PairwiseNet: Pairwise Collision Distance Learning for High-dof Robot Systems

Anonymous Author(s)
Affiliation
Address
email

Abstract: Motion planning of robot manipulation systems operating in complex environments remains a challenging problem, requiring evaluation of the collision distance and its derivative. Due to its computational complexity, recent studies have attempted to utilize data-driven approaches to learn the collision distance. However, their performance degrades significantly for complicated high-dof systems like multi-arm robots. Also, the model must be retrained every time the environment undergoes even slight changes. In this paper, we propose PairwiseNet, a model that estimates the minimum distance between two geometric shapes and overcomes many of the limitations of current models. By dividing the problem of global collision distance learning into smaller pairwise sub-problems, PairwiseNet can be used to efficiently the global collision distance. PairwiseNet can be used without further modifications or training for any system comprised of the same shape elements (such as in the training dataset). Experiments with multi-arm manipulation systems of various dof show that our model achieves significant performance improvements with respect to several performance metrics, especially the false positive rate with the collision-free guaranteed threshold. Results further demonstrate that our single trained PairwiseNet model is applicable to all multi-arm systems used in the evaluation.

Keywords: Robot Collision, Collision Distance, Machine Learning

1 Introduction

Motion planning algorithms such as RRT [1, 2, 3] and its many variants [4, 5, 6, 7], all require the *collision distance* - the minimum distance between the robot and its nearest obstacle (including other links for self-collision avoidance). Of these, some even require the derivatives of the collision distance. As is well-known, calculating this distance involves finding the minimum distance between each robot link and the obstacles, which can be computationally intensive, especially for high-dof robots with complex geometries.

To alleviate the computational burden, one possible solution is to train a collision distance function using data. By collecting sufficient data consisting of robot configurations and their corresponding collision distances, machine learning models such as kernel perceptrons [8], support vector machines (SVM) [9], and neural networks [10, 11, 12, 13, 14], can be employed to learn the collision distance function. This learned function can then be used to quickly determine if a given configuration is collision-free. While these data-driven approaches have demonstrated satisfactory results for low-dof robot systems, more often than not they perform poorly for higher-dof robots. The challenge lies in the fact that the collision distance function for higher-dof robots is complex and highly non-convex.

Another challenge faced by existing data-driven methods is their sensitivity to small environmental changes. For example, the addition of new obstacles or a change in the robot's base position can

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

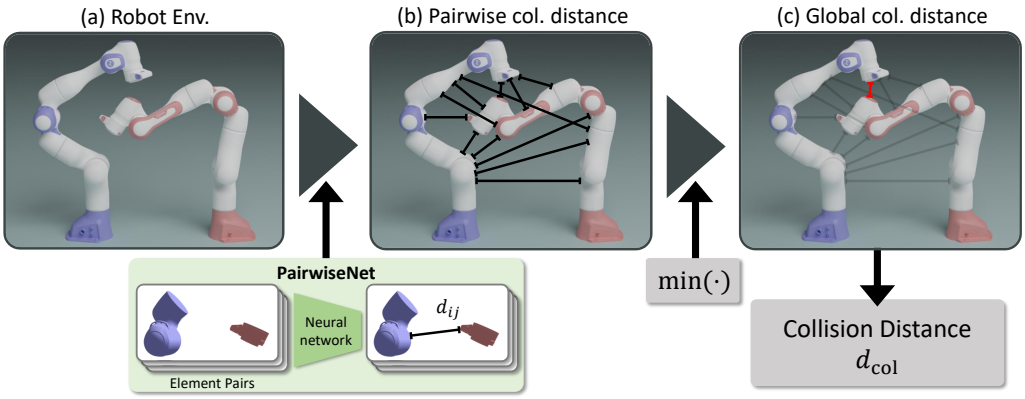


Figure 1: Illustration of the global collision distance estimation through PairwiseNet. (a) Robot environment at a given joint configuration. (b) Pairwise collision distances for all element pairs are determined through PairwiseNet. (c) The smallest of these distances becomes the global collision distance.

lead to a completely different collision distance function; for many such existing methods, the entire training procedure must be repeated, from data collection to model training. (One possible exception is [8], which proposes an efficient model update strategy for dynamic environment updates, but their method is limited to low-dof robot systems and still requires an additional training procedure.)

