ANALYZING SIMPLE HEURISTIC HIDING STRATEGIES FOR NON-PLAYER CHARACTERS IN GAMES

by

Navjot Singh

School of Computer Science
McGill University, Montréal

June, 2016

A THESIS SUBMITTED TO MCGILL UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF
MASTER OF SCIENCE

Copyright © 2016 Navjot Singh

Abstract

The need for a player to hide from opponents is an important part of many strategy and stealth games. Effective approaches to hiding for non-player character's (NPCs), however, are relatively rare, with most games either simply attributing player stealth to NPC companions, irrespective of their actual behaviour, or making use of highly scripted behaviours.

In this thesis we examine algorithmic approaches to hiding in a game context. We explore several computationally simple strategies for ensuring an NPC can remain hidden with respect to a moving observer on a defined path. We evaluate these strategies with respect to an upper-bound on how successful an NPC can be in hiding. This allows us to determine relative quality of hiding heuristics, and can also be used to answer several interesting questions, such as how many NPCs can be hidden, the effective area hidden using a strategy, and for finding the good hiding spots in the game level. Our evaluation is performed using multiple game levels, including simple theoretical geometries used to illustrate the inherent difficulty of hiding, as well as levels inspired from commercial games of various genres. We also consider additional important factors, such as different speeds of NPCs and variation in observer paths.

Our work suggests that simple cost-effective strategies are feasible for use in game levels to deliver a more natural NPC hiding behavior. Both complex and simpler heuristics for hiding perform similarly well, although the effectiveness of a strategy strongly depends on the structure of the game level, speed of the NPCs and the predefined path chosen for analysis.

Résumé

Le besoin d'un joueur de se cacher des ennemis est une part importante de plusieurs jeux de stratégie et de jeux d'infiltration. Les approches efficaces de dissimulation pour les Personnages Non-Joueurs (PNJ) sont relativement rares, la majorité des jeux n'accordant la capacité de dissumulation qu'aux PNJ ayant un rôle de compagnon, sans regard à leur comportement réel, ou font l'usage de comportements fortement scénarisés.

Dans cette thèse nous examinons des approches algorithmiques quant à la furtivité dans le contexte du jeu. Nous explorons de nombreuses stratégies informatiquement simples qui assurent qu'un PNJ puisse se dissimuler d'un observateur en mouvement sur un chemin défini. Nous évaluons ces stratégies en relation avec le résultat optimal du potentiel succès d'un PNJ à se cacher. Cela nous permet de déterminer la qualité relative d'heuristiques ayant trait à la capacité de se dissimuler, et peut aussi être utilisé pour répondre à plusieurs questions intéressantes, telles que combien de PNJ peuvent se cacher, la zone d'efficacité d'une cachette selon une stratégie et pour trouver les bonnes cachettes dans un niveau de jeu. Nous performons notre évaluation en utilisant de nombreux niveaux de jeu, en plus d'utiliser des niveaux inspirés de jeux commerciaux de genres divers. Nous considérons des facteurs additionnels, tels que les différentes vitesses des PNJ et la variation des chemins d'observateur.

Notre travail suggère que des stratégies rentables économes peuvent être utilisées dans des niveaux de jeu pour accomplir une furtivité plus naturelle dans le comportement des PNJ. Les heuristiques complexes tout comme les plus simples performent similairement bien, malgré que l'efficacité d'une stratégie dépende fortement de la structure du niveau du jeu, de la vitesse des PNJ et du chemin prédéterminé pour l'analyse.

Acknowledgments

First and foremost I would like to thank my supervisor, Prof. Clark Verbrugge for his continuous guidance, invaluable help and feedback, and patience throughout this thesis. I am extremely grateful to my family for their moral and financial support. I am thankful to my close friends for their encouragement and advice every step of the way.

Contents

List of Figures

List of Algorithms

Chapter 1
Introduction

Hiding is used in many games to avoid detection by enemies and cameras. In games like *Dishonored*, *Deus Ex* series, *Hitman* series and many role-playing and action adventure games, hiding is an effective technique to stealthily accomplish the game goals. These games use many elements for the player to hide and move under cover such as structure of the level, lighting, limited field of view, and visual and audio distractions. Another type of hiding system is the cover system used in first person shooters (FPS) such as *Call of Duty* series. These are designed for taking cover during combat, although not for absolute covert movements.

In all these games the levels are designed in terms of the player being the prime user for these hidden regions. The ally non-player characters (NPCs), if any, usually follow the player and hide where the player hides or in nearby defined regions. Most of the time their movements and behaviour are confined or scripted. There are some reactive enemy NPCs which take cover when engaged, but the cover areas are clearly defined in the level designs and there is minimal path planning to reach these regions. The NPC actions and movements are usually kept simple. The reason for this is that the AI would otherwise need to compensate for stealth in player movements and position, further increasing the game complexity with each additional NPC, which already includes the calculations for the strategy, path planning, and combat. This might result in the game becoming much more demanding for the system and the player and less fun. Moreover the player might see uncertain NPC movements due to constant heuristic calculations which will feel unnatural.

Hiding is also interesting in role playing games like *The Elder Scrolls V: Skyrim* and *Baldur's Gate* where it is used for stealth assaults, stealing and providing safety. Players have to interact with other players and NPCs in a massive open world. The NPCs have to be designed such as to behave as a normal player. This includes following a player and mimicking his/her behavior (if the player moves stealthily, the NPCs has to follow suit), able to steal and get away undetected, stalking, stealth backstabbing or just hiding and waiting for other events to happen. Stealth is not necessarily a central goal for these games, but one of the means to accomplish goals. While playing these games, one can observe inconsistencies during hiding for NPCs. The movements and actions do not feel completely refined and natural. Better solutions could be provided but that would increase complexity, and game developers are less inclined to put resources into fixing flaws which are not game breaking for already released games. Third party mods to improve stealth of NPCs in such games however have been developed. There are a few mods for stealth of the followers in *The Elder Scrolls V: Skyrim* which try to add to the capabilities of the NPC followers when following and executing orders to kill [17, 10]. These work well in the bounds of the game rules and improve on the normal game but the NPC movements still feel superficial and do not make the behaviour of the followers feel more natural to a player.

We are interested in exploring computationally inexpensive strategies for NPCs to have natural reactions to another agent's movements in a game level, aiming to avoid being discovered. The problem we are considering is a complex one, considering the processing needed for stealth NPC movements at each step for NPC hiding and simultaneously moving towards their goal positions. We will try to formulate it by using a simple model by focusing on the hiding heuristics. One agent moves along a fixed exploratory path while the other agent hides in the game level behind obstacles. We would refer to the agent who moves inside the game level on a fixed path as the *observer* and the agent who tries to hide as the *survivor*. The path that the observer moves on is discretized into path points for *point visibility* calculations.

We will now take a glimpse of what our game levels will look like and what are some common features in each of our experiments. We can see a basic game level setup in figure 1.1 which is an orthographic view of the level from above. The observer is shown

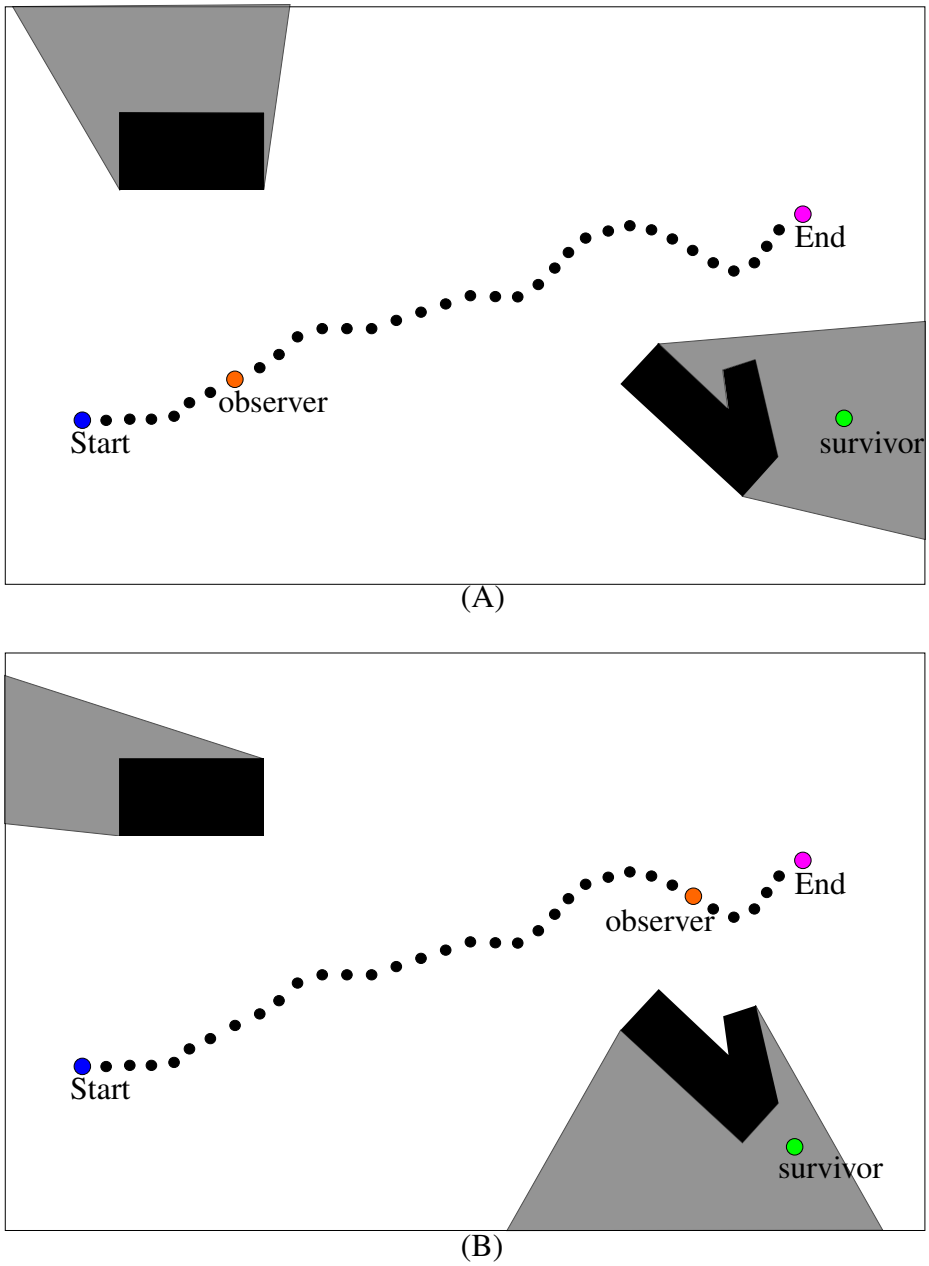


Figure 1.1 This shows a basic orthographic view from above of a basic setup for our experiments. The observer (orange circle) moves on a deterministic path (black dots) having 360° field of view with no limit for the distance of vision; the survivor's (green circle) aim is to moves in the dark gray regions which are not visible to the observer from its position. Here we see the survivor's positions in the shadow/hidden regions behind the obstacles, for the two observer positions.

as an orange circle. It has 360° field of view and can see infinitely far away. The black dots are the deterministic path taken by the observer from the start position, represented by the blue circle, to end position, represented by purple circle. The black polygons are the obstacles in the game level and the grey regions are the non-visible regions from the current viewpoint of the observer (we show two sub-figures *A* and *B* for two different positions of the observer). The survivor, shown as green circle, can hide in these grey regions (shadows). The speed of the observer and survivor are bounded in each experiment.

We will discuss three heuristic strategies and a strategy which represents the best performance possible for the heuristic strategies for NPC survival, which we shall call the *optimal* strategy. The first strategy for hiding is called *simple greedy*. The survivor will only move when it is absolutely necessary (i.e. when it is about to be discovered). This strategy requires minimal processing. The second strategy is an extension to the *simple greedy* strategy. The survivor will move when it is imperative, just like the *simple greedy* approach, but it will move as much into the nearest shadow region (region not visible to the observer) as possible constrained only by its speed. This provides a better chance of survival. The third strategy tries to use the shadow regions to its advantage. The survivor constantly tries to move in the local central region of the shadow region, to be safer from detection by the observer. We do not predict observer movements in any strategy because in game genres such as action fps, role playing or stealth games, the character movements are not predictable and they often frequent the same areas/paths, and we are looking for simple generic methods which might work for game levels of many game genres.

We will observe the results on different levels for each strategy and compare them to each other in effectiveness. Besides comparing the heuristic strategies to each other, we developed a fourth strategy, a perfect solution for NPC survival considering the same conditions and assumptions, to have a more comprehensive understanding of the success of each strategy. This strategy tries to find a perfect solution for the survivor for any starting position in the shadow areas in the game level. It depends on discretizing the game level completely, to create a 3D survivor movement graph structure. We only use this approach for analysis as it is expensive, and assumes that the survivors know the path of the observer beforehand, unlike other heuristic strategies.

There are a few assumptions for our experiments that we would like to state. For a com-

plete game level analysis, we use multiple survivor agents simultaneously. All our hiding approaches assume there to be no collisions between these survivor characters. Discretization of the 2D game level is performed for ease of complexity in calculations, but it is only used for survivor movements in the *optimal* approach. In other strategies, it is used for initial placement of survivors for each experiment.

We use the finely discretized game level environment to test another heuristic strategy, with collisions taken into account between the survivors. Each discretized point in the game level, which is in the shadow areas, is occupied by a survivor at the beginning and they have to move to safer areas as the observer moves on it's path, while avoiding collisions in terms of occupying the same space as any other survivor. We shall discuss all the above approaches in the chapters to come.

Specific contributions addressed in my work include,

- the introduction of three heuristic hiding strategies for non-player characters. These might assist in designing a more natural reactive behavioral mechanics for NPCs during game-play.
- formulation of an optimal solution. This helps to determine the peak performance which the heuristic strategies aim to achieve. This method also helps to visualize the complete discretized 2D level at various path points of the observer path movements.
- analysis of the complete game level for hiding. We are able to characterize different safety grades for regions in the game level.
- consideration of important quantitative metrics such as relative speeds, multiple observer paths and the number of NPCs in the game levels.
- evaluation and comparison of each strategy through experimentation using game levels from various genres such as stealth, role-playing, first-person shooters which demonstrate that our work could be useful in numerous cases.
- visualization and integration with Unity game engine provides a modular framework for convenient analysis of any 2D/3D game level.

This thesis contains five chapters. The chapters are organized in following fashion.

- Chapter 2 contains background and related work in the areas of visibility, and stealth and hiding strategies in robotics and games.
- Chapter 3 explains some basic concepts which were used for calculations involved in our work and introduces the game levels used, the strategies, comparisons between strategies and other quantitative metrics for each game level.
- Chapter 4 provides the results and discussions from experimentation performed on the game levels and comparisons between the strategies to show the effectiveness of these strategies.
- Chapter 5 offers a conclusion based on our findings and some future work discussion.

Chapter 2
Background and Related Work

In this chapter we will discuss some application areas, concepts and other research works which are related to our work. Computing visibility is a base concept for our work and is used in numerous other fields directly or indirectly in applications in computer games, graphics, robotics, motion planning, designing etc. Visibility is a vast topic with multiple sub-concepts and each have been well studied and numerous optimized methods have been published for each, such as for calculating *point visibility* of a polygon (with and without holes/obstacles), calculating *edge visibility* of a polygon, constructing *visibility polygons*, *triangulation* of polygons, *visibility kernel* of polygons, etc.

In robotics, search and navigation of an area for military, security and emergency services applications is widely useful. The robots have sensors to "see" and detect obstacles, and plan a path to its goal. Common scenarios include exploring the area to search for other agents, planning stealth movements and hiding from observers. In games, stealth techniques are usually used by the players to achieve their goals but some concepts are used by the NPCs such as following the player stealthily, taking cover in fps games, planning an ambush for the player.

2.1 Visibility

Calculating visibility is a problem with many established solutions for various types of visibility [11]. For our work, *point visibility* is used significantly. We will learn about *point*

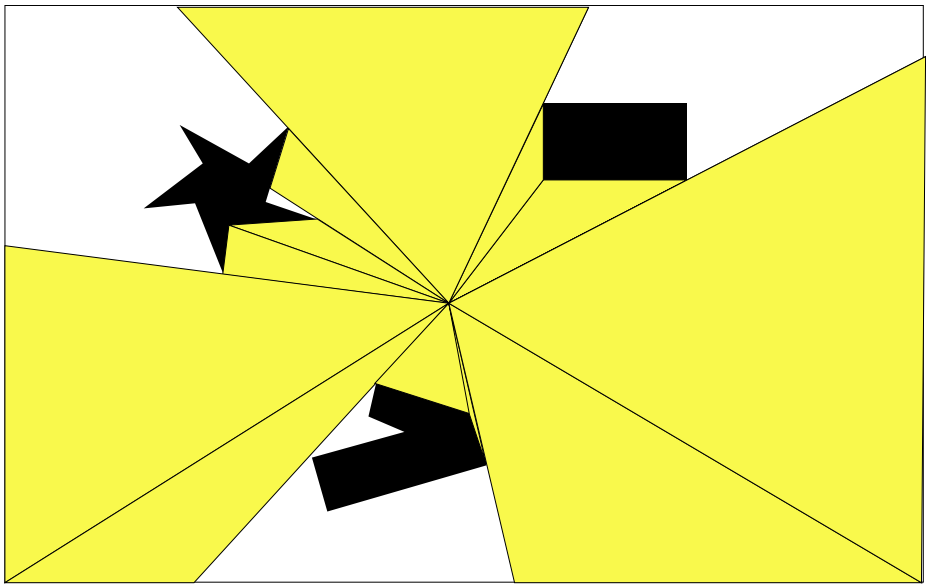


Figure 2.1 Point Visibility for a point in the center of a bounded region. The area (star shaped polygon) in yellow is the visibility polygon/region.

visibility through *visibility polygons*. The figure 2.1 represents a view from above of a game level. The region in yellow is the area which is visible for a character standing in the center of the game level and it is called *visibility polygon*. It is a star shaped polygon made up of triangles, shown as the lines emanating from the center of the level, which share a vertex which is the point for which *visibility polygon* is being calculated. This concept is called *point visibility* since we calculate a *visibility polygon* for a point. Another visibility type is the *edge visibility* which is defined as the complete region/polygon visible from a point or set of points on a specified edge within the polygon. There is *complete*, *strong* and *weak* edge visibility depending on the number of points on the edge from which each point in the visibility polygon must be visible. Although we are interested in what is visible from a path, we do not compute *edge visibility* directly because it is computationally expensive; instead, we calculate *point visibility* for a discretized path, using individual points on each line segment of a polygonal path in our region.

An optimal linear algorithm to calculate *visibility polygon* was given in 1981 [8], but due to its complexity it was hard to implement it. A simplified version was published in 1983 [19] but a minor error was discovered in that approach which was later fixed in

1987 [16]. An *angular sweep* optimal algorithm was published in 1985 [1] for visibility for a point in a polygon with *holes* and an efficient implementation was given in 2014 [3]. Another optimal algorithm for calculating visibility for polygon with *holes* came out in 1995 whose complexity depended on the number of *holes* [14].

A well-known problem in computational geometry is the *art gallery problem* which is a direct application of visibility in an area. It emerged from a real-world problem of using minimal number of guards to observe a complete art gallery. This has been well studied and several solutions were provided [25, 7, 9]. A common variant to the *art gallery problem* is the *watchmen route problem* where the objective is to compute the shortest route a watchman should take to guard an entire area [5, 4]. There are direct applications of *visibility polygons* in robotics to sense and "see" the environment and detect obstacles [13].

2.2 Motion planning in robotics

A prevalent topic of research in robotics is searching and/or exploring an area for various purposes. There are many military and security applications which solely rely on a robot sensing its environment, planning and navigating through an area stealthily, such as search and rescue, reconnaissance, scouting, surveillance, safe autonomous transport of payloads or people in observable areas.

Interesting work has been done for hiding in known environments in the presence of moving *sentries* who are on a lookout, by planning a covert path to an appropriate hiding position [22]. Path planning is required in hide and seek scenarios where an agent hides before the game starts while the seeker looks for it [21]. In general applications, designing a successful algorithm for the seeker is much more challenging compared to the hider which is why more research is focused on seeking element [24]. Our work is more directed towards the hider. The hiding agent (survivor) is already hidden at the beginning but it needs to adjust its position depending on the current position of the seeker (observer).

There have been a few research works for planning stealth movements to reduce the robot's visibility from the observers. An algorithm for navigating a three dimensional terrain containing multiple moving observers was given in 1993 [28]. Another solution

to a similar problem of stealth navigation on a similar environment was presented in 1994 using a harmonic function [26]. A fast algorithm for computing highly occluded paths on a terrain using graphics processing unit (GPU) was proposed in 2013 [18]. Distinctive solutions to planning stealth motions on a discretized occupancy grid map considering static observers in an unknown environment were also published [2, 30, 23]. A reactive robot navigation approach was given in 2004 [29] which involves mobile obstacles which results in a dynamic environment.

Pursuit evasion problem is a common scenario studied in visibility problems, whose most basic example is the *cops and robbers game*. The pursuer tries to find the evader by searching the complete environment and the evader in turn tries actively to avoid detection. Most research is focused on the movements of the pursuer in different environment scenarios such as inside of a *simply-connected polygonal environment* [31]. The effects of limiting the field of view of a searcher is also studied and compared to a 360° field of view [27]. Using multiple pursuers and planning their motion in a polygonal environment to detect an unpredictable evader was also explored in 1997 [12]. A single searcher may not be sufficient for some game levels to find the evader because of the risk of *re-contamination* i.e. the evader could hide in the area which was already explored and cleared by the searcher. However in 2005 it was proven that a single pursuer can locate an evader in any *simply-connected polygon* using a randomized strategy [15]. An informative survey about pursuit evasion problems was done in 2011 [6] which covers many search problems and discusses some open problems in these areas.

2.3 Stealth in games

NPCs movements in games are generally simple such as following the player, aiding the player in the game from different positions, or hiding and ambushing the player. There have been some work in developing tools for level design to study stealth paths taken by NPCs using Rapidly exploring Random Tree(RRT) for pathfinding through a discretized game level [32]. *Waypoint graphs* are crucial in planning NPC movements which can be thought of as a high level roadmap containing points representing physical positions of significance in the actual game level. This makes it easier to breakdown the game level

into a graph consisting of positions which can be connected to each other and a path can be formed between two *waypoints* efficiently using pathfinding algorithms such as $A*$. The game levels are studied through these graphs for pathfinding, identifying *ambush and pinch points* [33], developing strategies for NPCs for games involving combat in first person shooter or action adventure games by pre-calculating and storing tactical information about the relationship between *waypoints* [20]. The movements of survivors for our work are not calculated through *waypoint graphs* but could be used along with pathfinding algorithms such as $A*$ on *waypoint graphs* to get finer and more natural NPC movements.

Hiding is common in many games genres but it is generally focused towards the player hiding and enemy NPCs trying to find the player. Games in which NPCs hide and move stealthily are few and usually their behaviour is predetermined based on player movements. Games like *Skyrim* implement NPC follower hiding, but the followers mimic the player movements and its path taken and hence are not capable of independent stealth movements. We were unable to find any specific instances in games with NPCs hiding and moving independently.

In this thesis work we will focus on 2D environments to simplify visibility calculations and focus more on NPC hiding mechanisms. Most 3D game environments project faithfully onto a 2D surface, and the extra complexity of computing 3D visibility is not necessarily required.

The obstacles in our 2D environments are polygonal to ease the calculation of visibility, although our work on NPC hiding strategies can be safely extended to environments containing curved obstacles where the visibility can be calculated with more general visibility algorithms.

We are interested in analyzing the hiding strategies in real time and focus less on base visibility algorithms, although calculating visibility is crucial for our work. We chose a simpler more naive algorithm to calculate visibility which is not one of the optimal solutions regarding time and space complexity for calculation and storing *visibility polygons*.

The above research works in visibility and robotics have several assumptions about the environment knowledge, obstacle and boundary geometry, and pursuer/observer movements. We do not assume any prior knowledge of the environment or the observer move-

ments. In our experiments, the agents are not aware of each other or the environment. The survivor agents are vulnerable to making wrong decisions and getting stuck depending on the heuristic strategy. We perform path planning for survivors for each step of the observer path in a small localized area around each survivor. We do not use path finding algorithms to find a safe path for the survivors, as they do not have prior knowledge of observer path or direction of movement. Our *optimal* strategy can however be used for calculating the safe paths for the survivors at any position for a known observer path. It can be extended for other objectives such as identifying ambush positions, hiding treasures, and a useful visualization tool for designing game levels.

Chapter 3
Methodology

In this chapter we will try to explain the basic setup, computing visibility polygons for the game levels, the hiding strategies, discretization of the game level and the two character categories. We will also discuss the reasons and our thought process behind the algorithms, quantitative metrics and discretization process.

The game context we developed tries to simulate several generic situations which commonly occur in games such as hide-and-seek, cover systems for combat games, stealth for purposes of stealing and backstabbing in stealth and role playing games. Our work can definitely help in game designing in determining ambush positions and reducing the human testing needed for game levels by simulating common situations.

We are experimenting with 2D game levels containing obstacles. These game levels are simply polygons with holes. We are working with Unity game engine which makes it easier to import models, helps in visualization and makes it easier to work with 3D game levels considering different 2D perspectives. We assume the level is populated with two types of characters: an observer, moving along a path through the game level, and one or more survivors, whose sole purpose is to hide from the observer. In a hide-and-seek or exploratory context the observer path will be complex and exhaustive, while in other contexts the observer will be unaware of the survivors, and so we do not make strong assumptions about the path taken, and simply treat it as a route from a start position to a goal position in the game level, through which the observer moves at a constant speed.

