2004

Development of a Soccer-Playing Dynamically-Balanced Mobile Robot

Brett Browning
Carnegie Mellon University

Paul E. Rybski
Carnegie Mellon University

Jeremy Searock
Carnegie Mellon University

Manuela M. Veloso
Carnegie Mellon University

Development of a Soccer-Playing Dynamically-Balancing Mobile Robot

Brett Browning, Paul Rybski, Jeremy Searock, Manuela Veloso
Carnegie Mellon University
School of Computer Science
5000 Forbes Avenue, Pittsburgh, USA
{brettb, mmv,prybski}@cs.cmu.edu, jsearock@andrew.cmu.edu

***Abstract*—In this paper, we make two contributions. First, we present a new domain, called Segway Soccer, for investigating the coordination of dynamically formed, mixed human-robot teams within the realm of a team task that requires real-time decision making and response. Segway Soccer is a game of soccer between two teams consisting of Segway riding humans and Segway RMP-based robots. We believe Segway Soccer is the *first* game involving both humans and robots in cooperative roles and with similar capabilities. In conjunction with this new domain, we present our work towards developing a soccer playing robot using the Segway RMP platform and vision as its primary sensing modality. As Segway Soccer is set in the outdoors, we have developed novel vision algorithms to adapt to changes in lighting conditions. We present the domain of Segway Soccer, its inherent challenges, and our work towards this goal.**

Keywords-component; mobile robots, games.

I. Introduction

There has been considerable research into both human-robot interaction [12], and multi-agent teams [8,9,10]. Additionally, since the inception of RoboCup robot soccer [2], there has been considerable research into multi-robot teams operating in adversarial environments. To our knowledge, however, there has been no work to date that combines these attributes; namely, to examine human-robot interaction within an adversarial, multi-robot setting where humans and robots are team members with similar capabilities and no clear role hierarchy.

We are developing a new game, which we call Segway Soccer, that aims to fill this void. Segway Soccer is a game that requires mixed teams of humans and robots to cooperate to achieve the maximum reward in an adversarial task. To ensure interesting cooperation, both humans and robots are equipped with similar capabilities. We achieve this difficult task by requiring that both humans and robots use the same drive platform – the Segway™ platform developed by Segway LLC (Figure 1).

Our goal is to create a task that requires advanced robot intelligence, combined with robust human-robot interaction skills. We hope to extend the powerful aspects of RoboCup – competition, an adversarial domain requiring fast decisions, a well understood task – to incorporate human-robot interaction. The need for this new domain lies in the lack of study for human-robot interaction where decisions need to be made quickly. As robots become more integrated into society, they will inevitably have to interact with humans and/or legacy robots in complex tasks. For some of these tasks, decisions may need to be made quickly and roles of both humans and robots may not be clearly defined a priori.

Figure 1. The Segway RMP (left and right) and Segway HT (right) platforms developed by Segway LLC (*http://www.segway.com*).

In this paper, we describe our work towards developing a robot capable of participating in Segway Soccer. As this new domain is set in the outdoors, compensating for variable lighting conditions and less structured environments, but still retaining the ability to make and act on decisions quickly is a challenging task. We describe our initial solutions to meet this challenge.

The format of the paper is as follows. In Section II, we describe the specifics of Segway Soccer; its rules, structure, goals, and challenges. Section III describes our proof of concept, a soccer playing Segway RMP, which uses vision as its primary sensing modality. Finally, we conclude in section IV and present our on-going work.

II. Segway Soccer

In this section, we concretely describe the rules of Segway Soccer and the common hardware platforms used. We begin by describing the rules of the game of Segway soccer.