We present PairwiseNet, a collision distance estimation method that provides a promising alternative to existing data-driven approaches used for predicting the *global* collision distance. Instead of directly estimating the global collision distance, PairwiseNet focuses on estimating the *pairwise* collision distance - the minimum distance between two elements in the robot system. The PairwiseNet model takes as input the point cloud data of two geometric shapes and their relative transformation and outputs the minimum distance between these two shapes. To estimate the global collision distance, PairwiseNet first predicts the minimum distances for every possible pair of elements in the system. It then selects the minimum of these pairwise distances as the estimate for the global collision distance (see Figure 1). The efficient parallel batch computation of the neural network enables the rapid prediction of minimum distances between element pairs.

Compared to the complex and highly non-convex function of the global collision distance, the minimum distance function between a pair of elements is simpler and easier to train. By breaking down the challenging task of learning the global collision distance into smaller sub-problems of pairwise collision distance learning, our PairwiseNet achieves significant performance improvements for high-dof robot systems.

Another advantage of PairwiseNet is its applicability to any system composed of known shape elements (sufficiently trained shape elements for estimating pairwise collision distance). The trained PairwiseNet model can be used without the need for additional training or modifications in such systems. For example, consider a scenario in which a sufficiently large dataset containing pairwise collision distances between the links of a Panda robot is available. In this case, the PairwiseNet model trained using this dataset can be applied to any system consisting of multiple Panda robots, regardless of the number of robots or their respective positions. This is possible because the collision distance estimation for such systems can be broken down into pairwise collision distance estimations for each element pair, and these pairwise distances are already known by the trained PairwiseNet model. Therefore, as long as the system is exclusively comprised of shape elements that have been learned during training, the trained PairwiseNet model can be used for any system that meets this requirement. Even in cases where the system undergoes changes, such as changing the robot's base position or adding another robot arm, if the geometric shape elements of the system remain unchanged, the trained PairwiseNet model can be directly applied to the changed system without any modifications.

Our approach has been evaluated in high-dof multi-arm robot manipulation systems, ranging from two-arm (14-dof) to four-arm (28-dof) systems, as well as a single-arm robot with obstacles. The results demonstrate that our approach outperforms existing learning-based methods in terms of collision distance regression error, collision checking accuracy, and especially the False Positive Rate with the collision-free guaranteed threshold (Safe-FPR). Moreover, our approach exhibits superior performance even when using a single trained PairwiseNet model for all multi-arm systems.

2 Related Works

Several machine learning-based methods for collision distance estimation have been proposed due to their computationally efficient inference procedures for collision distances and derivatives. [6] used SVM classifiers to identify whether each part pair of a humanoid robot was in a safe or dangerous self-collision status given a specific joint configuration. The minimum distances of only the dangerous part pairs were estimated using a capsule-based BV algorithm, simplifying the calculation of collision distances and derivatives. [9] also employed an SVM classifier for a 14-dof dual-arm robot manipulation system. The SVM classifier inputs a vector consisting of the positions of all joints in the system and outputs a collision label of either <1 for a collision or >1 for a collision-free state. In [13], SVM and neural network models were trained to predict the collision label of a humanoid manipulation system at a given joint configuration. A separate collision classifier model was trained for each sub-part pair, such as the left arm and right leg, resulting in a total of 10 sub-models used for collision label predictions.

