Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 4, No. 2, 2023

Robot Motion Planning with Optimization-based Algorithm and Neural Network: Unsupervised Path Regression

Yunfeng Kang, Yixin Mai

School of Computation, Information and Technology, Technical University of Munich, Germany

Abstract: Planning collision-free motions for robots is challenging in environments with complex obstacle geometries. In this study, we combine the optimization-based algorithm and neural network to obtain a short and collision-free path through unsupervised path regression. The network can help to predict an educated initial path for an optimization-based planner, which will converge to a better solution. We compare different path representations, world representations, and also their combinations to improve the result. The motion planning problem is extended from a sphere robot in a single 2D world to a more complex static arm robot in multiple worlds. Through our experiments, we can find that in multiple worlds, the relative path with the signed distance field is the best combination for both robots. This is not only proved by metrics of feasible rate and loss but also by the visualization in the map.

Keywords: Robot Motion Planning; Optimization-based Algorithm; Neural Network; Path Regression; Path and World Representations.

1. Introduction

Robot motion planning is a critical domain of robotics that involves the strategic determination of a path for a robot to navigate from an initial joint configuration to a desired goal configuration. This process is crucial for enabling robots to autonomously perform complex tasks in various applications, ranging from industrial automation to autonomous vehicles and even in healthcare settings. The primary objective of robot motion planning is to ensure the successful execution of movements while addressing two key challenges: ensuring the avoidance of obstacles within the environment and preventing self-collision throughout the trajectory; the path of the robot's motion is as short as possible.

The utilization of deep learning and neural networks in robot motion planning has witnessed a remarkable surge in recent years. As robots become increasingly integrated into diverse applications and operate in complex environments, the need for advanced and efficient path-planning techniques has become more pressing. Traditional approaches to motion planning often struggle to handle the intricacies of real-world scenarios, leading to the exploration of novel methodologies that leverage the power of deep learning and neural networks.

1.1. Related Work

There are two widely known methods for addressing motion tasks in robotics: sampling-based planners (SMP) and optimization-based motion planners (OMP). These approaches differ significantly in their fundamental principles. SMP relies on randomness as its foundation and can ensure the discovery of a feasible and collision-free path on a global scale, if one exists. Rapidly exploring random trees (RRT) [1], probabilistic roadmap (PRM) [2], and their extensions are prominent examples of SMP, where they iteratively explore the configuration space and construct a graph of potential configurations until a branch reaches the desired goal. However, vanilla versions of SMP encounter challenges when applied to complex robots because the search space expands exponentially with the number of degrees of freedom (DoF). Consequently, their scalability is limited in such cases.

OMP, on the other hand, tackles the motion task by transforming it into an optimization problem [3] [4] [5]. It employs gradient-based techniques for non-convex optimization to search for a suitable solution. In theory, these methods have the potential to scale linearly with the number of degrees of freedom (DoF) and can rapidly converge to a smooth trajectory. This characteristic allows OMP to handle motion planning efficiently, offering advantages in terms of computational complexity.

Furthermore, alternative learning methods have been employed in tackling this problem. For instance, Jurgenson and Tamar [6] leveraged reinforcement learning combined with convolutional layers to process the occupancy map of the environment for 2D serial robots. They employed a classical planner as a fallback option when random exploration failed to find a viable solution, thereby utilizing expert knowledge to guide the training process implicitly. In a different approach, Pandy et al. [7] introduced a novel methodology that eliminates the need for dataset generation. Instead, they directly trained the network using the planning problem's objective function as the training loss, effectively bypassing the use of supervision. However, their approach was limited to representing environments with only a few obstacles using geometric primitives, which constrained its flexibility.

1.2. Contributions

Robot motion planning is a widely researched topic recently. In addition to the traditional optimization-based algorithms, nowadays the neural network [7] and reinforcement learning [8] can also be integrated to solve the motion planning problem. This paper presents a deep learning enhancement for an optimization-based planner in random environments with complex obstacle geometries. To sum up, our main contributions are as follows:

- Concatenation of the neural network and the optimization-based planner. The gradients of both can be

utilized together to further improve the predicted path of motion planning.

