THE UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Tracking a robot using overhead cameras for RoboCup SPL league

Jarupat Jisarojito

February 17, 2011

Supervisor: Bernhard Hengst

Abstract

This report describes the implementation of robot tracking system for RoboCup Standard Platform League (SPL). The robot needs to know where it is on the field so that it can play properly. However, the robot's camera only captures part of the field. Therefore, the robot localisation might not be accurate. The localisation can be improved using the robot ground truth information to adjust the localisation's parameters. The ground truth data is obtained by mount the cameras over the field and use those cameras to track the real position of the robot. The system uses 2 cameras mounted over the robot field. Each camera captures different half of the field. The process in each camera includes background subtraction, robot detection, position and heading calculation and image to real world coordinate transformation. The result of the system is the robot real position and heading.

Contents

1 Introduction

Robocup Standard Platform League (SPL) is a robot soccer where every teams compete using the same robot. The current robot is a humanoid robot called Nao. In order for the robot to play soccer, the robot needs to know where it is located on the field and where the ball is. Depending on the location, the robot will behave differently. The robot camera is in a way similar to human eye. The camera can only see some part of the field. Unlike human, the robot does not know where it actually is. The robot, however, can estimate where it is on the field using information from the camera, its vision system, and how it moves, its motion system. This information will pass through a localization filter. The filter uses this information to estimate where the robot thinks it is. However, the robot's localization is not accurate because the estimated position depends on what the robot sees through its vision system. Due to this inaccuracy, the robot might not behave the way we expected it to. Thus the performance is reduced.

The parameter in the localization filter parameter, such as constants or variances, can be adjusted to increase the localization accuracy. In order to do this, we need to compare the actual robot position or ground truth data with the robot localization data.

The aim of this project is to obtain the ground truth data of the robot. This ground truth data will have the robot real position and the robot heading.

The outline of this report is the following

Literature Review: This chapter will explore what research and implementation had been done.

Method: This section will explain the set up of the cameras that will be used to track the robot and the approaches taken to implement the tracking system.

Results: This section will describe the robot position and heading detected by the overhead cameras. It shows how accurate the position and heading compare to the robot actual position and heading.

Discussion: This section discusses about the cause of errors and how the project can be improved.

Conclusion: This section summarizes the whole report and gives some idea about the future work.

2 Literature Review

2.1 SSL-Vision: The Shared Vision System for the Robocup Small Size League

SSL-Vision is a default vision system for a robocup small size league. In robocup small size league, every team is allowed to mount cameras above or next to the field [5]. The process for each camera consists of image capture, color thresholding, runlength encoding, region extraction and sorting, conversion from pixel coordinate to real world coordinate, pattern detection and filtering and delivery of detection results via networks [9].

SSL-Vision uses YUV lookup table to convert YUV raw image to color threshold image. After thresholding, the program performs runlength encoding on threshold image. Using runlength encoding information, the program can extract regions and compute the bounding box and centroid of all regions. Lastly, the program sorts the regions by color and size.

The program maps pixel coordinate to real world coordinate using a camera calibration. SSL-vision uses Zhang's camera calibration technique, however a manual measurement of the camera's distance from the field is needed.

Finally, the program extracts the robot's identity, location and orientation as well as the ball's location. This part is based on CMDragon vision system[4].

2.2 A Global Vision System for a Robot Soccer Team

This paper describes a vision system for RoboRoos, robocup small size league team. This vision system also uses color thresholding, but unlike SSL-vision, RoboRoos vision uses RGB lookup table. The program does not identify the color outside the field to stop unnecessary process. Color threshold image is passed to thinning algorithm to remove any noise. The paper also introduces color normalization. Color normalization is used to handle non-uniform lighting across the field. Each region of interest is normalized by comparing the colour values in the region to the average colour values of the entire field [1]. The RoboRoos robots are identified using a long white strip and team marker. They introduce a stripness measurement. This stripness is calculated from the second moments. It is used to find the long white strip and calculate the robot orientation. To convert everything to real world coordinate, the RoboRoos use Tsai's camera calibration technique. After that, the robot's location and orientation from 2 cameras is merged. If the program finds two robots located within the predefined threshold distance, the program treats them as the same robot and final location is the average of the two locations.

