Available online at www.sciencedirect.com
ScienceDirect
Computational Statistics & Data Analysis 51 (2007) 2461–2486
COMPUTATIONAL STATISTICS & DATA ANALYSIS
www.elsevier.com/locate/csda

A practical approximation algorithm for the LMS line estimator☆

David M. Mount[a,*,1], Nathan S. Netanyahu[b,c], Kathleen Romanik[d,2], Ruth Silverman[c], Angela Y. Wu[e]

[a] *Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD, USA*
[b] *Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel*
[c] *Center for Automation Research, University of Maryland, College Park, MD, USA*
[d] *White Oak Technologies, Inc., Silver Spring, MD, USA*
[e] *Department of Computer Science, American University, Washington, DC, USA*

Received 1 March 2005; accepted 25 August 2006
Available online 2 October 2006

Abstract

The problem of fitting a straight line to a finite collection of points in the plane is an important problem in statistical estimation. Robust estimators are widely used because of their lack of sensitivity to outlying data points. The least median-of-squares (LMS) regression line estimator is among the best known robust estimators. Given a set of n points in the plane, it is defined to be the line that minimizes the median squared residual or, more generally, the line that minimizes the residual of any given quantile q, where $0 < q \leqslant 1$. This problem is equivalent to finding the strip defined by two parallel lines of minimum vertical separation that encloses at least half of the points.

The best known exact algorithm for this problem runs in $O(n^2)$ time. We consider two types of approximations, a *residual approximation*, which approximates the vertical height of the strip to within a given error bound $\varepsilon_r \geqslant 0$, and a *quantile approximation*, which approximates the fraction of points that lie within the strip to within a given error bound $\varepsilon_q \geqslant 0$. We present two randomized approximation algorithms for the LMS line estimator. The first is a conceptually simple quantile approximation algorithm, which given fixed q and $\varepsilon_q > 0$ runs in $O(n \log n)$ time. The second is a practical algorithm, which can solve both types of approximation problems or be used as an exact algorithm. We prove that when used as a quantile approximation, this algorithm's expected running time is $O(n \log^2 n)$. We present empirical evidence that the latter algorithm is quite efficient for a wide variety of input distributions, even when used as an exact algorithm.

Keywords: Least median-of-squares regression; Robust estimation; Line fitting; Approximation algorithms; Randomized algorithms; Line arrangements

☆ A preliminary version of this paper appeared in Proceedings of the Eighth Annual ACM–SIAM Symposium on Discrete Algorithms, 1997, pp. 473–482.

* Corresponding author. Tel.: +1 301 4052704; fax: +1 301 4056707.
E-mail addresses: mount@cs.umd.edu (D.M. Mount), nathan@cs.biu.ac.il, nathan@cfar.umd.edu (N.S. Netanyahu), kromanik@comcast.net (K. Romanik), ruthsilverman@verizon.net (R. Silverman), awu@american.edu (A.Y. Wu).

[1] This material is based upon work supported by the National Science Foundation under Grant no. 0098151.
[2] This research was carried out in part while this author was at the Center for Automation Research, University of Maryland, College Park, MD, USA.

0167-9473/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2006.08.033

1. Introduction

The problem of fitting a straight line to a finite collection of points in the plane is an important problem in statistical estimation. Robust estimators are of particular interest because of their lack of sensitivity to outlying data points. The basic measure of the robustness of an estimator is its *breakdown point*, that is, the fraction (up to 50%) of outlying data points that can corrupt the estimator. Rousseeuw's least median-of-squares (LMS) regression (line) estimator (Rousseeuw, 1984) is among the best known 50% breakdown-point estimators.

The *LMS line estimator* (with intercept) is defined formally as follows. Consider a set S of n points (x_i, y_i) in $\Re^2$. The problem is to estimate the parameter vector $\theta^* = (\theta_1^*, \theta_2^*)$ which best fits the data by the linear model

$$y_i = x_i\theta_1^* + \theta_2^* + e_i, \quad i = 1, \ldots, n,$$

where $(e_1, \ldots, e_n)$ are the (unknown) errors. Given an arbitrary parameter vector (θ_1, θ_2), let $r_i = y_i - (x_i\theta_1 + \theta_2)$ denote the ith residual. The LMS estimator is defined to be the parameter vector that minimizes the median of the squared residuals. This should be contrasted with ordinary least squares (OLS), which minimizes the sum of the squared residuals. Intuitively, if less than half of the points are outliers, then these points cannot adversely affect the median squared residual.

In addition to having a high breakdown-point, the LMS estimator is regression-, scale-, and affine-equivariant, i.e., its estimate transforms "properly" under these types of transformations. (See Rousseeuw and Leroy, 1987, pp. 116–117, for exact definitions.) The LMS estimator may be used in its own right or as an initial step in a more complex estimation scheme (Yohai, 1987). It has been widely used in numerous applications of science and technology, and is considered a standard technique for robust data analysis. Our main motivation for studying the LMS estimator stems from its usage in computer vision. (See, e.g., Meer et al., 1991; Netanyahu et al., 1997; Stein and Werman, 1992; Stewart, 1996.) For example, Fig. 1 demonstrates the enhanced performance obtained by LMS versus OLS in detecting straight road segments in a noisy aerial image (Netanyahu et al., 1997).

In some applications the number of outliers may differ from 50%, so it is common to generalize the problem definition to minimize the squared residual of a specified quantile. This is also called the least-quantile squared (LQS) estimator (Rousseeuw and Leroy, 1987). In our formulation, we will assume that in addition to the point set S, the algorithm is given a *residual quantile* q, where $0 < q \leqslant 1$, and it returns the line that minimizes the $\lceil nq \rceil$th smallest squared residual.

Define a *strip* $\sigma = (\ell_1, \ell_2)$ to be the closed region of the plane lying between two parallel lines ℓ_1 and ℓ_2. The *vertical height* of a nonvertical strip, height(σ), is the length of its intersection with any vertical line. For $0 < q \leqslant 1$, define LMS(S, q) to be the strip of minimum vertical height that encloses at least $\lceil nq \rceil$ points from the set S. It is easy to see that the center line of LMS$(S, \frac{1}{2})$ is the LMS regression line, and the height of the strip is twice the median of the absolute residuals. We call LMS$(S, 0.5)$ the *LMS strip*. The algorithms discussed here can be generalized to compute the strip that minimizes the perpendicular width of the strip. This is done by scaling each vertical height by the cosine of the angle corresponding to the slope, as described by Edelsbrunner and Souvaine (1990).

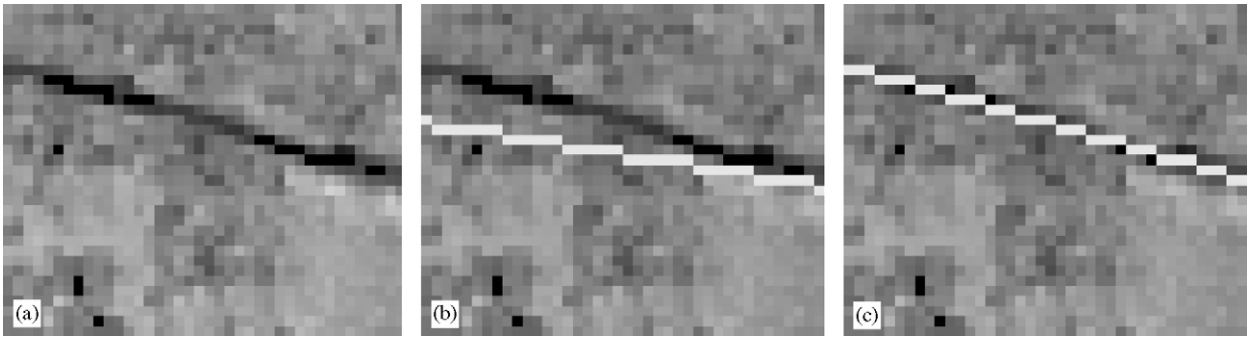

Fig. 1. (a) A road segment with outlying pixels; (b) its OLS line fit; (c) its LMS line fit.

1.1. Exact algorithms for LMS

The best algorithm known for finding the LMS strip from the perspective of worst-case asymptotic efficiency is the topological plane-sweep algorithm due to Edelsbrunner and Souvaine (1990). (See also Souvaine and Steele, 1987 for background and Rafalin et al., 2002 for an implementation.) It runs in $O(n^2)$ time and requires $O(n)$ space. Since the algorithm is based on traversing the entire line arrangement, whose complexity is quadratic in n, there is no obvious way to improve its worst case running time. Stromberg described how to compute LMS exactly in higher dimensions by enumeration of elemental sets (Stromberg, 1993), and the complexity increases exponentially with the dimension. His algorithm runs in $O(n^4 \log n)$ time for the LMS line estimator in the plane. A slightly improved algorithm which runs in $O(n^4)$ time is due to Agulló (1997).

More efficient algorithms exist for other robust line estimators, such as the Theil-Sen and repeated median line estimators (Cole et al., 1989; Dillencourt et al., 1992; Matoušek, 1991b; Matoušek et al., 1998) and estimators based on the regression depth (Langerman and Steiger, 2000). These algorithms rely on properties of these estimators that admit rapid searching for the optimal slope. The LMS estimator does not possess these properties, however. Indeed, recent research has shown that in some models of computation, no algorithm that is faster than $\Omega(n^2)$ exists (Erickson et al., 2004). This suggests that if a significantly faster algorithm did exist, dramatically different computational techniques would need to be employed.

Quadratic running time is unacceptably high for many applications involving large data sets. For this reason, what is often used in practice is a simple $O(n \log n)$ randomized algorithm due to Rousseeuw and Leroy (1987), called PROGRESS. (See also Rousseeuw and Hubert, 1997.) When used as a line estimator, this algorithm randomly samples some constant number of pairs of points. The number of pairs is independent of n, and it depends instead on the expected fraction of outliers and the user's desired confidence in the final result. For each sampled pair, the slope of the line passing through these points is computed, and in $O(n \log n)$ time it is possible to compute the line having this slope that minimizes the kth smallest squared residual, by reduction to a 1-dimensional LMS problem. Rousseeuw and Leroy (1987) show that if a constant fraction of the points do indeed lie on (or very close to) a line, then this sampling procedure will return the correct (or nearly correct) result with the desired confidence. Mount et al. (2004) generalized this analysis by considering how many points should be sampled as a function of the distribution of the inlying points and the desired accuracy of the final estimate. However, if the data fail to satisfy these assumptions, then there are no guarantees (even probabilistic) on the quality of the results of this algorithm. Likewise, the *feasible set algorithm* Hawkins (1993) which is based on random sampling and local improvement, can provide good performance in practice, but does not guarantee any error bounds on the computed estimate either. Hence, none of these approaches provides a truly satisfactory solution to the problem.

1.2. Approximating LMS

LMS can be viewed as a geometric optimization problem. Given the apparently high complexity of solving LMS exactly, it is natural to consider whether it can be solved *approximately* more efficiently. In general, given $\varepsilon \geqslant 0$, an *ε-approximation algorithm* for some (minimization) optimization problem is an algorithm that on any input generates a feasible solution whose cost is at most a factor of $(1+\varepsilon)$ greater than the optimal solution. Since the objective of LMS is to minimize the median residual, one reasonable approach is to compute a *residual approximation*. That is, in addition to the point set S and the quantile residual q, the algorithm is given a real parameter $\varepsilon_r \geqslant 0$. The algorithm computes a strip containing at least $\lceil nq \rceil$ points whose height is at most $(1+\varepsilon_r)$ times the height of $\mathrm{LMS}(S, q)$. Another reasonable approach is to compute a *quantile approximation*. The algorithm is given a quantile error ε_q, where $0 \leqslant \varepsilon_q < 1$. From q and ε_q we define bounds on the allowed quantile variation, for example,

$$q^+ = q \quad \text{and} \quad q^- = (1 - \varepsilon_q)q.$$

Such an approximation returns a strip that contains at least $\lceil nq^- \rceil$ points and is no wider than $\mathrm{LMS}(S, q^+)$. Thus, the approximation is slightly less robust but does not increase the size of the strip. (There are other reasonable ways to define q^+ and q^- based on a target quantile q and an error ε_q. Our complexity bounds hold under the general assumptions that $q^- \leqslant q \leqslant q^+$, $q^-/q^+ = 1 - \varepsilon_q$, and that all these quantities are fixed constants, independent of the input size.)

Generalizing these two approximations, we may also define a *hybrid approximation*. Given q, ε_r, and ε_q, such an approximation produces a strip containing at least a fraction q^- of the points that is no wider than $(1+\varepsilon_r)$ times the

height of LMS(S, q^+). In applications where running time is critical, the flexibility to approximate provides the user with a trade-off between accuracy and execution time.

A deterministic quantile approximation algorithm was presented by Mount et al. (2000). However, this algorithm makes heavy use of the sophisticated computational construct of ε-approximations (Matoušek, 1991a), and its implementation would involve very large constant factors, and hence it is of dubious practical value. Olson (1997) presented a randomized residual approximation algorithm, which produces a strip whose vertical width is at most twice that of the LMS strip, that is, $\varepsilon_r = 1$. This algorithm runs in $O(n \log^2 n)$ time in the plane and in $O(n^{d-1} \log n)$ time for fixed dimensions $d \geqslant 3$. This approximation bound is too weak to be of significant value for many applications, however. Har-Peled and Wang (2003) presented an $O(n + (k/\varepsilon_r)^c)$ time algorithm, which computes an ε_r-residual approximation to the LMS line estimator in any dimension, assuming k outliers. Here c is a constant that depends on the dimension. Unfortunately, this is only of value for very small numbers of outliers. Erickson et al. (2004) presented a sophisticated randomized residual approximation that runs in $O((1/\varepsilon_r)n^{d-1} \log n)$ time in dimension d, but its practical value is unclear.

