Implementation of Enhanced Vision for an Autonomous Map-based Robot Navigation

Cubahiro Roland · Donggyu Choi · Minyoung Kim · Jongwook Jang*

Dong-Eui University

E-mail : bukuja@gmail.com / dgchoi@office.deu.ac.kr / kmyco@deu.ac.kr / jwjang@deu.ac.kr

ABSTRACT

Robot Operating System (ROS) has been a prominent and successful framework used in robotics business and academia.. However, the framework has long been focused and limited to navigation of robots and manipulation of objects in the environment. This focus leaves out other important field such as speech recognition, vision abilities, etc. Our goal is to take advantage of ROS capacity to integrate additional libraries of programming functions aimed at real-time computer vision with a depth-image camera. In this paper we will focus on the implementation of an upgraded vision with the help of a depth camera which provides a high quality data for a much enhanced and accurate understanding of the environment. The varied data from the cameras are then incorporated in ROS communication structure for any potential use. For this particular case, the system will use OpenCV libraries to manipulate the data from the camera and provide a face-detection capabilities to the robot, while navigating an indoor environment. The whole system has been implemented and tested on the latest technologies of Turtlebot3 and Raspberry Pi4.

Keywords

ROS, Raspberry Pi4, Realsense Camera, OpenCV

Ⅰ. Introduction

In limited and supervised situations, autonomous robots are extremely successful. When tested in new and outdoor setting however, a robot must have some way to perceive its surroundings[1].

In this paper, the authors weigh in on the importance of perception for the vision of an autonomous robot[2]. They argue that human beings need vision to perceive the environment. This is also the case for autonomous cars which build up their capacity to recognize the environment on an infrastructure of various types of sensors, lidars, radars in order to trace the obstacles and determine the position of the car. All this data are provided by the use of camera and computer algorithms.

Vision in a robot system ranges from knowing the environment, to find the path toward a destination avoiding obstacles in the process, including self localization and an ability to recognize objects.

In this case, we will propose an implementation of an enhanced vision to a ROS-based robot system. The goal of the implementation is an autonomous robot capable of recognizing elements of the environment. The system is mounted on a Waffle Turtlebot3 robot running on ROS1.

Turtlebot3 is a ros-based robot used in research and other prototyping applications. Its software consists of an OpenCR board handling the low-level control of the system. Many devices are used to implement the functionalities of the robot. In the case of Turtlebot3 Waffle Pi a ARM Cortex-M7 micro controller handles the actuator and sensor control. In addition a Raspberry Pi4 board is connected.

This paper will focus on three features of the system: SLAM algorithm for mapping the surface, RealSense depth-image installed on the system which is used for all the applications of this paper's scope and, finally, OpenCV library is used to build a code that can detect human faces in the environment.

Ⅱ. System design

2-1. Navigation

* corresponding author

In this part we are looking at the specifics of the system implementation. In order to make the robot move, it first needs a map describing the surface to navigate, secondly it needs the position of the robot at a given time, thirdly, a sensing of the environment to recognize potential obstacles and finally a path calculation model to optimize the best route to destination.

In this case we will use gmapping algorithm to create a map. The gmapping provides laser-based SLAM data on a ROS node called slam_gmapping. which helps us to create a 2D grid map.

In order to get the pose estimation, Encoders and IMU (Inertial Measurement Units) are using the dead reckoning process (Figure 1), which calculates the rotations of the wheel to render the pose of the robot. The estimation pose can also rely on the sensors and camera to give a correct estimation of the robot's position.

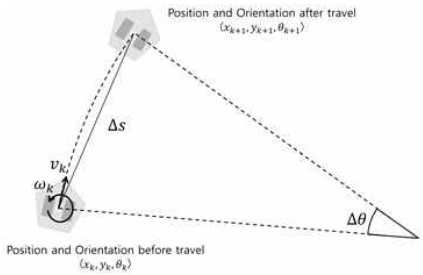

Figure 1. Dead Reckoning calculation

Thirdly, sensors allow us to detect any object standing as an obstacle. Different types of sensors are used for the task. Here, we will use a tree dimensional range sensor, the depth-image camera.

