Image Foresting Transform

A graph-based approach to image processing

Alexandre Xavier Falcão

afalcao@ic.unicamp.br

www.ic.unicamp.br/~afalcao/ift.html.

Institute of Computing

State University of Campinas

Campinas - SP - Brazil

Contributors

- Jorge Stolfi, Institute of Computing, State University of Campinas, Campinas, SP, Brazil.
- Roberto Lotufo, Faculty of Electrical and Computer Engineering, State University of Campinas, Campinas, SP, Brazil.
- Luciano Costa, Institute of Physics, University of Sao Paulo, Sao Carlos, SP, Brazil.
- Fernando Cendes, Dept. of Neurology, Faculty of Medical Sciences, University of Campinas, Campinas, SP, Brazil.
- Bruno Cunha, Ricardo Torres, Felipe Bergo, Gabriela Castellano, Romaric Audigier, Celso Suzuki, Eduardo Picado, and Carlos Camolesi.

Goals of research

The design, efficient implementation, and evaluation of effective image processing operators based on connectivity.

Motivation

- Unification
  Image operators can be derived from a single algorithm, favoring hardware-based implementations and the understanding of how operators relate to each other.
- Efficiency
  The IFT algorithm usually runs in linear time. Further optimizations are possible for specific applications.
- Simplicity
  An image operator requires a simple choice of parameters of the IFT algorithm and a local processing of its output.

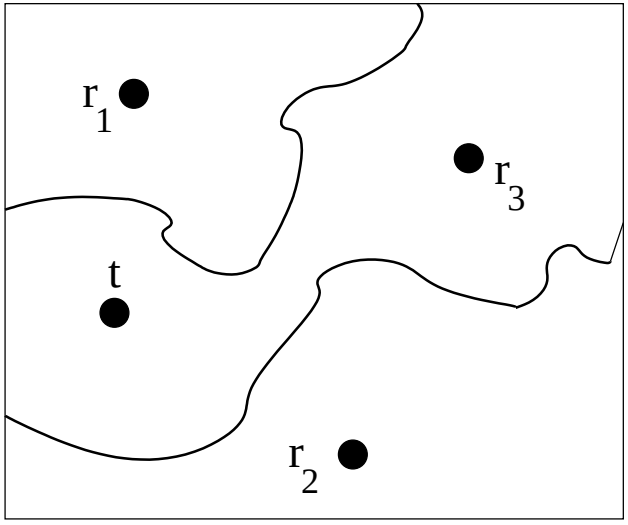
Purpose of the lecture

What may you expect from this lecture?

- Get a different way of looking at image processing problems.
- Understand the IFT and how it works for several applications.
- Make it easier to use the C source code of the IFT algorithm, which is available for download at www.ic.unicamp.br/~afalcao/ift.html.

What is it all about?

Many image operators can be directly/indirectly related to a partition of the image into influence zones associated with root pixels, where the zone of each root consists of the pixels that are more closely connected to that root than to any other, in some appropriate sense.

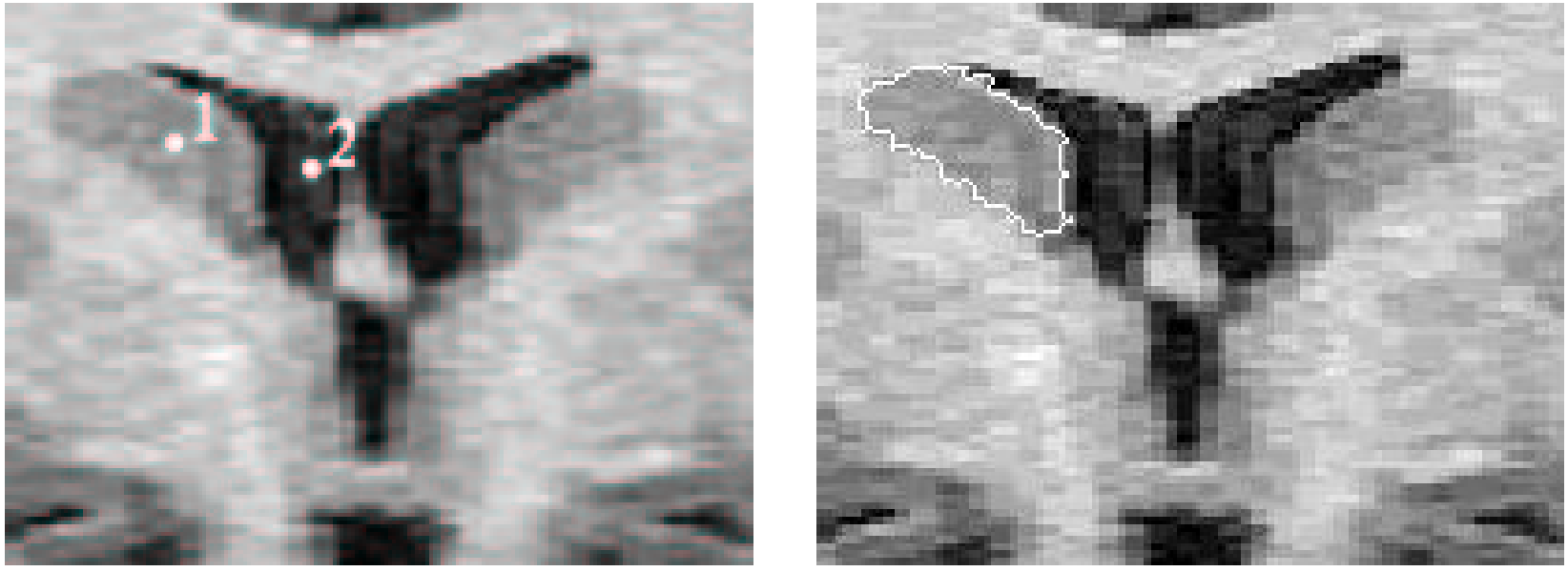


Pixel t is more closely connected to root r_3.

What is it all about?

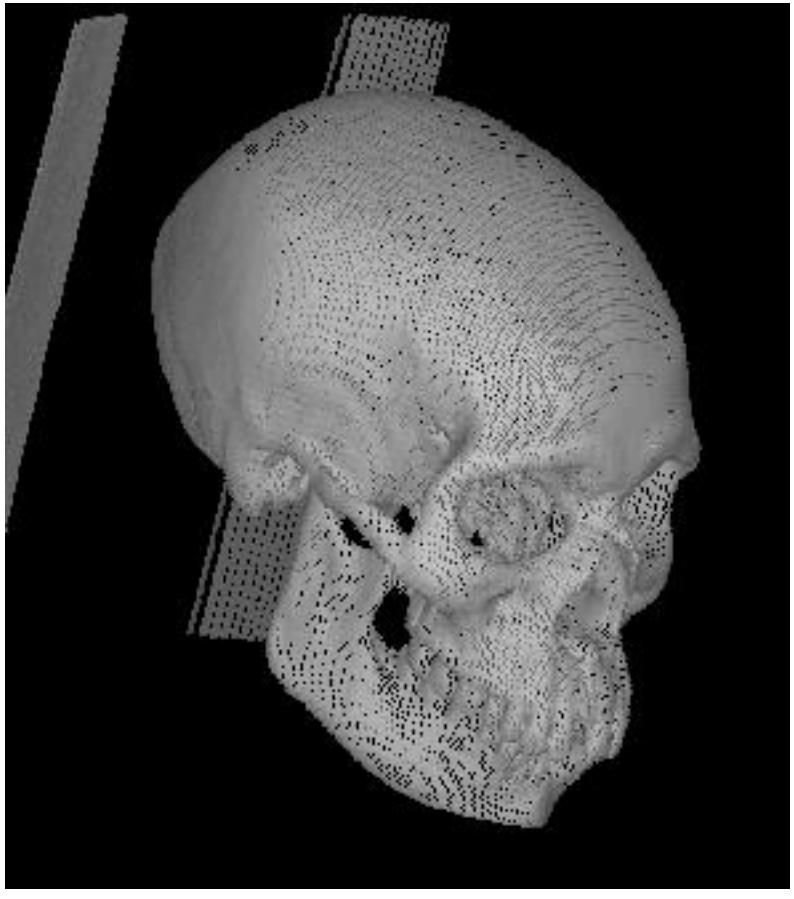
- The IFT unifies these image operators by computing an optimum-path forest in a directed graph derived from the image.
- The roots of the forest are drawn from a given set of seed pixels, which may also consist of all image pixels.
- The influence zone of a root r consists of the pixels reached from r by a path of minimum cost, considering the cost of all paths in the graph from the seed set to that pixels.
- The IFT algorithm outputs three attributes for each image pixel: an optimum path from the root set, the cost of that path, and its corresponding root.
- An image operator is reduced to a simple local processing of these attributes.

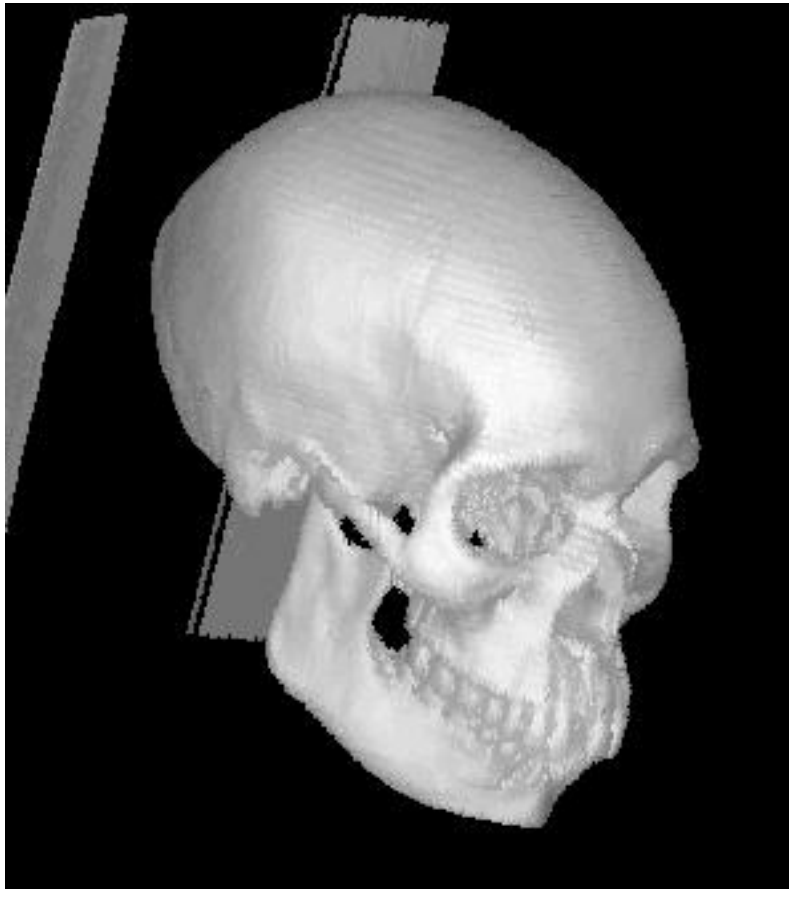
Region-based image segmentation



An object can be defined by pixels whose root belongs to a given set of internal pixels.

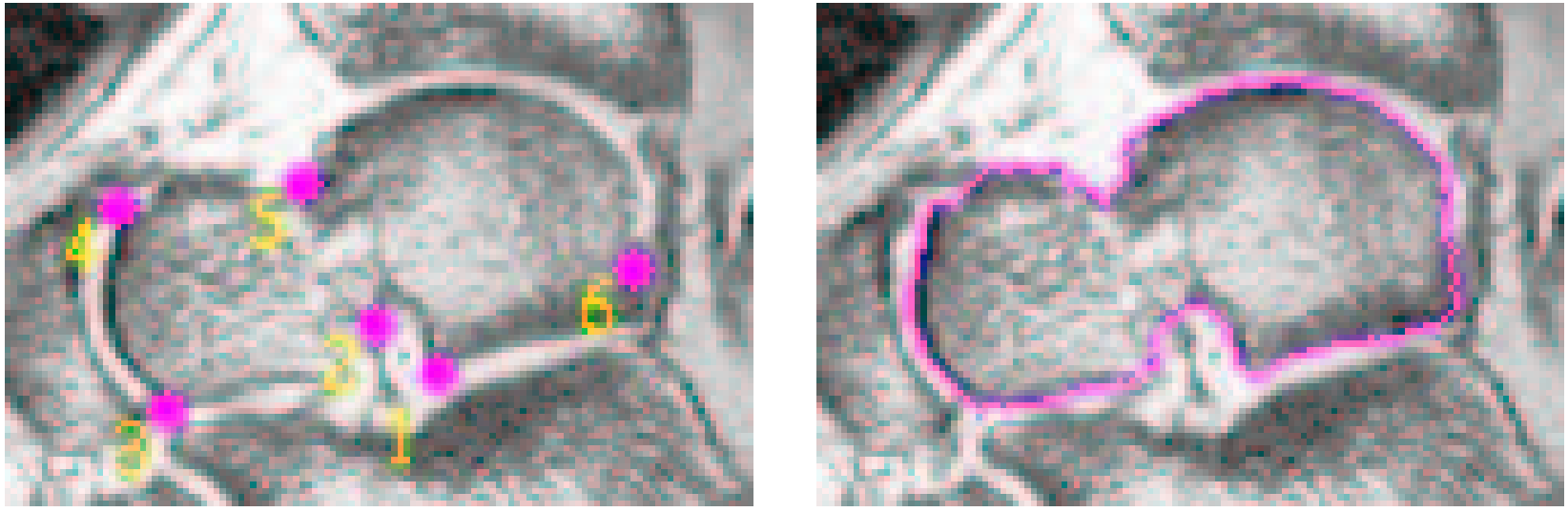
Image filtering





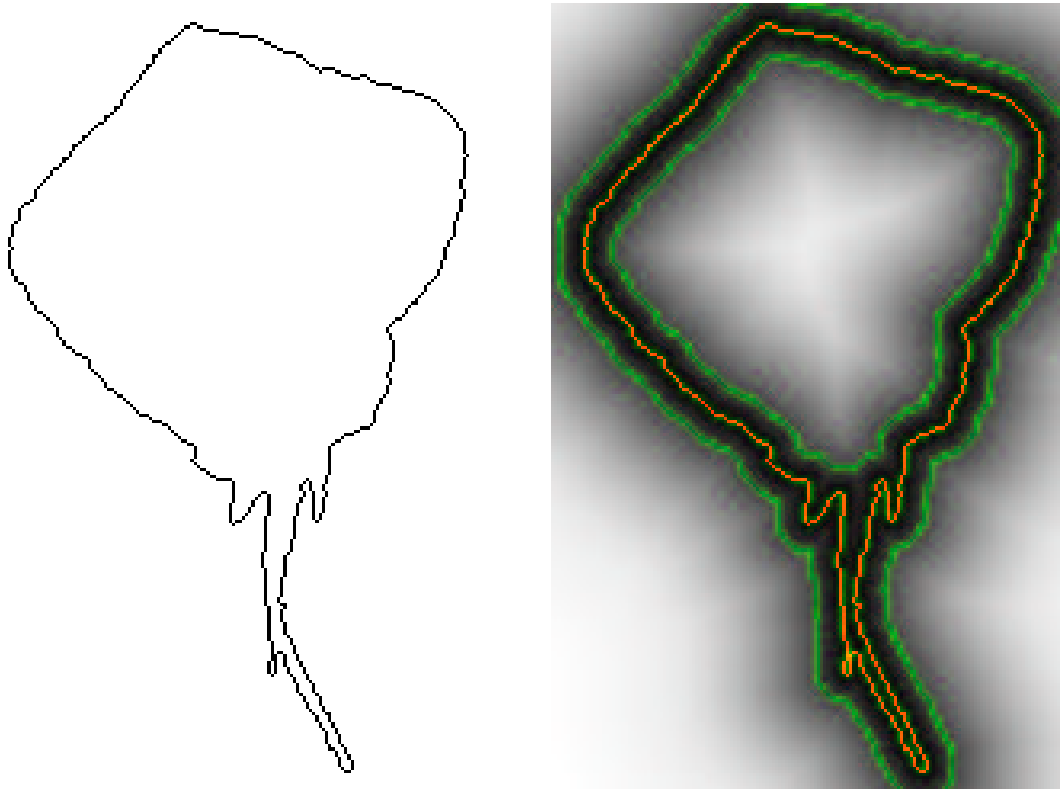
A filtered image can be obtained from the cost attribute.

Optimum boundary tracking



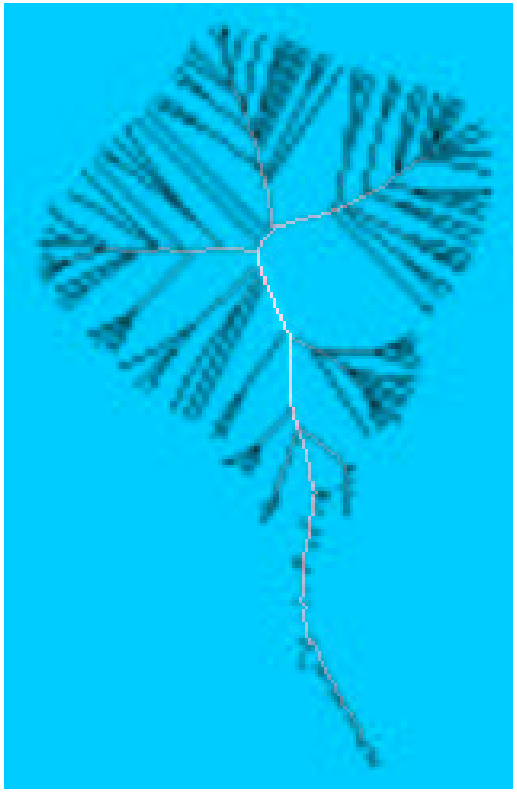
The path attribute can be used to find an optimum curve that is constrained to pass through a given sequence of landmarks (pixel sets) on the object's boundary.

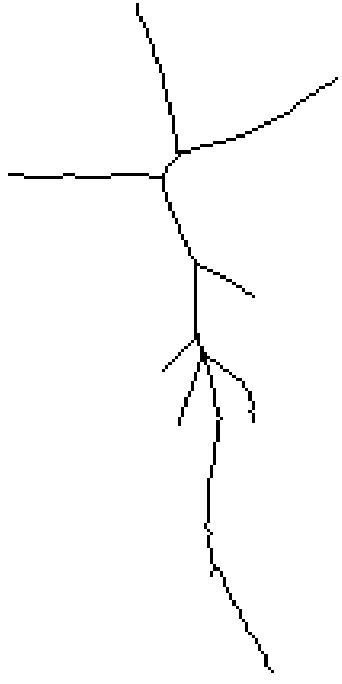
Euclidean distance transform (EDT)



The EDT of a given shape and its exact dilations/erosions can be obtained from the cost attribute.

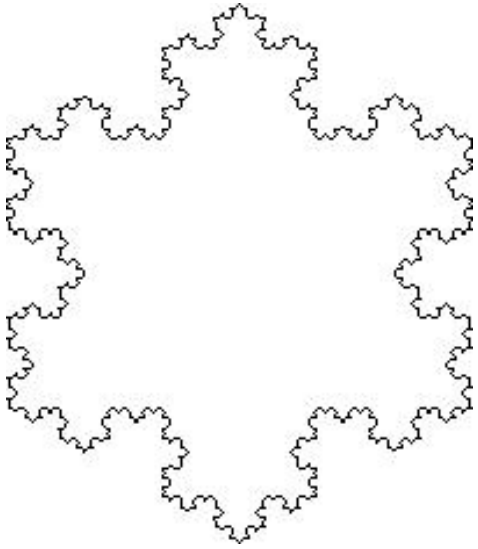
Multiscale skeletonization

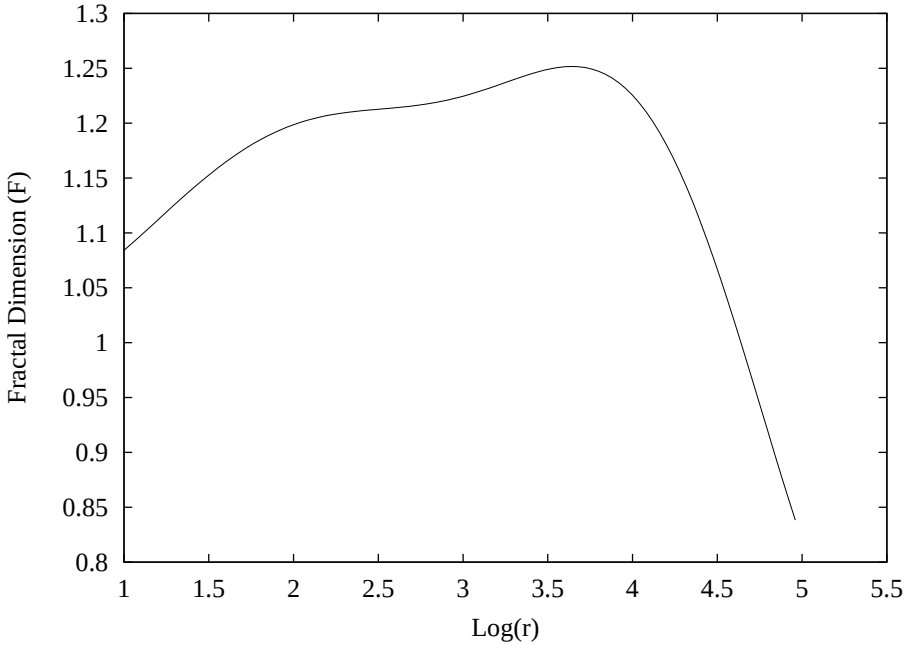




MS-skeletons can be obtained from the root attribute and thresholded into one-pixel-wide and connected skeletons.

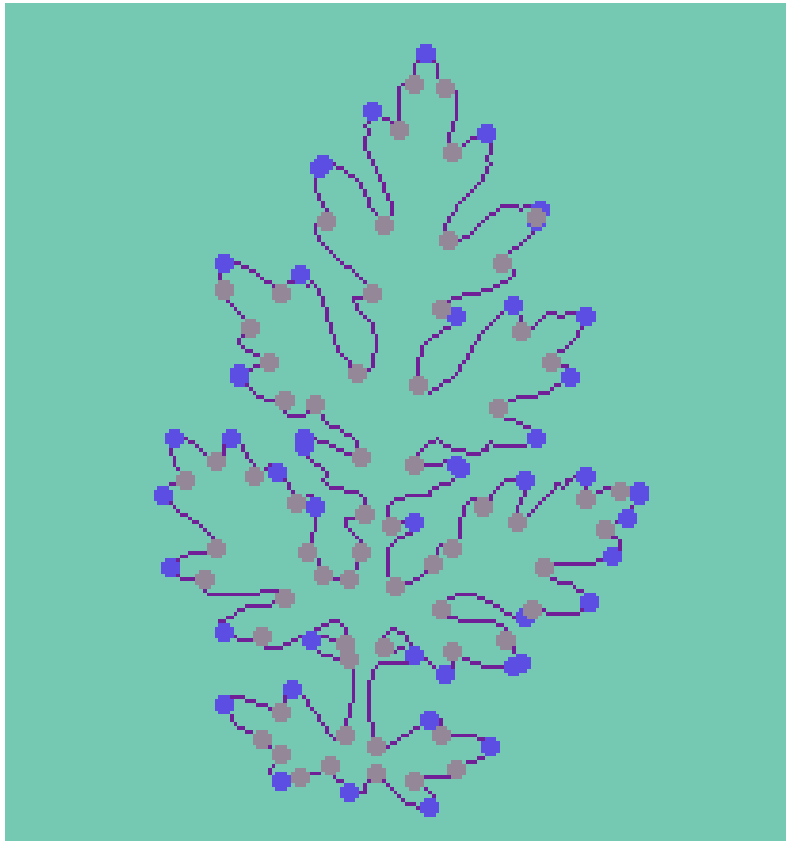
Multiscale fractal dimension





The self-similarity of a shape can be expressed in various scales from the cost attribute.

Salience points of a shape



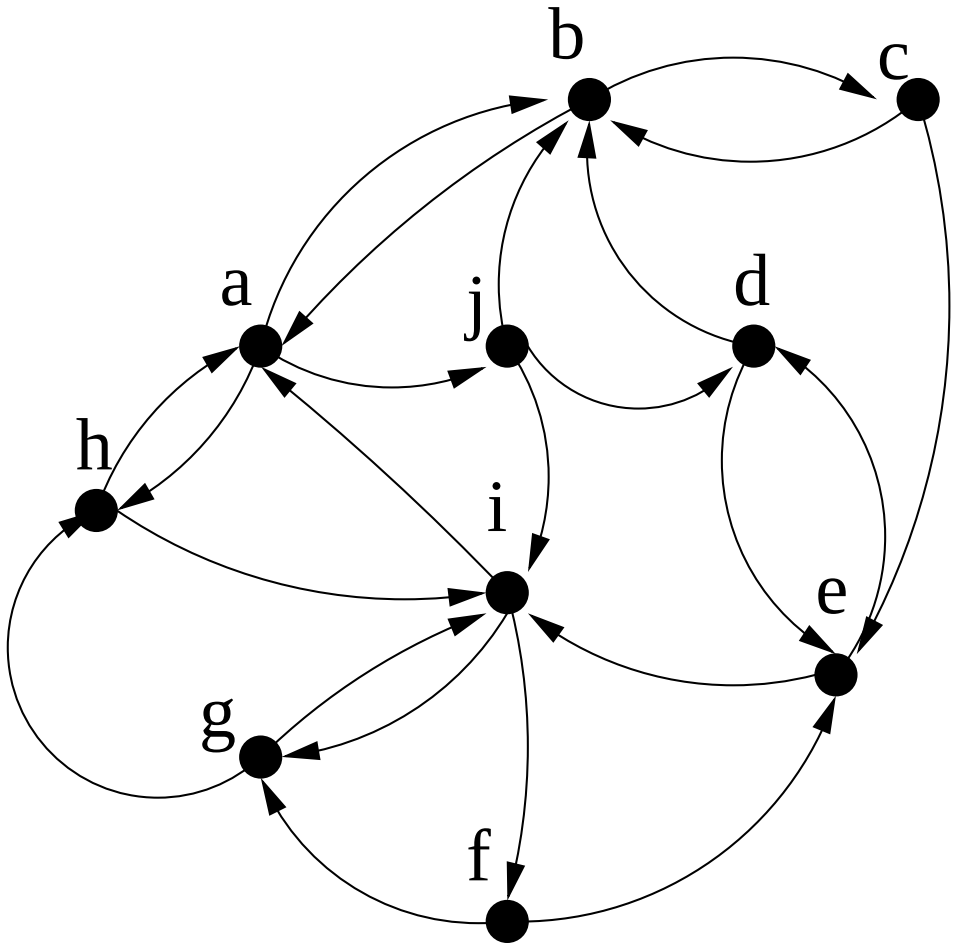
Higher curvature points of a shape can be located from the root attribute.

Outline

- Graph concepts
- Images as graphs
- Image Foresting Transform
- The IFT algorithm
- Applications
- Current work

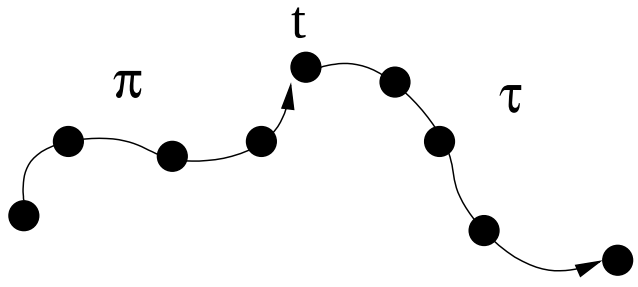
Directed graphs

A directed graph is a pair $(\mathcal{I}, \mathcal{A})$, where $\mathcal{I}$ is a set of nodes and $\mathcal{A}$ is a set of ordered pairs of nodes.



Paths

- A path is a sequence $\langle t_1, t_2, \ldots, t_k \rangle$ of distinct nodes in the graph, such that $(t_i, t_{i+1}) \in \mathcal{A}$ for $1 \leq i \leq k-1$.
- A path is trivial if $k = 1$.
- Path $\pi \cdot \tau$ denotes the concatenation of two paths, π and τ, where π ends at t and τ begins at t.



- Path $\pi = \tau \cdot \langle s, t \rangle$ denotes the concatenation of the longest prefix τ of π and the last arc (s, t).

Path-cost functions

- A path-cost function is a mapping that assigns to each path π a cost $f(\pi)$, in some ordered set $\mathcal{V}$ of cost values.
- A function f is said monotonic-incremental (MI) when

$$\begin{aligned} f(\langle t \rangle) &= h(t), \\ f(\tau \cdot \langle s,t \rangle) &= f(\tau) \odot (s,t), \end{aligned}$$

where $h(t)$ is a handicap cost value and $\odot$ satisfies: $x' \geq x \Rightarrow x' \odot (s,t) \geq x \odot (s,t)$ and $x \odot (s,t) \geq x$, for $x, x' \in \mathcal{V}$ and $(s,t) \in \mathcal{A}$.

