Real-Time Visual Robot Detection and Modeling with Situational Awareness

Juan Fasola
School of Computer Science
Carnegie Mellon University

Advisor: Professor Manuela Veloso

May 21, 2006

Contents

Abstract

Visual object recognition and world state modeling is a challenging problem for a mobile robot. This problem is even more difficult on the Sony AIBO robot platform in the domain of robot soccer, which must rely only on local sensor information retrieved from a single monocular camera with limited viewing angle. This thesis focuses on problems in the domain of robot soccer, and has three main contributions:

- The creation of a method for the visual detection of teammate and opponent robots on the playing field as seen from the camera of an AIBO robot. The locations of robots are detected within the image, which are then used to calculate the distances and bearings to each of the identified robots. We present visual results of the detection method as well as an evaluation of the reported distance/bearing estimates with empirical data.

- The modeling of the position and motion of the detected robots as reported by the visual detection method. Modeling the motion of the observed robots includes continually estimating the velocity vector after sufficient information has been collected. Experimental results on the accuracy of the modeling system are provided.

- The creation of game behaviors which make use of the introduced teammate and opponent robot modeling to increase situational awareness and make better strategic decisions, in addition to improving team game play in general.

This research was initiated within the CMDash robot soccer team, whose 2002 predecessor became the world champion of the international RoboCup competition. All parts of our research, including the detection, modeling, and behavior algorithms have been incorporated into the CMDash'06 team, demonstrating that the solutions provided indeed have a significant impact on the research of multi-robot teams in challenging environments, such as robot soccer.

Chapter 1

Introduction

Robot soccer presents a variety of challenging problems for a mobile robot, especially when no accurate global models of the environment are provided from an external source such as an overhead camera. Problems common to robot soccer include: real-time visual perception of objects, world modeling, localization, multi-agent coordination, and strategy assessment. This thesis first presents a solution for the visual recognition of teammate and opponent robots in the context of robot soccer, and then provides an effective method for modeling the presence and motion of these detected robots. We then introduce behaviors that take advantage of this information to make better strategic decisions and improve team game play in general.

1.1 RoboCup Soccer

Our domain of research is the international RoboCup competition, specifically the Four-Legged League, where teams of autonomous robots play soccer against one another. The goal behind this annual competition is to encourage and promote research in the areas of artificial intelligence and robotics. The robot game environment is a playing field that is 540 cm in length and 360 cm in width with green carpet surface. There are two goals on opposite sides of the field from each other, both measuring 80 cm wide and 30 cm tall. Four uniquely colored landmarks are positioned on the sides of the field to help the robots localize their positions on the field. The game is played with a small orange plastic ball, and each team has four players. Wireless communication between robot teammates is acceptable, however all processing must be performed onboard the robots and no human intervention is allowed. Figure 1.1 shows a model of the playing field.

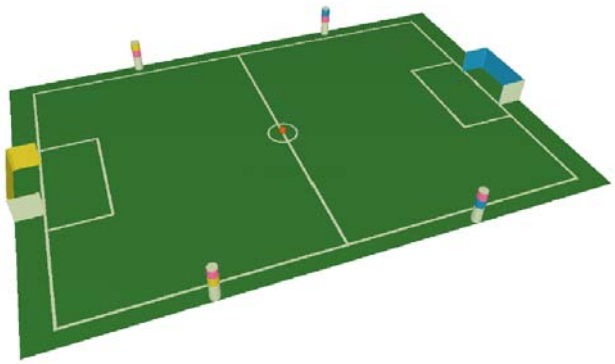


Figure 1.1: The robot soccer playing field

1.2 Robot Platform

The robot platform used in this research is the Sony AIBO ERS-7 Entertainment Robot, which is a quadruped robot designed to look like a small dog. The robot design is shown in Figure 1.2. The robot contains a total of twenty degrees of freedom with three in the head, three for each leg, and the remaining five in the tail, mouth, and ears. There is a single 64-bit 576 MHz MIPS processor on the robot which is used for all onboard computation. The robot also has 64 MB of RAM, two infrared distance sensors, a wireless wavelan card, and a CMOS camera located in its head that it uses for all visual sensing. The camera provides images at a resolution of 208x160 pixels and has a field of view of approximately 55º, which combined with the 180º pan of the head allows the robot to effectively scan the entire area in front of it. The robot is designed to be fully autonomous, with the ability to use its sensors to perceive the world around it and onboard computation to select and execute actions.

Figure 1.2: The Sony AIBO ERS-7

1.3 Overview of CMDash on board Modules

There are five modules that are responsible for the autonomous functionality of the robot: vision, localization, world model, behaviors and motion. The vision module is responsible for the perception of the environment, namely by identifying and reporting distances and bearings of field objects after analyzing the images returned by the onboard camera. The localization module estimates the location and orientation of the robot on the field by using the information on the relative distance and angle of the landmarks. When no markers are visible the location of the robot is estimated based on the robot's last known position, the motions executed and the sensor readings. The world model module is responsible for gathering information and modeling the locations of field objects that are not static, such as the ball and other robots.

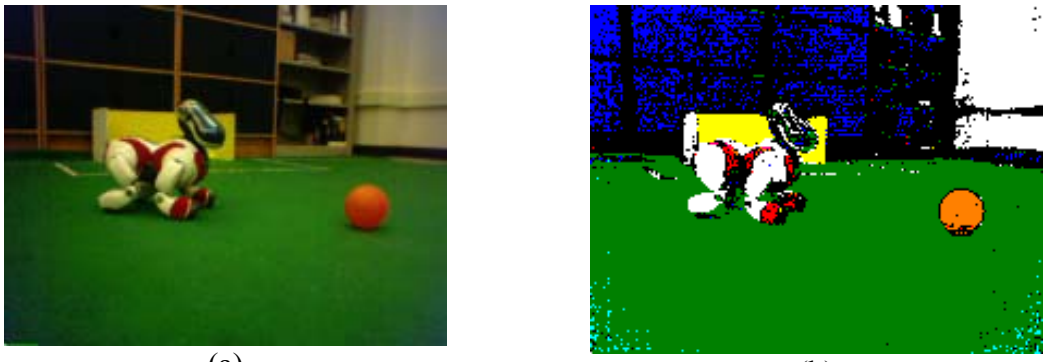
The information provided by the localization and world model modules is used by the behaviors to select the next action to execute, such as walking forward or kicking. The motion module interprets the command set by the behaviors and determines how to manipulate the joints in order to carry out the command.

1.4 Vision Module

All visual information about the robot's environment is gathered from the onboard camera located in the head of the robot. The camera outputs new vision frames at approximately 30 fps. In order to take full advantage of this onboard capability, the robot must be able to perform all necessary computation on the current vision frame before receiving the next vision frame. This computation includes not only performing all visual detection methods for the various objects on the field, but also updating localization estimates, world model information, running behaviors, and outputting motion commands to the motion module. In short, the visual processing must not only run in real-time, but run in about one third of the time it takes for a new vision frame to be recorded just so the other computation can be completed; in practice, this amounts to approximately 11 milliseconds. Furthermore, within the visual processing, many separate objects must be detected, which further limits the time allowed for any given detection method.

1.4.1 Color Segmentation and Region Creation

RoboCup provides objects of interest that are of discrete pre-defined colors, so as to simplify the task of visual detection of these objects. Our vision module is separated into both low-level and high-level processing. The low-level processing handles the conversion of the given vision frame from YUV-color space into the discrete color space of RoboCup colors, which include black, white, green, orange, yellow, cyan, pink, red, and blue. The conversion is achieved by performing a simple lookup in a color calibration table that identifies which discrete color is mapped to the given YUV color. An example conversion is shown in Figure 1.3. Although the table lookup runs very fast, the creation of this color table is extremely tedious and involves manually labeling pixels from images captured from the robot on the field under the same lighting conditions as expected during a game. The low-level processing also groups neighboring pixels of the same discrete color into color regions so as to facilitate detection methods at higher level processing. The high-level vision processing makes use of the segmented image and color regions while performing the detection methods for all the objects of interest, such as the ball, landmarks, goals, field lines, and robots.



(a) (b)

Figure 1.3: (a) RGB image of vision frame captured from the camera of an AIBO (b) Color segmented image result from the same vision frame

Chapter 2

Robot Detection

In the RoboCup soccer environment there are specific objects of interest whose positions on the field, when known by the robot, can be very helpful in making appropriate strategic decisions. For example, the robots on the field must be able to visually detect the location of the orange ball in order to move towards it in an attempt to perform some action on it, like kicking in the direction of the opponent goal or passing to a teammate. Another example are the landmarks on the sides of the playing field, whose locations relative to the robot help the robot determine where it is located on the field and how it is oriented, and at the same time, in what direction the opponent goal is located. The objects of interest that we are concerned about are the teammate and opponent robots. The detection and modeling of teammate and opponent robots on the field provides for the creation and execution of a variety of different behaviors and actions the robot can perform in response to such information. In this chapter we will describe our method of visual detection of robots and present experimental results on its effectiveness.

2.1 Visual Robot Detection Problem

The AIBO robots are white, in the shape of a small dog with four legs and a moving head. In a robot soccer game, the robots wear special "uniforms," which are colored patches of different (strange) shapes that cover parts but not all of their white body. The patches are either red or blue to denote the team but all the robots in a team are indistinguishable. Figure 2.1 shows examples of robots in their colored uniforms.

