Sweeping a Terrain by Collaborative Aerial Vehicles

Alon Efrat
CS Dept, University of Arizona
alon@cs.arizona.edu

Mikko Nikkilä Valentin Polishchuk
CS Dept, University of Helsinki
firstname.lastname@cs.helsinki.fi

ABSTRACT

Mountainous regions are typically hard to access by land; because of this, search operations in hilly terrains are often performed by airborne force such as Unmanned Aerial Vehicles (UAVs). We give algorithms for motion planning and coordination for a team of UAVs under various assumptions on the vehicles equipage/capabilities and present outputs of an implementation of the algorithms.

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Autonomous vehicles, sensors, workcell organization, planning

General Terms

Algorithms, Performance, Experimentation

Keywords

coordinated motion planning, mobile guards, sensors

1. INTRODUCTION

Motion planning and coordination for mobile guards is a fundamental problem arising in search-and-rescue operations, mapping, inspection and surveillance services, military and security tasks, and many other domains. Recent advances in flight technologies, navigation and communication allow one to employ Unmanned Aerial Vehicles (UAVs) for such missions; UAVs may be more appropriate than human-operated machinery because of low relatively acquisition, maintenance and operation costs, as well as for safety reasons – under certain circumstances (such as when operating in a hostile environment) UAVs provide the only means to complete the mission. For instance, when locating armed enemy forces that hide in a terrain, it is preferable to use spy microdrones instead of conventional, larger helicopters. Problems arising in situations like this crystalize into the following algorithmic question:

Is it possible to organize flight of guards over a given terrain so that any agent moving on the terrain surface will eventually be spotted by one of the guards?

We consider several versions of the question, motivated by variance in the guards capabilities (whether the guards can follow only a straight path or an arbitrary track and whether they have speed control). Note that there can be an arbitrary number of agents, each moving in an unpredictable way; moreover, the agents are adversarial (in the competitive-analysis parlance) in that they know everything about the guards' motion and may use the knowledge to plan their escape paths.

Related Work. Placing *static* guards to see a polygonal domain is the famous *Art Gallery Problem* [20]; finding a route for a *single* guard that sees a domain is the subject in the *Watchman Route Problem* [17, 10] (heuristics for *multiple watchmen* in polygons are given in [21]). In these problems the guards are not required to spot *moving targets* (they just need to see any *static* target); catching evading objects in polygons is known as the *Walkability Problem* [23]. In *Exploration Problems* the goal is to guide guards to eventually see every point in an *unknown* polygon [15]. *Pursuit-Evasion* [22] and *Lion-and-Man Problems* (in polygonal domains, as well as in graphs) study catching of targets whose locations are *known* to the pursuer(s) [11, 9].

The problem of sweeping terrain by a ichain of guards was proposed at the last-year ACM SIGSPATIAL GIS Conference [12]. Our work differs from [12] in several aspects. [12] considered only one version of the problem (the most involved one – unrestricted motion of guards with full cooridnation) while we study different variants (guards moving on straight/arbitrary tracks and guards with/without full coordination capabilities). More importantly, while [12] presented no theoretical results (the paper gave only a *heuristic*), we significantly advance on the theory frontier by designing and analyzing *exact* algorithms for our problems. Unlike [12], we focus on (various versions of) *feasibility* problems, and when we do consider optimization we optimize a different objective (the number of guards vs. the cost of flight for a *given* number of guards in [12]). The implementations in [12] and here are similar (we, however, did not use any code from [12]; the implementation in this paper is completely independent) – both papers use grid search for experiments. Our grid is finer – 60x60 vs. 16x16 in [12], and our output is more visually appealing because we present videos with 3D views (`http://www.cs.helsinki.fi/group/compgeom/terrain.html`) while [12] showed results of their implementation only from the top view (we also make our code available for users to generate their own terrains and guards motion, and to output videos of the sweeps). Last but not least, [12] required consecutive guards in the chain to always stay very close to each other (within 2 grid units, and this bound was set uniformly over the terrain), implying that the constrains on visibility imposed by the terrain were not taken into account; on the contrary, in this paper we allow guards across the chain to set the distance between them according to the terrain features (e.g., guards might need to be close if separated by a deep canyon, but could afford to be relatively far away if separated only by shallow valleys) – this distinction enables more effective sweeps, using fewer guards. In fairness to [12], their implementation, as mentioned above, solved a harder problem – *finding arbitrary* trajectories, while in our implementation *straight* tracks are *given*. There are also few inessential technical differences between [12] and this paper.

A classics of motion coordination (in the absence of any polygons or terrains) is the *Fréchet distance problem*: Given polygonal paths for a man and a dog, can the man walk the dog with the leash of a given length L? The *free-space diagram* for a problem instance are the points in the Cartesian product of the paths, whose coordinates are closer than L; the problem is solved by building the diagram and observing that the instance is feasible iff the diagram connects the paths starts to the paths ends [7]. We employ very similar free-space ideas for our trajectory coordination algorithms.

1.1 Model

We define the notation and state the problem. A terrain $\mathbb{T}$ is the graph of a piecewise-linear function f of two variables (Fig. 1, left). The standard computational-geometry way to define a terrain is by specifying a triangulation with the value of f (the terrain elevation) given at each vertex. Specifically, let $U = \{x, y \,|\, 0 \le x \le 1, 0 \le y \le 1\}$ be the unit square in the xy-plane and suppose that a triangulation of U is given. Assume that elevation $f(v)$ is assigned to every vertex v of the triangulation. Then the terrain $\mathbb{T}$ over each triangle abc of the triangulation is defined by the plane passing through $(a, f(a)), (b, f(b))$ and $(c, f(c))$. That is, if a point $p \in U$ belongs to triangle abc, then $f(p)$—the elevation of the terrain $\mathbb{T}$ at p—is given by the linear interpolation of $f(a), f(b), f(c)$ over the triangle. The terrain itself consists of triangles (lifted and slanted copies of triangles of the triangulation); we say that the vertices and edges of the triangles are vertices and edges of $\mathbb{T}$. Let n be the number of the triangles (the complexity of $\mathbb{T}$).

Let $\mathbb{S} = \{x, y, z \,|\, 0 \le x \le 1, 0 \le y \le 1, z = 1\}$ denote the unit square lifted to the plane $z = 1$. Borrowing notation from literature on geometric flows [18, 19, 8], we call the lower side of $\mathbb{S}$ (i.e., the segment $\{x, y, z \,|\, 0 \le x \le 1, y = 0, z = 1\}$) the *bottom* and denote it by $\mathcal{B}$; the upper, right and left sides of $\mathbb{S}$ are called *top, source* and *sink* resp. and are denoted by $\mathcal{T}$, S and T resp.

Tracks and Trajectories. We consider guards moving in $\mathbb{S}$. We make a distinction between the "track" and the "trajectory" for any guard g: the *track* π is the path that g follows (a curve in $\mathbb{S}$), while the *trajectory* is the function of time that specifies the location, $g(t)$, of the guard for each time t. We make two simplifying assumptions: (1) the track of each guard is an x-monotone curve, i.e., any line parallel to the y-axis intersects any track in at most one point (this assumption will make it intuitive to speak about the part

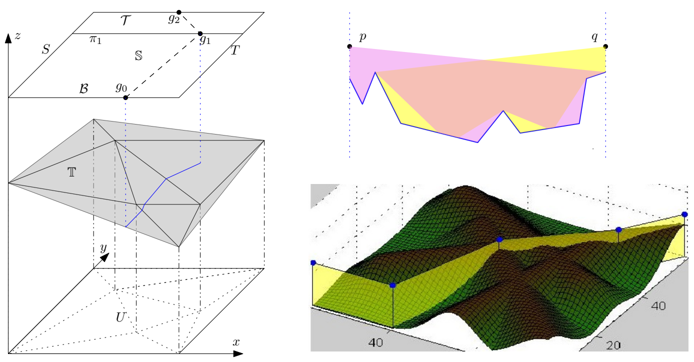

Figure 1: Left: The triangulation of U (dotted), terrain $\mathbb{T}$ (shaded), the square $\mathbb{S}$ and its sides ($\mathcal{B}$, $\mathcal{T}$, S and T), a chain $g_0 - g_1 - g_2$ of 3 guards ($K = 2$) moving along tracks $\mathcal{B} = \pi_0, \pi_1, \pi_2 = \mathcal{T}$, and the 1D terrain $\mathbb{T}(q_0, g_1)$ (blue). Right top: p, q are feasible. Right bottom: A feasible chain of guards (blue dots) must see the intersection of the terrain with the "curtain" (yellow) hung from the chain.

of $\mathbb{T}$ *between* two tracks), and (2) the maximum elevation of $\mathbb{T}$ is smaller than 1, i.e., $f < 1$; lifting the assumptions is possible but quite technical. All guards start the motion being aligned along S and finish aligned on T (i.e., all tracks start on S and end on T). For our purposes it is enough to consider guards with non-intersecting tracks. Let $K + 1$ be the number of guards; we index them 0 to K in the order as they appear on S. We assume that the guard g_0 moves along $\mathcal{B}$ and call it the *bottom* guard; similarly, the *top* guard g_K moves along $\mathcal{T}$.

