Visibility-Based Escort Problem

Drew Beathard Lance Fletcher Priyankari Perali

***Abstract*—In this work we present a solution to the visibility-based escort problem – a problem closely related to the well-researched pursuit-evasion problem. This novel problem entails a single escort agent tasked with protecting and escorting one or multiple VIP agents from line-of-sight threats in a 2-dimensional environment. The algorithm takes as inputs a simply-connected polygonal environment, the starting location of the escort, and a goal location where the VIP agents should be safely moved to. The solution comes in the form of a path that is not a sequence of exact locations but rather a sequence of regions in which a VIP agent can safely exist. We search for a solution using Breadth First Search(BFS) across a graphical construction of the environment. The proposed method is capable of calculating non-trivial escort agent strategies from various starting configurations in diverse environments.**

***Index Terms*—path planning, pursuit-evasion, robotic escort**

I. Introduction

The visibility-based pursuit-evasion problem is a well-researched problem within the field of robotics that been studied in many different variations [1], [2], [3], [4], [9]. The motivation of the visibility-based pursuit-evasion problem is to find a path for one or multiple pursuers to take which guarantees the capture of one or multiple evaders.

Visibility-based pursuit-evasion solutions can be applied to many real-world scenarios which inspires ongoing research in the area. One variation involves the shortest route to be taken by a single guard agent to visually observe all corners in a region [13]. Another version is concerned with finding the minimum number of agents that can visually observe a region at once. [14] There are variations in the line-of-sight capabilities and world knowledge of the escort agents. Most of these problems share the common requirement of visiting all corners and edges in order to ensure the discovery of any rogue evaders.

We propose the *visibility-based escort problem*, a novel problem that is closely related to the visibility-based pursuit-evasion problem. In the context of the classic pursuit-evasion problem, the pursuer is re-titled as the *escort* and has the mission of safely escorting other vulnerable agents (VIPs) through an environment by ensuring they remain out of the line of sight of adversaries.

The escort agent establishes regions safe for VIPs to move within by clearing areas of adversaries. A VIP is considered safe if it is not within the line of sight of a region that is potentially occupied by an adversary. Many concepts fundamental to the visibility-based pursuit-evasion algorithms can be applied to the escort problem. In this paper, we refer to the visibility-based pursuit-evasion problem simply as the pursuit-evasion problem for brevity; however, in literature these names refer to different problems.

The escort problem can be applied to many different real world applications which require the safe escorting of one or multiple entities through an environment. Utilizing robots as escorts decreases the amount of people who need to be placed in high risk situations. A possible application and the original inspiration for this research is the escorting of politicians. Additionally, this approach can be extended to law enforcement or hostage retrieval scenarios where officers or hostages are modeled as VIPs. The application of our method allows for robot escorts to provide an additional layer of security to humans when attempting to navigate through or escape from a hostile environment.

This paper is organized as follows: a review of previous related works (Section II), a formal problem statement (Section III), the method used to solve the escort problem (Section IV), experimental results (Section V), and concluding remarks (Section VI).

II. Related Work

A. Escort Problems

To the best of our knowledge, the type of escort problem described in this work is novel and thus no previous solutions exist. Other escort problems have been studied but they do not have the same objective as the proposed problem. In [5], the escorting is done by surrounding an entity whose path is not known with multiple robots. The goal is to limit the escape windows of the entity by equally distributing the robots around it. More related to our problem, [6] accounts for unknown adversaries having a view of the VIP; however, they attempt to limit the adversaries' view of the VIP by surrounding it with multiple robots. Our approach utilizes a single escort robot to clear areas in an environment of adversaries to find a safe path for the VIP to take.

