Autonomous vehicle collision avoidance and Soccer Games using Behaviour Trees

GROUP 1

Marco Schouten Justin Salér

1998-05-22 1992-12-07

schouten@kth.se justinr@kth.se

Abstract

We have investigated two multi-agent system scenarios, a collision-avoidance scenario with vehicles trying to avoid each other and a drone soccer scenario where we created the AI for a 3v3 soccer simulation. Collision Avoidance was tested on two different maps with different solutions: an open field and an intersection. For the open field, we used a dynamic dodging mechanism, whereas for the intersection we first found the shortest path between the start and the goal and then used a behaviour tree to avoid inter-vehicle collision. Our open field solution gave really good results. For the drone soccer, we also used behaviour trees, where we had three distinct behaviour trees for different roles; goalkeeper, defender and attacker. We also used a fourth behaviour tree running in parallel to the drone trees, which dynamically updated the drones' roles. We managed to get okay results, where the kicking accuracy was our greatest strength.

1 Introduction

It is a common belief and wish that autonomous vehicles will soon take over our streets. However, creating algorithms to control autonomous vehicle requires extensive research. To better understand collision avoidance for multi-agent systems, we have simulated a traffic situation, where vehicles crammed in a small area attempt manoeuvring collision-free. We have also investigated another multi-agent system, association football. In this scenario, we have created the logic for controlling a drone soccer team, which played against drones developed by other course attendants.

For the traffic problem, we used 50 vehicles on two different maps, one with an open field and one with an intersection. The two different maps came with very different issues. For the open field, the main task was dodging other vehicles, but for the intersection, the vehicles had to navigate without colliding with the walls, requiring the calculation of optimal paths and attempting to follow those. Our results for the open field scenario were very good but the intersection scenario solution would have required further development.

The soccer simulations were done with two teams, each with three drones, facing off against each other. The drones, when close enough, could kick the ball in any direction. This could be used for kicking towards the goal, but also for passing and to kick away the ball from the opponents. Similar AIs are used in football games such as FIFA, but our solution can also be extended to other team sports games. In extension to this, the simulations give a better understanding of multi-agent systems and could also be used by professional football players to find new tricks and strategies. The drones competed in two tournaments against drones controlled by algorithms created by other students. Our solution used behaviour trees, which is a state-machine-like approach for the decision-making of autonomous systems. We had three different roles with unique behaviour trees; a goalkeeper, an attacker and a defender. We also used a fourth behaviour tree, a manager, which dynamically gave the drones their roles.

1.1 Contribution

In this project, we implemented a variety of ad-hoc algorithms to deal with each problem optimally.

1. For the Collision Avoidance Problem we took a different approach according to the environment. For the intersection we used paths created with breadth first search and then used behaviour trees to avoid inter-vehicular collision. For the open field map, on the other hand, we

combined delayed input accelerations with a dodging mechanism that was activated whenever an object was detected within a predetermined range.

2. We solved the Drone Soccer with behaviour trees (BT). First, we designed a different BT for each role. Then we defined an external entity (the manager) that updated the drones' roles based on the state of the field.

1.2 Outline

In Section 2 we explore the state-of-the-art techniques and available literature for each problem to provide an overview of the pros and cons of each strategy to choose critically the most suitable method. In Section 3 we explain the details of our implementations. In Section 4 we analyse the results of our findings. In Section 5, we give a conclusion on our results and what could be improved.

2 Related work

Below here we investigate the available literature for each problem.

2.1 Collision Avoidance

Collision Avoidance is a flourishing area of research and has a variety of applications in the industry. There are countless scenarios in which different agents need to move smoothly at a given time and space (e.g. [Vrba et al., 2007]). There are various solutions to this problem, the most prominent:

1. *Proportional Navigation* is one of the most consolidated strategies dating back to [Adler, 1956]. Additionally Farwell's Rules for navigation [Allen, 2004] report that CBDR is "the time-honoured method for determining the spatial component of the risk of collision". Fundamentally it is based on the observation that two vehicles are on a collision course if their direct Line-of-Sight does not change direction as the range closes. In other words, the bearing angle remains constant [1]. The strength of this technique lies in needing only the bearing angle change, which can be easily estimated.