Visibility. We use the same notion of visibility as in [12] (motivated, e.g., by the idea that guards are equipped with a scanner having narrow angle of view): For a point $p \in \mathbb{S}$ let $p\updownarrow$ be the line parallel to the z-axis passing through p; for two points $p, q \in \mathbb{S}$ let H_{pq} be the plane passing through $p\updownarrow$ and $q\updownarrow$ (the plane is parallel to the z-axis). Let $\mathbb{T}_{pq}$ be the cross-section of $\mathbb{T}$ by H_{pq} ($\mathbb{T}_{pq}$ is a one-dimensional terrain, i.e., the graph of a piecewise-linear function of one variable) and let $\mathbb{T}(p, q)$ be the part of $\mathbb{T}_{pq}$ that lies between $p\updownarrow$ and $q\updownarrow$; say that p, q are *feasible* if the two points collectively see $\mathbb{T}(p, q)$. Finally, say that a set of trajectories $g_0(t), \ldots, g_K(t)$ for the guards is *feasible* if the guards $g_{k-1}(t), g_k(t), k = 1 \ldots K$ are feasible at any time t (Fig. 1, right). Since the chain $g_0 - \ldots - g_K$ of guards starts on S and ends on T, any evader moving on the terrain will be seen at some time by some guard in the chain (this is because the evader cannot move unnoticed to the left of the chain).

Versions of the problem

Depending on how correlated the motion of the guards is and on whether the tracks are given in the input, we arrive at different variants of our problem, described next.

Aligned guards vs. Coordinated motion. We consider two models of how guards move along their tracks.
- In the *aligned* model, at any time during the motion, the x-coordinate of all guards must be the same.
- In the *coordinated* model each guard is allowed to move arbitrarily along its track. (Naturally, even though the motion of each individual guard is unconstrained, it is required that collectively the guards sweep the terrain, i.e., that the

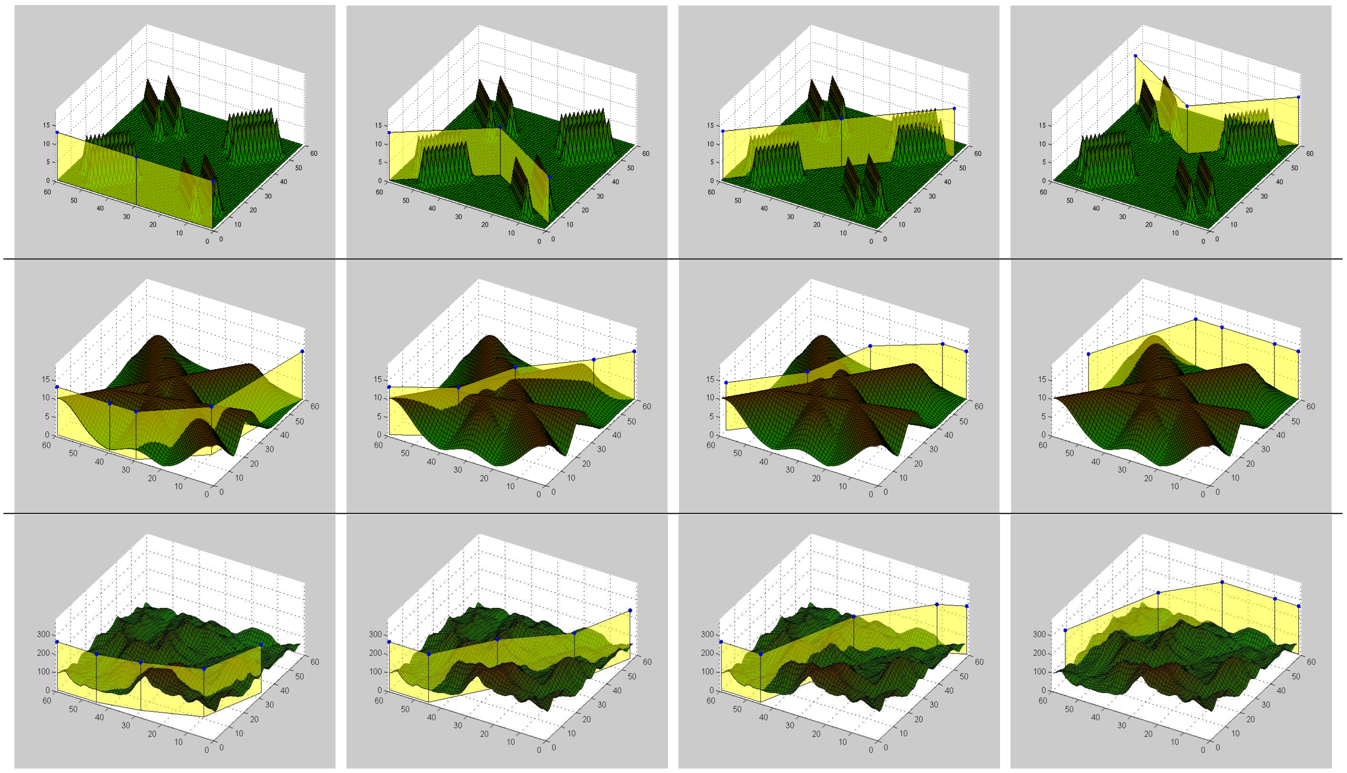

Figure 2: Synthetic (top and middle) and a real (bottom) terrains.

chain of guards is feasible at any time.)

Feasibility vs. Optimization. Another distinction between versions of our problem stems from whether the tracks are given in the input (and the problem then is only to find the *trajectories*, i.e., to synchronize the motion of guards), or finding the tracks is part of the problem.

- *Given tracks:* In the *feasibility* problem the tracks are given in the input (and hence, of course, the number of guards is given) and our task is to establish whether the guards can sweep the terrain.

- *Computing both tracks and trajectories:* In the *optimization* problem the goal is to minimize the number of guards necessary to sweep the terrain. This entails computing both the tracks and the trajectories for the guards.

1.2 Overview of the results

Section 2 describes a practical grid-based approach and points to results of its implementation; we also explain how to download and run our implementation for the user to generate her own videos of guards flying over terrains. We next give theoretical algorithmic and combinatorial bounds for our problems. We start from the simplest (yet non-trivial) case of a single guard moving along a straight track (Section 3), and show how to track changes in the visibility profile as the guard moves (Section 3.1). While the single-guard case may not necessarily be considered interesting on itself, the visibility tracking techniques developed in Section 3.1, extend to two and more guards moving along arbitrary *given* tracks (Section 4), as well as to the case of *finding* tracks for the guards (Section 5). We solve the feasibility problems for arbitrary tracks; however for optimization version (finding the minimum number of tracks) we have polynomial-time algorithms only for the case when the tracks are restricted to be straight.

2. IMPLEMENTATION

Videos with output of our implementation of terrain sweeping can be viewed at `http://www.cs.helsinki.fi/group/compgeom/terrain.html`. Fig. 2 shows some frames from videos, demonstrating the coordination between the guards – how some guards have to stay while others move. The two top rows are synthetic terrains produced with our simple terrain generator that takes a set of segments in the xy-plane, erects them to vertical walls of common height (it is easy to change the code to allow different wall heights for different segments) and smoothes the walls by Gaussians with a common variance (it is easy to allow different variances). The bottom row is a real terrain in Arizona (around 36.1°N 112.1°W) which we created as follows (inspired by the method used by Minecraft players to create terrains from the real world [1]): GoogleEarth plugin from Ambiotek [2] and MircoDEM Map Tool from the US Naval Academy [3] were used to get elevation data; the data was visualized and saved as a grayscale image, and a Java program extracted the elevation from the image based on the darkness of the color.

