In *The Eighth Workshop on Humanoid Soccer Robots at Humanoids 2013* (HUMANOIDS13), Atlanta, GA, October 2013.

UT Austin Villa 2013: Advances in Vision, Kinematics, and Strategy

Jacob Menashe, Samuel Barrett, Katie Genter, and Peter Stone
Department of Computer Science
The University of Texas at Austin
{jmenashe, sbarrett, katie, pstone}@cs.utexas.edu

***Abstract*— In RoboCup, although the fields are standardized and color coded, the area outside the fields often contains many objects of various colors. Sometimes objects off the field may look very similar to balls, robots, or other objects normally found on the soccer field. Robots must detect all of these objects, and then differentiate between the true positives and false positives. This paper presents a new method using Gaussian fitness scores to differentiate between true positives and false positives for balls, robots, and penalty crosses. We also present some other improvements in our code base following our 2012 championship, such as our usage of a virtual base for forward kinematics calculations, our ability to flexibly transition player roles given dynamic numbers of teammates, and our ability to quickly integrate new kicks of varying speeds into our strategy. With these improvements, our UT Austin Villa team finished third in the Standard Platform League at RoboCup 2013.**

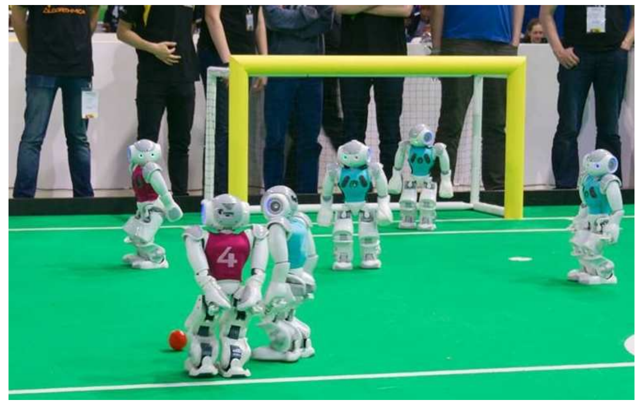

Fig. 1: 6 Naos on the 2013 RoboCup SPL field.

I. Introduction

RoboCup, or the Robot Soccer World Cup, is an international research initiative that works to advance robotics and artificial intelligence by using the game of soccer as a test domain. The long-term goal of RoboCup is to build and program a team of 11 humanoid robot soccer players that can beat the best human soccer team on a real soccer field by the year 2050 [1].

RoboCup is organized into several soccer leagues, including both simulation leagues and leagues that compete with physical robots. Our team, UT Austin Villa[1], competes in the Standard Platform League (SPL)[2]. The SPL uses teams of identical Aldebaran Nao humanoid robots[3], making it essentially a software competition. In the SPL, each team competes with up to 5 Nao robots. Teams compete on a 9 by 6 meter field, with two identical yellow goals and white tape marking the lines. See Figure 1 for an example of the robots and field setup. UT Austin Villa has competed in the SPL with the Nao robots every year since the Nao was introduced in 2008. Over these years, we have built a substantial code infrastructure for robot soccer that served as the base for our championship in 2012 [2], [3].

Many objects surround a RoboCup field, many of which may look similar to a ball or an opponent. For example, little children wearing orange shirts near the field may look like orange balls, while maroon flower pots scattered around the field may resemble maroon opponents. Hence, it is critical for a robot to be able to evaluate possible detections and accurately determine which instances are true positives. In this paper, we present some of the major improvements in our code base that were utilized at RoboCup 2013. Specifically, we present our usage of Gaussian fitness scores to evaluate possible detected objects. In the past, our team used a series of binary cutoffs for sanity checks on each object measurement to determine which detected objects are true positives. In this work, we present a better method in which, for each detected object, we simultaneously evaluate a variety of object measurements to determine if the detected instance is indeed the object of interest (a "true" positive). Additionally, we present our usage of a virtual base for forward kinematics calculations. We also discuss our ability to flexibly transition player roles given dynamic numbers of teammates and our ability to quickly integrate new kicks of varying speeds into our strategy.

