Team_GJH

Zvono Bednarcik, Hesham Alhuraibi, Filip Gondar, Jakub Gregus,
Gymnázium Jura Hronca,
Slovakia,
https://www.gjh.sk

Team Description Paper

Abstract. Our 3 robots use dynamic role changing strategies based on the assumption that all data received by the robots is the same. A linear model is used for the ball, which assumes elastic collisions. A linear model is used to interpret the movement abilities of the robots. All robots estimate optimum intercepts for ball interception based on these presumptions.
Based on the intercept data, roles are split into an Attacker, Backup and Goalie. The Attacker moves to the intercept in a calculated parabolic fashion, the Goalie mirrors the movement of the ball along the y axis. The Support positions himself in a support position. All calculations and estimations are done in a normalized, two dimensional cartesian space, resembling the field. The robots have developed complex behavior, and statistically outperform random agents, and previous versions.

1.Introduction

We are Team_GJH, a small student lead robotics team that has been attending Robocup Junior Soccer competitions for years. We operate from Gymnazium Jura Hronca, a Slovak high school. We have mainly halted the development of team robotics projects, mainly because of how difficult meeting up was due to COVID-19 restrictions.

Therefore, we were very happy when the software-only, soccer sim challenge was announced. We put all our resources and time to operate within COVID-19 restrictions and produce a working product that's able to complete its task. There were several difficulties and obstacles, however, our passion kept on motivating us to push forward and achieve our goals.

After the Slovak rounds we analyzed how our robots operated and noted down several ‘ToDo’ improvements. After hard work, we refined our GoTo function, it allows our robot to turn and move faster (within the speed limits of course) by moving bidirectionally. We improved our backup robot, if the ball is not moving the robot will detect that issue and position itself within a close approximate proximity to the new spawn of the ball. This allows us to score more goals and keep our goal secure. Our biggest issue was our own goals, in an attempt to reach the ball hastily our robots would sometimes score in our side. We managed to avoid our own goaling by making our robots calculate whether they will hit the ball in a way that causes it to go to our own goal. If so, they dodge the ball.

2. Observations about the environment

Just from observing simple "stock" ball following code that was upped to full speed, our team came to a couple of observations, from which we drew conclusions that were vital to the functionality of our robots:

- The robots are **slow,** and will spend the vast majority of the time not in contact with the ball (rather chasing it). Positioning is therefore key.
- Cluttering the ball (multiple robots chasing the ball) has no visible advantage and seems to just result in a mess. Robots are better used covering more area. This way they chase the ball less and have more shooting opportunities.
- There is no clear-cut obvious strategy for simple goal scoring, unlike in omni-wheel robots, which can just keep pointing towards the opponent’s ball and follow the ball. Scoring goals with 2-wheel robots will be relatively complex.

- Most goals are contactless (not pushed into the goal). A lot of goals arise from random long-range ricochets which move faster than the robots. Hence having some sort of a goalie makes a lot of sense.

3.Roles

As a team we very quickly agreed on some role system. Here are our Roles, and their respective functions:

- Attacker: somehow scores goals.
- Goalie: Occupies space near goal.
- Backup: somehow covers as much space on the field to maximize ball possession.
- (Rejected Idea) Chuck Norris: As a substitute to backup, if the enemy has a robot near the goal, he attempts to push their goalie into the enemy defense zone to get him de-spawned to ease the job for the attacker.
  However, testing showed that this simply is not possible. The robots have equal push power which will result in a stalemate even if he does get in a position where he can push the enemy robot. Most of the time, the enemy robot simply slips away as it is pushed. Additionally, the robot clusters the defense area , often performing saves for the enemy team.

For optimum play, roles should be switched dynamically. Initially we wanted to switch roles based on who was nearest to the ball. But soon we switched to finding out who would take the shortest to intercept the ball. This way a robot who is closest to the ball but still travelling slower than the ball and henceforth pointlessly chasing it would be substituted by a better positioned robot. However, for this we need to be capable of calculating the Optimum intercept.

4.Intercept Calculation

It assumes that the ball travel is perfectly linear (true) and that collisions are perfectly elastic (false, but a sufficient approximation). The class has a buffer of past ball positions and using that it estimates the equation of the ball's line travel or its velocity vector, by forming a vector equation of a line along the ball.

