Robust-by-Design Plans for Multi-Robot Pursuit-Evasion

Trevor Olsen, Nicholas M. Stiffler, and Jason M. O'Kane

Abstract— **This paper studies a multi-robot visibility-based pursuit-evasion problem in which a group of pursuer robots are tasked with detecting an evader within a two dimensional polygonal environment. The primary contribution is a novel formulation of the pursuit-evasion problem that modifies the pursuers' objective by requiring that the evader still be detected, even in spite of the failure of any single pursuer robot. This novel constraint, whereby two pursuers are required to detect an evader, has the benefit of providing redundancy to the search, should any member of the team become unresponsive, suffer temporary sensor disruption/failure, or otherwise become incapacitated. Existing methods, even those that are designed to respond to failures, rely on the pursuers to replan and update their search pattern to handle such occurrences. In contrast, the proposed formulation produces plans that are inherently tolerant of some level of disturbance. Building upon this new formulation, we introduce an augmented data structure for encoding the problem state and a novel sampling technique to ensure that the generated plans are robust to failures of any single pursuer robot. An implementation and simulation results illustrating the effectiveness of this approach are described.**

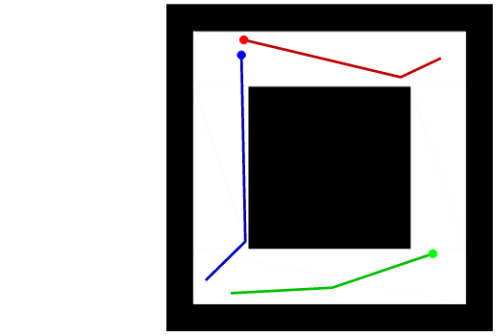
I. Introduction

Pursuit-evasion is a two-player game where the players are diametrically opposed. Members of one team, called the *pursuers*, actively seek out members of the second team, the *evaders*. The pursuers' goal is to capture (either physically or visually) the evaders; the evaders wish to evade capture for as long as possible.

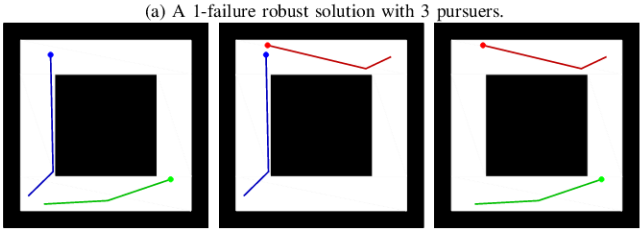
A number of tasks can be modeled as pursuit-evasion problems albeit with differing levels of antagonism displayed by the evaders. Some such examples include search-and-rescue scenarios (rescuers/rescuee), agriculture/livestock monitoring (stockman/predator), and active pursuit (hunter/prey), among many others. With the proliferation of robots into various domains, we have seen instances where robots are used in all of the aforementioned scenarios to perform the role of the pursuers [2], [4], [7].

This paper addresses a form of the pursuit-evasion problem where the pursuers' task is to establish visibility with all evaders that exist in a given environment. These games take place in a two-dimensional space in which each pursuer is capable of detecting any evader within its line of sight.

Specifically, this paper investigates how one might generate strategies for the pursuers that are robust in the face of malfunctions which may inhibit members of the team, as shown in Figure 1. A few such defects include utter failure, intermittent sensor failure (i.e. false negatives), and incorrectly transmitted information. To the authors' knowledge, the literature regarding failures in the realm of pursuit-evasion is scant. Our own previous work considered the situation where members of the pursuer team suffered complete and catastrophic failure, necessitating that the pursuer strategy be re-planned online [17]. This replanning step introduces execution delays and requires all of the remaining pursuers to be aware of failures when and where they occur. This paper addresses a larger range of potential robotic defects by generating joint motion strategies for the pursuers which guarantee detection of the evaders, regardless of any single pursuer malfunction and without the need to detect such malfunctions when they occur nor to halt the current execution to update the pursuers' motion strategies.



(a) A 1-failure robust solution with 3 pursuers.



(b) Red failure (c) Green failure (d) Blue failure

Fig. 1: A simple example of a 1-failure robust solution for $n = 3$ robots. Removal of any single pursuer does not preclude the remaining pursuers from capturing an evader in the $n = 2$ subsolution.

