Recognition of arithmetic line segments and hyperplanes using the Stern-Brocot tree

Bastien Laboureix, Isabelle Debled-Rennesson

Recognition of arithmetic line segments and hyperplanes using the Stern-Brocot tree

Bastien Laboureix[1] and Isabelle Debled-Rennesson[1]

Université de Lorraine, CNRS, LORIA, 54000 Nancy, France
bastien.laboureix@loria.fr
isabelle.debled-rennesson@loria.fr

Abstract. The classic problem of discrete structure recognition is revisited in this article. We focus on naive digital straight segments (DSS) and, more generally, naive arithmetic hyperplanes, and we present a new approach to recognise these discrete structures based on the Stern-Brocot tree. The algorithm for DSS recognition proposes an alternative method to the state of the art, keeping the linear complexity and incremental character. While most of the concepts can be generalised to planes in dimension 3 and hyperplanes in higher dimensions, certain points in the process of descending in the Stern-Brocot tree need to be explored further. The proposed algorithm calculates separating chords characterising the membership of planes to cones generated by the branch of the Stern-Brocot tree. This generalisation shows the close link between arithmetic hyperplanes and the generalised Stern-Brocot tree and opens up interesting perspectives for the recognition of pieces of arithmetic hyperplanes.

Keywords: digital straight segments, arithmetic hyperplanes, recognition algorithm, Stern-Brocot tree

1 Introduction

Discrete geometry is concerned with various structures of space $\mathbb{Z}^d$. This article focuses on arithmetic discrete lines, introduced in [16] and their generalisation to any dimension: arithmetic hyperplanes [1]. Our problem is to decide, given a finite subset S of $\mathbb{Z}^d$, whether S is a piece of arithmetic hyperplane and, if so, to compute its minimal parameters.

In dimension 2, the problem of naive digital straight segment (DSS) recognition was first addressed from the point of view of symbolic dynamics and the study of Sturmian words: see [15] for the initial article, [3] for a review of the links between discrete geometry and symbolic dynamics. A history of the DSS recognition problem can be found in [11]. The arithmetic definition of discrete lines and their geometric structures were used to obtain the incremental and linear algorithm for DSS recognition presented by I. Debled-Rennesson and J.-P. Réveillès in [8].

While the hyperplane recognition problem is largely solved in dimension 2, the problem remains difficult in dimensions 3 and higher. In dimension 3, numerous studies of discrete planes have been carried out using different approaches

(see the survey [4]). A generalisation of I. Debled-Rennesson's 2D algorithm was presented in 1994 in [9,14], but the algorithm, restricted to pieces of rectangular planes, loses in simplicity and the number of cases to be processed rapidly explodes. In 2005, Y.Gerard et al's algorithm [10] solves the recognition problem for any finite set of points and in any dimension, using the properties of the convex hull of the chord space. However, the algorithm announces a high complexity for dimension 3 and does not guarantee to obtain the minimum parameters. In 2008, the Charrier et al's algorithm ([6]) proposed a linear optimisation approach to the problem. This method provides a quasi-linear algorithm in terms of the number of points, but does not allow the minimal parameters of the hyperplane to be obtained (the complexity would then revert to that of the simplex algorithm: exponential). Finally, a serie of articles has been written on plane probing ([12] for the first version), which makes it possible to obtain the characteristics of the plane from successive oracles.

In this paper, we propose another DSS recognition algorithm based on the Stern-Brocot tree (introduced in [17] and [5]). The successive points of the segment are then taken into account, allowing us to go down the tree until we find the slope corresponding to the minimal parameters of the segment. While I. Debled-Rennesson's algorithm can be interpreted as a descent down the tree (see [7] and [18]), the method proposed in this article differs. We also obtain linear complexity while maintaining the incremental character.

The approach of recognition by descent down the Stern-Brocot tree can also be extended to higher dimensions. In [13], H. Lennerstad generalises the Stern-Brocot tree to any finite dimension d. We introduce a new concept: the notion of separating chord, used to determine the branch of descent down the tree corresponding to a location of the normal vector of the plane to be recognised. Each step of the algorithm now requires $\binom{d}{2}$ tests to determine in which of the $d!$ branches to continue the search. The proposed method recognises discrete hyperplanes and opens up interesting prospects for recognising pieces of discrete planes.

2 Discrete lines

We are working in $\mathbb{R}^d$ with its canonical scalar product. In particular, in this section $d = 2$. In 1991, J.-P. Réveillès defined the concept of a discrete line in [16]. We are interested here in a version of the naive discrete line where the thickness parameter is fixed to guarantee good connectedness properties:

Definition 1 (Naive discrete line) *Let $a, b, \mu \in \mathbb{R}$ with $(a, b) \neq (0, 0)$. The naive discrete line with slope $\frac{a}{b}$ (we agree that a slope $\frac{1}{0}$ is a vertical slope) and shift μ is the set $\mathcal{D}(a, b, \mu) \stackrel{def}{=} \{(x, y) \in \mathbb{Z}^2 \mid 0 \leqslant ax - by + \mu < \|(a, b)\|_\infty\}$*

Definition 2 (8-neighbourhood and 8-connectedness) *We say that 2 points $p, q \in \mathbb{Z}^2$ are 8-neighbours iff $\|p - q\|_\infty = 1$. A set $A \subset \mathbb{Z}^2$ is then 8-connected iff every pair of points of A is connected by a path for the 8-neighbourhood. A*

naive digital straight segment (DSS) is an 8-connected part of a naive line.

Naive lines are 8-connected and minimise thickness for this property ([16]), so they are widely used in discrete geometry. The terms "naive" and "8-connected" will be omitted for convenience. The DSS recognition problem is then to decide, given a set $S \subset \mathbb{Z}^2$, whether or not S is a DSS. Note that there is no uniqueness of the parameters (a, b, μ) of a segment: we therefore ask, if necessary, to calculate the minimal integer parameters of a naive DSS, i.e. those minimising $\|(a, b)\|_\infty$.