Examples of MI cost functions

- Additive cost function

$$\begin{aligned} f_{sum}(\langle t \rangle) &= h(t), \\ f_{sum}(\pi \cdot \langle s,t \rangle) &= f_{sum}(\pi) + w(s,t), \end{aligned}$$

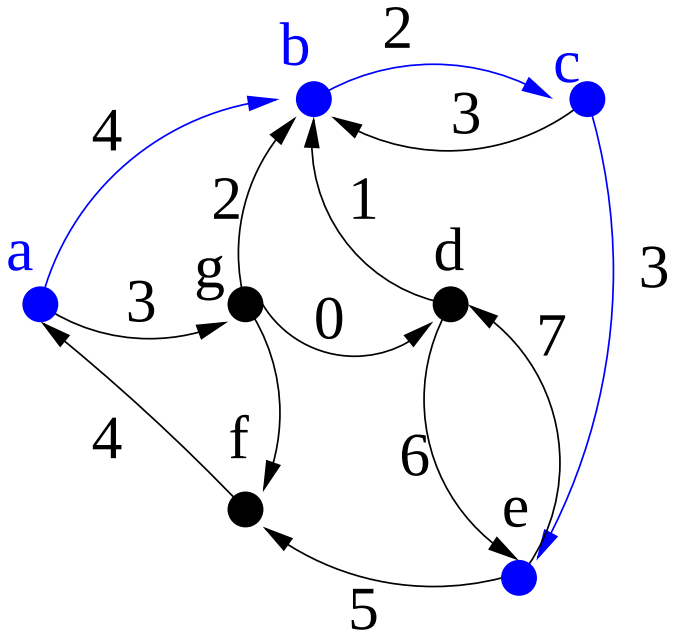
where $w(s,t)$ is a fixed non-negative arc weight.

- Max-arc cost function

$$\begin{aligned} f_{max}(\langle t \rangle) &= h(t), \\ f_{max}(\pi \cdot \langle s,t \rangle) &= \max\{f_{max}(\pi), w(s,t)\}, \end{aligned}$$

where $w(s,t)$ is a fixed arc weight.

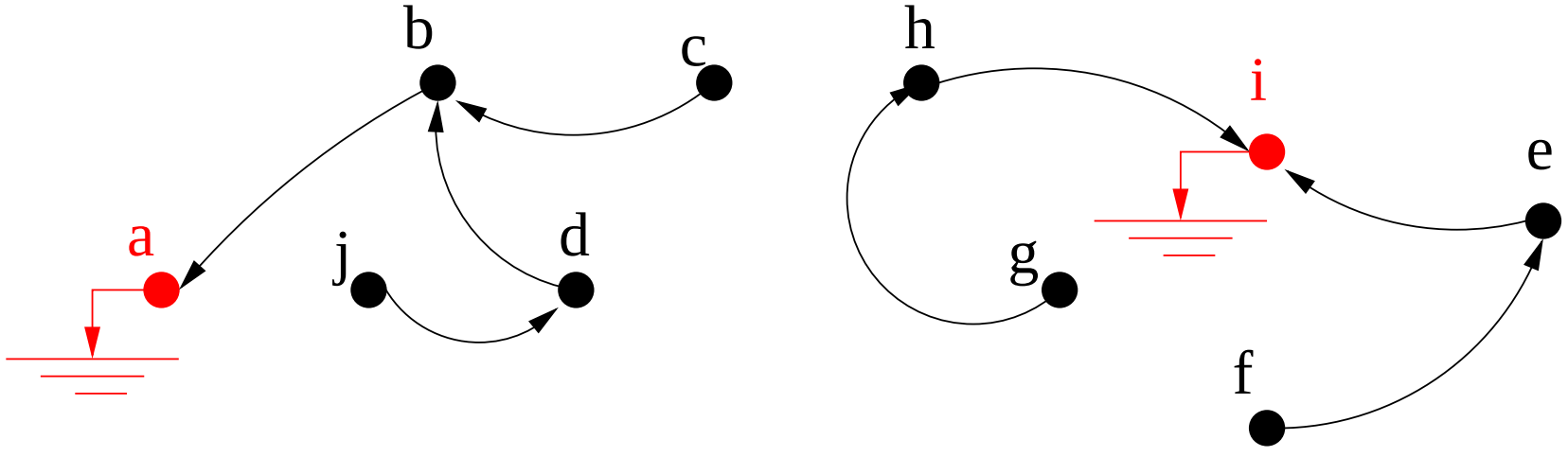
Examples of MI cost functions



1. $h(a) = 1$, $f_{max}(\langle a, b, c, e \rangle) = 4$ and $f_{sum}(\langle a, b, c, e \rangle) = 10$.
2. $h(a) = 5$, $f_{max}(\langle a, b, c, e \rangle) = 5$ and $f_{sum}(\langle a, b, c, e \rangle) = 14$.

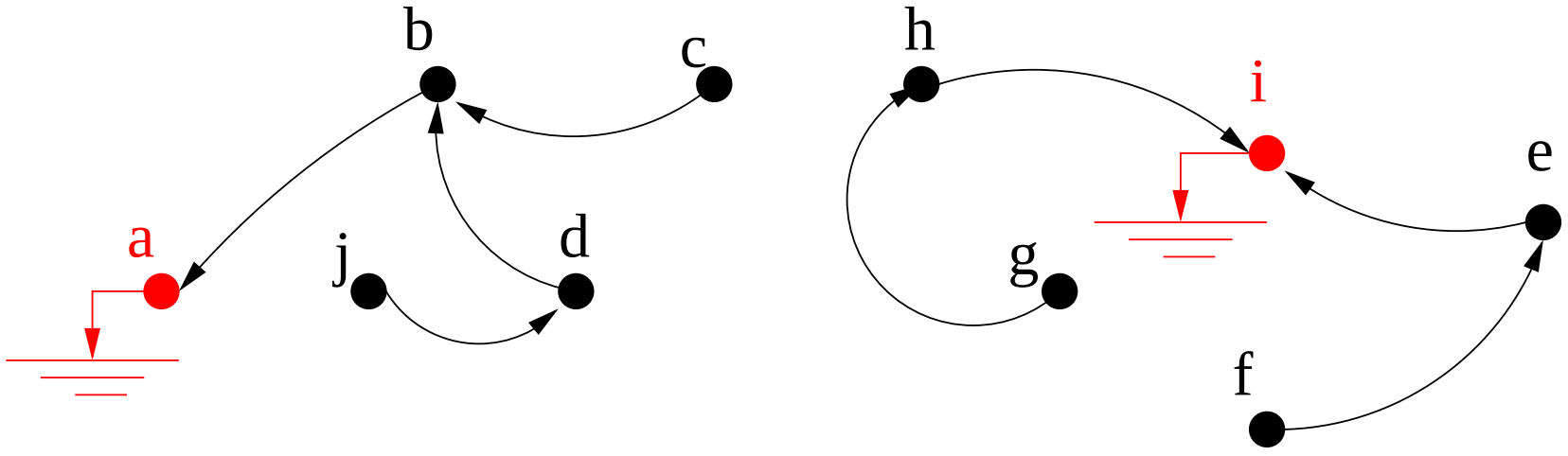
Predecessor map and spanning forest

- A predecessor map is a function P that assigns to each node $t \in \mathcal{I}$ either some other node in $\mathcal{I}$, or a distinctive marker $nil \notin \mathcal{I}$— in which case t is the root of the map.
- A spanning forest is a predecessor map which takes every node to nil in a finite number of iterations (i.e. it contains no cycles).



Paths of the forest P

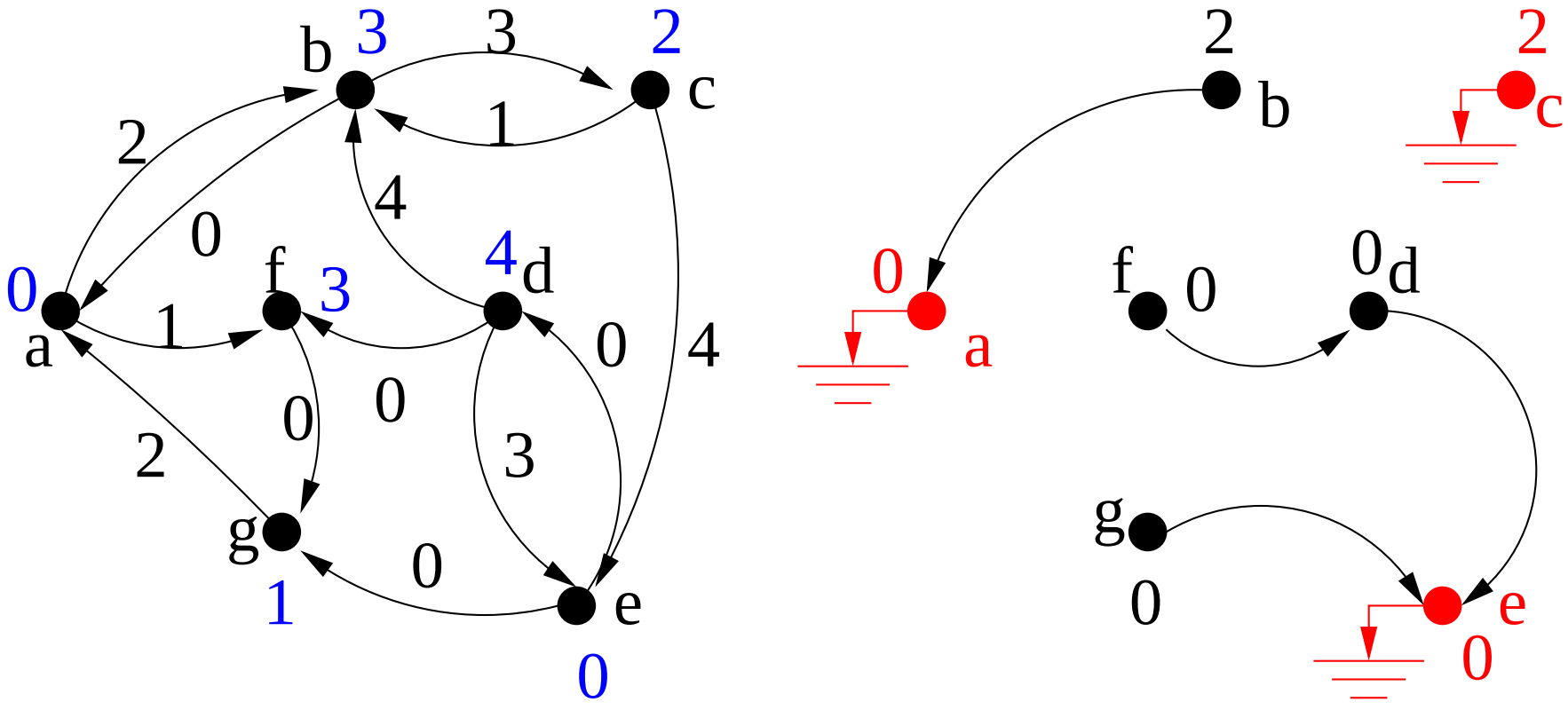
For any node $t \in \mathcal{I}$, there is a path $P^*(t)$ which is obtained in backward by following the predecessor nodes along the path.



For example, path $P^*(c) = \langle a, b, c \rangle$, where $P(c) = b$, $P(b) = a$, and $P(a) = nil$; and path $P^*(i) = \langle i \rangle$, where $P(i) = nil$.

Optimum-path forest

An optimum-path forest is a spanning forest P, where $f(P^*(t))$ is minimum for all nodes $t \in \mathcal{I}$. Consider cost function f_{sum} in the example below.



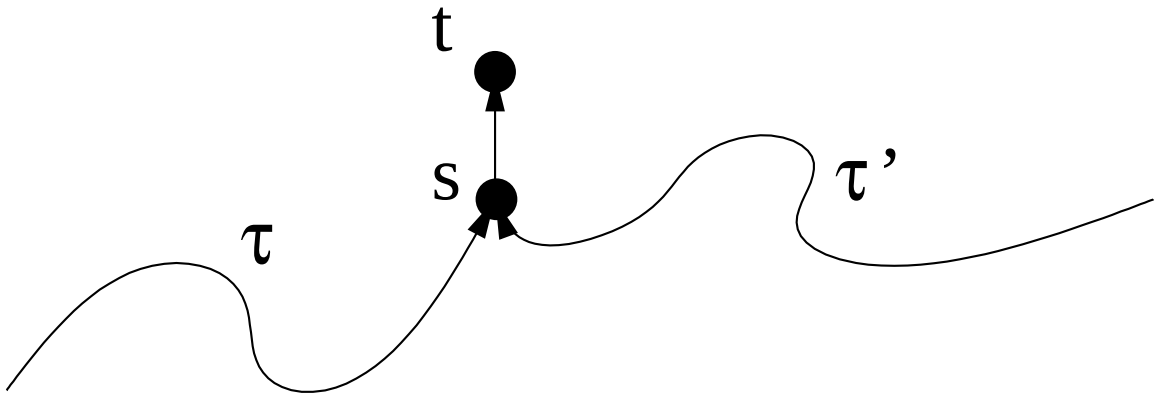
Dijkstra's algorithm

- Dijkstra's algorithm can be slightly modified to compute optimum-path forests for MI cost functions.
- Dijkstra's algorithm also works for non-MI cost functions under weaker conditions which are applied to only optimum paths.

Smooth path-cost functions

A cost function f is smooth if for any node $t \in \mathcal{I}$, there is an optimum path π ending at t which either is trivial, or has the form $\tau \cdot \langle s, t \rangle$ where

1. $f(\tau) \leq f(\pi)$,
2. τ is optimum,
3. for any optimum path τ' ending at s, $f(\tau' \cdot \langle s, t \rangle) = f(\pi)$.



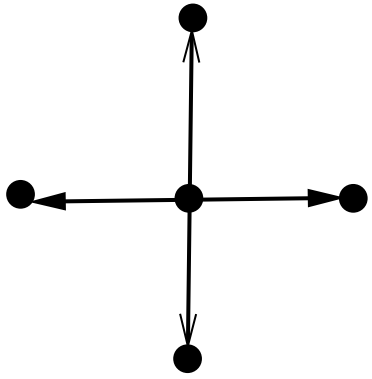
These conditions apply to only optimum paths.

An image as a directed graph

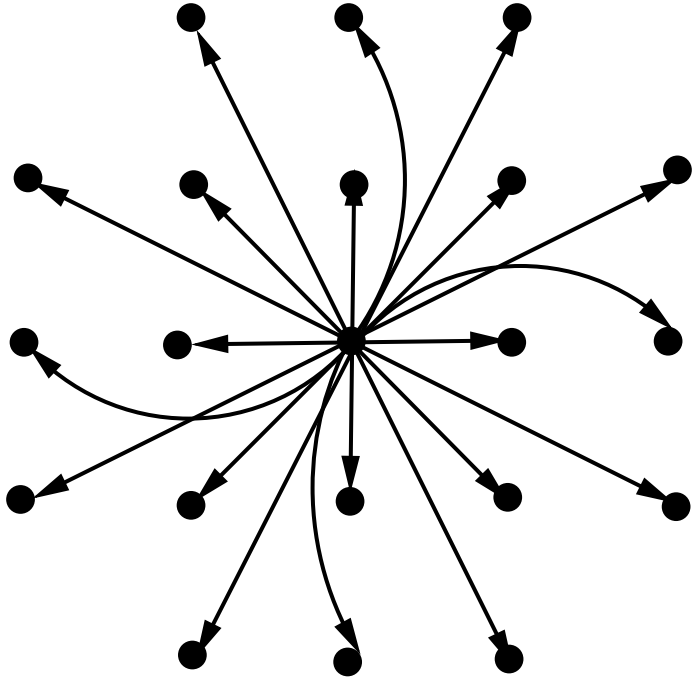
- A grayscale image $\mathbf{I}$ is a pair $(\mathcal{I}, I)$, where $\mathcal{I}$ is a finite set of pixels (points in $\mathbb{Z}^2$) and I assigns to each pixel $t \in \mathcal{I}$ a value $I(t)$ in some arbitrary value space.
- An adjacency relation $\mathcal{A}$ is a binary relation between pixels of $\mathcal{I}$, which is usually translation-invariant.
- Once $\mathcal{A}$ has been fixed, image $\mathbf{I}$ can be interpreted as a directed graph, whose nodes are the image pixels in $\mathcal{I}$ and whose arcs are defined by $\mathcal{A}$.

Examples of adjacency relations

Euclidean relation: A pixel $t = (x_t, y_t)$ is adjacent to a pixel $s = (x_s, y_s)$ (i.e. arc $(s, t) \in \mathcal{A}$) if $(x_t - x_s)^2 + (y_t - y_s)^2 \leq \rho$.



(A)

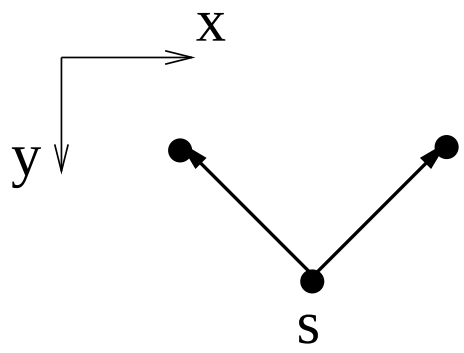


(B)

(A) $\rho = 1$ and (B) $\rho = 25$.

Examples of adjacency relations

- Box-shaped relation.
  Arc $(s,t) \in \mathcal{A}$ if $|x_t - x_s| \leq \frac{a}{2}$ and $|y_t - y_s| \leq \frac{b}{2}$.
- Set-based relation.
  Arc $(s,t) \in \mathcal{A}$ if $t - s \in \{(-1,-1),(1,-1)\}$.



(C)

Connectivity relation

- A pixel t is said connected to a pixel s if there is a path from s to t in the graph.
- The cost of a path in the graph is determined by an application-specific smooth path-cost function which usually depends on local image properties along the path, such as brightness, gradient, and pixel position.

This notion of connectivity can be exploited in many different ways.

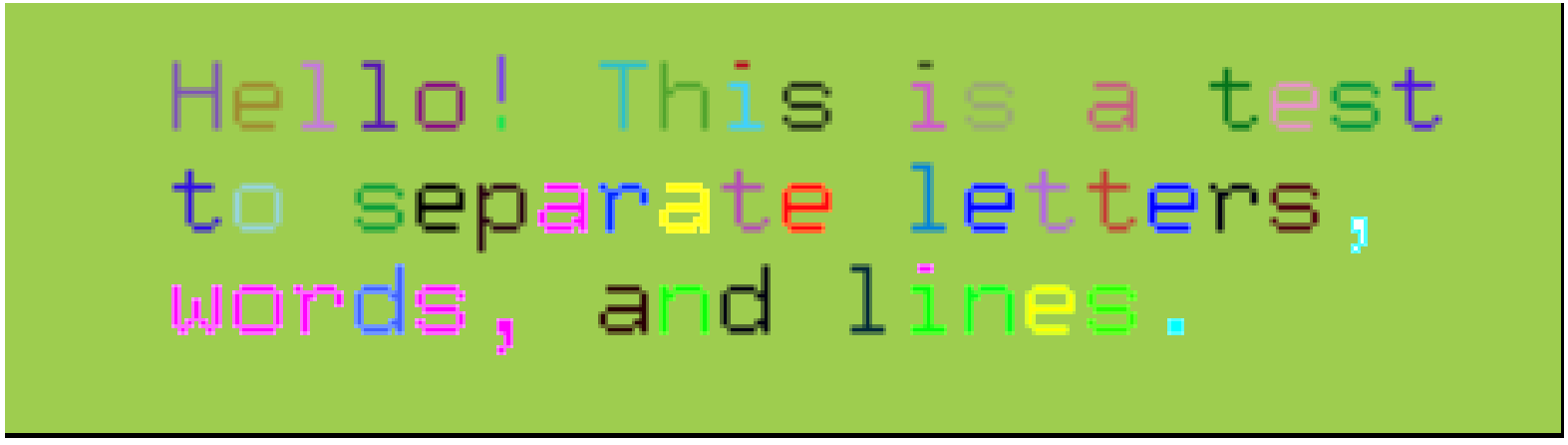
For example, consider the labeling of components using symmetric connectivity relations...

Labeling of components

Hello! This is a test
to separate letters,
words, and lines.

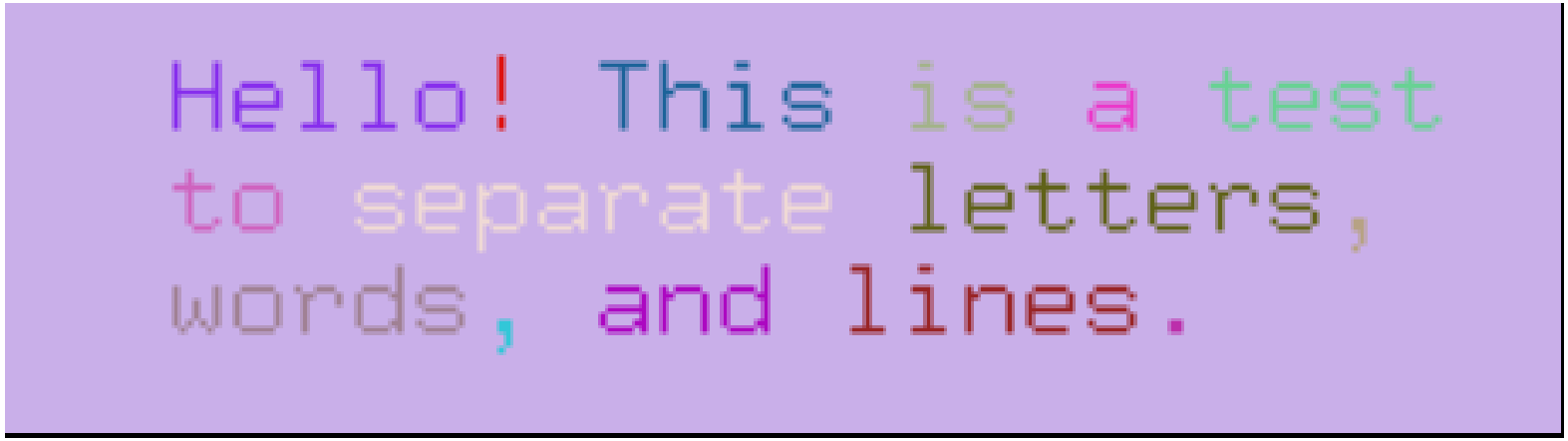
Labeling of components

For example, consider Euclidean adjacency relation with $\rho = 2$ for labeling letters of a binary text;



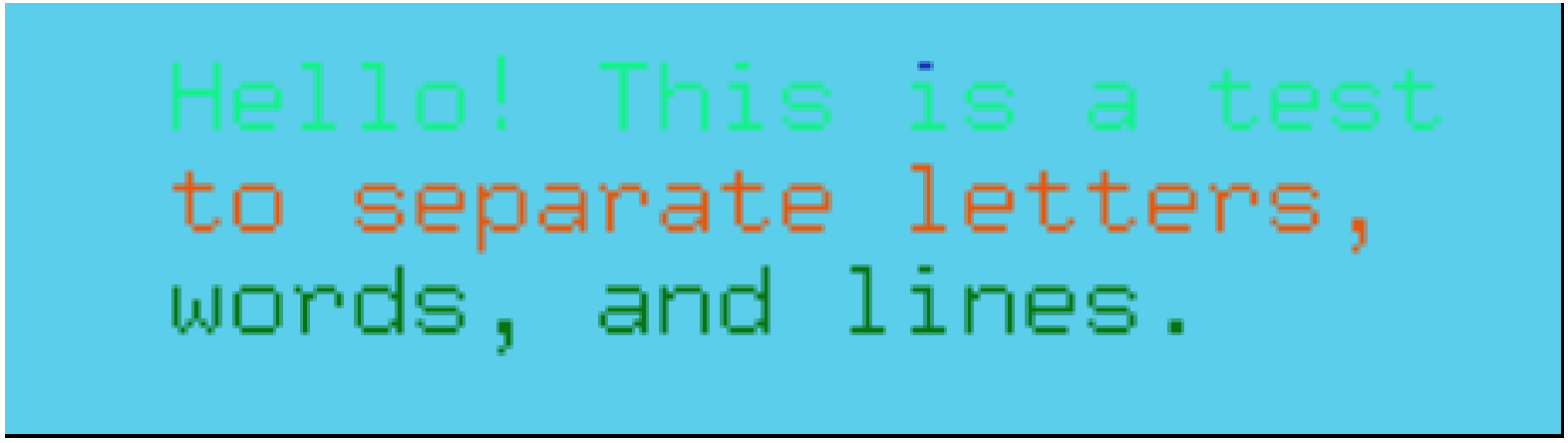
Labeling of components

Euclidean adjacency relation with $\rho = 25$ for labeling words of a text; and



Labeling of components

box-shaped adjacency relation with $a = 30$ and $b = 5$ for labeling lines of a text.



Labeling of components

Consider now $(s,t) \in \mathcal{A}$ if $(x_t - x_s)^2 + (y_t - y_s)^2 \leq 100$ and $|I(t) - I(R(s))| \leq 100$, where $R(s)$ is the initial pixel of the first path that reached s.

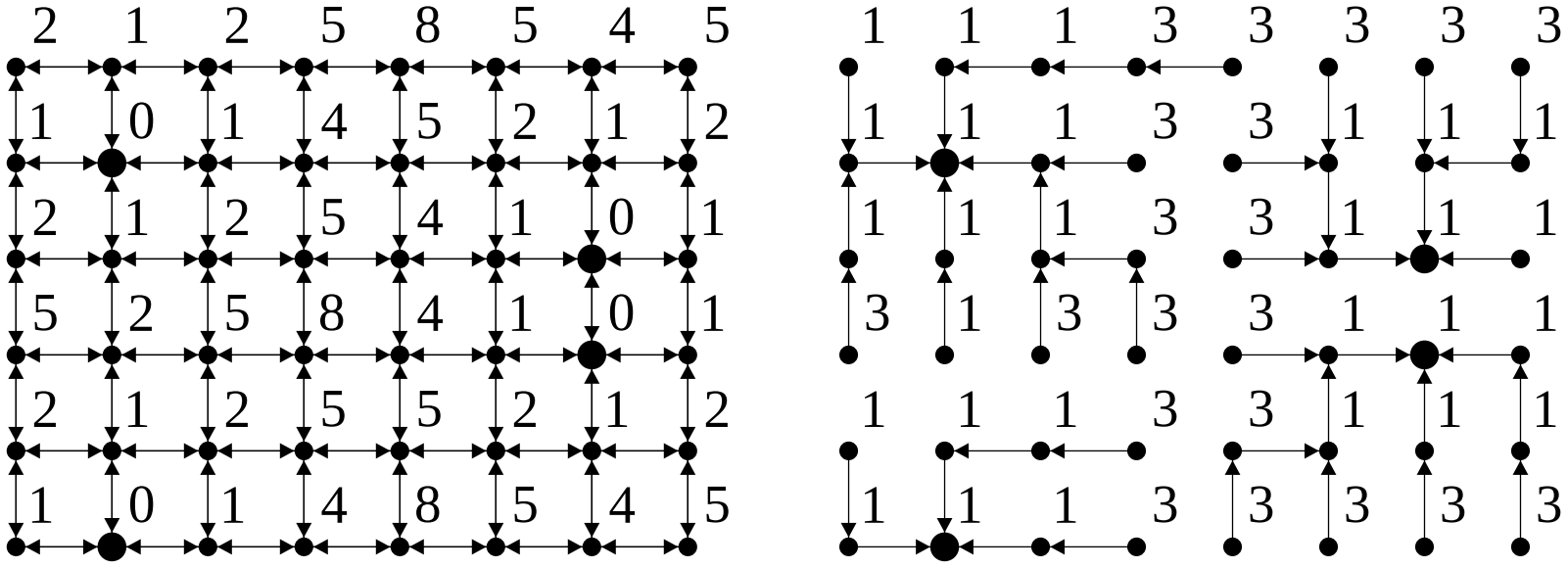
Labeling algorithm

Input: Image $\mathbf{I} = (\mathcal{I}, I)$ and adjacency relation $\mathcal{A}$.
Output: Labeled image $\mathbf{L} = (\mathcal{I}, L)$, where $L(t) = 0$ initially.
Auxiliary: FIFO Q and variable $l = 1$ initially.

1. For all $p \in \mathcal{I}$, such that $L(p) = 0$, do
2. Set $L(p) \leftarrow l$ and insert p in Q.
3. While $Q \neq \emptyset$ do
4. Remove s from Q.
5. For all t, such that $(s, t) \in \mathcal{A}$ and $L(t) = 0$, do
6. Set $L(t) \leftarrow L(s)$ and insert t in Q.
7. Set $l \leftarrow l + 1$.

Image Foresting Transform

The IFT takes an image $\mathbf{I}$, a smooth path-cost function f and an adjacency relation $\mathcal{A}$; and returns an optimum-path forest P.



A 4-connected image graph and the optimum forest for max-arc function f_{max} with $h(t) = I(t) + 1$ and $w(s,t) = |I(t) - I(s)|$.

Tie-breaking

The optimum-path forest P may not be unique, because a pixel may be reached from two or more roots at the same minimum cost. This ambiguity requires tie-breaking policies.

