Combinatorial optimization for fitting digital line and plane

Rita Zrour*, Yukiko Kenmochi*, Hugues Talbot*
Ikuko Shimizu†
Akihiro Sugimoto‡
* Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI, ESIEE Paris - CNRS, France
†Tokyo University of Agriculture and Technology, Japan
‡National Institute of Informatics, Japan

***Abstract*—We present a method for fitting a digital line/plane from a given set of 2D/3D grid points. In the framework of discrete geometry, a digital line or plane is defined as a set of grid points located between two parallel lines or planes separated by a small distance. Our purpose is to identify such a pair of Euclidean lines/planes that represents a given set of points. This problem is formulated as a mixed integer/linear programming problem, with the objective to maximize the number of points between the two lines/planes.**

I. Introduction

In the computer vision literature, recognition of straight lines and planes has been studied for decades. Since digital images always contain outliers, most robust recognition algorithms are based on statistical methods and solved by optimization methods using, for example, regression tools such as least squares, least absolute deviation, and least median of squares linear regression [9]. Most of these algorithms work in a continuous space. However since the data are digital, it seems logical to work with a digital model rather than a continuous one. Recently, purely discrete formulation of many classical and novel image processing problems has been proposed under the heading of "discrete geometry" [1]. In this domain, work is accomplished on discrete data within a discrete space. In other words, we don't have any digitization error since we treat only grid points whose coordinates are represented as integers.

In a discrete space, a digital line/plane is defined as a set of grid points located between two parallel lines/planes separated by a small distance [1], [7]. In discrete geometry, several algorithms were proposed for recognizing a digital line or plane for a given grid point cloud [3], [8]. They are, for example, based on linear programming or computational geometry using convex hulls or other geometrical properties. Those algorithms determine if a point cloud fits to a digital line or plane. Therefore, they are very sensitive to outliers; if a digital image contains outliers, they simply return the negative answer. Recently, new methods for blurred data were introduced in the framework of discrete geometry [4], [5], [6], [10]; for example [5] consists of minimizing the distance between two parallel lines/planes containing all points of a given point cloud. Therefore, outliers are not separated from a fitted digital line/plane. In other words, there is only notion of inliers so we cannot process outliers with these methods.

Our purpose in this paper is to fit a digital line or plane to a given set of 2D or 3D grid points. This fitting is accomplished by defining inliers and outliers. The work is accomplished in a discrete space so that discretisation errors are processed naturally. The fitting is accomplished by maximizing the inliers and rejecting the outliers. Figure 1 shows an example of fitting a digital line to a given set of points by rejecting the points that are considered as outliers. In order to maximize the inliers we use the L0-norm, unlike statistical methods that use L1-norm or L2-norm. L0-norm considers only the binary distances 0 or 1; inliers are thus at distance of 0 while outliers are at distance of 1. Remark that such simple distances can be used because our model is digital. We formulate the problem into a mixed integer linear programming that aims to find the parameters of the digital line or plane by minimizing the number of outliers. This formulation allows us to find an optimal solution; this constitutes an advantage over statistical methods that work on a continuous non integer space and sometimes yields only approximate solutions.

This paper is organized as follows. Section II presents discrete geometrical definitions of digital lines and planes. Section III proposes the formulation for fitting a set of points to a digital line or plane using the discrete geometrical definitions; we show that our problem is written as mixed integer linear programming problems. Section IV presents some experiments by testing the method. Section V discusses the computational issue. Section VI presents an improvement to consume less computation time. Finally, Section VII states some conclusions and perspectives.

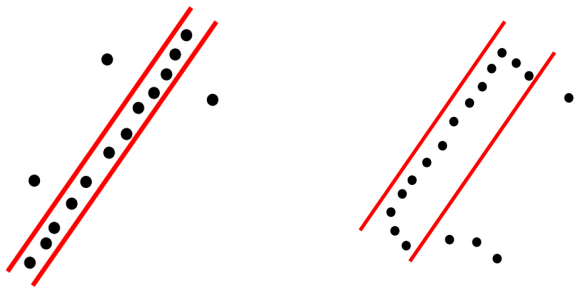

Fig. 1. Fitting a digital line to a given set of points rejecting some points considered as outliers

II. Discrete geometrical definitions of digital lines and planes

Unlike in the continous domain, in the framework of discrete geometry, we take into account digitization errors when we

represent straight lines and planes in a digital image. In this section we give the definitions of digital lines and planes found in [1].

