Multi-robot Navigation in the Context of a Chasing Game

Ivan Mauro Gomes Ribeiro, Jonáš Fiala and Ravikumar Shah

Abstract— **We attempt to implement a two-team game in which each team has a different goal. We describe and implement different methods for navigation and path planning and compare the relative merits of each one, and in addition test each method using different starting locations and relative speeds.**

I. INTRODUCTION

Games provide a good environment in which to test various navigation and path planning algorithms. In this project we implement a chasing game, where the robots are split into two teams called the Police and the Baddies. The Police attempt to chase and catch the Baddies. Baddies are considered caught once any Police robot gets within a specified radius of them. The goal of the Baddies is to evade capture for as long as possible and to reach a specified exit location. We test various navigation and planning algorithms for controlling the Baddies in simulation, and measure how they compare to each other. We also test the behaviour of each navigation and planning algorithm when the relative speeds of the teams change. We apply these to simulated models of the Turtlebot 3 robot [1].

In this project we consider four different navigation strategies:

- Braitenberg vehicles
- Potential Fields
- RRT* using arc-based motion primitives
- RRT* using straight-line-based motion primitives

A. Feedback Linearization

The robots we have used for this project are differential wheeled non-holonomic robots. However, potential field-based algorithms for navigation require the target robots to be holonomic. We can use feedback linearization in order to remedy this mismatch by changing our conceptual model of how our robots are controlled. In this project, we have done this by controlling a point p at a distance ϵ ahead of our robot. This ϵ was chosen experimentally before the project to balance accuracy of movement and minimise oscillatory behaviour. We are able to use the angular and forward velocity of the robot to control the position of the point p holonomically. This allows us to apply the potential field-based methods to control the robot. We found the mismatch between the position of the point p and the actual location of our robot to be negligible in our testing.

B. Braitenberg Vehicles

Braitenberg vehicles were first introduced as a thought experiment [2]. They have several sensors which are connected directly using simple mathematical functions to the motors that guide the robot. The model generates reactive behaviour based on events, but does not depend on internal state. They do not have a goal to reach and hence can't be used to direct a robot to some exit point, but the robot could come to this goal on its own through random exploration.

C. Potential Fields

The value of a potential field at some point in space gives you the "potential" of that point. The potential can be used to model different things. Where the global minimum of the potential gives the final desired state. Once we have set up a potential field, we can compute its gradient in order to get a vector field. This vector field gives us the direction which we should follow in order to minimise the potential of the robot and reach our goal, in essence performing a gradient descent.

Potential fields are useful when low-latency lookups are required, but have the disadvantage of encountering local minima or saddles. At these locations, it is not possible to tell what direction you should move in, and in general moving away from these local minima and towards the global minimum requires some form of non-deterministic behaviour. We simulate this in our project by making our potential fields time-variant. This reduces the chance of the robot getting stuck in one position forever.

*D. RRT and RRT**

A rapidly-exploring random tree (RRT) is a data structure introduced by LaValle designed for use in path planning algorithms. Nodes stored in this RRT represent positions that are free in the state space, and edges signify that there is a way to navigate between the two positions.

RRT* [3] is an algorithm that attempts to generate RRTs that contain the optimal path between the start and the goal. It uses a heuristic function to evaluate the cost of each node. In standard RRT algorithms, execution finishes once any path has been found between the start and the goal, or the number of iterations has been reached. In RRT*, you continue until the number of iterations has been reached, rewiring as you go, and use the cost function to obtain a path of minimal cost.

II. DESCRIPTION OF THE PROBLEM

Our objective is to create a system where we can test the various navigation and planning algorithms under different circumstances and measure how the performance of the system is impacted. For example, one measurement that could be extracted from the data is whether there is a critical relative speed for the Baddies, beyond which they have a much higher chance of escape. We also want to find out what effect the knowledge of the Police with regards to Baddie location has on their ability to catch the Baddies.

We decided that we should give all robots full knowledge of the map. Initially, both teams also had full knowledge of all robot positions. This was subsequently extended such that the Police only had exact locations for Baddies that they could see. This was defined as being Baddies that the Police had a line-of-sight of. Objects in the map could affect the certainty of this. This can be handled by handling the line of sight as a likelihood. For example, for Baddies that were near a wall, we introduced an uncertainty. In order to better utilise the navigation and planning algorithms we had selected, we decided to allow Baddies to always have knowledge of the position of all robots in the arena.

III. THE APPROACH

A. The Map

In order to perform experiments on the navigation algorithms a map was required. We decided to use a map that resembled a real-world city, with straight roads and blocks. Our first map had many dead-ends, which caused issues when using more naive implementations, so we modified the map to remove these dead ends. We added an opening to one edge of the arena that we called the exit. We considered any Baddies that reached this location as having escaped from the arena.

