

An output-sensitive algorithm to compute the normal vector of a digital plane

Jacques-Olivier Lachaud¹, Xavier Provençal¹, and Tristan Roussillon²

¹Université de Savoie, CNRS, LAMA, UMR 5127, F-73776, France

²Université de Lyon, CNRS, INSA-LYON, LIRIS, UMR 5205, F-69622, France

September 18, 2015

Abstract

A digital plane is the set of integer points located between two parallel planes. We solve the following problem: how to compute the exact normal vector of a digital plane given only a predicate that answers the question “is a point x in the digital plane or not”. Our approach is iterative and “as local as possible”. We provide a worst-case complexity bound in $\mathcal{O}(\omega \log \omega)$ calls to the predicate, where ω is equal to the arithmetic thickness parameter of the digital plane. Furthermore, our algorithm presents a much better average behavior in practice.

1 Introduction

The study of linear structures onto digital objects is of central importance in digital geometry. The first step was of course to examine 2D digital straightness (e.g. see the review of Klette and Rosenfeld [24]). These studies showed that digital straight lines have lots of properties. Their arithmetic definition as diophantine inequalities leads to optimal online recognition algorithm [11]. Their combinatorics present lots of self-symmetries [7, 2] and palindromes [2, 5]. Their recursive nature is directly related to the simple continued fraction approximation of the line slope [1, 35]. Geometrically, digital straight lines are both digitally convex and concave [31]. The structure of their Delaunay triangulation is also related to their recursive structure [30]. This fundamental studies on digital straightness were used for analysing 2D digital contours, by considering the finite pieces of digital straight lines included in the contours [33, 18]. The ones that were not included into any one else were called *fundamental* [13] or *maximal segments* [10]. They emerge as an essential tool for analysing digitized curves, e.g. to decompose a curve into convex and concave parts [17, 13, 31]. Their asymptotic properties [10] were the main ingredient for showing the multigrid convergence of several discrete estimators (tangent and length estimation [27], even curvature estimation [21, 28]). They prove also to be useful to analyse the meaningful scale at which a contour should be considered and therefore to determine the importance of the noise level [22].

Hence, similar hopes are put into the study 3D linear structures, often called digital planes (e.g. see [6]), in order to tackle the problem of 3D digital shape analysis. Their arithmetic definition has been less fruitful for designing recognition algorithm, except for some ideas developed in [12]. Most algorithms recognising pieces of digital planes [23, 34, 19] use computational geometry techniques based on convexity properties. Some of them uses arithmetic but only to simplify the search of the plane parameters [8]. Their combinatorial structure has appeared to be more complex as expected [26]. This fact became apparent when studying their connectedness [20, 15, 4, 14]. The first link between a digital plane and the two-dimensional continued fraction of its normal vector was exhibited in [16, 3], and lead to a desubstitution algorithm for recognizing (specific) pieces of digital planes. It is clear that having the notion of maximal planes, natural 3D extension of maximal segments, would be extremely interesting for analysing 3D digital surfaces. However, most

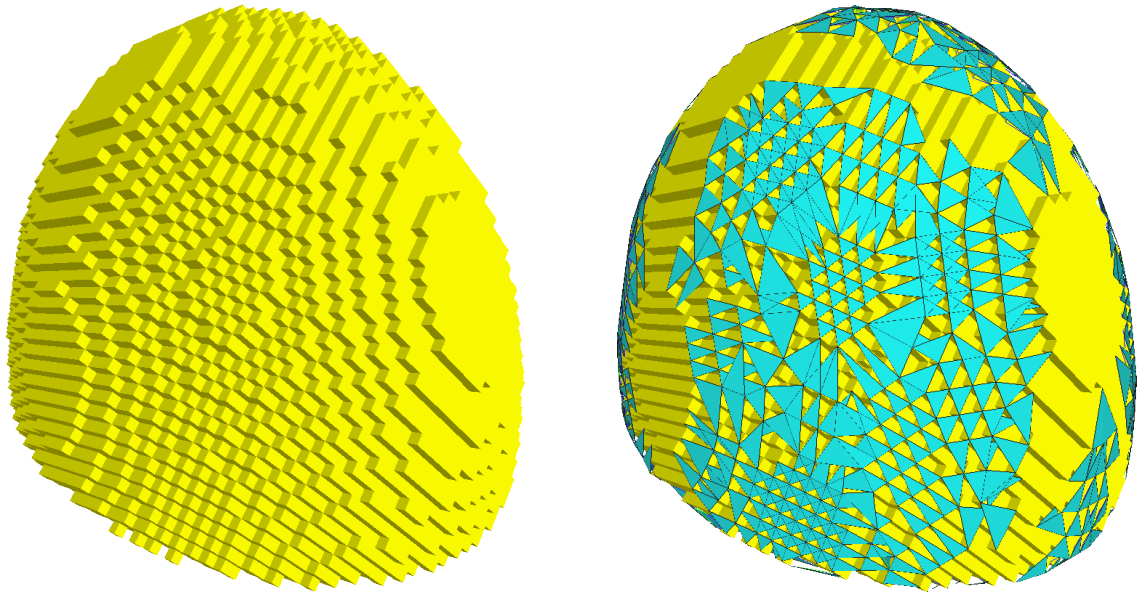


Figure 1: Even though this paper focuses on the computation of the normal vector of an infinite digital plane, the motivations come from 3D shape analysis (See Figure 2 and Figure 7 for detailed examples on a digital plane). In this image, algorithm FindNormal (Algorithm 2) is used to identify linear parts on the surface of a digitally convex object. Left : the input object is a 3D digital image described as a set of voxels. The predicate “Is $\mathbf{x} \in P$ ” is the characteristic function of the set of voxels’ vertices that belong to the boundary of the digital shape. Right : algorithm FindNormal is run starting from each reentrant corner of this set. The triangles shows the upper face of the output tetrahedrons.

works propose only a greedy segmentation into pieces of planes [25, 29]. A notable exception is the empirical approach of [9] which defines maximal planes as planar extension of maximal disks.

This work proposes a new algorithm to determine the normal vector \mathbf{N} of a digital plane P , given only the predicate “Is $\mathbf{x} \in P$?” where \mathbf{x} is any point of \mathbb{Z}^3 , and a starting point \mathbf{o} in the plane. This algorithm is local in the sense that a few points are progressively tested around the starting point \mathbf{o} . This algorithm is fast since it will not check all points surrounding \mathbf{o} . This algorithm is simple to implement, since it requires only a few elementary tests and additions of vectors. We prove that our algorithm extracts the exact characteristics of P (i.e., integer normal vector and integer offset). Furthermore, its worst-case time complexity is essentially some $O(\|\mathbf{N}\|_1 \log \|\mathbf{N}\|_1)$. Even better, the algorithm returns also the smallest lattice basis that generates the digital plane.

This algorithm sheds new light on the relation between the geometric and combinatoric properties of digital planes and three-dimensional continued fractions. Contrary to most of multidimensional continued fraction algorithms, our algorithm does not iteratively apply the same operation, but chooses at each step, among a set of possible operations, the best one with respect to the geometry of the digital plane. It extracts progressively independent Bezout vectors for the sought normal. Moreover, the specific Delaunay structure of the digital plane is also exploited to force the algorithm to be as local as possible. This is why it can extract the smallest lattice basis of the plane. In opposition with usual plane recognition algorithms [23, 34, 19, 8, 16] whose input is a set of already identified points, this algorithm decides on the fly which next points should be tested with “Is $\mathbf{x} \in P$ ”. Hence we believe that this algorithm induces the notion of maximal pieces of digital straight plane on a digital surface. This contribution is then the first step for analysing 3D digital shapes with linear geometry (see Figure 1).

The paper is organised as follows. We start in Section 2.1 by recalling basic notions about digital planes, we introduce the main definitions used in our algorithm and we state our main results. Since our algorithm has a

local approach, Section 2.2 details what is the set of tested points at each step, and how they are classified into a few configurations. Then Section 2.3 and Section 2.4 describe the operations that transform progressively the initial guess. Each operation is triggered by some configuration. The whole algorithm is presented in Section 2.5. Its correctness and its worst-case complexity are respectively established in Section 3 and 4. Then Section 5 shows how the process is kept as local as possible, using Delaunay triangulation property. Last, numerical experiments indicate that the average complexity of our algorithm is low with respect to $\|\mathbf{N}\|_1$.

2 Algorithm for plane recognition

We present a new algorithm that computes the normal vector \mathbf{N} of a digital plane

$$P = \{\mathbf{x} \in \mathbb{Z}^3 \mid 0 \leq \langle \mathbf{x}, \mathbf{N} \rangle < \omega\},$$

where vector \mathbf{N} is strictly positive and irreducible, i.e. $\mathbf{N} \in \mathbb{N}_+^3$ with $\gcd(\mathbf{N}) = 1$, and the plane is thick enough, i.e. $\omega \geq \|\mathbf{N}\|_1$. To make the exposition clearer, we are considering digital planes with shift to origin equal to zero. It should be noted however that our algorithm remains valid for an arbitrary shift of the origin, and its complexity bound remains the same.

The input of this algorithm is restricted to (1) an initial tetrahedron whose origin and vector extremities belong to P , and to (2) queries “is some point \mathbf{x} of \mathbb{Z}^3 in P ?”. Last, a constant \mathbf{Max} , greater or equal to ω , provides a stopping criterion. The algorithm principle is to move the initial tetrahedron with elementary vector operations, such that each tetrahedron is closer and closer to the boundary of the digital plane. Hence, the tetrahedron corresponds to the algorithm state, which is transformed by elementary operations until stopping criterion is met. The next subsections present more formally all these notions, and give also many useful properties that will be necessary to establish the validity and time complexity of this algorithm.

2.1 Main notions and principle of the algorithm

For any vector, say \mathbf{v} , of \mathbb{Z}^3 , we use the abbreviated notations $\bar{\mathbf{v}}$ for $\langle \mathbf{v}, \mathbf{N} \rangle$.

Definition 1. A (oriented) tetrahedron is a tuple $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w}) \in (\mathbb{Z}^3)^4$. A tetrahedron is valid if

$$\mathbf{o}, \mathbf{o} + \mathbf{u}, \mathbf{o} + \mathbf{v}, \mathbf{o} + \mathbf{w} \in P, \tag{1}$$

$$\det(\mathbf{u}, \mathbf{v}, \mathbf{w}) = 1. \tag{2}$$

$$\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}} > 0, \tag{3}$$

The normal of a tetrahedron \mathfrak{T} is the vector $\hat{\mathbf{N}}(\mathfrak{T}) := (\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u})$.

We choose to call the base of the tetrahedron o because, among with the three vectors u, v, w this defines a frame and o is its origin. To start with, the algorithm requires a valid (oriented) tetrahedron as input. Although it is not compulsory, the algorithm is generally initiated with a trivial affine \mathbb{Z}^3 frame, that is included in P .

An operation is a linear transformation of the tetrahedron:

Definition 2. An operation is a function $\lambda : (\mathbb{Z}^3)^4 \rightarrow (\mathbb{Z}^3)^4$ such that given $(\mathbf{o}'; \mathbf{u}', \mathbf{v}', \mathbf{w}') = \lambda((\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w}))$, there exists a matrix M_λ with integer coefficients that satisfies :

$$\begin{bmatrix} \mathbf{u}' \\ \mathbf{v}' \\ \mathbf{w}' \end{bmatrix} = M_\lambda \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

Moreover, an operation is called valid if $(\mathbf{o}'; \mathbf{u}', \mathbf{v}', \mathbf{w}') = \lambda(\mathfrak{T})$ is a valid tetrahedron, and $\bar{\mathbf{o}}' > \bar{\mathbf{o}}$. Note that the first condition implies that $\det(M_\lambda) = 1$.

The principle of our function FINDNORMAL for on-the-fly plane recognition follows:

Input: A predicate “is \mathbf{x} in P ?”, a valid tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$.

```

1 while true do
2   if There exist a valid operation  $\lambda$  then
3      $\mathfrak{T} \leftarrow \lambda(\mathfrak{T})$ 
4   else break;
5 return  $\mathfrak{T}$ 

```

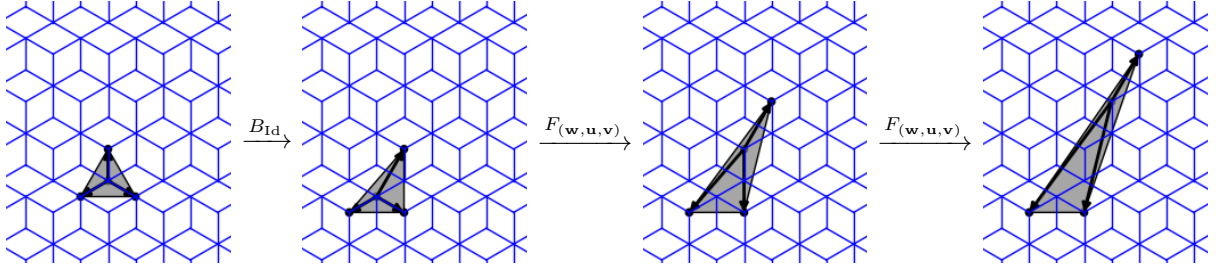


Figure 2: Illustration of the execution of FINDNORMAL on a digital plane with normal vector $\mathbf{N} = (3, 3, 4)$. The computation is initialised with the valid tetrahedron $\mathfrak{T} = (\mathbf{o}, e_1, e_2, e_3)$. From left to right, operations B_{Id} , $F_{(\mathbf{w}, \mathbf{u}, \mathbf{v})}$ and $F_{(\mathbf{w}, \mathbf{u}, \mathbf{v})}$ are used to modify the tetrahedron which, on the rightmost image, is $\mathfrak{T}' = (\mathbf{o}', (-1, 0, 2), (3, 0, -2), (-1, 0, 1))$ and we have $\hat{\mathbf{N}}(\mathfrak{T}') = (3, 3, 4)$.

