Efficient Robust Digital Hyperplane Fitting with Bounded Error

Dror Aiger, Yukiko Kenmochi, Hugues Talbot, and Lilian Buzer

Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, France

Abstract. We consider the following fitting problem: given an arbitrary set of N points in a bounded grid in dimension d, find a digital hyperplane that contains the largest possible number of points. We first observe that the problem is 3SUM-hard in the plane, so that it probably cannot be solved exactly with computational complexity better than $O(N^2)$, and it is conjectured that optimal computational complexity in dimension d is in fact $O(N^d)$. We therefore propose two approximation methods featuring linear time complexity. As the latter one is easily implemented, we present experimental results that show the runtime in practice.

Keywords: fitting, digital hyperplane, approximation, linear programming, randomization.

1 Introduction

This article is about efficient and effective hyperplane fitting in the presence of outliers, formulated in the discrete setting. This is a well-known problem with many applications in computer vision and image analysis. In the following we consider an arbitrary set of pixels S in discrete space $\mathbb{Z}^d$, (with typically $d = 2$ or 3), and ask the question of what is the closest co-dimension 1 hyperplane (*i.e.* a line in 2D or a plane in 3D) that best fits this set of pixels. Clearly this problem depends on the notion of fitting and the definition of hyperplane.

Irrespective of the formulation, recent applications of this problem include shape approximation [3,19], image registration [18,20] and image segmentation [12,14]. More generally, the problem of robust parameter estimation from noisy data is also closely related [10].

The problem is connected to the problem of robust regression, for instance using least squares, weighted least-squares, least-absolute-value regression and least median of squares (LMS) [4,15,17]. In this setting, an hyperplane $\mathbf{H}$ of co-dimension 1 corresponds to the following continuous model:

$$\mathbf{H} = \{(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d : a_1x_1 + a_2x_2 + \ldots + a_dx_d + a_{d+1} = 0\}, \quad (1)$$

with the $a_i \in \mathbb{R}$. Note it is common to add an additional normalization constraint such as $\sum_{i=1}^{d} |a_i| = 1$ or $\sum_{i=1}^{d} a_i^2 = 1$. Such a formulation corresponds to minimizing various cost functions. For instance least-square fitting, which has a closed-form solution, minimizes the geometric distance from all the given points to the model, whereas least-absolute-value regression optimizes the ℓ_1 distance instead. Both models are very efficient but not very robust to outliers. In contrast, LMS minimizes the median of either the geometric or ℓ_1 distance to the model, and is robust to the presence of outliers, as long as they do not represent more than half of the dataset. There exists polynomial, optimal algorithms for LMS, but their space and computational complexity grow at least as quickly as $O(N^d)$, and there does not seem to exist dependable implementations for $d \geq 3$.

1.1 Approximate Line and Plane Fitting

Well-known and most used methods for hyperplane fitting include the Hough Transform (HT) [6,11], RANSAC [7] and associated variations [5]. HT uses an accumulative approach in the discretized dual space and an ad-hoc detection of high accumulated values. The computational complexity of traditional HT is $O(N\delta^{d-1})$, where N is the number of given points and δ is the (discrete) size of image sides. This is linear for a fixed δ and a fixed d.

RANSAC and its variation consider random d-tuples of points (*i.e.* pairs and triplets respectively for 2D or 3D) within the pixel set S, forming respectively a candidate hyperplane, and compute a distance from S to the candidate hyperplane. Many distances can be considered, including robust versions that may or may not be true distances. A distance or score is associated with every candidate hyperplane. After a number of candidate trials that depend on the size of S, the best-fitting hyperplane featuring the minimum distance or score is given as the output. The computational complexity of RANSAC is linear in the size N of S if the number of inliers is a constant fraction of N.

Both HT and RANSAC are efficient, linear-time complexity algorithms. However neither can claim to find a global or even error-bounded approximate solution to an associated optimization problem.

1.2 Discrete, Optimal Formulation

Because the problem is inherently discrete, it is useful to consider a purely discrete formulation of the problem.

