Obstacle Avoidance using Dynamic Movement Primitives and Reinforcement Learning

Experimental Details

July 2025

1 Insertion

1.1 Data generation

The precise vertical descend is defined by a 1% tolerance relative to the trajectory length L. Except the S_{scope} that avoids collisions with the ground, a second is set with $\hat{v} = -L$, $m = 0.005L$ away from the goal location for the x positions of the trajectory. Additionally, s_2 is always set with $p_2 = 0.995L$ to constrain the other side of the descend. Five positions for $p_1 \in [0.067L, 0.87L]$ are set uniformly to simulate different arbitrary collision scenarios. The optimization target is set to $S^*_{shape} = -0.33L$.

1.2 Neural network training and performance

In less than 18 minutes one thousand trajectories are generated and the neural network is trained, now with an output dimension of $2 \times 20 = 40$. Testing the model accuracy shows a 100% avoidance success rate with $o = 0.02$, applied as $s_1 + o$, $s_2 - o$.

1.3 Point cloud detection

The peg and the box with the hole must be detected in the point cloud. After removing all points that don't belong to either peg or box, the peg and box can be distinguished by there dimensions. The peg has dimensions of $d_p = 0.04 \times 0.04 \times 0.1$ m, the outer dimensions of the box are $d_b = 0.1 \times 0.1 \times 0.04$ m, the dimensions of the hole are $d_h = 0.045 \times 0.045 \times 0.04$ m. Using DBSCAN as before, two clusters are found and their centers are used to define the positions of the peg and the hole. The task parameters are then computed based on the known object dimensions, the grasp height $h_p = 0.02$ m and the distances D_{0p},D_{ph} between the initial end-effector position P_0 and the peg position P_p and between P_p and the hole position P_h. The task parameters for the pick trajectory are then $s_1^{pick} = h_p$ and

$$s_2^{pick} = D_{0p} - 0.5d_p^{max} - 0.5w_G,$$

and for the insertion, $s_1^{insert} = d_{b,Z}$ and

$$s_2^{insert} = D_{ph} - 0.5d_p^{max} - 0.5d_b^{max},$$

where $d^{max} = \sqrt{2}d_X$ for either the peg or box. This ensures the generation of a collision-free trajectory avoiding the maximum diagonal dimension for both square objects.

2 3D Avoidance

2.1 Data generation

The training scenarios are discretized uniformly, considering 5 different wall configurations $C_w \in [-0.6, 1.0]$ and 6 different gap locations $P_{gap} \in [0, -S^*_{shape}]$, leading to a total number of $5 \times 6 = 30$ $\mathbf{PI^2}$ optimization runs. The wall configurations describe the fraction of the smaller wall where $C_w = 1.0$ is the case they have the same height. Which of the two walls is smaller is defined by the sign, $C_w > 0$ refer to $s_1 < s_2$

and vice versa. For the optimization the first wall is defined by $p_1 = 0.15L$ and $p_2 = 0.2L$, the other wall is defined by $p_3 = 0.8L$ and $p_4 = 0.85L$. In this case, $x_{\mathbf{e}_2}$ value of the trajectory is evaluated in two sections, defined by four points $(p_1, ..., p_4)$, with $\mathcal{I}_{\text{shape}} = [t_1, t_2] \cup [t_3, t_4]$. One wall pair is scaled with $|C_w|$ to keep the desired height difference for each generated trajectory. Besides the S_{scope}^{bottom}, four additional scope cost functions constrain the trajectory with two upper bounds and two lower bounds on the Y and Z coordinates of the trajectory inside the gap defined by $p_5 = 0.35L$ and $p_6 = 0.65L$, with $m = 0.013L$. The collection for the dataset starts once the scope constraints are satisfied. The optimization target is set to $S_{shape}^{*} = -0.33L$. Given the tight constraints for this model, the PI^2 optimization takes more time per run, compared to the previous models. For that reason, and because the sharp direction changes require higher acceleration and jerk, the $S_{acc,0}$ and S_{jerk} costs are omitted here.

2.2 Neural network training and performance

The PI^2 optimization of 30 runs generate 7.4k trajectories. The NN model outputs $3 \times 60 = 180$ forcing term parameters. Tests show more overfitting to the training data than the previous models. The interpolation between the ratios of the two wall heights C_w is not captured accurately enough to achieve a 100% avoidance success rate. When selecting scenarios among the same C_w used in the training, an offset of $o = 0.1$, applied to s_1, results a 100% success rate, interpolating only along the gap position P_{gap}. Unlike avoiding around or over an obstacle, a gap does not allow setting an offset, as collision can occur on both sides. The tests show that the learned model produces maximum errors in the gap of $e_{\mathbf{e}_3} = [-0.007L, 0.03L]$.

2.3 Point cloud detection

As the x and y positions of the walls are constant and known, their height in w_Z is measured by taking all points within a small area and taking their mean z value. The location of the gap is known to be along the y axis. The distances between all points within a small area along the y axis are computed. The maximum y distance is between the points on either side of the gap. From this, the inputs for the neural network can be computed: $s_1^{NN} = max(w_{1,Z}, w_{2,Z})/L + o$ is the maximum relative height plus offset, $s_2^{NN} \in [-0.6, 1.0]$ maps the order of both walls and the ratio of their heights to fit the training configurations C_w. s_3^{NN} is computed as s_1^{NN} in the previous experiments, but without adding an offset.