Computing Highly Occluded Paths Using a Sparse Network*

Niel Lebeck
University of Washington
nl35@cs.washington.edu

Thomas Mølhave
SCALGO USA
thomas@scalgo.com

Pankaj K. Agarwal
Duke University
pankaj@cs.duke.edu

ABSTRACT

Computing paths over a terrain that are highly occluded with respect to observers is an important problem in GIS. Given a fast algorithm for computing the visibility map, the path-planning step becomes the bottleneck. In this paper, we present an approach for quickly computing occluded paths over a terrain using a *sparse network*, a sparse 1-dimensional network over the terrain. We present different strategies for constructing the sparse network. Experimental results show that our approach results in significantly improved time for computing highly occluded paths between two query points, and that the different strategies offer a tradeoff between higher-quality paths and lower preprocessing times. Furthermore, there are strategies that achieve near-optimal paths with small preprocessing cost.

Categories and Subject Descriptors: F.2.2 [Nonnumerical Algorithms and Problems]: Geometrical problems and computations; H.2.8 [Database Management]: Database Applications—*Data Mining, Image Databases, Spatial Databases and GIS*

General Terms: Performance, Algorithms

Keywords: Terrain modeling, GIS, visibility, navigation

1. INTRODUCTION

Big terrain data sets are being collected and regularly updated by federal, state, and local government agencies, as well as by private companies, at an unprecedented rate, and the demand for these data sets is increasing. For example, the US Geological Survey (USGS) produces the regularly updated National Elevation Dataset (NED) which includes a national 1/3-arc second ($\sim$ 10 meter) digital elevation model (DEM) as well as 1/9-arc second ($\sim$ 3 meter) resolution DEMs for parts of the U.S. where there is sufficient coverage. But there is rising demand to map most of the US at 1/27-arc second ($\sim$ 1 meter) resolution, because bigger resolution can make a big difference in terrain analysis. While such high-resolution DEMs provide unprecedented opportunities, their large size requires the availability of efficient algorithms for terrain modeling and analysis, which is often the bottleneck in taking full advantage of these applications. This bottleneck is particularly apparent for path planning and visibility analysis on terrains.

This paper considers a visibility-based path-planning problem. Namely, the problem of computing a highly *occluded* path between two points a and b on a terrain Σ with respect to a set O of observers. This problem has a wide range of applications, including military applications (e.g. planning troop movements in enemy territory), city planning (e.g. planning the location of visually unappealing construction projects such as power lines), and virtual environments (e.g. video games).

Computing highly occluded paths requires performing two main operations on the terrain. Given a set O of observers, first compute the visibility information of the observers and aggregating this visibility information to compute the *coverage-map*, which for each point on the terrain Σ tells how visible it is from O. A minimal-cost path algorithm is then run using the coverage-map. Both of these steps are computationally expensive. In an earlier paper [14], we presented an algorithm for quickly computing visibility information in a manner that scales well to large terrains, using approximation techniques and the GPU (see also [10]). Existing shortest-path algorithms such as Dijkstra's algorithm and A* that run over the entire terrain do not scale well, and finding the shortest path becomes the bottleneck in the occluded path computation. This step becomes even more critical when we wish to perform repeated queries of finding highly occluded paths on Σ, the scenario in which we are interested. This raises the question whether highly-occluded-path queries can be answered more quickly by performing some preprocessing on the terrain.

*Work by P.A. & N.L. is supported by NSF under grants CCF-09-40671, CCF-10-12254, and CCF-11-61359, by Grant 2012/229 from the U.S.-Israel Binational Science Foundation, and by an ERDC contract W9132V-11-C-0003. Work by P.A. & T.M. is supported by U.S. Army Research Office contract W911NF-13-P-0018. Part of the work was done while the first author was at Duke University.

SIGSPATIAL'14, November 04 - 07 2014, Dallas/Fort Worth, TX, USA

http://dx.doi.org/10.1145/2666310.2666394

In this paper, we answer this question affirmatively by describing an algorithm that preprocesses the coverage map, extracts a sparse 1-dimensional network from the coverage map, and uses this network for answering queries.

Related work. There is extensive work in computational geometry, robotics, GIS, and virtual environment communities on path planning and visibility related problems. Here we focus on the work that is most closely related to our results. We refer the reader to the books [13, 20] for a review of visibility based planning and related problems.

Most of the research on optimal path-planning on terrains in computational geometry has focused on finding shortest paths. The best known algorithm takes quadratic time [7] , and there are faster sampling based approximation algorithms [21, 4]There is also some work on the so-called weighted-region problem, where a weighted planar subdivision is given and the goal is to find a path of the minimum weighted length. The exact algorithms are quite expensive in 2D, but faster approximation algorithms are known; see [3] and references therein. One can assign the weights of a region based on visibility of that region from a given set of observers and can formulate the problem of computing a highly occluded path as a weighted-region problem. However, this approach is not scalable because the number of regions grow rapidly with the size of the terrain and with the number of observers, and the algorithm is impractical even for small-size terrains.

There is work in GIS and virtual environment communities on finding paths that are occluded from a given set of observers [11, 15, 16]. These algorithms sample a set of points on the terrain or polygonal environment, assign a weight to each point based on the visibility from a given set of observers, and use a steepest-descent, A* search, or Dijkstra's algorithm to compute a desired path. However, many of these algorithms do not seem to scale to large terrains. For example, the largest terrains on which the algorithms in [15] were tested had size 257×257. Franklin et. al [11] propose a terrain compression algorithm and compute the paths on the compressed terrain. They present experimental results on terrains of sizes up to $160,000$ (400×400) grid points. Ferreira et al. [10] presented a scalable algorithm for computing viewsheds on large terrains, but it does not study path-planning.

Motivated by on-line navigation systems, several algorithms have been proposed for answering shortest-path queries on road networks [1, 22]. Most of these algorithms rely on a landmark based approach, i.e., it suffices to compute shortest paths between a few landmarks. The shortest paths between other pairs of nodes can be quickly computed by using the pre-computed shortest paths and connecting the query points to these paths locally. There is, however, relatively little work on shortest path queries in a continuous space such as a terrain. Agarwal et al. [2] presented a data structure for computing an approximate shortest path between two query points inside a polygonal region (possibly with holes). There is no known data structure for answering shortest-path queries on a terrain between two query points, with provable guarantees on its performance, but there are data structures for answering shortest-path queries on a terrain from a fixed source point [2]. However, the size of the data structures is at least linear in the size of the terrain.

Several approaches in robotics to the path planning problem rely on constructing a 1-dimensional network: algorithms for computing a shortest path in a planar domain constructs a 1D network such as the visibility graph or a spanner [13]. The algorithms for computing a maximum clearance path, i.e., a path that stays as far away from the boundary of the domain as possible, construct the medial axis of the domain [17]. The seminal path-planning algorithm by Canny [6] constructs a 1-dimensional network using ideas from Morse Theory and differential topology. Since then several path-planning algorithms based on 1D networks have been proposed; see e.g. the book [13]. A widely used approach for path planning is the so-called probabilistic road map (PRM) approach [13], which samples a set of random points and connects nearby points by simple paths (e.g. line segments) that lie inside the domain. It is worth pointing out that this line of work has not focused on constructing a sparse network, but on constructing a network that leads to good paths. Only recently has there been some work on compressing these networks [19].

Finally we note that the approach of transforming a 2D or 3D domain to a 1D network has been used in other GIS problems as well, e.g., hydrology analysis on terrains [8].