The Game

Segway Soccer is a game between two teams playing on a grass field in an outdoor environment with an orange, size 4 soccer ball. Teams can consist of humans, robots, or a mix of humans and robots. Figure 2 shows the field structure. The field consists of a grass surface in an outdoor environment. White tubular markers are placed around the field to indicate the field boundary. Each goal is uniquely colored and is delimited by two posts. A human referee maintains control of

This research was sponsored by the United States Army under Grant No. DABT63-99-1-0013. The content of the information in this publication does not necessarily reflect the position or the policy of the Defense Advanced Research Projects Agency (DARPA), the US Army or the US Government, and no official endorsement should be inferred

the game and transmits signals verbally, as per a normal referee, and via wireless communications to the robots via an assistant referee armed with a laptop and a wireless network. Team members may be robots, humans, or robots and humans. In all cases, the Segway platform is used to ensure each team member has identical physical capabilities. Humans ride Segway HT platforms, while robots use the Segway RMP base. We describe these platforms further in the ensuing section. Both humans and robots are colored to allow for easy team identification.

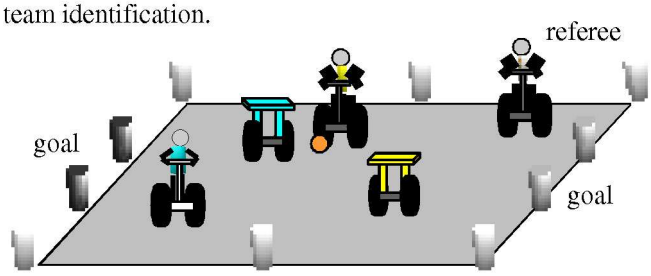

Figure 2. The Segway field. Teams consist of humans, robots, or robots and humans using the Segway platform, and an orange size 4 soccer ball.

The field dimensions follow a scale law as a function of the number of players on the field. For *n* players on each team the field dimensions can be calculated as:

$$length = \frac{n}{11} \cdot 100\,m, \quad width = \frac{n}{11} \cdot 60\,m \qquad (1)$$

As both Segway HT's and RMP's carry considerable mass, and are able to reach speeds of 8mph or greater, safety is a primary concern. To address this problem, the game follows a flow more familiar to Ultimate Frisbee[1]. When play begins, ball possession is decided with a coin toss. Afterwards, players gain possession based on proximity to the ball when it is "free". Once a player obtains possession, opponents are not allowed to contest the ball thereby preventing any unnecessary contact. Players are also not allowed to move with the ball (dribble), and instead must pass the ball to one another for the team to maintain possession. A time limit will be enforced on how long possession can be maintained by a single player before the ball must be passed on to a teammate before possession is overturned. When the ball is passed, the first player on any team to come within a specific distance of the ball will gain possession. The same player cannot re-aquire possession of the ball until after another player has obtained possession. Possession is also changed if the ball is kicked out of bounds or if a goal is scored. Although primarily a safety measure, this rule also ensures that players *must* pass the ball to advance. As a direct consequence teamwork, rather than purely single robot skills, becomes essential. The goal of exploring intelligent teamwork is therefore achieved.

Although the rules defined thus far allow for a multi-agent, adversarial game to be played, they do necessarily enforce human-robot interaction. If, for example, humans prove considerably more capable than their robot teammates, one can expect humans to dominate possession leading to little human-robot interaction opportunities. Should robots prove more capable than their human brethren, the reverse situation happens. Either case is undesirable. Our solution to this problem is to require that *both a human and a robot* be part of the sequence of passes leading to a goal score. How effective this solution is, remains to be seen.

Segway RMP as a Platform for Robotics Research

The Segway platforms, invented by Dean Kamen, are unique in their combination of wheeled dynamic balancing mobility. The human ridable Segway HT has two separately driven wheels and on-board computation that allows the platform to dynamically balance when a human is standing on it. The human rider controls the forward/backward velocity of the Segway by leaning forward to accelerate or backwards to decelerate. To turn, the rider twists a handle grip to turn in one direction or the other. This combination of controls are surprisingly easy to master even for the most novice of riders.