The hard part is to choose the correct operation such that it is valid. This is done mainly by testing well-chosen points with “Is \mathbf{x} in P ?” around the current tetrahedron. This is called a configuration and this is detailed in the next sections. We shall prove afterwards that if the algorithm perform only valid operations, then, at completion, the normal of the tetrahedron is exactly the normal \mathbf{N} of P . This algorithm is fully detailed in Algorithm 2, page 12, note that it requires an extra parameter **Max** which limits the solution space explored. The role of **Max** is detailed in Section 2.4.3. More precisely, our main result is:

Theorem 1. *Let P be a digital plane of normal \mathbf{N} . Assume $\omega \leq \mathbf{Max}$ and a valid input tetrahedron. Then function FINDNORMAL (Algorithm 2, page 12) produces a tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ with a normal vector equal to the normal of P . Furthermore, point \mathbf{o} is just below the upper plane (i.e. $\bar{o} = \omega - 2$), vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ are Bezout vectors for \mathbf{N} (i.e. $\bar{u} = \bar{v} = \bar{w} = 1$), and $(\mathbf{v} - \mathbf{u}, \mathbf{w} - \mathbf{u})$ form a basis of the lattice $\mathfrak{X}_{\mathbf{o}} = \{\mathbf{x} \in \mathbb{Z}^3 \mid \bar{\mathbf{x}} = 0\}$.*

Furthermore, assuming a computing model where addition, subtraction, multiplication by two and division by two takes constant time, then this algorithm has the following complexity.

Theorem 2. *Function FINDNORMAL (Algorithm 2, page 12) has a time complexity $O(\omega T \log_2 \mathbf{Max})$, where T is an upper bound on the time complexity of one call to predicate “is $\mathbf{x} \in P$?”.*

For instance, if **Max** is chosen to be close to $\|\mathbf{N}\|_1$, P is a standard plane (i.e. $\omega = \|\mathbf{N}\|_1$), and the oracle “is $\mathbf{x} \in P$?” is a constant time operation, then FINDNORMAL finds the correct normal \mathbf{N} in $O(\|\mathbf{N}\|_1 \log \|\mathbf{N}\|_1)$ operations.

2.2 Local configurations around tetrahedron

At each step, we have to ensure that the tetrahedron is valid, hence we have to make sure that each chosen operation is valid. The main difficulty is to show that the transformed tetrahedron is still valid. In Definition 1, property (1) is easy to ensure, since it consists in making several queries “Is $\mathbf{x} \in P$?”. Property (2) is also easy, since it suffices to choose linear transformation with determinant 1 (unimodular matrices).

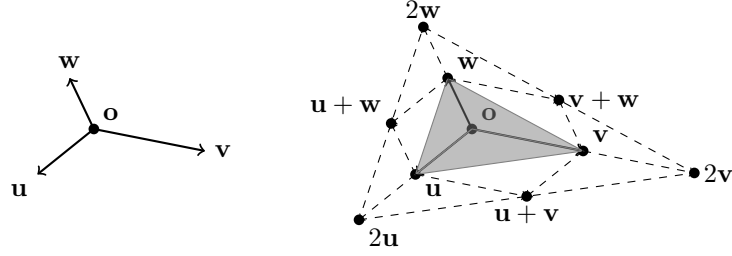


Figure 3: Left : a tetrahedron is given by a point \mathbf{o} and three vectors \mathbf{u} , \mathbf{v} , \mathbf{w} . Right : neighbor points that belong to $\mathcal{N}_{\mathfrak{T}}$. Note that in order to simplify the figure, labels are given relatively to \mathbf{o} , so that the point labeled \mathbf{u} is in fact $\mathbf{o} + \mathbf{u}$.

Property (3) is more delicate, since we wish to ensure that each $\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}}$ remains positive while true normal \mathbf{N} is unknown. We solve this issue by looking if several points around the current tetrahedron \mathfrak{T} belong or not to P . This is called the *configuration* of \mathfrak{T} . We deduce then inequalities on $\bar{\mathbf{u}}, \bar{\mathbf{v}}$ and $\bar{\mathbf{w}}$, which are used to choose the correct operation. The following result will be particularly helpful:

Lemma 1. *Let a point $\mathbf{p} \in P$ and two vectors \mathbf{x} and \mathbf{y} such that $\bar{\mathbf{x}} > 0, \bar{\mathbf{y}} > 0$.*

$$\mathbf{p} + \mathbf{x} \in P \quad \text{and} \quad \mathbf{p} + \mathbf{y} \notin P \quad \Rightarrow \quad \bar{\mathbf{x}} < \bar{\mathbf{y}}.$$

Proof. According to the arithmetical definition of a digital plane $\mathbf{p} + \mathbf{x} \in P$ means that $\bar{\mathbf{p}} + \bar{\mathbf{x}} < \omega$, whereas $\mathbf{p} + \mathbf{y} \notin P$ means that either $\bar{\mathbf{p}} + \bar{\mathbf{y}} < 0$ or $\bar{\mathbf{p}} + \bar{\mathbf{y}} \geq \omega$. In this case it must be the latter one since $0 \leq \bar{\mathbf{p}} < \omega$ and $\bar{\mathbf{y}} > 0$. As a consequence, since $\bar{\mathbf{x}}, \bar{\mathbf{y}} > 0$, we conclude that $\bar{\mathbf{x}} < \bar{\mathbf{y}}$. \square

Let us consider, around a valid tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$, the set of six neighbor points $\mathcal{N}_{\mathfrak{T}} = \{\mathbf{o} + \mathbf{u} + \mathbf{v}, \mathbf{o} + \mathbf{v} + \mathbf{w}, \mathbf{o} + \mathbf{u} + \mathbf{w}, \mathbf{o} + 2\mathbf{u}, \mathbf{o} + 2\mathbf{v}, \mathbf{o} + 2\mathbf{w}\}$. See figure 3 for an illustration of \mathfrak{T} and $\mathcal{N}_{\mathfrak{T}}$.

Local configurations involve only three points around either $\mathbf{o} + \mathbf{u}, \mathbf{o} + \mathbf{v}$ or $\mathbf{o} + \mathbf{w}$. In the sequel, we describe local configurations relatively to vector \mathbf{u} , i.e. we consider the three neighbors $\mathbf{o} + 2\mathbf{u}, \mathbf{o} + \mathbf{u} + \mathbf{v}, \mathbf{o} + \mathbf{u} + \mathbf{w}$ of point $\mathbf{o} + \mathbf{u}$. Other configurations are given by the six permutations of $(\mathbf{u}, \mathbf{v}, \mathbf{w})$. We denote by σ any permutation of $(\mathbf{u}, \mathbf{v}, \mathbf{w})$, while the identity permutation is written Id. For example, if $\sigma = (\mathbf{v}, \mathbf{w}, \mathbf{u})$, then we have $\sigma(\mathbf{u}) = \mathbf{v}, \sigma(\mathbf{v}) = \mathbf{w}$ and $\sigma(\mathbf{w}) = \mathbf{u}$.

Since we consider only three points and each of them may belong or not to P , there are exactly eight different local configurations relatively to vector \mathbf{u} . There are depicted in table 1.

The top row gather configurations such that $\mathbf{o} + 2\mathbf{u} \in P$, while the bottom row gather configurations such that $\mathbf{o} + 2\mathbf{u} \notin P$. Excepted the so-called *Empty* configuration, the configurations of the bottom row corresponds to configurations of the top row up to permutations. Indeed, if $\mathbf{o} + 2\mathbf{u} \notin P$ but $\mathbf{o} + \mathbf{u} + \mathbf{w} \in P$ (resp. $\mathbf{o} + \mathbf{u} + \mathbf{v} \in P$), then $\mathbf{o} + 2\mathbf{w} \in P$ (resp. $\mathbf{o} + 2\mathbf{v} \in P$), because if we assume that $\mathbf{o} + 2\mathbf{w} \notin P$ (resp. $\mathbf{o} + 2\mathbf{v} \notin P$), lemma 1 implies both that $\bar{\mathbf{u}} < \bar{\mathbf{w}}$ and $\bar{\mathbf{w}} < \bar{\mathbf{u}}$ (resp. $\bar{\mathbf{u}} < \bar{\mathbf{v}}$ and $\bar{\mathbf{v}} < \bar{\mathbf{u}}$), which raises a contradiction.

Therefore, we have only four main configurations, up to permutations, which are called *Translation*, *Brun-Selmer*, *FullySubtractive* and *Empty*. The implications about the relative order of $\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}}$ straightforwardly come from lemma 1. If the *Empty* configuration is observed for all permutation σ , we do not have enough information to perform a valid local operation and we must investigate a larger neighborhood (section 2.4). The name of the first three configurations stems from the name of the corresponding operations (section 2.3). More precisely, Fully Subtractive, Brun and Selmer are the names of three common generalized continued fraction algorithms [32]. Given $(a, b, c) \in \mathbb{R}^3$ with $0 \leq a \leq b \leq c$, these continued fractions algorithms are defined as :

$$\begin{aligned} \text{Fully Subtractive : } & (a, b, c) \mapsto (a, b - a, c - a) \\ \text{Brun : } & (a, b, c) \mapsto (a, b, c - b) \\ \text{Selmer : } & (a, b, c) \mapsto (a, b, c - a) \end{aligned}$$

Translation T_{Id}	Brun-Selmer B_{Id}	see B_{σ} , with $\sigma = (\mathbf{u}, \mathbf{w}, \mathbf{v})$	Fully Subtractive F_{Id}
no implication	$\bar{\mathbf{u}} < \bar{\mathbf{w}}$	$\bar{\mathbf{u}} < \bar{\mathbf{v}}$	$\bar{\mathbf{u}} < \bar{\mathbf{v}}, \bar{\mathbf{u}} < \bar{\mathbf{w}}$
Empty E_{Id}	see T_{σ} or B_{σ} , with $\sigma = (\mathbf{w}, \mathbf{u}, \mathbf{v})$	see T_{σ} or B_{σ} with $\sigma = (\mathbf{v}, \mathbf{u}, \mathbf{w})$	see T_{σ} or B_{σ} $\sigma \in \{(\mathbf{w}, \mathbf{u}, \mathbf{v}), (\mathbf{v}, \mathbf{u}, \mathbf{w})\}$

Table 1: Local configurations relatively to Id. Points that belong (resp. do not belong) to P are depicted with black (resp. white) disks. There are only four main configurations up to permutations (in bold).

Note that the Brun-Selmer configuration does not allow to determine if $\bar{\mathbf{u}} < \bar{\mathbf{v}}$ and thus the Brun-Selmer operation (Section 2.3), mimics either Brun or Selmer algorithm.

2.3 Local operations

In this section, we present the three types of local operations. Each of these operations are described in table 2 relatively to Id, while they may be considered relatively to any permutation. Each of these operations is valid because it is triggered by a specific configuration.

Lemma 2. *A local operation λ is valid on a valid tetrahedron \mathfrak{T} .*

Proof. Let $\mathfrak{T}' = (\mathbf{o}'; \mathbf{u}', \mathbf{v}', \mathbf{w}') = \lambda(\mathfrak{T})$ and, without loss of generality, assume that $\sigma = \text{Id}$. For all local operations, we have $\mathbf{o}' = \mathbf{o} + \mathbf{u}$ and thus $\bar{\mathbf{o}}' > \bar{\mathbf{o}}$ since $\bar{\mathbf{u}} > 0$. It remains to check that \mathfrak{T}' is a valid tetrahedron. In all three cases, the fact that $\{\mathbf{o}', \mathbf{o}' + \mathbf{u}', \mathbf{o}' + \mathbf{v}', \mathbf{o}' + \mathbf{w}'\} \subset P$ (Definition 1, property (1)) is straightforward from the configurations (see table 1). We also have that $\det(\mathbf{u}, \mathbf{v}, \mathbf{w}) = 1$ (Definition 1, property (2)), since the reader may easily check that the transformation matrix M_{λ} is unimodular in all three cases. It remains to check that $\bar{\mathbf{u}}', \bar{\mathbf{v}}', \bar{\mathbf{w}}' > 0$ for the three local operations (Definition 1, property (3)):

- The case $\lambda = T_{\text{Id}}$ is trivial.
- For the case $\lambda = B_{\text{Id}}$, $\bar{\mathbf{w}}' > 0$ because $\bar{\mathbf{w}} > \bar{\mathbf{u}}$ from lemma 1 (see table 1).
- For the case $\lambda = F_{\text{Id}}$, $\bar{\mathbf{v}}', \bar{\mathbf{w}}' > 0$ because $\bar{\mathbf{v}} > \bar{\mathbf{u}}$ and $\bar{\mathbf{w}} > \bar{\mathbf{u}}$ from lemma 1 (see table 1).