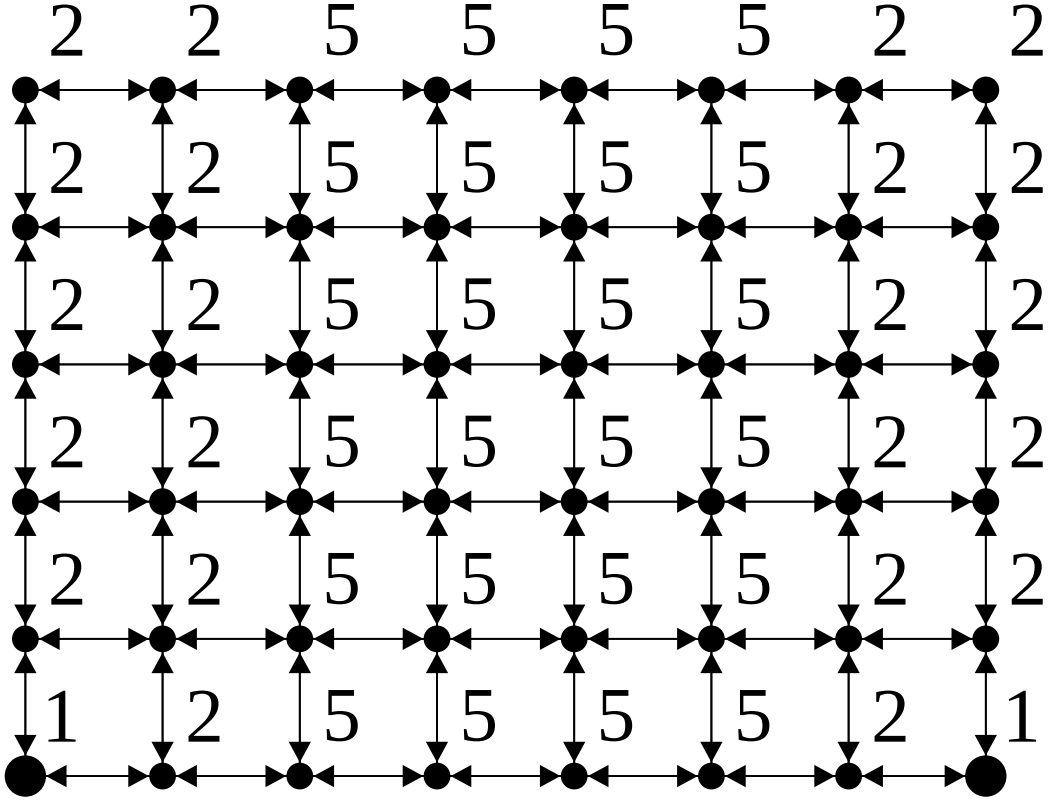
- FIFO policy
  Any connected set $\mathcal{X}$ of pixels with minimum cost with respect to two or more roots tends to be equally partitioned among the respective trees.
- LIFO policy
  The pixels in $\mathcal{X}$ will all get assigned to a single tree.

IFT algorithm

- The algorithm first identifies a root set and then computes optimum paths within wavefronts that grow from each root pixel.
- During this process, the algorithm propagates for each pixel t its predecessor $P(t)$ in the optimum path with terminus t, the cost $C(t)$ of that path, and its corresponding root $R(t)$.
- The process stops when the wavefronts encounter each other.
- A priority queue Q stores the pixels of the wavefronts and controls their shapes, by favoring growth toward directions of minimum-cost paths or according to FIFO/LIFO policy in the case of ties.

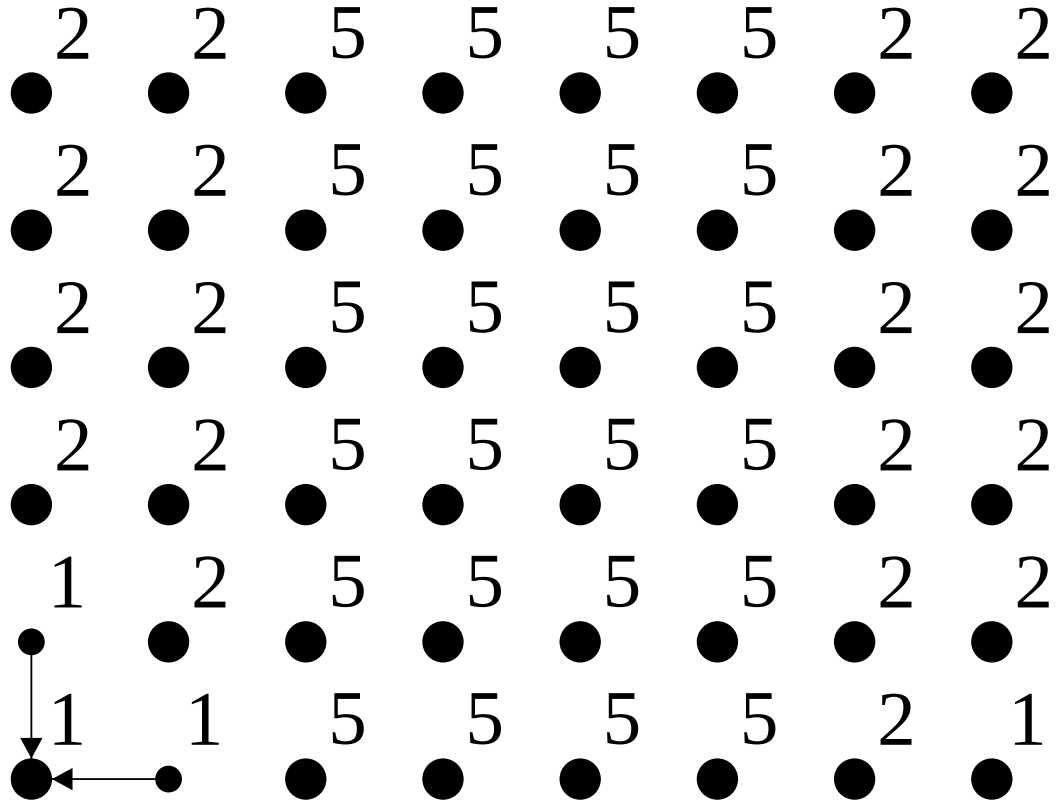
FIFO tie-breaking

Consider the max-arc function f_{max} with $h(t) = I(t)$, $w(s,t) = |I(t) - I(s)|$, and a 4-connected image graph.



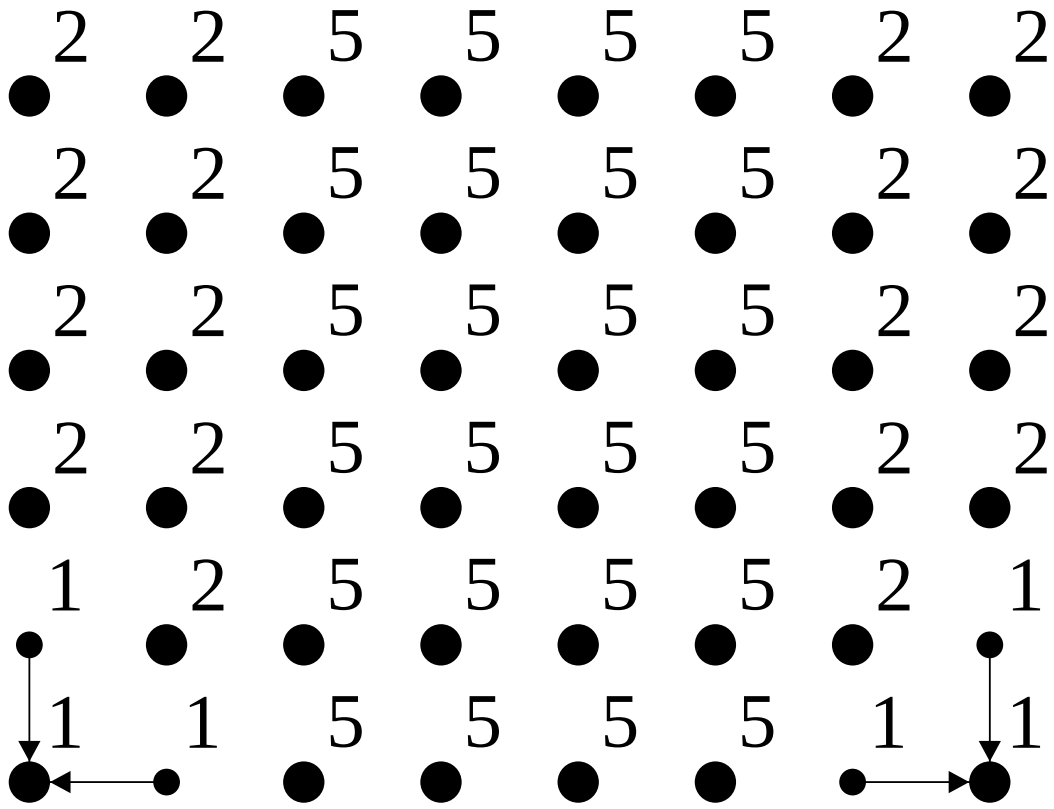
FIFO tie-breaking

After the first iteration of the algorithm.



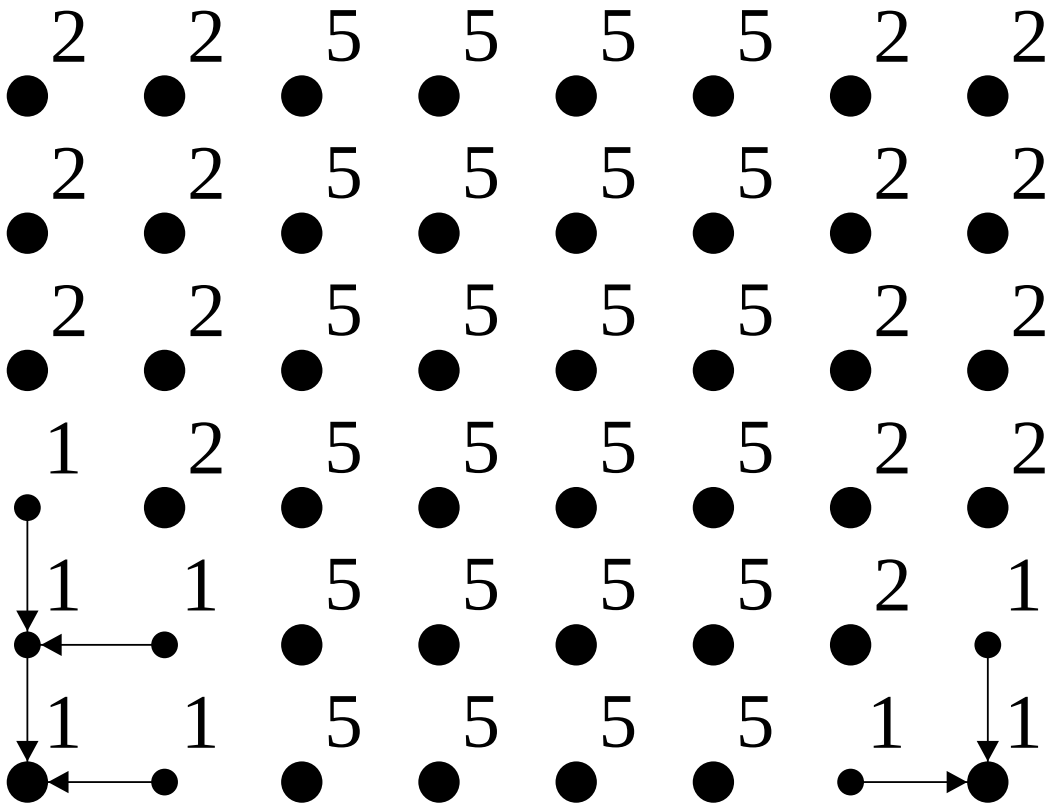
FIFO tie-breaking

After the second iteration of the algorithm.



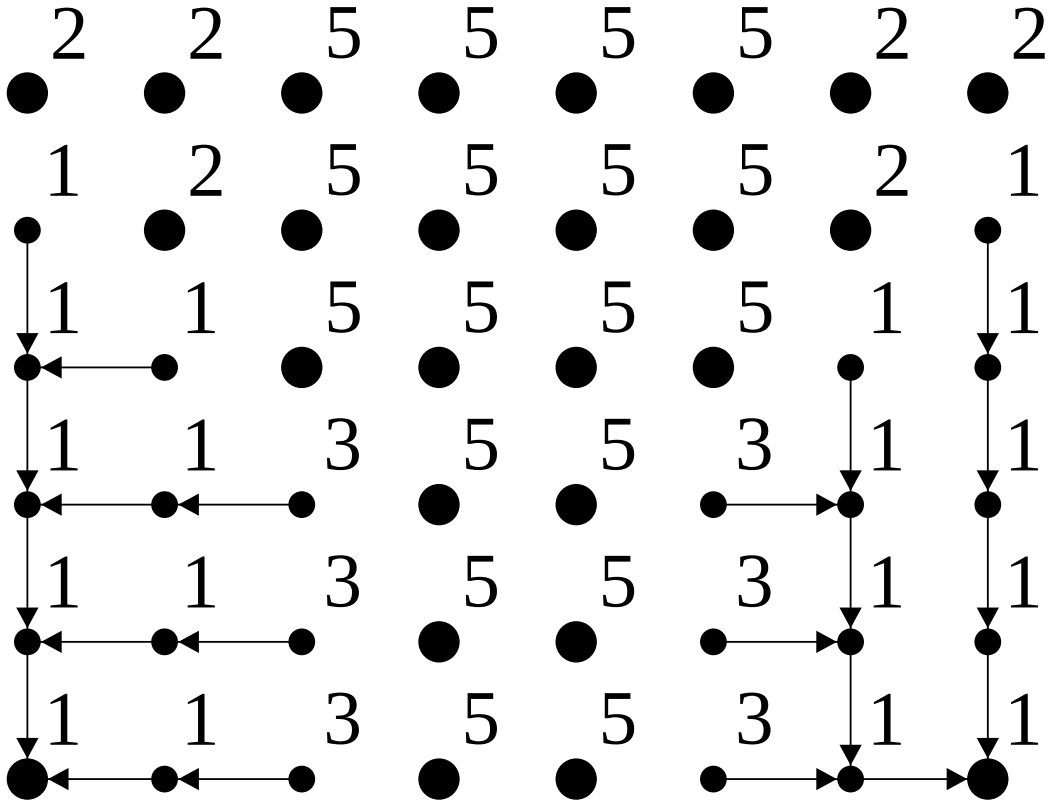
FIFO tie-breaking

After the third iteration of the algorithm.



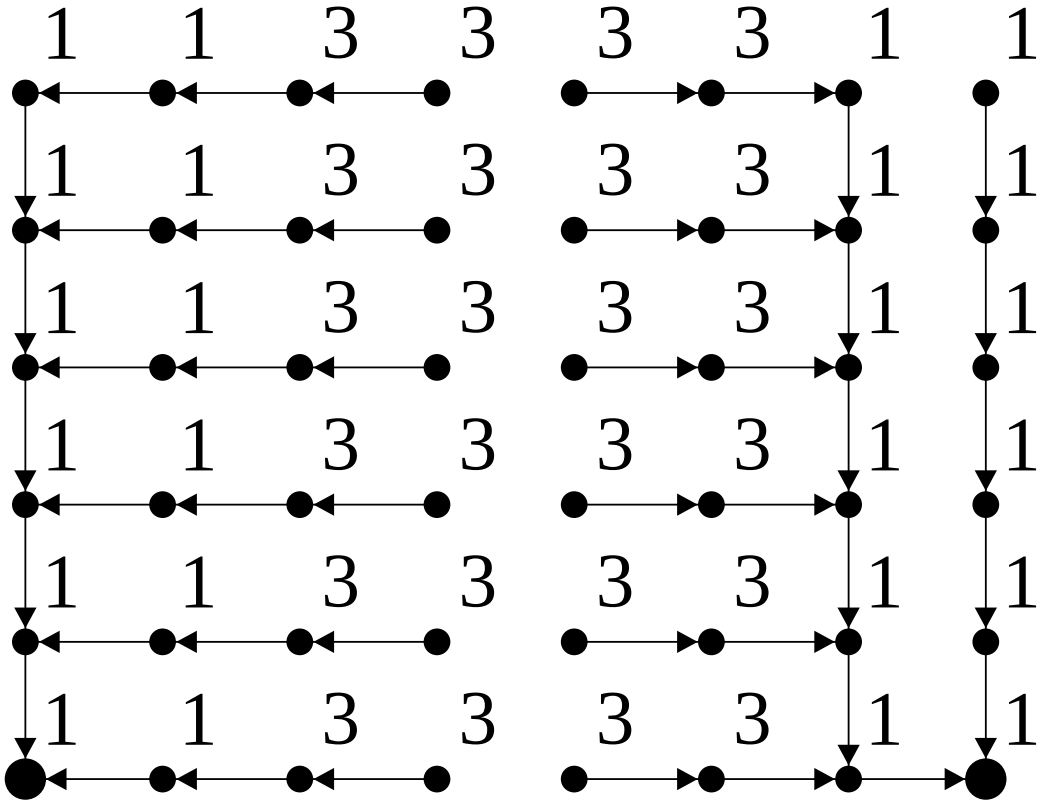
FIFO tie-breaking

After 15 iterations of the algorithm.



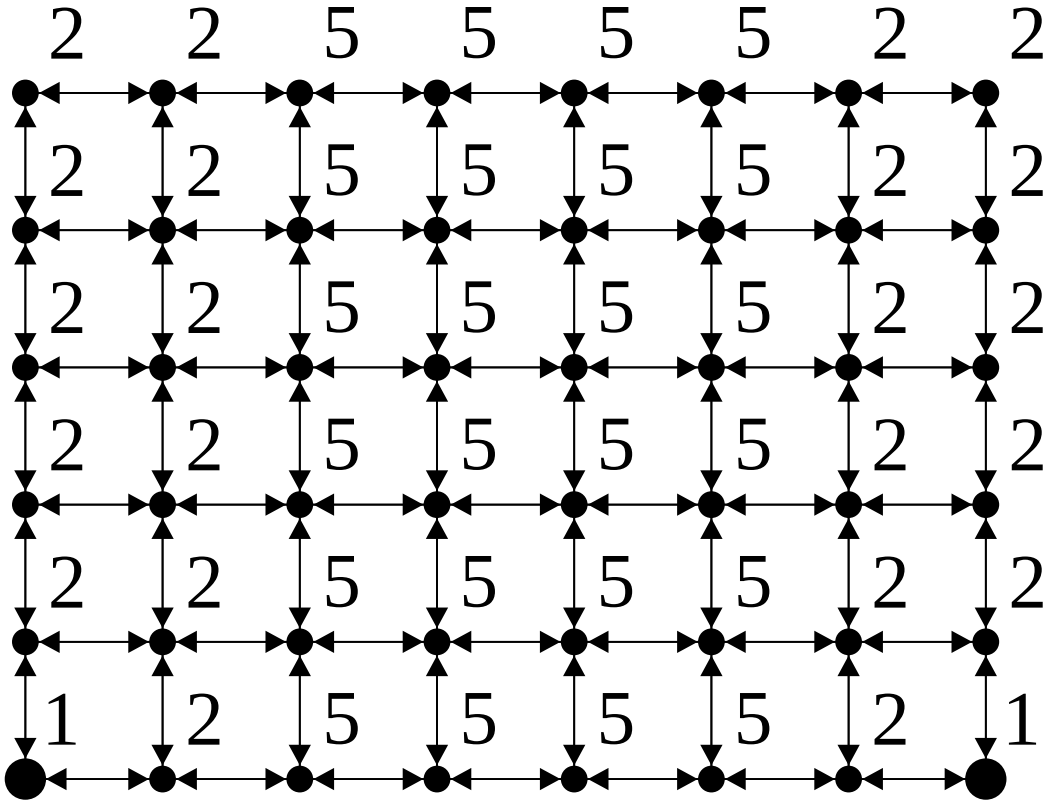
FIFO tie-breaking

The resulting optimum-path forest.



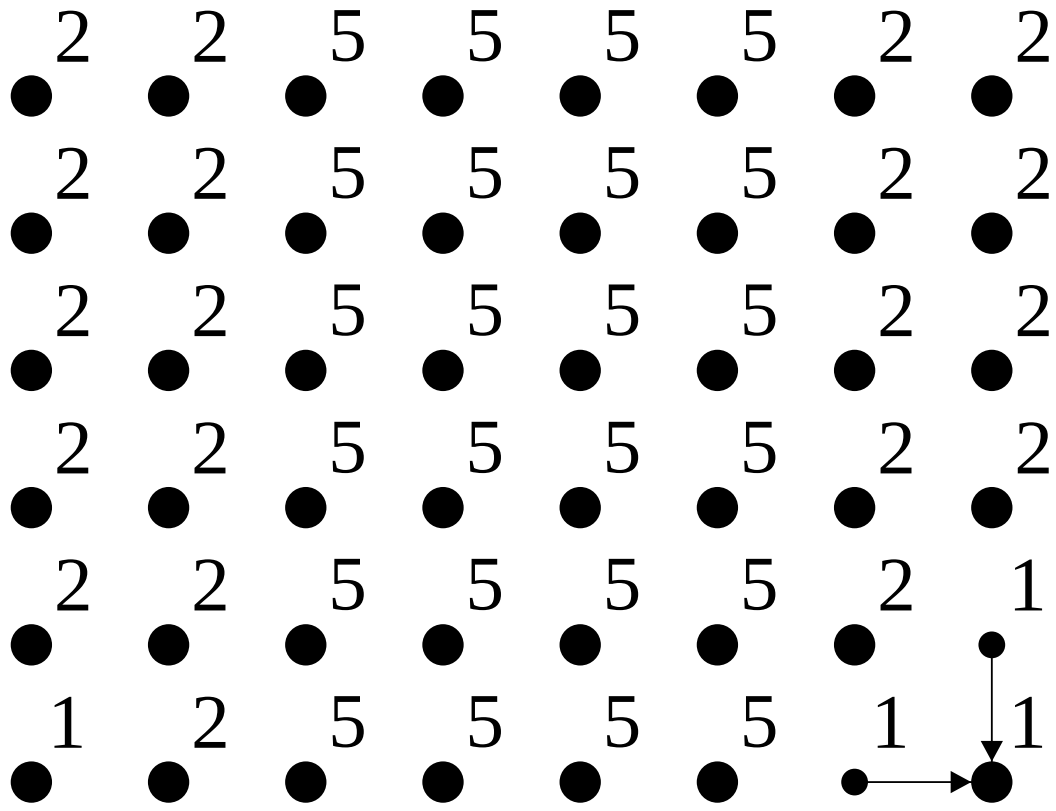
LIFO tie-breaking

Consider the same max-arc function f_{max} with $h(t) = I(t)$, $w(s,t) = |I(t) - I(s)|$, and 4-connected image graph.



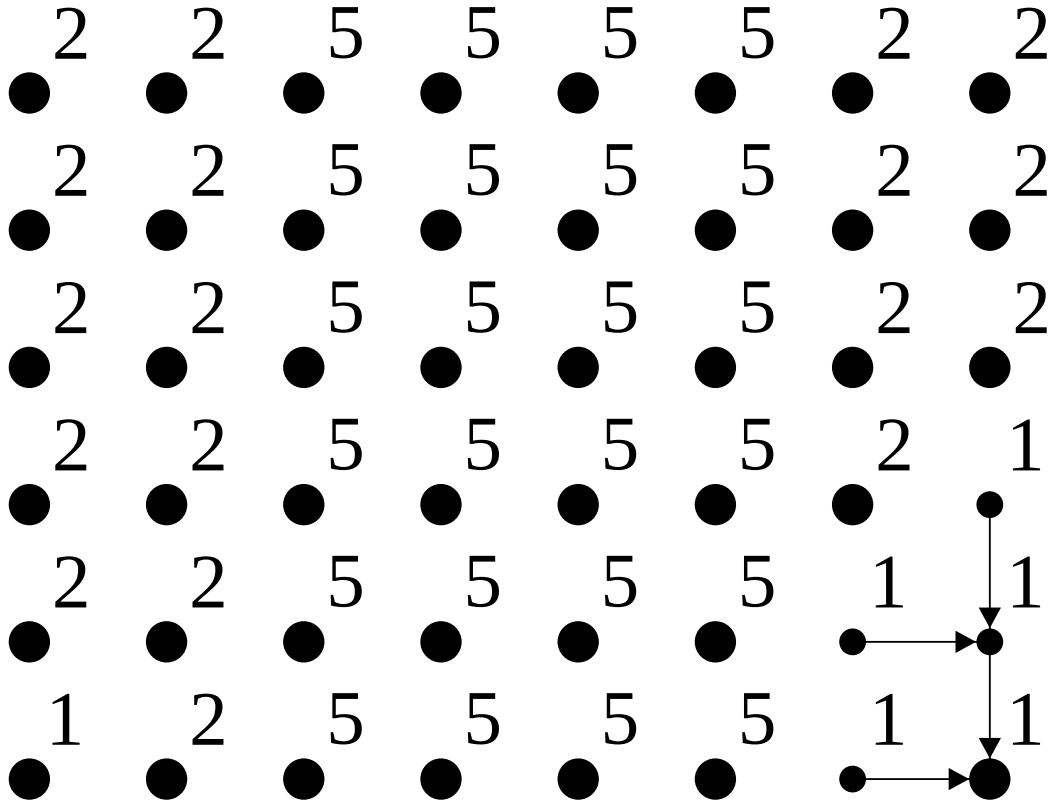
LIFO tie-breaking

After the first iteration of the algorithm.



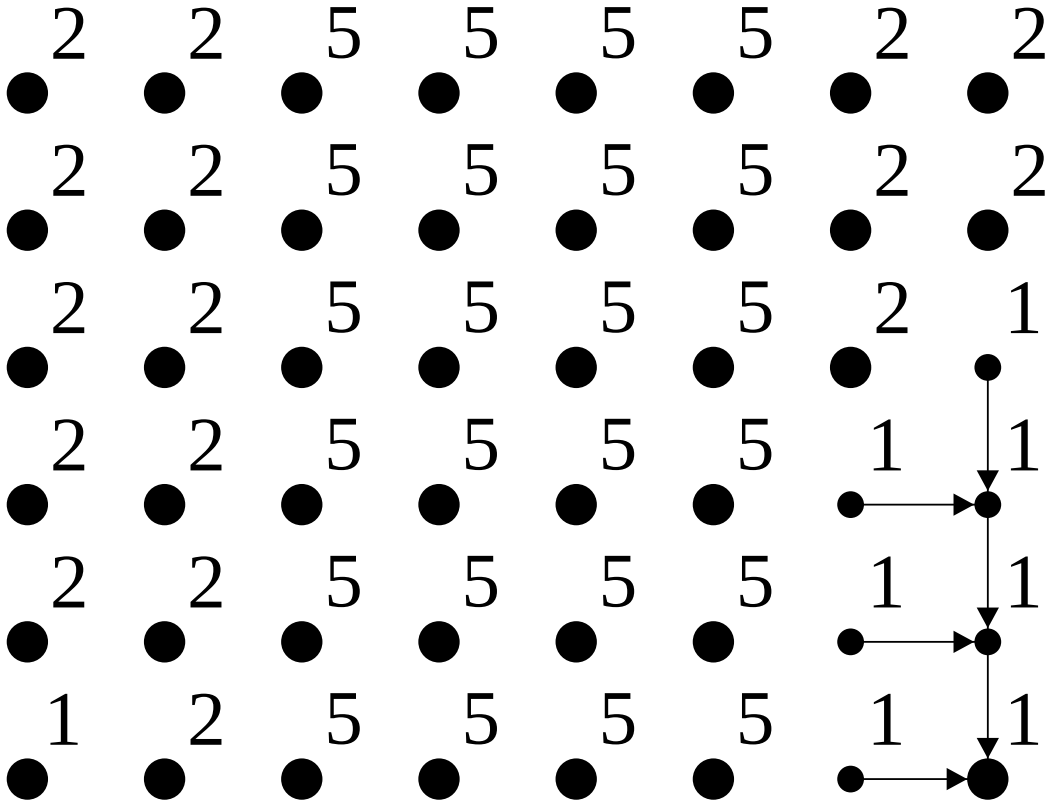
LIFO tie-breaking

After the second iteration of the algorithm.



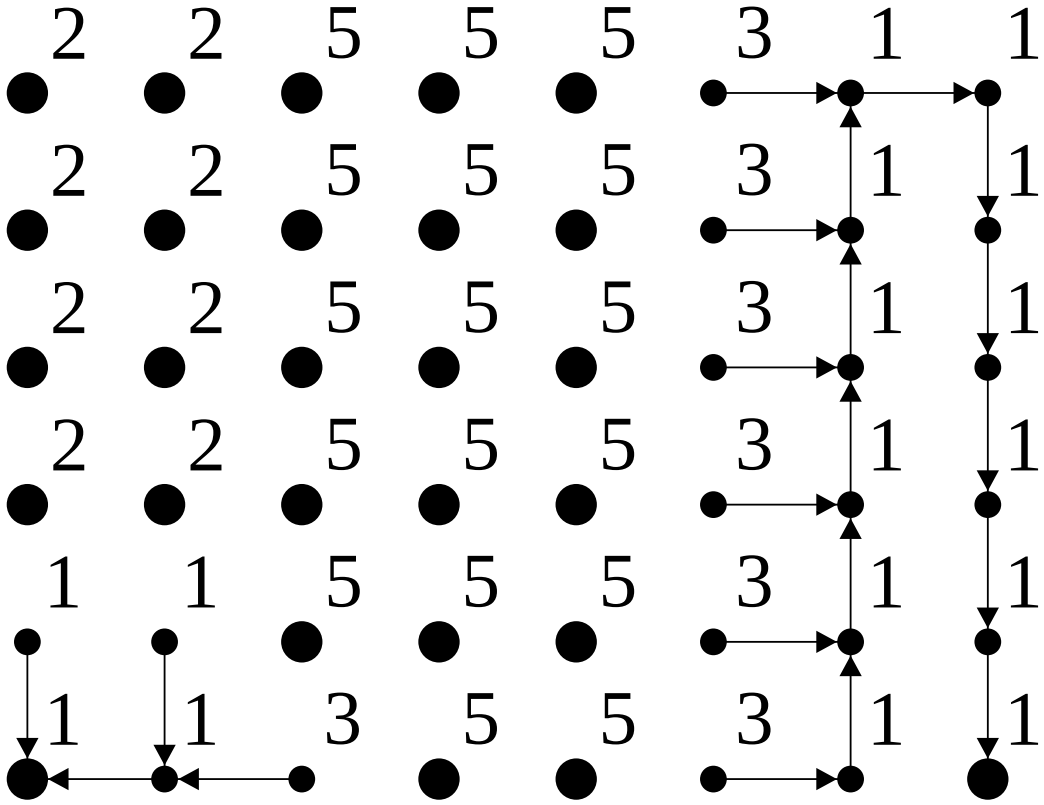
LIFO tie-breaking

After the third iteration of the algorithm.



