Robotics and Autonomous Systems 206 (2026) 105717

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Fast GPU evaluation of differentiable Signed Distance Fields for robotics via tensor decompositions

Andrea Govoni [a,*], Sylvain Calinon [b], Gianluca Palli [a]

[a] Unibo, Via Risorgimento 2, Bologna, 40136, Italy
[b] Idiap, Research Institute, Rue Marconi 19, Lausanne, 1920, Switzerland

ARTICLE INFO

Dataset link: https://github.com/GovoUnibo/CP-TT-RDF, https://zenodo.org/records/20041439

Keywords:
Signed distance fields
Tensor decomposition
Robotic implicit surfaces
Roofline analysis
GPU-accelerated robotics

ABSTRACT

Signed Distance Fields (SDF) are widely used in robotics for collision checking, reactive control and trajectory optimization, but high-fidelity batched evaluation on GPUs remains challenging. This work proposes a low-rank formulation based on tensor decompositions of Bernstein polynomial SDF models. In particular, link-wise coefficient tensors are represented in low-rank tensor formats, namely canonical polyadic (CP) and tensor train (TT), yielding compact and differentiable implicit models with analytical evaluation of both the distance field and its gradient. This is especially relevant for robotics applications in which collision handling often requires not only distance queries but also gradient information to generate repulsive actions, deformation directions and optimization updates. A roofline-inspired analysis is introduced to interpret the runtime behavior of dense and compressed models across polynomial orders and batch sizes. Experiments on SDFs representing the different links of a robot manipulator as implicit shapes show that low-rank significantly reduces runtime and memory with respect to the dense model, while preserving sufficient geometric accuracy for collision-aware applications. In particular, CP provides the best nominal runtime in most tested settings, whereas TT offers a more favorable memory scalability trade-off. Reactive collision-avoidance examples further confirm that the proposed models provide gradients accurate enough for real-time motion generation.

1. Introduction

Safe real-time motion in cluttered and dynamic environments requires fast evaluation of obstacle and self-collision distances. Signed Distance Functions/Fields (SDFs) provide a compact implicit representation that supports efficient field and gradient queries for collision checking, trajectory optimization, reactive control and mapping [1–6].

Recent robotics research has increasingly adopted SDFs as a unifying primitive across three intertwined directions. First, SDFs are increasingly embedded directly into the control loop (e.g., QP-IK [7–9] and MPC [10]), where stable gradients translate into repulsive actions [4] and smooth avoidance behaviors [3]. Second, there is a shift from discrete grid-based distance representations toward continuous SDF models, such as neural implicit representations, Gaussian processes and polynomial bases, to enable resolution-independent queries and smoother gradients for optimization [11–14]. Third, the notion of distance is being extended beyond workspace geometry toward configuration-dependent representations, yielding joint-space distance queries and derivatives that more directly match the decision variables of manipulators [9].

Despite these advancements, a key bottleneck remains in robotics applications that require large batches of distance and gradient queries under strict latency constraints. This is particularly relevant in GPU-based motion generation, reactive control and large-scale collision-aware optimization, where the practical value of an SDF model depends not only on geometric fidelity, but also on how its evaluation cost scales with batch size, approximation order and hardware characteristics.

To address this issue, we build on the link-wise Robot Distance Field (RDF) framework and on the dense Bernstein-polynomial SDF formulation introduced in [4,7]. These prior works already provide a detailed reconstruction of how the differentiable link-wise distance field is built [7] and exploit SDF gradients for collision-aware motion generation [4].

The contribution of this paper instead lies in introducing a low-rank Bernstein RDF formulation in which the learned coefficient tensor associated with each robot link is represented directly in canonical polyadic (CP), also known as CANDECOMP/PARAFAC, or Tensor-Train (TT) form. The proposed formulation combines the structure of Bernstein RDFs with low-rank tensor parameterizations. It preserves the

* Corresponding author.
E-mail address: andrea.govoni11@unibo.it (A. Govoni).

https://doi.org/10.1016/j.robot.2026.105717
Received 6 May 2026; Received in revised form 24 July 2026; Accepted 22 August 2026
Available online 12 September 2026
0921-8890/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

analytical differentiability and link-wise structure of the original Bernstein RDF while reducing model storage and in favorable GPU regimes, lowering the computational cost of large batched evaluations.

The technical development of this work proceeds as follows. First, in Section 3, we revisit the dense Bernstein RDF formulation and express it in tensor form, making explicit the role of the link-wise coefficient tensor in distance and spatial-gradient evaluation. Starting from this reference formulation, we show how the coefficient tensor can be represented in CP or TT form and derive the corresponding factor- and core-level contractions for batched distance and gradient queries without reconstructing the dense tensor at evaluation time. While analytical SDF gradients are inherited from the original Bernstein formulation, their direct evaluation from CP factors or TT cores is specific to the proposed low-rank model.

We then consider two complementary strategies for identifying the low-rank parameters. As explained in Section 4, an existing dense RDF can be compressed post hoc by decomposing in the weight tensor using classical TD method [15], or the CP factors and TT cores can be optimized directly from sampled SDF values in field space, avoiding the construction of the dense model. Both procedures instantiate the same low-rank Bernstein RDF model class, although they solve different approximation problems.

In Section 5 we analyze when the proposed representations can be used, while Section 6 compares the construction requirements, approximation accuracy, memory footprint, and evaluation cost of the dense, CP, and TT formulations. Analytical complexity expressions and a roofline-inspired analysis identify their practical trade-offs and break-even regimes. The experiments in Section 7 experiments, their accuracy–memory–runtime behavior and their suitability for gradient-based collision avoidance. Section 8 concludes the paper and Appendix 9 provides the algebraic derivations omitted from the main text.

2. Related work

Distance fields have long been used as implicit geometric representations providing distances, inside/outside queries and gradient directions useful for collision reasoning [1]. For online scene updates, piecewise-polynomial SDF models enable incremental learning of continuous distance fields with analytical gradients [11]. Hierarchical SDF schemes instead decompose the representation across multiple levels, supporting scalable mapping in large and unknown environments [16].

In robotics, distance fields, especially grid-based fields and Euclidean Signed Distance Fields (ESDFs), are widely used as implicit representations for collision reasoning [2,14]. However, their memory footprint and discretization artifacts motivate continuous alternatives, while GPU acceleration and sparse volumetric data structures have improved the real-time performance of grid-based approaches [12,17,18]. Neural SDFs are attractive for their compactness and differentiability and have been used to learn joint-space distance functions that enable efficient batched GPU queries, but may not explicitly encode the robot's kinematic structure [3,9,19,20]. In contrast, recent research has shifted toward representing robot geometry by modeling each link separately and composing the full robot geometry through SE(3), which enables gradients for optimization and control [4,7,8]. Additionally, point-based methods like RMMI [21] and GPU-accelerated SDF computation [22] offer scalability but lack of differentiable structure.

In trajectory optimization and motion generation, SDFs are widely used to define smooth collision costs, while recent works have shown how continuous SDF models and GPU-parallel pipelines enable fast collision checking, reactive control and low-latency large-scale optimization [22–24].

The computational advantages of SDF-based representations are evident from their successful integration into real-time control frameworks, especially those based on quadratic programming [3,8,9,21,25]. By contrast, their use in task-priority formulations remains comparatively less explored [4,26,27].

Low-rank tensor decomposition (TD) such as CP and TT provide a principled way to compress high-dimensional field representations while retaining efficient evaluation [14,15,28,29]. Building on these ideas, we develop a low-rank tensor formulation of polynomial SDF models with analytical gradients, designed for GPU execution and large-scale batched queries.

3. Signed distance functions with Bernstein bases

On compact domains, multivariate polynomials provide a standard approximation tool for continuous scalar fields. In the proposed framework, the polynomial order controls the expressiveness of the approximation, whereas the tensor rank controls how compactly the coefficient tensor is represented within the chosen basis. By the Stone–Weierstrass theorem, polynomial functions are dense in $C(\Omega)$ for compact sets $\Omega \subset \mathbb{R}^3$. Hence, on any bounded region of interest, sufficiently rich polynomial bases can approximate continuous target fields arbitrarily well [30,31]. By contrast, TD do not modify the approximation space induced by the selected polynomial basis. Rather, they represent the corresponding coefficient tensor in a structured low-rank format, which may be exact or approximate depending on the chosen rank.

Fig. 1 provides an illustrative comparison among the original mesh (Fig. 1(a)), the corresponding SDF given by the Bernstein approximation (Figs. 1(b) and 1(c)) and its level sets is provided (Fig. 1(d)).

Throughout this paper, $q \in \mathbb{R}^{n_q}$ denotes the robot configuration vector, where n_q is the number of robot degrees of freedom (DOF).

3.1. Dense representation of SDF

Given a closed surface $\mathcal{S}$, the SDF assigns to any point $p \in \mathbb{R}^3$ the minimum Euclidean distance to $\mathcal{S}$ with a sign indicating whether p is outside or inside:

$$f(p) = \begin{cases} d_{\min}(p, \mathcal{S}) & p \text{ outside } \mathcal{S}, \\ 0 & p \in \mathcal{S}, \\ -d_{\min}(p, \mathcal{S}) & p \text{ inside } \mathcal{S}. \end{cases} \tag{1}$$

In robotics each link SDF is conveniently represented in its own local frame. Let $T_l^w(q) \in SE(3)$ denote the homogeneous transform of link frame l in the world frame. Then, distance query point p is obtained by transforming the query point from world frame to the link frame:

$$f_l(p, q) = f_l\left((T_l^w(q))^{-1}\, p\right), \tag{2}$$

where $f_l(\cdot)$ denotes the SDF of link l expressed in the link local frame. Henceforth, we omit the explicit dependence on q and l when clear from the context.

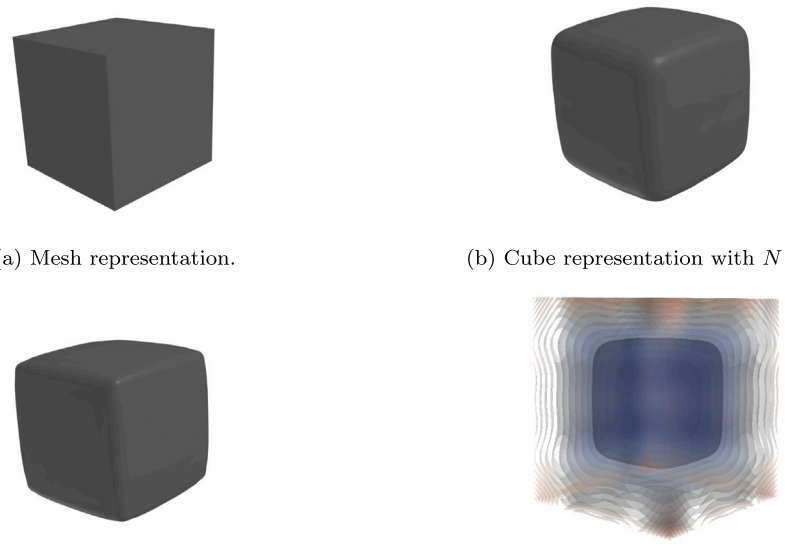
Although exact SDFs are not differentiable everywhere, for example at points with multiple closest-point projections, typically associated with the medial axis (see Sec. II.C in [1]), we seek smooth approximations better suited to optimization problems in robotics. In this respect, tensor-product Bernstein bases provide a constructive and uniformly convergent approximation on bounded domains, while also yielding derivative convergence on compact sets for sufficiently smooth functions [32]. Previous work [4,7] has shown that Bernstein polynomials are effective for approximating SDFs in robotic applications, as summarized next. Each coordinate x, y, z is mapped affinely to $t_x, t_y, t_z \in [0, 1]$ and we define the Bernstein basis of order N :

$$\boldsymbol{\Phi}(t) = \left[\phi_0(t), \ldots, \phi_{N-1}(t)\right], \tag{3}$$

$$\phi_i(t) = \binom{N-1}{i} t^i (1-t)^{N-1-i}. \tag{4}$$

Defining the tensor product basis,

$$\boldsymbol{\Psi}(p) = \boldsymbol{\Phi}(t_x) \otimes \boldsymbol{\Phi}(t_y) \otimes \boldsymbol{\Phi}(t_z), \tag{5}$$



(a) Mesh representation.

(b) Cube representation with $N = 32$.

(c) SDF representation with $N = 128$.

(d) Surface level curves of the cube.