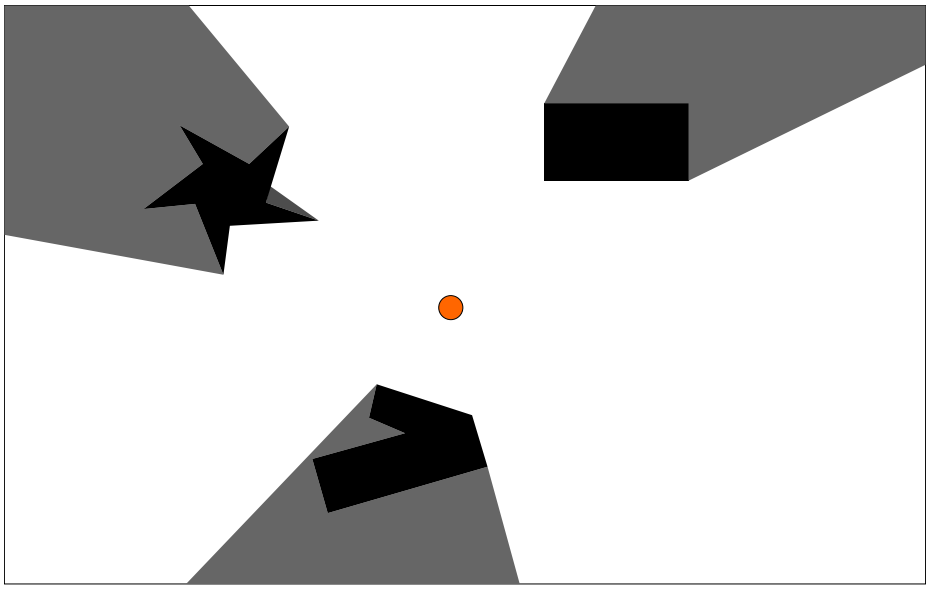


Figure 3.1 Visible and shadow regions in a 2D game level (3D as seen from above). The orange dot represents an observation point, grey areas represent non-visible regions, and black areas represent obstacles.

3.1 Basic Setup

Since we are considering a continuous observer path, we need to find visibility for the entire path. This is a form of weak visibility for which numerous solutions have been published, each giving complex solutions with respect to development and experimentation purposes. This however is not necessary for our purposes as we are looking for simpler solutions which can be accomplished through point visibility which is much more straightforward.

If every point of a polygon P is visible to some point of its edge e, P is said to be weakly visible from that edge. In our work, we don't calculate the visibility of an edge but of a line segment (which is the observer's path). By calculating point visibility of points on the line segment, we can get a close approximation of the visibility of the line segment. Point visibility of a point x in a polygon P is the portion of P visible from x. If we place a point source of light at x, the region which it illuminates is the visibility polygon.

By discretizing the observer path we will get a very close approximation of the results, which is completely acceptable, as the discretized path points are fine and change in visi-

bility polygons between neighboring path points is in most cases minor and gradual. We discretize the observer's path, and compute point visibility from each path point using a simple angular sweep algorithm, which we are going to explain shortly, to construct a star visibility polygon. The star visibility polygon can be viewed in the figure 3.1 as the white region. For simplification we considered the observer to have a full 360° Field of View (FoV) and infinite range, although these factors can be easily adjusted.

Our basic setup can be explained through the figure 1.1 in a previous chapter. The survivor characters aim to hide from the observer by remaining in the non-visible, or shadow part of the level. The shadow region can be understood by assuming the observer as a source of light and the obstacles/boundaries in the game level will create shadows in certain regions hidden behind the obstacles of the game level. The survivors will obviously start in a shadow region when the observer is at it's first path point.

3.2 Calculating Visibility Polygon

Now we will explain the general idea and the basic algorithm in detail for calculating visibility polygons for each of the observer path points. The optimal algorithm on which our naive approach is based on is given in the *section 8.5* of [25].

3.2.1 The idea

Now that we have our basic setup, we have to calculate the visibility polygons for each path point of the observer. We are going to calculate point visibility for each point assuming 360° FoV and range extending till the boundary of the game level. Figure 3.2 will assist us in explaining the method step by step. In the figure 3.2*(A)*, we sorted all the vertices inside the game level including the boundary vertices, in an anti-clockwise fashion starting from the positive x-axis with respect to the observer point shown in orange as the origin. The sorted vertices are numbered and are connected to the observation point through rays. In the next step we eliminate vertices which are hidden behind other obstacle/boundary edges. In the figure 3.2*(B)* we can see the eliminated rays corresponding to a few vertices in yellow. Vertices are excluded on the basis that the first point of intersection between

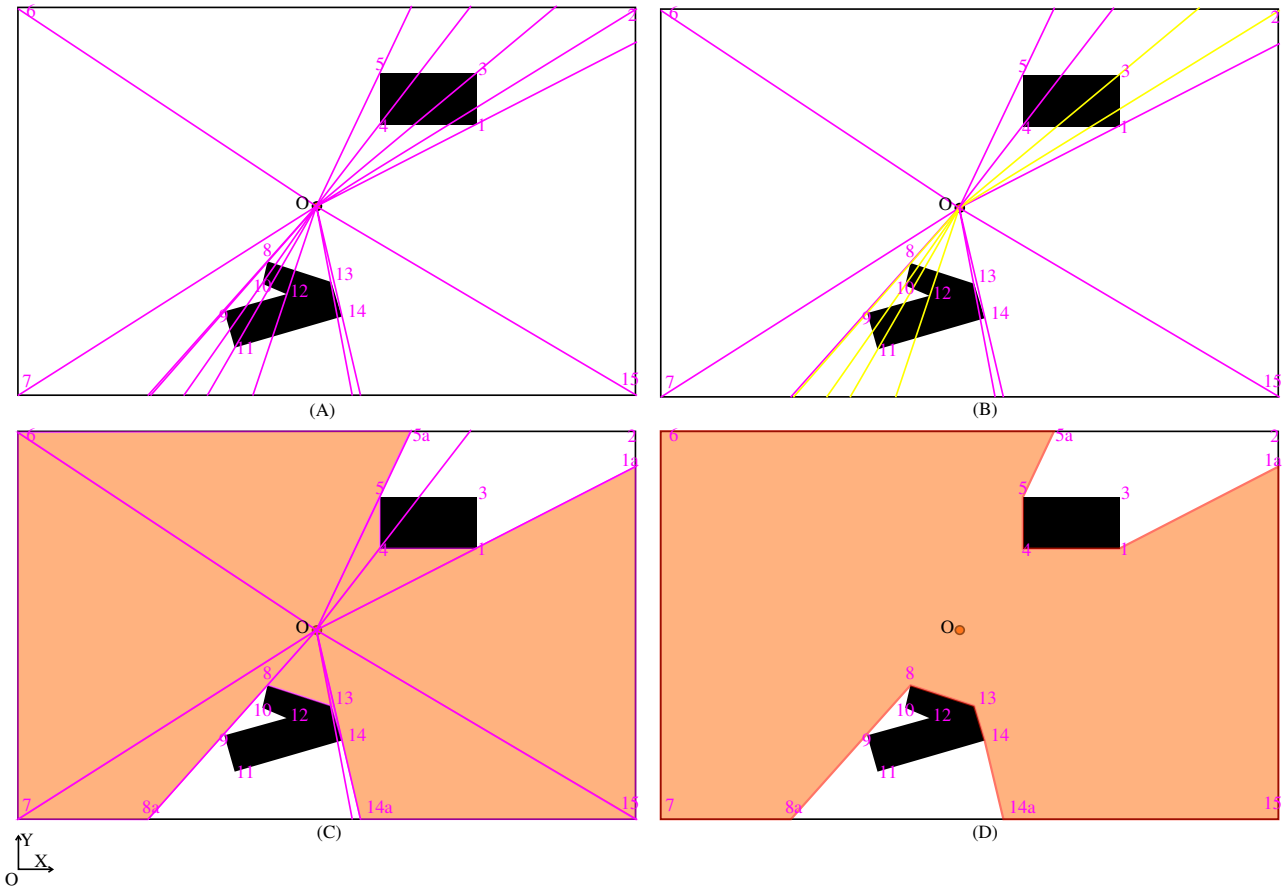


Figure 3.2 The 4 step process of calculating visibility polygons. The point *O* represents an observation point for which the visibility polygon is being calculated.

the ray (corresponding to the vertex) and any obstacle/boundary edge, starting from the observation point, is not the vertex itself. In the figure 3.2*(C)* we can see the markings of all intersection points for each ray. For instance, the ray connecting observation point and the vertex *1* has another intersection point at the boundary called *1a* besides vertex *1* itself. For ease of naming we will name this ray as *ray 1* and call other rays correspondingly. Now after eliminating invalid rays, we can focus on connecting two adjacent rays and using all intersection points of each ray to construct triangulations. Starting from *ray 1* and adjacent ray *ray 4*, the first triangle is formed between observation point *O*, *vertex 1* and *vertex 4*. Although *ray 1* has two valid intersection points *vertex 1* and *1a*, *vertex 1* is used for constructing the visibility triangle as the *1a* and *vertex 4* are not present on the same edge

of the obstacle/boundary. This is an important and simple concept in constructing visibility triangles between adjacent rays. We can see the same concept being used in the triangle O, *intersection point 5a* and *vertex 6*. After going through all adjacent rays, we have the triangles which will make up a star polygon of visibility. Each triangle shares a vertex, the origin O and has two other adjacent triangles on either sides. Constructing a star polygon by connecting these triangles is the last step. We will start by eliminating all edges of all triangles whose one vertex is the origin O. The next step is to simply connect all valid intersection points such as *5a* to the vertices with which they share a common ray starting from origin O, such as *vertex 5*. In the figure 3.2*(D)*, we can see the star polygon for the observer position at O, with an orange boundary with the following sequence of vertices *1,4,5,5a,6,7,8a,8,13,14,14a,15,1a*.

3.2.2 The method

The procedure to calculate a visibility polygon is explained in algorithm 1. We pass the observation point O and the *vertices* of all the obstacles and the boundary as a list to the procedure on line 1. These *vertices* would be in random order, so we have to sort them starting from the positive X axis (axes shown in figure 3.2) going anti-clockwise with respect to the observation point O, and save the sorted list of *vertices* as *verticesSorted* on line 2. The next step is to fill a list of all intersection points. We shoot rays from O towards all elements of *verticesSorted* and extend them till the boundary of the game level. Each ray would contain multiple intersection points of the ray with obstacles and the boundary. We save all the intersection points for $verticesSorted_i$ into $intersectionPts_i$ as a list of points on line 7. So *intersectionPts* is a list of lists of intersection points sorted in an anti-clockwise fashion, with each list $intersectionPts_i$ containing points which are sorted by their distance from O. We have to ignore the intersection rays for points which are hidden behind obstacles or/and boundary. The valid vertices to consider are the ones which are the first intersection points on the corresponding rays originating from O, which we can see from line 11 to 17. From line 18 to 27 triangulation is done on the valid intersection points on adjacent rays, pair wise in the same anti-clockwise fashion till we have gone all the way back to the first ray. The final step is to remove O from the edges of triangles and create a standalone star

Algorithm 1 Visibility Polygon

```
1:  procedure GENERATEVISIBILITYPOLYGON(O, vertices)
2:      verticesSorted ← SORTVERTICESANTICLOCKWISE(vertices)
3:      for i ← 1 to verticesSorted.Count() do
4:          intersectionPts[i] ← ∅
5:                          ▷ intersectionPts is a 2D array containing intersection points.
6:      for i ← 1 to verticesSorted.Count() do
7:          intersectionPts[i] ← GETINTERSECTIONPOINTS(verticesSorted[i])
8:      for i ← 1 to verticesSorted.Count() do
9:          intersectionPtsForVertex ← intersectionPts[i]
10:                         ▷ intersectionPtsForVertex is list of points on a straight line
11:         if intersectionPtsForVertex[0] ≠ verticesSorted[i] then
12:             intersectionPts[i] ← ∅
13:                                         ▷ Setting invalid vertices to null
14:     for i ← 1 to verticesSorted.Count() do
15:         if intersectionPts[i] = ∅ then
16:             intersectionPts.Remove(i)
17:                                         ▷ Removing invalid vertex indices
18:     T ← ∅
19:                             ▷ Triangle list T will save objects after triangulation
20:     for i ← 1 to intersectionPts.Count() do
21:         intersectionPtsForVertex ← intersectionPts[i]
22:         intersectionPtsForNextVertex ← intersectionPts[i+1]
23:         T.Add(TRIANGULATE(intersectionPtsForVertex, intersectionPtsForNextVertex))
24:     intersectionPtsForVertex ← intersectionPts[intersectionPts.Count()]
25:     intersectionPtsForNextVertex ← intersectionPts[1]
26:     T.Add(TRIANGULATE(intersectionPtsForVertex, intersectionPtsForNextVertex))
27:                     ▷ Connecting last line to the first to completely join all triangles
28:     P ← ∅
29:                             ▷ Polygon P will contain partial edges from T
30:     P ← CONSTRUCTSTARPOLYGON(T)
31:     return P
```

polygon from the triangle objects.

The visibility polygon changes with time as the observer moves on it's path. We compute all these polygons and save them for quick access. The next step is to calculate the shadow polygons which are nothing but the the remaining polygon/s after we take out the visibility polygon and obstacles from the game level. Obviously these change too, as the observer moves through the game level. The shadow polygons are the safe spaces for the survivor to hide.

Now we will take a look at the types of experiments and our hiding strategies since we have the visibility and shadow region data for the complete path of the observer to work with.

3.3 Hiding Strategies

We explore 4 main hiding strategies, which we will describe below. The first three are heuristic strategies which means that the survivor does not make any assumptions of where the observer will move in the future and will try to move towards a position where it will be hidden in immediate future. There is however one basic assumption for the survivor movements in the heuristic strategies. The survivor has a sense of the current position of the observer i.e. the survivor can sense when it is going to get caught at its current position if it does not move soon, while the observer moves to the next path point.

Apart from the heuristic strategies, we have a strategy for an optimal solution for any survivor position in the game level. The three heuristic and one optimal strategy does not take survivor collisions into account if there are multiple survivors present. This is because in most game situations, the number of survivor characters are limited and does not pose a frequent risk of collision and even if there is a chance, we assume that our discretized positions have enough space for a few survivor characters. We considered collisions in our *collision based* strategy which takes a look at a flock of survivors trying to hide while avoiding other survivors in the way.

Each game level is divided into a 2D spatial grid and we account for all the space in the game level except for obstacles. Each grid cell is a square and the the cell size is such that a survivor can fit perfectly inside it. For each strategy, a survivor can start from any grid

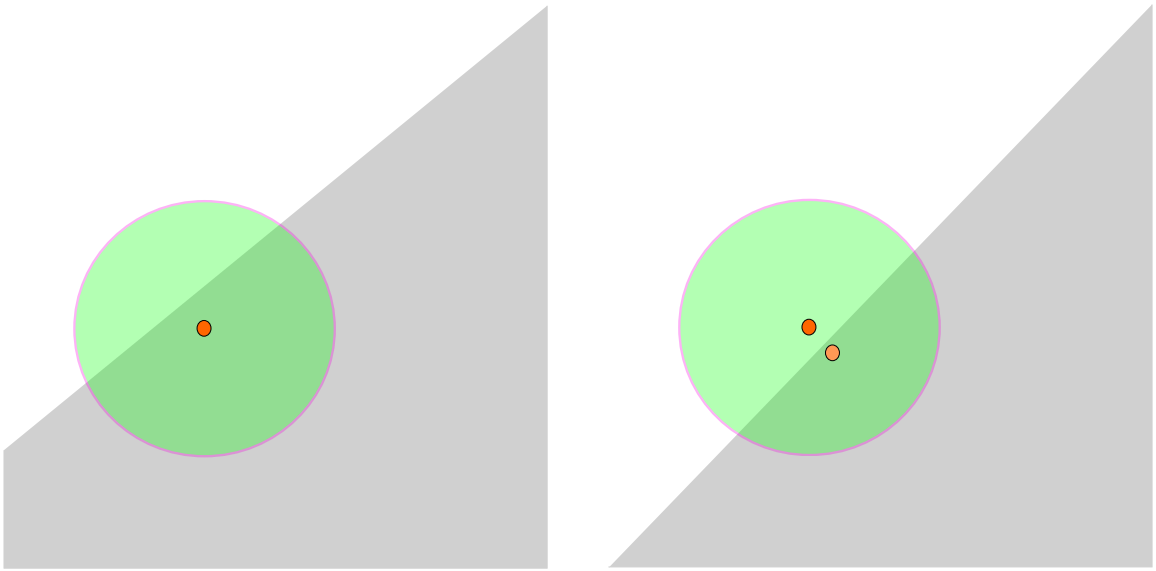
cell which is outside any obstacle and inside a shadow region for the observer's starting position.

The non heuristic strategies strategies make use of a further discretized environment i.e. time slicing. Each time slice contains a snapshot of the game level and shows data such as survivor position for each observer path point. There are few basic variable parameters such as the speed of the survivor relative to the observer, grid cell size, distance between observer path points and few others which are game level dependent and will be elaborated as we discuss each strategy. We will describe the three heuristic strategies first and then move on to the optimal strategy and finally the *collision based* approach (with collision avoidance).

3.3.1 Simple Greedy

Our first strategy is a straightforward greedy approach. For this, a survivor remains at its current position as long as it continues to be in one of the shadow polygons. As soon as survivor anticipates that it is close to getting caught (i.e. the shadow regions changes such that in next path position of the observer, the survivor will be visible to the observer), it tries to find a closest position near it where it will be safe. And this process continues for all observer path positions till the survivor is discovered or the observer reaches its goal position.

Figure 3.3 would help us describe this simple approach. The current survivor position is the orange dot, and the light green circle represents the potential movement area given by the $d_{survivor}$radius. The grey portion represents the current shadow. In the figure 3.3 (a), the current survivor position p_n is inside the shadow, but in the next observer step, it will be revealed, indicated by the new shadow boundary. Figure 3.3 (b) shows the shadow for p_{n+1}; the light orange position is selected as the closest hidden point inside the intersection of the circle of radius $d_{survivor}$ and the shadow for p_{n+1}. To determine the next safe position, we find the minimum distance or radius of movement which is necessary to be safe for the next observer position. This distance is always less than or equal to $d_{survivor}$. After finding this minimum radius, we construct a circle with this radius and consider 360 points on the periphery. Out of these 360 points, we select the first point inside the next safe region.



(a) (b)

Figure 3.3 Simple greedy hiding.

This strategy intuitively would work well when the change in the visibility polygon between each observer step is slow and gradual, giving a survivor enough time to move until it is out of immediate danger of being found. This approach also requires relatively little calculation however instinctively the reader can come to a conclusion that this strategy might not be suitable for cases when the visibility polygon changes dramatically, such as when an observer comes around a corner of an obstacle. We will see the effectiveness of this approach in the next chapter.

3.3.2 Max Shadow Greedy

The choice of the closest hidden spot in the simple greedy strategy is clearly naive in being minimally responsive to impending detection. The *max shadow greedy* strategy will give a better chance of survival through the maximum radius of movement for the survivor when it is in immediate danger of being seen. It is still however a greedy strategy in the sense that the survivor would take the full benefit of its radius of movement, yet it will only move when it is in critical position of being seen by the observer.

Our *max shadow greedy* approach is slightly more sophisticated, in aiming to select a new position that is not only in shadow, but as "deep" in shadow as possible within the survivor's movement radius. This does not improve the situation for gradual changes in shadow, but can better accommodate sudden larger changes. It is as efficient in terms of processing time as the simple greedy approach in finding the next safe position for the survivor but might have unnecessarily longer movements for it.

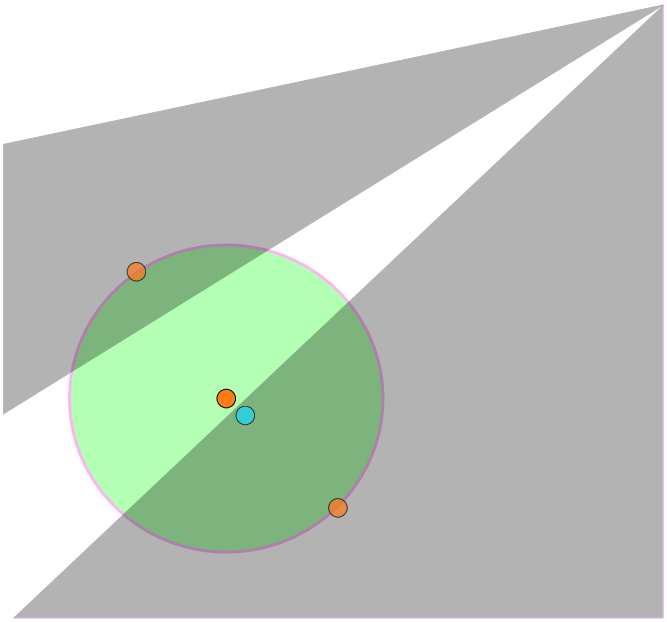


Figure 3.4 Demonstrating the max shadow greedy approach.

Figure 3.4 shows the design. The survivor is currently positioned at the orange circle at the center of the light green $d_{survivor}$ radius, but will be exposed given the new shadow for p_{n+1} indicated by the grey area. A simple greedy approach would move the survivor to the blue circle, minimally within the shadowed region. This point can tolerate further incremental changes in its distance to a shadow edge of $\leq d_{survivor}$, but a larger subsequent change will expose this point. The two light orange positions on the perimeter of the $d_{survivor}$ radius, on the other hand, are potentially safer, in that a sudden change in the shadow line of $d_{survivor} + k$ may be needed to expose them, where k is the current distance to the shadow. This of course does depend on the shape and evolution of the shadow region, and even our simplified example shows multiple reasonable candidates that are deeper in

shadow than the blue dot. To avoid a more complicated determination of which point is "deepest" in shadow (which is the basis of our next heuristic), here we compute the set of movement-perimeter arcs contained within shadow, and move to the center of the largest arc (i.e., the bottom right light orange dot).

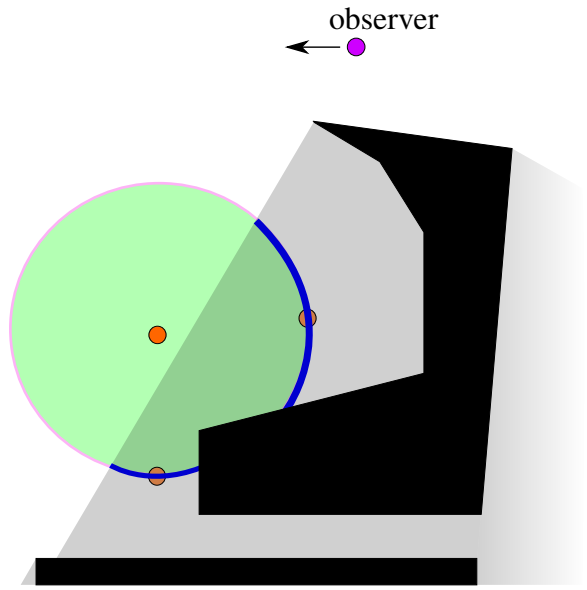


Figure 3.5 Largest shadow arc is not necessarily ideal.

This heuristic can of course fail in different ways. Sufficiently large changes in the visibility region can still not give the survivor enough time to hide, with no new candidate positions on the $d_{survivor}$ radius. The choice of largest arc can also fail, even when it does lead to a larger shadowed region than other choices; figure 3.5, for example, shows a situation where the survivor has two arcs on the boundary of $d_{survivor}$. Here the bigger arc leads to a point intuitively deeper in shadow, but which also has potential to trap it in a concavity, where continued movement of the observer (purple dot at the top) to the left will certainly expose it.

3.3.3 Shadow Centering

Both the greedy and max shadow greedy strategies are "last moment" survival strategies, only ever reacting to imminent discovery. These strategies will only start to work if the

survivor is in immediate danger. These are simple strategies which require little processing. We thus also introduce a strategy that aims to constantly improve a survivor's position by continually looking for a safer position inside the current shadow.

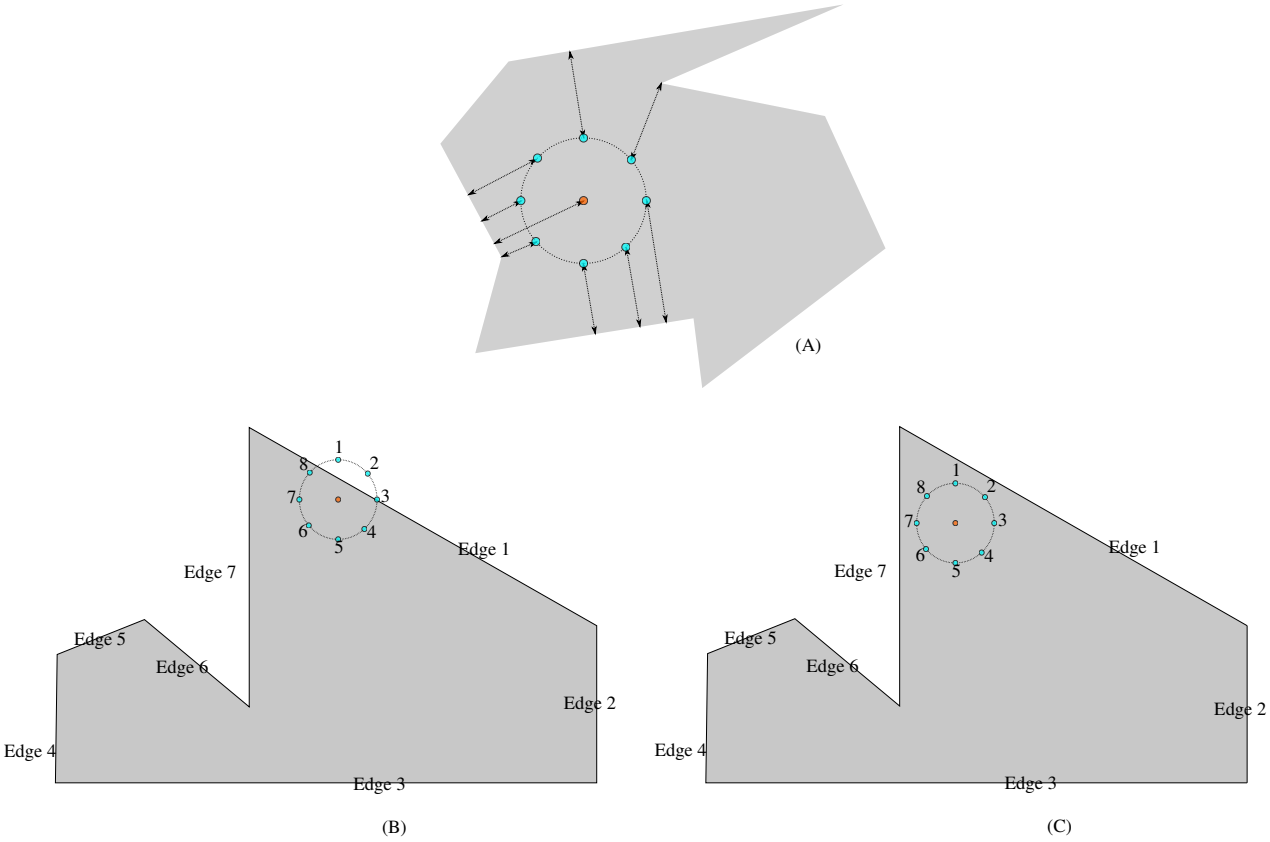


Figure 3.6 The shadow centering computation. The orange dot represents the current position of the survivor inside irregular shadow polygons (grey area), the circle around it represents the radius of movement (depends on the speed) and the aqua dots are the next possible positions. (**A**) shows a general case of an irregular shadow polygon and how the distances of each position to their closest shadow edges are considered. (**B**) and (**C**) are specific cases as explained in the description.

