Robot Grasping

Hongyang Yu
Gang Qiao

1 Introduction

In this assignment, we are required to be to design an algorithm for predicting good grasping approach configurations for a robotic end effector using machine learning techniques and working directly from pointcloud data. And we are give you pointcloud data of 3 cluttered scenes containing 5 individual objects on a tabletop. The main steps are as follows:

STEP 1: Generate initial hand configurations. Generate a large number of random initial hand configurations from each image to run them through the simulator.
STEP 2: Run simulator to get final hand configurations. Run the simulator whose input is initial configurations and RGBD images, and output is final hand configurations.
STEP 3: Get scores by scoring function. Use scoring function to get scores according to final hand configurations and RGBD images that input to simulator.
STEP 4: Extract features. Extract features on RGBD images and initial hand configurations.
STEP 5: Train Neural Network. Input extracted features and corresponding scores as labels to train the Neural Network.
STEP 6: Predict score. After finishing training the neural network, predict score from new test initial configurations and RGBD images.

The flowing chart of the project is shown in Figure 1.

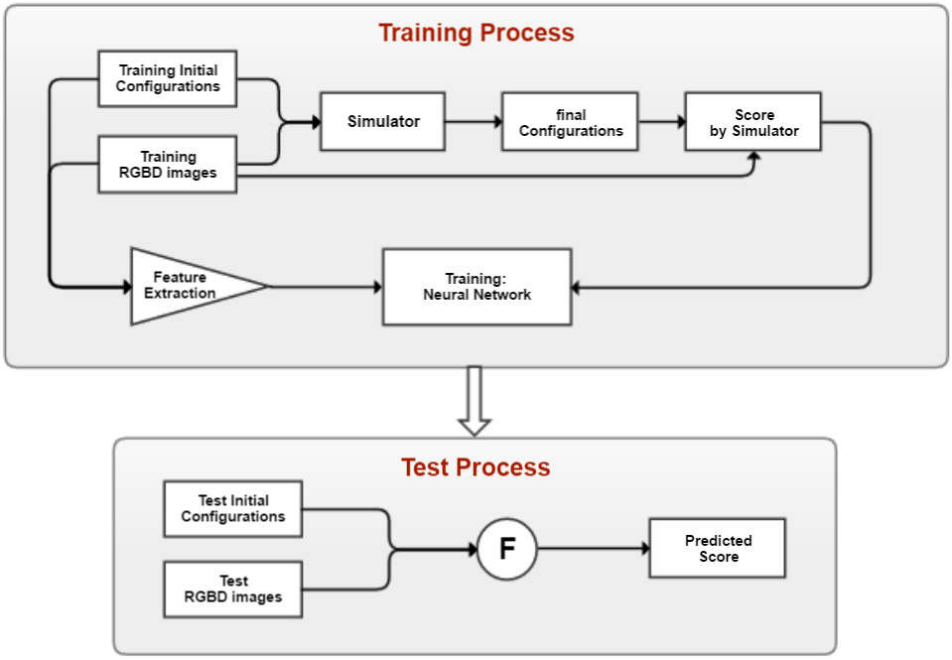


Figure 1: Flow chart of training and test process

2 Initial Configurations Generating

The basic idea of initial configurations generating is to

Firstly, we define a cube, whose center is the target object location and side length is L, as the initial hand configurations space. We scan each point with η step within the cube space as candidate point, and limited the distance d between candidate point and target object location. We choose points whose distances d's are between (l_1, l_2) as the location of hand. That is, we generate initial hand configurations on the surface of a nearly sphere whose radius is d. Then choose 10 quaternions to change the orientation of hand. Finally, we generate 1630 initial hand configurations for each scene. In our project, we choose $L = 0.6$, $\eta = 0.002$, $l_1 = 0.2$, $l_2 = 0.21$.

3 Scoring Function

3.1 Final Scoring Function

In this part, we come up with a scoring function to get scores. The input of our scoring function is the final hand configurations and RGBD images that input into the simulator. The basic idea of our scoring function is as follows:

1. Form a tetrahedron by the end point of three figures (points *A, B, C*) and the end point of palm (point *O*);
2. Choose each point Q_k within the tetrahedron *O-ABC* and project $\overrightarrow{PQ_k}$ to $\overrightarrow{OP}$, where P is the center of tetrahedron. Sum the dot project of each point Q_k. The equation is

$$S = -\frac{\sum_{k=1}^{N} \overrightarrow{OP} \cdot \overrightarrow{PQ_k}}{N} \tag{1}$$

A problem we have had is how to determine if a point is within a tetrahedron. We solve the problem by http://steve.hollasch.net/cgindex/geometry/ptintet.html. We use five determinants D0, D1, D2, D3 and D4 as the link above. Then we define that the score is 0 if the number of points within the tetrahedron is less than 800. Otherwise, the score get by equation 1.

A large score S means the hand holds the object close to the palm or figures are close to each other. That is, the hand can hold object tightly. Figure 2 shows the schematic diagram of the three pyramid. And figure 3 shows the result of our final scoring function. The blue part is the three pyramid.

