To appear in *Proceedings of the RoboCup International Symposium 2019* (RoboCup 2019), Sydney, Australia, July 2019.

UT Austin Villa: RoboCup 2019 3D Simulation League Competition and Technical Challenge Champions

Patrick MacAlpine† and Faraz Torabi‡ and Brahma Pavse‡ and Peter Stone‡

† Microsoft Research, USA
patmac@microsoft.com
‡The University of Texas at Austin, USA
faraztrb@utexas.edu, brahmasp@utexas.edu, pstone@cs.utexas.edu

Abstract. The UT Austin Villa team, from the University of Texas at Austin, won the 2019 RoboCup 3D Simulation League, and in doing so finished with an overall record of 21 wins, 1 tie, and 1 loss. During the course of the competition the team scored 112 goals while conceding only 5. Additionally the team won the RoboCup 3D Simulation League technical challenge by accumulating the most points across two league challenges: fewest self-collisions challenge and free challenge. This paper describes the changes and improvements made to the team between 2018 and 2019 that allowed it to win both the main competition and technical challenge.

1 Introduction

UT Austin Villa won the 2019 RoboCup 3D Simulation League for the eighth time in the past nine years, having also won the competition in 2011 [1], 2012 [2], 2014 [3], 2015 [4], 2016 [5], 2017 [6], and 2018 [7] while finishing second in 2013. During the course of the competition the team scored 112 goals while conceding only 5 along the way to finishing with an overall record of 21 wins, 1 tie, and 1 loss. Many of the components of the 2019 UT Austin Villa agent were reused from the team's successful previous years' entries in the competition. This paper is not an attempt at a complete description of the 2019 UT Austin Villa agent, the base foundation of which is the team's 2011 championship agent fully described in a team technical report [8], but instead focuses on changes made in 2019 that helped the team repeat as champions.

In addition to winning the main RoboCup 3D Simulation League competition, UT Austin Villa also won the RoboCup 3D Simulation League technical challenge by winning each of the two league challenges: fewest self-collisions challenge and free challenge. This paper also serves to document these challenges and the approaches used by UT Austin Villa when competing in the challenges.

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 details changes and improvements to the 2019 UT

Austin Villa team: reduction of self-collisions and use of a new pass mode, while Section 4 analyzes the contributions of these changes in addition to the overall performance of the team at the competition. Section 5 describes and analyzes the fewest self-collisions challenge, while also documenting the overall league technical challenge consisting of both the fewest self-collision challenge and a free/scientific challenge. Section 6 concludes.

2 Domain Description

The RoboCup 3D simulation environment is based on SimSpark [9, 10], a generic physical multiagent system simulator. SimSpark uses the Open Dynamics Engine (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.

Games consist of 11 versus 11 agents playing two 5 minute halves of soccer on a 30×20 meter field. The robot agents in the simulation are modeled after the Aldebaran Nao robot, which has a height of about 57 cm, and a mass of 4.5 kg. 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 speed/direction in which to move a joint.

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.

In addition to the standard Nao robot model, four additional variations of the standard model, known as heterogeneous types, are available for use. These variations from the standard model include changes in leg and arm length, hip width, and also the addition of toes to the robot's foot. Teams must use at least three different robot types, no more than seven agents of any one robot type, and no more than nine agents of any two robot types.

One significant change for the 2019 RoboCup 3D Simulation League competition was penalizing self-collisions. While the simulator's physics model can detect and simulate self-collisions—when a robot's body part such as a leg or arm collides with another part of its own body—having the physics model try to process and handle the large number of self-collisions occurring during games often leads to instability in the simulator causing it to crash. To preserve stability of the simulator self-collisions are purposely ignored by the physics model. However, not modeling self-collisions can result in robots performing physically impossible motions such as one leg passing through the other when kicking the ball. In order to discourage teams from having robots with self-colliding behaviors, a new feature was added to the simulator this year to detect and penalize self-collisions when they happen. This feature signals a self-collision as having occurred if two body parts of a robot overlap by more than 0.04 meters, and then all joints in any arm or leg of the robot involved in the self-collision are frozen and not allowed to move for one second. Freezing the joints in an arm or leg that has started to collide with another body part is an approximation of the physics model preventing body parts from moving through each other, and also detracts from the performance of the robot due to its limb being "numb" and immobile. After the second passes, the joints are unfrozen, and the robot is allowed to move its self-colliding body parts for two seconds without any self-collisions being reported. This two second period, during which previously collided body parts are no longer penalized and frozen for self-collisions, allows a robot time to reposition its body to no longer have a self-collision.

