Using edge statistics for object recognition

Laurel Issen

May 15, 2006

Abstract

Many perception experiments have shown that object boundaries, defined by edges, contain a wealth of information for recognizing the object. This project is an attempt to construct an edge-based method of recognizing a simple object in a computer vision system. I chose to explore this problem within the domain of robotic soccer, where the visual system must identify the soccer ball among other robots, field features, and objects outside the field. The method presented in this paper uses only edge boundary statistics and performs better than chance at this task. I also outline future work that could improve accuracy in computer recognition of the soccer ball.

There have been several initiatives in both cognitive science and computer science to recreate human vision with computer algorithms. In cognitive science, the goal is to figure out how the visual system works, and in computer science, the goal is to make intelligent machines that can sense their environment. This project is an initiative to bring the two fields together, by combining two particular projects.

The first of these projects is RoboCup, which is designed as a testbed for creating rational agents. Teams of researchers program Sony Aibo robots to play 4 vs. 4 soccer, where each robot is running its own program and acts based on input from its camera and touch sensors, as well as wireless communication from its robotic teammates. For a full description of building a code base in this domain, see Stone et al.[3]

One of the most challenging aspects of this research is the computer vision algorithms that are necessary for the robots to know what is happening on the soccer field. Currently, the most successful algorithms get the job done by relying on color information, such as the approach by Sridharan and Stone[2] where the visual space is segmented based on color, and this leads to candidate blobs which are then matched to possible objects.

The RoboCup domain supports this solution by making the important objects on the field be bright colors. The robots wear stick-on uniforms in royal blue or red, the goals are bright yellow or bright blue, the soccer ball is bright orange, and localizing beacons are combinations of pink, yellow, and blue. Unfortunately, relying on color is not robust enough for the task because in many cases, when lighting is not uniform on a particular playing field, or if it is nonstandard across different fields, results can be unreliable. Because the algorithm would need to accept a variety of brightness and RGB values as the same orange as the soccer ball, many other things may be accepted as orange to the robot, such as the hand of a person observing the game. Other times, because the ball is not well-lit, it will not be in the range of values accepted as orange, and the robot will not see the ball.

The RoboCup league understands these limitations and aims to move away from the color- centered vision systems. To encourage this, the league plans to replace the orange soccer ball with a black-and-white ball. My goal in this project was to use an edge-based algorithm to identify a black-and-white soccer ball in robot-eye images of game scenes. I decided to use an approach similar to Geisler et. al. which created an ideal observer at contour grouping based on edge co-occurrence statistics (Geisler, Perry, Super & Gallogly, 2000).[1]

1 Method

1.1 Input photographs

The photographs used for analysis were taken with the camera that comes on the Sony Aibo robots. I converted the images from their native YCbCr format to black and white, so the project was completed with no color information whatsoever. I used game-style images that were designed to be scenes that the robot might encounter in regular game play, including scenes with other robots, and scenes where the soccer ball is partially occluded. See Figure 1 for an example. I was not able to use actual game images because the robots were not programmed to recognize the black and white ball.

1.2 Edge extraction

I used the same edge extraction method as is described in Geisler et al. 2001.[1] This method is similar to the response of neurons in the primary visual cortex, so it is a logical way to begin processing an image. The first step is to filter the image with a non-oriented log Gabor function, which has a

Figure 1: One example of an input image

similar function to surround-off cells in the visual system. These cells increase response when the center of its visual field becomes more illuminated, and decrease response when the edges of its visual field are illuminated. As a result, as the log Gabor function approaches an edge, the response will go either from a negative to a positive response, or from a positive to a negative response. Each of the zero-crossings that the log Gabor functions produce are treated as potential edge elements. Then, each potential edge element is filtered by a series of oriented log Gabor filters, and the output of this function is analyzed to determine the orientation of each edge. Finally, each edge was assigned an edge strength by determining the normalized contrast energy at its preferred orientation.

1.3 Edge grouping

The output of edge extraction is a map of the x and y pixel coordinates of each edge, its orientation, and the edge strength. An edge which went from pure white to pure black would have the highest possible edge strength. Figure 2 shows a scatterplot representation of the edge output from Figure 1, ignoring edge orientation.

Based on these scatterplots, I determined there would be enough information in the edges that had at least 10% of the maximum edge strength to identify the edges that had come from the ball. This was because the edges along the outside of the ball tend to be strong, since they go from the dark grey ground to the white ball.

