Report Assignment 2 DD2380

GROUP 2:26
Vilmer Jonsson
2001-06-26
vilmerj@kth.se

March 26, 2024

Abstract

When several vehicles are maneuvering in the same area, it is important to be able to predict the future behavior of the other vehicles. This report presents a solution to two problems. Firstly, the traffic problem, where each vehicle is trying to reach its goal as fast as possible. Secondly, the formation problem, where several vehicles are trying to reach targets in a coordinated manner. The solution to the traffic problem is based on velocity obstacles, individual path planning and recalculating the path when needed. The solution to the formation problem is based on adjusting the velocity of the vehicles to reach the goal at the same time. The results show that the implementation works well. Two of the 16 tests were not completed, both of these had problems with cars merging. The best group had a similar approach to the formation problem but a slightly different approach to the traffic problem.

1 Introduction

When maneuvering a vehicle, there is often several other vehicles in the vicinity. These vehicles are maneuvered by other drivers, who all have their own goals and intentions. To be able to maneuver a vehicle safely, it is important to be able to predict the future behavior of these other vehicles. When driving a car, this is done by the driver, who uses his or hers knowledge of the traffic rules and experience to predict the future behavior of the other vehicles. Logically, if you were to construct a self-driving car, you would need a similar system to predict the future behavior of the other vehicles and to be able to make decisions based on this information.

This is one of the problems that this report will address. The other problem is similar, but instead of the vehicles trying to get to separate goals independently, they are trying to reach several close targets in a coordinated manner. The first problem will be called the *Traffic* problem and the second problem will be called the *Formation* problem.

The most important part of the solution to the Traffic problem is to be able to predict the future behavior of the other vehicles. This is done by using the concept of velocity obstacles. Velocity obstacles are a way of predicting whether a certain velocity will result in a collision with another vehicle. This was used since it is a simple and efficient way of ensuring that no collisions will occur.

The most important part of the solution to the Formation problem is to be able to get the vehicles to adjust their velocity to reach targets at the same time. The solution is based on a leader-follower approach, where one vehicle is the leader and the other vehicles try to keep a certain distance to the leader.

1.1 Contribution

In this project, a solution to the Traffic problem using velocity obstacles, individual path planning and recalculating the path when needed. For the Formation problem, the contribution is a method to get the vehicles to follow their individual paths and adjust their velocity to reach the goal in a coordinated manner.

1.2 Outline

In section 2, related work to both the Traffic and Formation problem will be introduced in order to give the reader more information about the past research in the area. In section 3, the implementation will be presented in more detail. In section 4, the experimental results will be presented and analyzed based on the best group's performance.

2 Related work

Avoiding other vehicles is a problem as old as traffic itself. However, in robotics the development were first done on trying to collide with objects. One of the first use-cases was a missile trying to hit a moving target. This was done by using the relative velocity of the missile and the target to calculate the optimal path for the missile. [1]

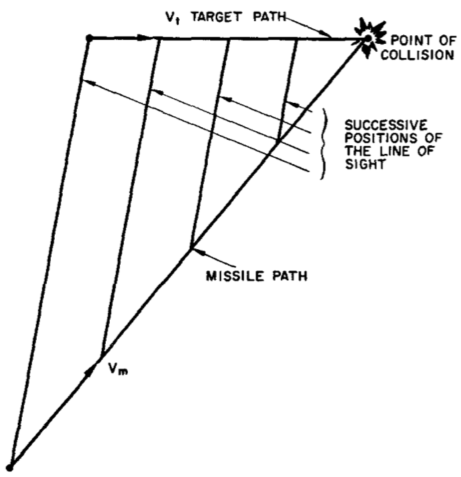

Figure 1: Proportional navigation, a method used to calculate the optimal path for a missile to hit a moving target. [1]

Velocity obstacles were first introduced by Fiorini and Shiller in 1998. They are a way of predicting whether a certain velocity will result in a collision with another vehicle. This is done by defining a cone which has one tip in the center of the vehicle and intersects with the other vehicle on the boundary of the cone. The whole cone is then moved along the velocity vector of the other vehicle. If the cone intersects with the vehicles' velocity vector, the velocity is not safe. [3]

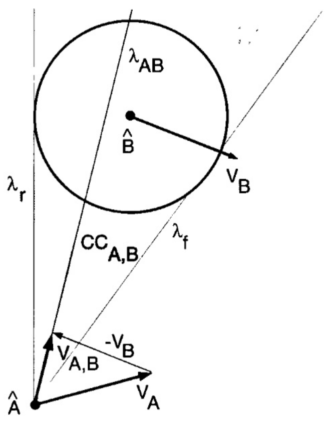

Figure 2: Velocity obstacles. The circle is the moving obstacle and the vehicles current position is A. [3]

