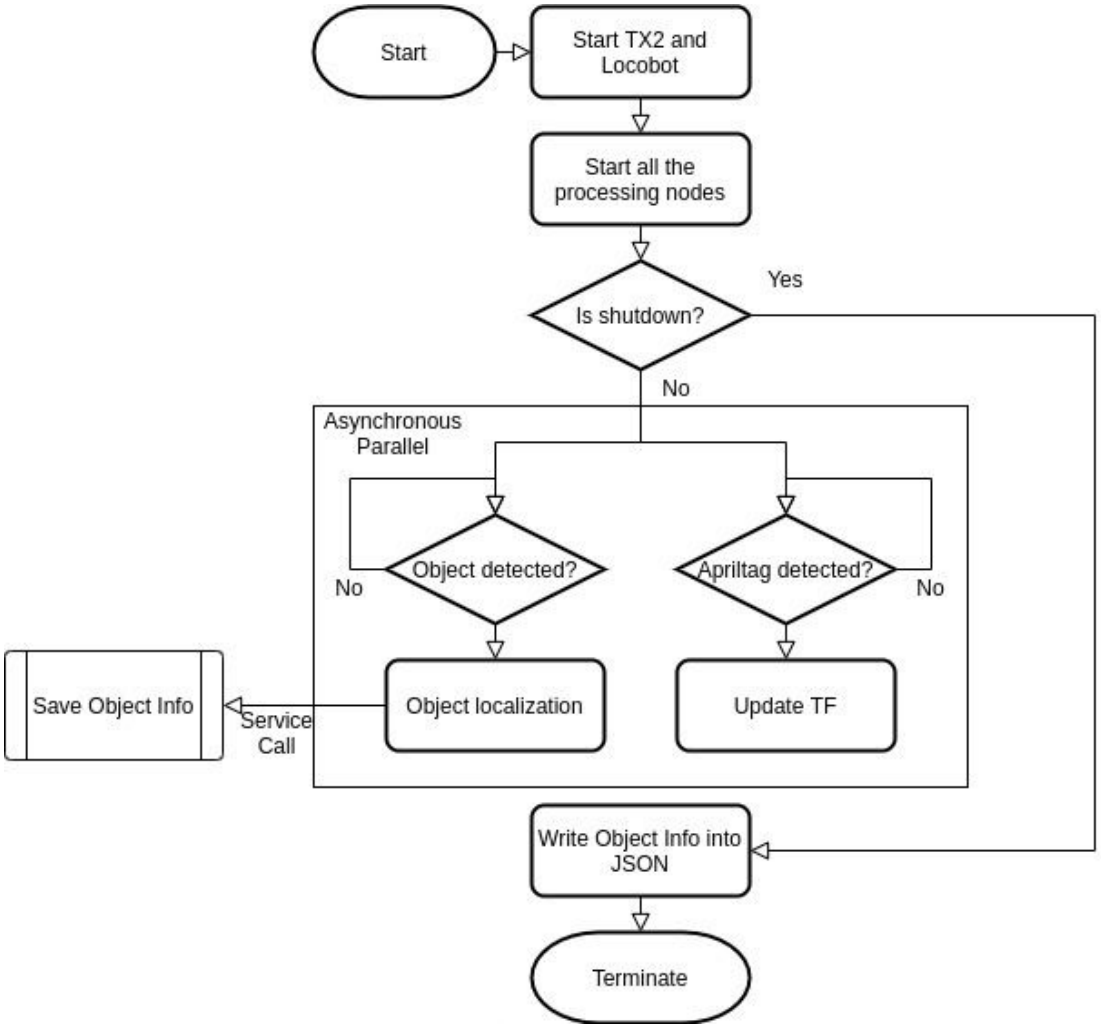
人本期末專題報告

組別4 組員: 游祖霖，陳建婷，黃柏叡

Methods



Object detection

Data Collection

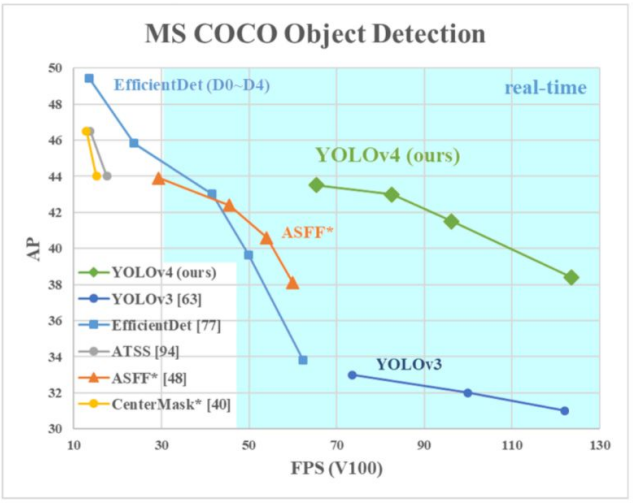
- Operate Locobot to record rosbag by using Rviz and D435
- six different backgrounds

Labelling Data

- Dataset: 500 images for each artifact(total 4000 images)
- Label bounding box using LabelImg
  - PascalVOC format(xml file)
  - write a python script to automate the change of filename in annotation files

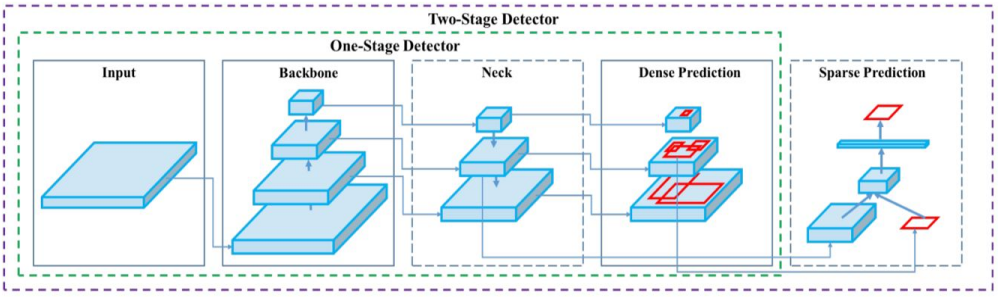
Training model

- YOLOV4



  - We choose yolov4 since its AP is much better than v3

Model Architecture



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

- Best weights: [yolo-obj_best.weights]

- We split 80% dataset for training and 20% for validation
- Iteration:16000 (2000*classes)

Detection on locobot

- D435(NUC)->TX2->laptop
  - rostopic on TX2: /teleop/image
  - laptop $ rosrun image_transport republish compressed in:=/teleop/image raw out:=/teleop/image
