Opening holes in Discrete Objects with Digital Homotopy

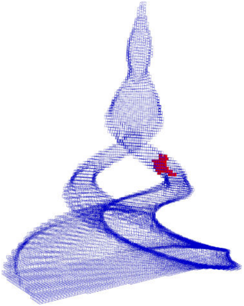
Aldo Gonzalez-Lorenzo   Alexandra Bac   Jean-Luc Mari

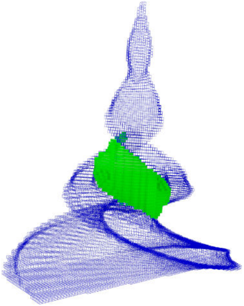
Aix-Marseille Université, CNRS, LSIS UMR 7296 (France)

20 september 2017

Structure

1 Opening and closing holes with homology





2 Closing holes in discrete objects

1. Opening and closing holes with homology
2. Closing holes in discrete objects

ELSEVIER

Pattern Recognition Letters 23 (2002) 523–531

Pattern Recognition Letters

www.elsevier.com/locate/patrec

A three-dimensional holes closing algorithm

Zouina Aktouf, Gilles Bertrand, Laurent Perroton *

Pattern Recognition 43 (2010) 3548–3559

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

ELSEVIER

Hole filling in 3D volumetric objects

Marcin Janaszewski [a,*], Michel Couprie [b], Laurent Babout [a]

Pattern Recognition Letters 32 (2011) 2231–2238

Contents lists available at ScienceDirect

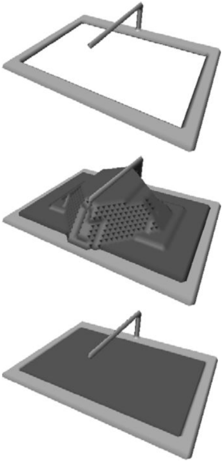
Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

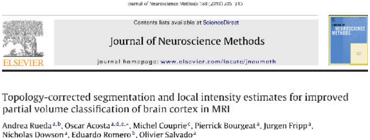
ELSEVIER

Robust algorithm for tunnel closing in 3D volumetric objects based on topological characteristics of points

Marcin Janaszewski [a,*], Michał Postolski [a,b], Laurent Babout [a]



A little bit of opening holes


Journal of Neuroscience Methods 188 (2010) 305–315

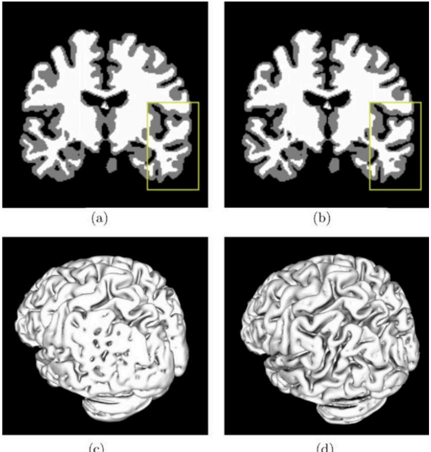
Contents lists available at ScienceDirect

Journal of Neuroscience Methods

journal homepage: www.elsevier.com/locate/jneumeth

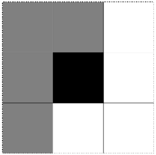
Topology-corrected segmentation and local intensity estimates for improved partial volume classification of brain cortex in MRI

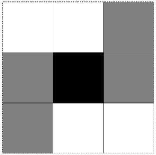
Andrea Rueda[a,b], Oscar Acosta[a,d,e,*], Michel Couprie[c], Pierrick Bourgeat[a], Jurgen Fripp[a], Nicholas Dowson[a], Eduardo Romero[b], Olivier Salvado[a]