Beyond the previously-studied pursuit-evasion component, which must reason about the regions of the environment which may contain an evader, the major challenge in generating such 'failure-robust' joint pursuer strategies is the need to track these 'contaminated' regions separately for each possible pursuer failure, while ensuring that a single solution can be extracted when the process is complete. Note that the best known complete algorithm for multi-robot visibility-based pursuit-evasion solves the problem in

T. Olsen and J. M. O'Kane are with the Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA. N. M. Stiffler is with the Department of Computer Science, University of Dayton, Dayton, OH 45469, USA. tvolsen@email.sc.edu, jokane@cse.sc.edu, nstiffler1@udayton.edu This material is based upon work supported by the National Science Foundation under Grant Nos. 1659514 and 1849291.

time doubly exponential in the number of pursuer agents. To address this shortcoming, sampling-based methods have been developed [18], [23] that utilize a graph structure whose vertices encode the pursuers pose information in addition to the regions of the environment where the evader may be; whereas edges indicate feasible pursuer motions within the environment.

This paper builds on this line of sampling-based approaches by providing an augmented graph structure that encodes the 'robustness' of the search (i.e. reasoning over any possible pursuer failure). In response to this added layer of complexity, a novel sampling method is introduced which creates condensed sample sets that ensure capture of the evader via a highly connected roadmap.

The remainder of the article begins with a review the related literature in Section II. Next, a formal description of the problem is provided in Section III. Section IV outlines the algorithm utilized to generate robust joint motion strategies. Simulation results, in which the algorithm was employed in several different representative environments, are described in Section V before the paper concludes with a summary and look towards future work Section VI.

II. Related work

Pursuit-evasion problems have a rich history in the fields of differential game theory [10], [11], graph theory [20], and geometric settings [9], [26]. In the context of graph theory, these problems are typically referred to as discrete pursuit-evasion games. One possible scenario for such a game requires the pursuer agent(s) to occupy the same vertex as the evader. Parsons introduced this problem on finite graphs [20]. Interest in graph theoretic pursuit-evasion problems grew quickly, spanning a broad range of variations such as: unrestricted evader movements [1] and infinite graphs [16].

Pursuit-evasion also has a rich history in the geometric/continuous domain [12], [21], [29]. The continuous nature of geometric environments allows for several different types of capture conditions, including those based on visibility [26]. The visibility-based pursuit-evasion game has been studied in extensive detail for both the case of a single pursuer as well as a team of several pursuers. For the single pursuer case, results include: completeness [9], solvability conditions [19], limited field of view [6], bounded evader velocity [27], optimality [24] and robustness [25].

Contrary to what one might expect, the addition of extra pursuer agents does not necessarily make the problem any easier. As the number of pursuers increases, the dimensionality of the problem increases creating a computational challenge. This concern with dimensionality has parallels in many areas of robotics. In path and motion planning, for example, sampling-based approaches have proven remarkably useful [13], [14]. This approach has also emerged in multi-robot pursuit-evasion problems, where sampling [18], [22] was able to successfully combat the complexity of a complete algorithm [23]. Other research in multi-robot pursuit-evasion includes the lack of environmental knowledge [15] and heuristics for the worst-case [8].

Fault tolerance and recovery is an active research thread in the robotics community as well [3], [5], [28], but less so in the context of pursuit-evasion. The authors previously introduced a scheme to handle complete and catastrophic failures [17]. The novelty of this paper is to address the situation where one of the pursuers is assumed to be faulty, without the need to detect the fault nor replan on the fly. In particular, we consider malfunctions such as complete failure with and without indication, potential erroneous information reporting, whether it be malicious or accidental and possible physical limitations leaving the robot immobilized for the remainder of the search.

III. Problem statement

A. The Environment, Pursuers and Evaders

The environment, F, is a closed, bounded and polygonal subset of $\mathbb{R}^2$. A team of n pursuers move continuously in F at a bounded speed. Given locations $w_1, \ldots, w_n$ for each of the n pursuers, we call the vector $\langle w_1, \ldots, w_n \rangle$ a joint pursuer configuration (JPC). We denote the location of the i^{th} pursuer at time t by the continuous function $f_i(t) : [0, \infty) \to F$. We call each such function f_i a motion strategy. Each pursuer is equipped with an omnidirectional sensor which extends to the nearest point on the boundary of the environment. For a pursuer located at $q \in F$, the visibility polygon, $V(q)$, is the set of points $r \in F$ such that the line segment $\overline{qr}$ is contained in F.

Similar to pursuers, evaders move continuously throughout F but differ in that they are capable of reaching arbitrarily high speeds. Since the objective of the pursuers is to locate the evaders regardless of the trajectories taken by the evaders, we can assume, without loss of generality, there exists a single evader. We denote the location of this evader at time t by the continuous function $e(t) : [0, \infty) \to F$. The pursuers' objective is to guarantee to locate the evader, as formalized in the next definition.

