Vision Based Exploration of Robot in Fully Unknown Environment

Tianyou Hu
Washington University in St. Louis

Follow this and additional works at: https://openscholarship.wustl.edu/mems500

Recommended Citation
https://openscholarship.wustl.edu/mems500/219
Vision Based Exploration of Robot in Fully Unknown Environment

Tianyou Hu
Department of Mechanical Engineering and Material Science
Washington University
St. Louis, Missouri
h.tianyou@wustl.edu

Abstract—This research aims to develop an algorithm capable of exploring a completely unfamiliar environment using vision-based techniques. The study applied simultaneous localization and mapping (SLAM) and navigation algorithms on the ROS simulator and Turtlebot robot, as well as the YOLOv7 neural network for object detection and depth estimation. The robot’s ability to track and track humans in its environment has been improved by combining data from YOLOv7 and Lidar, and applying neural networks to predict human trajectories. Research results have demonstrated the effectiveness of vision-based techniques in developing autonomous robots capable of navigating and exploring an unfamiliar environment, making it useful for applications such as search and rescue operations, environmental monitoring, and human-robot interaction.

Keywords—Object Detection, Trajectory Tracking, Deep Learning, ROS, Ubuntu, Linux

I. INTRODUCTION

This study utilizes Robot Operation System (ROS) 1 Noetic version and Ubuntu 20.04 Linux system, and was firstly implemented on ROS simulator then Turtlebot Raffle Pi robot (Fig. 1) in our lab. It starts from controlling the Turtlebot from remote PC. The remote PC is able to control Turtlebot’s linear and angular velocity, by which it can make Turtlebot move forward, backward, turn left and right.

By utilizing the Lidar on the top of Turtlebot, Simultaneously Localization and Mapping (SLAM) algorithm is able to draw a map of unknown environment, which was applied both on Gazebo and Turtlebot. Gazebo is a ROS based simulator which creates a virtual environment of real life. For example, it can create a house environment including furniture and walls. The test of paper [1] was done in Gazebo. Paper Fig. 2 shows that Turtlebot is able to built the map of its surroundings, in which green points are current feedback from Lidar and black points are drawn map of obstacle. Gmapping is the most famous SLAM method, which runs pretty well on Gazebo simulator. However, it doesn’t work well on Turtlebot. It is highly possible that map would be messed up when Turtlebot is twisting. This problem can be solved by using Karto SLAM method. Fig. 3 shows the map of our lab built by Karto. Then Rapidly Random Exploring Tree (RRT) algorithm was used for navigation in the map built by SLAM, shown in Fig. 4. In Fig. 4 Turtlebot was following the path of red arrow. The pink part is verifying its surrounding with the built map to find its position in the map. Both SLAM and RRT was conducted in Rviz environment. However, Turtlebot is not able to navigate in the unknown map, which is a big limitation. Article [2] provides an algorithm that combines SLAM and RRT to realize autonomous navigation in unknown map, which was used in the later part of this research.

Fig. 1. Turtlebot Waffle Pi

Fig. 2. SLAM
The next step of this research is to track human trajectory. Inspired by paper[3], this research combines object detection and Lidar data to obtain human position. YOLOv7 algorithm, a real time classification algorithm[4], was used for object detection to get pixel indices of human in raw image. Angle of human to the axial direction was calculated by knowing the angle range of Rasberry Pi camera on Turtlebot. Lidar data was used for obtain distance of targeted human from Turtlebot. Human’s x and y coordinates can then be calculated by simple trigonometric function. For a human, there would be many points detected by Lidar. The hard question is which point represent humans overall position. My algorithm firstly choose the closed point, the take the mean value of the points that is within the specific distance to the closest point. This specific distance is set 20 cm which is my shoulder width. The raw human’s trajectory tracked is coarse and not reliable. Savitzky-Golay filter was used to filter the path. However, each point may be too close to each other after filtering, leading to unreliable predictions by deep learning network. To refine the path, Python slice was used to take a points within a certain amount of points evenly. LSTM deep learning network was used for predictions. Depth estimation was also applied on Turtlebot to obtain pixel wise distance.

II. HUMAN TRAJECTORY TRACKING

A. YOLOv7 input and output

YOLOv7 is the 7th version of YOLO, which is more real-time than previous version. It inputs raw image and output raw image with frameworks on selected categories, shown in Fig. 5. It would also output the x and y indices of the lower left and upper right pixel of the framework. The computer vision code is written in CV2, whose y axis is opposite from usual x y coordinate system.
B. Theta

Once knowing the x and y indices, theta can be calculated. For Raspberry Pi camera, its image size is $640 \times 480$. The x and y axis contains 640 and 480 pixels respectively. It is then compressed in a square image for visualization. Which means for Fig. 5 the image size is $640 \times 480$ instead of same horizontal and vertical pixel amount. Theta is calculated as following formula where $i$ represents index of pixel. There are two thetas for left most and right most x indices respectively. Theta only represents 2 dimension angle.

$$\theta = \arctan \left( \frac{|i - 320|}{320} \right) \times \tan 31.1^\circ$$

Theta's ranges from $-31.1^\circ$ to $31.1^\circ$. The reason is that angle of Raspberry Pi camera is $62.2^\circ$. And the Lidar message lasers is counter clockwise.

