Online Geometric Reconstruction*

Bernard Chazelle
chazelle@cs.princeton.edu

C. Seshadhri
csesha@cs.princeton.edu

Department of Computer Science
Princeton University
Princeton, NJ 08540

ABSTRACT

We investigate a new class of geometric problems based on the idea of online error correction. Suppose one is given access to a large geometric dataset though a query mechanism; for example, the dataset could be a terrain and a query might ask for the coordinates of a particular vertex or for the edges incident to it. Suppose, in addition, that the dataset satisfies some known structural property $\mathcal{P}$ (eg, monotonicity or convexity) but that, because of errors and noise, the queries occasionally provide answers that violate $\mathcal{P}$. Can one design a filter that modifies the query's answers so that (i) the output satisfies $\mathcal{P}$; (ii) the amount of data modification is minimized? We provide upper and lower bounds on the complexity of online reconstruction for convexity in 2D and 3D.

1. INTRODUCTION

Classical error correction assumes the prior availability of exact data. In this way, we can encode the data in redundant form so as to allow its recovery after transmission through a noisy channel. But what if the data $\mathcal{D}$ comes to us already corrupted? If the clean data is out of reach, then obviously the very notion of corruption requires an assumption about prior state. Indeed, what justification would one have to call a signal noisy if we have no idea what clean data might look like. The prior could be a distribution or, more generally, a property $\mathcal{P}$. For example, the data could be the facial representation of a cell decomposition and $\mathcal{P}$ could specify that it be a Voronoi diagram. Or $\mathcal{D}$ could be a terrain that $\mathcal{P}$ constrains to be monotone or convex. Or $\mathcal{D}$ might consist of an architectural design, with $\mathcal{P}$ enforcing certain angular constraints (eg, right angles between adjacent walls). Or $\mathcal{D}$ could be a cloud of points in high dimensions that $\mathcal{P}$ forces on or near a low degree algebraic manifold.

In all cases, the same question arises: Is it possible to *filter* the data online so that (i) it satisfies $\mathcal{P}$ and (ii) the amount of modification is minimized? Such a filter can be used as the front-end to geometric codes whose correctness depends on certain properties (such as convexity, monotonicity, axis-parallelism). The offline version of the problem is a form of generalized regression: among all datasets satisfying $\mathcal{P}$, find the one nearest $\mathcal{D}$. The problem assumes a metric between datasets, which could be geometric, combinatorial, or a mixture of both. In this paper, it will be purely combinatorial. The true novelty of our setting is its *online* nature. To see why being online changes everything, think of an architectural scenario where objects are required to be isothetic. Suppose that the first query reveals the position of a door and its adjacent wall. If the door is ajar, the online filter has two equally valid options in order to enforce isotheticity: either it can move the door or it can move the wall. The latter option would have dire consequences, however, likely to lead to the modification of the entire structure. This simplistic example points to the main challenge of online filtering: *early decisions are crucial.*

We use the filtering model defined in [8]. Access to the dataset is provided by means of an oracle f. The client specifies a query x and the oracle returns $f(x)$. The query could specify the index x of a wall, with $f(x)$ providing the coordinates of its vertices; or the index of an adjacent wall. One should not think of f as a single function but rather as the set of methods (in the OOP sense) available to access the dataset and its underlying data structures. The *filter* works like this: given a query x, instead of simply returning $f(x)$, the filter provides the client with the cleaned-up answer $g(x)$. Figure 1 illustrates its inner workings. Upon receiving the query x, the filter spawns auxiliary queries $a, b, c, \ldots$ and computes $g(x)$ on the basis of of $f(a), f(b), f(c), \ldots$ The filter may go through several rounds before producing $g(x)$ and adaptively produce queries based on the previous answers. The set of all possible values $g(x)$ specifies a dataset $\mathcal{D}^f$ that satisfies property $\mathcal{P}$. The filter's decisions are *irreversible*: once a feature of $\mathcal{D}^f$ has been established, it can never be undone.

The quality of a filter is determined by how close $\mathcal{D}^f$ is to the object that satisfies $\mathcal{P}$ and differs from $\mathcal{D}$ as little as possible. This requires a metric Δ between datasets (defined on a case-by-case basis). For the purpose of this paper, we re-

*This work was supported in part by NSF grants CCR-998817, 0306283, and ARO Grant DAAH04-96-1-0181.

SCG'06, June 5–7, 2006, Sedona, Arizona, USA.

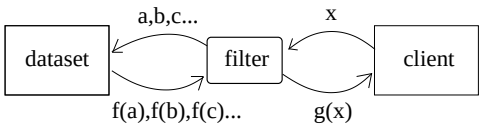

Figure 1: ***The online filter responds to a query*** x.

quire that all filters obey the *constant-distortion* condition:

$$\Delta(\mathcal{D}^f, \mathcal{D}) = O(\min\{\Delta(\mathcal{D}, \mathcal{D}') \mid \text{ all } \mathcal{D}' \text{ satisfying } \mathcal{P}\}).$$

Obviously, the true quality of a filter will also depend on how big the big-Oh constant factor is. In other cases, for the sake of speed, one might allow nonconstant distortions (but we won't in this paper). Note that, most often, setting the constant to 1 makes the reconstruction NP-hard. There are two aspects to a filter's complexity which we might sometimes want to distinguish: the *lookup complexity* is the number of dataset lookups (ie, $a, b, c, \ldots$); the *processing complexity*, which cannot be smaller, is the running time of the filter per query. The distinction is useful in instances where dataset access is slow or expensive (as is often the case in biology, medicine, geology, etc). In this paper, however, all query times refer to the processing complexity. Our emphasis also is on worst case query times. Obviously, it is also interesting to keep the amortized complexity low—and we'll do that—but keep in mind that to focus *only* on amortized complexity would render the online problem meaningless (since we could always solve the offline version and charge the first query for it). Worst case complexity is, indeed, an *essential* feature of online reconstruction.

1.0.0.1 Our Results and Previous Work.

In the following, n denotes the size of the input and the distance between two polygons (terrains) is defined as the minimum number of edges (faces) whose coordinates need to be modified to transform one into the other. All the algorithms below are randomized and succeed with high probability.[1] We give three filters for reconstructing convexity in two and three dimensions:

1. An optimal $\widetilde{O}(\sqrt{n}\,)$ time filter for reconstructing the convexity of a simple polygon presented as a doubly-linked list. We assume that each vertex is labeled with its rank in the list. We also give an $\widetilde{O}(n^{1/3})$ time ε-tester[2] and prove an $\Omega(n^{1/3})$ lower bound.

2. An optimal $\widetilde{O}(\log^3 n)$ time filter for reconstructing the convexity of a simple polygon presented as a balanced binary tree.

