Complexity Analysis for Digital Hyperplane Recognition in Arbitrary Fixed Dimension

Valentin E. Brimkov[1] and Stefan S. Dantchev[2]

[1] Fairmont State University, 1201 Locust Avenue, Fairmont, West Virginia 26554-2470, USA
vbrimkov@fairmontstate.edu

[2] University of Durham, Science Labs, South Road, Durham DH1 3LE, England
s.s.dantchev@durham.ac.uk.

Abstract. We consider the following problem. Given a set of points $M = \{p^1, p^2, \ldots, p^m\} \subseteq \mathbb{R}^n$, decide whether M is a portion of a digital hyperplane and, if so, determine its analytical representation. In our setting $p^1, p^2, \ldots, p^m$ may be *arbitrary* points (possibly, with rational and/or irrational coefficients) and the dimension n may be any arbitrary *fixed* integer. We provide an algorithm that solves this digital hyperplane recognition problem by reducing it to an integer linear programming problem of fixed dimension within an algebraic model of computation. The algorithm performs $O(m \log D)$ arithmetic operations, where D is a bound on the norm of the domain elements.

Keywords: *Digital hyperplane, digital plane recognition, integer programming.*

1 Introduction

Digital plane segment (DPS) recognition is a basic problem in image analysis, attracting a lot of interest in recent years. Several algorithms for this problem have been proposed. (See the recent survey [5] by Brimkov, Coeurjolly, and Klette). [24] suggests an algorithm based on convex hull separability. Algorithm involving plane characterization by evenness in grid adjacency models is discussed in [26]. [9] proposes an approach based on tests for existence of lower and upper supporting ("oblique") planes for the given set of points. [14] suggests recognition by least-square optimization. See also [27] for further contributions. A number of algorithms exploit the idea to reduce the problem to a relevant linear program and solve it by employing existing methods from linear programming. [10] suggests a method by converting DPS to a system of m^2 linear inequalities, where m is the cardinality of the given set of points. The system is solved by the Fourier elimination algorithm. One can also apply Fukuda's CDD algorithm for solving systems of linear inequalities by successive intersection of half-spaces defined by the inequalities. An efficient incremental algorithm based on a similar approach is proposed in [15]. In [8] Buzer presents an incremental linear time algorithm based on solving a linear program by appropriate modification of Megiddo's algorithm [18]. Most of the above-mentioned algorithms perform well in practice. However, with a few exceptions (e.g., [8]), rigorous time complexity analysis is not available.

In the present theoretical work we consider somewhat more general version of the DPS recognition problem: Given a set of points $M = \{p^1, p^2, \ldots, p^m\} \subseteq \mathbb{R}^n$, decide whether M is a portion of a digital hyperplane and, if so, determine that analytical digital hyperplane. Here $p^1, p^2, \ldots, p^m$ may be *arbitrary* points, possibly with integer and/or irrational coefficients. Such kind of data may result, e.g., from certain computational processes. The considerations take place in an *arbitrary dimension* n, provided that n is *fixed* (i.e., bounded by an arbitrary constant). We provide an algorithm that solves the above problem by reducing it to an integer linear programming problem of a fixed dimension within an algebraic model of computation. This last problem is solved by a (theoretically) efficient algorithm based on a number of well-known results from theory of algorithms and complexity (some of them earlier authors' contributions). The algorithm works on input data that are *arbitrary* real numbers. In particular, it applies to problems with integer or rational data. Our algorithm solves the problem with $O(m \log D)$ arithmetic operations, where D is a bound on the norm of the domain elements. The obtained theoretical results are somewhat in the spirit of Buzer's results [8] (first reported at DGCI'02).

To our knowledge of the available literature, this is the first integer programming based algorithm for a DPS recognition problem. The reason for absence of other similar methods is that ILP was believed to be inapplicable to DPS recognition due to its NP-hardness (see, e.g., related discussion in [8]). The present paper illustrates that from a theoretical point of view, for fixed dimensions, an integer linear program is almost as easy to solve as a linear program. Moreover, in some cases the proposed integer programming approach may have certain advantages over a linear programming approach, especially in avoiding very large integers that may result from a LP formulation. It also seems to us that our algorithm is the first one for DPS in *higher* dimensions, whose description is accompanied with rigorous complexity analysis. Another purpose of this work is to demonstrate the wealth of applying knowledge and results from other branches of theoretical computer science (such as theory of algorithms and complexity) to problems of digital geometry.

The paper is organized as follows. In Section 2 we recall some basic definitions from the theory of arithmetic planes and obtain the integer linear program corresponding to the considered problem. In Section 3 we present an integer programming algorithm that solves any integer program of the considered type. We conclude with some remarks in Section 4.

