Northern Bites 2006 Team Report

Henry Work
Prof. Eric Chown
Jesse Butterfield, Ferd Convery
Patrick Costello, Jeremy Fishman
Tracy McKay, Brendan Mortimer
Matthew Murchison, Quentin Reeve
George Slavov, Johannes Strom, Yi Zhuang

Department of Computer Science
Bowdoin College
8650 College Station
Brunswick, ME, 04011-8486, USA
hwork@bowdoin.edu
echown@bowdoin.edu
`http://robocup.bowdoin.edu`

February 21, 2007

Abstract

This document serves as a team report for the Northern Bites Robot Team that competed in the RoboCup 4-Legged League in 2006

1 Introduction

The Northern Bites 2006 season grew from small beginnings and accelerated into a project with large code base, a big development team, and a solid first-time performance at RoboCup 2006. The team was only one of two first-year teams to gain a bid to the World Championships where it finished roughly in a tie for 10th place.

The Northern Bites is a unique team. It hails from a small college (less than 1600 students) with an even smaller Computer Science department (averaging about 8 majors a year) with no graduate program. All team members are undergraduates – not necessarily majoring in Computer Science – most of whom do their work as an extracurricular activity without credit and while balancing a full course load. The primary interest of our team's participation is the educational value gained through the challenging research areas and diverse work necessary to field a RoboCup team. In line with our educational goals, we use very little code from other teams.

In 2005 the Northern Bites was created and managed by a single dedicated senior honors student who wrote the entire code base by himself and took the team to the US Open 2005 held in Atlanta, USA [4]. In the summer of 2006, new members signed on to the project and decided to start fresh. Lots of early development work was put into understanding the Sony Aibo/OPEN-R platform. The fall of 2005 was focused on making low-level modules (vision, motion, etc) functional, recruiting new members to the team, and working on the necessary development and debugging tools to maintain the expanding code base. It was not until January of 2006 that we were able to begin to design basic behaviors, initially written in C++, that utilized vision and motion systems to produce results loosely resembling soccer. Development sped up in the spring as seven students took independent studies in RoboCup development with the team's adviser, Professor Eric Chown. This meant that the team finally had members with a time commitment to the project.

We had to make a number of design decisions to come to grips with the team's limitations. This meant that certain things (e.g. using sensors other than the camera) were scrapped unless we were certain we could put them to good use. Things that we definitely wanted to work on (e.g., behaviors) had to be put off until we could at least see the world reasonably well and move around in it fairly quickly. One of the crucial things that we did was to put in an organizational structure sensitive to the construction of our team. For example, students that only worked on the team in their spare time were given jobs that were useful, but didn't require a tremendous amount of knowledge about the overall team. These tasks often involved data collection, kick tweaking, and other sorts of things that were necessary, but not time critical. This also freed up time for the students that were the main developers. If they could ask some of the floaters to do more menial jobs then that precious time was available to them to code. Perhaps the most crucial element of our strategy was to give control of the team to the students. Eric Chown, as the team's adviser, was the ultimate authority and decision maker, but the team captain (Henry Work) ran the squad on a day-to-day basis. Since students were not simply doing exactly what they were told – but had freedom to make choices – it gave them a sense of ownership. This provided the precious motivation that undergraduates often lack. Such motivation was especially important in light of the fact that many of the team members were doing this in their spare time, were not paid, and were not working towards any sort of dissertation.

Despite the impediments, the team came together fast enough and showed enough improvement to win acceptance at the at the World Championships in Germany. The incentive provided by this acceptance spurred the team on to develop code even faster and the reward came with two wins at the U.S. Open in Atlanta 2006 (one was in a friendly). In 2005 the team had not even managed a single goal. In Atlanta the team was frustrated by the slowness of developing in C++; this led the team to make the switch to Python as many other teams had previously. This changeover was not complete until there were only two weeks left before the World's Championships. The result was a complete rewrite of all of our behaviors. Under the circumstances our expectations were not exactly lofty. However, we had learned a great deal in Atlanta. Probably the key lesson was that it was crucial to get to the ball as fast as possible, even at the expense of having a less accurate grab. Consequently, this became the focus of a lot of what we did leading up to Germany.

