The Effectiveness of Brightness Metrics for Ball Detection

Marcus Christiansen, Will Gantt

December 16, 2016

Abstract

Broadly, the goal of this project was to determine whether brightness measurements could be an effective tool for ball detection. Specifically, we considered how such metrics might be incorporated into the spot filter; first, in the identification of blacks spots, second, in a comparison between the brightness values of the top and bottom hemispheres of the ball itself, and, third, in a comparison between the brightness values of the regions immediately above and below the ball. Regarding black spots, we found that there is little difference in brightness between true and false positive instances. In the comparison of hemispheres, we found that .... In the comparison of above-ball and below-ball regions, we saw ....

1 Overview

Since the recent decision of the RoboCup Standard Platform League to replace the standard orange ball with one more reminiscent of a classical soccer ball, teams have been faced with a slew of new vision challenges. As the ball's primary color is now white, teams must find novel ways to distinguish it from other white objects, including field lines, goal posts, and robots. Furthermore, they must differentiate the black pentagons of the ball from such things as robot joints and other dark spots on the field. Thus, color alone can no longer serve as a reliable marker of the presence of a ball. Consequently, the Northern Bites have implemented a variety of additional checks. These include determining the expected radius of the ball at different locations on the field, evaluating the spatial relationships between black spots, and verifying that a candidate ball is in fact on the field.

Although the ball detector concerns itself to a great extent with white and black, it makes almost no use of brightness inputs. Since black is notoriously difficult to characterize strictly on the basis of color (U- and V-) values, we wanted to see whether brightness (Y-values) could provide some helpful information in this regard.

2 Experiment

Currently, the Northern Bites robots identify candidate balls using a spot filter. The basic principle consists in scanning an image with a "filter," which comprises a small, square-shaped region nested within a larger one. At each position in the scan, the filter performs a simple check to determine whether the outer region is whiter on average than the inner one. The idea

is that a ball will tend to be darker in the middle, since the black spots typically fall closer to its center, and whiter around the edges.

The tests we ran operated on spots that the detector already identified. That is, the tests were intended to filter, rather than discover, candidate balls. The first test considered the median Y-value of each detected black spot, with the goal of determining whether there was any consistent, significant difference between true positives (black spots actually on the ball) and false positives. The second compared the median Y-values of the top and bottom halves of the inner region of a white spot (a candidate ball). We reasoned that, because the field is lit from above, the top half of a ball should tend to be brighter than the bottom half. By the same logic, we hypothesized that the shadow cast by the ball on the field itself should render the area of the field directly below the ball darker than the area directly above it. This was implemented as a third test.

All three tests were conducted on logs taken in the robotics lab at illuminance levels of 150, 300, and 450 lux.[1] For each level, we placed the robot in 12 different scenarios, and for each scenario, we made adjustments to some combination of the following variables:

- The presence of the ball on the field (i.e., *whether* the ball is on the field)
- The ball's position on the field.
- The robot's distance from the ball.
- The robot's position on the field.
- The number of other robots present on the field.

[1] In truth, measuring the brightness of an entire room with this level of accuracy is impossible, given imperfections in the light meter, as well as minor fluctuations in the illumination of the room itself. These values should therefore be taken as approximate.

2.1 Black Spot Detection

In the first test, we differentiated true and false positive black spots by looking at the output of the "DebugImageView" module in the tool, which circles in yellow all of the black spots that the ball detector has identified. For each log, we documented the total number of black spots detected, the number of true positives, the number of false positives, the Y-values of each spot, the number of other robots on the field, and the presence of the ball in the image. Once we had obtained this information for all logs at a given illuminance level, we calculated the median, mean, and standard deviation of the true and false positives.

