# Implicit Distance Functions: Learning and Applications in Control

Mikhail Koptev[1], Nadia Figueroa[2] and Aude Billard[1]

***Abstract*— This paper describes a novel approach to learn an implicit, differentiable distance function for arbitrary configurations of a robotic manipulator used for reactive control. By exploiting GPU processing, we efficiently query the learned collision representation and obtain an implicit distance between the robot and the environment. The differentiable nature of the learned function allows for calculating valid gradients wrt. any robot configuration, providing a repulsive vector field in joint-space that can be injected in various control methods to improve collision avoidance. We present preliminary results on solving collision avoidance for a 7DoF robot with a reactive inverse kinematics solution, as well as improving performance of a sampling-based model-predictive controller.**

## I. Introduction

In previous works [1], [2], we demonstrated that self-collisions of a redundant system can be represented as a static boundary in high-dimensional joint-space of a robot. We have shown that if this boundary is approximated as a continuously differentiable function (of class $\mathcal{C}^1$) the gradients wrt. any input robot configuration essentially represent a repulsive vector field directly in the joint-space of the robot. This repulsion can be used as a constraint in a control optimization routine or as a heuristic in sampling-based methods. In this paper, we extend this idea and present a novel approach towards constructing an implicit distance function for distance evaluation between a robot in arbitrary configurations and any point in the three-dimensional workspace of the robot. The learned neural model allows for efficient and highly-parallelizable batched distance and gradient queries via GPU. We demonstrate the applicability of the learned distance function in two reactive control schemes: i) as a collision-avoidance constraint in a QP-based inverse kinematics (IK) controller, extending [1], [2], and ii) as a heuristic to improve performance of a sampling-based joint-space model-predictive control (MPC) scheme [3], [4].

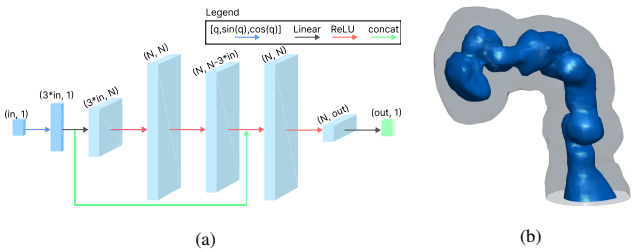
## II. Problem Formulation

Let's consider a robotic manipulator with $m$ degrees of freedom and $K$ links, with the state described by joint angles $q \in \mathbb{R}^m$, i.e., all revolute joints of the robot have joint-limits. Let's assume $\mathcal{B} \subset \mathbb{R}^3$ to be the set of points on the links of the robot. We can define minimal distances between the robot and arbitrary points $y \in \mathbb{R}^3$ in the workspace as

$$d^k_{min}(q, y) = \min_{x \in \mathcal{B}_k} ||f(q, x) - y||, \quad k = 1..K \qquad (1)$$

where $\mathcal{B}_k \subset \mathcal{B}$ is set of points of the $k$-th link of the robot, and $f : \mathbb{R}^m \times \mathcal{B} \to \mathbb{R}^3$ is forward kinematics function.

[1]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[2]University of Pennsylvania, Philadelphia, Pennsylvania, USA




(a) (b)

Fig. 1: (a) - NN architecture used to learn $\Gamma(q, y)$; (b) - Implicit distance isosurfaces $\Gamma(q, y) = 1$ (solid) and $\Gamma(q, y) = 10$ (transparent).

The *first goal* of this paper is to learn a regression function $\Gamma(q, y)$ to approximate $d^k_{min}(q, y)$, representing a distance field of the robot's workspace depending on configuration of the robot. Additionally, $\frac{\partial \Gamma(q,y)}{\partial q}$ is a vector field defined in the joint-space of the robot, providing information on the direction to (and away from) the collision. The *second goal* is to apply the learned function $\Gamma(q, y)$ and its gradient to enhance reactive control methods, by i) formulating a collision-avoidance constraint in a QP-based IK controller [2], and ii) introducing sampling heuristics into MPPI [4].