A. Digital lines

Let $\mathbb{R}$ be the set of real numbers. A line $\mathbf{L}$ in the 2D Euclidean space is defined by:

$$\mathbf{L} = \{(x, y) \in \mathbb{R}^2 \ : \ \alpha x + \beta y + \delta = 0\} \tag{1}$$

where $\alpha, \beta, \delta \in \mathbb{R}$.

Let $\mathbb{Z}$ be the set of integers. $\mathbb{Z}^2$ denotes the set of grid points whose coordinates are all integers in the 2D Euclidean space $\mathbb{R}^2$. We now consider a digitization of $\mathbf{L}$, which is given as a grid-point subset in $\mathbb{Z}^2$, called a digital line. There are various digitization techniques for a given $\mathbf{L}$ [1]. In this paper we adopt a grid-line digitization with respect to $\mathbf{L}$. Such a digitization of $\mathbf{L}$ is given by

$$\mathbf{D}(\mathbf{L}) = \{(x, y) \in \mathbb{Z}^2 \ : \ 0 \leq \alpha x + \beta y + \delta < \omega\} \tag{2}$$

where $\omega = \max(|\alpha|, |\beta|)$. $\mathbf{D}(\mathbf{L})$ is the set of all grid points that are closest to the intersection points of $\mathbf{L}$ with the grid lines of $\mathbb{Z}^2$ and that are in the upper/under side of $\mathbf{L}$. From the definition, it is obvious that we obtain a unique digital line $\mathbf{D}(\mathbf{L})$ from a given $\mathbf{L}$ [1].

B. Digital planes

A plane $\mathbf{P}$ in the 3D Euclidean space is defined by:

$$\mathbf{P} = \{(x, y, z) \in \mathbb{R}^3 \ : \ \alpha x + \beta y + \gamma z + \delta = 0\} \tag{3}$$

where $\alpha, \beta, \gamma, \delta \in \mathbb{R}$.

$\mathbb{Z}^3$ denotes the set of grid points whose coordinates are all integers in the 3D Euclidean space $\mathbb{R}^3$. Concerning a digitization of $\mathbf{P}$, which is given as a grid-point subset in $\mathbb{Z}^3$, called a digital plane, we adopt a grid-line digitization similarly to digital lines. Such a digitization of P is then given by

$$\mathbf{D}(\mathbf{P}) = \{(x, y, z) \in \mathbb{Z}^3 \ : \ 0 \leq \alpha x + \beta y + \gamma z + \delta < \omega\} \tag{4}$$

where $\omega = \max(|\alpha|, |\beta|, |\gamma|)$. As with discrete line, $\mathbf{D}(\mathbf{P})$ is unique for every $\mathbf{P}$ [1].

III. Digital line/plane fitting

As stated in Section I, given a finite subset $\mathbf{S}$ of $\mathbb{Z}^n$ where $n = 2, 3$, our purpose is to fit a digital line (2D case) or plane (3D case) to $\mathbf{S}$ and to estimate its parameters. In our case, we would like to reject some points considered as outliers of $\mathbf{S}$.

A. Formulation

If $\mathbf{S}$ contains no outlier, in 2D, we are simply interested in finding the solution set (α, β, δ) such that all points in $\mathbf{S}$ satisfying (2). Similarly in 3D, we are interested in finding the solution set $(\alpha, \beta, \gamma, \delta)$ such that all points in $\mathbf{S}$ satisfying (4).

If $\mathbf{S}$ contains outliers, however, we find no solution for any above linear inequality set. For such infeasible cases, we proposed to use integer linear programming to solve the problem. Integer linear programming allows us to separate the inliers from outliers.

The 2D formulation cas be described as follows: let N be the number of elements of $\mathbf{S}$ and p_i be a binary decision variable for each point $(x_i, y_i) \in \mathbf{S}$, where $i = 1, 2, \ldots, N$, such that $p_i = 0$ if a point (x_i, y_i) satisfies the linear inequalities of a digital line; otherwise, $p_i = 1$. The integer linear programming problem can be described by:

[LP 1-j]

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1,\ldots,N} p_i \\
\text{subject to} \quad & -Mp_i \leq \alpha x_i + \beta y_i + \delta < Mp_i + \omega \\
& \qquad \text{for all } i = 1, \ldots, N \qquad (5) \\
& 0 \leq p_i \leq 1 \text{ and integer} \qquad (6)
\end{aligned}$$

where M is a large number constant.