Similar to [9], [10] utilized joint positions as inputs for their multi-layer perceptron neural network model. Meanwhile, [11] employed a positional encoding vector of the joint configuration as input for their neural network model. [12] trains a neural network model to estimate the collision distance using an extended configuration containing both joint and workspace configurations as input, with the model outputting the collision distance of the system. DiffCo [8] is a collision classifier model based on kernel perceptron that generates both the collision score and its derivative. It utilizes the Forward Kinematics (FK) kernel, which calculates distances as the Euclidean distance between points on the robot body. DiffCo also utilizes an efficient active learning strategy that adjusts the trained collision score function for dynamic updates in the environment. Recently, GraphDistNet [14] was proposed as a Graph Neural Network (GNN) model for collision distance estimation. The model inputs the information on geometric shapes, which are represented as graphs for both the manipulator links and obstacles. GraphDistNet then utilizes the geometric relationship between the two graphs to predict the collision distance.

The effectiveness of these existing methods have been experimentally demonstrated only for low-dof robot systems; their performance degrades substantially for high-dof systems operating in complex environments. In particular, most learning-based collision distance estimation methods establish a collision-free guaranteed threshold to ensure that no collisions occur during actual manipulation. However, existing methods often suffer from a high false positive rate, resulting in overly cautious collision detection when utilizing the collision-free guaranteed threshold. In comparison, our method demonstrates effective collision distance estimation performance even in high-dof robot systems and maintains low false positive rates when using the collision-free guaranteed threshold. Additionally, it can be easily adapted to other environments, such as changes in the robot's base location, without requiring retraining or updating procedures.

3 Learning Pairwise Collision Distance

3.1 Problem Formulation

We assume the availability of a simulator environment of the target system, which includes the robot kinematics and geometric shapes of links and obstacles. We aim to determine the optimal model parameter ψ for the pairwise collision distance estimation model f_ψ, which can predict the collision distance between any pair of geometric shapes. The model takes the point cloud data of

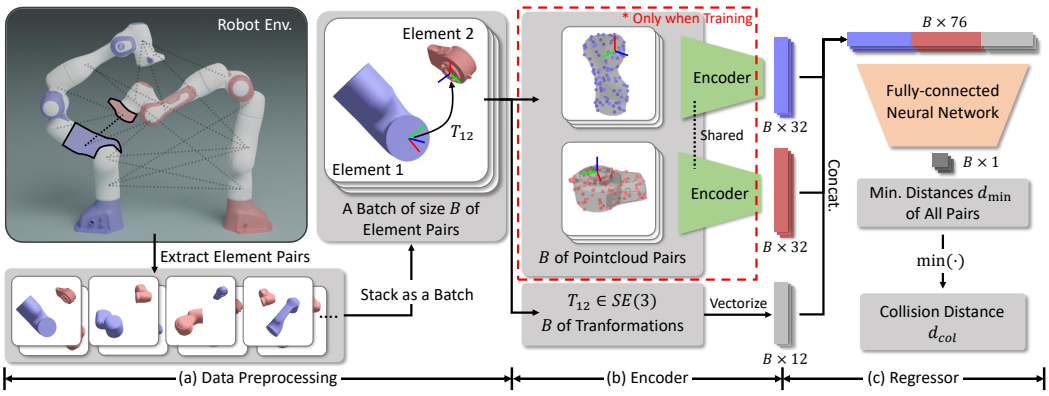


Figure 2: An illustration of estimating the global collision distance via PairwiseNet, our pairwise collision distance learning method.

two geometric shapes $\mathcal{P}_i$, $\mathcal{P}_j$ (expressed in each corresponding object coordinates) and the relative transformation $T_{ij} \in SE(3)$ as input, and outputs the estimated pairwise collision distance $\hat{d}_{ij}$ between two shapes.

