Discrete Morphology and Distances on graphs

Jean Cousty

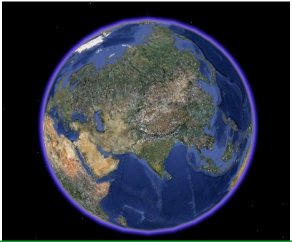
FOUR-DAY COURSE
on
Mathematical Morphology in image analysis
Bangalore 19-22 October 2010

Mathematical Morphology (MM) allows to process

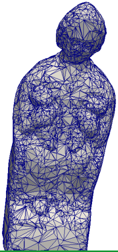
Continuous planes

Discrete grids

Continuous manifolds



Triangular meshes



Problem

- Is there generic structures that allow MM operators to be studied and implemented in computers?

Problem

- Is there generic structures that allow MM operators to be studied and implemented in computers?

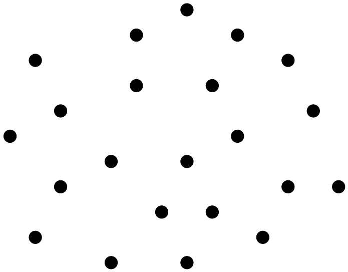
Proposition

- Graphs constitute such a structure for digital geometric objects

Outline

What is a graph ?

What is a graph ?

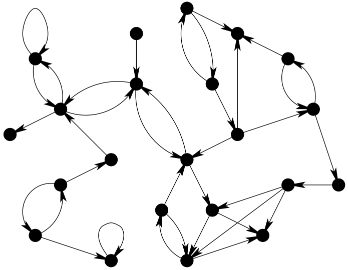


Definition

A graph G *is a pair* (V, E) *made of:*

- **A set** V
  whose elements $\{x \in V\}$ *are called* points *or* vertices *of* G

What is a graph ?

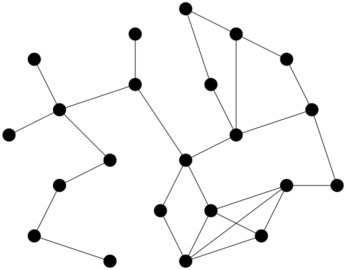


Definition

A graph G *is a pair* (V, E) *made of:*

- **A set** V
  whose elements $\{x \in V\}$ *are called* points *or* vertices *of* G
- **A binary relation** E **on** V *(i.e.,* $E \subseteq V \times V$*)*
  whose elements $\{(x, y) \in E\}$ *are called* edges *of* G

What is a graph ?



Definition

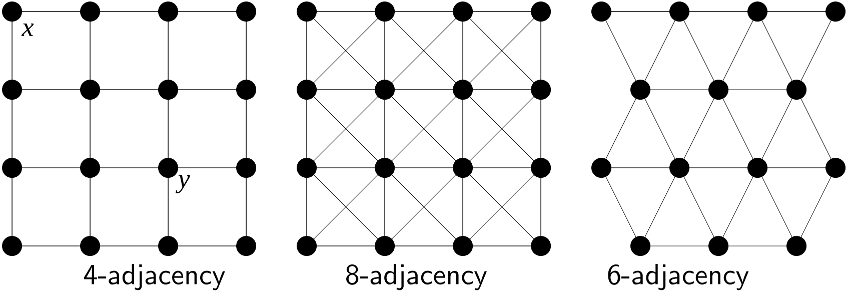
The graph (V, E) *is* symmetric *whenever:*

- $(x, y) \in E \implies (y, x) \in E$

The graph (V, E) *is* reflexive *if:*

- $\forall x \in V, (x, x) \in E$

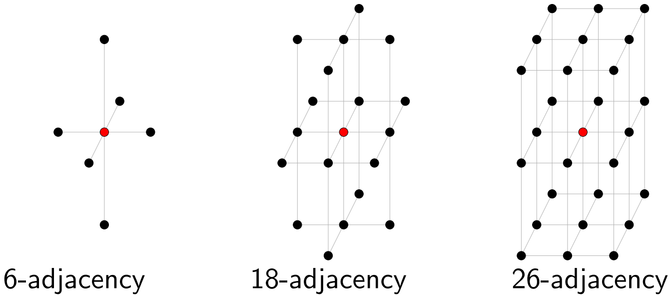
What is a graph ?



Symmetric & reflexive graph for 2D image analysis

- The vertex set V is the **image domain**
- The edge set E is given by an **“adjacency” relation**

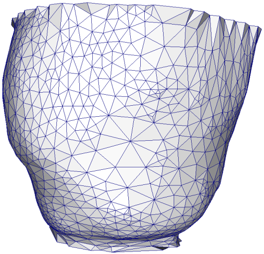
What is a graph ?

