Shape representation by skeletonization

Kálmán Palágyi

Dept. Image Processing and Computer Graphics
University of Szeged, Hungary

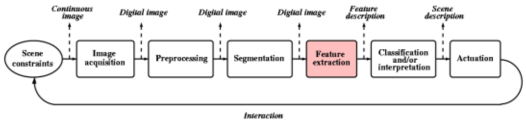
Syllabus

- Shape
- Shape features
- Skeleton
- Skeleton-like shape features
- Skeletonization
- Applications

Shape

Shape is a fundamental concept in computer vision.
It can be regarded as the basis for high-level image processing stages concentrating on scene analysis and interpretation.

The generic model of a modular machine vision system



G.W. Awcock, R. Thomas (*1996*)

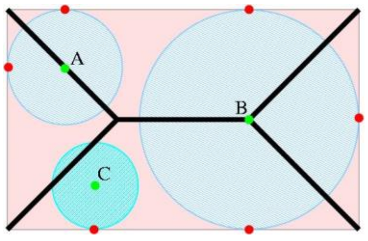
Shape representation techniques

- to apply a **transform** in order to represent an object in terms of the transform coefficients,
- to describe the **boundary** that surrounds an object,
- to describe the **region** that is occupied by an object.

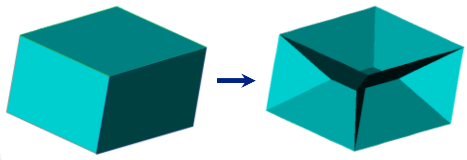
Skeleton

- result of the Medial Axis Transform: object points having at least two closest boundary points
- praire-fire analogy: the boundary is set on fire and skeleton is formed by the loci where the fire fronts meet and quench each other
- the locus of the centers of all the maximal inscribed hyper-spheres

Skeleton in 2D



Skeleton in 3D

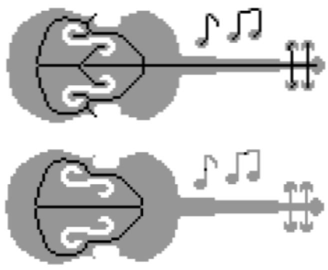


3D skeletons generally contain 2D segments (i.e., surface patches)

Skeleton

- represents
  - the general form of an object,
  - the topological structure of an object, and
  - local object symmetries.
- invariant to
  - translation,
  - rotation, and
  - (uniform) scale change.
- simplified and thin.

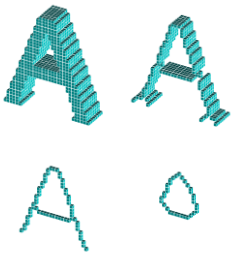
Skeleton-like shape features in 2D



centerline

topological kernel

Skeleton-like shape features in 3D



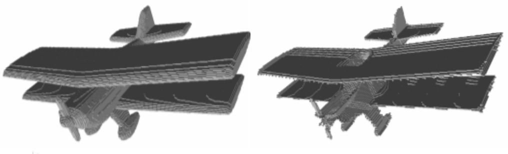
original object

medial surface

centerline

topological kernel

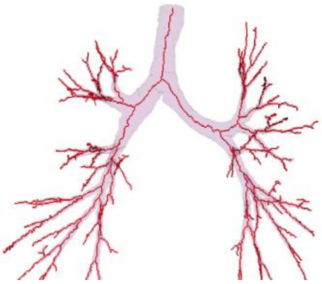
Skeleton-like shape features in 3D



original object

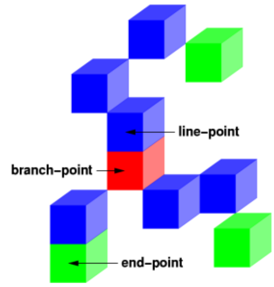
medial surface

Skeleton-like shape features in 3D



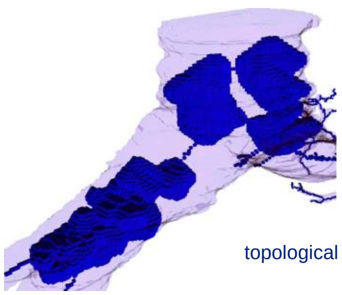
centerline

Skeleton-like shape features in 3D



centerline

Skeleton-like shape features in 3D



topological kernel

Skeletonization techniques

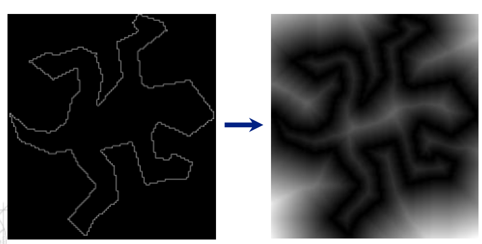
- distance transform
- Voronoi diagram
- thinning