Definition A collection of motion strategies $X = \langle f_1, \ldots, f_n \rangle$ is called a *solution* if, for any evader curve e, there exists a time $t \geq 0$ such that $e(t) \in \bigcup_{i \leq n} V(f_i(t))$.

Figure 2 exemplifies these definitions. We are interested in solutions that can still guarantee the detection of the evader, even if a single pursuer robot fails.

Definition Given a set of robots, A, where $|A| > k$, a solution X is *k-failure robust* if it remains a solution utilizing only the robots $A \setminus B$, for any $B \subset A$ with $|B| \leq k$.

Notice that, according to these definitions, a solution is a 0-failure robust solution. In this paper, we address the problem of generating 1-failure robust solutions.

B. Shadows and Their Events

The ideas presented in the following two subsections, which discuss the unseen regions of the environment, were initially presented by Guibas, Latombe, LaValle, Lin, and Motwani [9] in the context of a single-robot version of this problem. Because our algorithm, specifically the rSG-PEG

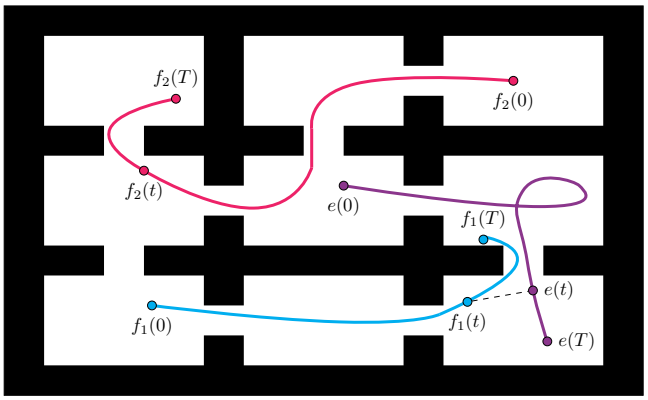


Fig. 2: Two pursuers actively search for the evader for T seconds. At time t, the pursuer moving along path f_1 has visibility of the evader.

data structure described in Section IV-A, uses these ideas in important ways, we summarize them here; details appear in the original paper.

1) Shadows: For a fixed time t, the *shadow region* is the set $S(t) = F \setminus \bigcup_{i \leq n} V(f_i(t))$. The maximally connected components of the shadow region are called *shadows*. Based on the pursuers' movement up until time t, it is either possible or not for an evader to be in a certain shadow while still undetected. In the case that a shadow could contain an undetected evader, the *status* of the shadow is *contaminated*, otherwise the shadow's status is *cleared*. We concisely summarize the status of each shadow at a given time by the *shadow label*, a binary string of length equal to the number of shadows. In a shadow label, a 1 bit represents a contaminated shadow, while a 0 bit indicates that the corresponding shadow is cleared. A shadow label consisting of all zeroes indicates that all shadows have been cleared, making it impossible for an undetected evader to be anywhere within the environment, thus resulting in a solution.

It will be useful to consider a notion of dominance between shadow labels at the same JPC. For two shadow labels $\ell = (a_1 a_2 \cdots a_m)$ and $\ell' = (b_1 b_2 \cdots b_m)$, ℓ is said to *dominate* ℓ' if $\ell \neq \ell'$ and for each $i = 1, \ldots, m$, $a_i \leq b_i$. That is, if a shadow is contaminated in ℓ, it must also be contaminated in ℓ' and any cleared shadow in ℓ' must also be cleared in ℓ.

2) Shadow Events: As pursuers move within F, the shadows will continually change shape and size. When the number of shadows changes, these occurrences are called *shadow events*, which can be classified into four types:

- *Appear*: A shadow can appear if a previously seen subset of the environment falls out of the visibility polygon of all of the pursuers. In this case, the newly formed shadow is marked as clear.
- *Disappear*: If a pursuer gains vision of an entire shadow, the shadow disappears. Here, the shadow is completely removed from the shadow label.
- *Merge*: Two or more shadows merge if they become a single connected component. When this occurs, the newly merged shadow is given the cleared label if and only if all of the merging are cleared.
- *Split*: If a pursuer's visibility polygon disconnects an existing shadow, we say the shadow was split. Both post-split shadows are given the status of the pre-split shadow.

IV. Algorithm Overview

This section describes an algorithm to generate a 1-failure robust solution for a given environment and number of pursuers.