The Segway RMP, or Robot Mobility Platform, is the focus of this paper. The RMP consists of a Segway HT that has been modified by Segway LLC, to provide an extensible robot control platform. Figure 1 shows the Segway RMP and HT. The RMP consists of three modifications to the base HT platform. First, a CAN Bus interface is exposed to enable two way, high speed electronic communication with the platform. Second, the Segway's control software is modified to enable a computer to send direct velocity commands to the platform. The third change is to attach a large mass of approximately 50lbs at a height of about 50cm from the robot wheel base. This mass, consisting of multiple steel plates, serves the purpose of raising the robot's center of gravity. This is necessary to slow down the rate of falling over for the robot to enable Segway's control loop to operate effectively at a realizable frequency.

Commands to the Segway RMP can either cause the robot to move or modify the general operation characteristics of the robot. Motion commands have a speed-rotation format of $(v, \omega)^T$, where v is the forward velocity and ω is the rotational velocity. These commands act as set points for the Segway RMP's PID control loop. The control loop is a position controlled, meaning the robot will continue to move as commanded until it reduces the position error to zero or the PID integrators are reset. The additional commands can disable the Segway, reset the PID integrators, select different gain schedules, or adjust the velocity/acceleration scales. The different gain schedules prove useful for different weight/height arrangements. In addition to receiving commands, the Segway returns status information derived from its internal sensors. The state information is sent at 100Hz and includes:

- Pitch, roll, yaw angles and rates
- Remaining battery charge
- Wheel velocity, displacement
- Forward displacement

As a platform for robotics research the Segway RMP offers many unique features. First, it is a robust, extensible platform capable of extended operation both in terms of distance traveled and in operation time in both indoors and outdoors. Although operation distance and time depend upon

[1] *Rules for Ultimate Frisbee can be found at: http://www.upa.org*

terrain and use, figures of 16km and 3 hours are not uncommon. The Segway is able to move at speeds considerably faster than most robotic platforms. It can carry a significant payload, in excess of 100 kg. The mechanical arrangement of the Segway means that sensors can be placed to give a human perspective on the surrounding world without compromising the robot's stability or maneuverability. The dynamic balancing gives the robot a certain measure of active compliance, which is more than useful when collisions occur. The one caveat to the Segway, is that to maintain a stable balanced operation it must remain within ±20° of vertical. If the robot exceeds this limits, it automatically disables its balancing and promptly falls over.

III. Developing a Segway Soccer Player

We now describe our work to develop a Segway RMP robot base capable of playing Segway Soccer. To build any autonomous robot, one must develop a *complete* system involving perception, cognition, and action. We begin by presenting an overview of our approach, followed by a detailed discussion of the vision, skill learning, development environment, and hardware.

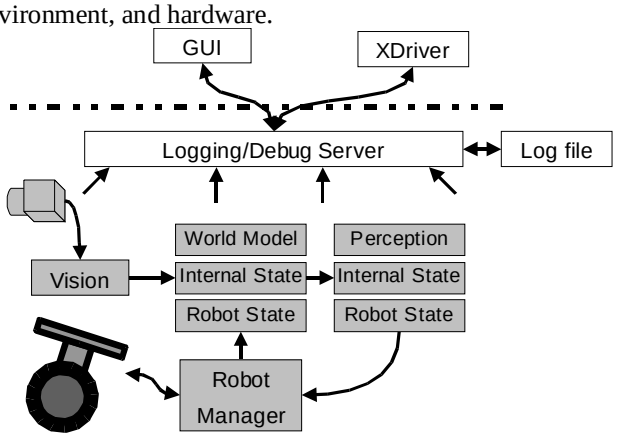

Figure 3. The control hierarchy used for the robot. The gray modules are the perception-cognition-action part of the system. The white are development infrastructure aids. Xdriver is a teleoperation program.

