A New Lattice-based Plane-probing Algorithm ⋆

Jui-Ting Lu[0000−0002−5341−1490], Tristan Roussillon[0000−0003−2524−3685], and David Coeurjolly[0000−0003−3164−8697]

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

Abstract. Plane-probing algorithms have become fundamental tools to locally capture arithmetical and geometrical properties of digital surfaces (boundaries of a connected set of voxels), and especially normal vector information. On a digital plane, the overall idea is to consider a local pattern, a triangle, that is expanded starting from a point of interest using simple probes of the digital plane with a predicate "Is a point x in the digital plane?". Challenges in plane-probing methods are to design an algorithm that terminates on a triangle with several geometrical properties: its normal vector should match with the expected one for digital plane (correctness), the triangle should be as compact as possible (acute or right angles only), and probes should be as close as possible to the source point (locality property). In addition, we also wish to minimize the number of iterations or probes during the computations. Existing methods provide correct outputs but only experimental evidence for these properties. In this paper, we present a new plane-probing algorithm that is theoretically correct on digital planes, and with better experimental compactness and locality than existing solutions. Additional properties of this new approach also suggest that theoretical proofs of the aforementioned geometrical properties could be achieved.

Keywords: Digital Plane Recognition · Plane-Probing Algorithm · Lattice Reduction

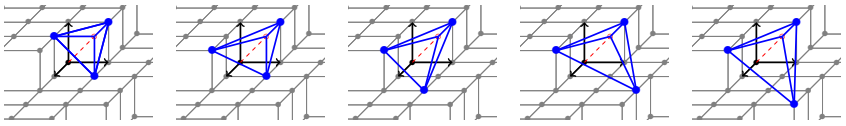
1 Introduction

A digital surface is a quadrangular mesh that corresponds to the boundary of a union of regularly spaced unit cubes (voxels). We are interested in processing the geometry of such surfaces, for instance to recognize local elementary structures such as digital plane segments [1, 4, 5, 14], or to estimate some differential quantities [2, 3]. When performing such local computations, we usually need to capture local geometric properties of the surface around a given point. This can be done either by considering a fixed neighborhood, e.g., using a Euclidean ball with fixed radius, or by adapting such neighborhood to local geometric properties. Probing algorithms target the latter case by iteratively growing a pattern with update rules given from *probing* of the geometry. Plane-probing algorithms analyze digital planes [10] without imposing a parameter that controls the size of the patch [6–9, 11]. The key objective of these techniques is to exploit arithmetical and geometrical properties of the digital plane being *explored* in order to retrieve its unknown arithmetical parameters, e.g., its normal vector. When applied on generic

⋆ This work has been partly funded by PARADIS ANR-18-CE23-0007-01 research grant.

non-planar (implicit or explicit) digital surfaces, outputs of plane-probing algorithms could be used to locally estimate the normal bundle of the surface, or could be a key ingredient for surface reconstruction [9].

Plane-probing algorithms can mainly be categorized into two types: tetrahedra-based plane-probing algorithms [8, 9, 11] and parallelepiped-based [6]. In this paper, we focus on tetrahedra-based plane-probing algorithms applied on a digital plane. Those algorithms update the three vertices of the tetrahedron base until it matches the normal of the digital plane. Meanwhile, the apex of the tetrahedron remains fixed (see Fig. 1).



(a) (b) (c) (d) (e)

Fig. 1: The evolution (from left to right) of a tetrahedra-based plane-probing algorithm for normal $(1, 2, 5)$.

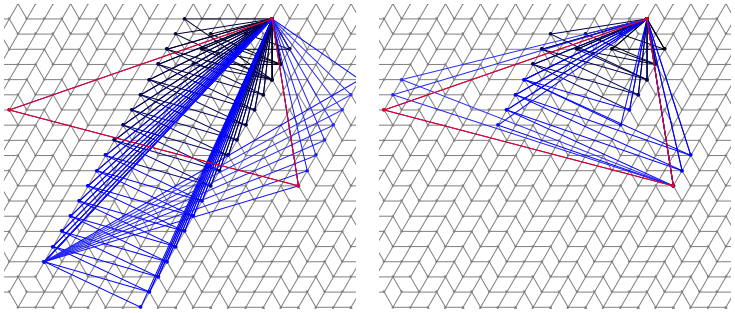
Among existing approaches to update the tetrahedra vertices, we can mention the H-algorithm and the R-algorithm [9]. The main advantage of such approaches is their proximity to the source point. Indeed, the apex of the tetrahedron does not move, stays right above the starting point and always projects into the opposite face, i.e., the base, in the direction of the starting point (see Fig. 1 and [9, Lemma 4]). However, we do not have an upper bound of the probed area. A comparison of H-algorithm and R-algorithm is illustrated in Fig. 2, where only the triangles corresponding to the bases are drawn. The outputs of the two algorithms are identical, but H-algorithm probes a larger region than R-algorithm does. We also spot more obtuse triangles in H-algorithm's evolution. Furthermore, as stated in [12], our new algorithm also leads to additional theoretical results such as the minimality of the lattice generated by the last triangle. In this article, we mainly focus on the algorithmic sides of the new approach. The paper is divided into three parts: in Sec. 2, we recall some notations used in [11] and describe the general framework of plane-probing algorithms. We precisely describe and analyze our new algorithm in Sec. 3, whereas Sec. 4 is devoted to experimental results.

2 Plane-probing Algorithm Variants

A standard and rational *digital plane* is an infinite digital set defined by a normal $\mathbf{N} \in \mathbb{Z}^3 \setminus \{\mathbf{0}\}$ and a shift value $\mu \in \mathbb{Z}$ as follows [10]:

$$\mathbf{P}_{\mu,\mathbf{N}} := \{\mathbf{x} \in \mathbb{Z}^3 \mid \mu \leq \mathbf{x} \cdot \mathbf{N} < \mu + \|\mathbf{N}\|_1\}.$$

In this paper, we suppose w.l.o.g. that $\mu = 0$ and that the components of $\mathbf{N}$ are positive, i.e., $\mathbf{N} \in \mathbb{N}^3 \setminus \{\mathbf{0}\}$. Given a digital plane $\mathbf{P} \in \{\mathbf{P}_{0,\mathbf{N}} \mid \mathbf{N} \in \mathbb{N}^3 \setminus \{\mathbf{0}\}\}$ of unknown normal vector, a *plane-probing algorithm* computes the normal vector $\mathbf{N}$ of $\mathbf{P}$ by sparsely

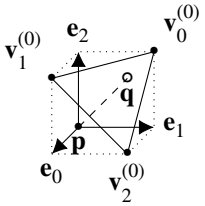


(a) (b)

Fig. 2: The evolution for normal $(1, 73, 100)$ with H-algorithm (a) and R-algorithm (b). Every triangle of the evolution is superimposed. The initial triangle is black. The next ones are more and more blue while iterating. The last one is red.

probing it with the predicate "is $\mathbf{x}$ in $\mathbf{P}$?" (InPlane($\mathbf{x}$) predicate hereafter). We describe below in a uniform way the algorithms H and R introduced in [9] as well as our new method (see also Algorithm 1).

Initialization. Let $(\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2)$ be the canonical basis of $\mathbb{Z}^3$. We assume that a starting point $\mathbf{p}$ satisfies three conditions: (1) $\mathbf{p} \in \mathbf{P}$, (2) the apex $\mathbf{q} := \mathbf{p} + (1, 1, 1) \notin \mathbf{P}$ and (3) the initial triangle $\mathbf{T}^{(0)} := (\mathbf{v}_k^{(0)})_{k \in \mathbb{Z}/3\mathbb{Z}} \subset \mathbf{P}$, where $\mathbf{v}_k^{(0)} := \mathbf{q} - \mathbf{e}_k$ for all $k \in \mathbb{Z}/3\mathbb{Z}$ (see inset figure and Algorithm 1, line 1).