- Three kinds of path representation are applied, namely the global path, the relative path, and the Non-uniform rational B-spline. They have their own characteristics and properties in different cases.
- Different world representations are also tried in our study. The binary occupancy grid and the signed distance field are used to represent the environments. The latter is more beneficial for motion planning tasks as it contains more position and distance information.
- Two kinds of robots are tested in our experiments, the sphere robot and the static arm robot respectively. This can make our experimental results more convincing.

2. Method

The motion planning problem is described in Fig. 1 by the world (hatched), the start configuration (green), and the goal configuration (red). The trajectory, which should be the short and feasible path from start to goal, is drawn in blue. Two different robots will be tested in random 2D environments, generated with simplex noise. The first robot is the sphere bot, and the second one is the static arm robot with 3 joints.

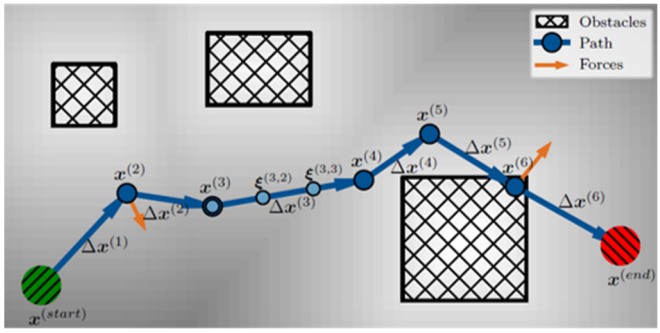


Fig 1. Demonstration of motion planning task

The sampled non-collided start and end point pairs are fed into the neural network to generate and optimize a path that can minimize the objective function, namely the sum of the weighted length and collision loss. The red points in Fig. 2 are the points colliding with obstacles, which are not qualified start and end points, and the blue points are the ones that meet the requirement.

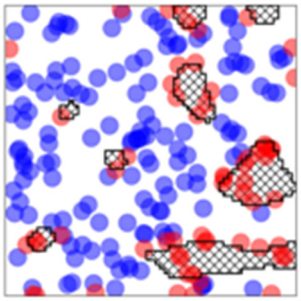


Fig 2. Sample points colliding and not colliding with obstacles

We formulate the motion planning task of getting from point A to point B as an optimization problem with the desired path as the optimum. The path q consists of a discrete set of waypoints of joint configurations. The task is encoded by an objective function O(q) in Eq. 1 measuring the quality of a given path. This specific objective is composed of two relevant terms, length cost and collision cost respectively. The length cost O_L and collision cost O_C are defined in [3].

$$O(q) = O_L(q) + \lambda O_C(q) \tag{1}$$

Here λ is a variable that measures the importance of collision cost. The larger λ means that the objective pays more attention to avoiding collisions. Then the trajectory could be not so short.

In this paper, we want to combine the existing optimization-based CHOMP [3] with a neural network to predict a short and collision-free path for robots. The pipeline is shown in Fig. 3. OMP itself can iteratively obtain a better path by updating the gradient, and the neural network also has the ability to learn a better path after training. Concatenating them both will easily result in a better path.

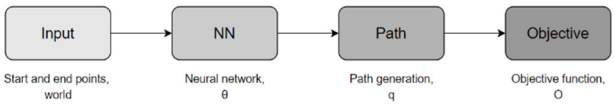


Fig 3. Concatenation of the neural network with the optimization-based algorithm

During backpropagation, the full gradient $\frac{dO}{d\theta} = \frac{dO}{dq}\frac{dq}{d\theta}$ is used to update the weights of our neural network. The first part $\frac{dO}{dq}$ can be calculated from the objective function from CHOMP and the second part $\frac{dq}{d\theta}$ can be obtained from PyTorch automatic differentiation engine.