Then, for each of the edge elements still left, I created edge groups based on the idea that if edge A is within a Euclidean distance of 2 pixels of edge B,

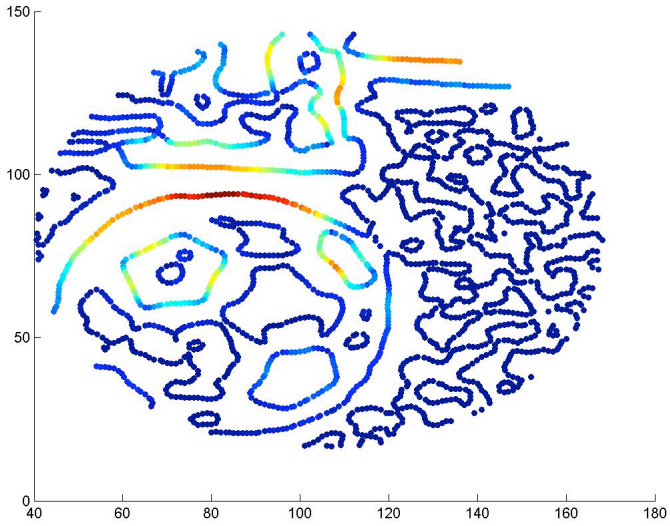

Figure 2: The scatterplot representation of the edge output, ignoring orientation. The dark red pixels are locations of the strongest edges, and the dark blue pixels are locations of the weakest edges. The scatterplot is not on the same scale as the image.

then edges A and B should be in the same group, and that the groups should be transitive. The algorithm does this by creating a matrix of the distance between each pair of points in an image, then starting with the first edge element, searches for other edges within the distance threshold to associate with the first group, and recursively repeats this edge-pair finding on the newly grouped edges as well. When the last edges added to the group do not result in any new edges being added, that group has been exhausted, and the algorithm repeats the process with the next ungrouped edge. Finally, all edges are contained in a numbered group, even if that group consists only of a single edge by itself.

I then eliminated any groups which included fewer than 15 edges, which left about 10 edge groups per picture. For each of these groups, I plotted one edge group at a time, overlaid onto the input images, and selected which edge groups contain edges that are part of the edge of the soccer ball.

1.4 Analyzing the properties of edge groups

1.4.1 Method 1

Now that I had groups for each of the eight images numbered and labeled, I needed to analyze the differences between edge groups that represent the outside of a soccer ball from edge groups which do not outline the ball. Once the distinguishing properties of the ball groups have been identified, I will be able to locate these groups in an input image and therefore identify the location of the ball.

Recall that the information we have for each edge in a group is its coordinates in the x-y plane, its orientation in radians, and the edge strength. Since a soccer ball will always project a circular image, I decided first to use pairs of edges to predict a center-point based on their relative locations and orientations. In theory, ball-edge groups would have a tight cluster of projected center points. Other curved surfaces like the other robots might have a higher standard deviation of projected center points. This method would also have the advantage of automatically computing the most likely location of the center of the ball, as a step along the way to determining which edge groups, if any, are likely to be the boundary of a soccer ball.

The orientation of an edge was defined by the angle in radians that the edge had traveled in a counter-clockwise direction from vertical, so I was able to find the slope and y-intersects of the line perpendicular to an edge by solving the following set of equations:

$$m = \tan\theta \tag{1}$$

$$b = y_0 - mx_0 \tag{2}$$

Then with the equations for both perpendicular lines, I found the intersection point, which would be the predicted center point of the circle. I had to include special cases for when the lines were parallel to each other, which on a perfect circle would only happen when the edges were on exactly opposite sides of the ball, so in this case the center point would just be the average of the x and y values. I also handled the case where either edge is perfectly horizontal.

When I had an array of estimated center points for each group, I calculated the average x value, the average y value, and the standard deviation of the group points from this mean. Our prediction was that the standard deviations for ball groups would be lower than that of non-ball groups.