2.3 Robust Adaptive Vision System for a Robot Soccer Team

This paper describes a previous version of RoboRoos vision system. This system has only one camera. The system used adaptive colour histogramming to remove the background, in this case the green field colour. It did not use frame different approach because the method is easily affect by changing of lighting or camera movement. After background removal process, the system applied thinning and dilation operation to the image to get rid of noise and fill in holes inside the blobs. The blob was tested against minimum threshold blob to make sure that the blob was big enough to be a robot. The system used blueness to detect the center of the robot. After that the program calculated the robot orientation using the centroid of the blob and the major axis detected from the robot marker. By using a marking scheme, the robot orientation and identification can be detected.

3 Method

Two cameras are mounted over the field. Each camera captures half of the field. Due to limited space, the cameras are not placed exactly straight over the field, but to the side of the field. This was done to make sure that the camera can capture whole half field.

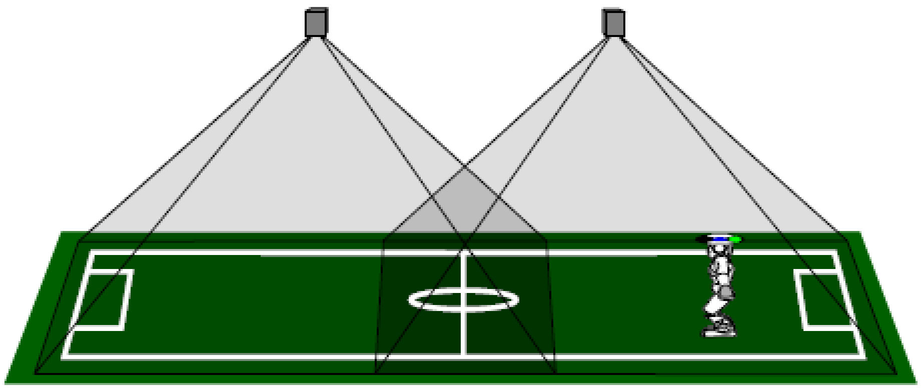


Figure 1: Illustration of how the camera should be placed. In real situation both camera are placed over the goal line.

3.1 Background Subtraction

To detect the object of interest, in this case, a robot, I needed be able to extract a robot from its background environment. I used frame difference to solve this problem. The background frame, where there would be no robot, is saved. Then, in every camera frame, the frame is subtracted by the saved background. Given that there is a robot in a new camera frame, then the different between a new image frame and the saved image would be a robot.

3.1.1 YUV Image

The image receives from the camera is a YUV:422 image. The frame difference method is applicable to any color space. Therefore, I chose to apply it to a YUV color space by using only y value in YUV color space as a threshold. The y value represents the luma or brightness of an image. However, the result was not as good as I expected, because in some lighting condition, the brightness of the field and a blue color in robot's head are very similar. This causes the threshold image to have some missing robot body.

3.1.2 RGB Image

The next approach to solve this problem is to convert the YUV image to RGB color space and then find the difference. The difference between camera frame and background image would still be in RGB color space. To be able to find a threshold image, the difference image is converted from RGB to gray scale image. From this gray scale image, I can specify a threshold value which would result in a threshold image.