Overview

Figure 3 shows the complete control architecture for the Segway RMP. The gray boxes show the main processing path that makes up perception, cognition, and action. In a dynamic, multi-robot environment, where robots are moving at speeds approaching 3.5m/s (8mph), the ability to perceive and respond to situations in minimum time is essential. Hence, it is critical to overall robot performance that the control loop formed by gray modules operates at full frame rate, and with minimum latency, in addition to executing the correct command for each situation. The white boxes show the supporting development environment, which although not critical to the normal operation of the system is critical to the development cycle. Not shown in this figure are the mechanical components to support ball manipulation. We now describe each major component and its role in the overall hierarchy, namely; the robot hardware, vision and tracking, state representation, robot cognition, and the development environment.

Robot Hardware

Just as the control hierarchy can be broken into perception, cognition, and action, so too can the physical hardware be broken into sensing, computation, and actuators. Figure 4 shows the main physical hardware components. For sensing, the robot uses a single color CCD camera. Specifically, the robot uses a Phillips 690 Web camera with a wide-angle lens providing a Field-of-View of around 110°. The camera provides 320x240 color pixels at 30Hz in a YUV 420 planar format. The only other sensors are those internal to the Segway as discussed above. For computation, two laptops are used with one dedicated to the intensive task of perception procession and the other dedicated to communication with the Segway RMP via the CAN Bus. The latter occurs via the Kvaser LAPCan II PCMCIA card. The remaining cognition algorithms are distributed between the two laptops depending upon computational requirements. In the current arrangement, only motion control and interfaces with the Segway RMP and CMU board run on the actuation laptop.

Vision was chosen as the primary sensor for its high information content, low cost, and suitably to the task of recognizing multiple fast moving objects in a complex world. As cameras are now a consumer item and are therefore beneficiaries of competition in terms of device development and price reduction. Thus, for building *teams* of robots it makes sense to use an affordable sensor. Moreover, if suitable algorithms can be developed to realize the vast potential of vision its capabilities will far outweigh any comparably priced sensor. We return to this discussion shortly.

The final component of the hardware system is a mechanism for propelling the ball. Although it is possible to propel the ball with a human ridable Segway through weight transfer by swinging the feet through underneath the rider (see figure 1), this motion is not effective with the Segway RMP platform due to the lack of an actuated mass decoupled from the robot base. Instead, we require a separate mechanism for imparting energy into the ball. We have developed a pneumatic kicking mechanism for just this task.

The task of ball manipulation is to develop motions and/or mechanisms to transfer a maximum amount of energy to the ball in as predictable a direction as possible. This problem has been studied extensively within the RoboCup robot soccer community [3,9,10]. We can categorize these different approaches based on the actuation mechanism into: pneumatic, spring, motor, and solenoid based kickers. Due to the size and power requirements of the robot, a solenoid solution is clearly inappropriate. Motor based approaches, such as the rotating bar of or the driven plate of [3,10], although simple lack the power transfer capabilities at the needed size or raise significant safety concerns. This leaves a spring based solution, or a pneumatic one. A spring-based solution has many appealing factors; namely it offers high power to volume and power to weight ratios. The caveat is that it requires specialized mechanical hardware creating a design challenge, and more significantly a maintenance challenge. Thus, we used a pneumatic approach due to its simplicity and high power to weight ratio. Figure 4 shows the main components of the pneumatic system. Essentially, two pistons form the drive mechanism. Two 6V actuated solenoid valves 'actuate' the device, while the regulator and electronic

switch maintain a set pressure and refill the tank with an on-board compressor when the pressure drops too low.

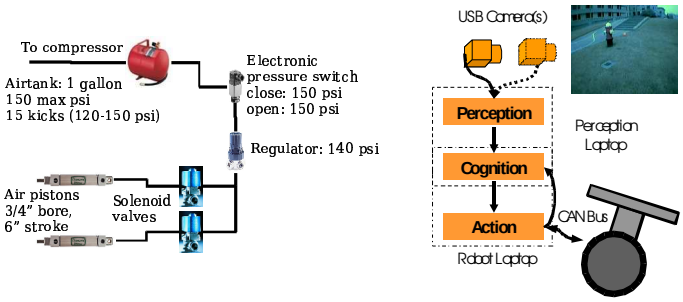

Figure 4. The main robot hardware components. The left shows the kicker mechanism, the right shows the processing architecture.