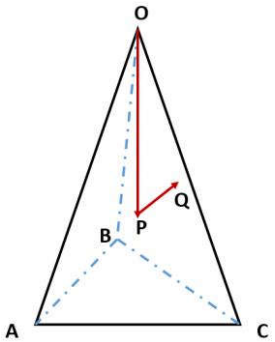
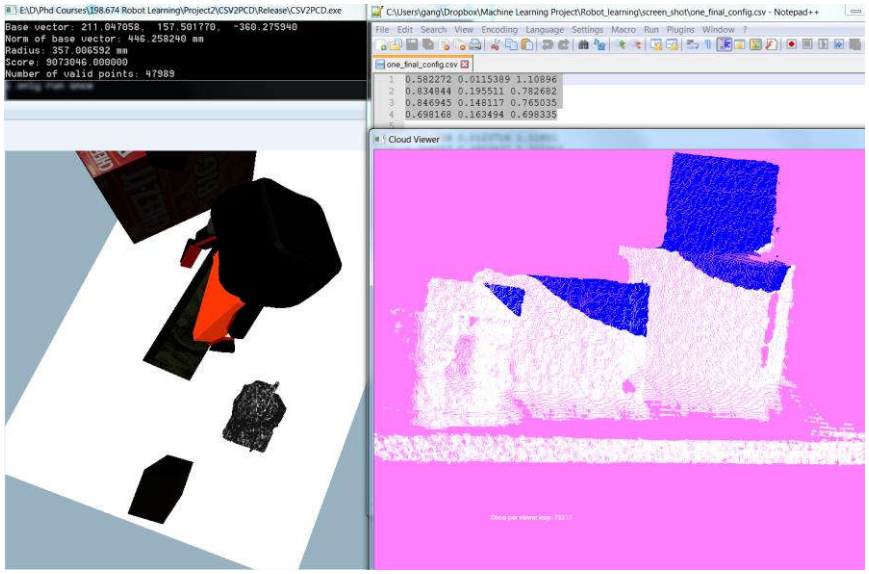


Figure 2: Schematic diagram of the tetrahedron

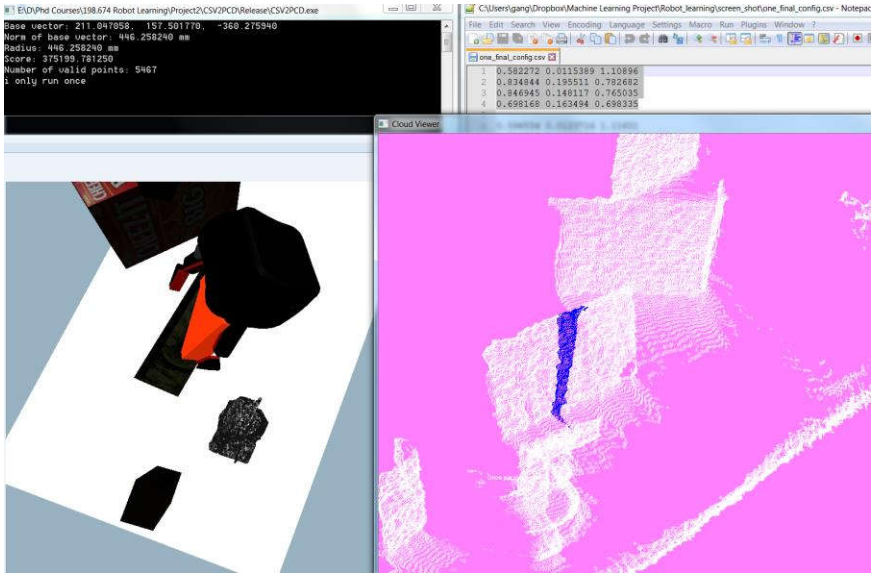
Figure 3: Result of our final scoring function

3.2 Scoring Function We Tried

Firstly, we read the paper “Efficient Optimization for Autonomous Robotic Manipulation of Natural Objects”. But the function is too complex. Then we decided to scoring function by ourselves. The basic idea of our first score function is to form a sphere whose center is the center of the three pyramid, and whose radius is the distance between the center and the end point of palm, and then count the number of points with the sphere. More points within the sphere implied a higher score. But when we visualized the sphere in the pointcloud, we found that it doesn’t work well. Then we come up with the method in 4.1. We compare the result of score function in 4.1 with that of score function in 4.2. The result is shown in Figure 4.



(a) Scoring Function we tried first



(b) Scoring function we finally used

Figure 4: Comparison of functions we realized

4 Feature Extraction

The coordinates of points in World Coordinate System (WCS) are translated into a local coordinate system attached with the hand, as if there was a camera viewing the target along z axis. Hence those points are projected back to the virtual camera, and cropped to the area of which the hand will touch. Max pooling over 26*29 grid is used to reduce the noise in that the hand will stop moving once touch the nearest points. Those points are then resized to 32*32 to feed to the learning function.

5 Learning Function

5.1 Convolutional Neural Network (CNN)

This is the learning function we finally used to do regression. We use RELU as neurons. We use stochastic gradient descent to train the CNN, and use maxpooling and dropout to avoid overfitting. Our loss function is Mean Square Error. The last layer is a linear neuron. The result is show in Figure 5. We do not good when score = 0 because we generated some initial configurations which will touch certain obstacles in simulator but can be cross from pointcloud, especially when the initial configurations is under the table.