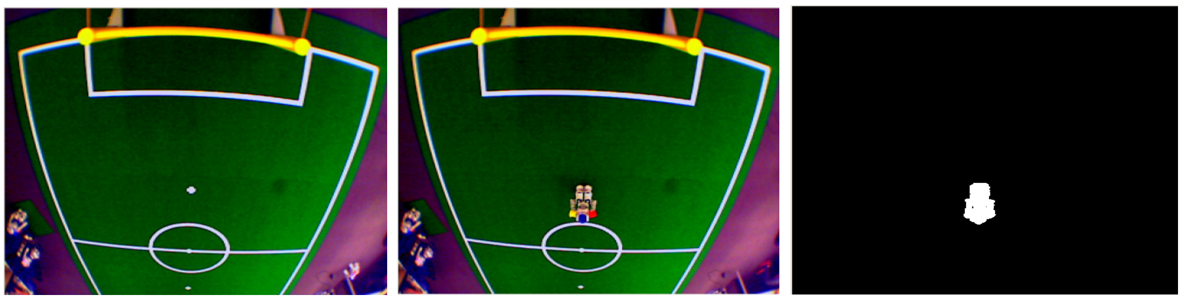


Figure 2: Background image (left). Current camera image (middle). The result of subtraction in threshold image (right).

3.1.3 Mask Image

The cameras mounted over the field have a wide angle lens to capture the whole half field. This wide angle lens also captures areas that are outside the field. This means that the camera would capture something that is not the object of interest, such as people legs. To get rid of this problem, a masking is applied. The area in an image which the robot would not be is masked out. There is no background subtraction in masked areas. The masked area would be color as black in a threshold image.

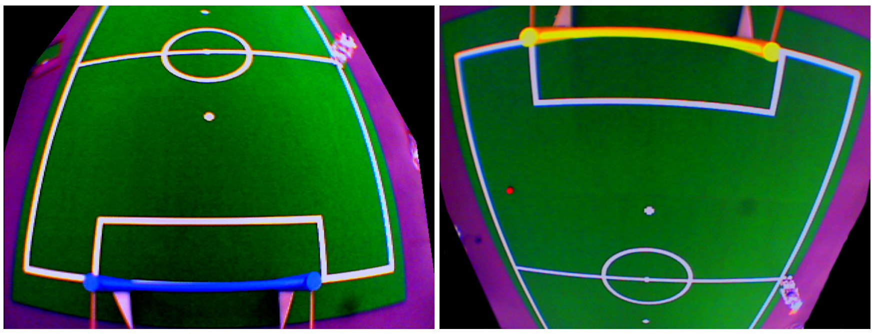


Figure 3: Black color is the masked out area. Left is blue goal camera. Right is yellow goal camera

3.1.4 Image Filter

The threshold image is not appropriate to use without filtering first. The filter which is applied to the threshold image is dilation. The dilation will fill any holes which might appear inside a blob. It will also expand the blob to make sure that the whole robot body is detected. The blob in this case represents objects detected in a frame difference method.

3.2 Robot detection

3.2.1 Robot Settings

The program will detect the robot position and heading correctly when the robot has the following settings. The head of the robot must be blue. The blue color is used to detect the robot position. The robot right shoulder is colored with red and the robot left shoulder is colored with yellow. The shoulders are used to detect the robot heading.

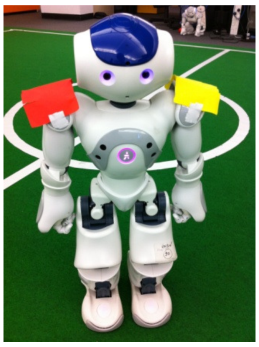


Figure 4: Real robot set up

3.2.2 Connected Component Labeling

From background subtraction, the threshold image is passed to a blob detection algorithm to detect all possible blobs that would be a robot. This is the first step for blob detection. The algorithm for labeling the connected region uses 8-connectness. The implementation follows the algorithm in [3]. This step results in a number of all blobs detected.

3.2.3 Analyze the blob

The threshold image is further scanned to get the size of each blob region, bounding box for the region. The region which is too small to be a robot is discarded. As mentioned in 3.2.1, the robot position and heading are calculated using the robot's blue, red and yellow region. Thus the regions are scanned for those three colors. In order to scan for these colors, the threshold image is converted to a hue image. All pixels, which are not background, are converted to its hue color representation.