B. Police Navigation

The Police navigated exclusively using the potential field method, with a sum of three different component fields v_{goal}, v_{avoid} and $v_{scatter}$. Initially a target Baddie is picked for each of the Police, with multiple Police assigned to chasing the same target. An attractive potential field is used to navigate the Police towards this target.

Obstacle avoidance posed more of a problem, as we wished to generalise the navigation to any arbitrary occupancy grid map, rather than requiring knowledge of all the individual obstacles. To create this obstacle gradient field we treat the occupancy map as an image I and convolve it with a Gaussian (blur), the variance of which determines how far away we will stay from obstacles. Then taking the first derivative we achieve a suitable gradient field $\nabla(G * I)$. It can be noted that this is identical to first taking the derivative, so as to get a sum of Dirac delta functions at all the edges of obstacles, and then performing convolution $G * \nabla I$. A potential downside of this method as compared to for example the Brushfire Algorithm is that we will not be able to avoid very thin obstacles, however this is not a problem given our city like map.

Lastly to achieve better cooperation of the Police in cornering a Baddie, we added a small repulsive force for all the Police away from each other. This means that two Police at an intersection will tend to choose different directions. We also aided this behaviour by making the Police follow a goal slightly forward of the Baddie, proportional to the distance of the Police from the Baddie. Even though it is an extremely simple and naive prediction, it worked well enough in practice.

C. Line of sight

The initial navigation technique for the Police used knowledge of where the Baddies were in the arena. This was subsequently extended such that only an approximate pose of the Baddie was given to the Police, the quality of which depended on line of sight. To approximate 'visibility' v at any point a we use the blurred map from the potential field navigation, visibility is 0 over obstacles, low near edges and then approaches 1. From this we calculate visibility v' from an arbitrary position a to b using a ray trace: For $P = \{a, a+s, a+2s, ..., b-s, b\}$, where s is the step, $v'(a,b) = \prod_{p \in P} v(p)^{|s|}$. By taking visibility to the power of step size we normalize for step size. Thus any ray trace crossing over an obstacle will be set to zero, peering around corners or seeing a Baddie far away will result in low visibility and close up we will achieve a visibility of near 1.

The total visibility $V(b)$ is calculated per Baddie as the maximum visibility from any of the Police, since we assume that the Police can communicate. We introduce noise by re-sampling the true pose from a Gaussian distribution with a standard deviation inversely proportional to the visibility.

D. Police Navigation with Baddie Localization

We relied on a particle filter for localization of the Baddies to determine a path. The particles are all initially spawned at the known initial location of the Baddie. Movement was simulated by having a hypothesis of the forward and angular velocity, along with the pose, of the Baddie within the velocity and pose constraints of the robot, these hypotheses are varied randomly. Given the forward and angular velocities we can calculate the delta pose per time step. There is also a small chance that a particle will randomly teleport to a nearby pose, this is to help deal with particle collapse.

Subsequently, the particle is weighted based on the given observed pose and standard deviation, we assume that we know the noise model exactly (Gaussian noise with std dev inversely proportional to visibility). The weight of a particle represents the probability that it is the correct hypothesis of the true pose of the Baddie. For particle pose p, observed pose μ and standard deviation σ, the weight is given as

$$w(p, \mu, \sigma) = \begin{cases} 0, & \text{if } p \text{ invalid} \\ \prod \text{norm.pdf}(p, \mu, \sigma), & \text{if } \sigma \neq \text{inf} \\ 1 - V(p), & \text{otherwise} \end{cases}$$

For the third case of $\sigma = \text{inf}$ we give particles with high visibility a low weight since the true hypothesis must have 0 visibility, we don't calculate visibility per particle in the other cases as it is quite expensive. There is a final fourth case where given a $\sigma < 0.01$ we assume that we know the exact hypothesis and hard set the particle pose to the observed pose, this helps if the particles have collapsed in an incorrect state.

E. Baddie Navigation with Braitenberg

During the initial implementation of the Baddies, they were programmed to navigate using the idea of Braitenberg vehicles. This was a naive method which provided a framework to build on. The main pitfall was in the closed arena, where a Baddie could indefinitely evade capture where naive path planning was used, as a result of its speed advantage.

