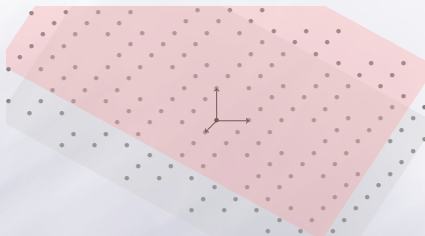


Computation of the Normal Vector to a Digital Plane
By Sampling Significant Points
DGCI'2016

J-O. Lachaud, X. Provençal, [T. Roussillon](#)

Digital plane



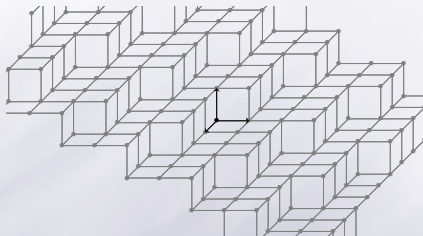
Standard (6-connected) digital plane

Let $\mathbf{N}(a, b, c)$ be a normal vector ($a, b, c \in \mathbb{Z}$, $\gcd(a, b, c) = 1$) and $\mu \in \mathbb{Z}$ be an intercept. A standard digital plane is defined as the set

$$\mathbf{P} = \{x \in \mathbb{Z}^3 \mid \mu \leq x \cdot \mathbf{N} < \mu + \omega\}.$$

We assume that $0 < a \leq b \leq c$, $\mu = 0$, $\omega = \|\mathbf{N}\|_1$.

Digital plane



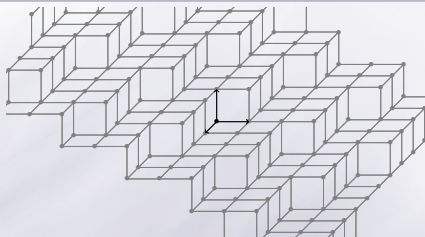
Standard (6-connected) digital plane

Let $\mathbf{N}(a, b, c)$ be a normal vector ($a, b, c \in \mathbb{Z}$, $\gcd(a, b, c) = 1$) and $\mu \in \mathbb{Z}$ be an intercept. A standard digital plane is defined as the set

$$\mathbf{P} = \{x \in \mathbb{Z}^3 \mid \mu \leq x \cdot \mathbf{N} < \mu + \omega\}.$$

We assume that $0 < a \leq b \leq c$, $\mu = 0$, $\omega = \|\mathbf{N}\|_1$.

Recognition problem



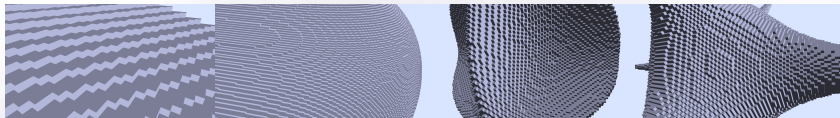
Problem

Knowing the digital set \mathbf{P} (and knowing that \mathbf{P} is a digital plane), find its normal vector \mathbf{N} .

Our approach in a nutshell

- ≡ we start from a trivial solution $\hat{\mathbf{N}}(1, 1, 1)$
- ≡ we iteratively improve it until $\hat{\mathbf{N}} = \mathbf{N}$

Extension to digital surfaces



Digital surface

A digital surface is defined as the topological border of a set of voxels. Note that a digital surface is locally a piece of standard digital plane (where digital points are the vertices of voxels)








Region growing algorithm

Knowing a digital surface, make a piece of digital plane grow while it is tangent and included into the digital surface. Note that we can add constraints (eg. closeness to the seed, compactness, ...).

Related works

Usual approach

We make a connected region grow (eg. breadth-first search in the adjacency graph) while it is a piece of digital plane (*recognition algorithm*).

-  E. Charrier and L. Buzer, *An efficient and quasi linear worst-case time algorithm for digital plane recognition*, DGCI'2008, LNCS, vol. 4992, Springer, 2008, pp. 346–357.
-  I. Debled-Rennesson and J.-P. Reveilles, *An incremental algorithm for digital plane recognition*, DGCI'1994, 1994, pp. 207–222.
-  Y. Gérard, I. Debled-Rennesson, and P. Zimmermann, *An elementary digital plane recognition algorithm*, Discrete Applied Mathematics 151 (2005), no. 1, 169–183.
-  C. E. Kim and I. Stojmenović, *On the recognition of digital planes in three-dimensional space*, Pattern Recognition Letters 12 (1991), no. 11, 665–669.
-  R. Klette and H. J. Sun, *Digital planar segment based polyhedrization for surface area estimation*, Proc. Visual form 2001, LNCS, vol. 2059, Springer, 2001, pp. 356–366.
-  L. Provot and I. Debled-Rennesson, *3d noisy discrete objects: Segmentation and application to smoothing*, Pattern Recognition 42 (2009), no. 8, 1626–1636.
-  P. Veelaert, *Digital planarity of rectangular surface segments*, Pattern Analysis and Machine Intelligence, IEEE Transactions on 16 (1994), no. 6, 647–652.

Motivation for another approach

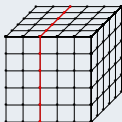
Main drawbacks of the usual approach

- (-) combinatorial explosion of pieces of digital plane, even of *maximal* ones ie. not included in greater pieces of digital plane.



D. Coeurjolly and I. Sivignon, *Minimum Decomposition of a Digital Surface into Digital Plane Segments is NP-Hard*, *Discrete Applied Mathematics*, 157(3), pp. 558–570.


- (-) maximal pieces of digital plane may be not tangent.



Our approach

We make a piece of digital plane grow while it is tangent.

Previous work into this direction

-  [LPR2016] J-O. Lachaud, X. Provençal, T. R. An output-sensitive algorithm to compute the normal vector of a digital plane. *Theoretical Computer Science*, 624:73–88, 2016.

Several operations based on the predicate “ $x \in \mathbf{P}$?”

- ≡ local: translation, Brun-Selmer, fully-subtractive
- ≡ non-local: generalization of Brun-Selmer et fully-subtractive

Algorithm

```
Input: a predicate “ $x \in \mathbf{P}$  ?”, a solution  $S$ .  
while true do  
  if there exists a valid operation  $\lambda$  then  
     $S \leftarrow \lambda(S)$   
  else break  
end  
return  $S$ 
```


Features of the proposed algorithm

A same framework, but a different algorithm

- ≡ we start by a trivial piece of digital plane of normal vector $\hat{\mathbf{N}}(1, 1, 1)$
- ≡ we make this piece of digital plane grow by **one operation**
- ≡ using only the predicate “is $x \in \mathbf{P}$?”
- ≡ **the piece of digital plane stays around the seed**
- ≡ the number of steps, which depends on ω , is output-sensitive.

Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

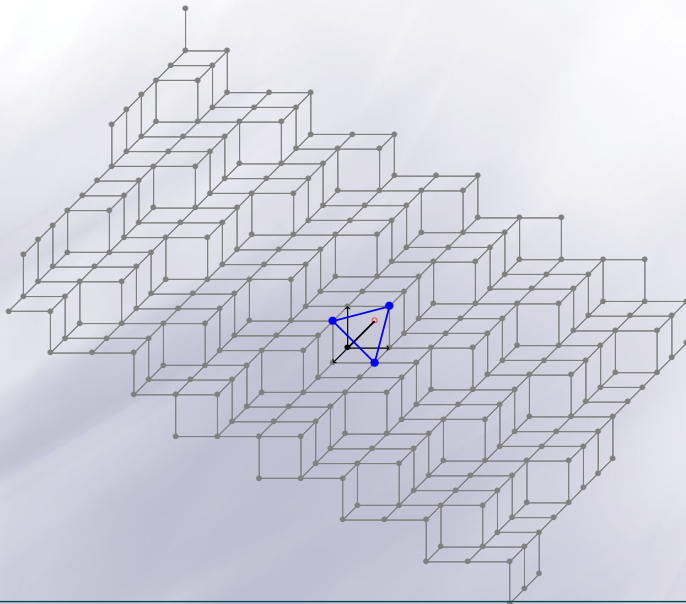


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

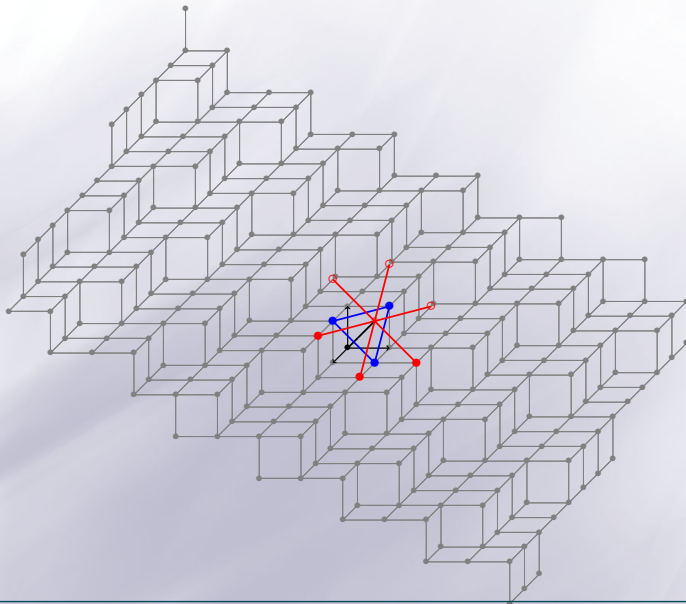


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

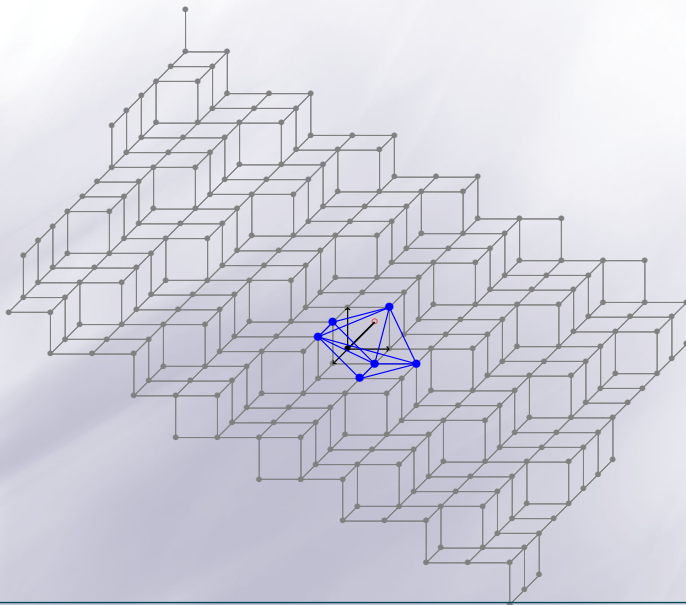


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

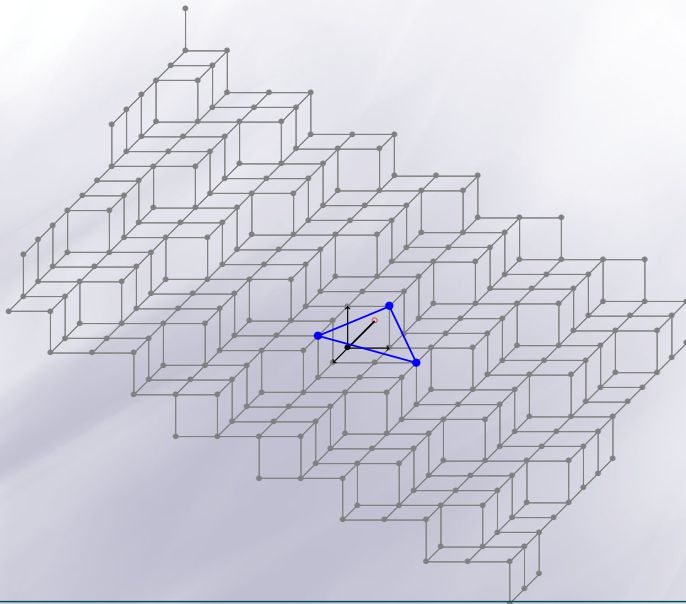


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

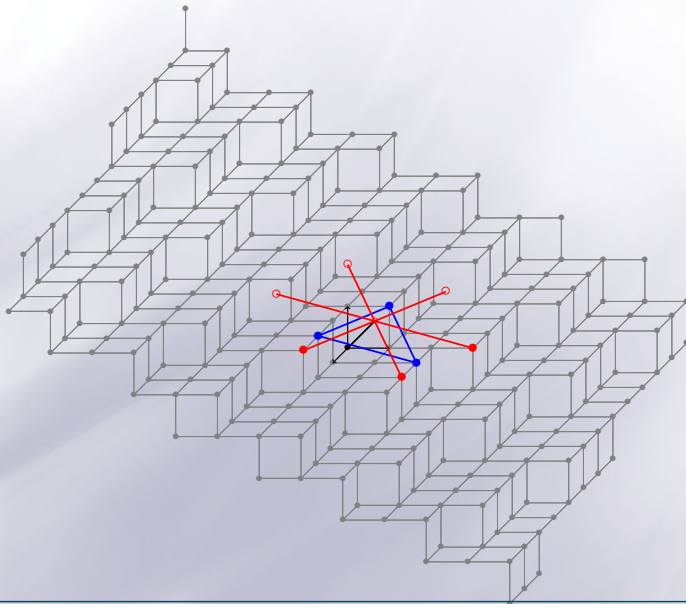