After that, each blob region is scanned again for the biggest blue, red and yellow region inside the blob. These color regions must also be larger than the threshold region value to be considered as the robot's color parts. For this step, I can further remove the blob which does not contain the blue region, because this blob cannot be a robot. The entire remaining blobs are considered to be possible robots.

3.3 Position and heading calculation

The next step is to use those color regions to calculate the robot position and heading. The blue region, which would represent the robot head, is used to calculate the robot position. The centroid of the blue region will be the robot position. This centroid is, however, in the image coordinate. To get the robot position, the centroid in the image coordinate is transformed to the world coordinate using method in 3.4.

The centroid of red and yellow region is also mapped to world coordinate before doing heading calculation. To calculate the robot heading, I relied on the fact that the red region is the robot's right shoulder and the yellow region is the robot's left shoulder. This means that if I draw a line joining the centroid of blue region with the centroid of red region and find the angle of this line with respect to the global coordinate. I would have an angle that is 90 degrees out of phase with the robot's heading. As a result, adding 90 degrees to this angle gives the robot's actual heading. The same approach is applied to the blue and yellow region. However, the yellow region is on the robot's left, thus the resulting angle is subtracted by 90 degrees.

Using this method, in a condition which the robot standing sideway and the camera only capture one side of the robot, the heading can still be calculated. If both red and yellow colors are detected, the resulting heading is averaged from the heading calculated using red region and the heading calculated using yellow region.

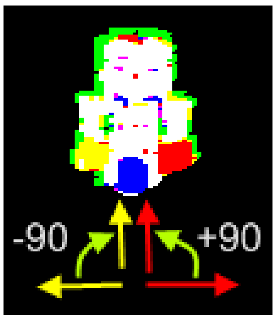


Figure 5: Image with classified colour. The red line represents a line between blue and red centroid. The yellow line represents a line between blue and yellow centroid.

3.4 Image to World coordinate transformation

3.4.1 Points calibration

The method I used to transform image coordinate to world coordinate is points mapping and points interpolation. Basically, the field is divided into a grid. Every grid points are mapped with corresponding image coordinate. This requires points to be mapped manually.

The half field is divided into 11x7 grids. The grid is 400 mm long in y direction and 500 mm long in x direction. The robot is placed on the grid point. The grid points are a world coordinate. It will be mapped with the centroid of the robot blue color region in image coordinate. However, some edge points might not be mapped because the robot's head is outside the camera image.

As a result of this mapping, there are some image points which are not mapped with world coordinate. To convert non mapped image points to world coordinate, the world coordinate is estimated using interpolation from the surrounding points.

3.4.2 Point Interpolation

I used barycentric coordinate [2] to interpolate the point. The first step is to find the three closest mapped points in image coordinate to the query image points. Using barycentric coordinate formula, the weight between the query point and the surrounding three points can be found.

$$\det(T) = (y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)$$

$$\lambda_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{\det(T)}$$

$$\lambda_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{\det(T)}$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2$$

Figure 6: Barycentric coordinate formula

where (x, y) is image coordinate of the query point.

(x1, y1), (x2, y2) and (x3, y3) are image coordinate of the three closest points.

λ are the weight of the three closest points.

Because those three points are mapped with world coordinate and the weight of those points are known, I can interpolate the world coordinate using the following formula.

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3$$
$$y = \lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3$$

Figure 7: Barycentric coordinate formula

where (x,y) is world coordinate of the query point

(x1, y1), (x2, y2) and (x3, y3) are world coordinate of the three closest points.

λ are the weight of the three closest points, which are found from previous step.

4 Results

To test the accuracy of the position and heading, the robot was placed at 15 random positions around the field.