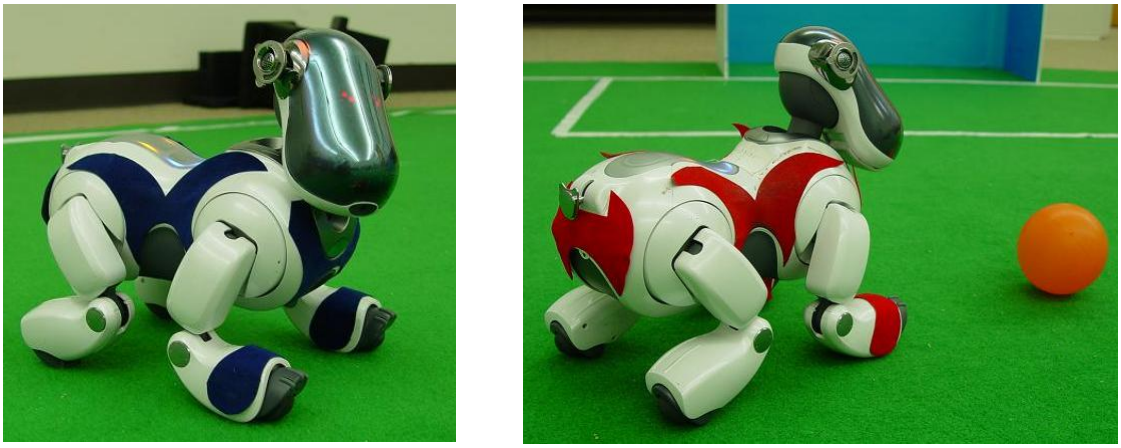
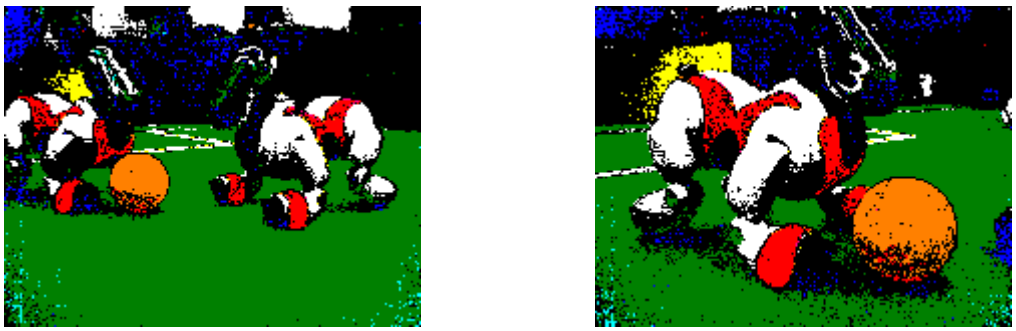


Figure 2.1: AIBO robots with RoboCup colored uniforms for both teams

One way to address the problem of robot detection is to search for red or blue blobs in the image, and if the blobs conform to some constraints, a robot is detected. In theory, this seems to be a reasonable solution, however the color of the blue uniforms is so dark that in the color segmented images they appear to be black. Since the background color is usually black, and there are a lot of shadows and segmentation noise that is also black, finding blue robots in the image with this technique is not as easy as it seems. If one were to try and correct the segmentation to emphasize the recognition of the color blue of the uniforms, then a lot of black pixels would become blue, and the problem would still remain. Also, there is more than one uniform patch on any given robot, and so determining whether or not blobs are part of the same robot or multiple robots correctly is non-trivial. Furthermore, the uniforms are only one aspect of the robots, and intuitively it seems that by using only the uniform color information, much information is ignored that would otherwise help to greatly improve the detection of robots. For these reasons we have decided not to use uniform colored blobs for robot detection, but focus rather on finding objects located on the playing field. The objects detected on the playing field are assumed to be robots, as the only objects on the field during game play, besides the ball, are robots. Figure 2.2 shows images taken from the robot's camera that are typical during game play, and constitute examples of the visual problem space.



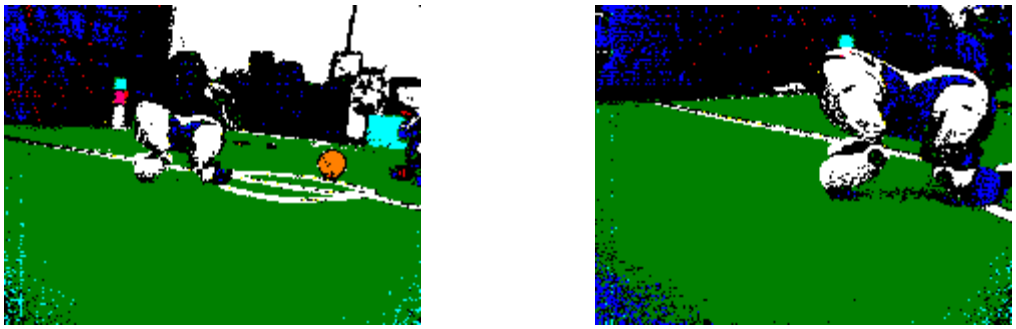


Figure 2.2: Color segmented images of vision frames taken from the robot's camera similar to those during actual game play

2.2 Robot Detection Algorithm

All the images that need to be processed come from the head of an AIBO that is standing on the field, therefore certain assumptions about the image are made to aid and speed up the detection of objects that are considered to be on the field. There are three main assumptions:

1) Field horizon: There exists a horizontal line that best separates the playing field from everything above it, called the field horizon. Every pixel below this horizon line belongs to either part of the field or something on the field.

2) Objects not green: Objects on the field, such as robots, people's legs and other things, are not green in color since the field is green and a distinction must be made between objects on the field and the field itself. Therefore, every pixel below the field horizon that is not green is part of an object that is located on the field.

3) Objects intersect field horizon: Objects standing on the field will always intersect the field horizon. This assumption is made because the robots to be detected will most likely always be standing, and from the point of view of another standing robot the playing field cannot be seen over the robot being viewed.

By using these assumptions the detection of robots in the image can be processed much more rapidly than if these assumptions were excluded.

There are three main steps in the robot detection algorithm, which are:

- Preprocessing the segmented image
- Finding field horizon location
- Identifying discrete objects (robots) below field horizon

An overview of the robot detection algorithm is presented in Figure 2.3.

Algorithm I: FINDROBOTS (*image*)

min ← ∞
RESCALEIMAGE(*image*)
for each column *col* in *image*

```
    row ← FindStartOfGreen(image, col)
    if  row < min
        min ← row
fieldHorizon ← min
for each column col in image
    objs_col ← FindObjectEnd(image, fieldHorizon, col)
return FormRobotBoundingBoxes(fieldHorizon, objs)
```

Figure 2.3: Algorithm for creating initial hypothesis windows for robot locations within image frame.

2.2.1 Preprocessing the Segmented Image

Before the robot detection algorithm can begin searching for robots within the segmented image, it must be preprocessed in order to speed up the detection process. There are two steps in the preprocessing, which are:

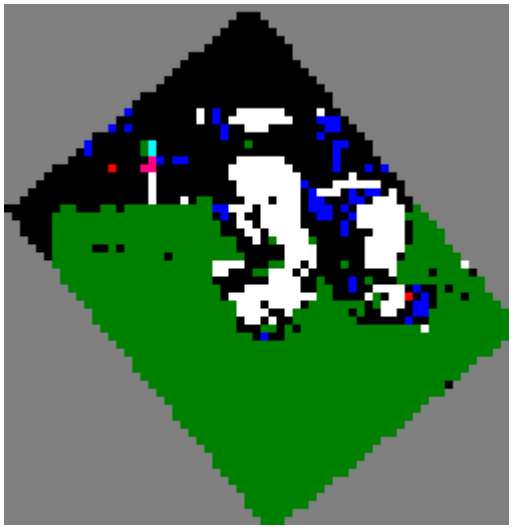
1) Rescaling segmented image: The color segmented image is scaled downwards by a factor of four along each dimension using a simple nearest neighbor approach, reducing the original 208x160 resolution image to 52x40. An example of image reduction is shown in Figure 2.4(a).

2) Rotate image to world space coordinates: The robot contains three degrees of freedom in its neck and head, therefore when the camera is rotated the image coordinates do not correctly correlate with world coordinates. In order to ensure that the "up" direction in image space is the same as the "up" direction in world space (perpendicular to the ground plane), the segmented image is rotated by the opposite amount of the camera rotation. Correcting for the rotation of the camera allows the algorithm to remain robust to different head positions and gaze directions. An example of image rotation is shown in Figure 2.4(b).

(a)

(b)

(c)

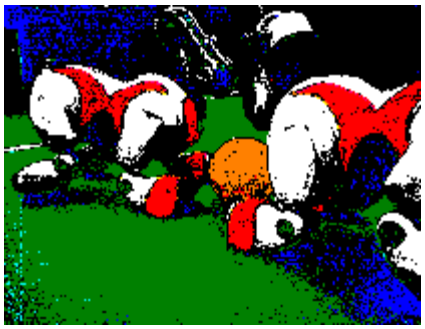


(d)

Figure 2.4: (a) Original 208x160 color segmented image from robot camera
(b) Reduced 52x40 image of same view as in (a)
(c) Reduced segmented image of vision frame captured from rotated camera (head)
(d) Reduced image corrected for rotation of camera in (c) – ready for processing

2.2.2 Finding the Field Horizon

The field horizon as described earlier is the line that best separates the background from the playing field in the image. The playing field will always be green in color, and the background will most likely not be. Therefore, finding the field horizon in the image constitutes scanning the image columns from top to bottom, searching for the start of green pixels that denote the beginning of the playing field. The start of the playing field is found once four green pixels are detected sequentially down an image column. Once all columns in the image have been scanned and the starting points of the field are recorded, the highest point in the image where the start of green is recorded denotes the location of the field horizon. Only one point is necessary to determine the field horizon line, as the image has already been rotated to world coordinates, so the field horizon line will always be a horizontal line. Figure 2.5 shows an example of the detected start of field points along each column and the final detected field horizon line. If there are no start of field points in a given image, i.e. the field is not visible, then the field horizon line is set to be at the top of the image. When the field horizon is at the top of the image, the whole image is interpreted as being an obstacle on the field, which in many cases is true in this scenario, i.e. the robot is off the field looking away from the field, or there actually is another robot obstructing the view of the robot.



