Cooperative Pathfinding in 3D Robot Soccer

Matthew Rayermann, Team Moist Cake

CS 344M Final Paper

Abstract

Pathfinding in a single agent space is a relatively simple task. In order to reach a goal, a single agent only has to worry about static environmental obstacles while finding an optimal path. The task of effectively planning paths becomes significantly more complex in a multi agent system though, with the possibility for collisions to occur between agents, who may or may not also be moving. These collisions can lengthen the amount of time it takes an agent to reach their goal, or even prevent them from reaching it at all. In RoboCup 3D virtual soccer, such collisions frequently occur between the robot agents who inhabit the domain. These collisions not only decrease the agents' ability to reach their goals in a timely manner, but they also greatly reduce the effectiveness of their team as a whole. This paper seeks to examine and discuss the feasibility and effectiveness of implementing cooperative pathfinding within RoboCup in an attempt to mitigate some of the problems faced by agents as they navigate their virtual soccer field.

1 Introduction

RoboCup is an international effort towards the advancement of research in autonomous multi-agent systems through having teams of robots play soccer. RoboCup includes competitions in both physical and virtual domains, each with varying characteristics and challenges. The 3D virtual league pits teams of virtual robots against each other in a semi-realistic real time 3D soccer simulator. While watching games in the 3D league, it is hard to not notice the many times that agents collide with other agents on both the opposing team and their own team. The virtual robots' lack of dexterity, balance, and coordination means that any sort of collision that results in falling down could potentially be catastrophic. The act of standing up can take multiple tries, consume precious time, and cause an agent to miss out on various opportunities for action which could have repercussions on their team's chance at winning or losing.

Collisions between agents play a huge role in their team's performance in the RoboCup 3D virtual league. As such, a reduction in these collisions could potentially cause a team to have a higher chance at winning its games. This paper will present a cooperative pathfinding algorithm which a RoboCup team could use. It will then show the results of experiments which will help determine whether or not the algorithm actually reduces the number of collisions between teammates. Finally, it will discuss any opportunities for the continuation of the presented work, and any lessons learned after observing the final implementation in action.

2 Cooperation through Reservation

The key ingredient in any type of cooperative multi-agent system is the ability for agents to share information with other agents and access any information which is shared by other agents. For agents who are cooperatively pathfinding, they will probably want to share information about where they are going, how they are moving, or what they are able to observe. For the algorithm presented in this paper, agents will communicate where they are going and when they expect to be at those locations. This will be done so the agents can take advantage of what is commonly referred to as a reservation table (Silver 2005). This concept of a reservation table has been implemented in a variety of cooperative pathfinding domains. These domains range from simple to incredibly complex. An example of reservation tables being used in a complex domain can found in autonomous vehicle intersection management (Dresner and Stone 2008). This proven versatility of reservation tables are what motivated their use in this project.

The actual use of reservation tables is quite easy to understand. Agents share their path, both spatially and temporally, with the reservation table. Depending on the implementation and domain, the reservation table may either always accept the agent's reservation, or it may choose to either accept or reject the reservation based on some predetermined criteria. Agents are also able to read the information stored in the reservation table, which they can then use to aid themselves in planning paths which avoid colliding with other agents.

The RoboCup 3D simulation domain provides several challenges for the

implementation of a reservation table. First, most implementations of reservation tables choose to place the table in a single centralized location. This may be a data structure in some shared memory, or it may be some other agent whose task is to manage reservations and coordinate the movements of the other agents. In RoboCup, there is no notion of a centralized controller, leader agent, or data storage location that all agents on a team can instantly access. Additionally, the only option for sharing information between agents is through the simulator's agent to agent communication system. Thus, the inability to have a centralized reservation table will be overcome by replicating the reservation table inside of each agent. However, in order to maintain consistency, liveliness, and fairness, agents will take turns planning a path while taking into account the information stored in their reservation table. Once they have formulated a plan, they will then broadcast said plan to their teammates which will allow each teammate to update their reservation table accordingly.