For the implementation we used grids (60x60) – both for terrains (a terrain is specified by the elevation at each gridpoint) and for the guards (each guard moves from a gridpoint to an adjacent gridpoint). The use of grids is justified by the complexity (in particular, non-linearity) of exact solutions as we show in subsequent sections. The implementation was programmed in Java and visualization of the

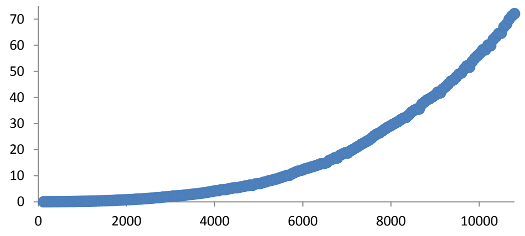

Figure 3: Runtimes (secs) vs. number of gridpoints.

output was done in MATLAB. Since our visibility model allows to parallelize the computations naturally (the motion coordination is done independently for each pair of consecutive guards in the chain) we tested the implementation on the high-performance Ukko cluster [4], with a single machine responsible for computing trajectories only for two guards. A node in the cluster is a Dell PowerEdge M610 with 32GB RAM and 2 Intel Xeon E5540 2.53GHz CPUs each having 4 cores.

Fig. 3 presents the runtimes of the implementation for a single machine (i.e., for two guards). We ran the experiments with the two guards having to sweep different-size synthetic terrains. All terrains had the y-side 60 gridpixels long (i.e., the distance between the guards tracks was 60), and we varied the x-side length. As expected, the runtime grows quadratically with the terrain length since we check feasibility of every position of the bottom guards against every position of the top guard (in the spirit of the free-space diagram for the Fréchet distance problem, see Section 4.2 for details). The solution for the real terrain from Fig. 2, bottom was found in 6.5secs which demonstrates practicality of our implementation. We also tested the implementation on more real terrains, producing solutions comparably fast.

We encourage the reader to play with our code available from (`http://www.cs.helsinki.fi/group/compgeom/rescue.zip`); we now explain how the user can work with the code to generate her own videos. The code takes input from two text files (that can be modified by the user): terrain.txt and tracks.txt. The first file is used for terrain generation (alternatively, the user can work with her own terrain as explained in the next paragraph); the second file specifies the tracks (all tracks lie at the same elevation and each track is a segment parallel to the x-axis). The file terrain.txt has the following format: the first line is the maximum height of the terrain (the height of the walls that are smoothed by Guassians), the second line is the standard deviation of the Gaussians, and each line after that specifies a segment by four numbers (x and y coordinates of the segment endpoints). The generator works as described above. The format of tracks.txt is as follows: the first line is the elevation at which the guards fly, and the following lines specify the tracks starting points. Finally, running rescueOperation.m script in MATLAB computes trajectories of the guards along the tracks and visualizes the guards flight; the animation is shown in MATLAB and then saved to m.avi.

The user can also feed her own terrain to our code instead of using our simple-minded terrain generator. Our MATLAB script offer two options for that. First, the script can load a text file with the terrain elevation at every grid point. Second, the script can use a grayscale image; the code will divide the image into pixels and set each pixel's elevation to

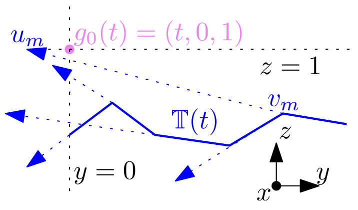

Figure 4: g_0 sees $\mathbb{T}(t)$.

the rgb value of the shade of gray in the pixel (this was the option we used for our real terrain, as described above).

3. A SINGLE BOTTOM GUARD

We now turn to exact algorithms for our problems, starting from a simple (but already non-trivial) case when there is only one guard – the bottom guard g_0. Even though this is a simplistic scenario, the techniques set up in this section lay foundations for the more complicated and interesting setting of multiple guards. Note that above we did not define visibility for a single guard; we therefore adjust our definitions: For $0 \le t \le 1$ let $\mathbb{T}(t)$ be the cross-section of the terrain $\mathbb{T}$ by the plane $x = t$ (the cross-section is a one-dimensional terrain); we say that t if *feasible* if $\mathbb{T}(t)$ is seen by the guard g_0 when its x-coordinate is t (i.e., when g_0 is at the point $(t, 0, 1)$). Our goal is simply to check whether every $t \in [0, 1]$ is feasible, i.e., whether the guard moving along $\mathcal{B}$ will successfully sweep $\mathbb{T}$ (since the track for g_0—the bottom $\mathcal{B}$—is given in the input, we are dealing with a feasibility problem).

Most of the discussion in this section will be restricted to the plane $x = t$, for some $t \in [0, 1]$; in particular, the relations "above", "below", etc. will be about the z-coordinate and "left/right" will refer to the y-coordinate (as if the plane $x = t$ is the usual, horizontal plane $\mathbb{R}^2$). Some more notation: For a point $p \in \mathbb{T}(t)$ let $\mathbb{T}_p$ be the part of $\mathbb{T}(t)$ to the left of p. For points a, b let $\overrightarrow{ab}$ denote the ray emanating from a and going through b. Every vertex v_m (resp. every edge ℓ) of $\mathbb{T}(t)$ is the intersection of the plane $x = t$ and an edge e (resp. a triangle Δ) of $\mathbb{T}$; we say that e is *responsible* for v_m and Δ is *responsible* for ℓ. For a triangle Δ of the terrain $\mathbb{T}$ (resp. an arbitrary segment ab) let $\underline{\Delta}$ (resp. $\underline{ab}$) denote its supporting plane (resp. supporting line).

We prove that the guard's position at t is feasible iff extensions of edges of $\mathbb{T}(t)$ intersect the elevation $z = 1$ at negative y-coordinates (Fig. 4); we build up to this result via a series of intermediate lemmas. We start with a simple criterion for when a point of $\mathbb{T}(t)$ is invisible to g_0. By definition, g_0 sees p if $\overrightarrow{g_0p}$ does not intersect $\mathbb{T}_p$; this can be equivalently stated as

LEMMA 1. *g_0 sees p iff $\mathbb{T}_p$ is below $\overrightarrow{g_0p}$.*

Let ℓ be an edge of $\mathbb{T}(t)$, let v be the left endpoint of ℓ and let $q \in \ell$ be some point on ℓ not seen by g_0 (Fig. 5). Our first important observation is that any point in ℓ to the left of q is also not seen by g_0; we state this as Lemma 2. Even though the lemma is intuitively obvious, we give its full proof in order to present the kind of arguments used to prove all results in the paper.

LEMMA 2. *No point in $[v, q]$ is seen by g_0.*

PROOF. By Lemma 1, some point $q^* \in \mathbb{T}_q$ lies above the ray $\overrightarrow{g_0q}$. Consider two cases (Fig. 5):

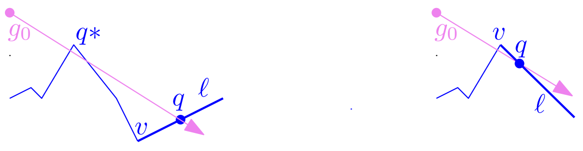

Figure 5: Slopes of the ray and the edge.

(1) If $\overrightarrow{g_0q}$ has higher slope than ℓ (i.e., if to the left of q the ray $\overrightarrow{g_0q}$ runs above ℓ), then $q^* \notin [v,q]$ since any point of the interval $[v,q]$ is below $\overrightarrow{g_0q}$; thus $q^* \in \mathbb{T}_q \setminus [v,q] = \mathbb{T}_v$. That is, there exists a point $q^* \in \mathbb{T}_v$ that lies above the ray $\overrightarrow{g_0q}$. But then, since $[v,q]$ is below $\overrightarrow{g_0q}$, the point q^* lies above the ray $\overrightarrow{g_0p}$ also for *any* point $p \in [v,q]$, which implies that p is not seen.
(2) If $\overrightarrow{g_0q}$ has smaller slope than ℓ (i.e., if to the left of q the ray $\overrightarrow{g_0q}$ runs below ℓ), then the vertex v alone blocks any point $p \in [v,q]$ (and, in fact, any point in ℓ) because v is above the ray $\overrightarrow{g_0p}$. $\square$

Let $v_0, \dots, v_M$ be the vertices of $\mathbb{T}(t)$ ordered by increasing y-coordinate; as with other notation in the paper, when we want to emphasize that the vertices' locations depends on t, we write them as functions of time $v_0(t), \dots, v_M(t)$. We now prove that it is sufficient to see only the vertices of $\mathbb{T}(t)$:

LEMMA 3. *If g_0 sees $v_0, \dots, v_M$, then it sees all of $\mathbb{T}(t)$.*

PROOF. If g_0 sees v_{m+1} then by Lemma 2 it also sees the whole edge $v_m v_{m+1}$. $\square$

Next we prove that it is sufficient to see only a subset of vertices – namely those that have a reflex vertex as a left neighbor. Specifically, for $m = 1 \dots M-1$ the vertex v_m is *reflex* if $v_m v_{m+1}$ is below the ray $\overrightarrow{v_{m-1}v_m}$; a vertex v_{m+1} is *post-reflex* if its predecessor, v_m is reflex.

