KgpKubs Team Description Paper
RoboCup Soccer Simulation 3D 2023

Sriyash Poddar, Abhishek Gandhi, Parth Jindal, Pranshul Narang, Aryabhatta Aryan, Ayush Deep, Prerit Paliwal, Shivam Raj, Shivanshu Gupta, Nidhi Nandwani, Ronit Nanwani, Smarak Kanjilal, and Yelisetty Karthikeya S M

Indian Institute of Technology, Kharagpur,
West Bengal, India
{aryabhattaaryan,ayushdeep2001,prerit.paliwal,shivsrj2580}@kgpian.iitkgp.ac.in

Abstract. This paper reports the recent developments made by the KgpKubs team. It describes the work on new attacking and defence strategies implemented and training using reinforcement-learning.

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, 2021 and 2022 we participated in RoboCup (3D Simulation League). We also took part in the Robocup Asia Pacific 2017 3D Simulation League, the Offenburg Tournament 2020 3D Simulation League and RoboCunp Brazil Open league 2022.

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 details the Gym-integration process used to train the bots using reinforcement learning. Section 9 describes our ongoing and future works.

2 Overview

The UT-AustinVilla team's code, which is accessible on Github at https://github.com/LARG/utaustinvilla3d/, serves as the foundation for this architecture. We have the freedom to simply adjust and develop thanks to the code's division into suitable modules.

Our approach is based on combining Delaunay Triangulation for robot location on the field with tactic assignment for accomplishing particular tasks. For positioning, each robot employs Delaunay Triangulation to determine the positions of all other robots before utilising the Hungarian Algorithm to determine the positions of each individual robot. The communication module also satisfies some positional needs. The Hungarian algorithm is not used when assigning several crucial roles, such as attacker, defender, and goalkeeper.

3 Positioning Module

KgpKubs uses Voronoi-Cell Delaunay Triangulation method to generate and coordinate player positions with respect to the varying circumstances. The division of the space into discrete areas according to their separation from the focal point produces Voronoi cells.

Delaunay triangulation is the Dual graph of Voronoi cell plane. It guarantees that no other focal point is located inside the Delaunay triangle's produced ring. Also, because of this characteristic, it avoids narrow triangles. As a result, interpolating any point inside the triangle results in a continuous equation with a smooth gradient in terms of the coordinates of the triangle's vertices.

A data set of agent positions with regard to specific ball positions is produced by the algorithm used to construct player positions using statistical data (bot and ball positions under various game situations). To create Delaunay triangles, 65 ball positions in key areas were found and triangulated using the incremental approach. The Gouraud Shading algorithm calculates the value of bot positions at every given place in terms of the values of bot positions stored at the vertices of the triangles that surround it after the triangles have been produced.

During game play, heuristics depending on numerous criteria, such as ball position and bot positions, override the positions of the bots with regard to the Voronoi point. This contributes significantly to the bots' dynamic location. 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 role assignment problem is solved in polynomial time using the Hungarian algorithm. This algorithm's time complexity is $O(n^3)$. A set of target points are obtained through Voronoi triangulation at intervals of one second based on the position of the ball. 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.