3 An Algorithm for Cooperative Pathfinding

3.1 The Initial Plan

Similar to pathfinding in a single agent space, the algorithm begins with the execution of a separate algorithm which attempts to find the shortest path to the agent's goal. The A* (A-star) pathfinding algorithm was chosen for this project. An extension of the well-known Dijkstra's Algorithm, A* improves on the performance of its predecessor through the use of a heuristic to more

accurately order its priority queue. In the case of this project, octile distance was chosen as the heuristic since agents will be allowed to move diagonally. The A* search is also performed in three different dimensions, two spatial and one temporal. The spatial dimensions are the result of abstracting the soccer field into a 2D grid, and the temporal dimension is required by the algorithm's usage of a reservation table. While doing the initial search, the agent and its A* can use the reservation table to reject locations which are being reserved at the same time that the agent thinks it will be at that location. Once the A* has finished finding an optimal path, the agent will have an ordered list of locations to travel to, each of which has an estimated time of arrival.

3.2 Reservation Windows

With a completed path in hand the agent now has the information it needs to begin traversing its path, moving from one location to the next. Before they may move though, the agent must reserve the first window, or small chunk, of their path. The concept of reservation windows has been previously used by others in similar applications (Wang and Botea, 2008). To reserve a window, an agent must first check for any conflicting reservations in its reservation table. If none are found, it completes the reservation by broadcasting its window and all associated information to any friendly agents. Once the agent has reserved their window, they may begin moving along their path. At the end of the window, the process repeats with the agent attempting to reserve their next window and not moving again until they are able to do

so. If for some reason they are unable to reserve the window, either because another agent has already reserved some portion of the window or because it is not the agent's turn to speak, then they simply replan the path from their current position or wait for their turn.

The motivation behind reserving the path in a series of windows is twofold. First, it prevents one agent with a long path from forcing other agents to take roundabout paths based on some far out estimate. Second, the 3D simulator's communication system limits the length of messages that agents are allowed to send. By reserving only one small window at a time, all of the information needed to communicate the reservation is able to fit in a single message.

3.3 Calculating Estimated Arrival Times

The accuracy of a reservation is almost entirely dependent on the accuracy of each estimated time of arrival for each location contained within the reservation. Thus, it is vital that agents know how fast they are moving so they can calculate how long it takes them to move from one location to another. Unfortunately, in the 3D simulator an agent's speed is not consistent. Agents are sometimes forced to slow down in order to maintain balance, they may lose speed while turning, or something else may happen which disrupts their current velocity. To overcome this challenge, agents keep a running average of how long it takes them to move from one location to another. They then use this average time when planning paths to look for conflicts and calculate estimated arrival times. They also use it to update their estimated arrival

times prior to reserving a window. This process results in arrival times which are estimated with a good degree accuracy.

3.4 Determining Reservation Conflicts

While planning its initial path, replanning a path, and attempting to reserve a window, an agent is constantly checking against its local copy of the reservation table to make sure that its attempt to reserve a location at a certain time would not resolve in a conflict. Given the near-infinite possible times that a location may be reserved, it would be pointless to look for exact time conflicts. Not only that, but reservation times are not guaranteed to be 100% accurate. Despite an agent's best efforts to calculate exactly how long they take to move between two locations, the number they use is still only a running average. For this reason, it is necessary to create a certain tolerance for which two times, although not exactly equal, will be considered equal for the sake of detecting conflicts.

4 Experimental Results

A simple experiment consisting of 3 samples was completed in order to determine the effectiveness of the above cooperative pathfinding algorithm. For each sample 10 trials took place. In each trial, 11 agents belonging to the same team were randomly placed on their team's half of the soccer field. The agents then moved around that half of the soccer field randomly for 300 seconds. Throughout the trial the number of collisions was recorded. A collision was defined as anytime when two agents who were upright and moving ran

into each other and both fell down.

4.1 Naive Sample

In the first sample, the agents did not attempt any form of cooperative pathfinding. Instead, they simply moved in a straight line to their goal at full speed. The data for this sample can be found in the below table.

Trial	Number of Collisions
1	18
2	23
3	16
4	19
5	17
6	17
7	21
8	16
9	26
10	22

The average number of collisions per trial was 19.5 and the median was 18.5. According to a one sample T interval, we can say with 95% confidence that the true mean is somewhere between 17.086 and 21.914 collision.

4.2 Spatial-Temporal Conflicts Sample

In the second sample, the agents used the cooperative pathfinding algorithm defined in this paper. The tolerance for detecting collisions was set at 3 times

the agent's estimated interval between locations. This value was chosen as it provided a reasonable buffer between reservations. The data for this sample can be found in the below table.