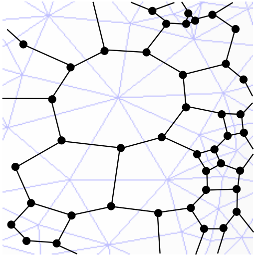


Symmetric & reflexive graph for 3D image analysis

- The vertex set V is the **image domain**
- The edge set E is given by an **“adjacency” relation**

What is a graph ?

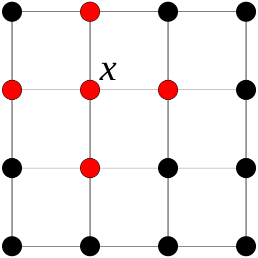




Symmetric & reflexive graph for mesh analysis

- The vertex set V is the **image domain**
- The edge set E is given by an **“adjacency” relation**

Neighborhood

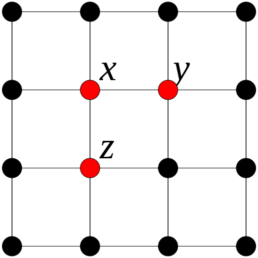


Definition

We call **neighborhood of a vertex (in G)** the set of all vertices linked (by an edge in G) to this vertex:

- $\forall x \in V, \Gamma(x) = \{y \in V \mid (x, y) \in E\}$

Neighborhood

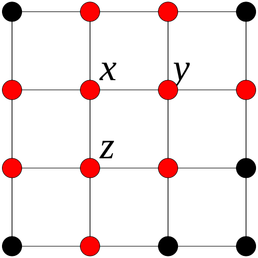


Definition

The **neighborhood (in G) of a subset of vertices**, is the union of the neighborhood of the vertices in this set:

- $\forall X \subseteq V, \Gamma(X) = \cup_{x \in X} \Gamma(x)$

Neighborhood



Definition

The **neighborhood (in G) of a subset of vertices**, is the union of the neighborhood of the vertices in this set:

- $\forall X \subseteq V, \Gamma(X) = \cup_{x \in X} \Gamma(x)$

Algebraic Dilation & graph

Property

- *Whatever the graph G, the map $\Gamma : \mathcal{P}(V) \rightarrow \mathcal{P}(V)$ is an (algebraic) dilation*
  - *Γ commutes with the supremum*

Morphological Dilation & graph

Property

- *If* V *is discrete and equipped with a translation* $\mathcal{T}$
- *If* X *and* B *are subsets of* V
- *Then,* $X \oplus B = \Gamma(X)$*, where* E *is made of all pairs* $(x, y) \in V \times V$ *such that* $y \in B_x$

Morphological Dilation & graph

Property

- *If V is discrete and equipped with a translation $\mathcal{T}$*
- *If X and B are subsets of V*
- *Then, $X \oplus B = \Gamma(X)$, where E is made of all pairs $(x, y) \in V \times V$ such that $y \in B_x$*

Conversely,

Property

- *If V is equipped with* **a translation $\mathcal{T}$**, *and an* **origin $o \in V$**
- *If G* **is translation invariant** *$(\forall x, y \in V, \Gamma(x) = \mathcal{T}_t(\Gamma(y)))$,*
- *Then, $\Gamma(X) = X \oplus B$, with $B = \Gamma(o)$*

Dilation, erosion, opening, closing & graph

Dilation, erosion, opening, closing & graph

Reminder

- The adjoint erosion of Γ:
  - obtained by duality
- Elementary openings and closings:
  - obtained by composition of adjoint dilations and erosion's

Dilation Algorithm

Algorithm

Input: A graph $G = (V, E)$ and a subset X of V

- $Y := \emptyset$
- **For each** $x \in V$ **do**
  - **if** $x \in X$ **do**
    - **For each** $y \in \Gamma(x)$ **do** $Y := Y \cup \{x\}$

Dilation Algorithm

Algorithm

Input: A graph $G = (V, E)$ and a subset X of V

- $Y := \emptyset$
- **For each** $x \in V$ **do**
  - **if** $x \in X$ **do**
    - **For each** $y \in \Gamma(x)$ **do** $Y := Y \cup \{x\}$

Data Structures

- Each element of V is represented by an integer between 0 and $|V| - 1$
- The map Γ is represented by an array of $|V|$ lists
- Sets X and Y are represented by Boolean arrays