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

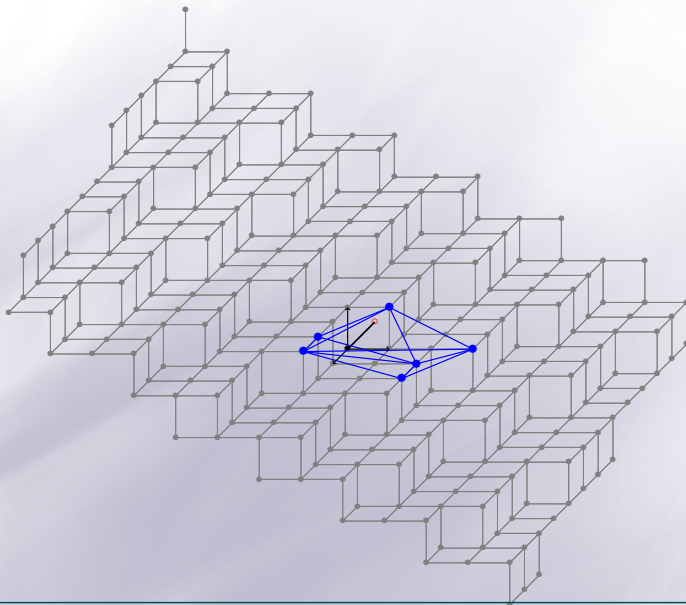


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

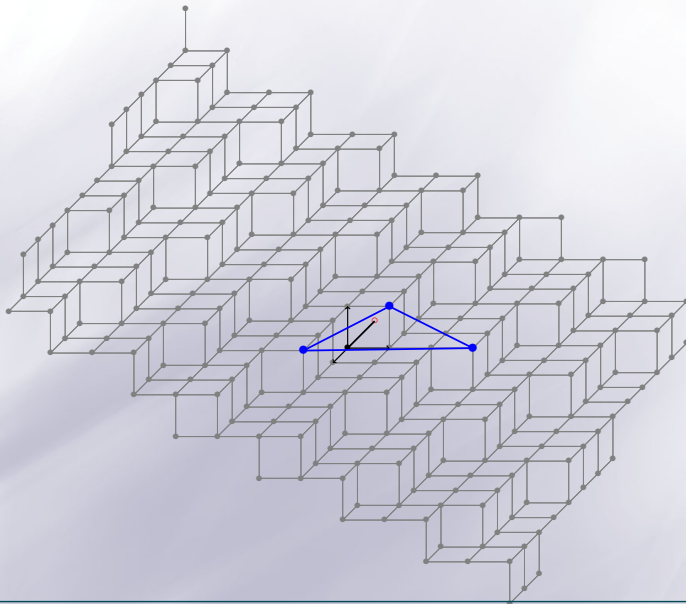


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

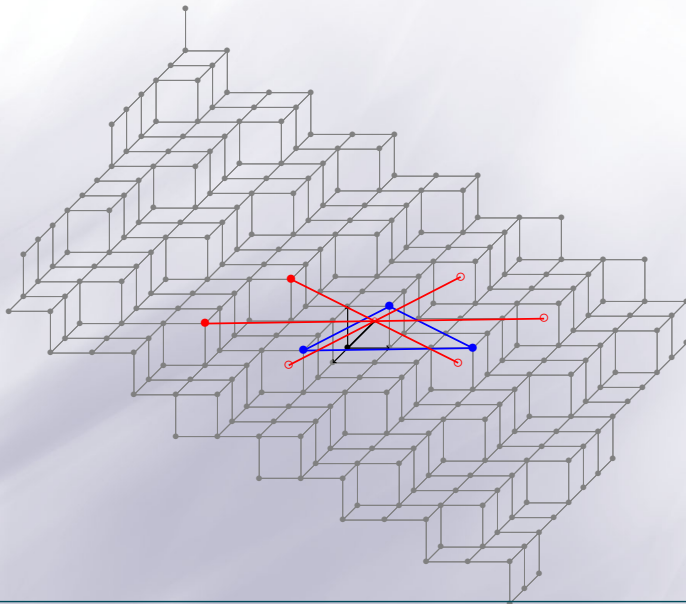


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

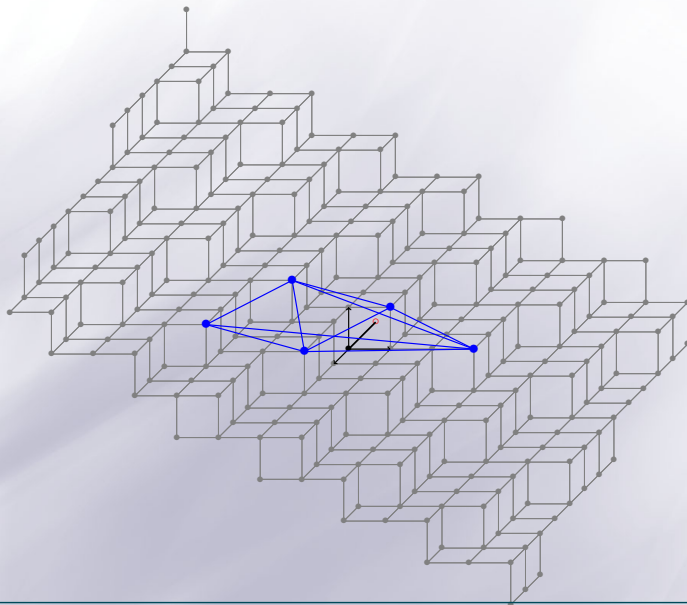


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin

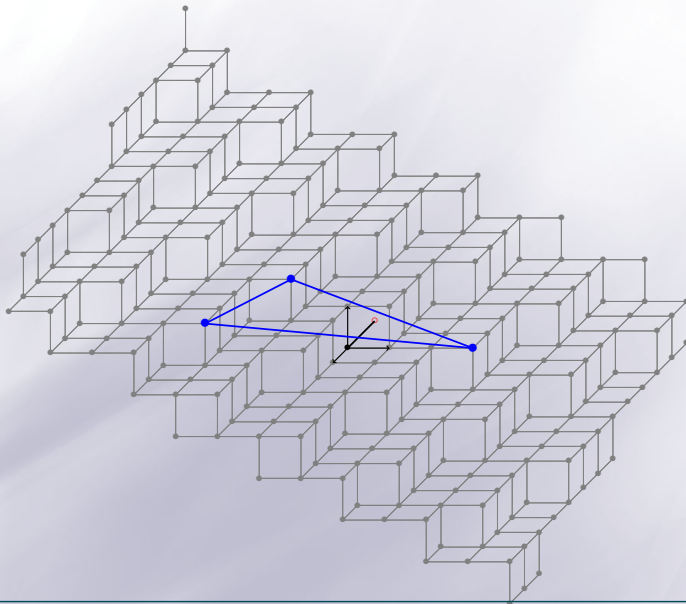
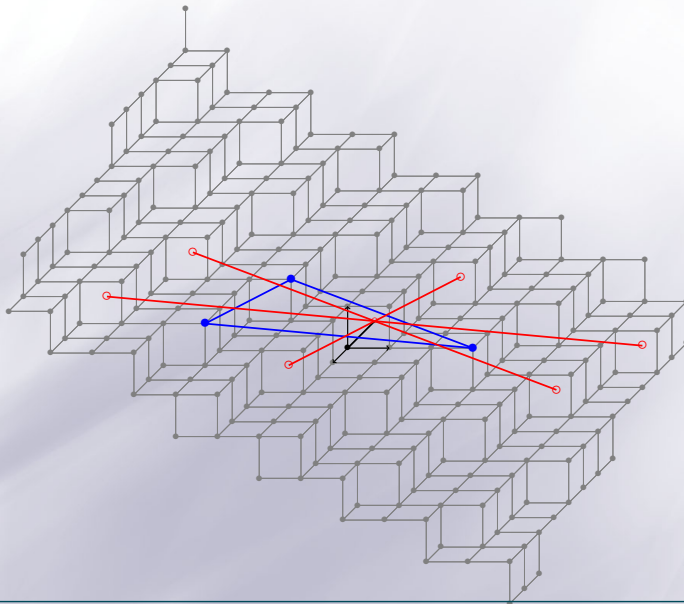