Fig. 1. Different cube representations, including the original mesh, the implicit Bernstein model, the SDF and its level sets. Higher polynomial orders N provide a more accurate approximation of sharp edges while preserving differentiability.

the Bernstein approximation can be written as

$$f_w(p) = \Psi(p)^T w \tag{6}$$

where $w \in \mathbb{R}^{N^3}$ is the vectorized form of coefficients. The resulting approximation of f is smooth and explicitly differentiable; as a result, the gradient ∇f is available everywhere. Differentiating (6) gives

$$\nabla_t f_w(p) = \nabla\Psi(p)w, \tag{7}$$

where the partial derivatives are obtained by differentiating the tensor-product basis (5):

$$\nabla_{t_x}\Psi = \Phi'(t_x) \otimes \Phi(t_y) \otimes \Phi(t_z), \tag{8}$$

$$\nabla_{t_y}\Psi = \Phi(t_x) \otimes \Phi'(t_y) \otimes \Phi(t_z), \tag{9}$$

$$\nabla_{t_z}\Psi = \Phi(t_x) \otimes \Phi(t_y) \otimes \Phi'(t_z). \tag{10}$$

where $\Phi'_x(t_x)$, $\Phi'_y(t_y)$ and $\Phi'_z(t_z)$ denote the element-wise derivatives of the corresponding Bernstein bases in Eq. (3). The resulting approximation is smooth and explicitly differentiable; therefore, the gradient of the approximated field is available everywhere in the domain.

While this representation is functional and dense, it is not the only possible choice; in the remainder of this paper, starting from this formulation, we develop two alternative representations that are computationally more efficient and amenable to GPU implementation.

Remark 1 (*Normalization and Chain Rule*)**.** Bernstein bases are evaluated in normalized coordinates $t = (t_x, t_y, t_z) \in [0,1]^3$ with $t_i = (p_i - p_{\min,i})/(p_{\max,i} - p_{\min,i})$, $i \in \{x, y, z\}$. Throughout the paper, t denotes Bernstein coordinates and p world coordinates. By the chain rule it holds

$$\nabla_p f_w(p) = \operatorname{diag}\left(\frac{1}{p_{\max} - p_{\min}}\right) \nabla_t f(p). \tag{11}$$

3.2. Separability and lower rank approximation

Evaluating the dense Bernstein model in Eq. (6) requires $\mathcal{O}(N^3)$ operations per query. To reduce both storage and evaluation cost, we can estimate the coefficient tensor w with a low-rank tensor decomposition (TD).

Let f_w denote the sampled SDF values generated by the dense model and let f_{TD} denote a generic SDF induced by a low-rank approximation, where w_{TD} is the coefficient vector reconstructed from the TD. Then,

$$\|f_w - f_{\mathrm{TD}}\|_2 = \|\Psi(w - w_{\mathrm{TD}})\|_2 \le \|\Psi\|_2 \, \|w - w_{\mathrm{TD}}\|_2. \tag{12}$$

Therefore, the error induced on the sampled field values is controlled by the coefficient reconstruction error, up to the operator norm of the basis-evaluation map. In addition, higher ranks generally provide a more expressive low-rank parametrization of the coefficient tensor, which can improve reconstruction accuracy and, consequently, reduce the approximation error on the sampled field values.

Although this does not directly guarantee preservation of geometric quantities such as the exact zero-level set, it provides a useful algebraic justification for seeking a low-rank representation w_{TD} of the original coefficient tensor w such that $\|w - w_{\mathrm{TD}}\|_2$ is as small as possible.

As shown in Fig. 2, we consider two standard low-rank parametrizations of w: CP, which represents w as a sum of rank-one terms and TT, which represents it as a chain of low-rank cores. The preference between CP and TT is established based on the observed error–cost trade-off, rather than on an a priori geometric complexity measure.

This distinction is important because a more compact TT parametrization does not automatically imply a faster evaluation than CP. When a target geometry can be approximated with a moderate CP rank, the CP representation is typically preferable from a computational point of view because of its simple separable structure and linear dependence on the CP rank. Conversely, TT becomes attractive mainly for geometries whose accurate CP approximation would require a substantially larger rank, while comparable accuracy can still be obtained with relatively small TT ranks. In such cases, TT can reduce storage and peak memory, although the corresponding runtime improvement may remain limited by implementation-dependent overheads.

3.3. Canonical Polyadic (CP) approximation

Motivated by the bound in Eq. (12), we seek a low-rank representation that keeps the reconstruction error small while improving the computational efficiency of field evaluation. When the coefficient tensor can be well approximated with a sufficiently low CP rank, the CP parametrization provides a compact and efficient representation of the

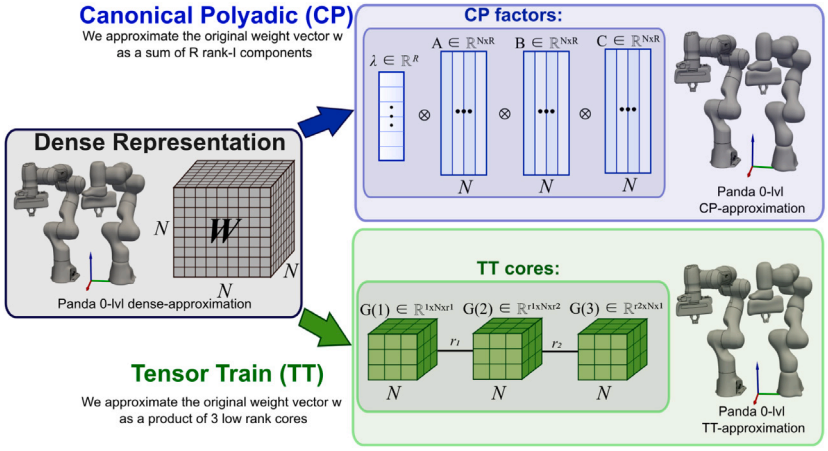


Fig. 2. Canonical polyadic and tensor train decompositions of the Bernstein SDF weight tensor. All three representations yield nearly identical zero-level surfaces.

field, reducing the per-query evaluation cost to $\mathcal{O}(RN)$. Accordingly, w is approximated by a CP model with R components:

$$w_{\mathrm{CP}} = \sum_{r=1}^{R} \lambda_r \left(a_r \otimes_{\mathrm{out}} b_r \otimes_{\mathrm{out}} c_r\right), \tag{13}$$

where $a_r, b_r, c_r \in \mathbb{R}^N$ and $\lambda_r \in \mathbb{R}$ for $r = 1, \ldots, R$, and $\otimes_{\mathrm{out}}$ denotes the outer product.

The resulting scalar field for a single point is

$$f_{\mathrm{CP}}(p) = \sum_{r=1}^{R} \lambda_r \left(\Phi(t_x)^T a_r\right)\left(\Phi(t_y)^T b_r\right)\left(\Phi(t_z)^T c_r\right). \tag{14}$$

The interested reader can find the proof in Appendix 9. Let

$$A = \begin{bmatrix} a_1 & \cdots & a_R \end{bmatrix}, \quad B = \begin{bmatrix} b_1 & \cdots & b_R \end{bmatrix}, \quad C = \begin{bmatrix} c_1 & \cdots & c_R \end{bmatrix},$$

collect the rank-one factors, so that $A, B, C \in \mathbb{R}^{N \times R}$ have columns a_r, b_r, c_r and let $\lambda \in \mathbb{R}^R$. Then, (14) can be written in compact form as

$$f_w(p) \approx f_{\mathrm{CP}}(p) = \left(\Phi(t_x)^\top A \circ \Phi(t_y)^\top B \circ \Phi(t_z)^\top C\right) \lambda, \tag{15}$$

where $\circ$ denotes the element-wise (Hadamard) product.

Differentiating with respect to the normalized coordinates yields

$$\nabla_t f_{\mathrm{CP}}(p) = \begin{bmatrix} \left(\Phi'(t_x)^\top A \circ \Phi(t_y)^\top B \circ \Phi(t_z)^\top C\right) \lambda \\ \left(\Phi(t_x)^\top A \circ \Phi'(t_y)^\top B \circ \Phi(t_z)^\top C\right) \lambda \\ \left(\Phi(t_x)^\top A \circ \Phi(t_y)^\top B \circ \Phi'(t_z)^\top C\right) \lambda \end{bmatrix}, \tag{16}$$

where $\Phi'_x(\cdot)$, $\Phi'_y(\cdot)$ and $\Phi'_z(\cdot)$ denote the element-wise derivatives of the corresponding Bernstein bases. The gradient in world coordinates follows from (11).

3.4. Tensor Train (TT) approximation

As an alternative low-rank parametrization, we also consider the TT format, which provides a more compact and compressed representation of the coefficient tensor while retaining sufficient approximation accuracy. In particular, the TT form reduces the per-query evaluation cost to $\mathcal{O}\big(N(r_1 + r_1 r_2 + r_2)\big)$. Accordingly, w is approximated by a TT model with ranks (r_1, r_2):

$$w_{\mathrm{TT}(i_1,i_2,i_3)} = \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} G^{(1)}_{1,i_1,\alpha_1} G^{(2)}_{\alpha_1,i_2,\alpha_2} G^{(3)}_{\alpha_2,i_3,1}, \tag{17}$$

where $G^{(1)} \in \mathbb{R}^{1 \times N \times r_1}$, $G^{(2)} \in \mathbb{R}^{r_1 \times N \times r_2}$, $G^{(3)} \in \mathbb{R}^{r_2 \times N \times 1}$ are TT cores and $i_{1,2,3}$ are the indices of the Bernstein functions along the x, y and z axes..

The associated SDF is

$$f_{\mathrm{TT}}(p) = \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \underbrace{\left(\sum_{i_1=0}^{N-1} G^{(1)}_{1,i_1,\alpha_1} \phi_{i_1}(t_x)\right)}_{a_x^{(\alpha_1)}(t_x)} \times \underbrace{\left(\sum_{i_2=0}^{N-1} G^{(2)}_{\alpha_1,i_2,\alpha_2} \phi_{i_2}(t_y)\right)}_{B_y^{(\alpha_1,\alpha_2)}(t_y)} \underbrace{\left(\sum_{i_3=0}^{N-1} G^{(3)}_{\alpha_2,i_3,1} \phi_{i_3}(t_z)\right)}_{c_z^{(\alpha_2)}(t_z)} \tag{18}$$

For compactness, let

$$a_x(\cdot) = \begin{bmatrix} a_x^{(1)}(\cdot) & \cdots & a_x^{(r_1)}(\cdot) \end{bmatrix}^\top,$$

$$B_y(\cdot) = \begin{bmatrix} B_y^{(1,1)}(\cdot) & \cdots & B_y^{(1,r_2)}(\cdot) \\ \vdots & \ddots & \vdots \\ B_y^{(r_1,1)}(\cdot) & \cdots & B_y^{(r_1,r_2)}(\cdot) \end{bmatrix},$$

$$c_z(\cdot) = \begin{bmatrix} c_z^{(1)}(\cdot) & \cdots & c_z^{(r_2)}(\cdot) \end{bmatrix}^\top.$$

collect the TT factors, so that $a_x(\cdot) \in \mathbb{R}^{r_1}$, $B_y(\cdot) \in \mathbb{R}^{r_1 \times r_2}$ and $c_z(\cdot) \in \mathbb{R}^{r_2}$. Then Eq. (18) can be written in the compact form

$$f_{\mathrm{TT}}(p) = a_x(t_x)^\top B_y(t_y)\, c_z(t_z), \tag{19}$$

The TT model is differentiable since it is a multilinear form in the Bernstein bases. Differentiating (19) gives the gradient as:

$$\nabla_t f_{\mathrm{TT}}(p) = \begin{bmatrix} a'_x(t_x)^\top B_y(t_y)\, c_z(t_z) \\ a_x(t_x)^\top B'_y(t_y)\, c_z(t_z) \\ a_x(t_x)^\top B_y(t_y)\, c'_z(t_z) \end{bmatrix}, \tag{20}$$

where a'_x, B'_y, c'_z are obtained by replacing $\phi_i(t)$ with $\phi'_i(t)$ in the definitions of a_x, B_y, c_z in (18). The gradient in world coordinates follows from (11).