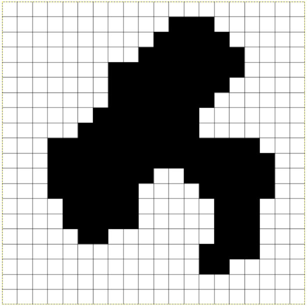
Structure

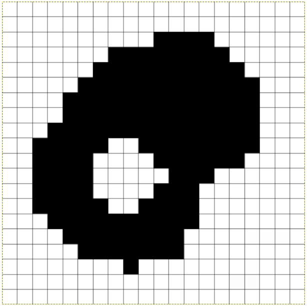
- Simple point



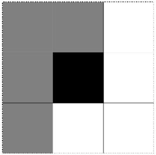


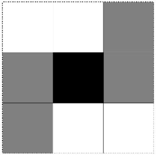
- Contractible object



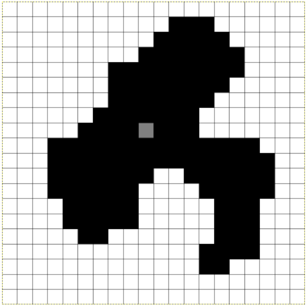


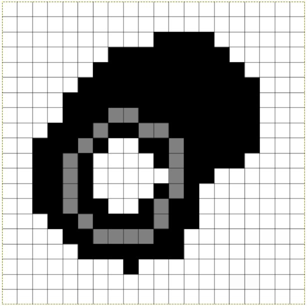
- Simple point





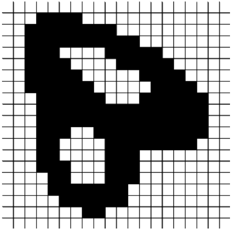
- Contractible object

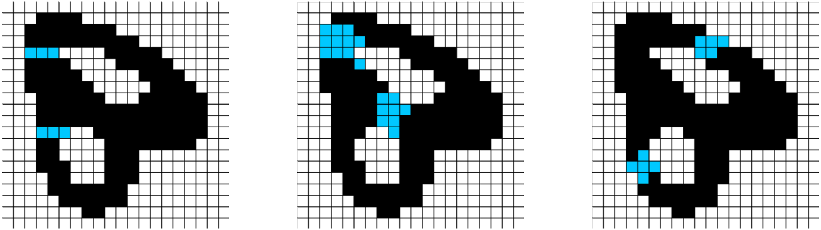




Homotopic opening

Let X be a discrete object.
$Y \subset X$ is a *homotopic opening* if $X - Y$ is contractible

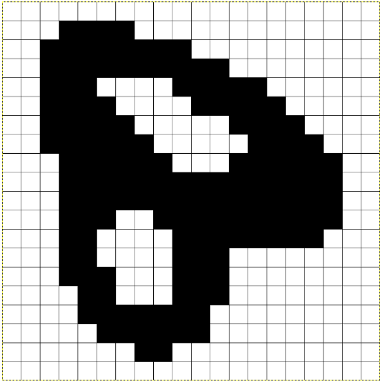




Structure

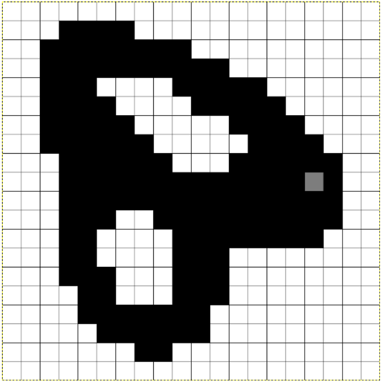
KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



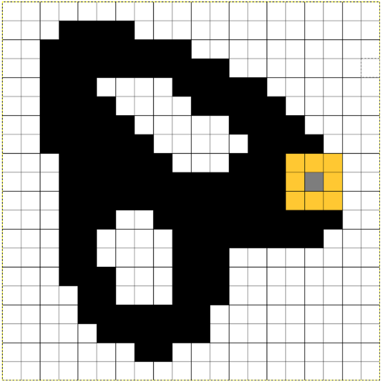
KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



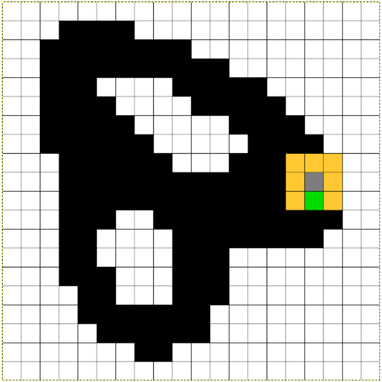
KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



