Computation of the normal vector to a digital plane by sampling signicant points

Jacques-Olivier Lachaud, Xavier Provençal, Tristan Roussillon

Computation of the normal vector to a digital plane by sampling significant points *

Jacques-Olivier Lachaud[1], Xavier Provençal[1], Tristan Roussillon[2]

[1] Université Savoie Mont Blanc, LAMA, UMR5127, F-73376, France
{jacques-olivier.lachaud|xavier.provencal}@univ-smb.fr
[2] Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, F-69622, France
tristan.roussillon@liris.cnrs.fr

Abstract. Digital planes are sets of integer points located between two parallel planes. We present a new algorithm that computes the normal vector of a digital plane given only a predicate "is a point x in the digital plane or not". In opposition with the algorithm presented in [7], the algorithm is fully local and does not explore the plane. Furthermore its worst-case complexity bound is $O(\omega)$, where ω is the arithmetic thickness of the digital plane. Its only restriction is that the algorithm must start just below a Bezout point of the plane in order to return the exact normal vector. In practice, our algorithm performs much better than the theoretical bound, with an average behavior close to $O(\log \omega)$. We show further how this algorithm can be used to analyze the geometry of arbitrary digital surfaces, by computing normals and identifying convex, concave or saddle parts of the surface.

Keywords: Digital geometry, digital plane, recognition, normal vector estimation, lattice reduction.

1 Introduction

The study of the linear geometry of digital sets has raised a considerable amount of work in the digital geometry community. In 2D, digital straightness has been extremely fruitful. Indeed, properties of digital straight lines have impacted the practical analysis of 2D shapes, especially through the notion of *maximal segments* [4,3], which are unextensible pieces of digital straight lines along digital contours.

In 3D, the main problem is that there is no more an implication between "being a maximal plane" and "being a tangent plane" as it is in 2D. This was highlighted in [2], where maximal planes were then defined as planar extension of maximal disks. To sum up, the problem is not so much to recognize a piece of plane, but more to group together the pertinent points onto the digital shape.

* This work has been partly funded by DigitalSnow ANR-11-BS02-009 research grant

A first algorithm was proposed by the authors in [7]. Given only a predicate "is point $\boldsymbol{x}$ in plane $\mathbf{P}$?" and a starting point in $\mathbf{P}$, this algorithm extracts the exact characteristics of $\mathbf{P}$. The idea is to deform an initial unit tetrahedron based at the starting point with only unimodular transformations. Each transformation is decided by looking only at a few points around the tetrahedron. This step is mostly local but may induce sometimes a dichotomic exploration. At the end of this iterative process, one face of the tetrahedron is parallel to $\mathbf{P}$, and thus determines its normal. The remarkable idea is that the algorithm decides itself on-the-fly which points have to be considered for computing the local plane geometry, in opposition to usual plane recognition algorithms [6,8,5,1]. This approach was thus also interesting for analysing 3D shape boundaries.

This paper proposes another algorithm that extracts the characteristics of plane $\mathbf{P}$ given only this predicate and a starting configuration. This new algorithm is also an iterative process that deforms an initial tetrahedron and stops when one face is parallel to $\mathbf{P}$. However, this new algorithm both complements and enhances the former approach. It complements it since it extracts triangular facets onto the plane whose Bezout point is above the facet, while the former algorithm extracts the ones whose Bezout point is not above the facet. See fig. 2 for an example of execution of both algorithms with the same input. It is also a geometrical algorithm using convex hull and Delaunay circumsphere property, while the former was mostly arithmetic. It enhances it for several reasons. First, its theoretical complexity is slightly better (it drops a log factor) and it is faster also in practice. Second, we control the position of the evolving tetrahedron, which always stays around the starting point. Third, each step is purely local and tests the predicate on only six points. Fourth, it can detect planarity defects onto digital surfaces that are not digital planes. Last, a variant of the proposed algorithm *almost always* produce directly a reduced basis of the lattice of upper leaning points, without lattice reduction. Of course, this algorithm presents one disadvantage with respect to the former one: the starting configuration must lie at a reentrant corner of the plane, more precisely below the Bezout point of the plane. If it starts at another corner, then the algorithm will stop sooner and outputs only an approximation of the normal of $\mathbf{P}$.

The paper is organized as follows. First, we give basic definitions and present our new algorithm. Second we show its correctness and exhibit worst-case upper-bound for its time complexity. Third we study how often this algorithm extracts a reduced basis of the lattice of upper leaning points and present a variant — with the same properties — that returns almost always a reduced basis. Afterwards we exploit this algorithm to determine the linear geometry of digital surfaces, and we show how to deal with starting configurations that are not under the Bezout point. Finally the pros and cons of this algorithm are discussed and several research directions are outlined.

Algorithm 1: Extracts a triangle aligned with the digital plane $\mathbf{P}$ by successive convex hull computations.

Input: a predicate "$\boldsymbol{x} \in \mathbf{P}$?", an initial triangle $\mathbf{T}^{(0)}$
1 $\mathbf{T} \leftarrow \mathbf{T}^{(0)}$;
2 **while** $\Sigma_{\mathbf{T}} \cap \mathbf{P} \neq \emptyset$ **do**
3 Compute the convex hull of $\mathbf{T} \cup (\Sigma_{\mathbf{T}} \cap \mathbf{P})$;
4 Find $\mathbf{T}'$, defined as the upper triangular facet intersected by $[\boldsymbol{pq}]$;
5 $\mathbf{T} \leftarrow \mathbf{T}'$;
6 **return** $\mathbf{T}$;

2 Notations and algorithm

In this section, we introduce a few notations before presenting our new algorithm. A digital plane is defined as the set

$$\mathbf{P} = \{\boldsymbol{x} \in \mathbb{Z}^3 | \mu \leq \boldsymbol{x} \cdot \mathbf{N} < \mu + \mathbf{s} \cdot \mathbf{N}\},$$

where $\mathbf{N} \in \mathbb{Z}^3$ is the *normal vector* whose components (a, b, c) are relatively prime, $\mu \in \mathbb{Z}$ is the *intercept*, $\mathbf{s}$ is the *shift vector*, equal to $(\pm 1, \pm 1, \pm 1)$ in the standard case.

