Hide-and-Seek: Simulating Robotic Pursuit and Evasion

Warren Kozak, Christian Park, and Omar Sobhy
Department of Computer Science, Carleton College, Northfield, Minnesota 55057, USA

ABSTRACT

Many problems in robotics reduce to pursuit and evasion. In this paper we model and compare common pursuit strategies in a bounded arena with static obstacles. Using ROS2 and Gazebo, we build a custom differential-drive robot and pit two pursuit controllers against an artificial potential field (APF) evader. The first is a pure pursuit controller, and the second is a cooperative pursuit algorithm that partitions the arena into Voronoi cells and falls back on pure pursuit. We vary the number of pursuers from two to four and run ten trials at each count. We find that pursuer count dominates the outcome. Two pursuers never caught the evader within the fifty-second cutoff, so the evader won every trial, while three and four pursuers won almost every trial at a near-identical average capture time of about fifteen seconds. Cooperative Voronoi pursuit therefore outperforms its pure pursuit fallback only once there are enough pursuers to keep the arena partitioned.

I. INTRODUCTION

Pursuit and evasion as an open problem is well established in traditional mathematical literature [1], however it is becoming increasingly practical. As robots become more and more useful and capable, modelling robotic pursuit and evasion is an active field of research that is ongoing. Pursuit algorithms are used to solve problems in the real world that involve efficiently intercepting "evaders" such as autonomous search and rescue, missile defense, obstacle avoidance in self-driving cars, as well as many others. In this paper, we seek to answer the question of what the most efficient method for pursuit of an evader is. In order to simplify this question and focus on the underlying pursuit algorithms, we make some critical assumptions in our work. The first of these is that obstacles in the environment are static. In a real pursuit scenario, obstacles such as people or vehicles are constantly in motion, however we simplify this in order to make the task of algorithm analysis easier. Additionally, using our `/manager` topic, we publish ground-truth poses of all robots for both pursuers and evaders to see. Rather than relying upon simulated odometry or LiDAR scans, our use of ground-truth pose takes out the uncertainty that exists in many physical pursuit scenarios, again allowing for easier comparison of algorithms.

The remainder of this paper consists of a short survey of existing literature, and an overview of our simulation environment and methods. We then move on to discuss our results, before wrapping up with a conclusion.

II. BACKGROUND

ROS, specifically version 2, is the industry standard software that serves as the basis for robots throughout industry and research [2]. At its heart, ROS is a network communication protocol, with different nodes communicating over the network through messages called topics. These topics allow the transmission of different kinds of specialized data, including sensor readings as well as motor control messages. Due to the prevalence of ROS throughout robotics, as well as the fact that we learned about this software in class, we used ROS as the basis for our project. Additionally, ROS has a vast ecosystem of open source plugins that can be used along with it. We used Ignition Gazebo, one of these plugins, to simulate the robotic pursuit and evasion outlined in this paper. Gazebo is a 3D physics engine and robotic environment that is capable of simulating realistic sensor data, such as input from LiDAR [3]. This is crucial for our project, as using real hardware for a multi-agent pursuit and evasion simulation is not feasible currently given our constraints.

Pure pursuit is one of the most well-established algorithms for modeling pursuit. It is a geometry-based algorithm in which the pursuer constantly calculates a look-ahead distance in order to predict where the target point to head towards should be. This in turn allows the pursuer to predict the location of the evader. The geometric derivation and parameter tuning is outlined in [4].

To coordinate multiple pursuers, we layered a Voronoi-based strategy on top of this, following [5]. At each timestep we partition the arena into Voronoi cells, where every cell holds the points closest to a given robot. Each pursuer whose cell is adjacent to the evader's cell then drives toward the midpoint of the edge its cell shares with the evader's cell. Pursuers whose cells are not adjacent to the evader utilize pure pursuit to attempt to get closer to the evader. Because each pursuer owns a different shared edge, they spread out and approach from different angles on their own, with no communication between them. We added a ring of mirror points just outside the circular arena so that every cell stays bounded. At a high level, the goal of this algorithm is to minimize the safe area of the evader, that is to push inward the boundaries of the evader's Voronoi cell. The time derivative of the area of this cell can be modeled with the equation

$$\frac{dA}{dt} = \frac{\partial A}{\partial x_e}\dot{x}_e + \sum_{i=1}^{N} \frac{\partial A}{\partial x_p^i}\dot{x}_p^i,$$

where x_e represents the position of the evader, A represents the area of a specific robot's Voronoi cell, and x_p^i represents the position of pursuer i. The algorithm's job then becomes to minimize $\frac{\partial A}{\partial x_p^i}\dot{x}_p^i$ for each pursuer i, as outlined above. Interestingly, the authors of [5] prove that in an enclosed environment, pursuers utilizing this Voronoi-based pursuit algorithm are mathematically guaranteed to eventually catch the evader. Together these two pieces made up our pursuit controller.

