Recover the Real Size of Stop Sign

Chenyang Zhu, RI, CMU
Mentor: Christoph Mertz, RI, CMU

1. Introduction

In this project, we are trying to recover the real size of stop sign. Its accomplished based on the results of ORB-SLAM and other simple computer vision algorithms. In the process, we use the video captured by a calibrated camera (smart phone camera). ORB-SLAM will generate/reconstruct a 3D point cloud map of the scene and the correspondences between 3D map points and key-points on every frame. Then we can detect or handily select the stop sign in each frame, resulting a bounding box containing the stop sign. With a bounding box, its easily to seek out the key-points on stop sign and its corresponding 3D map points, which represent the position of stop sign in the point cloud map. Then we can compute the distance between the stop sign and camera center. After this, based on the focal length of the camera, we can recover the size of stop sign with a simple triangulation method. One point to mention is that the recovered size is correct up to a scale which can be computed by rescale the slam map with GPS data.

2. Recover the size

2.1 ORB-SLAM related changes

In this project, we will utilize ORB_SLAM to generate the 3D point cloud of the whole scene. In this process, it is highly important to achieve to a file containing all the points' position information and the alignments between 2D and 3D points. The original ORB-SLAM source code only saves a file which contains the key-frames. So, we need to refine the code and change some parts of the source code. The main difference is the added new function save-map-points. This function will save all the 3D map points and its corresponding 2D observations in frames. All the data will be saved to a file called SaveMappoints.txt.

2.2 Data processing and bounding box select

After collecting all the points information, we need to extract the frames where include the stop signs. Based on the key-frames, filter out the frames which both contain stop signs and are the key-frames. The number of qualified frames is usual one or two. Then based on the timestamp of the selected frame, we can handily select out the stop sign with a rectangle. In the future work, we may combine deep learning method to detect the stop sign.

Based on the rectangle of stop sign, we will have access to the 2D observed key points in the rectangle. These 2D points' corresponding 3D map points are the stop sign's position in the reconstructed 3D map. In the 3D map, with the position of camera center and the position of stop sign, we can easily compute the distance between the camera and the stop sign.

2.3 Edge detection and image smoothing

In this next step, we need to find the pixel size of a stop sign in the selected frame. In the detection, we don't have a strict requirement on the precision. We just detect the upside edge and the downside edges, because the vertical direction is affected less than the horizontal direction. In the implementation of Matlab code, we simply use the library provided function to detect the edge. In the processed image, which is black & white image, there are many noises, and most of them are caused by the characters 'STOP'. One efficient method is to remove these unusable edges. Actually, in the whole image, what we care is the two edges of the sign, and the two edges are long enough for to distinguish them with the other detected edge parts. So we can remove those noises just based on the pixel size of the detected edges. The following image shows the process.

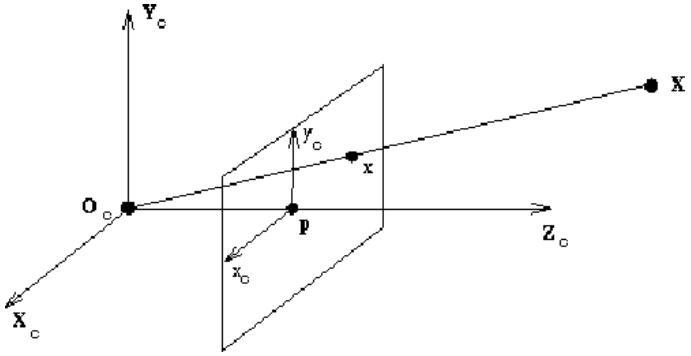
In this part, we need to pay attention to several tricky points. First of all, we need to make sure the sign is not crooked, if so, we need to some pre-processing. Secondly, we count the number of white pixels in the filtered image. If the number is larger than a threshold, we will count it as an edge.

2.4 Compute the size

Through the former 3 steps, we have access to the distance between camera center and the real stop sign in the scene, the pixel size of stop sign in frame and the focal length of the camera (calibrated camera). Then we can compute the real size of the stop sign with triangulation, which has been shown in the following image. The equation is,

$$real\ size = \frac{d_{gps}}{d_{slam}} * \frac{d_c * d_{pixel}}{f}$$

The computation is implemented by the above formula, and the definition of arguments included in the equation is described in the 4th part of this report, error contribution.

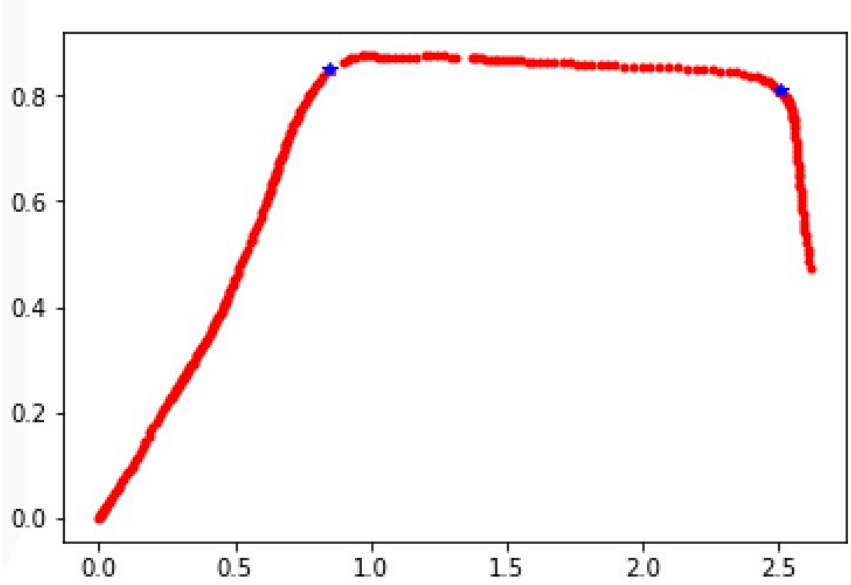


3. Rescale the map

In this part, we will implement the rescaling of the SLAM map. The map is created by ORB-SLAM. For comparison, we also build up another map from the GPS data. We are going to treat the GPS map as the ground truth for computing the real scale of SLAM map.