Illustration: $\mathbf{P}(2, 6, 11)$, starting from the origin



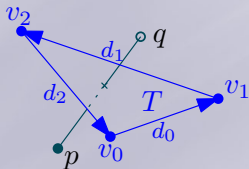
Notations

Segment $[pq]$

- ≡ We set $p \in \mathbf{P}$ and $q := p + (1, 1, 1)$ ($q \notin \mathbf{P}$).

Triangle

- ≡ At each step, a solution is described by a triangle T ,
- ≡ T is intersected by $[pq]$ (algorithm invariant),
- ≡ its vertices (ccw oriented) are denoted by v_k (k is mod. 3)
- ≡ and its edges are defined as $\forall k, d_k = v_{k+1} - v_k$.



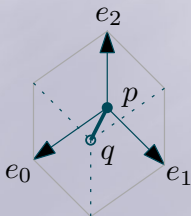
Initialization from an inside corner

Preconditions

- ≡ $p \in \mathbf{P} \Rightarrow q \notin \mathbf{P}$.
- ≡ $\{p + e_0 + e_1, p + e_1 + e_2, p + e_2 + e_0\} \subset \mathbf{P}$ (inside corner).

Starting triangle

- ≡ $\forall k, v_k := p + e_k + e_{k+1}$.
- ≡ $T := (v_k)_{k \in \{0,1,2\}}$.



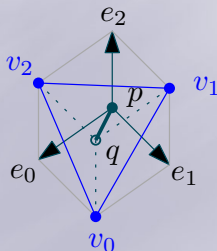
Initialization from an inside corner

Preconditions

- ≡ $p \in \mathbf{P} \Rightarrow q \notin \mathbf{P}$.
- ≡ $\{p + e_0 + e_1, p + e_1 + e_2, p + e_2 + e_0\} \subset \mathbf{P}$ (inside corner).

Starting triangle

- ≡ $\forall k, v_k := p + e_k + e_{k+1}$.
- ≡ $T := (v_k)_{k \in \{0,1,2\}}$.

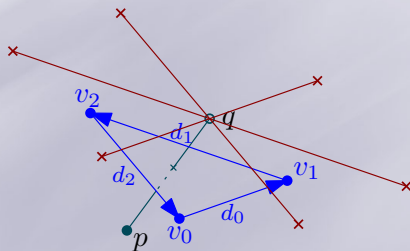


Growing procedure

Neighborhood

In order to make our piece of digital plane grow, we check whether the points located on a plane parallel to T , above T and “around” q are in \mathbf{P} or not. More precisely, we consider the set

$$\Sigma(T) := \{q \pm d_k\}_{k \in \{0,1,2\}}.$$



First version

Algorithm

```
Input: a predicate " $x \in \mathbf{P}$  ?", a starting triangle  $T$ .  
while  $((\Sigma(T) \cap \mathbf{P}) \neq \emptyset)$  do  
  Compute the convex hull of  $T \cup (\Sigma(T) \cap \mathbf{P})$   
  Find  $T'$ , the upper triangular facet intersected by  $[pq]$   
   $T \leftarrow T'$   
end  
return  $T$ 
```

Why does it work ?

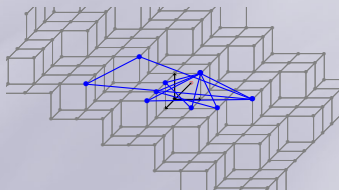
Height of a point

$\forall x \in \mathbb{Z}^3$, the dot product $x \cdot \mathbf{N}$ is called the *height* of x .



Rationale

we iteratively search for “higher” points of \mathbf{P} in direction \mathbf{N} , until three points whose height is maximal and equal to $\omega - 1$ are found.



Why does it work ?

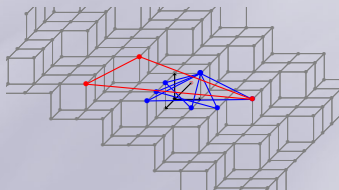
Height of a point

$\forall x \in \mathbb{Z}^3$, the dot product $x \cdot \mathbf{N}$ is called the *height* of x .



Rationale

we iteratively search for “higher” points of \mathbf{P} in direction \mathbf{N} , until three points whose height is maximal and equal to $\omega - 1$ are found.



Operation characterization

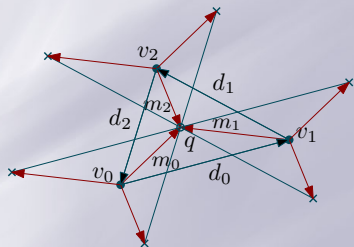
Edge vectors of the tetrahedron $T \cup q$, not in T

for each step, $\forall k, m_k := q - v_k$.

Operation

for each step, $\forall k, m'_k = \begin{cases} \text{(i)} & m_k \\ \text{(ii)} & m_k - m_l, l \in \{0, 1, 2\} \setminus k \end{cases}$

Case (i) occurs at least one time and at most two times over three.
(proof by case enumeration)



Operation characterization

Edge vectors of the tetrahedron $T \cup q$, not in T

for each step, $\forall k, m_k := q - v_k$.

Operation

for each step, $\forall k, m'_k = \begin{cases} \text{(i)} & m_k \\ \text{(ii)} & m_k - m_l, l \in \{0, 1, 2\} \setminus k \end{cases}$

Case (i) occurs at least one time and at most two times over three.
(proof by case enumeration)

Unimodular matrix

Let M the 3×3 matrix that consists of vectors $(m_k)_{k \in \{0,1,2\}}$. For each step, $\det(M) = 1$.
(proof by induction)

Algorithm correctness

Height of the last triangle

If p is a lower leaning point, ie $p.N = 0$ ($\Rightarrow q.N = \omega$), then $\forall k, v_k.N = \omega - 1$ at the last step.

