left-multiplied by a constant matrix, the partial derivatives of $\boldsymbol{Y}$ are given by

$$\frac{\partial \boldsymbol{A}\boldsymbol{Y}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}} \times_1 \boldsymbol{A}. \tag{2.32}$$

Proof.

$$\left(\frac{\partial \boldsymbol{A}\boldsymbol{Y}}{\partial \boldsymbol{x}}\right)_{ljk} = \frac{\partial}{\partial x_k} \sum_i a_{li} y_{ij} = \sum_i a_{li} \frac{\partial y_{ij}}{\partial x_k}$$

□

Right multiplication by a constant matrix

When the matrix function $\boldsymbol{Y}$ is right-multiplied by a constant matrix, the partial derivatives of $\boldsymbol{Y}$ are given by

$$\frac{\partial \boldsymbol{Y}\boldsymbol{B}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}} \times_2 \boldsymbol{B}^{\mathsf{T}}. \tag{2.33}$$

Proof.

$$\left(\frac{\partial \boldsymbol{Y}\boldsymbol{B}}{\partial \boldsymbol{x}}\right)_{ilk} = \frac{\partial}{\partial x_k} \sum_i y_{ij} b_{jl} = \sum_j b_{jl} \frac{\partial y_{ij}}{\partial x_k}$$

□

Derivative of the inverse of a matrix

Finally, another useful operation for our manipulability tracking formulation is the derivative of the inverse of the matrix $\boldsymbol{Y}$ with respect to the vector $\boldsymbol{x}$, which results in a 3rd-order tensor, namely

$$\frac{\partial \boldsymbol{Y}^{-1}}{\partial \boldsymbol{x}} = -\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}}^{\mathsf{T}} \times_1 \boldsymbol{Y}^{-1} \times_2 \boldsymbol{Y}^{-\mathsf{T}}. \tag{2.34}$$

Proof. We compute the derivative of the definition of the inverse $\boldsymbol{Y}^{-1}\boldsymbol{Y} = \boldsymbol{I}$ as

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{Y}^{-1}\boldsymbol{Y}) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{I}),$$

$$\frac{\partial \boldsymbol{Y}^{-1}}{\partial \boldsymbol{x}} \times_2 \boldsymbol{Y}^{\mathsf{T}} + \frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{x}} \times_1 \boldsymbol{Y}^{-1} = \boldsymbol{0}.$$

Then, by isolating $\frac{\partial \boldsymbol{Y}^{-1}}{\partial \boldsymbol{x}}$, we obtain

$$\frac{\partial \boldsymbol{Y}^{-1}}{\partial \boldsymbol{x}} = -\frac{\partial \boldsymbol{Y}^{\mathsf{T}}}{\partial \boldsymbol{x}} \times_1 \boldsymbol{Y}^{-1} \times_2 \boldsymbol{Y}^{\mathsf{T}-1}.$$

□

2.4 Learning Algorithms

This section introduces the two main learning algorithms used in this thesis. The Gaussian mixture regression (GMR) approach, based on Gaussian mixture models (GMM), is first described in the context of learning from demonstrations. This algorithm is extended to encode trajectories of symmetric positive definite (SPD) matrices to learn sequences of manipulability ellipsoids in Chapter 4. Gaussian processes (GPs), used as surrogate model in the geometry-aware Bayesian optimization (BO) framework proposed in Chapters 7 and 8, are then reviewed. The two presented learning algorithms are also combined in a novel model-based Gaussian process presented in Chapter 9.

2.4.1 Gaussian Mixture Model in Learning from Demonstrations

In the context of learning from demonstrations (LfD), robots are taught new skills by providing them with (possibly physical) demonstrations of the task at hand [Ravichandar et al., 2020]. In this context, robot motions can be generated from demonstrated trajectories using various deterministic — e.g., dynamical movement primitives (DMP) [Pastor et al., 2009] — or probabilistic methods, e.g., probabilistic movement primitives (ProMP) [Paraschos et al., 2013], Gaussian mixture regression (GMR) [Calinon, 2016], kernelized movement primitives (KMP) [Huang et al., 2019] or Gaussian process regression (GPR) [Schneider and Ertel, 2010]. For the applications tackled in this thesis (Chapters 3 and 4), we are interested in probabilistic approaches adapted for real-time computation, that can encode the variability of the predicted trajectory. This variability, also called aleatoric uncertainty, reflects the dispersion of the data collected during the demonstrations, which encode different solutions or trajectories for the execution of the task. Finally, we are interested in a method that can easily be combined with other approaches (Chapter 9). Therefore, we choose to use GMR, which we summarize in the following.

Gaussian mixture regression (GMR) is a parametric approach that exploits the Gaussian conditioning theorem to estimate the distribution of output data given input data [Ghahramani and Jordan, 1994; Cohn et al., 1996]. A Gaussian mixture model (GMM) is first estimated to encode the joint distribution of input and output datapoints, e.g., via an Expectation-Maximization (EM) algorithm. The output conditioned on the observed input is then predicted via a linear combination of conditional expectations. Hence, GMR does not fit the regression function directly, but relies instead on the learned joint distribution.

We denote $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{D_\mathcal{X}}$ and $\boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}^{D_\mathcal{Y}}$ the vectors of input and corresponding output data, respectively, with $D_\mathcal{X}$ and $D_\mathcal{Y}$ being the dimensionality of the input and output spaces. In a GMM with C components, the joint distribution of $(\boldsymbol{x}^\mathsf{T}, \boldsymbol{y}^\mathsf{T})^\mathsf{T}$, corresponding to a linear superposition of Gaussian, can be written as

$$p\Big(\big(\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{smallmatrix}\big)\Big) = \sum_{\ell=1}^{C} \pi_\ell \, \mathcal{N}\Big(\big(\begin{smallmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{smallmatrix}\big) | \, \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell\Big), \tag{2.35}$$

with π_ℓ, $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}_\ell$ the mixing coefficient (prior), mean and covariance matrix of the ℓ-th Gaussian component. The mixing coefficients satisfy $\sum_{\ell=1}^{C} \pi_\ell = 1$, $\pi_\ell > 0$, while $\boldsymbol{\mu}_\ell$ and $\boldsymbol{\Sigma}_\ell$ can be decomposed in function of input and output data as follows

$$\boldsymbol{\mu}_\ell = \begin{pmatrix} \boldsymbol{\mu}_\ell^\mathcal{X} \\ \boldsymbol{\mu}_\ell^\mathcal{Y} \end{pmatrix} \text{ and } \boldsymbol{\Sigma}_\ell = \begin{pmatrix} \boldsymbol{\Sigma}_\ell^\mathcal{X} & \boldsymbol{\Sigma}_\ell^{\mathcal{X}\mathcal{Y}} \\ \boldsymbol{\Sigma}_\ell^{\mathcal{Y}\mathcal{X}} & \boldsymbol{\Sigma}_\ell^\mathcal{Y} \end{pmatrix}.$$

The parameters of a GMM modeling a set of N observed data $\{(\boldsymbol{x}_n^\mathsf{T}, \boldsymbol{y}_n^\mathsf{T})^\mathsf{T}\}_{n=1}^N$ can be inferred by maximizing the likelihood of the model or, equivalently, the corresponding log-likelihood

$$\sum_{n=1}^{N} \log \left\{ \sum_{\ell=1}^{C} \pi_\ell \mathcal{N}\Big(\big(\begin{smallmatrix}\boldsymbol{x}_n\\ \boldsymbol{y}_n\end{smallmatrix}\big) | \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell\Big) \right\}. \tag{2.36}$$

To do so, the GMM parameters are first initialized, commonly with K-means, although other initializations methods are available. For example, in the case of time-driven data, the different components can be conveniently initialized by using several partitions of data equally spaced in time. After initializing the parameters, the log-likelihood of the model (2.36) can be maximized iteratively using an EM algorithm, which alternates between the two following update steps. First, the posterior probabilities or responsibilities

$\gamma_{\ell,n}$ of each component are computed in the *expectation*- or E-step with

$$\gamma_{\ell,n} = p\left(\ell \mid \left(\begin{smallmatrix}\boldsymbol{x}_n\\ \boldsymbol{y}_n\end{smallmatrix}\right)\right) = \frac{\pi_\ell\, \mathcal{N}\left(\left(\begin{smallmatrix}\boldsymbol{x}_n\\ \boldsymbol{y}_n\end{smallmatrix}\right) | \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell\right)}{\sum\limits_{i=1}^{C} \pi_i\, \mathcal{N}\left(\left(\begin{smallmatrix}\boldsymbol{x}_n\\ \boldsymbol{y}_n\end{smallmatrix}\right) | \,\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)}, \tag{2.37}$$

$$N_\ell = \sum_{n=1}^{N} \gamma_{\ell,n}. \tag{2.38}$$

Then, the mean, covariance matrix and prior of each GMM component is updated within the *maximization*- or M-step:

$$\boldsymbol{\mu}_\ell \leftarrow \frac{1}{N_\ell} \sum_{n=1}^{N} \gamma_{\ell,n} \begin{pmatrix}\boldsymbol{x}_n\\ \boldsymbol{y}_n\end{pmatrix}, \tag{2.39}$$

$$\boldsymbol{\Sigma}_\ell \leftarrow \frac{1}{N_\ell} \sum_{n=1}^{N} \gamma_{\ell,n} \left(\begin{pmatrix}\boldsymbol{x}_n\\ \boldsymbol{y}_n\end{pmatrix} - \boldsymbol{\mu}_\ell\right)\left(\begin{pmatrix}\boldsymbol{x}_n\\ \boldsymbol{y}_n\end{pmatrix} - \boldsymbol{\mu}_\ell\right)^{\mathsf{T}}, \tag{2.40}$$

$$\pi_\ell \leftarrow \frac{N_\ell}{N}. \tag{2.41}$$

These two steps are repeated until convergence of the algorithm, i.e., until the change of log-likelihood between two consecutive iterations falls below a given threshold. More details on GMM and the EM algorithm can be found, e.g., in [Bishop, 2006].

GMR computes the conditional distribution of the GMM joint distribution to infer the output vector conditioned on a given input vector. The resulting multimodal distribution can be approximated by a single Gaussian

$$\boldsymbol{y}|\boldsymbol{x} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}^{\mathcal{Y}}, \hat{\boldsymbol{\Sigma}}^{\mathcal{Y}}). \tag{2.42}$$

Its first and second order moments can be calculated from the conditional means and covariances of the multivariate Gaussian distributions associated with individual components using the laws of total mean and covariance, so that

$$\hat{\boldsymbol{\mu}}^{\mathcal{Y}}(\boldsymbol{x}) = \sum_{\ell=1}^{C} h_\ell(\boldsymbol{x}) \hat{\boldsymbol{\mu}}_\ell^{\mathcal{Y}}(\boldsymbol{x}), \tag{2.43}$$

$$\hat{\boldsymbol{\Sigma}}^{\mathcal{Y}}(\boldsymbol{x}) = \sum_{\ell=1}^{C} h_\ell(\boldsymbol{x}) \left(\hat{\boldsymbol{\Sigma}}_\ell^{\mathcal{Y}} + \hat{\boldsymbol{\mu}}_\ell^{\mathcal{Y}}(\boldsymbol{x})(\hat{\boldsymbol{\mu}}_\ell^{\mathcal{Y}}(\boldsymbol{x}))^{\mathsf{T}}\right) - \hat{\boldsymbol{\mu}}^{\mathcal{Y}}(\boldsymbol{x})(\hat{\boldsymbol{\mu}}^{\mathcal{Y}}(\boldsymbol{x}))^{\mathsf{T}}, \tag{2.44}$$

with componentwise conditional means and covariances

$$\hat{\boldsymbol{\mu}}_\ell^{\mathcal{Y}}(\boldsymbol{x}) = \boldsymbol{\mu}_\ell^{\mathcal{Y}} + \boldsymbol{\Sigma}_\ell^{\mathcal{YX}} \boldsymbol{\Sigma}_\ell^{\mathcal{X}-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_\ell^{\mathcal{X}}\right), \tag{2.45}$$

$$\hat{\boldsymbol{\Sigma}}_\ell^{\mathcal{Y}} = \boldsymbol{\Sigma}_\ell^{\mathcal{Y}} - \boldsymbol{\Sigma}_\ell^{\mathcal{YX}} \boldsymbol{\Sigma}_\ell^{\mathcal{X}-1} \boldsymbol{\Sigma}_\ell^{\mathcal{XY}}. \tag{2.46}$$

The so-called responsability h_ℓ of component ℓ is computed in closed form as

$$h_\ell(\boldsymbol{x}) = \frac{\pi_\ell \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_\ell^{\mathcal{X}}, \boldsymbol{\Sigma}_\ell^{\mathcal{X}}\right)}{\sum\limits_{i=1}^{C} \pi_i \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}_i^{\mathcal{X}}, \boldsymbol{\Sigma}_i^{\mathcal{X}}\right)}. \tag{2.47}$$

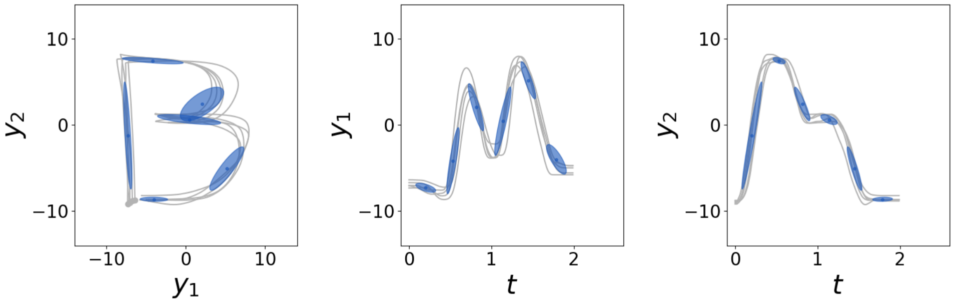
The computational complexity of GMR mainly depends on the number of GMM components as it governs the dimensionality of the maximum likelihood problem usually tackled by EM. Moreover, the number of GMM components is the only parameter that needs to be specified and can be estimated online, e.g., with a Bayesian nonparametric approach [Tanwani and Calinon, 2019]. Therefore, GMR is well adapted for real-time applications and its simplicity allows it to be combined easily with other complementary approaches.

Figure 2.4 shows an example of application of GMM and GMR. The training dataset consists of 5 demonstrations of a two-dimensional time-driven trajectory and is depicted by gray lines. A GMM ($C = 6$) is first trained to encode the joint distribution of the inputs t and outputs $\boldsymbol{y}$, as shown in Figure 2.4a. The conditional distribution of $\boldsymbol{y}$ given t is inferred by GMR, see Figure 2.4b. The covariance $\hat{\boldsymbol{\Sigma}}^{\mathcal{Y}}(\boldsymbol{x})$ of the distribution encodes the variability of the demonstrations. The output distribution is extrapolated in the absence of training data ($t > 2$).

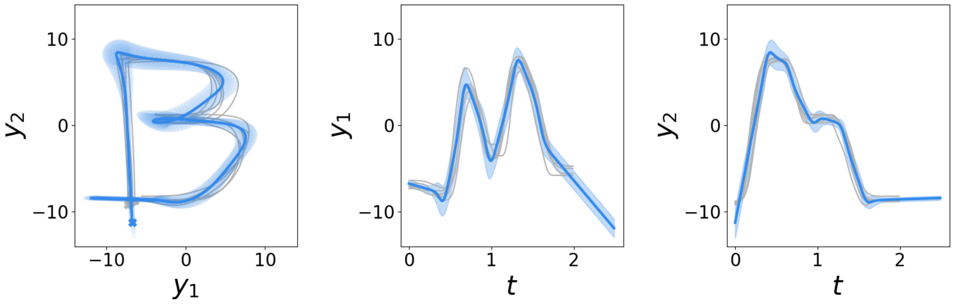
2.4.2 Gaussian Processes

Gaussian processes (GPs) form a class of non-parametric probabilistic models that aims at learning a deterministic input-output relationship, up to observation noise [Rasmussen and Williams, 2006]. Gaussian processes have been widely used in many different fields as an efficient tool to solve both regression and classification problems. In robotics, GPs have notably been used to infer robot trajectories in LfD [Schneider and Ertel, 2010; Silvério et al., 2018] or to estimate robot dynamics [Torres Alberto et al., 2014]. Moreover, GP models have also been used in policy search settings, particularly in data-efficient algorithms such as PILCO [Deisenroth and Rasmussen, 2011] and Bayesian optimization (BO) [Shahriari et al., 2016]. In particular, GPs are considered as the common surrogate model of BO and are used so in the Chapters 7 and 8 of this thesis. Gaussian processes are briefly introduced below.

In the noiseless GP framework, the output y is typically seen as a function of a controlled input $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{D_\mathcal{X}}$ with $D_\mathcal{X}$ being the dimensionality of the input space. Randomness comes in an instrumental way as the function $y(\boldsymbol{x})$ is assumed to be one realization of a Gaussian random process or random function denoted $f(\boldsymbol{x})$. A Gaussian Process



(a) The 6 GMM components are represented by blue ellipsoids.



(b) The mean of the GMR is represented by a continuous line and the variance by a light tube around the estimate.

Figure 2.4 – Illustration of (a) GMM and (b) GMR on a two-dimensional time-driven trajectory. The training data are shown in light gray for all graphs. The *left* graphs show the output trajectories estimated by GMR. The beginning of the trajectories is marked by a cross. The *right* graphs show the estimated trajectory for each output component as a function of the input t. Time and positions are given in seconds and centimeters, respectively.

$\mathcal{GP}(\mu, k)$ places a Gaussian prior over potential objective functions, so that

$$f(\boldsymbol{x}) \sim \mathcal{N}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x})), \tag{2.48}$$

with mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and positive-definite kernel (or covariance function) $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

Given a set of N observed data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, predictions of the objective function are then made by relying on the conditional distribution of $f(\boldsymbol{x})$ knowing that $f(\boldsymbol{x}_n)$ coincides with the observed outputs y_n at their corresponding observation inputs $\boldsymbol{x}_n$. With $\tilde{\boldsymbol{x}}$ representing an arbitrary test point, the random variable $\tilde{f} = f(\tilde{\boldsymbol{x}})$ conditioned on observations is also normally distributed

$$\tilde{f} \mid \tilde{\boldsymbol{x}}, \{\boldsymbol{x}_n, y_n\}_{n=1}^{N} \sim \mathcal{N}(\hat{\mu}(\tilde{\boldsymbol{x}}), \hat{\sigma}^2(\tilde{\boldsymbol{x}})), \tag{2.49}$$

with the following posterior mean and variance functions:

$$\hat{\mu}(\tilde{\boldsymbol{x}}) = \mu(\tilde{\boldsymbol{x}}) + \boldsymbol{k}(\tilde{\boldsymbol{x}})^\mathsf{T}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{\mu}), \tag{2.50}$$

$$\hat{\sigma}^2(\tilde{\boldsymbol{x}}) = k(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - \boldsymbol{k}(\tilde{\boldsymbol{x}})^\mathsf{T}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{k}(\tilde{\boldsymbol{x}}), \tag{2.51}$$

where $\boldsymbol{y}$ is the vector composed of the observations y_n, $\boldsymbol{\mu}$ is the vector composed of the means $\mu(\boldsymbol{x}_n)$, $\boldsymbol{K}$ is the covariance matrix with element (i, j) equal to $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{k}(\tilde{\boldsymbol{x}})$ is a vector of covariance terms between $\tilde{\boldsymbol{x}}$ and the observations $\boldsymbol{x}_n$ and σ_ϵ denotes the observation noise [Rasmussen and Williams, 2006]. The posterior mean and variance evaluated at any point $\tilde{\boldsymbol{x}}$ respectively represent the model prediction and uncertainty of the function at $\tilde{\boldsymbol{x}}$. This uncertainty is an epistemic uncertainty, therefore related to the presence or absence of data in the training dataset for a given region of the input space $\mathcal{X}$.

The mean and kernel functions completely specify the GP. The most common choice for the mean function is a constant value, while the kernel typically has the property that close points in the input space have stronger correlation than distant points. One popular kernel is the squared-exponential (SE) kernel

$$k_{\mathrm{SE}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \exp\left(-\beta d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2\right), \tag{2.52}$$

where $d(\cdot, \cdot)$ denotes the distance between two observations and the parameters β and θ control the horizontal and vertical scale of the function. Another widely used class of kernels is the Matérn covariance functions

$$k_{\text{Matérn}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\beta\sqrt{2\nu}\, d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)^{\nu} K_\nu \left(\beta\sqrt{2\nu}\, d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right), \tag{2.53}$$

where $\Gamma(\cdot)$ and K_ν denotes the gamma distribution and the modified Bessel function, respectively. The Matérn covariance functions can be expressed in simple form when the

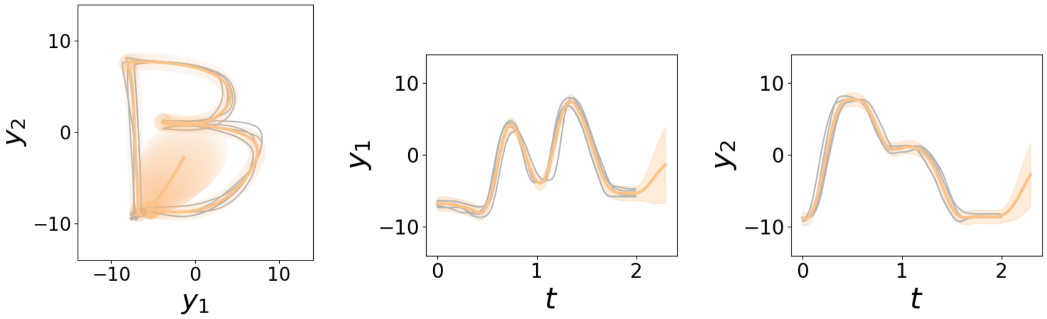


Figure 2.5 – Illustration of GPR on a two-dimensional time-driven trajectory. The training data are shown in light gray for all graphs. The mean is represented by a continuous line and the variance by a light tube around the estimate. The *left* graphs show the output trajectories estimated by GPR, respectively. The beginning of the trajectories is marked by a cross. The *right* graphs show the estimated trajectory for each output component as a function of the input t. Time and positions are given in seconds and centimeters, respectively.

parameter ν is half-integer. In particular, the Matérn kernels with parameter $\nu = 1/2$, $\nu = 3/2$ and $\nu = 5/2$ can be expressed as

$$k_{\text{Matérn},1/2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \exp\left(-\beta d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right), \tag{2.54}$$

$$k_{\text{Matérn},3/2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \left(1 + \sqrt{3}\beta d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \exp\left(-\sqrt{3}\beta d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right), \tag{2.55}$$

$$k_{\text{Matérn},5/2}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \left(1 + \sqrt{5}\beta d(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{5}{3}\beta^2 d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2\right) \exp\left(-\sqrt{5}\beta d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right). \tag{2.56}$$

The Matérn kernel with $\nu = 1/2$ is also known as the exponential kernel, while the case $\nu \to \infty$ corresponds to the SE kernel (2.52). In general, low ν-values correspond to rough GPs, while high ν-values generate smoother functions. The kernel parameters and the observation noise are usually inferred via maximum likelihood estimation (MLE), i.e., by setting the set of parameters Θ as those maximizing the log of the marginal likelihood [Rasmussen and Williams, 2006]

$$\log p(\boldsymbol{y}|\boldsymbol{x}, \Theta) = -\frac{1}{2}\boldsymbol{y}^{\mathsf{T}}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y} - \frac{1}{1}\log(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I}) - \frac{n}{2}\log(2\pi). \tag{2.57}$$

Figure 2.5 shows an example of application of Gaussian process regression (GPR) with a zero-valued mean function and a Matérn kernel ($\nu = 5/2$). The training data set consists of 5 demonstrations of a two-dimensional time-driven trajectory. The conditional distribution of $\boldsymbol{y}$ given t corresponds to the posterior distribution of the GP. We observe that, in the absence of training data ($t > 2$), the prediction tends to follow the prior mean value and shows a high uncertainty.

2.5 Conclusion

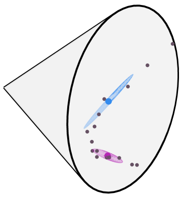
The mathematical tools, methods and algorithms introduced in this chapter are the backbone of this thesis. Several of the proposed models and solutions for different robot learning problems described in this thesis exploit one or several of the tools presented here. The following chapters constitute the core of this thesis and describe in details how robots can learn and refine skills while exploiting domain knowledge through geometry- and structure-awareness. Specific background will be further presented at the beginning of certain chapters.

Part I

Geometry-aware Robot Learning from Demonstrations

3 Gaussian Mixture Model on the SPD Manifold

One of the main contributions of this thesis is built on the problem of transferring skills to a robot. Specifially, the Part I of this thesis proposes a *geometry-aware* probabilistic framework to learn symmetric positive definite (SPD) matrices from demonstrations.



The proposed approach, which exploits Riemannian manifold theory and tensor approaches, is introduced in the following chapter. Moreover, a detailed analysis of the importance of geometry-awareness in the proposed learning formulation is proposed at the end of this chapter.

Publication note

The material presented in this chapter is adapted from the following publications:

- Jaquier, N. and Calinon, S. (2017). Gaussian mixture regression on symmetric positive definite matrices manifolds: Application to wrist motion estimation with sEMG. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 59–64;
- Rozo, L., Jaquier, N., Calinon, S., and Caldwell, D. G. (2017). Learning manipulability ellipsoids for task compatibility in robot manipulation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3183–3189;
- Jaquier, N., Rozo, L., Caldwell, D. G., and Calinon, S. (2020b). Geometry-aware manipulability learning, tracking and transfer. *Intl. Journal of Robotics Research.*

Source code

Source codes related to this chapter are available at:
https://github.com/NoemieJaquier/Manipulability.

3.1 Introduction

As data lying on Riemannian manifolds arise naturally in many domains and are shown to be well-adapted descriptors for various applications, there is an obvious necessity of being able to learn (from) these particular data. To do so, a naive approach considers these data as if they belonged to the Euclidean space and applies classical learning methods. Beyond the mathematical inconsistency and the inexactitude of this approach, the performance of the learning algorithm may be seriously compromised by ignoring the intrinsic geometry of the data. Moreover, a post-processing phase may be required if the output of the learning algorithm must lie on a specific manifold, which may bias the learning outcome. As an example, consider a learning problem where the output must lie on the SPD manifold. To guarantee the positive-definiteness of the outputs, an obvious naive solution adds a post-processing phase that increases the negative eigenvalues of the matrix until they become positive. However, this may generate a bunch of data with close-to-zero eigenvalues, which may be problematic for the task at hand and may not reflect the real distribution of the outputs on the SPD manifold.

Therefore, a smarter way to learn quantities belonging to Riemannian manifolds involves adapting the classical algorithms towards *geometry-aware* learning approaches. The resulting methods (see, e.g., [Fletcher, 2013; Mallasto and Feragen, 2018; Barachant et al., 2012]) allow an efficient exploitation of the available knowledge on the intrinsic geometry of the data and lead to better performance compared to the corresponding geometry-oblivious approaches. In this chapter, we are interested in learning SPD matrices by demonstrations. We focus here on a parametric method widely used in LfD, namely Gaussian mixture regression (see Section 2.4 for a short background), and propose a tensor-based formulation of Gaussian mixture model (GMM) and Gaussian mixture regression (GMR) that takes into account the geometry of the SPD manifold (introduced in Section 2.2.4). The proposed approach allows the encoding and retrieval of trajectories formed by SPD matrices.

Several works presented extensions of the GMM/GMR framework to Riemannian manifolds, notably to the sphere manifold. Simo-Serra et al. [2017] proposed an extension of EM algorithm to Riemannian manifold for data in vector form. Their approach allows each distribution to be located on its own tangent space. Kim et al. [2017] also reformulated GMM to handle the space of rotation in $\mathbb{R}^3$. Therefore, both Simo-Serra et al. [2017] and Kim et al. [2017] presented methods for regression from a mixture of Gaussians on Riemannian manifolds, although they only partially exploited the manifold structure in Gaussian conditioning. Zeestraten [2018] extended probabilistic encoding using Gaussians to Riemannian manifolds with data represented in vector form. The author demonstrated that the parallel transport of covariance matrices is essential for Gaussian conditioning — i.e., for GMR — on Riemannian manifolds. Other works focused on extending GMM to SPD matrices. Said et al. [2017] proposed an implementation of EM for Gaussian mixture on SPD manifolds. However, their formulation only mildly characterizes the (co)variability of SPD matrices as it is represented by a scalar. Moreover, Zhan and Ma [2012] proposed a modified EM algorithm for GMM on SPD manifolds, where SPD matrices are converted to vectors using Voigt notation. However, to the best of our knowledge, no extension of GMR to the SPD manifold has been presented in the literature.

In this chapter, we present the mathematical formulation of a geometry-aware Gaussian mixture model that encodes a set of demonstrations either lying on the SPD manifold or lying on a product space, where one space is a SPD manifold. This probabilistic formulation models the trend of the demonstrated sequences along with their variability, reflecting their dispersion through the different demonstrations (see Section 3.3). To do so, we exploit a normal distribution on the SPD manifold, where the covariability of matrices has the form of a 4th-order tensor (see Section 3.2 for a short background). As presented in Section 3.4, a distribution of trajectories can then be retrieved via a geometry-aware GMR on the SPD manifold. Finally, we show the importance of geometry-awareness in our learning framework in Section 3.5.

3.2 Background: Statistics on SPD Manifolds

Statistical tools adapted to Riemannian manifolds are required to analyze, compare and learn manifold-valued observations. We summarize here the main tools used in this chapter, namely the computation of mean and covariance values and the notion of normal distribution on Riemannian manifolds. For our formulation of GMM and GMR, we particularly focus on statistics over a set of SPD observations. For a thorough introduction to intrinsic statistics on Riemannian manifolds, we refer the interested reader to [Pennec et al., 2006].

3.2.1 Mean and Covariance of SPD Matrices

In this thesis, we use the Riemannian manifold framework to compute the mean and covariance of SPD matrix profiles. These notions are exploited in the definition of the normal distribution on Riemannian manifold, as well as in the analysis of manipulability ellipsoids profiles in Chapter 4.

A Riemannian treatment is necessary to ensure that the mean of a set of SPD matrices is unique and valid, meaning that it belongs to the space of SPD matrices. Similarly to the notion of mean value in the Euclidean space, the Fréchet mean [Fréchet, 1948] of a set of N datapoints $\boldsymbol{X}_n \in \mathcal{S}_{++}^d$ corresponds to the matrix $\boldsymbol{\Xi} \in \mathcal{S}_{++}^d$ that minimizes the variance of the observations w.r.t $\boldsymbol{\Xi}$, or equivalently the sum of the squared manifold distances $\sum_{n=1}^{N} d(\boldsymbol{\Xi}, \boldsymbol{X}_n)$, so that

$$\boldsymbol{\Xi} = \underset{\boldsymbol{\Xi} \in \mathcal{S}_{++}^{d}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{n=1}^{N} d(\boldsymbol{\Xi}, \boldsymbol{X}_n) \right). \tag{3.1}$$

As detailed in [Pennec et al., 2006], this optimization problem can be solved iteratively with a Gauss-Newton algorithm. At each iteration, the datapoints are first projected into the tangent space of the current estimate of the mean $\boldsymbol{\Xi}$ using the logarithmic map. Then, the Euclidean mean of these points is computed and projected to the manifold using the exponential map, which corresponds to the updated estimate of the mean, i.e.,

$$\boldsymbol{\Xi} \leftarrow \operatorname{Exp}_{\boldsymbol{\Xi}} \left(\frac{1}{N} \sum_{n=1}^{N} \operatorname{Log}_{\boldsymbol{\Xi}}(\boldsymbol{X}_n) \right). \tag{3.2}$$

The initial estimate of $\boldsymbol{\Xi}$ can for example be set as one of the points $\boldsymbol{X}_n$. This optimization problem tends to converge rapidly, typically in less than 10 iterations. Note that the Fréchet mean is defined not only for Riemannian manifolds, but in general for all the metric spaces[1]. We presented its definition on the SPD manifold for convenience.

Generally speaking, the covariance $\boldsymbol{\Sigma}$ of observations $\boldsymbol{x}_n \in \mathcal{M}$ belongs to the tangent space $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$ of their mean $\boldsymbol{\mu} \in \mathcal{M}$ and is obtained by computing the covariance of the points projected to the tangent space of the mean with $\operatorname{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)$, i.e.,

$$\boldsymbol{\Sigma} = \frac{1}{N-1} \sum_{n=1}^{N} \operatorname{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n) \operatorname{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)^{\mathsf{T}}. \tag{3.3}$$

As SPD matrices can be seen as 2nd-order tensors, their covariance can be computed as a 4th-order tensor, see (2.26). Therefore, the covariance tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d \times d \times d \times d}$ of a set of observations $\boldsymbol{X}_n \in \mathcal{S}_{++}^{d}$ is computed in the tangent space $\mathcal{T}_{\boldsymbol{\Xi}}\mathcal{S}_{++}^{d}$ of the mean $\boldsymbol{\Xi}$ as

$$\boldsymbol{\mathcal{S}} = \frac{1}{N-1} \sum_{n=1}^{N} \operatorname{Log}_{\boldsymbol{\Xi}}(\boldsymbol{X}_n) \otimes \operatorname{Log}_{\boldsymbol{\Xi}}(\boldsymbol{X}_n), \tag{3.4}$$

where $\otimes$ represents the tensor product (2.20). The concepts of mean and covariance on the SPD manifold $\mathcal{S}_{++}^{2}$ are illustrated in Figure 3.1.

3.2.2 The Normal Distribution for SPD Matrices

In order to define a normal distribution for SPD matrices, we first introduce the concept of normal distribution on Riemannian manifolds and of normal distribution of symmetric

---

[1] In some cases, Fréchet mean is equivalent to the Karcher mean, which designates local means of probabilitity measures on Riemannian manifold (see [Arnaudon et al., 2011; Pennec et al., 2006] for details).

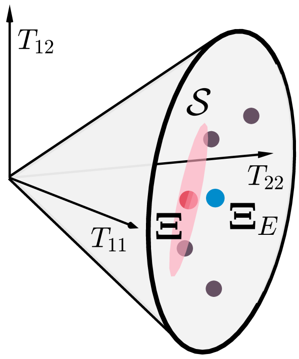


Figure 3.1 – Illustration of the concepts of mean and covariance on the SPD manifold $\mathcal{S}^2_{++}$. One point represents a matrix $\left(\begin{smallmatrix} T_{11} & T_{12} \\ T_{12} & T_{22} \end{smallmatrix}\right) \in \mathrm{Sym}^2$. Points inside the cone, such as the 4 dark gray dots, belong to $\mathcal{S}^2_{++}$. The Fréchet mean $\boldsymbol{\Xi}$ and covariance tensor $\boldsymbol{\mathcal{S}}$ for the 4 points on the manifold are displayed as red point and ellipsoid, respectively. The mean $\boldsymbol{\Xi}$ differs from the Euclidean mean $\boldsymbol{\Xi}_E$ depicted in blue.

matrices. These two concepts can then be combined to form a normal distribution on the SPD manifold.

The Normal Distribution on Riemannian Manifolds

Pennec et al. [2006] proposed to generalize the multivariate normal or Gaussian distribution to Riemannian manifolds by maximizing the entropy in the tangent space knowing the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the observations. As shown in [Pennec et al., 2006], the multivariate Gaussian distribution on Riemannian manifold can be locally formulated as

$$\mathcal{N}_{\mathcal{M}}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}\, e^{-\frac{1}{2}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})\,\boldsymbol{\Sigma}^{-1}\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})}. \tag{3.5}$$

This formulation has the advantage of being simple to use and has been successfully exploited in [Simo-Serra et al., 2017; Dubbelman, 2011; Zeestraten, 2018].

The Normal Distribution of Symmetric Matrices

Random vectors and covariance matrices of a multivariate normal distribution can be interpreted as 1st and 2nd-order tensors. Following this interpretation, Basser and Pajevic [2007] proposed to generalize this statement to a tensor-variate normal distribution of a 2nd-order random symmetric tensor $\boldsymbol{X} \in \mathrm{Sym}^d$ with a mean $\boldsymbol{\Xi} \in \mathrm{Sym}^d$ and 4th-order covariance tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d\times d\times d\times d}$ (2.26), so that

$$\mathcal{N}(\boldsymbol{X}|\boldsymbol{\Xi},\boldsymbol{\mathcal{S}}) = \frac{1}{\sqrt{(2\pi)^{\tilde{d}}|\boldsymbol{\mathcal{S}}|}}\, e^{-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{\Xi})\boldsymbol{\mathcal{S}}^{-1}(\boldsymbol{X}-\boldsymbol{\Xi})}, \tag{3.6}$$

where we used the tensor contraction (2.27) to compute the argument in the exponential and $\tilde{d} = d + d(d-1)/2$ is the number of distinct elements of the symmetric matrix belonging to the space Sym^d. Note that this distribution requires the computation of the inverse and the determinant of 4th-order tensors. This can be resolved by exploiting the symmetries of the matrices $\boldsymbol{X}$, Ξ and of the covariance tensor $\boldsymbol{\mathcal{S}}$, as shown by Basser and Pajevic [2007] and summarized below.

We denote here the element (i, j, m, n) of a 4th-order tensor $\boldsymbol{\mathcal{S}}$ by s_{ij}^{mn} with two covariant indices i, j and two contravariant indices m, n. The element (i,j) of a matrix $\boldsymbol{X}$ is denoted by x_{ij} with two covariant indices i, j. As stated previously, $\boldsymbol{X} \in \mathrm{Sym}^d$ is a symmetric matrix of dimension d, such that $x_{ij} = x_{ji}$. In this case, the covariance tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d\times d\times d\times d}$ inherits symmetries such that $s_{ij}^{mn} = s_{ji}^{mn} = s_{ij}^{nm}$ and $s_{ij}^{mn} = s_{mn}^{ij}$. Therefore, by using Mandel notation (see Section 2.3.3 and (2.29)), the symmetric matrix $\boldsymbol{X}$ can be written as a $\tilde{d}$-dimensional vector $\boldsymbol{x}$. Similarly, any 4th-order tensor $\boldsymbol{\mathcal{S}}$ satisfying the aforementioned symmetry properties can be mapped into a symmetric positive definite 2nd-order tensor $\boldsymbol{\Sigma} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$. For simplicity, we take the example of $d = 3$, which can be generalized to higher dimensions. $\boldsymbol{X}$ is in this case written as a $\tilde{d} = 6$-dimensional column vector $\boldsymbol{x}$ with

$$\boldsymbol{x} = \mathrm{mvec}(\boldsymbol{X}) = \begin{pmatrix} x_{11} \\ x_{22} \\ x_{33} \\ \sqrt{2}x_{12} \\ \sqrt{2}x_{13} \\ \sqrt{2}x_{23} \end{pmatrix}. \tag{3.7}$$

Similarly, a $3 \times 3 \times 3 \times 3$ tensor $\boldsymbol{\mathcal{S}}$ is converted to a 6×6 symmetric matrix $\boldsymbol{\Sigma}$ as

$$\boldsymbol{\Sigma} = \mathrm{mmat}(\boldsymbol{\mathcal{S}}) = \begin{pmatrix} s_{11}^{11} & s_{11}^{22} & s_{11}^{33} & \sqrt{2}s_{11}^{12} & \sqrt{2}s_{11}^{13} & \sqrt{2}s_{11}^{23} \\ s_{11}^{22} & s_{22}^{22} & s_{22}^{33} & \sqrt{2}s_{22}^{12} & \sqrt{2}s_{22}^{13} & \sqrt{2}s_{22}^{23} \\ s_{11}^{33} & s_{22}^{33} & s_{33}^{33} & \sqrt{2}s_{33}^{12} & \sqrt{2}s_{33}^{13} & \sqrt{2}s_{33}^{23} \\ \sqrt{2}s_{11}^{12} & \sqrt{2}s_{22}^{12} & \sqrt{2}s_{33}^{12} & 2s_{12}^{12} & 2s_{12}^{13} & 2s_{12}^{23} \\ \sqrt{2}s_{11}^{13} & \sqrt{2}s_{22}^{13} & \sqrt{2}s_{33}^{13} & 2s_{12}^{13} & 2s_{13}^{13} & 2s_{13}^{23} \\ \sqrt{2}s_{11}^{23} & \sqrt{2}s_{22}^{23} & \sqrt{2}s_{33}^{23} & 2s_{12}^{23} & 2s_{13}^{23} & 2s_{23}^{23} \end{pmatrix}. \tag{3.8}$$

Therefore, due to the properties of Mandel notation, the tensor contraction $\boldsymbol{X}\boldsymbol{\mathcal{S}}^{-1}\boldsymbol{X}$ is equal to $\boldsymbol{x}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$ and the normal probability density function $\mathcal{N}(\boldsymbol{X}|\boldsymbol{M}, \boldsymbol{\mathcal{S}})$ (3.6) is equivalent to

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^{\tilde{d}}|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \tag{3.9}$$

with $\boldsymbol{x} = \mathrm{mvec}(\boldsymbol{X})$, $\boldsymbol{\mu} = \mathrm{mvec}(\Xi)$ and $\boldsymbol{\Sigma} = \mathrm{mmat}(\boldsymbol{\mathcal{S}})$. This equivalence allows a simple computation of the symmetric-matrix-variate normal distribution (3.6).

Another advantage of the transformation (3.8) is that it allows us to compute easily the eigenvalues and eigentensor of the 4th-order tensor $\boldsymbol{\mathcal{S}}$. These can notably be used to directly compute the inverse $\boldsymbol{\mathcal{S}}^{-1}$ and determinant $|\boldsymbol{\mathcal{S}}|$ and will also be crucial for the parallel transport operation of 4th-order tensors between tangent spaces of the SPD manifold in the GMR presented in Section 3.4. Similarly to the eigenvalue decomposition of matrices, the eigenvalues λ and eigentensors $\boldsymbol{V} \in \mathbb{R}^{d \times d}$ of a 4th-order tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d \times d \times d \times d}$ satisfy the fundamental equation $\boldsymbol{\mathcal{S}} \boldsymbol{V} = \lambda \boldsymbol{V}$. They may also be found using the correspondence between the 4th-order covariance $\boldsymbol{\mathcal{S}}$ and the 2nd-order covariance $\boldsymbol{\Sigma}$. First, the eigenvalues λ of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mathcal{S}}$ are equal. Then, similarly to transformations (3.7) and (3.8), there is a one-to-one relationship between the elements of a given eigenvector $\boldsymbol{v} \in \mathbb{R}^{\tilde{d}}$ of $\boldsymbol{\Sigma}$ and the corresponding eigentensor $\boldsymbol{V} \in \mathrm{Sym}^d$ of $\boldsymbol{\mathcal{S}}$. In the example $d = 3$, given an eigenvector of $\boldsymbol{\Sigma}$, such that $\boldsymbol{v} = (v_{11}, v_{22}, v_{33}, v_{12}, v_{13}, v_{23})^{\mathsf{T}}$, the corresponding eigentensor of $\boldsymbol{\mathcal{S}}$ is

$$\boldsymbol{V} = \begin{pmatrix} v_{11} & \frac{1}{\sqrt{2}} v_{12} & \frac{1}{\sqrt{2}} v_{13} \\ \frac{1}{\sqrt{2}} v_{12} & v_{22} & \frac{1}{\sqrt{2}} v_{23} \\ \frac{1}{\sqrt{2}} v_{13} & \frac{1}{\sqrt{2}} v_{23} & v_{33} \end{pmatrix}, \tag{3.10}$$

which can be easily generalized to higher dimensions.

Similarly to the matrix case, the inverse and determinant of the covariance $\boldsymbol{\mathcal{S}}$ can be computed using the eigenvalues λ_k and eigentensors $\boldsymbol{V}_k$ as

$$\boldsymbol{\mathcal{S}}^{-1} = \sum_k \lambda_k^{-1} \, \boldsymbol{V}_k \otimes \boldsymbol{V}_k, \tag{3.11}$$

and the determinant is given by

$$|\boldsymbol{\mathcal{S}}| = \sum_k \lambda_k. \tag{3.12}$$

The Normal Distribution on the SPD Manifold

By combining the notions of normal distribution on Riemannian manifolds (3.5) and normal distribution for symmetric matrices (3.6), we can define the normal distribution on the SPD manifold as

$$\mathcal{N}_{\mathcal{M}}(\boldsymbol{X} | \boldsymbol{\Xi}, \boldsymbol{\mathcal{S}}) = \frac{1}{\sqrt{(2\pi)^{\tilde{d}} |\boldsymbol{\mathcal{S}}|}} \, e^{-\frac{1}{2} \mathrm{Log}_{\boldsymbol{\Xi}}(\boldsymbol{X}) \, \boldsymbol{\mathcal{S}}^{-1} \, \mathrm{Log}_{\boldsymbol{\Xi}}(\boldsymbol{X})}, \tag{3.13}$$

where $\boldsymbol{X} \in \mathcal{S}_{++}^d$, $\boldsymbol{\Xi} \in \mathcal{S}_{++}^d$ is the mean of the distribution and origin in the tangent space $\mathcal{T}_{\boldsymbol{\Xi}} \mathcal{S}_{++}^d$ and $\boldsymbol{\mathcal{S}} \in \mathcal{T}_{\boldsymbol{\Xi}} \mathcal{S}_{++}^d$ is the covariance tensor.

3.3 Gaussian Mixture Model on SPD Manifolds

The extension of GMM to SPD manifolds is now introduced by exploiting the normal distribution on SPD manifolds (3.13). Similarly to the Euclidean case, a GMM on the

SPD manifold is defined by

$$p(\boldsymbol{X}) = \sum_{\ell=1}^{C} \pi_\ell \mathcal{N}_{\mathcal{M}}(\boldsymbol{X}|\Xi_\ell, \boldsymbol{\mathcal{S}}_\ell), \tag{3.14}$$

with C being the number of components of the model, and π_ℓ representing the priors such that $\sum_\ell \pi_\ell = 1$.

The parameters of a GMM on the manifold of SPD matrices are estimated by an Expectation-Maximization (EM) algorithm. Specifically, the responsibility of each component ℓ is computed in the E-step as:

$$p(\ell|\boldsymbol{X}_n) = \frac{\pi_\ell \, \mathcal{N}_{\mathcal{M}}(\boldsymbol{X}_n|\Xi_\ell, \boldsymbol{\mathcal{S}}_\ell)}{\sum\limits_{i=1}^{C} \pi_i \, \mathcal{N}_{\mathcal{M}}(\boldsymbol{X}_i|\Xi_i, \boldsymbol{\mathcal{S}}_i)}, \tag{3.15}$$

$$N_\ell = \sum_{n=1}^{N} p(\ell|\boldsymbol{X}_n). \tag{3.16}$$

During the M-step, the mean $\boldsymbol{\Xi}_\ell$ is first updated iteratively until convergence for each component. The covariance tensor $\boldsymbol{\mathcal{S}}_\ell$ and prior π_ℓ are then updated using the new mean:

$$\boldsymbol{\Xi}_\ell \leftarrow \frac{1}{N_\ell} \mathrm{Exp}_{\boldsymbol{\Xi}_\ell}\left(\sum_{n=1}^{N} p(\ell|\boldsymbol{X}_n) \, \mathrm{Log}_{\boldsymbol{\Xi}_\ell}(\boldsymbol{X}_n) \right), \tag{3.17}$$

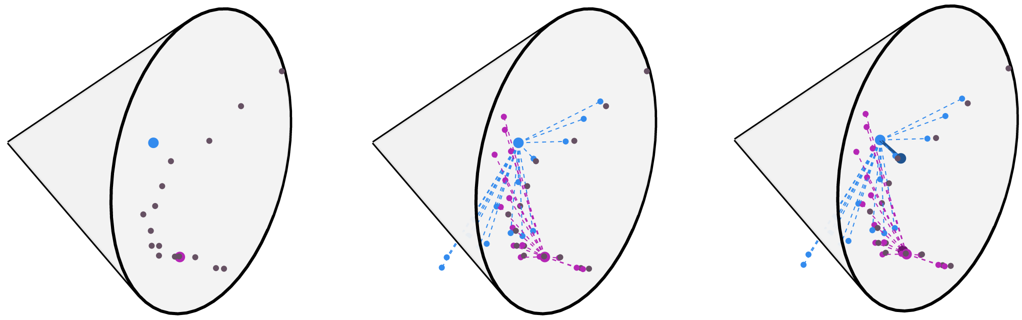
$$\boldsymbol{\mathcal{S}}_\ell \leftarrow \frac{1}{N_\ell} \sum_{n=1}^{N} p(\ell|\boldsymbol{X}_n) \, \mathrm{Log}_{\boldsymbol{\Xi}_\ell}(\boldsymbol{X}_n) \otimes \mathrm{Log}_{\boldsymbol{\Xi}_\ell}(\boldsymbol{X}_n), \tag{3.18}$$

$$\pi_\ell \leftarrow \frac{N_\ell}{N}. \tag{3.19}$$

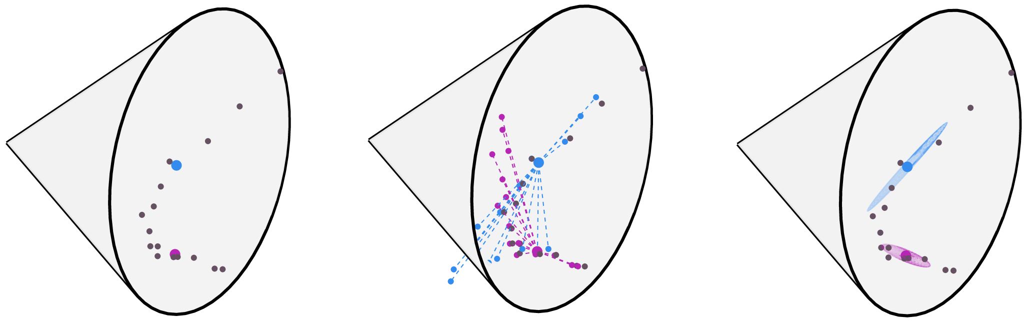
Note that the update step for the mean is equivalent to computing a weighted Fréchet mean. Figure 3.2 illustrates the M-step of one iteration of the EM algorithm to fit a GMM on the SPD manifold $\mathcal{S}_{++}^2$. The data are depicted by gray dots inside the SPD cone and the number of components of the GMM is $C = 2$.

3.4 Gaussian Mixture Regression on SPD Manifolds

GMR computes the conditional distribution $p(\boldsymbol{Y}|\boldsymbol{X})$ of the joint distribution $p(\boldsymbol{Z})$, where the sub-indices $\mathcal{X}$ and $\mathcal{Y}$ denote the sets of dimensions that span the input and output variables $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{S}_{++}^d$. We use the following block decomposition of the datapoints,



(a) Initial GMM means $\boldsymbol{\Xi}_\ell$ (b) Data in $\mathcal{T}_{\boldsymbol{\Xi}_\ell}\mathcal{S}^2_{++}$ (c) Weighted means in $\mathcal{T}_{\boldsymbol{\Xi}_\ell}\mathcal{S}^2_{++}$



(d) Updated means $\boldsymbol{\Xi}_\ell \in \mathcal{S}^2_{++}$ (e) Data in $\mathcal{T}_{\boldsymbol{\Xi}_\ell}\mathcal{S}^2_{++}$ (f) Updated GMM components

Figure 3.2 – Illustration of the M-step of one iteration of the EM algorithm to fit a GMM model ($C = 2$) on the SPD manifold $\mathcal{S}^2_{++}$. (a) Initial state. The data $\boldsymbol{X}_n \in \mathcal{S}^2_{++}$ are represented as gray dots. The initial component means $\boldsymbol{\Xi}_\ell$ with $\ell = 1, 2$ are depicted as a blue and purple dot, respectively. (b)-(d) Update of the component means. (b) The data are projected into the tangent spaces $\mathcal{T}_{\boldsymbol{\Xi}_\ell}\mathcal{S}^2_{++}$ using the logarithmic map $\text{Log}_{\boldsymbol{\Xi}_\ell}(\boldsymbol{X}_n)$. ($c$) The weighted Euclidean mean of the points (3.15), represented in dark colors, is computed in each tangent space and projected back to the manifold using the exponential map. The steps (b) and (c) are repeated until convergence (3.17). (d) Updated components means $\boldsymbol{\Xi}_\ell$. (e) Computation of the components covariance tensors $\boldsymbol{\mathcal{S}}_\ell$ in the tangent space of the means $\boldsymbol{\Xi}_\ell$ (3.18). (f) Updated means and covariances.

means and covariances:

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{X} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Y} \end{pmatrix}, \tag{3.20}$$

$$\boldsymbol{\Xi} = \begin{pmatrix} \boldsymbol{\Xi}_{\mathcal{X}\mathcal{X}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Xi}_{\mathcal{Y}\mathcal{Y}} \end{pmatrix}, \tag{3.21}$$

$$\boldsymbol{\mathcal{S}} = \left(\begin{array}{cc|cc} \boldsymbol{\mathcal{S}}^{\mathcal{X}\mathcal{X}}_{\mathcal{X}\mathcal{X}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\mathcal{S}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{X}\mathcal{X}} & \boldsymbol{0} & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\mathcal{S}}^{\mathcal{X}\mathcal{X}}_{\mathcal{Y}\mathcal{Y}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\mathcal{S}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{Y}\mathcal{Y}} \end{array}\right), \tag{3.22}$$

where we represent the 4th-order tensor by separating the components of the 3rd- and 4th-modes with horizontal and vertical bars, respectively. With this decomposition, manifold functions can be applied individually on input and output parts, for example the exponential map would be

$$\mathrm{Exp}_{\boldsymbol{\Xi}}(\boldsymbol{Z}) = \begin{pmatrix} \mathrm{Exp}_{\boldsymbol{\Xi}_{\mathcal{X}\mathcal{X}}}(\boldsymbol{X}) & \boldsymbol{0} \\ \boldsymbol{0} & \mathrm{Exp}_{\boldsymbol{\Xi}_{\mathcal{Y}\mathcal{Y}}}(\boldsymbol{Y}) \end{pmatrix}. \tag{3.23}$$

Similarly to GMR in Euclidean space [Rozo et al., 2016] and in manifolds where data are represented by vectors [Zeestraten, 2018], GMR on SPD manifold approximates the conditional distribution by a single Gaussian

$$\boldsymbol{Y}|\boldsymbol{X} \sim \mathcal{N}(\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}, \hat{\boldsymbol{\mathcal{S}}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{Y}\mathcal{Y}}), \tag{3.24}$$

where the mean $\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}$ is computed iteratively until convergence in its tangent space using

$$\boldsymbol{\Delta}_\ell = \mathrm{Log}_{\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}}(\boldsymbol{\Xi}_{\mathcal{Y}\mathcal{Y},\ell}) - \tilde{\boldsymbol{\mathcal{S}}}^{\mathcal{X}\mathcal{X}}_{\mathcal{Y}\mathcal{Y},\ell}\, \tilde{\boldsymbol{\mathcal{S}}}^{\mathcal{X}\mathcal{X}-1}_{\mathcal{X}\mathcal{X},\ell}\, \mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{\Xi}_{\mathcal{X}\mathcal{X},\ell}), \tag{3.25}$$

$$\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}} \leftarrow \mathrm{Exp}_{\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}}\Big(\sum_{\ell=1}^{C} h_\ell \boldsymbol{\Delta}_\ell\Big), \tag{3.26}$$

with h_ℓ describing the responsibilities of the GMM components in the regression, namely

$$h_\ell = \frac{\pi_\ell\, \mathcal{N}(\boldsymbol{X}|\boldsymbol{\Xi}_{\mathcal{X}\mathcal{X},\ell}, \boldsymbol{\mathcal{S}}^{\mathcal{X}\mathcal{X}}_{\mathcal{X}\mathcal{X},\ell})}{\sum_{i=1}^{C} \pi_i\, \mathcal{N}(\boldsymbol{X}|\boldsymbol{\Xi}_{\mathcal{X}\mathcal{X},i}, \boldsymbol{\mathcal{S}}^{\mathcal{X}\mathcal{X}}_{\mathcal{X}\mathcal{X},i})}. \tag{3.27}$$

The covariance $\hat{\boldsymbol{\mathcal{S}}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{Y}\mathcal{Y}}$ is then computed in the tangent space of the mean (3.26)

$$\hat{\boldsymbol{\mathcal{S}}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{Y}\mathcal{Y}} = \sum_{\ell=1}^{C} h_\ell \left(\tilde{\boldsymbol{\mathcal{S}}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{Y}\mathcal{Y},\ell} - \tilde{\boldsymbol{\mathcal{S}}}^{\mathcal{X}\mathcal{X}}_{\mathcal{Y}\mathcal{Y},\ell} \tilde{\boldsymbol{\mathcal{S}}}^{\mathcal{X}\mathcal{X}-1}_{\mathcal{X}\mathcal{X},\ell} \tilde{\boldsymbol{\mathcal{S}}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{X}\mathcal{X},\ell} + \boldsymbol{\Delta}_\ell \otimes \boldsymbol{\Delta}_\ell\right) - \hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}} \otimes \hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}, \tag{3.28}$$

where $\tilde{\boldsymbol{\mathcal{S}}}$ is the parallel transported covariance tensor

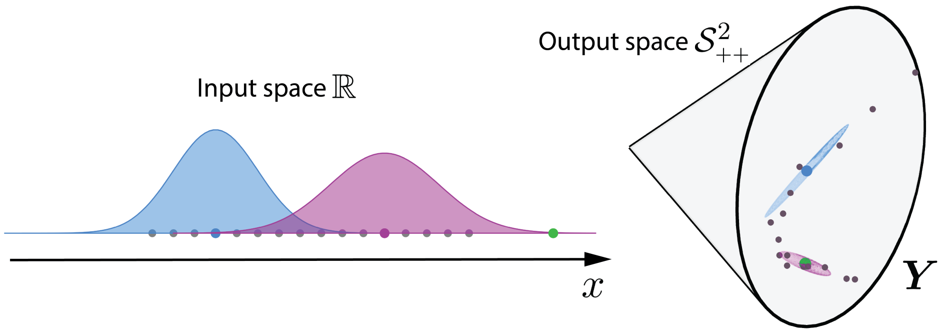
$$\tilde{\boldsymbol{\mathcal{S}}} = \Gamma_{\boldsymbol{\Xi}\to\hat{\boldsymbol{Z}}}(\boldsymbol{\mathcal{S}}) \quad \text{with} \quad \hat{\boldsymbol{Z}} = \begin{pmatrix} \boldsymbol{X} & \boldsymbol{0} \\ \boldsymbol{0} & \hat{\boldsymbol{\Xi}}_{\mathcal{O}\mathcal{O}} \end{pmatrix}. \tag{3.29}$$

This covariance can typically be used to define the controller gains of robotic systems for trajectory tracking problems, as we will see in Chapter 5. Note that the definition of the tangent space $\mathcal{T}_{\boldsymbol{\Xi}}\mathcal{M}$ (which has the structure of a Euclidean vector space) is what allows us to compute the conditional distribution above. Also in order to parallel transport a 4th-order covariance tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d\times d\times d\times d}$, we exploit the eigentensor decomposition described in Section 3.2. The approach is similar to the one presented in [Freifeld et al., 2014], where the authors showed that covariance matrices of vector elements belonging to a Riemannian manifold can be parallel transported between tangent spaces by transporting their eigenvectors. Therefore, the covariance $\boldsymbol{\mathcal{S}}$ is first converted to a 2nd-order tensor $\boldsymbol{\Sigma} \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ with $\tilde{d} = d + d(d-1)/2$ using Mandel notation with (3.8). We can then compute its eigentensors $\boldsymbol{V}_k$ with (3.10), which are used to parallel transport the covariance matrix between tangent spaces. Let $\tilde{\boldsymbol{V}}_k = \Gamma_{\boldsymbol{\Xi}\to\hat{\boldsymbol{X}}}(\boldsymbol{V}_k)$ be the k-th parallel transported eigentensor with (2.18) and λ_k the k-th eigenvalue. The parallel transported 4th-order covariance tensor is then obtained with

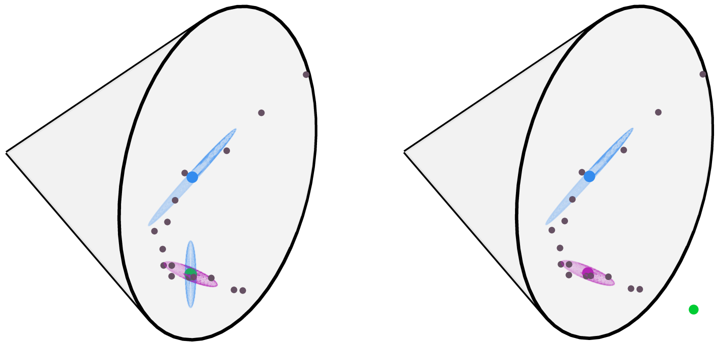
$$\Gamma_{\boldsymbol{\Xi}\to\hat{\boldsymbol{X}}}(\boldsymbol{\mathcal{S}}) = \sum_k \lambda_k \tilde{\boldsymbol{V}}_k \otimes \tilde{\boldsymbol{V}}_k. \tag{3.30}$$

Notice that the geometry-aware Gaussian mixture regression presented above can also be used in the case where only the input or the output belongs to the SPD manifold (see also the applications presented in Chapters 4). The input or output data encoded in the form of scalars or vectors are handled by defining the corresponding element either as a scalar or as a diagonal matrix, by placing the elements of the vector in the main diagonal, e.g., $\boldsymbol{X} = \text{diag}(\boldsymbol{x})$. Data on several manifolds can also be combined: One example is to build a GMM that encodes the position, orientation and manipulability ellipsoid of the end-effector of a robot, which belong to the Euclidean space $\mathbb{R}^3$, the sphere manifold $\mathcal{S}^3$ and the SPD manifold $\mathcal{S}^6_{++}$, respectively. The mapping functions and the parallel transport used in the GMR computation must therefore be adapted consequently.

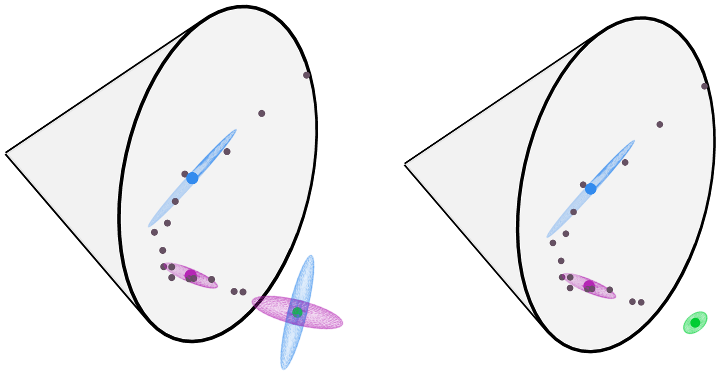
Figure 3.3 shows an example of GMR where the input x is a scalar and the output belongs to the SPD manifold $\mathcal{S}^2_{++}$. The distribution is encoded with a 2-component GMM. Figure 3.3a displays the demonstrated data as gray dots. Each point on the output space $\mathcal{S}^2_{++}$ (right) corresponds to one point in the input space $\mathbb{R}$ (left). The mean and covariance of the two GMM components are depicted in blue and purple, i.e., $\boldsymbol{\Xi}_{\mathcal{X}\mathcal{X}}$ and $\boldsymbol{\mathcal{S}}^{\mathcal{X}\mathcal{X}}_{\mathcal{X}\mathcal{X}}$ in the input space and $\boldsymbol{\Xi}_{\mathcal{Y}\mathcal{Y}}$ and $\boldsymbol{\mathcal{S}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{Y}\mathcal{Y}}$ in the output space. Note that the covariability of the inputs and outputs $\boldsymbol{\mathcal{S}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{X}\mathcal{X}}$ is not displayed. Then, given a new input x (depicted by a green dot), we compute the conditional distribution $p(\boldsymbol{Y}|x) \sim \mathcal{N}(\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}, \hat{\boldsymbol{\mathcal{S}}}^{\mathcal{Y}\mathcal{Y}}_{\mathcal{Y}\mathcal{Y}})$. The



(a) GMM model, new input x in green (*left*) and initial mean estimate $\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}$ in green (*right*)



(b) Transported covariances (c) Updated mean



(d) Transported covariances (e) Retrieved output distribution

Figure 3.3 – Illustration of GMR on the SPD manifold $\mathcal{S}_{++}^2$. (a) GMM model ($C = 2$) of the joint distribution of the inputs $x_n \in \mathbb{R}$ and the outputs $\boldsymbol{Y}_n \in \mathcal{S}_{++}^2$. The demonstrated data are depicted in gray. The new input x and the initial estimate of $\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}$ are depicted by green dots. $\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}$ is initialized as equal to the mean of the closest GMM component to x in the input space. (b) Parallel transport of the covariance tensors into the tangent space of $\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}$ to update the mean value (3.26). (c) Mean $\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}$ after convergence. (d) Parallel transport of the covariance tensors into the tangent space of $\hat{\boldsymbol{\Xi}}_{\mathcal{Y}\mathcal{Y}}$ to compute the covariance $\hat{\boldsymbol{\mathcal{S}}}_{\mathcal{Y}\mathcal{Y}}^{\mathcal{Y}\mathcal{Y}}$. ($e$) Mean and covariance of the conditional distribution $p(\boldsymbol{Y}|x)$ retrieved by the GMR.

computation of the mean $\hat{\boldsymbol{\Xi}}_{\mathcal{YY}}$ is illustrated in Figures 3.3b and 3.3c and the computation of the covariance $\hat{\boldsymbol{\mathcal{S}}}^{\mathcal{YY}}_{\mathcal{YY}}$ in Figure 3.3d. The resulting Gaussian is depicted in Figure 3.3e.

Finally, note that Mandel notation can also be used for the GMM/GMR operations, notably to avoid adding extra zeros in the implementation when data belonging to different manifolds are treated jointly. We can write (3.20), (3.21) and (3.22) as

$$\begin{aligned}
\boldsymbol{z} &= \begin{pmatrix} \mathrm{mvec}(\boldsymbol{X}) \\ \mathrm{mvec}(\boldsymbol{Y}) \end{pmatrix}, \\
\boldsymbol{\mu} &= \begin{pmatrix} \mathrm{mvec}(\boldsymbol{\Xi}_{\mathcal{XX}}) \\ \mathrm{mvec}(\boldsymbol{\Xi}_{\mathcal{YY}}) \end{pmatrix}, \\
\boldsymbol{\Sigma} &= \begin{pmatrix} \mathrm{mmat}(\boldsymbol{\mathcal{S}}^{\mathcal{XX}}_{\mathcal{XX}}) & \mathrm{mmat}(\boldsymbol{\mathcal{S}}^{\mathcal{YY}}_{\mathcal{XX}}) \\ \mathrm{mmat}(\boldsymbol{\mathcal{S}}^{\mathcal{XX}}_{\mathcal{YY}}) & \mathrm{mmat}(\boldsymbol{\mathcal{S}}^{\mathcal{YY}}_{\mathcal{YY}}) \end{pmatrix}.
\end{aligned}$$

The manifold operations are similarly adapted, e.g., the exponential map (3.23) and the logarithmic map become

$$\mathrm{Exp}_{\boldsymbol{\mu}}(\boldsymbol{z}) = \begin{pmatrix} \mathrm{mvec}\left(\mathrm{Exp}_{\boldsymbol{\Xi}_{\mathcal{XX}}}(\boldsymbol{X})\right) \\ \mathrm{mvec}\left(\mathrm{Exp}_{\boldsymbol{\Xi}_{\mathcal{YY}}}(\boldsymbol{Y})\right) \end{pmatrix}, \quad \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{z}) = \begin{pmatrix} \mathrm{mvec}\left(\mathrm{Log}_{\boldsymbol{\Xi}_{\mathcal{XX}}}(\boldsymbol{X})\right) \\ \mathrm{mvec}\left(\mathrm{Log}_{\boldsymbol{\Xi}_{\mathcal{YY}}}(\boldsymbol{Y})\right) \end{pmatrix},$$

and the following Gaussian distribution can then be used for the GMM and GMR implementations,

$$\mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^{\tilde{d}}|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{z})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{z})}.$$

3.5 Importance of Geometry-awareness for Learning SPD Matrices

In the previous sections, we introduced geometry-aware GMM and GMR formulations for SPD matrices. In this section, we show that the geometry-awareness of our formulations is crucial for successfully learning SPD matrices in addition to providing an appropriate mathematical treatment of the problem. To do so, we evaluate the proposed learning formulation compared to two Euclidean frameworks that ignores the geometry of the SPD manifold.

3.5.1 Importance of Geometry-awareness for Encoding SPD Matrices

We start our analysis by considering the computation of GMM on the SPD manifold. To do so, we consider an example where a 1-state GMM (i.e. a Gaussian) is trained from a dataset composed of points equally spaced along a geodesic on the SPD manifold $\mathcal{S}^2_{++}$.

Note that, such a dataset on the manifold is equivalent to a dataset of points equally spaced along a segment of line in a Euclidean space. We first compare the proposed geometry-aware GMM formulation with a GMM acting in the Euclidean space, i.e., ignoring that the data belong to the set of SPD matrices. Secondly, we compare the proposed approach with a Cholesky-based Euclidean GMM, where a Euclidean GMM is fitted to the lower-triangular matrices $\boldsymbol{L}$ obtained from the Cholesky decomposition $\boldsymbol{X} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$, where $\boldsymbol{X} \in \mathcal{S}_{++}^{2}$.

Figure 3.4 displays the mean and covariance of the geometry-aware and Euclidean GMM, as well as the mean of the Cholesky-based Euclidean GMM. Equivalently to the mean of points equally spaced along a segment of line in a Euclidean space, the mean $\Xi_{\mathcal{M}}$ of the geometry-aware GMM component lies in the middle of the geodesic. Moreover, similarly to the covariance of points along a line in the Euclidean space, the geometry-aware covariance of the GMM component, lying in the tangent space of the mean, is elongated along the geodesic direction and close-to-zero along the other dimensions. As previously observed in Figure 3.1, the Euclidean mean Ξ_{E} differs from the geometry-aware mean $\Xi_{\mathcal{M}}$. Moreover, the covariance of the GMM component of the geometry-aware and Euclidean approaches are not equivalent. The Euclidean covariance is computed by considering the Euclidean deviations of the data compared to their mean in the ambient space, while our geometry-aware approach computes the covariance from the deviation of the SPD data projected in the tangent space of their mean. Overall, the Euclidean formulation ignores the geometry of the SPD manifold and considers it as a flat space. Therefore, it misinterprets the configuration of the dataset, ignoring that the data are following a geodesic on the manifold, which would be equivalent to following a straight line in the Euclidean space.

Both geometry-aware and Cholesky-based Euclidean approaches obtain similar means of the GMM component. This is due to the fact that the Euclidean mean computed using the Cholesky decomposition is a good approximation of the mean computed on $\mathcal{S}_{++}^{d}$ if the SPD data are close enough to each other. However, the covariances of the GMM components of both approaches are not equivalent. Indeed, the covariance of our geometry-aware approach is computed using the SPD data projected in the tangent space of the mean, while that of the Cholesky-based Euclidean GMM corresponds to the covariance of the elements of the vectorized Cholesky decomposition, which ignores the geometry of the SPD manifold.

An other important aspect in GMM and GMR is the computation of the responsibilities of each components. In GMM, those are computed during the E-step and drive the computation of the weighted mean and covariances in the M-step (see (3.15)). Moreover, for GMR, they determine the influence of each GMM component during the regression process (see (3.27)). In the following example, we consider a 2-component GMM with isotropic covariances, as displayed in Figure 3.5a. We compare the influence of each GMM component in the SPD space based on the proposed geometry-aware framework

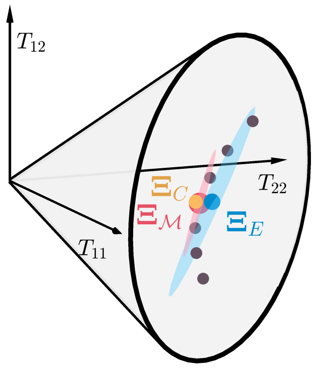


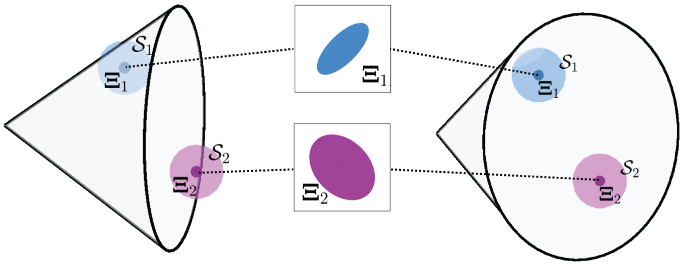
Figure 3.4 – Importance of geometry in GMM formulation. The demonstrated data, equally spaced along a geodesic, are depicted by gray dots. The geometry-aware and Euclidean GMM components are depicted in red and blue, respectively. The center of the Cholesky-based GMM component is depicted by a yellow dot. Note that its covariance is not depicted, as it corresponds to the covariance of the elements of the vectorized Cholesky decomposition.

against the responsibilities computed from the Euclidean and Cholesky-based Euclidean formulations. In order to compare equivalent distributions in the SPD and Cholesky spaces, the covariances of the Cholesky-based GMM components were computed by sampling from the geometry-aware Gaussian distributions.

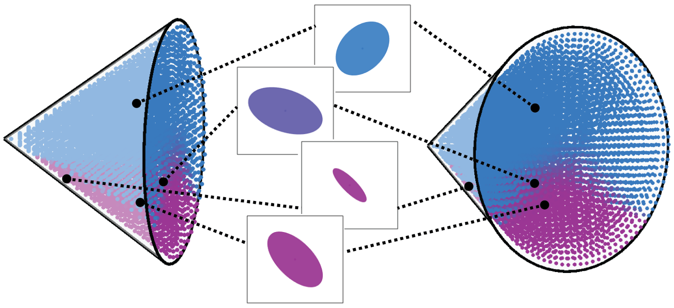
Figures 3.5b, 3.5c and 3.5d show the responsibility of each GMM component over the SPD cone for the geometry-aware, Euclidean and Cholesky-based Euclidean formulations, respectively. The points are colored with proportion of blue and magenta corresponding to their posterior probabilities of having been generated by the corresponding GMM components of Figure 3.5a. Therefore, points that could have been generated by either component appear in purple. For each formulation, four SPD matrices are depicted in the form of ellipsoids whose color also corresponds to their probability of belonging to each GMM component. As expected, the Euclidean formulation separates the SPD cone in two parts along a soft plane-shaped border located at equal Euclidean distance from the two components means. However, this separation does not consider the geometry of the space. For example, SPD matrices whose main axis is elongated in the same direction of that of one GMM component mean may have high probability of belonging to the other component. As an example, the third ellipsoid is oriented in the same direction as Ξ_2 but has a very high probability of belonging to Ξ_1 according to the Euclidean GMM. Unlike the Euclidean approach, both geometry-aware and Cholesky-based formulation draw a curved soft border between the SPD matrices with high probability to have been generated by one or the other Gaussian. However, solely the geometry-aware formulation fully considers the orientation of the ellipsoids to determine the component from which they were likely generated (observe also that the purple zone reaches the tip of the cone in the geometry-aware case).

3.5.2 Importance of Geometry-awareness for Retrieving SPD Matrices

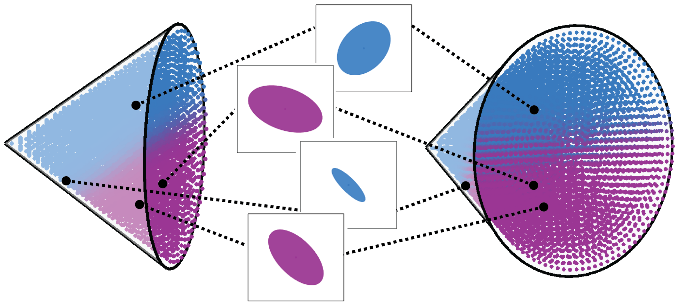
In the previous section, we showed the importance of geometry-awareness to learn a GMM on the SPD manifold and to perform GMR when the inputs of the model lie on the SPD manifold. In this section, we evaluate the proposed learning formulation to retrieve SPD



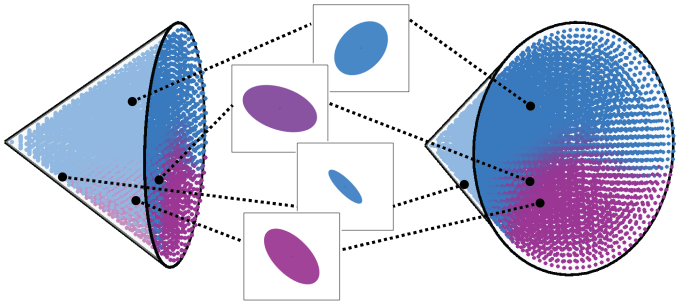
(a) GMM



(b) Geometry-aware responsibilities



(c) Euclidean responsibilities



(d) Cholesky-based Euclidean responsibilities

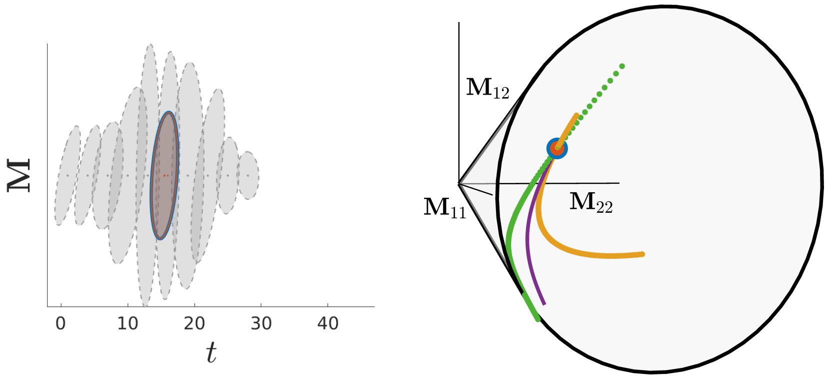
Figure 3.5 – Importance of geometry in the computation of the responsabilities of GMM components. (*a*) 2-states GMM with isotropic covariance matrices. (*b*)-(*d*) Responsibilities of each GMM component over the SPD cone. The points on the cones are colored with proportion of blue and magenta corresponding to the posterior probabilities of having been generated by each GMM component. Note that the responsibility of the two components for some of these points differ in function of the approach.

matrices, i.e., when the outputs of the model belong to the SPD manifold. Specifically, we consider an example where a time-varying sequence of 2-dimensional manipulability ellipsoids $\boldsymbol{M} \in \mathcal{S}_{++}^{2}$ is learned from demonstrations. In robotics, manipulability ellipsoids are kinetostatic performance measures that indicate the preferred directions in which force or velocity control commands may be performed at a given joint configuration of the robot. Manipulability ellipsoids and related applications are covered in details in the Chapters 4, 5 and 6 of this thesis.