2 Feasible Digital Plane Recognition

In order to make our further considerations clearer, we first consider the 2D version of the DPS recognition problem, that is, a digital line segment recognition.

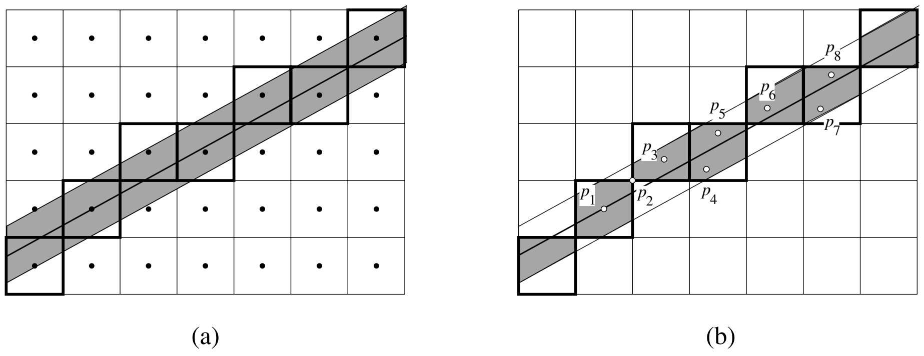

Fig. 1. Illustrations to the notions of feasibility. a) Feasible region related to a digital line. b) Feasible parts of pixels forming the feasible set of a digital line

Here we are given a set $M = \{p^1, p^2, \dots, p^m\}$ of *integer* points in the plane, and we look for a digital line that contains these points.

Several equivalent definitions of a digital line are known (see the survey by Rosenfeld and Klette [22].) Here we conform to the analytical definition proposed by Reveillès [21].

A (naive) *digital line*[1] is a set of pixels $L(a_1, a_2, b, \max(|a_1|, |a_2|)) = \{(x_1, x_2) \in \mathbb{Z}^2 | 0 \leq a_1x_1 + a_2x_2 + b + \lfloor \max(|a_1|, |a_2|)/2 \rfloor < \max(|a_1|, |a_2|)\}$, where $a_1, a_2, \mu \in \mathbb{Z}$. $L(a_1, a_2, b, \max(|a_1|, |a_2|))$ can be considered as a discretization of a straight line with equation $ax_1 + ax_2 + b = 0$. It involves all pixels (unit squares centered at integer points of the plane) whose centers fall in between two parallel boundary straight lines $a_1x_1 + a_2x_2 + b + \lfloor \max(|a_1|, |a_2|)/2 \rfloor = 0$ and $a_1x_1 + a_2x_2 + b + \lfloor \max(|a_1|, |a_2|)/2 \rfloor = \max(|a_1|, |a_2|)$.[2] We will call the strip $F(a_1, a_2, b) = \{(x_1, x_2) \in \mathbb{R}^2 | 0 \leq a_1x_1 + a_2x_2 + b + \lfloor \max(|a_1|, |a_2|)/2 \rfloor < \max(|a_1|, |a_2|)\}$ a *feasible region* of $\mathbb{R}^2$ relative to $L(a_1, a_2, b, \max(|a_1|, |a_2|))$. See Fig. 1a.

Now consider a pixel $p \in L(a_1, a_2, b, \max(|a_1|, |a_2|))$. As Fig. 1b suggests, a part of p is inside the feasible region $F(a_1, a_2, b)$, while the rest of it is outside $F(a_1, a_2, b)$. The former will be called the *feasible part* of p relative to the line $L(a_1, a_2, b, \max(|a_1|, |a_2|))$ and denoted $F_{a_1,a_2,b}(p)$. The points of $F_{a_1,a_2,b}(p)$ will be referred to as *feasible points* of p. Finally, the union of all feasible parts of all pixels in a segment of a digital line L will be called the *feasible set* of the digital line segment and denoted $F_L(a_1, a_2, b)$ (see Fig. 1b).

All above definitions and notions trivially extend to arbitrary dimension n. Thus a (naive) *digital hyperplane* is a set of n-cells[3]

[1] also called "arithmetic line."
[2] Because of the strict right inequality in the definition, pixels' centers cannot lie on the second line.
[3] n-dimensional counterparts of pixels.

$$H(a_1, a_2, \ldots, a_n, b, |a|_{\max})$$
$$= \left\{ (x_1, x_2, \ldots, x_n) \in \mathbb{Z}^n | 0 \le a_1x_1 + a_2x_2 + \ldots + a_nx_n + b + \left\lfloor \frac{|a|_{\max}}{2} \right\rfloor < |a|_{\max} \right\},$$

where $|a|_{\max} = \max(|a_1|, |a_2|, \ldots, |a_n|)$. (See [1, 2] for basic definitions and facts and [4] for further studies.) Its feasible region is

$$F(a_1, a_2, \ldots, a_n, b)$$
$$= \left\{ (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n | 0 \le a_1x_1 + a_2x_2 + \ldots + a_nx_n + b + \left\lfloor \frac{|a|_{\max}}{2} \right\rfloor < |a|_{\max} \right\}.$$

A feasible part $F_{a_1,a_2,\ldots,a_n,b}(p)$ of an n-cell p and a feasible set$F_H(a_1, a_2, \ldots, a_n, b)$ of a digital hyperplane H are defined analogously to the 2D case.

