To appear in *Proceedings of the RoboCup International Symposium 2012* (RoboCup 2012), Mexico City, Mexico, June 2012.

UT Austin Villa: RoboCup 2012 3D Simulation League Champion

Patrick MacAlpine, Nick Collins, Adrian Lopez-Mobilia, and Peter Stone

Department of Computer Science, The University of Texas at Austin
{patmac,no1uno,alomo01,pstone}@cs.utexas.edu

Abstract. The UT Austin Villa team, from the University of Texas at Austin, won the RoboCup 3D Simulation League in 2012 having also won the competition the previous year. This paper describes the changes and improvements made to the team between 2011 and 2012 that allowed it to repeat as champions.

1 Introduction

UT Austin Villa won last year's 2011 RoboCup 3D simulation competition in convincing fashion by winning all 24 games it played. During the course of the competition the team scored 136 goals and conceded none. This was a vast improvement over the team's previous performance in 2010 when the team finished just outside the top eight. However, despite further improvements for the 2012 competition, the results were much closer this year due to the vast improvement of other teams in the competition.

While many of the components of the 2011 UT Austin Villa agent were reused for the 2012 competition, including that of an optimized omnidirectional walk [1] which was the crucial component in winning the 2011 competition, a number of upgrades were made to the agent to maintain its performance relative to the improvement other teams made between the 2011 and 2012 competitions. Additionally, changes in the rules and format of the 2012 competition, particularly increases in field size and the number of players on a team, necessitated other modifications to the agent be made. This paper is not an attempt at a complete description of the 2012 UT Austin Villa agent, the foundation of which is the same as the 2011 agent fully described in a team technical report [2], but instead focuses on changes made in 2012 that helped the team repeat as champions.

The remainder of the paper is organized as follows. In Section 2 a description of the 3D simulation domain is given highlighting differences from the previous year's competition. Section 3 discusses how a hand-coded get up routine was optimized to make it faster. Section 4 describes an updated kicking system. Changes to a formation and positioning system needed for scaling to 11 agents on a team are detailed in Section 5. Results of the tournament and analysis of improvements to the agent are given in Section 6, and Section 7 concludes.

2 Domain Description

The RoboCup 3D simulation environment is based on SimSpark,[1] a generic physical multiagent system simulator. SimSpark uses the Open Dynamics Engine[2] (ODE) library for its realistic simulation of rigid body dynamics with collision detection and friction. ODE also provides support for the modeling of advanced motorized hinge joints used in the humanoid agents.

The robot agents in the simulation are homogeneous and are modeled after the Aldebaran Nao robot,[3] which has a height of about 57 cm, and a mass of 4.5 kg. The agents interact with the simulator by sending torque commands and receiving perceptual information. Each robot has 22 degrees of freedom: six in each leg, four in each arm, and two in the neck. In order to monitor and control its hinge joints, an agent is equipped with joint perceptors and effectors. Joint perceptors provide the agent with noise-free angular measurements every simulation cycle (20 ms), while joint effectors allow the agent to specify the torque and direction in which to move a joint. Although there is no intentional noise in actuation, there is slight actuation noise that results from approximations in the physics engine and the need to constrain computations to be performed in real-time. Visual information about the environment is given to an agent every third simulation cycle (60 ms) through noisy measurements of the distance and angle to objects within a restricted vision cone (120°). Agents are also outfitted with noisy accelerometer and gyroscope perceptors, as well as force resistance perceptors on the sole of each foot. Additionally, agents can communicate with each other every other simulation cycle (40 ms) by sending 20 byte messages.

For the 2012 competition games consisted of 11 versus 11 agents (up from 9 versus 9 agents in 2011). The field size was also increased to be 20 meters in width by 30 meters in length (the 2011 competition was played on a field 14 meters in width and 21 meters in length).

3 Optimization of the Get Up Routine

A vital skill for an agent in the 3D simulation competition is the ability to stand up from a prone position after having fallen over. In order to get up from a prone position, the robot must choose certain joint angles at certain times to control the movement of its body. The UT Austin Villa team devised a get up routine which iterates through a series of poses, transitioning from one pose to another after a pre-specified period of time. The poses are determined by a series of specified joint angles. Thus, the get up routine can be numerically parameterized by a sequence of time intervals and a set of joint angles. For the 2011 competition these values were chosen and hand-tuned manually.