Neighborhood and update rule. At every step $i \in \mathbb{N}$, the triangle $\mathbf{T}^{(i)}$ is defined from updated vertices $\{\mathbf{v}_k^{(i)}\}_{k \in \mathbb{Z}/3\mathbb{Z}}$ and represents the current approximation of the plane $\mathbf{P}$. All algorithms update one vertex of $\mathbf{T}^{(i)}$ per iteration. That vertex is replaced by a point of $\mathbf{P}$ from a candidate set, also called *neighborhood* in [9]. To properly define distinct neighborhoods, we first define the following sets:

$$S_H := \{(\alpha, \beta) \in \{(1, 0), (0, 1)\}. \tag{1}$$

$$S_R := \{(\alpha, \beta) \in \{(1, \lambda), (\lambda, 1) \mid \lambda \in \mathbb{N}\}. \tag{2}$$

$$S_L := \{(\alpha, \beta) \in \mathbb{N}^2 \setminus (0, 0)\}. \tag{3}$$

Note that $S_H \subset S_R \subset S_L$. At every step i and for any $S \in \{S_H, S_R, S_L\}$, the neighborhood is now defined as follows:

$$\mathcal{N}_S^{(i)} := \left\{ \mathbf{v}_k^{(i)} + \alpha(\mathbf{q} - \mathbf{v}_{k+1}^{(i)}) + \beta(\mathbf{q} - \mathbf{v}_{k+2}^{(i)}) \mid k \in \mathbb{Z}/3\mathbb{Z},\ (\alpha, \beta) \in S \right\}. \tag{4}$$

See Fig. 3 for an illustration of the neighborhoods. The H-algorithm is based on $\mathcal{N}_{S_H}^{(i)}$, which looks like an Hexagon, whereas the R-algorithm is based on $\mathcal{N}_{S_R}^{(i)}$, which consists of Rays. In this paper, we propose a lattice-based algorithm, denoted by the letter L, for lattice, and which uses the largest neighborhood $\mathcal{N}_{S_L}^{(i)}$.

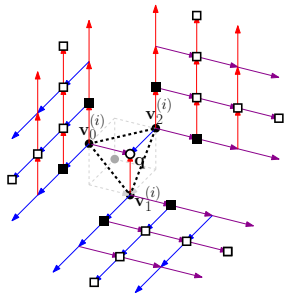


Fig. 3: Illustrations of the neighborhoods: $\mathcal{N}_{S_H}^{(i)}$ (black squares), $\mathcal{N}_{S_R}^{(i)}$ (white squares) and $\mathcal{N}_{S_L}^{(i)}$ includes every point on the lattices, excepted the triangle vertices.

Let $\mathcal{H}_+^{(i)}$ be the half-space delimited by $\mathbf{T}^{(i)}$ and containing $\mathcal{N}_S^{(i)}$. In addition, let $\mathcal{B}(\mathbf{T}, \mathbf{x})$ be the closed ball defined by $\mathbf{T}^{(i)}$ and a fourth point $\mathbf{x}$ not in the plane passing by $\mathbf{T}^{(i)}$. As in [11], for any pair of points $\mathbf{x}, \mathbf{x}'$, not in the plane passing by $\mathbf{T}^{(i)}$, we say that $\mathbf{x}'$ is *closer* to $\mathbf{T}^{(i)}$ than $\mathbf{x}$, denoted $\mathbf{x}' \preceq_{\mathbf{T}^{(i)}} \mathbf{x}$, if and only if $(\mathcal{B}(\mathbf{T}^{(i)}, \mathbf{x}') \cap \mathcal{H}_+^{(i)}) \subseteq (\mathcal{B}(\mathbf{T}^{(i)}, \mathbf{x}) \cap \mathcal{H}_+^{(i)})$. As $\preceq_{\mathbf{T}^{(i)}}$ is reflexive and transitive, and since all pairs of points in $\mathcal{H}_+^{(i)}$ are comparable [12], it defines a total preorder.

The algorithms replace a vertex of $\mathbf{T}^{(i)}$ with a point of the set $\mathcal{N}_S^{(i)} \bigcap \mathbf{P}$ that is a closest one according to $\preceq_{\mathbf{T}^{(i)}}$. More precisely, if $\mathcal{N}_S^{(i)} \bigcap \mathbf{P} \neq \emptyset$, there is at least an index $k \in \mathbb{Z}/3\mathbb{Z}$ and numbers $(\alpha, \beta) \in \mathbb{N}^2 \setminus (0,0)$ such that

$$\forall \mathbf{x} \in \mathcal{N}_S^{(i)} \cap \mathbf{P},\ \mathbf{v}_k^{(i)} + \alpha(\mathbf{q} - \mathbf{v}_{k+1}^{(i)}) + \beta(\mathbf{q} - \mathbf{v}_{k+2}^{(i)}) \preceq_{\mathbf{T}} \mathbf{x}. \tag{5}$$

Note that the triple (k, α, β) may not be unique when several points are in a cospherical position. The update rule is then [9, Lemma 2]:

$$\begin{cases} \mathbf{v}_k^{(i+1)} & := \mathbf{v}_k^{(i)} + \alpha(\mathbf{q} - \mathbf{v}_{k+1}^{(i)}) + \beta(\mathbf{q} - \mathbf{v}_{k+2}^{(i)}), \\ \mathbf{v}_{k+1}^{(i+1)} & := \mathbf{v}_{k+1}^{(i)}, \\ \mathbf{v}_{k+2}^{(i+1)} & := \mathbf{v}_{k+2}^{(i)}. \end{cases} \tag{6}$$

As shown in Algorithm 1, lines 5 to 7, equations (5) and (6) are used to update the current triangle.

Termination. The algorithms terminate at a step n, when the neighborhood has an empty intersection with the plane, i.e., when $\mathcal{N}_S^{(n)} \bigcap \mathbf{P} = \emptyset$ (Algorithm 1, line 3). The number of steps, n, is less than or equal to $\|\mathbf{N}\|_1 - 3$ [9, Theorem 1], which is a tight bound reached for any normal of components $(1, 1, r)$ with $r \in \mathbb{N} \setminus \{0\}$.

If $\mathbf{p}$ is one of the least high points in $\mathbf{P}$, i.e., $\mathbf{p} \cdot \mathbf{N} = 0$, the vertices of $\mathbf{T}^{(n)}$ are among the highest ones in $\mathbf{P}$, i.e., $\forall k \in \mathbb{Z}/3\mathbb{Z}, \mathbf{v}_k^{(n)} \cdot \mathbf{N} = \|\mathbf{N}\|_1 - 1$ [9, Theorem 2]. A consequence is that $\mathbf{T}^{(n)}$ is aligned with $\mathbf{P}$. In other words, its normal is equal to $\mathbf{N}$ [9, Corollary 4]. In addition, one can deduce from the vertices of $\mathbf{T}^{(n)}$ a basis of the lattice $\{\mathbf{x} \mid \mathbf{x} \cdot \mathbf{N} = \|\mathbf{N}\|_1 - 1\}$ [9, Corollary 5].

Even if [9] only introduces the neighborhoods H and R, the above-mentionned results and their proofs are correct for parameters $(\alpha, \beta) \in S_L$ in the update rule and, as a consequence, for the newly introduced neighborhood $\mathcal{N}_{S_L}^{(i)}$ as well.

Our motivation for introducing such a neighborhood is linked to the compactness. Indeed, starting from an identical triangle, a point chosen by the L-algorithm always lies in the circumscribed sphere that passes the point chosen by the H- or R-algorithm. Furthermore, it is proven that every circumscribed sphere of two consecutive triangles provided by the L-algorithm does not include any other points of the digital plane [12, Theorem 4]. In the next section, we show how to efficiently find a closest point in the L-neighborhood. The above-mentionned result will be crucial in Lemma 7.

Algorithm 1: Plane-probing algorithms H, R ([9]) and L (our method)

Input: The predicate InPlane $:=$ "Is a point $\mathbf{x} \in \mathbf{P}$?", a point $\mathbf{p} \in \mathbf{P}$ and the type of neighborhood $S \in \{S_H, S_R, S_L\}$ (see equations (1)-(4))
Output: A normal vector $\hat{\mathbf{N}}$ and a basis of the lattice $\{\mathbf{x} \mid \mathbf{x} \cdot \hat{\mathbf{N}} = \|\mathbf{N}\|_1 - 1\}$.

