Constrained Pursuit-Evasion Problems in the Plane

by

Warren A. Cheung

B.Sc., University of British Columbia, 2002

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

September 2005

Abstract

In pursuit-evasion problems, we are presented with one or more pursuers attempting to capture one or more evaders. We consider pursuers and evaders limited by a maximum speed moving in the two-dimensional plane with obstacles. We then investigate two problems in this domain. In the first, where we are given the starting configuration of pursuers and evaders, we identify all possible paths by the evaders that are not intercepted by pursuers, and the points reachable by the evaders before the pursuers by following these paths. In the second problem, we consider a pursuer forced to maintain visibility with an evader. We construct an example that demonstrates there exists, in addition to the two standard outcomes of the pursuer capturing the evader and the evader losing sight of the pursuer, a third tie outcome, where the pursuer never loses sight of the evader, but the evader can also avoid capture indefinitely. We give the conditions under which each of these three outcomes occur for our specific situation.

Contents

List of Figures

Acknowledgements

I would like to acknowledge the incalculable help and advice of William Evans. I would also like to acknowledge Mike Klaas and Jocelyn Smith for their preliminary work and insight on the original formulation of the visibility constrained pursuit-evasion problem, which started me down the road that led to this thesis. As well, I would like to acknowledge my second reader David Kirkpatrick for his suggestions and invaluable perspective. Last but never the least, I would like to thank my parents and family for their neverending love and support, and my friends for reminding me to smile.

Warren A. Cheung

The University of British Columbia
September 2005

Chapter 1

Introduction

1.1 Pursuit-Evasion Problems

Pursuit-evasion problems consider an asymmetric scenario where we have two classes of entities, pursuers and evaders. Pursuers attempt to "capture" evaders, whereas evaders attempt to avoid "capture", where "capture" can have meanings such as physical proximity or visual detection. We present here a sample of this rich problem area, providing some context for aspects pertinent to the rest of the thesis.

In general, the environment in which the pursuers and the evaders can move depends on the application, and can also include additional elements that modify movement. The most general case would be to allow motion in all three space dimensions[16], however, this is often simplified to a discretisation of the plane[13] or a graph[11]. We choose an intermediate environment — the two-dimensional plane, with movement constrained by polygonal obstacles. This maintains correlation with the physical world while being rich enough to model applications in areas such as land and nautical movement.

Pursuers and evaders may also have other movement constraints. A maximum speed or acceleration may be imposed, and the entities may have a maximum turning rate[13, 16]. We consider a simple movement model — the pursuers and evaders are modelled as point sources with a maximum speed and infinite acceleration. This is a reasonable approximation if we consider a zoomed out point of view, and a relatively large time scale.

The definition of capture can also vary. To capture the evader, a pursuer may have to close within a certain distance of the evader[13] or occupy the same position as the evader[8, 11]. In other applications, establishing or maintaining visibility can be sufficient[13, 18]. We shall consider the simplest model for capture, when the pursuer and the evader occupy the same position. These conditions are sufficient to create interesting interactions between pursuers and evaders.

The information provided to the pursuers and evaders can also be varied. A pursuer may be given an evader's entire strategy[13, 8], or it may be given a model of the evader's behaviour[13, 18, 11]. Likewise, an evader may have information about the pursuers. A pursuer may be required to discover the location of an evader, or it may have sensors which only provide information about the position of the evaders when certain a criteria is fulfilled, such as being within a maximum range from that pursuer. We consider the worst-case scenario for the evaders — the evaders wish to guarantee that capture by the pursuers is impossible.

Additionally, we consider visibility constrained motion in the latter portion of this paper. This problem, also known as target-tracking, is another form of the pursuit-evasion problem[13, 10, 8]. In this case, the pursuer can be considered a camera which sees the evader. The "capture" goal for the pursuer is relaxed or modified — the objective for the pursuer may be to maintain sight of the evader while minimising the distance to the evader (the classic "capture" situation) or minimising its own movement. In this situation, both the pursuer and the evader have limited speed, and move in an environment where obstacles can both impede movement and occlude visibility. Predictable evaders, where the path of the evader is known in advance, and unpredictable evaders are both considered. In this problem instance, the relationship between the pursuer and the evader can be viewed as much less confrontational — often, the evading entity may not be actively attempting to avoid capture, but may be following a preplanned route.

1.2 Evader's Worst-Case Pursuit-Evasion

In the subsequent chapters of this thesis, we consider two situations of worst-case (from the evader's perspective) pursuit-evasion, where the pursuer is given complete knowledge of the evader's movements in advance. In these situations, an evader avoids capture only if no possible combination of pursuer movements can capture the evader.

In Chapter 2, we are given a starting configuration for the pursuers and the evaders. We then demonstrate a method for computing the set of all the points that the evader can reach without chance of capture by the pursuer. Using L_1 distance, we show that this set can be described using polygonal boundaries that are polynomial in size and which can be computed in polynomial time.

In Chapter 3, we consider a pursuer forced to maintain visibility to an evader. The addition of the visibility constraint creates the possibility of a tie situation between the pursuer and the evader, when we consider evader paths of infinite length. The pursuer cannot capture the evader without losing sight of the evader.

However, there is an infinite length path for the pursuer where the visibility criteria is never violated, even though the distance between the pursuer and the evader is upper-bounded and lower-bounded by finite values. We demonstrate using a specific instance that three outcomes (pursuer can capture the evader, the evader can always break line-of-sight and the pursuer can tie) are possible, and give the conditions necessary for each outcome.

Again, in both situations, we consider the environment to be the two-dimensional plane. We also consider obstacles in the environment that block movement and occlude visibility. We limit the movement of each pursuer and each evader to a maximum speed.

Chapter 2

Pursuit-Evasion Voronoi Diagram

2.1 Introduction

Let P be the set of m pursuers. Let there be an evader e. For each site p (one of the pursuers or the evader in the set $P \cup \{e\}$), we will provide a starting point $\mathbf{p}$, a speed v_p and a starting time t_p. Let O be the set of line segment obstacles with n vertices that block the movement of both pursuers and evaders. The task is to partition the plane into regions reachable by the evader without any chance of capture by the pursuer and regions where capture by the pursuers is possible. Each region reachable by the evader includes a description of how the evader can reach the points in the region. This partition will be called the pursuit-evasion Voronoi diagram.

We can view the evader as a robber, escaping the scene of a jewelry heist, and the pursuers as the cops. The robber can compute the pursuit-evasion Voronoi diagram, where the evader starting position is the location of the jewelry store and the pursuer starting positions are the nearby police stations. The obstacles model regions with no streets such as parks and rivers. The evader starting time is the amount of time it takes the robber to load a car full of ill-gotten gains, and the starting time for each of the pursuers is the delay for the cops at that police station to wake up and get into their cars.

The pursuit-evasion Voronoi diagram computes "safe" regions. For any point in a "safe" region, the point can be reached by the robber without danger of capture by any of the cops, and the diagram describes a path to the point for the robber. The robber may wish to ensure a safe haven or the border crossing into another country lies within a "safe" region.

This does not mean that paths that pass through the regions not marked "safe" guarantee capture, but only that the possibility for capture exists along these paths. For example, there may be many cops leaving each police station taking different paths, whereupon following "safe" paths may be a better choice. However, there may be very few cops on duty, whereupon the robber may be unmolested even when taking paths that pass through the regions not marked "safe" in the pursuit-evasion Voronoi diagram.

Therefore, the pursuit-evasion Voronoi diagram provides a diagram which separates "safe" escape routes from risky ones, leaving the actual decisions to the user. As well, the robber can recompute the pursuit-evasion Voronoi diagram during the escape, as the position of the robber changes and new information about the positions of the cops becomes available, resulting in a new pursuit-evasion Voronoi diagram.

We also propose a method to merge the pursuit-evasion Voronoi diagram for several evaders. We shall refer to the set E be the set of s evaders. We can combine the pursuit-evasion Voronoi diagram of several evaders to obtain a combined diagram that identifies the points in the plane reachable by some evader, and a shortest path without capture that reaches the points in each region. Alternatively, given the set E of evaders and the pursuit-evasion Voronoi diagram for each evader, we provide a method to compute the region reachable by all the evaders.

We now give several precise mathematical definitions that will enable us to specify the exact meaning of capture and evasion.

Definition 2.1.1. Given a speed v, the *travel time* for a path with length l is l/v. Therefore, if we are given a starting time t, the absolute *arrival time* for a path length l, given speed v and starting time t, is $\frac{l}{v} + t$.

An evader e can reach a point $\mathbf{p}$ via an evasive path if there exists a path ϕ from $\mathbf{e}$ to $\mathbf{p}$ such that for every point $\mathbf{q}$ along the path, the arrival time for the evader e along the path to $\mathbf{q}$ is shorter than the shortest arrival time for any pursuer to $\mathbf{q}$. To make this more precise, let us consider a path ϕ, and points $\mathbf{x}$ and $\mathbf{y}$ on the path ϕ. Let $d_\phi(\mathbf{x}, \mathbf{y})$ be the distance along the subpath of ϕ from $\mathbf{x}$ to $\mathbf{y}$, where $d_\phi(\mathbf{x}, \mathbf{y}) = \infty$ if the subpath of ϕ from $\mathbf{x}$ to $\mathbf{y}$ crosses an obstacle in O. For convenience, let $d_\phi = d_\phi(\mathbf{x}, \mathbf{y})$ if ϕ is a path from $\mathbf{x}$ to $\mathbf{y}$. Let $\overline{d}(\mathbf{a}, \mathbf{p}) = \min_{\rho \in \Phi(\mathbf{a},\mathbf{p})} d_\rho(\mathbf{a}, \mathbf{p})$, where $\Phi(\mathbf{a}, \mathbf{p})$ is the set of all paths from $\mathbf{a}$ to $\mathbf{p}$. In other words, $\overline{d}(\mathbf{a}, \mathbf{p})$ is the length of the shortest obstacle-avoiding path from $\mathbf{a}$ to $\mathbf{p}$ if such a path exists, or ∞ otherwise.

Definition 2.1.2. A path ϕ from $\mathbf{e}$ to $\mathbf{p}$ is an *evasive path* for the evader e if and only if

$$\forall \mathbf{q} \in \phi, d_\phi(\mathbf{e}, \mathbf{q})/v_e + t_e \leq \min_{f \in P} \overline{d}(\mathbf{f}, \mathbf{q})/v_f + t_f$$

Let $d^*(e, \mathbf{p})$ be the length of the shortest evasive path to $\mathbf{p}$ for evader e if such a path exists, or ∞ otherwise.

Definition 2.1.3. An evader e can *reach* a point $\mathbf{p}$ if there is an evasive path from $\mathbf{e}$ to $\mathbf{p}$ for the evader e.

Definition 2.1.4. A point $\mathbf{p}$ belongs to the region for evader e if and only if

$$d^*(e, \mathbf{p})/v_e + t_e \leq \min_{j \in P} \overline{d}(\mathbf{j}, \mathbf{p})/v_j + t_j$$

Definition 2.1.5. A point $\mathbf{p}$ belongs to the region for pursuer q iff

$$\overline{d}(\mathbf{q}, \mathbf{p})/v_q + t_q \leq \min_{j \in P} \overline{d}(\mathbf{j}, \mathbf{p})/v_j + t_j$$

We call the partition of the plane into regions belonging to each pursuer or evader the *pursuit-evasion Voronoi diagram*. This partition will describe all the points which can be reached by the evader without being captured by the pursuer, and will also give a shortest path to reach these points. In the following sections, we propose an algorithm to calculate the pursuit-evasion Voronoi diagram that runs in polynomial time under the L_1 distance metric (See Figure 2.1 for an example of a pursuit-evasion Voronoi diagram). We also note that this partition can be used to upper and lower bound the L_2 diagram, and provide methods of merging the diagrams of multiple evaders.

2.2 Related Work

The pursuit-evasion Voronoi diagram is a variation of the traditional Voronoi diagram that integrates many properties of several other well-known variations of the Voronoi diagram. In particular, we shall show that the pursuit-evasion Voronoi diagram incorporates elements of the compoundly weighted and the constrained Voronoi diagrams[3, 17], while adding restrictions similar to the crystal growth Voronoi diagram[21, 20].

In a traditional Voronoi diagram in two dimensions, we are given a set of generator points, also known as sites. The plane is partitioned into regions, where all points in a region are closer to one site than any other site, with distance being calculated using a particular distance metric, such as L_2. We can note that although the distance to a point is used to partition the plane, this is equivalent to using the arrival time along a straight line path, if we consider all sites as having a speed of 1.

The compoundly weighted Voronoi diagrams [4, 17] allow sites to have both an additive weight and a multiplicative weight. We can model starting time and

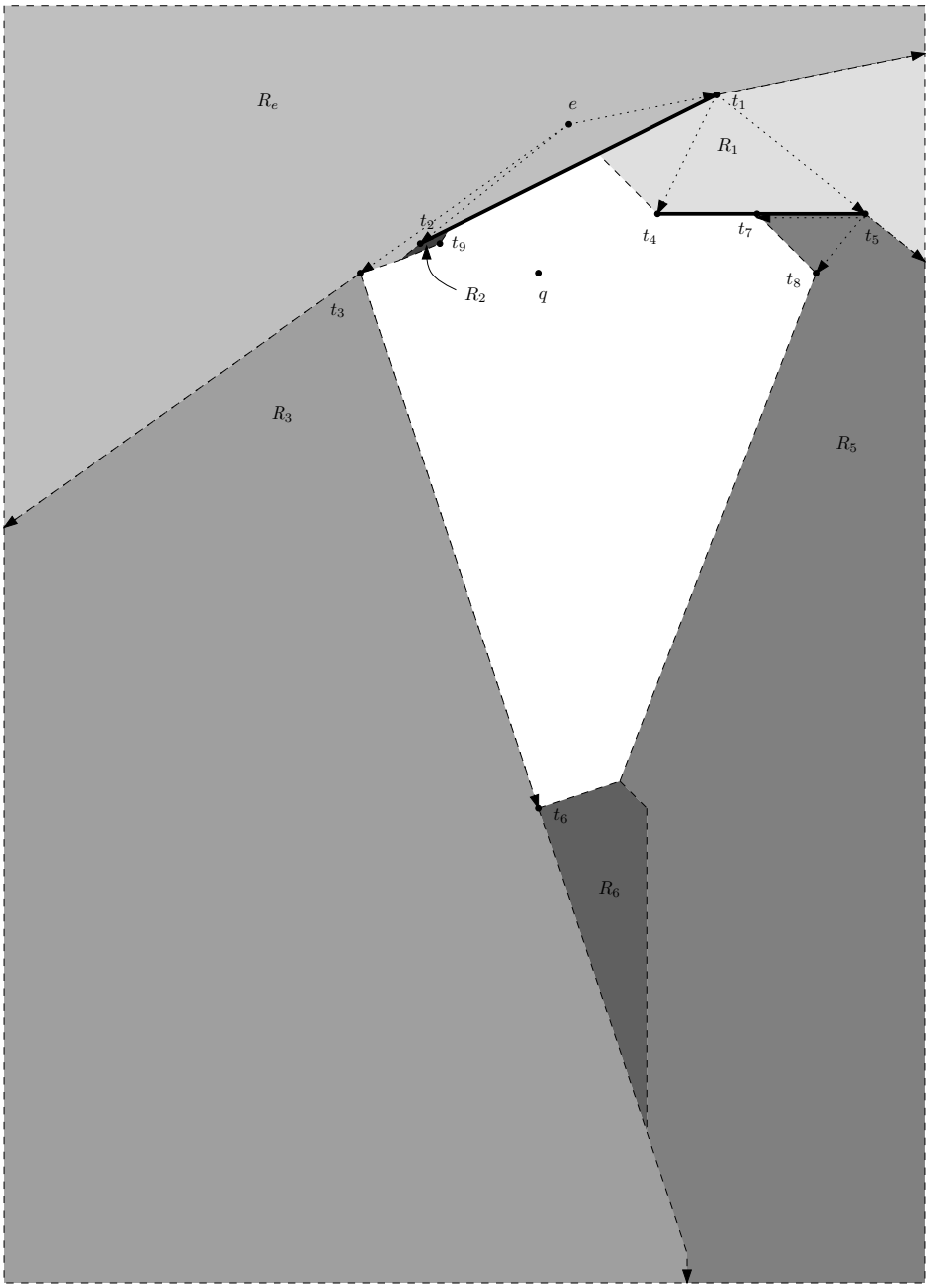


Figure 2.1: A simple example of the pursuit-evasion Voronoi diagram in L_1 for the evader e and the pursuer q, where the evader e has twice the speed of the pursuer. The thick black lines represent obstacles. Points in the region R_e can be reached the evader taking a straight line path from $\mathbf{e}$ to the point in R_e. Points in R_1 can be reached by the evader taking a path from e to t_1 and then to the point in R_1, and so on. Some points, such as t_4, t_7, t_8 and t_9 can be reached but do not have regions of their own.

(the inverse of) speed in the context of arrival time using additively weighted and multiplicatively weighted sites. The Voronoi diagram partitions the plane into regions, where a point is in the region for a site if the arrival time of a straight line path from the site is the earliest out of the arrival times of straight line paths from all the sites.

In an additively weighted Voronoi diagram, every site is assigned an additive weight. Again, if we consider all sites to have speed 1, the travel time along a straight line will be equivalent to the distance. Therefore, the arrival time at a point from a site is the distance between the point and the site plus the additive weight associated with the site.

In a multiplicatively weighted Voronoi diagram, every site is assigned a multiplicative weight. The travel time of the straight line path to a point from a site, in this case, is the distance between the site and the point multiplied by the multiplicative weight associated with the site. Therefore, we can consider the multiplicative weight to be the inverse of the speed for a site.

In a compoundly weighted Voronoi diagram, we define the arrival time at a point from a site as the distance given by the distance function multiplied by the multiplicative weight, plus the additive weight. This is equivalent to assigning each site a starting time and speed, and considering the arrival time of straight line paths to points.

The constrained Voronoi diagram [9, 2, 3] introduces a set of edges, also known as obstacles. The distance from a site to a point is defined to be the length of the shortest obstacle-avoiding path from the site to the point. We can view constrained Voronoi diagrams as a partition of the plane into regions. A point is in the region for a site if the arrival time to the point from the site, along a shortest obstacle-avoiding path, is the earliest of all the arrival times of obstacle-avoiding paths to the point from a site.

The multiplicatively weighted crystal growth Voronoi diagram introduces another constraint on the paths taken to the points in the region for a site — paths cannot pass through points passed through earlier by another site. This models the region obtained where each site represents a crystal and each crystal grows outwards simultaneously at the constant speed associated with the respective site. The crystals may only grow on empty space, and therefore must grow around the regions established by other sites. Schaudt and Drysdale[21, 20] detail an numerical expanding wavefront method to construct the crystal Voronoi diagram in L_2, as well as an event-driven expanding wavefront algorithm for computing the crystal Voronoi diagram for convex polygon distance functions, including L_1.

The pursuit-evasion Voronoi diagram problem will handle the presence of ob-

stacles and allow sites to be assigned velocities (multiplicative weights) and starting times (additive weights). In addition, our definition of evasive path distance treats points where the pursuer can capture the evader as additional obstacles the evader cannot pass through. However, rather than requiring every site's path to avoid every other site's region (as is the case for crystal Voronoi diagrams), we require that the evaders avoid the pursuers' regions, and allow pursuers to move freely through evaders' regions.

2.3 Motivation

Unlike most variations of the Voronoi diagram, we have asymmetry in the manner in which sites are dealt with, as we classify sites as either pursuers or evaders. Pursuers behave in a manner similar to sites in a constrained, compoundly weighted Voronoi diagram, considering all paths avoiding obstacles, whereas evaders behave similarly to sites in a constrained, compoundly weighted crystal growth Voronoi diagram, considering only paths that avoid both obstacles and capture by pursuers.

A natural way to view the pursuit-evasion Voronoi diagram is as the diagram obtained by the growth of two types of agents. Commonly-used examples in other Voronoi diagrams include the spread of bacteria in a petri dish or the growth of crystals[21, 20, 3]. In this case, one agent (modelled by the pursuer) is not limited by anything other than its speed, but the other (modelled by the evader) may only grow into clear space. For example, we can consider two bacterial strains, one strain that can be killed by the presence of an antibiotic, and the other strain producing that particular antibiotic and being immune to its effects. The first strain cannot grow into regions occupied by the second, as it would be killed by the presence of the antibiotic being produced, whereas the second strain does not have this problem. In a more generalised setting, we could also consider approximating other asymmetric populations of predators and prey.

Another setting which considers the growth of two types of agents is the spread of a biological or chemical agent. The pursuer in this case represents a counter-agent that controls the spread of the agent. The evader region describes the area that is contaminated by the agent before the arrival of the counter-agent. For example, the spread of the Black Death in Europe and the spatial spread of rabies among foxes[15] have both been modelled as epidemics with wavefronts which expand outwards at constant rate.