A video highlighting our performance at RoboCup 2013 can be found at `http://goo.gl/fcnO9V`.

II. Related Work

A. Object Detection

We researched a number of alternative methods for object detection in RoboCup when designing our approach. The B-Human team uses a sequential sanity checking method for ball detection[4], similar to what we developed in [2]. The Dutch Nao Team similarly uses blob detection as an underlying mechanism for object detection, and use width- and height-based sanity checking for distinguishing goals from non-goals [5]. rUNSWift's approach differs from the blob detection methods in that they use feature descriptors and a

[1] http://www.cs.utexas.edu/~AustinVilla/
[2] http://www.tzi.de/spl/
[3] http://www.aldebaran.com/

modified ICP algorithm to map field objects to expectations [6]. To our knowledge, we are the only team using Gaussian fitness computations for simultaneous feature evaluations on detected object candidates.

B. Virtual Base

We found little discussion on the topic of coordinate frame selection in RoboCup literature. B-Human uses a torso-centered coordinate frame that is rotated parallel with the ground plane [4]. This coordinate frame can be obtained by translating the virtual base coordinate frame along the vector $(0, 0, h)^{\intercal}$, where h is the torso height. In other words, the virtual base is B-Human's `TorsoMatrix` projected onto the ground. B-Human's `TorsoMatrix` and our virtual base are analogous to the `base_link` and `base_footprint` coordinate frames found in ROS [7], respectively.

C. Transitioning Player Roles

Our dynamic transitioning of player roles such that key roles were filled quickly and efficiently was motivated by previous work completed by the UT Austin Villa 3D simulation team[8]. In their work they found a valid role assignment function that minimized longest distance from each player to it's target location, avoided collisions, and was dynamically consistent. However, in our work we focus more on ensuring players filling critical roles can arrive in their new location quickly. Additionally, we do not account for potential collisions when deciding which players should fill which roles, as our estimates of where teammates and opponents are on the field are imperfect.

III. Utilizing Gaussian Fitness Scores for Object Detection

This year we introduced a significant improvement to our object detection system [2] in the form of Gaussian fitness estimates based on perceived object statistics. Our method involves empirically determining a mean feature vector for the target object based on a number of quantifiable measurements. We then evaluate potential detections with respect to this mean and determine a confidence score for each detection. This method served as our primary sanity checking mechanism for ball, robot, and penalty cross detection. The sequential sanity checking implementations described in [2] proved adequate for field lines and goals, so these were not prioritized for conversion to the Gaussian fitness checking method.

A. Measured Statistics

To illustrate our method we present a high-level overview of our detection algorithm with the statistics measured for each object. We begin by classifying the image and forming blobs as described in [2]. Algorithm 1 describes the general detection process. Our method is concerned with the implementation of `getFitnessValues`.

For orange blobs (i.e. potential ball detections), we compute the following:

- Orange percentage within the ball

Algorithm 1 Detection process for an arbitrary object.

```
procedure DetectObject(o)
    I ← getCameraImage()
    C ← classifyImageColors(I)
    B^o ← getBlobsByColor(C, getColor(o))
    F^o ← getFitnessValues(B^o)
    b, f ← arg max_{b∈B^o, f∈F^o} F^o

    if f > θ then
        return b
    end if
    return nil
end procedure
```

- Green or white percentage below the ball
- Circle deviation
- World height (i.e. the Z coordinate in the world frame)
- Discrepancy between width-based and kinematics-based distance estimates
- Distance from field edges along the ground plane (0 within the field)
- Velocity between frames

Note that width-based distances are determined with trigonometry based on the detected and known ball width, and kinematics-based distances use image coordinates and the camera matrix to project pixels onto the ground plane.

For each of these measurements we empirically determine a mean and one-dimensional standard deviation by examining individual measurements from known detections. We compile the means into a single feature vector, and use the standard deviations to define a diagonal covariance matrix. Some measurements, such as orange percentage, are bounded by a particular range and thus may not exhibit truly Gaussian fluctuations about a mean. Analysis of past readings may show an average orange percentage of 90%, but it is clear that 100% is optimal and we should not be penalizing blobs for having all orange pixels. In cases where the measurement range is bounded, we select the optimal value as the mean. For the orange percentage measurement, this implies that we set the mean to 100%.

