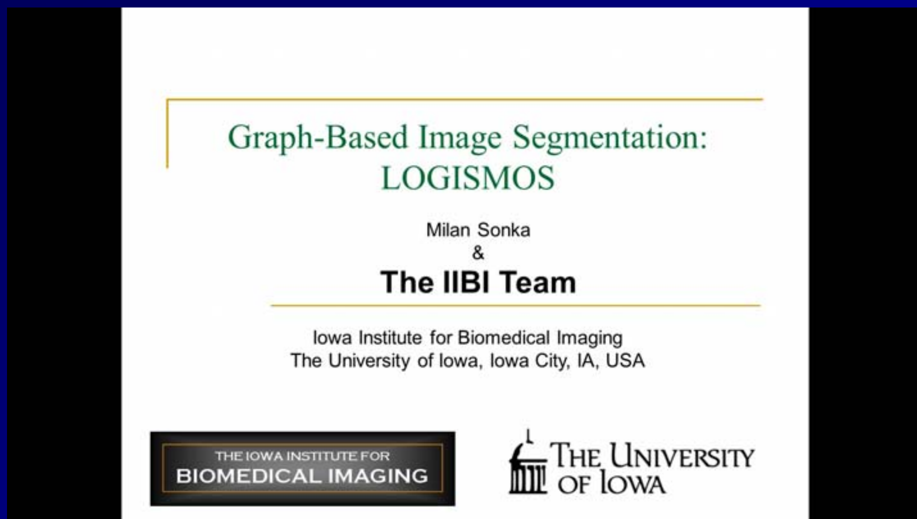
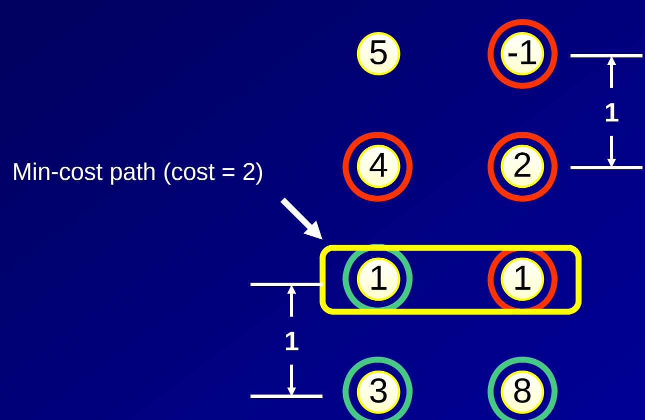
LOGISMOS – Single and Multiple Surface Segmentation



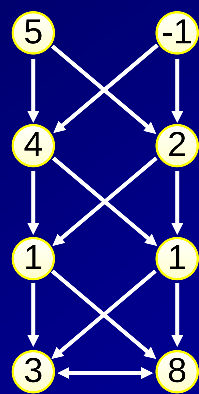
Example: Min *s*-*t* Cut approach to finding min-cost "path"

- Every pixel corresponds to a node in the graph, node costs used
- The path intersects with each column at exactly one node
- **Smoothness constraint**:
  max. vertical distance of neighbor-column nodes = 1



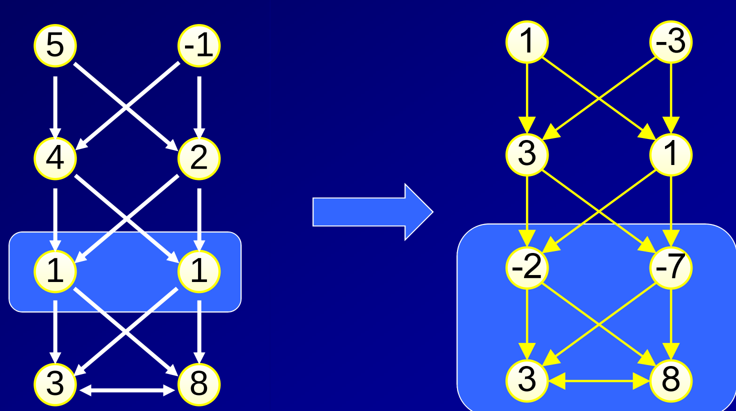
Edge Construction:

- Connect each node to its bottom-most neighbor in the adjacent column.
- Build vertical edges along each column, pointing downwards.