The pursuit-evasion Voronoi diagram can also be viewed in the context of worst-case motion planning. For example, we can consider the agent we are moving to be the evader, and moving objects in the world to be the pursuers. In this

simplification, we consider the pursuers and the evaders as points moving in a plane. Interception by a pursuer, in this example, corresponds to the potential of collision between the agent and an object. For any point in the evader's reachable region, there is a path guaranteed to be free of collisions for the agent. Instead of modelling the behaviour of the other agents, we assume the worst possible case. This allows us to generate paths which can never be intercepted, given the specified constraints. Many real-world motion planning applications, such as those which involve danger to human life, cannot tolerate collisions. For example, traffic control, camera operation in telesurgery and other virtual presence applications[14] are situations where any possibility of collision is unacceptable. Other Voronoi diagrams have also been used in this context — Kobayashi and Sugihara[12] use the an approximation of the crystal Voronoi Diagram to find a worst-case optimal path for an arbitrarily shaped evader among circular pursuers.

The concepts of growth and motion planning can also be combined. There are pathfinding applications that, in addition to fixed obstacles, also incorporate the idea of growing impediments. We can use the pursuit-evasion Voronoi diagram to determine the regions that can be reached before being rendered impassible by some expanding impediment to movement, such as the spread of a forest fire[19] or inundation by an expanding body of water. In this case, we treat the pursuer sites as a growing source which the evaders must avoid.

We shall show that Euclidean distance (the L_2 metric), although arguably the most natural measure of distance in the plane, results in diagrams which can involve circular arcs and spirals. We ultimately use the L_1 distance metric in our algorithm. L_1 distance is also known as rectilinear distance, and can be used to model distance travelled along city streets, when the streets run either North-South or East-West[1]. We shall also show a method to approximate the L_2 pursuit-evasion Voronoi diagram using the L_1 pursuit-evasion Voronoi diagram.

2.4 Voronoi Diagrams

The *distance* between two points will be defined using a distance metric. A distance metric $d(\mathbf{x}, \mathbf{y})$ satisfies three properties:

- $d(\mathbf{x}, \mathbf{y})$ is always non-negative, and is zero if and only if $\mathbf{x} = \mathbf{y}$
- $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$ (reflexivity)
- $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y})$ (triangle inequality)

Common distance metrics are the L_k distance metrics, where given points $\mathbf{a}$ and $\mathbf{b}$ with Euclidean coordinates (x_a, y_a) and (x_b, y_b), $d(\mathbf{a}, \mathbf{b}) = \sqrt[k]{(|x_a - x_b|)^k + (|y_a - y_b|)^k}$.

Definition 2.4.1. A compoundly weighted Voronoi diagram for two points $\mathbf{a}$ and $\mathbf{b}$, with multiplicative weights v_a and v_b, additive weights t_a and t_b, is a partition of the plane into a region for a and a region for b, where a point $\mathbf{p}$ is in the region for a if and only if $d(\mathbf{a}, \mathbf{p})/v_a + t_a \leq d(\mathbf{b}, \mathbf{p})/v_b + t_b$.

In the remainder of this section, we shall investigate several simple examples of the pursuit-evasion Voronoi diagram, and show the relationship between the pursuit-evasion Voronoi diagram and other kinds of Voronoi diagrams.

2.4.1 Pursuit-Evasion Voronoi Diagram for a single Pursuer and a single Evader

Let us consider the case when there is one pursuer a and one evader b with no obstacles. We shall note the differences between the use of the L_2 norm and the L_1 norm. To simplify, we shall consider when both the pursuer and evader start at the same time ($t_a = t_b$) and in the absence of obstacles.

If the speed of the pursuer is equal to the speed of the evader, the pursuit-evasion Voronoi diagram is a normal Voronoi diagram. Note that by definition, the boundary between the regions belongs to the pursuer. In the L_2 norm, this partitions the plane into two regions separated by a straight line, the perpendicular bisector of the line $\overline{\mathbf{ab}}$ (See Figure 2.2). We note that the asymmetry between the pursuer and evader does not impact the diagram in this case as all the paths taken by the evader are straight line paths that cannot be intercepted by the pursuer, and all the points owned by the pursuer cannot be reached by any path by the evader in a shorter amount of time.

For the L_1 norm, the pursuit-evasion Voronoi diagram is the normal L_1 Voronoi diagram. Let $\mathbf{a} = (x_a, y_a)$ and $\mathbf{b} = (x_b, y_b)$. Within the box bounded by $x = x_a$, $x = x_b$, $y = y_a$, $y = y_b$, we have a separator slanted at a 45 degree angle. Once the boundary intersects this bounding box, it will change direction. If the separator intersects the top or bottom edge of the bounding box, the separator continues as a vertical line. If the separator intersects the left or right sides of the bounding box, it continues as a horizontal line (See Figure 2.3). Note that if it intersects with a corner of the bounding box, we can have a region of equality in addition to a boundary line of equality, which is owned by the pursuer (See Figure 2.4). Again, all the points in the evader region are reachable by straight line paths that cannot be intercepted by the pursuer.

If the pursuer is faster than the evader, the pursuit-evasion Voronoi diagram is a multiplicatively weighted Voronoi diagram. In L_2, the boundary is formed by an Apollonius circle[4, 17, 21, 20] (See Figure 2.5 for an example). Without loss of generality, let us translate and rotate the diagram such that $\mathbf{b}$ is at the origin

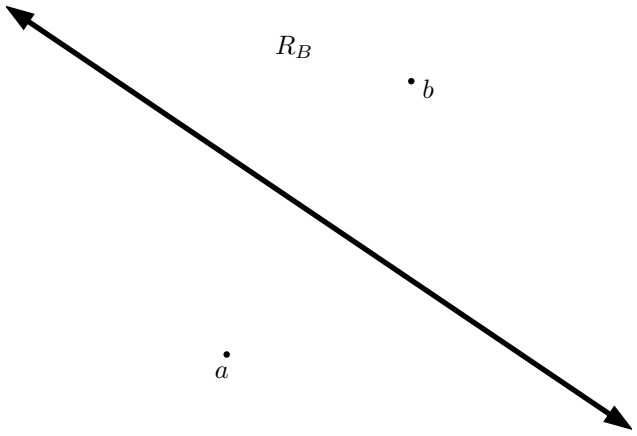


Figure 2.2: Pursuit-evasion Voronoi diagram in L_2 for the pursuer a and the evader b when the pursuer and evader have the same speed. The solid line separates the region for a from the region for b (Same as the standard Voronoi diagram for two sites in L_2). The evader region is R_b, and consists of points which are all reachable via a straight line path from the evader start point b.

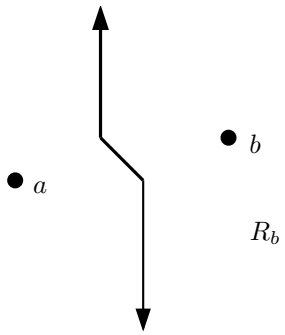


Figure 2.3: Pursuit-evasion Voronoi diagram in L_1 for the pursuer a and the evader b when the pursuer and the evader have the same speed. It is composed of a line at 45 degrees and vertical lines (Same as the standard Voronoi diagram for two sites in L_1). The evader region is R_b, and consists of points which are all reachable via a straight line path from the evader start point b.

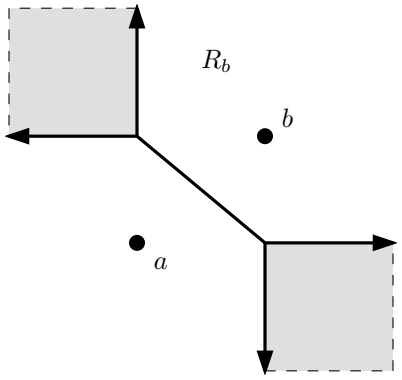


Figure 2.4: Another example of the pursuit-evasion Voronoi diagram in L_1 for the pursuer a and the evader b when the pursuer and the evader have the same speed. The solid line and the grey region represent the region where the distance from a is the same as the distance from b, and therefore belongs to the pursuer a. Again, the boundaries are the same as in the standard Voronoi diagram for two sites in L_1. R_b is the region reachable by the evader, and consists of points that are all reachable via a straight line path from the evader start point b.

and **a** lies on the positive x-axis at $(x_a, 0)$, with the pursuer starting at **a** and the evader starting at **b**. The multiplicative weights for sites **a** and **b** will be $\frac{1}{v_a}$ and $\frac{1}{v_b}$ respectively, with $v_a > v_b$. We get a circle such that the two points $(\frac{x_a v_b}{v_a + v_b}, 0)$ and $(\frac{-x_a v_b}{v_a - v_b}, 0)$ are opposite ends of the diameter of the circle. We note that the region for the evader in the pursuit-evasion Voronoi diagram is always contained in the corresponding region in a multiplicatively weighted Voronoi diagram, as capture by the pursuer is ignored the multiplicatively weighted Voronoi diagram. However, all the paths taken by the evader b in this example are straight line paths to the circle boundary, with the earliest interception point at the circle boundary. As no paths extend beyond the circle boundary, the evader region in the pursuit-evasion Voronoi diagram is the corresponding multiplicatively weighted Voronoi region.

If the evader is faster than the pursuer, the boundary between the pursuer and the evader in the pursuit-evasion Voronoi diagram is the boundary in a multiplicatively weighted crystal growth Voronoi diagram. In multiplicatively weighted crystal growth diagrams, the evader treats all points in the pursuer's region as obstacles, which is equivalent to the constraint for the pursuit-evasion Voronoi diagram. As well, in multiplicatively weighted crystal growth diagrams, the pursuer treats the points in the evader's region as obstacles, but this does not affect the diagram as the pursuer cannot reach any point in the evader region via a straight line path before the evader can reach it. In L_2, the boundary is partly an Apollonius circle,

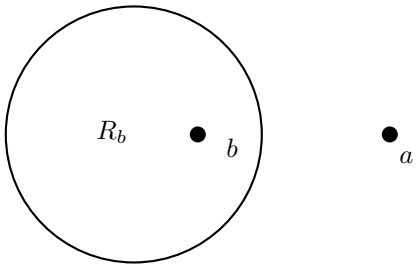


Figure 2.5: Pursuit-evasion Voronoi diagram in L_2 for the pursuer a and the evader b, where the pursuer is faster than the evader (Same as the multiplicative Voronoi diagram in L_2). R_b is the region reachable by the evader, and consists of points that are all reachable via a straight line path from the evader start point b.

same as in the multiplicatively weighted Voronoi diagram, and partly a logarithmic spiral[21, 20] (See the example in Figure 2.6).

The points in the evader region that can be reached by straight line paths, unstopped by the pursuer, lie in the region R_a. They form a boundary that incorporates part of the circular boundary from the multiplicatively weighted Voronoi diagram (see the portion of the boundary of the region R_a between the points t_1 to t_2 and closer to a in Figure 2.6). Note that the rest of the circular boundary (the dashed boundary in Figure 2.6), obtained by paths from the evader that pass through the pursuer controlled region, is not valid, as these evader paths can be intercepted by the pursuer. However, a valid evader path would be for the evader to run to a point tangent to the circular boundary (such as t_1 or t_2 in the example), and then to follow a path that is just outside the reach of the pursuer. The points along this path have the property that the distance along the path is proportional to the distance to the pursuer starting point, which describes a logarithmic spiral (See the solid portions of the boundary for the regions R_1 and R_2).

In general, shortest evasive paths can also leave the spiral boundary and continue in a straight line fashion towards their destination, possibly interacting with other pursuer points and forming more complex curves. Considering the L_2 norm as the distance function, if there exist pursuers or evaders (in other words, i and j in $P \cup E$) with differing speeds ($v_i \neq v_j$), or in the presence of obstacles ($O \neq \emptyset$), there can be boundaries between regions that are formed by second degree curves[17]. We obtain hyperbolic curves at the boundaries between points with differing additive weights, which can also occur with the addition of obstacles. As seen in the previous examples, we can also get circular arcs when multiplicative weights differ, and sections of logarithmic spirals are possible when the speed of an evader exceeds that of

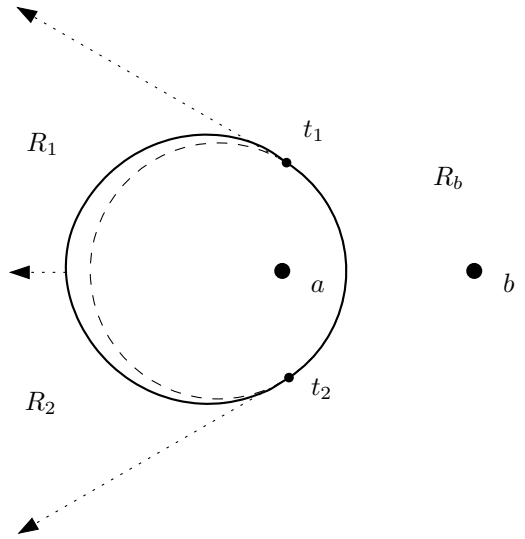


Figure 2.6: Pursuit-evasion Voronoi diagram in L_2 for the pursuer a and the evader b, where the evader is faster than the pursuer (Same as the crystal growth Voronoi diagram in L_2). The region R_a consist of points whose shortest evasive path is a straight line path from a. The region R_1 consists of points whose shortest evasive path is a straight line to the point t_1, following the solid line boundary and then a straight line to the destination (Similarly for R_2). The dotted circle inside is the circular boundary from the multiplicative Voronoi diagram, for comparison with the boundary which is composed of two logarithmic spirals starting at t_1 and t_2, and intersecting at the boundary between R_1 and R_2.

a pursuer. As we are unaware of general closed form solutions for the boundaries in the L_2 multiplicatively weighted crystal growth Voronoi diagram[21, 20], it is likely that the boundaries in the pursuit-evasion Voronoi diagram can be equally complex and would require numerical methods to compute.

We can note that when we are working in the L_1 metric, the solution for the curves which form the multiplicatively weighted Voronoi diagram for two points are straight lines. When we have a pursuer that is faster than the evader, the boundary of the pursuit-evasion Voronoi diagram is the same as the boundary for the corresponding multiplicatively weighted Voronoi diagram. When solving for the boundary of the multiplicatively weighted Voronoi diagrams in general for two points $\mathbf{a}$ and $\mathbf{b}$, we are solving the equation

$$d(\mathbf{x}, \mathbf{a})/v_a = d(\mathbf{x}, \mathbf{b})/v_b$$

Without loss of generality, we can consider the speed of $\mathbf{b}$ to be 1, $\mathbf{a}$ to lie at the origin, and $\mathbf{b}$ to lie in the first quadrant. Therefore, in L_1, we are considering

$$(|x| + |y|)/v_a = (|x_b - x| + |y_b - y|)$$

However, we can note that this can be broken up into nine cases. We have three possible kinds of values for x: $x \leq 0$, $0 \leq x \leq x_b$ and $x \geq x_b$. Similarly, there are three kinds of values for y: $y \leq 0$, $0 \leq y \leq y_b$ and $y \geq y_b$. Therefore, the above equation reduces to nine possible lines in each of the nine regions. Of course, not every region will have a valid solution — we obtain a polygon that surrounds the slower site. This gives us, like the diagram in L_2, the pursuit-evasion Voronoi diagram in L_1 when the evader is slower than the pursuer, if we note that all paths taken by the evader follow straight lines to the boundary without entering the pursuer's region.

For the crystal growth Voronoi diagram in L_1, the boundary is again composed of straight line segments and will only bend when it reaches the boundary of the nine regions we mentioned before. We refer to Schaudt[20], which details a general algorithm for computing the boundary of crystal growth Voronoi diagrams for L_1 (and other distance functions). Just as in the L_2 case, we need to carefully consider which points can be reached via straight line paths. Therefore, when $v_a > 1$, we will have part of the diagram that is identical to the multiplicative Voronoi diagram. However, the logarithmic spiral in L_2 simplifies to a series of lines in L_1. We can note the solution to the compoundly weighted Voronoi diagram in L_1 is very similar to that of the multiplicatively weighted Voronoi diagram, with boundaries composed of straight line segments. When we consider the paths which need to bend around the pursuer region in the crystal growth Voronoi diagram,

we can compute the portion that bends around the pursuer region (the part which corresponds to the logarithmic spiral in L_2) using the solution to the compoundly weighted Voronoi diagram — we can consider a site at the point where the path turns, and has the same speed as the original site but a starting time equal to the time it takes the original site to reach the turn. In this manner, we can follow the boundary around the pursuer region. Otherwise, by similar arguments as the L_2 case, the crystal growth Voronoi diagram in L_1 also describes the L_1 pursuit-evasion Voronoi diagram when the evader is faster than the pursuer (See Figure 2.7 for an example).

2.5 Pursuit-Evasion Voronoi Diagram

We shall begin with defining several terms useful in describing the pursuit-evasion Voronoi diagram and proving properties of the structures discovered by our algorithm.

Definition 2.5.1. The *crystal* around an obstacle vertex or pursuer start point $\mathbf{p}$ with respect to the evader e (and the set P of pursuers and the set O of obstacles) is the set of points

$$C = \{q | \min_{a \in P} \left(\overline{d}(\mathbf{a}, \mathbf{p})/v_a + t_a + d_{\overline{\mathbf{pq}}}/v_a\right) \leq (d^*(\mathbf{e}, \mathbf{q})/v_b + t_b)\}$$

The crystal is the region around $\mathbf{p}$ that no evasive path can pass through without being intercepted by a pursuer path that passes through $\mathbf{p}$. Note that this also implies that, for all points $\mathbf{q}$ in a crystal, *all* paths from the evader starting position will be intercepted en route to $\mathbf{q}$.

Note. The union of the crystals around every obstacle vertex and the pursuer start point is the pursuer's region.

Definition 2.5.2. The vertices on the boundary of a crystal C are *crystal vertices*.

Definition 2.5.3. A *shortest path tree* for an evader e is a tree whose root is $\mathbf{e}$ and whose other nodes are obstacle vertices or crystal vertices (with respect to $E = \{\mathbf{e}\}$) such that if any point $\mathbf{p} \neq \mathbf{e}$ is reachable by e then there exists a shortest evasive path $(\mathbf{e}, \ldots, \mathbf{p}', \mathbf{p})$, where $(\mathbf{e}, \ldots, \mathbf{p}')$ is a path in the shortest path tree.

Definition 2.5.4. The nodes in the shortest path tree for an evader e are called e's *tree nodes* and consist of the evader starting point, obstacle vertices and crystal vertices.

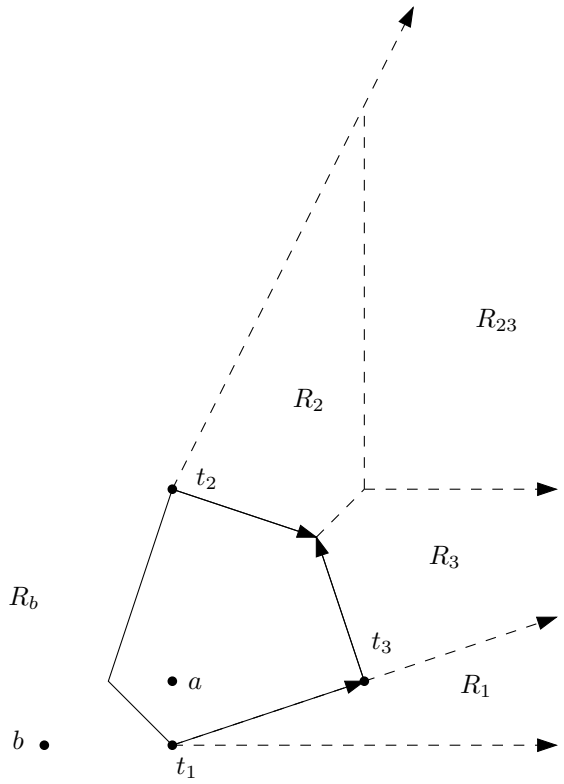


Figure 2.7: Pursuit-evasion Voronoi diagram in L_1 for the pursuer a and the evader b, where the speed of the evader is greater than the speed of the pursuer. R_b is the region of points reachable via a straight line path from b. R_1 is the region of points reachable via a path from b to t_1 and then via a straight line to the destination, and similarly for R_2 and R_3 (which is reached via t_1). Note that there is a degenerate situation in the region R_{23}, between the points t_2 and t_3. This is a region of equality, where paths via t_2 are equal in distance to paths via t_1, similar to the situation in Figure 2.4.

Lemma 1. *If a shortest obstacle avoiding path in L_1 from* $\mathbf{p}$ *to* $\mathbf{q}$ *exists, there must exist a shortest obstacle avoiding path from* $\mathbf{p}$ *to* $\mathbf{q}$ *consisting of straight line segments with endpoints at obstacle vertices,* $\mathbf{p}$ *and* $\mathbf{q}$.

This result was shown by Clarkson et al.[6].

Lemma 2. *If a shortest evasive path from* $\mathbf{p}$ *to* $\mathbf{q}$ *exists for evader e, there exists a shortest evasive path consisting of straight line segments with endpoints at the evader's starting point, crystal vertices and obstacle vertices.*

Proof. We can consider the boundary of the crystals as obstacles — no evasive path can pass through the crystal boundary, effectively blocking motion just like an obstacle. Also, there exists an evasive path to every point not in a crystal. Therefore, we can consider the problem of finding a shortest obstacle avoiding path, considering both the crystal boundaries and the original obstacles as obstacles. Therefore, Lemma 2 is a special case of Lemma 1. □

As tree nodes are the vertices of the pursuer boundary region and the vertices of the obstacles, a shortest path can be found that passes only via tree nodes and $\mathbf{p}$ and $\mathbf{q}$. Therefore, a shortest path tree must exist for any evader, set of pursuers and set of obstacles.