$$\frac{dO}{d\theta} = \frac{dO}{dq}\frac{dq}{d\theta}$$

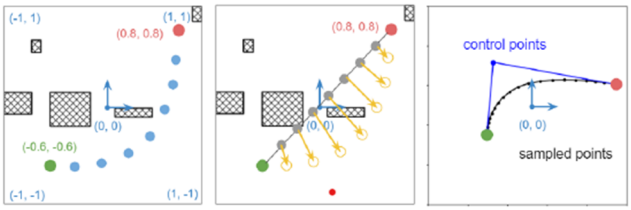
We focus on different path representations in order to get a better result and then world representations to make problems solvable in multiple worlds. In II-A, II-B, and II-C we will discuss different path representations and world representations.

2.1. Path Representations for a Sphere Robot

The simplest and most intuitive path representation for a sphere robot is the global coordinates of the waypoints from start to end (hereinafter called "global"), as shown in Fig. 4 (a).

The second representation is similar but with different initialization. We first connect start and end points and interpolate to get the uniformly distributed points in the connecting line, then the path can be represented as the relative coordinates to the previous points (hereinafter called "relative") as Fig. 4 (b).

The last way is the non-uniform rational B-spline. Basically, it uses control points to form a very smooth spline and the waypoints of the path can then be sampled from it (hereinafter called "Nurbs") as Fig. 4 (c).



(a) global (b) relative (c) Nurbs

Fig 4. Path representations

To make things much easier for the network, the world

coordinates are normalized to [-1, 1] with the center coordinate [0, 0].

2.2. Path Representations for a Static Arm Robot

The static arm robot is a robot whose base point is fixed at the center of the world and has multiple joints, as shown in Fig. 5. In our study, we choose the robot with 3 joints.

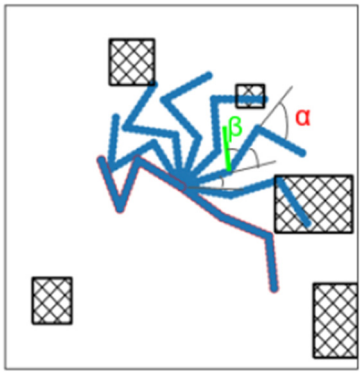


Fig 5. Evenly divided waypoints of the static arm robot

The first path representation is the global path as well. The global coordinates of the robot are represented by the rotation angles of its joints, ranging from $-\pi$ to π. The robotic arms with red outlines in Fig. 5 are the start and end states, with six waypoints in between. An example of the waypoint coordinates in Fig. 5 is $(\frac{\pi}{6}, \frac{\pi}{4}, -\frac{\pi}{3})$ and α corresponds to $-\frac{\pi}{3}$.

The second one is the relative path. The relative coordinates are the differences between the angles of the actual waypoints and the evenly divided waypoints. Fig. 5 shows the state of each waypoint in the initial unlearned state. If the second joint is at the position of the thick green line, the angle difference would be β. The differences would be learned in the model during training.

Nurbs cannot be used here because the paths will not be smoother by using control points in the 2D world. In the future, if the motion planning task of the robotic arm in 3D space needs to be solved, Nurbs would be useful.

2.3. World Representations

There are two kinds of images to represent the world. One is the binary occupancy grid in Fig. 6, the black and white figure (hereinafter called "binary"). The other is the signed distance field in Fig. 7 (hereinafter called "SDF"). On the boundary of each obstacle, the distance value is zero, inside the obstacles the value is negative, and outside positive.