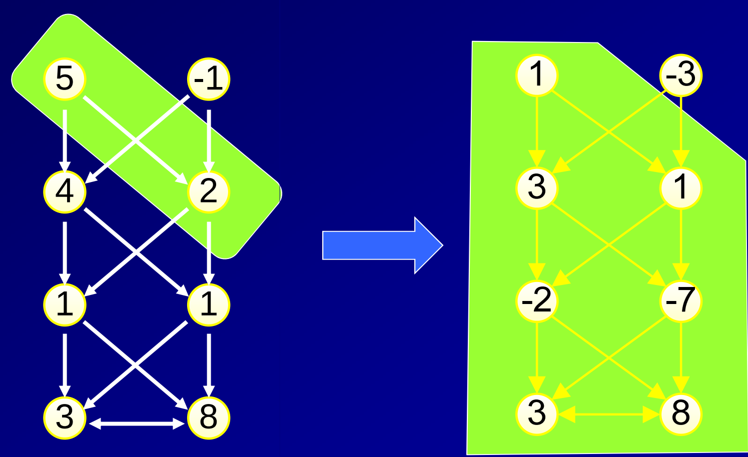
Goal of the transform:

- transform the graph in a form that can be solved by finding a minimum-cost closed set
- efficient optimization exists for minimum-cost closed set



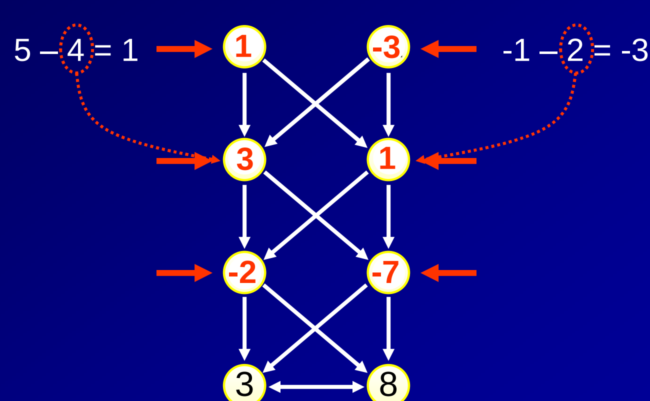
Goal of the transform:

- transform the graph in a form that can be solved by finding a minimum-cost closed set
- efficient optimization exists for minimum-cost closed set



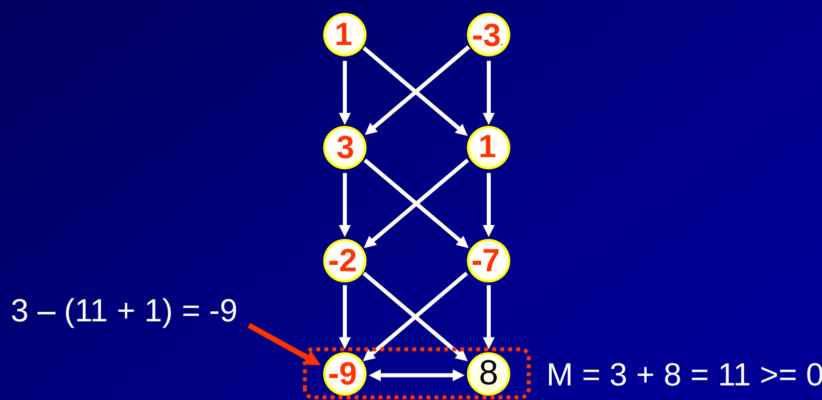
Cost Transformation:

- Along each column, subtract the cost of each node by the cost of the node immediately beneath it.
- The bottom-most two nodes are unchanged.