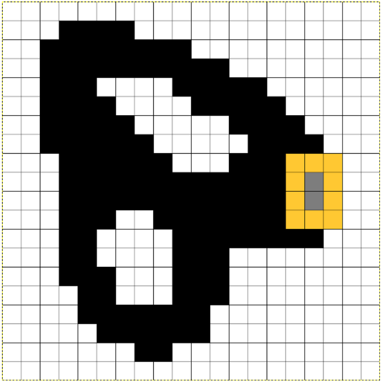
KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



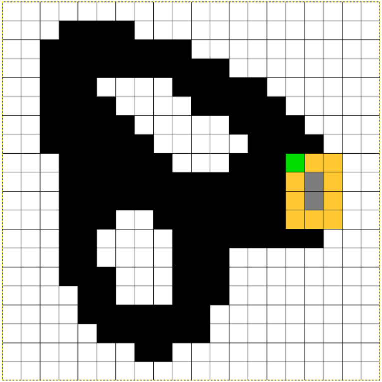
KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N_\alpha^*(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N_\alpha^*(x) \cap (X - C))$;
return $X - C$;



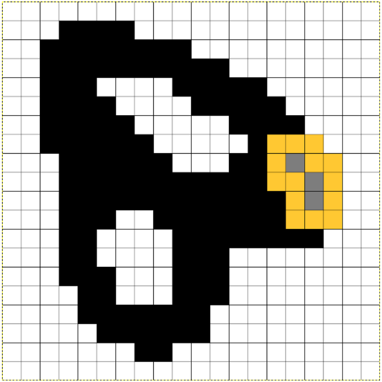
KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



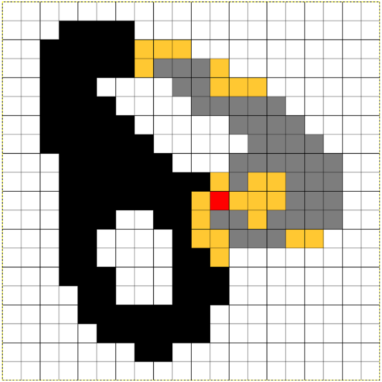
KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



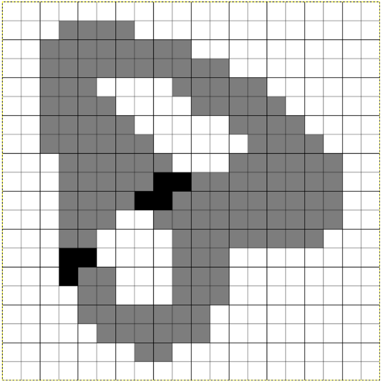
KISS algorithm

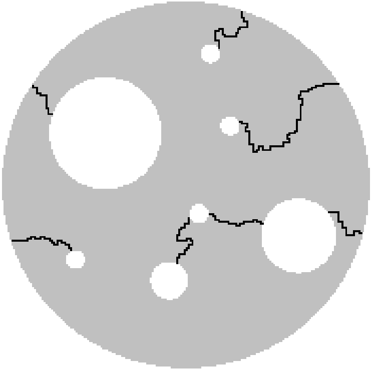
Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



KISS algorithm

Input: $X \subset \mathbb{Z}^d$
Output: Homotopic opening for X
$C \leftarrow \{x\}$, for $x \in X$;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ some point in S;
  $S \leftarrow S - \{x\}$;
  if x *is simple for* C **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

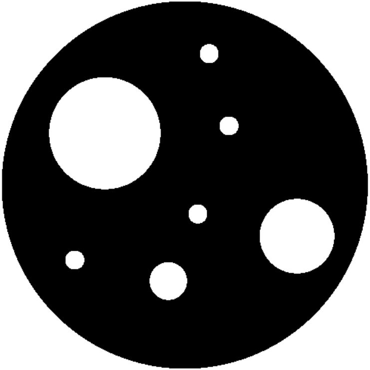




Better use the distance transform

Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$ such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