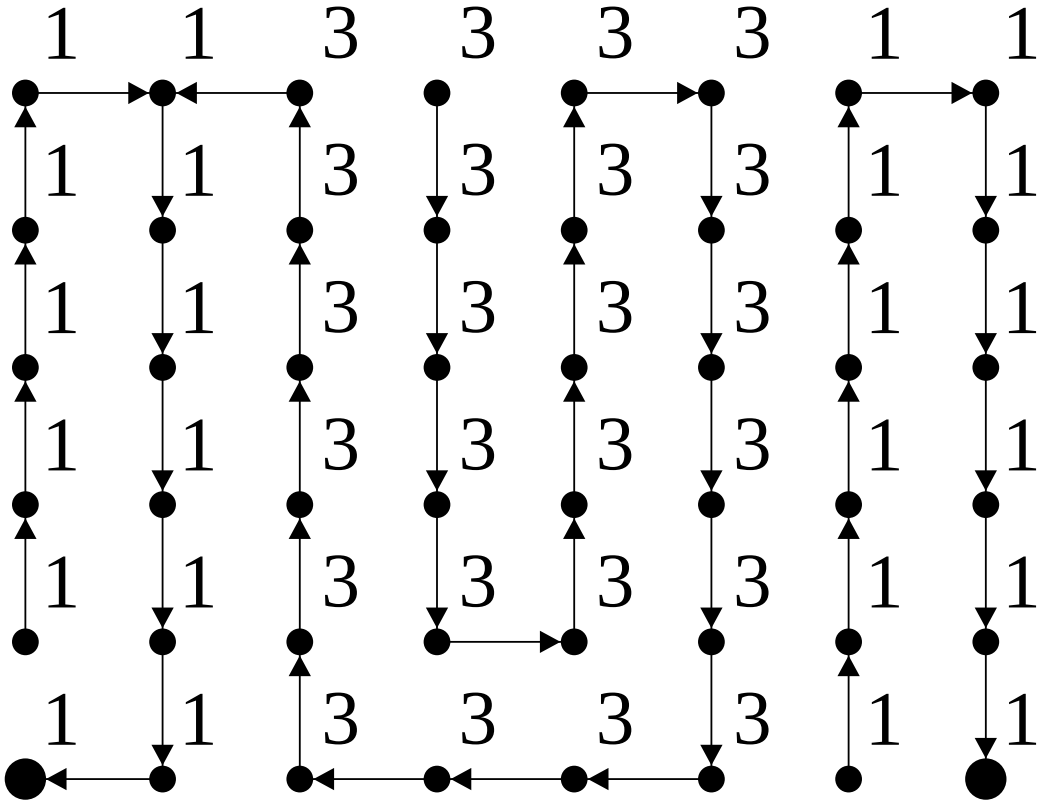
LIFO tie-breaking

After 15 iterations of the algorithm.



LIFO tie-breaking

The resulting optimum-path forest.



IFT algorithm with FIFO policy

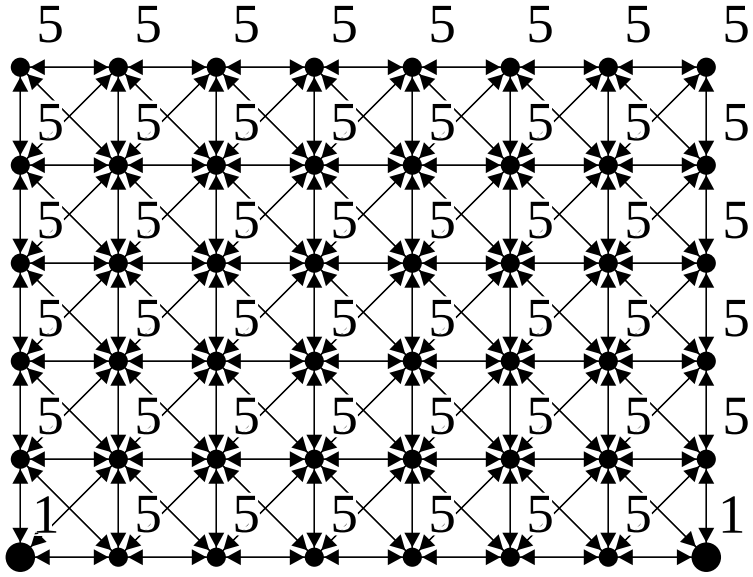
1. For all pixels $t \in \mathcal{I}$ do
2. $\quad C(t) \leftarrow f(\langle t \rangle)$, $P(t) \leftarrow nil$, $R(t) \leftarrow t$.
3. $\quad$ If $C(t) < +\infty$, insert t in Q.
4. While $Q \neq \emptyset$ do
5. $\quad$ Remove a pixel s from Q such that $C(s)$ is minimum.
6. $\quad$ For all t, such that $(s,t) \in \mathcal{A}$ and $C(t) > C(s)$, do
7. $\quad\quad c \leftarrow f(P^*(s) \cdot \langle s,t \rangle)$.
8. $\quad\quad$ If $c < C(t)$ then
9. $\quad\quad\quad$ If $C(t) \neq +\infty$, remove t from Q.
10. $\quad\quad\quad C(t) \leftarrow c$, $P(t) \leftarrow s$, $R(t) \leftarrow R(s)$, insert t in Q.

IFT algorithm with LIFO policy

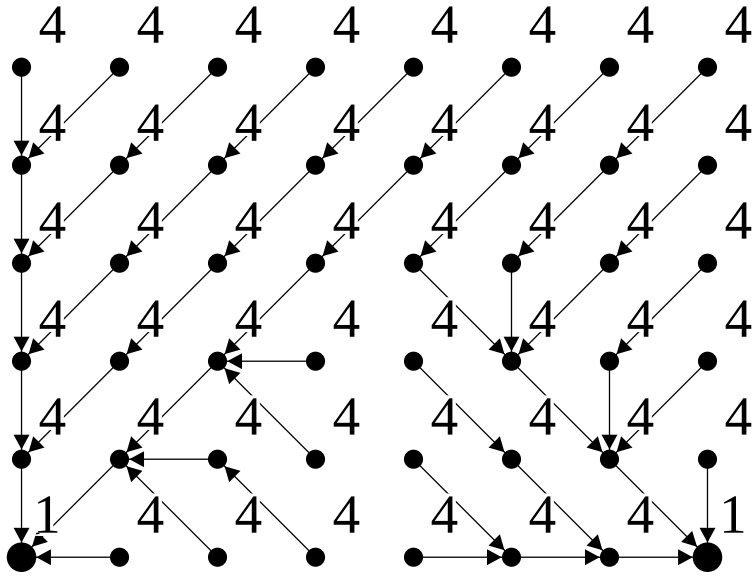
1. For all pixels $t \in \mathcal{I}$ do
2. $C(t) \leftarrow f(\langle t \rangle)$, $P(t) \leftarrow nil$, $R(t) \leftarrow t$, insert t in Q.
3. While $Q \neq \emptyset$ do
4. Remove a pixel s from Q such that $C(s)$ is minimum.
5. For all t, such that $(s,t) \in \mathcal{A}$ and $t \in Q$, do
6. $c \leftarrow f(P^*(s) \cdot \langle s,t \rangle)$.
7. If $c \leq C(t)$ then
8. Remove t from Q, $C(t) \leftarrow c$, $P(t) \leftarrow s$, $R(t) \leftarrow R(s)$, insert t in Q.

Unfair partitions with FIFO

Function f_{max} with $h(t) = I(t)$ and $w(s,t) = |I(t) - I(s)|$.



(a) Image graph



(b) Optimum forest

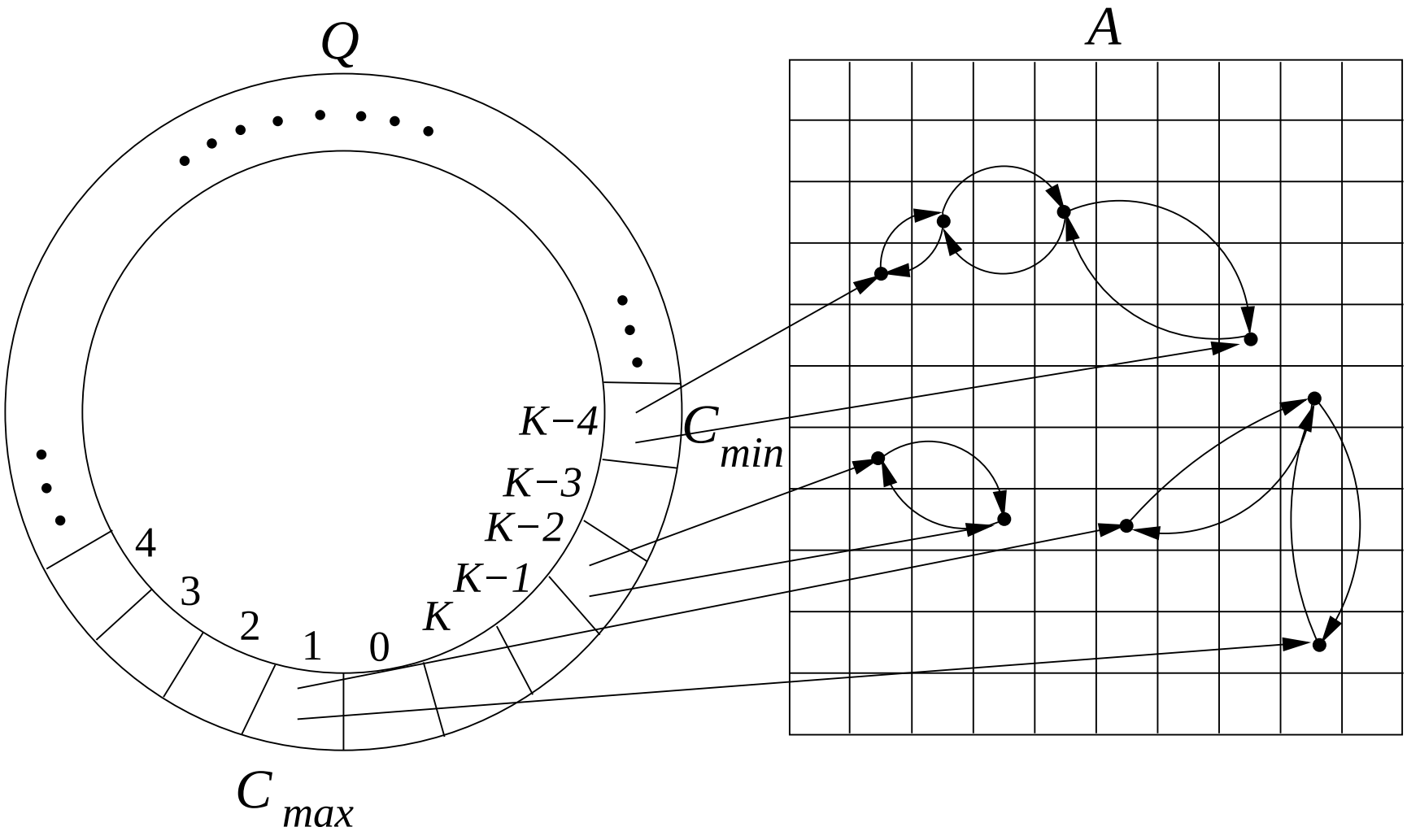
Other tie-breaking policies may be incorporated to the cost function.

Priority queue Q

- If Q is a binary heap, the algorithm will run in $O(m + n \log n)$ time and $O(n)$ storage, where $m = |\mathcal{A}|$ and $n = |\mathcal{I}|$.
- In most applications, we can find a small integer K such that $f(\pi \cdot \langle s, t \rangle) - f(\pi) \leq K$ and $f(\langle t \rangle) - f(\langle t' \rangle) \leq K$ (excluding $+\infty$ values). The cost of pixels in Q will be integers in the range $[C..C + K]$, for some cost C that varies during the algorithm. Then Q can be a circular queue of $K + 1$ entries; and the algorithm will run in $O(m + n + K)$ time and $O(n + K)$ storage.

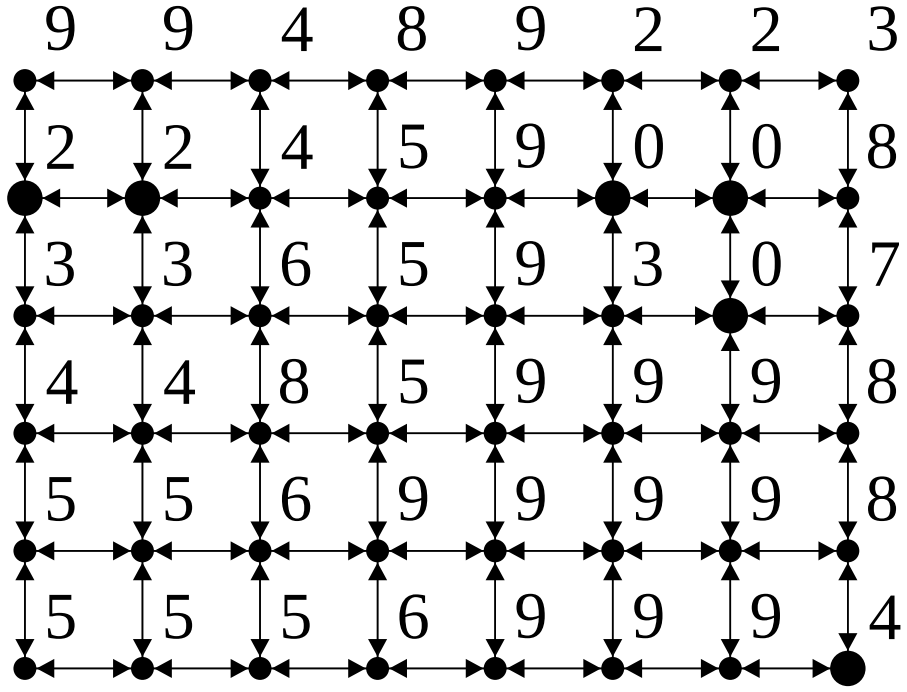
For small adjacency relations (sparse graphs), $m \ll n$, the algorithm will run in linear time $O(n)$.

Circular priority queue



Detection of regional minima

A regional minimum is a maximal connected set $\mathcal{X} \subseteq \mathcal{I}$ of same gray value, such that $I(s) < I(t)$ for any arc $(s,t) \in \mathcal{A}$ where $s \in \mathcal{X}$ and $t \notin \mathcal{X}$.



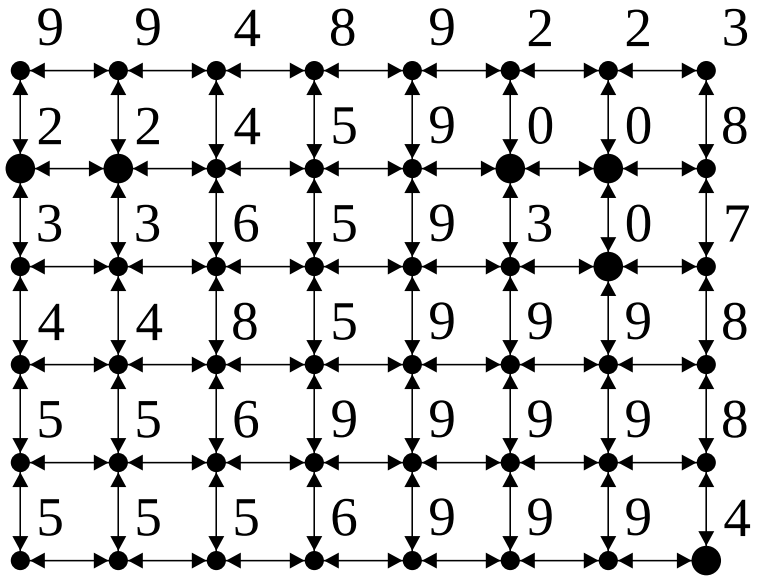
Cost function for regional minima

$$f_{ini}(\langle t \rangle) = I(t), \text{ for all } t \in \mathcal{I},$$

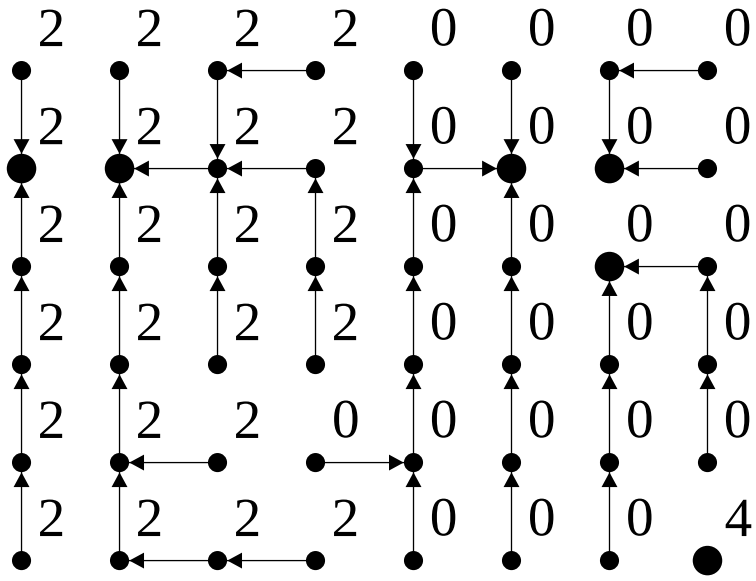
$$f_{ini}(\pi \cdot \langle s,t \rangle) = \begin{cases} f_{ini}(\pi), & \text{if } I(s) \leq I(t), \\ +\infty & \text{otherwise.} \end{cases}$$

- All image pixels are root candidates.
- Any optimum path starts at a regional minimum and never goes downhill.

Regional minima with FIFO



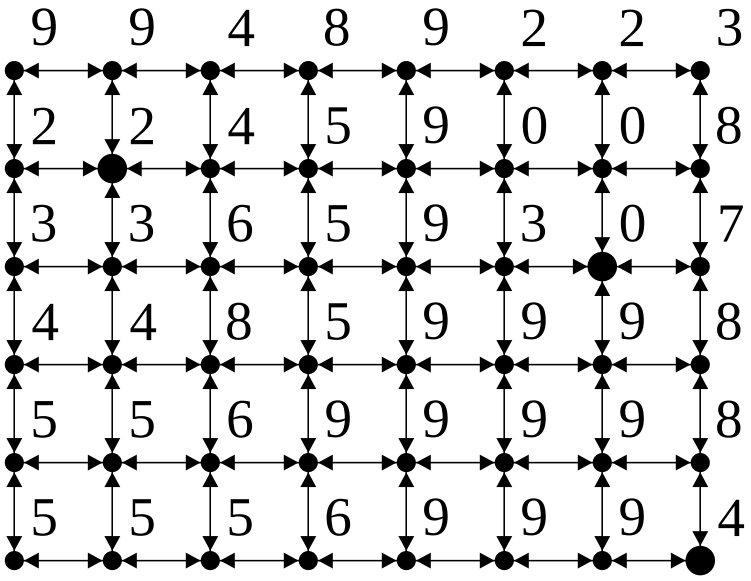
(a) Image graph



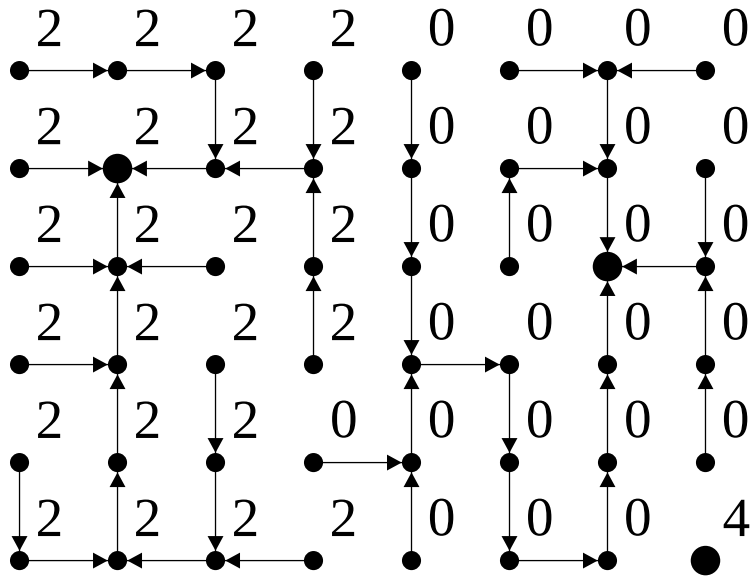
(b) Optimum forest

The regional minima form the root set of the forest and each minimum $\mathcal{X}$ is a connected component of this set.

Regional minima with LIFO



(a) Image graph



(b) Optimum forest

The number of minima is the number of roots, and their extent is detected by selecting the pixels t where $I(t) = I(R(t))$.

Seed pixels

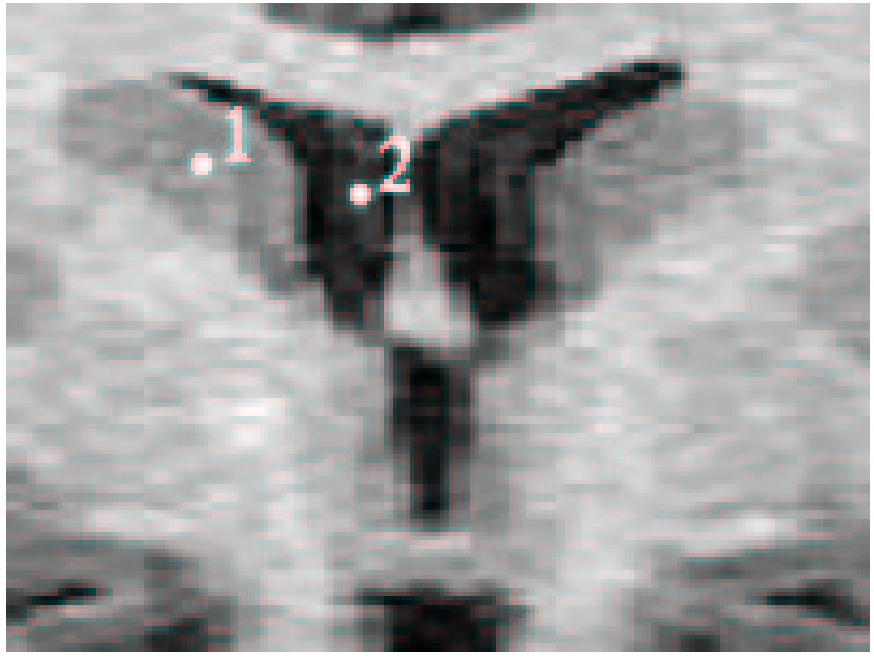
In some applications, we would like to use a smooth cost function f but constrain the search to paths that start in a given set $\mathcal{S} \subseteq \mathcal{I}$ of seed pixels. This constraint can be modeled by defining

$$f^{\mathcal{S}}(\pi) = \begin{cases} f(\pi), & \text{if } \pi \text{ starts in } \mathcal{S}, \\ +\infty & \text{otherwise.} \end{cases}$$

- It is valid for any MI function, but not for all smooth cost functions— in which case the IFT algorithm outputs a non-optimum spanning forest.
- A seed t may not become root, because there may be another path from $\mathcal{S}$ cheaper than $\langle t \rangle$.

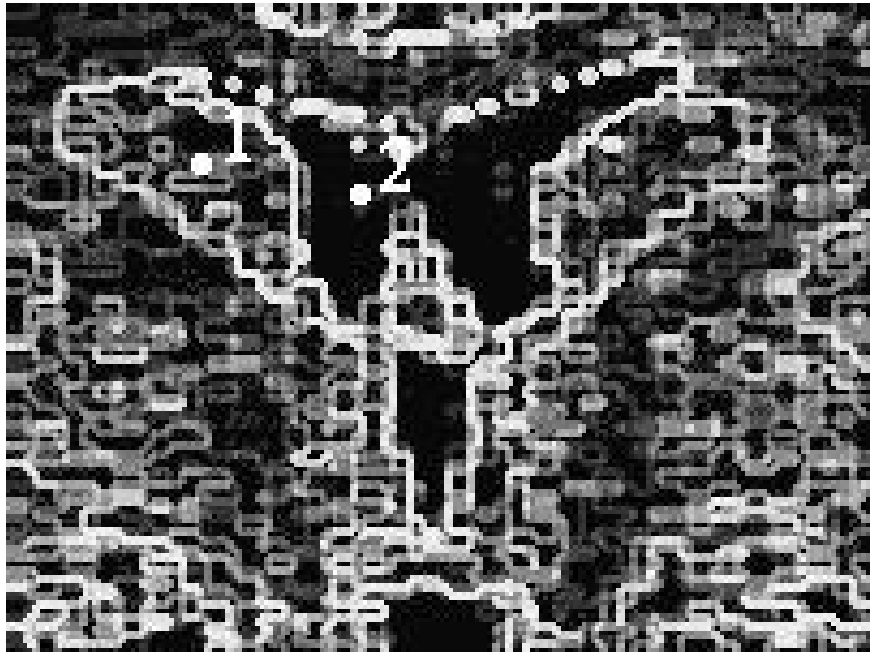
Region-based image segmentation

Consider an MR image $\mathbf{I} = (\mathcal{I}, I)$ of a brain, where the object of interest is the left caudate nucleus.



The IFT-watershed approach

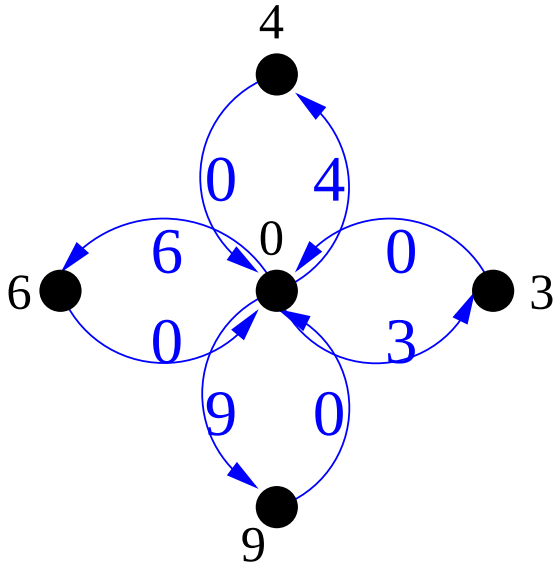
A gradient image $\mathbf{G} = (\mathcal{I}, G)$ obtained from $\mathbf{I}$ with the seeds selected inside and outside the left caudate nucleus.