Distance transform

Input:
Binary array *A* containing feature elements (1's) and non-feature elements (0's).

Output:
Distance map *B*: non-binary array containing the distance to the closest feature element.

Distance transform



feature mask — distance map

Distance transform

4	3	2	1	2	3	4
3	2	1	0	1	2	3
2	1	0	1	0	1	2
2	1	0	1	1	0	1
1	0	1	2	2	1	0
1	0	1	2	3	2	1
0	1	2	3	4	3	2

	1	
1	0	1
	1	

DT using Manhattan, city-block, or 4 distance

Distance transform

2	2	1	1	1	2	2
2	1	1	0	1	1	2
2	1	0	1	0	1	1
1	1	0	1	1	0	1
1	0	1	1	1	1	0
1	0	1	2	2	1	1
0	1	1	2	2	2	2

1	1	1
1	0	1
1	1	1

DT using chess-board or 8 distance

Distance transform

16	15	14	13	12	13	14	15	16
15	12	11	10	9	10	11	12	15
14	11	8	7	6	7	8	11	14
13	10	7	4	3	4	7	10	13
12	9	6	3	0	3	6	9	12
13	10	7	4	3	4	7	10	13
14	11	8	7	6	7	8	11	14
15	12	11	10	9	10	11	12	15
16	15	14	13	12	13	14	15	16

DT using (3,4)-chamfer distance

Linear time distance mapping

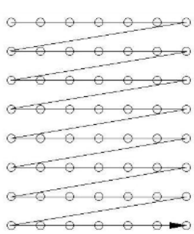
Input:
Binary array *A*=[`a(i,j)`] of size `n1xn2` containing feature elements (1's) and non-feature elements (0's)

Output:
Distance map *B* =[`b(i,j)`] is a non-binary array containing the distance to the closest feature element

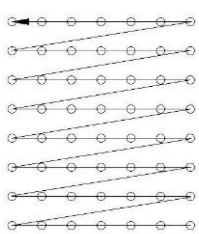
G. Borgefors (1984)

```
remark initialization
 for i=1 to n1 do
  for j=1 to n2 do
   if a(i,j)=1 then b(i,j)=0
   else             b(i,j)=∞
remark forward scan
 for i=1 to n1 do
  for j=1 to n2 do
   b(i,j)=min{
              b(i-1,j-1)+d2,
              b(i-1,j  )+d1,
              b(i-1,j+1)+d2,
              b(i  ,j-1)+d1,
              b(i  ,j  )
             }
remark backward scan
 for i=n1 downto 1 do
  for j=n2 downto 1 do
   b(i,j)=min{
              b(i  ,j  )   ,
              b(i  ,j+1)+d1,
              b(i+1,j-1)+d2,
              b(i+1,j  )+d1,
              b(i+1,j+1)+d2
             }
```


Linear time distance mapping

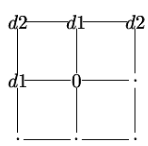


forward scan

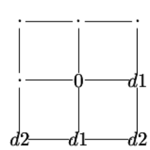


backward scan

Linear time distance mapping



forward scan



backward scan

best choice: $d1=3$, $d2=4$

Linear time distance mapping

0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

input (feature)

.	.	.	.
.	.	0	.
.	.	.	.
.	.	.	.
.	0	.	.
.	.	.	.

initialization („.” → ∞)

Linear time distance mapping

.	.	.	.
.	.	0	.
.	.	.	.
.	.	.	.
.	0	.	.
.	.	.	.

initialization („.” → ∞)

