ROBOCUP 2007 GOAL AND BEACON RECOGNITION
USING SONY AIBO® ERS-7M2 ROBOT DOGS

A Project
Presented
To the Faculty of
California State University, Chico

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Paulene M. Purdy 2007
Fall 2007
# TABLE OF CONTENTS

| List of Tables | iv |
| List of Figures | v |
| Abstract | vii |

## CHAPTER

### I. Introduction

- Purpose ........................................... 1
- Scope .................................................. 1
  - Soccer Field Goals ........................... 4
  - Beacons ........................................... 6
  - Hardware ........................................... 6
- Significance ...................................... 8
- Limitations ........................................ 8
- Definition of Terms ................................ 9

### II. Literature Review

- German Team ....................................... 10
  - Scanning ......................................... 10
  - Detecting Beacons ............................. 11
  - Detecting Goals ................................. 13
- University of Pennsylvania ..................... 13
  - Color Segmentation ............................ 13
- University of New South Wales .................. 14
  - Sampling ......................................... 14
  - Feature Detection ............................. 16
  - Object Recognition ............................ 16
  - Goal Recognition .............................. 16
  - Beacon Recognition ........................... 17
- Commonalities in Vision Approaches .......... 18
<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>III. Methodology</td>
<td>20</td>
</tr>
<tr>
<td>rUNSWift Code Base Chosen</td>
<td>20</td>
</tr>
<tr>
<td>rUNSWift’s Use of OPEN-R®</td>
<td>20</td>
</tr>
<tr>
<td>Navigating the rUNSWift Code Base</td>
<td>21</td>
</tr>
<tr>
<td>Creating the Color Table</td>
<td>24</td>
</tr>
<tr>
<td>Changes for the New Soccer Goals</td>
<td>27</td>
</tr>
<tr>
<td>Changes for the New Beacons</td>
<td>29</td>
</tr>
<tr>
<td>Changes to Debugging Tools</td>
<td>33</td>
</tr>
<tr>
<td>Version Control</td>
<td>34</td>
</tr>
<tr>
<td>IV. Findings and Results</td>
<td>38</td>
</tr>
<tr>
<td>SubVision Tool Results</td>
<td>38</td>
</tr>
<tr>
<td>BlueYellow Beacon</td>
<td>38</td>
</tr>
<tr>
<td>YellowBlue Beacon</td>
<td>38</td>
</tr>
<tr>
<td>Blue Goal</td>
<td>40</td>
</tr>
<tr>
<td>Yellow Goal</td>
<td>41</td>
</tr>
<tr>
<td>Notes on Testing Using SubVision Tool</td>
<td>41</td>
</tr>
<tr>
<td>Playing Soccer</td>
<td>42</td>
</tr>
<tr>
<td>Ready State</td>
<td>42</td>
</tr>
<tr>
<td>Set State</td>
<td>44</td>
</tr>
<tr>
<td>Playing State</td>
<td>45</td>
</tr>
<tr>
<td>V. Conclusions and Recommendations</td>
<td>54</td>
</tr>
<tr>
<td>Summary</td>
<td>54</td>
</tr>
<tr>
<td>Conclusions</td>
<td>55</td>
</tr>
<tr>
<td>Recommendations</td>
<td>56</td>
</tr>
<tr>
<td>References</td>
<td>58</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
</tr>
<tr>
<td>A. File Changes in Version Control</td>
<td>60</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Results of SubVision Testing</td>
<td>44</td>
</tr>
<tr>
<td>FIGURE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>1. Scale diagram of entire 2005 soccer field</td>
<td>2</td>
</tr>
<tr>
<td>2. Scale diagram of entire 2007 soccer field</td>
<td>3</td>
</tr>
<tr>
<td>3. Appearance of the yellow goal</td>
<td>4</td>
</tr>
<tr>
<td>4. Dimensions of the 2007 goals</td>
<td>5</td>
</tr>
<tr>
<td>5. Top and Side views of the 2007 beacons</td>
<td>7</td>
</tr>
<tr>
<td>6. GermanTeam scan lines and grids</td>
<td>12</td>
</tr>
<tr>
<td>7. UNSW scan lines</td>
<td>17</td>
</tr>
<tr>
<td>8. Architecture of rUNSWift 2005 and its interactions</td>
<td>21</td>
</tr>
<tr>
<td>9. List of top level directory</td>
<td>22</td>
</tr>
<tr>
<td>10. Listing of base directory</td>
<td>23</td>
</tr>
<tr>
<td>11. Listing of robot directory</td>
<td>24</td>
</tr>
<tr>
<td>12. The rc tool with no colors configured</td>
<td>26</td>
</tr>
<tr>
<td>13. The rc tool after colors configured</td>
<td>27</td>
</tr>
<tr>
<td>14. Vision module class diagram</td>
<td>30</td>
</tr>
<tr>
<td>15. Goal Visual Features in SubVision</td>
<td>31</td>
</tr>
<tr>
<td>16. BlueYellow Beacon in SubVision Objects Tab</td>
<td>35</td>
</tr>
<tr>
<td>17. BlueYellow Beacon in SubVision</td>
<td>36</td>
</tr>
<tr>
<td>18. SubVision version control structure</td>
<td>37</td>
</tr>
<tr>
<td>FIGURE</td>
<td>PAGE</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>19. Blue dog facing BlueYellow beacon</td>
<td>39</td>
</tr>
<tr>
<td>20. Blue dog facing YellowBlue beacon</td>
<td>39</td>
</tr>
<tr>
<td>21. YellowBlue beacon in SubVision</td>
<td>40</td>
</tr>
<tr>
<td>22. Blue dog facing blue goal</td>
<td>40</td>
</tr>
<tr>
<td>23. Blue goal in SubVision</td>
<td>41</td>
</tr>
<tr>
<td>24. Blue dog facing yellow goal</td>
<td>42</td>
</tr>
<tr>
<td>25. Yellow goal in SubVision</td>
<td>43</td>
</tr>
<tr>
<td>26. Robot states</td>
<td>45</td>
</tr>
<tr>
<td>27. Approximate kickoff positions</td>
<td>46</td>
</tr>
<tr>
<td>28. Dog positions on blue kickoff</td>
<td>47</td>
</tr>
<tr>
<td>29. Dogs playing just after kickoff</td>
<td>48</td>
</tr>
<tr>
<td>30. Blue dog kicking ball toward yellow goal</td>
<td>49</td>
</tr>
<tr>
<td>31. Blue dog kicking ball further toward yellow goal</td>
<td>50</td>
</tr>
<tr>
<td>32. Blue dog ready to kick ball into yellow goal</td>
<td>51</td>
</tr>
<tr>
<td>33. Blue dog kicking ball into yellow goal</td>
<td>52</td>
</tr>
<tr>
<td>34. Blue dog scoring</td>
<td>53</td>
</tr>
</tbody>
</table>
ABSTRACT