Definition 2.5.5. We define a path $\pi = (\mathbf{x_0}, \ldots, \mathbf{x_k})$ as the path consisting of the sequence of straight line segments $\overline{\mathbf{x_0x_1}}$, $\overline{\mathbf{x_1x_2}}$, …, $\overline{\mathbf{x_{k-1}x_k}}$.

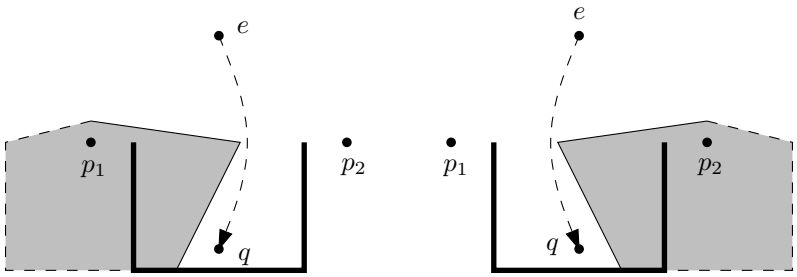
Definition 2.5.6. Given a path $\pi = (\mathbf{x_0}, \ldots, \mathbf{x_k})$, we define the '$\circ$' operator as $\pi \circ \mathbf{y} = (\mathbf{x_0}, \ldots, \mathbf{x_k}, \mathbf{y})$

We shall exploit the property shown in Lemma 2 in our algorithm for computing the pursuit-evasion Voronoi diagram in a plane with the presence of obstacles, which will run in polynomial time when using the L_1 distance metric.

2.6 Computing the Pursuit-Evasion Voronoi Diagram

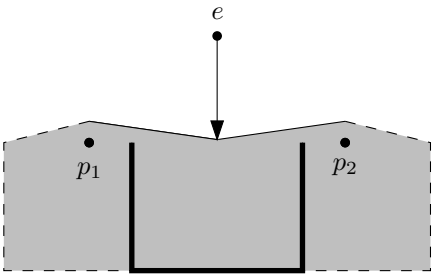
2.6.1 Overview of the Algorithm

At first glance, it might seem feasible to consider the diagram for each pursuer and evader independently, and then stitch these diagrams together. However, when considering shortest evasive paths, it is necessary to consider all pursuers, as a path is evasive only if it is not intercepted by any pursuer at any point along the path. For example, a point reachable by the evader when we consider two pursuers independently may not be reachable when both pursuers are considered simultaneously (See Figure 2.8).



(a) Part of the pursuit-evasion Voronoi diagram for the evader e with respect to pursuer p_1. The region in grey is part of the region owned by p_1. There exists an evasive path from **e** to **q**.

(b) Same situation as 2.8(a), except we only consider the evader e and the pursuer p_2.



(c) When we consider both p_1 and p_2, there no longer exists an evasive path to **q**.

Figure 2.8: An example demonstrating how an evader e may be able to reach a point **q** when considering the pursuers p_1 and p_2 independently, but is unable to reach **q** if both pursuers are considered together.

The approach we take during the algorithm is to repeatedly take an unprocessed tree node from a set of discovered points, and process the node by finding any new tree nodes reachable from the node to be processed. When all the tree nodes are found, the algorithm terminates. We can view the algorithm as being similar to a Dijkstra-style search for shortest paths from a single source in a visibility graph. In this case, in addition to considering obstacle vertices, we also need to consider crystal vertices, which are not known in advance. However, given a tree node, we will describe an operation that will find all its children in the shortest path tree (among other vertices) using geometric union, difference and intersection, and by computing the compoundly weighted Voronoi diagram for two points. As Dijkstra's algorithm only needs information about the successors of the processed nodes, we can implement Dijkstra's shortest path search without knowing the entire graph in advance, as long as we guarantee that by the time a node is processed, we can guarantee that it is a tree node, and has a correct shortest path distance estimate (See Algorithm 1 for an outline).

Algorithm 1 Outline of pursuit-evasion Voronoi diagram computation

while there exist unprocessed potential tree nodes **do**
  find an unprocessed tree node p to process.
  compute the region reachable from the tree node p.
  update the interim pursuit-evasion Voronoi diagram.
  check for new potential tree nodes.
  remove any points that can be determined to not be potential tree nodes.
end while

As tree nodes are processed, we discover regions reachable via shortest evasive paths through the tree node processed. Using these regions, we continually update a partition of the plane, where each region is associated with a processed path node. Each region for a tree node corresponds to the set of points where the shortest path to a point in the region is the path to the tree node (from the root) and the straight line path from the path to the point, out of all paths in the tree to a tree node concatenated with a straight line from that tree node to the point.

As the shortest obstacle-avoiding path for a pursuer to a point in the plane is independent of the evaders, we can compute the shortest path distances from each pursuer to each obstacle without considering the evaders.

Let O^* be the set of the obstacle vertices of all the obstacles in O. We note that $\overline{d}(\mathbf{a}, \mathbf{o})$, for pursuer $a \in P$ and $\mathbf{o} \in O^*$, is the shortest path from $\mathbf{a}$ to $\mathbf{o}$ in the visibility graph of $O^* \cup \mathbf{a}$, where an edge in the graph between vertices $\mathbf{x}$ and $\mathbf{y}$ has weight $d_{\overline{\mathbf{xy}}}$.

Definition 2.6.1. The set $\hat{S}$ is defined to be

$$\hat{S} = \left\{ \bigcup_{a \in P} \bigcup_{\mathbf{o} \in O^*} \left\{ (\mathbf{o}, v_a, \overline{d}(\mathbf{a}, \mathbf{o})/v_a + t_a \right\} \right\} \cup \hat{P}$$

where $\hat{P} = \bigcup_{p \in P} (\mathbf{p}, v_p, t_p)$.

For each pursuer $p \in P$ and obstacle vertex $\mathbf{o} \in O^*$, we create a triple $(\mathbf{o}, v_x, t_x)$ in $\hat{S}$, where v_x is the speed of p and t_x is the shortest arrival time of p at $\mathbf{o}$. Therefore, for any point $\mathbf{x}$, we can consider all of the obstacle vertices $\mathbf{o} \in O^*$ and pursuers $p \in P$ and compute the shortest path distance for each p with a path via each $\mathbf{o}$ to $\mathbf{x}$. By Lemma 1, if there exists a finite length shortest path to $\mathbf{x}$ it must be one of the paths considered. Therefore, the shortest arrival time from a pursuer to $\mathbf{x}$ can be expressed as

$$\min_{q \in P} (\overline{d}(\mathbf{q}, \mathbf{x})/v_q + t_q) = \min_{(\mathbf{s}, v_s, t_s) \in \hat{S}} (d_{\overline{\mathbf{sx}}}/v_s + t_s)$$

In general, we refer to a triple $\hat{x} = (\mathbf{x}, v_x, t_x)$ as having the equivalent of a starting position $\mathbf{x}$, a speed v_x and a starting time t_x.

We now consider the task of computing the regions associated with the evaders. To do this, we run Evader-Region (see Algorithm 2 for a detailed pseudo-code version of the algorithm). The result of Evader-Region is $\mathcal{P}$, which is a complete shortest evasive path partition for the evaders E.

Definition 2.6.2. Dequeue-Min(Q) returns a point $\mathbf{x} \in Q$ that has the minimum value for $d_e[\mathbf{x}]$. In other words, $\forall \mathbf{p} \in Q, d_e[\mathbf{p}] \geq d_e[\mathbf{x}]$.

Definition 2.6.3. Visible($\mathbf{x}, O$) is the set of all points $\mathbf{p}$ such that the straight line path $\overline{\mathbf{xp}}$ does not pass through an obstacle.

$$\text{Visible}(\mathbf{x}, O) = \bigcup_{\mathbf{p} \in \mathbb{R}^2} \{\mathbf{p} | d_{\overline{\mathbf{xp}}} < \infty\}$$

Definition 2.6.4. Voronoi-Region($\hat{p}, \hat{q}$) is the Voronoi region for the triple $\hat{p}$ in the compoundly weighted Voronoi diagram (without obstacles) for $\hat{p}$ and $\hat{q}$.

$$\text{Voronoi-Region}(\hat{p}, \hat{q}) = \{\mathbf{x} | d(\mathbf{x}, \mathbf{p})/v_p + t_p \leq d(\mathbf{x}, \mathbf{q})/v_q + t_q\}$$

Definition 2.6.5.

$$\text{Control-Region}(\hat{p}, \pi, R, \mathcal{E}) = R - \bigcup_{(\hat{x}, \pi_x, R_x) \in \mathcal{E}} (\text{Voronoi-Region}(\hat{x}, \hat{p}) \cap R_x) \qquad (2.1)$$

Algorithm 2 Evader-Region

$d_e[\mathbf{e}] := 0$, $\hat{P}_e := \emptyset$, $\pi_e[e] := (\mathbf{e})$, $\mathcal{E} = \emptyset$
Enqueue$(Q, \mathbf{e})$
while Q not empty **do**
  $\mathbf{p} :=$ Dequeue-Min(Q)\{Note: $d^*(e, \mathbf{p}) = d_e[\mathbf{p}]$\}
  $\hat{p} := \left(\mathbf{p}, v_e, \frac{d^*(e,\mathbf{p})}{v_e} + t_e\right)$
  $C := \bigcup_{\hat{x} \in \hat{S}}$ (Voronoi-Region$(\hat{x}, \hat{p}) \cap$ Visible$(\mathbf{x}, O)$) {Lemma 4}
  $R :=$ Visible$(\mathbf{p}, C \cup O)$ {Lemma 5}
  $R :=$ Control-Region$(\hat{p}, \pi_e[\mathbf{p}], R, \mathcal{E})$ {Definition 2.6.5,Lemma 6}
  $\mathcal{E} :=$ Merge-Region$(\hat{p}, \pi_e[\mathbf{p}], R, \mathcal{E})$ {Definition 2.6.6,Lemma 7}
  $B := R \cap (C \cup O)$ {Boundary of R shared with C and O}
  for all vertices $\mathbf{f}$ on the boundary B **do**
    if $d_e[\mathbf{f}] > d^*(e, \mathbf{p}) + d_{\overline{\mathbf{pf}}}$ **then**
      $d_e[\mathbf{f}] := d^*(e, \mathbf{p}) + d_{\overline{\mathbf{pf}}}$
      $\pi_e[\mathbf{f}] := \pi_e[\mathbf{p}] \circ \mathbf{f}$
      if $\mathbf{f} \notin Q$ **then**
        Enqueue$(Q, \mathbf{f})$
      end if
    end if
  end for
  $\hat{P}_e := \hat{P}_e \cup \{\hat{p}\}$
  for all $\mathbf{q} \in Q$ and $\mathbf{q} \notin O^*$ **do**
    if $\mathbf{q}$ is strictly inside the region $\bigcup_{(\hat{x}, \pi_x, R_x) \in \mathcal{E}} R_x$ **then**
      Dequeue$(Q, \mathbf{q})$
    end if
  end for
end while
return $\mathcal{P}$

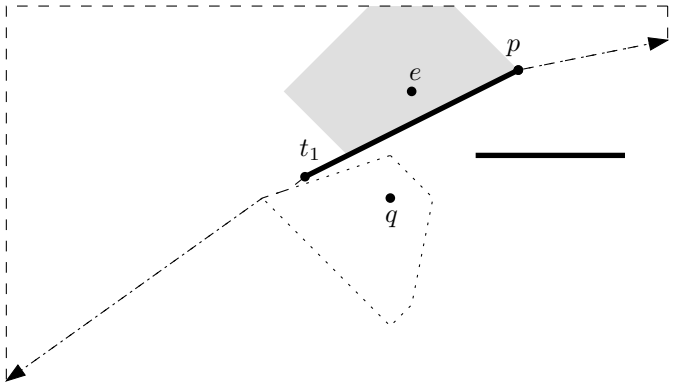


Figure 2.9: An alternative way of looking at the algorithm is to view the processing of each tree node as "increasing" the region of certainty. After processing the evader starting point **e**, we dequeue **p** from the queue, and therefore all points with shortest path arrival time earlier than **p** have the correct regions assigned, as shown by the grey region.

Definition 2.6.6. Let Merge-Region$(\hat{p}, \pi, R, \mathcal{E})$ be

$$\left\{ \bigcup_{(\hat{x}, \pi_x, R_x) \in \mathcal{E}} \{(\hat{x}, \pi_x, R_x - R)\} \right\} \cup \{(\hat{p}, \pi, R)\}$$

The Evader-Region algorithm iterates over the tree nodes for the evader e. After visiting a tree node, it computes a *partial partition* $\mathcal{E}$ for e with respect to the tree nodes visited so far by updating a previously computed partial partition.

Definition 2.6.7. We call $\mathcal{E}$ a *partial (shortest evasive path) partition* for a single evader e if there exists a non-crossing shortest evasive path tree T rooted at e such that for all tuples $(\hat{p}, \pi, R) \in \mathcal{E}$ where $\hat{p} = (\mathbf{p}, v_p, t_p)$:

1. **p** is a tree node
2. π is a shortest evasive path from e to **p** in T.
3. for all $\mathbf{q} \in R$, $\pi \circ \mathbf{q}$ is a shortest evasive path to **q** among all paths $\pi' \circ \mathbf{q}$ with $(p', \pi', R') \in \mathcal{E}$.

Note $\emptyset$ is a valid partial partition, and is also the initial partial partition when the algorithm starts.

We will show that after the algorithm visits all the tree nodes, we have a *complete partition* for the evader e.

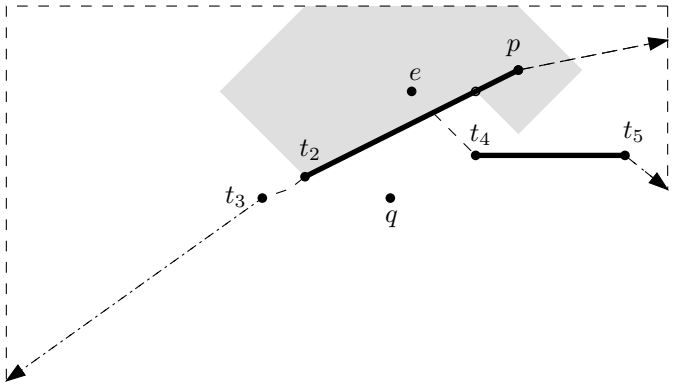


Figure 2.10: After processing $\mathbf{p}$, the algorithm will proceed to process $\mathbf{t_2}$. Therefore all points with shortest path arrival time earlier than $\mathbf{t_2}$ have the correct regions assigned, as shown by the increased grey region.

Definition 2.6.8. $\mathcal{E}$ is a *complete* partition for an evader e if $\mathcal{E}$ is a partial partition for e, with the additional constraint that if $\mathbf{p} \in T$ then $(\mathbf{p}, \pi, R) \in \mathcal{E}$.

The complete partition contains all the regions and paths for the pursuit-evasion Voronoi diagram, and is the final output of the algorithm.

2.6.2 Execution and Correctness of the Algorithm

To compute the complete partition for an evader e, the algorithm explores the shortest path tree for e, maintaining a queue of possible tree nodes Q and a partial partition $\mathcal{E}$. Initially, the queue contains $\mathbf{e}$, which is guaranteed to be the root of the tree, and the partial partition is the empty set.

For every iteration of the algorithm, we shall show it dequeues a tree node $\mathbf{p}$ to process from the queue Q by selecting the potential tree node in the queue with the earliest arrival time. Given that $\mathbf{p}$ has been correctly identified as a tree node, we wish to determine new regions and new potential tree nodes reachable by the evader via this tree node. We accomplish this by computing the region R that is reachable by straight line evasive paths from $\mathbf{p}$ (via a shortest evasive path to $\mathbf{p}$) and determining the subregion of these points where the arrival time from $\mathbf{p}$ is earlier than the arrival time from any other tree node already processed. We then investigate the portion of the boundary of the evader reachable region that was discovered by computing R, updating the arrival time of the vertices on this part of the boundary and adding to Q any new vertices as potential tree nodes. We also check for potential tree nodes in Q which lie within completely within R, and

remove them as they cannot be tree nodes as they are now known to not lie on the boundary. Finally, the partial partition $\mathcal{E}$ is updated by adding the tuple for $\mathbf{p}$, and changing the tuples in $\mathcal{E}$ so that the new $\mathcal{E}$ is also a partial partition.

Dequeuing a tree node p

To do this, while the queue contains points, we first dequeue the point $\mathbf{p}$ with the lowest potential arrival time from Q.

Lemma 3. *The point* $\mathbf{p}$ *is a tree node, and* $\pi_e[\mathbf{p}]$ *is a shortest evasive path to* $\mathbf{p}$.

Proof. We shall note that $d_e[\mathbf{p}]$ is nondecreasing for each point $\mathbf{p}$ dequeued from Q. This is because at each time we dequeue a point from Q, we dequeue the point with the minimum value. As all distances in a distance metric are non-negative, any point reachable by a shortest evasive path via $\mathbf{p}$ that we enqueue must have a greater value (See lines 13 and 16)

Suppose $\mathbf{p}$ is not a tree node. Then there must exist a tree node such that $\mathbf{p}$ lies within the region of another tree node. However, if it lies within the region of another tree node, the shortest path to $\mathbf{p}$ will be via this other tree node, and thus π cannot be the shortest evasive path from e to $\mathbf{p}$.

Suppose π is not a shortest evasive path from e to $\mathbf{p}$ in T. Let the shortest evasive path from e to $\mathbf{p}$ be π'. Let $\mathbf{x}$ be the last tree node in π' such that for some π_x and R_x, $(\mathbf{x}, \pi_x, R_x) \in \mathcal{E}$. Let $\mathbf{y}$ be the point that follows $\mathbf{x}$ in the path π'. For $\mathbf{y}$ to be on the path π' to $\mathbf{p}$, $d_e[\mathbf{y}] < d_e[\mathbf{p}]$. However, this would mean $\mathbf{y}$ has already been processed and thus there must exist a tuple $(\mathbf{y}, \pi_y, R_y) \in \mathcal{E}$. As this is a contradiction, π must be the shortest evasive path from e to $\mathbf{p}$. □

Computing the Region R

We then compute C, all the points of potential pursuer capture that surrounds $\mathbf{p}$. More precisely, we identify all the points along straight line paths that extend from $\mathbf{p}$ where a pursuer can intercept the path. In this case, we notice that interception occurs where equality between pursuer shortest obstacle-avoiding paths and evader shortest evasive paths. We then exploit the compoundly weighted Voronoi diagram for two sites and visibility to compute C.

Lemma 4. *The region* C *on line 6 is the set of all points such that a straight line path from* $\mathbf{p}$ *to the point (ignoring obstacles), given a speed of* v_p *and a starting time of* t_p*, can be intercepted by at least one of the pursuers in* P *in the presence of the obstacles* O.

Proof. Voronoi-Region($\hat{x}, \hat{p}$) computes all the points in the plane that are closer or equal distance to $\hat{x}$ than $\hat{p}$, in the absence of obstacles. By intersecting with Visible($\mathbf{x}, O$), we restrict the region to only include points that are reachable by a straight line path from $\hat{x}$. This has the effect of only considering straight line paths from $\mathbf{x}$ that do not pass through obstacles. Therefore, the region calculated comprises the points in the plane where a straight line path from $\hat{p}$ (which may or may not pass through obstacles) is intercepted by a pursuer via a straight line path from $\mathbf{x}$ (which may not pass through an obstacle).

Therefore, if we union the regions for all $\hat{x} \in \hat{S}$, we know by the construction of $\hat{S}$ that we will have considered the shortest obstacle-avoiding path by any pursuer to all points in the plane, as such a path involves a straight line from some $\hat{x} \in \hat{S}$, and we will have identified all the points in the plane whose straight line path from $\hat{p}$, ignoring obstacles, can be intercepted by some pursuer (See Figure 2.11). □

Now that we have all possible pursuer interception points, we can determine the region that can be reached by an evasive path via $\mathbf{p}$. A straight line path from $\mathbf{p}$ is valid until it is intercepted by an obstacle or a pursuer, at which point all points further along the line are not reachable. This corresponds neatly with the concept of visibility. This step therefore computes visibility considering, as obstacles, the region C computed previously (containing all points where capture could occur) as well as the actual obstacles O.

Lemma 5. *The region R on line 7 is the set of points that a straight line path from $\hat{p}$ can reach without being intercepted by a pursuer in P or being blocked by an obstacle in O.*

Proof. The region Visible($\mathbf{p}, O$) is the set of all points reachable in a straight line from $\mathbf{p}$ that do not involve paths that cross an obstacle. The region Visible($\mathbf{p}, C$) is the set of all points reachable in a straight line from $\mathbf{p}$ that cannot be intercepted by a pursuer. We note that both obstacles and interception points share the property that an evasive path may not pass through them. Therefore, the region Visible($\mathbf{p}, O$) ∩ Visible($\mathbf{p}, C$) = Visible($\mathbf{p}, O \cup C$) is the set of all points reachable in a straight line from $\mathbf{p}$ that do not involve paths that cross an obstacle or an interception by a pursuer (See Figure 2.12). □

To compute the region for a tuple in a partial partition, we need to ensure that not only are all the points in the region reachable via *some* evasive path through the tree node $\mathbf{p}$, but that the path is also a *shortest* evasive path. Therefore, we now introduce the function Control-Region. The purpose of this function is to prune all the points in the region R which have a later arrival time from $\mathbf{p}$ than from a discovered tree node in the partial partition $\mathcal{E}$.