Recently, such a discrete, optimization-based framework for this problem was proposed in [24] for 2D lines and 3D planes. This approach consists of considering a discrete line or plane using Reveillès definition [16], *i.e.* a set of grid points between a pair of continuous-domain lines or planes with rational coefficients separated by a distance w. For a given pixel set S of size N and a given w, an optimal hyperplane is one for which a maximal number of pixels lie between the two continuous hyperplanes. Even though the pair of hyperplanes form a convex set, the problem is combinatorial in nature, and a non-polynomial, branch-and-bound approach was initially suggested to find the optimal solution. This was later solved with an $O(N^d \log N)$ algorithm for $d = 2, 3$ in [21,22,23], and improved in [13] with an $O(N^2)$ solution in 2D using a topological sweep method. While a polynomial solution of degree equal to the dimension of the problem is useful, it is still too inefficient for many application. Typically, the problem is currently solvable for $N = 10^3$ but impractical for $N = 10^6$ in 3D.

In the rest of the paper we first show that the problem is 3SUM-hard in 2D, *i.e.* that a computational complexity of $O(N^2)$ is likely the best that can be expected for 2D, as well as $O(N^d)$ for higher dimension d. This motivated us to pursue an approximate solution with known bounded error, with linear-time complexity. We present two algorithms that give us different bounded errors; one is the number of inliers, and another is the width of digital hyperplanes. Since algorithms based on HT and RANSAC provide no error bound, and no guarantee of optimality of any kind, this render our solution competitive with RANSAC and HT, while insuring a good quality solution. We demonstrate this on a pair of artificial and real examples.

2 The Problem of Digital Hyperplane Fitting

2.1 Definition

An hyperplane in Euclidean space $\mathbb{R}^d$, $d \geq 2$, is defined by (1), and in this paper we use the following normalization constraint instead of the above more conventional ones: $-1 \leq a_i \leq 1$ for $i = 1, 2, \ldots, d$ such that there exists at least one a_i that equals to 1. Note that this normalization technique enables us to bound the range of every coefficient between -1 and 1, except for a_{d+1}: practically a_{d+1} is also bounded by the size of an input image.

A digital hyperplane, which is the digitization of a hyperplane $\mathbf{H}$ in the discrete space $\mathbb{Z}^d$, is then defined by the set of discrete points satisfying two inequalities:

$$\mathbf{D}(\mathbf{H}) = \{(x_1, x_2, \ldots, x_d) \in \mathbb{Z}^d : 0 \leq a_1 x_1 + a_2 x_2 + \ldots + a_d x_d + a_{d+1} < w\} \quad (2)$$

where w is a given constant. From the digital geometrical viewpoint, if we set $w = 1$, then the definition is equivalent to that of naive hyperplanes [16], where parameters can be rational numbers. As mentioned above, at least one coefficient a_i equals 1 among a_i for $i = 1, 2 \ldots, d$, so hereafter we set $a_d = 1$ for simplicity. In other words, we mainly deal with the following linear inequalities

$$0 \leq a_1 x_1 + a_2 x_2 + \ldots + a_{d-1} x_{d-1} + x_d + a_{d+1} < w \quad (3)$$

instead of the ones in (2). Note that practically we use d different types of the inequalities for representing digital hyperplanes depending on d settings of $a_i = 1$ for all $i = 1, \ldots, d$.

2.2 Problem Formulation

Given a set S of discrete points coming from the $[0, \delta]^d$ grid (*i.e.* the hypercubic grid with a distance between two neighbours $= 1$), the problem is to find a digital hyperplane that contains a maximum number of points, called an optimal digital hyperplane. Discrete points that are contained in the fitted digital hyperplane, are called inliers; the complement points are called outliers. Our problem is then equivalent to finding a digital hyperplane such that the number of inliers be maximum.

We need the following definition for geometric understanding:

Definition 1. *A* slab *is the region on and in between two parallel hyperplanes in d-space. An ω-slab is a slab of size ω, meaning that the distance between the two hyperplanes is ω.*

If the distance ω is taken in the x_d-axis direction in the space $(x_1, x_2, \ldots, x_d)$ and $\omega = w$, then $\mathbf{D}(\mathbf{H})$ can be geometrically interpreted as a w-slab. Thus, finding the optimal digital hyperplane for a given S is equivalent to finding a w-slab that contains the maximum number of points in S. Using standard geometric duality induced by (3), the problem of finding this w-slab is equivalent to finding a point that is covered by a maximum number of w-slabs in the dual space as follows: every point $\boldsymbol{p}$ in the d-dimensional primal space $(x_1, x_2, \ldots, x_d)$ is mapped to a hyperplane $\mathbf{H}$ in the d-dimensional dual space $(a_1, a_2, \ldots, a_{d-1}, a_{d+1})$ and the set of all hyperplanes in the primal that have a distance (in the x_d-axis direction) smaller than w to $\boldsymbol{p}$ are mapped to the set of points in the dual contained in a w-slab (distance w in the a_{d+1}-axis direction) one of whose sides is equal to $\mathbf{H}$. Details for the $d = 2$ case can be found in [13].

