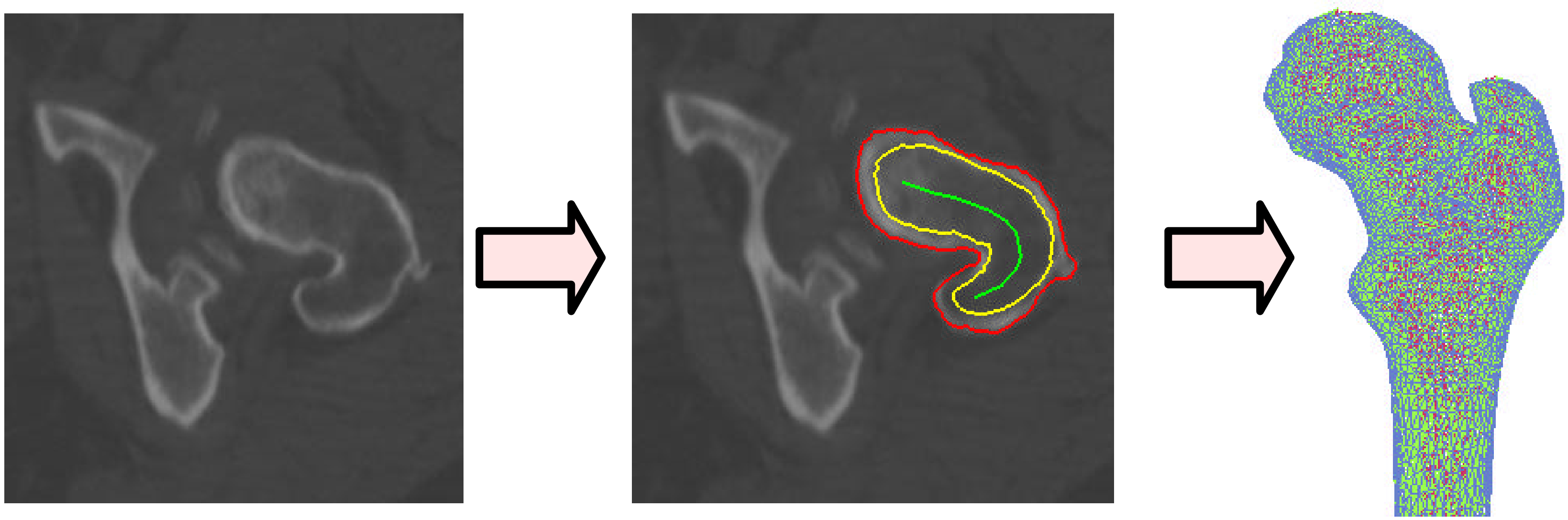
Segmentaion, Modeling and Registration

600.145 Introduction toComputer-Integrated Surgery

Russell H. Taylor

Segmentation & Modeling



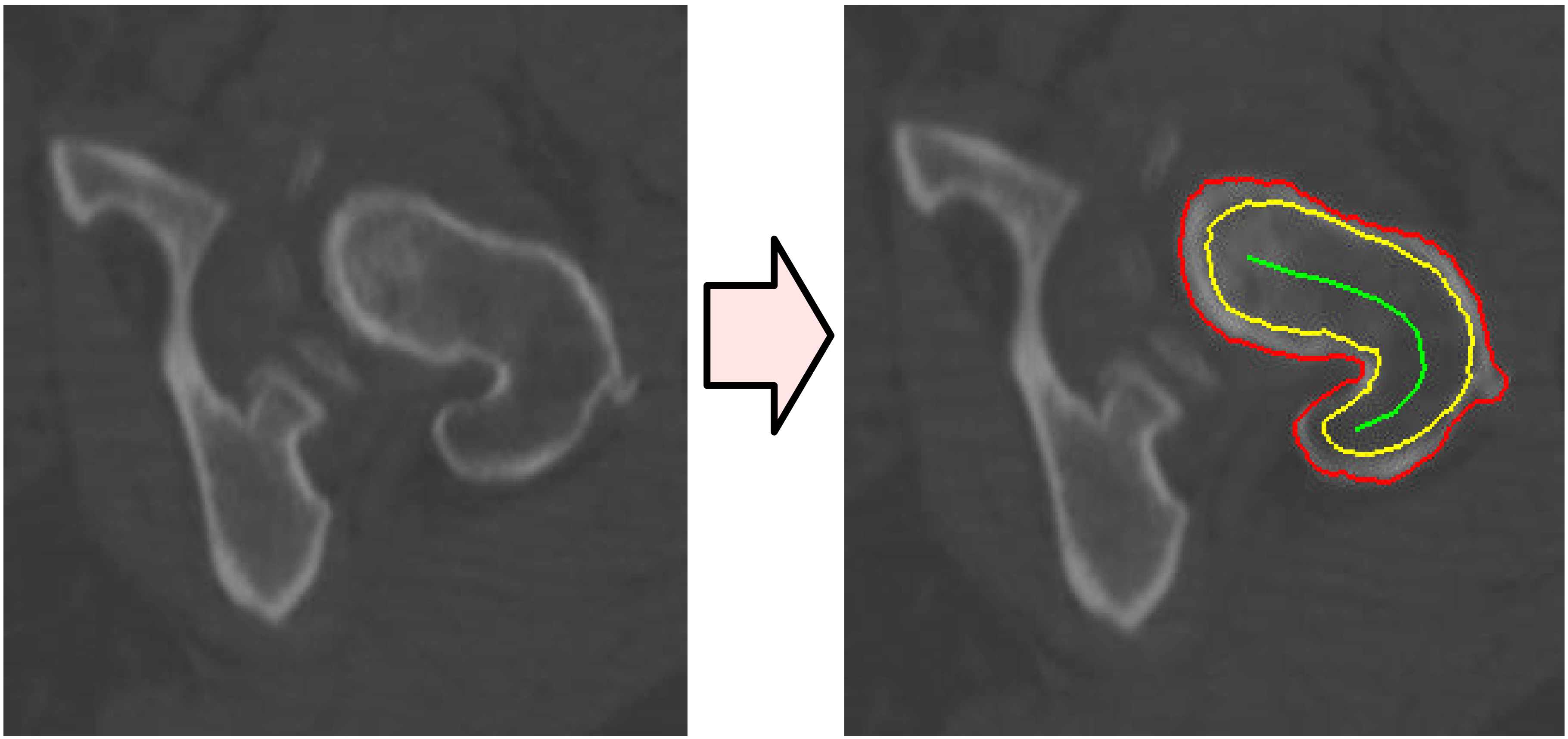
Images

Segmented Images

Models

Segmentation

- Process of identifying structure in 2D & 3D images
- Output may be
  - labeled pixels
  - edge map
  - set of contours



Approaches

- Pixel-based
  - Thresholding
  - Region growing
- Edge/Boundary based
  - Contours/boundary surface
  - Deformable warping
  - Deformable registration to atlases

Thresholding

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

Thresholding

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

Thresholding

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

Thresholding

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

Region Growing

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

Region Growing

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

Region Growing

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

Region Growing

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

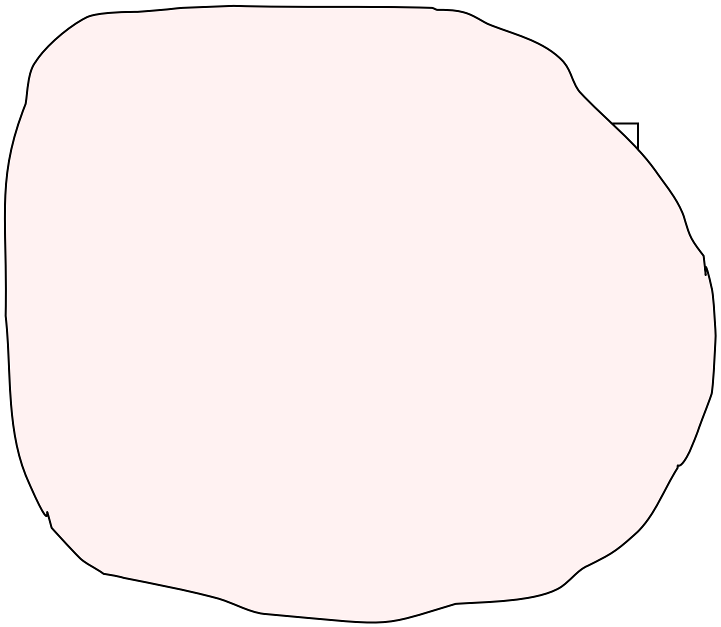
Region Growing

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

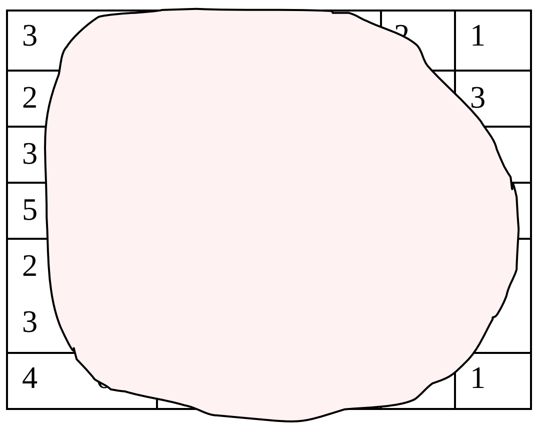
Deformable Surfaces

3	5	7	3	4	2	1
2	4	9	10	22	9	3
3	5	12	11	15	10	3
5	6	11	9	17	19	1
2	3	11	12	18	16	2
3	6	8	10	18	9	5
4	6	7	8	3	3	1

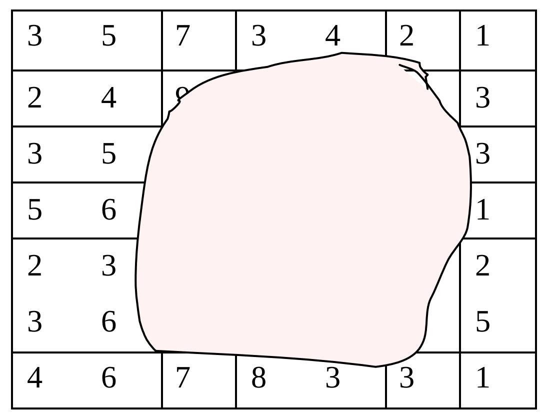
Deformable Surfaces



Deformable Surfaces



Deformable Surfaces



Deformable Surfaces

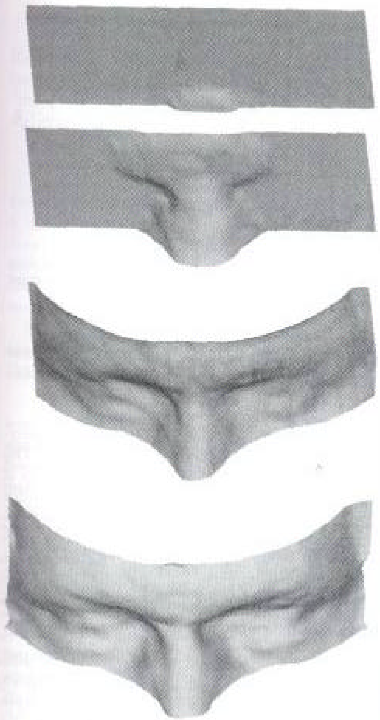
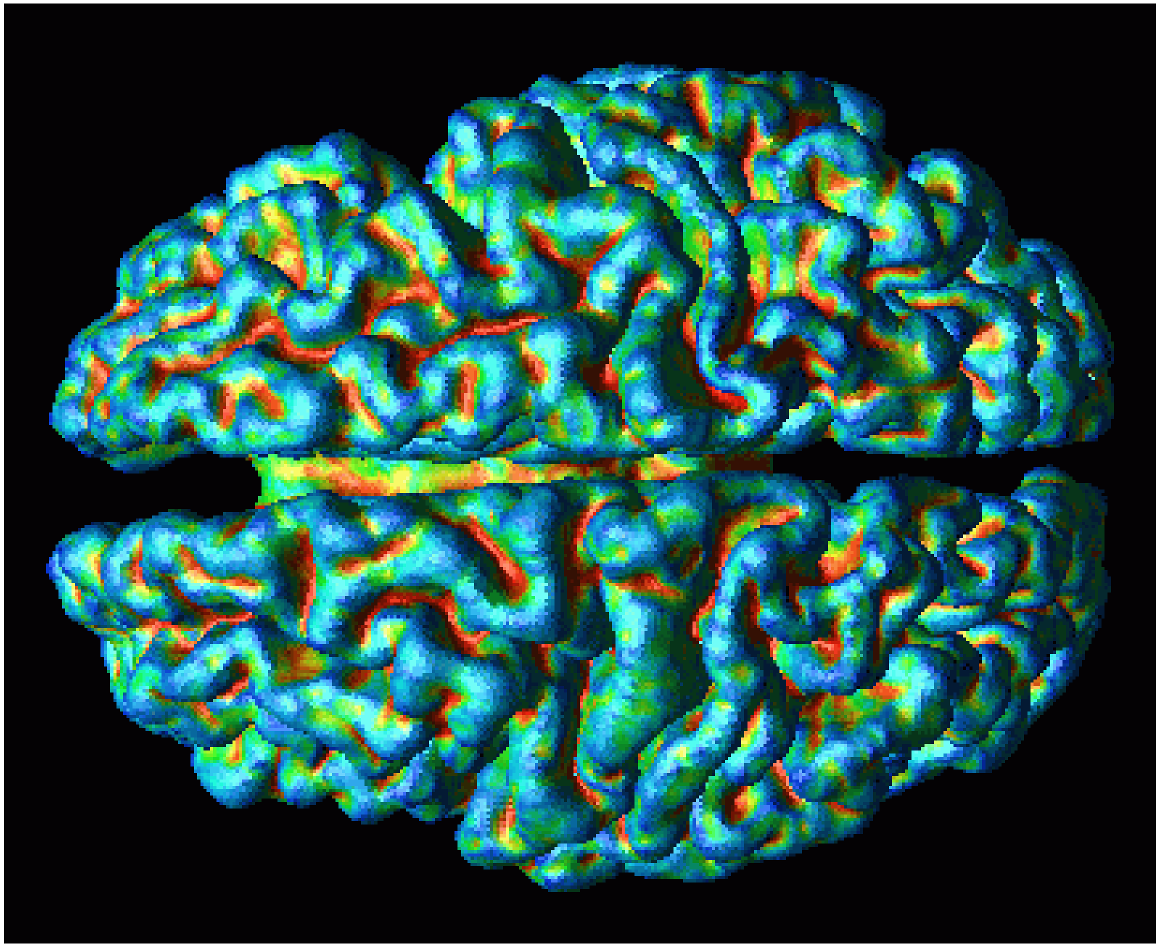


FIGURE 4.4 Evolution of the 3D surface "falling" on a 3D MRI image of a head. The initial surface is a plane on the border of the image.