□

2.4 Non-local operations

Local operations from Section 2.3 alone may not always find the normal vector of a digital plane since two digital planes with different normal vectors may be identical for arbitrary large regions. In order to compute the normal vector in all cases, we introduce non-local operations that are based on the exploration of a lattice above the current tetrahedron. Such non-local operations are triggered if there does not exist any valid local operation, i.e. we are in the case of the *Empty* configuration for all permutations σ .

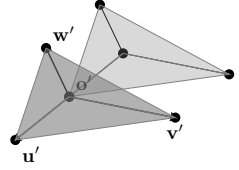
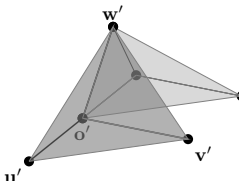
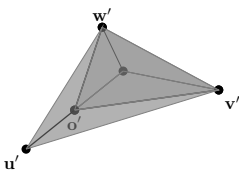
Translation T_{Id}	$\begin{aligned}\mathbf{o}' &= \mathbf{o} + \mathbf{u} \\ \mathbf{u}' &= \mathbf{u} \\ \mathbf{v}' &= \mathbf{v} \\ \mathbf{w}' &= \mathbf{w}\end{aligned}$	
Brun-Selmer B_{Id}	$\begin{aligned}\mathbf{o}' &= \mathbf{o} + \mathbf{u} \\ \mathbf{u}' &= \mathbf{u} \\ \mathbf{v}' &= \mathbf{v} \\ \mathbf{w}' &= \mathbf{w} - \mathbf{u}\end{aligned}$	
Fully Subtractive F_{Id}	$\begin{aligned}\mathbf{o}' &= \mathbf{o} + \mathbf{u} \\ \mathbf{u}' &= \mathbf{u} \\ \mathbf{v}' &= \mathbf{v} - \mathbf{u} \\ \mathbf{w}' &= \mathbf{w} - \mathbf{u}\end{aligned}$	

Table 2: The three local operations are described relatively to Id as functions $\lambda : (\mathbb{Z}^3)^4 \rightarrow (\mathbb{Z}^3)^4$ such that $(\mathbf{o}'; \mathbf{u}', \mathbf{v}', \mathbf{w}') = \lambda((\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w}))$ (Definition 2).

Let $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ be the current tetrahedron. Since we are in the case of the *Empty* configuration for all permutations, the six points above \mathfrak{T} are not in P . Let $\mathbb{L} = \{\mathbf{o} + 2\mathbf{u} + \alpha(\mathbf{u} - \mathbf{v}) + \beta(\mathbf{u} - \mathbf{w}) \mid \alpha, \beta \in \mathbb{Z}\}$. This lattice is illustrated on Figure 4. Note that \mathbf{u} does not play a particular role in this definition since any permutation of $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ defines the same lattice. On the other hand, in order to lighten the presentation of the upcoming section, given a permutation σ of $(\mathbf{u}, \mathbf{v}, \mathbf{w})$, we define :

$$\begin{aligned}\mathbb{L}_\sigma : \quad \mathbb{Z}^2 &\rightarrow \mathbb{Z}^3 \\ (\alpha, \beta) &\mapsto \mathbf{o} + 2\sigma(\mathbf{u}) + \alpha(\sigma(\mathbf{u}) - \sigma(\mathbf{v})) + \beta(\sigma(\mathbf{u}) - \sigma(\mathbf{w}))\end{aligned}$$

Also, by abuse of notation, let $\overline{\mathbb{L}}_\sigma(\alpha, \beta)$ stand for $\overline{\mathbb{L}_\sigma(\alpha, \beta)}$. Clearly (α, β) forms a coordinate system for points of \mathbb{L} .

2.4.1 Non-local operation $F_\sigma^{\alpha, \beta}$

This operation generalizes the local Fully Subtractive operation (since $F_\sigma^{0,0} = F_\sigma$).

Definition 3. *The generalized FullySubtractive operation $F_\sigma^{\alpha, \beta}$ is a non-local operation on a tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$, defined for any pair of integers α, β . For the specific case of the permutation Id , it is the linear transformation:*

$$F_{\text{Id}}^{\alpha, \beta} : (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w}) \mapsto (\mathbf{o} + \mathbf{u}; \mathbf{u} + \alpha(\mathbf{u} - \mathbf{v}) + \beta(\mathbf{u} - \mathbf{w}), \mathbf{v} - \mathbf{u}, \mathbf{w} - \mathbf{u}).$$

The effect of this operation is illustrated on Figure 4. Of course, this operation is valid only for specific preconditions, which are given by the following lemma. Note that even though both Definition 3 and Lemma 3 does not require it, our algorithm is designed in such way that α and β are always non-negative and one of them is strictly bigger than zero. By reference to Figure 4, this means that if we consider the

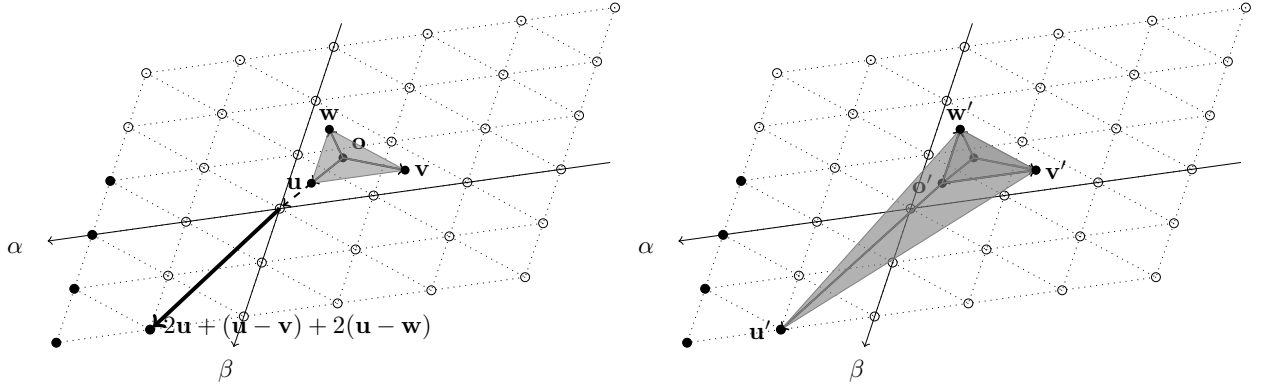


Figure 4: Illustration of the generalized FullySubtractive operation $F_{\text{Id}}^{1,2}$.

four quadrants defined by the α, β coordinate system, point $\mathbf{o} + \mathbf{u}$ is always sent to a point located in the lower left quadrant.

Lemma 3. Let $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ be a valid tetrahedron, if the three following hypothesis hold then $F_{\sigma}^{\alpha, \beta}$ is valid on \mathfrak{T} .

- (1) $\overline{\sigma(\mathbf{u})} < \overline{\sigma(\mathbf{v})}$ and $\overline{\sigma(\mathbf{u})} < \overline{\sigma(\mathbf{w})}$,
- (2) $\mathbb{L}_{\sigma}(\alpha, \beta) \in P$,
- (3) $\mathbb{L}_{\sigma}(\alpha - 1, \beta) \notin P$ or $\mathbb{L}_{\sigma}(\alpha, \beta - 1) \notin P$.

Proof. Without loss of generality, we set $\sigma = \text{Id}$ so that $\bar{\mathbf{u}} < \bar{\mathbf{v}}$ and $\bar{\mathbf{u}} < \bar{\mathbf{w}}$. Let $\mathfrak{T}' = (\mathbf{u}', \mathbf{v}', \mathbf{w}', \mathbf{o}') = F_{\text{Id}}^{\alpha, \beta}(\mathfrak{T})$, the fact that $\bar{\mathbf{o}'} > \bar{\mathbf{o}}$ is obvious since $\bar{\mathbf{u}} > 0$. It remains to check that \mathfrak{T}' is a valid tetrahedron :

- $\mathbf{o}', \mathbf{o}' + \mathbf{u}', \mathbf{o}' + \mathbf{v}', \mathbf{o}' + \mathbf{w}' \in P$ is implied by the the fact that \mathfrak{T} is a valid tetrahedron and hypothesis (2).
- $\bar{\mathbf{u}}', \bar{\mathbf{v}}', \bar{\mathbf{w}}' > 0$. For $\bar{\mathbf{v}}' > 0$ and $\bar{\mathbf{w}}' > 0$, this is a direct consequence of hypothesis (1). For $\bar{\mathbf{u}}' > 0$, we consider the case where $\mathbb{L}_{\sigma}(\alpha - 1, \beta) \notin P$, the case $\mathbb{L}_{\sigma}(\alpha, \beta - 1) \notin P$ being similar (see hypothesis (3)). By construction, we have $\mathbb{L}_{\sigma}(\alpha - 1, \beta) = \mathbb{L}_{\sigma}(\alpha, \beta) - (\mathbf{u} - \mathbf{v})$. Moreover, $\mathbb{L}_{\sigma}(\alpha - 1, \beta) \notin P$ implies that $\bar{\mathbb{L}}_{\sigma}(\alpha - 1, \beta)$ is either smaller than 0 or greater or equal to ω . In this case it must be the latter one since $0 \leq \bar{\mathbb{L}}_{\sigma}(\alpha, \beta) < \omega$ and $\bar{\mathbf{u}} - \bar{\mathbf{v}} < 0$. We have :

$$\begin{aligned}
\bar{\mathbb{L}}_{\sigma}(\alpha - 1, \beta) &\geq \omega, \\
\bar{\mathbb{L}}_{\sigma}(\alpha, \beta) &\geq \omega + \bar{\mathbf{u}} - \bar{\mathbf{v}}, \\
\bar{\mathbf{o}} + 2\bar{\mathbf{u}} + \alpha(\bar{\mathbf{u}} - \bar{\mathbf{v}}) + \beta(\bar{\mathbf{u}} - \bar{\mathbf{w}}) &\geq \omega + \bar{\mathbf{u}} - \bar{\mathbf{v}}, \\
\bar{\mathbf{u}} + \alpha(\bar{\mathbf{u}} - \bar{\mathbf{v}}) + \beta(\bar{\mathbf{u}} - \bar{\mathbf{w}}) &\geq \omega + \bar{\mathbf{u}} - \bar{\mathbf{v}} - (\bar{\mathbf{o}} + \bar{\mathbf{u}}), \\
\bar{\mathbf{u}}' &\geq \omega - (\bar{\mathbf{o}} + \bar{\mathbf{v}}) > 0
\end{aligned}$$

The last inequality comes from the fact that $\mathbf{o} + \mathbf{v} \in P$ and thus $\bar{\mathbf{o}} + \bar{\mathbf{v}} < \omega$.

- $\det(\mathbf{u}', \mathbf{v}', \mathbf{w}') = 1$. It suffices to check that $\det(M_{F_{\text{Id}}^{\alpha, \beta}}) = \det \begin{pmatrix} 1 + \alpha + \beta & -\alpha & -\beta \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} = 1$.

□

2.4.2 Non-local operation B_σ^β

This operation generalizes the local Brun-Selmer operation (since $B_\sigma^0 = B_\sigma$).

Definition 4. The generalized Brun-Selmer operation B_σ^β is a non-local operation on a tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$, defined for any integer β . For the specific case of the permutation Id, it is the linear transformation:

$$B_{\text{Id}}^{\alpha, \beta} : (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w}) \mapsto (\mathbf{o} + \mathbf{u}; \mathbf{u} + \beta(\mathbf{u} - \mathbf{w}), \mathbf{v} + \beta(\mathbf{u} - \mathbf{w}), \mathbf{w} - \mathbf{u}).$$

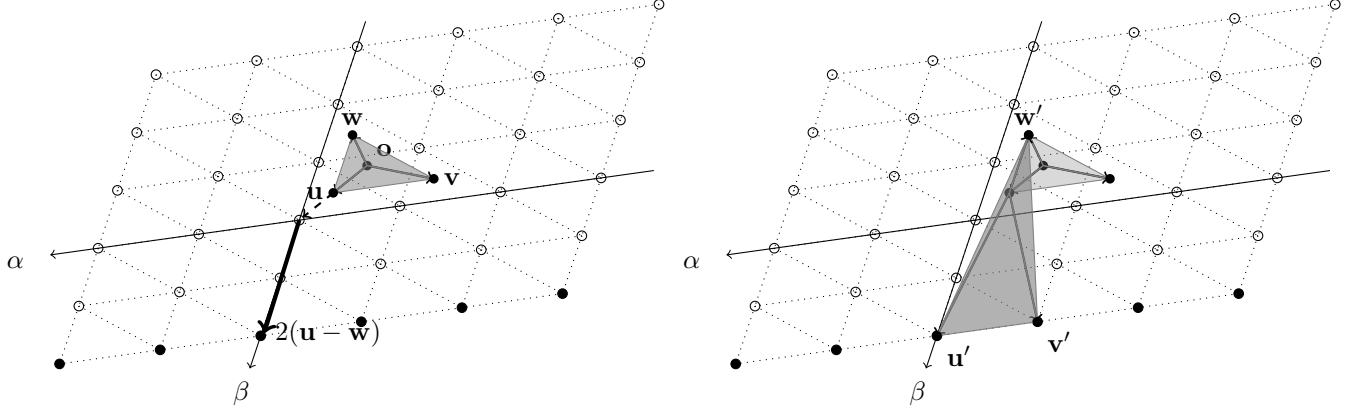


Figure 5: The generalized Brun operation B_{Id}^2 .