.	.	.	.
.	.	0	3
.	4	3	4
8	7	6	7
11	0	3	6
4	3	4	7

forward scan

Linear time distance mapping

.	.	.	.
.	.	0	3
.	4	3	4
8	7	6	7
11	0	3	6
4	3	4	7

forward scan

7	4	3	4
6	3	0	3
7	4	3	4
4	3	4	7
3	0	3	6
4	3	4	7

backward scan

Linear time distance mapping

0	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0
0	1	0	0
0	0	0	0

input (feature)

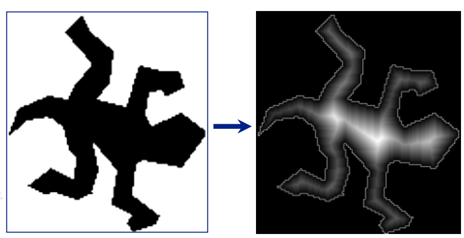
7	4	3	4
6	3	0	3
7	4	3	4
4	3	4	7
3	0	3	6
4	3	4	7

distance map

Distance-based skeletonization

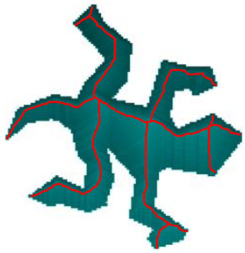
1. Calculate the distance map from the background (i.e., all zeroes in the input binary image are feature points)
2. Detect ridges (i.e., local maxima)

Distance-based skeletonization



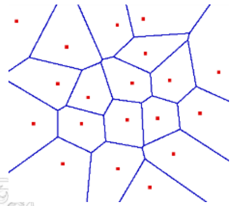
original binary image → distance map

Distance-based skeletonization



detected ridges

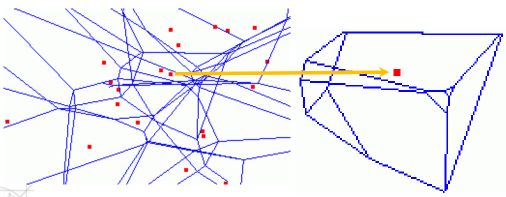
Voronoi diagram



Input:
Set of points (generating poins)

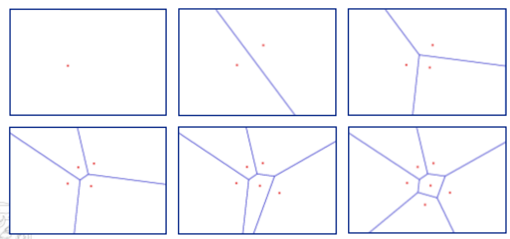
Output:
the partition of the space into cells so that each cell contains exactly one generating point and the locus of all points which are closer to this generating point than to others.

Voronoi diagram



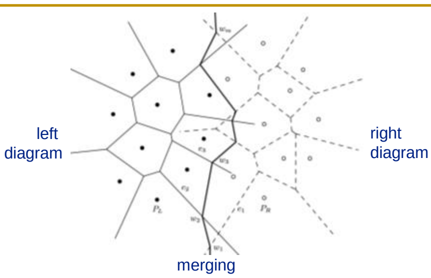
a cell (convex polyhedron) of a 3D Voronoi diagram

Voronoi diagram



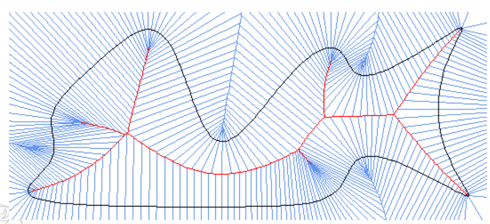
incremental construction $O(n)$

Voronoi diagram



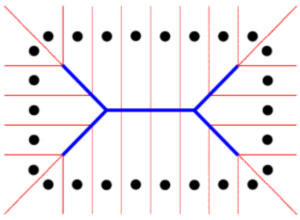
divide and conquer $O(n \log n)$

Voronoi diagram → skeleton



set of generating points = sampled boundary

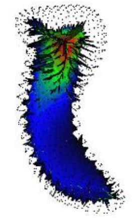
Voronoi diagram → skeleton



If the density of boundary points goes to infinity, then the corresponding Voronoi diagram converges to the skeleton.