The algorithm builds upon earlier sampling-based approaches to visibility-based pursuit-evasion [23]. The basic idea in that prior work is to construct a roadmap data structure that represents the pursuers' ability to move through the environment and to clear various collections of shadows. As JPCs are sampled and inserted into the roadmap, the prior algorithm tracks a set of reachable shadow labels attached to each vertex. When the data structure determines that an all-clear shadow label is reachable at some vertex, the algorithm extracts that solution and terminates successfully.

The algorithm we propose here differs from that baseline in three important ways, necessitated by the need to produce robust solutions. First, the data structure is augmented to track shadow labels not for all n robots, but instead for each of the n distinct subsets of size $n-1$. This ensures that, if *all* of these shadow labels achieve an all clear status, the resulting solution will be 1-failure robust. See Section IV-A. Second, we introduce a geometric caching optimization to the data structure, based on the observation that the number of times shadow labels are propagated across each edge is substantially higher than in prior settings. Section IV-B describes this change. Finally, we introduce a new sampling scheme tailored to the specific need for solutions in which at least two robots clear each shadow (Section IV-C).

A. Robust Sample-Generated Pursuit-Evasion Graphs (rSG-PEG)

Here, we describe how we enhanced the existing sample-generated pursuit-evasion graph (SG-PEG) data structure [23] to generate 1-failure robust solutions. An rSG-PEG, G, is a directed graph in which one vertex is designated as the root. Each vertex of G is labeled with a JPC; a directed edge $u \to w$ indicates that there exists a coordinate-wise straight line, collision free movement between the JPCs of u and w. For the sake of compactness, a vertex with JPC $w_1, \ldots, w_n$ will be named w.

Each vertex is also associated with a collection of *failure shadow labels*, each of which is an n-tuple $L = (\ell_1, \ldots, \ell_n)$, where each ℓ_i is a single shadow label. The interpretation is that the existence of failure shadow label $(\ell_1, \ldots, \ell_n)$ at vertex w implies that there exists a path in G from the root to w, such that for each $1 \leq i \leq n$, the pursuers in $\{1, \ldots, n\} \setminus \{i\}$ would reach shadow label ℓ_i by following that path. Thus, reaching a vertex with a failure shadow label in which all n sub-labels are all cleared results in a 1-failure robust solution within G.

An rSG-PEG is constructed by repeated calls to its primary operation, G.ADDSAMPLE(w), which performs the following steps.

1) A vertex w is added to G.
2) An edge $w \to u$ is added between between w and each other vertex u of G if the line segment $\overline{wu}$ is fully contained in F^n. Similarly the twin edge $u \to w$ is added to G.
3) During this process, each time an edge $a \to b$ is created, for each failure shadow label L attached to a, the algorithm computes a failure shadow label L' for b, computed by propagating each shadow label ℓ_i in L across the edge $a \to b$, as described below. If L' is not dominated by any other failure shadow label at b, it is retained at b. Here, dominance of failure shadow labels is defined by generalizing the idea of dominance of shadow labels. This process continues recursively, spreading new reachable failure shadow labels across G as needed.

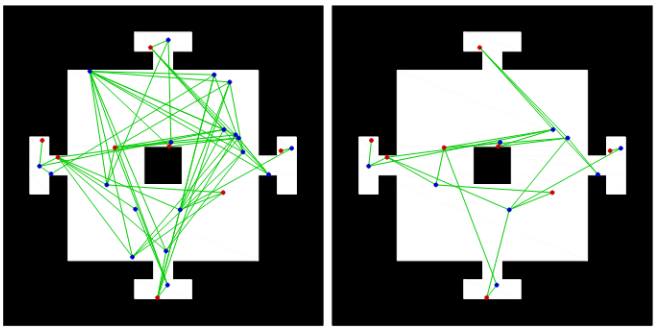
Adding a sequence of samples to an rSG-PEG is enough to form a 1-failure robust solution. However, propagating the shadow information along edges of the rSG-PEG is a computationally expensive operation that occurs frequently in the failure-robust scenarios considered here. The next section describes how to eliminate redundant geometric computations from this process.