The effect of this operation is illustrated on Figure 5. This operation is valid only for specific preconditions, which are detailed by the following lemma. Similarly to the previous operation, the Definition 4 and Lemma 4 does not require β to be strictly positive but our algorithm is designed in such way that it is always the case.

Lemma 4. Let $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ be a valid tetrahedron, if the three following hypothesis hold then B_σ^β is valid on \mathfrak{T} .

- (1) $\overline{\sigma(\mathbf{u})} = \overline{\sigma(\mathbf{v})} < \overline{\sigma(\mathbf{w})}$,
- (2) $\mathbb{L}_\sigma(0, \beta) \in P$,
- (3) $\mathbb{L}_\sigma(0, \beta - 1) \notin P$.

Proof. This proof is similar to the proof of Lemma 3. Again, without loss of generality, we set $\sigma = \text{Id}$ so that $\bar{\mathbf{u}} = \bar{\mathbf{v}} < \bar{\mathbf{w}}$. Let $\mathfrak{T}' = (\mathbf{o}', \mathbf{v}', \mathbf{w}', \mathbf{o}') = B_{\text{Id}}^\beta(\mathfrak{T})$, the fact that $\bar{\mathbf{o}}' > \bar{\mathbf{o}}$ is obvious since $\bar{\mathbf{u}} > 0$. It remains to check that \mathfrak{T}' is a valid tetrahedron :

- $\mathbf{o}', \mathbf{o}' + \mathbf{u}', \mathbf{o}' + \mathbf{v}', \mathbf{o}' + \mathbf{w}' \in P$ is implied by the fact that \mathfrak{T} is a valid tetrahedron, hypothesis (2) and the fact that $\bar{\mathbf{u}} = \bar{\mathbf{v}}$.
- $\bar{\mathbf{u}}', \bar{\mathbf{v}}', \bar{\mathbf{w}}' > 0$. The fact that $\bar{\mathbf{w}}' > 0$ is a direct consequence of hypothesis (1). For $\bar{\mathbf{u}}' > 0$ and $\bar{\mathbf{v}}' > 0$, by construction we have $\mathbb{L}_\sigma(0, \beta - 1) = \mathbb{L}_\sigma(0, \beta) - (\mathbf{u} - \mathbf{w})$. This point is not in P by hypothesis (3) and, as in the proof of Lemma 3, $\bar{\mathbb{L}}_\sigma(0, \beta - 1) \geq \omega$ while $\bar{\mathbf{u}} - \bar{\mathbf{w}} < 0$ so that :

$$\begin{aligned} \bar{\mathbb{L}}_\sigma(0, \beta - 1) &\geq \omega, \\ \bar{\mathbb{L}}_\sigma(0, \beta) &\geq \omega + \bar{\mathbf{u}} - \bar{\mathbf{w}}, \\ \bar{\mathbf{o}} + 2\bar{\mathbf{u}} + \beta(\bar{\mathbf{u}} - \bar{\mathbf{w}}) &\geq \omega + \bar{\mathbf{u}} - \bar{\mathbf{w}}, \\ \bar{\mathbf{u}}' &\geq \omega + \bar{\mathbf{u}} - \bar{\mathbf{w}} - (\bar{\mathbf{o}} + \bar{\mathbf{u}}), \\ \bar{\mathbf{u}}' &\geq \omega - (\bar{\mathbf{o}} + \bar{\mathbf{w}}) > 0. \end{aligned}$$

The hypothesis $\bar{\mathbf{u}} = \bar{\mathbf{v}}$, implies $\bar{\mathbf{v}}' = \bar{\mathbf{u}}' > 0$.

- $\det(\mathbf{u}', \mathbf{v}', \mathbf{w}') = 1$. It suffices to check that $\det(M_{B_{\text{Id}}^\beta}) = \det \begin{pmatrix} 1 + \beta & 0 & -\beta \\ \beta & 1 & -\beta \\ -1 & 0 & 1 \end{pmatrix} = 1$.

□

2.4.3 Exploring lattice above tetrahedron for finding valid non-local operations

According to Lemmas 3 and 4, determining a valid non-local operations is related to finding pairs of nearby points such that one belongs to P and the other not. In other words, we have to find the intersection between the lattice and the sought digital plane. Function `FINDINTERSECTION` (Algorithm 1) performs this task, by first an exponential march to find some point in P , followed by a dichotomy to determine the precise location of the intersection.

Algorithm 1: Function `FINDINTERSECTION`: finds the intersection between the lattice above tetrahedron \mathfrak{T} and the sought digital plane P , in the direction $\mathbf{u} - \mathbf{v}$. A prerequisite is that $\mathbf{o} + 2\mathbf{u} \notin P$.

Input: A valid tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$, a predicate “is \mathbf{x} in P ?” and a constant \mathbf{Max} .
Output: An integer k such that $\mathbf{o} + 2\mathbf{u} + k(\mathbf{u} - \mathbf{v}) \in P$ and $\mathbf{o} + 2\mathbf{u} + (k - 1)(\mathbf{u} - \mathbf{v}) \notin P$.

```

1  $\mathbf{s} \leftarrow \mathbf{o} + 2\mathbf{u}$  ;
2  $k \leftarrow 1$  ;
3 while  $\mathbf{s} + k(\mathbf{u} - \mathbf{v}) \notin P$  do
4    $k \leftarrow 2k$  ;
5   if  $k \geq \mathbf{Max}$  then return  $\infty$  ;
6  $j \leftarrow 0$  ;
7 while  $j \neq k - 1$  do
8    $m \leftarrow \lfloor \frac{j+k}{2} \rfloor$  ;
9   if  $\mathbf{s} + m(\mathbf{u} - \mathbf{v}) \notin P$  then  $j \leftarrow m$  ;
10  else  $k \leftarrow m$  ;
11 return  $k$  ;
```

The integer returned by `FINDINTERSECTION` allows to find pairs of adjacent points in \mathbb{L} such that one is in P and the other not. When no such pair exist in the given direction, this function stops after $\log_2 \mathbf{Max}$ iterations and returns ∞ .

The following two lemmas prove that the starting point $\mathbf{o} + 2\mathbf{u}$ is not too far above the digital plane, and guarantee that the first exponential march does not go through the plane.

Lemma 5. *Let $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ be a valid tetrahedron. If $\mathbf{o} + 2\mathbf{u} \notin P$, then $\omega \leq \bar{\mathbf{o}} + 2\bar{\mathbf{u}} < 2\omega$.*

Proof. Left inequality: If $\mathbf{o} + 2\mathbf{u} \notin P$, then either $\bar{\mathbf{o}} + 2\bar{\mathbf{u}} < 0$ or $\bar{\mathbf{o}} + 2\bar{\mathbf{u}} \geq \omega$. It must be the latter one since \mathfrak{T} is valid (Definition 1) and thus $0 \leq \bar{\mathbf{o}} < \omega$ and $\bar{\mathbf{u}} > 0$.

Right inequality: Since \mathfrak{T} is valid (Definition 1), we have $0 \leq \bar{\mathbf{o}} < \omega$, $\bar{\mathbf{u}} > 0$ and $0 \leq \bar{\mathbf{o}} + \bar{\mathbf{u}} < \omega$. These inequalities imply that $\bar{\mathbf{u}} < \omega$ and we conclude that $\bar{\mathbf{o}} + 2\bar{\mathbf{u}} < 2\omega$. □

Lemma 6. *Assuming $\omega \leq \mathbf{Max}$. For a valid tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ with $\mathbf{o} + 2\mathbf{u} \notin P$, `FINDINTERSECTION` returns an integer $0 < k < \infty$ if, and only if, $\bar{\mathbf{u}} < \bar{\mathbf{v}}$. Furthermore, if $\bar{\mathbf{u}} \leq \bar{\mathbf{v}}$ and `FINDINTERSECTION` returns ∞ , then $\bar{\mathbf{u}} = \bar{\mathbf{v}}$.*

Proof. If $0 < k < \infty$ then $\mathbf{o} + 2\mathbf{u} + k(\mathbf{u} - \mathbf{v}) \in P$. Hence $0 \leq \bar{\mathbf{o}} + 2\bar{\mathbf{u}} + k(\bar{\mathbf{u}} - \bar{\mathbf{v}}) < \omega$. But $\omega \leq \bar{\mathbf{o}} + 2\bar{\mathbf{u}}$ by lemma 5. We get immediately $k(\bar{\mathbf{u}} - \bar{\mathbf{v}}) < 0$, which implies $\bar{\mathbf{u}} < \bar{\mathbf{v}}$ since $k > 0$.

Reciprocally, suppose $\bar{\mathbf{u}} < \bar{\mathbf{v}}$. We have $u_0 \geq \omega$ and the integer series $(\bar{\mathbf{u}}_k)_{k \geq 0} = (\bar{\mathbf{o}} + 2\bar{\mathbf{u}} + k(\bar{\mathbf{u}} - \bar{\mathbf{v}}))_{k \geq 0}$ is strictly decreasing so there exists k such that $\bar{\mathbf{u}}_k < \omega$. If `FINDINTERSECTION` never quits the loop at line 3,

then there is no k such that $0 \leq u_k < \omega$. This means that the search goes through the plane P without seeing it, and there is some $0 < k' < \infty$ such that

$$\omega \leq \bar{\mathbf{o}} + 2\bar{\mathbf{u}} + k'(\bar{\mathbf{u}} - \bar{\mathbf{v}}) \text{ and } \bar{\mathbf{o}} + 2\bar{\mathbf{u}} + 2k'(\bar{\mathbf{u}} - \bar{\mathbf{v}}) < 0.$$

Summing up this two inequalities implies that $k'(\bar{\mathbf{u}} - \bar{\mathbf{v}}) < -\omega$. This is impossible since $\bar{\mathbf{o}} + 2\bar{\mathbf{u}} < 2\omega$ by lemma 5 and $\omega \leq \bar{\mathbf{o}} + 2\bar{\mathbf{u}} + k'(\bar{\mathbf{u}} - \bar{\mathbf{v}})$ (definition of k').

Hence there is a k for which algorithm FINDINTERSECTION quits first loop at line 3. Then, the dichotomy trivially outputs the exact k for which we have a pair of points outside and inside P separated by a vector $\mathbf{u} - \mathbf{v}$.

The only difficulty left is the forced exit at line 5. We show that this cannot happen if $\omega \leq \mathbf{Max}$. Indeed, if $\bar{\mathbf{u}} < \bar{\mathbf{v}}$, then the test at line 3 will fail before $k \geq \omega$. Let us assume that the test at line 3 succeeds, i.e. $\bar{\mathbf{o}} + 2\bar{\mathbf{u}} + k(\bar{\mathbf{u}} - \bar{\mathbf{v}}) \notin P$, hence

$$\bar{\mathbf{o}} + 2\bar{\mathbf{u}} + k(\bar{\mathbf{u}} - \bar{\mathbf{v}}) \geq \omega.$$

Since $\bar{\mathbf{o}} + \bar{\mathbf{u}} < 2\omega$ by lemma 5, we get $k(\bar{\mathbf{u}} - \bar{\mathbf{v}}) > -\omega$. Otherwise said, since $\bar{\mathbf{u}} < \bar{\mathbf{v}}$, $k < \frac{\omega}{\bar{\mathbf{v}} - \bar{\mathbf{u}}} < \omega \leq \mathbf{Max}$. To conclude, if $\bar{\mathbf{u}} < \bar{\mathbf{v}}$, the loop at line 3 will be over before the test at line 5 is triggered. Hence FINDINTERSECTION does not return ∞ in this case. As a corollary, if $\bar{\mathbf{u}} \leq \bar{\mathbf{v}}$, FINDINTERSECTION only returns ∞ if $\bar{\mathbf{u}} = \bar{\mathbf{v}}$. \square

We also have a result on the complexity of FINDINTERSECTION.

Lemma 7. *Assuming $\omega \leq \mathbf{Max}$. For a valid tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ with $\bar{\mathbf{o}} + 2\bar{\mathbf{u}} \notin P$, FINDINTERSECTION either returns ∞ or an integer $0 < k < \omega$. Furthermore it calls the predicate “Is $\mathbf{x} \in P$?” at most $\log_2 \mathbf{Max} + \log_2 \omega$ times.*

Proof. If $\bar{\mathbf{u}} < \bar{\mathbf{v}}$, the fact that integer k is no greater than ω is proven in the previous Lemma. Then the first loop cannot make more than $\log_2 k$ iterations, since k is doubled at each iteration. The second loop is a dichotomy whose number of iterations is the logarithm of the size of the interval, which is smaller than k . Hence the predicate is called at most $\log_2 k + \log_2 k$ times. Now, $2 \log_2 k \leq 2 \log_2 \omega \leq \log_2 \mathbf{Max} + \log_2 \omega$. If $\bar{\mathbf{v}} \leq \bar{\mathbf{u}}$, we know that the test at line 5 is triggered when $k \geq \mathbf{Max}$. Since k is doubled at each iteration, the predicate is called at most $\log_2 \mathbf{Max}$, which is smaller than $\log_2 \mathbf{Max} + \log_2 \omega$. \square

2.5 The detailed algorithm

The function FINDNORMAL that computes the normal vector \mathbf{N} of a digital plane P is presented step by step in Algorithm 2. It mostly sums up to test, for each possible operation, if its preconditions are satisfied and if so, then apply it. This is a trivial task to do in the case of local operations (lines : 4, 6, 8), where it is enough to look at the local configuration. If the local configuration is empty for any permutation σ , then non-local operations are sought for. Thanks to Lemma 6, calls to FINDINTERSECTION allow to determine the following informations:

- (1) A permutation σ such that $\overline{\sigma(\mathbf{u})} \leq \overline{\sigma(\mathbf{v})} \leq \overline{\sigma(\mathbf{w})}$.
- (2) The number \mathfrak{M} of minimums of the set $\{\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}}\}$.
- (3) In the case where \mathfrak{M} is 1 or 2, integers α and β such that $F_\sigma^{\alpha, \beta}$ or B_σ^β is a valid operation.

