Catching an Evader in a Polygonal Region with a Door

Jianghong Wan
Boston University College of Engineering
110 Cummington Mall, Boston, MA
rickywan@bu.edu

Shuwen Zheng
Boston University College of Engineering
110 Cummington Mall, Boston, MA
zhengsw@bu.edu

Abstract

We study the problem of two pursuers with limited view(2-searcher) searching for an evader in a simple-connected room P with a door d. The purpose of this game is to plan the motion of the pursuers so that the evader e will be caught within finite time without the evader fleeing through the door d. A 2-searcher is a searcher whose visibility lies solely on two rays emanating from the searcher's position, where the visibility rays can rotate at an angular velocity. The pursuers and the evader's movements are turn-based, and pursuers have no information regarding the position of the evader, whereas the evader knows the positions of the pursuers. We present a more capable algorithm that the pursuers can search more room structures and catch the evader. The simulation of the algorithm is evaluated by repeatedly running and testing to see if the evader is caught.

1. Introduction

Pursuit-evasion games are one of the more fundamental problems of robot motion planning, where one or more pursuers search for an evader in a region. Many variations of the pursuit-evasion have been studied recently, and the winning criteria falls into two big categories: detection or capture. For this project, we study the visibility-based pursuit-evasion game in a simple-connected polygonal region with a door, with two limited-view pursuers, and the pursuers need to catch the evader.

Majority of the studies focus on detection-aimed pursuit evasion games [1,3,5,9]. Suzuki and Yamashita [9] were the first ones to introduce the polygon search problem, and since then there have been various studies in this category. J.H. Lee, et al. [1,5] introduced an algorithm to plan the search path for a 2-searcher in a polygonal room with a door. The algorithm we used to search for the evaders when it's not been seen is very similar to what was discussed in this paper. However, in [1,5], the disadvantage of having only one pursuer was shown. There are restricted structures of rooms that are searchable for a 2-searcher, whereas in our case, having an additional pursuer enables searching for more structures.

Another category of pursuit-evasion is capture-oriented planning [2,7,8]. V. Isler, et al. [2,7] presented a strategy for two pursuers to effectively catch an evader in a simple-connected room by using the randomized algorithm and the solution to "lion and man problem". However, the two pursuers both have unlimited view so they can scan the polygon area easily. Also, a solution for two pursuers to capture one evader with limited visibility was provided [7], although it may take $O(n^5)$ time to successfully catch the evader. Klein and Suri presented [8] a strategy for 3 pursuers with omni-directional vision to catch one evader in any polygonal rooms is provided, including the non-simple connected rooms. It was proved that three pursuers were sufficient, and necessary for the worst case. However, the pursuers are required to have information about the position of the evader. In this project, we will show that 2 pursuers with limited view are able to catch the evader in a simple-connected polygonal room.

1.1. Definitions and Notations

A room (*P, d*) is defined as a simple polygonal region *P* and a door *d* on the boundary. The pursuers and the evader are modeled as a point within the region *P*. For a 2-searcher, its visibility lies solely on two rays emanating from its position, as shown in Fig.1. The two endpoints of its visibility ray are defined as *x* and *y*, *s* is one of the pursuers. See the figure below: s lies on some point on the line segment $\overline{xy}$. The chain $\overline{xsy}$ is called a variable-length two-segment chain *V* [1].

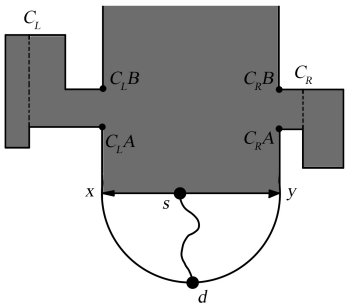

Figure 1: 2-searcher illustration

For convenience, we define the region that is visible from d to be the hall, and the cuts C_RB-C_RA, C_LA-C_LB to be the entrances of the essential cut.