2.2 Hemisphere Comparison

2.3 Region Above and Below Ball

3 Algorithm

As our goal in this project was merely to see whether brightness tests could be used effectively in the ball detector, and not to write game-ready code, we were not particularly concerned with algorithmic efficiency. We do, however, offer suggestions for how the most promising of these algorithms might be improved. The source code for the functions we used can be found at the end of this paper, but we have included high-level descriptions of what each does below.

`getMedianBrightness:` A spot consists of a smaller square region nested within a larger one. This function determines the median Y-value of the inner region of a black spot

by simply iterating over all of the pixels in that region, and adding the Y-value of each to a vector. The vector is then sorted and the element at the median index is returned.

`topOfBallBrighterThanBottom`: Takes the inner region of a ball (a white spot) and divides it into a top half and a bottom half. The median Y-values of each half are determined using the same method as in `getMedianBrightness`, and are returned together as a pair of doubles.

`aboveBallBrighterThanBelowBall`: This uses the same basic approach as the previous function but considers the rectangular areas (1) between the top edge of the outer region of the spot and the top edge of the inner region, and (2) between the bottom edge of the outer region and the bottom edge of the inner region. Both of these areas cover some of the ball, as well as some of the field. The function determines the median Y-values of each region and returns them as a pair of doubles.

4 Results

Unfortunately, neither of us has the knowledge required to conduct tests for the statistical significance of our results. Despite this, we are willing to conclude, based on the similar averages and large standard deviations of the Y-values of true and false positive black spots (shown in Figure 1 at the end of this paper), that brightness is not an effective feature for making this distinction.

Results from hemisphere test ...

Results from above-ball/below-ball test

Our experiments have at least shortcomings that we want to acknowledge. For one, the measurements provided for the level of illuminance in the lab are only rough approximations. Although we did our best to eliminate outside light sources and to accurately calibrate the light meter, we cannot guarantee that the illuminance level remained fully consistent across trials. We suspect that complete consistency in this respect would not have had a significant impact on the results of the hemisphere comparison or above-ball/below-ball tests. It is conceivable, however, that it would have had some effect on the black spot experiment in that it perhaps would have yielded a slightly more consistent difference between true and false positives.

For another, the experiments suffer from a lack of diversity in the data collected. All of our logs were taken at a single location — to wit, the RoboCup lab. We are confident that the hemisphere comparison test is sufficiently robust to work under a variety of lighting conditions, but we would like to have more data to support the claim. Regarding black spots, however, if the robot cannot distinguish the true from the false under lab conditions, we doubt that it would perform much better elsewhere.

5 Future Work

Clearly, there is much work that remains to be done on the Northern Bites' ball detection system. The results of our experiment suggest several problems particularly in need of attention.

First, the system is remarkably fragile. By this, we mean that even the slightest change in visual input can result in radically different output. Our comparison of logs taken in the same scenario under the same lighting conditions made this apparent. The differences between these logs are imperceptibly minor, and yet they

would routinely produce outputs that disagreed on such important judgments as whether a ball was present in the image, how many black spots there were, and where those spots were located. Figure 3 in the appendix provides a nice illustration of the problem.

Second, the `processDarkSpots` function in `BallDetector.cpp`, which calls all of the filtering functions for black spots, should have a hard upper limit on the number of candidate black spots allowed to remain as candidates after it has been called. One might reasonably expect the presence of other robots in the image to increase the number of false positives detected, but in several instances, the tool marked more than 10 identified black spots, and in a couple of the cases, the figure exceeded 25 spots (see Figure 2). We can conceive of no practical circumstance in which a robot would benefit from information on this many spots *after* the filters have already been run. We suggest that `processDarkSpots` limit itself to returning only the three spots for which it has the highest confidence.

Third, the detector struggles to locate balls when they are on a field line and when they are more than a couple meters away. Neither of these is a new problem, but given the relative frequency with which a robot is likely to encounter one of these situations, it is imperative that we find a way of dealing with them. The hemisphere comparison may offer some help in the former case, but additional methods would be of great use.