The unit of position coordinate is in millimeter. The error of the position is calculated using Euclidean distance between the real position and detected position. The blue goal camera has a little more error. This might be because the blue goal camera is a little bit further from the field compare to the yellow goal camera. Thus when the robot is near the center of the field, the color parts is very small to be detected correctly.

Camera	Average Position Error
Blue Goal	104.9573
Yellow Goal	90.82454

Table 1: Average position error

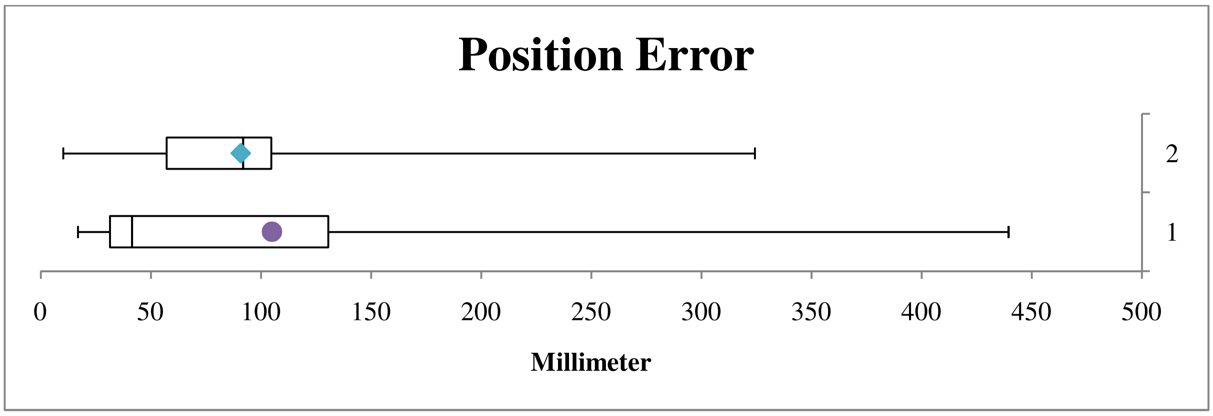


Figure 8: Box plot of heading error.1 is a camera on blue goal side of the field. 2 is a camera on yellow goal side of the field

As for the heading, because I have no accurate way of measuring the robot heading except from estimating from my eyes, the angle I used to test for heading accuracy is 0, 45, 90, 135, 180, -135, -90 and -45 degrees. The robot is also placed randomly on the field with random headings chosen from those 8 possible headings. Average heading error from each camera is calculated.

Camera	Average Heading Error
Blue Goal	17.6592
Yellow Goal	13.22699

Table 2: Average Heading Error

Figure 8 shows a box plot of the heading error. The ends of the whiskers represent minimum and maximum heading error. The dot of the graph shows an average error.