Dilation Algorithm: Complexity analysis

Algorithm

Input: A graph $G = (V, E)$ and a subset X of V

- $Y := \emptyset$ — $O(1)$
- **For each** $x \in V$ **do** — $O(|V|)$
  - **if** $x \in X$ **do** — $O(|V|)$
    - **For each** $y \in \Gamma(x)$ **do** $Y := Y \cup \{x\}$ — $O(|V| + |E|)$

Data Structures

- Each element of V is represented by an integer between 0 and $|V| - 1$
- The map Γ is represented by an array of $|V|$ lists
- Sets X and Y are represented by Boolean arrays

Toward granulometries: iterated dilation

- Usual granulometric studies of X require
  - $\Gamma^N(X)$ for each possible value of N

Toward granulometries: iterated dilation

- Usual granulometric studies of X require
  - $\Gamma^N(X)$ for each possible value of N
- How can $\Gamma^N(X)$ be computed?

Toward granulometries: iterated dilation

- Usual granulometric studies of X require
  - $\Gamma^N(X)$ for each possible value of N
- How can $\Gamma^N(X)$ be computed?
- Applying N times the preceding algorithm?
  - Complexity $O(N \times (|V| + |E|))$

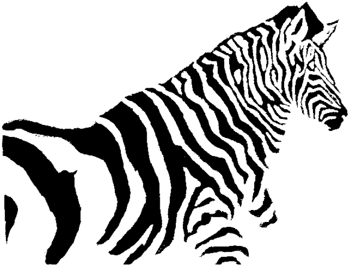
Toward granulometries: iterated dilation

- Usual granulometric studies of X require
  - $\Gamma^N(X)$ for each possible value of N
- How can $\Gamma^N(X)$ be computed?
- Applying N times the preceding algorithm?
  - Complexity $O(N \times (|V| + |E|))$

Problem

- Efficient computation of $\Gamma^N(X)$

Distance transforms: intuition



X (in black)

Distance transforms: intuition

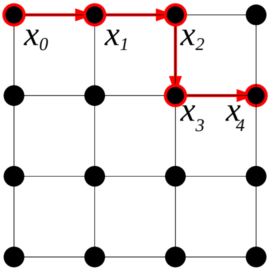
Distance transform of X

Distance transforms: intuition

./Figures/zebreDilation.avi
Thresholds: $\{\Gamma^N\}$

Paths

- Let $\pi = \langle x_0, \ldots, x_k \rangle$ be an ordered sequence of vertices
- π is a *path from x_0 to x_k* if:
  - any two consecutive vertices of π are linked by an edge: $\forall i \in [1, k]$, $(x_{i-1}, x_i) \in E$

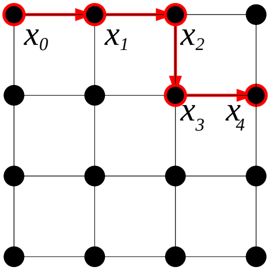


Length of a path

- Let $\pi = \langle x_0, \ldots, x_k \rangle$ be a path
- The *length of π*, denoted by $L(\pi)$, is the integer k

Length of a path

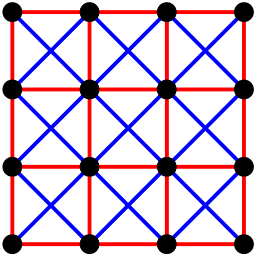
- Let $\pi = \langle x_0, \ldots, x_k \rangle$ be a path
- The *length of π*, denoted by $L(\pi)$, is the integer k



Path of length 4

Length of a path in a weighted graph

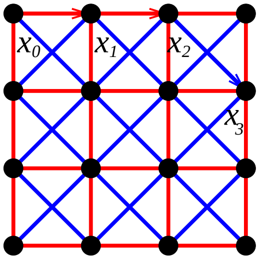
- Let ℓ be a map from E into $\mathbb{R}$: $u \rightarrow \ell(u)$, the *length* of the edge u
- The pair (G, ℓ) is called a *weighted graph* or a *network*



- Length of red edges: 1
- Length of blue edges: $\sqrt{2}$

Length of a path in a weighted graph

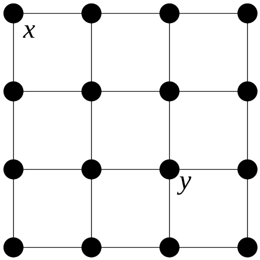
- Let ℓ be a map from E into $\mathbb{R}$: $u \to \ell(u)$, the *length* of the edge u
- The pair (G, ℓ) is called a *weighted graph* or a *network*
- The *length* of a path $\pi = \langle x_0, \ldots, x_k \rangle$ is the sum of the length of the edges along π: $L(\pi) = \sum_{i=1}^{k} \ell((x_{i-1}, x_i))$