How quickly a robot is able to recover from a fall is important so that it can rejoin play as fast as possible. For the 2012 competition parameters for the

[1] `http://simspark.sourceforge.net/`
[2] `http://www.ode.org/`
[3] `http://www.aldebaran-robotics.com/eng/`

get up routine were optimized through machine learning to decrease the time needed to stand up. The following subsections explain how this was done.

3.1 Fall Detection and Get Up Motion

The robot detects that it is not upright when its accelerometers indicate that the force of gravity is pulling in a direction not parallel to its torso. When this happens, the robot spreads its arms outward to its side at 90 degree angles. This way, when the robot lands it will fall on either its back or its front. After extending its arms to make sure it falls on its front or its back, the robot pauses for 0.7 seconds before determining which way it has fallen. The agent then proceeds to enter one of two get up routines depending on whether it is lying on its back or front. The get up routine used by a robot lying on its back iterates through the series of poses shown in Figure 1.

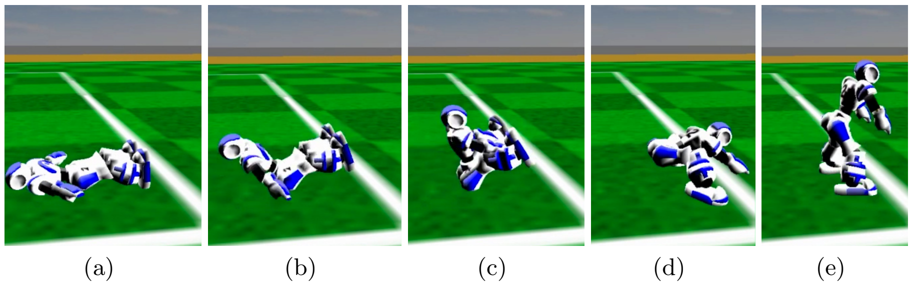


Fig. 1. Routine for getting up after falling backwards. The robot begins lying on its back (a) and then propels itself up with its arms (b). Next the robot throws its arms forward and contracts its legs to get its center of mass in front of its feet (c). Using momentum from the initial push the robot manages to roll into a squatting position (d) after which the robot can get up by extending its knees and hips (e).

3.2 Optimization Process

Parameters for the robot's get up were optimized using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm [3]. CMA-ES was chosen after previously finding it to be the most successful algorithm in optimizing parameters for similar robot skills such as walking forward and turning [4]. CMA-ES is a policy search algorithm that successively generates and evaluates sets of candidates sampled from a multivariate Gaussian distribution. Once CMA-ES generates a group of candidates, each candidate is evaluated with respect to a *fitness* measure. When all the candidates in the group are evaluated, the mean of the multivariate Gaussian distribution is recalculated as a weighted average of the candidates with the highest fitnesses. The covariance matrix of the distribution is also updated to bias the generation of the next set of candidates toward directions of previously successful search steps.

In order to optimize the parameters of the robot's get up routine, the robot is forced to fall and then the time it takes for the robot to get up is measured. A robot that is capable of standing can easily be given a continuous, meaningful fitness based on how long it takes to get up and its stability once standing (as measured by its likelihood of falling back down). Each evaluation begins with the robot being set into an upright, neutral stance by the simulator. The robot is given 1 second to make sure it is stable, and then it is forced to fall backwards. The robot uses accelerometer readings to determine whether it is in an upright position. If it detects that it is not, it will set an internal hasFallen flag and then begin its get up routine. Once the get up routine completes, the agent checks to see if it is upright, and if it is, it clears the hasFallen flag. Otherwise it continues trying its get up routine.

During the evaluation, the robot records how much time the hasFallen flag spends being true, and its fitness for the run is the negation of that. So if it falls once, gets up after 2.5 seconds, and stays up, its fitness is -2.5. After forcing the robot to fall, the evaluation runs for 4 seconds and then ends. If the robot gets up in 2 seconds, but is unstable and falls back down after being up for a second, then its fitness will be -3, since the total time it spent falling or getting up was 3 seconds. Punishing subsequent falls serves to ensure the get up routine is stable.