In order to simplify (5), it is possible to divide (5) by $\omega = \max(|\alpha|, |\beta|)$; this leads us to the following inequalities:

$$\frac{-M}{\omega} p_i \leq \frac{\alpha}{\omega} x_i + \frac{\beta}{\omega} y_i + \frac{\delta}{\omega} < \frac{M}{\omega} p_i + 1 \tag{7}$$

Looking into (7), we see that four cases can be distinguished:

1) $\omega = |\alpha|$, $\alpha \geq 0$
2) $\omega = |\alpha|$, $\alpha \leq 0$
3) $\omega = |\beta|$, $\beta \geq 0$
4) $\omega = |\beta|$, $\beta \leq 0$

These four cases can be reduced into the following two cases and (7) or (5) is rewritten as follows:

[LP 1-1] If $\omega = |\beta|$ (j=1),

$$-M' p_i \leq \alpha_1 x_i + y_i + \delta' < M' p_i + 1 \tag{8}$$

where $M' = \frac{M}{\omega}$, $\alpha_1 = \pm\frac{\alpha}{\omega}$, $\delta' = \frac{\delta}{\omega}$ or $-\frac{\delta}{\omega} + 1$. These substitutions yield the following constraint:

$$-1 \leq \alpha_1 \leq 1.$$

[LP 1-2] Otherwise (j=2),

$$-M' p_i \leq x_i + \alpha_2 y_i + \delta' < M' p_i + 1 \tag{9}$$

where $M' = \frac{M}{\omega}$, $\alpha_2 = \pm\frac{\beta}{\omega}$, $\delta' = \frac{\delta}{\omega}$ or $-\frac{\delta}{\omega} + 1$. These substitutions impose the following constraints:

$$-1 \leq \alpha_2 \leq 1.$$

Regarding the 3D case, its formulation is similar to the 2D one. Three cases are distinguished depending on the cases where $\omega = |\alpha|$, $\omega = |\beta|$ and $\omega = |\gamma|$. The integer linear programming is described for the case $\omega = |\alpha|$ by:

[LP 2-1]

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1,\ldots,N} p_i \\
\text{subject to} \quad & -M' p_i \leq x_i + \alpha_2 y_i + \alpha_3 z_i + \delta' < M' p_i + 1 \\
& \qquad \text{for all } i = 1, \ldots, N \qquad (10) \\
& -1 \leq \alpha_2 \leq 1, \qquad (11) \\
& -1 \leq \alpha_3 \leq 1, \qquad (12) \\
& 0 \leq p_i \leq 1 \text{ and integer} \qquad (13)
\end{aligned}$$

where M' is a large number constant. If we consider the case where $\omega = |\beta|$, then we replace (10) and (11) by the following inequalities respectively:

$$\textbf{[LP 2-2]} \quad -Mp_i \leq \alpha_1 x_i + y_i + \alpha_3 z_i + \delta' < Mp_i + 1 \quad (14)$$
$$-1 \leq \alpha_1 \leq 1. \quad (15)$$

If we consider the case where $\omega = |\gamma|$, then we replace (10) and (12) by the following inequalities respectively:

$$\textbf{[LP 2-3]} \quad -Mp_i \leq \alpha_1 x_i + \alpha_2 y_i + z_i + \delta' < Mp_i + 1 \quad (16)$$
$$-1 \leq \alpha_1 \leq 1. \quad (17)$$

It is noticed that we have two formulations in 2D and three formulations in 3D. We may generalize that we have n formulations in nD.

B. Algorithm

We have to solve two linear programming problems **[LP 1-1]** and **[LP 1-2]** for the 2D case, and three problems **[LP 2-1]**, **[LP 2-1]** and **[LP 2-3]** for the 3D case. Let F_j to be the solution of the objective function of the j-th problem **[LP n-j]** for $j = 1, 2, \ldots, n$ where $n = 1, 2$. Once all the $F_j s$ are computed, the minimum among them is chosen such that $F = \min_{j=1,2,\ldots,n}(F_j)$. It allows us to detect an optimal solution that yields fewer outliers and fits more points into a digital line or plane.

Algorithm 1 describes the different steps of fitting a digital line and plane to a given set of points for the 2D and 3D cases.