ROBOCUP 2007 GOAL AND BEACON RECOGNITION
USING SONY AIBO® ERS-7M2 ROBOT DOGS

by
Paulene M. Purdy 2007
Master of Science in Computer Science
California State University, Chico
Fall 2007

The Intelligent Systems Lab (ISL) at California State University, Chico owns Sony AIBO® ERS-7M2 robot dogs for use in RoboCup Four-Legged League competitions. ISL’s intent was to compete in the 2007 competition. In order to do so, ISL needed to research other RoboCup teams’ code bases to determine which would be the best with which to start. Several were researched, but eventually the code base chosen was written by the University of New South Wales (UNSW) for the 2005 RoboCup tournament. The UNSW code was modified to satisfy the 2007 RoboCup Soccer Rules which required new beacons (which are used as landmarks on the sides of the soccer field) and soccer goals. Modifications to the vision module were made to accommodate these environment changes. These changes included soccer goal structure and dimensions. Estimated locations for the beacons were also changed to be
midfield on each side of the soccer field. Since the new beacons are only two in number instead of four, many file changes had to be made to expect a fewer number of beacons. The 2007 beacons are different colors as well. No pink stripe exists on the beacons in the 2007 field specifications, so the code was changed to search for runs of blue pixels instead of pink. Care was taken so that this change did not affect the recognition of the blue goal.

The biggest challenge in this project was finding the time to work on the technical aspects. The majority of time was spent on setting up and tearing down the physical environment (field) needed to work with and test the code running on the robot dogs. Setup alone often consisted of moving desks, sweeping prior to putting the green field carpet down, vacuuming the carpet then, actually, putting up the walls, beacons, and goals.

Changes made to the code were tested in a couple of ways. Beacon and goal recognition was tested through a tool called SubVision which visually displays a goal or beacon (in appropriate colors) when recognized by the dog. The robot dog was placed at various angles facing a goal or beacon to ensure that it would still be recognized. The code changes were also tested by scrimmaging two teams of robot dogs—allowing us to confirm that the dogs could localize using the 2007 goals and beacons and navigate the field to score goals.
CHAPTER I

INTRODUCTION

Purpose

The Intelligent Systems Lab at the California State University, Chico intended to compete in the 2007 RoboCup Four-Legged League competition. New teams such as ISL at CSU, Chico are allowed to start working from code bases published by other universities that have competed in past years. So, ISL needed to research other teams’ code bases in order to determine which would be the best from which to start; then work with the chosen code to satisfy new requirements. ISL chose the 2005 University of New South Wales RoboCup Four-Legged League code [1] (also known as the rUNSWift code). The rUNSWift code (or something very similar) was used by the University of New South Wales in the RoboCup 2005 Four-Legged League competition [2]. The 2005 code base was changed to satisfy the new 2007 Four-Legged League Soccer Rules [3]. The 2007 rules specify different soccer goals and beacons. The next section gives more detail of the changes in these areas.

Scope

The scope of this project includes new requirements specified by the RoboCup 2007 Four-Legged League Soccer Rules [3]. The 2005 University of New South Wales code follows the RoboCup 2005 Four-Legged League Soccer
Rules [4]. The 2005 (and 2006) rules specify the soccer field shown in Fig. 1. The 2005 rules describe the field in the following manner [4, p. 1-2]:


**Lines**
All the lines on the soccer field (the halfway line, the lines surrounding the penalty areas, the goal lines, and the center line) are drawn with white stripes of 25 mm in width. The circle on the midfield line has a diameter of 360 mm from the middle of the white stripe on one side to the middle of the white stripe on the other side.

**Field Colors**
All items on the RoboCup field are color-coded:
- The field (carpet) itself is green.
- The lines on the field are white.
- The red team defends the yellow goal.
- The blue team defends the sky-blue goal.
The 2007 soccer field is defined in Fig. 2. The dimensions of the field are identical from 2005 through 2007 fields. The difference between the two diagrams appears to be twofold:

1. The 2005 soccer field goals have a solid back, whereas the 2007 goals have an open back.
2. The 2005 field had four beacons while the 2007 field has only two. The coloring of the beacons has changed as well.

Soccer Field Goals

The 2005 soccer field goals have a solid back. They are 300mm tall, 812mm wide. On the other hand, the 2007 soccer field goals look more like real soccer goals (see Fig. 3).

Fig. 3. Appearance of the yellow goal. Notice that the external surface of the rear portion of the goal is white. Reprinted with permission from RoboCup Technical Committee, “RoboCup Four-Legged League Rule Book.” Retrieved 1 February 2007 from the World Wide Web: http://www.tzi.de/4legged/pub/Website/Downloads/Rules2007.pdf.

The new goals no longer have a solid back that is the same height as the entire goal. The back of the new goals are only 100 mm high vs. the old full height of 300 mm. However, the front of the goal is now 350 mm high including the crossbar and its total width is 1000 mm including the two side bars. Note that, “The posts, top cross bar and the inner surface of the rear portion of the goal are
either yellow or sky-blue. The external surface of the rear portion of the goal is white” [3, p. 3]. The dimensions are shown in Fig. 4.

Fig. 4. Dimensions of the 2007 goals (without crossbar) viewed from a) above and from b) the side. Reprinted with permission from RoboCup Technical Committee, “RoboCup Four-Legged League Rule Book,” 1 February 2007; http://www.tzi.de/4legged/pub/Website/Downloads/Rules2007.pdf.
Beacons

The 2005 field has cylindrical beacons which are placed on the edge of the field at 1/4 and 3/4 of the length of the field and 1350mm from the center line (see Fig. 1). When looking from the yellow goal toward the sky-blue goal (see Fig. 5), the lower stripe (part B) of the two beacons on the right shares the color with the neighboring goal and the top stripe is pink. On the left side of the field, the lower stripe of the beacons is pink and the upper stripe (part A) is the goal color. Part C is always white.

