Online Learning of Continuous Signed Distance Fields Using Piecewise Polynomials

Ante Marić, Yiming Li, and Sylvain Calinon

***Abstract*—Reasoning about distance is indispensable for establishing or avoiding contact in manipulation tasks. To this end, we present an online approach for learning implicit representations of signed distance using piecewise polynomial basis functions. Starting from an arbitrary prior shape, our method incrementally constructs a continuous and smooth distance representation from incoming surface points, with analytical access to gradient information. The underlying model does not store training data for prediction, and its performance can be balanced through interpretable hyperparameters such as polynomial degree and number of segments. We assess the accuracy of the incrementally learned model on a set of household objects and compare it to neural network and Gaussian process counterparts. The utility of intermediate results and analytical gradients is further demonstrated in a physical experiment. For code and video, see https://sites.google.com/view/pp-sdf/.**

***Index Terms*—Signed Distance Fields; Incremental Learning; Representation Learning; Machine Learning for Robot Control**

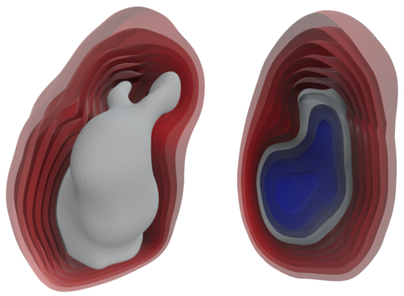
Fig. 1: Mesh and distance level sets of the *Stanford Bunny* reconstructed from piecewise polynomial basis functions. The model is learned incrementally from 1283 randomly sampled surface points and corresponding normal vectors.

I. Introduction

SCENE representation is a naturally emerging topic in robotics as a basis for physical interaction. In recent years, implicit modeling methods have been used as compact representations of environment properties such as distance, occupancy, and color. Signed distance functions (SDFs) model distances to closest occupied points by assigning zero values to surfaces, negative values to surface interiors, and positive values elsewhere. Previously used in environment mapping and collision avoidance settings [1], they have recently seen use in robotic manipulation as the field moves towards exploring contact-rich behaviors [2]. In such scenarios, distance representations can be exploited to quickly and robustly retrieve gradients for a variety of tasks. Furthermore, modeling the full range of distances, as opposed to only the zero level set, can be beneficial for reasoning about making or breaking contact, deformation, or penetration, with extensions to active agents such as users or robots. Commonly used implicit SDF models in robotics rely on neural architectures or Gaussian process models, while alternative formulations remain less explored. Earlier computer graphics work encodes SDFs using basis functions, with recent extensions showing promising results using piecewise polynomial representations [3]. In robotics, basis functions have been used to encode movements as superpositions of primitives [4]. Basis function representations of distance can consequently be seen as a step toward combining movement with shape primitives of robot environments.

When operating in previously unseen environments, incrementally building a distance model from incoming data enables the integration of feedback for more robust and adaptive behavior. We formulate an online method for learning signed distance fields represented as piecewise polynomial basis functions. Our method uses a simple incremental least squares approach and regularization scheme to approximate distance fields from incoming surface points. The resulting representation is C^1 continuous, with analytical access to gradients for downstream use. To achieve fast update time on modest hardware, performance of the piecewise polynomial representation can be balanced through interpretable hyperparameters such as polynomial degree and number of segments. It is also easy to incorporate priors through basis function superposition weights. We evaluate the accuracy of our method on a set of diversely shaped household objects, showing comparable results to Gaussian process and neural network baselines, while relying on a lower number of parameters, and without the need to store training data for prediction. The usability of intermediate results and analytical gradients is further demonstrated in a physical experiment where we use an evolving representation learned from noisy and partial point cloud data to survey and grasp an object of interest.

Manuscript received: December 21, 2023; Revised: March 17, 2024; Accepted: April 15, 2024.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the State Secretariat for Education, Research and Innovation in Switzerland and the European Commission's Horizon Europe Program through the INTELLIMAN project (https://intelliman-project.eu/, HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENSO project (http://sestosenso.eu/, HORIZON-CL4-Digital-Emerging Grant 101070310).

The authors are with the Idiap Research Institute, CH-1920 Martigny, Switzerland and also with the EPFL, 1015 Lausanne, Switzerland. ante.maric@idiap.ch; yiming.li@idiap.ch; sylvain.calinon@idiap.ch

Digital Object Identifier (DOI): see top of this page.

II. Implicit environment representations

In robotics, widely adopted scene representations use mapping approaches that rely on discretized occupancy grids and account for uncertainty [5]. Subsequent methods explicitly store distance information for real-time usage in dynamic settings by introducing distance fields [1], [6]. Following advancements in computer vision, recent work has been exploring implicit representations as scalable and compact alternatives for describing scenes without storing data in dense grids. Implicit representations of distance, volume, and color information have found application in robotics, with initial uses in navigation [7] and mapping [8]. Recent work uses implicit representations as visual encodings for grasping [9], whole-body manipulation [2], human-robot interaction [10], planning [11], and control [12].