I. Debled-Rennesson and J.-P. Réveillès's recognition algorithm presented in [8] proposes a linear algorithm in $|S|$ and incremental, in the sense that adding a point updates the parameters of the segment in $O(1)$. The algorithm is based on the notion of leaning point :

Definition 3 (Leaning points) *Let D be a naive discrete line with parameters (a, b, μ) and $p = (x, y)$ a point of S. We say that p is a lower (resp. upper) leaning point of D iff $ax - by + \mu = \|(a, b)\|_\infty - 1$ (resp. $ax - by + \mu = 0$). In a DSS, It is said to be extremal iff it is the lower (or upper) leaning point with the minimum or maximum abscissa.*

3 Recognition using the Stern-Brocot tree

Our segment recognition algorithm is based on a descent into the Stern-Brocot tree, presented in [17] and [5]. This tree lists all positive irreducible fractions and presents them in tree form (see Figure 1). We define the Stern-Brocot tree B_n truncated at level n by recurrence on n :

- B_0 consists of 2 nodes labelled $\frac{0}{1}$ and $\frac{1}{0}$ called inputs.
- given a truncated tree B_n, we list all the fractions it contains using the prefix depth traversal. Between 2 fractions $\frac{a}{b}$ and $\frac{c}{d}$, we insert into the tree the fraction $\frac{a}{b} \oplus \frac{c}{d} \stackrel{\text{def}}{=} \frac{a+c}{b+d}$.

Remark 4 *This addition of fractions is actually the addition of the pairs (a, b) and (c, d). The irreducible fraction $\frac{a}{b}$ and the pair (a, b) will later be happily confused.*

Theorem 5 (Stern-Brocot [17,5]) *The Stern-Brocot tree obtained by the union of $(B_n)_{n\in\mathbb{N}}$ contains exactly once each irreducible fraction of $\mathbb{Q}_+$.*

The principle of the algorithm is then to find the slope a/b of the segment S in the Stern-Brocot tree. By symmetry and translation, we can always assume that S lies in the first octant (i.e. verifies $0 \leqslant a \leqslant b$) and has point of minimum abscissa $(0, 0)$. Note that, from then on, the abscissas of the points of S are indexed on $[\![0, n-1]\!]$ where $n \stackrel{\text{def}}{=} |S|$.

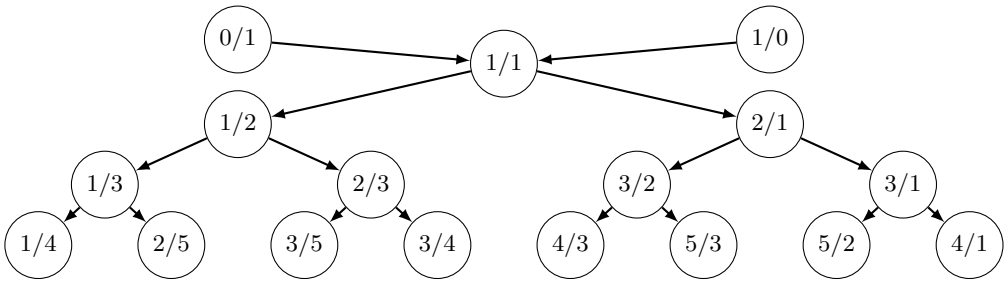

Fig. 1. Stern-Brocot tree

We first propose a naive version of the algorithm, before optimising it. If the segment is neither horizontal nor diagonal (extremal cases), we start with the $1/2$ slope. At each stage of the algorithm, we look at points $p_{\min}$ and $p_{\max}$ of minimum and maximum scalar products for the slope a/b under consideration, and $\delta = (x, y) \stackrel{\text{def}}{=} p_{\max} - p_{\min}$. If $r_{\frac{a}{b}}(x, y) \stackrel{\text{def}}{=} ax - by < b$, the difference between the maximum and minimum scalar products is small enough: the slope is therefore suitable and the shift is calculated using $p_{\min}$. Otherwise, to determine whether the slope is too small or too large, we look at the sign of the δ coordinates. The 2 coordinates have the same sign because S is in the first octant. Note also that, despite the non-uniqueness of $p_{\min}$ and $p_{\max}$, the sign of δ does not depend on the choice of $p_{\min}$ and $p_{\max}$, according to the algorithm's proof of correctness. If this is positive, $p_{\max}$ is to the right of $p_{\min}$: decreasing the difference in the scalar product between $p_{\min}$ and $p_{\max}$ means decreasing the slope, which is done by a descent to the left in Stern-Brocot. Similarly, if the sign of the coordinates of δ is negative, $p_{\max}$ is to the left of $p_{\min}$, so we increase the slope by going to the right in the Stern-Brocot tree. The detailed algorithm can be found in the appendix B.

For example, let's look at the flow of the naive algorithm (see Figure 2) on the segment with parameters $(5, 8, 3)$ and length 11. The algorithm starts with the slope $\frac{1}{2}$ in the Stern-Brocot tree. The minimum $r_{\frac{1}{2}}$ value of -3 is obtained in $(9, 6)$ and the maximum $r_{\frac{1}{2}}$ value of 0 is obtained in $(0, 0)$. So $\delta = (-9, -6)$. Then $r_{\frac{1}{2}}(\delta) = 3$, which is greater than 2, so the slope is unsuitable. As δ has negative coordinates, we increase the slope by going down the right-hand side of the tree to arrive at a slope of $\frac{2}{3}$. For the slope $\frac{2}{3}$, $\delta = (3, 1)$. The slope is still unsuitable ($r_{\frac{2}{3}}(\delta) = 3 \geqslant 3$) and δ is positive, so the slope is too steep: we go down the left-hand side of the tree to $\frac{3}{5}$. For this new slope, $\delta = (-5, -4)$, $r_{\frac{3}{5}}(\delta) = 5 \geqslant 5$. As δ is negative, we increase the slope by going down to the right towards $\frac{5}{8}$. Finally, for $\frac{5}{8}$, $r_{\frac{5}{8}}(\delta) = 7$, which is strictly smaller than 8, so the slope is appropriate. The shift is then the opposite of the minimum $r_{\frac{5}{8}}$ value, so the DSS parameters are $(5, 8, 3)$.