On the 2007 field the cylindrical beacons (see Fig. 2) are placed on the edge of the field on the half-way line. Part C is always white. When looking from the yellow goal toward the sky-blue goal, part B of the beacon on the right is sky-blue and part A is yellow. The beacon on the left is the opposite, that is part B is yellow and part A is sky-blue. [3, p. 3]

Hardware

The 2005 through 2007 Four-Legged League Soccer Rules specify models of Sony AIBO® robot dogs allowable in competition. They specify both the ERS-210 and ERS-7 (including ERS-7M2 and ERS-7M3) models. ISL owns two teams of Sony AIBO® ERS-7M2 robot dogs which were used for testing. The ERS-7M2 is approximately 317 mm long, 180 mm wide, and 278 mm in high. The ERS-7M2 has numerous input devices including a stereo microphone, head and back sensors, paw and chin sensors, distance sensor, acceleration sensor, vibration sensor and a color camera. This project focused on the images from the color camera as input to the goal and beacon recognition algorithms.
Significance

The goal of the international RoboCup soccer initiative is to develop a team of humanoid robots that can compete with the official human World Soccer Champion team and win [5]. Currently teams are using Sony AIBO® robotic dogs to develop code, concepts, strategies, and technologies that may be transferable to humanoid robots.

The Intelligent Systems Lab at CSU, Chico also has a goal to facilitate the development of cross-disciplinary courses and provide exciting collaborative research possibilities. The ISL enables both students and faculty to investigate, design, and implement control algorithms using non-traditional techniques…. [6]

The vision and localization modules of the rUNSWift 2005 code base [1] were modified in order to recognize the new soccer field goals and beacons for the 2007 RoboCup Four-Legged League. This work furthers the kind of work promoted by the RoboCup Soccer initiative by making improvements in vision and localization in a way that can be used for humanoid robots as well. Aside from making technically versatile improvements, this work furthers the goal of ISL to provide collaborative and cross-disciplinary research projects. Work was done within a team environment that often provided communication with Computer Science, Mechatronics, and Electrical Engineering majors.

Limitations

Limitations to this project included time, space, and team size. Only the spring semester of 2007 was used to develop solutions for the new soccer goals
and beacons as well as troubleshoot problems as they arose. Space for the field setup was an issue for most of the semester. We often had to move desks and chairs in a classroom after class hours to set up the field and tear it down again. Space was eventually found in another building; however, we had to set up and tear down the field in that room as well. Our team consisted of only three members (one graduate student and two undergraduate students) working part-time. Many other RoboCup teams have more extensive teams. Carnegie Mellon, for instance, has eleven Ph.D. students and eight undergraduates [7]. The German Team actually consists of groups from several universities—the Humboldt-Universität zu Berlin, the Universität Bremen, the Technische Universität Darmstadt, and DFKI Lab Bremen [9].

Definition of Terms

• UPenn – University of Pennsylvania

• UNSW – University of New South Wales

• rUNSWift – UNSW RoboCup team and/or code

• CSU, Chico – California State University, Chico

• ISL – Intelligent Systems Lab at California State University, Chico
CHAPTER II

REVIEW OF LITERATURE

Through the course of this project, multiple RoboCup Four-Legged code bases were reviewed including those written by the GermanTeam [8], UPenn [11], Carnegie Mellon [7], and UNSW [1]. These teams, as well as others, have publicly released their code so that new teams can use it as a base and “get up to speed” more quickly.

GermanTeam

The GermanTeam is a national robotic soccer team participating in the international RoboCup competitions in the Four-Legged League. The GermanTeam consists of a group of cooperating German universities—Humboldt-Universitat zu Berlin, Universitat Bremen, Technische Universitat Darmstadt, and DFKI Lab Bremen [9]. This section provides a summary of the essentials of the GermanTeam 2005 vision module.

Scanning

Prior to the start of scanning a horizon is calculated. Objects at eye level lie on the horizon. The horizon line is therefore the “line of intersection between the projection plane $P$ and a plane $H$ parallel to the ground at height of the camera” [10, p. 23]. The rotation of the camera (not the position of the camera or the camera’s height) affects the position of the horizon. After the horizon is
calculated a grid is constructed based on this line. Grid lines are both parallel and perpendicular to the horizon. Grid lines are greater in number near the horizon where a greater number of objects of interest are expected and fewer further from the horizon. Each of these grid lines are processed from top to bottom and left to right pixel by pixel. Each pixel is classified by color and an object of interest is indicated by a predetermined pattern of these classified colors.

Two separate grids are used in order to detect objects more quickly and to decrease the number of false positives. The main grid is constructed around and below the horizon to detect objects on the field (ball, field lines, robots, obstacles, and goals). A second grid (or in this case a set of grid lines parallel to the horizon) is constructed around and above the horizon to detect beacons. These two grids provide a context in which to recognize objects (see Fig. 6).

Detecting Beacons

The GermanTeam uses a beacon detector module to detect as well as analyze beacons [10]. The beacon detector module generates scan lines parallel to and above the horizon. The distance between the scan lines increases as the distance between the scan line and the horizon increases. These scan lines are used to search for pink pixels. Runs of pink found in this manner are clustered together according to proximity (both vertical and horizontal) and further “merged” into a single line segment which represents the middle of the pink
portions of the beacon. Three or four more scan lines are created perpendicular to the horizon to find the second color of the beacon (blue or yellow). A search is depending on the number of edge points found and certain sanity checks are satisfied such as the ratio between the horizontal and vertical length. If this likelihood value exceeds a determined threshold the object is considered a beacon [10].

The center of the pink and the flag type (YellowPink, PinkYellow, BluePink, or BlueYellow) is passed to a flag specialist module. This piece of code scans vertically and horizontally from the center of the pink area to find the borders of the flag to form an approximation of the beacon size. This width approximation is refined by additional horizontal scans. The height of the beacon is determined with additional vertical scans. In determining the leftmost, right-
most, topmost, and lowest points, the flag specialist is searching for the last pixel having one of the colors of the given current flag type. Gaps of no color are accepted [10].

Detecting Goals

The upper half of the horizon-aligned grid is scanned for goal-colored pixels. Vertical grid lines are scanned from left to right, so the left side of the goal is discovered first. From there the edge is followed upward and downward by scanning vertically and horizontally. At the top endpoint, the scan follows the “crossbar.” Once the right edge is found the right side of the goal is processed similarly to the left [10].