The other major change for the 2019 RoboCup 3D Simulation League competition from previous years was the addition of a new pass play mode to encourage more passing and teamwork. The pass play mode allows players some extra time on the ball to kick and pass it during which time the opponent is prevented from interfering with a kick attempt. A player may initiate the pass play mode as long as the following conditions are all met:

- The current play mode is PlayOn.
- The agent is within 0.5 meters of the ball.
- No opponents are within a meter of the ball.
- The ball is stationary as measured by having a speed no greater than 0.05 meters per second.
- At least three seconds have passed since the last time a player's team has been in pass mode.

Once pass mode for a team has started the following happens:

- Players from the opponent team are prevented from getting within a meter of the ball.
- The pass play mode ends as soon as a player touches the ball or four seconds have passed.
- After pass mode has ended the team who initiated the pass mode is unable to score for ten seconds—this prevents teams from trying to take a shot on goal out of pass mode.

3 Changes for 2019

While many components developed prior to 2019 contributed to the success of the UT Austin Villa team including dynamic role assignment [11], marking [12], and an optimization framework used to learn low level behaviors for walking and kicking via an overlapping layered learning approach [13], the following subsections focus only on those that are new for 2019: reduction of self-collisions and use of the new pass mode. A performance analysis of these components is provided in Section 4.1.

3.1 Reduction of Self-Collisions

The UT Austin Villa team specifies motions for kicking, getting up, and goalie diving skills through a periodic state machine with multiple key frames, where each key frame is a parameterized static pose of fixed joint positions. Figure 1 shows an example series of poses for a kicking motion. The joint angles are optimized using the CMA-ES [14] algorithm and overlapping layered learning [13] methodologies.

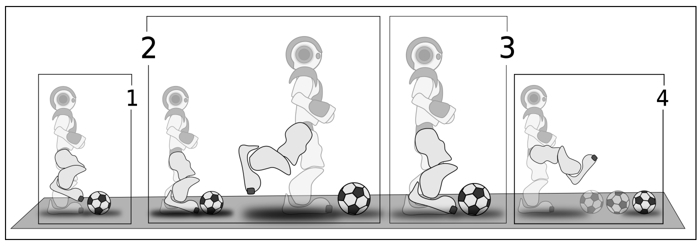

Fig. 1: Example of a fixed series of poses that make up a kicking motion.

During learning the robot runs through an optimization task where it performs a skill (e.g. attempting to kick a ball or standing up after having fallen over). At the conclusion of the optimization task a *fitness* value is awarded for how well the robot performed on the optimization task (e.g. how far the robot kicked a ball or how quickly it was able to stand up). Prior to 2019 robots were not penalized for self-collisions, so many of the skills that were learned for the robots inadvertently contained self-collisions as there was no incentive during learning to avoid them. The skills that contained self-collisions no longer worked correctly with this year's introduction of penalizing self-collisions, however, so it was necessary to try to reduce the number of self-collisions as much as possible in order to fix the broken skills.

As a first step toward reducing self-collisions, it is necessary to determine which skills contain self-collisions. In order to identify the sources of self-collisions, UT Austin Villa played thousands of games against different opponents. During these games whenever an agent had a self-collision the skill the agent was performing at the time of the self-collision was recorded along with the agent's uniform number—the agent's uniform number can be used to identify the agent's robot model as robot models are assigned to an agent based on the agent's uniform number, and there is a different set of skills for each robot model due to the physical differences between robot models [3]. The total number of self-collisions for each executed skill for every agent uniform number (1-11) was then computed from the recorded data across all the games played. Table 1 shows an example of this data for the agent with uniform number 2.