A. Implicit signed distance functions

Implicit signed distance functions model geometry through continuous functions, thus decoupling memory from spatial resolution. Seminal work utilizes neural networks for shape representation, showing higher performance than point cloud, mesh, or grid-based counterparts [13]. Additional efforts have been put into investigating regularization and modeling methods that allow learning such representations in online scenarios using raw point cloud data [14], [15]. Recent methods introduce neural architectures that jointly represent SDF and color to track and reconstruct unknown objects [16]. Similarly, Neural radiance fields (NeRFs) [17] jointly encode volume density and color. They have garnered much attention as environment representations, with recent extensions targeting real-time rendering [18] and dynamic scenes [19]. However, NeRFs do not offer direct access to distance or derivative information, which can be beneficial for interpreting task execution. Furthermore, neural representations often require large amounts of data and do not translate readily to lower data regimes found in modalities such as tactile or proximity sensing.

B. Gaussian process implicit surfaces

Probabilistic models like Gaussian process implicit surfaces (GPIS) have been used to represent distances with account for uncertainty [20]. To extend the approach for mapping purposes, scaling issues of the Gaussian process have been addressed through the use of clustering and hierarchical models [21]. Subsequent combinations with implicit regularization methods enable accurate modeling of unsigned distance [22]. These methods were later combined to give a unified mapping, odometry, and planning framework [23].

C. Basis function representations

Basis functions can describe complex representations as weighted superpositions of simple signals. In robotics, they are well-known as the underlying representation used to encode movement primitives [24], [25]. A detailed review can be found in [4]. Their role in computer graphics extends to higher-dimensional input space to represent shapes using SDFs [26]. Learning such representations from point clouds and normals can be achieved simply by solving a linear system of equations using a least squares approach or iterative optimization procedures [27]. A recent example utilizes piecewise polynomials to approximate high-fidelity SDFs using an adaptive grid [3]. The resulting SDF models offer analytical access to distance and gradients, and a desired order of continuity can be ensured by constraining the superposition weights. The following section describes an online formulation of piecewise polynomial SDF based on incremental learning, with the aim of adaptively guiding movement in manipulation tasks.

III. Piecewise polynomial SDF

A. Bernstein polynomial basis functions

The value of a univariate function $f(x)$ at input x can be represented as a weighted sum of K basis functions with

$$f(x) = \sum_{k=0}^{K} \phi_k(x)\, w_k = \boldsymbol{\phi}(x)\, \boldsymbol{w}, \tag{1}$$

where $\boldsymbol{\phi}$ can come from any family of basis functions. For our SDF representation, we use Bernstein polynomial bases, which give smooth function approximations on bounded intervals. For degree K, they can be computed as

$$\phi_k(x) = \frac{K!}{k!(K-k)!}\,(1-x)^{K-k}\, x^k, \tag{2}$$

$\forall k \in \{0, \dots, K\}$. Instead of considering a global encoding which might require the use of high-order polynomials, we split the problem into a set of local fitting problems that can consider lower-order polynomials. We retain C^1 continuity by introducing constraints of the form

$$w_K^a = w_0^b \tag{3}$$

$$w_1^b = -w_{K-1}^a + 2w_0^b, \tag{4}$$

where a and b are concatenated polynomials of degree K, and w_k^a is used to denote the k-th weight of polynomial a. Polynomial bases and their derivatives can then be expressed in matrix form as

$$\boldsymbol{\phi}(x) = \boldsymbol{T}(x)\boldsymbol{B}\boldsymbol{C}, \tag{5}$$

$$\frac{\partial \boldsymbol{\phi}(x)}{\partial x} = \frac{\partial \boldsymbol{T}(x)}{\partial x}\boldsymbol{B}\boldsymbol{C}, \tag{6}$$

with $\boldsymbol{T}(x) = \begin{bmatrix} 1 & x & x^2 & \cdots & x^K \end{bmatrix}$ being a polynomial feature map of input x, $\boldsymbol{B}$ the corresponding Bernstein coefficient matrix, and $\boldsymbol{C}$ a constraint matrix of the form

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots \\ 0 & 1 & \cdots & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ 0 & 0 & \cdots & 1 & 0 & \cdots \\ 0 & 0 & \cdots & 0 & 1 & \cdots \\ 0 & 0 & \cdots & 0 & 1 & \cdots \\ 0 & 0 & \cdots & -1 & 2 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{7}$$

enforcing (3) and (4).

