Pursuit Evasion Game: A comparison study between RRT* and FMT*

Saud Alsaif
salsaif@stanford.edu

Joseph Campion
jcampion@stanford.edu

Yehia Khoja
ykhoja@stanford.edu

Abstract

The pursuit evasion game is considered among the class of differential games, where, the agents have a continuous state space that is characterized by differential dynamics, but have conflicting goals. In simple instances, such a game can be solved analytically; however, in more complex versions, fast and reliable numerical methods are needed to find a solution path quickly. Currently, there are multiple options for such algorithms. In this paper, we have solved the pursuit evasion game using the two main algorithms, RRT* and FMT*, in order to compare their performance. Even though both are known to be fast, and asymptotically optimal, this study quantifies the differences in the optimal path found, and the average run time, showing that FMT* is better in both regards.

1 INTRODUCTION

Pursuit-evasion games are used to model conflict arising between dynamic agents with opposing interests. Initially, this was primarily used for military and combative applications such as missile guidance [1], collision avoidance [2], air combat [3], and path planning with the presence of an adversary [4]. Now, it is used in various fields like modeling stochastic feedback Nash equilibrium in economics [5]. In simple cases, such problem can be solved analytically. For example, solutions exist for the "homicidal chauffeur" [1] and the "lady in the lake" [14] problems. However, an analytical solution is harder to find in problems where the agents have more complex dynamics, or there are obstacles that need to be avoided in the state space. As a result, fast and reliable numerical methods are needed to solve such problems, and return an optimal path, if one exists. Currently, there are multiple types of algorithms that can numerically provide a solution. For example, several algorithms have been proposed in the context of mobile robotics, where most commonly planning is based on a 'space-time' state space, avoiding spatio-temporal regions representing possible motions of the dynamic obstacles [9]. Furthermore, several types of pursuit-evasion were studied from an algorithmic perspective. Some examples include, pursuit evasion games on graphs [10], and on timed roadmaps [11]. However, all these approaches impose severe limitations on the allowable agents' dynamics. For this, we consider in this paper sampling-based motion planning algorithms, which sample points in the state space, and analyze the feasibility and optimality of reaching, or moving towards these points. Examples of such algorithms are Probabilistic RoadMaps (PRMs) [12], Rapidly-exploring Random Trees (RRT) [13], a modified version of RRT, RRT* [6], and Fast Marching Trees (FMT*) [8]. Despite sharing the same underlying approach, these algorithms can vary in the way they collect samples. For example, RRT collects a new sample point with each step, and then analyzes whether the tree should be extend towards that point; whereas, FMT* collects a batch of samples across the state space, and then analyzes all the points collectively to determine the best path to reach each node. Furthermore, sampling based algorithm can vary depending on the number of paths they return. For example, RRT is a single-query algorithm which returns only a single path connecting the agent's initial position to a goal region. On the other hand, multi-query algorithms such as PRM provide paths to each sampled point, and therefore connect the agent's initial path to any point in the sample, including the goal region. This poses a significant differences in

Submitted as a final project to complete the requirements of AA203 at Stanford University.

the computational requirements needed if a path to a new goal region is queried. A single-query algorithm would have to recalculate, but a multi-query algorithm would simply reroute using the existing information in the tree.
Nevertheless, in this paper we aim to quantify the differences in performance of two main algorithms of each type, namely RRT*, single-query, and FMT*, multi-query. To do this, we will solve a pursuit-evasion game with only two agents, an evader and a pursuer, using both algorithms. In section 2, we present the problem formulation, where we describe the problem setup, dynamics, and assumptions. In section 3, we solve the pursuit-evasion game using both algorithms, RRT* and FMT*, as well as describe the algorithms. In section 4, we will present the results of our simulation and discuss our findings. Lastly, in section 5 we present a conclusion as well as possible extensions to the work done in this paper.