Algorithm 1: Fitting a digital line or plane to a given sets of grid points

input : A given set $\mathbf{S}$ of N points in $(n+1)$D for $n = 1$ or 2
output: The parameters $\alpha_1, \ldots, \alpha_n, \delta'$ of the nD hyperplane and p_i of every point in $\mathbf{S}$

```
begin
    initialize F = N ;
    for j = 1, ..., n do
        solve [LP n-j] ;
        set F_j to be the optimal value of the objective function;
        if F_j < F then
            F = F_j;
            set α_1, ..., α_n, δ' and p_i to be the optimal values;
    return;
end
```

IV. Experiments

A. 2D example

Our input data is a set of points $(x, y) \in \mathbb{Z}^2$ that satisfies $y = -6x$ and another set of points $(x, y) \in \mathbb{Z}^2$ that satisfies $y = -x + 5$. We used the free linear programming solver, `lp_solve` [11] to solve each **[LP n-j]** problem.

As mentioned in Section III-B, we compute two F_j, $j = 1, 2$ in 2D and then we choose the minimum between them. Figure 2 shows an example of fitting a digital line to a given set of points; between the two solutions, one is optimal since it fits more points into a digital line.

TABLE I
Parameter estimation results for 200 points oscillating between two discrete lines

Number of inliers	Number of outliers	α_1	α_2	δ'
192	8	1	1	-5
182	18	1	1	-5
170	30	1	1	-5
165	35	1	0.1667	1
175	25	1	0.1667	1

Table I provides an example of a fitting 200 points oscillating between the two discrete lines $0 \leq x + y + 5 < 1$ and the line $0 \leq x + 0.1667y + 1 < 1$. When the points move from one line to another, the numbers of inliers and outliers change and the line detected is the one that contains most inliers.

We also tested our method with respect to a real image as shown in Figure 3. The image dimensions are 520×693. Before applying our method, edge detection and mathematical morphological filtering are done for this image; the number of points in the image after this pre-processing is 1572 points. As shown in Figure 4, our method is then applied in order to fit a digital line to the set of points; Figure 5 shows the digital line detected (colored red).

We compared our method with the Hough transform [12] that is one of well-known efficient procedures for detecting lines in images. It consists of transforming each image point (x, y) into a curved line in the parameter space (θ, ρ) specified by:

$$\rho = x \cos\theta + y \sin\theta$$

It has been noticed through experiments that with the Hough transform, line detection usually requires interactive parameter adjusting. The parameters adjustement is done by changing the sampling intervals $\Delta\rho$, $\Delta\theta$ and the voting threshold t. Many lines can be detected if desired. Figures 6, 7, 8 show the Hough transformation applied to Figure 4 with different parameter values. Depending on the parameter setting, we see in the figures that the results are changed. If the threshold is too high, no line is detected. In contrast with the Hough transform, our method computes exact optimal solution without any parameter adjustement. It is noted that with our method, an iterative procedure can be applied in order to detect many lines in the image. This iterative procedure is accomplished by repeating the procedure for an optimal solution after taking off the points corresponding to the previous optimal line.

B. 3D example

Our method is also easily adapted to the 3D case in contrast to the Hough transform that does not scale well to 3D. The Hough transform requires memory consumption in 3D. For the experiments in 3D, we applied our method to a part of the same data used in [2]. Figure 9 shows a planar surface segmentation from a range image of blocks; the image size is 160×120. We have thirteen sets of segmented points in different colors except for those colored in light green that are detected as edge points. Among these thirteen sets, we tried to fit a digital plane