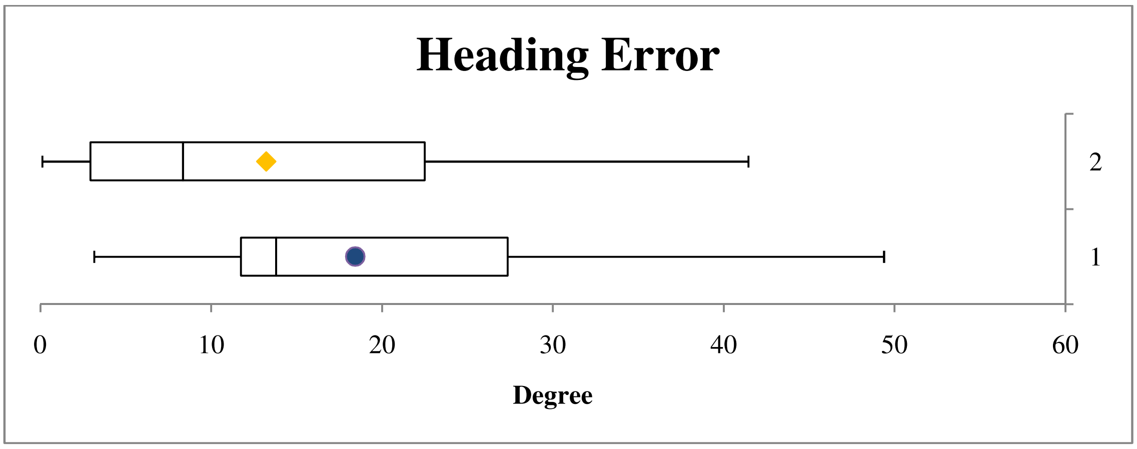
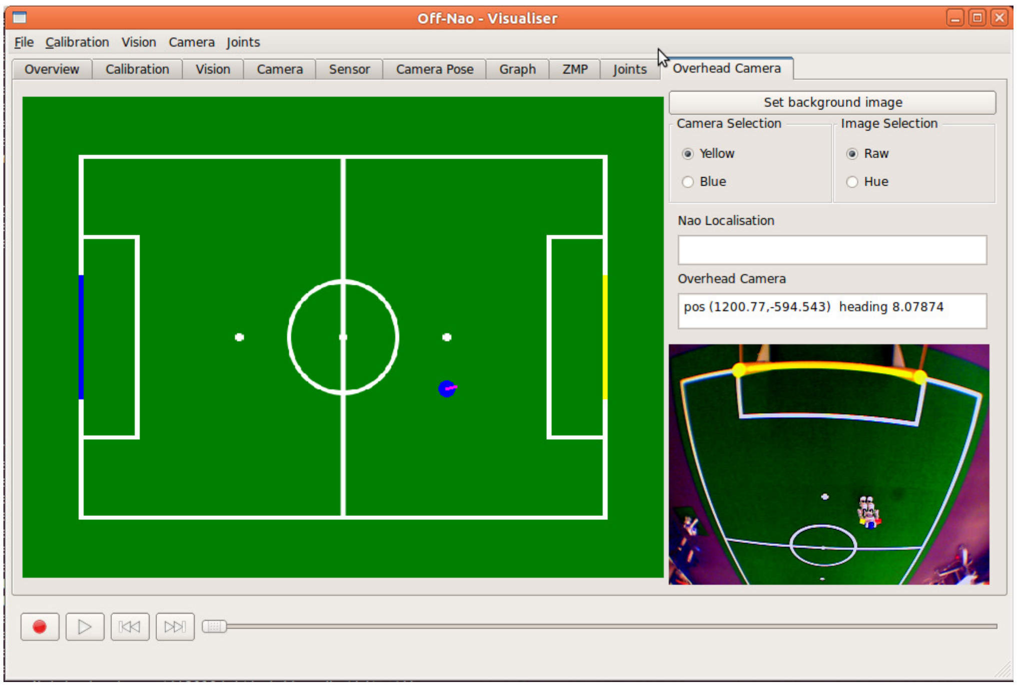


Figure 9: Box plot of heading error.1 is a camera on blue goal side of the field. 2 is a camera on yellow goal side of the field



Off-Nao - Visualiser
File Calibration Vision Camera Joints
Overview Calibration Vision Camera Sensor Camera Pose Graph ZMP Joints Overhead Camera
Set background image
Camera Selection
Yellow
Blue
Image Selection
Raw
Hue
Nao Localisation
Overhead Camera

pos (1200.77,-594.543) heading 8.07874

Figure 10: Overhead Camera Interface in offnao

5 Discussion

The program can properly identify the robot position and heading. However, in some area of the field where the lighting is very strong, the program sometimes cannot detect the robot position. This is due to the light reflection. Strong lighting and light reflection cause the robot color parts to look white from the camera image. In a case like this, the program cannot detect the robot position because it cannot detect the blue color region. The future work would be to make the program track a robot without color parts.