1.4.2 Method 1 Results

While the center-point method would work in theory, in practice, the small amounts of noise in the edge orientations correspond to large error margins in the results. This is because many pairs of edges within a group are very close to being parallel to each other, and so being off in one of the orientation angles by a small amount will shift the intersection point of the perpendicular lines drastically. Figure 3 shows the standard deviation plot of the ball groups versus the non-ball groups. The ball-group distribution looks identical to what you would expect from a sample of the non-ball-group distribution, so the standard deviation of predicted center points for a group would not be at all helpful in distinguishing groups of edges that came from the outside of the soccer ball from groups of edges that came from other objects.

1.4.3 Method 2

The next method that we tried involved seeing how closely the edges match the equation for a circle. This would likely be an effective method even ignoring the edge orientations, though the orientations might be another parameter that is useful to consider in distinguishing ball groups from non-ball groups. For each group, we will pick x_0, y_0, and r to minimize S in the equation

$$S = \frac{1}{n} \sum_{i=1}^{n} |(x_i - x_0)^2 + (y_i - y_0)^2 - r^2|$$

where (x_i, y_i) is the location of the ith edge in the group of n edges.

To find inputs which minimized the error, I used the fminsearch[1] function that comes with MATLAB. This function takes the error-finding function

[1] http://www.mathworks.com/access/helpdesk/help/techdoc/ref/fminsearch.html

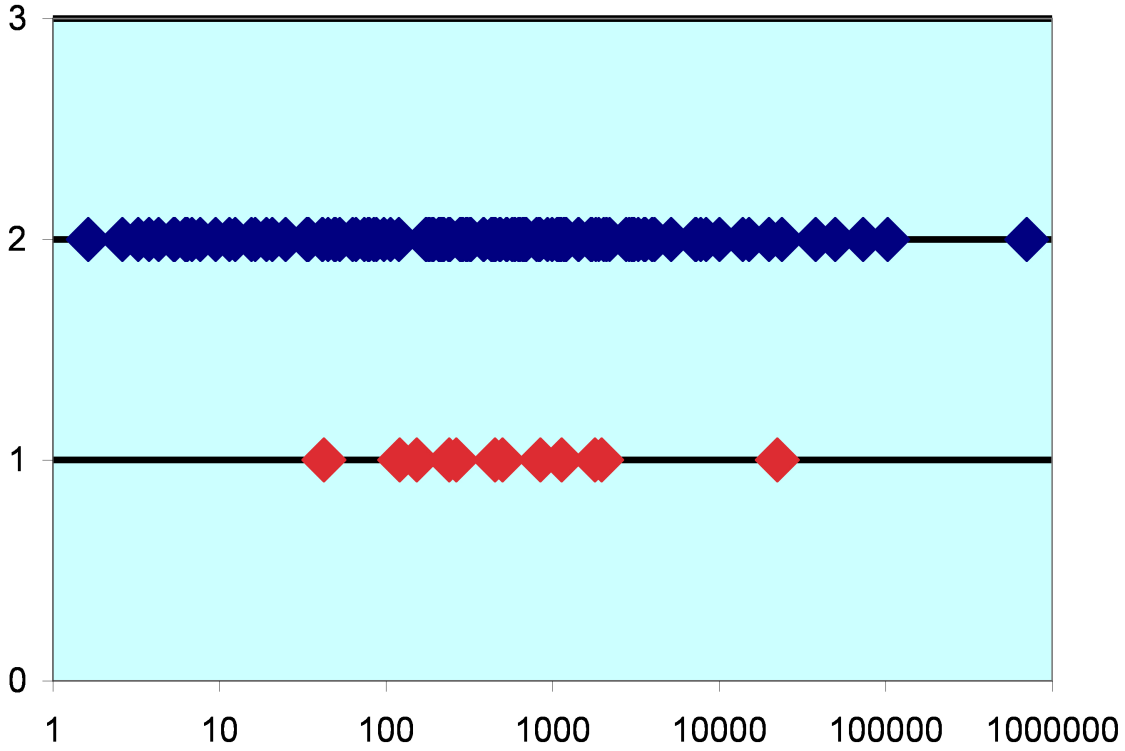

Figure 3: The standard deviation distribution of the non-ball groups (blue) and ball groups (red)

and a starting value for x_0, y_0, and r, and returns a minimal error and the x_0, y_0, and r values at that minimum. This function can, of course, return a local minimum instead of a global minimum, so I needed to choose my starting values carefully. I gave it an r starting value of 60 pixels, because this would be around the upper bound on how big a ball radius could be, and I started x_0 and y_0 at the group's centroid, that is, the coordinates of the mean x and $y$values for the group. This function also has a limit on how many computations it can do to find the minimum before it returns. If the function returned without converging on a solution, I coded it to return an undefined S_{min} value.