4. Learning the low-rank factors

The CP and TT models derived above define low-rank Bernstein RDF parameterizations. Their parameters can be obtained in two complementary ways. A first option is post-hoc tensor compression: a dense Bernstein coefficient tensor is computed or learned first, and then approximated using standard CP or TT decomposition algorithms, for instance through tensor-decomposition libraries such as TensorLy [33]. A second option is direct identification: the CP or TT factors are optimized directly from sampled SDF values, without explicitly materializing the dense tensor. Both procedures lead to the same class of low-rank Bernstein RDF models and therefore share the field and gradient evaluation formulas. In this work, we adopt the latter viewpoint and formulate factor identification as a regularized data-fitting problem.

Let $\mathcal{D} = \{(p_k, y_k)\}_{k=1}^{P}$ be a dataset of query points $p_k \in \mathbb{R}^3$ and corresponding target SDF values y_k. Given a chosen approximation order and tensor rank, the low-rank parameters are obtained by minimizing a ridge-regularized least-squares objective over mini-batches. Since the optimization problem is non-convex in the CP or TT factors, the obtained solution generally depends on initialization, sampling density and rank selection. Accordingly, the optimization stage is treated here as a practical fitting procedure rather than as a contribution on tensor-learning algorithms.

4.1. CP-RDF fitting

The CP parameters θ_{CP} are estimated by solving

$$\min_{\theta_{\mathrm{CP}}} \frac{1}{|\mathcal{B}|} \left\| f_{\mathrm{CP}}(p;\theta_{\mathrm{CP}}) - y_{\mathcal{B}} \right\|_2^2 + \rho \|\theta_{\mathrm{CP}}\|_2^2,$$

where $\mathcal{B} \subset \mathcal{D}$ denotes the current mini-batch and $\rho \geq 0$ is the regularization coefficient.

4.2. TT-RDF fitting

Similarly, the TT parameters θ_{TT} are estimated as

$$\min_{\theta_{\mathrm{TT}}} \frac{1}{|\mathcal{B}|} \left\| f_{\mathrm{TT}}(p;\theta_{\mathrm{TT}}) - y_{\mathcal{B}} \right\|_2^2 + \rho \|\theta_{\mathrm{TT}}\|_2^2.$$

In the reported experiments, all low-rank models are trained with mini-batch Adam. Although ALS- or MALS-type solvers could also be considered, Adam provided more stable behavior in our setting.

5. Exploit SDF gradients for surface projection and reactive collision avoidance

Following the RDF-based collision-avoidance strategy proposed in [4], we use the same geometric principle to demonstrate that the proposed compressed RDF models can be effectively exploited in reactive collision-avoidance algorithms.

A differentiable SDF allows projecting a generic point $p \in \mathbb{R}^3$ onto a desired level set of the distance field $S := \{p \in \mathbb{R}^3 \mid f(p) = d\}$, where $d = 0$ corresponds to the obstacle surface, while $d > 0$ defines an offset isosurface at clearance d. Starting from $p_0 := p$, the projection can be found by iteratively moving along the normalized gradient direction:

$$p_{n+1} = p_n - \alpha_n \frac{\nabla f(p_n)}{\|\nabla f(p_n)\|_2}, \qquad \alpha_n = f(p_n) - d. \tag{21}$$

and stop when $|f(p_n) - d| < \varepsilon$, returning $\hat{p} := p_n$. Assuming $\nabla f(p) \neq 0$, the displacement $(p - \hat{p})$ is aligned with the surface normal.

Given a point $p(q) \in \mathbb{R}^3$, its projection $\hat{p}$ onto the surface to an object can be computed using Eq. (21). In this way the vector that links the point p and the projection $\hat{p}$ defines the repulsion direction at the closest obstacle point.

As shown in Fig. 3, we adopt the adaptive step to a converge in one or two iterations, generally setting $\alpha_n = |f(p_n)|$. This update is meaningful when the approximated field is locally smooth and its gradient remains sufficiently aligned with the true outward normal in the neighborhood of the target level set. Given a robot with n_q degrees of freedom (DOF), let $J_r(q) \in \mathbb{R}^{3\times n_q}$ denote the geometric Jacobian of the most critical point p_r on the robot. A desired Cartesian repulsive velocity can be defined along the opposite direction of the signed-distance gradient, while its magnitude can be modulated by a Cartesian control law, e.g., a barrier function. The corresponding joint-space velocity is then obtained through the pseudoinverse of the point Jacobian as

$$\dot{q} = -J_r(q)^{\dagger} \nabla f(p_r) \frac{1}{f_{\ell}(p_r) - d^{\mathrm{safe}}}, \tag{22}$$

where $f_{\ell}(p_r)$ denotes the signed distance evaluated at the critical point p_r belonging to link ℓ and d^{safe} is the prescribed safety distance. The Jacobian $J_r(q)$ is the point Jacobian associated with p_r, i.e., computed up to the link ℓ involved in the closest interaction.

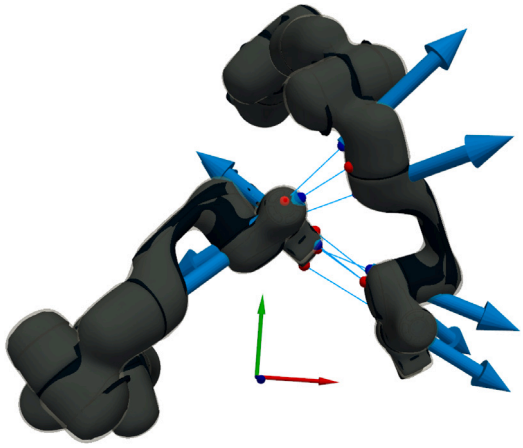


Fig. 3. Projection of collision points using CP as approximation function. The red points are the nearest collision points, while blue points on the robot are the points to move away. Blue arrows represent the repulsion gradients. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Analytical cost model and GPU regime

Building on the CP/TT factorizations of the Bernstein SDF, we estimate their GPU evaluation cost relative to the dense model through a roofline-inspired performance model [34,35], parameterized by the batch size P and the polynomial order N. The dense Bernstein model requires $\mathcal{O}(PN^3)$ floating-point operations, whereas the CP and TT formats reduce this cost to $\mathcal{O}(PRN)$ and $\mathcal{O}(PN(r_1+r_1r_2+r_2))$, respectively. In the common case $r_1 \approx r_2 \approx r$, the TT complexity simplifies to $\mathcal{O}(PNr^2)$.

Define the number of parameter counts as:

$$\Gamma_{\mathrm{w}} = N^3, \quad \Gamma_{\mathrm{CP}} = 3NR + R, \quad \Gamma_{\mathrm{TT}} = N(r_1 + r_1r_2 + r_2) \tag{23}$$

On modern GPUs, kernel runtime is constrained by both compute throughput C_{peak} (FLOP/s) and memory bandwidth B_{width} (bytes/s). The following analysis is not intended as an exact runtime predictor, but as a compact interpretative model to explain how batch size, polynomial order and decomposition rank affect the dominant GPU regime. Following a roofline-inspired model, the batch execution time is approximated as

$$T(P) \approx \max\left(\underbrace{\frac{F(P)}{C_{\mathrm{peak}}}}_{\text{compute}}, \underbrace{\frac{M_{\mathrm{fix}} + M_{\mathrm{stream}}(P)}{B_{\mathrm{width}}}}_{\text{memory}} \right), \tag{24}$$

where $F(P)$ is the total number of floating-point operations for a batch of size P, M_{fix} is the batch-independent cost of loading model parameters and $M_{\mathrm{stream}}(P)$ accounts for per-query data movement such as basis evaluations and input/output transfers.

In practice, the measured time also depends on implementation-level effects. To obtain a simpler expression for comparison and break-even analysis, we replace the max-form roofline model in Eq. (24) with its standard additive upper bound [36]:

$$T(P) \leq \frac{F(P)}{C_{\mathrm{peak}}} + \frac{M_{\mathrm{fix}} + M_{\mathrm{stream}}(P)}{B_{\mathrm{width}}}. \tag{25}$$

This upper bound is adopted only to derive a tractable affine surrogate just for comparison across dense and low-rank models. To account for deviations from the idealized roofline surrogate, we augment the model with a batch-independent overhead $T_{0,m}$ at small-batch regime and an

implementation-dependent term $\Delta_{\mathrm{impl},m}(P)$:

$$T^{\mathrm{aff}}_{\{\cdot\}}(P) \approx \frac{F_{\{\cdot\}}(P)}{C_{\mathrm{peak}}} + \frac{M_{\mathrm{fix},\{\cdot\}} + M_{\mathrm{stream},\{\cdot\}}(P)}{B_{\mathrm{width}}} + \Delta_{\mathrm{impl},\{\cdot\}}(P), \tag{26}$$

with

$$\Delta_{\mathrm{impl},\{\cdot\}}(P) = \delta_{P,\{\cdot\}} P + \delta_{0,\{\cdot\}}. \tag{27}$$

where $\delta_{P,\{\cdot\}}$ and $\delta_{0,\{\cdot\}}$ are terms accounting for implementation-dependent deviations from the simplified roofline model, respectively in the per-query slope and in the constant offset. For the small-batch regime, we define the clock surrogate as

$$T_{\{\cdot\}}(P) = \max\left(T_{0,\{\cdot\}}, T^{\mathrm{aff}}_{\{\cdot\}}(P)\right), \tag{28}$$

where $T^{\mathrm{aff}}_{m}(P)$ denotes the affine model obtained after adding the residual correction.

Let $c_{\mathrm{w}}, c_{\mathrm{cp}}, c_{\mathrm{tt}}$ denote the FLOP constants for dense, CP and TT evaluation, respectively. Let $s_{\mathrm{w}}, s_{\mathrm{cp}}, s_{\mathrm{tt}}$ denote the memory traffic per dense coefficient, CP parameter and TT parameter, respectively. Moreover, let s_ϕ denote the cost associated with loading the three Bernstein basis rows and let $s_o P$ collect the output writes together with any additional per-query auxiliary traffic scaling linearly with the batch size.

Then, we write

$$\begin{aligned} &F_w(\cdot) = c_w \Gamma_w P, \quad F_{\mathrm{CP}}(\cdot) = c_{\mathrm{cp}} \Gamma_{\mathrm{cp}} P, \quad F_{\mathrm{TT}}(\cdot) = c_{\mathrm{tt}} \Gamma_{\mathrm{tt}} P, \\ &M_{\mathrm{fix},w} = s_w \Gamma_w, \quad M_{\mathrm{fix,CP}} = s_{\mathrm{cp}} \Gamma_{\mathrm{cp}}, \quad M_{\mathrm{fix,TT}} = s_{\mathrm{tt}} \Gamma_{\mathrm{tt}}, \\ &M_{\mathrm{stream},w} = M_{\mathrm{stream,CP}} = M_{\mathrm{stream,TT}} \approx 3 s_\phi N P + s_o P. \end{aligned} \tag{29}$$

For simplicity, we assume that the per-query streaming traffic $M_{\mathrm{stream},\{\cdot\}}$ is dominated by basis loading and output writes and is therefore approximately the same for dense, CP and TT kernels. Substituting (29) into Eq. (25), we define the idealized coefficients

$$\begin{aligned} \alpha_{\{\cdot\}} &= \frac{c_{\{\cdot\}}}{C_{\mathrm{peak}}} + \frac{3 s_\phi}{B_w} \eta^1_{\{\cdot\}} + \frac{s_o}{B_w} \eta^2_{\{\cdot\}}, \\ \beta_{\{\cdot\}} &= \frac{s_{\{\cdot\}}}{B_w}, \end{aligned} \tag{30}$$

where $\{\cdot\}$ indicates the model-dependent factor and

$$\begin{aligned} \eta^1_w &= \frac{1}{N^2}, & \eta^2_w &= \frac{1}{N^3}, \\ \eta^1_{\mathrm{cp}} &= \frac{N}{3NR + R}, & \eta^2_{\mathrm{cp}} &= \frac{1}{3NR + R}, \\ \eta^1_{\mathrm{tt}} &= \frac{1}{r_1 r_2 + r_1 + r_2}, & \eta^2_{\mathrm{tt}} &= \frac{1}{N(r_1 r_2 + r_1 + r_2)}. \end{aligned}$$

Let T_w, T_{CP} and T_{TT} denote the execution times of the dense Bernstein, CP and TT models, respectively. Under the idealized affine surrogate, the measured runtime also depends on implementation-level effects that are not explicitly represented in the simplified roofline model, such as kernel-launch overhead, operator dispatch, backend selection, synchronization, tensor layout and possible materialization of intermediate tensors expressed in Eqs. (27) and (28).