3. A filter for reconstructing the convexity of a bounded aspect ratio[3] terrain presented in triangulated DCEL format with a worst case query time of $O(n^{13/14+\delta} + \varepsilon_{\mathcal{D}}^{-O(1)})$ and an amortized time of $O(n^{\delta})$. Here, δ is an arbitrarily small positive constant and $\varepsilon_{\mathcal{D}} n$ is the terrain's distance to convexity, ie, the minimum number of faces whose coordinates need to be modified in order to make the terrain convex. We also prove a lower bound that explains why the complexity must depend on $\varepsilon_{\mathcal{D}}$.

This paper presents new results that are intriguing enough, we hope, to inspire further work. To get a taste of why the results are surprising, consider the fact that exact 3D reconstruction offline is not known to be in polynomial time, so it is remarkable that allowing a constant factor approximation in the distance should allow us to answer online queries in *sublinear* time. Many tools are required to achieve this result, including the planar separator theorem, balanced trapezoidal decompositions, sampling in range spaces of unbounded VC dimension, and approximation algorithms for vertex cover. Another puzzling fact is why the 2D filter's complexity does *not* depend on the distance $\varepsilon_{\mathcal{D}}$ but its 3D counterpart does. By showing that online 3D reconstruction requires $\Omega(\varepsilon_{\mathcal{D}}^{-1})$ time, we prove that this difference is intrinsic and represents yet another complexity gap between 2D and 3D. We leave the online reconstruction of general terrains and, most interesting, of arbitrary polytopes as an open problem.

We are not aware of any line of work that our results can be compared with directly—except for [8], where online property-preserving data reconstruction was introduced and polylogarithmic time filters were provided for reconstructing monotone functions. Our work uses the same model but very different techniques. Of course, offline geometric reconstruction has been studied before, but usually the metric is geometric (like the Hausdorff distance) and not combinatorial. Examples include finding the best approximation of surfaces satisfying certain criteria [2, 4, 6, 10]. On the other hand, a notion of combinatorial distance is certainly present when studying the computational aspects of the Erdős-Szekeres theorem [13] or other Ramsey-like results. Geometric properties have been well studied within the purview of property testing [15, 17], program checking [25], and sublinear algorithms [12]. Efficient testers have been given for convexity [15, 17, 19], clustering [9, 21, 22, 27], and Euclidean MST [14, 16], but there too the relevance to our work is only tangential. Because of space limitations, we limit our discussion to convexity filters for terrains.

2. A CONVEXITY FILTER FOR POLYGONS

Rather than dealing with the general case, which has a number of technical complications that are not germane to the 3D case, we restrict ourselves to the case of terrains. The reason for this is that it provides a gentle warmup for the (vastly more difficult) 3D case. It will also allow us to highlight the fundamental difference between 2D and 3D filtering: the complexity's dependency on the distance to convexity for 3D terrains.

The input is a 2D terrain $\mathcal{D}$, ie, a polygonal curve $p_1, \ldots, p_n$ running monotonically from left to right. We assume a

[1] Throughout this paper, "with high probability" is shorthand for: with probability at least $1 - n^{-c}$, for an arbitrarily large constant $c > 0$, where n is the size of the input.

[2] Unless indicated otherwise, all our algorithms are randomized. An ε-tester for a property $\mathcal{P}$ determines whether an input of size n satisfies $\mathcal{P}$ or requires at least εn modifications to do so. We use the notation $\widetilde{O}(f) = O(f)(\log f)^{O(1)}$ and our claims of optimality are to be understood up to polylogarithmic factors.

[3] A terrain is said to have *bounded aspect ratio* if the xy-projections of the faces have bounded sides and angles.

doubly-linked list representation, which allows us to access a random edge and walk from it in either direction. (Note that we *cannot* perform a binary search among the edges, since they are not stored sorted in a table. An alternative model where we can—result 2 above—will also be investigated in the full paper.) Let $\varepsilon_{\mathcal{D}} n$ denote the minimum number of edges that need to be modified to make $\mathcal{D}$ convex. By convex, we mean the upper hull of the set of vertices of $\mathcal{D}$. Two edges $e = p_i p_{i+1}, f = p_j p_{j+1}$ are naturally ordered from left to right if $i < j$, which we write as $e \prec f$. We define $[e, f] = \{p_{i+1}p_{i+2}, \ldots, p_{j-1}p_j\}$. A pair (e, f) is a *violating* if the convex hull of $\{e, f\}$ does not have both edges on the boundary.

What makes the 2D case remarkable is that a certain easily testable property allows us to classify any given edge in one of two categories (good or bad) in a way that leads to a filtering mechanism with a constant approximation factor. (A similar classification was used for filtering monotone functions in [8].)

DEFINITION 2.1. *Given any $0 < \delta < 1/2$, an edge e is called* δ-bad *if there exists an edge f such that either (i) $e \prec f$ and the number of $g \in [e, f]$ that violate e is at least $(1/2-\delta)|[e, f]|$ or (ii) $f \prec e$ and the number of $g \in [f, e]$ that violate e is at least $(1/2 - \delta)|[f, e]|$. The edge f is referred to as a* witness *to e's badness. An edge that is not δ-bad is called* δ-good.

The following transitivity relation is immediate: if $e \prec f \prec g$ and (e, g) is a violating pair, then so is at least one of (e, f) or (f, g). The next lemma is crucial for proving the correctness of the filter.

LEMMA 2.2. *(i) The 0-good edges have no violating pairs; (ii) at least $\varepsilon_{\mathcal{D}} n$ edges are 0-bad; and (iii) no more than $(3 + 8\delta/(1 - 2\delta))\varepsilon_{\mathcal{D}} n$ edges are δ-bad.*

PROOF. (from [7]) Note that by transitivity, for any $e \prec f$ such that (e, f) is a violating pair, either e or f (or both) is 0-bad. Therefore, if we were to remove all the 0-bad edges, the remaining edges would be in convex position; hence (i) and (ii).

We start by assigning to each δ-bad e a witness f_e to its badness (if many witnesses exist, we just choose any one). If $f_e \succ e$, then e is called *right-bad*; else it is *left-bad*. (Obviously, the classification depends on the choice of witnesses.)

Let C be a set of $\varepsilon_{\mathcal{D}} n$ edges where $\mathcal{D}$ can be modified to make it convex. To bound the number of right-bad edges, we charge C with a credit scheme. (Then we apply a similar procedure to bound the number of left-bad edges.) Initially, each element of C is assigned one unit of credit. For each right-bad $e \notin C$ (in reverse order from right to left), *spread* one credit among all the g such that $e \preceq g \preceq f_e$ and (e, g) is a violation (note that g must be in C). We use the word "spread" because we do not simply drop one unit of credit into one account. Rather, viewing the accounts as buckets and credit as water, we pour one unit of water one infinitesimal drop at a time, always pouring the next drop into the least filled bucket.