Cost function for the caudate nucleus

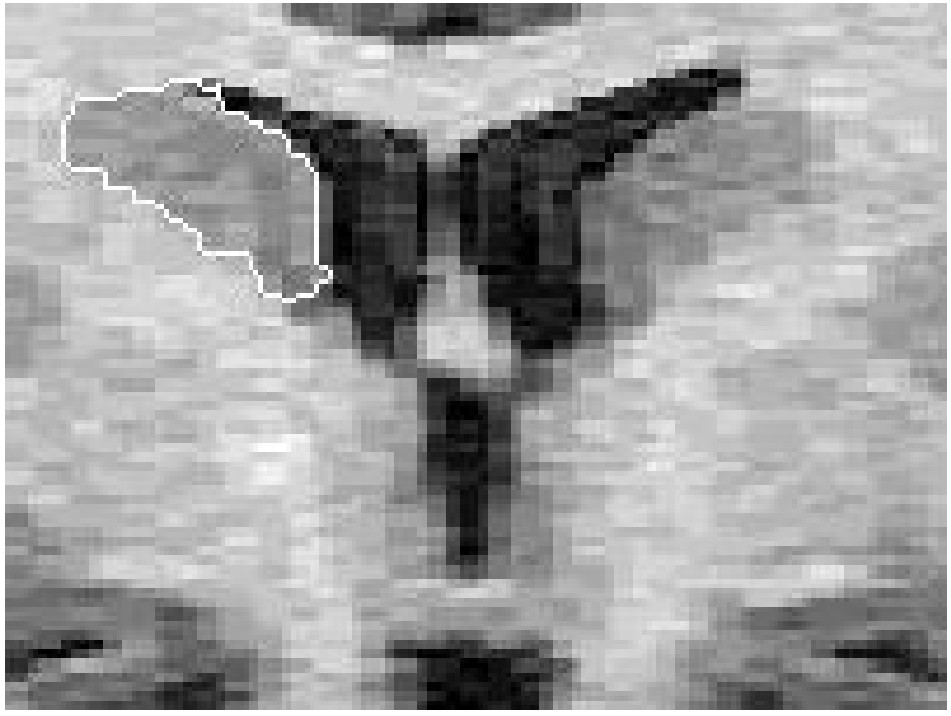
The path-cost function can be the maximum pixel intensity along the path.

$$
\begin{aligned}
f_{peak}(\langle t \rangle) &= I(t), \text{if } t \in \mathcal{S}, \text{ and } +\infty \text{ otherwise.} \\
f_{peak}(\pi \cdot \langle s,t \rangle) &= \max\{f_{peak}(\pi), I(t)\}.
\end{aligned}
$$



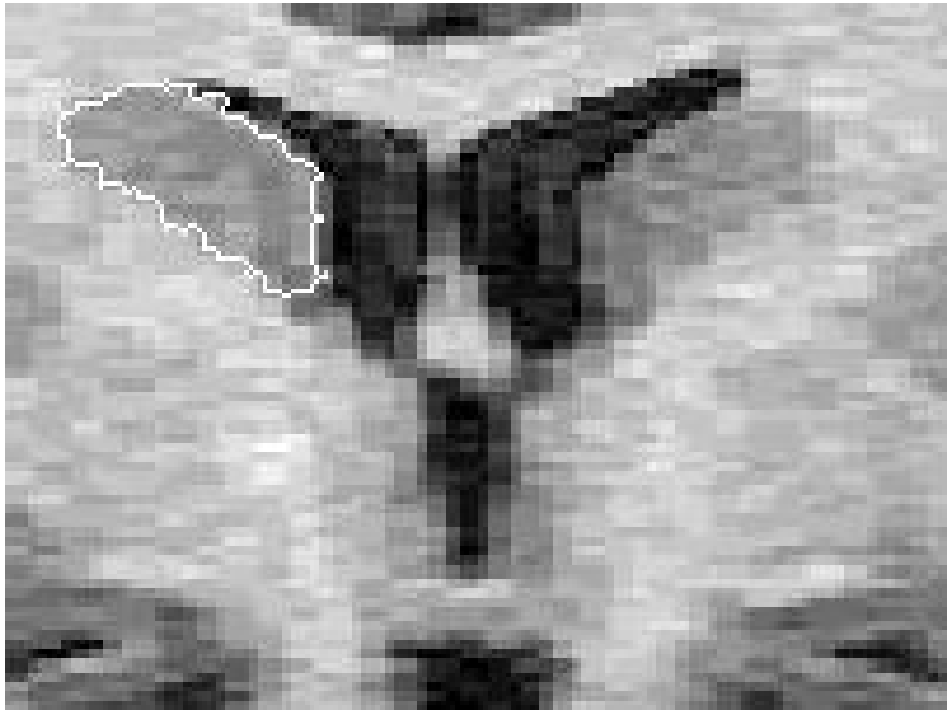
Left caudate nucleus

Result of the IFT with FIFO policy and 8-neighborhood, where the object was obtained from the root map R.



Left caudate nucleus

Result of the IFT with LIFO policy and 8-neighborhood, where the object was obtained from the root map R.

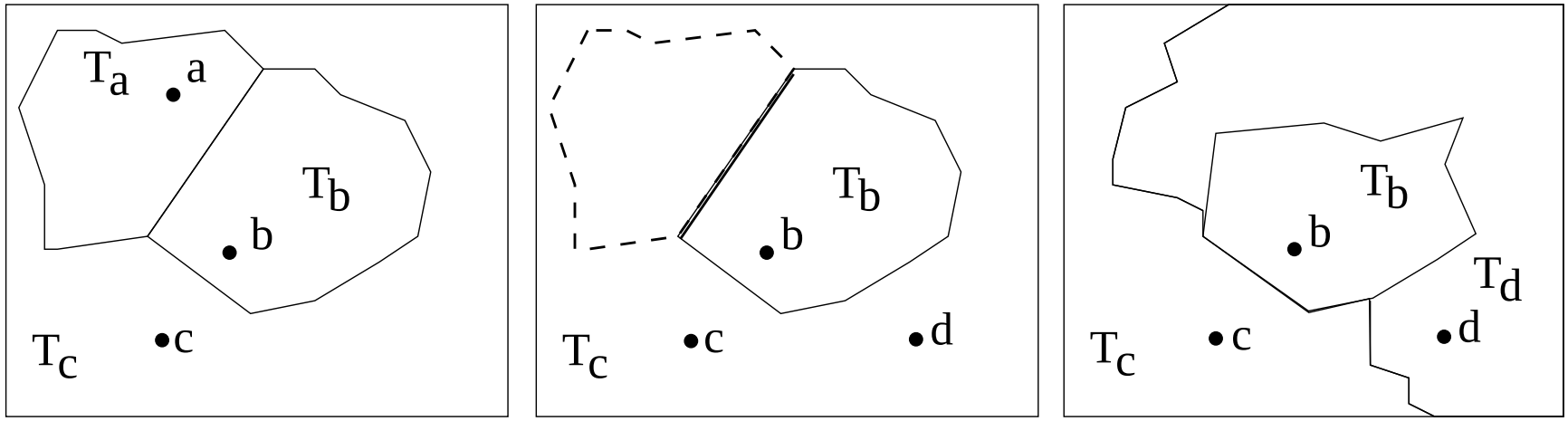


Differential IFT

- The differential IFT (DIFT) allows to compute sequences of IFTs in a differential way.
- At each iteration, trees may be marked for deletion and seeds added for a new dispute among the remaining roots.
- The optimum forest, costs, and roots are updated, without running the algorithm from the beginning.

The DIFT algorithm takes time proportional to the number of pixels whose optimum values need to be updated.

Differential IFT

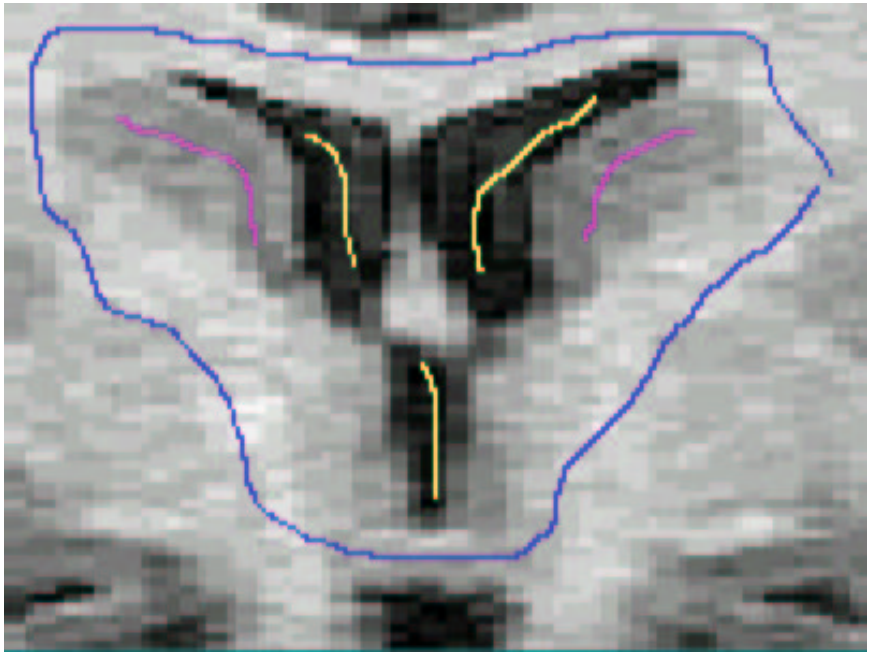


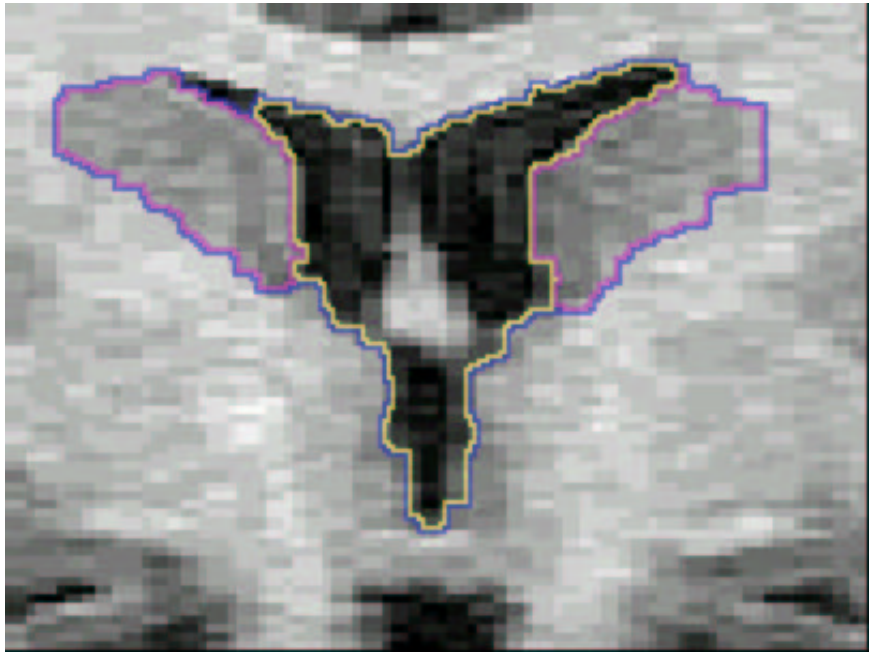
An optimum forest with trees T_a, T_b, and T_c rooted at pixels a, b, and c (left). T_a is marked for removal and d is added as seed (center). The dispute involves b, c, and d, where b and c are represented by pixels of T_b and T_c along the bold and dashed lines (i.e. frontier pixels between T_a and these trees). The new optimum forest (right).

3D interactive segmentation

- The DIFT algorithm has been used for multiple-object interactive 3D segmentation of MR-brain images using the watershed and fuzzy-connected approaches.
- It has provided efficiency gains from 10 to 17 with respect to our IFT implementations of these techniques.
- It has reduced the user's waiting time from 19–36 seconds to 2–3 seconds to visualize in 3D the results of each iteration, using data sets of sizes 5-9Mvoxels and an 1.1GHz Athlon PC with 384MB RAM.

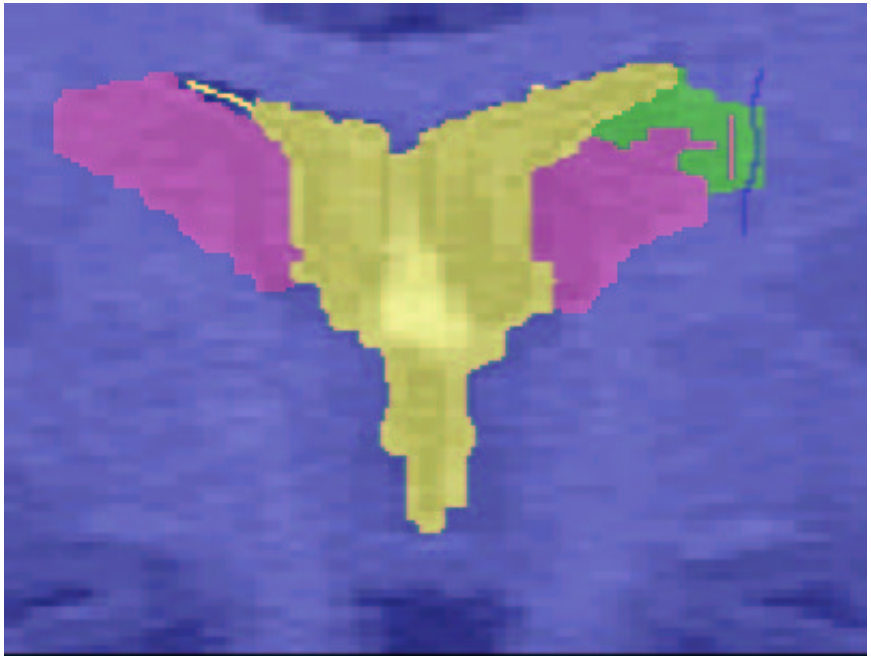
2D example of DIFT-watershed

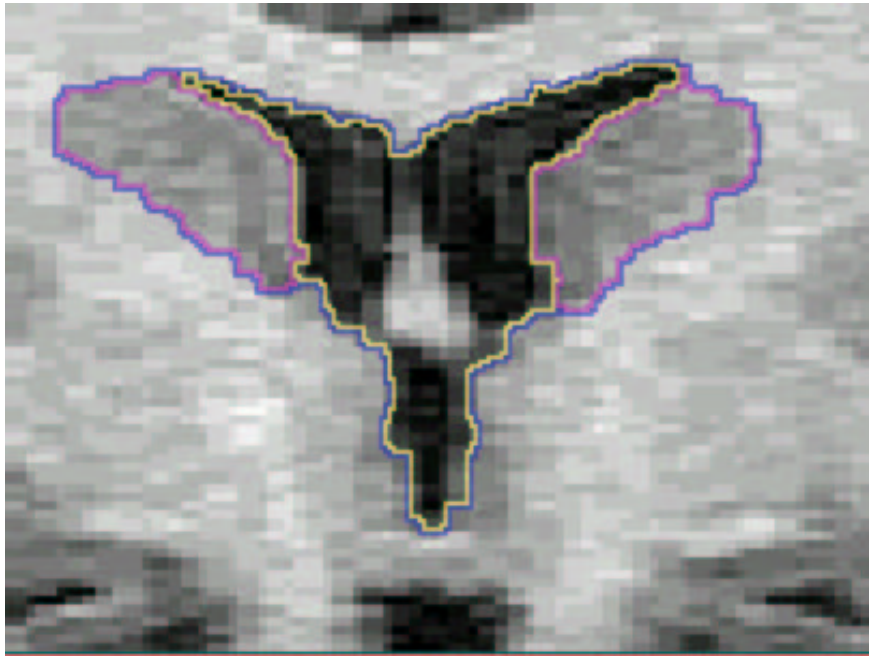




Seed selection for ventricles and caudate nuclei (left). First result shows part of the ventricles missing and part of the background as caudate nuclei (right).

2D example of DIFT-watershed

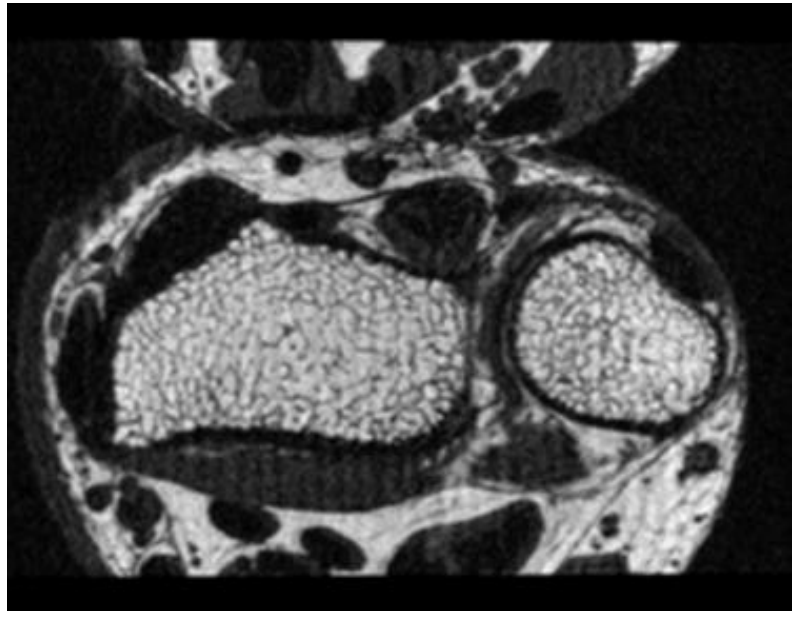




Seed selection for correction. The green region shows a single tree marked for removal (left). Corrected segmentation (right).

Segmentation from the cost map C

In some situations, objects are darker/brighter than their nearby background.



An MR image of a wrist where the bigger bone is the object of interest for segmentation.

Cost function for the wrist

First, suppose that the object was darker than its nearby background.

- $f_{peak}(\pi)$ assigns to π the maximum pixel intensity along it (i.e. once π starts in a certain gray level, its cost can never go down).
- If a single seed was selected inside the object, the resulting cost map using f_{peak} would be darker inside and brighter outside it.
- Then, the object could be segmented by simple thresholding.

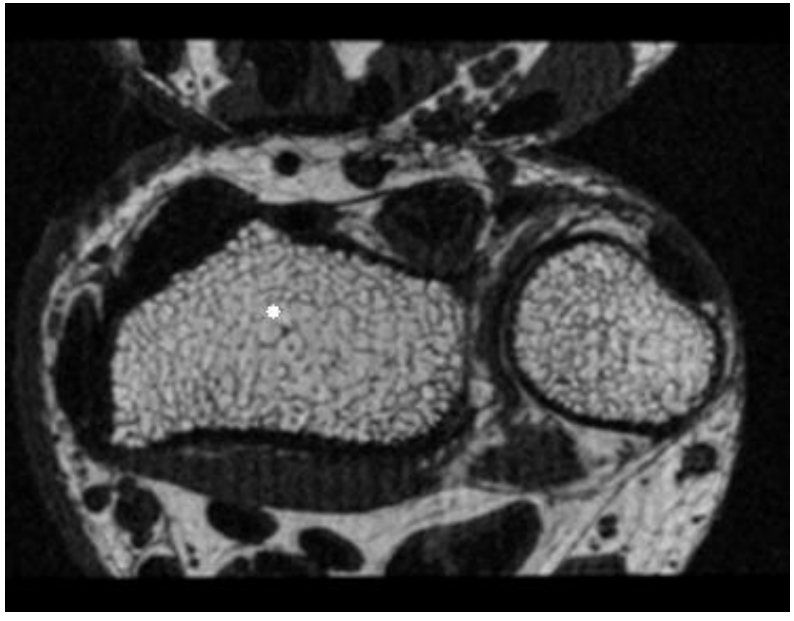
Cost function for the wrist

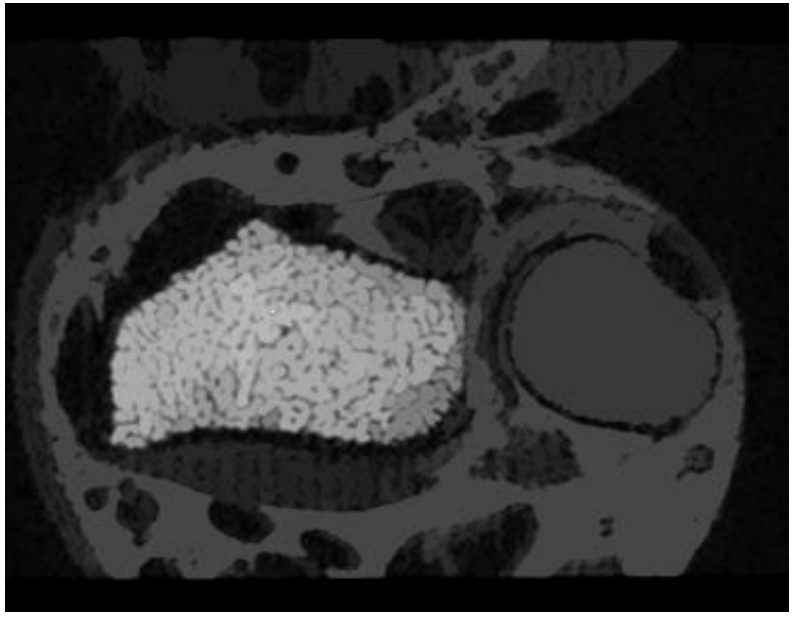
The dual operation applies to the case of the wrist bone and results from the complement of the cost map obtained by using a cost function

$$\begin{aligned} \bar{f}_{peak}(\langle t \rangle) &= K - I(t), \text{if } t \in \mathcal{S}, \text{ and } +\infty \text{ otherwise.} \\ \bar{f}_{peak}(\pi \cdot \langle s, t \rangle) &= \max\{\bar{f}_{peak}(\pi), K - I(t)\}, \end{aligned}$$

where K is the maximum of I.

Bone of the wrist

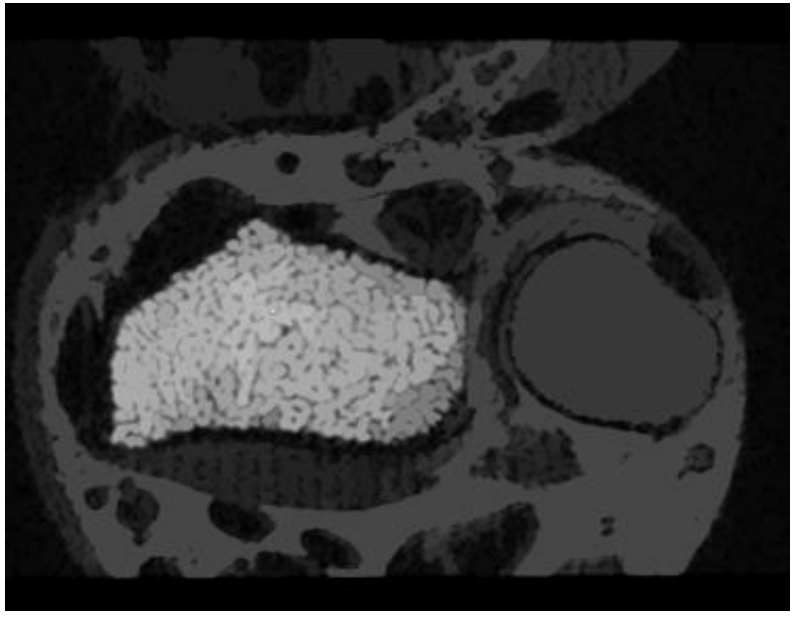


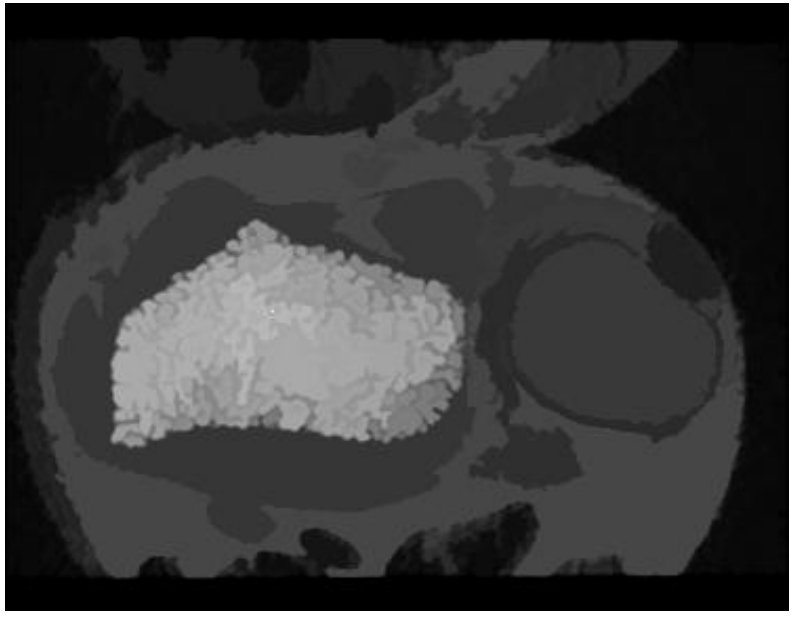


Seed selection (left) and the complement of the cost map obtained by using $\bar{f}_{peak}$ cost function (right).

Bone of the wrist

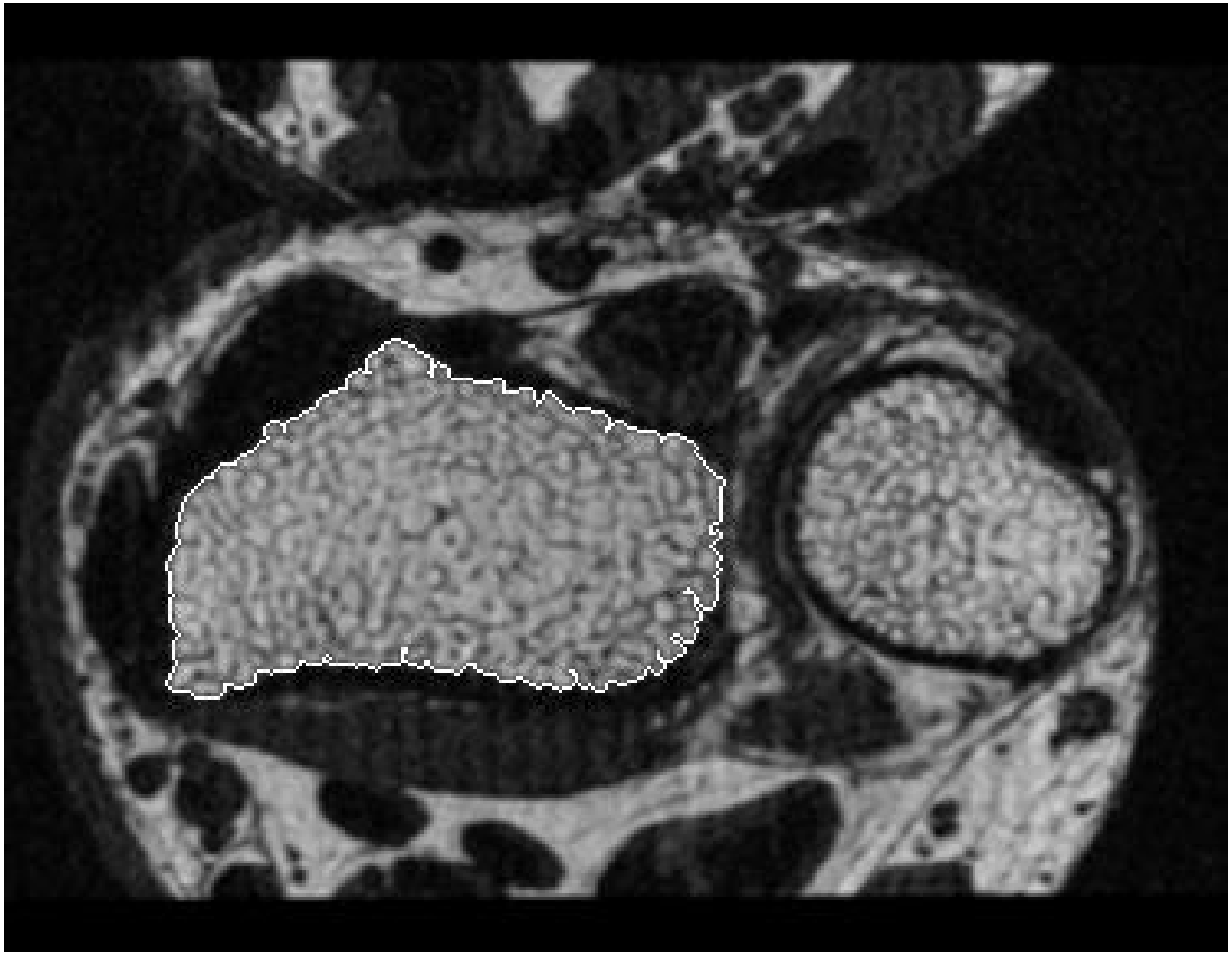
The holes within the bone (left) can be closed if we apply the same strategy using f_{peak} cost function and one pixel of the image border as seed (right).





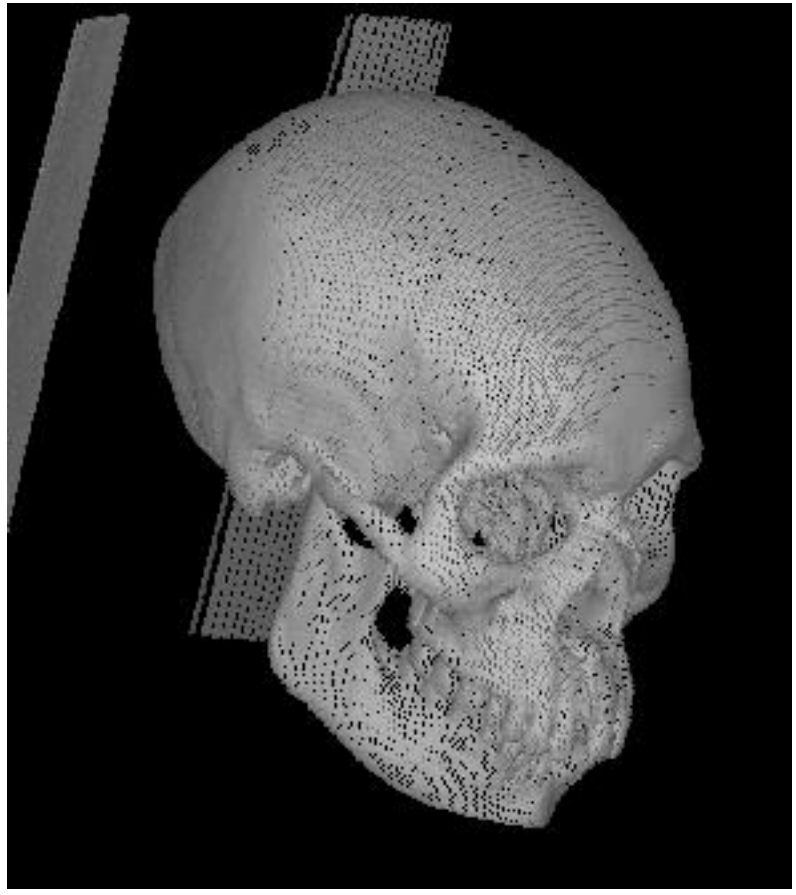
Bone of the wrist

The bone is segmented by thresholding the last image.