2 PROBLEM FORMULATION

The pursuit evasion game is considered among the class of differential games. The concept of differential games was introduced by R. Isaacs in [1]. A differential game is a game where the agents have continuous state space, characterized by differential dynamics. The version of pursuit evasion game that we consider in this paper is one that involves two agents, an evader and a pursuer. The evader's goal is to find a path from the initial state to a specified goal state in minimum time. The pursuer, on the other hand, tries to capture the evader before it reaches the goal state. Put formally, we consider a time invariant dynamical system characterized by the differential equation $\dot{x}(t) = f(x(t), u_e(t), u_p(t))$, where $x : t \mapsto x(t)X \in \mathbb{R}^d$ is the state trajectory, $u_e : t \mapsto u_e(t)U_e \in \mathbb{R}^{m_e}$ is the evader's control input, $u_p : t \mapsto u_p(t)U_p \in \mathbb{R}^{m_p}$ is the pursuer's control input. The sets X, U_e, and U_p are assumed to be compact, the control signals u_e and u_p are essentially bounded measurable, and $f(z, w_e, w_p)$ is locally Lipschitz in z and piecewise continuous in both w_e and w_p. There are also the obstacle set X_{obs}, a goal set X_{goal}, and a capture set X_{capt}. All are assumed to be open subsets of X, X_{goal} and X_{capt} are disjoint.
A unique state trajectory can be computed for the evader given an initial condition $x(0) \in X \setminus X_{obs}$ and the control inputs of the evader and the pursuer.The final time of the game is given by $T = \inf\{t \in \mathbb{R}_{\geq 0} : x(t) \in cl(X_{goal} \cup X_{capt})\}$. One objective function is considered for this zero-sum game, defined as: $L(u_e, u_p) = T$, if $x(T) \in cl(X_{goal})$; and $L(u_e, u_p) = +\infty$, otherwise. The evader tries to minimize the objective function by escaping to the goal region in minimum time, while the pursuer tries to capture the evader to maximize the objective.
In this pursuit evasion game, we will assume that the problem possesses a separable structure. That is, the states of the pursuer and the evader can be partitioned as $x = (x_e, x_p) \in X_e \times X_p = X$, the obstacle set can also be partitioned as $X_{obs} = (X_{obs,e} \times X_p) \cup (X_p \times X_{obs,e})$, where $X_{obs,e} \subset X_e$ and $X_{obs,p} \subset X_p$, the goal set is partitioned as: $X_{goal} = (X_{e,goal} \times X_p) \setminus X_{capt}$, where $X_{e,goal} \subset X_e$, and the dynamics are decoupled as follows:

$$\frac{d}{dt}x(t) = \frac{d}{dt}\begin{bmatrix} x_e(t) \\ x_p(t) \end{bmatrix} = f(x(t), u(t)) = \begin{bmatrix} f(x_e(t), u_e(t)) \\ f(x_p(t), u_p(t)) \end{bmatrix}, \text{ for all } t \in \mathbb{R}_{\geq 0}.$$

An algorithm for the solution of the pursuit evasion game is said to be *sound* if it returns a control input $\bar{u}_e$ such that $\max_{u_p} L(\bar{u}_e, u_p)$ is finite. An algorithm is said to be *complete* if it terminates in finite time returning a solution $\bar{u}_e$ if one exists or it reports failure.
In pursuit evasion games the pursuer(s) can be modeled as dynamic obstacles whose trajectories are unknown a priori. It is known that the motion planning problem with moving obstacles is NP-hard, which suggests that complete algorithms for the pursuit evasion game will be computationally intensive. That is why the authors in [7] have proposed to use sampling-based algorithms that are probabilistically sound and probabilistically complete. In our paper, we aim to use another motion planning algorithm that possesses the same probabilistic properties as the one proposes in [7], but is shown to find a more optimal solution in less time, as proven in [8].