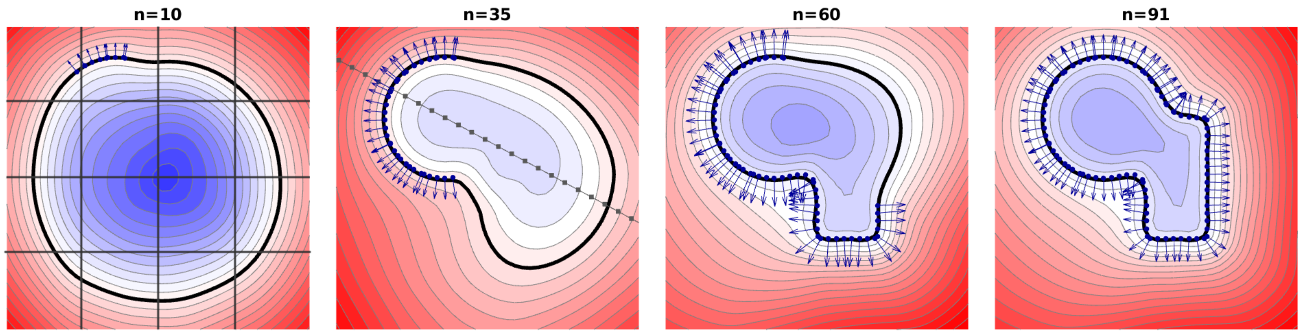

Fig. 2: Incremental model updates used to model a 2D shape on a 4×4 grid, starting from a circular prior. Sampled points and normals are shown in dark blue, and the reconstructed zero-level contour in black. The reconstructed SDF is visualized as a color map. The normal ray and regularization points of a single sample are displayed in the second image.

Successive Kronecker products can be used to extend the described representation to any number of input and output dimensions. For clarity and visualization purposes, we will continue the method description for a two-dimensional case. Namely, an extension to two-dimensional input space (Cartesian coordinates) and one-dimensional output (signed distance) can be calculated as

$$\boldsymbol{\Psi}(x,y) = \boldsymbol{\phi}(x) \otimes \boldsymbol{\phi}(y), \tag{8}$$

with partial derivatives and gradient computed analytically as

$$\frac{\partial \boldsymbol{\Psi}(x,y)}{\partial x} = \frac{\partial \boldsymbol{\phi}(x)}{\partial x} \otimes \boldsymbol{\phi}(y), \tag{9}$$

$$\frac{\partial \boldsymbol{\Psi}(x,y)}{\partial y} = \boldsymbol{\phi}(x) \otimes \frac{\partial \boldsymbol{\phi}(y)}{\partial y}, \tag{10}$$

$$\boldsymbol{\nabla\Psi}(x,y) = \frac{\partial \boldsymbol{\Psi}(x,y)}{\partial x} \otimes \frac{\partial \boldsymbol{\Psi}(x,y)}{\partial y}, \tag{11}$$

This can then be used to compute the distance and gradient values of the SDF at coordinate (x,y) with

$$f(x,y) = \boldsymbol{\Psi}(x,y)\,\boldsymbol{w}, \tag{12}$$

$$\boldsymbol{\nabla} f(x,y) = \boldsymbol{\nabla\Psi}(x,y)\,\boldsymbol{w}. \tag{13}$$

The same procedure can be applied to calculate higher-order derivatives for regularization purposes

$$\frac{\partial \boldsymbol{\nabla} f(x,y)}{\partial x} = \frac{\partial \boldsymbol{\nabla\Psi}(x,y)}{\partial x}\boldsymbol{w}. \tag{14}$$

The above representation extends analogously to accommodate three-dimensional Cartesian coordinates as input by applying an additional Kronecker product in (8).

B. Computation of weights

To approximate the SDF using polynomial basis functions, any method capable of solving a system of linear equations of the form $\boldsymbol{A}\boldsymbol{w} = \boldsymbol{s}$ can be employed. The simplest case can utilize a batch least squares estimate of the form

$$\boldsymbol{w} = (\boldsymbol{A}^\top \boldsymbol{A})^{-1}\boldsymbol{A}^\top \boldsymbol{s}, \tag{15}$$

or ridge regression as the regularized variant [28].

We use quadratic error terms in order to evaluate the fitting of distance and normal data for N incoming samples

$$c_d(\boldsymbol{x}_n) = \Big(\boldsymbol{\Psi}(\boldsymbol{x}_n)\boldsymbol{w}\Big)^2, \tag{16}$$

$$c_g(\boldsymbol{x}_n) = ||\boldsymbol{\nabla\Psi}(\boldsymbol{x}_n)\boldsymbol{w} - \boldsymbol{g}_n||^2, \tag{17}$$

with $\boldsymbol{x}_n = (x_n, y_n)$ denoting the n-th input sample, and $\boldsymbol{g}_n$ the corresponding sampled normal. An additional *tension* term is used to constrain the curvature of the resulting distance field by minimizing the sum of second-order partial derivatives on R control points

$$c_t(\boldsymbol{x}_r) = ||\boldsymbol{H}_f(\boldsymbol{x}_r)||_F^{\ 2}, \tag{18}$$

$\forall r \in \{1, \ldots, R\}$, where $c_t(\boldsymbol{x}_r)$ represents the squared Frobenius norm of the corresponding Hessian matrix

$$\boldsymbol{H}_f(x,y) = \left[\frac{\partial \boldsymbol{\nabla} f(x,y)}{\partial x} \;\; \frac{\partial \boldsymbol{\nabla} f(x,y)}{\partial y}\right]. \tag{19}$$