Our contributions. The main contribution of this paper is to demonstrate that a 1D network based approach is effective for answering a highly-occluded-path query between two points on a terrain. That is, we can construct a small size 1D network on the terrain that can be used to efficiently construct a highly occluded path between any two query points. The 1D network decomposes the terrain into regions. Given two query points, we first connect them to the boundary of the regions in which they lie and then compute a path between these boundary points along the 1D network. As mentioned above, such an approach has been effectively used for shortest-path queries in a road network [1, 22], but we are unaware of this approach being used for constructing a highly occluded path on a terrain.

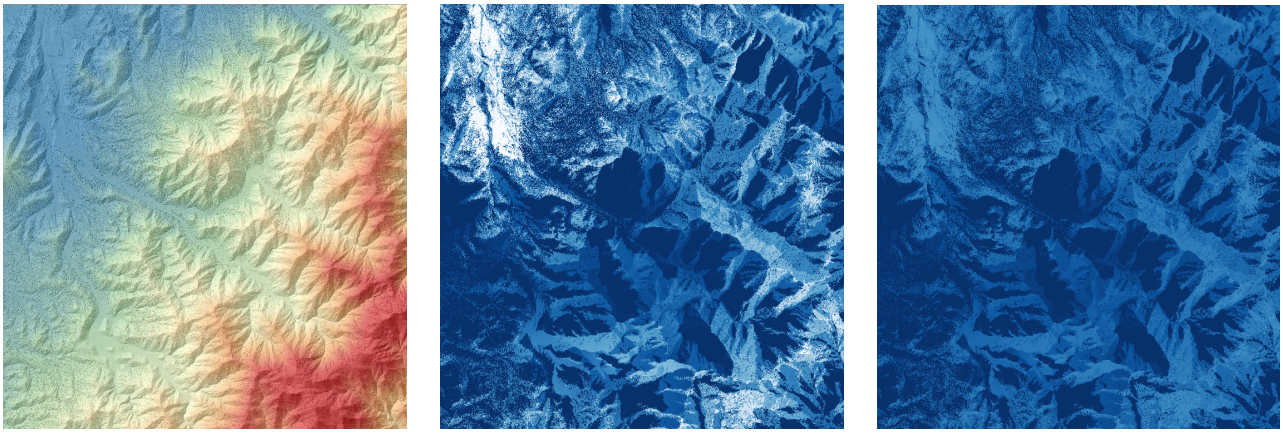
The paths computed using a sparse network are likely to be high-quality if the sparse network closely aligns with the lowest-cost subpaths over the terrain. On the other hand, if the subpaths in the sparse network are relatively high-cost, the sparse-network paths are likely to be expensive as well. As a result, the algorithm used to construct the sparse network is important. We propose and investigate three strategies for constructing a 1D network:

(i) a *learning based approach* which computes highly occluded paths between a family of pairs of points and uses them to construct the network.

(ii) a *sampling based approach*, a variant of the PRM approach mentioned above.

(iii) a *topology based approach* that traces one-dimensional critical curves of the coverage map of the terrain.

These approaches provide a trade-off between the preprocessing cost and the quality of the path. They are all based on the coverage map of a terrain, which is computed using the algorithm described in [14].



(a) (b) (c)

Figure 1. (a) The Afghanistan terrain, (b) the sum of the visibility maps for a set of observers O, and (c) the coverage map ω_O with $C = 1$ and $\alpha = 0.9$. In (a), red shades indicate higher elevations, and in (b) and (c), lighter shades of blue indicate higher values.

We report detailed experimental results to evaluate the effectiveness of our approach. The learning based approach produces a network that gives the highest-quality paths, but with a high preprocessing cost. The topology based approach generates a sparse network quickly, but the resulting paths are relatively low quality. The sampling based approach provides the best trade-off between path quality and preprocessing time, with relatively fast network construction and high-quality paths. The tradeoff between preprocessing time and path quality for each of these strategies can be further tuned by changing their parameters.

2. HIGHLY OCCLUDED PATHS

Let $\mathbb{M} \in \mathbb{R}^2$ be a convex domain, typically a simple connected region such as a rectangle or a disk, and let $h : \mathbb{M} \to \mathbb{R}$ be a *height* (elevation) function defined over $\mathbb{M}$. The graph of h is called a *terrain*, denoted by Σ, i.e., $\Sigma = \{(x, h(x)) \mid x \in \mathbb{M}\}$; Σ is a two-dimensional surface embedded in $\mathbb{R}^3$. For a point $p \in \mathbb{M}$, we use $\hat{p} = (p, h(p))$ to denote the corresponding point on Σ. For any two points $a, b \in \mathbb{R}^3$, a is *visible* from b, and vice-versa, if no point on segment ab lies below Σ, i.e., for any $\overline{q} \in ab$, with $\overline{q} = (q, z)$ where $q \in \mathbb{M}$ and $z \in \mathbb{R}$, we have $z \geq h(q)$. Otherwise a is *occluded* from b.

Let $o \in \mathbb{R}^3$ be an *observer* lying above Σ. We fix a parameter $r_{\max} > 0$. We assume that o has limited visibility in the sense that no point farther than $r_{\max}$ away from o is visible from o, and that the quality of visibility deteriorates with distance. More precisely, we define the visibility of a point $p \in \mathbb{M}$, or rather $\hat{p} \in \Sigma$, as

$$V_o(p) = \begin{cases} 1 - \min\left\{\frac{\|p-o\|}{r_{\max}}, 1\right\} & \text{if } \hat{p} \text{ is visible from } o, \\ 0 & \text{otherwise.} \end{cases}$$

That is, the visibility attenuates linearly with distance. One can use another kernel function (e.g. Gaussian kernel) to model the attenuation of visibility with distance. Furthermore, we assume that o is omnidirectional, i.e., it can see up to distance $r_{\max}$ in all directions. One can choose a more sophisticated model in which the visibility attenuation depends on the direction.

Let $O = \{o_1, \ldots, o_m\} \subseteq \mathbb{R}^3$ be a set of observers, each lying above Σ. We define the coverage map $\omega_O : \mathbb{M} \to \mathbb{R}_{\geq 0}$, with respect to O, as

$$\omega_O(p) = C\left(1 - \alpha \exp\left(\sum_{o_i \in O} V_{o_i}(p)\right)\right), \tag{1}$$

where $\alpha \in [0, 1]$ and $C > 0$ are constants. Figure 2 illustrates the form of this function. Note that our choice of function for the coverage map gives decreasing marginal score increases to additional observers. That is, visibility at p increasing from 0 to 1 has much more impact than from 10 to 11, as the latter point is already quite visible, so increase in visibility has little impact. Figure 1 shows the relationship among the terrain, the sum of the visibility maps, and the coverage map.

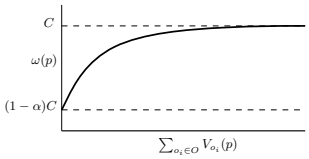


Figure 2. The functional form of the coverage map $\omega(p)$.

