KgpKubs Team Description Paper

Robocup 3D Simulation League 2020

Buridi Sree Aditya, Ranjith Kumar, Pratik Tibrewal, Shivansh Mundra, Eeshan Gupta, Shubhanan Shriniket, Prakhar Sharma, Prashant Ramnani, Sumegh Roychowdhury, Taapas Agrawal, Astitva Sharma, Kushal Kedia, Saurav Pandey, and Sriyash Poddar

Indian Institute of Technology, Kharagpur
West Bengal, India
taapas2897@gmail.com

Abstract. This paper reports the recent developments by the Kgpkubs team. It describes the work on movement, formation strategies, heuristic role assignment and techniques used to improve the game play.

1 Introduction

Kgpkubs is a team from the Indian Institute of Technology, Kharagpur, India. It aims to make autonomous soccer playing robots. For this, the team is currently focusing on the 3D Simulation and Small Size League Event in Robocup. Students from all departments and years are part of this including undergraduates and post-graduates. The principal investigator for the project is Prof. A.K. Deb and it is also mentored by Prof. Jayanta Mukhopadhyay , Prof. D.K. Pratihar and Prof. Sudeshna Sarkar. The research group is supported by the Centre for Excellence in Robotics, Indian Institue of Technology, Kharagpur. We have previously participated in FIRA RoboWorld Cup in the years 2013-2015 in the Mirosot League. In 2015, we secured Bronze position in the same. In 2016, 2017 and 2018 we participated in RoboCup (3D Simulation League). We also took part in the Robocup Asia Pacific 2017 3D Simulation League.

This work is organized as follows. First, we give an overview of our base architecture and strategy in Section 2. Section 3 describes about attacker and goalkeeper tactics. Section 4 describes about the Positioning and 5 about the Role Assignment Algorithm. Section 6 describes the approach we use for deciding and executing the pass. Section 7 describes the approach and results we got after training of our walk and kick. Section 8 describes the approach applied to learn low-level skills like walking and dribbling. Section 9 describes our ongoing work. Section 10 describes the various learning methods that we plan to use to optimize low level skills.

2 Overview

Our base architecture is based on the team UT-AustinVilla code available on Github `https://github.com/LARG/utaustinvilla3d/`. The code is divided into

appropriate modules and provides us with the flexibility to modify and develop easily.

Our strategy is based on using a mix of Delaunay Triangulation for proper positioning of robots on the field and assigning tactics for carrying out specific tasks. For Positioning, every robot calculates positions for all robots using Delaunay Triangulation and then uses the Hungarian Algorithm to find the position assignment for each robot. This result is then communicated by each robots taking turns. Upon receiving the results from the other 10 robots, the robot performs a voting to obtain the position and roles of the other robots. Some important roles like attacker, defender, goalie are assigned based on some heuristic methods overriding the Hungarian algorithm.

3 Tactics

Although we use Delaunay triangulation method to generate bot positions at certain instances of the game, it is not always possible to assign a predefined position to all agents. There is a need to obtain a separate tactic for those cases which may not yield desirable results with a predefined tactic data set. These cases are the most dynamic and important of all as they critically affect minor requisites which may arise during the game.

The Attacker bot is arguably the most dynamic bot on the field. This role is selected based upon certain heuristics like fallen status,distance from opponent etc. The attacker can quickly dodge, dribble, kick(fast and slow) and drive ball to goal. The attacker maintains ball possession by blocking the approaching opponent. A certain radius around the ball is checked for possible defenders and the dribble target is rotated by a suitable angle calculated by a function which takes into account the opponent's distance and velocity of approach. This results into the opponent colliding with our bot leading to a foul or our player crossing the opponent before it could reach the ball.

Goalkeeper is a static member of the game. It doesn't switches it's functionality with any other bot on the field. It occupy the near-to-goal area of the field and is most sensitive to minor changes in ball position and velocity. The goalkeeper dives when it knows the interpolated ball position is out of its reach in a given time window. Hence, the goalkeeper decides when to dive as an incorrectly timed or useless dive may actually do more bad than good. We have multiple types of dives that goalkeeper can use, depending upon game play.

The multi-agent coordination shows marked improvements upon previous year's approach; players obstructing the path of the attacker diverge away. This results in a decrease in collisions among our players and hence a more uninterrupted attacking game play is maintained.

4 Positioning Module

In soccer, player positioning and role allocation is a very important aspect of the game. Meticulous player positioning affects the general temperament of the

game and proper collaboration of various tactics is vital for a team to function efficiently.

Kgpkubs uses Voronoi-Cell Delaunay Triangulation method to generate and co-ordinate player positions with respect to the varying circumstances. Voronoi Cells are the result of a partitioning of the space into small regions based on their distances from their focal point. A point in a plane, say x, is said to lie in the Voronoi cell of a point y , if and only if the point x is more close to point y than any other point in the space.