B. Pursuit-Evasion

The escort problem presented has many similarities with the visibility-based pursuit-evasion problem in which one or multiple agents work together to find dynamically moving adversaries. Given an environment, a solution to the pursuit-evasion problem will specify a path for the pursuer to take such that it is guaranteed that any adversaries will eventually be discovered. In the escort problem, the escort has a similar role to the pursuer, since both are capable of clearing areas within the environment. Provably complete and optimal solutions to the pursuit-evasion problem have been found [8], [11]. The pursuit-evasion problem has real-world applications in

security and search and rescue. In [7], the pursuit-evasion problem is first presented in the form of a graph traversal algorithm. A generalization of the single-agent algorithm for multiple pursuers is presented in [2]. Much of our approach is based on concepts from the work of Guibas et al., specifically the concept of an information space which allows for the environment to be discretized into conservative regions. A conservative region is an area within the environment which maintains the same shadow information as a function of the location of the pursuer [8]. A more general overview of this scheme is covered in [10]. Section IV elaborates more on the use of conservative regions in this work. A major difference between the pursuit-evasion problem and the escort problem is the latter does not require every area within the environment to be viewed. This is because solutions to the escort problem only require guaranteeing the safety of the VIP along a path at all times and not a search for an adversary.

III. Problem Statement

The escort problem seeks to generate a strategy for a single escort agent to safely escort one or multiple VIP agents through an polygonal environment while protecting them from line-of-sight threats.

A. Environment and Agents:

- ***The Environment:*** A two-dimensional simply-connected polygonal closed free space $F \subset \mathbb{R}^2$.
- ***Visibility Polygon:*** The set of all regions $V(q) \subset F$ visible from a single point $q \in F$.
- ***VIP Agents:*** Considered the "important" agents that need to be protected at all times and escorted to the goal location. The solution to this problem finds a safe path for the VIP agents to take. A safe path ensures VIPs remain out of the line of sight of regions which potentially contain adversaries. Therefore, our solution does not need to explicitly consider the number of VIP agents or their specific locations.
- ***Adversary Agents:*** Adversaries pose a line-of-sight threat to VIP agents. Similar to VIPs, we do not explicitly consider the number of or location of adversaries. Rather, we employ a worst-case approach to ensure a correct solution is found by assuming that all regions which *could* contain an adversary do. This assumption is common in pursuit-evasion problems as evaders are capable of moving at an unbounded speed.
- ***Escort Agent:*** The escort agent is modeled as a single point that can move freely through the environment. The escort has omnidirectional vision. It is assumed that adversaries are unable to neutralize escorts, but escorts are able to neutralize adversaries. Let $e(t) \in F$ represent the escort agent's position at time t. All regions within the visibility polygon $V(e(t))$ are considered clear of adversaries.

B. Shadows

Shadows: All regions $S = F \setminus V(e(t))$ that fall outside of the visibility polygon of the escort agent. Shadows are classified as either *cleared* or *contaminated*.

- ***Cleared Shadows:*** The set of shadows where it is not possible for an adversarial agent to exist. Cleared shadows arise as a result of previous movements by the escort. Whenever a shadow appears in a 2D space as a result of escort movements it will remain cleared until coming into contact with a contaminated shadow. Figures 1 and 2 show an example of this.
- ***Contaminated Shadows:*** The set of shadows where it is possible for an adversary to exist. Since we model the speed of adversaries as unbounded, we assume that every point inside a contaminated shadow could contain an adversary. The VIPs must never be visible from any point within a contaminated shadow.

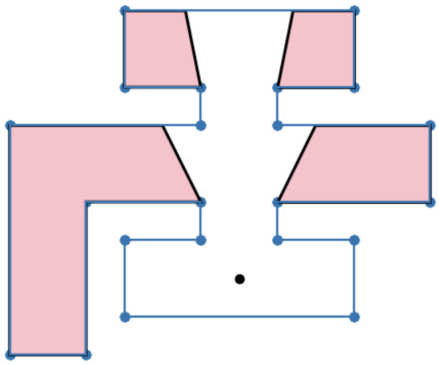

Fig. 1: Initial state of shadows for given escort, where all shadows are assumed to be contaminated. Contaminated shadows are represented in pink. Escort represented as black point.

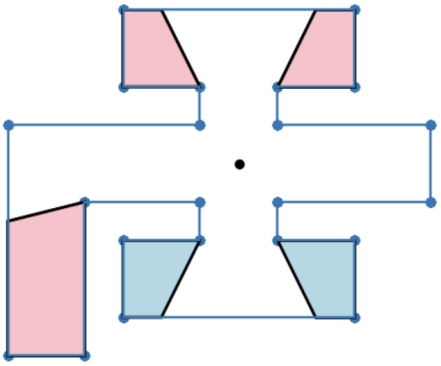

Fig. 2: Next state of shadows. Contaminated Shadows are represented in pink. Cleared Shadows are represented in light blue. Escort represented as black point.