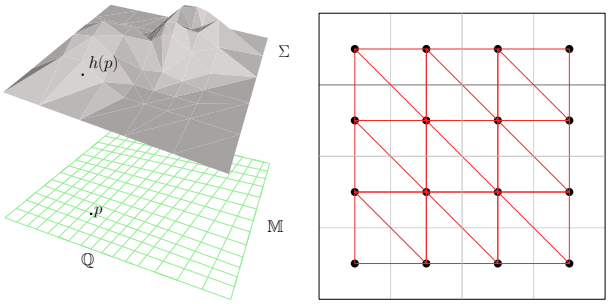
The cost of a path $\Pi \in \Sigma$ is defined to be

$$\omega_O(\Pi) = \int_\Pi \omega_O(p)\, \mathrm{d}p. \tag{2}$$

Since the set O will be fixed and will not be important for our discussion, we drop the subscript O from ω. So far we have considered an arbitrary terrain. A widely used *digital elevation model* (DEM) for a terrain is the so-called grid DEM. That is, we have a parameter $\rho > 0$ and assume that $\mathbb{M}$ is a square of side length $\rho 2^L$ for some integer $L \geq 0$. We partition $\mathbb{M}$ into $2^L \times 2^L$ *grid cells*, each of length ρ. For each grid cell (i, j), $0 \leq i, j < L$, let q_{ij} denote its center. Set $\mathbb{Q} = \{q_{ij} \mid 0 \leq i, j < 2^L\}$. Construct a triangulation of $\mathbb{M}$ by inserting edges between grid points. For a point $q_{ij} \in \mathbb{Q}$ we insert edges to horizontally adjacent grid point $q_{(i+1)j}$, to vertically adjacent grid point $q_{i(j+1)}$, and to the

diagonally adjacent grid point $q_{(i+1)(j+1)}$. Let $\Delta(\mathbb{Q})$ denote the resulting 2D triangulation.

The height function h specifies the height of all points in $\mathbb{Q}$. This is the grid DEM of the terrain. We can construct a triangulated surface Σ over $\mathbb{M}$ by lifting each triangle $q_{i_1j_1}q_{i_2j_2}q_{i_3j_3} \in \Delta(\mathbb{Q})$ to $\hat{q}_{i_1j_1}\hat{q}_{i_2j_2}\hat{q}_{i_3j_3}$; see Figure 3.



(a) (b)

Figure 3. (a) The terrain Σ corresponding to a grid $\mathbb{Q}$. (b) The triangulation $\Delta(\mathbb{Q})$ corresponding to the grid.

In a grid DEM, we consider paths along the edges of the *grid graph* $\Delta(\mathbb{Q})$, or rather along the edges of Σ. We consider the weighted graph $\mathcal{G} = (\mathbb{Q}, \mathcal{E})$, where $\mathcal{E}$ is the set of edges of $\Delta(\mathbb{Q})$. The weight of an edge $(q_1, q_2) \in \mathcal{E}$, denoted by $\mathtt{w}(q_1, q_2)$, is

$$\mathtt{w}(q_1, q_2) = \frac{\omega(q_1) + \omega(q_2)}{2} \|\hat{q}_1 - \hat{q}_2\|,$$

which is a finite approximation of the integral in 2 over the edge (q_1, q_2). Note that we consider the distance of the edge $(\hat{q}_1, \hat{q}_1)$ in Σ. The cost of a path $\Pi = q_0 q_1 \ldots q_k$ in $\mathcal{G}$ is defined as

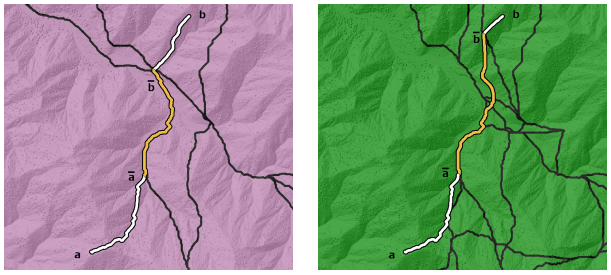
$$\mathtt{w}(\Pi) = \sum_{i=0}^{k-1} \mathtt{w}(q_i, q_{i+1}).$$

Given two grid points $q, q' \in \mathbb{Q}$, let $\Omega(q, q')$ denote the minimum-cost path in $\mathcal{G}$ (under the weight function $\mathtt{w}$). We refer to $\Omega(q, q')$ as the *most occluded path* between q and q' in $\mathcal{G}$.

3. SPARSE NETWORKS

A *sparse network* $\mathcal{S} = (\mathbb{X}, \Gamma)$ is a connected subgraph of the grid graph $\mathcal{G} = (\mathbb{Q}, \mathcal{E})$ with no degree one vertices in the interior of $\mathbb{M}$, which induces a planar decomposition of $\mathbb{M}$. The subgraph of $\mathcal{G}$ induced by $\mathbb{Q} \setminus \mathbb{X}$, i.e., removing the grid points of $\mathbb{X}$ and all the edges of $\mathcal{G}$ incident on the nodes of $\mathbb{X}$, consists of a set of connected components. The grid cells of $\mathbb{M}$ corresponding to each connected component is called a *face* of $\mathcal{S}$. For a grid point $q \in \mathbb{Q} \setminus \mathbb{X}$, let φ_q be the unique face of $\mathcal{S}$ that contains q.

$\mathcal{S}$ can be used to compute low-cost paths between two grid points $q, q' \in \mathbb{Q}$, by connecting q and q' to $\mathcal{S}$ and then finding a minimum-cost path in $\mathcal{S}$. For two points $a, b \in \mathbb{X}$, let $\bar{\Omega}_{\mathcal{S}}(a, b)$ denote the minimum-cost path in $\mathcal{S}$. Recall that $\mathcal{S}$ is connected, so a path between a and b exists. For a pair of points $a, b \in \mathbb{Q} \setminus \mathbb{X}$, we define the path $\bar{\Omega}_{\mathcal{S}}(a, b)$ as follows. If



(a) (b)

Figure 4. Two sparse networks and the paths between two points a and b that they produce.

a and b lie in the same face of $\mathcal{S}$, i.e., $\varphi_a = \varphi_b$, then $\bar{\Omega}_{\mathcal{S}}(a, b)$ is the minimum-cost path between a and b restricted to the face φ_a. So assume that they lie in different faces of $\mathcal{S}$. Let $\bar{a}$ (resp. $\bar{b}$) denote the point in $\mathbb{X}$ closest to a (resp. b), i.e., $\bar{a} = \arg\min_{a' \in \mathbb{X}} \mathtt{w}(a, a')$. For simplicity, let us assume that $\bar{a}$ and $\bar{b}$ are uniquely defined. Note that $\bar{a}$ (resp. $\bar{b}$) lies on the boundary of the face that contains a (resp. b). If $a \in \mathbb{X}$, then $\bar{a} = a$. We now define

$$\bar{\Omega}_{\mathcal{S}}(a, b) = \Omega_{\mathcal{S}}(a, \bar{a}) \circ \bar{\Omega}_{\mathcal{S}}(\bar{a}, \bar{b}) \circ \Omega_{\mathcal{S}}(\bar{b}, b).$$

We set $\bar{\mathtt{w}}_{\mathcal{S}}(a, b) = \mathtt{w}(\bar{\Omega}_{\mathcal{S}}(a, b))$; See Figure 4. If the network $\mathcal{S}$ is obvious from the context, we omit the subscript $\mathcal{S}$.

The query procedure has two components: (i) computing $\bar{a}$, $\bar{b}$, $\Omega(a, \bar{a})$, and $\Omega(b, \bar{b})$; and (ii) computing $\bar{\Omega}(\bar{a}, \bar{b})$. For very sparse networks the first term dominates, and as the network becomes dense, the second term dominates, so from the query-time point of view, one needs to construct $\mathcal{S}$ in which the two terms are balanced.