Delaunay triangulation is the Dual graph of Voronoi cell plane. It ensures that no other focal point lies inside the circumcircle of the Delaunay triangle formed. Also, due to this property it tends to avoid skinny triangles. As a result, interpolating any point inside the triangle yields to a smooth-gradient continuous equation in terms of the coordinates of the vertices of the triangle.

The algorithm used to generate player positions uses statistical data (bot and ball positions under different conditions of the game) and generates a data set of agent positions with respect to certain ball positions. 65 ball positions in strategic locations were identified and triangulated using the incremental algorithm to generate Delaunay triangles. Once the triangles are generated, the Gouraud Shading algorithm yields the value of bot positions at any given point in terms of the values of bot positions stored at the vertices of the triangle enclosing it.

Initially, our bots are positioned symmetrically with respect to the symmetrical Voronoi points but it caused some bots to exit the field or cluster within a specific area of the field causing collisions. The positions of the bots with respect to a specific Voronoi Point can be further overridden during the game play based on certain heuristic methods. The best possible positions of a bot during a certain game state are considered to replace the currently allotted position. This can be used in similar game states. This helps in the dynamic positioning of the bots to an appreciable extent. There are further plans to implement a neural network architecture to aid in the computation of the best possible positions for the bot, using information from real-life soccer situations.

These computed formation points are fixed for every scenario. In order to handle the dynamic situations within the game, heuristics are used to override the formation points.

5 Role Assignment Module

The Hungarian algorithm is used which solves the role assignment problem in polynomial time. Time complexity of this algorithm is $O(n^3)$. At every second, as per the ball position, a set of target points are received from Voronoi Triangulation. The Voronoi updates are not made after every few cycles to prevent certain associated penalties:

- To save time as Voronoi updation is a computationally-expensive action.
- To prevent erratic bot behaviour arising due to sudden changes in ball position.

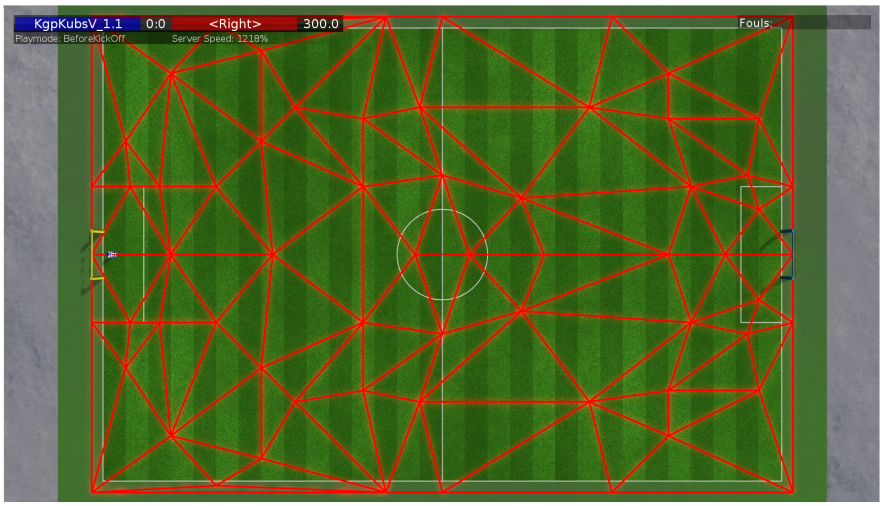

Fig. 1. Delaunay Triangles formed

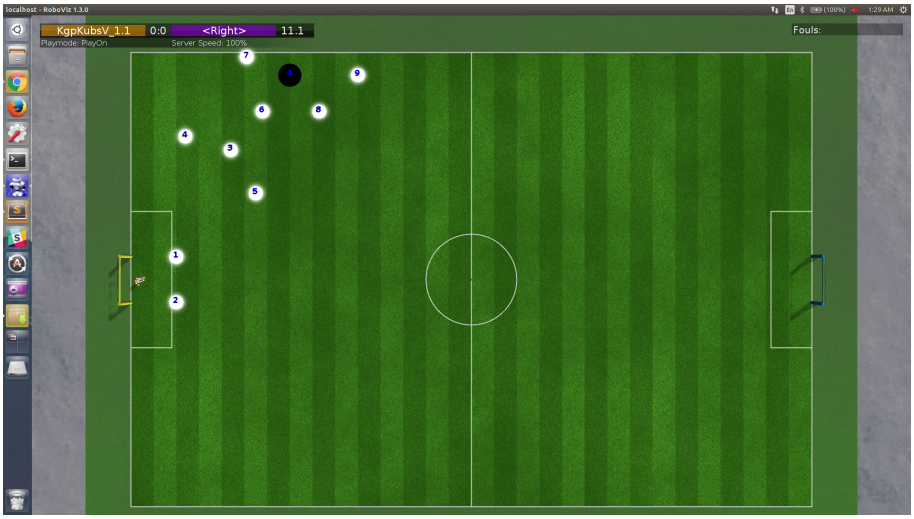

Fig. 2. Voronoi Points for a ball position

These set of target points are then matched to players on field by the Hungarian algorithm. The cost function used for the Hungarian algorithm is the euclidean distance between bot's current position and target location. This type of role matching has the following advantages :