1.3. Summary of results

Our goal in this paper is to investigate alternative computational approaches to LMS, which achieve efficiency in one of two ways:

- Computing an approximation (residual, quantile, or hybrid) to LMS.
- Computing an exact solution to LMS, but exploiting the fact that, for many typical input instances in practice, a significant fraction of the points are expected to lie on or near the LMS line.

We present two algorithms for LMS. The first is a conceptually simple quantile approximation algorithm. Assuming that q and ε_q are fixed constants independent of n, the algorithm's running time is $O(n \log n)$. The second is a more general and practical algorithm. It can be run either as an hybrid approximation algorithm or as an exact algorithm (by setting $\varepsilon_r = \varepsilon_q = 0$). It requires no complex data structures, and it is based on variations of the standard Merge-Sort algorithm (Cormen et al., 1990) to perform inversion counting (Dillencourt et al., 1992) and topological plane sweep (Edelsbrunner and Guibas, 1989).

The second algorithm employs a method, which we call *slope decomposition*, which recursively prunes away regions of the space of possible line slopes that cannot contain the LMS line. We show that for fixed q and $\varepsilon_q > 0$, this algorithm has an expected asymptotic running time of $O(n \log^2 n)$. In this respect the algorithm is similar to other practical geometric approximation algorithms based on a spatial subdivision, such as Har-Peled's diameter approximation (Har-Peled, 2001). We have implemented this algorithm, and we present empirical evidence that the algorithm's running time is quite efficient if either (1) it is run as an approximation algorithm or (2) it is run as an exact algorithm on data sets in which a sufficiently large fraction of points are close to a line.

Both algorithms are examples of *Las Vegas randomized algorithms*. This means that each algorithm makes discrete random choices, and our bounds on the running time of the algorithm hold in expectation when averaged over these random choices. (In fact our running time bounds not only hold in expectation, but hold with high probability, where the probability rapidly approaches unity as n increases.) It is important to note that, irrespective of the random choices, the algorithm always produces a result that is correct to within the user-provided approximation bounds. This is in contrast to a Monte Carlo randomized algorithm, which may produce an erroneous result but with provably low probability. It is also in contrast to classical expected-case algorithm analysis, in which the expected-case running time of the algorithm is a function of the input distribution. Our expected running times depend only on the random choices, and do not depend on any properties of the input. Randomized algorithms have become particularly popular in the field of algorithm design because they often admit much simpler and more elegant algorithmic solutions (Motwani and Raghavan, 1995).

The rest of the paper is organized as follows. In the next section, we provide a brief overview of our computational methods and provide basic definitions and notation, which will be used throughout the paper. In Section 3 we present the simple randomized algorithm, and in Section 4 we present the slope-decomposition algorithm. In Section 5 we present a theoretical analysis of the expected running time of the slope-decomposition algorithm, and in Section 6 we present empirical results on the running time of this algorithm.

2. Computational methods

Both of the algorithms presented here employ a similar computational framework, which is based on point-line duality and searching line arrangements in the plane. Duality has been shown to be very useful in many efficient algorithms for robust statistical estimation problems (Cole et al., 1989; Dillencourt et al., 1992; Edelsbrunner and Souvaine, 1990; Erickson et al., 2004; Matoušek, 1991b; Souvaine and Steele, 1987). We believe that the computational methods presented here may be of wider interest in other problems in computational statistics. In this section we present a brief overview of these techniques and how they will be applied in this paper.

2.1. LMS and point-line duality

Consider two planes, called the *primal plane* and *dual plane*. To distinguish them we let x and y denote the coordinate axes for the primal plane and u and v denote the axes for the dual plane. The *dual transformation* maps a primal point $p = (a, b)$ to the dual line $v = au - b$, which we denote by p^*. Conversely, it maps the (nonvertical) primal line $\ell : y = ax - b$ to the dual point (a, b), which is denoted by ℓ^*.

It is easy to show that a number of affine properties are preserved under the dual transformation. (See Edelsbrunner, 1987 for further information.) In particular, the following properties will be relevant to our later presentation. (See Fig. 2.) The first and third properties follow directly from the definition of the dual transformation and the second property follows from the first.

Order reversing: Point p lies above/on/below line ℓ in the primal plane if and only if the dual line p^* passes below/on/above the dual point ℓ^*, respectively.

Intersection preserving: Lines ℓ_1 and ℓ_2 intersect at point p in the primal plane if and only if dual line p^* passes through the dual points ℓ_1^* and ℓ_2^*.

Strip enclosure: A strip of vertical height h bounded by two nonvertical parallel lines ℓ_1 and ℓ_2 is mapped to a pair of dual points ℓ_1^* and ℓ_2^* such that the segment joining them is vertical and of length h. A point p lies within the strip if and only if the dual line p^* intersects this segment. We use the term *bracelet* to refer to the vertical segment resulting from applying the dual transformation to a strip.

Using these properties, we can interpret LMS in the dual setting. Recall that solving LMS for a quantile q on a set of n points reduces to computing the strip of minimum vertical width that encloses at least $\lceil nq \rceil$ of the points. By the strip enclosure property, this is equivalent to finding the bracelet of minimum length such that $\lceil nq \rceil$ of the dual lines intersect this segment. A bracelet that intersects at least k dual lines is called a *k-bracelet*. Therefore, the LMS problem is equivalent to computing the $\lceil qn \rceil$-bracelet of minimum length, which we call the *LMS bracelet*. (See Fig. 3.) The LMS approximation problems can also be reformulated in the dual setting. For example, a hybrid-approximation returns a $\lceil nq^- \rceil$-bracelet whose length is at most $(1 + \varepsilon_r)$ times the length of the LMS(S, q^+) bracelet.

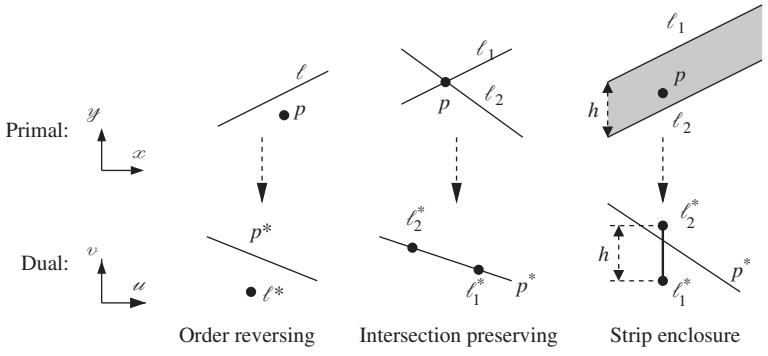

Fig. 2. Dual transformation and properties.

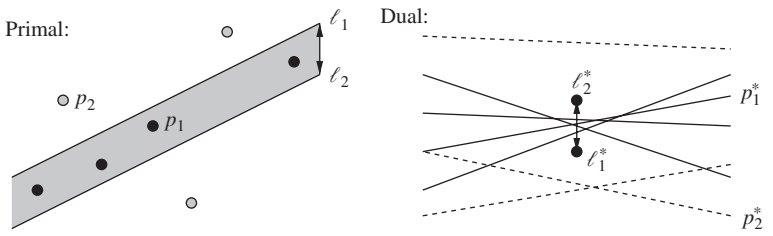

Fig. 3. The strip bounded by lines ℓ_1 and ℓ_2 in the primal plane dualizes to the bracelet $\overline{\ell_1^* \ell_2^*}$ in the dual plane. A point such as p_1 that lies inside the strip dualizes to a line p_1^* intersecting the bracelet, and a point such as p_2 that lies above the strip dualizes to a line p_2^* that passes below the bracelet.

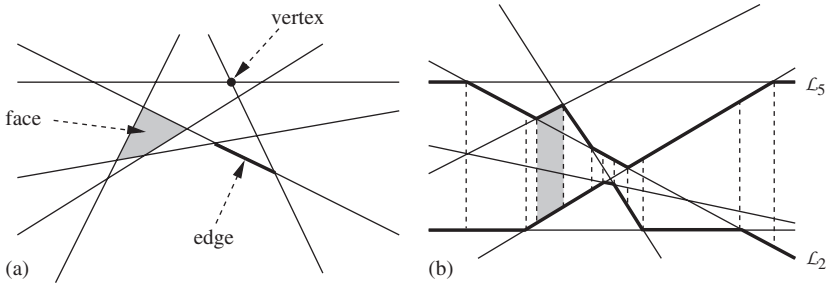

Fig. 4. Line arrangement (a), and levels $\mathscr{L}_2$ and $\mathscr{L}_5$ and the 4-trapezoids they define (b).

2.2. Line arrangements, levels, and LMS

A set of lines in the plane subdivides the plane into a cell complex, called a *line arrangement*. Consider a set of n nonvertical lines in the plane, and assume they are in general position (meaning that no two lines are parallel and no three intersect at the same point). It is easy to show that this complex has $\binom{n}{2}$ vertices, n^2 (possibly unbounded) edges, and $\binom{n}{2} + n + 1$ (possibly unbounded) faces (Edelsbrunner, 1987). (See Fig. 4(a).)

Given an arrangement of n lines, for $1 \leqslant j \leqslant n$ the *jth level* of the arrangement, denoted $\mathscr{L}_j$, is defined to be the locus of points such that there are at most $j-1$ lines lying strictly below the point, and at most $n-j$ lines lying strictly above the point (Edelsbrunner, 1987). (See Fig. 4(b).) Every point on every line of the arrangement lies on some level, and vertices of the arrangement lie on multiple levels. A level is a polygonal chain extending from $u = -\infty$ to $u = +\infty$, and is monotone with respect to the u-axis (meaning that any vertical line intersects the level in a single point). If $i < j$, then $\mathscr{L}_i$ intersects any vertical line on or below $\mathscr{L}_j$. It is easy to see that any bracelet that joins a point on $\mathscr{L}_i$ to $\mathscr{L}_j$ intersects at least $j - i + 1$ lines of the arrangement.

It will be useful to decompose the region between two levels into a collection of trapezoids with vertical sides. Consider any positive integer k, and for each $j \leqslant n - k + 1$, consider a set of trapezoids formed by shooting a vertical ray up from each vertex of $\mathscr{L}_j$ until it hits $\mathscr{L}_{j+k-1}$ and shooting a vertical ray down from each vertex of $\mathscr{L}_{j+k-1}$ until it hits $\mathscr{L}_j$. (See Fig 4.) Such a trapezoid is called a *k-trapezoid* since each vertical segment connecting the top of the trapezoid to the bottom is a bracelet intersecting at least k lines of the arrangement. It is easy to see that the shortest bracelet may be assumed to have an arrangement vertex as one of its endpoints, for otherwise it is possible to slide it either to the left or right without increasing its length. Thus, the shortest bracelet intersecting at least k dual lines is realized by the side of some k-trapezoid. This provides us with the following dual characterization of the LMS problem, which we will use throughout the paper.

Lemma 1. *LMS*(Sq) *is the dual of the shortest vertical side among all the k-trapezoids of the dual arrangement of S, where* $k = \lceil nq \rceil$.

2.3. Searching an arrangement

An arrangement of n lines in the plane has $O(n^2)$ vertices. Since at least one vertex of each k-trapezoid is a vertex of the arrangement, and since each vertex of the arrangement is incident to at most four k-trapezoids (upper-left, upper-right, lower-left, and lower-right), it follows that for any fixed k there are $O(n^2)$ k-trapezoids. This suggests a naive way of solving LMS, namely by processing the entire arrangement and computing all the k-bracelets. Edelsbrunner and Souvaine's $O(n^2)$ time and $O(n)$ space algorithm for the exact LMS line estimator does exactly this (Edelsbrunner and Souvaine, 1990). It employs a very clever technique, called topological plane sweep, which traverses the entire arrangement without explicitly storing the arrangement in memory.

Given the quadratic size of the dual line arrangement, any algorithm that constructs every k-trapezoid (or every k-bracelet) is doomed to $O(n^2)$ running time. In this paper we show that it is possible to do better, either when approximation is involved or when a significant number of the points lie close to the line of fit. Our approach is to use randomization to identify those regions of the arrangement that hold promise of containing the LMS bracelet, and eliminate the rest from consideration.

The two algorithms presented here achieve this goal in two different ways. The first algorithm makes use of random sampling. Rather than constructing the entire arrangement, we compute a random sample of an appropriately chosen size r from the original set of lines. By choosing r to be asymptotically smaller than $\sqrt{n}$, we can construct the entire arrangement of sample lines in $O(r^2) \leqslant O(n)$ time and space. We then compute the smallest k-bracelet, where k is roughly rq. A probabilistic argument shows that the resulting bracelet is expected to intersect close to the desired number of nq lines of the original arrangement, and so this is a likely candidate for being a quantile approximation to the LMS bracelet. But this is not a guarantee of correctness. To convert this into a Las Vegas algorithm (meaning that the result is always correct), we show how to test efficiently that the random sample is sufficiently well balanced. If so we accept the answer, and if not we reject the answer and generate another random sample. We show how to apply a data structure for simplex range searching developed in computational geometry (Matoušek, 1992) in order to perform this test.

Although the above algorithm is conceptually simple, the range searching data structure is quite complicated and difficult to implement. Our second algorithm is based on a novel method for searching the line arrangement without constructing it explicitly. This method involves decomposing the line arrangement into vertical slabs. Because the horizontal coordinate in dual space corresponds to the slope of the line in primal space, each such slab represents an interval of (primal) slopes. We call this method *slope decomposition*. An important element of this approach is the ability to reason about the structure of the arrangement within each such slab, without explicitly constructing the slab. (This general approach has also been used for the Theil-Sen line estimator (Cole et al., 1989; Dillencourt et al., 1992; Matoušek, 1991b).)

As an example, let us consider the concrete problem of counting the number of arrangement vertices that lie within a given slab of the arrangement, which is one of the utilities that will be needed by our algorithm. Although this number may generally be quadratic in n, there is a simple manner for counting the number of such vertices in $O(n \log n)$ time, which is based on counting the number of inversions in a sequence of integers. An *inversion* in a numeric sequence $\langle a_i \rangle_{i=1}^n$ is a pair of indices $i < j$ such that $a_i > a_j$. It is well known that the number of inversions in a sequence can be counted in $O(n \log n)$ time by a simple modification of the Merge-Sort algorithm[1] (Cormen et al., 1990).