Finally, the evader uses the Artificial Potential Field (APF) algorithm, which treats path planning as a physics problem with attractive and repulsive vector fields that, when summed together, decide in which direction the evader should move. Primarily, the evader experiences a repulsive force away from the pursuers. This algorithm is outlined in detail in [6].

III. METHODS & IMPLEMENTATION

First, we had to define a custom (and very simple) differential-drive robot for the Gazebo environment to use. We did this by writing a custom URDF file outlining the robot, which described all the important parts of the robot, including the two free-moving wheels on either side, the two caster wheels to keep the robot pitch stable, as well as the simple LiDAR sensor mounted to the top of the base frame (Fig. 1). To evaluate the performance of pure pursuit as well as cooperative pursuit with Voronoi partitions, we had to design a ROS2 package that would allow us to successfully run the algorithms. The general setup of the ROS2 environment is as follows. The most important piece of the simulation is the Manager node, which subscribes to each of the `/robot/pose` topics to gather ground truth pose information for use by our evasion and pursuit algorithms, and stores it in a data structure. This node then publishes this information to the `/sim_state` topic, providing ground truth pose information of all active robots for any node that subscribes to this topic. Each robot that is spawned in has its own controller node, which subscribes to the data published from `/sim_state`. Using these ground truth position values, each controller calculates a linear and angular velocity according to either the pure pursuit or Voronoi partitioning algorithms, which it then uses to create and publish a `Twist` message to the respective `/robot/cmd_vel` topic. Each robot that is spawned in using the launch file gets its own namespace, which is crucial for separating the robot-specific topics. Additionally, we integrated a simple obstacle avoidance algorithm into both the pursuer and evader robots. Each robot controller additionally reads from the `/robot/scan` topic, and reads through this distance data to locate nearby obstacles. In addition to summing repulsive and attractive vectors from other robots, both pursuers and evaders additionally sum a repulsive vector away from any obstacle that the LiDAR picks up, allowing the robots to successfully navigate obstacles.

Each controller reduces to picking a target point and steering toward it. The APF evader at position x_e sums a repulsive contribution from every pursuer p_i and every detected obstacle o_j,

$$F_e = \sum_i k_p \frac{x_e - p_i}{\|x_e - p_i\|^3} + \sum_j k_o \frac{x_e - o_j}{\|x_e - o_j\|^3}, \qquad (1)$$

and drives along F_e, so the push from each source falls off with the square of its distance. A pursuer running pure pursuit aims at a lookahead point set a fixed distance L_d ahead of the evader along its heading $\hat{h}$,

$$g = x_{ev} + L_d\,\hat{h}, \qquad (2)$$

and turns toward g with curvature

$$\kappa = \frac{2\,y_g}{L_d^2}, \qquad \omega = \kappa\,v, \qquad (3)$$

where y_g is the lateral offset of g in the pursuer's own frame and v is its fixed forward speed. A pursuer running the cooperative strategy instead steers toward the midpoint of the Voronoi edge its cell shares with the evader's cell,

$$g = \tfrac{1}{2}\,(e_1 + e_2)\,, \qquad (4)$$

where e_1 and e_2 are the endpoints of that shared edge, reverting to the pure pursuit target in Eq. (2) when no cell borders the evader's.

Our experiment sought to answer two questions, the first of which concerned adversarial performance of various pursuit and evasion algorithms. We planned to implement around 3 algorithms for each party, conduct trials for each pair, and interpret both the best overall performances and the strongest winning and losing matchups. However, for reasons that will be addressed in our challenges section, we were not able to run the amount of tests necessary to have meaningful results. As such, we shifted our focus to our second, more tightly scoped research question. Given an algorithm for each robot, we wanted to analyze the effect of different amounts of pursuers. To do this, we conducted numerous trials for different configurations of pursuers. We simulated trials for the two most efficient algorithms, Voronoi pursuit and APF evasion, under various different pursuer amounts. We collected 10 trials for each number, measuring the time for each and creating a comparison of time to agent count. While this may originally have been a secondary dimension to our primary research designed to showcase the differing coordination efficiency of various pursuit algorithms, we came across interesting results regardless.

