Mugla Journal of Science and Technology

MOBILE ROBOT CONTROL WITH ANDROID DEVICE SENSORS BY USING ROS

Abdullah Erkam Kırlı[1], Mustafa Burak Dilaver[1], Furkan Çakmak[1]*

[1]Computer Engineering Department, Electrical Electronics Faculty, Yildiz Technical University, 34220, Istanbul, Turkey
erkamkrl@gmail.com, mbdilaver@gmail.com, furkan@ce.yildiz.edu.tr

Received: 05.12.2016, Accepted: 02.06.2017 doi: 10.22531/muglajsci.272475
*Corresponding author

Abstract

In this study, a mobile device with Android operating system was used to control a six-wheel differential drive mobile robot. In the literature, it is seen that there is no open source and comprehensive study in this matter, so that this study aimed to give a sample work for future applications. The Robot Operating System (ROS) framework has been established on the mobile robot and the operations have been implemented on the ROS, while on the Android device, the improvements have been made using the ROS libraries. While the camera image and the calculated map information from the mobile robot are transferred to the Android device via the ROS, vice versa the data that controls the robot actions are transferred. The Ubuntu operating system on the Raspberry Pi 2 microcontroller used on the mobile robot was used with the Indigo version of the ROS. In addition, on the Android device, using the accelerometers and touch tones, the control of the robot has been provided in 2 different ways. Thus, it is aimed to ensure mobile robot control effectively in teleoperation mode.
Keywords: Mobile Robot Control, ROS, Android OS, Raspberry Pi Application

ROS ÜZERİNDEN ANDROID CİHAZ DUYARGALARI YARDIMIYLA MOBİL ROBOT KONTROLÜ

Öz

Bu çalışmada, üzerinde Android işletim sistemi yüklü bir mobil cihaz yardımıyla 6 tekerlekli diferansiyel sürüşlü bir mobil robotun kontrolü ele alınmıştır. Literatürde, bu konuda tam manasıyla açık kaynaklı ve kapsamlı bir çalışma yapılmadığı görüldüğünden, örnek bir çalışma ortaya konmak istenmiştir. Mobil robot üzerinde ROS (The Robot Operating System) çatısı kurulmuş ve işlemlerin ROS üzerinden gerçekleştirilmesi sağlanmışken, Android cihaz üzerinde ise ROS kütüphanelerinden yararlanılarak geliştirmeler yapılmıştır. Mobil robot üzerinden alınan kamera görüntüsü ve hesaplanan harita bilgisi ROS üzerinden Android cihaza aktarılırken, tam tersi yönde robotun sürüşünü kontrol eden veriler aktarılmaktadır. Mobil robot üzerinde kullanılan Raspberry Pi 2 mikrodenetleyicisi üzerinde Ubuntu işletim sistemi, ROS'un Indigo sürümüyle birlikte kullanılmıştır. Ayrıca Android cihaz üzerinde ivmeölçer ve dokunmatik duyargaları kullanılarak robotun kontrolü kullanıcı seçimine bırakılarak 2 farklı şekilde gerçekleştirilmiştir. Böylece mobil robot kontrolünün teleoperation modda efektif bir şekilde gerçekleştirilmesi amaçlanmıştır.
Anahtar Kelimeler: Mobil Robot Kontrolü, ROS, Android OS, Raspbery Pi Uygulaması

1 Introduction

The aim of this work was to provide control of an installed robot with the ROS framework [1] using accelerometers and touch sensors on a mobile device with an Android operating system. As an open source, the ROS, which is constantly evolving, has demonstrated that the Android environment can be used to communicate with each other over a wireless connection (Wi-Fi), as well as testing the libraries of ROS that have been put the experimental version on market. Images taken with a webcam on the robot are transferred to the Android device via the ROS nodes and reflected in the user interface on the Android device. In addition, the Hokuyo URG-04LX model laser sensor was used to scan up to 4 meters of 180° angle, and these scan results were presented to the user through the Android device. Using the Laser Scan Matcher (LSM) [2] method, the robot position is instantly informed to the user and gMapping [3] from the simultaneous localization and mapping algorithms (SLAM) is executed by including the position information and the generated results are transferred to the Android device through the same interface.