Heuristically, a survivor that stays deep within a shadowed region has an improved opportunity for staying in shadow over someone at the periphery, as a larger portion of their reachable future positions are likely to be in shadow in the future. Our approach is thus to have the survivor try to remain "centered" inside the shadow. This approach is not necessarily optimal in every case.

Depending on each game level, the shadow areas are most of the time non-convex polygons. The center of a non-convex polygon is not well-defined as there might be several local centers. As the observer moves, the shadow polygons change and so does their local central regions. This makes, a fixed central destination position, for the survivors unfeasible to compute in real time. Thus we will move on to heuristically computing a neighboring position which takes the survivors towards the central region for the current shadow polygon.

By performing a number of experiments on different game levels, we came up with a reasonably effective approach which is an aggregate of three basic computations for each of the next probable positions.

We will explain the basic concept through the figure 3.6. The orange dot represents the current position of the survivor inside an irregular shadow polygon (grey area), the circle around it represents the radius of movement (depends on the speed) and the aqua dots are the next possible positions. Even though the survivor is still hidden, an improved position is sought by selecting from 8 points on the survivor's reachable periphery plus the current position. For each of these 9 points, three numbers are computed, the distance to the closest edge of the shadow polygon, the distance to the furthest visible edge of the shadow polygon (visible from current position of the survivor) and the distance to the nearest vertex of the shadow polygon. The motivation behind this approach is that the survivor has the best chance of survival if it is furthest from the closest shadow edge, going towards the furthest shadow edge and moving away from the nearest vertex. These three actions can be quantified in actual distances and the aggregate of them for each point gives us the best possible position to move to. Figure 3.6 *(A)* exhibits the distances of each position to their closest shadow edges. Similarly distances to closest vertex and furthest edges are computed. The position which has an aggregate highest score for the aggregate of all three values is then selected as the next position of the survivor.

Let's assume that these three numbers are *x1*, *x2* and *x3* and the weights assigned to each are *w1*, *w2* and *w3* respectively, *x1* being the distance from the nearest shadow edge, *x2* being the distance from the furthest shadow edge and *x3* is the distance to nearest vertex. In our experiments we set the weights as $w1 = 1.0, w2 = 0.5$ and $w3 = 0.2$. $w1 > w2 > w3$ because the immediate need to get away from the nearest shadow edge is the greatest. For

x1 and *x3* we are looking for the largest values but for *x2* we are looking for the lowest value as we need to get closer to the furthest edge of shadow polygon. So our aggregate for a single point *j* out of the 9 points is

$$w1 \times x1_j + w2 \times (\max_{1 \leq i \leq 9} x2_i - x2_j) + w3 \times x3_j$$

By tweaking the values of $w1, w2$ and $w3$, better results could be obtained for each level and different observer paths, but we are using fixed values for standard testing.

Figure 3.6 *(B)* shows why a combination of techniques is more effective than just choosing one. Intuitively it makes sense that if we just constantly move away from the closest shadow edge, we will remain in the central region of the shadow. This is however incomplete as in the figure 3.6 *(B)* the closest edge is *Edge 1* so naturally the point *6* will be selected as the next safe position. Let's assume that the change in shadow polygon is negligible for the next move and the figure 3.6 *(C)* shows the survivor in it's new position closer to *Edge 7*. This new position is much safer, however the survivor has no reason to move from here towards any of the numbered points on the periphery of the max movement circle, as the current position is furthest from the nearest edges *Edge 1* and *Edge 7*. Point 4 is the same distance from *Edge 1* as the current position and point 5 is the same distance away from *Edge 7* as the current position. However considering the whole shadow polygon we can see that there are other much safer options. We need some other incentives to move towards a more central region away from any immediate risk of discovery. This is the reason for depending on three techniques.

This strategy seems better at handling sudden changes in the shadow regions. Moreover, it is adept at reducing the chances of being stuck in a concavity of an obstacle. The survivor is constantly on the move which requires more processing resources than other strategies.

3.3.4 Optimal

In order to better understand our hiding strategies we compare them to an optimal, or clairvoyant strategy. This approach assumes a known deterministic movement for the observer, and is computationally expensive, but can determine the full set of ideal movements and

positions for the survivors to stay hidden as long as it is viable under survivor movement limits (radius of movement depending on its speed).

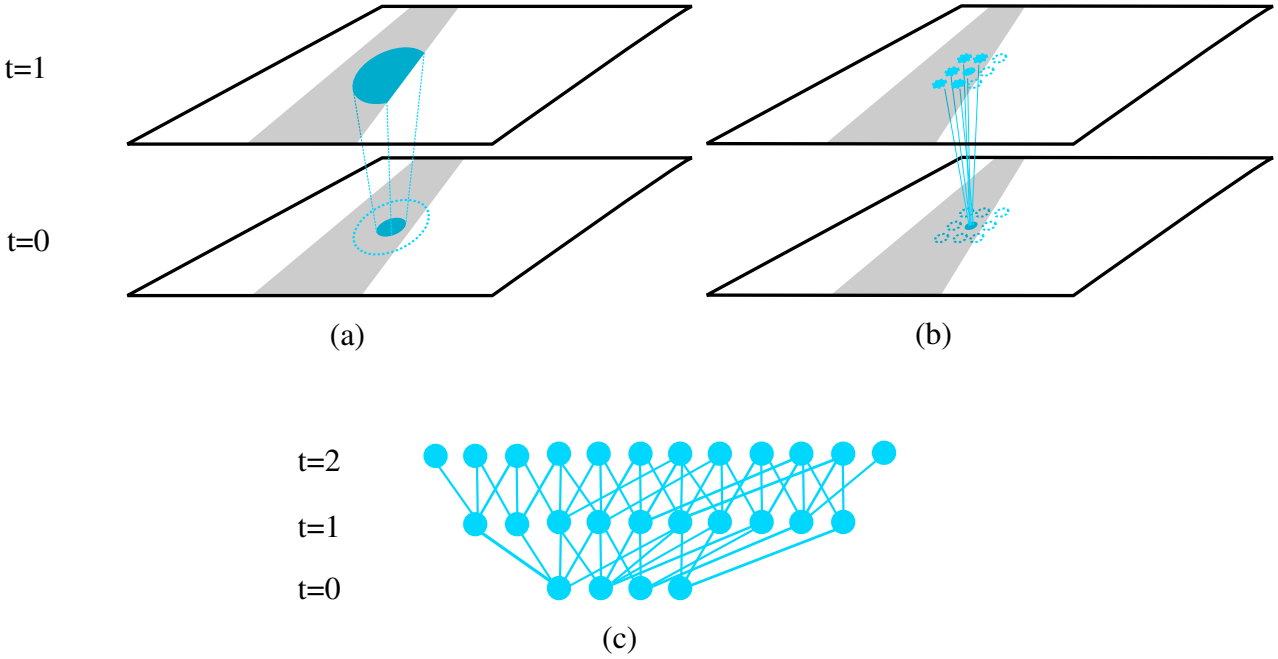


Figure 3.7 Optimal computation.

The basic idea using a discretized time model is illustrated in figure 3.7*(a)*. The bottom layer shows a point in the level domain, currently within shadow. The dotted radius around it indicates the set of possible positions this point can reach within a given time unit; the faster the survivor can move the larger this area, but in general some of this area will be in shadow, and some not. The layer above it shows the next time unit, where the resulting set of possible positions has been intersected with the shadow polygon in that time frame (which had negligible change) to give us a new set of points, each reachable from the initial point, and each still in shadow. This process is then repeated to compute a set of points in each subsequent time unit, each time expanding the set of hidden points according to reachability, and intersecting it with the shadows available in that time, as computed given the observer's next positions. Note that this model assumes time is discretized with sufficient granularity to ensure the shadow area changes in a simple, linear fashion, and so allows us to guarantee a motion from a hidden point in one time frame can be connected to

any reachable point still in shadow in the next frame.

The results of this process gives us a complex, 3D manifold that encodes all successful motions and positions from a given starting region. Our actual process further discretizes this, however, both in order to avoid the need to explicitly model arbitrary polyhedra, and to also give us a simple graph form we can use for easy pathfinding choices. Figure 3.7*(b)* shows our discrete positions in space for two time frames. Here we assume the level space is organized as a 2D grid, and instead of a polygonal region we compute a reachable set of points, pruning them in the next layer of any points now outside the new shadow region. We use this to construct a 3D graph, connecting each point in the lower layer to each point in the next layer up that is both reachable and within shadow. The result is a graph where the number of vertices depends on the time and space granularity, and the number of edges depends on survivor speed as a branching factor, but restricted to shadow locations in each layer. In the figure 3.7*(c)* we flattened the 3D graph to show the complexity of edges. At $t = 0$ we only show four positions which are safe. As the time progresses, we connect points in different time frames showing reachability and safety. We construct the rest of the 'time frame' graph till the observer reaches the end of it's path. Through this complete graph we can compute the maximum reachability/survivability of a survivor if it was placed at a certain position at $t = 0$. The problem breaks down to just finding the longest path through this directional graph starting from a position at $t = 0$.

If a path exists in this graph going from a starting point at $t = 0$ to a point in the layer at $t_{\max}$, then a survivor at that point has a set of movements that allows them to be safely hidden for the entirety of the observer movement. The number of positions which have such a successful strategy can also be computed, giving us a measure of the maximum amount of level space that can be filled with hiding survivors (ignoring survivor movement collisions), and the maximum length of a path through time from each initial position can also be computed to give a relative scale of the longevity of each hiding spot.

3.3.5 Collision Based

Till now we have considered one observer and one survivor at a time, with the only limitation for the survivor movements to be the obstacles and the visible regions to the observer.

We now introduce multiple survivors. In fact we fill the complete shadow region at the beginning of the experiment with survivors and let them move as a group, though they do not communicate with one another and only make decisions to the best of their own survival. This approach is an experiment to observe how survivors would move as a group while avoiding their neighbors and the visible regions. This is not another strategy of survivor movement, rather it is an efficient use of already existing 3D graph from the optimal strategy to minimize collisions by traversing the graph using a heuristic approach.

We will try to explain this approach with an example. Lets assume that we have 100 positions which are in the shadow regions at $t = 0$ when the observer is at the starting position of its path. We place 100 survivors on each of the starting positions or each of the nodes of the 3D graph. Now the goal of the each of the 100 survivors is to move to safe positions/nodes at $t = 1$ time frame and so on. Note that each node can be occupied by only one survivor unlike all the previous strategies which ignores the survivor collisions. Filling the nodes in the next time frame is the most crucial step. We tried a random selection method but the results were neither consistent nor efficient.

We will take a simplified case in figure 3.8 to explain our approach at allocating survivors to nodes. Note that we take a heuristic approach and it might not be the most efficient as we are trying to make the best possible choice locally for each time frame. In figure 3.8*a* we show a flattened 3D graph from the optimal strategy. The cyan colored nodes are not occupied at both time frame levels. The edges represent reachability which directly depends on the speed of the survivors. This is a directional bipartite graph with direction always from lower time frame level to the upper. Figure 3.8*b* shows the nodes at t=n are yellow which signifies that survivors have filled these nodes. The priority is given to the survivor which has the least amount of options available or in other terms, the least number of edges connecting to the node the survivor occupies. In the figure 3.8*b* the clear option is *node 4* so we move this particular survivor to its only option available in the next time frame $t = n + 1$. The next survivor to move forward is the one with 2 options/edges which is *node 1*. This process continues till all the survivors who can move from time frame $t = n$ have moved to the next time frame $t = n + 1$. The success percentage might not be 100% as it depends directly on the edges available for the nodes. This heuristic approach seems a better alternative to random selection.

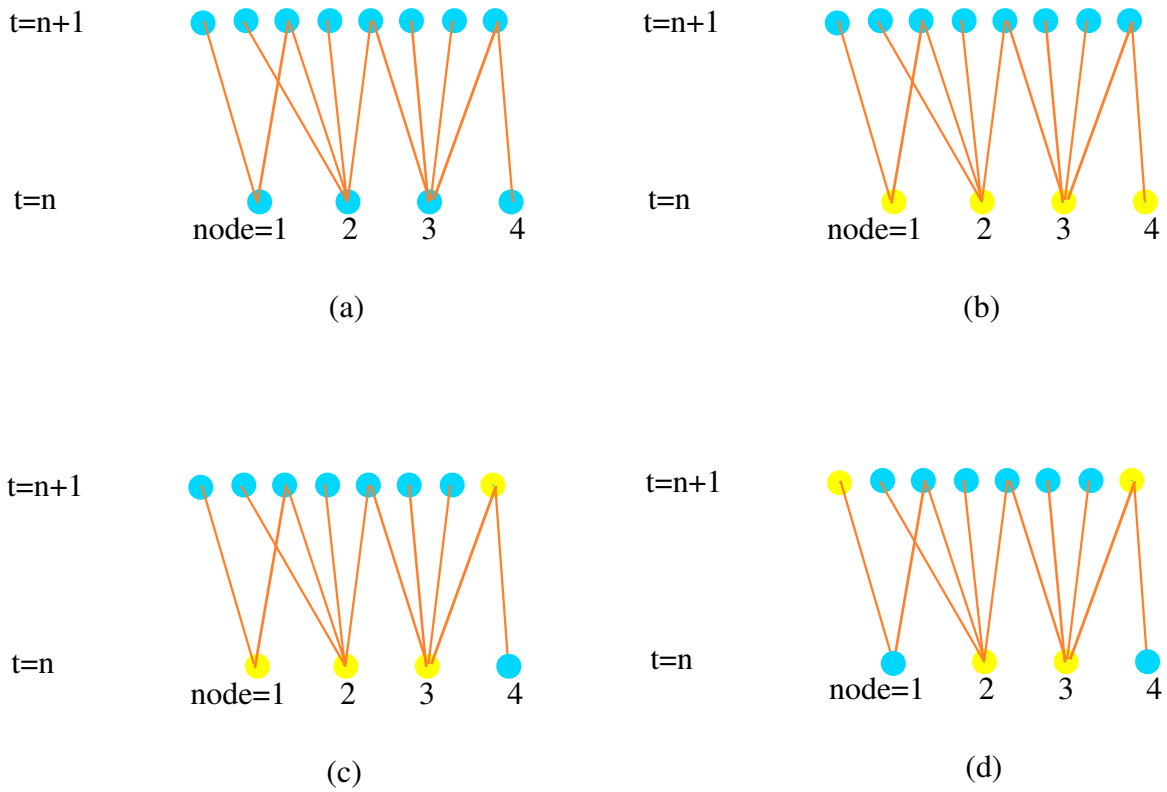


Figure 3.8 Collision based selection

This might not be the most comprehensive approach, but it gives us a realistic idea of survivability of NPCs. This approach however is marred by the same issues as other heuristic strategies. The local decisions for each survivor are prone to them getting stuck into positions/nodes which might have fewer and fewer options in the future time frames as we intentionally give priority to nodes with lesser options/edges to progress ahead, instead of nodes with more edges to make sure that more number of survivors move on to the future time frames.

We tested this approach on several game levels for a number of survivor speeds. This approach is not effective for survivor safety for the observer path, which was the motivation for implementing this method. The main reason is that it is not an intelligent approach in finding the longest path for each survivor. Lets consider this method in reverse. If we place all the survivors at nodes/positions at the last time frame when the observer is at the end point of its path and treat it as a starting position, there might be a better chance of survival

for the survivor NPCs. We use exactly the same approach as the figure 3.8 but in reverse time frames.

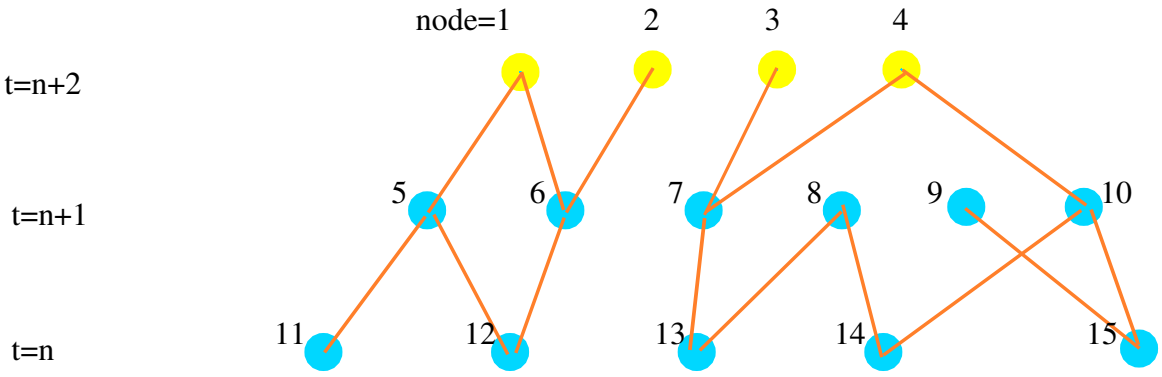


Figure 3.9 Reverse collision based selection

In the figure 3.9 we assume that $t = n+2$ is the last time frame when the observer is at the end position of it's path. We ignore that the direction of time is forward and assume that the observer is going backwards on it's path towards its starting position. As the direction of movement is reversed for the observer so is the direction of edges in the flattened 3D graph. We place the survivors at all the available nodes/positions at $t = n+2$ time frame and use the same collision avoidance techniques which was explained before in the figure 3.8. We have numbered the nodes/positions. We can see that *nodes 8* and *9* are dead ends if we were using the previous approach and survivors would get stuck if they chose to move to either of these nodes from previous time frames. But in this "reverse" approach, *nodes 8* and *9* would never be a choice for the survivors which currently are at $t = n+2$, to move towards, as there are no edges connecting them to the nodes at time frame $t = n+2$.

In the normal collision based approach the survivors have two obstacles, the dead ends and collisions with other survivors. In the "reverse" collision based there is no dead end issue. The experiments on same game levels for same survivor speeds and same observer paths, the "reverse" approach gives a lot better results (which we discuss in the next chapter) of providing us with a more comprehensive probability of survival.

Chapter 4
Results and Discussion

In this chapter we will introduce and experiment on several game levels which will consist of essential elements such as an observer, survivors, an observer path and obstacles. After performing experiments on each of the levels, we will discuss the effects of variation in parameters such as strategy engaged, survivor speeds and observer paths. The observer speed is held constant while we change the survivor speeds relative to the observer. The survivor speeds are chosen such as to show clearly the effect of choosing different radius of survivor movements for each game level. As we go along, we will also review our results and examine some special cases. We will end this chapter by discussions on comparisons of strategies and the effects of variation in parameters on all game levels tested.

As we progress, the levels will become more complex and larger in size. Our experiments are performed on some theoretical levels to show the affects of each strategy, and on some real game levels, from stealth, first person shooters (FPS) and role playing game (RPG) genres, to show the applicability of our strategies. The game levels which are directly inspired by some real game levels, have a few minor modifications to fit our needs for experimentation. These modifications have no real impact on the results but help us in performing experiments without numerical inaccuracies in the Unity environment.

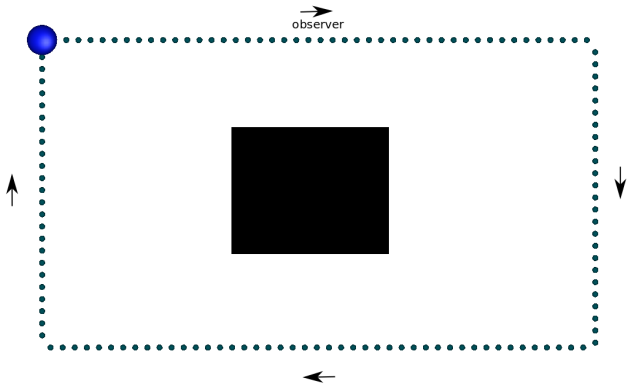


Figure 4.1 Level I setup

4.1 Level I

We begin with a simple game level as a proof of concept. Figure 4.1 shows a single obstacle in the center of the game level and the observer path goes around the obstacle and end at the same position it started. The navy blue circle represents the starting and ending position of the observer. The observer moves at a constant speed on its path while we will vary the speed of survivors. We consider three different speeds for the survivor, 60% of the observer speed, 80% of the observer speed and 110% of the observer speed. The level is discretized to place the survivors at their starting positions, which are the shadow regions at the beginning of the observer path. Let's take a look at the results for all strategies for each speed of the survivor.

4.1.1 60% of the observer speed

In the figures 4.2, 4.3, 4.4 and 4.5 we can notice several green points or squares, of different shades of green. The points/positions represent the starting positions of the survivors when the observer is at its starting position. The complete game level is discretized but we ignore the starting positions which are not in the shadow regions at the beginning. The shades represent each positions safety level. The darker the shade of green, the more safe the

survivor is if it starts from that position. This is easily distinguishable in the figure 4.2 as the darkest green points are the ones which survive the whole observer path.

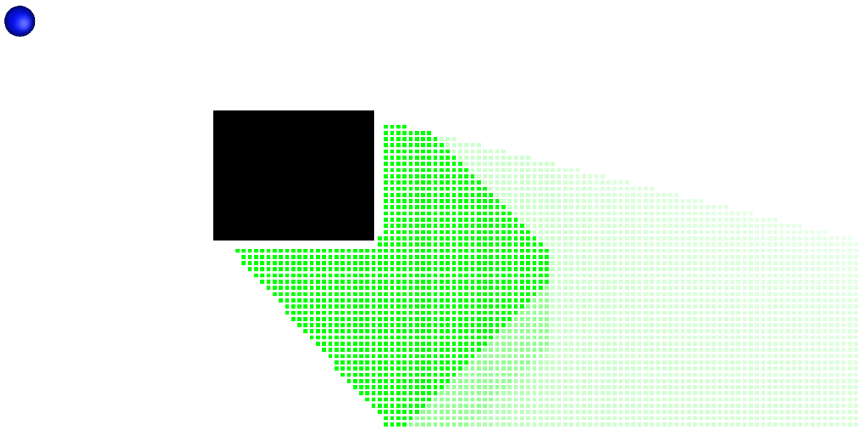


Figure 4.2 Optimal strategy applied to Level I at 60% of the observer speed.

For the *Simple greedy* approach, at this speed, the darkest green positions in the figures 4.3 represent the longest a survivor can remain undiscovered which is approximately 44% of the observer path. No survivor is able to remain hidden after the observer has traveled about 44% of the its path. The number for *Max shadow greedy* , as seen in the figure 4.4, is about 45% and for *Shadow centering* strategy, as seen in the figure 4.5, its about 46%.

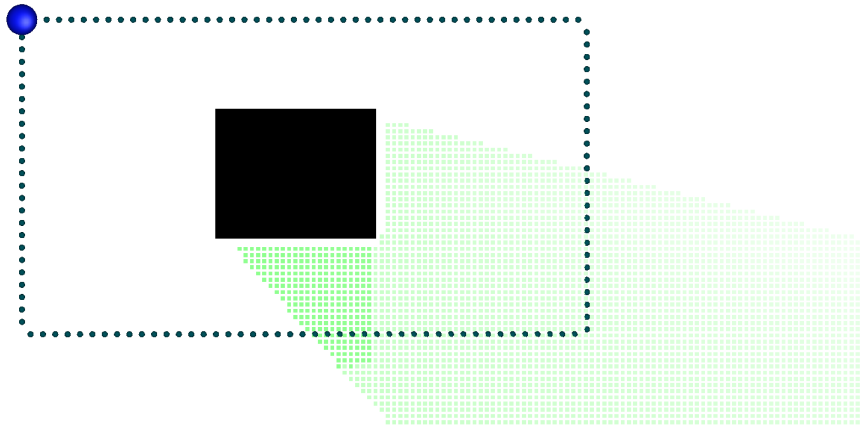


Figure 4.3 Simple greedy strategy applied to Level I at 60% of the observer speed.

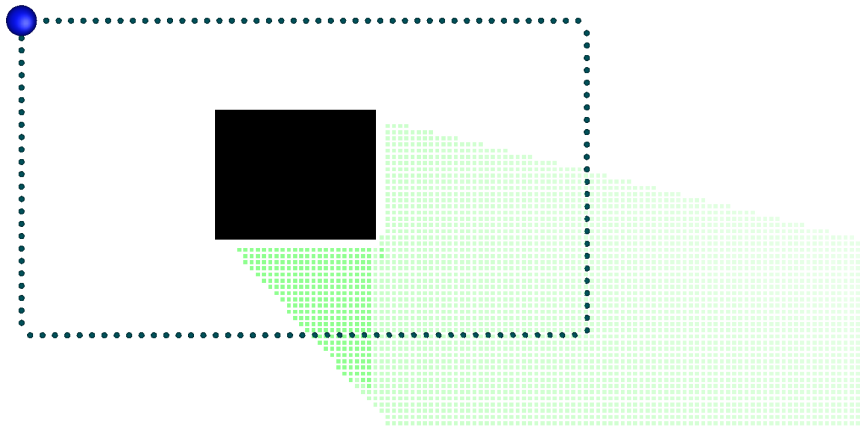


Figure 4.4 Max shadow greedy strategy applied to Level I at 60% of the observer speed.