LEMMA 4. *If g_0 sees all post-reflex vertices, then it sees all of $\mathbb{T}(t)$.*

PROOF. By Lemma 3 it is enough to prove that all vertices of $\mathbb{T}(t)$ are seen by g_0. Let v_{m+1} be the leftmost non-seen vertex (Fig. 6, left). Since v_m is seen, $\mathbb{T}_{v_m}$ is below $\overrightarrow{g_0v_m}$; in particular, the edge $v_{m-1}v_m$ is below $\overrightarrow{g_0v_m}$. But this means that to the left of v_m—the intersection point of the ray $\overrightarrow{v_{m-1}v_m}$ and the ray $\overrightarrow{g_0v_m}$—the former ray is above the latter. Finally, since v_m is not reflex, $v_m v_{m+1}$ is above $\overrightarrow{v_{nm-1}v_m}$, and hence is (even more) above $\overrightarrow{g_0v_m}$, which implies that for any $p \in [v_m v_{m+1}]$ the ray $\overrightarrow{g_0p}$ is above $\overrightarrow{g_0v_m}$, and hence is also above $\mathbb{T}_p$. $\square$

Finally, we observe that g_0 seeing a post-reflex vertex is equivalent to a simple algebraic condition. Specifically, for a vertex v_{m+1} let $u_m = \overrightarrow{v_{m+1}v_m} \cap \{z=1\}$ be the intersection of the plane $z=1$ (equivalently, the line $z=1$ in the plane $x=t$) with the ray $\overrightarrow{v_{m+1}v_m}$, and let y_m be the y-coordinate of u_m; assume that $y_m = -\infty$ if the ray never intersects $z = 1$, i.e., if $u_m = \emptyset$ (e.g., $y_m = -\infty$ in Fig. 4). The guard sees all post-reflex vertices iff for each post-reflex vertex v_{m+1} the guard is above the ray $\overrightarrow{v_{m+1}v_m}$, or (since the guard's (y,z)-coordinates are $(0,1)$) iff the rays intersects the line $z=1$ at non-positive y-coordinates:

LEMMA 5. *g_0 sees all post-reflex vertices iff $y_m \le 0$ for each post-reflex vertex v_{m+1}.*

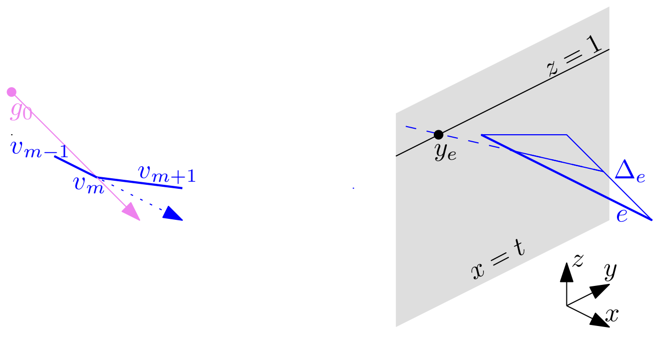

Figure 6: Left: v_m is not reflex. Right: $y_e = \Delta_e \cap \{x = t\} \cap \{z = 1\}$.

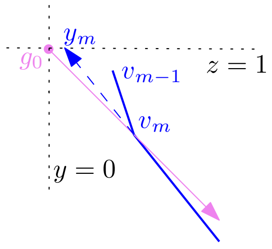

Figure 7: v_{m-1} blocks v_m.

From Lemmas 3–5 we obtain:

LEMMA 6. *g_0 sees $\mathbb{T}(t)$ iff $y_m \le 0$ for each reflex v_m.*

Lemma 6 implies that feasibility of g_0 can be checked by looking at y_m only for a *subset* of vertices of $\mathbb{T}(t)$ (namely, the reflex vertices of $\mathbb{T}(t)$). We could have gone further along this direction and prove that feasibility of g_0 can be checked by looking at y_m for an *even smaller subset* of vertices of $\mathbb{T}(t)$ (namely, the endpoints of "reflex chains" of $\mathbb{T}(t)$). Instead, we go in the opposite direction and prove that checking the guard's feasibility is equivalent to checking the sign of y_m for *all* (internal) vertices of $\mathbb{T}(t)$ (v_m is internal if $m \ne 0, M$):

LEMMA 7. *g_0 sees $\mathbb{T}(t)$ iff $y_m \le 0$ for every internal v_m.*

PROOF. A non-reflex vertex v_m with $y_m > 0$ is blocked already by v_{m-1} (Fig. 7). $\square$

It may seem like Lemma 7 is a weakening of Lemma 6, and that checking feasibility of g_0 is better be done via Lemma 6. Still, asymptotically, the feasibility check takes the same ($O(M)$) time no matter which of the two lemmas is used because every internal vertex of $\mathbb{T}(t)$ may be reflex (also, the number of endpoints of reflex chains of $\mathbb{T}(t)$ may be $\Omega(M)$). More importantly, Lemma 7 is simpler *conceptually* since it provides a uniform treatment of all vertices of $\mathbb{T}(t)$: there is no need to find out which vertices are reflex; the guard's feasibility can be checked by (blindly) checking the sign of y_m for each vertex of $\mathbb{T}(t)$. This will also be helpful when considering how the visibility of $\mathbb{T}(t)$ by g_0 changes as the guard moves – which we do next.

We introduce the last piece of notation in this section. An edge of the terrain $\mathbb{T}$ is *internal* if it is not a boundary edge (i.e., if its projection onto the xy-plane does not belong to the boundary of the unit square); in the remainder of the section we consider only internal edges of $\mathbb{T}$. For an edge e let

$e_{\min}, e_{\max}$ be the x-coordinates of the endpoints of e; assume that $\mathbb{T}$ has no edges perpendicular to the x-axis ($e_{\min} < e_{\max}$ for all e). There are two triangles incident to e; let Δ_e denote the incident triangle that is "locally further" from the xz-plane (formally, a ray in $\underline{\Delta_e}$ going from a point in Δ_e with x-coordinate between $e_{\min}$ and $e_{\max}$ and intersecting e, also intersects the xz-plane). Finally, for $t \in [e_{\min}, e_{\max}]$ let $y_e(t)$ be the y-coordinate of the intersection of the planes $\underline{\Delta_e}, x = t$ and $z = 1$ (Fig. 6, right).

Lemma 7 implies that the problem instance is infeasible iff $y_m(t) > 0$ for some vertex $v_m(t)$ of $\mathbb{T}(t)$ for some t. As t changes, vertices of $\mathbb{T}(t)$ appear, move and disappear. The crucial observation is that every vertex v_m (resp. every edge ℓ) of $\mathbb{T}(t)$ is the intersection of the plane $x = t$ and an edge e (resp. a triangle Δ) of $\mathbb{T}$. This means that the whole variety of vertices of $\mathbb{T}(t)$ for all $t \in [0, 1]$ can be obtained by intersecting every edge e of $\mathbb{T}$ with the plane $x = t$ for $t \in [e_{\min}, e_{\max}]$. That is, all possible vertices of $\mathbb{T}(t)$ for $t \in [0, 1]$ can be obtained by looking at the terrain $\mathbb{T}$ edge-by-edge. In particular, the (non-infinite) values of $y_m(t)$ for all possible vertices $v_m(t)$ of $\mathbb{T}(t)$ for all times $t \in [0, 1]$ coincide with the values of $y_e(t)$ for all edges e of $\mathbb{T}$.

Lemma 8. *y_e is a linear function of t. The function can be found in constant time.*

Proof. $y_e(t)$ is defined by 3 linear constraints ($x = t, z = 1$ and $(x, y_e, z) \in \underline{\Delta_e}$), one of which ($x = t$) changes linearly with t. $\square$

By Lemma 8, $y_e(t)$ attains its maximum at one of the extremes – $e_{\min}$ or $e_{\max}$. Thus, we can check in constant time whether $e \cap \{x = t\}$ will ever be a "visibility blocking vertex" (i.e., a vertex v_m with $y_m(t) > 0$ for some $t \in [e_{\min}, e_{\max}]$). Since there are $O(n)$ edges in $\mathbb{T}$, checking whether g_0 is ever in an infeasible position takes overall linear time:

Theorem 9. *It can be checked in $O(n)$ time whether a single bottom guard sweeps $\mathbb{T}$.*

3.1 Tracking the visibility

We argued above that the feasibility of g_0 moving along $\mathcal{B}$ can be checked efficiently by considering the terrain $\mathbb{T}$ edge-by-edge. We now present an alternative way to check the feasibility: Even though this leads to a less efficient algorithm than the one presented above (Theorem 9), the techniques developed here serve as the crucial ingredient for the algorithms in the next sections for the case of multiple guards.