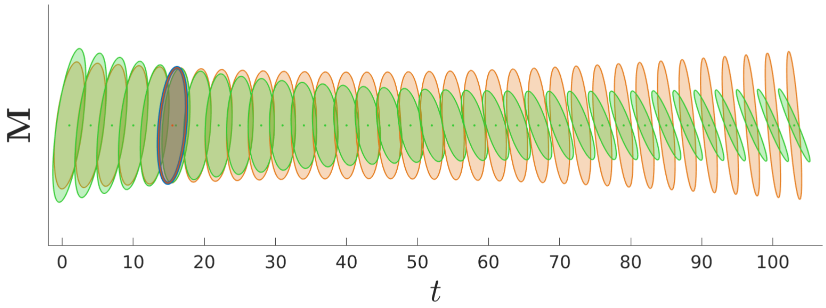
In this example, we encode a distribution of manipulability ellipsoids with a GMM acting in the Euclidean space and we then retrieve desired manipulability ellipsoids via the corresponding GMR. To ensure the validity of the desired manipulability ellipsoids, GMM and GMR are performed on lower triangular matrices $\boldsymbol{L}$ obtained via Cholesky decomposition. Thus, the positive-definiteness of the desired manipulability ellipsoids computed as $\hat{\boldsymbol{M}} = \hat{\boldsymbol{L}}\hat{\boldsymbol{L}}^{\mathsf{T}}$ is guaranteed, where $\hat{\boldsymbol{L}}$ is the estimated GMR output. Note that this property is not guaranteed in the case where GMM and GMR acting in the Euclidean space is applied directly to the manipulability ellipsoids $\boldsymbol{M}$. Therefore, we do not consider this approach in the comparison as the desired matrices $\hat{\boldsymbol{M}}$ may not be manipulability ellipsoids, i.e., not be SPD, in some cases.

Figure 3.6 compares the proposed approach (Sections 3.3, 3.4) and the manipulability learning using GMM/GMR acting in Euclidean space through Cholesky decomposition. The demonstration consists of a time series of changing manipulability ellipsoids. For each approach, a 1-state GMM is trained and a reproduction is carried out for a longer time period than the demonstration using GMR. As observed in Section 3.5.1, both geometry-aware and Cholesky-based Euclidean approaches obtain similar means of the GMM component (see Figures 3.6a, 3.6b). This is due to the fact that the Euclidean mean computed using the Cholesky decomposition is a good approximation of the mean computed on $\mathcal{S}_{++}^{d}$ if the SPD data are close enough to each other. However, as discussed in the Section 3.5.1, the covariances of the GMM components of both approaches are not equivalent.

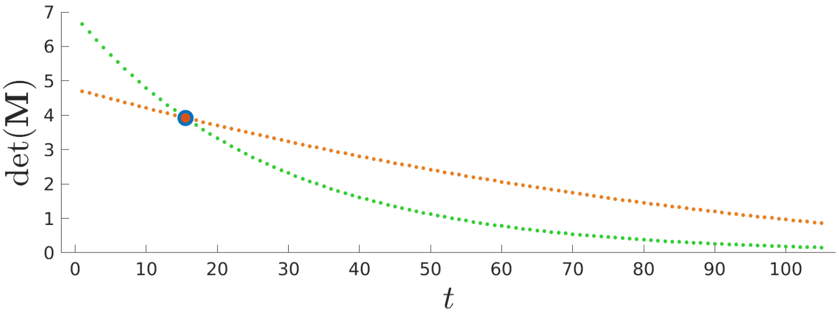
The manipulability ellipsoids profiles retrieved by the geometry-aware and Cholesky-based Euclidean GMR are similar around the mean of the GMM component, but diverge when moving away from it (see Figure 3.6c). This is because the estimated output in Euclidean space is only a valid approximation for input data lying close to the mean. In contrast, our approach is able to extrapolate the rotating behavior of the demonstrated manipulability ellipsoids as the recovered trajectory follows a geodesic on the SPD manifold (see Figure 3.6b). Note that this is the equivalent to following a straight line in Euclidean space, which is the expected result of a trajectory computed via Gaussian conditioning. This behavior is obtained by parallel transporting the GMM covariances to the tangent space of the mean of the estimated conditional distribution of GMR (3.29). Therefore, the Cholesky-based Euclidean GMR does not recover a trajectory following a geodesic on the manifold, leading to inconsistent extrapolated manipulability



(a) Demonstrations and GMM means (b) Retrieved profiles in $\mathcal{S}^2_{++}$



(c) Retrieved manipulability profiles over time



(d) Determinant of retrieved manipulability profiles over time

Figure 3.6 – Importance of geometry to retrieve SPD matrices via GMR. (*a*) Demonstrated data (depicted in light gray), and mean of the GMM component for the geometry-aware and Cholesky-based Euclidean approaches (overlapping blue and red ellipsoids, respectively). (*c*) Manipulability profiles retrieved by the geometry-aware and Cholesky-based Euclidean GMR, shown as green and orange ellipses, respectively. (*d*) Determinant of the retrieved manipulability profiles over time. (*b*) Mean of the GMM component and estimated profiles in the cone of SPD matrices. The manipulability profile obtained by our approach, shown in green, follows a geodesic. The profile obtained by the Cholesky-based Euclidean framework is depicted by the orange curve and does not follow a geodesic on the manifold. The geodesic containing the mean of the Cholesky-based Euclidean GMM, being a geometrically valid trajectory (depicted in purple), does not correspond to the trajectory obtained with the Cholesky-based Euclidean framework. Thus, the Cholesky-based Euclidean approach is geometrically flawed.

ellipsoids. This can also be observed in the evolution of the determinant of the retrieved manipulability profile (see Figure 3.6d). The determinant of the manipulability ellipsoids recovered by the Cholesky-based Euclidean GMR decreases linearly in a slower pace than the geometry-aware profile. This is due to the fact that the ellipsoids of the Euclidean approach grow along the vertical axis over time.

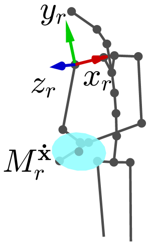
The reported results show that our geometry-aware approach accurately reproduces the behavior of the demonstrated data, and therefore provides a mathematically sound method for learning and retrieving SPD matrices. Note that similar behaviors are observed for GMM with any number of states, the number $C = 1$ was chosen here to facilitate the visualization of the results.

3.6 Conclusion

This chapter presented a novel LfD framework for encoding and retrieving SPD matrices. We exploited tensor representation and Riemannian manifolds to build a geometry-aware probabilistic learning model and showed the importance of geometry-awareness for learning on the SPD manifold. The presented approach is flexible as it allows us to easily learn combinations of data in the form of scalar, vector and SPD matrices, as well as data on different manifolds. It is extensible to other matrix-valued Riemannian manifolds, as long as the eigentensors decomposition of the covariance tensor can be computed. Applications of our learning approach are showcased in the next chapter of this thesis.

4 Learning Task-dependent Requirements via Manipulability Analysis

In the previous chapter, we introduced a geometry-aware probabilistic framework to learn symmetric positive definite (SPD) matrices from demonstrations. This chapter focuses on a specific application of the aforementioned approach, namely learning sequences of *manipulability ellipsoids* — which captures a posture-dependent ability to perform motion and exert forces along different task directions — from human demonstrations.



To do so, this chapter first presents a thorough manipulability-based analysis of human arms motion patterns in industrial activities. Secondly, we discuss how manipulability patterns can be transferred to robots via the probabilistic model presented in Chapter 3.

Publication note

The material presented in this chapter is adapted from the following publications:

- Rozo, L., Jaquier, N., Calinon, S., and Caldwell, D. G. (2017). Learning manipulability ellipsoids for task compatibility in robot manipulation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3183–3189;
- Jaquier, N., Rozo, L., Caldwell, D. G., and Calinon, S. (2020b). Geometry-aware manipulability learning, tracking and transfer. *Intl. Journal of Robotics Research*;
- Jaquier, N., Rozo, L., and Calinon, S. (2020c). Analysis and transfer of human movement manipulability in industry-like activities. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*

Supplementary material

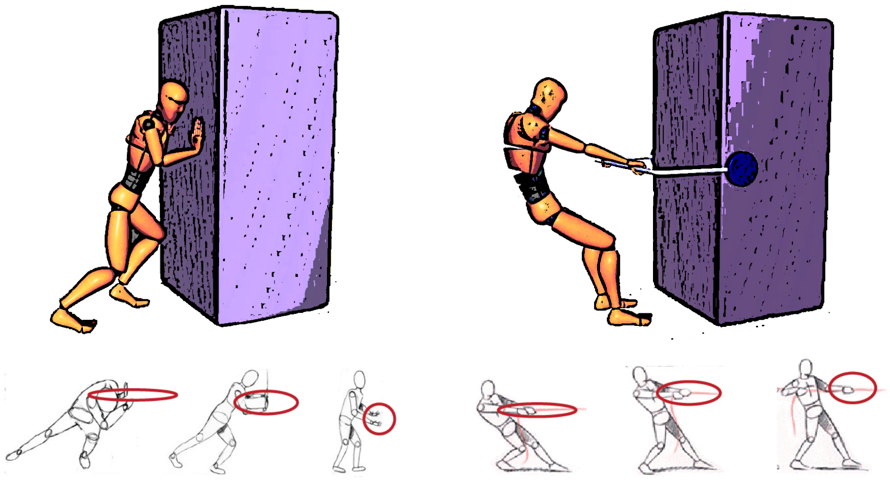
Videos related to this chapter are available at:

https://sites.google.com/view/manipulability.

Source code

Source codes related to this chapter are available at:

https://github.com/NoemieJaquier/Manipulability.



(a) Pushing task (b) Pulling task

Figure 4.1 – Illustration of pushing and pulling tasks for which the posture of the humans significantly influences their ability to carry out the task.

4.1 Introduction

Humans exhibit outstanding learning, planning and adaptation capabilities while performing different types of industrial tasks. Given some knowledge about the task requirements, humans are able to plan their limbs motion in anticipation of the execution of specific skills. For example, when an operator needs to drill a hole on a surface, the posture of her limbs varies to guarantee a stable configuration that is compatible with the drilling task specifications, e.g. exerting a force orthogonal to the surface.

In general, when performing manipulation tasks, we naturally put our limbs in a posture that best allows us to carry out the task at hand given specific workspace constraints (see also Figure 4.1). This posture adaptation alters the motion and strength characteristics of our arms so that they are compatible with specific task requirements. For example, human arm kinematics plays a central role when we plan point-to-point reaching movements, where joint trajectory patterns arise as a function of visual targets [Morasso, 1981], indicating that task requirements lead to arm posture variations. This insight was also observed in more complex situations, where not only kinematic but also other biomechanic factors affect the task planning [Cos et al., 2011]. For instance, Sabes and Jordan [1997] observed that our central nervous system plans arm movements considering its directional sensitivity, which is directly related to the arm posture. This allows humans to be mechanically resistant to potential perturbations. Interestingly, directional preferences of human arm movements are characterized by a tendency to exploit interaction torques for movement production at the shoulder or elbow, indicating that the preferred directions are largely determined by biomechanical factors [Dounskaia et al., 2014].

Roboticists have also investigated the impact of robot posture on manipulation tasks that involve pushing, pulling and reaching. It is well known that by varying the posture of a robot, we can change the optimal directions for motion generation or force exertion. This has direct implications in hybrid control, since the controller capability can be fully realized when the optimal directions for controlling velocity and force coincide with those dictated by the task [Chiu, 1987]. In this context, the so-called manipulability ellipsoid [Yoshikawa, 1985b] serves as a geometric descriptor that, given a joint configuration, indicates the capability to arbitrarily perform motion and exert a force along the different task directions.

Manipulability ellipsoids have been used to analyze the coordination of the human arm during reaching-to-grasp tasks for designing ergonomic environments [Jacquier-Bret et al., 2012], and to study the swing phase of human walking motion [Miripour Fard, 2019]. However, analyses of the human arm manipulability remain limited to few simple movements. Moreover, most of the conducted studies focus on the evolution of the manipulability volume and isotropy. In contrast, considering the direction of the major axis of manipulability ellipsoids has been proved useful in several human movement analysis works, notably in exoskeletons design and control. Goljat et al. [2017] used the shape of the muscular manipulability of the human arm for controlling arm exoskeletons. They computed a varying support based on the main direction of the user's force manipulability. Inspired by human walking studies, Kim et al. [2010] proposed an energy-efficient gait pattern for leg exoskeletons which aligns the direction of motion with the major axis of the dynamic manipulability ellipsoid. While the major axis of the ellipsoid provides some information about the arm movement, the importance of the ellipsoid shape should not be neglected, as a low dexterity in motion along a specific axis is closely related to a high flexibility in force along the very same direction [Lee, 1989].

Other geometric descriptors have been proposed in the literature to evaluate the velocity or force performance of robots at a given joint configuration. In contrast to manipulability ellipsoids that do not fully account for boundary limits in the space of joint velocities or torques, manipulability polytopes provide a linear estimate of the exact joint constraints in task space [Chiacchio et al., 1997; Lee, 1997]. Moreover, Ajoudani et al. [2015] introduced the concept of stiffness feasibility region (SFR) to represent the non-polytopic boundary where the realization of a desired Cartesian stiffness matrix is feasible. While the polytope approaches provide a more accurate estimate of the velocity or force generation capabilities of the robot compared to manipulability ellipsoids, their calculation is computationally expensive. SFR is a particular Cartesian stiffness descriptor and therefore does not generalize to other robot control settings. Manipulability ellipsoids are easy to compute, while representing an intuitive estimate of the robot ability to perform velocities, accelerations or exert forces along the different task directions.

In this chapter, we analyze single and dual-arm manipulability of human movements during the execution of industry-like activities from a geometry-aware perspective. To do so, we use kinematic data records of several participants performing various activities such as screwing and load carrying [Maurice et al., 2019]. Moreover, we consider an important characteristic of manipulability ellipsoids that was often overlooked in the literature, namely, the fact that they lie on the manifold of SPD matrices (see Section 4.2 for a short background). We exploit differential geometry to statistically study the manipulability profile of human movements. The mean and variance of the ellipsoids provide more information about human motion than the classical manipulability indices related to the ellipsoids volume and isotropy, as explained in Section 4.3. Finally, we introduce the novel idea that the observed task-dependent patterns can be learned by robots as manipulability requirements when executing similar tasks, bypassing the complexity of kinematic mapping approaches. In Section 4.4, we address the problem of learning manipulability-based posture variation from a robot-learning-from-demonstrations perspective by exploiting the framework introduced in Chapter 3.

4.2 Background: Manipulability Ellipsoids

Velocity and force manipulability ellipsoids introduced in [Yoshikawa, 1985b] are kinetostatic performance measures of robotic platforms. They indicate the preferred directions in which force or velocity control commands may be performed at a given joint configuration. More specifically, the velocity manipulability ellipsoid describes the characteristics of feasible Cartesian motion corresponding to all the unit norm joint velocities. The velocity manipulability of an n-DoF robot can be found by using the kinematic relationship between task velocities $\dot{\boldsymbol{x}}$ and joint velocities $\dot{\boldsymbol{q}}$,

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{4.1}$$

where $\boldsymbol{q}\in\mathbb{R}^n$ and $\boldsymbol{J}\in\mathbb{R}^{6\times n}$ are the joint position and Jacobian of the robot, respectively. Moreover, consider the set of joint velocities of unit norm $\|\dot{\boldsymbol{q}}\|^2=1$ describing the points on the surface of a hypersphere in the joint velocity space, which is mapped into the Cartesian velocity space $\mathbb{R}^6$ with[1]

$$\|\dot{\boldsymbol{q}}\|^2 = \dot{\boldsymbol{q}}^{\mathsf{T}}\dot{\boldsymbol{q}} = \dot{\boldsymbol{x}}^{\mathsf{T}}(\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}})^{-1}\dot{\boldsymbol{x}}, \tag{4.2}$$

by using the least-squares inverse kinematics solution $\dot{\boldsymbol{q}}=\boldsymbol{J}^{\dagger}\dot{\boldsymbol{x}}=\boldsymbol{J}^{\mathsf{T}}(\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}})^{-1}\dot{\boldsymbol{x}}$. Equation (4.2) represents the robot manipulability in terms of motion, indicating the flexibility of the manipulator in generating velocities in Cartesian space.[2]

In the literature, the velocity manipulability ellipsoid is usually defined as $(\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}})^{-1}$, whose principal axes are defined by the eigenvectors whose length is the inverse of the square root of the corresponding eigenvalues [Chiu, 1987]. For the sake of consistency, we here

---

[1] Scaling of the joint velocities may be used to reflect actuator properties.

[2] Dually, the force manipulability ellipsoid can be computed from the static relationship between joint torques and Cartesian forces [Yoshikawa, 1985b].

use an alternative definition of the velocity manipulability ellipsoid given by $\boldsymbol{M}^{\dot{\boldsymbol{x}}} = \boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}$. So, its major axis is aligned to the eigenvector associated with the maximum eigenvalue $\lambda_{\max}^{\boldsymbol{M}^{\dot{\boldsymbol{x}}}}$ of $\boldsymbol{M}^{\dot{\boldsymbol{x}}}$, whose length equals the square root of $\lambda_{\max}^{\boldsymbol{M}^{\dot{\boldsymbol{x}}}}$. Thus, the interpretation and representation of the manipulability ellipsoid from the corresponding matrix are facilitated. Note that the major axis of $\boldsymbol{M}^{\dot{\boldsymbol{x}}} = \boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}$ indicates the direction in which the greater velocity can be generated, that in turn is the direction along which the robot is more sensitive to perturbations. This occurs due to the principal axes of the force manipulability being aligned with those of the velocity manipulability, with reciprocal lengths (eigenvalues) due to the velocity-force duality (see [Chiu, 1987] for details).

Other forms of manipulability ellipsoids exist, such as the dynamic manipulability [Yoshikawa, 1985a], which gives a measure of the ability of performing end-effector accelerations along each task-space direction for a given set of joint torques. This has shown to be useful when the robot dynamics cannot be neglected in highly dynamic manipulation tasks [Chiacchio et al., 1991b]. Recent works have extended this measure to analyze the robot capacity to accelerate its center of mass for locomotion stability [Azad et al., 2017; Gu et al., 2015], showing the applicability of the aforementioned tools beyond robotic manipulation.

As mentioned previously, any manipulability ellipsoid $\boldsymbol{M}$ belongs to the set of SPD matrices $\mathcal{S}_{++}^{D}$ which describe the interior of a convex cone. Consequently, we must consider this particular aspect to properly analyze and learn manipulability profiles. For introductions to the corresponding Riemannian manifold and to the computation of manipulability statistics (i.e., statistics on the SPD manifold), we refer the reader to Sections 2.2.4 and 3.2, respectively.

4.3 Manipulability Analysis

Analyzing the manipulability of the human arms during various tasks may be relevant to define desired manipulability ellipsoids of robots. The manipulability profile of a user while performing a task may provide relevant information about task planning and motion generation. For example, the main axis of the ellipsoid may indicate future directions of motion, while a small manipulability may reveal a lack of velocity or force control of an operator along specific directions. These aspects may notably be exploited to better design and control exoskeletons and ergonomic devices. In this section, we propose a detailed analysis of human arm(s) manipulability in industry-like activities by exploiting the aforementioned mathematical tools.

4.3.1 Data Description

For our analysis, we use the industry-oriented dataset presented in [Maurice et al., 2019], which contains the whole-body posture data of 13 participants executing various industry-related activities. Each participant executed the tasks in 5 consecutive trials for 3 predefined sequences. In this chapter, we consider 3 screwing motions and 2 carrying tasks provided in the database. A screwing task consists of taking a screw and a bolt from a 75 cm-high table, walking to a shelf and screwing (with bare hands) at a specific height. The screwing movements realized at heights of 60 cm, 115 cm and 175 cm are denoted by SL, SM and SH for screw-low, -middle and -high, respectively. A carrying task involves taking a load from a 55 cm-high table, walking to a shelf and putting the load on it. Loads of 5 kg and 10 kg are placed in shelves of 20 cm and 110 cm high, respectively. The corresponding tasks are denoted by C5 and C10. The participants freely adapt their posture for each activity, as no explicit instructions were given for task execution. Therefore, we expect the arms manipulability to reflect the features of the natural motion of the participants.

The labels of the motions in all the trials are provided along with the dataset, including the general and detailed posture, as well as the current action labeled by 3 independent annotators. In this chapter, we study the human arm(s) manipulability according to the current action for the aforementioned carrying and screwing tasks. For each trial, we first identify the frames corresponding to each activity based on the labels and the order of activities defined in the corresponding sequence. Then, we select the frames corresponding to a subset of actions for each activity. The idea is to consider the motions corresponding to the relevant activities that are reproduced by all the participants. Therefore, we do not examine particular cases, e.g. when a screw falls on the ground and the participant needs to pick it up. The subset of actions considered for a screwing motion is composed of reaching (Re), picking (Pi), carrying (Ca), placing (Pl), fine manipulation (Fm), screwing (Sc) and releasing (Rl) the screw and the bolt. Concerning the carrying motion, we analyze the subset of actions composed of picking (Pi), carrying (Ca) and placing (Pl) the load.

The single- and dual-arm manipulability ellipsoids are computed for each time step of the different actions of the screwing and carrying tasks for 15 trials of 13 participants. The computation of the manipulability ellipsoids from the whole-body position and orientation data is described next.

4.3.2 Human Manipulability Computation

As the manipulability is a function of the Jacobian, we need a kinematic model of the human arm to compute its manipulability ellipsoid. In this chapter, we use the identification method for anthropomorphic arm kinematics proposed in [Ding and Fang,

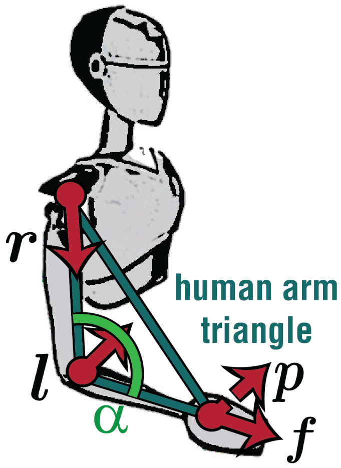


Figure 4.2 – Human arm triangle model of 5 parameters. The triangle vertex are located at the center of the shoulder, elbow and wrist joints.

2013; Fang and Ding, 2013]. The typical anthropomorphic arm model, where the arm is regarded as a 7-DoF manipulator, is exploited here along with the concept of human arm triangle, the latter used for joint-to-task space mapping. Thus, the joint angles and the corresponding Jacobian can be computed from the position and orientation of the wrist in task space, which in our case is given in the database.

Specifically, the human arm triangle model, shown in Figure 4.2, is defined by 5 parameters: the unit direction vector of the upper arm $\boldsymbol{r}$, the unit normal vector of the human arm triangle space $\boldsymbol{l}$, the angle between the upper and lower arm α, the unit normal vector of the plane of the palm (pointing outward the palm) $\boldsymbol{p}$, and the unit direction vector of the fingers $\boldsymbol{f}$. The parameters $\{\boldsymbol{r}, \boldsymbol{l}, \alpha\}$ can be inferred from the wrist position in task space (see [Ding and Fang, 2013] for details), while the parameters $\{\boldsymbol{p}, \boldsymbol{f}\}$ directly represent the wrist orientation. Moreover, it has been shown that the space spanned by the set of parameters $\{\boldsymbol{r}, \boldsymbol{l}, \alpha, \boldsymbol{p}, \boldsymbol{f}\}$ has a one-to-one relation with the joint space spanned by the seven joints $\{q_1 \dots q_7\}$ of the anthropomorphic arm model. The formulas for the mappings $\{\boldsymbol{r}, \boldsymbol{l}, \alpha\} \rightarrow \{q_1 \dots q_4\}$ and $\{\boldsymbol{p}, \boldsymbol{f}\} \rightarrow \{q_5 \dots q_7\}$ are given in [Ding and Fang, 2013] and [Fang and Ding, 2013], respectively. Also, the formulas of the forward and inverse kinematic mappings based on the human arm triangle model are reported in Appendix A.1.

This model allows us to compute the human arm Jacobian $\boldsymbol{J}$. The arm velocity manipulability ellipsoid is then computed as $\boldsymbol{M}^{\dot{x}} = \boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}$. For dual-arm manipulation tasks, such as carrying, we are interested in the manipulability ellipsoid of the dual-arm system. In this case, the set of joint velocities of constant unit norm $\|\dot{\boldsymbol{q}}_d\| = \|(\dot{\boldsymbol{q}}_l^{\mathsf{T}}, \dot{\boldsymbol{q}}_r^{\mathsf{T}})^{\mathsf{T}}\| = 1$ is mapped to the Cartesian velocity space $\dot{\boldsymbol{x}}_d = (\dot{\boldsymbol{x}}_l^{\mathsf{T}}, \dot{\boldsymbol{x}}_r^{\mathsf{T}})^{\mathsf{T}}$ through

$$\|\dot{\boldsymbol{q}}_d\|^2 = \dot{\boldsymbol{q}}_d^{\mathsf{T}}\dot{\boldsymbol{q}}_d = \dot{\boldsymbol{x}}_d^{\mathsf{T}}(\boldsymbol{G}_d^{\dagger\mathsf{T}}\boldsymbol{J}_d\boldsymbol{J}_d^{\mathsf{T}}\boldsymbol{G}_d^{\dagger})^{-1}\dot{\boldsymbol{x}}_d, \tag{4.3}$$

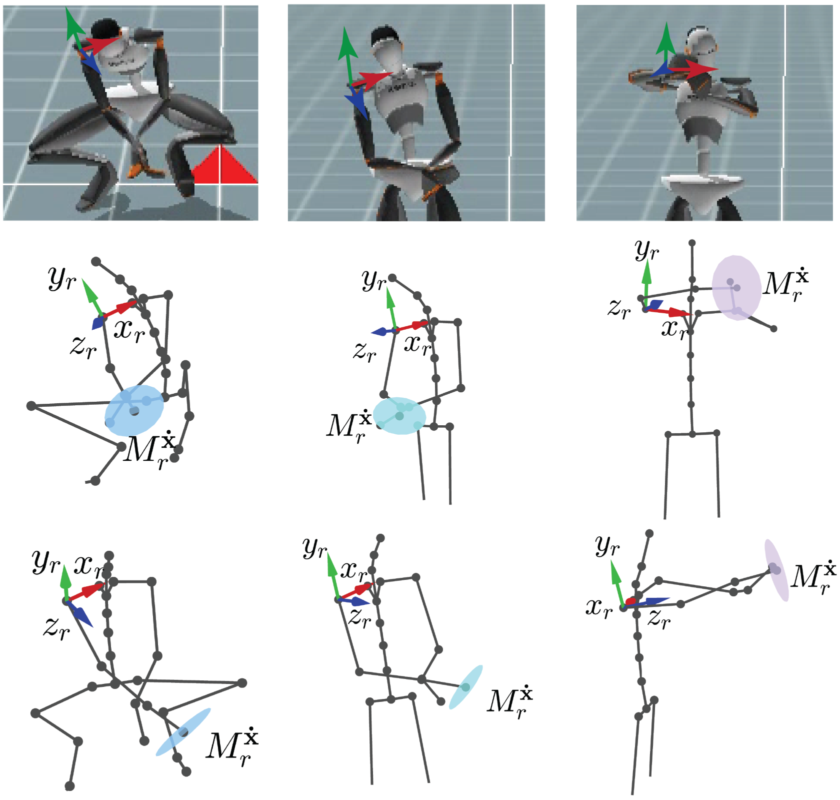
with Jacobian $\boldsymbol{J}_d = \mathrm{diag}(\boldsymbol{J}_l, \boldsymbol{J}_r)$, grasp matrix $\boldsymbol{G}_d = (\boldsymbol{G}_l, \boldsymbol{G}_r)$ and indices l and r denoting the left and right arm, respectively. Therefore, the dual-arm velocity manipulability is given by $\boldsymbol{M}_d^{\dot{\boldsymbol{x}}} = \boldsymbol{G}_d^{\dagger\top}\boldsymbol{J}_d\boldsymbol{J}_d^{\top}\boldsymbol{G}_d^{\dagger}$ [Chiacchio et al., 1991a]. Note that we assume two independent kinematic chains for the arms in the computation of $\boldsymbol{J}_d$. Moreover, the system is modeled under the assumption that the arms are holding a rigid object with a tight grasp. The force manipulability ellipsoid is the inverse of the velocity manipulability, i.e. $\boldsymbol{M}^{\boldsymbol{F}} = (\boldsymbol{M}^{\dot{\boldsymbol{x}}})^{-1}$.

4.3.3 Analysis

For screwing motions, we study the single-arm velocity manipulability ellipsoids. Figure 4.3 shows the posture of a participant while executing this task at different heights, along with the corresponding right-arm velocity manipulability ellipsoid. We observe that the shape of the ellipsoids is similar for the three screwing motions, regardless of the specified height. Namely, the ellipsoids shrink along the hand axis and isotropically elongate along the other directions. This indicates a high precision along the hand axis coupled with a high capability of motion on the orthogonal plane, where the hand is moving to execute the rotative screwing motion. This shows that the human arm manipulability is being adapted to the task requirements.

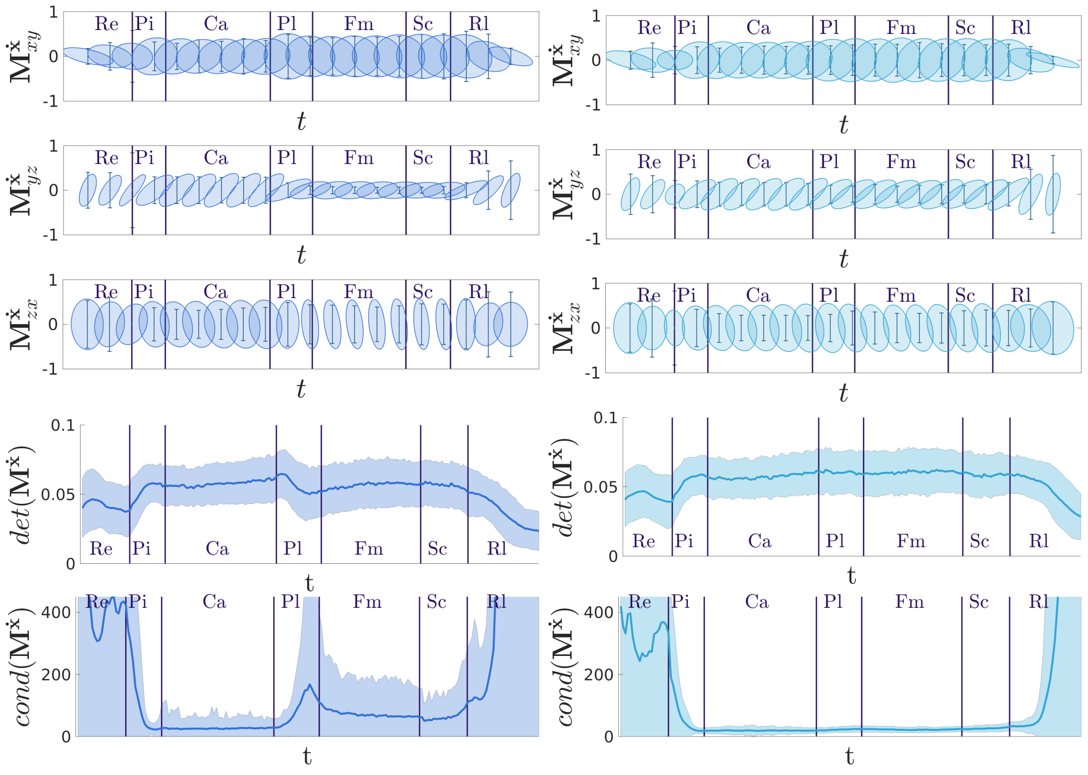
To study the evolution of the manipulability during the screwing task, a subset of manipulability ellipsoids equally spaced in time is first selected by subsampling the manipulability sequence of each action of the task. This results in a dataset containing the same number of manipulability ellipsoids for each action across all trials. The evolution of the right-arm velocity manipulability ellipsoid during the different screwing tasks is studied in Figures 4.4a-4.4c. All the graphs show inter-participants statistics and the ellipsoids are represented w.r.t the shoulder reference frame. The three *top* graphs of each column display the mean of the velocity manipulability ellipsoid profile for the corresponding task computed by (3.2). In each graph, the standard deviation of the ellipsoid along the vertical axis, equal to the square root of the corresponding diagonal element of the covariance tensor (3.4), is represented with error bars. For completeness, the two *bottom* graphs of each column show the mean and standard deviation of two classical manipulability indices, namely $\det(\boldsymbol{M})$ and $\operatorname{cond}(\boldsymbol{M})$, denoting the determinant and condition number of $\boldsymbol{M}$, respectively. The former approximates the manipulability ellipsoid volume, while the latter relates to the ellipsoid isotropy.

Interestingly, we observe that the evolution of the manipulability mean during the task is consistent across the screwing motions. The beginning and end of the task, namely Re and Rl actions, are characterized by narrow ellipsoids along the y axis (i.e. the vertical direction), due to the fact that the arm rests along the body. These also correspond to actions displaying the highest variance. As in Figure 4.3, the velocity manipulability mainly narrows along z and elongates along x and y during Fm and Sc actions for the three screwing heights. Therefore, the manipulability of the participants is generally

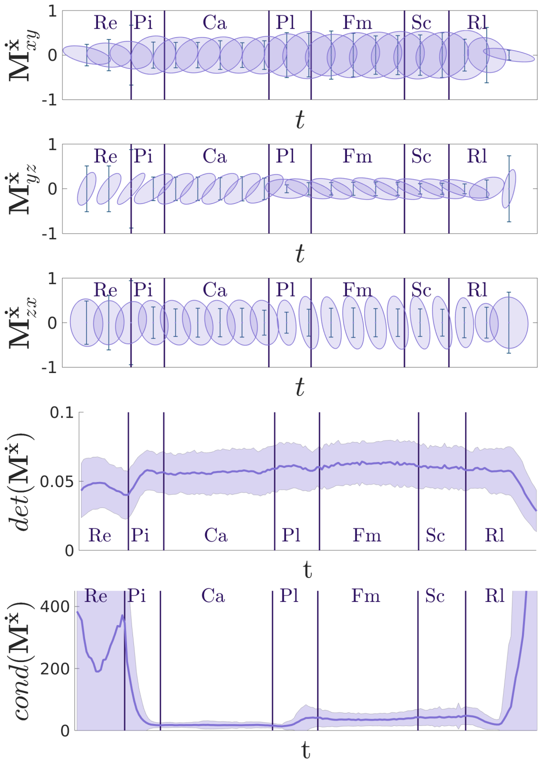


(a) SL (b) SM (c) SH

Figure 4.3 – Posture of the participant 541 during the screwing actions at 3 different heights. The *top* row shows the Xsens avatar view, while the *middle* and *bottom* rows display the skeleton model and right-arm velocity manipulability ellipsoid from 2 different viewpoints, along with the right shoulder reference frame. The manipulability ellipsoids are scaled by a factor 3 for visualization purposes.



(a) Screw low SL (b) Screw middle SM



(c) Screw high SH

Figure 4.4 – Temporal evolution of the single-arm velocity manipulability ellipsoid for the screwing tasks. The inter-participants statistics are displayed with ellipsoids represented w.r.t the shoulder reference frame. The three first rows depict 2D-projections of the ellipsoids mean, along with 3 standard deviations of the vertical axis of the ellipsoid for the specific graph. The two bottom rows display the mean and standard deviation of the determinant and condition number of the ellipsoid, respectively.

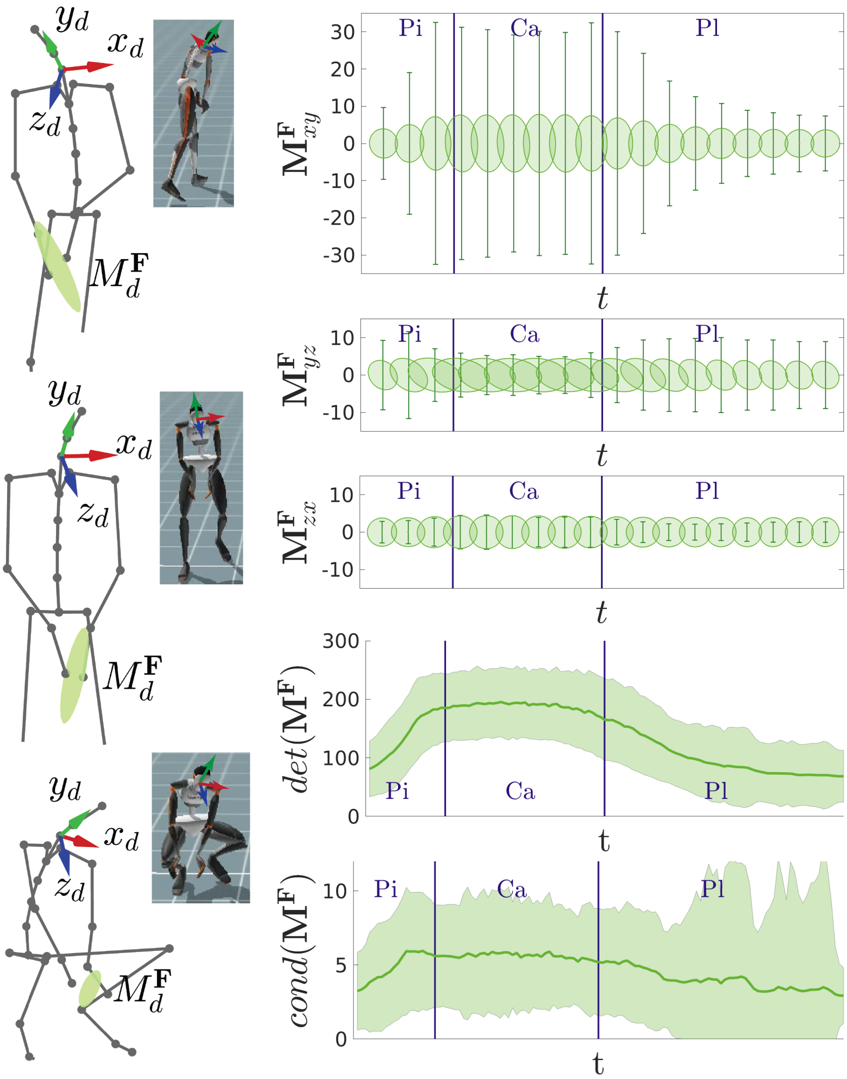
adapted to the task. Moreover, we notice that the ellipsoid shape along x and y is generally similar from `Ca` to `Sc` actions. However, the manipulability ellipsoid is more isotropic for carrying actions `Ca`. This is due to the fact that the participants usually prepare for screwing after having picked the screw and the bolt, i.e., they do not put back their arms at a neutral resting position but instead keep them in front of their torso. Therefore, these manipulability ellipsoids indicate an arm posture adaptation that anticipates the next action (i.e. planning phase), whose manipulability requirements are more specific.

Another relevant observation is that the classical manipulability indices, namely the determinant and condition number of the ellipsoids, tend to remain nearly constant during the whole movement, except in the reaching (`Re`) and releasing (`Rl`) phases. In contrast, as emphasized previously, the shape of the velocity manipulability ellipsoid varies consistently during the different actions of the screwing task. Then, the determinant and condition number are uninformative measures that prohibit a proper human manipulability analysis.

Note that a similar analysis may be conducted for the velocity manipulability ellipsoids of the left arm for the screwing tasks. The left arm manipulability evolves similarly as the one of the right arm along the different actions. This is expected due to the presence of strong symmetries between the arms in this particular task.

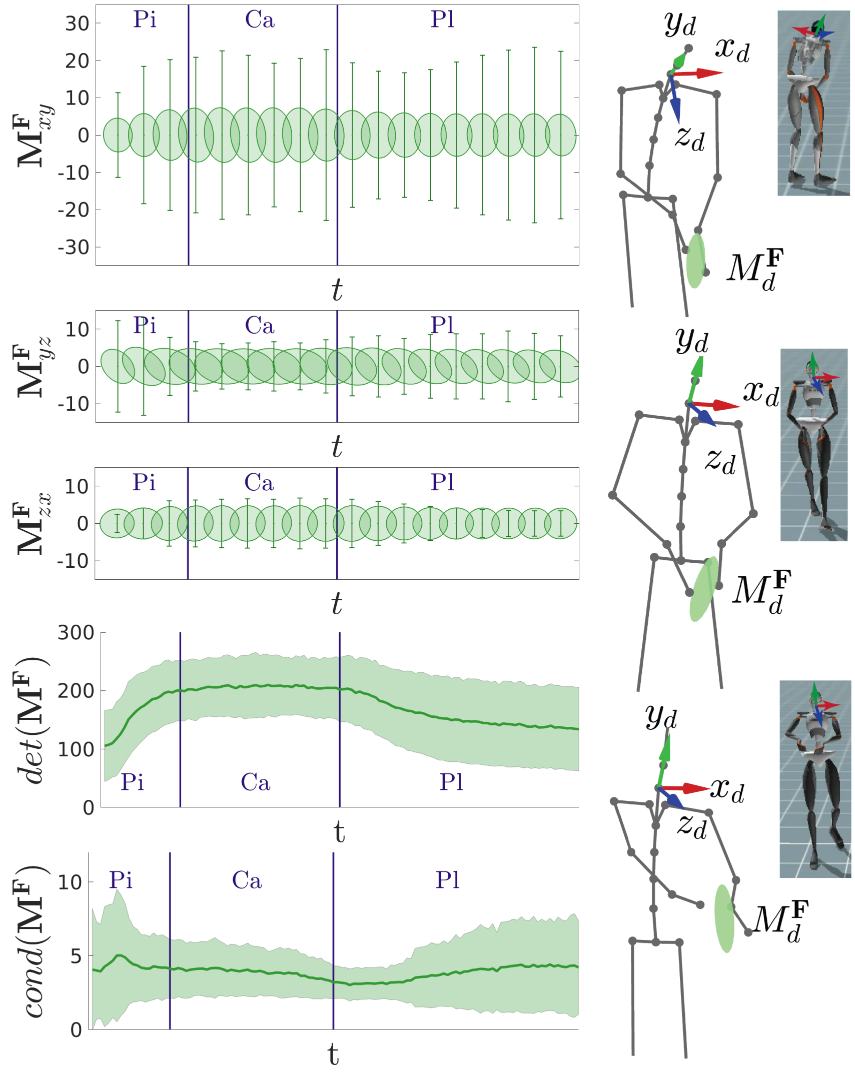
Regarding the carrying task, we analyze the dual-arm force manipulability. Figures 4.5a and 4.6b show the posture of a participant executing the `C5` and `C10` tasks, respectively. Similarly to the results reported in Figures 4.4a-4.4c for the screwing tasks, Figures 4.5b and 4.6a depict the evolution of the dual-arm force manipulability ellipsoid during the two carrying tasks. The inter-participant statistics are displayed and the ellipsoids are represented w.r.t the neck reference frame.

We observe that the main axis of the force manipulability ellipsoids are clearly aligned with the vertical axis during the `Ca` action. Therefore, the posture adopted by the participants favor high force exertion along the vertical axis, which is necessary for carrying heavy loads. However, we do not distinguish consistent differences in the magnitude of the manipulability between the 5 kg and 10 kg loads. Furthermore, the manipulability ellipsoids are almost isotropic at both the beginning of the `Pi` action and during the `Pl` actions. This especially accentuates during `Pl` actions of the `C5` task. This may be attributed to the fact that the shelf where the load is placed is close to the ground, which may require a different posture for this specific action. Overall, higher variations are observed during the carrying tasks compared to those of the screwing tasks. This may be associated with less-strict constraints in the carrying task when contrasted with the screwing motions. Also, the perception of the load may differ across participants, influencing the adopted postures.



(a) 5kg (C5) (b) 5kg (C5)

Figure 4.5 – Temporal evolution of the dual-arm force manipulability ellipsoid for the carrying task C5. *(a)* depict the posture of the participant 541 during the Pi, Ca and Pl actions (from top to bottom). The dual-arm force manipulability ellipsoids (scaled by a factor 0.03) are depicted along with the neck reference frame. The three first rows of *(b)* show 2D projections of the ellipsoids mean, along with 1 standard deviation of the vertical axis of the ellipsoid for the specific graph. The two bottom rows of *(b)* display the mean and standard deviation of the determinant and condition number of the ellipsoid, respectively. The inter-participant statistics are displayed with ellipsoids represented in the neck reference frame.



(a) 10kg (C10) (b) 10kg (C10)

Figure 4.6 – Temporal evolution of the dual-arm force manipulability ellipsoid for the carrying task C10. *(b)* depict the posture of the participant 541 during the Pi, Ca and Pl actions (from top to bottom). The dual-arm force manipulability ellipsoids (scaled by a factor 0.03) are depicted along with the neck reference frame. The three first rows of *(a)* show 2D projections of the ellipsoids mean, along with 1 standard deviation of the vertical axis of the ellipsoid for the specific graph. The two bottom rows of *(a)* display the mean and standard deviation of the determinant and condition number of the ellipsoid, respectively. The inter-participant statistics are displayed with ellipsoids represented in the neck reference frame.

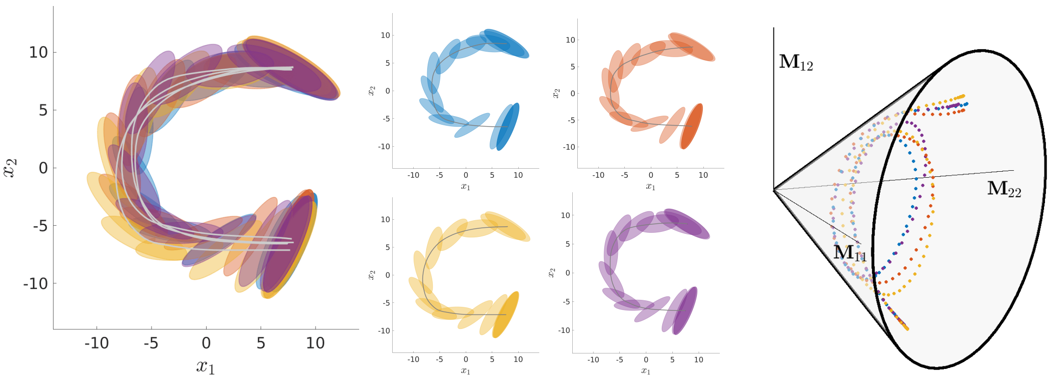
4.4 Learning Manipulability Ellipsoids

The manipulability analysis of human movements can be exploited to teach manipulability-based posture variations to robots executing similar tasks without the need of complex kinematic mappings. To do so, the first open problem is to appropriately encode sequences of demonstrated manipulability ellipsoids and subsequently retrieve a desired manipulability profile that encapsulates the patterns observed during the demonstrations. We tackle this problem from a learning-from-demonstrations perspective by exploiting the GMM/GMR framework on SPD manifold introduced in Chapter 3. Namely, we first encode a set of demonstrated manipulability ellipsoids over the manifold of SPD matrices with a GMM. This probabilistic formulation models the trend of the demonstrated manipulability sequences along with their variability, reflecting their dispersion through the different demonstrations. After, a distribution of the desired manipulability ellipsoids can be retrieved via GMR on the SPD manifold.

In order to illustrate the functionality of the proposed learning approach, we carried out an experiment using a couple of simulated planar robots with dissimilar embodiments and a different number of joints. The central idea is to teach a redundant robot to track a reference trajectory in Cartesian space with a desired time-varying manipulability ellipsoid. For the demonstration phase, a 3-DoF *teacher* robot follows a C-shape trajectory four times, from which we extracted both the end-effector position $\boldsymbol{x}_t$ and robot manipulability ellipsoid $\boldsymbol{M}_t(\boldsymbol{q})$, at each time step t. The collected time-aligned data were split into two training datasets of time-driven trajectories, namely Cartesian position and manipulability. We trained a classical GMM over the time-driven Cartesian trajectories and a geometry-aware GMM over the time-driven manipulability ellipsoids, using models with five components, i.e. $C=5$ (the number was empirically chosen by the experimenter).

During the reproduction phase, a 5-DoF *student* robot executed the time-driven task by following a desired Cartesian trajectory $\hat{\boldsymbol{x}}_t$ computed from a classical GMR as $\hat{\boldsymbol{x}}_t \sim p(\boldsymbol{x}\,|\,t)$. As secondary task, the robot was also required to vary its joint configuration for matching desired manipulability ellipsoids $\hat{\boldsymbol{M}}_t \sim p(\boldsymbol{M}|t)$, estimated by GMR over the SPD manifold.

Figure 4.7 shows the four demonstrations carried out by the 3-DoF robot, where both the Cartesian trajectory and manipulability ellipsoids are displayed. Note that the recorded manipulability ellipsoids slightly change across demonstrations as a side effect of the variation observed in both the initial end-effector position and the generated trajectory. Also notice that manipulability ellipsoids are represented as ellipsoids in the Cartesian space, while they correspond to points in the SPD manifold $\mathcal{S}^2_{++}$, as shown by Figure 4.7c. Figures 4.8a and 4.8b display the demonstrated ellipsoids (in gray) along with the center $\boldsymbol{\Xi}_\ell$ of the five components of the GMM encoding $\boldsymbol{M}$. These are centered at the end-effector position recovered by the classical GMR for the corresponding time steps represented in the geometry-aware GMM. Figure 4.8c shows the demonstrated

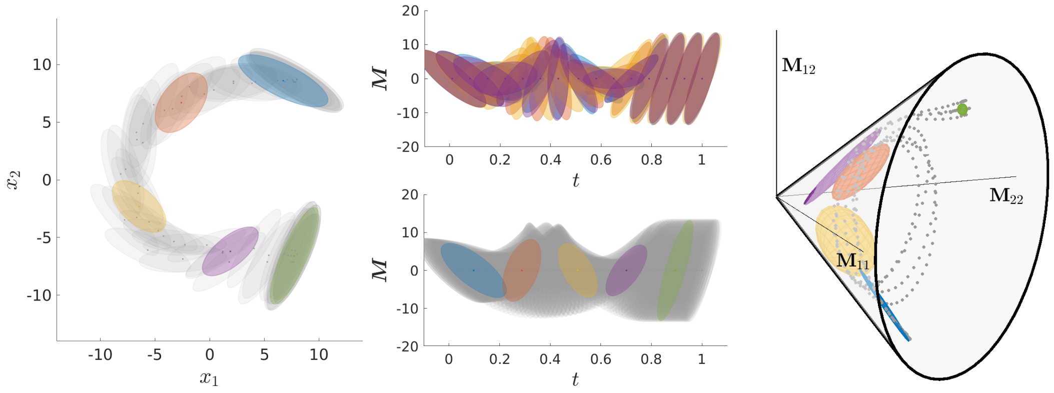


(a) Demonstrations in $\mathbb{R}^2$ (b) Individual demonstrations (c) Demonstrations in $\mathcal{S}^2_{++}$

Figure 4.7 – Four demonstrations of a 3-DoF planar robot tracking a C-shape trajectory. (a) displays the end-effector path (light gray solid lines) in the task space $\mathbb{R}^2$ and the manipulability ellipsoids at different time steps are shown for all the demonstrations. (b) shows the demonstrations independently to exhibit more clearly the differences produced in the manipulability ellipsoids due to the robot posture variation across demonstrations. (c) displays the demonstrated manipulability ellipsoids in the SPD cone $\mathcal{S}^2_{++}$. Position $\boldsymbol{x}$ is given in centimeters.

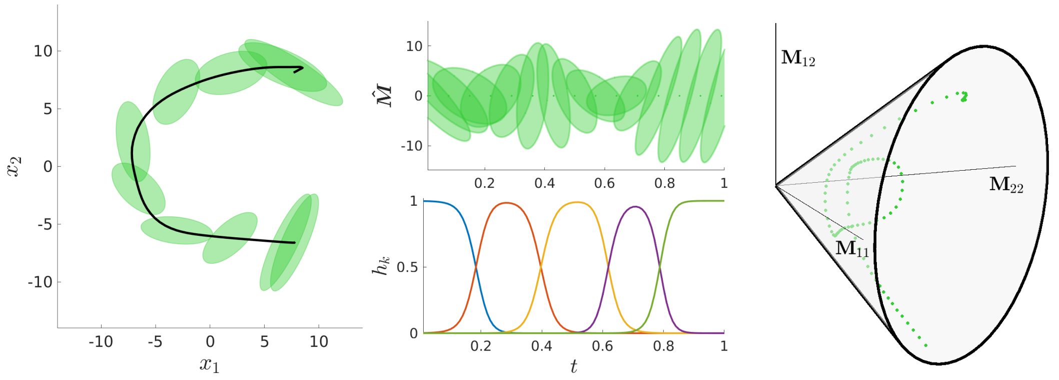
ellipsoids on the cone $\mathcal{S}^2_{++}$ along with the centers Ξ_ℓ and covariances $\boldsymbol{\mathcal{S}}_\ell$ of the five components of the GMM encoding $\boldsymbol{M}$. Note that, as the demonstrated ellipsoids, the centers Ξ_ℓ corresponds to points on $\mathcal{S}^2_{++}$, while each covariance $\boldsymbol{\mathcal{S}}_\ell$ is represented as an ellipsoid in the tangent space of the corresponding center Ξ_ℓ. Figure 4.9 shows the desired Cartesian trajectory and manipulability ellipsoid profile respectively estimated by classical GMR and GMR in the SPD manifold. Both manipulability and Cartesian path are references to be tracked by the student robot.

These results validate that the proposed learning framework permits to learn and plan the reproduction of reference trajectories, while fulfilling additional task requirements encapsulated in a profile of desired manipulability ellipsoids. The case of human-to-robot manipulability transfer, in which a robot learns a profile of manipulability ellipsoids from human demonstrations, is handled similarly to the simple planar example presented in this section. Therefore, a GMM model would first be learned from the demonstrated human manipulability (and other trajectories), as demonstrated in the planar example. Then, GMR would be used to retrieve a distribution of manipulability ellipsoids (along with other variables) that may be used by a robot to imitate the human manipulability for the given task. However, in order to reproduce the learned task, the *student* robot needs to track the desired manipulability profile obtained in the learning phase. To do so, a manipulability tracking formulation is required. This second challenge will be tackled in Chapter 5. The experiments involving robots learning human manipulability profiles are reported in Chapter 6, where both the learning and reproduction phases will be analyzed.



(a) GMM in task space (b) GMM over time (c) GMM in $\mathcal{S}^2_{++}$

Figure 4.8 – GMM encoding the four demonstrations of a C-shape trajectory. (a) displays the demonstrated manipulability ellipsoids (in gray) and centers $\boldsymbol{\Xi}_\ell$ of the 5-state GMM in the SPD manifold. (b) shows the demonstrated manipulability ellipsoids over time (top in color, bottom in gray), and the centers $\boldsymbol{\Xi}_\ell$ of the 5-state GMM in the SPD manifold (bottom). (c) displays the demonstrated manipulability ellipsoids (in gray) and centers $\boldsymbol{\Xi}_\ell$ and covariances $\boldsymbol{\mathcal{S}}_\ell$ of the 5-state GMM in the SPD cone $\mathcal{S}^2_{++}$. Position $\boldsymbol{x}$ and time t are given in centimeters and seconds, respectively.



(a) Reproduction in task space (b) Reproduction over time (c) Reproduction in $\mathcal{S}^2_{++}$

Figure 4.9 – Desired execution of a C-shape tracking task. (a) shows the desired Cartesian trajectory and manipulability profile depicted as a black curve and green ellipsoids, respectively. (b)-*top* displays the desired manipulability ellipsoids estimated by GMR over time. (b)-*bottom* illustrates the influence of GMM components on the time-driven GMR. The colors match the distributions shown in Fig. 4.8. (c) depicts the desired manipulability profile on the SPD manifold $\mathcal{S}^2_{++}$.

4.5 Conclusion

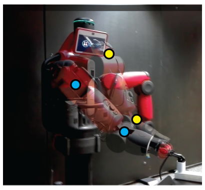
The contributions of this chapter were two-fold. First, we presented a detailed analysis of single and dual-arm manipulability ellipsoids for human movements during industry-like activities. Statistical analyses considering the intrinsic geometry of the manipulability ellipsoids were conducted on the kinematics data records of participants executing screwing and load carrying tasks. Our work showed that the evolution of the manipulability ellipsoid shape provides more information about human motion than the classical manipulability indices classically used in the literature. Secondly, we introduced a geometry-aware learning framework that allows the encoding and retrieval manipulability ellipsoids. Unlike classical learning frameworks that encode reference position, velocity and force trajectories, our approach offers the possibility of learning posture-dependent task requirements such as preferred directions for motion and force exertion in operational space, which are encapsulated in the demonstrated manipulability ellipsoids.

Given a desired manipulability ellipsoid profile and desired reference trajectories in the form of Cartesian position or force, the goal of the robot is to reproduce the task by tracking these reference trajectories. To do so, the next chapter will introduce controllers allowing a robot to reproduce a desired manipulability profile by exploiting its own redundant kinematic structure while minimizing the error between the current manipulability ellipsoid and the desired one. Moreover, Chapter 6 will illustrate how the manipulability analysis of human movements can be exploited to transfer manipulability-based posture variation to robots executing similar tasks without the need of complex kinematic mappings.

Part II

Designing Geometry-aware Controllers

5 Tracking Manipulability Ellipsoids



The Part I of this thesis focused on learning SPD matrices from demonstrations. After having learned a reference trajectory in the form of SPD matrices, the goal of the robot is to reproduce the task by tracking this sequence using appropriate controllers. The Part II of this thesis tackles this challenge.

In this context, the following chapter introduces a novel tracking control scheme in which the robot is requested to follow a desired profile of manipulability ellipsoids, either as its main task or as a secondary objective. The proposed formulation exploits tensor-based representations and takes into account that manipulability ellipsoids lie on the manifold of symmetric positive definite (SPD) matrices.

Publication note

The material presented in this chapter is adapted from the following publications:

- Jaquier, N., Rozo, L., Caldwell, D. G., and Calinon, S. (2018). Geometry-aware tracking of manipulability ellipsoids. In *Robotics: Science and Systems (R:SS)*;
- Jaquier, N., Rozo, L., Caldwell, D. G., and Calinon, S. (2020b). Geometry-aware manipulability learning, tracking and transfer. *Intl. Journal of Robotics Research*;

Supplementary material

Videos related to this chapter are available at:
https://sites.google.com/view/manipulability.

Source code

Source codes related to this chapter are available at:
https://github.com/NoemieJaquier/Manipulability.

Examples of manipulability tracking with the Centauro robot are available within the PyRoboLearn toolbox.

5.1 Introduction

In the previous chapter, we proposed a manipulability-based analysis of human movements, where we demonstrated that posture patterns arise during the execution of industrial tasks. This validates the hypothesis that we, as humans, tend to adopt an appropriate pose when carrying out manipulation tasks to help us regulate our motion and strengthen our capability to achieve the given task. Moreover, we introduced a manipulability learning framework that can retrieve a sequence of desired manipulability ellipsoids from human demonstrations. In order to be able to reproduce the learned manipulability profile for a given task, robots have to be endowed with adapted controllers.

Being aware of the impact of robot posture on reaching movements and manipulation tasks, several works considered manipulability ellipsoids as means to analyze and modify the posture of the robot to improve its performances during task execution. For example, manipulability ellipsoids have been used to measure the compatibility of robot postures with respect to fine and coarse manipulation [Chiu, 1987], and to improve minimum-time trajectory planning using a manipulability-aware inverse kinematics algorithm [Chiacchio, 1990]. Vahrenkamp et al. [2012] proposed a grasp selection process that favored high manipulability in the robot workspace. Other works have focused on maximizing the manipulability ellipsoid volume in trajectory generation algorithms [Guilamo et al., 2006], and task-level robot programming frameworks [Somani et al., 2016], to obtain singularity-free joint trajectories and high task-space dexterity. Nevertheless, as stated in [Lee, 1989], solely maximizing the ellipsoid volume to achieve high dexterity in motion may cause a reverse effect on the flexibility in force.

The aforementioned approaches do not specify a desired robot manipulability for the task. In contrast, Lee and Oh [2016] proposed an optimization method to find reaching postures for a humanoid robot that achieved desired (manually-specified) manipulability volumes. Similarly, a series of desired manipulability ellipsoids was predefined according to Cartesian velocity and force requirements in dual-arm manipulation tasks [Lee, 1989]. Note that both Lee [1989] and Lee and Oh [2016] predetermined the task-dependent robot manipulability, which required a meticulous and demanding analysis about the motion that the robot needed to perform, which becomes impractical when the robot is required to carry out a large set of different tasks. Furthermore, these approaches overlooked an important characteristic of manipulability ellipsoids, namely, the fact that they lie on the manifold of SPD matrices. This may influence the optimal robot joint configuration for the task at hand.

In this chapter, we address the problem of tracking robot manipulability ellipsoids from a novel geometry-aware control perspective. The proposed manipulability tracking formulation is inspired by the classical inverse kinematics problem in robotics, where a first-order differential relationship between the robot manipulability ellipsoid and the robot joints is established, as explained in Section 5.2. This relationship also demands to consider that manipulability ellipsoids lie on the SPD manifold, which is here tackled by exploiting tensor-based representations and differential geometry. Various exponentially-stable manipulability tracking controllers are then derived from this relationship. The geometry-awareness of our formulations is crucial for achieving successful manipulability tracking, as shown in Section 5.3 by comparison against geometry-oblivious tracking approaches and state-of-the-art manipulability-based optimization methods. Note that Riemannian geometry has also been successfully exploited in robot motion optimization [Ratliff et al., 2015] and manipulability analysis of closed chains [Park and Kim, 1998]. For sake of clarity, different aspects of the proposed tracking approaches are illustrated with simple examples using simulated planar robots throughout the chapter.

The proposed approach can be straightforwardly applied to different types of kineto-static and dynamic manipulability measures and can integrate the robot actuators contribution. This opens the door to manipulability tracking problems with various types of robots where different task requirements at kinematic and dynamic levels are needed, which may determine time-varying optimal directions for controlling a robot (e.g., velocity, force, acceleration) to perform successfully. Moreover, the proposed mathematical development is compatible with statistical methods providing 4th-order covariances (see Chapter 4), which are here exploited to reflect the manipulability tracking precision required by the task. This also allows our formulation to be easily combined with manipulability learning frameworks, where desired manipulability ellipsoid profiles are obtained from demonstrations of a specific task performed by either a human or a robot[1]. The functionality of the proposed approach is extensively evaluated in different manipulability tracking tasks involving a 7-DoF robot, a 16-DoF robotic hand and two legged robots (see Section 5.4).

5.2 Tracking Manipulability Ellipsoids

Several robotic manipulation tasks may demand the robot to track a desired trajectory with certain velocity specifications, or apply forces along different task-related axes. These requirements are more easily achieved if the robot adopts a posture that suits velocity or force control commands. In other tasks, the robot may be required to adopt a posture that comply several aligned velocity or force requirements. These problems can be viewed as matching a set of desired manipulability ellipsoids that are compatible with the task requirements. In this section, we introduce an approach that addresses this problem by exploiting the mathematical concepts presented in Sections 2.2.4, 2.3.2 and 2.3.4.

---

[1] The combination of the learning framework of Chapter 4 and the tracking framework presented in this chapter will be covered in Chapter 6.

5.2.1 Manipulability Jacobian

Given a desired profile of manipulability ellipsoids, the goal of the robot is to adapt its posture to match the desired manipulability, either as its main task or as a secondary objective. We here propose a formulation inspired by the classical inverse kinematics problem in robotics, which permits to compute the joint angle commands to track a desired manipulability ellipsoid.

First, the manipulability ellipsoid is expressed as a function of time

$$\boldsymbol{M}(t) = f\Big(\boldsymbol{J}\big(\boldsymbol{q}(t)\big)\Big), \tag{5.1}$$

for which we can compute the first-order time derivative by applying the chain rule as

$$\frac{\partial \boldsymbol{M}(t)}{\partial t} = \frac{\partial f(\boldsymbol{J}(\boldsymbol{q}))}{\partial \boldsymbol{q}} \times_3 \frac{\partial \boldsymbol{q}(t)}{\partial t}^{\mathsf{T}} = \boldsymbol{\mathcal{J}}(\boldsymbol{q}) \times_3 \dot{\boldsymbol{q}}^{\mathsf{T}}, \tag{5.2}$$

where $\boldsymbol{\mathcal{J}} \in \mathbb{R}^{6\times 6\times n}$ is the *manipulability Jacobian* of an n-DoF robot, representing the linear sensitivity of the changes in the robot manipulability ellipsoid $\dot{\boldsymbol{M}} = \frac{\partial \boldsymbol{M}(t)}{\partial t}$ to the joint velocity $\dot{\boldsymbol{q}} = \frac{\partial \boldsymbol{q}(t)}{\partial t}$. Note that the computation of the manipulability Jacobian depends on the type of manipulability ellipsoid that is used. We develop here the expressions for the force, velocity and dynamic manipulability ellipsoids.

The derivation of the manipulability Jacobian $\boldsymbol{\mathcal{J}}^{\dot{\boldsymbol{x}}}$ corresponding to the velocity manipulability ellipsoid $\boldsymbol{M}^{\dot{\boldsymbol{x}}} = \boldsymbol{J}\boldsymbol{J}^{\mathsf{T}}$ is straightforward by using (2.32) and (2.33) [2]

$$\boldsymbol{\mathcal{J}}^{\dot{\boldsymbol{x}}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{J} + \frac{\partial \boldsymbol{J}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J}. \tag{5.3}$$

Similarly, the manipulability Jacobian $\boldsymbol{\mathcal{J}}^{\boldsymbol{F}}$ corresponding to the force manipulability ellipsoid $\boldsymbol{M}^{\boldsymbol{F}} = (\boldsymbol{J}\boldsymbol{J}^{\mathsf{T}})^{-1}$ is obtained using (2.32), (2.33) and (2.34),

$$\boldsymbol{\mathcal{J}}^{\boldsymbol{F}} = -\left(\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{J} + \frac{\partial \boldsymbol{J}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J}\right) \times_1 \boldsymbol{M}^{\boldsymbol{F}} \times_2 \boldsymbol{M}^{\boldsymbol{F}}. \tag{5.4}$$

In a similar fashion, the manipulability Jacobian $\boldsymbol{\mathcal{J}}^{\ddot{\boldsymbol{x}}}$ corresponding to the dynamic manipulability ellipsoid $\boldsymbol{M}^{\ddot{\boldsymbol{x}}} = \boldsymbol{\Upsilon}\boldsymbol{\Upsilon}^{\mathsf{T}}$ with $\boldsymbol{\Upsilon} = \boldsymbol{J}\boldsymbol{\Lambda}(\boldsymbol{q})^{-1}$ (as defined in [Yoshikawa, 1985a], where $\boldsymbol{\Lambda}(\boldsymbol{q})$ is the robot inertia matrix), is computed as

$$\boldsymbol{\mathcal{J}}^{\ddot{\boldsymbol{x}}} = \frac{\partial \boldsymbol{\Upsilon}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{\Upsilon} + \frac{\partial \boldsymbol{\Upsilon}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{\Upsilon}, \tag{5.5}$$

---

[2] In the remainder of the chapter we drop dependencies on $\boldsymbol{q}$ to simplify the notation.

where

$$\begin{aligned}\frac{\partial \boldsymbol{\Upsilon}}{\partial \boldsymbol{q}} &= \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{\Lambda}^{-\mathsf{T}} + \frac{\partial \boldsymbol{\Lambda}^{-1}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J} \\ &= \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{\Lambda}^{-\mathsf{T}} - \frac{\partial \boldsymbol{\Lambda}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{\Upsilon} \times_2 \boldsymbol{\Lambda}^{-\mathsf{T}}.\end{aligned}$$

Details on the computation of the derivative of the Jacobian and inertia matrix w.r.t the joint angles are given in Appendices B.1 and B.2.

5.2.2 Geometry-aware Manipulability Tracking Formulation

Velocity-based controller

A solution to control a robot so that it tracks a desired end-effector trajectory is to compute the desired joint velocities using the inverse kinematics formulation derived from (4.1). We use here a similar approach to compute the joint velocities $\dot{\boldsymbol{q}}$ to track a desired manipulability profile. More specifically, by minimizing the norm of the residuals

$$\min_{\dot{\boldsymbol{q}}} \|\dot{\boldsymbol{M}} - \boldsymbol{\mathcal{J}} \times_3 \dot{\boldsymbol{q}}^{\mathsf{T}}\|_{\mathrm{F}} = \min_{\dot{\boldsymbol{q}}} \|\mathrm{mvec}(\dot{\boldsymbol{M}}) - \boldsymbol{\mathcal{J}}_{(3)}^{\mathsf{T}} \dot{\boldsymbol{q}}\|_{\mathrm{F}},$$

we can compute the required joint velocities of the robot to track a profile of desired manipulability ellipsoids as its main task with

$$\dot{\boldsymbol{q}} = (\boldsymbol{\mathcal{J}}_{(3)}^{\dagger})^{\mathsf{T}} \mathrm{mvec}(\dot{\boldsymbol{M}}), \tag{5.6}$$

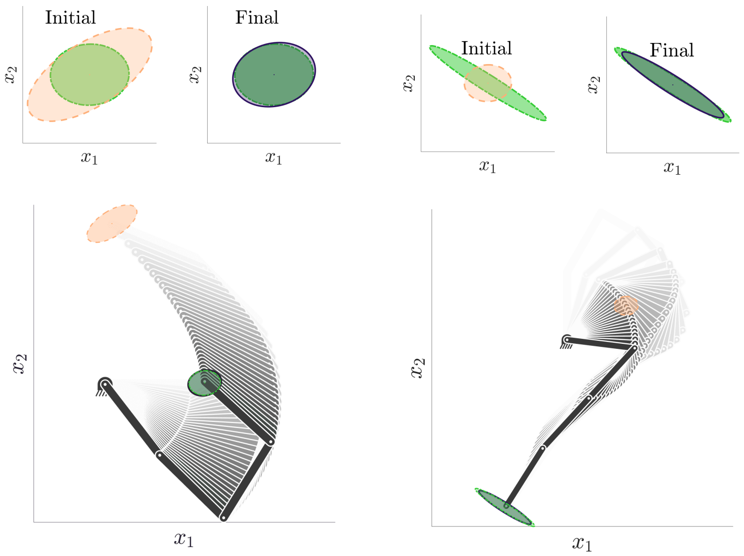
where $\mathcal{J}_{(3)}$ is the mode-3 matricization of the 3rd-order manipulability Jacobian tensor and $\mathrm{mvec}(\dot{\boldsymbol{M}})$ is the vectorization of the matrix $\dot{\boldsymbol{M}}$.

Note that (5.6) allows us to define a controller to track a reference manipulability ellipsoid as main task, similarly as the classical velocity-based control that tracks a desired task-space velocity. To do so, we propose to use a geometry-aware similarity measure to compute the joint velocities necessary to move the robot towards a posture where the error between the current manipulability ellipsoid $\boldsymbol{M}_t$ and the desired one $\hat{\boldsymbol{M}}_t$ is minimum. Specifically, the difference between manipulability ellipsoids is computed using the logarithmic map (2.17) from the current manipulability $\boldsymbol{M}_t$ on the SPD manifold. Therefore, the corresponding controller is given by

$$\dot{\boldsymbol{q}}_t = (\boldsymbol{\mathcal{J}}_{(3)}^{\dagger})^{\mathsf{T}}\, \boldsymbol{K}_{\boldsymbol{M}}\, \mathrm{mvec}\Big(\mathrm{Log}_{\boldsymbol{M}_t}(\hat{\boldsymbol{M}}_t)\Big), \tag{5.7}$$

where $\boldsymbol{K}_{\boldsymbol{M}}$ is a gain matrix.

Alternatively, for the case in which the main task of the robot is to track reference



(a) (b)

Figure 5.1 – Manipulability tracking as main task. The robot color goes from light gray to black to show the evolution of the posture. Initial, final, and desired manipulability ellipsoids are respectively depicted in yellow, dark purple, and green. The *top* rows show close-up plots corresponding to the initial and final manipulability.

trajectories in the form of Cartesian positions or force profiles, the tracking of a profile of manipulability ellipsoids is assigned a secondary role. Thus, the robot task objectives are to track the reference trajectories while exploiting the kinematic redundancy to minimize the difference between current and desired manipulability ellipsoids. In this situation, a manipulability-based redundancy resolution is carried out by computing a nullspace velocity that similarly exploits the geometry of the SPD manifold. Thus, the corresponding controller is given by

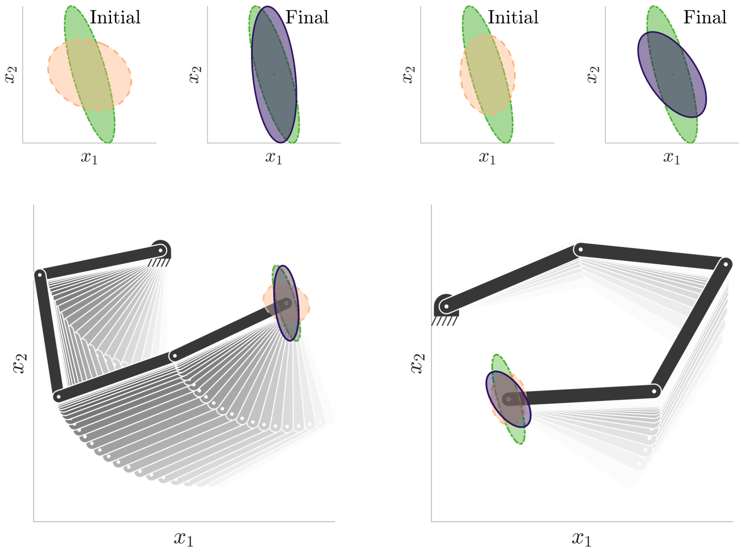
$$\dot{\boldsymbol{q}}_t = \underbrace{\boldsymbol{J}^{\dagger}\, \boldsymbol{K}_{\boldsymbol{x}}\, (\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t)}_{\text{main task}} + \underbrace{(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})}_{\text{nullspace op.}}\, \underbrace{(\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}}\, \boldsymbol{K}_{\boldsymbol{M}}\, \mathrm{mvec}\Big(\mathrm{Log}_{\boldsymbol{M}_t}(\hat{\boldsymbol{M}}_t)\Big),}_{\text{secondary task}} \tag{5.8}$$

where $\boldsymbol{K}_{\boldsymbol{x}}$ and $\boldsymbol{K}_{\boldsymbol{M}}$ are gain matrices for the position and manipulability terms.

Note that matricization and vectorization operations can be defined using Mandel notation to alleviate the computational cost of the controllers using tensor representations, such that

$$\mathcal{X}_{(3)} = \begin{pmatrix} \mathrm{mvec}\,(\boldsymbol{\mathcal{X}}_{:,:,1})^{\mathsf{T}} \\ \vdots \\ \mathrm{mvec}\,(\boldsymbol{\mathcal{X}}_{:,:,K})^{\mathsf{T}} \end{pmatrix} \tag{5.9}$$

for $2\times2\times K$ third-order tensors, with mvec$(\cdot)$ as in (2.29).



(a) (b)

Figure 5.2 – Manipulability-based redundancy resolution with Cartesian position control. The organization of the graphs and the colors are identical to Fig. 5.1.

Table 5.1 – Initial and final Riemannian affine-invariant distances $d_{\mathcal{S}^d_{++}}(\hat{\boldsymbol{M}}, \boldsymbol{M}_t)$ between the current and desired manipulability for the experiments illustrated in Fig. 5.1 and 5.2.

