Toward an Undergraduate League for RoboCup.

John Anderson[2], Jacky Baltes[2], David Livingston[3], Elizabeth Sklar[1], and Jonah Tower[1]

[1] Department of Computer Science
Columbia University
New York, NY, 10027, USA
sklar,jpt2002@cs.columbia.edu

[2] Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, R3T 2N2, Canada
andersj,jacky@cs.umanitoba.ca

[3] Department of Electrical and Computer Engineering
The Virginia Military Institute
Lexington, VA, 24450, USA
livingstondl@mail.vmi.edu

Abstract. This paper outlines ideas for establishing within RoboCup a league geared toward, and limited to, undergraduate students. Veterans of RoboCupJunior are outgrowing the league as they enter college and this has motivated us to develop a league especially for undergraduate students — the *ULeague*. The design of the league, presented here, is based on a simplied setup of the Small-size league by providing standard Vision and Communication packages.

1 Introduction.

With the rise in popularity of RoboCupJunior (RCJ) [4], a growing base of participants are graduating from high school and wanting to continue with RoboCup but are unable to because they do not have the resources required to enter the senior leagues. Some of these students may be attending a university that has an existing RoboCup team, so that they will perhaps have an opportunity to participate on a senior league team as advanced undergraduates. Other students attend universities where there is no RoboCup senior team and/or no robotics lab; and for these students, participation as undergraduates is not an option. We are thus motivated to create a RoboCup league especially for undergraduates, and we have spent recent months designing and prototyping this league — the *ULeague*.

The goal of the ULeague is to provide a stepping stone from RoboCupJunior to participation in the Small-size (F180) or Mid-size leagues. There is a significant leap in both expertise and resources necessary to be a competitive entrant in either of these leagues compared to the Junior league. The sophistication and costs of the robots used in the F180 league may be prohibitive. A typical Small-size team has robots that have omni-directional drives, dribble bars and powerful kicking mechanisms. Such a robot has four to six high quality motors and a powerful on-board processor to control them.

Each one of these robots costs around US$3,000. Added to this is the cost of high quality video cameras. These and other expenses typically drive the cost for a team to around US$20,000–US$30,000.

The ULeague is intended to provide a scaled-down and cheaper version of the Small-size league problem for undergraduate students. To achieve this goal, the ULeague factors out the most complex aspects of the Small-size league, namely vision processing and communication, by providing common platforms for these tasks, so teams may focus on robot development.

In addition to practical rationale, the ULeague also provides unique research challenges and opportunities. In the F180 league, it is possible for multiagent aspects of the game to have little impact on the performance of a team. A powerful dribble bar and kicker or other combination of physical features makes it possible for a single robot to be dominant. The physical constraints placed on robots in the ULeague are intended to force teams to rely more on coordination and cooperation to make progress in the game. Moreover, since a common architecture is employed, the ULeague can be thought of as a physical version of a simulation league. It is hoped that the ULeague will encourage collaboration with teams from the RoboCup Simulator League. This could mean faster dissemination of the research and progress made by teams in the Simulator league to physical robot competitions.

We have chosen the Small-size league as our primary model, since this league requires the least amount of space and the least expense in terms of equipment[4]. Given this, we have identified two major stumbling blocks for teams entering the Small league: *vision* and *communication*. So for the ULeague our idea is to provide a standard solution for these two aspects, provided by league organizers, and have teams build the rest (see figure 1).

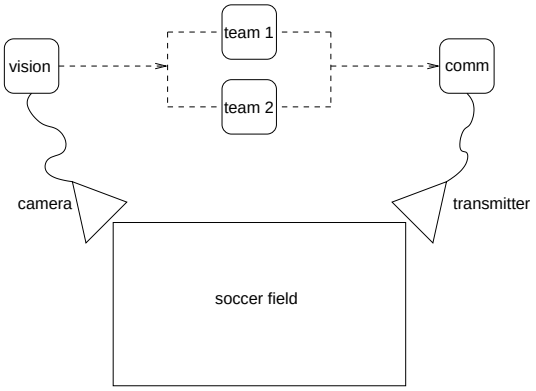

Fig. 1. High-level architecture of the ULeague.
The dashed lines represent an Ethernet connection.

[4] as opposed to the Mid-size, Four-Legged or Humanoid leagues

2 Vision.

The ULeague will use a standard vision software package to make it easier for teams to enter the ULeague and to speed up setup time at a competition. The current video server is the *Doraemon* package developed by Baltes [2], which is open source software released under the Gnu Public Licence[5]. Doraemon has been in development for over three years and has been used by several robotic teams in international competitions, including RoboCup. Doraemon includes real-time camera calibration, color calibration and object tracking components.