We now show that no edge in C ever receives an excess of $2 + 4\delta/(1 - 2\delta)$ units of credit. Suppose by contradiction that this were the case. Let e be the right-bad that causes some edge g's (g belongs to C) account to reach over $2 + 4\delta/(1 - 2\delta)$. By construction e is not in C; therefore, the excess occurs while right-bad e is charging an edge g such that $e \prec g \preceq f_e$ and (e, g) is a violation. Note that, because $e \notin C$, any g satisfying these two conditions belongs to C (let us denote the number of such edges by l) and thus gets charged. With the uniform charging scheme, this ensures that all of these l elements of C have the same amount of credit by the time they reach the excess value, which gives a total greater than $l(2 + 4\delta/(1 - 2\delta))$. By definition of right-badness, $l \geq (1/2 - \delta)|[e, f_e]|$. But none of these accounts could be charged before step f_e; therefore,

$$(1/2 - \delta)|[e, f_e]|(2 + 4\delta/(1 - 2\delta)) < |[e, f_e]|,$$

which is a contradiction.

Of the total of at most $2 + 4\delta/(1 - 2\delta)\varepsilon_{\mathcal{D}} n$ units of credit, $\varepsilon_{\mathcal{D}} n$ units of credit came from initially assigning the edges of C one unit of credit each. Therefore, there are at most $1 + 4\delta/(1 - 2\delta)\varepsilon_{\mathcal{D}} n$ right-bad edges. By applying a similar argument for left-bad edges (this time charging from left to right), we prove (iii). □

By repeating the procedure below a logarithmic number of times we can, with high probability, find whether an edge e is δ-bad or 2δ-good in $\widetilde{O}(\sqrt{n})$ time.

GOODNESS-TESTER(e, δ)

$\mathcal{S} \leftarrow$ portion of $\mathcal{D}$ that stretches $\sqrt{n}$ edges on both sides of e
if, for some $f \in \mathcal{S}$, a fraction $(\frac{1}{2} - \delta)$ of edges $(e, g) \in [e, f]$ (or $(g, e) \in [f, e]$) are violating
  then output δ-bad
$\mathcal{R} \leftarrow$ random set of $O(\delta^{-2}\sqrt{n})$ edges in $\mathcal{D}$
sort $\mathcal{R}$ from left to right
if, for some $f \in \mathcal{R}$, a fraction $\frac{1}{2}(1 - 3\delta)$ of edges $(e, g) \in [e, f] \cap R$ (or $(g, e) \in [f, e] \cap R$) are violating
  then output δ-bad
output 2δ-good

The convexity filter for 2D terrains works as follows: If the query edge e has been committed, then we return it unchanged. Similarly, if GOODNESS-TESTER(e, δ) outputs "2δ-good", we output the edge e and commit to its current value. Otherwise, we find the closest committed edges to the left (e^l) and to the right (e^r). Let $\mathcal{S}$ be the portion of $\mathcal{D}$ that stretches $\sqrt{n}$ edges on both sides of e. By using a modification of the algorithm in [8], we can find a "close" 2δ-good edge in $\mathcal{S}$ that precedes (resp. follows) e, which we denote by $e^l_{\mathcal{S}}$ (resp. $e^r_{\mathcal{S}}$). The meaning of close is that the edge, say $e^l_{\mathcal{S}}$, lies in the smallest interval $[f, e] \subseteq \mathcal{S}$ that contains a fraction of 2δ-good edges in $\mathcal{S}$ that lies in the interval $[\delta, 2\delta]$. Next, pick a random sample $\mathcal{R}$ of $O(\delta^{-2}\sqrt{n}\log n)$ edges in $\mathcal{D}$, and sort $\mathcal{R}$ from left to right. Repeating the closeness procedure above with respect to $\mathcal{R}$ (instead of $\mathcal{S}$) gives us the two edges $e^l_{\mathcal{R}}$ and $e^r_{\mathcal{R}}$. Among the three edges, $e^l, e^l_{\mathcal{S}}, e^l_{\mathcal{R}}$, (resp. $e^r, e^r_{\mathcal{S}}, e^r_{\mathcal{R}}$), let l^* (resp. r^*) denote the closest one to e. Join the right endpoint of l^* to the left endpoint of r^* with a segment σ that is used to reconstruct the interval $[l^*, r^*]$. Lift to σ all the edges in $[l^*, r^*]$ and commit to all of them. Output the new value of e (which has been lifted

to σ). We omit the proof of correctness, which is technical, and simply summarize the main features of the filter.

THEOREM 2.3. *Any n-edge 2D terrain has a convexity filter with a worst case query time of $\widetilde{O}(\sqrt{n})$ time and an amortized time of $O(\log n)$.*

3. A CONVEXITY FILTER FOR 3D TERRAINS

The dataset is an n-face triangulated terrain $\mathcal{D}$ represented in standard DCEL fashion. We assume that the xy-projection of any face of $\mathcal{D}$ is a triangle with both edge lengths and angles bounded above and below by constants (bounded aspect ratio condition). The reconstructed terrain $\mathcal{D}^c$ is convex in the sense of being the boundary of the upper hull of its vertex set. There are various equally reasonable definitions of the parameter $\varepsilon_{\mathcal{D}}$. For simplicity, we define $\varepsilon_{\mathcal{D}} n$ as the minimum number of faces of $\mathcal{D}$ that need to be removed in order to make the terrain convex. Note that this definition does not require us to "patch the holes." Choosing to do so, however, would only increase the distance by a constant factor, which, for the purpose of our filter, is immaterial. The edge table allows us to sample random edges. From this, it is elementary to implement a uniform sampler for triangular faces as well. (To sample vertices would be more difficult but, fortunately, we do not need that feature.)

The filter processes the terrain during the first query in sublinear time and then uses the resulting data structure to answer subsequent queries. The (sublinear) cost of the processing is charged entirely to the first query. The idea is to break up the terrain into connected *patches* of suitable size by removing a small set $\mathcal{F}$ of separating faces. The *fence* $\mathcal{F}$ decomposes the terrain into connected patches of suitable sizes. A critical feature of $\mathcal{F}$ is to be of sublinear size. To achieve this, we use a weakened version of the classical planar separator theorem. The weakening is required to make the computation sublinear. The final processing step is to convexify $\mathcal{F}$. This is a delicate operation which cannot be performed in isolation with the rest of the terrain: this is a perfect illustration of why early decisions are crucial in online filtering.