During the process of searching for a goal, blue or yellow line segments near beacons are ignored. The discovered object is checked to ensure there are green-colored pixels beneath each end before being accepted as a goal.

University of Pennsylvania

Color Segmentation

The University of Pennsylvania RoboCup code “segments the highest-resolution color images from the camera” [11, p. 4]. Individual pixels in the image are classified using their YCbCr values within the color segmentation method. Using a Gaussian mixture model (obtained through a number of training images), the YCbCr color cube is segmented into a number of known colors:
• Orange (Ball)
• Pink (Marker)
• Cyan (Marker and Goal)
• Yellow (Marker and Goal)
• Blue (Robot)
• Red (Robot)
• Green (Field)
• White (Lines and Border)

Classified pixels are merged into connected components by run-length encoding the images then merging those run-lengths into connected regions. Resulting regions are classified into known objects by various attributes including the bounding box, the centroid location, and the major and minor axes lengths [11].

University of New South Wales

The University of New South Wales (UNSW) used a technique called “sub-sampling” in their 2005 vision module. Sub-sampling involves sampling pixels in high-interest areas of an image in order to detect features and then use these features to recognize objects in the environment [12].

Sampling

Scan lines are “drawn” across or mapped onto the image both vertically and horizontally in order to sample the image. An artificial horizon is placed on the image at the height of the camera assuming the camera is upright and
looking straight ahead. Vertical lines are drawn down from the horizon in the following manner [12]:

- The first scan line is drawn downward from the center of the horizon to the bottom of the image.

- At a fixed spacing of 16 pixels additional “full” scan lines are drawn parallel to the first scan line on either side of it. Additional scan lines are drawn outward until they reside outside of the image frame.

- Drawn parallel to these full scan lines halfway between them are half scan lines which thus reside 8 pixels from their neighboring full scan lines.

- Between each full and half scan line is drawn a quarter scan line which is a constant 48 pixels in length and 4 pixels from its neighboring full and half scan line.

Horizontal scan lines are drawn in the following manner [12]:

- Beginning 16 pixels below the horizon, scan lines are constructed parallel to the horizon. For each scan line drawn the gap is doubled until a scan line lies completely below the image frame.

- Beginning just below the horizon, scan lines are constructed parallel to the horizon with an exponentially increasing gap until a line lies above the image frame.

These scan lines are drawn with this information in mind [12]:

- Objects close to the robot are located below the horizon and appear larger than objects further away. So, when the ball is close to the dog, the ball will
appear below the horizon and larger than other objects in the image. Field lines, although appearing larger, are sparser in the lower part of the image. Whereas the field lines further away appear further up in the image, but are more closely spaced.

- More information resides in the upper part of the image or near the horizon.

- Goals and beacons will be seen above the horizon. Note that the latter set of horizontal lines is created due to this reason. Fig. 7 shows the scan lines described above.

**Feature Detection**

The scan lines described previously are used in feature detection. The scan lines are processed from bottom to top and left to right. Each pixel in a scan line is compared to a previous pixel. If the difference between the colors of the pixels (the difference in the Y, Cr, Cb values in the image) vary greatly an edge of an object is inferred [12].

“Visual Features” representing pixels are created by this feature detection. These visual features are clumped together with surrounding visual features on the same scan line to form one visual feature [12]. This visual feature is given a type according to its color (e.g. VF_OBS obacle, VF_PINK, VF_BLUE, VF_YELLOW, VF_FIELDLINE, etc.).
Object Recognition

The visual features detected in the previous section are used in object recognition. Remember, the visual features output from feature detection really
represent a portion of a line of some type/color. These features are then grouped together according to their location in relation to other features of the same type. More specifically, each feature is placed in its own group and a merge algorithm is used to merge groups of features from adjacent scan lines that overlap [12]. This group of features becomes an object that is then run through checks that determine which object it is within our known soccer field environment.

Goal Recognition

Each object or group of features that is of type `VF_BLUE` or `VF_YELLOW` is run through goal recognition code. This code determines the width and height of the object and checks that the object is sitting on the green soccer field (that it is not floating). If the object is not floating then it is input into some sanity-checking code that verifies that the aspect ratio of the object is close to that of the soccer goal. If these checks succeed the object information is stored in the appropriate goal (blue/yellow).

Beacon Recognition

Each object or group of features that is of type `VF_PINK` is run through beacon recognition code. This code searches for a minimum number of yellow or blue pixels (six) first below then above the pink object within a two-pixel radius of the pink object. If this code and sanity checks succeed, the corresponding beacon (pink-yellow/pink-blue/yellow-pink/blue-pink) has been found. Thereafter, appropriate variables (elevation, position, height, width etc.) are set on the beacon object.
Commonalities in Vision Approaches

All of the schools referenced above have many commonalities. *Segmentation* is an approach that attempts to cut down on the number of components processed using a variety of methods in order for later algorithms to handle them. “Each method attempts to obtain a compact representation of its data set using some form of model of similarity” [13, p. 302]. One view of segmentation is that it attempts to find a minimum number of components of a data set that naturally belong together. This process is referred to as *clustering*.

Clustering can be performed in two ways—*partitioning* and *grouping*. Partitioning involves carving up a large data set while using some idea of commonality. Grouping involves taking separate distinct elements and grouping them together according to some idea of commonality [13]. Some form of segmentation or clustering is used by each of the universities referred to previously. Both the GermanTeam and rUNSWift use artificial scan lines that “sample” an image. The GermanTeam then uses a type of *partitioning* according to color around the goals whereas it uses a type of *grouping* within the beacon detector. One the other hand, UNSW uses grouping in order to recognize goals and beacons. The University of Pennsylvania declares the usage of color segmentation where it classifies each of the pixels in an image according to color. Then the pixels are “merged” (or grouped) into connected components and regions accordingly.

