A Rubik’s Cube Solution Using Deep Learning and Robotic Arm Coordination

Group 5-11
Advisor: JENN-JIER JAMES LIEN
Members: HSING-YU LIN, YEN-SHUN LEE

0. Outline

1.1 Introduction: Motivation and Objective

- Motivation

Due to the rapid development of human-robot collaboration in recent years, we are now able to utilize robotic arms for more human-centered applications. Therefore, we aim to guide individuals in solving a Rubik's Cube through the integration of robotic arms, image recognition systems, and algorithms designed for solving the cube.

- Objective

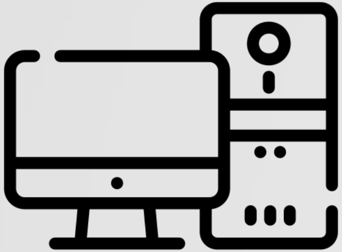
We are developing a Rubik's Cube collaboration system that integrates various software and hardware technologies to achieve human-robot interaction for solving the cube.

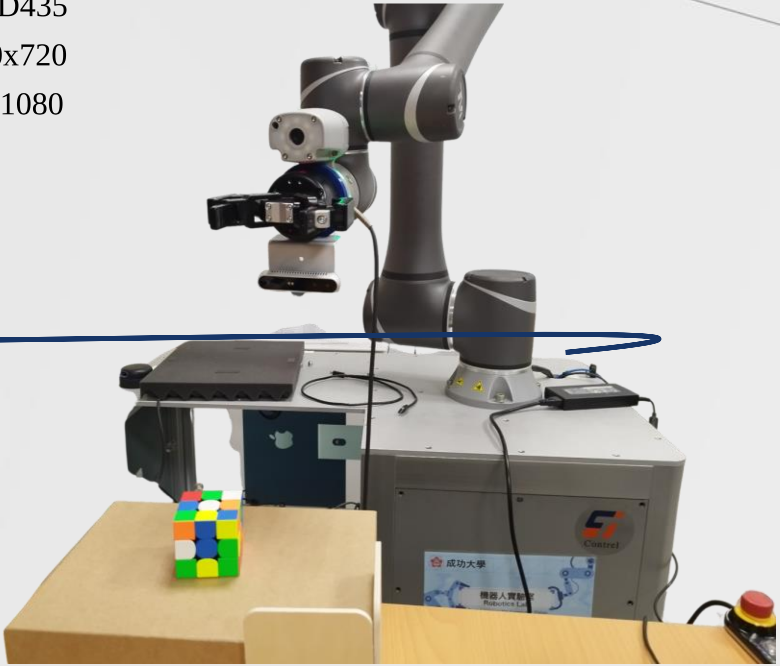
1.Sensors: RGB-D Depth Camera Realsense D435
2. Robotic Arm: TM Robot
3.Image Processing: OpenCV
4.Algorithm: Kociemba's two-phase algorithm
5.Deep Learning: PointNetGPD

2.1 Hardware : System Setup

Sensor: Obtain the position and orientation of a Rubik's Cube.

- Intel® RealSense™ Depth Camera D435
- Depth output resolution: Up to 1280x720
- RGB frame resolution: Up to 1920x1080





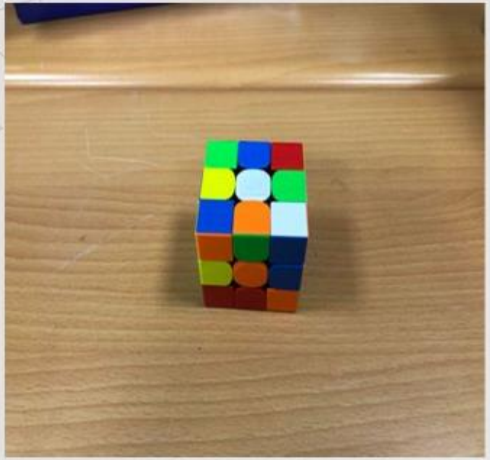
Computer: Visualize and Solve the Rubik's Cube

- Ubuntu 20.04.6 LTS
- Intel® Core™ i7-10750H CPU @ 2.60GHz × 12
- NV166 / Mesa Intel® UHD Graphics (CML GT2)

3.1.1 System Framework

1

Receive image from sensor



2

Grasp prediction

PointNetGPD model

Get the coordinate of the grasp pose

3

Scan and solving

1) Scan the Rubik's Cube
2) Make solution through Herbert Kociemba's two-phase algorithm (if the pattern is wrong, make new solution)
3) User enters rotation number
4) Rotate rubik's cube