We wish to construct a network $\mathcal{S}$ such that $\bar{\mathtt{w}}(a, b)$ is close to $\mathtt{w}(a, b)$. We note that this is trivially true in the two extreme cases. Namely, if $\mathcal{S}$ is very sparse (say, empty in the extreme case), then a and b are likely to lie in the same face of $\mathcal{S}$ and so $\bar{\Omega}(a, b) = \Omega(a, b)$. On the other hand, if $\mathcal{S}$ is very dense (say, $\mathcal{G}$ itself in the extreme case), then $\bar{a}$ (resp. $\bar{b}$) is very close to a (resp. b) and $\bar{\mathtt{w}}(\bar{a}, \bar{b}) \approx \mathtt{w}(\bar{a}, \bar{b}) \approx \mathtt{w}(a, b)$. However, in either case the time spent in computing $\bar{\Omega}(a, b)$ is roughly the same as computing $\Omega(a, b)$ and thus constructing $\mathcal{S}$ does not help.

Our goal is therefore to *construct a network so that not only is $\mathbb{X}$ small, but each face of $\mathcal{S}$ is also small, and for any two points $a, b \in \mathbb{X}$, $\bar{\mathtt{w}}(a, b)$ is close to $\mathtt{w}(a, b)$.* In the next section, we describe a number of strategies to construct such networks.

We remark that we could have used a more compact representation of $\mathcal{S}$ by removing all degree-two nodes. More precisely, let $\tilde{\mathbb{X}} \subseteq \mathbb{X}$ be the set of nodes in $\mathcal{S}$ with degree more than two. We can define $\tilde{\mathcal{S}} = (\tilde{\mathbb{X}}, \tilde{\Gamma})$, where $x, y \in \tilde{\mathcal{S}}$ are connected by an edge if there is a path from x to y that does not pass through any other node of $\tilde{\mathcal{S}}$. Although this representation is more compact, we decided not to use it because we have to store $\mathcal{S}$ anyway to compute φ_a and φ_b and to reconstruct the path $\bar{\Omega}(a, b)$, and the saving in the overall query time does not seem to be significant if we run Dijkstra's algorithm on $\tilde{\mathcal{S}}$ as compared to running the A* algorithm on $\mathcal{S} \cup \varphi_a \cup \varphi_b$.

4. CONSTRUCTING THE NETWORK

This section describes three different approaches for constructing a sparse network on Σ, based on the coverage map, with varying preprocessing costs and path quality. We assume that the locations of the observers are known [1], and that we have computed the coverage map $\omega : \mathbb{M} \to \mathbb{R}_{\geq 0}$ using the algorithm in [14].

4.1 Learning based approach

This approach is motivated by the observation that for a given coverage map, most occluded paths over a terrain tend to follow similar or identical trajectories when moving over the same region, even if the start and end points of the paths are very different—only the initial and final protions of the paths are different. As a result, we propose a learning based approach to constructing the sparse network. It computes highly occluded paths between many pairs of points and builds a sparse network from the common sub-paths on these paths. The initial path computation uses a shortest-path algorithm over the entire terrain, resulting in an expensive preprocessing step. The goal is, however, to construct a sparse network that will speed up future queries. The algorithm consists of three stages: The first stage computes optimal paths between a family of pairs of points; the second stage identifies the subpaths used by many paths; the third stage completes these subpaths into a connected network; see Figure 5.

First stage. We fix two parameters m and n. We choose a set $A = \{a_1, \ldots, a_n\}$ of n *source* points on the boundary of the terrain Σ, i.e., on the edges of the square $\mathbb{M}$. For each $i \leq n$, we choose a set $B_i = \{b_{i1}, \ldots, b_{im}\}$ of m *destination* points, also on the boundary of Σ. For each i, we compute the most occluded paths $\Omega(a_i, b_{ij})$ from a_i to all b_{ij}, for $1 \leq j \leq m$, using, say, Dijkstra's algorithm. Let $\mathcal{P}$ be the resulting set of mn paths.

Many different strategies can be used for choosing A and B_i's, but we use a simple random strategy to choose them. One could have chosen mn different pairs of source-destination points, but then the preprocessing cost would have been even higher, and the current approach also does a better job of identifying the shared structure of paths.

Second stage. We compute the *heat-map* $\mu : \mathbb{Q} \to \mathbb{R}_{\geq 0}$ such that for a point $q \in \mathbb{Q}$, $\mu(q)$ is the number of paths of $\mathcal{P}$ passing through q; μ gives information about the common sub-paths shared by the paths in $\mathcal{P}$. Figure 5(a) shows a small example of a heat-map.

Next, we fix a threshold $t \geq 0$ and compute the *thresholded heat-map* $\mu_t : \mathbb{Q} \to \{0, 1\}$, where for a grid point $q \in \mu$,

$$\mu_t(q) = \begin{cases} 1 & \text{if } \mu(q) \geq t, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

[1] If the locations of observers are not known, we can guess their locations, using an observer-placement algorithm, e.g., the one in [14].

Let $\mathbb{X}_1 \subseteq \mu$ be the set of points q for which $\mu_t(q)$ is 1. We construct an initial network $\mathcal{S}_1 = (\mathbb{X}_1, \Gamma_1)$, which is the subgraph of the grid graph $\mathcal{G}$ induced by $\mathbb{X}_1$. See Figure 5(b).

Third stage. $\mathcal{S}_1$ gives the highest-intensity common subpaths, but $\mathcal{S}_1$ need not be a connected 1D network. We therefore use postprocessing to convert $\mathcal{S}_1$ into a connected 1D network $\mathcal{S}$ with no degree one vertices in the interior of $\mathbb{M}$. First, the thresholded heat-map μ_t can have multiple dense clusters of non-zero points resulting in trivially small faces of $\mathcal{S}_1$, we delete these faces by contracting the edges on their boundaries. $\mathcal{S}_1$ may have multiple connected components $C_1, \ldots, C_k$. We connect them using a simple strategy that finds the cheapest path Π_{ij} between any two components

$$(i^*, j^*) = \arg \min_{1 \leq i \neq j \leq k} \{\mathtt{w}(q_i, q_j) \mid q_i \in C_i, q_j \in C_j\},$$

and connects C_{i^*}, C_{j^*} by adding the path Π_{ij} to the network. We can then connect C_i and C_j along this path and iterate on the resulting reduced set of connected components. Finally, the network can contain dangling paths ending in degree-1 nodes. These paths can be removed, as they do not affect connectivity. However, if a path ending in degree-1 node q is long, it can be useful for the network. In such a case we extend the path using a simple greedy algorithm. Let q' be the neighbor of q neighbor in $\mathcal{S}_1$. We extend the path from q by adding the point

$$q^* = \arg \max_{p \in N(q) \setminus q'} \mu(p),$$

repeating the procedure on q^* and continuing unless q^* is a point of $\mathbb{X}_1$ or a boundary point of $\mathbb{M}$. This procedure may result in degenerate "loops" where a sparse network path is greedily extended back onto itself, so we detect and remove such loops. Figure 5(c) shows the final sparse network. Let $\mathcal{S} = (\mathbb{X}, \Gamma)$ denote the final network.

4.2 Sampling based approach

The previous approach requires the computation of a large number of minimum-cost paths, resulting in a high preprocessing cost. We now describe an approach that involves computing a much smaller number of paths between points that are not too far away from each other.