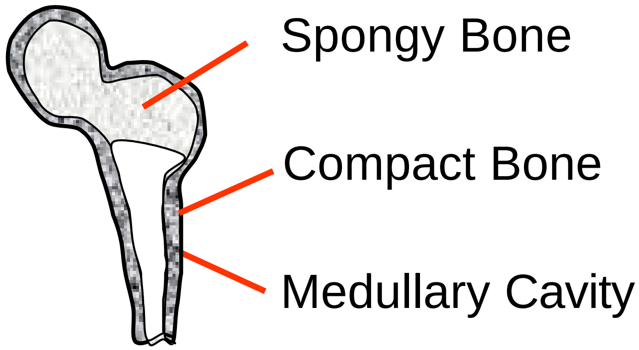
FIGURE 4.7 Segmentation of vertebra defined by a set of CT slices. Four steps of the deformation of a roughly spherical snake spline toward the vertebra are shown.

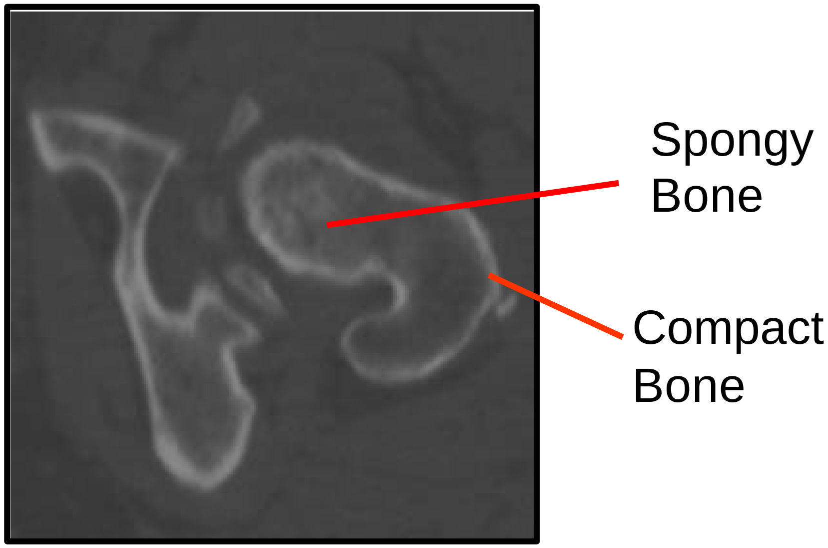
Deformable Surfaces

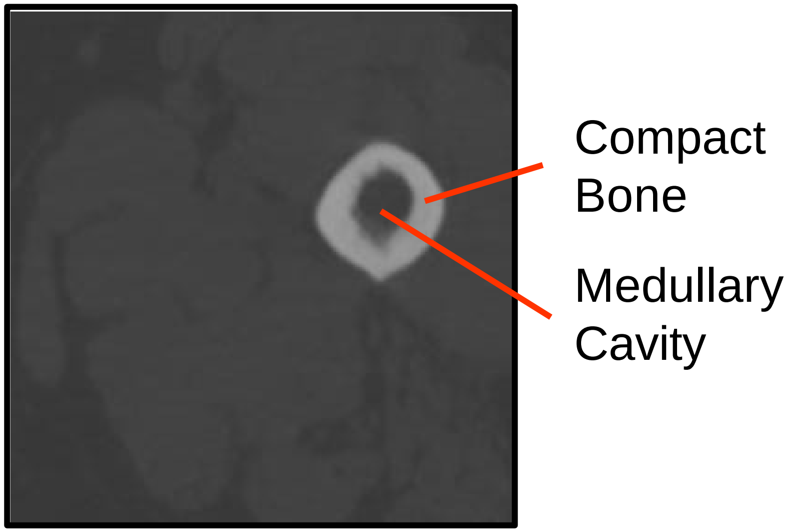


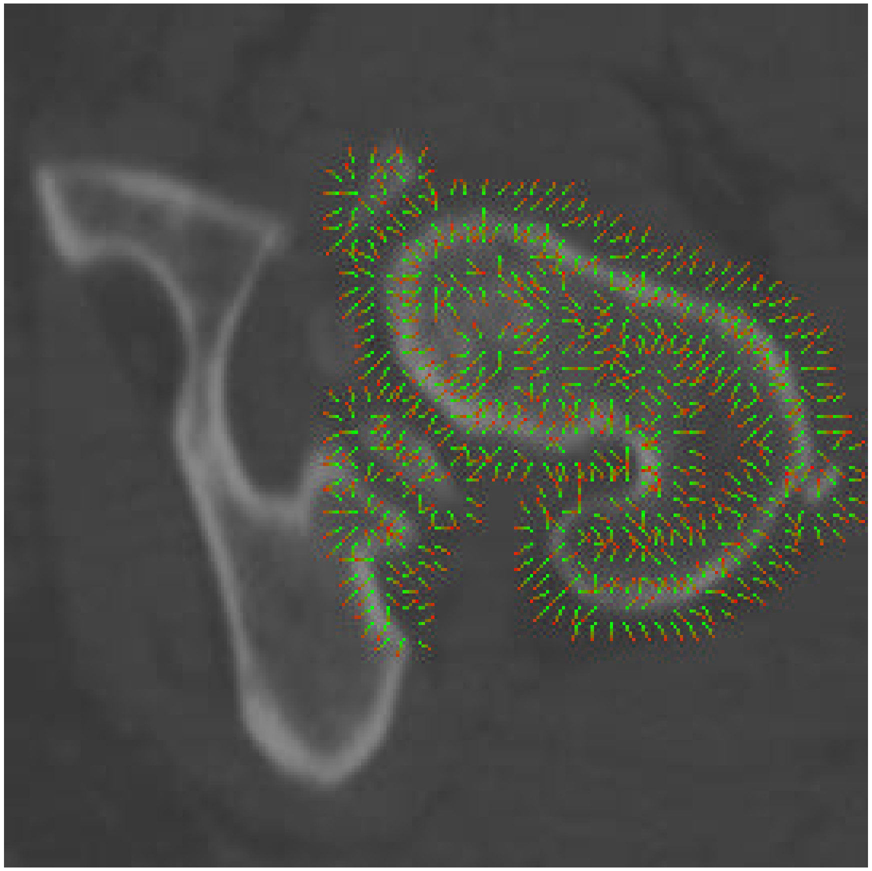
Bone Structure

- Compact bone
- Spongy bone
- Medullary Cavity

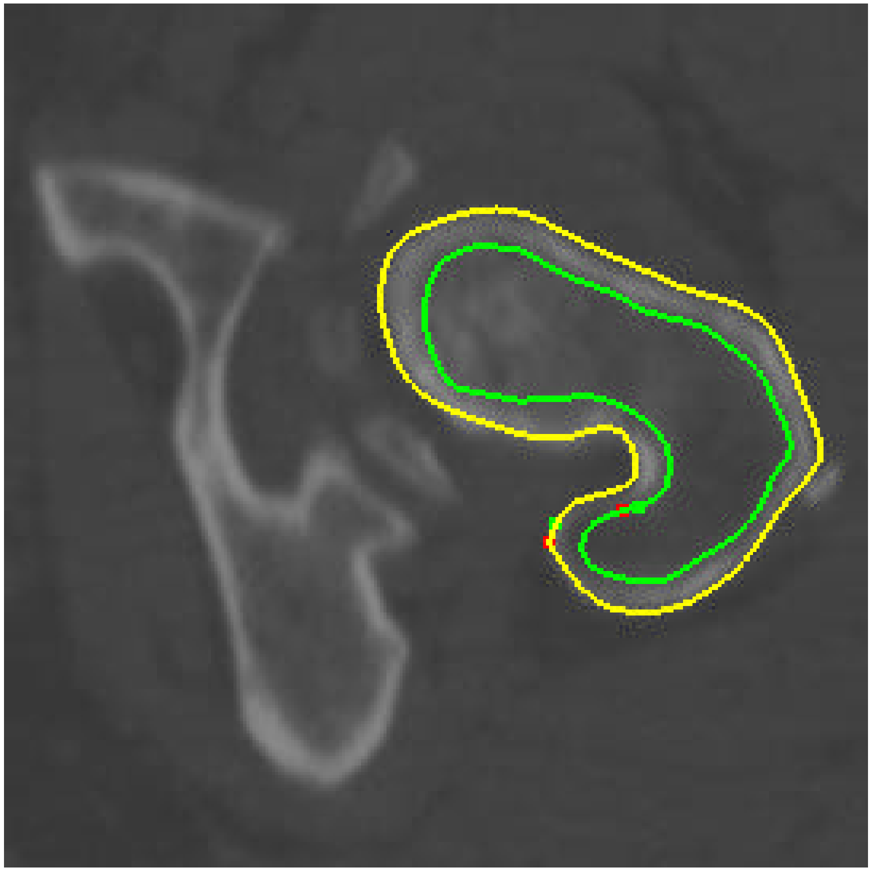






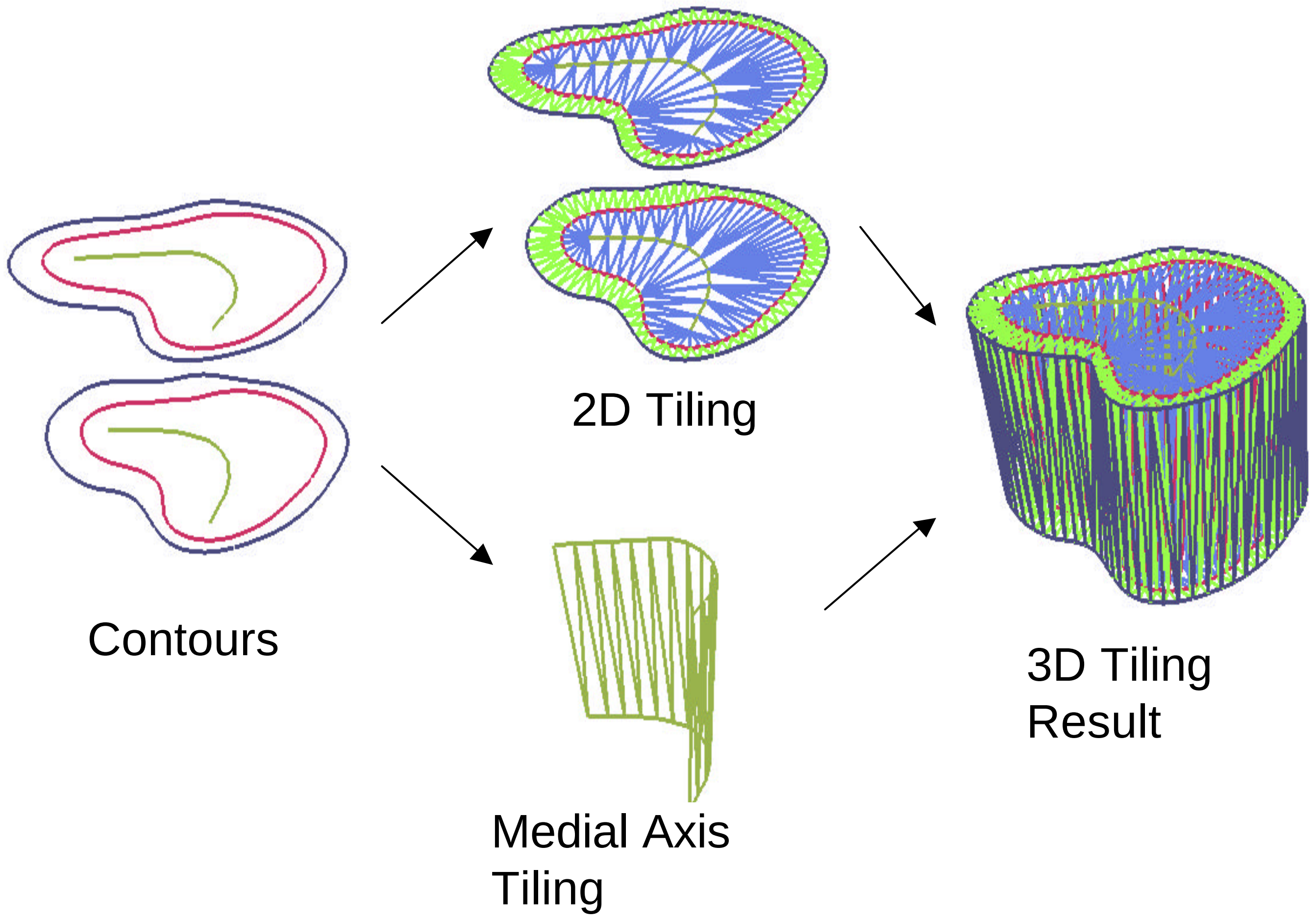


Needle graph of Image force



Bone Contours

Tiling Scheme



Modeling

- Representation of anatomical structures
- Models can be
  - Images
  - Labeled images
  - Boundary representations

Surface Representations

- Implicit Representations

$$\{\bar{x} \mid f(\bar{x}) = 0\}$$

- Explicit Representations
  - Polyhedra
  - Interpolated patches
  - Spline surfaces
  - ...

