KgpKubs Team Description Paper Robocup 3D Simulation League 2022

Kushal Kedia, Sriyash Poddar, Abhishek Gandhi, Arpit Godghate, Parth Jindal, Pranshul Narang, Aryabhatta Aryan, Ayush Deep, Prerit Paliwal, Shivam Raj, and Shivanshu Gupta

Indian Institute of Technology, Kharagpur,
West Bengal, India
{pmjindal,narang.pranshul,abhigandhi29}@iitkgp.ac.in

Abstract. This paper reports the recent developments made by the Kgpkubs team. It describes the work on movement and kick improvement, passing strategies and setplays 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, 2018 and 2021 we participated in RoboCup (3D Simulation League). We also took part in the Robocup Asia Pacific 2017 3D Simulation League as well as the Offenburg Tournament 2020 3D Simulation League.

The paper is organized as follows. Section 2 provides a brief overview of the base architecture and strategy. Section 3 describes the Positioning Module. Section 4 describes the Role Assignment Algorithm. Section 5 describes the Communication module. Section 6 provides an overview of the passing strategy. Section 7 describes the approach and results obtained using CMAES as an optimization algorithm for getup, walk and kick parameters of the inverse kinematic engine. Section 8 describes our ongoing and future works.

2 Overview

The 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. Certain positioning requirements are also fulfilled by communication module. Some important roles like attacker, defender, goalie are assigned based on some heuristic methods overriding the Hungarian algorithm.

3 Positioning Module

Kgpkubs uses Voronoi-Cell Delaunay Triangulation method to generate and coordinate 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.

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.

The positions of the bots with respect to the Voronoi point is further overridden during the game play based on heuristics dependent upon various factors including ball position, bot positions etc. This helps in 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.

4 Role Assignment Module

The Hungarian algorithm is used which solves the role assignment problem in polynomial time. Time complexity of this algorithm is $O(n3)$. 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.

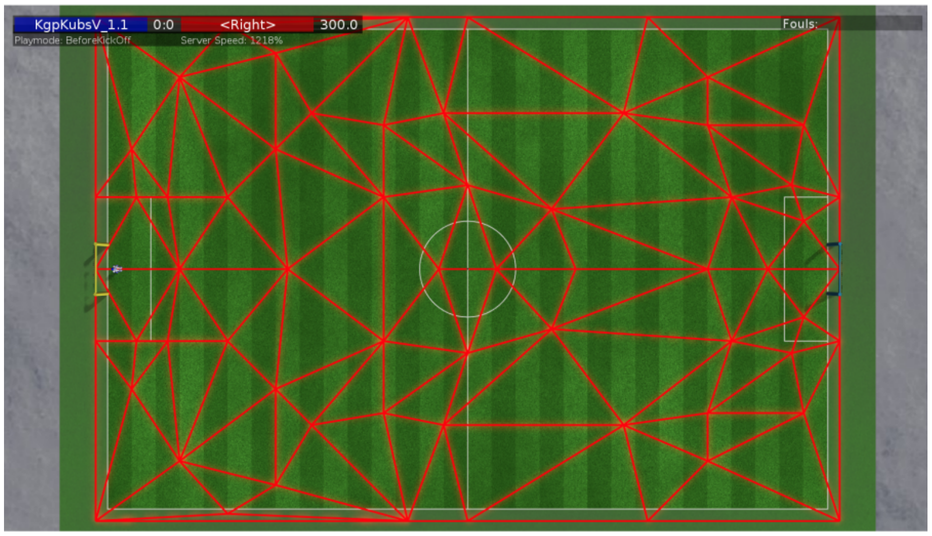

Fig. 1. Delaunay Triangles formed points

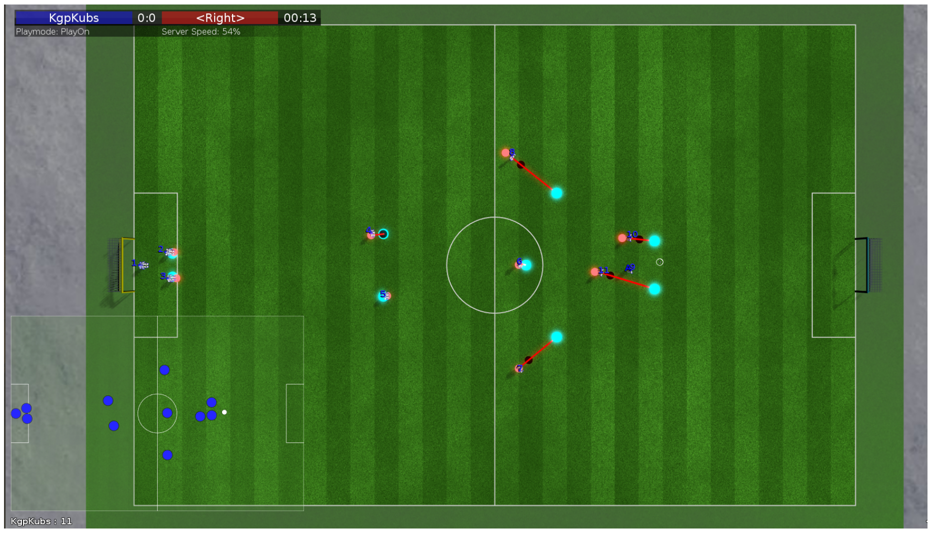

Fig. 2. Voronoi Points for a ball position

– To prevent erratic bot behaviour arising due to sudden changes in 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 :

1. Collisions are mostly avoided.
2. Longest distance is minimized.
3. 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.