1 $\mathbf{q} \leftarrow \mathbf{p} + (1,1,1)$; $(\mathbf{v}_k^{(0)})_{k \in \mathbb{Z}/3\mathbb{Z}} \leftarrow (\mathbf{q} - \mathbf{e}_k)_{k \in \mathbb{Z}/3\mathbb{Z}}$; `// initialization`
2 $i \leftarrow 0$;
3 **while** $\mathcal{N}_S^{(i)} \cap \{\mathbf{x} \mid \text{InPlane}(\mathbf{x})\} \neq \emptyset$ **do**
4 Let (k, α, β) be such that, for all $\mathbf{y} \in \mathcal{N}_S^{(i)} \cap \{\mathbf{x} \mid \text{InPlane}(\mathbf{x})\}$,
5 $\mathbf{v}_k^{(i)} + \alpha(\mathbf{q} - \mathbf{v}_{k+1}^{(i)}) + \beta(\mathbf{q} - \mathbf{v}_{k+2}^{(i)}) \leq_{\mathbf{T}^{(i)}} \mathbf{y}$; `// equation (5)`
6 $\mathbf{v}_k^{(i+1)} \leftarrow \mathbf{v}_k^{(i)} + \alpha(\mathbf{q} - \mathbf{v}_{k+1}^{(i)}) + \beta(\mathbf{q} - \mathbf{v}_{k+2}^{(i)})$; `// equation (6)`
7 $\forall l \in \mathbb{Z}/3\mathbb{Z} \setminus k,\ \mathbf{v}_l^{(i+1)} \leftarrow \mathbf{v}_l^{(i)}$;
8 $i \leftarrow i + 1$;
9 $B \leftarrow \{\mathbf{v}_0^{(i)} - \mathbf{v}_1^{(i)}, \mathbf{v}_1^{(i)} - \mathbf{v}_2^{(i)}, \mathbf{v}_2^{(i)} - \mathbf{v}_0^{(i)}\}$;
10 Let $\mathbf{b}_1$ and $\mathbf{b}_2$ be the shortest and second shortest vectors of B ;
11 **return** $\mathbf{b}_1 \times \mathbf{b}_2, (\mathbf{b}_1, \mathbf{b}_2)$; `// × denotes the cross product`

3 The L-algorithm

The most expensive task in Algorithm 1 is computing a point of $\mathcal{N}_S^{(i)} \cap \mathbf{P}$, which is *closest* according to $\leq_{\mathbf{T}^{(i)}}$ (see lines 4 and 5). A brute-force method would be computing the whole finite set $\mathcal{N}_S^{(i)} \cap \mathbf{P}$ and finding a point of that set closer than any others, which would require lots of probes. In practice, one does not need to probe so much, because one can safely discard a large part of $\mathcal{N}_S^{(i)} \cap \mathbf{P}$.

In this section, we focus on a step $i \in \{0, \dots, n\}$ and for the sake of simplicity, we drop the exponent (i) in the notations. Furthermore, we focus on the 2D lattice

$$\forall k \in \mathbb{Z}/3\mathbb{Z},\ \mathcal{L}_k := \left\{\mathbf{v}_k + \alpha\mathbf{m}_{k+1} + \beta\mathbf{m}_{k+2} \mid (\alpha, \beta) \in S_L\right\},$$

where $\mathbf{m}_k := \mathbf{q} - \mathbf{v}_k$ for all $k \in \mathbb{Z}/3\mathbb{Z}$. We propose an algorithm (Algorithm 2) that selects a small and sufficient set of candidate points included in $\mathcal{L}_k$.

3.1 A Smaller Candidate Set

We introduce, in the first place, two general geometrical results which will be useful.

Lemma 1. *Let two non-zero vectors* $\mathbf{u}, \mathbf{w} \in \mathbb{R}^3$ *and a closed ball whose border passes through the origin* o *and the point* $o + \mathbf{u} + \mathbf{w}$. *If* $\mathbf{u} \cdot \mathbf{w} \geq 0$, *at least one of the two points* $o + \mathbf{u}$ *and* $o + \mathbf{w}$ *lies in the ball.*

Proof. We focus on the plane including $o, o + \mathbf{u}$, $o + \mathbf{w}$ (and $o + \mathbf{u} + \mathbf{w}$). In this plane, if $\mathbf{u} \cdot \mathbf{w} \geq 0$, one half of the disk of diameter $[o, o + \mathbf{u} + \mathbf{w}]$ contains $o + \mathbf{u}$, whereas the other contains $o + \mathbf{w}$. Furthermore, any other disk whose border passes through o and $o + \mathbf{u} + \mathbf{w}$ must include one of the previous halves, thus one of the two points. Since any ball whose border passes through o and $o + \mathbf{u} + \mathbf{w}$ covers such a disk, the result follows (see Fig. 4-(a)). □

Lemma 2. *Let a non-zero vector* $\mathbf{u} \in \mathbb{R}^3$ *and a closed ball whose border passes through the origin* o *and the point* $o + \mathbf{u}$. *No point* $o + \delta\mathbf{u}$ *such that* $\delta > 1$ *lies in the ball.*

Proof. The intersection between the ball and the ray starting from o in direction $\mathbf{u}$ is the segment $[o, o + \mathbf{u}]$, which is equal, by convexity, to the set $\{o + \delta'\mathbf{u}\}_{0 \leq \delta' \leq 1}$. The points $o + \delta\mathbf{u}$ such that $\delta > 1$ do not lie in that set and therefore do not lie in the ball. □

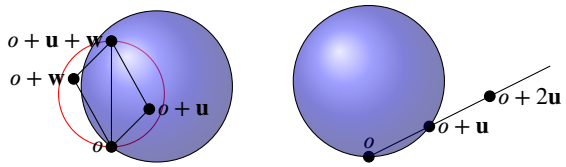


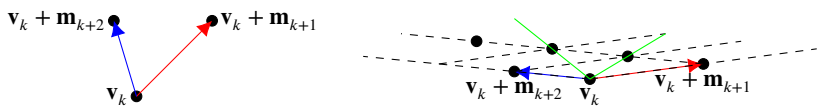
Fig. 4: Illustrations for (a) lemma 3 and (b) lemma 2.

An elementary application of the above lemmas is the following result:

Lemma 3. *For all* $k \in \mathbb{Z}/3\mathbb{Z}$, *let* Λ_k *be the set* $\left\{\mathbf{v}_k + \alpha\mathbf{u} + \beta\mathbf{w} \mid (\alpha, \beta) \in S_L\right\}$, *where* $\mathbf{u}$, $\mathbf{w}$ *are any two non-zero vectors of* $\mathbb{Z}^3$ *such that* $\mathbf{v}_k + \mathbf{u}, \mathbf{v}_k + \mathbf{w} \in \mathcal{H}_+$. *If* $\mathbf{u} \cdot \mathbf{w} \geq 0$, *we have either* $\mathbf{v}_k + \mathbf{u} \leq_{\mathbf{T}} \mathbf{x}$ *for all* $\mathbf{x} \in \Lambda_k$ *or* $\mathbf{v}_k + \mathbf{w} \leq_{\mathbf{T}} \mathbf{x}$ *for all* $\mathbf{x} \in \Lambda_k$.

Proof. Let us consider the ball $\mathcal{B}(\mathbf{T}, \mathbf{x})$ for a point $\mathbf{x} := \mathbf{v}_k + \alpha\mathbf{u} + \beta\mathbf{w}$, with $\alpha, \beta \geq 1$. Since $\alpha\mathbf{u} \cdot \beta\mathbf{w} \geq 0$, by Lemma 1, we know that either $\mathbf{v}_k + \alpha\mathbf{u}$ or $\mathbf{v}_k + \beta\mathbf{w}$ lies in $\mathcal{B}(\mathbf{T}, \mathbf{x})$. Let us assume w.l.o.g. that $\mathbf{v}_k + \alpha\mathbf{u} \in \mathcal{B}(\mathbf{T}, \mathbf{x})$, which means that $\mathbf{v}_k + \alpha\mathbf{u} \leq_{\mathbf{T}} \mathbf{x}$. By Lemma 2, we then conclude that $\mathbf{v}_k + \mathbf{u} \leq_{\mathbf{T}} \mathbf{v}_k + \alpha\mathbf{u} \leq_{\mathbf{T}} \mathbf{x}$. □

Thanks to the previous lemma, if $\mathbf{m}_{k+1} \cdot \mathbf{m}_{k+2} \geq 0$, one can consider only two points of $\mathcal{L}_k$ (Fig. 5 (a)). Otherwise, if both $\mathbf{m}_{k+1} \cdot (\mathbf{m}_{k+1} + \mathbf{m}_{k+2}) \geq 0$ and $(\mathbf{m}_{k+1} + \mathbf{m}_{k+2}) \cdot \mathbf{m}_{k+2} \geq 0$, one can again consider as few as three points: $\mathbf{v}_k + \mathbf{m}_{k+1}$, $\mathbf{v}_k + \mathbf{m}_{k+2}$ and $\mathbf{v}_k + \mathbf{m}_{k+1} + \mathbf{m}_{k+2}$. We now focus on the case where either $\mathbf{m}_{k+1} \cdot (\mathbf{m}_{k+1} + \mathbf{m}_{k+2})$ or $\mathbf{m}_{k+2} \cdot (\mathbf{m}_{k+1} + \mathbf{m}_{k+2})$ is strictly negative (see for instance Fig. 5 (b)).