Accordingly, the implementation-aware affine surrogate becomes

$$\begin{aligned} T_w &\approx \alpha_w \Gamma_w P + \beta_w \Gamma_w + \delta_{P,w} P + \delta_{0,w} \\ T_{\mathrm{CP}} &\approx \alpha_{\mathrm{cp}} \Gamma_{\mathrm{cp}} P + \beta_{\mathrm{cp}} \Gamma_{\mathrm{cp}} + \delta_{P,\mathrm{cp}} P + \delta_{0,\mathrm{cp}} \\ T_{\mathrm{TT}} &\approx \alpha_{\mathrm{tt}} \Gamma_{\mathrm{tt}} P + \beta_{\mathrm{tt}} \Gamma_{\mathrm{tt}} + \delta_{P,\mathrm{tt}} P + \delta_{0,\mathrm{tt}} \end{aligned} \tag{31}$$

Equivalently, defining the effective coefficients

$$\bar{\alpha}_{\{\cdot\}} = \alpha_{\{\cdot\}} + \frac{\delta_{P,\{\cdot\}}}{\Gamma_{\{\cdot\}}}, \qquad \bar{\beta}_{\{\cdot\}} = \beta_{\{\cdot\}} + \frac{\delta_{0,\{\cdot\}}}{\Gamma_{\{\cdot\}}}, \tag{32}$$

we can rewrite (31) in compact form as

$$\begin{aligned} T_w &\approx \bar{\alpha}_w \Gamma_w P + \bar{\beta}_w \Gamma_w, \\ T_{\mathrm{CP}} &\approx \bar{\alpha}_{\mathrm{cp}} \Gamma_{\mathrm{cp}} P + \bar{\beta}_{\mathrm{cp}} \Gamma_{\mathrm{cp}}, \\ T_{\mathrm{TT}} &\approx \bar{\alpha}_{\mathrm{tt}} \Gamma_{\mathrm{tt}} P + \bar{\beta}_{\mathrm{tt}} \Gamma_{\mathrm{tt}}. \end{aligned} \tag{33}$$

For sufficiently large batch sizes P, fixed costs are amortized and the runtime is increasingly governed by the effective per-query slope

$$\tilde{\alpha}_{\{\cdot\}} = \alpha_{\{\cdot\}} \Gamma_{\{\cdot\}} + \delta_{P,\{\cdot\}}.$$

This does not necessarily correspond to a strictly compute-bound regime, since $\tilde{\alpha}_{\{\cdot\}}$ still includes both arithmetic and streaming contributions. Therefore, the break-even conditions derived below should be interpreted as large-batch asymptotic guidelines under the calibrated surrogate model, rather than as exact predictors of the observed crossover points.

Proposition 1 (*Large-batch Break-even Conditions*)**.** *For sufficiently large batch sizes, the batch-independent terms in (31) become negligible with respect to the terms linear in P. Under the implementation-aware surrogate the following asymptotic speedup conditions hold:*

$$\begin{aligned} &\tilde{\alpha}_{\mathrm{cp}} < \tilde{\alpha}_w \;\Rightarrow\; T_{\mathrm{CP}} < T_w \quad \textit{for sufficiently large } P, \\ &\tilde{\alpha}_{\mathrm{tt}} < \tilde{\alpha}_w \;\Rightarrow\; T_{\mathrm{TT}} < T_w \quad \textit{for sufficiently large } P, \\ &\tilde{\alpha}_{\mathrm{tt}} < \tilde{\alpha}_{\mathrm{cp}} \;\Rightarrow\; T_{\mathrm{TT}} < T_{\mathrm{CP}} \quad \textit{for sufficiently large } P. \end{aligned} \tag{34}$$

Using the conditions in Eq. (23) these inequalities become

$$\begin{aligned} &\alpha_{\mathrm{cp}}(3NR + R) + \delta_{P,\mathrm{cp}} < \alpha_w N^3 + \delta_{wP}, \\ &\alpha_{\mathrm{tt}} N(r_1 r_2 + r_1 + r_2) + \delta_{P,\mathrm{tt}} < \alpha_w N^3 + \delta_{P,w} \\ &\alpha_{\mathrm{tt}} N(r_1 r_2 + r_1 + r_2) + \delta_{P,\mathrm{tt}} < \alpha_{\mathrm{cp}}(3NR + R) + \delta_{P,\mathrm{cp}}. \end{aligned} \tag{35}$$

In the idealized case, for a fixed Bernstein order N, rank selection is the key factor advantage of low-rank representations over the dense Bernstein model. In particular, suppose that the implementation-dependent corrections are equal or negligible across methods, i.e., $\delta_w \simeq \delta_{\mathrm{cp}} \simeq \delta_{\mathrm{tt}}$, and assuming comparable kernel efficiencies $\alpha_w \simeq \alpha_{\mathrm{cp}} \simeq \alpha_{\mathrm{tt}}$, (35) reduces to

$$\begin{aligned} R < \frac{N^3}{3N + 1} \quad &\Rightarrow\; T_{\mathrm{CP}} < T_w, \\ r_1 r_2 + r_1 + r_2 < N^2 \quad &\Rightarrow\; T_{\mathrm{TT}} < T_w, \\ r_1 r_2 + r_1 + r_2 < R\left(3 + \frac{1}{N}\right) \quad &\Rightarrow\; T_{\mathrm{TT}} < T_{\mathrm{CP}}. \end{aligned} \tag{36}$$

Remark 2. The simplified inequalities in (36) should be interpreted as idealized large-batch design guidelines. In practice, the observed crossover points depend on the implementation-dependent corrections δ_m and on the batch-independent overheads $T_{0,m}$ and are therefore not determined by parameter counts alone.

Remark 3. In our PyTorch-based implementation, the effective per-query coefficient includes not only FLOP counts and streaming traffic, but also implementation-level effects such as operator dispatch, backend selection, tensor layout and the possible materialization of intermediate tensors. For a fixed software stack and hardware platform, these effects can be absorbed into method-dependent constants δ_w, δ_{cp} and δ_{tt}, yielding a calibrated surrogate model for practical speedup prediction.

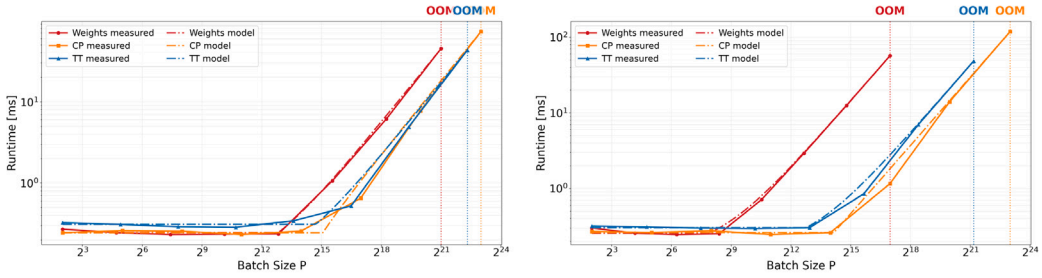
7. Results

The results were obtained by benchmarking the same PyTorch-based kernels on two GPUs with different compute–memory balances, namely a NVIDIA GeForce RTX 3080 Laptop GPU with a 12th Gen Intel Core i7-12700H CPU and an NVIDIA GeForce RTX 4060 Ti with a 12th Gen Intel Core i7-12700 CPU. Based on datasheet specifications,[1] the RTX 3080 $\{C_{\mathrm{peak}} \approx 29.8\,\mathrm{TFLOP/s}, B_w \approx 760\,\mathrm{GB/s}\}$ has a lower compute-to-bandwidth ratio than the RTX 4060 Ti $\{C_{\mathrm{peak}} \approx 22\,\mathrm{TFLOP/s}, B_w \approx 288\,\mathrm{GB/s}\}$. Therefore, the same kernel is expected to remain more affected by memory traffic over a wider batch-size range

[1] NVIDIA datasheets, https://www.techpowerup.com/gpu-specs/.

Table 1
Parameters of the implementation-aware runtime surrogate for the two representative settings shown in Figs. 4(a) and 4(b) for a single SDF object.

Method	Γ_m	$\alpha_{\{\cdot\}}$	$\beta_{\{\cdot\}}$	$\delta_{P,\{\cdot\}}$	$\delta_{0,\{\cdot\}}$	T_0 [ms]	MAE [ms]
Weights	512	7.691×10^{-13}	1.389×10^{-11}	1.606×10^{-08}	2.239×10^{-04}	0.2637	0.0457
Weights	13 824	1.643×10^{-13}	1.389×10^{-11}	4.429×10^{-07}	1.874×10^{-04}	0.0030	0.1103
cp	600	6.696×10^{-13}	1.389×10^{-11}	1.936×10^{-09}	2.851×10^{-04}	0.2478	0.0356
cp	2336	5.249×10^{-13}	1.389×10^{-11}	2.596×10^{-10}	2.401×10^{-04}	0.2440	0.0086
tt	1344	3.493×10^{-13}	1.389×10^{-11}	4.150×10^{-09}	2.335×10^{-04}	0.2800	0.0171
tt	4032	3.424×10^{-13}	1.389×10^{-11}	4.230×10^{-09}	2.860×10^{-04}	0.3123	0.0361



(a) Low-order setting: weights $N = 8$, CP $N = 8, R = 24$, TT $N = 8, r_1 = r_2 = 12$.

(b) Higher-order setting: weights $N = 24$, CP $N = 24, R = 32$, TT $N = 24, r_1 = r_2 = 12$.

Fig. 4. Measured and implementation-aware modeled runtime versus batch size P for dense weights, CP and TT RDF evaluation for a single SDF object. Solid lines denote measured runtimes, dash-dotted lines denote the calibrated surrogate and vertical dotted lines mark the first out-of-memory (OOM) batch.

on the RTX 4060 Ti, whereas the RTX 3080 offers a more favorable arithmetic-throughput-to-bandwidth balance. Since the same PyTorch-based kernels were used on both devices, the measured runtimes also reflect implementation-level effects such as operator dispatch, tensor layout and intermediate tensor materialization.

7.0.1. Runtime trends and surrogate-model fit

Using Eq. (28), Figs. 4(a) and 4(b) show that the proposed implementation-aware surrogate captures the main runtime trends of dense and compressed RDF evaluation across batch sizes on the RTX 4060 Ti setup. In both settings, the small-batch regime is dominated by an approximately constant latency floor, whereas for larger batches the runtime is increasingly governed by the effective per-query cost. In the low-order setting shown in Fig. 4(a), the three methods remain relatively close over a broad range of batch sizes and the advantage of compression becomes clearly visible only for larger P. In the higher-order setting of Fig. 4(b), this advantage becomes much more pronounced: the dense weights model grows rapidly in cost and reaches OOM at much smaller batch sizes, whereas CP and TT remain executable over a substantially wider range. Among the compressed models, CP achieves the lowest runtime in these experiments, while TT remains competitive and offers better memory scalability.

Table 1 reports the calibrated surrogate parameters for the two settings. Here, Γ_m is the method-dependent parameter count, $\alpha_{\{\cdot\}}$ and $\beta_{\{\cdot\}}$ are the coefficients of the idealized roofline-inspired surrogate, $\delta_{P,\{\cdot\}}$ and $\delta_{0,\{\cdot\}}$ account for implementation-level deviations from the ideal model and T_0 captures the small-batch latency floor. The last column reports the mean absolute error between estimated and measured runtimes over the tested batch sizes. The low affine MAE values show that the calibrated surrogate follows the measured trends well, especially for the compressed models and thus provides a practical interpretation of the observed crossover behavior. At the same time, the calibrated implementation-dependent corrections differ across methods, indicating that the idealized large-batch inequalities of Eq. (36) should be interpreted only as first-order guidelines for selecting the compression rank. Still, if the kernel implementation is not pathological and the empirical corrections scale consistently with the method-dependent parameter counts, the idealized conditions remain informative and can serve as a practical preliminary criterion for choosing the compression rank.