Specifically, to check whether g_0 sweeps $\mathbb{T}$ we check if the guard is feasible at $t = 0$ and then track how the visibility of $\mathbb{T}(t)$ changes with t. Let v_m be a vertex of $\mathbb{T}(t)$ and let $\ell = v_i v_{i+1}$ be an edge of $\mathbb{T}(t)$ lying to the right of v_m ($i \geq m$). We define the *ℓ-shadow of v_m*, $\mathrm{sh}_{v_m \to \ell}$, to be the point where the ray $\overrightarrow{g_0 v_m}$ intersects the supporting line of ℓ ($\mathrm{sh}_{v_m \to \ell} = \overrightarrow{g_0 v_m} \cap \underline{\ell}$); the ℓ-shadow shows how much of ℓ is blocked from the guard by v_m. Let sh_ℓ be the rightmost shadow on ℓ. The next lemma asserts that the visible part of ℓ is determined by vertices of $\mathbb{T}_{v_i}$ (Fig. 8):

Lemma 10. *The part of ℓ visible to g_0 is the part to the right of sh_ℓ.*

Proof. If a point $p \in \ell$ is visible, then $\mathbb{T}_p$ is below $\overrightarrow{g_0 p}$; in particular, all vertices of $\mathbb{T}_p$ are below the ray, and hence all rays $\overrightarrow{g_0 v_m}$ for $m \leq i$ are also below $\overrightarrow{g_0 p}$, implying that p is to the right of sh_ℓ. Conversely, if $p \in \ell$ is invisible, then some point on $\mathbb{T}_p$ is above $\overrightarrow{g_0 p}$, implying that some vertex v_m of $\mathbb{T}_p$ is above the ray, meaning that $\mathrm{sh}_{v_m \to \ell}$ is to the right of p. $\square$

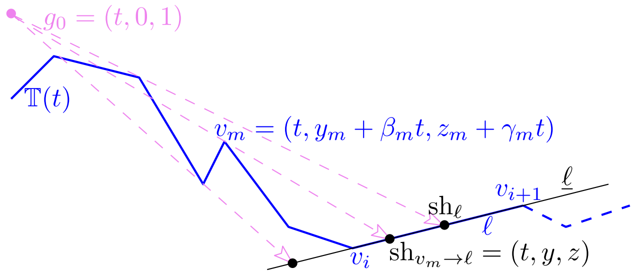
$g_0 = (t, 0, 1)$
$\mathbb{T}(t)$
$v_m = (t, y_m + \beta_m t, z_m + \gamma_m t)$
ℓ
v_{i+1}
sh_ℓ
ℓ
v_i
$\mathrm{sh}_{v_m \to \ell} = (t, y, z)$

Figure 8: Shadows of some vertices of $\mathbb{T}_{v_i}$.

Let us now track how sh_ℓ changes with time. While the plane $x = t$ intersects the same set of edges of $\mathbb{T}(t)$ (i.e., when t is between consecutive x-coordinates of vertices of the terrain $\mathbb{T}$) the terrain $\mathbb{T}(t)$ is the same combinatorially, i.e., has the same set of vertices and edges. For any edge ℓ and vertex v_m motion of the shadow $\mathrm{sh}_{v_m \to \ell}$ is described by an algebraic curve of low degree:

Lemma 11. $\mathrm{sh}_{v_m \to \ell}(t) = \left(t, \frac{p_2^y(t)}{p_1^y(t)}, \frac{p_2^z(t)}{p_1^z(t)}\right)$ *where p_1^y, p_1^z are linear and p_2^y, p_2^z are quadratic functions; the functions can be computed in constant time.*

Proof. Let e_m be the edge of $\mathbb{T}$ responsible for v_m and let Δ be the triangle responsible for ℓ ($v_m = e_m \cap \{x = t\}, \ell = \Delta \cap \{x = t\}$); let $x = t, y = y_m + \beta_m t, z = z_m + \gamma_m t$ be the (parametric equation of the) supporting line $\underline{e_m}$ of e_m (here (y_m, z_m) is the point where $\underline{e_m}$ intersects the yz-plane), and let $y = ax + bz + c$ be the (equation of the) supporting plane $\underline{\Delta}$ of Δ. The $(y.z)$-coordinates of $\mathrm{sh}_{v_m \to \ell}$ satisfy (see Fig. 8):

$$\frac{1 - z}{y} = \frac{1 - z_m - \gamma_m t}{y_m + \beta_m t}$$

or $y = \frac{y_m + \beta_m t - (y_m + \beta_m t) z}{1 - z_m - \gamma_m t}$. Substitute into the equation for $\underline{\Delta}$ with $x = t$: $y_m + \beta_m t - (y_m + \beta_m t)z = (1 - z_m - \gamma_m t)(at + bz + c)$ and solve for z:

$$z = \frac{y_m + \beta_m t - (1 - z_m - \gamma_m t)(at + c)}{b(1 - z_m - \gamma_m t) + (y_m + \beta_m t)}$$

or $z(t) = p_2^z(t)/p_1^z(t)$; substitute this back into the equation for $\underline{\Delta}$ to obtain $y(t) = p_2^y(t)/p_1^y(t)$. $\square$

Using Lemma 11 one could build the trajectory of the shadow in the 3D space; this however is not our goal. For us it is more important to know how the shadow moves *on the edge* $\ell = v_i v v_{i+1}$. For that we define the function $d_{v_m \to \ell}(t)$ as the signed distance from the vertex v_i of ℓ to $\mathrm{sh}_{v_m \to \ell}$ (*signed* means that the distance is negative if $\mathrm{sh}_{v_m \to \ell}$ is to the left of v_i on $\underline{\ell}$). The vertex v_i moves linearly with t along the edge e_i of $\mathbb{T}$ responsible for the vertex: $v_i(t) = (t, y_i + \beta_i t, z = z_i + \gamma_i t)$ where (y_i, z_i) is the point where $\underline{e_i}$ intersects the yz-plane. From this and Lemma 11 we have:

Lemma 12. $d_{v_m \to \ell}(t) = \sqrt{p_6(t)/p_4(t)}$ *where p_6, p_4 are polynomials of degrees 4 and 6 resp.*

By Lemma 10, the part of ℓ visible to g_0 is defined by the maximum of the functions $d_{v_m \to \ell}$ for all $m \leq i$; let $d_\ell(t) = \max_{m \leq i} d_{v_m \to \ell}(t)$ denote the rightmost visible point of ℓ (more precisely – the distance from the point to $v_i(t)$). Recall that the upper envelope (pointwise maximum) of graphs of i functions that pairwise intersect at most s times has $O(\lambda_{s+2}(i))$ complexity and can be built in $O(\lambda_{s+1}(i) \log i)$ time [14], where $\lambda_s(i)$ is the maximum length of an i-element order-s *Davenport-Schinzel (DS) sequence* (see [6, Sections 1 and 2] for definitions and notation relevant to DS sequences and their algorithmic and combinatorial properties).

LEMMA 13. *d_ℓ has $O(\lambda_{12}(i))$ complexity and can be built in $O(\lambda_{11}(i) \log i)$ time.*

PROOF. By Lemma 12, the functions $d_{v_m \to \ell}(t)$ for two different m's intersect at most 10 times. $\square$

Since the terrain $\mathbb{T}(t)$ has $O(n)$ edges,

LEMMA 14. *The functions d_ℓ for all edges of $\mathbb{T}(t)$ can be computed in $O(n\lambda_{11}(n))$ time.*

The above discussion was about the case when the plane $x = t$ was between of vertices of the terrain $\mathbb{T}$ with consecutive x-coordinates. Every time the plane crosses a vertex of $\mathbb{T}$, we recompute the functions d_ℓ using Lemma 14 (note that the vertices and edges of $\mathbb{T}(t)$ change every time the plane hits a vertex of $\mathbb{T}$). Since there are $O(n)$ vertices of $\mathbb{T}$ and since $\lambda_{11}(n) \leq n2^{O(2^{O(\alpha^4(n) \log \alpha(n))})}$ [6], overall we obtain:

LEMMA 15. *The functions $d_\ell(t)$, for all edges ℓ of the terrains $\mathbb{T}(t)$ for all $t \in [0, 1]$, each showing how much of the edge ℓ is seen by g_0 at time t, can be built in $O(n^3 2^{O(\alpha^4(n) \log \alpha(n))})$ time.*

By Lemma 10, the problem instance is infeasible iff $d_\ell(t) > 0$ for some edge ℓ at some time t.

4. GIVEN TRACKS

We now turn to multiple guards. In this section we solve the feasibility problems – can the guards sweep $\mathbb{T}$ moving along *given* tracks?