For completeness, we also present the statistics used for robots:

- Width over height
- Height over width
- (Kinematics-based distance)/(width-based distance)
- (Kinematics-based distance)/(height-based distance)
- Percentage of correct jersey color found in the jersey blob
- Percentage of green/white/grey pixels found below the jersey
- Percentage of jersey color/green/white/grey pixels found along the whole robot
- Chest height

As well as for the penalty cross:

- Green percentage in each octant around the cross
- Distance from known cross location
- Cross height
- Cross width

Once we have computed a set of measurements about a particular blob, we organize these measurements into a vector which is then analyzed with a multivariate Gaussian distribution to produce the fitness computation.

B. Fitness Computation

To compute the fitness P for a particular detection, we use the standard multivariate Gaussian PDF G with mean μ and covariance Σ, weighted such that $P(\mu, \mu, \Sigma) = 1.0$.

$$P(x, \mu, \Sigma) = \frac{G(x, \mu, \Sigma)}{G(\mu, \mu, \Sigma)} \quad (1)$$

This method allows us to simultaneously evaluate a variety of object measurements in a manner similar to a support vector machine (SVM). Rather than identify a series of binary cutoffs for sanity checks, we essentially construct a hyperplane for binary classification. In practice, this method was comparable in effectiveness to our earlier approaches of hand-tuning sequential sanity checks with a handful of added advantages, including the ability to discern between competing candidates and the ability to process measurements in parallel. In circumstances where a detection lies near a threshold for one particular measurement, our method is still able to recover the object of interest consistently based on the other measurements being considered. In a sequential checking system, it is far more likely for a single bad measurement to throw off an entire detection.

More sophisticated classification techniques such as SVM were considered as well, but our method was chosen based on the following advantages:

- Mean and standard deviation values can be estimated and adjusted with ease, without the need for training.
- Feature vectors can be understood and debugged by a human.
- Computations are fast and can be carried out on a large number of candidates each frame.

The output of our method is naturally a real number in the range [0,1]. Currently we use a cutoff of 0.3 to distinguish a true positive from a false positive, corresponding to approximately 1.5 standard deviations in the univariate case. Our future work includes making better use of these confidence values, either with individual object filters or in the overall localization system.

IV. Virtual Base for Forward Kinematics Calculations

A standard implementation concern on multi-pedal robot systems is the choice of a consistent egocentric coordinate frame for describing world objects and body parts. A good choice of coordinate frame will allow the programmer of such a system to intuitively understand the spatial relationships between these objects in the environment. For humanoid robots, a simple solution is to use the point in the torso at which all kinematic chains between extremities coincide.

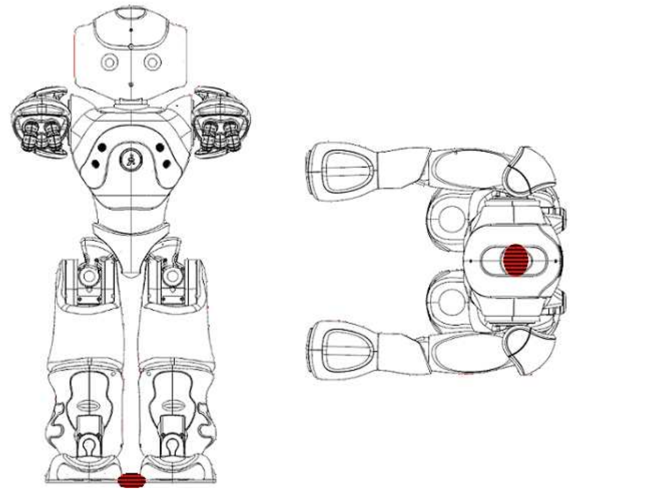

Fig. 2: Front and top views to demonstrate the location of the virtual base, by the striped ellipses between the feet (front view) and in the middle of the head (top view).