(a)

(b)

(c)

Figure 2.5: (a) Original color segmented image of vision frame
(b) Reduced image of same vision frame as in (a)
(c) Reduced image with detected field horizon and preliminary field points shown

2.2.3 Detecting Obstacle Regions

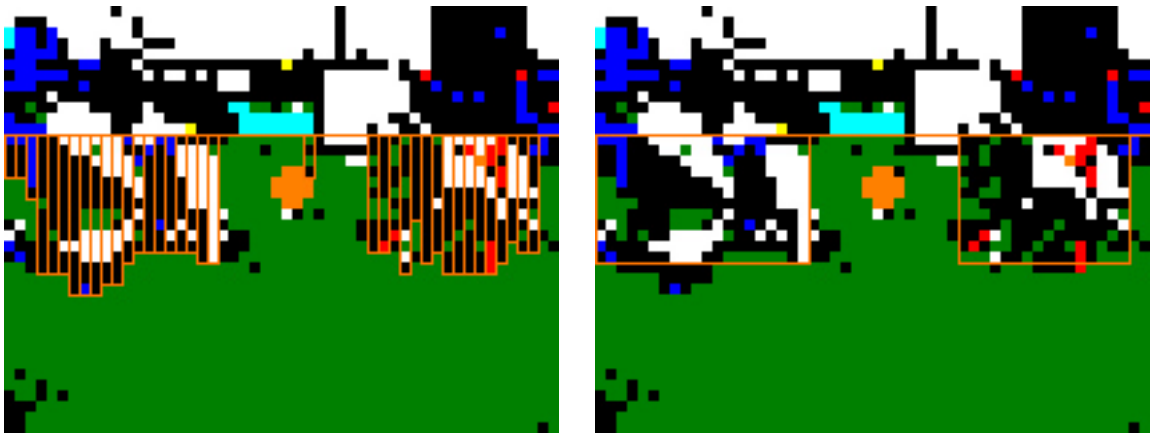
Often times field lines and the center circle of the field, which are white, are seen in the color segmented image and tend to interfere with the grouping of obstacle regions, causing bounding boxes of obstacle regions to be too wide or to combine with another obstacle region created one large bounding box. For this reason, an attempt is made at removing the field lines and center circle from the image before the bounding boxes for obstacle regions are formed. The field lines and center circle tend to be represented in the reduced segmented images as quick transitions between green (field color) to white (line color) and then back to green. Usually only one or two white pixels are contained in the transition. Therefore, the image columns are scanned below the field horizon for these specific transitions, and if detected, the white pixels are relabeled as green pixels, giving the impression of the field lines being part of the green playing field. Another aspect of the segmented images that tends to interfere with the grouping of obstacle regions are the cyan color shaded borders of the image. These artifacts are created due to a chromatic distortion on the edges of the raw YUV image taken by the robot's camera, where the colors on the edges take on a more blue-ish hue, causing the segmentation to present a fair amount of cyan colored pixels on the borders, instead of perhaps green, from the field color. Like the field lines, the cyan pixels are converted to green. This

process is ok since the objects on the field, i.e. the ball and the robots, do not contain the cyan color. Figure 2.6 shows an example of the process of removing the field lines and cyan edges from the segmented images.

(a) (b) (c)

Figure 2.6: (a) Original color segmented image from robot camera
(b) Reduced image with no further processing
(c) Reduced image with field lines and cyan borders removed

After the field lines and cyan edges have been removed, the objects located on the field can be easily identified. Using the assumption that robots are always on the field, we can search below the field horizon for pixels that are not green (not field) and not orange (not ball) and group them to form obstacles, which will eventually become our detected robots. Each column in the image is scanned below the field horizon searching for the start of a sequence of green/orange pixels, at which point the scanned length along the column is recorded and scanning is initiated on the following column. The amount of green or orange pixels necessary in a sequence is different depending on where the sequence lies in the image. For example, if there is one green pixel immediately below the field horizon, that is enough to abort scanning along that image column, however if only one green pixel were found further down the column away from the field horizon, it wouldn't be enough to stop scanning. Generally, if three green pixels are found in sequence, the scanning is aborted, or if four green pixels are found, in sequence or not. This procedure essentially finds the length along each column of an obstacle that intersects the field horizon line. Columns with small lengths are thrown out to prevent creating obstacle regions on account of image noise. Neighboring columns with lengths a reasonable amount apart are grouped together. The lengths of all the columns within a group are averaged to find the length for that group. Using this length, along with the starting and ending column positions, a bounding box is created for each group; these bounding boxes represent the obstacles located on the field. Resulting bounding boxes that are too small or in some way inadequate, i.e. too "skinny" or too lengthy (the width and height dimensions are multiple factors apart), are removed. The final bounding boxes are considered to encompass robots and represent the output of the robot detector. An illustrative interpretation of the column scanning and obstacle grouping are shown in Figure 2.7.



(a) (b)

Figure 2.7: (a) Reduced image with initial column scan results for obstacle colors shown (b) Reduced image with final robot bounding boxes after grouping neighboring object columns

2.3 Robot Color and Distance Estimates

The team color is determined by searching for red regions within the bounding box of the detected robots. If convincing red regions are found within a robot's bounding box, the robot is said to be on the red team; blue team otherwise. The distance to a detected robot is calculated by taking the center pixel of the bottom line of its bounding box and projecting a ray through it onto the ground plane. The distance to the intersection point is considered to be the distance to the robot. The robot detector is able to detect robots up to a maximum distance of approximately 1.2 m away.

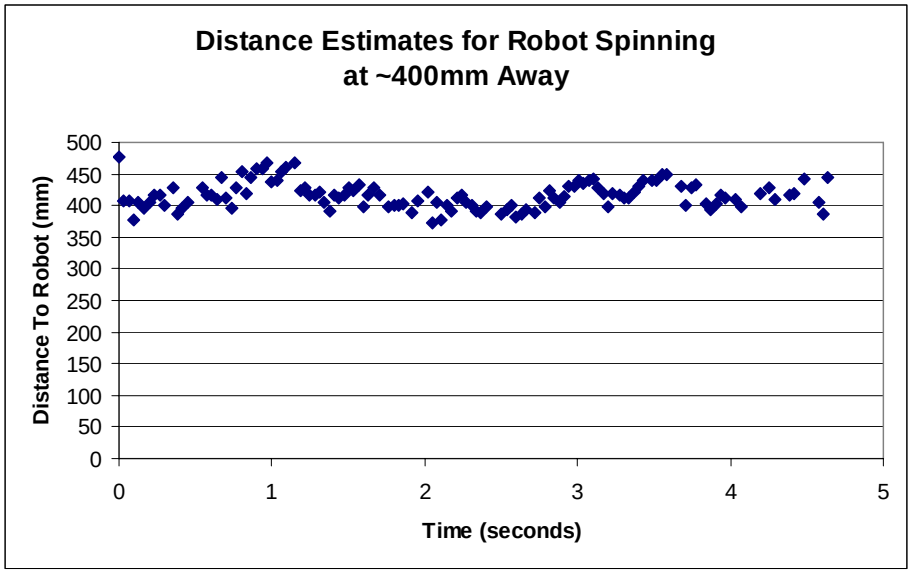
2.4 Results

To test the effectiveness of the robot detection algorithm we ran two experiments: the first tests the ability to detect robots at multiple different views at varying distances, and the second tests the detection of robots at various locations in the image.

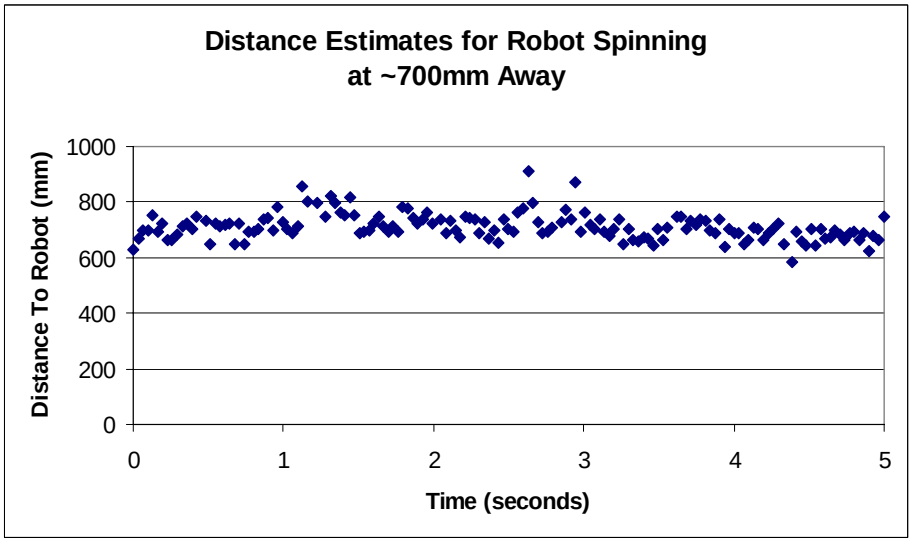
2.4.1 Experiment 1

The first experiment involves a standing robot observing another robot spinning in-place directly in front of it at three separate distances: 40 cm, 70 cm, and 1.2 m away. The robot being observed is spinning in-place in order to test the detector's ability to recognize robots at all possible orientations, while the standing robot maintains a fixed head position facing forward during observation. Experimental results are shown in Figure 2.8.