With this preparation, we are able to state the following generalization of a digital hyperplane segment recognition problem, which we call the *feasible digital hyperplane segment* recognition problem and abbreviate FeasDHS.

FeasDHS Recognition:

Given a set of points $M = \{p^1, p^2, \ldots, p^m\} \subseteq \mathbb{R}^n$, decide whether M is included in the feasible part $F_H(a_1, a_2, \ldots, a_n, b)$ of some digital hyperplane $H(a_1, a_2, \ldots, a_n, b, |a|_{\max})$, and, if so, determine its coefficients $a_1, a_2, \ldots, a_n, b$.

Note that in this setting more than one point p^i may belong to the same pixel of the discrete space. Moreover, a point p^i may have irrational coordinates, such as the point p^2 in Fig. 1b.

We now obtain formulation of FeasDHS in terms of an integer programming program.

It is not hard to realize that an element p^i of M is a feasible point of some n-cell v (i.e., $p^i \in F_{a_1,a_2,\ldots,a_n,b}(v)$) if and only if there exist integers $a_1, a_2, \ldots, a_n$, and b, such that the following conditions are met:

1. $0 \le a_1p_1^i + a_2p_2^i + \ldots + a_np_n^i + b + \left\lfloor \frac{|a|_{\max}}{2} \right\rfloor < |a|_{\max}$, and
2. $0 \le a_1 \left\lceil p_1^i \right\rfloor + a_2 \left\lceil p_2^i \right\rfloor + \ldots + a_n \left\lceil p_n^i \right\rfloor + b + \left\lfloor \frac{|a|_{\max}}{2} \right\rfloor < |a|_{\max}$.

   ($\lceil . \rfloor$ denotes the operator “the closest integer” to a given real number. If x is a “half-integer”, we set $\lceil x \rfloor = \lceil x \rceil$, e.g., $\lceil 3.5 \rceil = 4$.)

The first condition causes p^i to belong to the feasible region relative to a digital hyperplane with coefficients $a_1, a_2, \ldots, a_n$ and b, while the second one ensures that p^i belongs to an n-cell from the same digital hyperplane. Note that both conditions are essential: If Condition 1 is missing, then p^i may be outside the feasible region. If Condition 2 does not hold, then p^i may not belong to all n-cells from the digital hyperplane with coefficients $a_1, a_2, \ldots, a_n, b$.

When i runs from 1 to m, we get an integer linear problem with $n + 1$ unknowns $a_1, a_2, \ldots, a_n, b$ and $4m$ linear constraints.

As already mentioned, we will deal with the case when the dimension n is an arbitrary fixed integer. We will also suppose that the coefficients $a_1, a_2, \ldots, a_n, b$

we look for are bounded in size, i.e. $|a_1| \leq d_1, |a_2| \leq d_2, \ldots, |a_n| \leq d_n, |b| \leq d_{n+1}$, as the bounds $d_1, d_2, \ldots, d_{n+1}$ are a part of the problem input. In the next section we will see that this condition is dictated by the very nature of the problem, especially by the fact that some of the coefficients may be irrational numbers. From a practical point of view, this condition does not restrict the generality, as we can always suppose that the absolute value of the largest coefficient is bounded by, e.g., the largest positive integer that we may use in practice. Moreover, by assuming bounds on the plane coefficients one can avoid occurrance of very large numbers in the problem solution.

3 Algorithm for Integer Programming of Fixed Dimension

In this section we describe an efficient algorithm for integer linear programming programs as those corresponding to FeasDHS. Consider the following integer linear program:[4]

(ILP) Given a matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m, d \in \mathbb{R}^n$,
find $x \in \mathbb{Z}^n$ such that $Ax \leq b$, where $\mathbf{0} \leq x \leq d$.

To simplify our further considerations, we have assumed that the coordinates of a domain element $x = (x_1, \ldots, x_n)$ satisfy the conditions $0 \leq x_i \leq d_i$ rather than $|x_i| \leq d_i$, $1 \leq i \leq n$. Clearly, a problem with constraints of the first type is equivalent to one with constraints of the second type up to a change of the variables.