- Length of red edges: 1
- Length of blue edges: $\sqrt{2}$
- Path of length $2 + \sqrt{2} \approx 3.4$

Graph distance

- Let x and y be two vertices
- The distance between x and y is defined by:
  - $D(x, y) = \min\{L(\pi) \mid \pi \text{ is a path from } x \text{ to } y\}$



- $D(x, y) = 4$

Graph distance

Property

- *If the graph G is symmetric, then the map D is a distance on V:*
  - $\forall x \in V,\ D(x, x) = 0$
  - $\forall x, y \in V,\ x \neq y \implies D(x, y) > 0$ *(positive)*
  - $\forall x, y \in V,\ D(x, y) = d(y, x)$ *(symmetric)*
  - $\forall x, y, z \in V,\ D(x, z) \leq D(x, y) + D(y, z)$ *(triangular inequality)*

Graph distance

Property

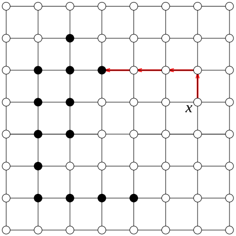
- *If the graph G is symmetric, then the map D is a distance on V:*
  - $\forall x \in V,\ D(x,x) = 0$
  - $\forall x, y \in V,\ x \neq y \implies D(x,y) > 0$ *(positive)*
  - $\forall x, y \in V,\ D(x,y) = d(y,x)$ *(symmetric)*
  - $\forall x, y, z \in V,\ D(x,z) \leq D(x,y) + D(y,z)$ *(triangular inequality)*

Terminology

- In this case of graph, the distance D is called *geodesic*

Distance Transform

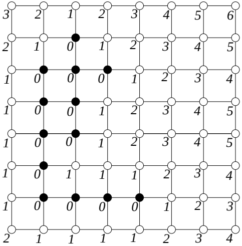
- Let $X \subseteq V$ and $x \in V$
- The distance between x and X is defined by
  - $D(x, X) = \min\{D(x, y) \mid y \in X\}$



- X black vertices
- $D(x, X)$

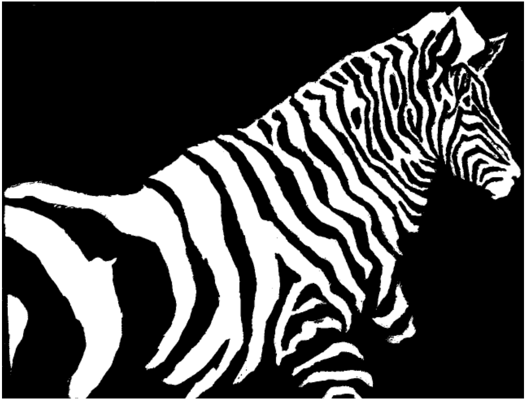
Distance Transform

- Let $X \subseteq V$ and $x \in V$
- The distance between x and X is defined by
  - $D(x, X) = \min\{D(x, y) \mid y \in X\}$
- The distance transform of X is the map from V into $\mathbb{R}$ defined by
  - $x \rightarrow D_X(x) = D(x, X)$



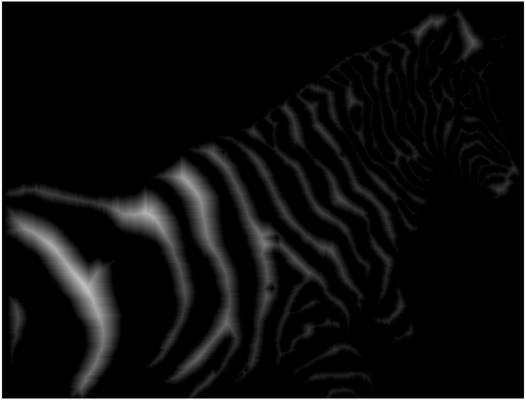
- X black vertices
- D_X

Illustration on an image



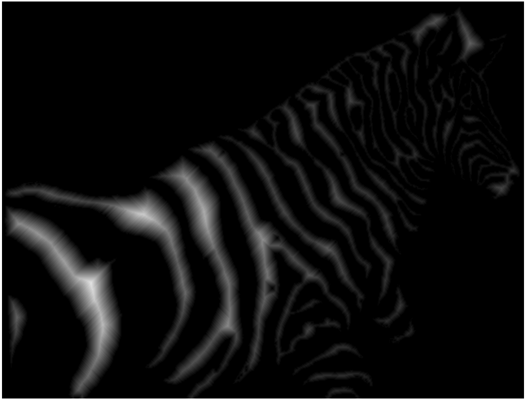
Original object X in black

Illustration on an image



D_X in the (non-weighted) graph induced by the 4-adjacency

Illustration on an image



D_X in the (non-weighted) graph induced by the 8-adjacency

Distance transforms & dilations (in non-weighted graphs)