Fourth, and as a corollary to the previous remark, the problem of detecting balls in the first place ought to be given greater priority than the problem of eliminating false ones. On the whole, we saw many more false negatives than false positives. Regardless of whether one thinks it is better to see no balls at all than to see a false one, this is a shortcoming of the system that should be addressed immediately.

Fifth, and finally, in light of the promising results of the hemisphere comparison check, we want to suggest two potential optimizations to our algorithm that could make it more feasible for use in games:

1. *Check fewer pixels.* Our algorithm checks all of the pixels in the inner diameter of a white spot, but we suspect that this is more information than is really needed. There are many ways one might reduce the number of pixels checked. The check for green above and below the ball, for example, considers just one radius's worth of pixels in either direction. If it should turn out that this is insufficient for the hemisphere comparison, one could try looking at several rows of pixels in each half. Either method would save considerable time.

2. *Execute only after other checks have been performed.* There is no sense in performing this comparison on all potential balls if other checks, such as `greenAroundBallFromCentroid` and `checkBallHasNoGreenAndSomeWhite` can reduce the pool of candidates more quickly or with greater confidence.

6 Conclusion

The purpose of our project was to determine whether brightness (Y-values) could serve as an effective filter in ball detection. In our experiment, we carried out three tests: The first compared statistics relating to the brightness of true positive and false positive black spots, and the second and third explored the difference in

brightness between the top and bottom hemispheres of the ball and the regions directly above and below it, respectively.

In the first test, we found that there was no significant difference between true and false positive black spots. In the second, ... And in the third, ...

Finally, we suggest that further efforts to improve ball detection not focus on quantitatively characterizing black. Our knowledge of previous work in this vein, in conjunction with the results of our project, have led us to conclude that color and brightness values simply do not provide the system with enough information to distinguish between the black pentagons of the ball and other dark spots in the image.

7 Reflection

We have learned a great deal over the course of the semester, and particularly, over the course of this project. To a veteran computer scientist, the knowledge we have gained will sound like mere truisms — and they are truisms. But it is one thing to be hear platitudes in the classroom and quite another to see their truth for oneself.

In many of the presentations on the last day of class, our classmates expressed considerable frustration in trying to understand the code base. We would echo this sentiment, but would add that we profited from the struggle. A computer science student at Bowdoin who does not participate in RoboCup and does not take this class may perfectly well graduate without having had to work with legacy code. This is a problem. A new graduate working as a software engineer will likely be doing little else *but* working with legacy code, and so must be comfortable navigating foreign territory. Having spent tens of hours this semester struggling to decipher source files that we did not write, we think it a shame that there are not more opportunities in the computer science curriculum to practice this skill, but are grateful to have been exposed to it at all.

We also gained an enormous appreciation for thorough testing. It became eminently clear to us, in dealing with the relatively low-quality sensors of the Naos and the fickle systems used to process sensory information (e.g. ball detection), that testing of the kind and quality typical in other classes we have taken was inadequate. Indeed, as we articulated in section 4, we believe our testing for this project is still inadequate. Regardless, processing information from nearly 1,000 logs gave us a better sense of the level of rigor necessary to produce good, useful data.

Finally, Bill's lectures on vision, in conjunction with the troubles we encountered in working on ball detection, evinced for us just how hard a problem computer vision is — particularly in environments like RoboCup where time and processing power are so limited. Although these considerations were not of primary importance to us in our project, the difficulty that the robots have in seeing the ball under low-stress conditions only underscored the significance of the challenge of strong game-time ball detection.

8 Effort and Grades

Whatever the ways in which we may have fallen short of our goals in this project or of expectations for the course more generally, it was not for lack of effort. In class, we paid attention during lecture, took good notes where appropriate, and did the assigned readings. For the three assignments that involved working with the robots, we did not procrastinate and were proactive in seek-

ing help when needed. For these reasons, we feel we deserve an A for our effort grade.