On the wiki pages of the ROS, there are several libraries for controlling the mobile robot from the joystick or keyboard [4]. The main motivation for achieving this work is that there are no instances that can work with other operating systems (Windows, iOS, etc.) running on Android or mobile devices.

Section 2 includes a detailed examination of the robot platform along with the equipment used and features of the Android device, used method in this project explains in Section 3, implementation and experimental results are deal with in Section 4, and finally evaluation and conclusion part are given in Section 5.

2 Robot Platform, Hardware and Android Device

The robot platform, hardware used in this study and device with the Android operating system in which the communication is established have been investigated in the subheadings.

One of the picture of the robot platform is given in Figure 1. It can be seen that the laser sensor, used for SLAM, is placed in front of the robot to ensure not obstruct laser sensor's angle of scan. Camera, used for safe driving, is placed the highest point of the robot to enlarge angle of view. Thus, operator can drive mobile robot more effective. There are two button in front of the robot to use in emergency situations. That buttons cut the power off between battery and the wheels to provide more

flexibility in case there will be an emergency. The wheels are not fixed to the robot's body to use the robot in rugged terrains.

Figure 1. Robot platform used in the study.

2.1 Raspberry Pi 2

Raspberry Pi 2 (RP2) is a single board and credit card-sized computer with a 4-core processor and 1 GB RAM capacity. Processing power of RP2 is quite good enough, especially the conversion of data from a laser sensor into a map. Ubuntu 14.04 LTS operating system has been installed on RP2 and used for operations such as giving velocity command to the robot, camera and laser data retrieval and communication with Android device.

2.2 Arduino Uno

In this study, Arduino Uno was used to communicate with RP2 and the shield which control the motors and to read data from a sonar sensor, used for measuring distance between obstacles and robot's back.

2.3 Robot Platform

The Dagu Wild Thumper 6WD [5] robot platform was chosen for this study, because besides all of adaptive properties, it has lots of empty areas for electronic cards and batteries and the driving strategy of this platform (differential-driven) is more proper for us.

2.4 Hokuyo Laser Range Finder

The Hokuyo URG-04LX laser sensor is used on the robot platform. This laser sensor has a 180° scan angle and it can measure up to 4 meters. The laser is used for building a map. Mapping method, described in detail in Chapter 3, was performed more accurint if the frequency of scans is high. This laser sensor is good enough to be used for gMapping.

Of course, one of the most important reasons for preference is compatibility with ROS.

2.5 RBG Camera

There is no specific feature that stands out in favor of the web camera. The same operations can be done in the same way with another camera if network traffic of that camera is not intensive (reviewed in chapter 4). Specifically, the model of webcam used in this work is Everest ATW-M10 and can display images at 1280x960 with 30 fps.

2.6 Battery

Main text should be written in Cambria font and 9 pt. In special cases, e.g. making an emphasis, other types of fonts can also be used.

2.7 Circuit Diagram

The connection of the hardware elements on the robot platform to each other is shown in Figure 2.

The voltage from the LiPo battery is reduced to 5V with the aid of one regulator to feed RP2. The Arduino Uno is connected to the Pololu motor drive shield via the USB port on the RP2 and the pins on it. The motors of the robot platform are connected to the h-bridge, so that the same voltage is applied to the three wheels on the left and on the same three wheels on the right.

Likewise, the 12V voltage required for the Hokuyo laser sensor to operate is supplied via a regulator and then connected to the RP2 via USB to send the scan data. RP2 card doesn't have any Wi-Fi antenna. So, an external Wi-Fi dongle is connected to RP2 to provide a network connection between the robot platform and the Android device.