A development to velocity obstacles called *Reciprocal Velocity Obstacles* were introduced in 2008 by van den Berg et al. This method is more efficient than velocity obstacles since it assumes that the other vehicle is also trying to avoid a collision. This means that both vehicles will try to avoid each other and do not have to steer away as much as with traditional velocity obstacles. It also avoids the problem of oscillations that can occur with traditional velocity obstacles, see Figure 3. [5]

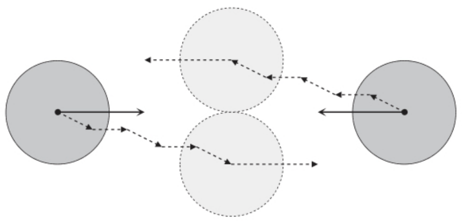

Figure 3: Oscillations when two agents avoid each other using traditional velocity obstacles. [5]

The same authors to the previous paper also introduced a method called Hybrid Reciprocal Velocity Obstacles in 2011. The difference to reciprocal velocity obstacles is that the vehicles are encouraged to pass each other on the same side. If the vehicle needs to go on the other side, it will give full priority to the other vehicle. This is done in order to avoid the agents "dancing" around each other since they can not decide which side to pass on. [4]

For formation keeping, one method is the leader-follower approach. This means that one vehicle is the leader and the other vehicles are followers. The followers then adjust their velocity to keep a certain distance to the leader. The leader gets input from an external source while the followers get input from the leader. This is a simple and efficient way of keeping formation. There is no feedback to the leader from the followers. [2]

3 Proposed method

Proposed Method section explaining what you did in more detail. Use references and explain how you adapt, build upon, or use elements of the algorithms proposed in the reference.

Traffic problem

The solution to the traffic problems rely heavily on the concept of velocity obstacles. Each agent is first assigned a path based on an A* algorithm on a visibility graph. The paths consist of several numbered checkpoints that the agent should pass in order to not collide with static obstacles, for example walls.

The agents begin to move along their paths and at each time step, they calculate the velocity obstacles for neighboring agents. If the agent is in a collision course with another agent, it will sample close velocities and choose the one that is closest to the desired velocity and does not result in a collision. If none of the sampled velocities are safe, the agent will take the "least unsafe" velocity.

Skipping checkpoints

The path finding algorithm will return the shortest path between the start and the goal. However, this path might not be the shortest if the agent is straying from the path. To avoid this, the agent will skip checkpoints if it is safe to do so. This is done by checking if there is a static obstacle between the agent and the next checkpoint. If there is not, the agent will skip the checkpoint and move towards the next one. This is checked at each time step.

Recalculating paths

The agent will always try to move towards the next checkpoint. However, if there is a static obstacle in the way, the agent will not be able to reach the checkpoint. In this case, the agent will recalculate the path to the goal and thus avoid the obstacle. This is done by using the A* algorithm on the visibility graph again.

Recovering

Recovering from crashes are an important part of the system. If an agent is in a crash, it will try to recover by moving backwards and turning at full steering angle. This is done to avoid getting stuck in a corner or similar. This method is not optimal and would need to be improved in order to work properly.

Formation problem

The formation problem is solved by using something similar to leader-follower but much simpler since the performance is only measured by two factors:

- The time of the last agent to reach the goal.
- The difference in time between the first and last agent to reach each checkpoint.

This means that the formation does not have to be perfect at all times, only at the checkpoints. The agents will adjust their velocity to reach the checkpoints at the same time. This is done by calculating the distance to the next checkpoint for every agent and adjusting the velocity based on the ratio defined in equation 1.

$$\frac{\text{distance to checkpoint}}{\text{greatest distance to checkpoint}} \tag{1}$$

If the greatest distance to the checkpoint is close to zero, the agent will adjust its velocity to reach the checkpoint as fast as possible. Note that the ratio is not used directly to scale the velocity linearly, but rather to determine whether the agent should speed up or slow down. If the ratio less than a certain threshold, the agent will stop and wait for the other agents to catch up.

Ideally, the agents should avoid each other similar to the traffic problem. However, this was not implemented due to time constraints. This was not needed for the tests that were performed, but would be needed in a more complex environment. The reason for it not being needed in the tests is that the agents do not hold formation at all times, only at the checkpoints. This means that the agents often will be far apart and not in danger of colliding with each other.

3.1 Implementation

Neighbors

One part of the implementation is to determine which agents are neighbors to the agent. This is done by checking the distance between the agents. If the distance is less than a certain threshold, the agent is considered a neighbor. The threshold is dynamic and increases with the velocity of the agent. An agent is also considered a neighbor if the two agent's displacement vectors, $\vec{v}_A + \vec{pos}_A$, are at a certain distance from each other. This is done since these agents will collide head on if they continue on their current paths.

Sample velocities

The car will sample velocities in an arc in front of it. The arc is defined by the maximum turning angle of the car, the distance between the arc and the car are defined by the current velocity of the car.