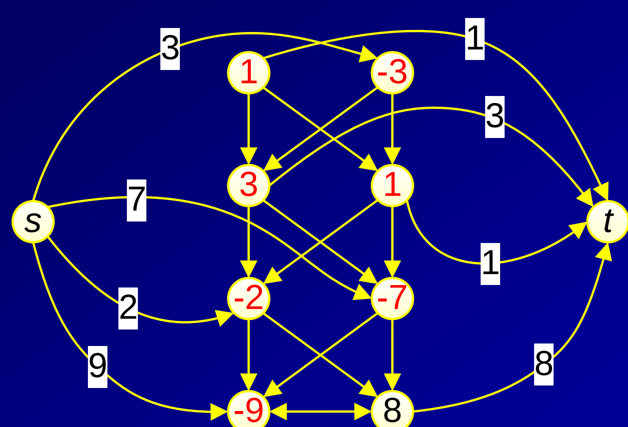
Need to avoid empty zero-cost closed set solution:

- M: sum of costs of the bottommost nodes
- If M >= 0:
  - Select ANY one of those nodes
  - Subtract (M + 1) from its cost



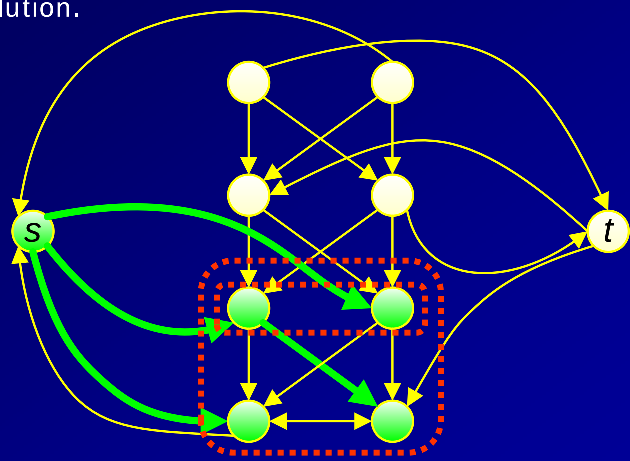
Compute the **Min-Cost Closed Set (cont'd)**:

- Can be solved by a **Min *s*-*t* Cut (Max Flow)** algorithm
- 2 auxiliary nodes – a start (*s*) & a terminal (*t*) are added
- An edge-weighted directed graph is built

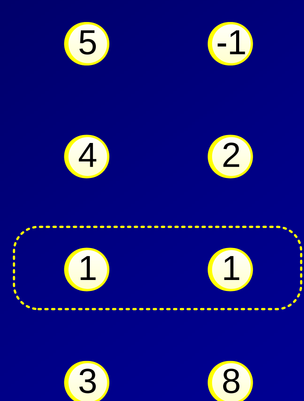


After applying the min *s*-*t* cut algorithm …

- Find all the nodes that can be reached from *s*, and the min-cost closed set is identified.
- The upper envelope of the min-cost closed set is the solution.



The upper envelope of the min-cost closed set is the solution.