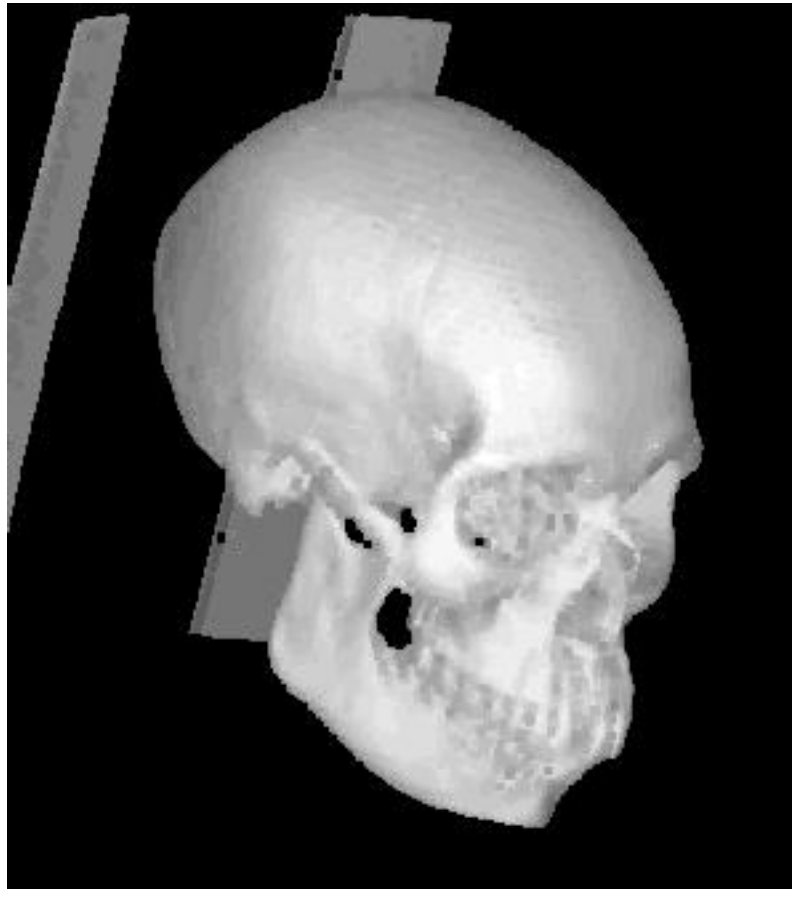
Filtering by reconstruction

It is known that holes may appear in 3D renditions whenever we use one-voxel to one-pixel projection.

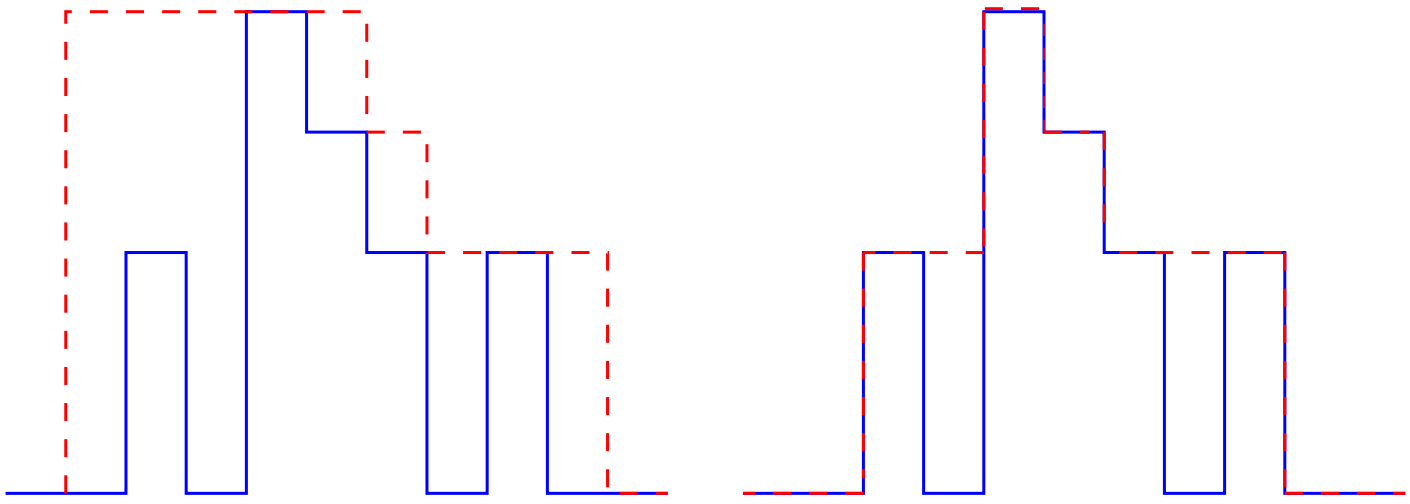


Filtering by reconstruction

A simple morphological closing operation using a disk of radius $\sqrt{2}$ (Euclidean adjacency) as structuring element closes the holes, but also looses details.



Cost function for the skull



(A) (B)

(A) The original image in blue and the closed image in (red). The closed image is taken as a handicap cost image in the f_{peak} function.

(B) The resulting cost map is shown in red.

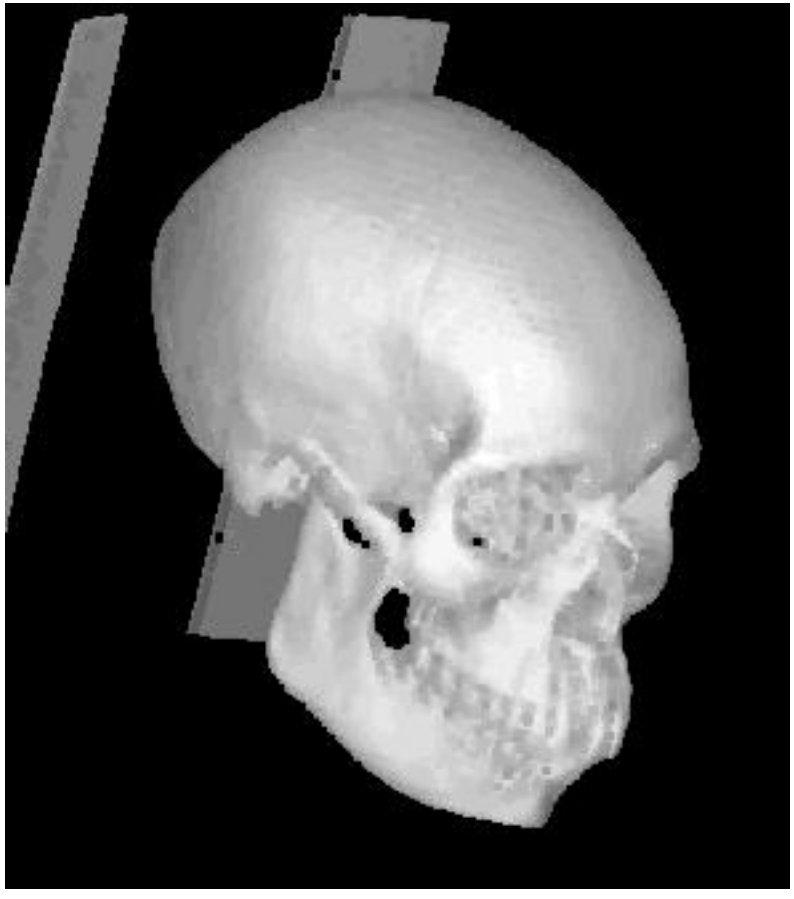
Cost function for the skull

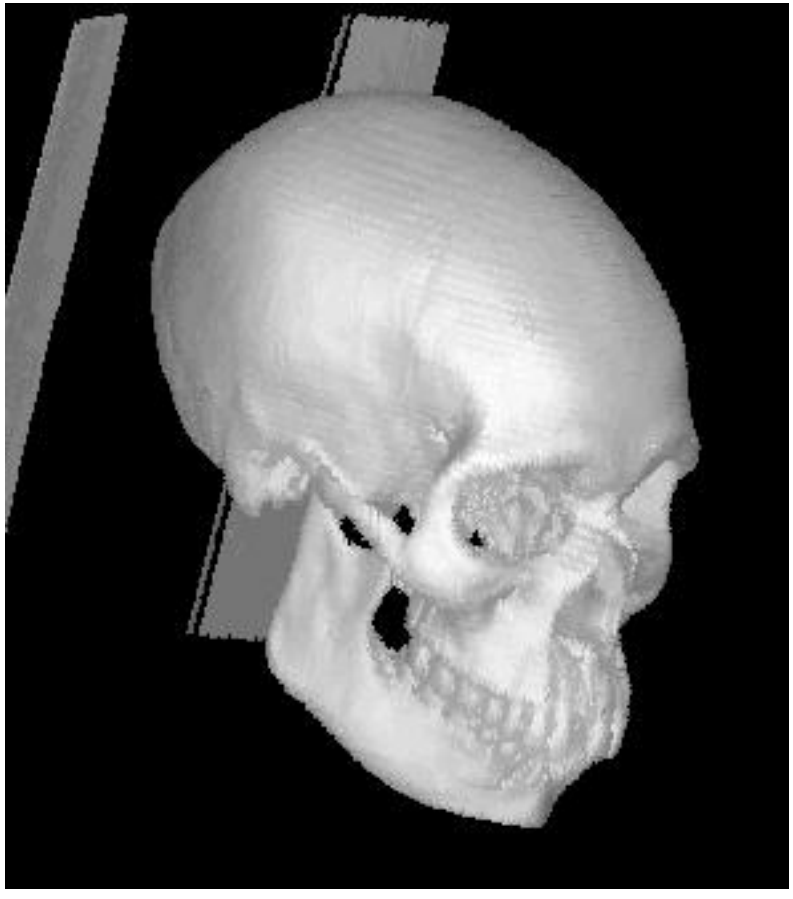
The cost function becomes a simple variation of f_{peak}.

$$\begin{aligned} f_{srec}(\langle t \rangle) &= h(t), \\ f_{srec}(\pi \cdot \langle s,t \rangle) &= \max\{f_{srec}(\pi), I(t)\}, \end{aligned}$$

where $h(t) \geq I(t)$, for all $t \in \mathcal{I}$. The cost map C is the result of the morphological superior reconstruction of I from h.

Filtering by reconstruction



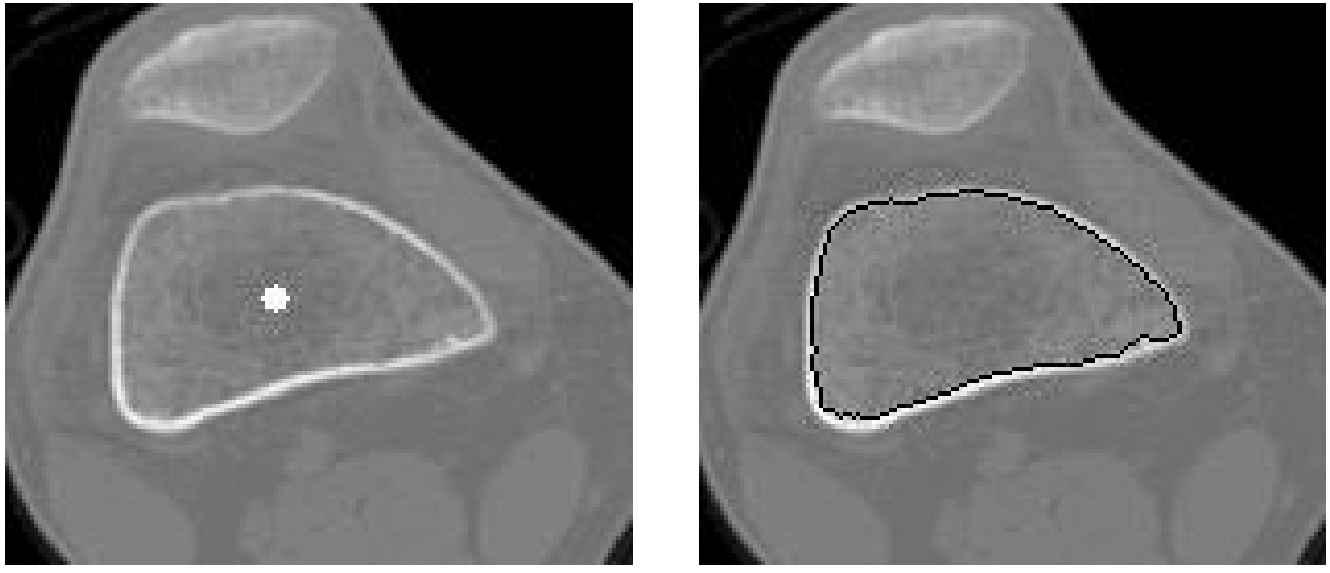


The operation is called closing by reconstruction.

Filtering by reconstruction

- Function f_{srec} has also a dual function, which provides in C the inferior reconstruction of I from h, when $h(t) \leq I(t)$ for all $t \in \mathcal{I}$.
- Together they provide many other image operators, such as area closing/opening,leveling, closing of basins, opening of domes, h-basins, h-domes, etc.
- The root map R obtained from f_{srec} function is the catchment basins of the watershed transform of C from a grayscale marker h, when $I(t) < h(t)$ for all $t \in \mathcal{I}$.
- If we use seed pixels, the root map R is the catchment basins of the watershed from markers, and marker imposition is obtained whenever $h(t) < I(t)$ for LIFO policy and $h(t) \leq I(t)$ for FIFO policy.

Segmentation by local reconstruction



A CT image of a knee where one seed t is selected inside the femur with handicap $h(t) > I(t)$, but less than the brightness of the femur's border (left). The interior of the bone can be extracted from the root map R (right).

Local superior reconstruction

The local superior reconstruction is a variant which fills up one or more basins, selected by a seed set $\mathcal{S}$, up to levels specified by given handicaps $h(t) > I(t)$ for $t \in \mathcal{S}$. Its cost function is

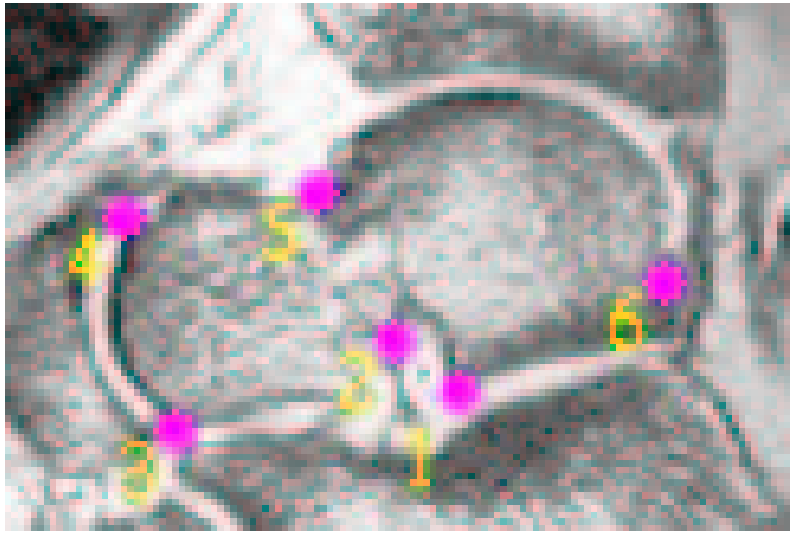
$$f_{lsrec}(\langle t \rangle) = h(t), \text{ if } t \in \mathcal{S}, \text{ and } +\infty \text{ otherwise,}$$

$$f_{lsrec}(\pi \cdot \langle s,t \rangle) = \begin{cases} f_{lrec}(\pi), & \text{if } f_{lrec}(\pi) > I(t), \\ +\infty, & \text{otherwise.} \end{cases}$$

It has been used for image filtering by area closing and image segmentation.

Boundary tracking

Consider the problem of finding an optimum curve that is constrained to pass through a given sequence $\langle \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k \rangle$ of k landmarks (pixel sets) on the object's boundary, in that order, starting in $\mathcal{T}_1$ and ending in $\mathcal{T}_k$.



MR-image where the talus is the object of interest.

Boundary tracking

- If f is MI, the curve consists of $k-1$ segments $\pi_1, \pi_2, \ldots, \pi_{k-1}$, where π_i is an f-optimum path connecting $\mathcal{T}_i$ to $\mathcal{T}_{i+1}$.
- The problem can be solved by $k-1$ executions of the IFT, where each stage can be terminated as soon as the last pixel of the target set $\mathcal{T}_{i+1}$ is removed from the queue.
- The curve is obtained from the predecessor maps $P_{k-1}, P_{k-2}, \ldots, P_1$.
- The curve can be closed by making $T_1 = T_k$ and computing an extra path from the last to the first pixel of the opened curve.

Cost function for the talus

$$f_{btrack}(\langle t \rangle) = \begin{cases} 0, & \text{if } i = 1 \text{ and } t \in T_1, \\ C_i(t), & \text{if } i > 1 \text{ and } t \in T_i, \\ +\infty, & \text{otherwise.} \end{cases}$$

$$f_{btrack}(\pi \cdot \langle s,t \rangle) = f_{btrack}(\pi) + w(s,t),$$

where $w(s,t) = K - \max\{G(s,t) \cdot \eta(s,t), 0\}$, where $G(s,t)$ is a gradient vector estimated at the midpoint of arc (s,t); $a(s,t)$ is the arc (s,t) rotated 90 degrees counter-clockwise; and K is an upper bound for $|G(s,t) \cdot a(s,t)|$.

Boundary orientation

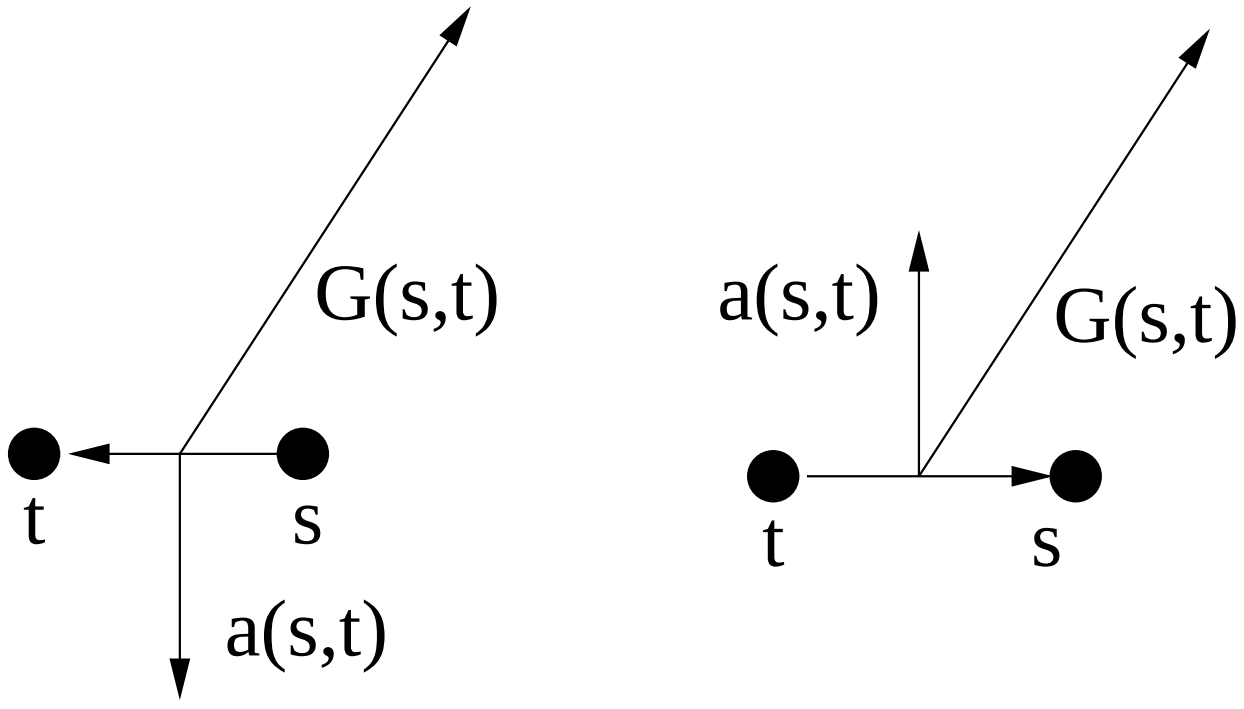
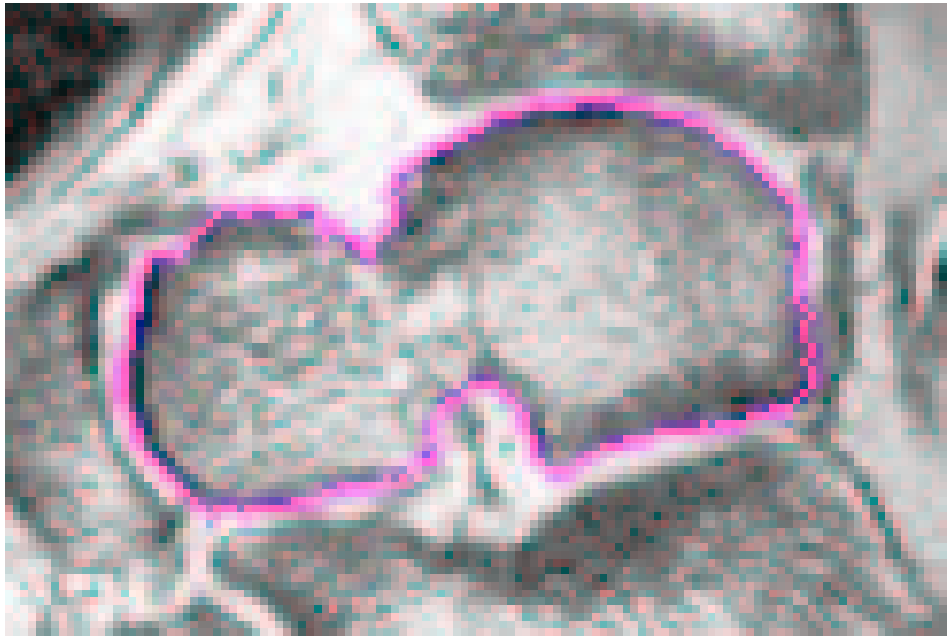


Figure on the left shows when the cost assignment will not favor (s, t) orientation and the other way around is shown on the right.

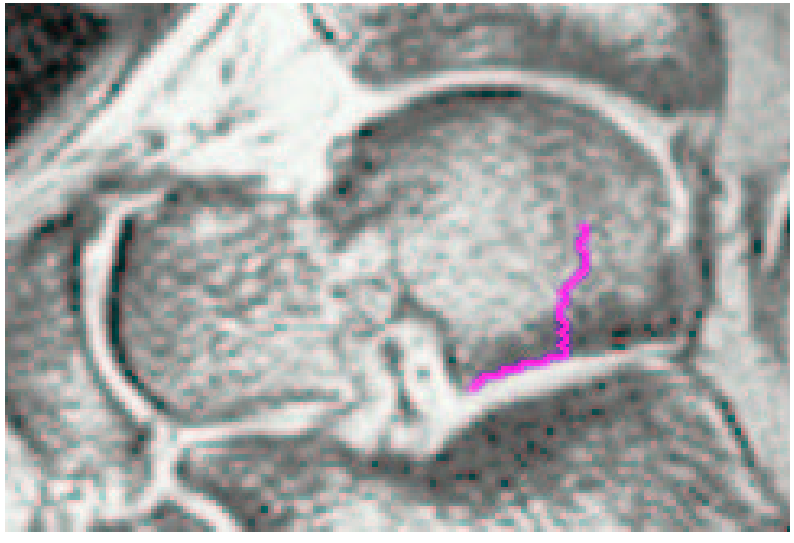
Optimum boundary of the talus

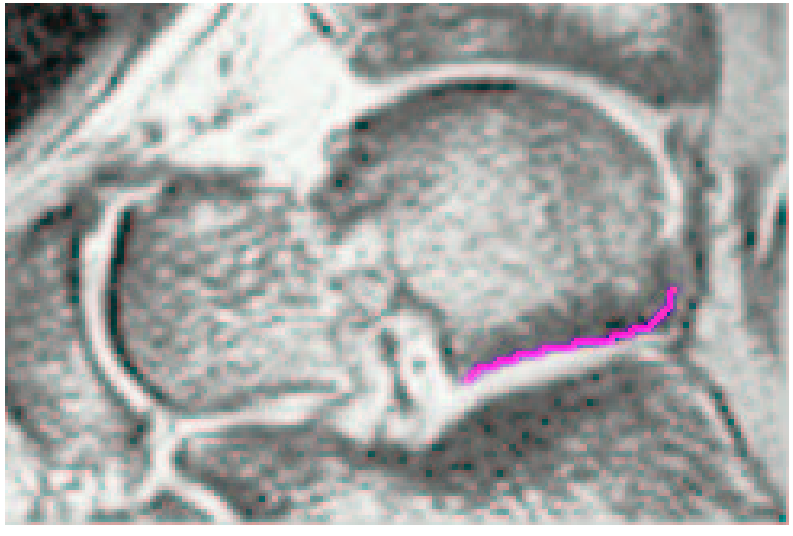


Result of the IFTs with FIFO policy and 8-neighborhood.

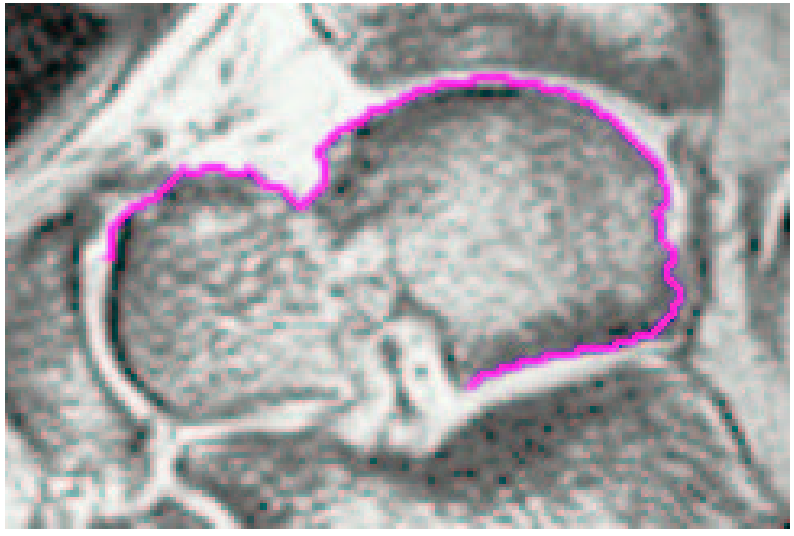
Live wire

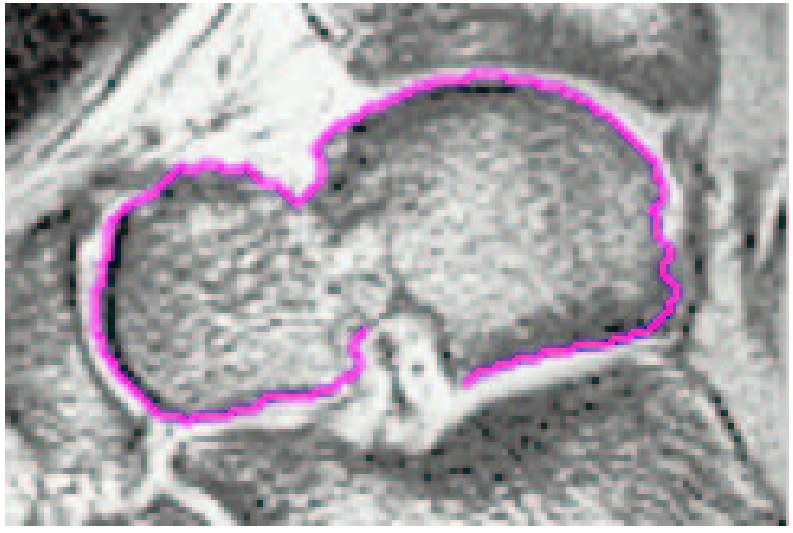
Live wire becomes a particular case of this optimum formulation for boundary tracking, where the sets T_i are chosen by the user during image segmentation.