Vision and Tracking

As described above, for this domain there are few sensors that can compete with color vision for cost, size, information content, and latency. Due to the size and speed of the Segway RMP platforms, it is necessary for the robot to operate predominantly outdoors on grass fields. Thus, algorithms that are robust to changes in illumination and color variation are needed. Unfortunately, there is a lack of vision algorithms that provide the combination of robustness to color illumination and can operate at full frame rate on a moderate processor. Given the need to perform *all* computation related to perception, cognition, and action, the need to operate in real-time means that only a fraction of computational resources are available for vision procession. To our knowledge, there are no vision algorithms that offer all of these features.

A number of fast, color-based algorithms and freely available source libraries have been developed for static lighting conditions. In particular, CMVision [6], is one such library for very quickly extracting color blobs from images that has been widely used in RoboCup research. CMVision operates using a fixed color table to map from pixel color vectors to a symbolic color value. The actual color table is generated a priori either by GUI tools or via supervised learning algorithms. Once each pixel in the image is mapped to a symbolic color value, a fast connected component analysis is performed. The resulting regions are then reported allowing for higher level recognition and tracking algorithms to be run. Although very effective, the fixed color map means that CMVision is unable to adapt as lighting conditions and consequently color values, change. For example, if the sun moves behind a cloud the color and luminance of the visual scene can change quite dramatically.

To address this issue, we have extended CMVision to detect well contrasting objects under variable illumination conditions. Our algorithm extends CMVision by replacing the pixel labeling process with an adaptive one derived from knowledge of object geometry and color relative to the scene. The complete vision algorithm works in five steps, as follows:

1. Transform the color space via vector projection
2. Build a histogram of resulting 1D pixel values
3. Analyze histogram for a peak satisfying constraints
4. Run connected components from CMVision
5. High-level filter resulting blobs to find ball

The first part of the process operates by projecting each pixel $p \in (1...N)$ with color values $(y_p, u_p, v_p)^T$ onto a 1D intensity space via a normalized dot product operation commonly found in image tracking applications [7]. Specifically, the operation is:

$$I_p = \frac{(y_p\ u_p\ v_p)\cdot(a\ b\ c)^T}{|a|+|b|+|c|} \qquad a,b,c \in \{-6,...5,6\} \tag{1}$$

The values for the prototype vector $(a, b, c)^T$ are selected manually. For the orange soccer ball, a prototype of $(1\ -3\ 5)^T$. Thus, the ball must be bright and red/orange in color. A global histogram of the resulting intensity image is constructed, which is then searched for the brightest peak. We developed a simple peak detection algorithm that takes into account constraints including a minimum peak size, and area. Concretely, the algorithm searches for the brightest peak

1. Starting from $p = N_{max}$ find p such that

$$Hist(p) > p_{Hmin},\ \ Hist(p) \geq \max_{i=p..N_{max}} Hist(i) \tag{2}$$

2. Starting from $m = p$, find m such that

$$Hist(m) \leq min_{i=m..p} Hist(i),\ m < p \tag{3}$$

Where m and p are indexes constrained to $\{p_{min},...N_{max}\}$, $Hist(i)$ is the histogram count for intensity i, and there are $[0, N_{max}]$ possible intensity values (in practice $N_{max} = 255$). The found peak is only accepted if the area under the peak, A_m, is within the constraints: $A_{min} \leq A_m \leq A_{max}$. Thus:

$$A_i = \sum_{j=i}^{N_{max}} Hist(j) \tag{4}$$

The minimum value, m, is chosen as the threshold value for the image. Each pixel is then labeled using this threshold and the components connected to form regions. A number of high level filters are then run on the largest regions satisfying a minimum size constraint. Each filter produces a confidence estimate $c \in [0, 1]$ based on known geometric properties of the object, and the product of the confidences is used as the confidence for that region. For the ball, these estimates are the same as for [11]; an expected bounding box size, an expected pixel count given the boundary box size, and the shape of the bounding box. Figure 5 shows the algorithm in action.