From the data in the second column of Table 1 it is clear there are many self-collisions across different kicks, as well as a very large number of self-collisions when trying to get up after the robot has fallen on its front. To reduce the number

Table 1: The number of self-collisions recorded by the agent with uniform number 2 (a type 4 robot model with toes) for different skills across 6000 games both before and after reducing self-collisions.

	Self-Collisions	
Skill	Before Reduction	After Reduction
SKILL_GETUP_BACK	152	89
SKILL_GETUP_FRONT	1363	0
SKILL_KICK_8M	14	0
SKILL_KICK_13M	67	0
SKILL_KICK_14M	313	0
SKILL_KICK_15M	964	0
SKILL_KICK_16M	485	0
SKILL_KICK_17M	126	0
SKILL_KICK_18M	649	0
SKILL_KICK_19M	812	0
SKILL_KICK_20M	737	1

of self-collisions occurring when executing these skills, the following strategies were employed:

Hand fix: When a self-collision occurs, the simulator reports which body parts of a robot collided with each other. For kicking skills the body parts that matter the most are those in the legs, so if a robot's arm is involved in a self-collision the arm's movement can probably be adjusted without affecting the kicking motion. Roughly half the kicking skills that had self-collisions involved the robots' arms in the self-collisions, so we were able to manually adjust the arms' joint angle positions to no longer self-collide while still exhibiting the same kicking motion through the ball.

Reoptimize current self-colliding behavior: In many cases it is not easy to hand adjust the motions of a skill to avoid a self-collision as doing so fundamentally changes the performance of the skill (e.g. adjusting the position of the legs of a robot for a kicking skill when the robot's legs self-collide). Instead of trying to fix things by hand, the current skill can be relearned with CMA-ES using the current self-colliding behavior as a starting point for learning, while also adding a large penalty value to the fitness of an agent if it has any self-collisions while performing the optimization task it is trying to learn.

Reoptimize starting from similar behavior: If the previous strategy does not work—possibly because the current behavior has too many self-collisions such that it is hard to find a behavior that does not have self-collisions when using the current self-colliding behavior as a starting point—one can instead attempt to learn using a similar related skill (e.g. similar distance kick) that has fewer collisions as a starting point for learning.

Reoptimize with a tighter threshold for self-collisions: Some skills have infrequent enough self-collisions that they do not always occur during a learning trial, but still experience a significant number of self-collisions during games. It can be especially hard to reduce the number of self-collisions for skills when self-collisions are not always detected during learning. As a way to decrease the chance of the robot assuming body positions that are right on the border of having self-collisions, one can decrease the allowed amount of overlap between body parts in the simulator before a self-collision is considered to have occurred. By decreasing the amount of allowed overlap between body parts during learning it is less likely that a learned behavior will have self-collisions exceeding the actual allowed amount of overlap.

All of the strategies mentioned were used to to reduce self-collisions in 35 of UT Austin Villa's previously learned skills. This reduction of self-collisions dramatically lowered the average number of self-collisions exhibited by the team during a game from 10.507 down to 0.137, thus removing almost 99% of previous self-collisions. The large reduction in self-collisions can be seen in the third column of Table 1. The impact of reducing self-collisions on the team's performance is evaluated in Section 4.1, and the number of self-collisions UT Austin Villa had compared to other teams is detailed in the evaluation of the fewest self-collisions challenge in Section 5.2.