4

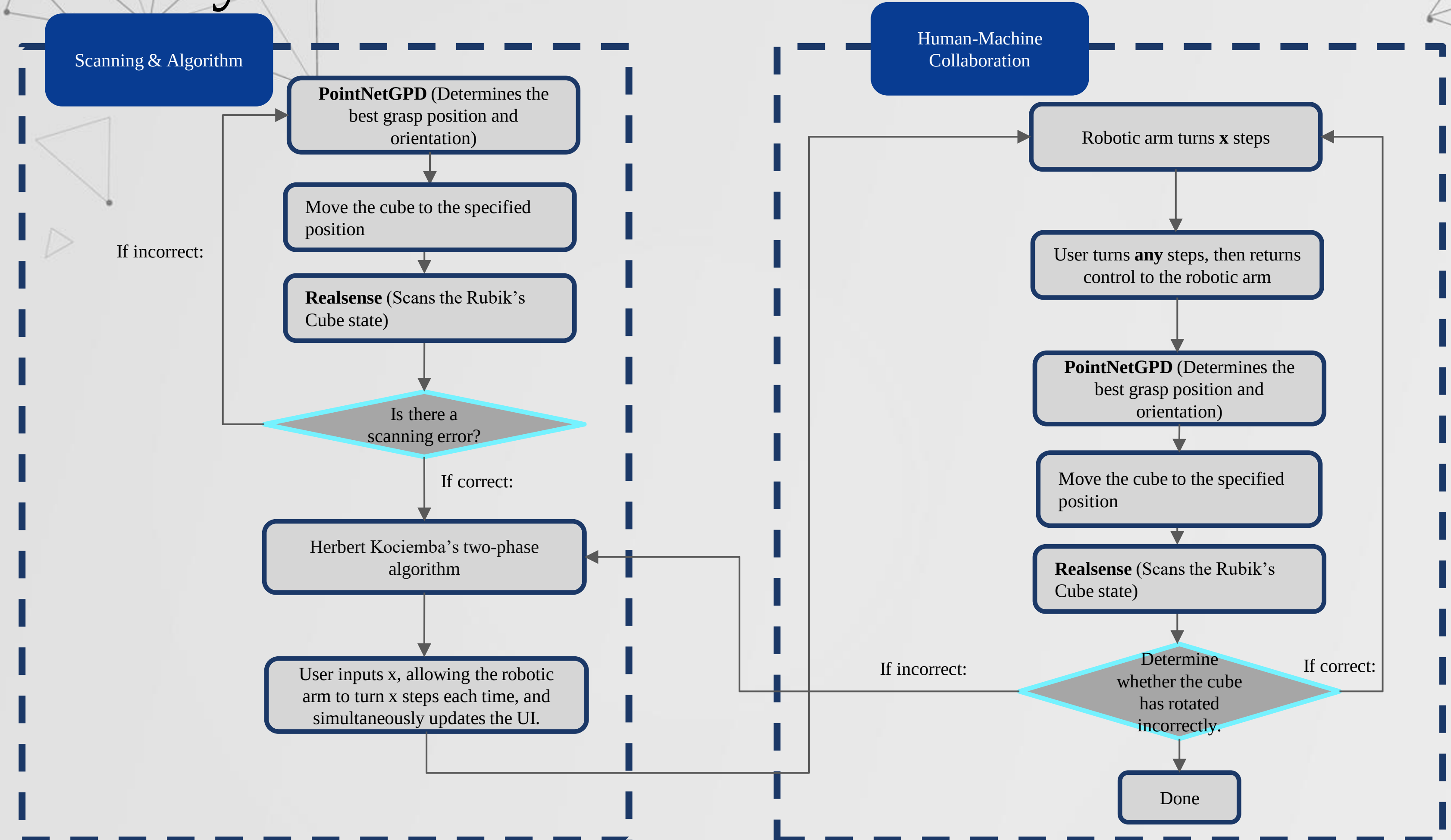
Humanity's Turn

1.Robot moves the rubik's cube to the specified position
2. Robot return to home state and continue to detect rubik's cube