- ≡ If $\exists k$ s.t. $d_k.N \neq 0$, then either $q - d_k$ or $q + d_k$ belongs to \mathbf{P} (because q belongs to the set of lowest points above \mathbf{P}), which implies that $\Sigma(T) \cap \mathbf{P} \neq \emptyset$, a contradiction.
- ≡ Therefore $\forall k, d_k.N = 0$ and $\forall k, m_k.N = c, c > 0$,
- ≡ which can be written as $MN = c\mathbf{1}$.
- ≡ Since M is invertible (unimodular), $N = (M)^{-1}c\mathbf{1}$ and $c = 1$ (components of N are relatively prime).

Algorithm correctness

Height of the last triangle

If p is a lower leaning point, ie $p.N = 0$ ($\Rightarrow q.N = \omega$), then $\forall k, v_k.N = \omega - 1$ at the last step.

- ≡ If $\exists k$ s.t. $d_k.N \neq 0$, then either $q - d_k$ or $q + d_k$ belongs to \mathbf{P} (because q belongs to the set of lowest points above \mathbf{P}), which implies that $\Sigma(T) \cap \mathbf{P} \neq \emptyset$, a contradiction.
- ≡ Therefore $\forall k, d_k.N = 0$ and $\forall k, m_k.N = c, c > 0$,
- ≡ which can be written as $MN = c\mathbf{1}$.
- ≡ Since M is invertible (unimodular), $N = (M)^{-1}c\mathbf{1}$ and $c = 1$ (components of N are relatively prime).

Algorithm correctness

Height of the last triangle

If p is a lower leaning point, ie $p.N = 0$ ($\Rightarrow q.N = \omega$), then $\forall k, v_k.N = \omega - 1$ at the last step.

- ≡ If $\exists k$ s.t. $d_k.N \neq 0$, then either $q - d_k$ or $q + d_k$ belongs to \mathbf{P} (because q belongs to the set of lowest points above \mathbf{P}), which implies that $\Sigma(T) \cap \mathbf{P} \neq \emptyset$, a contradiction.
- ≡ Therefore $\forall k, d_k.N = 0$ and $\forall k, m_k.N = c, c > 0$,
- ≡ which can be written as $MN = c\mathbf{1}$.
- ≡ Since M is invertible (unimodular), $N = (M)^{-1}c\mathbf{1}$ and $c = 1$ (components of N are relatively prime).

Algorithm correctness

Height of the last triangle

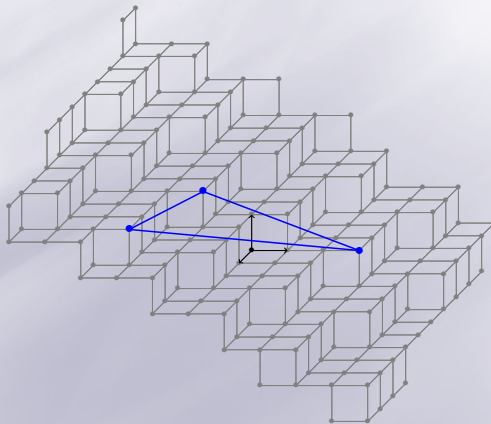
If p is a lower leaning point, ie $p.N = 0$ ($\Rightarrow q.N = \omega$), then $\forall k, v_k.N = \omega - 1$ at the last step.

- ≡ If $\exists k$ s.t. $d_k.N \neq 0$, then either $q - d_k$ or $q + d_k$ belongs to \mathbf{P} (because q belongs to the set of lowest points above \mathbf{P}), which implies that $\Sigma(T) \cap \mathbf{P} \neq \emptyset$, a contradiction.
- ≡ Therefore $\forall k, d_k.N = 0$ and $\forall k, m_k.N = c, c > 0$,
- ≡ which can be written as $MN = c\mathbf{1}$.
- ≡ Since M is invertible (unimodular), $N = (M)^{-1}c\mathbf{1}$ and $c = 1$ (components of N are relatively prime).

Lattice of upper leaning points

Corollary

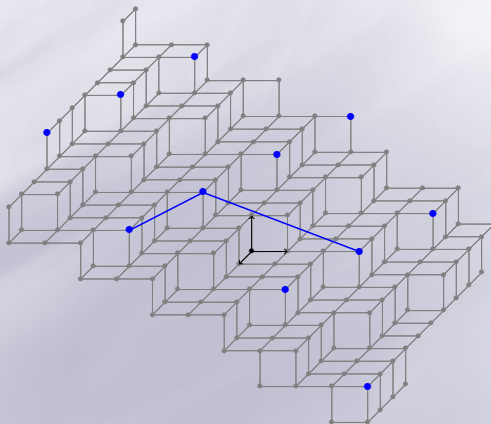
$(\pm d_k, \pm d_l)$ is a basis of the lattice of upper leaning points
 $\{x \in \mathbf{P} \mid x.N = \omega - 1\}$.



Lattice of upper leaning points

Corollary

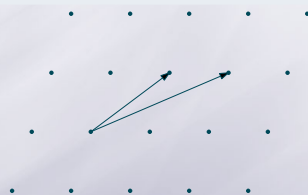
$(\pm d_k, \pm d_l)$ is a basis of the lattice of upper leaning points
 $\{x \in \mathbf{P} \mid x.N = \omega - 1\}$.



Lattice reduction

Reduced basis

A basis (u_1, u_2) is reduced iff $\|u_1\|, \|u_2\| \leq \|u_1 - u_2\| \leq \|u_1 + u_2\|$.

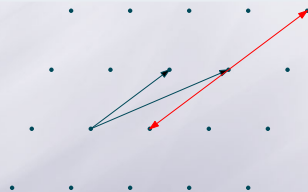


- ≡ the returned basis of this first version is generally not reduced
- ≡ reduction may be run as a post-processing
- ≡ but the following version returns basis that are almost always reduced

Lattice reduction

Reduced basis

A basis (u_1, u_2) is reduced iff $\|u_1\|, \|u_2\| \leq \|u_1 - u_2\| \leq \|u_1 + u_2\|$.

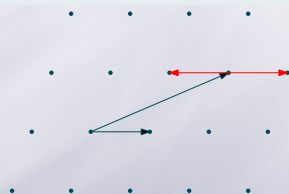


- ≡ the returned basis of this first version is generally not reduced
- ≡ reduction may be run as a post-processing
- ≡ but the following version returns basis that are almost always reduced

Lattice reduction

Reduced basis

A basis (u_1, u_2) is reduced iff $\|u_1\|, \|u_2\| \leq \|u_1 - u_2\| \leq \|u_1 + u_2\|$.

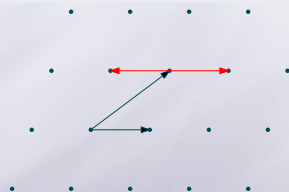


- ≡ the returned basis of this first version is generally not reduced
- ≡ reduction may be run as a post-processing
- ≡ but the following version returns basis that are almost always reduced

Lattice reduction

Reduced basis

A basis (u_1, u_2) is reduced iff $\|u_1\|, \|u_2\| \leq \|u_1 - u_2\| \leq \|u_1 + u_2\|$.