(a) (b)

Fig. 5: Angle between $\mathbf{m}_{k+1}$ and $\mathbf{m}_{k+2}$: (a) when $\mathbf{m}_{k+1} \cdot \mathbf{m}_{k+2} \geq 0$, (b) when $\mathbf{m}_{k+1} \cdot \mathbf{m}_{k+2} < 0$ and $\mathbf{m}_{k+2} \cdot (\mathbf{m}_{k+1} + \mathbf{m}_{k+2}) < 0$. Here, we also have $(\mathbf{m}_{k+1} + \gamma \mathbf{m}_{k+2}) \cdot (\mathbf{m}_{k+1} + (\gamma + 1)\mathbf{m}_{k+2}) < 0$ with $\gamma = 1$ (see Lemma 4 and the green angle).

Lemma 4. *Let* $\mathbf{u}$, $\mathbf{w}$ *be two non-zero vectors in* $\mathbb{Z}^3$. *If there exists* $\gamma \geq 1$ *such that*

$$(\mathbf{u} + \gamma \mathbf{w}) \cdot (\mathbf{u} + (\gamma + 1)\mathbf{w}) < 0, \tag{7}$$

then γ *is the unique integer greater than or equal to* 1 *that verifies*

$$(\mathbf{u} + (\gamma + 1)\mathbf{w}) \cdot \mathbf{w} > 0 > (\mathbf{u} + \gamma \mathbf{w}) \cdot \mathbf{w}. \tag{8}$$

In this case, $\gamma = \left\lfloor \frac{-\mathbf{u} \cdot \mathbf{w}}{\|\mathbf{w}\|^2} \right\rfloor$.

Proof. We refer to Fig. 5(b) for an example where $\mathbf{u} = \mathbf{m}_{k+1}$ and $\mathbf{w} = \mathbf{m}_{k+2}$. By rewritting the left-hand side of (7) as $(\mathbf{u} + \gamma \mathbf{w}) \cdot \big((\mathbf{u} + \gamma \mathbf{w}) + \mathbf{w}\big)$ and developing, we get

$$\|\mathbf{u} + \gamma \mathbf{w}\|^2 + (\mathbf{u} + \gamma \mathbf{w}) \cdot \mathbf{w} < 0 \Rightarrow (\mathbf{u} + \gamma \mathbf{w}) \cdot \mathbf{w} < 0\,,$$

which is the right-hand side of (8). Similarly, by rewritting the left-hand side of (7) as $\big((\mathbf{u} + (\gamma + 1)\mathbf{w}) - \mathbf{w}\big) \cdot (\mathbf{u} + (\gamma + 1)\mathbf{w}) < 0$ and developing, we have

$$\|(\mathbf{u} + (\gamma + 1)\mathbf{w})\|^2 - \mathbf{w} \cdot (\mathbf{u} + (\gamma + 1)\mathbf{w}).$$

As this expression is strictly negative by (7), we obtain $(\mathbf{u} + (\gamma + 1)\mathbf{w}) \cdot \mathbf{w} > 0$, which is the left-hand side of (8). To end, by developing (8) and isolating the γ, we obtain $\gamma + 1 > \frac{-\mathbf{u} \cdot \mathbf{w}}{\|\mathbf{w}\|^2} > \gamma$, thus unicity. □

Lemma 5. *Let* $\mathbf{u}$, $\mathbf{w}$ *be two non-zero vectors in* $\mathbb{Z}^3$. *If there exists* $\gamma \geq 1$ *verifying (7), then for all* $c \in \{0, 1, \ldots, \gamma - 1\}$, $(\mathbf{u} + c\mathbf{w}) \cdot (\mathbf{u} + (c + 1)\mathbf{w}) > 0$.

Proof. First, observe that for all $c \in \mathbb{N} \setminus \{0\}$,

$$(\mathbf{u} + (c - 1)\mathbf{w}) \cdot (\mathbf{u} + c\mathbf{w}) = (\mathbf{u} + c\mathbf{w}) \cdot (\mathbf{u} + (c + 1)\mathbf{w}) - 2\mathbf{w} \cdot (\mathbf{u} + c\mathbf{w}). \tag{9}$$

To determine the sign of $-2\mathbf{w} \cdot (\mathbf{u} + c\mathbf{w})$, note that we obviously have $c\mathbf{w}^2 < \gamma \mathbf{w}^2$ and, from the right-hand side of (8), $\gamma \mathbf{w}^2 < -\mathbf{u} \cdot \mathbf{w}$. As a result,

$$c\mathbf{w}^2 < -\mathbf{u} \cdot \mathbf{w} \Leftrightarrow \mathbf{w} \cdot (\mathbf{u} + c\mathbf{w}) < 0.$$

Since $-2\mathbf{w} \cdot (\mathbf{u} + c\mathbf{w}) > 0$, it is enough to show that the statement is true for $c = \gamma - 1$ because the result for the smaller values of c then follows by induction.

By (8), we also have $\mathbf{w} \cdot (\mathbf{u} + \gamma\mathbf{w}) < 0 < (\mathbf{u} + \gamma\mathbf{w})^2$. Therefore,

$$2\mathbf{w} \cdot (\mathbf{u} + \gamma\mathbf{w}) < (\mathbf{u} + \gamma\mathbf{w})^2 + \mathbf{w} \cdot (\mathbf{u} + \gamma\mathbf{w}) = (\mathbf{u} + \gamma\mathbf{w}) \cdot (\mathbf{u} + (\gamma + 1)\mathbf{w}).$$

From this lower bound and replacing c by γ in (9), we finally obtain $(\mathbf{u}+(\gamma-1)\mathbf{w}) \cdot (\mathbf{u}+\gamma\mathbf{w}) > 0$, which concludes the proof. □

The two previous lemmas provide a set of lattice bases whose vectors form an acute angle. Indeed, with $\mathbf{u} = \mathbf{m}_{k+1}$ and $\mathbf{w} = \mathbf{m}_{k+2}$ and assuming that γ exists, we have $(\mathbf{m}_{k+1} + (\gamma + 1)\mathbf{m}_{k+2}) \cdot \mathbf{m}_{k+2} > 0$ (Lemma 4) and for all $c \in \{0, 1, \dots, \gamma - 1\}$, $(\mathbf{m}_{k+1} + c\mathbf{m}_{k+2}) \cdot (\mathbf{m}_{k+1} + (c + 1)\mathbf{m}_{k+2}) > 0$ (Lemma 5). Then, it straightfowardly follows from Lemma 3 that the closest points in the set

$$\{\mathbf{v}_k + \mathbf{m}_{k+2}\} \cup \left\{\mathbf{v}_k + \mathbf{m}_{k+1} + c\mathbf{m}_{k+2} \mid c \in \{0, \dots, \gamma + 1\}\right\}$$

are closer than any other points in the set

$$\mathcal{L}_k \setminus \{\mathbf{v}_k + \alpha\big(\mathbf{m}_{k+1} + \gamma\mathbf{m}_{k+2}\big) + \beta\big(\mathbf{m}_{k+1} + (\gamma + 1)\mathbf{m}_{k+2}\big) \mid \alpha, \beta \geq 1\}.$$

One part of $\mathcal{L}_k$ cannot be covered because $(\mathbf{m}_{k+1}+\gamma\mathbf{m}_{k+2})\cdot(\mathbf{m}_{k+1}+(\gamma+1)\mathbf{m}_{k+2}) < 0$. In order to cope with that problem, we simply recursively apply the previous results.