[1] The bearing angle is the tan-1(approaching vehicle speed/driver's vehicle speed)

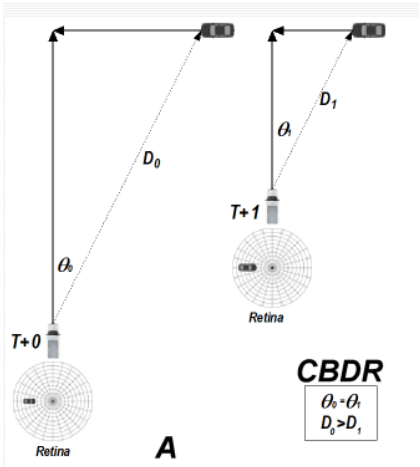


Figure 1: Constant bearing, decreasing range (CBDR) scheme for proportional navigation used for collision avoidance.

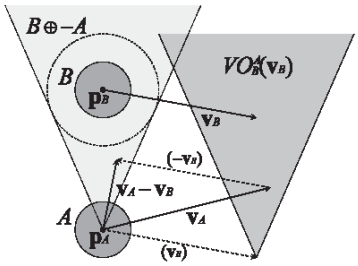


Figure 2: Reciprocal Velocity Obstacles scheme for disc-shaped obstacles

2. *Reciprocal Velocity Obstacles* proposed by [Van den Berg et al., 2008] tackle communication-free navigation for multiple agents. The greatest strengths of this idea are its scalability and its capability of working with both moving and stationary objects. It evaluates targets changes in position and velocity to build cone-shaped areas [2]. Hence, if the velocity of that object belongs to that area, it will lead to a collision. On this account, the navigator ensures a collision-free trajectory by applying controls that move the velocity out of those cone-shaped areas.

3. *Decentralised Collision Avoidance with Deep Reinforcement Learning* [Chen et al., 2017] proposed an innovative alternative solution to the state of the art reciprocal collision avoidance (ORCA) that harnesses

[2]Areas are cone-shaped since we are dealing with rigid bodies instead of point-like shapes.

the potential of deep learning. Retrieving optimal paths through a centralised path planning strategy often requires interaction between nearby vehicles. Note that for large scale problem instances this may be computationally prohibitive. The implementation of offline-deep learning computations yields a 26% increase in overall performance (time to reach the goal). The core idea is that by linking the speed and position of an agent and its neighbours with an estimated time to reach the goal, it is possible to create a value network that allows efficient queries to find collision-free velocity vectors.

2.2 Drone Soccer

Drone Soccer is a complex and multidisciplinary problem which requires to coordinate a set of drones to make them play a game of soccer.

- *Behaviour Trees* [Colledanchise and Ögren, 2018] are an efficient strategy that decomposes the behaviour of drones in a set of tasks organised in a tree-like structure. The core strength of BT is its modularity. It is easy to add new features to or to make changes to the behaviour of an agent. This is due to the structure of the tree that is traversed by Ticks [3]. For these reasons, BT is preferred for developing such AI compared to older approaches like Finite State Machines [Ho and Komura, 2011].

- *Integrating BT with Reinforcement Learning* [Kartasev, 2019] showed that is possible to translate BT conditions to discrete observations. In addition, it is possible to embed all-action nodes in the action space for RL. An interesting finding is that for small problem instance this translation is very efficient and learning may be obtained while preserving existing design.

- *Soccer Robot Applied in Robocup-MSL* [Kasaei et al., 2010] developed an AI for Soccer Games. They have broken down drones decision making according to a two-level hierarchy: low and high-level behaviour. Additionally, they investigated and described optimal tactics regarding the attack, defence and middle field to win the game.

[3] A Tick is a process of visiting the next node in the tree. Each Tick returns Success, Running or Failure

3 Proposed method

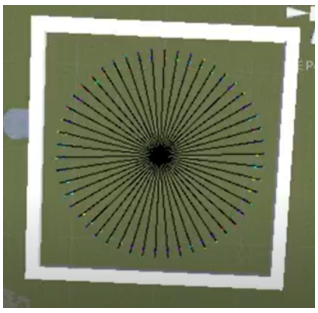
In this section, we are going to explain in details the algorithms we used to solve Collision Avoidance and Drone Soccer.

3.1 Collision Avoidance

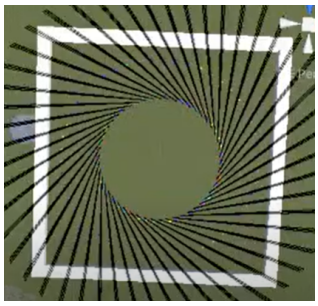
We investigated different approaches to have an optimal collision avoidance solution according to the type of terrain.