Voronoi skeleton

original 3D object



Voronoi skeleton

M. Styner (*UNC, Chapel Hill*)

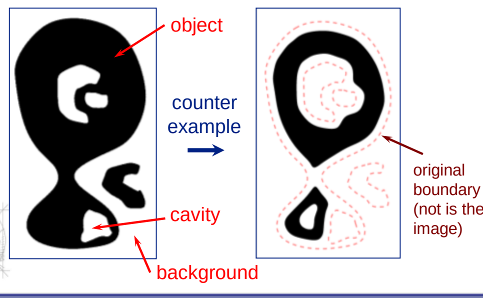
Thinning algorithms

repeat
remove „*deletable*" points from the actual binary image } one iteration step
until no points are deleted

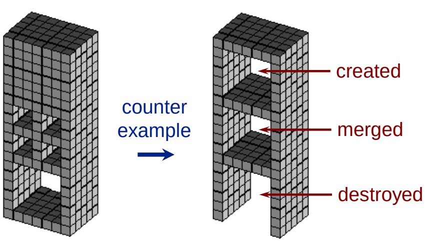
degrees of freedom:

- which points are regarded as „*deletable*"?
- how to organize one iteration step?

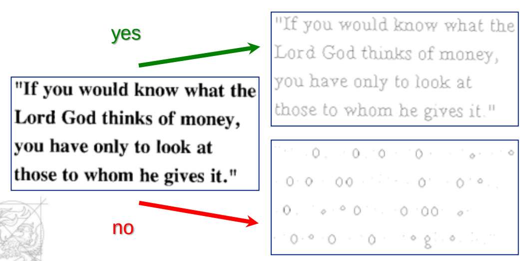
Topology preservation in 2D



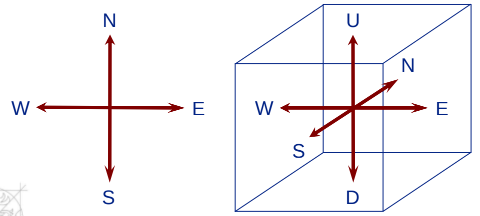
Topology preservation in 3D



Shape preservation



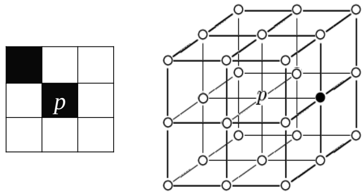
Some concepts



main directions in 2D and 3D

Some concepts

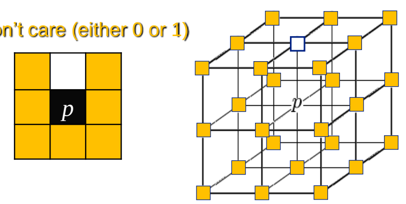
p is a **line-end point**
if it is adjacent to just one object point



Some concepts

p is a **border point** if it is (4/6-)adjacent to at least one non-object point

don't care (either 0 or 1)



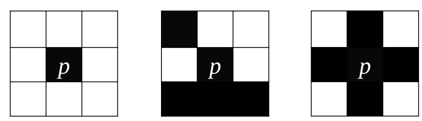
border point of type N

border point of type U

Some concepts

An object-point is **simple** if its deletion preserves the topology of the picture.

Examples of ***non***-simple points in 2D pictures:



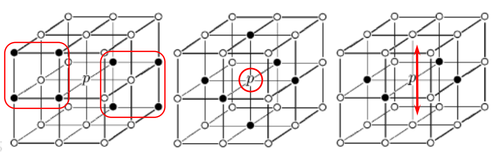
deleting an object

splitting an object

creating a cavity

Some concepts

Examples of ***non***-simple points in 3D pictures:



splitting an object

creating a cavity

creating a hole

Some concepts

*Simple*ness is a local property:
It depends on the 3x3 / 3x3x3 neighborhood of the point in question.

↓

It can be decided by using a precalculated LUT (look-up table) of size 128 bit / 8 Mbyte.