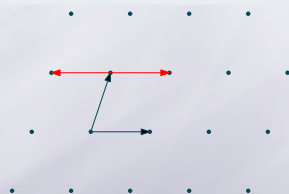


- ≡ the returned basis of this first version is generally not reduced
- ≡ reduction may be run as a post-processing
- ≡ but the following version returns basis that are almost always reduced

Lattice reduction

Reduced basis

A basis (u_1, u_2) is reduced iff $\|u_1\|, \|u_2\| \leq \|u_1 - u_2\| \leq \|u_1 + u_2\|$.



- ≡ the returned basis of this first version is generally not reduced
- ≡ reduction may be run as a post-processing
- ≡ but the following version returns basis that are almost always reduced

Second version

Algorithm

Input: a predicate " $x \in \mathbf{P} ?$ ", a starting triangle T .

while $(\Sigma(T) \cap \mathbf{P}) \neq \emptyset$ **do**

 Compute the set S^* of points $s^* \in (\Sigma(T) \cap \mathbf{P})$ such that the circumsphere of $T \cup s^*$ does not include any point $s \in (\Sigma(T) \cap \mathbf{P})$ in its interior

 Compute the convex hull of $T \cup S^*$

 Find T' , the upper triangular facet intersected by $[pq]$

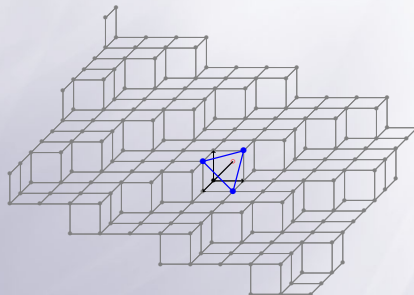
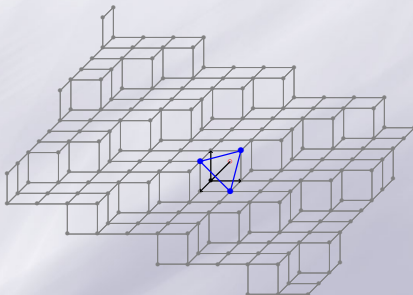
$T \leftarrow T'$

end

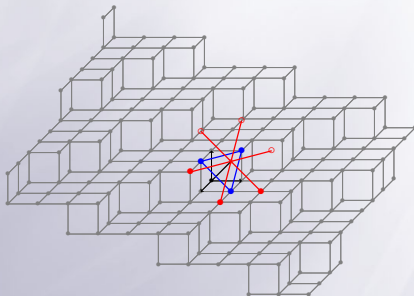
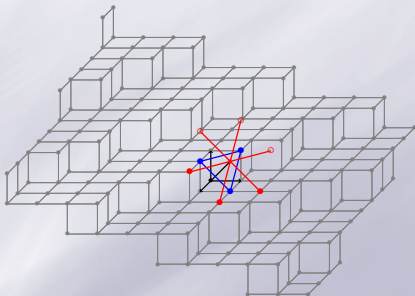
return T

NB. nothing change in the proofs because $S^* \subseteq (\Sigma(T) \cap \mathbf{P})$.

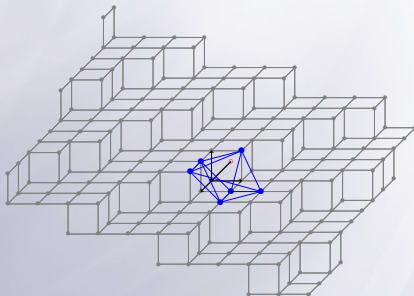
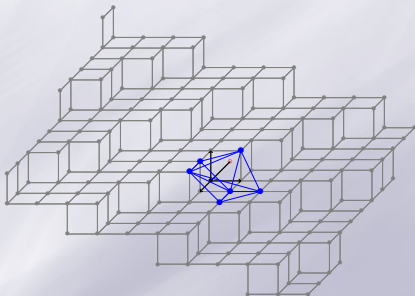
First vs second version: $(2, 3, 9)$, starting from the origin



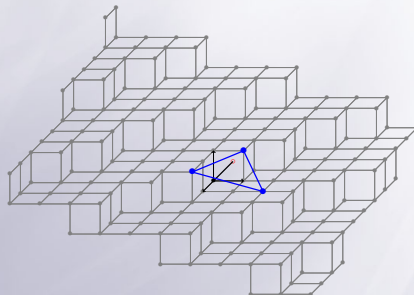
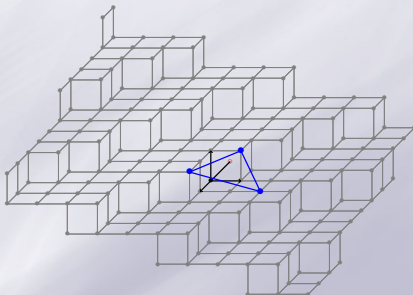
First vs second version: $(2, 3, 9)$, starting from the origin



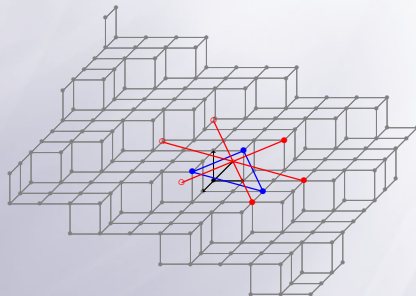
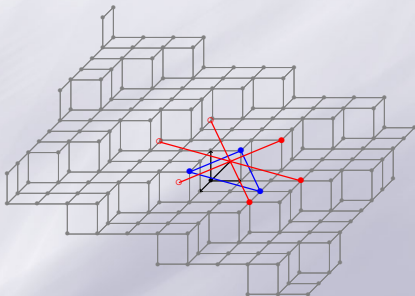
First vs second version: $(2, 3, 9)$, starting from the origin



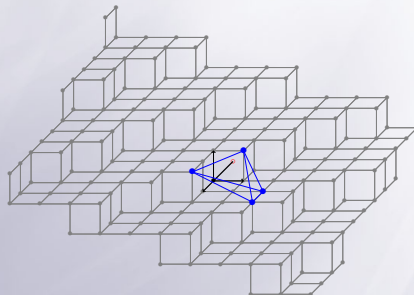
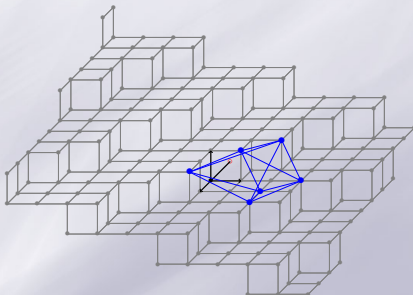
First vs second version: $(2, 3, 9)$, starting from the origin