The model weights can then be learned by minimizing a combined cost

$$c = \lambda_d^2 \sum_{n=1}^{N} c_d(\boldsymbol{x}_n) + \lambda_g^2 \sum_{n=1}^{N} c_g(\boldsymbol{x}_n) + \lambda_t^2 \sum_{r=1}^{R} c_t(\boldsymbol{x}_r), \tag{20}$$

with cost tuning coefficients λ_d, λ_g, and λ_t. We construct our input vectors as concatenations of flattened distance, gradient, and tension features, denoted in italics

$$\boldsymbol{\Psi}^* = [\mathit{\Psi}(x_1) \cdots \mathit{\Psi}(x_N)]^\top, \tag{21}$$

$$\boldsymbol{\nabla\Psi}^* = [\nabla\mathit{\Psi}(x_1) \cdots \nabla\mathit{\Psi}(x_N)]^\top, \tag{22}$$

$$\boldsymbol{H}_{\boldsymbol{\Psi}}^* = [H_{\mathit{\Psi}}(x_1) \cdots H_{\mathit{\Psi}}(x_R)]^\top. \tag{23}$$

Finally, we can minimize (20) by calculating (15) using

$$\boldsymbol{A} = \begin{bmatrix} \lambda_d \boldsymbol{\Psi}^* \\ \lambda_g \boldsymbol{\nabla\Psi}^* \\ \lambda_t \boldsymbol{H}_{\boldsymbol{\Psi}}^* \end{bmatrix}, \quad \boldsymbol{s} = \begin{bmatrix} \boldsymbol{0}_{\boldsymbol{\Psi}} \\ \lambda_g \boldsymbol{g} \\ \boldsymbol{0}_{\boldsymbol{H}} \end{bmatrix}, \tag{24}$$

where $\boldsymbol{g}$ is a vector of sampled normal components, and $\boldsymbol{0}_{\boldsymbol{\Psi}}$ and $\boldsymbol{0}_{\boldsymbol{H}}$ are zero vectors with lengths compatible with $\boldsymbol{\Psi}^*$ and $\boldsymbol{H}_{\boldsymbol{\Psi}}^*$, respectively.

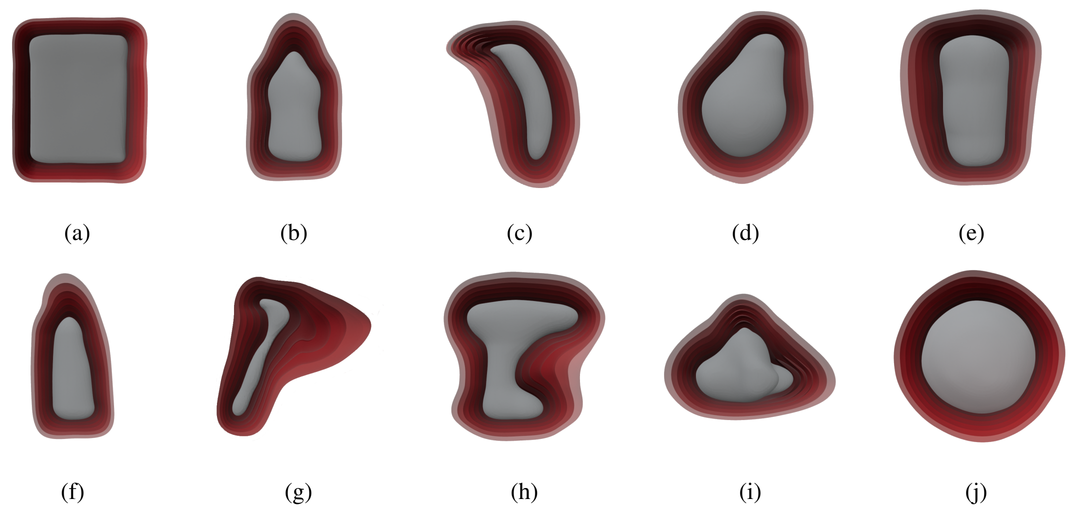
Fig. 3: Mesh (gray) and level set (red) reconstruction results for objects from the YCB [29] test set: (a) `003_cracker_box`, (b) `006_mustard_bottle`, (c) `011_banana`, (d) `016_pear`, (e) `019_pitcher_base`, (f) `021_bleach_cleanser`, (g) `048_hammer`, (h) `035_power_drill`, (i) `063-a_marbles`, (j) `053_mini_soccer_ball`. Models were learned from 800 non-uniformly sampled points and normals using 6 segments per input dimension.

C. Incremental formulation

Based on similar approaches used in the context of control [30], we update concatenated local models with an incremental variant of the least squares algorithm. Namely, we exploit the Sherman-Morrison-Woodbury relations [31] which connect subsequent inverses of a matrix after small-rank perturbations. This allows us to gradually refine an initial estimate by providing samples one by one or in batches.