In RoboCup, the torso-centered coordinate frame can be problematic. The field is flat, and objects of interest are almost always on the ground. Any intuitive coordinate frame should therefore have no X or Y rotation and should be centered at some point on the ground plane. This ensures that world objects on the field can be represented with a z-coordinate of 0. Additionally, a robot's egocentric frame should remain at some location that is central to the robot, allowing for symmetric definitions when designing body movements. For example, stepping forward with the left foot might require a relative foot position of $(x, y, 0)$, which would imply that stepping forward with the right would require a relative foot position of $(x, -y, 0)$.

No body part exists on a humanoid that meets these constraints. The centered body parts of the robot, such as the torso and head, are all elevated, with their ground offsets shifting as the robot moves. Likewise the ground-level body parts, i.e. the feet, are not in the center. We therefore chose to construct a virtual body part to meet these criteria, which we call the *virtual base*.

Intuitively, the virtual base is the point on the field directly below the torso with no X or Y rotations relative to the ground plane, and no Z rotation relative to the torso. Figure 2 shows the top and side views of the virtual base location for a particular pose.

The base transformation matrix is constructed and applied to other body parts as described in Algorithm 2, using pre-calculated transformation matrices in the torso-centered coordinate frame. The stance foot is taken as the foot with the greatest sum of pressure readings, on the assumption that the foot that the robot is standing on will experience greater pressure than the other foot. While the Nao pressure sensors aren't accurate enough for precise calculations, this sort of binary decision is quite accurate in practice. When the pressure sensors yield similar values, this indicates that both feet are flat on the ground. In this case, the position and rotation of the virtual base is the same regardless of the stance foot that is selected.

Algorithm 2 Construction and application of the virtual base. Once the virtual base is computed in the torso coordinate frame, all other body parts are converted to the virtual base frame and placed in B^v.

Compute t^f, the torso in the foot frame:
$f^t \leftarrow$ stance foot in torso frame
$t^f \leftarrow (f^t)^{-1} =$ torso in foot frame

Compute v^f, the virtual base in the foot frame:
$v^f \leftarrow$ identity transform
$(v^f_x, v^f_y, v^f_z) \leftarrow (t^f_x, t^f_y, 0)$
$v^f \leftarrow \texttt{rotateZ}(v^f, \texttt{getAngleZ}(t^f))$

Compute v^t, the virtual base in the torso frame:
$v^t \leftarrow \texttt{globalToRelative}(v^f, t^f)$

Compute all body parts in the virtual base frame:
$B^t \leftarrow$ Body parts in torso frame
$B^v \leftarrow \emptyset$
for all $b^t \in \{$ Body Parts $\}$ **do**
  $b^v \leftarrow \texttt{globalToRelative}(b^t, v^t)$
  $B^v \leftarrow B^v \cup \{b^v\}$
end for
return B^v

This coordinate frame is used throughout the codebase for visual object positions, body part transforms, and body motions. This choice of coordinate frame is intuitive for testing and debugging, allows for simple conversions between local and global coordinates, and ensures that coordinates are consistent over changes in stance.

V. Behavioral Improvements

In addition to the improvements discussed in Sections III and IV, UT Austin Villa made some other improvements between RoboCup 2012 and RoboCup 2013 that were also apparent in games. Most notable was our ability to flexibly transition player roles given dynamic numbers of teammates and our ability to quickly integrate new kicks with varying speeds into our strategy.

A. Transitioning Player Roles Based on the Number of Active Players

Given that teams played with five players at RoboCup 2013, as opposed to the four players used in 2012, the ability to coordinate players became even more important in 2013. In addition, given the number of robots that are removed from the field due to penalties and failing hardware, it is vital that the team be able to adapt to a variety of players being on the field. UT Austin Villa had planned about strategies for these settings in past years, but this year we introduced code that allowed us to quickly adapt the strategy of the team during the competition itself.