More precisely, if $\mathfrak{M} = 1$, then Lemma 6 ensures that there exists a permutation σ that satisfies the test on line 16. In such case, the validity of the operation used on line 17 is a consequence of Lemma 3. Afterward, if $\mathfrak{M} = 2$, then, again, there must exists a permutation σ that satisfies the test on line 20 and Lemma 4 guarantees the validity of the operation used on line 21. Finally, $\mathfrak{M} = 3$ is the only case where *done* is still false at line 23 and the algorithm stops.

Note also that sometimes there are several operations that are valid. For instance, two or three different translations may be possible at the same time. The algorithm picks any of them. This does not change the overall validity of the algorithm.

Algorithm 2: Function FINDNORMAL: iterative computation of the normal vector of a digital plane P , given an oracle predicate telling if a point belongs to P and an initial valid tetrahedron within P .

Input: A valid tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$, a predicate “is \mathbf{x} in P ?” and a constant \mathbf{Max} .

Output: A valid tetrahedron $\mathfrak{T}' = (\mathbf{o}'; \mathbf{u}', \mathbf{v}', \mathbf{w}')$ such that $\hat{\mathbf{N}}(\mathfrak{T}') = \mathbf{N}$, where \mathbf{N} is the normal to P .

```

1 stop ← False ;
2 while not stop do
3   for each permutation  $\sigma$  do  $C[\sigma] \leftarrow Configuration(\mathfrak{T}, \sigma)$  ;
4   if  $\exists \sigma$  such that  $C[\sigma] = Translation$  then
5      $\mathfrak{T} \leftarrow T_\sigma(\mathfrak{T})$  ;
6   else if  $\exists \sigma$  such that  $C[\sigma] = Brun - Selmer$  then
7      $\mathfrak{T} \leftarrow B_\sigma(\mathfrak{T})$  ;
8   else if  $\exists \sigma$  such that  $C[\sigma] = FullySubtractive$  then
9      $\mathfrak{T} \leftarrow F_\sigma(\mathfrak{T})$  ;
10  else
11    for each circular permutation  $\sigma$  do
12       $k[\sigma] \leftarrow FINDINTERSECTION((\mathbf{o}; \sigma(\mathbf{u}), \sigma(\mathbf{v}), \sigma(\mathbf{w})), P, \mathbf{Max})$ ;
13       $l[\sigma] \leftarrow FINDINTERSECTION((\mathbf{o}; \sigma(\mathbf{u}), \sigma(\mathbf{w}), \sigma(\mathbf{v})), P, \mathbf{Max})$ ;
14    done ← False ;
15    for each circular permutation  $\sigma$  do
16      if not done and  $k[\sigma] < \infty$  and  $l[\sigma] < \infty$  then
17         $\mathfrak{T} \leftarrow F_\sigma^{k[\sigma], 0}(\mathfrak{T})$  ;
18        done ← True ;
19    for each circular permutation  $\sigma$  do
20      if not done and  $k[\sigma] = \infty$  and  $l[\sigma] < \infty$  then
21         $\mathfrak{T} \leftarrow B_\sigma^{l[\sigma]}(\mathfrak{T})$  ;
22        done ← True ;
23    if not done then stop ← True ;
24 return  $\mathfrak{T}$ ;

```

3 Validity of algorithm FindNormal

In this section we prove the validity of algorithm FINDNORMAL (Algorithm 2) for determining the normal vector of a digital plane P , given only a predicate “is $\mathbf{x} \in P$?” and a valid input tetrahedron. More precisely we prove here that

Theorem 1. *Let P be a digital plane of normal \mathbf{N} . Assume $\omega \leq \mathbf{Max}$ and a valid input tetrahedron. Then function FINDNORMAL (Algorithm 2, page 12) produces a tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ with a normal vector equal to the normal of P . Furthermore, point \mathbf{o} is just below the upper plane (i.e. $\bar{\mathbf{o}} = \omega - 2$), vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ are Bezout vectors for \mathbf{N} (i.e. $\bar{\mathbf{u}} = \bar{\mathbf{v}} = \bar{\mathbf{w}} = 1$), and $(\mathbf{v} - \mathbf{u}, \mathbf{w} - \mathbf{u})$ form a basis of the lattice $\mathfrak{X}_{\mathbf{o}} = \{\mathbf{x} \in \mathbb{Z}^3 \mid \bar{\mathbf{x}} = 0\}$.*

We denote by $(\mathfrak{T}_i)_{i \geq 0}$ the sequence of tetrahedra such that \mathfrak{T}_0 is the input valid tetrahedron and $\mathfrak{T}_i = (\mathbf{o}_i; \mathbf{u}_i, \mathbf{v}_i, \mathbf{w}_i) = \text{op}_i(\mathfrak{T}_{i-1})$ for $i \geq 1$, where op_i is the operation applied on i -th iteration of the **while** loop (line 2) and \mathfrak{T}_i is the value of \mathfrak{T} after the action of op_i .

The proof follows these key steps:

1. Lemma 8: at each iteration of the **while** loop (line 2), a valid operation is applied on a valid tetrahedron. Hence operation op_i is valid and tetrahedron \mathfrak{T}_i is valid.

2. Lemma 9: the sequence $(\mathfrak{T}_i)_{i \geq 0}$ is finite and Algorithm 2 terminates in a finite number of steps n less than ω .
3. Lemma 10: when Algorithm 2 terminates, we have that $\hat{\mathbf{N}}(\mathfrak{T}_n) = \mathbf{N}$ and all other relations.

3.1 Each operation op_i is valid

Lemma 8. *Let n be the number of iterations done by Algorithm 2, which may be $+\infty$ for now. For any integer $1 \leq i \leq n$, operation op_i and tetrahedron \mathfrak{T}_i are valid.*

Proof. The proof is by induction on the iteration number i , showing that each \mathfrak{T}_i is valid. We know that \mathfrak{T}_0 is valid since it is the input tetrahedron. We then assume that \mathfrak{T}_{i-1} is valid.

First, according to the local configuration, Algorithm 2 may decide that op_i is a local operation (translation at line 4, Brun-Selmer at line 6, or FullySubtractive at line 8). Then Lemma 2 concludes that op_i is valid, hence \mathfrak{T}_i is valid by definition.

Second, if line 10 is reached, then the local configuration is empty for any permutation σ . It follows that $\mathbf{o} + 2\sigma(\mathbf{u}_{i-1}) \notin P$, $\mathbf{o} + \sigma(\mathbf{u}_{i-1}) + \sigma(\mathbf{v}_{i-1}) \notin P$ and $\mathbf{o} + \sigma(\mathbf{u}_{i-1}) + \sigma(\mathbf{w}_{i-1}) \notin P$. Then every call to FINDINTERSECTION at lines 11 and 12 satisfies its prerequisites. On the one hand, Lemma 7 holds for each permutation and tells that every call terminates. On the other hand Lemma 6 also holds and arrays k and l are filled with integers which are characteristics of the order between $\bar{\mathbf{u}}$, $\bar{\mathbf{v}}$ and $\bar{\mathbf{w}}$.

If op_i was some generalized FullySubtractive operation $F_\sigma^{k[\sigma], 0}$, then it must be that $k[\sigma] < \infty$ and $l[\sigma] < \infty$. Lemma 6 tells that $\sigma(\mathbf{u}) < \sigma(\mathbf{v})$ and $\sigma(\mathbf{u}) < \sigma(\mathbf{w})$, so that $\{\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}}\}$ has a unique minimum. Furthermore, $k[\sigma] < \infty$ induces $\mathbb{L}_\sigma(k[\sigma], 0) \in P$ and $\mathbb{L}_\sigma(k[\sigma] - 1, 0) \notin P$. It follows that Lemma 3 holds, so $F_\sigma^{k[\sigma], 0} = \text{op}_i$ is valid on \mathfrak{T}_{i-1} and \mathfrak{T}_i is valid.

If op_i was some generalized Brun-Selmer operation $B_\sigma^{l[\sigma]}$, then it must be that $k[\sigma] = \infty$ and $l[\sigma] < \infty$. At this point, this is enough to conclude that $\{\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}}\}$ has exactly two minimums. Indeed, if there was only one then an operation $F_\sigma^{\alpha, \beta}$ would already have been selected. On the other hand, by Lemma 6, $l[\sigma] < \infty$ implies $\sigma(\mathbf{u}) < \sigma(\mathbf{w})$, so it must be that $\sigma(\mathbf{u}) = \sigma(\mathbf{v}) < \sigma(\mathbf{w})$. Finally, $l[\sigma] < \infty$ induces $\mathbb{L}_\sigma(0, l[\sigma]) \in P$ and $\mathbb{L}_\sigma(0, l[\sigma] - 1) \notin P$. Lemma 4 thus holds, so $B_\sigma^{l[\sigma]} = \text{op}_i$ is valid on \mathfrak{T}_{i-1} and \mathfrak{T}_i is valid.

Otherwise, nothing is done and the function exits. We have just checked all cases, and in each case, op_i was valid on \mathfrak{T}_{i-1} , so \mathfrak{T}_i was valid. \square

3.2 Algorithm 2 terminates in a finite number of steps

Lemma 9. *Sequence $(\mathfrak{T}_i)_{i \geq 0}$ is finite and Algorithm 2 terminates in a finite number of steps n less than ω .*

Proof. The termination of Algorithm 2 is straightforward from the fact that each op_i being valid, we have $\bar{\mathbf{o}}_i > \bar{\mathbf{o}}_{i-1}$ while each \mathfrak{T}_i being valid implies $\bar{\mathbf{o}}_i < \omega$. Thus the sequence $(\bar{\mathbf{o}}_i)_{i \geq 0}$ is a bounded strictly increasing sequence of integers and thus finite. Furthermore, since \mathfrak{T}_0 is valid, we have $\mathbf{o}_0 \in P$, hence $\bar{\mathbf{o}}_0 \geq 0$. It follows easily that $\bar{\mathbf{o}}_i \geq i$, hence $n < \omega$. \square

3.3 When Algorithm 2 terminates, we have $\hat{\mathbf{N}}(\mathfrak{T}_n) = \mathbf{N}$

Lemma 10. *When Algorithm 2 terminates, we have $\hat{\mathbf{N}}(\mathfrak{T}_n) = \mathbf{N}$, $\bar{\mathbf{u}}_n = \bar{\mathbf{v}}_n = \bar{\mathbf{w}}_n = 1$, $\bar{\mathbf{o}}_n = \omega - 2$, and $(\mathbf{v}_n - \mathbf{u}_n, \mathbf{w}_n - \mathbf{u}_n)$ forms a basis of the lattice $\{x \in \mathbb{Z}^3 \mid \mathbf{N} \cdot x = 0\}$.*

Proof. Let $M = \begin{bmatrix} \mathbf{u}_n \\ \mathbf{v}_n \\ \mathbf{w}_n \end{bmatrix}$ and $\mathbf{1} = \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3$, where the vectors $(\mathbf{e}_k)_{1 \leq k \leq 3}$ form the canonical basis of \mathbb{Z}^3 .

As argued previously (Section 2.5), the only case that allows the algorithm to stop is when $\{\bar{\mathbf{u}}_n, \bar{\mathbf{v}}_n, \bar{\mathbf{w}}_n\}$ has exactly three minimums, which implies that $\bar{\mathbf{u}}_n = \bar{\mathbf{v}}_n = \bar{\mathbf{w}}_n = k, k \geq 1$. As a consequence, we have

$$MN = k\mathbf{1}.$$

Because \mathfrak{T}_n is valid (Lemma 8), $\det(\mathbf{u}, \mathbf{v}, \mathbf{w}) = 1$. Besides, by definition $\langle \mathbf{u}, \hat{\mathbf{N}}(\mathfrak{T}_n) \rangle = \langle \mathbf{u}, (\mathbf{u}-\mathbf{v}) \times (\mathbf{u}-\mathbf{w}) \rangle = \det(\mathbf{u}, \mathbf{v}, \mathbf{w})$. The same thing holds for $\langle \mathbf{v}, \hat{\mathbf{N}}(\mathfrak{T}_n) \rangle$ and $\langle \mathbf{w}, \hat{\mathbf{N}}(\mathfrak{T}_n) \rangle = \det(\mathbf{u}, \mathbf{v}, \mathbf{w}) = 1$, which means that

$$M\hat{\mathbf{N}}(\mathfrak{T}_n) = \mathbf{1}.$$

Since M is invertible, we have $\mathbf{N} = k\hat{\mathbf{N}}(\mathfrak{T}_n)$, which is equivalent to $\mathbf{N} = \hat{\mathbf{N}}(\mathfrak{T}_n)$, because the hypothesis $\gcd(\mathbf{N}) = 1$ implies that $k = 1$.

