Open Source based Low Cost Autonomous Robot with Object Recognition Intelligence

Matthew Cheng, Allen Chen, Albert Xu

Abstract

Object Classification, partnered with mapping, can have many advantages in Autonomous navigation compared to previous low-level image features for mapping. By using object classification to identify and classify landmarks, robots could better perform tasks such as mapping. Classifying objects in Human environments that are not artificial landmarks, such as furniture, appliances, or objects can increase the efficiency and effectiveness of mapping. In this paper we present a technique for classifying objects using a learned object classifier, and utilize this data for autonomous navigation. We also created an Open source ROS and Arduino based low-cost autonomous robot that could lower the cost and open a new way for scientific study and potential industrial applications.

Introduction

Autonomous navigation in robots has been a widely studied topic in the field of Robotics. Current techniques involve the use of Simultaneous Localization and Mapping(SLAM) Algorithms, which are useful for the building and updating of maps within unknown locations, while the robot retains information regarding its location. Utilizing a SLAM Algorithm, a robot is able to create a map of its surrounding areas and is able to move around in this environment [1]. Classifying objects into landmarks can have many advantages over current techniques of low-level image features for mapping, which are based off of a picture taken by the robot's camera. These low-level features may suffer from viewpoint dependent imaging conditions, and may create errors in maps. By utilizing higher-level features, such as shapes and objects, a robot will be able to receive a more specific clue to its current location. Integrated with Object Classification, Autonomous movement may become much more efficient than it has been before.

The history of the SLAM problem can be found in great detail in the works of Durrant-Whyte and Bailey [4] [5]. Recently, there is a trend of recognizing objects and signs through machine learning approaches. Object recognition using identifiable characteristics and features and machine learning techniques has been explored by many in the robotics community. Recognition and reading of door signs was proposed by Tomono *et al.* in [7]. It is a natural approach as machine learning technologies have major breakthroughs through fine-tuned Support Vector Machines(SVM) and deep learning technologies like Neural Network Learning with Convolutional Neural Network [9], for example, for a well-tuned multi-class SVM, the accuracy has reached 99.69% on recognizing hand-written digits [3]. The accuracy of Face recognition can achieve above 97% through deep learning [6]. Furthermore, the accuracy of Image Recognition has reached approximately 94% [8].

A work by Rogers III, *et al.* [2] integrated learned object classification with SLAM algorithms to program a robot to move around a hallway, guided by name placards mounted on doors. In this paper, object recognition is applied to door sign regions extracted from a resulting saliency mask. Rogers III, *et al.* [2] also incorporates SVM, specifically a combination of RBF kernel functions and Polynomial kernel functions, to build a binary classifier, in which positive represents instances of door signs and negative represents anything else that may look similar to a door sign. Rogers concluded that utilizing SVM for object classification helped relocate the robot when it was lost.

Rogers used character recognition for door sign detection, but did not make the robot autonomous.

Instead, the robot is tele-operated in the environment while its sensor and odometer data was logged by the Robot Operating System(ROS). Furthermore, the robot required a laptop to be mounted on it in order to run.

Encouraged by these great results from machine learning field, we could apply similar methods to our object recognition robot so that we could have a autonomous and intelligent machine in its map planning an navigation, an example being Google's driverless car.

Current techniques for testing robotic algorithms all involve costly and large robots. In Rogers' III paper, a modified Segway RMP- 200 is used, which is a fairly large and expensive robotic system. By improving the size and cost efficiency of such robots, future research may be much more streamlined. In this paper, we attempted to create a wireless and portable Arduino-based robot capable of object recognition via machine learning and autonomous navigation. We tried to integrate object recognition and autonomous navigation by creating an inexpensive and efficient robotic system.

Methods

We constructed a simple robotic system which could move around and scan its surroundings. The robot is shown in Figure 1.

The robot utilizes an Arduino Yun, Adafruit #1498, for wireless control of the robot. For the chassis, we utilized the Zumo Robot for Arduino v1.2, Adafruit #1639, which was controlled by an Aruino Uno board, Adafruit #50. A SHARP IR Distance Sensor, Adafruit #164 GP2Y0A21YK0F, was mounted on a Mini Pan Tilt kit, Adafruit #1967. All parts were connected with connection wires and a central half-sized breadboard, Adafruit #64. The IR sensor was connected and controlled by the Yun board, which was wirelessly connected to the computer to send data to the computer. The Zumo Chassis was designed to have an Arduino Uno as its controller, so they were a perfect fit for each other. However, since all the ports were occupied on the Uno and we also needed a WiFi function, the Arduino Yun was added on top of the Uno using a Master-slave I2C connection. The range sensor and scanning sensor are connected to the Yun. Furthermore, the Arduino Yun has a Atheros chip on it, which is a linux system that can access WiFi, which is used to transmit the data from all the sensors to the host computer, and allows the host computer to control the entire robot.