Additionally, in order to make sure it is stable enough to walk, the robot is asked to perform a movement action after getting up. A complete evaluation trial consists of seven falls and subsequent get ups where after each get up the robot does one of the following: walks forwards, walks backwards, walks left, walks right, turns left, turns right, or stands still. The average across all seven evaluations gives the fitness score for a trial.

3.3 Optimization Results

The optimization process discussed in Section 3.2 was performed on the routines for both the robot getting up from its front and back. Each optimization was run across 200 generations of CMA-ES, using a population size of 150, and was seeded with the original get up sequence hand-tuned parameter values (consisting of joint angle positions and time intervals between poses). Information about the number of parameters optimized, as well as the improvement in speed after optimization are shown in Table 1. Both optimizations were able to reduce the time required to get up to almost a third of their original hand-tuned times.

Optimization	Parameters	Hand-tuned Time (s)	Optimized Time (s)
Get Up from Front	9	2.62	0.96
Get Up from Back	26	2.20	0.84

Table 1. Get up optimizations with the number of parameters optimized and the time in seconds taken to get up before and after optimization.

4 Kicking

For the 2012 competition UT Austin Villa switched from a kicking system using directional inverse kinematics based kicks [5] to a hybrid system that has both fixed pose keyframe (better for distance) and inverse kinematics (more robust) based kicks. The parameters for all kicks were optimized using the CMA-ES algorithm described in Section 3.2. The following subsections detail the design and optimization of the kicking system.

4.1 Fixed Pose Keyframe Kicks

Fixed pose keyframe kicks consist of a sequence of body positions, defined by different joint angle positions, which the agent proceeds through in order to kick the ball. Three such kicks were used in the 2012 competition: KickLong, KickMedium, and KickQuick. For each of these the agent first places its support (non-kicking) leg near the ball and shift its weight to the support leg. Next it lifts its kicking leg, and pulls it backward, before finally swinging its kicking leg forward to strike the ball.

KickLong kicks the ball the farthest of the three fixed pose keyframe kicks and is only used on kickoffs. This kick's primary purpose is to push the ball as far as possible into the opponent's end of the field on a kickoff. KickLong also kicks the ball high in the air such that opposing agents can not block the kick as the ball travels over their heads. The extreme motion used by the agent to propel the ball causes the agent to fall flat on its back at the end of the kick.

KickMedium is a kick that also gets a lot of distance, but allows for the agent to remain stable (not fall over) at the end of the kick. KickMedium is used for free kicks as well as during regular play. The kick takes over two seconds to get off thus requiring opponents be at least 2.5 meters away before starting the kick.

Although it doesn't get as much distance as KickMedium, KickQuick is much faster to get off as it takes less than a second to make contact with the ball. KickQuick is designed for quickly kicking the ball when opponents are closing in, and on average gets enough height on the ball to chip it over approaching opponents' heads. KickQuick only requires that the closest opponent be at least 1.0 meters away before starting the kick. Like KickLong, KickQuick destabilizes the agent and causes it to fall over at the conclusion of the kick.

More information about the fixed pose keyframe kicks are found in Table 2.

4.2 Inverse Kinematics Based Kicks

A weakness of the fixed pose keyframe kicks in Section 4.1 is that they require very precise positioning relative to the ball (discussed in Section 4.3) in order for them to be executed. An alternative to this is to define a path relative to the ball that the robot's foot should follow during a kick, and then use inverse kinematics to move the foot along this path. The main advantage gained through such an approach is that a kick is able to adapt to the position of the ball and thus does not require as precise positioning by an agent to line up the kick.

As described in [5], the UT Austin Villa team constructs an inverse kinematics based kick (KickIK) by specifying waypoints relative to the ball for the foot to travel through, and then interpolates between these points using a Cubic Hermite Spline curve to determine the trajectory of the foot's path during a kick. Figure 2 shows the relative waypoints for a forward kick. Inverse kinematics for the agent are computed using OpenRAVE's [6] kinematics solver.