It is a much harder task to grade ourselves on the project itself. Doing so requires positing a set of criteria on which our work should be judged, which may differ, both in kind and in emphasis, from those our professors have in mind. It seems there is no avoiding this. So, putting effort aside for the moment, we will briefly discuss our performance with respect to three metrics we deem salient: the value of the work (to the team), the thoroughness of the work, and the amount we learned.

In showing that brightness cannot be used effectively to characterize black, but that it may still be useful in filtering out false balls, we believe we have made a small but meaningful contribution to the RoboCup team. With these results, Professor Chown and future RoboCup leaders can direct vision projects in more fruitful directions. We understand that producing something of utility to the team was not a requirement of the assignment, but think that the results of our experiments should nonetheless be taken into consideration.

However, these experiments could have been more thorough — not, perhaps, in the quantity of data taken, but in the kind and in the analysis of it. As noted earlier, we took all of our logs in the RoboCup lab, and we think the experiments would have benefited from greater diversity of location. The illuminance measurements were also not as precise as we would have hoped. Furthermore, we regret not having the time or knowledge needed to more thoroughly evaluate the statistical significance of our results.

Ultimately, projects such as this one are meant to teach, and this project has taught us much. We will not repeat all of the lessons we learned in its undertaking (q.v. section 7), but will reiterate that the project afforded us a valuable opportunity to learn to work with legacy code, to improve our testing practices, and to consider some important questions in computer vision.

Synthesizing these observations into a letter grade is a difficult exercise, but after discussing the matter at length and as objectively as possible, we think that a grade of A- for the project is fair.

	True			False		
Illumination (lx)	150	300	450	150	300	450
Mean	95.65	92.84	86.76	101.52	100.74	100.06
Median	93	94	84	102	99	101
Stdev	11.80	17.79	13.90	14.87	22.91	15.89

Figure 1: The mean, median, and standard deviation of brightness values for true and false positive black spots under lighting conditions of 150, 300, and 450 lux.