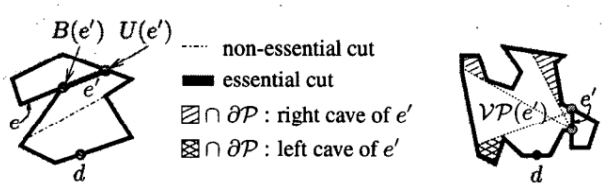

Figure 2: Definition of cut, essential cut, and cave [1]

As shown in Fig.2, an edge *e* of *P* belongs to a line *l*. An extended edge *e'* $\supset(l \cap P)$ is a line segment that shares an endpoint with *e*, and the other endpoint lies on the boundary of *P*, denoted ∂P. An extended Edge *e*' is called a *cut* if any path from *d* to *e* has to go across *e*'. A cut *e*' dominates another cut *c*' if any path from the door *d* to *e'* has to go across *c*'. A cut *c*' is said to be an *essential cut* if it is not dominated by any other cuts. We define the entrance [C_A, C_B] of an essential cut *c* to be the closest cut to *c* that is dominated by it. A cave *e* of *c*' is the part of the boundary that is not visible from *e'*. If it lies on the right of the visibility boundary, it is called a *right cave*, and symmetrically, it is called a *left cave* if it lies on the left of the visibility boundary.

2. Strategy

The game is turn-based and catch-oriented, which means that the two pursuers and the evader take turns to make movements, with the pursuers moving first, which move as fast as the evader. The evader is considered as caught if it is within a circle of radius r centered at each pursuer's location. The pursuers can communicate with each other at all time.

The initial conditions of the game are as follows. At the beginning of the game, the evader will appear randomly within the room. Two pursuers will be at the door.

The two pursuers in our game consists of a searcher s and a guard g. When the evader is not seen, the searcher will follow the algorithm in [1], and the guard will stand at a certain point to prevent recontamination cycle and evader fleeing through the door. Preliminarily, the evader will follow a greedy algorithm where it will try to keep a maximum distance to the pursuers while going towards the door. When the evader is seen, the pursuers follow the "lion strategy".

2.1 Locating the Evader

After the game starts, since pursuers do not know the location of the evader, they should first locate the evader and then catch the evader. The searcher follows the algorithm in [1] to search for the evader, and It searches all the cuts in ascending order in terms of *d(d, ci)*. In this way, with the help from the guard, we can ensure that the clean areas will not be re-contaminated. The movements of the searcher consist of six sub-movements. (See Fig 3) Now suppose the searcher wants to proceed to *z*. If the chain *C[x, z]* is totally visible to *y*, then it proceeds the *l-advance-by-sweep* by moving its left endpoint from *x* to *z* while *y* keeps fixed. If *C[x, z]* is not visible from *y* but there are no cuts between them, it proceeds *the l-advance-from-lid*, where the left endpoint moves along the chain *C[x,z]* while *y* is fixed. Another movement, *l-advance-by-rotate* is applied when there is a cut in between *C[x,z],* so *V* rotates as shown in the figure below, from $\overline{xy}$ to $\overline{ab}$. Symmetrically, there are three movements for the right endpoint y called *r-advance-by-sweep*, *r-advance-from-lid*, and *r-advance-by-rotate* [1].

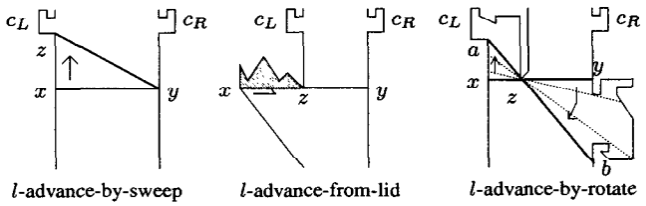

Figure 3: sub-movements of a 2-searcher[1]

It has been shown in [1] that a 2-searcher with those 6 sub-movements is sufficient to search any 2-searchable rooms. However, the conditions for a room to be 2-searchable is strict. The room shown in Fig. 4 is not searchable by a 2-searcher. When searching C_1, the evader can get to the door from C_2.