These code bases have many similarities, but differ slightly. Each code base must be changed in order to recognize the new 2007 beacons and goals.
The next chapter describes how the UNSW code base was changed in order to accomplish this task.
CHAPTER III

METHODOLOGY

rUNSWift Code Base Chosen

Several code bases were compiled and assessed including Carnegie Mellon, UPenn, GermanTeam, and rUNSWift. The rUNSWift code base was eventually chosen due to its extensive use of indicators (lights, ears, and tail), simple scripts, and useful tools. It was also chosen because the code was easily modified due to a simple structure, and changes made were readily seen on the robotic dog in minutes. The rUNSWift team has placed well in the RoboCup soccer competition over the years. They took second place in 1999, 2002, and 2006; took first place in 2000, 2001, and 2003; were quarter finalists in 2004; and took third place in 2005 (the code base CSU, Chico is using).

rUNSWift’s Use of OPEN-R®

The OPEN-R® Software Development Kit (SDK) is an interface provided by Sony for the AIBO® entertainment robots. OPEN-R® is used by rUNSWift (as well as the other code bases mentioned) to both gather information from and control the Sony AIBO®. rUNSWift’s high level architecture is shown in Fig. 8 [19]. Fig. 8 illustrates that OPEN-R® is a layer between the rUNSWift code base and the robot. Images are delivered via OPEN-R® to the vision module which in turn calls the behavior module (as well as others). The behavior module
calls the actuator module which controls the movement of the Sony AIBO® robot via an OPEN-R® interface.

Navigating the rUNSWift Code Base

The rUNSWift code base contains the following high level directories as shown in Fig. 9: base, bin, lib, notes, and robot. The base directory contains many tools for calibrating vision, locomotion, etc. (see Fig. 10). Many of these tools are not currently used. The bin directory contains scripts and binaries. Mostly it just provides a convenient directory that one can put into a shell PATH variable in order to run programs quickly. The lib directory contains third-party libraries such as XML libraries and the JOGL [14] (Java bindings for
OpenGL [15]) that was added. The notes directory contains Sony AIBO® Model information as well Frequently Asked Questions and other miscellaneous notes. The robot directory contains all the code and configurations that are compiled into binaries and placed onto the Sony AIBO® memory stick. This is the directory in which searching started for the vision module.

Inside the robot directory is a vision directory (see Fig. 11). The vision directory was the most likely place to start for changing the goal and beacon recognition. According to Sony’s OPEN-R® Level 2 Reference Guide [16] OFbkImageVectorData is the data structure used to send image data gathered by the robot’s camera to a programmer’s vision module used to process the image. Sony’s OPEN-R® SDK Programmer’s Guide shows how the stub.cfg file is used to provide a connection between an entry point and the programmer’s core class member function used to process an incoming event [17]. The vision directory’s stub.cfg contained a line, “Service : "Vision.YUV.OFbkImageVectorData.O", null, ResultCamera()”.

This line shows that a member function named ResultCamera() will be called
Fig. 10. Listing of base directory.

```
/base
  /calibrate
  /colour
    /colourcube
    /RC
      /RC
      /Tool
      /RoboComm
      /RoboShare
    /colourTableTools
  /classify
    /bitsNPieces
  /mc
  /ic
  /linearShifts
/client
  /client
  /share
  /Configuration
/debugging
/offlineBehaviours
/offvision
  /learning
/orient
/robolink
  /actionplot
/RoboCommander
/SimpleRoboCommander
  /WalkLearner
  /CPlaneClient
  /JointDebugger
  /QTPro
  /JoystickController
/subvision
/udpcheck
/varianceanalyser
/walkBase
/worldviewer
```
to process the image data from the robot’s camera. Inside the Vision.cc file in the vision directory is the ResultCamera() function that gets called. The vision code can be traced from that function.

Vision.cc instantiates VisualCortex which in turn instantiates SubVision and SubObject. SubVision takes care of creating scan lines. It then processes each scan line in order to create Visual Features of some type/color (VF_BLUE, VF_YELLOW, VF_PINK, VF_OBSTACLE, etc.) [12].

Creating the Color Table

The rUNSWift code base uses a color table to map YCrCb values of a pixel to a color (yellow, blue, pink, white, etc.). The mapped color is helpful in object recognition. After scripts and makefiles were changed to get the rUNSWift code base [1] compiled and running, the 2005-2006 soccer field was set up and the dogs were placed on the field to play. The need for calibrating the robots (specifically create a new color table) was evident since the dogs did nothing but
turn in circles. After a few weeks of emailing students from UNSW including alumni Alex North who worked on rUNSWift’s vision module, the following sequence of commands (using existing rUNSWift tools available in the base directory) was used to create a new color table:

1. base/robolink/robolink <last octet ip address of dog>
2. bin/sendData <last octet ip address octet> 3 (where 3 represents “RLNK_YUVPLANE” which sends back images from the dog’s camera)
3. base/colour/rlog2bfl log.rlog (where log.rlog is the output file of robolink)
4. base/colour/rc/runRC

All of these tools used in the method above reside under the base directory.

The first executable, robolink, is used to connect to one of the dogs. This program continues to be connected to await data being sent by the dog. The second executable, sendData, is used to tell the dog what data to send back. In this it will be sending back camera images. After several images are received via robolink, the robolink program can be ended with control-C. Then the rlog2bfl can be run over the log.rlog file which was output by the robolink program. The output of rlog2bfl are *.BFL files which are actual images. These images placed in a separate directory can then be loaded by runRC all at once by choosing BFL->Open Directory in the Graphical User Interface provided by the rc tool. (Fig. 12 shows the rc tool prior to configuring colors.)
In order to create a color table in the \texttt{rc} tool, one can choose an image and a color (Orange, Blue, XXXX, Yellow, Pink, Blue robot, Red robot, Green Field, Grey, White, Black), then click on pixels in the image that match the color that is selected. After you have gone through and identified all the colors by choosing matching pixels in the image, you can click on the "Save NNMC" button to save your \texttt{nnmc.cal} file. (Fig. 13 shows the \texttt{rc} tool after colors have been configured.) This is a 2MB color table that is used by the vision module. This file can be placed in the \texttt{robot/cfg} directory where the \texttt{robot/ins} script will copy it to the memory stick when run.
After creating the new color table using images of the 2005-2006 soccer field, the dogs were set to play. This time the robotic dogs no longer turned in circles, were able to localize on the field, and play soccer.

Changes for the New Soccer Goals

The rUNSWift vision module was examined to determine what changes needed to be made in order to recognize the new 2007 soccer goals [3]. The new 2007 soccer goals differ from the previous year’s goals in that the new goals are of slightly different height and width. The new goals no longer have a solid back that is the same height as the entire goal. The back of the new goals
are only 100 mm high vs. the old full height of 300 mm. However, the front of the goal is now 350 mm high including the crossbar and its total width is 1000 mm including the two side bars.