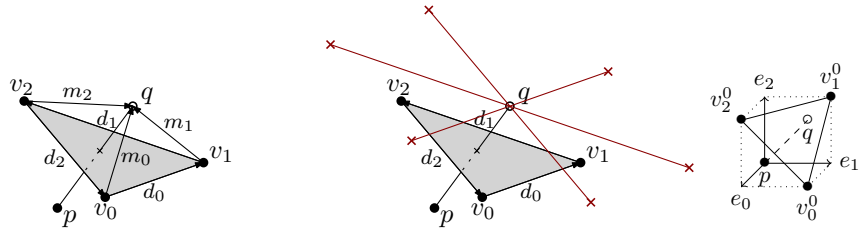
By translation, we assume w.l.o.g. that $\mu = 0$. Moreover, by symmetry, we assume w.l.o.g. that the normal of the plane lies in the first octant, i.e. its components are positive. We also exclude cases where a component is null since then it falls back to a 2D algorithm. Thus, $a, b, c > 0$ and $\mathbf{s} = (1, 1, 1)$. Finally, we denote by $\omega := \mathbf{s} \cdot \mathbf{N} = a + b + c$ the *thickness* of the standard digital plane $\mathbf{P}$.

The above definition of digital plane suggest to see the space as partitionned into layers of coplanar points, orthogonal to $\mathbf{N}$. The value $\boldsymbol{x} \cdot \mathbf{N}$, called *height*, is a way of sorting these layers in direction $\mathbf{N}$. Points of height 0 (resp. $\omega - 1$), which are extreme in $\mathbf{P}$, are called *lower leaning points* (resp. *upper leaning points*). Points of height ω, the closest ones above $\mathbf{P}$, are called *Bezout points*.

Algorithm 1 finds $\mathbf{N}$ given a predicate "is $\boldsymbol{x} \in \mathbf{P}$?". The algorithm starts at any reentrant corner as follows: the *corner point* $\boldsymbol{p}$ is in $\mathbf{P}$; the *shifted point* $\boldsymbol{q} := \boldsymbol{p} + \boldsymbol{s}$ is not in $\mathbf{P}$ since $\boldsymbol{s} \cdot \mathbf{N}$ is the thickness of $\mathbf{P}$; the initial *triangle* $\mathbf{T}^{(0)} := (\boldsymbol{v}_k^{(0)})_{k \in \{0,1,2\}}$ is such that $\forall k, \boldsymbol{v}_k^{(0)} := \boldsymbol{p} + \boldsymbol{e}_k + \boldsymbol{e}_{k+1}$ (fig. 1.c). It is easy to check that $\mathbf{T}^{(0)} \subset \mathbf{P}$ for any $\boldsymbol{p}$ such that $0 \leq \boldsymbol{p} \cdot \mathbf{N} < \min\{a, b, c\}$, which corresponds exactly to reentrant corners onto a standard digital plane. The algorithm then updates this initial triangle and iteratively aligns it with the plane by calling the above predicate for well-chosen points, until the solution equals $\mathbf{N}$.

At step $i \in \mathbb{N}$, the solution is described by a triangle denoted by $\mathbf{T}^{(i)}$. Algorithm 1 is designed so that $\forall i, \mathbf{T}^{(i)}$ is included in $\mathbf{P}$ and is intersected by segment $[\boldsymbol{pq}]$. Segment $[\boldsymbol{pq}]$ thus forces the exploration to be local.

Let k be an integer taken modulo 3, i.e. $k \in \mathbb{Z}/3\mathbb{Z}$. The three counterclockwise oriented vertices of $\mathbf{T}^{(i)}$ are denoted by $\boldsymbol{v}_k^{(i)}$ (fig. 1.a). For sake of clarity, we will



(a) triangle (b) neighborhood (c) step 0

Fig. 1. Main notations are illustrated in (a) and (b) (iteration number as exponent (i) is dropped for sake of clarity). The starting triangle is illustrated in (c).

write $\forall k$ instead of $\forall k \in \mathbb{Z}/3\mathbb{Z}$. We also introduce the following vectors (fig. 1.a):

$$\forall k,\ \boldsymbol{m}_k^{(i)} := \boldsymbol{q} - \boldsymbol{v}_k^{(i)} \tag{1}$$

$$\forall k,\ \boldsymbol{d}_k^{(i)} := \boldsymbol{v}_{k+1}^{(i)} - \boldsymbol{v}_k^{(i)} = \boldsymbol{m}_k^{(i)} - \boldsymbol{m}_{k+1}^{(i)}. \tag{2}$$

The normal of $\mathbf{T}^{(i)}$, denoted by $\hat{\mathbf{N}}(\mathbf{T}^{(i)})$, is merely defined by the cross product between two consecutive edge vectors of $\mathbf{T}^{(i)}$, i.e. $\hat{\mathbf{N}}(\mathbf{T}^{(i)}) = \boldsymbol{d}_0^{(i)} \times \boldsymbol{d}_1^{(i)}$.