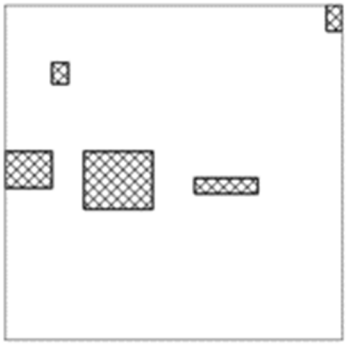


Fig 6. Binary occupancy grid

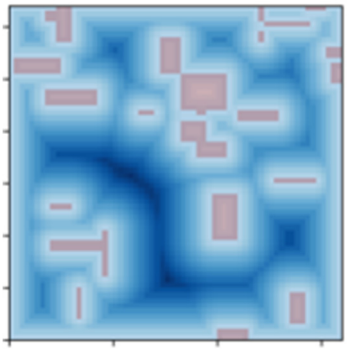


Fig 7. Signed distance field

We use CNN to process the black and white figures and use the multi-layer perceptron (MLP) network to process the downsampled and flattened SDF. The whole structure looks like Fig. 8. The pairs and worlds are processed respectively and then merged together to be further processed by another MLP network, which will regress the coordinates of the trajectory result.

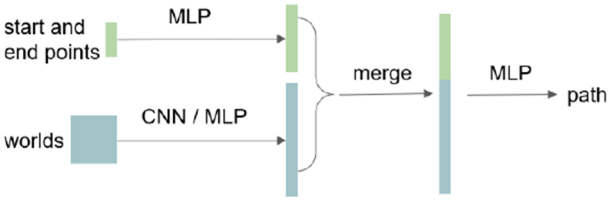


Fig 8. Network structure

To match worlds and pairs, the vector that encodes world information should be copied as many times as the number of the pairs in the world and then be concatenated to the vector that encodes the single pair. The new vector is then a single motion planning task corresponding to a start and end pair in a specific world.

3. Experiments and Results

Firstly, we need to determine the best value for the λ parameter. Our experiment is conditioned on a single two-dimensional world with random rectangular obstacles. We use three different cost weighting between length and collision (1:1, 1:5 and 1:10), hence different objective functions to train the neural network and then track the feasibility rate of the predicted paths for the test dataset. The feasibility results are shown in Fig. 9 and the planned paths are shown in Fig. 10.

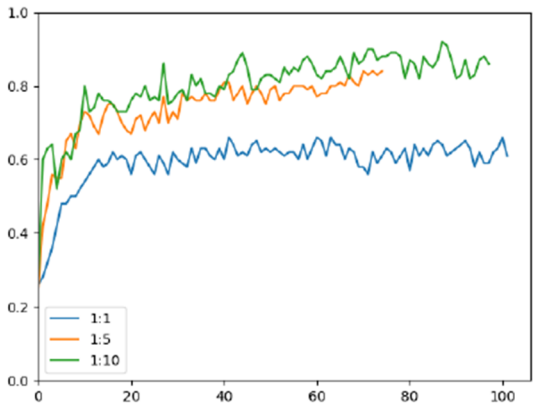
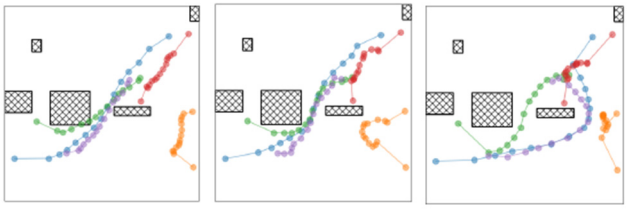


Fig 9. Test feasibility using different length and obstacle collision weighting



(a) 1:1 (b) 1:5 (c) 1:10

Fig 10. Training results using different length and collision weighting

When the ratio is 1:1, the feasibility rate is about 0.6. The network will give the same attention to length and non-collision. After training, the green path in Fig. 9 (a) cannot bypass the square obstacle in the middle in order to keep itself short.

When the ratio is 1:5, the network has a higher test feasibility rate which is about 0.8 because it focuses more on a collision-free path. The green path now can bypass the square obstacle in Fig. 9 (b).

When the ratio is 1:10, the feasibility rate can be even higher, but the predicted paths tend to detour to be safer. Compared to the short and collision-free paths going through the obstacles, the network prefers much safer but longer paths, as the blue and purple paths in Fig. 9 (c). In summary, we will choose λ as 5 for our latter experiments.