		Initial	Final
Main task	Fig. 5.1a	1.342	0.199
	Fig. 5.1b	2.950	0.3217
Redundancy resolution	Fig. 5.2a	2.194	0.955
	Fig. 5.2b	1.612	1.495

In order to show the functionality of the proposed approach where the goal of the robot is to reproduce a given manipulability ellipsoid either as its main task or as a secondary objective, we carried out experiments with a simulated 4-DoF planar robot. In the first case, the robot is required to vary its joint configuration to make its manipulability ellipsoid $\boldsymbol{M}_t$ coincide with the desired one $\hat{\boldsymbol{M}}$, without any task requirement at the level of its end-effector. In the second case, the robot needs to keep its end-effector at a fixed Cartesian position while moving its joints to match the desired manipulability ellipsoid. Figures 5.1 and 5.2 show how the manipulator configuration is successfully adjusted so that $\boldsymbol{M}_t \simeq \hat{\boldsymbol{M}}$ when the manipulability ellipsoid tracking is considered as the main task or as a secondary objective (see also Table 5.1). These results show that our geometry-aware controllers inspired by the inverse kinematics formulation are suitable to solve the manipulability ellipsoid tracking problem.

Stability analysis

We here analyze the stability properties of the proposed manipulability tracking controller given the geometry of the underlying manifold. First of all, note that the dynamical system operated by the controller (5.7) corresponds to

$$\dot{\boldsymbol{M}} = k_{\boldsymbol{M}} \operatorname{Log}_{\boldsymbol{M}}(\hat{\boldsymbol{M}}), \tag{5.10}$$

where the controller gain is assumed to be a positive scalar value for sake of simplicity. Then, we select the Lyapunov function V as

$$V(\boldsymbol{M}) \;\; = \;\; \langle \boldsymbol{F}, \boldsymbol{F} \rangle_{\hat{\boldsymbol{M}}}, \tag{5.11}$$

where $\boldsymbol{F} = \operatorname{Log}_{\hat{\boldsymbol{M}}}(\boldsymbol{M})$ is a vector field composed of the initial velocities of all geodesics departing from the origin $\hat{\boldsymbol{M}}$, and $\langle \cdot, \cdot \rangle_{\hat{\boldsymbol{M}}}$ is the inner product (2.14). With this, we state our first theorem:

Theorem 1. *The function* (5.11) *is a valid Lyapunov function.*

The proof of Theorem 1 is based on the following remark from [Pait and Colón, 2010]:

Remark 1. *The function* (5.11) *is a Lyapunov function for a dynamical system* $\dot{\boldsymbol{M}} = h(\boldsymbol{M})$ *such that* $h(\hat{\boldsymbol{M}}) = 0$ *if the Lie derivative* $\mathcal{L}_h V(\boldsymbol{M}) = 2\langle h, \boldsymbol{F} \rangle_{\hat{\boldsymbol{M}}}$ *is negative everywhere except at the origin* $\hat{\boldsymbol{M}}$*.*

Proof of Theorem 1. To verify the condition on $\mathcal{L}_h V$, we first express the velocity of the dynamical system (5.10) in the tangent space of $\hat{\boldsymbol{M}}$ using parallel transport as

$$\Gamma_{\boldsymbol{M} \to \hat{\boldsymbol{M}}}(\dot{\boldsymbol{M}}) = -k_{\hat{\boldsymbol{M}}} \operatorname{Log}_{\hat{\boldsymbol{M}}}(\boldsymbol{M}). \tag{5.12}$$

The Lie derivative $\mathcal{L}_h V$ of the proposed Lyapunov function for the dynamical system (5.12) is given by

$$\begin{aligned} \mathcal{L}_h V(\boldsymbol{M}) \;\; &= \;\; 2\langle -k_{\hat{\boldsymbol{M}}} \operatorname{Log}_{\hat{\boldsymbol{M}}}(\boldsymbol{M}), \operatorname{Log}_{\hat{\boldsymbol{M}}}(\boldsymbol{M}) \rangle_{\hat{\boldsymbol{M}}} \\ &= -2k_{\hat{\boldsymbol{M}}} \, \langle \operatorname{Log}_{\hat{\boldsymbol{M}}}(\boldsymbol{M}), \operatorname{Log}_{\hat{\boldsymbol{M}}}(\boldsymbol{M}) \rangle_{\hat{\boldsymbol{M}}} \\ &= -2k_{\hat{\boldsymbol{M}}} V. \end{aligned} \tag{5.13}$$

Therefore, we have

$$\begin{aligned} &V(\boldsymbol{M}) > 0, \;\; \mathcal{L}_h V(\boldsymbol{M}) < 0 \qquad \forall \;\; \boldsymbol{M} \neq \hat{\boldsymbol{M}}, \\ &V(\boldsymbol{M}) = \mathcal{L}_h V(\boldsymbol{M}) = 0 \iff \boldsymbol{M} = \hat{\boldsymbol{M}}, \end{aligned}$$

so that the function (5.11) is a valid Lyapunov function. ☐

Our main theorem can then be stated based on Theorem 1:

Theorem 2. *The controller* (5.7) *is exponentially stable.*

The proof of Theorem 2 is based on the following remark from [Wu, 2020]:

Remark 2. *The equilibrium point* $\hat{\boldsymbol{M}}$ *is exponentially stable if the Lyapunov function* $V(\boldsymbol{M})$ *satisfies the two following conditions*

$$c_1\, d^p_{\mathcal{S}^d_{++}}(\boldsymbol{M}, \hat{\boldsymbol{M}}) \leq V(\boldsymbol{M}) \leq c_2\, d^p_{\mathcal{S}^d_{++}}(\boldsymbol{M}, \hat{\boldsymbol{M}}), \tag{5.14}$$

$$\mathcal{L}_h V(\boldsymbol{M}) \leq -c_3\, d^p_{\mathcal{S}^d_{++}}(\boldsymbol{M}, \hat{\boldsymbol{M}}), \tag{5.15}$$

with $p > 0$, $c_i > 0$ *for* $i = \{1, 2, 3\}$ *and* $d_{\mathcal{S}^d_{++}}(\cdot, \cdot)$ *the affine-invariant distance* (2.15).

Proof of Theorem 2. The Lyapunov function (5.11) can be equivalently expressed as $V(\boldsymbol{M}) = d^2_{\mathcal{S}^d_{++}}(\boldsymbol{M}, \hat{\boldsymbol{M}})$. Therefore, the conditions (5.14) and (5.15) hold with $p = 2$, $0 < c_1 \leq 1$, $c_2 \geq 1$ and $c_3 = 2k_{\hat{M}} > 0$. □

Finally, it can be easily shown that this result holds with $c_3 = 2\lambda_{\min}(\boldsymbol{K}_{\hat{M}})$ for a positive-definite controller gain matrix $\boldsymbol{K}_{\hat{M}}$, where $\lambda_{\min}(\cdot)$ returns the minimum eigenvalue of the matrix.

Note that the Lyapunov function (5.11) is similar to the one usually defined to demonstrate the exponential stability of the classical inverse kinematic-based velocity controller $\dot{\boldsymbol{q}}_t = \boldsymbol{J}^\dagger \boldsymbol{K}_{\boldsymbol{x}}\,(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t)$. In that case, the Lyapunov function is defined as $V(\boldsymbol{x}) = (\hat{\boldsymbol{x}} - \boldsymbol{x})^\mathsf{T}(\hat{\boldsymbol{x}} - \boldsymbol{x})$, which is equivalent to the inner product $\langle \boldsymbol{e}, \boldsymbol{e} \rangle$ with the error $\boldsymbol{e} = \hat{\boldsymbol{x}} - \boldsymbol{x}$. In the case of manipulability tracking, the inner product $\langle \cdot, \cdot \rangle$ is defined in the SPD manifold and the error $\boldsymbol{e}$ is computed as $\mathrm{Log}_{\hat{M}}(\boldsymbol{M})$. Finally, it is worth highlighting that when the manipulability tracking is assigned a secondary role, the controller (5.8) does not influence the stability of the main task of the robot as the manipulability-based redundancy resolution is carried out in the corresponding nullspace.

Acceleration-based controller

Similarly to the velocity-based controller, we propose a geometry-aware acceleration-based controller that allows the computation of the joint accelerations $\ddot{\boldsymbol{q}}$ required to track a desired manipulability trajectory (i.e. desired manipulability and manipulability velocity profiles). The approach is inspired by the inverse kinematics formulation and its differential relationships used to compute the joint accelerations necessary to track desired end-effector positions and velocities.

To formalize the acceleration-based controller, let us first define the second-order time derivative of the manipulability ellipsoid computed from (5.2) by applying the product rule

$$\frac{\partial^2 \boldsymbol{M}(t)}{\partial t^2} = \boldsymbol{\mathcal{J}}(\boldsymbol{q}) \times_3 \ddot{\boldsymbol{q}}^{\mathsf{T}} + \dot{\boldsymbol{\mathcal{J}}}(\boldsymbol{q}) \times_3 \dot{\boldsymbol{q}}^{\mathsf{T}}, \tag{5.16}$$

(see Appendix B.3 for details on the computation of $\dot{\boldsymbol{\mathcal{J}}}(\boldsymbol{q})$). So, by minimizing the ℓ^2-norm of the residuals, we can compute the required joint accelerations of the robot to track a desired trajectory of manipulability ellipsoids as its main task with

$$\ddot{\boldsymbol{q}} = (\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}} \Big(\text{mvec}(\ddot{\boldsymbol{M}}) - \dot{\boldsymbol{\mathcal{J}}}^{\mathsf{T}}_{(3)} \dot{\boldsymbol{q}} \Big). \tag{5.17}$$

Similarly as in the classical acceleration-based controller that tracks a desired end-effector trajectory, we can define a controller to track a reference manipulability ellipsoid trajectory based on (5.17). To do so, we exploit the geometry of the SPD manifold to compute the difference between the current manipulability ellipsoid $\boldsymbol{M}_t$ and the desired one $\hat{\boldsymbol{M}}_t$, as previously specified for the velocity-based controller. Moreover, since the first-order time derivative of manipulability ellipsoids lies on the tangent space of the SPD manifold (i.e. the space of symmetric matrices Sym^d), the difference between the current manipulability velocity $\dot{\boldsymbol{M}}_t$ and the desired one $\widehat{\dot{\boldsymbol{M}}_t}$ is computed as a subtraction in the Euclidean space. Therefore, a reference manipulability acceleration command can be specified by

$$\text{mvec}(\ddot{\boldsymbol{M}}_t) = \boldsymbol{K}_p \text{mvec}\Big(\text{Log}_{\boldsymbol{M}_t}(\hat{\boldsymbol{M}}_t) \Big) + \boldsymbol{K}_d \text{mvec}\Big(\widehat{\dot{\boldsymbol{M}}_t} - \dot{\boldsymbol{M}}_t \Big), \tag{5.18}$$

which resembles a proportional-derivative controller where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are gain matrices. Then, the reference joint acceleration $\ddot{\boldsymbol{q}}$ can be computed using (5.17) and (5.18). Note that this reference joint acceleration can correspond to a main task of the robot or to a secondary tracking objective. In the latter case, a manipulability-based redundancy resolution can also be implemented in a similar way as (5.8).

5.2.3 Actuators Contribution

In many practical applications, the joint velocities of the robot are limited. The definition of manipulability ellipsoid can then be extended to include these actuation constraints, as shown in [Lee, 1997]. We here provide the definition of the force, velocity and dynamic manipulability ellipsoids and the corresponding manipulability Jacobians considering joint actuation constraints.

To include the joint velocity constraints of the robot in the definition of the velocity manipulability ellipsoid, we use the following weighted forward kinematics formulation

$$\dot{\boldsymbol{x}} = \underbrace{(\boldsymbol{J}\boldsymbol{W}^{\dot{q}})}_{\tilde{\boldsymbol{J}}} \underbrace{(\boldsymbol{W}^{\dot{q}-1}\dot{\boldsymbol{q}})}_{\dot{\tilde{\boldsymbol{q}}}}, \tag{5.19}$$

where $\boldsymbol{W}^{\dot{q}} = \text{diag}(\dot{q}_{1,\max}, \ldots, \dot{q}_{n,\max})$ is a diagonal matrix whose elements correspond to the maximum joint velocities of the robot. Then, considering the set of joint velocities of constant unit norm $\|\tilde{\dot{\boldsymbol{q}}}\| = 1$ mapped into the Cartesian velocity space through

$$\|\tilde{\dot{\boldsymbol{q}}}\|^2 = \tilde{\dot{\boldsymbol{q}}}^{\mathsf{T}}\tilde{\dot{\boldsymbol{q}}} = \dot{\boldsymbol{x}}^{\mathsf{T}}(\tilde{\boldsymbol{J}}\tilde{\boldsymbol{J}}^{\mathsf{T}})^{-1}\dot{\boldsymbol{x}}, \tag{5.20}$$

the velocity manipulability ellipsoid is given by $\tilde{\boldsymbol{M}}^{\dot{\boldsymbol{x}}} = \tilde{\boldsymbol{J}}\tilde{\boldsymbol{J}}^{\mathsf{T}} = \boldsymbol{J}\boldsymbol{W}^{\dot{\boldsymbol{q}}}\boldsymbol{W}^{\dot{\boldsymbol{q}}\mathsf{T}}\boldsymbol{J}^{\mathsf{T}}$, which represents the flexibility of the manipulator in generating velocities in Cartesian space considering its maximum joint velocities as illustrated in Figure 5.3a. Note that the actuators contribution $\boldsymbol{W}^{\dot{\boldsymbol{q}}}\boldsymbol{W}^{\dot{\boldsymbol{q}}\mathsf{T}}$ also has a geometrical interpretation based on the fact that the robot joint position $\boldsymbol{q}$ lies on the flat n-torus manifold [Park, 1995].

By following the methodology of Section 5.2.1, the change in the robot manipulability ellipsoid is related to the joint velocity via

$$\frac{\partial \tilde{\boldsymbol{M}}(t)}{\partial t} = \tilde{\boldsymbol{\mathcal{J}}}(\boldsymbol{q}) \times_3 \dot{\boldsymbol{q}}^{\mathsf{T}}. \tag{5.21}$$

Therefore, the velocity manipulability Jacobian including joint velocity limits is given by

$$\tilde{\boldsymbol{\mathcal{J}}}^{\dot{\boldsymbol{x}}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{J}\boldsymbol{W}^{\dot{\boldsymbol{q}}}\boldsymbol{W}^{\dot{\boldsymbol{q}}\mathsf{T}} + \frac{\partial \boldsymbol{J}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J}\boldsymbol{W}^{\dot{\boldsymbol{q}}}\boldsymbol{W}^{\dot{\boldsymbol{q}}\mathsf{T}}. \tag{5.22}$$

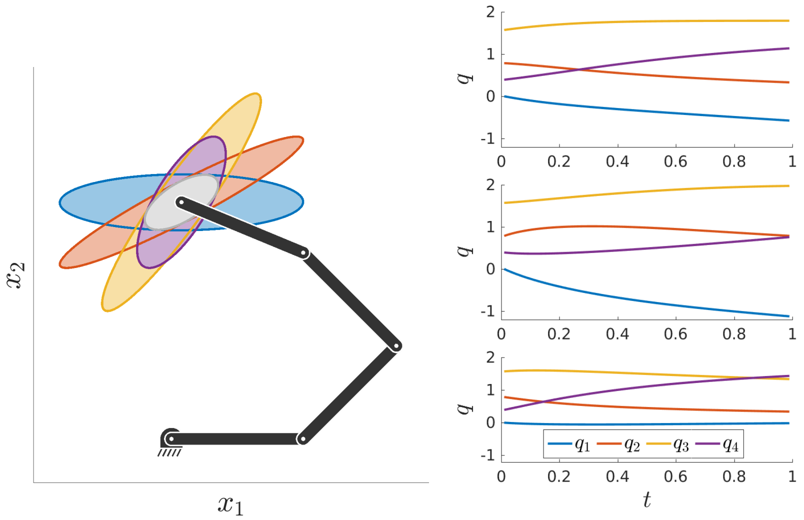
Figure 5.3b shows the effect of including the actuator contribution when tracking a velocity manipulability ellipsoid. Notice that the robot joint q_1 significantly moves when given the highest velocity limit. In contrast, its influence on the manipulability tracking task is minimal when given the lowest velocity limit. This demonstrates the importance of considering the robot actuator specifications when tracking manipulability ellipsoids in real platforms.

In a similar way, the force manipulability ellipsoid considering the maximum joint torques is defined as $\tilde{\boldsymbol{M}}^{\boldsymbol{F}} = (\boldsymbol{J}\boldsymbol{\Omega}^{\boldsymbol{\tau}}\boldsymbol{J}^{\mathsf{T}})^{-1}$, where $\boldsymbol{\Omega}^{\boldsymbol{\tau}} = (\boldsymbol{W}^{\boldsymbol{\tau}}\boldsymbol{W}^{\boldsymbol{\tau}\mathsf{T}})^{-1}$ and $\boldsymbol{W}^{\boldsymbol{\tau}} = \operatorname{diag}(\tau_{1,\max}, \ldots, \tau_{n,\max})$. Then, the corresponding manipulability Jacobian is given by

$$\tilde{\boldsymbol{\mathcal{J}}}^{\boldsymbol{F}} = -\left(\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{J}\boldsymbol{\Omega}^{\boldsymbol{\tau}} + \frac{\partial \boldsymbol{J}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J}\boldsymbol{\Omega}^{\boldsymbol{\tau}}\right) \times_1 \tilde{\boldsymbol{M}}^{\boldsymbol{F}} \times_2 \tilde{\boldsymbol{M}}^{\boldsymbol{F}}.$$

Finally, the dynamic manipulability ellipsoid considering the maximum joint torques is $\tilde{\boldsymbol{M}}^{\ddot{\boldsymbol{x}}} = \boldsymbol{\Upsilon}\boldsymbol{\Omega}^{\boldsymbol{\tau}-1}\boldsymbol{\Upsilon}^{\mathsf{T}}$ with corresponding manipulability Jacobian defined as

$$\tilde{\boldsymbol{\mathcal{J}}}^{\ddot{\boldsymbol{x}}} = \frac{\partial \boldsymbol{\Upsilon}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{\Upsilon}\boldsymbol{\Omega}^{\boldsymbol{\tau}-1} + \frac{\partial \boldsymbol{\Upsilon}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{\Upsilon}\boldsymbol{\Omega}^{\boldsymbol{\tau}-1}. \tag{5.23}$$



(a) (b)

Figure 5.3 – Illustration of actuators contribution. (*a*) Velocity manipulability ellipsoids obtained when setting a maximum joint velocity, for each joint, five times higher than the rest. The manipulability corresponding to equal maximum joint velocity is shown in gray. (*b*) Joint trajectories obtained with manipulability tracking (as in Fig. 5.1) for equal maximum joint velocities (*top*), highest velocity limit for q_1 (*middle*), and lowest velocity limit for q_1 (*bottom*).

5.2.4 Exploiting 4th-order Precision Matrix as Controller Gain

An open problem regarding the proposed tracking approach is how to specify the values of the gain matrix $\boldsymbol{K_M}$, which basically determines how the manipulability tracking error affects the resulting joint velocities. In this sense, we propose to define $\boldsymbol{K_M}$ as a precision matrix, which describes how accurately the robot should track a desired manipulability ellipsoid. In learning from demonstration applications, such gain matrix would typically be set as proportional to the inverse of the observed covariance $\boldsymbol{\mathcal{S}}$ (see Chapter 4). This encapsulates variability information of the task to be learned. Our goal here is to exploit this information to demand the robot a high precision tracking for directions in which low variability is observed, and vice-versa.

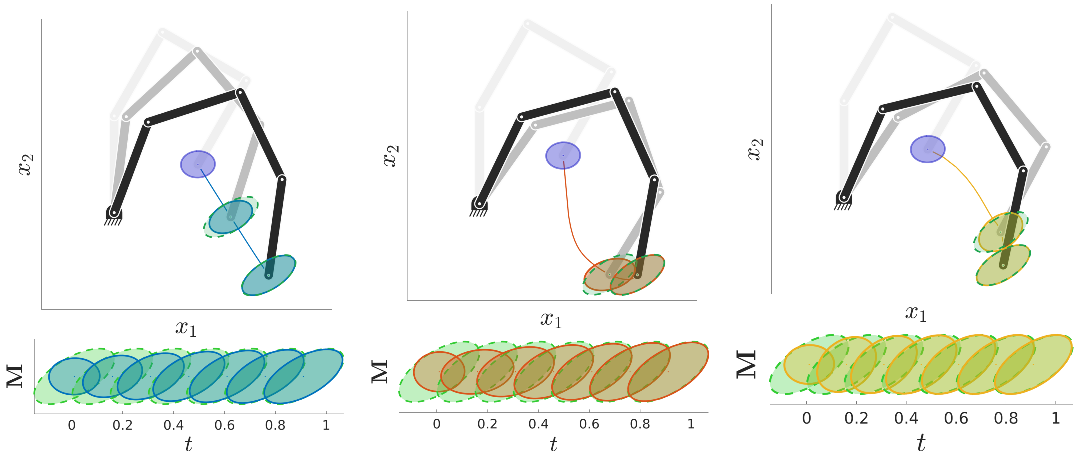
We therefore introduce the required precision $\boldsymbol{\mathcal{S}}^{-1}$ for a given manipulability tracking task into the controllers defined in Section 5.2.2. To do so, we define the gain matrix $\boldsymbol{K_M}$ as a function of the precision tensor. Specifically, we define the controller gain matrix as a full SPD matrix, which is computed from the matricization of the precision tensor $\boldsymbol{\mathcal{S}}^{-1}$ along its two first dimensions, with a proportion defined by

$$\boldsymbol{K_M} \propto \mathcal{S}^{-1}_{(1,2)}. \tag{5.24}$$

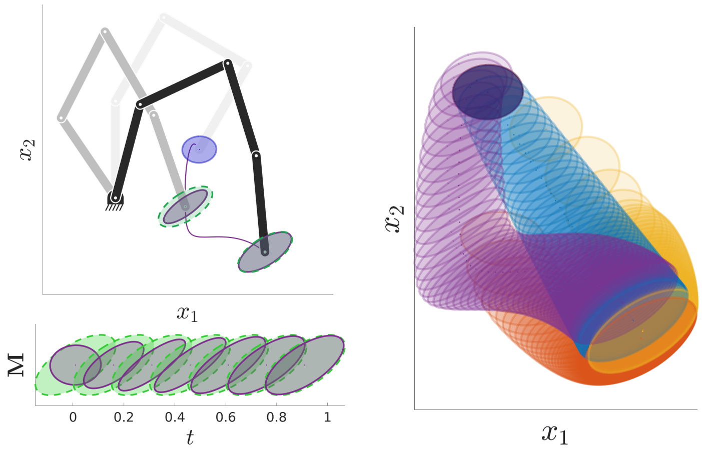
To show how precision matrices work as controller gains in our manipulability tracking problem, we tested different forms of $\boldsymbol{K_M}$ aimed at reproducing a given manipulability ellipsoid as a main task with a simulated 4-DoF planar robot. The robot is required to move its joints to track a desired manipulability ellipsoid, where the controller gain matrix $\boldsymbol{K_M}$ is a diagonal matrix with the diagonal elements of (5.24) to take into account the variation of each component of the manipulability ellipsoid. We tested four different precision tensors. First, equal variability for all components of the manipulability ellipsoid matrix is given. Then, the variability along the first or the second main axis of the manipulability ellipsoid, corresponding to the first and second diagonal elements of the gain matrix $\boldsymbol{K_M}$, is reduced. This means that the robot needs to prioritize the tracking of one of the ellipsoid main axes over the other. In the fourth test, the variability of the correlation between the two main axes of the manipulability ellipsoid is lowered. In this last case, the manipulability controller prioritizes the tracking of the ellipsoid orientation over the shape.

Figure 5.4 shows how the manipulator posture is adapted to track the desired manipulability ellipsoid with a priority on the component with the lowest variability. Note that, when high tracking precision is required for one of the main axes of the ellipsoid, the robot initially seeks to fit the shape of the ellipsoid along that specific axis, and subsequently it matches the whole manipulability ellipsoid. In this case, the precision ratio between the prioritized and the rest of components of the gain matrix is $10:1$. When high tracking precision is assigned to the correlation of the ellipsoid axes, the robot first tries to align its manipulability with the orientation of the desired ellipsoid, and afterwards the whole manipulability is matched. In this case, the precision ratio between the prioritized correlation and the other components of the gain matrix is $3:1$. Notice that the precision tensor naturally affects the computed joint velocities required to track a given ellipsoid, which consequently influences the resulting motion of the end-effector as a function of the precision constraints, as shown in Figure 5.4e. After convergence, the desired manipulability ellipsoid is successfully matched for all experiments. These results show that our geometry-aware tracking permits to take into account the variability information of a task to define the manipulability tracking precision.

Therefore, our manipulability tracking approach may be readily combined with the manipulability learning framework introduced in Chapter 4. In order to illustrate this, we show the reproduction phase of the experiment carried out in Section 4.4. The 5-DoF *student* robot was requested to track a desired Cartesian trajectory as main task, while varying its joint configuration for matching desired manipulability ellipsoids as secondary task. The student robot used the geometry-aware controller defined by (5.8), where $\boldsymbol{K_M}$ was defined either as a scalar value or as a diagonal matrix with the diagonal elements of (5.24) with the precision tensor being equal to the inverse of the covariance tensor $\hat{\boldsymbol{\mathcal{S}}}^{\mathcal{YY}}_{\mathcal{YY}}$ retrieved by GMR (3.28). Our goal here was to exploit the learned variability information of the task to demand the robot a high precision tracking where low variability was observed in the demonstrations, and vice-versa. Successful reproductions of the demonstrated task using our manipulability-based redundancy resolution controller with

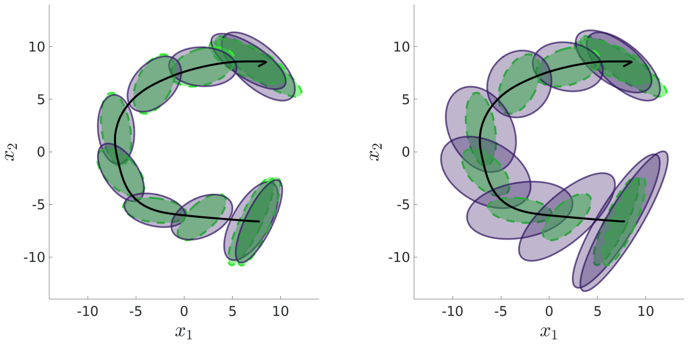


(a) Uniform tracking precision (b) High precision for x_1 (c) High precision for x_2



(d) High precision in correlation (e) Trajectories comparison

Figure 5.4 – Manipulability tracking as main task with diagonal gain matrices defined from different precision tensors. The *top* plots depict the end-effector trajectory (solid colored line) and the posture of the robot along with the corresponding manipulability at time $t = 0$, 0.25 and 1s. The evolution of the manipulability along time is shown in the *bottom* plots. (a): equal tracking precision for all components. (b) and (c): tracking precision is 10 : 1 higher for x_1 and x_2, respectively. (d): correlation between x_1 and x_2 axes is assigned a high tracking accuracy. (e) Evolution of the robot manipulability and end-effector trajectory for the gain matrices used in (a)-(d). The colors match those of the previous graphs. Initial and desired manipulability ellipsoids are depicted in dark blue and green on all graphs. Time t is in seconds.



(a) $\boldsymbol{K_M}$ as a scalar value (b) $\boldsymbol{K_M}$ retrieved from GMR

Figure 5.5 – Reproductions of a learned C-shape tracking task with desired manipulability ellipsoids. The end-effector trajectory is shown in black solid line, while the desired and reproduced manipulabilities are depicted in green and dark purple, respectively. (*a*) $\boldsymbol{K_M}$ is a scalar value, (*b*) $\boldsymbol{K_M}$ is the diagonal of the precision tensor retrieved by GMR. The required tracking precision is higher at the start and end of the task as a consequence of the low observed variability.

scalar and variability-based matrix gains are shown in Figures 5.5a and 5.5b, respectively. Note that the variability-based matrix gain changes the required tracking precision, where higher precision is enforced only at the beginning and the end of the task, which results in lower control efforts in between. These results validate that the proposed approach allows the robot to reproduce reference profiles of desired manipulability ellipsoids while adapting the tracking precision according to the demonstrated requirements of the task.

5.2.5 Nullspace of the Manipulability Jacobian

As traditionally done when designing redundancy resolution controllers, the nullspace of the manipulability Jacobian can also be exploited to fulfill secondary objectives when manipulability tracking is the main task or corresponds to a task with higher priority in multi-task scenarios. More specifically, a joint velocity $\dot{\boldsymbol{q}}_N$, aimed at fulfilling secondary objectives, can be projected into the nullspace of our manipulability tracking controller (5.7) using the nullspace operator $\Big(\boldsymbol{I} - (\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}} \boldsymbol{\mathcal{J}}^{\mathsf{T}}_{(3)}\Big)$. Therefore, the resulting redundancy resolution controller is given by

$$\dot{\boldsymbol{q}}_t = (\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}} \boldsymbol{K_M} \,\mathrm{mvec}\Big(\mathrm{Log}_{\boldsymbol{M}_t}(\hat{\boldsymbol{M}}_t)\Big) + \Big(\boldsymbol{I} - (\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}} \boldsymbol{\mathcal{J}}^{\mathsf{T}}_{(3)}\Big)\dot{\boldsymbol{q}}_N. \tag{5.25}$$

In order to show the functionality of this nullspace operator, we carried out experiments with a simulated 6-DoF planar robot. The main task of the robot is to track a desired manipulability ellipsoid while keeping a desired pose for its first joint q_0, which is considered as secondary task. Thus, the nullspace velocity is defined as a simple proportional

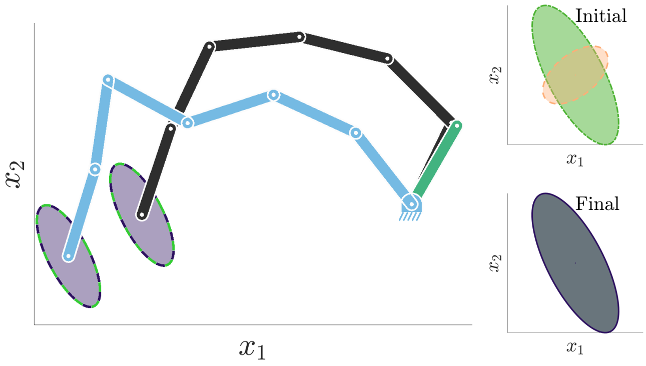


Figure 5.6 – Use of the nullspace of the manipulability Jacobian. Two 6-DoF planar robots are required to track a desired manipulability ellipsoid as main task (green ellipses). The black robot also keeps its first joint at a fixed position (depicted by the green link), which is a secondary objective projected into the nullspace of the manipulability Jacobian. The final manipulability ellipsoids (in purple) fully overlap the desired ones, showing a precise manipulability tracking. The initial manipulability ellipsoid is depicted in yellow.

controller $\dot{\boldsymbol{q}}_N = \boldsymbol{K}_{\boldsymbol{q}}^{\mathcal{P}}(\hat{\boldsymbol{q}} - \boldsymbol{q}_t)$ where $\hat{\boldsymbol{q}}$ is the desired joint configuration and $\boldsymbol{K}_{\boldsymbol{q}}^{\mathcal{P}}$ is a matrix gain defined so that only joint position errors in the first joint are compensated. Figure 5.6 shows that the black manipulator configuration is adjusted to track the desired manipulability ellipsoid and keep, as accurately as possible, the desired joint position for q_0. Note that the black robot is able to find an alternative joint configuration that permits not only to closely track the desired manipulability, but also fulfill secondary objectives projected into its nullspace, in contrast to the blue robot which exclusively implements a manipulability tracking task. These results show that the nullspace of the manipulability Jacobian is suitable to carry out a secondary task along with manipulability tracking as main objective.

5.3 Importance of Geometry-awareness for Manipulability Tracking

In the previous sections we introduced different types of geometry-aware manipulability tracking controllers. In this section, we show that the geometry-awareness of our formulations is crucial for successfully tracking manipulability ellipsoids in addition to providing an appropriate mathematical treatment of the problem.

5.3.1 Comparisons with Euclidean Tracking

We first compare the proposed tracking formulation against a controller ignoring the geometry of SPD matrices (i.e., treating the problem as Euclidean). Moreover, we evaluate our controller when the tracking of manipulability ellipsoids is assigned a secondary role. This evaluation compares our formulation against three Euclidean controllers, and the gradient-based approach presented in [Rozo et al., 2017]. For the case in which the manipulability tracking is the main objective, we consider a 4-DoF planar robot that is required to track a desired manipulability ellipsoid by minimizing the error between its current and desired manipulability ellipsoids $\boldsymbol{M}$ and $\hat{\boldsymbol{M}}$. We first compare the proposed approach (5.7) with the following Euclidean manipulability tracking controller

$$\dot{\boldsymbol{q}}_t = (\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}} \boldsymbol{K_M} \mathrm{mvec}(\hat{\boldsymbol{M}}_t - \boldsymbol{M}_t), \tag{5.26}$$

where the difference between two manipulability ellipsoids is computed in Euclidean space, i.e., ignoring that manipulability ellipsoids belong to the set of SPD matrices. Secondly, we compare the proposed approach to the Cholesky-based Euclidean manipulability controller

$$\dot{\boldsymbol{q}}_t = (\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}} \boldsymbol{K_M} \mathrm{mvec}(\boldsymbol{\Delta L}_t \boldsymbol{\Delta L}_t^{\mathsf{T}}), \tag{5.27}$$

where $\boldsymbol{\Delta L} = \hat{\boldsymbol{L}} - \boldsymbol{L}$ and matrices $\boldsymbol{L}$ are obtained from the Cholesky decomposition of $\boldsymbol{M}$, such that $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$. This controller ensures that the difference between two manipulability ellipsoids is positive definite, but ignores that they belong to the SPD manifold. For completeness, we also compare our approach with the Cholesky-Jacobian-based Euclidean manipulability controller

$$\dot{\boldsymbol{q}}_t = (\mathcal{J}^{\dagger}_{\mathrm{chol}(3)})^{\mathsf{T}} \boldsymbol{K_M} \ \mathrm{vec}(\hat{\boldsymbol{L}} - \boldsymbol{L}), \tag{5.28}$$

where $\mathcal{J}_{\mathrm{chol}} = \frac{\delta \boldsymbol{L}}{\delta \boldsymbol{q}} = \frac{\delta \boldsymbol{L}}{\delta \boldsymbol{M}} \boldsymbol{\mathcal{J}}$ is the Cholesky-based manipulability Jacobian, so that $\dot{\boldsymbol{L}} = \mathcal{J}_{\mathrm{chol}} \times_3 \dot{\boldsymbol{q}}^{\mathsf{T}}$. This approach tracks a desired manipulability solely through its Cholesky decomposition with an adapted manipulability Jacobian. Similarly to (5.27), it ensures the positive-definiteness of manipulability ellipsoids, but ignores that they belong to the SPD manifold. For all the following comparisons, the gain matrices $\boldsymbol{K_M}$ are identity matrices.

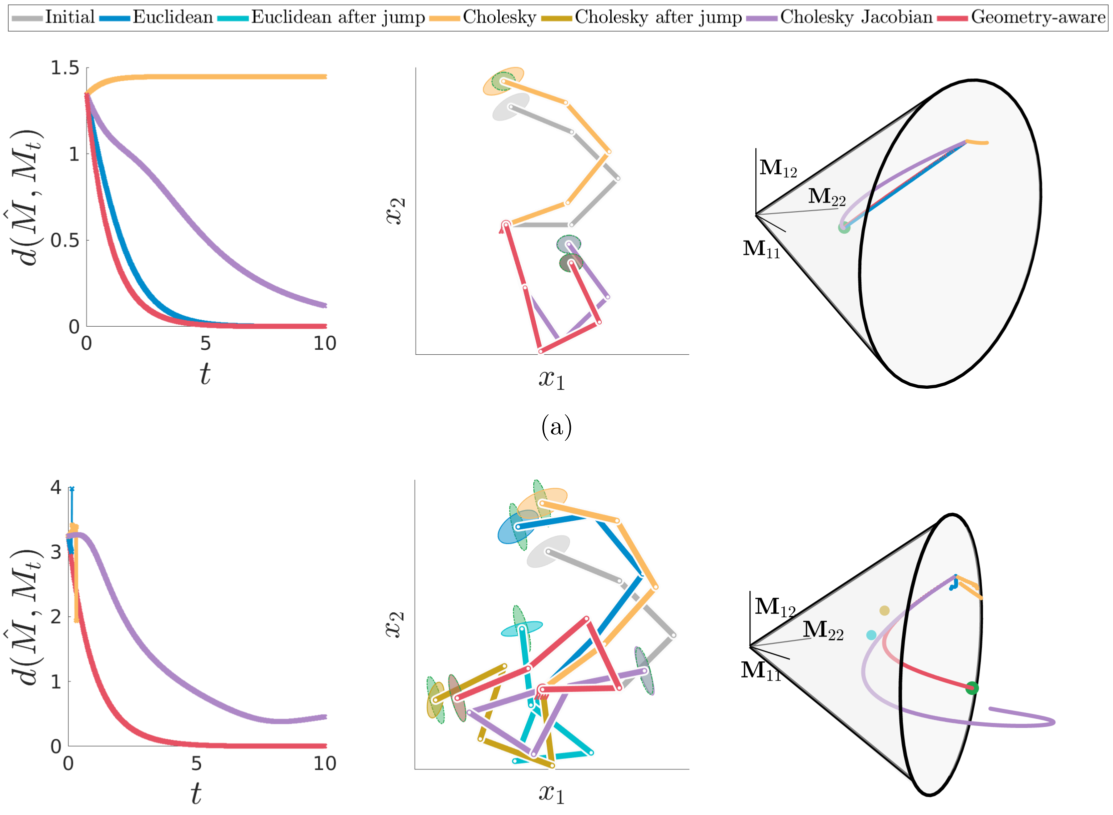
Figure 5.7 shows the convergence rate for the proposed geometry-aware controller, the Euclidean-based approach, the Cholesky-based Euclidean and Cholesky-Jacobian-based Euclidean formulations. Two tests were carried out by varying the initial configuration of the robot and the desired manipulability ellipsoid. In the first case, the Euclidean and geometry-aware formulations converge to similar robot joint configurations with a distance between the current and desired manipulability close to zero (see Figures 5.7a-*left*, *middle* and Table 5.2). However, in the second test, the Euclidean formulation induces a sudden change in the joint configuration, resulting in an abrupt increase on the error measured between the current and desired manipulability ellipsoids (see Figures 5.7b-*left*, *middle*). In real scenarios, such unstable robot behavior would certainly be harmful and unsafe. This erroneous tracking performance can be explained by the fact that the Euclidean path between two SPD matrices is a valid approximation of the geodesic only if these are close enough to each other, as shown in Figure 5.7a-*right*. When this approximation is

Table 5.2 – Final distances $d_{\mathcal{S}^d_{++}}(\hat{\boldsymbol{M}}, \boldsymbol{M}_t)$ between the current and desired manipulability ellipsoids for the performance comparison of the different manipulability tracking formulations.

Approach	Euclidean	(after jump)	Cholesky	(after jump)	Cholesky Jacobian	Geometry-aware
Fig. 5.7a	$1.3e^{-4}$	-	1.446	-	0.1204	$6e^{-5}$
Fig. 5.7b	2.997	3.977	3.385	1.944	0.455	$1.4e^{-4}$

not valid (see Figure 5.7b-*right*), the Euclidean controller outputs inconsistent reference joint velocities that destabilize the robotic system, therefore failing to track the desired manipulability. Note that the Cholesky-based Euclidean formulation does not converge in both cases (see Table 5.2) and induces a sudden change in joint configuration of the robot in the second scenario, similarly to the Euclidean formulation. This can be explained by the fact that the path induced by this method is not close to geodesics on the SPD manifold as shown by Figure 5.7-*right*. As opposed to the two Euclidean formulations, the Cholesky-Jacobian-based Euclidean controller does not induce unstable robot behaviors and converges towards the desired manipulability ellipsoid for both cases. However, this method shows a poor convergence rate compared to our geometry-aware approach, as shown by Fig. 5.7-*left*. This can be explained by the fact that, although this approach generates curved paths on the SPD manifold, these paths do not resemble geodesics and tend to induce detours to reach the desired manipulability ellipsoid (see Fig. 5.7-*left*). This is particularly visible for the second test, where the resulting joint configuration is farther from the initial pose of the robot compared to the joint configuration obtained by the proposed geometry-aware controller (see Fig. 5.7b-*middle*, *right*).

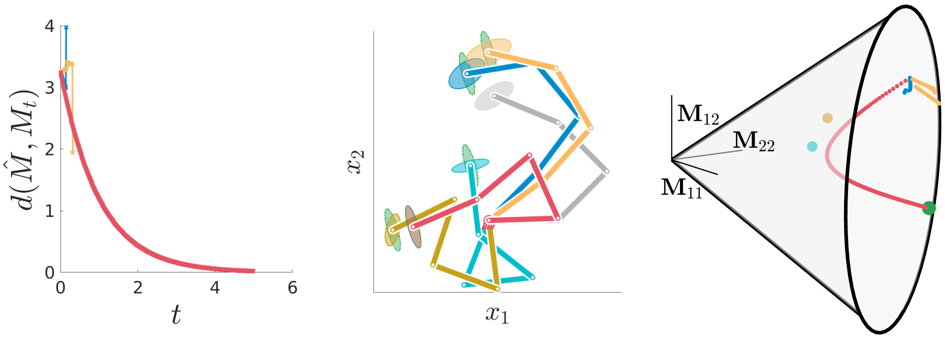
Previously, we hypothesized that the sudden changes in joint configuration when using the Euclidean and Cholesky-based Euclidean formulations in the second scenario are due to the path induced by the methods on the SPD manifold. In order to confirm this hypothesis, we reproduced the second test with lower gain values. Figure 5.8 shows the convergence of the proposed geometry-aware controller, the Euclidean-based approach and the Cholesky-based Euclidean formulation for gain matrices equal to $\boldsymbol{I}$, $0.5\boldsymbol{I}$, $0.1\boldsymbol{I}$ and $0.05\boldsymbol{I}$. We observe that, even for very low gains, both Euclidean and Cholesky-based Euclidean formulations lead to a sudden change in the joint configuration, resulting in an abrupt increase on the error measured between the current and desired manipulability ellipsoids (see Figure 5.8-*left*, *middle*). Interestingly, the sudden changes occur at similar location along the path between the initial and desired manipulability ellipsoid independently of the gain value for both formulations (see Figure 5.8-*right*), therefore confirming our above statement. This can also be seen by looking at the yellow and dark blue robots of Figure 5.8-*middle* depicting the configurations before the jump, which are almost identical in all the graphs.



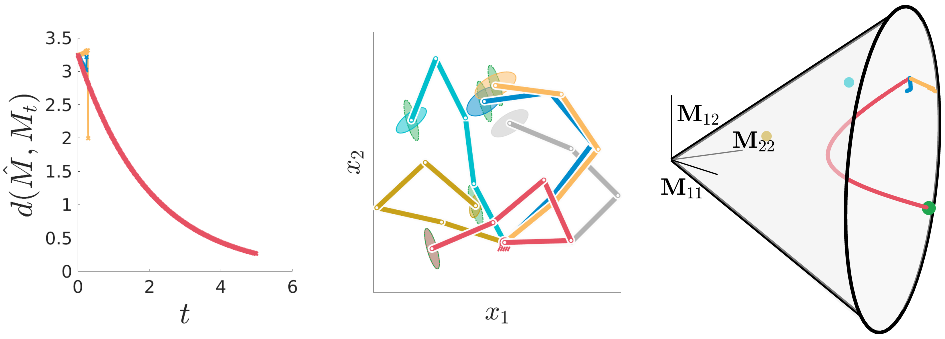
(a)

(b)

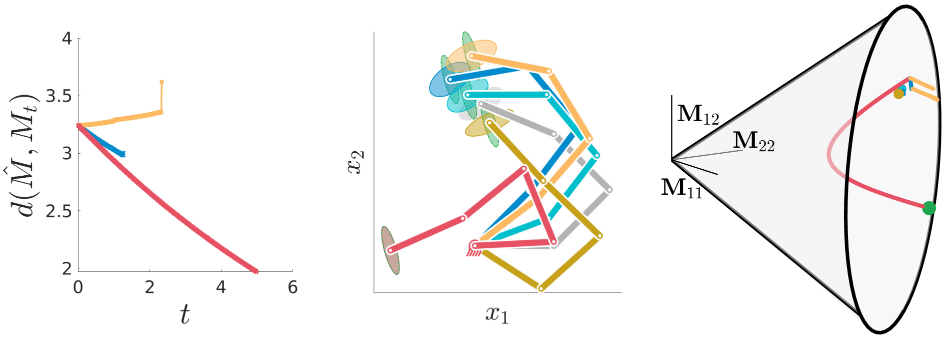
Figure 5.7 – Performance of different manipulability tracking formulations. The *left* graphs show the affine-invariant distance between the current and desired manipulability ellipsoids over time. The distances for the Euclidean, Cholesky-based Euclidean, Cholesky-Jacobian-based Euclidean and geometry-aware approaches are respectively depicted in blue, yellow, lila and red. The *middle* graphs display the initial and final robot postures and the final manipulability ellipsoids. The initial posture is depicted in light gray, while the final posture and corresponding manipulability for the three methods are depicted in the same color as the distances. The desired manipulability is depicted in green. *Middle-(b)* also shows the sudden change in the robot posture for both Euclidean methods (5.26) and (5.27). The robot posture before and after the abrupt change is shown in blue and light blue, respectively for (5.26) and in yellow and olive, respectively for (5.27). The *right* graphs depict the evolution of the manipulability ellipsoids in the SPD manifold. The colors correspond to those of the previous graphs with the green dot representing the desired manipulability. The isolated light blue and olive dots in the *right-(b)* graph represent the manipulability ellipsoids after the abrupt changes in the robot joint configuration.



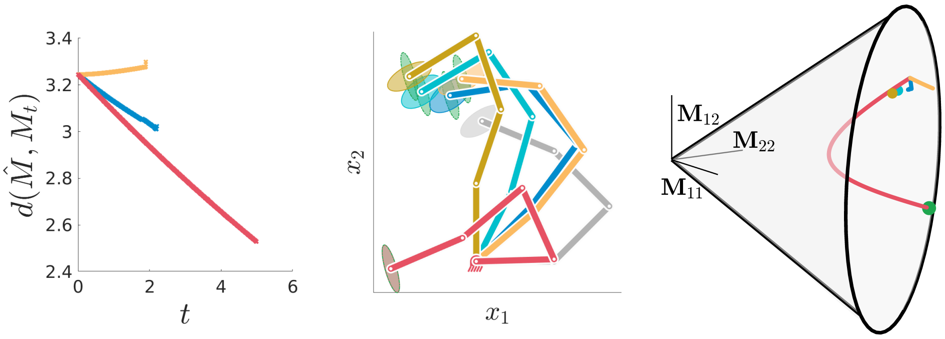
(a) $\boldsymbol{K_M} = \boldsymbol{I}$



(b) $\boldsymbol{K_M} = 0.5\boldsymbol{I}$



(c) $\boldsymbol{K_M} = 0.1\boldsymbol{I}$



(d) $\boldsymbol{K_M} = 0.05\boldsymbol{I}$

Figure 5.8 – Comparison of the performance of different manipulability tracking formulations for different gains $\boldsymbol{K_M}$.The organization of the graphs and the colors are identical to Fig. 5.7. The Cholesky-Jacobian-based Euclidean formulation is not shown.

In the case in which the manipulability tracking task becomes a secondary objective, the 4-DoF planar robot is required to keep its end-effector at a fixed Cartesian position $\hat{\boldsymbol{x}}$ while minimizing the distance between its current and desired manipulability ellipsoids $\boldsymbol{M}$ and $\hat{\boldsymbol{M}}$. The four following approaches are considered for comparison with the proposed formulation (5.8). Firstly, we analyze the corresponding Euclidean manipulability-tracking controller

$$\dot{\boldsymbol{q}}_t = \underbrace{\boldsymbol{J}^{\dagger}\boldsymbol{K_x}(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t)}_{\text{main task}} + \underbrace{(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})}_{\text{nullspace op.}}\ \underbrace{(\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}}\boldsymbol{K_M}\text{mvec}(\hat{\boldsymbol{M}}_t - \boldsymbol{M}_t)}_{\text{secondary task}}, \tag{5.29}$$

where the difference between two manipulability ellipsoids is computed in Euclidean space, i.e., ignoring that manipulability ellipsoids belong to the set of SPD matrices. Secondly, we implement the corresponding Cholesky-based Euclidean manipulability controller

$$\dot{\boldsymbol{q}}_t = \boldsymbol{J}^{\dagger}\boldsymbol{K_x}(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})(\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}}\boldsymbol{K_M}\text{mvec}(\boldsymbol{\Delta L}_t\boldsymbol{\Delta L}_t^{\mathsf{T}}), \tag{5.30}$$

which ignores that manipulability ellipsoids lie on the SPD manifold but ensure a positive definite difference between two ellipsoids. Thirdly, we analyze the Cholesky-Jacobian-based Euclidean manipulability controller

$$\dot{\boldsymbol{q}}_t = \boldsymbol{J}^{\dagger}\boldsymbol{K_x}(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t) + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})(\boldsymbol{\mathcal{J}}^{\dagger}_{\text{chol}(3)})^{\mathsf{T}}\boldsymbol{K_M}\text{vec}(\hat{\boldsymbol{L}} - \boldsymbol{L}), \tag{5.31}$$

which tracks manipulability ellipsoids through their Cholesky decomposition. Fourthly, we evaluate the gradient-based approach of [Rozo et al., 2017] that implements the controller

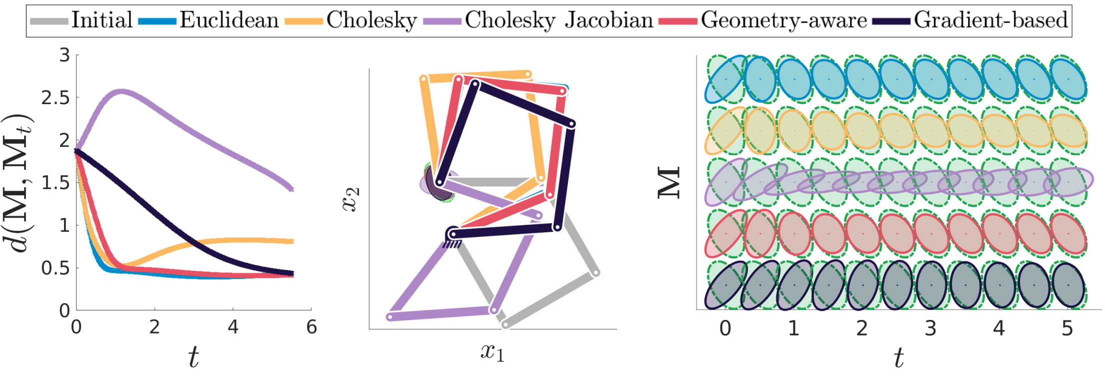
$$\dot{\boldsymbol{q}}_t = \boldsymbol{J}^{\dagger}\boldsymbol{K_x}(\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t) - (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})\alpha\nabla g_t(\boldsymbol{q}), \tag{5.32}$$

where α is a scalar gain and

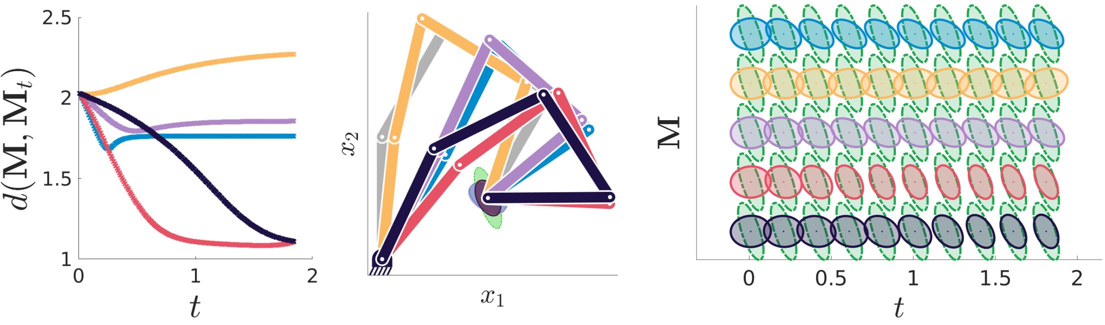
$$g_t(\boldsymbol{q}) = \log\det\left(\frac{\hat{\boldsymbol{M}}_t + \boldsymbol{M}_t}{2}\right) - \frac{1}{2}\log\det\left(\hat{\boldsymbol{M}}_t\boldsymbol{M}_t\right) \tag{5.33}$$

is a cost function based on Stein divergence (a distance-like function on the SPD manifold [Sra, 2012]). The gain matrices $\boldsymbol{K_M}$ are fixed as identity matrices and the scalar gain is set to 1 for the comparison.

Figure 5.9 shows the convergence rate for the manipulability-based redundancy resolution of the aforementioned approaches. Two tests were carried out by varying the initial configuration of the robot and the desired manipulability ellipsoid. In both cases, both geometry-aware and gradient-based approaches converge to a similar final robot configuration (see Figures 5.9a, 5.9b-*middle*), with similar values of the affine-invariant distance between the final and desired manipulability ellipsoids (see Figures 5.9a, 5.9b-*left* and Table 5.3). More importantly, the proposed geometry-aware manipulability



(a)



(b)

Figure 5.9 – Performance comparison of the different manipulability-based redundancy resolution formulations. Two cases are shown with varying initial robot configuration and desired manipulability. The *left* graphs show the convergence of the affine invariant distance between the current and the desired manipulability ellipsoid over time. The distances for the Euclidean, Cholesky-based Euclidean, Cholesky-Jacobian-based Euclidean, geometry-aware and gradient-based approaches are respectively depicted in blue, yellow, lila, red, and purple. The *middle* graphs show the initial and final posture of the robot along with the final manipulability ellipsoids. The initial posture of the robot is depicted in light gray. The final postures and the corresponding manipulability ellipsoids for the different methods are depicted in the same color as the distances. The desired manipulability ellipsoid is depicted in green. The *right* graphs show the evolution of the manipulability ellipsoids over time for the different approaches.

Table 5.3 – Final distances $d_{\mathcal{S}^d_{++}}(\hat{\boldsymbol{M}}, \boldsymbol{M}_t)$ between the current and desired manipulability ellipsoids for the performance comparison of the different manipulability-based redundancy resolution formulations.

Approach	Euclidean	Cholesky	Cholesky Jacobian	Geometry-aware	Gradient-based
Fig. 5.9a	0.433	0.808	1.418	0.416	0.436
Fig. 5.9b	1.763	2.271	1.856	1.101	1.110

tracking approach shows a faster convergence than the gradient-based method (see also Figures 5.9a, 5.9b-*right*), with a lower computational cost (3.5 ms and 4.2 ms per time step, with non-optimized Matlab code on a laptop with 2.7GHz CPU and 32 GB of RAM). This notable difference may be attributed to the fact that despite both methods take into account the geometry of manipulability ellipsoids, our approach is more informative about the kinematics of the robot through the use of the manipulability Jacobian $\boldsymbol{\mathcal{J}}(\boldsymbol{q})$.

Note that for some specific initial robot configurations and desired manipulability ellipsoids, the Euclidean manipulability-tracking controller (5.29) shows a slightly faster convergence rate than our method (see Figure 5.9a). However, this Euclidean formulation again leads to unstable behaviors in some configurations (see Figure 5.9b), where the distance between the final and desired manipulability ellipsoids remains high compared to the two geometry-aware approaches. This poor tracking performance can be attributed to the fact that the Euclidean difference between two SPD matrices is an approximation that is only valid if the matrices are close enough to each other. Thus, similarly to Euclidean controller aimed at tracking manipulability ellipsoids as first task (5.26), the Euclidean manipulability-based redundancy resolution is only effective if the current and desired ellipsoids are very similar. Moreover, the distance between the final and desired manipulability ellipsoids remains higher than for the three other methods by using the Cholesky-based Euclidean manipulability-based redundancy resolution. This tendency is similar to the observations made for the tracking of manipulability ellipsoids as main objective and is due to the fact that the controller (5.30) induces paths on the manifold that are not close to geodesics. Furthermore, the Cholesky-Jacobian-based Euclidean controller shows a poor tracking performance for the two considered scenarios. Notably, the distance between the current and desired ellipsoids is largely increased before decreasing slowly in the first case (see Fig. 5.9a). Moreover, in some configurations, the final distance remains high compared to the geometry-aware approaches as shown by Fig. 5.9b. These behaviors are due to the fact that the controller (5.31) does not follow geodesic paths on the SPD manifold.

The reported results supported our hypothesis that geometry-aware manipulability controllers result in good tracking performance while providing stable convergence regardless of the manipulability tracking error. This was observed when manipulability tracking was the main task and a secondary objective of the robot. Moreover, our manipulability-based redundancy resolution approach outperforms the gradient-based method. Furthermore, our controller permits to directly exploit the variability information of a task, given in the form of a 4th-order covariance tensor, through the gain matrix of the controller. This allows the robot to exploit the precision required while tracking a manipulability ellipsoid either as main or secondary objective. This operation is not available in the gradient-based method used for comparison, since the corresponding controller gain is a scalar.

5.3.2 Comparisons with Manipulability-based Optimization

We compare our tracking approach against two state-of-the-art manipulability-based optimization methods widely used to improve robots posture for task execution. We first evaluate our geometry-aware controller against manipulability volume maximization. Then, we compare our controller to the compatibility index maximization [Chiu, 1987], where the distance from the ellipsoid center to its surface is maximized along a specified direction. To do so, we consider two 8-DoF planar robots that are required to track a desired Cartesian velocity trajectory that leads to an L-shape path in the Cartesian space. In order to achieve high dexterity in motion, the first robot is requested to track a desired manipulability ellipsoid whose main axis is elongated along the direction of motion. The second robot varies its posture in order to maximize either the manipulability volume or the compatibility index along the direction of motion.

Figure 5.10a shows the resulting joint configurations and manipulability ellipsoids of the two robots at different stages of the task where the second robot maximizes the manipulability volume as secondary objective. We observe that the main axis of the manipulability ellipsoid obtained with the volume maximization approach is often perpendicular to the direction of motion, which often occurs as this method does not consider any geometric information about the desired manipulability ellipsoid. Also, since the resulting posture leads to ellipsoids that are not consistent with the task requirement (task velocity control directions) and degrade the robot capabilities, this becomes unstable when the gain of the velocity tracking controller is increased to achieve higher Cartesian velocities, as shown in Figure 5.11. Conversely, the robot tracking a desired manipulability ellipsoid successfully completes the task even when higher velocities are required.

The main advantage of maximizing the compatibility index over the volume is that the directions in which the ellipsoid should be elongated are specified. However, this approach favors robot configurations that may be close to singularities as the manipulability ellipsoids corresponding to these posture are flat ellipsoids that can be largely elongated (see Figure 5.10c). This effect exacerbates when the compatibility index maximization is the main task of the robot, as this is not required to match a specific position in Cartesian space. Chiu [1988] extended the compatibility index optimization approach by defining

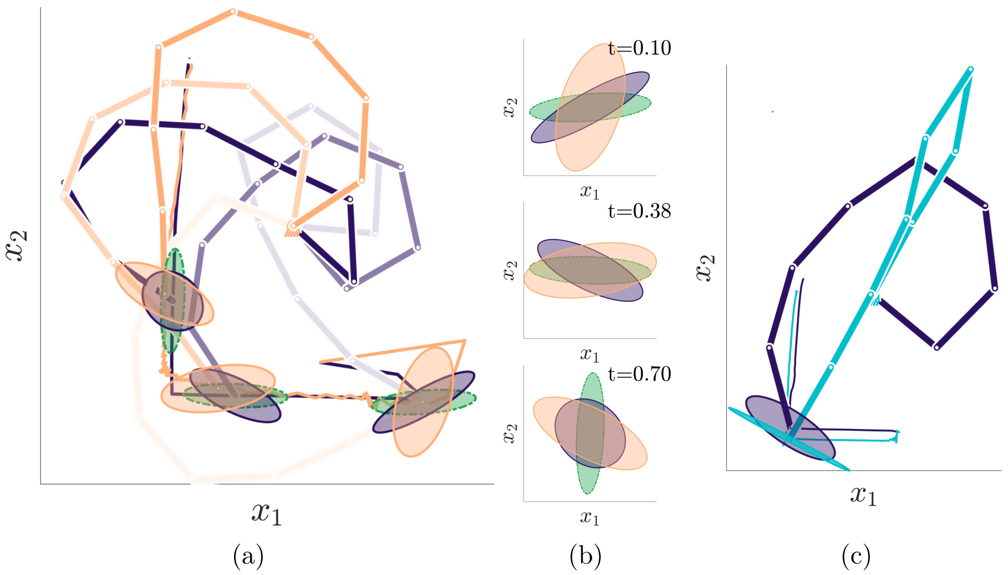


Figure 5.10 – (*a*) Comparison of our manipulability tracking controller (in purple) with the manipulability volume maximization (in yellow). The main axis of the desired manipulability ellipsoids (in green) are aligned with the direction of motion in order to allow high velocities during the task execution. The robot colors become darker with the evolution of the movement. (*b*) Close-up plots of the manipulabilities represented in (*a*). (*c*) Comparison of our manipulability tracking controller (in purple) with the compatibility index maximization (in light blue).

the compatibility cost as a weighted sum, allowing the maximization or minimization of the ellipsoid along several directions. This method provides more flexibility on the resulting ellipsoid due to the weighted combination, at the cost of a laborious tuning. Moreover, the orientation and elongation of the main axes of the ellipsoid after the optimization are hard to infer from the cost weights.

In contrast to the considered manipulability-based optimization methods, the proposed geometry-aware controllers seeks to fit the full desired manipulability ellipsoid in all its directions. Singular configurations can therefore be easily avoided by defining appropriate desired manipulability ellipsoids. Moreover, our manipulability controller allows the tracking of any manipulability ellipsoid, including those providing a compromise between dexterity in motion and force exertion along any axis. This is not possible when using the compatibility index approach as it always favors the dexterity in motion over force or vice-versa. This is not possible when using the compatibility index approach of [Chiu, 1987] as it always favors the dexterity in motion over force or vice-versa. Although this compromise might be achievable using the compatibility index approach of [Chiu, 1988], our method does not require a laborious tuning process. Manipulability tracking is also hard to achieve through manipulability volume maximization as there is no explicit control on the resulting ellipsoid main axes.

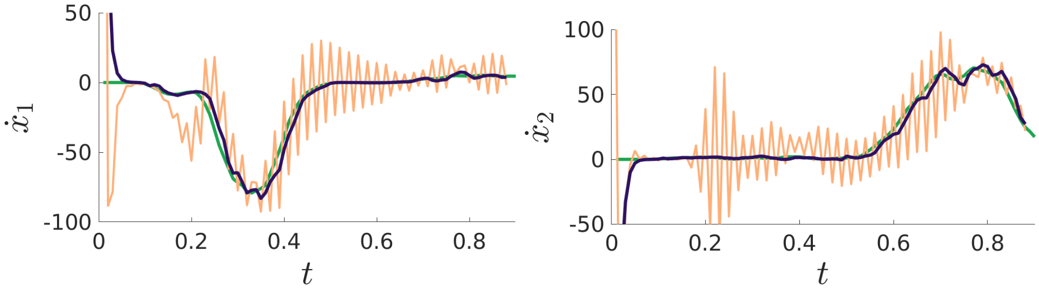


Figure 5.11 – Cartesian velocities achieved with our manipulability tracking controller (purple) and the volume maximization approach (yellow) as secondary objective for a Cartesian velocity controller. The gain of the velocity controller are equal for both approaches. The desired velocities are shown in green.

5.4 Experiments

In this section, we extensively evaluate the proposed tracking formulation with different robotic platforms and different types of manipulability ellipsoids in simulation. The approach is evaluated to track a desired force manipulability in a pushing and an insertion task with a Baxter robot, to track a desired manipulability for grasping with an Allegro hand and to track a desired center of mass manipulability with NAO and Centauro robots.

5.4.1 Manipulability Tracking for a Robotic Arm

The performance of the proposed manipulability tracking framework was first tested in a pushing task and a peg-in-hole task (plugging an electric cable into a power socket), achieved by the 7-DoF arm of the Baxter robot. In the first experiment, the robot is required to track a desired manipulability ellipsoid aligned with a force that is perpendicularly applied to a wall, while the robot end-effector can freely move on the wall plane (see Figure 5.12). This task aims at emulating how humans vary their body posture when applying a force with known direction but unknown amplitude to successfully push an object. In this case, the robot controller is defined as (5.8), where the desired robot position $\hat{\boldsymbol{x}}$ only considers the constraint of being on the wall plane, while $\hat{\boldsymbol{M}}_t$ corresponds to a force manipulability whose main axis is orthogonal to the wall and kept constant over the course of the task.

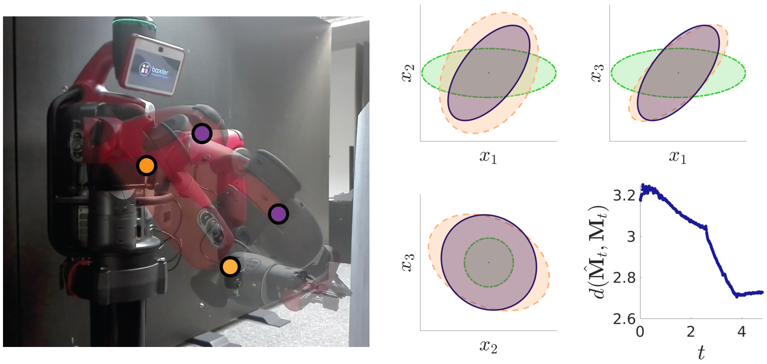
Figure 5.12 shows the resulting manipulability using the redundancy resolution controller (5.8). As expected, the robot modified its joint configuration in order to match, as accurately as possible, the desired force manipulability, therefore adopting a posture compatible with the force requirements of the pushing task. Note that the matching can in this experiment only be achieved partially, because the robot is also required to keep a position constraint at the level of its end-effector, and therefore the joints configuration to match the desired manipulability is restricted to avoid interference with the primary objective. In order to verify that the robot found an appropriate pose to match the desired manipulability while fulfilling the position constraint, we replaced the manipulability tracking term in (5.8) by joint velocity commands driven by Brownian noise. This test, carried out in simulation for ten minutes, allowed us to explore the space of possible poses satisfying the primary objective, for which corresponding manipulability ellipsoids were computed. The minimum distance between the computed and desired manipulability ellipsoids obtained in this test coincided with the distance achieved by our controller.

In the peg-in-hole scenario, the robot first needs to track a specific Cartesian trajectory to approach a hole and subsequently insert a peg into it, as shown in Figure 5.13. Desired manipulability ellipsoids were defined according to the task requirements for the two different parts of the peg-in-hole process. Initially, a desired velocity manipulability is aligned with the direction of motion of the end-effector governed by the reference trajectory. Then, a desired force manipulability is set to be aligned with the force applied to insert the peg, which is executed at the end of the task. Note that both alignments refer to the major axis of the ellipsoids. Similarly as the pushing task, the robot used the redundancy resolution controller (5.8), where $\hat{\boldsymbol{x}}$ was defined as the desired Cartesian trajectory to track, while $\hat{\boldsymbol{M}}_t$ was set based on the aforementioned velocity and force manipulability ellipsoids required by the task.

In order to show the effects of the manipulability tracking controller on the robot posture over the course of the task, we also executed the peg-in-hole experiment with $\boldsymbol{K_M} = \boldsymbol{0}$, which means that the manipulability tracking was fully disabled. A small difference between the robot postures can be observed during the approaching part, which shows how the manipulability controller influences the trajectory tracking phase (see Figure 5.13a). More notably, the robot significantly varied its posture when the insertion part took place, so that its manipulability coincided as accurately as possible with the desired force manipulability ellipsoid (see Figures 5.13b, 5.13c). This variation of the joint configuration consequently allows the robot to adopt a posture compatible with the control force required along the vertical direction of the task.

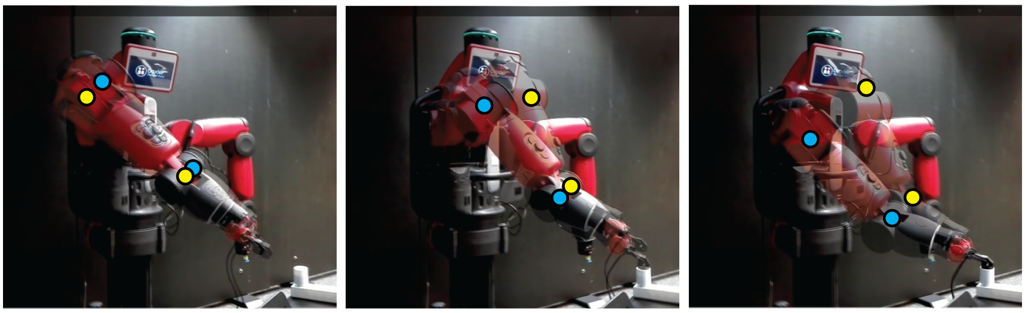
5.4.2 Manipulability Tracking for a Robotic Hand

In the context of robotic hands, manipulability ellipsoids have been used to analyze their performances in grasping tasks [Prattichizzo et al., 2012]. In this experiment, we aim at modifying the posture of a robotic hand to match a desired manipulability ellipsoid while grasping an object. For the case of multiple arm systems, the set of joint velocities of constant unit norm $\|\dot{\boldsymbol{q}}_a\| = \|(\dot{\boldsymbol{q}}_1^{\mathsf{T}}, \ldots, \dot{\boldsymbol{q}}_C^{\mathsf{T}})^{\mathsf{T}}\| = 1$ is mapped to the Cartesian velocity



(a) Initial and final poses (b) Manipulability evolution

Figure 5.12 – Manipulability tracking for a pushing task with the Baxter robot. (*a*) The initial and final pose of the robot are respectively depicted by orange and purple dots on the elbow and wrist bend joints. (*b*) The three first graphs show the initial, final and desired manipulability ellipsoids respectively depicted in yellow, dark purple and green. The *bottom-right* graph shows the evolution of the distance between the current and desired manipulability ellipsoid over time (in seconds).



(a) Approaching phase (b) Preparation phase (c) Insertion phase

Figure 5.13 – Manipulability tracking for an insertion task with the Baxter robot. The poses of the robot obtained with and without manipulability tracking are respectively depicted by yellow and blue dots on the elbow and wrist bend joints when the robot (*a*) approaches the hole, (*b*) prepares to insert the peg, (*c*) ends the insertion of the peg in the hole.

space $\dot{\boldsymbol{x}}_a = (\dot{\boldsymbol{x}}_1^{\mathsf{T}}, \dots, \dot{\boldsymbol{x}}_C^{\mathsf{T}})^{\mathsf{T}}$ through

$$\|\dot{\boldsymbol{q}}_a\|^2 = \dot{\boldsymbol{q}}_a^{\mathsf{T}} \dot{\boldsymbol{q}}_a = \dot{\boldsymbol{x}}_a^{\mathsf{T}} (\boldsymbol{G}_a^{\dagger\mathsf{T}} \boldsymbol{J}_a \boldsymbol{J}_a^{\mathsf{T}} \boldsymbol{G}_a^{\dagger})^{-1} \dot{\boldsymbol{x}}_a, \tag{5.34}$$

with the Jacobian $\boldsymbol{J}_a = \text{diag}(\boldsymbol{J}_1, \dots, \boldsymbol{J}_C)$, the grasp matrix $\boldsymbol{G}_a = (\boldsymbol{G}_1, \dots, \boldsymbol{G}_C)$ and C the number of arms. Therefore, the velocity manipulability ellipsoid of the C-arms system is given by $\boldsymbol{M}_a^{\dot{\boldsymbol{x}}} = \boldsymbol{G}_a^{\dagger\mathsf{T}} \boldsymbol{J}_a \boldsymbol{J}_a^{\mathsf{T}} \boldsymbol{G}_a^{\dagger}$ [Chiacchio et al., 1991a]. Note that the system is modeled under assumptions that the arms are holding a rigid object with a tight grasp. Moreover, we assume C independent kinematic chains for the arms.

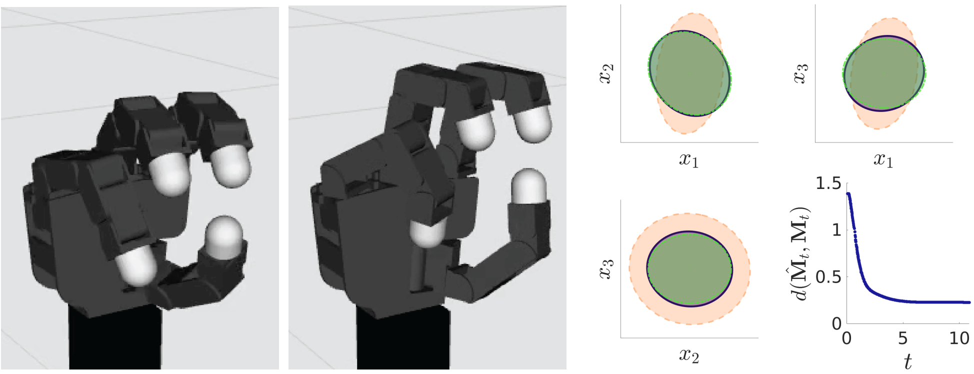
In this first experiment, the Allegro hand was required to track a desired manipulability, while maintaining relative positions between the different fingers. This experiment aims at emulating how humans adapt their finger configuration to the task at hand while grasping an object. In this experiment, the desired velocity manipulability ellipsoid was designed by the experimenter to be a medium-size isotropic ellipsoid. The purpose of this design is to provide the hand with the capability to perform a displacement of the object while being resistant to external perturbations in all the directions. For example, in the case where the hand is holding a pen, it is desirable that the pen can be moved with dexterity, while the hand should resist to perturbations due to the pen-surface contacts.

The fingers were controlled according to a leader-follower strategy [Luh and Zheng, 1987]. Therefore the thumb joints were moved to track the desired manipulability ellipsoid using the controller (5.7) and the other fingers were required to maintain constant relative end-effector positions with respect to the thumb end-effector, while tracking the manipulability as secondary objective with the redundancy controller (5.8). The center of the object was considered as the central position between the four fingers of the hand and the contact points were assumed to be at the finger tips.

Figures 5.14a and 5.14b show an example of adaptation of the posture of the hand to track a desired velocity manipulability ellipsoid for a grasp defined by the user. As expected, the robot modified its joint configuration in order to match, as accurately as possible, the desired velocity manipulability (see Figure 5.14c). Note that the manipulability tracking in this experiment can only be achieved partially, because the robotic hand is also required to maintain the initial grasp. Nevertheless, this tracking may be further improved if the dimensionality of the nullspace of the main task is higher (e.g. not all the finger tips are position-constrained), or using a higher DoF robotic hand.

5.4.3 Manipulability Tracking for a Humanoid Center of Mass

An interesting use of manipulability ellipsoids arises when these are defined at the center of mass (CoM) of humanoid robots, which permits to analyze their capabilities to accelerate the CoM in locomotion [Azad et al., 2017; Gu et al., 2015], or to evaluate how resistant they can be to external perturbations using the force manipulability at



(a) Initial pose (b) Final pose (c) Manipulability evolution

Figure 5.14 – Manipulability tracking for grasping tasks with the Allegro hand in simulation. (a) and (b) show the initial and final pose of the robot, respectively. (c) The initial, final and desired manipulability ellipsoids are depicted in yellow, dark purple and green, respectively. The *bottom-right* graph shows the evolution of the distance between the current and desired manipulability ellipsoid over time (in seconds).

a specific humanoid posture. With the goal of getting some insights on the role of CoM manipulability ellipsoids in legged robots, we designed manipulability tracking experiments using two different floating-base robots in simulation, namely, the humanoid NAO and the Centauro robot [Baccelliere et al., 2017].

