ROMS (TASK1) – Yohanes de Britto Hertyasta

Task 1. In this task we would like to hear your hypothesis (rough planning) on a part of the robot picking systems pipeline.
/TechnicalAssignment_E
Problem definition: In Robot Picking systems the goal is to be able to pick various objects and place them in predefined places. Since those items are in boxes, you must avoid collisions with the boxes when picking. To do this, we need to recognize the three-dimensional position and orientation of the box and know where the walls of the box are located. We assume that we know in advance the intrinsic parameters of the camera capturing the target boxes and products, and the CAD models of the boxes are given. It is also assumed that an RGB image and a depth image (aligned with the color image) are acquired at each pick.

Q.1) Knowledge: Based on the above assumptions, how would you generate the point cloud? If using Python's Open3D, which function would you execute?

A.1)
Assuming the camera position does not change, I will go through the camera's point cloud settings and adjust the max height, width and depth to minimize unnecessary data from the get-go, therefore cutting down our preprocessing time.

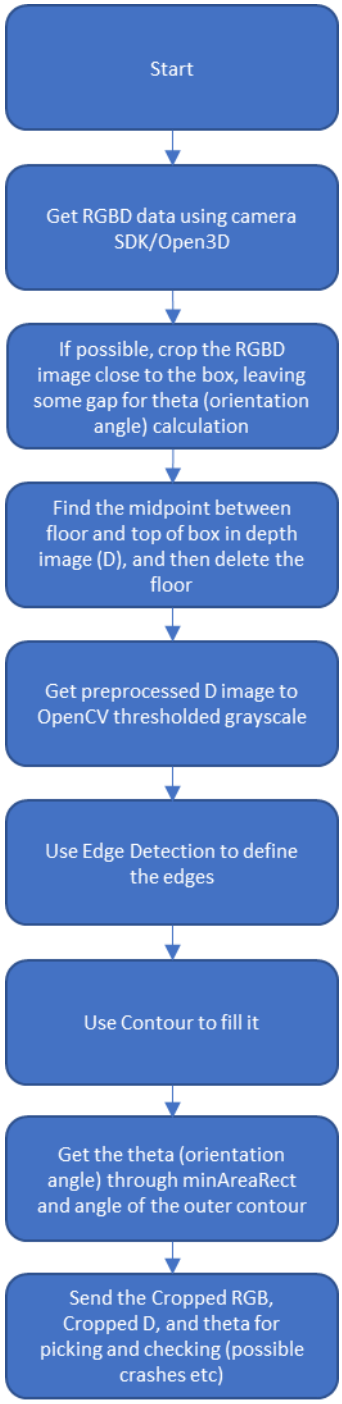
Specifically, if we are to use Python's Open3D, assuming that we also use RealSense, I would use *t.io.RSBagReader*. To generate point cloud, we would then use *geometry.PointCloud.create_from_depth_image* for the output of the previous part. The settings could be changed in the accompanying JSON file. The point cloud itself would then be converted to NumPy array for processing, especially if the next pipeline of work dictates the necessity of using organized point cloud, which involves another set of steps altogether.

Q.2) Design: How would you estimate the 3D position and orientation of the target box? Please propose the following two approaches using a brief flowchart, considering that products can be fulfilled in the box. But it should not be beyond the top of the box. It's enough that you can share concepts of each approach.

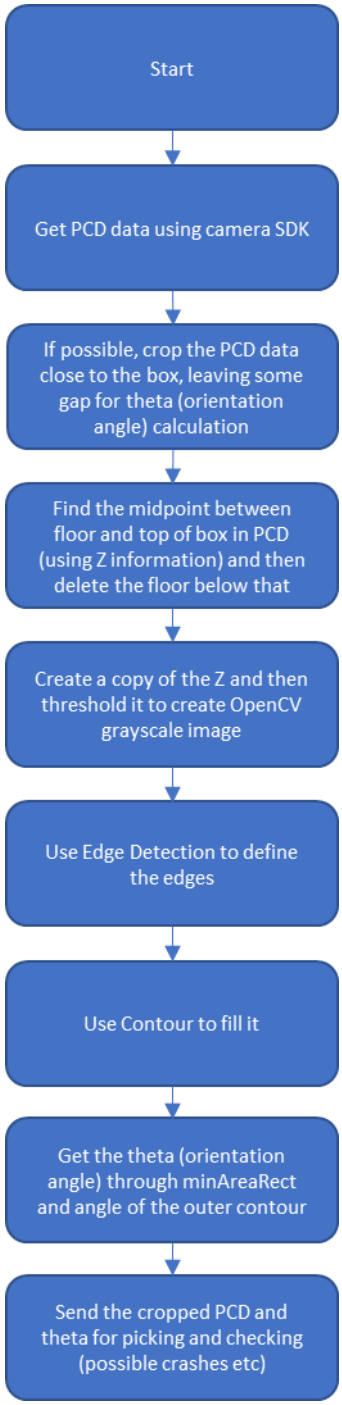
A.2)