C. Index of Lidar

Lidar message is not fixed. The number that Lidar receive distances in the $360^\circ$ is usually around 219. In order to know the distance human. Index of the the Lidar need to be found. By knowing the left most and right most theta, we know Lidar index if the left and right most pixel. Then all the Lidar points inside the framework can be found. The closest point is firstly choose, then mean value of the points that is within the specific distance to the closest point was taken as the human position. This specific distance is set 20 cm which is my shoulder width.

D. Human Recognition and Position Tracking

Shown in Fig. 7, The algorithm is able to track real-time multiple human position. Depth represents y coordinate. Left or right of center represents x coordinate.
E. Savitzky-Golay filter

Savitzky-Golay filter was used to filter path. Fig. 8 shows the raw path detected by YOLO and Lidar when human was walking S path. Fig. 9 shows the filtered path. The kernel of Savitzky-Golay filter is set 49. The raw path is coarse and is highly likely to cause wrong predictions by deep learning network. After filtering, the path is clear and reliable.
F. Slicing & Deep Learning Network

Although the filtered path is clear enough, the distance between each point is too close, which will cause the unstable predictions. Fig. 10 shows that when points are close to each other, the prediction is non-sense. By using python slice, the filtered path is transformed to Fig. 11. This is because the utilized deep learning network (Long Short Term Memory) LSTM [5] is pre-trained and is suitable for predicting limited points which have certain distance from each other.

The LSTM shows excellent prediction performance when the trajectory is simple and clear. Fig. 12 shows LSTM prediction for the same path at different time. By comparing the first 2 images, the prediction of first image is almost the same
as that part the second image. All the predictions make sense based on the given path. The second the fourth image is not able to predict future path because of the uncertainty of this path sample. The default prediction length of LSTM is 11 which is the prediction length of first image and it works the best.

LSTM is also able to recognize S path and make corresponding prediction, shown in Fig. 13.
III. CODING FOR TRAJECTORY TRACKING

This parts illustrate how to use my code and its details. The code runs on Ubuntu 20.04 an ROS Noetic. Once the Turtlebot is initialized, run “roslaunch ros_autonomous_slam autonomous_explorer.launch” in a new terminal, and it will open Rviz environment. Then open a new terminal and run “roslaunch yolov7_ros yolov7.launch”, which will create YOLOv7 node. To visualize YOLO classification, simply change image topic in Rviz to yolo topic. Now YOLO is tracking real-time position.

Then open a new terminal and run “python3 get_person_position.py” which will track human position and save it in txt file. The interface is shown in Fig. 14. get_person_position.py outputs 2 thetas, 2 indices of Lidar, closest point distance and x y coordinate.

Then open four terminals and run the following code. The first one implement filter and slicing, outputting the animation of refined path.

The second terminal (upper right) convert .txt file which contains path to .ndjson file. Ndjson is the version of path used by LSTM model. For the printings by this terminal, there is test row category, meaning that the converted .ndjson file only contains 30 frame which are just 30 points.

The third (lower left) terminal implements the prediction. There are 2 arguments that need to be changed every time the code is run. They are obs_length and pred_length representing how many points taken for predicting and how many points are going to be predicted.

The last terminal visualize the prediction by creating an animation of mp4 version. The length represents obs_length.
IV. DEPTH ESTIMATION

Real time depth estimation\cite{6} is also applied on Turtlebot because the Raspberry Pi camera is not able to detect distance. The test environment is shown in Fig. 16 and the processed output is shown in Fig. 17. In the interface, you can tell that Turtlebot is able to set the goal outside the given map to autonomously navigate. I believe the depth estimation algorithm will be useful in my future step for this project.

Fig. 16. Test Environment

Fig. 17. Real-time Depth Estimation
V. CONCLUSION

In conclusion, this research demonstrated the effectiveness of vision-based techniques, such as object detection, simultaneous localization and mapping (SLAM), and deep learning, for developing autonomous robots capable of exploring and navigating completely unfamiliar environments. The combination of YOLOv7 and Lidar data improved the robot's ability to track and follow humans in its surroundings. The study also showed that the Karto SLAM method was more effective than Gmapping for building accurate maps of unknown environments. Finally, the research proposed an algorithm that combined SLAM, RRT, and LSTM for autonomous navigation and human trajectory tracking, making it useful for applications such as search and rescue operations, environmental monitoring, and human-robot interaction.

REFERENCES