We define a suitable range space (of unbounded VC dimension!) whose sampling gives us enough global information about the whole terrain to guide the reconstruction of $\mathcal{F}$. The convexified $\mathcal{F}$ fences off the patches in such a way that it is possible from then on to answer any subsequent query by treating its associated patch in isolation from the rest of the terrain. But, before we can get to online reconstruction, we need to define two key procedures: one is an offline algorithm for convexifying the terrain within twice the minimum distance; the other estimates the distance $\varepsilon_{\mathcal{D}}$ in sublinear time.

Any filter must explore both global and local properties. The difficulty lies in gathering enough information in sublinear time. Any approach must involve a combination of sampling and local search. The filter essentially works as follows: first, it estimates $\varepsilon_{\mathcal{D}}$ using the sublinear time procedure. If the distance is very small, then the offline algorithm is used for convexification. Otherwise, it constructs a fence $\mathcal{F}$ and convexifies it. It is critical that this convexification be done by taking the global structure into account—this is achieved by choosing a large enough sample of faces of $\mathcal{D}$ (which then is used to define the range space mentioned in the previous paragraph) and convexifing $\mathcal{F}$ so that it is in convex position with most of the sample. This creates a "skeleton" which captures the global properties of $\mathcal{D}$ and also splits $\mathcal{D}$ into a set of small connected patches. Next, each patch is reconstructed independently, ensuring that it stays in convex position with the convexified $\mathcal{F}$. Since a patch is a small connected portion of $\mathcal{D}$, it can be visited exhaustively by local search (thereby, the filter gains information about the local properties of $\mathcal{D}$). In the following subsections, we discuss the various components that constitute the filter. Finally, we put the pieces together and describe the filter itself.

OFFLINE-RECONSTRUCTION

```
Initialization:  T ← ∅
  for each face f of D:
    if f is in convex position with T
      then T ← T ∪ {f}
      else  find face g ∈ T not in
            convex position with f
            T ← T \ {g}
    output D^c ← T
```

3.0.0.2 *Offline Reconstruction.*

We describe a 2-approximation offline convexification algorithm, ie, one that, given $\mathcal{D}$ as input, returns a terrain $\mathcal{D}^c$ that is convex and is at distance at most $2\varepsilon_{\mathcal{D}} n$ from it. The convexification proceeds incrementally. Beginning with the empty terrain T, we consider each face of $\mathcal{D}$ one by one and add it to T if it is in convex position[4] with every face currently in $\mathcal{D}^c$. To do this in quasilinear time, we maintain T in a dynamic data structure which supports insertion, deletion, and queries in $O(n^\delta)$ time for any fixed $\delta > 0$.

Denote by T_v (resp. T_p) the set of vertices (resp. face-supporting planes) in T. A terrain face f is in convex position with T if and only if (i) its three vertices lie below each plane in T_v; and (ii) the points of T_p all lie below the plane supporting f. By duality, both tests can be reduced to *dynamic halfspace emptiness* in 3D: maintain a set of points under insertion and deletion, and for any query plane find whether whether all points lie on one side, and if they do not, report one point on each side. Agarwal and Matoušek [5] have given a halfspace range reporting algorithm which, with minor modifications, allows us to do that in $O(\nu^\delta)$ amortized time for each query/insert/delete, where ν is the number of points at the time of the operation.

THEOREM 3.1. *Offline convex reconstruction of an n-face terrain can be performed with an approximation factor of 2 in $O(n^{1+\delta})$ time, for any fixed $\delta > 0$.*

3.0.0.3 *Estimating the Distance to Convexity.*

Consider the violation graph G whose nodes are the faces of $\mathcal{D}$ and whose edges join any two faces not in convex position. The minimum vertex cover is of size $\varepsilon_{\mathcal{D}} n$, and any

[4] Two triangles are said to be in convex position if both of them are faces of their convex hull.

maximal matching M in G is of size $|M| \leq \varepsilon_{\mathcal{D}} n \leq 2|M|$. Fix any constants $0 < \alpha < \beta < 1$ such that $\alpha \geq \frac{1}{2}(3\beta - 1)$, and let S be a random sample formed by picking each vertex of G independently with probability $p = n^{\alpha-\beta+\delta_0}$ for arbitrarily small $\delta_0 > 0$. Note that the offline reconstruction algorithm in effect builds a maximal matching M_S for the subgraph $G_{|S}$ of G induced by S. The sample S can be easily specified in $O(|S|)$ time, so that computing M_S takes $O(pn)^{1+\delta}$ time, which is $O(n^{1+\alpha-\beta+\delta_1})$ for arbitrarily small $\delta_1 > 0$. As we show below, knowing M_S allows us to distinguish between the two cases: $\varepsilon_{\mathcal{D}} \geq n^{-\alpha}$ and $\varepsilon_{\mathcal{D}} \leq n^{-\beta}$.

1. $\varepsilon_{\mathcal{D}} \geq n^{-\alpha}$: If ξ is the number of edges of M in G_S, then $\mathbf{E}\,\xi = p^2|M|$ and, by Chernoff's bound [1], $\text{Prob}\,[\,\xi < \frac{1}{2}p^2|M|\,] < e^{-\Omega(p^2|M|)} = e^{-\Omega(p^2 n^{1-\alpha})} < e^{-n^{\delta_2}}$, for arbitrarily small $\delta_2 > 0$. So, with high probability, $G_{|S}$ contains a perfect matching of size at least $\frac{1}{2}\,p^2|M| \geq \frac{1}{4}\,p^2 n^{1-\alpha}$. Its minimum vertex cover is at least that size; therefore, $|M_S| \geq \frac{1}{8}\,p^2 n^{1-\alpha}$.

2. $\varepsilon_{\mathcal{D}} \leq n^{-\beta}$: If χ denotes the number of vertices of M in S, then $\mathbf{E}\,\chi = 2p|M|$ and, by Chernoff's bound, $\text{Prob}\,[\,\chi \geq 2p|M| + y\,] < e^{-y^2/|M|}$, for any $y > 0$. Setting $y = \frac{1}{8}\,p^2 n^{1-\alpha} - 2p|M| > \frac{1}{9}\,p^2 n^{1-\alpha}$, we find that $\text{Prob}\,[\,\chi \geq \frac{1}{8}\,p^2 n^{1-\alpha}\,] < e^{-\Omega(p^4 n^{1+\beta-2\alpha})} < e^{-n^{\delta_3}}$, for arbitrarily small $\delta_3 > 0$. Since the vertices of M provide a vertex cover for G, it follows that, with high probability, $|M_S| < \frac{1}{8}\,p^2 n^{1-\alpha}$.