Definition 1. *For any pair of linearly independent non-zero vectors* $(\mathbf{u}, \mathbf{w}) \in \mathbb{Z}^3 \times \mathbb{Z}^3$, *we define a sequence of vector pairs* $\Omega_{\mathbf{u},\mathbf{w}} = \left\{(\mathbf{u}_j, \mathbf{w}_j)\right\}_{j\geq 0}$ *as follows:*

1. $\mathbf{u}_0 = \mathbf{u}$ *and* $\mathbf{w}_0 = \mathbf{w}$.
2. *For any* $j \geq 0$, *the pair* $(\mathbf{u}_{j+1}, \mathbf{w}_{j+1})$ *exists if and only if there exists* $\gamma_j \geq 1$ *such that*

$$(\mathbf{u}_j + \gamma_j\mathbf{w}_j) \cdot (\mathbf{u}_j + (\gamma_j + 1)\mathbf{w}_j) < 0, \tag{10}$$

then

$$\mathbf{u}_{j+1} = \mathbf{w}_j\,, \quad \mathbf{w}_{j+1} = \mathbf{u}_j + \gamma_j\mathbf{w}_j. \tag{11}$$

Definition 2 (Candidate set). *For* $k \in \mathbb{Z}/3\mathbb{Z}$ *and for any pair of vectors* $(\mathbf{u}, \mathbf{w})$ *in the set* $\{(\mathbf{m}_{k+1}, \mathbf{m}_{k+2}), (\mathbf{m}_{k+2}, \mathbf{m}_{k+1})\}$, *we define*

$$C_k := \bigcup_{(\mathbf{u}_j,\mathbf{w}_j)\in\Omega(\mathbf{u},\mathbf{w})} \{\mathbf{v}_k + \mathbf{w}_j\} \cup \left\{\mathbf{v}_k + \mathbf{u}_j + c\mathbf{w}_j \mid c \in \{0, \dots, \gamma_j + 1\}\right\}.$$

The finiteness of C_k stems from the finiteness of $\Omega_{\mathbf{u},\mathbf{w}}$:

Lemma 6. *The sequence* $\Omega_{\mathbf{u},\mathbf{w}} = \left\{(\mathbf{u}_j, \mathbf{w}_j)\right\}_{j\geq 0}$ *is finite.*

Proof. From (11), we have for any $j \geq 0$, $-\mathbf{u}_{j+1} \cdot \mathbf{w}_{j+1} = -\mathbf{w}_j \cdot (\mathbf{u}_j + \gamma_j\mathbf{w}_j)$. Developing the last expression, we obtain $-\mathbf{u}_j \cdot \mathbf{w}_j - \gamma_j\|\mathbf{w}_j\|^2$, which is strictly less than $-\mathbf{u}_j \cdot \mathbf{w}_j$. Therefore, the sequence of natural numbers $\left\{-\mathbf{u}_j \cdot \mathbf{w}_j\right\}_{j\geq 0}$ is strictly decreasing. Since, in addition, $-\mathbf{u}_j \cdot \mathbf{w}_j \geq \|\mathbf{w}_j\|^2$, while there exists $\gamma_j \geq 1$, the sequence $\Omega_{\mathbf{u},\mathbf{w}}$ is finite. □

3.2 Even smaller candidate set

The set C_k described in the previous section is a union of subsets of aligned points. We show below that, for each subset, the last point is always closer than the other ones:

Lemma 7. *For any vectors* $\mathbf{u}, \mathbf{w} \in \mathcal{L}_k$ *such that there exists* $\gamma \geq 1$ *such that* $(\mathbf{u}+\gamma\mathbf{w}) \cdot (\mathbf{u}+(\gamma+1)\mathbf{w}) < 0$*, then for any* $0 \leq c \leq \gamma - 1$*, we have* $\mathbf{u}+\gamma\mathbf{w} \leq_{\mathbf{T}} \mathbf{u}+c\mathbf{w}$.

Proof. We assume w.l.o.g. that $k = 0$ and we use the notation $\delta^0_{\mathbf{T}}(\mathbf{x}, \mathbf{y})$ introduced in [11], where $\mathbf{x}$ and $\mathbf{y}$ are relative points of $\mathbb{Z}^3$ when considering $\mathbf{v}_0$ as origin. We recall that if $\mathbf{v}_0 + \mathbf{x} \in \mathcal{H}_+$, then $\mathbf{v}_0 + \mathbf{x} \leq_{\mathbf{T}} \mathbf{v}_0 + \mathbf{y} \Leftrightarrow \delta^0_{\mathbf{T}}(\mathbf{x}, \mathbf{y}) \geq 0$.

In order to show that for all $0 \leq c \leq \gamma - 1$, $\delta^0_{\mathbf{T}}(\mathbf{u}+\gamma\mathbf{w}, \mathbf{u}+c\mathbf{w}) \geq 0$, we use the following identity [11, equation (6)]:

$$\delta^0_{\mathbf{T}}(\mathbf{z}, \mathbf{z}' + \mathbf{z}'') = \delta^0_{\mathbf{T}}(\mathbf{z}, \mathbf{z}') + \delta^0_{\mathbf{T}}(\mathbf{z}, \mathbf{z}'') + \left(2\mathbf{z}' \cdot \mathbf{z}''\right) \det\left[\mathbf{m}_0 - \mathbf{m}_1, \mathbf{m}_0 - \mathbf{m}_2, \mathbf{z}\right]. \quad (12)$$

Indeed, as $c = \gamma - (\gamma - c)$, we obtain (with $\mathbf{z} = \mathbf{z}' = \mathbf{u}+\gamma\mathbf{w}$ and $\mathbf{z}'' = -(\gamma - c)\mathbf{w}$):

$$\begin{aligned}\delta^0_{\mathbf{T}}(\mathbf{u}+\gamma\mathbf{w}, \mathbf{u}+c\mathbf{w}) = &\underbrace{\delta^0_{\mathbf{T}}(\mathbf{u}+\gamma\mathbf{w}, \mathbf{u}+\gamma\mathbf{w})}_{=0} + \underbrace{\delta^0_{\mathbf{T}}(\mathbf{u}+\gamma\mathbf{w}, -(\gamma-c)\mathbf{w})}_{\geq 0, \text{ see item 1}} \\ &- 2(\gamma - c) \underbrace{(\mathbf{u}+\gamma\mathbf{w}) \cdot \mathbf{w}}_{\leq 0 \text{ by Lemma 4 (8)}} \underbrace{\det\left[\mathbf{m}_0 - \mathbf{m}_1, \mathbf{m}_0 - \mathbf{m}_2, \mathbf{u}+\gamma\mathbf{w}\right]}_{>0, \text{ see item 2}}.\end{aligned}$$

1. Let $\mathcal{H}_-$ be the half-space lying below the plane incident to $\mathbf{T}$. Let us set $\mathbf{x} := \mathbf{u}+\gamma\mathbf{w}$ and $\mathbf{y} := -(\gamma - c)\mathbf{w}$. By definition, $\mathbf{v}_0 + \mathbf{x} \in \mathcal{H}_+$ and $\mathbf{v}_0 + \mathbf{y} \in \mathcal{H}_-$. We have to prove that $\mathbf{v}_0 + \mathbf{x} \leq_{\mathbf{T}} \mathbf{v}_0 + \mathbf{y}$. Let $\mathbf{x}^\star$ be the closest point chosen for update. By definition, $\mathbf{x}^\star \leq_{\mathbf{T}} \mathbf{v}_0 + \mathbf{x}$, which implies that $(\mathcal{H}_- \cap \mathcal{B}(\mathbf{T}, \mathbf{v}_0 + \mathbf{x})) \subseteq (\mathcal{H}_- \cap \mathcal{B}(\mathbf{T}, \mathbf{x}^\star))$ (see Lemma 8 in appendix). Due to the above inclusion relation, since $\mathbf{v}_0 + \mathbf{y}$ is not in the interior of $\mathcal{B}(\mathbf{T}, \mathbf{x}^\star)$ [12, Theorem 4], $\mathbf{v}_0 + \mathbf{y}$ is not in the interior of $\mathcal{B}(\mathbf{T}, \mathbf{v}_0 + \mathbf{x})$ either, i.e., $\mathbf{v}_0 + \mathbf{x} \leq_{\mathbf{T}} \mathbf{v}_0 + \mathbf{y}$.
2. For any $(\alpha, \beta) \in S_L$, $\det\left[\mathbf{m}_0 - \mathbf{m}_1, \mathbf{m}_0 - \mathbf{m}_2, \alpha\mathbf{m}_1 + \beta\mathbf{m}_2\right] = \alpha + \beta > 0$, because $\det\left[\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2\right] = 1$ [9, Lemma 3]. Notably, $\det\left[\mathbf{m}_0 - \mathbf{m}_1, \mathbf{m}_0 - \mathbf{m}_2, \mathbf{u}+\gamma\mathbf{w}\right] > 0$. □

Lemma 7 shows that the last point should be the closest. However, in the case where this last point is not in $\mathbf{P}$, we can resort to a binary search as in [9, Algorithm 4].