7.1. Computational results

Table 2 illustrates the trade-off between approximation accuracy, computational efficiency and batch scalability across the considered continuous robot distance representations in literature. All locally reproduced runtime and maximum-batch results in Table 2 were obtained on the NVIDIA GeForce RTX 3080 Laptop GPU setup described above, whereas the SDF-SC values were taken from the original paper and should therefore not be interpreted as being measured under identical hardware conditions. From the accuracy perspective, the CP-RDF family yields the strongest results overall, with CP-RDF ($N = 64$, $R = 64$) achieving the lowest reported RMSE of 0.5 mm.. Among the baseline methods, SDF-SC and RNDF provide the most competitive accuracy values, with RNDF showing consistently low approximation error across both model sizes. From the computational perspective, Neural-JSDF provides the most favorable performance in the small-batch regime, where it achieves the lowest latency and highest throughput among the directly measured methods. In the medium- and large-batch regimes, CP-RDF ($N = 16, R = 32$) becomes the most efficient method, attaining the best overall throughput at 10k and 100k queries. Finally, with respect to scalability, RNDF exhibits the highest executable batch capacity, particularly for $K = 64$, which supports substantially larger batches than the other evaluated methods, followed by RNDF ($K = 128$) and Neural-JSDF. This trend is consistent with the fact that these methods process batches at the level of robot-point queries, rather than explicitly expanding computation over individual robot links as in per-link formulations. The public RMMI configuration appears less favorable in both accuracy and runtime; moreover, its timing results should be interpreted as proxy measurements, since the publicly available setup does not include the original checkpoints required for a fully matched end-to-end evaluation.

Tables 3 and 4, together with Fig. 5, summarize the accuracy–memory–runtime trade-off of the dense, CP and TT representations.

Table 2
Quantitative comparison of continuous Panda robot distance representations in terms of approximation accuracy and computational cost. The first table reports RMSE in millimeters, while the second table reports GPU inference time in milliseconds, throughput in queries per millisecond, and the maximum executable batch size. Results for approximation are taken from related GitHub repositories, while for computational results only SDF-SC is taken from the original paper since no repository is available; RMMI is obtained from the public iSDF setup [6].

Accuracy comparison	
Method	RMSE [mm]
Neural-JSDF [3]	14.61
RobotSDF [5]	6.38
Neural CDF [25]	18.90
RNDF ($K = 64$)	2.49
RNDF ($K = 128$)	1.86
SDF-SC [8]	1.6
RMMI [21]	20.7
RDF ($N = 8$)	4.8
CP-RDF ($N = 8, R = 32$)	4.9
CP-RDF ($N = 64, R = 32$)	0.7
TT-RDF ($N = 64, r = 32, 32$)	0.8

Computational comparison							
Method	1k		10k		100k		Max pts
	ms	q/ms	ms	q/ms	ms	q/ms	
Neural-JSDF	0.304	3286	1.122	8912	9.089	11 002	3.2M
SDF-SC	1.05	952	1.21	8264	n/a	n/a	n/a
RDF ($N = 8$)	1.27	787	2.12	4717	OOM	OOM	20k
RNDF ($K = 64$)	1.540	649	2.897	3452	26.443	3782	12.05M
RNDF ($K = 128$)	2.340	427	3.092	3235	32.265	3099	3.2M
RMMI†	0.471	2123	2.126	4704	22.034	4538	1.6M
CP-RDF ($N = 16, R = 32$)	0.327	3058	1.1	9091	9.0	11 111	2.2M
TT-RDF ($N = 16, r_1 = r_2 = 12$)	0.692	1445	1.6	6250	12.92	7740	1.9M

Table 3
Approximation errors of the dense Bernstein model w and its CP and TT low-rank representations for four polynomial orders N. The dense baseline is reported once for each N, since it is independent of the CP rank R_{cp} and TT rank r_{tt}. Errors are reported in mm.

N	Dense w			R_{cp}	CP			r_{tt}	TT		
	MAE [mm]	RMSE [mm]	Max [mm]		MAE [mm]	RMSE [mm]	Max [mm]		MAE [mm]	RMSE [mm]	Max [mm]
8	4.4	4.9	40.0	8	4.8	5.8	42.8	(4, 4)	4.5	5.2	41.0
				16	4.2	5.3	41.7	(8, 8)	4.1	5.0	42.7
16	3.1	3.7	40.2	16	3.2	3.9	40.0	(8, 8)	3.3	3.9	40.0
				32	3.1	3.8	33.5	(12, 12)	3.2	3.8	34.0
24	0.9	1.5	30.1	24	1.0	1.5	31.1	(12, 12)	1.0	1.6	30.7
				48	0.9	1.5	30.5	(24, 24)	0.9	1.5	30.6
64	OOM	OOM	OOM	32	0.5	0.8	5.9	(16, 16)	0.6	0.8	6.2
				64	0.4	0.5	4.2	(32, 32)	0.5	0.7	4.6

Table 3 shows that the tensor rank acts as a direct control parameter for the approximation fidelity: increasing R_{cp} or r_{tt} generally brings the compressed models closer to the corresponding dense Bernstein representation, while moderate ranks already preserve comparable SDF accuracy.

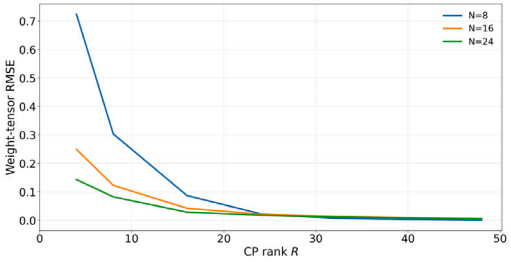
As the number of Bernstein bases increases, the dense model becomes more accurate, but its cubic parameter growth, $\Gamma_w = N^3$, quickly makes it impractical in GPU memory. This is already evident at $N = 64$, where the dense model is out of memory, while CP and TT remain feasible and accurate. Compression is therefore beneficial not only for runtime, but also because it enables finer resolutions under the same hardware budget. Fig. 5 confirms and clarifies these trends. For each polynomial order N, the dense Bernstein coefficient tensor is first used as the reference and then approximated over a sweep of CP and TT ranks. Figs. 5(a) and 5(b) report the RMSE between the dense coefficients and the tensor reconstructed from the corresponding low-rank factors, showing that the reconstruction error decreases as the rank increases. The lower panels, Figs. 5(c) and 5(d), also show the corresponding cost of increasing the rank: peak GPU memory grows with the model complexity, but remains well below the dense reference, as indicated by the dashed line, for the considered polynomial orders. The rank therefore provides a practical design parameter for balancing approximation accuracy against memory consumption and runtime.

At the same time, Table 4 shows that these compressed models retain a substantially lower evaluation cost, with the advantage becoming more pronounced as the polynomial order and query-set size increase. CP is generally faster at the tested operating points, whereas TT provides an alternative trade-off when sufficient accuracy can be achieved with small TT ranks. The timing results should be interpreted together with the approximation settings. For each polynomial order, the CP and TT ranks were selected as representative operating points providing practical accuracy–cost trade-offs. For example, at $N = 8$ with $R = 8$ and $(r_1, r_2) = (6, 6)$ one has $N^3 = 512$ and $3NR + R = 200 < N^3$, while $r_1 r_2 + r_1 + r_2 = 48 < N^2 = 64$. Therefore, both idealized sufficient break-even conditions in Proposition 1 are satisfied. Nevertheless, for very small batch sizes, fixed and implementation-dependent overheads can have a comparatively larger impact on runtime, so the benefit of the reduced asymptotic per-query cost may remain only weakly visible. As the number of query points increases, these costs are progressively amortized and the advantage of the compressed representations becomes more apparent.

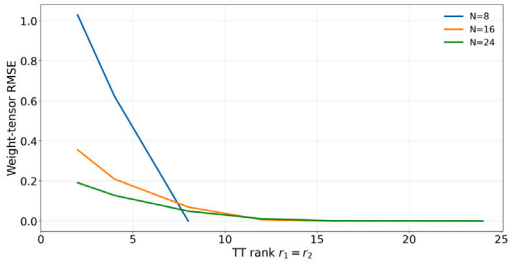
This trend becomes much stronger as N increases. Moving from $N = 8$ to $N = 16$ and then to $N = 24$, the dense model rapidly becomes more expensive, whereas CP and TT preserve a much lower evaluation cost for ranks that still maintain good fidelity. As a result, the speedups increase significantly with resolution, showing that the

Table 4
Computation times of the dense Bernstein model w and its CP and TT low-rank representations on two GPUs. For each polynomial order N, the dense timing t_w is reported once, since it is independent of R_{cp} and r_{tt}. In each cell, A is reported on the first line and B on the second line, with point-cloud sizes $(A, B) = (2\times10^3, 7\times10^3)$. All timings are reported in ms. *Note:* OOM denotes out of memory.

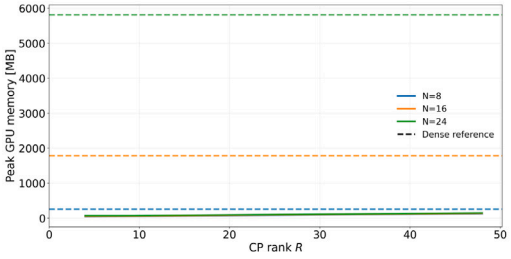
N	R_{cp}	r_{tt}	RTX 3080			RTX 4060Ti		
			t_w [ms]	t_{cp} [ms]	t_{tt} [ms]	t_w [ms]	t_{cp} [ms]	t_{tt} [ms]
8	8	(6, 6)	A: 0.4 B: 0.9	A: 0.3 B: 0.5	A: 0.4 B: 0.5	A: 0.3 B: 1.4	A: 0.3 B: 0.4	A: 0.3 B: 0.4
	32	(8, 8)		A: 0.3 B: 0.6	A: 0.3 B: 0.7		A: 0.3 B: 0.5	A: 0.3 B: 0.5
16	16	(8, 8)	A: 1.3 B: 5.9	A: 0.3 B: 0.6	A: 0.4 B: 0.9	A: 2.1 B: 9.1	A: 0.3 B: 0.4	A: 0.4 B: 0.9
	32	(12, 12)		A: 0.3 B: 0.7	A: 0.4 B: 1.1		A: 0.3 B: 0.5	A: 0.4 B: 1.1
24	24	(12, 12)	A: 4.7 B: 19.1	A: 0.5 B: 0.8	A: 0.5 B: 1.0	A: 6.7 B: 26.6	A: 0.5 B: 0.6	A: 0.5 B: 1.1
	48	(24, 24)		A: 0.3 B: 0.8	A: 0.5 B: 1.6		A: 0.3 B: 0.6	A: 0.4 B: 1.6
64	32	(16, 16)	A: OOM B: OOM	A: 1.1 B: 4.2	A: 5.9 B: 26.8	A: OOM B: OOM	A: 1.2 B: 4.4	A: 6.1 B: 15.7
	64	(32, 32)		A: 1.5 B: 7.7	A: 11.7 B: 50.4		A: 1.6 B: 7.8	A: 11.6 B: 52.5



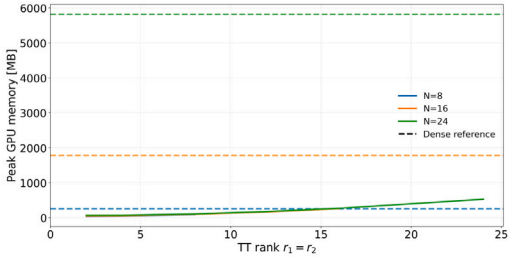
(a) Representation error of the CP- with respect to the corresponding dense Bernstein model, reported as weight RMSE for different ranks.



(b) Representation error of the TT- with respect to the corresponding dense Bernstein model, reported as weight RMSE for different ranks.



(c) Peak GPU memory usage of dense, CP-RDF models as a function of polynomial order and compression rank.



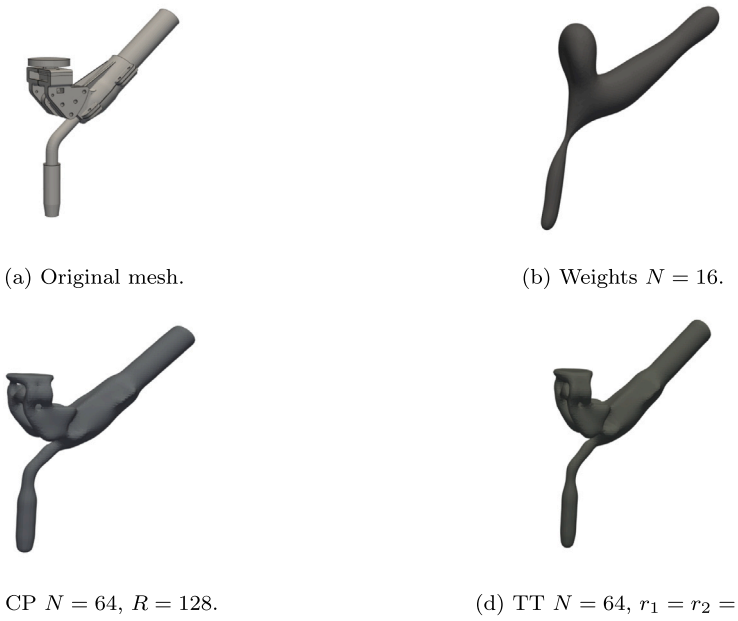
(d) Peak GPU memory usage of dense, TT-RDF models as a function of polynomial order and compression rank.

