Helping BAXTER See
CS 543 Final Report

Brandon Boyce and Jessica Mullins

May 12th, 2014

Abstract

The goal of this project is to explore the use of computer vision in the robotics field. In this project, a BAXTER robot is used to accomplish a variety of tasks, with computer vision guiding his actions. The two tasks completed in this projected required thorough knowledge of a variety of computer vision techniques and of the Robot Operating System (ROS). These tasks are described below in order of increasing difficulty.

1 Introduction

Our project aimed to investigate the potential of computer vision in robotics. Specifically, we wanted to explore using computer vision to assist a robot in identifying objects in a scene. Providing robots, both flight-capable and ground-based, with a method to identify objects in a scene could lead to a variety of unique applications including automated crop harvesting, advanced search and rescue techniques, and assistance with building maintenance.

We began our project by finding and reviewing some literature in this research area. We looked into Computer Vision research, specifically targeting research in the area of object detection via color. We found [1] and [2], which present methods for identifying objects in traffic using color and shape. After looking at those papers we decided to use a basic, simple color based object detection algorithm. There are many tutorials online that demonstrate this algorithm. We chose to adapt our code from the tutorial in [3]. We used OpenCV for object detection and recognition. We have used the OpenCV tutorial page as a reference for getting familiar with using OpenCV.

The BAXTER robot from Rethink Robotics was utilized in the project [4]. BAXTER is a robot that has two arms, each with 7-degrees of freedom, a front facing camera, and a variety of end effectors that can be exchanged based on the requirements of the current task. BAXTER comes equipped with cameras on his head and both of his hands. The camera in his right hand was used for this project. For more information about BAXTER see [4].

In order to learn how to use BAXTER, we began identifying some helpful ROS tutorial websites such as [5] and [6]. These resources assisted us in implementing the robotic control portion of the project.

To explore computer vision in combination with ROS we identified two goals for our project. Each goal includes a task for BAXTER to complete. They are as follows:

1. **Dunk a ball:**

   Make BAXTER detect a ball, grab the ball, detect a hoop, calculate the distance to the hoop, move the ball over the hoop, and drop the ball through the hoop.

2. **Sort multiple balls by color:**

   Make BAXTER distinguish between differently colored balls, then, in a similar process to Task 1, sort the balls by placing them in the appropriately colored buckets.

2 Methods

Our approach was based on using color detection techniques to identify objects [7]. The colors of the balls were known beforehand for this project. After determining which color of ball that we would make BAXTER look for, we made BAXTER randomly scan a "search area" in which the ball was