In order to improve the guess at step i, the algorithm checks if the points of a small neighborhood around $\boldsymbol{q}$, parallel and above $\mathbf{T}^{(i)}$, belong to $\mathbf{P}$ or not. This neighborhood is defined as follows (fig. 1.b):

$$\Sigma_{\mathbf{T}^{(i)}} := \left\{ \boldsymbol{q} \pm \boldsymbol{d}_k^{(i)} \right\}_{k \in \{0,1,2\}}. \tag{3}$$

The algorithm then computes $\Sigma_{\mathbf{T}^{(i)}} \cap \mathbf{P}$ by making six calls to the predicate. In algorithm 1, the new triangle $\mathbf{T}^{(i+1)}$ is simply defined as the other triangular facet of the convex hull of $\mathbf{T}^{(i)} \cup (\Sigma_{\mathbf{T}^{(i)}} \cap \mathbf{P})$ which intersects $[\boldsymbol{pq}]$. See fig. 2 for an example. The algorithm stops when $\Sigma_{\mathbf{T}^{(n)}} \cap \mathbf{P}$ is empty. We show in theorem 2 that, when $\boldsymbol{p}$ is a lower leaning point (its height is 0), the vertices of the last triangle $\mathbf{T}^{(n)}$ are upper leaning points of $\mathbf{P}$ (their height is $\omega - 1$). Corollary 1 implies that the normal to $\mathbf{T}^{(n)}$ is the normal to $\mathbf{P}$ and corollary 2 implies that triangle edges form a basis of the lattice of upper leaning points to $\mathbf{P}$.

3 Validity and complexity

In this section, we first prove the two invariants of algorithm 1 and fully characterize its main operation (lines 3-5 of algorithm 1). Then we prove that if $\boldsymbol{p}$ is a lower leaning point, algorithm 1 retrieves the true normal $\mathbf{N}$ in less than $\omega - 3$ steps.

Let us assume that the algorithm always terminate in a finite number of steps whatever the starting point and let n be the last step (the proof is postponed, see theorem 1).

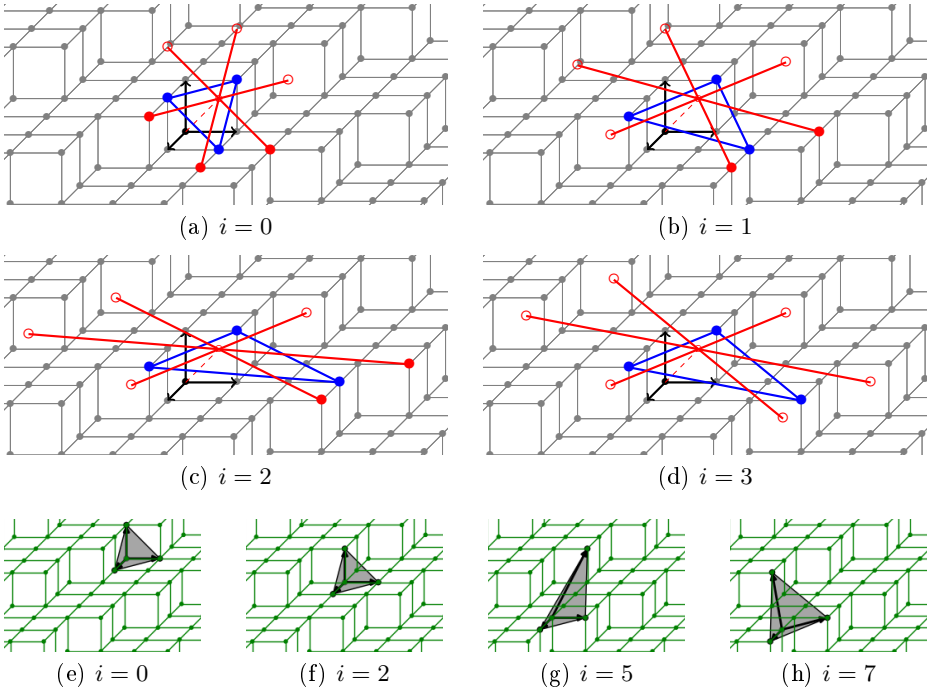


Fig. 2. Illustration of the running of algorithm 1 and algorithm from [7] on a digital plane of normal vector $(1, 2, 5)$. Images (a) to (d) shows the four iterations of algorithm 1 starting from the origin. For $i \in \{0, 1, 2, 3\}$, triangles $\mathbf{T}^{(i)}$ are in blue, whereas neighborhood points are depicted with red disks (resp. circles) if they belong (resp. do not belong) to $\mathbf{P}$. In (d), the normal of the last triangle is $(1, 2, 5)$. Images (e) to (h) shows iterations 0 (initial), 2, 5 and 7 (final) of the algorithm from [7]. The initial tetrahedron (a) is placed at the origin and the final one (h) has an upper triangle with normal vector $(1, 2, 5)$.

3.1 Algorithm invariants

The following two invariants are easy to prove by induction since by construction, each new triangle is chosen as a triangular facet of the convex hull of some subset of $\mathbf{P}$, which is intersected by $]\boldsymbol{pq}]$.

Invariant 1 $\forall i \in \{0, \ldots, n\},\ \forall k,\ \boldsymbol{v}_k^{(i)} \in \mathbf{P}$.

Invariant 2 $\forall i \in \{0, \ldots, n\}$, *the interior of* $\mathbf{T}^{(i)}$ *is intersected by* $]\boldsymbol{pq}]$.

The only difficulty in invariant 2 is to show that the triangle boundary is never intersected by $]\boldsymbol{pq}]$. Below, for lack of space, we only show by contradiction that $]\boldsymbol{pq}]$ does not intersect any edge of $\mathbf{T}^{(i+1)}$, if invariant 2 is assumed to be true for $i \in \{0, \ldots, n-1\}$.

Proof. If $]\boldsymbol{pq}]$ and some edge $[\boldsymbol{xy}]$ of $\mathbf{T}^{(i+1)}$ intersect, points $\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{x}$ and $\boldsymbol{y}$ are coplanar. But, since the boundary of $\mathbf{T}^{(i)}$ is not intersected by $]\boldsymbol{pq}]$ and since points of $\Sigma_{\mathbf{T}^{(i)}}$ are located on lines that are parallel to the sides of $\mathbf{T}^{(i)}$, $\boldsymbol{x}$ and $\boldsymbol{y}$ must be opposite points in $\Sigma_{\mathbf{T}^{(i)}}$, i.e. $\boldsymbol{x} := \boldsymbol{q} + \boldsymbol{d}_l^{(i)}$ for some $l \in \{0, 1, 2\}$ and $\boldsymbol{y} = \boldsymbol{q} - \boldsymbol{d}_l^{(i)}$.

