Computer Vision Term Project:
3D Keypoints Prediction for Peg-insertion

Group 28,　109350008 張詠哲　110705013 沈昱宏

Dataset and code: https://github.com/zyz-2299mod10/CV_final

Introduction

Peg insertion is a significant challenge in robotics, with applications in assembly tasks across various scenarios, such as factory, household. This task requires high precision, so it remains challenging due to its low tolerance for deviations.
As the following figure shows. In our task, the peg, hole and obstacles are randomly placed in the workspace. We use a pre-defined grasping pose to grasp the object and move it to the hole. Therefore, Our task represents two challenges. 1) How to plan a trajectory in a cluttered workspace without causing collisions between the robot, the target object, and obstacles. 2) How to control the delicate motions from grasping a horizontal peg to insert it.

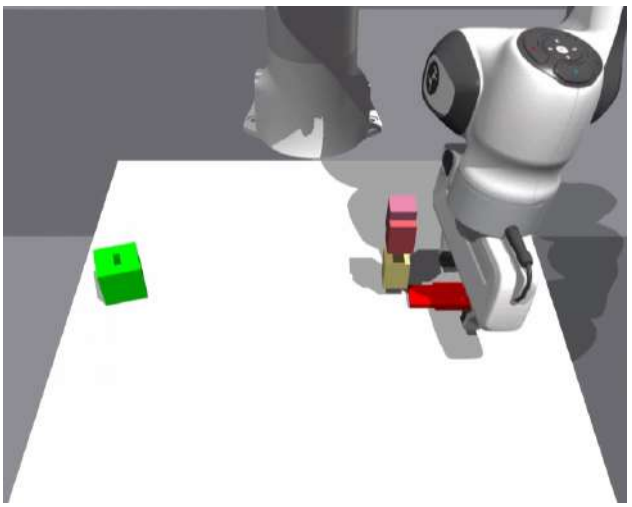

Figure 1. Our task Schematic diagram

We found that although the existing methods achieve precise insertion, there are 2 problems in their experimental setting. 1) The camera in their experimental setting needs to be close enough to the hole, which might not be applicable in long distance planning. 2) Experimental environments are clean, having no obstructions blocking the path, which might not be applicable in real-world scenarios.
Thus, In this project. We aim to combine existing methods, OAKN (hole-pose predictor in CFVS [1]) and cuRobo [2] to achieve collision-free peg insertion. And we will focus more on extending the flexibility and diminishing the sim2real gap.

Method

Framework

The overall framework shows in the following figure. For the coarse pipeline, we only want to get the coarse hole position. When the robot moves to a position that is close enough for the hole, we will use the fine pipeline to get the more accurate hole position and the rotation. One thing worth mentioning is that we added a small Gaussian noise (mean = 0, variance = 0.002) to the depth image in order to diminish sim2real gap. The following is the detailed explanation about pipeline.

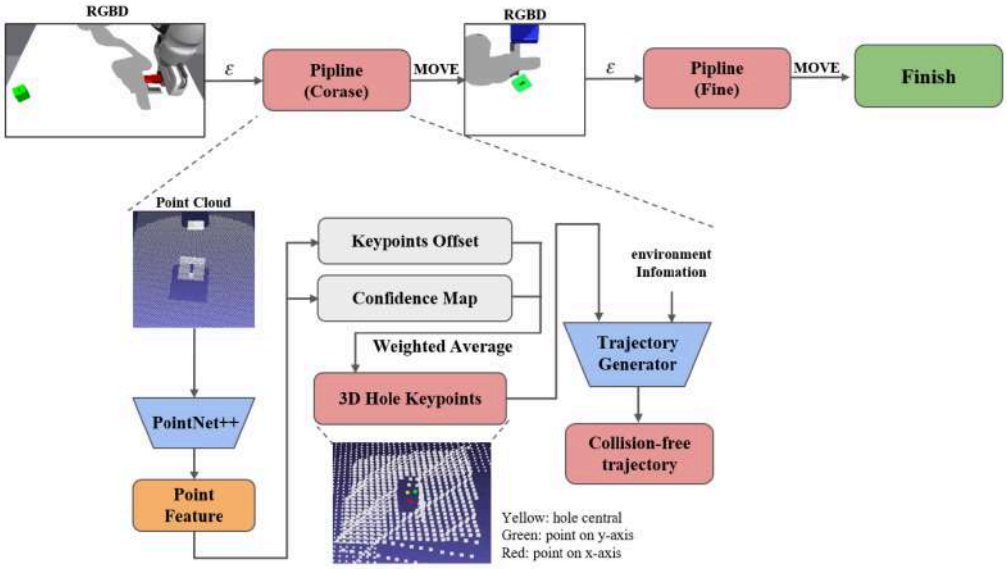

Figure 2. Our framework

Our pipeline contains 2 parts: hole pose estimation and the trajectory generation. Because this course is computer vision, we will focus more on the hole pose estimation part.