THEOREM 3.2. *Given any small $\delta > 0$ and any constants $0 < \alpha < \beta < 1$ such that $\alpha \geq \frac{1}{2}(3\beta - 1)$, we can compute a 0/1 bit $b(\mathcal{D})$ in $O(n^{1+\alpha-\beta+\delta})$ time, such that $b(\mathcal{D}) = 0$ if $\varepsilon_{\mathcal{D}} \geq n^{-\alpha}$, $b(\mathcal{D}) = 1$ if $\varepsilon_{\mathcal{D}} \leq n^{-\beta}$, and $b(\mathcal{D})$ takes on any value otherwise.*

3.0.0.4 Fencing Off the Terrain.

Here we describe an algorithm that finds a sublinear set of faces (the fence) of $\mathcal{D}$ whose removal breaks $\mathcal{D}$ into connected components (patches) also of sublinear size. As we discussed earlier, this procedure will be run in the first query. Let G be the planar triangulation formed by the projection of $\mathcal{D}$ onto the xy-plane. Pick a random sample of $r = n^a$ edges in G, for fixed $0 < a < 1$, and build its (say, x-oriented) trapezoidal map $\mathcal{M}_r$. As is well known, with high probability, each trapezoid intersects $O((n/r)\log n)$ triangles. Consider the dual $\mathcal{H}_r$ graph of $\mathcal{M}_r$, where each node is a trapezoid and two nodes are joined if the corresponding (closed) trapezoids intersect. The graph is planar and so, by iterated application of the planar separator theorem [23], for any fixed $0 < b < 1$, we can find, in $O(r \log r)$ time, a set V of $O(r^{1-b/2})$ nodes whose removal leaves $\mathcal{H}_r$ with no connected component of size exceeding r^b. The fence $\mathcal{F}$ is defined as the polyhedral surface formed by the terrain's faces whose projections intersect the trapezoids associated with the nodes of V. With high probability, the fence consists of $O(r^{1-b/2}(n/r)\log n)) = O(n^{1-ab/2}\log n)$ triangles and its removal from the terrain leaves connected patches, each one consisting of $O(n^{1+ab-a}\log n)$ triangles. Note that, because it involves triangles (and not trapezoids), the removal may create much greater fragmentation than is caused within $\mathcal{H}_r$ by the removal of V. Finding the fence takes time

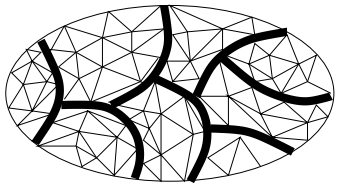

Figure 2: ***The thick black line, the fence, is a itself collection of $o(n)$ triangles.***

$O(n^a \log n + n^{1-ab/2}\log n)$ time, Renaming ab by b, we have proven:

LEMMA 3.3. *For any $0 < b < a < 1$, in $O((n^a + n^{1-b/2})\log n)$ time, it is possible to find a fence $\mathcal{F}$ consisting of $O(n^{1-b/2}\log n)$ triangles, whose removal from the terrain leaves connected patches consisting of $O(n^{1+b-a}\log n)$ triangles each.*

3.0.0.5 Reconstructing the Fence.

It might be tempting to convexify the fence by applying the offline algorithm to it, but this could doom future reconstruction (the "crucial early decisions" phenomenon). Instead, we must allow the global shape of the terrain to influence the reconstruction. To do so, we choose a random sample Σ of the faces of $\mathcal{D}$—the size of Σ shall be set later. Let Σ^c be the offline convexification of the terrain Σ provided by Theorem 3.1, and let Σ^f the intersection of the halfspaces bounded above by the planes supporting the faces of Σ^c (the dual convex hull). The reconstructed fence $\mathcal{F}^c$ is obtained by lifting the triangles of $\mathcal{F}$ vertically and "wrapping" them over the surface of Σ^f. Note that such a lifting could replace one fence face by many faces, but based on the bounded aspect ratio condition, we have a bound (proven later in this section) for the total size of $\mathcal{F}^c$. (The bound holds only when Σ is sufficiently large, but our choice of Σ will ensure that.)

LEMMA 3.4. *The size of $\mathcal{F}^c$ is $O(n^{1-b/4}\log n)$.*

We now explain why $\mathcal{F}^c$ captures the global structure of $\mathcal{D}$. We define a range space $(X, \mathcal{R})$, which, although of unbounded VC dimension, has enough sampling power to guide the convexification of the fence. Regarding both $\mathcal{D}$ and $\mathcal{F}$ as sets of triangles, we define the ground set $X = \mathcal{D} \setminus \mathcal{F}$. Given two sets S, T of triangles, let $\kappa(S, T)$ be the set of triangles in T that are not in convex position with at least one triangle of S. Considering all possible sets Γ of $|\mathcal{F}^c|$ triangles, we define $\mathcal{R} = \{\,\kappa(\Gamma, X) : |\Gamma| = |\mathcal{F}^c|\,\}$. A triangle is specified by 9 reals, and so, viewing Γ as a point in $p_\Gamma \in \mathbb{R}^{9|\mathcal{F}^c|}$, it is easy to see why the convex position status of each of its triangles with respect to X is completely specified by the location of p_Γ in a certain $9|\mathcal{F}^c|$-dimensional arrangement of $O(n|\mathcal{F}^c|)$ hyperplanes. It follows that the primal shatter function φ grows as $\varphi(m) = O(m|\mathcal{F}^c|)^{9|\mathcal{F}^c|}$, and that, with high probability, if Σ is chosen to be a random sample of X of size $O(r^2|\mathcal{F}^c|\log|\mathcal{F}^c|) = O(r^2 n^{1-b/4}\log^2 n)$, it is a $(1/r)$-approximation for $(X, \mathcal{R})$, for any $r > 0$.

By Theorem 3.1, the number of triangles in Σ that were modified during the convexification is at most $2\varepsilon_\Sigma|\Sigma|$. Using an argument similar to the proof of Theorem 3.2, we can

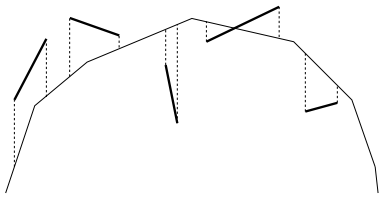

Figure 3: ***The fence is reconstructed by lifting its faces to the upper boundary of*** Σ^c.

show that, with high probability, this number is $O(\varepsilon_{\mathcal{D}}|\Sigma|)$. This implies that $|\kappa(\mathcal{F}^c, \Sigma)| = O(\varepsilon_{\mathcal{D}}|\Sigma|)$. Since

$$\left| \frac{|\kappa(\mathcal{F}^c, \Sigma)|}{|\Sigma|} - \frac{|\kappa(\mathcal{F}^c, X)|}{|X|} \right| \leq \frac{1}{r},$$

we could easily bound the "damage" caused by the convexification of the fence as follows:

LEMMA 3.5. $\kappa(\mathcal{F}^c, X) \leq nr^{-1} + O(\varepsilon_{\mathcal{D}} n)$.