However, since $\boldsymbol{q} \notin \mathbf{P}$, $\boldsymbol{x}$ and $\boldsymbol{y}$ cannot be both in $\mathbf{P}$ by linearity and thus cannot be the ends of an edge of $\mathbf{T}^{(i+1)}$, which is included in $\mathbf{P}$ by invariant 1. □

3.2 Operation characterization

The two following lemmas characterize the operation that transforms a triangle into the next one (lines 3-5 of algorithm 1).

Lemma 1. $\forall i \in \{0, \ldots, n-1\}$, $\mathbf{T}^{(i)}$ *and* $\mathbf{T}^{(i+1)}$ *share exactly one or two vertices.*

Proof. Let γ be the number of common vertices between $\mathbf{T}^{(i)}$ and $\mathbf{T}^{(i+1)}$. Since $\mathbf{T}^{(i)}$ and $\mathbf{T}^{(i+1)}$ have three vertices each, $0 \leq \gamma \leq 3$, but we prove below that (i) $\gamma \neq 3$ and (ii) $\gamma \neq 0$.

Inequality (i) is trivial. Indeed, since the volume of the convex hull of $\mathbf{T}^{(i)} \cup (\Sigma_{\mathbf{T}^{(i)}} \cap \mathbf{P})$ is not empty for $i \in \{0, \ldots, n-1\}$ (see algorithm 1, l. 2), we necessarily have $\mathbf{T}^{(i)} \neq \mathbf{T}^{(i+1)}$, which implies that $\gamma \neq 3$.

In order to prove inequality (ii), let $\mathcal{P}$ be the plane passing by the points of $\Sigma_{\mathbf{T}^{(i)}}$ (points of $\Sigma_{\mathbf{T}^{(i)}}$ are coplanar by (2) and (3)). Note that $\boldsymbol{q} \in \mathcal{P}$ by (3).

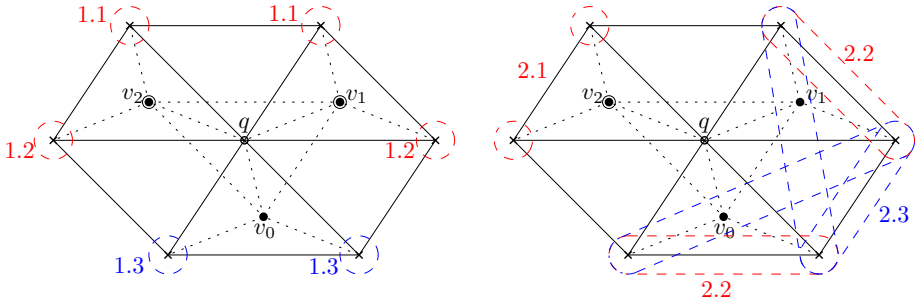
Let us assume that $\mathbf{T}^{(i+1)}$ is included in $\mathcal{P}$. By invariant 1, $\mathbf{T}^{(i+1)} \subset \mathbf{P}$. However, $\boldsymbol{q} \notin \mathbf{P}$ by definition of $\boldsymbol{q}$. As a consequence, the upper leaning plane of $\mathbf{P}$, i.e. $\{\boldsymbol{x} \in \mathbb{R}^3 | \boldsymbol{x} \cdot \mathbf{N} = \omega - 1\}$, strictly separates $\mathbf{T}^{(i+1)}$ from $\boldsymbol{q}$ in the plane $\mathcal{P}$. In addition, $\boldsymbol{p}$ is located strictly below $\mathbf{T}^{(i)}$ by invariant 2 and is thus also below $\mathcal{P}$, which is parallel and above $\mathbf{T}^{(i)}$ by definition. As a consequence, $\mathbf{T}^{(i+1)}$ is clearly not intersected by $]\boldsymbol{pq}]$, which contradicts invariant 2. We conclude that $\mathbf{T}^{(i+1)}$ is not included in $\mathcal{P}$, which means that $\gamma \neq 0$. □

The following lemma fully characterizes the main operation of algorithm 1:

Lemma 2. $\forall i \in \{0, \ldots, n-1\}$, $\forall k$, $\begin{cases} \textit{either } \boldsymbol{v}_k^{(i+1)} = \boldsymbol{v}_k^{(i)}, \\ \textit{or } \boldsymbol{v}_k^{(i+1)} = \boldsymbol{v}_k^{(i)} + \boldsymbol{m}_l^{(i)}, l \neq k. \end{cases}$

We know by lemma 1 that $\mathbf{T}^{(i)}$ and $\mathbf{T}^{(i+1)}$ share one or two vertices. The proof is thus in two parts. In each part however, we use infinite cones of apex $\boldsymbol{q}$ to check whether invariant 2 is true or not. Let us define $\forall i \in \{0, \ldots, n\}$, ${\mathcal{T}_\infty}^{(i)}$ as the set of infinite rays emanating from $\boldsymbol{q}$ and intersecting the triangular facet $\mathbf{T}^{(i)}$. Invariant 2 is equivalent to the following

invariant 2': $\forall i \in \{0, \ldots, n\}$, the interior of ${\mathcal{T}_\infty}^{(i)}$ contains $\boldsymbol{p}$.



(a) $\sharp(\mathbf{T}^{(i)} \cap \mathbf{T}^{(i+1)}) = 2$ (b) $\sharp(\mathbf{T}^{(i)} \cap \mathbf{T}^{(i+1)}) = 1$

Fig. 3. Illustration of the proof of lemma 2. Case where $\mathbf{T}^{(i)}$ and $\mathbf{T}^{(i+1)}$ share $\boldsymbol{v}_1^{(i)}$ and $\boldsymbol{v}_2^{(i)}$ in (a), but only $\boldsymbol{v}_2^{(i)}$ in (b) (exponent (i) is omitted in the figures).