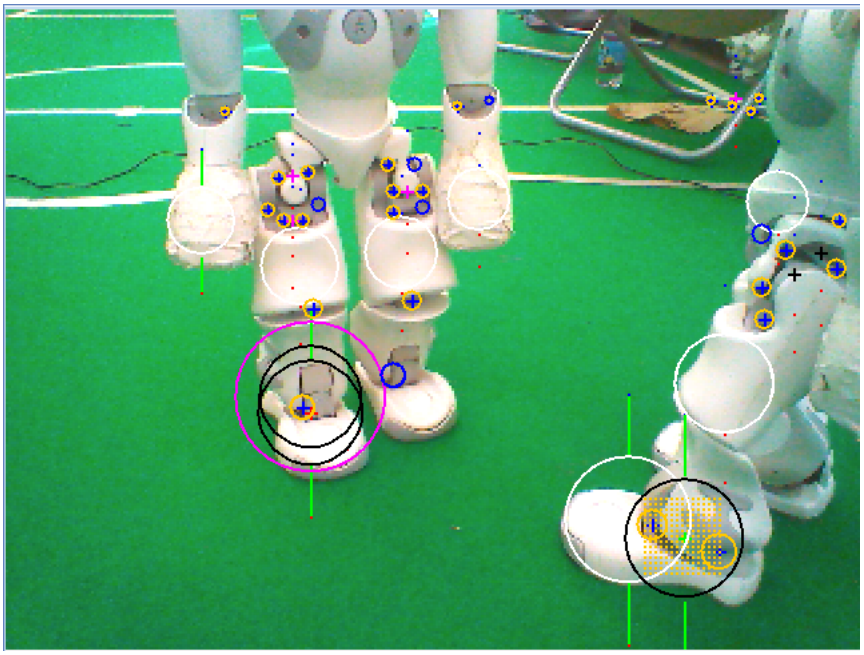

Figure 2: A record-breaking 25 black spots found in the joints of fellow robots.

```
/*******************************************************************************
 * Inputs:      the spot whose brightness is to be checked
 * Outputs:     the median brightness of the inner region of the spot
 * Description: the purpose of this function is to determine whether there
 *              is any significant difference between the brightness (y
 *              values) of true positive and false positive black spots.
 *              Given a spot, the function simply returns the median y value
 *              of all the pixels in its inner region.
 *******************************************************************************/
float BallDetector::getMedianBrightness(Spot spot) {

    // convert to raw coordinates
    int leftX = spot.ix() + width / 2 - spot.innerDiam / 4;
    int rightX = spot.ix() + width / 2 + spot.innerDiam / 4;
    int bottomY = -spot.iy() + height / 2 + spot.innerDiam / 4;
    int topY = -spot.iy() + height / 2 - spot.innerDiam / 4;

    int numPixels = 0;        // the number of pixels checked
    std::vector<int> yValues; // a vector for all the y values

    // iterate through all the pixels in the inner region
    for (int y=topY; y<=bottomY; y++) {
      for (int x=leftX; x<=rightX; x++, numPixels++) {

        // add the y value of the current pixel to the vector
        // of all y values
        yValues.push_back(*(yImage.pixelAddr(x,y)) / 4);
      }
    }
    // sort the y values
    std::sort(yValues.begin(), yValues.end());

    return yValues[yValues.size() / 2];

} // end getMedianBrightness


/*****************************************************************
 * Inputs:      the white spot to check
 *
```

```
 * Outputs:     the median brightness values of the top and bottom
 *              hemispheres, as a pair of doubles (these could just
 *              as well be integers)
 *
 * Description: compares the median brightness of pixels in the
 *              top half of the ball to that of pixels in the bottom
 *              half.
 ***********************************************************************/
std::pair<double,double> BallDetector::topOfBallBrighterThanBottomMedian(Spot spot) {

    // convert to raw coordinates
    int leftX = spot.ix() + width / 2 - spot.innerDiam / 4;
    int rightX = spot.ix() + width / 2 + spot.innerDiam / 4;
    int midX = spot.ix() + width / 2;

    printf("X: [%d, %d, %d]\n", leftX, rightX, midX);

    int bottomY = -spot.iy() + height / 2 + spot.innerDiam / 4;
    int topY = -spot.iy() + height / 2 - spot.innerDiam / 4;
    int midY = -spot.iy() + height / 2;

    printf("Y: [%d, %d, %d]\n", bottomY, topY, midY);

    // counters for the number of top and bottom pixels checked
    int topPixels = 0;
    int bottomPixels = 0;

    // vectors for the brightness values
    std::vector<int> topYvalues;
    std::vector<int> bottomYvalues;

    // put the y values of the pixels in the top half of the
    // ball into the topYvalues vector
    for (int y=topY; y<midY; y++) {
      for (int x=leftX; x<=rightX; x++, topPixels++) {

        if (debugBall) // show the region checked
          debugDraw.drawDot(x,y,WHITE);
        topYvalues.push_back(*(yImage.pixelAddr(x,y)));
      }
```

```
    }

    // put the y values of the pixels in the bottom half of the
    // ball into the bottomYvalues vector
    for (int y=midY + 1; y<=bottomY; y++) {
      for (int x=leftX; x<=rightX; x++, bottomPixels++) {