A diagram of the general structure of the robot can be seen in Figure 2.

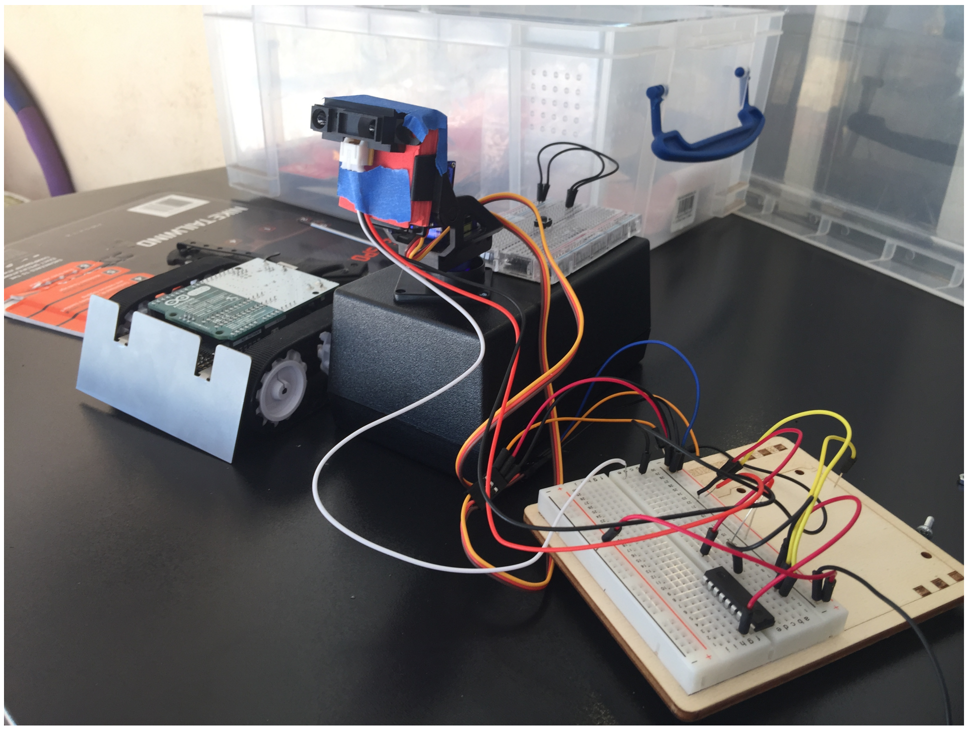

Figure 1. Image of Robot The Arduino Yun is mounted on top of the robot, with the Arduino Uno Board mounted on the Zumo Chassis(seen on the left). The IR sensor, mounted on the Mini Pan Tilt, is seen on top of the robot. Wires and a breadboard were used to connect the Yun, the sensor, and the Uno.

In order to program our robot, we utilized the Robot Operating System(ROS), and scikit-learn, a package for machine learning utilizing Python. ROS was used for the programming of the navigation and general control of the Arduino board.

ROS is a large software library that was used in making the autonomous robot. The libraries used were the control and navigation libraries, which control the robot given directions; the amcl library, which is a probabilistic localization system; the gmapping library, which is a laser-based mapping algorithm; and a sensor stream, which sends data from the range sensor on the Arduino to the computer that the ROS is installed on. By combining the amcl and gmapping libraries, SLAM is achieved. Then, by sending the data from the range sensor through the SLAM libraries, the approximate location of the robot would be discovered, and then using the navigation and control libraries, the commands to give the robot to get it to the target destination would also be discovered.