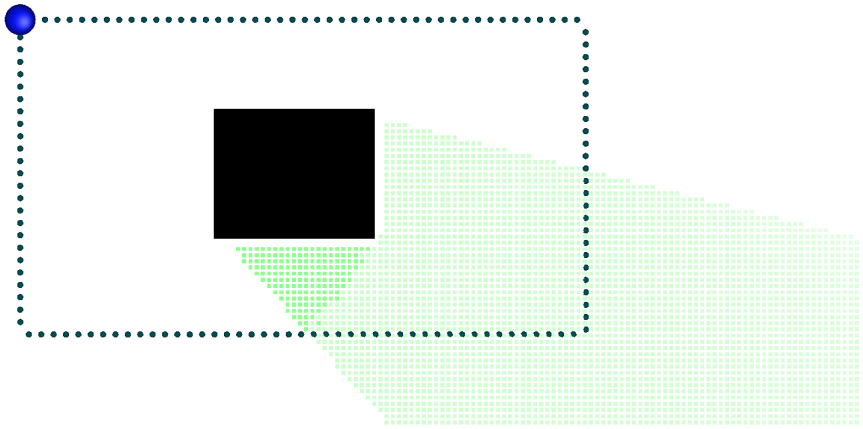


Figure 4.5 Shadow centering strategy applied to Level I at 60% of the observer speed.

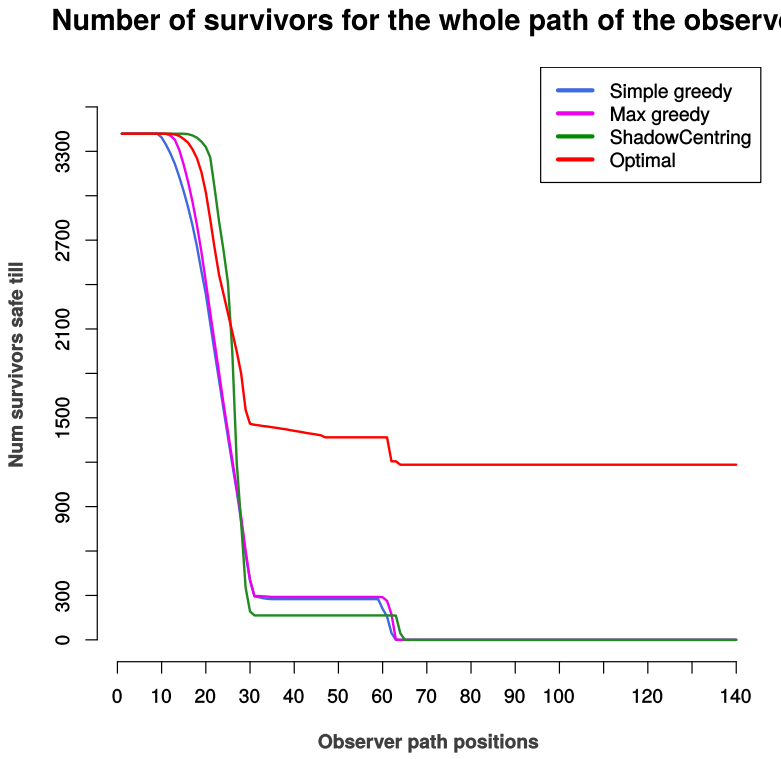
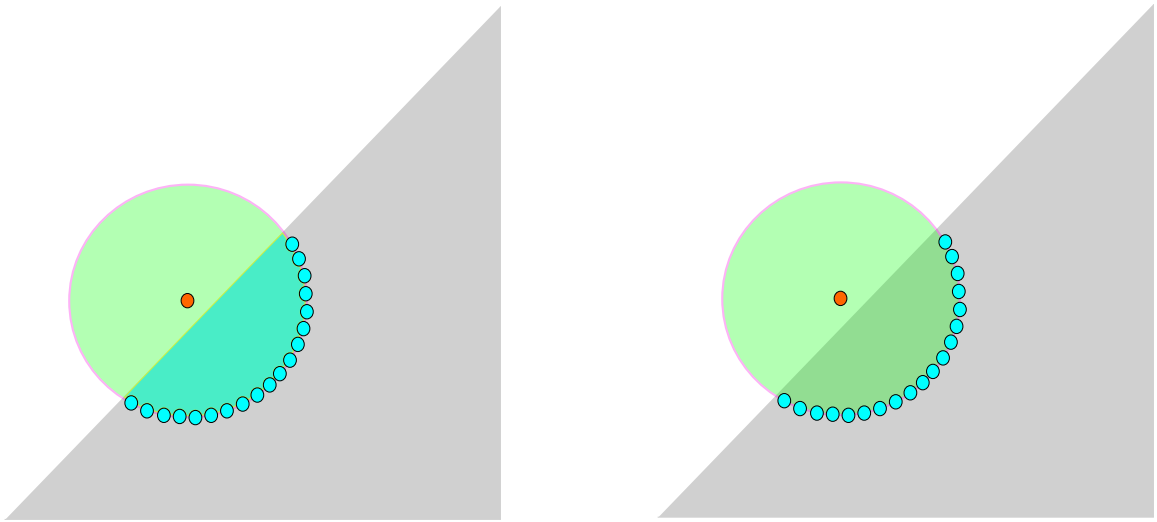


Figure 4.6 Number of survivors remaining till the observer path for 60% speed of the observer.

From the figure 4.6 and comparing figures 4.3, 4.4 and 4.5 to figure 4.2, we can compare the effectiveness of each heuristic strategy, for the number of survivors remaining at any observer path position, to the *optimal* strategy. Figure 4.6 shows that at this low speed, there is not much difference between the two *greedy* strategies. *Shadow centering* performs a bit better in the beginning, but drops sharply around observer path point 28. No heuristic strategy is able to save the survivors till the end of the observer path at this speed of movement.

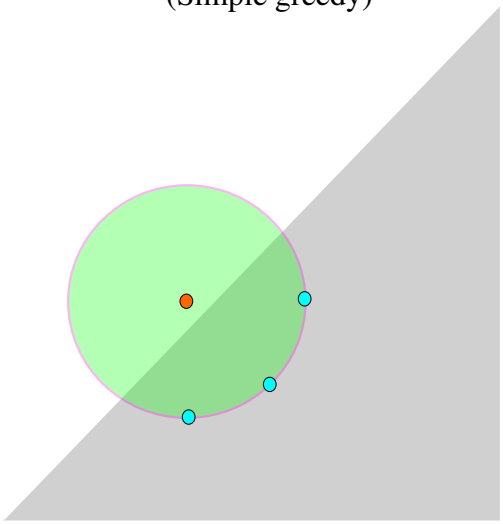
The sharp drop in number of survivors remaining happens when the observer comes around a corner and a large portion of area gets discovered. The *Shadow centering* strategy seems to have sharper drops, and as we progress further to higher survivor speeds, this trend will be exacerbated. This is due to the fact that each survivor tries to move towards the central region of the local shadow polygon and the net effect is a congregation of survivors in small central region (remember we do not take into account the survivor collisions in these heuristic strategies). When the observer comes around a corner, this region which was a central hot spot for the survivors a few observer steps before, becomes visible to the observer and all the survivors hiding in that region gets discovered.

There is an anomaly in the figure 4.6 which needs to explained. The *optimal* strategy is supposed to be the best one for the survivors at any observer path position, yet *shadow centering* surpasses it for the number of survivors remaining, for a slight portion of the observer path starting from the 20th observer path point. This is not a bug in the algorithms but a side effect of the points chosen to move for different strategies. We will look at the figure 4.7 for the explanation. The orange point represents the current position of the survivor and the green circle around it represents the extent of its movement or the circle of movement which depends on the speed of the survivor. The grey region represents the shadow for the next observer path position; so our assumption is that the survivor is safe in the present observer position but it is going to be discovered for the next path position as it is not inside the grey region. The blue region or points represent the potential safe points/region for each strategy. The *simple greedy* strategy can choose any point inside the safe arc or on the boundary of the circle of movement, the *max shadow greedy* and shadow centering will only choose on the boundary for the maximum safety, while the *optimal* strategy is designed such that only the predefined discretized points, which are

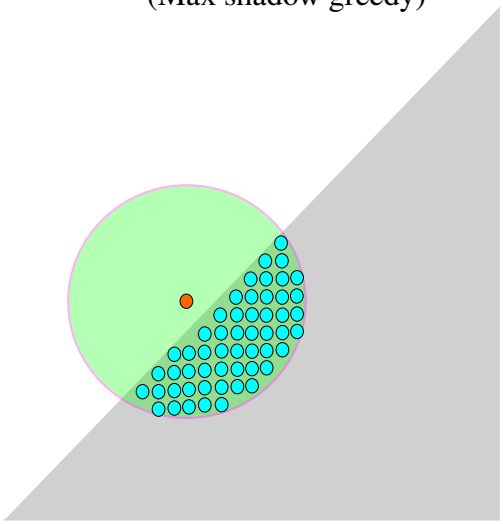


(Simple greedy)

(Max shadow greedy)



(Shadow centering)



(Optimal)

Figure 4.7 Explanation for the anomaly in figure 4.6.

inside the safety arc, are viable options while the boundary points are ignored. This is done to eliminate any confusion as to what defines that which points on the boundary, can be considered safe or unsafe. So the *optimal* strategy is slightly restricted and the distance the survivor can move is marginally smaller than other strategies. This might give somewhat advantage to other strategies in our experiments for a few observer path positions, but at the end of the observer path, the *optimal* will surpass all other strategies as it is designed to make smarter decisions.

4.1.2 80% of the observer speed

We increase the survivor speed, to study the effect of higher speeds for each strategy. We can directly compare the effect by comparing the figure 4.6 with the figure 4.8. We can see that there is improvement in the results for *Shadow centering* strategy. Some survivors are able to remain unseen till the observer path position 100, while for 60% of the observer speed, none remain undiscovered after $65th$ observer path position. The *Simple greedy* and *Max shadow greedy* strategies virtually have no gain in the number of survivors remaining undiscovered. This is an important observation for such a game level with obstacles which are independent (i.e. the obstacles which have freedom of movement all around them for the survivors).

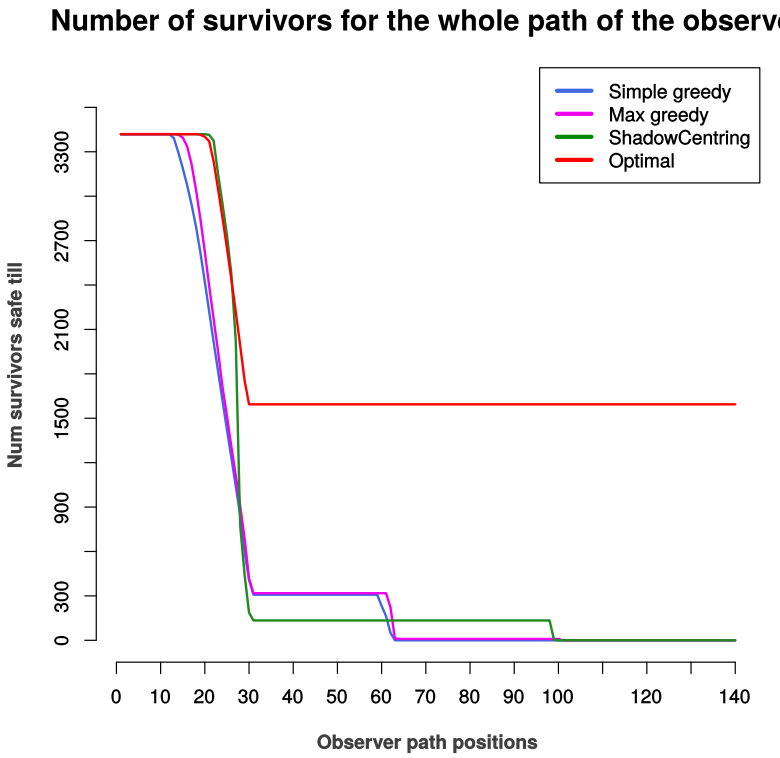


Figure 4.8 Number of survivors remaining till the observer path for 80% speed of the observer.

4.1.3 110% of the observer speed

Even at 10% more speed than the observer, none of the survivors were able to survive till the whole path of the observer, as seen in the figure 4.9, but the heuristic strategies performed slightly better than the previous speed of 80%. We should remember, the closer the plot for a heuristic strategy to the optimal plot, the better the strategy is, for that game level.

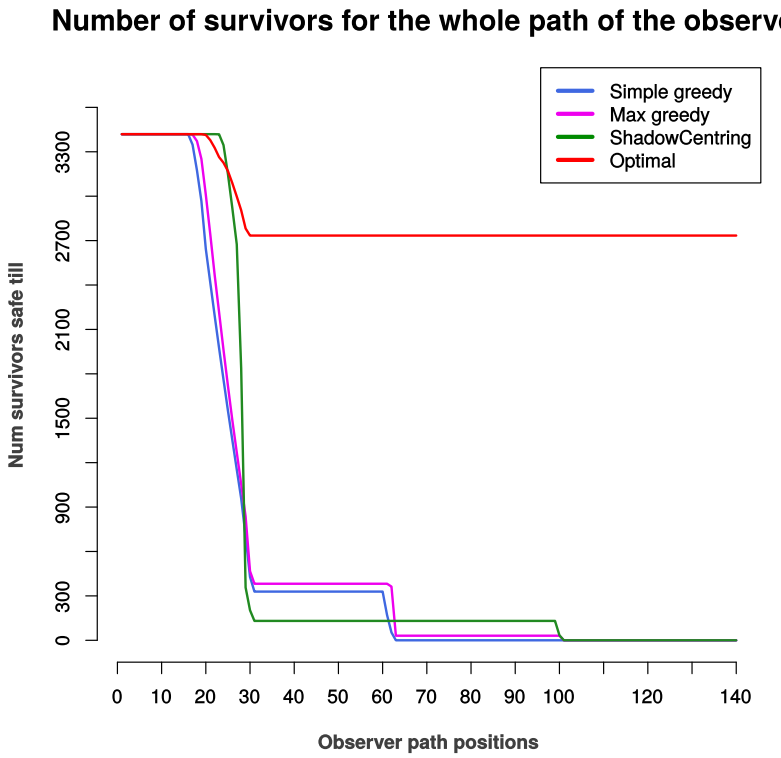


Figure 4.9 Number of survivors remaining till the observer path for 110% speed of the observer.

Although none of the heuristic strategies worked out well in our experiments, there was improvement in survivability, by increasing the survivor speeds for this simple theoretical game level. All our results for this level have steep drops in number of survivors remaining, due to the observer coming around a corner and discovering a lot of unseen area where many survivors were hiding. More smoother results can be obtained using a circular obstacle, as there will be no chance of the observer coming around a corner and discovering a large portion of unseen region in a single observer step.

4.1.4 Collision Based

We will take a look at the collision based approach now but we cannot directly compare it to any of the other strategies because we perform the experiment backwards, where the observer starts from its last position till it reaches its first position. So the observer path is

completely reversed for this approach. Another reason is that we do not consider collisions in any other strategy but *collision based* approach. We explained the complete process in the subsection 3.3.5 on the page 28.

In the figure 4.10 the starting position on the x-axis is actually the last position of the original observer path and vice versa. The curves in the figure for different speeds are quite intuitive in that the speed of the survivors is the main factor for better survivability. The survivors get discovered as the observer moves on its path as this is another heuristic approach. The survivors have plenty of empty space to move, but due to the collisions between themselves, they are not able to move efficiently as a group.

After we have performed this approach, and analyze the results with the original observer path as a reference, we can say that even with a simple heuristic strategy such as this one, there are clear survivor paths for some survivors which they can take to hide from the observer without bumping into each other. For instance, if we start with a 100 survivors for this strategy, and at the end of the experiment, 5 survivors were able to remain hidden from the observer without any collisions with other survivors, we can be sure that there are at least 5 survivors who can independently hide from the observer for its complete path using just a simple heuristic strategy. This information might be helpful in designing a level for stealth, RPGs or FPS games.

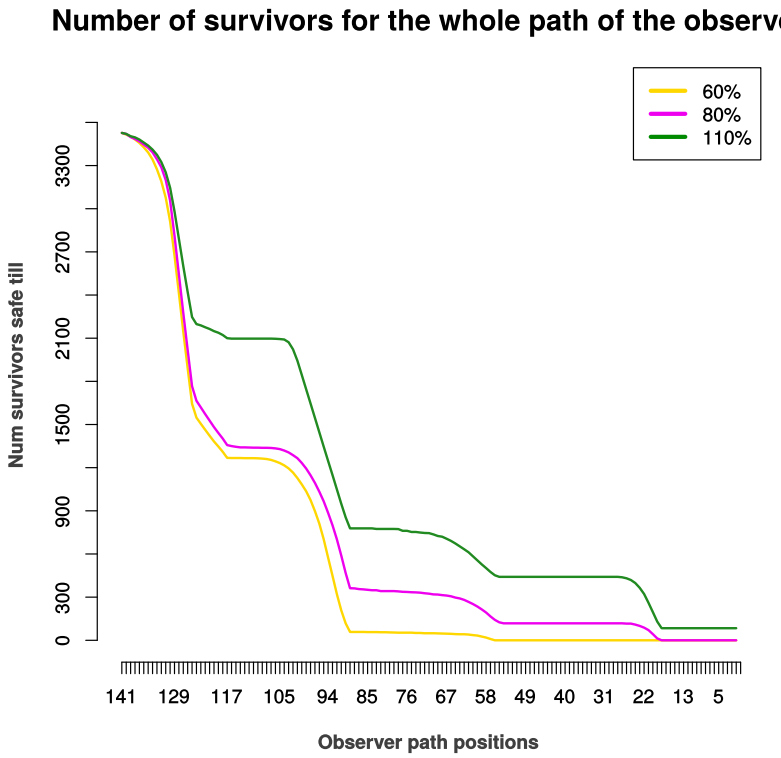


Figure 4.10 Number of survivors remaining till the observer path for collision based approach

The result of this experiment are as expected, the greater the speed, more number of options/positions are available for the survivors which means less collisions and more survivors remain at the end. This is true for such game levels where there is more available open space for the survivors. We will observe similar trend in similar cases in other game levels, and different trends for more complex game levels as we progress.

4.2 Level II

The level now we experiment with, as seen in the figure 4.11, is a little more complex with multiple obstacles including a non-convex polygonal obstacle. The starting position of the observer is the blue circle while it ends its path at the pink circle. This game level will give a lot more space for the survivors to hide behind the obstacles as the observer moves

through the game level.

There are a couple of interesting positions to hide in this level. One of them is the concavity in the obstacle at the bottom right. We will observe how the greedy heuristic strategies are weak at handling these concave hiding spots if these spaces are seen by the observer while moving on its path. Another special spot is above the top left obstacle. A triangular region remains unseen for the whole of the observer path. So even a static survivor in this safe region will remain undiscovered for this observer path.

We experimented with 3 different survivor speeds, 70%, 90% and 150% of the observer speed, for optimal and heuristic strategies.

Figure 4.11 Level II setup

4.2.1 70% of the observer speed

Even at this low speed, the *Shadow centering* strategy performs better when compared with the *Level I*, and even a few survivors remain undiscovered till the end of the observer path. The *Max shadow greedy* strategy performs a bit better than *Simple greedy* strategy but the curves are quite similar as we can observer in the figure 4.12.

As we move ahead to higher survivor speeds we shall see that the *Shadow centering* strategy keeps the lead among the heuristic strategies. This was expected as it was designed for such game levels in which the change in shadow regions is gradual which gives ample

time for the survivors to move to a new local central regions of the shadows. While both the *greedy* strategies cannot make the survivors move out and away from the concavity in the bottom right obstacle, the survivors in that region are bound to be discovered.

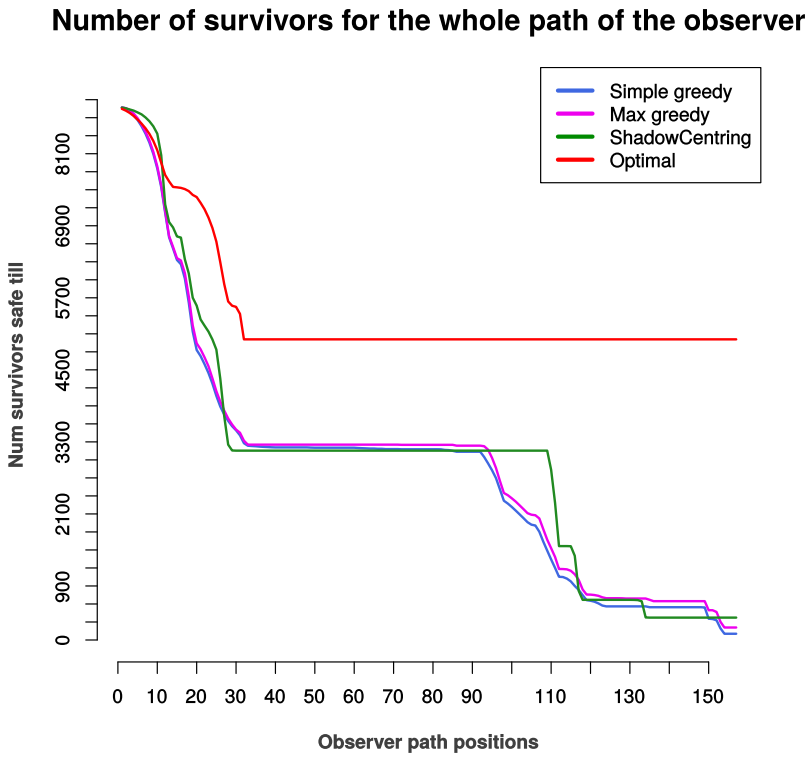


Figure 4.12 Number of survivors remaining till the observer path at 70% of the observer speed.

4.2.2 90% of the observer speed

At this speed the *Shadow centering* strategy shows a significant improvement, as seen in the figure 4.13. It only dips lower than the other heuristic strategies around $30th$ observer path position which is when the observer comes around the corner of bottom left obstacle and a huge area is visible to it and due to lower speeds, the survivors cannot move around and be behind that obstacle again. It could perform a lot better if the survivors could start from the central regions of the local shadow regions and they do not have to spend time to move towards the central shadow regions while the observer moves on its path.

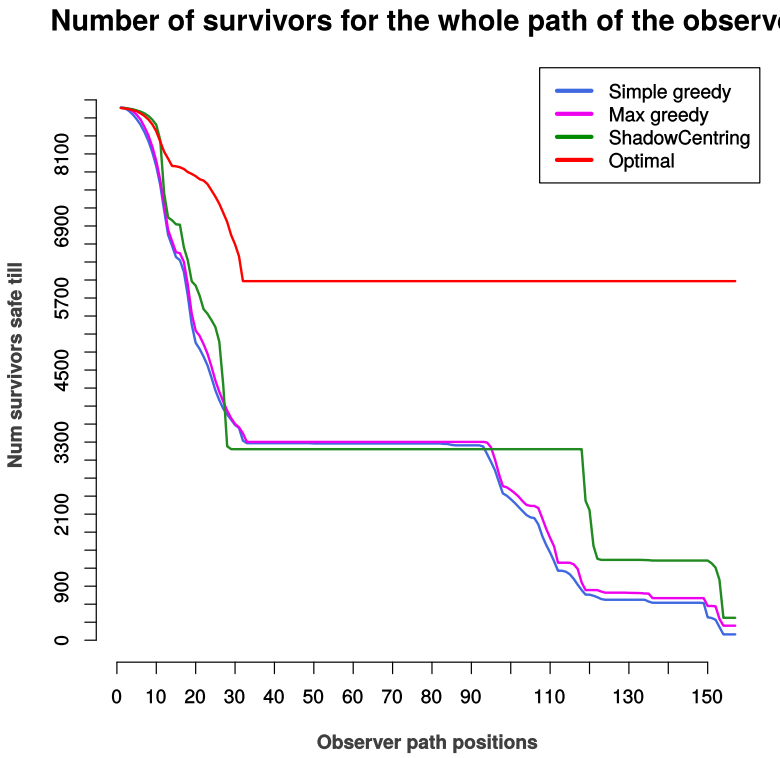


Figure 4.13 Number of survivors remaining till the observer path for 90% speed of the observer.

4.2.3 150% of the observer speed

At 1.5 times the speed of the observer, we can see, in the figure 4.14, the true potential of *Shadow centering* approach in this game level of gradually changing shadow regions. It performs a lot better than the *greedy* strategies as a significantly larger number of the survivors actually remain undiscovered till the end. However it is not at all close to what the optimal results are at such speeds.

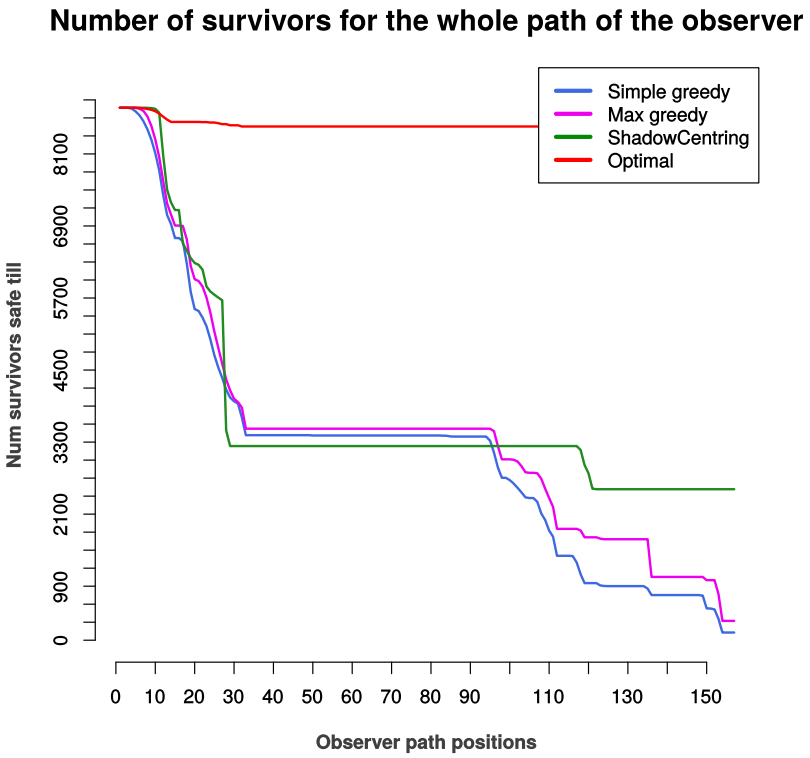


Figure 4.14 Number of survivors remaining till the observer path for 150% speed of the observer.