In order to count the number of vertices of the arrangement that lie within the slab, we first sort and label the lines of the arrangement according to the order in which they intersect the left side of the slab. This can be done in $O(n \log n)$ time. We then transfer the labels of each line to the right side of the slab and consider the resulting sequence of integers. It is easy to see that each inversion in the resulting sequence corresponds to the intersection of two lines within the slab, and therefore the number of inversions in the permuted sequence is equal to the number of line intersections. (See Fig. 5.) In this way, we can extract structural information about the slab, without explicitly constructing it. In Section 4 we will provide further details about the algorithm's operation.

[1] Here is a brief intuitive explanation for those familiar with Merge-Sort. During each step of the merging phase of the algorithm, for each element x of the left subsequence, the number of elements of the right subsequence that are of lesser value than x is known. These elements all appeared after x in the original sequence, and so they all represent inversions. These counts are summed over all the merge steps yielding the total number of inversions. A more detailed explanation can be found in Dillencourt et al. (1992).

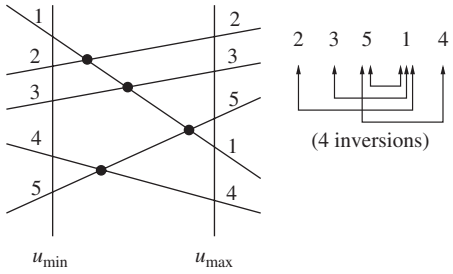

Fig. 5. Counting arrangement vertices through inversion counting.

quantApprox(S, q, ε_q) :

(1) Let $q^+ \leftarrow q$ and $q^- \leftarrow (1-\varepsilon_q)q^+$.

(2) For any $\delta > 0$ and a constant c (to be specified later) let

$$r \leftarrow \min\left(n, c \cdot \max\left(\frac{1}{\varepsilon_q q^+}, n^{1/4-\delta}\right)\right).$$

Randomly sample a set R of r (dual lines) of S. Let $q' \leftarrow (q^- + q^+)/2 = (1-\varepsilon_q/2)q^+$, and let $k \leftarrow \lceil rq' \rceil$.

(3) Compute the $(k+1)$-trapezoids of the arrangement defined by R. For each such trapezoid, compute the number of lines of S that intersect the trapezoid. If for any trapezoid this number is at least $\lceil nq^+ \rceil$ then reject the sample and go back to Step (2).

(4) Compute the k-trapezoids of the line arrangement defined by R. Let $\sigma_{\min}$ be the shortest vertical side of any k-trapezoid. Count the number of lines of S that intersect $\sigma_{\min}$. If this count is less than $\lceil nq^- \rceil$ then reject the sample and go back to Step (2).

(5) Return (the dual of) $\sigma_{\min}$ as the desired approximate LMS strip.

Fig. 6. Quantile approximation for LMS.

3. A randomized quantile approximation algorithm

In this section we present an efficient randomized (Las Vegas) quantile approximation algorithm for LMS. Recall that the inputs consist of a set S of n lines in the dual plane, the desired quantile q $(0 < q < 1)$, and the quantile error ε_q $(0 < \varepsilon_q < 1)$, and we let $q^+ = q$ and $q^- = (1-\varepsilon_q)q^+$. Recall that the goal is to return a $\lceil nq^- \rceil$-bracelet that is no wider than LMS(S, q^+).

It will simplify the presentation to assume that the points are in general position. In particular, we assume that no two points of S have the same x-coordinate, no three points are collinear and no two distinct pairs of points define parallel lines. These assumptions can be obviated by taking greater care in the implementation of the algorithm, and can certainly be overcome formally through the use of symbolic perturbation approaches from computational geometry for handling degenerate point configurations (Edelsbrunner and Mücke, 1990).

Here is an intuitive description of the algorithm. Let $q' = (q^- + q^+)/2$. The algorithm begins by randomly sampling a sufficiently large set R of dual lines and computes $\sigma = \text{LMS}(R, q')$. Since this bracelet intersects a fraction of $q' > q^-$ of the sampled lines, we would expect it to intersect at least a fraction of q^- lines of S. Since it is the smallest bracelet that intersects a fraction $q' < q^+$ of the sample, we would expect that it is not longer than the smallest bracelet that intersects a fraction of q^+ of S. Thus, we expect that σ is the desired result. To produce a Las Vegas algorithm, we will show how to test these two conditions efficiently. If either is violated, then a new sample is generated. A detailed description is presented as procedure *quantApprox* in Fig. 6.

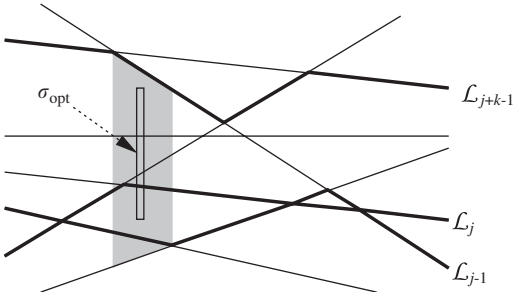

Fig. 7. Proof of Lemma 2.

Lemma 2. *The result $\sigma_{\min}$ returned by the procedure quantApprox is an ε_q-quantile approximation to* $\text{LMS}(S, q^+)$.

Proof. It suffices to show that $\sigma_{\min}$ is intersected by at least $\lceil nq^- \rceil$ lines of S and is no wider than $\text{LMS}(S, q^+)$. Step (4) guarantees that the former condition holds. To establish the latter condition let $\sigma_{\text{opt}} = \text{LMS}(S, q^+)$, and suppose to the contrary that σ_{opt} were smaller than $\sigma_{\min}$. Consider the smallest j such that the lower endpoint of σ_{opt} lies on or below level $\mathscr{L}_j$ in the sampled arrangement. (See Fig. 7.) The upper endpoint of σ_{opt} must lie on or below level $\mathscr{L}_{j+k-1}$, for otherwise the k-trapezoid between these levels that intersects σ_{opt} would have a smaller height than σ_{opt}, contradicting the choice of $\sigma_{\min}$.

Thus, there is a $(k+1)$-trapezoid bounded by levels $\mathscr{L}_{j-1}$ and $\mathscr{L}_{j+k-1}$ that fully contains σ_{opt}. Since σ_{opt} is intersected by at least $\lceil nq^+ \rceil$ lines, this $(k+1)$-trapezoid is intersected by at least as many lines. However, the algorithm would have rejected this sample in Step (3), a contradiction. □

The execution time of the algorithm depends on the time needed to perform Steps (2)–(4) times the number of samples taken by the algorithm. Later we will show that the probability that a sample is rejected is very low, implying that the expected number of iterations is a constant. For now let us consider the running time of the algorithm for a single sample.

Step (2) can be performed in $O(n^{1/4-\delta}) < O(n)$ time. The computation of both the k- and $(k+1)$-trapezoids can be accomplished by a single application of topological plane sweep in $O(r^2) < O(n^{1/2})$ time (Edelsbrunner and Guibas, 1989). Clearly the number of lines intersecting $\sigma_{\min}$ can be counted in $O(n)$ time, which bounds the running time for Step (4).

We reduce Step (3) to a range searching problem. Let τ be a $(k+1)$-trapezoid of the arrangement defined by R. Assuming general position, except for the lines that define τ, each line of the arrangement that intersects τ intersects exactly two sides of τ. Thus, it suffices to count the number of lines of S that intersect each side of τ and then divide the sum by 2. Each side of a trapezoid in dual space corresponds to the double-wedge (two of the four quadrants defined by two intersecting lines) in the primal plane (de Berg et al., 2000). A dual line intersects a side of τ if and only if the corresponding primal point lies within the double wedge. Thus, counting the number of lines that intersect the four sides of τ reduces to answering four double-wedge range queries in the primal plane. Each such range query can be reduced to two (unbounded) simplex range queries in the plane.

Matoušek has shown that with $O(n \log n)$ preprocessing time and $O(n)$ space, simplex range searching queries in the plane can be answered in $O(n^{1/2})$ time (Matoušek, 1992). In fact, Matoušek's results are stronger in that they admit space–time trade-offs. Given m units of storage, where $m \geqslant n$, queries can be answered in time $O((n/m^{1/d}) \log^{d+1}(m/n))$. For our purposes, this additional flexibility is not required. (Although Matoušek's data structure is asymptotically the best known, it is rather complex. Simpler methods such as kd-trees (Samet, 1990) might provide better performance in practice.) Since there are $O(r^2)$ trapezoids, and hence $O(r^2)$ queries to be answered, the total time to answer all queries is $O(r^2 n^{1/2} (\log n)^{O(1)})$, which is $O(n)$ time, since q^+ and ε_q are fixed. Thus, the total running time per stage is dominated by the $O(n \log n)$ preprocessing time.

All that remains to show is that the probability that the sample is rejected from either Step (3) or (4) of the algorithm is sufficiently small. From this it will follow that the expected number of stages is $O(1)$. In fact, we will show that the probability of rejecting a sample is $O(1/(nq\varepsilon_q))$, implying that the probability that more than s samples are required is $O(1/(nq\varepsilon_q)^s)$. Thus, the $O(n \log n)$ running time occurs with high probability.

Our analysis makes use of the Chernoff bounds on the tail of the binomial distribution (Chernoff, 1952). Proofs of these particular bounds have been presented by Motwani and Raghavan (1995) and Hagerup and Rüb (1990).

Lemma 3 (*Chernoff bounds*). *Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli trials such that for $0 \leqslant p \leqslant 1$ and $1 \leqslant i \leqslant n$, $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$. Then for $X = \sum_i X_i$, $\mu = np$, and any $0 < \delta \leqslant 1$*

$$\Pr[X < (1-\delta)\mu] < \exp(-\mu\delta^2/2),$$
$$\Pr[X > (1+\delta)\mu] < \exp(-\mu\delta^2/3).$$

We begin by bounding the probability that the algorithm rejects a sample in Step (3). Recall that this happens if the arrangement defined by the random sample R contains some $(k+1)$-trapezoid that is intersected by at least $\lceil nq^+ \rceil$ lines of S. Recall that $q' = (q^- + q^+)/2$. For all sufficiently large n, a $(k+1)$-trapezoid intersects a fraction of roughly $q' < q^+$ of the random sample, and so we would not expect that a random trapezoid would be intersected by a fraction of the lines of S larger than q^+. We will see that this probability decreases exponentially with $r\varepsilon_q q^+$, and hence at least polynomially in n. The proof employs a structure common to other randomized analyses in computational geometry. See, for example, Mulmuley (1994).

Lemma 4. *The probability of rejecting a sample in Step* (3) *of the algorithm is* $O(1/(nq\varepsilon_q))$.

Proof. Let $T(S)$ denote the set of all j-trapezoids for $2 \leqslant j \leqslant n$. Each trapezoid $\tau \in T(S)$ can arise as a $(k+1)$-trapezoid of R's arrangement only if four *defining lines* of S are sampled. These consist of the two lines forming the upper and lower sides of the trapezoid together with the two lines whose intersection points with the upper and lower sides generate the arrangement vertices that determine the left and right sides of the trapezoid. Let these defining lines be denoted $D(\tau)$. Since a trapezoid is defined by at most four lines with have $|D(\tau)| \leqslant 4$. For any random sample R, a $(k+1)$-trapezoid τ for R is an element of $T(S)$. Given a trapezoid $\tau \in T(S)$, let $I(\tau)$ denote the number of lines of S that intersect τ. We say that a trapezoid of $T(S)$ is *crowded* if $I(\tau) \geqslant \lceil nq^+ \rceil$. Let $C(S) \subseteq T(S)$ denote this set of crowded trapezoids. We are interested in bounding the probability that at least one of the crowded $(k+1)$-trapezoids is a $(k+1)$-trapezoid in the arrangement generated by R.

Given any crowded trapezoid τ, let $E_D(\tau)$ denote the event that $D(\tau) \subseteq R$, and $E_I(\tau)$ denote the event that τ is crowded and is intersected by $k+1$ lines of the random sample. If $\tau \in C(S)$ is to be a $(k+1)$-trapezoid of R, then clearly both $E_D(\tau)$ and $E_I(\tau)$ must occur. Since these conditions are necessary for the existence of τ as a $(k+1)$-trapezoid, we have

$$\Pr[\tau \text{ is a crowded } (k+1)\text{-trapezoid of } R] \leqslant \Pr[E_D(\tau) \wedge E_I(\tau)] = \Pr[E_D(\tau)] \cdot \Pr[E_I(\tau) | E_D(\tau)].$$

First we bound $\Pr[E_I(\tau)|E_D(\tau)]$. Assuming that we have already sampled the lines of $D(\tau)$, let us consider the remaining sampled lines of R. Letting r' denote the number of such lines, we have $r' = r - |D(\tau)| \geqslant r - 4$. These lines are a subset of the remaining lines of S. Letting n' denote the size of this subset, we have $n' = n - |D(\tau)| \geqslant n - 4$. We may view these as r' Bernoulli trials, where a sampled line is considered a success if the line intersects τ and a failure otherwise. Let p denote the probability that a randomly sampled line from the remaining n' lines intersects τ. There are at least $I(\tau) - 4$ lines of the remainder of S intersecting τ. Since we are only interested in crowded trapezoids, we assume that $I(\tau) \geqslant nq^+$, implying that

$$p \geqslant \frac{I(\tau) - 4}{n'} \geqslant \frac{nq^+ - 4}{n} = \left(1 - \frac{4}{nq^+}\right) q^+.$$

The expected number of successes is $\mu = pr'$. Since $r' \geqslant r - 4$, we have

$$\mu \geqslant (r-4)\left(1 - \frac{4}{nq^+}\right) q^+.$$

Since $E_I(\tau)$ has occurred, τ is intersected by exactly $k+1$ lines of the sample. Recalling that $k = \lceil rq' \rceil$ and that $q' = (1 - \varepsilon_q/2)q^+$, this implies that the number of actual successes observed in the r' trials is

$$X = k + 1 - |D(\tau)| < k - 1 \leqslant rq' = r\left(1 - \frac{\varepsilon_q}{2}\right) q^+.$$

We will apply the Chernoff bound to establish an upper bound on the probability of this occurring. To do this we first observe that, since q^+ is fixed, as n increases (and hence so does r), $r = (r-4)(1-4/(nq^+)) + \mathrm{O}(1)$. Thus, by increasing the constant factor $(1-\varepsilon_q/2)$ slightly to $(1-\varepsilon_q/4)$, it follows that, for all sufficiently large values of n (depending on q^+ and ε_q) we have

$$r\left(1-\frac{\varepsilon_q}{2}\right) \leqslant (r-4)\left(1-\frac{4}{nq^+}\right)\left(1-\frac{\varepsilon_q}{4}\right).$$

Now multiplying both sides by q^+ and using our bounds on X and μ we have $X < \mu(1-\varepsilon_q/4)$. That is, the number of successes observed in the trials falls short of the expected value μ by a factor of $(1-\varepsilon_q/4)$. By applying the Chernoff bound with $\delta = \varepsilon_q/4$, it follows that the probability of this occurring is at most $\exp(-\mu\varepsilon_q^2/32)$. Since $\mu \geqslant rq^+ = cn^{1/4-\delta}q^+$, this probability decreases faster than any polynomial in n, and hence is certainly $\mathrm{O}(1/(n^2\varepsilon_q q^+))$.