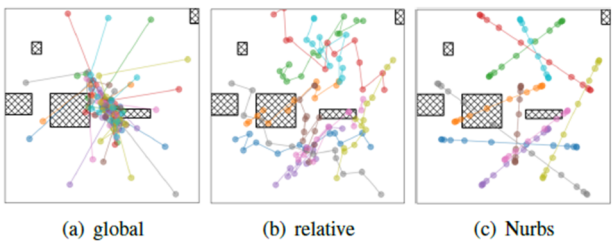
Based on the different path and world representations, we first compared different path representations in a single 2D world for a sphere robot (Section III-A), then we compared different combinations of path and world representations in multiple worlds (Section III-B). After that, we extended the motion planning task from a simple sphere robot to a more complex static arm robot (Section III-C and Section III-D).

3.1. Sphere Robot in a Single World

The settings of the experiment are as follows:

- dataset: 2000 training pairs, 500 test pairs
- model: 8-layer MLP with ReLU, BatchNorm
- optimizer: SGD with lr 0.001 and momentum 0.9
- objective function: length: collision = 1: 5

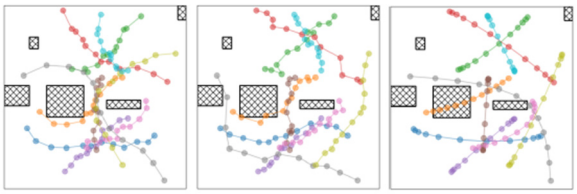
The output paths at the first epoch can be seen as the initial guess of the network based on different path representations, as shown in Fig. 11. All waypoints cluster around the origin as the result of the global representation. For relative, the path has a zigzag but basically follows a straight connecting line. The predicted paths of Nurbs are smooth straight lines.



(a) global (b) relative (c) Nurbs

Fig 11. Predicted paths in the first epoch

After training, the behavior of the predicted paths can be different for the same start and end pairs based on different initialization. For example, the yellow path in Fig. 12 (a) is stuck in a suboptimal solution but the predictions are better in the other two cases. And the gray paths in Fig. 12 are all different.

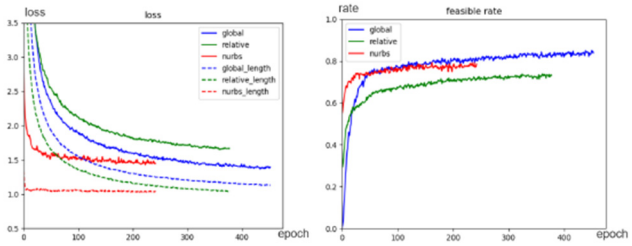


(a) global (b) relative (c) Nurbs

Fig 12. Predicted paths in the last epoch

All methods can work well for the simple tasks of which the optimal solution is the interpolation of start and end points. Nurbs works best in this case, but it is hard for it to represent a winding path such as the orange one in Fig. 12 (c) to avoid the collision because of the limited representation.

The validated total loss, feasible rate, and the average length loss of the feasible paths are tracked during training, as shown in Fig. 13. Global representation has the best feasible rate above 80% because it is flexible enough to predict collision-free paths. Nurbs predicts the shortest path because it has a straight line as a very good initialization. Overall, global is the best path representation in this experiment. The relative has the worst performance despite the flexibility and the good initialization, because it lacks global information about the obstacles and we don't explicitly feed it in the single world.



(a) loss (b) feasible rate

Fig 13. Results of the first experiment

3.2. Sphere Robot in Multiple Worlds

To compare different combinations of path and world representation in multiple worlds, 500 worlds and 500 pairs per world are used for training while 100 worlds and 100 pairs for the test. The other settings keep the same as the first experiment except that the network architecture is changed according to Section II-C. The results are shown in Table I.