B. Fast Label Propagation via Shadow Influence Caching

To accelerate the propagation of shadow labels across edges of the rSG-PEG, we construct shadow influence relations for each edge. For a given edge $w \to u$ and a given pursuer index i, let $S_{w,i}$ and $S_{u,i}$ denote the shadows at w and u formed in the absence of pursuer i. The *shadow influence relation* is a relation $R \subseteq S_{w,i} \times S_{u,i}$, in which $(s_w, s_u) \in R$ if and only if there exists a trajectory for the evader to travel undetected from s_w to s_u as the pursuers move from w to u. These shadow influence relations can be computed according to the update rules described in Section III-B.2 by discretizing the paths of the pursuers. This is a time-consuming computational geometry operation, but it must be performed only once for each edge-pursuer pair. Once the relation R is computed, any future shadow label can be propagated efficiently by setting each shadow s_u at u to be contaminated if and only if there exists a contaminated shadow s_w in the source shadow label for which $(s_w, s_u) \in R$.

C. Method of Sampling

Armed with an rSG-PEG suitably enhanced to generate robust solutions, it remains to devise a sampling strategy tailored to place JPCs in locations likely to lead to a solution quickly. Because of the computational expense of maintaining the list of non-dominated failure shadow labels at each vertex, it is of paramount importance that each sample we add to the rSG-PEG captures crucial and unique data. To do this, we rely on a method of scattering points throughout F called webs. This method was previously introduced by Olsen et al. [18], and is enhanced below. Using webs, we generate a sequence of samples with high connectivity between successive samples and strong coverage of the environment by multiple pursuers.

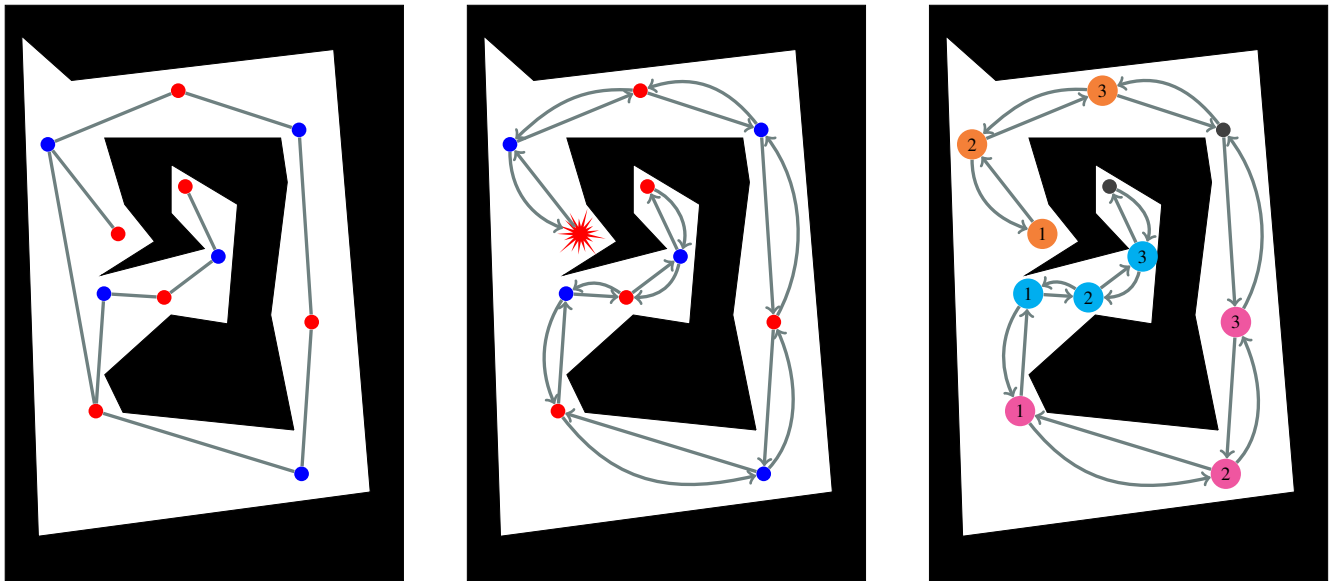


(a) Original web. (b) Sparse web.

Fig. 3: A visualization of the noise reduction in sparse webs. The red points represent the initial points while the intersection points are drawn in blue

1) Sparse Webs: A *sparse web*, $W = P \cup Q$, is the union of two sets of points from the environment. The *initial points* P are placed sequentially and uniformly at random outside of the visibility polygon of any previously placed initial points. This process continues until $\bigcup_{p \in P} V(p) = F$. Next, we construct the intersection points Q. For each unique pair of points $(p_i, p_j) \in P \times P$, we add to Q a uniformly random point from $V(p_i) \cap V(p_j)$ if $V(p_i) \cap V(p_j) \neq \emptyset$ and $Q \cap V(p_i) \cap V(p_j) = \emptyset$. The final condition, which is the alteration over the original form of webs, reduces the number of points in W while maintaining coverage and connectivity. See Figure 3. The initial points ensure that we have complete visibility coverage of the environment while the intersection points provide straight-line connectivity between the initial points, allowing a pursuer to move freely about an entire sparse web. Sparse webs generated in this way are used to generate sample JPCs, as described next.