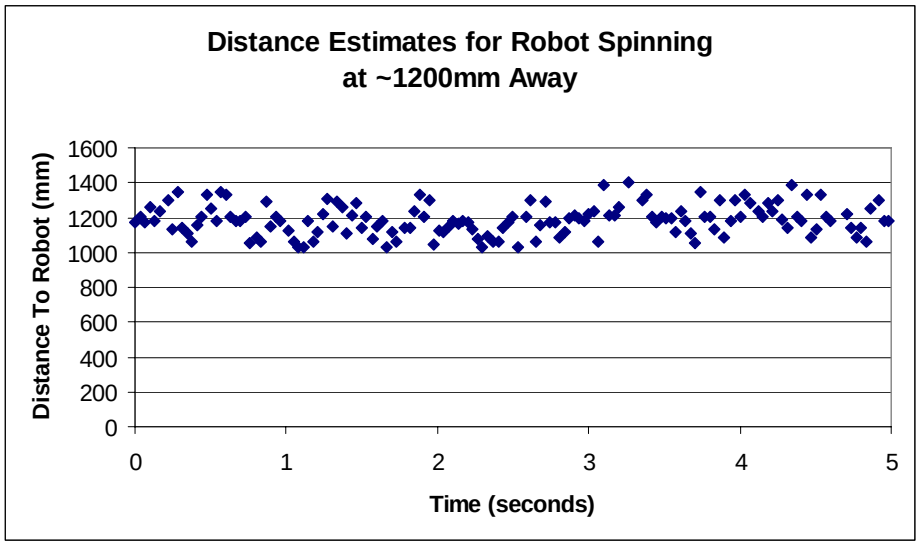
(a) (b) (c)


Distance Estimates for Robot Spinning
at ~400mm Away
Distance To Robot (mm)
500
450
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
Time (seconds)

(d)


Distance Estimates for Robot Spinning
at ~700mm Away
Distance To Robot (mm)
1000
800
600
400
200
0
0
1
2
3
4
5
Time (seconds)

(e)



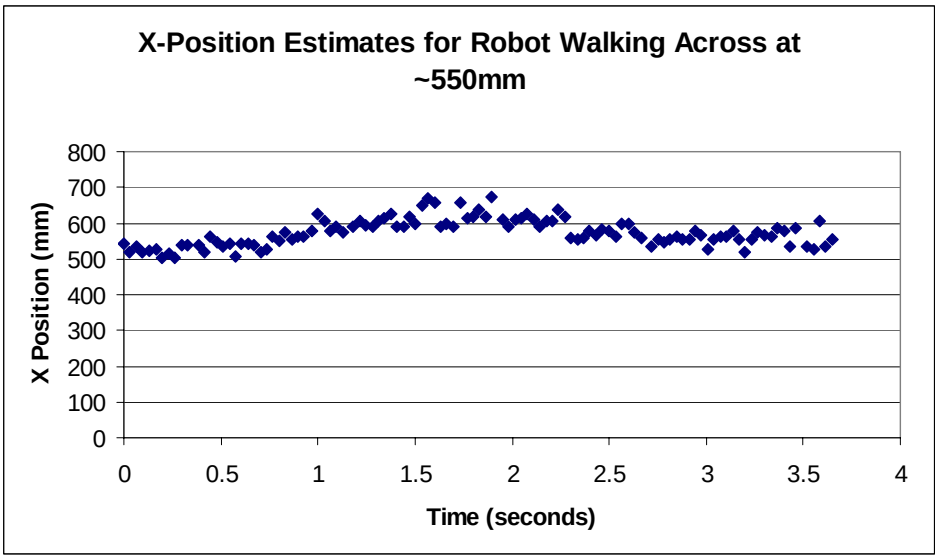
(f)

Figure 2.8: (a) Robot at ~40 cm away (b) Robot at ~70 cm away (c) Robot at ~1.2 m away (d), (e), (f) Graphs of recorded distance estimates for observed spinning robot

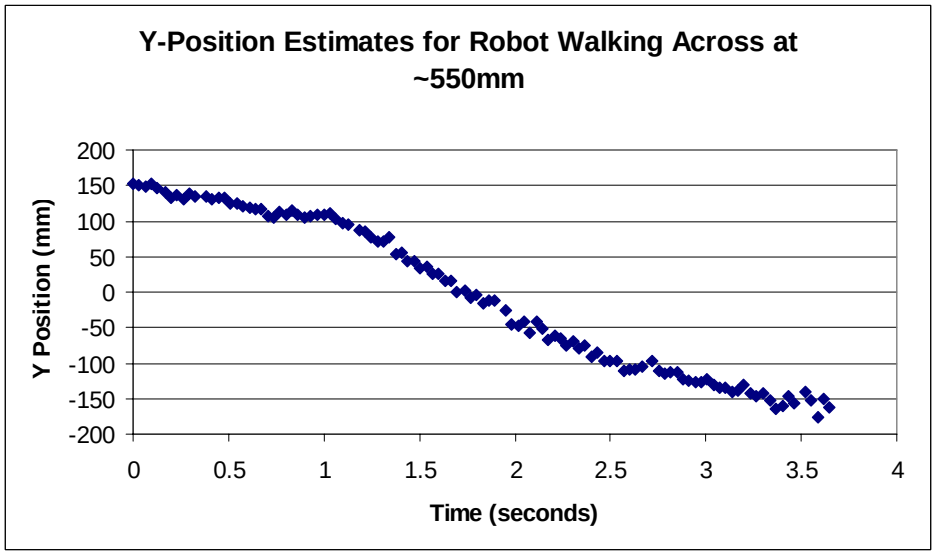
2.4.2 Experiment 2

The second experiment involves a standing robot observing a moving robot directly in front of it walking from left to right along a line a distance of approximately 550 mm from the robot. Figure 2.9 shows the results which include both the x and y coordinates of the detected robot during the time of robot crossing. The recorded (x,y) positions are in body local coordinates, where the x-axis represents forwards/backwards displacement and the y-axis represents lateral displacement. The standing robot maintains a fixed head position looking forward for the duration of the experiment.

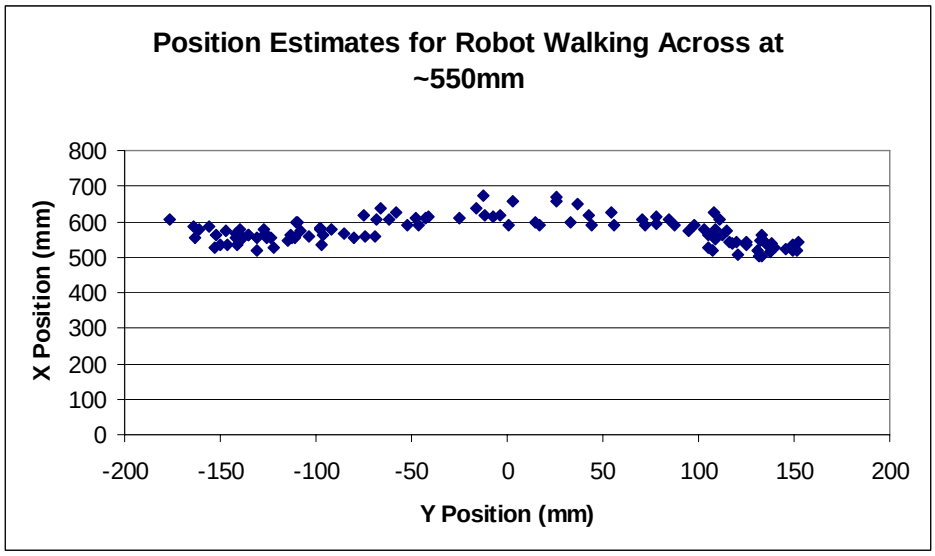
(a)


X-Position Estimates for Robot Walking Across at ~550mm
X Position (mm)
800
700
600
500
400
300
200
100
0
0
0.5
1
1.5
2
2.5
3
3.5
4
Time (seconds)

(b)


Y-Position Estimates for Robot Walking Across at ~550mm
Y Position (mm)
200
150
100
50
0
-50
-100
-150
-200
0
0.5
1
1.5
2
2.5
3
3.5
4
Time (seconds)

(c)



(d)

Figure 2.9: (a) Images of robot walking in front of viewing robot at ~550 mm away
(b) Recorded x-position estimates for walking robot during crossing
(c) Recorded y-position estimates for walking robot during crossing
(d) Recorded robot location estimates during time of robot crossing

2.4.3 Visual Results

Figure 2.10 provides more visual results of the robot detector, which consist of images not belonging to any experiment, but showcasing interesting qualities such as the ability to detect multiple robots in an image, detect close robots, far robots, combinations of both, and effectively determine the team color. The red team is represented by an orange bounding box, and the blue team by a green bounding box.

Figure 2.10: Segmented images with visual robot detection results displayed

Chapter 3

Robot Modeling

The visual robot detection system has been shown to provide fairly accurate estimates on the ego-centric locations of teammate and opponent robots found within any given vision frame. However, by the very nature of the vision system, the detection results are discarded upon the arrival of a new vision frame and the detection process is re-initiated. The loss of robot location information prevents the gathering and retrieval of valuable information about the game state. By using the detection results to form models of other robots on the playing field, the robot is able to move its head freely to visually detect and track other objects such as the ball, and yet still be aware and react to the location of previously detected teammate or opponent robots. In addition, modeling visually detected robots allows for the retrieval of information not available from just a single vision frame, such as velocity estimation and prediction of motion. In this chapter we will discuss how the robot detection results are used in modeling the position and motion of teammate and opponent robots on the playing field.

3.1 Updating Robot Models/Merging Robot Detection Results

Every vision frame presents new robot detection results which must be incorporated into the global model of the game state and potentially matched against previous observations of robot locations. Merging the robot detection results with current models of teammate and opponent robots is necessary to avoid situations where two or more robot models are created which pertain to the same physical robot. Such situations are undesirable for two reasons:

1) The world model ceases to accurately represent the current game state
2) The robot models are not able to incorporate new locations into their position history, thus limiting the accuracy of the motion model

One method of modeling robot locations which does not involve merging subsequent detection results is to project the current field of view of the robot's camera onto the playing field of the world model, remove all robot observations contained within the viewing area, and replace them with new robot detection results from the current vision frame if any exist. This method of robot modeling avoids the merging/matching problem by relying on the notion that when the world model determines that there are teammate or opponent robots located in the field of view of the robot's camera, the new robot detection results will provide the most updated locations for such robots, and

therefore allow for the previous observations to be discarded beforehand. While this method provides a simple mechanism for modeling robot locations and avoids, for the most part, creating multiple models for single physical robots, it does not maintain a position history for the robots being modeled and hence does not provide enough information for velocity estimation/motion modeling. Since we are interested in modeling the motion of teammate and especially opponent robots, we have not adopted this simplified modeling method and instead have implemented a version which maintains a history of recorded positions on the playing field for each modeled robot and hence performs the necessary merging/matching of new robot detection results with those made previously.