3.1.1 Open Field

Our solution is inspired by *Reciprocal Velocity Obstacles*. The strategy is to make each drone move towards the goal until another drone is close enough (parametrized by a threshold of 10.0 m). Once an obstacle is within the specified range, the drone activates a dodging mechanism to prevent collisions. To this end, it receives a radial acceleration proportional to its relative distance to the closest drone and orthogonal to its previous velocity. Hence, following a circular trajectory around the obstacle, it ensures a safe distance. The principle that makes this work is that drones update their velocity after a parametrized delay (0.010 s). In this way, the drones do not get stuck moving forever in a circle around each other. Additionally, to get better controls, the max velocity is capped around (15.0f m/s).



(a) Beginning, moving towards the goal



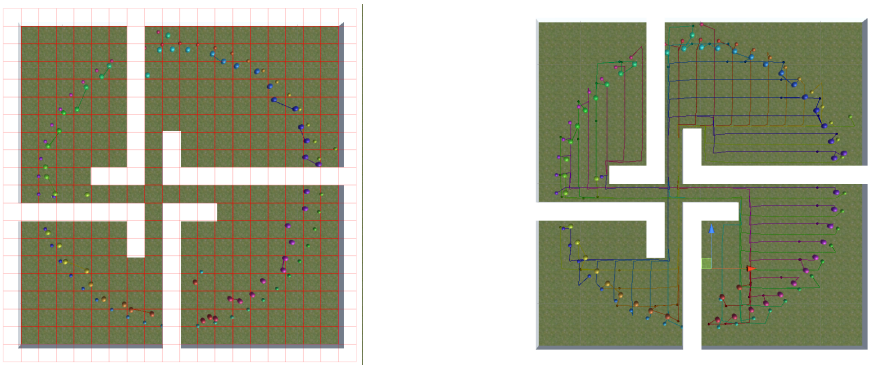
(b) Dodging (moving to the side)

(c) Dodging and moving towards the goal (moving towards the center and to the side) inside

Figure 3: Open field.

3.1.2 Intersection

For the intersection map, the drones moved on a right-hand-traffic track, and then used a behaviour trees solution to avoid other drones. As seen in figure 4, the terrain was divided into a $d \times d$ grid, and then breadth-first search was used to find the shortest path from each drone to it's goal. After finding the shortest path, the path was shifted $0.4d$ to the right. This way, the drones moved in a way similar to normal two-directional roads, without any risk of full-frontal collision. Each drone was also randomly assigned a priority to decide which of two drones would drive in unclear situations.



(a) Grid for creating the paths.

(b) Paths of each drone

Figure 4: For the intersection we used breadth first search to create a path for each drone.

Using the behaviour tree the drones could identify four different situations and act accordingly.

1. If there are no risks at colliding with other drones, continue on one's path.
2. If there is a high risk to collide with another drone, e.g. it is directly in front of the drone and the drones are moving in different directions, stop.
3. If there is a high risk to collide with another drone from behind, build a queue and follow it closely behind.
4. If there is a chance of two vehicles to collide, e.g. both are turning at the same time, the lower priority drone lets the higher priority drone through.

3.2 Drone Soccer

To solve the drone soccer we defined one behaviour tree for each role: Goalie, Defender, Attacker. Additionally, we designed a behaviour Tree for a manager, responsible for assigning roles to the drones given the game state.

Among the trees, a common condition was checking who had the ball to choose its action. We summarise the tasks for each role:

- Goalie: shoot the ball away, pass the ball, catch the ball.
- Defender: kick the ball to the side, a position as the last guard, intercept the ball, block an enemy attacker
- Attacker: kick the ball to score, position itself to receive the ball, move forward with the ball.

The most interesting task for the Goalie, was to determine whether it was worth to leave the goal to chase the ball. To this end we implemented a function that determines whether the goalie would reach the ball before the enemy drones. This was done by calculating the time it would take for the goalkeeper as well as the enemy drones to reach the ball. If the goalkeeper can reach the ball before the enemy drones, and the ball is not to close to our goal, the goalkeeper chases the ball to try and pass it to a teammate. The time to reach the ball was calculated as

$$t = \frac{\sqrt{2|\Delta\vec{X}|a_d + (\Delta V_x)^2} - \Delta V_x}{a_d}, \tag{1}$$

where $\Delta\vec{X}$ is the distance between the drone and the ball and $\Delta V_x = \frac{\Delta\vec{V}\dot{\Delta}\vec{X}}{|\Delta\vec{X}|}$ is the relative velocity in the direction of the ball. a_d is the acceleration of the drone in the direction of $\Delta\vec{X}$, with the assumption that the drone is accelerating directly towards the ball. This assumption is just an estimation if any of the drones considers $\Delta\vec{V}$ when chasing the ball, like e.g. a PID controller. To balance the assumption and make the goalkeeper less risk-taking, the goalkeeper's acceleration is assumed to be lower than the enemies.

