The Camera: Botball's Most Underrated Sensor
Adam Farabaugh and Evan Wilson
Hampton High School
adam.farabaugh1@gmail.com, emwilson02@gmail.com

The Camera: Botball's Most Underrated Sensor

1 Computer Vision in Botball

As many of you have probably realized, in the past few years KIPR has included elements in the Botball games that require using the vision system to score points. There were green and orange tribble stacks in 2008, the fuels that switched sides on the ramps in 2009, the size- and location-variant oil slicks and corresponding ducks in 2010, and the luggage sorting this season. When our club competed in the 2010 Global Botball Tournament, we were surprised to see how few teams were utilizing the camera: out of sixty-odd teams, only a handful used it in their strategies at all.

One of Botball's unique characteristics is that it is entirely autonomous - there are no remote controls and no operator interaction. This means that Botball robots must either rely on the accuracy of dead-reckon navigation, or utilize the various sensors included in each year's Botball kit. Teams should recognize that the camera is one of the supplied sensors, and a very sophisticated one at that. It goes beyond reporting reflectance and measuring distance and enters the realm of object recognition, which potentially allows for extremely sophisticated behaviors that do not rely on assuming a position on the game board or the location of an object.

Computer vision has been demonstrated through many robotics research projects, some as well known as those by the University of Tokyo’s high-speed hand [1], that utilize vision rather than other sensors. This is done because vision, in situations where objects have colors and lines that make them stand out from their backgrounds, provides unparalleled information about objects and their orientation.

This paper's intent is to help fellow Botballers see past the faults of the CBC vision system as it currently exists and, hopefully, inspire more teams to stretch themselves, branch out, and add a higher level of sophistication to their strategies by implementing the camera in their robots. It will also present the vision library the Hampton team has developed to take advantage of as much of the current vision system as possible.

2 Pros and Cons of the Vision System

Since the migration from the XBC to the CBC, many Botballers have discussed the merits and problems that come with the current vision system.

2.1 The Camera's Advantages

As stated before, the camera system's primary advantage is that it allows for object recognition. Because the game pieces in Botball are often colored brightly and with great contrast, the color model system allows for the recognition of four separate objects. Since each robot can have a camera, these four models are more than enough to apply to a single robot's strategy (and we will discuss potential uses for unused models later).

KIPR's implementation of the vision system through their tracking library provides nearly countless ways to use the data the camera provides, more so than any other sensor. The objects that the camera recognizes have many attributes, like size, coordinates, dimensions, and other attributes that may not immediately appear useful but have niche applications, such as the angle of the blob's major and minor axes, and the confidence of blob recognition.

2.2 The Camera's Disadvantages

Along with the pros comes a long list of cons. The first major difficulty is that either the camera or the driver that mediates between it and the CBC experiences extremely high latency: the time between an object’s appearance in front of the camera and the time at which it is recognized as a blob is very large. Another problem with the camera is that the color model definitions are limited in the amount and area of the color space they can cover; most notably, a color model cannot be calibrated to recognize black. Also, the camera has no depth perception, so although one may think to use blob size to approximate distance, blob size is extremely reliant on lighting conditions and therefore simulating depth perception with the camera is difficult.

A mechanical disadvantage of the camera is its flex-tube cable, which makes it difficult of a team wishes to actuate the camera to "see" at different height levels or angles. This issue has partially been mitigated because of the inclusion of a USB extension cable that is flexible for use on one robot.

3 Shifting Paradigms to Get Ahead

When our team decided to use the camera in the 2010 game, we made a lot of mistakes. The most detrimental were to use the camera while the robot was moving, creating issues because of latency; using the camera to see far away, making the recognition overly sensitive to light conditions; and relying on the orientation of the camera to compare the center of a blob to a specific x-value, making it too easy to accidentally bump the camera and ruin a run. We think it is too easy for a team to try to use the camera one year, run into a mountain of problems, and then decide to not try using the camera again in the following years.

When we saw this year's game, however, we knew immediately that we wanted to try to use the camera again: after some discussion of our lack of success using vision last year, we decided to shift paradigms to eliminate the cons of the camera and take advantage of as many pros as we could.