F. Baddie Navigation with Potential Fields

The subsequent iteration focused on the implementation of a potential field. The potential field method created a field in the robot work space, from which we could compute the gradient and derive the robot's control inputs using feedback linearization. This proved effective in allowing a Baddie to evade capture however it would always be cornered in a closed arena if it was being chased. In an open arena, the Baddie being targeted was usually still captured, however on occasion both Baddies could escape. This was again a naive method, the navigation was considerably inefficient as a result of local minima in the field and the dynamic positioning of the Police.

G. Baddie Navigation with RRT* based methods

The next implementation initially began using exclusively RRT*, which built a navigation tree of next states using randomly generated control starting at a vertex $Q1$. We sample a random state $Qrand$ in the control space, find the nearest vertex already in the RRT $Qnear$, and use some local planning to find a vertex $Qnew$ in the free space that is close to $Qrand$ and is on a collision-free path with $Qnear$. Initially, a movement was considered valid if there was a circular arc between the vertices. If these conditions are satisfied, we can add the edge and vertex to the tree. These steps are repeated until there is a path from $Q1$ to the goal.

This implementation lacked flexibility and generated paths deemed too rigid. We found that even when running 5000 iterations the algorithm struggled to find a path from the current pose to the exit, which made this method infeasible. We therefore decided not to try to evaluate it.

Subsequent optimisation used a more hybridized approach, with RRT* used for high level navigation, and potential fields for the point-to-point path navigation. A path was considered valid if there was an unobstructed line of sight between the two vertices. This was a much easier constraint for the algorithm to satisfy, particularly in our grid-like map, which meant that we got significantly better performance than the earlier attempt which had more restrictive motion primitives. This method was deemed usable, and we evaluated it against the other more naive methods.

IV. EVALUATION

Evaluation of the algorithms was performed using the observed chance of escape for each navigation method. We also considered what we thought of their pathing, but this has been excluded from the results section as it is not easily quantifiable and is more subjective. We discuss this briefly in the conclusion.

A. How the experiments were run

Several experiments were conducted in order to collect results. In order to ensure that observed changes corresponded only to changes in the modified variables, we specified a starting state from which each experiment was to be run. The static elements of this state included:

- The number of Police and Baddies
- The initial positions of the Police and Baddies
- The speed of the Police
- The layout of the map
- The algorithm for assigning Baddies for Police to chase

Figure 1 displays our map and initial robot positions.

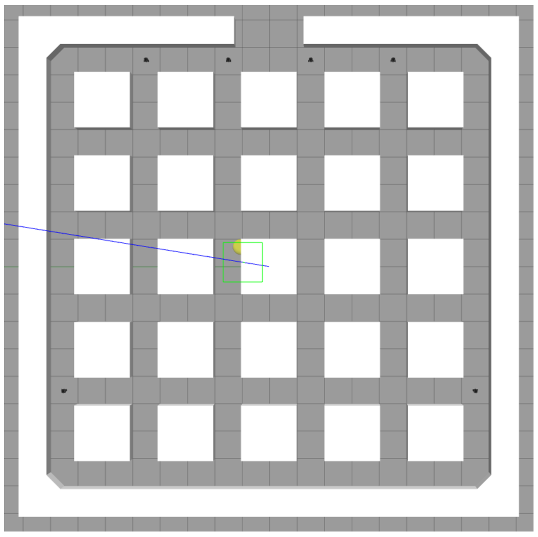

Fig. 1. Our initial setup. The four robots at the top of the image are the Police. The two robots near the bottom are the Baddies

We defined the following as the variables in our experiments:

- the speed of the Baddies relative to the Police
- the navigation method of the Baddies
- the Police's knowledge of Baddie positions

In each experiment only one parameter was varied. All other variables were kept static in order to isolate the effect of each change. Each experiment was repeated several times in order to account for the variability of the non-deterministic algorithms being tested.

B. Results

We see in Figure 2 that at each speed, the RRT*-based implementation has a significantly higher rate of escape compared to the other methods. We can also infer from this figure that the critical relative speed for the potential field method appears to be around 2x as fast as the Police. At this speed we observe a large increase in the chance of the Baddies escaping. We note also that for the potential field method the chance of escape seems to decrease when a higher speed is selected. We believe this is due to the fact that our simulation was updating at a frequency of 20 Hertz. The higher velocity meant that the robots moved further

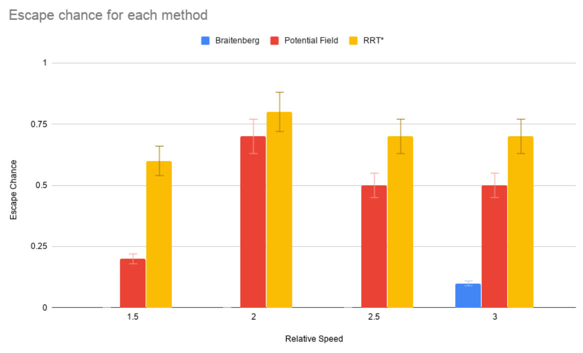

Fig. 2. Escape chance of our selected methods

during each tick, which meant that they ended up sometimes bumping into objects and being slowed down.