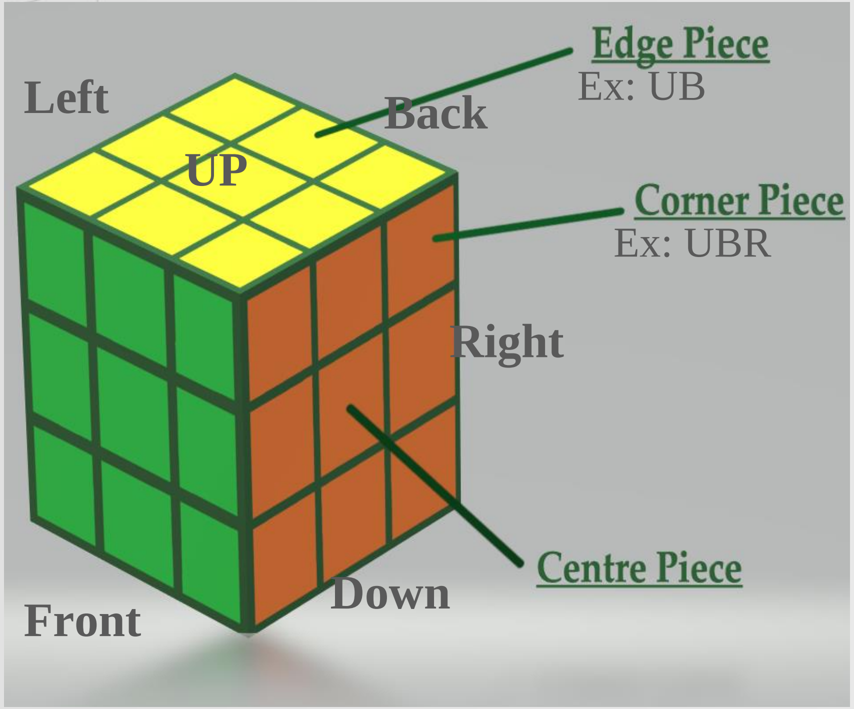
5

The user manipulates the Rubik's Cube as desired and then returns it to the robotic arm for the next iteration if the Rubik's Cube is not solved, or concludes the process if the Rubik's Cube is solved

