Plane-probing algorithms for the analysis of digital surfaces

Tristan Roussillon

Université de Lyon, INSA Lyon, LIRIS, France

DGDVC, 30/03/2021

PARADIS (ANR-18-CE23-0007-01) research grant
Data

voxel sets in 3d digital images
Digital surfaces

pros/cons

+ efficient spatial data structures
+ set operations (union, intersection, ...)
+ integer-only, exact computations
+ ...
- poor geometry
Digital surfaces

pros/cons

+ efficient spatial data structures
+ set operations (union, intersection, \ldots)
+ integer-only, exact computations
+ \ldots

− poor geometry
Analysis of digital surfaces

- enhance the geometry by estimating normal vectors

⇒ applications: measurements, deformation for simulation or tracking, surface fairing, rendering...
A lot of methods

- fitting,
- Voronoi diagram,
- integral invariants,
- convolution,
- energy minimization,
- probabilistic approaches,
- ...
Flaw

Existing methods are not quite satisfactory

- parameter required ($\approx$ width of a neighborhood)
- that parameter is hard to pick
  - get decent estimates in flat/smooth parts
  - preserve sharp features
Challenge

Desiderata

- parameter-free method
- theoretical guarantees
  - exact on flat parts
  - converge on smooth parts as resolution increases

Key idea

- bound neighborhoods by their thickness instead of their width
- digitized planes have a thickness bounded by a small constant
Plane-probing algorithms

Definition
Given a digitized plane $\mathbf{P}$ and a starting point $\mathbf{p} \in \mathbf{P}$, a plane-probing algorithm computes the normal vector of $\mathbf{P}$ by sparsely probing it with the predicate "is $\mathbf{x} \in \mathbf{P}$?".

H and R

R$^1$

PH, PR, PR$^1$


Implemented in DGTAL (dgtal.org)
Outline

Context and motivation

Plane-probing algorithms
  Generalized Euclidean algorithm
  Delaunay triangulation
  Generalization

Application to digital surfaces
Euclidean algorithm

Given a couple of integers,

\[ \begin{align*}
\text{subtract the smaller from the larger one, and repeat} \\
\text{until both numbers are equal.}
\end{align*} \]

Example

<table>
<thead>
<tr>
<th>step</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( b )</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

we focus on the sequence of subtractions, assume \( \gcd(a, b) = 1 \)
One geometrical interpretation of the Euclidean algorithm

\[
\begin{align*}
\mathbf{m}_1 &= (1, 0), \quad \mathbf{m}_1 \cdot \mathbf{N} = a = 3 \\
\mathbf{m}_2 &= (0, 1), \quad \mathbf{m}_2 \cdot \mathbf{N} = b = 8
\end{align*}
\]
One geometrical interpretation of the Euclidean algorithm

\[ m_1 = (1, 0), \quad m_1 \cdot N = a = 3 \]
\[ m_2 = (-1, 1), \quad m_2 \cdot N = b = 5 \]
One geometrical interpretation of the Euclidean algorithm

\[ m_1 = (1, 0), \quad m_1 \cdot N = a = 3 \]
\[ m_2 = (-2, 1), \quad m_2 \cdot N = b = 2 \]
One geometrical interpretation of the Euclidean algorithm

\[ m_1 = (3, -1), \quad m_1 \cdot N = a = 1 \]
\[ m_2 = (-2, 1), \quad m_2 \cdot N = b = 2 \]
One geometrical interpretation of the Euclidean algorithm

\[ m_1 = (3, -1), \quad m_1 \cdot N = a = 1 \]
\[ m_2 = (-5, 2), \quad m_2 \cdot N = b = 1 \]
An algorithm to compute \( N \)

\( N \) is unknown, but a predicate IsBlack is given

\( \text{IsBlack}(m_1 - m_2)? \text{IsBlack}(m_2 - m_1)? \)
An algorithm to compute $\mathbf{N}$