- The *level-set of D_X at level k* ($X \subseteq V, k \in \mathbb{R}$) is defined by:
  - $D_X[k] = \{x \in V \mid D_X(x) \leq k\}$

Distance transforms & dilations (in non-weighted graphs)

- The *level-set of D_X at level k* ($X \subseteq V, k \in \mathbb{R}$) is defined by:
  - $D_X[k] = \{x \in V \mid D_X(x) \leq k\}$

Theorem

- $\Gamma^k(X) = D_X[k]$, *for any* $X \subseteq V$ *and any* $k \in \mathbb{N}$

Computing distance transforms (in non-weighted graphs)

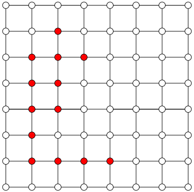
Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

Computing distance transforms (in non-weighted graphs)

Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

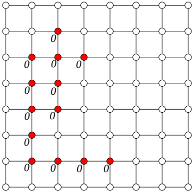


- **Example of execution**
- **S in red**
- **T in blue**
- $k = 0$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X;\ T := \emptyset;\ k := 0;$
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T;\ T := \emptyset;\ k := k + 1;$

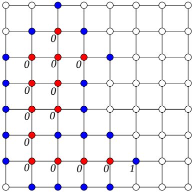


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 0$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

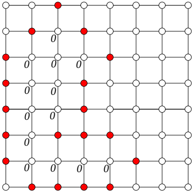


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 0$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X;\ T := \emptyset;\ k := 0;$
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T;\ T := \emptyset;\ k := k + 1;$

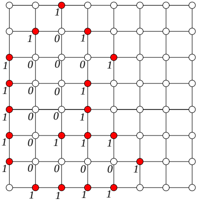


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 1$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

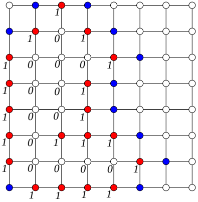


- **Example of execution**
- **S in red**
- **T in blue**
- $k = 1$

Computing distance transforms (in non-weighted graphs)

Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

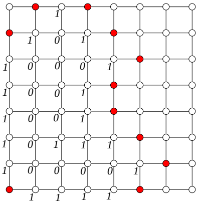


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 1$

Computing distance transforms (in non-weighted graphs)

Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

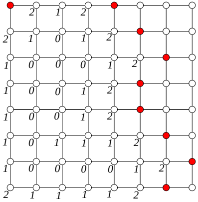


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 2$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
    - **For each** $x \in S$ **do** $D_X := k$
    - **For each** $x \in S$ **do**
        - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
    - $S := T$; $T := \emptyset$; $k := k + 1$;

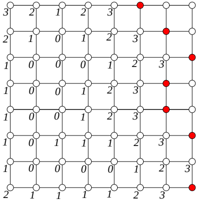


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 3$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

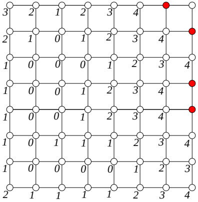


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 4$

Computing distance transforms (in non-weighted graphs)

Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

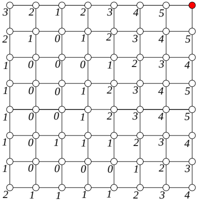


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 5$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X;\ T := \emptyset;\ k := 0;$
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\};$
  - $S := T;\ T := \emptyset;\ k := k + 1;$

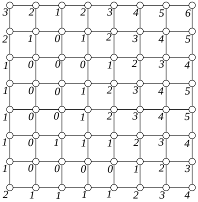


- **Example of execution**
- S **in red**
- T **in blue**
- $k = 6$

Computing distance transforms (in non-weighted graphs)

Algorithm: Input: $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X;\ T := \emptyset;\ k := 0;$
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T;\ T := \emptyset;\ k := k + 1;$