Trial	Number of Collisions
1	10
2	15
3	13
4	13
5	15
6	13
7	11
8	10
9	11
10	11

The average number of collisions per trial was 12.2 and the median was 12. According to a one sample T interval, we can say with 95% confidence that the true mean is somewhere between 10.86 and 13.54 collisions.

4.3 Spatial Conflicts Sample

For the third and final sample, the agents calculated collisions by only looking at whether or not an agent had reserved the same space instead of the same space at a similar time. This sample was taken because after observing the performance of agents in the previous sample it could be seen that the effects of falling down were not being thoroughly accounted for by the algo-

rithm. This caused agents' estimated arrival times to sometimes be highly inaccurate. So by ignoring the time portion of the reservation, it wouldn't matter if an agent failed to vacate a location in time because it fell down. The data for this sample can be found in the below table.

Trial	Number of Collisions
1	10
2	9
3	8
4	9
5	6
6	7
7	9
8	10
9	5
10	9

The average number of collisions per trial was 8.2 and the median was 9. According to a one sample T interval, we can say with 95% confidence that the true mean is somewhere between 6.9935 and 9.4065 collisions.

4.4 Conclusions

A number of conclusions can be drawn from the experimental results. First, cooperative pathfinding does appear to reduce the number of collisions between teammates. The confidence interval for the naive method was clearly higher than the intervals for both cooperative methods. Additionally, the

chances of the cooperative methods having the same true mean as the naive method, according to two sample T tests, are .003% and .00002% respectively.

The other important conclusion to note is that the spatial conflicts method performed better than the spatial-temporal conflicts method. The confidence interval for the spatial conflicts method is clearly below the confidence interval for the spatial-temporal conflicts method. A two sample T test reports that the chance of the two methods having the same true mean is .009%. Thus, in the algorithm's current state it is clearly better for an agent to ignore the estimated arrival times in its reservation table, and simply see if the location they wish to reserve is being reserved at any time.

5 Final Thoughts and Future Work

5.1 Post Implementation Observations

Although the presented algorithm was effective in reducing the number of collisions between teammates, the results of the experimentation clearly show that further work could be performed to increase the algorithm's performance. The dominant method of detecting collisions based on the spatial properties of reservations is not ideal, and leads to less optimal paths as agents are forced to go around all reservations.

The most obvious opportunity for improvement can be found in the algorithm's poor handling of agents falling down, either on their own or as a result of a collision. The time consumed when an agent falls over and

tries, sometimes repeatedly, to get up, causes the agent's running travel time average to become highly inaccurate. It also causes the agent to miss the remainder of its reservation and stay in a location for far too long of a time. This essentially invalidates the temporal properties of a reservation, thus explaining the fewer collisions found when detecting conflicts purely based on location. Fixing this could involve a fallen agent communicating that they have fallen over and in what location, with the other agents then being forced to replan their paths around the fallen agent. The calculation of the agent's estimated travel time average would also have to either be reset or suspended for as long as the agent was on the ground.

Finally, the presented algorithm fails to correctly detect the collision of two diagonal lines which do not intersect on a specific grid coordinate, but do intersect as a whole. Since the lines technically do not share any of the same coordinates, it is impossible for the algorithm to see that they collide. A solution to this problem has currently not been thought of, but the exploration into one would help in reducing the number of remaining collisions.

5.2 Future Work

The fact that agents are no longer taking the true optimal path, a straight line, to their goal is the largest disadvantage of the presented algorithm over the naive non cooperative solution. For this reason, the ideal course for future work would involve creating a similar cooperative pathfinding algorithm which allowed agents to travel on the straight line path to their goal. To

implement such an algorithm it would be necessary to calculate the supercover, or all of the grid locations that a line passes through, of the agent's line. This could be accomplished through the use of a modified Bresenham algorithm. The limitation on message length in the 3D simulator would make determining what grid locations to include in a reservation window difficult though. That being said, a method to accurately create reservation windows could probably be devised. Any other future work would most likely be motivated by the need to make the agents' arrival estimations more accurate. This could be done in a number of ways, which include those specified in section 5.1.

6 Citations

1. D. Silver. Cooperative Pathfinding. In AIIDE, pages 117122, June 2005.

2. K.-H. C. Wang and A. Botea. Fast and Memory-Efficient Multi-Agent Pathfinding. In ICAPS, pages 380387, 2008.

3. Kurt Dresner and Peter Stone. A Multiagent Approach to Autonomous Intersection Management. Journal of Artificial Intelligence Research, 31:591656, March 2008.