In the following sections, the problem of digital hyperplane fitting will be considered either in the primal space (Sections 3 and 5) or in the dual space (Section 4).

3 Theoretical Observation on Exact Fitting

We consider the $[0, \delta]^d$ grid, and a set S of N discrete points is given. As our input is a binary image, N is necessarily smaller than δ^d. We show in this section that, for any N that is smaller than δ^d, the exact solution of the fitting problem is as hard to obtain as that of $O(N^d)$ problems; for $d = 2$, we call such a class of problems the 3SUM problem. For the sake of simplicity, we start our discussion in the 2D plane.

3SUM is the following computational problem, introduced by Gajentaan and Overmars [9] and conjectured to require roughly quadratic time complexity: given a set T of n integers, are there elements a, b, c in T such that $a + b + c = 0$? A problem is called 3SUM-hard if solving it in subquadratic time implies a subquadratic time algorithm for 3SUM.

Observation 1. *The problem of digital line fitting is 3SUM-hard.*

Proof. The problem of finding three colinear points among a given set of discrete points was proven to be 3SUM-Hard in [9]. We now reduce our problem of digital line fitting to the problem of finding three colinear points.

Given a set S of discrete points, we can compute the value δ corresponding to the length of the bounding box of S in linear time. For three points $\boldsymbol{a} = (x_{\boldsymbol{a}}, y_{\boldsymbol{a}})$, $\boldsymbol{b} = (x_{\boldsymbol{b}}, y_{\boldsymbol{b}})$, $\boldsymbol{c} = (x_{\boldsymbol{c}}, y_{\boldsymbol{c}})$ such that $x_a \neq x_b$, the vertical distance between a straight line $l_{\boldsymbol{ab}}$ going through $\boldsymbol{a}$ and $\boldsymbol{b}$ and a point $\boldsymbol{c}$ is given by

$d(l_{\boldsymbol{ab}}, \boldsymbol{c}) = \frac{|(\boldsymbol{c}-\boldsymbol{a}) \wedge (\boldsymbol{b}-\boldsymbol{a})|}{|(0,1) \wedge (\boldsymbol{b}-\boldsymbol{a})|}$. Note that $\wedge$ is the exterior product between two vectors, which is an algebraic generalization of the cross product between two 3D vectors to any d dimension (here, $d = 2$), so that the result is a bivector. As we process integer coordinates, if we consider three non colinear points, we thus know that $|(\boldsymbol{c} - \boldsymbol{a}) \wedge (\boldsymbol{b} - \boldsymbol{a})| \geq 1$. Then, we know that $|(0, 1) \wedge (\boldsymbol{b} - \boldsymbol{a})| = |x_{\boldsymbol{a}} - x_{\boldsymbol{b}}| \leq \delta$ by definition of δ. Therefore, we can conclude that for any three non-colinear points $\boldsymbol{d}$, $\boldsymbol{e}$, $\boldsymbol{f}$ of S, we have: $d(l_{\boldsymbol{de}}, \boldsymbol{f}) \geq \frac{1}{\delta}$. We can scale the points of S in linear time by an integer factor δ in order to obtain the new set S'. Then, any three non-colinear points $\boldsymbol{d}'$, $\boldsymbol{e}'$, $\boldsymbol{f}'$ of S' must satisfy $d(l_{\boldsymbol{d}'\boldsymbol{e}'}, \boldsymbol{f}') \geq 1$.

To conclude, if our algorithm for finding an optimal digital line detects at least three points $\boldsymbol{d}$, $\boldsymbol{e}$, $\boldsymbol{f}$ that can be covered by a digital line, this means that they satisfy $d(l_{\boldsymbol{de}}, \boldsymbol{f}) < 1$. However, as we know that in S', three non colinear points would generate a thickness equal to or greater than 1, this means that these three points are colinear. Thus, using our algorithm and a linear-time reduction, we can detect if S contains three colinear points, therefore our problem is 3SUM-hard.

The extension of 3SUM problem to higher dimensions is considered as the problem of detecting affine degeneracy of a given collection of N hyperplanes, *i.e.* finding a subset of $d + 1$ hyperplanes intersecting in a common point. This is conjectured to require $O(N^d)$ time. In this paper, we do not seek a proof since the case for $d = 2$ is sufficient for our purpose, as we seek a linear algorithm.