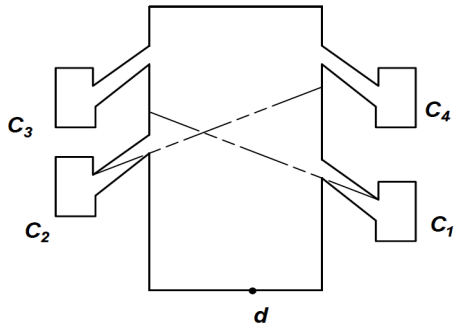

Figure 4: A room that is not 2-searchable

However, with one additional 2-searcher, the room is searchable. As Shown in Fig. 5 When searcher s is searching C_1, its left endpoint x keeps fixed, and it searches cut C_1 by moving *y* along the boundary of C_1. While the searcher is searching C_1, the guard is keeping its endpoints [*x',y'*] to position [C_2A, C_1A]. The position of guard *g* is at the argmin(d(d, $\overline{C2AC1A}$)) on the line C_2A-C_1A, as shown in the Fig. 5, is the point where the distance to the door is minimum, so that when the evader

is detected by $\overline{x'y'}$, *d(d, g) <= d(d, e).* With the guard, the evader would not be able to flee to the door without being seen, and if seen, the guard will always reach the door position no later than the evader. The region below $\overline{x'y'}$ is guaranteed to be clear, so by *g* continuously moving forward, the algorithm keeps "squeezing" the

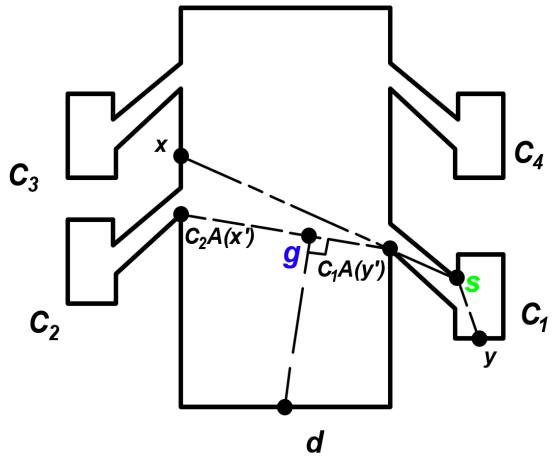

Figure 5: Searching strategy

contaminated area until the evader is found.

2.2 Catching the Evader

Once the e collides with *V* of either of the pursuers, it becomes visible to them. There are three conditions of being visible. (shown in the Fig. 6) 1.) the evader is seen inside a cut, in this case, C_1, by $\overline{sy}$, where the searcher s directly proceeds the lion strategy, and x',y' will be fixed at the entrance of the cut, [C_1A, C_1B], and then g moves to *g'* at the center of the entrance, which guarantees capture;

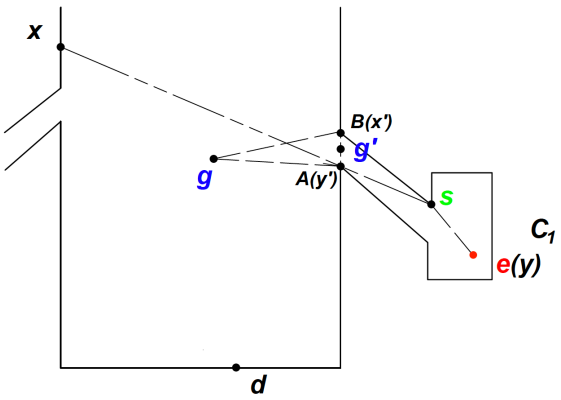

Figure 6: Finding the evader inside a cut

2.) if the evader is seen by $\overline{sx}$, then both *x* and *x'* focus on *e*, meanwhile *s* stop searching and start to chase *e*. Since the guard is always closer to the door than the searcher, if *e* precedes to *d*, it has to go across C_2A-C_1A first. So, when the evader reaches the line C_2A-C_1A, g starts to move directly to *d* to prevent escaping. (See Fig.7) 3.) If the evader is first seen by $\overline{x'y'}$, *g* moves directly to d, while fixing *x', y'* at *e*. Meanwhile, *s* starts to chase for *e*.