4.1 Aligned guards

We first consider the aligned model (in which at any time during the motion all guards must have the same x-coordinate). Since there is no question of finding trajectories for the guards (the trajectories are implied by the tracks and by the fact that the guards are aligned), the problem is "*Do* the guards sweep $\mathbb{T}$ while moving along the tracks?"

Bottom and top guards. We start from the simplest scenario of two guards g_0, g_1 moving along $\mathcal{B}$ and $\mathcal{T}$ resp. The problem is solved by a simple two-fold application of the visibility tracking tools developed in Section 3.1. First, determine how the visibility of edges of $\mathbb{T}(t)$ by g_0 changes with time – this is exactly what we did in Section 3.1. Next, do the same thing for g_1 (a little extra work is to keep track of the length of each edge, but this is easy since the length changes linearly with t). At time t the two guards do not see $\mathbb{T}(t)$ iff on some edge of the terrain the shadow from g_0 gets to the right of the shadow of g_1. Thus, by Lemma 15, it can be checked in overall $O(n^3 2^{O(\alpha^4(n) \log \alpha(n))})$ time whether aligned bottom and top guards sweep $\mathbb{T}$.

Non-straight tracks. Suppose now that the bottom and the top of $\mathbb{S}$ are not straight-line segments, but instead are arbitrary non-crossing x-monotone polygonal paths π_0, π_1 each connecting the source S to the sink T; assume that the domain of the terrain $\mathbb{T}$ lies between (the xy-projections of) π_0 and π_1. Testing whether the two aligned guards g_0, g_1, moving along the tracks π_0, π_1, sweep $\mathbb{T}$ can be done similarly to the case when the tracks were straight-line segments; the only change is that the functions describing how the shadows move along edges of $\mathbb{T}(t)$ will have to be recomputed also every time a guard turns from one segment of its track to the next (when each track consisted of one segment, we recomputed the functions only at the vertices of $\mathbb{T}$). That is, Lemma 14 will have to be applied $O(n + C)$ times where C is the total number of edges in the tracks. Similarly to Lemma 15, we obtain that it can be checked in $O((n + C)n^2 2^{O(\alpha^4(n) \log \alpha(n))})$ time whether two aligned guards sweep the part of $\mathbb{T}$ between their tracks. (Note that if the tracks are not parallel to each other, the guards will have to move with different speeds to keep the alignment; our solutions extend to the case when all guards move at the same speed, possibly reaching T at different times.)

Multiple guards. Suppose now that $K + 1$ aligned guards $g_0 - \cdots - g_K$ move along given (non-crossing) tracks $\mathcal{B} = \pi_0, \pi_1, \ldots, \pi_{K-1}, \pi_K = \mathcal{T}$; let C be the total complexity of the tracks. Let $\mathbb{T}(\pi_k, \pi_{k+1})$ be the part of $\mathbb{T}$ between π_k and π_{k+1}, i.e., the restriction of the terrain onto the set between the (xy-projections of the) tracks. In our visibility model, $\mathbb{T}(\pi_k, \pi_{k+1})$ can be swept only by g_k, g_{k+1}. Thus, testing whether $\mathbb{T}$ is swept by the $K + 1$ guards boils down to testing whether g_k, g_{k+1} sweep $\mathbb{T}(\pi_k, \pi_{k+1})$ for each $k = 0 \ldots K$. The pairwise feasibility can be done as above. Charging the time for testing g_k, g_{k+1} to the complexity of $\mathbb{T}(\pi_k, \pi_{k+1})$, we obtain:

THEOREM 16. *It can be checked in $O((n+C)n^2 2^{O(\alpha^4(n) \log \alpha(n))})$ time whether aligned guards moving along given tracks sweep $\mathbb{T}$.*

4.2 Finding trajectories: Coordinated motion

We now switch to the coordinated model (in which the guards do not have to form a line perpendicular to the x-axis). We still consider a feasibility problem – "*Can* the guards move along given tracks so as to sweep $\mathbb{T}$?" (cf. the question "*Do* the guards sweep $\mathbb{T}$ while moving along the tracks?" for the aligned model in Section 4.1), i.e., our goal is to find the trajectories for the guards.

Bottom and top guards. As in the previous section, we start from the simplest case when there are only two guards – bottom and top; also as in the previous section, the solution to the general case of arbitrary number of guards moving along arbitrary tracks will be reduced directly to the two-guards case. For $(t_0, t_1) \in [0, 1]^2$ let $g_0(t_0)$ be the point $(t_0, 0, 1) \in \mathcal{B}$ and let $g_1(t_1) = (t_1, 1, 1) \in \mathcal{T}$; let $\mathbb{H}(t_0, t_1)$ be the vertical plane though $g_0(t_0), g_1(t_1)$. Let $\mathbb{T}(t_0, t_1)$ be the cross-section of $\mathbb{T}$ by $\mathbb{H}(t_0, t_1)$; $\mathbb{T}(t_0, t_1)$ is a one-dimensional terrain. The point (t_0, t_1) is *feasible* if $\mathbb{T}(t_0, t_1)$ is collectively seen by the two guards $g_0(t_0), g_1(t_1)$.

To solve our problem we will build the set of all feasible points within the unit square in the t_0t_1-plane (Fig. 9). The instance is feasible iff this feasible set contains a path from $(0, 0)$ to $(1, 1)$ – the path defines coordinated motion of the guards to sweep $\mathbb{T}$. (Our feasible set is very much like the free-space diagram for the Fréchet distance [7].) To build the feasible set we will extend the techniques developed for the

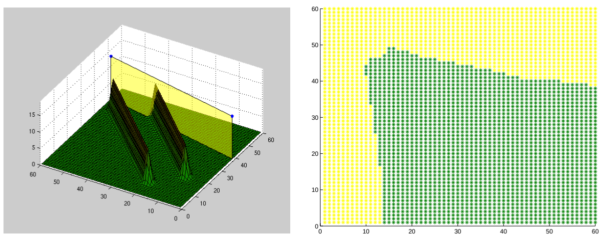
Figure 9: Left: A terrain consisting of a valley between two steep "walls". Right: The free space (yellow) for the bottom and the top guards; the guards are infeasible when g_0 is much to the right of g_1.

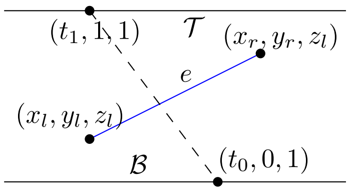

Figure 10: Top view: $(t_0 t_1) \in \chi_e \Leftrightarrow [((t_1, 1) - (t_0, 0)) \times ((x_l, y_l) - (t_0, 0))] \geq 0, [((t_1, 1) - (t_0, 0)) \times ((x_r, y_r) - (t_0, 0))] \leq 0$**.**

aligned case, where the guards had one degree of freedom (t), to the case of each guard having its own degree of freedom (t_0, t_1). First of all (extending the idea of considering $\mathbb{T}(t)$ only between vertices of $\mathbb{T}$), for every edge e of $\mathbb{T}$ we compute the set χ_e of points (t_0, t_1) for which the plane $\mathbb{H}(t_0, t_1)$ intersects e: $\chi_e = \{(t_0, t_1) \in [0,1]^2 \,|\, \mathbb{H}(t_0, t_1) \cap e \neq \emptyset\}$. Boundaries of the sets χ_e for all edges e of $\mathbb{T}$ split the square $[0,1]^2$ into cells such that for any (t_0, t_1) within one cell σ the plane $\mathbb{H}(t_0, t_1)$ intersects the same edges of $\mathbb{T}$ and hence $\mathbb{T}(t_0, t_1)$ stays the same combinatorially; we will sometimes use $\mathbb{T}_\sigma$ to denote a generic terrain $\mathbb{T}(t_0, t_1)$ for $(t_0, t_1) \in \sigma$ (i.e., $\mathbb{T}_\sigma$ is a "combinatorial" terrain whose sequence of vertices is fixed, but the location of the vertices may change). It is straightforward to verify that boundary of χ_e is defined by two lines (Fig. 10); hence, since $\mathbb{T}$ has $O(n)$ edges, there are $O(n^2)$ cells σ. We thus have:

Lemma 17. *If feasible points within σ can be found in time $O(T_\sigma)$, the overall feasible set can be built in $O(n^2 T_\sigma)$ time.*

As with the aligned guards, we will consider feasibility of (t_0, t_1) separately for each edge ℓ of $\mathbb{T}_\sigma$. Specifically, let $\tau_\ell \subseteq \sigma$ be the set of points $(t_0, t_1) \in \sigma$ for which any point on the edge $\ell(t_0, t_1)$ of $\mathbb{T}(t_0, t_1)$ is seen by (at least one of) the guards $g_0(t_0), g_1(t_1)$; the feasible set (within the cell σ) is the intersection of the sets τ_ℓ for all edges ℓ of $\mathbb{T}_\sigma$. Since there are $O(n)$ edges in $\mathbb{T}_\sigma$, we have:

Lemma 18. *If τ_ℓ can be built in $O(T_\ell)$ time and has complexity $O(C_\ell)$, then $T_\sigma = O(nT_\ell + n^2 C_\ell^2)$.*

Extending definitions from Section 3.1, for a vertex v_m of $\mathbb{T}_\sigma$ let $\text{sh}_{v_m \to \ell}(t_0, t_1)$ be the shadow of $g_0(t_0)$ cast onto the edge $\ell(t_0, t_1)$ of $\mathbb{T}(t_0, t_1)$, i.e., the intersection of the ray $\overrightarrow{g_0(t_0) v_m(t_0, t_1)}$ with the edge. Similarly to Lemma 11, $\text{sh}_{v_m \to \ell}$ is a constant-degree algebraic function of (t_0, t_1) which can be computed in constant time (we omit the exact formulas

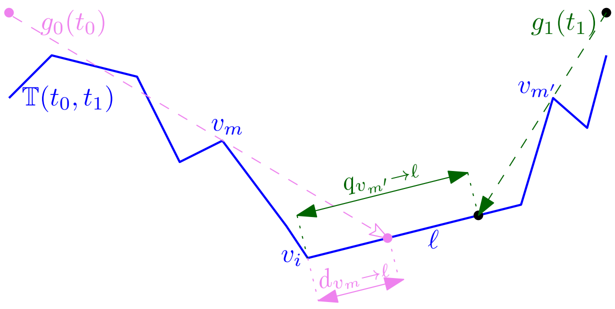

Figure 11: The distances $d_{v_m \to \ell}$ and $q_{v_m \to \ell}$.

since they are huge); also the distance $d_{v_m \to \ell}$ from the left endpoint $v_i(t_0, t_1)$ of ℓ to $\text{sh}_{v_m \to \ell}$ can be computed in constant time (cf. Lemma 12). Similarly, we can compute the distance $q_{v_m \to \ell}$ from $v_i(t_0, t_1)$ to the shadow of $g_1(t_1)$ cast by each vertex v_m with $m > i$ (Fig. 11).

We now state few notions and facts from theory of arrangements [13, Ch.23.3] using our terms. Given a collection $D = \{d_{v_m \to \ell}\}_{m \leq i}$ of functions $d_{v_m \to \ell} : [0,1]^2 \mapsto \mathbb{R}$, the *maximization diagram* of D is the subdivision of $[0,1]^2$ according to which function from the collection is maximum; i.e., within each cell of the diagram the same function from the collection is maximum. Let $D = \{d_{v_m \to \ell}\}_{m \leq i}, Q = \{q_{v_m \to \ell}\}_{m > i}$ be two sets of constant-degree algebraic functions, $O(n)$ functions in each. Let $d_\ell = \max_{m \leq i} d_{v_m \to \ell}$ be the upper envelope of D (this is the same definition as in Section 3.1); let $q_\ell = \max_{m > i} d_{v_m \to \ell}$ be the upper envelope of Q. Let $\mathcal{O}$ be the overlay of the maximization diagrams of D and Q; within each cell of $\mathcal{O}$ the same function from D defines d_ℓ and the same function from Q defines q_ℓ. Thus, given the overlay $\mathcal{O}$ one can compute, in constant time per cell, the set of cells for which $d_\ell \leq q_\ell$, or in other words, the set $\{(t_0, t_1) \in [0,1]^2 : d_\ell(t_0, t_1) \leq q_\ell(t_0, t_1)\}$; call this set the *q-set*. The overlay $\mathcal{O}$ can be computed in $O(n^3)$ time and has complexity $O(n^{2+\varepsilon})$ [5, 16]; see also end of Ch. 23.3 in [13]. Thus

Lemma 19. *The q-set has complexity $O(n^{2+\varepsilon})$ and can be computed in $O(n^3)$ time.*

Returning to our guards, analogously to the aligned-guards case (Section 4.1), some point on edge ℓ of $\mathbb{T}(t_0, t_1)$ is seen collectively by $g_0(t_0), g_1(t_1)$ iff the shadow of g_0 on ℓ is to the left of the shadow of g_1. That is, $\tau_\ell = \{(t_0, t_1) | d_\ell(t_0, t_1) \leq q_\ell(t_0, t_1)\}$, or in (other) words, τ_ℓ is exactly the q-set. From Lemma 19,

Lemma 20. *τ_ℓ has complexity $C_\ell = O(n^{2+\varepsilon})$ and can be built in $T_\ell = O(n^3)$ time.*

From Lemmas 17, 18 and 20, we obtain the main technical result of the paper:

Theorem 21. *It can be checked in $O(n^{8+\varepsilon})$ time whether the bottom and top guards can coordinate their motion so as to sweep $\mathbb{T}$.*

Non-straight tracks. When the two tracks are arbitrary polygonal paths, we compute feasibility of guards positions separately for every pair of edges of the tracks. This adds a factor of C^2 to the running time, where C is the total complexity of the tracks:

Corollary 22. *It can be checked in $O(C^2n^{8+\varepsilon})$ time whether two guards can coordinate their motion so as to sweep the part of $\mathbb{T}$ between their tracks.*

Multiple guards. Similarly to Section 4.1, for multiple guards moving along given non-crossing tracks we can check whether they can form a chain sweeping $\mathbb{T}$ by doing pairwise checks for consecutive guards in the chain; the checking time can be again charged to the complexity of the terrain between the consecutive guards.

Corollary 23. *It can be checked in $O(KC^2n^{8+\varepsilon})$ time whether K guards moving along given tracks, each track of complexity C, can coordinate their motion so as to sweep $\mathbb{T}$.*

5. OPTIMIZING THE TRACKS

We now consider the version of the problem in which the tracks are not given in the input but instead must be *output*; the goal is to minimize the number of guards (equivalently, the number of tracks) needed to sweep the terrain. (We will not be concerned with finding trajectories for the guards; as soon as the tracks are known, the results in the previous section apply for finding the trajectories.) It can be shown (we omit the argument) that the tracks can be found in the greedy, "bottommost" fashion: find the highest track π_1 for g_1 such that g_1 and the bottom guard g_0 collectively see the part of $\mathbb{T}$ between $\mathcal{B}$ and π_1; recurse. (Here, "highest" refers to the largest y-coordinate, meaning "as far from the bottom as only possible, at every x-coordinate, without jeopardizing the visibility.) It follows that the general problem (computing multiple tracks) reduces to finding just one track running as far up as possible from a given track. (This is similar to the feasibility versions in the preceding sections, where feasibility checking for an arbitrary number of guards reduced to pairwise-checks.)

5.1 Aligned guards

As in Section 4.1, our solution is based on tracking how the shadows from g_0, g_1 move along each edge of the terrain $\mathbb{T}(t)$ (recall that $\mathbb{T}(t) = \mathbb{T} \cap \{x = t\}$); we again restrict ourselves to an interval $I \subseteq [0,1]$ of the times t between consecutive x-coordinates of $\mathbb{T}$ (so that $\mathbb{T}(t)$ does not change combinatorially). The shadow from g_0 is built exactly like above (Section 3.1): for any t and every edge $\ell = v_iv_{i+1}$ of $\mathbb{T}(t)$ we build the function $d_\ell(t)$ indicating the distance from v_i to the rightmost shadow from g_0 on ℓ. The shadow from g_1, however, will now be a function of *two* variables: the first one, t, is the x-coordinate of g_1 (as before); the second variable is the guard's *y-coordinate* (Fig. 12). For every t, y we define $q_\ell(t, y)$ as the distance from $v_i(t)$ to the leftmost shadow of g_1, assuming the guard is at $(t, y, 1)$. As in Section 4.1, q_ℓ is the upper envelope of $O(n)$ functions (but now each function is two-variate) – one function per vertex v_m of $\mathbb{T}(t)$ for $m > i$; each function is constant-degree algebraic and is computable in constant time. Also as in Section 4.1, some point on ℓ is not seen by $g_0(t)$ and $g_1(t, y)$ iff $d_\ell(t) > q_\ell(t, y)$. After computing the functions d_ℓ and q_ℓ we can, for each t, find the maximum y for which $d_\ell(t) \leq q_\ell(t, y)$; these maximums altogether define the function $y_\ell^{\max}(t) = \max\{y \,|\, d_\ell(t) \leq q_\ell(t, y)\}$ that indicates, for each t, how far g_1 can be while still maintaining that $\mathbb{T}(t)$ is seen by g_0, g_1. Finally, the sought track π_1 (more precisely, its part π_1^I over the interval I) is the lower envelope

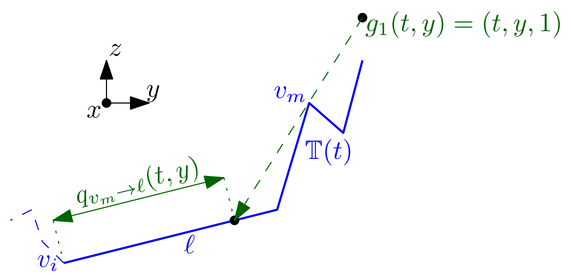

Figure 12: The y-coordinate of g_1 can be arbitraty (not fixed to 1, as it was when g_1 moved along $\mathcal{T}$).

of these functions for all edges of ℓ: $\pi_1^I = \{x, y, z \,|\, x \in I, y = \min_\ell y_\ell^{\max}(x), z = 1\}$.