The status of each shadow is not guaranteed to be constant. This is due to the occurrence of shadow events. These shadow events have four types: merge, disappear, appear, and split [8].

The dynamic nature of shadows also extends to their status as being either cleared or contaminated. When handling transitions between states, the contaminated shadows are transferred via the overlapping rule. The overlapping rule states that if a shadow of the current state overlap with any contaminated shadows of the previous state, that shadow of the current state will be contaminated.

An example of how the overlapping rule affects the state is shown in Fig. 1 and Fig. 2. Fig 1 represents the initial state. For any initial state within this problem, we assume all the shadows to be contaminated. Fig. 2 represent the next state. As shown, the escort moves upward, resulting in a different set of shadows. However, in this case we only consider the shadows that overlapped any of the previous shadows to be contaminated. As a result, the bottom two shadows (blue) are clear. This is demonstration of the the common rule that any newly appearing shadows start in a cleared state.

C. Safe-zones

- ***Safe-Zones:*** The set of all regions that fall outside of the accumulated visibility polygon of all contaminated shadows. Simply put, these are the regions where a VIP agent can safely exist. Safe zones are an important element of our state representation. We only consider those which are *VIP-contaminated*.
- ***VIP-contaminated Safe-Zones:*** The subset of safe-zones that are reachable by a VIP considering the previous actions of an escort agent up to a point in time. Figure 3 provides an example of a configuration in which there exists both VIP-contaminated safe-zones and VIP-unreachable safe-zones.

It is important to note that the overlapping rule as used for shadows extends to safe-zone state transitions. Safe-zones that are not VIP-contaminated are those that do not satisfy the overlapping rule i.e. if there is no point in a current safe zone that overlaps with with a previous state's VIP-contaminated safe zone, the current safe zone is considered VIP-unreachable. An example of this is shown in Fig. 3. In Fig. 3, there are two safe-zones. However, the right most safe-zone (light purple) is not considered to be VIP-contaminated. This is because it does satisfy the overlapping rule. The safe-zone has no overlap with the previous state's VIP-contaminated safe-zone (light green).

D. Conservative Regions

Conservative Regions: Regions within the environment where every position within a region maintains the same shadow information. The boundaries of conservative regions specify where shadow events occur. If an escort crosses the boundary of a conservative region, it is known that a significant change to the shadows in the environment will occur. These regions are calculated with a method of ray shooting between vertices within the environment. An in-depth explanation of this process can be found in [8] and [11].

IV. Method

The current method consists of three main components. First, a graph of positions is constructed from the conservative region discretization of the environment. Second, a transition black box function is used to calculate a new state given a previous state and a new escort location. Finally, a breadth first search algorithm is applied to the information graph to find a path for the escort to take to create a safe path for the VIPs to reach the goal location.

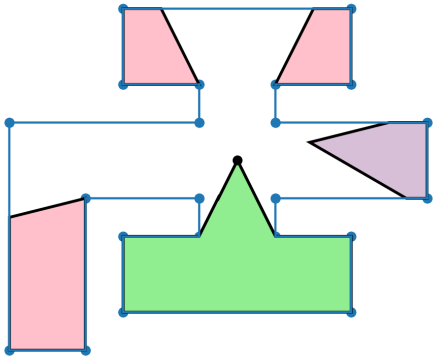

Fig. 3: The problem state. Contaminated shadows represented in pink. VIP-contaminated safe zones represented in light green. All other safe zones represented in light purple. Escort represented as black point.

A. Problem State Representation

In the escort problem, a *state* contains all the information needed to determine the status of the environment. A state contains the escort's position, a list of all contaminated shadows, and a list of all VIP-contaminated safe-zones. From this information, it can be determined what areas of the environment are VIP-safe. An important characteristic of a state is it is not solely based on the position of the escort, but also the path taken by the escort to be at a position. It is possible for the escort to be in a position, take a path that eventually returns back to the same position, and the state can be different. This is because the state's list of contaminated shadows and VIP-contaminated safe-zones are dependent on the previous actions of the escort. For example, in figure 2, it can be seen that the escort is located at the center of the environment with two cleared shadows present at the bottom. These shadows are cleared because the escort had previously cleared the area which now contains the two cleared shadows. If the escort had started at the center of the environment those shadows would be contaminated.

