Submitted on March 8, 2018 to CSI 693.
Volume ??, Number ??,

Multi-robot Autonomous Soccer using OpenCV and Hough Transform Gradients

Kaylee Yuhas and Brandon Marlowe

Abstract RoboCup is an international robotic soccer challenge that promotes robotics and artificial intelligence research. Every year, robotics researchers enter the competition in which their team of robots tries to play a game of soccer. In our project, we simplified this challenge and created a soccer game played between two TurtleBots, one acting as a kicker and the other as a goalie. This project centers around computer vision, using OpenCV and Hough Transform Gradients to track important objects in the scene.

1 Introduction

For a little over 20 years, members of the robotics community have been working on the challenge of creating robots that can play a game of soccer together. This international challenge, called RoboCup, promotes robotics and artificial intelligence research, with the ultimate goal to "develop a robot soccer team which beats the human world champion team." The advancements in robotics and artificial intelligence resulting from the RoboCup challenge have many applications within the field of mobile robotics.

We created a simplified version of RoboCup, using two robots, and programmed them to autonomously play a game of soccer. The robots are on opposing teams, where one acts as a goalie (defends the goal) and another acts as a kicker (tries to kick the ball into the goal). Our project focused primarily on object tracking, but also involved voice recognition. This paper is split into four main sections: Related Work, Project Details, Results, and Work Distribution. We will discuss work already done in the area, specifics on our project's implementation, the results of our project, and who did what on this project. Lastly, we will conclude our paper and discuss lessons learned and our ideas on future work.

2 Related Work

Object tracking is an important area in computer vision in which an object is located in every frame of a video. Tracking objects is typically done using a feature of the object, such as its color or edges [10]. Tracking using edges can have many benefits over color-based tracking algorithms. For example, environmental factors, such as harsh sunlight or low-light conditions, could affect the perceived color of the ball, which would interfere with the color-based tracking algorithms, but would not affect the edge-based tracking algorithms. Further, any round object could be used as the ball in the RoboCup challenge, whereas a color-based tracking algorithm requires that the ball be the color specified in the code (unless the code were to change or take the ball's color as input). For these reasons, many RoboCup competitors detect and track the ball without using the color information [6].

Previous work has used both the object's edges and color for the recognition task [5]. This combination allowed the object, such as the RoboCup ball, to be recognized regardless of illumination. A more recent technique has been able to use just the ball's shape to locate it within an image [6]. This is done by representing the image as a histogram of the orientation of gradient vectors for each pixel. The technique requires a large number of recursive calls for each frame, which could make this algorithm computationally expensive, unless it is optimized using hardware.

The Hough Transform algorithm, developed in 1959, is a well known method for detecting simple shapes within images. While the method does identify shapes well, it is not particularly fast. The issue of speed led to several extensions of the original algorithm, one of which is sometimes referred to, unofficially, as the Hough gradient method [3]. Carolyn Kimme, Dana Ballard, and Jack Sklansky of the University of California, in 1975, took the original Hough Transform, and incorporated the direction of the gradient of the pixels in the image to eliminate noise. By doing so, the processing time to identify circles within images improved greatly.

Kwok and Fox proposed a new approach that estimates the ball's location using multiple Kalman filters and Rao-Blackwellised particle filters to sample various actions that could occur between the ball and the robot and the ball and the environment [4]. This implementation resulted in better real-world tracking compared to vanilla Kalman filters.

Schulz et al. used particle filters and a variant of Joint Probabilistic Data Association Filters to track and measure multiple different objects, such as nearby people [7]. The researchers also used a probabilistic motion model to estimate the motion of the participants. A robot using their tracking algorithm was able to successfully track a person for 102 meters over the course of 147 seconds. While we will likely not use this algorithm in our own work, it gave impressive results.

3 Project Details

3.1 Overview For this project, we programmed two robots to autonomously play soccer with each other. One robot acts as a kicker, and the other robot will act as a goalie. The kicker tracks both the red soccer ball and a green circle placed over the goalie. The goalie only tracks the red soccer ball.

The kicker will begin in some arbitrary location, in any area, representing the field. The goalie will be placed in any location representing the goal. For best results, the goal should be between 1 or 2 meters wide. Our project is easily transportable, as it does not rely on a map. In other words, as long as you have two TurtleBots, a red soccer ball, and a green circle to hold over the goalie, our project will work.