We first knew that we couldn't use the camera while moving, or to see very far away. We also didn't want to rely on the exact direction the camera pointed. We did, however, decide that the tracking library does a good job of providing vast amounts of data, so we wanted to utilize that information as much possible.

Some of this shift in usage technique manifested itself in our strategy or the mechanical design of our robot. We constructed our luggage sorting mechanism deposit luggage about an inch away from the camera lens so as to eliminate the chance of missing a blob because of distance. We also strategically placed the camera so it can see which luggage bin is in front at the very beginning of the game, before the robot ever moves.

The rest of our changed technique in using the camera comes in software: we have written a comprehensive vision data processing library, to be open-sourced sometime in the future, that takes advantage of the fact that the camera is not a very good webcam, but the CBC is a very good computer.

4 Investing in a Vision Library

We mentioned above that KIPR's tracking library allows access to a wealth of data about each blob, but that teams rarely utilize this functionality. We wanted to take advantage of this data by writing a software library that can manipulate data produced by sorting blobs by their attributes, filtering out blobs by their attributes, and merging blobs that have attributes.

The software we have developed basically takes the data that KIPR's tracking library produces, and stores it into a matrix of data in memory. Currently it only stores attributes such as size, x and y coordinates, and the locations of the boundaries of the blob's bounding boxes, but more can easily be added. We wrote functions that can sort blobs in ascending or descending order by any of these attributes, remove blobs below or above size or position thresholds from the arrays, or merge blobs that are close in distance.

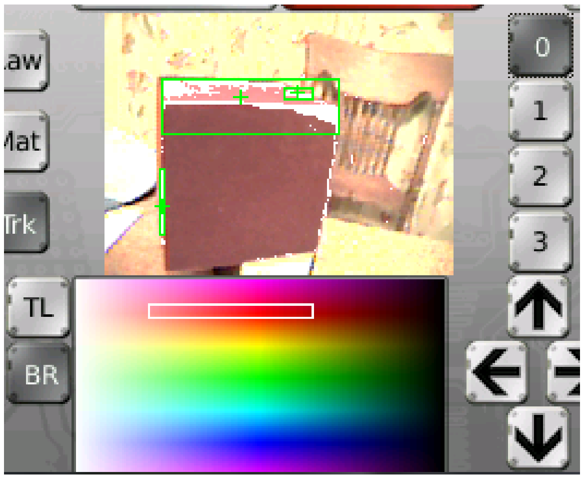
One of the aspects we especially like about our library is that it can merge the blobs from two color models into one, so that they can be sorted, filtered, merged, and compared together. We made this with the intention of using two or three color models for the red foam blocks, since they appear to be different shades in the cameras view depending on the angle which light hits them, but we are sure this same functionality could be used for many more things.

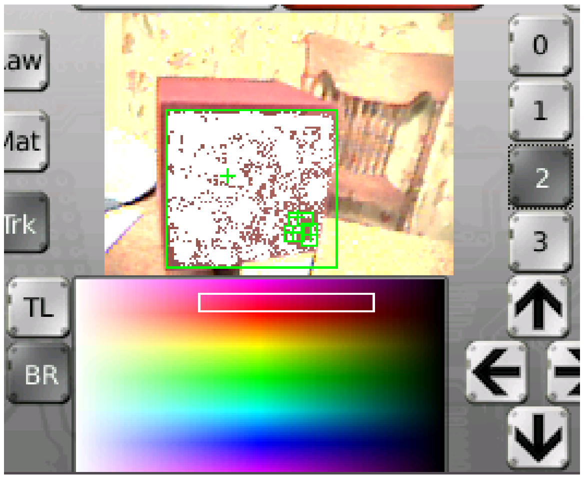
Coding this did not use much in-depth knowledge of C; we used arrays and a structure. Some knowledge of pointers is required to write sorting functions, but each comparison function is basically a template that can be copied and expanded to other attributes. We did implement some common math optimizations that to eliminate floating point operations [2]. Using this library; however, is simple. Its implementation is similar to using any function that is included in the standard C libraries and KIPR library. Depending on the amount of interest, we would like to eventually open-source this library to the Botball community.

4.1 The HHS Vision Library

We can demonstrate this vision library by explaining how we used it to more reliably detect the position of the red foam cubes in this year's game. The library cannot currently draw what is shown in the later images of this example, but they do demonstrate conceptually what the library does.