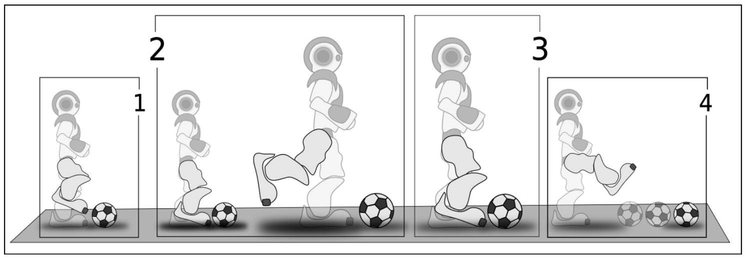


Fig. 2. Waypoints relative to the ball that define the path of the foot for an inverse kinematics based kick. (1) Lift leg to center behind ball. (2) Pull leg back from ball. (3) Bring leg back to position of ball. (4) Kick through ball.

4.3 Kick Positioning

Outside of the kicking motion itself, how a robot positions itself in proximity to the ball before executing a kick is probably the most critical component to successfully striking the ball. When lining up to kick the ball, the UT Austin Villa agent first approaches a target position behind the ball to kick from. The agent is not allowed to proceed with the kick until it is within certain distance thresholds of this target position both along the vectors perpendicular and parallel to the ball from the target position. Before executing a kick the agent must also be within a set angular threshold of facing toward the ball.

Using distance and angle thresholds when positioning to kick is a change from UT Austin Villa's 2011 inverse kinematics based kicking system's positioning [5]. The 2011 agent's kick was triggered as soon as inverse kinematics calculations determined the robot's foot could reach all necessary points along a curve to kick through the ball. After the 2011 competition it was found that using inverse kinematics calculations as a trigger for when to kick is problematic for a moving robot. This is due to the robot's momentum causing its body's position and orientation relative to the ball to change right after deciding to kick. These changes in position and orientation, although often quite small, are enough to prevent the robot's foot from being able to reach the ball and force the robot to reposition itself after aborting the kick.

4.4 Optimization Process

The CMA-ES algorithm discussed in Section 3.2 is used to optimize joint angles for the fixed pose keyframe kicks mentioned in Section 4.1, as well as the X, Y, and Z positions of the waypoints defining a curve for an inverse kinematics based kick detailed in Section 4.2. Roll, pitch, and yaw positions of the foot are also optimized for each of the waypoints of an inverse kinematics based kick. Additionally, for all kicks, the positioning parameters discussed in Section 4.3 of a target point behind the ball, and distance and angle thresholds for being in position to kick, are learned.

When optimizing kick parameters the ball is placed at the center of the field and the agent, placed 1.5 meters behind the ball (or directly behind the ball in the case of KickLong as it is only used for kickoffs), is asked to walk forward and kick the ball toward the center of the opponent's goal. An agent is given a reward for how far it is able to kick the ball in the forwards direction (*distForward*). To promote accuracy a slight penalty is also given for the distance the ball is kicked to either side (*distSideways*). Additionally, as it is important to quickly position behind the ball so as to kick it before an opponent approaches, a penalty is given for the amount of time it takes to position for a kick (*timePositioning*). The following equation gives the reward an agent receives when performing a kick (where distances are in meters and time is in seconds):

$$reward = distForward - .75 * distSideways - timePositioning/8.0$$

If, while positioning to kick, the agent should run into the ball causing the ball to travel greater than .3 meters from its starting spot, a reward of -1 is given for the kick. This is done to ensure the agent doesn't cheat during optimizations by dribbling the ball forward before kicking to gain extra distance. Also the agent is given a reward of -1 when kicking if it falls over while attempting kicks for which it is expected to be stable after performing (KickMedium and KickIK).

All kick optimizations were done across 200 generations of CMA-ES using a population size of 150. Ten kicks were performed for each candidate set of kick parameters being evaluated. Candidates were then assigned a fitness value equal to the average reward of these kicks.

4.5 Optimization Results

Results of optimizing UT Austin Villa's different kicks are shown in Table 2. KickLong was seeded with the 2011 team's kick used for kickoffs. The 2011 kick was optimized in a similar fashion to the 2012 kicks, however for 2012 six additional parameters were learned for adjusting the joint angles of the support (non-kicking) leg. Adding these parameters to the optimization, which increased the number of parameters optimized from 18 to 24, provided a huge performance boost as the kick distance more than doubled from the 2011 kick seed's distance of 5.3 meters. Allowing the agent to fall over after kicking, as opposed to requiring it to be stable as was done in 2011, resulted in a further gain in performance of about a meter as it allowed the agent to throw its body at the ball.