We can see in Figure 3 that inexact knowledge leads to the Police having a much lower chance of catching the Baddies. This could be improved upon by using a different method for predicting Baddie intentions.

Figure 4 showcases the behaviour of our Baddies when confronted with a Police in their path. We see that the Baddie robot alters its path to follow one which avoids the Police and reaches the goal.

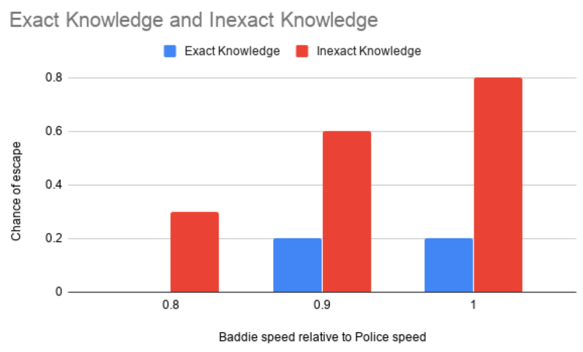

Fig. 3. The chance of escape when the Police had exact knowledge of Baddie positions and when they used our particle-filter implementation

V. CONCLUSION

We conclude from our observations that the RRT* planning and navigation algorithm that we introduced performs better in all cases than the naive potential field and Braitenberg methods. We observe that for Braitenberg vehicles there does not seem to be a critical relative speed at which they begin escaping with high probability. For potential fields we observed this critical speed to be around 2x the speed of the Police. We have not been able to pinpoint a speed at which RRT* drastically increases its chances of escape, but it appears to lie between 1x and 1.5x. We also conclude that Police knowledge of Baddie localisation has a very significant impact on their ability to catch Baddies at lower speeds. We expect that this becomes less important at higher speeds as the Baddies already have high chances of escape

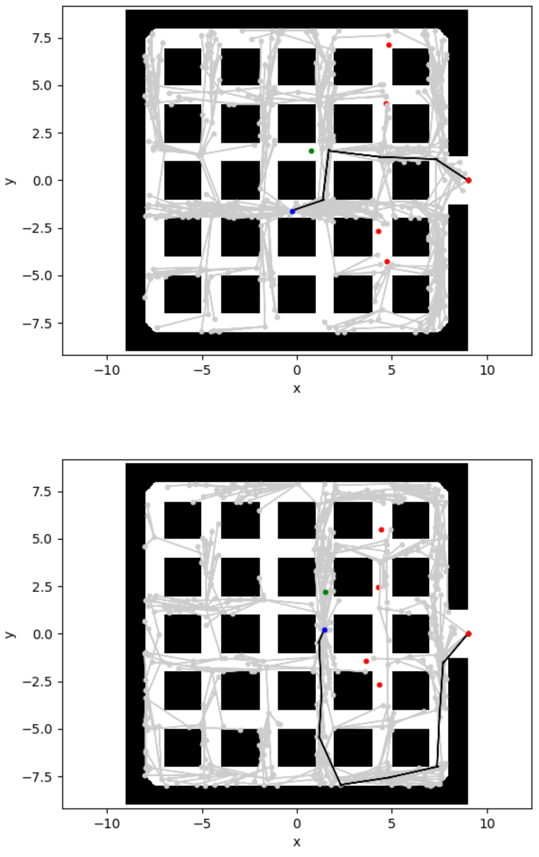

Fig. 4. The Baddie changing paths. The navigating robot is in blue, Police are red inside the arena and the red dot at the exit is the goal. the green dot is the other Baddie

It is also our subjective opinion that the paths generated by the RRT* algorithm seemed "smarter" - they seemed to actively avoid planning paths that went near the Police, which we believe led to better chances of escaping. Our code is available at [4]

VI. CONTRIBUTIONS

Ivan	Base framework, RRT*-based navigation, data collection
Jonas	Map generation, Potential field based navigation, Line of sight system and helped out with the particle filter
Ravi	Localization using a Particle filter for Police Navigation, some initial robot following that was later replaced

REFERENCES

[1] http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/
[2] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, in The MIT Press, Cambridge, Massachusetts, 1984.
[3] S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning, in Int. J. Robot. Res., vol. 30, no. 7, pp. 846–894, Jun. 2011.
[4] https://github.com/JonasAlaif/mrs_project