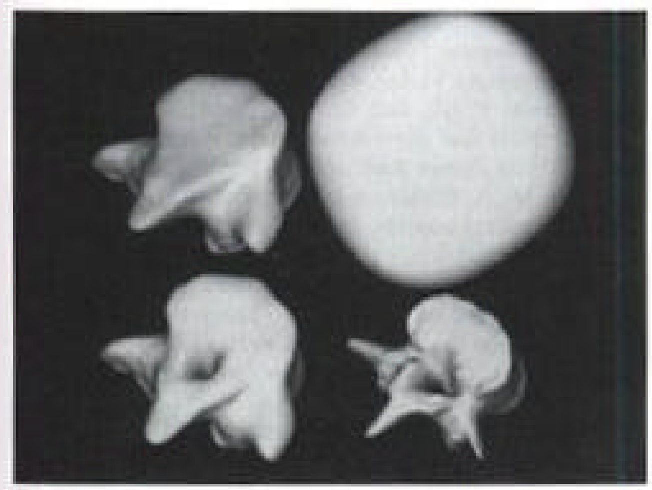
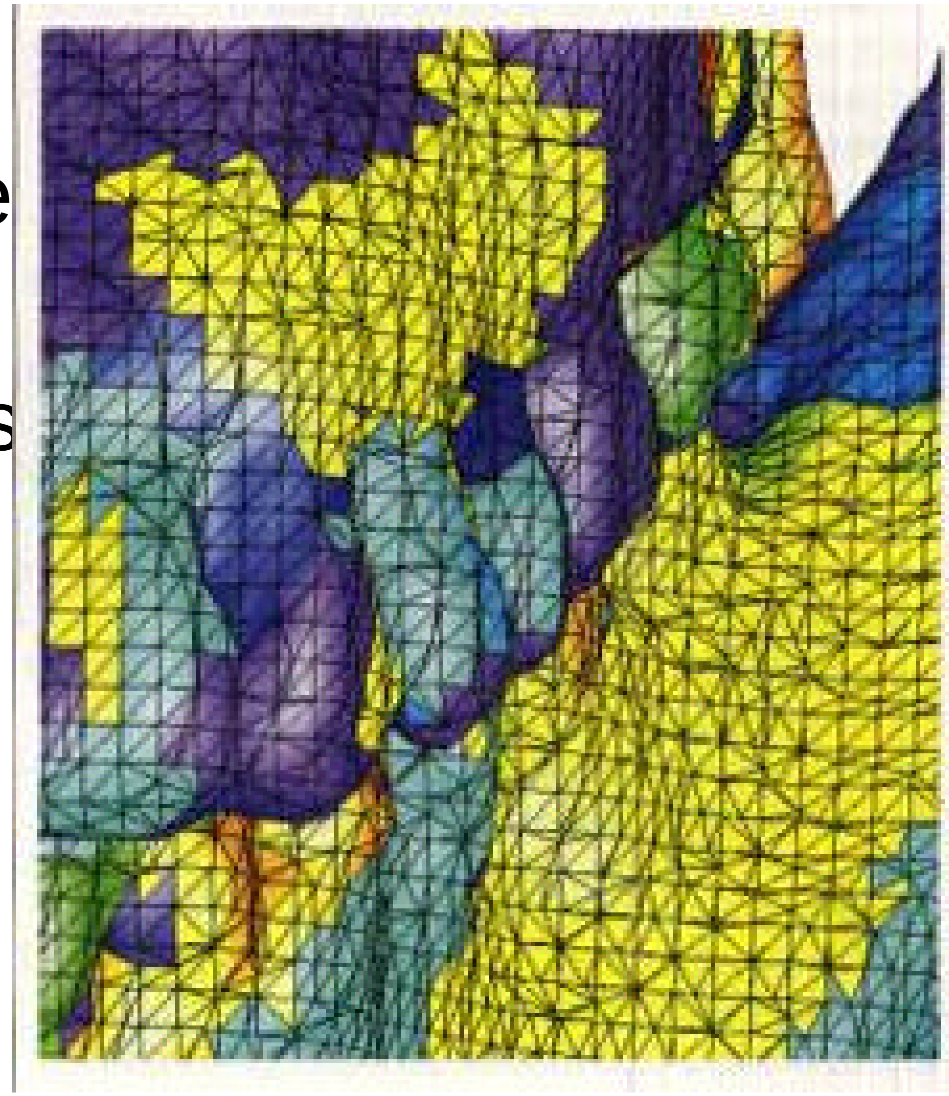


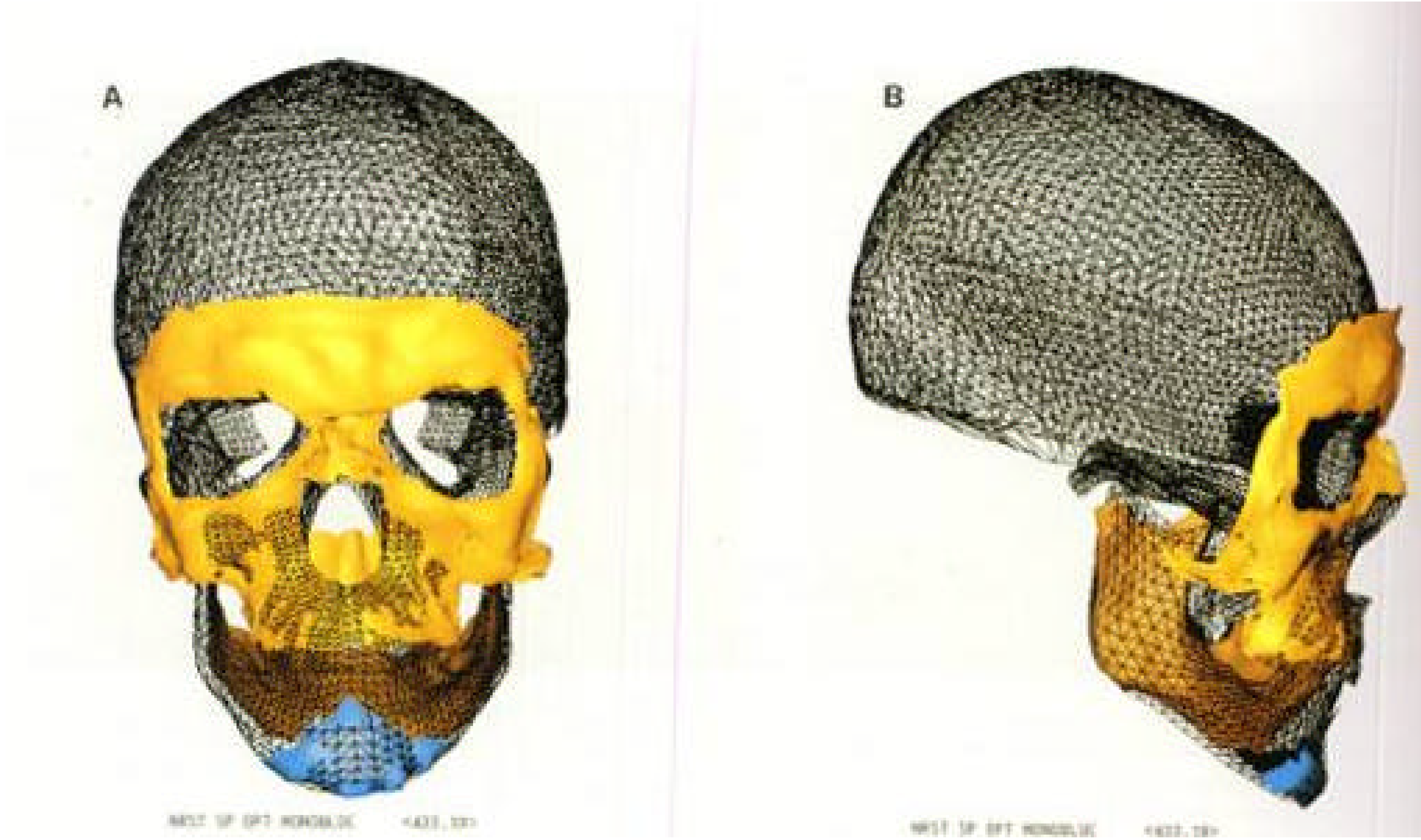
FIGURE 4.7 Segmentation of vertebra defined by a set of CT slices. Four steps of the deformation of a roughly spherical snake spline toward the vertebra are shown.

Source: CIS p 73 (Lavallee image)

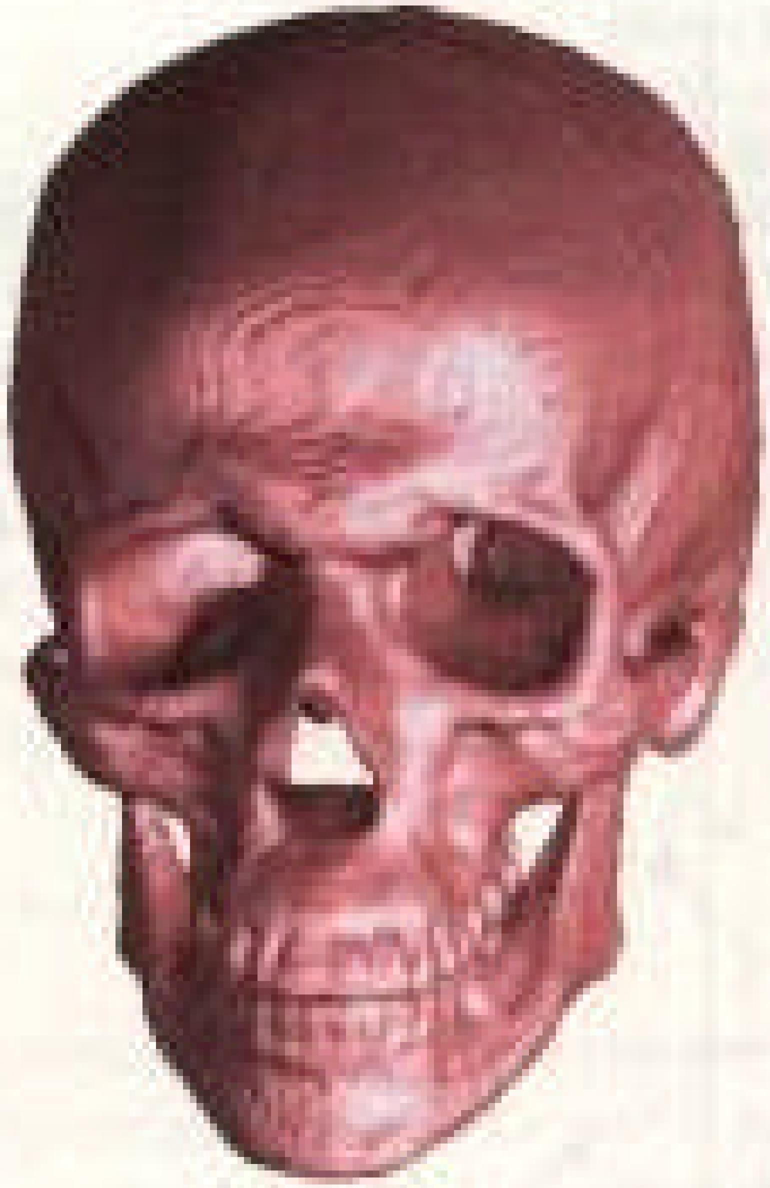
Polyhedral Boundary Reps

- Common in computer graphics
- Many data structures
  - Winged edge
  - Connected triangles
  - etc.



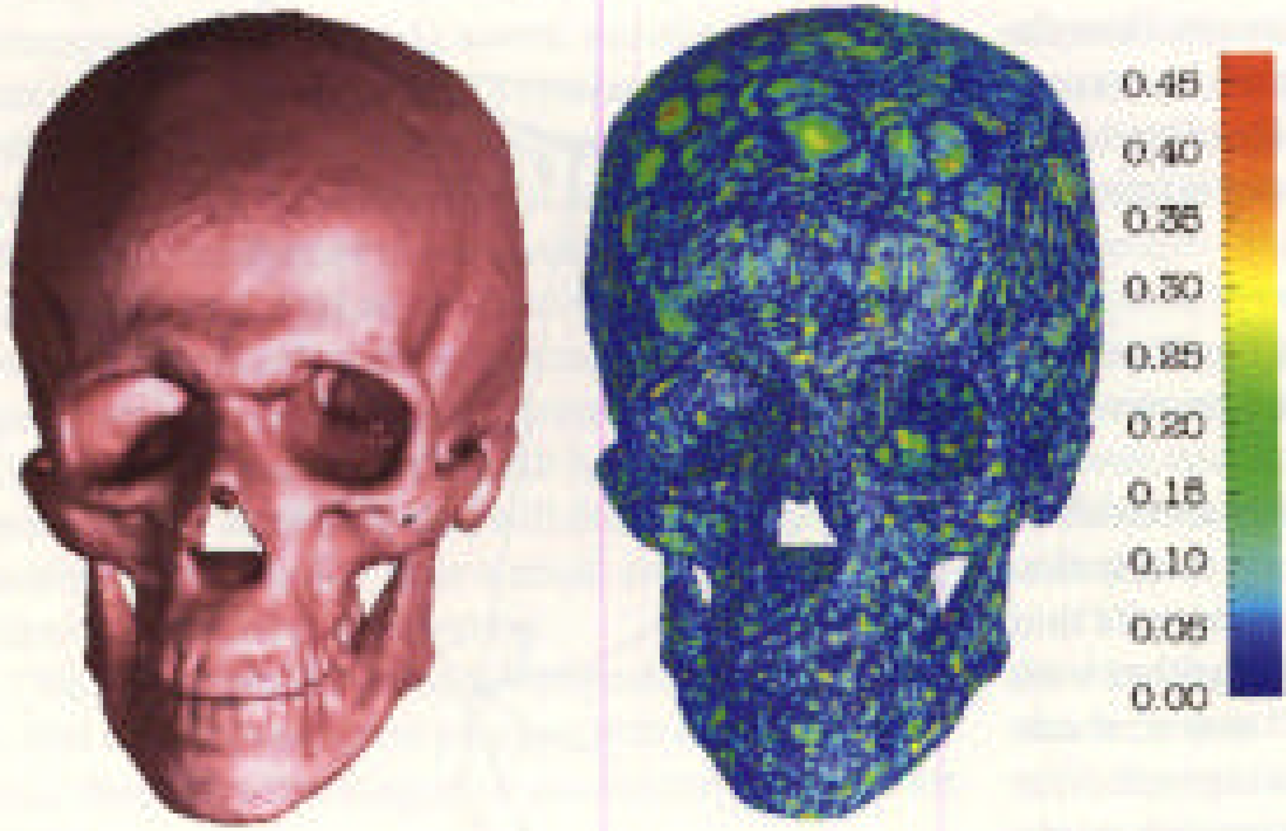


Source: C. Cutting, CIS Book



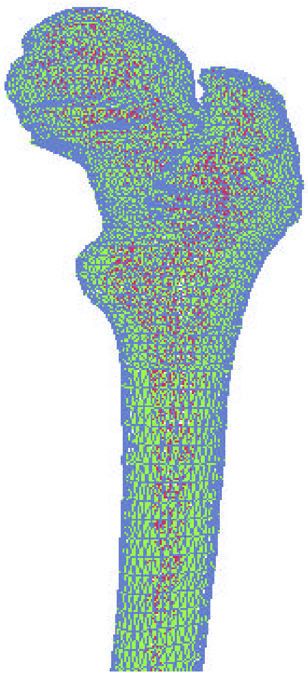
11 Original skull model (349,792 triangles).

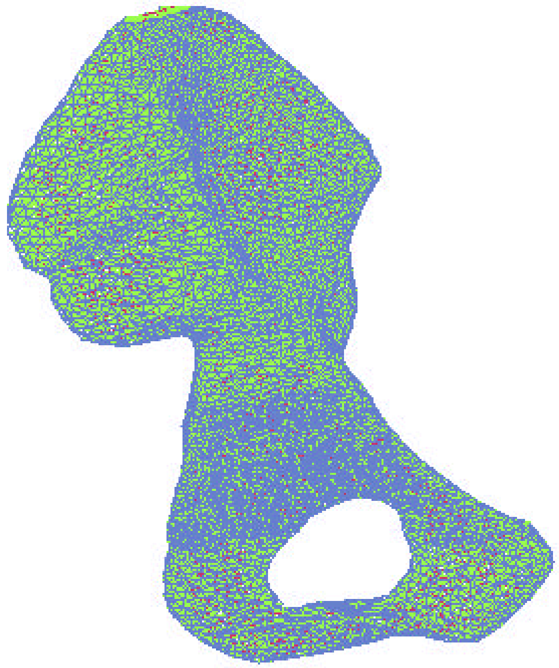
12 Simplified skull (a) mesh and (b) color-coded approximation errors in pixel units: $\varepsilon = 0.5$ (36.60 percent of original triangles).