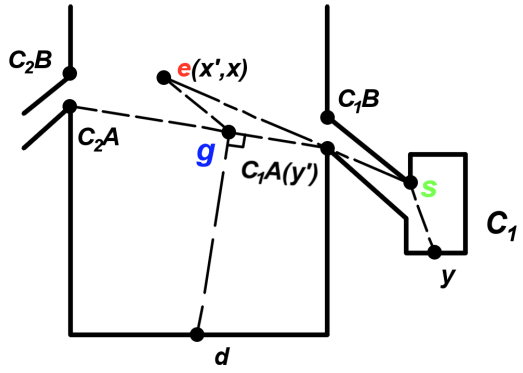

Figure 7: Searching strategy

In the case of losing visibility of the evader after seeing it, shown in Fig. 8. After the evader is detected, it moves into C_2, which is not visible from neither *g* nor *s*. To prevent losing track of the evader, the guard will keep tracks of the entrances that *e* has gone across. After losing sight, it will fix *x',y'* to the entrance C_iA, C_iB, in this case, C_2A,C_2B, and move to the middle point of line C_2A,C_2B, at the same time *s* enters the region C(C_2A, C_2B) to search for the evader. After detecting the evader, the lion strategy will be applied.[2]

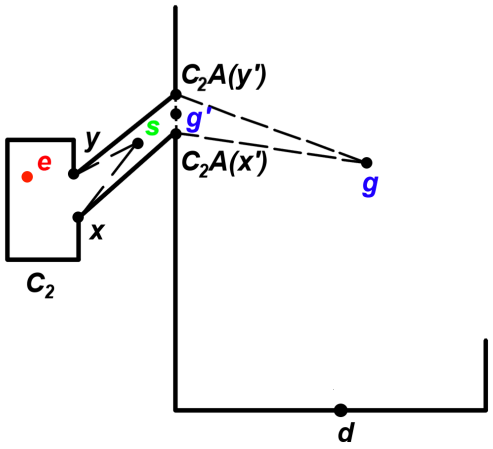

Figure 8: Re-searching the evader

2.3 Strategy for the Evader

The motion plan for the evader is simple. The evader will randomly appear at either cut, and it moves with respect to a completely random manner, with restriction that it can only move to one of its neighbors. If seen, a

heuristic value is calculated, and the evader would move to the neighbor with the least heuristic value, which is the path that moves away from the pursuers.

3. Simulation

Our simulation runs on MATLAB. The game is turn-based, and every turn both pursuers and the evader can only move to one of their neighbors. The moving order is searcher, guard, evader, respectively. Each turn after the searcher and guard move to their designated location, they move their endpoints to the target configuration. The searchers and the guard move to their goal position via A*, and the evader is moving in a completely random manner, with random initial position.

3.1. Expected Output and Evaluation Method

The expected output of the simulation is that the evader would always get caught in finite time in a room which is not searchable by algorithm in [1].

Since the movement of the evader is completely random, we evaluate our simulation by repeatedly run the game, and see if the evader escapes. For the evaluation, we run the simulation 100 times.

3.2. Pseudocode

Data: graph room, searcher start node s_{start}, guard start node g_{start}, evader start position e_{start}
for room numbers
    s_{path} = A*(next essential cut);
    g_{path} = A*(next guard position);
end
while $e(i) \notin Area(s_{view}(i), s_{view}(i-1))$ & $e(i) \notin Area(g_{view}(i), g_{view}(i-1))$
    $g(i+1) = s_{path}\ (i+1)$;
    Set view sight $s_{view}(i)$ based on searching algorithm;
    $g(i+1) = g_{path}\ (i+1)$;
    Set view sight $g_{view}(i)$ based on searching algorithm;
    attrective(i+1) = d(*e(i)*, *door*);
    replusive (i+1) = d(*e(i), s(i+1)*) + d(*e(i), g(i+1)*);
    e(i+1) = neighbor nodes of *e(i)* with the max (*attrective(i+1) - replusive (i+1)*);
end
while d(*e(i), s(i)*) > radius or d(*e(i), g(i)*) > radius
  if line(*e(i), s(i)*) is visible
    s(i+1) = lion(*s(i), e(i)*);
    g(i+1) = entrance of the cut where evader was found;
    h(i+1) = d(*e(i), s(i+1)*) + d(*e(i), g(i+1)*);
    e(i+1) = neighbor nodes of *e(i)* with the max *h(i+1)*
  else
    s_{path} = A*(last visibile *e*);
    s(i+1) = s_{path} (2);
    s(i+1) = entrance of the cut where evader was found;
    h(i+1) = d(*e(i), s(i+1)*) + d(*e(i), g(i+1)*);
    e(i+1) = neighbor nodes of *e(i)* with max *h(i)*
  end
end