3.1.1 Update Algorithm

Merging new robot observations with previous observations from the world model involves matching robot models with new robot position estimates, or rather, identifying which robot model, if any, should get updated with a new position estimate. Robots are represented in the world model as (x,y) locations in the global coordinate system of the soccer playing field, however for matching purposes it is also necessary to take into account the physical dimensions of the robot. All robots on the playing field are approximately 20cm wide and 35cm long. Since no information regarding the orientation of visually detected robots is available, robots are modeled as circles with radius equaling 2*[maximum dimension/2] = 2*[35cm/2] = 35cm, centered at the estimated (x,y) location on the playing field. The modeled robot radius is double the actual physical robot radius to account for error in the location estimates.

The following is an outline of the robot model update algorithm:

- The time-to-live counter is decremented for all robot models in the current field of view of the robot projected onto the playing field. The time-to-live counter represents the number of frames allowed that contain the robot model inside the projected field of view, but not detected from vision, and is initially set to 10 frames. Once the time-to-live counter reaches zero, the robot observation is removed from the world model, as it is deemed inaccurate. This field of view reasoning is similar to the one used in the simplistic method presented earlier, however instead of immediately removing robot observations that are within the projected field of view, the time-to-live counter is introduced to provide robustness against errors in the visual detection method and field of view estimation.

- All robot observations returned by vision are compared against the robot models already existing in the world model, in order to match those observations that correspond to the same physical robot. In finding a match for a new observation, all potential matches of robot models are evaluated and the best of them all, the one most resembling the new observation, is updated with the new position estimate. A robot model is considered a potential match with a new observation from vision when the distance between their respective locations on the field is less than two robot radius lengths. The procedure iterates through the robot

models and the first potential match found is considered to be the best match until another potential match is found, at which point the two are compared against one another and the one that is the better fit with the new observation is made the new best match. This procedure continues until all robot models that are potential matches have been evaluated, and the best among them determined. The algorithm for comparing two potential matches takes into account the distance between both robot models and the visual observation location, the team color of each and that of the visual robot, and the position history size of the robot models (larger is better). If both robot models match up equally with the visual observation, then the best match is kept the same.

- If a robot model is found to match the new observation returned by vision, its position is updated in the world model. However, if no existing robot model matches the visual observation, then a new robot model is created in the world model at the given location and initialized with a timeout value of 4 seconds. The timeout value is provided in order to remove robot models considered out of date. It should be noted that every time a visual robot observation is matched with an existing robot model, the timestamp is updated and the timeout is essentially reset, so it is possible for any given robot model to exist within the world model for longer than 4 seconds, as long as its position is updated and existence reaffirmed from visual observations.

3.1.2 Robot Model Update Results

Visual results of updates to robot models in the world model are shown in Figure 3.1. The results are recorded observations from a standing robot tracking a moving robot, accomplished by moving its head to try and keep the robot in the center of its field of view. The segmented image frames are shown along with the world model state, which shows the field of view region during the visual observation and the location of the robot model, represented by a circle with an X inside it, for the captured frame. The results provide visual evidence of the ability of the modeler to match new visual robot observations with existing robot locations in the world model and perform position updates accordingly, which is essential to tracking the motion of detected robots and maintaining an effective model of the world state.

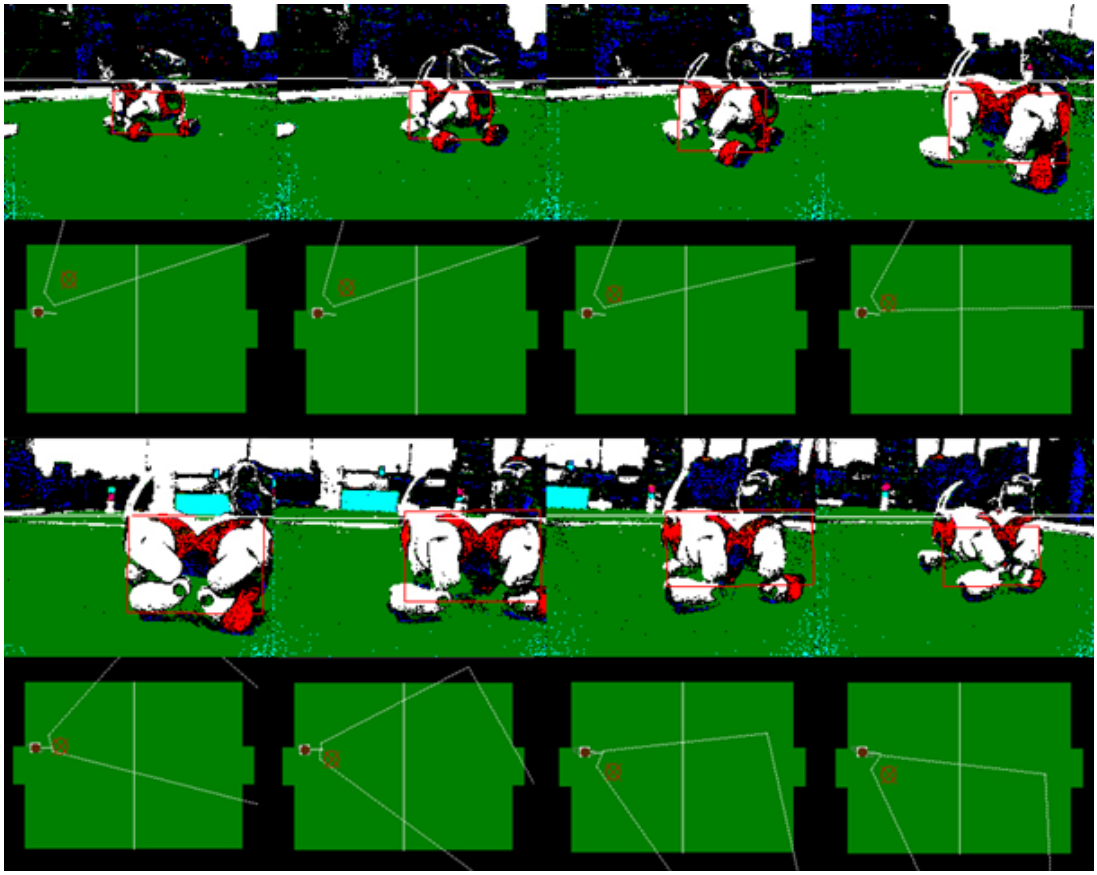


Figure 3.1: Illustrative results showing visual robot detection output and corresponding world model state for a sampling of vision frames from a standing robot viewing another robot walking in front of it from left to right

3.2 Motion Tracking

Tracking the motion of robots in the world model is useful for many aspects of the game, but most importantly, by providing information about the movement of robots on the field the current situation of the game can be more readily assessed. For example, by having a robot observe the motion of an opponent robot walking towards the ball and correctly predict that the opponent will arrive to the ball first, the robot can decide to assume a more defensive role that would try to prevent the opponent from getting an accurate shot on goal, rather than to continue walking towards the ball and potentially getting scored on. In order to achieve this motion and speed assessment of robots on the playing field, velocity estimates are calculated based on the observed sequence of locations through time of the robots being tracked.

3.2.1 Robot Position History

Each time a new visual robot observation is matched with an existing robot model, the updated position of the robot is stored in the position history list for that robot model, along with a timestamp denoting the time at which the position was recorded. Keeping a history of robot positions with an associated timestamp is essential to the calculation and estimation of the robot velocity. There is a maximum allowable size to the position history list, which is meant to allow for enough observations to accurately estimate velocities while still ensuring that only the most recent data is used to improve reliability. The maximum allowable size is set to the frame count equivalent of 2 seconds, so in other words, calculating the velocity of teammate or opponent robots only uses, at a maximum, the last 2 seconds of recorded position data. In order to keep the data within the position history list as current as possible, whenever a new position is added and the list was previously at its maximum capacity, the oldest recorded robot position is removed from the list.

3.2.2 Velocity Estimation

In order to facilitate the velocity calculation on a robot model, the robot position history is actually separated into two lists, the x-position list and the y-position list. Each x or y position entry in the lists are associated with the time it was recorded. To avoid the calculation of spurious or noisy velocity estimates, the position history lists must contain the minimum number of recorded positions before any velocity calculation is performed for the robot model. The minimum number of positions is equal to 10, and is intended to represent about one-third of a second of motion information.

The velocity calculation is performed as follows:

- *x-velocity*: The slope of the least-squares fitting line through the (timestamp, x-position) data points is calculated and represents $\Delta x/\Delta t = vx$, the estimated velocity of the robot in the x-direction.
- *y-velocity*: The slope of the least-squares fitting line through the (timestamp, y-position) data points is calculated and represents $\Delta y/\Delta t = vy$, the estimated velocity of the robot in the y-direction.