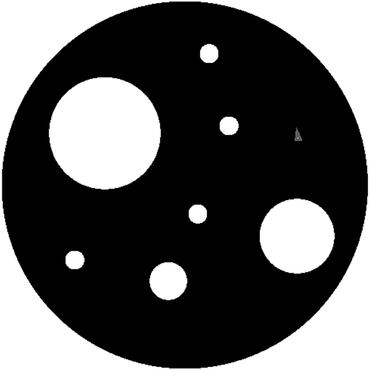


Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$ such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

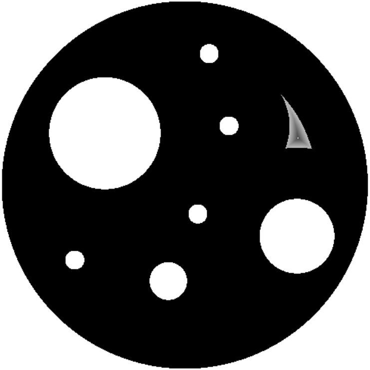


Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$
such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

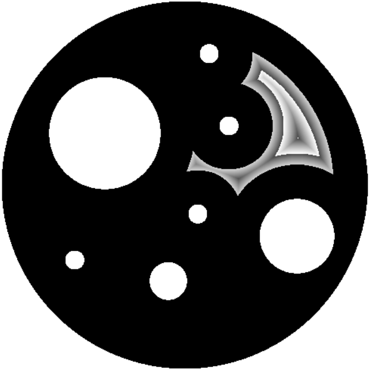


Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$ such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

Algorithm 1: random propagation

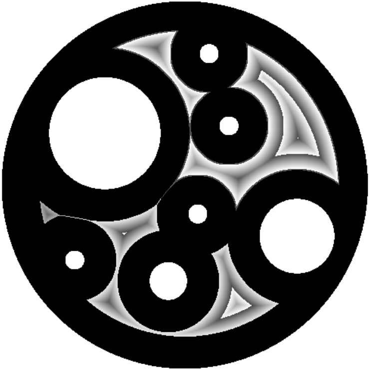
$C \leftarrow \{x\}$, for random $x \in X$
such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$
such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

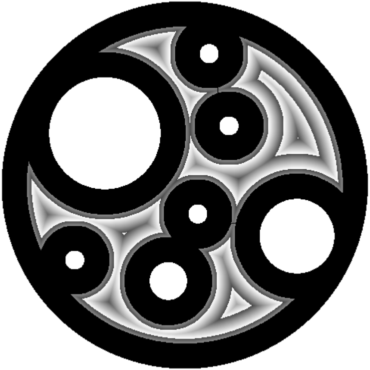


Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$
 such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such
   that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow$
     $S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

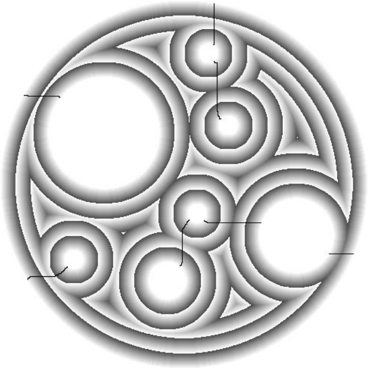


Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$
such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;