Fig. 5. Compression accuracy and memory requirements of the CP- and TT-based Bernstein RDF representations for different polynomial orders and tensor ranks. The top row reports the coefficient-tensor reconstruction error with respect to the corresponding dense Bernstein model, measured as weight RMSE. The bottom row reports the peak GPU memory usage during inference, with the dense model included as a reference.

practical advantage of compression becomes more pronounced as the basis size grows.

Within the operating points of Table 4, CP often remains faster than TT. However, this should not be interpreted as a general superiority of CP, since the table is primarily dominated by the comparison against the dense baseline. Its main message is that both compressed formats strongly outperform the dense representation at sufficiently large polynomial orders or batch sizes, while TT mainly offers better storage efficiency and lower peak memory, enabling larger feasible batch sizes.

A clearer CP–TT comparison is given in Table 5, where both compressed welding-torch models are evaluated at $N = 64$, with CP rank



(a) Original mesh. (b) Weights $N = 16$.

(c) CP $N = 64$, $R = 128$. (d) TT $N = 64$, $r_1 = r_2 = 12$.

Fig. 6. Visual comparison of a welding torch with different parameter configurations.

Table 5
Compact CP–TT comparison for Fig. 6 at $N = 64$, with CP rank $R = 128$ and TT ranks $(12, 12)$. Measured on cuda in SDF only mode under a 10.88 GiB memory budget.

Metric	CP/TT	Best
Runtime memory	193.0/123.0 KiB	TT (1.57× smaller)
Peak memory @ 100k	320.3/195.1 MiB	TT (1.64× lower)
Eval. time @ 100k [ms]	4.87/3.76	TT (1.29× faster)
Max query points	2.22M/3.39M	TT (1.53× more)

$R = 128$ and TT ranks $r_1 = r_2 = 12$. In this setting, the large-batch condition of Proposition 1 favors TT, since $r_1 r_2 + r_1 + r_2 = 168 < R\left(3 + \frac{1}{N}\right) \approx 386$. Thus, the proposition provides a useful first criterion for rank selection. The measurements in Table 5 confirm this prediction: TT requires less model memory, reduces peak memory at 100k queries, achieves a lower evaluation time, and supports a larger maximum batch size under the same GPU memory budget. Therefore, for this operating point, TT is preferable to CP, not because it is generally faster, but because the selected ranks satisfy the break-even condition and yield a better memory–runtime scalability trade-off.

The same advantage is visible in Fig. 6: while the dense model cannot be evaluated at $N = 64$ under the considered memory constraints, the compressed CP and TT models remain feasible at this polynomial order, producing zero-level surfaces closer to the original mesh.

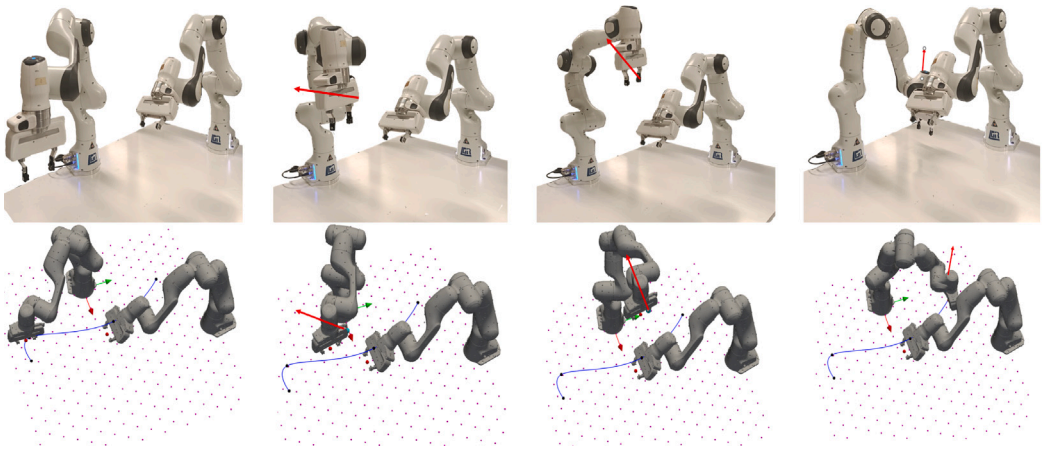
7.2. Reactive collision-avoidance validation with CP- and TT-RDF

To assess the practical applicability of the proposed framework in reactive collision avoidance, we performed dual-arm experiments using both TT-RDF and CP-RDF robot models, as shown in Figs. 7 and 9. The purpose of this test is to assess whether the newly introduced CP- and TT-based RDF representations preserve, during reactive collision avoidance, the same clearance-maintaining behavior previously demonstrated for the dense RDF formulation. To this end, we employ the control framework introduced in [4]. Based on the approximation results reported in Table 3, we selected configurations that provide accurate distance reconstruction while keeping the online evaluation cost compatible with the control-loop timing requirements. For the collision-map construction, points were sampled over the surfaces of all robot links, together with additional samples on the supporting table, resulting in a world point cloud of approximately 1000 points queried by each link. The RDF-based collision-avoidance computation alone remained below 1 ms for all the considered representations. The complete control-cycle time, which also includes the remaining control computations, data transfers, and GPU synchronization overheads. As discussed in [4], the complete control framework using the dense RDF baseline operated at approximately 400 Hz, whereas the lower evaluation cost of the CP- and TT-based representations enabled average update rates in the 500–750 Hz range. These values should be interpreted as measured average rates rather than hard real-time guarantees because of GPU timing variability.

As shown in the right Figs. 8 and 10, the activation matrix was designed to progressively increase the contribution of the collision-avoidance task as the distance decreases below 0.25 m, reaching full activation at 0.10 m. A safety threshold 0.05 m was then imposed through the barrier function of Eq. (22).

Fig. 7 illustrates the collision-avoidance experiment executed along the same nominal path using both the dense and TT-RDF representations. In this scenario, one Panda robot is assigned a reference trajectory, while the second robot is placed in the workspace and acts as a static obstacle. As shown in Fig. 7(a), the moving robot is initially far from the obstacle and therefore the collision-avoidance action is inactive. As the robot approaches the critical region, the repulsive action is activated, as visible in Fig. 7(b). The corresponding gradient, shown by the red arrow, generates a repulsive effect on the robot and starts deforming the nominal motion. In Fig. 7(c), the robot is in the active avoidance phase: the repulsive gradient acts on the hand/finger region and pushes the robot upward, causing a deviation from the nominal path in order to preserve a safe separation from the obstacle. Finally, once the obstacle has been bypassed and the distance becomes larger than the activation threshold, the repulsive action vanishes and the robot resumes the original trajectory, completing the assigned motion, as shown in Fig. 7(d).

The same behavior can be observed in Fig. 8, where the TCP plots 8(a)–8(b) show the deviation from the nominal trajectory at $t = 4$ s and its subsequent recovery at $t \sim 15$ s, while the corresponding minimum robot–obstacle distances are reported in 8(c)–8(d). Since the two RDF representations provide slightly different approximations of



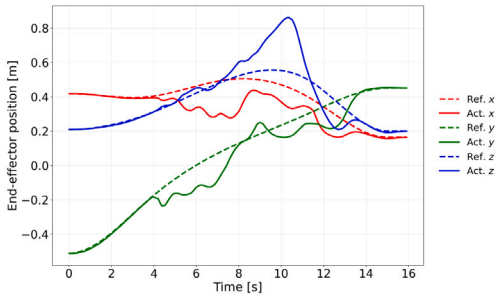
(a) The robot is far from the obstacle, so collision avoidance is inactive.

(b) As the robots approach each other, collision avoidance is activated.

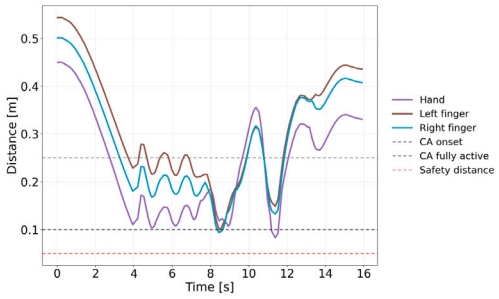
(c) The moving robot deviates from the nominal trajectory to maintain safe separation.

(d) Once the obstacle is cleared, the robot resumes and completes the reference motion.

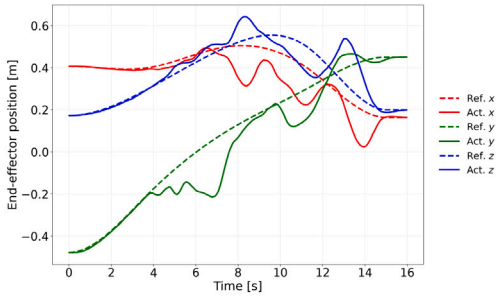
Fig. 7. Representative stages of the dual-arm collision-avoidance experiment using the dens- and TT-RDF model. One Panda robot follows the assigned trajectory, while the other remains fixed and acts as an obstacle. As the robots approach each other, repulsive actions are activated, causing a temporary deviation from the nominal path to maintain safe separation. The top row shows the real setup and the bottom row the corresponding schematic view with trajectories, point clouds and repulsive gradients. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



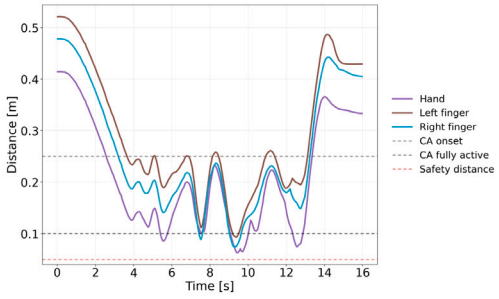
(a) TCP trajectory obtained with the dense-based method, compared with the reference trajectory.



(b) Robot-link distance and corresponding Task-Priority activation values for the weights-based method.



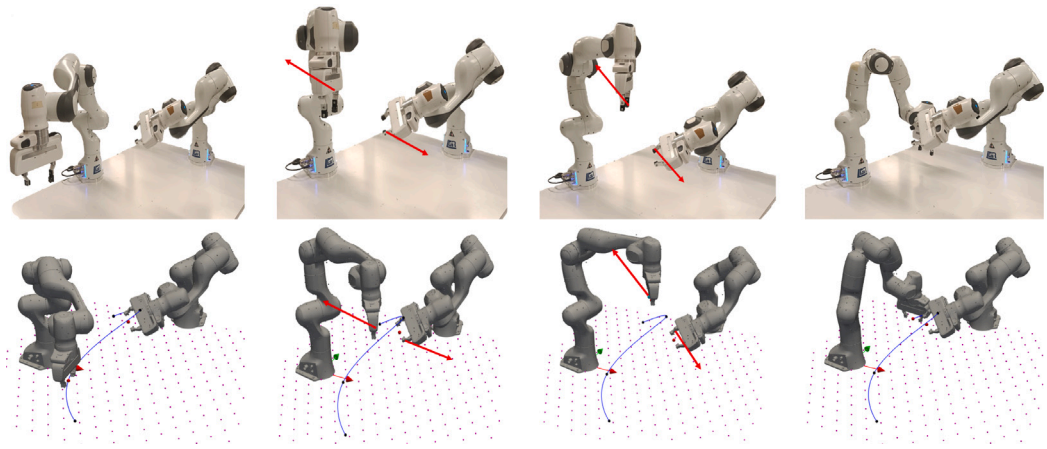
(c) TCP trajectory obtained with the TT-based method, compared with the reference trajectory.



(d) Robot-link distance and corresponding Task-Priority activation values for the GP-based method.

Fig. 8. Comparison between the dense-based and TT-based methods based on task execution of Fig. 7. The first column reports the TCP trajectory together with the reference trajectory, while the second column shows the robot-link distance and the corresponding activation values used within the Task-Priority framework.

the distance field and its gradient, the resulting avoidance responses and TCP trajectories are not identical. Nevertheless, in both cases the minimum distance remains above the prescribed safety threshold throughout the motion, confirming that the gradients generated by both representations produce an effective repulsive action and preserve the required clearance from the obstacle.