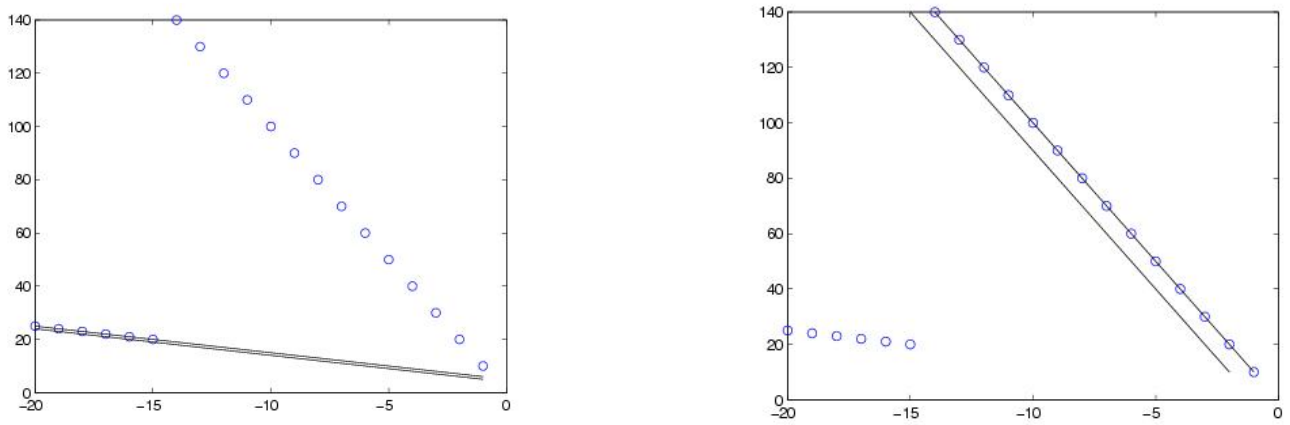

Fig. 2. Two solutions are obtained by solving **[LP 1-1]** and **[LP 1-2]** when fitting a digital line to the given set of points. The first one rejects 14 outliers while the second one rejects 6 outliers. It is thus clear that the second one gives the optimal result since it rejects less points.

Fig. 3. An original image

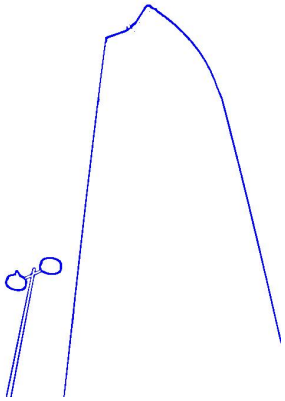

Fig. 4. A binary image after edge detection and filtering done for the image in Figure 3

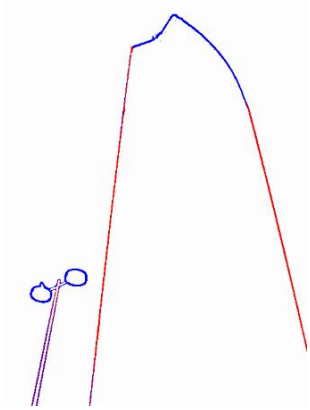

Fig. 7. Line fitting (colored red) using the Hough transform with $\Delta\theta = 1$ and t=100

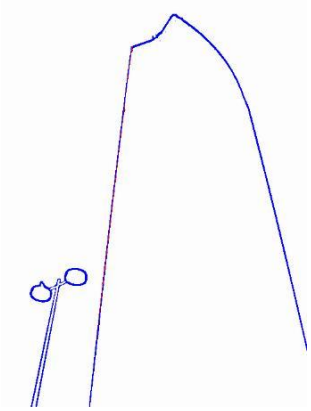

Fig. 8. Line fitting (colored red) using the Hough transform with $\Delta\theta = 1$ and t=400

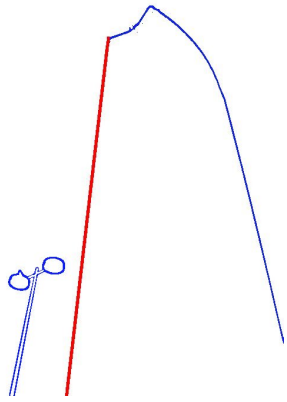

Fig. 5. Digital line fitting (red line)

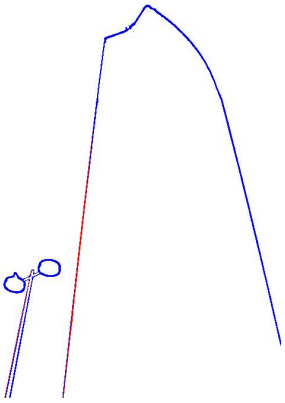

Fig. 6. Line fitting (colored red) using the Hough transform with $\Delta\theta = 3$ and t=400

to seven sets colored in green, brown, turquoise and violet. Table II shows the plane parameters for every set of points. The "Number of outliers" in Table II indicates the number of points that do not fit in the plane and N indicates the total number of points, whether they fit or not. Figures 10, 11, 12 and 13 show the digital planes obtained after the fitting for the original set of points of Figure 9 colored green, brown, turqoise and violet respectively. Outliers are also shown in these images; they are colored in pink.