As solving the problem exactly in arbitrary dimension is likely at least quadratic, we next suggest two approximations. The first has a theoretical proved characteristics but is not easy to implement. The second is a more practical algorithm that features a proven worst case runtime and can be easily implemented. Moreover, its practical runtime can be much better than what is suggested by its worst case analysis.

4 Approximation with Bounded Error in Number of Inlier Points

In this section, we show that if the optimal number of inlier points is not too small (*i.e.* $\Omega(N)$), an approximation of the optimal digital hyperplane can be found in linear time, with respect to N and the runtime also depends on the given approximation value ε. We use the dual space and make a simple use of the tool to solve approximately the problem of linear programming with violations, presented in [2]. In this approximation, we do not use the fact that the points lie on a grid and it is correct for any set of points in $\mathbb{R}^d$.

We start with the results of Aronov et al. [2] on linear programming with violations. They obtained a randomized algorithm which is correct with high probability. Afshani et al. also [1] obtained a Las Vegas algorithm (*i.e.* one that either provides the correct answer or informs about failure).

Theorem 1 (Aronov et al. [2]). *Let L be a linear program with n constraints in $\mathbb{R}^d$, and let f be the objective function to be minimized. Let k_{opt} be the minimum number of constraints that must be violated to make L feasible, and let $\boldsymbol{v}$ be the point minimizing $f(\boldsymbol{v})$ with k_{opt} constraints violated. Then one can output a point $\boldsymbol{u} \in \mathbb{R}^d$ such that $\boldsymbol{u}$ violates at most $(1+\varepsilon)k_{opt}$ constraints of L, and $f(\boldsymbol{u}) \leq f(\boldsymbol{v})$. The results returned are correct with high probability. The expected running time (which also holds with high probability) of this algorithm is $O(n + \varepsilon^{-4} \log n)$ for $d = 2$, and $O(n(\varepsilon^{-2} \log n)^{d+1})$ for larger d.*

In our case, we are only interested in finding a point in the dual space that is covered by the maximum number of w-slabs. We reduce this problem to the problem of linear programming with violations and solve it using the result of [2]. The following observation is a result of replacing each w-slab with two halfspaces that have the w-slab as their intersection, represented by (3). We thus have $2N$ constraints and a point in space is violating (*i.e.*, not covered by) k w-slabs among N, if and only if it is violating k halfspaces among the $2N$. Therefore, the tool for finding a point violating the minimum number of halfspaces finds also a point that is covered by the maximum number of w-slabs. Let n_{opt} be the maximum possible number of w-slabs that can contain a point in d-space (*i.e.*, inliers) and let k_{opt} the optimal number of violations, $N = k_{opt} + n_{opt}$. For the approximation parameter, we observe that since we find a point that violates at most $(1+\varepsilon)k_{opt}$ slabs, we actually find a point that is covered by at least $N - (1+\varepsilon)k_{opt}$ w-slabs. Let n be the number of points that are covered by the w-slab we just found. If $n_{opt} = \Omega(N)$ we have $n > (1 - c\varepsilon)n_{opt}$ for some fixed c.

We now observe the following:

Observation 2. *Given a set of N points in the d-dimensional space and some $\varepsilon > 0$, we can find a w-slab that contains at least $(1-\varepsilon)n_{opt}$ points, where n_{opt} is the maximum possible number of points that can be found in a w-slab, assuming $n_{opt} = \Omega(N)$. The runtime is $O(N + \varepsilon^{-4} \log N)$ for $d = 2$ and $O(N(\varepsilon^{-2} \log N)^{d+1})$ for larger d.*

This result can be immediately used for our original problem of finding the optimal digital hyperplane by using the set of grid points.

5 Approximation with Bounded Error in Digital Hyperplane Width

In this section, we show another kind of approximation that makes use of the fact that the input points are in a bounded grid as well. The meaning of this approximation is slightly different from the previous one. The advantage of this version is that it is easy to implement, unlike the previous one that is mainly of theoretical interest and is probably hard to implement. Moreover, using this approximation, we can do both: prove its worst case runtime and also allow to get optimal solution with practical good runtime and/or to have an approximation to any desired level in better runtime than the worst case. We need the following result from Fonseca and Mount [8].