The game will start once the robots hear the word "start." If the kicker does not see the red soccer ball, it will spin counterclockwise until it sees the ball in its frame. Upon seeing the red ball, the kicker will approach it, collecting the red ball in its U-Guide attachment, attached to the base of the TurtleBot. This attachment can be seen in Figure 1 below.

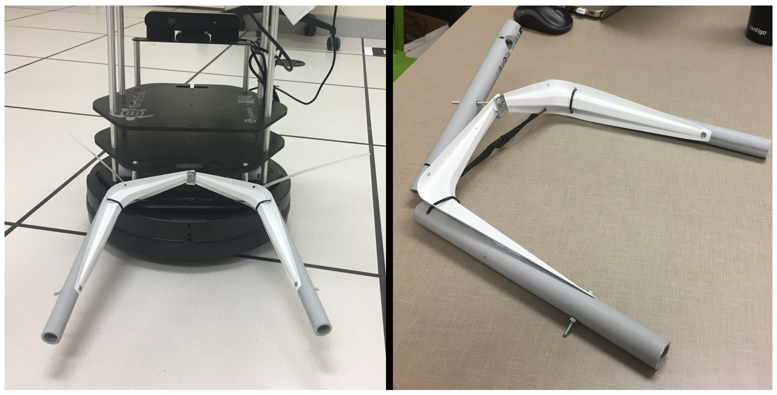

Figure 1: U-Guide attachment used on the kicker robot to capture the red soccer ball.

After collecting the red ball, the kicker then begins to track the green circle, representing the goalie's current position. If it does not see the green circle, the kicker will again rotate counterclockwise. Once the green circle enters the frame, the kicker quickly approaches it, stopping 2.5 meters away from the green circle (and therefore the goalie). When the kicker suddenly stops, the red soccer ball still has some forward momentum and rolls into the goal, which may be blocked by the goalie.

When the goalie starts the game, it will begin to search for the red ball. If it does not see the red ball, it will not rotate. However, if it does see the red ball and the ball is greater than 1.5 meters away, the goalie will rotate to align itself with the red ball. Once the ball is within 1.5 meters of the goalie, the goalie will move forward and in whichever direction it sees the ball approaching from (either the left, right, or center). The goalie moves forward between 0.75 and 1.25 meters, so the goalie and kicker stay a safe distance from each other. After moving forward and potentially to the side, the goalie then pauses for 2 seconds and moves back to its initial starting location.

The game ends when both TurtleBots hear the voice command "stop." Upon hearing this, both TurtleBots stop moving and are no longer able to track the ball. A game can either be stopped after the first attempt at scoring a goal or after an arbitrary number of attempts. If the game is not stopped after the first attempt, the kicker will go back to tracking only the red ball (until it has captured it, at which point, it will head toward the goalie instead).

3.2 Implementation The bulk of our project was centered around the use of computer vision. Both the kicker and goalie robots need to track the red ball throughout the game. The kicker must also track the green ball upon capturing the red ball. Beyond just tracking the ball(s), the robots must also adjust their behavior and positioning in real-time based on the location of the specified balls. All of these actions cannot be accomplished well and autonomously without the use of computer vision.

We have chosen to use the OpenCV library for our real-time vision tasks. OpenCV is open-source, extremely well documented, and has a large community base [9]. Additionally, ROS supports OpenCV extremely well, making software integration a non-issue. The ROS "vision_opencv" stack provides two packages, "cv_bridge" and "image_geometry." The former creates a communication bridge between ROS messages and OpenCV, and the latter contains methods for handling image/pixel geometry. We utilized the Hough gradient method to detect the red ball and green circle and identify their centers. The Hough gradient method is an extension of the Hough Transform, which was created to identify classes of simple shapes [1]. The Hough gradient method works by first detecting edges within an image. Then, the local gradient for all non-zero points in the edge image is considered. Candidate centers are selected from points that fall within a threshold range and the candidate centers are sorted in descending order based on their determined value. Next, for each of the candidate centers, only the non-zero pixels extending from them are considered and sorted according to their distance from the candidate center. Finally, a single radius, best supported by the non-zero pixels, is chosen and the circle is formed. Locating the center of the red soccer ball was greatly beneficial to the kicker and goalie so that they could align themselves with the ball at all times.