One reason that reduces the accuracy of the heading is the image to world transformation. The centroids of red and yellow region are transformed using the same mapped points as the blue region. However, the red and yellow regions, which are the robot's shoulder, have different height from the robot's head. This means that the interpolated world coordinate of red and yellow region centroids are not accurate as the blue region.

One possible method to solve this problem is to implement camera calibration and geometric model for the robot and a camera. Camera calibration uses to calibrate 2 dimensional points, but the robot is 3 dimensional object. Thus the model is needed to consider the effect of the robot's height to the point transformation.

6 Conclusion

This project implements a system to track a robot. The system would detect a robot real position and heading. The system uses robot blue head to detect the robot and uses red and yellow shoulders to calculate the robot heading. The result of the tracking system can be used to improve the robot localization. By comparing the robot localization and ground truth information, the localization parameters can be adjusted so that the localization result matches the ground truth data.

7 References

1. Ball, David, Gordon Wyeth, and Stephen Nuske. "A Global Vision System for a Robot Soccer Team." *Proceedings of the 2004 Australasian Conference on Robotics and Automation* (2004).
2. "Barycentric Coordinate System." *Wikipedia, the Free Encyclopedia*. Web. 14 Feb. 2011. <http://en.wikipedia.org/wiki/Barycentric_coordinate_system_(mathematics)>.
3. "Connected Component Labeling." *Wikipedia, the Free Encyclopedia*. Web. 14 Feb. 2011. <http://en.wikipedia.org/wiki/Connected_Component_Labeling>.
4. J, Bruce, and Veloso M. "Fas and Accurate Vision-based Pattern Detection and Identification." *Proceedings of the 2003 IEEE International Conference on Robotics and Automation* (2003).
5. "RoboCup Small Size League." *Small Size Robot League - Start*. Web. 14 Feb. 2011. <http://small-size.informatik.uni-bremen.de/>.
6. Wyeth, Gorden, and Ben Brown. "Robust Adaptive Vision for Robot Soccer." *Proceedings of the IEEE International Conference on Robotics and Automation* (2000).
7. Y, Tsai R. "A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses." *IEEE Journal of Robotics and Automation* (1987).
8. Zhang, Zhengyou. "A Flexible New Technique for Camera Calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2000).
9. Zickler, Stefan, Oliver Birbach, Mahisorn Wongphati, and Manuela Veloso. "SSLvision: The Shared Vision System for the RoboCup Small Size League." *RoboCup 2009: Robot Soccer World Cup XIII* Lecture Notes in Computer Science 5949 (2010): 425-36.

8 Appendix

8.1 Program Option

Set background image

This is required because if there is a change in the background image, the user can set new background image easily. Also, if there is a change in lighting, the background image needs to be set too. This is due to the fact that the program is easily affected by change in lighting.

Select the camera and type of image display

I cannot get two cameras working at the same time because both cameras are using the same USB bus. As a result, the user needs to manually select which camera to use. The image type selection is used for debugging. The raw selection displays raw image from the camera. The hue selection shows foreground objects detected and their hue color display.

8.2 Code Design Explanation

This section will briefly describe the function of some important class. So that the reader will have general idea how each class is related to one another and to edit or modify the code. Note that the program processes each frame separately. The information of the robot detected by one camera is also independent from the other camera.

8.2.1 OverheadReader Class

This is class inherit from Reader class. It emits newOverheadData. When newOverheadData is emitted, OverheadCameraTab class will call redraw function and drawOverlay function to display new information in offnao.

I cannot use NetworkRead or RecordReader because the first one requires offnao to connect to the robot; the second one requires a dump file.

8.2.2 OverheadCameraTab Class

This is the main display tab. When it receives newOverheadData, redraw function, drawOverlay function and drawOverlayNao are called. Inside redraw function, run function of OverheadCamera class is called. DrawOverlay draws robot overlay on the field from overhead camera information. DrawOverlayNao draws robot overlay when there is a local copy of robot localization.