The drone will sample velocities based on acceleration. This is done since the drone takes two accelerations, lateral and longitudinal, as input commands. The drone will sample accelerations in a circle around the current velocity of the drone. The radius of the circle is defined by the maximum acceleration of the drone. The resulting velocities are calculated by equation 2. If the sampled velocity's magnitude is greater than the maximum velocity of the drone, it will be scaled down to the maximum velocity. Note that this is a simplification of acceleration's effect on velocity.

$$v_{new} = v_{current} + 0.5a \tag{2}$$

The sampled velocities are sorted based on the distance to the desired velocity. The velocity that is closest to the desired velocity and does not result in a collision is chosen. Naturally, if two agents head right towards each other, they will both choose to pass each other on the same side, exactly as hybrid velocity obstacles.

Least unsafe velocity

Determining the least unsafe velocity when none of the sampled velocities are safe is done by a simple heuristic. Every sampled velocity is tested on all velocity obstacles until it finds an obstacle it is colliding with.

Performance issues

Since velocity obstacles, checkpoint skipping and path recalculating are all computed after start, the performance of the system is crucial to prevent lag. Some things done to improve performance is to only calculate velocity obstacles for neighbors the neighboring agents, sample velocities only if the agent is in a collision course with another agent and not check all sampled velocities if one is found to be safe.

4 Experimental results

Table 1: Results on traffic problem for car and drone compared to the best time.

Vehicle	Car					Drone				
Map	open	semi open	intersection	highway	onramp	open	semi open	intersection	highway	onramp
My time	22.9	42.7	88.1	-[1]	-[1]	17.62	21.4	31	41.6	41.82
Best time	18.98	22.6	50.1	65.62	43.34	17.62	18.74	29.22	41.02	41.82
Best group	3	3	11	2	25	26	7	27	25	26

Table 2: Results on formation problem for car and drone compared to the best time.

Vehicle	Car		Drone	
Map	6	6B	6	6B
My time	93.25	94.38	109.18	66.58
Best time	90.28	67.53	84.96	64.2
Best group	2	6	11	7

4.1 Analysis of Outcome

The outcome of the final tests can be seen in Table 1 and Table 2. The results show that the drone performed better than the car in both the traffic and formation problem. The main reason for this is probably that while the car is faster than the drone, it is also larger and is affected more by crashes. The drone is also more agile and can steer in all directions, while the car's steering model is more limited.

The results also show that my implementation was the best at two of the maps and often is very close to the best time, especially in the drone traffic problem. This is a good result and shows that the implementation is working well, but there is still room for improvement. Two of the maps were not completed, the highway and on-ramp maps for the car.

The best performing group was group 2. For the traffic problem, they relied a bit more on a dynamic path planning where they iterated over the paths and weighted the different nodes based on how many other cars pass that same node. That seems like a good idea and would probably decrease the number of crashes. For the formation problem, they had a similar approach to mine, but they only switched leader between the agents at the checkpoints. I think my approach is better since it is more flexible and does not rely on the agents being in a certain order. However, other parts of their implementation might have been better than mine, for example, they used a grid for their A* algorithm which ensures that the paths

[1]No completion, counts as 2000 seconds.

never collide with each other.

The fact that two maps were not completed should be seen as a failure. However, since I worked alone and thus had half the time to work on the project, I will not include an analysis of failure. However, I think that one of my problems was that the recovery mechanism was not good enough. The car often got stuck in corners and could not recover. Another problem is that the cars should slow down and form a queue when merging. Discover merging situations could be done by checking if the velocity vectors of the vehicles interfere with each other. The problem could also be solved by sampling velocities that are closer to the vehicle as well, and braking if they are free. For the car, the collision avoidance only sample velocities based on a different turn radius, not on braking or acceleration.

It should also be said that collisions did occur in my solution. This is not optimal and should be avoided. This makes the car drive into each other and slows them down or makes them stuck. This could be solved by sampling velocities that are closer to the vehicle as well, and braking if they are free.

5 Summary and Conclusions

The goal of this project was to implement solutions to the traffic and formation problems. For the traffic problem, the solution was to use velocity obstacles to avoid collisions with other agents. For the formation problem, the solution was to adjust the velocity of the agents to reach the goal at the same time. The results show that the drone performed better than the car in both problems. The implementation was close to the best time in most cases, but two maps were not completed. The best group was group 2, who had a similar approach to the formation problem but a slightly different approach to the traffic problem. The implementation failed on two of the maps for the car problem. Both of these maps were based on merging. A more sophisticated strategy for these scenarios would be needed to complete these maps.

The formation problem was not as complex as the traffic problem. The agents only had to reach the checkpoints at the same time, not hold formation at all times. However, in a real-life scenario where the agents would have to hold formation at all times, the formation problem would be much harder to solve.

References

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

[2] Fei Chen and Dimos V. Dimarogonas. Leader–follower formation control with prescribed performance guarantees. *IEEE Transactions on Control of Network Systems*, 8(1):450–461, 2021.

[3] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[4] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706, 2011.

[5] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*, pages 1928–1935. IEEE, 2008.