All of the distance calculations used in this project were found using only the RGB camera. In fact, it was only dependent on various properties of the TurtleBot (i.e. RBG camera focal length, height above the floor, etc) and the ball's diameter. The values used with the AIR lab TurtleBots and the provided red ball are shown in Table 1 on the next page. If this project were to be used by another team of researchers with different TurtleBots and a different ball, these values would have to be adjusted in the code to get an accurate distance measurement. Further, because the project only relied on the RGB camera, the depth camera was not used at any point, preventing possible interference between the kicker and goalie.

Because speech recognition was such a minimal component of our project, we did not require robust natural language processing capabilities. PocketSphinx is a

RGB camera focal length (mm)	138.90625
RGB camera height (mm)	300.0
RGB camera image height (pixels)	480.0
RGB camera image width (pixels)	640.0
Ball Diameter (mm)	203.2

Table 1 RGB camera and ball specifications.

lightweight, open-source speech recognition library designed by Carnegie Mellon University intended for use on embedded systems, and uses minimal computational resources. PocketSphinx supports multiple languages for development, including Python and C++, which integrates well with the supported languages in ROS [8]. Prior to settling on PocketSphinx, we compared several other open-source speech recognition libraries, which included Kaldi, HTK, and Julius. According to a recent paper, Julius is less accurate than PocketSphinx, so we could easily remove it as an option [2]. While Kaldi is significantly more accurate than HTK, Julius, and PocketSphinx, after attempting to install and briefly use Kaldi, it was evident the library is far more complex and resource demanding than PocketSphinx. Finally, we attempted to use HTK, and found the setup to be much more time consuming than the other libraries we considered. Additionally, HTK, similar to Kaldi, requires more computational resources than PocketSphinx. Because of the aforementioned reasons, we have decided to use PocketSphinx as the speech recognition library for our project.

3.3 Issues Encountered and Solutions As previously discussed, if the kicker has not yet captured the red ball, it only tracks the red ball and does not yet track the green circle. However, upon capturing the red ball, it no longer tracks the red ball and only tracks the green circle. This was done to prevent the laptop from overheating.

We also had many issues when trying to create an object that the kicker could use to track the goalie. Our initial plan was to attach a blue styrofoam ball to the top of the goalie TurtleBot so that the kicker could track it. Due to limitations with the TurtleBot's RGB camera and reflections from the ball's paint, the kicker was not able to detect the blue ball in the frame. We then tried printing out a blue circle, but the kicker was still unable to detect it. From here, we switched to using a green ball and circle. When the green styrofoam ball was attached to the goalie TurtleBot, the kicker could detect it at times, but it was very inconsistent. The resulting circles found using the Hough gradient method were often much smaller than the actual ball itself (there was a poor radius approximation) and the algorithm would often be unable to find any circles at all. After much trial and error, we found that the TurtleBot can consistently locate a green circle if held up on a laptop screen above the goalie itself. A photo of the image we used is shown in Figure 2. We suspect this worked well because there is a harsh contrast between the green circle and the white background which was easily detected by the camera and used in the Hough gradient method.

The largest problem we encountered was designing the behavior of the goalie. Initially, we specified various positions in a map of the AIR lab representing the

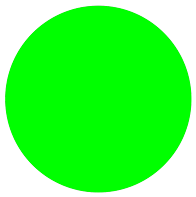

Figure 2: Green circle held over the goalie during the game.

left, right, and middle of the goal. The goalie would autonomously navigate to the goal position based on which direction it saw the ball coming in. For example, if the kicker robot and the ball were approaching on the left side of the frame, the goalie would navigate to the left goal position. This approach presented a few issues. Firstly, the goalie would often get lost, rotating for a long while trying to estimate its location and navigate to its destination. Secondly, the distance the goalie was moving was not very far and we decided that publishing Twist messages to control the robot's linear and angular velocities would be more appropriate. Therefore, in the second solution, Twist messages were created, but we still relied on the goalie's current pose to know when to stop moving. For example, if the goalie wanted to move to the left side of the goal, it would first start to rotate left and would only stop once it had turned 90 degrees counterclockwise from its starting point. This was often very error-prone due to the uncertainty in the robot's pose (found by subscribing to the /amcl_pose topic).