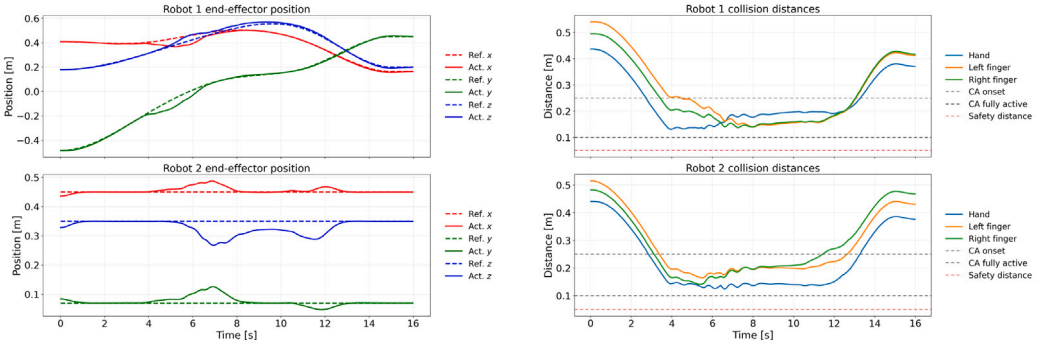
(a) The robot is far from the obstacle, so collision avoidance is inactive.

(b) As the robots approach each other, collision avoidance is activated.

(c) The moving robot deviates from the nominal trajectory to maintain safe separation.

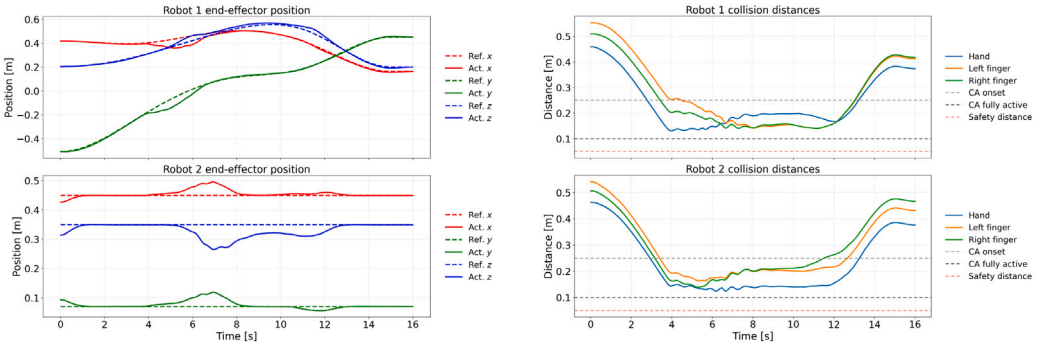
(d) Once the obstacle is cleared, the robot resumes and completes the reference motion.

Fig. 9. Representative stages of the dual-arm collision-avoidance experiment using the CP-RDF model. One Panda robot follows the assigned trajectory, while the other remains fixed and acts as an obstacle. As the robots approach each other, repulsive actions are activated, causing a temporary deviation from the nominal path to maintain safe separation. The top row shows the real setup and the bottom row the corresponding schematic view with trajectories, point clouds and repulsive gradients.



(a) TCP evolution for both robot under dense collision method

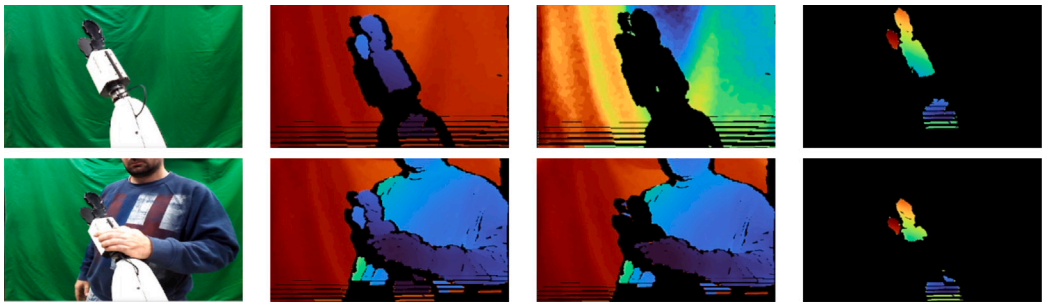
(b) Robot-hand distance evolution using the dense-based formulation.



(c) TCP evolution for both robot under CP- collision method

(d) Robot-hand distance evolution using the CP-based formulation.

Fig. 10. Comparison between the dense-based and CP-based methods based on task execution of Fig. 9. The left column shows the TCP evolution of the two robots: Robot 1 tracks the desired trajectory, whereas Robot 2 maintains a fixed reference point. The right column reports the corresponding robot-distance evolution obtained using the dense-based and CP-based formulations, respectively.



(a) RGB image from the camera.

(b) Complete depth map of the scene.

(c) Depth map after robot filtering.

(d) Robot-only depth removed from the complete depth map.

Fig. 11. Sequence of filtering results in two scenarios. In the first four images, only the robot is present and the robot contribution is removed from the depth map. In the last four images, a human hand touches the robot and is not removed by the filter, remaining visible in the reconstructed scene.

The same type of experiment was repeated using the CP-RDF model and the corresponding sequence is reported in Fig. 9. Also in this case, the left robot follows a prescribed trajectory while the second robot is commanded to hold a position. Initially, when the two robots are sufficiently far apart, collision avoidance is inactive for both robots, see Fig. 9(a). As shown in Fig. 9(b), when the moving robot enters the activation region, two repulsive gradient become active and starts steering the robots away from their nominal motions. In Fig. 9(c), the robot is still in the avoidance phase and, in order to maintain a safe clearance, the repulsive action continues to deviate the first robot from the trajectory and the second robot from its commanded position. As shown in Fig. 9(d), the repulsive term is deactivated and the moving robot can finish its nominal motion, reaching the final configuration.

The evolution of the dual-arm task is reported in Fig. 10. Since both manipulators contribute to the collision-avoidance action, the deviations of their TCPs from the respective reference trajectories, shown in Figs. 8(a) and 8(b), are substantially smaller than those observed in the previous scenario. The trajectories obtained with the dense and CP-based formulations are nevertheless not identical, as the two link representations yield slightly different distance and gradient estimates and, consequently, different corrective motions. Despite these differences, the minimum inter-robot distance profiles reported in Figs. 10(c) and 10(d) remain above the prescribed safety threshold, confirming that both formulations preserve the required clearance throughout the motion.

7.3. Point cloud filtering from camera data

Beyond collision avoidance, we also use the link-wise RDFs to remove measurements associated with the robot body from the acquired point cloud. This configuration-dependent filtering produces a cleaner and less cluttered representation of the scene, making external objects and people easier to isolate and process. Fig. 11 presents the results of the proposed RDF-based filtering pipeline on the TIAGo platform, equipped with a head-mounted Xtion RGB-D camera. Starting from the point cloud acquired by the sensor, the objective is to remove the robot contribution from the observed scene while preserving the surrounding environment. To this end, the filtering stage exploits the RDF models of the robot links that may appear within the camera field of view, namely the links from the fourth arm link to the wrist force/torque sensor, the gripper body and the two fingers. Each link is modeled using a Bernstein basis of order $N = 32$ and a CP decomposition of rank $R = 48$. Since this configuration provides a sufficiently accurate representation of the link geometry, the CP formulation is adopted for filtering due to its lower computational cost in online evaluation. Robot points are then identified as those whose signed distance to the implicit surface is below a prescribed threshold $d_{\text{theshold}} < 20$ mm, introduced to account for sensor noise, point cloud discretization and residual approximation errors.

Although the filtering is performed on the point cloud, the results are presented as reconstructed depth images, since this representation provides a clearer visual assessment of the filtering quality. For each scenario, the figure shows the RGB image, the complete depth map, the filtered depth map without the robot and the depth map of the removed robot points.

The experiment was carried out on point clouds containing approximately 11,000 points per frame. The filtering task consists of identifying the points belonging to the robot using the implicit RDF representation of its geometry. The average processing time of the filtering pipeline, including point cloud acquisition from the ROS topic and generation of the filtered outputs shown in Fig. 11, is 27.3 ms per frame. This remains below the acquisition period of the RGB-D stream running at 30 Hz, thus enabling real-time operation.

Two representative scenarios are reported in Fig. 11. In the first row, only the robot is visible in the scene and the method removes the robot contribution from the sensed data, producing both a scene-only reconstruction and a robot-only reconstruction. In the second row, a person places a hand around the wrist area of TIAGo and the hand is preserved in the filtered scene rather than being removed together with the robot. These results indicate that the proposed method can remove the robot structure while preserving a consistent point cloud representation of external objects, even when they are in contact with the robot surface.

8. Conclusions

This paper presented a low-rank tensor formulation of Robot Distance Fields for fast and differentiable signed-distance evaluation on GPUs. Starting from Bernstein polynomial SDFs, CP and TT formats were used to define low-rank link-wise RDF representations while preserving analytical computation of both the field and its gradient. The same formulation supports both post-hoc compression of dense Bernstein coefficient tensors and direct fitting of low-rank factors from sampled SDF values; the experiments focused on the latter route.

This latter aspect is particularly important in robotics, since many collision-aware methods rely not only on distance values but also on gradient information to define repulsive motions, trajectory corrections and optimization directions. In this sense, the proposed compressed RDFs are not merely compact geometric surrogates: they remain directly usable within gradient-based collision avoidance pipelines.

The results showed that tensor decomposition is beneficial for RDF evaluation because it reduces the cost of the dense Bernstein representation and enables higher-order implicit models under realistic GPU memory constraints. Among the two investigated decompositions, CP

provided the strongest overall contribution in the tested settings: when a moderate CP rank is sufficient to approximate the geometry, its separable structure leads to the best runtime and throughput. TT, instead, provided a more limited computational benefit. Its main advantage was observed in terms of storage, peak memory and feasible batch size, especially for objects whose accurate CP approximation requires a much larger rank than the TT ranks needed to obtain comparable fidelity. Thus, TT should be regarded as a memory-scalability option rather than as a generally faster alternative to CP.

Overall, low-rank RDFs provide an effective compromise between accuracy, runtime, memory usage and differentiability. In this trade-off, CP appears to be the most effective decomposition for runtime-oriented GPU evaluation, whereas TT becomes useful in more specific regimes where stronger compression is required to keep high-fidelity models within the available memory budget.

A complementary direction for future work concerns the structured modeling of the candidate query set. Adaptive sampling strategies could additionally reduce the number of required queries by concentrating them in regions with a higher estimated collision likelihood. Combining adaptive query selection with SDF robot representations could therefore further improve the overall efficiency of collision-aware robotic applications.

More broadly, the proposed low-rank RDF formulation could also serve as a compact geometric component within future embodied-AI systems and robot foundation models. In such settings, fast batched distance and gradient evaluations could support collision filtering, safety assessment, and local refinement of actions or trajectories generated by learned policies. The reduced memory footprint may additionally facilitate the concurrent execution of large learning-based models and geometric safety modules on shared computational hardware. Investigating how robot-specific RDF representations can be integrated with generalist and potentially multi-embodiment policies represents a promising direction for future research.

CRediT authorship contribution statement

Andrea Govoni: Writing – original draft, Validation, Software, Conceptualization. **Sylvain Calinon:** Writing – review & editing, Supervision. **Gianluca Palli:** Writing – review & editing, Supervision, Project administration.

Code availability

The implementation supporting this work is available on Zenodo at: https://doi.org/10.5281/zenodo.20041439 and maintained at https://github.com/GovoUnibo/CP-TT-RDF.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andrea Govoni reports article publishing charges and equipment, drugs, or supplies were provided by Horizon Europe. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work received support in terms of equipment and team activities from the Horizon Europe project *IntelliMan – AI-Powered Manipulation System for Advanced Robotic Service, Manufacturing and Prosthetics*, under Grant Agreement No. 101070136.