V. Computation time analysis

The formulation by a mixed integer linear programming proves the efficiency of separating outliers from inliers while minimizing the number of outliers. However this formulation suffers from an important computation time when the number of points increases. In order to fit a digital line or plane to a given set $\mathbf{S}$ of N points, we need to solve all the n linear integer programming problems that consist of minimizing $\sum_{i=1,\ldots,N} p_i$. In terms of the number of variables, each of the **[LP** $n - j$**]** problems has N variables (p_i for $i = 1, 2, \ldots N$) and $n + 1$ variables ($\alpha_1, \alpha_2, \ldots, \alpha_n, \delta'$). This increases the computation time.

In section IV, for the experiments, we used a set of points $(x, y) \in \mathbb{Z}^2$ that satisfy $y = -6x$ and another set of points $(x, y) \in \mathbb{Z}^2$ that satisfies $y = -x + 5$. It was noticed that when the number of points N increases, the time complexity increases. Figure 14 shows the time variation when changing the number of points and fixing the number of outliers to 10.

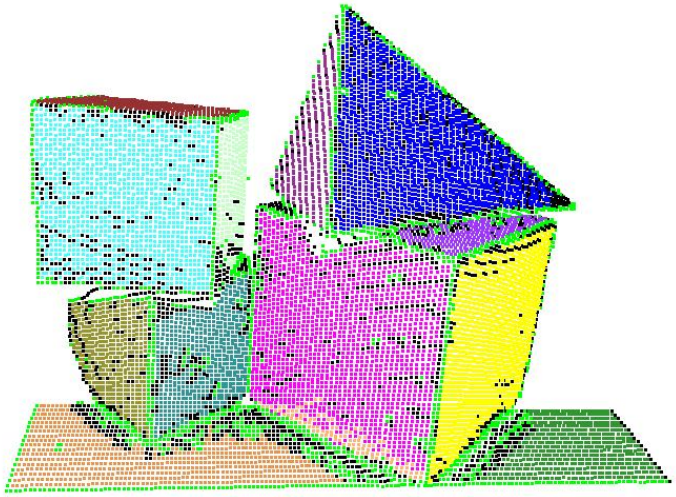

Fig. 9. Planar surface segmentation from a range image of blocks

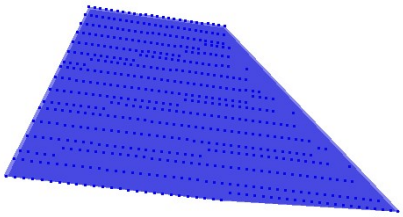

Fig. 10. A fitted digital plane for green points in Figure 9

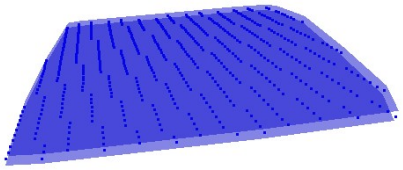

Fig. 11. A fitted digital plane for brown points in Figure 9

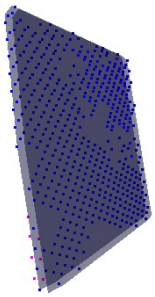

Fig. 12. A fitted digital plane for turquoise points in Figure 9

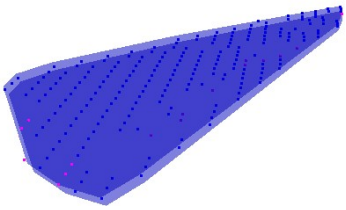

Fig. 13. A fitted digital plane for Violet points in Figure 9

It is possible to see that the computation time increases as the number of points increases.

The computation time also depends on the number of outliers. Figure 15 shows the computation time variation of **[LP 1-1]** and **[LP 1-2]** when changing the number of outliers while fixing the number of points to 200. It should be noted that the computation time depends on the percentage of outliers with respect to the number of points; when the number of outliers is around 50 percent of the intial number of points, the computation time increases sharply. Figure 15 shows the time when the number of outliers is less than 25 percent or more than 75 percent of the number of points; when it is between these two numbers, the time increases rapidly to several hours (time not shown in the diagram since it exceeds the y-axis range). Moreover, it should be noted that when we increase the dimension from two to three, the time complexity also increases, since the number of formulated **[LP]** problems increases and the number of variables increases. Sometimes many hours are needed to solve a **[LP]** problem which makes Algorithm 1 impractical.