- **Example of execution**
- S **in red**
- T **in blue**
- $k = 7$

Computing distance transforms (in non-weighted graphs)

Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

Correctness: sketch of the proof by induction

- At the end of step k, $D_X(y) = k$ *if and only if* there is a path of length k from X to y

Computing distance transforms (in non-weighted graphs)

Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$;
  - $S := T$; $T := \emptyset$; $k := k + 1$;

Data Structures

- Elements of V represented by integers in $[0, |V| - 1]$
- Γ represented by an array of $|V|$ lists
- S and T implemented as lists

Computing distance transforms (in non-weighted graphs)

Algorithm: **Input:** $X \subseteq V$, **Results:** D_X

- **For each** $x \in V$ **do** $D_X(x) := \infty$
- $S := X$; $T := \emptyset$; $k := 0$;
- **While** $S \neq \emptyset$ **do**
  - **For each** $x \in S$ **do** $D_X := k$
  - **For each** $x \in S$ **do**
    - **For each** $y \in \Gamma(x)$ **if** $D_X(y) = \infty$ **do** $T := T \cup \{y\}$; $D_X(y) := -\infty$
  - $S := T$; $T := \emptyset$; $k := k + 1$;

Complexity

- $O(|V| + |E|)$

Computing distance transforms in weighted graphs

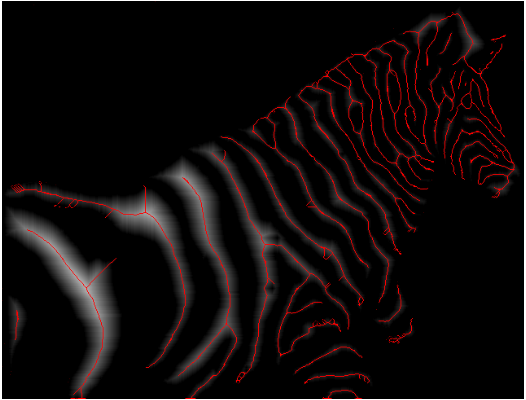
- Disjkstra Algorithm (1959)
- Complexity (using modern data structure)
  - Same as sorting algorithms

Computing distance transforms in weighted graphs

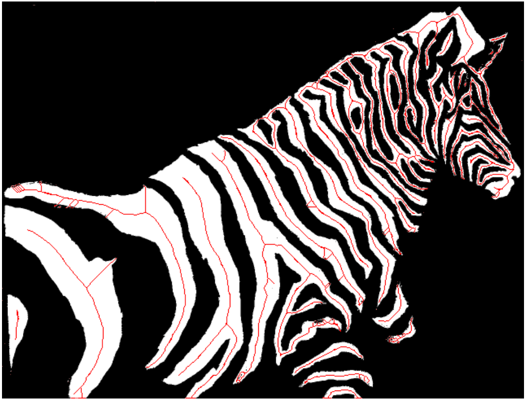
- Disjkstra Algorithm (1959)
- Complexity (using modern data structure)
  - Same as sorting algorithms
  - For small integers distances: $O(|V| + |E|)$

Computing distance transforms in weighted graphs

- Disjkstra Algorithm (1959)
- Complexity (using modern data structure)
  - Same as sorting algorithms
  - For small integers distances: $O(|V| + |E|)$
  - For floating point numbers distances: $O(\log\log(|V|) + |E|)$



- Visually, the salient loci of the DT form a “centered skeleton”



- Visually, the salient loci of the DT form a "centered skeleton"
- **Medial axis** constitute a first notion of such skeletons
  - Introduced by Blum in the 60's

Medial Axis: grass fire analogy

./Figures/feudeprairie.avi

Maximal balls

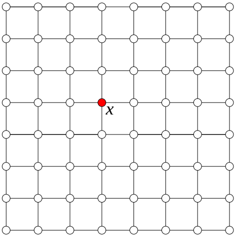
Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*