We now prove Lemma 3.4. For ease of notation, we shall assume that $\mathcal{F}$ has $O(n^u \log n)$ faces and Σ has size $\widetilde{O}(n^v)$. We will be concerned only with xy-projections of $\mathcal{D}$ and Σ^f. In the following proof, *triangle* refers to the xy-projection of a face of $\mathcal{D}$, while *facet* refers to the xy-projection of a face of Σ^f. *Edges* refer to the edges of triangles. Note that all facets are convex, disjoint from each other (except for their boundaries) and contain a triangle. By the bounded aspect ratio assumption, the radius lengths of the incircle and circumcircle of any triangle are bounded from above and below (by some constant). Let v' be some value less than v.

DEFINITION 3.6. *For $\alpha < n^{-(1-v')}$, an* α-thin facet *is a facet with two edges e_1, e_2 such that the minimum distance between e_1 and e_2 is less than $n^{-(1-v')}$ and the angle between e_1 and e_2 is less than α. The edges e_1 and e_2 are called* sharp *edges. The* min-thinness *of a facet f is the minimum α such that f is α-thin.*

The sharp edges of an α-thin facet form a *wedge* that contains the facet (Figure 4).

CLAIM 3.7. *With high probability, there are at most $O(\alpha n)$ α-thin facets.*

PROOF. Consider some α-thin facet f which contains triangle t. The distance between t and the sharp edges is at least $\Omega(\alpha^{-1})$ (since t must be inside the wedge created by these sharp edges, and has at least constant in-radius). There exists a rectangle of $\Omega(\alpha^{-1})$ width and $\Omega(1)$ height that is present completely inside f but does not intersect t (Figure 4). Therefore, we can also show that there exist at least $\Omega(\alpha^{-1})$ edges of $\mathcal{D}$ that have at least a constant fraction of their length inside f (let these edges be *bad* for f). Note that the offline convexification must have removed these bad edges. With high probability (by taking a Chernoff bound for each rectangle and then a union bound over all such rectangles, there being polynomially many relevant ones), at least $(c\alpha n^{1-v})^{-1}$ of these edges are chosen in Σ (for some sufficiently large constant c). Let the total number of α-thim facets be N. Since all facets are disjoint, an edge can be bad for only a constant number of α-thin facets. Therefore, the total number of bad edges is $> N(c\alpha n^{1-v})^{-1}$. But this quantity has to be $O(n^v)$, since that is the number of edges removed by convexification. This implies that the number of α-thin facets is $O(\alpha n)$. □

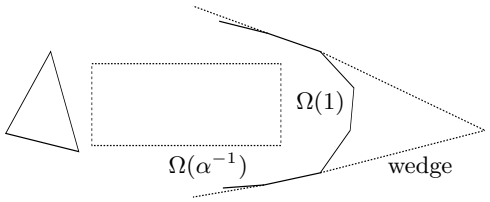

Figure 4: ***Wedge***

CLAIM 3.8. *For any $v' < v$, the total complexity of the lifted fence is $O((n^{1+u-v'} + n^{v'}) \log n)$.*

PROOF. The complexity of lifting a fence face is simply the complexity of the corresponding triangle when laid over the xy-projection of Σ^f. A triangle is said to *generate* all the faces created by overlaying. The total complexity of all triangles generating $O(n^{1-v'})$ faces is $O(n^{1+u-v'} \log n)$ (since there are at most $O(n^u \log n)$ fence triangles). Consider some triangle t generating $k > n^{1-v'}$ faces. Each of these faces is created by the intersection of t with a facet.

We will first show that many of these facets are k^{-1}-thin. Since triangles do not intersect, no facet can be completely contained inside t. At least $\Omega(k)$ facets intersect some edge e of t. We take a circle C of constant (but large enough) radius such that C contains e and the minimum distance between e and C is $\Omega(1)$. The intuition for the following proof is quite simple—since C is a constant sized circle and many facets intersect it, many facets have to be thin (Figure 5. Since the area of a facet is $\Omega(1)$, there can be at most a constant number of facets contained completely inside C. Therefore, at least $\Omega(k)$ facets intersect both e and C. A facet can intersect C in two ways - inward and outward (Figure 6). Consider a facet f (containing triangle t') that intersects C only in the inward direction. Either more than half of t' is contained in C or the arc length of some intersection between C and f is $\Omega(1)$. Only a constant number of facets can have only inward intersection. Therefore, $\Omega(k)$ facets intersect C outwards, and (by a Markov argument) $\Omega(k)$ of these intersections have arc length $< k^{-1}$. Consider any such facet f' - the two edges intersecting C has a minimum distance of less than k^{-1}. Since f' intersects e, the angle between the edges must be $O(k^{-1})$, and f' is $O(k^{-1})$-thin. This shows that for any triangle t generating k faces, $\Omega(k)$ of these faces are $O(k^{-1})$-thin facets. We will say that t is a *witness* for these $O(k^{-1})$-thin facets, since the intersection of C with these facets shows their $O(k^{-1})$-thinness. For any facet f of min-thinness α, note that at most $(\alpha n^{1-v'})^{-1}$ triangles are witnesses for any thinness (since the minimum distance between sharp edges is $< n^{-(1-v')}$ and the angle is α.)

We sum up the constributions (in terms of complexity) of all triangles generating more that $n^{1-v'}$ faces. Let this sum

be S. By the arguments given above, βS comes from thin facets that are witnessed (for some fixed constant $\beta < 1$). Some facets are counted more than once in βS because many triangles can witness one facet. Let us consider all witnessed facets with min-thinness in the range $[\alpha/2, \alpha]$. There are at most $O(\alpha n)$ such facets and each is counted in S at most $2(\alpha n^{1-v'})^{-1}$ times. Therefore, the total contribution of this in S is $O(n^{v'})$. We apply this argument for the values $\alpha = n^{-(1-v')}, 2^{-1}n^{-(1-v')}, 2^{-2}n^{-(1-v')}, \cdots, (cn)^{-1}$ (for some sufficiently large constant c) and get that $S = O(n^{v'} \log n)$. □

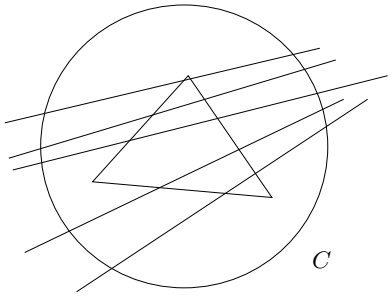

Figure 5: ***Facets intersecting with*** C

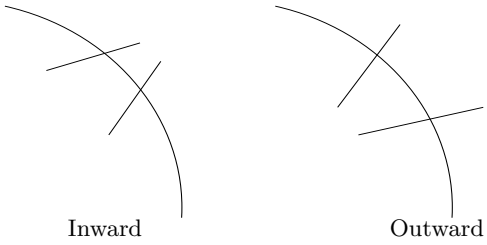

Figure 6: ***Inward and Outward***

Noting that $u = 1 - b/2$ and $v > 1 - b/4$, we can set $v' = (1+u)/2$. By the previous claim, the complexity of the lifted fence can be made $O(n^{1-b/4} \log n)$, proving Lemma 3.4.