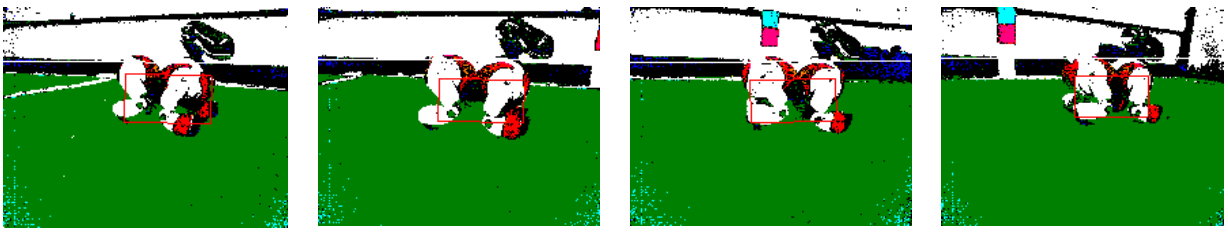
The calculated x and y velocities are combined to produce the velocity vector estimate for the robot model, and this is the vector that is reported to behaviors. The velocity vector, once attached to the current position of the robot model, can be used to predict the motion of the robot and enhance situational awareness during game play. Another method would be to predict the future location of the robot based on the calculated least-squares line fit over the data, by finding the x and y positions along the line at a certain time in the future. It should be noted, however, that velocity estimates are only available for robot models that are currently within the field of view of the robot, since no assumptions can be made about robots moving outside of view and we want to minimize error in the estimates as much as possible so as not to provide false information to behaviors.

3.2.3 Velocity Estimation Results

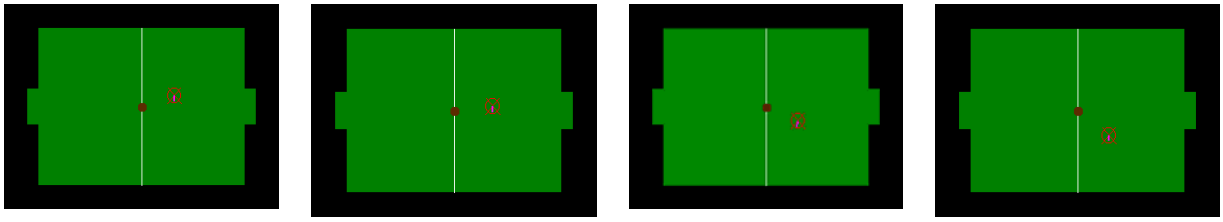
To test the reliability of the velocity estimates of robots in the world model two experiments were run. Both experiments involve a standing robot viewing a moving robot walking across the field from left to right, except at different speeds. Visual results are also shown that detail the visual detection results along with the corresponding world model state of the same frame, where the velocity vector for the robot model is represented by a line attached to the robot location.

3.2.3.1 Experiment 1

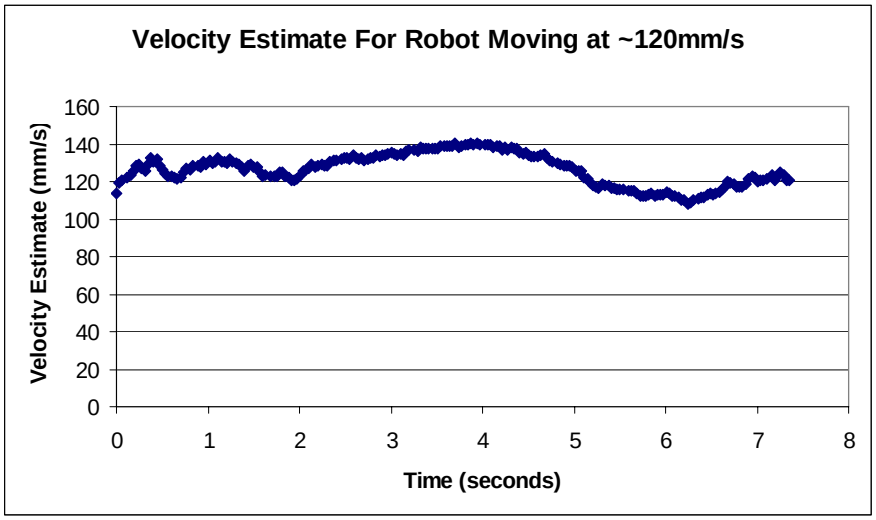
In this experiment a standing robot was placed on the field and was running a behavior to track detected robots with its head by attempting to position the closest detected robot in the center of its camera's field of view. A moving robot was placed on the field directly in front of the standing robot and was oriented such that when walking forward it progressed from left to right in front of the standing robot. The moving robot was made to walk at a constant speed of 120 mm/s in the forward direction. The calculated robot velocity estimates from the standing robot were recorded and the results are presented in Figure 3.2.



(a)



(b)

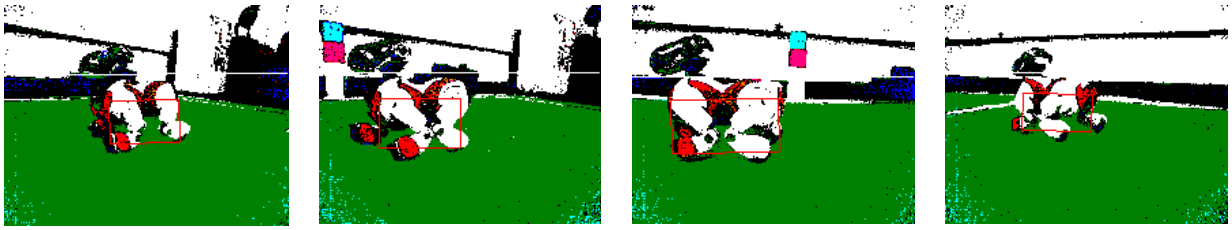


(c)

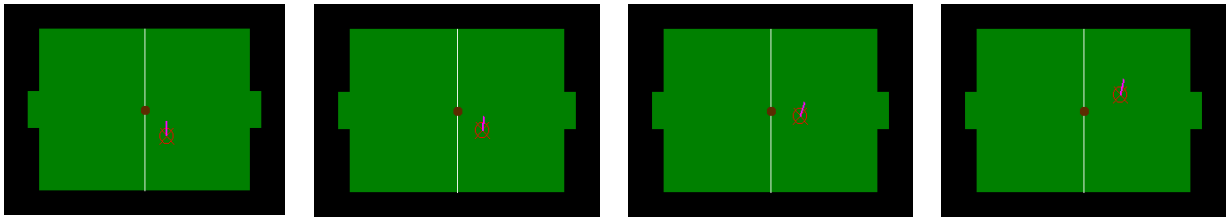
Figure 3.2: (a) Sample robot detection results during robot crossing
(b) Robot model positions/velocities for frames shown in (a)
(c) Recorded velocity estimates from standing robot during time of crossing

3.2.3.2 Experiment 2

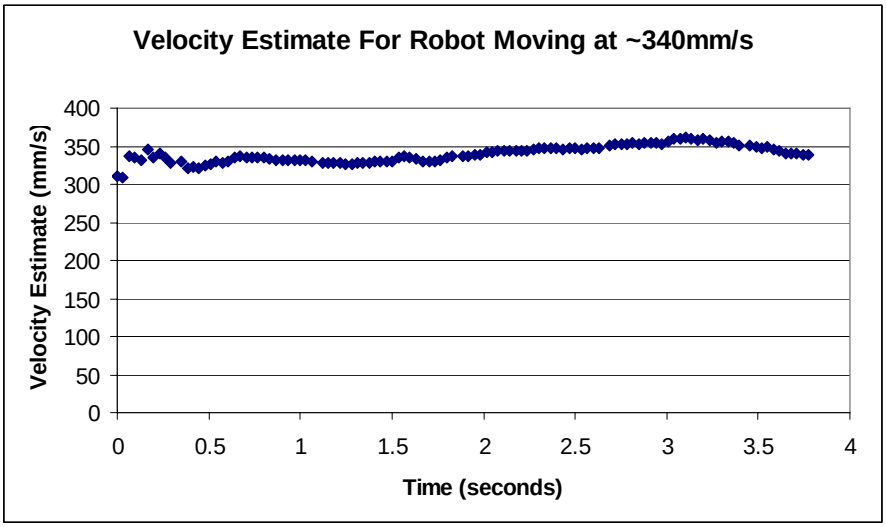
This experiment contained the same setup as in the previous experiment, with the only difference being that the moving robot was made to walk at a constant speed of 340 mm/s in the forward direction, and was walking from right to left. Velocity estimation results are shown in Figure 3.3.



(a)



(b)

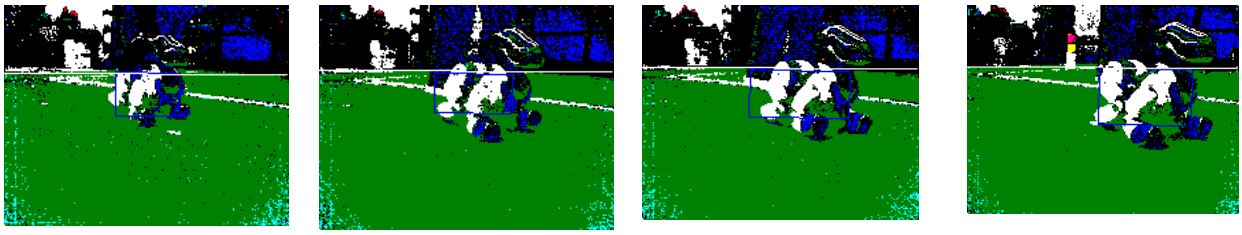


(c)

Figure 3.3: (a) Sample robot detection results during robot crossing
(b) Robot model positions/velocities for frames shown in (a)
(c) Recorded velocity estimates from standing robot during time of crossing

3.2.3.3 Visual Results

Figures 3.4 and 3.5 both show visual results of the velocity estimation system. The velocity estimate is represented by a vector, which is the visual representation of the direction and magnitude (speed) of motion of the robot to which it is attached. The visual robot detection results are also provided for comparison/evaluation. Both examples illustrate how well the velocity estimation fits with the actual path of motion of the robot. Figure 3.5 presents an interesting case where a moving robot is tracking another moving robot that is walking directly in front of it at the same speed and direction. The velocity estimate correctly models the movement of the robot being observed, even though all the vision frames are relatively the same, with the robot placed in the center, and don't provide visual cues for the movement of the robot.