We fix two parameters m and k. First, we choose a set $\mathbb{P} \subseteq \mathbb{Q}$ of m points; see below how $\mathbb{P}$ is chosen. Next, for each point $p \in \mathbb{P}$, we compute its k nearest neighbors in $\mathbb{P} \setminus \{p\}$, using the distance along Σ. Let $\mathsf{N}(p)$ denote this set of k points. We then compute the path $\Omega(p, q)$ between p and each point $q \in \mathsf{N}(p)$ using an A* algorithm. We add each point on $\Omega(p, q)$ to the sparse network $\mathcal{S}$ (if it is not present already), and add each edge (u, v) of the path to $\mathcal{S}$. Finally, we find the grid point x_p (resp. y_p) on a horizontal (resp. vertical) edge of $\mathbb{M}$ that is closest to p. We add the paths $\Omega(p, x_p)$ and $\Omega(p, y_p)$ to $\mathcal{S}$ as above. Figure 6 illustrates this procedure.

We now discuss different strategies for choosing the set $\mathbb{P}$ of sample points.

Random. Select m random points of $\mathbb{Q}$; each point is chosen independently with uniform probability.

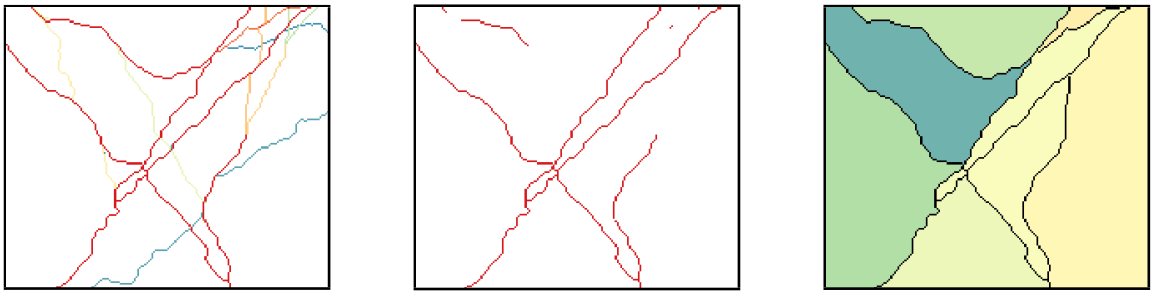


Figure 5. Constructing a sparse network from a heatmap: (a) The original heatmap; (b) the thresholded heatmap; (c) The sparse network.

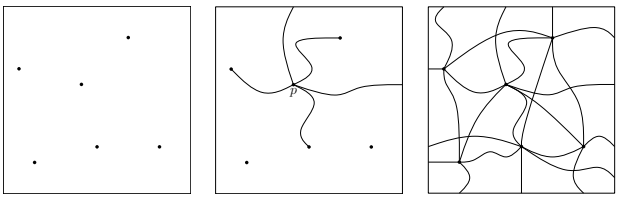


Figure 6. An illustration of sampling-based construction for $k = 3$. (a) The set of points $\mathbb{P}$. (b) Connecting point p to its k nearest neighbors and two border points via most occluded paths. (c) The full sparse network.

Even-low. Greedily select m grid points that have low coverage-map values and are spaced apart. Given parameters d_h and d_v, the minimum horizontal and vertical gap between points, first select the point p_1 with the lowest coverage-map value. Then, for each subsequent choice i, select the point p_i which is the point more than d_h distance away from each point in $\{p_1, \ldots, p_{i-1}\}$ horizontally and more than d_v distance away from each point in $\{p_1, \ldots, p_{i-1}\}$ vertically with the lowest coverage-map value. If no such point exists, set $d_h = d_h/2$ and $d_v = d_v/2$, and repeat. Then set $\mathbb{P} = \{p_1, \ldots, p_m\}$.

Random-low. For a parameter $c \leq \frac{1}{m}$, randomly select m grid points from among the $1/c$ points on Σ with the lowest coverage-map values.

Remarks. *(i)*: Note that sampling based strategies are popular in e.g. robotics [12, 13] but they are commonly applied slightly different. Instead of constructing the network $\mathcal{S}$ like we do in our strategies, they construct an abstract graph $G = (V, E)$ with vertices corresponding to $\mathbb{P}$ and edges (p, p') corresponding to points p, p' with weight set to the cost of the path $\Omega(p, p')$. We chose our approach for the same reason as described at the end of Section 3.
(ii): We have proposed strategies that choose a set of random points from a subset of points on Σ. If we had domain knowledge, we could have used a deterministic strategy that chooses a few "landmarks", an approach commonly used for path queries on road networks.

4.3 Topological approach

The objective of the topological approach is to construct a sparse network that follows the boundaries of "valleys" of the coverage map, i.e., the corridors of low-visibility cells which are prime candidates for network edges.

For a vertex u in the grid graph $\mathcal{G} = (\mathbb{Q}, \mathcal{E})$, let $\mathsf{N}^+(u)$ denote the set of "up" neighbors v of u with $\omega(v) > \omega(u)$. We define the *ascending neighbor* $\uparrow(u)$ to be the up neighbor of u with the highest value, i.e.,

$$\uparrow(u) = \arg \max_{v \in \mathsf{N}^+(u)} \omega(v).$$

If $\mathsf{N}^+(u) = \emptyset$, $\uparrow(u)$ is undefined. We define the *ascent graph* $\mathcal{A} = (\mathbb{Q}, \mathcal{E}')$ to be a subgraph of $\mathcal{G}$, where $\mathcal{E}' = \{(u, v) \mid v = \uparrow(u)\}$. Since all edges of $\mathcal{A}$ are directed from low to high values in ω and each vertex has at most one outgoing edge, the ascent graph $\mathcal{A}$ is a forest, with each maximum of the coverage map being the root of a tree in this forest. This forest partitions the vertices of $\mathbb{Q}$ and this induces a subdivision of $\mathbb{M}$. Intuitively, the boundary of the faces of this subdivision forms the desired network. Formally, for a vertex $v \in \mathbb{Q}$, let T_v denote the tree of $\mathcal{A}$ that contains v. We call a vertex $v \in \mathbb{Q}$ a *boundary vertex* if there is a neighbor u of v in $\mathcal{G}$ such that $T_u \neq T_v$. We define the subgraph of $\mathcal{G}$ induced by the boundary vertices as an initial network $\mathcal{S}_1$. Next, similar to the learning approach, we perform a post-processing step and produce the final network $\mathcal{S}$. The post-processing step removes dangling paths in $\mathcal{S}_1$ and also the faces with empty interiors. The latter appear along the boundary vertices of two "adjacent" trees, say, T_1, T_2, of $\mathcal{A}$. We call a boundary vertex of T_2 *redundant* if it is adjacent only to the boundary vertices of T_1 or T_2. We remove redundant boundary vertices of T_2 and the edges incident on them. We omit the details, which are straightforward, from here.

Because of the high resolution of $\mathbb{Q}$, the coverage map ω is likely to contain many local maxima, implying that the ascent graph contains many small regions corresponding to insignificant features. Thus, we preprocess the coverage map using the notion of *topological persistence* [9, 8], as extended to volume [5], before computing the ascent graph. We remove low-persistence features (features with low volume) and then compute the ascent graph on the resulting simplified coverage map.

Remarks. This strategy can also be applied to the terrain Σ directly, with the intuition that valleys and low areas in the terrain are often hidden from view; see the next section for further discussion on this.