3.1.2 System Workflow



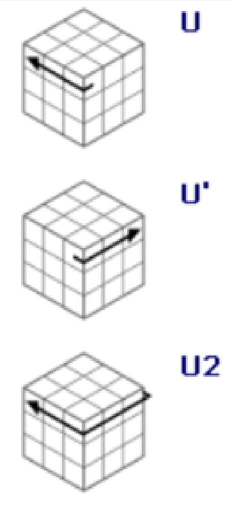
3.2.1 Rubik's cube



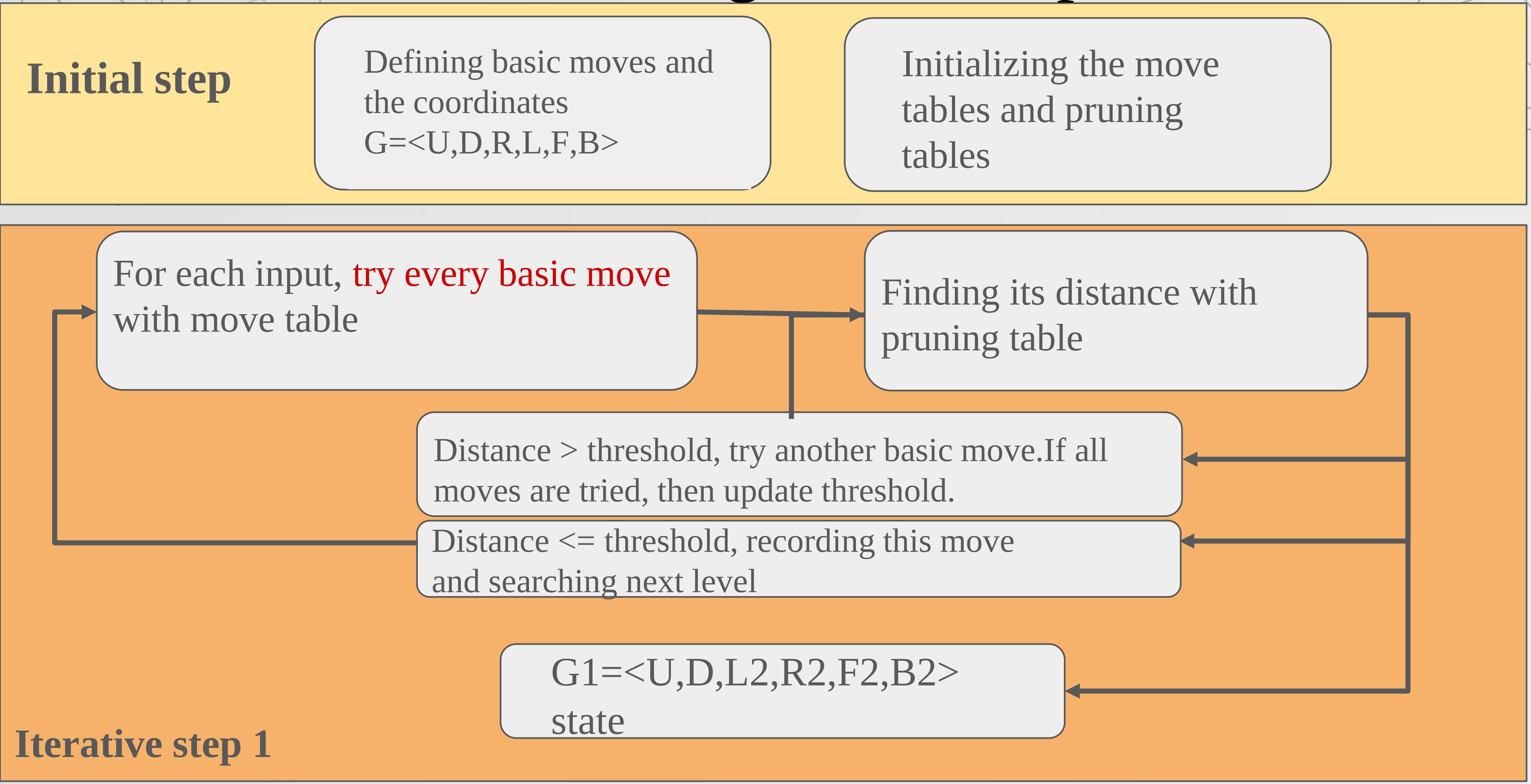
Signmaster Notation & Moves

F = Front   R = Right