3 ALGORITHMS

In this section, both RRT* and FMT* are used to solve the pursuit evasion game. Pursuit evasion with RRT* was implemented by Karaman and Frazzoli in [7]. The contribution of this paper is a

similar implementation of the pursuit evasion game using FMT*, which has been show to be faster and result in more optimal solutions than RRT* [8]. The RRT* algorithm without pursuit evasion can also be found in [7].

3.1 Pursuit Evasion with RRT*

The pursuit evasion game implemented with RRT* is in Algorithm 1. For a certain number of iterations N, the tree for both the pursuer and evader are constructed by extending one branch at a time. If a node of the evader tree is within the capture radius used by the function `NearCapture`, and the cost to reach the node of the pursuer tree is less than that of the evader node, then the node is removed from the evader tree.

Algorithm 1 Pursuit Evasion with RRT*

$V_e' \leftarrow \{x_{e,init}\}$; $E_e' \leftarrow \emptyset$; $V_p' \leftarrow \{x_{p,init}\}$; $E_p' \leftarrow \emptyset$;
while i<N **do**
  $G_e \leftarrow (V_e, E_e)$; $G_p \leftarrow (V_p, E_p)$;
  $z_{e,rand} \leftarrow \texttt{Sample}_e(i)$;
  $(V_e, E_e, z_{e,new}) \leftarrow \texttt{Extend}_e(G_e, z_{e,rand})$
  if $z_{e,new} \neq$ NULL **then**
    $Z_{p,near} \leftarrow \texttt{NearCapture}(G_p, z_{e,new}, |V_p|)$
    for all $z_{p,near} \in Z_{p,near}$ **do**
      if $\texttt{Time}(z_{p,near}) \leq \texttt{Time}(z_{e,new})$ **then**
        $\texttt{Remove}(G_e, z_{e,new})$
      end if
    end for
  end if
  $z_{p,rand} \leftarrow \texttt{Sample}_p(i)$
  $(V_p, E_p, z_{p,new}) \leftarrow \texttt{Extend}_p(G_p, z_{p,rand})$
  if $z_{p,new} \neq$ NULL **then**
    $Z_{e,near} \leftarrow \texttt{NearCapture}(G_e, z_{p,new}, |V_e|)$
    for all $z_{e,near} \in Z_{e,near}$ **do**
      if $\texttt{Time}(z_{p,new}) \leq \texttt{Time}(z_{e,near})$ **then**
        $\texttt{Remove}(G_e, z_{e,near})$
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while
return G_e, G_p

3.2 Pursuit Evasion with FMT*

The procedure for FMT* from [8] is found in Algorithm 2. The contribution of this paper is Algorithm 3, which shows an implementation of FMT* in the pursuit evasion game. First, all of the candidate nodes for each tree are identically and independently sampled from a uniform distribution on X_{free}. Here it is assumed that the evader and pursuer share the same free state space. Then, the trees for both the pursuer (T_p) and the evader (T_e) are constructed using FMT* (lines 2-3). Also returned by the `BuildTreeFMT` function are the indices of the nodes for each tree. In lines 4-8, for each node that is shared by both trees, if the cost to reach that node for the pursuer is less than that of the evader, then the node is removed from the evader tree. In the `RemoveBranch` function, all of the descendant nodes are also removed from the tree starting at x. Next, the node with the lowest cost-to-go function within the goal region is determined (lines 9 and 13). Finally, the optimal path to the goal region is returned (line 15). If a path does not exist, Failure is returned by the algorithm (lines 10-11).