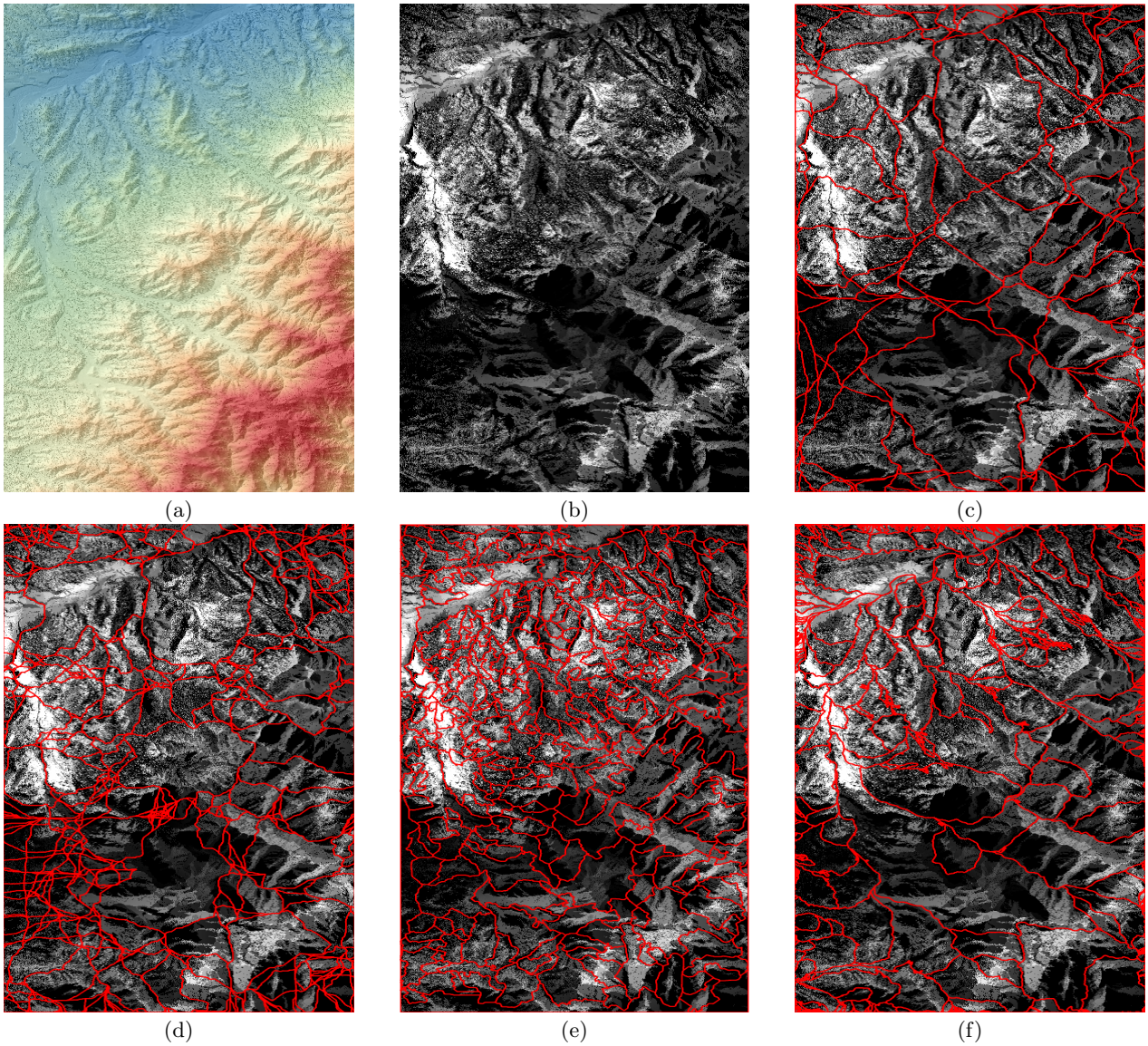


Figure 7. (a) The Afghanistan terrain. (b) The coverage map. (c) The learning network. (d) The sampling network. (e) The topology network. (f) The topology-terrain network.

5. EXPERIMENTS

We present detailed experimental results to evaluate the effectiveness of the sparse-network approach. First, we compare the quality of the paths computed on different sparse networks with that of the optimal path, i.e., the optimal path computed directly from the coverage map and terrain. Next, we compare the preprocessing time of different strategies and their path qualities. We also study how the path quality changes as we increase the size of a network. Finally, we compare the query time using a sparse network to that of computing an optimal path on the coverage map.

Experimental setup. The experiments were performed on a machine with an Intel Core i7-3770 CPU running at 3.40 GHz, 24GB of internal memory, and an NVIDIA GeForce GTX 660 graphics card interfaced with OpenGL. The code was written in C++.

We used two real terrain models in our experiments, each provided by the US Army Topographic Engineering Center (TEC). One dataset covers a roughly $4 \times 8\text{km}^2$ region in Afghanistan at a resolution of 2 meters. The terrain model consists of about 7.1 million grid points with mostly mountainous topology. It is a so-called *Digital Surface Model* (DSM) and as such contains non-terrain features relevant for visibility, such as trees and a few buildings. The second data set is a larger, higher-resolution 1 meter grid covering a $9 \times 9\text{km}^2$ region in Ft. Leonard Wood in Missouri. The model consists of 81 million grid points with mostly rolling hills and a prominent riverbed. It is a *Digital Terrain Model* (DTM) containing only bare-earth elevation data.

We have implemented the three network-construction strategies presented in Section 4, namely, the learning, sampling, and topology strategies. For the sampling strategy, we chose sample points randomly. In addition, we also implemented

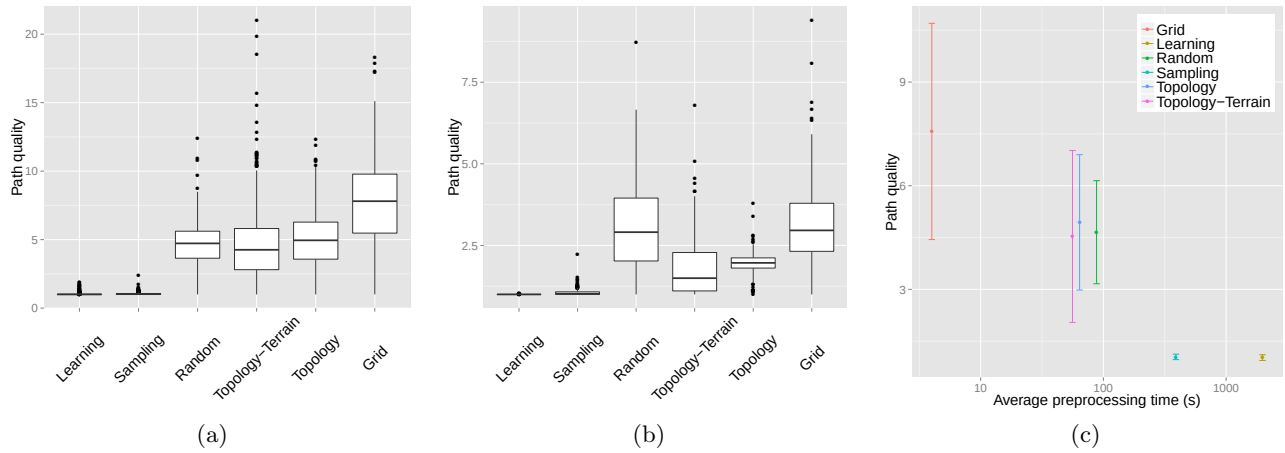


Figure 8. Boxplot showing the distributions of path-cost ratios (ρ_S) for each of the network construction strategies on (a) Afghanistan and (b) Ft. Leonard Wood terrains. (c) Path-cost ratio plotted against preprocessing time for each of the network-construction strategies on the Afghanistan terrain.

three strategies that do not use the coverage map, as a baseline for comparison: (i) *random*: similar to the sampling strategy, choose a set of m random grid points and connect each of them to its k-nearest neighbors. Instead of connecting two points by the most occluded path, this approach connects the points with Euclidean shortest paths. (ii) *topology-terrain*: applies the topological approach described in Section 4.3 to the original terrain, instead of the coverage map. (iii) *grid*: creates a network by partitioning the terrain into a 10×10 grid.

