Collin Shaddox, Hunter Lewis, and Graham Empey

Dr. Gabriel Ferrer

CSCI 235

11 December 2014

**An Examination of the Potential Efficacy of Object Recognition Through Imaging**

This project was intended to create an implementation of the BRIEF algorithm that is successful at recognizing target objects in a variety of environments. BRIEF is an algorithm that finds features, or remarkable differences in intensities across two pixels, in an image, then compares features between the images. Our implementation included a basic BRIEF program(FeatureImageMatchLocalizer), a program that incorporates RGB values(RGBFeatureImageMatchLocalizer), and several generated patterns that change the qualities of features. Object recognition has a wide variety of potential uses, such as a searcher robot or a camera looking for distinct objects in a complex environment.

**Methods:**

Our basic object recognition algorithm was built from the edu.hendrix.img.database package provided. We removed all localizer elements (save the names), and originally replaced it with the readings of a sonar at the time was taken and currently when the image matcher was running. This was intended to compare distances from the robot to the target object in order to choose the picture corresponding to the closest distance. This was removed shortly after implementation however, because of the sonar’s common inability to receive useful information from curved surfaces.

The “localizers” display an image distance value to the user. The range of values that the correct image will give for image distance vary from object to object, as shown in the Data section. If the image distance of the picture that is the closest match to the picture seen is less than a preset threshold, then the robot will beep, indicating that it believes it has found the correct object. In its current state, the user must determine her own correct values for the threshold.

Because the BRIEF implementation we are using does not factor any color information into its image matching, we wrote a method called colorCheck() to use as a secondary measure in determining an object match. When BRIEF matching thinks it has detected to goal object, we then run a color check to determine whether the matched image has a significantly different color profile from the goal image. If the color profiles are similar enough, then the robot confirms the object match and beeps.

In order to compare the color profiles of the matched and goal images, we took two approaches. The first was to extract the rbg values from each pixel of the two images in bufferedImage form, and then simply sum the red, blue, and green values of each pixel to get a total for each color for each image. We then compare the total for each color, and if an experimentally determined threshold is satisfied, the method returns true. This approach certain problems, for example, the red color sum for two images could be very similar if one image is entirely covered by pixels with a medium red value and the other image has just a few extremely red pixels but none otherwise. A lot of color data is lost by summing color values over all pixels because we can not differentiate between the number of pixels at a certain level of redness.

Our second approach addresses this issue with the idea of a color histogram. In this method, instead of summing all the red values for each pixel, we count the number of pixels in the image within certain ranges of red values and add these pixel counts to an array forming a histogram. Similarly we have a blue and green histogram. It would perhaps be overkill to have 256 different bins in the histogram corresponding to the possible 256 discrete color values, so when extracting the color values from each buffered image we ignore the four lowest order bits in the encoding for each pixel’s color value. This breaks our histogram into 16 bins each ranging 16 color values so that we have a more manageable histogram for each color. To compare the histograms for each image, we find the aggregate difference (i.e. the sum of differences between each corresponding bin of the two histograms) for each color, and check each difference against a threshold value, returning true if each threshold is satisfied.

**Data:**

**Test 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Photographer File | Testing Object | 25cm | 55cm | 85cm |
| Blue Coozie | Blue Coozie | 650 | 700 | 860 |
| Red Coozie | 615 | 725 | 850 |
| Tennis Ball | 760 | 700 | 775 |
| Red Coozie | Blue Coozie | 700 | 700 | 815 |
| Red Coozie | 750 | 700 | 815 |
| Tennis Ball | 885 | 760 | 815 |
| Tennis Ball | Blue Coozie | 650 | 615 | 700 |
| Red Coozie | 630 | 615 | 700 |
| Tennis Ball | 515 | 550 | 585 |