Since the distance, the robot can travel per certain time can be approximated linearly (albeit imperfectly). We simply plot future ball positions and wait till one is close enough to our robot to be reached in said time it will take for the

ball to get there. This equation can only be solved graphically especially when factoring in the collisions. Collisions are solved by inverting one of the components of the ball's velocity vector.

Based on the intercepts of all robots (each robot calculating intercepts for each robot). The robot with the fastest intercept becomes the attacker. The robot closest to our goal becomes the goalie. Remaining robot is automatically backed up.

5. Movement Function

Our movement function, “GoTo” function receives the desired coordinates to reach and the current coordinates of the robot. With the help of basic trigonometric calculations, the function returns the angle to the desired coordinates, which means by how many degrees our robot needs to turn to face our desired spot, and it also provides the distance between our current spot and the desired spot. Depending on the angle and distances, we set the speed of the wheels so that our robot is able to move to its destination. The GoTo function is used all the time, with its help we can reach our optimal position as fast as possible and intercept the ball in a way which would help us score.

6. Goalie

The goalies’ main functions is protecting the goal and later blocking the ball in the corner to allow a respawn of the ball. The blocking of the ball is done simply by selecting an x-axis coordinate which is already predetermined based on our team color. The y axis position is based on calculating the ratio of the ball's position to the goal line. There were many trials and errors to fine tune this process but in the end, we managed to get it to track the ball's position perfectly. The other main function is getting the ball into the corner. For this we have a predetermined set line which once the ball crosses the function is called. Firstly, the robot makes his way to the goal post to prevent a goal with a set coordinate and then we set the desired ball positions to the ball positions so that the ball in the end gets stuck in the corner and it respawns give as an opening at the opponent’s goal.

7. Attacker

After expenting with different linear approaches that all seemed to fail, we concluded that the best function that fits the movement of the robot such that he hits the ball at an angle that sends the ball to the enemy goal is a quadratic function. Taking the known future intercept of the ball our robot calculates every loop, it must find a quadratic function that passes through the intercept and the robot. However since quadratic functions have 3 constants we have to find, we also need a third data point to solve our equation for a,b,c where the quadratic function is $f(x) = ax^2 + bx + c$. The third data point is that the slope of the line at the point of intersection must be such that it intersects with the ball.

Using the derivative $f'(x) = 2ax + b$ we can get a third equation. This gives us 3 equations which we can then solve using Gauss-Jordanian Elimination.

If the parabola passes through the bounds of the field, we ignore it and travel linearly. If the ball is behind the future intercept, we also ignore the parabola and travel linearly to our intercept point. To make sure we approach the ball from behind (such that we are between the ball and our goal), we chase our intercept with lead (hence the robot overcompensates and travels to a position behind the ball rather than to the ball).

If the outcome of this decision making is to follow the parabola, the robot determines the derivative of the parabola at its location. The derivative is used to find the equation on the tangent line. A point on the tangent line is chosen, and the robot sets that point as its target movement point.

8. Support

Support started out as the simplest script. His position was just a function of the positions of his two teammates. He was positioned such that he mirrored the attacking robot on the Y axis and was in between the attacker and the goalie on the X axis. We also considered sending him forward more, so that he can be closer to the attack. While this would improve our attack, it also led to us losing control of the ball more often and much more ball chasing. In the end we opted for the more passive playstyle.

One opportunity which we later realized we could capitalize on is the predictability of the lack of progress respawn mechanic. Using the same algorithm as used in the main environment, that being summing the change in movement

over the last X instances, we could easily detect a lack of progress. Depending on which half the lack of progress takes place, our support robot either takes up an offensive or defensive position near the center. This advantageous positioning is a source of many goals.

9.Conclusion

Considering no previous history of the competition, we have managed to create an AI that is capable of outperforming random agents, primitive AIs and even past versions of itself. Our approach shows a lot of promise but also certain limitations. Certain premises, like the elasticity of collisions, that a lot of our decision making is based on, or the range approximation are simply not true. Certain processes are unoptimized, and there is still a bug that can temporarily immobilize a robot in niche circumstances.

Despite this, our upgrades to the robot going into the Slovakian competition are huge. When simulating matches against random agents or simple AI, our robots score around 15-20 goals per half. Against our previous version, simulated 100 minute matches ended within the range of 22-30 (new robots) : 1-4 (old robots), demonstrating massive improvement.

We have also done research into reinforcement learning, even successfully converging models in 1v0 situations with crafted rewards. Due to limited time and processing power our robots are human coded for this year.