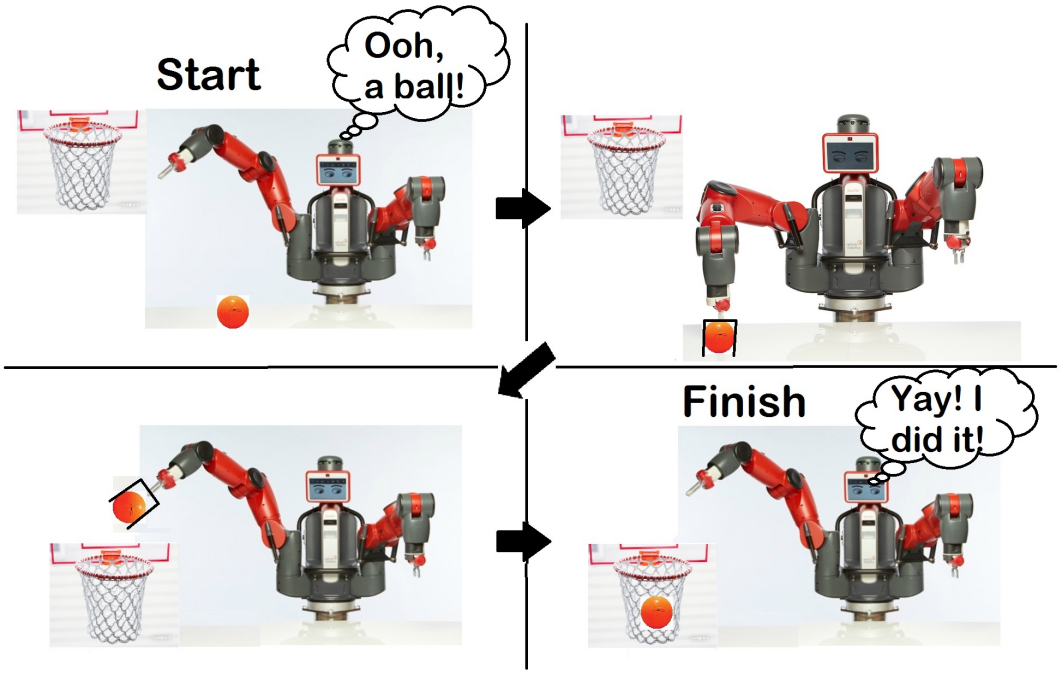


Figure 1: Proposed Task 1, visualized in cartoon graphics

placed. After finding and grabbing the ball, BAXTER was tasked with finding either the hoop or the appropriate bucket in which to place the ball. After the hoop or appropriate bucket was found, BAXTER would have to determine the distance to the new target using camera calibration techniques along with knowledge of similar triangles [8]. A more detailed explanation of our methodology is outlined here:

1. Detect the ball [7]:
   (a) Move BAXTER's arm to some initial position.
   (b) Look for the ball by scanning the environment through the motion of BAXTER's right hand.
   (c) Get images from the camera located on BAXTER's right hand.
   (d) After getting the image, convert the RGB (Red, Green, Blue) image into an HSV (Hue, Saturation, Value) image.
   (e) Next, convert the HSV image to a binary image using the OpenCV *inRange*() function. Using this function allows us to specify which color to look for in the image.
   (f) Using our binary image, find the largest contour, or white areas, in the image. Our desired object is the largest contour.
   (g) The center of the largest contour gives us the location (x,y) of the object.
2. Center the ball in the image:
   (a) Chose a "alignment point" within in the image as the point where the center of the ball should be aligned.
   (b) Using the "alignment point" described in the previous step, the image was divided in a grid-like fashion into four quadrants. The first quadrant was the top-right portion of the image and subsequent quadrants were numbered in a counterclockwise manner. After determining which quadrant the

center of the ball was currently in, move BAXTER's right hand closer to the "alignment point".

(c) Repeat the previous step until the (x,y) location of the center of the ball was within some tolerance of the (x,y) location of the "alignment point".

3. When the ball is at the "alignment point" in the image, move BAXTER's hand appropriately, and grab the ball.

4. Change the desired color to the color of the hoop or appropriate bucket.

5. Repeat Steps 1 and 2 to find the hoop or appropriately colored bucket

6. Using the algorithm from [8] calculate the distance that the hoop or appropriate bucket is away from BAXTER.

   (a) Calibrate the camera using the known size of the target object (s) and a preset distance (z) from the camera. After placing the target object at z, take a picture of the target and find the apparent width in pixels (d). Then the focal length of the camera will be equal to $d * z/s$ [8].

   (b) At run time, get the image of the target and compare the current apparent width of pixels (d') to the calibrated width of pixels. Using triangle similarity we will get current distance equal to $d' * f/d'$ [8].

   (c) We can combine the distance information with the (x,y) coordinates from part A to get the hoop's location in 3D.

7. After BAXTER has determined the distance to the hoop or appropriate bucket, we navigate his arm over the object, and have him drop the ball.

8. In the "Sort multiple balls by color" task, repeat the above steps until all balls are sorted.

It should be noted that the scanning motion mentioned in Step 1b took place within a "searching area" that was predetermined based on limits of BAXTER's limbs and the dimensions of the table being used to hold the balls. There was a unique "searching area" for the balls, the hoop, and the buckets, so the "searching area" through which BAXTER's limbs were moved in Step 1b was based on the current object that BAXTER was looking for.

It should also be noted that an iterative approach was used in Step 2 instead of a "one-motion" approach in order to avoid moving to the center of an improper bounding box. Improper bounding boxes could be caused when less than the entire object is within the image or when BAXTER's grippers are partially obstructing the view of the object as his hand moves. The use of an iterative approach allows for motion towards these improper bounding boxes, and, as the object becomes fully visible, the appropriate bounding box should be created, suggesting that the center of the object should be visible. Therefore, because the use of the iterative approach should be able to guarantee that the center of the object is visible, this approach should also be able to guarantee that the center of the object gets aligned with the "alignment point" discussed in Step 2.