B = Back   U = Up
L = Left   D = Down

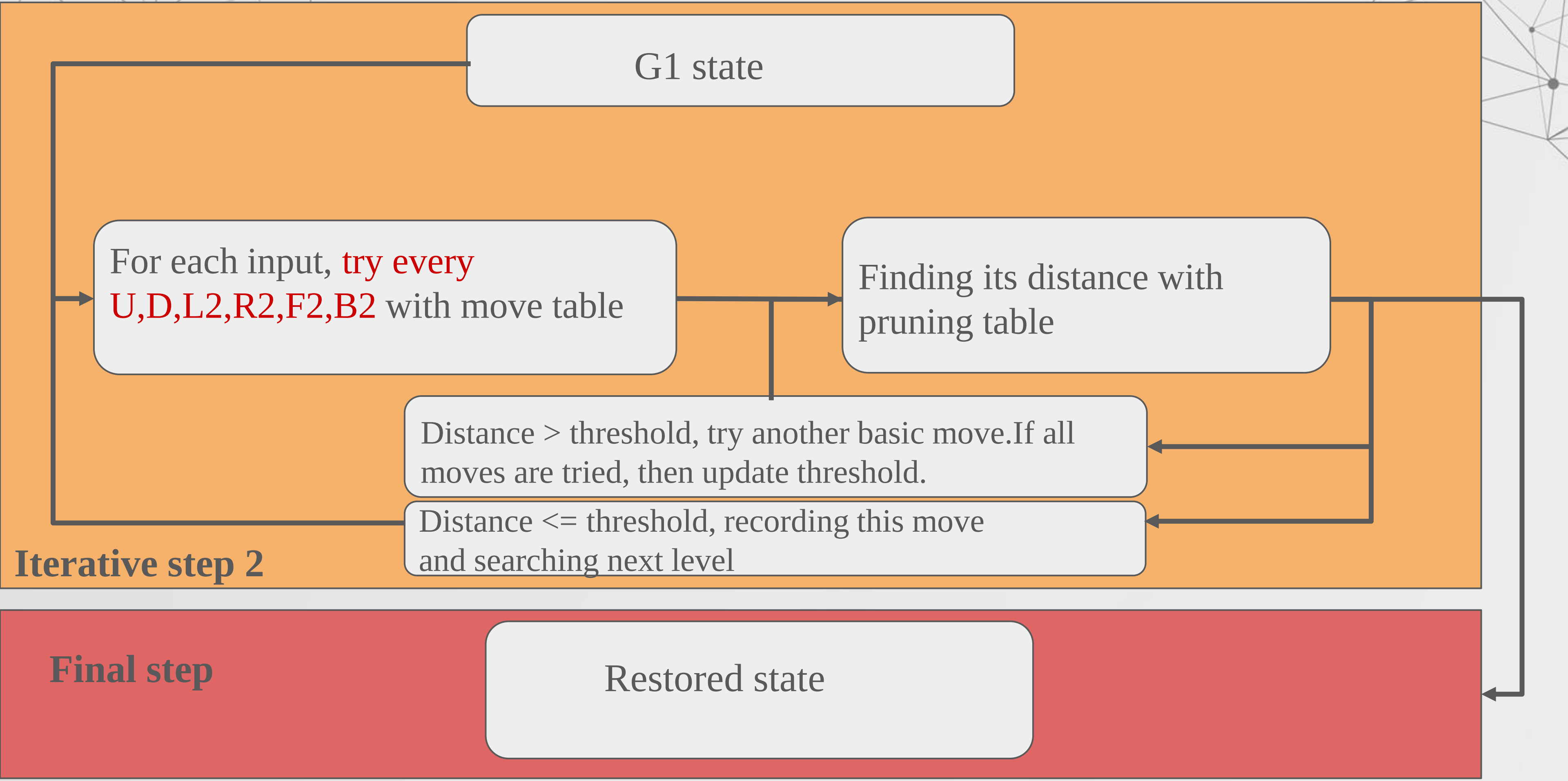
Ex: U moves



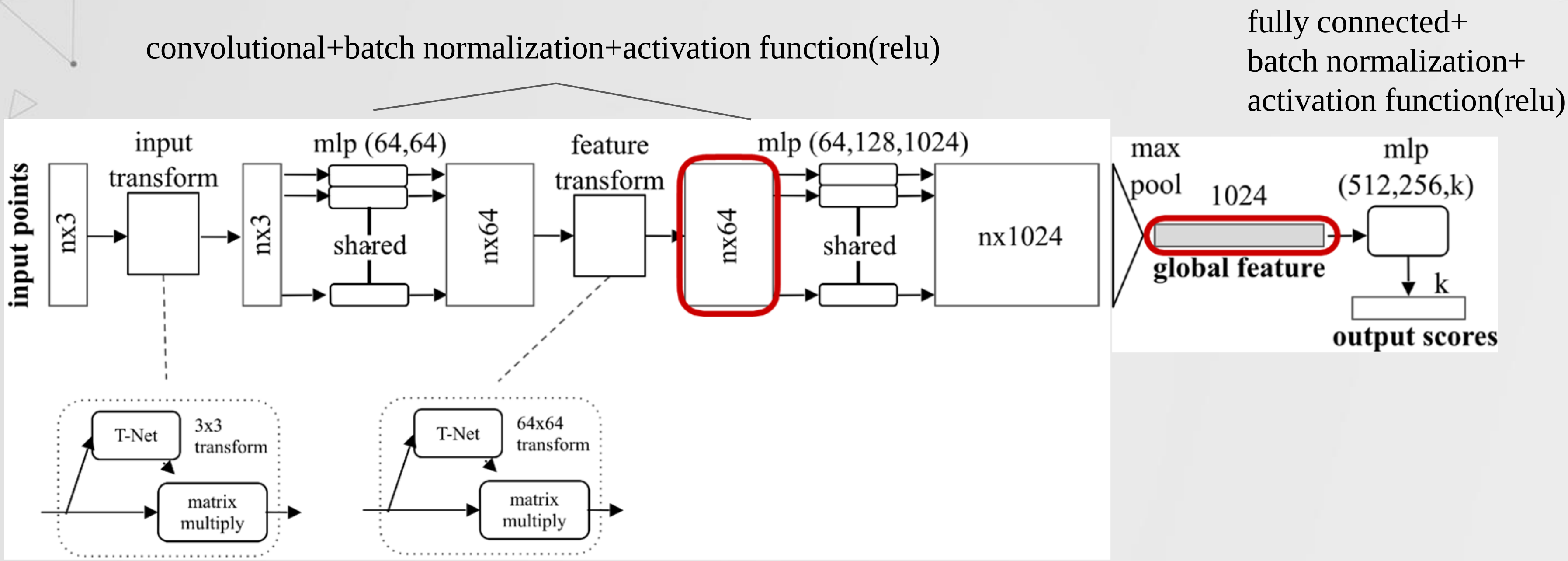
3.2.2 The Two-Phase-Algorithm-Implementation