B. Position and Information Graphs

After defining a state for the escort problem, it is now known that if the escort takes a calculated path throughout the environment a solution will be produced if one exists. But finding a path for the escort to take is non-trivial. Similar to methods used in the pursuit-evasion problems [8], the escort problem involves two graphs: the position graph G_P, and the information graph G_I.

G_P is a graph where each node represents a unique *position* that the escort can traverse to and from. Since the environment

is continuous, a method is needed to determine the exact positions that compose G_P. A simple but inefficient method would be to discretize the environment into a grid where each cell's centroid is a node in G_P. In this work a more efficient method based on conservative regions is employed. Since conservative regions distinguish where important shadow events occur and since safe-zones are calculated from the shadow information, these regions also identify important events related to the escort problem. As shown in figure 4, once the environment has been discretized into conservative regions, the centroid of each region is used as a node in G_P. Adjacent conservative regions correlate to adjacent nodes in G_P. Not enough information is contained within G_P to search for a path for the escort to take, but it does provide information about the relationship between specific positions in the environment. Ideally, the environment would be discretized into regions which distinguish critical safe-zone events, i.e. safe-zones appearing and disappearing; however, currently it is not known how to obtain this kind of discretization.

In G_I, each node corresponds to a unique *state*. The breadth first search is conducted on this graph since it is capable of determining when a safe-path for the VIP is found. Performing a search on a graph consisting of problem states allows for more complex paths to be taken by the escort. This gives the escort the ability to be at a position, take a path which clears some portion of the environment, and then return back to the original position. Therefore, this graph representation provides a way to search for a series of escort movements that are likely to result in a solution.

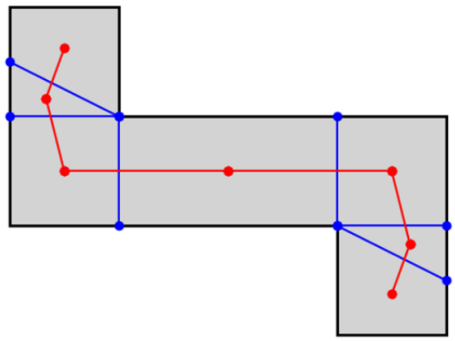

Fig. 4: The graph constructed from conservative regions. The blue lines represent boundaries of the conservative regions. The red dots represent the possible positions of states within the information graph. The red lines specify which positions are reachable from other positions.

C. Transition Black Box

The transition black box is a function that determines a new problem state, given a current problem state and a new escort position.

The function takes the current problem state and escort position as inputs. From this, it determines the shadows and safe-zones associated within the given escort position. These are computed based off the descriptions in Section III. Then, the contaminated shadows are determined by checking for any overlap between the new shadows and the problem state's contaminated shadows. If there is any overlap, meaning that at least one point is shared between a new shadow and a problem state's contaminated shadow, then the new shadow is considered to be contaminated. This is repeated for every shadow associated with the new escort position. The same methodology is used to determine new problem state's VIP-contaminated safe-zones.

The function's objective is to maintain the information between problem states as well as acknowledge any changes within the information that may have occured.

Algorithm 1 TransitionBlackBox

Input: problemState ps, Position $newEscortPos$
Output: problemState ps'

$vipContaminSafe \leftarrow [\,]$
$contaminShadows \leftarrow [\,]$
$shadows \leftarrow computeShadows(newEscortPos)$
for each $s \in shadows$ **do**
  for each $cs \in ps.contaminShadows$ **do**
    if $overlap(s, cs)$ **then**
      $conatminShadows.append(s)$
    end if
  end for
end for
$safezones \leftarrow computeSafeZones(shadows)$
for each $sz \in safezones$ **do**
  for each $v \in ps.vipContaminSafe$ **do**
    if $overlap(sz, v)$ **then**
      $vipContaminSafe.append(sz)$
    end if
  end for
end for
$ps'.escortPosition \leftarrow escortPosition$
$ps'.contaminShadows \leftarrow contaminShadows$
$ps'.vipContaminSafe \leftarrow vipContaminSafe$
return ps'

D. Breadth First Search

The search phase is executed through the implementation of Breadth First Search (BFS) across problem states beginning with the initial problem state. The algorithm terminates when there exists a path to a problem state with a VIP-contaminated safe-zone that contains the VIP's goal position.