Moreover, $\mathbf{o}_n + \mathbf{u}_n \in P$ (\mathfrak{T}_n is valid), and $\mathbf{o}_n + 2\mathbf{u}_n \notin P$ at termination. So $\bar{\mathbf{o}}_n + \bar{\mathbf{u}}_n < \omega$ and $\omega \leq \bar{\mathbf{o}}_n + 2\bar{\mathbf{u}}_n$. But $\bar{\mathbf{u}}_n = 1$ implies $\bar{\mathbf{o}} = \omega - 2$.

Last, the unit parallelepiped in the frame $(\mathbf{o}_n; \mathbf{u}_n, \mathbf{v}_n, \mathbf{w}_n)$ does not contain any other integer points. Otherwise, in this frame, such an integer point would have non integer coordinates, which contradicts that transition matrices have only integer coefficients (Definition 2). This proves that the face $(\mathbf{o}_n + \mathbf{u}_n, \mathbf{o}_n + \mathbf{v}_n, \mathbf{o}_n + \mathbf{w}_n)$ of \mathfrak{T}_n does not contain any integer point except its vertices. Hence $\mathbf{v}_n - \mathbf{u}_n$ and $\mathbf{w}_n - \mathbf{u}_n$ form a basis of the lattice $\{x \in \mathbb{Z}^3 \mid \mathbf{N} \cdot x = 0\}$. \square

4 Time complexity of algorithm FindNormal

It is relatively straightforward to find an upper bound for the time complexity of algorithm FINDNORMAL. There are six different permutations, and three different circular permutations. Permutations are numbered such that circular permutations have indices 0, 1, 2 and the remaining ones have indices 3, 4, 5. Therefore, accesses to arrays C , k and l takes constant times.

Theorem 2. *Function FINDNORMAL (Algorithm 2, page 12) has a time complexity $O(\omega T \log_2 \mathbf{Max})$, where T is an upper bound on the time complexity of one call to predicate “is $\mathbf{x} \in P$?”.*

Proof. First, Lemma 9 indicates that the number of iterations n of the main loop of FINDNORMAL (line 2) is lower than ω , the thickness of P .

Second, computing the six different configurations requires $O(1)$ operations and six calls to predicate “is $\mathbf{x} \in P$?”. Then testing or applying any local operation requires $O(1)$ operations. If all configurations are empty, then six calls to FINDINTERSECTION are done. Lemma 7 tells that predicate “is $\mathbf{x} \in P$?” is tested at most $\log_2 \mathbf{Max} + \log_2 \omega$ times per call to FINDINTERSECTION. Last, performing any non-local operation is also constant time. To sum up, overall complexity of one iteration is in $O(1) + O(T) + (\log_2 \mathbf{Max} + \log_2 \omega)O(T) + O(1)$. Since $\omega \leq \mathbf{Max}$, we get the result. \square

It is straightforward to see that the upper bound on the number of iterations can be reached. It suffices to consider a plane P of normal $\mathbf{N} = (1, 1, 1)$ and thickness ω , and a starting tetrahedron $(\mathbf{o}; \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$. Then the algorithm performs $\omega - 1$ translations until exiting.

One can see also that the last iteration may reach the complexity $O(\log_2 \mathbf{Max})$, for instance in the example quoted above. It is unknown to the authors if there are planes and starting tetrahedra such that the bound $\Theta(\omega \log_2 \mathbf{Max})$ is reached. Experimental evaluation shows a much better worst-case and average-case behavior (see Section 6).

5 Keeping algorithm as local as possible with Delaunay corrections

Let \mathfrak{T} be the tetrahedron produced by the function FINDNORMAL. Theorem 1 states that two edges of triangle \mathbf{uvw} form a basis of the lattice $\mathfrak{X}_\circ = \{\mathbf{x} \in \mathbb{Z}^3 \mid \bar{\mathbf{x}} = 0\}$. Given such a lattice, it is well known that its Delaunay triangulation is a set of triangles such that the two shortest edges of each triangle form a reduced basis of \mathfrak{X}_\circ . In this section we introduce an optimization for Algorithm 2 that ensures that the edges of the tetrahedron produced as output form a reduced basis of \mathfrak{X}_\circ .

As presented in Algorithm 2, function FINDNORMAL is not deterministic. For instance, at line 6, there might be two permutations that provides the Brun-Selmer configuration. In such case, even though Theorem 1

ensures the validity of the algorithm, different outputs may be produced, as illustrated in Figure 6. Since at each step of the algorithm the six points in the neighborhood defined by vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ are considered, smaller vectors imply more locality. One could modify the algorithm in such way that when more than one permutation provide a given configuration, the one that produces the smallest tetrahedron is selected. Nevertheless, there exists cases where no non-deterministic choices are made and still the output tetrahedron does not produce a reduced basis.

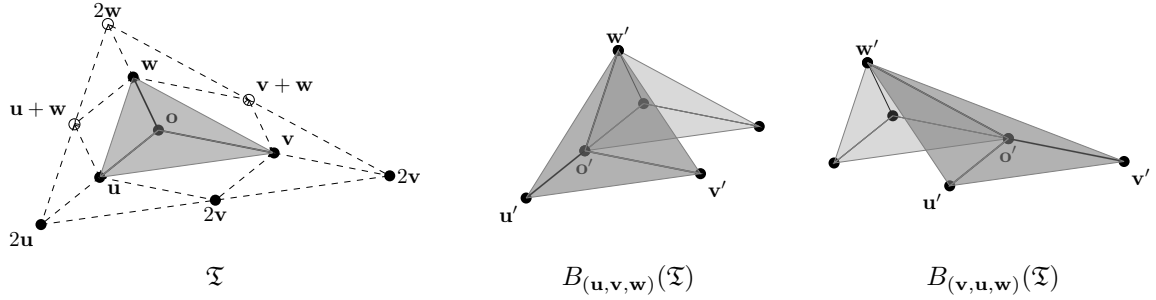


Figure 6: Left: a tetrahedron such that two permutations provide a Brun-Selmer configuration. Middle and right: tetrahedra obtained by each Brun-Selmer operation.

Given (\mathbf{a}, \mathbf{b}) a basis of a two dimensional lattice, there is an iterative algorithm to compute a reduced basis from (\mathbf{a}, \mathbf{b}) . This algorithm consist of simply replacing the longest vector among (\mathbf{a}, \mathbf{b}) by the shortest one among $\mathbf{a} + \mathbf{b}$ and $\mathbf{a} - \mathbf{b}$, if it is smaller.

Let \mathfrak{B} be the function that, given a tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ returns (\mathbf{a}, \mathbf{b}) , the two shortest vectors among $\{\mathbf{v} - \mathbf{u}, \mathbf{w} - \mathbf{u}, \mathbf{v} - \mathbf{w}\}$. We say that \mathfrak{T} defines a reduced basis if $(\mathbf{a}, \mathbf{b}) = \mathfrak{B}(\mathfrak{T})$ is such that $\max(|\mathbf{a}|, |\mathbf{b}|) \leq \min(|\mathbf{a} - \mathbf{b}|, |\mathbf{a} + \mathbf{b}|)$.

In order to modify our algorithm such that the output tetrahedron defines a reduced basis, we introduce the *Delaunay correction*, D_σ as:

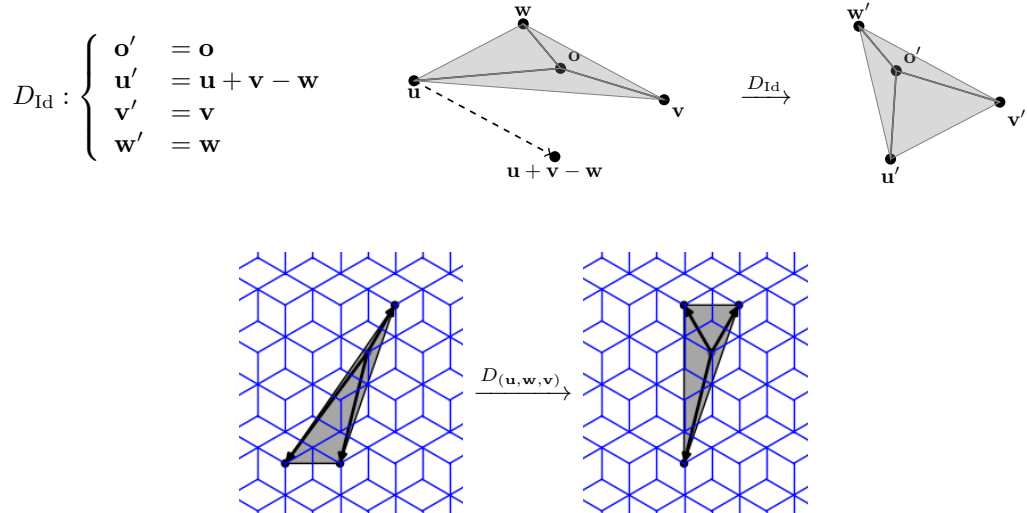


Figure 7: Continuation of the example showed in Figure 2, the Delaunay Correction $D_{(\mathbf{u}, \mathbf{w}, \mathbf{v})}$ replaces $\mathbf{u} = (3, 0, -2)$ by $\mathbf{u} + \mathbf{w} - \mathbf{v} = (0, -1, 1)$ so that $\mathbf{v}' - \mathbf{u}' = (2, 2, -3)$ and $\mathbf{w}' - \mathbf{u}' = (-1, 1, 0)$ form a reduced basis of $\mathfrak{X}_\mathbf{o} = \{\mathbf{x} \in \mathbb{Z}^3 \mid \langle \mathbf{x}, (3, 3, 4) \rangle = 0\}$.

Lemma 11. Let $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ and $(\mathbf{a}, \mathbf{b}) = \mathfrak{B}(\mathfrak{T})$, there exists two permutations σ, σ' such that

- (1) $\mathbf{a} + \mathbf{b} \in \{\mathbf{v}' - \mathbf{u}', \mathbf{w}' - \mathbf{u}', \mathbf{v}' - \mathbf{w}'\}$ where $D_\sigma(\mathfrak{T}) = (\mathbf{o}'; \mathbf{u}', \mathbf{v}', \mathbf{w}')$.
- (2) $\mathbf{a} - \mathbf{b} \in \{\mathbf{v}'' - \mathbf{u}'', \mathbf{w}'' - \mathbf{u}'', \mathbf{v}'' - \mathbf{w}''\}$ where $D_{\sigma'}(\mathfrak{T}) = (\mathbf{o}''; \mathbf{u}'', \mathbf{v}'', \mathbf{w}'')$.

Proof. Without loss of generality, suppose $\mathbf{a} = \mathbf{v} - \mathbf{u}$ and $\mathbf{b} = \mathbf{w} - \mathbf{u}$. Let $\sigma = (\mathbf{v}, \mathbf{w}, \mathbf{u})$, we have $\mathfrak{T}' = (\mathbf{o}; \mathbf{u}, \mathbf{v} + \mathbf{w} - \mathbf{u}, \mathbf{w})$ and $\mathbf{v}' - \mathbf{u}' = (\mathbf{v} + \mathbf{w} - \mathbf{u}) - \mathbf{u} = (\mathbf{v} - \mathbf{u}) + (\mathbf{w} - \mathbf{u}) = \mathbf{a} + \mathbf{b}$. The proof for $\mathbf{a} - \mathbf{b}$ is similar with $\sigma' = (\mathbf{v}, \mathbf{u}, \mathbf{w})$. \square

Let \mathfrak{T} be a tetrahedron obtained from function FINDNORMAL and define $\mathcal{L}(\mathfrak{T})$ as the lattice generated by vectors $(\mathbf{a}, \mathbf{b}) = \mathfrak{B}(\mathfrak{T})$ starting from point $\mathbf{o} + \mathbf{u}$,

$$\mathcal{L}(\mathfrak{T}) = \{\mathbf{o} + \mathbf{u} + \alpha\mathbf{a} + \beta\mathbf{b} \mid (\mathbf{a}, \mathbf{b}) = \mathfrak{B}(\mathfrak{T}) \text{ and } \alpha, \beta \in \mathbb{Z}\}.$$

One easily checks that $\mathcal{L}(\mathfrak{T}) = \mathcal{L}(D_\sigma(\mathfrak{T}))$. The three vectors of the tetrahedron \mathfrak{T} being such that $\bar{\mathbf{u}} = \bar{\mathbf{v}} = \bar{\mathbf{w}}$, we have that $\mathcal{L}(\mathfrak{T})$ is orthogonal to \mathbf{N} and completely included in P . Thus, for all σ , we have $\sigma(\mathbf{u}) + \sigma(\mathbf{v}) - \sigma(\mathbf{w}) \in P$. From these observations, we conclude that a reduced basis may be computed from the output of function FINDNORMAL by iteratively applying well chosen operations D_σ to the tetrahedron as long as a shorter basis is obtained and the result would still be a valid tetrahedron. Moreover, as stated in the following lemma, operation D_σ preserves the normal vector of the tetrahedron. It also provides sufficient conditions for D_σ to preserve the validity of the tetrahedron. As a consequence, there is no need to wait for the end of the execution of Algorithm 2 before using this operation.

Lemma 12. Let $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ be a valid tetrahedron such that no permutation provides the Translation or Brun-Selmer configuration and suppose there exists a permutation σ such that $\mathbf{o} + \sigma(\mathbf{u}) + \sigma(\mathbf{v}) - \sigma(\mathbf{w}) \in P$. In such case, $\mathfrak{T}' = D_\sigma(\mathfrak{T})$ is a valid tetrahedron with $\hat{\mathbf{N}}(\mathfrak{T}') = \hat{\mathbf{N}}(\mathfrak{T})$.