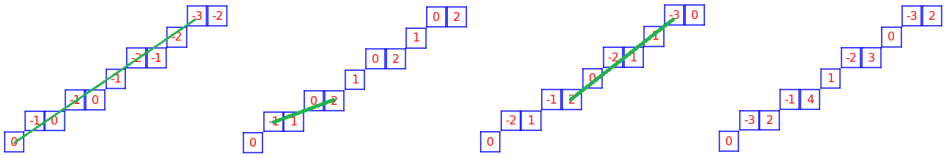

Fig. 2. Flow of the algorithm on the segment with parameters $(5, 8, 3)$ and length 11

Theorem 6 *The naive recognition algorithm decides whether a finite set S of the first octant is a DSS, returns its minimum parameters if it is, and then terminates in $O(nh)$ where $n = |S|$ and h is the height of the parameters of S in the Stern-Brocot tree.*

Sketch of proof : The formal proof is in appendix A. An invariant ensures that the slope of the piece studied is always between the lower and upper slopes given by the Stern-Brocot tree. The sign of the observed chord can then be used to choose the right descent branch in the Stern-Brocot tree.

This naive version is not particularly interesting in terms of complexity and is not an incremental method. However, it is easily optimised (see algorithm 3 for the full detailled incremental algorithm) based on the following observations:

- you can add the points of the segment one by one and calculate the parameters of the new segment by starting again from the old parameters in the Stern-Brocot tree: the method then becomes incremental;
- there is no point in calculating the remainders (i.e. values of $r_{\frac{a}{b}}(M)$) for all the points M in the segment. The remainders associated with the at most 4 extremal upper and lower leaning points (which can be calculated in $O(1)$) are sufficient to guarantee that an equation is satisfied for all points on the segment (by convexity). The most that can be done is to calculate a constant number of remainders in the Extremal-rests function: the call is therefore made in $O(1)$. The complexity therefore drops to $O(n+h) = O(n)$.

The incremental algorithm 3 thus provides an efficient version of segment recognition using the Stern-Brocot tree. However, the advantage of the naive algorithm is that it can be generalised to finite subsets of $\mathbb{Z}^2$ that are not necessarily connected, using the same method of descent in the tree.

4 Stern-Brocot tree in higher dimensions

The notion of a discrete line naturally extends to higher dimensions : The notion of a discrete line extends naturally into higher dimensions, as does the notion of a multidimensional Sturmian word presented in [2]:

Definition 7 (Naive arithmetic hyperplane [1]) *Let $v \in \mathbb{R}^d$ be non-zero and $\mu \in \mathbb{R}$. The naive arithmetic (or discrete) hyperplane with normal vector v and shift μ is the set $\mathbb{P}(v, \mu) \stackrel{def}{=} \{x \in \mathbb{Z}^d \mid 0 \leqslant \langle x, v\rangle + \mu < \|v\|_\infty\}$*

The general problem of plane or hyperplane recognition then consists in deciding whether a finite subset S of $\mathbb{Z}^d$ is part of a naive discrete plane or hyperplane, and computing, if so, its minimal integer parameters i.e. minimising $\|v\|_\infty$. We then propose an algorithm based on an extension of the Stern-Brocot tree in d dimension.

The Stern-Brocot tree naturally extends into 3D and higher dimensions, as presented in [13]. This tree lists the d-uplets of natural integers that are prime to each other in their set. Instead of using 2 end points as before, we use d. These points are represented in $(d-1)$-simplex form as follows (illustrated in Figure 5 (a) and (b) ; with normalisation, the sum of 2 vectors is represented by their barycentric sum) :

- Initially, the d endpoints are the points e_i of the canonical basis of $\mathbb{Z}^d$.
- Given d endpoints $p_1, ..., p_d$ (affinely independent in $\mathbb{R}^{d-1}$ by induction on the tree), we construct the $(d-1)$-simplex Γ with endpoints $p_1, ..., p_d$.
- Given a permutation σ of $\mathfrak{S}_d$, consider the simplex $\Gamma[\sigma]$ formed by the points $q_1, ..., q_d$ where $q_j \overset{\text{def}}{=} \sum_{i=1}^{j} p_{\sigma(i)}$.
- The children of the simplex Γ in the tree are then the simplexes $\Gamma[\sigma]$ for $\sigma \in \mathfrak{S}_d$.

By symmetry, we can assume that the normal vector v of the piece of hyperplane under study has positive coordinates. The d initial ends of the Stern-Brocot tree are then the e_i vectors of the canonical basis. The question is then, given d ends of the tree, to know into which of the $d!$ subtrees to descend. For example, initially, the permutation to choose is the coordinate sorting permutation. For the rest of this paper, we will consider dimension 3, as the results obtained can be generalised without any problem to any finite dimension.

In order to choose the right sub-simplex, we define the notions of chord and separation (by chords) as follows:

Definition 8 (Chord) *Let $v \in \mathbb{Z}^d$ and $\mu \in \mathbb{Z}$. We say that the chord $\delta \in \mathbb{Z}^d$ appears (see Figure 6) in the plane $\mathbb{P}(v, \mu)$ iff there exist $x, y \in \mathbb{P}(v, \mu)$ such that $\delta = y - x$. Note that, by Bézout's theorem (in the rational v case) or by density (in the irrational v case), this definition does not depend on the μ shift considered.*

Remark 9 *Let us note that a chord δ appears in a plane of normal vector v iff there exist $x, y \in \mathbb{P}(v, \mu)$ such that $\delta = y - x$ and $\begin{cases} 0 \leqslant \langle x, v\rangle + \mu < \|v\|_\infty \\ 0 \leqslant \langle y, v\rangle + \mu < \|v\|_\infty \end{cases}$.*
So, by subtracting, δ appears in a plane with normal vector v iff $|\langle \delta, v\rangle| < \|v\|_\infty$.

Definition 10 (Separation (by chord)) *Let $p_1, p_2, p_3 \in \mathbb{N}^3$ be Stern-Brocot endpoints. We say that a pair $(\delta_-^{(i)}, \delta_+^{(i)})$ is a separation (see Figure 5 (c)) for (p_1, p_2, p_3) according to p_i iff for all $v \in \mathcal{C}(p_1, p_2, p_3)$ (convex cone generated by p_1, p_2, p_3):*

Algorithm 3: Incremental DSS recognition algorithm

Input: $S \subset \mathbb{Z}^2$, set of points indexed from 0 to $n-1$ in the first octant
Output: Decide whether S is a segment and, if so, return its minimum parameters (a, b, μ).