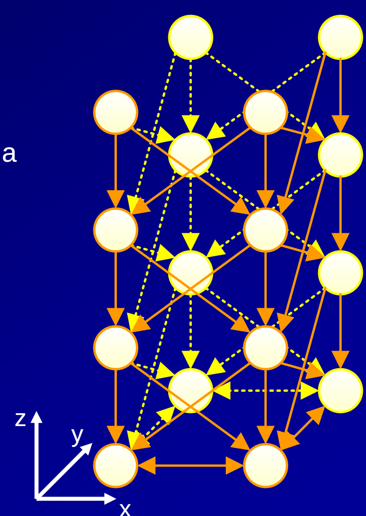


3D Surface

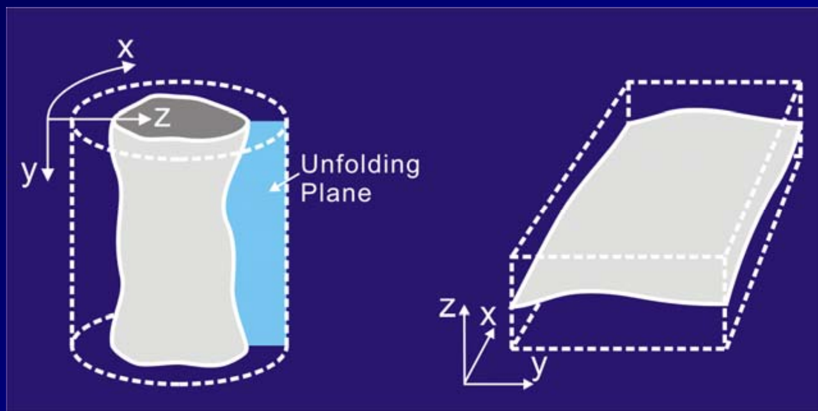
- The surface intersects with ***exactly one*** voxel of each ***column*** of voxels parallel to the z-axis
- The difference in z-coordinates between neighboring voxels on a valid surface in x and y directions
  - ***smoothness constraint*** (**Δ**x, **Δ**y)

3-D Case

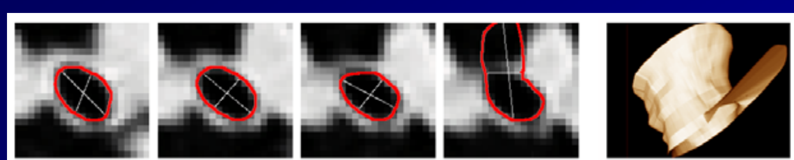
- Principles presented in 2-D are applicable to 3-D
- Detect a **surface** instead of a **path**
- Construct Edges in both x- and y-directions
- x- and y-direction may have different smoothness constraints