We now analyze the complexity of the above construction. The function $y_\ell^{\max}$ for a single edge ℓ is determined by the overlay of maximization diagram of d_ℓ and q_ℓ; as in Section 4.2, the function thus has $O(n^{2+\varepsilon})$ complexity and can be computed in $O(n^3)$ time. Hence the track π_1^I (the lower envelope of $O(n)$ such functions) trivially has $O(n^{6+\varepsilon})$ complexity and can be computed in the same time. Since there are $O(n)$ intervals I, we have:

Lemma 24. *π_1 can be computed in $O(n^{7+\varepsilon})$ time.*

We can compute the next track π_2 similarly to π_1. Unfortunately, the running time for computing π_2, following from our analysis, is $O(n^{13+\varepsilon})$. Indeed, we would have to compute π_2^J (in $O(n^{6+\varepsilon})$ time) separately for each interval $J \in [0,1]$ in which π_1 is the graph of the same function, and the only bound we have on the number of intervals is the complexity of π_1, i.e., $O(n^{7+\varepsilon})$. This means that our analysis does not lead to a polynomial-time algorithm for computing the tracks π_k for non-constant k. (We believe that more efficient algorithms are possible for computing $y_\ell^{\max}$; still, it can be the case that the complexity of the track π_k does grow super-polynomially with k.)

On the other hand, if the tracks are required to be straight, then the furthest-from-$\mathcal{B}$ feasible track $\overline{\pi_1}$ should run at the smallest y-coordinate of π_1 (which can be found in $O(n^{7+\varepsilon})$ time, by Lemma 24). Thus, we can keep computing the tracks $\overline{\pi_2}, \overline{\pi_3}, \ldots$, without the blowup in complexity, until the track $\overline{\pi_K}$ (where K is the optimal number of tracks) goes over $\mathcal{T}$. From Lemma 24 we have:

Theorem 25. *The minimum number K of straight tracks needed to sweep $\mathbb{T}$ with aligned guards can be found in $O(Kn^{7+\varepsilon})$ time.*

5.2 Coordinated guards

We now find the minimum number of tracks for guards that can coordinate their motion. We use free-space-diagram ideas like in Section 4.2, but now the free space has the third dimension y (the y-coordinate of g_1) and we look for the largest ω for which the cross-section of the free space by the plane $y = \omega$ still has the required path from $(t_0, t_1, y) = (0, 0, \omega)$ to $(t_0, t_1, y) = (1, 1, \omega)$. (This is analogous to solving the *optimization* version of the Fréchet distance problem—computing the smallest leash length L—by finding for which L the free space seizes to have the necessary path.) As in Section 4.2, we split the (t_0, t_1, y)-space into $O(n^3)$ cells within which the vertical plane through $(t_0, 0, 1), (t_1, y, 1)$

crosses $\mathbb{T}$ in combinatorially the same terrain $\mathbb{T}(t_0, t_1, y)$; we, again, build the free space (vertices) separately in each cell σ. Also as before, for each edge $\ell = v_i v_{i+1}$ of $\mathbb{T}_\sigma$ we build the shadows from g_0 and g_1, and define the distance functions $d_\ell(t_0, t_1), q_\ell(t_0, t_1, y)$ indicating how far the shadows are from v_i (the difference is that q_ℓ is now tri-variate). The free space as far as seeing the edge ℓ is concerned, i.e., the set $F_\ell = \{(t_0, t_1, y) \mid d_\ell \leq q_\ell\}$, is equal to the q-set from the overlay of the maximization diagrams of d_ℓ and q_ℓ. Finally, the free space within σ is the intersection of the cells F_ℓ for all edges ℓ of the combinatorial terrain $\mathbb{T}_\sigma$. Overall we obtain:

Theorem 26. *The minimum number of straight tracks needed to sweep $\mathbb{T}$ can be found in polynomial time.*

6. CONCLUSION

We considered sweeping terrains by chains of flying guards and gave algorithms for several versions of problems arising in this context. Our solutions can be extended in multiple ways. The assumption that $\mathbb{S}$ is the unit square at unit height was made only to simplify the notation; the solutions work for $\mathbb{S}$ being an arbitrary rectangle (and, actually, an arbitrary polygon) situated at an arbitrary elevation (in fact, we can also *find* the minimum elevation at which the guards have to fly in order to seep the terrain, i.e., solve yet another optimization problem). In the feasibility versions, it is straightforward to handle tracks that are arbitrary curves in 3D (not necessarily staying at the same elevation). Bounds on guards speed can be imposed by bounding slopes of feasible paths through the free-space diagrams.

We also reported on an implementation and provided videos with its output. Our code is freely available. The implementation can be extended to optimize distance traveled, time to complete the mission, fuel consumption (which may depend on the change of elevation of flight) and other objectives; it may also be made more interactive.

The biggest problem that we left open is minimizing the number of guards that are allowed to move on arbitrary tracks; we conjecture that for aligned guards this can be done in polynomial time while for coordinated guards the problem is NP-hard. As a little consolation we remark that our algorithms for straight-track guards from Section 5 extend to the case of constant-degree-of-freedom tracks, i.e., tracks coming from a family of curves parameterized by a constant number of parameters (e.g., arbitrary lines in $\mathbb{S}$ or even in 3D).

7. REFERENCES

[1] `http://www.minecraftforum.net/topic/992750-mapping-using-real-world-terrain-data/`.

[2] `http://www.ambiotek.com/topoview`.

[3] `http://www.usna.edu/Users/oceano/pguth/website/microdem/microdem.htm`.

[4] `http://www.cs.helsinki.fi/en/compfac/high-performance-cluster-ukko`.

[5] P. K. Agarwal, O. Schwarzkopf, and M. Sharir. The overlay of lower envelopes and its applications. *Discrete Comput. Geom.*, 15:1–13, 1996.

[6] P. K. Agarwal and M. Sharir. Davenport-Schinzel sequences and their geometric applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 1–47. 2000.

[7] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995.

[8] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk. Maximum thick paths in static and dynamic environments. *CGTA*, 43(3):279–294, 2010.

[9] B. Bollobás, I. Leader, and M. Walters. Lion and man – can both win? *Israel Journal of Mathematics*, 189(1):267–286, 2012.

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

[11] A. Dumitrescu, I. Suzuki, and P. Zylinski. Offline variants of the lion and man problem. *Theoretical Computer Science*, 399(3):220 – 235, 2008.

[12] A. Efrat, J. S. B. Mitchell, S. Sankararaman, and P. Myers. Efficient algorithms for pursuing moving evaders in terrains. In *ACM GIS'12*.

[13] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC, 1997.

[14] J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *IPL*, 33(4):169–174, 1989.

[15] T. Kamphans. *Models and algorithms for online exploration and search.* PhD thesis, 2011.

[16] V. Koltun and M. Sharir. The partition technique for overlays of envelopes. *SIJCOMP*, 32(4):841–863, 2003.

[17] J. S. B. Mitchell. Approximating watchman routes. In *SODA'13.*

[18] J. S. B. Mitchell. On maximum flows in polyhedral domains. *J. Comp. Syst. Sci.*, 40:88–123, 1990.

[19] J. S. B. Mitchell and V. Polishchuk. Thick non-crossing paths and minimum-cost flows in polygonal domains. In *SoCG'07.*

[20] J. O'Rourke. *Art Gallery Theorems and Algorithms.* 1987.

[21] E. Packer. Computing multiple watchman routes. In *WEA'08.*

[22] G. Rote. Pursuit-evasion with imprecise target location. In *SODA'03.*

[23] L. H. Tseng, P. J. Heffernan, and D. T. Lee. Two-guard walkability of simple polygons. *Int. J. Comput. Geometry Appl.*, 8(1):85–116, 1998.