Incremental recognition(S):
$i \leftarrow 0$ (maximum points index considered) ;
while *the piece of segment considered is horizontal or diagonal and* $i < n$. **do**
  $i \leftarrow i + 1$;
if $i = n$ **then**
  Return $(0, 1, 0)$ or $(1, 1, 0)$ depending on whether the piece in question is horizontal or diagonal. ;
else
  test $\leftarrow \{(0, 0), (i - 1, 0)\}$ or $\{(0, 0), (i - 1, i - 1)\}$ either horizontal or diagonal ;
$\text{pente}_{\text{inf}} \leftarrow \frac{0}{1}$ (lower limit) ;
$\text{pente}_{\text{sup}} \leftarrow \frac{1}{1}$ (upper limit) ;
$\frac{a}{b} \leftarrow \text{pente}_{\text{inf}} \oplus \text{pente}_{\text{sup}}$ (current slope initially 1/2) ;
$\mu \leftarrow 0$;
while $i < n$ *(we have not yet considered the entire segment)* **do**
  correct $\leftarrow$ FALSE (indicates whether the current slope is suitable) ;
  while $b < n$ *and not correct* **do**
    $p_{\min}, p_{\max} \leftarrow$ Extremal-rests$(a, b, \textit{test} \cup \{S[i]\})$ (minimum and maximum residual points for the a/b slope) ;
    $\delta \leftarrow p_{\max} - p_{\min}$;
    if $\langle \delta, (a, -b) \rangle < b$ *(the slope a/b is suitable)* **then**
      $\mu \leftarrow -\langle p_{\min}, (a, -b) \rangle$;
      $i \leftarrow i + 1$ (if the slope is suitable, move on to the next point);
      test $\leftarrow$ all the extremal leaning points using the parameters (a, b, μ) of the equation for the current segment piece ;
      correct $\leftarrow$ TRUE ;
    if *not correct* **then**
      if δ *has an abscissa* > 0 *(the current slope is too steep)* **then**
        $\text{pente}_{\text{sup}} \leftarrow \frac{a}{b}$;
      if δ *has an abscissa* < 0 *(the current slope is too shallow)* **then**
        $\text{pente}_{\text{inf}} \leftarrow \frac{a}{b}$;
      $\frac{a}{b} \leftarrow \text{pente}_{\text{inf}} \oplus \text{pente}_{\text{sup}}$ (new current slope) ;
  if *not correct and* $b \geqslant n$ **then**
    Return "Not a segment"
Return(a, b, μ)

Algorithm 4: Extremal rests

Input: a, b parameters of the slope tested, *test* set of points to be tested
Output: $p_{\min}, p_{\max}$ points of *test* having the minimum and maximum residues with the slope $\frac{a}{b}$.

Extremal-rests($\textit{test}, a, b$):
$p_{\min}, p_{\max} \leftarrow$ first element of *test* ;
for p *in test* **do**
  $p \leftarrow S[i]$;
  if $\langle p, (a, -b) \rangle < \langle p_{\min}, (a, -b) \rangle$ **then**
    $p_{\min} \leftarrow p$;
  if $\langle p, (a, -b) \rangle > \langle p_{\max}, (a, -b) \rangle$ **then**
    $p_{\max} \leftarrow p$;
Return$(p_{\min}, p_{\max})$;

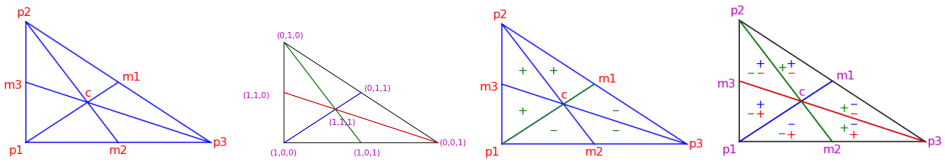

Fig. 5. (a) a step in the construction of the tree in dimension 3. The points p_1, p_2, p_3 are the extremities of the simplex. The points m_i represent the points $p_{i+1} + p_{i+2}$ (where the indices are taken modulo 3). Point c represents $p_1 + p_2 + p_3$. The permutation σ such that $\sigma(i) = (i + 1) \mod 3$ then gives the simplex with extremities $p_{\sigma(1)} = p_2$, $p_{\sigma(1)} + p_{\sigma(2)} = p_2 + p_3 = m_1$ and $p_{\sigma(1)} + p_{\sigma(2)} + p_{\sigma(3)} = c$, i.e. the triangle p_2, m_1, c at the top right of the figure.
(b) an example with initialisation at $p_i = e_i$.
(c) illustration of a separation along p_1. In the sub-triangles marked $-$, the plane contains the chord $\delta_-^{(1)}$ but not the chord $\delta_+^{(1)}$. In the triangles marked $+$, the plane contains the chord $\delta_+^{(1)}$ but not the chord $\delta_-^{(1)}$. On the dividing line in green, neither chord appears.
(d) partition of the triangle according to the chords appearing in the planes. In green, blue and red, the boundary lines. The signs then correspond to the chords appearing in the planes of each sub-zone.

- *the* $\delta_-^{(i)}$ *chord appears in* $\mathbb{P}(v, \mu)$ *iff* $v \in \mathcal{C}(p_i, m_i, p_{i-1}) \setminus \mathcal{C}(p_i, m_i)$ *(triangle* p_i, m_i, p_{i-1} *without side* p_i, m_i*).*
- *the chord* $\delta_+^{(i)}$ *appears in* $\mathbb{P}(v, \mu)$ *iff* $v \in \mathcal{C}(p_i, m_i, p_{i+1}) \setminus \mathcal{C}(p_i, m_i)$*.*

The $\delta_-^{(i)}$ *and* $\delta_+^{(i)}$ *chords are then said to be separating.*

By symmetry, we place ourselves in the case where the vector v has increasing positive coordinates. Thus, the chord δ appears in the plane of normal vector v iff $0 \leqslant |\langle v, \delta\rangle| < |\langle v, e_3\rangle|$. Given extremities p_1, p_2, p_3, in order to find a separation for p_i, it suffices to place the following constraints on $(\delta_-^{(i)}, \delta_+^{(i)})$:

$$\begin{cases} \left\langle p_i, \delta_-^{(i)} \right\rangle &= \langle p_i, e_3\rangle \\ \left\langle m_i, \delta_-^{(i)} \right\rangle &= \langle m_i, e_3\rangle \\ \left\langle p_{i-1}, \delta_-^{(i)} \right\rangle &= \langle p_{i-1}, e_3\rangle - 1 \end{cases} \quad \text{and} \quad \begin{cases} \left\langle p_i, \delta_+^{(i)} \right\rangle &= \langle p_i, e_3\rangle \\ \left\langle m_i, \delta_+^{(i)} \right\rangle &= \langle m_i, e_3\rangle \\ \left\langle p_{i+1}, \delta_+^{(i)} \right\rangle &= \langle p_{i+1}, e_3\rangle - 1 \end{cases}$$