To conduct these trials, we needed a way to manage the entire state of the world. On top of publishing state, the Manager node also runs the logic that ends a trial and automates our data collection. At every update it reads the ground-truth poses off `/sim_state` and computes the

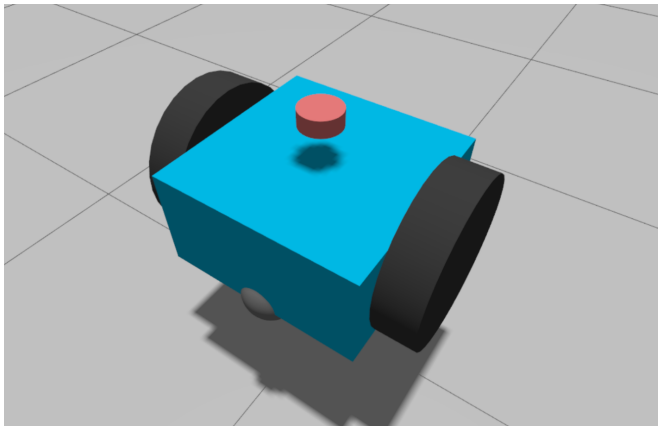

FIG. 1. The custom differential-drive robot used for this project.

distance between the evader and each pursuer, declaring a pursuer win the moment any of those distances drops below our capture radius of 0.9 meters. If no pursuer catches the evader within 50 seconds, the trial is scored as an evader win instead. Because the Manager already owns every pose, it can detect a capture and log how long it took before resetting all of the robots to a new randomized starting configuration for the next run, with no manual intervention between trials. This is what let us batch many trials back to back for a given pursuer count, and it is how we gathered the 10 trials each for the 2, 3, and 4 pursuer cases reported in our results. Holding the update loop at a fixed rate also kept every controller's pursuit and evasion calculations in sync, so the initial placement of the robots was effectively the only variable that changed from one trial to the next.

IV. RESULTS

We ran ten trials per pursuer count with the Voronoi strategy, scoring a pursuer win only when a pursuer reached the evader within the 50 second cutoff and an evader win otherwise (Fig. 3). Pursuer count was the dominant factor, and the result changed abruptly between two and three pursuers rather than changing smoothly with the count. With two pursuers the pursuers never closed within the cutoff in any of the ten trials, so the evader won every time; even when the two pursuers did eventually corner the evader it took far longer than the limit, averaging about 188 seconds. With three pursuers the outcome reversed, with the pursuers winning nine of ten trials and losing only the single run that dragged past the cutoff. With four pursuers the pursuers won all ten.

What stands out is that adding a fourth pursuer barely changed how fast a successful capture happened. Among captures inside the 50 second window, three pursuers averaged 16.0 seconds and four pursuers averaged 15.1 seconds, essentially identical, with both well under the cutoff. The decisive change was moving from two pursuers to three, which flipped the result from a guaranteed evader win to a near-guaranteed pursuer win. We attribute the two pursuer failure to the Voronoi cooperation collapsing into its pure pursuit fallback, since with only two cells the pursuers projected the same lookahead point along the evader's heading, bunched up, and chased from a single direction, which let the evader keep peeling away and stretch the chase past the cutoff.

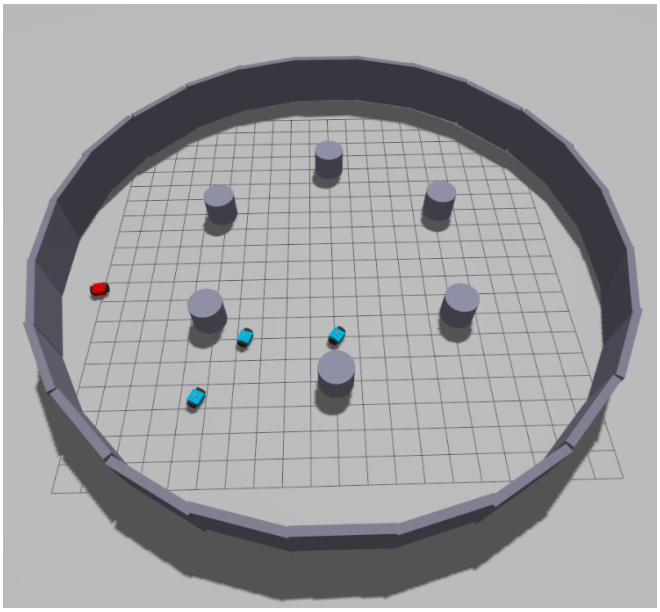

FIG. 2. The Gazebo simulation environment used for all trials. The arena has rounded walls and a small number of static obstacles that both pursuers and evaders must navigate around.

V. CHALLENGES

The hardest challenge we encountered was designing a competent evader. We used a rather naive artificial potential field, where the evader just moves opposite the summed repulsion from the pursuers. This resulted in the evader oscillating in place when the forces roughly cancelled each other out.