(a) Collisions are mostly avoided.
(b) Longest distance is minimized
(c) It is dynamically consistent

There is a superficial layer which overrides the Hungarian mapping by evaluating certain Heuristics for specific roles to give better assignments. Role mapping is prioritized which assists in dynamic positioning of the robots.

6 Passing

Passing is one of the most essential elements for a multi-agent football playing system. A fuzzy logic based passing has been implemented. For each team member within a threshold radius, it takes into account :

- target player's distance from source player
- distances of opponent players from source player and target player
- proximity to goal of source player and target player
- angle to rotate for source player to face target player

If all those binary conditions are satisfied, a favourable position to pass is deemed, otherwise we continue as is. Few points to notice:

- while calculating distance, the values given to an opponent player ahead and behind of our own player are not assigned to be the same.
- only those places to pass where the kick can be kicked accurately are considered (at the time of writing, kgpkubs had not developed a kick which can cover the entire field).

7 Kicking

A simulation environment, based on Rcssserver3d was developed in which kicking was trained. 2 types of kicks were trained:

- Short Kick based on Inverse-Kinematics
- Long Range Kick

The skill file for kick has 22 joint angles as parameters, out of which 20 joint angles were used, excluding the 2 head joints which had no contribution to the kicking behaviour.

The Short Kick trained was accurate in both angle and direction specified. The Long Kick had a good ground clearance but lacked accuracy and consistency. While training the computation complexity was decreased by limiting the kick action to 5 time-frames and each time frame had a different duration which was also used as a hyperparameter for training. The trained parameters for Long kick had high variance in the results and the Nao-bot takes too much time to align itself in exact orientation to kick. This behaviour was undesirable since one can't afford to lose time during the match. Hence, parameters were manually tuned further such as keeping the threshold distance between ball and bot as fixed value and varying the angle, which not only reduced the reaction time of kick but also increased the distance of kick by 1m , making the Long Kick range standing at 10m, which is a significant improvement on our previous model.

8 CMA-ES

For optimizing low level skills, like walking, it may seem that reinforcement learning is more suitable but on the contrary CMA-ES performs at par with RL algorithms. We treat it as a black-box optimization algorithm, and have not interfered with the algorithm itself. Different cost function were used for optimizing different skills.The agent's behavior was decomposed into a representative set of subtasks, and multiple walk engine parameter sets were learnt in conjunction with each other.For example, asking the bot to walk on a straight line and using the difference in x-coordinate optimizes walking straight as well as improves its speed.The fitness of a given parameter set is proportional to the success in the given task during a given time. Similarly you can optimize other walk types (like lateral walking) and other skills like kicking and improving the time taken to get up once the bot falls down.We have achieved excellent results in terms of speed and stability, we improved our walking speed from 4.5 metres in 10 seconds to 10.8 metres in 10 seconds. Here we provide a brief article about the working of CMA-Evolutionary Strategy:

CMA-ES is a policy search algorithm that successively evaluates sets of candidates. Each candidate is evaluated with respect to a fitness measure. The next set of candidates is generated by sampling multivariate normal distribution that is biased toward directions of previously successful search steps. Recombination amounts to selecting a new mean value for the distribution. Mutation amounts to adding a random vector, a perturbation with zero mean. Adaptation of the covariance matrix amounts to learning a second order model of the underlying objective function. It is a parallel search algorithm so it can be run on a large server to make the optimization feasible. Some parameters were carefully chosen for optimization keeping others constant to reduce the search space.

9 Ongoing Work

Previously, work was done on the development of new code base in Python. Basic functionality of code is to establish connection between server and agent. The messages received from server are parsed using a parser and are processed to extract information.

Utility Class consists of functions for sending and receiving commands from the simulator which includes establishing connection to the server, initializing a bot, passing proper joint torques to the joints present in the Nao-Bot. On top of that Utility Class, a Humanoid Environment is created. It consists of initialization, step, make observation and reset functions which Reinforcement Learning Algorithms requires for their functioning. Environment Class also stores Robot's Perceptor Values. Each Robot has its independent Environment Class in this architecture.

Using this environment, we implemented Augmented Random Search algorithm, to optimize the skill of getting up. The environment was integrated with Gazebo which provides a stable and robust client-server connections. We further

aim to apply and get results from other similar algorithms and compare their results.