Proof. We first assume that $\mathbf{T}^{(i)}$ and $\mathbf{T}^{(i+1)}$ share two vertices. Let us assume w.l.o.g. that the index of the vertex of $\mathbf{T}^{(i)}$ that is not in $\mathbf{T}^{(i+1)}$ is 0.

Since $\boldsymbol{v}_0^{(i+1)} \in \Sigma_{\mathbf{T}^{(i)}} \cap \mathbf{P}$, there are six cases for $\boldsymbol{v}_0^{(i+1)}$, which are grouped below two by two (see fig. 3.a). Invariant 1 is true for all cases. We show however by contradition that the first four cases (items 1.1 and 1.2) are not possible because otherwise, invariant 2' is not true:

1.1 Let us assume that $\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_l^{(i)} + \boldsymbol{m}_0^{(i)}, l \in \{1,2\}$. In these cases, ${\mathcal{T}_\infty}^{(i+1)}$ is adjacent to ${\mathcal{T}_\infty}^{(i)}$ along facet $(\boldsymbol{q}, \boldsymbol{v}_1^{(i)}, \boldsymbol{v}_2^{(i)})$. We conclude that the interior of ${\mathcal{T}_\infty}^{(i+1)}$ and ${\mathcal{T}_\infty}^{(i)}$ are disjoint and cannot both contains $\boldsymbol{p}$, which raises a contradiction.

1.2 If we assume now that $\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_1^{(i)} + \boldsymbol{m}_2^{(i)}$ or $\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_2^{(i)} + \boldsymbol{m}_1^{(i)}$. In these cases, $\boldsymbol{v}_0^{(i+1)}$ lies on the plane passing by $\boldsymbol{q}$, $\boldsymbol{v}_1^{(i)}$ and $\boldsymbol{v}_2^{(i)}$. We conclude that the interior of ${\mathcal{T}_\infty}^{(i+1)}$ is empty and cannot contains $\boldsymbol{p}$, a contradiction.

1.3 Hence, $\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_0^{(i)} + \boldsymbol{m}_l^{(i)}, l \in \{1,2\}$. In this case, ${\mathcal{T}_\infty}^{(i+1)}$, which contains ${\mathcal{T}_\infty}^{(i)}$, contains $\boldsymbol{p}$.

We now assume that $\mathbf{T}^{(i)}$ and $\mathbf{T}^{(i+1)}$ share one vertex. Let us assume w.l.o.g. that its index is 2.

By definition, $\boldsymbol{v}_0^{(i+1)}, \boldsymbol{v}_1^{(i+1)} \in \Sigma_{\mathbf{T}^{(i)}} \cap \mathbf{P}$. There are $\binom{6}{2} = 15$ cases to consider for $\boldsymbol{v}_0^{(i+1)}$ and $\boldsymbol{v}_1^{(i+1)}$, which are grouped below into four different items. See fig. 3.b for the last three items. Invariant 1 is true for all cases. We show however that invariant 2 is true for only the three cases described at item 2.3:

2.0 (3 cases) Let us assume that $\boldsymbol{v}_0^{(i+1)}$ and $\boldsymbol{v}_1^{(i+1)}$ are opposite with respect to $\boldsymbol{q}$, i.e. one is equal to $\boldsymbol{q} + \boldsymbol{d}_k^{(i+1)}$, $k \in \{0,1,2\}$, whereas the other is equal to $\boldsymbol{q} - \boldsymbol{d}_k^{(i+1)}$. Since $\boldsymbol{q} \notin \mathbf{P}$, $\boldsymbol{q} + \boldsymbol{d}_k^{(i+1)}$ and $\boldsymbol{q} - \boldsymbol{d}_k^{(i+1)}$ cannot be both in $\mathbf{P}$ by linearity, which raises a contradiction.

2.1 (7 cases) Let us assume that $\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_2^{(i)} + \boldsymbol{m}_l^{(i)}, l \in \{1,2\}$ or $\boldsymbol{v}_1^{(i+1)} = \boldsymbol{v}_2^{(i)} + \boldsymbol{m}_l^{(i)}, l \in \{1,2\}$. In all cases, ${\mathcal{T}_\infty}^{(i+1)}$ is adjacent to ${\mathcal{T}_\infty}^{(i)}$ by edge

$(\boldsymbol{q}, \boldsymbol{v}_2^{(i)})$, by facet $(\boldsymbol{q}, \boldsymbol{v}_0^{(i)}, \boldsymbol{v}_2^{(i)})$ or by facet $(\boldsymbol{q}, \boldsymbol{v}_1^{(i)}, \boldsymbol{v}_2^{(i)})$. As a consequence, the interior of ${\mathcal{T}_\infty}^{(i+1)}$ cannot contains $\boldsymbol{p}$.

2.2 (2 cases) Let us assume that $\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_l^{(i)} + \boldsymbol{m}_{l+1}^{(i)}$ and $\boldsymbol{v}_1^{(i+1)} = \boldsymbol{v}_l^{(i)} + \boldsymbol{m}_{l+2}^{(i)}$, for $l \in \{0, 1\}$. In both cases, ${\mathcal{T}_\infty}^{(i+1)}$ is adjacent to ${\mathcal{T}_\infty}^{(i)}$ by edge $(\boldsymbol{q}, \boldsymbol{v}_l^{(i)})$. As a consequence, the previous argument also applies.

2.3 (3 cases) ${\mathcal{T}_\infty}^{(i+1)}$ contains ${\mathcal{T}_\infty}^{(i)}$ and thus contains $\boldsymbol{p}$ in the following cases: $\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_0^{(i)} + \boldsymbol{m}_{l_0}^{(i)}, l_0 \in \{1, 2\}$ and $\boldsymbol{v}_1^{(i+1)} = \boldsymbol{v}_1^{(i)} + \boldsymbol{m}_{l_1}^{(i)}, l_1 \in \{0, 2\}$, with $(l_0, l_1) \neq (1, 0)$.