We went to Germany as only one of two brand-new teams to the Four-Legged League and ended up qualifying for the final 16. In the round of 16 we won one of our matches 5-1 and captured tenth place (based on goal differential) and came close to making the top eight. We are extremely proud of the progress that we made in 2005-2006 and we are extremely excited to compete in 2007. This team report does not serve the purpose that many other reports do in showing off the newest research and innovations made within the league. This report documents the successes and failures of a first-year RoboCup team – in an old league – trying to catch up to teams with long histories, and without the benefit of relying on any other code base. We hope that this report will aid other small school teams trying to start up a new and fresh RoboCup team – in any of the leagues – and will encourage them to come up with original ways to approach the problems.

2 System

2.1 Modules

Our 2006 code base had three main modules, or 'Aperios Objects': Vision, Motion, and Comm. This layout thankfully allowed the team's motion and communications modules to maintain a high frame rate (and thus, walk fast) despite speed issues in early versions of both the vision and localization systems that plagued the Vision module (Secs. 3, 5). The team decided on the Subversion code management system to keep track of the expanding code and the expanding number of developers who work on it. Subversion offered the team tremendous flexibility while maintaining the code base's integrity among many different developers.

2.2 Python

One advantage of being a relatively new team is that it is possible to more or less restart from scratch at virtually any time. Among the lessons that we learned early in competition was that we simply needed better tools and a better development environment. Early behaviors in the winter of 2006 were written in C++ and were very difficult to modify: they required turning off the dog, re-compiling the code, re-writing the memory stick, and rebooting the dog – even for the simple changing of constants! What we did then is fairly typical of how we got our team up to speed – we drew inspiration from what other teams had done, even while implementing our own solutions. Other RoboCup teams have successfully implemented interpretive languages such as Lua [6], Perl [1], and Python [15, 16]. These languages are not only considered to be more sophisticated but also do not require compilation and allowed for a layer to be embedded on top of the C++ and then be dynamically reloaded while the dog remains turned on. Thus, using one of these languages produces a much faster development cycle. We made the decision to implement a port of Python 2.3.3 written by rUNSWift 2005 team while beginning initial behavior testing in C++ [14]. We chose Python because of familiarity of it within the Bowdoin CS department and also some extensive knowledge within the team.

Our Python system wasn't fully running until about a week and a half before RoboCup 2006. Amazingly, however, with the fully harnessed power of Python, we were able to do more with behaviors in a week than we were able to do in the previous six months with C++. The speed of development increased immensely by giving us finer control over our behaviors and decreasing time between modifications to seconds instead of minutes. One of the comments we got in Germany from teams that had seen us in Atlanta was that they were amazed by the amount of progress that we made between Atlanta and Germany. This can largely be attributed to our switch to Python (and the adrenaline rush of competition).

3 Vision

3.1 Vertical Scan

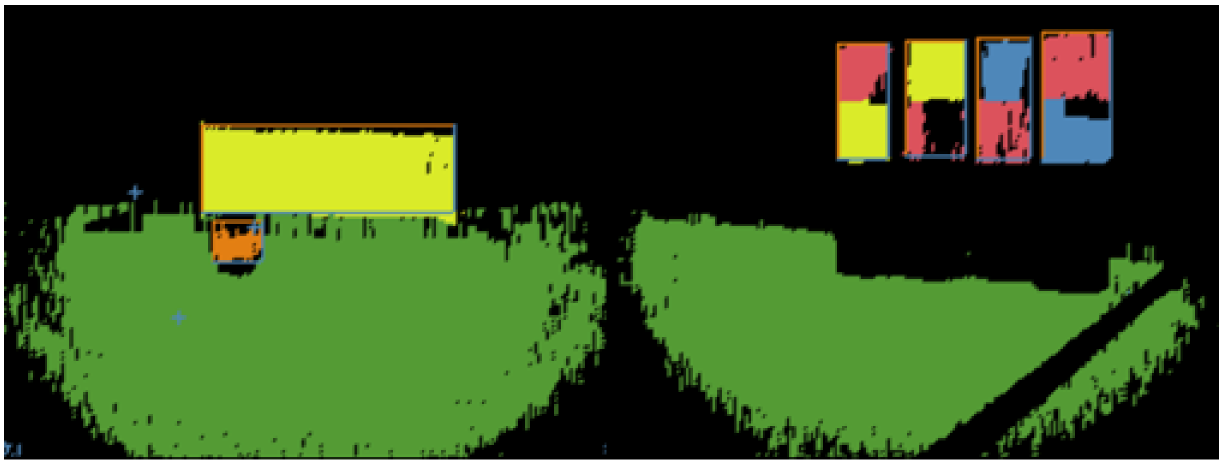

Figure 1: *Two examples of the Vertical Scan system.*

For a good part of the 2006 development year, the team used a recycled version of the 2005 vision system. The system, dubbed 'Vertical Scan' was used in the code base up until after the 2006 U.S. Open. The system was slow and extremely basic. In 2005 the goal simply had been to get something working. In 2006 we started working toward building a more complete system.