The input entries are arbitrary real numbers and the adopted model of computation is an algebraic computation model. This kind of model has been traditionally used in scientific computing, algebraic complexity, computational geometry, and (although not explicitly) numerical analysis (see, e.g., [19, 20, 25]). In that model, the assumption is that all the real numbers in the input have unit size, and the basic algebraic operations $+, -, *, /$ and the relation $\leq$ are executable at unit cost. Thus the algebraic complexity of a computation on a problem instance is the number of operations and branchings performed to solve the instance.

At this point it is important to mention that the requirement in the ILP formulation for bounded domain (i.e., $\mathbf{0} \leq x \leq d$) is essential and predetermined by the intrinsic nature of the problem, namely by the fact that the coefficients may be *irrational* numbers. In such a case, a problem with unbounded domain may be, in general, *undecidable*, as shown in [6].

In the rest of this paper we present an algorithm for ILP when the value of n is fixed. The algorithm consists of two stages: a reduction of the given real

[4] In the feasDHS formulation we have certain rounding operations. It is well-known [3] that rounding of a real number x can be performed in $\log |x|$ basic arithmetic operations. Thus the coefficients of the second inequality in the feasDHS definition can be computed in $O(m \log |x|_{\max})$ time overall, where $|x|_{\max} = \max(|x_1|, |x_2|, \ldots, |x_n|)$.

input to an integer input determining the same admissible set, followed by an application of Lenstra's algorithm [16]. The first stage involves simultaneous Diophantine approximation techniques, while the second employs two well-known algorithms: the Lovász' basis reduction algorithm [17] and the Hermite normal form algorithm (see, e.g., [13]).

3.1 Subroutines to the Main Algorithm

Lovász Lattice Basis Reduction Algorithm. The input to Lovász algorithm consists of linearly independent vectors $b_1, b_2, \dots b_n \in \mathbb{Q}^n$, considered as a basis for a lattice L. The algorithm transforms them iteratively. At the end, they form a basis for L which is reduced in the Lovász sense. First we recall some definitions, then describe the Lovász lattice basis reduction algorithm itself, following [11].

With a basis $b_1, b_2, \dots b_n$, we associate the orthogonal system $b_1^*, b_2^*, \dots b_n^*$, where b_i^* is the component of b_i which is orthogonal to $b_1, b_2, \dots b_{i-1}$. The vectors $b_1^*, b_2^*, \dots b_n^*$ can be computed by Gram-Schmidt orthogonalization:

$$b_1^* = b_1,\ \ b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*,\ \ 2 \le i \le n,\ \mu_{i,j} = \langle b_i, b_j^* \rangle \Big/ \left\| b_j^* \right\|^2 .$$

The basis $b_1, b_2, \dots b_n$ is *size-reduced* if all $|\mu_{i,j}| \le \frac{1}{2}$. Given an arbitrary basis $b_1, b_2, \dots b_n$, we can transform it into a size-reduced basis with the same Gram-Schmidt orthogonal system, as follows:

For every i from 2 to n; for every j from $i-1$ to 1;
Set $b_i := b_i - \lceil \mu_{i,j} \rfloor b_j$ and update $\mu_{i,k}$ for $1 \le k \le i-1$, by setting $\mu_{i,k} = \mu_{i,k} - \lceil \mu_{i,j} \rfloor \mu_{j,k}$.

We outline a variant of the Lovász lattice basis reduction algorithm next.

1. *Initiation.* Compute the Gram-Schmidt quantities $\mu_{i,j}$ and b_i^* for $1 \le j < i \le n$. Size-reduce the basis.
2. *Termination condition.* If $\|b_i^*\|^2 \le 2 \left\|b_{i+1}^*\right\|^2$ for $1 \le i \le n-1$, then stop.
3. *Exchange step.* Choose the smallest i such that $\|b_i^*\|^2 > 2 \left\|b_{i+1}^*\right\|^2$. Exchange b_i and b_{i+1}. Update the Gram-Schmidt quantities. Size-reduce the basis. Go to 2.

Gram-Schmidt quantities in Step 3 are updated as follows:

$$\|b_i^*\|_{new}^2 = \left\|b_{i+1}^*\right\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2,\ \ \left\|b_{i+1}^*\right\|_{new}^2 = \|b_i^*\|^2 \left\|b_{i+1}^*\right\|^2 \Big/ \|b_i^*\|_{new}^2$$

$$\mu_{i+1,i}^{new} = \mu_{i+1,i} \|b_i^*\|^2 \Big/ \|b_i^*\|_{new}^2$$

$$\begin{pmatrix} \mu_{i,j}^{new} \\ \mu_{i+1,j}^{new} \end{pmatrix} = \begin{pmatrix} \mu_{i+1,j} \\ \mu_{i,j} \end{pmatrix} \text{ for } 1 \le j \le i-1$$

$$\begin{pmatrix} \mu_{j,i}^{new} \\ \mu_{j,i+1}^{new} \end{pmatrix} = \begin{pmatrix} 1 & \mu_{i+1,i}^{new} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\mu_{i+1,i} \end{pmatrix} \begin{pmatrix} \mu_{j,i} \\ \mu_{j,i+1} \end{pmatrix} \text{ for } i+2 \le j \le n.$$

The other $\|b_i^*\|^2$'s and $\mu_{i,j}$'s do not change.