Therefore, in our final solution, we did not take the robot's pose into account at all (thereby eliminating the need of the depth camera). Instead, we simply had the goalie move forward and in the direction of the red ball, pause, and then reverse its movement to return to its initial location. This worked well enough for the goalie to block many of the balls approaching the goal. However, sometimes if the ball moved too quickly and too close to the robot, it would spin in place very quickly. Further, due to the uncertainty in its movements, the goalie did not always end up in its initial position after reversing. Over time, the goalie would inch further and further forward, which could be dangerous if it gets within range of the kicker.

4 Results

Given the limited time we had to test the whole system, each of the robots performed their tasks rather well. To test the behavior of the goalie and the kicker, we tested them each individually and together.

First, we tested the kicker on its own, having it collect the red ball and then move toward the goal (designated by the green circle on a laptop placed on a chair) and kick the ball into the goal. An image of this setup can be seen in Figure 3. In nearly all of the cases, the kicker collected the red ball and rotated toward the goal, smoothly approaching it and releasing the ball. However, not all of the attempts resulted in the ball entering the goal area. The reason for this is that the floors in the AIR lab were quite uneven, causing the ball to drift from what would have otherwise been a good path. Further, we often had to push the ball back into the U-Guide attachment as the kicker was rotating for the same reasons.

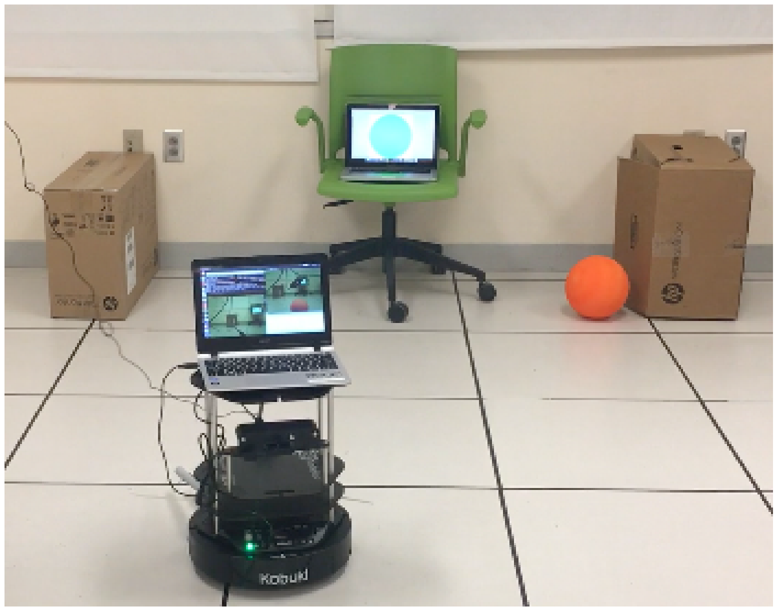

Figure 3: Kicker testing setup.

Next, we tested the goalie on its own, rolling the red ball towards it and noting its behavior. When the ball was rolled at the speed at which the kicker would "kick" it at (roughly 0.6 meters per second), the goalie blocked 4 of the 6 attempts. But, once the ball was moving at much higher speeds, the goalie's performance worsened. However, this was not an immediate concern for us because the kicker could not travel at these high speeds, so it would not be an issue during the actual game.

Lastly, we tested the full system in which the kicker and goalie are both playing with one another. Both the kicker and goalie could consistently recognize and track the red ball, and approximate the distance from it quite well. There were also no collisions between the two, as they kept a safe distance from one another. They also both understood the voice commands well and we never had to repeat the commands more than two or three times.

5 Work Distribution

Both team members worked equally on this project. Brandon created the U-Guide attachment which greatly improved the kicker's ability to keep the red ball close to it as it moved. Without this attachment, the code to rotate the kicker would have been much more complex because the ball would not rotate with it. Further, Brandon also did a lot of testing in Gazebo. Kaylee was using a VM to run Ubuntu 14.04 and Gazebo was often too slow to be useful or would crash entirely on her computer.

Most of the testing Kaylee did was on the TurtleBots themselves once she was confident with the code.

Both team members heavily contributed to the code, as demonstrated by their commit history. A link to the public repository can be found at the following link: Main Repo. Note that this is the second repo we used in this project. There were commits in the first repo, but its file structure and size made it difficult to work with, leading to the switch to a second, simpler repo. The first repo (no longer used) can be found here: Earlier Repo.