□

Lemma 3. *Let $\mathbf{M}^{(i)}$ be the 3×3 matrix that consists of the three row vectors $(\boldsymbol{m}_k^{(i)})_{k \in \{0,1,2\}}$. Then, $\forall i \in \{0, \ldots, n\}$, $\det(\mathbf{M}^{(i)}) = 1$.*

Proof. We can easily check that $\det(\mathbf{M}^{(0)}) = 1$.

We now prove that if $\det(\mathbf{M}^{(i)}) = 1$ for $i \in \{0, \ldots, n-1\}$, then $\det(\mathbf{M}^{(i+1)}) = 1$. By lemma 1 and lemma 2, we have $\boldsymbol{v}_k^{(i+1)} = \boldsymbol{v}_k^{(i)} + \boldsymbol{m}_l^{(i)} \Leftrightarrow \boldsymbol{m}_k^{(i+1)} = \boldsymbol{m}_k^{(i)} - \boldsymbol{m}_l^{(i)}, l \neq k$ for at most two rows over three, while the remaining one or two rows correspond to identity. Such matrix operations do not change the determinant, which concludes. □

3.3 Termination

In the following proofs, we compare the position of the points along direction $\mathbf{N}$. For the sake of simplicity, we use the bar notation $\bar{\cdot}$ above any vector $\boldsymbol{x}$ to denote its height relative to $\mathbf{N}$. Otherwise said, $\overline{\boldsymbol{x}} := \boldsymbol{x} \cdot \mathbf{N}$. Even if $\mathbf{N}$ is not known, $\overline{\boldsymbol{q}} \geq \omega$ by definition and for all $\boldsymbol{x} \in \mathbf{P}$, $0 \leq \overline{\boldsymbol{x}} < \omega$.

Theorem 1. *The number of steps in algorithm 1 is bounded from above by $\omega - 3$.*

Proof. First, it is easy to see that $\forall k, \boldsymbol{m}_k^{(0)} = \boldsymbol{e}_{k+2}$. Thus, $\sum_k \overline{\boldsymbol{m}}_k^{(0)} = \omega$.

Let us recall that $\forall i \in \{0, \ldots, n\}$, $\forall k$, $\boldsymbol{m}_k^{(i)} = \boldsymbol{q} - \boldsymbol{v}_k^{(i)}$. We have $\overline{\boldsymbol{q}} \geq \omega$ by definition of $\boldsymbol{q}$ and $\forall i \in \{0, \ldots, n\}$, $\forall k$, $0 \leq \overline{\boldsymbol{v}}_k^{(i)} < \omega$ by invariant 1 and by definition of $\mathbf{P}$. As a consequence, $\forall i \in \{0, \ldots, n\}$, $\forall k$, $\overline{\boldsymbol{m}}_k^{(i)} > 0$ and $\sum_k \overline{\boldsymbol{m}}_k^{(i)} \geq 3$.

Moverover, by lemma 2 and since any $\overline{\boldsymbol{m}}_k^{(i)}$ is strictly positive $\forall i \in \{0, \ldots, n\}$, we clearly have $\forall i \in \{0, \ldots, n-1\}$, $\sum_k \overline{\boldsymbol{m}}_k^{(i)} > \sum_k \overline{\boldsymbol{m}}_k^{(i+1)}$.

The sequence $(\sum_k \overline{\boldsymbol{m}}_k^{(i)})_{i=0,\ldots,n}$ is thus a strictly decreasing sequence of integers between ω and 3. □

Remark that this bound is reached when running the algorithm on a plane with normal $\mathbf{N}(1, 1, r)$. We now give the main result when the starting point $\boldsymbol{p}$ is a lower leaning point, i.e. $\overline{\boldsymbol{p}} = 0$. Note that in this case $\overline{\boldsymbol{q}} = \omega$. Since we focus below on the last step n, we omit the exponent (n) in the proofs to improve their readability.

Theorem 2. *If $\boldsymbol{p}$ is a lower leaning point (i.e. $\overline{\boldsymbol{p}} = 0$ and thus $\overline{\boldsymbol{q}} = \omega$), the vertices of the last triangle are upper leaning points, i.e. $\forall k, \overline{\boldsymbol{v}}_k^{(n)} = \omega - 1$.*

Proof. If there exists $k \in \{0,1,2\}$ such that $\overline{\boldsymbol{d}}_k \neq 0$, then either (i) $\overline{\boldsymbol{d}}_k < 0$ or (ii) $\overline{\boldsymbol{d}}_k > 0$. Since $\overline{\boldsymbol{q}} = \omega$ and $|\overline{\boldsymbol{d}}_k| < \omega$, either (i) $\boldsymbol{q} + \boldsymbol{d}_k \in \mathbf{P}$ or (ii) $\boldsymbol{q} - \boldsymbol{d}_k \in \mathbf{P}$. This implies that $\Sigma_{\mathbf{T}} \cap \mathbf{P} \neq \emptyset$, which is a contradiction because $\Sigma_{\mathbf{T}} \cap \mathbf{P} = \emptyset$ at the last step (see algorithm 1, l. 2). As a consequence, $\forall k, \overline{\boldsymbol{d}}_k = 0$ and $\forall k, \overline{\boldsymbol{m}}_k = \alpha$, a strictly positive integer.

Let us denote by $\mathbf{1}$ the vector $(1,1,1)^T$. We can write the last system as $\mathbf{MN} = \alpha\mathbf{1}$. Since $\mathbf{M}$ is invertible (because $\det(\mathbf{M}) = 1$ by lemma 3), $\mathbf{N} = \mathbf{M}^{-1}\alpha\mathbf{1}$ and as a consequence $\alpha = 1$ (because the components of $\mathbf{N}$ are relatively prime and $\mathbf{M}^{-1}$ is unimodular).

We conclude that $\forall k, \overline{\boldsymbol{m}}_k = 1$ and, straightforwardly, $\overline{\boldsymbol{v}}_k = \omega - 1$. □

The following two corollaries are easily derived from lemma 3 and theorem 2.