Algorithm 2 $\texttt{BuildTreeFMT}_\alpha(x_{\alpha,init}, X_{sample})$

$V_\alpha \leftarrow \{x_{\alpha,init}\} \cup X_{sample}$; $E_\alpha \leftarrow \emptyset$;
$W_\alpha \leftarrow V_\alpha \setminus \{x_{init}\}$; $H_\alpha \leftarrow \{x_{\alpha,init}\}$
$z \leftarrow x_{\alpha,init}$
$N_z \leftarrow \texttt{Near}(V \setminus \{z\}, z, r_n)$
while $i < N$ **do**
    $H_{new} \leftarrow \emptyset$
    $X_{near} \leftarrow \texttt{Intersect}(N_z, W)$
    for $x \notin X_{near}$ **do**
        $N_x \leftarrow \texttt{Near}(V_\alpha \setminus \{x\}, x, r_n)$
        $Y_{near} \leftarrow \texttt{Intersect}(N_x, H_\alpha)$
        $y_{min} \leftarrow \arg\min_{y \in Y_{near}} \{\texttt{Cost}(y, T_\alpha = (V_\alpha, E_\alpha)) + \texttt{Cost}(\overline{yx})\}$
        if $\texttt{CollisionFree}(y_{min}, x)$ **then**
            $E_\alpha \leftarrow E_\alpha \cup \{(y_{min}, x)\}$
            $H_{new} \leftarrow H_{new} \cup \{x\}$
            $W_\alpha \leftarrow W_\alpha \setminus \{x\}$
        end if
    end for
    $H_\alpha \leftarrow (H_\alpha \cup H_{new}) \setminus \{z\}$
    if $H_\alpha = \emptyset$ **then**
        return Failure
    end if
    $i \leftarrow i + 1$
end while
return $T_\alpha = (V_\alpha, E_\alpha)$, H_α

Algorithm 3 Pursuit Evasion with FMT*

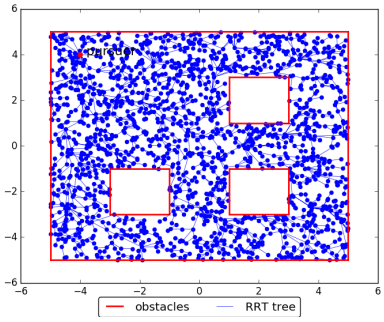
$X_{sample} \leftarrow \texttt{Sample}(n)$
$(T_e = (V_e, E_e), H_e) \leftarrow \texttt{BuildTreeFMT}_e(x_{e,init}, X_{sample})$
$(T_p = (V_p, E_p), H_p) \leftarrow \texttt{BuildTreeFMT}_p(x_{p,init}, X_{sample})$
for $x \in H_e \cap H_p$ **do**
    if $\texttt{Cost}(x, T_p = (V_p, E_p)) \leq \texttt{Cost}(x, T_e = (V_e, E_e))$ **then**
        $\texttt{RemoveBranch}(x, T_e = (V_e, E_e))$
    end if
end for
$Z_{goal} \leftarrow \texttt{Intersect}(X_{goal}, H_e)$
if $Z_{goal} = \emptyset$ **then**
    return Failure
else
    $z \leftarrow \arg\min_{y \in Z_{goal}} \{\texttt{Cost}(y, T_e = (V_e, E_e))\}$
end if
return $\texttt{Path}(z, T_e = (V_e, E_e))$

4 SIMULATION RESULTS AND DISCUSSION

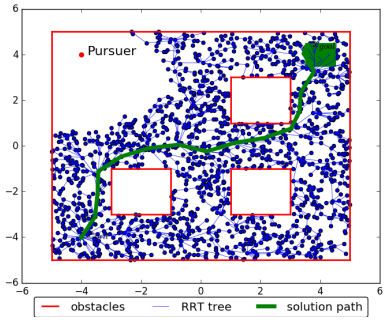
In this section, we present our simulations for the pursuit evasion game using both RRT* and FMT*. The performance of each algorithm is compared based on two metrics:

- The execution time against several numbers of iterations.
- The cost of the optimal path returned by the algorithm.