Figure 5. The left image shows a raw image, the right the processed result with ball pixels labeled and a bounding box drawn around the identified ball.

Robot Cognition

Through our previous work, we have developed a control architecture for multi-robot control in adversarial, highly dynamic environments. The architecture, which we call Skills-Tactics-Plays [5], consists of Skills for low-level control policies, Tactics for high-level single robot behavior, and plays for team coordination. Here we briefly review the key components of skills and tactics, and then focus on the new developments for recording skills to speed up development.

Skills, tactics, and plays in the STP architecture form a control hierarchical. Here we focus on skills and tactics that form the components for single robot intelligence. A skill is a focused control policy for carrying out complex actions in the world. For this task, example skills include a particular method to kick the ball, a technique to prepare for a kick, or a method for stealing the ball from an opponent. A tactic encapsulates a complete single robot behavior such as shooting the ball in the goal, passing, acting as a goal keeper. Skills form the high-level action primitives for tactics, thus tactics affect the world by commanding skills to execute and passing parameters appropriately. Skills can be connected into a finite-state-machine for a given tactic. Thus, a tactic can perform a range of complex actions by triggering the appropriate sequence of skill execution. For example to shoot the ball at the goal, the *shoot* tactic executes a sequence of skills such as *gotoBall*, *positionForKick*, and when aimed at the target the final *kick* skill. Transitions between skills are controlled based on the perceived state of the world using a decision tree. Finally, following the usual behavior based approach [1], tactics and skills both execute in parallel and compute their decisions once per vision frame at 30Hz. The tactics set the parameters for skill execution, and possibly the executing skill, while the skill makes its decision on the world state and generates output for the robot low-level modules.

Figure 6 shows the skill state machine for the shoot tactic. To ease the complexity of skills, we have also developed a robot control module. This module implements a motion control algorithm as in [4]. Additionally, we have developed skill recording mechanisms which we describe next. Thus, the skills actuate the robot through the motion control module by setting a robot relative way point, through the motion playback module, or by directly actuating the robot (not shown). The robot command is sent to the Segway via the CAN Bus.

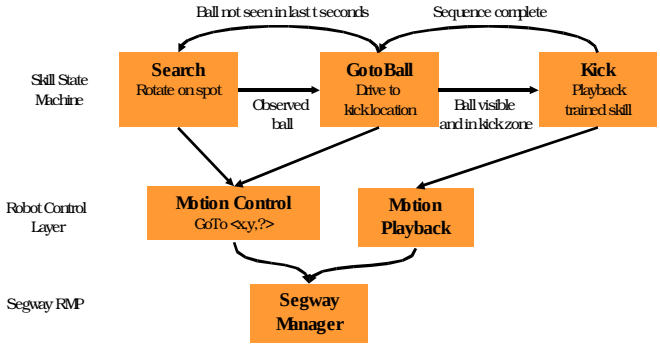

Figure 6. The skill state machine for the shoot tactic. Also shown is the motion playback portion of the skill recording mechanism.

In an adversarial domains, the efficiency, accuracy, and robustness of skill execution is a major factor in determining robot performance. Skill performance is a direct function of the robot mechanics, control system, and the local environment. As such, skills do not transfer well from one environment to the next, from one robot platform to another, and from simulation to reality. Finally, as skills are inherently tied to the low-level actions of the robot, they typically require many parameters to define their execution. Tuning these parameters by hand is time consuming and error prone. Thus, we desire a technique to enable rapid tuning of skill parameters.

We have developed a skill recording system, whereby a human operator can teach a robot a skill. The human operator guides the robot via tele-operation through a complex motion. The commands sent to the robot at each processing cycle are recorded to a file, which can then be edited manually if desired. The recorded commands can be played back at run time verbatim. The result is a complex motion which can be executed by a tactic as a skill.