When attempting to grab the ball, the "alignment point" was chosen to be closer to the top of the image than the bottom of the image. This choice was made because BAXTER's hand camera is below the hand grippers. Therefore, because the "alginment point" was near the top of the image, after the ball was grabbed it would only obstruct, at most, the top half of the image. This would then leave the bottom half of the image unobstructed when attempting to find the hoop or appropriately colored bucket. Furthermore, when attempting to center the hoop or appropriate bucket, the "alignment point" was chosen to be closer to the bottom of the image since the top half was obstructed by the ball by this point.

Inverse kinematics were used to move BAXTER's limbs to different (x,y,z) locations while ensuring that his grippers were consistently in the desired orientation. When determining the distance to objects, precautions were taken to ensure that the robot would not harm itself. For this reason, the distance to the ball was predetermined to ensure that BAXTER would not attempt to make his hand go through the table. Furthermore, if the distance to the hoop or appropriate bucket was calculated to be further than the maximum reach of BAXTER's limbs, we simply made BAXTER move to maximum extension.

(a) Target is the pink ball **(b)** Conversion to HSV **(c)** Conversion to binary

(d) Identify largest contour **(e)** Ball detected

Figure 2: Images showing the process of identifying the ball

3 Results

In this section, we first begin by discussing the initial and improved object detection results. We then discuss the results from the two tasks described in Sections 1 and 2.

3.1 Object Detection and Recognition

3.1.1 Initial Results

The following is an example of our initial object detection on a single image.

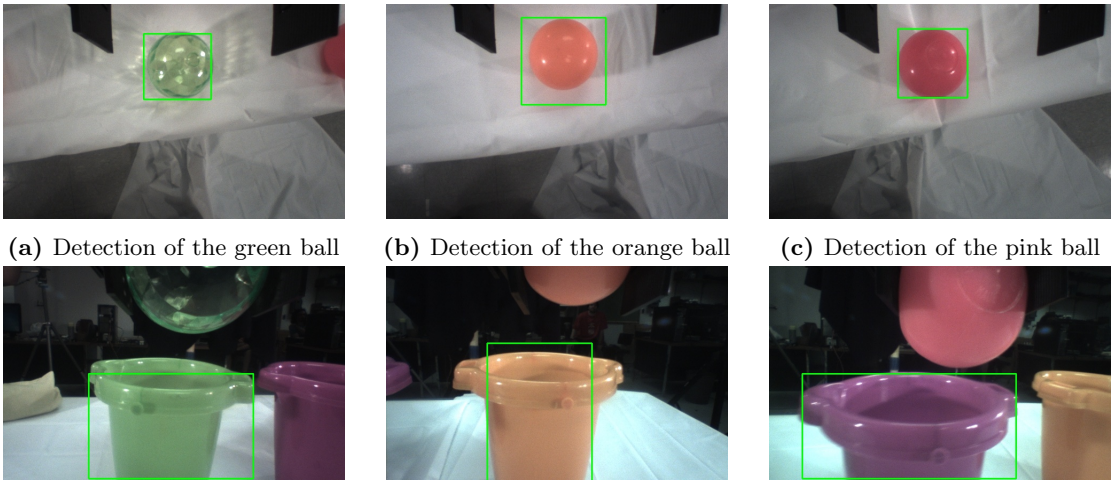
Figure 2a is the image that will be processed, and the desired target is the pink ball. Our first step is to convert the RGB image into an HSV image, as seen in Figure 2b. Then, we have convert the HSV image to the binary one seen in Figure 2c using the OpenCV *inRange*() function. Next, we look for the largest contour and suppress all others. The contours that were created for the pink pen have now been removed, as seen in Figure 2d. Finally, we get the center (x,y) position for the largest contour and use that as the center of our object. We then draw a bounding box to show the detected object, as seen in Figure 2e.