It also took us a long time to get the Voronoi pursuit to work. It was quite difficult to implement the algorithm despite its seemingly simple nature as described by the paper. At first glance, computing each agent's cell and finding its shared edge with the evader's cell in real time, then steering toward that edge's midpoint seemed easy, but it proved to be considerably more involved as we moved toward implementation. We encountered a variety of bugs that caused the pursuers to act strangely and unexpectedly, freezing into place or locking onto wrong target points at times for reasons that initially seemed unclear.

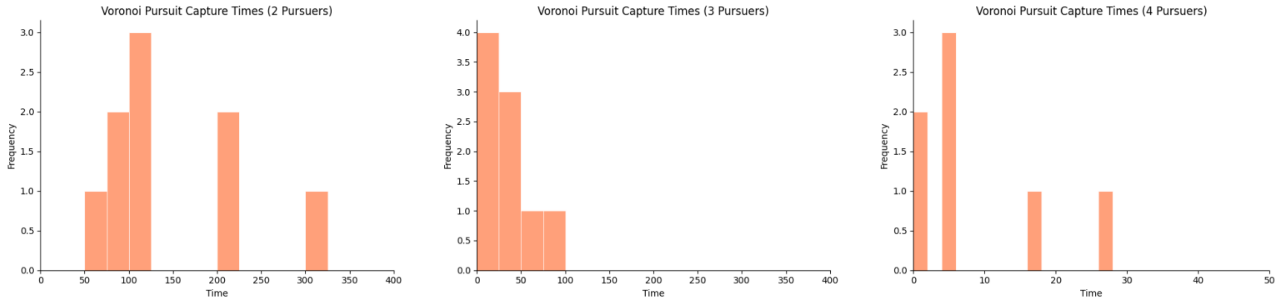

FIG. 3. Visualized capture times with 2, 3, and 4 pursuers using Voronoi cooperative pursuit.

We also struggled with designing an appropriate Gazebo environment for the robots. Initially, we had a rectangular environment with sharp corners, which ended up allowing the pursuer to corner the evader almost instantaneously every single trial no matter the setup of the initial variables. As a result, we redesigned the arena with rounded corners and more open space (Fig. 2), so the evader could not get pinned against a hard angle and always had somewhere to keep moving. This made the outcome depend on the pursuers' strategy and number rather than the shape of the walls, which is what finally let the differences between pure pursuit and Voronoi pursuit actually show up in our results.

VI. CONCLUSION

We built a hide-and-seek simulation in Gazebo where 2 to 4 differential drive pursuers try to capture a single evader inside a bounded, circular area. We compared two pursuer strategies, pure pursuit and cooperative Voronoi partition pursuit against an artificial potential field evader, and measured time to capture across pursuer counts. We found that both cooperation and numbers heavily influence the outcomes. Voronoi pursuit outperformed its pure pursuit fallback once enough pursuers were present, because the partitioning made them approach from distinct directions and corner the evader rather than clustering behind it as they did when the cooperation broke down at two pursuers. We also found that the decisive change was the jump from two to three pursuers. Two pursuers never caught the evader inside the 50 second window, while three and four pursuers won almost every trial at a near-identical average capture time of about fifteen seconds, so adding a fourth pursuer did little to speed up a successful capture. We additionally found that APF evaders do not have any counter-strategy when encircled.

Given more time, we believe future work could incorporate a smarter, less reactive evader since most of the pursuer wins came from the evader drifting into a corner rather than the pursuers truly outplaying them. On the pursuer side we would add explicit coordination, such as assigning a flanker or blocker role. We would also make the Voronoi pursuit obstacle-aware so pursuers route around walls instead of driving into them. Lastly, having the pursuers localize the evaders through LiDAR data rather than refer to ground-truth poses would make the problem much harder and much more realistic, akin to a real pursuit setting.

[1] R. Isaacs, *Differential Games* (John Wiley and Sons, 1965).
[2] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, Robot operating system 2: Design, architecture, and uses in the wild, Science Robotics **7**, eabm6074 (2022).
[3] N. Koenig and A. Howard, Design and use paradigms for Gazebo, an open-source multi-robot simulator, in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 3 (Sendai, Japan, 2004) pp. 2149–2154.
[4] R. C. Coulter, *Implementation of the Pure Pursuit Path Tracking Algorithm*, Tech. Rep. CMU-RI-TR-92-01 (Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, 1992).
[5] Z. Zhou, W. Zhang, J. Ding, H. Huang, D. M. Stipanović, and C. J. Tomlin, Cooperative pursuit with Voronoi partitions, Automatica **72**, 64 (2016).
[6] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, The International Journal of Robotics Research **5**, 90 (1986).