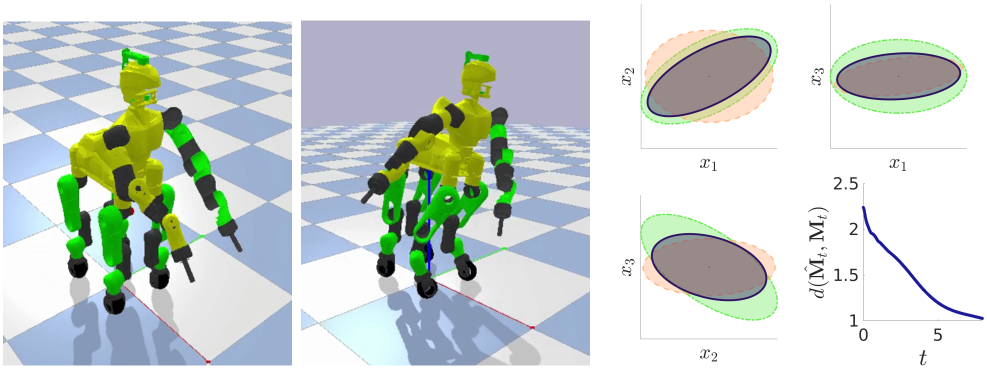
Specifically, we required the robots to track a desired manipulability ellipsoid defined at its CoM while keeping balance. We assumed a strict hierarchy of tasks that gave the highest priority to the task of maintaining the CoM position over the support polygon and zero velocity at all contact points with the floor, while the manipulability tracking was considered a secondary task. Under the aforementioned assumptions, we implemented the inverse kinematics-based controller for floating-base robots proposed in [Mistry et al., 2008], which we briefly introduce here. First, let us define the Jacobian for the primary task as

$$\boldsymbol{J}_b = \begin{bmatrix} \boldsymbol{J}_{\text{feet}} \\ \boldsymbol{J}_{\text{CoM,xy}} \end{bmatrix}, \tag{5.35}$$

where $\boldsymbol{J}_{\text{feet}}$ represents the Jacobians for the position/orientation of the robot feet while $\boldsymbol{J}_{\text{CoM,xy}}$ is the Jacobian for the projection of the CoM onto the (x, y) plane (assuming the gravity vector is in the z direction). Next, we define the vector of primary desired velocities $\boldsymbol{x}_b$ (i.e. velocities of the robot feet and CoM), noting that all the robot feet velocities must equal zero in order to maintain constraints, therefore

$$\dot{\boldsymbol{x}}_b = \begin{bmatrix} \boldsymbol{0} \\ \dot{\boldsymbol{x}}_{\text{CoM}} \end{bmatrix}, \tag{5.36}$$

where $\dot{\boldsymbol{x}}_{\text{CoM}}$ is the velocity at the robot CoM so that it lies in the support polygon.



(a) Initial pose (b) Final pose (c) Manipulability evolution

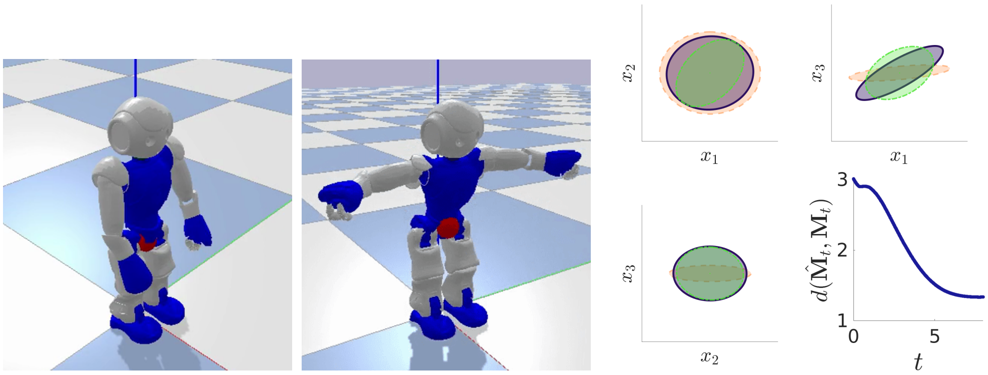
Figure 5.15 – Tracking of the COM manipulability with the Centauro robot in simulation. (a) and (b) show the initial and final pose of the robot, respectively. (c) The initial, final and desired manipulability ellipsoids are depicted in yellow, dark purple and green, respectively. The *bottom-right* graph shows the evolution of the distance between the current and desired manipulability ellipsoid over time, given in seconds.

Regarding the secondary task, that is, the manipulability tracking at the robot CoM, we first compute the Jacobian at the CoM $\boldsymbol{J}_{\text{CoM}}$ for floating-base robots as in [Mistry et al., 2008], which allows us to calculate manipulability ellipsoids of the types introduced in Section 5.2. Depending on which type of manipulability we require the robot to track, we can use any of the manipulability Jacobians (5.3), (5.4) or (5.5) to compute the desired joint velocities $\dot{\boldsymbol{q}}$ for the manipulability tracking task using (5.7). So, the full joint velocity controller for legged robots required to keep balance while tracking a desired manipulability ellipsoid at their CoM is defined as

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \boldsymbol{0}_{6\times n} \end{bmatrix}^{\mathsf{T}} \left(\boldsymbol{J}_b^{\dagger} \dot{\boldsymbol{x}}_b + \boldsymbol{N}_b \, (\boldsymbol{\mathcal{J}}_{(3)}^{\dagger})^{\mathsf{T}} \, \boldsymbol{K}_{\boldsymbol{M}} \, \mathrm{mvec}\Big(\mathrm{Log}_{\boldsymbol{M}_t}(\hat{\boldsymbol{M}}_t) \Big) \right), \tag{5.37}$$

where the first term is included in order to account for the virtual joints of legged robots, n is the number of DoF of the robot, and $\boldsymbol{N}_b$ is the nullspace of the Jacobian (5.35).

We ran several experiments for testing the manipulability tracking at the CoM of the Centauro (Figure 5.15) and NAO (Figure 5.16) robots using the controller (5.37). The tests consisted of manually setting a desired manipulability ellipsoid to be tracked at the CoM of the robot, and running a joint velocity controller given the reference provided by (5.37). Notably, both Centauro and NAO tracked the desired manipulability as precisely as possible without compromising the balancing task. Figures 5.15c and 5.16c show the distance between the desired and current CoM manipulability, which decreases over time as the robot adapts its posture to carry out a good tracking while keeping its balance. An interesting aspect about defining and tracking CoM manipulability ellipsoids is the final posture that the robots achieve. Figure 5.15b shows the final posture achieved



(a) Initial pose (b) Final pose (c) Manipulability evolution

Figure 5.16 – Tracking of the COM manipulability with NAO in simulation. (a) and (b) show the initial and final pose of NAO, respectively. The CoM of the robot is depicted by a red sphere. (c) The initial, final and desired manipulability ellipsoids are depicted in yellow, dark purple and green, respectively. The *bottom-right* graph shows the distance between the current and desired manipulability over time (given in seconds).

by Centauro when tracking a CoM manipulability whose projection on the (x_1, x_2) plane is a tilted ellipse, which makes the robot adopt a posture where the front legs and torso rotate on the same plane (which corresponds to the floor in the virtual environment). The final posture of NAO displayed in Figure 5.16b shows that both arms are completely extended along the humanoid frontal axis, in an attempt to align them with one of the main axis of the CoM manipulability ellipsoid. However, both the balancing task and the lower number of DoF constrain NAO to closely match the desired manipulability.

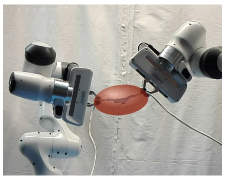
5.5 Conclusion

This chapter presented a novel approach to track robot manipulability ellipsoids. Our work extends the classical inverse kinematics problem to manipulability ellipsoids, by establishing a mapping between a change of manipulability ellipsoid and the robot joint velocity. We exploited tensor representation and Riemannian manifolds to build geometry-aware manipulability tracking controllers and showed the importance of geometry-awareness for manipulability tracking.

Our tracking formulation enables robots to modify their posture in an exponentially stable way so that desired manipulability ellipsoids are tracked, either as a main control task or as a redundancy resolution problem where the manipulability tracking is considered a secondary objective. Compared to state-of-the-art manipulability-based optimization schemes, our tracking formulation allows the reproduction of any manipulability ellipsoid beyond the maximization of manipulability parameters. The proposed tracking approach covers different manipulability ellipsoids proposed in the literature, such as velocity, force and dynamic manipulability ellipsoids [Doty et al., 1995]. A relevant aspect about our approach is its generic structure, which means that we can track manipulability ellipsoids for a large variety of robots, as reported in the previous section, where a robotic arm, a robotic hand, a Centauro robot and a humanoid were used to test our tracking approach. This shows that our approach can be used in a large variety of contexts and that many further applications can be considered.

We also showed that our manipulability tracking framework is compatible with statistical methods providing 4th-order covariances, allowing us to exploit task variations to characterize the precision of the manipulability tracking problem, with stronger tracking along low variability directions. Therefore, our manipulability tracking framework may be readily combined with the manipulability learning framework presented in Chapter 4. A complete *manipulability transfer* framework will be introduced in the next chapter, where a human or robot *teacher* demonstrates how to perform a task with a desired time-varying manipulability profile, while a *student* robot reproduces the task by exploiting its own redundant kinematic structure so that its manipulability ellipsoid matches the demonstration.

6 A Human-to-robot Manipulability Transfer Framework



In the Part I of this thesis, we introduced a manipulability learning framework, allowing the encoding and retrieval of sequences of manipulability ellipsoids from demonstrations. In order to be able to reproduce a learned manipulability profile, the previous chapter proposed to endow robots with adapted geometry-aware controllers.

In this chapter, we propose to combine the aforementioned manipulability learning and tracking approaches into a complete geometry-aware *manipulability transfer* framework, allowing robots to learn and reproduce manipulability ellipsoids from expert demonstrations.

Publication note

The material presented in this chapter is adapted from the following publications:

- Jaquier, N., Rozo, L., Caldwell, D. G., and Calinon, S. (2020b). Geometry-aware manipulability learning, tracking and transfer. *Intl. Journal of Robotics Research*;
- Jaquier, N., Rozo, L., and Calinon, S. (2020c). Analysis and transfer of human movement manipulability in industry-like activities. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

Supplementary material

Videos related to this chapter are available at:

https://sites.google.com/view/manipulability.

Source code

Source codes related to this chapter are available at:

https://github.com/NoemieJaquier/Manipulability.

6.1 Introduction

In the two previous chapters, we presented a manipulability learning and a manipulability tracking frameworks. In Chapter 4, the learning approach provided a sequence of desired manipulability ellipsoids learned from demonstrations. After the learning phase, the goal of a learner robot is to reproduce the given profile of desired manipulability ellipsoids. However, existing approaches built on manipulability-based optimizations are not suitable as they do not allow the tracking of specific manipulability ellipsoids. The tracking framework presented in Chapter 5 used manually-specified robot manipulability ellipsoids for the task. However, this may be tedious and cumbersome when the robot needs to carry out different and complex tasks. In this chapter, we show that integrating the proposed learning and tracking approaches solves the aforementioned problems and offers a complete geometry-aware manipulability transfer framework where manipulability ellipsoid profiles are learned from demonstrations and reproduced accurately.

To do so, we introduce the novel idea that manipulability-based posture variation for task compatibility can be addressed from a robot learning from demonstration perspective. Specifically, we cast this problem as a *manipulability transfer* between a teacher and a learner. The former demonstrates how to perform a task with a desired time-varying manipulability profile, while the latter reproduces the task by exploiting its own redundant kinematic structure so that its manipulability ellipsoid matches the demonstration. Unlike classical learning frameworks that encode reference position, velocity and force trajectories, our approach offers the possibility of transferring posture-dependent task requirements such as preferred directions for motion and force exertion in operational space, which are encapsulated in the demonstrated manipulability ellipsoids.

This idea opens two main challenges, namely, *(i)* how to encode and retrieve a sequence of manipulability ellipsoids, and *(ii)* how to track a desired time-varying manipulability either as the main task of the robot or as a secondary objective. The former problem is addressed by the learning framework proposed in Chapter 4, while the latter is solved through the manipulability tracking formulation introduced in Chapter 5.

The proposed approach can be straightforwardly applied to different types of kineto-static and dynamic manipulability measures. This opens the door to manipulability transfer scenarios with various types of robots where different task requirements at kinematic and dynamic levels can be learned and successfully transfered between agents of different embodiments. In particular, this framework also permits to transfer other velocity, force or impedance specifications with any priority order with respect to the manipulability tracking controller. After describing the proposed manipulability transfer framework (see Section 6.2), the functionality of the proposed approach is evaluated in different manipulability transfer scenarios involving a simulated legged robot and two dual-arm robots (see Section 6.3). We first show that the task-dependent patterns observed by analyzing single- and dual-arm manipulability in human movements during screwing

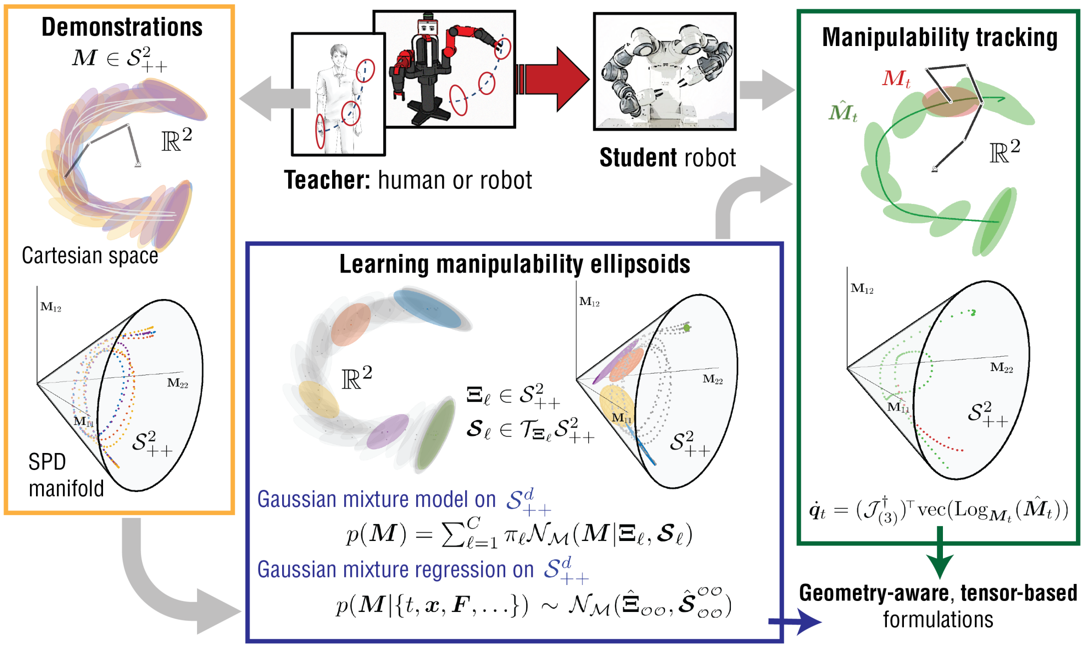

Demonstrations
$M \in \mathcal{S}^2_{++}$
$\mathbb{R}^2$
Cartesian space
$\mathbf{M}_{12}$
$\mathbf{M}_{22}$
$\mathbf{M}_{11}$
$\mathcal{S}^2_{++}$
SPD manifold
Teacher: human or robot
Student robot
Manipulability tracking
M_t
$\hat{M}_t$
$\mathbb{R}^2$
$\mathbf{M}_{12}$
$\mathbf{M}_{22}$
$\mathbf{M}_{11}$
$\mathcal{S}^2_{++}$
$\dot{q}_t = (\mathcal{J}^{\dagger}_{(3)})^{\top} \mathrm{vec}(\mathrm{Log}_{M_t}(\hat{M}_t))$
Learning manipulability ellipsoids
$\mathbb{R}^2$
$\Xi_\ell \in \mathcal{S}^2_{++}$
$\boldsymbol{\mathcal{S}}_\ell \in \mathcal{T}_{\Xi_\ell}\mathcal{S}^2_{++}$
$\mathbf{M}_{12}$
$\mathbf{M}_{22}$
$\mathbf{M}_{11}$
$\mathcal{S}^2_{++}$
Gaussian mixture model on $\mathcal{S}^d_{++}$
$p(M) = \sum_{\ell=1}^{C} \pi_\ell \mathcal{N}_{\mathcal{M}}(M|\Xi_\ell, \boldsymbol{\mathcal{S}}_\ell)$
Gaussian mixture regression on $\mathcal{S}^d_{++}$
$p(M|\{t, x, F, \ldots\}) \sim \mathcal{N}_{\mathcal{M}}(\hat{\Xi}_{\mathcal{OO}}, \hat{\boldsymbol{\mathcal{S}}}^{\mathcal{OO}}_{\mathcal{OO}})$
Geometry-aware, tensor-based formulations

Figure 6.1 – Overview of the proposed manipulability transfer framework

and carrying tasks in Chapter 4 can be transferred to robots executing similar tasks, bypassing the complexity of kinematic mapping approaches. Then, the manipulability transfer is showcased in a bimanual setup where an unplugging task is kinesthetically demonstrated to a 14-DoF dual-arm robot, which then transfers the learned model to a different dual-arm system that reproduces the unplugging task successfully.

6.2 Manipulability Transfer

An overview of the proposed manipulability transfer framework is presented by Figure 6.1. In the first step, trajectories demonstrated by a human or robot *teacher* are collected, including the manipulability profile corresponding to the task. Note that the trajectories can jointly encode various variables, e.g., time, position, velocity, force measurements. In the learning phase, a Gaussian mixture model (GMM) is built on the SPD manifold to encode the trajectories and a sequence of desired manipulability ellipsoids is retrieved using Gaussian mixture regression (GMR) on the SPD manifold. The learning approach was detailed in Chapters 3 and 4 of this thesis. In the reproduction phase, a *student* robot tracks the given desired manipulability profile by using one of the controllers of the manipulability tracking framework detailed in Chapter 5. Note that these controllers can straightforwardly be combined with classical control approaches, e.g., position, velocity, force or impedance controllers.

6.3 Experiments

6.3.1 Human-to-robot Manipulability Transfer

In Chapter 4, we presented a detailed analysis of single and dual-arm manipulability ellipsoids for human movements during industry-like activities. Here, we illustrate how this manipulability analysis can be exploited to transfer manipulability-based posture variation from humans to robots executing similar tasks without the need of complex kinematic mappings. Namely, we propose to transfer the manipulability requirements of screwing and carrying tasks (SM and C5) from a human to a Centauro robot [Baccelliere et al., 2017]. To do so, we exploit the proposed manipulability transfer framework for the Centauro robot to learn and reproduce manipulability ellipsoids from human demonstrations. For both tasks, the demonstrations consist of the 15 recorded trials of the participant 541 of the dataset [Maurice et al., 2019] previously used for the manipulability analysis in Chapter 4. A subset of actions is considered for each task. The simulated experiments were performed using Pyrobolearn [Delhaisse et al., 2019].

In Chapter 4, we showed that a manipulability ellipsoid profile of the transition actions, such as the carrying action Ca for the screwing task, can be seen as part of a planning process for the next action of interest (i.e. screwing Sc). We exploit this idea for the manipulability transfer of the screwing task. We propose to learn and reproduce the manipulability profile of the actions preceding Sc, namely carrying (Ca), placing (Pl) and fine manipulation (Fm), aiming at reaching an appropriate posture to efficiently execute the main task. To do so, a desired time-driven manipulability profile $\hat{\boldsymbol{M}}_t$ is first learned from the demonstrations with a Gaussian mixture model (GMM) on the SPD manifold. The robot is then required to track the desired manipulability profile while keeping balance and positioning its end-effector at a specific location, whose height is equal to the one of the SM motion. We assume a task hierarchy that prioritizes the position control of the center of mass (CoM) over the support polygon and zero velocity at all contact points with the floor, while both the end-effector position and manipulability tracking are considered secondary tasks. The corresponding full joint velocity controller for legged robots is defined similarly as (5.37) (see [Mistry et al., 2008] for details), i.e.,

$$\dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{I}_{n\times n} \\ \boldsymbol{0}_{6\times n} \end{bmatrix}^{\mathsf{T}} \left(\boldsymbol{J}_b^{\dagger} \dot{\boldsymbol{x}}_b + \boldsymbol{N}_b \dot{\boldsymbol{q}}_{\boldsymbol{N}_b} \right), \tag{6.1}$$

where the first term accounts for the virtual joints of a floating-base robot, n is the number of DoF of the robot, $\dot{\boldsymbol{x}}_b$ and $\boldsymbol{J}_b$ are defined in (5.36) and (5.35), $\boldsymbol{N}_b$ is the corresponding nullspace projection matrix and $\dot{\boldsymbol{q}}_{\boldsymbol{N}_b}$ is the joint velocities of the secondary tasks. In the first part of the motion, corresponding to the carrying action Ca, the manipulability tracking is prioritized over the position tracking in order to allow the robot to reach a good initial posture for executing the task. Therefore, we defined the joint velocities of the secondary task as

$$\dot{\boldsymbol{q}}_{\boldsymbol{N}_b} = \underbrace{(\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}}\, \boldsymbol{K}_{\boldsymbol{M}}\, \mathrm{mvec}\Big(\mathrm{Log}_{\boldsymbol{M}}(\hat{\boldsymbol{M}})\Big)}_{\text{main task}} + \underbrace{\Big(\boldsymbol{I} - (\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}} \boldsymbol{\mathcal{J}}^{\mathsf{T}}_{(3)}\Big)}_{\text{nullspace op.}} \underbrace{\boldsymbol{J}^{\dagger}\, \boldsymbol{K}_{\boldsymbol{x}}\, (\hat{\boldsymbol{x}} - \boldsymbol{x})}_{\text{secondary task}}, \tag{6.2}$$

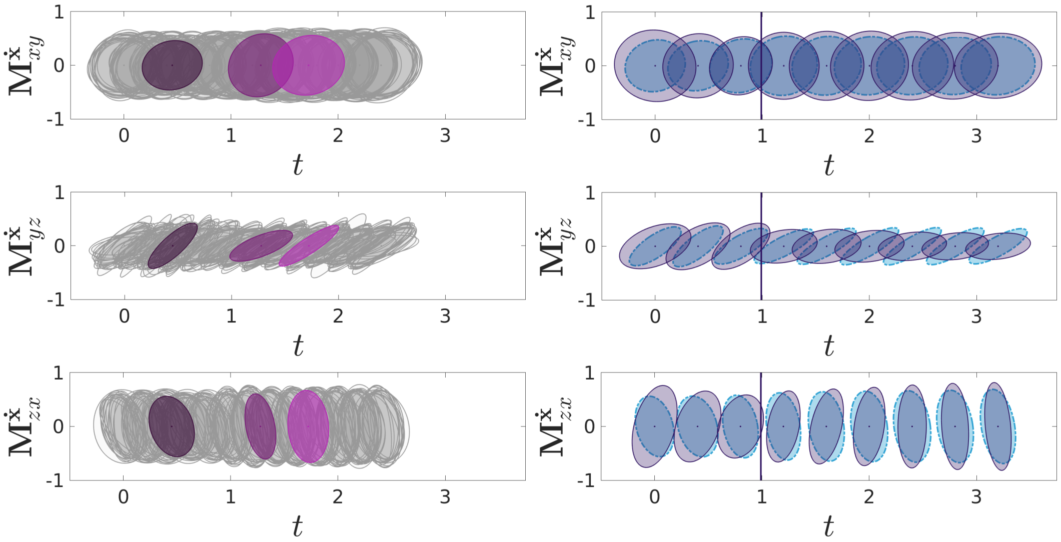
where the position tracking controller is projected onto the nullspace of the manipulability controller, see also 5.25. Then, for the remaining part of the motion, the position tracking is prioritized so that the robot can reach the screwing location with a posture adapted to the task requirements. Thus, the corresponding joint velocities are given by

$$\dot{\boldsymbol{q}}_{\boldsymbol{N}_b} = \underbrace{\boldsymbol{J}^{\dagger}\, \boldsymbol{K}_{\boldsymbol{x}}\, (\hat{\boldsymbol{x}} - \boldsymbol{x})}_{\text{main task}} + \underbrace{(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})}_{\text{nullspace op.}}\, \underbrace{(\boldsymbol{\mathcal{J}}^{\dagger}_{(3)})^{\mathsf{T}}\, \boldsymbol{K}_{\boldsymbol{M}}\, \mathrm{mvec}\Big(\mathrm{Log}_{\boldsymbol{M}}(\hat{\boldsymbol{M}})\Big)}_{\text{secondary task}}, \tag{6.3}$$

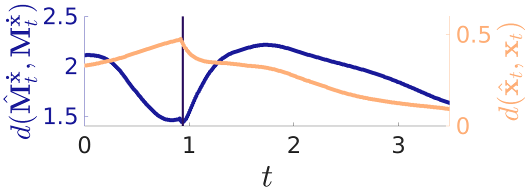
based on the manipulability-based resolution controller (5.8).

Figure 6.2a displays the demonstrations along with the centers of the components of the geometry-aware GMM encoding $\boldsymbol{M}_t^{\hat{\boldsymbol{x}}}$. The number of components of the model ($C = 3$) was selected by the experimenter. Figures 6.2b and 6.3 show the evolution of the posture and the manipulability of Centauro during the reproduction of the pre-screwing motion with its right arm. The manipulability tracking has priority over the position control by (6.2) from $t = 0$ to $t = 1$s, and the priority order is reversed for the remaining time using (6.3). Therefore, during the first part of the motion, we can observe that the robot mainly adapts its posture to fulfill the manipulability requirements. This naturally results in the robot orienting its end-effector outwards w.r.t its torso. In this phase, the position error slightly increases, while the manipulability error decreases (see Figure 6.2c). The robot starts the second part of the task by moving its right arm to decrease the position error. Interestingly, despite the desired position could be reached solely by extending the arm, the robot instead uses its torso to rotate the arm in order to reach the desired end-effector position while still tracking the desired manipulability accurately (see Figure 6.3c). Note that tracking the desired manipulability for this task naturally favors arm postures where the end-effector is oriented outwards, matching a screwing motion whose main direction is orthogonal to a vertical plane in the robot workspace. This corresponds to the recorded screwing motion of the participant 541. However, considering a precise orientation remains necessary for successfully executing a complete screwing task, which can be straightforwardly included in our control formulation. Finally, notice that we manually encoded the transition between the controllers (6.2) and (6.3) in this experiment. A data-driven automatic prioritization of the manipulability and position requirements may also be implemented following the approach presented in [Silvério et al., 2019].

For the transfer of the `C5` task, we consider the part of the motion where the human picks up the load to carry it, i.e. `Pi` and `Ca` actions. After having learned the desired dual-arm

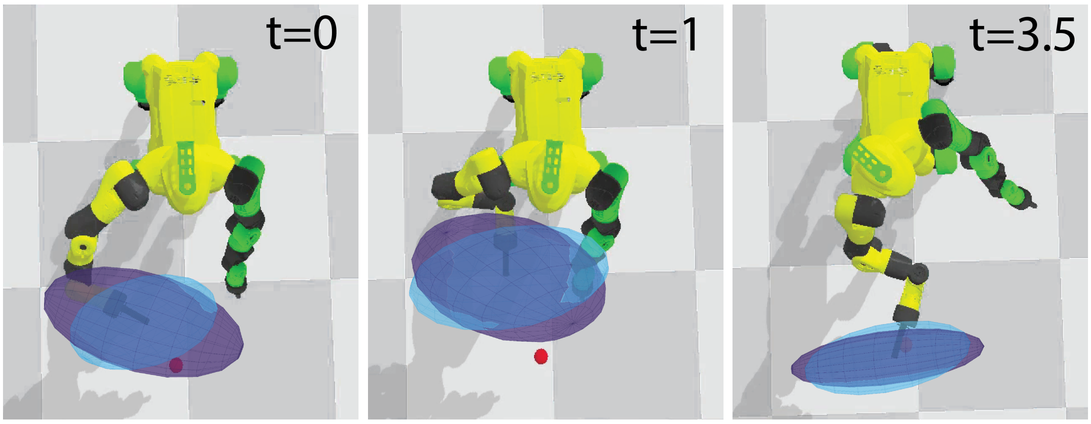


(a) Demonstrations and learned GMM (b) Desired and reproduced trajectories



(c) Distances evolution

Figure 6.2 – Partial reproduction of a screwing task using human-robot manipulability transfer. (*a*) shows the demonstrated manipulability profile (in gray) and the centers of the 3-states GMM on the SPD manifold (in purple) over time. (*b*) shows the 2D projections of the time-varying desired manipulability profile, learned by demonstrations, and the reproduced manipulability depicted by blue and dark purple ellipsoids, respectively. (*c*) depicts the distance between the current and desired manipulability (and end-effector position) over time (given in seconds). The change of controller during the pre-screwing motion is indicated by a vertical line.



(a) Initial posture (b) Switch of tracking priority (c) Final posture

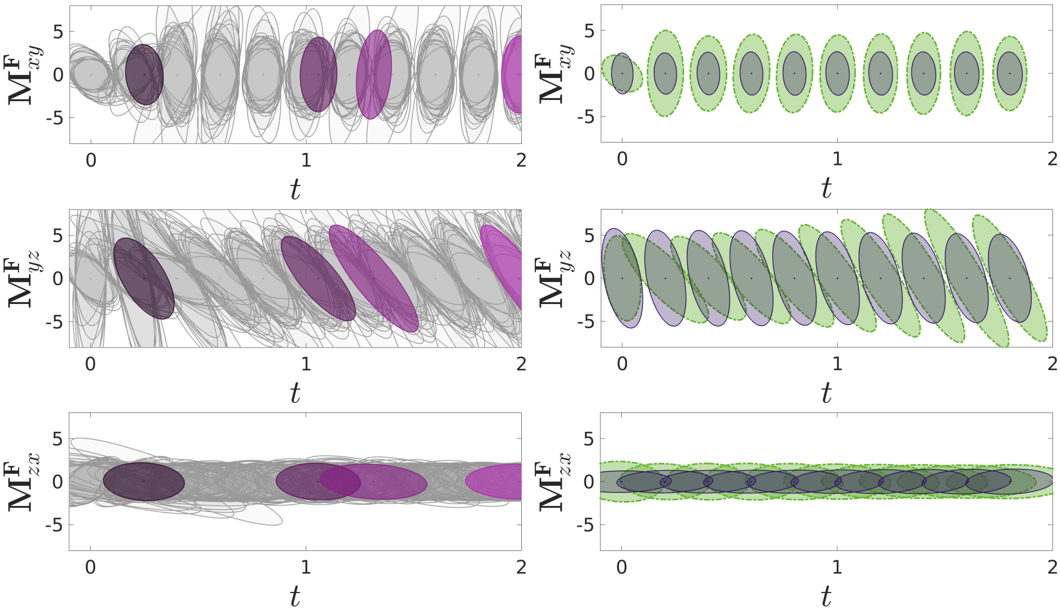
Figure 6.3 – Snapshots of the robot executing a partial reproduction of a screwing task using human-robot manipulability transfer. The snapshots correspond to (a) the initial posture, (b) the posture when the tracking priority switches and (c) the final posture of the robot. The desired and reproduced manipulability are depicted by blue and purple ellipsoids, respectively. The desired end-effector position is depicted by a red dot. The time is given in seconds.

manipulability profile from demonstrations, we employ the main controller (6.1) with the nullspace controller (6.3) for the whole task while keeping the end-effectors position fixed. Note that we consider all the DoFs of the robot for computing the dual-arm manipulability ellipsoid $\boldsymbol{M}_d^{\boldsymbol{F}} = \left(\boldsymbol{G}_d^{\dagger\top}\boldsymbol{J}_d\boldsymbol{J}_d^{\top}\boldsymbol{G}_d^{\dagger}\right)^{-1}$ (4.3), so that the dual-arm Jacobian is defined as $\boldsymbol{J}_d = (\boldsymbol{J}_l^{\top}, \boldsymbol{J}_r^{\top})^{\top}$. Figure 6.4a displays the demonstrations along with the centers of the components of the geometry-aware GMM encoding $\boldsymbol{M}_{d,t}^{\boldsymbol{F}}$. The number of components of the model ($C = 4$) was selected by the experimenter. Figures 6.3 and 6.4b show the evolution of the posture and the manipulability of Centauro during the reproduction of the dual-arm carrying motion. We observe that the robot adapts the posture of its arms to fulfill the manipulability requirements of the task. However, constraining the end-effectors to fixed positions significantly reduces the DoF redundancy that the robot can exploit to track accurately the desired manipulability (see also Figure 6.4c). This issue may be alleviated by allowing the robot to vary the positions of its end-effector while maintaining a constant distance between them after picking up the load.

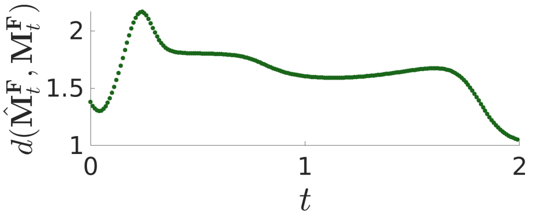
6.3.2 Manipulability Transfer Between Robots for a Bimanual Task

The performance of the proposed manipulability transfer framework was also tested in a bimanual unplugging of an electric cable from a power socket. The central idea is to teach different dual-arm robots to execute a task requiring a specific manipulability profile via kinesthetic teaching provided only to one of the bimanual robots.

In the first part of the experiment, the two 7-DoF arms of a Baxter robot are kinesthetically



(a) Demonstrations and learned GMM (b) Desired and reproduced trajectories



(c) Distance evolution

Figure 6.4 – Partial reproduction of a carrying task using human-robot manipulability transfer. (*a*) shows the demonstrated manipulability profile (in gray) and the centers of the 4-states GMM on the SPD manifold (in purple) over time. (*b*) shows the 2D projections of the time-varying desired manipulability profile, learned by demonstrations, and the reproduced manipulability depicted by green and dark purple ellipsoids, respectively. (*c*) depicts the distance between the current and desired manipulability (and end-effector position) over time (given in seconds).

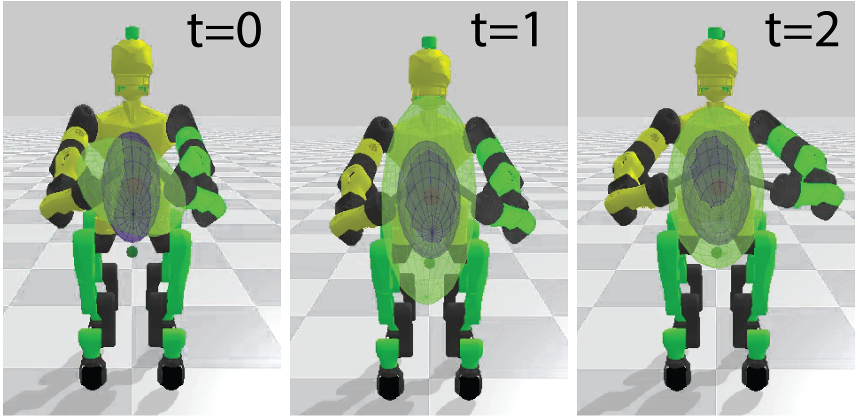
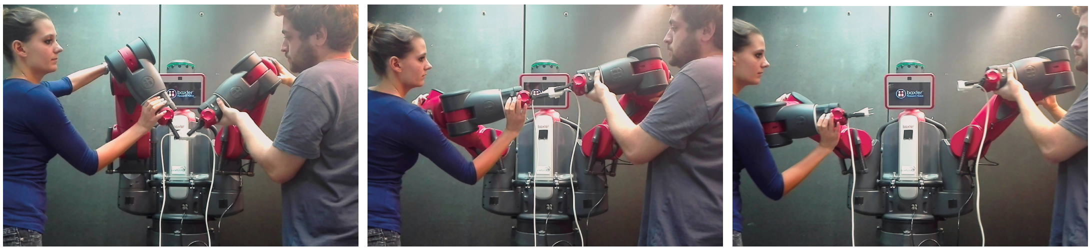


Figure 6.5 – Snapshots of the robot executing a partial reproduction of a carrying task using human-robot manipulability transfer. The snapshots correspond to the initial, intermediate and final postures of the robot. The desired and reproduced manipulability are depicted by green and purple ellipsoids, respectively. The time is given in seconds.

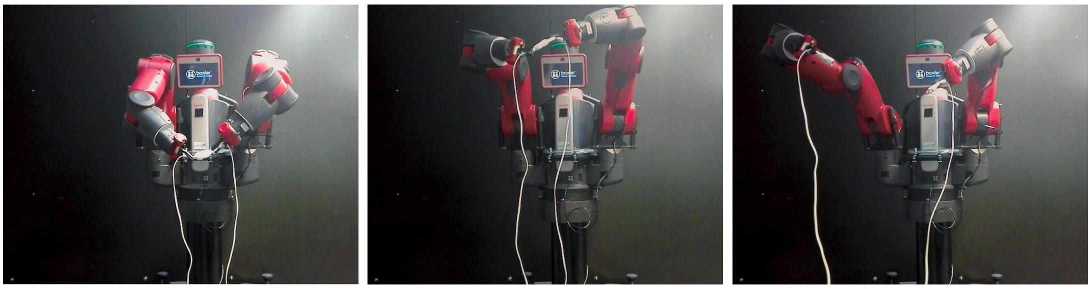
guided to provide demonstrations (see Figure 6.6a). The posture of the arms is modified by the user so that the main axis of the dual force manipulability ellipsoid of the system $\boldsymbol{M}_d^{\boldsymbol{F}} = (\boldsymbol{G}_d^{\dagger\top}\boldsymbol{J}_d\boldsymbol{J}_d^{\top}\boldsymbol{G}_d^{\dagger})^{-1}$ is aligned with the direction of extraction. Then, the arms are moved in opposite directions to unplug the electric cable from the socket. We extracted both the relative position $\boldsymbol{\Delta x}_t$ between the end-effectors of both arms and the force manipulability ellipsoid of the system $\boldsymbol{M}_{d,t}^{\boldsymbol{F}}$. The collected data were time-aligned and split in two datasets of time-driven trajectories, namely relative Cartesian positions and manipulability. We trained a classical GMM over the time-driven relative positions and a geometry-aware GMM over the time-driven manipulability ellipsoids. The number of components of each model ($C = 4$) was selected by the experimenter.

In the second part of the experiment, the unplugging task is reproduced by both the Baxter robot and a pair of Franka Emika Panda robots (see Figures 6.6b, 6.6c). For both reproductions, the relative position between the end-effectors and the desired manipulability of the system were computed at each time step by a classical GMR as $\widehat{\boldsymbol{\Delta x}}_t \sim p(\boldsymbol{\Delta x}|t)$ and a geometry-aware GMR as $\hat{\boldsymbol{M}}_{a,t}^{\boldsymbol{F}} \sim P(\boldsymbol{M}_d^{\boldsymbol{F}}|t)$. In both cases, the left robotic arm was required to move its joints to track the desired manipulability ellipsoid (5.7), while the right arm was required to maintain the desired relative Cartesian position with respect to the left arm, while tracking the desired manipulability as secondary objective (5.8). Note that the actuation contribution of each robot was taken into account to compute the manipulability ellipsoids through the whole experiment.

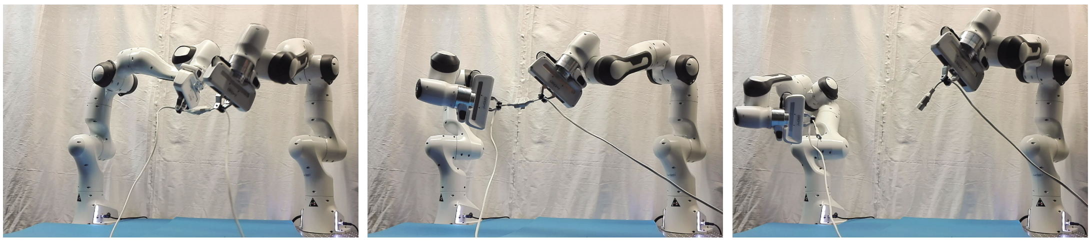
Figure 6.7 displays the two demonstrations recorded by kinesthetically guiding the Baxter robot along with the components of the GMM encoding $\boldsymbol{\Delta x}_t$ and the centers of the components of the geometry-aware GMM encoding $\boldsymbol{M}_d^{\boldsymbol{F}}$. The first and third dimensions of $\boldsymbol{\Delta x}_t$ are not represented as they do not vary significantly during the experiment.



(a) Demonstrations provided by the user on the Baxter robot

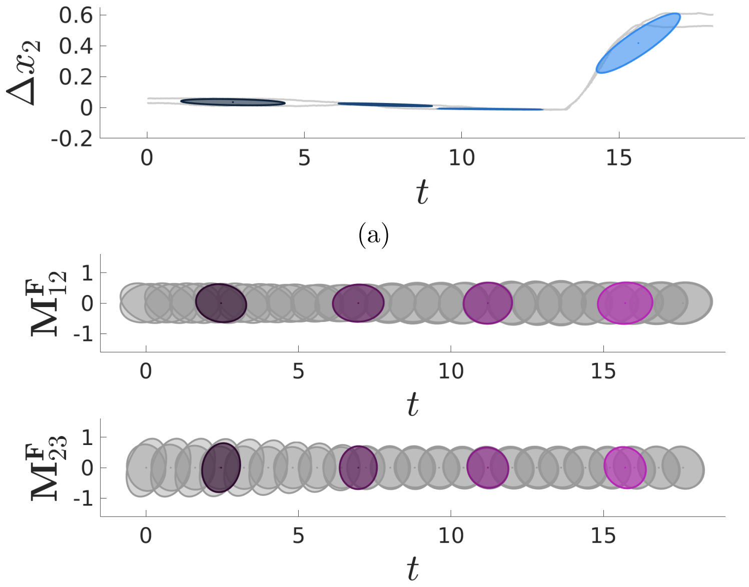


(b) Reproduction by the Baxter robot



(c) Reproduction by the two Franka Emika Panda robots

Figure 6.6 – Snapshots of the bimanial unplugging task. The robots pose at the beginning of the task, before and after the extraction of the cable from the socket are respectively shown in the *left*, *middle* and *right* column.



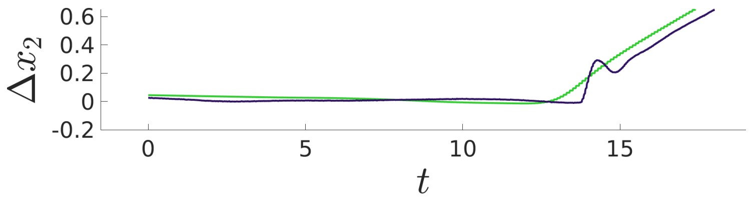
(a)

(b)

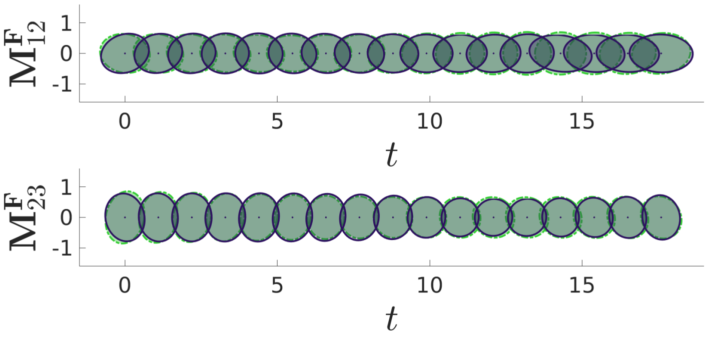
Figure 6.7 – Demonstrations and GMM encoding the bimanual unplugging task. (*a*) Demonstrated relative end-effectors position for the Baxter robot (in gray) and components of the 4-states GMM (in blue). Only the most representative dimension is displayed. The distance between the two arms increases when the cable is unplugged from the socket. (*b*) Demonstrated force manipulability profile (in gray) and centers of the 4-states GMM in the SPD manifold over time (in purple). The position $\boldsymbol{x}$ and time t are given in meters and seconds, respectively.

Figure 6.8 shows the relative Cartesian position and manipulability ellipsoid profile to be tracked and the reproduction results when the Baxter robot executed the task. Baxter successfully tracked the desired manipulability ellipsoid while maintaining the required relative distance between its end-effectors.

Figure 6.9 shows the relative Cartesian position between the arms and the manipulability ellipsoid profile obtained during the reproduction of the task by the two Panda robots. These successfully achieved the required task and tracked the desired manipulability ellipsoid profile obtained from model trained with the data recorded on the Baxter robot. Note the manipulability matching is not exact in this case due to the differences between Baxter and the Panda robots. Indeed, even if the actuation capabilities of each robot are taken into account in our manipulability transfer framework, the capabilities of the two dual-arm system differ due to other physical specificities, e.g. the relative position of the bases of the arms.

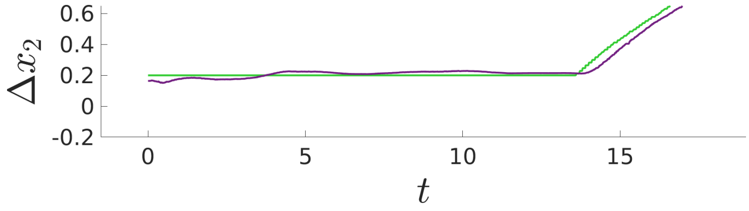


(a)

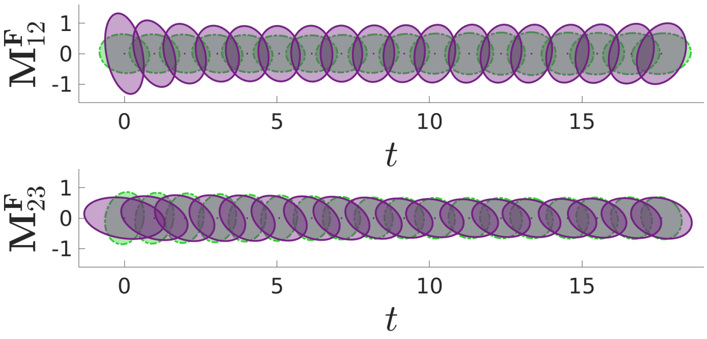


(b)

Figure 6.8 – Reproduction of the bimanual unplugging task with Baxter. The desired and reproduced trajectories are represented in green and dark blue respectively. (a) shows the desired and reproduced relative position between the end-effectors along the second dimension. (b) shows the desired and reproduced (overlapping) manipulability ellipsoids.



(a)



(b)

Figure 6.9 – Reproduction of the bimanual unplugging task with the two Panda robots. The desired and reproduced trajectories are represented in green and purple respectively. (a) shows the desired and reproduced relative position between the end-effectors along the second dimension. (b) shows the desired and reproduced manipulability ellipsoids.

6.4 Discussion

The manipulability transfer results reported in Section 6.3 showed the effectiveness of the proposed approach for transferring manipulability ellipsoids from humans to robots or between robots that differ in their kinematic structure, which has remained a challenge in the robot learning community. Our learning framework allows a robot to learn posture-dependent task requirements without explicitly encoding a model in the joint space of the demonstrator, which would require complex kinematic mapping algorithms and would make task analysis less interpretable at first sight. In addition, the proposed framework extends the robot learning capabilities beyond the transfer of position trajectories, force profiles and impedance parameters.

It is important to emphasize the fact that the manipulability tracking precision strongly depends on the kinematic redundancy (i.e., number of DoFs of the robot) when the task is considered a secondary objective, as the higher it is, the more capable the robot is to perform more than one task simultaneously. Note that, in the case of legged robots (which are often characterized by a high number of DoFs), the manipulability tracking may still be slightly compromised because of the set of constraints imposed by the balancing task, as observed in Sections 5.4.3 and 6.3.1. However, if these robots are provided with the possibility of modifying their feet position while keeping balance, then the manipulability tracking may be further improved. This clearly requires more sophisticated balancing controllers, but gives robots more freedom to adapt their posture and achieve better manipulability tracking. Notice that in the case of robotic hands, a similar behavior arises when the finger tips are constrained according to some grasping requirements, which might affect the manipulability tracking when projected into the nullspace of the primary task. Finally, a better manipulability tracking can be achieved when the corresponding controller is given a high or the highest priority. This may be viewed as part of a planning process, as shown in the human-to-robot manipulability transfer experiments.

It is important to notice that the proposed manipulability tracking approach is a local method in the sense that the solution depends on the current configuration of the robot which is a function of the Jacobian. This makes the tracking convergence dependent on the current configuration of the robot, which sometimes may limit the tracking performance. However, the robot may achieve a better tracking if it is allowed to look for other initial postures. As an example, the robot may not track precisely the desired manipulability ellipsoids for a given initial posture, due for instance to its joint limits. However, if the robot slightly modifies its initial posture, it may find a better starting configuration to subsequently minimize the error between the desired and current manipulability ellipsoids in a larger proportion, even if the new initial posture initially increases this error.

As stated previously, the affine-invariant metric set matrices with null eigenvalues at an infinite distance of any SPD matrix. Consequently, special care must be taken when the robot is at a singular configuration as the corresponding manipulability ellipsoid significantly shrinks along the dimensions with null eigenvalues. To overcome this problem, [Yun, 2020] proposed to use a distance metric on the Riemannian manifold of positive semi-definite (PSD) matrices of rank 1, as defined in [Bonnabel and Sepulchre, 2010], to match a desired major axis of the manipulability ellipsoid. However, it is not clear how this approach would be extended to track of a full desired manipulability ellipsoid when the robot moves away from the singularity. In practice, singular configurations can be easily avoided by defining appropriate desired manipulability ellipsoids, whose eigenvalues are far enough from zero. Another alternative would be to use the Bures metric [Bures, 1969], which is a distance on the space of SPD matrices that is robust to singular matrices.

Overall, the proposed manipulability transfer framework may be exploited in a large variety of applications, where the posture of the robot may have an impact on its performance while executing the task. In addition to varying the robot posture for task compatibility, tracking a desired manipulability profile as a secondary task may typically complement a main control task to avoid singularity, handle perturbations during task execution, optimize the execution time or minimize the energy consumption [Kim et al., 2010]. In particular, manipulability transfer may be utilized from a motion planning point of view. To do so, the robot may first track a desired manipulability as main control task in a planning phase, where the robot adapts its posture in order to anticipate the next action. Following this planning phase, the robot executes the desired action with a posture adapted to the task requirements. In this phase, the desired manipulability is tracked as a secondary task. In a related context, the manipulability learning and tracking frameworks may be exploited to plan or adapt the configuration of the robot in anticipation of potential collisions or to minimize the potentially damaging impact forces if the collision cannot be avoided. To do so, a profile of dynamic impact manipulability ellipsoids, which is dually related to the dynamic manipulability [Walker, 1994], may be learned in function of the potential collisions and tracked as a secondary task. Moreover, in the context of rehabilitation and assistance, the proposed learning and tracking formulations may be exploited in control strategies for exoskeletons. In [Petrič et al., 2019], the exoskeleton posture is optimized to achieve an isotropic manipulability by sensing the human muscular manipulability. In this setting, a varying exoskeleton manipulability profile may be retrieved using GMR as a function of the sensed muscular manipulability.

From a mathematical point of view, it is worth highlighting the importance of considering the structure of the data we work with. While alternative solutions to handle SPD matrices are present in literature (e.g. those using Cholesky decomposition), we showed that Euclidean manipulability-tracking controllers lead to unstable behaviors in contrast to the stable behavior displayed by our geometry-aware controller (see Section 5.3). Equally important, the manipulability ellipsoids profiles retrieved by the geometry-aware and Euclidean GMR were similar only around the mean of the GMM components, but diverged when moving away from it. This is because the estimated output in Euclidean space is only a valid approximation for input data lying close to the mean, as reported in Section 3.5. Therefore, geometry-awareness is crucial for successful learning and tracking of manipulability ellipsoids in the proposed manipulability transfer framework.

6.5 Conclusion

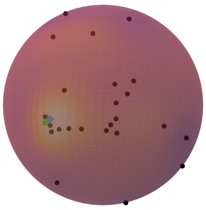
This chapter presented a novel framework for transferring manipulability ellipsoids to robots. The proposed approach is built on a probabilistic learning model that allows the encoding and retrieval of manipulability ellipsoids, and on the extension of the classical inverse kinematics problem to manipulability ellipsoids, by establishing a mapping between a change of manipulability ellipsoid and the robot joint velocity. We exploited tensor representation and Riemannian manifolds to build a geometry-aware learning framework and exponentially-stable tracking controllers and showed the importance of geometry-awareness for manipulability transfer. This approach enables the learning of posture-dependent task requirements and provides a skill transfer strategy going beyond the imitation of trajectory, forces or impedance behaviors. Furthermore, it allows manipulability transfer between agents of different embodiments, while taking into account their individual characteristics, and is adapted to complex scenarios involving any manipulability ellipsoid shape and various types of robots.

An open problem concerns the adaptation of the desired manipulability ellipsoid to exploit the capabilities of the learner in situations in which this learner can reach a better manipulability than the teacher for the task at hand. The next chapter makes a step towards solving this problem, by proposing a geometry-aware Bayesian optimization framework that seek optimal parameters lying on Riemannian manifolds.

Part III

Geometry-aware Policy Optimization

7 GaBO: Geometry-aware Bayesian Optimization



The Parts I and II of this thesis presented geometry-aware methods to learn symmetric positive definite (SPD) matrices from demonstration and to control robots to track the learned SPD sequences. After learning a skill from demonstrations, it is often desirable that the robot is able to refine or adapt this skill to a new situation. Therefore, the Part III of this thesis focuses on this problem.

In this chapter, we consider Bayesian optimization (BO) — known as a data-efficient and gradient-free approach — for direct policy search and propose a *geometry-aware Bayesian optimization* (GaBO) framework. Our approach, built on Riemannian manifold theory, allows BO to efficiently optimize non-Euclidean parameters.

Publication note

The material presented in this chapter is adapted from the following publication:

- Jaquier, N., Rozo, L., Calinon, S., and Bürger, M. (2019b). Bayesian optimization meets Riemannian manifolds in robot learning. In *Conference on Robot Learning (CoRL)* (Best presentation award).

The corresponding presentation can be watched at:

https://youtu.be/b7StSnt85S4?t=7763.

Supplementary material

A video illustrating the concepts presented in this chapter is available at:

https://sites.google.com/view/geometry-aware-bo.

Source code

Source codes related to this chapter are available at:

https://github.com/NoemieJaquier/GaBOtorch (pytorch-based code),
https://github.com/NoemieJaquier/GaBOflow (tensorflow-based code).

7.1 Introduction

In the previous chapters, we designed learning and control methods adapted to data lying on Riemannian manifolds. We showed the need and demonstrated the impact of introducing information about the geometry of the data into the proposed learning and control frameworks. In some cases, the skills learned by robots need to be refined or adapted to a new situation. This may for example be the case if we want to transfer a desired manipulability ellipsoid to a learner robot that may reach a better manipulability than the one demonstrated by the teacher for the task at hand. In this chapter, we propose to exploit the geometry of Riemannian manifolds to refine non-Euclidean robot control parameters and optimize parametric policies in a data-efficient, gradient-free and geometry-aware approach.

When robots learn new skills or adapt their behavior to unseen conditions, their learning process needs to be safe, fast and data-efficient as the robot is a physical system interacting with the environment, making every single interaction costly. In reinforcement learning (RL) for robotics, Bayesian Optimization (BO) [Shahriari et al., 2016] has gained increasing interest due to its success on optimizing parametric policies in several challenging scenarios [Cully et al., 2015; Englert and Toussaint, 2016; Marco et al., 2017; Rai et al., 2018]. Its popularity is due to its ability to model complex noisy cost functions in a data-efficient manner, contrasting to data-hungry methods used in deep RL [Arulkumaran et al., 2017]. However, BO performance degrades as the search space dimensionality increases, opening the door to different approaches dealing with the curse of dimensionality [Li et al., 2017; Rana et al., 2017b; Kirschner et al., 2019]. Its performance also depends on the generalization capabilities of Gaussian process (GP) models (the common surrogate model of BO), which is strongly impacted by the definition of both the mean and kernel functions.

A recent approach to improve BO performance is via domain knowledge, commonly introduced into the GP mean function [Cully et al., 2015] or through the design of task-specific kernels [Antonova et al., 2017], as detailed in Section 7.2. Nevertheless, several of these solutions are not task-agnostic, requiring new kernels as the task domain varies. A more scalable approach is to provide domain knowledge that generalizes over several tasks. Along this line, we propose to provide BO with information about the geometry of the search space, a key feature often overlooked in BO applications. Geometry-awareness is particularly relevant when the parameter space is not Euclidean, which is common in robotic applications, where a variety of manifolds arise [Ratliff et al., 2018; Zeestraten, 2018]. For example, forces and torques belong to the Euclidean manifold $\mathbb{R}^d$, stiffness, inertia and manipulability lie in the manifold of symmetric positive definite matrices $\mathcal{S}_{++}^d$, the special orthogonal group $SO(3)$ or the unit-sphere $\mathcal{S}^3$ are used to represent orientations, and the special Euclidean group $SE(3)$ describes robot poses.

We hypothesize that bringing geometry-awareness into BO may improve its performance and scalability. To do so, we bring Riemannian manifold theory to Bayesian optimization (see Section 7.3 for a short background). We first propose to use geometry-aware kernels allowing GP to properly measure the similarity between parameters lying on a Riemannian manifold. Second, we exploit Riemannian manifold tools to consider the geometry of the search space when optimizing the acquisition function (see Section 7.4). These two contributions lead to a fully geometry-aware BO framework (hereinafter called GaBO) which naturally handles the constraints of parameters lying on smooth differentiable manifolds. We test GaBO with different benchmark functions and, as a proof of concept, in a set of simulated scenarios aimed at refining simple control policies for robotic manipulation (see Section 7.5). Our results show that GaBO outperforms the classical BO formulation in convergence, accuracy and scalability as the space dimensionality increases.

7.2 Related Work

BO has been widely applied in diverse robotic applications, such as behavior adaptation for damaged legged robots [Cully et al., 2015], controller tuning for balancing [Marco et al., 2016], biped locomotion [Antonova et al., 2017; Rai et al., 2018], whole body control [Yuan et al., 2019], physical human-robot interaction [Ghadirzadeh et al., 2016; Kupcsik et al., 2015], and manipulation [Drieß et al., 2017]. A key aspect of the success of BO is the use of domain knowledge, mainly introduced into the surrogate model or the acquisition function. This prior information aims at decreasing the problem complexity and improving the convergence and accuracy of BO. This section reviews how domain knowledge has been exploited to improve BO performance.

Cully et al. [2015] used simulated walking behaviors as prior knowledge that allowed a robot to quickly adapt to drastic hardware changes. Pautrat et al. [2018] extended this idea and exploited the GP mean function to introduce prior knowledge about the robot behavior, which was later used to learn new tasks using a pool of given priors. They also proposed a new acquisition function that harmonized the expected improvement and the prior model likelihood. Wilson et al. [2014] used trajectory data extracted during policy execution to learn the initial state distribution, the transition and reward functions. These were used to generate Monte-Carlo estimates of policy performance that were used to define the GP mean function. The authors also introduced a Kullback-Leibler divergence kernel that used trajectory information to measure the relatedness between policies. In contrast to these works, we do not focus on imposing prior knowledge through the GP mean function but instead exploit the geometry of the parameter space to drive the BO exploration.

Marco et al. [2016] tuned the parameters of a pole-balancing robot controller using BO with an acquisition function maximizing the information gain. The authors emphasized that a GP with common kernels may degrade the learning outcome [Marco et al., 2017], and thus kernels leveraging the controller structure may be preferred. Thus, they proposed two kernels exploiting the structure of linear quadratic regulators, which outperformed the common squared-exponential (SE) kernel. The shortcomings of the SE kernels were also analyzed by Martinez-Cantin [2017], who proposed a set of adaptive kernels designed to model functions from nonstationary processes. This proved useful in direct policy search where failures result in large discontinuities or flat regions while fast variations are observed around the optimal policy. Rai et al. [2018] used a gait feature transformation to generalize a range of locomotion controllers and robot morphologies. This transform reparameterized the original space of controller parameters to a 1D space, where a classical SE kernel was used. Note that such transformation can also be interpreted as a metric function embedded into the SE kernel. In contrast to previous works which introduced prior knowledge into the BO kernel by relying on domain experts, Antonova et al. [2017] used a neural network to learn a kernel using simulated bipedal locomotion patterns. This neural-network kernel could be learned from the BO cost directly or from the trajectories in a cost-agnostic manner. Unlike these works, we do not design task-specific kernels but instead exploit geometry-aware kernel functions that can be used for different tasks whose search spaces share the same geometry.

Another way to introduce domain knowledge into BO is via the acquisition function. Englert and Toussaint [2016] and Drieß et al. [2017] included domain knowledge into the tuning of robot control parameters through an acquisition function using success information embedded into a GP classifier, which was learned during task trials in the real robot. Other works exploit domain knowledge to deal with the BO curse of dimensionality. Yuan et al. [2019] proposed a search space partitioning method to tackle the high dimensionality of a whole-body quadratic programming controller. The acquisition function was given only a subset of independent physically-meaningful partitions of the parameter space. Along this line, our work includes domain knowledge into the acquisition function by exploiting geometry-aware optimization that handles parameters lying on Riemannian manifolds.

The work by Oh et al. [2018] is one of the few in BO literature where geometry-awareness is considered. The authors applied a cylindrical transformation to the search space to overcome boundary issues when a sphere-like domain is given, and used a geometry-aware kernel. Their method was more accurate, efficient and scalable when compared to state-of-the art BO. However, Oh et al. [2018] did not include geometry information into the optimization of the acquisition function. As mentioned previously, our approach is fully geometry-aware as both the kernel function and the optimization of the acquisition function consider the geometry of the search space.

7.3 Background: Bayesian Optimization

Bayesian optimization (BO) is a sequential search algorithm aiming at finding a global maximizer (or minimizer) of an unknown objective function f

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}}\, f(\boldsymbol{x}), \tag{7.1}$$

where $\mathcal{X} \subseteq \mathbb{R}^{D_\mathcal{X}}$ is some design space of interest, with $D_\mathcal{X}$ being the dimensionality of the parameter space. The black-box function f has no simple closed form, but can be observed point-wise by evaluating its value at any arbitrary query point $\boldsymbol{x}$ in the domain. This evaluation produces noise-corrupted (stochastic) outputs $y \in \mathbb{R}$ such that $\mathbb{E}[y|f(\boldsymbol{x})] = f(\boldsymbol{x})$, with observation noise σ.

In this setting, BO specifies a prior belief over the possible objective functions. Then, at each iteration n, this model is refined according to observed data $\mathcal{D}_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ via Bayesian posterior update. An acquisition function $\gamma_n : \mathcal{X} \mapsto \mathbb{R}$ is constructed to guide the search for the optimum. This function evaluates the utility of candidate points for the next evaluation of f; therefore, the next query point $\boldsymbol{x}_{n+1}$ is selected by maximizing γ_n, i.e.,

$$\boldsymbol{x}_{n+1} = \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}}\, \gamma_n(\boldsymbol{x}; \mathcal{D}_n). \tag{7.2}$$

After N queries, the algorithm makes a final recommendation $\boldsymbol{x}_N$, representing its best estimate of the optimizer.

The prior and posterior of f are commonly modeled using a Gaussian Process $\mathcal{GP}(\mu, k)$ with mean function $\mu : \mathcal{X} \mapsto \mathbb{R}$ and positive-definite kernel (or covariance function) $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ (see Chapter 2.4.2 for a background on GPs). Therefore, the function f follows a Gaussian prior $f(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$ as in (2.48). With $\tilde{\boldsymbol{x}}$ representing an arbitrary test point, the random variable $f(\tilde{\boldsymbol{x}})$ conditioned on observations is also normally distributed (2.49) with posterior mean and variance functions denoted here as $\mu_n(\tilde{\boldsymbol{x}})$ and $\sigma_n^2(\tilde{\boldsymbol{x}})$. The posterior mean and variance evaluated at any point $\tilde{\boldsymbol{x}}$ respectively represent the model prediction and uncertainty of the objective function at $\tilde{\boldsymbol{x}}$. In BO, these functions are exploited to select the next query $\boldsymbol{x}_{n+1}$ by means of the acquisition function. The mean and kernel functions completely specify the GP and thus the model of the function f. The most common choice for the mean function is a constant value, while the kernel typically has the property that close points in the input space have stronger correlation than distant points. One popular kernel is the squared-exponential (SE) kernel (2.52)

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \exp\left(-\beta\, d(\boldsymbol{x}_i, \boldsymbol{x}_j)^2\right),$$

where $d(\cdot, \cdot)$ denotes the distance between two observations and the parameters β and θ control the horizontal and vertical scale of the function. The kernel parameters and the observation noise are usually inferred via maximum likelihood estimation (MLE).

Algorithm 1: Geometry-aware Bayesian Optimization (GaBO)

```
Input: Initial set of observed data $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N_0}$ with $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathcal{M}$
Output: Final recommendation $\boldsymbol{x}_N$
for $n = 0, 1 \ldots, N$ do
    Update the hyperparameters of the geometry-aware GP model;
    Select the next query point $\boldsymbol{x}_{n+1} \in \mathcal{M}$ by optimizing the acquisition function
      on the manifold, i.e., $\boldsymbol{x}_{n+1} = \text{argmax}_{\boldsymbol{x} \in \mathcal{X}} \gamma_n(\boldsymbol{x}; \mathcal{D}_n)$;
    Query the objective function to obtain $y_{n+1}$ ;
    Augment the set of observed data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\boldsymbol{x}_{n+1}, y_{n+1})\}$ ;
end
```

The acquisition function balances exploitation (e.g. selecting the point with the highest posterior mean) and exploration (e.g. selecting the point with the highest posterior variance) using the information given by the posterior functions. In this chapter, we use an improvement-based acquisition function, namely, expected improvement (EI) [Močkus, 1975]. which is built on a function that incorporates the amount of improvement upon the previous best observation f_n^*, and can be analytically computed as follows

$$\gamma_{EI}(\boldsymbol{x}; \mathcal{D}_n) = (\mu_n(\boldsymbol{x}) - f_n^*)\Phi\left(\frac{\mu_n(\boldsymbol{x}) - f_n^*}{\sigma_n(\boldsymbol{x})}\right) + \sigma_n(\boldsymbol{x})\phi\left(\frac{\mu_n(\boldsymbol{x}) - f_n^*}{\sigma_n(\boldsymbol{x})}\right), \tag{7.3}$$

where $\Phi(\cdot)$ is the normal cumulative distribution function and $\phi(\cdot)$ represents the corresponding probability density function. For EI, the next query intuitively corresponds to the point where the expected improvement over the previous best observation f_n^* is maximal.

7.4 Bayesian Optimization on Riemannian Manifolds

In this section, we present the geometry-aware BO (GaBO) framework that naturally handles the cases where the design space of parameters $\mathcal{X}$ is a Riemannian manifold or a subspace of a Riemannian manifold, i.e. $\mathcal{X} \subseteq \mathcal{M}$. To do so, we first need to model the unknown objective function f with a Gaussian process adapted to manifold-valued data. This is achieved by defining geometry-aware kernels measuring the similarity of the parameters on the manifold. Moreover, the selection of the next query point $\boldsymbol{x}_{n+1}$ is achieved by optimizing the acquisition function on the manifold $\mathcal{M}$. The GaBO framework is summarized in Algorithm 1.

7.4.1 Geometry-aware Kernels

The choice of the kernel function is crucial for the GP as it encodes our prior about the function f. As our parameters $\boldsymbol{x}$ belong to a Riemannian manifold, it is relevant to

Table 7.1 – Selected values of $\beta_{\min}$ for different Riemannian manifolds used in this chapter.

$\mathcal{M}$	$\mathcal{S}^2$	$\mathcal{S}^3$	$\mathcal{S}^4$	$\mathcal{S}^2_{++}$	$\mathcal{S}^3_{++}$
$\beta_{\min}$	6.5	2.	1.2	0.6	0.2

include this *a priori* knowledge in the choice of the kernel. A straightforward approach to adapt distance-based kernels to Riemannian manifolds is to replace the Euclidean distance d by the geodesic distance $d_{\mathcal{M}}$ in the definition of the kernel. Thus, the geodesic generalization of the SE kernel is given by (see [Jayasumana et al., 2015])

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \theta \exp\left(-\beta\, d_{\mathcal{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j)^2\right). \tag{7.4}$$

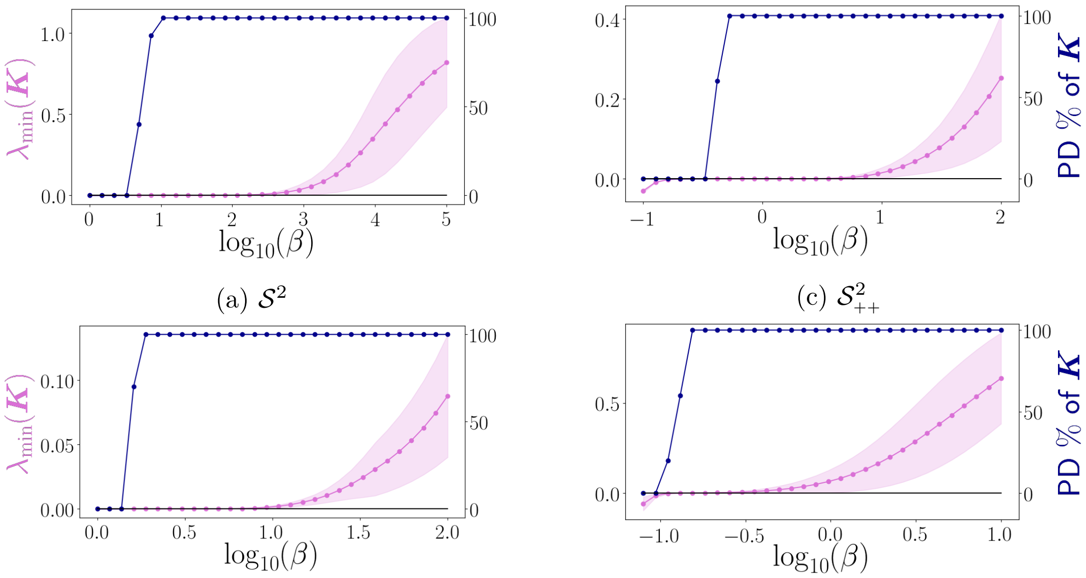
Although it has been successfully used in some applications, Feragen et al. [2015] showed that such a kernel is valid, i.e., positive definite (PD) for all the parameters values, only if the manifold is isometric to an Euclidean space. This implies that the geodesic SE kernel is not valid for curved manifolds such as $\mathcal{S}^d$ and $\mathcal{S}^d_{++}$. However, the same authors recently conjectured that there exists intervals of the lengthscale parameter $\beta > \beta_{\min}$ resulting in PD kernels [Feragen and Hauberg, 2016]. In this work, we follow this approach and determine experimentally the intervals of lengthscales β for which kernel matrices are PD for the manifolds of interest.

In order to compute $\beta_{\min}$, we sample 500 points from 10 Gaussian distributions on the manifold with random mean and covariance matrix $\boldsymbol{I}$. We then compute the corresponding kernel matrix $\boldsymbol{K}$, where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for a range of β values with $\theta = 1$. We repeat this process 10 times for each value of β and compute the percentage of PD geodesic kernel matrices $\boldsymbol{K}$. As the minimum eigenvalue function is continuous and $\boldsymbol{K}_{\beta\to\infty} \to \boldsymbol{I}$, we fix $\beta_{\min}$ equal to the minimum value of β for which 100% of the matrices $\boldsymbol{K}$ are PD. Figure 7.1 shows the percentage of PD geodesic kernel matrices and the distribution of their minimum eigenvalue $\lambda_{\min}$ as a function of β for $\mathcal{S}^3$ and $\mathcal{S}^3_{++}$. The values of $\beta_{\min}$ for the manifolds considered in this chapter are provided in Table 7.1.

Other types of kernels are available for specific manifolds and may also be used in BO [Oh et al., 2018]. For example, the geodesic Laplacian kernel is valid on spheres and hyperbolic spaces [Feragen et al., 2015]. Moreover, kernels have been specifically designed for several manifolds (see e.g. [Gong et al., 2012] for the Grassmannian).

Influence of $\beta_{\min}$ on GaBO

Introducing a minimum value on the parameter β of the kernel function (7.4) may influence the surrogate GP model and therefore the values of the acquisition function



(a) $\mathcal{S}^2$

(c) $\mathcal{S}^2_{++}$

(b) $\mathcal{S}^3$

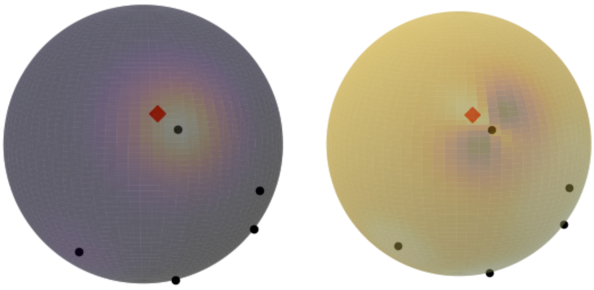
(d) $\mathcal{S}^3_{++}$

Figure 7.1 – Experimental selection of $\beta_{\min}$ in $\mathcal{S}^d$ and $\mathcal{S}^d_{++}$. The percentage of PD geodesic SE kernel matrices $\boldsymbol{K}$ computed from 10 different sets of 500 samples on the manifold is depicted in blue (right axis). The corresponding distribution of minimum eigenvalue $\lambda_{\min}$ of $\boldsymbol{K}$ is depicted in purple (left axis).

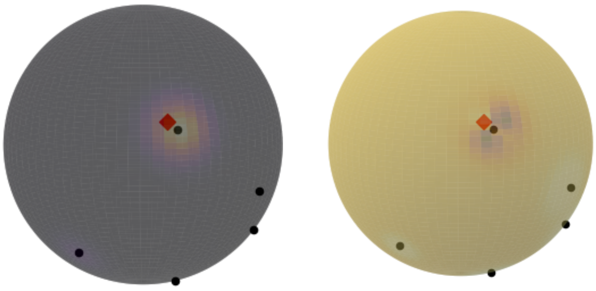
used to choose the next query point. We here illustrate and analyze the effect of $\beta_{\min}$ on GaBO. Overall, the impact of β_{min} on the BO framework will remain limited in most of the cases.

The value of β determines the spatial influence of the observations in the Gaussian process modeling the function f. In other words, the value of β controls the maximum size of the extrapolation region around one observation. Small β values permit to extrapolate in a large zone around one observation and therefore are well suited to model functions that evolve slowly with only few observations. For this type of functions, where the optimum β tends to be close to β_{min}, increasing β_{min} may result in a slight increase of the number of observations needed to properly model the function. Therefore, GaBO may need a few more query points to converge.

Figure 7.2 shows an example of the influence of the β value on the choice of the next query point. Two different cases with $\beta = 6.5$ and $\beta = 25$ are considered. The left graphs display the mean of a GP based on 5 observations on the sphere with colors ranging from yellow (low values) to dark purple (high values). The corresponding acquisition function is displayed on the right graphs with the same colors. The next query point, corresponding to the maximum of the acquisition function is depicted by a red square. We observe that the zone of influence of the observations is reduced for $\beta = 25$ compared to $\beta = 6.5$. This modifies the acquisition function, whose maximum is slightly closer to



(a) $\beta = 6.5$



(b) $\beta = 25$

Figure 7.2 – Illustration of the effect of β on GaBO. The GP mean and corresponding acquisition function are depicted by colors ranging from yellow (low values) to dark purple (high values) on the left and right graphs, respectively. The observations and next query point are depicted by black dots and a red square.

the observation for a higher value of β. Therefore, the number of query points needed to reach the optimum of the function may slightly increase if the value β_{min} is increased. Note that the difference between the query points obtained with $\beta = 6.5$ and $\beta = 25$ remains small for a consequent difference between the two tested β values. Therefore, a slight increase of β_{min} will have a limited impact on the number of iterations needed by GaBO to converge.

In the case of rapidly-varying functions that are therefore better modeled by β values significantly higher than β_{min}, a slight change on β_{min} will not impact GaBO.

7.4.2 Optimization of Acquisition Functions

After refining the geometry-aware GP that models the unknown function f, the next query point $\boldsymbol{x}_{n+1}$ is selected by maximizing the acquisition function γ_n. In order to take into account the geometry of the domain $\mathcal{X} \subseteq \mathcal{M}$, we propose to optimize γ_n using optimization techniques on manifolds. Note that the acquisition functions are not altered but their search space is modified. In this context, optimization algorithms on Riemannian manifolds constitute a powerful alternative to constrained optimization. These geometry-aware algorithms reformulate constrained problems as unconstrained optimization problems on manifolds and consider the intrinsic structure of the space of interest. Also, they tend to show lower computational complexity and better numerical properties [Hu et al., 2019]. In this section, we describe the two optimization algorithms used in the present and following chapters, namely the conjugate gradient (CG) and the trust-region (TR) algorithms on Riemannian manifolds.

Some problems may require to bound the search domain to a subspace, for example, to cope with physical limits or safety constraints in robotic systems when optimizing end-effector orientations or impedance parameters. In such cases, and particularly when the manifold is not a closed space, it is imperative to limit the domain of GaBO by defining boundary conditions inside the manifold. Therefore, the acquisition function is maximized over the domain $\mathcal{X} \subset \mathcal{M}$. While most of the literature on manifold optimization focuses on problems where the only constraint is that the solution belongs to the manifold, only few works proposed to extend constrained optimization algorithms on Riemannian manifolds. In this context, we propose to extend constrained CG and TR methods to Riemannian manifolds to cope with boundary conditions and linear constraints in the optimization.

Notice that some problems may require to optimize several parameters belonging to different manifolds. In this case, the domain of GaBO is a product of manifolds. Consequently, the kernel function corresponds to the product of the kernels on the different manifolds and the optimization of the acquisition function operates directly on the product of manifolds.