We have also created a demo video to show real-time object detection which can be seen at https://www.youtube.com/watch?v=hYJn1Z8A_bs&feature=youtu.be

As you can see in the video we are able to detect the pink ball, although at times our bounding box does not entirely encompass the ball. We reviewed our code and looked at how we could improve our approach. After troubleshooting and reviewing the computer vision literature cited above, we were able to improve our object detection.

3.1.2 Final Results

Our project implemented an object detection algorithm based on color. In comparison to the initial results, we were able get the bounding box to completely enclose the target object. This was accomplished through improving the range of HSV values we were using when converting the HSV image to a binary image. In addition, we were able to detect colors other than the pink color used in the initial results, namely orange & green. The final results are displayed below in Figure 3.



(a) Detection of the green ball **(b)** Detection of the orange ball **(c)** Detection of the pink ball

(d) Detection of the green bucket **(e)** Detection of the orange bucket **(f)** Detection of the pink bucket

Figure 3: Images showing the successful detection of each colored ball and bucket

3.2 Depth Determination

In order to determine how far BAXTER's hand would need to extend, depth determination code was developed. This code was used to determine the depth of the hoop and the colored buckets for the "Dunk a ball" and "Sort multiple balls by color" tasks, respectively.

To determine the depth of an object in an image, we followed the methods outlined above in Step 6 of Section 2 [8]. Figure 4 shows some basic results of the depth detection code. The coordinates shown on the image show the center of the box in (x,y,z), where z is the depth.

Figure 4a shows the results of the depth detection when the ball is placed at the calibrated starting distance of 45 cm from the camera. The accuracy of the code can clearly be seen since the z value is equal to 45. As we move toward the camera, Figure 4b, the depth is re-calculated, and the z value becomes smaller. Similarly, as we move away from the camera, Figure 4c we can see the z value becoming larger.

3.3 Dunk a ball

Using the methodology described in Section 2, we were successful in making the BAXTER robot dunk the ball in the basketball hoop. Furthermore, we were able to make the BAXTER robot detect the ball no matter where the ball was placed in the "search area". The robot was also able to detect the ball with near perfect accuracy, assuming that there was a ball in his field of view. When considering safety, the robot was able to pick up the ball and dunk it without damaging itself.

The errors that were addressed during this portion of the testing included false positives, failed grabs, and blocking of the hand camera after the ball was grabbed.

Because it was noticed that most of the false positive errors were due to small objects, like cords on the floor, we placed a threshold that the maximum contour had to be larger than in order to be considered the contour of a ball.

To minimize the number of failed grabs, we adjusted the lighting and HSV values to improve the quality of our bounding boxes. Additionally, we reduced the tolerance distance between the center of the ball and the "alignment point" described in Step 2 of Section 2 to maximize the chances that the grippers would be able to reach around the entire ball.

In order to prevent obstruction of the hand camera, the "alignment point" was chosen to be near the top of the image to ensure that, at most, only the top half of the image was blocked when attempting to find the hoop or appropriately colored bucket. This was further discussed in the end of Section 2.

(a) Depth calculation from the calibration distance of 45 cm **(b)** Depth calculation when moving the ball closer to the camera **(c)** Depth calculation when moving the ball further from the camera

Figure 4: Images showing the depth determination of a ball. Images were zoomed from originals to allow for values to be visible.

A video showing the BAXTER robot successfully dunking a ball can be found at http://youtu.be/zMfi38ndXvY. This video shows two view points, the view point of the authors and the view point from BAXTER's hand camera.

3.4 Sort multiple balls by color

Using the methodology described in Section 2, we were successful in making the BAXTER robot sort three differently colored ball into three correspondingly colored buckets. Similar to Section 3.3, we were able to make the BAXTER robot detect the balls and the buckets no matter where they were placed in their "search areas". Again, the robot was also able to detect the balls and buckets with near perfect accuracy, assuming that a ball or bucket was in his field of view. When considering safety, the robot was able to pick up the balls and place them into the buckets without damaging itself.

The errors that were addressed during this portion of the testing included early positives when detecting the buckets and difficulty noticing the transparent green ball that was used.