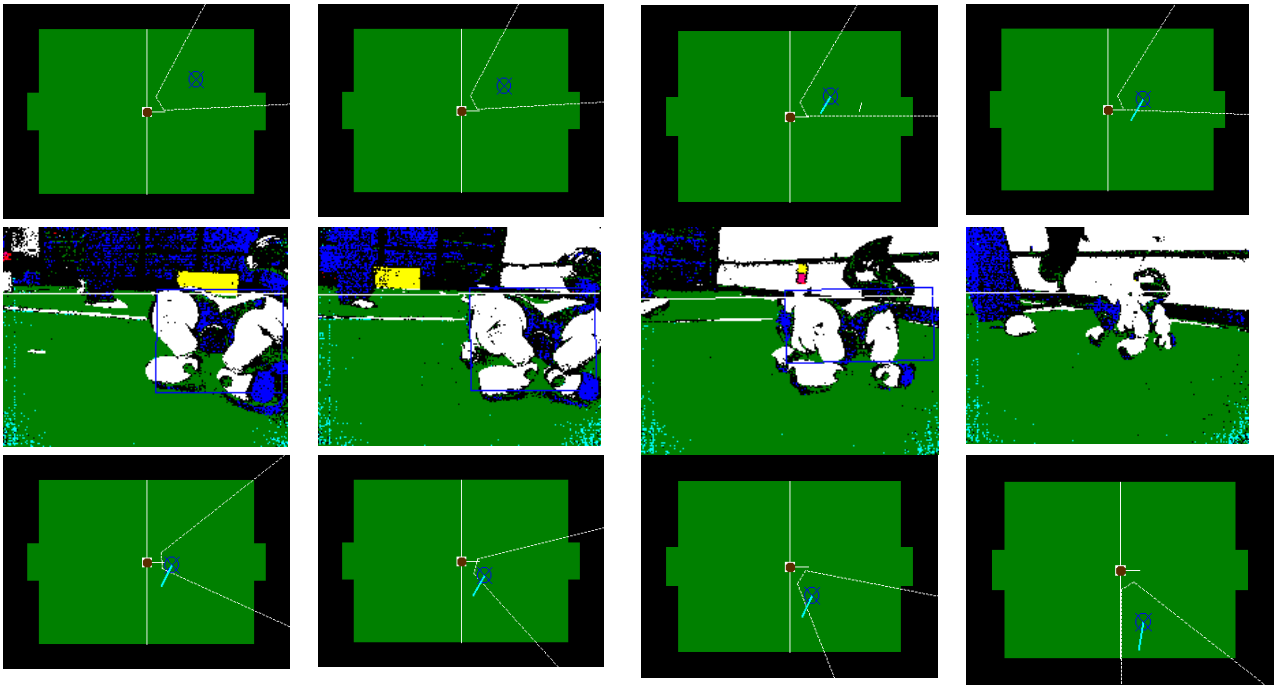
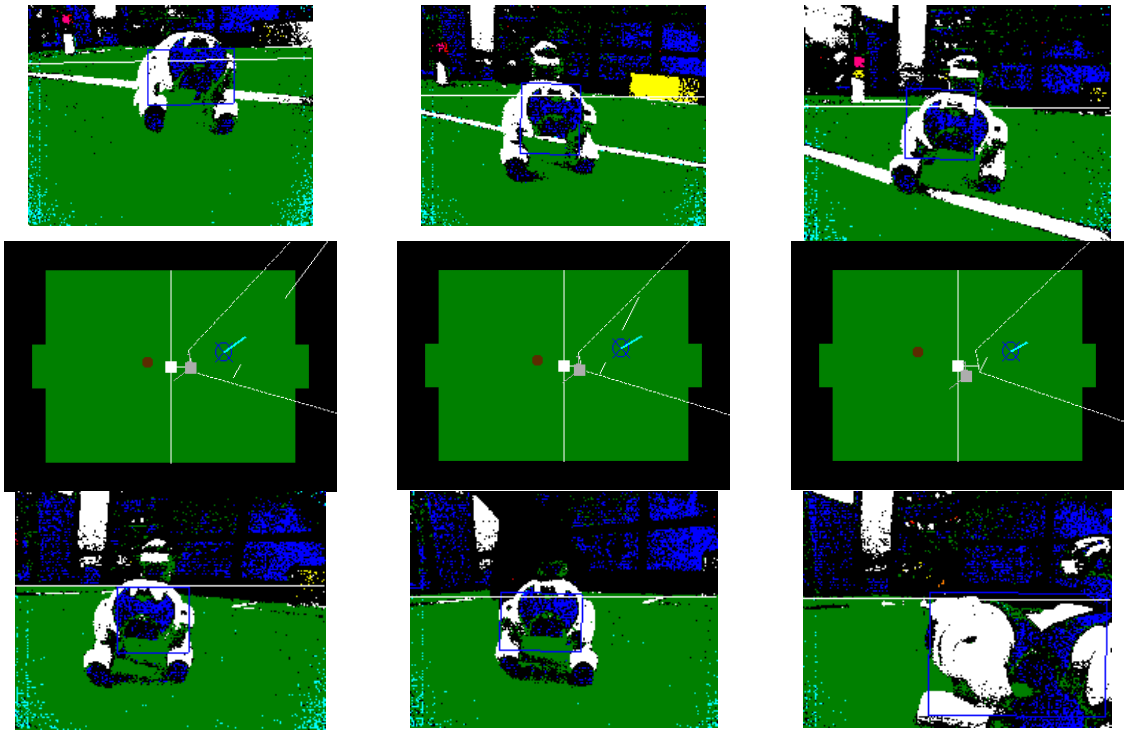


Figure 3.4: Visual robot detection/modeling/velocity results from standing robot viewing moving robot walking in nearly linear fashion from left to right



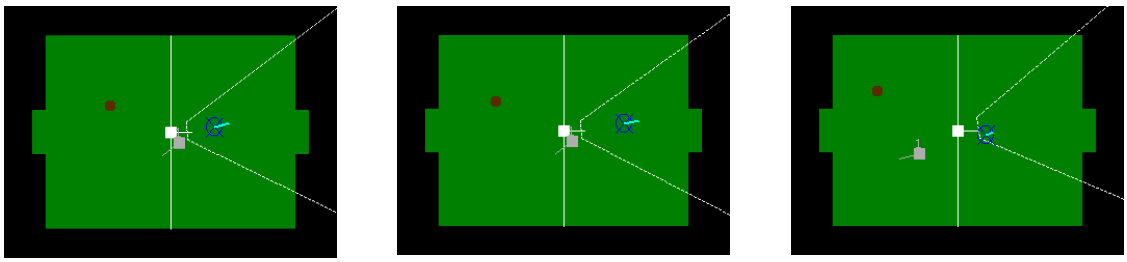


Figure 3.5: Visual robot detection/modeling/velocity estimate results from moving robot viewing another moving robot in front of it

Chapter 4

Game Behaviors

The introduction of visual detection and modeling of teammate and opponent robots allows for the creation of a variety of different game behaviors that without such information were either impossible or very hard to implement. Behaviors are able to gain more information about the current state of the game, which allows for better situational assessment and strategy building. Teammate robots can actually react to the presence of opponent robots and reason about the appropriate actions to take based on the situation, instead of blindly reacting only to the ball. For example, consider the situation when a robot walks towards the ball and it suddenly disappears from view when another robot walks in front of it. If the robot has no knowledge on the location of other robots on the field, it would most likely decide to start spinning in place to look for the ball since it has no idea where it could be and all directions of view are equally likely to contain the ball. However, if the robot had information about the locations of robots on the field, it could reason that the ball is being occluded by another robot and that is why it cannot be seen. The robot could then decide to try and move around the robot in question and search for the ball there. Many more behaviors are possible with the introduction of robot modeling, some of which have successfully been implemented and are presented in this chapter.

4.1 Navigation with Obstacle Avoidance

One of the most helpful behaviors as far as improving game play has to be the ability to navigate towards a desired target, whether it be an object like the ball or a global point on the field, while avoiding obstacles. Obstacles on the field are other robots, so having robot locations within the world model and being able to easily access that information greatly facilitates the creation of such a behavior. Not only is it desirable to avoid robots when traveling towards a desired location to prevent running into them, which wastes valuable time and makes reaching the target much more difficult, but also to avoid penalties called against pushing robots. If a robot on one team runs into a robot on the opposing team, intentionally or unintentionally, and continues to do so for three seconds it will get pulled from the field and undergo a removal penalty of thirty seconds before being allowed to re-enter the game. Penalties such as this greatly affect the penalized team as they are forced to play with fewer players, which improves the chances of scoring of the opposing team. Creating an effective obstacle avoidance navigation

behavior will help to minimize these penalties, in addition to helping the robots get to where they want to go.

4.1.1 Navigation Behavior

The navigation behavior that was developed consists of two states and has the following state machine, beginning with the initial state:

- *Go To Point*: The robot is directed to walk in a straight-line path directly towards the desired target. If the robot has been in this state for at least 0.3 seconds, is roughly facing the target and there is a robot blocking the path in front, pick the most appropriate side to move to, left or right, depending on whether or not it's open and its proximity to the target, and transition to *Avoid*.
- *Avoid*: The robot is directed to move to the side chosen in the previous state while maintaining a minimum distance away from the robot in front. Therefore, the robot need not move directly sideways, but can add a forward walk component as long as the distance to the other robot remains above or equal to the minimum allowable. If the robot has been in this state for at least 0.5 seconds and there is no robot blocking the path in front, or the robot has been in this state for more than 2 seconds, transition to *Go To Point*.