The Defender features were inspired by real football tactics. One was to position itself as the last guard, i.e. it must be closer to its door than all enemy attackers. Another one was to kick the ball to the side to remove the ball from the centre of the field. In this way, we managed to build an effective defence, as the enemies had fewer angles to score if the ball is to one side instead of being at the centre.

Although our attackers did not have a passing game, they had two strengths. Firstly, they were good at chasing the ball and then slowly moving forward

with it towards the goal. Secondly, their kick had high accuracy: they did not shoot often, but whenever they could, they scored almost all the time. We developed a "kick to position" function, which added the force to the ball required to make it follow the desired trajectory at maximum possible speed given the input force limitation. We retrieved the input kick direction in the following steps:

1. Compute θ: the angle between ball velocity and the desired direction.

$$\theta = \arccos(\frac{Dir \cdot V_{ball}}{||Dir||\ ||V_{ball}||})$$

2. Compute ϕ that balances the wrong direction of the ball while maximising the parallel component to the desired direction.

$$\begin{cases} |\text{Input Force}| \sin(\phi) = |V_{ball}| \sin(\theta) \\ \phi = \arcsin(\frac{|V_{ball}| \sin(\theta)}{|\text{Input Force}|}) \end{cases}$$

3. Get the parallel and perpendicular components that define the input Force w.r.t the desired direction.

$$\begin{cases} v_1 = |\text{Input Force}| * \cos(\phi) \\ v_2 = |\text{Input Force}| * \sin(\phi) \end{cases}$$

4. Get α, the angle between X axis and the desired direction (like [1]).

5. Rotate this vector to the XZ plane.

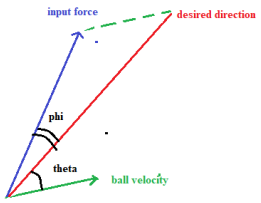
$$\begin{cases} X = v_1 \cos(\alpha) - v_2 \sin(\alpha) \\ Z = v_1 \sin(\alpha) + v_2 \cos(\alpha) \end{cases}$$

4 Experimental results

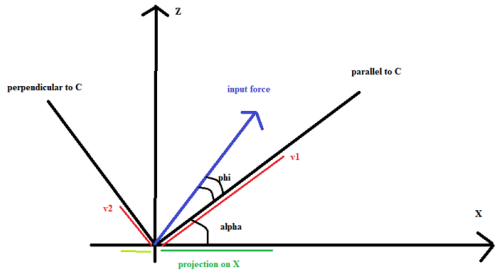
In this section, we will comment on the best results for each problem, and discuss how our methods can be improved.

4.1 Experimental setup

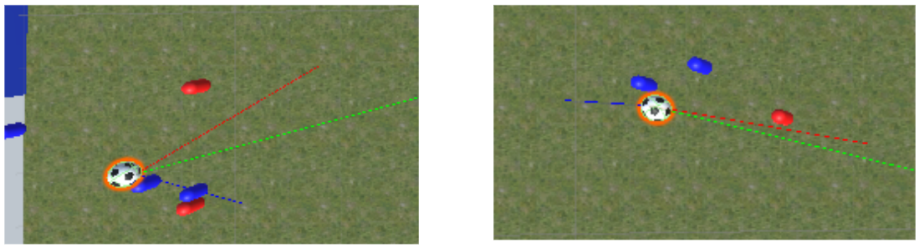
We have tested our code in a simulated UNITY 3D environment. Additionally we used PandaBT to implement the Behaviour Trees.



(a) Force scheme



(b) projecting parallel and perpendicular components w.r.t the desired direction to X and Z



(a) blue is ball velocity, red is input kick, green is desired direction

4.2 Analysis of Outcome

The results can be seen in table 1 and table 2. For the drone soccer we achieved adequate results, but for the open field map our solution really shined. When comparing with 13 other teams, our solution for the open field collision avoidance was the fastest. Many other teams used the same solution for both the maps, which might have led to a more rigid solution. But the main advantage of our solution is the lack of collisions, which may also explain the short time.