Assuming that we could get good data without preprocessing (proper RGBD situation, organized point cloud reflective of the actual position), below are the methods I propose and used before:

a) From RGB and Depth image



b) From Point cloud



Q.3) Assessment and Decision: Please explain the Pros/Cons of the above two approaches. In addition, please describe how you would prioritize the development of the two approaches presented in Q.2)? Please also explain why:

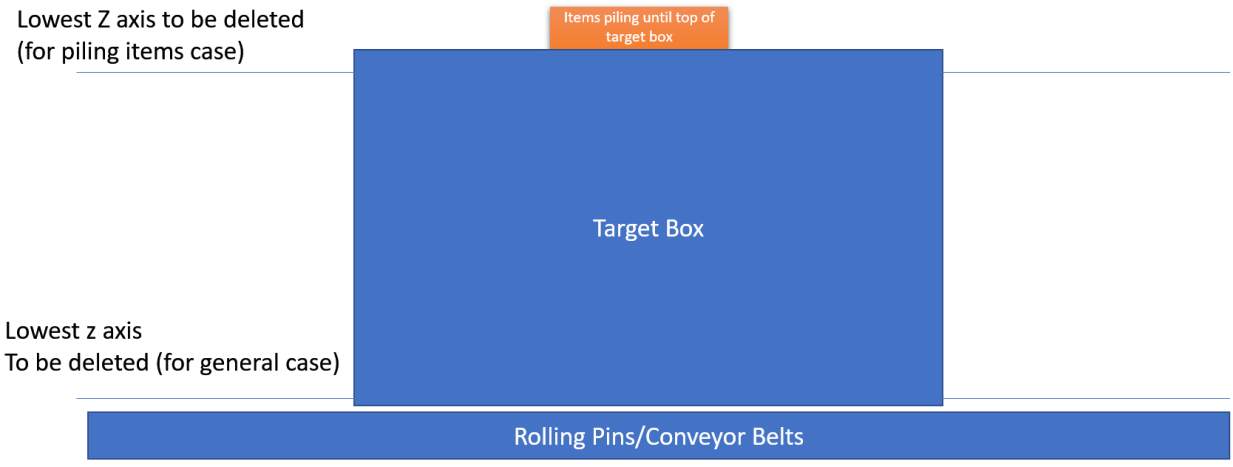
A.3)

Based on my own experience, both approaches are similar (converting every information to 2D at the end), therefore in terms of pros, which is an advantage compared to the other's cons, it all depends on the specifications (whether it stipulates the use of RGB-D only or point cloud is possible). Because I am more experienced with point cloud, I would lean towards point cloud.

Using RGB-D will provide more information on the RGB side, which is good for visualization, but as a double-edged sword, it might introduce many noises, especially if we're talking about the lighting and time of day of the area. This is also why in my personal experience, I always in the end never really focus on the RGB information, but always lean towards the depth information instead. The pros of this method is the initial speed, but point cloud has more potential information that might be beneficial for processing later (and frankly speaking, easier to preprocess).

Using point cloud, the initial speed compared to RGB-D left more to be desired (especially with Open3D and its extra steps, in relation to Q1), even more if we are to convert the point cloud into its organized form. However, because this step is only done once, it will not be an issue. And in order to speed up things, at the beginning step upon getting the point cloud, I usually imposed a height threshold to cut the lower half of the point cloud (usually floor of box, as seen in the material picture), and in turn, everything on/below that height like the conveyor outside of the box and the floor), which is harder to do when working with simply RGB-D image. If we want to go even more, we can extend the deletion threshold to near the top of the objects, therefore, with objects filling up to the target box, it is easier to delete the noises. With less redundant information to work on, we can cut possible time for actual processing.