        if (debugBall) // show the region checked
          debugDraw.drawDot(x,y,BLACK);
        bottomYvalues.push_back(*(yImage.pixelAddr(x,y)));
      }
    }

    // sort both vectors
    std::sort(topYvalues.begin(), topYvalues.end());
    std::sort(bottomYvalues.begin(), bottomYvalues.end());

    // get the median brightness of each (have to divide by 4 because
    // of a weird property of pixels in the y image)
    float topMedian = topYvalues[topYvalues.size() / 2] / 4;
    float bottomMedian = bottomYvalues[bottomYvalues.size() / 2] / 4;

    // return the medians in pair form
    std::pair<double,double> medianBrightnesses =
            std::make_pair(topMedian, bottomMedian);

    printf("Top of ball has median brightness %f\n", topMedian);
    printf("Bottom of ball has median brightness %f\n", bottomMedian);

    return medianBrightnesses;

} // end topOfBallBrighterThanBottomMedian


/*****************************************************************
 * Inputs:      the white spot to check
 *
 * Outputs:     the median brightness values of rectangular regions
 *              directly above and below the ball.
 *
```

```
 * Description: This function uses the assumption that there will
 *              most likely be a shawdow below a ball, and thus that
 *              the area below the ball will be darker than the area
 *              above the ball. This function compares the median
 *              brightness of the two areas.
 ************************************************************************/

std::pair<int,int> BallDetector::aboveBallBrighterThanBelowBall(Spot spot) {

    int leftX = spot.ix() + width / 2 - spot.outerDiam / 4;
    int rightX = spot.ix() + width / 2 + spot.outerDiam / 4;

    //Top Rect
    int topRectBottomY = -spot.iy() + height / 2 - spot.outerDiam / 4;
    int topRectTopY = -spot.iy() + height / 2 -
            (spot.outerDiam + spot.innerDiam) / 4;

    //Bottom Rect
    int bottomRectTopY = -spot.iy() + height / 2 + spot.outerDiam / 4;
    int bottomRectBottomY = -spot.iy() + height / 2 +
            (spot.outerDiam + spot.innerDiam) / 4;

    // vectors for the y values of the above and below ball regions
    std::vector<int> topYValues;
    std::vector<int> bottomYValues;

    // Check Top Rect
    for (int x = leftX; x <= rightX; x++) {
        for (int y = topRectTopY; y <= topRectBottomY; y++) {
            if (debugBall) {
                debugDraw.drawDot(x,y,RED);
            }
            topYValues.push_back(*(yImage.pixelAddr(x,y)));
        }
    }

    // Check Bottom Rect
    for (int x = leftX; x <= rightX; x++) {
        for (int y = bottomRectTopY; y <= bottomRectBottomY; y++) {
            if (debugBall) {
```

```
                debugDraw.drawDot(x,y,BLUE);
            }
            bottomYValues.push_back(*(yImage.pixelAddr(x,y)));
        }
    }

    // sort both vectors
    std::sort(topYValues.begin(), topYValues.end());
    std::sort(bottomYValues.begin(), bottomYValues.end());

    // get the median y values for both regions
    int topMedian = topYValues[topYValues.size() / 2] / 4;
    int bottomMedian = bottomYValues[bottomYValues.size() / 2] / 4;

    // return an integer pair of the medians
    std::pair<int,int> medianBrightnesses =
            std::make_pair(topMedian, bottomMedian);

    printf("Area above ball has median brightness %d\n", topMedian);
    printf("Area below ball has median brightness %d\n", bottomMedian);

    return medianBrightnesses;

} // end aboveBallBrighterThanBelowBall
```