$\mathbf{N}$ is unknown, but a predicate $\text{IsBlack}$ is given

$\text{IsBlack}(\mathbf{m}_1 - \mathbf{m}_2) \text{? } \text{IsBlack}(\mathbf{m}_2 - \mathbf{m}_1)$?
An algorithm to compute $\mathbf{N}$

$\mathbf{N}$ is unknown, but a predicate $\text{IsBlack}$ is given:

$\text{IsBlack}(\mathbf{m}_1 - \mathbf{m}_2)$? $\text{IsBlack}(\mathbf{m}_2 - \mathbf{m}_1)$?
An algorithm to compute $N$

$N$ is unknown, but a predicate $\text{IsBlack}$ is given
$\text{IsBlack}(m_1 - m_2) \,? \, \text{IsBlack}(m_2 - m_1)$?
An algorithm to compute $N$

$N$ is unknown, but a predicate $\text{IsBlack}$ is given

$\text{IsBlack}(m_1 - m_2)$? $\text{IsBlack}(m_2 - m_1)$?
Extension to 3d

No unique extension to the Euclidean algorithm!

Assuming $0 \leq a \leq b \leq c$ :

- **Brun**: $(a, b, c) \rightarrow (a, b, c - b)$;
- **Selmer**: $(a, b, c) \rightarrow (a, b, c - a)$;
- **Farey**: $(a, b, c) \rightarrow (a, b - a, c)$;
- **Fully-Subtractive**: $(a, b, c) \rightarrow (a, b - a, c - a)$;
- **Poincaré**: $(a, b, c) \rightarrow (a, b - a, c - b)$.
- ... 

Note: the same operation is done at each step
A class of generalized Euclidean algorithms

Given three positive numbers \((a, b, c)\), with \(\gcd(a, b, c) = 1\),

▶ while they are not all equal to 1,
▶ subtract from a number \(x \in \{a, b, c\}\) a strictly smaller
  number \(y \in \{a, b, c\}\), \(y < x\).

Example

\[
\begin{align*}
\mathbf{m}_1 &= (1, 0, 0), & \mathbf{m}_1 \cdot \mathbf{N} &= a = 1 \\
\mathbf{m}_2 &= (0, 1, 0), & \mathbf{m}_2 \cdot \mathbf{N} &= b = 2 \\
\mathbf{m}_3 &= (0, 0, 1), & \mathbf{m}_3 \cdot \mathbf{N} &= c = 3
\end{align*}
\]
A class of generalized Euclidean algorithms

Given three positive numbers \((a, b, c)\), with \(\gcd(a, b, c) = 1\),

- while they are not all equal to 1,
- subtract from a number \(x \in \{a, b, c\}\) a strictly smaller number \(y \in \{a, b, c\}\), \(y < x\).

Example

\[
\begin{align*}
\mathbf{m}_1 &= (1, 0, 0), & \mathbf{m}_1 \cdot \mathbf{N} &= a = 1 \\
\mathbf{m}_2 &= (0, 1, 0), & \mathbf{m}_2 \cdot \mathbf{N} &= b = 2 \\
\mathbf{m}_3 &= (0, -1, 1), & \mathbf{m}_3 \cdot \mathbf{N} &= c = 1
\end{align*}
\]
A class of generalized Euclidean algorithms

Given three positive numbers \((a, b, c)\), with \(\gcd(a, b, c) = 1\),

- while they are not all equal to 1,
- subtract from a number \(x \in \{a, b, c\}\) a strictly smaller number \(y \in \{a, b, c\}, y < x\).