Theorem 2 (Fonseca and Mount [8]). *For a set of N points in the unit d-dimensional cube and some $\varepsilon > 0$, one can build a data structure with $O(\varepsilon^{-d})$ storage space, in $O(N + \varepsilon^{-d}\log^{O(1)}(\varepsilon^{-1}))$ time, such that for a given query hyperplane $\mathbf{H}$, the number of points on and bellow $\mathbf{H}$ can be approximately reported in $O(1)$ time, in the following sense: all the points (below $\mathbf{H}$) that have a larger distance*[1] *than ε from $\mathbf{H}$ are counted. Points that are closer to $\mathbf{H}$ on both sides may or may not reported.*

See the next section for the detail of building a specially-designed data structure for the query.

We can now state and prove the main theorem of this section:

Theorem 3. *Given a set of N points on a grid $[0, \delta]^d$, and some $\varepsilon > 0$, $w > 0$, a digital hyperplane of width $w + 5\varepsilon$ that contains $n > n_{opt}$ points, can be found in $O(N + (\frac{\delta}{\varepsilon})^d \log^{O(1)}(\frac{\delta}{\varepsilon}))$ time where n_{opt} is the maximum number of points that any digital hyperplane of width w in $[0, \delta]^d$ can contain.*

Proof. For simplicity, we can assume that all points are given in the unit d-dimensional cube such that their coordinates are integer multiplication of $1/\delta$. Let $w' = w/\delta$ be the new width and $\varepsilon' = \varepsilon/\delta$ be the new approximation parameter. We first build the data structure for halfspace range count in $O(N + \varepsilon'^{-d}\log^{O(1)}(\varepsilon'^{-1}))$ time (Theorem 2) and then we query all $O(\varepsilon'^{-d})$ hyperplanes twice (every digital hyperplane is a w-slab that is the intersection of two halfspaces) in constant time for each one of them.

For the approximation, we will consider the 2-dimensional case, which is simpler to describe. Our algorithm queries all $(w' + 3\varepsilon')$-slabs by querying two parallel lower halfplanes of distance $w' + 3\varepsilon'$ and subtract their returned numbers. As seen in Figure 1, any optimal digital line must be contained between two such parallel lines of distance $w' + 3\varepsilon'$, $\mathbf{Q}_1$, $\mathbf{Q}_2$. Let n_{opt} be the optimal number of points contained in the optimal digital line. Let n_1 be the number of points returned by querying a line $\mathbf{Q}_1$ and n_2 be the number of points returned by querying $\mathbf{Q}_2$. We only build the database for lower halfplanes (*i.e.* those contain $(0, -\infty)$). Note that we query all possible halfplanes that exist in the data structure, which is discretized so that the minimum distance between two is ε. The runtime is indeed linear in N and in δ^d since we only have $O((\frac{\delta}{\varepsilon})^d)$ halfplanes.

Recall that by the approximation guaranteed by the data structure, if the optimal line is contained between $\mathbf{Q}_1$ and $\mathbf{Q}_2$, then $n_1 - n_2 \geq n_{opt}$. On the other hand, a larger number can be found elsewhere when we query some other slab, but it is guaranteed (see Figure 1) that all reported points lie within a slab of width $w' + 5\varepsilon'$. Thus we either found a slab containing the optimal digital line or we found another slab. In both cases, the number of reported points is greater than or equal to n_{opt} and the width of the slab containing them is $w' + 5\varepsilon'$.

A typical use for digital lines would be to use $w = 1$ and $0 < \varepsilon < 0.5$.

[1] The ℓ_2 distance is used in [8] while the ℓ_1 distance is used in this paper. However, the theorem still holds since the ℓ_1 distance is always greater than or equal to the ℓ_2 distance.

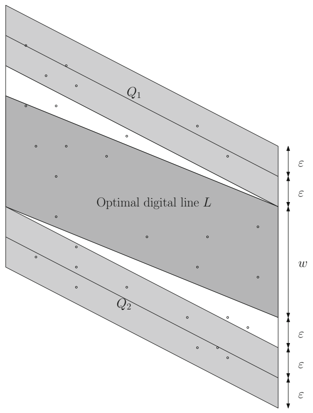

Fig. 1. The relations between optimal and approximate digital lines (see the text). Note that the approximate digital line does not **necessarily** contains the optimal line.

6 Implementation

We first note that for most practical situations, the set of points given as input is obtained through some sort of feature extraction algorithm (e.g. corner or edge detector). These extractors always require runtime which is at least linear in the size of the bounding box (in voxels). This means that our approximate algorithm is optimal in the sense of its runtime.