Having bounded the probability that a crowded trapezoid occurs given that its defining lines are in the sample, it follows that the probability of finding a crowded trapezoid in the arrangement defined by the random sample is at most

$$\begin{aligned}&\sum_{\tau\in C(S)} \Pr[E_D(\tau)]\cdot\Pr[E_I(\tau)|E_D(\tau)]\\ &\quad\leqslant \sum_{\tau\in C(S)} \mathrm{O}\left(\frac{1}{n^2\varepsilon_q q^+}\right)\Pr[E_D(\tau)] = \mathrm{O}\left(\frac{1}{n^2\varepsilon_q q^+}\right)\sum_{\tau\in C(S)}\Pr[E_D(\tau)] \leqslant \mathrm{O}\left(\frac{1}{n^2\varepsilon_q q^+}\right)\sum_{\tau\in T(S)}\Pr[E_D(\tau)].\end{aligned}$$

Note that the last summation is taken over all possible trapezoids that might arise in the arrangement of any random sample, and hence is equal to the expected number of trapezoids in any such arrangement. Since each such trapezoid is defined by at most four lines, it follows that this is $\mathrm{O}(r^4) = \mathrm{O}(n^{1-4\delta}) = \mathrm{O}(n)$. Combining this with the fact that $q^+ = q$, we obtain the desired probability bound of

$$\mathrm{O}\left(\frac{1}{n^2\varepsilon_q q^+}\right)\mathrm{O}(n) = \mathrm{O}\left(\frac{1}{nq\varepsilon_q}\right).$$

The above analysis holds only for sufficiently large n and r. To fix this, we may adjust the factor of c in Step (2) of the algorithm shown in Fig. 6 to force $r = n$ for all smaller values. By doing so the algorithm will return the exact result. □

Next we bound the probability that a sample is rejected in Step (4) of the algorithm. Recall that a sample is rejected in Step (4) only if a vertical side of some k-trapezoid of R's arrangement intersects fewer than nq^- dual lines of S. For all sufficiently large n, a k-trapezoid intersects a fraction of roughly $q' = (q^- + q^+)/2 > q^-$ of the random sample, and so we would not expect that a random trapezoid would be intersected by a fraction of the lines of S smaller than q^-. We will show that this probability decreases exponentially with $r\varepsilon_q q^+$, and hence at least polynomially in $n\varepsilon_q q^+$.

Lemma 5. *The probability of rejecting a sample in Step* (5) *of the algorithm is* $\mathrm{O}(1/(nq\varepsilon_q))$.

Proof. The proof's structure is very similar to that of Lemma 4, and so we will highlight only the major differences. Let $V(S)$ be the set of vertical sides of the trapezoids $T(S)$ defined in Lemma 4. Each such vertical segment σ is *defined* by three lines of S, denoted $D(\sigma)$. For any random sample R, a vertical side σ of any k-trapezoid for R is an element of $V(S)$. Given a segment $\sigma \in V(S)$, let $I(\sigma)$ denote the number of lines of S that intersect σ. We say that σ is *sparse* if it is intersected by fewer than $\lceil nq^-\rceil$ dual lines of S, that is if $I(\sigma) < \lceil nq^-\rceil$. Let $Sp(S) \subset V(S)$ denote this set of segments. We will bound the probability that at least one of the sparse segments is the side of some k-trapezoid in the arrangement generated by R.

Let $E_D(\sigma)$ denote the event that $D(\sigma) \subseteq R$, and $E_I(\sigma)$ denote the event that σ is sparse and is intersected by k lines of the random sample. If $\sigma \in \mathrm{Sp}(S)$ is to be the side of a k-trapezoid of R, then clearly both $E_D(\sigma)$ and $E_I(\sigma)$ must occur. As in Lemma 4, it suffices to bound the product $\Pr[E_D(\sigma)]\cdot\Pr[E_I(\sigma)|E_D(\sigma)]$. Consider $\Pr[E_I(\sigma)|E_D(\sigma)]$. Assuming that we have sampled the lines in $D(\sigma)$, we consider the remaining sampled lines of R. Letting r' denote the number of such lines we have $r' = r - |D(\sigma)| = r-3$. These lines are a subset of the remaining lines of S whose size is denoted by n', where $n' = n - |D(\sigma)| = n-3$. These are r' Bernoulli trials, where a sampled line is a success if

the line intersects σ. Let p denote the probability that a randomly sampled line from the remaining n' lines intersects σ. There are $I(\sigma) - 3$ of the remaining lines of S intersecting σ. Since we are interested only in sparse segments, we assume that $I(\sigma) \leqslant nq^-$. By the same logic used in Lemma 4, this implies that $p = (I(\sigma) - 3)/n' \leqslant q^-/(1 - 3/n)$. The expected number of successes is $\mu = pr' \leqslant r(1 - \varepsilon_q)q^+/(1 - 3/n)$. Since $E_I(\sigma)$ has occurred and recalling that $k = \lceil rq' \rceil \geqslant r(1 - \varepsilon_q/2)q^+$, the number of successes in the r' trials is

$$X = k - |D(\sigma)| \geqslant r\left(1 - \frac{\varepsilon_q}{2}\right)q^+ - 3.$$

As in Lemma 4, we will apply the Chernoff bound to establish an upper bound on the probability of this occurring. Again, by adjusting the ε_q factor, we observe that for all sufficiently large values of n and r (depending on q^+ and ε_q) we have

$$r\left(1 - \frac{\varepsilon_q}{2}\right)q^+ - 3 > \frac{r(1 - \varepsilon_q)q^+}{1 - 3/n}\left(1 + \frac{\varepsilon_q}{4}\right).$$

By substituting the bounds on X and μ we have $X > \mu(1 + (\varepsilon_q/4))$. By applying the Chernoff bound with $\delta = \varepsilon_q/4$, it follows that the probability of this occurring is $O(1/(n^2\varepsilon_q q^+))$.

Following the same reasoning as in Lemma 4, the probability that a sparse segment occurs as the side of any k-trapezoid in the random arrangement is at most $O(1/(n^2\varepsilon_q q^+)) \sum \Pr[E_D(\tau)]$, where the summation is taken over all the sides of the k-trapezoids in the arrangement of any random sample. Hence, this is equal to the expected number of segments in any such arrangement. Since each segment is defined by at most three lines, it follows that this is $O(r^3) = O(n)$. Combining this with the fact that $q^+ = q$, we obtain the desired bound of $O(1/(nq\varepsilon_q))$.

As in Lemma 4, the above analysis holds only for sufficiently large n and r, but we may adjust the factor of c in Step (2) of the algorithm so that the algorithm returns a correct result for all other cases. □

Determining the exact constraints on the constant c in Step (2) of the algorithm in Fig. 6 would involve a tedious analysis of the asymptotic assumptions used in Lemmas 4 and 5. However, the basic constraints are that n needs to be asymptotically larger than r^4, where r is the size of the random sample, and r needs to be larger than some constant times $1/(q\varepsilon_q)$. If we assume that q and ε_q are constants, independent of n, then the probability of rejecting a sample is $O(1/n)$. This in turn implies that the $O(n \log n)$ running time occurs with high probability.

Theorem 1. *Consider a set S of n points in the plane, a fixed quantile q $(0 < q < 1)$, and a quantile approximation factor ε_q $(0 < \varepsilon_q < 1)$. The* quantApprox *algorithm returns an ε_q-quantile approximation to* $\mathrm{LMS}(S, q)$, *and with high probability it runs in* $O(n \log n)$ *time.*

In general, as long as $1/(q\varepsilon_q)$ grows asymptotically slower than $n^{1/4-\delta}$, for any $\delta > 0$, it follows that the probability of rejecting a sample will be bounded away from 1 for all sufficiently large n, and hence the expected running time will be $O(n \log n)$, although we cannot guarantee the running time bound holds with high probability.

4. A practical approach: slope decomposition

There are a number of reasons that the simple randomized algorithm presented in the previous section is not a good choice for implementation. It relies on fairly sophisticated data structures to perform simplex range searching, and the analysis suggests that the constant factors in sample sizes are uncomfortably large. (Although they could be improved by a more careful analysis.) Furthermore, the algorithm is not sensitive to the structure of the data set, in the sense that its running time is the same irrespective of whether the data set contains a strong linear feature.

In this section we introduce a more practical approach to LMS in the plane. This algorithm is also randomized, but it uses no sophisticated data structures (just arrays) or sophisticated algorithms (just sorting and plane sweep in arrangements). It can be run either as a hybrid approximation algorithm or exactly (by setting both the approximation factors to 0). The algorithm takes advantage of the input structure to improve its efficiency.

Recall that the inputs consist of a set S of n lines in the dual plane, the desired quantile q $(0 < q < 1)$, the quantile error ε_q $(0 \leqslant \varepsilon_q < 1)$, and the residual error ε_r $(\varepsilon_r \geqslant 0)$. As before, we let $q^+ = q$ and $q^- = (1 - \varepsilon_q)q^+$. Recall that the problem is to return a $\lceil nq^- \rceil$-bracelet that is no wider than $(1 + \varepsilon_r)\mathrm{LMS}(S, q^+)$. To simplify the presentation, we define $k^- = \lceil nq^- \rceil$ and $k^+ = \lceil nq^+ \rceil$.

Recall that the problem has been dualized to the (u, v)-plane. The algorithm operates by a branch-and-bound search. It subdivides the dual plane into a collection of pairwise disjoint vertical slabs. More formally, a *slab* is defined to be the set of points of the dual plane whose u-coordinates lie within a half-open interval $(u_{\min}, u_{\max}]$. The algorithm starts with a single slab $(-\infty, +\infty]$, which covers the entire dual plane. Recalling that the u-coordinate of each point in the dual plane represents the slope of a line in the primal plane, each slab intuitively represents a set of primal lines within a given range of slopes, and hence the name *slope decomposition*.

The algorithm's goal is to determine a slab that contains an approximate LMS bracelet and eliminate all the others. For each slab, the intersections of the lines of S with the vertical sides of the slab are sorted from bottom to top. From this an upper bound on the height of the smallest k^--bracelet is computed by considering bracelets along just the left and right sides of the slab. Let $\sigma_{\min}$ denote the smallest such bracelet seen at any point in the algorithm's execution. Within each slab, we will show how to compute a lower bound on the size of any k^+-bracelet lying within this slab. If this lower bound times $(1 + \varepsilon_r)$ is larger than the height of $\sigma_{\min}$, then this slab may be eliminated from further consideration, since it cannot provide a bracelet that contradicts the approximate optimality of $\sigma_{\min}$. If a slab cannot be eliminated by this test, it is split by a vertical line passing through a random intersection point within the slab, and the above processing is performed recursively on the two resulting sub-slabs. This process is repeated until either all slabs are eliminated or until each remaining slab has fewer than $O(n)$ intersection points. At this point, plane sweep may be applied to the slab to compute the shortest k^--bracelet in the slab in $O(n \log n)$ time.

In Sections 4.1 and 4.2 we describe the upper- and lower-bound processing, respectively, and in Section 4.3 we describe the overall algorithm. The algorithm's running time analysis will be presented in Sections 5 and 6.

4.1. Upper bound

Consider any vertical slab. Let u' denote the u-coordinate of either of its two sides. Sort the lines of S from bottom to top according to the v-coordinate of the intersection of the line with the vertical line $u = u'$. (For the special cases $u' \in \{\pm\infty\}$ this involves sorting the lines in slope order.) For finite u' let $\langle e_j(u')\rangle_{j=1}^n$ denote the resulting sorted sequence of points. When u' is clear from context, we shorten the notation to just e_j. (See Fig. 8(a).)

A corresponding set of bracelets for this slope is constructed as follows. Observe that the jth intersection from the bottom, e_j, lies on level $\mathscr{L}_j$ of the dual line arrangement. For j running from 1 to $n - k^- + 1$, consider the bracelet with endpoints (e_j, e_{j+k^--1}). By construction, this bracelet intersects at least k^- lines. The algorithm maintains a variable $\sigma_{\min}$, which denotes the smallest such bracelet encountered at any point in the execution of the algorithm. Thus, height$(\sigma_{\min})$ is an upper bound on the smallest k^--bracelet. (See Fig. 8(a) for the case $k^- = 3$.)