Due to these overall dimension changes, the estimated dimensions used by the rUNSWift code base [1] needed to be changed. UNSW had created several Field Definition files—FieldDef.h and OsakaFieldDef.h. Another file was created called ChicoFieldDef.h where the GOAL_WIDTH was set to 100 (cm) and GOAL_HEIGHT was set to 35 (cm). These variables are used for known constant obstacles on the field as well as checking the sanity of an observed goal’s size.

Due to the overall changes in the appearance of the goals—meaning the lower 100 mm back and the crossbar toward the front—there was concern that the vision code would no longer recognize them as goals. So, first, the straight-on view of the goal was considered. If a person (or robot in this case) were to look straight at the new goal it would appear almost like a picture frame. The crossbar and the two side bars would form the top and the sides of the frame and the 100 mm back of the goal would form the bottom of the frame (even though the back of the goal is further back from the crossbar and side bars).

Later other views of the goal from different angles were considered. In each of these cases the image ranged from a skewed picture frame to a solid post.

Having knowledge of the way the new goals would be seen allowed the goal recognition to be assessed. All pieces—the crossbar, side bars, and
backing—needed to be recognized as part of a single goal of the appropriate size. The file SubVision.cc adds all Visual Features (representing portions of scan lines of an image) into a features list. Later this features list is parsed by SubObject.cc in the method pruneFeatures() where all Visual Features of type VF_BLUE and VF_YELLOW are placed into a goalFeatures list. When the findGoals() method is called in SubObject.cc, the first method called is groupFeatures(). The groupFeatures() method groups Visual Features that are of the same type (VF_BLUE or VF_YELLOW in this case) which are vertically above or below one another within a specified number of scan lines. (See Fig. 14 for a class diagram of the main classes and methods used in the vision module.)

The Visual Features associated with the goals are formed from horizontal scan lines. The Visual Features themselves are portions of those horizontal scan lines. So, in effect, the Visual Features representing the goals will be grouped as shown by the pink lines in Fig. 15. Fig. 15 shows that the Visual Features will form a goal object of the appropriate size that will then be able to pass the sanity checks for the aspect ratio.

Changes for the New Beacons

The vision module was then examined to determine what changes were needed to recognize the new 2007 beacons. The new 2007 RoboCup soccer field has only two beacons vs. the four beacons in the 2005-2006 soccer field. The two new beacons are placed at mid-field on each side of the field.
These new beacons no longer have a pink stripe in position A or position B near the top of the beacon. These stripes are now only yellow and blue. If a person were to stand on the field looking toward the blue goal, the beacon on the left side of the field would have a blue stripe at position A (top most position) and a yellow stripe at position B. The beacon on the right side of the field has the two colors reversed.
The 2005 rUNSWift SubVision.cc code originally tried to detect the beacons by looking for a string of pink pixels greater than a specified threshold (TH_BEACON_PINK = 5). Since the beacons no longer have a pink stripe, blue was chosen to be used (although yellow could have been used as well). The color blue was initially a random choice. However, after some scanning through images in the rc tool after color configuration, it seemed that the white walls often appeared yellow in places. The blue color was considered to be a better choice since fewer false positives were expected.

Beacon detection was complicated by the fact that there is a goal that is blue and one that is yellow—the same colors used in the beacons. The blue portions of the horizontal scan lines that are processed could be part of a beacon or the blue goal. Since at this point it is unknown which is the case, the Visual Features of type VF_BLUE are created after the number of blue pixels have exceeded the TH_BEACON_BLUE = 4 threshold (vs. the TH_GOAL_BLUE = 12 threshold which might have excluded the creation of the beacon blue Visual
Features). These VF_BLUE Visual Features are then just placed into a features list. The SubObject.cc pruneFeatures() method parses the features list so that Visual Features of type VF_BLUE and VF_YELLOW are placed into a goalFeatures list as mentioned previously. Now the Visual Features of type VF_BLUE must also be placed into the beaconFeatures list. (See Fig. 14 for a class diagram of the main classes and methods used in the vision module.)

When the SubObject findBeacons() method is called the groupFeatures() method is called just as it was for the goals. This groups all of the VF_BLUE Visual Features that are vertically above or below one another. This allows all VF_BLUE Visual Features of a beacon to be in one group and all the VF_BLUE Visual Features of a goal to be in another group. After these Visual Features are grouped, classifyBeacon() is called. This method determines the top most blue pixel and bottom most blue pixel in the blob. It then searches for a minimum number of yellow pixels (six) within a two-pixel radius of first the bottom of the blue blob then the top. If enough yellow pixels are found within the two-pixel radius of the bottom of the blue blob, a BlueYellow candidate beacon is said to be found. If enough yellow pixels are found within the two-pixel radius of the top of the blue blob, a YellowBlue candidate beacon is said to be found. The candidate beacon’s height and position is then estimated and sanity checks are run. In this case the sanity checks just ensure that the beacon is above the horizon.
Note that a group of Visual Features which represent the blue goal may also be processed by the beacon code. However, not enough yellow pixels will be found below or above the group to consider it a beacon and the group will be discarded.

One should also note that beacon VF_BLUE Visual Features will be processed by the goal code. The goal sanity checking code will discard the group of Visual Features that represent the beacon because they will be seen as “floating”. (The VF_BLUE blob will not be seen as sitting on the green field.)

Simple changes (not described here) were made to other files due to the number of beacons changing from four to two, the beacons’ positions on the field, and the change in beacon colors. These files include FieldLineVision.cc, gps.cc, gps.h, PyEmbed.cc, SanityChecks.cc, SlamChallenge.cc, VisualCortex.cc, and VisualCortex.h.

Changes to Debugging Tools

In order to support the new beacons one of the debugging tools had to change as well. This tool, called SubVision, receives vision data wirelessly from a Sony AIBO®. The SubVision tool receives RLNK_SUBVISION and RLNK_SUBOBJECT messages from the robotic dog as well as many other potential messages. The RLNK_SUBVISION messages contain information about the Visual Features that have been found by SubVision.cc in the camera images that the robot has processed. The RLNK_SUBOBJECT messages contain
information about the objects (i.e. goals, beacons, ball, obstacles, etc.) found by SubObject.cc.