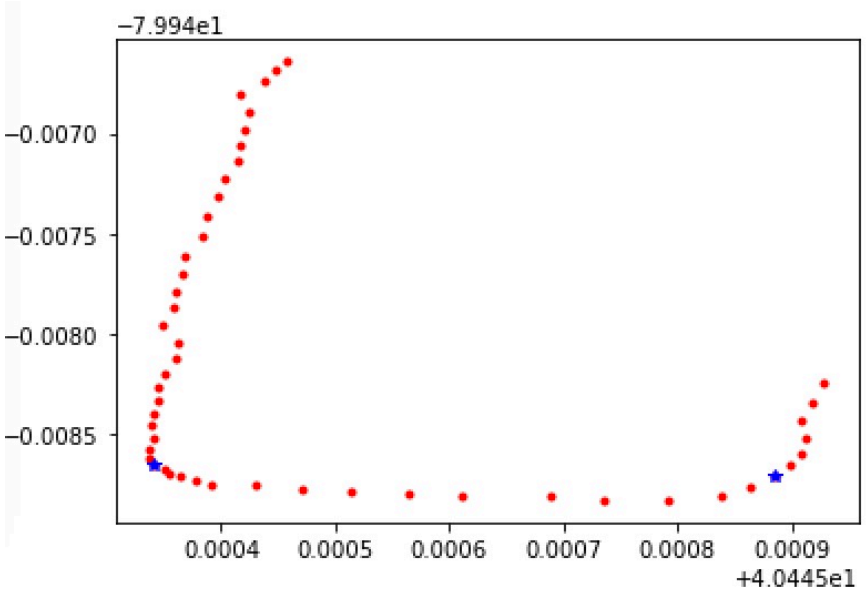
3.1 SLAM map generation

The ORB-SLAM project has been made open source on Github. In the newest version, ORB-SLAM2, the output result generated by the project is a text file containing all the key frames selected from all the video frames. For each key frame, the file reserve these information: frame number, timestamp, camera center position and son on. What we focus is the camera center. We need to extract the camera center positions and draw the map of them. This is the SLAM map.



3.2 GPS map

The smart phone is equipped on the car, so the GPS also represents the location of the car. Then we can get the GPS map, which should have a similar shape with the SLAM map. One reminder is that ORB-SLAM has drawbacks when the car is turning, which means the turning angle is hard to detect, but it is easy to recognize a straight path.



3.3 Maps comparison and rescale

In comparing these two maps, we generally select the two endpoints of a straight path by hands, because in this usage, we will only have a single map of a large district, which means we only need to rescale the map once. Therefore, execute this task by hands is not expensive. Of course, we can implement it by simple algorithm, like line fitting or cluster. It depends on personal choice.

4. Error Contribution

The several tests have been done showing us an error around 8% of the real size. The formula of computing the size is :

$$real\ size = \frac{d_{gps}}{d_{slam}} * \frac{d_c * d_{pixel}}{f}$$

where d_{gps} represents the real distance of a path on the map, d_{slam} representing the path length in slam map. $\frac{d_{gps}}{d_{slam}}$ is the computed scale of slam-map. d_c is the distance between camera center and stop sign in the 3D slam-map. d_{pixel} is the pixel level size of the stop sign in the frame. f is the focal length of smart phone's camera.

Based on the above equation, d_{pixel} comes from edge detection; d_c comes from 3D slam-map and f comes from the calibration information. All the three arguments promise a relatively high precision. While the other two parameters, d_{gps} and d_{slam}, are extracted from paths in the GPS and slam maps, which dominate in the error contribution. One reason is that the GPS map is not accurate. Some location points drift during the data collection process. In addition, the GPS map is a sparse map. In some situations, it's too sparse for users to select the correct ending points of the path. Another reason is about slam map. Slam-map is accurate if we only consider the map points representing the positions of the car. However, ORB-SLAM is not sensitive to the rotation of view angle, i.e. the rotation of cameras. So the map partition of the turnings contains high uncertainty, which also makes the selection of path's ending points tough. What we can't ignore is the scale of maps. In the experiments we've done, we only had access to a small scale map, which only covers several blocks. This makes the selection of path ending points even harder. A miss is as good as a mile. If we exploit a large scale map, the error of ending points selection could be ignored directly. To some degree, the affections of maps' precision problems can also be reduced.

5. Summary

All the descriptions above have introduced and explain the main concepts of this project. We are aiming to recover the real size of the stop signs in cities to help municipal department replace the old signs with new ones. The old and new signs have a big difference of their sizes, which means we do not have a strict request of the result precision. In the future, we might develop an application used by smart phone, the algorithm will be used in the application. We also have the chance to turn the application into a product.

6. Acknowledgement

This project is implemented during my internship in Navigation lab in Robotics Institute of Carnegie Mellon University. Chenyang is highly thankful for the helps and instructions from his mentor, Dr. Christoph Mertz, scientist in Robotics Institute, Carnegie Mellon University.