The algorithm avoids cycles through the implementation of a hashing function built into the problem state class. If the hash value has been seen before, the search determines that the exact state in terms of escort position and shadow states has been seen before should not be added to the queue.

The search cannot directly jump from one node to another node along the conservative region graph. Instead, each leg of the search needs to be broken into a group of sequences sufficiently frequent that there are no missed safe-zone critical events. An example of such an occurrence would be the

Algorithm 2 SafePathBFS

Input: problemState ps, Position g
Output: Path P

$P \leftarrow [\,]$
$visited \leftarrow$ Dictionary(problemState, Boolean)
$Q \leftarrow$ empty queue
$endState \leftarrow$ null, $pathFound \leftarrow$ False
Q.enqueue(ps)
while Q is not empty and $\neg pathFound$ **do**
    $curr \leftarrow Q$.dequeue()
    $visisted[curr] \leftarrow$ True
    if $g \in curr.vipContaminSafe$ **then**
        $endState \leftarrow curr$, $pathFound \leftarrow$ True
    end if
    for each neighbor n of $curr$ **do**
        $ns \leftarrow$ TransitionBlackBox($curr, n$)
        if $ns \notin visited$ **then**
            $visited[ns] \leftarrow$ True
            Q.enqueue(ns)
        end if
    end for
end while
if $pathFound$ **then**
    $s \leftarrow endState$
    while $s.parent$ **do**
        P.append($s.position$)
        $s \leftarrow s.parent$
    end while
end if
return P

disappearance of a VIP-contaminated safe zone and reappearance shortly after. This safe-zone would reappear as VIP-unreachable because of the short segment of time where the VIP would have been exposed. It is likely for such an occurrence to go undetected if the steps between each leg of the search are too sparse.

V. Results

We implemented and tested the described method on a variety of escort scenarios to analyze its ability to find safe paths for VIPs through path planning of the escort agent. The Python computation geometry library scikit-geometry [12] was used for geometric visualization.

As seen in figure 5 our method is capable of finding solutions in a variety of scenarios involving environments which vary in complexity and size. Each sub-figure within figure 5 shows the final state of the environment from the found solution. Each solution shows the VIP-contaminated safe-zone (light-green) contains the goal point (green dot), thus providing a safe path for the VIPs to take. Additionally, it can be seen that the paths taken by the escort are non-trivial and often require the clearing of many areas within the environment before a safe path can be found. An notable observation of our solutions is that not every contaminated shadow (pink) was cleared in the final solution (except environment c). This distinguishes the solutions found by our method from the solutions to a typical pursuit-evasion problem. When given a large environment with a relatively small distance between the initial VIP location and the goal point, our method will clear the least amount of contaminated shadows needed; however, a pursuit-evasion based path for the escort would take longer to compute since every shadow must be cleared for a solution to be found.

In all the escort scenarios tested, during the escort's execution of the solution path, the VIP-contaminated safe-zones never move as a whole, but only expand. We attempted to find escort scenarios where a solution would cause the VIP-contaminated safe-zone to be shifted throughout the environment and not expanded, but were unable to do so. It is suspected that holed environments have solutions which would cause the safe-zones to relocate across the environment rather than expand.

Table I shows the computation time needed for the various escort scenarios shown in 5. As expected, paths which require the escort to clear more contaminated areas take longer to find. A majority of the required computation time is due to the amount of geometric calculations performed in the transition black box function. Despite the escort problem requiring the paths for both the VIP and the escort to be found, our representation of the problem state simplifies the problem. Rather than searching a state space which accounts for the exact position of both the escort and the VIP, our method need only take into account the position of the escort. Searching a state space which includes the positions of two agents would be computationally more expensive.