Here are two images obtained with VNC, part of the NHS alternate CBC firmware, of the CBC tracking screen. These are two interpretations of the same image with two different color models:



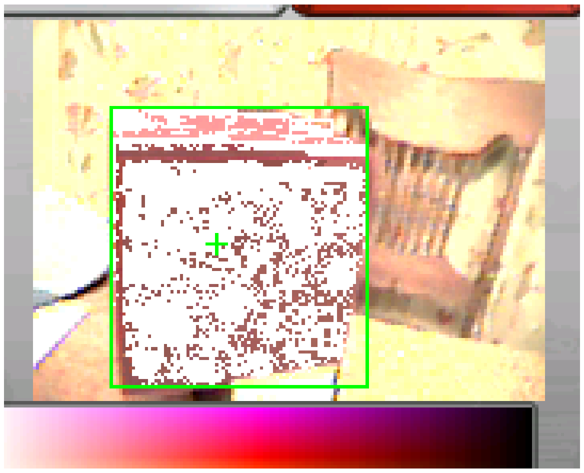


As you can see, these two separate models are each good at a different thing. The first easily "sees" the facets of the cube that are in direct light, while the second detects the parts of the cube that are not reflecting light well towards the camera and are therefore darker. We tried combining the models in the color space below the images, but the image got extremely noisy, worsening as the ambient light decreases. Since we don't want our robots to be affected so much by light conditions, we can use the vision library to combine the blobs from both channels into a virtual channel, then manipulate that data to obtain the coordinates of the center of the entire blob:

```
int main(){
   init_hhs_vision();
   clear_all_blob_data();
   track_update();
   get_all_blob_data();
```

```
    // merges channels 0 and 2, putting result in virtual channel 5,
    // prefers large blobs over small ones if there are more than 10 total
    merge_channels(0, 2, 5, LARGE);

    // merge blobs that are within 8 pixels of each other
    merge_by_distance(5, 8);

    // sorts the virtual channel around the center of the image, preferring
    // smaller distances to the center of the image
    sort_by_point(5, 80, 60, SMALL);

    printf("Block coords: x=%d, y=%d", vision_data_array[5][0].x,
    vision_data_array[5][0].y);

    close_hhs_vision();
}
```

The result of this merging and sorting would look like this (if the library could draw to the tracking screen):



Clearly, the bounding box is much better conformed to the outside of the block. The centroid of the blob has shifted too far to the left, because our library does not currently weight the centroids by amount of pixels like the KIPR library does, but that can be implemented.

5 Things to Implement and Future Computer Vision in Botball

There are additions to be made to this library that we are well aware of, but simply did not have time to implement. Most of them involve speeding up the processing; we have already replaced processor-hogging square roots with an integer-division approximation. Another improvement in speed would come if we read the blob data directly from memory instead of using the tracking library functions to fill the data array.

As of the time of writing, according to recent committs on the KIPR repository of the CBC firmware on Github.com, it appears that we may soon be able to change color models at runtime. This would probably allow some more functional expansion of the vision library.

Jeremy Rand offers some compelling improvements to the camera's capabilities. Our library should be able to work along with his work to provide a more refined vision experience from start to finish. More functionality may also be added to our library when we are able to test the new functions of Rand's research that can be utilized by those adventurous enough to try modified firmware.

6 Conclusion

We hope that Botball teams will reassess their use of the camera in their strategies. It is an extremely useful tool and offers capabilities that make entirely new approaches to the game possible. The camera is a sensor that allows Botball to be a very advanced program in terms of available equipment, and using it creates invaluable learning opportunities for implementing complex, decision-making code.

With the compilation of our vision library, we were trying to make it easier for ourselves to use the camera in the future. However, these building blocks for complete programs are very useful for any team looking to use the camera. We want to continue to share our ideas and techniques with other Botball teams so that each year the robots will keep becoming more creative and innovative.

References

[1] Ishikawa Oku Laboratory, University of Tokyo. Sensor Fusion: High Speed Robots. http://www.k2.t.u-tokyo.ac.jp/fusion/HighspeedHand/index-e.html

[2] Thomas Jakobsen. Advanced Character Physics. http://classic-web.archive.org/web/20070610223835/http://www.teknikus.dk/tj/gdc2001.htm