3.2 Sub-Sampled

A good bit of research and development was done following Alex North's research on 'Sub-Sampled' Vision processing [7]. The idea that using the separations between colors as a basis of vision, which are robust over lighting changes and do not require significant color calibrations required under the standard Look Up Table (LUT) approached used by so many 4-Legged League teams, was very exciting to us. This approach also has more practical purposes outside of traditional 4-Legged league lighting setups. However, upon our own implementation, we discovered that it required a lot more calibration and due to poor lighting in our lab the separations between colors were more easily found using a standard LUT approach. Due to mounting amount of work needed to simply get the dogs' playing soccer, development in this area of research was scrapped in favor of a 1-table lookup that was more easily implemented and required less testing.

3.3 Color Calibration

One of the critical developments made by our team was in building a useful calibration tool. Our goal was to have an online system that could quickly and accurately build our color table. The crucial insight that made this possible was the work on edge detection done by other teams [11, 7]. In a nutshell, if the dog was able to accurately determine edges then those edges could be used to segment the image. Once the image is segmented then it is easy to build a tool that allows the user to choose a color, then click on a point and essentially teach the program that the entire region is the chosen color (this is done by a simple recursive algorithm that terminates at the edges). The points in that region can then be used to fill in the LUT. Our team had already developed software called AiboConnect [2] that made the development of this tool rather simple. AiboConnect allows desktop machines to communicate with the Aibos and to send commands and receive sensor information. Essentially we built the calibration tool within AiboConnect and very quickly had a tool that could build accurate color tables in a very short amount of time (we can build a very good calibration in well under an hour). The tool could also easily be converted to provide input to a machine learning method such as Support Vector Machines (SVMs) that have been successfully used by other Robocup teams [9, 5].