3.0.0.6 Online Reconstruction.

We now have all the necessary tools required to make the filter. As we mentioned earlier, the first query is the occasion of some preliminary processing whose cost is entirely charged to the query itself. By Theorem 3.2, we estimate the distance to convexity in $O(n^{1+\alpha-\beta+\delta})$ time. If $b(\mathcal{D}) = 1$, then we convexify the terrain in $O(n^{1+\delta})$ time by appealing to Theorem 3.1. Since $\varepsilon_{\mathcal{D}} < n^{-\alpha}$, the running time is $O(\varepsilon_{\mathcal{D}}^{-\alpha^{-1}+\delta})$.

Assume now that $b(\mathcal{D}) = 0$, which implies that $\varepsilon_{\mathcal{D}} > n^{-\beta}$. By Lemma 3.5, setting $r = n^{\beta}$ shows that $\kappa(\mathcal{F}^c, X) \leq O(\varepsilon_{\mathcal{D}} n)$. A crucial aspect of the reconstructed fence is that the convexification of any patch can be done in isolation, as long as we include the fence triangles bounding the patch in question. This follows from this transitivity lemma, whose proof we omit from this abstract. (We use the subscript xy to denote the projection onto the xy-plane.)

3D-Terrain-Filter

```
if first query
then if b(D) = 1
       then convexify D using
         Offline-Reconstruction
       else build fence F and convexify
         into F^c and go to (1)
else
 (1) identify patch containing query face f
     if needed, convexify extended patch
     with Offine-Reconstruction
```

Lemma 3.9. *Let f, g be two faces of a (possibly discontinuous) terrain and let F, G be two sets of faces such that: (i) removing the region F_{xy} disconnects f_{xy} from g_{xy}; same is true of G_{xy}. If f (resp. g) is in convex position with F (resp. G), then f and g are in convex position with each other.*

Given a query f, unless $b(\mathcal{D}) = 1$, the filter finds the patch corresponding to f. If it is the first access to the patch, then it proceeds to reconstruct the entire patch together with all its bordering fence triangles (what we call the *extended patch*). Otherwise, it simply outputs the corresponding face in the reconstructed patch. By Lemma 3.3, computing the fence takes $O((n^a + n^{1-b/2}) \log n)$ time. By our setting of $r = n^{\beta}$, to find Σ requires $\widetilde{O}(r^2 n^{1-b/4}) = \widetilde{O}(n^{1+2\beta-b/4})$ time, and convexification can be done in time $O(n^{1+2\beta-b/4+\delta})$. Reconstructing the fence adds nothing to the asymptotic complexity. Any query that involves convexifying the corresponding patch takes $O(n^{1+b-a+\delta})$. Putting everything together, we see that in the worst case the time for answering any query is

$$O(n^{1+\alpha-\beta+\delta} + \varepsilon_{\mathcal{D}}^{-\alpha^{-1}+\delta} + (n^a + n^{1-b/2}) \log n + n^{1+2\beta-b/4+\delta} + n^{1+b-a+\delta}).$$

The constraints $0 < b < a < 1$, $0 < \alpha < \beta < 1$, and $\alpha \geq \frac{1}{2}(3\beta - 1)$ are all satisfied if we set α to be an arbitrarily small positive constant and $a = 13/14$, $b = 6/7$, and $\beta = 1/14$. It is immediate that the amortized query time is $O(n^{\delta})$.

Theorem 3.10. *Any n-face 3D bounded aspect ratio terrain $\mathcal{D}$ has a convexity filter with a worst case query time of $O(n^{13/14+\delta} + \varepsilon_{\mathcal{D}}^{-O(1)})$ and an amortized time of $O(n^{\delta})$, for an arbitrarily small $\delta > 0$.*

3.0.0.7 Lower Bound.

We show that any 3D convexity filter has a worst case query time of $\Omega(\varepsilon_{\mathcal{D}}^{-1})$ time, thus revealing a fundamental complexity gap between the two and three-dimensional cases. Recall that the 2D filter made essential use of a certain transitivity feature of convexity violation: if e, f, g are edges in clockwise order and (e, g) is not in convex position, then at least one of (e, f) or (f, g) is not either. In designing our filter, we used a 3D variant of this by letting the fence play the role of f. But, unlike in 2D, the fence cannot be a constant

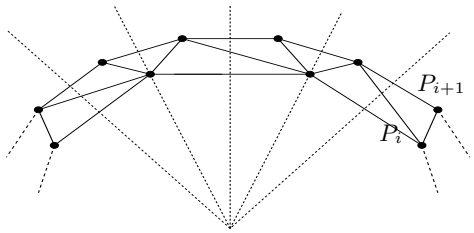

Figure 7: ***The concentric rings of the xy-projection.***

size object. Why this implies a lower bound is explained below.

We appeal to Yao's *minimax lemma* to deal with the fact that our algorithms are randomized. We briefly sketch the proof, beginning with the case $\varepsilon_{\mathcal{D}} = \Theta(\log n)/n$. Assume that the filter changes at most $c\varepsilon_{\mathcal{D}} n$ faces, for some fixed $c > 1$. We define a distribution of inputs and show that, for any deterministic algorithm that performs reconstruction, some query takes $\Omega(n/\log n)$ expected time over that distribution.

We start with a fixed $\mathcal{D}$ and build the distribution around it. Fix some parameter $m > 0$. The xy-projection of $\mathcal{D}$ consists of $\Theta(\log m)$ concentric regular polygons $P_0, P_1, \cdots, P_{k-1}$ centered at the origin (Figure 7): (i) the innermost polygon P_0 has a constant number of vertices; (ii) P_i has $2^{i-1}|P_1|$ vertices and every other edge is parallel to an edge of P_{i-1}; (iii) P_{k-1} has $m/(c_1 \log m)$ vertices, for fixed $c_1 > 0$. The radii of the P_i's are chosen so that the boundaries are fairly close to each other but disjoint. Next, we lift these polygons vertically so that their edges are all horizontal tangents to the paraboloid $\mathcal{C} : Z = -(X^2 + Y^2)$ at their midpoints (Figure 8). Each band between consecutive polygons is triangulated appropriately and the construction is lifted to $\mathcal{C}$ to form a convex terrain with (lifted) P_0 as its highest face. Finally, we add an extra polygon P_k that is a slightly scaled-up version of P_{k-1}. The band between P_{k-1} and P_k consists of $m/(c_1 \log m)$ trapezoids, each one of which is now divided up into a stack of $c_1 \log m$ parallel subtrapezoids. After lifting, each subtrapezoid finds itself tangent to $\mathcal{C}$. Triangulating all faces produces $n = \Theta(m)$ faces.