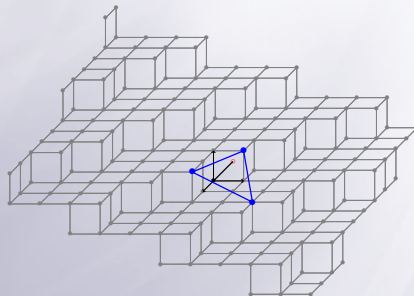
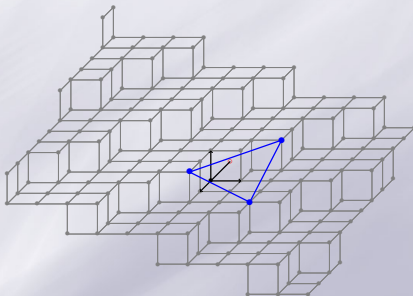
First vs second version: $(2, 3, 9)$, starting from the origin



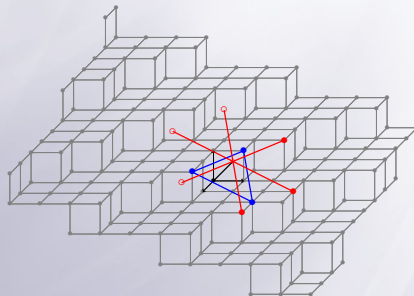
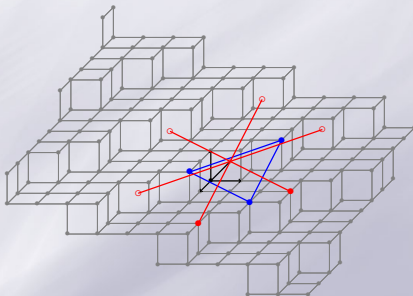
First vs second version: $(2, 3, 9)$, starting from the origin



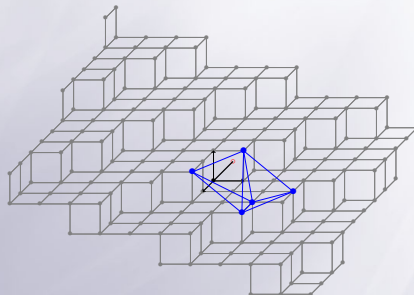
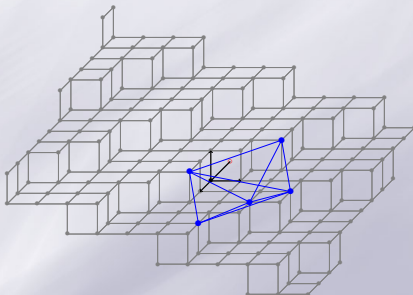
First vs second version: $(2, 3, 9)$, starting from the origin



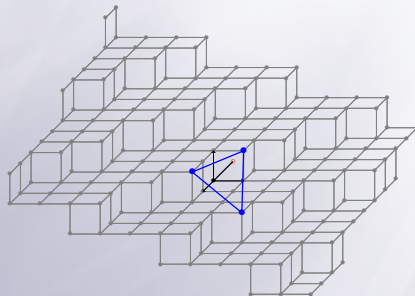
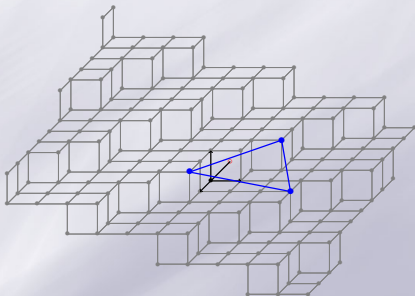
First vs second version: $(2, 3, 9)$, starting from the origin



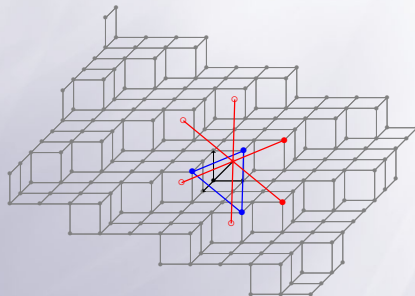
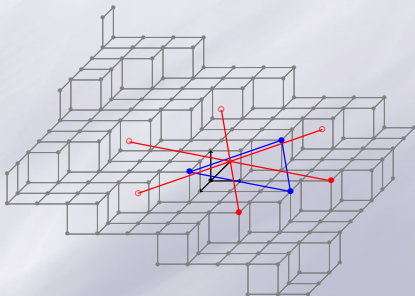
First vs second version: $(2, 3, 9)$, starting from the origin



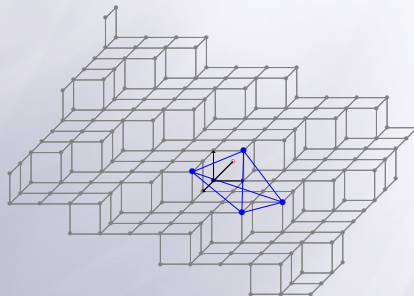
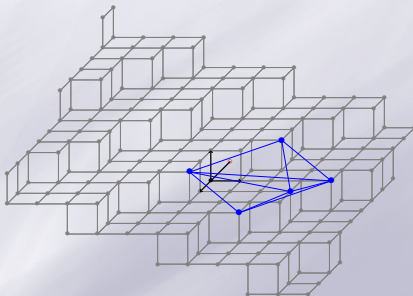
First vs second version: $(2, 3, 9)$, starting from the origin



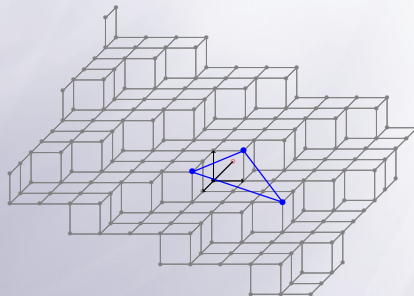
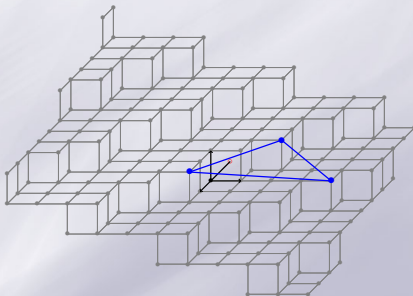
First vs second version: $(2, 3, 9)$, starting from the origin



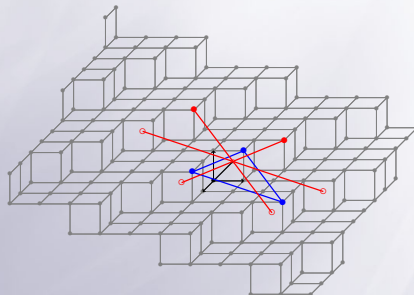
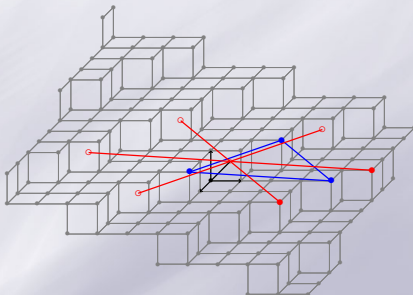
First vs second version: $(2, 3, 9)$, starting from the origin