Corollary 1. *If $\boldsymbol{p}$ is a lower leaning point, the normal of the last triangle is equal to $\mathbf{N}$, i.e. $\hat{\mathbf{N}}(\mathbf{T}^{(n)}) = \mathbf{N}$.*

Corollary 2. *If $\boldsymbol{p}$ is a lower leaning point, $(\boldsymbol{d}_0^{(n)}, \boldsymbol{d}_1^{(n)})$ is a basis of the lattice of upper leaning points $\{\boldsymbol{x} \in \mathbf{P} | \overline{\boldsymbol{x}} = \omega - 1\}$.*

4 Lattice reduction and Delaunay condition

Corollary 2 implies that the edge vectors of the last triangle form a basis of the lattice of upper leaning points to $\mathbf{P}$.

Let us recall that a basis $(\boldsymbol{x}, \boldsymbol{y})$ is reduced if and only if $\|\boldsymbol{x}\|, \|\boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\| \leq \|\boldsymbol{x} + \boldsymbol{y}\|$. Given $(\boldsymbol{x}, \boldsymbol{y})$ a basis of a two dimensional lattice, there is an iterative algorithm to compute a reduced basis from $(\boldsymbol{x}, \boldsymbol{y})$. This algorithm consists in replacing the longest vector among $(\boldsymbol{x}, \boldsymbol{y})$ by the shortest one among $\boldsymbol{x} + \boldsymbol{y}$ and $\boldsymbol{x} - \boldsymbol{y}$, if it is smaller.

We have found many examples for which the basis returned by algorithm 1 is not reduced, even if we take the two shortest vectors in $\{\boldsymbol{d}_k^{(n)}\}_{k \in \{0,1,2\}}$ (see fig. 4.a for an example). To cope with this problem, we can either apply the above reduction algorithm to the returned basis, or keep triangles as "small" as possible so that the last triangle leads to a reduced basis, without any extra reduction steps. To achieve this aim, we use a variant of algorithm 1; see algorithm 2.

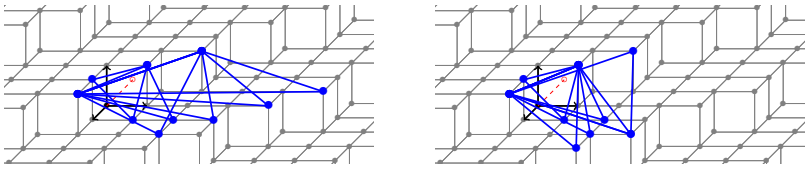
Note that in algorithm 2, we compute the convex hull of $T \cup S^\star$, but since $S^\star \subset \Sigma_{\mathbf{T}} \cap \mathbf{P}$ and since $\Sigma_{\mathbf{T}} \cap \mathbf{P} \neq \emptyset \Rightarrow S^\star \neq \emptyset$, all the results of sec. 3 remain valid.

In fig. 4, the results of the two algorithms for the digital plane of normal $(2,3,9)$ are compared. The triangles returned by algorithm 2 are more compact and closer to segment $[\boldsymbol{pq}]$. The last triangle returned by algorithm 2 leads to a reduced basis, but this is not true for algorithm 1.

Algorithm 2: Variant of algorithm 1 based on a Delaunay condition.

Input: a predicate "$\boldsymbol{x} \in \mathbf{P}$?", an initial triangle $\mathbf{T}^{(0)}$

$\mathbf{T} \leftarrow \mathbf{T}^{(0)}$;
while $\Sigma_{\mathbf{T}} \cap \mathbf{P} \neq \emptyset$ **do**
  Compute the set $S^\star$ of points $s^\star \in (\Sigma_{\mathbf{T}} \cap \mathbf{P})$ such that the circumsphere of $\mathbf{T} \cup \{s^\star\}$ does not include any point $s \in (\Sigma_{\mathbf{T}} \cap \mathbf{P})$ in its interior ;
  Compute the convex hull of $\mathbf{T} \cup S^\star$;
  Find $\mathbf{T}'$, defined as the upper triangular facet intersected by $[\boldsymbol{pq}]$;
  $\mathbf{T} \leftarrow \mathbf{T}'$;
return $\mathbf{T}$;



(a) alg. 1 (b) alg. 2

Fig. 4. Triangles returned by algorithm 1 in (a) and algorithm 2 in (b) for a digital plane with normal $(2, 3, 9)$ (the corner point $\boldsymbol{p}$ is set to the origin). In (b), triangles returned by algorithm 2 are smaller and the last one defines a reduced basis.

algorithm	algorithm 1	algorithm 2
avg. nb. steps	21.8	20.9
nb. non-reduced	4803115 (73%)	**924 (0.01%)**
avg. nb. reductions	5.5	1

In practice, algorithm 2 drastically reduces the cases of non-reduced basis. We have compared the two variants for normal vectors ranging from $(1, 1, 1)$ to $(200, 200, 200)$. There are 6578833 vectors with relatively prime components in this range. It turns out that less than 0.01% of the basis returned by algorithm 2 are non-reduced against about 73% for algorithm 1.

5 Applications to digital surfaces

In this section, we consider a set of voxels, Z, where voxels are seen as unit cubes whose vertices are in $\mathbb{Z}^3$. The *digital boundary,* BdZ, is defined as the topological boundary of Z. Since a digital boundary looks locally like a digital plane, it is natural to run our algorithm at each reentrant corner of the digital boundary with predicate "Is $\boldsymbol{x}$ in BdZ" in order to estimate the local normal vector to the volume Z (see fig. 5).