Figure 5 Environment	Time (seconds)
a	18.57
b	29.46
c	124.99
d	211.33
e	668.54
f	222.9

TABLE I: Computation time needed to find solution for each escort scenario shown in Figure 5.

VI. Conclusion

In this paper we introduce the novel visibility-based escort problem and provide an algorithm capable of solving it. The algorithm utilizes an environment discretizion method, transition function, and a state search algorithm to determine a non-trivial path for the escort, resulting in a safe path for the VIP agents to follow. The method presented only requires the path of the escort to be determined and not the path of the VIP, thus decreasing the size of the state space needing to be searched. With our approach we demonstrate that the escort does not need to clear the entire environment of adversaries in order to find a safe path for the VIPs. This proves the

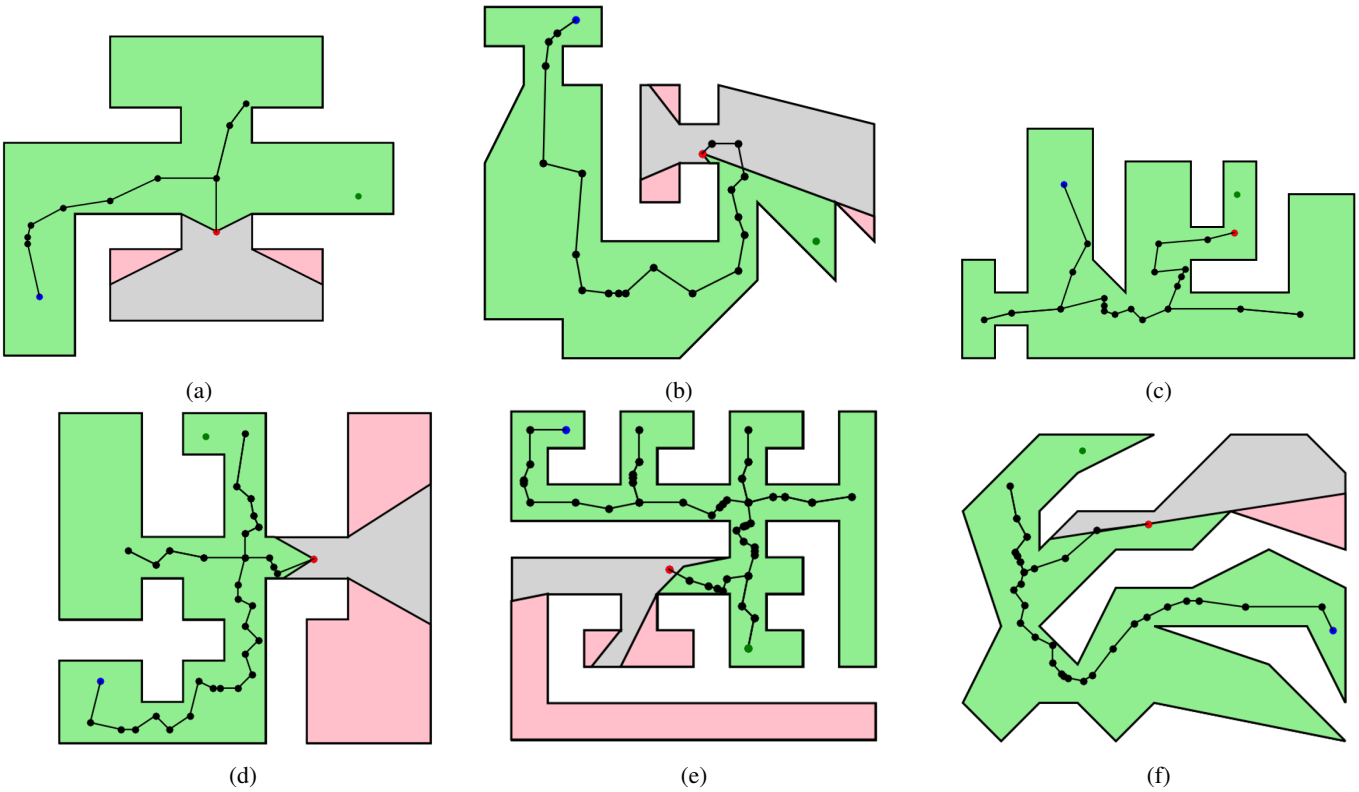

Fig. 5: Final state of each environment based on the found solution. In each sub-figure the blue dot represents the starting position of the escort. The red dot represents the last position of the escort. The black lines depict the path taken by the escort. The green dot represents the goal position for the VIPs. The light green regions represent VIP-contaminated safe-zones at the end of the escort's path. Similarly, the light pink regions represent contaminated shadows at the end of the escort's path.