    Because darknet_ros has to subscribe raw image topic

The result will be published and we can see the result image with bounding boxes in Rviz.
We did face some issues with yolov4 detection on darknet_ros, we ended up having to modify a forked version of darknet_ros.

Results

We are able to obtain pretty good detection accuracy during the competition. Also, using a GPU for detection during the competition
gives us relatively high FPS, which helps in searching the environment quickly in a time-constrained setting.

Detection images in the competition environment.

However, it was a mixed bag of results when it comes to adversarial attacks. Admittedly, we didn't use any defence trick to prevent the attacks.

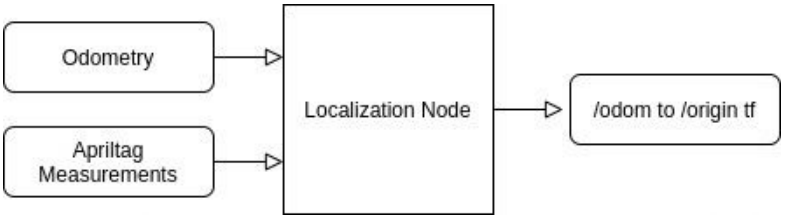
In the left photo, various other traffic signs were detected as the camera sign, whereas on the right photo, a computer cable was detected as a rope.
see our results