Although we do not go into details here due to space limitations, we can mention the following points:

– Our algorithm works well on digitally convex shapes Z. The set of points $\boldsymbol{x} \in$ BdZ located below some facet $\mathcal{F}$ of Conv(BdZ) whose normal vector

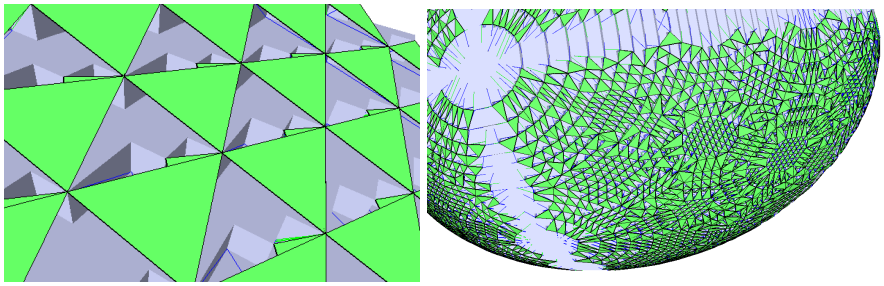


Fig. 5. Our algorithm has been runned at each reentrant corner of a digital plane (left) and an ellipse (right). The last triangle of each run is printed.

is in the first octant and such that points $\boldsymbol{x} + \boldsymbol{s}$ are strictly above $\mathcal{F}$, is a piece of a digital plane, $\mathbf{P}$. If the starting reentrant corner is a lower leaning point of $\mathbf{P}$, the last triangle computed by our algorithm is included in $\mathcal{F}$ (see fig. 5). We call such triangles *patterns* of $\mathbf{P}$.

- When starting from a corner that is not a lower leaning point of a facet of Conv(BdZ), the algorithm returns a triangle called *reduced pattern* of $\mathbf{P}$ which is approximately aligned with the plane.
- It is possible to detect and remove reduced patterns with a simple algorithm. Put in queue every point $\boldsymbol{q}$ of all computed triangles, together with their three vectors $\boldsymbol{m}_k$. Loop on the queue while it is not empty by popping them in sequence. For each popped point $\boldsymbol{q}$ and vectors $\boldsymbol{m}_k$, check if $\boldsymbol{q}$ translated by any of $\boldsymbol{m}_k$ is a shifted point $\boldsymbol{q}'$ of another triangle. In this case, mark $\boldsymbol{q}'$ as reduced pattern and put it back in queue but now with the vectors $\boldsymbol{m}_k$. Otherwise, do nothing. When the algorithm stops, since reduced patterns have shifted points above the shifted point of the lower leaning point, all reduced patterns are marked.
- If the shape is not convex, the algorithm can be adapted to detect planarity defects. The idea is that $\Sigma_{\mathbf{T}} \cap \mathrm{Bd}Z$ should contain at most three elements if it is locally a piece of plane. Moreover these elements must always be consecutive neighbors around $\boldsymbol{q}$ in $\Sigma_{\mathbf{T}}$. We thus stop the algorithm whenever at least one of the two previous conditions fails, meaning that the surface is locally non convex. We also check that each new triangle is separating onto the digital boundary: for any point $\boldsymbol{x} \in \mathrm{Bd}Z$ below a triangle of shift $\boldsymbol{s}$, the point $\boldsymbol{x} + \boldsymbol{s}$ must be above the triangle.

6 Conclusion and perspectives

In this paper, we proposed a new algorithm that computes the parameters of a digital plane. In opposition to usual plane recognition algorithms, this algorithm greedily decides on-the-fly which points have to be considered like in [7]. Compared to [7], this algorithm is however simpler because it consists in iterating one geometrical operation, which is fully characterized in lemma 2. In addition, the

returned solution, which is described by a triangle parallel to the sought plane, is close to the starting point. On one hand, the starting point must projects onto the triangle and on the other hand, the two shortest edge vectors of the triangle almost always form a reduced basis (in the second version of our algorithm).

For the future, we would like to find a theoretically sound way of always getting a reduced basis, without any extra reduction operation. Moreover, we would like to find a variant of our algorithm in order to retrieve complement triangles whose Bezout point is not above the triangle. For sake of completeness, we are also interested in degenerate cases, where at least one component is null. After having achieved these goals, we would have a complete working tool for the analysis of digitally convex surfaces. In order to correctly process concave or saddle parts however, we must precisely associate to a given triangle a piece of digital plane containing the triangle vertices and having the same normal vector than the triangle one. Although there are many such pieces of digital plane for a given triangle, we hope that this work will provide a canonical hierarchy of pieces of digital plane.

References

1. E. Charrier and L. Buzer. An efficient and quasi linear worst-case time algorithm for digital plane recognition. In *Discrete Geometry for Computer Imagery (DGCI'2008)*, volume 4992 of *LNCS*, pages 346–357. Springer, 2008.
2. E. Charrier and J.-O. Lachaud. Maximal planes and multiscale tangential cover of 3d digital objects. In *Proc. Int. Workshop Combinatorial Image Analysis (IWCIA'2011)*, volume 6636 of *Lecture Notes in Computer Science*, pages 132–143. Springer Berlin / Heidelberg, 2011.
3. 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(2):471–502, February 2007.
4. H. Doerksen-Reiter and I. Debled-Rennesson. Convex and concave parts of digital curves. In R. Klette, R. Kozera, L. Noakes, and J. Weickert, editors, *Geometric Properties for Incomplete Data*, volume 31 of *Computational Imaging and Vision*, pages 145–160. Springer, 2006.
5. Y. Gérard, I. Debled-Rennesson, and P. Zimmermann. An elementary digital plane recognition algorithm. *Discrete Applied Mathematics*, 151(1):169–183, 2005.
6. C. E. Kim and I. Stojmenović. On the recognition of digital planes in three-dimensional space. *Pattern Recognition Letters*, 12(11):665–669, 1991.
7. J.-O. Lachaud, X. Provençal, and T. Roussillon. An output-sensitive algorithm to compute the normal vector of a digital plane. *Theoretical Computer Science*, 2015. Accepted. To appear.
8. P. Veelaert. Digital planarity of rectangular surface segments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):647–652, 1994.