3.3. Scheduled Searching

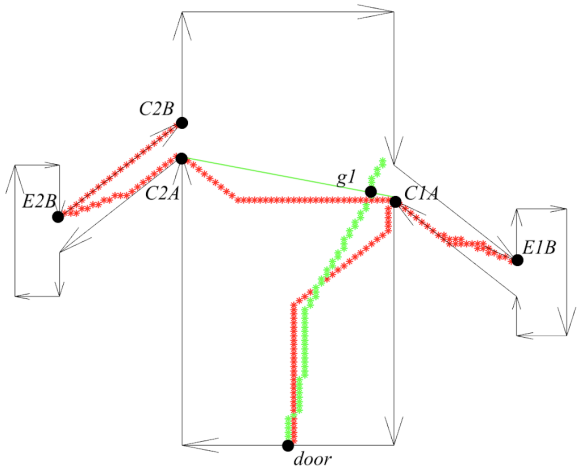

Figure 9: Searching Paths

Fig. 9 shows the path of the searcher and the guard. The searcher follows the red line, and the green dotted line represents the path of the guard. The searcher path is as follows: d→ C_1A → E_1A → C_1A → C_2A → E_2B → C_2B, and the path for the guard is from d to g1, and then proceeds to the area behind g_1. The paths are generated by A*.

In the region below C_1A, both guard and searcher keep the configuration to be [0; π], which forms a horizontal line. After the searcher reaches C_1A, it fixes its right endpoint at C_1A, while rotating its left point to E_1A, and then move to E_1A while the endpoints fixed. After it reaches E_1B, its left endpoint rotates for 360 degrees, completely clearing the cut. Then it goes back to C_1A, with its endpoints fixed at E_1B and C_1A. While moving towards C_2, its two endpoints rotate and fix to [C_2A, C_2B]. As it moves towards E_1B, symmetrically, it fixes its left endpoint at C_2A, and right endpoint at E_2B. Upon reaching E_2B, the right visibility ray rotates for 360 degrees to clear the C.

Finally, As Figure 10 shows, the searcher moves to the hall with its endpoints unmoved, and then rotate its left visibility ray for 360 degrees to clear the rest of the region. Whenever the evader is seen, the searcher would break its searching schedule and proceeds the lion strategy.

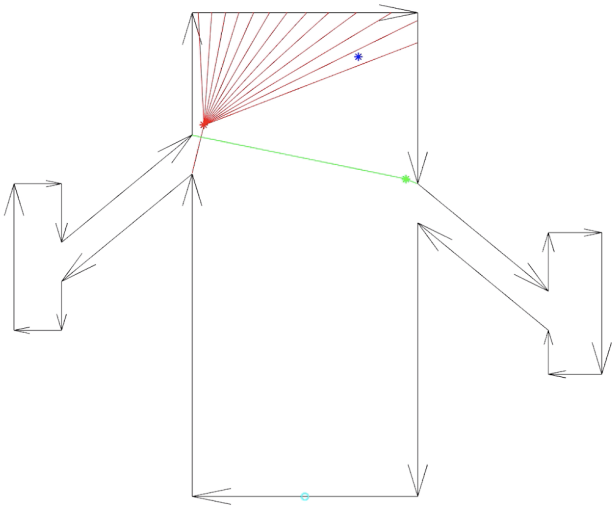

Figure 10: Scanning the hall

When the searcher is searching an essential cut, shown in Fig.11, E1 in this case, where it doesn't have the visibility to d, the guard would move its endpoints to the boundary of the cleared region, making sure that the region below it is cleared. In this case, the evader has been seen by the searcher, and then the searcher proceeds the lion strategy.