$$\hat{d}_{ij} = f_\psi(\mathcal{P}_i, \mathcal{P}_j, T_{ij}) \tag{1}$$

After training the model, the global collision distance can be determined by the procedure as shown in Figure 2. First, a set of element pairs and corresponding transformations $\mathcal{S}(q) = \{(\mathcal{P}_i, \mathcal{P}_j, T_{ij}(q))\}_{i,j}$ in the given joint configuration q is extracted from the target robot system. Next, PairwiseNet determines the pairwise collision distance between each element pair in $\mathcal{S}(q)$, and the minimum distance found among these is taken as the global collision distance of the robot system. The global collision distance estimator function F_ψ can be expressed in the form of

$$\hat{d}_{\text{col}}(q) = F_\psi(q; f_\psi, \mathcal{S}) \tag{2}$$

$$= \min_{(\mathcal{P}_i, \mathcal{P}_j, T_{ij}(q)) \in \mathcal{S}(q)} f_\psi(\mathcal{P}_i, \mathcal{P}_j, T_{ij}(q)) \tag{3}$$

where $\hat{d}_{\text{col}}(q)$ is the estimated global collision distance in the joint configuration q. Using the batch computation of the neural network model, we can efficiently estimate the minimum distances of element pairs.

3.2 Network Architecture

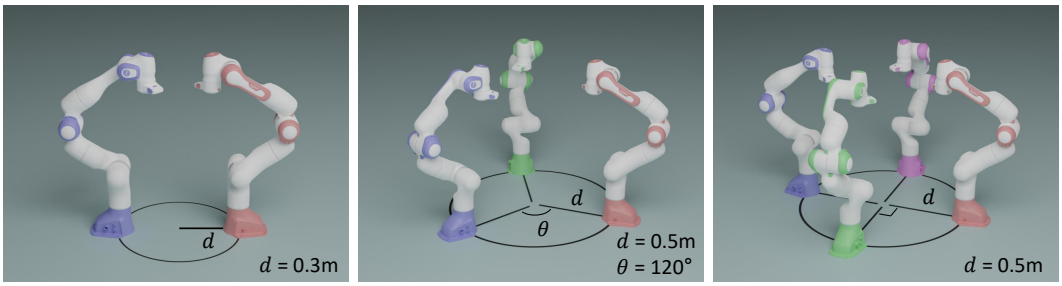
PairwiseNet consists of two main components: an *encoder* that creates a shape feature vector from a point cloud data of a geometric shape (Figure 2b), and a *regressor* that predicts the minimum distance between two shape feature vectors and a transformation (Figure 2c). The encoder employs two EdgeConv layers from Dynamic Graph Convolutional Neural Network [15] to extract 32-dimensional shape feature vectors from the point cloud data. The regressor then combines the two shape feature vectors and the transformation into a single vector and uses four fully connected layers with hidden state dimensions of $(128, 128, 128)$ to output the minimum distance (Figure 2c). The training of PairwiseNet uses a mean-squared error (MSE) between the estimated and actual collision distances as the loss function

$$L = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\mathcal{P}_i, \mathcal{P}_j, T_{ij}, d_{ij}) \in \mathcal{D}_{\text{train}}} ||f_\psi(\mathcal{P}_i, \mathcal{P}_j, T_{ij}) - d_{ij}||^2 \tag{4}$$

where $d_{ij} \in \mathbb{R}$ is the ground-truth pairwise collision distance, and $\mathcal{D}_{\text{train}}$ denotes the training dataset.

3.3 Efficient Inference Strategy of PairwiseNet

Our approach includes an efficient inference strategy for the global collision distance calculation by eliminating the need to run the encoder, a deep neural network that transforms the point cloud data



(a) (b) (c)

Figure 3: Test environments for the collision distance learning performance evaluation. We selected (a) two arms, (b) three arms, and (c) four arms robot systems.

into feature vectors. Since the point cloud data of element pairs remains unchanged regardless of the joint configuration, shape feature vectors of element pairs can be calculated and saved once for each robot system before calculating the collision distance. Using these pre-calculated shape feature vectors, PairwiseNet is able to estimate the collision distance only using the regressor, a simple neural network composed of fully-connected layers. Implemented in PyTorch [16], PairwiseNet is capable of performing collision distance estimation for a joint configuration in less than 0.5ms.

4 Experiments

4.1 Collision Distance Learning for Multi-arm Robot Systems

Target Systems For the test environments, we selected three high-dof multi-arm robot systems as illustrated in Figure 3. We employed 7-dof Franka Emica Panda robot arms for our test environments.