To test the validity of this approach, we recorded a kick motion for the Segway RMP to kick the ball. To best execute a kick on rough terrain, the Segway RMP must first lean forward, drive through the ball actuating the kicker at the correct time, and then slow down to avoid running over the ball and robbing it of its momentum. Although writing an algorithm to generate such motion presents only minor difficulty, tuning the parameters to achieve the right behavior takes a non-negligible amount of work due to the need to test and adjust parameters. In contrast, it is relatively straight forward to provide the robot with a good example via tele-operation (especially if one is already experienced with driving the robot).

The mechanism described here can only be used to reproduce directly recorded motions. There is no ability to generalize beyond the examples it has been given. Our future work is to extend this approach to incorporate generalization mechanisms. With such an ability we hope to significantly reduce the complexity required to develop complex skills.

Development Interface

One of the key aspects to robot development that is often overlooked in the literature relates to the support infrastructure to aid development. In our experience, good development infrastructure can greatly ease the development burden, however, there has been no scientific study of what 'good' development infrastructure is. Based on our prior experiences, we have developed a number of infrastructure tools for the Segway RMP to aid development. Concretely, we have developed a Debug Server and GUI client for providing contextual text and graphic debug information at execution time. We have also developed a logging/playback system to be able to record what the robot 'sees', 'thinks', and 'does' and play it back for later analysis. Finally, we have developed an off-line vision testing tool to speed up the vision development process. We now describe in detail each of these components.

Figure 7 shows the GUI output for allowing a remote user to view several aspects of the robot's sensory and internal state. This is particularly useful for developing behaviors for the robot as one can quickly see what the robot's model of the world is. The GUI client programs connect to the Debug server (see figure 3) over a TCP socket. In the

RawRobotView display, the output of the vision system can be seen and the user can use it to view the output of the various tracking modules. For instance, the outputs from the ball or obstacle tracking modules can be directly viewed. Other output interfaces, such as the *RobotLocalMap*, show an ego-centric view of the robot and show the positions of detected objects around it, while the *RobotPoseView*, shows an ego-centric side view of the robot so that the reported tilt angle can be visualized. Finally, all of the text output generated from the soccer server, such as debug and state information, can be viewed on the *RobotTextWindow* display.

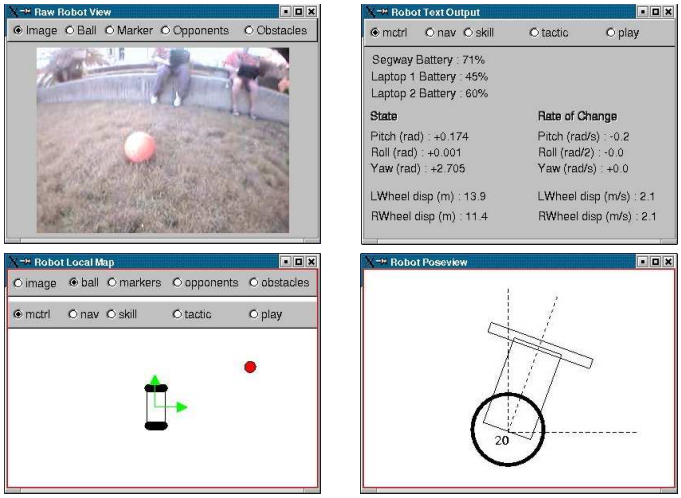

Figure 7. The GUI output showing the raw video from the Segway (top left), the robot's state real-time information (top right), the robot's local world model (bottom left), and a graphic of the robot state (bottom right). The GUI may connect to the robot 'live' or to a log player for off-line analysis.

The Logging server allows the Segway to record all sensor and state information at a configurable rate, so that it can be analyzed off-line as well as replayed. The latter is especially useful given the speed of action and the large information content. The user can request a number of different sensor channels to log, where these might be all of the robot's pose and velocity state information, the raw video, the segmented video, or even the specific positions of the tracked targets (ball and players). Logging all of the raw video data along with all of the robot state information is fairly processor intensive and is not typically done unless there is a specific need for it.