Throughout the development process, both team members found bugs and various issues, fixed them, and added new functionality. Kaylee fixed various bugs and issues such as the kicker robot not correctly rotating to search for the goalie upon collecting the red ball. Further, the kicker would often incorrectly move backward after collecting the red ball, which she fixed as well. Brandon also fixed various bugs and issues, such as finding runtime errors that prevented the kicker code from running. Also, he helped simplify the final design on the goalie blocking behavior.

6 Conclusion

Overall, we both feel this project was successful and demonstrated our understanding of the main concepts within robotics (computer vision, voice recognition, etc). Additionally, the development process was very enjoyable because of the interactive nature of the work, and gave us the opportunity to expand and apply knowledge gained from previous class projects. Seeing the kicker and goalie's behavior improve with each change made to the source code along the way was very exciting for both of us. In the end, we wish we had more time to further develop the complexity of both the kicker and goalie's behavior.

7 Future Work

There is still additional work that can be done to make the goalie and kicker's behavior more robust. As discussed in the "Issues Encountered and Solutions" subsection, there were many designs of the goalie's blocking behavior. While we finally went with a simple design that worked well in our tests, we would like to eventually add some artificial intelligence techniques. We would like the goalie to be able to predict the trajectory of the ball with high accuracy and act accordingly. Similarly, we would like to apply artificial intelligence techniques to the kicker as well, having it predict the goalie's behavior so that it can make an optimal kick.

8 Acknowledgements

We would like to thank Dr. Shiqi Zhang for always letting us in the lab whenever we asked, even over the weekend. Further, his guidance was very helpful throughout the project.

References

[1] Bradski G., and Kaehler A., “Learning OpenCV,” 2013. Learning OpenCV: Computer Vision in C++ with the OpenCV Library (3rd ed.). O’Reilly Media, Inc.. 4

[2] Gaida, C., Lange, P., Malatawy, A., Petrick, R., Proba, P., & Suendermann-Oeft, D., “Comparing Open-Source Speech Recognition,” 2014. 5

[3] Kimme C., Ballard D., Sklansky J., “Finding circles by an array of accumulators.,” 1975. Commun. ACM 18, 2 (February 1975), 120-122. DOI=http://dx.doi.org/10.1145/360666.360677. 2

[4] Kwok C., Fox D., “Map-Based Multiple Model Tracking of a Moving Object.,” Nardi D., Riedmiller M., Sammut C., Santos-Victor J. (eds) RoboCup 2004: Robot Soccer World Cup VIII. RoboCup 2004. Lecture Notes in Computer Science, vol 3276. Springer, Berlin, Heidelberg. 2

[5] Lovell N. , “Illumination Independent Object Recognition,” Bredenfeld A., Jacoff A., Noda I., Takahashi Y. (eds) RoboCup 2005: Robot Soccer World Cup IX. RoboCup 2005. Lecture Notes in Computer Science, vol 4020. Springer, Berlin, Heidelberg. 2

[6] Mobalegh H., Helgadottir L.I., Rojas R., “Shape Based Round Object Detection Using Edge Orientation Histogram.,” Behnke S., Veloso M., Visser A., Xiong R. (eds) RoboCup 2013: Robot World Cup XVII. RoboCup 2013. Lecture Notes in Computer Science, vol 8371. Springer, Berlin, Heidelberg 2

[7] Schulz D., Burgard W., Fox D., Cremers A., “Tracking Multiple Moving Targets with a Mobile Robot Using Particle Filters and Statistical Data Association,” 2001. 2

[8] Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R., Ng A., “ROS: an open-source Robot Operating System,” 2009. ICRA Workshop on Open Source Software. 5

[9] “vision_opencv,” wiki.ros.org, March 1 2018. 4

[10] Yilmaz A., Javed O., Shah M., “Object Tracking: A Survey,” 2006. ACM Comput. Surv. DOI=10.1145/1177352.1177355. 2

Yuhas
Department of Electrical Engineering and Computer Science
Cleveland State University
2121 Euclid Ave.
Cleveland, OH 44115
k.yuhas62@vikes.csuohio.edu

Marlowe
Department of Electrical Engineering and Computer Science
Cleveland State University
2121 Euclid Ave.
Cleveland, OH 44115
bpmarlowe@gmail.com