Several features separate Doraemon from similar software written for other global vision robotic soccer teams. It supports full 24-bit color. It supports maximum field capture rates of 50 (PAL) or 60 (NTSC) fields per second. It includes sophisticated Tsai camera calibration [5], allowing the system to calibrate the geometry of the scene from any view, which means that it is not necessary to have the camera mounted directly overhead relative to the playing field, nor is it necessary to add a wide-angle lens to the camera. It tracks objects in two or three dimensions (the latter requires using multiple cameras or stereoscopic vision); and it employs a clean object-oriented design that makes it easy to define different types of robots. Currently robots using two colored markers or bar codes are available, but there are also more sophisticated object recognizers that use only a single colored spot on the robot [2]. The developers are currently working on a pattern-recognition process using neural networks that does not require any markers [3].

Installation and setup of Doraemon consists of four phases: (1) setting up the camera and the viewing angle, (2) calibrating the camera geometry, (3) calibrating the rotation-translation matrix, and (4) calibrating colors. Each phase is detailed below.

2.1 Setup of the Camera.

In the F180 league, each team provides their own video camera and mounts it in their desired position. Most teams choose to place the camera directly overhead. This means that the local organizing committee must provide a physical structure above the playing field onto which teams can mount their cameras. However, the limited angle of view requires that a wide-angle lens be mounted on the camera in order to have a view of the whole playing field. Doraemon has more complex camera calibration routines and is not limited to overhead views (see figure 2).

2.2 Camera Calibration.

Doraemon's camera calibration uses the well-established Tsai camera calibration [5] which is popular among computer vision researchers. It is also suitable for global vision in robotic soccer since it can compute the calibration from a single image. The Tsai camera calibration computes six external parameters (x, y and z of the camera position as well as angles of roll, pitch and yaw) and six internal parameters (focal length, center of the camera lens, uncertainty factor S_x, and κ_1, κ_2 radial lens distortion parameters) of a camera using a set of calibration points. Calibration points are points in the image with

[5] `http://sourceforge.net/projects/robocup-video`

Fig. 2. A sample view of the playing field

known world coordinates. In practice, Tsai calibration requires at least 15 calibration points.

Doraemon uses a fast, robust and flexible method for extracting calibration points from the environment. A simple colored calibration carpet is used. The user selects a colored rectangle and specifies the distance in the x and y direction between the centers of the rectangle. Doraemon's calibration is iterative, so it can compensate for missing or misclassified calibration points.

2.3 Rotation Matrix.

The Tsai calibration results in the overlay of a world coordinate system over the scene. In principle this is sufficient to play soccer. However, Doraemon uses an additional rotation and translation matrix to establish the world coordinate system. Instead of being forced to compute the twelve parameters, the rotation and translation matrix allows one to rotate, scale, shear and translate the playing field. This results in a set of linear equations with three unknowns and allows a rotation and translation matrix to be established with only three points. The four corner points of the playing field are used (resulting in one more point than required), and a least-squared error approximation of the matrix is produced.

2.4 Colors.

Doraemon uses a sophisticated 12-parameter color model that is based on red (R), green (G) and blue (B) channels as well as the difference channels red-green (R-G), red-blue (R-B) and green-blue (G-B). Simple thresholding of the R, G and B channels is sensitive to the brightness of the color. Even on a robotic soccer playing field with its controlled lighting, it is difficult to detect more than four colors robustly using only these channels. Other color models that have been proposed in the literature are less sensitive to brightness (e.g. the HSI, YUV, or SCT models). However, these models are computationally expensive, which greatly impacts the performance of the video server. The color model used in Doraemon attempts to balance the best of both worlds. The difference channels in the color model are similar to the hue values in the HSI or similar models, but have the advantage that they can be computed faster than the HSI model.

2.5 Output of the Video Server.

Doraemon transmits the position, orientation and velocity of all objects being tracked to the team clients listening on the Ethernet. The messages are transmitted in ASCII via

UDP broadcast in a single package. Each line is terminated by an end of line character (`\n`). For the ULeague, each message contains eleven lines: two lines of header information, one line for ball information and eight lines for robot information. An example is shown in figure 3.