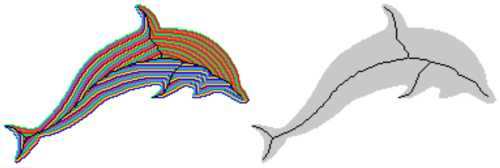
A 2D parallel thinning algorithm

repeat
for each directions N,E,S, and W **do**
an object point is **deletable** if it is
- a *border* point according to the actual direction,
- *not* a *line-end* point, and
- *simple*
delete all **deletable** points simultaneously
until no points are deleted

A. Rosenfeld (*1975*)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Image Processing and Computer Graphics

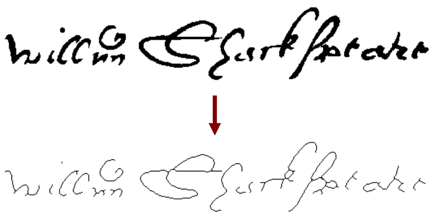
A 2D parallel thinning algorithm



A. Rosenfeld (*1975*)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Image Processing and Computer Graphics

A 2D parallel thinning algorithm



A. Rosenfeld (*1975*)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Image Processing and Computer Graphics

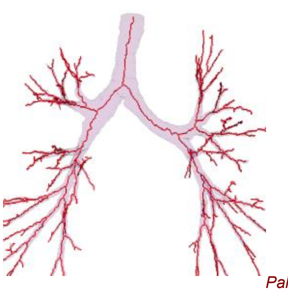
A 3D sequential curve-thinning algorithm

repeat
for each direction U,N,E,S,W, and D **do**
mark object point p if it is
- a *border* point according to the actual direction,
- not a *line-end* point, and
- a *simple* point
for each marked point q **do**
delete q if it is
- not a *line-end* point, and
- a *simple* point in the actual picture
until no points are deleted

Palágyi et al. (*2001*)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Image Processing and Computer Graphics

A 3D sequential curve-thinning algorithm



Palágyi et al. (*2001*)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Image Processing and Computer Graphics

Thinning

- allows centerline extraction in 3D
- makes easy implementation possible
- takes the least computational costs
- can be executed in parallel

Medical applications

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
SZEGEDI TUDOMÁNYEGYETEM
Department of Image Processing and Computer Graphics

Tubular structures (e.g., blood vessels, airways) are frequently found in living organs.
They can be represented by their centerlines (extracted by 3D curve-thinning algorithms).

Cooperation with Medical University Graz

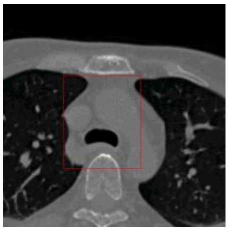
- assessment of laryngotracheal stenosis
- assessment of infrarenal aortic aneurysm
- unravelling the colon

Medical University of Graz

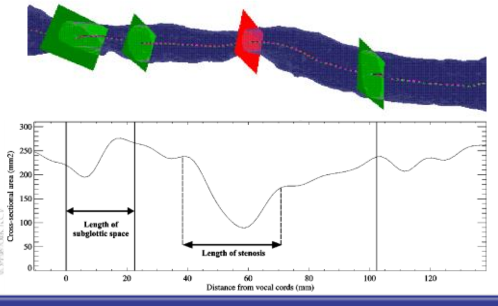
E. Sorantin et al.

Assessment of laryngotracheal stenosis

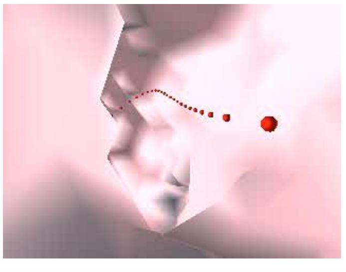
- Data from multirow detector spiral CT
- Laryngo-Tracheal Tract (LTT) segmentation based on based on fuzzy connectedness
- LTT centerline extraction by 3D curve-thinning
- Diameter estimation based on the LTT cross-sectional profile along the centerline



Assessment of laryngotracheal stenosis

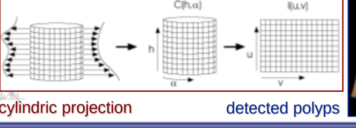


Virtual colonoscopy



Unravelling the colon

segmented colon and its centerline



cylindric projection

detected polyps

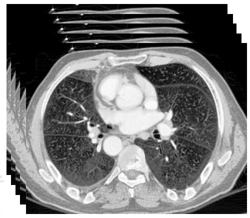
Cooperation with The University of Iowa

Quantitative analysis of pulmonary airway trees

The University of Iowa

The University of Iowa College of Engineering Imaging Group

Quantitative analysis of pulmonary airway trees



Multi-detector Row Spiral CT

512 x 512 voxels

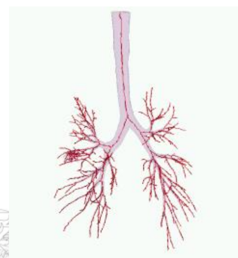
500 – 600 slices

0.65 x 0.65 x 0.6 mm^3 (almost isotropic)