3.3 Algorithm and Complexity

Fig. 6 sums up the process of filtering the set $\mathcal{L}_k$. However, we have to discard the points that are not in $\mathbf{P}$. For this purpose, we use the predicate InPlane in the whole procedure detailed in Algorithm 2. We set $\omega := \|\mathbf{N}\|_1$. The worst-case number of predicate calls is in $O(\omega)$ for the H-algorithm, $O(\omega \log \omega)$ for the R-algorithm [9] and $O(\omega)$ for the R^1-algorithm [11]. We give below a upper bound for the L-algorithm.

Theorem 1. *Algorithm 2 requires* $O(\log \omega)$ *calls to the predicate* InPlane*: "is* $\mathbf{x}$ *in* $\mathbf{P}$*?".*

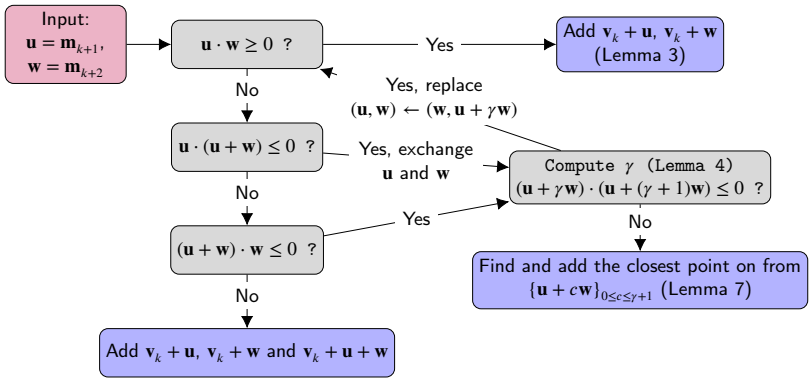


Fig. 6: Roadmap

Proof. We consider the sequence of vectors $(\mathbf{u}_j, \mathbf{w}_j)_{0 \leq j \leq j_{max}}$. For any $j \geq 2$, if we rewrite the equation (11) with only $\mathbf{u}_{j-2}$, $\mathbf{u}_{j-1}$ and $\mathbf{u}_j$, we obtain the relation $\mathbf{u}_j = \mathbf{u}_{j-2} + \gamma_j \mathbf{u}_{j-1}$. We use the bar notation $\bar{\cdot}$ above any vector $\mathbf{x}$ to denote its height relative to $\mathbf{N}$. Otherwise said, $\bar{\mathbf{x}} := \mathbf{x} \cdot \mathbf{N}$. Then, we have $\bar{\mathbf{u}}_j = \bar{\mathbf{u}}_{j-2} + \gamma_j \bar{\mathbf{u}}_{j-1} \geq \bar{\mathbf{u}}_{j-2} + \bar{\mathbf{u}}_{j-1}$ (because $\gamma_j \geq 1$ and $\bar{\mathbf{u}}_{j-1} \geq 0$ by recurrence). By induction, we have for all $2 \leq j \leq j_{max}$, $\bar{\mathbf{u}}_j \geq 2^{\lfloor \frac{j}{2} \rfloor}(\bar{\mathbf{u}}_0 + \bar{\mathbf{u}}_1)$, which leads to $j_{max} \in O(\log \omega)$, because the last point must be in $\mathbf{P}$, i.e., $\bar{\mathbf{u}}_{j_{max}} \leq \omega$. Note that there is only one call to the predicate at each rank $2 \leq j \leq j_{max}$ (and at most four calls before), hence a total of $O(\log \omega)$ calls at the last rank. It remains to notice that the final search also requires at most $O(\log \omega)$ calls with an appropriate procedure such as [9, Algorithm 4]. □

A straightforward corollary is that the total number of predicate calls is in $O(\omega \log \omega)$ for the L-algorithm, because there are $O(\omega)$ steps (see Sec. 2) and $O(\log \omega)$ calls to the predicate at every step due to the use of Algorithm 2 (Theorem 1).

4 Experimental results

Overall performance. First of all, Fig. 7 compares the number of predicate calls for different plane-probing algorithms in a simple family of digital planes. The figure also shows the result of an optimized variant of the L-algorithm, denoted L-opt, that decreases the number of calls at each step by some values that are bounded by a constant. The points of the H-neighborhood are included in the L-neighborhood and are necessarily probed by our method. However, some of the points remain inside the H-neighborhood after the vertex update. Therefore, instead of probing all of them repeatedly at each iteration, one can use a cache so as not to probe twice the same point.

To provide more statistics, we have considered a large collection of implicit digital planes with normal vectors in a set χ with relatively prime components, in the range $(1, 1, 1)$ to $(200, 200, 200)$ ($|\chi| = 6578833$). For all variants of plane-probing algorithms, including our L-algorithm, we compare in Table 1 (left): the number n of steps, the number $\mathcal{N}^i_{call}$ of calls to the predicate per iteration and the total number of calls

Algorithm 2: CREATECANDIDATELIST(InPlane, $\mathbf{T}$, $\mathbf{q}$, k)

Input: The predicate InPlane, the triangle $\mathbf{T}$, the point $\mathbf{q}$ and an index $k \in \{0, 1, 2\}$
Output: A list $Cand_k$ of candidate points around vertex $\mathbf{v}_k$

Initialize $Cand_k$; $(\mathbf{m}_1, \mathbf{m}_2) \leftarrow (\mathbf{q} - \mathbf{v}_{k+1}, \mathbf{q} - \mathbf{v}_{k+2})$;
Add $\mathbf{v}_k + \mathbf{m}_1$ (resp. $\mathbf{v}_k + \mathbf{m}_2$) to $Cand_k$ if InPlane($\mathbf{v}_k + \mathbf{m}_1$) (resp. InPlane($\mathbf{v}_k + \mathbf{m}_2$));
if InPlane($\mathbf{v}_k + \mathbf{m}_1$) *and* InPlane($\mathbf{v}_k + \mathbf{m}_2$) **then**
    $(\mathbf{u}, \mathbf{w}) \leftarrow (\mathbf{m}_1, \mathbf{m}_2)$;
    while $\mathbf{u} \cdot \mathbf{w} < 0$ **do**
        if $\mathbf{u} \cdot (\mathbf{u} + \mathbf{w}) \leq 0$ *or* $\mathbf{w} \cdot (\mathbf{u} + \mathbf{w}) \leq 0$ **then**
            if $\mathbf{u} \cdot (\mathbf{u} + \mathbf{w}) \leq 0$ **then**
                $(\mathbf{u}, \mathbf{w}) \leftarrow (\mathbf{w}, \mathbf{u})$;
            Compute $\gamma = \left\lfloor \frac{-\mathbf{u}\cdot\mathbf{w}}{\|\mathbf{w}\|^2} \right\rfloor$;
            if $(\mathbf{u} + \gamma\mathbf{w}) \cdot (\mathbf{u} + (\gamma + 1)\mathbf{w}) < 0$ **then**
                if InPlane($\mathbf{u} + \gamma\mathbf{w}$) **then**
                    Add $\mathbf{v}_k + \mathbf{u} + \gamma\mathbf{w}$ to $Cand_k$;
                    $(\mathbf{u}, \mathbf{w}) \leftarrow (\mathbf{w}, \mathbf{u} + \gamma\mathbf{w})$;
                else
                    Find a closest point $\mathbf{x}^\star \in \{\mathbf{v}_k + \mathbf{u} + c\mathbf{w}\}_{0 \leq c \leq \gamma+1}$ such that InPlane($\mathbf{x}^\star$) and add it to $Cand_k$; **break**;
            else
                Find a closest point $\mathbf{x}^\star \in \{\mathbf{v}_k + \mathbf{u} + c\mathbf{w}\}_{0 \leq c \leq \gamma-1}$ such that InPlane($\mathbf{x}^\star$) and add it to $Cand_k$; **break**;
        else
            Add $\mathbf{v}_k + \mathbf{u} + \mathbf{w}$ to $Cand_k$ if InPlane($\mathbf{v}_k + \mathbf{u} + \mathbf{w}$); **break**;
return $Cand_k$;

$\sum_{i=0}^{n-1} \mathcal{N}^i_{call}$. The results are obtained from a C++ implementation using the DGtal Library [13]. The numbers do not perfectly match with the table shown in [11] due to different implementation choices (*e.g.*, the ordering in case of co-spherical points).