- **Two arms**: the system consists of two arms positioned facing each other, with a separation distance of 0.6 meters, with a total of 14-dof.
- **Three arms**: the system consists of three equidistant arms on a circle of radius 0.5 meters, each facing towards the center, with a total of 21-dof.
- **Four arms**: the system consists of four equidistant arms on a circle of radius 0.5 meters, each facing towards the center, with a total of 28-dof.

For the test dataset of each target environment, we sampled one million joint configurations from a uniform distribution within the joint limits.

Training Dataset for PairwiseNet We collected pairwise collision distance data from dual-arm robot manipulation systems. For the diversity of the dataset, we utilize dual-arm robot systems with various relative positions between the two arms as illustrated in Figure 4. We sample θ and ϕ with eight equally spaced values within the range $[0, 2\pi)$, and sample R with five equally spaced values within the range $[0.1\text{m}, 1.0\text{m}]$. In total, we use 320 different combinations of (R, θ, ϕ), resulting in 320 different dual-arm robot systems. For each system, we sampled joint configurations uniformly within the joint limits and extracted a set of element pairs $\mathcal{S}(q)$ at each joint configuration q (Figure 2a). To obtain the ground-truth pairwise collision distance $d_{ij} \in \mathbb{R}$ between the element pair, We use the collision distance estimation

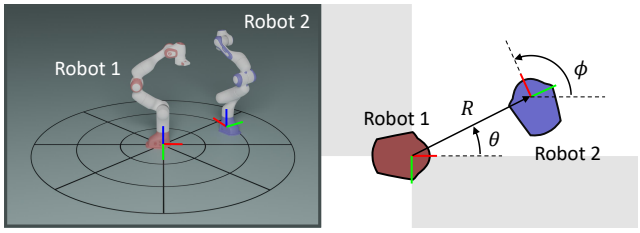


Figure 4: An illustration of multi-arm robot systems for generating the training dataset. Training data points are generated from dual-arm robot environments with various relative positions between two arms.

algorithm implemented in PyBullet [17]. If the two elements collide, the collision distance is the negative of their penetration depth. The resulting training dataset $\mathcal{D}_{\text{train}}$ contains 3 million data points.

Baselines We trained our method and other existing collision distance estimation methods.

- **Capsule**: A bounding volume method with capsule-shape collision primitives used in [18, 6].
- **JointNN**: A fully-connected neural network model that directly uses joint configurations as inputs (the input representation used in [6, 13]).
- **PosNN**: A fully-connected neural network model that uses joint positions as inputs (the input representation used in [10, 9]).
- **jointNERF** [11]: A fully-connected neural network model that uses positional embedding vectors of joint configurations as inputs.
- **ClearanceNet** [12]: A neural network model that takes joint configurations as inputs and utilizes two fully-connected layers, each followed by a dropout layer.
- **DiffCo** [8]: A kernel perceptron model that takes joint configurations as inputs and outputs the collision score.

The existing collision distance learning methods were trained on one million uniformly sampled data points within the joint limits for each target system. For the DiffCo model, we were limited to a dataset size of 50,000 data points, which was the maximum feasible size for kernel perceptron training on our hardware (AMD Ryzen Threadripper 3960X, 256GB RAM, and NVIDIA GeForce RTX4090 with 24GB VRAM).

Performance Evaluation We evaluate the performance of collision distance learning using four metrics: MSE, AUROC, Accuracy, and Safe-FPR. These metrics target both the collision distance regression and collision classification, with a robot configuration being classified as a collision if the collision distance is below the threshold ϵ (for DiffCo, if the collision score is above the threshold). MSE represents the mean squared error between the ground truth and estimated global collision distance. AUROC is the area under the receiver operating characteristic curve for the collision classification. Accuracy is the classification accuracy of collisions with the threshold $\epsilon = 0$. Lastly, in order for the trained collision distance estimation model to be used in actual path planning tasks, a sufficiently conservative threshold must be used to ensure that collisions cannot occur. However, the more conservative the threshold used, the more false alarms will occur where non-collision robot configurations are incorrectly classified as collisions. Safe-FPR is used to evaluate performance in these situations, representing the false alarm rate when using the least but sufficient conservative threshold that can classify all collision configurations in the test dataset as collisions.