After initializing the weight precision matrix $\boldsymbol{P} = \text{cov}(\boldsymbol{w})^{-1}$ to $\boldsymbol{P}_0$, it can be incrementally updated as

$$\boldsymbol{P}_{\text{new}} = \boldsymbol{P} - \underbrace{\boldsymbol{P}\boldsymbol{A}_n^\top \left(\sigma^2 \boldsymbol{I} + \boldsymbol{A}_n \boldsymbol{P} \boldsymbol{A}_n^\top\right)^{-1}}_{\boldsymbol{K}} \boldsymbol{A}_n \boldsymbol{P}, \tag{25}$$

where σ^2 is the measurement noise variance, $\boldsymbol{A}_n$ the input matrix, and $\boldsymbol{I}$ an identity matrix of compatible dimensions. Starting from prior weights $\boldsymbol{w} = \boldsymbol{w}_0$, updates can then be calculated by using the Kalman gain $\boldsymbol{K}$

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + \boldsymbol{K}\Big(\boldsymbol{s}_n - \boldsymbol{A}_n \boldsymbol{w}\Big). \tag{26}$$

The above iterative computation has no requirement of storing the training points and enables us to impose priors on our model through $\boldsymbol{P}_0$ and $\boldsymbol{w}_0$. We impose a spherical prior on the weights and initialize the precision matrix as a scaled identity matrix for recursive ridge regression.

In order for our model to accurately approximate a distance function, the tension term needs to be enforced throughout the input space. We achieve this in an online setting by uniformly sampling a number of control points on the normal rays of incoming surface samples, as displayed in Figure 2. The full computation steps are summarized in Algorithm 1. We apply the same approach for three-dimensional inputs, with example reconstructions shown in Figures 1 and 3.

Algorithm 1 Incremental computation of weights.

```
$\boldsymbol{P} = \boldsymbol{P}_0$ // Initialize precision matrix
$\boldsymbol{w} = \boldsymbol{w}_0$ // Initialize weights
for $n \leftarrow 1$ to $N$ do
    $\boldsymbol{A}_n = \boldsymbol{A}(\boldsymbol{x}_n)$ // Construct input matrix
    $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{A}_n^\top \left(\sigma^2 \boldsymbol{I} + \boldsymbol{A}_n \boldsymbol{P} \boldsymbol{A}_n^\top\right)^{-1}$ // Compute gain
    $\boldsymbol{P} \leftarrow \boldsymbol{P} - \boldsymbol{K}\boldsymbol{A}_n\boldsymbol{P}$ // Update precision matrix
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{K}\Big(\boldsymbol{s}_n - \boldsymbol{A}_n \boldsymbol{w}\Big)$ // Update weights
end
```

IV. Evaluation

A. Reconstruction accuracy

We first evaluate the reconstruction accuracy of our approach by comparing it to two baseline methods used in online settings:

1) A neural network model based on *iSDF* [15], using 4 hidden layers of 256 neurons, with corresponding positional embeddings and regularized loss.
2) A GP model based on *LogGPIS* [22] using the Matérn 3/2 kernel.

Our comparison model utilizes cubic Bernstein polynomials with 6 segments per input dimension. All models are implemented in *PyTorch* and run on an *NVIDIA GeForce MX550* GPU.

Accuracy evaluations are done on varying volumes of real point cloud data from the YCB dataset [29]. The utilized test set consists of 10 household objects of diverse shapes, depicted in Figure 3. All three methods are trained on the same depth-only point cloud and normal data, with ground truth SDFs reconstructed from high-definition meshes with 512k polygons. Note that *LogGPIS* models unsigned Euclidean distance fields, and is therefore evaluated against absolute values of the ground truth.

To evaluate the reconstruction accuracy of our method we use error metrics similar to [15]. Reconstructed distances are evaluated using the mean absolute error (MAE)

$$\mathrm{MAE}(\boldsymbol{x}) = \left|\hat{s}(\boldsymbol{x}) - s(\boldsymbol{x})\right|, \tag{27}$$

with $\hat{s}(\boldsymbol{x})$ denoting the estimated signed distance at point $\boldsymbol{x}$, and $s(\boldsymbol{x})$ the ground truth value. Figure 4 shows the resulting MAE comparisons. Qualitative reconstruction results of our method for all objects in the test set are shown in Figure 3.

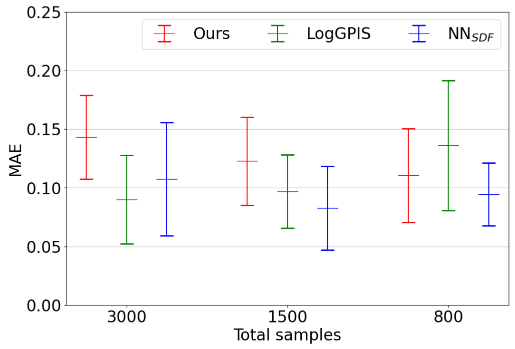

Fig. 4: Distance reconstruction accuracy compared on varying amounts of training data.

Additional comparisons are made with respect to reconstructed distance gradients by calculating the gradient cosine distance (GCD)

$$GCD(\boldsymbol{x}) = 1 - \frac{\nabla_{\boldsymbol{x}}\hat{s}(\boldsymbol{x})\nabla_{\boldsymbol{x}}s(\boldsymbol{x})}{\|\nabla_{\boldsymbol{x}}\hat{s}(\boldsymbol{x})\|\|\nabla_{\boldsymbol{x}}s(\boldsymbol{x})\|}, \tag{28}$$

with $\nabla_{\boldsymbol{x}}\hat{s}(\boldsymbol{x})$ denoting the estimated distance gradient, and $\nabla_{\boldsymbol{x}}s(\boldsymbol{x})$ the corresponding ground truth. Figure 5 displays the GCD comparisons.