The time constants are to avoid oscillations in the behavior and in the case of the *Avoid* state, also to encourage continuing progress towards the desired target. The behavior is very simple, thanks to the information provided about robot locations in the world model, yet it is very effective during actual game situations. Figure 4.1 provides an example of a situation where the navigation algorithm is used. The red robot detects the presence of the blue robot in its path towards the ball and decides to avoid it by moving towards the left side, as it is open and closest to the target.

(a)

(b)

Figure 4.1: (a) Photo of the obstacle avoidance situation with the path chosen by the algorithm shown in light green.
(b) Segmented image taken during situation with robot detector result shown

4.1.2 Results

The navigation behavior has been successfully integrated into the CMDash team game play, and has helped to improve the team performance during actual competition. Figure 4.2 provides video capture images taken during a game that show the execution of the navigation behavior and how it allowed our robot to gain control of the ball even though two separate robots from the opposing team were in its way initially.

Figure 4.2: Video capture images of blue robot performing navigation towards ball with obstacle avoidance during a game

4.2 Dodging Opponents when Shooting

The behavior to avoid, or dodge, opponent robots when attempting to shoot on goal or down the field is a similar yet slightly different behavior from that of navigating while avoiding obstacles. The dodge behavior only executes after the robot has gained control of the ball beneath its chin and wants to shoot, or kick the ball, towards a specific location. Once the robot grabs the ball, the Turn behavior executes, which attempts to change the orientation of the robot so that it is facing the desired target. Once the Turn has reached the desired orientation, it checks for the existence of opponent robots in front of the robot by accessing all robot locations from the world model, and seeing if any of them fall within the region in front of the robot. If so, the Turn behavior then calls the Dodge behavior, whose job it is to try and avoid the opponent robot before shooting towards the target, so as to get a clear shot off.

4.2.1 Dodging Behavior

The dodging behavior has three states:

- *Initial*: The direction of motion is chosen, left or right, in order to avoid the opponent robot in front. The dodge direction is chosen simply based on which side is closer to the inside of the field, because dodging towards the outside of the field and accidentally leaving the field causes loss of possession of the ball. Transition to *Dodge*.
- *Dodge*: The robot is directed to move to the side chosen in the previous state at full speed, with no forward walking component. If the robot has been in this state for at least 0.3 seconds and there are no longer robots in front, then transition to *Kick*. If the robot has been holding the ball for approximately 3 seconds, then abort the behavior to avoid penalty.
- *Kick*: Kick the ball forwards towards the target. The behavior is then reset, and ready to be called again by the Turn behavior if necessary.

4.2.2 Results

The dodging behavior, like the navigation behavior, has been successfully integrated into the CMDash team game play, and has helped tremendously in increasing the chances of our team scoring after gaining control of the ball while near the opponent goal area. The dodging behavior is also very effective in avoiding opponent robots when trying to clear the ball down the field. Figure 4.3 shows video capture images of the dodge behavior being executed during competition. The red robot after grabbing the ball detects the presence of the opponent goalie robot, then quickly side steps to its right side until the goalie is no longer impeding the shot, then kicks forward and scores a goal.

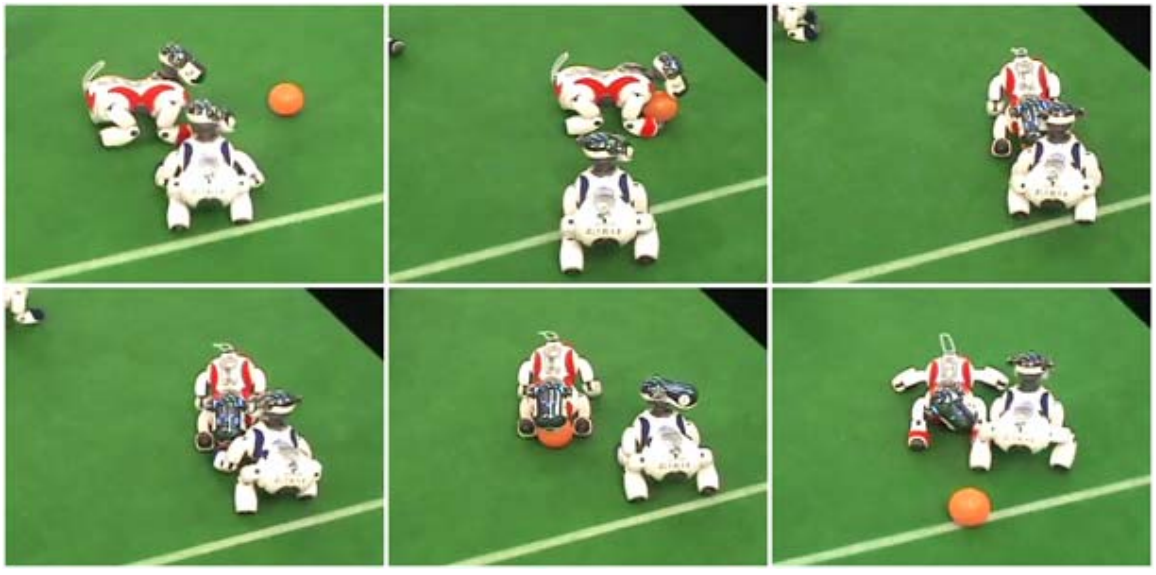


Figure 4.3: Video capture images of red robot dodging opponent blue robot before shooting and scoring goal

4.3 Follow Target Robot

One of the first behaviors created after the introduction of robot detection and modeling was the behavior to follow a detected robot wherever it went. The behavior was initially created as a way to test the effectiveness of the robot detection algorithm, but has real potential to be used during actual game play. For example, if the strategy were adopted to perform man-on-man, or rather robot-on-robot coverage of the opponent team, this behavior would be ideal. The behavior would have the robot follow an opponent robot around wherever it went, and if it acquires the ball, then the following robot is right there to try and steal it or impede the opponent from getting a clear shot on goal. The following robot target behavior can also be useful for following a teammate robot in setup for some sort of designed play. If the play calls for close-quarter interactions then using visual information instead of wireless communication of positions is the natural choice to avoid issues with errors in localization estimates. Figure 4.4 shows images taken from the view of a following robot running the behavior. Also, the follow behavior was used to produce the results shown in Figure 15 for velocity tracking.

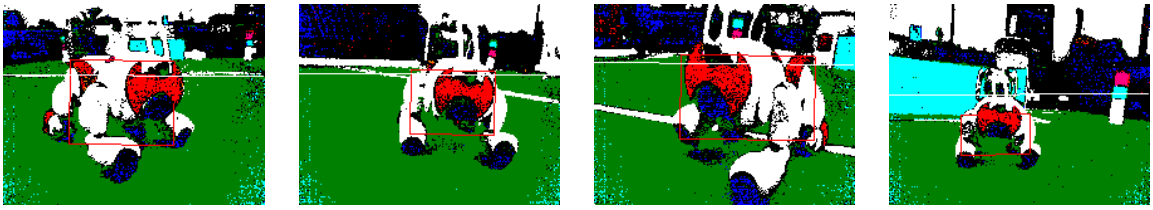


Figure 4.4: Segmented images taken during following of red robot

4.4 Defender Aware of Attacking Robot

A behavior that is in its initial stages of development involves having the defender robot, when deciding to clear a ball that has entered its zone, first look to see if an attacking robot from the opposing team is at the ball and if so, defend first before trying to retrieve the ball. Many situations have arisen during competition where our defender goes out to try and clear the ball from the defensive zone, yet before it gets to grab the ball, an attacking opponent robot pushes the ball such that the defender loses sight of it and ends up at a disadvantage since the ball has gone behind it with the attacking robot in prime position to re-acquire the ball and shoot towards our goal. This behavior attempts to avoid these painful situations by waiting for the opponent robot to make a move first, so the defending robot always knows where the ball is and can more effectively defend against a shot, in addition to waiting for an appropriate time to try and grab the ball to clear it. The behavior is almost fully developed and can, as of now, have the defender go for the ball it needs to clear and clear it if no opponent robots are around. If there is an opponent robot at the ball, the defender can effectively detect this situation and begins to protect its goal against a shot by following the attacking robot's every move. If the attacker tries to dodge the defender, the defender will detect this and move along with it so it can't get a clear shot on goal. The only thing remaining to be implemented is the decision of when to try and grab the ball again for clearing. As of now the behavior

looks very promising and should help our defender more effectively assess dangerous situations during game play and react appropriately to diffuse them.

4.5 Future Work

There are various more behaviors that can be created to take advantage of the information provided by the robot detection and modeling procedures, all of which can add a new dimension to the functionality, performance, and teamwork of our robot soccer team. Some of the behaviors which are particularly interesting include: passing to visual teammates, using robot vision for teammate coordination when approaching the ball or during a designed play, communicating information about moving opponents to teammates that are beyond the visible range of detection, and positioning around the field so as to remain in open regions away from opponent robots.

Chapter 5

Conclusion

The visual detection of objects and modeling their locations and motion in a dynamic environment remains a challenging problem in robotics research. This thesis has three contributions to robotics research. We introduced an algorithm for the visual detection of robots on the soccer playing field, which are dynamic objects whose appearance changes depending on its orientation. We presented a method for the modeling of position and motion of the detected robots, whose techniques can be easily applied to many different research projects across a variety of domains. Finally, we described ways of using the information introduced by the detection and modeling of robots to improve the performance of the robot soccer team by implementing applicable behaviors such as navigating to a desired target while avoiding obstacles.

Empirical results are provided for both methods presented in this thesis for appropriate evaluation of our research. The detection and modeling methods have also proven effective during actual competition. CMDash'06 competed at the RoboCup US Open 2006 at the Georgia Institute of Technology, where we came in first place among all the US participants, thanks in large part to the introduction of robot detection and modeling and its use by behaviors such as dodging opponents when shooting towards the goal. This demonstrates that the techniques introduced in this thesis have a significant impact on robotics research in adversarial, multi-robot domains.