value of utilizing our method rather than employing the more exhaustive pursuit-evasion approach.

Many questions regarding the escort problem remain to be addressed in future research. A particularly important direction for further research should focus on discretizing the environment into safe-zone conservative regions. Similar to how a conservative region in the pursuit-evasion problem maintains the same shadow information, some regions within the environment could also maintain the same safe-zone information. Initial investigations in finding these types of conservative regions have proven it to be challenging problem.

Another scenario which deserves further research is that containing multiple escort agents. This scenario would allow safe paths for VIPs to be discovered for more complex environments; however, with multiple agents, the state space searched increases significantly and could result in computational complexity issues found in other related multi-agent problems [2]. Lastly, although we did not conduct any testing in holed polygonal environments, we believe our method will extend to such environments without issue.

Acknowledgment

We would like to thank Jason O'Kane for his guidance from start to finish with the project. Additionally, we would like to thank Nicholas Stiffler for providing code which helped in the implementation of our method.

References

[1] Brian P. Gerkey, Sebastian Thrun, and Geoff Gordon. Visibility-based pursuit-evasion with limited field of view. in Proc. of the National Conference on Artificial Intelligence. San Jose, California, July 2004.

[2] N. M. Stiffler and J. M. O'Kane, "A complete algorithm for visibility-based pursuit-evasion with multiple pursuers," 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 1660-1667, doi: 10.1109/ICRA.2014.6907074.

[3] J. W. Durham, A. Franchi, and F. Bullo. Distributed pursuit-evasion without mapping or global localization via local frontiers. Autonomous Robots, 32(1):81–95, 2012.

[4] Tovar B, LaValle SM. Visibility-based Pursuit—Evasion with Bounded Speed. The International Journal of Robotics Research. 2008;27(11-12):1350-1360. doi:10.1177/0278364908097580

[5] Antonelli, Gianluca & Arrichiello, Filippo & Chiaverini, Stefano. (2008). The Entrapment/Escorting Mission. Robotics & Automation Magazine, IEEE. 15. 22 - 29. 10.1109/M-RA.2007.914932.

[6] Bhatia, Taranjeet & Solmaz, G & Turgut, Damla & Bölöni, Ladislau. (2015). Two algorithms for the movements of robotic bodyguard teams.

[7] D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and D. R. Lick, editors, Theory and Application of Graphs, pages 426–441. SpringerVerlag, Berlin, 197

[8] 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, 9(5):471–494, 199

[9] Search and pursuit-evasion in mobile robotics A survey Timothy H. Chung · Geoffrey A. Hollinger · Volkan Isler

[10] J. Yu and S. M. LaValle, "Shadow Information Spaces: Combinatorial Filters for Tracking Targets," in IEEE Transactions on Robotics, vol. 28, no. 2, pp. 440-456, April 2012, doi: 10.1109/TRO.2011.2174494.
[11] Stiffler NM, O'Kane JM. Complete and optimal visibility-based pursuit-evasion. The International Journal of Robotics Research. 2017;36(8):923-946. doi:10.1177/0278364917711535
[12] Wolf Vollprecht, https://github.com/scikit-geometry/scikit-geometry
[13] W Chin and S Ntafos. 1986. Optimum watchman routes. In Proceedings of the second annual symposium on Computational geometry (SCG '86). Association for Computing Machinery, New York, NY, USA, 24–33. https://doi.org/10.1145/10515.10518
[14] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. 2021. The Art Gallery Problem is -complete. J. ACM 69, 1, Article 4 (February 2022), 70 pages. https://doi.org/10.1145/3486220