2) Robust cycle samples (RCS): We wish to generate samples in a way that is effective in finding a 1-failure robust solution. To do this, we construct a graph H whose vertex set is a sparse web W and whose edge set consists of all unique pairs of points from $W \times W$ who can be joined by a line segment contained within the environment. The construction of webs ensures that H is, in essence, a visibility roadmap of F. Once H is constructed, a random vertex h is selected as the root node for a depth first search (DFS) on H. Throughout the DFS, we construct an ordered list D of vertices visited, both on their initial discovery and via backtracking. We continue this process until each point of H has been discovered and we successfully backtracked to the root node. We do not add the root node to D during the final stage of the DFS backtracking. This process produces a list of points $D = (D[0], \ldots, D[d-1])$ which contains every point from W at least once. The construction also guarantees connectivity between any adjacent points in D, giving a spanning walk, with repeats, around W. Since h is the first element added to D, $h = D[0]$.



(a) A sparse web.

(b) To form the walk D, DFS is performed on the sparse web starting at the star vertex. In this example, the length the walk D is $d = 20$.

(c) The first three samples generated by RCS. Notice that the 3 pursuers start $\lceil d/n \rceil = 7$ units apart from each other on D.

Fig. 4: The stages of RCS.

To generate n-robot JPCs, we evenly space the n pursuers along D and walk along the cycle generated by the DFS on H. That is, we create the first sample JPC by choosing $D[(i-1) \cdot d/n]$ for each robot $i = 1, \ldots, n$. Subsequent sample JPCs are generated in a similar fashion. See Figure 4. To generate the k^{th} subsequent sample, we select $D[((i-1) \cdot d/n + k \mod d)]$ for each robot $i = 1, \ldots, n$ and for each $k = 1, \ldots, 2d/n$. Having k range over $1, \ldots, 2d/n$ ensures that each point in D has been visited by at least two pursuers. Thus, should a failure occur for any single pursuer, that region of the environment has been seen by at least one other pursuer.

V. Evaluation

We implemented the algorithms described in Section IV in C++. All of the experiments below were conducted on a single core of a 6 core Intel i5-9600K CPU running 64-bit Ubuntu 20.04 at 3.7 GHz with 16 GB of RAM. Figure 5 shows an example of a 1-failure robust solution computed by this implementation, using both shadow influence caching and RCS.

To evaluate the algorithm quantitatively, we compared (a) the new shadow label updating system described in Section IV-B against the original naive approach, and (b) the RCS technique described in Section IV-C.2 to web sampling (WS).

For (a) we executed the main algorithm, using RCS, to find a 1-failure robust solution utilizing 3 pursuers in the environment shown in Figure 3. We conducted 25 trials using shadow influence caching and 25 trials using the naive method that computes shadow influence anew each time. All 50 trials successfully found 1-failure robust solutions. The average computation time (in seconds) utilizing shadow influence caching was 47.02 with a standard deviation of 15.12; without shadow influence caching, the average was 425.88 seconds, with a standard deviation of 512.64. The results demonstrate an overwhelming benefit to the use of shadow influence caching, which the authors, perhaps counter-intuitively, did not observe in the standard (i.e. 0-failure robust) setting.

For (b), we compared the efficiency of planning with RCS in contrast with planning with WS, a sampling strategy originally presented by Olsen et al. [18]. To do so, we attempted to find a 1-failure robust solution utilizing $n = 3$, 4, and 5 pursuers across the 3 environments found in Figures 2, 3 and 5. For each such scenario, we conducted 50 simulations. Tables I, II and III encapsulate the results, showing the mean (μ) and standard deviation (σ) of the planning time (in seconds) as well as number of vertices and edges in the rSG-PEG. Each such simulation was allotted a timeout of 10 minutes; if the corresponding simulation failed to produce a 1-failure robust solution in that time, it was deemed a failure. The results for 3 robots (Table I) show that RCS is, at worst, a marginal improvement over WS in all experiments. The addition of another robot further separated the results of these sampling methods. In particular, the standard deviation of the planning time was significantly decreased, resulting in more consistent run times. The highly connective nature of RCS allows us to increase the number of robots without severely suffering from the curse of dimensionality.