We implemented the algorithm described in Section 5 in `C++` on a standard PC. In contrast to the optimal algorithm, the new algorithm is very simple to implement using recursion. We implemented the algorithm for $d = 3$ as this case is relatively expensive to solve with an optimal algorithm that would have a runtime of $O(N^3)$. Optimal algorithms in 3D and above are also effectively difficult to implement. Our implementation is conceptually similar in any dimension. For every box in space we first build the range counting data structure recursively. We first find the points that lie inside each one of the 8 children of the box (this is performed in logarithmic time using orthogonal range searching) and then we create the range counting data structure for each one of them recursively using the box of size ε as the smallest box. Then for building the data structure for the current box we create the set of all possible digital planes (depending on ε) and query each one of them in all children, summing the number of points.

We now go over all planes in the outer box (the bounding box of the set of input points) and for each plane, we query the number of points bellow this plane and bellow a parallel plane of distance $w + 5\varepsilon$ as described in the previous section. We simply take the pair (which is a slab) in which the number of points resulting from the subtraction of the two is the largest. A practical problem however is the memory requirement, since the algorithm has space (and roughly time) complexity $O((\delta/\varepsilon)^3)$, so in 3D it may require large amounts of memory depending on the approximation parameter.

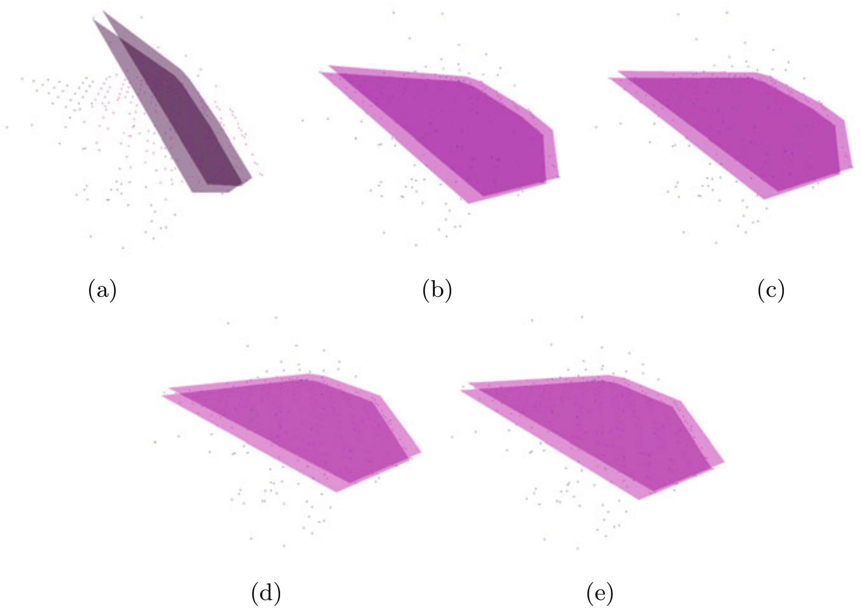

Fig. 2. Experimental results of digital plane fitting for a 3D synthetic volume data generated by 400 points in a digital plane and 100 randomly generated points (outliers). The approximation algorithm with bounded error in digital plane width is applied with the value of ε set to be 2.0 (a), 1.0 (b), 0.5 (c), and 0.25 (d): rose points are inliers and blue points are outliers. A fitted digital plane is visualized as a pair of rose parallel planes. The optimal solution (e) obtained by the exact algorithm [22] is also illustrated.

7 Experimental Results

We used two 3D digital images for our experiments: some synthetic test data and a real electron nano-tomography image.

The 3D synthetic data was created such that 400 grid points were samples from a digital plane formulation with setting $w = 1$, and 100 grid points were added randomly. Four different ε values were used: 2.0, 1.0, 0.5 and 0.25. For comparison, the exact algorithm [22] with time complexity $O(N^3 \log N)$ was also applied. As seen in Table 1, the computation time is long, however it does yield the optimal solution containing 406 inlier points (*i.e.* the 400 expected points plus 6 random ones).

The approximation results indicate, as illustrated in Fig. 2 and Table 1, that the smaller the value of ε, the more precise the solution; when $\varepsilon = 2$, a solution relatively far from the optimal was obtained. Table 1 also shows that two different numbers of points were obtained for each value of ε: the first one is the number of points in a fitted digital plane with width w, and the second one is the number of points in a fitted digital plane with width $w+5\varepsilon$. Note that the second number is guaranteed to be greater than or equal to the optimal number of inlier points by design, and it indeed converges to the optimum as ε is decreasing (see Table 1).