Maximal balls

Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*

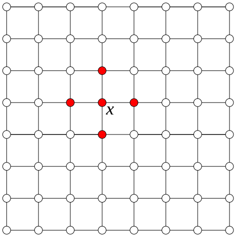


- $\Gamma^0(x)$

Maximal balls

Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*

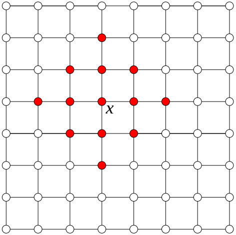


- $\Gamma^1(x)$

Maximal balls

Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*

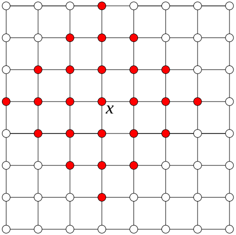


- $\Gamma^2(x)$

Maximal balls

Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*



- $\Gamma^3(x)$

Maximal balls

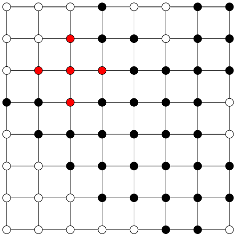
Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*
- $\Gamma^r(x)$ is called a *maximal ball in X* if:
  - $\Gamma^r(x) \subseteq X$
  - $\forall y \in V$, $\forall r' \in \mathbb{N}$, if $\Gamma^r(x) \subseteq \Gamma^{r'}(y) \subseteq X$, then $\Gamma^r(x) = \Gamma^{r'}(y)$

Maximal balls

Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*
- $\Gamma^r(x)$ is called a *maximal ball in X* if:
  - $\Gamma^r(x) \subseteq X$
  - $\forall y \in V$, $\forall r' \in \mathbb{N}$, if $\Gamma^r(x) \subseteq \Gamma^{r'}(y) \subseteq X$, then $\Gamma^r(x) = \Gamma^{r'}(y)$

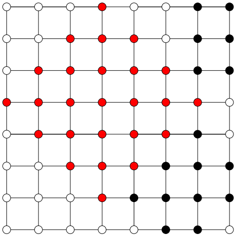


- X in red and black
- **A ball which is not maximal in X**

Maximal balls

Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*
- $\Gamma^r(x)$ is called a *maximal ball in X* if:
  - $\Gamma^r(x) \subseteq X$
  - $\forall y \in V, \forall r' \in \mathbb{N}$, if $\Gamma^r(x) \subseteq \Gamma^{r'}(y) \subseteq X$, then $\Gamma^r(x) = \Gamma^{r'}(y)$

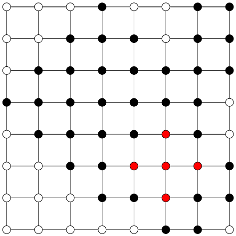


- X in red and black
- **A maximal ball**

Maximal balls

Definition

- $\Gamma^r(x)$ is called the *ball of radius r centered on x*
- $\Gamma^r(x)$ is called a *maximal ball in X* if:
  - $\Gamma^r(x) \subseteq X$
  - $\forall y \in V, \forall r' \in \mathbb{N}$, if $\Gamma^r(x) \subseteq \Gamma^{r'}(y) \subseteq X$, then $\Gamma^r(x) = \Gamma^{r'}(y)$



- X in red and black
- **A maximal ball**

Medial Axis

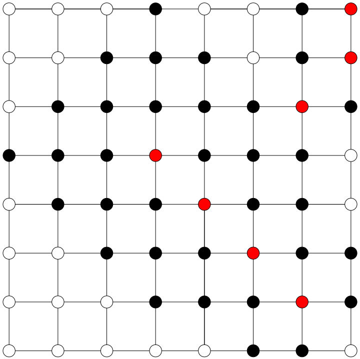
Definition

- The *medial axis of X* is the set of centers of maximal balls in X
  - $MA(X) = \{x \in X \mid \exists r \in \mathbb{N}, \Gamma^r(x)$ is a maximal ball in $X\}$

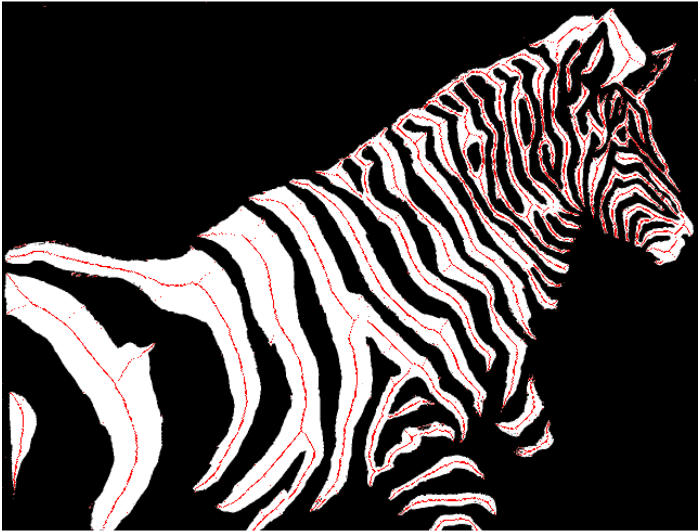
Medial Axis

Definition

- The *medial axis of X* is the set of centers of maximal balls in X
  - $MA(X) = \{x \in X \mid \exists r \in \mathbb{N}, \Gamma^r(x) \text{ is a maximal ball in } X\}$