When it receives newNaoData, it saves the robot localization position locally. The redraw function is not called. This is because, if the redraw function is called, the field is cleared and will display only localization data. When newOverheadData calls redraw, the field is cleared again and diplays only overhead data. This result is a blinking of the robot overlay drawn on the field.

8.2.3 OverheadCamera Class

The main function of this class is run function. Inside the run function, the process of background subtraction, image filter, blob detection, image to world coordinate transformation and position and heading calculation are called.

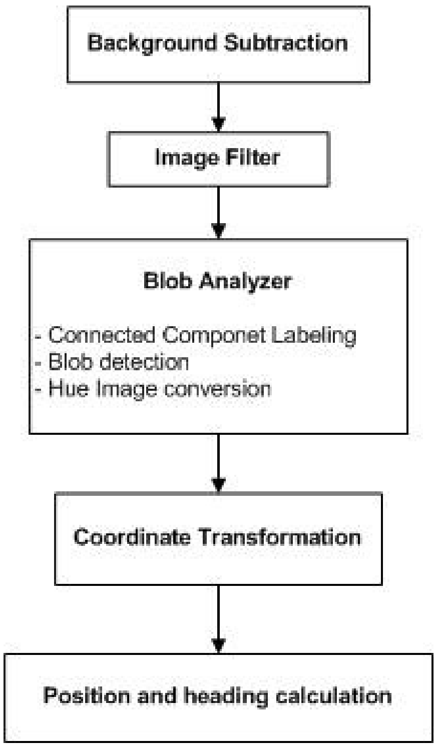

Background Subtraction
Image Filter
Blob Analyzer
- Connected Componet Labeling
- Blob detection
- Hue Image conversion
Coordinate Transformation
Position and heading calculation

Figure 11: Processing stack for OverheadCamera run function

8.3 Measure Results

8.3.1 Blue goal camera results

Trial	Real x	Real y	Real heading	Measured x	Measured y	Measured heading
1	-2400	-110	90	-2383.72	-1108.78	104.214
2	-2400	1100	-135	-2374.86	1090.94	-138.755
3	-2050	1540	45	-2000	1775.76	15.2983
4	-1110	1520	0	-1071.07	1506.13	-13.397
5	-2400	0	-90	-2376.06	60.1695	-69.4185
6	0	600	-180	34.7692	609.723	-130.625
7	0	0	-45	-14.4881	8.75724	-50.8905
8	-1200	0	135	-1107.36	0	164.62
9	-1890	690	0	-1864.31	689.816	12.1921
10	-1720	-850	180	-1683.62	-860.932	164.383
11	-2635	-1765	90	-2697.68	-1330.04	101.602
12	-1380	-1750	45	-1275.53	-1663.53	-
13	-1210	-615	0	-1173.07	-633.904	-3.15223
14	-650	0	135	-419.355	0	-171.399
15	-500	-950	-135	-381.059	-990.051	-122.39

8.3.2 Yellow goal camera results

Trial	Real x	Real y	Real heading	Measured x	Measured y	Measured heading
1	1200	0	-45	1226.66	8.91795	-29.5451
2	2400	-1100	-90	2415.29	-1049.44	-89.889
3	2400	1100	-135	2408.67	1105.43	-132.246
4	0	0	-180	104.126	25.5319	138.56
5	0	600	135	105.346	610.063	104.282
6	0	-600	90	93.75	-562.5	92.7216
7	600	0	45	924.242	0	-56.2321
8	2400	0	0	2412.43	2.28713	3.11271
9	2600	1630	-45	2687.4	1593.66	-52.6776

10	1180	1570	-90	1237	1545.57	-65.1596
11	490	1360	-135	583.001	1324.13	-132.118
12	400	-1640	-180	505.319	-1553.19	-
13	1240	-1650	135	1264.23	-1621.02	125.967
14	2650	-1660	90	2688.81	-1607.64	121.959
15	1830	-640	45	1905.94	-569.585	41.2505