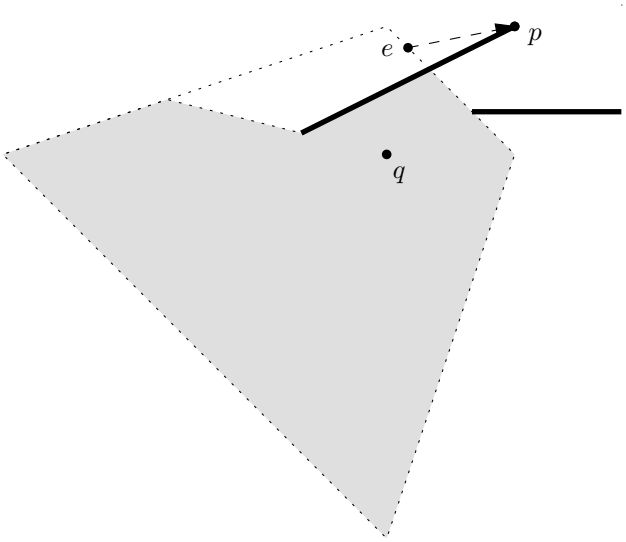


Figure 2.11: After processing the evader starting point **e**, we process the point in Q with the earliest arrival time, **p**, The outer dotted boundary is the boundary of Voronoi-Region($\mathbf{q}, \mathbf{p}$), and is computed ignoring all the obstacles. The shaded region is Voronoi-Region($\mathbf{q}, \mathbf{p}$)$\cap$Visible($\mathbf{q}, O$), the region C where the pursuer could intercept the evader, and considers the region inside the outer boundary that is visible from the pursuer starting point q. Heavy black lines indicate obstacles. The evader's starting position is **e** and there is a single pursuer starting at **q**. Note that C does not allow the pursuer to pass through obstacles, but the evader may pass through obstacles. Obstacle constraints for the evader will be dealt with in the next step.

Lemma 6. *Assuming $\mathcal{E}$ is a partial partition,* Control-Region *(Definition 2.6.5) computes the region of points where a path from $\hat{p}$ has the shortest arrival time with respect to the points in the set of regions $\mathcal{E}$.*

Proof. By Lemma 5, the region R consists of all points reachable along a straight line path from $\hat{p}$. In $\mathcal{E}$, each triple $(\hat{x}, \pi_x, R_x)$ describes a region where a point **q** is in R_x if and only if the arrival time from $\hat{x}$ to **q** is the minimum arrival time over all triples in $\mathcal{E}$. Therefore, to add the region R to $\mathcal{E}$, we need to remove all the points in R such that the arrival time from $\hat{p}$ is greater than from a discovered tree node in $\mathcal{E}$. This can be calculated by taking, for every triple $(\hat{x}, \pi_x, R_x)$ in $\mathcal{E}$, the intersection of R_x and R and the Voronoi region for $\hat{x}$ in the diagram for $\hat{x}$ and $\hat{p}$ (See Figure 2.13). □

Once the region for the tree node **p** has been computed, we need to also update the regions for all the tuples in the partial partition $\mathcal{E}$, removing points

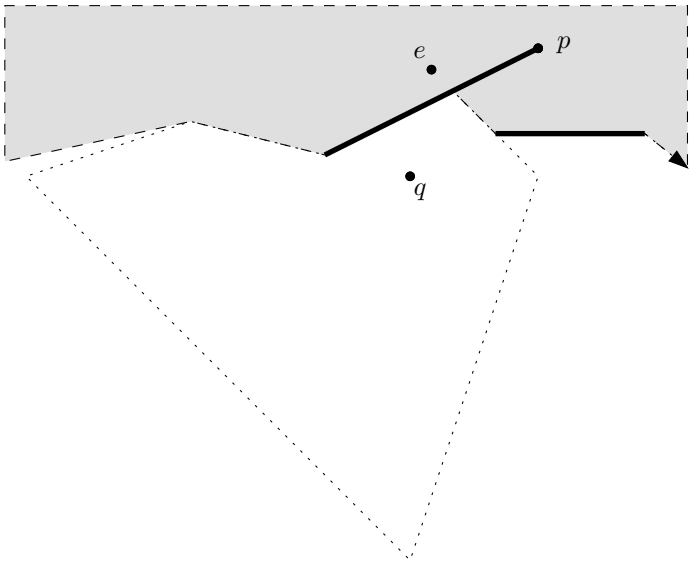


Figure 2.12: Considering the region C and the obstacles as occluding visibility, the greyed region indicates all the points "visible" from **p**. In other words, all points in the region can be reached via a straight line evasive path from **p**.

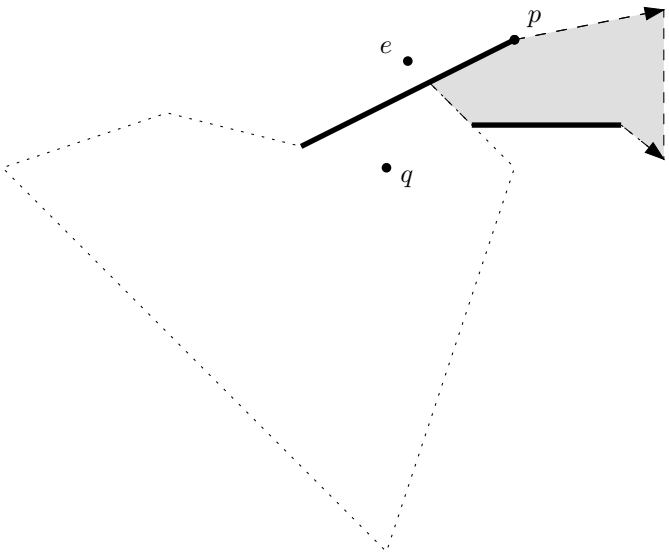


Figure 2.13: From **p**'s reachable region, we remove the set of points that have shortest evasive paths described in $\mathcal{E}$. In this diagram, those points are the ones reachable via a straight line path from **e**.

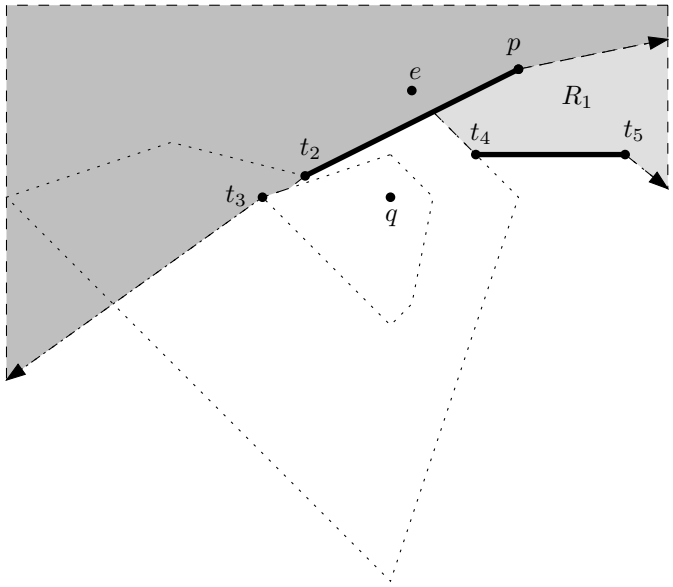


Figure 2.14: Once the light grey region for tree node **p** is merged into the diagram with the dark grey region for the evader starting position **e**, we remove from the queue all potential tree nodes which no longer satisfy the criteria for a tree node. The larger dotted polygon is the region C computed for **p**. The smaller dotted polygon is the region C computed for **e** during the last iteration of the loop, and is responsible for the bend in the boundary of the grey reachable region between t_2 and t_3. Note that t_4 is in a degenerate position — although it is an obstacle vertex, the evader cannot take any paths via t_4 as arrival time of the shortest evasive path for the evader at t_4 is exactly the same as the arrival time of the shortest obstacle-avoiding path for the pursuer.

which no longer satisfy the shortest evasive path property of the partial partition.

Lemma 7. *The procedure* Merge-Region *(Definition 2.6.6) adds the region R reachable from $\hat{p}$ via the path π to the partial partition $\mathcal{E}$.*

Proof. As the region R contains only points with the shortest arrival time from $\hat{p}$ (with respect to the points in $\mathcal{E}$), any region in $\mathcal{E}$ that overlaps with R must be diminished by that overlap before R can be added as a new tuple, to preserve $\mathcal{E}$ as a partial partition.

The procedure Merge-Region removes the region R from each existing region in $\mathcal{E}$ via set subtraction and then adds the tuple $(\mathbf{p}, \pi, R)$ to $\mathcal{E}$. □

We can see the partial partition computed after two iterations of the main algorithm loop in Figure 2.14. The vertices on the boundary of R_1 create new tree nodes t_4 and t_5. The unprocessed tree nodes t_2,t_3,t_4 and t_5 all lie on the boundary for the union of the grey reachable regions and are therefore still valid (See line 23).

Updating the Queue Q

At this point, we have the region for point $\mathbf{p}$ and we can compute the section of the boundary that was updated by the addition of this region. In the Dijkstra algorithm style, we need to update the distance estimates of all the reachable points. For our algorithm, this means considering vertices on the boundary as potential tree nodes which need to be updated. Therefore, any new vertices found on the section of the boundary updated need to be added to the queue and old vertices with a shorter distance estimate also need to be updated. As well, since there are points in the queue which may not be tree nodes, we remove any points which, given the new region computed, are no longer on the boundary and therefore are not tree nodes.

Lemma 8. *At line 4, the queue Q, including the point to be dequeued, contains all tree nodes that are on the boundary of the reachable region of $\mathcal{E}$.*

Proof. We shall show this lemma via induction.

Basis Step At line 1, Q is initialised with the evader starting point $\mathbf{e}$ and $\mathcal{E}$ is empty. Therefore, the first point processed by the loop is $\mathbf{e}$. As $\mathcal{E}$ is empty, after the first iteration of the loop, Q contains all the points on the reachable of the tree node $\mathbf{e}$.

Inductive Step Assume that for some iteration of the algorithm loop, at line 4, the lemma holds. Then during the subsequent iteration of the loop, we add a tuple $(\mathbf{p}, \pi, R)$ to $\mathcal{E}$. Only two situations can arise — new tree nodes on the

boundary are discovered and previously discovered tree nodes are no longer on the boundary. As seen in Lemma 7, the only change in the reachable region is in the region R. The crystal vertices and obstacle vertices on the boundary of R are added via line 16, and thus all new tree nodes are added to Q if not already present. Similarly, tree nodes which lie completely within the new region R are no longer on the boundary and are removed on line 23. Therefore, Q is updated correctly.

Therefore, every tree node on the boundary of the current reachable region of $\mathcal{E}$ is found on the queue Q. □

Algorithm Invariant

Lemma 9. *$\mathcal{E}$ is always a partial partition with respect to evader e.*

Proof. Initially, $\mathcal{E}$ for an evader e is set in line 1 to be empty, which is a valid partial partition.

Let us consider any tuple $(\mathbf{p}, \pi, R)$ added to $\mathcal{E}$. By Lemma 3, $\mathbf{p}$ is a tree node and π is the shortest path to $\mathbf{p}$. Therefore, by the operation proved by Lemma 6, we know that for all $\mathbf{q} \in R$, $\pi \circ \mathbf{q}$ is a shortest evasive path to $\mathbf{q}$ among all paths $\pi' \circ \mathbf{q}$ with $(p', \pi', R') \in \mathcal{E}$. Therefore, $\mathcal{E}$ is always a partial partition. □

Termination and Final Output

Lemma 10. *At line 2, $\mathcal{E}$ contains all the tree nodes of a shortest path tree for e. In other words, there exists a shortest path tree T such that for all $\mathbf{p} \in T$, there exists a region R where $(\mathbf{p}, \pi, R) \in \mathcal{E}$ and π is the path to $\mathbf{p}$ in the tree T.*

Proof. Let us consider the possibility that a reachable tree node t is not dequeued via the algorithm. Given a shortest path via tree nodes to reach the tree node t from an evader e, let us consider the first tree node s in the path not dequeued from Q. As the first point in the path is e, which is guaranteed to be dequeued, s must have a predecessor r, which by definition of s was processed. Let us therefore consider the iteration of the loop when r was processed. We can note that when r is processed, all tree nodes reachable from r are added to Q on line 16. As s is a tree node reachable from r, it must have been added to the Q and would therefore be processed. □

Lemma 11. *The final value for the region $\mathcal{E}$ calculated for an evader e is the complete partition for the evader e.*

Proof. As all tree nodes reachable from e are processed during the execution of the algorithm as seen in Lemma 10, and the region associated with each tree node is computed as seen in Lemma 6 and added to the partial partition $\mathcal{E}$ as seen in Lemma 7, $\mathcal{E}$ is the complete partition for evader e. □

Figure 2.1 shows the complete partition for a single evader e. R_i is the region for tree node $\mathbf{t_i}$, and R_e is the region for the root of the tree $\mathbf{e}$.

2.7 The Pursuit-Evasion Voronoi Diagram for Multiple Evaders

We now consider the set of evaders E. Given $\mathcal{D}$, the set containing the complete partition for every evader, we can merge the complete partitions into a total partition for the evaders.

Definition 2.7.1. We call $\mathcal{P}$ a *total shortest evasive path partition* for a set of evaders E if:

1. For each tuple $(\hat{p}, \pi, R) \in \mathcal{P}$, a shortest evasive path to any $\mathbf{q} \in R$ from the evaders in E is $\pi \circ \mathbf{q}$ starting at an evader $e \in E$ and:
   (a) $\mathbf{e}$ is the first point in π
   (b) $\hat{p} = (\mathbf{p}, v_e, t_p)$ where $\mathbf{p}$ is the last point in π and t_p is the arrival time of the evader e at $\mathbf{p}$
2. The set of the regions R in the tuples $(\hat{p}, \pi, R) \in \mathcal{P}$ form a partition of the set of all points reachable by the evaders E.

Starting with an empty set of regions, we incrementally add each region from each partition for each evader. After all the regions are processed, we have the total partition for the evaders E.

Algorithm 3 Total-Partition

```
1: for all 𝓔 ∈ 𝒟 do
2:    for all (p̂, π_p, R) ∈ 𝓔 do
3:       R := Control-Region(p̂, π_p, R, 𝒫)
4:       𝒫 := Merge-Region(p̂, π_p, R, 𝒫)
5:    end for
6: end for
7: return 𝒫
```

Lemma 12. *$\mathcal{P}$ is a total partition for all evaders E and describes all points reachable by the evaders.*

Proof. $\mathcal{P}$ is constructed by applying Control-Region and Merge-Region to merge each region in each complete partition $\mathcal{E}$ together. Therefore, $\mathcal{P}$ contains the regions associated with every tree node of every evader processed by the algorithm with respect to the pursuers. □

2.8 Complexity Analysis

We show that the algorithm runs in polynomial time. First, we demonstrate that there are a polynomial number of tree nodes, and that the complexity of the complete partition is also polynomial. Given the size of the output is polynomial, we show that the algorithm runs in polynomial time.

2.8.1 Size of the Pursuit-Evasion Voronoi Diagram

We note that Schaudt[20] shows that the multiplicatively weighted crystal Voronoi diagram has polygonal boundaries in L_1, but was unable to show a bound on the size of the diagram. They do conjecture that the size of the diagram for i sites is $O(i)$. Given that n is the number of obstacle vertices in O and m is the number of pursuers in P, we show the number of tree nodes processed (which includes all the points that form the boundary of regions) is of size $O((n+m)^2(mn+m))$. We show the complete partition for a single evader is $O(nT^2 + T^2)$, where T is the number of tree nodes. When we consider a set of evaders E, s is the size of that set. The total partition for s evaders is $O(nT^2s^2 + T^2s^2)$.

Lemma 13. *There are at most $O((n + m)(mn + m))$ crystal vertices at which shortest evasive paths can bend, and therefore $O((n + m)(mn + m))$ internal tree nodes.*

Proof. We note that internal tree nodes are either the evader start point, obstacle vertices, or crystal vertices. As there is exactly one evader start point and at most n obstacle vertices that can be internal tree nodes, we shall focus our analysis on the number of crystal vertices that can be internal tree nodes.

A crystal vertex $\mathbf{c}$ is a vertex of the pursuer region. It must have a parent $\mathbf{c}'$, and a shortest evasive path to $\mathbf{c}$ is via a path to $\mathbf{c}'$ and then to $\mathbf{c}$. $\mathbf{c}$ can only occur when the shortest evasive path via $\mathbf{c}'$ arrives at the same time as the shortest obstacle-avoiding path from the pursuers. Let p be the pursuer which follows this shortest obstacle-avoiding path, and $\mathbf{o}$ be the last obstacle vertex (otherwise, the

starting point $\mathbf{p}$) of this obstacle-avoiding path. Therefore, the boundary at $\mathbf{c}$ is defined by the shortest evasive path via $\mathbf{c}'$ by the evader e and the obstacle avoiding path via $\mathbf{o}$ by the pursuer p.

If $\mathbf{c}$ is an internal tree node, there must be a shortest evasive path that bends at $\mathbf{c}$ to reach another tree node, when we restrict our attention to shortest evasive paths that consist of straight line segments which pass through tree nodes and the destination point.

If a shortest evasive path is forced to bend at $\mathbf{c}$, the boundary of the pursuer region at $\mathbf{c}$ must also bend. In L_1, changes in direction on the boundary can only occur when the boundary crosses one of the axes centred on $\mathbf{c}'$ or one of the axes centred on $\mathbf{o}$, as the boundary at $\mathbf{c}$ is created by the interaction between the evader via $\mathbf{c}'$ and the pursuer p via $\mathbf{o}$.

If $\mathbf{c}$ occurs on an axis of $\mathbf{c}'$, the boundary must cross that axis of $\mathbf{c}'$. However, as the path from $\mathbf{c}'$ to $\mathbf{c}$ also follows that axis, the boundary will cross the path and so a shortest evasive path from $\mathbf{c}'$ to $\mathbf{c}$ cannot bend around the boundary at $\mathbf{c}$. Therefore, the $\mathbf{c}$ must lie on an axis of $\mathbf{o}$.

We shall now consider $\mathbf{o}$, an obstacle vertex or an evader starting point. Let us consider that the origin is at $\mathbf{o}$. We shall count how many times shortest paths from p via $\mathbf{o}$ create crystal vertices on each of the four axes. We shall examine the positive x-axis, but the three other axes are symmetrical.

We note that if the pursuer p is faster than the evader, then their meeting point is not an internal tree node. Again, a crystal vertex $\mathbf{c}$ is where the shortest evasive path for the evader has the same arrival time as the shortest obstacle-avoiding path for the pursuer. If the pursuer is faster, any path taken by the evader via $\mathbf{c}$ can be captured by the pursuer arriving earliest at $\mathbf{c}$ following the same path from $\mathbf{c}$ to the destination as the evader. As the pursuer has a higher speed, the pursuer will arrive earlier. Therefore, in this analysis, we shall only consider pursuers which have speed less than the evader.

We shall divide the positive x-axis into non-overlapping continuous ranges. We associate a range of points with a pursuer-point pair $(p', \mathbf{o}')$, with p' being a pursuer and $\mathbf{o}'$ being an obstacle vertex or the starting point for p', when all points in this range are reached by a shortest obstacle avoiding path from the pursuer p' and the last point where the path bends (or the starting point otherwise) is $\mathbf{o}'$. We shall count the number of crystal vertices which occur in the ranges for all the pairs $(p^*, \mathbf{o})$, where p^* is any pursuer and $\mathbf{o}$ is the obstacle vertex or pursuer starting point we are considering — all the pursuer-point pairs involving $\mathbf{o}$.

Within a continuous range R for $(p, \mathbf{o})$ on the positive x-axis of $\mathbf{o}$, at most one internal tree node can exist. If an internal tree node exists at some $x = i$ on

the positive x-axis, then the arrival time for the pursuer p via $\mathbf{o}$ is the same as the arrival time for the evader via $\mathbf{c}'$. All the points $x > i$ in this range cannot be reached by the pursuer before the evader, as the evader is faster than the pursuer. Therefore, we have at most one crystal vertex in each range for $(p, \mathbf{o})$.