It was noticed that occasionally the camera would notice a bucket when BAXTER's arm would lift up from it's downward facing "find ball" position to it horizontal "find bucket" position. Because BAXTER's arm would always move to a starting position before beginning its random search, occasionally the appropriate bucket would end up approximately aligned with the "alignment point" discussed in Section 2. If this happened, there would be a early positive, which would make BAXTER lunge forward and drop the ball after returning to his starting position because he thought the bucket was already aligned with the "alignment point". This was corrected by ignoring the object detection code while BAXTER was moving to his starting position.

The green ball that was used in this project was slightly transparent and reflective. This caused difficulties when using the HSV image approach because, as BAXTER's arm moved, it would occasionally cast a shadow that could greatly effect whether or not the code saw the green ball. To counteract this, we attempted to place equal lighting from all angles so that the green ball would look approximately the same from all angles. This also prevented BAXTER's arm from placing a shadow over the balls.

Videos showing the BAXTER robot successfully sorting three colored balls can be found at http://youtu.be/zKR_vvnJlfA and http://youtu.be/X_zfMMy9RWE. These video show two view points, the view point of the authors and the view point from BAXTER's hand camera, respectively.

4 Discussion and Conclusions

The method used in this project was successful and allowed for completion of this both tasks. Additionally, object detection via color is fast, easy to implement, and requires no training data. However, drawbacks include the variability in success based on lighting and the difficulties when attempting to search for multiple similarly colored balls in one image.

Additionally, depth detection is limited by our preset calibration information. Because we were testing

in a laboratory setting, we were able to guarantee that the camera and object were in approximately the same locations for every iteration, but in a real application this would not be the case.

Therefore, though the project was highly successful, it only reflects the results that can be obtained in a laboratory setting when most of the factors are known. In a real application, the simple object detection code that was used in the project would need to be replaced by a more sophisticated object detection technique.

5 Statement of Individual Contributions

It should be noted that both members have assisted with all parts of this project, and assisted each other when needed. The descriptions given below are simply a general division of labor to help us parallelize the project, and to make this team as successful as possible.

5.1 Brandon Boyce

Brandon Boyce focused on learning how to control BAXTER's limbs and how to obtain images from his cameras. He was also responsible for integrating the code that Jessica developed into the ROS language and the BAXTER control code.

5.2 Jessica Mullins

Jessica Mullins has focused on detecting a ball from a single image and from webcam images. She also worked on the depth detection code. Her code can to output the locations of the centers of the detected object in a 3D space. The details of the approach were outlined in the Section 2.

5.3 Meetings

The group met weekly on Mondays at 4pm, and other times during the week when it was deemed necessary to accomplish the desired goals. Members also worked individually at times. Additionally, the group collaborated using Google Docs and email.

6 References

1. He, Yinghua, Hong Wang, and Bo Zhang. "Color-based road detection in urban traffic scenes." Intelligent Transportation Systems, IEEE Transactions on 5.4 (2004): 309-318

2. Fleyeh, Hasan. "Color detection and segmentation for road and traffic signs." Cybernetics and Intelligent Systems, 2004 IEEE Conference on. Vol. 2. IEEE, 2004

3. OpenCV Tutorial, Color Detection and Object Tracking, http://opencv-srf.blogspot.ro/2010/09/object-detection-using-color-seperation.html

4. Rethink Robotics, http://www.rethinkrobotics.com

5. GitHub on Rethink Robotics, https://github.com/RethinkRobotics/sdk-docs/wiki/

6. ROS Tutorials, wiki.ros.org/ROS/tutorials

7. progCode.in, Object Detection with OpenCV, http://www.progcode.in/object_detection_with_opencv.php

8. Stack Overflow, Finding distance from camera to object of know size, http://stackoverflow.com/questions/6714069/finding-distance-from-camera-to-object-of-known-size

9. Saxena, Ashutosh, Justin Driemeyer, and Andrew Y. Ng. "Robotic grasping of novel objects using vision." The International Journal of Robotics Research 27.2 (2008): 157-173.

10. OpenCV Tennis balls recognizing tutorial, http://wiki.elphel.com/index.php?title=OpenCV_Tennis_balls_recognizing_tutorialImageNet, and http://www.image-net.org

11. ImageNet, http://www.image-net.org

12. OpenCV tutorials, http://docs.opencv.org/doc/tutorials/tutorials.html