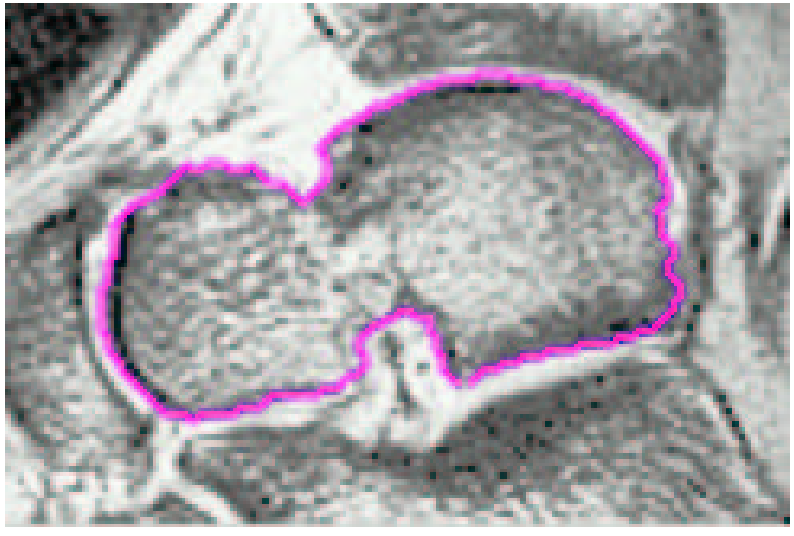




Live wire

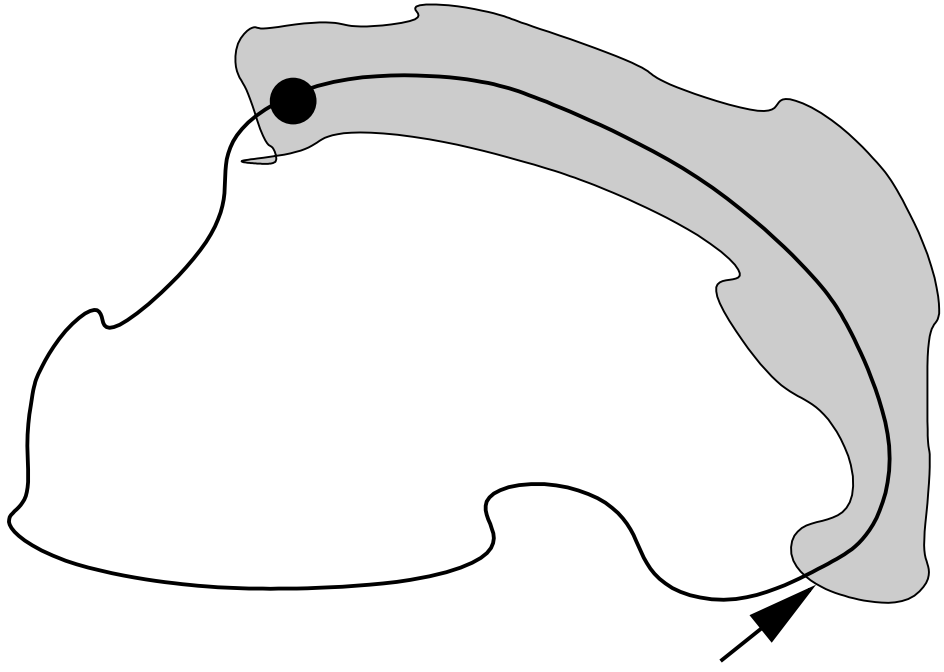






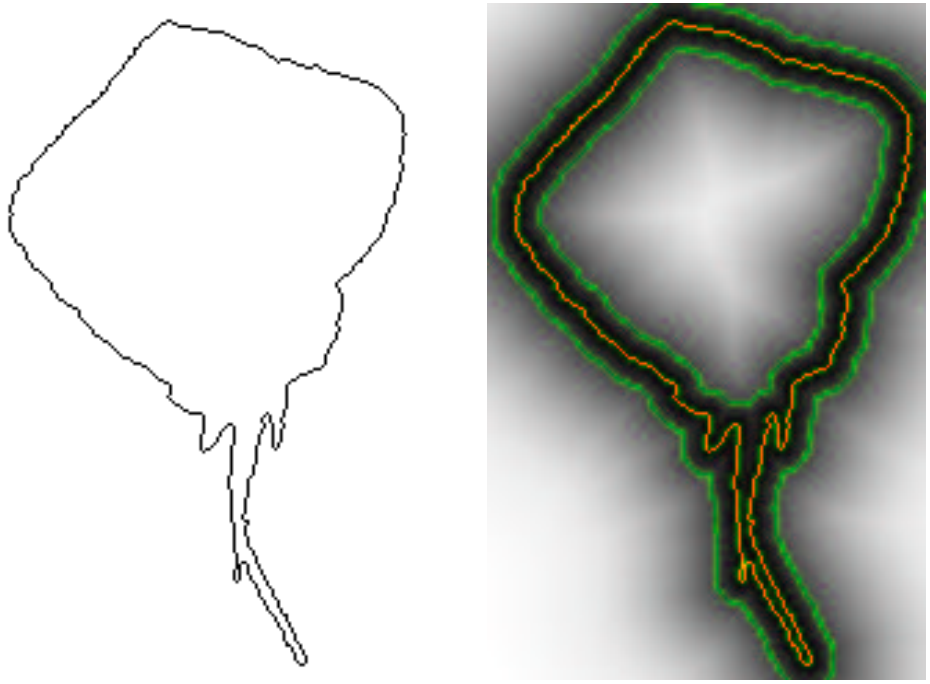
Live-wire-on-the-fly

Note that its most recent variant, called live-wire-on-the-fly, is another example of IFT being computed in a differential way.



Euclidean distance transform (EDT)

The EDT of a given seed set $\mathcal{S} \in \mathcal{I}$ assigns to each image pixel $t \in \mathcal{I}$ a value $C(t)$, which is the minimum Euclidean distance from t to $\mathcal{S}$.



A fish contour (left) and its EDT (right).

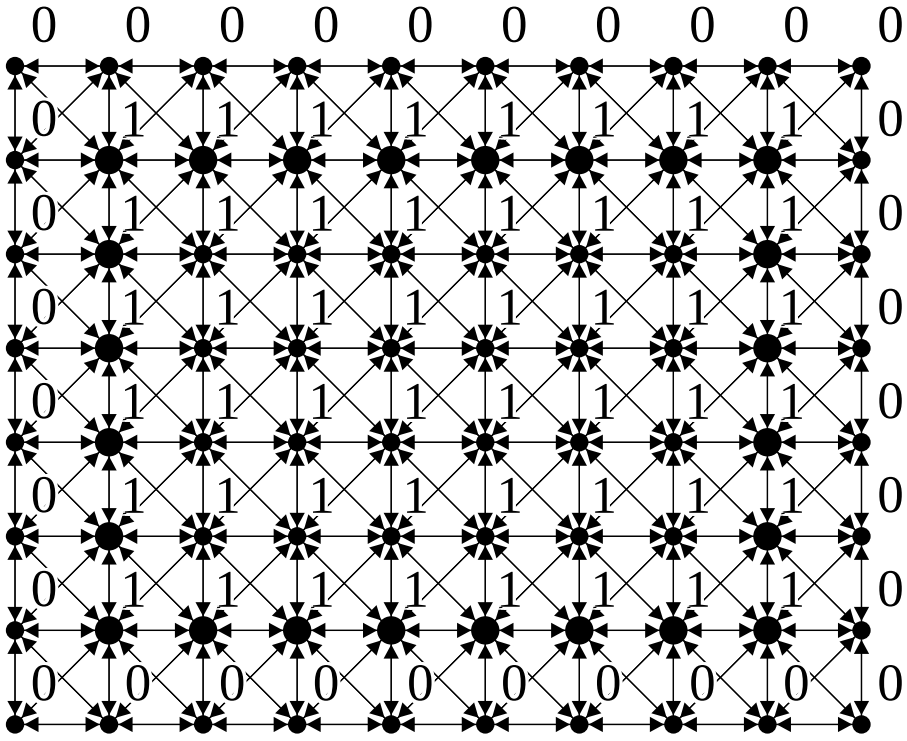
Cost function for the EDT

$$
\begin{aligned}
f^{\mathcal{S}}_{euc}(\langle t \rangle) &= 0, \text{ if } t \in \mathcal{S}, \text{ and } +\infty \text{ otherwise.} \\
f^{\mathcal{S}}_{euc}(\pi \cdot \langle s, t \rangle) &= (x_t - x_r)^2 + (y_t - y_r)^2,
\end{aligned}
$$

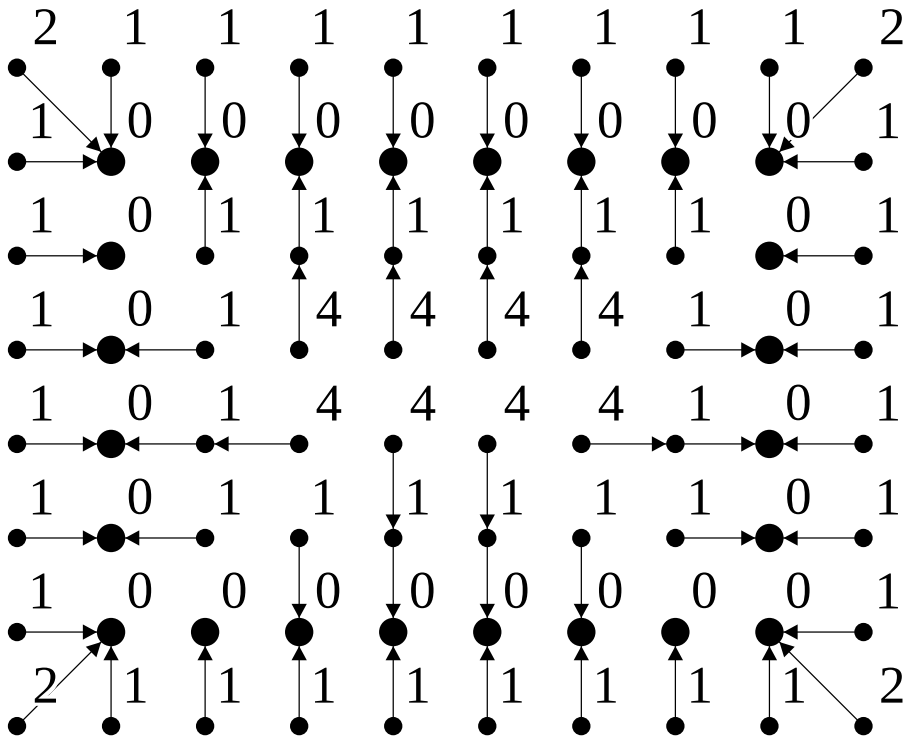
where r is the initial pixel of π (or better, $R(s)$ during the execution of the algorithm). The EDT can be obtained by computing the IFT with FIFO policy and the squared root of its resulting cost map C.

Example of the EDT

Consider a directed graph whose nodes are the image pixels of a binary image $\mathbf{I}$ and whose arcs are defined by an 8-neighborhood relation. The seed set $\mathcal{S}$ is composed by contour pixels.



The Euclidean distance map



The cost map C shows the squared Euclidean distance values.

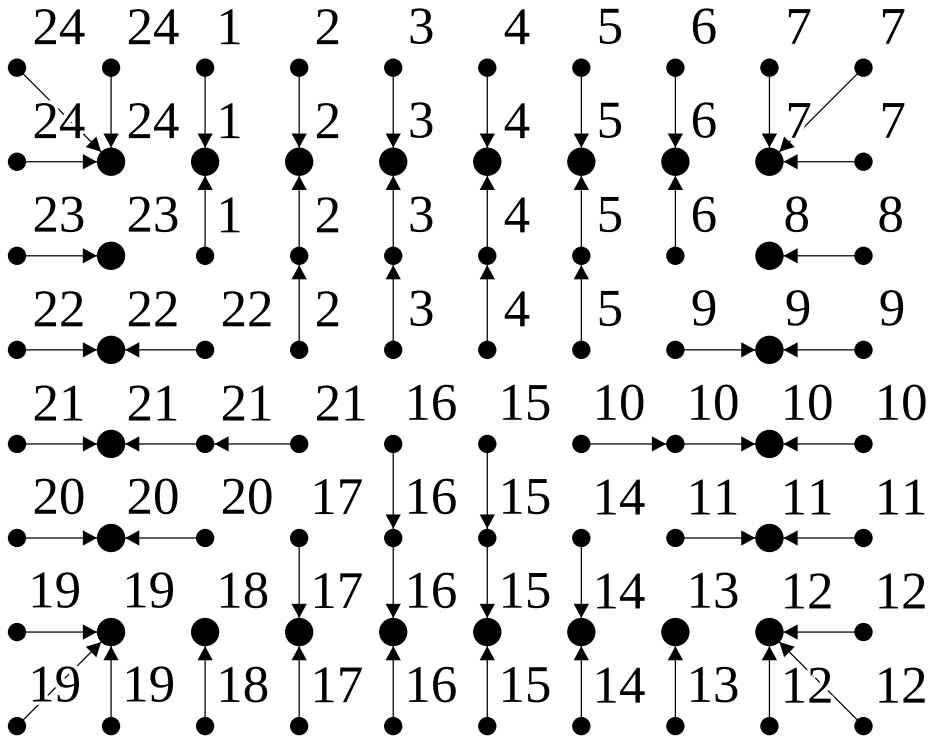
C can be used to compute the multiscale fractal dimension of $\mathcal{S}$.

Discrete Voronoi regions

- The influence zones of the contour pixels define discrete Voronoi regions in the root map R.
- Each contour pixel $t \in \mathcal{S}$ can be associated with a subsequent integer number $L(t)$ from 1 to N, while circumscribing the contour.
- A label map L may be created from the root map R by setting $L(t) \leftarrow L(R(t))$, or the labels may be propagated during the algorithm.

The label map L can be used to compute multiscale skeletons and salience points along the contour.

Discrete Voronoi regions



The label map L shows the influence zone of each contour pixel.

Some observations

- Although $f_{euc}^{\mathcal{S}}$ is not smooth for some combinations of $\mathcal{S}$ and $\mathcal{A}$, 8-neighborhood seems to be enough for most practical situations.
- Even when $f_{euc}^{\mathcal{S}}$ is not smooth, and the IFT can not output exact distance values, 8-connected regions are essential to obtain one-pixel-wide and connected skeletons.

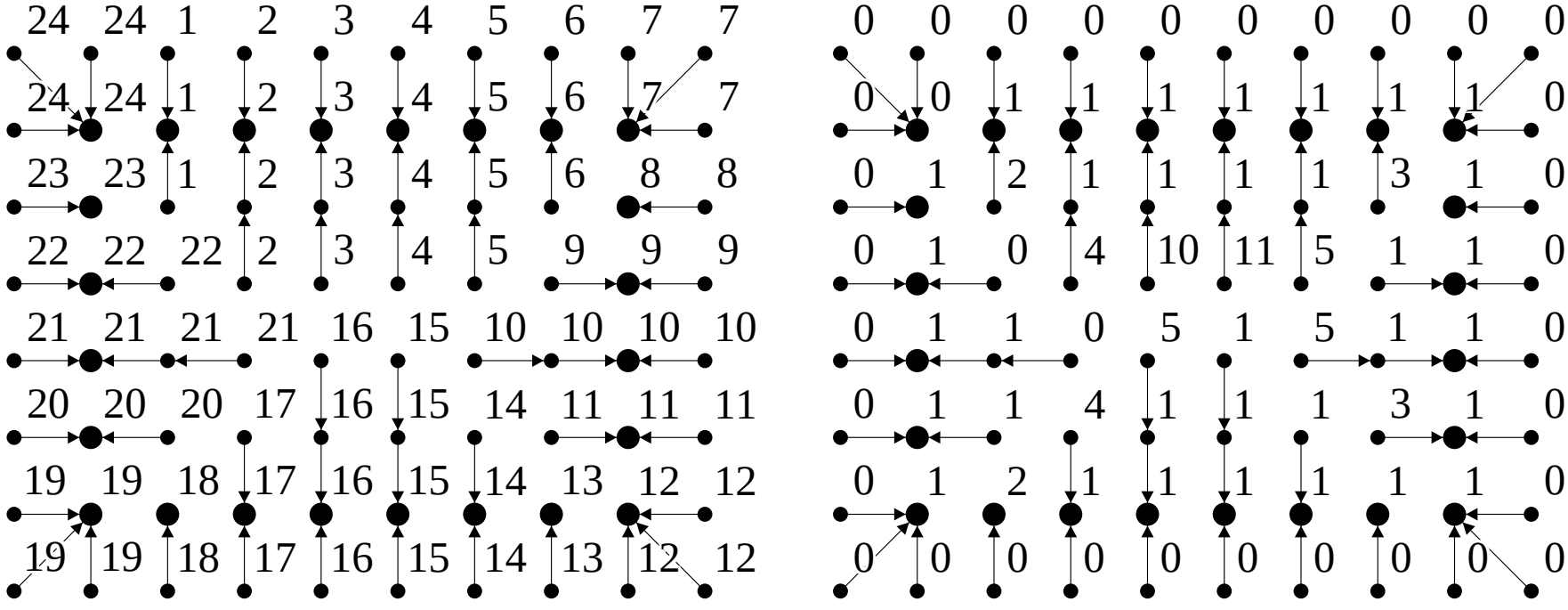
Multiscale skeletons

Internal and external multiscale skeletons of a close curve $\mathcal{S} \subset \mathcal{I}$ can be obtained from the label map L by assigning to each pixel $s \in \mathcal{I}$ a difference value

$$D(s) = \max_{\forall (s,t) \in \mathcal{A}} \{\min\{L(t) - L(s), N - (L(t) - L(s))\}\},$$

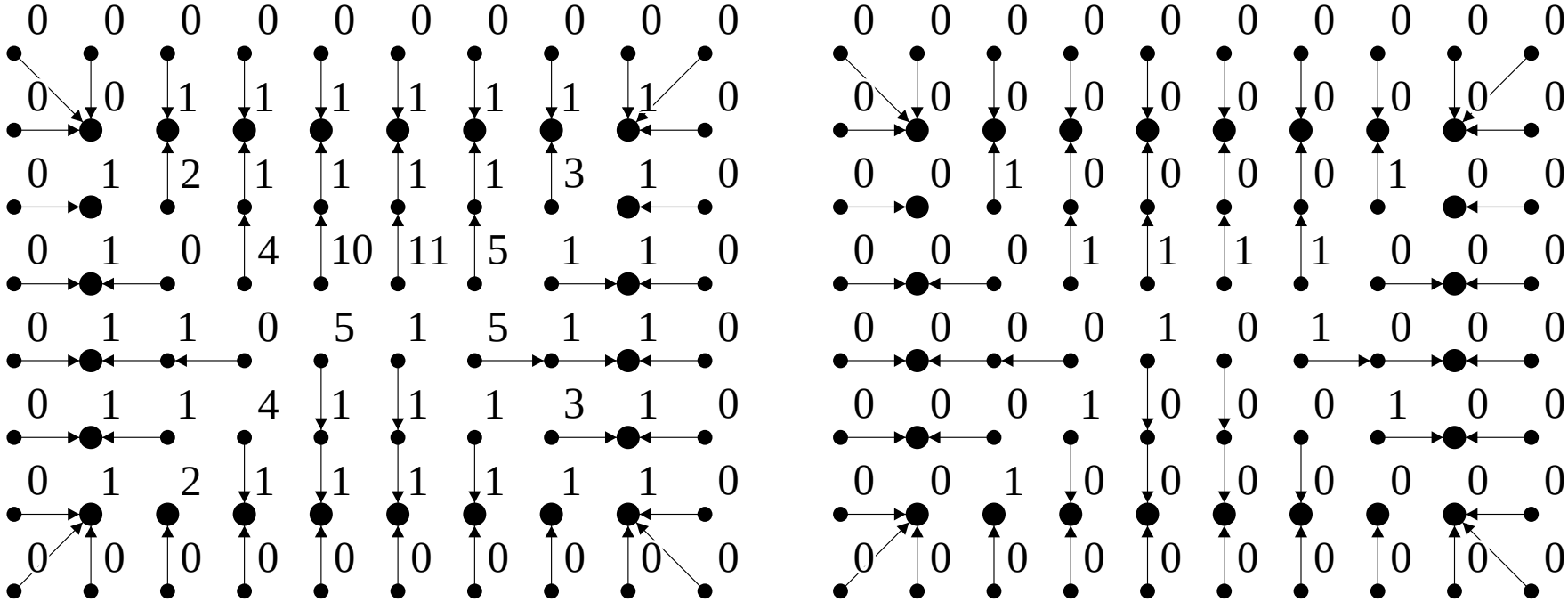
where N is the number of contour pixels and $\mathcal{A}$ is the 4-neighborhood relation.

Example of MS-skeletons



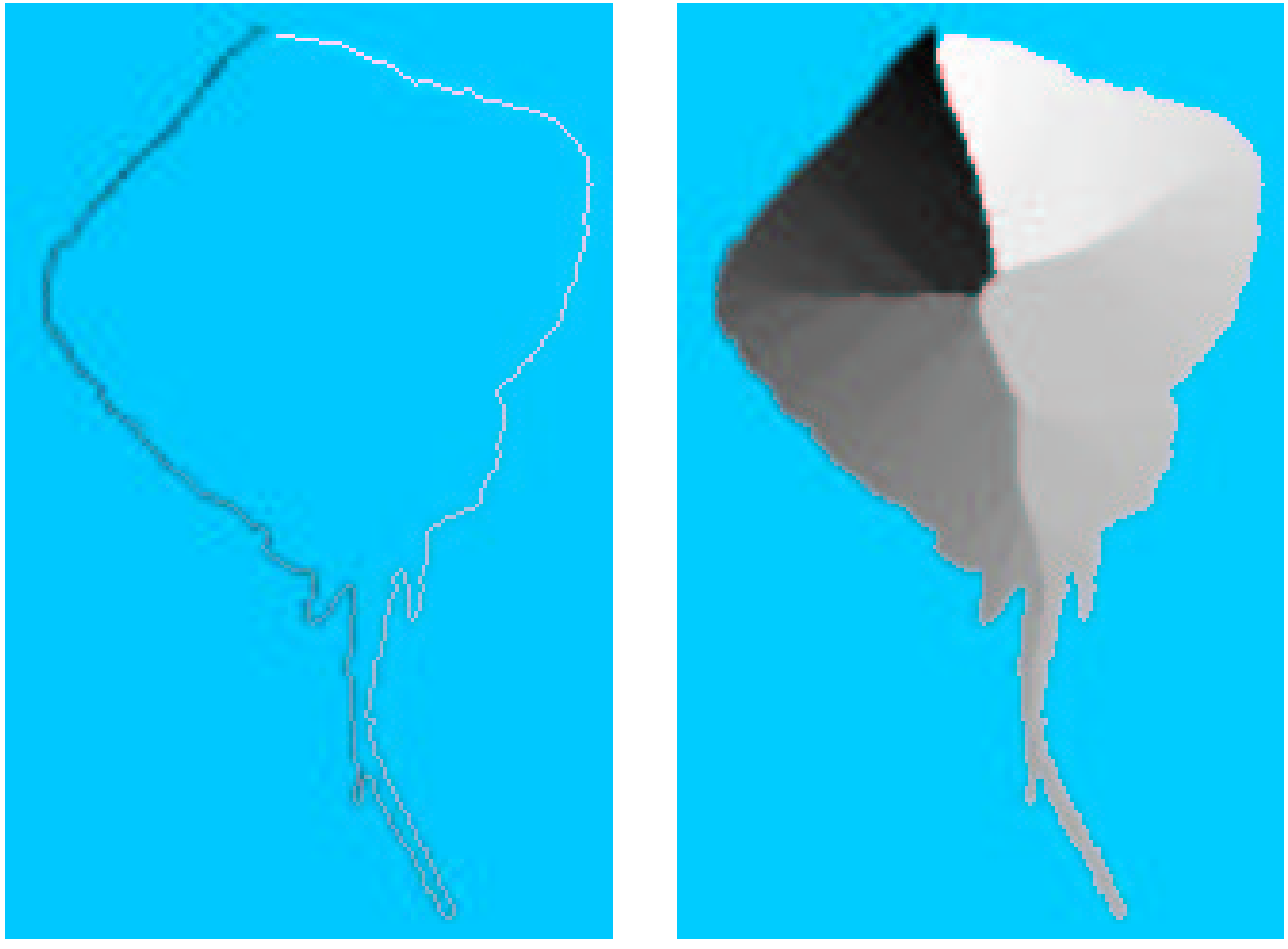
The label map L (left) and the skeleton scale map D inside the object only (right).

Example of MS-skeletons



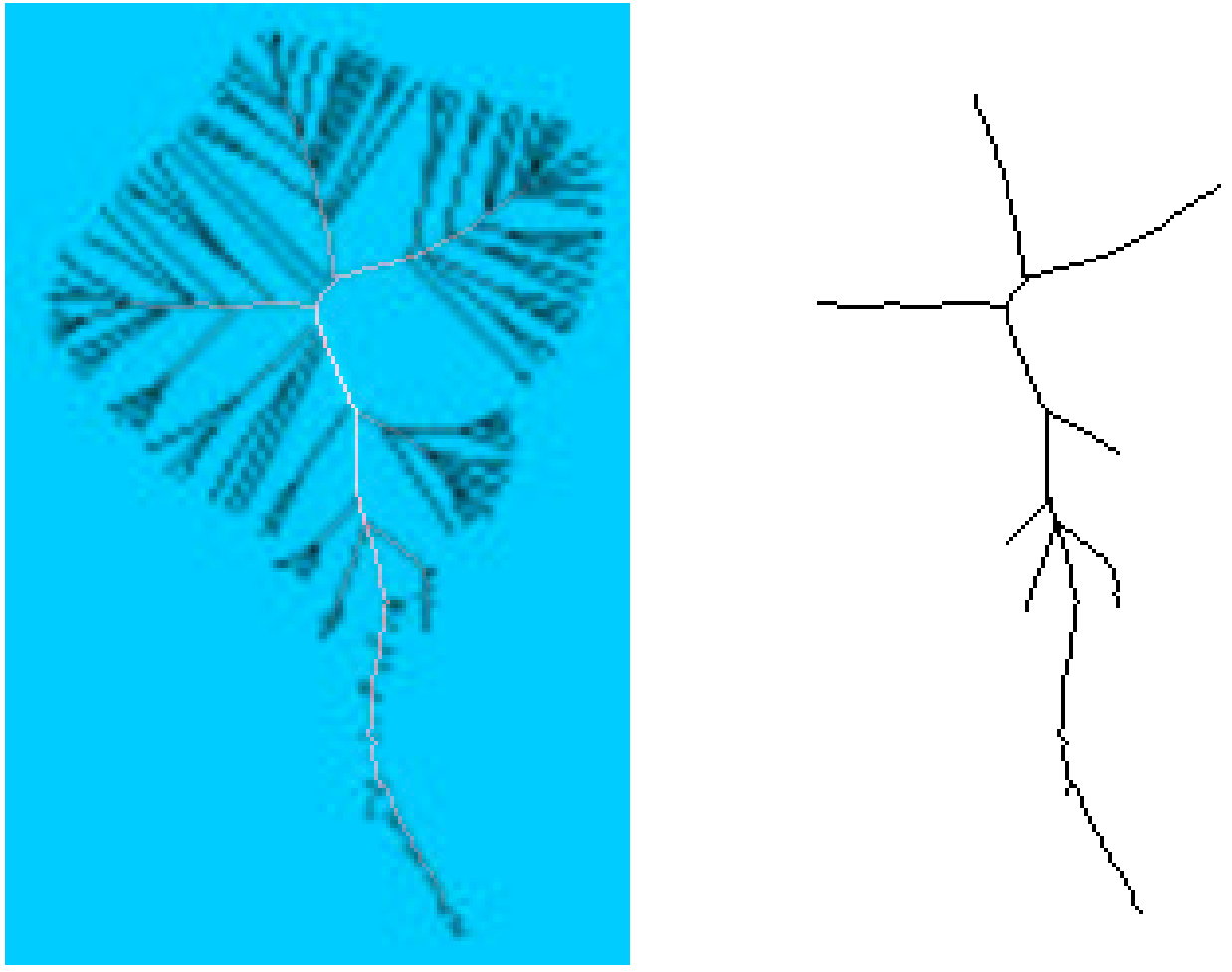
One-pixel wide and connected skeletons (right) can be obtained by thresholding D (left) at various scales.

MS-skeletons for the fish contour



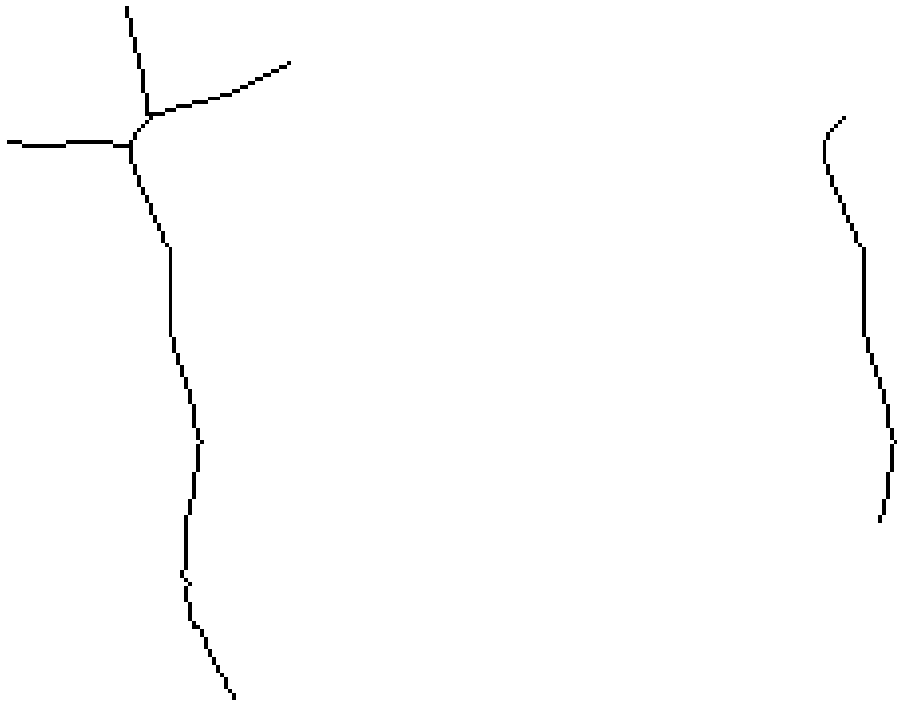
Labeled contour (left) and label map L (right).

Multiscale skeletons



MS-skeletons (left) and a skeleton obtained by thresholding (right).