LineNum	content
0	`9 1073821804 0.00139797`
1	`-76.3836 -1820.48 2356.39`
2	`1 ball NoFnd 971.056 840.711 35 0 -2.31625 58.1464`
3	`0 b0 Found 1185.59 309.187 100 0.0596532 436.282 -43.083`
4	`0 b1 Found 1158.59 508.198 100 0.0596532 499.282 285.083`
5	`0 b2 Found 1086.95 1009.187 100 0.0596532 499.282 285.083`
6	`0 b3 Found 2185.59 309.187 100 0.0596532 499.282 285.083`
7	`0 y0 Found 989.95 304.588 100 -0.10185 413.528 -1.08564`
8	`0 y1 NoFnd 1689.95 1004.588 100 -0.10185 413.528 -1.08564`
9	`0 y2 Found 189.95 704.588 100 -0.10185 413.528 -1.08564`
10	`0 y3 Found 1789.95 1304.588 100 -0.10185 413.528 -1.08564`

Fig. 3. Sample output message from Doraemon.

The first line of each message contains (a) the number of objects that the video server is currently tracking; (b) the absolute frame number; and (c) the time difference in seconds between this message and the previous message. A client can use the absolute frame number and time difference value to determine if any frames were dropped by the video server.

The second line of each message contains the coordinates (C_x, C_y, C_z), in millimeters, of the camera with respect to the real-world coordinate system. This is used in distributed vision or stereoscopic vision applications and will not be used in the ULeague. The example shows that in this case, the camera was mounted 2.3m above the playing field (line 1 in figure 3).

Following this package header, there is one line for each object that the video server is tracking [6]. Each object line contains the following information:

- the type of object (`0`=robot, `1`=ball);
- the name of the object;
- whether the object was found in the image or if the video server did not find the object and predicted the positions based on previous motion (`Found` or `NoFnd`);
- the x, y, and z coordinates of the object, in millimeters;
- the orientation of the object in radians; and
- the velocity of the object in the x and y directions.

The names used for each of the nine ULeague objects are `ball` for the ball, `b0` through `b3` for the four robots on the blue team and `y0` through `y3` for the four robots on the yellow team (lines 2-10 in figure 3). The example shows that ball (`ball`) was not found and that the video server's best estimate of where the ball is is the position $(x = 971, y = 840, z = 35)$. A ball has no orientation and the video server always gives it an orientation of 0 radians. Note that the height (z-coordinate) of all objects is fixed if only a single camera view is used. For example, the height of the ball is 35mm and the height of the robot in the next line (below) is 100mm. The best guess for the velocity of the ball is a vector $(dx = -2, dy = 58)$. The example also shows that the robot `b0`

[6] In the case of the ULeague, this will always be 9.

was found at position $(x = 1185, y = 309, z = 100)$. The orientation of the robot is approximately 0 degrees and its motion is given by the vector $(dx = 436, dy = -43)$. Note that since this particular robot is a differential-drive type robot, it is not surprising that the direction of the velocity and the orientation of the robot coincide. The actual velocity of the robot is approximately 43 cm/s. The information for robots `b1` through `y3` is in the same format as described above for robot `b0`.

3 Communication.

The ULeage Communications component consists of two paths. One path goes from each team's client program to the Communication Server. We refer to this as the input, or *read*, path. The second path goes from the Communication Server to the robots, and we refer to this as the output, or *write*, path. We have defined protocols for both paths. The Communication Server contains two threads, one for reading messages from clients and one for writing messages to robots.

Input messages are passed along an Ethernet link, connecting each team's computer to the computer where the Communication Server is running (see figure 1). The Comm Server listens for messages from clients on a socket. The clients send ASCII messages of the form: `[name]:[msg]\n` where `[name]` is the name of the robot (as above, `b0` through `b3` and `y0` through `y3`) and `[msg]` is an 8-bit (one byte) message to be sent to the specified robot, i.e., a number between 0 and 255 (however value 0 is reserved as a NULL command, described below). Thus an example of a complete message would be: `y0:123\n`. The Comm Server maintains an 8-byte command buffer, one byte per robot. Each time an input message is received, the Comm Server updates the byte corresponding to the robot specified in the message.

Output messages are transmitted to robots using a standard Infra-Red (IR) transmitter connected to the computer via a serial or USB port. The output thread writes continuously to the output port, sending messages of the form: `[START][command-buffer][CHKSUM]` where `[START]` is a one-byte start value (255) and `[CHKSUM]` is a stop byte with an included 3-bit checksum (so the values of the checksum will range from 0 to 7). The checksum computation is taken from the Lego networking protocol as implemented in BrickOS[7].

4 Platform.

The ULeague will not use a standard platform like the RoboCup Four-Legged league. However, we have agreed that we need to choose standard platform specifications in order to keep teams on an equal plane regarding costs of the robots. Thus we provide maximum specifications for processor capability, in terms of size (RAM) and speed. This allows the option either to purchase a pre-designed robot kit or to build one from basic components. Several popular robotic kits that are within the range we are currently

[7] BrickOS is a free OS for the Lego Mindstorms RCX. For more details, see `http://brickos.sourceforge.net`.

testing include: *Basic Stamp Board of Education (BOE Bot)*[8], *Handyboard*[9] and *LEGO Mindstorms 2.0*[10].