13 Simplified skull (a) mesh and (b) color-coded approximation errors in pixel units—with aggressive border straightening: $\varepsilon = 0.5$ (15.58 percent of original triangles).

Tetrahedral Mesh Models





Model	Num of Vertices	Num of Tetrahedra	Num of Slices	Total Num of Voxels inside	Avg Num of voxels Per Tetra	Volume (mm^3)	Avg Vol. Per Tetra (mm^3)
Femur	6163	31,537	83	1,802,978	57.1	312,107	9.9
Pelvis	8219	32,741	110	1,941,998	59.3	347,070	10.6

Density Functions

- Advantages
  - Efficient in storage
  - Continuous function
  - explicit form
  - convenient to integrate, to differentiate, to interpolate, and to deform

Density Functions (cont')

n-degree Bernstein polynomial in barycentric coordinates

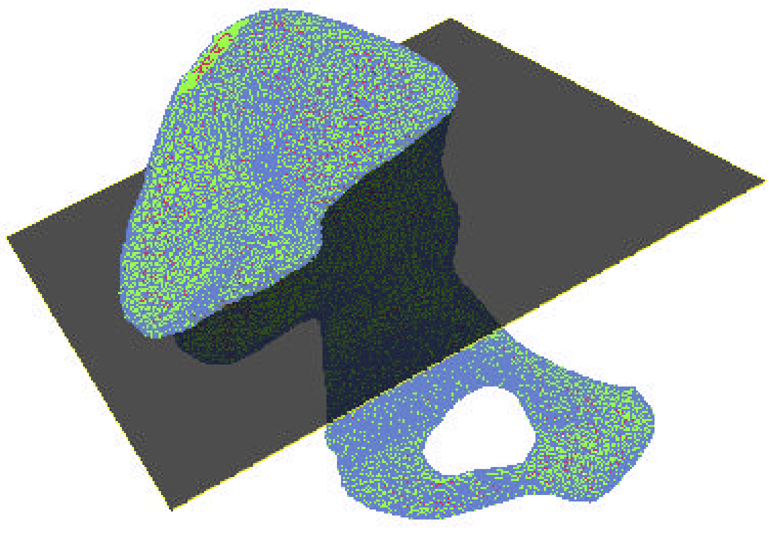
$$D(m) = \sum_{i+j+k+l=n}^{n} C_{i,j,k,l} B^n_{i,j,k,l}(m)$$

$C_{i,j,k,l}$ polynomial coefficient

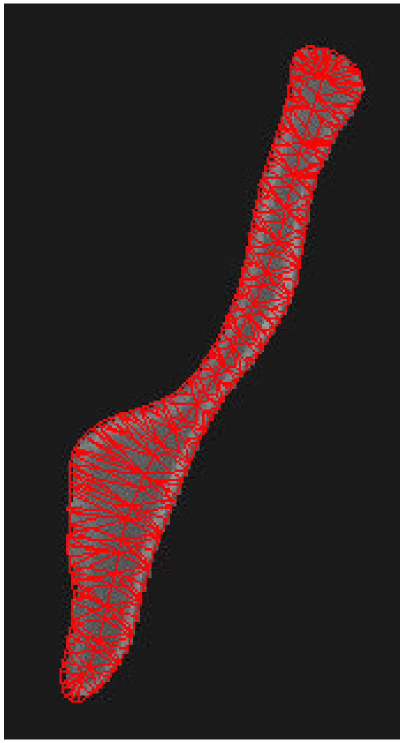
$$B^n_{i,j,k,l}(m) = \frac{n!}{i!\,j!\,k!\,l!} m_x^i m_y^j m_z^k m_w^l$$ barycentric Bernstein basis

Accuracy vs Degree of Density Function

- Use CT data set as ground truth
- cut an arbitrary plane through the model

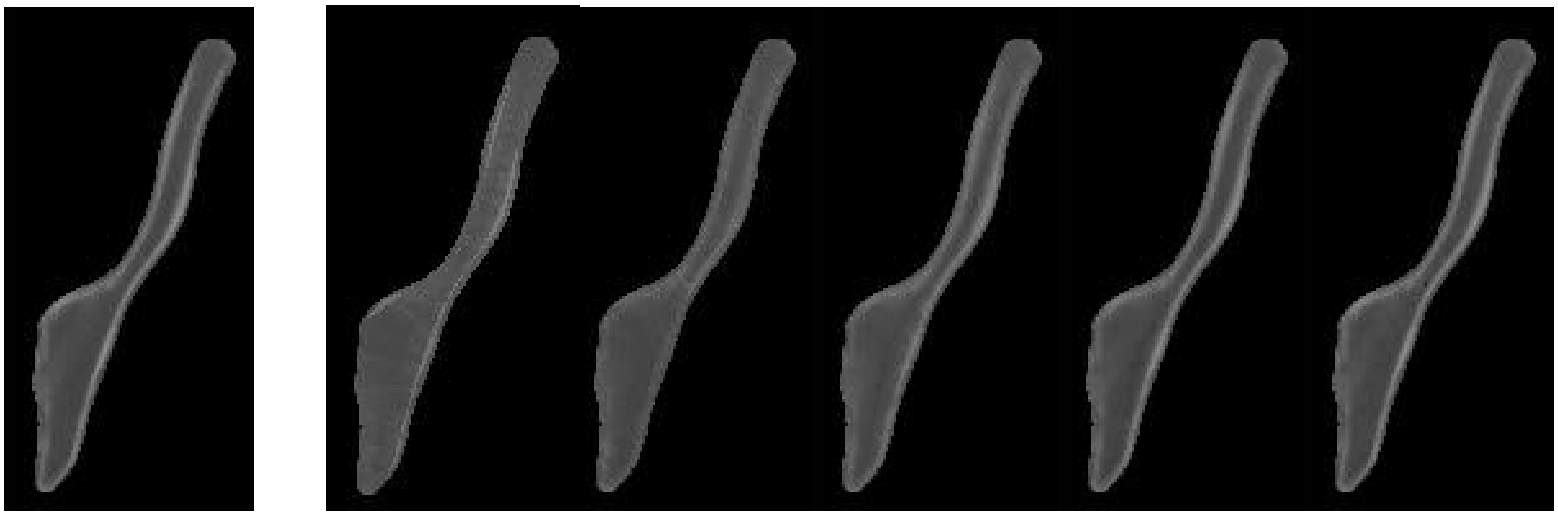


Cutting Plane



Partitions

Accuracy vs Degree of Density Function (cont')



Ground Truth n=0 n=1 n=2 n=3 n=4

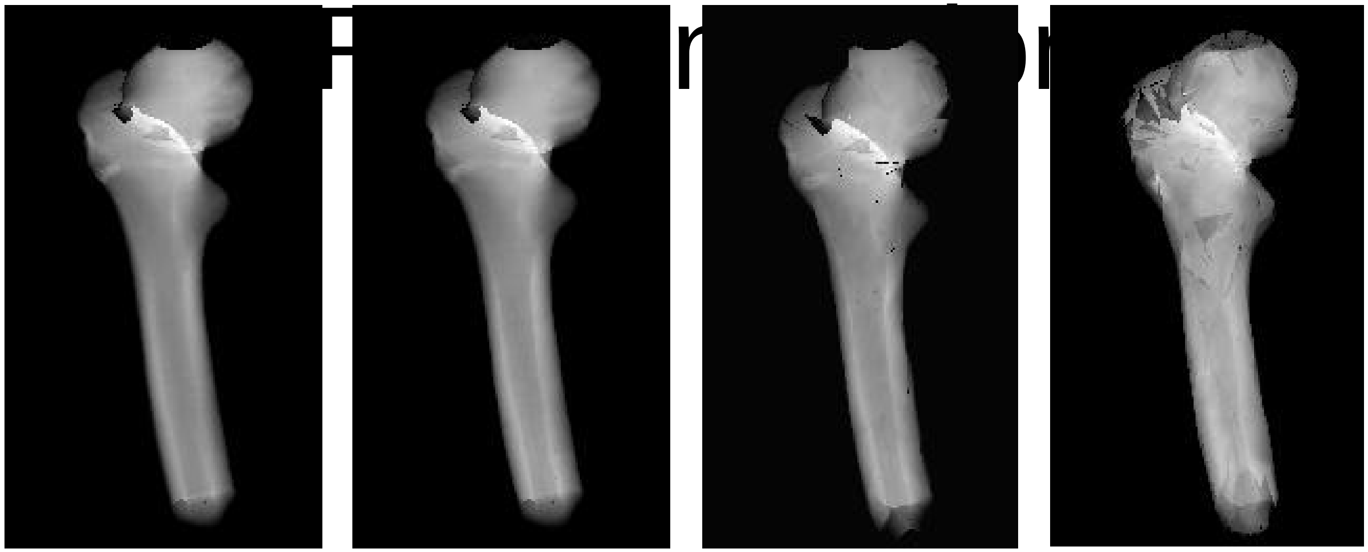
Degree	0	1	2	3	4	5	6	7	8
Coeff Number	1	4	10	20	35	56	84	120	165
Avg. Density Err (%)	3.291	1.583	0.766	0.442	0.298	0.216	0.167	0.149	0.128

Multiple resolution femur model

Level	Num of Verts	Num of Tetras	Avg Density Diff (%)	Std Dev of Density Diff (%)	Avg Vol. Per Tetra (mm^3)	Avg Num of voxels Per Tetra	Storage in CT image (bytes)	Storage in Tetra model (bytes)
1	6163	31537	2.7%	1.9%	9.896	57	3,605,956	1,840,028
2	2350	12719	4.9%	3.1%	23.4	105	3,605,956	740,464
3	446	2448	8.4%	6.3%	159.9	681	3,605,956	142,440

Multiple resolution half pelvis model

Level	Num of Verts	Num of Tetras	Avg Density Diff (%)	Std Dev of Density Diff (%)	Avg Vol. Per Tetra (mm^3)	Avg Num of voxels Per Tetra	Storage in CT image (bytes)	Storage in Tetra model (bytes)
1	8219	32741	3.1%	2.2%	10.6	59.3	3,883,996	1,932,124
2	1272	6138	6.4%	3.8%	55.2	316.4	3,883,996	358,992
3	512	2438	8.8%	6.5%	137.9	796.6	3,883,996	142,672



	Num of Tetra	Running time	Avg. elems Passed through	Avg Intensity Diff (%)	Std Dev of Intensity Diff (%)
CT Data set	N/A	29.4 s	132.6 voxels	N/A	N/A
Density Model	31537	9.2s	43.1 tetras	3.2%	2.4%
Density Model	12719	5.6 s	21.8 tetras	7.6%	5.7%
Density Model	2448	1.9 s	7.3 tetras	14.4%	10.3%

Registration

Overall Goal: Given two coordinate systems,

$$\mathbf{Ref}_\mathbf{A} \;\&\; \mathbf{Ref}_\mathbf{B}$$

and coordinates

$$\mathbf{x}_\mathbf{A} \;\&\; \mathbf{x}_\mathbf{B}$$

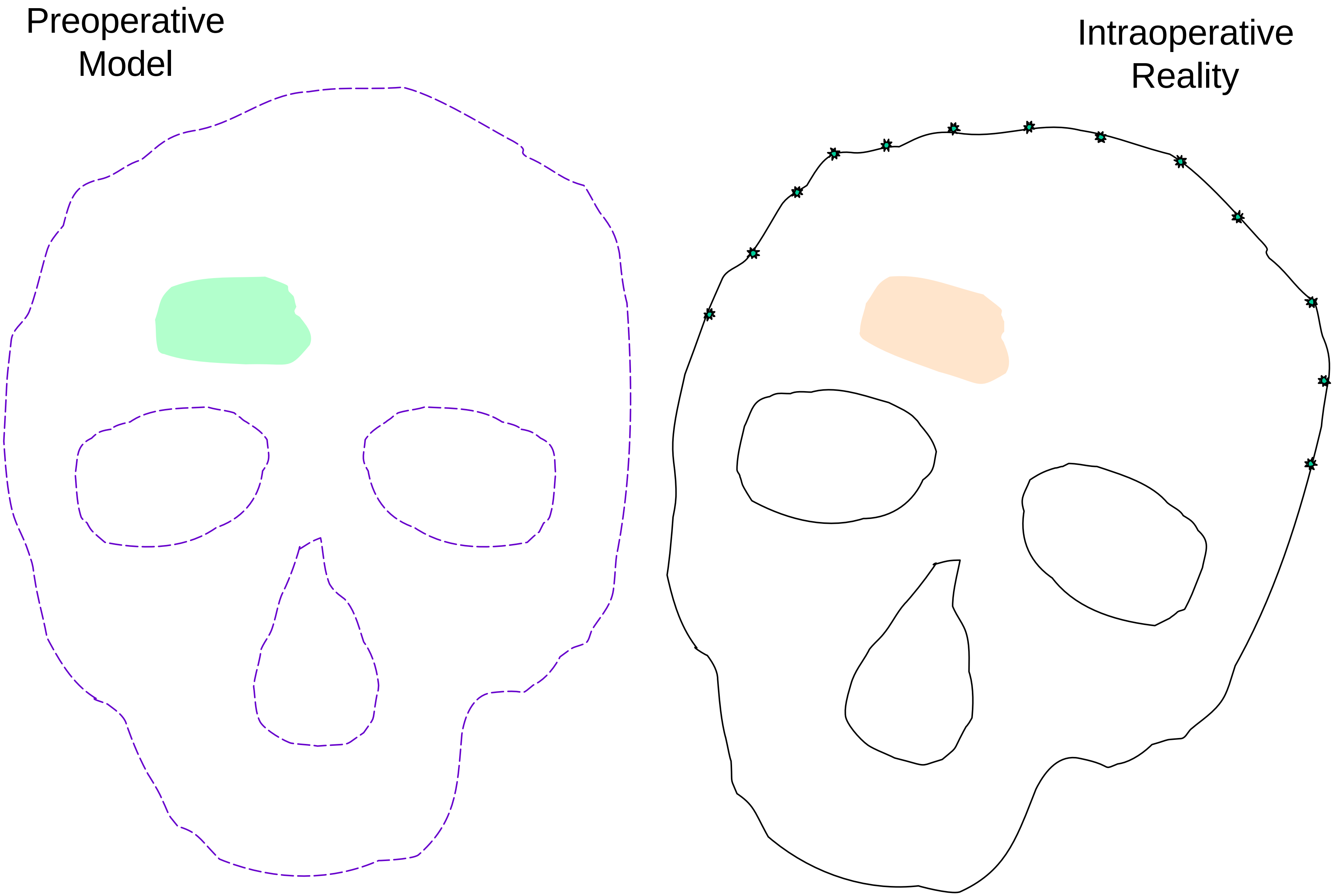
associated with corresponding features in the two coordinate systems, the general goal is to determine a transformation function T that transforms one set of coordinates into the other:

$$\mathbf{x}_\mathbf{A} = \mathbf{T}(\mathbf{x}_\mathbf{B})$$

What needs registering?