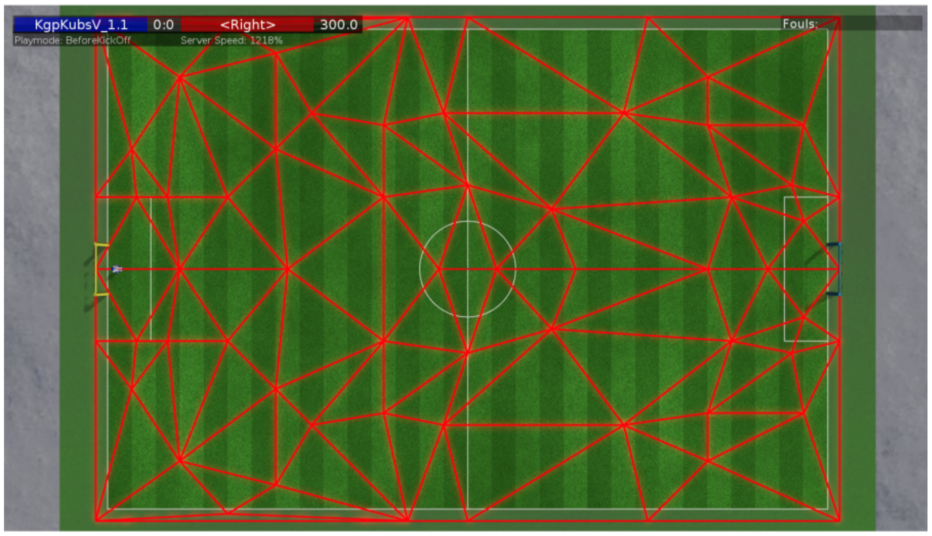

Fig. 1. Delaunay Triangles formed points

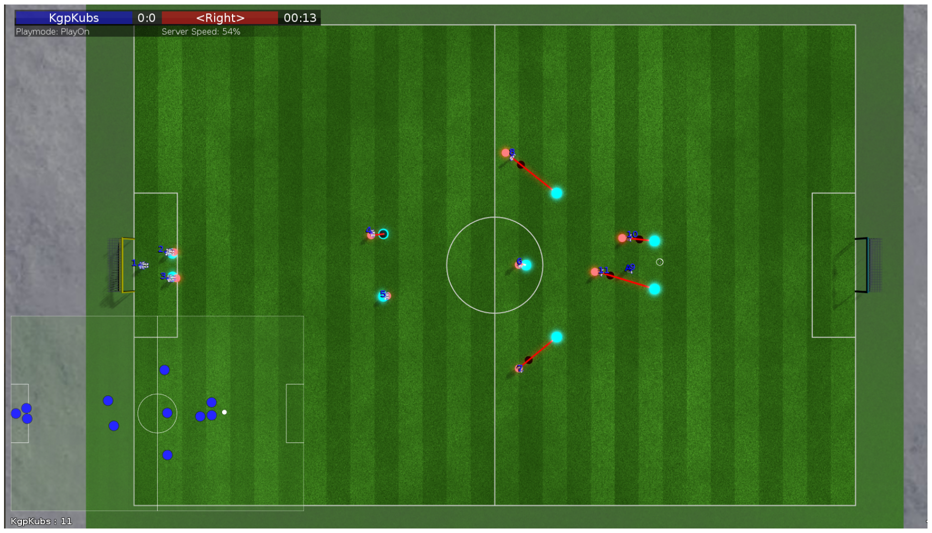

Fig. 2. Voronoi Points for a ball position

– To prevent erratic bot behaviour arising due to sudden changes in ball position.

The Hungarian algorithm then matches this set of target points to the players on the field. The euclidean distance between the bot's present location and the desired location serves as the cost function for the Hungarian algorithm. 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.

Now, when selecting an attacker, the angle at which the bot must rotate to go towards the goal is also taken into account. This facilitates a better choice of attacker since it addresses the issue of the bot's prolonged rotation time prior to dribbling the ball forward.

5 Communication

The server sends noisy and restricted perceptual information once every three cycles (60 milliseconds). Inter-agent communication is used to add to each agent's knowledge about the world and improve decision making.

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
State of FSM	4

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

5.1 FSM

The states in the FSM updates the flags which are maintained to choose the strategy is to be executed. 4 to 5 states corresponds to Kick-Off, Corner-Kick, Attacking strategy and others. All the bots are in a certain state, and when the threshold conditions are met, the updated flag is communicated with all the teammates and bots transition to the next state.
For example: During kickOff, the information whether the first pass is completed or not is determined by a transition from one state to another in the FSM, and hence we can communicate among the bots.