5 Rules of play.

The field set-up and rules of play are based on those of the Small-size league [1], with adjustments as outlined below.

Field. The field of play is rectangular, 274cm by 152cm in size. The playing surface is heavy green felt or carpet. There are goals at the long ends of the field: 70cm wide by 22cm high by 18cm deep. The field of play is marked with lines and does not have walls (see figure 4). The organizers will place a digital video camera above the field, which will send images to a central vision computer as described in section 2. The organizers will also place IR transmitter(s) around the field, to send messages to all the robots on the field (see section 3). The organizers will also supply a means to transmit the referee's signals to the robots; this will be a keyboard/mouse interface to the communications computer.

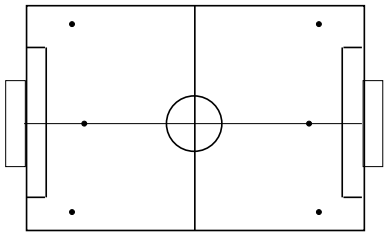

Fig. 4. The ULeague field of play.

Robots and Ball. The ball is a standard orange-colored golf ball. Each team consists of four robots. Teams may designate one of these robots to be a *goalie*, but this is not required. Each robot must fit inside a cylinder 22cm in diameter and 22cm in height. Robots are intended to use wheels or rubber tracks for locomotion. Metal spikes and velcro are specifically prohibited for the purpose of locomotion. Robots do not communicate with each other directly; however, their client programs may communicate with each other through sockets on the team computer. Before a game, each of the two teams has a color assigned, either yellow or blue. The teams change colors and ends of the field for the second half of a match. The organizers will supply circular markers of each color, to be placed on the top of each robot so that they are visible by the overhead camera.

Play. A human referee controls the game. The match lasts two equal periods of 10 minutes, with a half-time interval of 10 minutes. The clock will run at all times, with allowances only made for major stoppages as per the FIFA laws. The game begins with a kick-off by the team that wins the coin toss. The other team kicks off to begin the second half of the game. The ball is out of play when it has completely crossed any

[8] `http://www.parallaxinc.com`
[9] `http://www.handyboard.com`
[10] `http://www.legomindstorms.com`

of the field boundaries. The referee may stop all the robots using the referee interface, retrieve the ball and resume play. A goal is scored when it fully crosses the goal line, between the goal walls. The winning team is the one that scores the greater number of goals during a match.

6 Summary.

We have presented our design for a new undergraduate league within RoboCup, as a means for students who grow too old for RoboCupJunior to stay involved in the initiative and as an entry level for new undergraduates and universities to gain experience with RoboCup soccer. As well, the ULeague gives undergraduates who have tried the RoboCup Simulator League a chance to experiment with physical robots without needing to build a sophisticated and expensive hardware setup. The ULeague architecture consists of a common platform for Vision and Communication. Both will be provided by the league organizers at any competition venues.

There are still several open questions relating to the design of the ULeague. We propose a field without walls, but how will this work in practice? Our proposed communication mechanism has not been tested in a RoboCup competition yet; again, how will this work in practice? The exact restrictions on robot platforms have not been defined yet; what should these be?

The first full exhibition of this league will be held at RoboCupJunior-2003, in Padua, Italy (July 2003). Presently, we have teams from three universities developing the league (University of Manitoba, Canada; Columbia University, USA; Virginia Military Institute, USA). Teams from Australia, Iran and Singapore are also following along with the development. We hope to open the league to any interested parties in 2004. Regularly updated information and a discussion board can be found on our web page: `http://www.robocupjunior.org/uleague`.

7 Acknowledgements.

We gratefully acknowledge the work of the many students involved at the universities of each of the co-authors.

References

1. http://www.itee.uq.edu.au/~wyeth/f180rules.htm.
2. Jacky Baltes. Yuefei: Object orientation and id without additional markers. In *RoboCup-01: Robot Soccer World Cup V*, New York, 2002. Springer.
3. Jacky Baltes and John Anderson. Learning orientation information using neural nets. In *submitted to Robocup-03 Symposium*, 2003.
4. E. Sklar, A. Eguchi, and J. Johnson. RoboCupJunior: learning with educational robotics. In *Proceedings of RoboCup-2002: Robot Soccer World Cup VI*, 2002.
5. Roger Y. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1986.