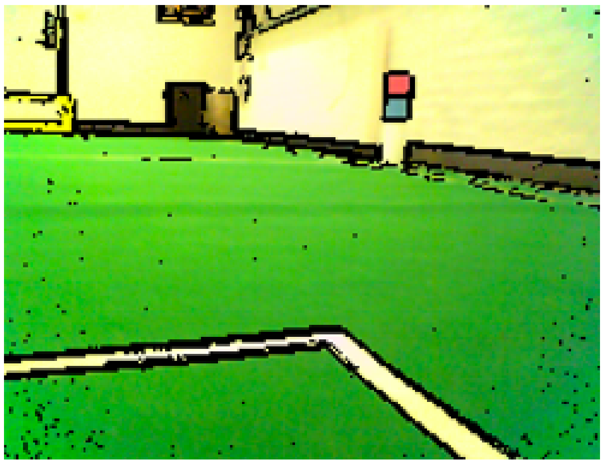

Figure 2: *Calibration tool on a sample image with edge detection.*

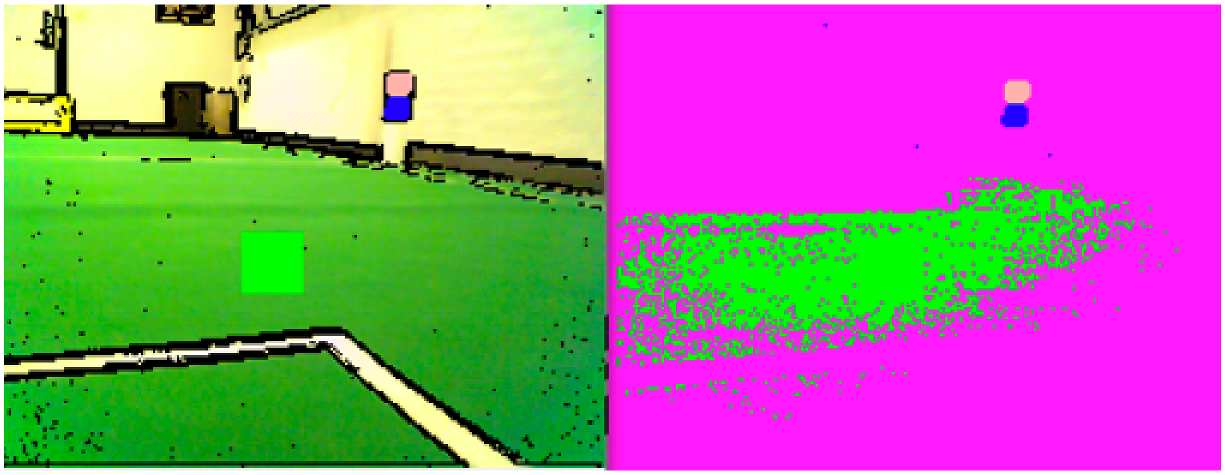

Figure 3: *Calibration Tool after three mouse clicks. The left-hand image shows three areas where the mouse was clicked, two on the post, one on the carpet. Although the algorithm operates recursively we limit the total area it can cover to a fixed rectangle size. The right-hand image shows how the system classifies the image based on its updated LUT. The magenta areas are places where the LUT hasn't learned a value yet.*

3.4 Blobbing

The team uses a fairly standard implementation of run-length encoding. We scan the image and collect up runs of identical colors. These are stored in a data structure and processed after the scanning process. During that processing we use fairly standard filters to try and screen out noise.

3.5 Landmarks

All distance and angle measurements were estimated to the focal point of the camera using ratios of the blob heights and widths compared to what they typically measure at 1 meter away. This was simple to calibrate but suffered from that very simplicity. When the object was only partially in the image frame we experienced problems. We worked out a lot of these cases and basically re-estimated using inferred data and by relying on relative constants of the playing field. For example, we only used height of the goal blob when it was fully within the vision frame, and width only when height wasn't, and neither when the goal blob was creeping off screen in both x and y directions. We typically used heights for the post because of blurring due to poor camera settings (see Sec. 3.7) and over-reporting blob widths.

We built a special case for the ball distances, because they are so important, where we use scan lines to find three realistic points on the ball and then used geometry to figure out the ball's center coordinates on the image frame. This worked really well for when the ball showed up in the corners of the image frame and was particularly close to the dog.

Clearly the solution to this problem is to have a quality pose estimate for the robot. We simply did not have time to do this for last year's team. We have already corrected it for this year's team.

3.6 Dog / Gap Recognition

We mapped red and navy in our color table so as to do simple blobbing of the uniforms worn by both teams. We didn't take the time to consider difficult cases like when multiple dogs appear in the image or estimating the pose or field location of the seen robot. We did some work on dog recognition, but did not push it too far as we didn't find it all that important to most of our behaviors. One issue we did have to confront was the poor separation between the orange ball color and the red uniform colors at RoboCup 2006. We didn't have this issue in our lab and so we weren't ready for the issue before Germany. There were several reasons for this, but mainly we have discovered that our lab was very poorly lit as described in Sec. 3.7. The result was that once we got into well lit rooms our vision system behaved differently. Chasing our own or another team's uniform certainly wasn't optimal, so we devised some checks on the spot to deal with the issue – which we effectively did after some re-calibration.

We conducted research on 'gap' detection or 'green space' using a similar technique used by the NUBots [2]. From a dog's perspective, it doesn't care what obstacles are in front of it, only that it needs to move around them. This applies to dogs that are very close in which their uniforms aren't visible, referees, and negates the need for complicated and difficult dog recognition software.

3.7 Camera Settings

For most of last year we were operating with camera settings that caused bad blur due to a slow shutter speed. This was a necessity because our lab was so poorly lit. Unfortunately when debugging our vision system we generally dealt with fairly static images where the dog was not moving much if at all. This means that we had almost no experience with issue like blur. Luckily, we used fast shutter speeds in Germany and saw a noticeable improvement in object recognition when the dog was moving around. This was obviously one of the problems we encountered by coming late into RoboCup.

4 Motion

4.1 2005 System

The 2005 system for Motion was a simple system that used scripted joint commands for walking and moving the head around. This system had its advantages: they were easy to program and very adaptable. However, the walk was slow compared to most teams and were unable walk orthogonally or spin (or combinations of the two) with various degrees of speed.