VI. Improvement by adding a constraint

A. Problem

In most cases, each of the **[LP]** problems may give different solutions and one of the solutions is optimal since it gives the minimum $\sum_{i=1,\ldots,N} p_i$. In other cases, more than one system may

TABLE II
PARAMETER ESTIMATION RESULTS FOR THE GIVEN SETS OF POINTS

Plane colors	Number of points N	Number of outliers	α_1	α_2	α_3	δ'
$Orange$	699	6	0.015573	1	-0.568409	-330.105673
$Green$	573	0	-0.008850	1	-0.575221	-334.061947
$Brown$	545	0	0.038462	1	-0.392308	-157.484615
$Turquoise$	536	7	-1	0.5	1	517.5
$Purple$	248	0	1	-0.089888	0.235955	99.202247
$Violet$	232	16	0.062500	1	-0.562500	-260.312500
$Mossgreen$	223	0	1	-0.084856	-0.195822	-125.229765

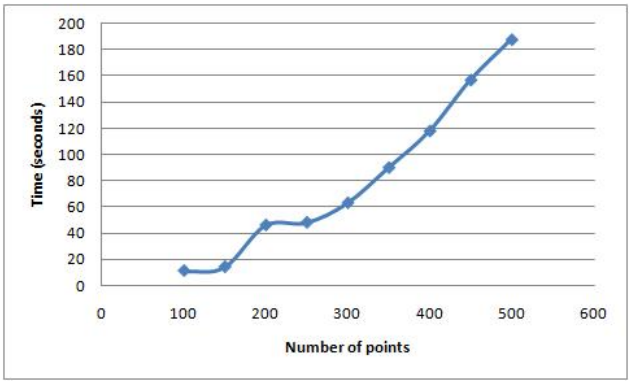

Fig. 14. Time variation as the number of points in the point cloud increases.

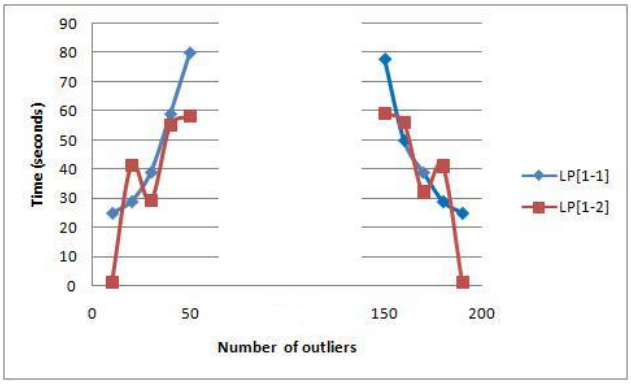

Fig. 15. Time variation as the number of outliers increases.

give a similar solution which gives a minimal $\sum_{i=1,\ldots,N} p_i$.

We cannot know in advance which system is most suitable to the set **S** of points. This fact forces us to solve two **[LP]** systems in 2D and three in 3D and then to find among them the one that gives the minimum $\sum_{i=1,\ldots,N} p_i$. The computation time is a factor that depends not only on the number of points, but it also depends on the percentage of outliers with respect to the number of points (Section V). This increases the computation time when solving the **[LP]** problem that does not give the optimal fitting solution. Thus, it is likely that the method can be improved if we can find the right **[LP]** problem among the 2 (2D) or 3 (3D) **[LP]** problems.

B. Adding a constraint

We present a solution that decreases the computation time since it permits to detect rapidly the right equation that minimizes the objective function $\sum p_i$. It consists of adding the following constraint to each **[LP]** problem proposed in Section III-A:

$$\sum_{i=1,\ldots,N} p_i = K \tag{18}$$

where K is a constant that starts from 0, meaning no outliers, and then increases until a solution is found. When a solution is found there is no need to continue such an increment since we are sure that the solution is optimal. This solution gives us the faster computation time since each time the number of outliers is fixed, by fixing the range of searching solutions for each of the n **[LP]** problems (nD case). This is a good approach when there is a limited number of outliers, which implies that at least one of the **[LP]** problems has a limited number of outliers. In such case even if the rest of the **[LP]** problems features a large number of outliers, they will have no solution with the constraint added. However when the number of outliers is relatively large, the computation time may increase and this approach is inefficient. Another approach is thus needed to solve the mixed integer programming problem. Figure 16 shows the variation of time as the constant K increases while fixing the number of points. It is possible to see that when K increases, the computation time also increases.