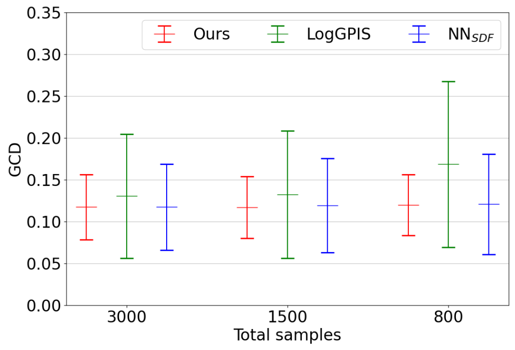

Fig. 5: Accuracy of reconstructed gradient fields with varying amounts of training data.

Many manipulation tasks inherently involve contact, thus requiring higher fidelity distance approximations closer to object surfaces. On the other hand, intermediary tasks such as reaching or collision avoidance often involve larger distances. Table I displays reconstruction accuracy near and far from object surfaces for different amounts of training samples.

$|s| < 0.05$

Samples	Ours	GP	NN
3000	0.0329 ± 0.0201	0.0620 ± 0.0409	0.0326 ± 0.0152
1500	0.0332 ± 0.0202	0.0549 ± 0.0405	0.0326 ± 0.0162
800	0.0338 ± 0.0208	0.0501 ± 0.0369	0.0330 ± 0.0194

$|s| > 0.05$

Samples	Ours	GP	NN
3000	0.1481 ± 0.0337	0.0916 ± 0.0392	0.1174 ± 0.0561
1500	0.1261 ± 0.0551	0.1044 ± 0.0325	0.0818 ± 0.0339
800	0.1079 ± 0.0522	0.1252 ± 0.0461	0.0798 ± 0.0299

TABLE I: Comparison of the mean absolute error (MAE) near and far from object surfaces for varying amounts of training samples.

B. Computational requirements

Our representation relies on a sparse number of basis function weights as parameters, and does not store training data for prediction. The total number of parameters is further reduced by imposing constraints through concatenation, as described in Section III. For three-dimensional input and S concatenated cubic polynomials, the number of utilized parameters corresponds to $N_w(S) = 8(S+1)^3$. The models utilized in our experiments therefore store and update only $N_w(6) = 2744$ parameters. Comparatively, the NN_{SDF} baseline has 200193 learnable parameters, and *LogGPIS* requires storing the training samples to inform mean and covariance predictions.

The computation time of updates to our model increases quadratically based on the number of weights. For a set input dimension and cubic polynomials, the time complexity of a single incremental update is $\mathcal{O}(B^3 + {N_w}^2 B + N_w B^2)$, with B denoting the sample batch size. Figure 6 shows mean update times for different numbers of segments and varying batch sizes. Total training times of the evaluated models are compared in Table II.

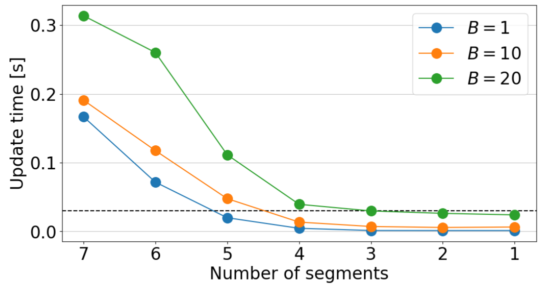

Fig. 6: Update time for cubic polynomials with varying numbers of segments and different batch sizes. The desired real-time cutoff of 30 ms is denoted by a horizontal line.

Training time [s]

Samples	LogGPIS	NN$_{SDF}$	Ours (S=6)	Ours (S=4)
3000	28.58	8.722	32.36	3.725
1500	6.886	7.002	15.97	1.835
800	1.657	4.720	8.535	0.9919

TABLE II: Total training time comparisons for varying numbers of training samples.