In order to test and debug the video processing algorithms, the raw frames of video can be loaded into a *vtest* program which will process each frame using all of the Segway's video processing code. This is particularly useful for gathering large amounts of video data for testing purposes. The robot can simply be tele-operated around the environments where it will be expected to operate, and new video processing code can be tested without having to drive the robot.

IV. Summary and Future Work

The Segway RMP and HT present a new and exciting robotics research platform. Based on this platform we have devised a new domain called Segway Soccer for investigating human-robot interaction within the confines of a real-time, adversarial task. We have developed the single robot capabilities to control a Segway RMP in an outdoor environment. Specifically, we have developed robust outdoor vision, a skill-tactic-play control hierarchy, and infrastructure to support skill training, logging, and playback. All of the algorithms described here have been fully implemented on the Segway RMP and run in real-time at the full frame rate of 30Hz. Using these algorithms the robot can successfully perform a shoot tactic in an outdoor environment with variable lighting. We have made available videos of the robot in action, including videos derived from the robot vision logs. We encourage interested readers to download videos of the robot operating at *http://www.cs.cmu.edu/~robosoccer/segway*. Our future work will focus on extending the behavior repertoire of the robot and moving towards mixed human-robot teams.

Acknowledgment

The authors would like to thank M. Sokolsky, D. Rozner, D. Govindaraju, L. Xu, and B. Sklar for their work on the Segway RMP robot. The authors would also like to thank Dr. Douglas Gage and Segway LLC. for their support in this project.

References

[1] R. C. Arkin. "Behavior-Based Robotics". MIT Press, 1998.

[2] M. Asada *et. al.* "An overview of RoboCup-2002 Fukuoka/Busan". AI Magazine, 24(2): pages 21-40, Spring 2003.

[3] S. Behnke et al., "Using Hierarchical Dynamical Systems to Control Reactive Beahvior." RoboCup-99:Robot Soccer World Cup III. Berlin: Springer, 2000, pp.189-192.

[4] M. Bowling and M. Veloso. "Motion Control in Dynamic Multi-Robot Environments". In M. Veloso, E. Pagello, and H. Kitano, (eds). RoboCup-99: Robot Soccer World Cup III, pp. 222-230, Springer Verlag, Berlin, 2000.

[5] B. Browning, J. Bruce, M. Bowling, M. Veloso. "STP: Skills, tactics, and plays for multi-robot control in adversarial environments". IEEE Journal of Control and Systems Engineering, under submission.

[6] J. Bruce, T. Balch, and M. Veloso. “Fast and Inexpensive Color Image Segmentation for Interactive Robots”. In Proceedings of IROS-2000, Japan, October 2000.

[7] R. Collins and Y.Liu, "On-Line Selection of Discriminative Tracking Features," IEEE International Conference on Computer Vision, ICCV'03, Nice, France, October 2003, pp.346-352.

[8] M.B. Dias and A. Stentz. "Opportunistic Optimization for Market-Based Multirobot Control". Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2002, 2002.

[9] M. Ferraresso, et al., “Collaborative Emergent Actions Between Real Soccer Robots.” RoboCup-2000:Robot Soccer World Cup IV. Berlin: Springer, 2001, pp.297-300.

[10] N. Kiat, Q. Ming, T. Hock, Y. Yee, and S. Yoh, “LuckyStar II-Team Description Paper.” RoboCup-2000:Robot Soccer World Cup IV. Berlin: Springer, 2001, pp. 543-546.

[11] S. Lenser, J. Bruce, and M. Veloso. CMPack: “A Complete Software System for Autonomous Legged Soccer Robots”. In Proceedings of the Fifth International Conference on Autonomous Agents, May 2001.

[12] M. Nicolescu, M. J Mataric, "Learning and Interacting in Human-Robot Domains", Special Issue of IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans , Vol. 31, No. 5, pages 419-430, C. C. White and K. Dautenhahn (Eds.), 2001.