Example

\[
\begin{align*}
\mathbf{m}_1 &= (1, 0, 0), & \mathbf{m}_1 \cdot \mathbf{N} &= a = 1 \\
\mathbf{m}_2 &= (-1, 1, 0), & \mathbf{m}_2 \cdot \mathbf{N} &= b = 1 \\
\mathbf{m}_3 &= (0, -1, 1), & \mathbf{m}_3 \cdot \mathbf{N} &= c = 1
\end{align*}
\]
Digital plane

Let \( \mathbf{N} \in \mathbb{Z}^3 \) whose components \((a, b, c)\) are coprime integers s.t. \(0 < a \leq b \leq c\),

\[
P_N := \{ \mathbf{x} \in \mathbb{Z}^3 \mid 0 \leq \mathbf{x} \cdot \mathbf{N} < \| \mathbf{N} \|_1 \}
\]
Interpretation of a generalized Euclidean algorithm

Internals

\[ (m_1, m_2, m_3) := (e_1, e_2, e_3), \quad q := (1, 1, 1) \notin \mathbb{P}_N \]

\[ \Rightarrow \text{triangle} \ (q - m_1, q - m_2, q - m_3) \]

\[ \Rightarrow \text{hexagon} \ \{q + m_i - m_j \mid i, j \in \{1, 2, 3\}, \ i \neq j\} \]
Interpretation of a generalized Euclidean algorithm

Internals

\[ (m_1, m_2, m_3) := (e_1, e_2, e_3), \quad q := (1, 1, 1) \notin P_N \]

\[ \Rightarrow \text{triangle} \ (q - m_1, q - m_2, q - m_3) \]

\[ \Rightarrow \text{hexagon} \ \{ q + m_i - m_j \mid i, j \in \{1, 2, 3\}, i \neq j \} \]
Interpretation of a generalized Euclidean algorithm

Internals

\[(m_1, m_2, m_3) := (e_1, e_2, e_3), \quad q := (1, 1, 1) \notin P_N\]

\[\Rightarrow \text{triangle } (q - m_1, q - m_2, q - m_3)\]

\[\Rightarrow \text{hexagon } \{q + m_i - m_j \mid i, j \in \{1, 2, 3\}, i \neq j\}\]
Interpretation of a generalized Euclidean algorithm

Internals

► \((m_1, m_2, m_3) := (e_1, e_2, e_3), \ q := (1, 1, 1) \not\in \mathbb{P}_N\)

⇒ triangle \((q - m_1, q - m_2, q - m_3)\)

⇒ hexagon \(\{q + m_i - m_j \mid i, j \in \{1, 2, 3\}, i \neq j\}\)
Interpretation of a generalized Euclidean algorithm

Internals

- $(m_1, m_2, m_3) := (e_1, e_2, e_3)$, $q := (1, 1, 1)$ \( \not\in \mathbb{P}_N \)

- triangle $(q - m_1, q - m_2, q - m_3)$

- hexagon $\{q + m_i - m_j \mid i, j \in \{1, 2, 3\}, i \neq j\}$
a plane-probing algorithm

\[ \Pi := \{ \mathbf{P}_N \mid \mathbf{N} \in \mathbb{Z}^3 \setminus \{0\} \} \]

**Input**

- \( \mathbf{P} \in \Pi \) described by the predicate \( \text{InPlane}: \text{"is } \mathbf{x} \in \mathbf{P}?" \)
- a starting point \( \mathbf{p} \) s.t. \( \text{InPlane}(\mathbf{p}), \mathbf{q} := \mathbf{p} + (1, 1, 1) \)

**Main trick**

- Assume \( \mathbf{p} \cdot \mathbf{N} = 0 \) (\( \Rightarrow \mathbf{q} \cdot \mathbf{N} = \|\mathbf{N}\|_1 \)), where \( \mathbf{N} \), the normal of \( \mathbf{P} \)
- \( \text{InPlane}(\mathbf{x}) \Leftrightarrow (\mathbf{x} - \mathbf{q}) \cdot \mathbf{N} < 0. \)
Properties of generalized Euclidean algorithms

At each step

P1 \( \mathbf{p} \) and \( \mathbf{q} \) both project into triangle \((\mathbf{q} - \mathbf{m}_1, \mathbf{q} - \mathbf{m}_2, \mathbf{q} - \mathbf{m}_3)\)
along \((1, 1, 1)\)