In average, the L-algorithm requires a fewer number of steps to obtain the exact normal vector of the plane. We also remark that it usually examines fewer points at each step than the R-algorithm. However, it does not beat the R^1-algorithm, the optimized version of R-algorithm, in terms of the number of calls to the predicate.

Locality. We wish to estimate the proximity of the probes to the initial vertex during the iterations. We define the *max distance* $Dist_{max}$ of the last triangle $\mathbf{T}^{(n)}$ as $\max_k \{\|\mathbf{m}_k^{(n)}\|\}$. Since the last triangle obtained by the L-algorithm has only acute or right angles [12, Corollary 1], one can derive the following upper bound (see the last section of [12]):

$$\|Dist_{max}\| \leq \sqrt{\frac{2}{3}\|\mathbf{N}\|_2^2 + \frac{2}{\sqrt{3}}\|\mathbf{N}\|_2 + \frac{1}{\|\mathbf{N}\|_2^2}}\,. \tag{13}$$

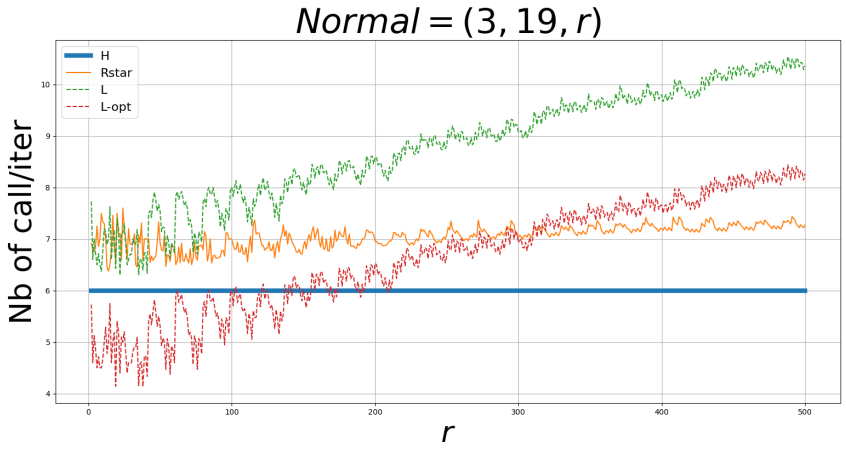


Fig. 7: Number of calls to predicate per iteration for $\mathbf{N} \in \{(3, 19, r), 1 \leq r \leq 500\}$.

	n	$\mathcal{N}^i_{call}$		$\sum_{i=0}^{n-1} \mathcal{N}^i_{call}$
alg.	avg.	avg.	max.	avg.
H	25.3756	6.00	6	152.25
R	19.2534	17.73	25	271.31
R[1]	19.2534	9.77	15	131.23
L	19.2529	12.03	21	144.85

	$\lvert\mathbf{N}$ s.t. $C_\mathbf{N} > 0\rvert$	$C_\mathbf{N}$	
alg.	tot.	tot.	avg.
H	247457	75235972	471.46
R	90	424	2.44
R[1]	-	-	-
L	0	0	0

Table 1: Statistics of plane-probing algorithms (on planes whose normal is in χ on the left or lying between $(1, 1, 1)$ and $(80, 80, 80)$ on the right). $\mathcal{N}^i_{call}$ denotes the number of calls to predicate at a step i and n is the number of steps. $C_\mathbf{N}$ denotes the number of points lying both in $\mathbf{P}$ and in $\mathcal{B}_{\mathbf{T}^{(i)},\mathbf{T}^{(i+1)}}$, the closed ball that passes through the vertices of two consecutive triangles.

In Fig. 8, we measure the max distance of the last triangle computed by the L-algorithm for all normals whose l_2-norm is less than 200 and compare them with the above theoretical bound. Both the theoretical bound and the bound given by experiments shows that the max distance is linear with respect to $\|\mathbf{N}\|_2$ and the thickness of the digital plane.

Compactness. For all $i \in \{0, \cdots, n-1\}$, let $\mathcal{B}_{\mathbf{T}^{(i)},\mathbf{T}^{(i+1)}}$ be the closed ball that passes through the vertices of two consecutive triangles. We tested for all normals in the set χ, that for the L-algorithm the sequence of radii of $\left\{\mathcal{B}_{\mathbf{T}^{(i)},\mathbf{T}^{(i+1)}}\right\}_{0\leq i\leq n-1}$ is non-decreasing. This is not the case for H-algorithm nor R-algorithm. An example is shown in the Fig. 10. We also count the number of points in $\mathbf{P}$ and strictly inside the balls $\left\{\mathcal{B}_{\mathbf{T}^{(i)},\mathbf{T}^{(i+1)}}\right\}_{0\leq i\leq n-1}$ for all normals of coprime coordinates between $(1, 1, 1)$ and $(80, 80, 80)$ (See Table 1, right). No points are found in any balls for the L-algorithm while 75235972 points are found for H-algorithm and 424 points for R-algorithm.

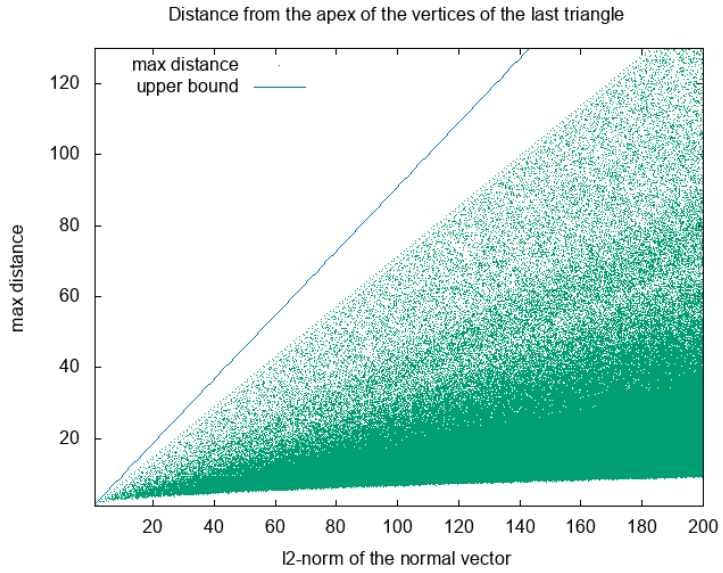
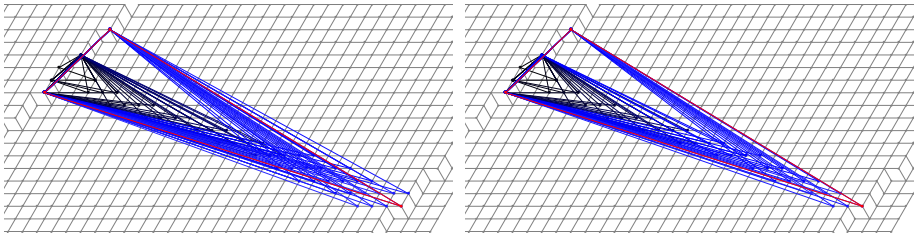


Fig. 8: The relation between maximum distance and the l_2-norm of normal vectors. Each green dot corresponds to the output of the L-algorithm for a given normal vector in χ. The theoretical upper bound is in blue.

In particular, we observe for all vectors in χ that the steps of L-algorithm are included in the ones of R-algorithm. This implies that L-algorithm always needs fewer steps than the R-algorithm. For example, for the plane of normal vector $\mathbf{N} = (2, 5, 6)$ (see Fig. 9), the L-algorithm uses 40 steps while the R-algorithm uses 50 steps to find the exact normal vector. In Fig. 10, we also observe that the curves of L-algorithm stop earlier than other plane-probing algorithms.