3.2 Pass Mode Strategy

To best take advantage of the new pass mode, players must carefully decide when to activate it. If players were to naively activate pass mode at every opportunity to do so they would have a difficult time scoring as a team must wait ten seconds after their pass mode ends before they are allowed to score. If a team never uses pass mode, however, they will miss out on opportunities to kick the ball without their opponent being able to interfere with the kick. Given these considerations, the following is the strategy UT Austin Villa employs for using pass mode:

– Only activate pass mode when an opponent is within 1.25 meters of the ball. Activating pass mode before the opponent is close is unnecessary as the opponent is not yet a threat to interfere with a kick, and the later pass mode is activated the later it will time out leaving more time to kick the ball before pass mode eventually ends.
– Do not use pass mode when a player is close enough to take a shot on goal and score. Goals cannot be scored for ten seconds after pass mode ends, so it is better to attempt a shot and try to score than to pass the ball and then have to wait ten seconds to score.
– Do use pass mode if a player is not behind the ball even if the player is close enough to the opponent's goal to take a shot and score. The player will have to take some time to walk around the ball to get in position to take a shot, and at that point it is likely the opponent will have gotten close enough to the ball to interfere with a potential shot.

The gain in team performance when using UT Austin Villa's pass mode strategy is evaluated in Section 4.1.

4 Main Competition Results and Analysis

In winning the 2019 RoboCup competition UT Austin Villa finished with an overall record of 21 wins, 1 tie, and 1 loss.[1] During the course of the competition the team scored 112 goals while conceding only 5. Despite the team's strong performance at the competition, the relatively few number of games played at the competition, coupled with the complex and stochastic environment of the RoboCup 3D simulator, make it difficult to determine UT Austin Villa being better than other teams by a statistically significant margin. At the end of the competition, however, all teams were required to release their binaries used during the competition. Results of UT Austin Villa playing 1000 games against each of the other six teams' released binaries from the competition are shown in Table 2.

Table 2: UT Austin Villa's released binary's performance when playing 1000 games against the released binaries of all other teams at RoboCup 2019. This includes place (the rank a team achieved at the 2019 competition), average goal difference (values in parentheses are the standard error), win-loss-tie record, and goals for/against.

Opponent	Place	Avg. Goal Diff.	Record (W-L-T)	Goals (F/A)
magmaOffenburg	2	2.403 (0.048)	913-9-78	2496/93
WrightOcean	3	2.735 (0.042)	952-5-43	3006/271
HfutEngine	5	4.733 (0.054)	995-0-5	4751/18
BahiaRT	4	6.360 (0.054)	1000-0-0	6361/1
FCPortugal	6	7.309 (0.052)	1000-0-0	7489/180
ITAndroids	7	9.670 (0.063)	1000-0-0	9721/51

UT Austin Villa finished with at least an average goal difference greater than 2.4 goals against every opponent. Additionally, UT Austin Villa's win percentage was greater than 91% against each team, and out of the 6000 games that were played in Table 2 the team only lost 14. These results show that UT Austin Villa winning the 2019 competition was far from a chance occurrence. The following subsection analyzes the contributions of reducing self-collisions and use of a new pass mode (both described in Section 3) to the team's dominant performance.

4.1 Analysis of Components

To analyze the contribution of new components for 2019—reduction of self-collisions and use of the new pass mode (Section 3)—to the UT Austin Villa team's performance, we played 1000 games between a version of the 2019 UT Austin Villa team with each of these components turned off—and no other changes—against each of the RoboCup 2019 teams' released binaries. Results comparing the performance of the UT Austin Villa team with and without using these components are shown in Table 3.

Results show that without using pass mode or reducing self-collisions the team's performance drops significantly. Furthermore, if UT Austin Villa had not

[1] Full tournament results can be found at `http://www.cs.utexas.edu/~AustinVilla/?p=competitions/RoboCup19#3D`

Table 3: Different versions of the UTAustinVilla team when playing 1000 games against the released binaries of all teams at RoboCup 2019. Values shown are average goal difference with values in parentheses being the difference in performance from the team's released binary.