The first two conditions for each system ensure that the limit of appearance of the chords $\delta_-^{(i)}$ and $\delta_+^{(i)}$ is the green line passing through p_i and m_i (see Figure 5 (c)).The last condition ensures that $\delta_-^{(i)}$ appears in the plane with normal vector p_{i-1} (resp. $\delta_+^{(i)}$ in the plane with normal vector p_{i+1}). The characterisation by equivalence of the separation $(\delta_-^{(i)}, \delta_+^{(i)})$ then follows immediately from these observations by convexity. Finally, let us add that the both systems each admit a unique solution because they have determinant ± 1 (by induction on the tree).

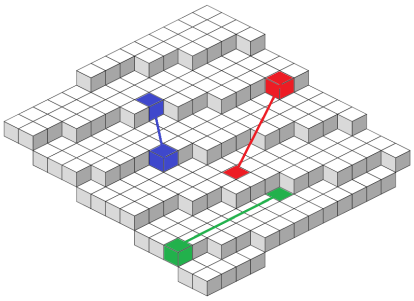

Fig. 6. Chords in the plane with normal vector $(7, 17, 57)$. The plane contains, among others, the chords $(7, 0, 0)$ (green), $(2, 3, -1)$ (blue) and $(6, 4, -2)$ (red). However, the plane does not contain the chords $(0, 0, 1)$ or $(0, 4, 0)$.

By separating the zones according to each p_i, we obtain a partition of the triangle p_1, p_2, p_3 into 6 sub-triangles (see Figure 5 (d)).

Remark 11 *The condition* $\left\langle p_{i+1}, \delta_+^{(i)} \right\rangle = \langle p_{i+1}, e_3 \rangle - 1$ *can be replaced by* $\left\langle p_{i+1}, \delta_+^{(i)} \right\rangle = \xi$ *for any* ξ *in the interval* $[\![-(\langle p_{i+1}, e_3 \rangle - 1), \langle p_{i+1}, e_3 \rangle - 1]\!]$. *However, choosing* $\xi = \langle p_{i+1}, e_3 \rangle - 1$ *results a priori in smaller separating chords and is therefore generally more relevant.*

The separations can therefore be used to characterise the membership of planes to certain cones. For example, we can look at the separation for the Stern-Brocot tree in dimension 2, with extremities 0 and 1/2 and center 1/3. After solving the system, we obtain $\delta_- = (3, 0)$ and $\delta_+ = (3, 2)$. The straight lines whose slope is between 0 and 1/2 are therefore :

- in the interval $[0, 1/3[$ iff they contain the chord $(3, 0)$, i.e. a level of size at least 4.
- in the interval $]1/3, 1/2]$ iff they contain the chord $(3, 2)$, i.e. a level of size exactly 2
- the 1/3 slope is the only one not to contain any of the 2 chords: the levels are all exactly 3 in size.

We can then return to the Stern-Brocot tree in dimension 2 with the various separations obtained for the interval $[0, 1]$, as in Figure 7.

5 Recognition algorithm in arbitrary dimension

Using the separations, we can choose the appropriate sub-triangle for descending the Stern-Brocot tree. We then obtain a recognition algorithm (see algorithm 8).

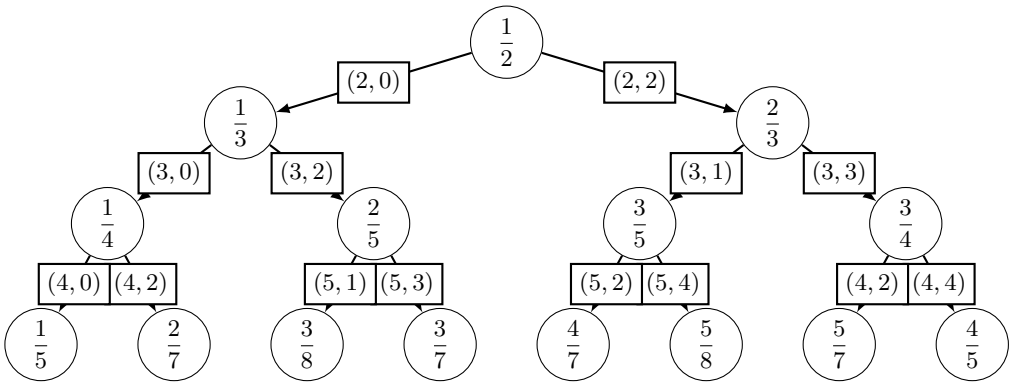

Fig. 7. Stern-Brocot tree labelled by separations

Before looking at the tricky points of the algorithm (in bold), let's look at 2 runs of the algorithm on square pieces of planes with parameters $v = (4, 7, 11), \mu = 2, \text{size} = 12$ and $v = (7, 8, 10), \mu = 8, \text{size} = 10$ in Figure 10 and 11. Note that the algorithm can also be applied to any piece of plan, not necessarily rectangular. The chords detected are indicated in the plane. The sub-triangle to descend to is indicated by a point. In the second example, the chords $(7, -10, 4)$ and $(9, -9, 0)$ belong to the whole $\mathbb{P}(v, \mu)$ plane but not to the piece shown, which is too small for this. The algorithm therefore fails and returns "Inconsistency in the chords", even though the piece is indeed that of a plane. Finally, the algorithm may only detect 2 chords and no third: this means that there is a dividing line and 2 possible sub-triangles. The algorithm presented here then chooses one of the 2 triangles, but it would also be possible to write a version that performs a Stern-Brocot in dimension 2 on the line in question.

The above algorithm has a few tricky points that need to be explained in detail:

In its current version, the algorithm necessarily stops, either because it reaches the parameters of the plane under consideration, or because the separating chords it has to deal with lead to inconsistency. In fact, their length increases at each stage, so they end up exceeding the size of the considered piece of plane. The "While" stopping condition deserves a simple bound on the coordinates of v. The leaning points algorithm presented in [14] allows us to obtain a bound in dimension 3 for rectangular pieces of plane. The simplex algorithm used in [6] gives a bound in the general case, but much larger. This part of our algorithm is therefore easy to modify.