This approach had the advantage that the small amounts of noise in the information about the edges themselves will correspond to low amounts of noise in the output. It had the same advantage as before of predicting the center point of the ball as part of the algorithm, and it also predicted the radius of the ball, which would aid in determining its distance. I hypothesized that groups representing the boundary of the soccer ball would have a much smaller S_{min} than other groups will have.

1.4.4 Method 2 Results

When I plotted the S_{min} results, I realized that I needed to normalize the data because non-circular groups which take a small amount of space could still have lower error than circular groups which took a large amount of space. I was already normalizing for the number of points in the error function itself, but I decided to plot each groups error divided by its estimated radius. I also decided to give any groups which had an undefined S_{min}, or a predicted radius of less than 2, an S_{min}/r value of 10 for ease in plotting. Figure 4 shows a histogram of the normalized error values for ball groups and non-ball groups.

Based on this histogram, I decided that my determination function should predict that any group which had an S_{min}/r value less than 1.5 represented the boundary of a ball, and any group with an S_{min}/r value greater than or equal to 1.5 did not represent the boundary of a ball. This function as-is would be correct on 62% of the input groups. That is, out of the 87 edge groups in 8 pictures, this algorithm would correctly identify 7 out of the 9 ball groups, and would correctly eliminate 47 out of the 78 non-ball groups. The two ball groups which would be incorrectly rejected turned out to be groups which contained some edge elements outside the soccer ball. I explain this problem and possible solutions in the next section.

I also determined what would happen if I only guaranteed to recognize soccer balls which were near enough to have a predicted radius of more than

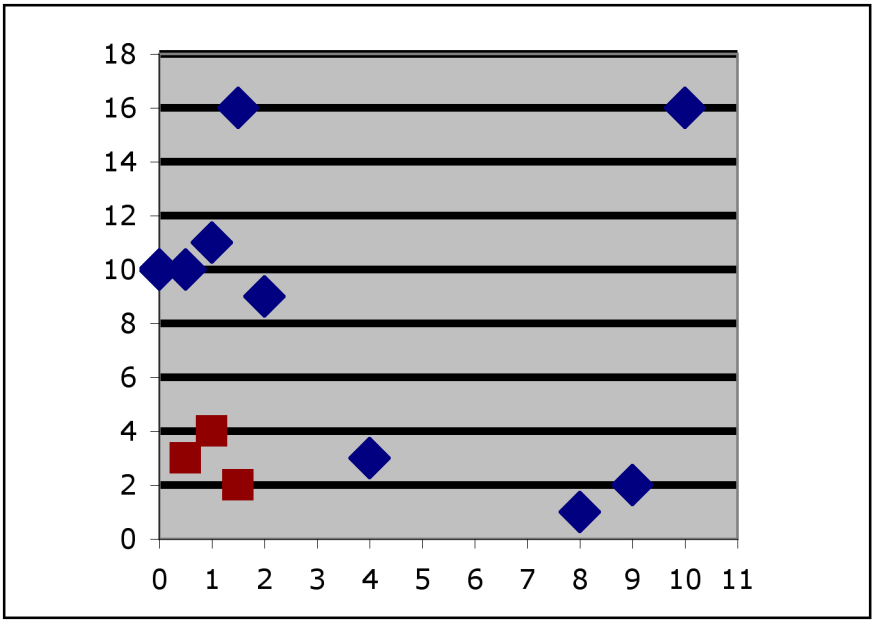

Figure 4: The normalized error histogram of the non-ball groups (blue) and ball groups (red)

10 pixels. This way, I could throw away any groups which had $r < 10$. By doing this, the algorithm improved to 70% accuracy of the input groups, still correctly identifying the 7 out of 9 ball groups but improving the correct eliminations to 54 out of 78.

2 Future work

2.1 Distinguishing among round groups

One problem with using only the edge boundary statistics is that groups that represent the edge of the ball, and groups that represent the edge of other round objects such as the other robots and the black pentagons on the soccer balls, do not have statistics which are different enough to distinguish them reliably. The prediction algorithm could be improved by checking for strong edges within the bounds of the predicted circle, which are likely to occur in soccer balls because of the black-and-white pattern, but not in other objects. An alternative could be to sample random pixels within the ball to see if the distribution of brightness values is similar to those found in other balls.