3.2.2 The Two-Phase-Algorithm-Implementation

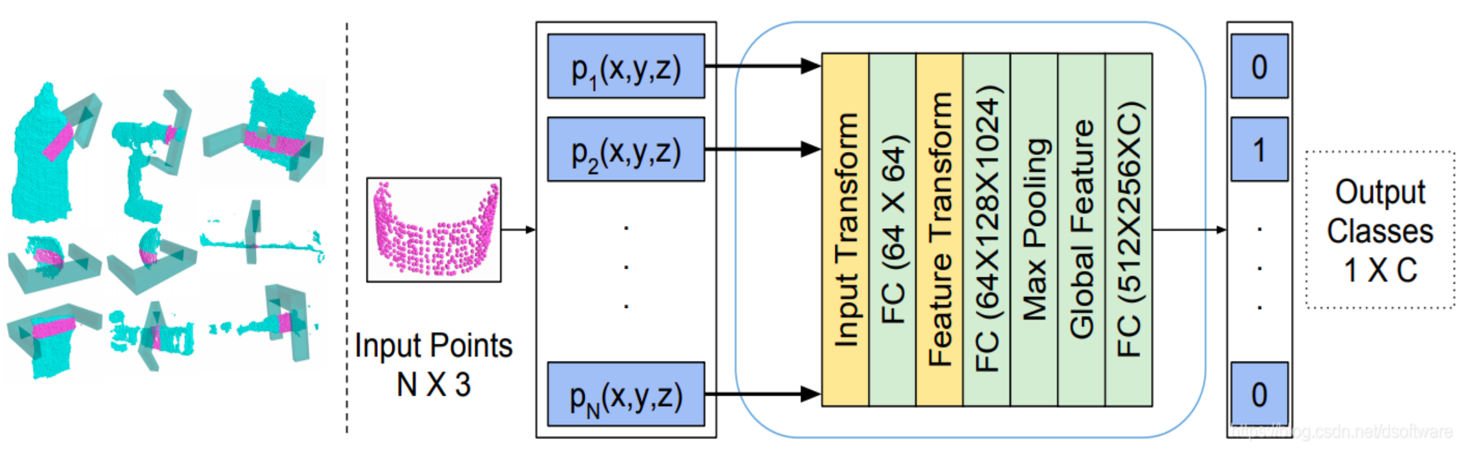


3.3.1 Pointnet(Classification)-Concept

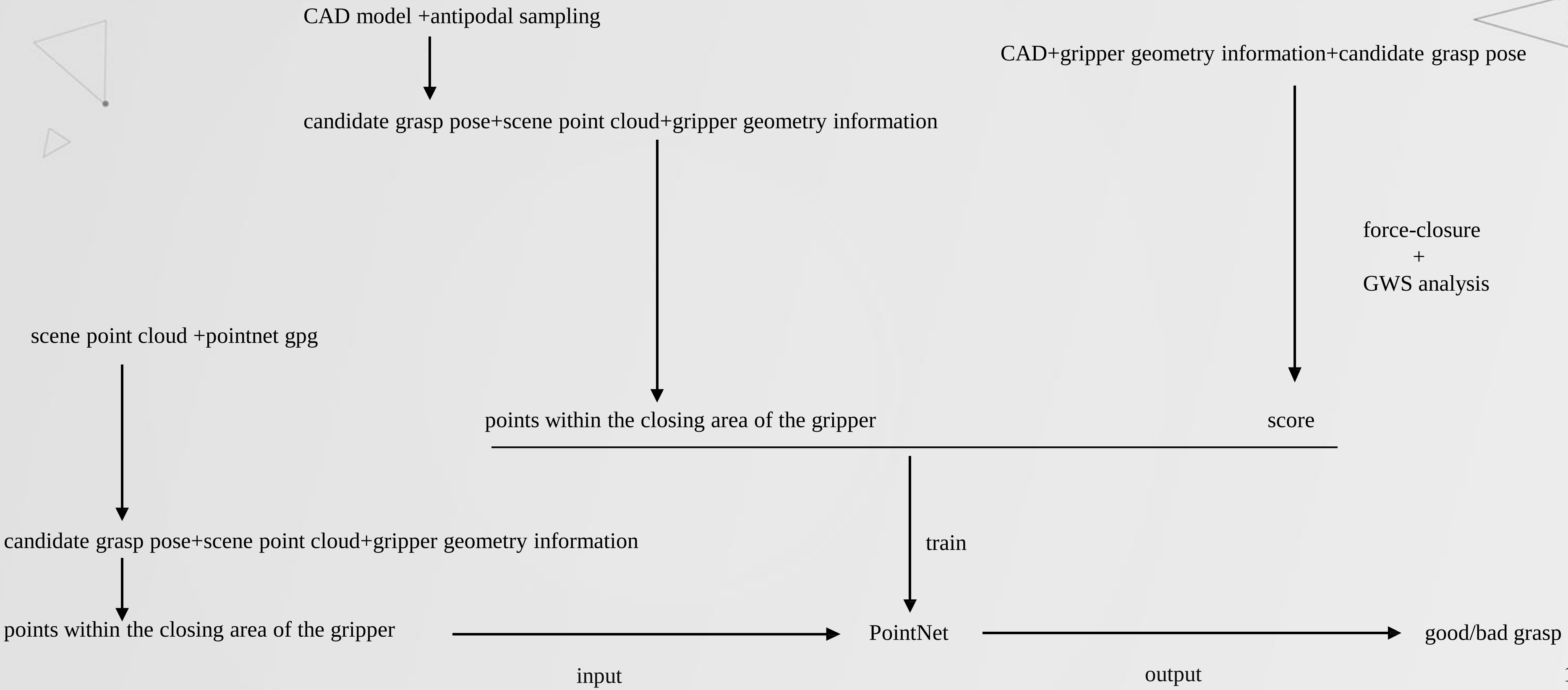


3.3.2 PointnetGPD-Concept

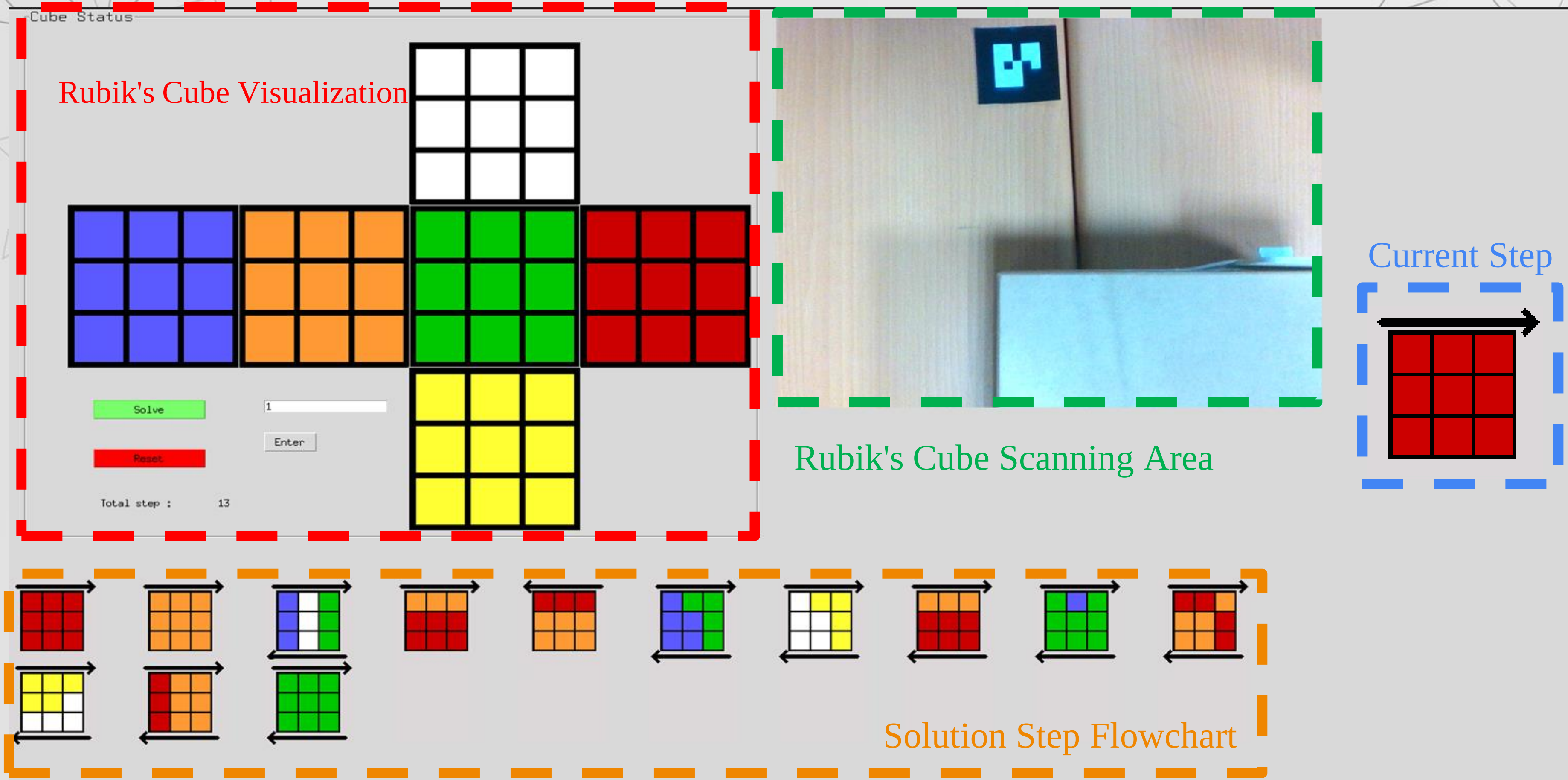
step1 : Taking raw sensor input from a common RGB-D camera, convert the depth map into a point cloud
step2 : several grasp candidates will be sampled with essential geometry information as constraints.
step3 : For each candidate, the point cloud within the gripper will be cropped and transformed into local gripper coordinate and finally fed into our grasp quality evaluation network.
step4 :The grasp with the highest probability will be executed.