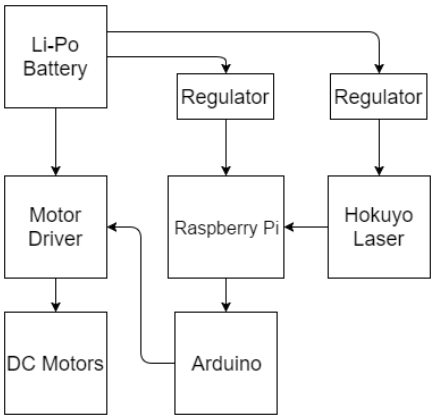

Figure 2. The connection of the hardware elements.

2.8 Android Device

During this study, Asus Memopad 7 with Android 5.0 operating system, which is shown in Figure 3, is used. Besides multi touch feature, this device has accelerometers sensor. This sensor's data is used for driving the robot with air gesture.

Figure 3. Asus Memopad 7.

The key features of the used Android device are listed in Table 1.

Table 1. Android device features.

Processor	Quad-core 1.2 GHz Cortex-A7
Memory	1 GB
Graphic Processor	PowerVR SGX544
Capacity	8 GB
Screen Size	7.0"
Resolution	800 x 1280 pixels

3 Methods Used

The complementary methods for controlling the mobile robot on teleoperation mode via Android device are given in subheadings. Operator must know where the robot is at a certain time to give drive command more accurate. The information of robot position is called as odometry. There are a lot of odometry extraction method in literature. These can be listed as follows. The first one is wheel odometry. This type of odometry can produce if only there is encoder on the wheels. Compare to other types, this is not good enough to use in mapping methods. Another one is visual odometry which are extracted from sequential images provided from a camera. The third one is obtained from laser scans. In this study, odometry information is produced from laser scans using Laser Scan Matcher algorithm.

3.1 Laser Scan Matcher (LSM)

Laser Scan Matcher is a method that produced odometry information as a result of matching the laser scans received in consecutive time intervals to each other. It is aimed to increase the convergence of the odometry results produced by LSM to the mapping algorithm and to gain from the calculation time.

The LSM algorithm, which is used in the study and shared codes as an open source in the ROS repository, is a point-to-line metric Iterative Closest Point (PLICP), an improved version of the Iterative Closest Point (ICP) method [2].

The ICP algorithm is an iterative method used to calculate trans-rotation (rotation $R(\theta)$, translation t) at the $\{p_i\}$ point recorded on the S^{ref} reference surface. From the given set of $\{p_i\}$, the recorded q is as given by the trans-rotation value in Equation (1).

$$p \oplus q = p \oplus (t, \theta) \triangleq R(\theta)p + t \tag{1}$$

The ICP tries to find the value of q which minimizes the distance between the transformed point p_i and the Euclidean projection to the reflections S^{ref}. The ICP minimization function is like that given by (2). Here, $\prod\{S^{ref}, p\}$, S^{ref} represents the Euclidean projection to the ref.

$$\min_{q} \sum_{i} \left\| p_i \oplus q - \prod\{S^{ref}, p_i \oplus q\} \right\|^2 \tag{2}$$

There is no closed form solution for Equation (2). Thus, the initial q_0 value can be generated based on the trans-rotation value as given by the iterative constraint function as in Equation (3).

$$\min_{q_{k+1}} \sum_{i} \left\| p_i \oplus q_{k+1} - \prod\{S^{ref}, p_i \oplus q_k\} \right\|^2 \tag{3}$$

Different ICP approaches can be given for different $\prod\{S^{ref}\}$ definitions. The PLICP algorithm generates a closed form solution to give a nearest linear distance to a given point. While the point-dot recording approach converges linearly, PLICP converges quadratically and appears to be given by the constraint function in Equation (4). n_i^T value is used as a transpose of a given point to the nearest normal line of the reference line.

$$\min_{q_{k+1}} \sum_{i} (n_i^T [p_i \oplus q_{k+1} - \prod\{S^{ref}, p_i \oplus q_k\}])^2 \tag{4}$$

The PLICP algorithm can be defined as shown below to denote y_{t-1} reference laser scan, y_t second laser scan, q_0 initial trans-rotation value, i second laser scan index, j reference laser scan index and k iteration count.