After termination of the above algorithm, we have a size-reduced basis for which $\|b_i^*\|^2 \le 2 \left\|b_{i+1}^*\right\|^2$, $1 \le i \le n-1$. We call such a basis *reduced in the Lovász sense.* The following lemma was proved in [7].

Lemma 1. *The algebraic complexity of Lovász' basis reduction algorithm applied to an $n \times n$ rational matrix with entries of size $O(S)$, is $O(Sn^5 \log n)$, and the bit-size of the entries in the reduced basis is $O(Sn^3)$.*

Hermite Normal Form Algorithm. In the algorithm's description we follow [23]. The input for the algorithm is an $m \times n$ ($m \leq n$) integer matrix A of full rank. The algorithm uses a matrix of the form

$$A' = \left(A \,\middle|\, \begin{matrix} M & & \\ & \ddots & \\ & & M \end{matrix} \right),$$

where M is the absolute value of some nonsingular $m \times m$ minor of A. A' has the same Hermite normal form as A. The algorithm consists of the following five steps:

1. Cause all the entries of the matrix A to fall into the interval $[0, M)$, by adding to the first n columns of A' proper integer multiples of the last n columns;
2. For k from 1 to m do 3-4;
3. If there are $i \neq j$, $k \leq i, j \leq n + k$, such that $a'_{k,i} \geq a'_{k,j} > 0$, then subtract from the ith column the jth one multiplied by $\left\lfloor \frac{a'_{k,i}}{a'_{k,j}} \right\rfloor$. Then reduce the ith column modulo M. Go to 3;
4. Exchange the kth column and the only column with $a'_{k,i} > 0$;
5. For every i from 2 to n; for every j from 1 to $i - 1$, add an integer multiple of the ith column to the jth one, to get $a'_{i,i} > a'_{i,j} \geq 0$.

We have the following lemma [7].

Lemma 2. *Let A be an $m \times n$ ($m \leq n$) integer matrix of full rank with entries of size $O(S)$. Then the algebraic complexity of the Hermite normal form algorithm that reduces A into its Hermite normal form, is $O(m^2 n(\log m + S))$, and the bit-size of all resulting integers is $O(Smn)$.*

Since the above lemma admits a short proof, we sketch it next in order to provide the reader with an idea how statements of this kind can be demonstrated.

We introduce the function

$$F\left(a'_{k,k}, a'_{k,k+1}, \ldots a'_{k,n+k}\right) := \prod_{k \leq i \leq n+k} a'_{k,i}$$

for $a'_{k,i} > 0$. After one iteration of Step 3, we have

$$F_{new}/F = (a'_{k,i} - \lfloor a'_{k,i}/a'_{k,j} \rfloor a'_{k,j})/a'_{k,i},$$

which implies both $F_{new}/F < 1/2$ and $F_{new}/F < a'_{k,j}/a'_{k,i}$. It is not hard to see that one iteration of Step 3 can be performed in time $O\left(m \log(a'_{k,i}/a'_{k,j})\right) =$

$O\left(m\log(F/F_{new})\right)$. So, Step 3 takes $O\left(m\log\frac{F_{start}}{F_{end}}\right)$ time. We have that $F_{start} < M^{n+1}$, $F_{end} \geq 1$. Moreover, we have the following simple fact: if a is a non-zero rational number of bit-size at most S, then $1\big/2^S \leq |a| \leq 2^S$. This last fact implies the following property of matrices: given a non-singular $n\times n$ rational matrix B whose entries are of bit-size at most S, then $1\Big/2^{n^2S} \leq |\det(B)| \leq n!2^{nS}$. From here we obtain $M = O\left(m!2^{mS}\right)$. Hence, the overall running time of Step 3 is $O\left(nm\left(\log m + S\right)\right)$. Then, the complexity of the Hermite normal form algorithm is $O\left(nm^2\left(\log m + S\right)\right)$. Since all the resulting integers are smaller than M, their bit-size is $O\left(Smn\right)$.

3.2 Simultaneous Diophantine Approximation

Our algorithm employs in one of its steps the well-known algorithm for finding a simultaneous Diophantine approximation to a given rational vector. Specifically, we will use the following lemma.