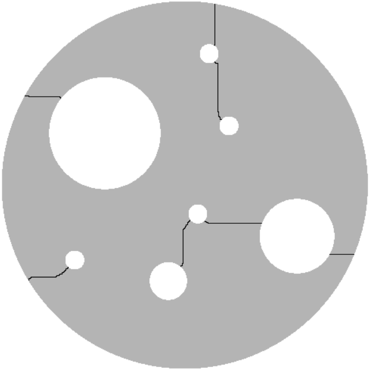


Complexity

dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$

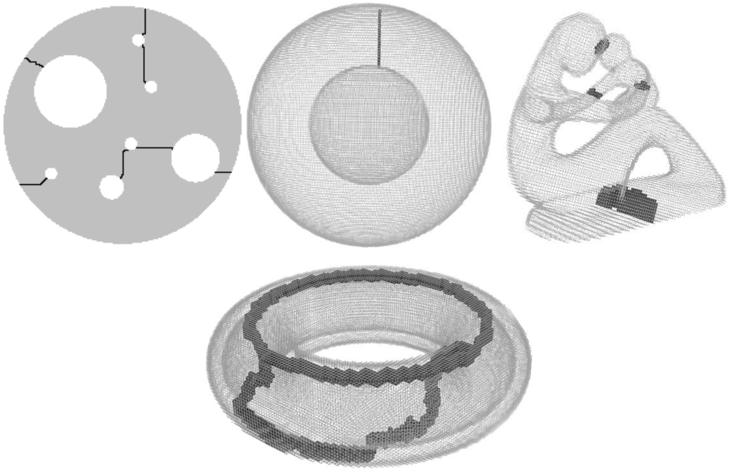
Algorithm 1: random propagation

$C \leftarrow \{x\}$, for random $x \in X$
 such that $dt_X(x)$ is maximal;
$S \leftarrow N^*_\alpha(x) \cap (X - C)$;
while $S \neq \emptyset$ **do**
  $x \leftarrow$ random point in S such
   that $dt_X(x)$ is maximal;
  $S \leftarrow S - \{x\}$;
  if *x is simple for C* **then**
    $C \leftarrow C \cup \{x\}$;
    $S \leftarrow$
     $S \cup (N^*_\alpha(x) \cap (X - C))$;
return $X - C$;



Complexity

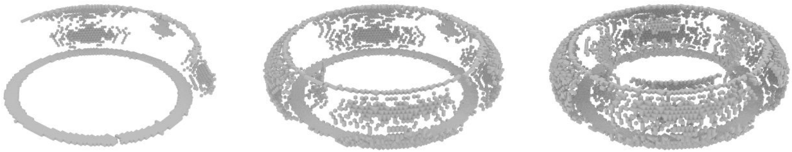
dD: $3^d n(\lg n + f(d))$
3D: $n \lg n$



Problems:

- Not straight lines → later
- Torus → next

What happens to the torus?

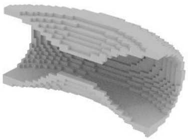


Too many points with equal distance

Solution: propagate by layers (Algorithm 2)

Algorithm 2: propagation by layers

$C \leftarrow \{x\}$, for random $x \in X$ with highest dt_X value;
repeat
  $m \leftarrow \max\{dt_X(x) \mid x \in N^*_\alpha(C) \cap X,$ x simple for $C\}$;
  $L \leftarrow dt_X^{-1}([m-1, m]) \cap N^*_\alpha(C)$;
  foreach $x \in L$ **do**
    if x *is simple for* C **then**
      $C \leftarrow C \cup \{x\}$;
until *idempotency*;
return $X - C$;

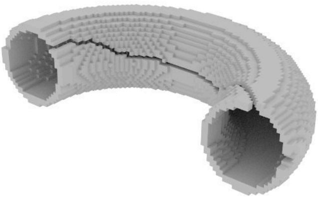


Complexity

dD: $3^d n^2 f(d)$
3D: n^2

Algorithm 2: propagation by layers

$C \leftarrow \{x\}$, for random $x \in X$ with highest dt_X value;
repeat
  $m \leftarrow \max\{dt_X(x) \mid x \in N^*_\alpha(C) \cap X, x$ simple for $C\}$;
  $L \leftarrow dt_X^{-1}([m-1, m]) \cap N^*_\alpha(C)$;
  foreach $x \in L$ **do**
    if *x is simple for C* **then**
      $C \leftarrow C \cup \{x\}$;
until *idempotency*;
return $X - C$;