Conjugate Gradient on Riemannian Manifolds

In this chapter, we use the conjugate gradient (CG) algorithm on Riemannian manifolds [Smith, 1994], described in Algorithm 2, to maximize the acquisition function γ_n (or minimize $\phi_n = -\gamma_n$), at each iteration n of GaBO. The recursive process of the method involves the same steps as the Euclidean CG, namely: (*i*) a line minimization along the search direction $\boldsymbol{\eta}_k$ (step 3); (*ii*) the iterate update along the search direction $\boldsymbol{\eta}_k$ (step 4); and (*iii*) the computation of the next search direction $\boldsymbol{\eta}_{k+1}$ combining the gradient of the function at the new iterate $\boldsymbol{z}_{k+1}$ and the previous search direction $\boldsymbol{\eta}_k$ (steps 5-6). These different steps are illustrated for one iteration of the CG algorithm on the sphere manifold $\mathcal{S}^2$ in Figure 7.3. The differences with the Euclidean version are:

1. As the gradient $\nabla\phi(\boldsymbol{z}_k)$, and thus the search direction $\boldsymbol{\eta}_k$, belong to the tangent space of $\boldsymbol{z}_k$, the exponential map is needed to update the iterate along the search direction;
2. The step size α_k is fixed by solving

$$\operatorname*{argmin}_{\alpha_k} \phi_n\Big(\mathrm{Exp}_{\boldsymbol{z}_k}(\alpha_k \boldsymbol{\eta}_k)\Big), \tag{7.5}$$

with a linesearch on the manifold (an example of linesearch algorithm on manifolds is provided in Algorithm 3);

3. The previous search direction $\boldsymbol{\eta}_k$ has to be parallel transported to the tangent space of $\boldsymbol{z}_{k+1}$ to be combined with the gradient of the new iterate $-\nabla\phi_n(\boldsymbol{z}_{k+1}) \in \mathcal{T}_{\boldsymbol{z}_{k+1}}\mathcal{M}$.

Note that we presented the CG on manifold with the Hastenes-Stiefel update parameter β_k^{HS}, but other update techniques can also be extended to Riemannian manifolds [Absil et al., 2007b].

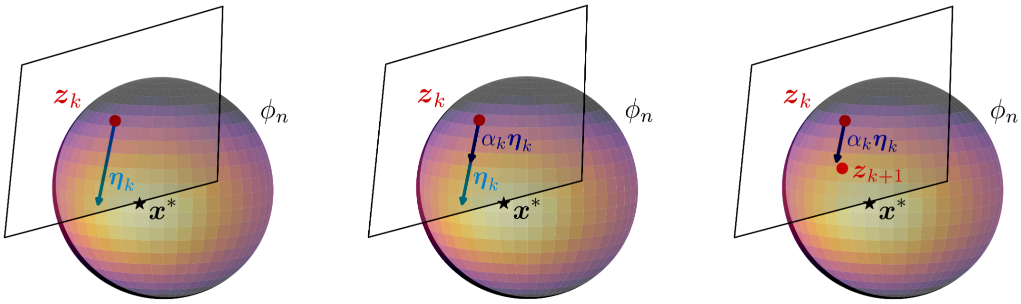
Algorithm 2: Optimization of acquisition function with conjugate gradient on Riemannian manifolds

```
Input: Acquisition function $\gamma_n$, initial iterate $\boldsymbol{z}_0 \in \mathcal{M}$
Output: Next parameter point $\boldsymbol{x}_{n+1}$
Set $\phi_n = -\gamma_n$ as the function to minimize and $\boldsymbol{\eta}_0 = -\nabla\phi_n(\boldsymbol{z}_0)$ as search direction ;
for $k = 0, 1, \dots, K$ do
    Compute the step size $\alpha_k$ using linesearch on Riemannian manifold (Algo. 3);
    Set $\boldsymbol{z}_{k+1} = \text{Exp}_{\boldsymbol{z}_k}(\alpha_k \boldsymbol{\eta}_k)$ ;
    Compute $\beta_k^{\text{HS}} = \frac{\langle \nabla\phi_n(\boldsymbol{z}_{k+1}), \nabla\phi_n(\boldsymbol{z}_{k+1}) - \nabla\phi_n(\boldsymbol{z}_k) \rangle_{\boldsymbol{z}_k}}{\langle \boldsymbol{\eta}_k, \nabla\phi_n(\boldsymbol{z}_{k+1}) - \nabla\phi_n(\boldsymbol{z}_k) \rangle_{\boldsymbol{z}_k}}$ ;
    Set $\boldsymbol{\eta}_{k+1} = -\nabla\phi_n(\boldsymbol{z}_{k+1}) + \beta_k^{\text{HS}} \, \Gamma_{\boldsymbol{z}_k \to \boldsymbol{z}_{k+1}}(\boldsymbol{\eta}_k)$ ;
    if a convergence criterion is reached then
        break
    end
end
Set $\boldsymbol{x}_{n+1} = \boldsymbol{z}_{k+1}$
```

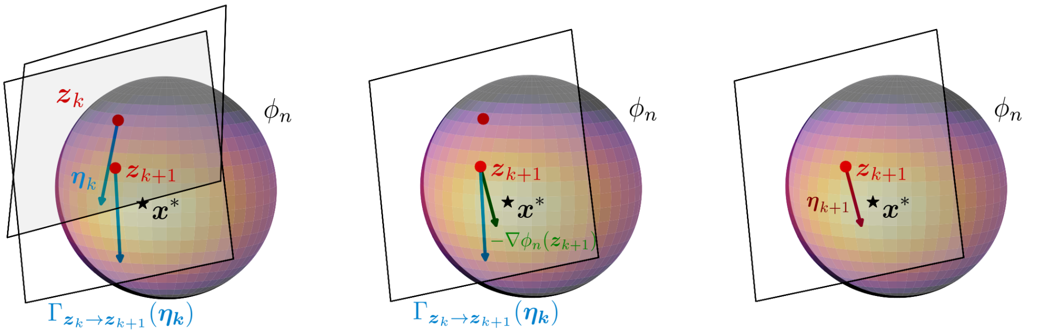
Algorithm 3: Adaptive linesearch for conjugate gradient on Riemannian manifolds

```
Input: Function $\phi_n = -\gamma_n$, iterate $\boldsymbol{z}_k$, search direction $\boldsymbol{\eta}_k$, initial stepsize $\alpha_k$,
       contraction factor $c$
Output: Final stepsize $\alpha_k$
for $j = 0, 1 \dots, J$ do
    Set $\boldsymbol{z} = \text{Exp}_{\boldsymbol{z}_k}(\alpha_k \eta_k)$ ;
    if $\phi_n(\boldsymbol{z}) > \phi_n(\boldsymbol{z}_k) + 0.5\alpha_k \langle \nabla\phi_n(\boldsymbol{z}_k), \boldsymbol{\eta}_k \rangle_{\boldsymbol{z}_k}$ then
        break
    end
    $\alpha_k = c\alpha_k$ ;
end
if $\phi_n(\boldsymbol{z}) > \phi_n(\boldsymbol{z}_k)$ then
    $\alpha_k = 0$
end
```

As mentioned previously, the acquisition function may need to be optimized over a bounded subspace of the manifold. In this context, we propose to extend the bound-



(a) Initial search direction (b) Step size computation (c) Updated iterate



(d) Search direction transport (e) Gradient computation (f) Updated search direction

Figure 7.3 – Illustration of one iteration of the conjugate gradient algorithm on the sphere. The function $\phi_n = -\gamma_n$ to minimize on $\mathcal{S}^2$ is depicted by colors ranging from yellow (low value) to dark purple (high value). Its optimum $\boldsymbol{x}^*$ is represented as a black star. (a) Initial state of the k-th iteration. The iterate $\boldsymbol{z}_k \in \mathcal{S}^2$ and the search direction $\boldsymbol{\eta}_k \in \mathcal{T}_{\boldsymbol{z}_k}\mathcal{S}^2$ are represented. (b) Computation of the step size α_k using linesearch (step 3 of Algo. 2). (c) Computation of the next iterate $\boldsymbol{z}_{k+1}$ (step 4 of Algo. 2). (d) Parallel transport of $\boldsymbol{\eta}_k \in \mathcal{T}_{\boldsymbol{z}_k}\mathcal{S}^2$ into the tangent space of the new iterate $\mathcal{T}_{\boldsymbol{z}_{k+1}}\mathcal{S}^2$. (e) Computation of the gradient $-\nabla\phi_n(\boldsymbol{z}_k) \in \mathcal{T}_{\boldsymbol{z}_{k+1}}\mathcal{S}^2$. (f) Update of the search direction (step 6 of Algo. 2).

constrained CG method [Vollebregt, 2014] to Riemannian manifolds to cope with boundary conditions in the optimization. To do so, the steps 4-6 of Algorithm 2 are updated as described in Algorithm 4. At each iteration, if the updated iterate $\boldsymbol{z}_{k+1} \notin \mathcal{X}$, it is projected back onto the feasible domain and the search direction is reinitialized. In $\mathcal{S}^d$, we bound the design space by setting limits on the components of $\boldsymbol{x} \in \mathcal{S}^d$. If a component is out of the limits, we fix it as equal to the closest limit and reformatted the remaining components so that $\boldsymbol{x}$ still belongs to the manifold. For $\mathcal{S}^d_{++}$, we define limits on the eigenvalues λ of the SPD matrix. If an iterate $\boldsymbol{z}_{k+1} \notin \mathcal{X}$, we project it back to the domain by reducing/increasing the maximum/minimum eigenvalue of the iterate.

Algorithm 4: Update of steps 4-6 of Algorithm 2 for a domain $\mathcal{X} \subset \mathcal{M}$

```
4 Set $\boldsymbol{z}_{k+1} = \mathrm{Exp}_{\boldsymbol{z}_k}(\alpha_k \boldsymbol{\eta}_k)$ ;
  if $\boldsymbol{z}_{k+1} \in \mathcal{X}$ then
      Execute steps 5-6 of Algorithm 2
  else
      Project $\boldsymbol{z}_{k+1}$ to $\mathcal{X}$;
      Set $\boldsymbol{\eta}_{k+1} = -\nabla\phi(\boldsymbol{z}_{k+1})$;
  end
```

Figure 7.4 shows an example of the application of Algorithm 4 on the unit circle $\mathcal{S}^1$. In this example, an acquisition function is optimized over the domain $\mathcal{S}^1$ with the constraint $y \leq 0.6$. Assume that the point initially proposed by the algorithm in step 4 is $\boldsymbol{z} = (x, y)^\mathsf{T} = (0.49, 0.87)^\mathsf{T}$, depicted by a red dot. This point does not satisfy the constraint $y \leq 0.6$, and needs to be projected back onto the feasible domain. As explained previously, we first fix the value of the y component to the closest limit and obtain the point $\tilde{\boldsymbol{z}} = (0.49, 0.6)^\mathsf{T}$, depicted by a yellow dot. Note that $\tilde{\boldsymbol{z}} \notin \mathcal{S}^1$ as its norm is not equal to 1, thus we need to project this point on the manifold. To guarantee that the bound constraints remain satisfied, we reformat only the components that were not affected by the constraints. In this example, y is fixed at 0.6 and only the component x varies to obtain a point on the manifold. Therefore, we obtain the final point $\boldsymbol{z}_{k+1} = (0.8, 0.6)^\mathsf{T} \in \mathcal{X}$, depicted by a blue dot.

Trust Region on Riemannian Manifolds

In the next chapter, we exploit trust-region (TR) methods on Riemannian manifolds, as introduced in [Absil et al., 2007a], to optimize the acquisition function γ_n at each iteration n of GaBO. The recursive process of the TR methods on Riemannian manifolds, described in Algorithm 5, involves the same steps as its Euclidean equivalence, namely: (i) the optimization of a quadratic subproblem m_k trusted locally, i.e., in a region around the iterate (step 3); (ii) the update of the trust-region parameters — typically the trust-region radius Δ_k — (steps 5-11); (iii) the iterate update, where a candidate is

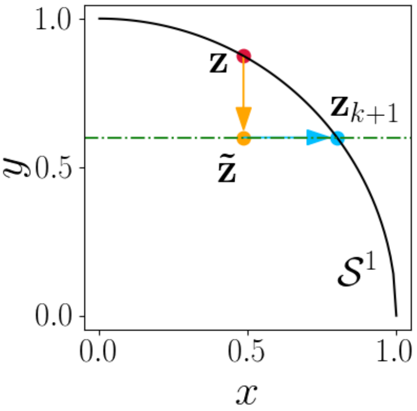


Figure 7.4 – Application of Algorithm 4 on $\mathcal{S}^1$. The manifold $\mathcal{S}^1$ is depicted by a black curve and the constraint $y \leq 0.6$ is shown by the green dotted line. The initial point $\boldsymbol{z}$, fixed point $\tilde{\boldsymbol{z}}$ and final point $\boldsymbol{z}_{k+1}$ are depicted by red, yellow and blue dots, respectively. Arrows show the order of the projections.

accepted or rejected in function of the quality of the model m_k (steps 12-16). The differences with the Euclidean version are:

1. The trust-region subproblem given by

$$\operatorname*{argmin}_{\boldsymbol{\eta} \in \mathcal{T}_{\boldsymbol{z}_k}\mathcal{M}} m_k(\boldsymbol{\eta}) \text{ s.t. } \|\boldsymbol{\eta}\|_{\boldsymbol{z}_k} \leq \Delta_k, \tag{7.6}$$

$$\text{with } m_k(\boldsymbol{\eta}) = \phi_n(\boldsymbol{z}_k) + \langle -\nabla\phi_n(\boldsymbol{z}_k), \boldsymbol{\eta}\rangle_{\boldsymbol{z}_k} + \frac{1}{2}\langle \boldsymbol{H}_k, \boldsymbol{\eta}\rangle_{\boldsymbol{z}_k}, \tag{7.7}$$

   is defined and solved in the tangent space $\mathcal{T}_{\boldsymbol{z}_k}\mathcal{M}$, with $\nabla\phi_n(\boldsymbol{z}_k) \in \mathcal{T}_{\boldsymbol{z}_k}\mathcal{M}$ and $\boldsymbol{H}_k$ some symmetric operator on $\mathcal{T}_{\boldsymbol{z}_k}\mathcal{M}$. Therefore, its solution $\boldsymbol{\eta}_k$ corresponds to the projection of the next candidate in the tangent space of the iterate $\boldsymbol{z}_k$. A truncated CG algorithm to solve the subproblem is provided in Algorithm 6.

2. As a consequence of the previous point, the candidate is obtained by computing $\text{Exp}_{\boldsymbol{z}_k}(\boldsymbol{\eta}_k)$.

The symmetric operator $\boldsymbol{H}_k$ on the tangent space $\mathcal{T}_{\boldsymbol{z}_k}\mathcal{M}$ typically approximates the Riemannian Hessian Hess $\phi_n(\boldsymbol{z}_k)\,[\boldsymbol{\eta}]$, which may be expensive to compute. For example, one may use the approximation of the Hessian with finite difference approximation introduced in [Boumal, 2015], that has been shown to retain global convergence of the Riemannian TR algorithm. Also notice that the steps 4 and 11 of Algorithm 6 correspond to solving the second-order equation

$$\langle \boldsymbol{\nu}_j, \boldsymbol{\nu}_j\rangle_{\boldsymbol{z}_k} + 2\tau_\Delta\langle \boldsymbol{\nu}_j, \boldsymbol{\delta}_j\rangle_{\boldsymbol{z}_k} + \tau_\Delta^2\langle \boldsymbol{\delta}_j, \boldsymbol{\delta}_j\rangle_{\boldsymbol{z}_k} = \Delta_k^2, \tag{7.8}$$

for τ_Δ, which was obtained from $\|\boldsymbol{\nu}_j + \tau_\Delta\boldsymbol{\delta}_j\|_{\boldsymbol{z}_k} = \Delta_k$ by using the relationship between the norm and the inner product and the properties of inner products.

For the cases where the domain of GaBO needs to be restricted to a subspace of the manifold, we propose to extend the TR algorithm to cope with linear constraints. Similarly

Algorithm 5: Optimization of acquisition function with trust region on Riemannian manifolds

```
Input: Acquisition function $\gamma_n$, initial iterate $\boldsymbol{z}_0 \in \mathcal{M}$, maximal trust radius
       $\Delta_{\max} > 0$, initial trust radius $\Delta_0 < \Delta_{\max}$, acceptance threshold $\rho$
Output: Next parameter point $\boldsymbol{x}_{n+1}$
Set $\phi_n = -\gamma_n$ as the function to minimize ;
for $k = 0, 1 \ldots, K$ do
    Compute the candidate $\mathrm{Exp}_{\boldsymbol{z}_k}(\boldsymbol{\eta}_k)$ by solving the subproblem
        $$\boldsymbol{\eta}_k = \operatorname*{argmin}_{\boldsymbol{\eta} \in \mathcal{T}_{\boldsymbol{z}_k}\mathcal{M}} m_k(\boldsymbol{\eta}) \text{ s.t. } \|\boldsymbol{\eta}\|_{\boldsymbol{z}_k} \leq \Delta_k,$$
     with $m_k(\boldsymbol{\eta}) = \phi_n(\boldsymbol{z}_k) + \langle -\nabla\phi_n(\boldsymbol{z}_k), \boldsymbol{\eta}\rangle_{\boldsymbol{z}_k} + \frac{1}{2}\langle \boldsymbol{H}_k, \boldsymbol{\eta}\rangle_{\boldsymbol{z}_k}$ (Algo. 6);
    Evaluate the accuracy of the model by computing $\rho_k = \frac{\phi_n(\boldsymbol{z}_k) - \phi_n\big(\mathrm{Exp}_{\boldsymbol{z}_k}(\boldsymbol{\eta}_k)\big)}{m_k(\boldsymbol{0}) - m_k(\boldsymbol{\eta}_k)}$;
    if $\rho_k < \frac{1}{4}$ then
        Reduce the trust radius $\Delta_{k+1} = \frac{1}{4}\Delta_k$ ;
    else if $\rho_k > \frac{3}{4}$ and $\|\boldsymbol{\eta}_k\|_{\boldsymbol{z}_k} = \Delta_k$ then
        Expand the trust radius $\Delta_{k+1} = \min(2\Delta_k, \Delta_{\max})$;
    else
        $\Delta_{k+1} = \Delta_k$ ;
    end
    if $\rho_k > \rho$ then
        Accept the candidate and set $\boldsymbol{z}_{k+1} = \mathrm{Exp}_{\boldsymbol{z}_k}(\boldsymbol{\eta}_k)$ ;
    else
        Reject the candidate and set $\boldsymbol{z}_{k+1} = \boldsymbol{z}_k$ ;
    end
    if a convergence criterion is reached then
        break
    end
end
Set $\boldsymbol{x}_{n+1} = \boldsymbol{z}_{k+1}$
```

Algorithm 6: Truncated conjugate gradient for solving the trust-region subproblem (step 3 of Algorithm 5)

```
Input: Trust-region subproblem 7.6 to minimize, given $\phi_n(\boldsymbol{z}_k)$, $\boldsymbol{H}_k$
Output: Update vector $\boldsymbol{\eta}_k$
Set the initial iterate $\boldsymbol{\nu}_0 = \boldsymbol{0}$, residual $\boldsymbol{r}_0 = \nabla\phi_n(\boldsymbol{z}_k)$ and search direction $\boldsymbol{\delta}_0 = -\boldsymbol{r}_0$;
for $j = 0, 1 \ldots, J$ do
    if $\langle \boldsymbol{\delta}_j, \boldsymbol{H}_k\boldsymbol{\delta}_j \rangle_{\boldsymbol{z}_k} \leq 0$ then
        Compute $\tau_\Delta \geq 0$ s.t. $\|\boldsymbol{\nu}_j + \tau_\Delta\boldsymbol{\delta}_j\|_{\boldsymbol{z}_k} = \Delta_k$ ;
        Set $\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \tau_\Delta\boldsymbol{\delta}_j$ ;
        break
    end
    Compute the step size $\alpha_j = \frac{\langle \boldsymbol{r}_j, \boldsymbol{r}_j \rangle_{\boldsymbol{z}_k}}{\langle \boldsymbol{\delta}_j, \boldsymbol{H}_k\boldsymbol{\delta}_j \rangle_{\boldsymbol{z}_k}}$ ;
    Set $\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \alpha_j\boldsymbol{\delta}_j$ ;
    if $\|\boldsymbol{\nu}_{j+1}\|_{\boldsymbol{z}_k} \geq \Delta_k$ then
        Compute $\tau_\Delta \geq 0$ s.t. $\|\boldsymbol{\nu}_j + \tau_\Delta\boldsymbol{\delta}_j\|_{\boldsymbol{z}_k} = \Delta_k$ ;
        Set $\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \tau_\Delta\boldsymbol{\delta}_j$ ;
        break
    end
    Set $\boldsymbol{r}_{j+1} = \boldsymbol{r}_j + \alpha_j\boldsymbol{H}_k\boldsymbol{\delta}_j$;
    Set $\boldsymbol{\delta}_{j+1} = -\boldsymbol{r}_{j+1} + \frac{\langle \boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1} \rangle_{\boldsymbol{z}_k}}{\langle \boldsymbol{r}_j, \boldsymbol{r}_j \rangle_{\boldsymbol{z}_k}}\boldsymbol{\delta}_j$ ;
    if a convergence criterion is reached then
        break
    end
end
Set $\boldsymbol{\eta}_k = \boldsymbol{\nu}_{j+1}$
```

to the Euclidean case [Byrd et al., 1987; Yuan, 1999], the trust-region subproblem can be augmented as

$$\operatorname*{argmin}_{\boldsymbol{\eta}\in\mathcal{T}_{\boldsymbol{z}_k}\mathcal{M}} m_k(\boldsymbol{\eta}) \text{ s.t. } \|\boldsymbol{\eta}\|_{\boldsymbol{z}_k} \leq \Delta_k^2 \text{ and } \|(\boldsymbol{c}_k + \nabla\boldsymbol{c}_k^{\mathsf{T}}\boldsymbol{\eta})^{-}\|_{\boldsymbol{z}_k} \leq \xi_k, \tag{7.9}$$

where $\boldsymbol{c}_k$ is a vector of linearized constraints $\boldsymbol{c}_k = (c_1(\boldsymbol{z}_k)\dots c_M(\boldsymbol{z}_k))^{\mathsf{T}}$, $\nabla\boldsymbol{c}_k$ is the corresponding gradient, $(x)^{-} = x$ for equality constraints $c_m(\boldsymbol{z}_k) = 0$ and $(x)^{-} = \min(0, x)$ for inequality constraints $c_m(\boldsymbol{z}_k) \geq 0$. The subproblem (7.9) can be solved with the augmented Lagrangian or the exact penalty methods on Riemannian manifolds presented in [Liu and Boumal, 2019].

In the context of Bayesian optimization, a common assumption is that the optimum should not lie in the border of the search space. Therefore, the acquisition function does not need to be exactly maximized close to the border of the search space. However, it is important to stay in the search space to cope with physical limits or safety constraints of the robotic system. By exploiting these two considerations, we propose to optimize the subproblem (7.9) in a simplified way, by adapting Algorithm 6 to cope with the constraints. At each iteration, we verify that the iterate $\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \alpha_j\boldsymbol{\delta}_j$ satisfies the constraints. If the constraints are not satisfied, the value of the step size α_j is adjusted and the algorithm is terminated. This process is described in Algorithm 7 and is used to augment the steps 5, 12 and 14 of Algorithm 6. Note that the proposed approach ensures that the constraints are satisfied, but is not guaranteed to converge to optima lying on a constraint border. However, we did not observe any significant difference in the performance of GaBO by using this approach compared to more sophisticated methods.

Algorithm 7: Addition to steps 5, 12 and 14 of Algorithm 6 to solve the trust-region subproblem (7.9).

```
Set $\boldsymbol{c}_k = c(\boldsymbol{z}_k)$ ;
if $\|(\boldsymbol{c}_k + \nabla\boldsymbol{c}_k^{\mathsf{T}}\boldsymbol{\nu}_{j+1})^{-}\|_{\boldsymbol{z}_k} \geq 0$ then
    Compute $\tau_c \geq 0$ s.t. $\|\left(\boldsymbol{c}_k + \nabla\boldsymbol{c}_k^{\mathsf{T}}(\boldsymbol{\nu}_j + \tau_c\boldsymbol{\delta}_j)\right)^{-}\|_{\boldsymbol{z}_k} = 0$;
    Set $\boldsymbol{\nu}_{j+1} = \boldsymbol{\nu}_j + \tau_c\boldsymbol{\delta}_j$ ;
    break
end
```

Other Optimization Methods on Riemannian Manifolds

We described the two optimization algorithms that we used to maximize the acquisition function in the GaBO framework and we proposed to extend them to cope with boundary conditions on Riemannian manifolds. These two algorithms lead to satisfying performance to optimize acquisition functions in the context of GaBO (see also Section 7.5) and have a low computational cost compared to some other optimization algorithms on Riemannian manifolds. Notice that, as the solutions obtained with CG or TR may depend on the initial iterate, a good practice consists in starting the optimizations from several initial points and select the best found solution as the next query point of the BO.

It is worth noticing that other optimization algorithms on Riemannian manifolds may be used, see e.g., [Bonnabel, 2013] for a stochastic gradient descent algorithm, [Borckmans et al., 2010] for an extension of particle swarm optimization and [Absil et al., 2007b] for algorithms related to Newton's method. In particular, the Riemannian augmented Lagrangian method and the Riemannian exact penalty method introduced in [Liu and Boumal, 2019] have been shown to efficiently cope with equality and inequality constraints and may be exploited to bound the search domain to a subspace of a manifold. Compared to these two approaches, the advantage of the constrained CG an TR algorithms on Riemannian manifolds proposed in this chapter is that they do not require any parameters tuning.

7.5 Experiments

We test GaBO using some benchmark test functions and two simulated experiments with a 7-DOF Franka Emika Panda robot. All the implementations were built on the Python libraries GPflow [Matthews et al., 2017], GPflowOpt [Knudde et al., 2017] and Pymanopt [Townsend et al., 2016]. The simulated experiments were performed using Pyrobolearn [Delhaisse et al., 2019]. All the BO implementations use EI as acquisition function and were initialized with 5 random samples.

7.5.1 Benchmark Functions

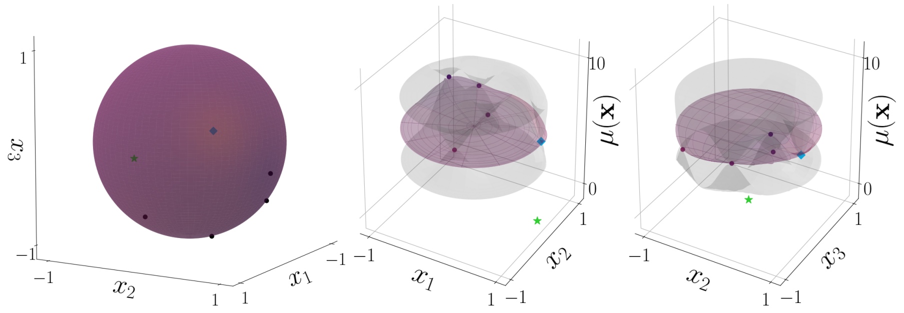
We use a couple of benchmark test functions to study the performance of GaBO in the Riemannian manifolds $\mathcal{S}^d$ and $\mathcal{S}^d_{++}$. To do so, we first project the test functions to these manifolds and then carry out the optimization by running 100 trials with random initialization. The selection of the kernel parameters is carried out by MLE. The search domain $\mathcal{X}$ of the test functions corresponds to the complete manifold for $\mathcal{S}^d$ and to SPD matrices with eigenvalues $\lambda \in [0.001, 5]$ for $\mathcal{S}^d_{++}$.

Figure 7.5 shows an example of the evolution of the surrogate model of GaBO for the Ackley function on $\mathcal{S}^2$. The left column displays the mean of the GP model on the sphere with colors ranging from yellow (low values) to dark purple (high values). The observed points $\boldsymbol{x}_n$ are depicted by black dots. The global minimum $\boldsymbol{x}^* = (0, 0, 0)^\mathsf{T}$ is shown as a green star and the current best guess is depicted by a blue square. The middle and right columns show the GP model on the sphere projected on the dimensions x_1, x_2 and x_2, x_3, respectively. The mean value of the GP is represented on the vertical dimension with the same colors as the left column. The variance of the model, represented with two standard deviations, is depicted by a gray envelope around the mean. We observe that the region around the optimum of the objective function is well modeled by GP model after 10 iterations. The current best guess is also close to the optimum. After 30 iterations, the GP results in a good approximation of the objective function with a low variance in the region around the optimum value.

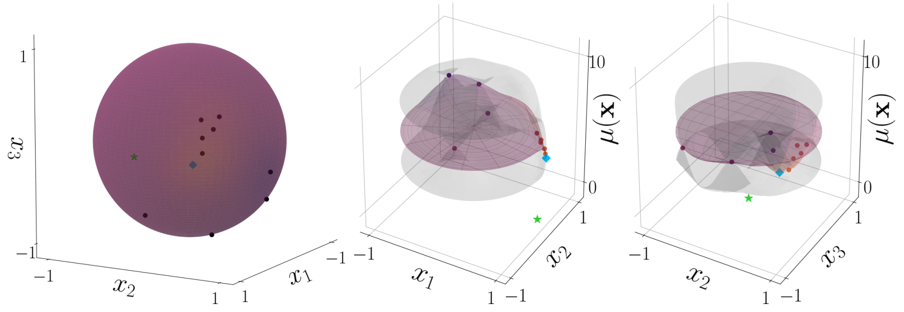
In the case of $\mathcal{S}^d$, we compare GaBO against the classical BO which carries out all the operations in the Euclidean space (hereinafter called Euclidean BO). As the query points must belong to $\mathcal{S}^d$, i.e., $\|\boldsymbol{x}\| = 1$, the maximization of the acquisition is considered a constrained problem in Euclidean BO. We minimize the Ackley function for parameter spaces of dimensionality $d = \{2, 3, 4\}$ to analyze the methods performance as the parameter dimension increases. Figure 7.6 displays the median and the first and third quartiles of the logarithm of the simple regret along 80 BO iterations for the three aforementioned hypersphere manifolds. We observe that, although the performance of the two algorithms is comparable for low-dimensional hyperspheres, GaBO outperforms the Euclidean BO when the dimension increases: GaBO converges faster to a better optimizer with lower variance than Euclidean BO. This is particularly evident for the Ackley function on $\mathcal{S}^4$, where the median regret of Euclidean BO remains far from the minimum value and displays high variance, while GaBO converges to a value close to the optimum for all the trials after 70 iterations. The fact that GaBO can be slightly slower than the Euclidean BO to converge in $\mathcal{S}^2$ may be due to the relatively high value of β_{min} in the kernel (7.4) for this manifold. As explained in Section 7.4.1, high β_{min} limits the spatial influence of each observation on the modeling of the function f. Thus, more observations are needed to model slowly-evolving regions of the function. A solution to this is to bound the domain of the optimization to a subspace of the manifold, which is often necessary in most real applications, and to determine the value β_{min} of the kernel for this subspace, resulting in a lower β_{min}.

Regarding the manifold $\mathcal{S}^d_{++}$, we compare our method against the Euclidean BO (augmented with the constraint $\lambda_{\text{min}} > 0$) and an alternative implementation that takes advantage of the Cholesky decomposition of an SPD matrix $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$, so that the resulting parameter is the vectorization of the lower triangular matrix $\boldsymbol{L}$ (hereinafter called Cholesky BO). In this case, we test two functions, namely a bimodal Gaussian distribution and the Ackley function. Figure 7.7 shows the regret for 300 iterations of GaBO, Cholesky BO and Euclidean BO for the two test functions in $\mathcal{S}^2_{++}$. We observe that GaBO outperforms Cholesky and Euclidean BO in both cases.

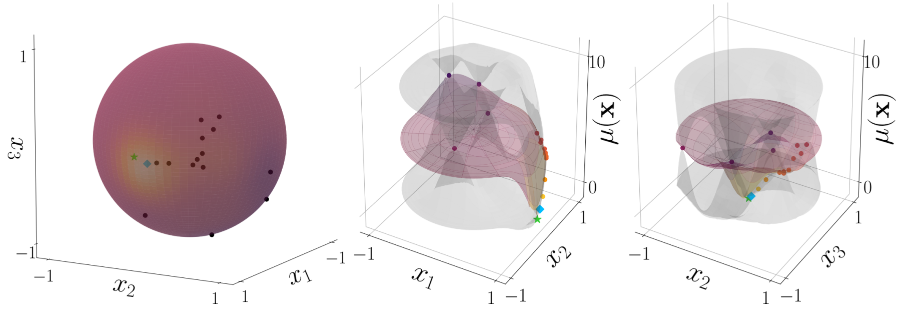
Table 7.2 provides the computation time per iteration of GaBO, Euclidean BO and Cholesky BO for the optimization of the Ackley function on the different manifolds used in the chapter. Overall, the computation time of GaBO is slightly higher than the Euclidean equivalent. The increase observed for $\mathcal{S}^2_{++}$ when compared to $\mathcal{S}^d$ is due to the fact that the exponential and logarithmic maps are more complex in $\mathcal{S}^D_{++}$ than their equivalent on the hyperspheres, as they involve matrix exponential and logarithm computations. However, the increase of the computation time is not a major drawback as



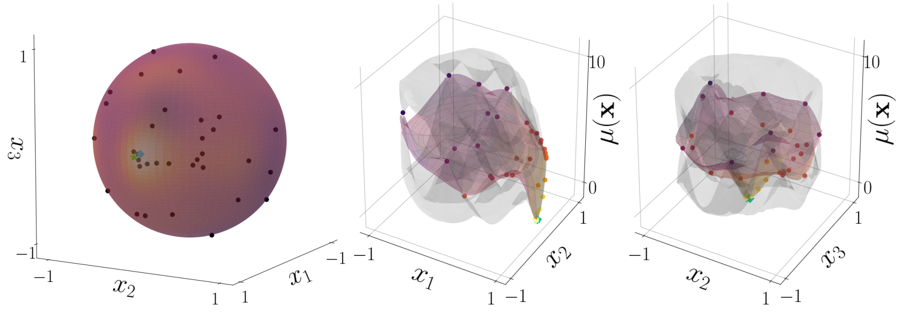
(a) Initial model



(b) Iteration 5

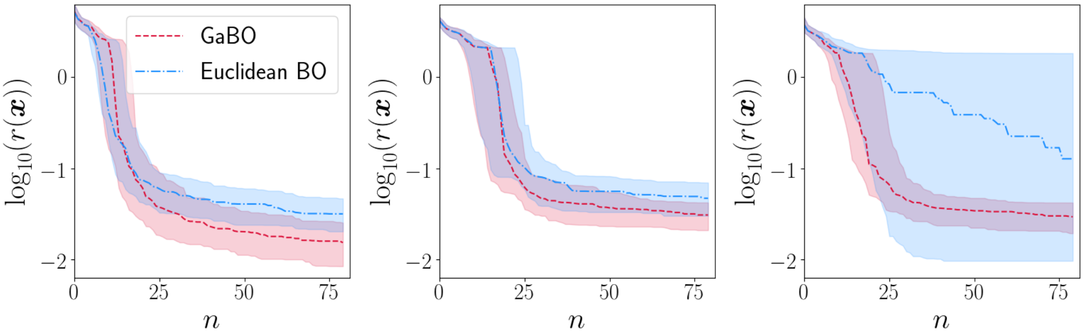


(c) Iteration 10



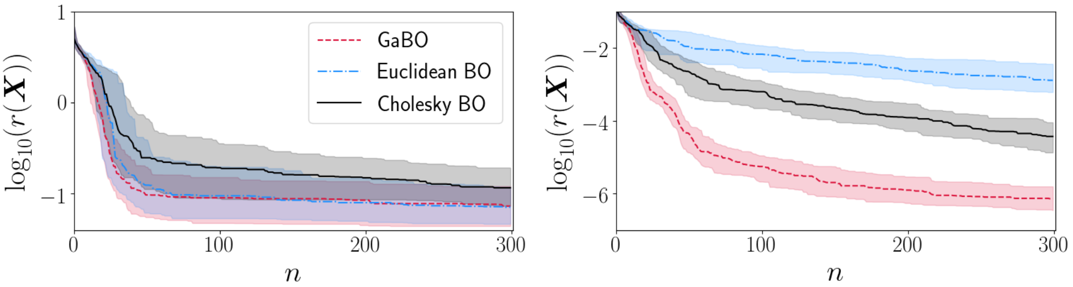
(d) Iteration 30

Figure 7.5 – Surrogate GP of GaBO for the minimization of the Ackley function in $\mathcal{S}^2$. The GP mean is depicted by colors ranging from yellow (low values) to dark purple (high values) and ± two standard deviations in gray (projected graphs). The global minimum and the current best guess are shown as a green star and a blue square.



(a) $\mathcal{S}^2$ (b) $\mathcal{S}^3$ (c) $\mathcal{S}^4$

Figure 7.6 – Logarithm of the simple regret in function of the BO iterations for the Ackley function in $\mathcal{S}^d$ with $d = \{2, 3, 4\}$ over 100 trials. The first and third quartiles are represented with a light tube around the median line.



(a) Ackley function in $\mathcal{S}^2_{++}$ (b) Bimodal function in $\mathcal{S}^2_{++}$

Figure 7.7 – Logarithm of the simple regret for benchmark test functions, i.e., (a) Ackley function and (b) bimodal distribution, in the Riemannian manifold of SPD matrices $\mathcal{S}^2_{++}$ over 100 trials. The first and third quartiles are represented with a light tube around the median line for the three BO methods.

physical robotic experiments will take significantly longer than the optimization process to collect the next value of the function f.

Table 7.2 – Computation time for one iteration of the different BO frameworks when optimizing the Ackley function on several manifolds. All the time values are given in seconds.

	$\mathcal{S}^2$	$\mathcal{S}^3$	$\mathcal{S}^4$	$\mathcal{S}^2_{++}$
GaBO	1.55 ± 0.32	1.49 ± 0.42	1.94 ± 0.42	6.08 ± 5.36
Euclidean BO	0.48 ± 0.14	0.49 ± 0.17	0.53 ± 0.14	0.24 ± 0.17
Cholesky BO	-	-	-	0.54 ± 0.12

7.5.2 Simulated Robotic Experiments

We here evaluate GaBO performance when looking first for the optimal orientation for a simple regulation task, and second for the optimal stiffness matrix of an impedance control policy, which is of interest for variable impedance learning approaches. For both experiments, we use a simulated 7-DoF Franka Emika Panda robot. The robot was initialized from a joint position $(0., 0.3, 0., -1., 0., 1.5, 0.)^\mathsf{T}$ for all the experiments.

In the first experiment, we use BO as an orientation sampler aiming at satisfying the requirements defined by a cost function. This may be useful for tasks where the orientation reference of a controller needs to be refined to improve the task execution. In this experiment, the velocity-controlled robot samples an orientation reference $\boldsymbol{x} \equiv \hat{\boldsymbol{q}}_o$ around a prior orientation $\tilde{\boldsymbol{q}}_o$, fixed by the user, with the aim of minimizing the cost function

$$f(\boldsymbol{q}) = w_q \|\Delta(\tilde{\boldsymbol{q}}, \boldsymbol{q})\|^2 + w_{\boldsymbol{\tau}} \|\boldsymbol{\tau}\|^2 + w_m \text{cond}(\boldsymbol{M}), \tag{7.10}$$

where $\boldsymbol{q}$ is the current end-effector orientation, and $\text{cond}(\boldsymbol{M})$ is the condition number of the linear velocity manipulability ellipsoid. This cost function aims at minimizing the error between the prior and the current end-effector orientation with low joint torques and an isotropic manipulability ellipsoid. We run 30 trials with random initialization for the cost function (7.10) with parameters fixed as in Table 7.3. The corresponding prior orientation $\tilde{\boldsymbol{q}}_o$ is illustrated in Figure 7.8a. Figure 7.8b depicts the best final orientation obtained after 80 iterations of GaBO. We observe that the orientation is close to the prior $\tilde{\boldsymbol{q}}_o$. However, the prior and best final orientations do not completely match due to the additional terms in the cost (7.10) that is minimized by GaBO. Figure 7.9 shows the total cost for 80 iterations of GaBO and Euclidean BO. We observe that GaBO converges faster to a better optimizer with a close-to-zero variance over the trials. Note that the difference between the two methods is accentuated compared to the benchmark function in $\mathcal{S}^3$.

Table 7.3 – Cost parameter values for the orientation sampling experiment.

$\tilde{\boldsymbol{q}}$	w_q	$w_{\boldsymbol{\tau}}$	w_m
$(0.408, 0.408, 0, 0.816)^{\mathsf{T}}$	1	10^{-4}	0.1

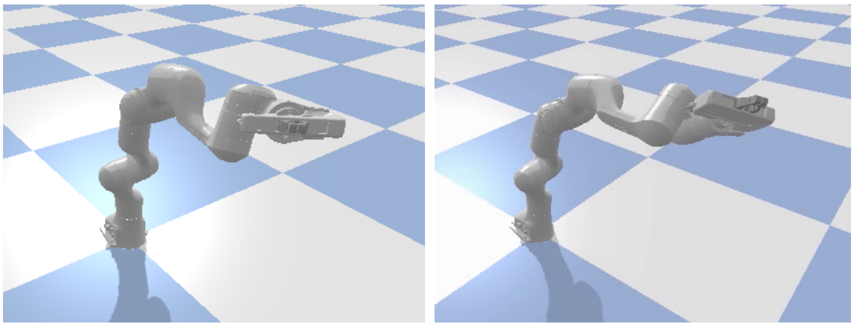

(a) Prior orientation (b) Optimized orientation

Figure 7.8 – Illustrations of the end-effector orientation learning. (a) shows the prior orientation $\tilde{\boldsymbol{q}}_o$ and (b) depicts the best orientation obtained after the optimization of the cost (7.10) with GaBO.

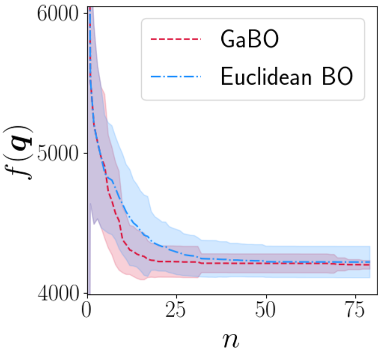


Figure 7.9 – Total cost for end-effector orientation learning in $\mathcal{S}^3$ over 30 trials. The mean and one standard deviation are represented.

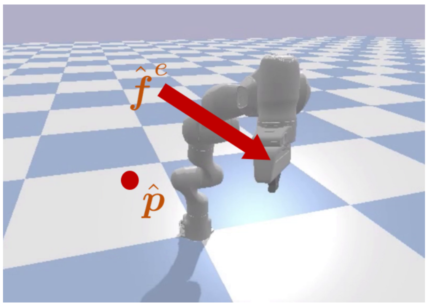


Figure 7.10 – Illustration of the direct policy search experiment. The robot is depicted in its initial configuration. The desired final position $\hat{\boldsymbol{p}}$ and the external force $\boldsymbol{f}^e$ are indicated by a red dot and a red arrow, respectively.

Since direct policy search has been a successful RL approach in robotics, for our second experiment, we seek to find the optimal Cartesian stiffness of a torque-controlled 7-DOF robotic arm implementing a Cartesian control policy

$$\boldsymbol{f} = \boldsymbol{K}^{\mathcal{P}}(\hat{\boldsymbol{p}} - \boldsymbol{p}) - \boldsymbol{K}^{\mathcal{D}}\dot{\boldsymbol{p}}, \tag{7.11}$$

where $\boldsymbol{p}$ and $\dot{\boldsymbol{p}}$ are the linear position and velocity of the robot end-effector, $\boldsymbol{K}^{\mathcal{P}}$ and $\boldsymbol{K}^{\mathcal{D}}$ are stiffness and damping matrices, and $\boldsymbol{f}$ is the control force (transformed to desired torques via $\boldsymbol{\tau} = \boldsymbol{J}^{\mathsf{T}}\boldsymbol{f}$). The robot task consists of tracking a desired Cartesian position $\hat{\boldsymbol{p}}$ while a constant external force $\boldsymbol{f}^e$ is applied to its end-effector (see Figure 7.10). The policy parameter corresponds to the stiffness matrix, that is $\boldsymbol{x} \equiv \boldsymbol{K}^{\mathcal{P}}$. The stiffness-damping ratio is fixed as critically damped. We tested GaBO, Euclidean BO and Cholesky BO using two different cost functions. The cost function f_1, defined as

$$f_1(\boldsymbol{K}^{\mathcal{P}}) = w_{\boldsymbol{p}}\|\hat{\boldsymbol{p}} - \boldsymbol{p}\|^2 + w_d \det(\boldsymbol{K}^{\mathcal{P}}) + w_c \mathrm{cond}(\boldsymbol{K}^{\mathcal{P}}), \tag{7.12}$$

aims at accurately tracking the desired position using a low-volume isotropic stiffness matrix, while f_2 aims at tracking the desired position accurately with low torques with

$$f_2(\boldsymbol{K}^{\mathcal{P}}) = w_{\boldsymbol{p}}\|\hat{\boldsymbol{p}} - \boldsymbol{p}\|^2 + w_{\boldsymbol{\tau}}\|\boldsymbol{\tau}\|^2, \tag{7.13}$$

for $\boldsymbol{K}^{\mathcal{P}} \in \{\mathcal{S}^2_{++}, \mathcal{S}^3_{++}\}$. For f_2, a -1 reward was added if the desired position was reached. The scalars w are parameters that weight the different components of the costs. In the case of $\mathcal{S}^2_{++}$, only a 2×2 submatrix of the full stiffness $\boldsymbol{K}^{\mathcal{P}}$ is optimized with BO, while the other components stay constant. Instead, all the components of the 3×3 matrix $\boldsymbol{K}^{\mathcal{P}}$ are optimized in $\mathcal{S}^3_{++}$. We run 15 randomly initialized trials for each cost function. The parameters of the functions were fixed as given in Tables 7.4 and 7.5.

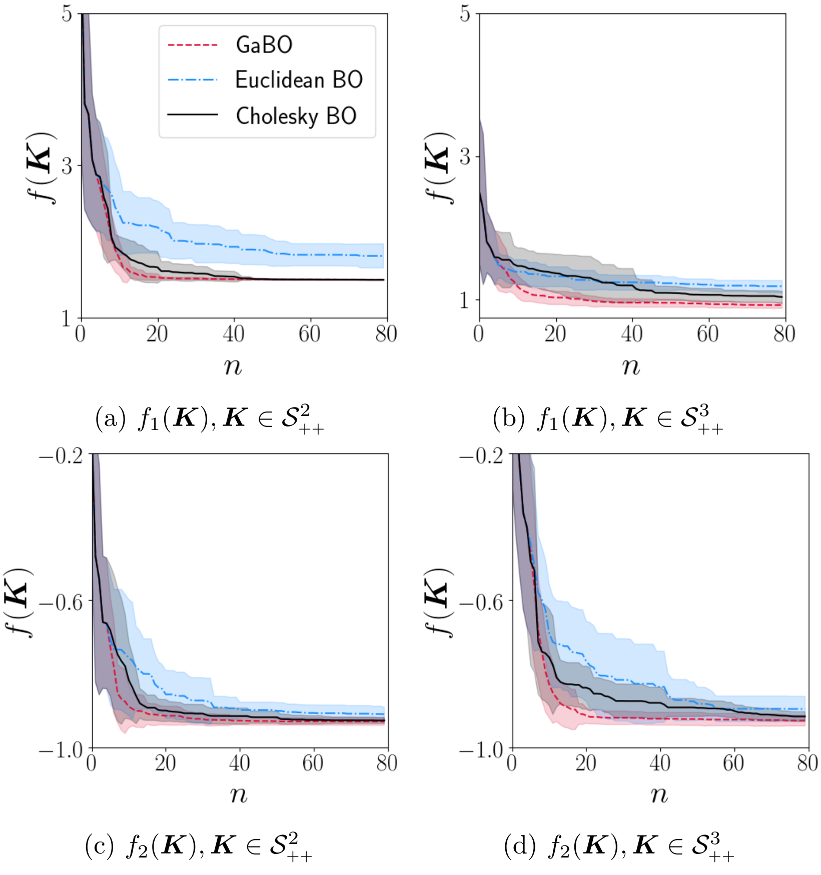
Figure 7.11 shows the total cost of the stiffness learning for the two cost functions in $\mathcal{S}^2_{++}$ and $\mathcal{S}^3_{++}$. We observe that Cholesky BO tends to outperform Euclidean BO as the complexity of the cost is increased, and that GaBO outperforms the other methods for all the test cases. Moreover, while the performance of Euclidean and Cholesky BO strongly degrades as the dimensionality increases, GaBO still provides accurate and low-variance solutions.

Table 7.4 – Cost parameter values of f_1 in the optimal stiffness learning experiment.

$\hat{\boldsymbol{p}}$ [m]	$\dot{\boldsymbol{p}}$	$w_{\boldsymbol{p}}$	w_d	w_c	$\boldsymbol{f}^e$ [N]
$(0.66, -0.01, 0.69)^\mathsf{T}$	$\mathbf{0}$	1	10^{-13}	10^{-4}	$(0, 20, -20)^\mathsf{T}$

Table 7.5 – Cost parameter values of f_2 in the optimal stiffness learning experiment.

$\hat{\boldsymbol{p}}$ [m]	$\dot{\boldsymbol{p}}$	$w_{\boldsymbol{p}}$	$w_{\boldsymbol{\tau}}$	$\boldsymbol{f}^e$ [N]
$(0.5, -0.4, 0.75)^\mathsf{T}$	$\mathbf{0}$	1	10^{-5}	$(0, 20, -20)^\mathsf{T}$



(a) $f_1(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^2_{++}$ (b) $f_1(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^3_{++}$

(c) $f_2(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^2_{++}$ (d) $f_2(\boldsymbol{K}), \boldsymbol{K} \in \mathcal{S}^3_{++}$

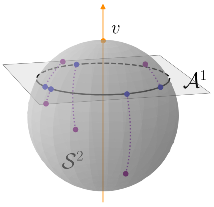
Figure 7.11 – Total cost for stiffness learning of a Cartesian impedance controller using direct policy search over 15 trials. The mean and one standard deviation are represented.

7.6 Conclusion

We proposed GaBO, a geometry-aware Bayesian optimization framework that exploited the geometry of the search space to properly seek optimal parameters that lie on Riemannian manifolds. To do so, we used geometry-aware kernels that allow GP to properly measure the similarity between parameters lying on a Riemannian manifold. Moreover, we exploited Riemannian manifold tools to consider the geometry of the search space when optimizing the acquisition function. GaBO provided faster convergence, better accuracy and lower solution variance when compared to geometry-unaware BO implementations. Interestingly, these differences were accentuated as the manifold dimensionality increased. Moreover, our proof-of-concept experiments open the door towards optimizing RL policies for complex robot learning scenarios where geometry-awareness may be relevant.

Following the line drawn by the aforementioned observations, the next chapter proposes to evaluate the GaBO framework in high-dimensional BO problems and to scale it towards optimization in high-dimensional search spaces lying on Riemannian manifolds.

8 Handling High-dimensional Problems with GaBO



In the previous chapter, we introduced GaBO, a geometry-aware Bayesian optimization framework that exploits the geometry of the search space to seek optimal parameters on Riemannian manifolds. We showed that GaBO outperforms geometry-unaware BO implementations, especially when the dimensionality of the search space increased.

In this chapter, we propose to scale GaBO towards optimization in high-dimensional search spaces. To do so, we first propose to evaluate the performance of GaBO in high-dimensional problems. Moreover, we introduce HD-GaBO, a high-dimensional GaBO framework minimizing objective functions on Riemannian manifolds that possess an intrinsic low-dimensionality.

Source code
Source codes related to this chapter are available at:
https://github.com/NoemieJaquier/GaBOtorch.

8.1 Introduction

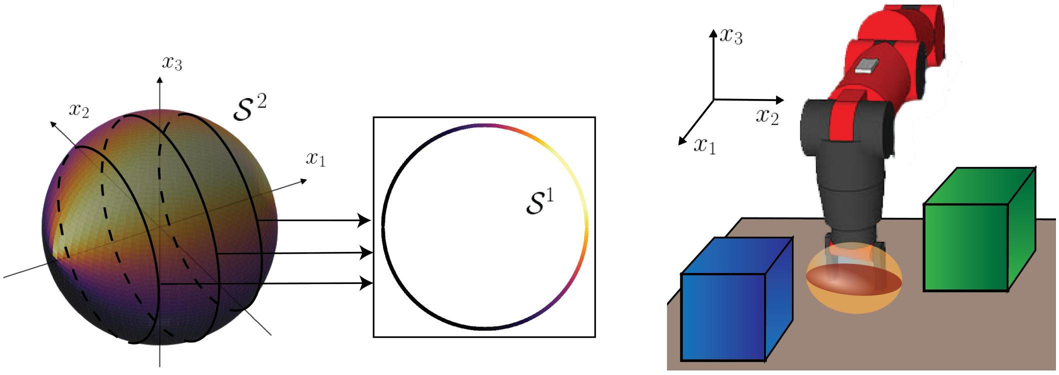
In the previous chapter, we showed that the performance of BO can be improved via the introduction of geometry-based domain knowledge. To do so, we presented GaBO, a geometry-aware BO framework that seeks optimal parameters on Riemannian manifolds. As mentioned in Section 7.1, BO performance is prone to degrade as the search space dimensionality increases, leading to research BO methods that can leverage the structure present in high-dimensional objective functions [Frazier, 2018]. In this context, the aim of this chapter is two-fold: (*i*) we evaluate GaBO for high-dimensional BO problems, and (*ii*) we propose to extend the GaBO framework towards minimizing objective functions that evolve along underlying low-dimensional latent spaces of Riemannian manifolds. We start this chapter by reviewing existing high-dimensional BO methods.

A common assumption in high-dimensional BO approaches is that the objective function depends on a limited set of features, i.e. that it evolves along an underlying low-dimensional latent space. Following this hypothesis, Wang et al. [2013] proposed REMBO, a solution based on random embeddings where the data are projected into a random linear space in which the optimization is carried out. Various extensions of REMBO have then been proposed in the literature, see e.g., [Munteanu et al., 2019; Binois et al., 2020]. Although these methods have been shown to perform well on a variety of problems at a low computational cost, they assume simple bound-constrained domains and may not be straightforwardly extended to complicatedly-constrained parameter spaces. Contrasting with random projections, other approaches proposed to learn the underlying low-dimensional — linear or nonlinear — latent space [Djolonga et al., 2013; Moriconi et al., 2019; Zhang et al., 2019]. While the latent space is often learned via unsupervised methods, Zhang et al. [2019] proposed to further exploit the evaluations of the objective function to determine the latent space in a supervised manner. Moriconi et al. [2019] developed an encoder-decoder-based BO model, where both parts are learned in a supervised manner. The encoder part is composed of a GP model that jointly learns a parametric mapping into a latent space and the regression parameters [Calandra et al., 2016], while the decoder part maps queries from the latent to the ambient space with a multi-output GP. After designing a sequential variational auto-encoder to embed the observations into a latent space, Antonova et al. [2019] also exploited the observed values of the objective function to compress the search space in the regions leading to undesirable behaviors. Although the aforementioned approaches shape the latent space as a function of the observed values of the objective function, the integration of a priory domain knowledge related to the parameter space is not considered in the learning process. Moreover, the decoder parts of the aforementioned approaches may not comply easily to recover query points in a complex parameter space.

Other relevant works in high-dimensional BO substitute or combine the low-dimensional assumption with an additive property, assuming that the objective function is decomposed as a sum of functions of low-dimensional sets of dimensions [Kandasamy et al., 2015; Li et al., 2016; Mutný and Krause, 2018]. Therefore, by considering a Gaussian process (GP) surrogate model for the BO process, the acquisition function of each group can be optimized independently. In a similar line, inspired by the dropout algorithm in neural networks, Li et al. [2017] proposed to deal with high-dimensional parameter spaces by optimizing only a random subset of the dimensions at each iteration. Similarly, Yuan et al. [2019] exploited domain knowledge to partition the parameter space into physically meaningful sets and alternatively optimize each subspace. Although the aforementioned strategies are well adapted for simple Euclidean parameter spaces, they may not generalize easily to complex domains. In the case where the parameter space is not Euclidean or must satisfy complex constraints, the problem of partitioning the space into subsets of specific dimensions becomes difficult. Moreover, these subsets may not be easily and independently optimized as they must satisfy global constraints acting on the parameters domain.

As discussed in the previous chapter, introducing domain-knowledge via geometry-aware kernels and acquisition functions has recently been shown to improve the performance and scalability of BO. Following this research line, we hypothesize that building and exploiting geometry-aware latent spaces may improve the performance of BO in high dimensions by considering the intrinsic geometry of the parameter space. Figure 8.1 illustrates this idea for Riemannian manifolds. The objective function on the sphere $\mathcal{S}^2$ depicted in Figure 8.1a does not depend on the value x_1 and is therefore better represented on the low-dimensional latent space $\mathcal{S}^1$. In the example represented in Figure 8.1b, the stiffness $\boldsymbol{X} \in \mathcal{S}^3_{++}$ of a robot is optimized for the robot to push objects lying on a table. In this case, the stiffness along the vertical axis x_3 does not influence the ability of the robot to push the objects. We may thus optimize the stiffness along the axes x_1 and x_2, which correspond to optimizing a SPD matrix in the latent space $\mathcal{S}^2_{++}$. Therefore, similarly to the high-dimensional BO frameworks where a Euclidean latent space of the Euclidean parameter space is exploited, here the objective functions may be efficiently represented in a latent space that inherits the geometry of the original Riemannian manifold. In general, this latent space is unknown and may not be aligned with the coordinate axes.

Following these observations, we propose a novel high-dimensional geometry-aware BO framework (hereinafter called HD-GaBO) for optimizing parameters lying on high-dimensional Riemannian manifolds. Our approach is based on a geometry-aware GP surrogate model that learns a mapping onto a latent space inheriting the geometry of the original space along with the representation of the objective in this latent space (see Section 8.2). The next query point is then selected on the low-dimensional Riemannian manifold using geometry-aware optimization methods. We evaluate the performance of GaBO and HD-GaBO on various high-dimensional benchmark functions. Our results show that bringing-geometry awareness into BO improves its scalability, especially for parameter spaces whose underlying geometry is complex. Moreover, we show that HD-GaBO is able to efficiently and reliably optimize high-dimensional objective functions that possess an intrinsic low dimensionality.



(a) $\mathcal{S}^2 \rightarrow \mathcal{S}^1$

(b) $\mathcal{S}^3_{++} \rightarrow \mathcal{S}^2_{++}$

Figure 8.1 – Illustration of the low-dimensional assumption on Riemannian manifolds. (a) The function on $\mathcal{S}^2$ is not influenced by the value of x_1 and may be represented more efficiently on the manifold $\mathcal{S}^1$. (b) The stiffness matrix of a robot is optimized to push objects on a table. As the stiffness along the axis x_3 does not influence the pushing performance, the cost function may be better represented in a latent space $\mathcal{S}^2_{++}$

8.2 High-dimensional Geometry-aware Bayesian Optimization

In this section, we present the high-dimensional geometry-aware BO (HD-GaBO) framework that naturally handles the case where the design space of parameters $\mathcal{X}$ is (a subspace of) a high-dimensional Riemannian manifold, i.e. $\mathcal{X} \subseteq \mathcal{M}^D$. We assume here that the objective function satisfies the low-dimensional assumption and thus only varies within a low-dimensional latent space. Moreover, we assume that this latent space can be identified as a low-dimensional Riemannian manifold $\mathcal{M}^d$ inheriting the geometry of the original manifold $\mathcal{M}^D$, with $d \ll D$. Notice that the same assumption is generally made by Euclidean high-dimensional BO frameworks, as the objective function is represented in a latent space $\mathbb{R}^d$ of $\mathbb{R}^D$. In particular, we model the objective function $f : \mathcal{M}^D \rightarrow \mathbb{R}$ as a composition of a structure-preserving mapping $m : \mathcal{M}^D \rightarrow \mathcal{M}^d$ and a function $g : \mathcal{M}^d \rightarrow \mathbb{R}$, so that $f = g \circ m$. A model of the objective function is thus available in the latent space $\mathcal{M}^d$, which is considered as the optimization domain to maximize the acquisition function. As the objective function can be evaluated only in the original space $\mathcal{M}^D$, the query point $\boldsymbol{z} \in \mathcal{Z}$, with $\mathcal{Z} \subseteq \mathcal{M}^d$, obtained by the acquisition function is projected back into the high-dimensional manifold with the inverse projection mapping $m^{-1} : \mathcal{M}^d \rightarrow \mathcal{M}^D$.

In HD-GaBO, the latent spaces are obtained via nested approaches on Riemannian manifolds featuring parametric structure-preserving mappings $m : \mathcal{M}^D \rightarrow \mathcal{M}^d$. Moreover, the parameters $\boldsymbol{\Theta}_m$ and $\boldsymbol{\Theta}_g$ of the mapping m and function g are determined jointly in a supervised manner using a geometry-aware GP model, as detailed in Section 8.2.1. Therefore, the observed values of the objective function are exploited not only to design the BO surrogate model, but also to drive the dimensionality reduction process towards expressive latent spaces for a data-efficient high-dimensional BO. Considering nested approaches also allows us to build a mapping m^{-1} that can be viewed as the pseudo-inverse of the mapping m. As explained in Section 8.2.3, the corresponding set of parameters $\boldsymbol{\Theta}_{m^{-1}}$ includes the projection mapping parameters $\boldsymbol{\Theta}_m$ and a set of reconstruction parameters $\boldsymbol{\Theta}_r$, so that $\boldsymbol{\Theta}_{m^{-1}} = \{\boldsymbol{\Theta}_m, \boldsymbol{\Theta}_r\}$. Therefore, the parameters $\boldsymbol{\Theta}_r$ are determined as to minimize the reconstruction error, as detailed in Section 8.2.2.

The proposed HD-GaBO framework is summarized in Algorithm 8. Similarly to GaBO, geometry-aware kernel functions are utilized in HD-GaBO (see Section 8.2.1), and the acquisition function is optimized using techniques on Riemannian manifolds, although the optimization is carried out on the latent Riemannian manifold in HD-GaBO.

Algorithm 8: HD-GaBO

```
Input: Initial observations $\mathcal{D}_0 = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N_0}$, $\boldsymbol{x}_i \in \mathcal{M}^D$, $y_i \in \mathbb{R}$
Output: Final recommendation $\boldsymbol{x}_N$
for $n = 0, 1 \dots, N$ do
    Update the hyperparameters $\{\boldsymbol{\Theta}_m, \boldsymbol{\Theta}_g\}$ of the geometry-aware mGP model ;
    Project the observed data into the latent space, so that $\boldsymbol{z}_i = m(\boldsymbol{x}_i)$ ;
    Select the next query point $\boldsymbol{z}_{n+1} \in \mathcal{M}^d$ by optimizing the acquisition function
      in the latent space, i.e., $\boldsymbol{z}_{n+1} = \text{argmax}_{\boldsymbol{z} \in \mathcal{Z}} \gamma_n(\boldsymbol{z}; \{(\boldsymbol{z}_i, y_i)\})$ ;
    Update the hyperparameters $\boldsymbol{\Theta}_{m^{-1}}$ of the inverse projection ;
    Obtain the new query point $\boldsymbol{x}_{n+1} = m^{-1}(\boldsymbol{z}_{n+1})$ in the original space ;
    Query the objective function to obtain $y_{n+1}$ ;
    Augment the set of observed data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\boldsymbol{x}_{n+1}, y_{n+1})\}$ ;
end
```

8.2.1 HD-GaBO Surrogate Model

The choice of latent spaces is crucial for the efficiency of HD-GaBO as it determines the search space for the selection of the next query point $\boldsymbol{x}_{n+1}$. In this context, it is desirable to base the latent-space learning process not only on the distribution of the observed parameters $\boldsymbol{x}_n$ in the original space, but also on the quality of the corresponding values y_n of the objective function. Therefore, we propose (*i*) to supervisedly learn a structure-preserving mapping onto a low-dimensional latent space, and (*ii*) to learn the representation of the objective function in this latent space along with the corresponding mapping. To do so, we exploit the so-called manifold Gaussian process (mGP) model introduced in [Calandra et al., 2016]. It is important to notice that the term *manifold* denotes here a latent space, whose parameters are learned by the mGP, which does not generally correspond to a Riemannian manifold.

In a mGP, the regression process is considered as a composition $g \circ m$ of a parametric projection mapping m onto a latent space and a function g. Specifically, a mGP is defined as a GP so that $f \sim \mathcal{GP}(\mu_m, k_m)$ with mean function $\mu_m : \mathcal{X} \to \mathbb{R}$ and positive-definite covariance function $k_m : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined as

$$\mu_m(\boldsymbol{x}) = \mu\Big(m(\boldsymbol{x})\Big), \tag{8.1}$$

$$k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) = k\Big(m(\boldsymbol{x}_i), m(\boldsymbol{x}_j)\Big), \tag{8.2}$$

with $\mu : \mathcal{Z} \to \mathbb{R}$ and $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ a positive-definite covariance function. The mGP parameters $\{\boldsymbol{\Theta}_m, \boldsymbol{\Theta}_g\}$ are estimated by maximizing the marginal likelihood of the model, so that

$$\{\boldsymbol{\Theta}_m^*, \boldsymbol{\Theta}_g^*\} = \underset{\boldsymbol{\Theta}_m, \boldsymbol{\Theta}_g}{\operatorname{argmax}}\, p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\Theta}_m, \boldsymbol{\Theta}_g). \tag{8.3}$$

In the original mGP model [Calandra et al., 2016], the original and latent spaces are subspaces of Euclidean spaces, so that $\mathcal{X} \subseteq \mathbb{R}^D$ and $\mathcal{Z} \subseteq \mathbb{R}^d$, respectively. Note that the idea of jointly learning a projection mapping and a representation of the objective function with a mGP was also exploited in the context of high-dimensional BO in the work of Moriconi et al. [2019]. In [Calandra et al., 2016; Moriconi et al., 2019], the mapping $m : \mathbb{R}^D \to \mathbb{R}^d$ was represented by a neural network. However, in the HD-GaBO framework, the design space of parameters $\mathcal{X} \subseteq \mathcal{M}^D$ is a high-dimensional Riemannian manifold and we aim at learning a geometry-aware latent space $\mathcal{Z} \subseteq \mathcal{M}^d$ that inherits the geometry of $\mathcal{X}$. Thus, we define a structure-preserving mapping $m : \mathcal{M}^D \to \mathcal{M}^d$ as a nested projection from a high- to a low-dimensional Riemannian manifold of the same type, as described in Section 8.2.3. Moreover, as in GaBO, we use a geometry-aware kernel function k that allows the GP to properly measure the similarity between parameters $\boldsymbol{z} = m(\boldsymbol{x})$ lying on the Riemannian manifold $\mathcal{M}^d$. Therefore, the surrogate model of HD-GaBO is a geometry-aware mGP, that leads to a geometry-aware representation of the objective function in a locally optimal low-dimensional Riemannian manifold $\mathcal{M}^d$.

Importantly, the predictive distribution for the mGP $f \sim \mathcal{GP}(\mu_m, k_m)$ at test input $\tilde{\boldsymbol{x}}$ is equivalent to the predictive distribution of the GP $g \sim \mathcal{GP}(\mu, k)$ at test input $\tilde{\boldsymbol{z}} = m(\tilde{\boldsymbol{x}})$. Therefore, the predictive distribution can be straightforwardly computed in the latent space. This allows the optimization function to be defined and optimized in the low-dimensional Riemannian manifold $\mathcal{M}^d$ instead of the original high-dimensional parameter space $\mathcal{M}^D$. Then, the selected next query point $\boldsymbol{z}_{n+1}$ in the latent space needs to be projected back onto the manifold $\mathcal{M}^D$ in order to evaluate the value of the objective function.

8.2.2 Input Reconstruction from the Latent Embedding to the Original Space

After optimizing the acquisition function, the selected query point $\boldsymbol{z}_{n+1}$ in the latent space needs to be projected back onto the manifold $\mathcal{M}^D$ in order to evaluate the value of the objective function. In order to solve this problem in the Euclidean case, Moriconi et al. [2019] proposed to learn a reconstruction mapping $r : \mathbb{R}^d \to \mathbb{R}^D$ based on multi-output GPs. In contrast, we propose here to further exploit the nested structure-preserving mappings in order to project the selected query point back onto the original manifold. As shown in Section 8.2.3, a pseudo-inverse parametric projection mapping $m^{-1} : \mathcal{M}^d \to \mathcal{M}^D$ can be built from the nested Riemannian manifold approaches. This inverse mapping depends on a set of parameters $\boldsymbol{\Theta}_{m^{-1}} = \{\boldsymbol{\Theta}_m, \boldsymbol{\Theta}_r\}$. Note that the parameters $\boldsymbol{\Theta}_m$ are learned with the mGP surrogate model, but we still need to determine the reconstruction parameters $\boldsymbol{\Theta}_r$. While the projection mapping m aimed at finding an optimal representation of the objective function, the corresponding pseudo-inverse mapping m^{-1} should (ideally) project the data $\boldsymbol{z}$ in the latent space $\mathcal{M}^d$ onto their original representation $\boldsymbol{x}$ in the original space $\mathcal{M}^D$. Therefore, the parameters $\boldsymbol{\Theta}_r$ are obtained by minimizing the sum of the squared residuals on the manifold $\mathcal{M}^D$, so that

$$\boldsymbol{\Theta}_r^* = \underset{\boldsymbol{\Theta}_r}{\operatorname{argmin}} \sum_{i=1}^{n} d^2_{\mathcal{M}^D}\Big(\boldsymbol{x}_i, m^{-1}(\boldsymbol{z}_i, \boldsymbol{\Theta}_m, \boldsymbol{\Theta}_r)\Big), \tag{8.4}$$

where $d_{\mathcal{M}^D}$ is the distance between two points on the manifold $\mathcal{M}^D$ (see also Section 2.2).

8.2.3 Nested Manifolds Mappings

As mentioned previously, the surrogate model of HD-GaBO learns to represent the objective function in a latent space $\mathcal{M}^d$ inheriting the geometry of the original space $\mathcal{M}^D$. To do so, the latent space is obtained via nested approaches, which map a high-dimensional Riemannian manifold to a low-dimensional latent space inheriting the geometry of the original Riemannian manifold. While various other dimensionality reduction techniques have been proposed on Riemannian manifolds [Fletcher and Joshi, 2004; Goh and Vidal, 2008; Sommer et al., 2010, 2014; Hauberg, 2016; Pennec, 2018], the resulting latent space is usually formed by curves on the high-dimensional manifold $\mathcal{M}^D$. This would require to optimize the acquisition function on $\mathcal{M}^D$ with complex constraints, which may not be handled efficiently by optimization algorithms. In contrast, nested manifold mappings reduce the dimension of the search space in a systematic and structure-preserving manner, so that the acquisition function can be efficiently optimized on a low-dimensional Riemannian manifold with optimization techniques on manifolds. Moreover, intrinsic latent spaces may naturally be encoded with nested manifold mappings in various applications (see Figure 8.1). Nested mappings for the sphere and SPD manifolds are presented in the following.

Sphere manifold

The concept of nested spheres, introduced in [Jung et al., 2012], is illustrated in Figure 8.2. Given an axis $\boldsymbol{v} \in \mathcal{S}^D$, the sphere is first rotated so that $\boldsymbol{v}$ aligns with the origin, typically defined as the north pole $(0, \dots, 0, 1)^\mathsf{T}$. Then, the data $\boldsymbol{x} \in \mathcal{S}^D$, depicted in purple, are projected onto the subsphere $\mathcal{A}^{D-1}$ defined as

$$\mathcal{A}^{D-1}(\boldsymbol{v}, r) = \{\boldsymbol{w} \in \mathcal{S}^D : d_{\mathcal{S}^D}(\boldsymbol{v}, \boldsymbol{w}) = r\}, \tag{8.5}$$

where $r \in (0, \pi/2]$, so that $x_D = \cos(r)$. The last coordinate of $\boldsymbol{x}$ is then discarded and the data $\boldsymbol{z} \in \mathcal{S}^{D-1}$, depicted by blue dots in Figure 8.2c, are obtained by identifying the subsphere $\mathcal{A}^{D-1}$ of radius $\sin(r)$ with the nested unit sphere $\mathcal{S}^{D-1}$ via a scaling operation. Specifically, given an axis $\boldsymbol{v}_D \in \mathcal{S}^D$ and a distance $r_D \in (0, \pi/2]$, the projection mapping $m_D : \mathcal{S}^D \to \mathcal{S}^{D-1}$ is computed as

$$\boldsymbol{z} = m_D(\boldsymbol{x}) = \underbrace{\frac{1}{\sin(r_D)}}_{\text{scaling}} \; \underbrace{\boldsymbol{R}_{\text{trunc}}}_{\substack{\text{rotation} \\ \text{+ dim. red.}}} \; \underbrace{\left(\frac{\sin(r_D)\boldsymbol{x} + \sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x}) - r_D\Big)\boldsymbol{v}_D}{\sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x})\Big)} \right)}_{\text{projection onto } \mathcal{A}^{D-1}}, \tag{8.6}$$

with $\boldsymbol{R} \in \mathrm{SO}(D)$ as the rotation matrix that moves $\boldsymbol{v}$ to the origin on the manifold and $\boldsymbol{R}_{\text{trunc}}$ the matrix composed of the $D-1$ first rows of $\boldsymbol{R}$. Notice also that the order of the projection and rotation operations is interchangeable. In (8.6), the data are simultaneously rotated and reduced after being projected onto $\mathcal{A}^{D-1}$. However, the same result may be obtained by projecting the rotated data $\boldsymbol{Rx}$ onto $\mathcal{A}^{D-1}$ using the rotated axis $\boldsymbol{Rv}$ and multiplying the obtained vector by the truncated identity matrix $\boldsymbol{I}_{\text{trunc}} \in \mathbb{R}^{D-1 \times D}$. This fact will be later exploited to define the SPD nested mapping. Then, the full projection mapping $m : \mathcal{S}^D \mapsto \mathcal{S}^d$ is defined via successive mappings (8.6), so that

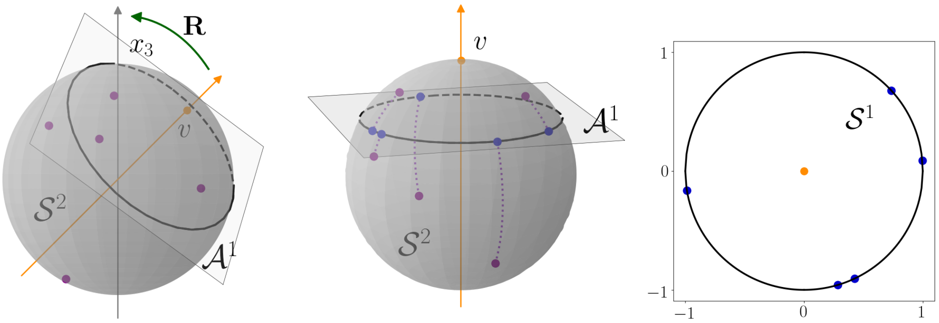
$$m = m_{d+1} \circ \ldots \circ m_{D-1} \circ m_D, \tag{8.7}$$

with parameters $\{\boldsymbol{v}_D, \ldots \boldsymbol{v}_{d+1}, r_D, \ldots r_{d+1}\}$ such that $\boldsymbol{v}_m \in \mathcal{S}^m$ and $r_m \in (0, \pi/2]$. Importantly, notice that the distance $d_{\mathcal{S}^d}(m(\boldsymbol{x}_i), m(\boldsymbol{x}_j))$ between two points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{S}^D$ projected onto $\mathcal{S}^d$ is invariant w.r.t the distance parameters $\{r_D, \ldots r_{d+1}\}$ (see Appendix C.1 for a proof). Therefore, when using distance-based kernels, the parameters set of the mGP projection mapping corresponds to $\boldsymbol{\Theta}_m = \{\boldsymbol{v}_D, \ldots \boldsymbol{v}_{d+1}\}$. The optimization (8.3) is thus carried out with optimization techniques on Riemannian manifolds on the domain $\mathcal{S}^D \times \cdots \times \mathcal{S}^{d+1} \times \mathcal{M}_g$, where $\mathcal{M}_g$ is the space of GP parameters $\boldsymbol{\Theta}_g$ (usually $\mathcal{M}_g \sim \mathbb{R} \times \ldots \times \mathbb{R}$).

As shown in [Jung et al., 2012], an inverse transformation $m_D^{-1} : \mathcal{S}^{D-1} \to \mathcal{S}^D$ can be computed as

$$\boldsymbol{x} = m_D^{-1}(\boldsymbol{z}) = \boldsymbol{R}^\mathsf{T} \begin{pmatrix} \sin(r_{d+1})\boldsymbol{z} \\ \cos(r_{d+1}) \end{pmatrix}. \tag{8.8}$$

Therefore, the query point selected by the acquisition in the latent space can be projected back onto the original space with the inverse projection mapping $m^{-1} : \mathcal{S}^d \to \mathcal{S}^D$ given



(a) Rotation of $\mathcal{S}^2$ (b) Projection onto $\mathcal{A}^1$ (c) $\mathcal{A}^1$ identified with $\mathcal{S}^1$

Figure 8.2 – Illustration of the nested sphere projection mapping. Data on the sphere $\mathcal{S}^2$, depicted by purple dots, are projected onto the subsphere $\mathcal{A}^1$, which is then identified with the sphere $\mathcal{S}^1$. The projected data are depicted as blue dots.

by

$$m^{-1} = m_D^{-1} \circ \ldots \circ m_{d+1}^{-1}. \tag{8.9}$$

As the axes parameters are determined within the mGP model, the set of reconstruction parameters is given by $\boldsymbol{\Theta}_r = \{r_D, \ldots, r_{d+1}\}$.