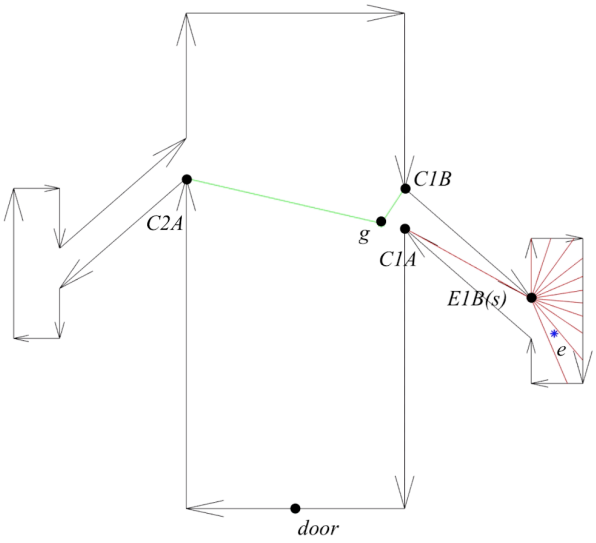

Figure 11: Searching in a cut

After finishing searching all the essential cuts, if the evader has not found, the searcher proceeds to the hall, rotates its right visibility ray for 360 degrees to scan the hall. If the evader is not found in either of the cuts, it is guaranteed to be in the region of the hall that is above the guard's visibility chain. By scanning the hall, the evader is guaranteed to be found.

3.4. Catching The Evader

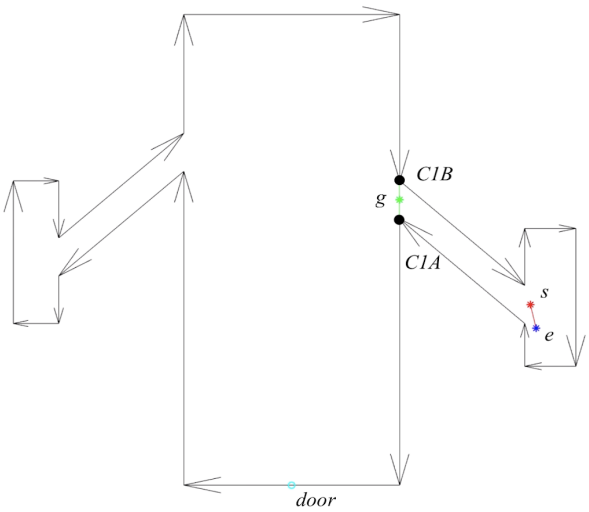

Figure 12: Found the evader

When the evader has been seen, the searcher locks its endpoints at the evader, and proceeds lion strategy. The guard moves its endpoints to the entrances of the cut and moves to the middle of the entrance to prevent evader escaping from the searcher. (See Fig. 12)

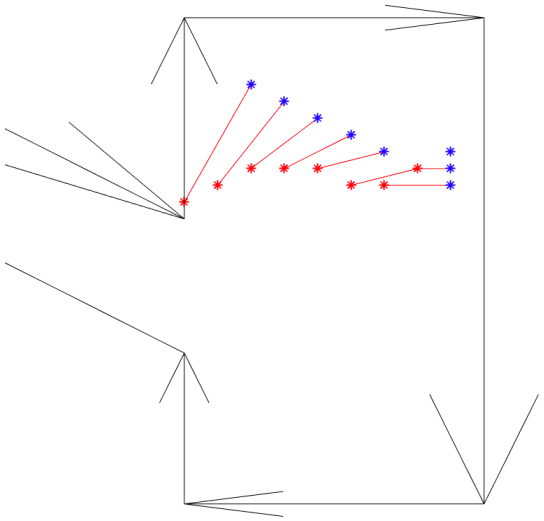

Figure 13: Lion Strategy

When the evader has been seen, the lion strategy is applied, as shown in Figure 13. The center C of the lion circle is the position of the searcher when it firstly sees the evader. At turn t, the searcher moves from the position of time *(t-1), s(t-1)* to *s(t),* which lies on the line C-evader.