For each of the two terrain models, the coverage map, used for computing and evaluating paths as well as for constructing the sparse network, was formed by selecting the 25 best observers for the terrain according to the persistence-based observer selection algorithm described in [14] and using the GPU based algorithm of the same paper. Linear attenuation of visibility was used on each terrain, with a 2km maximum visibility distance on the Afghanistan terrain and a 6.8km maximum visibility distance on the Ft. Leonard Wood terrain. We set $C = 6$ and $\alpha = 59/60$ in (1), the functional form of the coverage map.

To measure query time and path quality, we randomly choose 200 source and destination point pairs. For each pair, we compute the paths using A*, which yields the optimal path cost, as well as using a sparse network computed by each of the strategies.

Network construction. Figure 7 shows the Afghanistan terrain and its coverage map, as well as the networks resulting from the learning, sampling, topology, and topology-terrain strategies, overlaid on top of the coverage map. The comparison shows that the networks constructed by the learning and the sampling strategies align with the regions of low visibility in the coverage map. The topology network similarly has edges in low-visibility areas, but it is a denser network than the other two, particularly in the high-variance regions with lots of trees in the upper-left portion of the terrain. The topology-terrain network has several edges that run through regions of high visibility, a byproduct of the fact that it is not constructed using the coverage map.

Path quality. For two points a and b and a sparse-network strategy S, the *path-cost ratio* is

$$\rho_S(a,b) = \frac{\bar{\mathtt{w}}_S(a,b)}{\mathtt{w}(a,b)}.$$

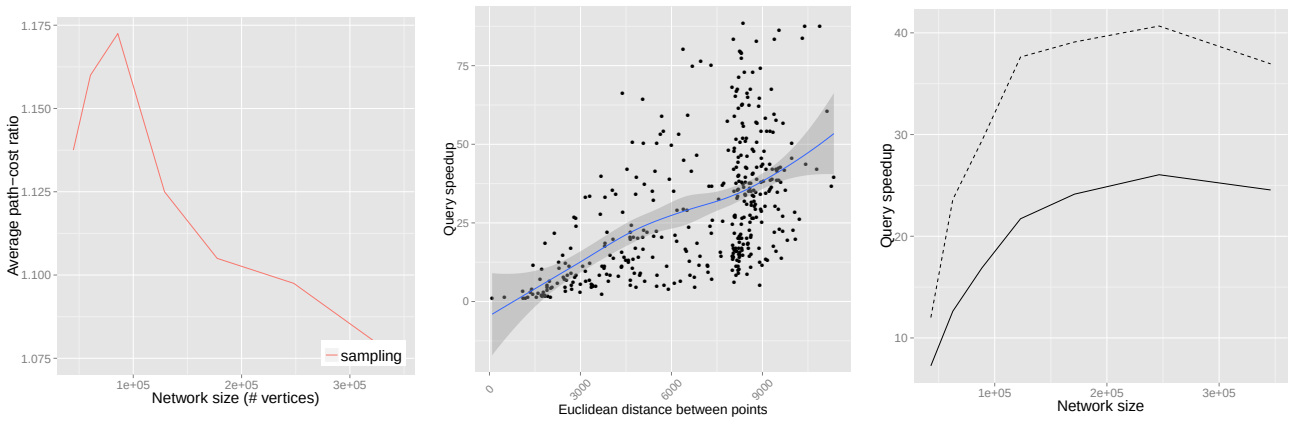
We use ρ_S to measure the quality of the paths computed by S. Ideally, we want ρ_S to be as close to 1 as possible.

Figure 8 shows that the learning strategy gives the highest-quality paths: on both terrains, the median path-cost ratio is barely above 1, and the distribution is concentrated very close to the median, indicating that the vast majority of paths are very close to optimal. The sampling strategy gives paths that are almost as good, with medians close to 1 and distributions concentrated below 1.5 on all terrains. The topology strategy does not perform as well, with higher medians and more spread.

The strategies that do not use the coverage map also fail to return high-quality paths, with random, topology-terrain, and grid strategies having distributions centered around much higher medians and with far more spread. The fact that these three strategies give much higher-cost paths indicates that their ability to accurately capture low-visibility subpaths suffers without access to the coverage map itself.

Preprocessing time. Figure 8(c) plots ρ_S against preprocessing time for each of the strategies on the Afghanistan terrain, with error bars indicating the standard deviation of the path-cost ratio distribution. The learning strategy requires much more time to construct the network than the other strategies, due to the high number of paths it computes. The preprocessing cost for the sampling strategy is much lower, and even less for the topology strategies. Finally, the grid strategy has an extremely low preprocessing time, due to the fact that it involves very little computation. A similar experiment on the Ft. Wood terrain showed trends similar to those in Figure 8(c).

In general, there is an inverse relationship between preprocessing time and path cost: strategies that have lower preprocessing times have higher path costs.



(a) (b) (c)

Figure 9. (a) Network size vs average path-cost ratios on the Afghanistan terrain. (b) Euclidean distance between points vs query speedup for the sampling strategy on the Ft. Wood terrain. (c) Network size (number of vertices) vs average query speedup for the sampling strategy on the Afghanistan terrain, for all paths in the experiment (solid line) and only the paths separating endpoints at least 8km away from each other (dashed line).

Path quality vs network size. We now investigate the relationship between network size and path quality.

We control the size of each network indirectly by controlling certain parameters. For the learning strategy, we vary the threshold t in the heat map (see (3)); for the sampling strategy, we vary m, the number of sample points;[2] for the topology strategy, we vary the persistence threshold.

All networks exhibit a similar trade-off between path quality and their size, so we focus on the sampling strategy. Figure 9(a) shows the (average) path quality as a function of network size for the sampling strategy, averaged over four runs. Generally, the path quality improves as the network size increases unless the network size is so small that the source and destination points lie in the same face for most queries, in which case the algorithm computes an optimal path (the initial portion of the curve in Figure 9(a)). We also note that after a certain size, the marginal increase in the path quality is very little.

Query time. The query time of all strategies was roughly the same, so we focus on comparing the query time of the sampling strategy with that of the A* algorithm, which runs on the coverage map and the terrain. We define the *query speedup* to be the query time of the A* algorithm divided by that of the sampling strategy. The large the value, the higher the speedup, and the better it is.

If two query points lie on the same face of a sparse network, the query-processing algorithm reduces to A*, so the sparse-network approach does not have any advantage over A* and the query speedup is roughly 1.