SPD manifold

Although not explicitly named as such, the dimensionality reduction technique for the SPD manifold introduced in [Harandi et al., 2014, 2018] can be understood as a nested manifold mapping. Specifically, Harandi et al. [2014, 2018] proposed a projection mapping $m : \mathcal{S}_{++}^D \to \mathcal{S}_{++}^d$, so that

$$\boldsymbol{Z} = m(\boldsymbol{X}) = \boldsymbol{W}^\mathsf{T} \boldsymbol{X} \boldsymbol{W}, \tag{8.10}$$

with $\boldsymbol{W} \in \mathbb{R}^{D \times d}$. Note that the matrix $\boldsymbol{Z} \in \mathcal{S}_{++}^d$ is guaranteed to be positive definite if $\boldsymbol{W}$ has a full rank. As proposed in [Harandi et al., 2014, 2018], this can be achieved without loss of generality by imposing orthogonality constraint on $\boldsymbol{W}$ such that $\boldsymbol{W} \in \mathcal{G}_{D,d}$, i.e., $\boldsymbol{W}^\mathsf{T}\boldsymbol{W} = \boldsymbol{I}$, where $\mathcal{G}_{D,d}$ denotes the Grassmann manifold corresponding to the space of d-dimensional subspaces of $\mathbb{R}^D$ [Edelman et al., 1998]. Therefore, in the case of the SPD manifold the projection mapping parameter set is $\boldsymbol{\Theta}_m = \{\boldsymbol{W}\}$. Specifically, the optimization (8.3) is carried out on the product of Riemannian manifolds $\mathcal{G}^{D,d} \times \mathcal{M}_g$.

In the previous chapter, we proposed to use the SE kernel based on the affine-invariant SPD distance (2.15) for GaBO on the SPD manifold. During the GP parameters optimization in GaBO, the distances between each pair of SPD data only depend on the data and are solely computed at the beginning of the optimization process. In contrast, in HD-GaBO, the distances between the projected SPD data vary as a function of $\boldsymbol{W}$ and therefore must be computed at each optimization step. This results in a computationally expensive optimization of the mGP parameters. In order to alleviate this computational burden, we propose to use the SE kernel based on the Log-Euclidean SPD distance $d_{\mathcal{S}_{++}^d}(\boldsymbol{X}_i, \boldsymbol{X}_j) = \|\log(\boldsymbol{X}_i) - \log(\boldsymbol{X}_j)\|_{\mathrm{F}}$ [Arsigny et al., 2006]. Moreover, as shown in [Harandi et al., 2018], we can approximate $\log(\boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{W}) \simeq \boldsymbol{W}^\top \log(\boldsymbol{X}) \boldsymbol{W}$, so that

$$d_{\mathcal{S}_{++}^d}(\boldsymbol{W}^\top \boldsymbol{X}_i \boldsymbol{W}, \boldsymbol{W}^\top \boldsymbol{X}_j \boldsymbol{W}) \simeq \|\boldsymbol{W}^\top \left(\log(\boldsymbol{X}_i) - \log(\boldsymbol{X}_j)\right) \boldsymbol{W}\|_{\mathrm{F}}. \tag{8.11}$$

Therefore, the difference between the logarithm of SPD matrices is fixed throughout the optimization process. This allows us to the optimize the mGP parameters at a lower computational cost without affecting consequently the performance of HD-GaBO. Note that the Log-Euclidean based SE kernel is positive definite for all the values of the parameter β [Jayasumana et al., 2015].

In order to project the query point $\boldsymbol{Z} \in \mathcal{S}_{++}^d$ back onto the original space $\mathcal{S}_{++}^D$, we propose to build an inverse projection mapping based on m. It can be easily observed that using the pseudo-inverse $\boldsymbol{W}$ so that $\boldsymbol{X} = \boldsymbol{W}^{\dagger^\top} \boldsymbol{Z} \boldsymbol{W}^\dagger$ does not guarantee the recovered matrix $\boldsymbol{X}$ to be positive definite. Therefore, we propose a novel inverse mapping inspired by the nested sphere projections. To do so, we observe that an analogy can be drawn between the mappings (8.6) and (8.10). Namely, the mapping (8.10) first consists of a rotation $\boldsymbol{R}^\top \boldsymbol{X} \boldsymbol{R}$ of the data $\boldsymbol{X} \in \mathcal{S}_{++}^D$ with $\boldsymbol{R}$ a rotation matrix whose D first columns equal $\boldsymbol{W}$, i.e., $\boldsymbol{R} = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{V} \end{pmatrix}$. Similarly to the nested sphere case, the rotated data can be projected onto a subspace of the manifold $\mathcal{S}_{++}^D$ by fixing their last coordinates. Therefore, the subspace is composed of matrices $\begin{pmatrix} \boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{W} & \boldsymbol{C} \\ \boldsymbol{C}^\top & \boldsymbol{B} \end{pmatrix}$, where $\boldsymbol{B} \in \mathcal{S}_{++}^{D-d}$ is a constant matrix. Finally, this subspace may be identified with $\mathcal{S}_{++}^d$ by multiplying the projected matrix $\begin{pmatrix} \boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{W} & \boldsymbol{C} \\ \boldsymbol{C}^\top & \boldsymbol{B} \end{pmatrix}$ with a truncated identity matrix $\boldsymbol{I}_{\text{trunc}} \in \mathbb{R}^{D \times d}$. Therefore, the mapping (8.10) may be equivalently expressed as

$$\boldsymbol{Z} = m(\boldsymbol{X}) = \boldsymbol{I}_{\text{trunc}}^\top \begin{pmatrix} \boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{W} & \boldsymbol{C} \\ \boldsymbol{C}^\top & \boldsymbol{B} \end{pmatrix} \boldsymbol{I}_{\text{trunc}} = \boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{W}. \tag{8.12}$$

From the properties of block matrices with positive block-diagonal elements, the projection is positive definite if and only if $\boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{W} \geq \boldsymbol{C} \boldsymbol{B} \boldsymbol{C}^\top$. This corresponds to defining the side matrix as $\boldsymbol{C} = (\boldsymbol{W}^\top \boldsymbol{X} \boldsymbol{W})^{\frac{1}{2}} \boldsymbol{K} \boldsymbol{B}^{\frac{1}{2}}$, where $\boldsymbol{K} \in \mathbb{R}^{d \times D-d}$ is a contraction matrix, so that $\|\boldsymbol{K}\| \leq 1$ [Bhatia, 2007]. Based on (8.12), the inverse mapping $m^{-1} : \mathcal{S}_{++}^d \to \mathcal{S}_{++}^D$ is given by

$$\boldsymbol{X} = m^{-1}(\boldsymbol{Z}) = \boldsymbol{R} \begin{pmatrix} \boldsymbol{Z} & \boldsymbol{Z}^{\frac{1}{2}} \boldsymbol{K} \boldsymbol{B}^{\frac{1}{2}} \\ \boldsymbol{B}^{\frac{1}{2}} \boldsymbol{K}^\top \boldsymbol{Z}^{\frac{1}{2}} & \boldsymbol{B} \end{pmatrix} \boldsymbol{R}^\top, \tag{8.13}$$

with reconstruction parameters $\boldsymbol{\Theta}_r = \{\boldsymbol{V}, \boldsymbol{K}, \boldsymbol{B}\}$. The optimization (8.4) is thus carried out on the product of manifolds $\mathcal{G}_{D-d,d} \times \mathbb{R}^{d,D-d} \times \mathcal{S}_{++}^{D-d}$ subject to $\|\boldsymbol{K}\| \leq 1$ and $\boldsymbol{W}^\top \boldsymbol{V} = \boldsymbol{0}$. The latter condition is necessary for $\boldsymbol{R}$ to be a valid rotation matrix. We solve this optimization problem with the augmented Lagrangian method on Riemannian manifolds introduced in [Liu and Boumal, 2019].

8.3 Experiments

In this section, we first evaluate the performance of GaBO in a high-dimensional BO setting using some high-dimensional benchmark test functions on the sphere and SPD manifolds. Then, we test the proposed HD-GaBO framework to optimize functions lying on a low-dimensional Riemannian manifold embedded in a high-dimensional space. Both GaBO and HD-GaBO use the geodesic generalization of the SE kernel (7.4) and their acquisition functions are optimized using trust region on Riemannian manifolds (see Algorithm 5). The other state-of-the-art approaches use the classical SE kernel (2.52) and the constrained acquisition functions are optimized using sequential least squares programming [Kraft, 1988]. All the implementations are built on the Python libraries GPyTorch [Gardner et al., 2018], BoTorch [Balandat et al., 2019] and Pymanopt [Townsend et al., 2016]. Here, all the tested methods use EI as acquisition function and are initialized with 5 random samples.

8.3.1 Optimization of High-dimensional Objectives with GaBO

We use several benchmark functions to study the performance of GaBO in high dimensions on the Riemannian manifolds $\mathcal{S}^D$ and $\mathcal{S}_{++}^D$. As in the previous chapter, the test functions are projected onto the manifolds. The search space corresponds to the complete manifold for $\mathcal{S}^D$ and to SPD matrices with eigenvalues $\lambda \in [0.001, 5]$ for $\mathcal{S}_{++}^D$. We carry out the optimization by running 30 trials with random initialization. The GP parameters are selected using MLE.

In the case of $\mathcal{S}^D$, we compare GaBO against the classical BO (hereinafter called Euclidean BO) and two high-dimensional BO approaches (namely, dropout BO [Li et al., 2017] and SIR-BO [Zhang et al., 2019]), which carry out all the operations in the Euclidean space. The dropout approach optimizes only a random subset of the dimensions at each iteration. Therefore, it does not make any particular assumption on the objective function. Moreover, it can be easily extended to a parameter space lying on a sphere by optimizing the vector $\boldsymbol{z}$ composed of randomly selected dimensions of $\boldsymbol{x}$ with the constraint $\|\boldsymbol{z}\| \leq 1$. The remaining dimensions are then adapted so that the query point $\boldsymbol{x}$ satisfies $\|\boldsymbol{x}\| = 1$. As suggested in [Li et al., 2017], we implement dropout with a mixed fill-in strategy with probability $p = 0.1$ of randomly choosing the value of the remaining dimensions. While the SIR-BO approach follows the low-dimensional assumption, it has been shown to perform well also in cases where the objective function lies in a high-dimensional space. As in Euclidean BO, the acquisition function of SIR-BO is optimized in the Euclidean space with the constraint $\|\boldsymbol{x}\| = 1$ to ensure that the query points belong to $\mathcal{S}^D$. Other approaches, such as the MGPC-BO of Moriconi et al. [2019], are not considered here due to the difficulty of adapting when the parameters lie on Riemannian manifolds.

We minimize the Ackley, Rosenbrock and Styblinski-Tang functions

$$f_{\text{Ackley}}(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + \exp(1),$$

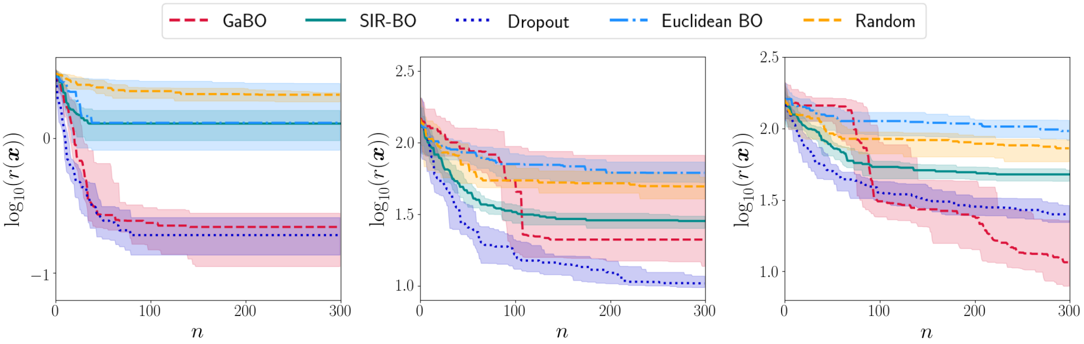
$$f_{\text{Rosenbrock}}(\boldsymbol{x}) = \sum_{i=1}^{D-1}\left(100(x_{i+1}-x_i^2)^2 + (x_i-1)^2\right),$$

$$f_{\text{Styblinski-Tang}}(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{D}\left((sx_i)^4 - 16(sx_i)^2 + 5(sx_i)\right) \text{ with } s = 3,$$

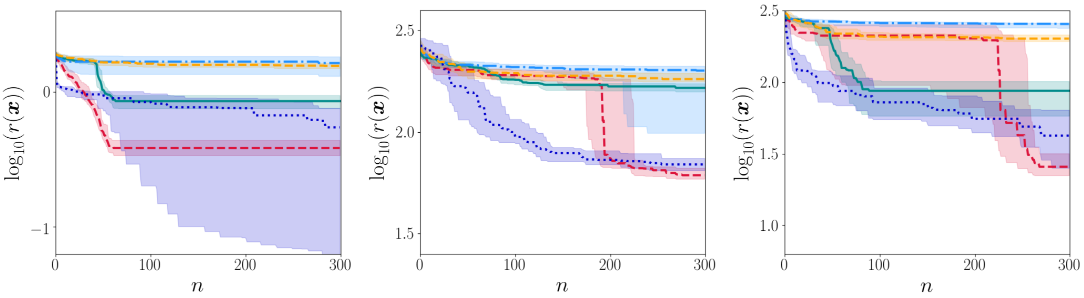
for dimensionality $D = \{10, 50\}$ to analyze the methods performance in spaces of different high dimensions. The latent space dimensionality is fixed at $d = 5$ for dropout and SIR-BO. Figure 8.3 displays the median and the first and third quartiles of the logarithm of the simple regret along 300 BO iterations for the two aforementioned sphere manifolds and three benchmark test functions. We observe that GaBO is competitive with respect to the high-dimensional BO approaches for all the functions and the considered dimensions. While GaBO initially shows a lower convergence rate than dropout and SIR-BO for the Rosenbrock and Styblinski-Tang functions, it always converges to a better optimizer and displays a lower variance than SIR-BO after 300 iterations. Moreover, GaBO also converges to a lower final value with a lower convergence variance than dropout for the functions in $\mathcal{S}^{50}$. Note also that Euclidean BO shows a poor performance for all the benchmark functions and both dimensions.

Regarding the manifold $\mathcal{S}_{++}^{D}$, we compare GaBO against the Euclidean BO and Euclidean SIR-BO (augmented with the constraint $\lambda_{\min} > 0$) and alternative implementations of BO, dropout and SIR-BO (hereinafter called Cholesky BO, Cholesky dropout and Cholesky SIR-BO) that exploit the Cholesky decomposition of an SPD matrix $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$. This means that the resulting parameter is the vectorization of the lower triangular matrix $\boldsymbol{L}$. Note that we do not consider here the Euclidean version of the dropout method due to the difficulty of optimizing a subset of a SPD matrix while satisfying the global constraint $\lambda_{\min} > 0$. We minimize the Ackley, Rosenbrock and Styblinski-Tang functions ($s = 5$) for dimensionality $D = \{5, 10\}$, which correspond to optimizing 15 and 55 parameters, respectively. The dimension of the latent space is fixed at $d = 3$ for dropout and SIR-BO.

Figure 8.4 displays the median and the first and third quartiles of the logarithm of the simple regret along 300 BO iterations for the three benchmark test functions on the SPD manifolds $\mathcal{S}_{++}^{5}$ and $\mathcal{S}_{++}^{10}$. We observe that, although it has not been specifically designed for high-dimensional functions, GaBO outperforms all the geometry-unaware BO methods for all the test cases: GaBO converges faster to a better optimizer than all the other approaches. Moreover, GaBO displays a faster convergence rate than all the other methods for the Rosenbrock and Styblinski-Tang functions. In the case of the Ackley function in $\mathcal{S}_{++}^{5}$, the variance of GaBO is low above, but high below

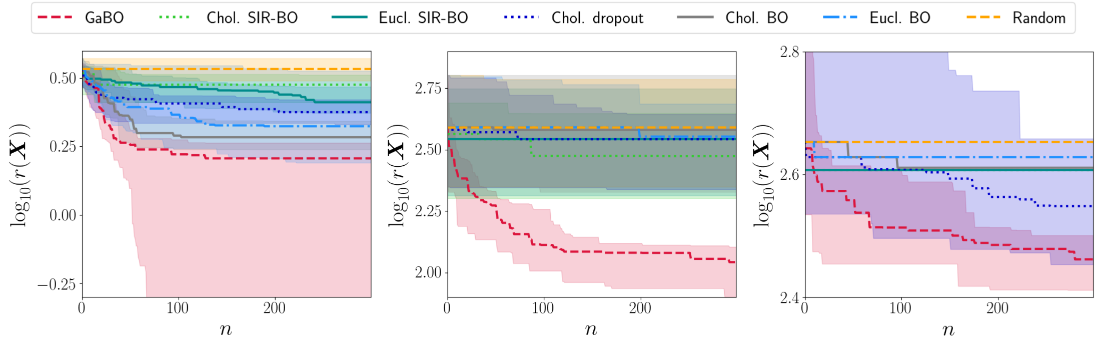


(a) Ackley, $\mathcal{S}^{10}$ (b) Rosenbrock, $\mathcal{S}^{10}$ (c) Styblinski-Tang, $\mathcal{S}^{10}$

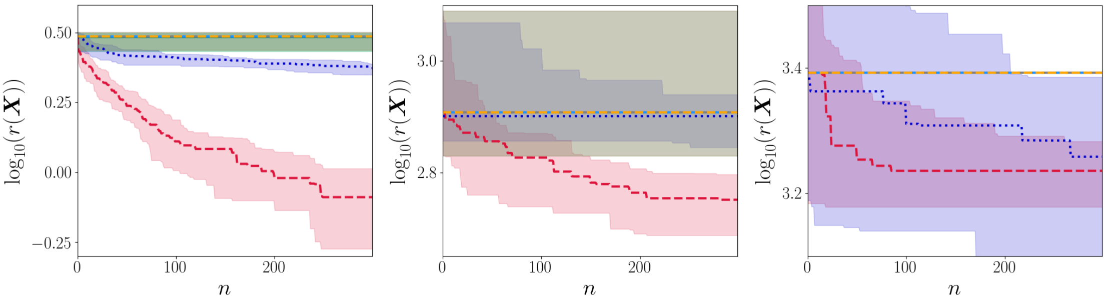


(d) Ackley, $\mathcal{S}^{50}$ (e) Rosenbrock, $\mathcal{S}^{50}$ (f) Styblinski-Tang, $\mathcal{S}^{50}$

Figure 8.3 – Logarithm of the simple regret for high-dimensional benchmark test functions in $\mathcal{S}^D$ with $d = \{10, 50\}$ over 30 trials. The first and third quartiles are represented with a light-color patch around the median line for the BO methods and the random search baseline.



(a) Ackley, $\mathcal{S}_{++}^{5}$ (b) Rosenbrock, $\mathcal{S}_{++}^{5}$ (c) Styblinski-Tang, $\mathcal{S}_{++}^{5}$



(d) Ackley, $\mathcal{S}_{++}^{10}$ (e) Rosenbrock, $\mathcal{S}_{++}^{10}$ (f) Styblinski-Tang, $\mathcal{S}_{++}^{10}$

Figure 8.4 – Logarithm of the simple regret for high-dimensional benchmark test functions in $\mathcal{S}_{++}^{D}$ with $D = \{5, 10\}$ over 30 trials. The first and third quartiles are represented with a light-color patch around the median line for the six BO methods and the random search baseline for the Ackley and Rosenbrock functions. For a better visualization of the Styblinski-Tang function, only the quartiles of GaBO and dropout BO are shown.

the median value, indicating that GaBO sometimes converges to an optimizer much closer to the optimum than the median. Moreover, while the variance of GaBO is higher than the other approaches for the Ackley function in $\mathcal{S}_{++}^{10}$, GaBO is also the only method whose optimizer is consequently lower than the initial values of the BO. In particular, these results highlight the benefit of incorporating domain knowledge into the BO framework and the importance of geometry-awareness for optimizing objective functions on Riemannian manifolds. It is interesting to notice that, while dropout showed good performance to optimize benchmark test functions on the sphere manifold, it is competitive with GaBO only for the Styblinski-Tang function on the SPD manifold. This may be attributed to the simpler geometry of the sphere manifold, which may be easier to handle for geometry-unaware approaches compared to the more complex geometry of the SPD manifold. Therefore, exploiting the geometry of the search space is even more crucial when the space dimension is high.

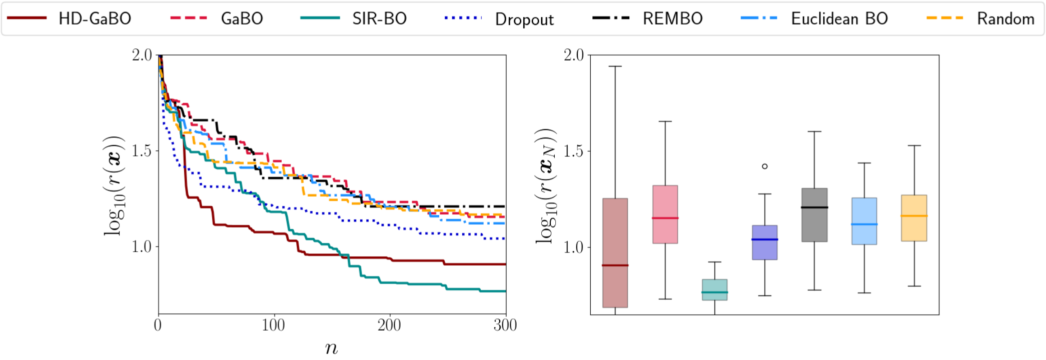
8.3.2 Optimization of High-dimensional Objectives with HD-GaBO

After studying the performance of GaBO for high-dimensional benchmark test functions lying on a Riemannian manifold, we evaluate the proposed HD-GaBO framework to optimize high-dimensional functions that lie on an intrinsic low-dimensional space. We consider benchmark test functions defined on a low-dimensional manifold $\mathcal{M}^d$ embedded in a high-dimensional manifold $\mathcal{M}^D$. Therefore, the test functions are defined as $f : \mathcal{M}^D \to \mathbb{R}$, so that $y = f(m(\boldsymbol{x}))$ with $m : \mathcal{M}^D \to \mathcal{M}^d$ being the nested projection mapping, as defined in Section 8.2.3. The projection mapping parameters are randomly set for each trial. The search space corresponds to the complete manifold for $\mathcal{S}^D$ and to SPD matrices with eigenvalues $\lambda \in [0.001, 5]$ for $\mathcal{S}^D_{++}$. We carry out the optimization by running 30 trials with random initialization. The GP parameters are estimated using MLE.

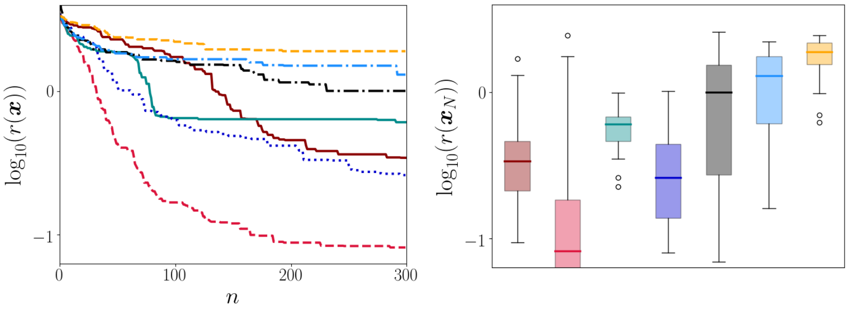
In the case of the sphere manifold $\mathcal{S}^D$, we compare HD-GaBO against GaBO, the Euclidean BO and three high-dimensional BO approaches, namely dropout BO [Li et al., 2017], SIR-BO [Zhang et al., 2019], and REMBO [Wang et al., 2013]. For Euclidean BO, dropout and SIR-BO, the query point is guaranteed to belong to the sphere as explained in the previous experiment section. In the case of REMBO, the next query point is given by $\boldsymbol{x}_{n+1} = \boldsymbol{A}\boldsymbol{z}_{n+1}$ with $\boldsymbol{z}_{n+1} \in \mathbb{R}^d$ obtained by optimizing the acquisition function, so that $\boldsymbol{z}_{n+1} = \mathrm{argmax}_{\boldsymbol{z}}\, \gamma_n(\boldsymbol{z}_{n+1}; \mathcal{D}_n)$ with the constraint $\|\boldsymbol{A}\boldsymbol{z}\| = 1$, where $\boldsymbol{A} \in \mathbb{R}^{D \times d}$ is the projection matrix used by REMBO. We minimize the Rosenbrock, Ackley, and product-of-sines functions

$$f_{\text{product-of-sines}}(\boldsymbol{x}) = 100 \sin(x_1) \prod_{i=1}^{d} \sin(x_i),$$

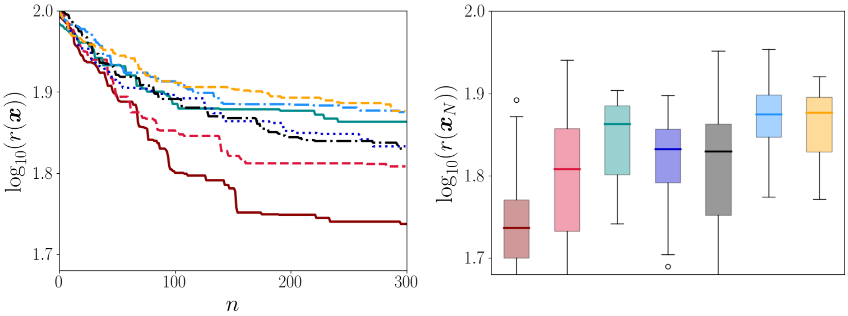
defined on the low-dimensional manifold $\mathcal{S}^5$ embedded in $\mathcal{S}^{50}$, i.e., $d = 5$ and $D = 50$. Figure 8.5 displays the median of the logarithm of the simple regret along 300 BO iterations (*left* graphs), and the median, with corresponding first and third quartiles after 300 iterations (*right* graphs) for the three aforementioned benchmark functions. We observe that HD-GaBO generally converges fast and provides good optimizers for all the test cases. Moreover, it outperforms all the other BO methods for the product-of-sines function: it provides fast convergence and better optimizer with low variance. In contrast, SIR-BO, which leads to the best optimizer for the Rosenbrock function, performs poorly to optimize the product-of-sines function. Similarly, dropout achieves a similar performance as HD-GaBO for the Ackley function, but it is outperformed by HD-GaBO in the two other test cases. Moreover, it is worth noticing that GaBO converges faster to the best optimizer than the other approaches for the Ackley function and performs better than all the geometry-unaware approaches for the product-of-sines function. This highlights the importance of using geometry-aware approaches for optimizing objective functions lying on Riemannian manifolds.



(a) Rosenbrock



(b) Ackley



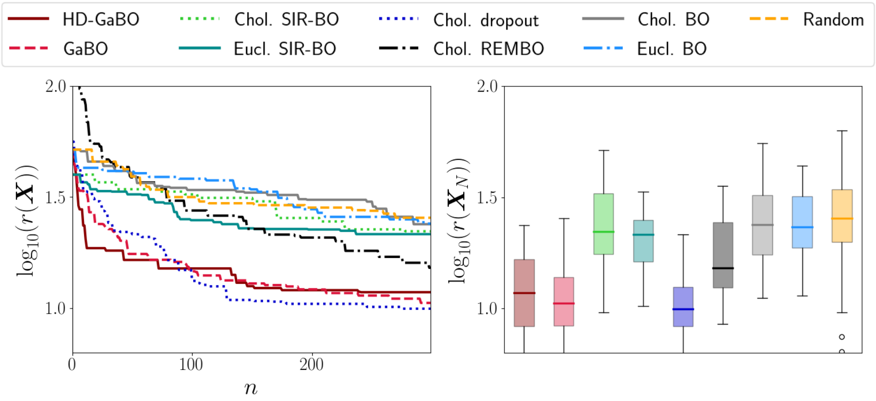
(c) Product of sines

Figure 8.5 – Logarithm of the simple regret for benchmark test functions in the nested sphere manifold $\mathcal{S}^5$ of $\mathcal{S}^{50}$ over 30 trials. The *left* graphs show the evolution of the median for the six BO approaches and the random search baseline. The *right* graphs display the distribution of the logarithm of the simple regret of the BO recommendation $\boldsymbol{x}_N$ after 300 iterations. The boxes extend from the first to the third quartiles and the median is represented by a horizontal line.

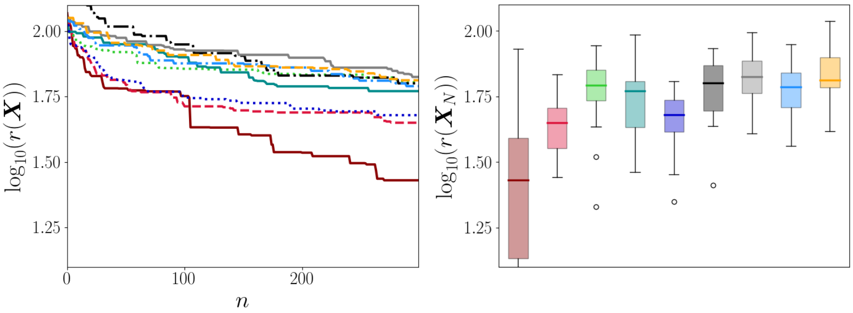
Regarding the SPD manifold $\mathcal{S}^D_{++}$, we compare HD-GaBO against GaBO, the Euclidean BO and SIR-BO, and the Cholesky-decomposition-based alternative implementations of BO, dropout, SIR-BO and REMBO. Similarly to dropout, we do not consider the Euclidean version of REMBO due to the difficulty of optimizing the acquisition function in the latent space while satisfying the constraint $\lambda_{\min} > 0$ for the query point in the high-dimensional manifold. We minimize the Rosenbrock, Styblinski-Tang ($s = 5$), and product-of-sines functions defined on the low-dimensional manifold $\mathcal{S}^3_{++}$ embedded in $\mathcal{S}^{10}_{++}$, i.e., $d = 3$ and $D = 10$. Figure 8.6 displays the median of the logarithm of the simple regret along 300 BO iterations (*left* graphs) and the median of the logarithm of the simple regret of the final recommendation $\boldsymbol{X}_N$ after 300 iterations, with corresponding first and third quartiles (*right* graphs) for the three aforementioned benchmark test functions. We observe that HD-GaBO consistently converges fast and provides good optimizers for all the test cases. Moreover, it outperforms all the other approaches for the Styblinski-Tang function, with faster convergence and a better optimizer. Similarly to the sphere cases, some methods are still competitive with respect to HD-GaBO for some of the test functions but perform poorly in other cases. For example, Cholesky dropout shows a similar convergence as HD-GaBO for the Rosenbrock function, but its median optimizer remains close to the initial values for the product-of-sines function. Inversely, Cholesky SIR-BO performs well for the product-of-sines function, but poorly for the Rosenbrock function. Furthermore, Cholesky BO converges fast to a good optimizer for the product-of-sines function, but its median optimizers do not differ much from the initial values for the other functions. Interestingly, GaBO performs well for both Rosenbrock and Styblinski-Tang functions. Moreover, the Euclidean BO methods generally perform poorly compared to their Cholesky equivalences, suggesting that, although they do not account for the manifold geometry, Cholesky-based approaches provide a better representation of the SPD parameter space than the Euclidean methods.

8.4 Conclusion

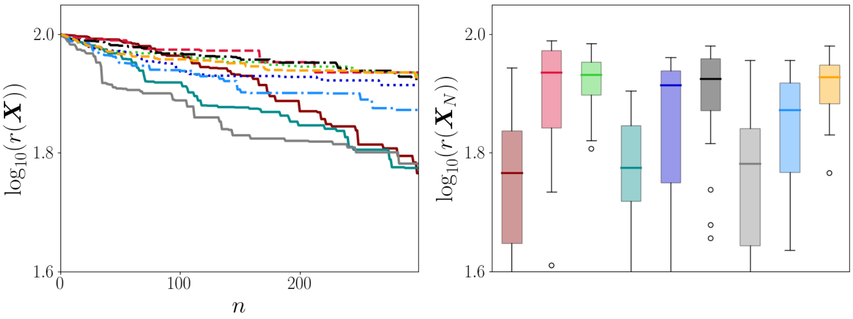
In this chapter, we evaluated the scalability of GaBO for seeking optimal parameters for functions lying on high-dimensional Riemannian manifolds. Moreover, we proposed HD-GaBO, a high-dimensional geometry-aware Bayesian optimization framework that exploits nested structure-preserving mappings to optimize high-dimensional functions that lie on a low-dimensional latent space. To do so, we used a geometry-aware GP that jointly learned a nested mapping and a representation of the objective function in the latent space. We also considered the geometry of the latent space while optimizing the acquisition function and we took advantage of the nested mappings to determine the next query point in the high-dimensional parameter space. Overall, our results highlight the relevance of geometry-awareness to scale BO to parameter spaces lying on high-dimensional Riemannian manifolds. GaBO was shown to be competitive with high-dimensional BO methods on the sphere manifold and outperformed all the other approaches on the SPD



(a) Rosenbrock



(b) Styblinski-Tang



(c) Product of sines

Figure 8.6 – Logarithm of the simple regret for benchmark test functions in the nested SPD manifold $\mathcal{S}^3_{++}$ of $\mathcal{S}^{10}_{++}$ over 30 trials. The *left* graphs show the evolution of the median for the eight BO approaches and the random search baseline. The *right* graphs display the distribution of the logarithm of the simple regret of the BO recommendation $\boldsymbol{x}_N$ after 300 iterations. The boxes extend from the first to the third quartiles and the median is represented by a horizontal line.

manifold. In the case of high-dimensional functions with underlying low dimensionality, HD-GaBO consistently performed well while optimizing various objective functions, unlike state-of-the-art methods that are geometry-unaware.

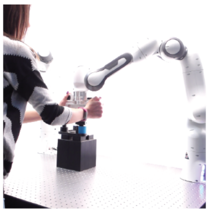
A limitation of the proposed HD-GaBO framework is that it depends on nested mappings that are specific to each Riemannian manifold. Therefore such mappings may not be available for all the existing manifolds. Future work will explore how to overcome this limitation. Moreover, we will evaluate the performance of HD-GaBO for optimizing robotic parameters in complex learning scenarios, where geometry-awareness may be relevant and objective functions may be better represented in an underlying latent space. Potential scenarios include the optimization of manipulability ellipsoids in $\mathcal{S}_{++}^6$ and the optimization of stiffness and damping matrices for controlling both the position and the orientation of the end-effector. From a broader point of view, HD-GaBO may also be exploited to optimize unit-norm parameters and SPD descriptors in machine learning applications, as well as covariance matrices of multivariate distributions.

Leveraging Structure-awareness in Robot Learning

Part IV

9 A Model-based Gaussian Process

In the Parts I, II and III of this thesis, we proposed to introduce geometry-awareness into learning, control and optimization approaches in order to handle parameters lying on Riemannian manifolds. We showed that adding domain knowledge in the form of geometry-awareness into these different methods enhances robot learning, control and optimization capabilities.



In the last Part of this thesis, we hypothesize that learning approaches can be enhanced with the introduction of domain knowledge not only through geometry-awareness, but also through structure-awareness. In the next chapter, structure-awareness is considered with the incorporation of *prior* models into learning approaches, with the presentation of a novel *model-based* Gaussian process approach.

Publication note

The material presented in this chapter is adapted from the following publication:

- Jaquier, N., Ginsbourger, D., and Calinon, S. (2019a). Learning from demonstration with model-based Gaussian process. In *Conference on Robot Learning (CoRL)*.

Supplementary material

A video illustrating the experiment presented in this chapter is available at:
https://sites.google.com/view/gmr-based-gp.

Source code

Source codes related to this chapter are available at:
https://github.com/NoemieJaquier/GMRbasedGP.

9.1 Introduction

In the previous chapters of this thesis, we provided different learning, control and optimization frameworks with additional domain knowledge by including information about the geometry of the data. This allowed us to design geometry-aware approaches, adapted to variables lying on Riemannian manifolds. An other way to introduce domain knowledge into learning methods is through *prior* models. In this chapter, we propose to introduce a prior model into a non-parametric approach, namely a Gaussian process. By doing so, the resulting approach possesses the advantage of the Gaussian process, including data efficiency and the various advantages of generative models, combined with interesting properties of the prior model, as explained in the following.

In the context of learning from demonstrations (LfD), robot motions can be generated from demonstrated trajectories using various probabilistic methods, e.g. Gaussian mixture regression (GMR) [Calinon, 2016], dynamical movement primitives (DMP) [Pastor et al., 2009], probabilistic movement primitives (ProMP) [Paraschos et al., 2013] kernelized movement primitives (KMP) [Huang et al., 2019] or Gaussian process regression (GPR) [Schneider and Ertel, 2010]. The covariance matrices of the prediction distributions computed by GMR, ProMPs and KMP encode the variability of the predicted trajectory. This variability, reflecting the dispersion of the data collected during the demonstrations, carries important information for the execution of the task. For example, the phases of the task in which a high precision is required, e.g. picking an object in a specific location, are characterized with a low variability, and vice-versa. During the reproduction, the variability is typically used to define robot tracking precision gains and permits the combination of different controllers [Silvério et al., 2018]. However, the approaches encoding variability do not take into account the availability of data in the different phases of the task. Inversely, the covariance matrices of the prediction distribution of GPs correspond to the prediction uncertainty, which reflects the presence or absence of training data in different phases of the task. This uncertainty measurement has been used, for example, to modulate the behavior of the robot far from the training data [Silvério et al., 2018] or to actively make requests for new demonstrations in unexplored regions of the input space [Maeda et al., 2017].

In LfD, it is often desirable to precisely refine parts of the demonstrated trajectories (e.g. due to changes in the environment), while maintaining the general trajectory shape (mean and variability) as in the demonstrations. It is also desirable to adapt the behavior of the robot, e.g. its compliance at different phases of the tasks, as a function of the variability of the demonstrations or the presence of (un)certainty in the reproduction. As none of the aforementioned methods provide both information features simultaneously, several approaches have been developed to take into account prediction uncertainty and variability. In [Schneider and Ertel, 2010], the reproduced trajectories are computed as a product of the predictions of local GPs, obtained by clustering the input space similarly to the approach of [Nguyen-Tuong et al., 2009]. Therefore, by adapting the parameters of each GP, the resulting uncertainty is adapted as a function of the variability of the different phases of the demonstrations. In [Umlauft et al., 2017], the prediction uncertainty and variability are inferred separately. The trajectories are predicted using a combination of GP and dynamical movement primitives (DMP), therefore providing uncertainty measurement. On the other hand, the variability in the reproduction is determined by inferring the components of the corresponding covariance matrix with GPs.

In this chapter, we propose an approach that aims at encapsulating the variability information of the demonstrations in the prediction uncertainty. We take inspiration from multi-output Gaussian processes (MOGPs) under the linear model of coregionalization (LMC) assumption to design a non-stationary, multi-output kernel based on GMR. In contrast with the aforementioned approaches, both variability and uncertainty information are encoded in a single GP. Moreover, we define the prior mean of the process as equal to the mean provided by GMR. This permits to ignore the training data in the generation of new trajectories and to consider only via-points constraints as observed data, therefore alleviating the computational cost of the GP. Moreover, setting the tracking precision as a function of the retrieved covariance allows us to demand the robot to precisely track the via-points while lowering the required tracking precision in regions of high variability.

The remainder of the chapter is organized as follows. While GMR and GPs were introduced in the Background chapter (see Section 2.4), Section 9.2 provides a short introduction to MOGPs. The proposed GMR-based GP is introduced in Section 9.3 and validated in a real-robot experiment in Section 9.4. Finally, Section 9.5 presents a discussion on similarities and differences of the proposed approach compared to other probabilistic methods, notably ProMP and KMP.

9.2 Background: Multi-output Gaussian Processes

Multi-output Gaussian processes (MOGPs) generalize GPs to vector-valued output by predicting jointly the output components (see [Alvarez et al., 2012] for a review). Therefore, MOGPs exploit the potential relation between the output components, which are not taken into account if predictions are computed separately for each dimension. Similarly to standard GP, the vector-valued objective function is modeled in MOGP with a vector-valued Gaussian process $\mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{K})$, inducing finite-dimensional prior distributions over potential objective functions, so that

$$\boldsymbol{f}(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x})), \tag{9.1}$$

with mean function $\boldsymbol{\mu} : \mathcal{X} \mapsto \mathbb{R}^D$ and positive-definite kernel (or covariance function) $\boldsymbol{K} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^{D \times D}$ with D the output dimension.

Given a set of N observed realizations $\{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$, the posterior distribution follows by Gaussian conditioning

$$\tilde{\boldsymbol{f}} \,|(\tilde{\boldsymbol{x}}), \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}^{\mathrm{P}}(\tilde{\boldsymbol{x}}), \hat{\boldsymbol{\Sigma}}^{\mathrm{P}}(\tilde{\boldsymbol{x}})), \tag{9.2}$$

with conditional mean and covariance functions given by

$$\hat{\boldsymbol{\mu}}^{\mathrm{P}}(\tilde{\boldsymbol{x}}) = \boldsymbol{\mu}(\tilde{\boldsymbol{x}}) + \boldsymbol{K}(\tilde{\boldsymbol{x}})^{\mathsf{T}} \left(\boldsymbol{K} + \boldsymbol{\Sigma}_{\varepsilon}\right)^{-1} (\boldsymbol{y} - \boldsymbol{\mu}), \tag{9.3}$$

$$\hat{\boldsymbol{\Sigma}}^{\mathrm{P}}(\tilde{\boldsymbol{x}}) = \boldsymbol{K}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}) - \boldsymbol{K}(\tilde{\boldsymbol{x}})^{\mathsf{T}} \left(\boldsymbol{K} + \boldsymbol{\Sigma}_{\varepsilon}\right)^{-1} \boldsymbol{K}(\tilde{\boldsymbol{x}}), \tag{9.4}$$

where $\boldsymbol{y}$ is the vector composed of the concatenated observations, $\boldsymbol{\mu}$ is the vector composed of concatenated means, $\boldsymbol{K}$ is the covariance matrix between the observations, i.e.,

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{pmatrix}, \ \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{\mu}(\boldsymbol{x}_N) \end{pmatrix} \ \text{and } \boldsymbol{K} = \begin{pmatrix} \boldsymbol{K}(\boldsymbol{x}_1, \boldsymbol{K}_1) & \dots & \boldsymbol{K}(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ \boldsymbol{K}(\boldsymbol{x}_N, \boldsymbol{K}_1) & \dots & \boldsymbol{K}(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}, \tag{9.5}$$

$\boldsymbol{K}(\tilde{\boldsymbol{x}})$ is a matrix of covariance terms between $\tilde{\boldsymbol{x}}$ and the observations $\boldsymbol{x}_n$, $\boldsymbol{\Sigma}_{\varepsilon}$ is the covariance matrix of the observation noise $\boldsymbol{\varepsilon}$, which is assumed centered Gaussian and independent of the process. The covariance $\hat{\boldsymbol{\Sigma}}^{\mathrm{P}}(\boldsymbol{x})$ expresses the prediction uncertainty for all components and between them. In typical cases, the further away the input data lies from the training dataset the larger the prediction variance.

The class of covariance kernels that we consider in this chapter is formulated as a sum of separable kernel functions generated under the linear model of coregionalization (LMC) assumption [Goovaerts, 1997]. This class of kernel functions is often called separable as the dependencies between inputs and outputs are decoupled. Therefore, the kernel $\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ between two input vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is expressed as

$$\boldsymbol{K}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \sum_{q=1}^{Q} \boldsymbol{\Upsilon}_q k_q \left(\boldsymbol{x}, \boldsymbol{x}'\right), \tag{9.6}$$

where the so-called coregionalization matrices $\boldsymbol{\Upsilon}_q \in \mathbb{R}^{D \times D}$ are positive semi-definite matrices representing the interaction among the output components. The choice of the scalar-valued kernels k_q and the design of the coregionalization matrices $\boldsymbol{\Upsilon}_q$ are crucial for the GP as they represent our prior knowledge about the function that is being learned.

9.3 A Model-based Gaussian Process

In this section, we propose to combine GMR and GPR to form a GMR-based GP. The proposed approach takes advantage of the ability of GPs to encode various prior beliefs through the mean and kernel functions and allows the variability information retrieved by GMR to be encapsulated in the uncertainty estimated by the GP. Moreover, the proposed approach enjoys the properties of generative models, therefore new trajectories can be easily generated through sampling and conditioning.

9.3.1 GMR-based GPs Formulation

In order to avoid confusion, we denote in this chapter the mean of the GMR distribution (2.43) as $\hat{\boldsymbol{\mu}}^{\mathrm{M}}$. Moreover, the componentwise covariances (2.46) are denoted as $\hat{\boldsymbol{\Sigma}}_\ell$ for simplicity. We define the GMR-based GP as a GP with prior mean

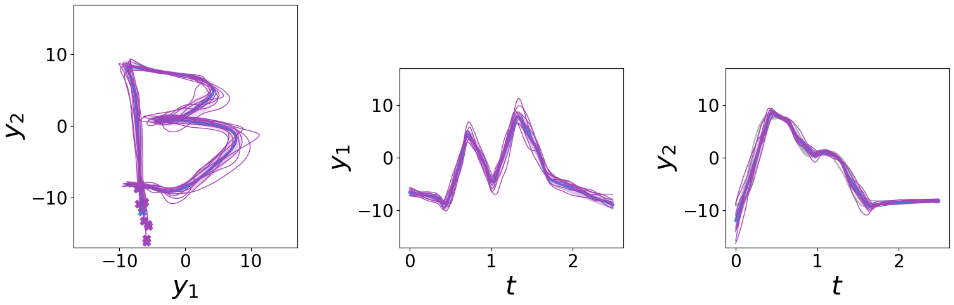
$$\boldsymbol{\mu}(\boldsymbol{x}) = \hat{\boldsymbol{\mu}}^{\mathrm{M}}(\boldsymbol{x}), \tag{9.7}$$

and a kernel in the form of a sum of C separable kernels associated with the C components of the considered GMM

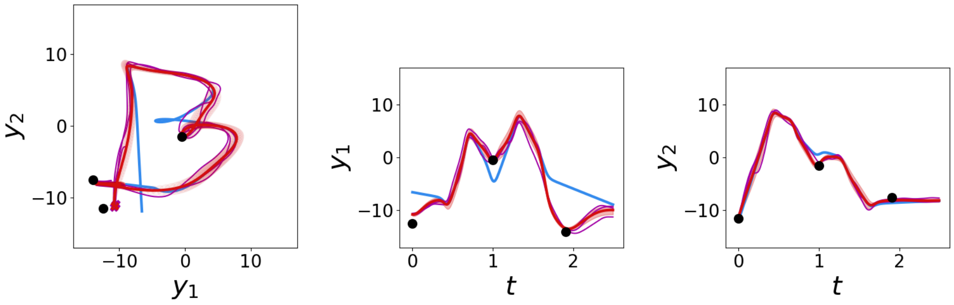
$$\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\ell=1}^{C} h_\ell(\boldsymbol{x}_i) h_\ell(\boldsymbol{x}_j) \hat{\boldsymbol{\Sigma}}_\ell \, k_\ell(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{9.8}$$

The prior mean of Eq. (2.43) allows the GP to follow the GMR predictions far from training data. Moreover, the constructed GP is also covariance non-stationary due to its spatially-varying coregionalization matrices [Gelfand et al., 2004]. The input-dependent coregionalization matrices $h_\ell(\boldsymbol{x}_i) h_\ell(\boldsymbol{x}_j) \hat{\boldsymbol{\Sigma}}_\ell$ corresponding to this conception are determined by GMR (via (2.46), (2.47)). Alternatively, one can say that the GMR responsabilities h_ℓ weight the importance of the kernels $k_\ell(\boldsymbol{x}_i, \boldsymbol{x}_j)$ according to the proximity of the input data to the center of GMM components. Thus, the kernels associated to the centers close to the given input data are more relevant than distant centers. The covariance matrices $\hat{\boldsymbol{\Sigma}}_\ell$ allows the dependencies between the output data to be described for each GMM component. Note that both the coregionalization matrices and the number of separable kernels are determined by GMR. Therefore, the only parameters to determine are the hyperparameters of the kernels k_ℓ which can be estimated, for example, by maximizing the likelihood of the GP. Moreover, the variance parameters σ_f of the kernels k_ℓ are fixed to 1 as they are already scaled by the covariance matrices $\hat{\boldsymbol{\Sigma}}_\ell$. Thus, the estimation of hyperparameters is simplified compared to standard LMC.

Figure 9.1a shows the prior mean and 10 sample trajectories generated from the proposed GMR-based GP where k_ℓ are Matérn kernels ($\nu = 5/2$). The training data, consisting of 5 demonstrations of a two-dimensional time-driven trajectory, are identical to the training data of Figures 2.4 and 2.5. The GMR-based GP is based on a 6-component GMM (see Figure 2.4a). The hyperparameters of the model, namely the lengthscales $\sigma_{l,\ell}$ of the k_ℓ and the covariance of the noise $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} = \sigma_\epsilon \boldsymbol{I}$, were optimized by maximum likelihood estimation within the GPy framework [GPy, 2012]. Note that the prior mean of the process corresponds to the mean obtained by GMR in Figure 2.4b. Moreover, the prior uncertainty provided by the GMR-based GP is lower in the regions where the



(a) Prior trajectories



(b) Posterior trajectories

Figure 9.1 – Illustration of GMR-based GP on a two-dimensional time-driven trajectory. (a) Sample trajectories generated from GMR-based GP. The prior mean of the process and the sample trajectories are represented by continuous blue and purple lines, respectively. The covariance $\boldsymbol{K}_{t^{(\mathrm{o})}_{1:N}}$ of the process is represented by a light purple tube around the prior mean. (b) Sample and predicted trajectories generated from the posterior distribution of the GMR-based GP. The prior, sampled and predicted trajectories are represented by continuous blue, dark pink and red lines, respectively. The uncertainty of the prediction is represented by a light red tube around the predicted mean. Via-points, considered as observations for the GMR-based GP, are represented by black dots. The trajectories are extrapolated from training data for $t > 2$.

variability of the demonstrations is low, e.g. at the bottom of the straight vertical line of the B letter, and higher in the regions of higher variability, e.g. in the curves in the right part of the B.

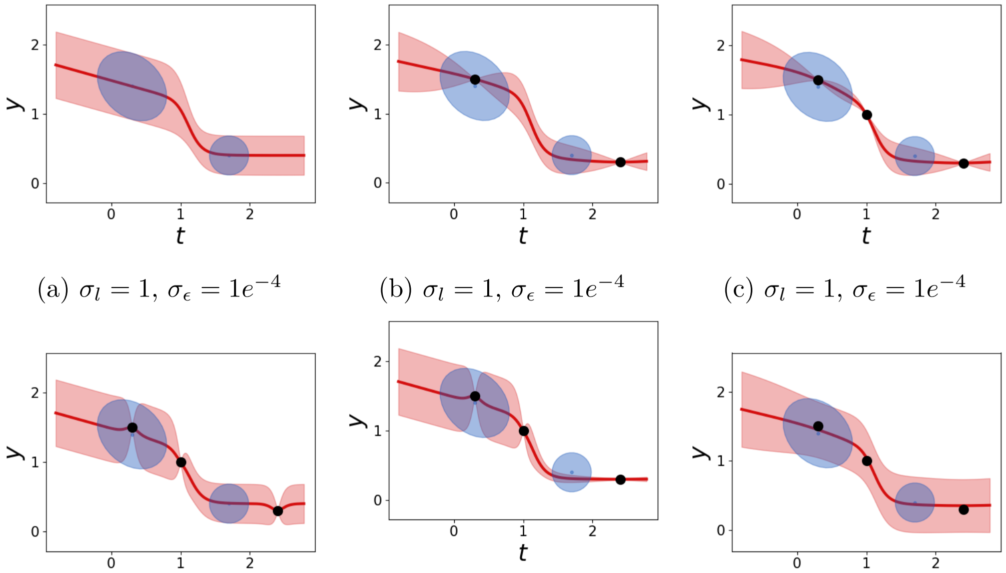
9.3.2 GMR-based GPs Properties

A particularity of the presented GMR-based GP is that the information on the demonstrations distribution is included in the prior mean $\boldsymbol{\mu}(\boldsymbol{x})$ and covariance $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x})$ after determining the hyperparameters. Therefore, the training data can be ignored and our model can be conditioned uniquely on new observations. Figure 9.2a shows the mean and uncertainty recovered by a 1D-output GMR-based GP without any observation. The process was constructed based on a GMM with two components with k_ℓ defined as Matérn kernels ($\nu = 5/2$). The lengthscale parameters σ_l of the k_ℓ and the covariance of the noise of the process σ_ϵ are fixed as equal to 1 and $1e^{-4}$, respectively. Note that the distribution corresponds to the prior of the GMR-based GP, therefore the mean is exactly equal to the mean computed by GMR. Moreover, if a component ℓ is entirely responsible for a test datapoint so that $h_\ell = 1$, the corresponding uncertainty is equal to the conditional covariance of the component $\hat{\boldsymbol{\Sigma}}_\ell$ augmented with σ_ϵ, as observed for $t < 0.6$ and $t > 1.8$ in Figure 9.2a. In the case where several components are responsible for the datapoint, its uncertainty is a weighted sum of the conditional covariances, as observed for the zone in between the two GMM components. Therefore, the prior uncertainty obtained by GMR-based GP without observation reflects the variability provided by GMR. However, note that the prior uncertainty of GMR-based GP is not equal to $\hat{\boldsymbol{\Sigma}}^{\mathrm{M}}$.

In the cases where it is desirable to adapt trajectories towards new start-, via- or end-points $(\boldsymbol{\xi}_v, \boldsymbol{\zeta}_v)$, those particular points are used to define a new set of V observation with inputs and outputs $\{(\boldsymbol{x}_v, \boldsymbol{y}_v)\}_{v=1}^{V} = \{(\boldsymbol{\xi}_v, \boldsymbol{\zeta}_v)\}_{v=1}^{V}$ which is then used to infer the posterior distribution of the GMR-based GP with (9.2). Figures 9.2b and 9.2c show examples where 2 and 3, via-points were added to the trajectory. We observe that the mean of the process goes through the via-points and the uncertainty becomes very small in these locations. Note that conditioning a trajectory towards via-points with GMR alone is not straightforward due to the fact that covariance terms between two datapoints are equal to zero.

As in a standard GP, the predicted mean and uncertainty depend strongly on the kernel parameters. In Figure 9.2d, the lengthscale parameters of the kernels k_ℓ have been reduced to $\sigma_l = 0.1$. We observe that the uncertainty remains low in a small neighborhood around the via-points but increases rapidly further from them. Moreover, one of the advantages of the GMR-based GP is that each kernel k_ℓ can be chosen individually and their parameters are determined separately. Therefore, different behaviors can be obtained as a function of the location in the input space, as shown by Figure 9.2e where the lengthscale parameters of the kernels corresponding to the left and right GMM component are equal to 0.1 and 5, respectively. Similarly to a standard GP, the noise of the process determines the behavior of the GMR-based GP at the via-points location. As shown by Figure 9.2f where $\sigma_\epsilon = 0.1$, the constraint of passing exactly through the via-points is alleviated and the mean of the GMR-based GP passes close-by the via-points while the uncertainty is equal to the noise of the process. Note that the noise parameter can also be defined separately for each kernel k_ℓ.

Figure 9.1b shows the predicted mean and corresponding uncertainty as well as three trajectories sampled from the posterior distribution of the GMR-based GP on the B trajectory. As explained previously, the initial demonstrations data were used for hyperparameters estimation but not incorporated as conditioning data and three via-points have been added as new observations. We observe that the estimate and the



(a) $\sigma_l = 1$, $\sigma_\epsilon = 1e^{-4}$

(b) $\sigma_l = 1$, $\sigma_\epsilon = 1e^{-4}$

(c) $\sigma_l = 1$, $\sigma_\epsilon = 1e^{-4}$

(d) $\sigma_l = 0.1$, $\sigma_\epsilon = 1e^{-4}$

(e) $\sigma_{l_{left}} = 0.1$, $\sigma_{l_{right}} = 5$, $\sigma_\epsilon = 1e^{-4}$

(f) $\sigma_l = 1$, $\sigma_\epsilon = 0.1$

Figure 9.2 – Illustration of the properties of GMR-based GP on time-driven 1D-trajectories. The trajectories are predicted by GMR-based GP. The process was constructed based on a 2-component GMM, represented by blue ellipses. The estimate of GMR-based GPR is shown by a red continuous line with the corresponding uncertainty represented by a light tube around the mean. The initial observations are discarded after determining the hyperparameters. The lenghtscale parameters and noise covariance are fixed as $\sigma_l = 1$ and $\sigma_\epsilon = 1e^{-4}$, respectively. (a) The posterior distribution without any new observation is represented. $(b$-$c)$ Two, respectively 3, via-points, represented with black dots, are added as observations of the GMR-based GP. (d) The lengthscale parameters are changed to $\sigma_l = 0.1$, therefore modifying the evolution of the uncertainty around the observations compared to b-c. (e) The lengthscale parameters are fixed as $\sigma_l = 0.1$, $\sigma_l = 5$ for the left and right GMM component, respectively. (f) The noise covariance of the process is fixed as $\sigma_\epsilon = 0.1$ $(\sigma_l = 1)$.

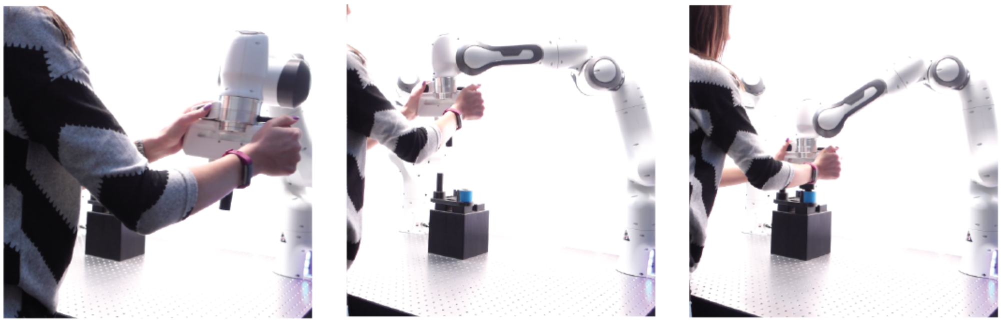


Figure 9.3 – Demonstrations of the peg-in-hole task are provided via kinesthetic teaching. The user guides the robot, which holds the peg in its gripper, to approach a hollow cylinder, position the peg above the hole to prepare the insertion and insert the peg. The blue hollow cylinder is placed 20 cm above the table for the demonstrations.

posterior trajectories are adapted to pass close to the via-points. In the zones far from the via-points, the predicted trajectory follows the prior mean of the process.

9.4 Experiments

In this section, we evaluate the proposed approach in a peg-in-hole task achieved by the 7-DoF Franka Emika Panda robot. In the first part of the experiment, 3 demonstrations of the task were collected by kinesthetically guiding the robot to first approach a hollow cylinder and then insert the peg in it (see Figure 9.3). For all the demonstrations, the hollow cylinder was placed 20 cm above the table. The collected data, encoding time t and Cartesian position $(y_1, y_2, y_3)^\mathsf{T}$, were time-aligned. We trained a GMM and determined the hyperparameters of a GMR-based GP, as well as a MOGP with the separable kernel (9.6) ($Q = C$) based on the time-driven demonstrated trajectories. Matérn kernels ($\nu = 5/2$) were chosen as individual kernels k_ℓ and k_q for the GMR-based GP and MOGP, respectively. The number of components of the GMM ($C = 4$) was selected by the experimenter. Figure 9.4a shows the demonstrated trajectories and corresponding GMM states.

In the second part of the experiment, an obstacle was added in between the initial position of the robot and the hollow cylinder. Moreover, the hollow cylinder was positioned directly on the table, i.e. 20 cm below its location during the demonstrations. Via-points were determined by the experimenter to modulate trajectories so that the robot avoids the obstacle in the desired manner and its final position corresponds to the new location of the hollow cylinder. The performances of the proposed GMR-based GP, the MOGP and GMR to reproduce the task in the modified environment were compared.

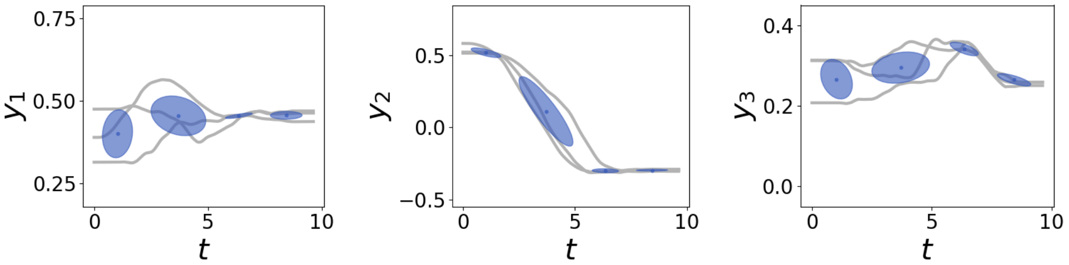
As explained in the previous section, the via-points were used to define a new set of observations for the GMR-based GP, while the original training data are discarded after inferring the hyperparameters. In the case of the MOGP, the mean and uncertainty of the reproduction considering V via-points are updated for each testing input $\boldsymbol{x}$ by conditioning the distribution (9.2) on the desired via-points. The mean and variability of the reproduction obtained by GMR can be updated in a similar way. However, as the covariances between different datapoints are not encoded in GMR, the generated trajectory is discontinuous. Therefore, we did not reproduce it with the robot and we show instead in the following graphs the GMR reproduction where no via-points are considered, whose mean corresponds to the prior mean of the GMR-based GP.

The task was reproduced using a linear quadratic regulator (LQR) controller tracking the trajectory predicted by GMR-based GPR, MOGP or GMR [Calinon, 2016]. The required tracking precision was set as proportional to the inverse of the posterior covariance $\hat{\boldsymbol{\Sigma}}$ of the different methods. This information is exploited to demand a high precision tracking in the regions of the trajectories where high certainty (GMR-based GP and MOGP) or low variability (GMR) are observed, and vice-versa.

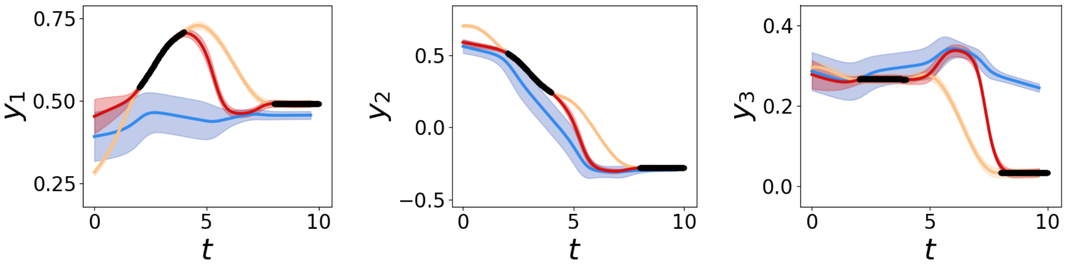
Figure 9.5 shows snapshots of the robot reproducing the peg-in-hole task using the proposed GMR-based GP (*top row*) and the MOGP (*bottom row*). We observe that the robot is able to circumvent the obstacle with both methods. However, the peg insertion fails when the MOGP is used, as the peg is located in front of the cylinder at the end of the trajectory. This is due to the fact that the trajectory generated by the MOGP straightly links the two zones characterized with via-points, while the GMR-based GP trajectory tends to follow the prior mean defined by GMR in between the two zones, as shown in Figure 9.4b-*bottom*. This behavior allows the robot to position the peg above the hole before approaching the cylinder and perform the insertion as demonstrated in the first phase of the experiment. Inversely, by using the MOGP, the robot approaches the hollow cylinder from the side, and therefore is unable to insert the peg. These different behaviors are illustrated in Figure 9.6, where the 3D trajectories reproduced with the different methods are represented. In order for the MOGP to successfully reproduce the insertion, a supplementary via-point could be added prior to the insertion. However, this supplementary via-point is not needed by the GMR-based GP thanks to its prior mean.

Moreover, as shown in Figure 9.4b-*bottom*, the uncertainty computed by the MOGP is low along the whole trajectory, resulting in a rigid behavior of the robot for the whole reproduction. In contrast, the GMR-based GP ensures a high tracking precision in the two parts of the trajectory characterized by the via-points, while the robot can be more compliant elsewhere depending on the variability encoded by the GMR, notably at the beginning of the reproduced trajectory.

Table 9.1 gives the computation time for one test data with the training data of the 3 demonstrations with a non-optimized Python code on a laptop with 2.7GHz CPU and 32 GB of RAM. We observe that GMR-based GP is slightly more computationally expensive than GMR and MOGP. However, this remains reasonable for our real-time experiment.

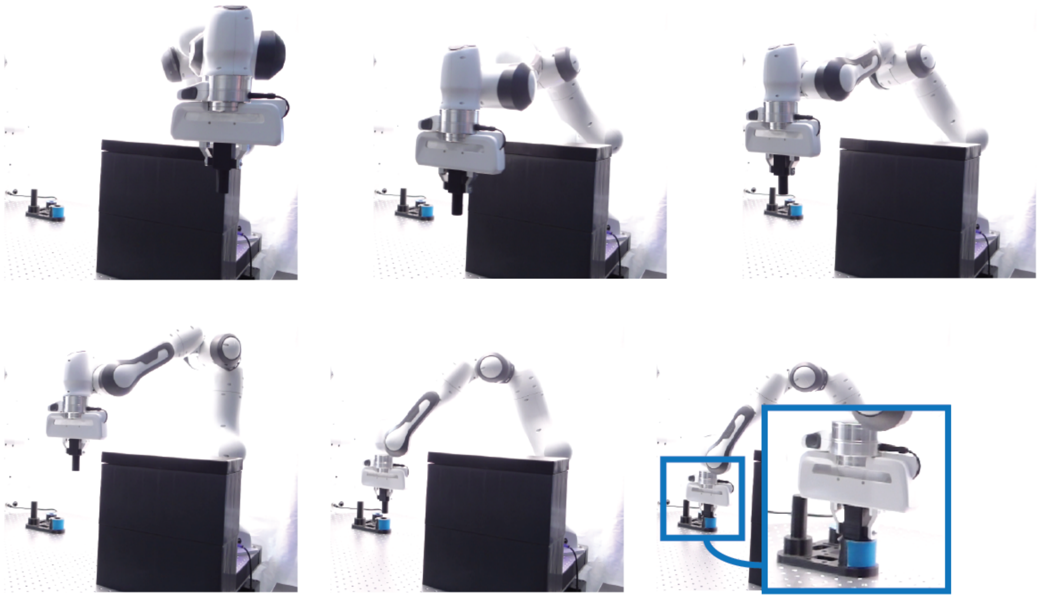


(a) Demonstrations and learned GMM

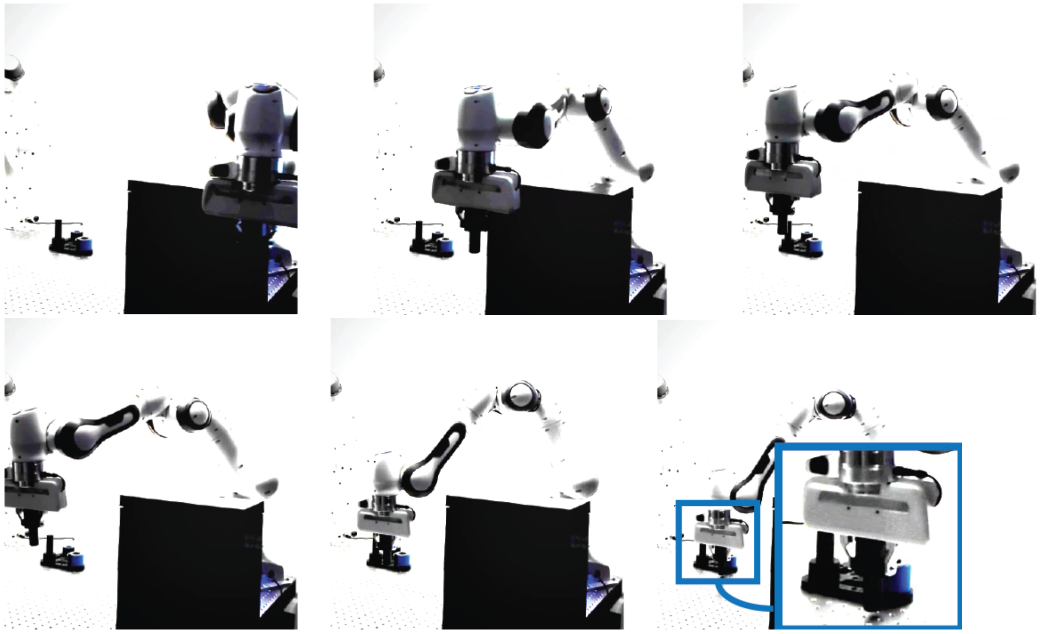


(b) Reproduction with GMR-based GP, MOGP and GMR

Figure 9.4 – Demonstrations and reproductions of the peg-in-hole task. (a) Demonstrated trajectories (in light gray) and corresponding GMM ($C = 4$) represented as blue ellipses. The Cartesian positions y_1, y_2, y_3, considered as outputs, are represented as a function of the time, considered as input. (b) Reproduced trajectories. The means of the trajectories generated by GMR-based GP, MOGP and GMR are represented by red, yellow and blue lines, respectively, with their respective covariance depicted by light tubes around the estimates. The via-points defined to modulate the trajectories generated by GMR-based GP and MOGP are depicted with black dots.



(a) Reproduction with GMR-based GP



(b) Reproduction with MOGP

Figure 9.5 – Snapshots of the robot reproducing the peg-in-hole task using GMR-based GP (*top row*) and MOGP (*bottom row*). Both methods allow the robot to circumvent the obstacle. However, the peg is successfully inserted in the hole with the GMR-based GP, while the robot fails to insert the peg with MOGP (observe that the blue hollow cylinder is behind the peg at the end of the trajectory).

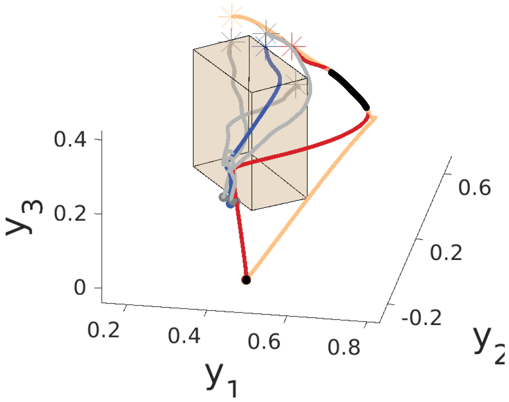


Figure 9.6 – 3D representation of the trajectories followed by the robot in the reproduction of the peg-in-hole task. The demonstrations are depicted in light gray. The trajectories reproduced with GMR-based GP, MOGP and GMR are represented in red, yellow and blue, respectively. The beginnings of the demonstrated and reproduced trajectories are depicted by stars. The obstacle added in the reproduction phase is represented as a brown parallelepiped. While both GMR-based GP and MOGP trajectories allow the robot to circumvent the obstacle, the insertion of the peg in the hollow cylinder fails in the MOGP case as the robot approaches the hollow cylinder from its side.

Table 9.1 – Computation time of GMR, MOGP and GMR-based GP for one test data in the real robot experiment. All the time values are given in milliseconds [ms].

GMR	MOGP	GMR-based GP
1 ± 0.1	4 ± 0.6	13 ± 1

9.5 Discussion

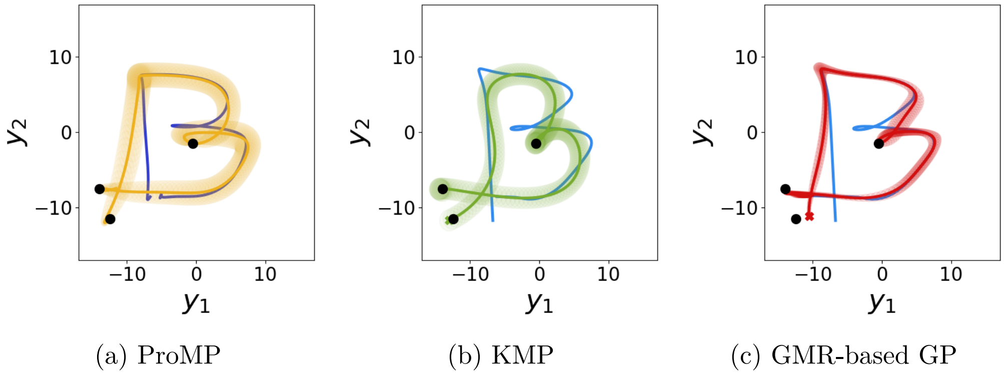
By defining a prior mean as GMR and by encapsulating the variability of the demonstration in its uncertainty, the proposed GMR-based GP allows efficient reproductions of tasks learned by demonstration while adapting the learned trajectories towards new start-, via- or end-points. We discuss here similarities and differences between the proposed approach and other algorithms widely used to learn trajectories.

As briefly mentioned in the previous section, adapting trajectories with GMR is difficult as conditioning on via-points results in discontinuous trajectories and re-optimizing the underlying GMM to fulfill via-points constraints is not straightforward. In contrast, the trajectories can be easily adapted to go through start-, via- or end-points by conditioning on the desired observations in the case of GP and ProMP. Trajectories can also be adapted using DMP. However, DMP does not handle variation and uncertainty information. Moreover, as DMP and ProMP encode trajectories by relying on basis functions equally spaced in time, selecting appropriate basis functions becomes difficult with high-dimensional inputs. In contrast, kernel methods and GMR, for which GMM learns the organization of basis functions, generalize well to high-dimensional inputs. By using GP, the trajectories are modeled without considering the correlations between the output components. This problem can be alleviated by replacing GP by MOGP. Notice that the computational complexity of testing for the proposed GMR-based GP is importantly reduced compared to MOGP, as the set of observations used in the testing part is only composed of desired via-points, therefore resulting in a computational complexity of $\mathcal{O}(V^2D^2)$ instead of $\mathcal{O}(N^2D^2)$, with D the output dimension and $V \ll N$.

Figure 9.7 shows an example of modulated trajectories recovered by ProMP, KMP and GMR-based GP. The training data, consisting of 5 demonstrations of a two-dimensional time-driven trajectory, are identical to the training data of Figures 2.4, 2.5 and 9.1. As in Figure 9.1b, via-points are represented by black dots. ProMP is evaluated with 20 Gaussian basis functions, while both KMP and GMR-based GP are based on the 6-component GMM of Figure 2.4a. As expected the three methods are able to generate trajectories passing through the via-points.

Overall, KMP shares strong similarities with the proposed GMR-based GP. Both approaches are kernel-based and can therefore cope with high-dimensional inputs. Moreover, both make use of GMR, to retrieve a reference trajectory in the case of KMP and to define the prior mean as well as kernel parameters for GMR-based GP. The reference trajectory of KMP and the prior mean of GMR-based GP are depicted by a light blue line in the example of Figure 9.7. Notice that both are identical. Therefore, the correlations between the output components are taken into account in both models and they predict full covariances for inferred outputs. Note that both approaches can make use of other algorithms than GMR to capture the distribution of the demonstrations.

Compared to KMP which can be related to kernel regression, the framework of GMR-based GP allows the representation of more complex behaviors, notably by defining priors for the process. Our approach benefits of the properties of generative models, allowing sampling of new trajectories from prior and posterior models (as shown in Fig. 9.1b), and is highly flexible due to the kernels k_ℓ that can be chosen individually, resulting in different behaviors of the model in the different regions of the input space. Moreover, GMR-based GP provide an uncertainty information encapsulating the variability in the variance parameter of the kernel, while KMP introduces the measure of the variability of the demonstrations as the covariance matrix of the observation noise (see also Figure 9.2 compared to Figure 9.2). As a consequence, for the example of a robot tracking via-points, KMP will adapt the distribution according to the covariance, representing the variability of the demonstrations, given initially by GMR. In contrast, our approach allows us to set via-points that the robot is required to track precisely, where the covariance tends to zero due to GP properties. This is relevant for the case in which controller gains are set as a function of the observed covariance, as we can ensure high precision due to close-to-zero



(a) ProMP (b) KMP (c) GMR-based GP

Figure 9.7 – Comparison of the predicted trajectories generated by (a) ProMP, (b) KMP and (c) GMR-based GP with three via-points. The mean is represented by a continuous line and the variance by a light tube around the estimate. Via-points are represented by black dots. The mean of the trajectories recovered from the demonstrations (without via-points) are depicted by blue lines.