KickMedium and KickQuick were designed from the same seed as KickLong except some of the frames of motion for them were sped up or removed in order to make the kicks faster to get off. As the speed of the seed kick for

KickQuick had to be greatly modified to get it to kick over twice as fast as the original 2011 kick seed, all possible joint angles on the kick were opened up for optimization resulting in more than double the amount of parameters being optimized compared to that of KickLong and KickMedium. Both KickMedium and KickQuick's kick distances were optimized to be well over that of the 2011 kick.

While KickIK has the shortest distance of all the kicks, it is the fastest to get off and, due to its use of inverse kinematics, is generally more robust than the fixed pose keyframe kicks. Additionally, as inverse kinematics allow the kick to adjust to different ball positions, the optimized thresholds for positioning behind the ball can be greater than that of the fixed pose keyframe kicks allowing for faster kick positioning. The time required to position for KickIK is noticeably faster than the time taken to position using the 2011 inverse kinematics based kicks. This is due to the 2011 kicks using inverse kinematics calculations instead of distance and angular thresholds as a trigger for when to kick.

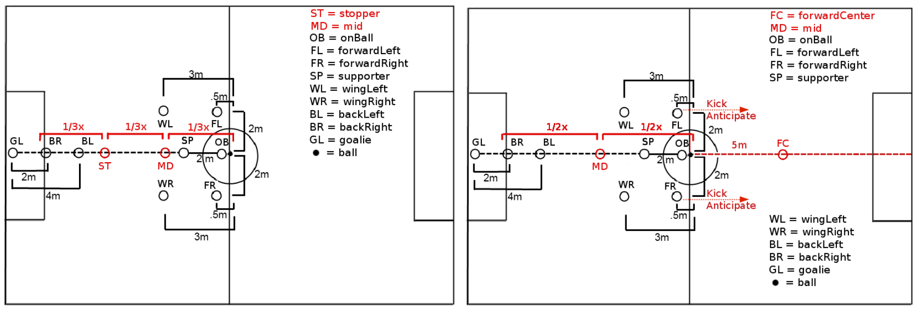
It is worth mentioning that while many of the kicks get a lot of height, which is great for kicking over opponents, height was never something that was optimized for. Optimizing for distance results in the ball being kicked in the air as it is able to travel farther when airborne with no friction from the ground slowing it down.

Kick	Parameters	Distance (m)	Height (m)	Time (s)	Stable
KickLong	24	12.20 (11.86)	1.57 (1.36)	2.44	No
KickMedium	24	10.80 (10.46)	1.30 (0.59)	2.12	Yes
KickQuick	51	8.79 (7.30)	1.11 (0.99)	0.92	No
KickIK	42	6.05 (4.82)	0.25 (0.06)	0.25	Yes

Table 2. Kick optimizations with the number of parameters optimized, the maximum height and distances recorded from ten kicks (with the median value shown in parentheses), the time taken to execute each, and also whether or not the agent is stable and doesn't fall over after executing the kick.

5 Dynamic Positioning

With the increase in team size from 9 to 11 agents for the 2012 competition two new role positions were added to UT Austin Villa's base formation: *stopper* and *mid* roles. Both role positions, shown in Figure 3(a), stay on a line running from the center of the goal to the ball. The *stopper* role stays 1/3 of the way between the top of the goal box and the ball while the *mid* role stands 2/3 of the way between these two points. UT Austin Villa also created a more offensive formation designed to take advantage of its new longer kicks described in Section 4. This formation, shown in Figure 3(b), replaces the *stopper* role with a *forwardCenter* role positioned 5 meters beyond the ball along a line from the ball to the center of the opponent's goal. The *mid* role is pushed back to be halfway between the top of the goal box and the ball. A key feature of this formation is the notion of *kick anticipation* where an agent attempting to kick the ball alerts its teammates of the target position the ball is being kicked to. When this target position is



(a) Base Formation (b) Kicking Formation

Fig. 3. Formations used by UT Austin Villa. Added positions for the 2012 competition are shown in red.

broadcasted both *forwardLeft* and *forwardRight* role positions move to the area that the ball is being kicked to in anticipation of the kick.