P2 matrix \( \mathbf{M} := [\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3] \) is unimodular, i.e. \( \det (\mathbf{M}) = 1 \)

Termination

- number of steps \( \leq \|\mathbf{N}\|_1 - 3 \) (6 calls to InPlane per step)
- at the end, if \( \mathbf{p} \cdot \mathbf{N} = 0 \) (\( \Rightarrow \) \( \mathbf{q} \cdot \mathbf{N} = \|\mathbf{N}\|_1 \))
  \( \forall k \in \{1, 2, 3\}, \; \mathbf{m}_k \cdot \mathbf{N} = 1 \)
  \( \Rightarrow \) the normal of triangle \((\mathbf{q} - \mathbf{m}_1, \mathbf{q} - \mathbf{m}_2, \mathbf{q} - \mathbf{m}_3)\) is \( \mathbf{N} \)

whichever the subtraction we choose
Example

Digital plane of normal \((5, 2, 3)\)
Example

Digital plane of normal $(5, 2, 3)$
Example

Digital plane of normal $(5, 2, 3)$
Digital plane of normal \((5, 2, 3)\)
Example

Digital plane of normal \((5, 2, 3)\)
Example

Digital plane of normal \((5, 2, 3)\)
All possible final triangles

Digital plane of normal (2, 3, 5)
About final triangles

- vertices $\in \Lambda := \{x \in \mathbb{Z}^3 \mid x \cdot N = \|N\|_1 - 1\}$
- do not contain any other point of $\Lambda$ (P2)
- projection of $p$ along $(1,1,1)$ (P1)

Digital plane of normal $(2,2,5)$

```
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
```
Towards a selection criterion

- The Delaunay triangulation of $\Lambda$ gives acute triangles
- $p$ projects into one of them (if no co-circularity)

Digital plane of normal $(2, 2, 5)$
Towards a selection criterion

- The Delaunay triangulation of $\Lambda$ gives acute triangles
- $p$ projects into one of them (if no co-circularity)

Digital plane of normal (2, 2, 5)
Algorithm H (candidates in an hexagon)

At each step:

- consider a candidate set $S$
- filter $S$ through InPlane
- pick a closest point $s^*$: the circumsphere of $T \cup s^*$ doesn’t contain any other
- update $T$ with this point

The last triangle is very often acute, but not always
Algorithm H (candidates in an hexagon)

At each step:

- consider a candidate set $S$
- filter $S$ through InPlane
- pick a closest point $s^*$: the circumsphere of $T \cup s^*$ doesn’t contain any other
- update $T$ with this point

The last triangle is very often acute, but not always
Algorithm H (candidates in an hexagon)

At each step:

- consider a candidate set $S$
- filter $S$ through InPlane
- pick a closest point $s^*$:
  the circumsphere of $T \cup s^*$ doesn’t contain any other
- update $T$ with this point

The last triangle is very often acute, but not always
Algorithm H (candidates in an hexagon)

At each step:

- consider a candidate set $S$
- filter $S$ through InPlane
- pick a closest point $s^*$: the circumsphere of $T \cup s^*$ doesn’t contain any other
- update $T$ with this point

The last triangle is very often acute, but not always
Algorithm H (candidates in an hexagon)

At each step:
- consider a candidate set $S$
- filter $S$ through InPlane
- pick a closest point $s^*$: the circumsphere of $T \cup s^*$ doesn’t contain any other
- update $T$ with this point

The last triangle is very often acute, but not always
Algorithm R (candidates along rays)

- same algorithm as before, only $S$ differs
- $S$ is infinite but the filtering by InPlane gives a finite point set
- $O(\|N\|_1)$ steps, $O(\log(\|N\|_1))$ calls to InPlane per step
- the last triangle is always acute
Example

Digital plane of normal \((2, 3, 9)\)
Example

Digital plane of normal (2, 3, 9)
Example

Digital plane of normal \((2, 3, 9)\)
Example

Digital plane of normal (2, 3, 9)
Example

Digital plane of normal \((2, 3, 9)\)
Example

Digital plane of normal $(2, 3, 9)$
Algorithm $R^1$

Features

- has the same output as $R$
- but $O(\|N\|_1)$ calls to InPlane instead of $O(\|N\|_1 \log \|N\|_1)$

How?