VI. Conclusion

This paper addressed the issue of potential single robot failures in the multi-robot visibility-based pursuit-evasion problem. We introduced a modification to an existing data structure to ensure that each solution generated by our algorithm remains a solution in the event of any single robotic failure. The inclusion of shadow caching proved to

Table I: Simulation results (n = 3).

	success rate	planning time (s)		Vertices		Edges	
		μ	σ	μ	σ	μ	σ
Figure 2							
RCS	100%	131.86	64.67	50.86	19.77	63.58	29.68
WS	98%	151.10	50.98	59.76	16.75	73.92	26.58
Figure 3							
RCS	100%	50.85	20.10	23.36	7.19	30.62	11.62
WS	100%	53.01	20.22	26.56	8.14	34.44	15.04
Figure 5							
RCS	100%	89.84	47.37	47.30	18.09	72.32	39.65
WS	100%	109.00	57.44	50.02	18.24	77.70	41.80

Table II: Simulation results (n = 4).

	success rate	planning time (s)		Vertices		Edges	
		μ	σ	μ	σ	μ	σ
Figure 2							
RCS	100%	140.57	60.01	64.72	64.57	74.34	87.02
WS	100%	194.53	74.96	87.14	90.13	104.10	133.88
Figure 3							
RCS	100%	58.21	21.82	23.26	24.01	28.34	40.63
WS	100%	75.40	30.11	26.56	9.21	28.54	10.95
Figure 5							
RCS	100%	82.79	44.73	38.34	26.62	46.10	38.11
WS	100%	113.08	57.75	46.34	25.11	52.60	33.74

Table III: Simulation results (n = 5).

	success rate	planning time (s)		Vertices		Edges	
		μ	σ	μ	σ	μ	σ
Figure 2							
RCS	100%	182.10	69.91	251.26	305.28	304.30	425.75
WS	98%	255.51	128.64	237.27	605.98	367.49	1233.18
Figure 3							
RCS	100%	84.93	46.79	88.48	238.54	258.76	1102.28
WS	100%	96.86	42.19	57.34	87.28	80.12	162.37
Figure 5							
RCS	100%	86.71	47.72	65.56	95.32	90.30	183.12
WS	100%	129.01	70.41	78.50	95.09	93.82	149.29

be cardinal to combat the rapid data expansion in the rSG-PEG. A new sampling method ensured that each region of the environment was observed by two or more pursuers, which allowed us to effectively spread out sampling points and reduce the computation time compared to previous sampling methods.

Future work may include the extension to k-failure robust problems, in the case where $k > 1$. In the interest of computation time —note that the most obvious extension of the present work would require shadow labels of size exponential in k— this general problem would likely require a new data structure which considers some sort of subset lattice structure where the elements represent sets of robots that failed.

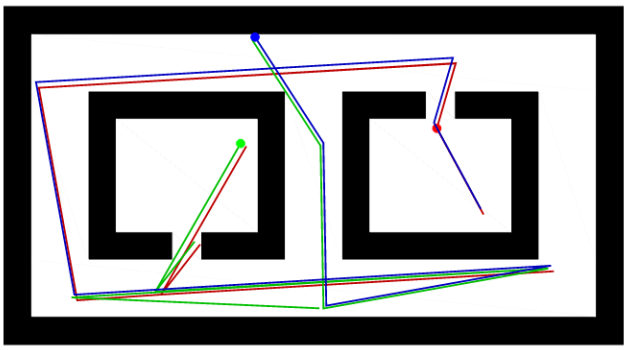


Fig. 5: A 1-failure robust solution. Notice each unique region of the environment is examined by at least 2 pursuers. The paths taken by the pursuers were slightly shifted for illustrative purposes.

References

[1] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking, "Randomized pursuit-evasion in graphs", in *Automata, Languages and Programming*, P. Widmayer, S. Eidenbenz, F. Triguero, R. Morales, R. Conejo, and M. Hennessy, Eds., 2002, pp. 901–912.

[2] R. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence", *Journal of International Humanitarian Action*, vol. 3, Dec. 2018.

[3] D. Crestani, K. Godary-Dejean, and L. Lapierre, "Enhancing fault tolerance of autonomous mobile robots", *Robotics and Autonomous Systems*, vol. 68, pp. 140–155, 2015.

[4] J. Dias, C. Paredes, I. Fonseca, H. Araujo, J. Batista, and A. de Almeida, "Simulating pursuit with machines. experiments with robots and artificial vision", in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, 1995, pp. 472–477.