**Test 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Photographer File | Test Object | 25cm | 55cm | 85cm |
| Tissue Box | Tissue Box | 485 | 340 | 485 |
| Fortran Book | 1400 | 910 | 570 |
| Tennis Ball | 1200 | 1000 | 1050 |
| Fortran Book | Tissue Box | 1120 | 950 | 760 |
| Fortran Book | 750 | 645 | 620 |
| Tennis Ball | 930 | 850 | 945 |

**Test 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Object | 25cm | 55cm | 85cm |
| Blue Coozie | 440 | 450 | 570 |
| Tennis Ball | 400 | 450 | 630 |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Object | 25cm | 55cm | 85cm |
| Red Coozie | 425 | 550 | 575 |
| Tennis Ball | 330 | 475 | 575 |

**Test 4 (Control, 256 Point Pairs, 20x15)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Background | Test Object | 20cm | 40cm | 60cm |
| Posterboard | Coozie | 755 | 795 | 700 |
| Tennis Ball | 780 | 900 | 790 |
| Tissue Box | 1130 | 980 | 930 |
| Classroom | Coozie | 730 | 680 | 580 |
| Tennis Ball | 700 | 780 | 585 |
| Tissue Box | 815 | 870 | 800 |
| Robotics Lab Ground | Coozie | 1400 | 1370 | 1320 |
| Tennis Ball | 1300 | 1350 | 1380 |
| Tissue Box | 1210 | 1230 | 1180 |

**Test 5 (Pattern 128 Pairs)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Background | Test Object | 20cm | 40cm | 60cm |
| Posterboard | Coozie | 390 | 360 | 325 |
| Tennis Ball | 445 | 460 | 360 |
| Tissue Box | 540 | 450 | 440 |
| Classroom | Coozie | 430 | 400 | 370 |
| Tennis Ball | 360 | 400 | 350 |
| Tissue Box | 380 | 410 | 400 |
| Robotics Lab Ground | Coozie | 710 | 670 | 640 |
| Tennis Ball | 640 | 650 | 690 |
| Tissue Box | 600 | 560 | 580 |

**Test 6 (Pattern 512 Pairs)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Background | Test Object | 20cm | 40cm | 60cm |
| Posterboard | Coozie | 1580 | 1750 | 2100 |
| Tennis Ball | 1750 | 1870 | 1420 |
| Tissue Box | 2100 | 1830 | 1760 |
| Classroom | Coozie | 1680 | 1800 | 1700 |
| Tennis Ball | 1530 | 1680 | 1600 |
| Tissue Box | 1750 | 1770 | 1720 |
| Robotics Lab Ground | Coozie | 2950 | 2870 | 2720 |
| Tennis Ball | 2900 | 2830 | 2870 |
| Tissue Box | 2450 | 2560 | 2500 |

**Test 7 (Pattern 28x21)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Background | Test Object | 20cm | 40cm | 60cm |
| Posterboard | Coozie | 180 | 160 | 155 |
| Tennis Ball | 185 | 140 | 135 |
| Tissue Box | 290 | 300 | 310 |
| Classroom | Coozie | 220 | 195 | 205 |
| Tennis Ball | 240 | 230 | 200 |
| Tissue Box | 265 | 230 | 270 |
| Robotics Lab Ground | Coozie | 350 | 395 | 355 |
| Tennis Ball | 370 | 410 | 315 |
| Tissue Box | 405 | 365 | 350 |

**Test 8 (Pattern 16x12)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Background | Test Object | 20cm | 40cm | 60cm |
| Posterboard | Coozie | 1220 | 1190 | 1130 |
| Tennis Ball | 1370 | 1310 | 1090 |
| Tissue Box | 1400 | 1090 | 1300 |
| Classroom | Coozie | 1370 | 1220 | 1370 |
| Tennis Ball | 1330 | 1230 | 1310 |
| Tissue Box | 1260 | 1300 | 1370 |
| Robotics Lab Ground | Coozie | 1980 | 1920 | 1990 |
| Tennis Ball | 2150 | 2080 | 2130 |
| Tissue Box | 1850 | 1860 | 1850 |