Appendix 9

Proof (*CP Equivalency*)**.** Starting from the CP approximation $w_{\mathrm{CP}} = \sum_{r=1}^{R} \lambda_r (a_r \otimes b_r \otimes c_r)$ and $\Psi(p) = \Phi \otimes \Phi_y \otimes \Phi_z$ and using Kronecker product and vectorization.

$$(u \otimes v \otimes w)^{\top}(x \otimes y \otimes z) = (u^{\top}x)(v^{\top}y)(w^{\top}z), \tag{37}$$

we have

$$\begin{aligned} f(p) &= \Psi(p)^{\top} w \;\approx\; \Psi(p)^{\top} w_{\mathrm{CP}} \\ &= (\Phi \otimes \Phi_y \otimes \Phi_z)^{\top} \sum_{r=1}^{R} \lambda_r \, (a_r \otimes b_r \otimes c_r) \\ &= \sum_{r=1}^{R} \lambda_r \left(\Phi^{\top} a_r\right) \left(\Phi_y^{\top} b_r\right) \left(\Phi_z^{\top} c_r\right), \end{aligned} \tag{38}$$

which proves the claimed separable evaluation form. □

Proof (*TT Equivalency*)**.** Using Fubini theorem for sums, we have

$$\begin{aligned} \sum_{i,j} \sum_{\alpha} a_{i,\alpha} \, b_{j,\alpha} &= \sum_{\alpha} \sum_{i,j} a_{i,\alpha} \, b_{j,\alpha} \\ &= \sum_{\alpha} \Big(\sum_{i} a_{i,\alpha} \Big) \Big(\sum_{j} b_{j,\alpha} \Big). \end{aligned} \tag{39}$$

Let the 1-D bases be $\{\phi_i\}_{i=0}^{N-1}$ and define the 3-D separable basis vector

$$\Psi(p) = \sum_{i_1=0}^{N-1} \sum_{i_2=0}^{N-1} \sum_{i_3=0}^{N-1} \phi_{i_1}(t_x) \, \phi_{i_2}(t_y) \, \phi_{i_3}(t_z) \, e_{i_1,i_2,i_3} \;\in\; \mathbb{R}^{N^3},$$

where e_{i_1,i_2,i_3} is the canonical basis associated to w. Then, substituting w_{TT},

$$\begin{aligned} &f(p) = \Psi(p)^{\top} w_{\mathrm{TT}} = \\ &\sum_{i_1,i_2,i_3} \phi_{i_1}(t_x) \phi_{i_2}(t_y) \phi_{i_3}(t_z) \sum_{\alpha_1,\alpha_2} G^{(1)}_{1,i_1,\alpha_1} \, G^{(2)}_{\alpha_1,i_2,\alpha_2} \, G^{(3)}_{\alpha_2,i_3,1} \\ &= \sum_{\alpha_1,\alpha_2} \left(\phi(t_x)^{\top} G^{(1)}_{1,:,\alpha_1}\right) \left(\phi(t_y)^{\top} G^{(2)}_{\alpha_1,:,\alpha_2}\right) \left(\phi(t_z)^{\top} G^{(3)}_{\alpha_2,:,1}\right), \end{aligned} \tag{40}$$

with $\phi(t) := [\phi_0(t), \ldots, \phi_{N-1}(t)]^{\top}$. □

Appendix 10. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2026.105717.

Data availability

The source code and supplementary video have been provided as Supplementary Material during submission. A GitHub repository is also referenced in the manuscript. Code: https://github.com/GovoUnibo/CP-TT-RDF; archive: https://zenodo.org/records/20041439.

References

[1] M. Jones, J. Baerentzen, M. Sramek, 3D distance fields: a survey of techniques and applications, IEEE Trans. Vis. Comput. Graphics 12 (2006) 581–599, http://dx.doi.org/10.1109/TVCG.2006.56.

[2] S.T. Bukhari, D. Lawson, A.H. Qureshi, Differentiable composite neural signed distance fields for robot navigation in dynamic indoor environments, in: 2025 IEEE International Conference on Robotics and Automation, ICRA, 2025, pp. 16169–16175, http://dx.doi.org/10.1109/ICRA55743.2025.11128172.

[3] M. Koptev, N. Figueroa, A. Billard, Neural joint space implicit signed distance functions for reactive robot manipulator control, IEEE Robot. Autom. Lett. 8 (2023) 480–487, http://dx.doi.org/10.1109/LRA.2022.3227860.

[4] A. Govoni, M. Cavuoto, Y. Li, S. Calinon, G. Palli, Real-time collision avoidance with robot distance fields in a task-priority framework, Robot. Auton. Syst. 200 (2026) 105394, http://dx.doi.org/10.1016/j.robot.2026.105394, URL: https://www.sciencedirect.com/science/article/pii/S0921889026000679.

[5] Y. Yang, J. Liu, H. Zhou, A.R. Kwabena, Y. Zhong, Y. Xie, Robotsdf: Implicit morphology modeling for the robotic arm, Sensors 24 (2024) http://dx.doi.org/10.3390/s24165248, URL: https://www.mdpi.com/1424-8220/24/16/5248.

[6] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, M. Mukadam, Isdf: Real-time neural signed distance fields for robot perception, in: Robotics: Science and Systems, 2022.

[7] Y. Li, Y. Zhang, A. Razmjoo, S. Calinon, Representing robot geometry as distance fields: Applications to whole-body manipulation, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, 2024, pp. 15351–15357, http://dx.doi.org/10.1109/ICRA57147.2024.10611674.

[8] X. Zhu, Y. Xin, S. Li, H. Liu, C. Xia, B. Liang, Efficient collision detection framework for enhancing collision-free robot motion, in: Proceedings of the 2025 IEEE International Conference on Robotics and Automation, ICRA, 2025, pp. 16162–16168, http://dx.doi.org/10.1109/ICRA55743.2025.11128733, URL: https://arxiv.org/abs/2409.14955.

[9] Y. Li, X. Chi, A. Razmjoo, S. Calinon, Configuration space distance fields for manipulation planning, 2024, URL: https://arxiv.org/abs/2406.01137, arXiv:2406.01137.

[10] M. Jacquet, M. Harms, K. Alexis, Neural nmpc through signed distance field encoding for collision avoidance, Int. J. Robot. Res. (2025) http://dx.doi.org/10.1177/02783649251401223, onlineFirst.

[11] A. Marić, Y. Li, S. Calinon, Online learning of continuous signed distance fields using piecewise polynomials, IEEE Robot. Autom. Lett. 9 (2024) 6020–6026, http://dx.doi.org/10.1109/LRA.2024.3397085.

[12] L. Wu, C. Le Gentil, T. Vidal-Calleja, Vdb-gpdf: Online gaussian process distance field with vdb structure, IEEE Robot. Autom. Lett. 10 (2025) 374–381, http://dx.doi.org/10.1109/LRA.2024.3505814.

[13] U. Ali, L. Wu, A. Müller, F. Sukkar, T. Kaupp, T. Vidal-Calleja, Interactive distance field mapping and planning to enable human-robot collaboration, IEEE Robot. Autom. Lett. 9 (2024) 10850–10857, http://dx.doi.org/10.1109/LRA.2024.3482128.

[14] L. Brudermuller, T. Lembono, S. Shetty, S. Calinon, Trajectory prediction with compressed 3d environment representation using tensor train decomposition, in: 2021 20th International Conference on Advanced Robotics, ICAR, 2021, pp. 633–639, http://dx.doi.org/10.1109/ICAR53236.2021.9659407.

[15] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (2009) 455–500, http://dx.doi.org/10.1137/07070111X.

[16] V. Vasilopoulos, S. Garg, J. Huh, B. Lee, V. Isler, Hio-sdf: Hierarchical incremental online signed distance fields, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, 2024, pp. 17537–17543, http://dx.doi.org/10.1109/ICRA57147.2024.10610367.

[17] A. Millane, H. Oleynikova, E. Wirbel, R. Steiner, V. Ramasamy, D. Tingdahl, R. Siegwart, Nvblox: Gpu-accelerated incremental signed distance field mapping, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, 2024, pp. 2698–2705, http://dx.doi.org/10.1109/ICRA57147.2024.10611532.

[18] Y. Bai, Z. Miao, X. Wang, Y. Liu, H. Wang, Y. Wang, Vdbblox: Accurate and efficient distance fields for path planning and mesh reconstruction, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2023, pp. 7187–7194, http://dx.doi.org/10.1109/IROS55552.2023.10342123.

[19] C. Quintero-Peña, W. Thomason, Z. Kingston, A. Kyrillidis, L.E. Kavraki, Stochastic implicit neural signed distance functions for safe motion planning under sensing uncertainty, in: 2024 IEEE International Conference on Robotics and Automation, ICRA, 2024, pp. 2360–2367, http://dx.doi.org/10.1109/ICRA57147.2024.10610773.

[20] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, G. Chalvatzaki, Regularized deep signed distance fields for reactive motion generation, 2022, arXiv preprint arXiv:2203.04739, URL: https://arxiv.org/abs/2203.04739.

[21] N. Marticorena, T. Fischer, J. Haviland, N. Suenderhauf, Rmmi: Reactive mobile manipulation using an implicit neural map, in: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2025, pp. 1797–1804, http://dx.doi.org/10.1109/IROS60139.2025.11245922.

[22] V. Vasilopoulos, S. Garg, P. Piacenza, J. Huh, V. Isler, Ramp: Hierarchical reactive motion planning for manipulation tasks using implicit signed distance functions, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2023, pp. 10551–10558, http://dx.doi.org/10.1109/IROS55552.2023.10342397.

[23] T. Zhang, J. Wang, C. Xu, A. Gao, F. Gao, Continuous implicit sdf based any-shape robot trajectory optimization, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2023, pp. 282–289, http://dx.doi.org/10.1109/IROS55552.2023.10342104.

[24] B. Sundaralingam, S.K.S. Hari, A. Fishman, C. Garrett, K.V. Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, D. Fox, Curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023, URL: https://arxiv.org/abs/2310.17274, arXiv:2310.17274.

[25] K. Long, H. Parwana, G. Fainekos, B. Hoxha, H. Okamoto, N. Atanasov, Neural configuration distance function for continuum robot control, 2025, URL: https://arxiv.org/abs/2409.13865, arXiv:2409.13865.

[26] S. Xu, G. Li, J. Liu, Obstacle avoidance for manipulator with arbitrary arm shape using signed distance function, in: 2018 IEEE International Conference on Robotics and Biomimetics, ROBIO, 2018, pp. 343–348, http://dx.doi.org/10.1109/ROBIO.2018.8665083.

[27] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Proceedings. 1985 IEEE International Conference on Robotics and Automation, 1985, pp. 500–505, http://dx.doi.org/10.1109/ROBOT.1985.1087247.

[28] A. Cichocki, N. Lee, I. Oseledets, A.H. Phan, Q. Zhao, D.P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Found. Trends Mach. Learn. 9 (2015) 249–429, http://dx.doi.org/10.1561/2200000059.

[29] A. Chen, Z. Xu, A. Geiger, J. Yu, H. Su, Tensorf: Tensorial radiance fields, in: S. Avidan, G. Brostow, M. Cissé, G.M. Farinella, T. Hassner (Eds.), Computer Vision – ECCV 2022, Springer Nature Switzerland, Cham, 2022, pp. 333–350.

[30] A. Pinkus, Weierstrass and approximation theory, Online lecture notes, URL: https://www.math.pku.edu.cn/teachers/litj/notes/numer_anal/WeierstrassApprox_Pinkus.pdf. (Accessed 26 January 2026).

[31] A. Pinkus, Density in approximation theory, Surv. Approx. Theory 1 (2005) 1–45, See Thm. 6.2 (Dieudonné) and Stone–Weierstrass discussion.

[32] A. Fellhauer, Approximation of smooth functions using Bernstein polynomials in multiple variables, 2016, http://dx.doi.org/10.48550/arXiv.1609.01940, arXiv preprint arXiv:1609.01940.

[33] J. Kossaifi, Y. Panagakis, A. Anandkumar, M. Pantic, Tensorly: Tensor learning in python, J. Mach. Learn. Res. 20 (2019) 1–6, URL: http://jmlr.org/papers/v20/18-277.html.

[34] S. Williams, A. Waterman, D. Patterson, Roofline: An insightful visual performance model for multicore architectures, Commun. ACM 52 (2009) 65–76, http://dx.doi.org/10.1145/1498765.1498785.

[35] Y. Wang, C. Yang, S. Farrell, Y. Zhang, T. Kurth, S. Williams, Time-based roofline for deep learning performance analysis, in: 2020 IEEE/ACM Fourth Workshop on Deep Learning on Supercomputers, DLS, 2020, pp. 10–19, http://dx.doi.org/10.1109/DLS51937.2020.00007.

[36] M. Ghane, J. Larkin, L. Shi, S. Chandrasekaran, M.S. Cheung, Power and energy-efficiency roofline model for gpus, 2018, URL: https://arxiv.org/abs/1809.09206, arXiv:1809.09206.