The environment of the simulation was set up as follows: the state space is the region defined by $[-5, 5] \times [-5, 5]$, the initial position of the evader $x_e(0) = [-4, -4]$ and the pursuer $x_p(0) = [-4, 4]$, with the goal region as $[3.5, 4.5] \times [3.5, 4.5]$, and three obstacle regions: $[-3, -1] \times [-3, -1]$, $[3, 1] \times [-3, -1]$, and $[3, 1] \times [3, 1]$. The pursuer and the evader are assumed to have Euclidean dynamics, and the cost of each edge in the tree is the Euclidean distance between the nodes that

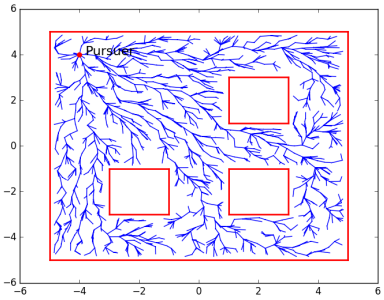


(a) Pursuer's RRT* tree

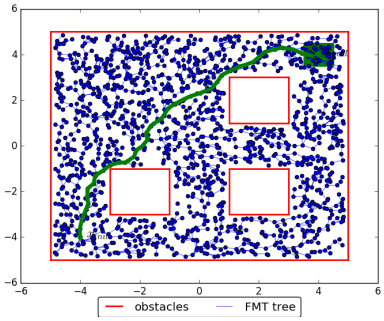


(b) Evader's RRT* tree after removing X_{capt}

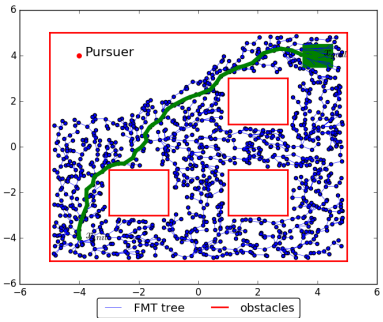
Figure 1: The simulation of the pursuit evasion game with RRT*, any node in the capture set is removed along with its children. The pursuer's and the evader's trees are built simultaneously.



(a) Pursuer's FMT* tree



(b) Evader's FMT* tree

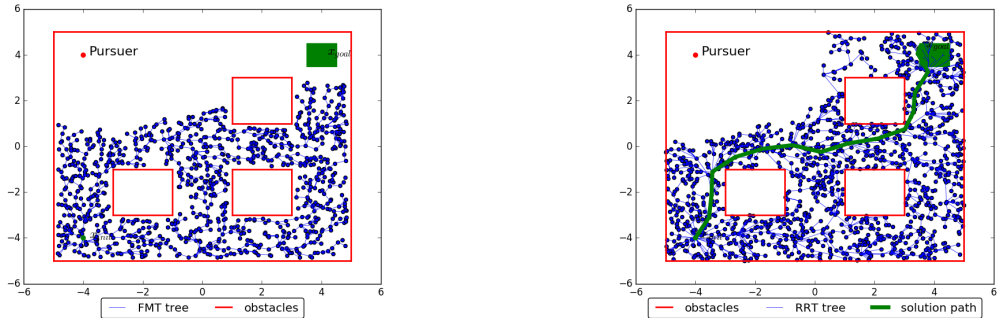


(c) Evader's tree after removing unsafe nodes.

Figure 2: The simulation of the pursuit evasion using FMT*, any node in the capture set is removed along with its children. The pursuer's and the evader's trees are built sequentially.

define it. The General cost of each node x_i in the tree is the sum of all edges from $x(0)$ to x_i. In the main simulation, the pursuer is assumed to have half the evader's speed. The resulting RRT* trees and optimal path are illustrated in Figure 1, while the resulting FMT* trees and optimal path are illustrated in Figure 2. The figure shows that the resulting path from FMT* is shorter than that of RRT*. In Figure 3 an average of 10 runs for each RRT* and FMT* show that FMT* builds the evader's optimal path in a shorter time, also the cost of FMT* path is more optimal than RRT*'s. This means that an evader that employs FMT* to find the evasion path is going to find it in a shorter time than if it used RRT*, this result is extended in the case of multiple pursuers, since it will only mean additional single FMT* run times, which are shown in [8] to be more efficient than RRT*. However, there is a drawback to using FMT* sequentially as we have presented, this drawback is illustrated in Figure 4. When the pursuer's speed is increased, the capture set increases, but RRT* manages

Figure 3: a) The optimal path cost of each algorithm with different numbers of iterations, b) The execution time of each algorithm with different number of iterations. FMT* is shown to be a superior solver for the pursuit evasion game in terms of optimality and in terms of run time.