1. Hole pose estimation

In this part, we take the hole pose estimation architecture from CFVS [1], and conduct some modification. They first use the PointNet++ to extract the input point cloud feature for further prediction (keypoints offset, confidence map). Finally we want to get the 3 keypoints $\{k_1, k_2, k_3\}$, which represent the hole central, point on y-axis, point on x-axis.

1.1. Keypoints offset

This step is to get the keypoints candidates of each point on the point cloud. We need to predict the keypoints offset $\{\Delta k_{i,1}, \Delta k_{i,2}, \Delta k_{i,3}\}^{N}_{i=1}$ on every points, then get the potential keypoints position by adding the point position to the predicted offset $\{k_{i,1}, k_{i,2}, k_{i,3}\}^{N}_{i=1} = x_i + \{\Delta k_{i,1}, \Delta k_{i,2}, \Delta k_{i,3}\}^{N}_{i=1}$.

1.2. Confidence map

This step is to focus on the important point. We predict the weights $\{w\}_{i=1}^{N}$, $w \in [0, 1]$ of each point on the point cloud.

1.3. 3D hole keypoints

Once we get the keypoints candidate and the weight on every point. We finally get the 3 keypoints we want by computing the weighted average. But we found that it will cause little error due to the noise. Therefore, we conduct ICP (iterative closest points) to tune the hole pose. By aligning the socket CAD model to the predicted hole pose, then calculate the transformation matrix between predicted hole pose and the observation hole pose by ICP. Finally we can get the tuned keypoints by dot product the transformation matrix to the origin keypoints.

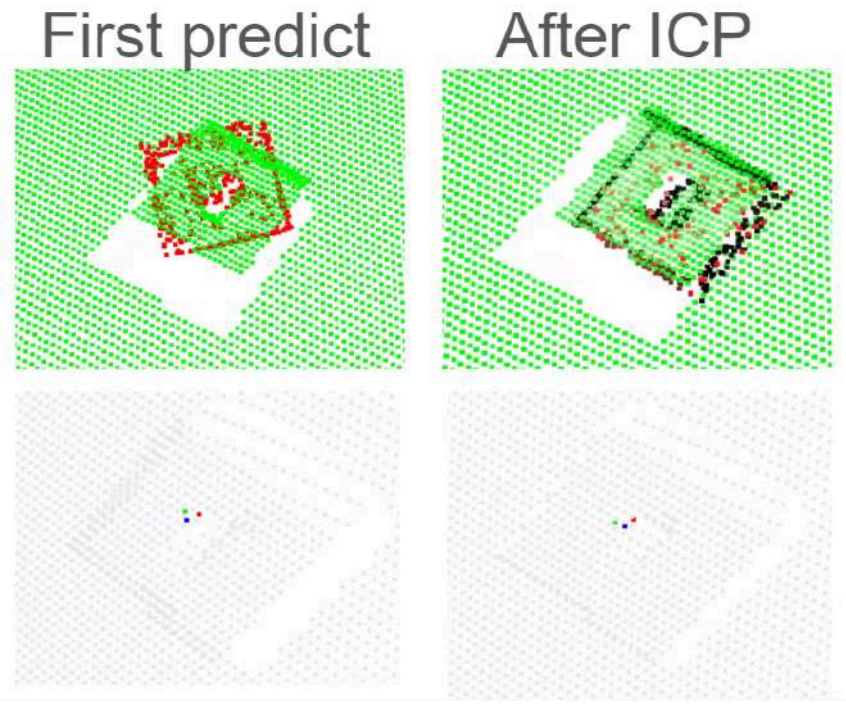

Figure 3. ICP comparison

2. Collision-free trajectory generation

In this part, we use cuRobo [2] as our trajectory generator. It takes planning as an optimization problem with cost function and constraint. Make sure the robot and peg won't collide with the environment.

$$argmin_{\theta[1,T]} C_{task}(X_g, \theta_T) + \sum_{t=1}^{T} C_{smooth}(.)$$

Final time-step

Target pose

Data collection

Our dataset contains 2 parts: side-view and the close-view. In each dataset, we will collect the point cloud which is reconstructed by the depth image, keypoints offset, and confidence map. We collect this data in IsaacGym, which is a simulator. And add some Gaussian noise on the depth image.