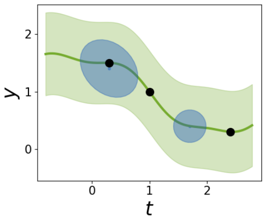


Figure 9.8 – Example of time-driven 1D-trajectory predicted by KMP with three via-points. The reference trajectory of KMP is based on the 2-component GMM of Fig. 9.2a, represented by blue ellipses. The lengthscale parameter of the Gaussian kernel is fixed as $\sigma_l = 1$. The via-points are represented with black dots.

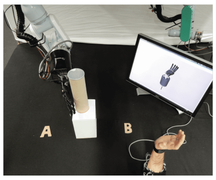
prediction variances. In general, the tracking precision can be set independently from the prior model with GMR-based GP (see Figures 9.2c, 9.2f).

9.6 Conclusion

This chapter presented a new class of multi-output GP with non-stationary prior mean and kernel based on GMR. Incorporating the GMR prior model into the GP allows the resulting approach to benefit from the properties of both GMR and GPs. Notably, our approach inherits of the properties of generative models and benefits of the expressiveness and versatility of GPs. Moreover, within this framework, the variability of the demonstrations is encapsulated in the prediction uncertainty of the designed GP. Correlations between the different output components are taken into account by the model. Moreover, the proposed method takes advantage of the prior model, obtained from the demonstrations, for trajectory modulation, considering only via-points constraints as observed data to generate new trajectories. Therefore, incorporating the given prior into the GP reduces its computational load by not requiring to store all datapoints. Moreover, by doing so, our framework allows a precise tracking of via-points while the compliance of the robot can be adapted as a function of the variability of the demonstrations in other parts of the trajectories.

Extensions of this work will investigate more in details the properties and limits of the proposed approach. Moreover, we plan thorough comparisons between GMR-based GP and other approaches of interest, such as KMP. Finally, the proposed approach may also be considered in an active learning framework, where new datapoints are queried in regions of high uncertainty.

10 Exploiting Tensor Structures in a Mixture of Experts



The Part IV of this thesis proposes to introduce domain knowledge into learning approaches in the form of structure-awareness. In the previous chapter, we presented a model-based Gaussian process, illustrating the benefits of incorporating structure-awareness into a non-parametric approach through prior models.

In this chapter, we focus on applications where a particular structure is intrinsically present in the data. More specifically, we introduce a mixture-of-experts model adapted to data organized in the form of *matrices* or *tensors*.

Publication note

The material presented in this chapter is adapted from the following publications:

- Jaquier, N., Connan, M., Castellini, C., and Calinon, S. (2017). Combining electromyography and tactile myography to improve hand and wrist activity detection in prostheses. *Technologies*, 5(4);
- Jaquier, N., Haschke, R., and Calinon, S. (2020a). Tensor-variate mixture of experts for proportional myographic control of a robotic hand. *Submitted to Robotics and Autonomous Systems, Available as arXiv:1902.11104*.

Supplementary material

A video related to this chapter is available at:
https://sites.google.com/view/tensor-mixture-of-experts.

Source code

Source codes related to this chapter are available at:
https://github.com/NoemieJaquier/TME.

The dataset related to the experiment presented in Section 10.4.3 is available at:
http://www.idiap.ch/paper/mdpi/data/exp2/.

10.1 Introduction

In this thesis, we formerly investigated two different approaches to include additional domain knowledge into learning algorithms. The main part of the thesis proposed to design geometry-aware approaches by introducing information about the geometry of the data into the models. Moreover, in the previous chapter, we proposed a novel model-based Gaussian process, showing that structure can also be incorporated into learning approaches in the form of prior models. In this chapter, we focus again on data properties. Motivated by the problem of recognizing hand movements from a matrix of tactile sensors, we propose to investigate models adapted to matrix and tensor data. We denote these approaches as *structure-aware*, in the sense that it takes the intrinsic structure of the data into account.

In the context of prosthetic hands, tactile myography (TMG) has recently been proposed as a complementary or alternative approach to the traditional surface electromyography (sEMG) to achieve simultaneous and proportional control of multiples degrees of freedom (DOFs) of a hand prosthesis (see, e.g., [Phillips and Craelius, 2005; Kõiva et al., 2015]). In this context, the aim of TMG is to measure the pressure related to the deformation induced by the muscles activity of the forearm. This signal is then used to determine the corresponding hand and wrist movements. The TMG sensor, developed in [Kõiva et al., 2015] and displayed in Figure 10.1, is composed of 320 resistive taxels organized in a 8 × 40 array forming a bracelet. Therefore, the data provided by the sensor are intrinsically matrix-valued. Previous studies showed that ridge regression (RR) directly applied to the data of the bracelet allows the prediction of different finger and wrist movements [Kõiva et al., 2015], which could outperform detection using sEMG [Nissler et al., 2017]. However, RR does not take into account the matrix structure of the TMG data as they are vectorized before the application of the regression method. Also, despite our data may contain patterns that could be treated by deep learning strategies such as convolutional neural networks, the use of such approaches would require large training datasets to be efficient, which does not fit with the requirement of our application, targeting personalized calibration of prosthetic hands from small datasets. Therefore, in this chapter, we hypothesize that taking the intrinsic matrix structure of the data into account in the regression process may lead to improved performance to recognize hand movements based on TMG data.

Tactile myography is not the only example where data collected in robotics are naturally represented as matrices or tensors. Examples also include images and video streams [Zhao et al., 2014], as well as electroencephalography (EEG) data [Miwakeichi et al., 2004; Washizawa et al., 2010]. Most approaches described in the literature consist of reorganizing the elements of these tensors into vectors before applying learning algorithms based on linear algebra operating on vector spaces. This flattening operation ignores the underlying structure of the original data. Moreover, the dimensionality of the resulting problem dramatically increases, creating high computational and memory requirements.

Finally, the number of model parameters to estimate in the learning method may become high, which constitutes an important issue in the cases where only few training data are available.

With the burst of multidimensional data available in various fields of research, important efforts were turned toward extending standard dimensionality reduction and learning techniques to tensor data. In this context, Zare et al. [2018] proposed a review of tensor decomposition methods by dividing them into three categories of problems usually targeted by principal component analysis (PCA), namely low-rank tensor approximation, low-rank tensor recovery, and feature extraction. In particular, multilinear PCA (MPCA) [Lu et al., 2008] and weighted MPCA [Washizawa et al., 2010] were proposed to extract features from tensor objects as a preprocessing step for classification. Similarly, linear discriminant analysis was extended to multilinear discriminant analysis in [Tao et al., 2007] and factor analysis was adapted to tensor data in [Tang et al., 2013]. In the context of regression, Guo et al. [2012] proposed generalizations of ridge regression (RR) and support-vector regression (SVR) methods to tensor data, where they showed the superiority of tensor-based algorithms over the vector-based algorithms in various applications. A similar extension of RR to tensor data was proposed in [Zhou et al., 2013] with an application in magnetic resonance imaging (MRI). Following a similar process, tensor-variate logistic regression (LR) was proposed in [Hung and Wang, 2013; Tan et al., 2013] for the classification of multidimensional data. Moreover, kernel-based frameworks such as Gaussian processes (GPs) [Zhao et al., 2014] were also adapted to tensors [Signoretto et al., 2011].

In this chapter, we introduce a tensor-variate-mixture-of-experts (TME) model for regression. Mixture of experts (ME) models, first introduced by Jacobs et al. [1991], combine the predictions of several experts based on their probability of being active in a given region of the input space. Each expert acts as a regression function, while a gate determines the regions of the input space where each expert is trustworthy. The output of the model is a weighted sum of the experts predictions. Over the years, ME models were widely improved with different gates, regression and classification models for the experts (see [Yuksel et al., 2012] for a review of applications). Notably, for wrist movements recognition based on electromyographic (EMG) signals, the ME model with linear experts can achieve similar performance as more complex nonlinear methods, at a lower computational cost [Hahne et al., 2014].

In order to handle tensor data in a ME model, we propose to use tensor-variate models for both the experts and the gates. In Section 10.3, we show that the experts can be defined as tensor linear models and that the gates can be set as tensor-variate softmax functions. Both elements are based on the inner product between the input tensor data and model parameters (see Section 10.2). The resulting TME model is trained with an EM algorithm using the CANDECOMP/PARAFAC (CP) decomposition, also called canonical polyadic decomposition [Carroll and Chang, 1970; Harshman, 1970].

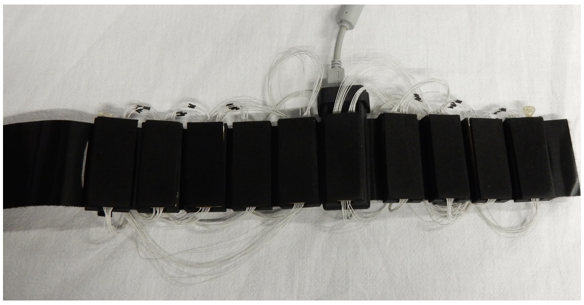


Figure 10.1 – TMG sensor used for the experiments. The bracelet is here rolled out, showing its 10 modules of 8×4 resistive cells.

The functionality of the proposed approach is first evaluated and compared to the corresponding vector-based approach using artificially generated data (Sections 10.4.1, 10.4.2). The effectiveness of our approach is tested in an offline experiment with the aim of detecting finger and wrist movements from TMG data (Section 10.4.3). We show that the TME model outperforms the standard ME model and achieves similar performance as a GP at a lower computational cost, with the advantage of being easily interpretable due to the tensor structure. We finally validate the use of the proposed approach in a real-time teleoperation experiment, where participants controlled a robotic arm and hand by moving their wrist and closing/opening their hand (Section 10.4.4).

10.2 Background: Linear Models For Tensors

We introduce here the tensor decomposition necessary for the proposed TME, as well as the extensions of two forms of the generalized linear model, namely ridge regression and logistic regression, to tensor-variate data. Note that the basic tensor operations used in this chapter were introduced in Chapter 2.3.

10.2.1 Tensor Rank and CANDECOMP/PARAFAC Decomposition

Tensor rank

A rank-one tensor $\boldsymbol{\mathcal{Y}}$ of order M is a tensor that can be written as the outer product of M vectors, i.e.,

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{u}^{(1)} \circ \boldsymbol{u}^{(2)} \circ \ldots \circ \boldsymbol{u}^{(M)}. \tag{10.1}$$

In general, the smallest number of rank-one tensors that generate a tensor $\boldsymbol{\mathcal{Y}}$ as their sum is defined as the rank R of the tensor $\boldsymbol{\mathcal{Y}}$. Therefore, a rank-R tensor can be written as a sum of R rank-one tensors, i.e.,

$$\boldsymbol{\mathcal{Y}} = \sum_{r=1}^{R} \boldsymbol{u}_r^{(1)} \circ \boldsymbol{u}_r^{(2)} \circ \ldots \circ \boldsymbol{u}_r^{(M)}. \tag{10.2}$$

The CANDECOMP/PARAFAC decomposition

The CANDECOMP/PARAFAC (CP) decomposition, also called canonical polyadic decomposition, [Harshman, 1970; Carroll and Chang, 1970] factorizes a tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_M}$ as a sum of R rank-one tensors, i.e.,

$$\boldsymbol{\mathcal{Y}} \approx \sum_{r=1}^{R} \boldsymbol{u}_r^{(1)} \circ \boldsymbol{u}_r^{(2)} \circ \ldots \circ \boldsymbol{u}_r^{(M)}. \tag{10.3}$$

The CP decomposition can also be expressed in terms of the m-mode matricization and vectorization of the tensor $\boldsymbol{Y}_{(m)}$ and $\mathrm{vec}(\boldsymbol{\mathcal{Y}})$ as

$$\boldsymbol{Y}_{(m)} \approx \boldsymbol{U}^{(m)} \boldsymbol{U}^{(-m)\mathsf{T}}, \tag{10.4}$$

$$\mathrm{vec}(\boldsymbol{\mathcal{Y}}) \approx (\boldsymbol{U}^{(M)} \odot \ldots \odot \boldsymbol{U}^{(1)}) \mathbf{1}_R, \tag{10.5}$$

where $\odot$ denotes the Khatri-Rao product (2.23), $\boldsymbol{U}^{(m)} \in \mathbb{R}^{I_m \times R}$ are factor matrices defined as

$$\boldsymbol{U}^{(m)} = [\boldsymbol{u}_1^{(m)} \ldots \boldsymbol{u}_R^{(m)}], \tag{10.6}$$

$$\boldsymbol{U}^{(-m)} = (\boldsymbol{U}^{(M)} \odot \ldots \odot \boldsymbol{U}^{(m+1)} \odot \boldsymbol{U}^{(m-1)} \odot \ldots \odot \boldsymbol{U}^{(1)}), \tag{10.7}$$

and $\mathbf{1}_R \in \mathbb{R}^R$ is a vector containing R ones.

If $\boldsymbol{\mathcal{Y}}$ follows exactly a CP decomposition (10.3), the inner product (2.24) can equivalently be written as

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \langle \boldsymbol{X}_{(m)},\, \boldsymbol{U}^{(m)} \boldsymbol{U}^{(-m)\mathsf{T}} \rangle = \langle \boldsymbol{X}_{(m)} \boldsymbol{U}^{(-m)},\, \boldsymbol{U}^{(m)} \rangle, \tag{10.8}$$

by exploiting (10.4) and the properties of the Frobenius norm and matrix trace.

10.2.2 Generalized Linear Model for Tensors

Given a vector-valued input data $\boldsymbol{x}$, the generalized linear model (GLM) is given by

$$y = f(\boldsymbol{x}^{\mathsf{T}} \boldsymbol{w} + b) = f(\langle \boldsymbol{x}, \boldsymbol{w} \rangle + b), \tag{10.9}$$

where y is the predicted output, $\boldsymbol{w}$ is a vector of weights, b is the bias and $f(\cdot)$ is a function, see Figure 10.2a. This model can be naturally extended to matrix-valued data $\boldsymbol{X}$ with

$$\begin{aligned} y &= f(\boldsymbol{w}^{(1)\mathsf{T}} \boldsymbol{X} \boldsymbol{w}^{(2)} + b) \\ &= f\Big(\langle\, \boldsymbol{X}, \boldsymbol{w}^{(1)} \circ \boldsymbol{w}^{(2)} \,\rangle + b\Big), \end{aligned} \tag{10.10}$$

where $\boldsymbol{w}^{(1)}$ and $\boldsymbol{w}^{(2)}$ are vectors of weights. Following a similar procedure, the model can be generalized to M-dimensional tensor-valued data with

$$y = f\Big(\langle\, \boldsymbol{\mathcal{X}}, \boldsymbol{w}^{(1)} \circ \ldots \circ \boldsymbol{w}^{(M)} \,\rangle + b\Big), \tag{10.11}$$

as shown in Figure 10.2b. The key advantages of this representation, compared to vector-valued representation $y = f(\text{vec}(\boldsymbol{\mathcal{X}})^{\mathsf{T}}\boldsymbol{w} + b)$, are that the underlying structure of the tensor-valued data is taken into account in the model and that the number of parameters is reduced from $\prod_{m=1}^{M} I_m$ to $\sum_{m=1}^{M} I_m$. Moreover, more complex features can be represented by encoding the weight tensor as a sum of R rank-one tensors with

$$y = f\left(\left\langle\, \boldsymbol{\mathcal{X}}, \sum_{r=1}^{R} \boldsymbol{w}_r^{(1)} \circ \ldots \circ \boldsymbol{w}_r^{(M)} \,\right\rangle + b\right). \tag{10.12}$$

This model is represented in Figure 10.2c.

Similarly to the vector case, we obtain the tensor-valued linear and logistic regression models by defining the function $f(\cdot)$ as identity and as the softmax function, respectively.

10.2.3 Tensor Ridge Regression (TRR)

Given a vector-valued input data $\boldsymbol{x}$, the classical regression model is of the form

$$y = \boldsymbol{x}^{\mathsf{T}}\boldsymbol{w} + b + \epsilon = \langle \boldsymbol{x}, \boldsymbol{w} \rangle + b + \epsilon, \tag{10.13}$$

where y is the predicted output, $\boldsymbol{w}$ is a vector of weights, b is the bias and ϵ is a zero-mean Gaussian noise variable. Following (10.12), and as shown in [Guo et al., 2012; Zhou et al., 2013], the model can be generalized to M-dimensional tensor-valued data

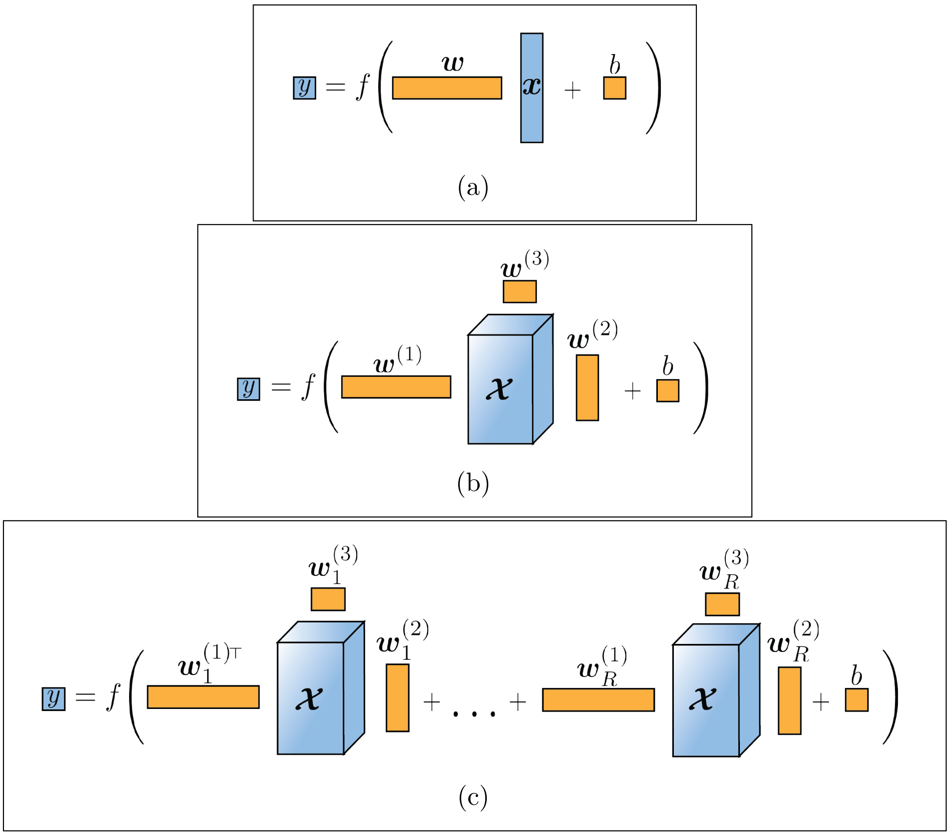
$$\begin{aligned} y &= \langle\, \boldsymbol{\mathcal{X}}, \sum_{r=1}^{R} \boldsymbol{w}_r^{(1)} \circ \ldots \circ \boldsymbol{w}_r^{(M)} \,\rangle + b + \epsilon \\ &= \langle\, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}} \,\rangle + b + \epsilon, \end{aligned} \tag{10.14}$$

therefore taking the underlying structure of the data into account.

Given a dataset of M-dimensional tensor inputs and corresponding outputs $\{\boldsymbol{\mathcal{X}}_n, y_n\}_{n=1}^{N}$, the parameters of the tensor ridge regression (TRR) model (10.14) are learned by maximizing its likelihood, or equivalently its corresponding log-likelihood

$$\ell(\boldsymbol{\mathcal{W}}, b, \sigma) = \sum_{n=1}^{N} \log \mathcal{N}\Big(y_n \Big| \langle\, \boldsymbol{\mathcal{X}}_n, \boldsymbol{\mathcal{W}} \,\rangle + b,\, \sigma^2\Big), \tag{10.15}$$

where σ^2 is the variance of the zero-mean Gaussian random variable ϵ. By using the inner product equality (10.8), one can observe that the model (10.14) is linear in



(a)

(b)

(c)

Figure 10.2 – Illustrations of generalized linear models (GLM). (*a*) GLM for vector-valued data. The data and model parameters are depicted in blue and orange, respectively. (*b*)-(*c*) Extensions of GLM to tensor-valued data. The representation of (*c*) allows the encoding of more complex behaviors as the weight tensor is encoded as a sum of rank-1 tensors.

$\boldsymbol{W}^{(m)} = [\boldsymbol{w}_1^{(m)} \ldots \boldsymbol{w}_R^{(m)}]$ individually, so that the parameters $\{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(M)}, b\}$ can be learned by optimizing a sequence of generalized linear models (see [Guo et al., 2012; Zhou et al., 2013] for details). Therefore, by adding to the log-likelihood function a zero-mean Gaussian prior on the weight tensor, equivalent to the regularization term $-\lambda_{\mathcal{W}} \sum_{m=1}^{M} \|\boldsymbol{W}^{(m)}\|_{\mathrm{F}}^2$, the bias b and factor matrices $\boldsymbol{W}^{(m)}$ are updated at each iteration until convergence with

$$\begin{aligned} \operatorname{vec}(\boldsymbol{W}^{(m)}) &\leftarrow (\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} + \lambda_{\mathcal{W}}\boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathsf{T}}(\boldsymbol{y} - b\,\mathbf{1}), \quad (10.16) \\ b &\leftarrow \frac{1}{N}\sum_{n=1}^{N} y_n - \langle \boldsymbol{\mathcal{X}}_n, \boldsymbol{\mathcal{W}} \rangle, \quad (10.17) \end{aligned}$$

where the n-th row of the matrix $\boldsymbol{\Phi}$ is equal to $\operatorname{vec}\Big(\boldsymbol{X}_{n,(m)}\boldsymbol{W}^{(-m)}\Big)$, the n-th element of the vector $\boldsymbol{y}$ is y_n, $\mathbf{1} \in \mathbb{R}^N$ is a vector of N ones and $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. Note that other types of regularization are also proposed in [Guo et al., 2012].

10.2.4 Tensor Logistic Regression (TLR)

In the classical multi-class logistic regression model, the posterior probability of the class $\mathcal{C}_i$ is given by the softmax function

$$p(\mathcal{C}_i|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{v}_i + a_i)}{\sum_{j=1}^{C}\exp(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{v}_j + a_j)}, \quad (10.18)$$

where $\boldsymbol{\theta}$ denotes the parameters of the model and C the number of classes. Similarly as ridge regression, the logistic regression model can be extended to tensor-valued data by encoding the tensor of weights as a sum of R rank-one tensors, leading to the tensor-valued softmax function [Hung and Wang, 2013; Tan et al., 2013]

$$\pi_i = p(\mathcal{C}_i|\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}) = \frac{\exp\Big(\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{V}}_i \rangle + a_i\Big)}{\sum_{j=1}^{C}\exp\Big(\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{V}}_j \rangle + a_j\Big)}, \quad (10.19)$$

where $\boldsymbol{\mathcal{V}}_i = \sum_{r=1}^{R} \boldsymbol{v}_{i,r}^{(1)} \circ \ldots \circ \boldsymbol{v}_{i,r}^{(M)}$. Similarly to TRR, the tensor logistic regression (TLR) model takes into account the underlying structure of the data and reduces the number of parameters in the model compared to a vector-based representation of the tensor-valued data.

Given a dataset of inputs and corresponding unit vector outputs $\{\boldsymbol{\mathcal{X}}_n, y_n\}_{n=1}^{N}$, where $y_{n,i} = 1$ indicates that the n-th data belong to the i-th class, the log-likelihood of the multivariate tensor logistic regression model is

$$\begin{aligned}\ell\Big(\{\boldsymbol{\mathcal{V}}_i, a_i\}_{i=1}^C\Big) &= \log \prod_{n=1}^N \prod_{i=1}^C \pi_i^{y_{n,i}} \\ &= \sum_{n=1}^N \left(\sum_{i=1}^C y_{n,i} \Big(\langle \boldsymbol{\mathcal{X}}_n, \boldsymbol{\mathcal{V}}_i \rangle + a_i \Big) - \log \sum_{i=1}^C \exp\Big(\langle \boldsymbol{\mathcal{X}}_n, \boldsymbol{\mathcal{V}}_i \rangle + a_i \Big) \right). \end{aligned} \tag{10.20}$$

Note that a regularization term $-\lambda_{\boldsymbol{\mathcal{V}}} \sum_{i=1}^C \sum_{m=1}^M \|\boldsymbol{V}_i^{(m)}\|_{\mathrm{F}}^2$ can be added to the log-likelihood function to avoid overfitting. The parameters $\{\boldsymbol{V}_i^{(1)}, \ldots, \boldsymbol{V}_i^{(M)}, a_i\}_{i=1}^C$ can be learned by minimizing the negative log-likelihood of the model via any gradient-based optimizer, e.g., Newton's method or limited memory BFGS. By exploiting (10.8), the gradients of the regularized negative log-likelihood used in the optimization process can be computed as

$$\begin{aligned}\frac{\delta\Big(-\ell(\{\boldsymbol{\mathcal{V}}_i, a_i\}_{i=1}^C)\Big)}{\delta \boldsymbol{V}_i^{(m)}} &= \sum_{n=1}^N (\pi_{n,i} - y_{n,i}) \mathrm{vec}(\boldsymbol{X}_{n,(m)} \boldsymbol{V}_i^{(-m)}) \\ &\quad + 2\,\lambda_{\boldsymbol{\mathcal{V}}}\, \mathrm{vec}(\boldsymbol{V}_i^{(m)}), \end{aligned} \tag{10.21}$$

$$\frac{\delta\Big(-\ell(\{\boldsymbol{\mathcal{V}}_i, a_i\}_{i=1}^C)\Big)}{\delta a_i} = \sum_{n=1}^N (\pi_{n,i} - y_{n,i}), \tag{10.22}$$

where $\pi_{n,i} = p(\mathcal{C}_i | \boldsymbol{\mathcal{X}}_n, \theta)$.

10.3 Tensor-variate Mixture of Experts

A mixture of experts (ME) regression model [Jacobs et al., 1991] aims at solving a nonlinear supervised learning problem by combining the predictions of a set of experts. The model is composed of a gate determining a soft division of the input space, and several experts making predictions in the different regions of the input space. In this section, we propose to generalize the ME regression model to tensor-variate data by using tensor-variate models for the experts and for the gate.

10.3.1 TME Model

Given a tensor-variate input $\boldsymbol{\mathcal{X}}$ and an output $\boldsymbol{y}$, the tensor-variate mixture distribution is

$$p(\boldsymbol{y}|\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}) = \sum_{i=1}^C p(i|\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}_g)\, p(\boldsymbol{y}|i, \boldsymbol{\mathcal{X}}, \boldsymbol{\theta}_e), \tag{10.23}$$

where C is the number of experts, $p(i|\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}_g)$ is the probability of the i-th expert to be activated (gate's rating) and $p(\boldsymbol{y}|i, \boldsymbol{\mathcal{X}}, \boldsymbol{\theta}_e)$ is the model of the i-th expert. We define

$\boldsymbol{\theta} = \{\boldsymbol{\theta}_g, \boldsymbol{\theta}_e\}$, where $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_e$ denote the parameters of the gate and the set of experts, respectively.

Similarly to the original ME model, we define the gate of the TME model by the tensor-variate softmax function, so that

$$p(i|\boldsymbol{\mathcal{X}}, \boldsymbol{\theta}_g) = \pi_i, \tag{10.24}$$

with π_i defined by (10.19), $\boldsymbol{\mathcal{V}}_i = \sum_{r=1}^{Rg} \boldsymbol{v}_{i,r}^{(1)} \circ \ldots \circ \boldsymbol{v}_{i,r}^{(M)}$ and R_g the rank of the weight tensors $\boldsymbol{\mathcal{V}}_i$. The experts follow the Gaussian model

$$p(\boldsymbol{y}|i, \boldsymbol{\mathcal{X}}, \boldsymbol{\theta}_e) = \mathcal{N}\Big(\boldsymbol{y}\Big|\boldsymbol{\psi}_i(\boldsymbol{\mathcal{X}}) + \boldsymbol{b}_i, \boldsymbol{\Sigma}_i\Big), \tag{10.25}$$

where $\boldsymbol{\psi}_i(\boldsymbol{\mathcal{X}}) = \begin{pmatrix} \langle\, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}}_{i,1} \,\rangle \\ \vdots \\ \langle\, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}}_{i,D} \,\rangle \end{pmatrix}$, $\boldsymbol{\mathcal{W}}_{i,d} = \sum_{r=1}^{R_{e_i}} \boldsymbol{w}_{i,d,r}^{(1)} \circ \ldots \circ \boldsymbol{w}_{i,d,r}^{(M)}$ and R_{e_i} is the rank of the weight tensors $\boldsymbol{\mathcal{W}}_{i,d}$. Note that one weight tensor $\boldsymbol{\mathcal{W}}_{i,d}$ is defined for each element of $\boldsymbol{y} \in \mathbb{R}^D$. This is similar to the vector case, where different vectors $\boldsymbol{w}_i$ weight the input for each element of the output, so that $\boldsymbol{y} = \begin{pmatrix} \boldsymbol{w}_1^{\mathsf{T}} \\ \vdots \\ \boldsymbol{w}_D^{\mathsf{T}} \end{pmatrix} \boldsymbol{x} + \boldsymbol{b}$.

Figure 10.3 illustrates the proposed model. Single predictions are computed by using the expectation of the TME model (10.23), so that

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{C} \pi_i(\boldsymbol{\psi}_i(\boldsymbol{\mathcal{X}}) + \boldsymbol{b}_i). \tag{10.26}$$

10.3.2 Training of TME

Similarly to ME, the TME model can be trained using the expectation-maximization (EM) algorithm. By introducing a set of binary latent variables $\{\boldsymbol{z}_n\}$ where $z_{n,i} = 1$ indicate that the data n was generated by the i-th mixture component, the expected complete data log-likelihood is given by

$$Q(\boldsymbol{\theta}) = \sum_{n=1}^{N}\sum_{i=1}^{C} r_{n,i} \log\Big(\pi_{n,i}\mathcal{N}\Big(\boldsymbol{y}_n\Big|\boldsymbol{\psi}_i(\boldsymbol{\mathcal{X}}_n) + \boldsymbol{b}_i, \boldsymbol{\Sigma}_i\Big)\Big), \tag{10.27}$$

where $r_{n,i}$ denotes the responsibility of the i-th component for the n-th data point so that $r_{n,i} = p(z_{n,i} = 1|\boldsymbol{\mathcal{X}}_n, \boldsymbol{\theta})$. In the E-step, the responsibilities $r_{n,i}$ are computed using

$$r_{n,i} = \frac{\pi_{n,i}\, \mathcal{N}\Big(\boldsymbol{y}_n\Big|\boldsymbol{\psi}_i(\boldsymbol{\mathcal{X}}_n) + \boldsymbol{b}_i, \boldsymbol{\Sigma}_i\Big)}{\sum_{j=1}^{C} \pi_{n,j}\, \mathcal{N}\Big(\boldsymbol{y}_n\Big|\boldsymbol{\psi}_j(\boldsymbol{\mathcal{X}}_n) + \boldsymbol{b}_j, \boldsymbol{\Sigma}_j\Big)}. \tag{10.28}$$

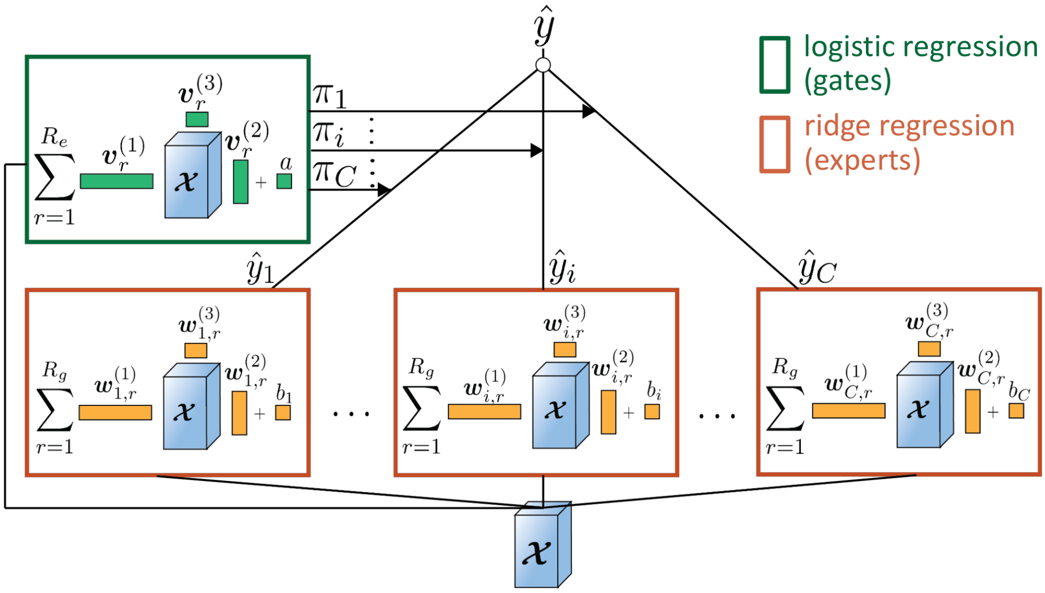

$\hat{y}$
logistic regression (gates)
ridge regression (experts)
π_1
π_i
π_C
$\sum_{r=1}^{R_e}$
$\boldsymbol{v}_r^{(1)}$
$\boldsymbol{v}_r^{(2)}$
$\boldsymbol{v}_r^{(3)}$
a
$\hat{y}_1$
$\hat{y}_i$
$\hat{y}_C$
$\sum_{r=1}^{R_g}$
$\boldsymbol{w}_{1,r}^{(1)}$
$\boldsymbol{w}_{1,r}^{(2)}$
$\boldsymbol{w}_{1,r}^{(3)}$
b_1
$\boldsymbol{w}_{i,r}^{(1)}$
$\boldsymbol{w}_{i,r}^{(2)}$
$\boldsymbol{w}_{i,r}^{(3)}$
b_i
$\boldsymbol{w}_{C,r}^{(1)}$
$\boldsymbol{w}_{C,r}^{(2)}$
$\boldsymbol{w}_{C,r}^{(3)}$
b_C
$\boldsymbol{\mathcal{X}}$

Figure 10.3 – Proposed TME model. The gate is represented in the green box and the experts are represented in orange boxes.

In the M-step, the parameters are updated to maximize the expected complete data log-likelihood (10.27). First, the parameters of the experts $\boldsymbol{\theta}_e$ are updated iteratively until convergence, based on (10.16) and (10.17), with

$$\mathrm{vec}(\boldsymbol{W}_{i,d}^{(m)}) \leftarrow (\tilde{\boldsymbol{\Phi}}_{i,d}^{\mathsf{T}}\tilde{\boldsymbol{\Phi}}_{i,d} + \lambda_{\mathcal{W}}\boldsymbol{I})^{-1}\tilde{\boldsymbol{\Phi}}_{i,d}^{\mathsf{T}}(\tilde{\boldsymbol{y}}_d - b_{i,d}\boldsymbol{1}), \tag{10.29}$$

$$b \leftarrow \frac{1}{N}\sum_{n=1}^{N} \tilde{y}_{n,d} - \langle \tilde{\boldsymbol{\mathcal{X}}}_n, \boldsymbol{\mathcal{W}}_{i,d}\rangle, \tag{10.30}$$

where the n-th row of the matrix $\tilde{\boldsymbol{\Phi}}_{i,d}$ is equal to $\mathrm{vec}\Big(\tilde{\boldsymbol{X}}_{n,(m)}\boldsymbol{W}_{i,d}^{(-m)}\Big)$, the n-th element of the vector $\tilde{\boldsymbol{y}}$ is $\tilde{y}_{n,d}$ and $\tilde{\boldsymbol{\mathcal{X}}}_n = \sqrt{r_{nk}}\boldsymbol{\mathcal{X}}_n$, $\tilde{\boldsymbol{y}}_n = \sqrt{r_{nk}}\boldsymbol{y}_n$ are the scaled input tensors and output vectors, respectively. The covariance of the experts Gaussian model is then updated as

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{n=1}^{N} r_{n,i}\Big(\boldsymbol{y}_n - \boldsymbol{\psi}_i(\boldsymbol{\mathcal{X}}_n) - \boldsymbol{b}_i\Big)^{\mathsf{T}}\Big(\boldsymbol{y}_n - \boldsymbol{\psi}_i(\boldsymbol{\mathcal{X}}_n) - \boldsymbol{b}_i\Big)}{\sum_{n=1}^{N} r_{n,i}}. \tag{10.31}$$

Finally, the gate parameters $\boldsymbol{\theta}_g$ are updated by maximizing the log-likelihood of the multivariate tensor logistic regression model

$$\ell\Big(\{\boldsymbol{\mathcal{V}}_i, a_i\}_{i=1}^{C}\Big) = \log \prod_{n=1}^{N}\prod_{i=1}^{C} \pi_{n,i}^{r_{n,i}} - \lambda_{\mathcal{V}}\sum_{i=1}^{C}\sum_{m=1}^{M}\|\boldsymbol{V}_i^{(m)}\|_{\mathrm{F}}^2, \tag{10.32}$$

based on (10.20). Similarly to tensor logistic regression, a gradient-based optimizer is used to minimize the negative log-likelihood with gradients given by (10.21) and (10.22), where $y_{n,i}$ is replaced by $r_{n,i}$.

Note that regularization terms have been added in the M-step to avoid overfitting. The E-step and M-step are iterated until convergence of the algorithm.

10.3.3 Model Selection and Initialization

Selecting the number of experts in ME is known to be a difficult problem [Yuksel et al., 2012]. When the structure of the application allows it, as in the experiments of Section 10.4, the number of experts can be determined by the experimenter. Otherwise, standard strategies used for ME, such as exhaustive search, growing or pruning models, or Bayesian estimates can be adapted to TME.

The TME model assumes fixed ranks R_g and R_{e_i} for the gate and experts weight tensors, respectively. The appropriate rank can be estimated using cross-validation or through usual model selection criterion, e.g., the Bayesian information criterion (BIC).

Previous works on TRR [Guo et al., 2012; Zhou et al., 2013] and TLR [Tan et al., 2013] showed that both TRR and TLR models converge to a similar solution independently of the initial weight values. Therefore, the weight tensors of TME are initialized with random values in our experiments. In order to facilitate the convergence, we initialized the weights of the expert model $\boldsymbol{\mathcal{W}}_i$ and of the gate $\boldsymbol{\mathcal{V}}_i$ as equal to the weights $\boldsymbol{\mathcal{W}}$ obtained from TRR.

10.4 Experiments

In this section, we first evaluate the functionality and the performance of the proposed TME on artificially generated data. The approach is then applied to the detection and recognition of hand movements from tactile myography (TMG) data. An offline experiment and a real-time teleoperation experiment, where participants controlled a robotic arm and hand based on their hand movements, illustrate the effectiveness of the proposed TME model.

10.4.1 2D Shape Example

In this illustrative example, we propose to evaluate the performance of the proposed TME model for different ranks under various sample sizes and signal strengths. To do so, we generate artificial data following the model (10.23) from known parameters $\boldsymbol{\theta}$ and we evaluate the recovery of these parameters by the model. In this illustrative example, we consider matrix-variate inputs $\boldsymbol{X} \in \mathbb{R}^{64\times 64}$ whose elements are independent and normally distributed. The output y is normally distributed with a mean given by a 2-class TME model with zero biases

$$\hat{y} = \frac{\exp\Big(\langle \boldsymbol{X}, \boldsymbol{V} \rangle\Big)}{1 + \exp\Big(\langle \boldsymbol{X}, \boldsymbol{V} \rangle\Big)} \langle \boldsymbol{X}, \boldsymbol{W}_1 \rangle \; + \frac{1}{1 + \exp\Big(\langle \boldsymbol{X}, \boldsymbol{V} \rangle\Big)} \langle \boldsymbol{X}, \boldsymbol{W}_2 \rangle, \tag{10.33}$$

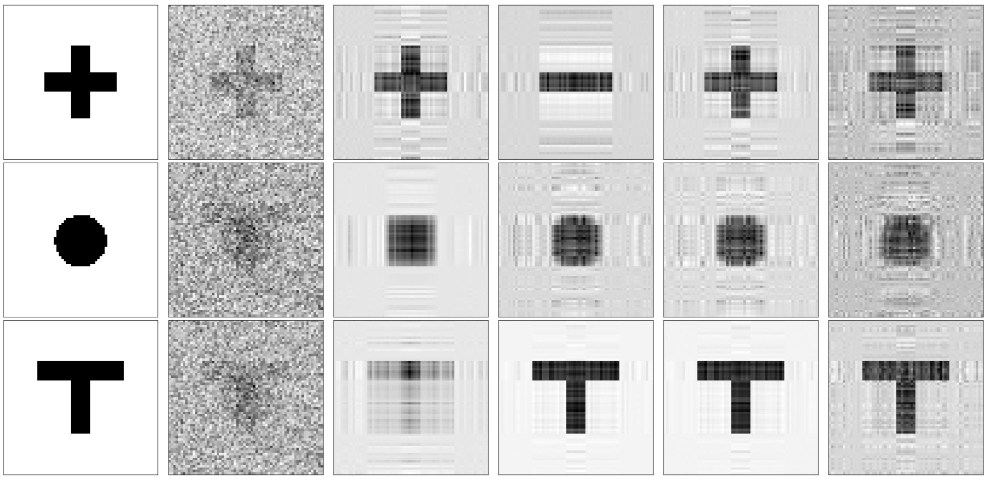
and a standard deviation σ. The weight matrices $\boldsymbol{V}, \boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{64 \times 64}$ are equal to the binary 2D shapes represented in Figure 10.4a, where the black and white regions correspond to 1 and 0, respectively. The use of these 2D shapes was inspired by the illustrative example presented in [Zhou et al., 2013].

We first examine the performance of the proposed TME model for ranks R_g and $R_e = R_{e_1} = R_{e_2}$ varying from 1 to 3 with a sample size $N = 1000$ equally divided between the two classes and a noise level σ equal to 10% of the standard deviation of the mean $\hat{y}$. The regularization terms $\lambda_{\boldsymbol{V}}$ and $\lambda_{\boldsymbol{\mathcal{W}}}$ were fixed as equal to 0.1. Moreover, we compare the TME model with the standard ME regression model whose gate is defined by the softmax function (10.18) and experts follow a Gaussian model with a mean given by (10.13).

Figure 10.4 shows the original and recovered weight matrices by the ME and TME models along with the root-mean-square error (RMSE) for the estimation of the weight matrices and the BIC value for TME. We observe that TME outperforms ME for all the tested rank values as the maximum RMSE value achieved by TME is 0.21 ($R_g = 1$, $R_e = 3$) versus 0.3 for the ME model. Moreover, we observe that the weight matrices retrieved by ME are noisier and the shapes of the experts weights are not clearly delimited and tend to be fused together compared to those retrieved by TME. Similar results were obtained for different sample sizes and noise levels.

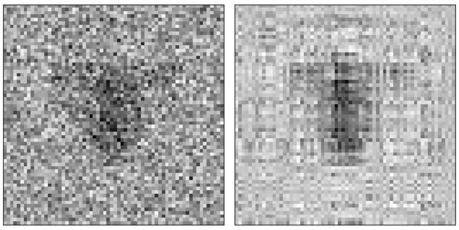
Similar observations can also be made by comparing the weight matrices retrieved by RR and MRR for the same data, as shown in Figure 10.5. Although both methods retrieve one weight matrix fusing the three original ones due to their formulation, the weight matrix retrieved by RR looks noisier than the one retrieved by MRR. This confirms that taking the structure of the data into account improves the quality of the recovered weight matrices.

Due to their structure, a rank-2 setting is sufficient to capture a cross or a T-shape pattern, while a low rank setting does not allow to exactly represent a disk shape. As expected, the cross and T-shape are fully recovered by TME in the cases where $R_g \geq 2$ and $R_e \geq 2$, respectively, while approximations of the shapes are obtained for lower ranks (see Figures 10.4c–10.4f). Moreover, while the disk shape is approximated by a square in a rank-1 setting (Figure 10.4c), it is already fairly recovered by a rank-2 or rank-3 setting (see Figures 10.4d–10.4f).



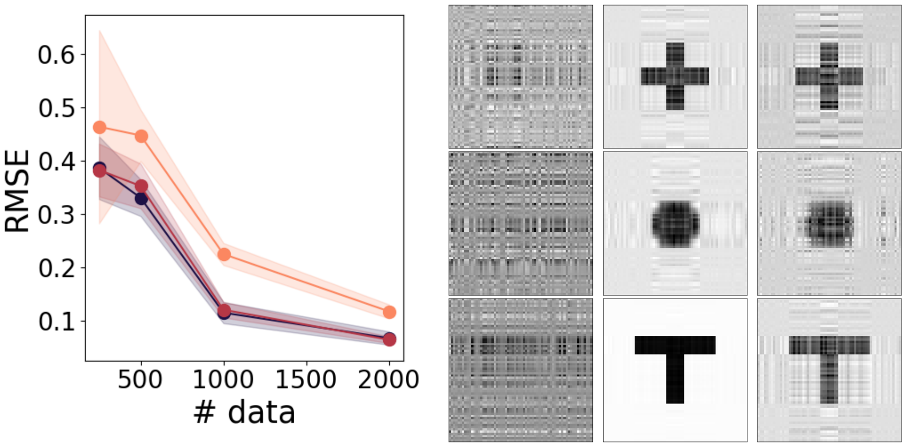
(a)	(b) ME	(c) TME (2,1)	(d) TME (1,2)	(e) TME (2,2)	(f) TME (3,3)
RMSE	0.30	0.16	0.18	0.12	0.18
BIC	-	10209	8940	9835	11216

Figure 10.4 – (*a*) True weight matrices of the gate and experts functions (from *top* to *bottom*: $\boldsymbol{V}$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$). (*b*) Recovered weight matrices by ME. (*c-f*) Recovered weight matrices by TME for different ranks (R_g, R_e).



(a) RR (b) MRR

Figure 10.5 – Recovered weight matrix by RR and MRR.



(a)

(b)

Figure 10.6 – (a) Evolution of the RMSE of the estimation of TME weight matrices in function of the sample size N for different noise levels. The curves corresponding to noise levels σ equal to 1, 10 and 50% of the standard deviation of the mean $\hat{y}$ are represented in dark blue, red and yellow, respectively. The mean and two standard deviations over 10 trials with different matrix-variate inputs $\boldsymbol{X}$ are represented. The sample size is equally divided between the two classes. (b) Recovered weight matrices $\boldsymbol{V}$, $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ by TME for ranks $R_g = R_e = 2$ with, from *left* to *right*, sample sizes $N = 500$, 2000 and 2000 and noise levels $\sigma = 1\%$, 1% and 50% of $\hat{y}$.

Consistently with the aforementioned observations, the minimum RMSE value is obtained by TME with a rank-(2, 2) setting. Moreover, TME with ranks $R_g = R_e = 3$ obtains a slightly higher RMSE than with ranks $R_g = R_e = 2$. This can be explained by the fact that approximating the cross and t-shape with a rank-3 setting, while a rank-2 setting is sufficient, leads to an overfitted estimation with a higher influence of the noise contained in the training data. According to the BIC values reported for the tested TME models, the model with $R_g = 1$ and $R_e = 2$ should be selected (lowest BIC cost). However, in practice, one may prefer the rank-(2,2) setting in this case, suggesting that other rank selection methods, such as cross-validation, may be used in function of the application. Note that similar observations were made for different sample sizes and noise levels.

As shown in Figure 10.6, the estimation accuracy increases with the sample size and decreases with the noise level σ, validating the consistency of the proposed method.

10.4.2 Shape Example of Higher Dimensions

In order to evaluate the performance of the proposed TME model for higher tensor dimensions, we extended the experiment to tensor-variate inputs $\boldsymbol{\mathcal{X}}$ of order 3 and 4. The dimension of the inputs and coefficients was reduced from 64 to 16 in order to allow the comparison with ME. Indeed, the number of elements of a third-order cube

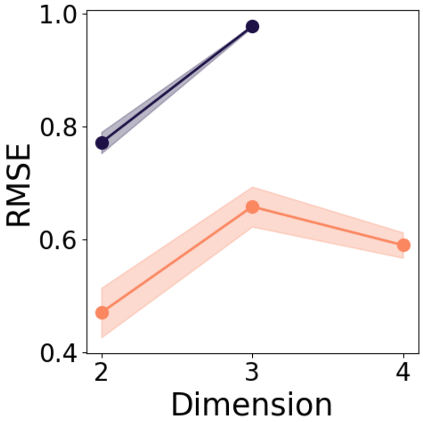


Figure 10.7 – Evolution of the RMSE in function of the dimension of the data for the estimation of the weight tensors by ME and TME, depicted in dark blue and yellow, respectively. The mean and two standard deviations over 10 trials with different inputs are represented.

tensor of dimension 64 is 262144. Therefore, with a standard implementation of ME, 262144×262144 matrices need to be stored and inverted, which cannot be handled in a straightforward manner with a standard computer. Note that other techniques, notably sparse methods, could be used to handle such a case. However, as this is out of the scope of this chapter, we simply reduced the dimension of our coefficients.

For this second illustrative example, the tensor-variate inputs $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{16\times\ldots\times16}$ and the outputs y were generated as in the previous experiment, with tensor coefficients instead of matrices in (10.33). The weight tensors $\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{W}}_1$ and $\boldsymbol{\mathcal{W}}_2 \in \mathbb{R}^{16\times\ldots\times16}$ were defined as 2D, 3D and 4D binary coefficients with shapes similar to the ones represented in Figure 10.4a extended to higher dimensions. For this experiment, the background and shape regions correspond to 0 and 1. We compared the performances of ME with the proposed TME model with ranks $R_g = R_e = 2$ with a sample size $N = 200$ equally divided between the two classes and a noise level σ equal to 10% of the standard deviation of the mean $\hat{y}$. The regularization terms $\lambda_{\boldsymbol{\mathcal{V}}}$ and $\lambda_{\boldsymbol{\mathcal{W}}}$ were fixed as equal to 0.1.

Figure 10.7 shows the mean and two standard deviations of the RMSE obtained for the estimation of the weight tensors by ME and TME for different dimensions. Note that no result is presented for ME with coefficients of dimension 4 due to the computational load of storing and inverting 65536×65536 matrices. We observe that TME outperforms ME for all dimensions. Moreover, the RMSE explodes for ME with dimension 3 as it almost reaches the maximum RMSE of 1, while it grows slowly for TME from dimensions 2 to 4. Considering that the number of training data was constant and that the number of elements of the coefficients grew considerably with the number of dimensions, this result highlights the benefits of TME (and of considering tensor-based approaches).

10.4.3 Detection of Hand Movements from Tactile Myography

As mentionned previously, previous studies showed that ridge regression (RR) directly applied to the data of the bracelet allows the prediction of different finger and wrist movements. However, RR does not take into account the matrix structure of the TMG data as they are vectorized before the application of the regression method. Moreover, the same weight vectors are used independently of the activated movements, which may result in false positive detection of activations. Therefore, the motivations to use TME for this application are the following: (*i*) the structure of the data is taken into account in the regression process; (*ii*) the problem is decomposed in two subparts, namely detecting which movements are activated and determining their individual level of activation; and (*iii*) the low computational complexity to evaluate one test sample allows TME to be used for real-time detection of hand and wrist movements from TMG data.

In this experiment, we investigate the performance of TME to recognize different finger and wrist movements offline. The dataset was gathered from 9 healthy participants requested to replicate the movements of a stimulus in the form of a 9-DOF hand model while wearing the tactile bracelet (see Figure 10.8). Ground truth was obtained from the values of the animated hand model displayed on a monitor. This method has the drawback of possibly reducing the precision of the prediction of the intended activations due to the delay required by the participant to replicate the displayed movement. However, this approach has been widely used in the literature [Nielsen et al., 2011; Sierra González and Castellini, 2013] as it allows the association of intended activations with input signal patterns in the case of amputees (since ground truth data can obviously not be collected by other means in this case). Each participant executed three times a sequence of six movements, namely wrist flexion, wrist extension, wrist supination, thumb flexion, index flexion and little-finger flexion. Each stimulus followed a cycle of 20 s, consisting of a transition phase (5 s), an activation phase (6 s), a transition phase (5 s) and a relaxing phase (4 s). Data were recorded during the whole cycle in order to obtain the whole range of activation from rest to complete finger and wrist movements. Data acquired from the tactile bracelet and the visual stimulus were synchronized by linearly interpolating the samples of each channel. TMG data were low-pass filtered (first-order Butterworth filter with cut-off frequency of 1 Hz) to remove high-frequency disturbances.

The training dataset is composed of data recorded at zero and full activation. The testing dataset is composed of data recorded during the transition parts, containing the whole range of intermediate activation levels. Therefore, the evaluation of the performance of the model is compatible with the evaluation in forecasted studies with amputees, as they cannot provide accurate intermediate training data.

We compared the performance of vector-based and tensor-based algorithms on this dataset, namely RR, ME, TRR and TME. We also contrasted the results of these

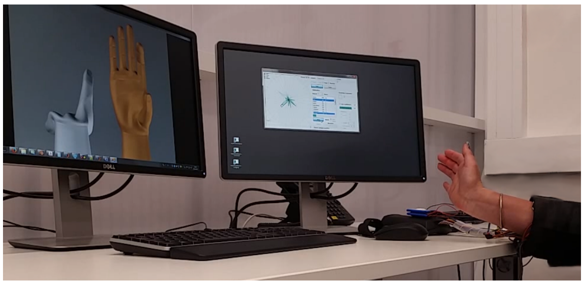


Figure 10.8 – Data collection setup for the detection of hand movements from TMG. The participant, wearing the tactile bracelet, imitates the gray animated hand model.

(mixtures of) linear models with a computationally more involved nonlinear method based on Gaussian process regression (GPR).

The TMG data were centered for all the methods. The regularization parameter of RR and ME were fixed to 0.1. GPR was used with a radial basis function (RBF) kernel whose parameters were optimized using GPy [GPy, 2012]. The rank of TRR was determined using 5-fold cross-validation and the regularization parameter $\lambda_{\mathcal{W}}$ was fixed to 0.1. For TME, as for ME, one expert was considered for each of the six finger and wrist movements. In order to facilitate the training process, we considered a common value R_e for the ranks R_{e_i}. The ranks R_g and R_e were determined using 5-fold cross-validation for $2 \leq R_g, R_e \leq 6$. The regularization parameters of TME, $\lambda_{\mathcal{W}}$ and $\lambda_{\mathcal{V}}$, were fixed by the experimenter as equal to 0.1. We noticed that small variations of these regularization parameters did not change significantly the results for the different regression methods.

Table 10.1 shows the mean and standard deviation over the 9 participants of the RMSE between the ground truth and the prediction for the aforementioned methods. We observe that taking into account the structure of the data in the regression process improves the quality of predictions as both TRR and TME outperform their vector counterparts. More surprisingly, taking into account the structure of the data allows a linear method (TRR) to achieve performance comparable to those obtained by a nonlinear method. GP and TME achieve the best performance compared to the other methods, with the TME approach with linear models obtaining only a slightly lower RMSE (0.303 ± 0.074) than the nonlinear GP (0.305 ± 0.060).[1] Moreover, TME obtained the minimum RMSE for 5 participants out of 9.

Figure 10.9 shows an example for GP and TME of the original and recovered activations for all movements over time. We observe that TME is generally recovering a more stable signal than GP when one movement is not activated. In the regions of zero activations, the signal recovered by GP tends to oscillate around zero. However, the signal recovered

---

[1] Note that GP with RBF kernel slightly outperformed GP with linear (0.455 ± 0.061), Matérn 32 (0.308 ± 0.061) and Matérn 52 (0.306 ± 0.059) kernels, therefore all the results are presented with RBF kernel.

Table 10.1 – Performance comparison in terms of RMSE between different regression methods to predict fingers and wrist movements from TMG data.

RR	ME	GP	TRR	TME
0.45 ± 0.07	0.33 ± 0.07	0.30 ± 0.06	0.35 ± 0.11	0.30 ± 0.07

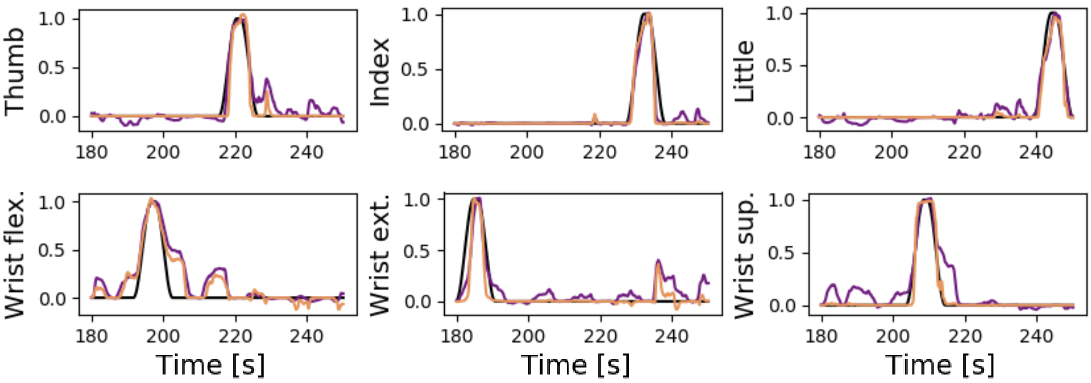


Figure 10.9 – Original and recovered activations of the different fingers and wrist movements over time. The whole range of activation is represented from 0 to 1 on the vertical axis for each movement. The ground truth is shown by black curves, while the signals recovered by TME and GP are displayed in yellow and purple, respectively.

by TME can have a bigger delay than GP to detect an activation different than zero (see, e.g., Fig. 10.9, wrist extension).

Table 10.2 shows the average testing computation time for the tested regression methods. The computation times were measured using a non-optimized Python code on a laptop with 2.7GHz CPU and 32 GB of RAM.We observe a testing time of 1 ms for TME, which is reasonable for real-time applications allowing predictions at a frequency > 50 Hz, as usually targeted by real-time detection of hand movements. Importantly, as opposed to GP, the computation testing time of TME is independent of the number of training data and depends only on the number of experts. Therefore, TME can be adapted to real-time predictions independently on the number of provided training data. During training, TME converged with less than 10 iterations of the EM algorithm for all the participants, with a total training time of several minutes. This is mainly due to the fact that a TLR model is optimized at each step of the EM algorithm. While the training time of TME could be easily improved by using dedicated tensor libraries such as [Kossaifi et al., 2019], and despite we used a naive implementation in our experiments, the training time remained reasonable for the method to be applied in real-time, as we show in the next subsection.

Table 10.2 – Average testing computation time for the different regression methods. The methods are trained on ~ 1000 data samples. The testing computation time is measured for one data sample.

	RR	ME	GP	TRR	TME
Testing [ms]	0.0065	0.08	1.5	0.035	1.0

10.4.4 Real-time Teleoperation with Tactile Myography

To evaluate our method in a scenario closer to the end-user case, we conducted a real-time teleoperation experiment in which 11 non-amputated participants (one female and ten males) controlled a robotic hand and arm based on the activation of the muscles on their forearm.

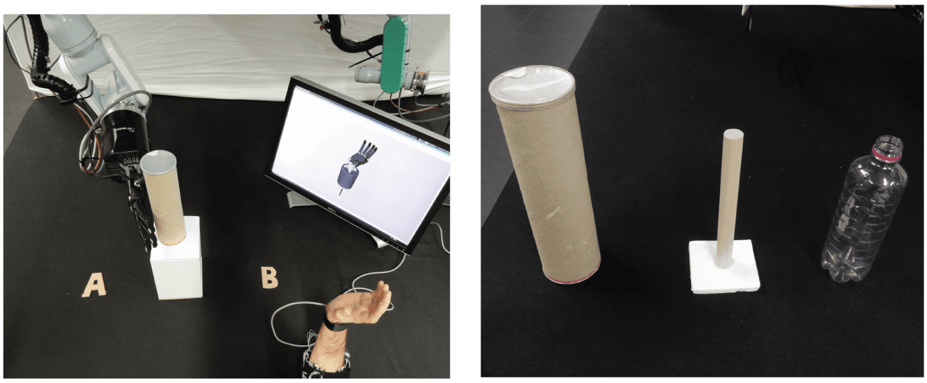
In the first part of the experiment, a protocol similar to the data collection of the experiment of Section 10.4.3 was applied to collect TMG data associated with the hand postures of the participants. The tactile bracelet was placed on the forearm of the participants with the closing gap on the ulna bone. The participants, wearing the tactile bracelet and sitting in front of a monitor, were asked to replicate the movements of a model of the 24-DOFs dexterous motor hand of the Shadow Robot Company [Shadow robot company, 1997]. Similarly to the previous experiment, ground truth was obtained from the values of the animated hand model. Each participant executed four times the sequence of four movements, namely wrist flexion, wrist extension, power grasp and fingers extension. The participants were asked to perform the different movements in a relaxed way (particularly, the fingers were relaxed during wrist movements). Each stimulus follows a cycle of 14 s composed of a transition phase (2 s), an activation phase (6 s), a transition phase (2 s) and a relaxing phase (4 s). The data collected during the activation and relaxing phases, i.e., at zero and full activations, were used to train the regression models.

During the second part of the experiment, the participants teleoperated a Shadow robot hand mounted on a 7-DOFs Mitsubishi PA10 robot arm. They were sitting in front of the robotic system with the palm of the Shadow robot hand facing right, as showed in Figure 10.10a. The different movements taught to the model in the first part of the experiments were mapped to the robotic system as follows: wrist flexion and extension were used to move the arm forward (in the direction of the palm) and backward (in the direction of the back of the hand), respectively. Power grasp and fingers extension were used to close and open the Shadow robot hand. When wrist flexion or extension was detected above a certain activation threshold, the velocity of the robot arm was incremented in the corresponding direction proportionally to the detected activation. Similarly, the posture of the robotic hand was incremented proportionally to the activation of power grasp or fingers extension if they were detected above a predefined threshold. The detected activations were also displayed on the Shadow robot hand model as in the first part of the experiment.

At the beginning of the second part of the experiment, the participants could get used to the learned mapping by controlling the simulated Shadow robot hand for a few minutes. Then, while teleoperating the real robotic system, the participants were asked to control the arm in order to approach it close to an object placed on a cube, to grasp this object and to bring it to a specific location on the left (A) or on the right (B) side and to release it. The complete setup is shown in Figure 10.10a. Three objects with different diameters were considered, namely a chips cylinder (⌀75 mm), a thin woodstick (⌀21 mm) and a PET bottle (⌀63 mm), as shown in Figure 10.10b. A total of 8 tasks were executed by each participant. The first six tasks consisted of bringing each object to A and then to B. Once a contact with the object was detected by the tactile fingertip sensors of the Shadow robot hand, the grasp pose was automatically maintained by the hand so that the participants could relax their fingers and focus on the wrist motion to steer the arm. The grasp pose was released as soon as a fingers extension command was detected. The maintenance of the grasp and the release were announced verbally by the system. The two last tasks consisted of bringing the PET bottle to A and to B without any holding assistance by the robot system. The time to complete each task was limited to two minutes. In case the object felt from the cube or was released out of the desired area, the experimenter replaced it at the initial position and the participant continued to execute the task in the remaining time. In case the participants could not control both the arm and the hand, e.g., if the arm was drifting continuously in one direction, the arm commands were disabled and the participants were requested to maintain a grasp on the object for 10 s before releasing it. Each participant tried to complete the 8 tasks with two different regression methods, namely TME and RR, trained on the data collected in the first part of the experiment. RR was chosen for comparison since it is considered as the baseline method for regression with TMG data. The order in which the two methods were tested was alternated between the participants.

Figure 10.11 shows snapshots of a participant executing different tasks. 6, respectively 7, out of the 11 participants were able to control both the arm and the hand during the whole experiment by using TME and RR, respectively. Note that the cases during which the arm commands had to be disabled occurred mainly for the second tested method (4 participants out of 5 testing TME as second method and 3 out of 4 participants testing RR as second method), suggesting a decrease of performance over time. One participant was not able to control both arm and hands for both methods and an other participant was able to control them for the 4 first tasks of the first tested method only.

The success rates, or ratios of successful tasks, for the case in which the participants controlled both robot arm and hand are presented in Table 10.3. We observe that TME outperformed the performance RR by 15% when all the objects and both locations A



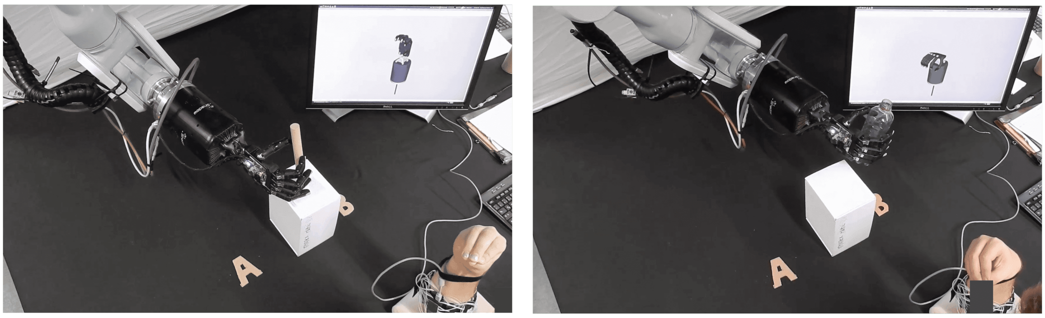
(a) (b)

Figure 10.10 – (*a*) Setup of the teleoperation experiment. The participant is requested to grasp the chips cylinder and to place it on A or B. The distance between the initial position of the object and A or B is approximately 15 cm. The detected activations are displayed on the monitor by an animated hand model. (*b*) Objects used during the teleoperation experiment.

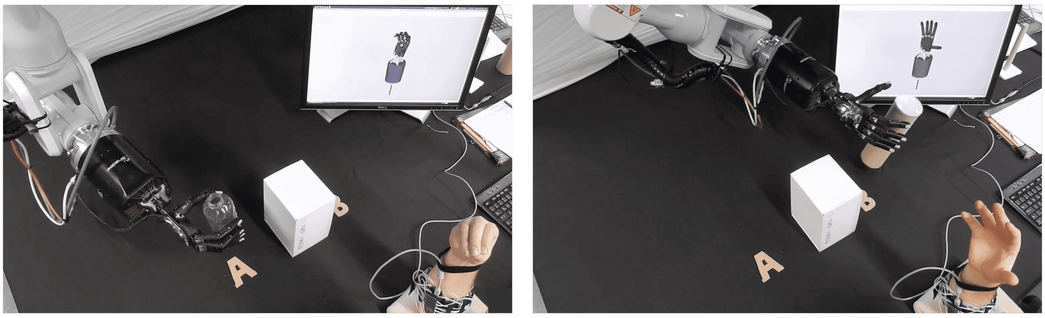
and B are considered. The time needed to accomplish successful tasks were 55.6 ± 31.1 s and 53.9 ± 31.5 s for TME and RR, respectively, showing almost no difference between the two methods.

For both methods, the tasks involving the woodstick result in the lowest success rate. Due to the small diameter of this object, the arm had to be positioned very precisely and a complete grasp activation had to be detected in order to perform a successful grasp. Therefore, the tasks involving the woodstick were the hardest to complete for the participants. In the case of TME, the success rate for the chips cylinder is lower than for the PET bottle. This can be explained by the fact that a small activation of the grasp movement was sometimes detected when the participants were flexing their wrist to make the arm move in the direction of the object. However, the robotic hand had to be completely opened to be able to be placed around the chips cylinder before grasping, while it could still be placed around the bottle if a small grasping activation was detected. In the case of RR, the success rate diminishes for the bottle compared to the chips cylinder. This may suggest a stronger decrease of performance over time with this method.

We observe that the success rates for the bottle are similar with and without the holding assistance activated for both methods. This result is particularly interesting as it shows that combinations of hand and wrist movements in this experiment, namely grasping with wrist flexion, and grasping with wrist extension, can be detected while training only on individual movements. Moreover, both aforementioned combinations were equally detected as the number of completed tasks for each location A or B was similar, i.e., 4 and 3 successful tasks for A and B with TME and 2 for each location with RR. Moreover,



(a) Grasping the woodstick (b) Bringing the bottle to B



(c) Reaching A while holding the bottle (d) Releasing the chips cylinder at B

Figure 10.11 – Snapshots of the teleoperation experiment for different tasks. The detected activations are displayed by a Shadow robot hand model on the monitor.

some of the participant did not wait that the contact with the object was detected before bringing it to its final location. Therefore, they managed to complete other tasks without using the holding assistance.

Table 10.4 shows the proportion of the failed tasks for which the time ran out during each of the task steps, namely grasping, moving and releasing the object, for TME and RR. We observe that the proportions are similar for both methods with the grasping step being the main cause of failure, followed by the moving step. Failures during grasping occurred mainly because the detected grasp activation was not sufficient to grasp the object or when it was activated too soon, therefore resulting in the object being pushed out of the support box. Failures while moving were due to difficulties to detect wrist flexion and extension or to the object falling down while the participant was trying to reach A or B. Finally, if the fingers extension movement was not detected properly while the holding assistance was activated, the opening of the hand was not triggered, resulting in failure to release the object.

The success rates for the case in which the arm commands where disabled and the participant was only requested to grasp the object are presented in Table 10.5. In this case, RR outperforms TME, especially for the tasks involving the woodstick. This can be explained by the fact that, in most of the cases, the arm commands had to be disabled for

Table 10.3 – Success rates over all the tasks and for each object in the case where the participants teleoperated both the robot arm and hand. The success rates are given in percent [%].

	Total	Chips cylinder	Woodstick	Bottle	Bottle (no HA)
TME	45.8	41.7	16.7	66.7	58.3
RR	30.6	44.4	12.5	35.7	28.6

Table 10.4 – Proportion of the failed tasks due to running out of time during the grasp, the displacement and the release of the object [%].

	Grasping	Moving	Releasing
TME	57.7	30.8	11.5
RR	58.1	32.6	9.3

RR because the arm was drifting on the left or on the right, while it had to be disabled for TME because no activation was detected to move the arm, so that the participant could not position it to grasp the object. Generally, the detected activation of the grasp movement was also limited, therefore some tasks were difficult to achieve, especially those involving the woodstick.

10.5 Discussion

The proposed TME model allows the structure of tensor-valued data to be taken into account in the regression problem. Overfitting can then be reduced, which is particularly important when only few tensor-valued training data are available. We showed the effectiveness of the approach to detect hand movements from TMG data, outperforming the other tested methods in an offline experiment and allowing participants to teleoperate a robotic arm and hand in real-time. Particularly, the method was able to successfully detect intermediate and combination of activation, while trained only with zero and complete individual movements. Notably, participants managed to activate wrist flexion or extension along with power grasp. This indicates that a holding assistance may not be required. However, some participants reported that the holding assistance was helpful as a feedback indicating that the grasp was effective or to make them feel more comfortable while teleoperating the arm, as they could focus on one movement only.

A decrease of performance over time, indicated by the necessity of deactivating the arm commands while testing the second method, seemed to have occurred during the real-time experiment. Moreover, some participants reported that they felt that the control of

Table 10.5 – Success rates over all the tasks and for each object, for the case in which the commands of the arm had to be disabled. The success rates are given in percent [%].

	Total	Chips cylinder	Woodstick	Bottle	Bottle (no HA)
TME	38.8	40.0	0	40.0	40.0
RR	53.8	50.0	66.7	50.0	50.0

the robotic arm and hand was harder to perform over time. Moreover, we qualitatively observed that this problem seemed to occur particularly for participants who were trying to apply high forces to execute the different movements. We hypothesize that this is due to small displacements of the TMG bracelet over time, inducing a shift of the testing data compared to the training data. This problem could be overcome by improving the placement of the bracelet and by adapting the model over time. Techniques such as covariate shift adaptation in the case of sEMG [Vidovic et al., 2016] could be investigated.

It is important to emphasize the fact that the participants were able to adapt in some extent to the predictions of the method. They slightly modified their hand movements in order to obtain the desired action of the robotic arm and hand. Therefore, we observe a form of active learning, where the method learned from the training data, while the participants learned from the method in order to achieve the desired performance.

In both experiments, the ranks of the experts were given by a common value. The performance of TME may be further improved by selecting a specific rank for each expert. However, to avoid increasing computation time, automatic rank selection procedures have to be investigated. In particular, the automatic rank selection presented in [Guo et al., 2012] could be exploited to determine the rank of the expert models. The suggested method uses a $\ell_{1,2}$ norm regularization and optimizes the model with iteratively reweighted least squares (IRLS) algorithm. This approach seems promising as the authors reported in their experiments that the automatic procedure provided the same rank as the one selected by cross-validation.

10.6 Conclusion

This chapter presented an extension of mixture of experts to tensor-valued data. Our method brings together the advantages and robustness of mixture models and tensor methods. Therefore, it allows an efficiently combination of predictions from experts specialized in different regions of the input space, while taking into account the structure of tensor-valued data in the soft space division and in the predictions of the experts. By incorporating information on their structure in the model, the data are efficiently exploited, so that a model trained with a small amount of training data is able to achieve a good performance, while overcoming the overfitting problem. This is particularly important in robotics as the amount of training data is often small compared to the dimensionality of the data. The effectiveness of our model was illustrated with artificially generated data and in two experiments aiming at detecting hand movements by measuring the pressure induced by the muscles activity of the forearm with tactile myography. We showed that the testing computational time of the proposed model is low, due to a computational cost independent of the number of training data, therefore making it compatible with real-time robotic applications.

The proposed TME model could be improved by investigating automatic rank selection procedures with the objective of automatically determining all the ranks of the model, therefore avoiding the use of cross-validation in the training process. Moreover, extensions of the proposed TME model could be studied for other applications in robotics. Notably, TME could be extended to more complex models, such as hierarchical TME [Yuksel et al., 2012]. It is worth noting that the proposed TME model permits to incorporate structural information of the data as a special case of neural network. Extensions of this model could then also lead to interesting perspectives in the development of neural network structures for tensor data that would have better interpretability, that could be trained with small amount of data, and that would provide better generalization results by avoiding overfitting.

11 Conclusions and Future Work

This thesis proposed to include domain knowledge into robot learning, control, and optimization algorithms in order to enhance robot learning and control capabilities. In the three first parts of this thesis, we exploited Riemannian manifolds theory in order to design *geometry-aware* learning, control and optimization approaches that efficiently consider the geometry of non-Euclidean parameter spaces, which are ubiquitous in robotics. In the last part, we considered the introduction of domain knowledge from a broader perspective and showed that learning approaches can benefit from being *structure-aware*, where the structure may come either from prior models or from being intrinsically contained in the data. While individual conclusions were drawn for each of the approaches presented in this manuscript, this final chapter aims at highlighting general observations that can be drawn from the proposed models and solutions described throughout the thesis. We then conclude with a discussion on future research directions.

11.1 Conclusions

In Part I, we complemented the LfD paradigm to encode and retrieve data in the form of SPD matrices. By being geometry-aware, the proposed GMM/GMR framework considers meaningful relationships between the data and guarantees the generation of outputs belonging to the desired manifold. In addition, the proposed approach can be straightforwardly used to learn trajectories combining data on several manifolds. For example, the position, orientation and manipulability ellipsoids of a robot during a manipulation task can be jointly learned from demonstrations by considering the product of manifolds $\mathbb{R}^3 \times \mathcal{S}^3 \times \mathcal{S}^6_{++}$. Moreover, the same state variables can easily be encoded for multiple arms systems by considering their shared manipulability, e.g., we may consider the product $\mathbb{R}^3 \times \mathcal{S}^3 \times \mathbb{R}^3 \times \mathcal{S}^3 \times \mathcal{S}^6_{++}$ for a dual-arm manipulation task. It is worth noticing that, in contrast to state-of-the-art approaches, the proposed approach automatically encodes and retrieves full SPD matrices, i.e., including off-diagonals elements. This opens the door to learning complex robot behaviors, e.g., by considering full controller gains matrices instead of diagonal ones or by learning complete manipulability ellipsoids instead of optimizing only the determinant or condition number of these matrices.

The Part II of this thesis focused on a specific category of SPD matrices, namely manipulability ellipsoids. After learning sequences of manipulability in Part I, we exploited Riemannian manifold theory and tensor representations to build manipulability tracking controllers, allowing to reproduce the learned SPD trajectories. Compared to the state-of-the-art manipulability-based optimization schemes, our approach allows the reproduction of any manipulability ellipsoid in an exponentially-stable manner. Based on the aforementioned manipulability learning and tracking frameworks, we presented a complete manipulability transfer framework that enabled the learning of posture-dependent task requirements. The presented approach provided a skill transfer strategy between agents of different embodiments going beyond the imitation of trajectory, force of impedance behaviors.

In Part III, we proposed GaBO, a geometry-aware Bayesian optimization that exploits the geometry of the search space to refine, adapt and optimize parameters lying on Riemannian manifolds. We showed that providing BO with information about the geometry of the parameters leads to faster convergence, better accuracy and lower solution variance compared to geometry-unaware BO implementations. We showed that these findings remain valid and are even accentuated when the dimension of the search space increases, as GaBO even outperforms high-dimensional BO approaches. Finally, we proposed a high-dimension GaBO framework based on the low-dimensional assumption, scaling GaBO towards problems lying on a Riemannian manifold latent space embedded in a high-dimensional Riemannian manifold.