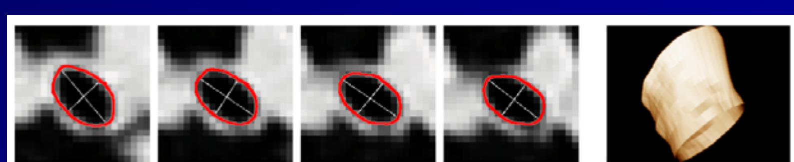
Terrain-like or Tubular Surfaces



Airway Segmentation

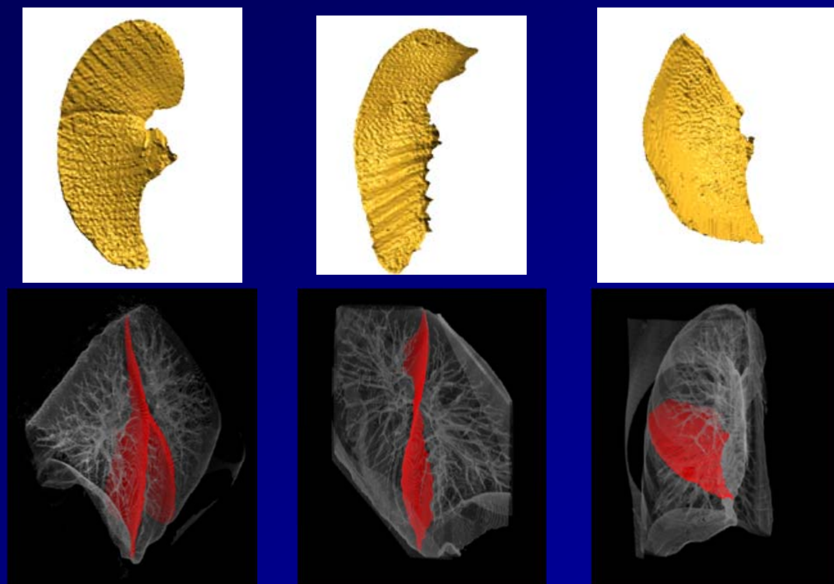


Slice-by-slice Dynamic Programming

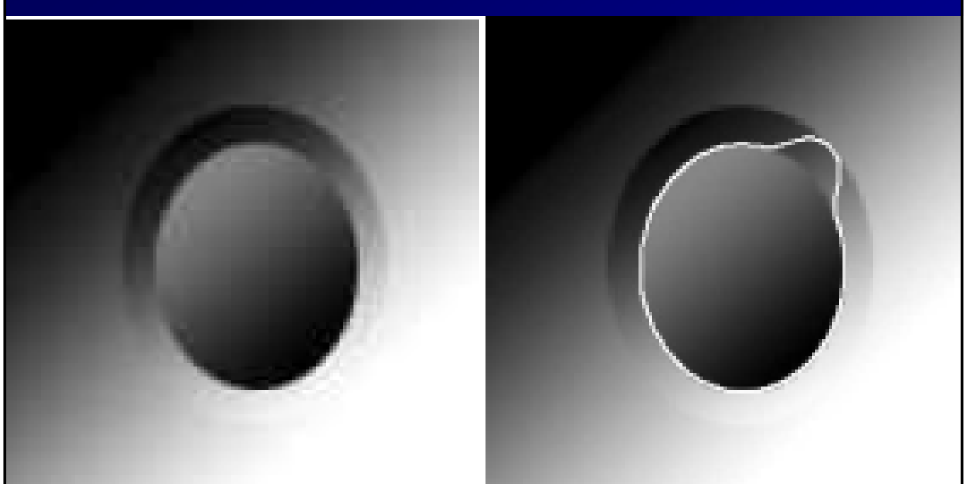


3-D Optimal Surface Detection

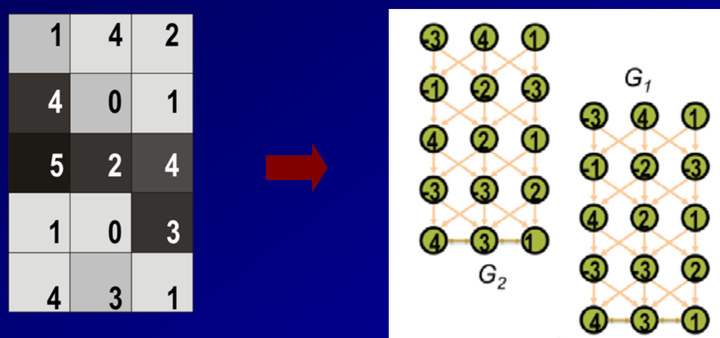
Segmentation of Pulmonary Fissures



Interacting Surfaces



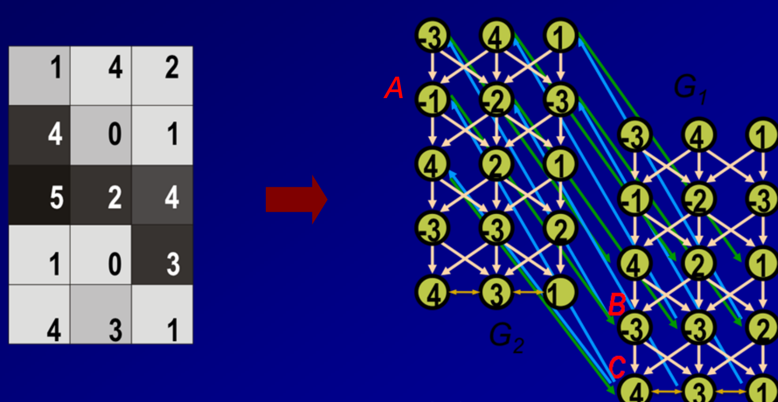
Multiple (2) Interacting Surfaces



- Translation to avoid empty set