- **Preoperative Data**
  - 2D & 3D medical images
  - Models
  - Preoperative positions
- **Intraoperative Data**
  - 2D & 3D medical images
  - Models
  - Intraoperative positioning information
- **The Patient**

A typical registration problem



Definitions

- **Rigid Transformation:** Essentially, our old friends 2D & 3D coordinate transformations:

  $$\mathbf{T(x) = R \bullet x + p}$$

  The key assumption is that deformations may be neglected.

- **Elastic Transformation:** Cases where must take deformations into account. Many different flavors, depending on what is being deformed

Uses of Rigid Transformations

- Register (approximately) multiple image data sets
- Transfer coordinates from preoperative data to reality (especially in orthopaedics & neurosurgery)
- Initialize non-rigid transformations

Uses of Elastic Transformations

- Register different patients to common data base (e.g., for statistical analysis)
- Overlay atlas information onto patient data
- Study time-varying deformations
- Assist segmentation

Typical Features

- Point fiducials
- Point anatomical landmarks
- Ridge curves
- Contours
- Surfaces
- Line fiducials

Sampled 3D data to surface models

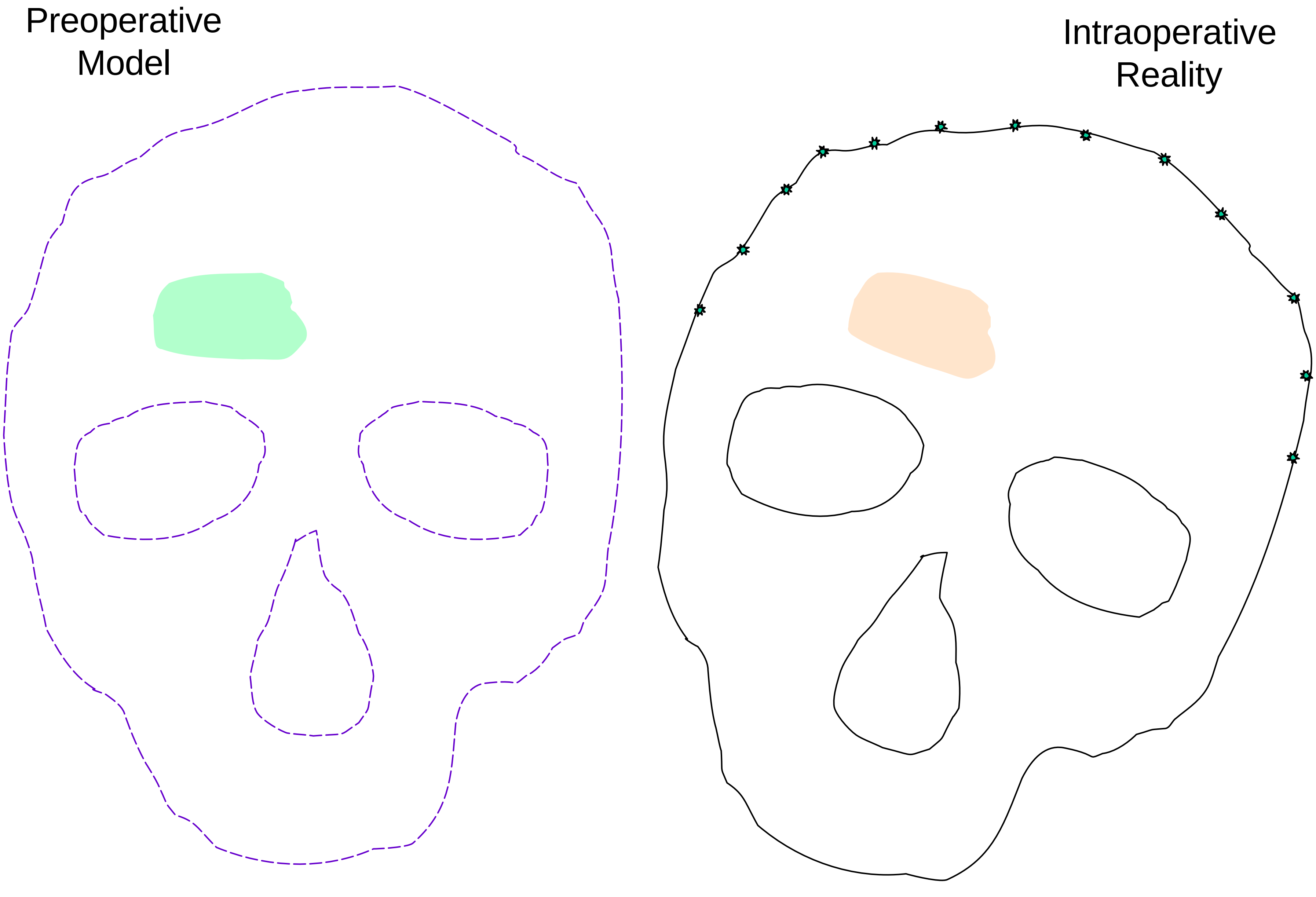
Outline:

- Select large number of sample points
- Determine distance function $d_S(\mathbf{f}, \mathcal{F})$ for a point $\mathbf{f}$ to a surface feature $\mathcal{F}$.
- Use d_S to develop disparity function D.

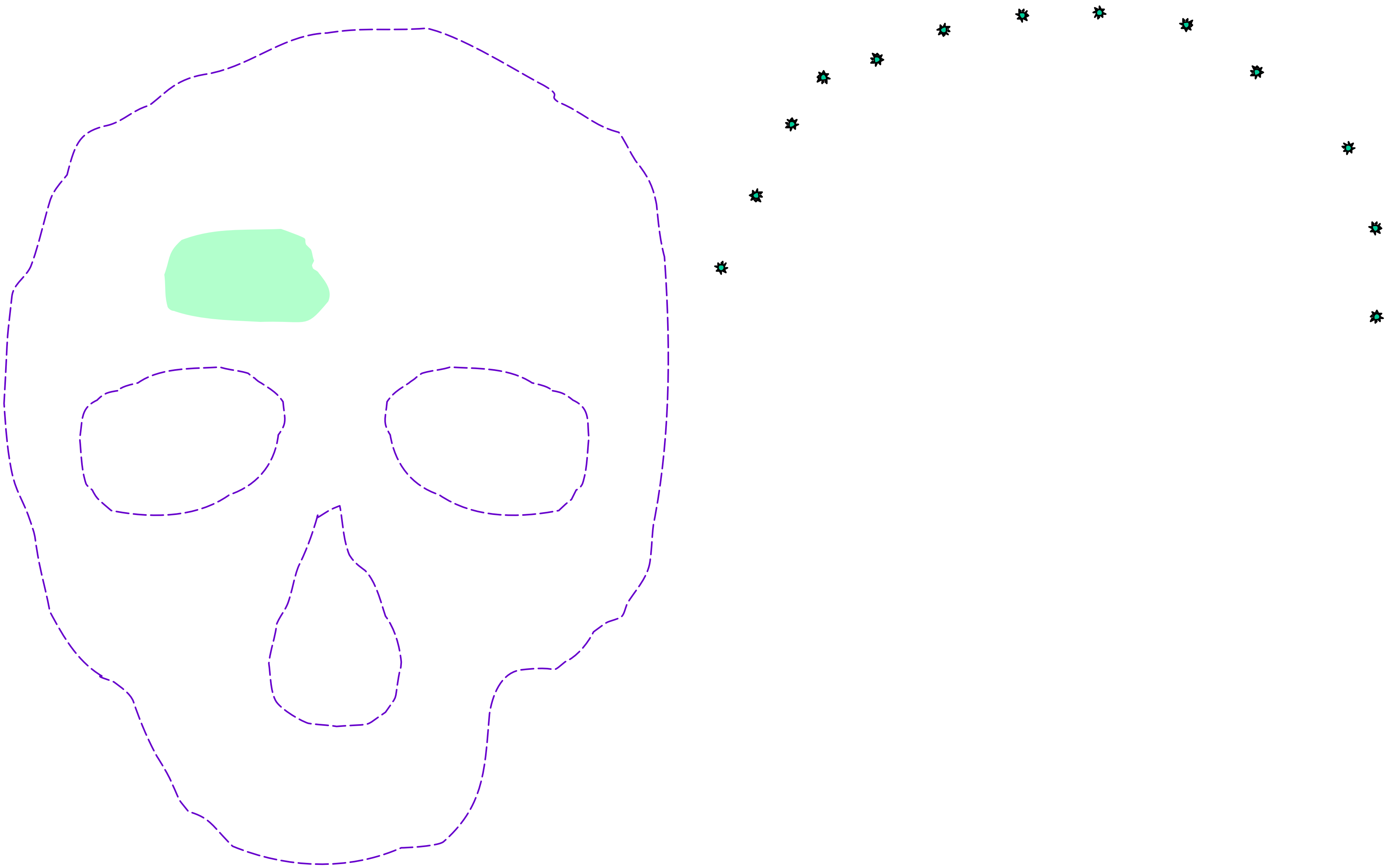
Examples

- Head-in-hat algorithm [Levin et al., 1988; Pelizzari et al., 1989]
- Distance maps [e.g., Lavallee et al]
- Iterative closest point [Besl and McKay, 1992]