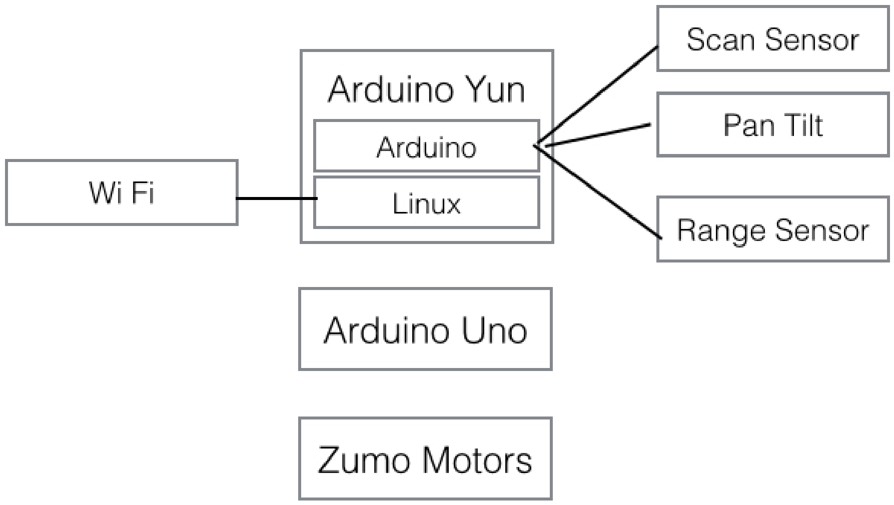

Figure 2. Diagram of Robotic System

0.0.1 Robot Range-Sensor Scanning Process

We began by first connecting the two Arduino boards via the I2C Wire Library, making our Arduino Yun board the master board, and making the Uno the slave board. We then set up the Robot Model via URDF, the sensors, and utilized the ros_control package for controlling our robot. The navigation package was used for robot navigation. We based our odometry model off that of Clark [10].

The Arduino Yun contained an ATmega32u4(32u4) and an Atheros AR9331. The 32u4 is the chip that controls the Arduino portion of the Arduino Yun, and the Atheros is a chip with built-in WiFi capabilities and a linux system.

The system operated by establishing a secure shell in the Atheros with a computer, accessing the python scripts installed. Running one of the scripts sent a key and value to the 32u4 through the Arduino Bridge class. Upon receiving the command, the Arduino then performed the scan, sending the range sensor's data back through the Bridge. When the Atheros received the data, it relays it once more to the original computer, which can then process the data and give new commands. This process is described in Figure 3.

Between the robot's hardware (Arduino) and software (ROS), there were several variables and components that had to be linked between the two. First, a URDF (universal robot description format) file

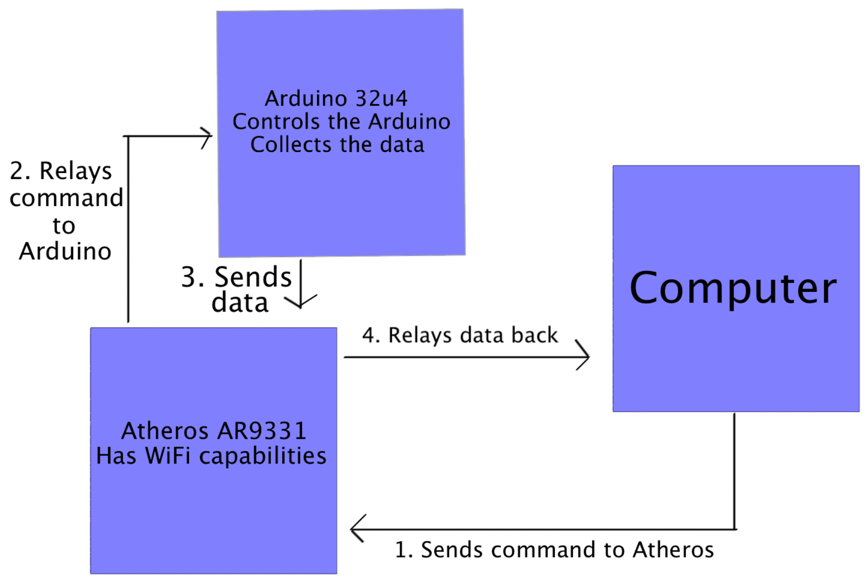

Figure 3. Diagram of Arduino system

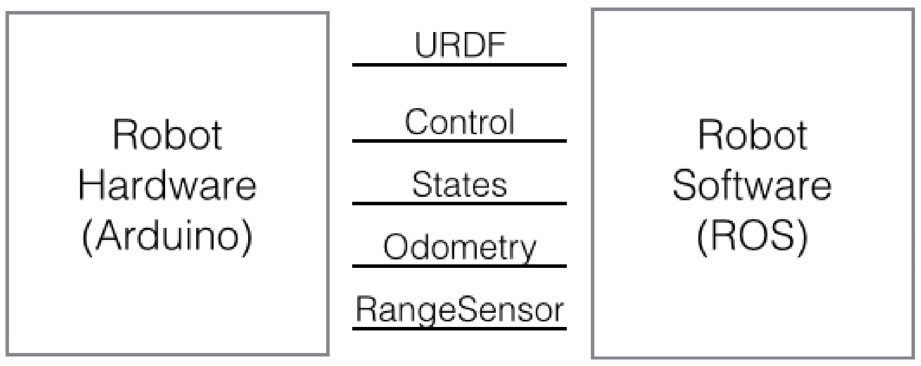

Figure 4. ROS and Arduino Integration

was made for the robot, encoding its size, shape, and physical properties for the ROS to use. When the communications between the Arduino and ROS are established, the ROS sends the control signals to the Arduino, while the Arduino sends sensor data back. Specifically, the ROS tells the Robot what speeds the motors should be set to, and what direction to point its range sensor, while the Arduino tells the