Multiple Interacting Surfaces

- Relations between surfaces modeled by "inter-surface" arcs



Minimum distance of 2 pixels, maximum of 3 pixels

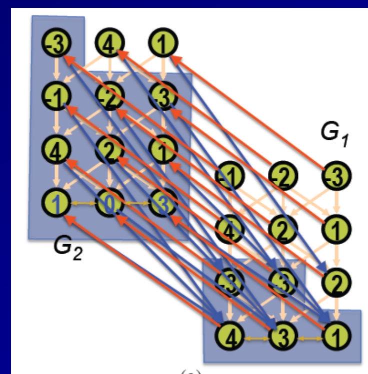
Arc (A,B): If A is in the closure, B must also be in the closure (minimum distance)

Arc (C,A): If C is in the closure, A must also be … (maximum distance)

Multiple Interacting Surfaces

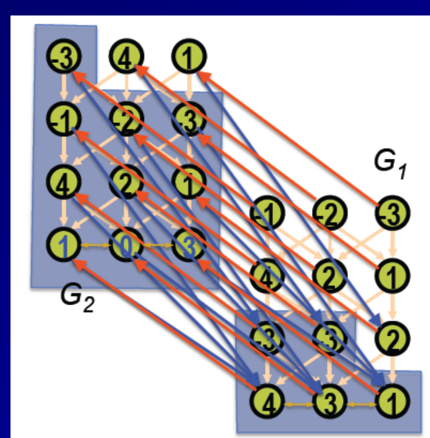
Minimum distance of 1 pixel, maximum of 3 pixels