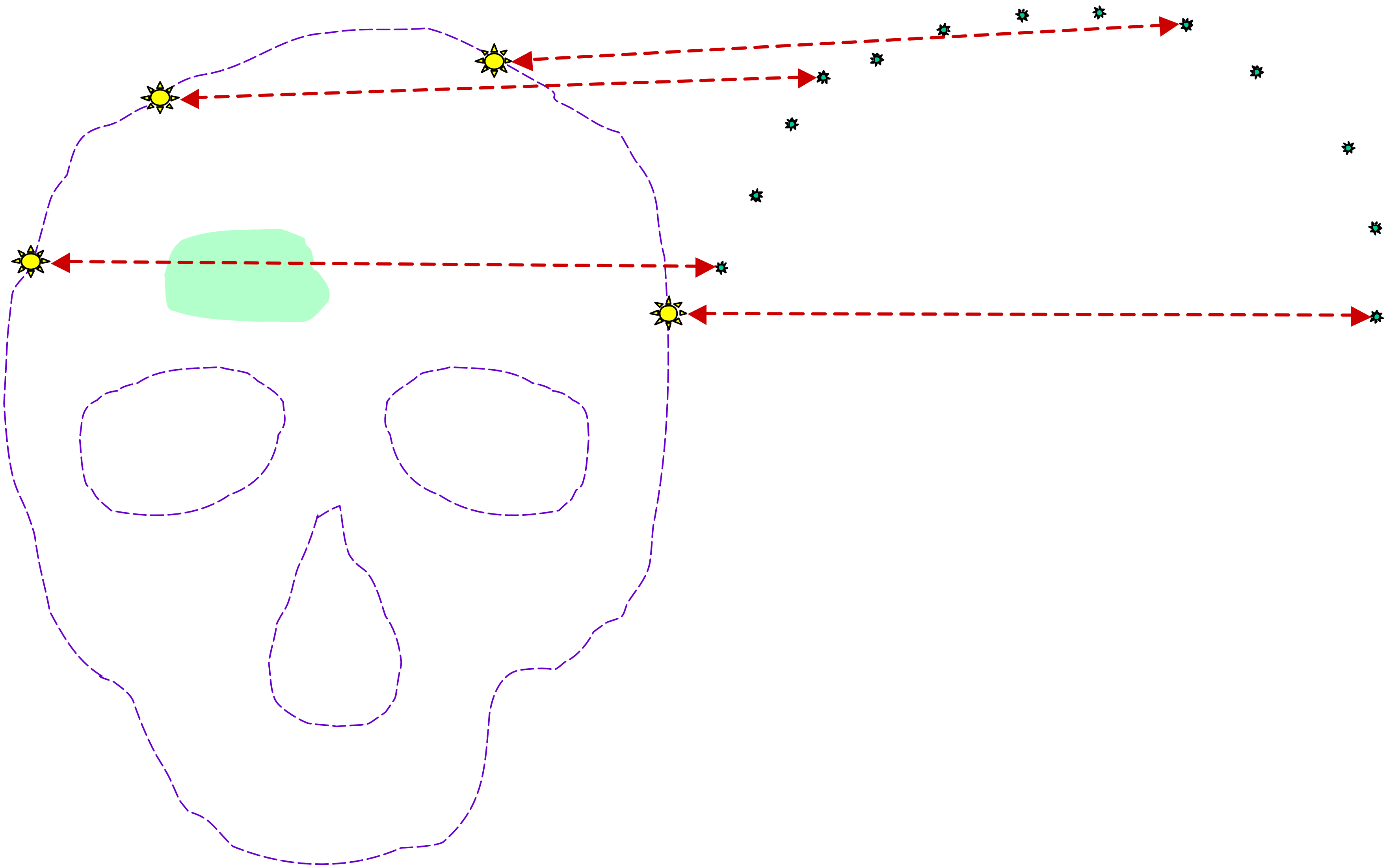
A typical registration problem



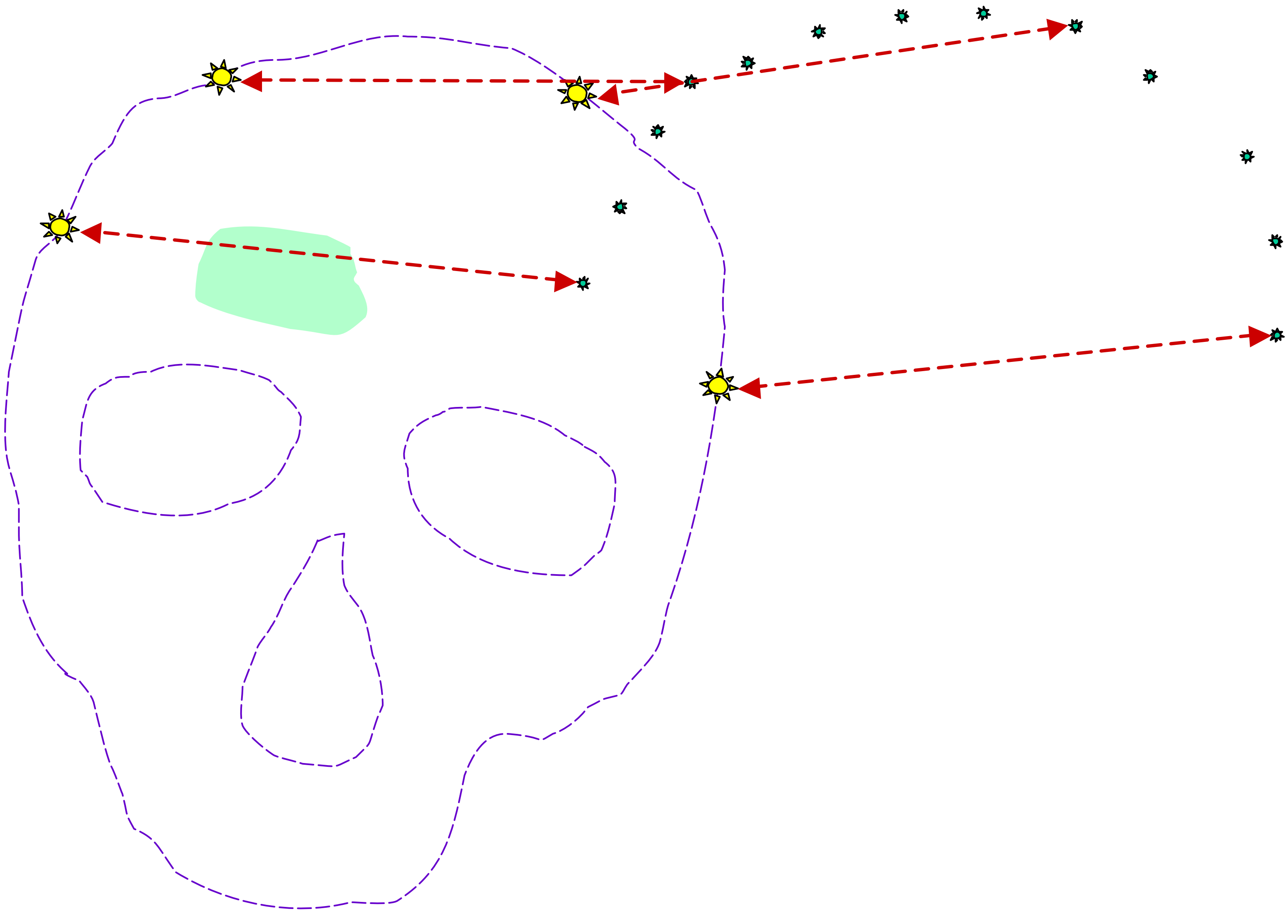
What the computer knows



Find corresponding points & pull!



Find corresponding points & pull!

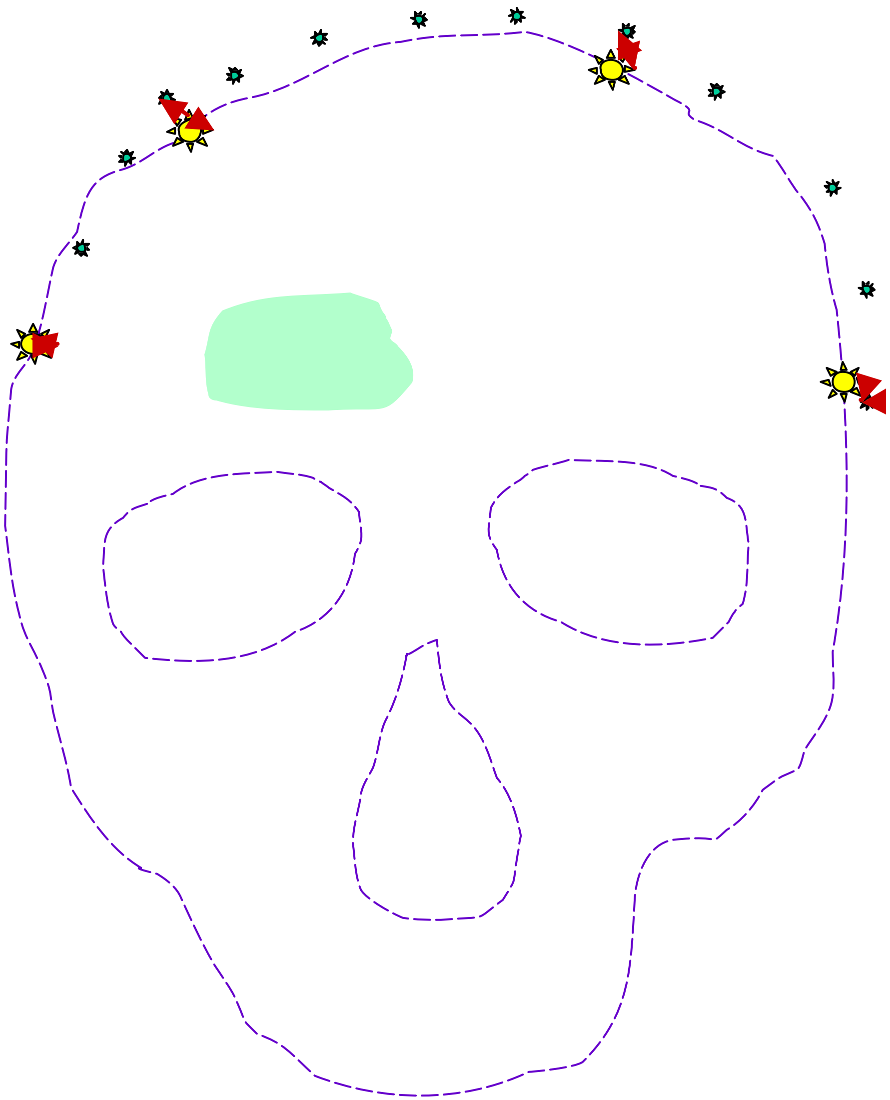


Find corresponding points & pull!

Iterate this until converge

Find new point pairs every iteration

Key challenge is finding point pairs efficiently.



Head in Hat Algorithm

- Levin et al, 1988; Pelizzari et al, 1989
- Origially used for Pet-to-MRI/CT registration
- Given $\mathbf{f}_i \in \mathcal{F}_A$, and a surface model $\mathcal{F}_B$, computes a rigid transformation $\mathbf{T}$ that minimizes

$$D = \sum_i [d_S(\mathcal{F}_B, \mathbf{T} \cdot \mathbf{f}_i)]^2$$

  where d_S is defined below, given a good initial guess for $\mathbf{T}$.

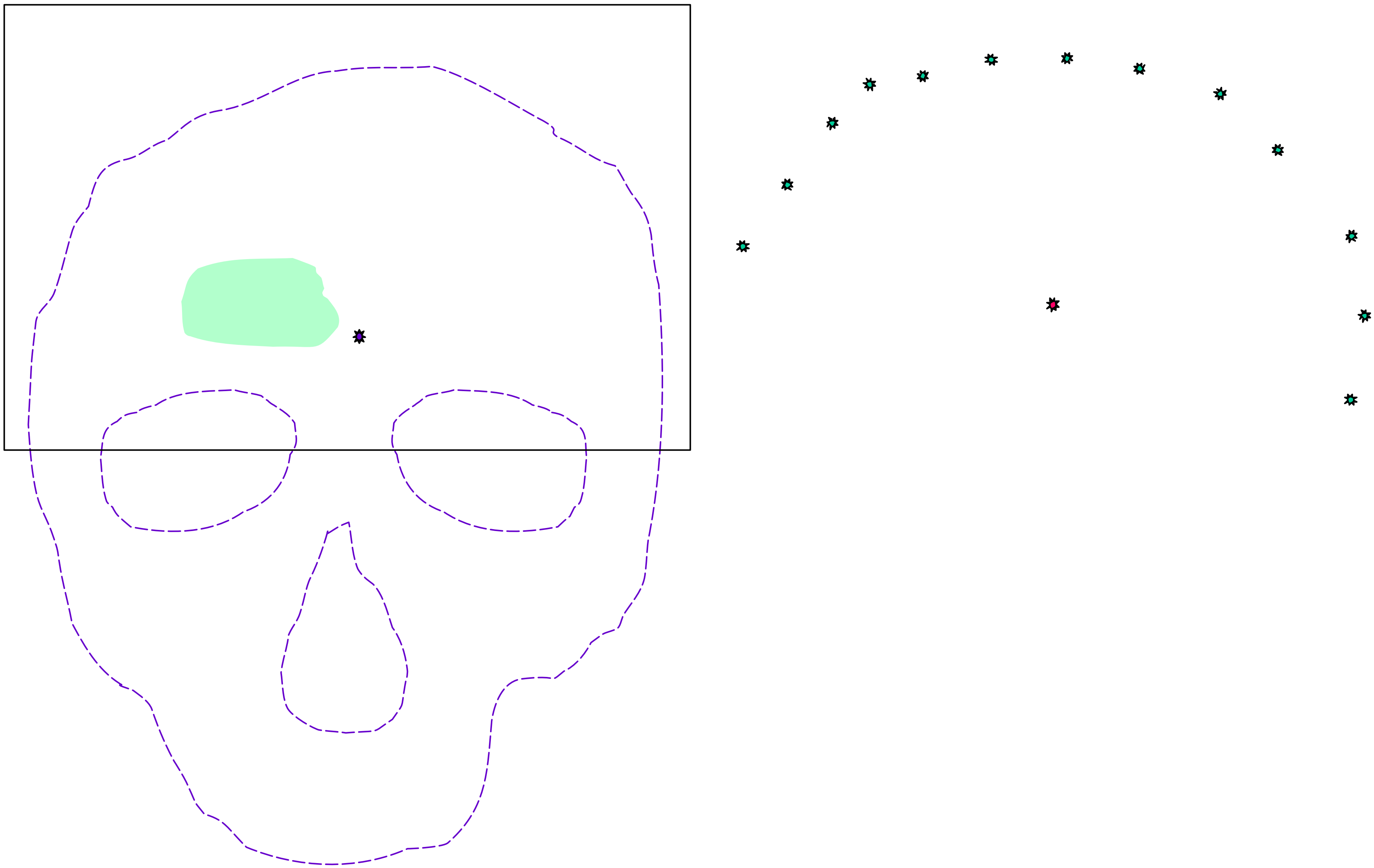
- Optimization uses standard numerical method (steepest gradient descent [Powell]) to find six parameters (3 rotations, 3 translations) defining $\mathbf{T}$.

Head in Hat Algorithm

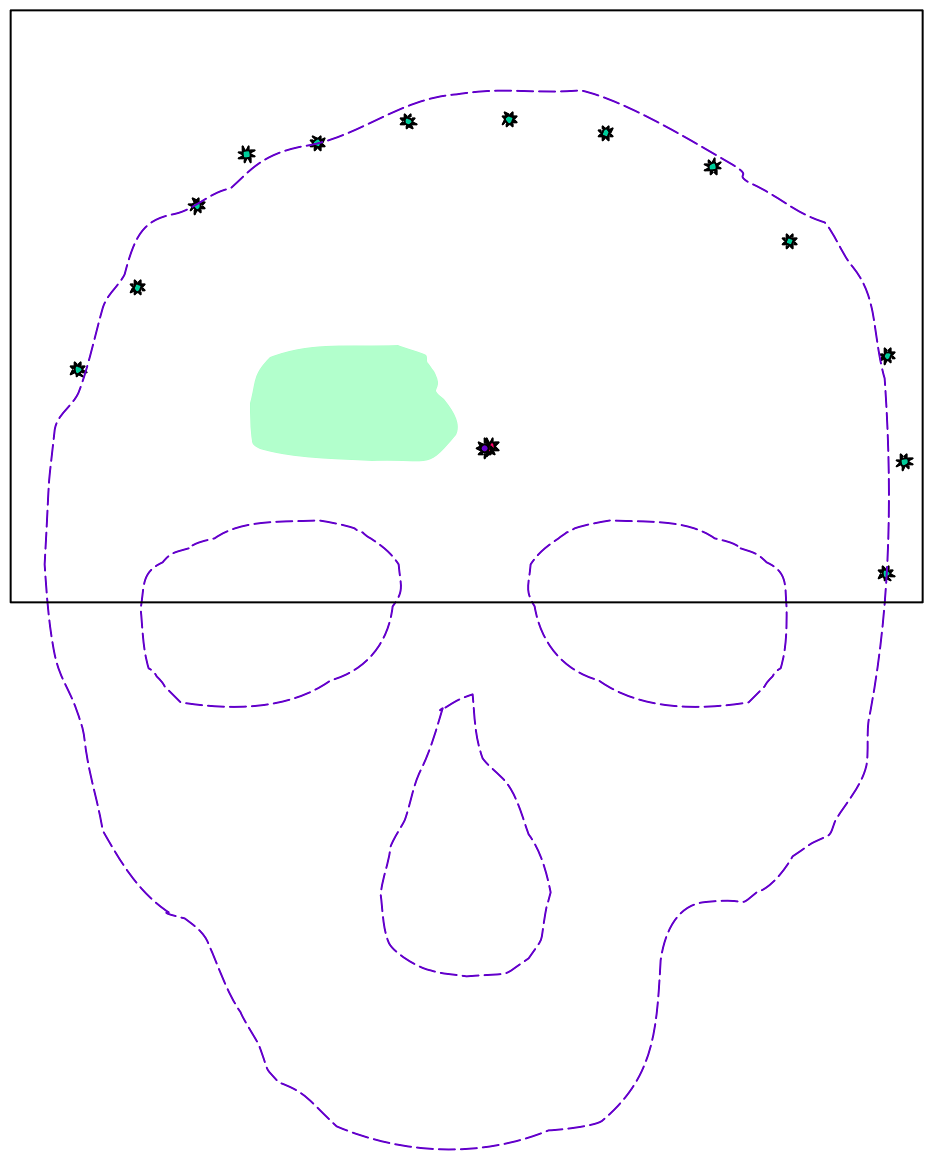
Definition of $d_S(\mathcal{F}_B, \mathbf{f}_i)$

1. Compute centroid $\mathbf{g}_B$ of surface $\mathcal{F}_B$.
2. Determine a point $\mathbf{q}_i$ that lies on the intersection of the line $\mathbf{g}_B - \mathbf{f}_i$ and $\mathcal{F}_B$.
3. Then, $d_S(\mathcal{F}_B, \mathbf{f}_i) = \|\mathbf{q}_i - \mathbf{f}_i\|$