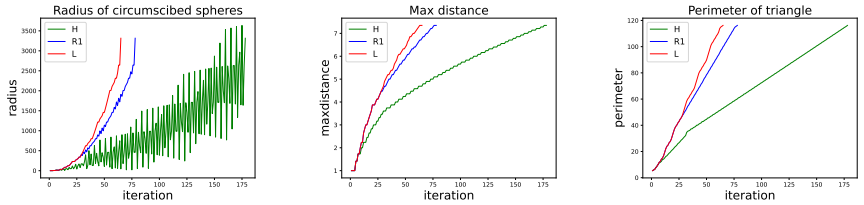
5 Conclusion

In this paper, we present a new plane-probing algorithm, called L-algorithm, that takes into account more candidate points at each step than its predecessors. We also observe that the L-algorithm requires fewer steps than the R-algorithm and H-algorithm. In contrary, at each step, it needs to examine more candidate points to find the closest point in its neighborhood. Despite this downside, the point selected by the L-algorithm provides more interesting compactness features at every step. The circumspheres of consecutive triangles has non-decreasing radii and do not include any point in the plane. In other words, the L-algorithm creates a local 3D delaunay triangulation of the digital plane. We also proposed an optimization of the plane-probing algorithm to reduce the number of predicate calls. This improvement could be extended to the cases where we probe on a digital surface.



(a) (b)

Fig. 9: The evolution of the algorithm R(a) and L(b) for the normal vector $(2, 5, 156)$.



(a) (b) (c)

Fig. 10: Various measures for $\mathbf{N} = (198, 195, 193)$ per iteration during probings with the H, the R^1 and our L-algorithm: (from left to right) radius of $\mathcal{B}_{\mathbf{T}^{(i)},\mathbf{T}^{(i+1)}}$, maximal distance to $\mathbf{q}$, and perimeters of the triangles.

In the future, we wish to bound from above the distance of all vertices to the starting point in order to objectively measure the localness of plane-probing algorithms. Another perspective is to further optimize the L-algorithm to reduce its complexity to an amortized linear complexity.

Acknowledgements We would like to thank the anonymous reviewers for having devoted their time and effort to their extensive and constructive feedback. It helps us considerably improve the content of this paper.

A Inclusion relation in $\mathcal{H}_-$ used in Lemma 7

Lemma 8. *If* $\mathbf{y} \preceq_{\mathbf{T}} \mathbf{x}$, *then* $(\mathcal{B}(\mathbf{T}, \mathbf{x}) \cap \mathcal{H}_-) \subseteq (\mathcal{B}(\mathbf{T}, \mathbf{y}) \cap \mathcal{H}_-)$.

Proof. For any pair $\mathbf{x}, \mathbf{y} \in \mathcal{H}_-$, we denote $\mathbf{y} \preceq_{\mathbf{T}} \mathbf{x}$ if and only if $(\mathcal{B}(\mathbf{T}, \mathbf{y}) \cap \mathcal{H}_-) \subseteq (\mathcal{B}(\mathbf{T}, \mathbf{x}) \cap \mathcal{H}_-)$. As for $\preceq_{\mathbf{T}}$, note that $\preceq_{\mathbf{T}}$ is a total preorder. Let us consider now two points $\mathbf{x}' \in (\partial\mathcal{B}(\mathbf{T}, \mathbf{x}) \cap \mathcal{H}_-)$ and $\mathbf{y}' \in (\partial\mathcal{B}(\mathbf{T}, \mathbf{y}) \cap \mathcal{H}_-)$ (both points lie on the boundary of either $\mathcal{B}(\mathbf{T}, \mathbf{x})$ and $\mathcal{B}(\mathbf{T}, \mathbf{y})$ in $\mathcal{H}_-$). Note that, by construction, $\mathcal{B}(\mathbf{T}, \mathbf{y}') = \mathcal{B}(\mathbf{T}, \mathbf{y})$ and $\mathcal{B}(\mathbf{T}, \mathbf{x}') = \mathcal{B}(\mathbf{T}, \mathbf{x})$.

Since the relation $\preceq_{\mathbf{T}}$ is total, we have either $\mathbf{y}' \preceq_{\mathbf{T}} \mathbf{x}'$ or $\mathbf{x}' \preceq_{\mathbf{T}} \mathbf{y}'$. As the second case implies the remark statement by definition, we focus below on the first case. By

definition, $\mathbf{y}' \preceq_{\mathbf{T}} \mathbf{x}'$ implies $(\mathcal{B}(\mathbf{T},\mathbf{y}) \cap \mathcal{H}_-) \subseteq (\mathcal{B}(\mathbf{T},\mathbf{x}) \cap \mathcal{H}_-)$. Since we assume $\mathbf{y} \preceq_{\mathbf{T}} \mathbf{x}$, we also have $(\mathcal{B}(\mathbf{T},\mathbf{y}) \cap \mathcal{H}_+) \subseteq (\mathcal{B}(\mathbf{T},\mathbf{x}) \cap \mathcal{H}_+)$ by definition. If we take the union of both sides of the inclusion, we have $\mathcal{B}(\mathbf{T},\mathbf{y}) \subseteq \mathcal{B}(\mathbf{T},\mathbf{x})$. If $\mathcal{B}(\mathbf{T},\mathbf{y}) = \mathcal{B}(\mathbf{T},\mathbf{x})$, the overall remark statement is trivially true. If $\mathcal{B}(\mathbf{T},\mathbf{y}) \subset \mathcal{B}(\mathbf{T},\mathbf{x})$, we have a contradiction as both balls are constructed from the same triangle $\mathbf{T}$. □

References

1. Charrier, E., Buzer, L.: An efficient and quasi linear worst-case time algorithm for digital plane recognition. In: Discrete Geometry for Computer Imagery (DGCI'2008), LNCS, vol. 4992, pp. 346–357. Springer (2008)
2. Coeurjolly, D., Lachaud, J.O., Levallois, J.: Integral based Curvature Estimators in Digital Geometry. In: Gonzalez-Diaz, R., Jimenez, M.J., Medrano, B. (eds.) 17th International Conference on Discrete Geometry for Computer Imagery (DGCI 2013). pp. 215–227. 7749, Springer Verlag, Seville (Spain), Spain (Mar 2013)
3. Cuel, L., Lachaud, J.O., Mérigot, Q., Thibert, B.: Robust geometry estimation using the generalized voronoi covariance measure. SIAM Journal on Imaging Sciences **8**(2), 1293–1314 (2015)
4. Debled-Rennesson, I., Reveillès, J.: An incremental algorithm for digital plane recognition. In: Proc. Discrete Geometry for Computer Imagery. pp. 194–205 (1994)
5. Gérard, Y., Debled-Rennesson, I., Zimmermann, P.: An elementary digital plane recognition algorithm. Discrete Applied Mathematics **151**(1), 169–183 (2005)
6. Lachaud, J.O., Meyron, J., Roussillon, T.: An optimized framework for plane-probing algorithms. Journal of Mathematical Imaging and Vision **62**, 718–736 (2020)
7. Lachaud, J.O., Provençal, X., Roussillon, T.: An output-sensitive algorithm to compute the normal vector of a digital plane. Journal of Theoretical Computer Science (TCS) **624**, 73–88 (Apr 2016)
8. Lachaud, J.O., Provençal, X., Roussillon, T.: Computation of the normal vector to a digital plane by sampling signicant points. In: 19th IAPR International Conference on Discrete Geometry for Computer Imagery. Nantes, France (Apr 2016)
9. Lachaud, J.O., Provençal, X., Roussillon, T.: Two Plane-Probing Algorithms for the Computation of the Normal Vector to a Digital Plane. Journal of Mathematical Imaging and Vision **59**(1), 23 – 39 (Sep 2017)
10. Reveillès, J.P.: Géométrie Discrète, calculs en nombres entiers et algorithmique. Thèse d'etat, Université Louis Pasteur (1991)
11. Roussillon, T., Lachaud, J.O.: Digital Plane Recognition with Fewer Probes. In: 21st IAPR International Conference on Discrete Geometry for Computer Imagery. Lecture Notes in Computer Science, vol. 11414, pp. 380–393. Couprie M. and Cousty J. and Kenmochi Y. and Mustafa N., Springer, Cham, Marne-la-Vallée, France (Mar 2019)
12. Roussillon, T., Lu, J.T., Lachaud, J.O., Coeurjolly, D.: Delaunay property and proximity results of the L-algorithm. Research report, Université de Lyon (Jul 2022), https://hal.archives-ouvertes.fr/hal-03719592
13. The DGtal Project: DGtal (2010), https://dgtal.org
14. Veelaert, P.: Digital planarity of rectangular surface segments. IEEE Transactions on Pattern Analysis and Machine Intelligence **16**(6), 647–652 (1994)