Opponent	No Pass Mode	Self-Collisions	No Pass Mode + Self-Collisions
UTAustinVilla	-0.677 (-0.677)	-1.986 (-1.986)	-2.075 (-2.075)
magmaOffenburg	1.843 (-0.560)	1.814 (-0.589)	1.312 (-1.091)
WrightOcean	2.104 (-0.631)	1.487 (-1.248)	0.765 (-1.970)
HfutEngine	4.391 (-0.342)	3.509 (-1.224)	3.333 (-1.400)
BahiaRT	6.255 (-0.105)	5.409 (-0.951)	4.863 (-1.497)
FCPortugal	6.966 (-0.343)	4.869 (-2.440)	4.653 (-2.656)
ITAndroids	9.379 (-0.291)	6.461 (-3.209)	6.128 (-3.542)

used either pass mode or reduced self-collisions, the team would have only beaten WrightOcean by an average of 0.765 goals which correlates to 60.8% of games being wins, 23.9% ties, and 15.3% losses.

4.2 Additional Tournament Competition Analysis

To further analyze the tournament competition, Table 4 shows the average goal difference for each team at RoboCup 2019 when playing 1000 games against all other teams at RoboCup 2019.

Table 4: Average goal difference for each team at RoboCup 2019 (rows) when playing 1000 games against the released binaries of all other teams at RoboCup 2019 (columns). Teams are ordered from most to least dominant in terms of winning (positive goal difference) and losing (negative goal difference).

	UTA	mag	Wri	Hfu	Bah	FCP	ITA
UTAustinVilla	—	2.403	2.735	4.733	6.360	7.309	9.670
magmaOffenburg	-2.403	—	0.021	1.376	2.783	3.286	3.464
WrightOcean	-2.735	-0.021	—	1.160	2.503	4.599	5.105
HfutEngine	-4.733	-1.376	-1.160	—	0.315	0.981	1.525
BahiaRT	-6.360	-2.783	-2.503	-0.315	—	0.633	0.386
FCPortugal	-7.309	-3.286	-4.599	-0.981	-0.633	—	0.084
ITAndroids	-9.670	-3.464	-5.105	-1.525	-0.386	-0.084	—

It is interesting to note that the ordering of teams in terms of winning (positive goal difference) and losing (negative goal difference) is transitive—every opponent that a team wins against also loses to every opponent that defeats that same team. Relative goal difference does not have this same property, however, as a team that does better against one opponent relative to another team does not always do better against a second opponent relative to that same team. UT Austin Villa is dominant in terms of relative goal difference, however, as UT Austin Villa has a higher goal difference against each opponent than all other teams against the same opponent.

5 Technical Challenges

During the competition there was an overall technical challenge consisting of two different league challenges: free and fewest self-collision challenges. For each league challenge a team participated in, points were awarded toward the overall technical challenge based on the following equation:

$$\texttt{points}(rank) = 25 - 20 * (rank - 1)/(numberOfParticipants - 1)$$

Table 5: Overall ranking and points totals for each team participating in the RoboCup 2019 3D Simulation League technical challenge as well as ranks and points awarded for each of the individual league challenges that make up the technical challenge.

	Overall		Free		Fewest Self-Collisions	
Team	Rank	Points	Rank	Points	Rank	Points
UTAustinVilla	**1**	**45**	**2**	**20**	**1**	**25**
FCPortugal	2	36.7	1	25	5	11.7
magmaOffenburg	2	36.7	3	15	2	21.7
ITAndroids	4	28.3	4	10	3	18.3
WrightOcean	5	15	—	—	4	15
BahiaRT	6	13.3	5	5	6	8.3
HfutEngine	7	5	—	—	7	5

Table 5 shows the ranking and cumulative team point totals for the technical challenge as well as for each individual league challenge. UT Austin Villa won the fewest self-collisions challenge and finished second in the free challenge resulting in a first place finish in the overall technical challenge. The following subsections detail UT Austin Villa's participation in each league challenge.