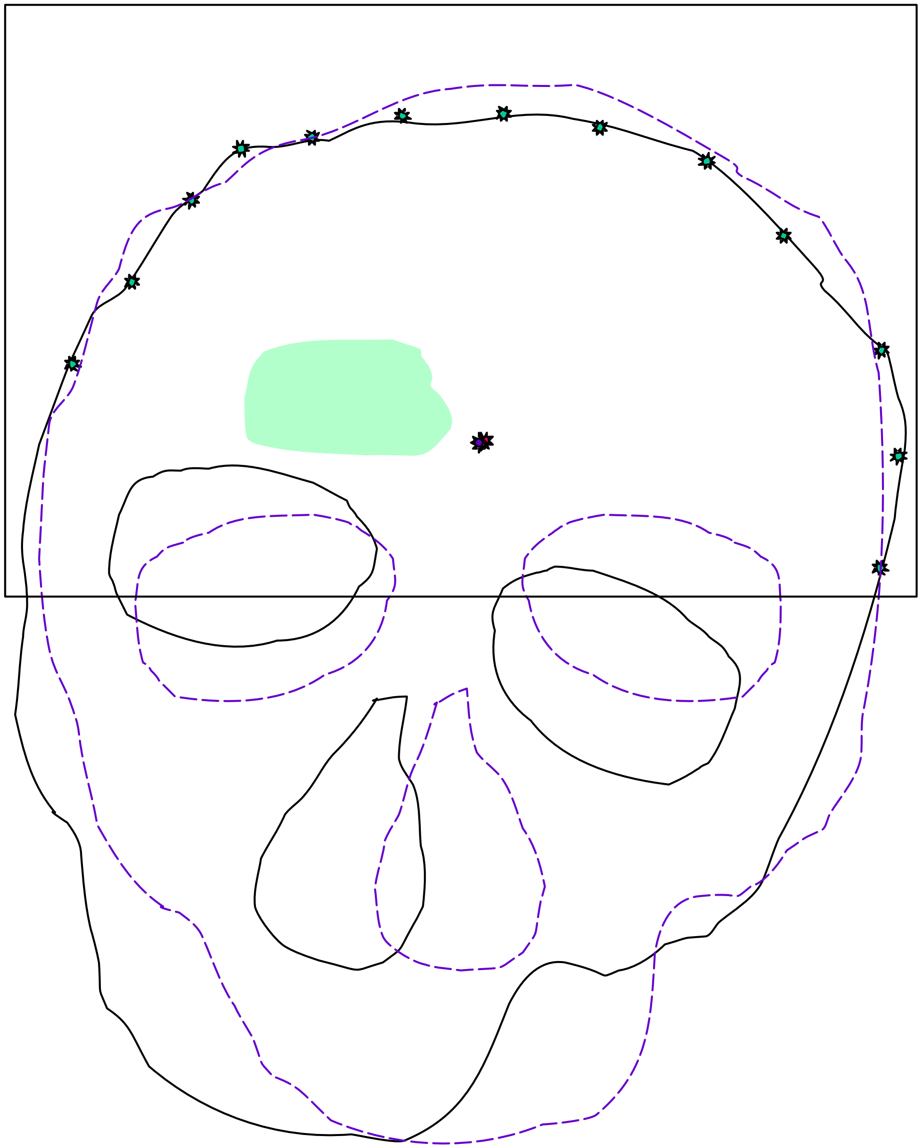
Head-in-hat algorithm: step 0



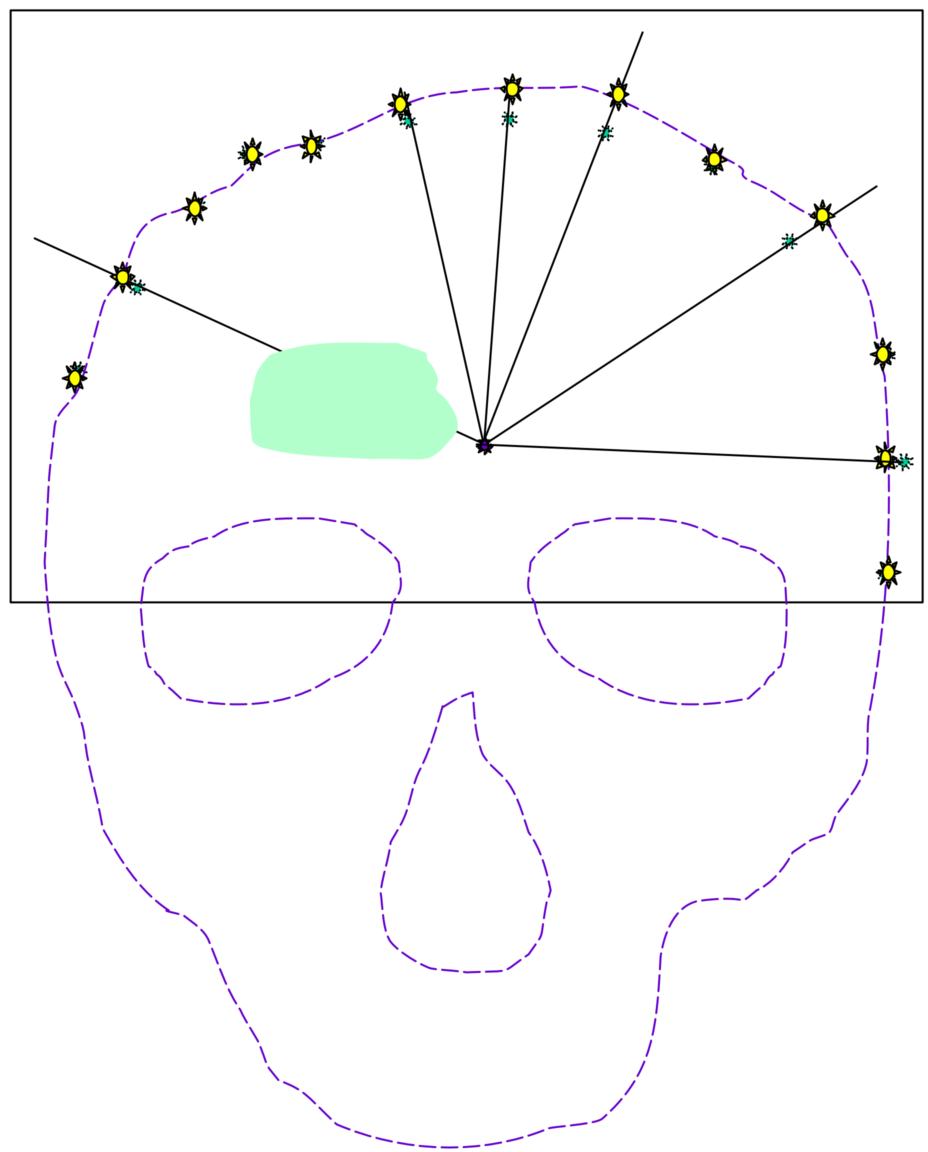
Head-in-hat algorithm: step 1



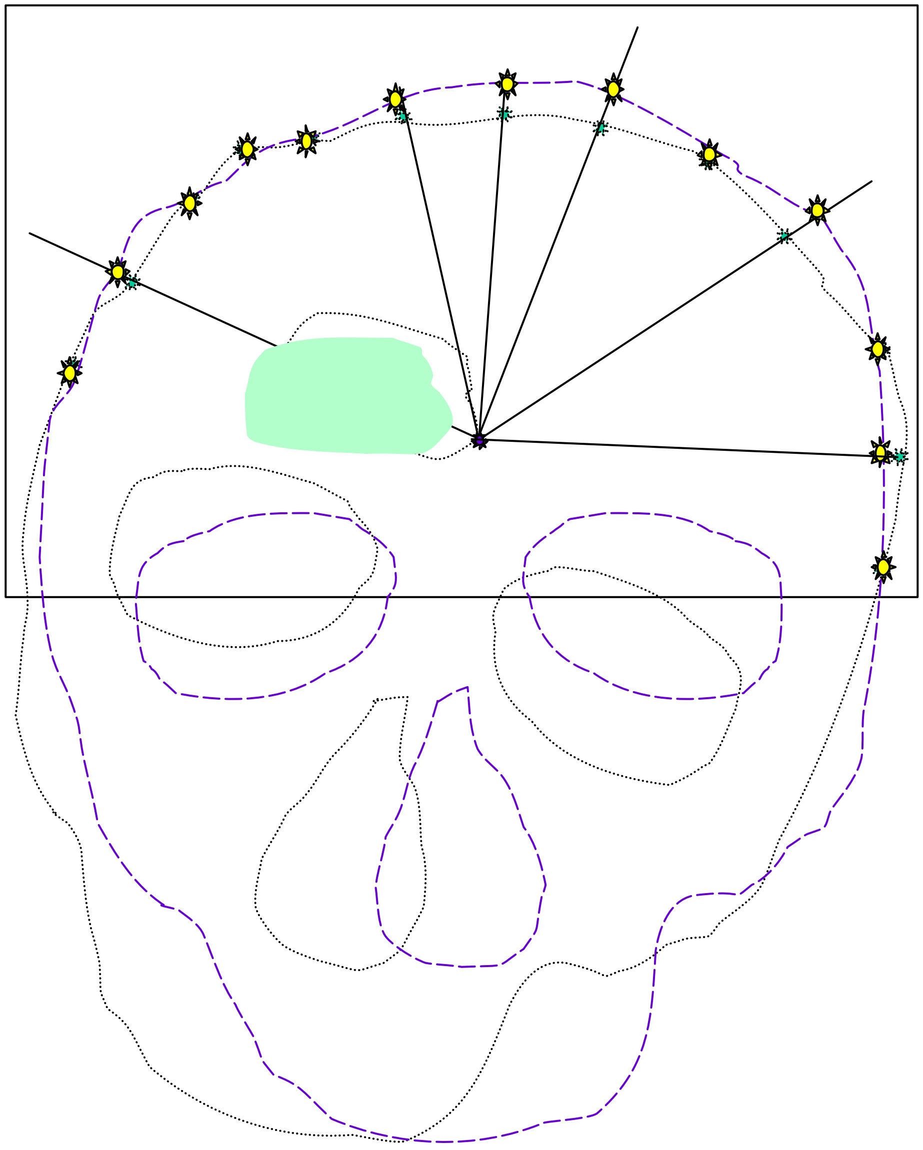
Head-in-hat algorithm: step 1



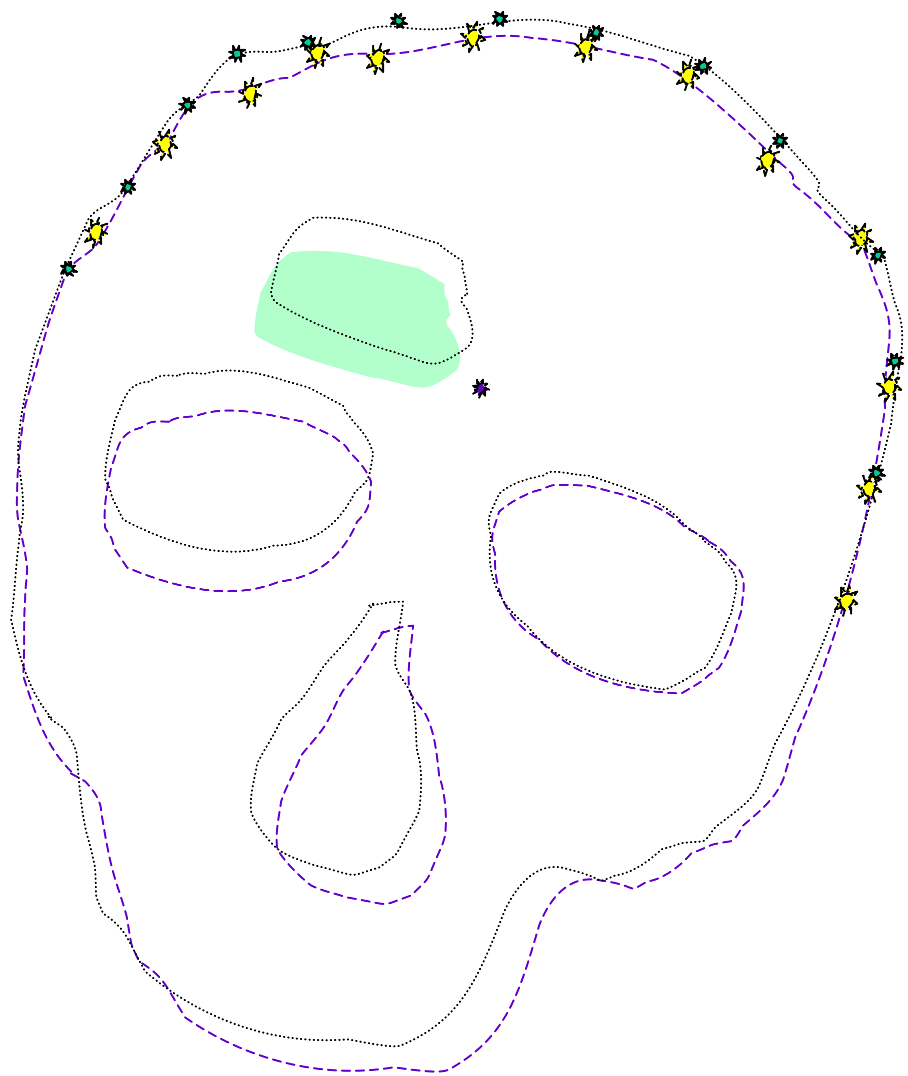
Head-in-hat algorithm: step 2



Head-in-hat algorithm: step 2



Head-in-hat algorithm: step 3



Head in Hat Algorithm

- **Strengths**
  - Moderately straightforward to implement
  - Slow step is intersecting rays with surface model
  - Works reasonably well for original purpose (registration of skin of head) if have adequate initial guess

- **Weaknesses**
  - Local minima
  - Assumptions behind use of centroid
  - Requires good initial guess and close matches during convergence