4.2 UPenn Engine

In the Summer of 2005 the very first project was to incorporate another teams' inverse-kinematics walking engine. The decision to port another teams' walk code was not taken lightly: we strongly believe in providing fresh and innovative solutions to RoboCup problems. However, we chose to do to this due to the massive amount of work in front of us at the time. Moreover, the new team had no experience with the Aibo platform or the OPEN-R system, understood the amount of time necessary to create a well calibration walk engine, and most honestly, had little programming experience under its belt. Finally there was simply the issue of what we might learn from building such a system – this is not the type of material typically taught in our liberal arts curriculum. Of course the downside to this was that we had to figure out just how the ported system worked.

Porting the inverse-kinematics system was pretty instrumental in understanding the Aperios operating system as well as gaining a relatively fast walk. We chose the UPennalizer's 2005 Walk Engine based on their modular code release and well-documented code [3, 1]. Initially, we had little idea of how the motion system worked and it wasn't until some months later when we were experimenting with behaviors did we understand how to combine orthogonal and spinning directions in the system. Happy with a relatively fast walk and with many other projects outstanding, we did not attempt any machine learning to improve the robot's speed. We did hand-tweak the parameters of the system in the Spring of 2006, but didn't end up using parameters that resulted in a faster walk until Germany, where it was clear that our speed was going to severely limit our impact against the better teams. We switched over to the faster gait midway through the competitions under the assumption that even if we couldn't grab the ball reliably, it was better to just to get in the way.

Along with the motion system, we ported over the UPenn head inverse-kinematic system that guided the head to points in the x,y,z coordinate system relative to the focal point of the lens. We did not see the benefit of this system because we were embracing very simple behaviors and the regular joint angles, already familiar to us, were more intuitive.

4.3 Special Moves

We wrote our own system for creating special moves like kicks, goalie saves, victory dances, and head pans. We wrote two queues, one for the head and one for the legs, that handled a series of joint angles and controlled how many frames the action wanted to operated for (speed). We eventually ran into issues where behaviors would occasionally queue one action every frame for hundreds frames which resulted in thousands of joint angles being queued. This was fixed with just a cap to the queue and better debugging to make sure that behavioral we were only calling one action at a time.

4.4 Future Work

In the month after RoboCup 2006 we made a variety of improvements on the motion scale. We wrote behaviors to be able to quantify the dead reckoning of the dog using landmarks so that the whole process is very automated and easy to revisit if our gait changes. We experimented with machine learning using policy gradient [12] and hill climber [8] methods. We also did some research to see how the results faired when scaled over multiple robots.

5 Localization

5.1 First Try

Our first localization approach was very crude. The deadline of competition was looming and we had a high percentage of the total code base to finish. We decided to go with the easiest system we could think of. Our first assumption was that there would be a high degree of error relying on dead reckoning (particularly when the robot had been moved by a referee) – this is somewhat a reflection of the background our team leader has in human cognitive mapping, as humans are lousy at dead reckoning. Dead reckoning, we believed, was not particularly useful and should be added last, if at all; our system was to be entirely based on visual recognition of posts and goals. This also reflects a general desire that our team should be as cognitive as possible. There is no evidence that humans do anything remotely like Monte Carlo localization and we initially hoped we would be able to do something more cognitively plausible. This proved to be an unfortunate decision.

This 'vision-only' approach caused us many problems. Firstly, our vision system, especially in the beginning, was slow and inaccurate. Distance estimates were very noisy, particularly if the landmark was only partially observed. Our vision system was only running around 8-10 fps when we took the team to Atlanta in April. Moreover, a solely vision-based localization system is going to have issues when you don't see landmarks; when the dog chases the ball and doesn't see landmarks the system will break down pretty quickly.

Another issue with our first localization system was that it was very blunt. It used essentially no trigonometry and instead filled an array representing the field with points in a circle with a radius of the distance from the object and centered at the object. It then looked for areas with a high density of points. This creates two problems. The first is that seeing the same post ten times drowns out seeing a new post once. In order to get a good position estimate, you not only needed to see several posts, but you needed to see them a similar number of times. The second problem was that it was incredibly slow; working with an array meant that everything ran at a speed of $O(n^2)$. Our first attempt at localization left us with a system that gave inconsistent results and slowed down the dog, leaving us worse off than with no localization system at all. While we were able to improve the performance of this system somewhat with some hacks, dead reckoning, and by switching from arrays to lists, the system could not provide positions that were precise or accurate enough, so it was scrapped.