Lemma 3. *(see, e.g., [23–Corollary 6.4c]) There exists a polynomial algorithm which, given a vector $a \in \mathbb{Q}^n$ and a rational number ε, $0 < \varepsilon < 1$, finds an integral vector p and an integer q such that $||a - \frac{1}{q}p|| < \varepsilon/q$, and $1 \leq q \leq 2^{n(n+1)/4}\varepsilon^{-n}$.*

We will also need an algorithm that reduces the constraints with real coefficients to constraints with integer coefficients, determining the same admissible set. The first phase of this reduction is a substitution of a given *real* vector with an appropriate *rational* vector, justified by the following lemma.

Lemma 4. *Given a vector $\alpha \in \mathbb{R}^n$ with $|\alpha_j| \leq 1, j = 1, 2, \ldots, n$, and $D \in \mathbb{Z}_+$, there exists an $O(n^4 \log n(n + \log D))$ algorithm that finds $p \in \mathbb{Z}^n$ and $q \in \mathbb{Z}_+$ such that $|\alpha_j - p_j/q| < 1/(qD)$, $j = 1, 2, \ldots, n$, and $1 \leq q \leq \lceil 2^{n(n+5)/4}D^n\rceil$.*

The required $p \in \mathbb{Z}^n$ and $q \in \mathbb{Z}_+$ can be found as follows.

Diophantine Approximation to a Real Vector

1. For each α_j, $1 \leq j \leq n$, find the closest rational fraction a_j with denominator $G = \lceil 2^{n(n+5)/4}D^{n+1}\rceil$.
2. Apply the algorithm of Lemma 3 with input $a = (a_1, \ldots, a_n) \in \mathbb{Q}^n$ and $\varepsilon = 1/(2D)$. □

By Lemma 3, the output is a vector $p \in \mathbb{Z}^n$ and an integer $q \in \mathbb{Z}_+$ with

$$||a - (1/q)p|| < 1/(2qD) \text{ and } 1 \leq q \leq \lceil 2^{n(n+5)/4}D^n\rceil.$$

Clearly, $|\alpha_j - a_j| \leq 1/(2G)$. Then we have

$$\left|\alpha_j - \frac{p_j}{q}\right| \leq |\alpha_j - a_j| + \left|a_j - \frac{1}{q}p_j\right|$$

$$\leq |\alpha_j - a_j| + \left\|a - \frac{1}{q}p\right\| < \frac{1}{2G} + \frac{1}{2qD}$$

$$\leq \frac{1}{2.\lceil 2^{n(n+5)/4} D^n \rceil .D} + \frac{1}{2qD} \leq \frac{1}{qD},$$

i.e., the obtained vector p and integer q are as desired.

Consider first Step 1. For a given real number α_j, the closest rational fraction with denominator $G = \lceil 2^{n(n+5)/4} D^{n+1} \rceil$ can be found in time $O(\log G) = O(n^2 + n \log D)$. Thus the overall time complexity of Step 1 is $O(n^3 + n^2 \log D)$.

Step 2 involves the simultaneous Diophantine approximation algorithm applied to the particular class of inputs $a \in \mathbb{Q}^n$, $\varepsilon = 1/(2D)$ obtained in Step 1. As a matter of fact, this is a specialization of the Lovász basis reduction algorithm, applied to a certain matrix. It has been proved in [6–Lemma 4.4] that the *number of iterations* performed in this step is $O(n^4 \log n(n + \log D))$. Then the overall time complexity of the algorithm of Lemma 4 is $O(n^4 \log n(n + \log D))$, as well.

The algorithm of Lemma 4 can be used to substitute any *real* constraint $ax \leq b$ with an *integer* one, preserving the same admissible integer points x with $\mathbf{0} \leq x \leq d$, $d \in \mathbb{R}^n$. More precisely, we have the following lemma.

Lemma 5. *Let $T = \{x \in \mathbb{Z}^n : ax \leq b; \mathbf{0} \leq x \leq d\}$, where $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, $d \in \mathbb{Z}^n_+$. Then there exists an algorithm which finds a vector $r \in \mathbb{Z}^n$ and a number $r_0 \in \mathbb{Z}$ such that $T = \{x \in \mathbb{Z}^n : rx \leq r_0; \mathbf{0} \leq x \leq d\}$. The algorithm involves at most n applications of the algorithm from Lemma 4, with $D = ||d||$.*

Proof of the above fact is available in [6–Lemma 5.1]. Now we are able to complete the algebraic complexity analysis of integer programming of fixed dimension, which we do in the next section.

3.3 Algorithm for ILP

In this section we use the results from the previous section to obtain an $O(m \log D)$ algorithm for ILP, where $D = ||d||$, as defined in Lemma 5.

As already mentioned, the algorithm consists of two stages. In the *first stage*, it reduces the constraints with real coefficients to constraints with integer coefficients which determine the same admissible set of integer points. In the *second stage*, the Lenstra's algorithm [16] is applied to the integer data problem obtained as an output of the first stage.