6 Tactics

6.1 Attacking

At the same ball position, we have both attacking and defensive strategy, which are dynamically selected based on the position of all the bots. Five bots form an attacking formation in the opponent half. The bot in possession of the ball optimally chooses a point near any of the five attacking bots and makes a pass. The pass target is chosen greedily based on factors like distance from goal and field of view towards goal.

Set-plays 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 Kick-Off, Kick-In and Corner Kick set-play as an example.

- **Kick-Off:** The play during kick-off is very crucial to get attacking advantage at start of game. The goal scoring chances can be increased exponentially via movement of bots to good attacking positions. In our strategy, we attempt to do the same by an initial pass made to a closest player in the Before-Kickoff Formation. Three bots closest to the mid-line are sent to dynamically calculated goal-scoring positions in opponent half. The bot in possession chooses optimal point to pass the ball towards the attacking bot. This increases the probability of scoring as the bots are shooting at a closer distance from the goal.
- **Kick-In:** Based on a threshold value and the ball position, a possible set of bots are chosen to which making a pass is feasible. From this set, a bot is optimally chosen such that the x-coordinate of the bot is as far as possible in the opponent half. As the Kick-In mode allows a free pass, passing the ball deep in the opponent half enables us to take advantage of the free space and increases the chance of scoring a goal.
- **Corner Kick:** The algorithm chooses five 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. This avoids unnecessary shots and increases the chance of scoring a goal.

The default formation specified in Voronoi was ineffective for implementing complex strategies like this in different setplays. This required a different and more attacking positioning; thus, we designed separate positioning modules which are activated based on the strategy in use.
All these setplay strategies use an informed passing mechanism to ensure optimal use of gaps between opponent bots, wherein the target is accordingly modified and sent to the receiving bot. Thus, maximizing our chances of successful pass and receive.

6.2 Defence

- **Man-marking strategy:** When the opponent team bot enters the defence area to align itself to the goal, it gets easy access to make a goal. During this time, employing a man-marking approach may be useful. When the possession of the ball is with the opponent, the closest opponent to the goal is chosen to be man-marked. A team player closest to that opponent is sent at a distance of 1m from the opponent on the line connecting the opponent and the goal centre. The strategy focuses on blocking the clear field of the opponent to the goal.
- **Defensive clearance:** When the ball is in our half, and our bots are outnumbered, the *passmode* function is called which restricts the opponents at a distance of 1m for 10 seconds, giving us enough time to kick the ball in an empty area.

7 CMA-ES Training

Without proper Reinforcement Learning Architecture, we fine-tuned some of our skills on CMA-ES in continuation to the training we had done last year. We were able to stabilize several of our skills, including front getup, back getup and sprint. Below we have provided the details of each of the skill training.

Skill	Old results	New Results
Get-up front	time = 2.2s	time = 2.12s
Get-up back	time = 1.6s	time = 1.53s
Normal walk	speed = 0.76m/s	speed = 0.79m/s
Sprint-walk	max-speed = 0.84m/s	max-speed = 0.88m/s
Inverse-kinematics kick	distance = 3.5m	distance = 7m
Framed Long kick	distance = 12m	distance = 16m

7.1 Sprint Training

The initial sprint parameters would make the bot unstable several times during the match. We removed some of the redundant drills and added a few drills which simulated real-time match scenarios in a better way. The sprint training was done on the previous normal walk parameters. The following is the reward function that we used:

$$fitness_{walk} = (distance_{covered} * time_{alloted}/time_{taken}) - distance_{overshoot} \quad (1)$$

$$fitness_{walk} -= 5 * times_{fallen} \quad (2)$$

7.2 Getup Behavior Training

We observed that during the match the bot would swing back and forth after getting up which resulted in an unstable getup. We trained the bot on the initial seed itself with changed walk parameters. The drill was changed such that the bot would have to getup and walk for a duration before falling again. In case the bot fell, it was heavily penalised. The following is the reward function we used:

$$fitness_{getup} = -(getuptime_{initial} + (getuptime + 10)_{SubsequentFalls}) \quad (3)$$