The terrain $\mathcal{D}$ is convex: we introduce convexity violations by choosing one *stack* $\mathcal{S}$ of subtrapezoids, and tilting them ever so slightly so that: (i) each subtrapezoid violates one common triangle of P_1; (ii) the stack $\mathcal{S}$ violates $O(1)$ triangles per (P_i, P_{i+1}) band. (This requires local retriangulation and the addition of $O(\log m)$ faces.) Now the crux of the argument rests on setting c_1 large enough so that $2c_1 \log m > c\varepsilon_{\mathcal{D}} n$. (We omit the discussion of why that inequality can be achieved.) In this way, a query to the common violating triangle of P_1 cannot return the triangle unchanged. Indeed, if it did, then the entire stack $\mathcal{S}$ of $2c_1 \log m$ triangles would later have to be modified, which would prove the filter faulty. It can also be shown that to modify the violating triangle of P_1 appropriately requires knowing where the stack $\mathcal{S}$ is placed around the (P_{k-1}, P_k) band, which takes $\Omega(|P_k|)$ expected time. The extension to larger values of $\varepsilon_{\mathcal{D}}$ uses a similar construction, and we omit it from this abstract.

THEOREM 3.11. *Any convexity filter for a terrain $\mathcal{D}$ of n faces has a worst case query time of $\Omega(\varepsilon_{\mathcal{D}}^{-1})$ for any n such that $(\log n)/n \leq \varepsilon_{\mathcal{D}} = o(n)$.*

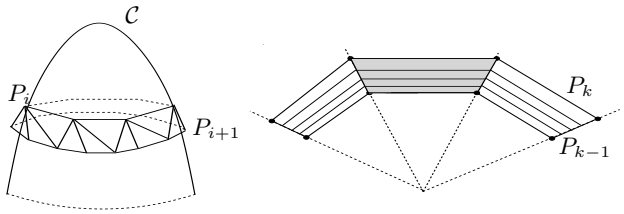

Figure 8: ***A hard terrain to reconstruct.***

4. REFERENCES

[1] Alon, N., Spencer, J. *The probabilistic method*, John Wiley, 2nd edition, 2000.

[2] Agarwal, P.K., Desikan P.K. *An efficient algorithm for terrain simplification*, Proc. SODA (1997), 139-147.

[3] Agarwal, P.K., Hagerup, T., Ray, R., Sharir, M., Smid, M., Welzl, E. *Translating a planar object to maximize point containment*, Proc. ESA (2002), 42-53.

[4] Agarwal, P.K., Har-Peled, S., Mustafa, N., Wang, Y. *Near-linear time approximation algorithms for curve simplification*, to appear in Algorithmica.

[5] Agarwal, P.K., Matoušek, J. *Dynamic half-space searching and its applications*, Algorithmica 14 (1995), 325–345.

[6] Agarwal, P.K., Suri, S. *Surface approximation and geometric partitions*, SIAM J. Computing 27 (1998), 1016–1035.

[7] Ailon, N., Chazelle, B., Comandur, S., Liu, D. *Estimating the distance to a monotone function*, Proc. RANDOM (2004), 229–236.

[8] Ailon, N., Chazelle, B., Comandur, S., Liu, D. *Property preserving data reconstruction*, Proc. ISAAC (2004), 16–27.

[9] Alon, N., Dar, S., Parnas, M., Ron, D. *Testing of clustering*, Proc. FOCS (2000), 240–250.

[10] Amenta, N., Choi, S., Dey, T.K., Leekha, N. *A simple algorithm for homeomorphic surface reconstruction* Proc. SOCG (2000), 213-222.

[11] Chazelle, B. *The Discrepancy Method : Randomness and Complexity*, Cambridge University Press, 2000; paperback version, 2001.

[12] Chazelle, B., Liu, D., Magen, A. *Sublinear geometric algorithms*, Proc. STOC (2003), 531–540.

[13] Chvatal, V., Klincsek, G. *Finding largest convex subsets*, Congressus Numerantium: 29 (1980), 453–460.

[14] Czumaj, A., Ergun, F., Fortnow, L., Magen, A., Newman, I., Rubinfeld, R., Sohler, C., *Sublinear-time approximation of Euclidean minimum spanning tree*, Proc. SODA (2003), 813–822.

[15] Czumaj, A., Sohler, C. *Property testing with geometric queries*, Proc. ESA (2001), 266–277.

[16] Czumaj, A., Sohler, C. *Estimating the weight of metric minimum spanning trees in sublinear-time*, Proc. STOC (2004), 175–183.

[17] Czumaj, A., Sohler, C., Ziegler M. *Property testing in computational geometry*, Proc. ESA (2000), 155-166.

[18] de Berg, M., van Kreveld, M., Overmars, O., Schwarzkopf, O. *Computational Geometry - Algorithms and Applications*, Springer-Verlag, 1997.

[19] Ergun, F., Kannan, S., Kumar, S. Ravi, Rubinfeld, R., Viswanathan, M. *Spot-checkers*, Proc. STOC (1998), 259–268.

[20] Frederickson, G.N. *Fast algorithms for shortest paths in planar graphs, with applications*, SIAM J. Comput. 16 (1987), 1004–1022.

[21] Indyk, P. *A sublinear-time approximation scheme for clustering in metric spaces*, Proc. FOCS (1999), 154–159.

[22] Indyk, P. *Sublinear-time algorithms for metric space problems*, Proc. STOC 1999), 428–434.

[23] Lipton, R.J., Tarjan, R.E. *A separator theorem for planar graphs*, SIAM Journal on Applied Mathematics 36 (1979),

177–189.

[24] Lipton, R.J., Tarjan, R.E. *Applications of a planar separator theorem*, SIAM J. Comput. 9 (1980), 615–627.

[25] Mehlhorn, K., Näher, S., Seel, M., Seidel, R., Schilz, T., Schirra, S., Uhrig, C. *Checking geometric programs or verification of geometric structures*, Proc. SOCG (1996), 159–165.

[26] Miller, G.L. *Finding small simple cycle separators for 2-connected planar graphs*, J. Computer and System Sciences 32 (1986), 265–279.

[27] Mishra, N., Oblinger, D., Pitt, L. *Sublinear time approximate clustering*, Proc. SODA (2001), 439–447.