We discovered an odd pattern at these higher speeds of the survivor which we can see in 4.15. There seems to be inconsistency in the results as we do not see a smooth transition from less safe to more safe regions which is displayed as lighter to darker green regions. There seem to be safer positions in between unsafe regions and vice-versa. However this pattern does not show up in lower speeds.

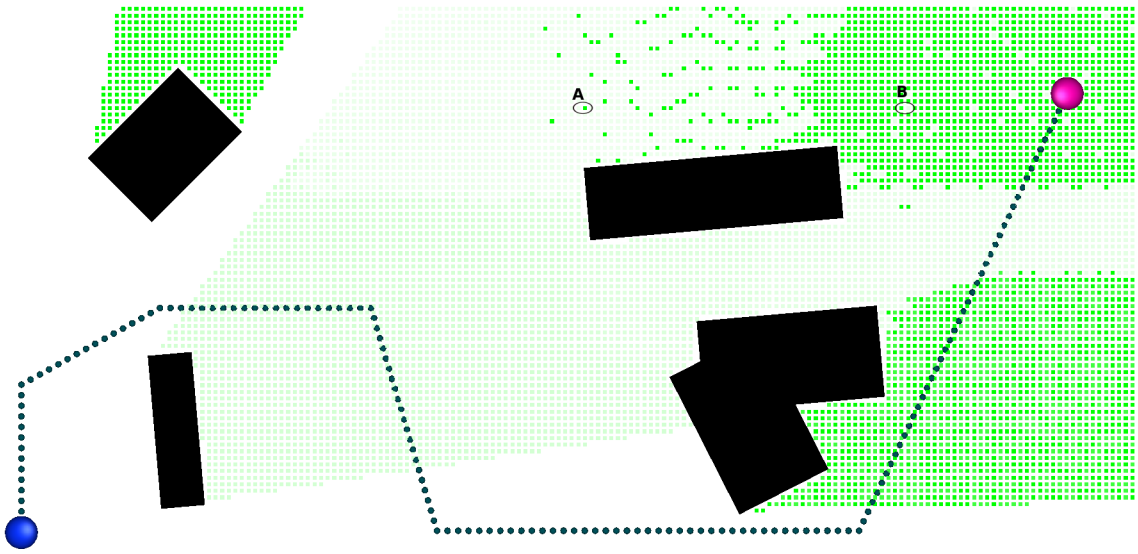


Figure 4.15 Shadow centering strategy applied to Level II at 150% of the observer speed. The two small enclosed regions **(A)** and **(B)** are two such areas where the neighboring survivor starting positions have significant difference in survival time as seen by the differences in shading of green positions.

We pause here to try to explain this odd phenomenon in the figure 4.15 by expanding on the decisions which were made for heuristically selecting the safe positions by each survivor at different points in time starting from the beginning of the observer path.

As described earlier for the *Shadow centering* approach, we use an 8 way approach i.e. there are 8 possible positions around the current position of the survivor, excluding the current position which is also valid, that it can move to. This approach provides limited choice as compared to the *greedy* strategies which has 360 positions to choose from. We could use a larger number of potential positions but we have purposely limited the choice for the reason of time constraints to perform each experiment, as all survivors are processed in *Shadow centering* approach, while only the survivors which are at a risk of being discovered shortly, move to new positions in both the *greedy* strategies. This means that the *Shadow centering* approach is capable of performing more accurately if more than 8 choices are provided for each survivor. The improvement through more accuracy is not significant at lower speeds or for the levels in which this strategy is not effective. So if this approach is the best one for a level or/and a path of the observer, a more exhaustive *Shadow*

centering approach, will merely emphasize this result. In this particular level, it is already the most effective heuristic strategy out of the three.

Lets look at the region ***A*** which encloses two starting positions in the figure 4.15, to analyze exactly what decisions are made by each survivor. We plot the complete survivor paths for both starting positions inside the enclosed region ***A***, one of which is very light green and the other one is a darker green shade, in the figure 4.16. In the figure 4.16, we see two screen shot of the game level, each representing the path taken by a survivor in the region ***A*** of the figure 4.15. The path is numbered starting from ***1***, and we can see the similar numbering on the observer path. The numbering is done to show the position of both the survivor and the observer at a particular instance in time. So for an instance in time which is represented by a number, say number ***4***, we can compare the positions of a survivor and the observer. The numbering ends when the survivor is discovered or the observer reaches the end of its path. Note that the numbered positions are not actually consecutive, but a fixed number of positions are skipped between each numbered position to avoid cluttering yet displaying a decent amount of path information which shows a general direction taken by the survivor.

In the figure 4.16 we see two screen shots each representing a survivor path, for two immediate neighboring survivor starting positions. For the upper screen shot in the figure 4.16, the survivor is discovered at the $5th$ numbered position and if we compare it to the lower screen shot, which is for the neighboring position, the survivor actually makes it till the end of the observer path without being discovered. If we see the $5th$ position at the lower screen shot, we can see the decision of the survivor to move towards the right, is the main reason it did not get discovered. Although in both screen shots the paths of the survivors look identical till the $4th$ numbered position, they actually are a little different from the beginning.

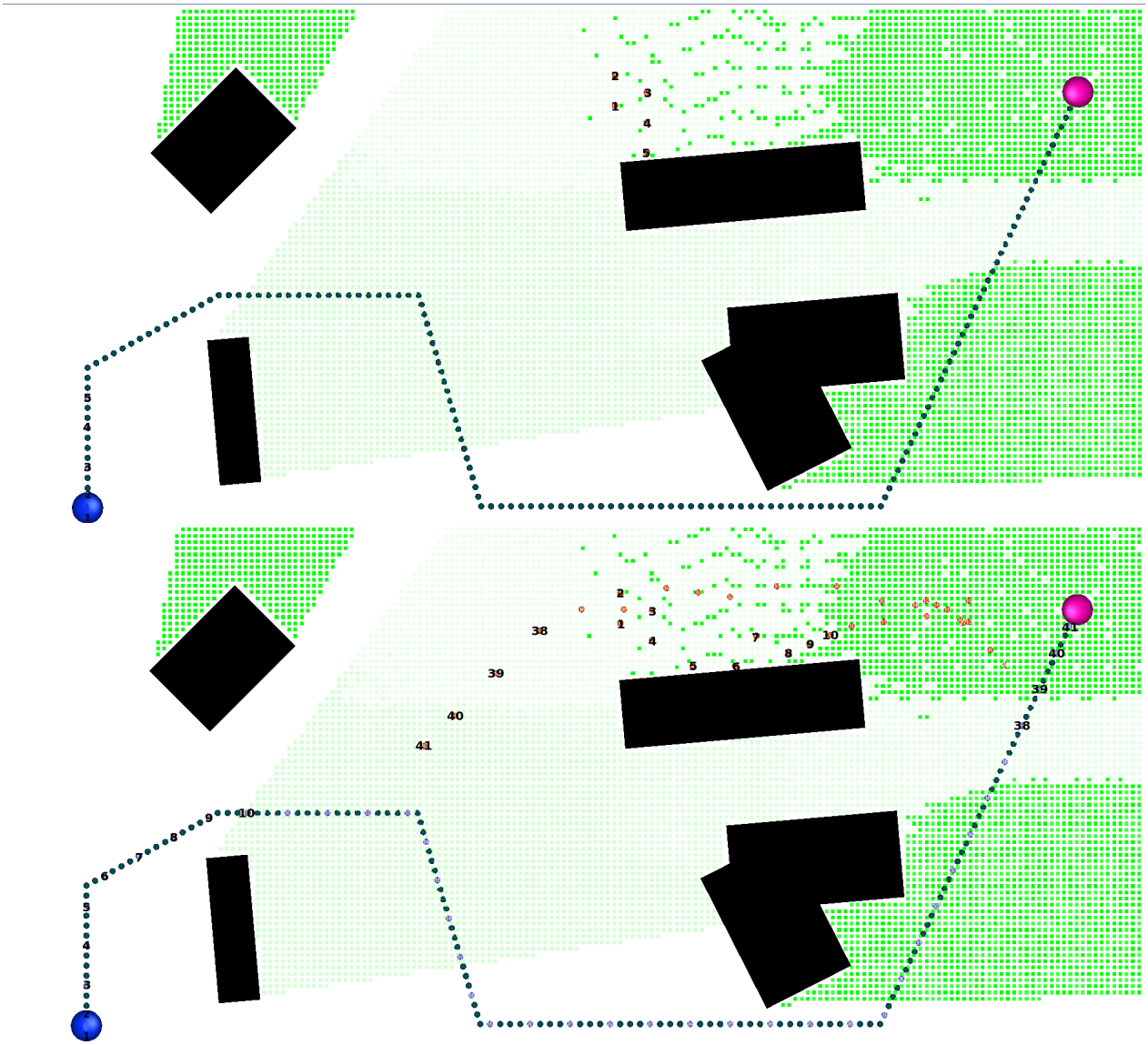


Figure 4.16 Path taken by two neighboring survivors in the enclosed region ***A*** of the figure 4.15

If we observe the screen shots in the figure 4.17, we can see a similar situation happening for the two neighboring positions in the enclosed area ***B*** of the figure 4.15. One survivor is caught at the $6th$ numbered position while its neighbor makes a better choice and successfully remains hidden till the end of the observer path.

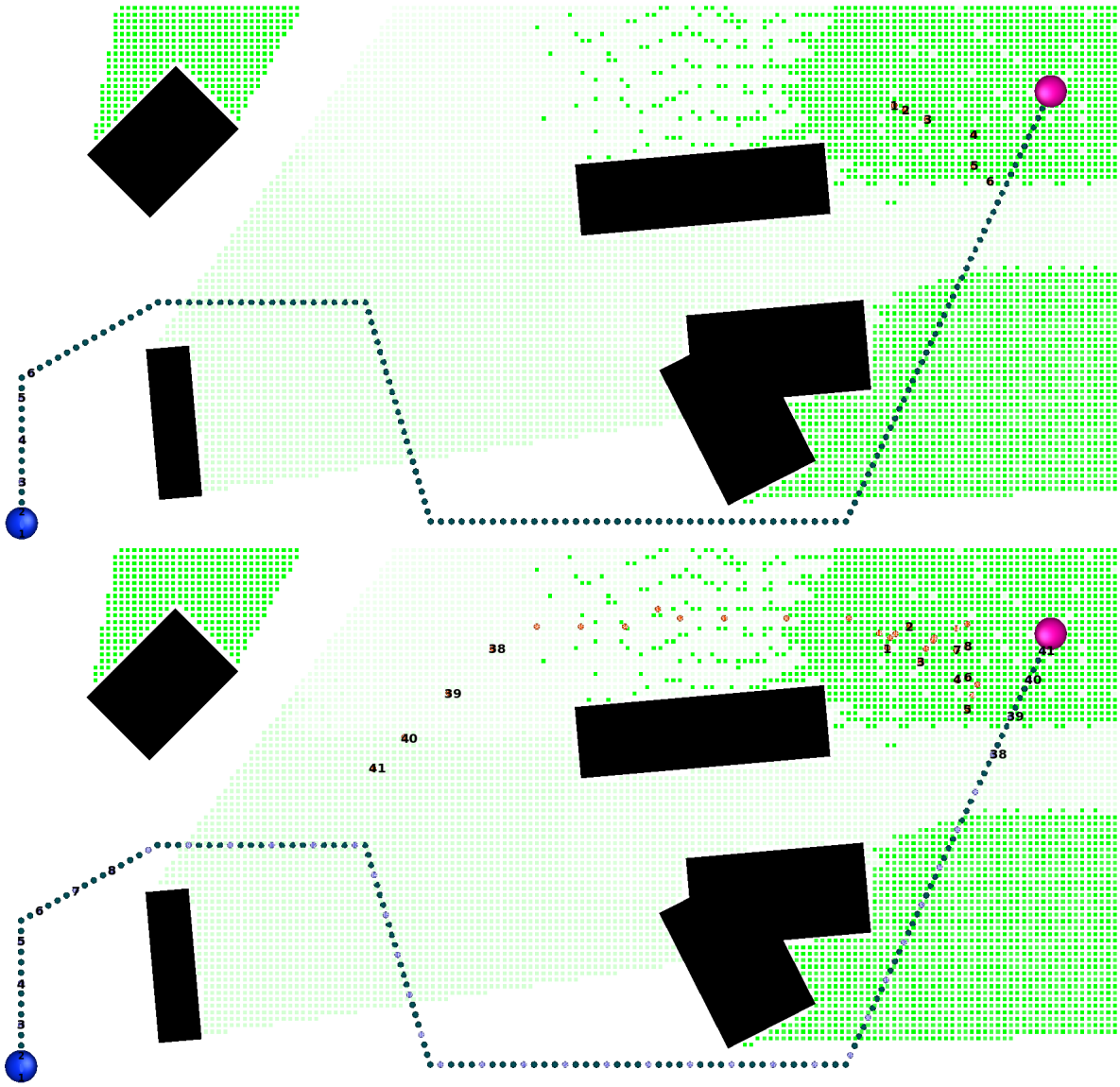


Figure 4.17 Path taken by two neighboring survivors in the enclosed region ***B*** of the figure 4.15

4.2.4 Collision Based

As seen for *Level I*, we perform this experiment for a reversed observer direction of movement. For *Level I* the observer path was symmetrical as the start and end positions were the same and there was only a single square obstacle present in the center of the level. For *Level II*, non symmetrical obstacles and observer path, means that there will be some

different starting parameters such as the number of survivors at the beginning or starting position of the observer path (which is the end position, for all other strategies), which can be seen when we compare the figures 4.14 and 4.18. We see a similar trend as *Level I*, as the two game levels are comparable. Both have open spaces for the survivors to move freely around the obstacles as they are detached from the boundary. The figure 4.18 clearly depicts that the plot is directly affected by the speed of the survivor only, as the curves for all three speeds are shifted vertically, but are identical. Many survivors are able to remain undiscovered, as there are more obstacles to hide behind.

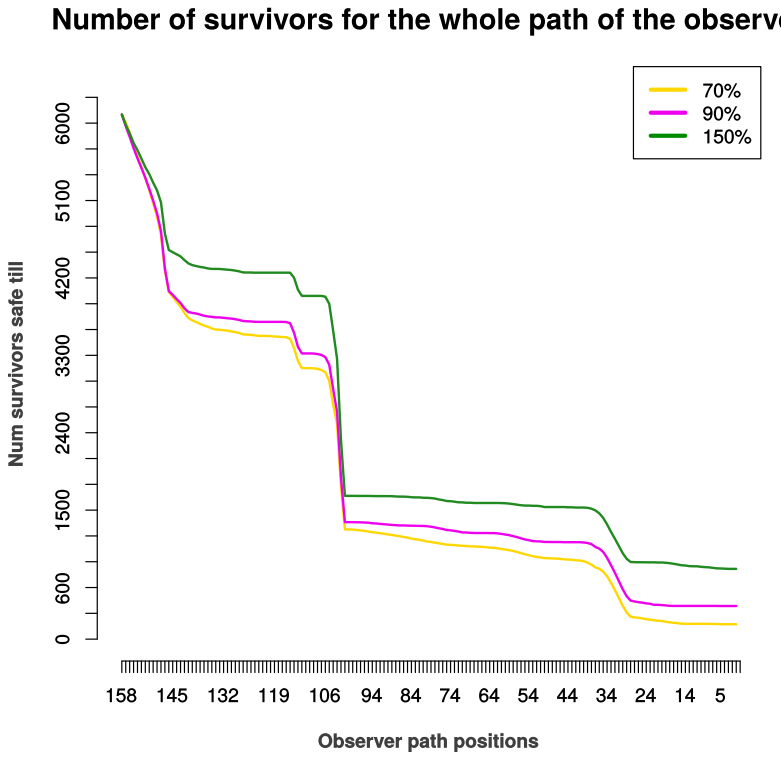


Figure 4.18 Number of survivors remaining till the observer path for collision based approach

4.3 Level III

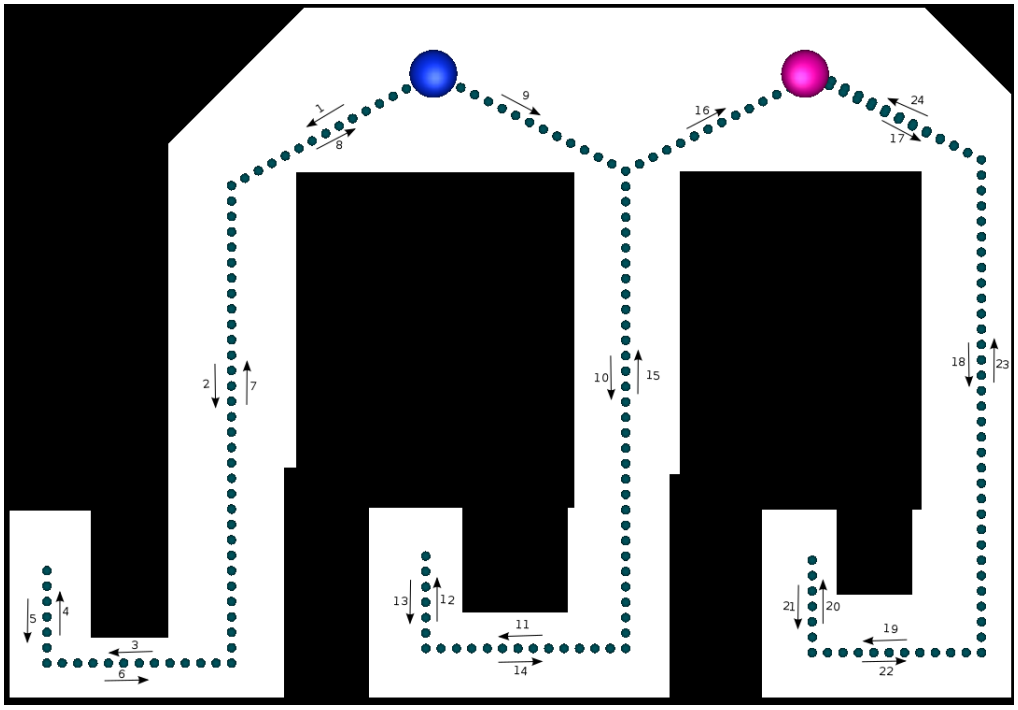


Figure 4.19 Level III setup

We will look into another theoretical level before moving onto the commercial game levels. The level in the figure 4.19 shows three alcove regions and the observer goes into each of these regions one by one. This level is inspired by a level from a paper from 2011 [6] (figure 7), which tries to answer the question of how many pursuers (in our case observers) are needed to find an evader (in our case survivor).

By comparing it to both the previous levels, we can see that the boundary acts as the obstacle itself, as there are no independent obstacles in this level. The spaces for actual movement of the survivors are very constricted. This level is intentionally designed to test how long it will take to find all the survivors, and if any can survive till the end of the observer path.

From what we have discussed till now about our heuristic strategies, you can probably deduce that none of these approaches would work very well in this level, no matter what

speed we use in the experiments, as they are not designed to foresee the observer direction and path of movement. However this level is a valuable test for our optimal strategy, and how we should approach designing more intelligent heuristic strategies for hiding.

We tested our strategies on this level for 3 survivor speeds, 75%, 110% and 155% of the observer speeds, as we can show clear distinction between certain results from our strategies.

4.3.1 75% of the observer speed

We can see in the figure 4.20, there are multiple sharp drops in survivors remaining at certain observer positions and each of these drops happen as the observer suddenly appears from around the corner and finds many survivors, as they have no space to move to, due to constricted movements and dead ends. All three of the heuristic strategies behave quite similarly. We would like to discuss the behavior of the optimal strategy through some screen shots as it shows some interesting results.

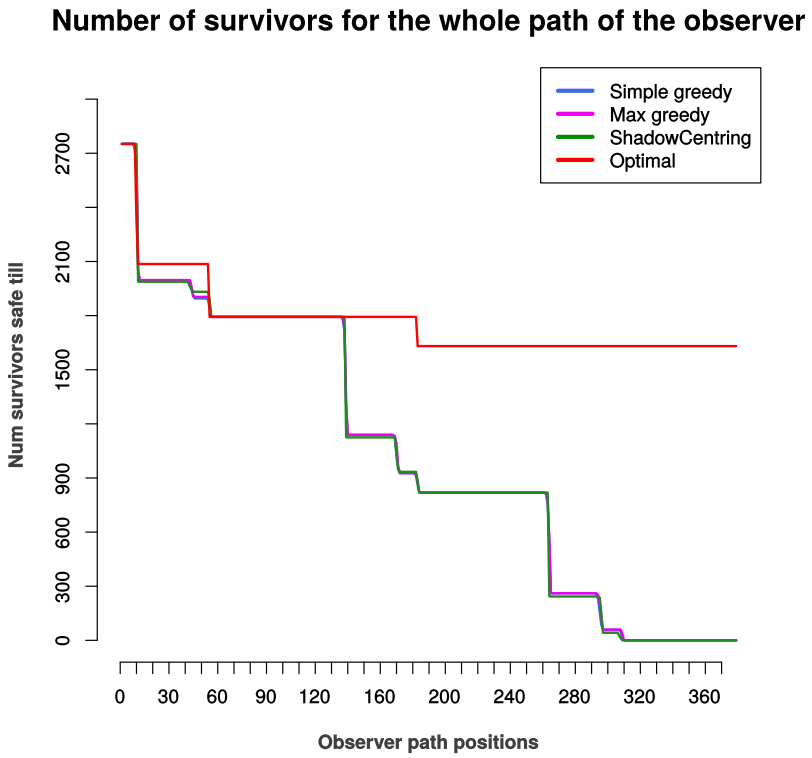


Figure 4.20 Number of survivors remaining till the observer path for 75% speed of the observer.

The figure 4.21 is a screen shot of the optimal results for the survivor speed equals to *75%* of the observer speed. The green spots gives us the information of how safe a survivor would be if it started from that position. The most green spots are 100% safe i.e. they survive for the whole observer path. The numbered positions are the specific positions of the observer and a survivor. The observer moves on its path, for which we show numbered positions with blue circles, while the survivor, which starts from the position marked as 1 on an orange circle, moves to always hide from the observer. This survivor is interesting as some of its neighbors are not able to survive the complete path of the observer. We will see how it barely makes it to a hiding position, at such low speeds. The numbered positions are not evenly spaced in time, but they are there to show where the observer and survivor were at the same exact time. The complete paths are not shown, to avoid confusion and cluttering.

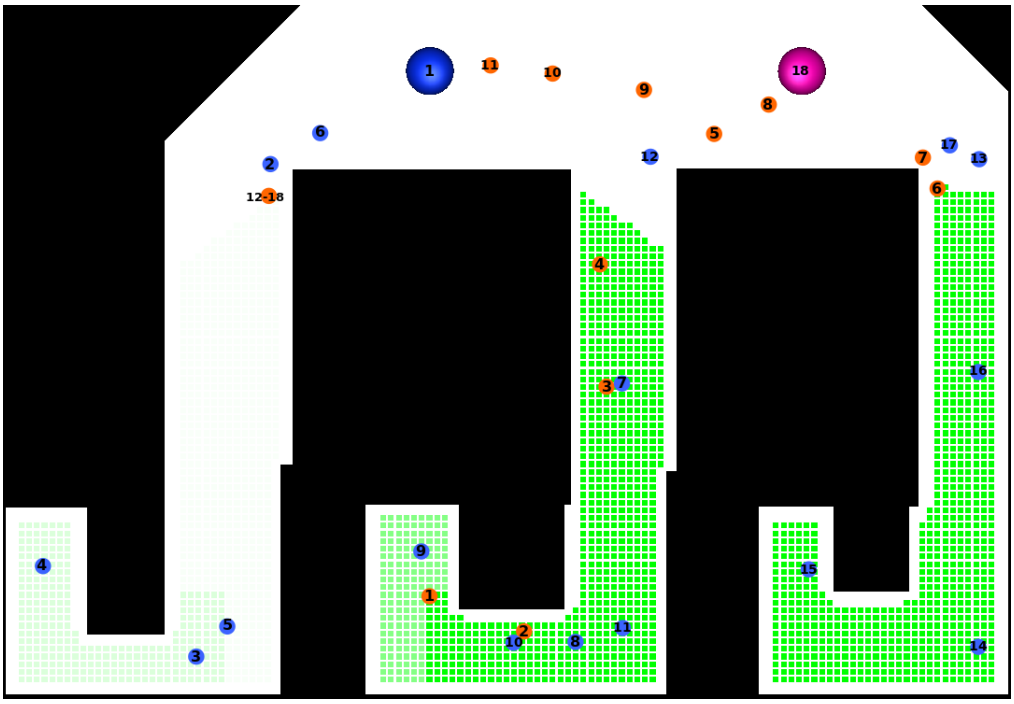


Figure 4.21 Number of survivors remaining till the observer path

We start for the numbered position 1 for both the survivor and the observer. The survivor is well hidden in the middle alcove region. The observer will first check the left alcove region and then move on to the middle one, and by that time, the survivor has to come out of it and hide somewhere else as to not be discovered. We can see from the from the numbered positions 1 to 5 the survivor has successfully come out of the middle alcove region. Notice that the distance covered by the observer is more than the survivor as the speed of the survivor is 75% of the speed of the observer. At position 6, the observer has come out of the left alcove region and is heading towards the middle one. The survivor was barely successful in hiding itself at this time slice in the right alcove region. The survivor hides till it is safe to come out when the observer has gone into the middle alcove, and we see how it makes the jump, from the numbered position 7 to 10, across the middle alcove region while the observer cannot see it. Now it only has to hide and wait till the observer completes its path. So the survivor hides in the left alcove region which is now safe, as we can see from the numbered positions 12 to 18.

This is not the only path the survivor can take to be safe, but it provides a general idea of

the crucial decisions for path selection. There are three important sub-paths of the survivor which are crucial for its survival for the complete observer path. These include the range of positions from 5 to 6, 8 to 10 and 11 to 12.

4.3.2 110% of the observer speed

At this speed, as we can see from the figure 4.22, there is not much change for the heuristic strategies as we did not expect much from them, but the optimal strategy is able to save all the survivors in the middle alcove this time. This is why we do not see a dip in the number of survivors remaining at around $190th$ observer path position, which we saw in the figure 4.20. Moreover, the number of survivors in the left alcove region are able to survive for longer as they are able to go deeper in the local region due to higher speeds.