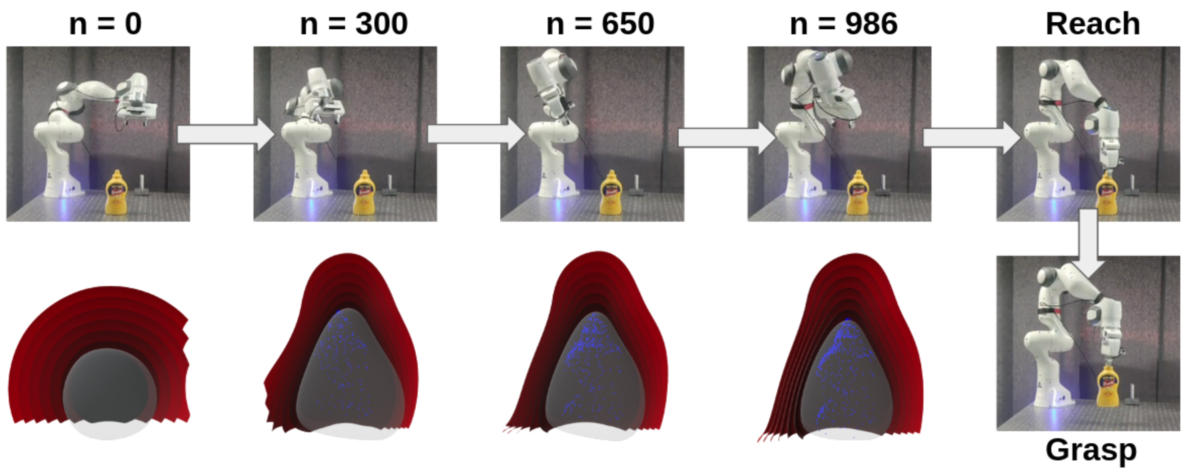

Fig. 7: A visualization of our physical experiment and results, showing the evolution of a piecewise polynomial SDF model starting from a spherical prior. As n samples are collected, the updated model is queried online to control the manipulator by following a trajectory tangential to the SDF level sets while keeping normal orientation. The reconstructed distance field is then used to generate and execute a collision-free grasp.

For queries and reconstruction of distance and gradient fields, our method scales linearly with the number of weights, as we calculate only a weighted sum of basis functions. The time complexity of querying our model for a batch of B points is $\mathcal{O}(N_w B)$, with mean time of 0.134 ms for a single query of distance and gradient. Table III shows the mean time of full SDF reconstructions on a dense 128^3 grid.

SDF reconstruction time [s]

Samples	LogGPIS	NN$_{SDF}$	Ours (S=6)	Ours (S=4)
3000	47.03	0.9828	5.7436	5.4086
1500	19.06	0.9865	5.9605	5.5912
800	7.636	0.9842	5.8488	5.3287

TABLE III: SDF reconstruction time with respect to varying numbers of training samples.

C. Physical experiment

To test the viability of our method in a real-world setting, we perform a physical experiment in an online learning scenario using a Franka Emika 7-DoF manipulator. Starting from a spherical prior, we leverage the continuity and smoothness of our representation by querying gradients to execute a trajectory tangential to the SDF level sets. As the robot moves around the object of interest, partial-view depth data is collected by an in-hand Intel RealSense D415 sensor. The collected surface samples are used to incrementally update the SDF, directly shaping the trajectory of the manipulator. Once the object shape is sufficiently explored, the reconstructed distance field is used to generate a valid grasp and execute it without colliding with the object. To enable fast performance on CPU, we utilize cubic polynomials with 4 segments per input dimension and point-by-point updates. The described learning and control framework is run with a mean timestep of 33.7 ms on a 3.6 GHz Intel Core i9-9900K CPU. Figure 7 shows our experiment setup and resulting SDF used to perform a grasp.

V. Discussion

Figures 4 and 5 demonstrate that our incrementally learned representation can achieve similar distance and gradient reconstruction accuracy as baseline methods on the provided test set. Table I further shows that these results are consistent across data volumes, and that valid field reconstructions are maintained at different distances. As shown in Section IV-B, our model relies on a low number of parameters to achieve these results, and the incremental learning scheme does not require storing the training data after model updates. Performance and accuracy of the proposed model can be balanced by adjusting the number of segments or degree of utilized polynomial basis functions. Inference time scales linearly with the number of weights and batch size, allowing for fast queries that correspond to calculating a weighted sum of basis functions. This is further reflected by full SDF reconstruction times in Table III. At the expense of increased memory usage, faster large-scale inference (e.g., for visualization purposes) can be achieved by precomputing the basis function values. Additional efforts in performance optimization might consider leveraging local weight updates and further parallelization.

Qualitative reconstructions in Figure 3 display visually accurate distance and mesh reconstructions across the test set, with a noted over-smoothing effect due to polynomial approximation on a low-resolution grid. The over-smoothing effect becomes less pronounced as spatial resolution is increased by further segmenting the input space. However, as our implementation relies on a uniform grid for segmenting the input space, this can rapidly increase the number of weights and result in higher computation times reflected by Figure 6 and training time comparisons in Table II. Scaling the model for higher accuracy and larger or more complex environments might therefore require combining the online paradigm with adaptive grids or hierarchical models such as octrees, which will be a topic of our future work.

The physical experiment showcases the interplay of incremental updates and fast queries of analytical gradients to adapt the trajectory of a manipulator as surface samples are collected. Furthermore, it validates the use of an incrementally updated prior for motion planning with noisy and partial point cloud data. For fast performance on CPU, the utilized model was reduced to 4 segments per input dimension, showing that the resulting distance and gradient fields are still accurate enough for a surveying and grasping task. This demonstrates the viability of using a basis function representation of the SDF in a simple manipulation scenario, and opens the way toward methods further leveraging the properties of Bernstein polynomials and related families of basis functions for learning and reconstruction. Future work will focus on additional quantitative evaluation to investigate how accuracy and robustness of the proposed method scale to more complex manipulation tasks, higher levels of measurement noise, and more dynamic scenes. Lastly, we intend to explore contact-rich scenarios where incremental updates with tactile and proximity data might be of particular interest, and investigate the use of more informative priors conditioned on modalities such as RGB or tactile data.