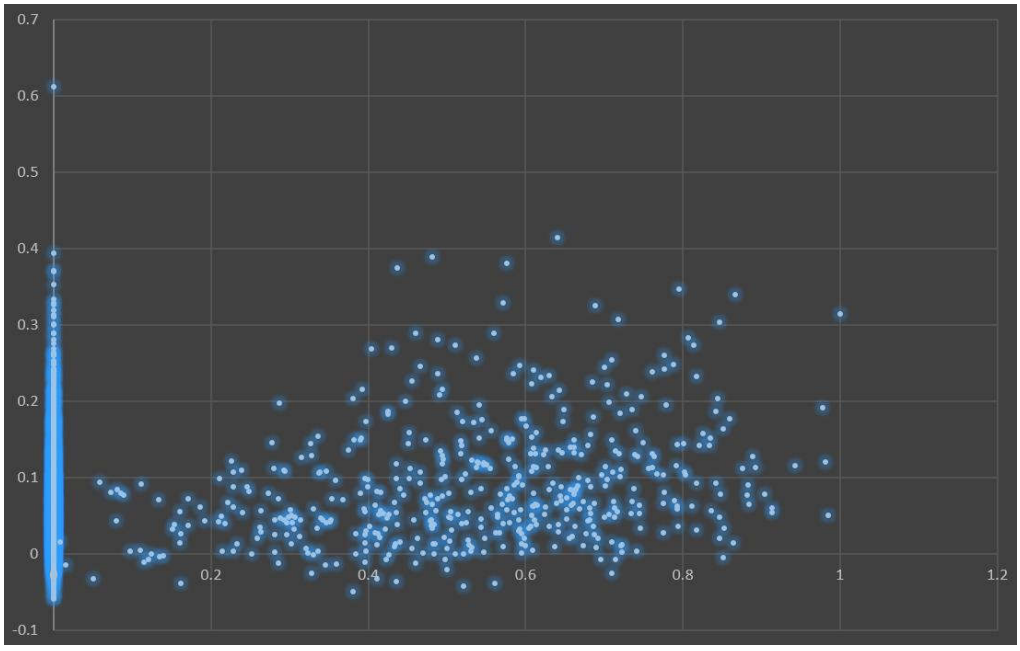


Figure 5. Result of CNN

5.2 Support Vector Regression (SVR)

This is the learning function we tried. We use scene 1 and scene 2 to train, and scene 3 to validate. The result is shown in Figure 6. The x-axis is the score we predicted, while the y-axis is the ground truth score.

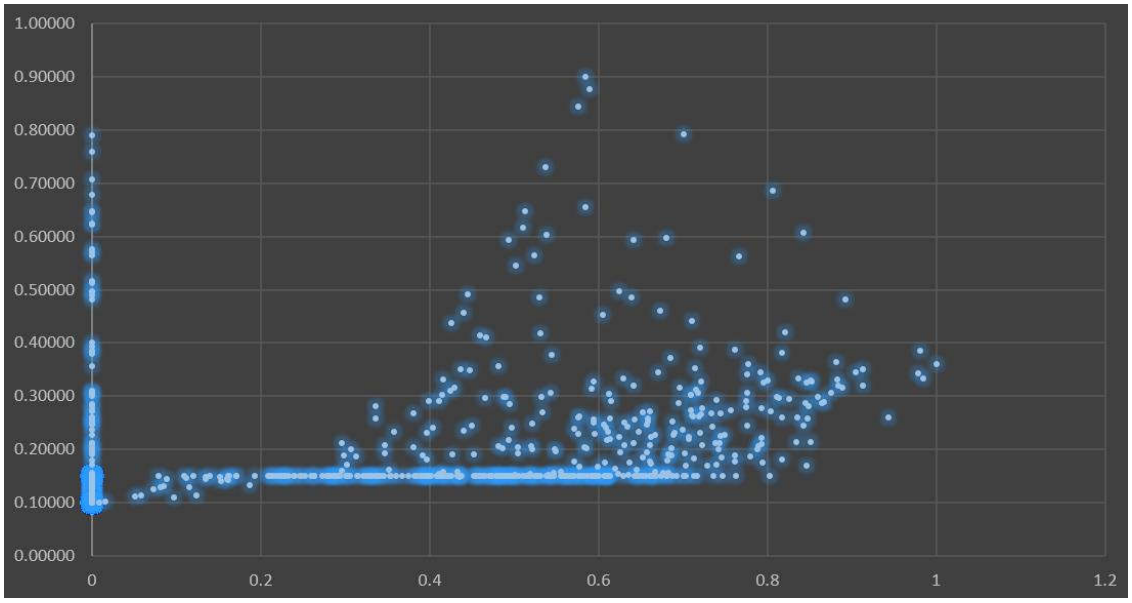


Figure 6. Result of SVR

Acknowledge: In this project, we discuss with Zhen Lyu and Yicheng. There might be some ideas we shared.