Let us now consider the arrival times of shortest obstacle avoiding paths for pursuers via $\mathbf{o}$ along the positive x-axis. When $\mathbf{o}$ is a pursuer starting point, only that pursuer will take paths via this point. However, for the obstacle vertices, any of the pursuers could take paths via that obstacle vertex. For any pursuer p, a point at position x on the positive x-axis via $\mathbf{o}$ has arrival time $t(x)$ given by the linear equation $t(x) = t_o + x/v_p$, where t_o is the earliest arrival time of p at $\mathbf{o}$ (See Figure 2.15 for an example). If we consider the minimum arrival time along the axis $t_{min}(x)$, we will the obtain a continuous piecewise linear function that is composed of parts of these linear equations. As we are taking the minimum, the slope must be positive and non-increasing, and each pursuer p is responsible for at most one piece of the function, which is where a range $(p, \mathbf{o})$ can occur (See Figure 2.16 for an example).

However, we need to also consider ranges that involve points other than $\mathbf{o}$. Note that these ranges can interrupt what would have been a continuous range for $(p, \mathbf{o})$ into several ranges for $(p, \mathbf{o})$, each separated by ranges involving points other than $\mathbf{o}$ (and this is the only other method of obtaining more ranges associated with $\mathbf{o}$).

Let $(p_i, \mathbf{o_i})$ be a pursuer-point pair where $\mathbf{o_i} \neq \mathbf{o}$. Let us count the number of times ranges assigned to this pursuer-point pair can interrupt the axis. The arrival time of p_i from $\mathbf{o_i}$ to the x-axis is given by the two linear equations:

$$t'(x) = \begin{cases} t_i + (|y_i| + x_i - x)/v_i & x \leq x_i \\ t_i + (|y_i| + x - x_i)/v_i & x \geq x_i \end{cases}$$

where t_i is the earliest arrival time of p_i at $\mathbf{o_i}$ and v_i is the speed of pursuer p_i. Note that only one of these equations has a positive slope (See Figure 2.17 for an example).

Let us consider when a region R' for $(p_i, \mathbf{o_i})$ can lie to the left of a region R for $(p^*, \mathbf{o})$ (for some pursuer p^* and the point $\mathbf{o}$). The arrival time for pursuer p_i via $\mathbf{o_i}$ is the same as the arrival time for pursuer p^* via $\mathbf{o}$ at the boundary between a region for $(p_i, \mathbf{o_i})$ and $(p^*, \mathbf{o})$, is less than to the left and greater to the right. Therefore, this can only occur when $t'(q)$ intersects $t_{min}(x)$, with a steeper positive slope at the intersection point. However, as $t_{min}(x)$ is a piecewise linear function with positive and non-increasing slope, this can only occur at most once (See Figure 2.18 for an example).

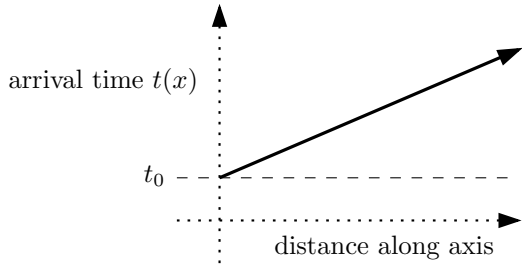


Figure 2.15: An example of the function $t(x)$ that describes the arrival time of a pursuer p along the axis of an obstacle vertex or pursuer starting point $\mathbf{o}$, when the pursuer takes a path via $\mathbf{o}$. t_o is the arrival time of the shortest obstacle-avoiding path from p to $\mathbf{o}$. The slope is the $\frac{1}{v_p}$. Note that both the slope and the arrival time must be positive.

We have at most $n+m$ obstacle vertices or pursuer starting points. Each of these have 4 axes. Each axis can have up to $O(mn+m)$ interruptions, one for each of the other pursuer-point pairs, and therefore at most $O(m)+O(mn+m) = O(mn+m)$ regions — $O(mn+m)$ for the interruptions and $O(m)$ regions for each pursuer via $\mathbf{o}$. Therefore we have at most $(n+m) \times O(mn+m) = O((n+m)(mn+m))$ internal tree nodes.

□

Lemma 14. *A shortest evasive path tree contains $O((n+m)^2(mn+m))$ nodes where m is the number of pursuers and n is the number of obstacle vertices.*

Proof. The tree nodes processed by the algorithm consist of the evader's starting point, some of the obstacle vertices, and the meeting points of the evader and the pursuers. Note that these meeting points are crystal vertices.

We know (by Lemma 13) that there are $O((n+m)(mn+m))$ internal tree nodes. It remains to count the leaf tree nodes. Leaf nodes may be obstacle vertices or meeting points of the evader and a pursuer. The obstacle vertices and the meeting points that could be internal tree nodes are counted in the $O((n+m)(mn+m))$ bound. It remains to count meeting points that can only be leaf tree nodes. These points are meeting points that occur on the axes centred at an internal tree node or the meeting point of the evader and a faster pursuer.

A meeting point on an axis centred at an internal tree node arises from a pursuer intercepting a shortest evasive path that follows the axis from the internal tree node. This is a point of capture that blocks the evader from following the axis any further. Thus there are no other meeting points along this axis centred at this

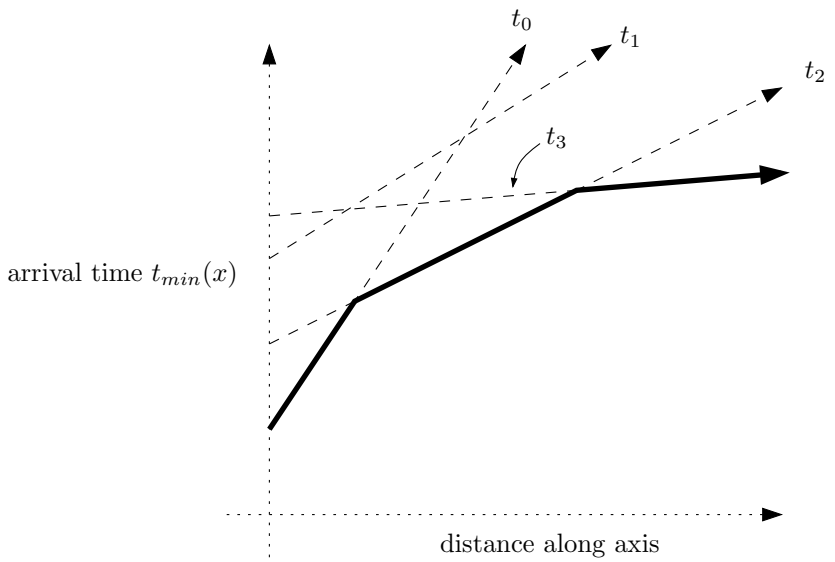


Figure 2.16: The thick black line is an example of the function $t_{min}(x)$ that describes the minimum arrival time of any pursuer the axis of an obstacle vertex or pursuer starting point $\mathbf{o}$, when the pursuers take paths via $\mathbf{o}$. t_0, t_1, t_2 and t_3 are the $t(x)$ arrival time functions for four different pursuers. Note that $t_{min}(x)$ is a continuous piecewise linear function with non-increasing slope, where each pursuer contributes at most one linear piece. Each piece, from left to right, has a smaller slope but a greater arrival time. Note that the pursuer line t_1 does not contribute to $t_{min}(x)$, as that pursuer arrived too late and was too slow.

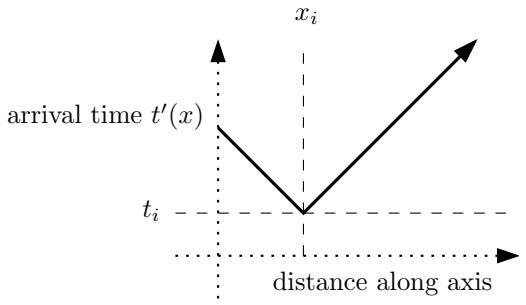


Figure 2.17: An example of the function $t'(x)$ that describes the arrival time of a pursuer p_i along the axis of an obstacle vertex or pursuer starting point $\mathbf{o}$, when the pursuer takes a path via a point $\mathbf{o_i} = (x_i, y_i) \neq \mathbf{o}$. t_i is the arrival time of the shortest obstacle-avoiding path from p_i to the x-axis of $\mathbf{o}$, which occurs at $x = x_i$. The slope is $\pm\frac{1}{v_p}$. Note that the arrival time must be positive.

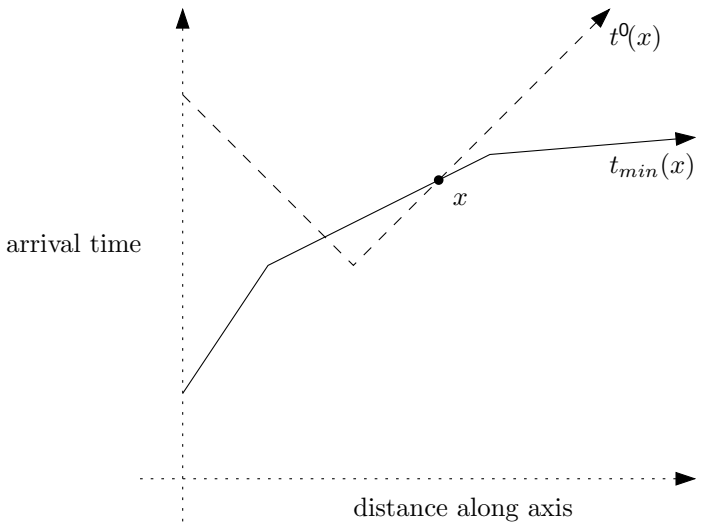


Figure 2.18: An example of when the function $t'(x)$ for a pursuer-point pair $(p_i, \mathbf{o_i})$ intersects the function $t_{min}(x)$ for the point $\mathbf{o}$, where $\mathbf{o} \neq \mathbf{o_i}$. x shows an intersection where we can have a region for $(p_i, \mathbf{o_i})$ to the left of a region for $(p^*, \mathbf{o})$, due to the slope for $t'(x)$ being greater than the slope for $t_{min}(x)$. To the left of the intersection point x, $t'(x)$ has an earlier arrival time than $t_{min}(x)$, indicating a region that does not belong to a pursuer passing through $\mathbf{o}$. To the right of x, $t_{min}(x)$ has the earlier arrival times. Note that $t'(x)$ cannot intersect $t_{min}(x)$ after x, as $t_{min}(x)$ has non-increasing slope.

internal tree node. Since each internal tree node has four axes, the number of these meeting points is at most four times the number of internal tree nodes, which is $O((n+m)(mn+m))$.

A meeting point between the evader and a faster pursuer results in a leaf tree node as once a faster pursuer meets the evader, the evader cannot follow any path from that point that is evasive. We perform the following conservative analysis to upper-bound the number of such interactions.

For each point $\mathbf{o}$ which is either an obstacle vertex or the starting point for a pursuer, we consider how many times a pursuer taking a path via $\mathbf{o}$ can meet an evader along an axis centred at $\mathbf{o}$. The evader must come in a straight line from an internal tree node, $\mathbf{v}$, to such a meeting point. At most two straight-line paths from $\mathbf{v}$ can create meeting points on $\mathbf{p}$'s shortest obstacle avoiding path from $\mathbf{o}$ along this axis. This is because we have the minimum arrival time along the axis by any pursuer via $\mathbf{o}$ described by a continuous, piecewise linear function with positive, non-increasing slope $t_{min}(x)$, and the arrival time of shortest evasive paths from $\mathbf{v}$ described by at most two linear equations. Note that we have at most two intersections described by these equations, and possibly fewer if the evader can be captured before reaching the point of intersection from $\mathbf{v}$ (See Figure 2.19 for examples). Therefore, there are at most $4 \times 2 \times (n+m)(mn+m)(n+m) = O((n+m)^2(mn+m))$ such meeting points.

As a result, there are $O((n+m)^2(mn+m))$ tree nodes. □

We shall refer to T as the number of tree nodes in the complete partition.

Theorem 1. *The complexity of the complete partition for a single evader is* $O((n+1)T^2) = O((n+1)(n+m)^4(mn+m)^2)$. *The complexity of the total partition for the evaders* E *is* $O((n+1)T^2s^2) = O((n+1)(n+m)^4(mn+m)^2s^2)$.

Proof. All the regions computed are regions composed of the crystal vertices on the boundary of the pursuer region, or else they are vertices where a boundary ends against obstacles or vertices on the boundary between evader regions. There are at most T crystal vertices on the boundary of the pursuer region, and therefore at most T^2 edges on this boundary.

Let us now consider the edges created due to obstacles and visibility, and edges separating regions in the complete partition. Each of the tree nodes has a region, and each tree node region can interact with at most every other tree node region. However, Voronoi diagrams between two points clipped by visibility of obstacles are used to generate all the boundaries, and so each boundary is at most $O(n) + O(1)$ in size. Therefore, the complexity of these boundaries is at most $O(T^2 \times (O(n) + O(1))) = O((n+1)T^2)$.

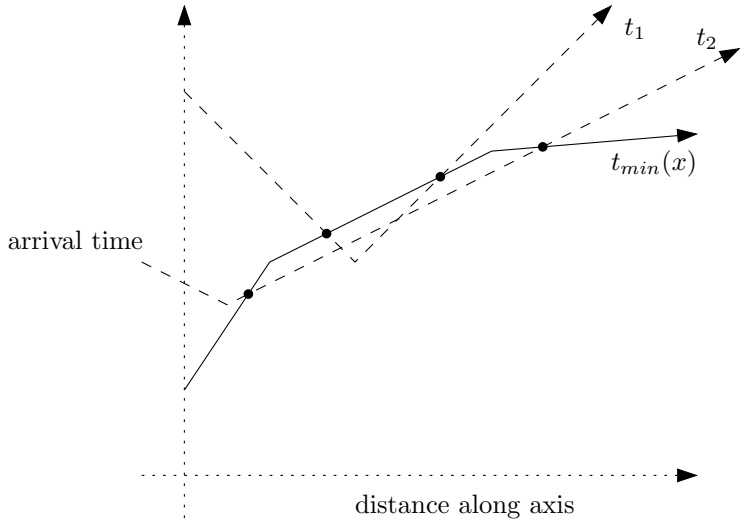


Figure 2.19: This shows two examples (t_1 and t_2) where a function $t'(x)$ for an evader taking a path through an internal tree node (as shown by the lines t_1 and t_2) intercepts a function $t_{min}(x)$ for pursuers going through a point $\mathbf{o}$. In both these examples, there are two interception points. For clarity, the interception points are marked.

Therefore, by Lemma 14, the complexity of the complete partition for a single evader is $O(T^2 + O((n+1)T^2) = O((n+1)T^2) = O((n+1)(n+m)^4(mn+m)^2)$.

If we consider the total partition for a set of s evaders, this is equivalent to considering $s \times T$ tree nodes, and will at most consider all of these tree nodes and their regions. The complexity of the total partition for the evaders E is therefore $O((n+1)(Ts)^2) = O((n+1)(n+m)^4(mn+m)^2s^2)$ □

Note. Note that a partial partition consists of a subset of the regions of the total partition, and so is upper-bounded by the complexity of the complete partition.

2.8.2 Time Complexity of the Algorithm

Let the complexity of the complete partition for an evader be D. Let the complexity of the total partition for a set of evaders be L.

Let $\Phi(T, D)$ be the cost of a geometric union, intersection, difference or visibility operation on T regions with total complexity of at most $O(D)$ edges, with the final partition having complexity of at most $O(D)$ edges. The union of T regions can be performed via T pairwise union steps, where each step is will take at most $O(D \log D)$ time, as each region and the final output is no more complex than $O(D)$

(See the MapOverlay algorithm in Berg et al.[7]). The visible region (also known as the visibility polygon) can be computed in time $O(D \log D)$, as shown by Suri and O'Rourke[22].

Theorem 2. *Computing the complete partition for an evader takes time* $O(mn + m)^3 \times \Phi(T, D)$. *To merge complete partitions takes additional time* $\Phi(sT, L) \times s$.

Proof. To compute the complete partition, the algorithm processes the T tree nodes.

For each iteration of the main loop, a partial partition is computed which is no larger than the complete partition D. Each step of the algorithm considers at set of at most T polygons of complexity at most $O(D)$ and performs geometric union, intersection, difference and visibility. Note the Voronoi diagram computations create two point compoundly weighted Voronoi diagrams which is a constant time operation. Therefore, the complete partition can be computed in time $T \times \Phi(D) = O(mn + m)^3 \times \Phi(D)$

A total partition of the evaders processed involves merging the complete partition of each of the evaders into a diagram which is smaller or equal to the size of the total partition for all the evaders using the same kinds of operations. The total complexity of the polygons in this case is L and the number of we need to consider T polygons for each of the s evaders. Therefore, time $\Phi(sT, L) \times s$ is spent computing the total partition of all the evaders. □

2.9 Extensions and Future Work

2.9.1 Reachable Region for All Pursuers

The pursuit-evasion Voronoi diagram gives regions where only one evader is guaranteed to reach the target location without capture. However, in some applications, we have a set of evaders E and we may wish to guarantee that none of the evaders are captured en route to the destination. We outline Algorithm 4 to compute this *all-evader-reachable region*. Let $\mathcal{D}$ be the set of all complete partitions for the evaders E.

Algorithm 4 All-evader-reachable region modification

for all $\mathcal{E} \in \mathcal{D}$ **do**
  $\mathcal{R}_e := \bigcup_{(\hat{x}, \pi_x, R_x) \in \mathcal{E}} R_x$
  $\mathcal{P} := \mathcal{P} \cap \mathcal{R}_e$
end for

$\mathcal{R}_e$ is the entire region reachable by evader e, and therefore the union of these regions gives us $\mathcal{P}$, the region shared by all evaders.

2.9.2 Bounding the L_2 Pursuit-Evasion Voronoi Diagram

Using the L_1 pursuit-evasion Voronoi diagram, we can compute a region R^+ and a region R^- such that $R^- \subseteq R \subseteq R^+$, where R is the evader region in the L_2 pursuit-evasion Voronoi diagram for a set of evaders E, pursuers P and obstacles O. If we consider $d'(\mathbf{a}, \mathbf{b})$ to be the L_1 distance between $\mathbf{a}$ and $\mathbf{b}$, and $d(\mathbf{a}, \mathbf{b})$ to be the respective L_2 distance, we can note that for any $\mathbf{a}$ and $\mathbf{b}$:

$$d'(\mathbf{a}, \mathbf{b}) \geq d(\mathbf{a}, \mathbf{b})$$
$$\frac{\sqrt{2}}{2} d'(\mathbf{a}, \mathbf{b}) \leq d(\mathbf{a}, \mathbf{b})$$

We shall also refer to ${d_\phi}'(\mathbf{a}, \mathbf{b})$ and $d_\phi(\mathbf{a}, \mathbf{b})$ as the path distance functions in L_1 and L_2 respectively, and similarly for $\overline{d'}(\mathbf{a}, \mathbf{b})$ and $\overline{d}(\mathbf{a}, \mathbf{b})$. Note that by definition, the inequalities above also imply that the following inequalities are true:

$${d_\phi}'(\mathbf{a}, \mathbf{b}) \geq d_\phi(\mathbf{a}, \mathbf{b}) \tag{2.2}$$
$$\overline{d'}(\mathbf{a}, \mathbf{b}) \geq \overline{d}(\mathbf{a}, \mathbf{b}) \tag{2.3}$$
$$\frac{\sqrt{2}}{2} {d_\phi}'(\mathbf{a}, \mathbf{b}) \leq d_\phi(\mathbf{a}, \mathbf{b}) \tag{2.4}$$
$$\frac{\sqrt{2}}{2} \overline{d'}(\mathbf{a}, \mathbf{b}) \leq \overline{d}(\mathbf{a}, \mathbf{b}) \tag{2.5}$$

We can now compute R^+ by underestimating distances with respect to evaders and overestimating distances with respect to pursuers. R^+ is evader-reachable region in the L_1 pursuit-evasion Voronoi diagram with the original pursuers P and modifying the speed of each evader in E by multiplying with the factor $\frac{2}{\sqrt{2}}$.

Theorem 3. *The region R^+ (using the modified pursuit-evasion Voronoi diagram situation in L_1) includes all points in R, the evader-reachable regions for the original pursuit-evasion Voronoi diagram situation in L_2.*

Proof. If a point is in R, by Definition 2.1.2, there must be a path ϕ from the evader to that point such that

$$\forall \mathbf{q} \in \phi, d_\phi(\mathbf{e}, \mathbf{q})/v_e + t_e \leq \min_{f \in P} \overline{d}(\mathbf{f}, \mathbf{q})/v_f + t_f$$

We apply the inequality 2.4 to obtain:

$$\forall \mathbf{q} \in \phi, \frac{\sqrt{2}}{2} d_\phi(\mathbf{e}, \mathbf{q})/v_e + t_e \leq \min_{f \in P} \overline{d}(\mathbf{f}, \mathbf{q})/v_f + t_f$$

$$\forall \mathbf{q} \in \phi, d_\phi(\mathbf{e}, \mathbf{q})/\left(\frac{2}{\sqrt{2}} v_e\right) + t_e \leq \min_{f \in P} \overline{d}(\mathbf{f}, \mathbf{q})/v_f + t_f$$

We then apply the inequality 2.3 to show the following is also true:

$$\forall \mathbf{q} \in \phi, d_\phi(\mathbf{e}, \mathbf{q})/\left(\frac{2}{\sqrt{2}} v_e\right) + t_e \leq \min_{f \in P} \overline{d'}(\mathbf{f}, \mathbf{q})/v_f + t_f \tag{2.6}$$

As Equation 2.6 is the definition of the evasive paths to all the points in the region R^+, if the path ϕ was a valid evasive path in the original L_2 situation, it is also a valid evasive path in the modified L_1 situation. Therefore, any point in R must also be in R^+. □

Conversely, we can compute R^- by overestimating the distances with respect to evaders and underestimating distances with respect to pursuers. Specifically, R^- is the evader-reachable region in the L_1 pursuit-evasion Voronoi diagram using the evaders E and the set of pursuers obtained by modifying the velocity each pursuer in the set of pursuers P by multiplying with the factor $\frac{2}{\sqrt{2}}$.