Table 1. Results of the Second Experiment

Sphere Robot				
world repr.	path repr.	total loss	length loss	feasible rate
SDF	global	1.68	0.69	72.4%
	relative	1.64	0.66	72.5%
binary	global	1.71	0.71	72.0%
	relative	1.75	0.67	70.6%

Compared with binary worlds, SDF has slightly better results with a lower length loss and a higher feasible rate. It is easy to imagine because the signed distance field has additional distance information relative to the nearest obstacles compared with the binary world. With respect to path representation, relative works better for short paths as before, due to the good initialization, and now also works well

for non-collision paths. Because the global information of multiple worlds is now available for the network. For a sphere robot in multiple worlds, the combination of SDF and relative is the best. We do not use Nurbs in multiple worlds because it lacks flexibility and can only get a poor result.

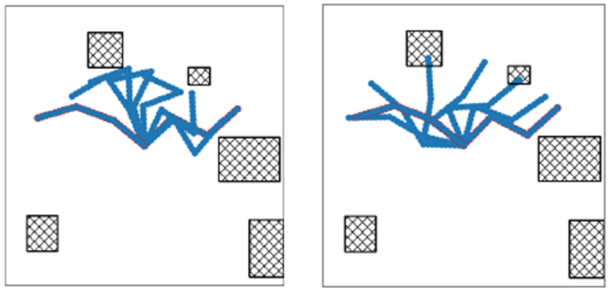
3.3. Static Arm Robot in a Single World

The experiment setting of the static arm robot is similar to Section III-A, except that:

- dataset: 3000 training pairs, 1000 test pairs
- model: MLP, Highway, BatchNorm
- objective function: length : collision = 1 : 1

It is obvious that the global path in Fig. 14 (a) is relatively better at avoiding obstacles. The third joint of the robotic arm can consciously avoid nearby obstacles. In contrast, in the relative path, the robotic arm simply moves from the starting position to the end with a uniform change in the angles of the joints.

The feasible rate and loss of the global path are 70.3% and 0.87, which are both better than that of the relative path, 65.8% and 1.26. This helps to verify from another aspect that in the single world, the global path performs better.



(a) global (b) relative

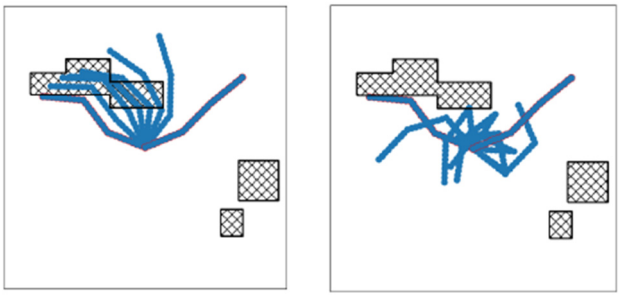
Fig 14. Predicted paths for a static arm robot in the last epoch

3.4. Static Arm Robot in Multiple Worlds

In the last experiment, four combinations in total are compared for a static arm robot in multiple worlds. The neural network used here is changed according to Section II-C. There are 40 worlds and 3000 pairs per world are used for training and 8 worlds and 1000 pairs for testing. Other parameters keep the same as Section III-C.

We first combine and compare the global path with the binary and SDF world respectively. In the unlearned state, the robotic arm tends to choose a simple and direct movement, uniformly moving from above, which is similar to the scenario in Fig. 15 (a). The collision with obstacles will push the model to learn a better path. After training, the robotic arm will finally choose to move from below. It curls up and then stretches itself.

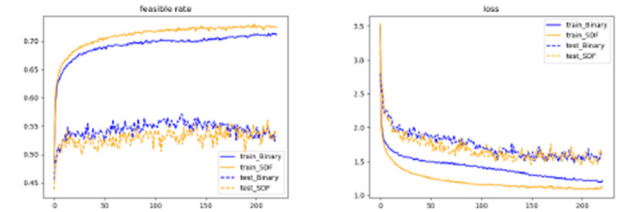
During the training, the network with SDF world representation would be adjusted faster to reach the desired state. This is clearly shown in Fig. 15 which records the states of binary and SDF in the 60th epoch. But in the last epoch, the movement of binary and SDF has no discernible difference. For the relative path, the results are similar.