Table 1. The runtimes, parameters and numbers of points of fitted digital planes in Fig. 2

	runtime	parameters				nb. of points	
		a_1	a_2	a_3	a_4	w	$w+5\varepsilon$
Approximate fitting							
$\varepsilon = 2.0$	31 msec	−0.0625	−0.0625	1	−10.9	110	485
$\varepsilon = 1.0$	266 msec	−0.5	−0.5	1	−3.0002	300	435
$\varepsilon = 0.5$	2172 msec	−0.5	−0.5	1	−3.0002	300	421
$\varepsilon = 0.25$	16640 msec	−0.5625	−0.546875	1	−2.55002	362	410
Exact fitting [22]							
with exact comp.	35 min 29.109 sec	−47/81	−43/81	1	−203081/81000	406	
without exact comp.	4 min 36.908 sec	−0.580247	−0.530864	1	−2.507173		

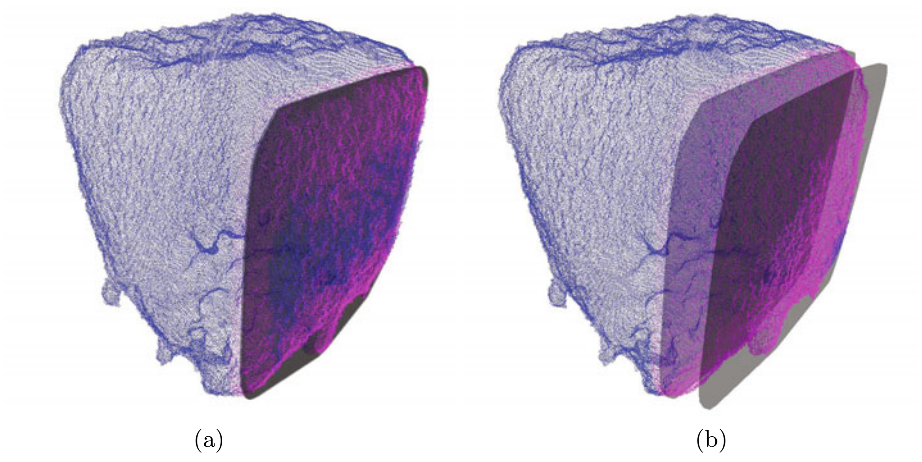

Fig. 3. Results of digital plane fitting for a pre-processed 3D binary nano-tomography image containing 205001 points. The approximation algorithm with bounded error in digital plane width is applied with $\varepsilon = 4$ for $w = 1$ (a) and $w = 25$ (b).

We can observe as well from Table 1 that the smaller the value of ε, the higher the runtime. Therefore, it is necessary in practice to find an appropriate value for ε, which provides a sufficiently approximate solution within a reasonable timeframe.

The second data we used was a 3D binary image generated from a electron nano-tomography image containing a cubical crystal. The image is very noisy due to the dimension of the sample and the physically-constrained tomography reconstruction method. The original image is of $512 \times 511 \times 412$ with gray values. After binarizing the image by a threshold, we detected the boundary points by using the 6-neighborhood and extracted the maximum connected component by using the 26-connectivity. Finally, we obtained 205001 points in the $512 \times 511 \times 412$ grid. This number of points is too large to apply the exact algorithm [22]: it would have required on the order of 10^{18} operations, *i.e.* many months of runtime. Instead we ran the approximate algorithm with some adjustments of the values for ε and w. We set $\varepsilon = 4$ to obtain its runtime around 12 seconds with $w = 1$; the result is illustrated in Fig. 3 (a). As we saw that the cube wall is very noisy, we also performed a fitting with $w = 25$ to obtain a thicker digital plane (see Fig. 3 (b)).

8 Conclusion

In this article we have presented two approximate discrete hyperplane fitting methods with outliers. The first uses an approach based on linear programming with violations. It is continuous in nature and features interesting complexities but is difficult to implement. The second is discrete in nature, it uses an accumulation and query data structure and is easy to implement. This method features bounded error defined in this way: for a given N points, a width w and an error factor ε, the hyperplane found contains in a width equal to $w + 5\varepsilon$ at least as many points as the optimum would with a width of w. It features a computational complexity in $O(N + \left(\frac{\delta}{\varepsilon}\right)^d)$, where δ is the distance between two neighbours in the hypergrid. The algorithm is therefore linear in the number of points in the set being considered, but exponential in the approximation factor with d, the geometric dimension. Memory requirements are also exponential with ε and d in the same way.