Theorem 4. *All the points in R^- (using the modified pursuit-evasion Voronoi diagram situation in L_1) are points in R, the evader-reachable regions for the original pursuit-evasion Voronoi diagram situation in L_2.*

Proof. If a point is in the region R^-, there must exist a path ϕ from the evader to that point such that

$$\forall \mathbf{q} \in \phi, {d_\phi}'(\mathbf{e}, \mathbf{q})/v_e + t_e \leq \min_{f \in P} \overline{d'}(\mathbf{f}, \mathbf{q})/\left(\frac{2}{\sqrt{2}} v_f\right) + t_f$$

We can manipulate this equation and apply the inequality 2.5 to obtain:

$$\forall \mathbf{q} \in \phi, {d_\phi}'(\mathbf{e}, \mathbf{q})/v_e + t_e \leq \min_{f \in P} \frac{\sqrt{2}}{2} \overline{d'}(\mathbf{f}, \mathbf{q})/v_f + t_f$$

$$\forall \mathbf{q} \in \phi, {d_\phi}'(\mathbf{e}, \mathbf{q})/v_e + t_e \leq \min_{f \in P} \overline{d}(\mathbf{f}, \mathbf{q})/v_f + t_f$$

We then also apply the inequality 2.2 to obtain:

$$\forall \mathbf{q} \in \phi, d_\phi(\mathbf{e}, \mathbf{q})/v_e + t_e \leq \min_{f \in P} \overline{d}(\mathbf{f}, \mathbf{q})/v_f + t_f \tag{2.7}$$

Therefore, as Equation 2.7 describes a valid evasive path in the original L_2 situation, all the shortest evasive paths in the modified L_1 pursuit-evasion Voronoi diagram situation are also evasive paths in the corresponding L_2 situation. This means that any point in R^- must also be a point in R. □

2.9.3 Other Techniques and Optimisations

We can note that the algorithm described here considers all tree nodes when computing the pursuit-evasion Voronoi diagram. Some of the crystal vertices processed do not have a region associated with them - all shortest evasive paths via these crystal vertices are intercepted by the pursuers (For example, t_4, t_7, t_8 and t_9 in Figure 2.1). Running time could be improved by identifying these nodes before they are processed and removing them from the queue.

As well, we can note that the use of geometric primitive operations of union, intersection, visibility and the two point compoundly weighted Voronoi diagram in our algorithm allows us to show in a straightforward manner that the regions we compute have the desired properties. However, it may be possible to compute the regions and boundaries needed more directly. For example, we compute the region R with the properties needed by Lemma 6 via several intermediate steps — it may be possible to do this more efficiently by computing this region using some other algorithm in a single step.

For example, a commonly used technique for computing boundaries like Voronoi regions are algorithms which follow the boundary of regions to construct the diagrams[20]. As the queue in our algorithm also represents the boundary, it may prove possible to adapt such techniques to more efficiently fill the queue with tree nodes by performing a boundary following operation as opposed to computing entire Voronoi diagram regions. Once the tree nodes have been discovered, we could perform Voronoi diagram operations to obtain the regions for each tree node. However, it may be necessary to subdivide the plane into the Voronoi regions for the pursuer sites, which would involve computing the constrained compoundly weighted Voronoi diagram, which may not have an efficient solution.

Another technique often used in computing Voronoi diagrams is to consider relationships with other geometric constructions. For example, in computing the multiplicatively weighted Voronoi diagram, Aurenhammer and Edelsbrunner[5] consider an embedding in three dimensions. Similarly, we have the classic Voronoi Diagram/Delaunay triangulation duality. However, the obstacle and capture avoiding path constraint as well as the asymmetrical nature of the sites may make it difficult to find such a relationship.

One strength of this work is the fact that the algorithm can consider obstacles, additive and multiplicative weight while utilising only relatively simple primitive operations — we use geometric union, difference and intersection. We also use the compoundly weighted Voronoi diagram, but only considering two points. Simplifications such as removing the obstacles, considering pursuers of with the same speed or only considering one pursuer could also make it possible to apply different

techniques or compute the diagram more efficiently. For example, when all pursuers have the same speed, the number of internal tree nodes by the analysis in Lemma 13 is reduced to $O(n+m)$, making the total tree nodes in Lemma 14 $O((n+m)^2)$.

2.9.4 Future Work

The pursuit-evasion Voronoi diagram could be extended to cover the concept of *airlift distance*[1]. In this case, we add to the input an arbitrary graph A on c points in the plane which represent airports. The edge weights in the graph represent flight durations, and need not fulfil the triangle inequality. In addition to moving normally through the plane, paths may also enter and exit the graph via the airports. We note that the pursuit-evasion Voronoi diagram can be extended to include airlift distance by considering reachable points in A as potential tree nodes, and considering two points connected by an edge in A to be visible from one another. This would allow modelling subways and other high-speed transit systems often found in a city setting.

Another interesting extension would be to consider computing the pursuit-evasion Voronoi diagram in L_2 directly, and to determine if methods could be adapted to compute the pursuit-evasion Voronoi diagram in polynomial time exists in this case. L_2 is more directly applicable to situations such as geographic spread outside of populated regions, where the L_1 movement restrictions are not in place. Again, we note that a direct application of our algorithm is impractical in this case, as our current representation cannot deal with the boundaries generated in an efficient manner. Schaudt and Drysdale[21] were forced to resort to numerical techniques when computing multiplicatively weighted crystal Voronoi diagrams in L_2, which indicates that similar difficulties may arise for the pursuit-evasion Voronoi diagram in L_2.

A natural extension of this work is to consider metrics whose unit circles are arbitrary convex polygon distance functions. The L_1 metric is the special case where the convex polygon is a diamond. If, considering two points, the Voronoi diagram and the pursuit-evasion Voronoi diagram for a particular distance function are polygons with finite complexity, it should be possible to adapt the algorithm to compute the pursuit-evasion Voronoi diagram for that distance function. Schaudt[20] notes that the complexity of the boundaries for the multiplicatively weighted crystal Voronoi diagrams are polygonal for all convex polygon distance functions, which indicates that we may be able to achieve a similar result for the pursuit-evasion Voronoi diagram. If this is possible, we could use convex polygon distance functions to obtain better bounds on the L_2 distance function, in a similar manner to the method mentioned previously using the L_1 pursuit-evasion Voronoi diagram. Another potentially related direction for extension would be to consider

polygonal pursuers and evaders, and considering the Hausdorff distance functions and working in configuration space. In their work on multiplicatively weighted crystal Voronoi diagrams, Schaudt and Drysdale[21, 20] consider both convex polygon distance functions and L_2 distance, although in the absence of obstacles.

We can also ask the decision problem: Given a set of pursuers, an evader and obstacles, does there exist an evasive path that can reach a particular point? The pursuit-evasion Voronoi diagram can be computed to answer this problem, and can terminate early if the query point falls within a reachable region — in this case we know a shortest path exists. As well, we can also terminate early if, during the course of the algorithm, the point to be processed has a shortest evasive path with an arrival time after that of the arrival time of the pursuer at the query point. In this case, no shortest evasive path can reach the query point before the pursuer. In the worst case, we may need to process all the tree nodes for the evaders, and therefore we will have effectively computed the complete partition for the evader. However, it may be possible to prune the shortest path tree to achieve a better time complexity, or to search backwards from the query point for an evasive path.

2.10 Conclusion

In this chapter, we describe the pursuit-evasion Voronoi diagram, which considers a set of pursuers, a set of evaders and obstacles in the plane. The pursuit-evasion Voronoi diagram computes all points reachable by the evaders via paths which are not intercepted by pursuers. We also give an algorithm to compute this diagram, which proceeds in a fashion similar to Dijkstra's algorithm, exploring shortest paths in the plane and computing partial diagrams that describe points reachable by evasive paths in the plane. We show the algorithm we describe runs in polynomial time when we use the L_1 distance metric.

Chapter 3

Visibility Constrained Pursuit-Evasion

3.1 Introduction to Visibility Constrained Pursuit

Suppose a cop, the pursuer, is chasing a robber, the evader, around a large bank. If the pursuer ever loses sight of the evader, the pursuer loses. If the pursuer ever touches the evader, the pursuer wins. A third possibility, the one we explore in this chapter, is a tie situation where the pursuer neither wins nor loses. The pursuer can always keep the evader in sight, but can never succeed in intercepting the evader. We call this situation a tie. Like the pursuit-evasion Voronoi diagram, we assume that the pursuer and the evader each have their own constant maximum speeds. We also assume that the pursuer wants to win if possible and the evader wants the pursuer to lose if possible.

In this example, we add a new constraint to the pursuit-evasion dynamic. In addition to all paths being unable pass through obstacles and paths taken by the evader being forced to avoid the pursuer, the pursuer is now restricted to paths that keep the evader in sight. The positions which the pursuer can occupy are no longer independent of the position the evader can occupy.

3.2 Related Work

Searching for an arbitrarily fast-moving evader in a polygonal region is another common form for the visibility-based pursuit-evasion problem. Park et al.[18] have recently provided a solution for this problem when it involves a single pursuer. The evader starts off out of sight from the pursuer, and the goal for the pursuer is to establish line-of-sight to the evader. The pursuer has finite speed, whereas the

evader is arbitrarily fast. Both the pursuer and the evader move in a polygonal region whose boundary occludes visibility. The pursuer then needs to find a path that guarantees the capture of the evader. This situation can be viewed as "worst-case" from the point of view of the pursuer — we assume that if there exists a path for the evader to avoid capture, the evader will be able to take that path.

LaValle et al. [13] discuss the problem where a moving observer attempts to maintain visibility to a moving target in an environment where motion may be restricted to certain regions and visibility may be obscured in certain configurations. They detail a method that determines minimal distance paths for the observer in the case of a known target path, and suggest strategies in the case where the target moves unpredictably.

They motivate this problem as being different from previous target tracking problems, as their focus is finding a path for the moving observer to allow visibility to be maintained with the target, as opposed to the usual computer vision problem of target recognition and tracking. In this case, the target corresponds to the evader and the observer corresponds to the pursuer. As well, they consider more flexible objectives for the pursuer while obeying the visibility constraint. For example, the pursuer may wish to capture the target, described as minimising the distance to the target, or may wish instead to minimise the distance travelled by the pursuer.

They compute optimal paths for the observer by discretising space and time, which reduces the infinite possible locations of the observer and the target to a discrete set of positions, in effect changing this problem to visibility-based pursuit-evasion in a graph. They also consider various cost functions, such as allowing the observer to lose visibility with the target while incurring a cost penalty.

González-Baños et al.[10] consider the same problem of target tracking. They extend the problem to partially predictable targets, where the maximum speed of the target is known, but the motion strategy for the target is not. They modify the target planner to optimise the motion strategy over a finite set of time steps, considering a probabilistic distribution for the movement of the target.

Alon Efrat et al.[8] consider the visibility-based pursuit-evasion problem in a general polygonal environment and for polygonal targets, while considering paths of finite length for the evader. They provide a method for constructing a type of optimal path that attempts to capture the evader if possible and otherwise minimises the distance to the evader. They show the possibility of "leaning curves" in the optimal path, which can occur when the edge of an occluding boundary forces the shortest path for the pursuer to follow a curved path around an obstacle vertex to keep the evader in sight. They show a method for approximating these leaning curves, but were unable to show a general closed form solution. This algorithm can

be applied to paths of infinite length, such as the evader path we consider in this chapter, but the algorithm will not terminate if the pursuer never loses sight of the evader and cannot capture. We shall prove in this chapter that this situation can occur.

3.3 Visibility Constrained Pursuit-Evasion

Let us consider the pursuer and the evader to be two points, $\mathbf{A}(t)$ and $\mathbf{B}(t)$, parameterised by time $t \geq 0$, moving continuously in the obstacle-free part of the plane with $\mathbf{A}(0) = \mathbf{a}$ and $\mathbf{B}(0) = \mathbf{b}$. We can scale the unit for time such that, without loss of generality, the speed of the pursuer, $\frac{d\mathbf{A}(t)}{dt}$, is at most v_a, and the speed of the evader is at most 1. We say the pursuer sees the evader at time t if the line segment between $\mathbf{A}(t)$ and $\mathbf{B}(t)$ is uninterrupted by obstacles, and we say the pursuer cannot see the evader if this line segment intersects an obstacle of the polygonal scene.

Given a polygonal scene, the starting positions $\mathbf{a}$ and $\mathbf{b}$ for the pursuer and the evader, the maximum speed v_a and 1 for the pursuer and the evader, and the path $\mathbf{B}(t)$ for the evader, the question is does there exist a path $\mathbf{A}(t)$ the pursuer can follow to capture the evader and thereby win.

Definition 3.3.1. A *capture path* $\mathbf{A}(t)$ is an obstacle avoiding path with $\mathbf{A}(0) = \mathbf{a}$ such that there exists a time $\tau \geq 0$ where $\mathbf{A}(\tau) = \mathbf{B}(\tau)$ and $\mathbf{A}(t)$ sees $\mathbf{B}(t)$ for all $t \leq \tau$. A *viewing path* $\mathbf{A}(t)$ is an obstacle avoiding path with $\mathbf{A}(0) = \mathbf{a}$ such that $\mathbf{A}(t)$ sees $\mathbf{B}(t)$ for all $t \geq 0$.

We say the pursuer *wins* if there exists a capture path for the pursuer. We say the pursuer *loses* if for all paths the pursuer can take, there exists a time when the pursuer cannot see the evader. We say the pursuer *ties* if there exists a path such that the pursuer can always see the evader but the pursuer cannot capture the evader.

Note that this simplifies the problem significantly from the pursuit-evasion Voronoi diagram. This is similar to answering whether a specific evader path is ever intercepted by the pursuer, as opposed to describing all possible evasive paths.

Definition 3.3.2. A path $\mathbf{A}(t)$ is an *optimal viewing path* if, given the pursuer's starting position and speed, $\mathbf{A}(t)$ is the shortest path for the pursuer that is also a viewing path.

Definition 3.3.3. A path $\mathbf{A}(t)$ is an *optimal capture path* if, given the pursuer's starting position and speed, $\mathbf{A}(t)$ is the shortest path for the pursuer that is also a capture path.

In general, we say that both optimal capture paths and optimal viewing paths are optimal paths, and share similar properties.

Orbiting a Regular Unit-Sided Polygon

Let us now consider the polygonal obstacle P to be a regular convex polygon with n sides, each side with length 1. Let the vertices be labelled $< \mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_{n-1}} >$. The pursuer cannot see the evader at time t if and only if the line segment connecting $\mathbf{A}(t)$ with $\mathbf{B}(t)$ properly intersects P. As well, we shall refer to a general vertex $\mathbf{p_k} = \mathbf{p_{k \bmod n}}, k \in \mathbb{N}$ and $k \geq n$.

Let us consider a path taken by the evader, $\mathbf{B}(t)$, where the speed $\frac{dB}{dt}$ is 1 and the motion of the path takes the evader clockwise following the perimeter of P, starting with $\mathbf{B}(0)$ at vertex $\mathbf{p_0}$ of P. Therefore, for $k \in \mathbb{Z}$, $\mathbf{B}(k)$ lies on a vertex of P.

Definition 3.3.4. The *sight line* $S(i)$ for a polygon P with the vertices in clockwise order $< \mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_{n-1}} >$ is the ray $\overrightarrow{\mathbf{p_{i+1}}\mathbf{p_i}}$, for $0 \leq i < n-1$. $S(n-1)$ is $\overrightarrow{\mathbf{p_0}\mathbf{p_{n-1}}}$. For $i \geq n$, the sight line $S(i)$ is the same as sight line $S(i \bmod n)$

Definition 3.3.5. A point $\mathbf{p}$ is at a height h along a sight line $S(i)$ if it lies at a distance h from $p_{i \bmod n}$ away from the polygon along the sight line $S(i)$.

Let $\mathbf{A}(t)$ be an optimal path for the pursuer. $\mathbf{A}(0)$ starts at a height a along the sight line $S(0)$. We shall only consider speeds $v_a > 1$, otherwise it is impossible for the pursuer to capture the evader.

Lemma 15. *When the evader follows a path along the boundary of a convex polygonal obstacle, an optimal path for the pursuer only turns at time* $\mathbf{A}(k)$, $k \in \mathbb{N}$ *(i.e. the pursuer only turns when the evader turns).*

Proof. Suppose there exists a path $\mathbf{A}(t)$ where at a time $k+\epsilon$, $0 < \epsilon < 1$, the pursuer changes direction. However, as the path $\mathbf{A}(t)$ is an optimal path, for $k \leq t \leq k+1$, $\mathbf{B}(t)$ lies on the edge of the polygon $\overline{p_k p_{k+1}}$. Therefore, all the points for the path $\mathbf{A}(t)$ for $k \leq t \leq k+1$ must lie on the halfplane formed by $\overline{p_k p_{k+1}}$ away from the polygon, otherwise they would lose sight of the evader. Therefore, we can create a new path $\mathbf{A'}(t)$ such that $\mathbf{A'}(t) = \mathbf{A}(t)$ for $0 \leq t \leq k$ and for $t > k+1$. For the path $\mathbf{A'}(t)$ at time $k < t \leq k+1$, we can follow a straight line path from $\mathbf{A}(k)$ to $\mathbf{A}(k+1)$, rather than turning at $\mathbf{A}(k+\epsilon)$ (See Figure 3.2). This makes the path $\mathbf{A'}(t)$ shorter than $\mathbf{A}(t)$, therefore, $\mathbf{A}(t)$ cannot be an optimal path. □

Note. Lemma 15 implies that shortest paths will be straight line segments from time $k \in \mathbb{N}$ to time $k+1$.

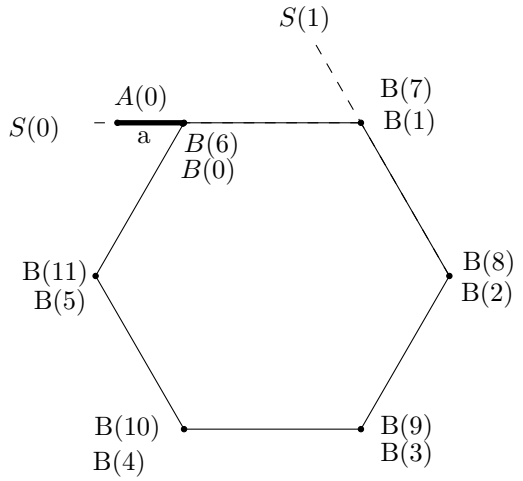


Figure 3.1: The starting configuration when P is a regular hexagon (n=6).

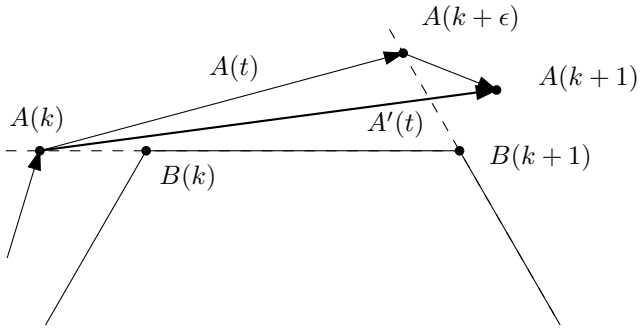


Figure 3.2: An optimal path can only turn at the time k where $k \in \mathbb{N}$. If $A(t)$ is an optimal path that turns at a time $A(k)$ where k is not in $\mathbb{N}$, we can construct a shorter optimal path $A'(t)$.

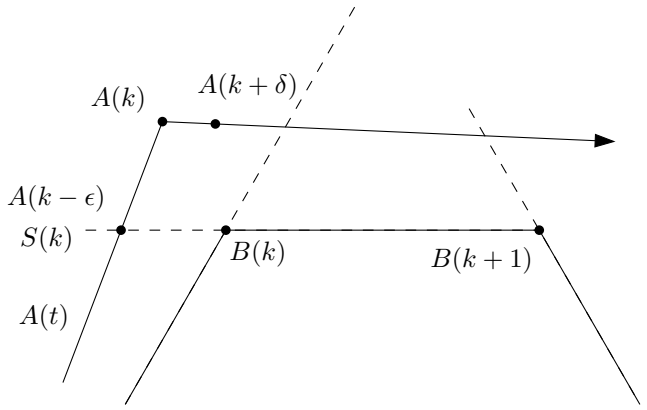


Figure 3.3: This shows an example situation where the optimal path for A has does not turn on a sight line $S(k)$ at time k.