The strategy works by having each robot communicate both its position as well as a bid on being the chaser whenever it sends messages to its teammates. This bid is calculated based mainly on the robot's distance from the ball, its angle to the ball, and whether chasing the ball would require going downfield or upfield. The robot with the best bid becomes the chaser and tries to approach and kick the ball according to the kicking strategy described in Section V-B. The remaining players are assigned to the remaining roles, such as playing as a forward, defender, or midfielder. Each of these roles is specified using parameters such as the desired positions relative to the ball, maximum distances to be traveled upfield, and how the robot can position to prevent open shots at its own goal.

If fewer than five robots are on the field, roles are filled in order of priorities that are assigned to each position. Given the roles that are currently active, robots are assigned to these roles based on their distances from the roles' desired locations. As many of these role positions are symmetric with respect to the side direction, the roles are largely assigned based on the difference between the robot's location and the role's desired location along the dominant field dimension.

To adapt more fully to the current game situation, the priorities and locations of the roles can be adjusted based on the region in which the ball is located. In addition, these formations can be adjusted based on the current game score as well as the time remaining in the game. All role locations and priorities can be quickly adjusted via configuration files.

In addition to these changes, the strategy was improved to more fully reason about passing and set plays. While the UT Austin Villa team previously positioned robots in order to receive passes and preferred passing to teammates, the robots did not communicate their pass intentions. One improvement introduced by the 2013 team was to communicate when a pass was about to occur, which allowed the receiving player to adapt to the expected destination of the pass. Furthermore, more set plays for the kick off were introduced, and the robots autonomously chose which set play to use based on the team's locations as well as the location of the opponents. Similarly to the role configurations, these set plays were easily specified via configuration files. As discussed in [3], the simulation tool developed by the UT Austin Villa team proved invaluable to testing the various formations and set plays that were used in competition.

B. Quickly Integrating Kicks of Varying Speeds

UT Austin Villa has a kick engine that can kick various distances in addition to several kicks that can be executed directly from the walk engine. In addition, UT Austin Villa also developed a longer kick based on one designed by Northern Bites [9] in the 2012 competition. This kick was added to adapt to the larger 9m by 6m field that was introduced for the 2013 competition. All of these kicks require different amounts of execution time, go different distances, and travel within different accuracy ranges of their desired direction. These kicks can all be integrated by planning about the destination of the kick and how the team

Fig. 3: Setup for the baseline and velocity experiments.

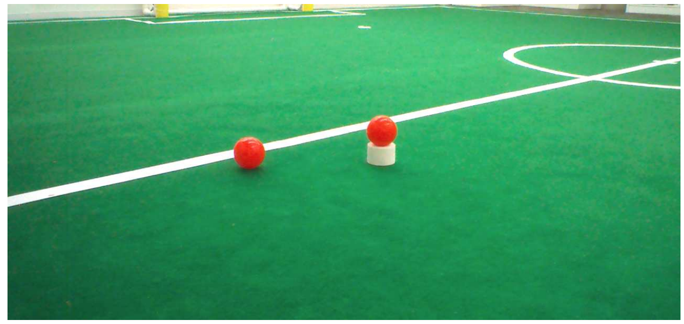

Fig. 4: Setup for the height experiment.

would like to move the ball as described in [10]. In addition, we introduced some reasoning to avoid making slower kicks when opponents were detected within a specified distance and angle to the kicker. This adjustment served to avoid having shots blocked by opponents walking into the desired path of the kick while the kick is executing. The ease of adapting the strategy to the lengths of kicks proved especially useful due to the differences in the kick distances at different venues.

VI. Evaluation

Our method of computing fitness scores in object detection provides a useful improvement over binary accept/reject approaches. Here we show the ability of the ball detector to consistently and significantly differentiate and rank its detections. We present a variety of experiments involving an object of interest (the left ball) and a decoy (the right ball). These experiments show that the ball detector is able to use computed fitness values to effectively rank and discern between the two balls.

A. Baseline

We begin with our baseline in which two nearly identical ball detections are found side-by-side, as shown in Figure 3. Ideally, two such detections would yield identical fitness values. We sampled fitness computations from 500 frames and obtained mean values of .92 and .89 with standard deviations of .04 and .02 for the left and right balls, respectively. Assuming Gaussian distributions, this implies a 73% chance of picking the left ball over the right. This bias can be explained with subtle differences in lighting conditions. The following experiments exhibit more pronounced biases that are the result of significant differences in fitness values.