From Lemmas 4 and 5, we obtain that the overall time complexity of the reduction stage is $O(mn^5 \log n(n + \log D))$. Furthermore, the bit-size of the generated integers is $O(n^2(n + \log D))$. Therefore, the overall bit-size of the reduced problem is $O(mn^3(n + \log D))$.

We now complete the complexity analysis of the second stage of the algorithm. That stage is an application of the Lenstra's [16] algorithm to the integer linear problem obtained as output of the first stage. A recursive step of Lenstra's algorithm reduces an n-dimensional problem to a set of subproblems of dimension $n - 1$, whose number is exponential but depending only on n. The basic algorithms used in this reduction are the Lovász basis reduction algorithm and the Hermite normal form algorithm. In addition, in order to compute a homothetic approximation to the underlying polyhedron with constant homothety ratio, a number of linear programming problems of dimension $(m + 2n) \times n$ have to be solved.

The Lovász basis reduction algorithm and the Hermite normal form algorithm are both applied to matrices of dimension depending only on n. Moreover, all entries of these matrices are of bit-size $O(\log D)$, as the value of n is fixed. Then, by Lemmas 1 and 2, the complexity of the two algorithms as well as the bit-size of the integers they generate, are bounded by $O(\log D)$.

During the execution of the Lenstra's algorithm, there are $O(\log D)$ linear programming problems to be solved. Each of them can be solved in time $O(m+n)$ (i.e., *linear* in m) using the well-known Megiddo's algorithm [18]. Hence, if n is fixed, the overall complexity of this stage is $O(m \log D)$. This completes the proof of the following theorem.

Theorem 1. *There is an $O(m \log D)$ algorithm for ILP with a fixed number of variables, where $D = ||d||$.*

3.4 Theoretical Versus Practical Efficiency

The proposed algorithm solves the considered problem ILP within an algebraic computation model by performing $O(m \log D) = O(m \log ||d||)$ arithmetic operations for any fixed dimension n. Usually, algorithms of such kind of complexity are considered as theoretically efficient. However, in order to make a reasonable foresight about the practical efficiency of the computation, one also has to evaluate the implicit constant hidden in the big-O notation.

Specifically, in order to solve an ILP (and thus the original hyperplane recognition problem) the algorithm uses as subroutines a number of well-known algorithms for some basic combinatorial problems. Keeping in mind the algorithm description, it is not hard to realize that the overall number of these problems is exponential in n. In practice, however, it might not be a problem for two reasons. First, n is a constant (usually a small one) and, second, the average-case time complexity of ILP is believed to be much lower than the worst-case time complexity. Moreover, the Lovász lattice basis reduction algorithm and the Hermite normal form algorithm are polynomial in n and therefore very efficient even for relatively large dimensions. The only exception is the Megiddo's linear programming algorithm, whose time complexity involves an implicit constant factor of the order $\Omega(2^{n^2})$. For relatively small dimensions Megiddo's algorithm is known to perform well in practice. For moderately large dimensions one can use instead of Megiddo's algorithm some recent more practical algorithms that have better theoretical running time,[5] are easier to implement, and perform well in practice.

For large dimensions n, however, the algorithm is clearly inefficient, like all other algorithms involving the Megiddo's method (in particular, Buzer's digital plane recognition algorithm mentioned above). Nevertheless, results of this kind

[5] For example, [12] provides a randomized linear programming algorithm whose running time involves an implicit constant factor that is subexponential in n.

provide useful insight on certain limitations that an efficient computation may feature.

The ultimate test for our algorithm is, of course, an efficient implementation that would allow us to run it on real data and compare it with the existing algorithms. We see this as an important direction for future research.

4 Concluding Remarks

We have presented an $O(m \log D)$ algorithm for solving the digital hyperplane segment recognition problem in arbitrary fixed dimension, where $D = ||d||$ is a bound on the norm of the domain elements (possible hyperplane coefficients). The input may be a set of points with arbitrary real coordinates. The algorithm also applies to classical digital plane recognition where the given points have integer coefficients.

The algorithm works on an integer linear program formulation and solves it theoretically efficiently. We believe that this result, together with some other theoretical results, will contribute to the better understanding structural, algorithmic, and complexity issues of digital plane recognition.

Acknowledgements

The authors thank the two anonymous referees for their useful remarks and suggestions.