The UT Austin Villa team used a dynamic role and formation positioning system described in [7] to position its players. This system, at its base, assigns agents to the precomputed role positions on the field so as to avoid collisions and minimize the longest distance any agent has to travel. The positioning system is similar to the one used for the 2011 competition, but was upgraded with enhancements listed in [7]: using path costs and assigning the *supporter* (previously called *stopper* in [7]) role to the nearest agent.

When considering assignments of agents to positions there are $n!$ possible combinations. Using dynamic programming the positioning system only needs to evaluate $n2^{n-1}$ assignments of agents to positions in order to compute an optimal assignment. With the increase from 9 to 11 agents for the 2012 competition scalability becomes a concern, however, because all computations must be performed within 20 ms (the cycle time of the server). As the goalie positions itself only $n = 8$ or 1024 combinations were required to be computed in 2011, but this jumped to $n = 10$ or 5120 combinations to process for the 2012 competition. Despite only needing 3.3 ms to calculate positioning in 2011, the 5X increase in positioning computations for 2012 took up most of an agent's allotted processing time, and left it with little time for other components to do necessary computations.

In order to keep the positioning system from taking too long, a self-monitoring mechanism was put in place where the agent records the amount of time taken to compute positioning role assignments as well as the number n of agents it has assigned to role positions. Should the positioning system take longer than MAX_TIME (set to 10 ms) to run, the agent reduces the maximum number of agents ($maxN$) the positioning system is allowed to evaluate by setting $maxN = maxN - 1$. Alternatively if the positioning system takes less than $MAX\_TIME/2$ to complete then the number of allowed agents to evaluate for positioning is increased by setting $maxN = maxN + 1$. When $maxN$ is less than the number of n agents that need to be positioned then $n - maxN$ agents furthest from the ball are greedily assigned to their nearest role positions. The intuition in greedily assigning agents furthest from the ball to possibly suboptimal role position assignments is that they are less critical to game performance compared to agents closer to the ball. By monitoring the running time of the positioning system, and reducing how many computations it does if it is taking too long, the system can scale to different numbers of agents as well as adapt on the fly to computers with different processors and fluctuating CPU loads.

6 Tournament Results and Analysis

In winning the 2012 RoboCup competition UT Austin Villa finished with a record of 12 wins, 2 losses, and 3 ties.[4] During the competition the team scored 39 goals and only conceded 4. This was not nearly as dominant of a performance as was seen in the 2011 competition when the team won all 24 games it played while scoring 136 goals and conceding none. Several reasons can be attributed to this dip in performance. There was a general decrease in goals scored during the tournament due to the larger field and increase in the number of agents on a team. Additionally early in the tournament there were network problems causing instability that resulted in many teams' agents to lose their balance and have trouble walking. This was very noticeable during the first round when UT Austin Villa suffered both of its losses. UT Austin Villa eventually beat both the teams it lost to (magmaOffenburg and RoboCanes) during the semifinals and finals rounds. A large amount of credit must also be given to the other teams in the tournament as they exhibited a substantial improvement in overall play.

As reported in [5], the 2011 UT Austin Villa team was able to beat all teams in the 2011 competition by at least 1.45 goals on average, and when playing 100 games against every team from the 2011 tournament, UT Austin Villa won every game but 21 of them which were ties (no losses). As seen in Table 3, the 2012 UT Austin Villa team was only able to beat the 2012 2nd place team (RoboCanes) by an average of 0.88 goals and tied them 32 times across 100 games. Although the data in Table 3 shows that UT Austin Villa winning the 2012 RoboCup competition was statistically significant, and that the team didn't lose any games or concede any goals against the other top teams, there was a decent chance of the tournament being decided by penalty kicks due to UT Austin Villa tieing the 2nd place team almost 1/3 of the time. It is thus not surprising that the championship game wasn't decided until the second half of extra time (which UT Austin Villa won 2-0).