	Open Field	Intersection
1	**20s (G1)**	62s (G11)
2	29s (G10)	96s (G3)
3	33s (G11)	96s (G4)
Us	20s (1)	300s (7)

Table 1: Results for the collision avoidance problem on the two courses. The table shows the top five results for each terrain, as well as our time. The cell contains the time in second as well as the group name, except the last cell that contains our time and our position. For reference we are G1 (group 1).

	Wins	Draws	Losses	Score
1 (G2)	17	5	2	56
2 (G10)	18	1	5	55
3 (G13)	17	3	4	54
Us (6)	12	4	7	40

Table 2: Results for the drone soccer. The table shows the top five results, as well as our result. The score was calculated as 3 points for each win and 1 for each draw. For reference we are G1 i.e. group 1.

For the intersection map our solution was not fast enough. The advantages that our solution had, it's fluidity, is countered by too many way points that are required to pass by the drones. This could have been circumvented by adjusting the paths, but would but attempting to tune the hyper parameters to the adjustments didn't succeed in time. The best result for the intersection map was by Group 11, who used a combination of our approaches of the two problems, where they follow a path but move their drones to avoid other drones. Their solution has a lot more inter-drone collision than ours though.

In the drone soccer final tournament we won 12 matches, drew 4 matches and lost 7. When running locally, we only lost to two teams, group 9 and group 13. Group 9 had an innovative solution with three attacker, but it is possible it was a fluke, since we won further matches against them when testing. Group 13 was tougher, but that match was also very close, with a 1-0 goal from them in the last second. They had a strong goalie and their drones were good at blocking ours. Other teams that got a good total score, Group 2, Group 4 and Group 10, we easily beat when running locally (2-0 against Group 2, 3-0 against Group 4 and 3-1 against Group 10), so it is hard to determine what they did well. They all had a good passing game that could have been good against other teams, but proved futile against our accurate attackers.

5 Summary and Conclusions

In this project, we solved two problems for multi-agent systems in the UNITY environment: Collision Avoidance and Drone Soccer. We obtained the best results for the open field and good results for the Drone Soccer and intersection.

For the intersection, a lot of small changes could have been done that would probably be able to improve the time immensely. There is also the

possibility to borrow from the open field solution, have preset paths but then try to dodge other drones. One thing that we liked with our solutions for Collision Avoidance is that they were designed to make the drones reach their goal position given the hard constraint of avoiding collisions. There were a few collisions in the intersection due to bugs, but that could have been tuned away.

For a further improvement to Drone Soccer, we could improve our Soccer AI by adding a few tasks to the attacker, namely to improve the passing game strategy. Both attackers acting identically and not considering each other was the weakest point of our solution. The manager could also have been extended to not only update the roles, but to also add strategical directions based on the field. An example is which side to attack from.

References

[Adler, 1956] Adler, F. P. (1956). Missile guidance by three-dimensional proportional navigation. *Journal of Applied Physics*, 27(5):500–507.

[Allen, 2004] Allen, C. H. (2004). Farwell's rules of the nautical road. US Naval Institute.

[Chen et al., 2017] Chen, Y. F., Liu, M., Everett, M., and How, J. P. (2017). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE.

[Colledanchise and Ögren, 2018] Colledanchise, M. and Ögren, P. (2018). *Behavior trees in robotics and AI: An introduction*. CRC Press.

[Ho and Komura, 2011] Ho, E. S. and Komura, T. (2011). A finite state machine based on topology coordinates for wrestling games. *Computer Animation and Virtual Worlds*, 22(5):435–443.

[Kartasev, 2019] Kartasev, M. (2019). Integrating reinforcement learning into behavior trees by hierarchical composition.

[Kasaei et al., 2010] Kasaei, S. H. M., Kasaei, S. M. M., Kasaei, S. A. M., and Taheri, M. (2010). Design of an action selection mechanism for cooperative soccer robots based on fuzzy decision making algorithm. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience*, 1(3):5–18.

[Van den Berg et al., 2008] Van den Berg, J., Lin, M., and Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935. IEEE.

[Vrba et al., 2007] Vrba, P., Mařík, V., Přeučil, L., Kulich, M., and Šišlák, D. (2007). Collision avoidance algorithms: Multi-agent approach. In *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, pages 348–360. Springer.