Path planning allows us to create a path from point A to point B. The path generated is made of the global path planning for the large view of the map and the local path planning for the reading of the close areas of the robot. Dynamic Window Approach, a collision avoidance algorithm is applied on the system[3]. Below are the equations which helps to calculate the movement of the wheels, Figure 2 [4]:

$$\Delta s = v_k T_e \qquad \Delta\theta = \omega_k T_e$$

$$x_{(k+1)} = x_k + \Delta s \cos\left(\theta_k + \frac{\Delta\theta}{2}\right)$$

$$y_{(k+1)} = y_k + \Delta s \sin\left(\theta_k + \frac{\Delta\theta}{2}\right)$$

$$\theta_{(k+1)} = \theta_k + \Delta\theta$$

Figure 2. Equation for Position and Orientation of the robot

2-2. Vision

The sensors information can help to correct the robot's position (Odometry) as we saw in last section. However, the robot's perception must interact intelligibly in the environment. For this purpose we use Intel RealSense Depth camera d435, an optimized tool thanks to its capacity to give high frame rate data with depth accuracy.

The camera feed from the RealSense is integrated in ROS communication nodes through a cross-platform toolkit which extract data from the camera.

2-3. OpenCV

OpenCV is a library for real-time computer vision. OpenCV was written in C++ but has interface in other languages. Its image processing modules are helpful and adequate for the target of this paper. We will use the Haar cascade detection algorithm, a machine learning algorithm trained with the help of many positive and negative images for object detection.

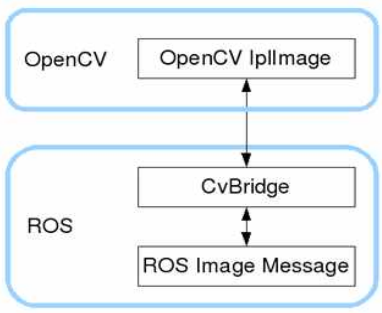

Figure 3. Dead Reckoning calculation

CvBridge is a ROS library that provides an interface between ROS and OpenCV.

ROS circulates image feeds on its communication

nodes (sensor_msgs/Image) which can be accessed and manipulated with the help of cv_bridge (Figure 3). Once the camera feed is available, a C++ script is used to filter the content and publish it back on ROS.

The main functions of the code are explained in the following table:

Table 1. Snippet of the Face Detection code

	Code	Description
1	image_sub = it_.subscribe("/camera/color/image_raw", 1)	To suscribe and publish again the videdo feed on the /camera/color/image_raw" node.
2	cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);	To convert the video feed in RGB format useful for content manipulation.
3	faceDetection.load("/.../haarcascade_frontalface_default.xml");	To load the Haar Cascade pre-trained model saved on the local computer.

III. Results

The function of the system is first to be able to run the robot autonomously with a provided map. While doing so, the robot must be able to detect faces when it encounters people in the way. Both goals were achieved during the test performance. Figure 4 shows the mapping process and the video stream provided by the same camera with a face detected.

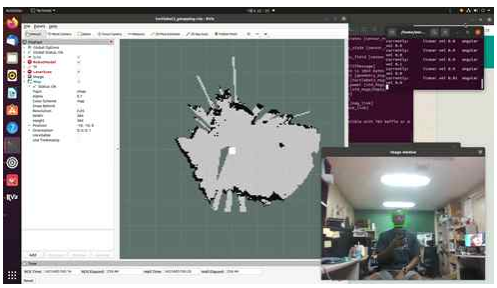

Figure 4. Visualization of the mapping and face detection output.

IV. Conclusion

Applications of this system are numerous, and the system can be optimized for greater performances. However, the system has some limitations:

The SBC used in this case (Raspberry pi 4) is the latest version and has greater performance. But it froze when multiple tasks are launched together. This is not preferable when the system is running. Better platform with powerful calculations capacities like the Jetson Nano can be used instead.

The Haar Cascade classifier model is light and easy to use, but its detection accuracy is low. In practical uses, it can give unexpected results. There are more accurate models with a vast range of objects inputs such as the YOLO or Fast R-CNN.

Acknowledgement

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program(IITP-2021-2020-0-01791, 'Busan AI Grand ICT Research Center Support Project') supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation). And, This research was supported by the BB21plus funded by Busan Metropolitan City and Busan Institute for Talent & Lifelong Education(BIT).

References

[1] B. Abhishek, S. Gautham, D. Varun Rufus Raj Samuel, K. Keshav, U. P. Vignesh and S. R. Nair, "ROS based stereo vision system for autonomous vehicle," *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pp. 2269-2273, Chennai, India, 2017.

[2] G. Ros, S. Ramos, M. Granados, A. Bakhtiary, D. Vazquez and A. M. Lopez, "Vision-Based Offline-Online *Perception Paradigm for Autonomous Driving," IEEE Winter Conference on Applications of Computer Vision*, HI, USA, pp. 231-238, 2015.

[3] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim, *ROS Robot Programming*, 1st ed., pp. 315, Seoul Republic of Korea, 2017.

[4] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim, *ROS Robot Programming*, 1st ed., pp. 317, Seoul Republic of Korea, 2017.