7.3 Position to Dribble Training

When the bot is close to the ball, it has to position itself and align with the ball in order to dribble it towards the goal. This positioning was initially done by implementing a sidewalk. The strategy was changed to positioning the bot around the ball using omni-directional walk itself, wherein the bot would move at right angles to an imaginary circle formed around the ball. The training was done on trained sprint walk parameters. The following is the reward function we used:

$$fitness_{dribble} = (distance_{covered} * time_{alloted}/time_{taken}) - distance_{overshoot} \quad (4)$$

$$fitness_{dribble} *= exp^{-6.0*(deflectionfromtarget)} \quad (5)$$

8 Ongoing work

8.1 OpenAI Based RL Gym

The RCSSSERVER3D simulator with NAO agents can be used in conjunction with the BahiaRT-GYM (https://bitbucket.org/bahiart3d/bahiart-gym/) toolbox for creating OpenAI-Gym environments. In order to train our bots using Reinforcement Learning instead of CMA-ES, which has been formerly employed, we built upon the work in [17].

The agent, connects to the proxy server instead of rcssserver3d directly. In the codebase, a new TCP-Socket connection has been made to listen to the GYM. Through this socket, uNum of the agent is sent for the GYM to detect and proceed further. The agent performs the steps given below in a cycle:

- It receives the message from the Proxy Server
- It waits for the GYM to send certain instructions in the form of message
- It deciphers the message, performs the required actions or modifications
- It sends the data back to Proxy server.

This process is repeatedly performed and trained for many generations under certain RL Policy.

8.2 Get-up Environment

We designed and implemented a custom OpenAI Gym Environment to facilitate the application of RL techniques to develop a robust getup behaviour. The **action space** is defined by the 22 joint angles that controls movement of bot. The action space is constrained according to physical limits of the bot to avoid self-collision/incapability issues during training. The **observation space** comprises 28 variables, including joint angles, acceleration and gyro rates along all 3 axes. These variables provide ample information about state of the bot essential for learning. The **maximum episode time** is fine-tuned to ensure learning process is efficient and do not take unreasonable amount of time. The **reset** function is overloaded to pass random joint angles through the proxy server to cause the bot to fall. The drill starts only after the bot has fallen completely, thus ensuring that environment is in consistent state at start of each episode. **Reward** is calculated at each timestep to avoid sparse reward conditions. The reward function encourages the bot to stand up stably and maintain balance throughout the training process. The equation for the reward function is:

$$Reward = \alpha * com_z - \beta * (accel_Z - 9.81) - \gamma * (accel_X + accel_Y)$$

where com_z represents the height of the bot's centre of mass along the Z-axis, $accel_Z$ is the acceleration of the bot along the Z-axis. The hyperparameters $\alpha, \beta, and\ \gamma$ are weighting factors that control the contribution of each term in the reward function to the overall reward. The three terms encourage the bot to maintain stable posture while standing up, rewarding it for keeping the centre of mass at a certain height. At the same time, the bot is penalised for accelerating too much along any axis, which could lead to instability and falling over. The rewards (+100/-10) given at end of each drill informs the RL policy about success and failure of bot's behaviour. The relative values ,timing and frequency of rewards are carefully calibrated to ensure that the bot is learning the desired low-level skills.

9 Future Works

We have already created custom environment for primitive getup training using reinforcement learning. Our aim is to train other low-level skills like walk, dribble and kick using RL policies to ensure that the bot learns to perform well even in a stochastic environment. The current inverse kinematic walk engine is intended to be replaced with a better holistic approach based on future work. We further aim to implement distributed coordinated strategies to decrease the complexity faced in a multi-robot systems. 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.

10 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)
17. Simões, M.A.C., Mascarenhas, G., Fonseca, R., dos Santos, V.M.P., Mascarenhas, F., Nogueira, T., 2022. BahiaRT Setplays Collecting Toolkit and BahiaRT Gym. Software Impacts 14, 100401.