Algorithm PLICP
Input: y_{t-1}, y_t, q_0

$S^{ref} \leftarrow y_{t-1}$ *Segment line segment*
$k \leftarrow 0$
repeat
  $p_i^w \leftarrow p_i \oplus q_k$
  $j_1^i, j_2^i \leftarrow p_i^w(j_1^i, j_2^i \in y_{t-1})'$ *nearest two point*
  $C_k \leftarrow$ *all* $\langle i, j_1^i, j_2^i \rangle$
  C_k' *clear outlie*
  $J(q_{k+1}, C_k) \leftarrow \sum_i (n_i^T [R(\theta_{k+1})p_i + t_{k+1} - p_{j_1^i}])^2$
  J' *minimun Value at* q_{k+1}
  $k \leftarrow k + 1$
until$(maks\_iteration\_number)$ *or* $(coverage)$

Output: q

3.2 gMapping

GMapping [3] is one of the Simultaneous Localization and Mapping algorithms which are used in a lot of robot research. GMapping is a method that uses only the laser distance measurement sensor and passes to the mapping steps after selecting the optimum starting point with the help of odometry information produced by the LSM.

GMapping is based on the Rao-Blackwellized Particle Filter (RBPF) [6], a multi-particle grid-based method. Each particle holds its own belief in the previous position of the robot and builds its own map. Each incoming laser scan updates the beliefs of the particles. The beliefs of the particles are based on the robot position and orientation. In addition, gMapping uses its own laser scan mapping method in the optimization step. The gMapping algorithm implemented on the ROS and used in the scope of the study consists of 6 important steps as shown below.

1. **Measurement acquisition:** A new laser scan is taken.
2. **Scan Matching:** Performed by pairing of previous and current laser scans.
3. **Sampling:** Previously taken $\{{x_{t-1}}^{(i)}\}$ particles help and suggested distribution $\pi(x_t|z_{1:t}, u_{0:t})$to perform best $\{{x_t}^{(i)}\}$ particles are calculated.
4. **Weighting:** Each particle in (5) as given one $w^{(i)}$ weight is calculated.

$$w^{(i)} = \frac{p({x_t}^{(i)}|z_{1:t}, u_{0:t})}{\pi({x_t}^{(i)}|z_{1:t}, u_{0:t})} \tag{5}$$

5. **Re-Sampling:** Particles whose weight is below a certain threshold value are redrawn from heavy particles.
6. **Mapping:** Each position example ${x_t}^{(i)}$ and all observations $p({m_t}^{(i)}|{x_{1:t}}^{(i)}, z_{1:t})$ based on ${m_t}^{(i)}$ map is calculated.

The gMapping algorithm produces maps in occupancy grid type which supported by the ROS. This type of fusing data from different sensor sources can be held in a high resolution grid. However, running gMapping in large scale areas, where the number of grids is increasing, is considerably more costly than small areas. The occupancy rate of a grid $p(x, y)$ can be found as given by the number of grid points that the grid has in total in Equation (6).

$$p(x, y) = \frac{\#full}{\#full + \#empty} \tag{6}$$

4 Application and Experimental Results

The Android application and experimental results are discussed in this section.

4.1 Android Application

The developed Android application interface shows images from the camera on the robot in the top half of the screen, and map data generated using laser sensor in the bottom half. Teleoperating drive can be done with the joystick interface or accelerometer, which operates via the touch sensor. The maximum speed of the robot is set so that it can be gradually changed by four buttons at the bottom of the interface. An image of the interface is given in Figure 4.