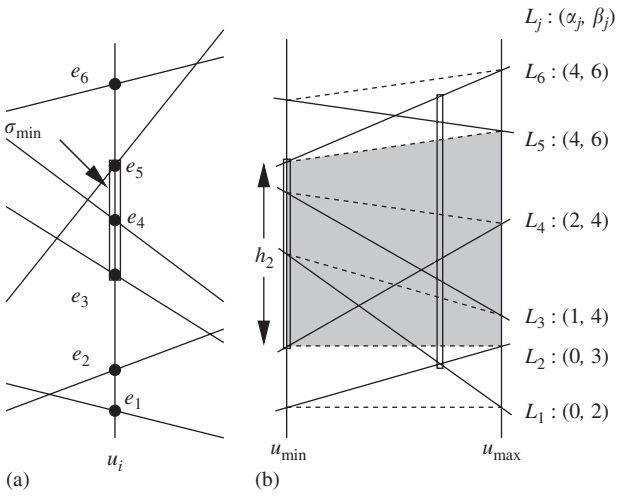

Fig. 8. Sorted intersection points along slab sides (for $k^- = 3$) (a), and pseudo-levels and the lower bound (b).

4.2. Pseudo-levels and the lower bound

Next we consider how to compute a lower bound on the height of any bracelet lying within some fixed slab $U = (u_{\min}, u_{\max}]$. Because it will generally be too expensive to compute the complete line arrangement lying within the slab, we will infer this bound just from relative orders in which the lines of S intersect the two vertical sides of the slab. Since points $e_j(u_{\min})$ and $e_j(u_{\max})$ both lie on level $\mathscr{L}_j$ of the line arrangement, we approximate the intersection of the slab and level $\mathscr{L}_j$ by a line segment joining these two points. We call this the jth *pseudo-level*, denoted $L_j(U)$, or simply L_j when U is clear from context. (Pseudo-levels are shown as broken lines in Fig. 8(b).) It will be convenient to define L_0 to be a pseudo-level at $-\infty$.

Given some slab U, we say that a line $\ell \in S$ lies *strictly below* L_j if its intersections with the left and right sides of U lie strictly below $e_j(u_{\min})$ and $e_j(u_{\max})$, respectively. We define *strictly above* analogously. The following observation about bracelets and pseudo-levels will be useful.

Lemma 6. *Consider a slab U and a bracelet σ that intersects at least one pseudo-level in this slab. Let L_j and $L_{j'}$ denote the lowest and highest pseudo-levels of this slab that σ intersects, respectively. Then σ does not intersect any line of S that lies strictly below L_j or strictly above $L_{j'}$.*

Proof. We prove that σ does not intersect any line of S that lies strictly below L_j, and the other claim follows by a symmetrical argument. Since L_j is the lowest pseudo-level that σ intersects, σ lies strictly above L_{j-1}. No line of S intersects the sides of the slab between L_j and L_{j-1}. Thus, every line of S whose intersection with the slab is strictly below L_j intersects the slab on or below L_{j-1}, and such a line cannot intersect σ. □

Given a fixed slab U, we define two quantities, which will be useful in the derivation of the lower bound. For each pseudo-level L_j, let α_j denote the number of dual lines of S that lie strictly below L_j, and let β_j denote the number of dual lines that do not lie strictly above L_j. (See Fig. 8(b) for an example.) Note that every line counted by α_j is also counted by β_j, and hence $\alpha_j \leqslant \beta_j$. Also, since no line lies strictly below the lowest level or above the highest level we have $\alpha_1 = 0$ and $\beta_n = n$. Given the sorted sequence of pseudo-level endpoints along each side of U, it is an easy matter to compute these values in $O(n)$ time by a scan of the sorted sequence. This can be done iteratively. Given the values of α_{j-1} and β_{j-1}, the values of α_j and β_j can be computed in $O(1)$ time based on an analysis of the relative slopes of L_{j-1} and L_j and the slopes of the lines of S passing through $e_{j-1}(u)$ and $e_j(u)$, for $u \in \{u_{\min}, u_{\max}\}$.

Consider a bracelet σ lying within any slab U. Consider any dual line $\ell \in S$ that intersects σ, and let L_j and $L_{j'}$ denote the lowest and highest pseudo-levels that σ intersects, respectively. By Lemma 6, ℓ lies neither strictly below L_j nor strictly above $L_{j'}$. This implies that ℓ is not counted by α_j but is counted by $\beta_{j'}$. Therefore, $\beta_{j'} - \alpha_j$ is an upper bound on the number of such lines. Thus we have the following.

Lemma 7. *For any k, consider a k-bracelet σ lying in some slab U, and let L_j and $L_{j'}$ denote the lowest and highest pseudo-levels that σ intersects, respectively. Then $\beta_{j'} - \alpha_j \geqslant k$.*

For example, in Fig. 8(b), L_2 and L_5 are the lowest and highest pseudo-levels intersected by the bracelet σ, respectively. Lemma 7 implies that σ intersects at most $\beta_5 - \alpha_2 = 6 - 0 = 6$ lines. In this case the bound is tight, but by shrinking σ slightly we could have made the actual number of lines smaller without altering the bound.

The key to the branch-and-bound is to use the α and β values to establish a lower bound on the size of any k^+-bracelet contained within the slab. For $1 \leqslant j \leqslant n$, let B_j denote the set of k^+-bracelets in this slab whose lower endpoints lie strictly above L_{j-1}. (Recall that there is a special pseudo-level L_0 at $-\infty$.) Since the lowest pseudo-level that such a bracelet can intersect is L_j, we appeal to Lemma 7, which implies that if $L_{j'}$ is the highest pseudo-level that σ intersects then $\beta_{j'} - \alpha_j \geqslant k^+$. Define a *pseudo-trapezoid* to be the region bounded by the left and right sides of a slab and two (not necessarily consecutive) pseudo-levels. By observing that the union of the B_j's over all possible values of j includes all the k^+-bracelets in this slab, we obtain the following lower bound on the length of any k^+-bracelet.

Lemma 8. *Consider any slab U and any j, where $1 \leqslant j \leqslant n$, such that $\alpha_j \leqslant n - k^+$. Let $j' \leqslant n$ be the smallest index such that $\beta_{j'} - \alpha_j \geqslant k^+$. Let h_j denote the length of the shorter vertical side of the pseudo-trapezoid bounded by L_j*

and $L_{j'}$. Then the smallest k^+-bracelet in B_j has size at least h_j. Furthermore, the smallest k^+-bracelet in U has size at least $h = \min_j h_j$, over all pseudo-levels satisfying these conditions.

Fig. 8(b) illustrates h_2 for the case $k^+ = 6$, which is defined by the pseudo-trapezoid between levels L_2 and L_5. Recall that $\sigma_{\min}$ is the minimum-height sampled bracelet encountered by the algorithm in any of the slabs processed thus far. If the lower bound on the height of the smallest k^+-bracelet in the current slab times $(1 + \varepsilon_r)$ exceeds height$(\sigma_{\min})$, we may infer that no k^+-bracelet of U can be sufficiently small to contradict the approximate optimality of $\sigma_{\min}$ as a solution. Thus, we have the following.

Corollary 1 (*Elimination criterion*). *Given the conditions of Lemma* 8, *if* $(1+\varepsilon_r)h \geqslant height(\sigma_{\min})$, *then this slab may be eliminated from further consideration.*

A slab that is not eliminated is said to be *active*. If all slabs are eliminated by this criterion, then $\sigma_{\min}$ may be reported as the final (approximate) answer. Otherwise, any active slab will be considered a candidate for a refinement process, called *splitting*, which we describe next.

4.3. Overall processing

Slabs that remain active need to be split and reprocessed until their status can be resolved. In theory, slabs could be repeatedly split until no pair of lines of S intersect within the interior of the slab. (In this case the upper- and lower-bounds become tight.) However, processing a slab requires at least $O(n \log n)$ time, even if there are no intersection points in the slab. A more efficient stopping point is when the number of intersection points S lying within the slab is $O(n)$. In this case the minimum-height k^--bracelet can be computed exactly in $O(n \log n)$ time by a straightforward plane-sweep through the slab. Plane-sweep is an algorithmic method that simulates a left-to-right sweep of a vertical line through the arrangement, stopping at each intersection point to update the state of the computation. It is described in its general setting in the book by de Berg et al. (2000). For further information on its application in statistical estimation see Dillencourt et al. (1992), Edelsbrunner and Souvaine (1990), and Rafalin et al. (2002).

In order to apply this test, we first need to count the number of intersections of the dual lines within each active slab. Recall from Section 2 this can be solved in $O(n \log n)$ time by reducing it to the problem of counting the number of inversions in a sequence of integers.

If the number of intersection points in the slab is more than our $O(n)$ threshold, the slab is split by selecting a random intersection point within the slab. Such a random point can be computed as part of the inversion counting process. (See Dillencourt et al., 1992 for details.) Let u' denote the u-coordinate of this intersection point. This point subdivides the slab $U = (u_{\min}, u_{\max}]$ into two sub-slabs, $(u_{\min}, u']$ and $(u', u_{\max}]$.

We can now present the complete algorithm, as the procedure *approxLMS* in Fig. 9. Recall that the input consists of the set of lines S, quantile q and error bounds ε_q and ε_r. The algorithm uses two procedures, *upper-bound* and *lower-bound*, which perform upper- and lower-bound processing, respectively. The algorithm's correctness follows directly from the preceding discussion of this section.

5. Analysis of the slope-decomposition algorithm

In this section we investigate the asymptotic running time of the slope-decomposition algorithm. The processing of each slab is called a *stage*. The processing time for each stage is dominated by the time needed to perform sorting, inversion counting, and applying plane sweep when the slab has $O(n)$ intersection points. It is easy to see that all of these can be performed in $O(n \log n)$ time. In the worst case, none of the slabs are eliminated by the criterion given in Corollary 1, and so the algorithm repeatedly splits each slab until only $O(n)$ intersection points remain. This leads to a total of $O(n)$ expected stages, and hence a total running time of $O(n^2 \log n)$. In Section 6 we provide empirical evidence that, even when run as an exact algorithm, the slope-decomposition algorithm's running time is considerably better than this superquadratic bound would suggest. In this section we will show that, when used as a quantile approximation, the expected-case running time of the slope-decomposition algorithm is $O(n \log^2 n)$ assuming that q and $\varepsilon_q > 0$ are fixed. More precisely, the following is the main result of this section.

approxLMS$(S, q, \varepsilon_r, \varepsilon_q)$:

(1) Let $n \leftarrow |S|$; $q^+ \leftarrow q$; $q^- \leftarrow q^+(1-\varepsilon_q)$; $k^- \leftarrow \lceil nq^- \rceil$; $k^+ \leftarrow \lceil nq^+ \rceil$.
(2) Let $U \leftarrow (-\infty, +\infty]$ be the initial active slab. Initialize height$(\sigma_{\min}) \leftarrow +\infty$.
(3) Apply the following steps as long as there exists an active slab.
   (a) Select any active slab $U = (u_{\min}, u_{\max}]$. Apply inversion counting (as described above) to count the number of intersection points I in the slab.
   (b) If $I \leq cn$ (for some given constant c), apply plane sweep to the slab to determine the smallest k^--bracelet in this slab and declare the slab to be inactive.
   (c) Otherwise, randomly sample an intersection point within the slab. (This can be performed by a simple extension to inversion counting [6].) Let u' denote the u-coordinate of this point. Split the slab U into two sub-slabs $U_1 \leftarrow (u_{\min}, u']$ and $U_2 \leftarrow (u', u_{\max}]$.
   (d) Invoke *upper-bound*(u'). (This may update $\sigma_{\min}$.)
   (e) For $i \in \{1, 2\}$, call *lower-bound*(U_i). If U_i is active on return, add it to the set of active slabs, and otherwise eliminate it.

upper-bound(u') :

(1) Sort the intersection points of the n dual lines of S with the vertical line $u = u'$. Let $\langle e_j(u') \rangle_{j=1}^n$ denote the resulting sequence.
(2) Letting $v(j)$ denote the v-coordinate of $e_j(u')$, let

$$h \leftarrow \min_{1 \leq j \leq n-k^-+1} v(j + k^- - 1) - v(j),$$

If $h <$ height$(\sigma_{\min})$ (we have found a better bracelet), then set $\sigma_{\min}$ to the corresponding bracelet.

lower-bound$(U = (u_{\min}, u_{\max}])$:

(1) Set $\alpha_0 \leftarrow \beta_0 \leftarrow 0$. For j running from 1 to n:
   (a) Pair $e_j(u_{\min})$ with $e_j(u_{\max})$ to form the pseudo-level L_j. (The endpoints were already computed in upper-bound processing.)
   (b) Compute α_j and β_j based on α_{j-1} and β_{j-1} and the slopes of the lines passing through $e_{j-1}(u)$ and $e_j(u)$, for $u \in \{u_{\min}, u_{\max}\}$.
(2) Label U as inactive by default. Initialize $j' \leftarrow 1$. For j running from 1 to n:
   (a) Increment j' until either $j' > n$ or $(\beta_{j'} - \alpha_j) \geq k^+$. If $j' > n$ (no sufficiently short k^+-bracelet exists in the slab), mark U as inactive and exit the loop.
   (b) Let h denote the length of the shorter vertical side of the pseudo-trapezoid bounded between L_j and $L_{j'}$. If

$$(1 + \varepsilon_r)h < \text{height}(\sigma_{\min}),$$

mark U as active and return.

Fig. 9. The hybrid approximation algorithm.

Theorem 2. *Given n points in the plane and constants q and ε_q, the expected case running time of the slope-decomposition algorithm is*

$$O\left(\left(\frac{1}{(q\varepsilon_q)^2} + \log n\right) n \log n\right).$$

The rest of this section is devoted to proving this theorem. It suffices to show that the expected number of stages that are processed by the algorithm is $O(1/(q\varepsilon_q)^2 + \log n)$. We may assume that $\varepsilon_r = 0$ in the analysis, since the algorithm can only terminate sooner as ε_r increases.