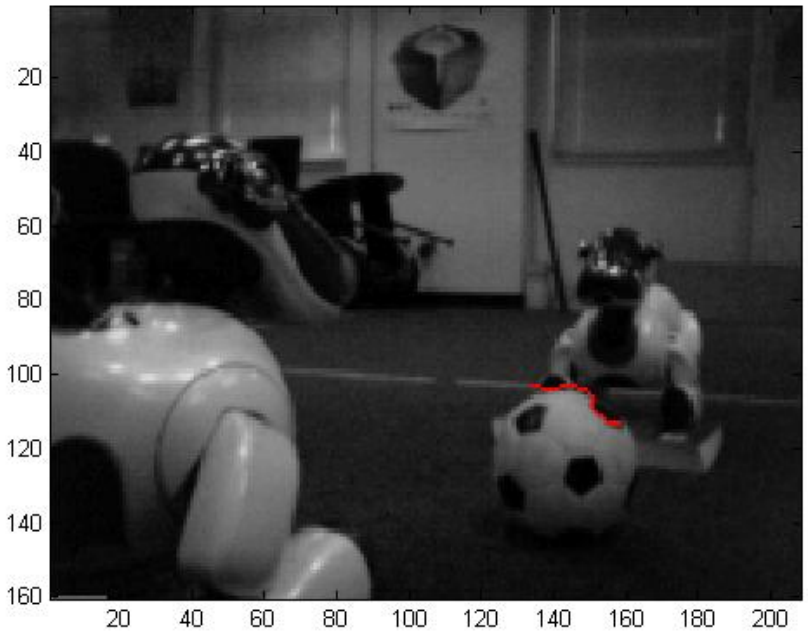

Figure 5: This complete group should stop before continuing into the black patch on the right, and should continue the line of the ball instead of the line of the robot on the left, based on the properties of good continuation.

2.2 Improving edge grouping

It may also be beneficial to improve the edge grouping algorithm to minimize the amount of edges that are put in a group that they should not belong to. Figure 5 shows an example of a group boundary that does not follow the true edge of the ball. Not only does it continue around the black patch into the internal boundary of the ball, like many of the ball groups do, but it also loses the edge of the ball to follow another robot's foot. A sophisticated approach that might solve both problems would be to add a good-continuation criteria. We could use this by applying the ideal observer from Geisler et al. 2001[1] to get a probability that the two edges came from the same contour, and set a threshold probability as well as a threshold distance for adding an edge to the group.

3 Conclusions

While these methods do not implement real-time algorithms that will be used as-is in the robots, this project aims to show possibilities for how to maximize information that will be useful for object recognition from edge analysis. Also, since I was able to make progress toward recognizing a simple object using edge analysis alone, using these methods in conjuction with other methods could reinforce object recognition and make it more robust. It may be a few years before robots would have the hardware to do all of the computations necessary in real-time. One of the most expensive calculations was the application of log Gabor filters to the image, which in our implementation had to be done serially, but in the human brain this happens in parallel. If a robotic visual system had extra hardware optimized for this task, real-time decisions based on edge locations and orientations would be much more feasible.

4 Acknowledgments

Thanks to Bill Geisler, my advisor, Peter Stone, my second reader, and Risto Miikkulainen for serving on my committee and helping me with my experimental design as well as the written work. Thanks also to Jeff Perry for helping me with technical details, and to Vijaya Ramachandran and Yadirah Chujachi for assistance with the administrative tasks.

References

[1] Geisler, W.S., Perry, J.S., Super, B.J., & Gallogly, D.P. (2001). Edge co-occurrence in natural images predicts contour grouping performance. *Vision Research, 41,* 711–724.

[2] Sridharan, M., & Stone, P. (2005). Real-time vision on a mobile robot platform. *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

[3] Stone, P., Dresner, K., Erdougan, S. T., Fidelman, P., Jong, N. K., Kohl, N., Kuhlmann, G., Lin, E., Sridharan, M., Stronger, D., & Hariharan, G. (2004). The UT Austin Villa 2003 Four-Legged Team. *RoboCup-2003: Robot Soccer World Cup VII.*