The tool SubVision was modified to display only two beacons with the correct colors in its graphical user interface. It was also modified to receive information for only two beacons. Notice the Objects tab in the SubVision GUI in Fig. 16 has only two possible beacons (BlueYellow and YellowBlue), whereas there were four (BluePink, PinkBlue, YellowPink, and PinkYellow) in the previous version. Also notice that there are only five objects listed in the “Flag” section of the graphical user interface, whereas there used to be seven. When a beacon or goal is detected in an image and displayed as in previous figures, the Objects tab will display information about the appropriate object. The Flag section of the Objects tab will also highlight in red the object detected. (Changes to other tabs were also made, but were not used in testing of this project.)

In Fig. 17 you can see the image that is displayed for the BlueYellow beacon. Notice that the beacon is drawn in the correct colors.

Version Control

In order to keep track of the changes referred to previously, version control was used. The version control program chosen for the task was SubVersion [18].

The original rUNSWift 2005 code base was checked into trunk. Then this version was “tagged” by copying it to a tag called “runswift2005”. A branch
was also created from trunk called “spring2007”. The spring2007 branch was where script and makefile changes were checked in to allow the rUNSWift codebase to build correctly in the environment. These initial script and makefile changes were tagged as “prebeacons2007”.

Another branch called “beacons2007” was created from the previously described spring2007 branch. This branch was used to make all the vision changes needed to support the new soccer goals and beacons. This was done
so that other members of the team could work on the code known to be working (spring2007 branch) while the new vision changes were made that might temporarily break the code. After the vision code was tested, the beacons2007 branch was merged to the spring2007 branch where further work could continue that included the new vision changes. A tag, “postbeacons2007”, was created identifying the spring2007 branch after the beacons2007 branch was merged into it. The version control structure described above is shown in Fig. 18. and the list of files affected on each branch is listed in Appendix A.
Fig. 18. SubVersion version control structure.
CHAPTER IV

FINDINGS AND RESULTS

SubVision Tool Results

The SubVision tool created by the University of New South Wales [1] previously mentioned was used to test the vision and localization changes. The AIBO® was configured by the sendData tool to send back RLINK_SUBVISION and RLINK_SUBOBJECT messages. These messages were received and interpreted by the SubVision run on a wireless laptop. The following sections detail the results.

BlueYellow Beacon

One of the Sony AIBO® dogs was placed on the field facing the BlueYellow beacon as Fig. 19 shows. The SubVision tool showed that the BlueYellow beacon was recognized by drawing the beacon with the correct colors over the Visual Features (portions of scan lines of the same type/color). Fig. 17 shows the image displayed by the SubVision tool.

YellowBlue Beacon

The same Sony AIBO® dog was placed on the field facing the YellowBlue beacon as shown in Fig. 20. The SubVision tool showed that
the YellowBlue beacon was recognized by drawing the beacon with the correct colors over the Visual Features. Fig. 21 shows the image displayed by the SubVision tool.

Fig. 20. Blue dog facing YellowBlue beacon.
Blue Goal

The Sony AIBO® was placed on the field looking toward the blue goal as shown in Fig. 22. The SubVision tool showed that the blue goal was recognized by drawing a blue box over the grouped Visual Features. Fig. 23 shows the image displayed by the SubVision tool.

Fig. 22. Blue dog facing blue goal.
Yellow Goal

The Sony AIBO® was placed on the field looking toward the yellow goal as shown in Fig. 24. The SubVision tool showed that the yellow goal was recognized by drawing a yellow box over the grouped Visual Features. Fig. 25 shows the image displayed by the SubVision tool.

Notes on Testing Using SubVision Tool

Although the Sony AIBO® ERS-7M2 delivers images to the code (through callbacks) at a rate of 30 images per second, SubVision messages (such as RLINK_SUBVISION and RLINK_SUBOBJECT) are only delivered to the SubVision tool at a rate of 1 message every 5 seconds. Each of the tests (for each goal and beacon) was done at varying degrees (approximately 0, 30, 45, 60, 90, 120, 135, 150, and 180 degrees) in a semicircle around the landmark (see Table 1). Each position was done at least twice for at least 10 seconds.
each. This means that a \texttt{RLNK\_SUBOBJECT} message would be sent twice per test—a total of 4 times per position.

Playing Soccer

Localization was only tested from a high level. The modified localization code was determined to be working by looking at two of the game states—the ready state and the “playing” state. A state diagram is shown in Fig. 26. Although Fig. 26 references six states, for the author’s purpose only the “ready”, “set”, and “playing” states are discussed here.

Ready State

The robotic dogs can be placed in the ready state individually by a single tap on the Sony AIBO® ERS-7 head (or on the back switch on the Sony AIBO® ERS-210) or all at once by clicking the “Ready” button on the Game Controller application used at the RoboCup competition. In the ready state “the
robots walk to their legal kickoff positions” [3, p. 10]. Approximate kickoff positions are shown in Fig. 27. Fig. 27 illustrates expected kickoff positions when the red team is kicking off. Notice that there is one dog on the red team closer to the ball and the rest are further back. The blue dogs stand further back from the ball defending their own goal awaiting the kickoff.

The localization in the ready state was determined to be working since the dogs found their way to their kickoff positions as in Fig. 28. Fig. 28 shows the positions of the dogs after the ready state has been in progress for a bit. It shows the positions when the blue team is kicking off. Notice that two of the blue robotic dogs are forward—one directly in front of the ball to kick off the other in the rUNSWift “Supporter” position [1]. Three of the red dogs are further back on their own side of the field awaiting the kickoff by the blue team.
### TABLE 1

RESULTS OF SubVision TESTING

<table>
<thead>
<tr>
<th>Degrees</th>
<th>Yellow Goal Recognized</th>
<th>Blue Goal Recognized</th>
<th>BlueYellow Beacon Recognized</th>
<th>YellowBlue Beacon Recognized</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>30</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>45</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>60</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>90</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>120</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>135</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>150</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>180</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Set State**

The set state is the state after the ready state and prior to the playing state. The dogs are allowed to move their heads and tails but are not allowed to “move their legs or locomote in any fashion” [3, p. 10]. The set state is triggered from the ready state either by 1) tapping the ERS-7’s head (or the ERS-210’s back switch) a second time, 2) clicking the “Set” button in the GameController’s graphical user interface, or 3) the ready state automatically transitions into the set state after a timeout by the GameController. Although localization occurs
during the set state, it was not used to determine whether or not the localization code works since there is no visual feedback from this state. It is only explained here for clarity sake.

**Playing State**

The playing state is the state in which the robots play soccer. The playing state is triggered from the set state by tapping the ERS-7’s head (or the ERS-210’s back switch) a third time or the “Play” button is clicked in the GameController’s graphical user interface.
Localization was determined to be working if the robots appeared to be trying to head in the direction of their opponent’s goal and score. Figs. 29 through 34 display this process.