In order to perform this analysis, we need to make one additional assumption about the implementation of one step of the algorithm. As described in Fig 9, in Step 3(a) the algorithm selects an arbitrary active slab to be processed. For this analysis, we need to assume that the probability that a slab is chosen is proportional to the number of intersection points of the lines of S that lie within this slab. By doing so, we may assume that the random intersection point sampled from this slab in Step 3(c) of *approxLMS* is in fact a random point sampled from among all the intersection points in all the active slabs. In order to do this, we modify the algorithm so that it weights each slab by the number of intersection points prior to sampling. As presented, the algorithm counts the number of intersection points in a slab only after it has been sampled. It is an easy matter to modify the algorithm so that the inversion counting is performed as soon as each slab is created in Step 3(c).

With this assumption in mind, let us summarize the processing of a single stage. A random intersection point is sampled from the current slab. The shortest k^--bracelet with this u-coordinate is computed in $O(n \log n)$ time, and $\sigma_{\min}$ is updated if necessary. Next the slab is split, and for each of the two slabs, the α and β values are computed. For every pseudo-level L_j, we find the smallest $j' \geqslant j$ such that $\beta_{j'} - \alpha_j \geqslant k^+$. If the shorter side of the pseudo-trapezoid bounded between pseudo-levels $L_{j'}$ and L_j is as large as $\sigma_{\min}$, then this pseudo-trapezoid is marked as inactive. If all pseudo-trapezoids in this slab are inactive, then the entire slab is eliminated.

Our analysis of the algorithm is based on the observation that whenever a slab is processed, one of three things can happen.

(1) The slab is eliminated.
(2) The slab remains active, and $\sigma_{\min}$ is updated (due to finding a smaller bracelet).
(3) The slab remains active, and $\sigma_{\min}$ is not updated.

Let us call the associated stages type-1, type-2, or type-3, depending on which of these three events occurs. First observe that the number of type-1 stages, each of which eliminates a slab, cannot exceed the total number of stages of types 2 and 3, each of which creates one new slab. Thus, it suffices to analyze stages of types 2 and 3. We begin with type-2 stages.

Lemma 9. *The expected number of type-*2 *stages is* $O(\log n)$.

Proof. Consider the line arrangement defined by S. Let each vertex of the arrangement be associated with the smallest bracelet lying on the vertical line passing through this vertex. Let us freeze the algorithm at some point, and let X denote the set of arrangement vertices whose associated smallest bracelets are of height less than the current value of $\sigma_{\min}$. A type-2 stage occurs whenever one of the elements of X is sampled for processing. After the slab processing is complete, $\sigma_{\min}$ decreases, and as a result only those elements of X whose bracelets are smaller than the updated value will remain in X. Because each element of X has an equal probability of being chosen, the expected number of elements of X that remain after processing is half of the number before sampling. Because there are $O(n^2)$ vertices in the arrangement, it follows that the expected number of type-2 stages is $O(\log(n^2)) = O(\log n)$. □

Type-3 stages are more difficult to bound because they do not make tangible progress towards the final solution. We will argue that any slab that is generated from the processing of a type-3 stage contains at least $(nq\varepsilon_q)^2/4$ vertices of the arrangement. By definition, a type-3 slab is neither eliminated nor is it able to update $\sigma_{\min}$. Intuitively, this means that the slab's pseudo-levels do not provide a sufficiently good approximation to the true levels of the arrangement. We shall argue that this only occurs if the slab has a significant number of arrangement vertices.

Lemma 10. *If a slab is created from processing a type-*3 *stage, the number of intersections in the newly created slab is at least* $(nq\varepsilon_q)^2/4 - O(1)$.

Proof. Because a slab generated by a type-3 stage is not eliminated, it must have at least one pseudo-trapezoid that was not declared to be inactive. This means that it contains at least one pseudo-trapezoid lying between two pseudo-levels j' and j, such that $\beta_{j'} - \alpha_j \geqslant k^+$, and that the shorter side of this trapezoid is less than $\sigma_{\min}$. On the other hand, we know that $j' - j < k^-$ since otherwise the presence of a k^--bracelet of height less than $\sigma_{\min}$ would have resulted in $\sigma_{\min}$ being updated, and hence this stage would be of type-2 and not of type-3.

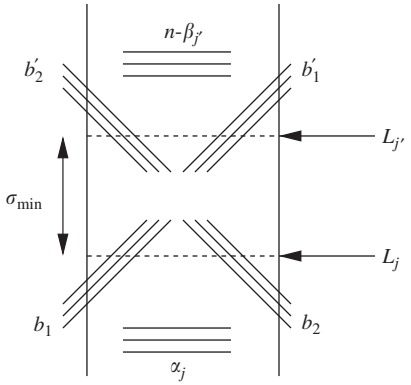

Fig. 10. Counting the lines intersecting L_j and $L_{j'}$.

Each of the n dual lines intersects the slab in a line segment. Let b_1 denote the number of such line segments whose left endpoint is strictly below L_j and whose right endpoint is on or above L_j. Let b_2 be the number of line segments whose right endpoint is strictly below L_j and whose left endpoint is on or above L_j. Define b'_1 and b'_2 analogously using $L_{j'}$. See Fig. 10.

Note that the sets of lines contributing to b_1 and b_2 are disjoint from each other. Some of the lines that contribute to b_1 may contribute to b'_1, and the same holds for b_2 and b'_2 as well. Also observe that every segment of b_1 intersects every segment in b_2 within the slab, and the same holds for b'_1 and b'_2. It follows that the number of intersections within the slab is at least $(b_1b_2 + b'_1b'_2)/2$.

There are j lines of S that are strictly below the left endpoint of L_j, of which α_j have a right endpoint below the right endpoint of L_j and b_1 have a right endpoint on or above L_j, and so $j = \alpha_j + b_1$. Applying the same reasoning to the right side we have

$$b_1 = b_2 = j - \alpha_j.$$

Doing the same for the lines lying strictly above the left endpoint of $L_{j'}$, we have $(n - \beta_{j'}) + b'_2 = n - j'$, and applying this to the right side as well we have

$$b'_1 = b'_2 = \beta_{j'} - j'.$$

Thus, the number of intersections is at least $(b_1^2 + (b'_1)^2)/2$. From the facts that $j' - j \leqslant k^-$ and $\beta_{j'} - \alpha_j \geqslant k^+$ we have

$$\begin{aligned} b_1 + b'_1 &= (j - \alpha_j) + (\beta_{j'} - j') = (\beta_{j'} - \alpha_j) - (j' - j) \\ &\geqslant k^+ - k^- = \lceil nq \rceil - \lceil q(1 - \varepsilon_q)n \rceil \geqslant nq\varepsilon_q - \mathrm{O}(1). \end{aligned}$$

Subject to this constraint, the number of intersections in the slab is minimized when $b_1 = b'_1 = (nq\varepsilon_q - \mathrm{O}(1))/2$, which implies that the number of intersections is at least $(nq\varepsilon_q)^2/4 - \mathrm{O}(1)$. □

To complete the analysis of the number of type-3 stages, we will use this lemma together with the fact that there are only $\mathrm{O}(n^2)$ intersection points altogether to bound the number of such stages. The difficulty in applying this lemma directly is that it is possible that when splitting a type-3 stage, we may induce a very uneven split, in which virtually all of the intersection points fall on one side and very few fall on the other. The lemma tells us that the sub-slab having a sufficiently small number of intersection points cannot again be a type-3 stage, but this does not say anything about the larger sub-slab.

To address this issue we say that a type-3 stage is *balanced* if after processing, both of the two sub-slabs produced by the split contain no more than a fraction of $\frac{3}{4}$ of the original number of intersection points. Observe that a stage is unbalanced only if we sample an intersection point from the first or last quarter of the u-coordinates of the slab. Because of our general position assumption and the fact that sampling is random, the probability of a balanced type-3

stage is roughly $\frac{1}{2}$. Thus, it suffices to count just the expected number of balanced type-3 stages, since the expected number of unbalanced stages will not be larger.

Lemma 11. *The number of balanced type-3 stages is* $O(1/(q\varepsilon_q)^2)$.

Proof. It will simplify things to assume that no type-2 stages occur (as they have already been analyzed). Whenever a balanced type-3 stage is processed, we know from Lemma 10 that the algorithm has partitioned a slab containing at least $(nq\varepsilon_q)^2/4$ intersection points into two sub-slabs, each containing at least one quarter the original number of intersection points. In the worst case, no slabs are eliminated, and this splitting process continues recursively until every slab has fewer than $(nq\varepsilon_q)^2/4$ intersection points. Since such a slab resulted from a balanced split, it has at least $(nq\varepsilon_q)^2/16$ intersection points. Since there are $\binom{n}{2}$ total intersection points in the arrangement, the number of such minimal slabs is at most

$$\frac{\binom{n}{2}}{(nq\varepsilon_q)^2/16} \leqslant \frac{8}{(q\varepsilon_q)^2}.$$

Since each balanced type-3 stage increases the number of slabs by one, this also bounds the number of type-3 stages. □

Combining the previous two lemmas establishes Theorem 2.

6. Experimental results

We implemented the slope-decomposition algorithm described in Section 4 in C++, which we call ALMS, for *Approximate LMS*. In order to establish the algorithm's practical value, we tested this implementation on a number of point distributions. Our principal goal is determining the running time and accuracy of the algorithm. So rather than selecting points from some application, we chose instead to perform experiments on synthetically generated data sets, which we felt reflected challenging cases for the algorithm and in which we could control the various parameters such as the input size, the distribution of inliers and outliers, and so on. In each case, we sampled points from two distributions, one for inliers and one for outliers. In all cases points were sampled from a square of side length 2 centered at the origin. Throughout the constant c in Fig. 9 was set to 10, thus a slab having fewer than $10n$ intersection points is resolved by plane sweep.

For each distribution, an *inlier probability* p_i is specified. With this probability, a point is sampled from an inlier distribution, and otherwise (with probability $p_o = 1 - p_i$) it is sampled from an outlier distribution. The same inlier distribution was used in all cases and the inlier probability was fixed at $p_i = 0.30$. We fixed the LMS quantile q to 0.25 throughout, so it is quite likely that sufficiently many inliers lie near the line of fit. All points were constrained to lie within a *bounding square* of side length 2 centered at the origin.

Inliers were generated as follows. A line equation was first generated by sampling a slope uniformly from the interval $[-1, 1]$ and a y-intercept uniformly from the interval $[-0.25, 0.25]$. Points were generated uniformly along the intersection of this line and the bounding square. A Gaussian distributed *residual noise* was then added to the y-coordinate of each point, with mean 0.0 and a given standard deviation. We used a *residual standard deviation* of 0.01, unless otherwise noted. (This choice was based on our interest in applications in computer vision, where inliers are typically perturbed by a single pixel due to digitization errors.) Outliers were sampled from one of a number of different distributions, as listed below. These are illustrated in Fig. 11.

line + unif: Outliers are sampled uniformly from the bounding square. See Fig. 11(a).

line + half-unif: In many applications outliers are distributed asymmetrically with respect to the line of fit. To model this, we generated points as in the *line + unif* distribution, but limited outliers to lie above the line. See Fig. 11(b).

line + segments: Often outliers lie on multiple line segments. To model this we generated outliers on a collection of line segments. A fixed number of segments is given (we used 10 throughout). The slope of each segment was generated uniformly in the interval $[-1, 1]$. The length of each segment was drawn from a Gaussian distribution with mean 1 and

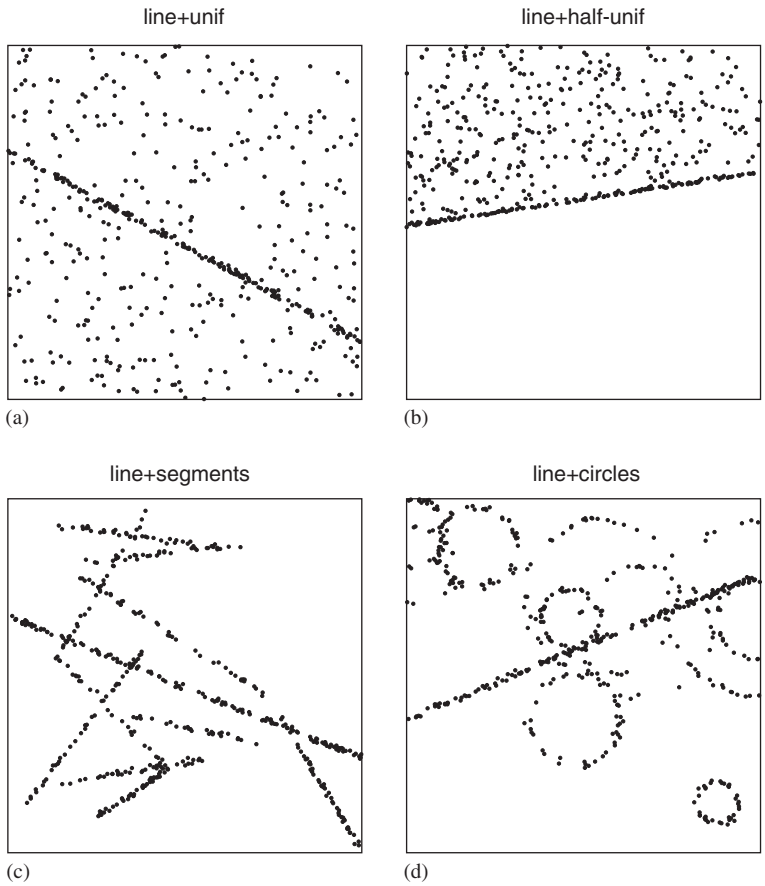

Fig. 11. Examples of point distributions. Each example contains 500 points in a square of side length 2. There are roughly 30% inliers and 70% outliers. The inliers are perturbed by Gaussian distributed residual noise with a standard deviation of 0.01. In (c) and (d) there are 10 clusters of line segments and circles, respectively.

standard deviation 0.25. Each segment was centered uniformly within the bounding square, subject to the constraint that it lies entirely within the square. To generate each outlier point, a segment is chosen at random, a point is generated uniformly at random along the segment, and a Gaussian residual added with the given residual standard deviation. See Fig. 11(c).