Detecting chords in a finite set is much more problematic. First of all, the naive method is not optimal (all pairs of points are tested). However, this stage can be improved by taking into account only the leaning points of each level (zone of constant z height), thus bringing us back to a constant number of points per level. Nevertheless, after a certain stage, the chords force us to leave the S

Algorithm 8: Plane recognition

Input: $S \subset \mathbb{Z}^3$ finished in the first 48th of space
Output: Decides whether S is a piece of discrete plane and, if so, returns parameters corresponding to a plane containing it

Plane-recognition(S):
$p_1, p_2, p_3 \leftarrow (0,0,1), (0,1,1), (1,1,1)$ (extremities) ;
while ***TRUE*** **do**
  $m_1, m_2, m_3, c \leftarrow p_2 + p_3, p_1 + p_3, p_1 + p_2, p_1 + p_2 + p_3$ (middle and central points) ;
  for $v \in \{m_1, m_2, m_3, c\}$ *(testing the different points of the triangle)* **do**
    $(p_{\min}, p_{\max}) \leftarrow$ Extremal-rests(S, v) ;
    if $\langle p_{\max} - p_{\min}, v \rangle < \|v\|_\infty$ **then**
      $\mu \leftarrow -\langle p_{\min}, v \rangle$;
      Return(v, μ)
  Calculate the separations $(\delta_-^{(i)}, \delta_+^{(i)})$ for each p_i. ;
  See which chords appear in S ;
  if *the chords observed are inconsistent (see Figure 5 (d))* **then**
    Return(**"Inconsistency in the chords"**)
  Select the sub-triangle corresponding to the chords observed (see Figure 5 (d)) and update p_1, p_2, p_3 ;

Algorithm 9: Extremal rests

Input: $S \subset \mathbb{Z}^3$ finite, $v \in \mathbb{Z}^3$
Output: Calculate the minimum and maximum remainders of the points $p_{\min}$ and $p_{\max}$ of S according to v (with a simple loop).

subset under consideration. We therefore observe the same problem as with plane probing algorithms (see [12] for the first version). Generally speaking, the deeper the algorithm dives into the Stern-Brocot tree, the larger the separating chords become and the greater the risk of them leaving the study space. When the algorithm returns "Incoherence in the chords", there are two possible outcomes:

- the piece under consideration is not a piece of plane. Such an entry inevitably leads to the set of separating chords which appear in the plane not having an associated sub-triangle (see Figure 5 (d)), hence the inconsistency.
- the piece under consideration is indeed a piece of plane but is too small for some separating chords to appear on the piece. The incoherence of chords detected then simply means that the algorithm cannot conclude without considering more points.

Note that, on a sufficiently large piece of plane (a fortiori on an infinite plane), the second case mentioned cannot occur. Therefore, the reference "Incoherence in chords" implies that the input is not a plane.

The set of solution parameters of the problem forms a convex. In dimension 2, this property demonstrates that the algorithm will necessarily stop at the minimal parameters, and that there is uniqueness of the descent branch in the

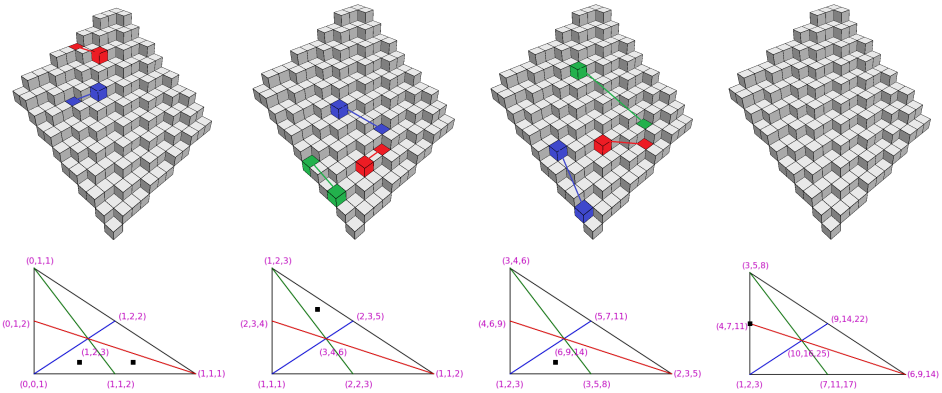

Fig. 10. Flow of the algorithm on the piece of square plane with normal vector $(4, 7, 11)$, shift 2 and size 12.

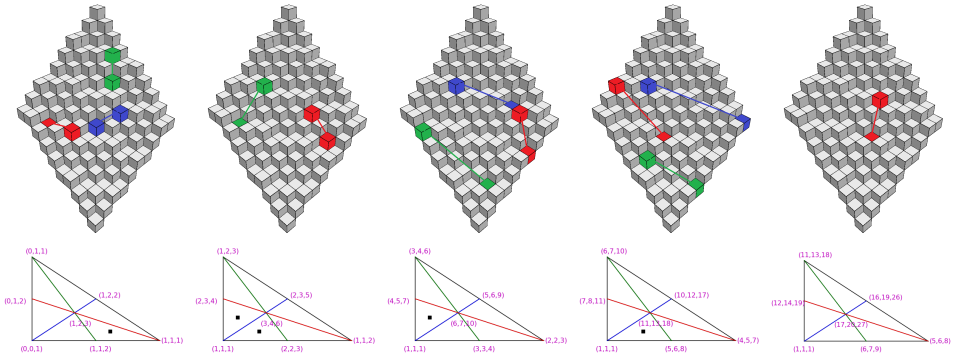

Fig. 11. Flow of the algorithm on the piece of square plane with normal vector $(7, 8, 10)$, shift 2 and size 10.

algorithm. In dimension 3, a convex may not intersect any remarkable point of the triangle and may straddle several zones. There are therefore configurations where the zone to be chosen for descent is not unique.

6 Conclusion

This paper presents a new approach for detecting pieces of arithmetic hyperplanes from the Stern-Brocot tree. In dimension 2, for DSS, the proposed algorithm is incremental and has linear complexity. The Stern-Brocot tree can also be used to characterise the various slope intervals of discrete straight lines using separating chords. The extension of the algorithm to dimensions 3 and higher retains the idea of separations, with greater combinatorial complexity. Unfortunately, although they appear in the infinite planes studied, the chords considered

may be too large to be recognisable in a piece of plane. Even if this algorithm deserves a more accomplished version, it still creates a deep link between arithmetic hyperplanes and the generalised Stern-Brocot tree, via the separating chords.