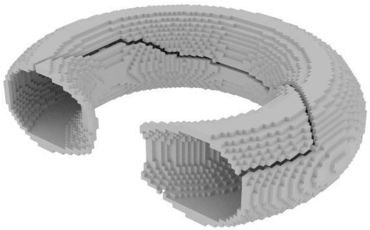


Complexity

dD: $3^d n^2 f(d)$
3D: n^2

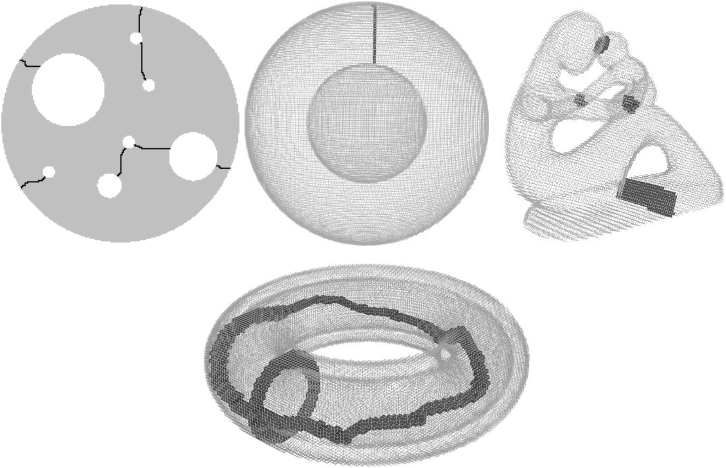
Algorithm 2: propagation by layers

$C \leftarrow \{x\}$, for random $x \in X$ with highest dt_X value;
repeat
  $m \leftarrow \max\{dt_X(x) \mid x \in N^*_\alpha(C) \cap X,\ x \text{ simple for } C\}$;
  $L \leftarrow dt_X^{-1}([m-1, m]) \cap N^*_\alpha(C)$;
  foreach $x \in L$ **do**
    if x *is simple for* C **then**
      $C \leftarrow C \cup \{x\}$;
until *idempotency*;
return $X - C$;



Complexity

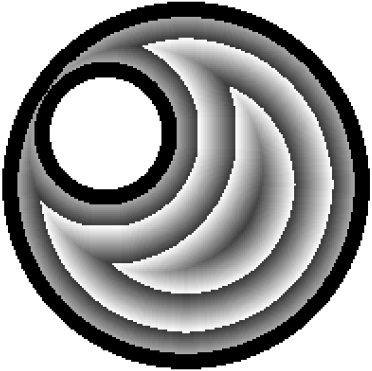
dD: $3^d n^2 f(d)$
3D: n^2



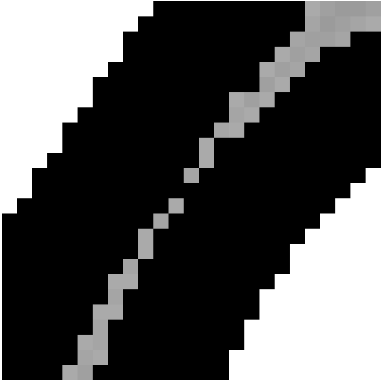
Problems:

- No straight lines → next
- ~~Torus~~

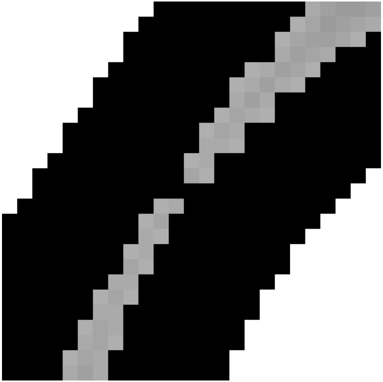
Why do we obtain those segments?



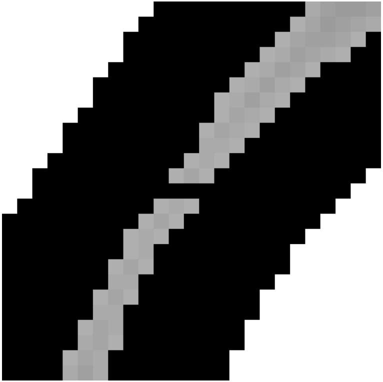
Why do we obtain those segments?



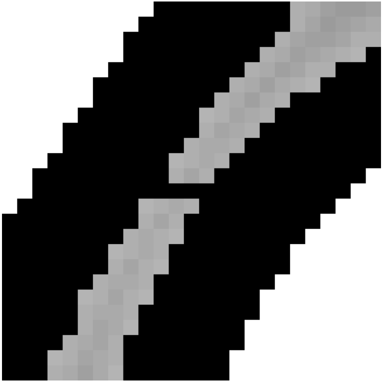
Why do we obtain those segments?