1. local probing: 6 rays $\rightarrow$ at most 2 rays and 1 point
2. geometrical study: 2 rays $\rightarrow$ 1 ray and 1 point
3. efficient algorithm: 1 ray and 1 point $\rightarrow$ a closest point
Example

Digital plane of normal (67, 1, 91)
Example

Digital plane of normal (1, 73, 100)
Recap

Main features

- \( \mathbf{N} \) from a point \( \mathbf{p} \) s.t. \( \mathbf{p} \cdot \mathbf{N} = 0 \)
- by sparse and local computations:
  - \( \mathbf{p} \) projects into all triangles
  - with \( \mathbf{R} \) and \( \mathbf{R}^1 \), the current triangle is acute every two steps, always acute at the end
- \( O(\|\mathbf{N}\|_1) \) calls to \text{InPlane} \) with \( \mathbf{H} \) and \( \mathbf{R}^1 \),
  \( O(\|\mathbf{N}\|_1 \log (\|\mathbf{N}\|_1)) \) with \( \mathbf{R} \)

Drawbacks

1. do not retrieve \( \mathbf{N} \) from any point
2. do not retrieve all triangles of the lattice \( \Lambda \)
Problem #1: starting from any point

Input
▶ $P$ of normal $N$
▶ $\text{InPlane}: \text{"is } x \in P?\text{"}$

Equivalence used so far
▶ assume $q \cdot N = \|N\|_1$
▶ $\text{InPlane}(x) \Leftrightarrow (x - q) \cdot N < 0$

Generalized equivalence
▶ assume $q \cdot N \geq \|N\|_1$
▶ $\exists l \in \mathbb{N}$ s.t. $\text{InPlane}(q + l(x - q)) \Leftrightarrow (x - q) \cdot N < 0$. 
Predicate NotAbove

**Data:** InPlane, \( q \) and an integer \( L \geq 2 \| N \|_1 \)

**Input:** A point \( x \in \mathbb{Z}^3 \) s.t. \( q \cdot N - \| N \|_1 \leq x \cdot N \)

**Output:** True iff \( (x - q) \cdot N < 0 \) in \( O(\log (L)) \) calls to InPlane

1. \( u \leftarrow x - q \); // direction
2. \( l \leftarrow 1; \)
3. while \( l < L \) do
4.     if InPlane\((q + lu)\) then return True ;
5.     if InPlane\((q - lu)\) then return False ;
6.     \( l \leftarrow 2l; \)
7. return False;

it is enough to use NotAbove instead of InPlane
**Predicate NotAbove**

**Data:** \( \text{InPlane}, \ q \) and an integer \( L \geq 2 \| \mathbf{N} \|_1 \)

**Input:** A point \( x \in \mathbb{Z}^3 \) s.t. \( q \cdot \mathbf{N} - \| \mathbf{N} \|_1 \leq x \cdot \mathbf{N} \)

**Output:** True iff \( (x - q) \cdot \mathbf{N} < 0 \) in \( O(\log(L)) \) calls to \( \text{InPlane} \)

1. \( u \leftarrow x - q \); // direction
2. \( l \leftarrow 1; \)
3. while \( l < L \) do
4.     if \( \text{InPlane}(q + lu) \) then return True;
5.     if \( \text{InPlane}(q - lu) \) then return False;
6.     \( l \leftarrow 2l; \)
7. return False;

\[ q \quad u \quad x \]

It is enough to use NotAbove instead of InPlane
Problem #2: retrieving all triangles

Triangle \rightarrow\text{Parallelepiped}

- \text{top point } q
- \text{upper triangle } (q - m_1, q - m_2, q - m_3)
- \text{lower triangle } (q - m_2 - m_3, q - m_3 - m_1, q - m_1 - m_2)
- \text{bottom point } q - \sum_k m_k
Problem #2: retrieving all triangles