3.3.3 PointnetGPD-Implementation



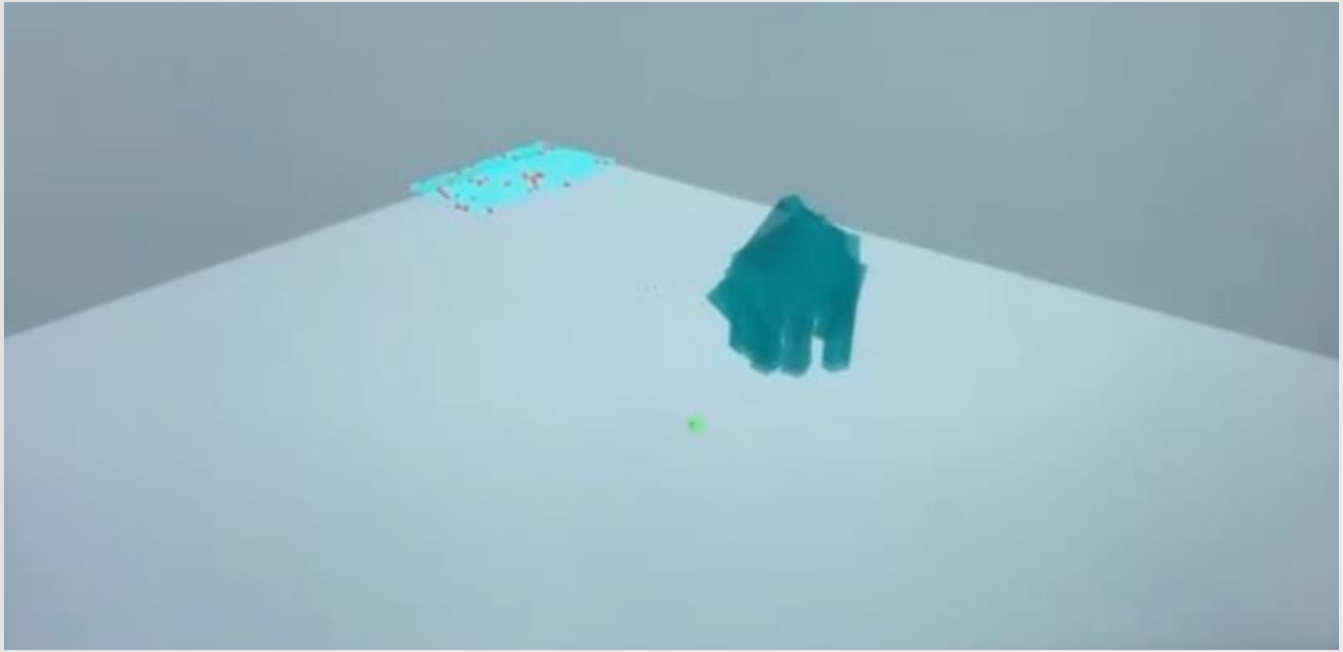
3.4 User Interface



4.1 Experimental Results

Grasping Pose Candidates Produced by PointNetGPD:

The image below displays the predicted good grasping pose candidates generated by the PointNetGPD model. Since multiple good candidates may exist, overlapping instances can occur in the image. The model selects the candidate with the highest score as the final output.



5 Reference

1. Kociemba's two-phase algorithm

http://kociemba.org/cube.htm

https://github.com/hkociemba/RubiksCube-TwophaseSolver

2. Pointnet

https://arxiv.org/pdf/1612.00593

https://github.com/charlesq34/pointnet

3. Pointnetgpd

https://arxiv.org/pdf/1809.06267

https://github.com/lianghongzhuo/PointNetGPD