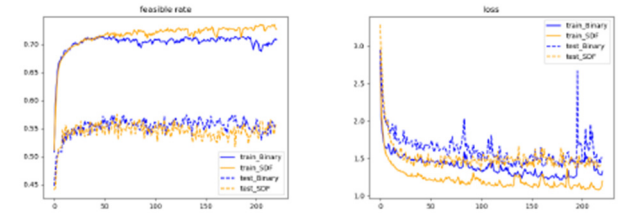
(a) binary (b) SDF

Fig 15. States in the 60th epoch of binary and SDF representation

In order to explain our first conclusion above more intuitively and systematically, four diagrams in Fig. 16 are used to demonstrate them.



(a) Feasible rate of global path (b) Loss of global path



(c) Feasible rate of relative path (d) Loss of relative path

Fig 16. Comparison of different path and world representations

Overall, as shown in Fig. 16 and Table II, the SDF world representation leads to a higher feasible rate and a smaller total loss. In addition, SDF can converge more quickly. With regard to path representation, relative outperforms global in multiple worlds as expected. The characteristics exhibited by the movement of the robotic arm are verified from the curves, and the SDF model outperforms the binary model.

Table 2. Results of the Fourth Experiment

Static Arm Robot			
world repr.	path repr.	total loss	feasible rate
SDF	global	1.59	53.5%
	relative	1.34	56.5%
binary	global	1.63	53.1%
	relative	1.52	55.7%

From the above four experiments, we can draw the conclusions:

- In a single world, global path representation works

better than relative. Because the world information is not explicitly fed in, the relative path has no access to the global information about the obstacles.

- Nurbs works best for the simple task in the 2D world, hence is suitable in worlds with few obstacles. But the path representation is not as flexible as the others and hard to generalize to other cases.
- In multiple worlds, world representation SDF works better, because it has the distance information to the nearest obstacles. Path representation relative then outperforms global due to its good initialization.

4. Conclusion

The key innovative trick of our project is the cascade of the classical optimization-based algorithm and the neural network. It can further improve the performance of the model. A lot of experiments are investigated including two robots, three path representations, and two world representations. In general, we have to balance flexibility and smoothness and should choose the proper representation in a specific setting.

However, the results in the case of multiple worlds can be further improved since the test feasible rates are only about 60%. One possible reason is that if one of the robotic arm's waypoints touches an obstacle, the path will be judged as infeasible, even though its movement seems appropriate.

There could be some possible solutions. Firstly, adding more "hard" pairs to train has the potential to get more useful information about the obstacles and increase the feasible rate. The neural network is easier to predict a simple interpolation path and get a relatively good result if the dataset contains lots of "simple" tasks of which the optimal solution is the interpolation. Secondly, we can find the best setting of the relation between the number of worlds and the number of pairs in each world. This might influence the network to focus more on the world or focus more on the pairs.

In the future, the tasks can be further extended to 3D space and other robots. Also, reinforcement learning can be integrated to learn the optimization algorithm automatically and speed up the motion planning process.

References

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

[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” IEEE transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996.

[3] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “Chomp: Covariant hamiltonian optimization for motion planning,” The International Journal of Robotics Research, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[4] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “Stomp: Stochastic trajectory optimization for motion planning,” in 2011 IEEE international conference on robotics and automation. IEEE, 2011, pp. 4569–4574.

[5] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequential convex optimization and convex collision checking,” The International Journal of Robotics Research, vol. 33, no. 9, pp. 1251–1270, 2014.

[6] T. Jurgenson and A. Tamar, “Harnessing reinforcement learning for neural motion planning,” arXiv preprint arXiv: 1906. 00214, 2019.

[7] M. P´andy, D. Lenton, and R. Clark, “Unsupervised path regression networks,” in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 1413–1420.

[8] K. Li and J. Malik, “Learning to optimize,” arXiv preprint arXiv: 1606.01885, 2016.