ROS the values on its range sensor, and the values on the odometer, both of which are used in the ROS's calculations. This process is detailed in Figure 4

The ROS calculates the orders to be given to the Arduino through SLAM (simultaneous localization and mapping). By scanning the robot's surroundings with the range sensor and matching it to a given map of the general area, the robot can localize itself. Then, it can also update the map if there is a small discrepancy. Also, the robot would be able to find the path to its destination through the map. All three of the above functions are included in the ROS library.

0.0.2 Machine Learning

In this paper we used Support Vector Machines (SVM) and its multi-class classification approach for recognizing various shapes when robotics navigate the routes. The advantages of SVM are that it is very effective in high dimensional spaces, esp. where the number of dimensions is greater than the number of samples. This is very useful in our case as we may not have enough training and testing data in real life. Since SVM has a great success in hand-written recognition with accuracy >99%, we took a similar approach by transforming and normalizing the scanned object shapes into 8x8 image files, then applied SVM for learning a shape model for real life shape detection.

We generalized our robot's peripheral to very discrete shapes: triangles, rectangles, and circles. As a robot uses its distance range detection device to scan its surroundings, it would only be able to perceive part of a shape's structure. To get the shape data for training and testing, we used an IR distance sensor to scan the distances between objects and the robotic so that we could have a shape of the object. This is explained in Figure 5.

We have the circle, rectangle and wall to represent typical shapes in real life during navigation. For each object, we scanned 30 data sets; 20 sets were be used for model training and 10 for testing and validation. The following algorithm applies when training and testing:

1. For each scanned shape data, parse the data into a temp file called "test.txt" with class labels like "circle"(0), "box"(1), "bag"(2), "wall"(10), and "unknown"(-1).

2. Convert the data from [radius, angle] into [x,y] coordinates and do basic filtering on noises.

3. Normalize the raw [x,y] data points into 8*8 image file for SVM training.

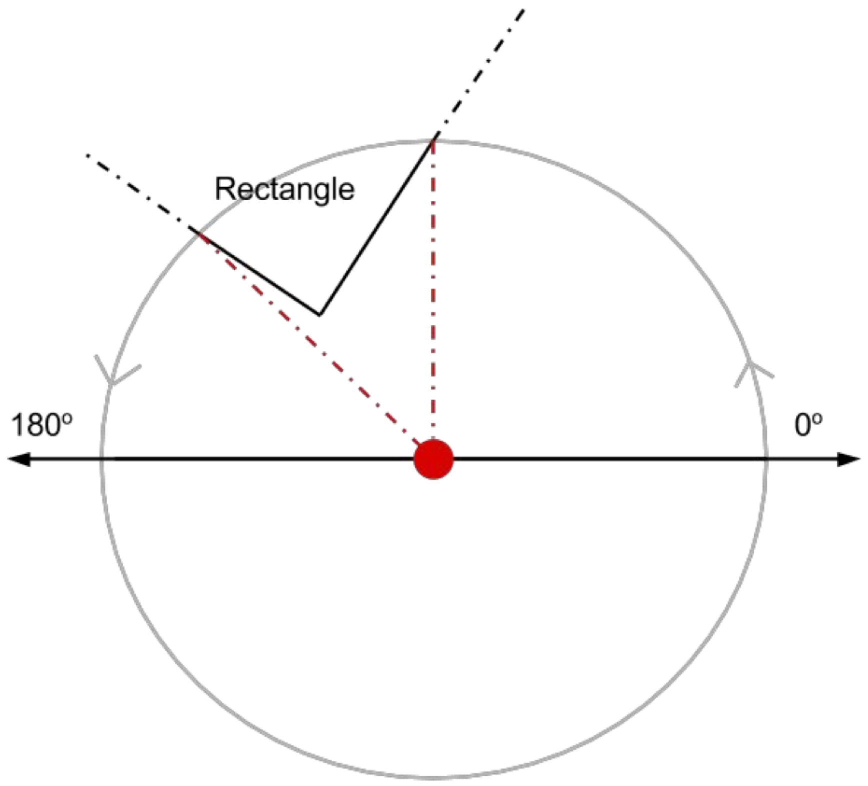

Figure 5. Object Shape Scanning

4. Apply SVM for training and testing, access accuracy and false positive rates, etc.

This algorithm is also explained in Figure 6.