**Test 9 (Color Checking 1)**

**Reporting difference between red totals for each image, by absolute value**

|  |  |  |  |
| --- | --- | --- | --- |
| Goal object:  Red Coozie | 20 cm | 40 cm | 60 cm |
| Red Coozie | 550000 | 220000 | 1400000 |
| Blue Coozie | 302000 | 250000 | 760000 |

**Test 10 (Color Checking 2)**

**Reporting aggregate difference for red histograms**

|  |  |  |  |
| --- | --- | --- | --- |
| Goal object:  Red Coozie | 20 cm | 40 cm | 60 cm |
| Red Coozie | 14100 | 24400 | 26600 |
| Blue Coozie | 16400 | 29500 | 27500 |

**Results:**

The first test showed that for monochromatic objects without distinguishable features, the image distance values for the correct object are not significantly lower than those of the incorrect objects. The tennis ball performed better than the coozies due to having more distinguishable features on the surface.

The second test shows that for objects that have many distinguishable features on their surfaces perform significantly better than the basic test objects used in Test 1. The relativity of the numbers clearly show which object is correct, especially for the tissue box, which had images of flowers all over the side and a unique shape.

Test 3 was a test to see if the correct object was identified when pictures were taken of each object in one Photographer file. The most important takeaway from these tables is that the correct object was chosen every single time.

Tests 4-8 deal with the use of different patterns and their efficacy. These generated patterns vary the feature size or the number of features chosen. Doubling the amount of features provided useless, while using half as many features was just as reliable as the control. This is most likely because choosing too many features makes the robot look at poor features, quite possibly in the background. Test 7 gives more stable image distance readings than the control, but not enough data has been found to suggest that it is better in all cases. Test 8 does not perform significantly different than the control. Perhaps the change in area of features affects the stability of readings more than the dimensions of the features.

We were never able to make either method of color checking very effective in the end. While both algorithms would run and return significant numbers, there was some elusive problem which caused the color checking to become completely ineffective, that is, when searching for a red object the color difference values returned were sometimes larger when comparing to the red object than when comparing to a blue object, which is the exact opposite of what we expect. The first algorithm was so plagued by this problem that it almost never gave useable readings. Because this problem seems to occur somewhat at random, and the robot’s reported data is highly variable, our attempted methods of color checking, while occasionally effective, are not reliable and only hinder object recognition in their current state. If the algorithms could be modified to give more stable readings, these approaches would certainly be effective to some degree.

**Conclusion:**

There were several approaches to the problem of object recognition that we attempted to implement, but realized were not entirely feasible within our time constraints. If these ideas were to be implemented in a similar project we believe that the depth of information gained from the data may increase, and even make data gathering easier for the tester. We developed a training program to be run after Photographer, which would prompt the user to take pictures of its target object. The image distance values that were given from the object would be recorded and accounted for so that a reasonable threshold could be derived for that specific target. We outlined a background removal algorithm that we believe would optimize this image matching behavior. It would start with the user taking a picture of a target object, ideally with a completely monochrome background. Using a GUI we started coding, the user will be able to pull up their picture on a JPanel, and drag a shape around their target image, then all the pixels within that shape(Circle, Rectangle) object is identified as the target object. What the algorithm will do after that is make new images based on the user’s selected pixels, and every pixel that was not within the user’s selection is sectioned into quarters. These quarters are then set to be random colors/ intensities limited by the user’s choice of background color range. The objective of this algorithm is to account for BRIEF’s dependence on the features of an entire object. Hopefully, BRIEF would then be looking for features exclusive to an object and variations of features having to do with the object and numerous backgrounds.

The problem space of this project has migrated very far from the field of robotics, and landed on the exclusive area of computer vision, however this has many applications for autonomous robots. After analyzing our data and observing the readings from the robot, we’ve concluded that our BRIEF implementation would be better suited analyze albums already in memory. Our project focused on object recognition through the live feed of a webcam, and we observed the readings would never be the same even if we tried making the environment exactly the same as before.