In our opinion, the notion of separating chord deserves to be explored in greater depth, in particular to make more explicit the fact that hyperplanes belong to Stern-Brocot cones. In addition, as the enumeration of the tree vertices is governed by a simple induction relation, we would like in a future work to take an interest in the induction relations induced on the set of separating chords.

References

1. E. Andres, R. Acharya, and C. Sibata. Discrete analytical hyperplanes. *Graphical Models and Image Processing*, 59(5):302–309, 1997.
2. Sebastián Barbieri and Sébastien Labbé. Indistinguishable asymptotic pairs and multidimensional sturmian configurations. *arXiv preprint arXiv:2204.06413*, 2022.
3. V. Berthé. Discrete geometry and symbolic dynamics. In *The Kiselmanfest: An International Symposium in Complex Analysis and Digital Geometry*, 2006.
4. V. E. Brimkov, D. Coeurjolly, and R. Klette. Digital planarity - A review. *Discret. Appl. Math.*, 155(4):468–495, 2007.
5. A. Brocot. *Calcul des rouages par approximation: nouvelle méthode.* A. Brocot, 1862.
6. E. Charrier and L. Buzer. An efficient and quasi linear worst-case time algorithm for digital plane recognition. In *International Conference on Discrete Geometry for Computer Imagery*, pages 346–357. Springer, 2008.
7. I. Debled-Rennesson. *Etude et reconnaissance des droites et plans discrets.* PhD thesis, Université Louis Pasteur (Strasbourg)(1971-2008), 1995.
8. I. Debled-Rennesson and J.-P. Reveillès. A linear algorithm for segmentation of digital curves. *Int. J. Pattern Recognit. Artif. Intell.*, 9(4):635–662, 1995.
9. I. Debled et J.P. Reveillès. An incremental algorithm for digital plane recognition. In *4th International Conference DGCI'94*, 1994.
10. Y. Gérard, I. Debled-Rennesson, and P. Zimmermann. An elementary digital plane recognition algorithm. *Discrete Applied Mathematics*, 151(1-3):169–183, 2005.
11. R. Klette and A. Rosenfeld. Digital straightness - a review. *Discret. Appl. Math.*, 139(1-3):197–230, 2004.
12. J.-O. Lachaud, X. Provençal, and T. Roussillon. An output-sensitive algorithm to compute the normal vector of a digital plane. *Theoretical Computer Science*, 624:73–88, 2016.
13. H. Lennerstad. *The n-dimensional Stern-Brocot tree.* 2012.
14. M. M. Mesmoudi. A simplified recognition algorithm of digital planes pieces. In *Discrete Geometry for Computer Imagery: 10th International Conference*, volume 2301 of *LNCS*, pages 404–416, 2002.
15. Marston Morse and Gustav A Hedlund. Symbolic dynamics ii. sturmian trajectories. *American Journal of Mathematics*, 62(1):1–42, 1940.
16. J.-P. Reveillès. *Géométrie discrete, calcul en nombres entiers et algorithmique.* PhD thesis, Université Louis Pasteur, 1991.
17. M. Stern. Über eine zahlentheoretische funktion. 1858.
18. F. De Vieilleville and J.-O. Lachaud. Revisiting digital straight segment recognition. In *Discrete Geometry for Computer Imagery: 13th International Conference*, volume 4245 of *LNCS*, pages 355–366. Springer, 2006.

A Proof of Theorem

Theorem 6 : The naive recognition algorithm below decides whether a finite set S of the first octant is a segment, returns its minimum parameters if it is, and then terminates in $O(nh)$ where $n = |S|$ and h is the height of the parameters of S in the Stern-Brocot tree.

Proof. **The algorithm terminates** because $n - b$ is a strictly decreasing variant with positive values.

Correction of returned parameters : Let us first show that if the algorithm returns (a, b, μ) then (a, b, μ) are parameters of the segment S. Suppose the algorithm returns (a, b, μ). If $b = 1$ then the algorithm has concluded on a horizontal or diagonal segment, hence the result.
Otherwise, let us call $p_{\min}, p_{\max}$ the output of Extremal-rests(S, a, b). We then have $\langle p_{\max} - p_{\min}, (a, -b)\rangle < b$ and $\mu = -\langle p_{\min}, (a, -b)\rangle$.
Let p be a point on S. By obvious Extremal-rests correction, $\langle p_{\min}, (a, -b)\rangle \leqslant \langle p, (a, -b)\rangle \leqslant \langle p_{\max}, (a, -b)\rangle$.
So $0 \leqslant \langle p, (a, -b)\rangle - \langle p_{\min}, (a, -b)\rangle \leqslant \langle p_{\max}, (a, -b)\rangle - \langle p_{\min}, (a, -b)\rangle < b$.
So $0 \leqslant \langle p, (a, -b)\rangle + \mu < \|(a, b)\|_\infty$, which proves that S is a segment with parameters (a, b, μ).
In particular, if S is not a segment, the algorithm returns "Not a segment".

It remains to show that if S is a segment then the algorithm returns its minimal (a_S, b_S, μ_S) parameters. If S is horizontal or diagonal, the result is trivial.