Table 1 presents the evaluation results of PairwiseNet and other existing methods. PairwiseNet outperforms in terms of all performance metrics in all three multi-arm robot systems compared to the existing methods, with the top-performing metrics highlighted in bold. Notably, PairwiseNet even shows better performance metrics than Capsule, a computationally expensive BV-based collision distance estimation method.

Figure 5 shows the collision distance estimation results for the four-arm robot system during a collision-free trajectory. The robot configurations at the start and end points of the path involve all four robot arms intricately intertwined, so the robot arms move through the near-collision region. The ground-truth collision distance and the estimated collision distances by PairwiseNet and other baselines during the trajectory are represented in the bottom-right plot. The performance of PairwiseNet can be confirmed, as it is the only method that accurately estimates the complex ground-truth collision distance of the four-arm robot system.

Generalizability Existing learning methods have used individual models trained from each system's individual dataset to estimate collision distance in the three target robot systems. In contrast, PairwiseNet estimates collision distance with only one model for all three target robot systems, and

Table 1: Collision Distance Estimation Performances

Env.	Methods	MSE	AUROC	Accuracy ($\epsilon = 0$)	safe-FPR
Fig. 3 (a) (Two arms)	Capsule	5.47e-4	0.9995	0.9776	0.0247
	JointNN	3.63e-4	0.9955	0.9794	0.3200
	PosNN	2.71e-4	0.9970	0.9823	0.1476
	jointNERF	2.98e-4	0.9962	0.9808	0.2371
	ClearanceNet	1.11e-3	0.9853	0.9621	0.4570
	DiffCo*	-	0.9824	0.9818	0.3141
	PairwiseNet (our)	**0.24e-4**	**0.9998**	**0.9941**	**0.0200**
Fig. 3 (b) (Three arms)	Capsule	5.96e-4	0.9993	0.9775	0.0241
	JointNN	9.69e-4	0.9902	0.9721	0.2679
	PosNN	5.41e-4	0.9951	0.9801	0.1336
	jointNERF	8.22e-4	0.9920	0.9747	0.2213
	ClearanceNet	4.63e-3	0.9499	0.9395	0.6067
	DiffCo*	-	0.9603	0.9453	0.5858
	PairwiseNet (our)	**0.24e-4**	**0.9997**	**0.9944**	**0.0189**
Fig. 3 (c) (Four arms)	Capsule	6.66e-4	0.9986	0.9468	0.0694
	JointNN	1.59e-3	0.9718	0.9183	0.6988
	PosNN	7.18e-4	0.9885	0.9478	0.5371
	jointNERF	1.30e-3	0.9778	0.9280	0.6260
	ClearanceNet	6.67e-3	0.8738	0.8202	0.9965
	DiffCo*	-	0.8811	0.8306	0.9874
	PairwiseNet (our)	**0.46e-4**	**0.9994**	**0.9858**	**0.0650**

*DiffCo outputs the collision score from -1 (collision-free) to 1 (collision).