5.1 Free Challenge

During the free challenge, teams give a five minute presentation on a research topic related to their team. Each team in the league then ranks the presentations with the best receiving a score of 1, second best a score of 2, etc. Additionally several respected research members of the RoboCup community outside the league rank the presentations, with their scores being counted double. The winner of the free challenge is the team that receives the lowest score. Table 6 shows the results of the free challenge in which UT Austin Villa was awarded second place.

Table 6: Results of the free challenge.

Team	Score
FCPortugal	28
UTAustinVilla	**38**
magmaOffenburg	40
ITAndroids	51
BahiaRT	68

UT Austin Villa's free challenge submission[2] presented research on learning skills by observing a single demonstration of a skill by another agent [15]. In

[2] Free challenge entry description available at `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2019/files/UTAustinVillaFreeChallenge2019.pdf`

particular, we showed that an agent could use a PID controller as an inverse dynamics model to mimic and improve upon its opponent's soccer skills by combining the use of a single demonstration and the environment-provided sparse reward. Moreover, this single demonstration consists of only joint angles per time-step, i.e., the learner is only exposed to how the opponent's joint configuration is transitioning each time-step, it has no knowledge of the torque applied to achieve the transition. Using the yearly released binary files, we artificially created the opponent demonstration by triggering desired behaviors by, for example, placing the ball in specific locations to induce a long distance kick. In order to retrieve the joint angles per time-step for specific tasks, we modified the simulator to output the joint angles of the agent when performing the task.

The other teams participating in the free challenge also presented interesting work:[3] FCPortugal presented work on how to learn fast human-like running and sprinting behaviors [16, 17], magmaOffenburg talked about learning a walk behavior utilizing toes from scratch, ITAndroids discussed Bottom-Up Meta-Policy Search (BUMPS) for learning robot skills, and BahiaRT presented a set of tools for learning set plays from demonstration [18].

5.2 Fewest Self-Collisions Challenge

Results of the fewest self-collisions challenge are shown in the second column of Table 7. UT Austin Villa won the challenge by only having one recorded self-collision during the entire competition. The average number of self-collisions when each team plays 1000 games against each of the other teams' released binaries is show in the third column of Table 7. UT Austin Villa also had the fewest number of self-collisions when playing 1000 games against each of the other teams' released binaries suggesting that UT Austin Villa winning the fewest self-collisions challenge was statistically probable.

Table 7: Average number of self-collisions per game for each team as recorded for the fewest self-collisions challenge and as measured when playing 1000 games against each of the other teams' released binaries

	Avg. Self-Collisions per Game	
Team	Challenge	Many Games
UTAustinVilla	**0.1**	**0.137**
magmaOffenburg	0.2	0.315
ITAndroids	2.0	2.936
WrightOcean	3.2	4.360
FCPortugal	3.5	2.732
BahiaRT	5.7	7.392
HfutEngine	8.5	8.069

[3] All participating teams' free challenge entry descriptions available at `http://archive.robocup.info/Soccer/Simulation/3D/FCPs/RoboCup/2019/`

6 Conclusion

UT Austin Villa won the 2019 RoboCup 3D Simulation League main competition as well as the overall league technical challenge.[4] Data taken using released binaries from the competition show that UT Austin Villa winning the competition was statistically significant. The 2019 UT Austin Villa team also improved from 2018 as it was able to beat the team's 2018 champion binary by an average of 0.7 (+/- 0.044) goals across 1000 games.[5]

In an effort to both make it easier for new teams to join the RoboCup 3D Simulation League, and also provide a resource that can be beneficial to existing teams, the UT Austin Villa team has released their base code [19].[6] This code release provides a fully functioning agent and good starting point for new teams to the RoboCup 3D Simulation League (it was used by two other teams at the 2019 competition: WrightOcean and HfutEngine). Additionally the code release offers a foundational platform for conducting research in multiple areas including robotics, multiagent systems, and machine learning.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-1637736, IIS-1651089, IIS-1724157), ONR (N00014-18-2243), FLI (RFP2-000), ARL, DARPA, Intel, Raytheon, and Lockheed Martin. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research. Patrick MacAlpine is an employee of Microsoft and supported by Microsoft Research.