B. Velocity

To demonstrate the effectiveness of detection ranking, we begin by looking at velocity measurements. Accounting for perceived velocity allows the ball detector to avoid switching randomly between ball candidates when there are multiple valid detections in view. We use the same setup as in Figure 3, however in this case we include velocity measurements as part of the fitness computation. By setting the expected velocity to 0, we can essentially tell the detector to prefer choosing one ball consistently rather than switching back and forth. After ranking detections based on the first frame of data, the detector settles on the left ball, which biases all subsequent detections. As we see in Table 1, the chance of switching back to the right ball is significantly reduced. It is worth noting that this distinction can be arbitrary - if the detector were to have selected the right ball first, then that would bias future estimates toward the right ball. In the setting of a RoboCup game, our teammates share estimates of the ball location. The effect of this is that the ball selected by the team is reinforced as the true ball, reducing the possibility that a ball off the field might be chosen by a particular robot.

C. Height

In this experiment we place an object under the right ball to increase its height, which in turn increases the discrepancy between width-based distance and kinematics-based distance calculations. This is demonstrated in Figure 4. Velocity measurements were disabled for this experiment to remove the bias toward the ball that was previously selected. Table 1 again shows a strong preference for the ball that is on the ground.

D. Size

This experiment uses a larger soccer ball as a decoy, as shown in Figure 5. This experiment works similarly to the previous one in that the ball distance estimates are thrown off, in addition to worsening "orangeness" measurements due to the blackened areas of the ball. As we see in Table 1, the left ball is strongly preferred as expected, however the soccer ball's fitness value is still high enough that it could be detected as the object of interest in the absence of a better candidate. By allowing for candidate ranking, we're able to ensure a negligible chance of picking the soccer ball over the actual ball.

E. Competition Overview

After placing first at RoboCup 2012 in the SPL, UT Austin Villa finished third at RoboCup 2013 in Eindhoven, The Netherlands. 22 teams entered the 2012 competition, where the tournament consisted of two round robin rounds, followed by an elimination tournament with the top 8 teams. The first round consisted of a round robin with seven groups of three teams each, with the top teams from each group advancing. In the second round, there were four groups of

Fig. 5: Setup for the size experiment.

Experiment	μ, σ (Left)	μ, σ (Right)	P_G	P_S
Baseline	.92, .04	.89, .02	.7291	0.68
Velocity	.88, .12	.15, .20	.9992	0.00
Height	.90, .02	.78, .04	.9965	0.87
Size	.89, .04	.39, .01	$>$ .9999	1.00

TABLE 1: Statistics for computed fitness values for the given experiments. Each experiment uses a sample of 500 frames. P_G represents the probability that the new Gaussian method will select the object of interest (i.e. the left ball). This calculation assumes all fitness values are normally distributed. P_S represents the probability of the same event using the original sequential checking method in the same experimental setup.

four teams each, with the top two teams from each group advancing. From the quarterfinals on, the winner of each game advanced to the next round.

All of UT Austin Villa's scores are shown in Table 2 and discussed shortly below. We document the competition results in this paper as an informal evaluation of the team as a whole.

At RoboCup 2013, UT Austin Villa began by winning the first round robin after defeating Berlin United 4:0 and Cerberus 5:0. In these games we tested and tuned various strategies. UT Austin Villa faltered in the first game in the second round robin, losing 2:4 to SPQR. As we discussed in Section V, we particularly struggled in this game because both teams played the entire game without inner-team communication due to issues with the field's wireless router. UT Austin Villa still emerged second in this second round robin pool though, after beating rUNSWift 2:1 and TJArk 6:0. UT Austin Villa faced Northern Bites in the quarter-finals, capturing a 7:0 win. UT Austin Villa then lost to B-Human 0:8 in the semi-finals before beating rUNSWift 4:0 in the 3rd place game.