Medial Axis: illustration on images



Example of application: Virtual Coloscopy

./Figures/ct.avi

Example of application: Virtual Coloscopy

./Figures/segmentation.avi

Example of application: Virtual Coloscopy

./Figures/paths.avi

Example of application: Virtual Coloscopy

./Figures/colono.avi

Computational characterization

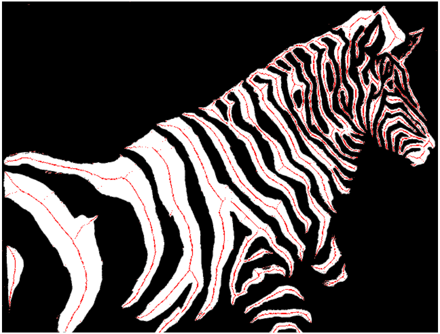
- The point $x \in V$ is a local maximum of D_X if
  - for any $y \in \Gamma(x)$, $D_X(y) \leq D_X(y)$

Property

- *The medial axis of X is the set of local maxima of $D_{\overline{X}}$*

Homotopic transform

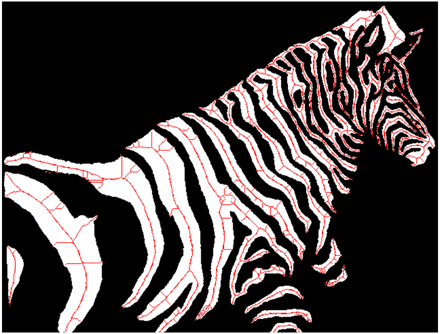
- Medial axis of connected objects can be disconnected



Medial Axis

Homotopic transform

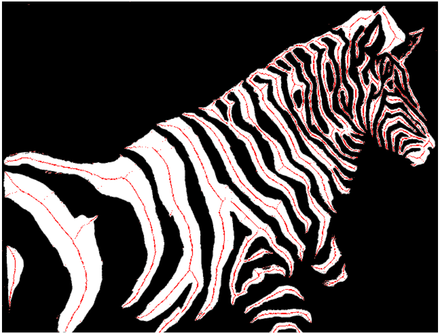
- Medial axis of connected objects can be disconnected



Homotopic skeleton

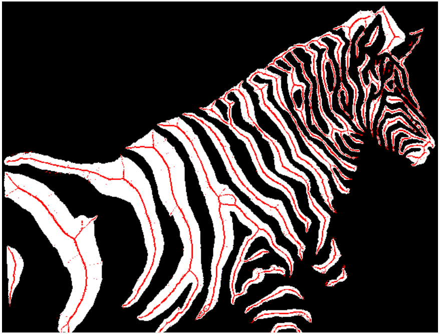
- Kong & Rosenfeld. *Digital topology: introduction and survey* CVGIP-89
- Couprie and Bertrand, *New characterizations of simple points in 2D, 3D and 4D discrete spaces*, TPAMI-09

Euclidean distance and medial axis



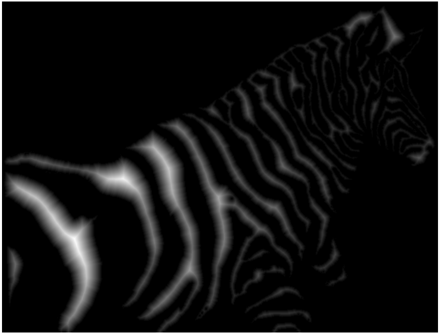
Medial axis for the D_4 graph distance

Euclidean distance and medial axis



Medial axis for the Euclidean distance

Euclidean distance and medial axis



Euclidean distance transform

- Saito & Toriwaki, *New algorithms for Euclidean distance transformation of an n-dimensional digitized picture with applications*, PR-94
- Remy & Thiel, *Exact Medial Axis with Euclidean Distance* IVC-05

Opening function

Opening function

Opening function

Figures/OpeningFunction.png

Opening function

Figures/OpeningFunction.png

- Vincent, *Fast grayscale granulometry algorithms*, ISMM'94
- Chaussard et al., *Opening functions in linear time for chessboard and city-bloc distances* (in preparation)

Summary

- Introduction of the graph formalism for MM
- Distance Transform
- Linear time algorithm for morphological operators in graphs
- Medial axis