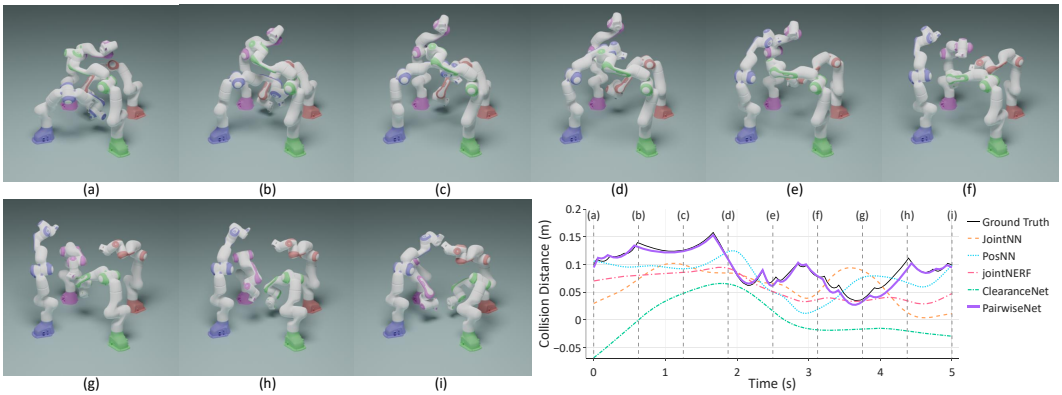


Figure 5: Collision distance estimation for the four-arm robot system: Images depict the four-arm robots following a trajectory, and the plot illustrates ground truth and estimated collision distances of PairwiseNet and other baselines during the trajectory.

its performance is even better than that of existing methods that use individual models for each system. Therefore, PairwiseNet, trained with sufficient datasets, can apply the same model without additional training, even if the robot base positions or number of robot arms change.

4.2 Collision Detection in a Real-world Environment

We perform real-world environments with a 7-dof Panda robot arm (Figure 6). This workspace is populated with obstacles such as shelves and tables that add complexity to the robot arm's operational landscape. To validate the performance of our collision distance estimation, we provide a human-guided demonstration with the robot arm occasionally sweeping close to, and sometimes colliding with, tables and shelves.

The training procedure of PairwiseNet and other baselines, as well as the dataset and model architectures used, are identical to those employed in the experiments for multi-arm robot systems. In the case of PairwiseNet, the dataset consists of element pairs representing tables and shelves, which

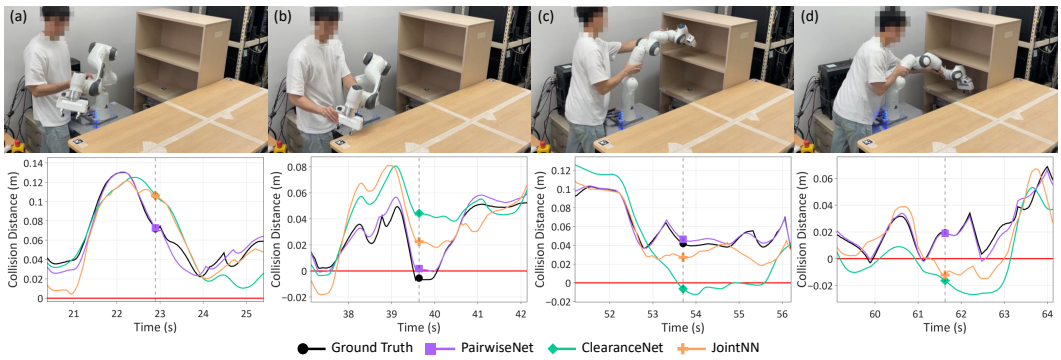


Figure 6: Collision distance estimation in the 7-dof Panda robot arm environment with obstacles (tables and shelves). The top images display a human-guided robot arm, while the corresponding plots at the bottom illustrate the ground truth and estimated collision distances from PairwiseNet and other baselines at time $t =$ (a) 22.9s, (b) 39.6s, (c) 53.7s, and (d) 61.6s, respectively.

are further divided into sub-parts. Specifically, a table is divided into one plate and four legs, while a shelf is divided into six plates. The pairwise collision distances between these sub-parts and the robot links are calculated by PyBullet for the training dataset.

The collision distance estimation results from the demonstration are presented in Figure 6. Note that Figure 6 displays only four snapshots from the demonstration; the complete video can be accessed in the supplementary materials. Compared to the evaluated baselines, PairwiseNet consistently exhibits the highest accuracy in estimating the actual collision distance. It reliably detects collisions between the robot arm and obstacles in the majority of cases. In contrast, the other baselines either fail to trigger collision alarms when a collision occurs (Figure 6 (b)) or produce false collision alarms when no collision is present (Figure 6 (c), (d)). PairwiseNet stands out by consistently and accurately identifying collisions between the robot and obstacles.