5 Communication

The server sends noisy and restricted perceptual information once every 3 cycles (60 milliseconds).Inter-agent communication is used to add to each agent's knowledge about the world and improve decision making. The 3D simulator provides an "audio" channel for agents to communicate. An agent may broadcast such a SAY message once every 2 cycles (40 milliseconds); agents receive HEAR messages corresponding to all the SAY messages sent in the previous cycle.

Similar to the work in [14], we use the audio channel for a variety of purposes as shown in the table 1:

Information	Number of bits
Current server time in cycles	16
Ball last seen server time	16
Sender's perceived ball X coordinate	10
Sender's perceived ball Y coordinate	10
Sender's perceived self X coordinate	10
Sender's perceived self Y coordinate	10
Is the sender fallen?	1
X coordinate of target for communication pass	10
Y coordinate of target for communication pass	10
ID of player receiving the ball for communication pass	4

Table 1. Number of bits allocated to each piece of information communicated.

6 Tactics

6.1 Communication Pass

We develop a communication-based passing strategy i.e, two bots communicate with each other during game play for sending and receiving a pass if required. A bot can request another bot to position itself for receiving the ball. This helps us in passing the ball to various positions on the field even if a team-bot is not there at the moment to receive the ball. The received information from the server for ball-pass is updated in every bot's world-model.

6.2 Empty Pass

Passing accuracy is a key factor while keeping possession in a game. We use a general strategy in identifying potentially empty regions, use communication pass to request another team-bot to prepare itself for receiving it and pass the ball to the identified target.

In cases when there are a lot of opponents around the bot or due to noise in environment data, the potential empty region targets may fluctuate. To avoid this, exponential weighted average is applied on the calculated target over a few time frames. We calculate the final target as the exponential weighted average of all the fluctuating targets.

$$\textbf{finalTarget} = \textbf{finalTarget * } \beta + \textbf{newTarget * (1 - } \beta), \quad \beta \in \textbf{[0, 1]} \tag{1}$$

6.3 Set-Plays

Set-plays can always be a turning point in the match. Hence, correct bot formation and efficient passing is very important. This year we focus on attacking set-plays with the aim being to pass the ball to a bot that can score. We specify the corner kick set-play as an example.

Corner Kick: The algorithm chooses 5 strategic points near the goalpost. The aim is to pass the ball to one of the strategic points from where the chance of scoring is maximized. We calculate the scoring chance as a heuristic based on nearby opponent positions, how much of the goal is open for scoring and distance of player to goalpost.

The default formation specified in Voronoi was not very effective for corner kick setplay, as we required more bots near goalpost for executing our strategy. So, to keep the bots centered around these strategic points, We design new formations specific to our strategy.
The use of ***empty pass*** along with ***communication pass*** strategy to decide the passing target ensures optimal use of gaps between opponent bots further aiding in the set-play. The strategy focuses on avoiding unnecessary shots at target and will allow the bot to kick only when there is a high probability of scoring.

7 CMA-ES Training

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

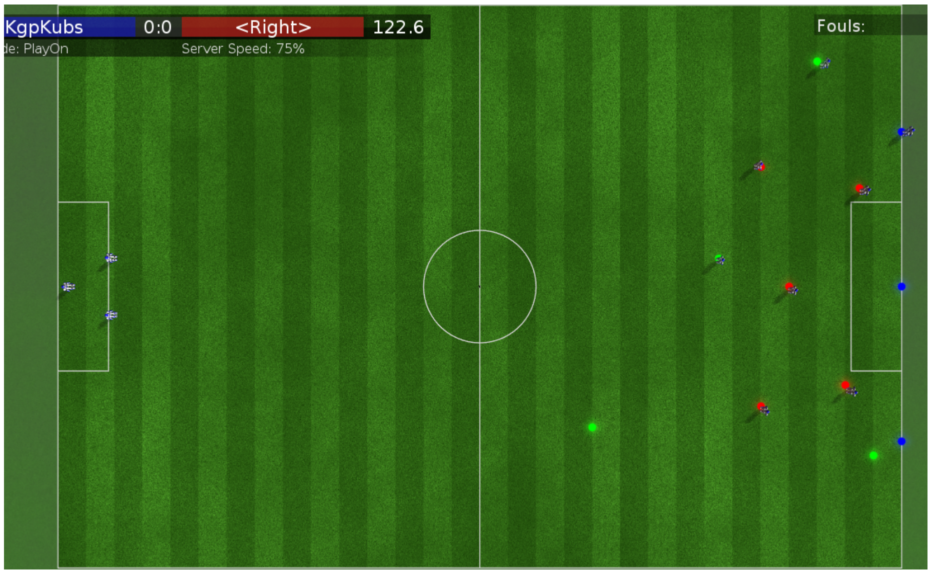

Fig. 3. Corner Kick Formation
Blue dots: corner kick positions, Red dots: 5 strategic points
Green dots: other formation points

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.