(a) Evader's FMT* tree with fast pursuer.

(b) Evader's RRT* tree with fast pursuer.

Figure 4: The pursuit evasion game with a faster pursuer, FMT* failed to find a solution while RRT* maintained its suboptimal solution.

this increase better than FMT*, since FMT* uses a fixed sample set, so when a node is removed, its children are not rewired to other nodes that are not in the capture set, they are removed with their parent node. This procedure limits FMT* flexibility to accommodate expanding capture sets. One remedy to this would be to build the evader's tree with the knowledge of the pursuer's tree, so when a sample is in the capture set, it will simply not be added to the evader's tree, which will give the evader's tree more flexibility in dealing with the expanding capture set.

5 CONCLUSIONS

In this paper, we reviewed the work done on the pursuit evasion game by Karaman and Frazzoli in [7], where they proposed a solution based on RRT*. We replicated their results, and proposed and showed that solving the game using FMT* gives a more optimal solution in a shorter time, and that solution would scale more efficiently with the number of nodes and the number of pursuers. Extensions to this work include: improving the FMT* implementation to make it more robust and flexible to expanding capture sets, implementing the algorithms on agents with more complex dynamics like Dubins car, and finally relaxing the separable dynamics assumption, and investigating the game where the pursuer's dynamics depend on the evader's trajectory, which will make the problem harder but more applicable to adversarial environments. The code associated with this paper can be found at `https://github.com/salsaif/AA-203-project`.

References

[1] R. Isaacs. *Differential Games*. Wiley, 1965.

[2] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. IEEE Transactions on Automatic Control, 50(7):947–957, 2005.

[3] R. Lachner, M. H. Breitner, and H. J. Pesch. Three dimensional air combat: Numerical solution of complex differential games. In G. J. Olsder, editor, New Trends in Dynamic Games and Applications, pages 165–190. Birkhauser, 1995.

[4] O. Vanek, B. Bosansky, M. Jakob, and M. Pechoucek. Transiting areas patrolled by a mobile adversary. In Proc. IEEE Conf. on Computational Intelligence and Games, 2010.

[5] Leong, C. K.; Huang, W. (2010). "A stochastic differential game of capitalism". Journal of Mathematical Economics. 46 (4): 552. doi:10.1016/j.jmateco.2010.03.007

[6] Karaman, S., and Frazzoli, E. (2011). *Sampling-based algorithms for optimal motion planning.* The international journal of robotics research, 30(7), 846-894.

[7] Karaman, S., and Frazzoli, E. (2010). *Incremental sampling-based algorithms for a class of pursuit-evasion games.* In Algorithmic foundations of robotics IX (pp. 71-87). Springer Berlin Heidelberg.

[8] Janson, L., Schmerling, E., Clark, A., and Pavone, M. (2015). *Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions.* The International journal of robotics research, 34(7), 883-921.

[9] J. C. Latombe. Robot Motion Planning. Kluwer Academic Publishers, Boston, MA, 1991.

[10] T. Parsons. Pursuit-evasion in a graph. In Theory and Applications of Graphs, pages 426–441. 1978.

[11] V. Isler, D. Sun, and S. Sastry. Roadmap based pursuit-evasion and collision avoidance. In Robotics: Science and Systems Conference, 2005.

[12] L.E. Kavraki, P. Svestka, J Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4):566–580, 1996.

[13] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. International Journal of Robotics Research, 20(5):378–400, May 2001.

[14] T. Basar and G.J. Olsder. Dynamic Noncooperative Game Theory. Academic Press, 2nd edition, 1995.