## III. Learning Implicit Distance function

Let's consider the expanded state-space $\mathbb{R}^m \times \mathbb{R}^3$, consisting of the robot state $q \in \mathbb{R}^m$ and a euclidean point $y \in \mathbb{R}^3$ in the workspace of the robot. For each expanded state there exists a unique minimal distance between the robot in configuration $q$ and the point at position $y$. Hence, in this space a static distance field function exists. We propose to build a neural representation $\Gamma(q, y) : \mathbb{R}^m \times \mathbb{R}^3 \to \mathbb{R}$, by learning the minimal distances between the robot and arbitrary points in the workspace. Knowing the exact robot geometry, at training time we collect a dataset of exact values $d^k_{min}(q, y)$ (1) for various $q$ and $y$. Each sample contains the concatenated robot state $q$, workspace point $y$, and target vector $d_{min} = [d^1_{min}...d^K_{min}]$ consisting of minimal distances between links of the robot and point $y$. We sample randomly similar to [2]. Final dataset is balanced and contains three million entries, where 50% configurations are collided (i.e., $d^k_{min}(q, y) < 3\text{cm}, \ \forall k$), and 50% are configurations with minimal distance exceeding 3cm. Additionally, the collided half of the dataset is balanced to include collisions for links in equal proportions.

We, therefore, seek to learn $\Gamma(q, x)$ to be able to evaluate distances between the robot and points on the moving obstacles and use $\frac{\partial \Gamma(q,y)}{\partial q}$ to represent the repulsive vector field. Ideally, we want $\Gamma(q, y) = d^k_{min}(q, y)$ for any state $q$ within joint-limits. To learn $\Gamma(q, y)$ we use MLP with five hidden layers. We choose ReLU as a nonlinear activation function for the sake of faster forward and backward passes.

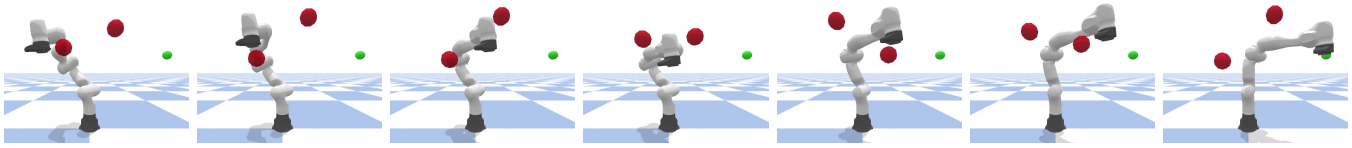


Fig. 2: Goal (green point) reaching task via QP Inverse Kinematics (2), while avoiding moving obstacles (red spheres oscillate vertically)

$\Gamma(q, y)$ implicitly learns the robot's forward kinematics (FK); thus, we find it useful to build a feature vector as a concatenation of joint angles, and their corresponding sine and cosine values: $q_{in} = [q, cos(q), sin(q)]$. Additionally, we introduce a skip-connection between the input and the fourth layer of the network. This serves two purposes: first, we compensate for vanishing gradients, and second, we reintroduce the input trigonometrical features to deeper layers for better FK approximation. A schematic representation of the neural network is shown in Fig. 1a. There, $in = 10$ (7 for robot DoF and 3 for workspace point position), $out = 9$ (representing $d^k_{min}$ for $k = 1..9$ links), and $N = 256$. Total number of weights in this perceptron is $N_w = 199,915$. The learned function $\Gamma(q, y)$ predicts minimum distances with RMSE of 0.77cm and standard deviation 0.86cm. For points closer than 5cm, classification accuracy of $\text{sign}(\Gamma(q, y) - 1)$ is 0.94 (averaged between all links). The isosurfaces for different values of $\Gamma(q, y)$ are visualized in Fig. 1b.

## IV. Towards Reactive Control with Learned IDF

### A. Reactive Collision-Avoidance IK

Similar to [2] we use the learned function $\Gamma(q, y)$ to formulate a constraint in a QP IK solver:

$$\min_{\Delta q, \delta} \delta^T Q \delta + \Delta q^T R \Delta q$$
$$\text{s.t.} \quad \begin{cases} f(q) + \dfrac{\partial f(q)}{\partial q} \Delta q = x + \delta \\ q_i^- < q_i + \Delta q_i < q_i^+, \quad i = 1..m \\ -\dfrac{\partial \Gamma_k(q, y_s)}{\partial q}^T \Delta q \leq \Gamma_k(q, y_s) - r_s, \quad k = 1..K. \end{cases} \tag{2}$$

In (2), $R$ is a damping term, and $Q$ is a weight matrix for slacks of Cartesian tasks $x$ (positions and orientations of end-effector). $q_i^-$ and $q_i^+$ are joint-limits, and $\Gamma_k(q, y)$ are $k$-th component of implicit distance function output vector. Finally, $y_s$ are centers of spherical obstacles $s = 1..S$ with radii $r_s$. For simplicity of demonstration, we consider $S$ separate spherical obstacles. The last constraint guarantees collision avoidance, repelling the robot from collision when $\Gamma_k(q, y_s)$ has low values. The method works with $200Hz$ frequency (Intel i7 4.2GHz, Nvidia 1060). Snapshots of a reaching example with moving obstacles are shown in Fig. 2.