For optimizing low level skills 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.

7.1 Primitive Getup Training

The initial getup time from the front and back position was greater than 5 seconds. The new parameters trained for getup significantly improves the getup time while at the same time ensuring the stability of the bot. In the training drill, the bot is forced to fall and the reward is calculated as a function of time it takes to get up. We also account for COM's velocity after standing up in the reward up until it reduces to a threshold. The new trained getup parameters reduces the getup time to 2.2 seconds and 1.6 seconds for getup front and back respectively.

$$fitness_{getup} = -(getuptime + \alpha \cdot (\Sigma_{t=0}^{1} |v_x^t|)) \tag{2}$$

7.2 Walk Training

Normal Walk Training: The walk that we were initially using is clocked at an average of around 0.67 m/s. Although we reached speeds higher than this through running different drills, but we were not able to include them in the actual game as they were very unstable when transitioning to kicking and dribbling. This year we followed the Overlapping Layered Learning technique [12]. With the initial seed being previous year's walk parameters, The newly trained walk clocks at around 0.76-0.77 m/s.

$$fitness_{walk} = (distance_{covered} * time_{alloted}/time_{taken}) - distance_{overshoot} \quad (3)$$

$$fitness_{walk} -= 5 * times_{fallen} \quad (4)$$

Here is a video showing the drill which we used. In totality, 25 parameters were trained for the Omni-directional Walk Engine.The training was done for around 1000 generations.

Sprint Walk Training: Similar to the work in [12] as well as [13], A completely different set of parameters used just for sprinting are trained which the bot uses when the target is in a range of 15 degrees from it's own orientation. Drills and rewards were similar to Normal Walk training and the seed was trained Normal Walk parameters. Training was performed for a total of 200 generations. With the newly-included sprint parameters, the max speed achieved was around 0.84-0.85 m/s.

7.3 Getup Behavior

Once the new walk is trained, Getup parameters are once again optimized for improving compatibility with the newly trained walk. In training, the bot is forced to fall, getup and walk for 5 meters.The new parameters as a result of this drill increased the Getup time slightly i.e. for front from 2.2 to 2.5 and for back from 1.6 to 1.8 but with better compatibility and stability.

$$fitness_{getup} = -(getuptime_{initial} + (getuptime + 10)_{SubsequentFalls}) \quad (5)$$

7.4 Kick Training

Inverse-Kinematics Kick Training: The initial kick distance using an Inverse Kinematics kick engine was 3m-4m.Using CMAES, the kick engine's parameters are optimized with the new kick distance being 6m. Two different reward function are used in training, The first reward function simply accounts for distance travelled and discounts for any deviation from a target direction similar to the one in [15].

$$distance\_reward = (distTravelledForward) * e^{-(angleOffSet)^2/360} \quad (6)$$

The second reward function is for reducing kick time:

$$fitness_{IK\_Kick} = \begin{cases} -1 & : AgentFall \\ e^{distance\_reward} - approachTime & : otherwise \end{cases}$$

Here is a video showing the previous inverse kinematic kick while, here is a video showing the currently trained inverse kinematic kick.

8 Ongoing and Future Works

Having used various black-box optimization techniques such as CMAES to optimize low level skills such as walk and kick, We took inspiration from the [16] to build a reinforcement learning environment based on OpenAI standards for training directly on the simspark simulator rather than on a different physics simulator and transferring the learned skills to the simspark simulator. We aim to train skills such as walking, kicking and standing up for a Nao-v40 bot. The aim is to replace the current inverse kinematic walk engine with a better holistic approach based on further work. The environment sits right on top of the open-sourced UT codebase release: `https://github.com/LARG/utaustinvilla3d/` and can be used by any other team who have their codebase built on top of it. We hope to open source the gym environment in near future.

9 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 Collisionavoiding 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.
12. P. MacAlpine, P. Stone: Overlapping Layered Learning, in:Artificial Intelligence 254 (2018).
13. P. MacAlpine, S. Barrett, D. Urieli, V. Vu, P. Stone: 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), 2012.
14. Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Shivaram Kalyanakrishnan, Francisco Barrera, Adrian Lopez-Mobilia, Nicolae Stiurca, Victor Vu, Peter Stone: UT Austin Villa 2011: 3D Simulation Team Report
15. Patrick MacAlpine and Peter Stone: UT Austin Villa: RoboCup 2017 3D Simulation League Competition and Technical Challenged Champions.
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: 19th IEEE International Conference on Autonomous Robot Systems and Competitions (IEEE ICARSC’2019), Gondomar, Porto, Portugal, April 24-26 (2019)