Since roughly 70% of the points are outliers, and we distribute them randomly among 10 segments, we expect that only 7% of the points will fall on any one of the segments. Since we use a quantile of 0.25, it is quite unlikely that the estimator will fail to select the inlier line.

line + circles: To model outliers that lie on nonlinear curves, we generated outliers that lie on circular arcs. A fixed number of circles is given. (We used 10 throughout.) The centers of the circles are sampled uniformly from within the bounding square. For each circle a random radius is sampled from a Gaussian distribution with mean 0.30 and standard deviation 0.10. To generate each outlying point, a circle is chosen at random, and a random point is sampled uniformly along the portion of the circle that lies within the bounding square. The point is perturbed normal to the circle by a Gaussian residual with the given residual standard deviation. See Fig. 11(d).

We ran a number of experiments with the ALMS program and for comparison we also implemented the best known exact algorithm based on topological plane sweep, due to Edelsbrunner and Souvaine (1990). (We also implemented a classical plane-sweep algorithm, which uses a heap to order events, but we used the topological plane sweep because it was the faster of the two.) We ran four different experiments to demonstrate various aspects of the ALMS program's performance.

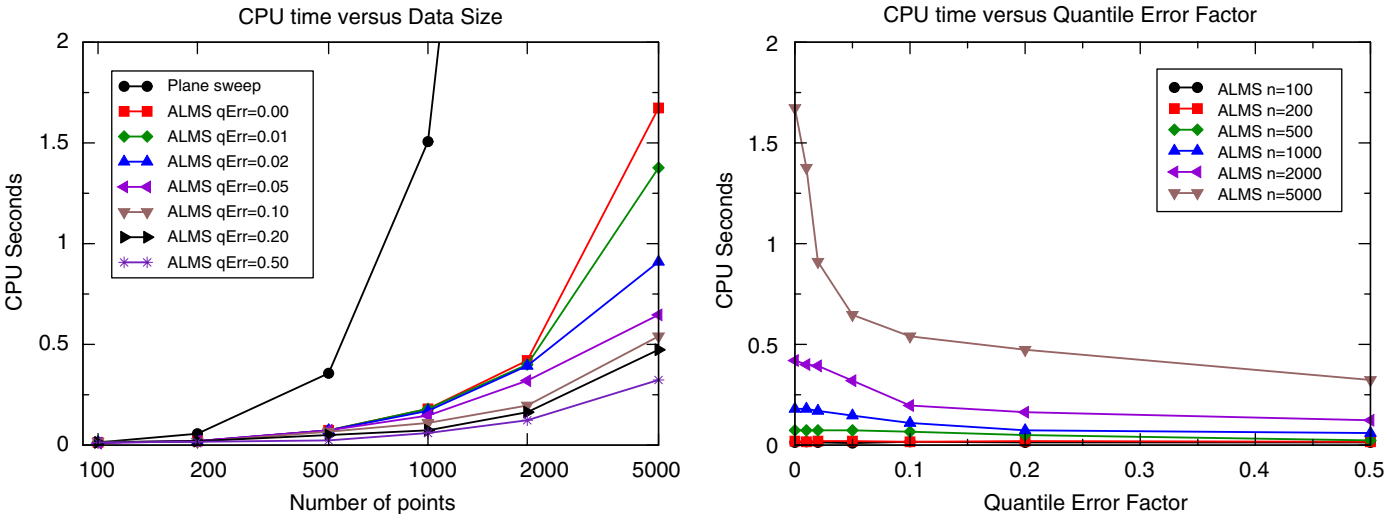

Fig. 12. Running time versus data size and quantile error factor.

6.1. Input size and quantile error factor

In order to investigate the performance of ALMS as a quantile approximation algorithm, we compared its running time as a function of both data size and the quantile error factor to the topological plane-sweep algorithm. We considered point sets from the *line + unif* distribution with $p_i = 0.30$ and used a residual standard deviation of 0.01. We generated point sets of sizes ranging from 100 to 5000 and varied the quantile error ε_q, from 0.00 to 0.50. (For example, since $q = 0.25$, in the case $\varepsilon_q = 0.50$, the LMS strip need only enclose 12.5% of the points.) In all cases the residual error ε_r was set to 0. We also ran the exact algorithm, based on topological plane sweep. The results are shown in Fig. 12. Each point is the average of three runs on different data sets.

The running times of the plane-sweep algorithm were so large that they did not fit on this graph. For example, for $n = 5000$, the average CPU time for the plane-sweep algorithm was roughly 37 s, in contrast to the ALMS whose running time was under 2 s. Observe that as ε_q increases the running time decreases rapidly.

Recall from Theorem 2 that we bounded the expected number of stages of the slope-decomposition algorithm by $1/(q\varepsilon_q)^2 + \log n$. To see whether our experiment bears out this relationship, we repeated the above experiments. Throughout we used the *line + unif* distribution, with 30% inliers, and set $q = 0.25$. In the first case we fixed $\varepsilon_q = 0.01$ and varied the data size from 100 up to 40,000. We recorded the number of stages that the algorithm ran. The results are shown in Fig. 13(a). Note that the x-axis is on a logarithmic scale. This plot suggests that the number of stages grows linearly or slightly sublinearly as a function of $\log n$, which matches the analysis. Next, we fixed $n = 5000$ and varied the quantile error ε_q from 0.001 up to 0.5 and recorded the number of stages. The results are shown in Fig. 13(b). In all cases we averaged the results over 12 runs.

Although the analysis would suggest a rapid increase in the number of stages for small ε_q, the plot suggests that the performance is much better. In fact, for comparison, we plotted the function $10(1 + \log_{10}(1/\varepsilon_q))$ with a broken line, suggesting that, at least for this example, the number of stages grows no faster than $O(\log(1/\varepsilon_q))$. This discrepancy is not altogether surprising. The upper bound of $1/(q\varepsilon_q)^2$ is based on a worst-case assumption about the behavior of lines within the slab. As is common in algorithms that involve hierarchical search, the actual behavior of the algorithm on a given input instance depends on subtle and complex combinatorial properties of the input, which are very hard to characterize mathematically.

It is interesting to note that ALMS is significantly faster than the plane-sweep algorithm even when it is run as an exact algorithm, that is, with $\varepsilon_q = \varepsilon_r = 0$. One reason for this is that, because a sufficiently large number of points lie close to the line of fit, the branch-and-bound search used by ALMS can quickly eliminate many unpromising regions of the slope space, and concentrate on the few slope intervals of interest. Indeed an inspection of the number of plane sweeps reveals that (in the exact case) ALMS only needed to perform plane sweep on from 4 to 8 slabs for each run. Since each

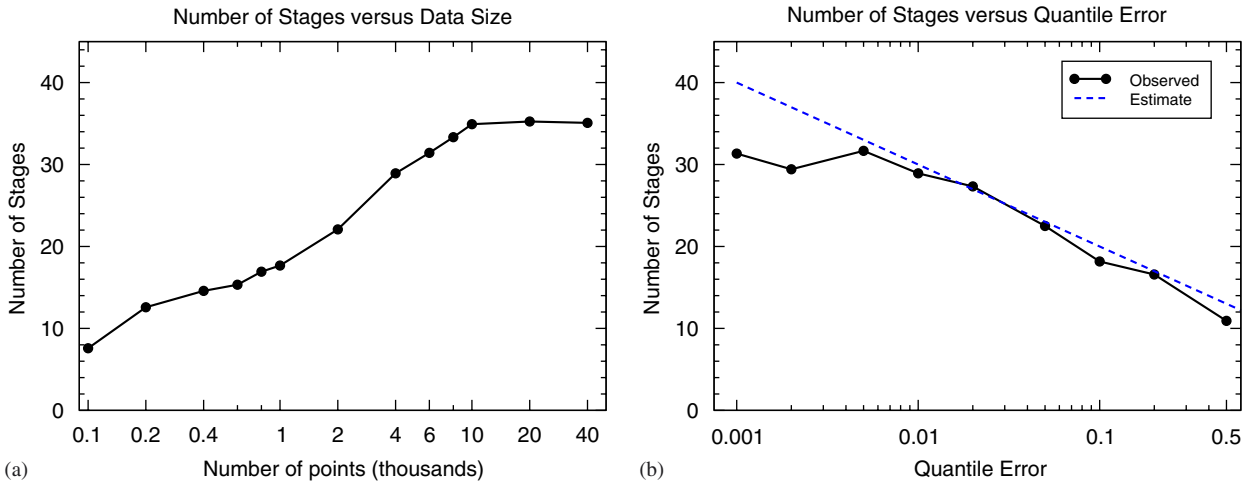

Fig. 13. Number of stages versus: (a) data size and (b) quantile error factor. In the second case we plot the function $10(1 + \log_{10}(1/\varepsilon_q))$ for comparison.

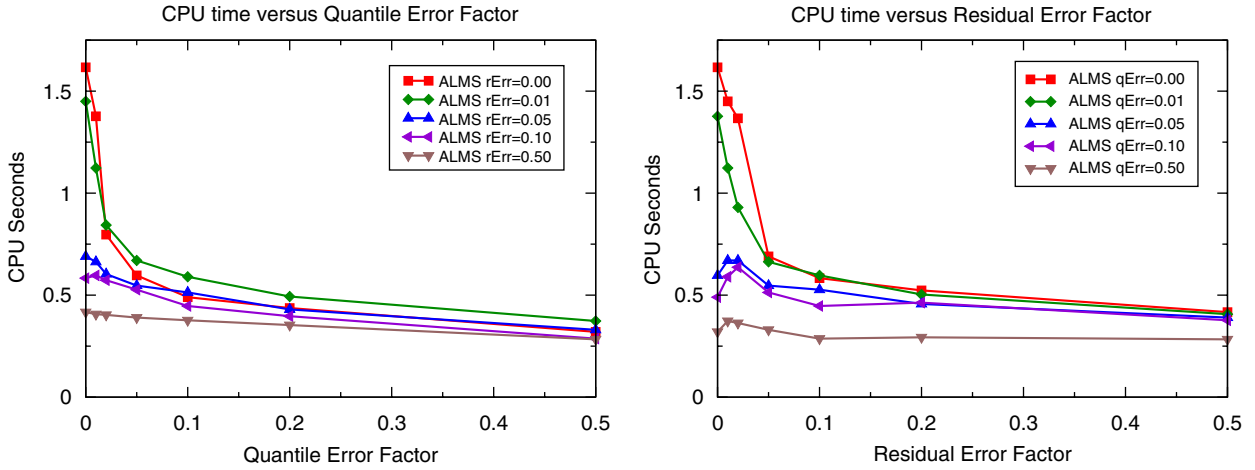

Fig. 14. Running time versus quantile and residual error factors.

slab contains at most $10n$ intersection points, this represents a significant reduction over the $\binom{n}{2}$ intersections points that would otherwise need to be considered. For example, for $n = 5000$ points, we observed speedups over topological plane sweep on the order of 20. As ε_q and ε_r increase, the speedups are even more dramatic.

6.2. *Quantile error and residual error factors*

In this experiment we considered the running time of the algorithm as a function of both the quantile error factor ε_q and residual error factor ε_r. We used $n = 5000$ points generated using the *line + unif* distribution with an inlier residual standard deviation of 0.01. Each point is the average of three runs on different input sets. The results are shown in Fig. 14. It is notable that the variations in running times are quite similar for both approximation factors. As expected, the running times decrease significantly as either approximation factor is increased. An interesting phenomenon seen in the figure is that for relatively large ε_q and small ε_r the running time actually increases slightly as ε_r increases. This is probably due to the algorithm eliminating the slab that contains the optimum LMS bracelet relatively early in the processing, which in turn hampers the algorithm's ability to eliminate slabs in the later part of the processing.

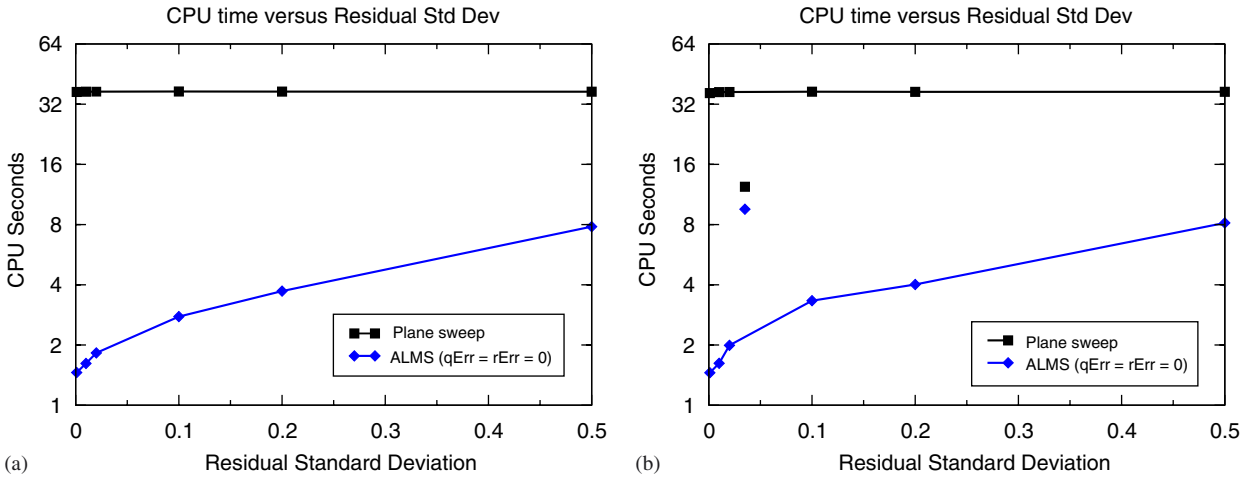

Fig. 15. Running time versus residual standard deviation for the topological plane-sweep algorithm (exact) and ($\varepsilon_q = \varepsilon_r = 0$) for the: (a) *line* + *unif*; and (b) *line* + *segments* distributions.