Proof. Let $\mathfrak{T}' = (\mathbf{o}'; \mathbf{u}', \mathbf{v}', \mathbf{w}')$ and, without loss of generality, assume that $\sigma = \text{Id}$. We show that \mathfrak{T}' satisfies the three conditions required by Definition 1. First, $\mathbf{o}', \mathbf{o}' + \mathbf{u}', \mathbf{o}' + \mathbf{v}', \mathbf{o}' + \mathbf{w}' \in P$ is straightforward from the definition of D_σ and the hypothesis. Second, $\bar{\mathbf{u}}' > 0$ is a consequence of the fact that no permutation provides the Translation or Brun-Selmer configuration. By contradiction, assume that $\bar{\mathbf{u}}' < 0$ so that $\bar{\mathbf{u}} + \bar{\mathbf{v}} < \bar{\mathbf{w}}$. Since \mathfrak{T} is valid, we have that $\bar{\mathbf{u}}, \bar{\mathbf{v}}, \bar{\mathbf{w}}$ are strictly positive and strictly smaller than $\omega - \bar{\mathbf{o}}$. The previous inequalities imply that $\mathbf{o} + \mathbf{u} + \mathbf{v} \in P$. Again, without loss of generality, assume that $\bar{\mathbf{u}} \leq \bar{\mathbf{v}}$, then we have that $\mathbf{o} + 2\mathbf{u} \in P$ and there are two possibilities for $\mathbf{o} + \mathbf{u} + \mathbf{w}$:

- (1) $\mathbf{o} + \mathbf{u} + \mathbf{w} \in P$, in such case, $\mathbf{o} + 2\mathbf{u}$, $\mathbf{o} + \mathbf{u} + \mathbf{v}$ and $\mathbf{o} + \mathbf{u} + \mathbf{w}$ are all in P and form the Translation configuration.
- (2) $\mathbf{o} + \mathbf{u} + \mathbf{w} \notin P$, in such case, the points $\mathbf{o} + 2\mathbf{u}$, $\mathbf{o} + \mathbf{u} + \mathbf{v}$ and $\mathbf{o} + \mathbf{u} + \mathbf{w}$ form the Brun-Selmer configuration.

Both cases are contradictory to the hypotheses, hence $\bar{\mathbf{u}}' > 0$.

Third, $\det(\mathbf{u}', \mathbf{v}', \mathbf{w}') = 1$ since D_σ defines a unimodular matrix. Finally, we have that D_σ preserves the normal vector of a tetrahedron:

$$\begin{aligned} \hat{\mathbf{N}}(\mathfrak{T}') &= (\mathbf{v}' - \mathbf{u}') \times (\mathbf{w}' - \mathbf{u}') = (\mathbf{v} - (\mathbf{u} + \mathbf{v} - \mathbf{w})) \times (\mathbf{w} - (\mathbf{u} + \mathbf{v} - \mathbf{w})), \\ &= (\mathbf{w} - \mathbf{u}) \times (2\mathbf{w} - \mathbf{u} - \mathbf{v}) = (\mathbf{w} - \mathbf{u}) \times (\mathbf{u} - \mathbf{v} + 2(\mathbf{w} - \mathbf{u})), \\ &= (\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u}) = \hat{\mathbf{N}}(\mathfrak{T}). \end{aligned}$$

\square

By abuse of notation we note $|\mathfrak{B}(\mathfrak{T})| = |\mathbf{a}| + |\mathbf{b}|$ where $(\mathbf{a}, \mathbf{b}) = \mathfrak{B}(\mathfrak{T})$. Let Π be the function that, given a tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$, test if the orthogonal projection of \mathbf{o} on the plane defined by $\mathbf{o} + \mathbf{u}$, $\mathbf{o} + \mathbf{v}$, $\mathbf{o} + \mathbf{w}$ is inside the triangle $\mathbf{o} + \mathbf{u}$, $\mathbf{o} + \mathbf{v}$, $\mathbf{o} + \mathbf{w}$ or not. Then, just before the **else if** of line 8, add the following lines:

else if $\exists \sigma$ such that $\sigma(\mathbf{u}) + \sigma(\mathbf{v}) - \sigma(\mathbf{w}) \in P$ and $|\mathfrak{B}(D_\sigma(\mathfrak{T}))| < |\mathfrak{B}(\mathfrak{T})|$ and $\Pi(D_\sigma(\mathfrak{T}))$ **then**
 $\lfloor \mathfrak{T} \leftarrow D_\sigma(\mathfrak{T}) ;$

We introduce the Π function because unlike all previous operations, D_σ may output a tetrahedron such that the point \mathbf{o} is not under the upper face $\mathbf{o} + \{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ of the tetrahedron. Note that Theorem 1 does not require this property, so this would not affect the validity of the algorithm. Nevertheless, we use this criterion here in order to obtain shorter vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ and thus more locality.

Lemma 13. *Let \mathfrak{T} be a tetrahedron such that $\Pi(\mathfrak{T}), \sigma(\mathbf{u}) + \sigma(\mathbf{v}) - \sigma(\mathbf{w}) \in P$ and $|\mathfrak{B}(D_\sigma(\mathfrak{T}))| < |\mathfrak{B}(\mathfrak{T})|$, for some permutation σ . Then, if $\neg \Pi(D_\sigma(\mathfrak{T}))$ then there exist a permutation σ' such that $\sigma'(\mathbf{u}) + \sigma'(\mathbf{v}) - \sigma'(\mathbf{w}) \in P$, $|\mathfrak{B}(D_{\sigma'}(\mathfrak{T}))| < |\mathfrak{B}(\mathfrak{T})|$ and $\Pi(D_{\sigma'}(\mathfrak{T}))$.*

Proof. Without loss of generality, suppose $\sigma = \text{Id}$. Let T_1 be the triangle $\mathbf{o} + \{\mathbf{u} + \mathbf{v} - \mathbf{w}, \mathbf{v}, \mathbf{w}\}$ and T_2 be the triangle $\mathbf{o} + \{\mathbf{u}, \mathbf{v} + \mathbf{u} - \mathbf{w}, \mathbf{w}\}$. It suffices to see that the union of T_1 and T_2 is the parallelogram $\mathbf{o} + \{\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{u} + \mathbf{v} - \mathbf{w}\}$. Consequently, let $\sigma' = (\mathbf{v}, \mathbf{u}, \mathbf{w})$, we have that :

$$(\Pi(\mathfrak{T}) \text{ and } \neg \Pi(D_\sigma(\mathfrak{T}))) \implies \Pi(D_{\sigma'}(\mathfrak{T})).$$

□

Let Algorithm 2[★] be this modified version of Algorithm 2.

Proposition 1. *Let P be a digital plane of normal \mathbf{N} . Assume $\omega \leq \mathbf{Max}$ and a valid input tetrahedron. Then Algorithm 2[★] produces a tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{u}, \mathbf{v}, \mathbf{w})$ with a normal vector equal to the normal of P . Furthermore, point \mathbf{o} is just below the upper plane (i.e. $\bar{\mathbf{o}} = \omega - 2$), vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ are Bezout vectors for \mathbf{N} (i.e. $\bar{\mathbf{u}} = \bar{\mathbf{v}} = \bar{\mathbf{w}} = 1$), and $\mathfrak{B}(\mathfrak{T})$ form a reduced basis of the lattice $\mathfrak{X}_\mathbf{o} = \{\mathbf{x} \in \mathbb{Z}^3 \mid \bar{\mathbf{x}} = 0\}$.*

Proof. First, D_σ is only used when it provides a strictly smaller pair of vectors $\mathfrak{B}(\mathfrak{T})$. Thus, it might only be used a finite number of times consecutively. The termination of the algorithm is guaranteed since D_σ does not modify the point \mathbf{o} while all the other operations strictly increases $\bar{\mathbf{o}}$.

On the other hand, the modifications made on Algorithm 2 consist in applying operation D_σ in cases where Lemma 12 guarantees the validity of the resulting tetrahedron. Hence Theorem 1 still applies. In particular, at one point in the execution of the algorithm, the tetrahedron \mathfrak{T} is such that $\bar{\mathbf{u}} = \bar{\mathbf{v}} = \bar{\mathbf{w}}$. At this point, the only operation that might be applied is D_σ and it is applied a finite number of times until $\mathfrak{B}(\mathfrak{T})$ forms a reduced basis. □

6 Conclusion

In this paper, we described and analyzed an output sensitive algorithm that computes the normal vector of a digital plane. The number of points tested is bounded by $\mathcal{O}(\omega \log \mathbf{Max})$ which, in the case of standard digital planes (thickness $\omega = \|\mathbf{N}\|_1$), is proportional to $\|\mathbf{N}\|_1 \log(\|\mathbf{N}\|_1)$.

Although it is not clear that this bound may effectively be reached, there are some very simple cases for which a high number of operations is required. For instance, let $\mathbf{N} = (1, n, n)$, $\mu = 0$ and $\omega = 2n + 1$, starting with the tetrahedron $\mathfrak{T} = (\mathbf{o}; \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$, Algorithm 2 applies the FullySubtractive operation exactly $n - 1$ times. Thus the computation time may not be less than $\Omega(n)$.

Nevertheless, in general, function FINDNORMAL is significantly faster than the worst-case upper bound of Theorem 2. We have plotted the number of points tested using the predicate “*is $\mathbf{x} \in P$?*” with respect to the 1-norm of the normal vector in Figure 8. These results show a large gap between the median case and the worst case. Figure 9 shows the number of operations used for both versions of function FINDNORMAL. Both curves show the median number of operations used for the same 5000 random normal vectors of Figure 8, one using Algorithm 2 as displayed at page 12 (the basic algorithm), the other using the modified one where Delaunay corrections ensure that the output provides a reduced basis. These results show that Delaunay corrections do not affect the overall behavior of the algorithm.

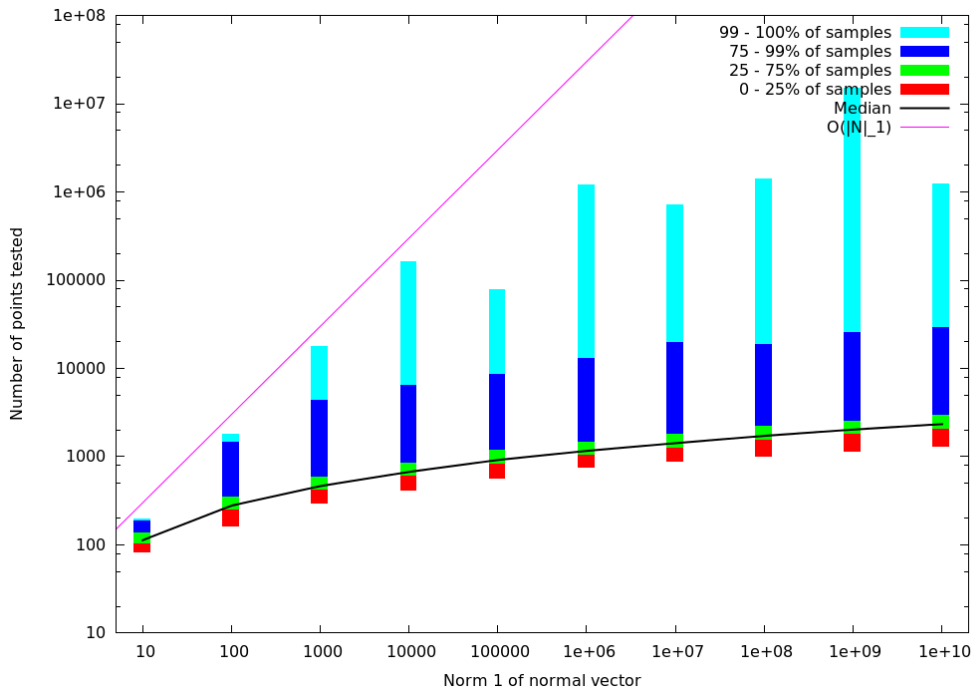


Figure 8: Number of points tested using the predicate “ $is \mathbf{x} \in P?$ ” with respect to the 1-norm of the normal vector. Each column is the results of 5000 computations. For each computation, a normal vector was randomly chosen in such way that its 1-norm is located in the interval displayed by the width of each column.

The results regarding time complexity presented in this paper can easily be improved. For instance, Appendix A shows a better bound on the output values algorithm `FINDINTERSECTION`, for the case where it is not ∞ . Nevertheless, this is not enough in order to prove an asymptotic worst-case time complexity below $\mathcal{O}(n \log n)$. However, finding tight theoretical bounds for both worst-case and average case time complexity is still an open question.

There are many future directions of research raised by this work. On the theoretical side, it would be important to relate our recognition algorithm to other multidimensional continued fraction algorithms, and exhibit for instance its ergodicity properties. On a more practical level, it would be very interesting to have a similar algorithm where we can control the global displacement between the input and the output tetrahedra. Indeed, our objective is to design an algorithm that extracts the local linear geometry of arbitrary digital surfaces (see Figure 1 for preliminary results). The predicate “ $is \mathbf{x} \in P?$ ”, defined simply as “does \mathbf{x} belong to the digital surface?”, is now only locally a digital plane. Keeping the global displacement as low as possible guarantees that we can extract the local geometry of the digital surface almost everywhere. Having a fuzzy predicate “ $is \mathbf{x} \in P?$ ” returning for instance a probability between 0 and 1 would also be a natural and interesting extension to our method. We are thinking of adapting our algorithm to a branch-and-bound algorithm, exploring several possibilities while favoring states with highest probabilities. Such extensions would also be relevant for the geometric analysis of digital objects coming from 3D imaging tools like scanners, where partial volume effects occur.