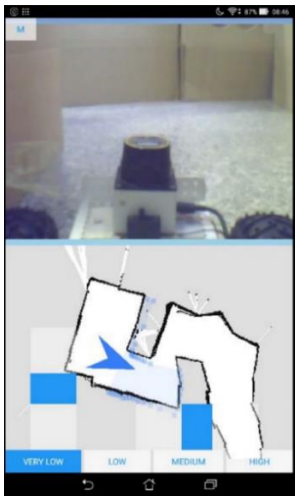

Figure 4. Android application interface.

4.2 Application

The robot platform was driven in the circular path given below and it was seen that the robot could successfully map.

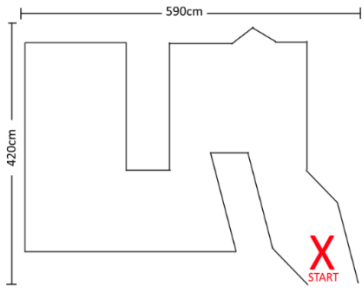

Figure 5. The sketch of the labyrinth where the robot was tested.

The video of this experiment is given on the link which can be found under Appendix A. As can be seen from the video, the operator who has not seen the robot has been able to perform the robot's journey successfully using only the touch senses given from the interface.

4.3 Experimental Results

When the resolution of the image taken from the camera on the robot and the Frame Per Second (FPS) values are changed, the loads occurring at the processing power and network traffic are observed as shown in Table 2. Since the map data is also transferred as an image, the load on the system is similar to the transfer of the webcam image.

Table 2. Web camera test chart.

Resolution	FPS	Processor Load (%)	Network Traffic (Kbps)
240 x 320	30	8.75	110
240 x 320	15	3.5	45
480 x 640	30	15	180
480 x 640	15	12.5	150
960 x 1280	30	27.5	330

5 Conclusion

The aim of this work is to provide a two-way communication between a mobile robot running on the ROS operating system and a mobile device running on the Android operating system.

As a result of the experiments, it is observed that the mobile robot can transmit data to the Android device via the ROS, and depending on the size of the transmitted data, the capacity of the microcontroller is observed and it is decided at which resolution and level the data should be sent for optimum load distribution.

The implementation of an application of ROS Android platform, which is still in development by the developers, and the successful completion of the problems will be a preliminary evaluation for those who want to work in this area and will make the development process very easy.

Moreover, the fact that mobile robot control can be carried out so easily via an Android device, which can now be found in every home, will save the need for additional control for robots to drive and save researchers who want to work in this way.

6 Acknowledgment

This paper was presented at the “Akıllı Sistemlerde Yenilikler ve Uygulamaları - ASYU2016” conference as a full text paper.

We are grateful to the Probabilistic Robotics Group of the Department of Computer Engineering at Yıldız Technical University for providing us with all kinds of material and spiritual support for our work.

7 References

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in ICRA workshop on open source software, vol. 3, no. 3.2, 2009, p. 5.

[2] A. Censi, "An ICP variant using a point-to-line metric," Robotics and Automation, 2008. ICRA 2008. IEEE Int. Conf. on., pp.19,25, 19-23 May 2008.

[3] G. Grisetti; C. Stachniss.; W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," Robotics, IEEE Transactions on , 23(1):34-46, 2007.

[4] “turtlebot_teleop - ROS Wiki”. [7.6.16]. http://wiki.ros.org/turtlebot_teleop.

[5] Dagu Wild Thumper 6WD All-Terrain Chassis, Silver, 75:1, [3.5.16] https://www.pololu.com/product/1561.

[6] A. Doucet, J. de Freitas, K. Murphy, and S. Russel, “Rao-Blackwellized particle filtering for dynamic Bayesian networks,” in Proc. Conf. Uncertainty Artif. Intell., Stanford, CA, 2000, pp. 176–183.

Appendix A

The experiment’s video link: https://youtu.be/rXAVSb0B5KE