6.3. Inlier noise

As mentioned above, the ALMS algorithm can be quite efficient, even when it is run as an exact algorithm, that is, when $\varepsilon_q = \varepsilon_r = 0$. We hypothesized above that this is because a sufficient number of points are close enough to the line of fit that most of the slope intervals can be eliminated by the branch-and-bound process.

This raises the question of how sensitive the algorithm is to points lying close to the line of fit. In this experiment we generated $n = 5000$ points using both the *line* + *unif* and the *line* + *segments* distributions. We kept the inlier probability fixed at 0.30 and varied the inlier standard deviation from 0 to 0.5. Note that since the points lie in a square of side-length 2 a standard deviation of 0.50 represents a strip of roughly half the width of the square.

The results are shown in Fig. 15. The algorithm's performance was quite similar for both distributions. Each point on the graph is the average of three runs on different input sets. As expected, the running time of ALMS increases with the residual standard deviation, because of the greater variation of slopes among lines that are determined by pairs of inliers. It is notable that, even with a relatively large standard deviation of 0.50, the running time of ALMS is still considerably faster than that of topological plane-sweep. In fact, we ran an experiment in which we attempted to fit a line to a set of points distributed uniformly in the square, that is, having only outliers. Even in this extreme case, ALMS ran faster than the plane-sweep algorithm by a factor of roughly 2, both in terms of CPU time and the number of intersection points processed by plane sweep.

6.4. Different distributions and actual error

In all the prior experiments we considered mostly the *line*+*unif* distribution. In this experiment we considered all the distributions. First, we compared running times of the ALMS algorithm against those of the topological plane-sweep algorithm. We ran each experiment for $n = 5000$ points, inlier residual standard deviation of 0.01, with residual error factor ε_r ranging from 0.0 to 0.5 and with a fixed quantile error factor $\varepsilon_q = 0$. We averaged the results over three trials in each case. We report the CPU times for the two algorithms and their *speed-up*, that is, the ratio of the two CPU times. The speed-ups are illustrated in Fig. 16, and show that ALMS runs from 20 to 90 times faster than topological plane sweep over all the distributions. As expected, the speed-ups increase as ε_r increases. All the running times are shown in Table 1. Observe that for all distributions the speed-up of ALMS over the plane-sweep algorithm is considerable, even when $\varepsilon_r = 0$. Further, the speed-ups are comparable for all the distributions.

In each case we also computed the actual relative residual error committed by the algorithm, defined as follows. If we let r^* denote the width of the exact LMS strip (computed by the plane-sweep algorithm) and let r denote the

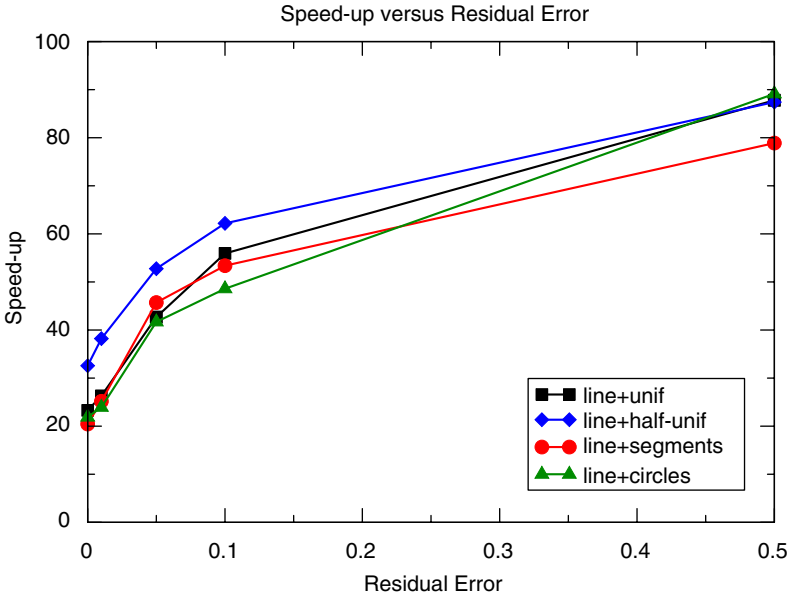

Fig. 16. CPU time speed-up of ALMS over topological plane sweep versus residual error ε_r for various distributions.

Table 1
Running time and actual error of the topological plane-sweep and ALMS algorithms for $n = 5000$ points, various distributions and various values of ε_r

Distribution	ε_r	CPU time			Actual error
		Sweep	ALMS	Speed-up	
line + unif	0.00	37.68	1.54	24.47	0.0000
	0.01	37.64	1.37	27.41	0.0000
	0.05	37.62	0.84	44.79	0.0020
	0.10	37.63	0.63	59.72	0.0030
	0.50	37.64	0.41	91.81	0.0133
line + half-unif	0.00	37.39	1.11	33.68	0.0000
	0.01	37.55	0.93	40.37	0.0000
	0.05	37.46	0.68	55.08	0.0101
	0.10	37.41	0.58	64.14	0.0145
	0.50	37.45	0.40	92.87	0.0469
line + segments	0.00	37.71	1.80	20.91	0.0000
	0.01	37.52	1.47	25.47	0.0000
	0.05	37.48	0.80	47.04	0.0040
	0.10	37.50	0.69	54.61	0.0077
	0.50	37.50	0.46	81.52	0.0085
line + circles	0.00	37.52	1.70	22.07	0.0000
	0.01	37.51	1.53	24.52	0.0000
	0.05	37.47	0.87	43.07	0.0014
	0.10	37.58	0.74	50.55	0.0030
	0.50	37.54	0.41	90.83	0.0126

The speed-up is the ratio of the two running times. In all cases $\varepsilon_q = 0$.

width reported by the ALMS algorithm then we define the *actual residual error* to be

$$\varepsilon_{\mathrm{a}} = \frac{r - r^*}{r^*}.$$

Since $r^* \leqslant r \leqslant (1+\varepsilon_r)r^*$, it follows that $0 \leqslant \varepsilon_a \leqslant \varepsilon_r$. It is interesting to note that the actual errors committed by the ALMS algorithm are typically smaller than ε_r by an order of magnitude. (See Table 1.)

7. Concluding remarks

We have presented two algorithms for the least median-of-squares (LMS) regression line estimator in the plane. The first is a theoretically efficient randomized quantile approximation, which runs in $O(n \log n)$ time assuming that the quantile q and the quantile error ε_q are fixed constants. The second is a practical randomized approach based on a branch-and-bound search of dual space, called slope decomposition. When run as a quantile approximation, its expected running time is $O(n \log^2 n)$ time. Both algorithms are Las Vegas, meaning that their running time may be affected by the randomization, but not their correctness. We implemented the latter algorithm, called ALMS, and presented empirical evidence that it is quite practical, and runs significantly faster than the best known exact algorithm based on topological plane sweep. (The source code is available by request from the first author of this paper.) We have also described a number of computational techniques that may be of general interest in other problems in computational statistics.

The ALMS algorithm has an important practical feature. Unlike topological plane sweep, which is asymptotically efficient in the worst case, but whose running time is largely invariant of the input distribution, ALMS takes advantage of the presence of many points that lie on or close to a line. Although this assumption may not hold in the worst case, it is quite a reasonable assumption for many instances in which line fitting is performed.

The most obvious gap in our understanding of the slope-decomposition algorithm is why it seems to perform as well as it does, even when run as an exact algorithm and on data sets in which points do not lie close to a line. Clearly the branch-and-bound search manages to eliminate large regions of the dual line arrangement with relatively little work. The recent lower bound results of Erickson et al. (2004) suggest that such a data-sensitive approach may be necessary to achieve efficiency in practice.

A natural and important question is whether the branch-and-bound approach can be generalized to compute the LMS hyperplane in higher dimensions. A number of elements of the slope-decomposition algorithm are particular to the plane. In particular, this includes the concepts of decomposing dual space into a collection of intervals according to the slope of lines and using inversion counting within each vertical slab to count and sample intersection points efficiently.

References

Agulló, J., 1997. Exact algorithms for computing the least median of squares estimate in multiple linear regression algorithm. In: Dodge, Y. (Ed.), L_1-Statistical Procedures and Related Topics, IMS Lecture Notes Monograph Series. Institute of Mathematical Statistics, vol. 31, pp. 133–146.

Chernoff, H., 1952. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann. Math. Statist. 23, 493–507.

Cole, R., Salowe, J., Steiger, W., Szemerédi, E., 1989. An optimal-time algorithm for slope selection. SIAM J. Comput. 18 (4), 792–810.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., 1990. Introduction to Algorithms. MIT Press, Cambridge, MA.

de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., 2000. Computational Geometry: Algorithms and Applications. second ed. Springer, Berlin.

Dillencourt, M.B., Mount, D.M., Netanyahu, N.S., 1992. A randomized algorithm for slope selection. Internat. J. Comput. Geom. Appl. 2, 1–27.

Edelsbrunner, H., 1987. Algorithms in Combinatorial Geometry. EATCS Monographs on Theoretical Computer Science, vol. 10. Springer, Heidelberg, West Germany.

Edelsbrunner, H., Guibas, L.J., 1989. Topologically sweeping an arrangement. J. Comput. System Sci. 38, 165–194 (Corrigendum in 42 (1991) 249–251.).

Edelsbrunner, H., Mücke, E.P., 1990. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Trans. Graph. 9, 66–104.

Edelsbrunner, H., Souvaine, D.L., 1990. Computing median-of-squares regression lines and guided topological sweep. J. Amer. Statist. Assoc. 85, 115–119.

Erickson, J., Har-Peled, S., Mount, D.M., 2004. On the least median square problem. In: Proceedings of the 20th Annual ACM Symposium on Computational Geometry, pp. 273–279; Discrete Comput. Geom., to appear.

Hagerup, T., Rüb, C., 1990. A guided tour of Chernoff bounds. Inform. Process. Lett. 33, 305–308.

Har-Peled, S., 2001. A practical approach for computing the diameter of a point-set. In: Proceedings of the 17th Annual ACM Symposium on Computational Geometry, pp. 177–186.

Har-Peled, S., Wang, Y., 2003. Shape fitting with outliers. In: Proceedings of the 19th Annual ACM Symposium on Computational Geometry, pp. 29–38.

Hawkins, D.M., 1993. The feasible set algorithm for least median of squares regression. Comput. Statist. Data Anal. 16, 81–101.

Langerman, S., Steiger, W., 2000. An optimal algorithm for hyperplane depth in the plane. In: Proceedings of the 11th Annual ACM–SIAM Symposium on Discrete Algorithms, pp. 54–59.

Matoušek, J., 1991a. Cutting hyperplane arrangements. Discrete Comput. Geom. 6, 385–406.
Matoušek, J., 1991b. Randomized optimal algorithm for slope selection. Inform. Process. Lett. 39, 183–187.
Matoušek, J., 1992. Efficient partition trees. Discrete Comput. Geom. 8, 315–334.
Matoušek, J., Mount, D.M., Netanyahu, N.S., 1998. Efficient randomized algorithms for the repeated median line estimator. Algorithmica 20, 136–150.
Meer, P., Mintz, D., Rosenfeld, A., Kim, D.Y., 1991. Robust regression methods for computer vision—a review. Internat. J. Comput. Vision 6, 59–70.
Motwani, R., Raghavan, P., 1995. Randomized Algorithms. Cambridge University Press, Cambridge.
Mount, D.M., Netanyahu, N.S., Piatko, C., Silverman, R., Wu, A.Y., 2000. Quantile approximation for robust statistical estimation and k-enclosing problems. Internat. J. Comput. Geom. Appl. 10, 593–608.
Mount, D.M., Netanyahu, N.S., Zuck, E., 2004. Analyzing the number of samples required for an approximate Monte-Carlo LMS line estimator. In: Hubert, M., Pison, G., Struyf, A., Van Aelst, S. (Eds.), Theory and Applications of Recent Robust Methods, Statistics for Industry and Technology. Birkhauser, Basel, pp. 207–219.
Mulmuley, K., 1994. Computational Geometry: An Introduction Through Randomized Algorithms. Prentice-Hall, Englewood Cliffs, NJ.
Netanyahu, N.S., Philomin, V., Rosenfeld, A., Stromberg, A.J., 1997. Robust detection of road segments in noisy aerial images. Pattern Recogn. 30, 1673–1686.
Olson, C.F., 1997. An approximation algorithm for least median of squares regression. Inform. Process. Lett. 63, 237–241.
Rafalin, E., Souvaine, D., Streinu, I., 2002. Topological sweep in degenerate cases. In: Proceedings of the Fourth International Workshop on Algorithm Engineering and Experiments, Lecture Notes in Computer Science, vol. 2409. Springer, Berlin, Germany, pp. 155–156.
Rousseeuw, P.J., 1984. Least median-of-squares regression. J. Amer. Statist. Assoc. 79, 871–880.
Rousseeuw, P.J., Hubert, M., 1977. Recent developments in PROGRESS. In: L_1-Statistical Procedures and Related Topics. IMS Lecture Notes Monograph Series, vol. 31. Institute of Mathematical Statistics.pp. 201–214.
Rousseeuw, P.J., Leroy, A.M., 1987. Robust Regression and Outlier Detection. Wiley, New York.
Samet, H., 1990. The Design and Analysis of Spatial Data Structures. Addison-Wesley, Reading, MA.
Souvaine, D.L., Steele, J.M., 1987. Time- and space- efficient algorithms for least median of squares regression. J. Amer. Statist. Assoc. 82, 794–801.
Stein, A., Werman, M., 1992. Robust statistics in shape fitting. In: Proceedings of the IEEE Computer Society Conference Computer on Vision and Pattern Recognition.Champaign, IL. June.
Stewart, C.V., 1996. MINPRAN: a new robust estimator for computer vision. IEEE Trans. Pattern Anal. Mach. Intell. 17, 925–938.
Stromberg, A.J., 1993. Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression. SIAM J. Sci. Comput. 14 (6), 1289–1299.
Yohai, V.J., 1987. High breakdown-point and high efficiency robust estimates for regression. Ann. Statist. 15, 642–656.