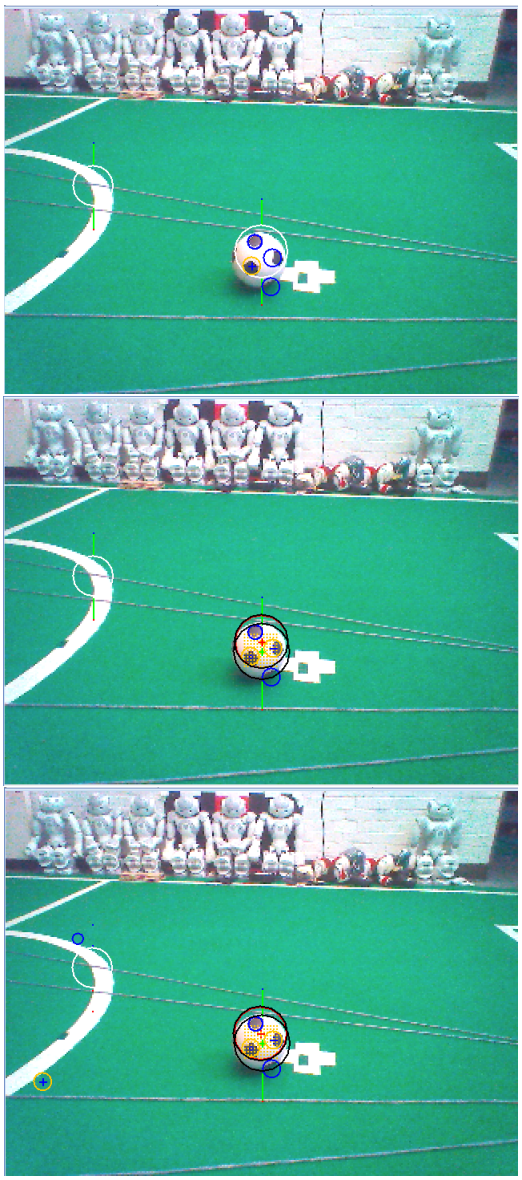

Figure 3: Three logs taken in the same scenario under lighting conditions of 150 lux. The blue crosses inscribed by yellow circles indicate identified black spots. The red circle (barely visible) in the bottom two images marks a positively identified ball. Note the marked difference in outputs from the ball detector among the three images.

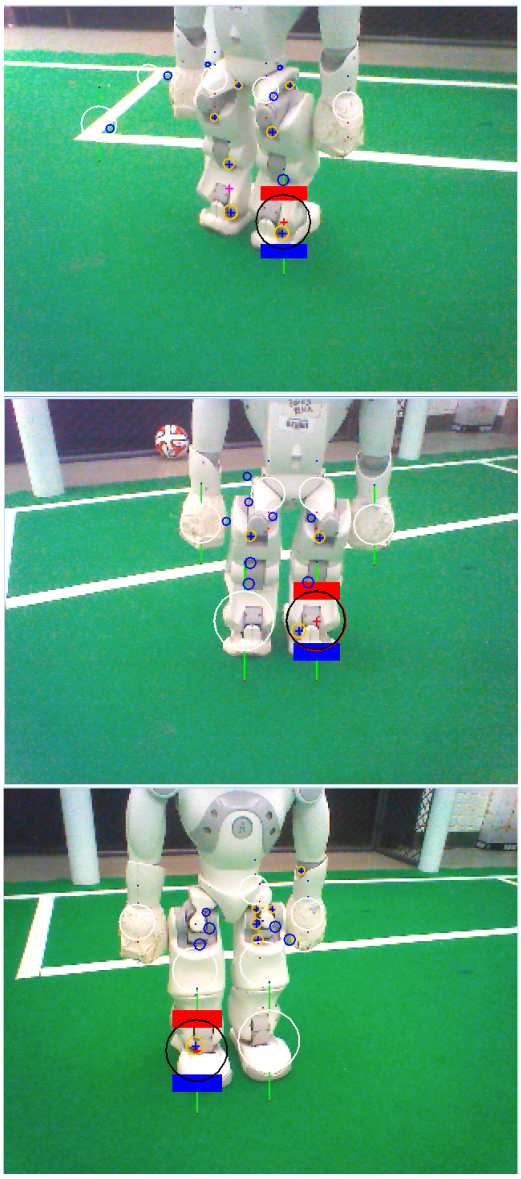

Figure 4: False balls detected in the feet and ankles of another robot. The red and blue rectangles indicate the areas checked by the brightness comparison of the regions directly above and below the ball. The hemisphere check was also performed, but is not visualized here. The latter method proved more effective at differentiating true and false positive balls.