The decisions made in localization are somewhat reflective of the inexperience of undergraduates. The great part about giving undergraduates control of the code is that it is a real motivator as they feel a sense of ownership in the team. However, that must really be tempered by the understanding that they simply do not have a wide enough background in some areas (like localization) to work without supervision. There are very good reasons why most teams use Monte Carlo localization or Kalman Filters – we should have paid them more heed.

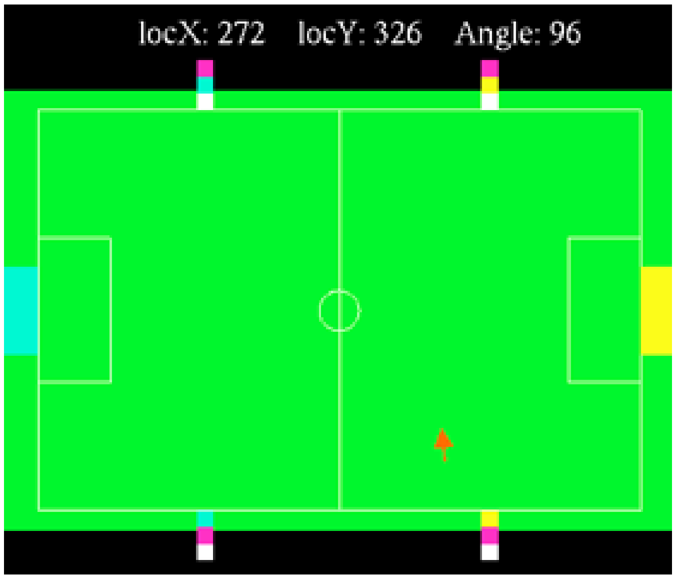

Figure 4: *Localization visualizer*

5.2 Second Try

Our second localization approach took the opposite extreme, eliminating particles entirely and relying on a more intuitive approach. Each frame it calculated possible positions based on distances to recently seen posts and goals (this time using trigonometry, which gave us much better results), line recognition, and dead reckoning. It then looked for agreement among the five or so position estimates created, picking whichever one came up most often. This system was far superior to the original in that the results were much more accurate and did not slow the dog down at all. However, the results were still not very precise, with the robot's position sometimes oscillating by as much as 100cm.

This system also did not deal with noise effectively, sometimes giving results that were off by half a field length. This system showed some real promise, but it did not appear that it would ever achieve the accuracy of the established methods of a Kalman or a Monte Carlo filter, so it too was scrapped. Here is another sign of the issues of an all-undergraduate team: the student that was developing the new system graduated and there was no one in place to take it over.

5.3 Competition

Localization was used very little at the competitions in Atlanta and Germany. We attempted to position ourselves on the field for kickoff and also for basic ball searching routines per dog. This achieved some success, but due to noisy estimates we were not able to get the accuracy desired and oftentimes ran off the field! We did have a ball localization system worked out – as we recognized this as very important – but our self localization wasn't reliable enough to give us any good ball estimates. We also attempted to use heading to choose the shortest direction to spin towards the opponent's goal once we had the ball grabbed. This was only occasionally successful due to heading inconsistencies. In Atlanta, the localization slowed the system down to a crawl. In Germany, the updated system was a lot faster but only marginally more accurate.

5.4 Future Work

We have chosen to implement an Extended Kalman Filter based off the NUBot's success with the system's ability to be both fast and deal with noise effectively [10]. The filter has been fully written in Python as of February 2007 and has shown significantly better results in both self and ball localization.

6 Communication

We implemented a UDP-based communication system after reading reports that most other teams in the league switched to the protocol due to its low-latency and the game's fast-paced, dynamic environment [10, 13]. Each dog broadcasts pertinent information to every dog on the same subnet. We used a header to differentiate our team from other teams. To deal with latency issues, we included the time in milliseconds since the GameController or the Referees set the dog into 'Play' state in each packet. When a dog received this packet, it compared this time to its own estimate to get a rough idea of how old the received packet was. If it was too old, we simply ignored it.

The only utilized part of the communicated info between teammates was each dog's distance estimate to the ball. We were able to successfully keep two dogs from approaching the ball at the same time. There were very few incidents where neither went for it or when two dogs would keep switching between approaching and hanging off. We also set this value to a special number when the ball was grabbed so that teammates knew that the dog had control of the ball.