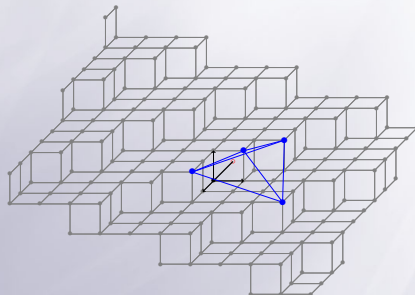
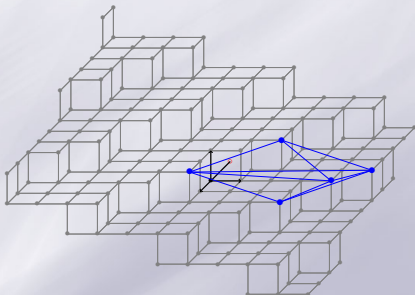
First vs second version: $(2, 3, 9)$, starting from the origin



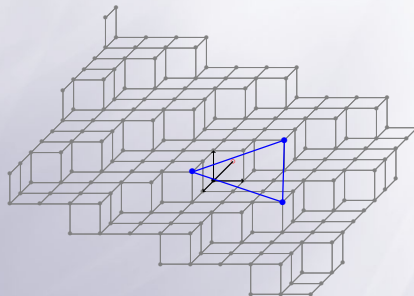
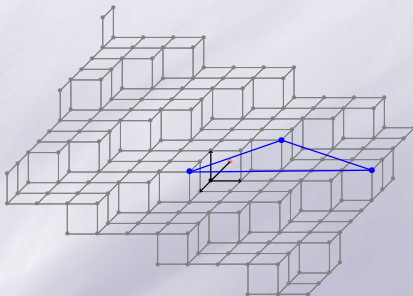
First vs second version: $(2, 3, 9)$, starting from the origin



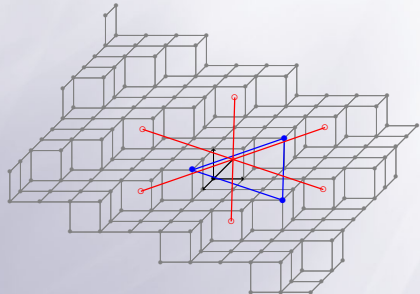
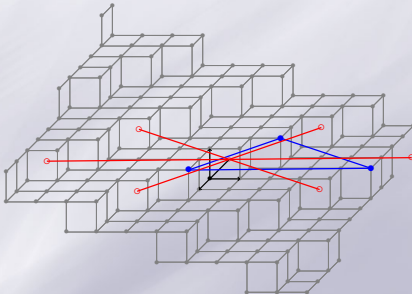
First vs second version: $(2, 3, 9)$, starting from the origin



First vs second version: $(2, 3, 9)$, starting from the origin



First vs second version: $(2, 3, 9)$, starting from the origin



Experimental analysis

Setting

- ≡ Vectors from (1,1,1) to (200,200,200)
- ≡ Number of tests : 6578833

Results

Version	First	Second
avg nb steps	21.8	20.9
nb non reduced	4803115 (73%)	924 (0.01%)

Conclusion

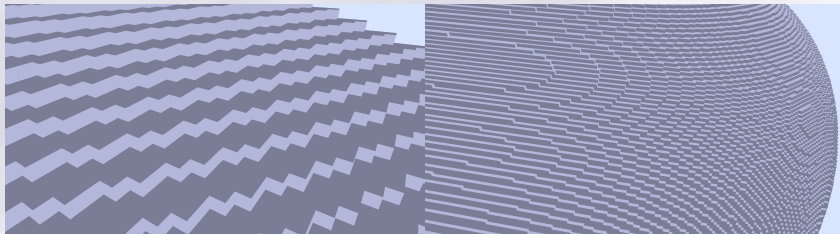
A new local and output-sensitive algorithm

- ≡ it starts by a trivial triangle of normal vector $\hat{N}(1, 1, 1)$
- ≡ it makes this triangle grow by **one operation**
- ≡ using the predicate “is $x \in \mathbf{P}$?”
- ≡ if the seed is a lower leaning point, it stops when $\hat{N} = \mathbf{N}$.
- ≡ a basis of upper leaning points is returned, which is located **around the seed and almost always reduced** (version 2).
- ≡ the number of steps is less than ω , each step is constant-time.

Extension to digital surfaces

Main idea

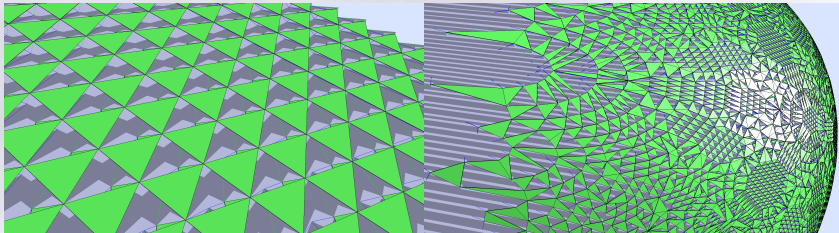
- we run our algorithm from each inside corner
- we discard “bad” triangles coming from “bad” corners.



Extension to digital surfaces

Main idea

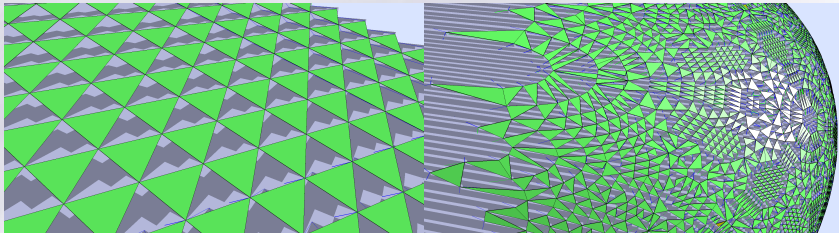
- we run our algorithm from each inside corner
- we discard “bad” triangles coming from “bad” corners.



Extension to digital surfaces

Main idea

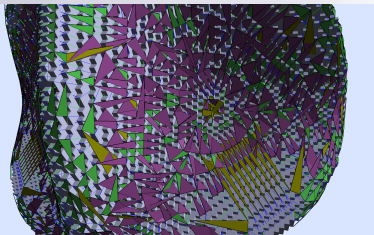
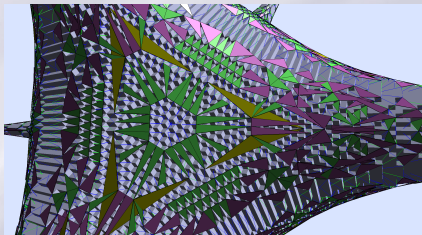
- ≡ we run our algorithm from each inside corner
- ≡ we discard “bad” triangles coming from “bad” corners.



Extension to digital surfaces

Main issues

- (?) process degenerate cases (flat in one direction)
- (?) find complementary triangles
- (?) discard triangles that intersect the background (in concave parts)



Thank you for your attention