Lemma 16. *Whenever an optimal path $\mathbf{A}(t)$ turns on a sight line at a time $k \in \mathbb{N}$, the turn $\mathbf{A}(k)$ occurs on sight line $S(k)$.*

Proof. Let us consider an optimal path $\mathbf{A}(t)$ where a turn $\mathbf{A}(k)$ does not occur on sight line $S(k)$ at time $k \in \mathbb{N}$. Let us consider the first time k when this occurs. As the evader, for time $k \leq t \leq k+1$, lies on the polygon edge $\overline{p_k p_{k+1}}$, $\mathbf{A}(t)$ must lie in the halfplane formed by $\overline{p_k p_{k+1}}$ away from the polygon during this time interval. As $\mathbf{A}(k-1)$ lies on sight line $S(k-1)$ and $\mathbf{A}(k)$ does not turn on sight line $S(k)$, $\mathbf{A}(k)$ must turn at some point past the light line $S(k)$ in this halfplane, and cannot lie on the extension of the line $\overline{p_{k-1} p_k}$ (otherwise, the pursuer could have caught the evader instead). Therefore, let us choose a time $k+\delta$ such that $0 < \delta < 1$ and $\mathbf{A}(k+\delta) \neq \mathbf{A}(k)$ and $\mathbf{A}(k+\delta)$ lies in the halfplane formed by $\overline{p_{k-1} p_k}$ away from the polygon. This must occur as the pursuer is in this halfplane at time k, but must leave this halfplane to cross sight line $S(k+1)$ by time $k+1$. As well, there must exist a time $t = k-\epsilon$, $0 < \epsilon < 1$ where $\mathbf{A}(t)$ crosses the sight line $S(k)$ (See Figure 3.3).

Now, we can create a shorter path $\mathbf{A}'(t)$, where $\mathbf{A}'(t) = \mathbf{A}(t)$ for $0 \leq t \leq k-\epsilon$ and $t \geq k+\delta$, and where $\mathbf{A}'(t)$ takes a straight line path from $\mathbf{A}(k-\epsilon)$ to $\mathbf{A}(k+\delta)$ for the time $k-\epsilon < t < \delta$ (See Figure 3.4). This new portion of path is valid, as the entire path can see the evader on the edge $\overline{p_{k-1} p_k}$ and the edge $\overline{p_k p_{k+1}}$. As $\mathbf{A}'(t)$ does not turn along the section of path between times $k-\epsilon$ and $k+\delta$, unlike the original path $\mathbf{A}(t)$ which turned at time k, $\mathbf{A}'(t)$ must be a shorter optimal path, which is a contradiction. □

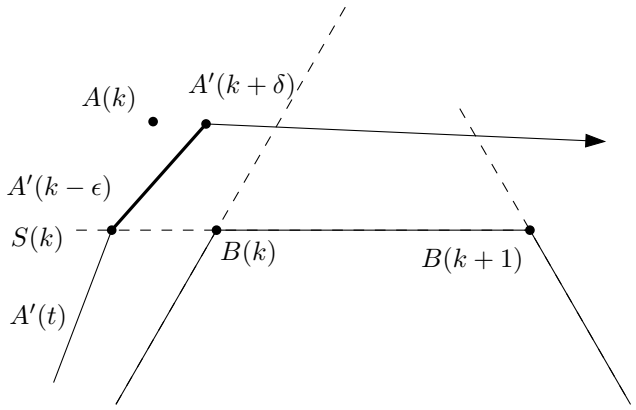


Figure 3.4: We can replace the original path with a shorter path that goes from $A(k-\epsilon)$ directly to $A(k+\delta)$, which removes the turn p.

Definition 3.3.6. Let the path $\mathbf{A}^*(t)$ be the path where $\mathbf{A}^*(0)$ starts at a height a on sight line $S(0)$. For each $i = 1, \ldots, k$, $\mathbf{A}^*(i)$ is the lowest point on sight line $S(i)$ that can be reached from $\mathbf{A}^*(i-1)$ without losing sight of the evader, and k is the lowest integer such that there exists $\epsilon, 0 \leq \epsilon < 1$ where $|\overline{S(k)\mathbf{B}(k+\epsilon)}| = v_a \times \epsilon$. In an optimal capture path, $k \neq \infty$, and $\mathbf{A}^*(k+\epsilon) = \mathbf{B}(k+\epsilon)$.

3.3.1 The Orbital Path for the Pursuer

We can now construct a path for the pursuer that allows her to keep the evader in sight without catching the evader as follows (Figure 3.5): At each time $k \in \mathbb{Z}$, the pursuer will be at the point at height a on the sight line $S(k)$. To follow this path and keep the evader in sight, the pursuer must have a speed

$$v_{critical} = \sqrt{(1+a)^2 + a^2 - 2a(1+a)\cos(2\pi/n)}.$$

As seen in Figure 3.6, $v_{critical}$ is the length of the third edge in a triangle that has edges a and $1+a$, with the angle $\frac{2\pi}{n}$ between them, solved by applying the Law of Cosines.

This path allows the pursuer to keep the evader in sight, but does not allow the pursuer to catch the evader. As we can see by rotational symmetry, the relative positions of the pursuer, evader and the obstacle are the same at time $k+1$ as they were at time k for all $k \in \mathbb{N}$. We call this path the *orbital path* for the pursuer.

Note. For regular polygons with less than five sides (squares and equilateral triangles), $v_{critical} > 1 + a$ since $\cos(2\pi/n) \leq 0$. This means the velocity required to

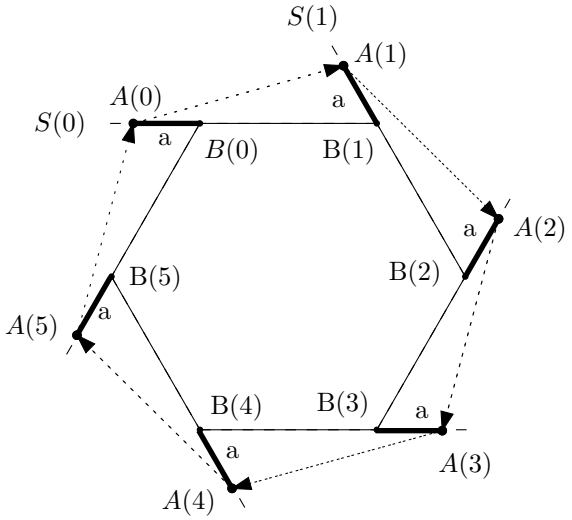


Figure 3.5: Orbital path for the pursuer. The pursuer can always keep the evader in sight, but can never catch the evader.

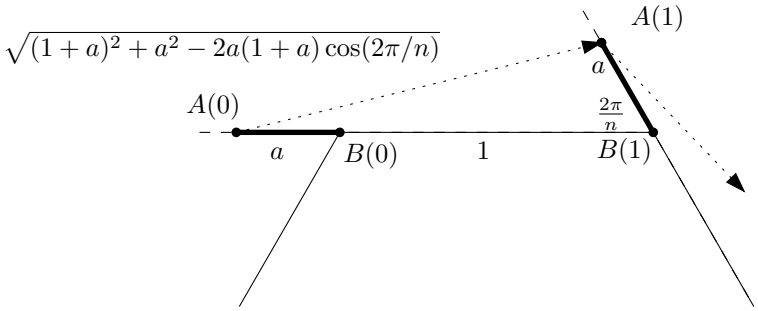


Figure 3.6: A close-up of the path for the pursuer to show $v_{critical}$

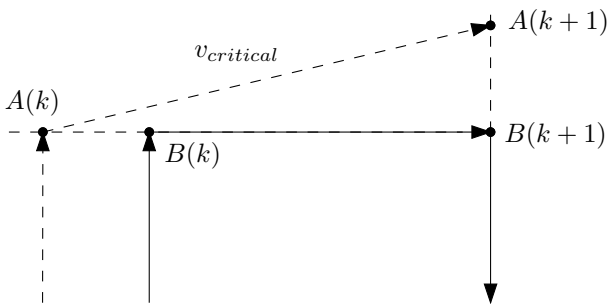


Figure 3.7: When $\frac{2\pi}{n} \geq \frac{\pi}{2}$, the distance $v_{critical}$ is longer than the distance to capture the evader directly. For example, when $n = 4$ (square), the distance from $\mathbf{A}(k)$ to $\mathbf{A}(k+1)$ is longer than the distance from $\mathbf{A}(k)$ to $\mathbf{B}(k+1)$.

follow the orbital path is greater than the velocity needed to simply move directly at the evader and capture. In this case, a pursuer able to follow the orbital path can always capture the evader as well. (See Figure 3.7).

Note that $\angle S(k)\mathbf{p_{k+1}}S(k+1) = \frac{2\pi}{n}$ for $k \in \mathbb{N}$. For a given v_a and position $\mathbf{A}^*(k)$, we can solve for the lowest reachable position $\mathbf{A}^*(k+1)$. Let x be the height of $\mathbf{A}^*(k)$ and let x' is the height of $\mathbf{A}^*(k+1)$. We use the Law of Cosines and obtain the minimum value for x':

$$x' = (x+1)\cos\left(\frac{2\pi}{n}\right) - \sqrt{{v_a}^2 - (x+1)^2\sin^2\left(\frac{2\pi}{n}\right)} \tag{3.1}$$

Definition 3.3.7. We define $\delta(x) = x - x'$. $\delta(x)$ is therefore the decrease in distance in relative position between the pursuer and the evader at time $k+1$ versus time k where $k \in \mathbb{N}$ and $k+1$, when the pursuer follows the path $\mathbf{A}^*(t)$.

$\delta(x)$ is only defined for $x \geq v_a - 1$. When $x \leq v_a - 1$, the pursuer can capture the evader by the time $k+1$ by moving straight towards the pursuer along $S(k)$, and will not need to reach the subsequent sight line.

$\delta(v_a - 1) = v_a - 1$, as the pursuer can capture the evader at time $k+1$ at $\mathbf{B}(k+1)$, at a height of 0 along $S(k+1)$. We shall also only consider $\delta(x)$ when $x \leq \frac{v_a}{\sin\left(\frac{2\pi}{n}\right)} - 1$. For values of $x > \frac{v_a}{\sin\left(\frac{2\pi}{n}\right)} - 1$, the pursuer, starting the time k, is unable to reach any point on the sight line $S(k+1)$ before the time $k+1$. Therefore, we only consider $\delta(x)$ is only relevant when $v_a - 1 \leq x \leq \frac{v_a}{\sin\left(\frac{2\pi}{n}\right)} - 1$.

Lemma 17. *For $v_a - 1 \leq x < \frac{v_a}{\sin\left(\frac{2\pi}{n}\right)} - 1$, $\delta(x)$ is concave down. Therefore, there exists a minimum value δ_{min} for $\delta(x)$, which occurs at either the lowest or the highest height achieved.*

Proof. Let us consider the second derivative for $\delta(x)$:

$$\frac{d^2\delta}{dx^2}(x) = -\frac{(x+1)^2 \sin^4\left(\frac{2\pi}{n}\right)}{({v_a}^2 - (x+1)^2 \sin^2\left(\frac{2\pi}{n}\right))^{\frac{3}{2}}} - \frac{\sin^2\left(\frac{2\pi}{n}\right)}{\sqrt{{v_a}^2 - (x+1)^2 \sin^2\left(\frac{2\pi}{n}\right)}}$$

When $x \geq \frac{v_a}{\sin\left(\frac{2\pi}{n}\right)} - 1$, ${v_a}^2 \leq (x+1)^2 \sin^2\left(\frac{2\pi}{n}\right)$ and $\frac{d^2\delta}{dx^2}(x)$ is either undefined or imaginary. Otherwise, $\frac{d^2\delta}{dx^2}(x)$ is real and negative. Therefore, $\delta(x)$ is concave down as the second derivative $\frac{d^2\delta}{dx^2}(x) < 0$ in the range $v_a - 1 \leq x < \frac{v_a}{\sin\left(\frac{2\pi}{n}\right)} - 1$. □

Theorem 5. *When the evader follows a path along the boundary of a regular convex polygonal obstacle and $\cos\left(\frac{2\pi}{n}\right) \geq a/(1+a)$, whether v_a is greater, less than or equal to $v_{critical}$ determines if the pursuer wins, loses or ties (respectively).*

When the evader follows a path along the boundary of a regular convex polygonal obstacle and $\cos\left(\frac{2\pi}{n}\right) < a/(1+a)$, if $v_a < (1+a)\sin\left(\frac{2\pi}{n}\right)$ the pursuer will lose, otherwise the pursuer will win.

Proof. There are two cases to consider. If $\cos\left(\frac{2\pi}{n}\right) \geq a/(1+a)$, the pursuer will win, lose or tie depending on a and v_a. Otherwise, if $\cos\left(\frac{2\pi}{n}\right) < a/(1+a)$ then for all v_a, the pursuer cannot tie - the pursuer wins or the pursuer loses.

Case. $\cos\left(\frac{2\pi}{n}\right) \geq a/(1+a)$:

In this case, when the pursuer starts at a height a along sight line $S(0)$, the closest point reachable on sight line $S(1)$ from this starting point is higher than or equal to a (See Figure 3.8 for an example).

1. *If $v_a = v_{critical}$, the pursuer ties, but does not win.*

   More specifically, we show that when $v_a = v_{critical}$, there does not exist any capture paths for the pursuer, but there does exist a path where the pursuer will never lose sight of the evader.

   When $v_a = v_{critical}$, the height of the pursuer at each sight line is a, as $x = x' = a$. Therefore, $\mathbf{A}^*(t)$ is the orbital path for the pursuer, and does not capture the evader. Therefore, the evader can tie.

   Suppose there exists a capture path for the pursuer. Let $\mathbf{A}(t)$ be the optimal capture path. As $\mathbf{A}^*(t)$ is not a capture path, there must exists time $k \in \mathbb{N}$ (by Lemma 15) where $\mathbf{A}(t)$ diverges from $\mathbf{A}^*(t)$. By the definition of $\mathbf{A}^*(t)$,

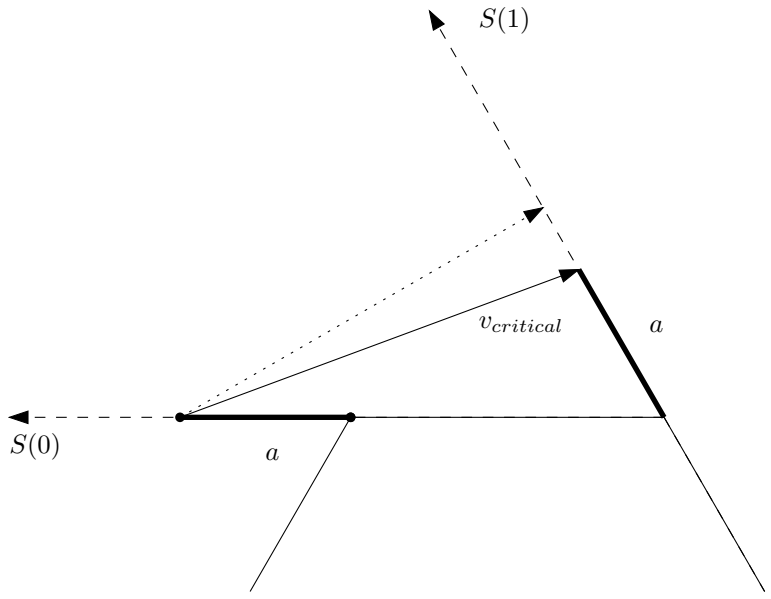


Figure 3.8: When $\cos\left(\frac{2\pi}{n}\right) \geq a/(1+a)$, the shortest reachable point from the pursuer starting point is higher than or equal to the point reachable with speed $v_{critical}$. The dashed line shows the path to the point on sight line $S(1)$ closest to the pursuer starting point.

when $\mathbf{A}(t)$ intersects sight line $S(k+1)$, it must do so at a height higher than $\mathbf{A}^*(k+1)$. Let us consider the time $k+i$ (for $i > 1, i \in \mathbb{N}$) when the path $\mathbf{A}(t)$ turns. By Lemma 16, this must occur on sight line $S(k+i)$. However, as the path $\mathbf{A}^*(t)$ forms a regular convex polygon and $\mathbf{A}(t)$ diverged away from the polygon, the height of $\mathbf{A}(k+i)$ must be greater than the height of $\mathbf{A}^*(k+i)$.

We can now construct a path $\mathbf{A}'(t)$, where $\mathbf{A}'(t) = \mathbf{A}^*(t)$ for $0 \leq t \leq k+i$. After the time $k+i$, $\mathbf{A}'(t)$ will follow a path parallel to $\mathbf{A}(t)$, turning on the same sight lines at the same time. However, as $\mathbf{A}'(k+i)$ is at a height lower than $\mathbf{A}(k+i)$, the distance travelled by $\mathbf{A}'(t)$ is strictly less than $\mathbf{A}(t)$ for $k \leq t \leq k+1$, and will always intersect sight lines at a lower point.

We can note that this is true if we consider any pair of consecutive sight lines j and $j+1$, for time $t \geq k+1$. If we know that $\mathbf{A}'(j)$ intersects $S(j)$ at a point lower than $\mathbf{A}(t)$, and $\mathbf{A}'(t)$ follows a course parallel $\mathbf{A}(t)$, then if we consider the triangle formed by sight lines $S(j)$ and $S(j+1)$ with the two paths, the length of the segment of $\mathbf{A}(t)$ in the triangle must be larger than the length of $\mathbf{A}'(t)$, and intersects $S(j+1)$ at a higher point.

As well, since every segment of the path for $\mathbf{A}'(t)$ between sight lines is strictly shorter than the path for $\mathbf{A}(t)$ between sight lines, $\mathbf{A}'(t)$ has less distance to travel between sight lines than $\mathbf{A}(t)$. Therefore $\mathbf{A}'(t)$ can choose to cross sight lines at exactly the same time as $\mathbf{A}(t)$, and so if $\mathbf{A}(t)$ can keep the evader in sight, then so can $\mathbf{A}'(t)$ (See Figure 3.9).

After crossing the last sight line before capture, the path $\mathbf{A}'(t)$ will aim to capture the evader. As $\mathbf{A}'(t)$ starts at a lower height along the same sight line as $\mathbf{A}(t)$, and both follow a straight line path along this sight line to capture the evader, $\mathbf{A}'(t)$ will capture the evader before $\mathbf{A}(t)$ while travelling less distance. Therefore, we have shown the path $\mathbf{A}'(t)$ is shorter than the path $\mathbf{A}(t)$ from the time $t = k+i$ until capture.

However, we can note that at time $k+i$, $\mathbf{A}'(t)$ is at height a on sight line $S(k+i)$. Therefore, we can construct a path $\mathbf{A}''(t)$ that follows the path given by $\mathbf{A}'(t), t \geq k+i$, but starts following this path on sight line $S(0)$ at time 0. This path $\mathbf{A}''(t)$ must be shorter than the path $\mathbf{A}'(t)$ and captures the evader. This implies that $\mathbf{A}''(t)$ is shorter than $\mathbf{A}(t)$. As this is a contradiction, the pursuer cannot win. Therefore, the pursuer can only tie.

2. *If $v_a > v_{critical}$, the pursuer wins.*

   More specifically, we show that if $v_a > v_{critical}$, there exists a path for the pursuer such that the pursuer captures the evader.

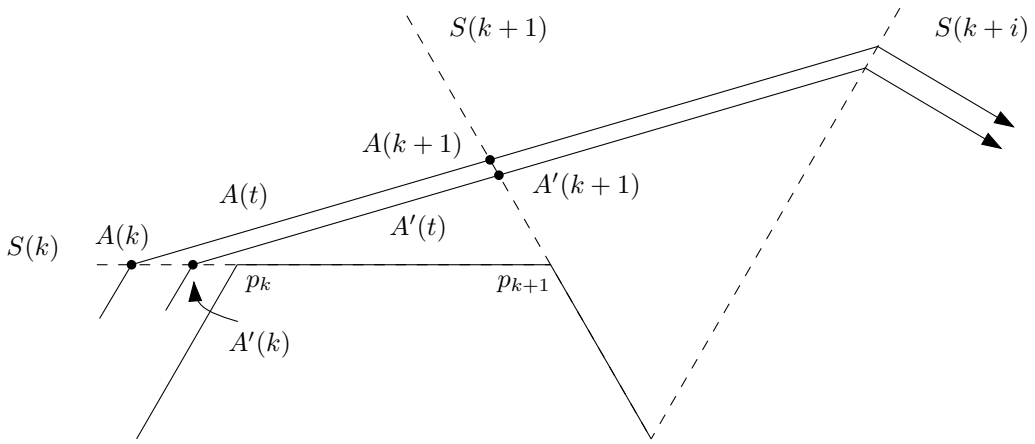


Figure 3.9: The path $A'(t)$ is parallel to the path $A(t)$ between sight lines, but starts at a lower height at time k. Therefore, between any pair of successive sight lines, $A'(t)$ will be shorter than $A(t)$. The path $A'(t)$ can then cross sight lines at the same time as $A(t)$, and $A'(t)$ will be between a sight line $S(j)$ and $S(j+1)$ during the same time as $A(t)$. Note that if we consider k as a time when $A(t)$ turns, when the pursuer passes sight line $S(k+j)$, she will be able to see the evader along the edges $\overline{p_k p_{k+1}}, \overline{p_{k+1} p_{k+2}}, \ldots, \overline{p_{k+j-1} p_{k+j}}$. This is true for j up until $j = i$ where $S(k+i)$ is the sight line where $A(t)$ turns next, and so between any two successive sight lines, if $A(t)$ can keep the evader in sight, then $A'(t)$ can keep the evader in sight.