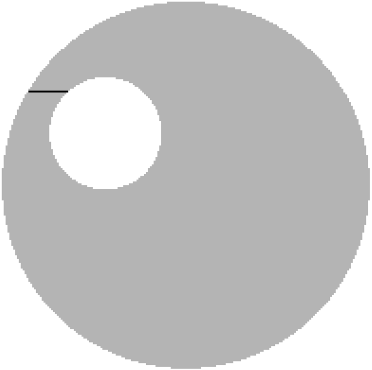
Why do we obtain those segments?



Why do we obtain those segments?



Why do we obtain those segments?



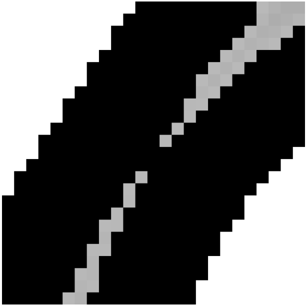
Solution: keep fronts separated (Plugin)

Plugin: simple balls

Input: $C \subset X$, $x \in x$, $r \geq 0$
Output: Can we add point x to C?

foreach $y \in N_\alpha(x)$ **do**
  if *$C \cup (B(y, r) \cap X)$ is collapsible to C* **then**
    return true;

return false;



Complexity

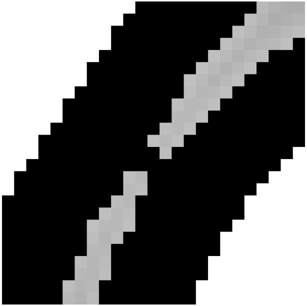
dD: $f(d) \rightarrow |N_\alpha| r^{2d} f(d)$
3D: $f(d) \rightarrow r^6$

Plugin: simple balls

Input: $C \subset X$, $x \in x$, $r \geq 0$
Output: Can we add point x to C?

foreach $y \in N_\alpha(x)$ **do**
  if $C \cup (B(y, r) \cap X)$ *is collapsible to* C **then**
    return true;

return false;



Complexity

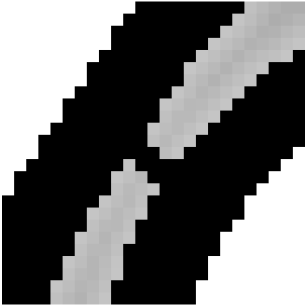
dD: $f(d) \to |N_\alpha| r^{2d} f(d)$
3D: $f(d) \to r^6$

Plugin: simple balls

Input: $C \subset X$, $x \in x$, $r \geq 0$
Output: Can we add point x to C?

foreach $y \in N_\alpha(x)$ **do**
  if $C \cup (B(y, r) \cap X)$ *is collapsible to* C **then**
    return true;

return false;



Complexity

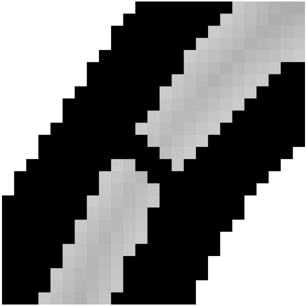
dD: $f(d) \rightarrow |N_\alpha| r^{2d} f(d)$
3D: $f(d) \rightarrow r^6$

Plugin: simple balls

Input: $C \subset X$, $x \in x$, $r \geq 0$
Output: Can we add point x to C?

foreach $y \in N_\alpha(x)$ **do**
  if $C \cup (B(y,r) \cap X)$ *is collapsible to* C **then**
    return true;

return false;

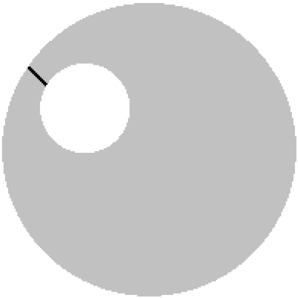


Complexity

dD: $f(d) \to |N_\alpha| r^{2d} f(d)$
3D: $f(d) \to r^6$

Plugin: simple balls

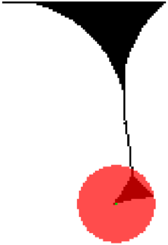
Input: $C \subset X$, $x \in x$, $r \geq 0$
Output: Can we add point x to C?
foreach $y \in N_\alpha(x)$ **do**
  if $C \cup (B(y,r) \cap X)$ *is collapsible to* C **then**
    return true;
return false;