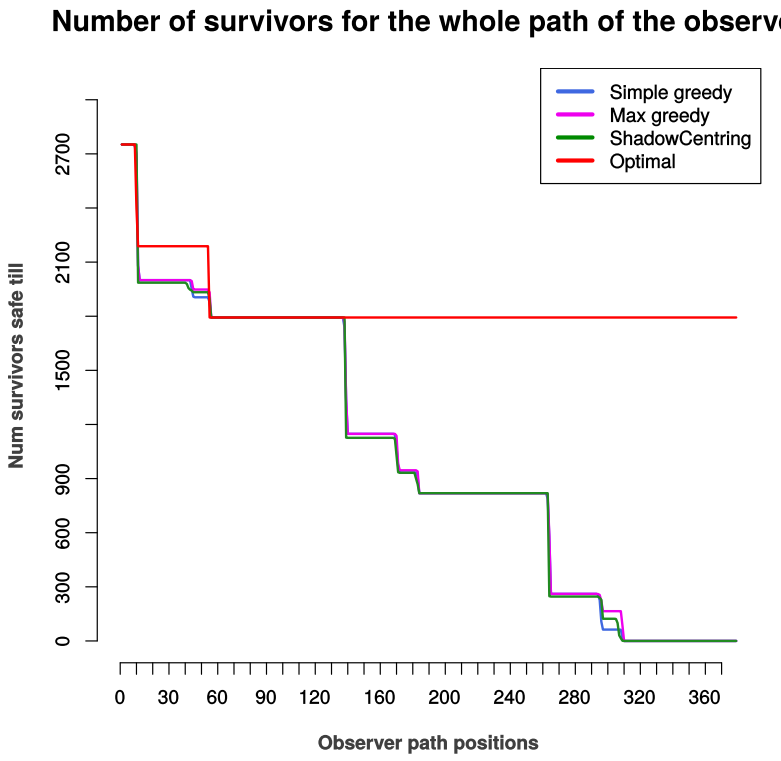


Figure 4.22 Number of survivors remaining till the observer path for 110% speed of the observer.

4.3.3 155% of the observer speed

We see a similar change for optimal strategy as they previous speed. The number of survivors in the left alcove region are able to survive for longer than 110% speed of the observer. The *Max shadow greedy* approach seems to do a little better which is directly due to increased speeds only. The heuristic strategies do not have enough space and freedom of movement to work properly. Now we can look at how the setup of the level affects our collision based strategy as there is not enough space for free movement of survivors.

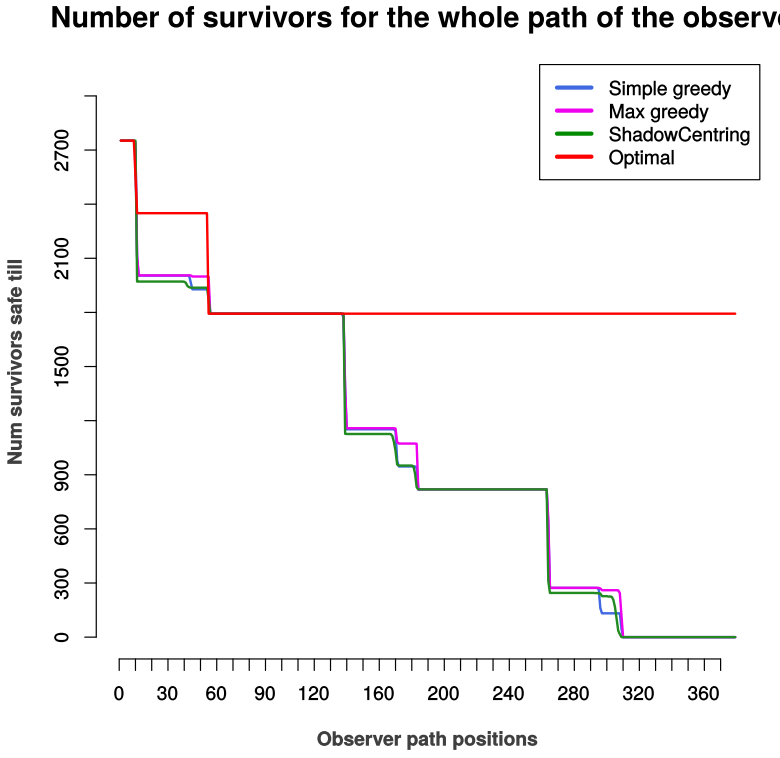


Figure 4.23 Number of survivors remaining till the observer path for 155% speed of the observer.

4.3.4 Collision Based

The affect of increasing speed of the survivors for such a level for normal heuristic strategies is barely visible. However for the collision based strategy, the increase in speed has

a negative affect on the survivability of the survivor NPCs. This can be seen in the figure 4.24. There are only 2 options for the survivors trapped in the alcoves, remain at current positions and wait for them to be discovered, or move towards the open area at the top when the observer is inside one of the alcoves. For higher speeds of the survivor, the probability of moving to this open area increases, as they can move to this open area faster and have the incentive of lesser conflicts with other survivors. For lower speeds, the survivors cannot move to this open area fast enough, so more of the survivors remain inside the alcove regions which are yet to be discovered by the observer. The dip at the beginning in the figure 4.24 is understandable as many survivors are discovered when the observer goes into the first alcove region, which in this case is the rightmost region. The interesting drop comes at around $120th$ observer position when the observer comes out of the first alcove region and the survivors who are in the open space at the top are discovered. This explains why more survivors having higher speeds are discovered, than the ones having lower speeds as fewer of them are able to reach the open space at the top. Another interesting point to note is that the observer retraces its step consistently and visits the open area at the top routinely. Needless to say, none of the survivors can remain undetected because of the constricted game level design.

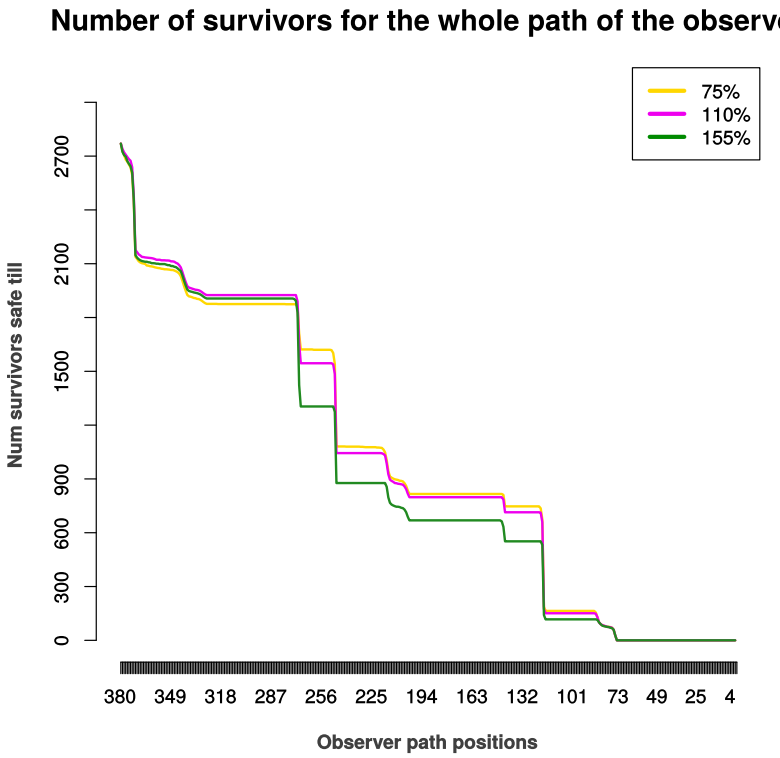


Figure 4.24 Number of survivors remaining till the observer path for collision based approach

4.4 Level IV

We will continue our experiments on game levels inspired from some real games. The one shown in figure 4.25 is called Cargo Dock from Metal Gear Solid, which is an action-adventure stealth video game. The player has to perform certain actions and reach a goal position while not being discovered. The player in the real game is replaced by an automated observer for our purposes, and while the player in the real game has to remain undiscovered by other NPCs, the observer's purpose is to explore the level without being stealthy. The NPCs in the game are substituted by automated survivors who have to hide from the observer. The roles of characters are reversed for our purposes but this works as a fine level for experimentation to study stealth capabilities of NPCs. The observer path shown here is a random exploratory path based on the goals for the real game level. This

level seems to be combination of features we have seen in previous levels such as independent obstacles, alcove regions and constrained spaces for movement.

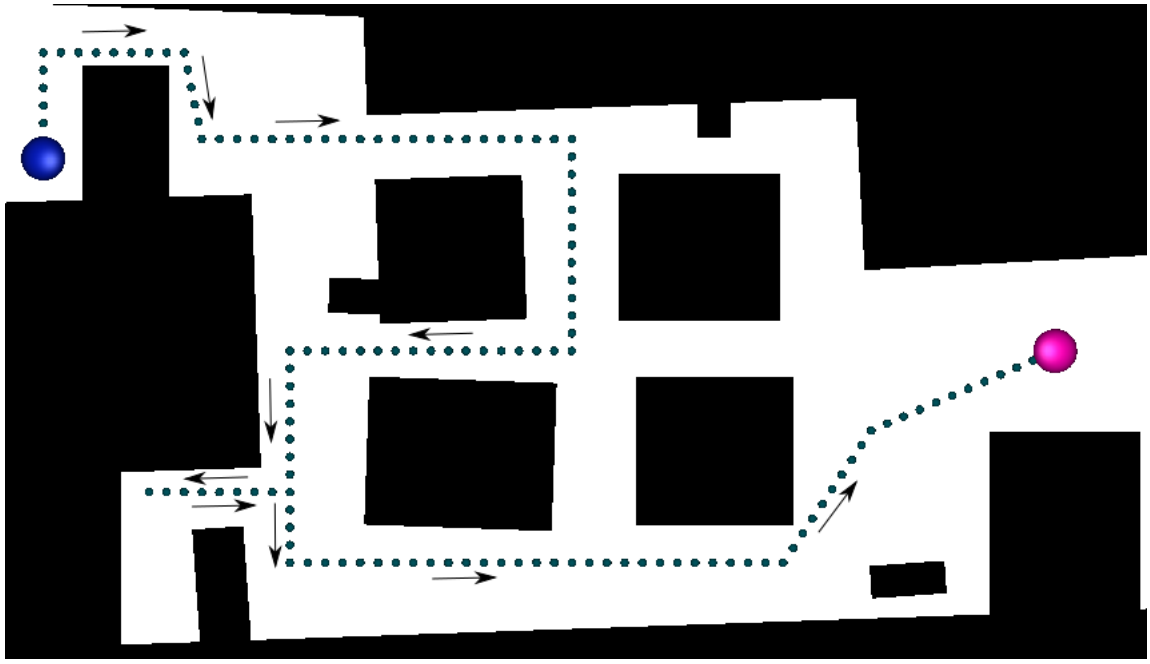


Figure 4.25 Level IV setup

4.4.1 90%, 125% and 200% of the observer speed

We performed experiments with three speeds for the survivor and now we will compare the results obtained from these tests. We combined the results of the three survivor speeds, which can be seen in the figure 4.26, as the plots are similar and we do not see any unusual behavior.

The observer starts from an alcove region itself, from where most of the game level is not visible to it, which gives some time to the survivors to get an early start and begin moving towards safer regions while the observer comes out of the alcove region. The optimal strategy benefits from it the most and performs quite well as this level is full of obstacles to hide behind and wait. However the heuristic strategies do not perform well, which was expected as the observer path contains sharp turns around corners of obstacles. There are few positions available where a static survivor will survive too, for this observer path. The

Max shadow greedy approach generally performs better than the other two heuristic strategies but is no where near how optimal strategy performs. The *Shadow centering* strategy performs much worse as most of the survivors congregate in the center of local shadow regions and are discovered when the observer comes around a corner. The survivors who are much safer in their original positions are moved towards the central regions of shadows, but instead of making them more safe, the survivors are discovered by the observer, hence the huge drops in number of survivors for this strategy.

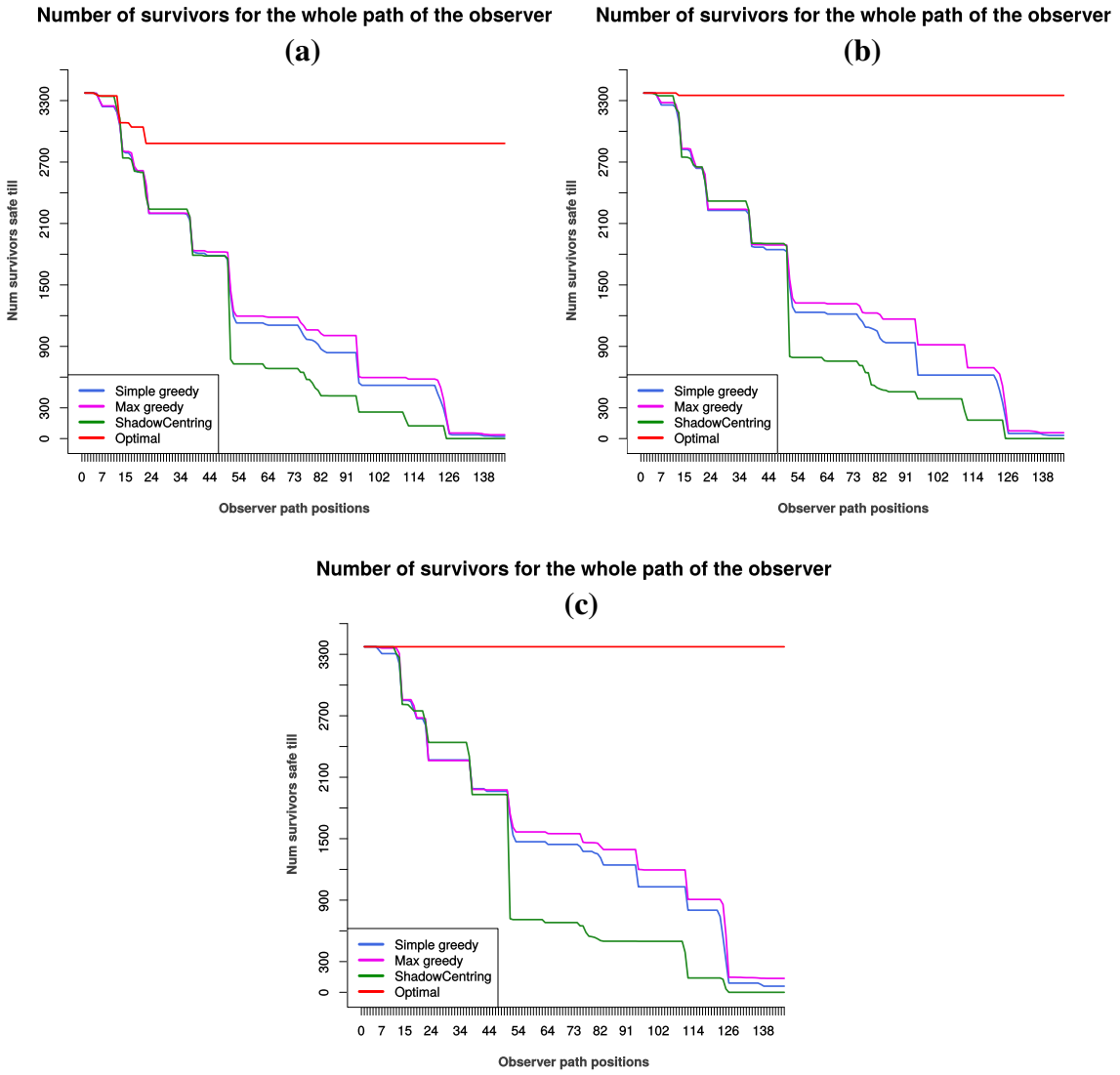


Figure 4.26 Number of survivors remaining till the observer path for **(a)** 90% **(b)** 125% and **(c)** 200% speed of the observer.

4.4.2 Collision Based

The heuristic strategies are not effective in this level, even though those require some informed decisions to be made for the survivor movements. We will see if the *Collision based* strategy might work differently for such a level. The figure 4.27 shows the plots for all three

speeds for *Collision based* approach. The plots are similar, except increasing the speed has better chance of survival for the survivor NPCs. For the survivor speed of 200% the observer speed, 99 survivors are able to remain undetected. This means that 99 survivors have independent paths and are able to survive without colliding with one another.

Even though this approach is based on localized selection, the selection decisions made are still informative as they are made in reverse, starting from the goal position backwards. A normal random approach if performed, having conditions similar to the three heuristic strategies without collisions, performs much worse than the heuristic approaches, as no survivors are able to remain undetected, except for the static ones which hide in the areas which are never explored by the observer, who will survive even if we do not use any strategy at all. To give the reader some perspective, the *Max shadow greedy* strategy is able to save 136 survivors at 200% of the observer speed, however collision are not taken into account.

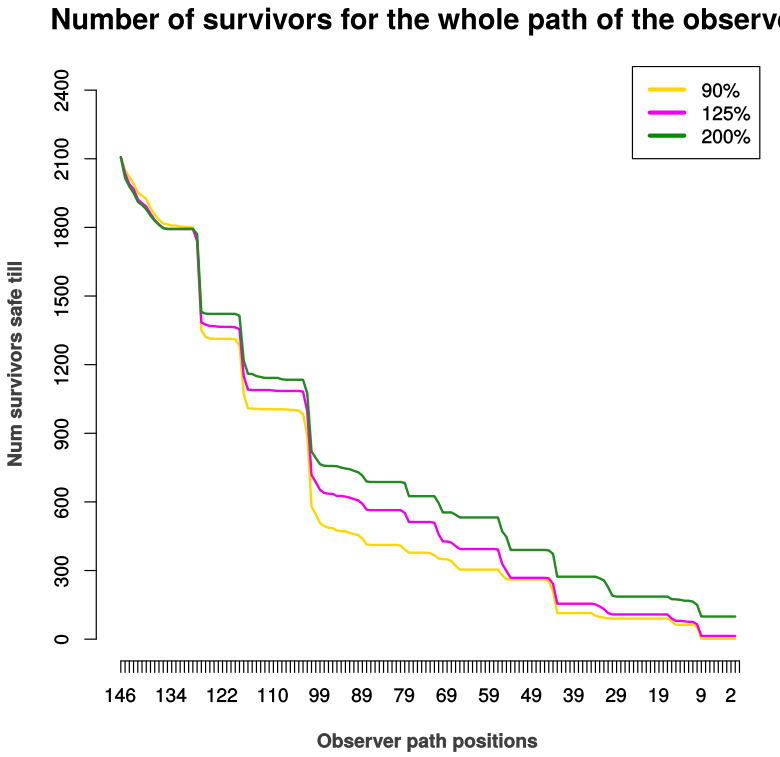


Figure 4.27 Number of survivors remaining till the observer path for collision based approach

4.5 Level V

Now we will examine a game level from *Wasteland 2* which is a turn-based and party-based role-playing game. It involves tactical combat and features a semi-overhead view with a rotatable camera. The game level we analyze now is inspired by the level called *Temple of Titan-the underground*. We examine two short observer paths in this level, to see if various strategies behave differently for the same level at different positions of the observer. For longer observer paths which explore most of the game level, there is sufficient time for the optimal strategy to outperform by a wide margin. Moreover, the longer observer paths are suitable for levels which have a destination to move towards such as stealth action games as seen in the previous level. For a role playing game level such as this, examining shorter paths is more relevant, as the game objective generally depend on exploration of certain area such as rooms, hallways and so forth. Using shorter observer paths, we can take a closer look at a specific area of the game level and get local information about that area.

4.5.1 Path I

This path as seen in the figure 4.28 explores some portion of the right side of the game level. The observer starts from the bottom right of the game level, the blue circle, and moves up and towards the left, the pink circle. The survivors that are in the left half of the game level do not need to move to be safe, so their behavior is not interesting. We performed experiments for three speeds for each of the paths which we will see now.

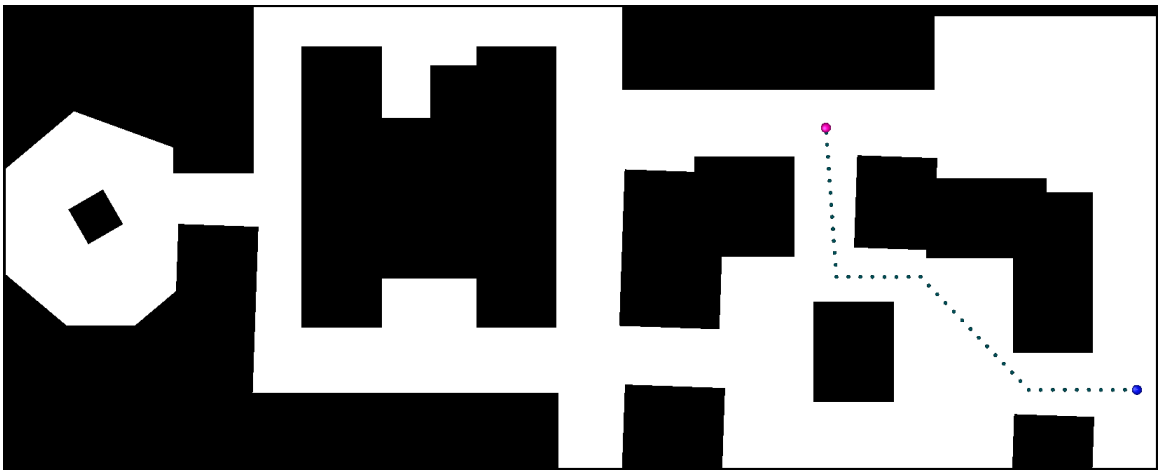


Figure 4.28 Level V setup for Path I

4.5.1.1 36%, 70% and 100% of the observer speed

The figure 4.29 shows the results for the three speeds used in the tests for the *Path I*. The performance of the three heuristic strategies does not have considerable differences. This is because the observer path is short and there is frequent change in the region which is being explored while the observer moves on its path. Compare it with the *Level III* results where the observer moves through narrow regions and the area being explored changes frequently. This level shares some structural elements to *Level III* and *Level IV*. The *Shadow centering* approach ends lower than the other strategies. This is because the end goal, as seen in the figure 4.28, lie in an open region just outside the narrow pathway. For this reason, a large portion of the survivors who have gathered at the local shadow regions are discovered when the observer comes out of the narrow pathway. For the *optimal* strategy, the initial big dip in the plots cannot be avoided as the observer moves into a region where survivors are hiding and they get cornered early on in the observer path. Another observation we can make is that the plots for heuristic strategies seems to be a lot closer to the optimal strategy and may give the impression that the heuristic strategies perform a lot better than the previous levels we tested. This phenomenon occurs because the observer path is short and do not explore the complete game level. So a significant number of survivors are safe regardless of the observers actions.

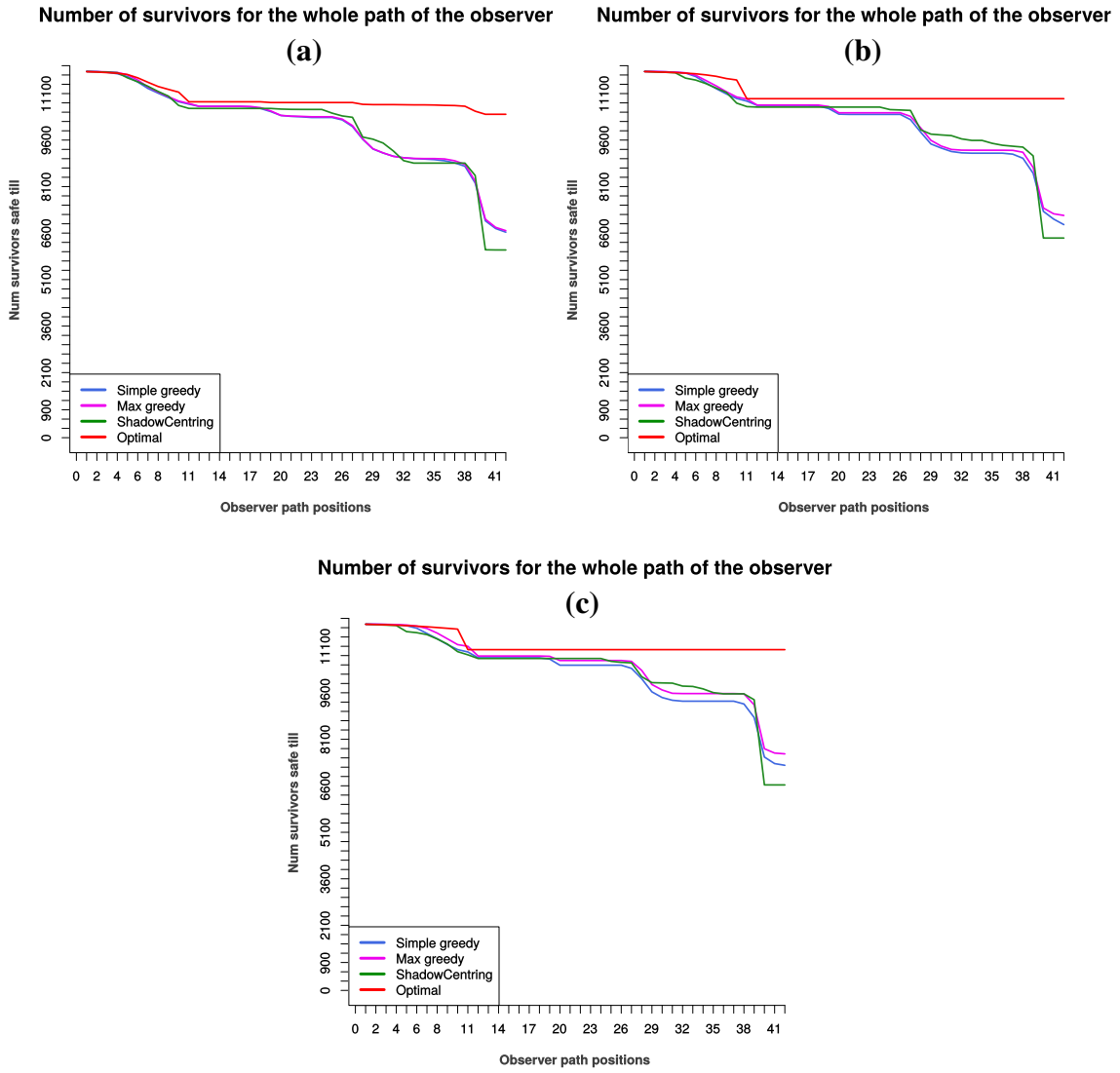


Figure 4.29 Number of survivors remaining for Path I, till the observer path points for **(a)** 36% **(b)** 70% and **(c)** 100% speed of the observer.