With python code-base, we tried to use deep reinforcement learning algorithms especially deep deterministic policy gradient, proximal policy optimization to better simple actions like getting up, walking and kicking by training the inverse kinematics parameters of the current walk engine.

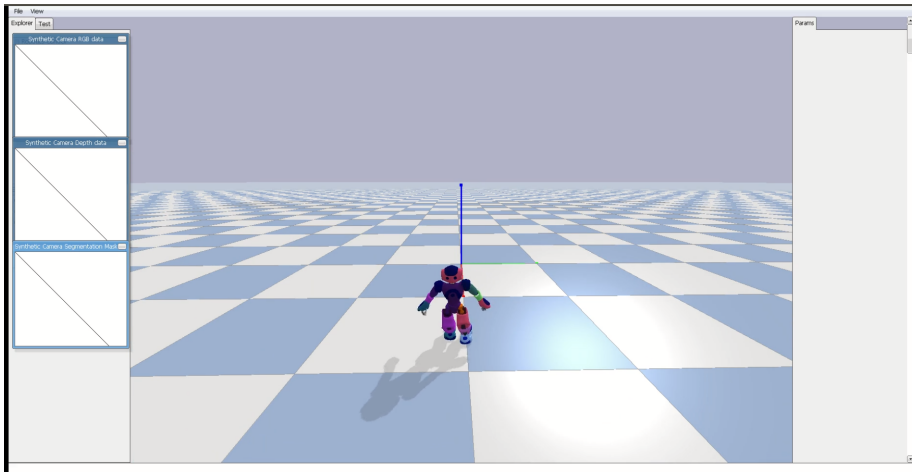

Fig. 3. Nao environment based on OpenAI gym standards

10 Future Work

We tried to implement various evolutionary algorithms to optimize low level skills. We also used the previously built python code base to achieve better results on similar tasks. However, the simulation server was not adequate for training skills using computationally expensive optimization algorithms.

Instead, we are working to build a reinforcement learning environment based on OpenAI gym standards, using a suitable physics engine,and robot models. We aim to train skills such as walking, kicking and standing up for a Nao-v40 bot. We intend to transfer the above results to the robosoccer game play to improve on-field performance. The aim is to replace the current inverse kinematic walk engine with a better holistic approach based on further work.

Further work also involves developing multi-agent coordination and strategies, using the above environment and other deep reinforcement learning algorithms. Having a general framework for multi-agent coordination is an essential part of a good soccer playing team of robots.

Furthermore we are aiming to use neural networks for automating the positioning module and being able to parallelize the algorithm for use on multi-core systems and considerably increasing the learning speed.

Acknowledgements

The team also acknowledges the mentorship and guidance of our professors Prof. Alok Kanti Deb, Prof. Sudeshna Sarkar, Prof. Jayanta Mukhopadhyay and Prof. Dilip Kumar Pratihar. This research is supported by the Centre for Excellence in Robotics, Indian Institue of Technology, Kharagpur. We also thank our former team members who made all of this possible.

References

1. Patrick MacAlpine and Peter Stone. "Prioritized Role Assignment for Marking." In Proceedings of the RoboCup International Symposium (RoboCup 2016) in Leipzig, Germany, July 2016.
2. Patrick MacAlpine, Eric Price, and Peter Stone. SCRAM: Scalable Collision-avoiding Role Assignment with Minimal-makespan for Formational Positioning. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI), January 2015.
3. MacAlpine, Patrick, Francisco Barrera, and Peter Stone. "Positioning to win: A dynamic role assignment and formation positioning system." RoboCup 2012: Robot Soccer World Cup XVI. Springer Berlin Heidelberg, 2013. 190-201.
4. Barrett, Samuel, et al. "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 (2012).
5. Jun, Youngbum, Robert Ellenburg, and Paul Oh. "From concept to realization: designing miniature humanoids for running." J. on Systemics, Cybernetics and Informatics 8.1 (2010): 8-13.
6. Erbatur, Kemalettin, and Okan Kurt. "Natural ZMP trajectories for biped robot reference generation." Industrial Electronics, IEEE Transactions on 56.3 (2009): 835-845.
7. Strom, Johannes, George Slavov, and Eric Chown. "Omnidirectional walking using zmp and preview control for the nao humanoid robot." RoboCup 2009: robot soccer world cup XIII. Springer Berlin Heidelberg, 2009. 378-389.
8. Liu, Juan, et al. "Apollo3D Team Discription Paper."
9. Akiyama, Hidehisa, and Itsuki Noda. "Multi-agent positioning mechanism in the dynamic environment." RoboCup 2007: Robot soccer world cup XI. Springer Berlin Heidelberg, 2007. 377-384.
10. H. Mania, A. Guy, and B. Recht. "Simple random search provides a competitive approach to reinforcement learning." arXiv:1803.07055, 2018.
11. T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971, 2015.