5 Conclusions

In this paper, we presented PairwiseNet, a novel collision distance estimation method that estimates the minimum distance between a pair of elements instead of directly predicting the global collision distance of the robot system. By simplifying the problem into smaller sub-problems, our approach achieves significant performance improvements for high-dof robot systems compared to methods that directly predict the global collision distance. Additionally, PairwiseNet is capable of handling environmental changes such as robot base repositioning without requiring additional training or fine-tuning. We evaluate and compare the collision distance estimation performance of PairwiseNet for both high-dof multi-arm robot systems and single-arm systems in the presence of obstacles, and validate its accurate collision distance estimation and generalization to environmental changes.

Limitations and Future works The generalizability of PairwiseNet is currently limited to systems that involve small changes in the environment and exclusively consist of known shape elements. While PairwiseNet demonstrates robust performance in estimating collision distances within this scope, its applicability to environments with significant variations or unknown shape elements have not been fully investigated. Future work is aimed at enhancing the generalizability of PairwiseNet to handle larger environmental changes, and to accommodate systems that include previously unseen shape elements. This could involve, e.g., expanding the training dataset to incorporate a wider range of environmental variations, and incorporating techniques for handling unknown or novel shape elements.

References

[1] S. M. LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[2] J. J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[3] J. Huh and D. D. Lee. Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–69. IEEE, 2016.

[4] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar. Real-time (self)-collision avoidance task on a hrp-2 humanoid robot. In *2008 ieee international conference on robotics and automation*, pages 3200–3205. IEEE, 2008.

[5] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger. Integration of reactive, torque-based self-collision avoidance into a task hierarchy. *IEEE Transactions on Robotics*, 28(6): 1278–1293, 2012.

[6] C. Fang, A. Rocchi, E. M. Hoffman, N. G. Tsagarakis, and D. G. Caldwell. Efficient self-collision avoidance based on focus of interest for humanoid robots. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1060–1066. IEEE, 2015.

[7] J. J. Quiroz-Omaña and B. V. Adorno. Whole-body control with (self) collision avoidance using vector field inequalities. *IEEE Robotics and Automation Letters*, 4(4):4048–4053, 2019.

[8] Y. Zhi, N. Das, and M. Yip. Diffco: Autodifferentiable proxy collision detection with multi-class labels for safety-aware trajectory optimization. *IEEE Transactions on Robotics*, 2022.

[9] N. B. Figueroa Fernandez, S. S. Mirrazavi Salehian, and A. Billard. Multi-arm self-collision avoidance: A sparse solution for a big data problem. In *Proceedings of the Third Machine Learning in Planning and Control of Robot Motion (MLPC) Workshop.*, number CONF, 2018.

[10] D. Rakita, B. Mutlu, and M. Gleicher. Relaxedik: Real-time synthesis of accurate and feasible robot arm motion. In *Robotics: Science and Systems*, pages 26–30. Pittsburgh, PA, 2018.

[11] 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 *Conference on Robot Learning*, pages 750–759. PMLR, 2022.

[12] J. Chase Kew, B. Ichter, M. Bandari, T.-W. E. Lee, and A. Faust. Neural collision clearance estimator for batched motion planning. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 73–89. Springer, 2020.

[13] M. Koptev, N. Figueroa, and A. Billard. Real-time self-collision avoidance in joint space for humanoid robots. *IEEE Robotics and Automation Letters*, 6(2):1240–1247, 2021.

[14] Y. Kim, J. Kim, and D. Park. Graphdistnet: A graph-based collision-distance estimator for gradient-based trajectory optimization. *IEEE Robotics and Automation Letters*, 7(4):11118–11125, 2022.

[15] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[17] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[18] A. El Khoury, F. Lamiraux, and M. Taix. Optimal motion planning for humanoid robots. In *2013 IEEE international conference on robotics and automation*, pages 3136–3141. IEEE, 2013.