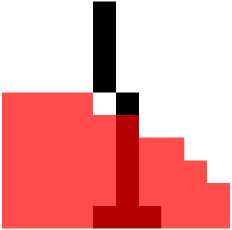


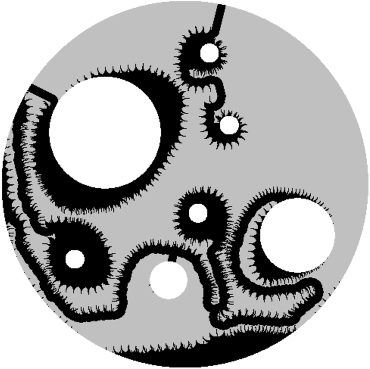
Complexity

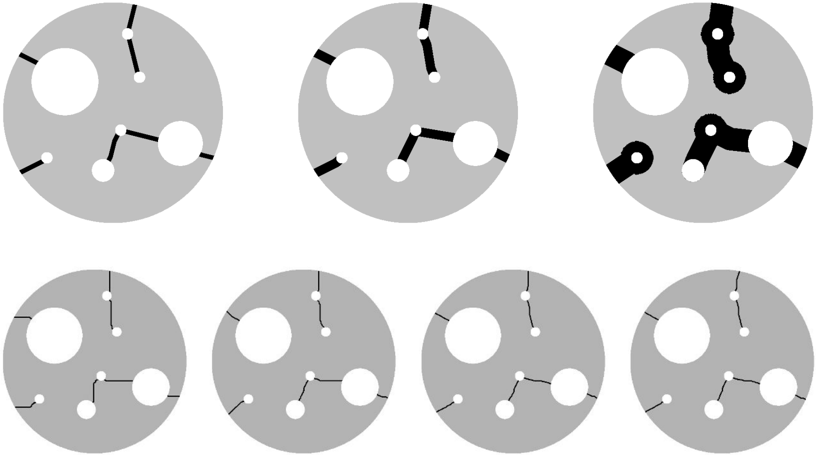
dD: $f(d) \to |N_\alpha| r^{2d} f(d)$
3D: $f(d) \to r^6$

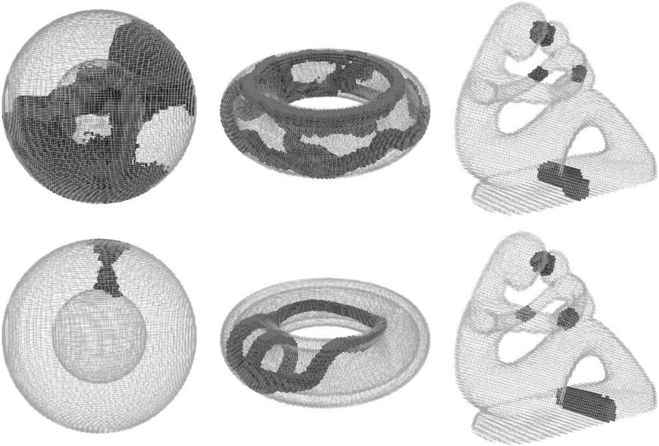
Why the neighborhood? The *tangency problem*











Problems:

- 3D isn't great $\rightarrow$ some day

Structure

- 2 algorithms + 1 parameter for ball radius
- Algorithm 1: faster ($n \lg n$) but worse output
- Algorithm 2: slower (n^2) but better output
- Study the *tangency problem* in 3D

- 2 algorithms + 1 parameter for ball radius
- Algorithm 1: faster ($n \lg n$) but worse output
- Algorithm 2: slower (n^2) but better output
- Study the *tangency problem* in 3D

Download this presentation:

`http://aldo.gonzalez-lorenzo.perso.luminy.univ-amu.fr/downloads.html`

Thank you for your attention