We note then when $v_a > v_{critical}$, $x' < x$ for the starting time 0 and so $\delta(x) > 0$. The pursuer can also reach the sight line $S(1)$ from $S(0)$, so $a \leq \frac{v_a}{\sin(\frac{2\pi}{n})} - 1$. We note that by Lemma 17 that the minimum δ_{min} for $\delta(x)$ in the range $v_a - 1 \leq x \leq a \leq \frac{v_a}{\sin(\frac{2\pi}{n})} - 1$ occurs either at $x = v_a - 1$ or $x = a$, if the height never exceeds a. We can then note that $\delta_{min} = \delta(a) > 0$, or else $\delta_{min} = \delta(v_a - 1) = v_a - 1 > 0$, and so δ_{min} in the range $v_a - 1 \leq x \leq a$ is always positive. Therefore, the pursuer is always decreasing in height relative to the evader. Thus the pursuer must catch the evader in at most $\lceil \frac{a - v_a + 1}{\delta_{min}} \rceil + 1$ steps. This is because the path will take at most $\lceil \frac{a - v_a + 1}{\delta_{min}} \rceil$ steps to reach the state $x \leq v_a - 1$ and then a final step to capture the evader.

3. *If $v_a < v_{critical}$, the pursuer loses.*

   More specifically, when $v_a < v_{critical}$, there does not exist any viewing or capture paths for the pursuer.

   Let us look at the function $\delta(x)$. For $x = a$, we know that $\delta(x)$ must be negative as $v_a < v_{critical}$ — the pursuer is too slow to maintain the same height and must go to a greater height. However, if we choose a height h sufficiently small, v_a will be the orbital speed for the height h, and so $\delta(h)$ will be 0. Therefore, the slope of $\delta(x)$ must have become negative for some $x < a$. As $\delta(x)$ is concave down by Lemma 17, the slope of $\delta(x)$ can only get more negative as we consider increasing x values, and so the slope of $\delta(x)$ is negative for $x \geq a$. $\delta(x)$ will continue to become more negative for $x > a$. Therefore, $\delta(x)$ has a maximum value at $x = a$, for $a \leq x \leq \frac{v_a}{\sin(\frac{2\pi}{n})} - 1$. Let us call the minimum height loss on sight line $S(1)$ the value $\sigma = -\delta(a)$.

   Let us now consider a pursuer starting at some height $x \geq a$ on sight line $S(0)$. If we consider the lowest reachable point on sight line $S(1)$, we have just shown that this must be a point at height $x' \geq x + \sigma$. Let us now consider the regular n-sided regular convex polygon P formed by the orbital path at height x'. We note that any straight line path from sight line $S(0)$ at height x, which is inside P, to a sight line $S(i)$, where $i > 1$, must pass through a point at height $y \geq x'$ on sight line $S(1)$, which lies on or outside the regular convex polygon. It cannot pass via a point lower than x', as x' is defined as the lowest reachable point in sight line $S(1)$ (without losing sight of the evader). Therefore, when this path turns on the sight line $S(i)$, it must do so on a point outside the polygon P, as a straight line path starting inside a regular convex polygon and intersecting the edge of the polygon must end up outside the polygon (See Figure 3.10). Therefore, the turn at $S(i)$ occurs at a height y' where $y' > x'$, as all the points at height x' are on the polygon P. As

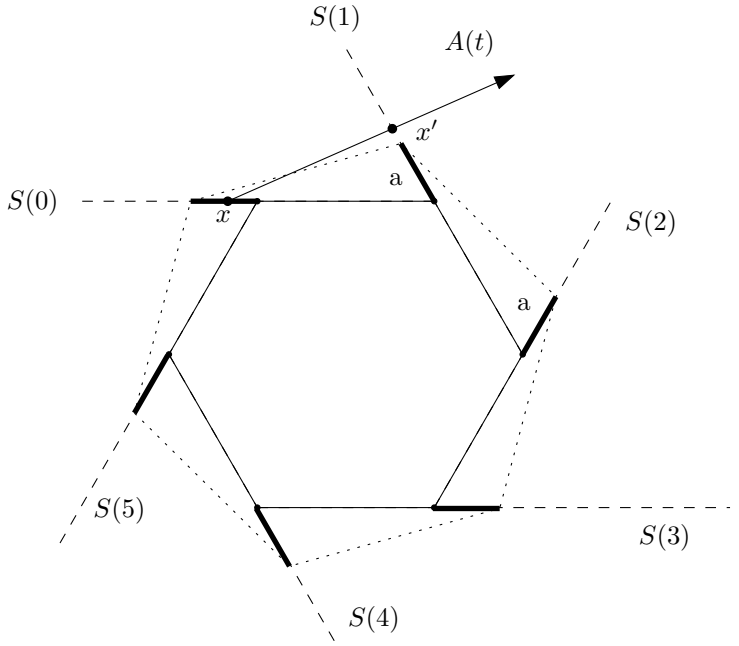


Figure 3.10: The dashed line is the orbital path for height a, with the thick line segments indicating a length of a. If x is at a height less than a on sight line $S(0)$, and the path $A(t)$, starting at x, passes through the point x' on sight line $S(1)$ at a height greater than or equal to a, it must intersect all subsequent sight lines at a height greater than a, since the orbital path forms a regular polygon at height a on every sight line. A straight line path starting inside a convex polygon cannot intersect the polygon more than once.

$x' \geq x + \sigma$, this means that $y' > x + \sigma$ and so any path starting at a height $x \geq a$ must turn at some sight line at a new height which is greater than or equal to $x + \sigma$.

Therefore, this shows no matter where the pursuer decides to turn, the new height at the sight line where the pursuer turns is higher by at least σ. After a finite number of turns (at most $\left\lceil \left(\frac{v_a}{\sin(\frac{2\pi}{n})} - 1 - a \right) / \sigma \right\rceil + 1$ turns), the pursuer will exceed the height $\frac{v_a}{\sin(\frac{2\pi}{n})} - 1$, after which the pursuer will be unable to reach the subsequent sight line without losing sight of the evader.

Case. $\cos\left(\frac{2\pi}{n}\right) < a/(1+a)$:

In this case, when the pursuer starts at a height a along sight line $S(0)$, the closest point reachable on sight line $S(1)$ from this starting point is lower than a. Therefore, although the pursuer could choose to orbit the polygon with speed

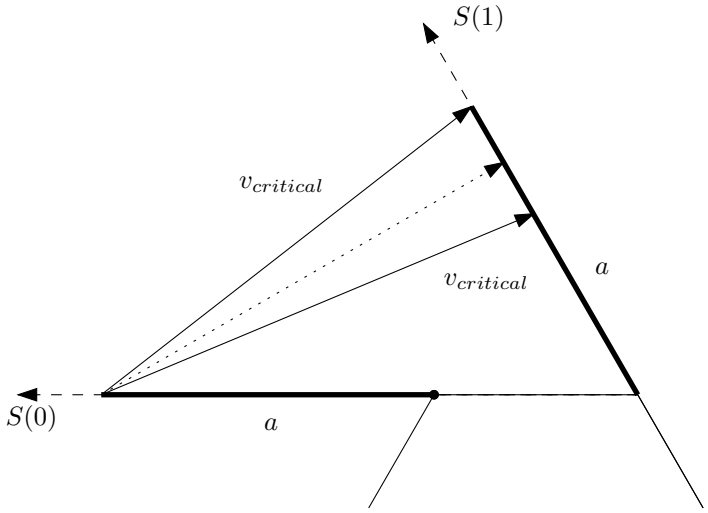


Figure 3.11: When $\cos(2\pi/n) < a/(1+a)$, the lowest reachable point on $S(1)$ from the pursuer starting point is lower than the point reachable via the orbital path. Therefore, with speed $v_{critical}$, the pursuer can reach a point with height less than a on sight line $S(1)$. The dashed line shows the path to the point on sight line $S(1)$ closest to the pursuer starting point.

$v_{critical}$, it can also reach a point with height less than a on sight line $S(1)$.

1. *If* $v_a < (1+a)\sin\left(\frac{2\pi}{n}\right)$ *the pursuer will lose.*

   This is true as the pursuer is unable to keep the evader in sight when the evader rounds the first corner of the polygon P. She cannot reach the sight line $S(1)$ before the evader reaches $\mathbf{B}(1)$, therefore, at time $t+\epsilon$ for some small $\epsilon > 0$, the evader cannot be seen by the pursuer.

2. *If* $v_a \geq (1+a)\sin\left(\frac{2\pi}{n}\right)$ *the pursuer will win.*

   By Lemma 17, the second derivative $\frac{d^2\delta}{dx^2}(x) < 0$ for $v_a - 1 \leq x \leq a$.

   We know that $\cos\left(\frac{2\pi}{n}\right) < \frac{a}{1+a}$. Therefore,

$$\begin{aligned} x - (x+1)\cos\left(\frac{2\pi}{n}\right) &> x - (x+1)\frac{a}{1+a} \\ &= \frac{x(1+a) - (x+1)a}{1+a} \\ &= \frac{x-a}{1+a} \end{aligned}$$

We can apply this to show:

$$\delta(x) = x - (x+1)\cos\left(\frac{2\pi}{n}\right) + \sqrt{{v_a}^2 - (a+1)^2\left(\sin^2\left(\frac{2\pi}{n}\right)\right)}$$

$$> \frac{x-a}{1+a} + \sqrt{{v_a}^2 - (a+1)^2\left(\sin^2\left(\frac{2\pi}{n}\right)\right)}$$

As $v_a \geq (a+1)\sin\left(\frac{2\pi}{n}\right)$,

$$\begin{aligned}
\delta(a) &> \frac{a-a}{1+a} + \sqrt{{v_a}^2 - (a+1)^2\left(\sin^2\left(\frac{2\pi}{n}\right)\right)} \\
&= \sqrt{{v_a}^2 - (a+1)^2\left(\sin^2\left(\frac{2\pi}{n}\right)\right)} \\
&\geq \sqrt{(a+1)^2\left(\sin^2\left(\frac{2\pi}{n}\right)\right) - (a+1)^2\left(\sin^2\left(\frac{2\pi}{n}\right)\right)} \\
&= 0
\end{aligned}$$

We again now note that $\delta(a) > 0$ and $\delta(v_a - 1) = v_a - 1 > 0$, and so δ_{min} in the range $v_a - 1 \leq x \leq a$ is always positive. Therefore, the pursuer is always decreasing in height relative to the evader and thus the pursuer must catch the evader in at most $\lceil \frac{a-v_a+1}{\delta_{min}} \rceil + 1$ steps. This is because it will take at most $\lceil \frac{a-v_a+1}{\delta_{min}} \rceil$ steps to reach a height $x \leq v_a - 1$, and then a final step to capture the evader.

□

3.4 Orbital Velocity Relation to Starting Position

We can now derive a relation between the starting position and the relative velocities of the pursuer and the evader when the pursuer follows the orbital path. Let r_b be the distance from the centre of the polygon to the evader's start position. Let r_a be the distance from the centre of the polygon to the the pursuer's starting position. Let l_a be the distance travelled by the pursuer during time t.

The angle between the evader's position at time $k \in \mathbb{N}$ and the evader's position at time $k+1$ is $\frac{2\pi}{n}$, measured from the centre of the polygonal obstacle. This is because the orbital path around the n sided polygon is also an n sided polygon. We can now express l_a and l_b in terms of the angle.

$l_a = 2 \times (r_a \times \sin \frac{\pi}{n})$

$l_b = 2 \times (r_b \times \sin \frac{\pi}{n})$

Therefore, given the distance travelled by the pursuer, $v_a \times t = l_a$, and the distance travelled by the evader, $1 \times t = l_b$ we can note the following:

Lemma 18. *The orbital velocity of the pursuer is the ratio of the distances of the pursuer and the evader from the centre of the polygon. That is*

$$\frac{l_a}{l_b} = \frac{r_a}{r_b} = v_a \tag{3.2}$$

As noted in the previous section, the orbital trajectory defined is not stable—any perturbation from the orbital path will cause the pursuer to either win or lose.

3.5 A Visualisation of the Visibility Constrained Pursuit-Evasion Problem

This section describes another method of viewing the visibility constrained pursuit-evasion problem, which gives an alternate view of what is occurring.

We define the set $\mathcal{V}(t)$ to be a set of points that changes with respect to time t. A point $\mathbf{p}$ is in the set $\mathcal{V}(k)$ if and only if $\mathbf{p}$ can see $\mathbf{B}(t)$ at the time k. For any time t, $\mathcal{V}(t)$ contains one or more connected components — $\mathcal{V}(t)$ for a given t is never empty since it always contains the point $\mathbf{B}(t)$. For all times t, we also remove the regions occupied by the obstacles from $\mathcal{V}(t)$. The set of points in $\mathcal{V}(t)$ now represents all the valid pursuer positions in the plane, at any given time. Therefore, for the pursuer not to lose, $\mathbf{A}(t) \in \mathcal{V}(t)$ for all t. As well, we have a maximum speed for the path, therefore, $|\frac{dA}{dt}| \leq v_a$.

If we consider a starting point $\mathbf{a}$ and starting time t_a, we can create a "cone" which opens upwards in the positive z direction with a slope $\frac{1}{\alpha}$. However, we need to stop the growth of the "cone" whenever it encounters an obstacle. In effect, we have changed the problem into attempting to find a path in three dimensions (the $\mathcal{V}(t)$ volume) that obeys the speed criteria ($|\frac{dA}{dt}| \leq v_a$) and only travels in the positive z direction. If no such path exists, the evader can eventually lose sight of the pursuer. If the path $\mathbf{A}(t)$ exists and intersects the path $\mathbf{B}(t)$, then the pursuer can capture the evader. If the path $\mathbf{A}(t)$ exists but cannot intersect the path $\mathbf{B}(t)$, then we have a tie situation.

3.6 Future Work

The natural extension for this work would be to consider generalising this solution for more complex environments and more complex evader shapes. It should be possible to show that for many more types of periodic paths for the evader, a tie situation can occur, by applying a similar analysis on the relative distance between the pursuer and the evader in the optimal capture path for the pursuer. As well, it would be interesting to see if a similar result can be shown for polygonal evaders, however it will be necessary for additional work to be done to handle the curved paths which may arise. Ultimately, it may be possible to discover common features of infinite length paths that cause tie situations. This may allow us to answer the decision questions — For a given evader path and a pursuer starting position and speed, can the pursuer capture the evader? If not, will the evader ultimately lose sight of the pursuer?

Another direction for future research would be to generalise the results of Efrat et al.[8] to paths of infinite length. It would be interesting to see if it would be possible to detect the tie situation, thereby guaranteeing that the algorithm would terminate. It may be possible to detect such a situation by detecting that the optimal path for the pursuer and the evader repeats the same relative positions for the pursuer and the evader at two different times.

Given a polygonal scene, we can also ask the more general question — given two points $\mathbf{A}(0)$ and $\mathbf{B}(0)$ and two speeds α and β and a time bound t_0, does there exist a path $\mathbf{B}(t)$ with speed at most β such that for all paths $\mathbf{A}(t)$ with speed at most α $\mathbf{A}(t)$ loses sight of $\mathbf{B}(t)$ for some $t \leq t_0$? If not, does there exist a path $\mathbf{B}(t)$ where for all paths $\mathbf{A}(t)$ the pursuer cannot capture the evader?

Another variation could consider multiple pursuers and evaders. Given the paths for multiple evaders and starting positions for the pursuers. The visibility constraint could be modified such that at all times, every evader is in sight by at least one pursuer. We can then ask, under the visibility constraint, is it possible for the pursuers to capture all the evaders. If we relax the visibility constraint, we could also ask what is the maximum number of evaders that could be kept in sight and captured.

3.7 Conclusion

In this chapter, we analyse an example of visibility constrained pursuit-evasion when a point-sized evader moves along a path that follows the boundary of a regular convex polygon at a constant rate without ever stopping. In this case, we show that all three possible outcomes are possible - the evader can lose sight of the pursuer, the pursuer

can capture the evader, and a tie can occur, where the pursuer cannot capture the evader, but the evader cannot lose sight of the pursuer. We determine the conditions under which each of these three situations occur for a situation involving a pursuer, an evader and a regular polygonal obstacle by showing properties of the shortest paths the pursuer can take.

Chapter 4

Conclusion

In this thesis, we consider two instances of the pursuit-evasion problem. We show that, when using the L_1 distance metric, the size of the pursuit-evasion Voronoi diagram is polynomial in the number of evaders, pursuers and obstacles, and we give a polynomial time algorithm to compute this diagram. We use a variation of Dijkstra's algorithm for computing single source shortest paths to explore the plane, while computing regions reachable by the evaders. This allows us to consider pursuers and evaders, each with their own starting point, starting time and speed, as well as line segment obstacles in the plane.

We also show that, when we a have a single pursuer chasing an evader around a regular convex polygon and the pursuer is forced to keep the evader in sight, there are three possible situations which occur depending on the starting conditions: the evader can force the pursuer to lose visibility, the pursuer can capture the evader, and a tie situation where the pursuer does not lose sight of the evader, but cannot capture the evader. For these three situations, we give the conditions under which they occur.

Bibliography

[1] Oswin Aichholzer, Franz Aurenhammer, and Belén Palop. Quickest paths, straight skeletons, and the city voronoi diagram. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 151–159, New York, NY, USA, 2002. ACM Press.

[2] Takao Asano and Tetsuo Asano. Voronoi diagram for points in a simple polygon. In *Proceedings of the Japan-US Joint Seminar, 1986, Discrete Algorithms and Complexity*, pages 51–64, 1987.

[3] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

[4] F. Aurenhammer and H. Edelsbrunner. An optimal aglorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recognition*, 17:251–257, 1984.

[5] F. Aurenhammer and H. Edelsbrunner. An optimal algorithm for constructing the weighted voronoi diagram in the plane. *Pattern Recognition*, 17(2):251–257, 1984.

[6] K. L. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n \log^2 n)$ time. In *Proc. Third Annual Symposium on Computational Geometry*, Waterloo, Ontario, June 1987.

[7] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry: algorithms and applications*, pages 33–40. Springer-Verlag New York, Inc., Secaucus, NJ, USA, second, revised edition, 2000.

[8] Alon Efrat, Héctor H. González-Baños, Stephen G. Kobourov, and Lingeshwaran Palaniappan. Optimal strategies to track and capture a predictable target. In *ICRA*, pages 3789–3796. IEEE, 2003.

[9] W. Randolph Franklin, Varol Akman, and Colin Verrilli. Voronoi diagrams with barriers and on polyhedra for minimal path planning. *The Visual Computer*, 1(2):133–150, 1985.

[10] H. H. González-Baños, H. Guibas, L. Latombe, J. LaValle, S. Lin, D. Motwani, and R. Tomasi. Motion planning with visibility constraints: Building autonomous observers. In *Robotics Research - The Eighth Int. Symp.*, pages 95–101, 1998.

[11] Volkan Isler, Sampath Kannan, and Sanjeev Khanna. Randomized pursuit-evasion with limited visibility. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1060–1069, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[12] Kei Kobayashi and Kokichi Sugihara. Crystal voronoi diagram and its applications to collision-free paths. In *ICCS '01: Proceedings of the International Conference on Computational Sciences-Part I*, pages 738–747, London, UK, 2001. Springer-Verlag.

[13] S. LaValle, H. H. González-Baños, C. Becker, and J. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 731–736, 1997.

[14] T. Li and T. Yu. Planning tracking motions for an intelligent virtual camera. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1353–1358, 1999.

[15] J.D. Murray. *Mathematical Biology*, pages 655–695. Springer, second, corrected edition, 1993.

[16] Dennis Nieuwenhuisen and Mark H. Overmars. Motion planning for camera movements in virtual environments. Technical Report 004, Institute of Information and Computer Sciences, Utrectht University, Utrecht, the Netherlands, 2003.

[17] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial tessellations: Concepts and applications of Voronoi diagrams*, pages 128–138. Wiley & Sons, 1992.

[18] Sang-Min Park, Jae-Ha Lee, and Kyung-Yong Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. In *ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming,*, pages 456–468, London, UK, 2001. Springer-Verlag.

[19] Waldir L. Roque and Howie Choset. The green island formation in forest fire modeling with voronoi diagrams. Abstract for *3rd CGC Workshop on Computational Geometry*, 1998.

[20] Barry Schaudt. *Multiplicatively Weighted Crystal Growth Voronoi Diagrams*. PhD thesis, Dartmouth College, 1992.

[21] Barry Schaudt and Robert L. (Scot) Drysdale III. Multiplicatively weighted crystal growth Voronoi diagrams (extended abstract). In *Symposium on Computational Geometry*, pages 214–223, 1991.

[22] Subhash Suri and Joseph O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. In *SCG '86: Proceedings of the second annual symposium on Computational geometry*, pages 14–23, New York, NY, USA, 1986. ACM Press.