3.5. Lose Sight When Catching

In case of losing visibility to the evader while capturing it, the path of the evader is stored. Once the visibility ray between the evade and the searcher is blocked by any vertices, which means that the searcher has lost its sight to the evader, the searcher will proceed to the last known position of the evader, and continue to rotate its visibility ray for 360 degrees to continue scan for the evader. Since the guard is blocking the entrance of the cut, the evader is guaranteed to be within the region beyond the entrance, so the evader is guaranteed to be found.

4. Conclusion and Future Work

We propose a new algorithm for searching in a polygonal room with two 2-searchers, which has more capability of [1], and present a simulation of it on MATLAB. It combines algorithms in [1] and [2], to search and capture an evader in an environment, which is not searchable by a single 2-searcher. Based on the premise that the pursuers have complete information about the map, they can search the room throughout the paths generated by A* algorithm, and along with adjustments to the visibility endpoints, and corporations between the guard and the searcher, they can "squeeze" the contaminated area while making sure the evader is not able to reach the door. Once the evader has been found, the searcher can effectively approach and capture the evader using lion strategy. With over 100 simulations with random initial positions and paths of the evader, we conclude that our algorithm can search for the room that [1] is not capable of.

4.1. Future Work

An interesting future investigation on this algorithm is to determine if it can be implemented on non-simple connected region. In reality, most of the rooms are simple connected, however, it is very likely that there are obstacles in the room.

It is also interesting to add some elements in the algorithm, for example, if there is a potential barrier, how to adjust the algorithm to successfully search the room? Additionally, what if the pursuers only know partially about the map, and need to learn the map while searching it? Those elements would complexity of the algorithm yet make it more interesting and realistic.

Another interesting development is to improve the algorithm for the evader, since due to lack of time, the evader's escaping strategy is not sophisticated designed. If possible, the strategy of the evader could be planned as A* with moving obstacles, where its goal position is the door, and the obstacles to be the visibility rays and the circle centered at the pursuers' position.

5. Link for Simulation

The MATLAB files used for simulation can be downloaded from the link https://github.com/shawnz9/me570finalproject/tree/master

References

[1] J.H. Lee, S.Y. Shin, and K.Y. Chwa. Visibility-based pursuit evasion in a polygonal room with a door.

[2] Isler, Volkan & Kannan, Sampath & Khanna, Sanjeev. (2004). Locating and Capturing an Evader in a Polygonal Environment. Technical Reports (CIS). 17. 10.1007/10991541_18.

[3] Tan, Xuehou. (2008). A unified and efficient solution to the room search problem. Computational Geometry. 40. 45-60. 10.1016/j.comgeo.2007.04.001.

[4] R. Murrieta-Cid, R. Monroy, S. Hutchinson and J. Laumond, "A Complexity result for the pursuit-evasion game of maintaining visibility of a moving evader," *2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, 2008, pp. 2657-2664.

[5] Lee, Jae-Ha & Park, Sang-Min & Chwa, Kyung-Yong. (2002). Simple algorithms for searching a polygon with flashlights. Information Processing Letters. 81. 265-270. 10.1016/S0020-0190(01)00235-6.

[6] J.H. Lee, S.Y. Shin, and K.Y. Chwa. On the Polygonal Search Conjecture.

[7] Isler, V., Kannan, S., & Khanna, S. (2004). Randomized Pursuit-Evasion with Limited Visibility. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Vol. 15, pp. 1053-1062)

[8] Klein, K., & Suri, S. (2011). Capturing an Evader in Polygonal Environments: A Complete Information Game. *ArXiv, abs/1110.4838*.

[9] Suzuki, I., & Yamashita, M. (1992). Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, *21*(5), 863-888.