In summary, the three first parts of this thesis prove that geometry-awareness is crucial for successfully learning, controlling and refining robotics parameters lying on Riemannian manifolds, while providing a proper mathematical treatment of these parameters. Our approaches not only guarantee that recovered data lie on the desired parameter space, but lead to better performances in terms of accuracy, stability, convergence and scalability due to their ability in considering meaningful relationships between points in manifold spaces. It is worth noticing that the proposed approaches are task-agnostic in the sense that they are adapted to learn and refine a full range of skills with the only constraint that the manifold or combination of manifolds must be provided to the different algorithms according to the task specifications.

In Part IV, we proposed to introduce domain knowledge by leveraging structure-awareness in two learning algorithms. We first presented a new class of multi-output GPs with non-stationary prior mean and kernel based on GMR. We showed that incorporating a prior model into a GP allowed the resulting approach to benefit from the properties of both the prior model and the GP architecture. Then, we introduced an extension of mixture of experts to tensor-valued data. We showed that, by incorporating information on their structure into the mixture-of-expert model, the data are efficiently exploited. Therefore, a structure-aware model trained with a small amount of data is able to overcome the overfitting problem and to achieve better performance than the corresponding structure-unaware approach.

Overall, the different problems and applications considered throughout this thesis have been shown to greatly benefit from the introduction of domain knowledge into the corresponding algorithms. Incorporating supplementary information on the data into learning, control and optimization approaches generally leads to a better exploitation of these data and therefore improve the performances of the algorithms. While geometry- and structure-aware approaches generally comes with a higher computational cost than their unaware counterparts, this extra computational burden was not an issue in the applications considered in this thesis. Moreover, approximations can be made to fasten some of the proposed algorithms, e.g., simpler forms of retraction approximating the exponential maps may be exploited [Absil et al., 2007b].

11.2 Future Work

In this closing section, we discuss some of the potential research directions based on the contributions of this thesis.

Extensions of the Manipulability Analysis and Transfer Framework

A consequent part of this thesis treated the analysis, learning, tracking and transfer of manipulability ellipsoids. Although only the position-related parts of the manipulability were used in most of our examples, all the presented methods are valid to treat either the orientation-related part or the full manipulability ellipsoids. Following this line, an interesting extension to the manipulability-based analysis presented in Chapter 4 consists in similarly analyzing the role of the orientation-related part — along with its evolution with respect to the position-related part — of the human manipulability in different manipulation tasks. Moreover, the presented manipulability analysis and transfer framework could also be applied in a variety of other human motions, e.g., for locomotions patterns or for in-hand manipulation of objects. The latter could also benefit to the field of prosthetic hands, in which a notion of manipulability could be implemented in order to facilitate graping and manipulation motions for the user. Finally, an other interesting direction of research arises from the observation that human manipulability ellipsoids may be analyzed from a planning point of view. As discussed in Chapter 4, the manipulability ellipsoids observed in between two actions indicate a posture adaptation that anticipates the next action. Therefore, human manipulability ellipsoids may be exploited in human-robot collaboration scenarios, where the robot may anticipate the actions of the human and adapt its behavior consequently. Moreover, based on the human-to-robot transfer experiments of Chapter 6, it would be interesting to further investigate how robot manipulability may be exploited in a planning framework, i.e., for a robot to adapt its posture in anticipation of the next action.

Extensions of GaBO

The GaBO framework presented in Chapter 7 endows BO with the ability to efficiently refine and optimize parameters belonging to Riemannian manifolds. As discussed in Chapter 7, GaBO may be used to optimize simultaneously several parameters belonging to different manifolds by defining the search space as a product of manifolds. In LfD, demonstrated trajectories are often encoded via a GMM. In such cases, it may be desirable to refine the learned model in order to adapt to changes in the task or in the environment. To do so, Rozo [2019] proposed to refine the means of the learned task GMM according to user's guidance, provided via interaction forces. In this context, GaBO could be used to refine not only the means, but the covariances matrices of the Gaussians components and potentially a full GMM. Moreover, when user's guidance is not available, an interesting direction of research consists in the development of a model able to decide which components of the GMM should be updated and when. Going one step further, the developed model could also decide to add or remove components when required. Therefore, future work will investigate extensions of the GaBO framework to automatically and efficiently refine and optimize GMM models. Note that this includes not only GMMs defined in Euclidean spaces, e.g., encoding position-based trajectories, where the component means belong to $\mathbb{R}^d$ and their covariances to $\mathcal{S}_{++}^d$, but also GMMs on Riemannian manifolds, e.g., encoding manipulability-based trajectories, where the component means belong to $\mathcal{S}_{++}^d$ and their covariances to $\mathrm{Sym}^{d\times d\times d\times d}$. Due to the number of parameters to optimize in a GMM, this problem is intrinsically linked with high-dimensional Bayesian optimization. In particular, two interesting research directions may be followed in that field to complement and extend the approaches presented in Chapters 7 and 8. First, heterogeneous objective functions may not be efficiently represented by a global surrogate model. Therefore, we may take inspiration of the approach proposed in [Eriksson et al., 2019] and design local surrogate models for GaBO. Secondly, the acquisition functions may sometimes be sharp (i.e, flat with some few peaks) when only few data are observed, especially for high-dimensional parameter spaces. Therefore, future work may investigate solutions to this problem in the context of GaBO. To do so, an interesting starting point would be the work presented in [Rana et al., 2017b], which proposed to use elastic GPs to improve the optimization of acquisition functions in Euclidean space.

Towards Manifold Learning

While the geometry of the non-Euclidean spaces considered in this thesis was known, many applications involve non-Euclidean parameters spaces whose geometry is not directly available. As an example, the trajectory of a robot in a cluttered environment corresponds to a subset of the Euclidean space $\mathbb{R}^3$, which does not include the space covered by the obstacles, as the end-effector cannot, and should not, access this part of the space. To handle such cases, a possibility in to learn the corresponding manifold from the collected data [Hauberg et al., 2012; Arvanitidis et al., 2019]. Such approaches may potentially be used to encode sensory information collected by the robot, to describe the space of a task that the robot should learn to execute, or to meaningfully measure the difference between two possible robot trajectories. Therefore, there is a clear interest for coupling robot learning algorithms with Riemannian manifold learning approaches.

A Appendices to Chapter 4

A.1 Identification Method for the Kinematics of an Anthropomorphic Arm

The computation of the human arm manipulability is based on the identification method for anthropomorphic arm kinematics proposed in [Ding and Fang, 2013; Fang and Ding, 2013]. This appendix provides the formulas of the forward and inverse kinematic mappings based on the human arm triangle model depicted in Figure 4.2.

Algorithm 9: Inverse mapping from task space to human arm triangle space

```
Input: Previous and current positions $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$, current orientation $\boldsymbol{R}^{\boldsymbol{x}}$
Output: Human arm triangle model parameters $\{\boldsymbol{r}, \boldsymbol{l}, \alpha, \boldsymbol{f}, \boldsymbol{p}\}$
Compute $\alpha = \frac{\arccos(l_{se}^2 + l_{ew}^2 - \|\boldsymbol{x}\|^2)}{2 l_{se} l_{ew}}$;
Compute $\boldsymbol{l}$:
    if $\arccos\left(\frac{\tilde{x}_1}{\sqrt{\tilde{x}_1^2+\tilde{x}_2^2}}\right) > \arccos\left(\frac{x_1}{\sqrt{x_1^2+x_2^2}}\right)$ then
        $\boldsymbol{l} = \frac{\tilde{\boldsymbol{x}} \times \boldsymbol{x}}{\|\tilde{\boldsymbol{x}} \times \boldsymbol{x}\|}$;
    else if $\arccos\left(\frac{\tilde{x}_1}{\sqrt{\tilde{x}_1^2+\tilde{x}_2^2}}\right) < \arccos\left(\frac{x_1}{\sqrt{x_1^2+x_2^2}}\right)$ then
        $\boldsymbol{l} = \frac{\boldsymbol{x} \times \tilde{\boldsymbol{x}}}{\|\boldsymbol{x} \times \tilde{\boldsymbol{x}}\|}$;
    else
        if $\arctan\left(\frac{\tilde{x}_3}{\sqrt{\tilde{x}_1^2+\tilde{x}_2^2}}\right) > \arctan\left(\frac{x_3}{\sqrt{x_1^2+x_2^2}}\right)$ then
            $\boldsymbol{l} = \frac{\tilde{\boldsymbol{x}} \times \boldsymbol{x}}{\|\tilde{\boldsymbol{x}} \times \boldsymbol{x}\|}$;
        else if $\arctan\left(\frac{\tilde{x}_3}{\sqrt{\tilde{x}_1^2+\tilde{x}_2^2}}\right) < \arctan\left(\frac{x_3}{\sqrt{x_1^2+x_2^2}}\right)$ then
            $\boldsymbol{l} = \frac{\boldsymbol{x} \times \tilde{\boldsymbol{x}}}{\|\boldsymbol{x} \times \tilde{\boldsymbol{x}}\|}$;
        else
            $\boldsymbol{l}$ is an arbitrary unit vector;
Compute $\boldsymbol{r} = \left(\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}, \ \boldsymbol{l} \times \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}, \ \boldsymbol{l}\right) \boldsymbol{R}_3(\delta) \left(\begin{smallmatrix}1\\0\\0\end{smallmatrix}\right)$ with $\delta = \arccos\left(\frac{\|\boldsymbol{x}\|^2 + l_{se}^2 + l_{ew}^2}{2 l_{se} \|\boldsymbol{x}\|}\right)$;
Extract $\boldsymbol{f}$ and $\boldsymbol{p}$ from $\boldsymbol{R}^{\boldsymbol{x}} = \left(\boldsymbol{f}, \ \boldsymbol{p} \times \boldsymbol{f}, \ \boldsymbol{p}\right)$
```

In order to compute the manipulability of the human arm, we first need to infer the joint angles of the anthropomorphic arm model in function of the position and orientation of the wrist in task space, which in our case is given in the database [Maurice et al., 2019]. The corresponding inverse kinematic mapping is computed with Algorithms 9 and 10. The inverse mapping $\{\tilde{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{R}^{\boldsymbol{x}}\} \rightarrow \{\boldsymbol{r}, \boldsymbol{l}, \alpha, \boldsymbol{f}, \boldsymbol{p}\}$ from the task space to the human arm triangle space is first computed with Algorithm 9, where $\tilde{\boldsymbol{x}}$ and $\boldsymbol{x}$ are the previous and current wrist positions, respectively, and $\boldsymbol{R}^{\boldsymbol{x}}$ is the rotation matrix describing the orientation of the wrist in the task space.

Algorithm 10: Inverse mapping from human arm triangle space to joint space

```
Input: Human arm triangle model parameters $\{\boldsymbol{r}, \boldsymbol{l}, \alpha, \boldsymbol{f}, \boldsymbol{p}\}$
Output: Joint angles $\{q_1 \dots q_7\}$
Compute $q_1 = \arccos\left(\frac{-r_2}{\sqrt{r_2^2+r_3^2}}\right)$;
    if $-r_3 < 0$ then
        $q_1 \leftarrow 2\pi - q_1$;
    end
Compute $q_2 = -\arccos\left(\frac{s_1}{\sqrt{s_1^2+s_2^2}}\right)$ with $\boldsymbol{s} = \boldsymbol{R}_1^{\mathsf{T}}(q_1)\boldsymbol{r}$;
Compute $q_3$:
    $\Delta q_3 = \arccos(\boldsymbol{t}^{\mathsf{T}}\boldsymbol{l})$ with $\boldsymbol{t} = \boldsymbol{R}_1(q_1)\begin{pmatrix}-\sin(-q_2)\\ -\cos(-q_2)\\ 0\end{pmatrix}$;
    if $(\boldsymbol{r} \times \boldsymbol{t})^{\mathsf{T}}\boldsymbol{l} \geq 0$ then
        $q_3 = \frac{\pi}{2} + \Delta q_3$;
    else
        $q_3 = \frac{\pi}{2} - \Delta q_3$;
Compute $q_4 = \alpha - \pi$;
Compute $\boldsymbol{f}$ and $\boldsymbol{p}$ in the wrist frame:
    $\boldsymbol{f}' = \boldsymbol{R}^{\boldsymbol{w}\mathsf{T}}\boldsymbol{f}$, $\boldsymbol{p}' = \boldsymbol{R}^{\boldsymbol{w}\mathsf{T}}\boldsymbol{p}$ where $\boldsymbol{R}^{\boldsymbol{w}} = \begin{pmatrix}\boldsymbol{w}_1, & \boldsymbol{w}_2, & \boldsymbol{w}_3\end{pmatrix}$
    with $\boldsymbol{w}_1 = \begin{pmatrix}\boldsymbol{r}, & \boldsymbol{l} \times \boldsymbol{r}, & \boldsymbol{l}\end{pmatrix}\boldsymbol{R}_3(\alpha - \pi)\begin{pmatrix}1\\ 0\\ 0\end{pmatrix}$, $\boldsymbol{w}_3 = \boldsymbol{l}$ and $\boldsymbol{w}_2 = \boldsymbol{w}_3 \times \boldsymbol{w}_1$ ;
Compute $q_5 = \arccos\left(\frac{-p'_2}{\sqrt{{p'_2}^2+{p'_3}^2}}\right)$;
    if $p'_3 > 0$ then
        $q_5 \leftarrow 2\pi - q_5$;
    end
Compute $q_6 = \arccos\left(\frac{-p'_1}{\|\boldsymbol{p}'\|}\right)$;
Compute $q_7 = \arccos\left(\begin{pmatrix}\sin(q_6)\\ -\cos(q_6)\cos(q_5)\\ -\cos(q_6)\sin(q_5)\end{pmatrix}^{\mathsf{T}}\boldsymbol{f}'\right)$;
    if $\left(-\boldsymbol{p}' \times \begin{pmatrix}\sin(q_6)\\ -\cos(q_6)\cos(q_5)\\ -\cos(q_6)\sin(q_5)\end{pmatrix}\right)^{\mathsf{T}}\boldsymbol{f}' < 0$ then
        $q_7 \leftarrow -q_7$;
    end
```

Then the inverse mapping $\{\boldsymbol{r}, \boldsymbol{l}, \alpha, \boldsymbol{f}, \boldsymbol{p}\} \rightarrow \{q_1 \dots q_7\}$ from the human arm triangle space to the joint space is computed with Algorithm 10. In the two aforementioned algorithms, l_{se} and l_{ew} denote the human arm length from the shoulder to the elbow and from the elbow to the wrist, respectively. Moreover, the rotation matrix $\boldsymbol{R}_i(\delta)$ corresponds to a rotation of an angle δ around the i-th axis of coordinate.

Algorithm 11 presents the forward mapping $\boldsymbol{q} \rightarrow \{\boldsymbol{x}, \boldsymbol{R}^{\boldsymbol{x}}\}$ from the joint space to the task space, that may be used to compute the human arm Jacobian. We refer the interested reader to [Ding and Fang, 2013; Fang and Ding, 2013] for the complete development of the formulas presented in this appendix.

```
Algorithm 11: Forward mapping from joint space to task space
  Input: Joint angles $\{q_1 \dots q_7\}$
  Output: Wrist position $\boldsymbol{x}$ and orientation $\boldsymbol{R}^{\boldsymbol{x}}$
  Human arm triangle model parameters:
      Compute $\alpha = \pi + q_4$ ;
      Compute $\boldsymbol{l} = \begin{pmatrix} \sin(q_2)\sin(q_3) \\ -\cos(q_1)\cos(q_2)\sin(q_3) - \sin(q_1)\cos(q_3) \\ -\sin(q_1)\cos(q_2)\sin(q_3) + \cos(q_1)\cos(q_3) \end{pmatrix}$;
      Compute $\boldsymbol{r} = \begin{pmatrix} \cos(q_2) \\ \cos(q_1)\sin(q_2) \\ \sin(q_1)\sin(q_2) \end{pmatrix}$;
      Compute $\boldsymbol{p} = \boldsymbol{R}^{\boldsymbol{w}}\boldsymbol{p}'$ with $\boldsymbol{p}' = \begin{pmatrix} -\cos(q_6) \\ -\sin(q_6)\cos(q_5) \\ -\sin(q_6)\sin(q_5) \end{pmatrix}$ ;
      Compute $\boldsymbol{f} = \boldsymbol{R}^{\boldsymbol{w}}\boldsymbol{f}'$ with
      $\boldsymbol{f}' = \left( \begin{pmatrix} \sin(q_6) \\ -\cos(q_6)\cos(q_5) \\ -\cos(q_6)\sin(q_5) \end{pmatrix}, \quad -\boldsymbol{p}' \times \begin{pmatrix} \sin(q_6) \\ -\cos(q_6)\cos(q_5) \\ -\cos(q_6)\sin(q_5) \end{pmatrix}, \quad -\boldsymbol{p}' \right) \boldsymbol{R}_3(q_7) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ ;
  Task space parameters:
      Compute $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{r}, & \boldsymbol{l} \times \boldsymbol{r}, & \boldsymbol{l} \end{pmatrix} \begin{pmatrix} l_{ew} + l_{se}\cos(\pi + \alpha) \\ -l_{se}\sin(\pi - \alpha) \\ 0 \end{pmatrix}$;
      Compute $\boldsymbol{R}^{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{f}, & \boldsymbol{p} \times \boldsymbol{f}, & \boldsymbol{p} \end{pmatrix}$ ;
```

B Appendices to Chapter 5

B.1 Symbolic Manipulability Jacobian for a Serial Kinematic Chain

The computation of the manipulability Jacobian involves computing the derivative of the robot Jacobian w.r.t. the joint angles. Those derivatives can be computed in a symbolic form as shown in [Bruyninckx and De Schutter, 1996]. We remind here the symbolic derivative for the hybrid representation of the Jacobian $\boldsymbol{J} \in \mathbb{R}^{6\times n}$ that is used in the computation of the manipulability Jacobian $\boldsymbol{\mathcal{J}}$.

The i-th column of the Jacobian is denoted by

$$\boldsymbol{J}^i = \begin{pmatrix} \boldsymbol{w}^i \\ \boldsymbol{v}^i \end{pmatrix}, \tag{B.1}$$

with $\boldsymbol{w}^i \in \mathbb{R}^3$ and $\boldsymbol{v}^i \in \mathbb{R}^3$ the rotational and translational components of the Jacobian.

The derivative of the Jacobian w.r.t. the joint angles is a third order tensor $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \in \mathbb{R}^{6\times n\times n}$ with mode-1 fibers or columns

$$\left(\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}}\right)_{:ij} = \frac{\partial \boldsymbol{J}^i}{\partial q^j} = \begin{cases} \boldsymbol{P}_\Delta(\boldsymbol{J}^j)\boldsymbol{J}^i & \text{if } j \leq i \\ -\boldsymbol{M}_\Delta(\boldsymbol{J}^j)\boldsymbol{J}^i & \text{if } j > i \end{cases}, \tag{B.2}$$

where

$$\boldsymbol{P}_\Delta(\boldsymbol{J}^j) = \begin{pmatrix} [\boldsymbol{w}^j \times] & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & [\boldsymbol{w}^j \times] \end{pmatrix}, \tag{B.3}$$

$$\boldsymbol{M}_\Delta(\boldsymbol{J}^j) = \begin{pmatrix} \boldsymbol{0}_{3\times 3} & \boldsymbol{0}_{3\times 3} \\ [\boldsymbol{v}^j \times] & \boldsymbol{0}_{3\times 3} \end{pmatrix}, \tag{B.4}$$

and × the cross product between two vectors. The notation $[\boldsymbol{w}^j\times]$ in a matrix denotes that the corresponding component of the result of the right-multiplication of the matrix by a vector is equal to the cross product between $\boldsymbol{w}^j$ and the corresponding vector component, e.g. $\boldsymbol{P}_\Delta(\boldsymbol{J}^j)\boldsymbol{J}^i = \begin{pmatrix} \boldsymbol{w}^j \times \boldsymbol{w}^i \\ \boldsymbol{w}^j \times \boldsymbol{v}^i \end{pmatrix}$.

Note that the time derivative of the Jacobian can therefore be computed as

$$\frac{d\boldsymbol{J}}{dt} = \sum_{j=1}^{n} \frac{\partial \boldsymbol{J}}{\partial q^j} \dot{q}_j. \tag{B.5}$$

B.2 Symbolic Dynamic Manipulability Jacobian for a Serial Kinematic Chain

The derivative of the robot inertia matrix w.r.t. joint angles is necessary for the computation of the dynamic manipulability Jacobian. It can be computed in closed form as follows.

The inertia matrix $\boldsymbol{\Lambda}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ can be written as

$$\boldsymbol{\Lambda}(\boldsymbol{q}) = \sum_{i=1}^{n} \boldsymbol{J}_i^{\mathsf{T}} \begin{pmatrix} \boldsymbol{\Lambda}_i & \boldsymbol{0} \\ \boldsymbol{0} & m_i \boldsymbol{I} \end{pmatrix} \boldsymbol{J}_i, \tag{B.6}$$

where $\boldsymbol{J}_i$, $\boldsymbol{\Lambda}_i$ and m_i are the Jacobian, inertia matrix and mass of link i, respectively [Park, 1995; Murray et al., 1994].

The derivative of the inertia matrix is the third order tensor $\frac{\delta \boldsymbol{\Lambda}}{\delta \boldsymbol{q}} \in \mathbb{R}^{n\times n\times n}$ computed as

$$\frac{\delta \boldsymbol{\Lambda}}{\delta \boldsymbol{q}} = \sum_{i=1}^{n} \frac{\partial \boldsymbol{J}_i^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{J}_i^{\mathsf{T}} \boldsymbol{M}_i + \frac{\partial \boldsymbol{J}_i}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J}_i^{\mathsf{T}} \boldsymbol{M}_i, \tag{B.7}$$

where $\boldsymbol{M}_i = \begin{pmatrix} \boldsymbol{\Lambda}_i & \boldsymbol{0} \\ \boldsymbol{0} & m_i \boldsymbol{I} \end{pmatrix}$ and $\frac{\partial \boldsymbol{J}_i}{\partial \boldsymbol{q}}$ is computed with Eq. (B.2).

B.3 Symbolic Derivative of the Manipulability Jacobian for a Serial Kinematic Chain

In some cases, e.g. in the acceleration tracking controller, the time derivative of the manipulability Jacobian is required. This time derivative can be computed symbolically by exploiting the first and second derivative of the Jacobian w.r.t. the joint angles.

The time derivative of the velocity manipulability Jacobian $\boldsymbol{\mathcal{J}}^{\dot{\boldsymbol{x}}} \in \mathbb{R}^{6\times n\times n}$ defined as

$$\boldsymbol{\mathcal{J}}^{\dot{\boldsymbol{x}}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{J} + \frac{\partial \boldsymbol{J}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J}, \tag{B.8}$$

is obtained by exploiting the chain rule as

$$\frac{\partial \boldsymbol{\mathcal{J}}^{\dot{\boldsymbol{x}}}}{\partial t} = \frac{\partial}{\partial t}\left(\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \boldsymbol{J} + \frac{\partial \boldsymbol{J}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \boldsymbol{J}\right) \tag{B.9}$$

$$= \frac{\partial^2 \boldsymbol{J}}{\partial t \partial \boldsymbol{q}} \times_2 \boldsymbol{J} + \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \times_2 \frac{\partial \boldsymbol{J}}{\partial t} + \frac{\partial^2 \boldsymbol{J}^{\mathsf{T}}}{\partial t \partial \boldsymbol{q}} \times_1 \boldsymbol{J} + \frac{\partial \boldsymbol{J}^{\mathsf{T}}}{\partial \boldsymbol{q}} \times_1 \frac{\partial \boldsymbol{J}}{\partial t}. \tag{B.10}$$

The time derivative of the Jacobian is given by Eq. (B.5) and the time derivative of the derivative of the Jacobian w.r.t. joint angles is given by

$$\frac{\partial^2 \boldsymbol{J}}{\partial t \partial q_j} = \sum_{k=1}^{n} \frac{\partial^2 \boldsymbol{J}}{\partial q_k \partial q_j} \dot{q}_k, \tag{B.11}$$

where the second derivative of the Jacobian w.r.t. the joint angles is a fourth order tensor $\frac{\partial^2 \boldsymbol{J}}{\partial \boldsymbol{q}^2} \in \mathbb{R}^{6 \times n \times n \times n}$ with mode-1 fibers or columns

$$\left(\frac{\partial^2 \boldsymbol{J}}{\partial \boldsymbol{q}^2}\right)_{:ijk} = \frac{\partial^2 \boldsymbol{J}^i}{\partial q^k \partial q^j} = \begin{cases} \Big(\boldsymbol{P}_\Delta(\boldsymbol{J}^j)\boldsymbol{P}_\Delta(\boldsymbol{J}^k)\Big)\boldsymbol{J}^i + \boldsymbol{P}_\Delta(\boldsymbol{J}^j)\Big(\boldsymbol{P}_\Delta(\boldsymbol{J}^k)\boldsymbol{J}^i\Big) & \text{if } k \leq j \leq i \\ \boldsymbol{P}_\Delta(\boldsymbol{J}^j)\Big(\boldsymbol{P}_\Delta(\boldsymbol{J}^k)\boldsymbol{J}^i\Big) & \text{if } j \leq k \leq i \\ -\boldsymbol{P}_\Delta(\boldsymbol{J}^j)\Big(\boldsymbol{M}_\Delta(\boldsymbol{J}^k)\boldsymbol{J}^i\Big) & \text{if } j \leq i < k \\ -\Big(\boldsymbol{P}_\Delta(\boldsymbol{J}^k)\boldsymbol{M}_\Delta(\boldsymbol{J}^j)\Big)\boldsymbol{J}^i - \boldsymbol{M}_\Delta(\boldsymbol{J}^j)\Big(\boldsymbol{P}_\Delta(\boldsymbol{J}^k)\boldsymbol{J}^i\Big) & \text{if } k \leq i < j \\ -\Big(\boldsymbol{P}_\Delta(\boldsymbol{J}^k)\boldsymbol{M}_\Delta(\boldsymbol{J}^j)\Big)\boldsymbol{J}^i & \text{if } i < k < j \\ -\Big(\boldsymbol{P}_\Delta(\boldsymbol{J}^j)\boldsymbol{M}_\Delta(\boldsymbol{J}^k)\Big)\boldsymbol{J}^i & \text{if } i < j \leq k \end{cases} \tag{B.12}$$

where $\boldsymbol{P}_\Delta(\boldsymbol{J}^j)$ and $\boldsymbol{M}_\Delta(\boldsymbol{J}^k)$ are defined as in (B.3) and (B.4), respectively. The time derivative of the force manipulability Jacobian $\boldsymbol{\mathcal{J}}^{\boldsymbol{F}}$ and the manipulability Jacobian $\boldsymbol{\mathcal{J}}^{\ddot{\boldsymbol{x}}}$ corresponding to the dynamic manipulability ellipsoid can be computed symbolically in a similar way using Eqs. (B.5) and (B.11). Moreover, their derivative w.r.t. joint angles can be computed symbolically using the chain rules, Eqs. (B.2) and (B.12).

C Appendices to Chapter 8

C.1 Distances between Points on Nested Spheres

The geometry-aware mGP used in HD-GaBO involves the computation of kernel functions based on distances between data projected onto nested Riemannian manifolds with the projection mapping $m : \mathcal{S}^D \mapsto \mathcal{S}^d$ (8.7). We compute here the distance between projected data on nested spheres and show that this distance is invariant to the parameters $\{r_D, \dots r_{d+1}\}$.

To do so, we first compute the distance $d_{\mathcal{S}^{D-1}}(m_D(\boldsymbol{x}_i), m_D(\boldsymbol{x}_j))$ between two points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{S}^D$ projected onto $\mathcal{S}^{D-1}$. Given an axis $\boldsymbol{v}_D \in \mathcal{S}^D$ and a distance $r_D \in \,]0, \pi/2]$, the projection mapping $m_D : \mathcal{S}^D \mapsto \mathcal{S}^{D-1}$ is computed as (8.6)

$$\boldsymbol{z} = m_D(\boldsymbol{x}) = \underbrace{\frac{1}{\sin(r_D)}}_{\text{scaling}} \underbrace{\boldsymbol{I}_{\text{trunc}}\boldsymbol{R}}_{\text{dim. red. + rot.}} \underbrace{\left(\frac{\sin(r_D)\boldsymbol{x} + \sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x}) - r_D\Big)\boldsymbol{v}_D}{\sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x})\Big)}\right)}_{\text{projection onto } \mathcal{A}^{D-1}}. \tag{C.1}$$

By exploiting the identity

$$\sin(\alpha - \beta) = \sin(\alpha)\cos(\beta) - \cos(\alpha)\sin(\beta), \tag{C.2}$$

and the distance formula $d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x}) = \arccos(\boldsymbol{v}_D^\mathsf{T}\boldsymbol{x})$, we can further rewrite (C.1) as

$$\boldsymbol{z} = m_D(\boldsymbol{x}) = \underbrace{\frac{1}{\sin(r_D)}}_{\text{scaling}} \underbrace{\boldsymbol{I}_{\text{trunc}}\boldsymbol{R}}_{\text{dim. red. + rot.}} \underbrace{\left(\frac{\sin(r_D)}{\sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x})\Big)}(\boldsymbol{x} + \boldsymbol{v}_D^\mathsf{T}\boldsymbol{x}\boldsymbol{v}_D) + \cos(r_D)\boldsymbol{v}_{\boldsymbol{D}}\right)}_{\text{projection onto } \mathcal{A}^{D-1}}. \tag{C.3}$$

The distance $d_{\mathcal{S}^{D-1}}\Big(m_D(\boldsymbol{x}_i), m_D(\boldsymbol{x}_j)\Big)$ is given by

$$d_{\mathcal{S}^{D-1}}\Big(m_D(\boldsymbol{x}_i), m_D(\boldsymbol{x}_j)\Big) = d_{\mathcal{S}^{D-1}}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \arccos(\boldsymbol{z}_i^{\mathsf{T}}\boldsymbol{z}_i). \tag{C.4}$$

By defining the projection onto $\mathcal{A}^{D-1}$ as the function $\boldsymbol{z} = p(\boldsymbol{x})$, we can compute

$$\begin{aligned}
\boldsymbol{z}_i^{\mathsf{T}}\boldsymbol{z}_j &= \frac{1}{\sin^2(r_D)} p(\boldsymbol{x}_i)^{\mathsf{T}} \boldsymbol{R}^{\mathsf{T}} \boldsymbol{I}_{\text{trunc}}^{\mathsf{T}} \boldsymbol{I}_{\text{trunc}} \boldsymbol{R}\, p(\boldsymbol{x}_j), && \text{(C.5)}\\
&= \frac{1}{\sin^2(r_D)}\Big(p(\boldsymbol{x}_i)^{\mathsf{T}} \boldsymbol{R}^{\mathsf{T}} \boldsymbol{R}\, p(\boldsymbol{x}_j) - \cos^2(r_D)\Big), && \text{(C.6)}\\
&= \frac{1}{\sin^2(r_D)}\Big(p(\boldsymbol{x}_i)^{\mathsf{T}} p(\boldsymbol{x}_j) - \cos^2(r_D)\Big), && \text{(C.7)}\\
&= \frac{1}{\sin^2(r_D)}\left(\frac{\sin^2(r_D)\Big(\boldsymbol{x}_i - \boldsymbol{v}_D^{\mathsf{T}}\boldsymbol{x}_i\boldsymbol{v}_D\Big)^{\mathsf{T}}\Big(\boldsymbol{x}_j - \boldsymbol{v}_D^{\mathsf{T}}\boldsymbol{x}_j\boldsymbol{v}_D\Big)}{\sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x}_i)\Big)\sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x}_j)\Big)} + \cos^2(r_D)\boldsymbol{v}_D^{\mathsf{T}}\boldsymbol{v}_D - \cos^2(r_D)\right), && \text{(C.8)}\\
&= \frac{\Big(\boldsymbol{x}_i - \boldsymbol{v}_D^{\mathsf{T}}\boldsymbol{x}_i\boldsymbol{v}_D\Big)^{\mathsf{T}}\Big(\boldsymbol{x}_j - \boldsymbol{v}_D^{\mathsf{T}}\boldsymbol{x}_j\boldsymbol{v}_D\Big)}{\sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x}_i)\Big)\sin\Big(d_{\mathcal{S}^D}(\boldsymbol{v}_D, \boldsymbol{x}_j)\Big)}, && \text{(C.9)}
\end{aligned}$$

so that $\boldsymbol{z}_i^{\mathsf{T}}\boldsymbol{z}_j$, and thus the distance (C.4), are invariant w.r.t r_D. Note that Equation (C.6) was obtained by using the fact that the last coordinate of the projections $\boldsymbol{R}\, p(\boldsymbol{x}_i)$ and $\boldsymbol{R}\, p(\boldsymbol{x}_j)$ is equal to $\cos(r_D)$ from the nested sphere mapping definition. We then used the rotation matrix property $\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R} = \boldsymbol{I}$ to obtain (C.7) and the unit norm property of $\boldsymbol{v}_D$, so that $\boldsymbol{v}_D^{\mathsf{T}}\boldsymbol{v}_D = 1$ to obtain (C.9).

As the distance (C.4) is invariant w.r.t. r_D for any dimension D and as the mapping m is a composition of successive mappings m_D, we can straightforwardly conclude that the distance $d_{\mathcal{S}^d}\Big(m(\boldsymbol{x}_i), m(\boldsymbol{x}_j)\Big)$ with $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{S}^D$ and $d \leq D$ is invariant w.r.t. the parameters $\{r_D, \ldots r_{d+1}\}$.

Bibliography

Absil, P.-A., Baker, C. G., and Gallivan, K. A. (2007a). Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7:303–330.

Absil, P. A., Mahony, R., and Sepulchre, R. (2007b). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press.

Ajoudani, A., Tsagarakis, N. G., and Bicchi, A. (2015). On the role of robot configuration in cartesian stiffness control. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1010–1016.

Alvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266.

Antonova, R., Rai, A., and Atkeson, C. (2017). Deep kernels for optimizing locomotion controllers. In *Conference on Robot Learning (CoRL)*, pages 47–56.

Antonova, R., Rai, A., Li, T., and Kragic, D. (2019). Bayesian optimization in variational latent spaces with dynamic compression. In *Conference on Robot Learning (CoRL)*.

Arnaudon, M., Barbaresco, F., and Yang, L. (2011). Medians and means in Riemannian geometry : existence , uniqueness and computation. In *Matrix Information Geometry*, pages 169–197. Springer Berlin Heidelberg.

Arsigny, V., Fillard, P., Pennec, X., and Ayache, N. (2006). Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Barath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34:26–38.

Arvanitidis, G., Hauberg, S., Hennig, P., and Schober, M. (2019). Fast and robust shortest paths on manifolds learned from data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1506–1515.

Azad, M., Babič, J., and Mistry, M. (2017). Dynamic manipulability of the center of mass: A tool to study, analyse and measure physical ability of robots. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3484–3490.

Baccelliere, L., Kashiri, N., Muratore, L., Laurenzi, A., Kamedula, M., Margan, A., Cordasco, S., Malzahn, J., and Tsagarakis, N. G. (2017). Development of a human size and strength compliant bi-manual platform for realistic heavy manipulation tasks. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5594–5601, Vancouver, Canada.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2019). BoTorch: Programmable Bayesian optimization in PyTorch. *arXiv preprint 1910.06403*.

Barachant, A., Bonnet, S., Congedo, M., and Jutten, C. (2012). Multiclass brain-computer interface classification by Riemannian geometry. *IEEE Trans. on Biomedical Engineering*, 59(4):920–928.

Barfoot, T. D. and Furgale, P. T. (2014). Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Transactions on Robotics*, 30(3):679–693.

Basser, P. and Pajevic, S. (2007). Spectral decomposition of a 4th-order covariance tensor: Applications to diffusion tensor MRI. *Signal Processing*, 87(2):220–236.

Bhatia, R. (2007). *Positive definite matrices*. Princeton University Press.

Bhattacharya, S., Ghrist, R., and Kumar, V. (2014). Multi-robot coverage and exploration on Riemannian manifolds with boundaries. *Intl. Journal of Robotics Research*, 33(1):113–137.

Biess, A., Flash, T., and Liebermann, D. G. (2011). Riemannian geometric approach to human arm dynamics, movement optimization, and invariance. *Physical Review E*, 83(3).

Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Robot programming by demonstration. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, pages 1371–1394. Springer, Secaucus, NJ, USA.

Binois, M., Ginsbourger, D., and Roustant, O. (2020). On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of Global Optimization*, 76(1):69–90.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bonnabel, S. (2013). Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229.

Bonnabel, S. and Sepulchre, R. (2010). Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1055–1070.

Borckmans, P. B., Ishteva, M., and Absil, P.-A. (2010). A modified particle swarm optimization algorithm for the best low multilinear rank approximation of higher-order tensors. In *Swarm Intelligence*, pages 13–23.

Boumal, N. (2015). Riemannian trust regions with finite-difference Hessian approximations are globally convergent. In *Geometric Science of Information (GSI)*, pages 467–475.

Bruyninckx, H. and De Schutter, J. (1996). Symbolic differentiation of the velocity mapping for a serial kinematic chain. *Mechanism and Machine Theory*, 31(2):135–148.

Bures, D. (1969). An extension of Kakutani's theorem on infinite product measures to the tensor product of semifinite w^*-algebras. *Transactions of the American Mathematical Society*, 135:199–212.

Byrd, R. H., Schnabel, R. B., and Shultz, G. A. (1987). A trust region algorithm for nonlinearly constrained optimization. *SIAM Journal on Numerical Analysis*, 24(5):1152–1170.

Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. (2016). Manifold Gaussian processes for regression. In *Proc. IEEE Intl Joint Conf. on Neural Networks (IJCNN)*.

Calinon, S. (2016). A tutorial on task-parametrized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29.

Carroll, J. D. and Chang, J. J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319.

Chiacchio, P. (1990). Exploiting redundancy in minimum-time path following robot control. In *American Control Conference*, pages 2313–2318.

Chiacchio, P., Bouffard Vercelli, Y., and Pierrot, F. (1997). Force polytope and force ellipsoid for redundant manipulators. *Journal of Robotic Systems*, 14(8):613–620.

Chiacchio, P., Chiaverini, S., Sciavicco, L., and Siciliano, B. (1991a). Global task space manipulability ellipsoids for multiple-arm systems. *IEEE Trans. on Robotics and Automation*, 7(5):678–685.

Chiacchio, P., Chiaverini, S., Sciavicco, L., and Siciliano, B. (1991b). Reformulation of dynamic manipulability ellipsoid for robotic manipulators. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2192–2197.

Chirikjian, G. and Kobilarov, M. (2014). Gaussian approximation of non-linear measurement models on Lie groups. In *IEEE Conference on Decision and Control (CDC)*, pages 6401–6406.

Chiu, S. (1987). Control of redundant manipulators for task compatibility. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1718–1724.

Chiu, S. L. (1988). Task compatibility of manipulator postures. *Intl. Journal of Robotics Research*, 7(5):13–21.

Cohn, D., Ghahramani, Z., and Jordan, M. (1996). Active learning with statistical models. *Artificial Intelligence Research*, 4:129–145.

Cos, I., Bélanger, N., and Cisek, P. (2011). The influence of predicted arm biomechanics on decision making. *Journal of Neurophysiology*, 105:3022–3033.

Cully, A., Clune, J., Tarapore, D., and Mouret, J. B. (2015). Robots that can adapt like animals. *Nature*, 521:503–507.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. (2018). Hyperspherical variational auto-encoders. In *Conference on Uncertainty in Artificial Intelligence*.

Deisenroth, M. P., Neumann, G., and Peters, J. (2013). *A Survey on Policy Search for Robotics*. Foundations and Trends in Robotics.

Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Intl. Conf. on Machine Learning (ICML)*, pages 465–472.

Delhaisse, B., Rozo, L., and Caldwell, D. G. (2019). Pyrobolearn: A Python framework for robot learning practitioners. In *Conference on Robot Learning (CoRL)*.

Ding, X. and Fang, C. (2013). A novel method of motion planning for an anthropomorphic arm based on movement primitives. *IEEE/ASME Transactions on Mechatronics*, 18(2):624–636.

Djolonga, J., Krause, A., and Cevher, V. (2013). High-dimensional Gaussian process bandits. In *Neural Information Processing Systems (NIPS)*.

do Carmo, M. (1992). *Riemannian Geometry*. Birkhäuser Basel.

Doty, K., Schwartz, E., Melchiorri, C., and Bonivento, C. (1995). Robot manipulability. *IEEE Transactions on Robotics and Automation*, 11(3):462–468.

Dounskaia, N., Wang, W., Sainburg, R. L., and Przybyla, A. (2014). Preferred directions of arm movements are independent of visual perception of spatial directions. *Experimental Brain Research*, 232(2):575–586.

Drieß, D., Englert, P., and Toussaint, M. (2017). Constrained Bayesian optimization of combined interaction force/task space controllers for manipulations. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 902–907.

Dryden, I. L. (2005). Statistical analysis on high-dimensional spheres and shape spaces. *Annals of Statistics*, 33(4):1643–1665.

Dubbelman, G. (2011). *Intrinsic Statistical Techniques for Robust Pose Estimation.* PhD thesis, University of Amsterdam, Netherlands.

Edelman, A., Arias, T. A., and Smith, S. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal of Matrix Analysis and Applications*, 20(2):303–351.

Englert, P. and Toussaint, M. (2016). Combined optimization and reinforcement learning for manipulations skills. In *Robotics: Science and Systems (R:SS).*

Eriksson, D., Pearce, M., Gardner, J. R., Turner, R., and Poloczek, M. (2019). Scalable global optimization via local Bayesian optimization. In *Neural Information Processing Systems (NIPS).*

Fang, C. and Ding, X. (2013). A set of basic movement primitives for anthropomorphic arms. In *IEEE Intl. Conf. on Mechatronics and Automation*, pages 639–644.

Feragen, A. and Hauberg, S. (2016). Open problem: Kernel methods on manifolds and metric spaces. what is the probability of a positive definite geodesic exponential kernel? In *29th Annual Conference on Learning Theory*, pages 1647–1650.

Feragen, A., Lauze, F., and Hauberg, S. (2015). Geodesic exponential kernels: When curvature and linearity conflict. In *Conf. on Computer Vision and Pattern Recognition (CVPR).*

Fisher, N. I., Lewis, T., and Embleton, B. J. J. (1987). *Statistical analysis of spherical data.* Cambridge University Press.

Fletcher, P. T. (2013). Geodesic regression and the theory of least squares on Riemannian manifolds. *International Journal of Computer Vision*, 105(2):171–185.

Fletcher, P. T. and Joshi, S. C. (2004). Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors. In *In Proc. of CVAMIA and MMBIA Worshops*, pages 87–98.

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2017). On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. *arXiv preprint 1807.02811.*

Fréchet, M. (1948). Les éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l'Institut Henry Poincaré*, 10(4):215–310.

Freifeld, O., Hauberg, S., and Black, M. J. (2014). Model transport: Towards scalable transfer learning on manifolds. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1378–1385.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. (2018). GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Neural Information Processing Systems (NIPS).*

Gelfand, A. E., Schmidt, A. M., Banerjee, S., and Sirmans, C. F. (2004). Nonstationary multivariate process modeling through spatially varying coregionalization. *TEST*, 13(2):263–312.

Ghadirzadeh, A., Bütepage, J., Maki, A., Kragic, D., and Björkman, M. (2016). A sensorimotor reinforcement learning framework for physical human-robot interaction. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2682–2688.

Ghahramani, Z. and Jordan, M. (1994). Supervised learning from incomplete data via an EM approach. In *Neural Information Processing Systems (NIPS)*, volume 6, pages 120–127.

Goh, A. and Vidal, R. (2008). Clustering and dimensionality reduction on riemannian manifolds. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7.

Goljat, R., Babič, J., Petrič, T., Peternel, L., and Morimoto, J. (2017). Power-augmentation control approach for arm exoskeleton based on human muscular manipulability. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 5929–5934.

Gong, B., Shi, Y., Sha, F., and Grauman, K. (2012). Geodesic flow kernel for unsupervised domain adaptation. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2066–2073.

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation.* Oxford University Press, USA.

GPy (since 2012). GPy: A Gaussian process framework in Python. http://github.com/SheffieldML/GPy.

Gu, Y., Lee, G., and Yao, B. (2015). Feasible center of mass dynamic manipulability of humanoid robots. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 5082–5087.

Guilamo, L., Kuffner, J., Nishiwaki, K., and Kagami, S. (2006). Manipulability optimization for trajectory generation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2017–2022.

Guo, W., Kotsia, I., and Patras, I. (2012). Tensor learning for regression. *IEEE Transactions on Image Processing*, 21(2):816–827.

Habas, C., Berthoz, A., Flash, T., and Bennequin, D. (2020). Does the cerebellum implement or select geometries? A speculative note. *The Cerebellum*, 19(2):336–342.

Hahne, J. M., Bießmann, F., Jiang, N., Rehbaum, H., Farina, D., Meinecke, F. C., Müller, K. R., and Parra, L. C. (2014). Linear and nonlinear regression techniques for simultaneous and proportional myoelectric control. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(2):269–279.

Harandi, M., Salzmann, M., and Hartley, R. (2014). From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In *Proc. European Conf. on Computer Vision (ECCV)*.

Harandi, M., Salzmann, M., and Hartley, R. (2018). Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(1):48–62.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.

Hauberg, S. (2016). Principal curves on Riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1915–1921.

Hauberg, S. r., Freifeld, O., and Black, M. J. (2012). A geometric take on metric learning. In *Neural Information Processing Systems (NIPS)*, volume 25, pages 2024–2032.

Hu, J., Liu, X., Wen, Z., and Yuan, Y. (2019). A brief introduction to manifold optimization. *arXiv preprint 1906.05450*.

Huang, Y., Rozo, L., Silvério, J., and Caldwell, D. G. (2019). Kernelized movement primitives. *Intl. Journal of Robotics Research*, 38(7):833–852.

Hung, H. and Wang, C. (2013). Matrix variate logistic regression model with application to EEG data. *Biostatistics*, 14(1):189–202.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixture of local experts. *Neural Computation*, 3(1):79–87.

Jacquier-Bret, J., Gorce, P., and Rezzoug, N. (2012). The manipulability: A new index for quantifying movement capacities of upper extremity. *Ergonomics*, 55(1):69–77.

Jaquier, N. and Calinon, S. (2017). Gaussian mixture regression on symmetric positive definite matrices manifolds: Application to wrist motion estimation with sEMG. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 59–64.

Jaquier, N., Connan, M., Castellini, C., and Calinon, S. (2017). Combining electromyography and tactile myography to improve hand and wrist activity detection in prostheses. *Technologies*, 5(4).

Jaquier, N., Ginsbourger, D., and Calinon, S. (2019a). Learning from demonstration with model-based Gaussian process. In *Conference on Robot Learning (CoRL)*.

Jaquier, N., Haschke, R., and Calinon, S. (2020a). Tensor-variate mixture of experts for proportional myographic control of a robotic hand. *Submitted to Robotics and Autonomous Systems, Available as arXiv:1902.11104.*

Jaquier, N., Rozo, L., Caldwell, D. G., and Calinon, S. (2018). Geometry-aware tracking of manipulability ellipsoids. In *Robotics: Science and Systems (R:SS).*

Jaquier, N., Rozo, L., Caldwell, D. G., and Calinon, S. (2020b). Geometry-aware manipulability learning, tracking and transfer. *Intl. Journal of Robotics Research.*

Jaquier, N., Rozo, L., and Calinon, S. (2020c). Analysis and transfer of human movement manipulability in industry-like activities. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS).*

Jaquier, N., Rozo, L., Calinon, S., and Bürger, M. (2019b). Bayesian optimization meets Riemannian manifolds in robot learning. In *Conference on Robot Learning (CoRL).*

Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. (2015). Kernel methods on Riemannian manifolds with Gaussian RBF kernels. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 37(12):2464–2477.

Jost, J. (2008). *Riemannian Geometry and Geometric Analysis*, volume 5 of *Universitext*. Springer-Verlag Berlin Heidelberg.

Jung, S., Dryden, I. L., and Marron, J. S. (2012). Analysis of principal nested spheres. *Biometrika*, 99(3):551–568.

Kandasamy, K., Schneider, J., and Poczos, B. (2015). High dimensional Bayesian optimisation and bandits via additive models. In *Intl. Conf. on Machine Learning (ICML).*

Kendall, D. G. (1984). Shape manifolds, procrustean metrics, and complex projective spaces. *Bulletin of the London Mathematical Society*, 16(2):81–121.

Kim, S., Haschke, R., and Ritter, H. (2017). Gaussian mixture model for 3-DoF orientations. *Robotics and Autonomous Systems*, 87:28–37.

Kim, W., Lee, S., Kang, M., Han, J., and Han, C. (2010). Energy-efficient gait pattern generation of the powered robotic exoskeleton using DME. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2475–2480.

Kirschner, J., Mutny, M., Hiller, N., Ischebeck, R., and Krause, A. (2019). Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In *Intl. Conf. on Machine Learning (ICML)*, pages 3429–3438.

Knudde, N., van der Herten, J., Dhaene, T., and Couckuyt, I. (2017). GPflowOpt: A Bayesian optimization library using tensorFlow. *arXiv preprint 1711.03845.*

Kõiva, R., Riedenklau, E., Viegas, C., and Castellini, C. (2015). Shape conformable high spatial resolution tactile bracelet for detecting hand and wrist activity. In *IEEE Intl. Conf. on Rehabilitation Robotics (ICORR)*, pages 157–162.

Kolda, T. and Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Kossaifi, J., Panagakis, Y., Anandkumar, A., and Pantic, M. (2019). Tensorly: Tensor learning in Python. *Journal of Machine Learning Research*, 20(26):1–6.

Kraft, D. (1988). A software package for sequential quadratic programming. Technical report, Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen.

Kupcsik, A., Hsu, D., and Lee, S. (2015). Learning dynamic robot-to-human object handover from human feedback. In *Intl. Symp. on Robotics Research.*

Lee, I. and Oh, J. (2016). Humanoid posture selection for reaching motion and a cooperative balancing controller. *Journal of Intelligent and Robotics Systems*, 8(3-4):301–316.

Lee, J. (1997). A study on the manipulability measures for robot manipulators. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1458–1465.

Lee, J. (2012). *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics.* Springer, New York.

Lee, S. (1989). Dual redundant arm configuration optimization with task-oriented dual arm manipulability. *IEEE Transactions on Robotics and Automation*, 5(1):78–97.

Lee, T. and Park, F. C. (2018). A geometric algorithm for robust multibody inertial parameter identification. *IEEE Robotics and Automation Letters*, 3(3):2455–2462.

Li, C., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S., and Shilton, A. (2017). High dimensional Bayesian optimization using dropout. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2096–2102.

Li, C.-L., Kandasamy, K., Póczos, B., and Schneider, J. (2016). High dimensional Bayesian optimization via restricted projection pursuit models. In *International Conference on Artificial Intelligence and Statistics (AISTATS).*

Liu, C. and Boumal, N. (2019). Simple algorithms for optimization on Riemannian manifolds with constraints. *Applied Mathematics & Optimization*, pages 1–33.

Lu, H., Plataniotis, K. N., and Venetsanopoulos, A. N. (2008). MPCA: Multilinear principal component analysis of tensor objects. *IEEE Transactions on Neural Networks*, 19(1):18–39.

Luh, J. Y. S. and Zheng, Y. F. (1987). Constrained relations between two coordinated industrial robots for motion control. *Intl. Journal of Robotics Research*, 6(3):60–70.

Maeda, G., Ewerton, M., Osa, T., Busch, B., and Peters, J. (2017). Active incremental learning of robot movement primitives. In *Conference on Robot Learning (CoRL)*, pages 37–46.

Mallasto, A. and Feragen, A. (2018). Wrapped Gaussian process regression on Riemannian manifolds. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5580–5588.

Mandel, J. (1965). Generalisation de la théorie de plasticité de W. T. Koiter. *International Journal of Solids and Structures*, 1(3):273–295.

Marco, A., Hennig, P., Bohg, J., Schaal, S., and Trimpe, S. (2016). Automatic LQR tuning based on Gaussian process global optimization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 270–277.

Marco, A., Hennig, P., Schaal, S., and Trimpe, S. (2017). On the design of LQR kernels for efficient controller learning. In *IEEE Conference on Decision and Control (CDC)*, pages 5193–5200.

Martinez-Cantin, R. (2017). Bayesian optimization with adaptive kernels for robot control. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3350–3356.

Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. (2017). GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6.

Maurice, P., Malaisé, A., Amiot, C., Paris, N., Richard, G.-J., Rochel, O., and Ivaldi, S. (2019). Human movement and ergonomics: An industry-oriented dataset for collaborative robotics. *Intl. Journal of Robotics Research*, 38(14):1529–1537.

Miripour Fard, B. (2019). A manipulability analysis of human walking. *J. Biomech.*, 83:157–164.

Mistry, M., Nakanishi, J., Cheng, G., and Schaal, S. (2008). Inverse kinematics with floating base and constraints for full body humanoid robot control. In *IEEE/RAS Intl. Conf. on Humanoid Robots (Humanoids)*, pages 22–27.

Miwakeichi, F., Martínez-Montes, E., Valdés-Sosa, P. A., Nishiyama, N., Mizuhara, H., and Yamaguchi, Y. (2004). Decomposing EEG data into space-time-frequency components using parallel factor analysis. *NeuroImage*, 22(3):1035–1045.

Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404.

Morasso, P. (1981). Spatial control of arm movements. *Experimental Brain Research*, 42:223–227.

Moriconi, R., Sesh Kumar, K. S., and Deisenroth, M. P. (2019). High-dimensional Bayesian optimization with manifold Gaussian processes. *arXiv preprint 1902.10675*.

Munteanu, A., Nayebi, A., and Poloczek, M. (2019). A framework for Bayesian optimization in embedded subspaces. In *Intl. Conf. on Machine Learning (ICML)*, volume 97, pages 4752–4761.

Murray, R. M., Li, Z., and Shankar Sastry, S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc.

Mutný, M. and Krause, A. (2018). Efficient high dimensional Bayesian optimization with additivity and quadrature fourier features. In *Neural Information Processing Systems (NIPS)*.

Nguyen-Tuong, D., Peters, J., and Seeger, M. (2009). Local Gaussian process regression for real time online model learning and control. In *Advances in neural information processing systems 21*, pages 1193–1200.

Nielsen, J. L. G., Holmgaard, S., Jiang, N., Englehart, K. B., Farina, D., and Parker, P. A. (2011). Simultaneous and proportional force estimation for multifunction myoelectric prostheses using mirrored bilateral training. *IEEE Transactions on Biomedical Engineering*, 58(3):681–688.

Nissler, C., Connan, M., Nowak, M., and Castellini, C. (2017). Online tactile myography for simultaneous and proportional hand and wrist myocontrol. In *Proceedings of Myoelectric Control Symposium (MEC)*.

Oh, C., Gavves, E., and Welling, M. (2018). BOCK: Bayesian optimization with cylindrical kernels. In *Intl. Conf. on Machine Learning (ICML)*, pages 3868–3877.

Pait, F. and Colón, D. (2010). Some properties of the Riemannian distance function and the position vector X, with applications to the construction of Lyapunov functions. In *IEEE Conference on Decision and Control (CDC)*, pages 6277–6280.

Paraschos, A., Daniel, C., Peters, J., and Neumann, G. (2013). Probabilistic movement primitives. In *Neural Information Processing Systems (NIPS)*, pages 2616–2624.

Park, F. C. (1995). Optimal robot design and differential geometry. *Journal of Mechanical Design*, 117(B):87–92.

Park, F. C., Kim, B., Jang, C., and Hong, J. (2018). Geometric algorithms for robot dynamics: A tutorial review. *Applied Mechanics Reviews*, 70(1).

Park, F. C. and Kim, J. W. (1998). Manipulability of closed kinematic chains. *ASME Journal of Mechanical Design*, 120(4):542–548.

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 763–768.

Pautrat, R., Chatzilygeroudis, K., and Mouret, J.-B. (2018). Bayesian optimization with automatic prior selection for data-efficient direct policy search. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 7571–7578.

Pellionisz, A. and Llinás, R. (1985). Tensor network theory of the metaorganization of functional geometries in the central nervous system. *Neuroscience*, 16(2):245–273.

Pennec, X. (2018). Barycentric subspace analysis on manifolds. *Annals of Statistics*, 46(6A):2711–2746.

Pennec, X., Fillard, P., and Ayache, N. (2006). A Riemannian framework for tensor computing. *Intl. Journal on Computer Vision*, 66(1):41–66.

Pennec, X., Sommer, S., and Fletcher, T. (2019). *Riemannian Geometric Statistics in Medical Image Analysis*. Elsevier.

Petrič, T., Peternel, L., Morimoto, J., and Babič, J. (2019). Assistive arm-exoskeleton control based on human muscular manipulability. *Frontiers in Neurorobotics*, 13.

Phillips, S. L. and Craelius, W. (2005). Residual kinetic imaging: A versatile interface for prosthetic control. *Robotica*, 23(3):277–282.

Prattichizzo, D., Malvezzi, M., Gabiccini, M., and Bicchi, A. (2012). On the manipulability ellipsoids of underactuated robotic hands with compliance. *Robotics and Autonomous Systems*, 60(3):337–346.

Rai, A., Antonova, R., Song, S., Martin, W., Geyer, H., and Atkeson, C. (2018). Bayesian optimization using domain knowledge on the ATRIAS biped. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1771–1778.

Rana, M. A., Li, A., Ravichandar, H., Mukadam, M., Chernova, S., Fox, D., Boots, B., and Ratliff, N. (2019). Learning reactive motion policies in multiple task spaces from human demonstrations. In *Conference on Robot Learning (CoRL)*.

Rana, M. A., Mukadam, M., Ahmadzadeh, S. R., Chernova, S., and Boots, B. (2017a). Towards robust skill generalization: Unifying learning from demonstration and motion planning. In *Conference on Robot Learning (CoRL)*, pages 109–118.

Rana, S., Li, C., Gupta, S., Nguyen, V., and Venkatesh, S. (2017b). High dimensional Bayesian optimization with elastic Gaussian process. In *Intl. Conf. on Machine Learning (ICML)*, pages 2883–2891.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Ratliff, N. (2013). Learning geometric reductions for planning and control. In *Intl. Conf. on Machine Learning (ICML), Workshop on robot learning*.

Ratliff, N., Toussaint, M., and Schaal, S. (2015). Understanding the geometry of workspace obstacles in motion optimization. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4202–4209.

Ratliff, N. D., Issac, J., and Kappler, D. (2018). Riemannian motion policies. *arXiv preprint 1801.02854*.

Ravichandar, H., Polydoros, A. S., Chernova, S., and Billard, A. (2020). Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1).

Rozo, L. (2019). Interactive trajectory adaptation through force-guided Bayesian optimization. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 7596–7603.

Rozo, L., Calinon, S., Caldwell, D. G., Jiménez, P., and Torras, C. (2016). Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527.

Rozo, L., Jaquier, N., Calinon, S., and Caldwell, D. G. (2017). Learning manipulability ellipsoids for task compatibility in robot manipulation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3183–3189.

Sabes, P. and Jordan, M. (1997). Obstacle avoidance and a perturbation sensitivity model for motor planning. *Journal of Neuroscience*, 17:7119–7128.

Said, S., Bombrun, L., Berthoumieu, Y., and Manton, J. H. (2017). Riemannian Gaussian distributions on the space of symmetric positive definite matrices. *IEEE Trans. on Information Theory*, 63(4):2153–2170.

Schneider, M. and Ertel, W. (2010). Robot learning by demonstration with local Gaussian process regression. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 255–260.

Selig, J. M. (2005). *Geometric Fundamentals of Robotics*. Monographs in Computer Science. Springer.

Shadow robot company (since 1997). Shadow robot company. https://www.shadowrobot.com/.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Sierra González, D. and Castellini, C. (2013). A realistic implementation of ultrasound imaging as a human-machine interface for upper-limb amputees. *Frontiers in Neurorobotics*, 7(17).

Signoretto, M., De Lathauwer, L., and Suykens, J. A. K. (2011). A kernel-based framework to tensorial data analysis. *Neural Networks*, 24(8):861–874.

Silvério, J., Calinon, S., Rozo, L., and Caldwell, D. G. (2019). Learning task priorities from demonstrations. *IEEE Transactions on Robotics*, 35(1):78–94.

Silvério, J., Huang, Y., Rozo, L., Calinon, S., and Caldwell, D. G. (2018). Probabilistic learning of torque controllers from kinematic and force constraints. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 6552–6559.

Simo-Serra, E., Torras, C., and Moreno-Noguer, F. (2017). 3D human pose tracking priors using geodesic mixture models. *Intl. Journal on Computer Vision*, 122(2):388–408.

Smith, S. T. (1994). Optimization techniques on Riemannian manifold. *Fields Institute Communications*, 3:113–136.

Solà, J., Deray, J., and Atchuthan, D. (2018). A micro Lie theory for state estimation in robotics. *arXiv preprint 1812.01537*.

Somani, N., Rickert, M., Gaschler†, A., Cai, C., Perzylo, A., and Knoll, A. (2016). Task level robot programming using prioritized non-linear inequality constraints. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 430–437.

Sommer, S., Lauze, F., Hauberg, S., and Nielsen, M. (2010). Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximations. In *European Conf. On Computer Vision*, pages 43–56.

Sommer, S., Lauze, F., and Nielsen, M. (2014). Optimization over geodesics for exact principal geodesic analysis. *Advances in Computational Mathematics*, 40(2):283–313.

Sra, S. (2012). A new metric on the manifold of kernel matrices with application to matrix geometric means. *Neural Information Processing Systems*, pages 144–152.

Sra, S. and Hosseini, R. (2015). Conic geometric optimization on the manifold of positive definite matrices. *SIAM Journal on Optimization*, 25(1):713–739.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

Tan, X., Zhang, Y., Tang, S., Shao, J., Wu, F., and Zhuang, Y. (2013). Logistic tensor regression for classification. In *Intelligent Science and Intelligent Data Engineering*, pages 573–581.

Tang, Y., Salakhutdinov, R., and Hinton, G. (2013). Tensor analyzers. In *Intl. Conf. on Machine Learning (ICML)*, pages 163–171.

Tanwani, A. K. and Calinon, S. (2019). Small-variance asymptotics for non-parametric online robot learning. *The International Journal of Robotics Research*, 38(1):3–22.

Tao, D., Li, X., Wu, X., and Maybank, S. J. (2007). General tensor discriminant analysis and Gabor features for gait recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(10):1700–1715.

Torres Alberto, N., Mistry, M., and Stulp, F. (2014). Computed torque control with variable gains through Gaussian process regression. In *IEEE/RAS Intl. Conf. on Humanoid Robots (Humanoids)*, pages 212–217.

Townsend, J., Koep, N., and Weichwald, S. (2016). Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5.

Tuzel, O., Porikli, F., and Meer, P. (2006). Region covariance: A fast descriptor for detection and classification. In *European Conference on Computer Vision (ECCV)*, pages 589–600.

Tyagi, A. and Davis, J. W. (2008). A recursive filter for linear systems on Riemannian manifolds. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Umlauft, J., Fanger, Y., and Hirche, S. (2017). Bayesian uncertainty modeling for programming by demonstration. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 6428–6434.

Vahrenkamp, N., Asfour, T., Metta, G., Sandini, G., and Dillmann, R. (2012). Manipulability analysis. In *IEEE/RAS Intl. Conf. on Humanoid Robots (Humanoids)*, pages 568–573.

Vidovic, M., Hwang, H., Amsüss, S., Hahne, J. M., Farina, D., and Müller, K.-R. (2016). Improving the robustness of myoelectric pattern recognition for upper limb prostheses by covariate shift adaptation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(9).

Vollebregt, E. A. (2014). The bound-constrained conjugate gradient method for non-negative matrices. *Journal of Optimization Theory and Applications*, 162(3):931–953.

Walker, I. D. (1994). Impact configurations and measures for kinematically redundant and multiple armed robot systems. *IEEE Transactions on Robotics and Automation*, 10(5):670–683.

Wang, Z. and Vemuri, B. C. (2004). An affine invariant tensor dissimilarity measure and its applications to tensor-valued image segmentation. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Z., Zoghiy, M., Hutterz, F., Matheson, D., and De Freitas, N. (2013). Bayesian optimization in high dimensions via random embeddings. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1778–1784.

Washizawa, Y., Higashi, H., Rutkowski, T., Tanaka, T., and Cichocki, A. (2010). Tensor based simultaneous feature extraction and sample weighting for EEG classification. In *Intl. Conf. on Neural Information Processing (ICONIP)*, pages 26–33.

Wilson, A., Fern, A., and Tadepalli, P. (2014). Using trajectory data to improve Bayesian optimization for reinforcement learning. *Journal of Machine Learning Research*, 15:253–282.

Wu, D. (2020). Intrinsic construction of Lyapunov functions on Riemannian manifold. *arXiv preprint 2002.11384*.

Xu, J. and Durrett, G. (2018). Spherical latent spaces for stable variational autoencoders. In *In Proc. of Conf. on Empirical Methods in Natural Language Processing (EMNLP)*.

Yoshikawa, T. (1985a). Dynamic manipulability of robotic manipulators. *Journal of Robotic Systems*, 2:113–124.

Yoshikawa, T. (1985b). Manipulability of robotic mechanisms. *Intl. Journal of Robotics Research*, 4(2):3–9.

Yuan, K., Chatzinikolaidis, I., and Li, Z. (2019). Bayesian optimization for whole-body control of high degrees of freedom robots through reduction of dimensionality. *IEEE Robotics and Automation Letters*, 4(3):2268–2275.

Yuan, Y. (1999). A review of trust region algorithms for optimization. In *In Proc. of the Intl Congress on Industrial & Applied Mathematics (ICIAM)*, pages 271–282.

Yuksel, S. E., Wilson, J. N., and Gader, P. D. (2012). Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193.

Yun, A. (2020). *Safe Reconfigurable Robot Systems: Design, Programming, and Reactive Motion Planning.* PhD thesis, Seoul National University.

Zare, A., Ozdemir, A., Iwen, M. A., and Aviyente, S. (2018). Extension of PCA to higher order data structures: An introduction to tensors, tensor decompositions, and tensor PCA. *Proceedings of the IEEE*, 106(8):1341–1358.

Zeestraten, M. J. A. (2018). *Programming by demonstration on Riemannian manifolds.* PhD thesis, Universitá degli Studi di Genova.

Zhan, X. and Ma, B. (2012). Gaussian mixture model on tensor field for visual tracking. *IEEE Signal Processing Letters*, 19(11):733–736.

Zhang, M., Li, H., and Su, S. (2019). High dimensional Bayesian optimization via supervised dimension reduction. In *Proc. of Intl Joint Conf. on Artificial Intelligence (IJCAI)*.

Zhao, Q., Zhou, G., Zhang, L., and Cichocki, A. (2014). Tensor-variate Gaussian processes regression and its application to video surveillance. In *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1265–1269.

Zhou, H., Li, L., and Zhu, H. (2013). Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). CHOMP: Covariant hamiltonian optimization for motion planning. *Intl. Journal of Robotics Research*, 32(9-10):1164–1193.

Noémie Jaquier
noemie.jaquier@gmail.com Swiss
http://www.njaquier.ch Born on 27.10.1992

In order to integrate safely in our society, I believe that robots need to be provided with efficient and reliable learning capabilities. In this context, I aim at enhancing robot abilities by developing models that efficiently integrate the geometric information encapsulated in the data. My research combines robot learning and control with differential geometry.

Research Experience

Doctoral Research Assistant — Aug. 2016 – July 2020

Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, and **Idiap Research Institute**, Martigny, Switzerland

My thesis aims at enhancing robot learning and control capabilities by introducing domain knowledge into the corresponding models. The developed approaches are built on Riemannian manifolds to represent rigid-body orientations, inertia matrices, manipulability ellipsoids, or controller gain matrices. I worked on introducing geometry-awareness into a probabilistic learning-from-demonstrations technique, into control methods and into a Bayesian optimization framework to transfer and refine robot skills.
Until April 2019, I was part of the collaborative project TACT-HAND (SNSF/DFG) collaborative project.

PhD Sabbatical — April 2019 – Sept. 2019

Bosch Center for Artificial Intelligence (BCAI), Renningen, Germany

During my PhD Sabbatical, I focused on refining the skills learned by a robot and adapting them to new situations. I introduced a novel geometry-aware Bayesian optimization (GaBO) framework that exploits the geometry of the search space to properly seek optimal parameters that lie on Riemannian manifolds.

Student Research Assistant — Feb. 2016 – July 2016

Idiap Research Institute, Martigny, Switzerland

I developed a framework to improve the drawing skills of a humanoid robot with visual feedback. Cameras and proximity sensors were exploited to adjust the robot’s arm movements and compensate for the imprecision due to the intrinsic compliance of the arms.

R&D Intern — July 2015 – Sept. 2015

SenseFly, Cheseaux-sur-Lausanne, Switzerland

I was part of the research and development team for aerial imaging drones for professional applications. I developed flight control algorithms for an intelligent mapping and inspection quadrotor.

Education

PhD Electrical Engineering — 2016 – 2020

Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
and **Idiap Research Institute**, Martigny, Switzerland
Thesis title: **“Robot Skills Learning with Riemannian Manifolds: Leveraging Geometry-awareness in Robot Learning, Optimization and Control”**.
Advisors: Prof. Hervé Bourlard and Dr. Sylvain Calinon
Thesis committee: Prof. Auke Ijspeert, Prof. Nicolas Flammarion, Prof. Frank Park, Dr. Jean-Baptiste Mouret

MSc Robotics and Autonomous Systems with a Minor in Computational Neurosciences — 2014 – 2016

Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Thesis title: “**Improving the Drawing Skills of a Humanoid Robot with Visual Feedback”**, at Idiap Research Institute, Martigny, Switzerland
Thesis advisor: Dr. Sylvain Calinon
Thesis committee: Prof. Auke Ijspeert, Dr. Freek Stulp

BSc Microengineering — 2011 – 2014

Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Honors and Awards

Idiap PhD Student Award — Dec. 2019

At the end of each year, the Idiap Research is awarding a research award to a PhD student of the institute.

Best Presentation Award of the Conference on Robot Learning (CoRL) 2019 — Nov. 2019

For the presentation of the paper “**Bayesian Optimization Meets Riemannian Manifolds in Robot Learning**”.
My presentation can be watched at https://www.youtube.com/watch?v=b7StSnt85S4&feature=youtu.be&t=7763.

Robotics: Science and Systems (R:SS) Pioneer — June 2018

RSS Pioneers is a day-long invitation-only workshop for senior graduate students and postdocs, that seeks to bring together a cohort of the world’s top early career researchers in all areas of robotics.

Publications

Under-review Manuscripts

N. Jaquier, R. Haschke, and S. Calinon, **“Tensor-variate Mixture of Experts for Proportional Myographic Control of a Robotic Hand”**, submitted to *Robotics and Autonomous Systems*. (Preprint as arXiv:1902.11104.)

Journal Articles

N. Jaquier, L. Rozo, D.G. Caldwell, and S. Calinon, **“Geometry-aware Manipulability Learning, Tracking and Transfer”**, *International Journal of Robotics Research (IJRR)*, 2020. (Preprint as arXiv:1811.11050.)

N. Jaquier, M. Connan, C. Castellini, and S. Calinon, **“Combining Electromyography and Tactile Myography to Improve Hand and Wrist Activity Detection in Prostheses”**, *MDPI Technologies*, 5:4, Special issue on assistive robotics, 2017.

Conference Proceedings

N. Jaquier, L. Rozo, and S. Calinon, **“Analysis and Transfer of Human Movement Manipulability in Industry-like Activities”**, in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

H. Girgin, E. Pignat, **N. Jaquier**, and S. Calinon, **“Active Improvement of Control Policies with Bayesian Gaussian Mixture Model”**, in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020.

N. Jaquier, L. Rozo, S. Calinon, and M. Bürger, **“Bayesian Optimization Meets Riemannian Manifolds in Robot Learning”**, in *Conference on Robot Learning (CoRL)*, 2019. (Oral presentation; Best presentation award.)

N. Jaquier, D. Ginsbourger, and S. Calinon, **“Learning from Demonstration with Model-based Gaussian Process”**, in *Conference on Robot Learning (CoRL)*, 2019.

N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon, **“Geometry-aware Tracking of Manipulability Ellipsoids”**, in *Robotics : Science and Systems (R:SS)*, 2018.

N. Jaquier and S. Calinon, **“Gaussian Mixture Regression on Symmetric Positive Definite Matrices Manifolds: Application to Wrist Motion Estimation with sEMG”**, in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 59-64.

L. Rozo, **N. Jaquier**, S. Calinon, and D. G. Caldwell,. **“Learning Manipulability Ellipsoids for Task Compatibility in Robot Manipulation”**, in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017, pp. 3183-3189.

N. Jaquier, C. Castellini, and S. Calinon, **“Improving Hand and Wrist Activity Detection using Tactile Sensors and Tensor Regression Methods on Riemannian Manifolds”**, *Myoelectric control Symposium (MEC)*, 2017.

Workshop Papers

N. Jaquier, and S. Calinon, **“Geometry-aware Control and Learning in Robotics”**, in *R:SS Pioneers Workshop*, 2018.

N. Jaquier, L. Rozo, and S. Calinon, **“Geometry-aware Robot Manipulability Transfer”**, in *Learning and Inference in Robotics: Integrating Structure, Priors and Models (LAIR) R:SS Workshop*, 2018.

Outreach activities

N. Jaquier, and S. Calinon, **“Improving the Control of Prosthetic Hands with Tactile Sensing”**, *Micro & Nano Magazine, Micronarc*, 2018, pp. 42-43.

Open-source Software Development

GaBOtorch (https://github.com/NoemieJaquier/GaBOtorch)

This repository contains the source code to perform Geometry-aware Bayesian Optimization (GaBO) on Riemannian manifolds. This Python code is based on the PyTorch, GPyTorch, BoTorch and Pymanopt libraries.

GaBOflow (https://github.com/NoemieJaquier/GaBOflow)

This repository contains the source code to perform Geometry-aware Bayesian Optimization (GaBO) on Riemannian manifolds. This Python code is based on the Tensorflow, GPflow and GPflowOpt and Pymanopt libraries.

Manipulability (https://github.com/NoemieJaquier/Manipulability)

This MATLAB codes show simple examples for manipulability learning, tracking and transfer tasks. These approaches offer the possibility of transferring posture-dependent task requirements between agents of different embodiments.

GMR-based GP (https://github.com/NoemieJaquier/GMRbasedGP)

This repository shows examples of application of Gaussian mixture regression (GMR), Gaussian process (GP) and GMR-based Gaussian process on 2D trajectories. This Python code is based on the GPy library.

Invited talks

13.07.2019	Festival Scientastic, EPFL campus Valais-Wallis, Switzerland.
19.05.2019	Minisymposium on “Algebraic Geometry for Kinematics and Dynamics in Robotics”, SIAM conference on Applied Algebraic Geometry, Bern, Switzerland.
22.11.2018	Valais/Wallis AI workshop “AI for rehabilitation and Prosthetics”, HES-SO Valais, Switzerland.
28.11.2017	Swiss Machine Learning Day, EPFL, Switzerland.
14.11.2017	Applied Mathematics Seminar, Applied Mathematic Dept., UCL, Belgium.
24.09.2017	Workshop on “Micro-data: the next frontier in robot learning ?”, IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) , Vancouver, Canada.
23.05.2017	Numerical Analysis Seminar. Mathematic Dept., University of Geneva, Switzerland.
31.01.2017	Applied Machine Learning Days, EPFL, Switzerland.
20.01.2017	Class for operating room technicians, Upper School of the Health (ES Santé), Lausanne, Switzerland.

Workshop organisation

Workshop on Bringing Geometric Methods to Robot Learning, Optimization and Control

This workshop will be offered at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.
Main organizer. Workshop website: https://sites.google.com/view/iros2020-geometric-methods/home

Minisymposium on Algebraic Geometry for Kinematics and Dynamics in Robotics

Offered at the SIAM Conference on Applied Algebraic Geometry 2019.
Main organizer and presenter. Workshop website: https://www.idiap.ch/workshop/magkdr/.

Teaching Experience

Students co-supervisions

2019 – 2020	Clément Vaes, MSc Student at Université Catholique de Louvain. This master thesis was co-supervised by Dr. Estelle Massart, Prof. Pierre-Antoine Absil and Dr. Sylvain Calinon. **Thesis title:** **“Classification of Biologic Signals on the Symmetric Positive Definite Manifold”**.
2019	Gauthier Müller, MSc Student at Ecole Polytechnique Fédérale de Lausanne (EPFL). This semester project was co-supervised by Dr. Sylvain Calinon. **Project title:** **“TME vs CNN: a smaller comparison”**.

Teaching Assistance

2013	Lecturer’s Assistant for Analysis class (first year BSc students), Ecole Polytechnique Fédérale de Lausanne (EPFL).

Reviewing service

Robotics and Autonomous Systems (RAS)
IEEE Robotics and Automation Letters (RA-L)
IEEE Transactions on Industrial Informatics
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
IEEE International Conference on Robotics and Automation (ICRA)

Software Skills

Programming languages

- Matlab
- Python
- C++

Robots

- Panda (Franka Emika)
- Baxter (Rethink Robotics)
- WAM (Barrett Technology Inc.)

Languages

English (fluent)
French (native)
German (basic knowledge)
Spanish (basic knowledge)