Round	Opponent	Score
Round Robin 1	Berlin United	4-0
Round Robin 1	Cerberus	5-0
Round Robin 2	SPQR	2-4
Round Robin 2	rUNSWift	2-1
Round Robin 2	TJArk	6-0
Quarterfinal	Northern Bites	7-0
Semifinal	B-Human	0-8
3rd Place	rUNSWift	4-0

TABLE 2: RoboCup 2013 Results

VII. Conclusion

This paper introduces our usage of Gaussian fitness scores to evaluate possible detected objects. For each detected object, we simultaneously evaluate a variety of object measurements to determine if the detected instance is a true positive. Using this method, instead of a series of binary cutoffs for sanity checks, allows us to essentially construct a hyperplane for binary classification. We found that using this method was similar in effectiveness to hand-tuned sequential sanity checks, but provided the additional benefit of simultaneous measurement evaluation. In cases where a detected object lies near a threshold for a particular sanity check, using Gaussian fitness scores allows us to still recover true positives based on the other measurements considered. Hence, using fitness scores makes it less likely that one bad measurement will falsely eliminate true positives.

We also introduce other improvements that were made in our code base for RoboCup 2013, including our usage of a virtual base for forward kinematics calculations, our ability to flexibly transition player roles given dynamic numbers of teammates, and our ability to quickly integrate new kicks of varying speeds into our strategy. Combining all of our improvements helped us play cohesively and intelligently at RoboCup 2013, allowing us to finish third in the SPL.

References

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "RoboCup: The robot world cup initiative," in *Proceedings of The First International Conference on Autonomous Agents*. ACM Press, 1997.

[2] S. Barrett, K. Genter, T. Hester, P. Khandelwal, M. Quinlan, P. Stone, and M. Sridharan, "Austin Villa 2011: Sharing is caring: Better awareness through information sharing," The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-12-01, January 2012.

[3] S. Barrett, K. Genter, Y. He, T. Hester, P. Khandelwal, J. Menashe, and P. Stone, "UT Austin Villa 2012: Standard Platform League world champions," in *RoboCup 2012: Robot Soccer World Cup XVI*, X. Chen, P. Stone, L. E. Sucar, and T. V. der Zant, Eds. Springer Verlag, 2012.

[4] T. Röfer, T. Laue, J. Müller, A. Fabisch, F. Feldpausch, K. Gillmann, C. Graf, T. J. de Haas, A. Härtl, A. Humann, D. Honsel, P. Kastner, T. Kastner, C. Könemann, B. Markowsky, O. J. L. Riemann, and F. Wenk, "B-Human team report and code release," 2011, http://www.b-human.de/downloads/bhuman11_coderelease.pdf.

[5] C. Verschoor, A. Wiggers, D. ten Velthuis, A. Keune, M. Cabot, S. Nugteren, E. van Egmond, H. van der Molen, R. Iepsma, M. van Bellen, M. de Groot, E. Fodor, R. Rozeboom, and A. Visser, "Dutch nao team - technical report," 2011.

[6] S. Harris, P. Anderson, B. Teh, Y. Hunter, R. Liu, B. Hengst, R. Roy, S. Li, and C. Chatfield, "Robocup standard platform league - runswift 2012 innovations," in *Australasian Conference on Robotics and Automation*, 2012.

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[8] P. MacAlpine, F. Barrera, and P. Stone, "Positioning to win: A dynamic role assignment and formation positioning system," in *RoboCup-2012: Robot Soccer World Cup XVI*, ser. Lecture Notes in Artificial Intelligence, X. Chen, P. Stone, L. E. Sucar, and T. V. der Zant, Eds. Berlin: Springer Verlag, 2013.

[9] O. Neamtu, W. Dawson, E. Googins, B. Jacobel, L. Mamantov, D. McAvoy, B. Mende, N. Merritt, E. Ratner, N. Terman, J. Zalinger, J. Morrison, and E. Chown, "Northern Bites code release," 2012, https://github.com/northern-bites.

[10] S. Barrett, K. Genter, T. Hester, M. Quinlan, and P. Stone, "Controlled kicking under uncertainty," in *The Fifth Workshop on Humanoid Soccer Robots at Humanoids 2010*, December 2010.