References

1. 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)
2. MacAlpine, P., Collins, N., Lopez-Mobilia, A., Stone, P.: UT Austin Villa: RoboCup 2012 3D simulation league champion. In: RoboCup-2012: Robot Soccer World Cup XVI. LNAI. Springer (2013)

[4] More information about the UT Austin Villa team, as well as video from the competition, can be found at the team's website: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/#2019`

[5] So as to be compatible with the 2018 version of the team's binary the simulator was modified to not report when pass play mode was active to the 2018 team (the play mode was reported as still being PlayOn during pass mode to the 2018 team's agents), and self-collisions were not penalized.

[6] Code release at `https://github.com/LARG/utaustinvilla3d`

3. MacAlpine, P., Depinet, M., Liang, J., Stone, P.: UT Austin Villa: RoboCup 2014 3D simulation league competition and technical challenge champions. In: RoboCup-2014: Robot World Cup XVIII. LNAI. Springer (2015)
4. MacAlpine, P., Hanna, J., Liang, J., Stone, P.: UT Austin Villa: RoboCup 2015 3D simulation league competition and technical challenges champions. In: RoboCup-2015: Robot World Cup XIX. LNAI. Springer (2016)
5. MacAlpine, P., Stone, P.: UT Austin Villa: RoboCup 2016 3D simulation league competition and technical challenges champions. In: RoboCup 2016: Robot World Cup XX. LNAI. Springer (2016)
6. MacAlpine, P., Stone, P.: UT Austin Villa: RoboCup 2017 3D simulation league competition and technical challenges champions. In: RoboCup 2017: Robot World Cup XXI. LNAI. Springer (2018)
7. MacAlpine, P., Torabi, F., Pavse, B., Sigmon, J., Stone, P.: UT Austin Villa: RoboCup 2018 3D simulation league champions. In: RoboCup 2018: Robot World Cup XXII. Lecture Notes in Artificial Intelligence. Springer (2019)
8. 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)
9. Obst, O., Rollmann, M.: SPARK A Generic Simulator for Physical Multiagent Simulations. In: Multiagent System Technologies Proceedings of the MATES 2004. Volume 3187 of Lecture Notes in Artificial Intelligence. Springer (2004) 243–257
10. Xu, Y., Vatankhah, H.: Simspark: An open source robot simulator developed by the RoboCup community. In: RoboCup 2013: Robot World Cup XVII. Springer (2014) 632–639
11. MacAlpine, P., Price, E., Stone, P.: SCRAM: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15). (2015)
12. MacAlpine, P., Stone, P.: Prioritized role assignment for marking. In: RoboCup 2016: Robot World Cup XX. LNAI. Springer (2016)
13. MacAlpine, P., Stone, P.: Overlapping layered learning. Artificial Intelligence **254** (2018) 21–43
14. Hansen, N.: The CMA Evolution Strategy: A Tutorial. (2009) `http://www.lri.fr/~hansen/cmatutorial.pdf`.
15. Pavse, B.S., Torabi, F., Hanna, J., Warnell, G., Stone, P.: Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration. In: Imitation, Intent, and Interaction (I3) Workshop at ICML 2019. (2019)
16. Abreu, M., Lau, N., Sousa, A., Reis, L.P.: Learning low level skills from scratch for humanoid robot soccer using deep reinforcement learning. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE (2019) 1–8
17. Abreu, M., Reis, L.P., Lau, N.: Learning to run faster in a humanoid robot soccer environment through reinforcement learning. In: RoboCup 2019: Robot World Cup XXIII. Lecture Notes in Artificial Intelligence. Springer (2019)
18. Simões, M., Nogueira, T.: Towards setplays learning in a multiagent robotic soccer team. In: 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), IEEE (2018) 277–282
19. MacAlpine, P., Stone, P.: UT Austin Villa RoboCup 3D simulation base code release. In: RoboCup 2016: Robot World Cup XX. LNAI. Springer (2016)