### B. Improving Sampling-based MPPI

Consider a discrete-time system. At time $t$ the robot is controlled by joint-space acceleration command $u_t$, sampled from a policy $\pi_t = \Pi_{h=1}^H \pi_{t,h}$, where $H$ is a look-ahead horizon, and policies $\pi_{t,h}$ are simple Gaussians defined by means $\mu_{t,1}, ..., \mu_{t,H}$ and covariances $\Sigma_{t,1}, ..., \Sigma_{t,H}$. At every iteration the sampling-based MPC algorithm, proposed in [4], samples a batch $\{u_{n,h}\}_{n=1..N}^{h=1..H}$ of $N$ control sequences of length $H$ from current distribution $\pi_t$. After that, the roll-out states $\{x_{n,h}\}_{n=1..N}^{h=1..H}$ are computed using the approximate dynamics function and corresponding costs $\{c_{n,h}\}_{n=1..N}^{h=1..H}$ are calculated. These costs are a weighted sum of goal-reaching, joint-limit avoidance, contingency stopping, and self and environmental collision avoidance costs. Gaussian policies are then updated using a sample-based gradient:

$$\mu_{t,h} = (1 - \alpha_\mu)\mu_{t-1,h} + \alpha_\mu \frac{\sum_{i=1}^N w_i u_{i,h}}{\sum_{i=1}^N w_i}, \tag{3}$$

where $\alpha_\mu$ is filtering coefficient and $w_i(c_{i,h})$ are exponentially-weighted task-specific costs for sampled control sequences. For full derivation, update rule for covariances, and equations for weights $w_i$ please refer to [3], [4].

We use values of the learned function $\{\Gamma_k(q, y_s)\}_{k=1..K}^{s=1..S}$ to find the closest link $\tilde{k}$ and obstacle $\tilde{s}$, and use corresponding gradients to reproject sampled controls $u_{t,h}$ from (3):

$$u^*_{t,h} = (1 - f_c)u_{t,h} + f_c(u^\tau_{t,h} + u^{rep}_{t,h}). \tag{4}$$

In (4), $f_c \in [0, 1]$ is a correction coefficient, $u^\tau_{t,h}$ is joint-acceleration forcing the robot to move in tangential hyper-plane of collision boundary, and $u^{rep}_{t,h}$ is a repulsion pushing the robot away from collisions. We obtain them by projecting the originally sampled vector $u_{t,h}$ onto $\frac{\partial \Gamma_{\tilde{k}}(q, y_{\tilde{s}})}{\partial q}$.

Coefficient $f_c(q, \dot{q})$ depends on value $\Gamma_{\tilde{k}}(q, y_{\tilde{s}})$ and cosine similarity between $\dot{q}$ and $\frac{\partial \Gamma_{\tilde{k}}(q, y_{\tilde{s}})}{\partial q}$. This way, we modify accelerations that move the robot towards collision to instead explore the space around the obstacles.

TABLE I: Sampling-based MPC performance comparison. Values averaged between 100 reaching experiments similar to Fig.2. Hardware used: Intel i7 4.2GHz, Nvidia 2080Ti

| Method | Success rate | Iterations | Time, s | Freq., Hz |
|---|---|---|---|---|
| Original [4] | 0.96 | 663 | 5.43 | 121 |
| Our modification | 1.00 | 296 | 2.78 | 106 |

An overview of the method performance is provided in Table I. Overall, gradient heuristics in sampling significantly speed-up the planning in presence of obstacles with a slight decrease in control frequency.

## V. Discussion & Future work

Our ongoing work focuses on expanding the method to obstacles other than spheres. The nature of our approach allows us to efficiently process arbitrary point clouds, outputting minimal distances and repulsive fields in joint space for each link-point pair. These vector fields work in synergy with sampling-based methods improving the exploration. Still, for optimized performance, it is required to develop an algorithm to obtain a sparse point-cloud representation of obstacles and complex static environments and combine all repulsive vector fields into an efficient heuristic.

## REFERENCES


[1] S. S. M. Salehian, N. Figueroa, and A. Billard, "A unified framework for coordinated multi-arm motion planning," *The International Journal of Robotics Research*, vol. 37, no. 10, pp. 1205–1232, 2018. [Online]. Available: https://doi.org/10.1177/0278364918765952

[2] M. Koptev, N. Figueroa, and A. Billard, "Real-time self-collision avoidance in joint space for humanoid robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1240–1247, 2021. [Online]. Available: https://doi.org/10.1109/LRA.2021.3057024

[3] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017. [Online]. Available: https://doi.org/10.2514/1.G001921

[4] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots, "STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *5th Annual Conference on Robot Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=ceOmpjMhlyS