Localization

Our approach to the localization problem can be separated into 2 parts:

a. Robot localization
b. Object localization

Robot Localization



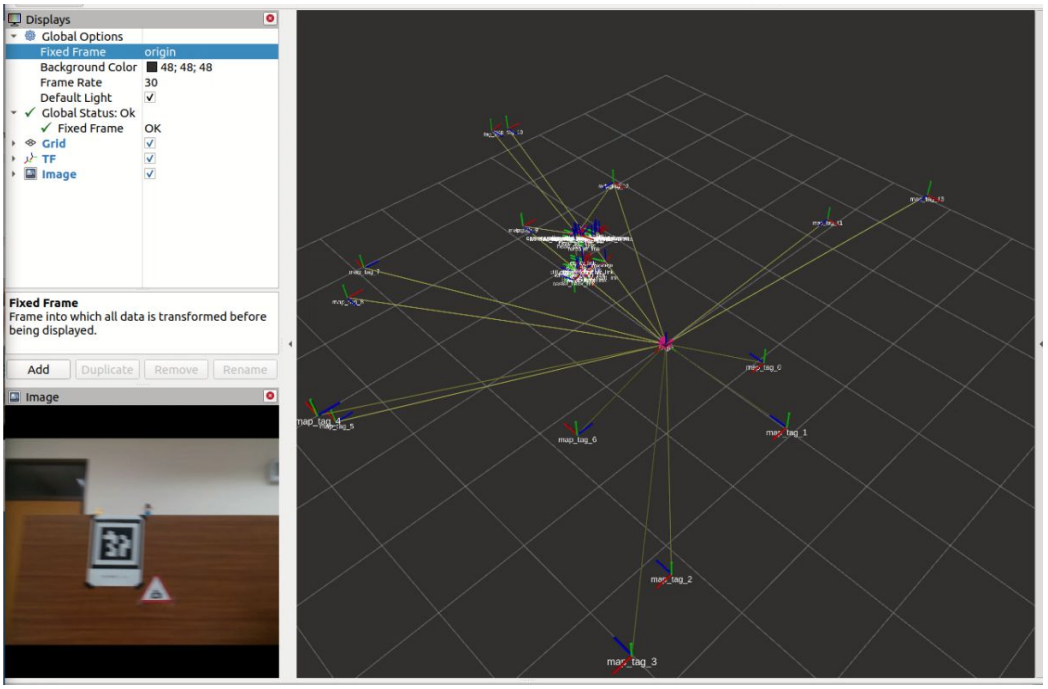
Our method involves using the provided Odometry (which we believe is a fusion of wheel odometry and inertial measurements) as the underlying localization method.

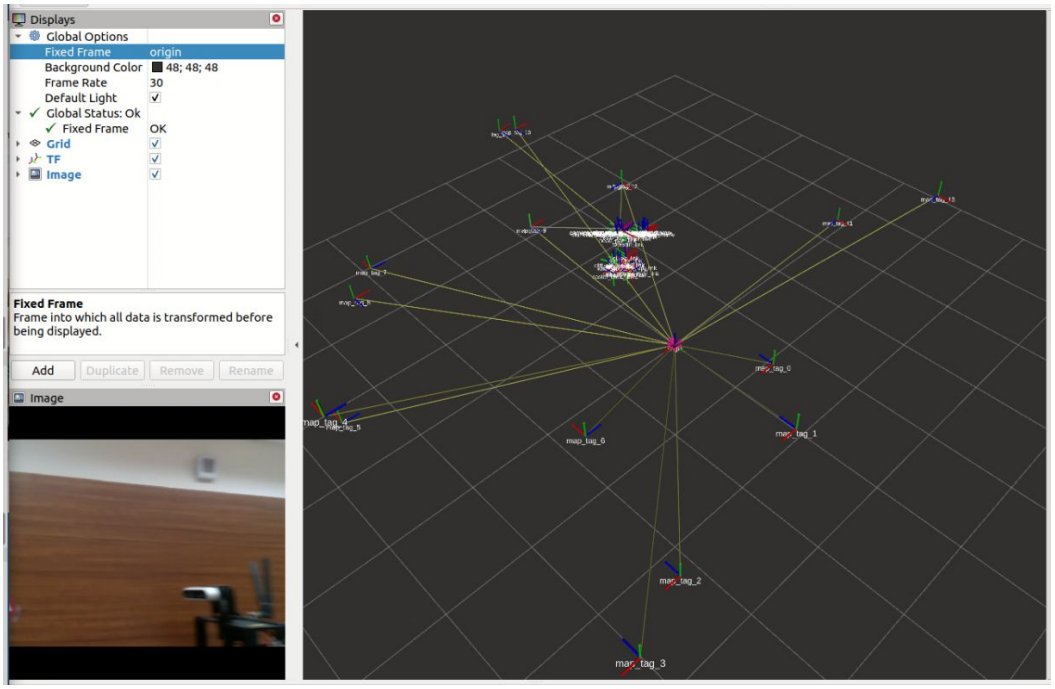
Then, we use the relative pose to the closest Apriltag detected to provide a transformation from the Odometry frame ('map' frame on locobot) to the 'global' frame.