Otherwise, we show the following **invariant**: $\text{pente}_{\text{inf}} < \frac{a_S}{b_S} < \text{pente}_{\text{sup}}$.
The invariant is initially true because S is in the first octant and is neither horizontal nor diagonal.
Assuming the invariant is verified, let's perform an additional loop for the current slope $\frac{a}{b}$. Let $p_{\min}, p_{\max}$ be the output of Extremal-rests(S, a, b) and let $\delta = (x, y) \stackrel{\text{def}}{=} p_{\max} - p_{\min}$.
Note first of all that, as S is in the first octant, the abscissa x of δ is non-zero.
As $p_{\min} = (x_{\min}, y_{\min})$ and $p_{\max} = (x_{\max}, y_{\max})$ are points of S, $0 \leqslant a_S x_{\min} - b_S y_{\min} + \mu_S < b_S$ and $0 \leqslant a_S x_{\max} - b_S y_{\max} + \mu_S < b_S$.
By subtracting the 2 identities, we deduce $-b_S < a_S x - b_S y < b_S$.
So $\frac{a_S}{b_S} x - y < 1$.
In addition, $\langle \delta, (a, -b)\rangle \geqslant b$ so $ax - by \geqslant b$ hence $\frac{a}{b} x - y \geqslant 1 > \frac{a_S}{b_S} x - y$.
So $\frac{a}{b} x > \frac{a_S}{b_S} x$.
If $x > 0$ then $\frac{a}{b} > \frac{a_S}{b_S}$ so $\frac{a_{\text{inf}}}{b_{\text{inf}}} < \frac{a_S}{b_S} < \frac{a}{b}$ which proves the invariant.
If $x < 0$ then $\frac{a}{b} < \frac{a_S}{b_S}$ so $\frac{a}{b} < \frac{a_S}{b_S} < \frac{a_{\text{sup}}}{b_{\text{sup}}}$ which proves the invariant.

End of proof of correction. Once the invariant has been proved, note that $b_S < n$. In fact, according to [7], the slope $\frac{a_S}{b_S}$ is characterised from the slope between 2 lower (or upper) leaning points of the segment. The denominator b_S of this slope cannot therefore exceed the greatest difference in abscissa in the

segment, i.e. $n-1$.Ṡo the fraction $\frac{a_S}{b_S}$ appears in the Stern-Brocot tree. The fraction $\frac{a_S}{b_S}$ will therefore necessarily be visited by the invariant and the algorithm will then conclude.

Minimality : Finally, let us show that the parameters (a, b, μ) returned are indeed the minimal parameters (a_S, b_S, μ_S) of the segment. Let $\frac{a'}{b'}$ be the common ancestor of $\frac{a}{b}$ and $\frac{a_S}{b_S}$ in the Stern-Brocot tree. $\frac{a'}{b'}$ is therefore a convex combination of $\frac{a}{b}$ and $\frac{a_S}{b_S}$, i.e. $\frac{a'}{b'} = \lambda_1 \frac{a}{b} + \lambda_2 \frac{a_S}{b_S}$.
Let $(x, y) \in S$.
$0 \leqslant \frac{a}{b}x + \frac{\mu}{b} - y < 1$ et $0 \leqslant \frac{a_S}{b_S}x + \frac{\mu_S}{b_S} - y < 1$.
Taking the convex combination of the 2 equations, we obtain $0 \leqslant \left(\lambda_1 \frac{a}{b} + \lambda_2 \frac{a_S}{b_S}\right) x + (\lambda_1 \frac{\mu}{b} + \lambda_2 \frac{\mu_S}{b_S}) - y < 1$.
So, if we assume $\mu' \stackrel{\text{def}}{=} \lambda_1 \frac{\mu}{b} + \lambda_2 \frac{\mu_S}{b_S}$, we obtain $0 \leqslant \frac{a'}{b'}x + \frac{\mu'}{b} - y < 1$. So (a', b', μ') are parameters of S.
These parameters were visited by the algorithm, so $(a, b, \mu) = (a', b', \mu')$. In addition, since (a', b', μ') is an ancestor of (a_S, b_S, μ_S), $b' \leqslant b_S$ therefore, by minimality, $(a_S, b_S, \mu_S) = (a', b', \mu')$.
So the parameters returned by the algorithm are indeed minimal.

Complexity : the algorithm moves down the Stern-Brocot tree at each loop iteration. The number of loop iterations is therefore $O(h)$. In a loop, the only costly part is the call to the Extremal-rests function, which is an array traversal, so the cost is $O(n)$. The total complexity is therefore $O(nh)$.

B Naive segment recognition algorithm

Algorithm 12: Naive segment recognition

Input: $S \subset \mathbb{Z}^2$

Output: Decide whether S is a segment and if so return its minimum parameters (a, b, μ).

Naive recognition(S):
if *S is horizontal (resp. diagonal)* **then**
  Return $(0, 1, 0)$ (resp. $(1, 1, 0)$) ;
$n \leftarrow |S|$;
if *the abscissas of S are not indexed by* $[\![0, n-1]\!]$ **then**
  Return "Not a segment" ;
$\text{pente}_{\text{inf}} \leftarrow \frac{0}{1}$ (lower limit) ;
$\text{pente}_{\text{sup}} \leftarrow \frac{1}{1}$ (upper limit) ;
$\frac{a}{b} \leftarrow \text{pente}_{\text{inf}} \oplus \text{pente}_{\text{sup}}$ (current slope initially worth 1/2) ;
while $b < n$ **do**
  $p_{\min}, p_{\max} \leftarrow$ Extremal-rests(S, a, b) (minimum and maximum residual points for the slope a/b) ;
  $\delta \leftarrow p_{\max} - p_{\min}$;
  if $\langle \delta, (a, -b) \rangle < b$ *(the slope a/b is suitable)* **then**
    $\mu \leftarrow -\langle p_{\min}, (a, -b) \rangle$;
    Return$((a, b, \mu))$
  if *δ has an abscissa* > 0 *(the current slope is too steep)* **then**
    $\text{pente}_{\text{sup}} \leftarrow \frac{a}{b}$;
  if *δ has an abscissa* < 0 *(the current slope is too small)* **then**
    $\text{pente}_{\text{inf}} \leftarrow \frac{a}{b}$;
  $\frac{a}{b} \leftarrow \text{pente}_{\text{inf}} \oplus \text{pente}_{\text{sup}}$ (new current slope) ;
Return "Not a segment ;

Algorithm 13: Extremal rests

Input: $S \subset \mathbb{Z}^2$, a, b parameters of the slope tested

Output: $p_{\min}, p_{\max}$ points of S having the minimum and maximum residues with the slope $\frac{a}{b}$.

Extremal-rests(S, a, b):
$p_{\min}, p_{\max} \leftarrow S[0], S[1]$;
for *i going from* 1 *to* $n-1$ **do**
  $p \leftarrow S[i]$;
  if $\langle p, (a, -b) \rangle < \langle p_{\min}, (a, -b) \rangle$ **then**
    $p_{\min} \leftarrow p$;
  if $\langle p, (a, -b) \rangle > \langle p_{\max}, (a, -b) \rangle$ **then**
    $p_{\max} \leftarrow p$;
Return$(p_{\min}, p_{\max})$;