We will now take a look at some screen shots which provide some interesting information about certain regions for the *optimal* strategy at 36% of the observer speed. The experiments help in discern safety of some regions, which might not be apparent without testing. We see some interesting regions clearly as seen in the figure 4.30. The green positions are 100% safe while the yellow are less safe and the red regions are discovered the

earliest. We used the RGB range of colors rather than shades of green as it is not possible to distinguish the subtle shades of greens.

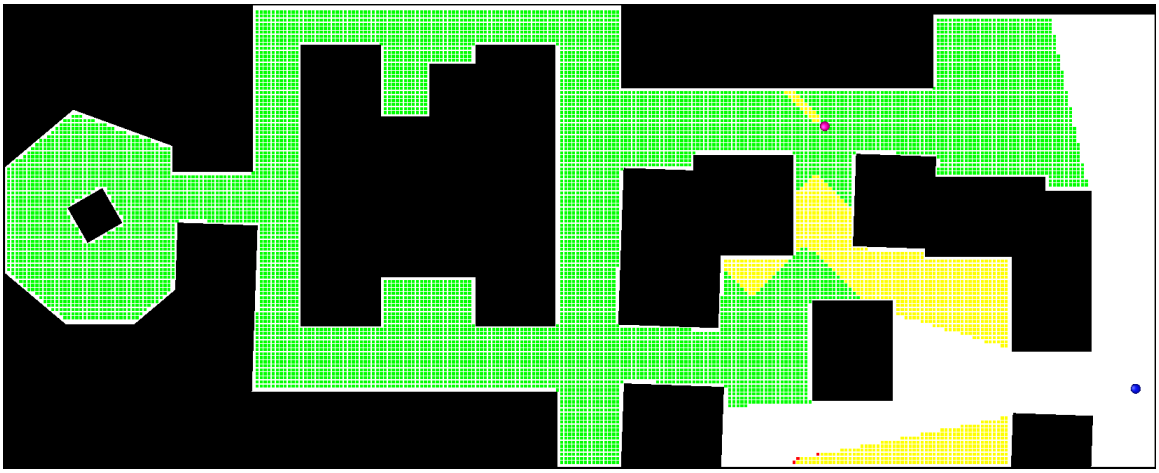


Figure 4.30 Screenshot of the survivability of survivor NPCs at the initial hiding positions for 36% of the observer speed.

In the figure 4.30, we see an odd region near the goal position of the observer. This narrow strip is yellow, which means the survivors starting from these positions are not 100% safe, while its neighbors are able to survive the complete trip of the observer. Such a region might not be evident without performing the experiments. We will examine 3 positions, one inside this unsafe region and two of its completely safe neighbors, to see what paths are chosen by each of the survivors. We are going to show one of the longest paths for each of the three positions considered, but there are multiple paths for each position.

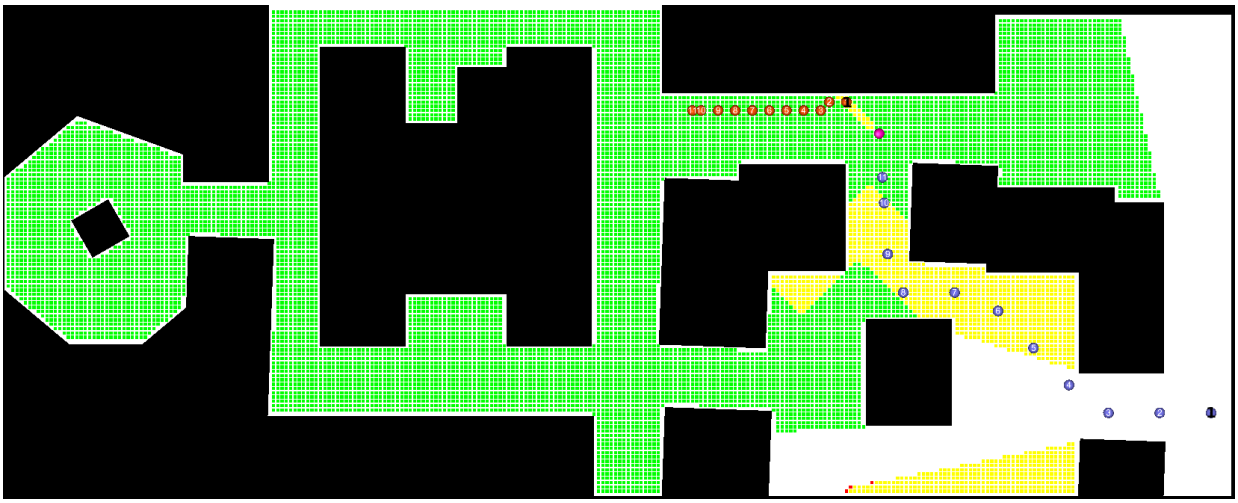


Figure 4.31 Screenshot of the path chosen by one of the survivor inside the unsafe yellow region

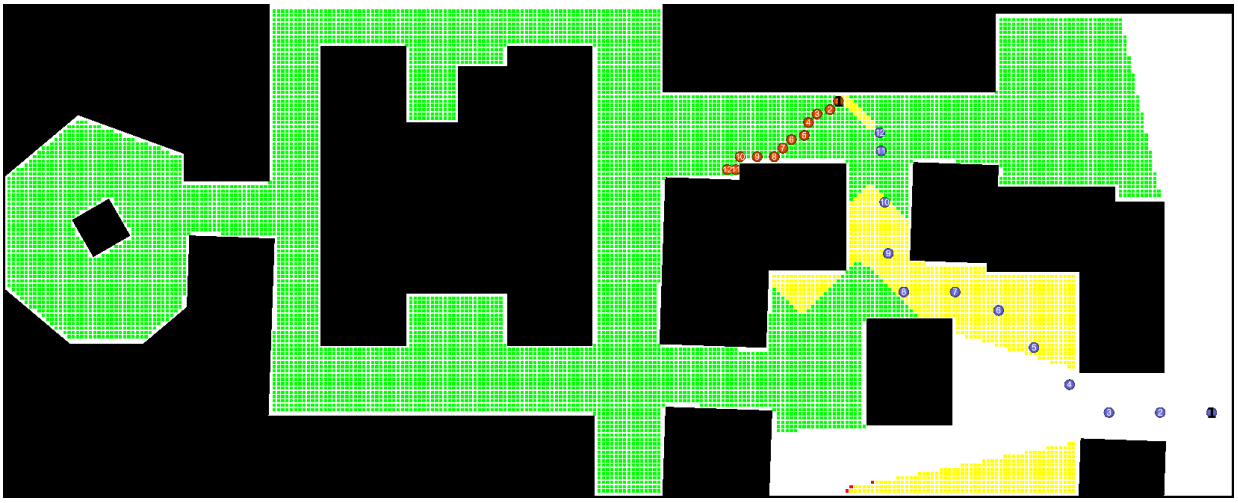


Figure 4.32 Screenshot of the path chosen by one of the survivor to the left of the unsafe yellow region

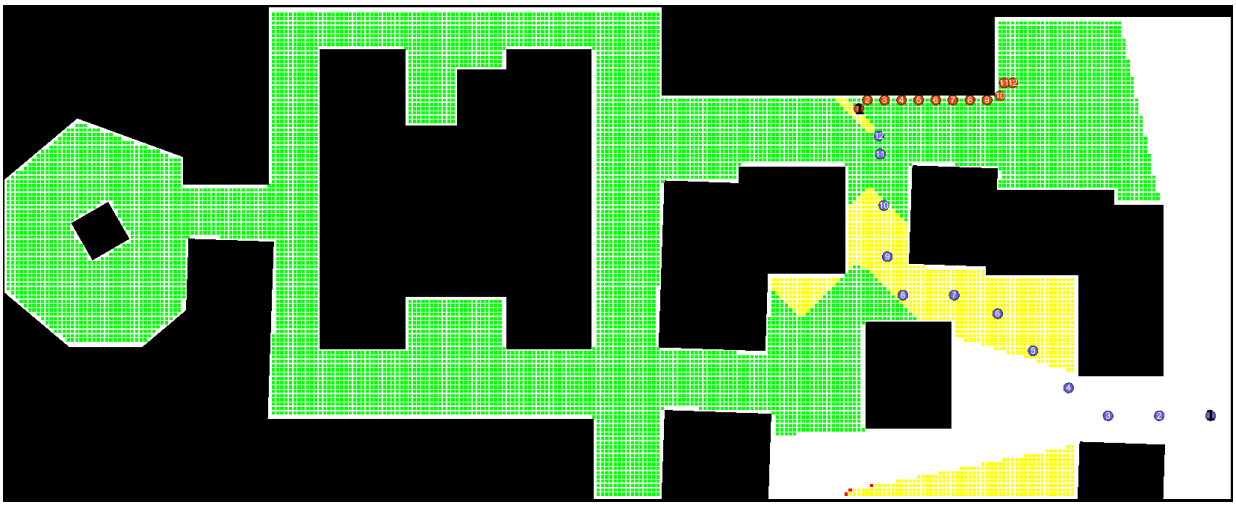


Figure 4.33 Screenshot of the path chosen by one of the survivor to the right of the unsafe yellow region

The figures 4.31, 4.32 and 4.33 shows the paths taken by three neighboring survivors, which are shown by orange spheres and the observer positions are represented by purple spheres. The first position is numbered to show the starting position of both the observer and the survivor. These results are at 36% of the observer speed and at such low speeds the survivors have limited mobility. The survivor in the figure 4.31 is discovered shortly before the observer reaches its goal position, while its neighbors are barely able to hide behind the obstacles and able to remain undetected, which we can see in the figures 4.32 and 4.33.

4.5.1.2 Collision Based

The figure 4.34 presents some interesting results. In the beginning lower speeds perform better than higher speeds. As in this approach the observer starts from its goal position and moves backwards, the approximate regions shown as orange ellipses in the figure 4.35 are the first ones to be explored. The survivors starting at these areas are the first to be discovered and there is no possibility of them escaping it as it would require higher speeds. The best solution for them is to remain in those general regions and wait to be exposed by the observer. The survivors having lower speeds cannot move far and tend to remain in these areas. This is the main reason the lower speed survivors perform a little better till all the survivors are discovered around $12th$ observer path position. For the remaining

observer path, higher speeds outperform as the remaining survivors are able to move further away from the visible region to the observer and are not trapped by the observer on one side and the obstacles on another.

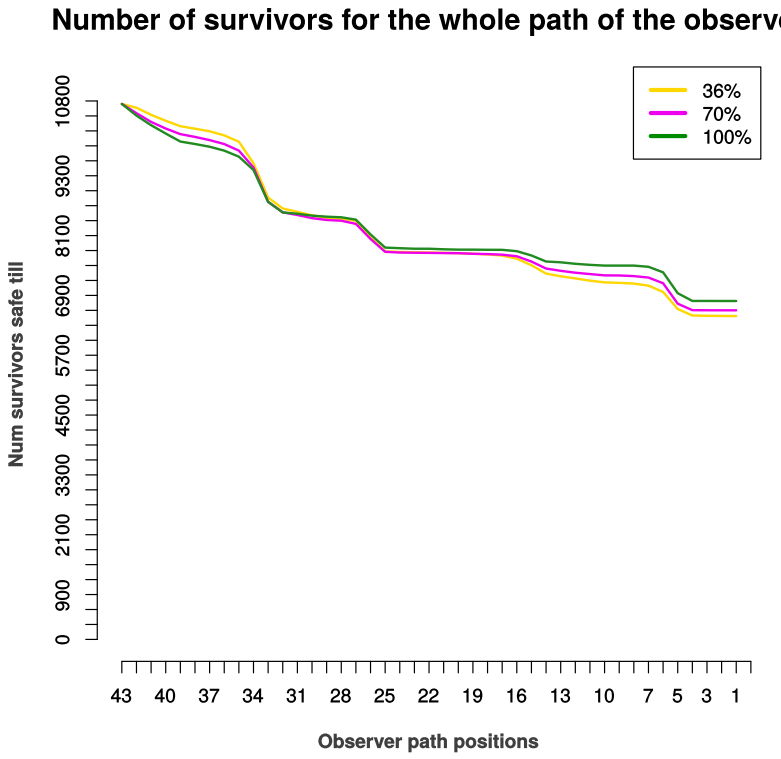


Figure 4.34 Number of survivors remaining till the observer path for collision based approach

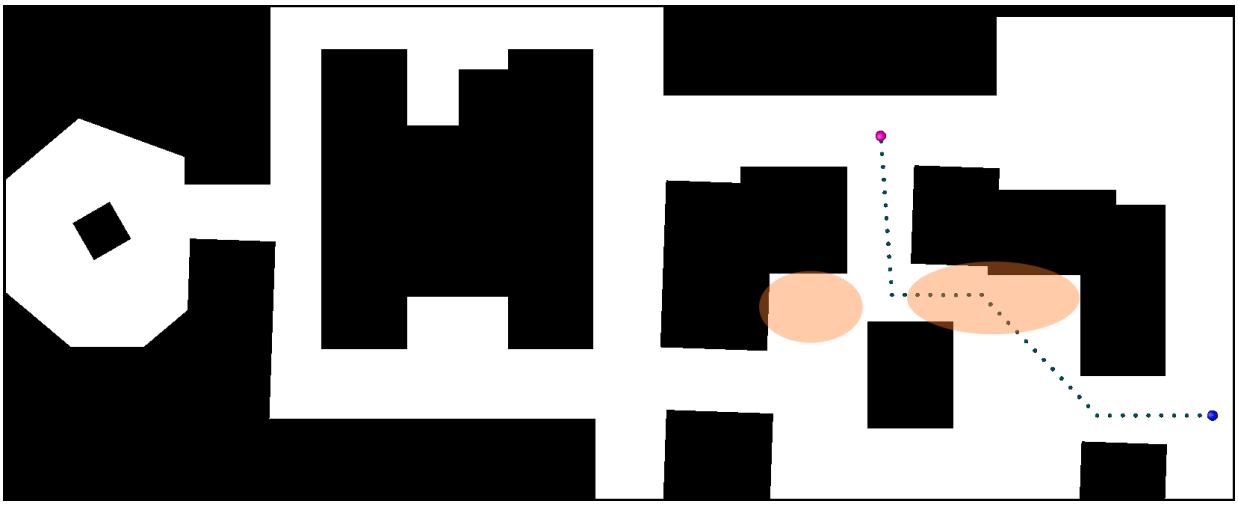


Figure 4.35 Level V setup for Path I showing high risk regions.

4.5.2 Path II

We look at the left half of the game level where we will explore narrow pathways, alcoves and chamber with independent obstacle. The path is shown in the figure 4.36. The observer starts from the blue circle and ends at the pink circle in the figure 4.36. We once again have results for three different speeds for this observer path which we will show next.

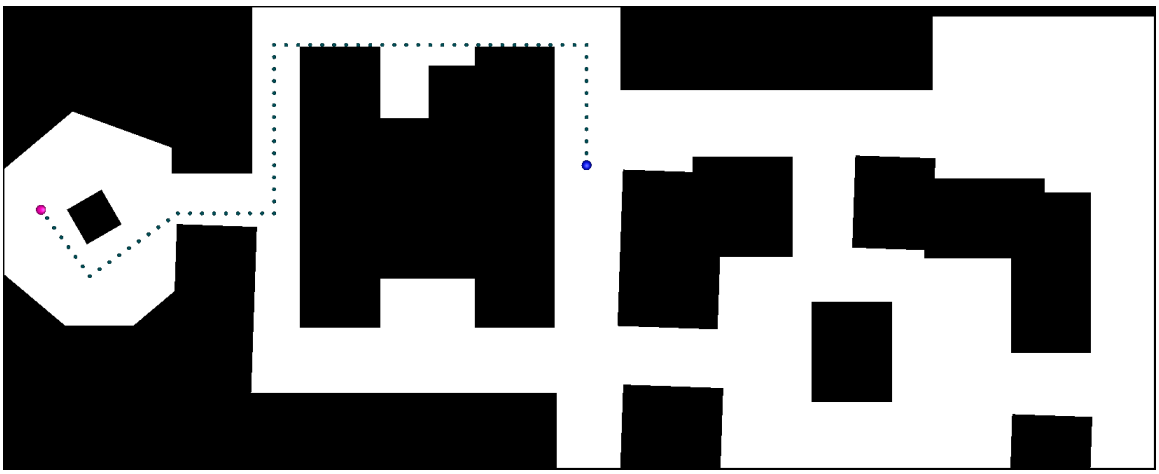


Figure 4.36 Level V setup for Path II

4.5.2.1 36%, 70% and 100% of the observer speed

For all three speeds, the behavior of the heuristic strategies seems similar, as seen in the figure 4.37. The *Shadow centering* strategy performs a little worse overall. We can see a similar trend for *Level IV* results which share structural similarities with this level such as narrow pathways, sharp corners and alcoves. The maximum speed for which we test is not enough for the optimal strategy to save all the survivors. The only dip in the survivors at 100% speed of the observer is when the survivors hiding in the alcove at the top are discovered around $24th$ observer path position.

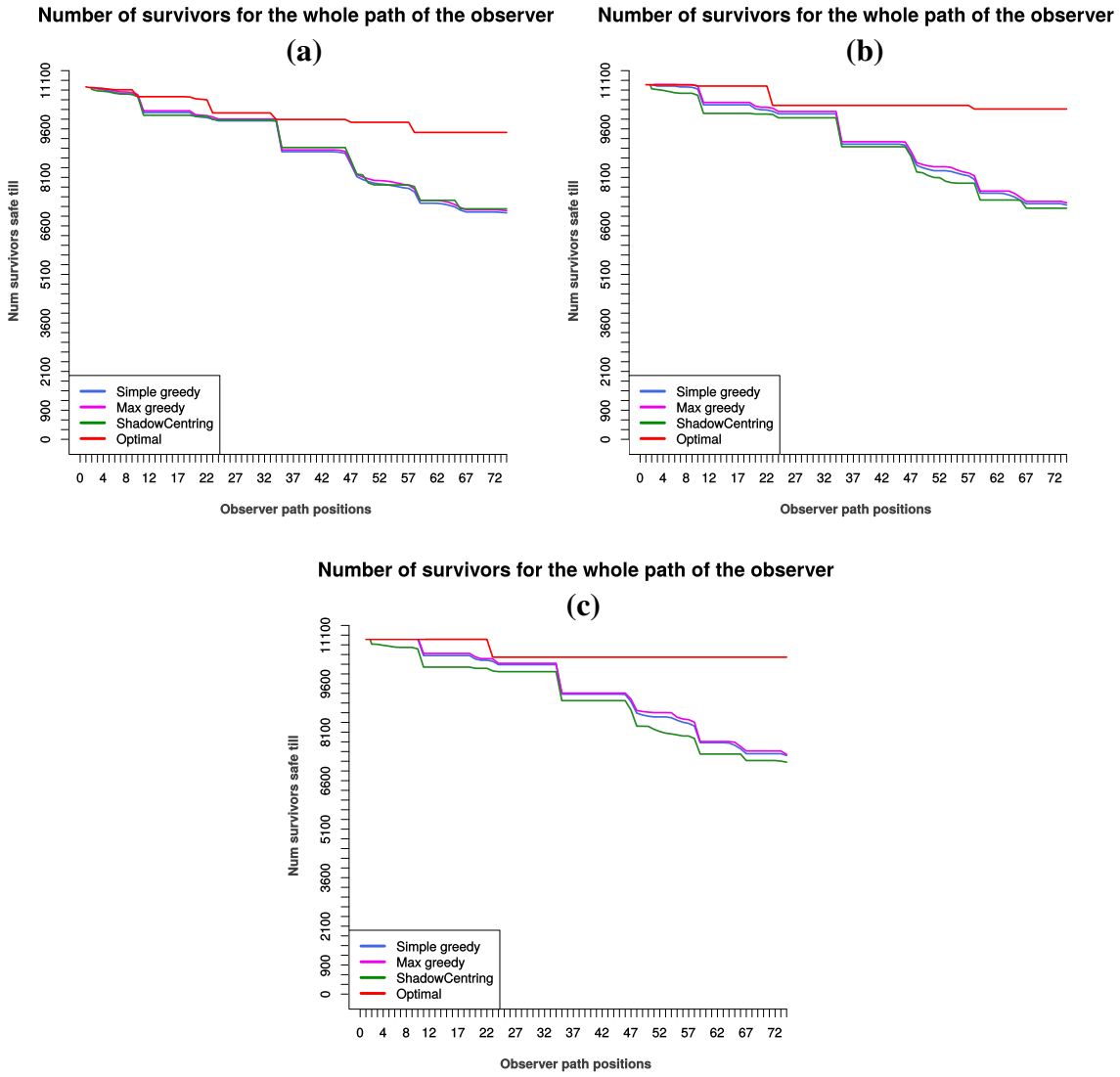


Figure 4.37 Number of survivors remaining for Path II, till the observer path points for **(a)** 36% **(b)** 70% and **(c)** 100% speed of the observer.

4.5.2.2 Collision Based

The observer path is such that the survivors are confined to narrow regions and without an intelligent approach, they are destined to be discovered. The observer starts from the end position which is inside a room with a single independent obstacle. The observer path is

such that it is able to see all around the obstacle in the first 15 positions of its path, from the goal position. In the figure 4.38 we can see that survivors with lesser speeds have an advantage in this strategy. In the beginning the room with the single square obstacle is the only space which is partially empty. Other spaces in the game level are filled with other survivors and obstacles. So the only place to move for the survivors is inside the room. The survivors with higher speeds have a greater chance of moving into the room when the observer starts and the shadow regions change. When the observer turns the first corner, more survivors having higher speeds are discovered than the ones with lower speeds. Since more survivors with higher speeds move inside the room and get discovered when the observer turns around a corner of the obstacle, we see the pattern in the figure 4.38. At the end the number of survivors remaining are the same which is expected from a constricted level such as this.

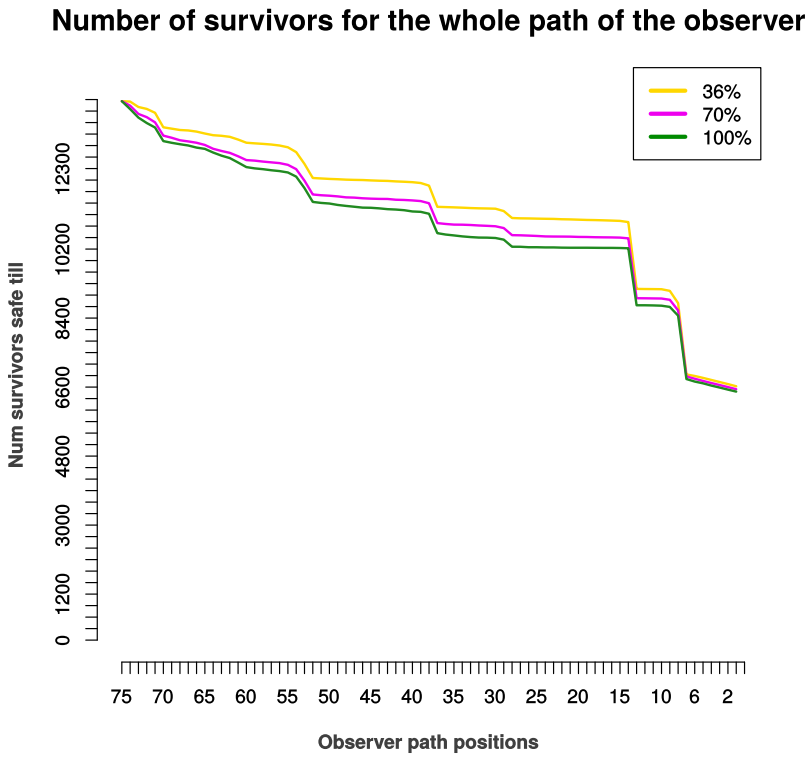


Figure 4.38 Number of survivors remaining till the observer path for collision based approach

4.6 Level VI

This level is inspired by the game level called *Crash* from *Call of Duty 4: Modern Warfare*. *Call of Duty 4: Modern Warfare* is a 2007 first-person shooter video game. *Crash* is a medium-sized multiplayer map and as in most multiplayer first person shooter maps, there are no defined paths for the players or NPCs. The players move for short distances at a time while taking cover behind obstacles. So we emphasize on a portion of the game level at a time through shorter observer paths. We modified the original level by vertically dissecting a portion on the right side of the game level for our experimentation. We exclude some part of the original level because, as seen in *Wasteland* experiments, analyzing the complete level for shorter observer paths results in agents in unexplored areas trivially surviving, and just inflates results by a large constant factor.

This level has a variety of obstacles such as independent obstacles from tiny ones to massive ones, alcoves and intrusive boundary. We experimented with multiple paths and three survivor speeds for all strategies and we will present the results next.

4.6.1 Path I

This observer path, as seen in the figure 4.39, starts from the top of the level at the blue circle and ends at bottom of the level at the pink circle. The observer discovers the open area in the middle and then moves to narrower regions containing alcoves and small obstacles.