The local planning time over the faces containing the source and destination points remains essentially constant, so the sparse-network query time increases very slowly as distance increases between the query points, while A* query times increase more dramatically. Figure 9(b) illustrates this trend, plotting query-time ratio against Euclidean distance between the endpoints of the path for the sampling strategy on the Ft. Leonard Wood terrain. For one pair of points on the graph only 87m apart, the sampling and A* query times are approximately equal, while for a pair of points over 11km apart, the sampling strategy provides an 87-fold speedup over A*.

Sparse-network path computation gains an advantage over A* as the sparse-network size increases as well, as shown in Figure 9(c) for sampling networks on the Afghanistan terrain. For a very small network with only a few large faces, the majority of queries are between points on the same face and reduce to A*. The learning network on the Ft. Leonard Wood terrain is such a network. As the network size increases, the faces in the network become smaller, speeding up the local path-planning portions of the computation. As the networks become extremely dense, however, query times increase, decreasing the sparse network's advantage over A*; this trend can be observed in the rightmost portion of the graph. Figure 9(c) also reinforces the previous observation that distance between points affects query times, as for all network sizes, the average query-time ratio over only paths between far-away points is consistently higher than the average query-time over all paths.

6. DISCUSSION

In this paper we proposed a sparse-network based approach for computing the highly occluded paths on a terrain, developed multiple strategies for constructing such a network, and presented detailed experimental results to demonstrate the effectiveness of this approach. Our results suggest that a significant speed-up can be achieved by using this approach instead of computing an optimal path on the coverage map.

We conclude by discussing a few open problems. An immediate open question is whether the construction of a sparse network can be expedited by parallelizing the algorithm, say,

[2]In out experiments, as we increased m, we kept the old sample points, so the set $\mathbb{P}$ in one network is the subset of another.

using a GPU, especially since we use a GPU to construct the coverage map. A more interesting question, however, is to explore better strategies for constructing a sparse network. Among the strategies presented in this paper, the sampling strategy seems to provide the best trade-off between path quality and preprocessing time, but there are several interesting directions to pursue.

We believe that a topology based algorithm can outperform the sampling strategy, but one needs a better criterion than the ascent graph to construct the network. There is some recent work in machine learning on tracing "lower-dimensional" critical curves (e.g. ridges) [18], which might lead to a more effective topology based strategy for constructing a sparse network. Furthermore, it will be useful to develop a hierarchical or an incremental algorithm for constructing the network, which starts with a very sparse network and then locally refines the network as needed. This approach is likely to provide a better trade-off between path quality and preprocessing time. Another interesting direction is to develop an effective strategy that does not require computing the coverage map and works directly with the original terrain. Our results show that a simple minded approach will not work. An approach that effectively exploits the correlation between visibility and the topology of terrain is needed. Finally, developing a sparse-network-construction algorithm that comes with a provable guarantee on its worst-case performance, both in terms of the path quality and the query time, is another potential avenue for future work.

Acknowledgments.. The authors thank James Rogers and Arnold Boedihardjo from the US Army ERDC-TEC for providing us the two datasets that we used in this paper, and the anonymous reviewers for their helpful comments.

References

[1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. F. Werneck, A hub-based labeling algorithm for shortest paths in road networks, *Proc. 10th Annu. Sympos. Experimental Algos.*, 2011, pp. 230–241.

[2] P. K. Agarwal, R. Sharathkumar, and H. Yu, Approximate euclidean shortest paths amid convex obstacles, *Proc. 20th Annu. ACM-SIAM Sympos. Discrete Algos.*, 2009, pp. 283–292.

[3] L. Aleksandrov, H. Djidjev, A. Maheshwari, and J.-R. Sack, An approximation algorithm for computing shortest paths in weighted 3-d domains, *Discr. Comput. Geom*, 50 (2013), 124–184.

[4] L. Aleksandrov, A. Maheshwari, and J.-R. Sack, Approximation algorithms for geometric shortest path problems, *Proc. 42nd Annu. ACM Sympos. Theory of Comput.*, 2000, pp. 286–295.

[5] L. Arge and M. Revsbæk, I/o-efficient contour tree simplification, in: *Proc. 20th Annu. Sympos. Algos. Comput.*, 2009, pp. 1155–1165.

[6] J. F. Canny, A new algebraic method for robot motion planning and real geometry, *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, 1987, pp. 39–48.

[7] J. Chen and Y. Han, Shortest paths on a polyhedron, *Int. J. Comput. Geom. Appl.*, 6 (1996), 127–144.

[8] A. Danner, T. Mølhave, K. Yi, P. K. Agarwal, L. Arge, and H. Mitasova, TerraStream: from elevation data to watershed hierarchies, *Proc. 15th Intl. Sympos. Geog. Info. Sys.*, 2007, pp. 1–8.

[9] H. Edelsbrunner, D. Letscher, and A. Zomorodian, Topological persistence and simplification, *Proc. 41st Annu. IEEE Sympos. Found. Comput. Sci.*, 2000, pp. 454–463.

[10] C. R. Ferreira, S. V. G. Magalhães, M. V. A. Andrade, W. R. Franklin, and A. M. Pompermayer, More efficient terrain viewshed computation on massive datasets using external memory, *Proc. 20th Intl. Sympos. Geog. Info. Sys.*, 2012, pp. 494–497.

[11] W. R. Franklin, M. Inanc, Z. Xie, D. M. Tracy, B. Cutler, and M. V. A. Andrade, Smugglers and border guards: the geostar project at RPI, *Proc. 15th Intl. Sympos. Geog. Info. Sys.*, 2007, pp. 30:1–30:8.

[12] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *IEEE T. Robotics and Automation*, 12 (1996), 566–580.

[13] S. M. LaValle, *Planning algorithms*, Cambridge University Press, 2006.

[14] N. Lebeck, T. Mølhave, and P. K. Agarwal, Computing highly occluded paths on a terrain, *Proc. 21st Intl. Sympos. Geog. Info. Sys.*, 2013, pp. 14–23.

[15] M. Lu, J. Zhang, P. Lv, and Z. Fan, Max/min path visual coverage problems in raster terrain, *CAD and Comp. Graphics*, 2007, pp. 497–500.

[16] M. S. Marzouqi and R. A. Jarvis, New visibility-based path-planning approach for covert robotic navigation, *Robotica*, 24 (2006), 759–773.

[17] C. Ó'Dúnlaing, M. Sharir, and C.-K. Yap, Retraction: A new approach to motion-planning (extended abstract), *Proc. 15th Annu. ACM Sympos. Theory of Comput.*, 1983, pp. 207–220.

[18] U. Ozertem and D. Erdogmus, Locally defined principal curves and surfaces, *J. Mach. Learn. Res.* 12 (2011), 1249–1286.

[19] D. Shaharabani, O. Salzman, P. K. Agarwal, and D. Halperin, Sparsification of motion-planning roadmaps by edge contraction, *Proc. IEEE Intl. Conf. Robitics Auto.*, 2013, pp. 4098–4105.

[20] S. Shekhar and S. Chawla, *Spatial Databases: A Tour*, Prentice Hall, 2003.

[21] K. R. Varadarajan and P. K. Agarwal, Approximating shortest paths on a nonconvex polyhedron, *SIAM J. Comput.*, 30 (2000), 1321–1340.

[22] A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou, Shortest path and distance queries on road networks: towards bridging theory and practice, *Proc. ACM Intl. Conf. Manage. Data*, 2013, pp. 857–868.