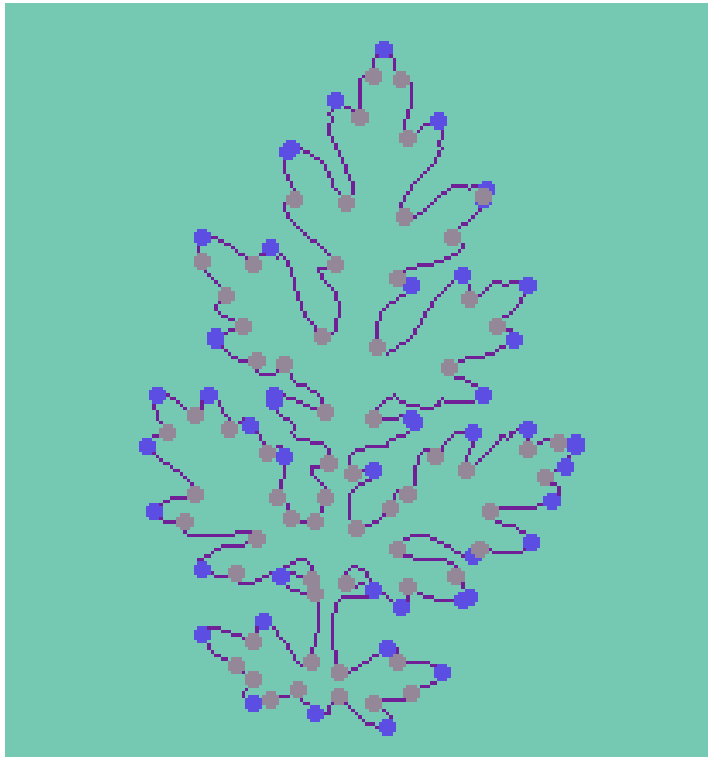
Multiscale skeletons



The skeletons become more simplified as the threshold increases.

Salience points

The salience points of a curve can be informally defined as its higher curvature points.

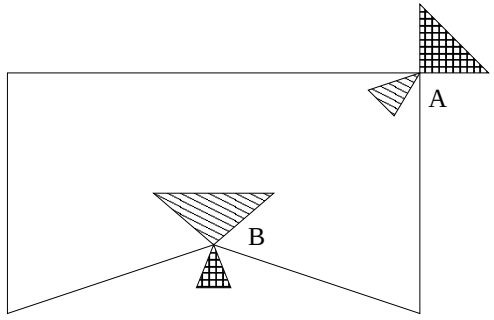


Convex points are shown in blue and concave points in red.

Salience values

Consider a small radius r forming a narrow band $\mathcal{B}$ of size $2r$ around a curve $\mathcal{S}$.

- Each salience point has two influence areas within $\mathcal{B}$, one outside and one inside $\mathcal{S}$.
- Convex points have higher influence areas outside than inside, and the other way around is true for concave points.



The salience value of a point is its highest influence area.

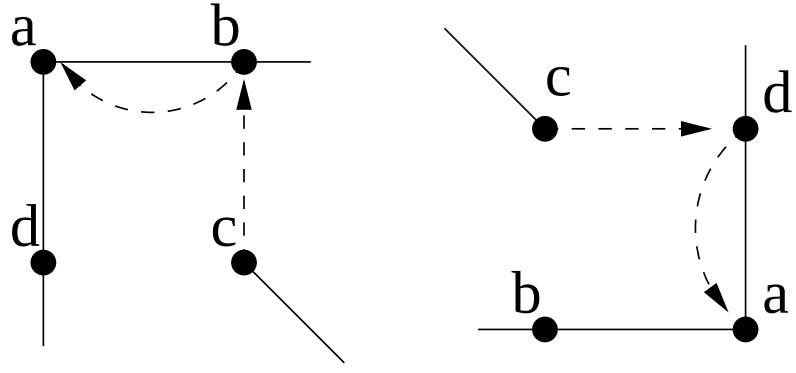
Detection of salience points

- The salience value A of a point relates to the aperture angle θ by the formula $A = \frac{\theta r^2}{2}$.
- A can be coomputed from the histogram of the label map L.
- Salience points can be detected by thresholding θ.
- Small values of r reduce cross-influence in this method from opposite parts of the contour which come close to each other.

The method works fine for skeletons, but not for contours which are usually more intricate shapes.

Detection of salience points on contours

Salience points of the contour can be related to salience points of its internal and external skeletons by means of the skeleton scale map D and root map R.



For a clockwise labeled contour: if $R(c) = b$, a is reached by skipping $\frac{D(c)}{2}$ pixels in anti-clockwise from b (left); and if $R(c) = d$, a is reached from d by skipping $\frac{D(c)}{2}$ pixels in clockwise (right). But, how do we know if the root of c is b or d?

Detection of salience points on contours

The skeleton scale value at a pixel $s \in \mathcal{I}$ is

$$D(s) \quad = \quad \max_{\forall (s,t) \in \mathcal{A}} \{\min\{L(t) - L(s), N - (L(t) - L(s))\}\},$$

where N is the number of contour pixels and $\mathcal{A}$ is the 4-neighborhood relation.

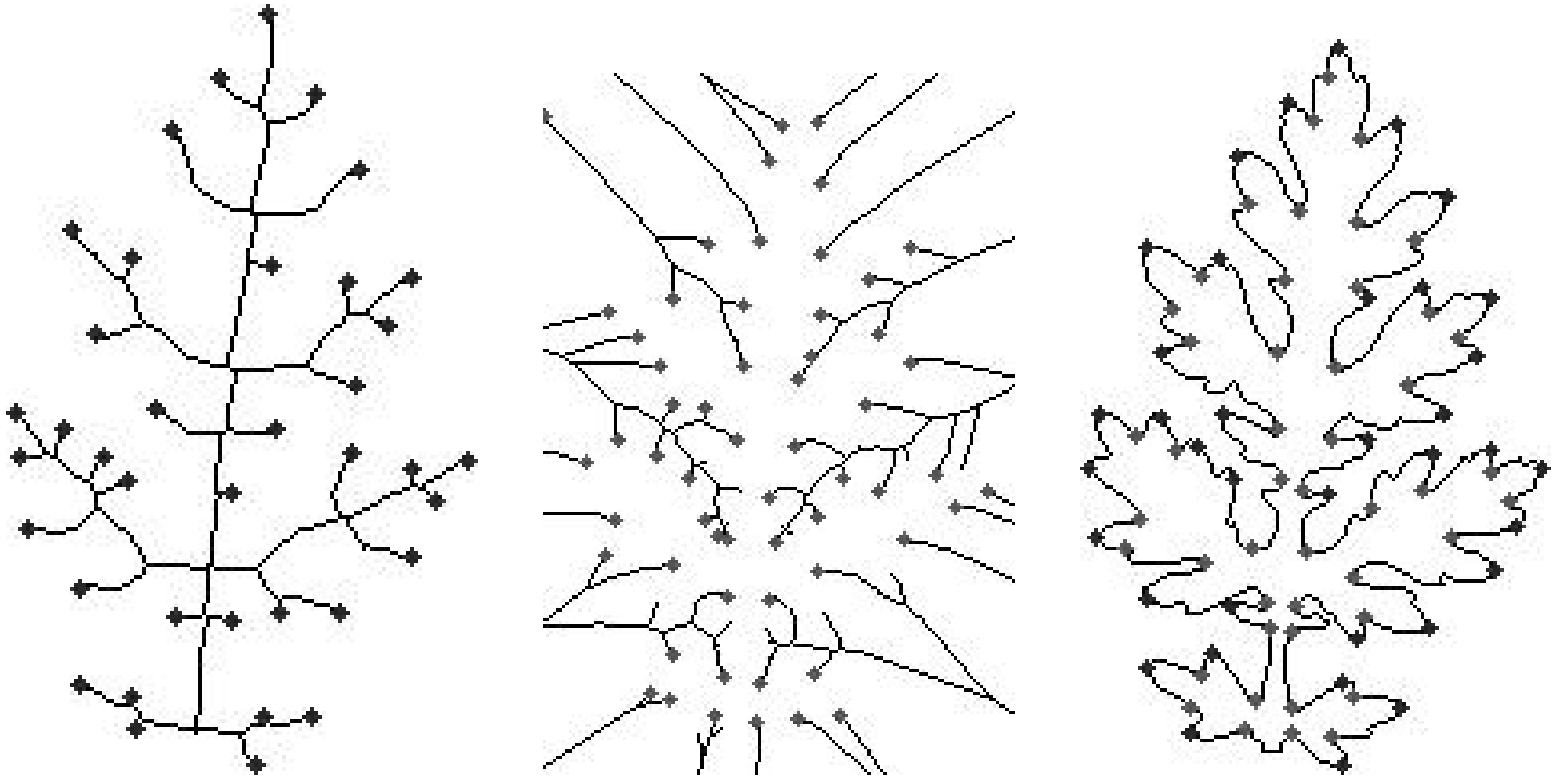
- The root of c will be b whenever $L(t) - L(s) > N - (L(t) - L(s))$, where $L(t) = L(d)$ and $L(s) = L(b)$.
- The root of c will be d whenever $L(t) - L(s) \leq N - (L(t) - L(s))$, where $L(t) = L(b)$ and $L(s) = L(d)$.

Signed ms-skeletons

The solution is to sign the ms-skeletons D.

1. For all pixels $s \in \mathcal{I}$, do
2. Set $\delta_{max} \leftarrow -\infty$.
3. For all pixels t, such that $(s,t) \in A$, do
4. Set $\delta \leftarrow \min\{L(t) - L(s), N - (L(t) - L(s))\}$ and $\sigma \leftarrow +1$.
5. If $\delta = [N - (L(t) - L(s))]$, then set $\sigma \leftarrow -1$.
6. If $\delta > \delta_{max}$, then set $\delta_{max} \leftarrow \delta$ and $sign \leftarrow \sigma$.
7. Set $D(s) \leftarrow sign \times \delta_{max}$.

Salience points of a contour

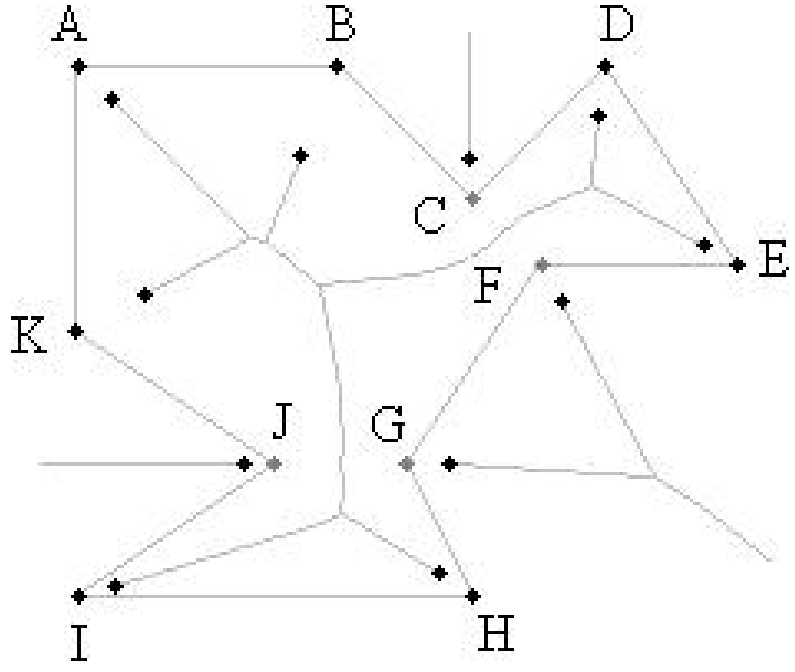


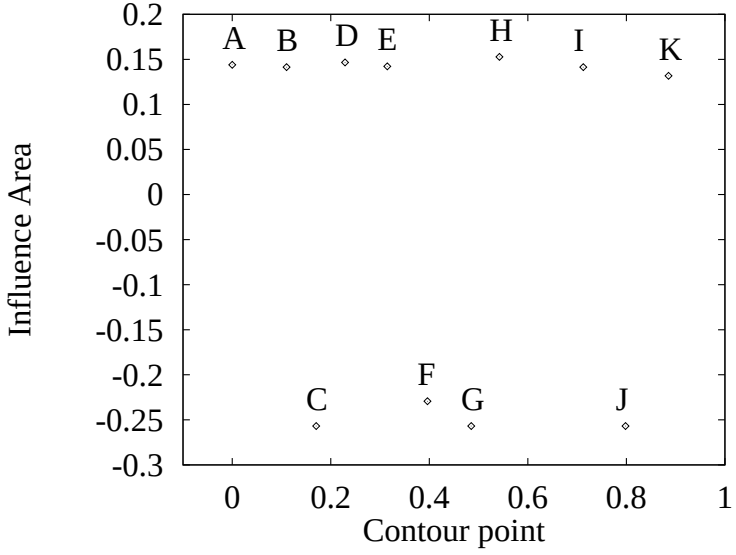
Salience points on the internal (left) and external (center) skeletons.

Salience points on the contour (right).

Contour salience descriptor

The contour salience descriptor consists of the relative position of each salience point along the contour with respect to a starting point, the salience values for these points, and a matching algorithm.





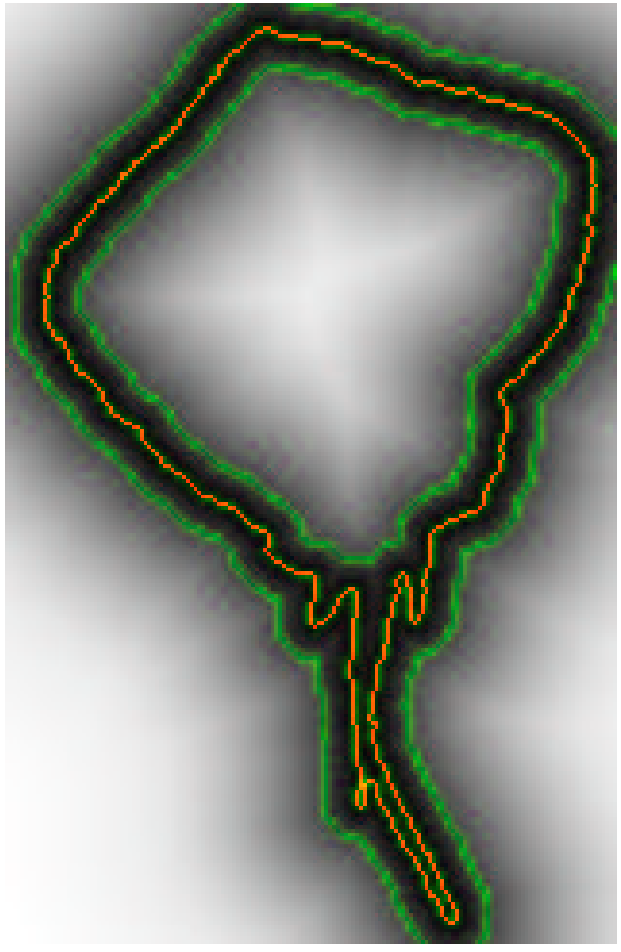
Fractal dimension

The Fractal dimension of a set $\mathcal{S}$ by Minkowski-Bouligand is a number F within $[0, 2]$.

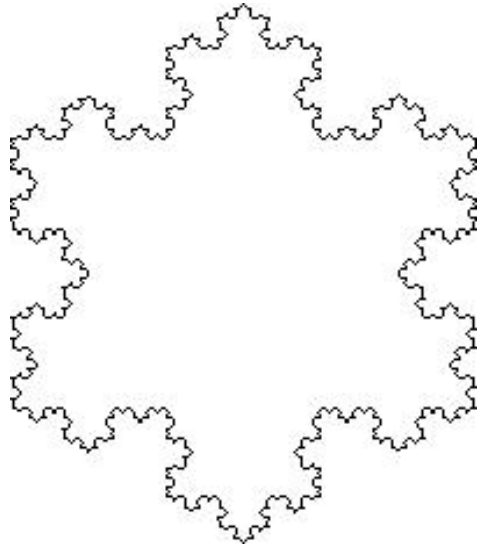
$$F = 2 - \lim_{r \to 0} \frac{\ln(A(r))}{\ln(r)},$$

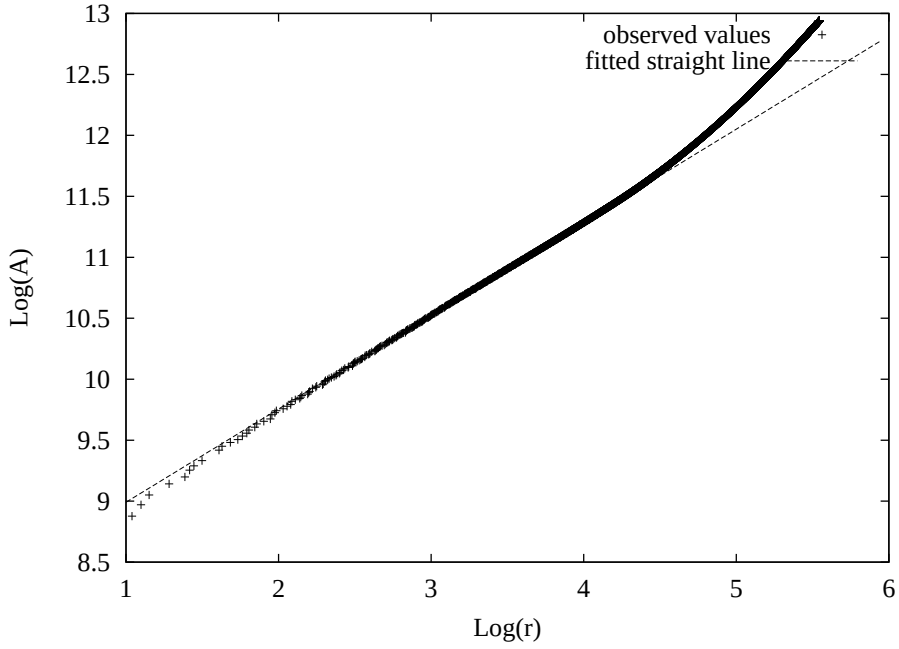
where $A(r)$ is the area of $\mathcal{S}$ dilated by a radius r. It represents the self-similarity of $\mathcal{S}$ for r close to 0. Note that, A is simply the cummulative histogram of the Euclidean distance map which can be obtained from the cost map C.

Euclidean distance map



Standard approach for computing F

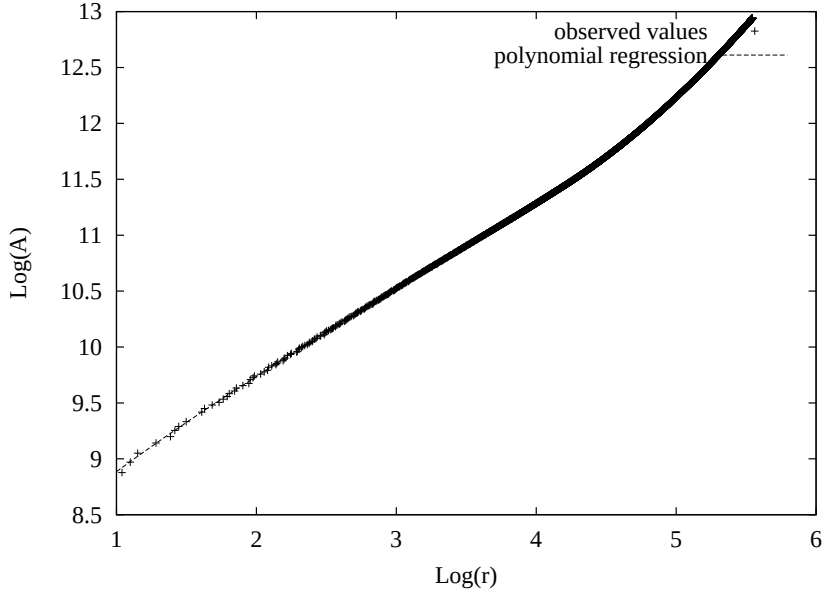


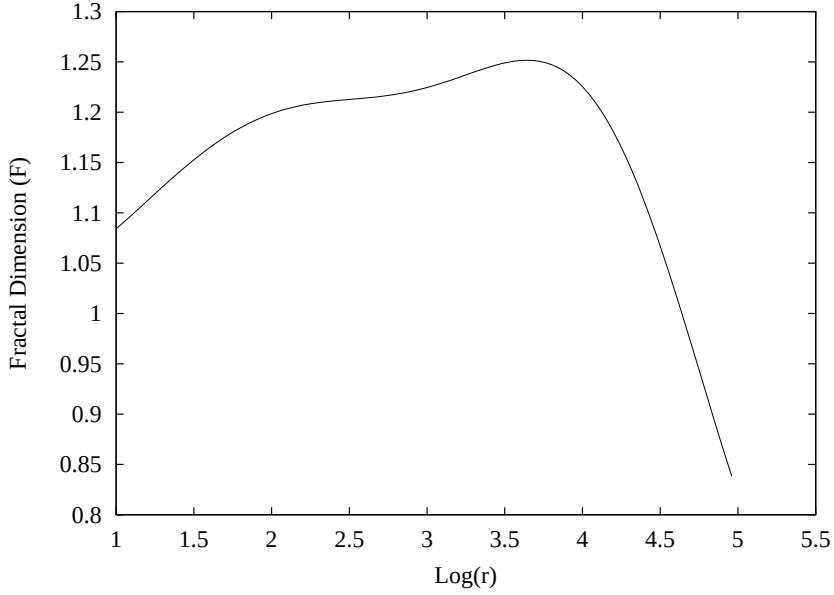


A shape similar to the Koch star, which has $F \approx 1.26$ (left). A line is fitted to the curve (right), and its first derivative is used to estimate F ($F \approx 1.23$).

Multiscale fractal dimension

We use polynomial regression for fitting (left) and F becomes a polynomial curve, called multiscale fractal dimension (right).





Note that, the maximum value of the curve is close to 1.26, providing much richer description of the self-similarity of $\mathcal{S}$.

Shape analysis

- Multiscale Fractal Dimention (MFD) and Contour Salience Descriptor (CSD) have been compared to Curvature Scale Space (CSS), Bean Angle Statistics (BAS), Moment Invariants (MIV), Fourier Descriptors (FD), and Simple Fractal Dimension (SFD) for shape classification using a database of 11,000 fish contours and 1,100 classes.
- CSD was the most effective among them, for this particular application, and MFD was competitive with FD and BAS, and superior to SFD and MIV.

Combining image operators

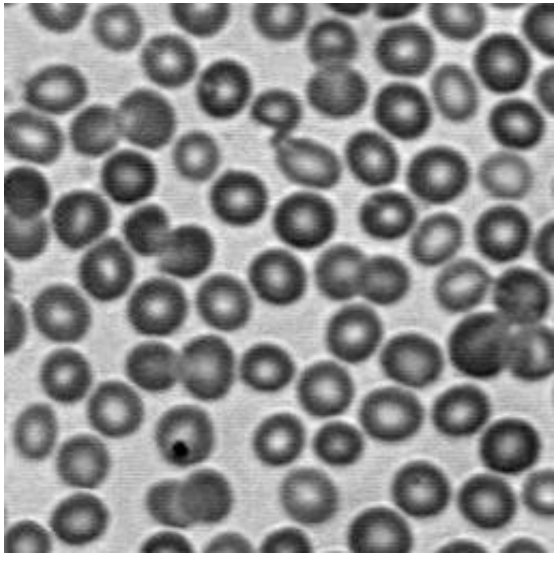
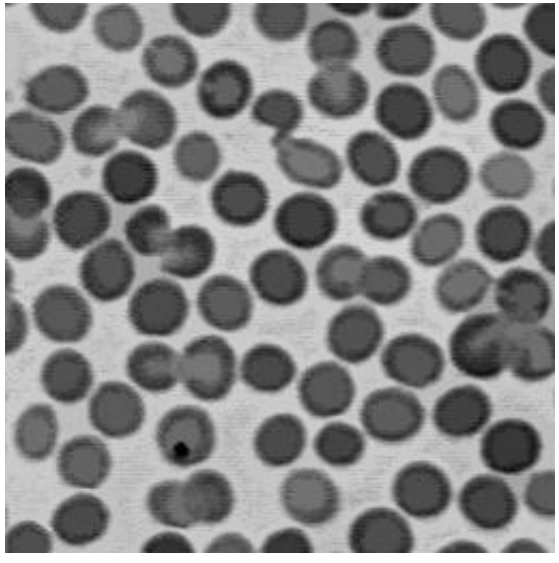
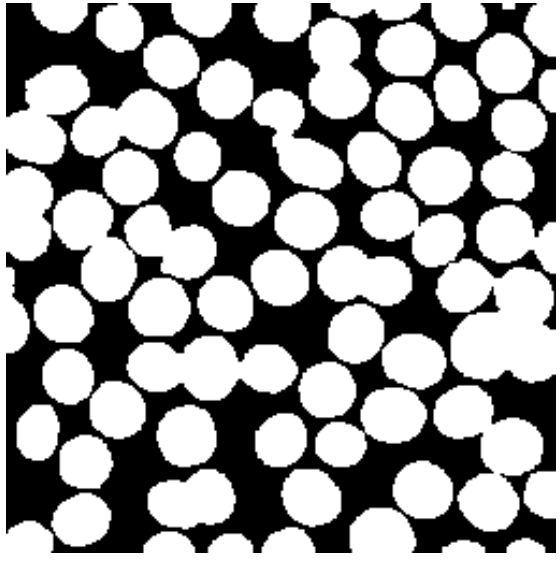
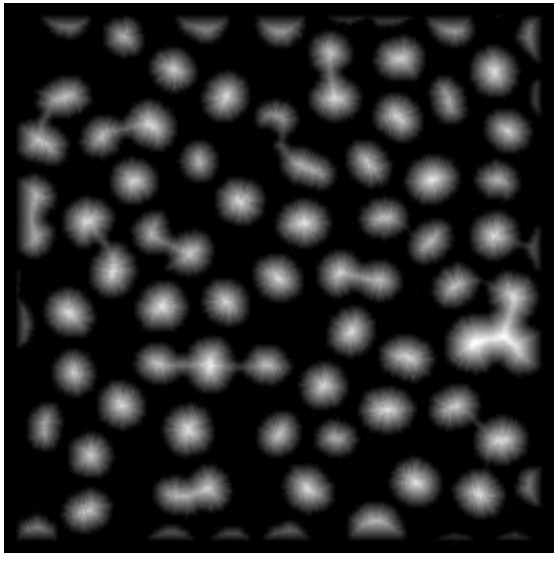
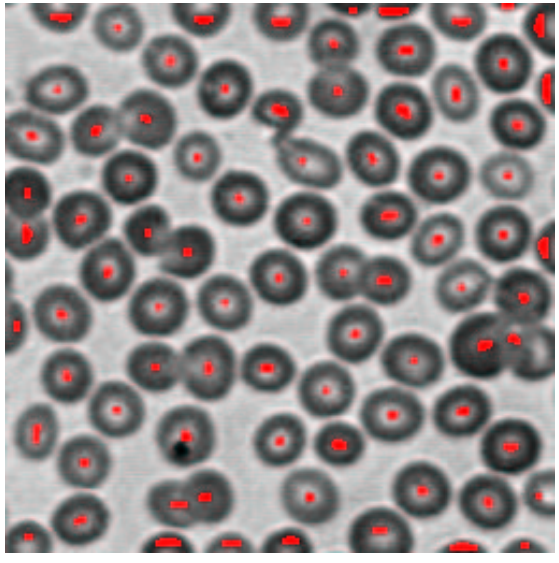
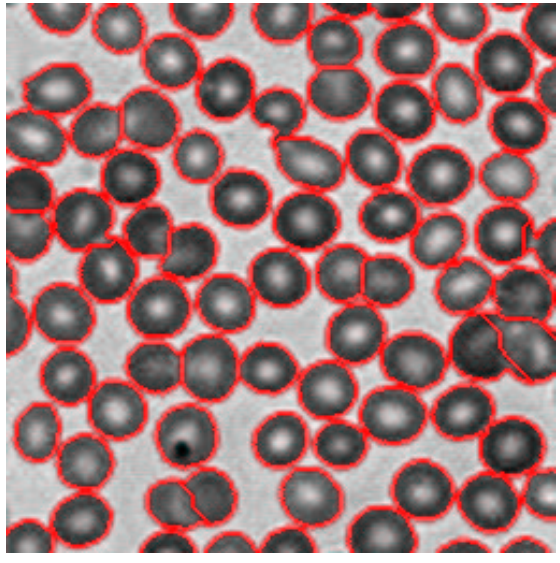




Image of blood cells (left). Area opening by IFT (center). Thresholding and morphological opening (right).

Combining image operators

Distance transform by IFT (left). Regional maxima by IFT (center).

Local reconstruction by IFT (right).

Current work

- Procedures for automatic seed selection during interactive 3D segmentation.
- More effective path-cost functions for 3D segmentation of MR-images of the brain and the evaluation of the methods.
- Methods that combine boundary tracking and active contours for image segmentation.
- Automatic image operators based on the DIFT.
- 3D multiscale skeletonization.
- Further improvements on multiscale fractal dimension and contour salience descriptor.

Conclusion

- The IFT provides a different way to look at image processing operations.
- To develop an image operator based on the IFT, one needs to
  - relate the operator to an optimum image partition problem with connectivity restriction,
  - find suitable adjacency relation and smooth path-cost function, and
  - apply some local processing to the output of the IFT algorithm.