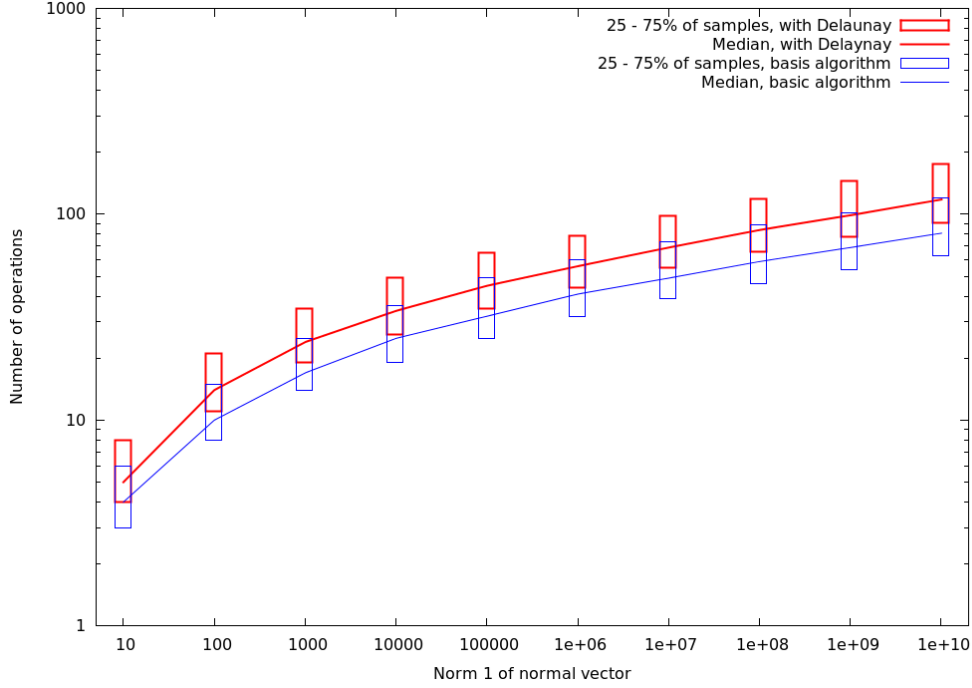


Figure 9: Number of operations (including Delaunay corrections) with respect the 1-norm of the normal vector. Each point is the median of 5000 computations using the same normal vectors as in Figure 8. The vertical intervals show the localization of the 50% central values.

A On a tighter bound for time complexity

It is possible to find tighter bounds for the values returned by `FINDINTERSECTION` and hence on the complexity of non-local operations.

Lemma 14. *If \mathcal{T} is a valid tetrahedron, $\mathbf{o} + 2\mathbf{u} \notin P$ and `FINDINTERSECTION` returns an integer $0 < k < \infty$, then $k < \min(\bar{\mathbf{u}}, \bar{\mathbf{v}})$.*

Proof. Let us denote $\mathbf{p}_k := \mathbf{o} + 2\mathbf{u} + k(\mathbf{u} - \mathbf{v})$. Since `FINDINTERSECTION` did not return ∞ , we have $\bar{\mathbf{u}} < \bar{\mathbf{v}}$, $\mathbf{p}_k \in P$ and $\mathbf{p}_{k-1} \notin P$. We proceed as follows:

- We show that $\bar{\mathbf{p}}_k > \bar{\mathbf{o}} + \bar{\mathbf{u}}$. Indeed, we have

$$\begin{aligned} \bar{\mathbf{o}} + \bar{\mathbf{u}} + (\bar{\mathbf{v}} - \bar{\mathbf{u}}) &< \omega && \text{(since } \mathbf{o} + \mathbf{v} \in P) \\ \bar{\mathbf{p}}_k + (\bar{\mathbf{v}} - \bar{\mathbf{u}}) &\geq \omega && \text{(since } \bar{\mathbf{p}}_k + (\bar{\mathbf{v}} - \bar{\mathbf{u}}) = \bar{\mathbf{p}}_{k-1} \notin P \text{ and } \bar{\mathbf{v}} - \bar{\mathbf{u}} > 0) \end{aligned}$$

By subtracting the first to the second, we get immediately $\bar{\mathbf{p}}_k > \bar{\mathbf{o}} + \bar{\mathbf{u}}$.

- We also have $\bar{\mathbf{p}}_k < \bar{\mathbf{p}}_{k-1} < \dots < \bar{\mathbf{p}}_0$ since $\bar{\mathbf{v}} - \bar{\mathbf{u}} > 0$.
- Putting these two relations together gives:

$$\bar{\mathbf{o}} + \bar{\mathbf{u}} < \bar{\mathbf{p}}_k < \bar{\mathbf{p}}_{k-1} < \dots < \bar{\mathbf{p}}_0 = \bar{\mathbf{o}} + 2\bar{\mathbf{u}}.$$

This is a strictly increasing sequence of $k + 1$ integers in an interval containing $\bar{\mathbf{u}}$ integers. It follows that $k + 1 \leq \bar{\mathbf{u}}$, which concludes. \square

This lemma induces a natural corollary on the complexity of non-local operations:

Corollary 1. *If operation $F_\sigma^{k,0}$ was performed on tetrahedron \mathfrak{T}_i by FINDNORMAL, then $k < \min(\bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i, \bar{\mathbf{w}}_i)$. If operation B_σ^l was performed on tetrahedron \mathfrak{T}_i by FINDNORMAL, then $l < \min(\bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i, \bar{\mathbf{w}}_i)$.*

References

- [1] J. Berstel, *Mots, mélanges offerts à M.P. Schützenberger*, ch. Tracé de droites, fractions continues et morphisme itérés, pp. 298–309, Hermès, 1990, In french.
- [2] J. Berstel and A. de Luca, *Sturmian words, Lyndon words and trees*, Theoret. Comput. Sci. **178** (1997), no. 1-2, 171–203. MR 1453849 (98j:68138)
- [3] V. Berthé and T. Fernique, *Brun expansions of stepped surfaces*, Discrete Mathematics **311** (2011), no. 7, 521–543.
- [4] V. Berthé, D. Jamet, T. Jolivet, and X. Provençal, *Critical connectedness of thin arithmetical discrete planes*, Proc. Discrete Geometry for Computer Imagery (DGCI'2013) (Rocio Gonzalez-Diaz, Maria-Jose Jimenez, and Belen Medrano, eds.), Lecture Notes in Computer Science, vol. 7749, Springer Berlin Heidelberg, 2013, pp. 107–118 (English).
- [5] J.-P. Borel and F. Laubie, *Quelques mots sur la droite projective réelle*, Journal de théorie des nombres de Bordeaux **5** (1993), no. 1, 23–51 (fre).
- [6] V. Brimkov, D. Coeurjolly, and R. Klette, *Digital planarity—a review*, Discrete Applied Mathematics **155** (2007), no. 4, 468–495.
- [7] A. M. Bruckstein, *The self-similarity of digital straight lines*, Proc. 10th Int. Conf. Pattern Recognition (ICPR'1990), Atlantic City, NJ, vol. 1, 1990, pp. 485–490.
- [8] E. Charrier and L. Buzer, *An efficient and quasi linear worst-case time algorithm for digital plane recognition*, Discrete Geometry for Computer Imagery (DGCI'2008), LNCS, vol. 4992, Springer, 2008, pp. 346–357.
- [9] E. Charrier and J.-O. Lachaud, *Maximal planes and multiscale tangential cover of 3d digital objects*, Proc. Int. Workshop Combinatorial Image Analysis (IWCIA2011), Lecture Notes in Computer Science, vol. 6636, Springer Berlin / Heidelberg, 2011, pp. 132–143.
- [10] F. de Vieilleville, J.-O. Lachaud, and F. Feschet, *Maximal digital straight segments and convergence of discrete geometric estimators*, Journal of Mathematical Image and Vision **27** (2007), no. 2, 471–502.
- [11] I. Debled-Renesson and J.-P. Reveillès, *A linear algorithm for segmentation of discrete curves*, International Journal of Pattern Recognition and Artificial Intelligence **9** (1995), 635–662.
- [12] I. Debled-Renesson and J.-P. Reveillès, *An incremental algorithm for digital plane recognition*, Proc. Discrete Geometry for Computer Imagery (DGCI'94), 1994, pp. 207–222.
- [13] H. Doerksen-Reiter and I. Debled-Renesson, *Convex and concave parts of digital curves*, Geometric Properties for Incomplete Data (R. Klette, R. Kozera, L. Noakes, and J. Weickert, eds.), Computational Imaging and Vision, vol. 31, Springer, 2006, pp. 145–160.
- [14] E. Domenjoud, X. Provençal, and L. Vuillon, *Facet connectedness of discrete hyperplanes with zero intercept: The general case*, Proc. Discrete Geometry for Computer Imagery (DGCI'2014), Lecture Notes in Computer Science, vol. 8668, Springer, 2014, pp. 1–12.
- [15] E. Domenjoud and L. Vuillon, *Geometric Palindromic Closure*, Uniform Distribution Theory **7** (2012), no. 2, 109–140.

- [16] T. Fernique, *Generation and recognition of digital planes using multi-dimensional continued fractions*, Pattern Recognition **42** (2009), no. 10, 2229–2238.
- [17] F. Feschet, *Canonical representations of discrete curves*, Pattern Analysis & Applications **8** (2005), no. 1, 84–94.
- [18] F. Feschet and L. Tougne, *Optimal time computation of the tangent of a discrete curve: Application to the curvature*, Proc. 8th Int. Conf. Discrete Geometry for Computer Imagery (DGCI'99), Lecture Notes in Computer Science, no. 1568, Springer Verlag, 1999, pp. 31–40.
- [19] Y. Gérard, I. Debled-Rennesson, and P. Zimmermann, *An elementary digital plane recognition algorithm*, Discrete Applied Mathematics **151** (2005), no. 1, 169–183.
- [20] D. Jamet and J.-L. Toutant, *Minimal arithmetic thickness connecting discrete planes*, Discrete Appl. Math. **157** (2009), no. 3, 500–509. MR 2479142 (2010d:52038)
- [21] B. Kerautret and J.-O. Lachaud, *Curvature estimation along noisy digital contours by approximate global optimization*, Pattern Recognition **42** (2009), no. 10, 2265 – 2278.
- [22] B. Kerautret and J.-O. Lachaud, *Meaningful scales detection along digital contours for unsupervised local noise estimation*, IEEE Transaction on Pattern Analysis and Machine Intelligence **43** (2012), 2379–2392.
- [23] C. E. Kim and I. Stojmenović, *On the recognition of digital planes in three-dimensional space*, Pattern Recognition Letters **12** (1991), no. 11, 665–669.
- [24] R. Klette and A. Rosenfeld, *Digital straightness – a review*, Discrete Applied Mathematics **139** (2004), no. 1-3, 197–230.
- [25] R. Klette and H. J. Sun, *Digital planar segment based polyhedrization for surface area estimation*, Proc. Visual form 2001, LNCS, vol. 2059, Springer, 2001, pp. 356–366.
- [26] S. Labbé and C. Reutenauer, *A d -dimensional extension of christoffel words*, Discrete and Computational Geometry (to appear), 26 p., arXiv:1404.4021.
- [27] J.-O. Lachaud, A. Vialard, and F. de Vieilleville, *Fast, accurate and convergent tangent estimation on digital contours*, Image and Vision Computing **25** (2007), no. 10, 1572–1587.
- [28] J. Levallois, D. Coeurjolly, and J.-O. Lachaud, *Parameter-free and multigrid convergent digital curvature estimators*, Discrete Geometry for Computer Imagery, LNCS, no. 8668, Springer, 2014, pp. 162–175.
- [29] L. Provot and I. Debled-Rennesson, *3d noisy discrete objects: Segmentation and application to smoothing*, Pattern Recognition **42** (2009), no. 8, 1626–1636.
- [30] T. Roussillon and J.-O. Lachaud, *Delaunay properties of digital straight segments*, Proc. International Conference on Discrete Geometry for Computer Imagery (DGCI2011) (Nancy, France), Lecture Notes in Computer Science, vol. 6607, Springer, apr 2011, pp. 308–319.
- [31] T. Roussillon and I. Sivignon, *Faithful polygonal representation of the convex and concave parts of a digital curve*, Pattern Recognition **44** (2011), no. 10-11, 2693–2700.
- [32] F. Schweiger, *Multidimensional continued fractions*, Oxford Science Publications, Oxford University Press, Oxford, 2000. MR 2121855 (2005i:11090)
- [33] A. W. M. Smeulders and L. Dorst, *Decomposition of discrete curves into piecewise straight segments in linear time*, Vision geometry: Proc. AMS special session, October 20-21, 1989 (Hoboken, New Jersey) (R. A. Melter, A. Rosenfeld, and P. Bhattacharya, eds.), vol. 119, American Mathematical Society, 1991, pp. 169–195.

- [34] P. Veelaert, *Digital planarity of rectangular surface segments*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **16** (1994), no. 6, 647–652.
- [35] K. Voss, *Discrete Images, Objects, and Functions in \mathbb{Z}^n* , Springer-Verlag, 1993.