1	4	2
4	0	1
5	2	4
1	0	3
4	3	1

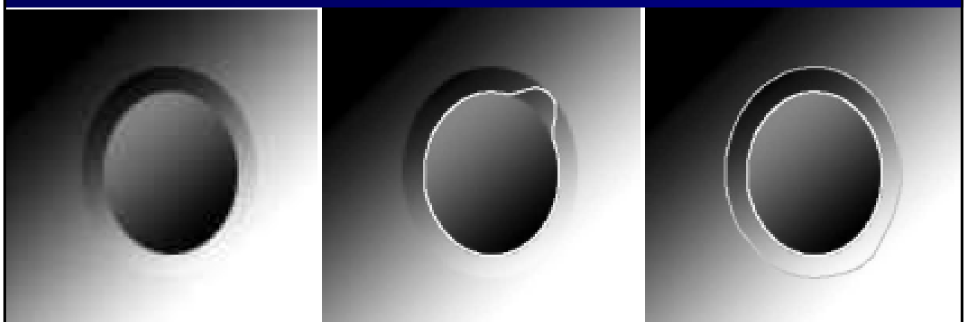


Multiple Interacting Surfaces

1	4	2
4	0	1
5	2	4
1	0	3
4	3	1



Interacting Surfaces
3D Phantom

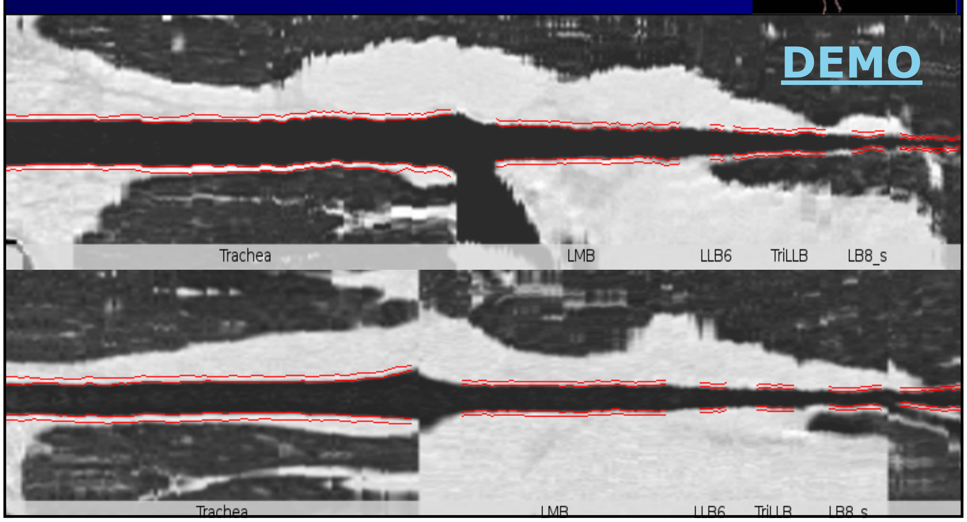


Original

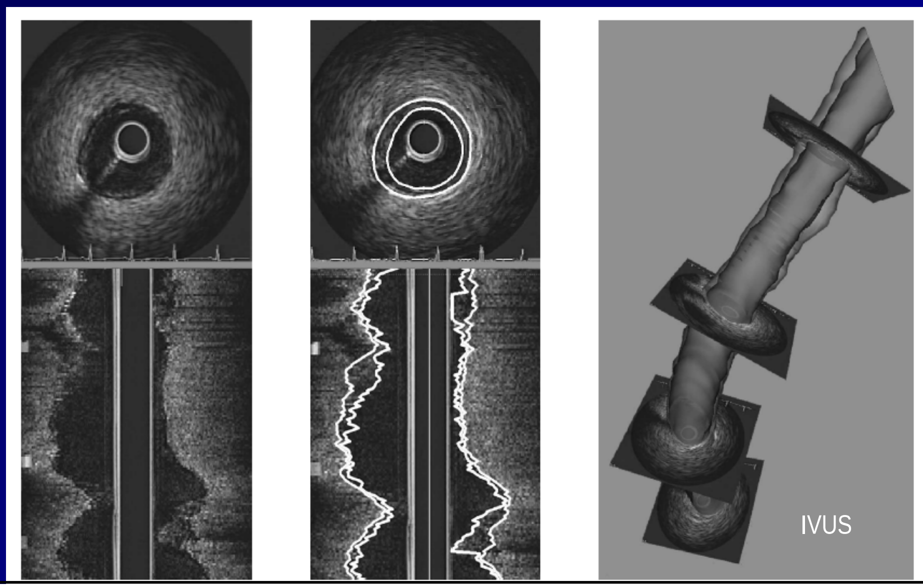
Optimal graph search single surface (no shape term)

Optimal graph search 2 interacting surfaces (no shape term)

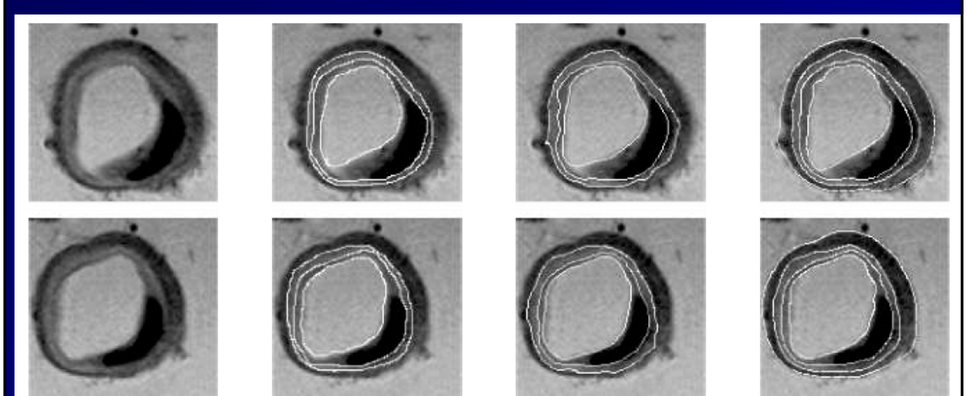
Pulmonary Airway Inner/Outer Surface Detection



IVUS
Simultaneous IEL + EEL



Excised Iliac Artery - MR



Original	Manual tracing	Semi-automated 2-D DP 2.4 clicks/slice	Fully automated 4 simultaneous surfaces in 3D