[5] B. R. Donald, "A geometric approach to error detection and recovery for robot motion planning with uncertainty", *Artificial Intelligence*, vol. 37, no. 1, pp. 223–271, 1988.

[6] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view.", *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[7] P. González-De-Santos, A. Ribeiro, C. Fernández-Quintanilla, F. López-Granados, M. Brandstötter, S. Tomic, S. Pedrazzi, A. Peruzzi, G. Pajares, G. Kaplanis, M. Pérez-Ruiz, C. Valero, J. Cerro, M. Vieri, G. Rabatel, and B. Debilde, "Fleets of robots for environmentally-safe pest control in agriculture", *Precision Agriculture*, vol. 18, pp. 574–614, 2016.

[8] L. Gregorin, S. Givigi, E. Freire, E. Carvalho, and L. Molina, "Heuristics for the multi-robot worst-case pursuit-evasion problem", *IEEE Access*, vol. 5, pp. 17 552–17 566, Aug. 2017.

[9] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment", *International Journal on Computational Geometry and Applications*, vol. 9, no. 5, pp. 471–494, 1999.

[10] Y. C. Ho, A. Bryson, and S. Baron, "Differential games and optimal pursuit-evasion strategies", *IEEE Trans. Automatic Control*, vol. 10, pp. 385–389, 1965.

[11] R. Isaacs, *Differential Games*. New York: Wiley, 1965.

[12] V. Isler, D. Sun, and S. Sastry, "Roadmap based pursuit-evasion and collision avoidance", in *Proc. Robotics: Science and Systems*, 2005.

[13] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning", *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[14] Z. Kingston, M. Moll, and L. Kavraki, "Sampling-based methods for motion planning with constraints", *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, May 2018.

[15] A. Kolling and S. Carpin, "Multi-robot pursuit-evasion without maps", in *Proc. IEEE International Conference on Robotics and Automation*, 2010.

[16] F. Lehner, "Pursuit evasion on infinite graphs", *Theoretical Computer Science*, vol. 655, pp. 30–40, 2016.

[17] T. Olsen, N. M. Stiffler, and J. M. O'Kane, "Rapid recovery from robot failures in multi-robot visibility-based pursuit-evasion", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021.

[18] T. Olsen, A. M. Tumlin, N. M. Stiffler, and J. M. O'Kane, "A visibility roadmap sampling approach for a multi-robot visibility-based pursuit-evasion problem", in *Proc. IEEE International Conference on Robotics and Automation*, 2021.

[19] S. Park, J. Lee, and K. Chwa, "Visibility-based pursuit-evasion in a polygonal region by a searcher", in *Proc. International Colloquium on Automata, Languages and Programming*, 2001, pp. 281–290.

[20] T. D. Parsons, "Pursuit-evasion in a graph", in *Theory and Application of Graphs*, Y. Alavi and D. R. Lick, Eds., Berlin: Springer-Verlag, 1976, pp. 426–441.

[21] F. Shkurti, N. Kakodkar, and G. Dudek, "Model-based probabilistic pursuit via inverse reinforcement learning", in *Proc. IEEE International Conference on Robotics and Automation*, 2018, pp. 7804–7811.

[22] N. M. Stiffler and J. M. O'Kane, "A complete algorithm for visibility-based pursuit-evasion with multiple pursuers", in *Proc. IEEE International Conference on Robotics and Automation*, 2014.

[23] N. M. Stiffler and J. M. O'Kane, "A sampling based algorithm for multi-robot visibility-based pursuit-evasion", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[24] N. M. Stiffler and J. M. O'Kane, "Complete and optimal visibility-based pursuit-evasion", *International Journal of Robotics Research*, vol. 36, pp. 923–946, Jul. 2017.

[25] N. M. Stiffler and J. M. O'Kane, "Planning for robust visibility-based pursuit-evasion", in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[26] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region", *SIAM Journal on Computing*, vol. 21, no. 5, pp. 863–888, Oct. 1992.

[27] B. Tovar and S. M. LaValle, "Visibility-based pursuit-evasion with bounded speed", *International Journal of Robotics Research*, vol. 27, pp. 1350–1360, 12 2008.

[28] M. Visinsky, J. Cavallaro, and I. Walker, "Robotic fault detection and fault tolerance: A survey", *Reliability Engineering & System Safety*, vol. 46, no. 2, pp. 139–158, 1994.

[29] R. Zou and S. Bhattacharya, "On optimal pursuit trajectories for visibility-based target tracking game", *IEEE Trans. Robotics*, vol. 35(2), pp. 449–465, 2019.