ELEC 3210 Final Project Report

CHENG On Kiu 20516055 LAM Man Hei 20518106

Introduction

In this project, we use ROS to simulate a mobile robot movement in V-REP simulated environment and perform several tasks including movement control by keyboard, mapping with simulated laserscan data, image recognition and localization and visual servoing.

Environment

Linux platform: Ubuntu 18.04 LTS
ROS version: Melodic
V-REP: 3.6.2 PRO EDU
Libraries used: CvBridge, cv2, math, numpy
Packages used: key_teleop, hector_mapping, rviz

Design

Task 1 Keyboard Controller (CHENG On Kiu)

The ROS package *key_teleop* is implemented to perform keyboard control, it uses *curses* to receive pressed keys and publish corresponding angular or linear velocity control commands to topic */vrep/cmd_vel* in order to move the robot. ← → keys control the angle movement and ↑ ↓ keys control vertical movement. An extra key was added to switch between manual control and auto tracking. If “a” is pressed, it will stop manual control and disable the laserscan through publishing false boolean topic */vrep/laser_switch*.

Task 2 Mapping with Rviz (LAM Man Hei)

Our SLAM are implemented with a well known ROS SLAM package hector_mapping. Since the SLAM algorithm is implemented by the package. The fine adjustments of the algorithm of are written in a launch file and thus the SLAM with custom config is launched when we used the “roslaunch” command.

The node subscribes the “/vrep/scan” laser scan topic to get the LiDAR information for mapping and localization. After that, it publishes the map through the “/map” topic and the estimated pose through the “/slam_out_pose” topic.

To visualize the map, the RVIZ is launched with our custom configuration listed in the “project.rviz” in the custom “hector_slam” package. RVIZ window subscribes to the map and poses the topic and visualizes it in the canvas.

Task 3 Area Judging (LAM Man Hei)

To judge the current area where the robot is in, we first need to know the current pose of the robot. The x and y coordinate can be known by subscribing to the “slam_out_pose” topic. After that, we can create the bounding boxes of the 4 different areas and judge if the robot is within one of these bounding boxes by comparing them with the xy coordinates of the robot.

The area judgment is published to the “/area” topic, which is subscribed by the keyboard controller and displayed in the controller UI.

Task 4 Image Recognition and Localization (LAM Man Hei)
To recognize the image in the environment, the keypoints and descriptors of each image are extracted with the Speeded-Up Robust Features (SURF) algorithm.

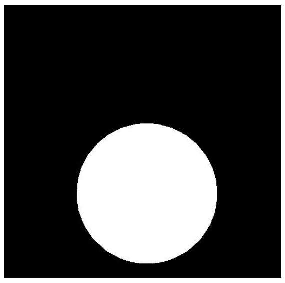
Then, the camera output from the topic “/v-rep/image” is also computed with SIFT to get the keypoints and descriptors, which will be compared with the 5 images precomputed descriptors and calculated their similarity by KNN algorithm. if the input frame is similar to one of the images, which means that that image is inside the frame. However, to make sure it is not a false trigger, it will keep checking more frames to make sure the image is indeed in the frame.

After confirming an image is in the camera output, the contour of the image is found and obtain its position, width and height. Combining with the given camera model, the exact location of the image can be calculated under the frame “/camera_link”.

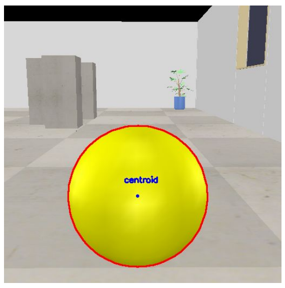
However, since “/camera_link” is disconnected with the global frame that is used in RVIZ. “static_transform_publisher” from the package tf is used to transform the “/camera_link” so that the marker indicating the image location can be correctly shown with respect to the map.

Task 5 Visual Servoing (CHENG On Kiu)
Workflow:

1. Use CvBridge to get a camera image from topic */vrep/image*, convert it from BGR to HSV format by *cv2.cvtColor*. Mask the image with yellow colour by *cv2.inRange*. The shape of the ball will be obtained.



2. Get the contour of the ball by *cv2.findContours*, find its centroid coordinate by *cv2.moments*



3. Compare the current centroid coordinate with the ideal centroid coordinate (calculated with the size of image and radius of the ball) that keeps the ball in the center of view and close to the robot without getting out of bound. Angular and linear velocity commands will publish to topic */vrep/cmd_vel* according to the horizontal and vertical difference

between the two centroids. The farther it shifts, the larger velocity command will publish.

This tracking function will be triggered when laserscan is off (by receiving */vrep/laser_switch* topic) and the camera can detect a ball contour.

Task 6 Launch File (LAM Man Hei)
A single launch file “start.launch” is created to launch all our nodes all at once. After running “roslaunch start.launch” command, a RVIZ window with custom config is launched, and all our custom nodes will be running. However, since there are multiple configurations with hector mapping node, the parameters are written in a separate launch file in order to declutter the master launch file.

Applications

Above works can be applied in different industries and implement automatic operations. For example in the logistics industry, with the use of slam mapping, robots in warehouses can move to target cabinet automatically. Utilizing image recognition, robots can recognize goods needed and pick them up for delivering. It has higher working efficiency and lower cost compared to manual operation. For home automation, cleaning robots can record places they have cleaned already and avoid cleaning the same place. So that they can finish cleaning in the shortest time.

Conclusion

There are two limitations in our work. Firstly, since SLAM does not have an odometry input, when the robot has a sudden shift, SLAM cannot discover thus the robot location in the map will have errors. Secondly, since there is no PID control in the ball tracking function, the robot oscillates a lot while tracking.

In the future, we can implement a more robust SLAM algorithm with odometry information, so the map created is more accurate and without deformation. Also, we may create an auto-discovery mode that utilizes the generated map to plan the path automatically, so we do not need to control the robot manually. Finally, we can implement PID control for ball tracking for a smoother tracking path.

Demo and Presentation Link: https://youtu.be/FEm4Ev3ZY8M
Project GitHub Link: https://github.com/Doma1204/ELEC3210-Project

Reference

1. Key Teleop: http://wiki.ros.org/key_teleop
2. Hector Mapping: http://wiki.ros.org/hector_mapping
3. SURF Algorithm (OpenCV): https://docs.opencv.org/master/df/dd2/tutorial_py_surf_intro.html
4. HSV Colour Space: https://www.itread01.com/content/1542732610.html
5. OpenCV(cv2.findContours): https://blog.csdn.net/hjxu2016/article/details/77833336
6. OpenCV(find center): https://www.learnopencv.com/find-center-of-blob-centroid-using-opencv-cpp-python/