Fig. 29 shows the robotic dogs playing soccer just after the red team kicked off. One of the blue dogs has possession of the ball [20]. Fig. 30 shows the same blue dog as in Fig. 29 still in possession of the ball moving toward the yellow goal with other players grouped around the ball as well [20]. Fig. 31 shows a blue dog kicking the ball further toward the yellow goal. Here the dog is breaking away from the pack and pushing the ball ahead of it [20]. Fig. 32 shows the same blue dog as in Fig. 31 just prior to kicking the ball into the yellow goal [20]. The dog has no obstacles between itself and the goal. Fig. 33 shows the same dog as in Fig. 32 in the stance used to “kick” the ball with his nose [20].
Fig. 28. Dog positions on blue kickoff.

Finally, Fig. 34 shows the same dog as in Fig. 33 scoring a goal by kicking the ball into the yellow goal [20].
CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Summary

This project began with the examination of the existing 2005 rUNSWift Four-Legged League code and the new 2007 Four-Legged League Soccer Rules [1], [3]. The new soccer rules had a couple of environmental changes that impacted the vision module of the 2005 rUNSWift code.

One environmental change was the new soccer goals that were a slightly different size and shape than the previous years’ soccer goals. The 2005 rUNSWift code was versatile enough to recognize the new goals with only slight modifications to the expected size of the goals.

A second environmental change given by the 2007 Four-Legged League Soccer Rules was the beacons. In previous years there were four beacons placed at one quarter and three quarters of the way down on each side of the field. The old beacons were YellowPink, PinkYellow, BluePink, and PinkBlue where the yellow beacons were closest the yellow goal and the blue beacons were closest the blue goal. The 2007 soccer rules specify the use of only two beacons. These two beacons are placed at midfield on each side. If one is looking toward the blue goal the BlueYellow beacon is on the left side and the YellowBlue beacon is on the right side of the field. Since the new beacons
contain no pink color, the code was changed to look for runs of the color blue. This was only slightly complicated by the fact that one of the goals is blue as well. In addition, expected positioning of the beacons had to change.

**Conclusions**

The changes made to the 2005 rUNSWift Four-Legged League code which were presented and discussed in this manuscript allow the robotic dogs to play on a field built to the specifications of the 2007 Four-Legged Soccer League Rule Book. The robotic dogs can localize within the new field environment as well as score goals during a soccer competition.

However, during the testing of this project, intermittent spurious readings were encountered when the dog faced a goal from midfield and further back. After some investigation, it was found that items outside of the dogs’ immediate environment (above and beyond the white walls that enclose the soccer field) may have been causing the errant detection of beacons within or around the goal.

Testing of this project also revealed that when a dog such as the goalie faced into one of the new soccer goals, the dog did not detect the goal—or any other significant object for that matter. The dog continued to face into the goal not recognizing where it was. In order to find an acceptable solution for this problem, more work needs to be done in this area.
Recommendations

In order to deal with errant beacons observed in testing, the author recommends that the color table be altered so that many of these darker pixels in the background are not recognized as blue or yellow pixels. It would be better that some of the darker pixels in the beacons and goals not be classified as blue or yellow. The author speculates that the algorithms used to detect beacons and goals will still be able to recognize these landmarks without overfitting the color table to those landmarks.

In order to prevent false positive beacon detection, the author also recommends that additional sanity checks are implemented in the beacon detection code. Currently the only sanity check is that the beacon must reside above the horizon. Additional sanity checks may include searching for the white portion of the beacon below the lower color of the beacon and possibly aspect ratio checking between the colored portion of the beacon and the white portion.

In regards to the goalie being unable to identify the goal when it is in the goal, the author recommends behavior changes in order to compensate for the goal not being within the camera image. Currently the dogs already employ a “no green backoff” where the dog moves back when it can no longer see the green field. Some testing has shown that this backoff may not be long or far enough for the dog to see landmarks (goals or beacons) in the environment which may change the course chosen by the dog. Some investigation into the “no green backoff” behavior should be made to compensate for this problem. The
author also suggests that the behavior module initiate additional head movement or body turning when no landmarks are found in an image. This may allow the goalie to find the goal post.

In addition, the author strongly recommends that in order to facilitate testing and experimentation of robots for use in the RoboCup competition a static environment be made available. This means that a test soccer field should be set up and available at all times for testing purposes. Repeated set up and tear down of the field is infeasible. Much more progress can be made if a test field is readily available.
REFERENCES


FILE CHANGES IN VERSION CONTROL

The following files were added/changed on the spring2007 branch then tagged as “prebeacons2007.”

added /base/RoboCommander/roboCmd
added /GameController
added /lib/jogl.jar
/base/client/Makefile
/base/colour/colourcube/Makefile
/base/colour/Makefile
/base/colour/rc/Makefile
/base/colour/rc/runRC
/base/RoboCommander/Makefile
/base/robolink/actionplot/Makefile
/base/robolink/actionplot/rlog2action.cpp
/base/robolink/Makefile
/base/robolink/robolink.cc
/base/subvision/calibrateEdge.cpp
/base/subvision/FormSubvisionImpl.cpp
/base/subvision/FormSubvisionImpl.h
/base/subvision/FormSubvision.ui
/base/subvision/gpsinfo.cpp
/base/subvision/wireless/Wireless.h
/bin/dogtel

The following files were added/changed on the beacons2007 branch then later merged to the spring2007 branch.

/base/subvision/flagObject.cpp
/base/subvision/FormSubvisionImpl.cpp
/base/subvision/FormSubvisionImpl.h
/base/subvision/FormSubvision.ui
/base/subvision/gpsinfo.cpp
/base/subvision/houghDisplay.cpp
The following files were added/changed on the spring2007 branch after the prebeacons2007 tag was made.

```
added /base/classpath.linux
added /base/classpath.win
added /bin/dir_aliases
/bin/GameController
/robot/PyCode/Makfile
/robot/GameController/GameController.cc
/robot/GameController/GameControllerConfig.h
/robot/GameController/GameController.h
/robot/GameController/Makfile
/robot/GameController/stub.cfg
/robot/share/RoboCupGameControlData.h
```

All of the preceding files listed in Appendix A were tagged as “postbeacons2007” on the spring2007 branch.