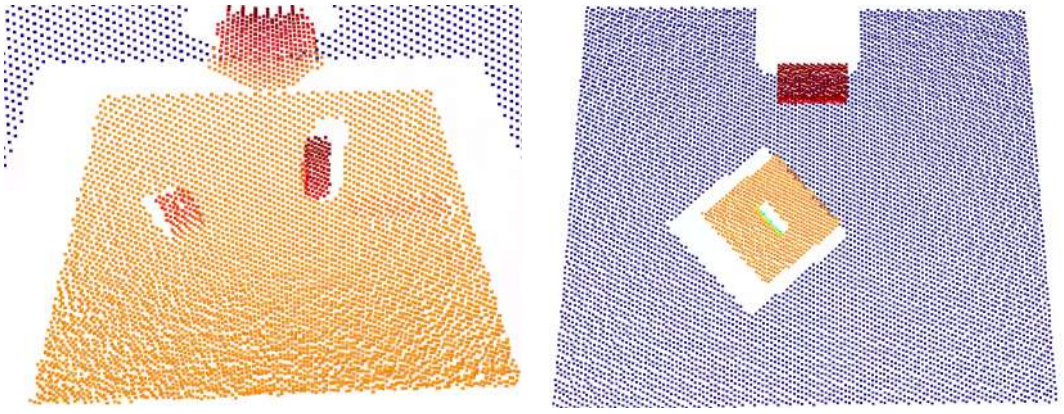

Figure 4. side-view and close-view

Result

Experiment setup

For the experiment setup. The hole and obstacles will be at random places above the workspace, the peg will be at random position. And we will make sure the hole and obstacles in front of the peg. The distance between peg and hole is greater than 40 cm.

Experiment result

1. side-view prediction

	Average error
Translation (mm)	6.5325
Rotation (degress)	X

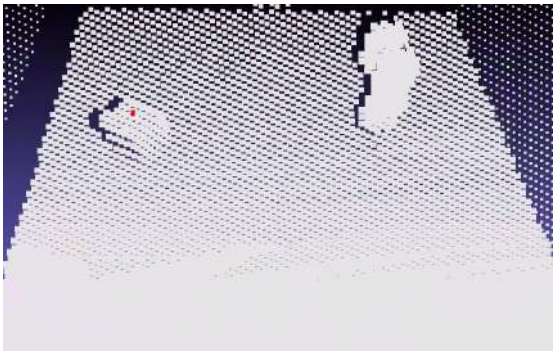

Figure 5. side-view hole position prediction (red dot)

2. Close-view prediction

without ICP	Average error
Translation (mm)	3.1254
Rotation (degress)	5.2435

with ICP	Average error
Translation (mm)	1.5828
Rotation (degress)	1.2284

As the following table shows. The position error on side-view predictions are a little large (6.5325 mm), and close-views get more accurate predictions on both position and rotation. In addition, ICP did help to reduce the error.

Conclusion

In this project, we build a pipeline for peg insertion, which combines the hole pose estimation and the collision-free trajectory planning as demonstrated in the simulator. To deal with the flexibility and reducing sim2real gap, we add some noise on depth image and add a side camera to achieve long distance planning.

Contribution

We have equal contribution for

1. coding
2. report
3. presentation

Libraries and open sources

CFVS: https://github.com/luben3485/CFVS?fbclid=IwAR3RkjYBOHqbM40DGTHFBvev4hj5ZTTnuhZfmfX%5B%E2%80%A6%5Dc3-MRi-tFbwSOYG_NxFxiJ5C2_JIf73jbEaU5AbbUEgzCsThjSdCDR9M7-RWJUY
ICP: https://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html
cuRobo: https://github.com/NVlabs/curobo

Reference

[1] Lu, Bo-Siang, et al. "CVFS: Coarse-to-fine visual servoing for 6-dof object-agnostic peg-in-hole assembly." *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[2] Sundaralingam, Balakumar, et al. "cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation." *arXiv preprint arXiv:2310.17274* (2023).

[3] Charles R. Qi, Li Yi, Hao Su, & Leonidas J. Guibas. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.

[4] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, & Gavriel State. (2021). Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning.

[5] Zhang, J., Yao, Y., & Deng, B. (2021). Fast and Robust Iterative Closest Point. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1–1.