We subscribed to the idea that there should be no role negotiation or team captain role assignment. Each dog had its own rules of its role, and when there were conflicts, the decision defaulted to the dog with the higher player number. We fully plan on keeping this spirit as we retool communicated information between dogs in 2007.

7 Behaviors

7.1 Strategy

As one might expect from a team that started writing its behaviors so late, our team was basically a "see the ball and chase it" team. Once we had the ball, we spun towards the goal and kicked. The relative success of our team speaks to how much work there still is to do in RoboCup (and conversely how many opportunities still exist for innovation). Our strategy is also very reasonable for a team with a poor localization system. We focused our energies on finding the ball, moving to it quickly and then trying to move it in the right direction. Ultimately this is an example of something basic about RoboCup as in real sports – there is no substitute for fundamentals. For our team the fundamentals mainly involved seeing the ball, moving to it, grabbing and kicking. The fundamentals that we lacked mainly involved localization. Our first priority this year has been to work on more fundamentals - a real localization system, an improved vision system, pose estimation, etc. Despite our desire to work on cool behaviors we have adhered to a strict plan to get the low levels in order before attempting the higher levels.

7.2 Chase and Grab

Controlling the ball gives a player an unique advantage in that other players cannot get the ball themselves once it is firmly within one player's grasp. Realizing that the grab is essential to play, we began working on this skill very early. Initially, we needed to work out all the behavioral elements of controlling the motion and vision systems in sync – building low and high level behavior methods – to actually approach the ball from afar. Once close, we need to align ourselves as well as possible, and then when in range we would execute pre-defined head motion that lowered the head in a manner that didn't knock the ball away from the dog's body and then lowered the mouth joint to control the ball in place. We also toggled with our motion system parameters to find a gait where the dog could spin and walk forward with the ball reasonably well without knocking the ball away.

One tricky part with the system was controlling the head effectively. We didn't want the head to oscillate too much, and so we included thresholds which only moved the head where there was significant ball angle movement. Moreover, we wanted to keep from moving the head in more than two degrees of freedom, drawing an example from UPennalizer's code base [3]. Using only the head yaw and pan joints or just the head neck and pan joints kept the ball tracking movements more effective.

Our grabbing system also relied heavily on our vision system. Estimating distances and angles to the ball when very close to the dog's body was important, as was seeing far away balls on the horizon. Our goal for 2007 is to adopt the NUBot policy of ‘no artificial slowdown’ while approaching the ball for a grab and then to force improvements in the lower level systems until we grab as consistently and quickly as possible.

7.3 Goalie

One lesson learned in 2006 was that while the Goalie seems like a simple player to write, in many ways designing a good goalie is even harder than building a good player. For us this is mainly because, unlike with a player, a purely reactive goalie will be horrible. Goalies must necessarily have a reasonable localization system and a sense of what is going on in the game. In a nutshell our Atlanta goalie was worthless, it even spent time facing the wrong goal. As with everything else we furiously redesigned it in preparation for Germany. Because of the time limitations we again opted for the simplest approach. Our goalie would basically stay in the goal mouth and only move laterally except when clearing the ball. The resulting goalie, written in less than a week, was surprisingly effective. The lesson here was: if you know you have limitations then it is best to try and design behaviors that minimize them. Ironically, now that we have the fundamentals in place for our robots, writing a decent goalie will be much harder this year. We’ll have to make decisions about when the goalie should go out to cut down on shooting angles, clear the ball, etc. Of course this is the fun part.

8 Summary

The main thing we learned in our first real year of competition is reflective of comments made by teams like the NUBots previously: the most important factor in success is moving as fast as possible to the ball. In the U.S. Open we did not move quickly to the ball, but focused on being ready to grab, and consequently our team was consistently outmaneuvered by good teams. In Germany we concentrated on moving as fast as possible and our team showed dramatic improvement. Indeed our team’s success is rather amazing in light of how poor our localization system was. At some level this is probably a simple reflection of the state of four-legged soccer: for most teams the teamwork aspects have not evolved far beyond grab, turn and shoot. One of our main goals for 2007 is to incorporate elements of strategy in our play. We have noticed, for example, that teams do not seem to have many (or any) set plays even on kickoff. RoboCup soccer is actually somewhat more like hockey at this point than soccer because of the rules and the small number of players. It seems, therefore, that adopting some hockey strategies might be useful: putting dogs in front of the goalie to screen its vision, having the goalie move in and out of the net to cut down on shooting angles, etc. For us this is where the fun really lies. Can we make our team look like a team instead of a collection of individual players? We think the answer will be ”yes.”