The biggest difference of this approach compared to the one we learned in the class is that it is not necessary to have any Apriltag in the camera view to do localization.

In our method, we are able to get robot pose, $\boldsymbol{p}_t$ for all time $\boldsymbol{t}$, in the local frame and also the global frame.

Example:





However, this method is still not robust enough. There are 2 main problems.

The first is that the underlying Odometry has noticeable drifts.

Secondly, Apriltag measurements are very noisy when the robot is moving, and also when the tag is at a distance away. In this competition, we could always stop and let the robot get better measurements on the Apriltag, so it may not appear to be a big problem.

Potential improvement(s) from here:
We could use ICP to get better estimates from multiple tags instead of only one. Also, we can use these Apriltags measurements to build a Graph-based SLAM as the local SLAM, which I think would have a pretty good result with implementation of loop closure.

Object Localization
Once our robot is localized, object localization becomes quite simple.

First, we use bounding boxes and the subsequent depth measurement from D435 to find the coordinates of the object in the camera coordinates frame through perspective projection.

Because we have a connected TF from the camera to base_link and also from base_link to origin, we simply need to transform the coordinates to the origin frames (by matrix operations or simply transformPoint provided by ros::tf).

Experiments

We have done various experiments to test, analyze and improve our detection and localization method.

For object detection, we tried using mask-rcnn initially, as it is one of the best instance segmentation models. We trained the model using Facebook detectron2 system and were able to get a good result on a small subset of our dataset. However, we figured out that the mask labelling process would take up much of our limited time, and also that we might have to create our own detection node for ros whereas we can just use darknet_ros for yolo. Thus, we decided to use yolo v4 as our detection model instead of mask-rcnn.

Most of our codes were developed using data recorded thru rosbag. This enables us to do rapid testing and experimenting with different approaches and also to do bug fixes without having to set up robots and potentially deal with tricky network problems. ROSBAG rocks!

Conclusion

We are able to implement a robotic system for search and rescue (SAR) missions. We think that this competition is a good small scale simulation to test various methods. These validated methods can then be applied to a bigger scale system for real-world deployments.

Reference

LabelImg : https://github.com/tzutalin/labelImg
Yolov4 training : https://github.com/AlexeyAB/darknet
Yolo on ROS : https://hackmd.io/HmkBKksoTJ-tPq0bdhk9rw
darknet_ros (yolov4): https://github.com/tom13133/darknet_ros
fix for darknet_ros:
https://github.com/zlewe/pyrobot_minisubt/blob/master/darknet_ros/darknet_ros/src/YoloObjectDetector.cpp
ROS tf: http://wiki.ros.org/tf

Team Assignment

游祖霖: Labelling, Localization, darknet_ros fix

陳建婷: Labelling, Detection

黃柏叡: Labelling, JSON handler

Other than the codes given by TAs, our codes are all written by ourselves.
Our source code on Github: https://github.com/zlewe/pyrobot_minisubt