Nonetheless, as we show in the article that the exact solution is 3SUM-hard, this method is useful. The computational complexity is given in the worst case, in practice it can be much better. The algorithm is in particular well-behaved when there are few outliers. Due to lack of space, we did not show it in the paper, but the algorithm converges to the optimal solution, and in the discrete case there exists a finite ε for which the algorithm provides the optimal solution, even though this ε might be too small to be practical.

References

1. Afshani, P., Chan, T.M.: On approximate range counting and depth. Discrete & Computational Geometry 42(1), 3–21 (2009)
2. Aronov, B., Har-Peled, S.: On approximating the depth and related problems. SIAM J. Comput. 38(3), 899–921 (2008)
3. Bhowmick, P., Bhaattacharya, P.: Fast approximation of digital curves using relaxed straightness properties. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(9), 1590–1602 (2007)
4. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press, Cambridge (2004)
5. Chum, O.: Two-View Geometry Estimation by Random Sample and Consensus. Ph.D. thesis, Czech Technical University, Prague, Czech Republic (July 2005)
6. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Comm. ACM 15, 11–15 (1972)
7. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)

8. da Fonseca, G.D., Mount, D.M.: Approximate range searching: The absolute model. Comput. Geom. 43, 434–444 (2010)
9. Gajentaan, A., Overmars, M.H.: On a class of $o(n^2)$ problems in computational geometry. Comput. Geom. 5, 165–185 (1995)
10. Hartley, R., Zisserman, R.: Multiple view geometry in computer vision. Cambridge University Press, Cambridge (2003)
11. Hough, P.V.C.: Machine analysis of bubble chamber pictures. In: Proc. Int. Conf. High Energy Accelerators and Instrumentation (1959)
12. Kenmochi, Y., Buzer, L., Sugimoto, A., Shimizu, I.: Discrete plane segmentation and estimation from a point cloud using local geometric patterns. International Journal of Automation and Computing 5(3), 246–256 (2008)
13. Kenmochi, Y., Buzer, L., Talbot, H.: Efficiently computing optimal consensus of digital line fitting. In: International Conference on Pattern Recognition, pp. 1064–1067 (2010)
14. Köster, K., Spann, M.: An approach to robust clustering - application to range image segmentation. IEEE Transaction on Pattern Analysis and Machine Intelligence 22(5), 430–444 (2000)
15. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical recipes: the art of scientific computing, 3rd edn. Cambridge University Press, Cambridge (2007)
16. Reveillès, J.P.: Géométrie discrète, calcul en nombres entiers et algorithmique. Ph.D. thesis, Thèse d'Etat, Université Louis Pasteur (1991)
17. Rousseeuw, P.J.: Least median of squares regression. Journal of the American Statistical Association 79(388), 871–880 (1984)
18. Shum, H.Y., Ikeuchi, K., Reddy, R.: Principal component analysis with missing data and its application to polyhedral object modeling. IEEE Transaction on Pattern Analysis and Machine Intelligence 17(9), 854–867 (1995)
19. Sivignon, I., Dupont, F., Chassery, J.M.: Decomposition of a three-dimensional discrete object surface into discrete plane pieces. Algorithmica 38, 25–43 (2004)
20. Zitova, B., Flusser, J.: Image registration methods: a survey. Image and Vision Computing 21, 977–1000 (2003)
21. Zrour, R., Kenmochi, Y., Talbot, H., Buzer, L., Hamam, Y., Shimizu, I., Sugimoto, A.: Optimal consensus set for digital line fitting. In: Proceedings of IWCIA 2009, Cancun, Mexico (2009)
22. Zrour, R., Kenmochi, Y., Talbot, H., Buzer, L., Hamam, Y., Shimizu, I., Sugimoto, A.: Optimal consensus set for digital plane fitting. In: Proceedings of 3DIM 2009, Workshop of ICCV, Kyoto, Japan (2009)
23. Zrour, R., Kenmochi, Y., Talbot, H., Buzer, L., Hamam, Y., Shimizu, I., Sugimoto, A.: Optimal consensus set for digital line and plane fitting. International Journal of Imaging Systems and Technology (2010) (in print)
24. Zrour, R., Kenmochi, Y., Talbot, H., Shimizu, I., Sugimoto, A.: Combinatorial optimization for fitting of digital line and plane. In: Wada, T., Huang, F., Lin, S. (eds.) PSIVT 2009. LNCS, vol. 5414. Springer, Heidelberg (2009)