Triangle $\rightarrow$ Parallelepiped

- top point $q$
- upper triangle $(q - m_1, q - m_2, q - m_3)$
- lower triangle $(q - m_2 - m_3, q - m_3 - m_1, q - m_1 - m_2)$
- bottom point $q - \sum_k m_k$
Problem #2: retrieving all triangles

Triangle $\rightarrow$ Parallelepiped

- top point $q$
- upper triangle $(q - m_1, q - m_2, q - m_3)$
- lower triangle $(q - m_2 - m_3, q - m_3 - m_1, q - m_1 - m_2)$
- bottom point $q - \sum_k m_k$
Problem #2: retrieving all triangles

Triangle $\rightarrow$ Parallelepiped

- top point $q$
- upper triangle $(q - m_1, q - m_2, q - m_3)$
- lower triangle $(q - m_2 - m_3, q - m_3 - m_1, q - m_1 - m_2)$
- bottom point $q - \sum_k m_k$
Staying close to the digital plane

Update rule

- when the parallelepiped has less than 4 vertices in $P$, 
  $\Rightarrow$ the lower triangle is updated (top moves, not bottom)
- otherwise
  $\Rightarrow$ the upper triangle is updated (bottom moves, not top)
- invariant: at least one point in $P$ (bottom), one not (top)

Generalized versions of H, R and $R^1$

For each $X \in \{H, R, R^1\}$, $PX$ uses a parallelepiped and the above update rule with NotAbove instead of InPlane.
Recap

Main features

- N from any point p such that InPlane(p),
- all triangles of the lattice \( \Lambda = \{ x \in \mathbb{Z}^3 \mid x \cdot N = ||N||_1 - 1 \} \)
- PH and PR\(^1\) require \( O(||N||_1) \) calls to NotAbove
  \( \Rightarrow O(||N||_1 \log (||N||_1)) \) calls to InPlane.
Recap

Main features

- $\mathbf{N}$ from any point $\mathbf{p}$ such that $\text{InPlane}(\mathbf{p})$,
- all triangles of the lattice $\Lambda = \{ \mathbf{x} \in \mathbb{Z}^3 \mid \mathbf{x} \cdot \mathbf{N} = \|\mathbf{N}\|_1 - 1 \}$
- PH and PR$^1$ require $O(\|\mathbf{N}\|_1)$ calls to NotAbove

$\Rightarrow O(\|\mathbf{N}\|_1 \log (\|\mathbf{N}\|_1))$ calls to InPlane.
Outline

Context and motivation

Plane-probing algorithms
  Generalized Euclidean algorithm
  Delaunay triangulation
  Generalization

Application to digital surfaces
A similar algorithm for a digital surface $S$

Input

▶ a predicate $\text{InSurface} : x \in S$ ?
▶ a starting square face $s$ in $S$

Additional constraints

▶ find an origin and a basis from $s$

▶ stop if non-planar configurations (parallelepiped/hexagon/rays)
A similar algorithm for a digital surface $S$

Input

▶ a predicate $\text{InSurface}: x \in S$?
▶ a starting square face $s$ in $S$

Additional constraints

▶ find an origin and a basis from $s$

▶ stop if non-planar configurations (parallelepiped/hexagon/rays)
Example: flat parts and sharp edges
Example: flat parts and sharp edges
Example: flat parts and sharp edges
Example: flat parts and sharp edges
Example: convex shapes
Example: convex shapes
Example: convex shapes
Example: not convex shapes
Example: not convex shapes
Perspectives

Digital planes

- What piece of digital plane is enough to find \( N \)?

Digital surfaces

- try all candidates, obtuse triangles may be interesting
- perform a dense probing to process non-convex parts
- estimator: multigrid convergence, experimental comparison
- reconstruction: find of way of gluing triangles together
My first answer:

(a) snow  
(b) wood  
(c) foam