In online application, we received the scanned distance data, applied the learned SVM model to make a predication about the shape label defined above, and then informed the robot about which shape it is detecting, so that robot could make smart planning and adjustments from here.

Results

We have successfully integrated ROS and Object classification, which has allowed us to retrieve raw data from the robot, analyze it, predict what object we have scanned, and inform the robot on further action.

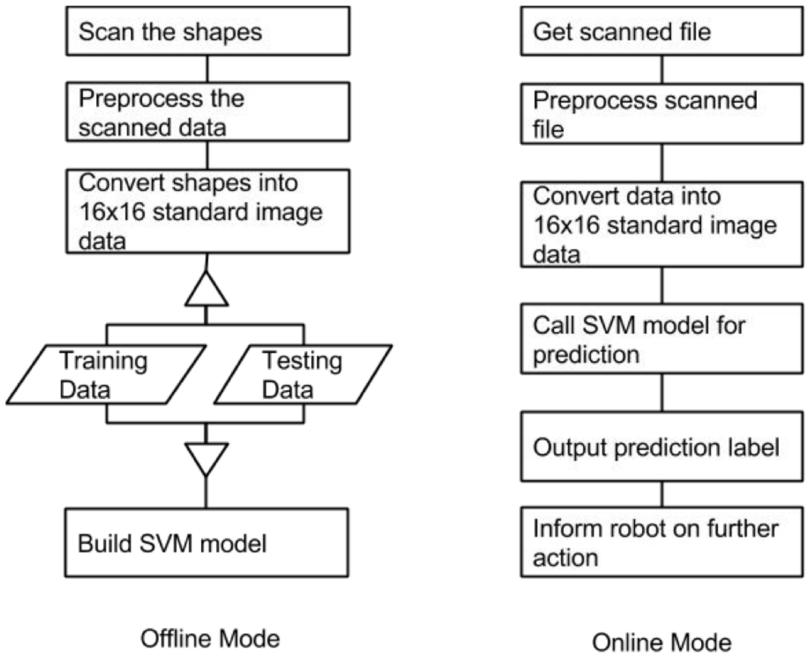
Scan the shapes
Preprocess the scanned data
Convert shapes into 16x16 standard image data
Training Data
Testing Data
Build SVM model
Offline Mode
Get scanned file
Preprocess scanned file
Convert data into 16x16 standard image data
Call SVM model for prediction
Output prediction label
Inform robot on further action
Online Mode

Figure 6. Training and Testing

Discussion

Conclusion

The next step in this field would be to improve the object classification such that more objects and landmarks can be recognized, so that not only shapes will be recognized by the robot. Robots could potentially recognize objects such as appliances, furniture, posters, or labels. These objects can reveal more information than just position for localization. For example, a robot could use object classification in a living room to distinguish between types of furniture to look for other equipment.

Bibliography

[1] Aznar F, Pujol FA, Pujol M, Rizo R, Pujol M-J *Learning Probabilistic Features for Robotic Navigation Using Laser Sensors.* PLoS ONE 9(11): e112507. doi:10.1371/journal.pone. 0112507, 2014

[2] Rogers III, John G., *et al. Simultaneous localization and mapping with learned object recognition and semantic data association.* Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011.

[3] Pedregosa *et al.* *Scikit-learn: Machine Learning in Python.* JMLR 12, pp. 2825-2830, 2011.

[4] H. Durrant-Whyte and T. Bailey. *Simultaneous localisation and mapping (SLAM): Part I the essential algorithms.* Robotics and Automation Magazine, June 2006.

[5] J. Folkesson, P. Jensfelt, and H.I. Christensen. *The M-space feature representation for SLAM.* IEEE Transactions on Robotics, 2007.

[6] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, *DeepFace: Closing the Gap to Human-Level Performance in Face Verification*, Lior Wolf, CVPR 2014

[7] M. Tomono and S. Yuta. *Mobile robot navigation in indoor environments using object and character recognition.* In International Conference on Robotics and Automation, 2000.

[8] Yangqing Jia *et al. Caffe: Convolutional Architecture for Fast Feature Embedding* arXiv preprint arXiv:1408.5093, 2014.

[9] http://neuralnetworksanddeeplearning.com/chap6.html

[10] Chris Clark. *COS 495 - Lecture 5 Autonomous Robot Navigation.* 2011. url: https://www.cs.princeton.edu/courses/archive/fall11/ cos495/COS495-Lecture5-Odometry.pdf.