Below is the example of two deletion of the point cloud lower half.



Based on that information, as stated before, I would lean towards using point cloud because despite its initial speed, its potential information and ease of implementation for later steps outweighs the cons.

ROMS (TASK2) – Yohanes de Britto Hertyasta

Task 2. Object (car and pedestrian) detection.
Test dataset: Test dataset: 100 images containing cars and pedestrians. Please feel free to use any images you could find online (ref. open images).
Annotation: You can either find the test dataset together with their annotations or you can simply annotate by yourself.
Model:
Take any appropriate model you could find online. Please, do not spend time on training (or fine-tuning) the model. The objective of this task is not the model accuracy. The objective is to assess the evaluation and deployment ability.

Q.1) Evaluation:
Given a model and the test dataset (100 images), evaluate the model performance in terms of detection performance (please decide on the metrics by yourself) and the inference time.

A.1)
Dataset used: https://data.nvision2.eecs.yorku.ca/JAAD_dataset/ (pruned to 100 images per task description).
I took 1 random frame (RNG seed: 39914030) from each video file, and then take random captured frames (RNG seed: 49911062) amounting to 100 pictures for this test dataset.

The model I used is MobileNetV2, trained using COCO dataset. It is not a specialized model, but it is capable, to some extent, to detect human and cars across different dataset (even though accuracy is quite questionable).

Detection performance, if the metrics are by "all detected objects", which means foregoing any undetected (even though it is possible for the undetected objects to be the target, and), the performance of the model is quite good for detecting cars and pedestrians. It is, however, facing difficulties in detecting obscured pedestrians. To maximize pedestrian detection, I used 0.55(55%) confidence threshold and 0.7 IoU threshold.

Q.2) Assessment and Decision:
Usually, the developed AI pipeline should be modified considering the business requirements. Let's assume that the business team has two requirements:

1. Not to miss any pedestrians and to have highest possible precision in detecting cars.
2. Detecting the pedestrians as accurately as possible, i.e. the predicted bounding box should fit the ground truth bounding box as much as possible (>0.8 IoU). For "cars", however, rough detection/localization should be fine (0.5 IoU).

Considering the business requirements, please decide on the most appropriate confidence threshold and IoU threshold for each object (4 thresholds in total). After deciding on the thresholds, please report the corresponding precision and recall values for each object (cars and pedestrians).

A.2)
Referring to A.1 above, to maximize the accuracy of detection for pedestrians, 0.55 is the appropriate one with 0.7 IoU. For cars, in this model, I used the same variables, because upon several trial-and-errors, the results of using this model with this model is more car-heavy. Please note that because the

model is unoptimized and untuned, requirement no.1 of not missing any is way beyond the scope for now.
For precision and recall, I only sampled the first 10 images due to work time constraints. Below are the results.
Therefore, below are the thresholds and precision and recall:

Pedestrian Confidence Threshold: 0.55
Pedestrian IoU Threshold: 0.7
Car Confidence Threshold: 0.55
Car IoU Threshold: 0.7

Pedestrian Precision: 0.875
Pedestrian Recall: 1.0
Car Precision: 0.913
Car Recall: 1.0

Q.3) Deployment: We use GitHub for software development, and Docker for the environment. Please share your software so that it meets the following requirements.
a) Please build an environment so that the inference program to a test image implemented in Q.2 can run on Docker. docker-compose is also acceptable.
b) Please clone the following repository and create a branch whose name is "yourname/feature-task2": https://github.com/roms-jp/TechnicalAssignment_J
c) Please share your Dockerfile and the programs needed to run them on GitHub so that we can run them, and include instructions for running them in the README.
d) Please create a Pull Request on GitHub from your branch to the master branch with instructions on which document we are supposed to follow to reproduce your environment.

A.3)
(Just in case I didn't get to push it in time, please use my clone repo.)

1. git clone https://github.com/aki-hp/TechnicalAssignment_J.git (or the pulled branch)
2. cd TechnicalAssignment_J
3. docker build -t ubunturomstask:v1 .
4. docker run ubunturomstask
   (In docker)
5. cd necessaryFiles
6. python3 testCocoPed.py

The results can be seen in imagesJAAD_annot folder.

7. (optional) cd imagesJAAD_annot

I tested this outside of Docker, so please copy the results outside of docker for seeing it all.