VI. Conclusion

This paper presented an online formulation of signed distance fields using piecewise polynomial basis functions. Starting from an arbitrary prior shape, our method incrementally builds a smooth distance representation from incoming surface points. It offers analytical access to gradients and ensures a desired order of continuity through constraints on the basis function weights. Furthermore, performance of the underlying model can be balanced through interpretable hyperparameters. Our results show that a low number of parameters can be used to achieve similar reconstruction accuracy to Gaussian process and neural network baselines on a test set of household objects. Finally, we demonstrated the use of the online basis function representation in a physical surveying and grasping task with noisy partial observations, and discussed possible extensions for further scalability.

References

[1] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning," in *Proc. IEEE/RSJ IROS*, pp. 1366–1373, 2017.

[2] Y. Li, Y. Zhang, A. Razmjoo, and S. Calinon, "Representing robot geometry as distance fields: Applications to whole-body manipulation," in *Proc. IEEE ICRA*, 2024.

[3] E. Pujol and A. Chica, "Adaptive approximation of signed distance fields through piecewise continuous interpolation," *Computers & Graphics*, vol. 114, pp. 337–346, 2023.

[4] S. Calinon, "Mixture Models for the Analysis, Edition, and Synthesis of Continuous Time Series," in *Mixture Models and Applications* (N. Bouguila and W. Fan, eds.), pp. 39–57, Springer, Cham, 2019.

[5] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[6] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: Fast incremental euclidean distance fields for online motion planning of aerial robots," *Proc. IEEE/RSJ IROS*, pp. 4423–4430, 2019.

[7] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-Only Robot Navigation in a Neural Radiance World," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 4606–4613, 2022.

[8] E. Sucar, S. Liu, J. Ortiz, and A. Davison, "iMAP: Implicit Mapping and Positioning in Real-Time," *Proc. IEEE/CVF ICCV*, pp. 6209–6218, 2021.

[9] M. Breyer, J. J. Chung, L. Ott, S. Roland, and N. Juan, "Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter," in *Proc. CoRL*, pp. 1602–1611, 2020.

[10] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, "Regularized Deep Signed Distance Fields for Reactive Motion Generation," in *Proc. IEEE/RSJ IROS*, pp. 6673–6680, 2022.

[11] D. Driess, J.-S. Ha, M. Toussaint, and R. Tedrake, "Learning Models as Functionals of Signed-Distance Fields for Manipulation Planning," in *Proc. CoRL*, 2021.

[12] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba, "3D Neural Scene Representations for Visuomotor Control," in *Proc. CoRL*, pp. 112–123, 2021.

[13] J. J. Park, P. R. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," *Proc. IEEE CVPR*, pp. 165–174, 2019.

[14] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit Geometric Regularization for Learning Shapes," in *Proc. ICML*, vol. 119, pp. 3789–3799, 2020.

[15] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "iSDF: Real-Time Neural Signed Distance Fields for Robot Perception," in *Proc. RSS*, 2022.

[16] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Muller, A. Evans, D. Fox, J. Kautz, and S. Birchfield, "BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects," *Proc. IEEE CVPR*, 2023.

[17] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Comm. ACM*, vol. 65, p. 99–106, Dec 2021.

[18] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, July 2023.

[19] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural Radiance Fields for Dynamic Scenes," in *Proc. IEEE CVPR*, 2020.

[20] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *Proc. IEEE ICRA*, pp. 2845–2850, 2011.

[21] B. Lee, C. Zhang, Z. Huang, and D. D. Lee, "Online Continuous Mapping using Gaussian Process Implicit Surfaces," *Proc. IEEE ICRA*, pp. 6884–6890, 2019.

[22] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja, "Faithful Euclidean Distance Field From Log-Gaussian Process Implicit Surfaces," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2461–2468, 2021.

[23] L. Wu, K. M. B. Lee, C. Le Gentil, and T. Vidal-Calleja, "Log-GPIS-MOP: A Unified Representation for Mapping, Odometry, and Planning," *IEEE Transactions on Robotics (T-RO)*, vol. 39, no. 5, pp. 4078–4094, 2023.

[24] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors," *Neural Computation*, vol. 25, pp. 328–373, 02 2013.

[25] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic Movement Primitives," in *Proc. NIPS*, pp. 2616–2624, 2013.

[26] B. Jüttler and A. Felis, "Least-Squares Fitting of Algebraic Spline Surfaces," *Advances in Computational Mathematics*, vol. 17, pp. 135–152, Jul 2002.

[27] G. Taubin, "Smooth Signed Distance Surface Reconstruction and Applications," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 38–45, 2012.

[28] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[29] B. Calli *et al.*, "Yale-CMU-Berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[30] J.-A. Ting, S. Vijayakumar, and S. Schaal, "Locally weighted regression for control," in *Encyclopedia of Machine Learning*, pp. 613–624, Springer, 2010.

[31] W. W. Hager, "Updating the Inverse of a Matrix," *SIAM Rev.*, vol. 31, pp. 221–239, 1989.