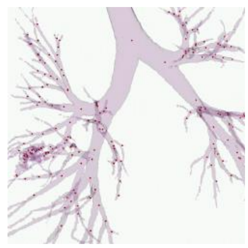
Quantitative analysis of pulmonary airway trees

lung segmentation

Quantitative analysis of pulmonary airway trees

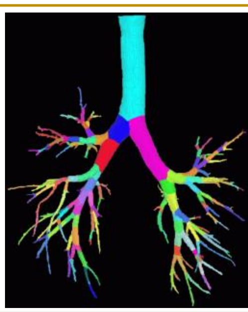


centerline



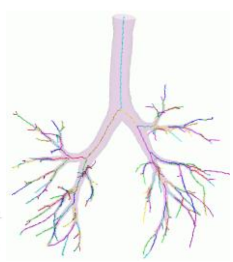
identified branch-points

Quantitative analysis of pulmonary airway trees

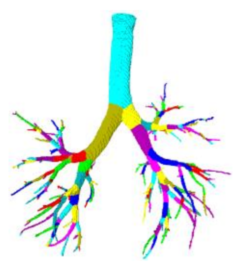


branch partitioning

Quantitative analysis of pulmonary airway trees

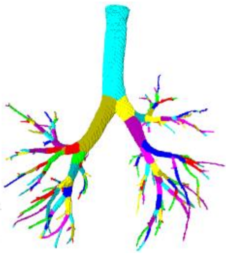


centerline labeling



label propagation

Quantitative analysis of pulmonary airway trees


labeled tree


formal tree (in XML)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS SZEGEDI TUDOMÁNYEGYETEM Department of Image Processing and Computer Graphics

Quantitative analysis of pulmonary airway trees

Quantitative indices for tree branches

- length (Euclidean distance between the parent and the child branch points)
- volume (volume of all voxels belonging to the branch)
- surface area (surface area of all boundary voxels belonging to the branch)
- average diameter (assuming cylindric segments)

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS SZEGEDI TUDOMÁNYEGYETEM Department of Image Processing and Computer Graphics

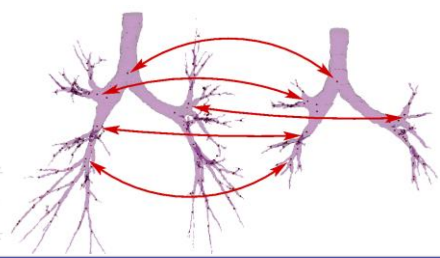
Quantitative analysis of pulmonary airway trees

- The automated method for skeletonization, branch-point identification and quantitative analysis of tubular tree structures is robust, efficient, and highly reproducible
- The method was validated in computer and physical phantoms and in vivo CT scans of human lungs.
- The validation studies demonstrated high reproducibility of derived quantitative indices of the tubular structures ($p<<0.001$).

UNIVERSITAS SCIENTIARUM SZEGEDIENSIS SZEGEDI TUDOMÁNYEGYETEM Department of Image Processing and Computer Graphics

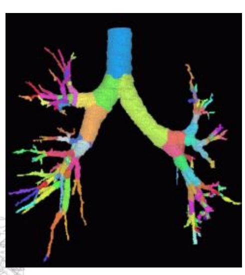
Tree matching

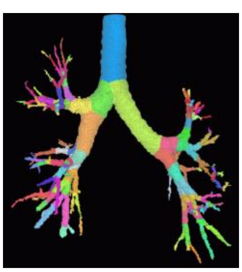
It is based on identified branch-points in the centerline.



UNIVERSITAS SCIENTIARUM SZEGEDIENSIS SZEGEDI TUDOMÁNYEGYETEM Department of Image Processing and Computer Graphics

Tree matching


functional residual capacity (FLC)


total lung capacity (TLC)