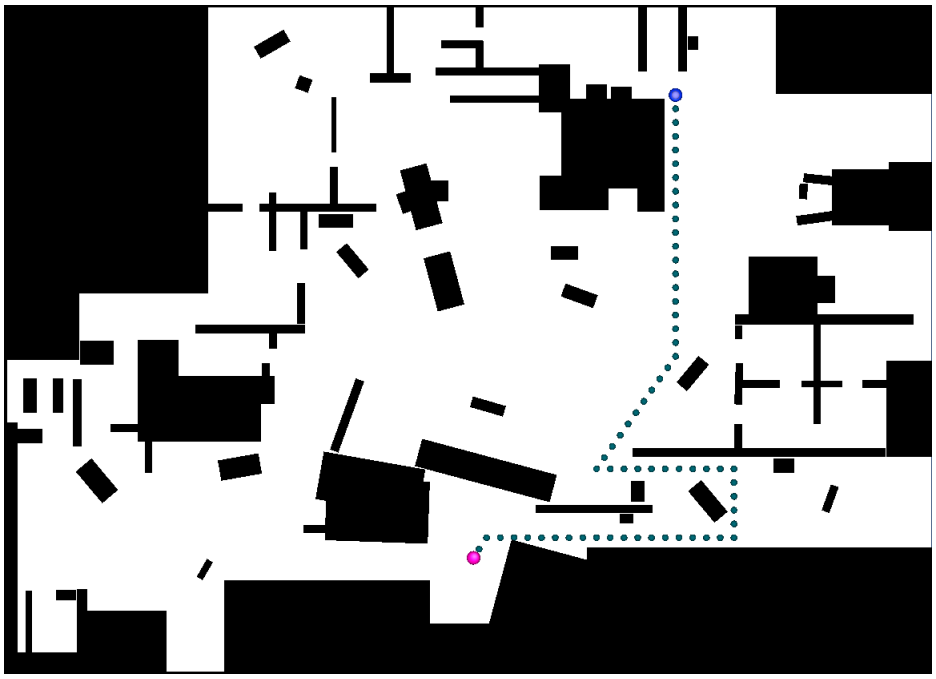


Figure 4.39 Level VI setup for Path I

4.6.1.1 60%, 100% and 120% of the observer speed

The greedy strategies perform better than the *Shadow centering* strategy even though there are many independent obstacles. The results from *Level II* demonstrated that the *Shadow centering* strategy performs better when the change in shadow regions is gradual in the presence of independent obstacles. In this case, however, the obstacles are small and the change in shadows is rapid. The shadows formed by the smaller obstacles are much more prone to an accelerated rate of change in shape and size. This is the reason that the *Shadow centering* approach could not perform well as we can see for all speeds for *Path I* in the figure 4.40. The *optimal* strategy performed much better as most of the survivors in the level are already hidden due to the number of obstacles and the scale of the game level.

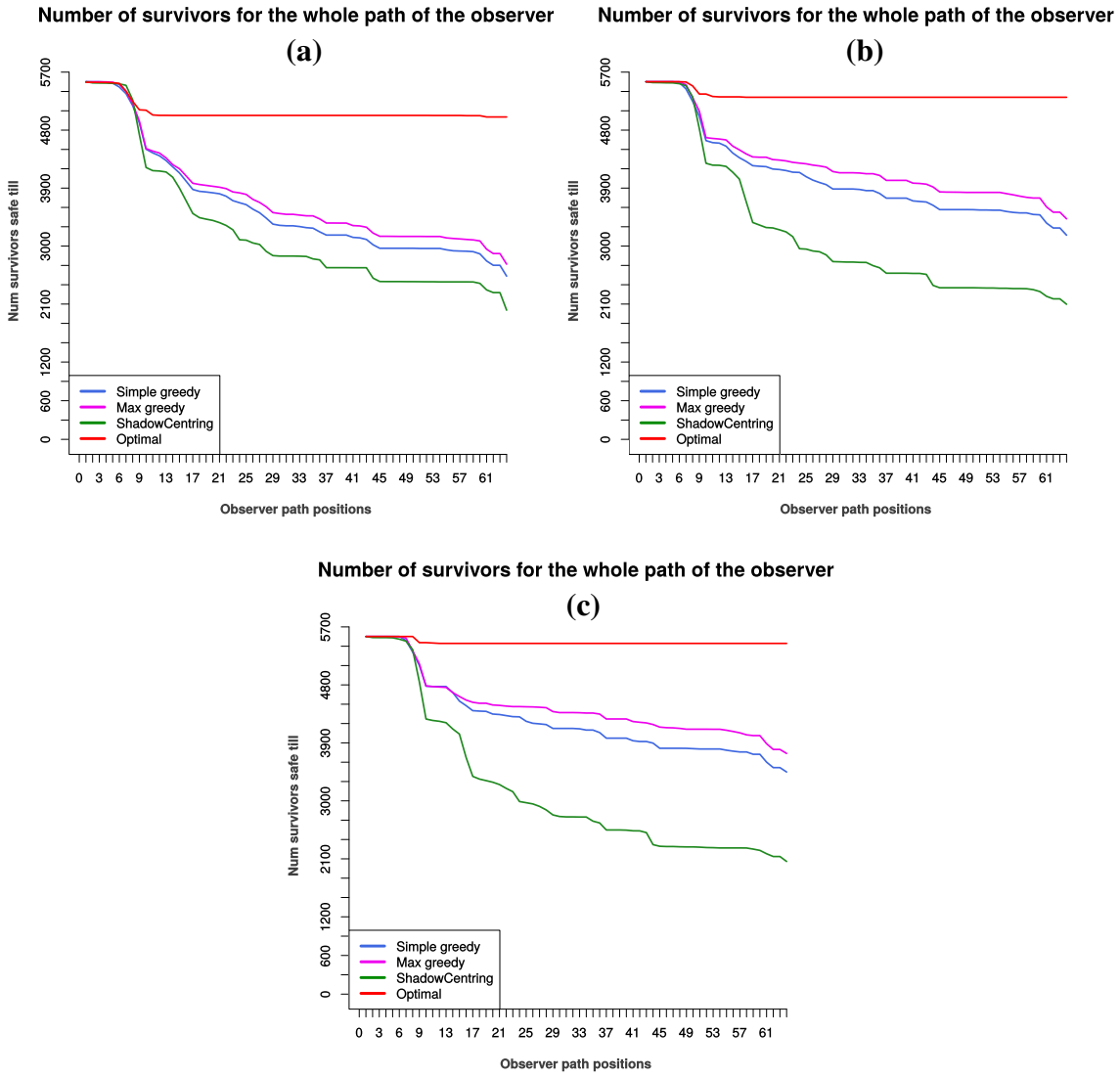


Figure 4.40 Number of survivors remaining for Path I, till the observer path points for **(a)** 60% **(b)** 100% and **(c)** 120% speed of the observer.

4.6.1.2 Collision Based

From the figure 4.41 we can see that until 32*nd* position of the observer path, the plot shows some interesting results. This is the position when the observer comes out of the narrow region into the open space in the middle of the game level. The observer discovers

a new region at the beginning of the observer path for *collision based* approach, as it turns a corner. The survivors at 100% and 120% of the observer speed have similar plots. The survivors having 60% of the observer speed seem to avoid getting detected better, as they are hidden behind the obstacles and do not have enough speed to move to a more open region. The first main drop for 60% speed comes around the $45th$ position of the observer when the observer discovers a region behind a small obstacle in the lower part of the level. All three plots become similar around $32nd$ position of the observer. We have seen such behavior for lower speeds in some previous levels such as *Level III*.

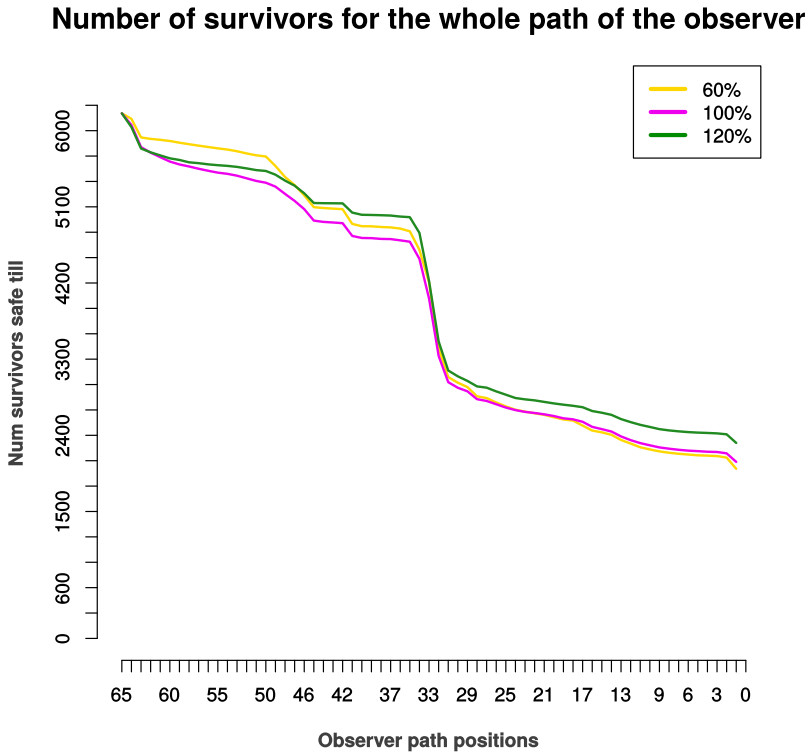


Figure 4.41 Number of survivors remaining till the observer path for collision based approach

4.6.2 Path II

We consider another short observer path, as seen in the figure 4.42, to examine some other interesting patterns in the plots. This observer path starts from the bottom and ends in the

middle of the game level. We experimented with all the strategies for three survivor speeds again.

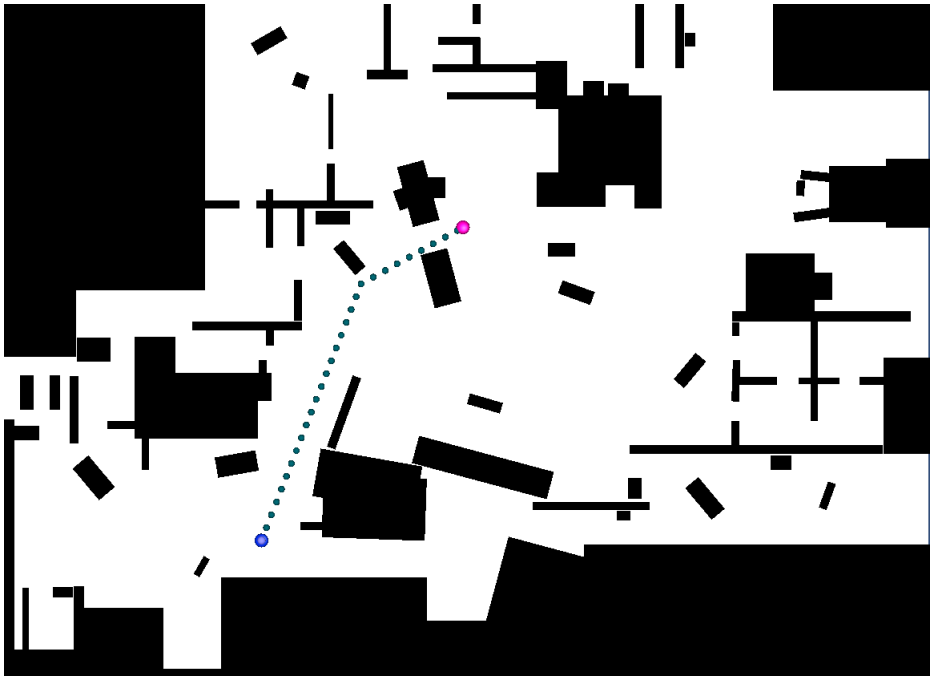


Figure 4.42 Level VI setup for Path II

4.6.2.1 60%, 100% and 120% of the observer speed

The figure 4.43 demonstrates a similar pattern as seen for *Path I*. The greedy strategies perform better than the *Shadow centering* approach. The number of survivors remain the same for all strategies in the beginning until the observer comes out of the narrow path into the open space in the middle.

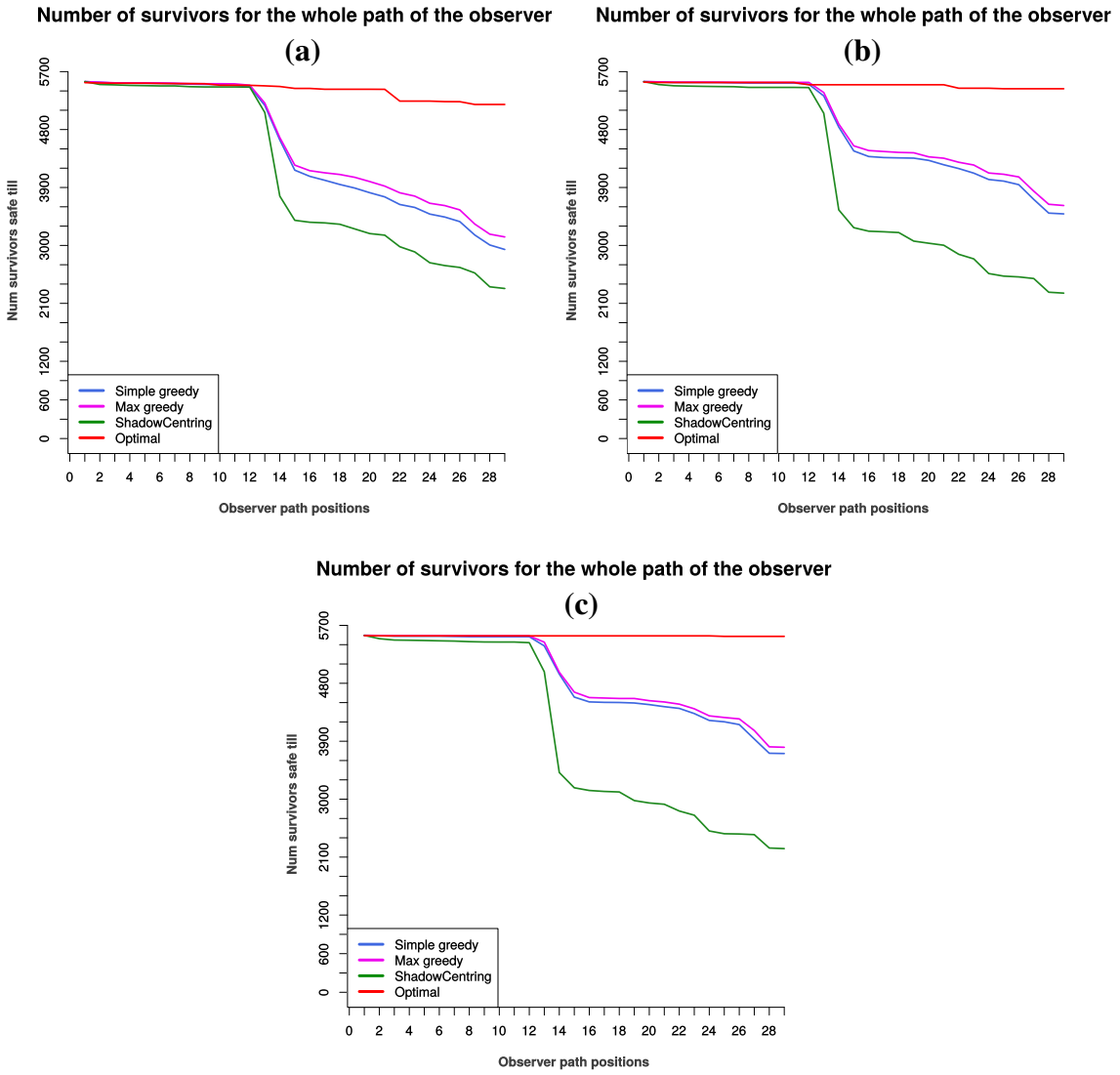


Figure 4.43 Number of survivors remaining for Path II, till the observer path points for **(a)** 60% **(b)** 100% and **(c)** 120% speed of the observer.

4.6.2.2 Collision Based

The plots for all survivor speeds are similar as seen from the figure 4.44 for the *collision based* approach for *Path II*. The survivors having 120% speed perform better than the lower speeds. The difference between 60% and 100% speeds is meagre. The survivors at 60%

speed seem to do better in the very beginning, but we have seen similar behavior for other levels and *Path I*, when the survivors having lower speeds are unable to move further as compared to higher speed survivors, and hence hide better behind the obstacles.

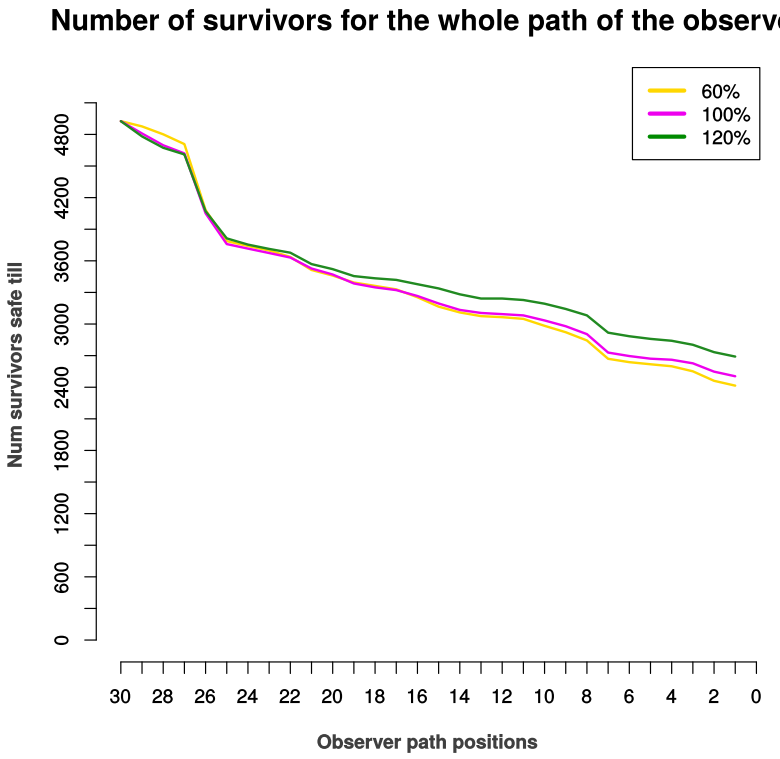


Figure 4.44 Number of survivors remaining till the observer path for collision based approach

Chapter 5
Conclusion and Future Work

We started our work from a basic idea of making an AI character hide from another character using simple techniques. Using a simple theoretical game level with a single obstacle as a proof of concept, we thought of simple heuristic strategies which could be used to hide. Greedy approach is a common method used in various fields, and is especially suitable for our purposes. We adopted the approach and adjusted it for our needs in a discretized game level environment. By studying the shadow polygons, we came up with the *Shadow centering* strategy. To measure the success of our strategies we had to compare them with an ideal strategy. For that purpose, we created an *optimal* strategy based on the discretized game level and radius of movement (based on the survivor speed).

The idea behind our work is to facilitate understanding and designing the game level, inspecting various observer paths and identifying the safety level of areas in the game map for different survivor speeds. This is performed through simulations on the discretized game level.

Our main assumption for the heuristic and optimal strategies was that the survivor NPCs do not collide with each other. With the *Collision based* approach, we tried to develop a heuristic technique which can tell us if there are different hidden paths for the survivor NPCs to survive as a group.

We experimented with various game levels, considering multiple observer paths and multiple relative survivor speeds. We tested theoretical levels and through them, we described our algorithms in greater detail. We examined real game levels from various genres

such as stealth shooters, role playing strategy games(RPGs) and first person shooters.

The results we found were heavily dependent on initial placement of survivors in the shadow polygons, the distribution and placement of obstacles, the size and structure of obstacles, and the observer paths. By increasing the survivor speeds for each level, the number of survivors generally went up but it did not change the relative trends between different strategies.

The sharp drops in the number of survivors in our plots, is directly attributed towards the sudden change in visible regions due to sharp vertices of the obstacles near the observer path. The performance of each strategy vary with the structure and size of the obstacles, and the observer paths, which alter the visibility polygon, and hence the effectiveness of each approach. If two observer paths are at different regions of the game level, which have contrasting structural formations, we will see different results for the strategies, as observed in *Level V*.

The survivors which start further away from the frequently changing visibility polygon edges, are able to hide for longer. For *shadow centering* strategy, the survivors already near the local shadow central regions fared better as others have to move to these central shadow regions and they might get discovered while moving to these areas.

Our experiments show that there is no single strategy which can come close to the performance of *optimal* strategy for all game levels as they vary widely in structural design. Generally, *max shadow greedy* strategy outperformed other heuristic strategies. A hybrid approach might provide a more generic solution to our NPC hiding issue. This involves using *shadow centering* approach under normal circumstances to move towards a more safe region and shifting to a greedy approach during close encounters with the observer's visibility polygon.

Blending intelligent techniques with the heuristic strategies might give better results. Predicting the future observer path based on its current movements and moving towards a region already searched out by the observer, are two ideas which if explored further, might be crucial in developing more robust hiding techniques. However these ideas might not work well in multiplayer first person shooter games as the player movements are harder to predict and they often frequently visit the regions already searched. They might be more effective in role playing and stealth games. Moreover these techniques involve path finding

methods which are more resource intensive and that might affect the game performance.

Studying the rate at which visibility changes while going around obstacles and the change in the type and the size of area being exposed might be another interesting idea which could be formulated into a hiding heuristic.

Our *optimal* approach is based on a brute force technique. We analyze all discretized positions on the game level and save them in a graph structure. From this structure, different survivor paths can be calculated through a simple search such as the longest path, the shortest safe path, the path which ends at a position closest to a given goal position. This might be quite helpful in stealth path planning in certain applications.

For *collision based* approach, which considers multiple survivors at the same time, it would be interesting to explore different communication abilities of local spatial knowledge being shared by the survivors to make more effective decisions.

We would also like to study the effects of limited visibility of the observer in range and the field of view, using multiple observers in the game level, and adding uncertainty to the observer's path. Use of waypoint graphs in the game levels and convex hulls for obstacles might be helpful for simple heuristic strategies on avoiding being trapped in concavities of obstacles, but it might not be possible for many game levels.

Bibliography

[1] Tetsuo Asano. An efficient algorithm for finding the visibility polygon for a polygonal region with holes. *IEICE Transactions (1976-1990)*, 68(9):557–559, 1985.

[2] E. Birgersson, A. Howard, and G. S. Sukhatme. Towards stealthy behaviors. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1703–1708 vol.2, Oct 2003.

[3] Francisc Bungiu, Michael Hemmer, John Hershberger, Kan Huang, and Alexander Kröller. Efficient computation of visibility polygons. *arXiv preprint arXiv:1403.3905*, 2014.

[4] Svante Carlsson, Håkan Jonsson, and Bengt J Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete & Computational Geometry*, 22(3):377–402, 1999.

[5] Wei-Pang Chin and Simeon Ntafos. Optimum watchman routes. In *Proceedings of the second annual symposium on Computational geometry*, pages 24–33. ACM, 1986.

[6] Timothy H. Chung, Geoffrey A. Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316, 2011.

[7] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

[8] Hossam El Gindy and David Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2(2):186–197, 1981.

[9] Steve Fisk. A short proof of chvátal's watchman theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978.

[10] flexcreator. Stealth Kills for Followers - Standalone. http://www.nexusmods.com/skyrim/mods/37277/?, 2013. [Online; accessed 30-May-2016].

[11] Subir Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, New York, NY, USA, 2007.

[12] Leonidas J. Guibas, Jean-Claude Latombe, Steven M. Lavalle, David Lin, and Rajeev Motwani. *Algorithms and Data Structures: 5th International Workshop, WADS'97 Halifax, Nova Scotia, Canada August 6–8, 1997 Proceedings*, chapter Visibility-based pursuit-evasion in a polygonal environment, pages 17–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997.

[13] Leonidas J. Guibas, Rajeev Motwani, and Prabhakar Raghavan. The robot localization problem in two dimensions. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '92, pages 259–268, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.

[14] Paul J Heffernan and Joseph S.B. Mitchell. An optimal algorithm for computing visibility in the plane. *SIAM Journal on Computing*, 24(1):184–201, 1995.

[15] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *Trans. Rob.*, 21(5):875–884, October 2005.

[16] Barry Joe and R.B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987.

[17] kryptopyr. Better Stealth AI for Followers - No Torch while Sneaking. https://steamcommunity.com/sharedfiles/filedetails/?id=104003521, 2012. [Online; accessed 30-May-2016].

[18] Niel Lebeck, Thomas Mølhave, and Pankaj K. Agarwal. Computing highly occluded paths on a terrain. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 14–23, New York, NY, USA, 2013. ACM.

[19] D.T. Lee. Visibility of a simple polygon. *Computer Vision, Graphics, and Image Processing*, 22(2):207–221, 1983.

[20] Lars Lidén and Valve Software. Strategic and tactical reasoning with waypoints. *AI Game Programming Wisdom, Charles River Media*, pages 211–220, 2002.

[21] Katelyn Mann. Evolving robot behavior to play hide and seek. *Journal of Computing Sciences in Colleges*, 18(5):257–258, 2003.

[22] M. Marzouqi and R. A. Jarvis. Covert robotics: hiding in known environments. *Robotics, Automation and Mechatronics, 2004 IEEE Conference on (Volume:2)*, 2:804–809, 2004.

[23] Mohamed S. Marzouqi and Ray A. Jarvis. New visibility-based path-planning approach for covert robotic navigation. *Robotica*, 24(6):759–773, November 2006.

[24] Geib C. W. Moore, M. B. and B. D. Reich. Planning for reactive behaviors in hide and seek. *Proceedings of the 5th Conference on Computer Generated Forces and Behavioral Representation*, 1995.

[25] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987.

[26] Srinivas Ravela, Richard Weiss, Bruce Draper, Brian Pinette Allen Hanson, Allen Hanson, and Edward Riseman. Stealth navigation: Planning and behaviors. In *Proceedings of ARPA Image Understanding Workshop*, pages 1093–1100, 1994.

[27] Ichiro Suzuki and Masafumi Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal of Computing*, 21(5):863–888, 1992.

[28] Y. A. Teng, D. DeMenthon, and L. S. Davis. Stealth terrain navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):96–110, Jan 1993.

[29] A. Tews, M. J. Mataric, and G. S. Sukhatme. Avoiding detection in a dynamic environment. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3773–3778 vol.4, Sept 2004.

[30] A. D. Tews, G. S. Sukhatme, and M. J. Mataric. A multi-robot approach to stealthy navigation in the presence of an observer. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 3, pages 2379–2385 Vol.3, April 2004.

[31] Benjamín Tovar and Steven M. LaValle. Visibility-based pursuit-evasion with bounded speed. In *Algorithmic Foundation of Robotics VII*, pages 475–489. Dept. of Computer Science, University of Illinois at Urbana-Champaign 61801, USA, 2008.

[32] Jonathan Tremblay, Pedro Torres, Nir Rikovitch, and Clark Verbrugge. An exploration tool for predicting stealthy behaviour. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2013.

[33] Dustin White. Clarifications and extensions to tactical waypoint graph algorithms for video games. *ACM-SE 45 Proceedings of the 45th annual southeast regional conference*, pages 316–320, 2007.