Last year our team was light on innovations and heavy on catch-up. Hopefully some of the lessons we learned might help other new teams in the future (even assuming a change in platform). It is still possible for a new team to catch up with the pack (it remains to be seen whether or not we can catch up to the top-flight teams). So far RoboCup has provided a great learning platform for our students and now it is time to take the next step – to make it a research vehicle where we can help push the state-of-the-art further.

Acknowledgments

This work was sponsered by the NSF under Grant No. 0092605. The authors gratefully aknowledges that support. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF).

References

[1] G. Buchman, D. Cohen, P. Vernaza, and D. Lee. The university of pennsylvania robocup 2005 legged soccer team report. Technical report, The University of Pennsylvania, 2005.

[2] E. Chown, G. Foil, H. Work, and Y. Zhuang. Aiboconnect: A simple programming environment for robotics. In *The proceedings of the 19th International FLAIRS Conference*, 2006.

[3] D. Cohen, Y. Ooi, P. Vernaza, and D. Lee. Upennalizers code release. Technical report, The University of Pennsylvannia, 2004.

[4] G. Foil. Developing an autonomous robot soccer team at an undergraduate institution. honors thesis, Bowdoin College, 2005.

[5] N. Henderson. Digital image processing in robot vision. Technical report, University of Newcastle, 2005.

[6] H. Kobayashi, J. Inoue, T. Osaki, T. Okuyama, S. Yanagimachi, T. Sasaki, K. Oi, S. Hirama, A. Ishino, A. Shinohara, A. Kamiya, S. Abe, W. Matsubara, and T. Nakamura. Jolly pochie team report. Technical report, Kyushu University and Tohoku University, 2005.

[7] A. North. *Object Recognition from sub-sampled image processing.* PhD thesis, University of New South Wales, 2005.

[8] M. Quinlan. *Machine Learning on AIBO Robots.* PhD thesis, University of Newcastle, 2006.

[9] M. Quinlan, S. Chalup, and R. Middleton. Application of svms for colour classification and collision detection with aibo robots. Technical report, University of Newcastle., 2003.

[10] M. Quinlan, S. Nicklin, K. Hong, N. Henderson, S. Young, T. Moore, R. Fisher, P. Douangboupha andS. Chalup, R. Middleton, and R. King. The 2005 nubots team report. Technical report, University of Newcastle, 2005.

[11] T. Rofer, T. Laue, H-D. Burkhard, J. Hoffmann, M. Jungel, D. Gohring, M. Lotzsch, U. Duffert, M. Spranger, . B. Altmeyer, V. Goetzke, O. v. Stryk, R. Brunn, M. Dassler, M. Kunz, M. Risler, M. Stelzer, D. Thomas, S. Uhrig, U. Schwiegelshohn, I. Dahm, M. Hebbel, Nistico W, C. Schumann, and M.Wacher. Germanteam 2004. Technical report, Germany, 2004.

[12] P. Stone, , and N. Kohl. Policy gradient reinforcement learning for fast quadrupedal locomotion. Technical report, The University of Texas at Austin, 2004.

[13] P. Stone, K. Dresner, P. Fidelman, N. Kohl, G. Kuhlmann, M. Sridharan, , and D. Stronger. The ut austin villa 2005 robocup four-legged team. Technical report, The University of Texas at Austin., 2005.

[14] W. Uther, C. Sammut, B. Hall, A. Blair, B. Hengst, C. Lam, D. Whaite, T. Wong, J. Xu, C. Chan, and K. Pham. runswift code release. Technical report, University of New South Wales., 2005.

[15] W. Uther, C. Sammut, B. Hall, A. North, A. Sianty, N. Morioka, W. Chen, and J. Shammay. runswift code release. Technical report, University of New South Wales, 2005.

[16] M. Veloso, P. Rybski, S. Chernova, C. McMillen, J. Fasola, F. vonHundelshausen, D. Vail, A. Trevor, S. Hauert, and R. Espinoza. Cmdash'05: Team report. Technical report, Carnegie Mellon University, 2005.