References

1. Andres, E., *Modélisation Analytique Discrète d'Objets Géométriques*, Thèse de habilitation à diriger des recherches, Universit''e de Poitiers, Poitiers, France, 2001
2. Andres, E., R. Acharya, C. Sibata, Discrete analytical hyperplanes, *Graphical Models Image Processing* **59**, 302–309 (1997)
3. Blum, L., M. Shub, S. Smale, On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness, Recursive Functions and Universal Machines, *Bull. Amer. Math. Soc.* (NS) **21**, 1–46 (1989)
4. Brimkov, V.E., E. Andres, R.P. Barneva, Object Discretizations in Higher Dimensions, *Pattern Recognition Letters*, **23**, 623–636 (2002)
5. Brimkov, V.E., D. Coeurjolly, R. Klette, Digital Planarity - A Review, CITR-TR 142, 2004
6. Brimkov, V.E., S.S. Danchev, Real Data – Integer Solution Problems within the Blum-Shub-Smale Computational Model, *J. of Complexity* **13**, 279–300 (1997)
7. Brimkov, V.E., S.S. Dantchev, On the Complexity of Integer Programming in the Blum-Shub-Smale Computational Model, In: Theoretical Computer Science. Exploring New Frontiers of Theoretical Informatics, van Leeuwen, J., O. Watanabe, M. Hagiya, P.D. Mosses, T. Ito (Eds.), LNCS-1872, 286-300 (2000)
8. Buzer, L., A Linear Incremental Algorithm for Naive and Standard Digital Dines and Planes Recognition, *Graphical Models* **65** 61–76 (2003)

9. Debled-Rennesson, I., J.-P. Reveillès, A New Approach to Digital Planes, *Vision Geometry III*, SPIE-2356, 12–21 (1994)
10. Françon, J., J.M. Schramm, M. Tajine, Recognizing Arithmetic Straight Lines and Planes, 6th Int. Conf. *Discrete Geometry for Computer Imagery*, Springer, LNCS-1176, 141–150 (1996)
11. Hastad, J., B. Just, J.C. Lagarias, C.P. Schnoor, Polynomial Time Algorithms for Finding Integer Relations among Real Numbers, *SIAM J. Comput.* **18**, 859–881 (1989)
12. Kalai, G., A Subexponential Randomized Simplex Algorithm, *24th Annual ACM Symposium on the Theory of Computation*, ACM Press, 475-482 (1992)
13. Kannan, R., A. Bachem, Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix, *SIAM J. Comput.* **8**, 499–507 (1979)
14. Klette, R., I. Stojmenović, J. Žunić, A Parametrization of Digital Planes by Least Square Fits and Generalizations, *Graphical Models Image Processing* **58**, 295–300 (1996)
15. Klette, R., H.-J. Sun, Digital Planar Segment Based Polyhedrization for Surface Area Estimation, In: Arcelli, C., L.P. Cordella, and G. Sanniti di Baja, editors, *Visual Form 2001*, Springer, Berlin, pages 356–366 (2001)
16. Lenstra, H.W., Jr., Integer Programming with a Fixed Number of Variables, *Math. Oper. Res.* **8**, 538–548 (1983)
17. Lenstra, A.K., H.W. Lenstra, Jr., L. Lovász, Factoring Polynomials with Rational Coefficients, *Math. Ann.* **261**, 515–534 (1982)
18. Megiddo, N., Linear Programming in Linear Time when the Dimension is Fixed, *J. of ACM* **31** (1), 114–127 (1984)
19. Novak, E., The Real Number Model in Numerical Analysis, *J. of Complexity* **11**, 57–73 (1994)
20. Preparata, F.P., M.I. Shamos, *Computational Geometry*, Springer-Verlag, Berlin Heidelberg New York, 1985
21. Reveillès, J.-P., Géométrie Discrète, Calcul en Nombres Entiers et Algorithmique, Thèse d'état, Univ. Louis Pasteur, Strasbourg, 1991
22. Rosenfeld, A., R. Klette, Digital Straightness, In: *Electronic Notes in Theoretical Computer Science* **46** (2001)
23. Schrijver, A., *Theory of Linear and Integer Programming*, Wiley, Chichester New York Brisbane Toronto Singapore, 1986
24. Stojmenović, I., R. Tosić, Digitization Schemes and the Recognition of Digital Straight Lines, Hyperplanes and Flats in Arbitrary Dimensions, *Vision Geometry, Contemporary Mathematics Series*, **119** 197–212 (1991)
25. Strassen, V., Algebraic Complexity Theory, In: van Leeuwen, J. (Ed.), *Handbook of Theoretical Computer Science*, Vol. A, Elsevier, Amsterdam, 633–672 (1990)
26. Veelaert, P., Digital Planarity of Rectangular Surface Segments, *IEEE Pattern Analysis and Machine Int*, **16**, 647–652 (1994)
27. Vittone, J., J.-M. Chassery, Recognition of Digital Naive Planes and Polyhedrization, 9th Int. Conf. *Discrete Geometry for Computer Imagery*, Springer, LNCS-1953, 296–307 (2000)