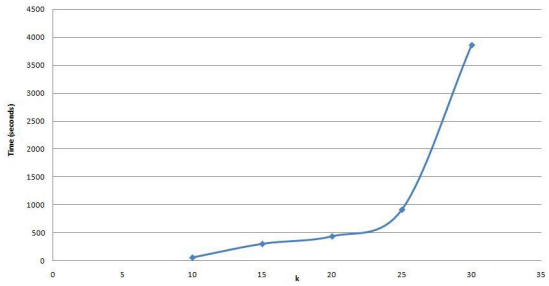

Fig. 16. Time variation with respect to the constant K

VII. CONCLUSION AND PERSPECTIVES

In this paper, we present a fitting method of a given set of grid points to a digital line or plane by formulating the problem into a mixed integer programming. The solution satisfies all points except for some outliers. The experimental results show that our method is useful for the estimation of the digital line and plane from a point cloud by determining the inliers and rejecting the outliers. It should be noted that there is no digitization error in the framework of discrete geometry

so that outliers detected are not considered to be digitization errors. However, the algorithm proposed to solve the problem is time consuming when the numbers of points increased. Several solutions are expected to decrease the computation time. These solutions should be studied and tested to see their influences on the computation time as well as on the precision of the fitting; they can be divided into three directions:

- Reducing the number of points or outliers: reducing the number of points in the point cloud can be done by applying a multi-scale approach that increases the size of image pixel or volume and reduces the number of points; this decreases the number of points and may allow us to treat outliers more quickly.
- Finding some outliers in a pre-processing step: finding outliers in a pre-processing step may reduce the computation time since it decreases the parameters space. It is accomplished by using, for example, the Hough transform. The Hough transform will enable us to reject outliers before applying our method formulated into a mixed integer programming.
- Proposing a new formulation: there already exist several well-known algorithms for fitting such as least median of squares linear regression [9]. However, the task of formulating the problem of finding digital lines or planes using these methods remains to be done. This formulation is currently under study trying to find a better and less time consuming algorithm.

ACKNOWLEDGMENT

This work was supported in part by ANR grant SURF 05-BLAN-0071.

REFERENCES

[1] R. Klette, A. Rosenfeld, "Digital Geometry: Geometric methods for digital pictures analysis", *Morgan Kauffmann*, San Francisco, 2004.

[2] Y. Kenmochi, L. Buzer, A. Sugimoto, I. Shimizu, "Discrete plane segmentation and estimation from a point cloud using local geometric patterns", *International Journal of Autumation and Computing*, 5 (3), pp.246–256, 2008.

[3] I. Debled-Rennesson, "Etude et reconnaissance des droites et plans discrets", *Ph.D. thesis, Université Louis Pasteur, Strasbourg, France*, 1995.

[4] I. Debled-Rennesson, J.L. Rémy, J. Rouyer-Degli, "Linear segmentation of discrete curves into fuzzy segments", *Discrete Applied Mathematics*, 151, pp.122–137, 2005.

[5] L. Provot, L. Buzer, I. Debled-Rennesson, "Recognition of blurred pieces of discrete plane", *Proceedings of Discrete Geometry for Computer Imagery*, Springer-Verlag, LNCS 4245, pp.65–76, Szeged, Hungary, 2006.

[6] I. Debled-Rennesson, F. Feschet, J. Rouyer-Degli, "Optimal blurred segments decomposition of noisy shapes in linear time", *Computer and graphics*, 30 (1), 2006.

[7] J.P. Reveillès, "Géométrie discrète, calcul en nombres entiers et algorithmique", *Thèse d'Etat, Université Louis Pasteur, Strasbourg, France*, 1991.

[8] L. Buzer. "An elementary algorithm for digital line recognition in the general case", *proceedings of the 12th International Conference on Discrete Geometry for Computer Imagery, DGCI-2005*, LNCS, Springer-Verlag, 3429, pp.299–310, 2005.

[9] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, "Numerical recipes (Third Edition)", *Cambridge University Press*,

[10] A. Faure, F. Feschet, "Tangential cover for thick digital curves", *proceedings of the 14th International Conference on Discrete Geometry for Computer Imagery, DGCI-2008*, LNCS, Springer-Verlag, 4992, pp.358–369, 2008.

[11] lp_solve. http://lpsolve.sourceforge.net/5.5/

[12] R.O. Duda, P.E. Hart, "Use of the Hough transformation to detect lines and curves in pictures", *Communications of the ACM*, 15 (1), pp.11–15, 1972.