It is worth mentioning that the optimized get up mentioned in Section 3 for when an agent is lying on its front (a situation that occurs much less frequently than that of an agent lying on its back) was never used in the competition as the tournament's early network problems, and resulting agent instability, were

[4] Full tournament results, as well as a highlights video of the competition, can be found at `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2012/html/results_3d/`

Opponent	Average Goal Difference	Record (W-L-T)	Goals (For-Against)
RoboCanes	0.88 (0.08)	68-0-32	88-0
Bold Hearts	1.64 (0.09)	89-0-11	164-0
magmaOffenburg	1.87 (0.10)	94-0-6	187-0

Table 3. UT Austin Villa's released binary's performance when playing 100 games against released binaries of the 2nd, 3rd, and 4th places teams in the tournament: RoboCanes, Bold Hearts, and magmaOffenburg respectively. Values in parentheses are the standard error.

causing the get up routine to fail. Also it was noticed that kicking was often just turning the ball over to the other team, and so in the later rounds of the tournament the kicking formation was abandoned in favor of the base formation (Figure 3), and the agent only kicked if it thought the kick would score a goal.

In order to quantify gains in performance due to improvements in the agent for the 2012 competition, versions of the UT Austin Villa agent missing different improvements were created and then played against the other teams that made the semifinals. This includes an agent that does not use the optimized get ups in Section 3, an agent without the improved kickoff using KickLong, as well as one that does use KickLong on kickoffs but otherwise just dribbles (shown to be the best performing agent for 2011 in [5]) instead of using any of the other optimized kicks discussed in Section 4, and an agent using the base formation (Figure 3(a)) and positioning system scaled to support 11 agents, but without the improvements to positioning mentioned in Section 5. Additionally an agent missing all improvements was evaluated, as well as a version of the agent using the kicking formation (Figure 3(b)) and kick anticipation. Game performance of different agent variants can be seen in Table 4.

Agent	Average Goal Difference
2012 UT Austin Villa Released Binary	1.46
No Improved Kickoff	1.37
No Kicking	1.36
No Improved Positioning	1.19
No Improved Get Up	.95
No Improvements (2011 Base)	.92
Kicking Formation with Kick Anticipation	.82

Table 4. Average goal difference when playing 100 games against the released binaries of the other teams in the semifinals: RoboCanes, Bold Hearts, and magmaOffenburg.

In Table 4 we see that all improvements to the agent were beneficial as missing any single one of them hurt performance and resulted in a lower average goal difference. The most important improvement was that of the optimized get up which resulted in approximately a half a goal increase in performance against the other top teams at the competition. Without any of the improvements for 2012 the team's average goal difference dropped by over half a goal. This includes a 0.59 average goal difference against the 2nd place team (RoboCanes) resulting in a nearly even split between the number of games won and tied against this opponent. It is fortunate that the kicking formation with kick anticipation was abandoned midway through the tournament as it gave the worst performance.

7 Conclusion

UT Austin Villa, bolstered by improvements to its get up routine, kicking, and positioning systems, repeated as 3D simulation league champions at the 2012 RoboCup competition. Although the team was still largely able to lean on dribbling using its stable and fast omnidirectional walk [1] to win the competition, the focus of the team for the 2013 competition will be to continue to improve on its kicking system and integrate passing into the team's strategy. It became clear during the championship match,[5] during which the team was unable to score until the second period of extra time — capped off by a last second goal on a kick from outside the goal box, that kicking will need to be a vital part of the team's strategy if UT Austin Villa is to win a third championship in a row.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-0917122), ONR (N00014-09-1-0658), and the Federal Highway Administration (DTFH61-07-H-00030). Patrick MacAlpine is supported by a NDSEG fellowship.

References

1. MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12). (2012)
2. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011 3D Simulation Team report. Technical Report AI11-10, The Univ. of Texas at Austin, Dept. of Computer Science, AI Laboratory (2011)
3. Hansen, N.: The CMA Evolution Strategy: A Tutorial. (2009) `http://www.lri.fr/~hansen/cmatutorial.pdf`.
4. Urieli, D., MacAlpine, P., Kalyanakrishnan, S., Bentor, Y., Stone, P.: On optimizing interdependent skills: A case study in simulated 3d humanoid robot soccer. In: Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011). (2011)
5. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition. In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012). (2012)
6. Diankov, R., Kuffner, J.: Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA (2008)
7. MacAlpine, P., Barrera, F., Stone, P.: Positioning to win: A dynamic role assignment and formation positioning system. In: RoboCup-2012: Robot Soccer World Cup XVI. Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin (2013)

[5] Videos of the championship match, as well as more information about the UT Austin Villa team, can be found on the UT Austin Villa team's homepage: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/`