Iterative Closest Point

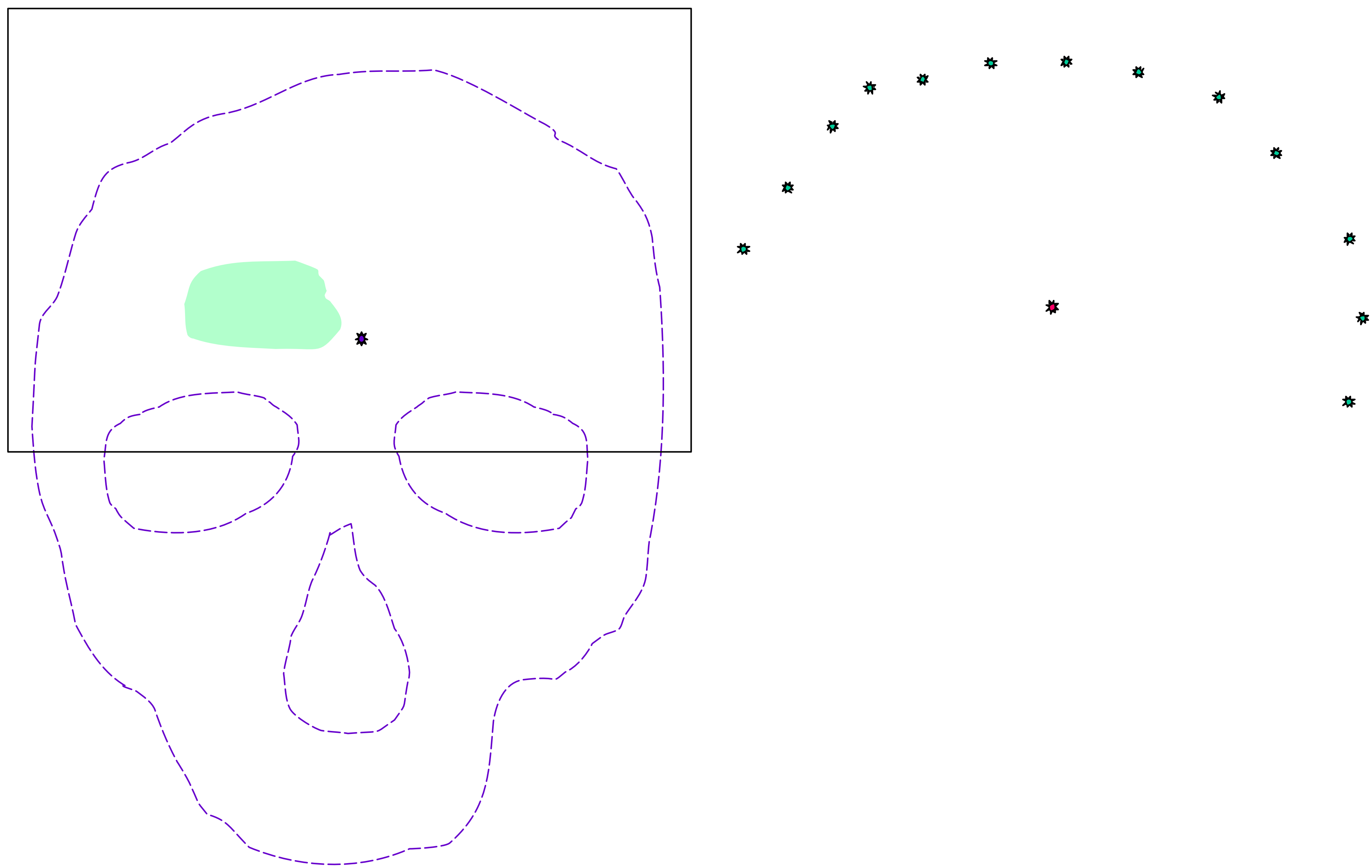
- Besl and McKay, 1992
- Start with an initial guess, $\mathbf{T}_0$, for $\mathbf{T}$.
- At iteration k

  1. For each sampled point $\mathbf{f}_i \in \mathcal{F}_A$, find the point $\mathbf{v}_i \in \mathcal{F}_B$ that is closest to $\mathbf{T}_k \cdot \mathbf{f}_i$.
  2. Then compute $\mathbf{T}_{k+1}$ as the transformation that minimizes

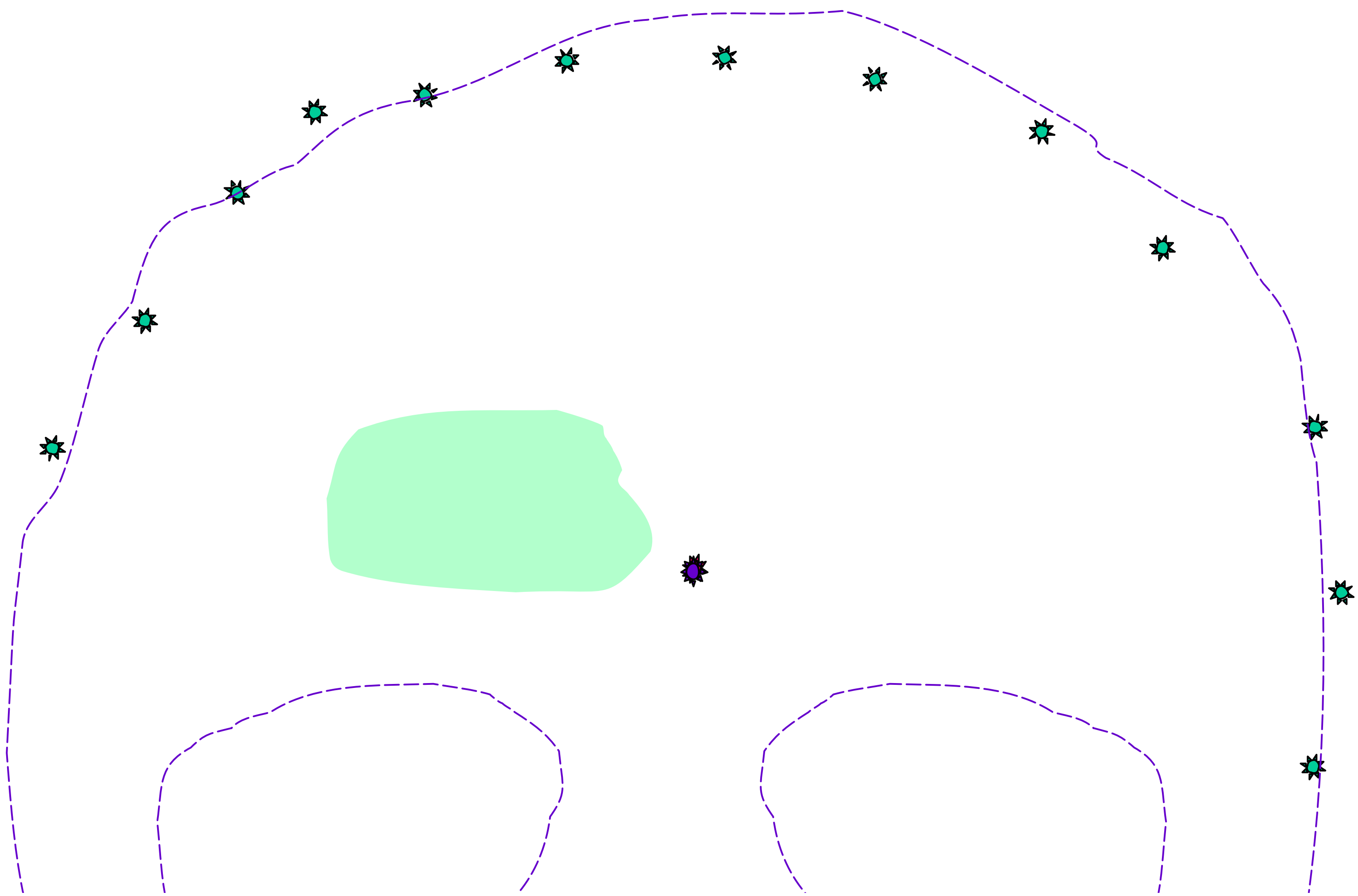
$$D_{k+1} = \sum_i \|\mathbf{v}_i - \mathbf{T}_{k+1} \cdot \mathbf{f}_i)\|^2$$

- Physical Analogy

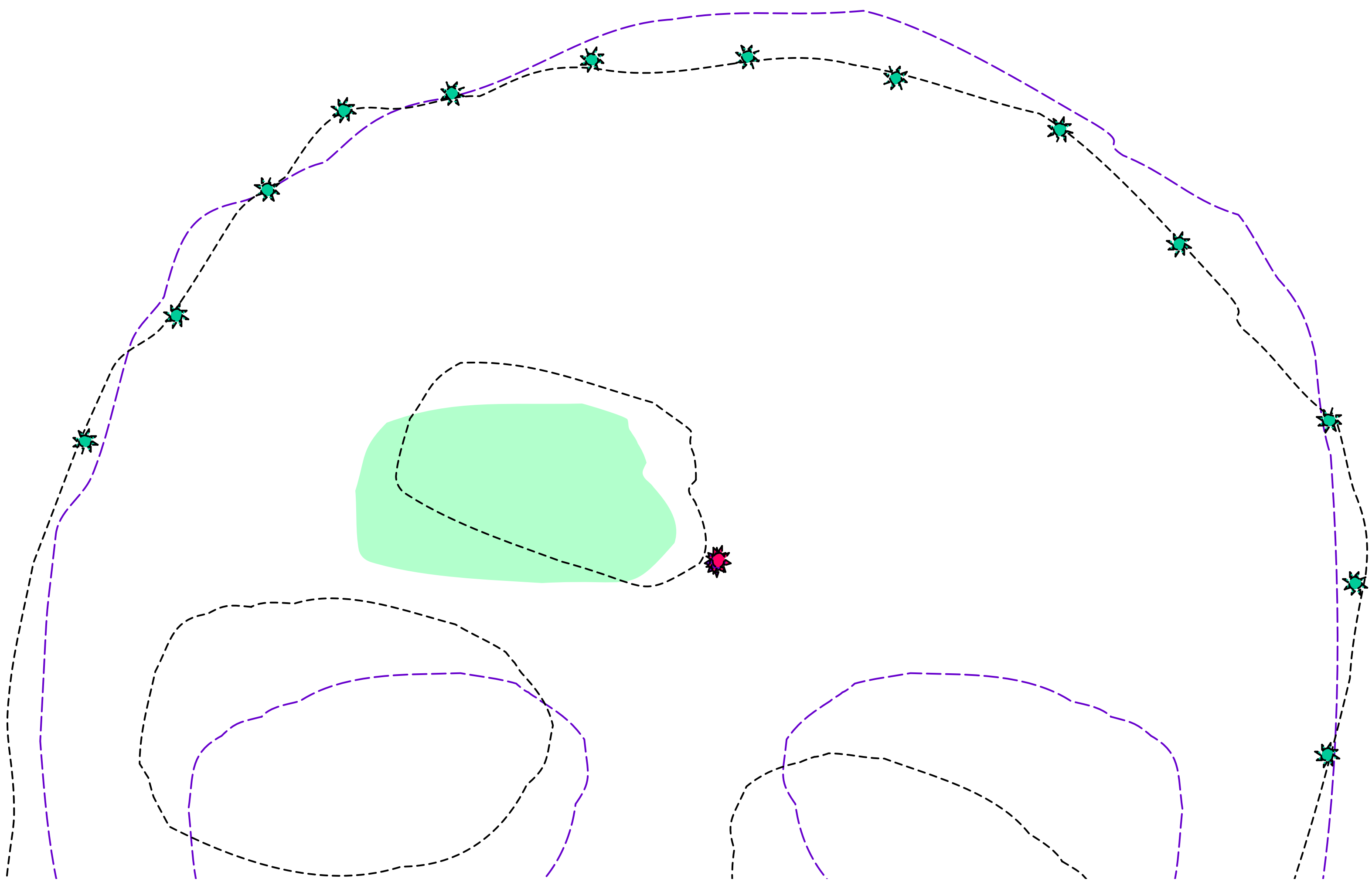
Iterative Closest Point: step 0



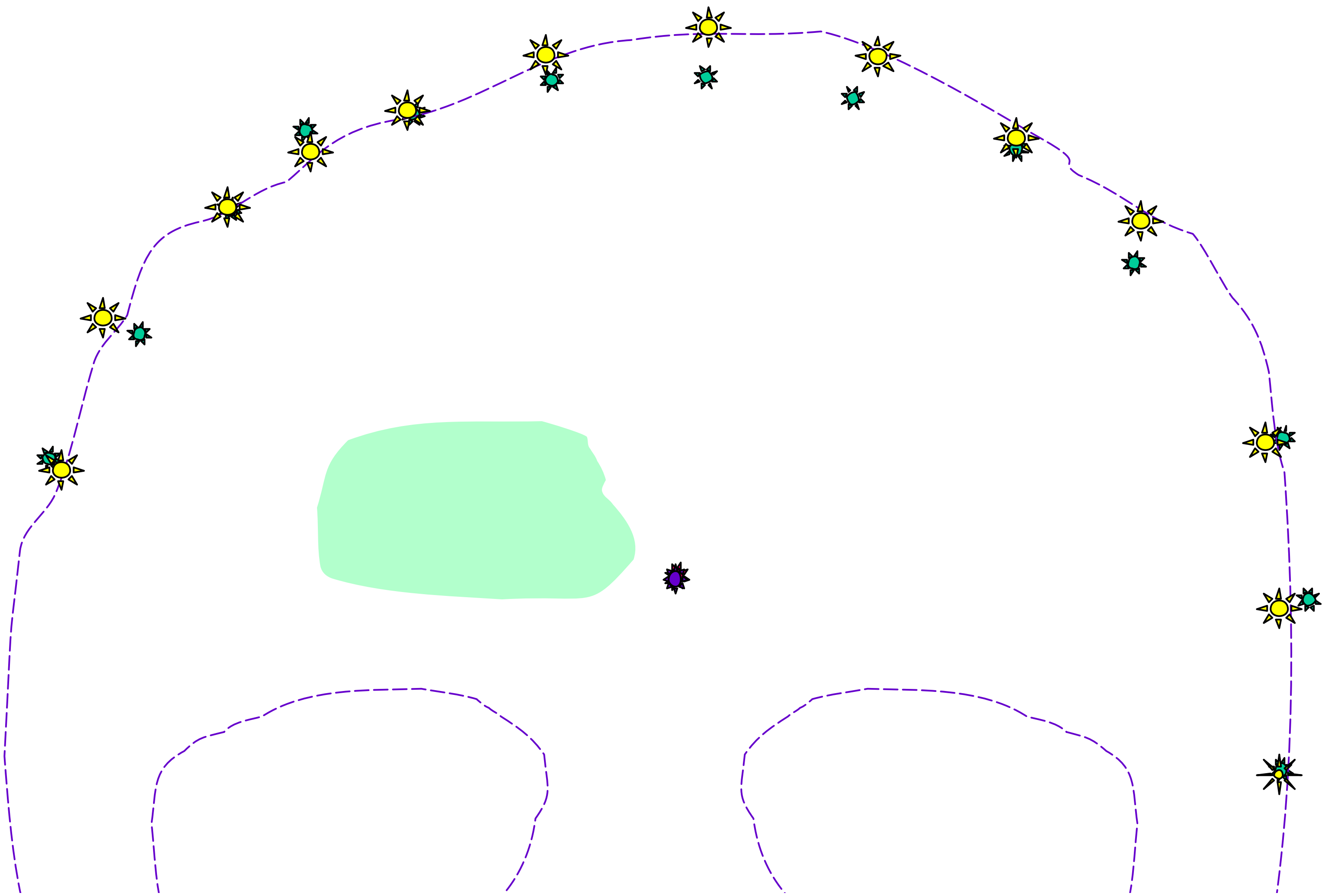
Iterative Closest Point: step 1



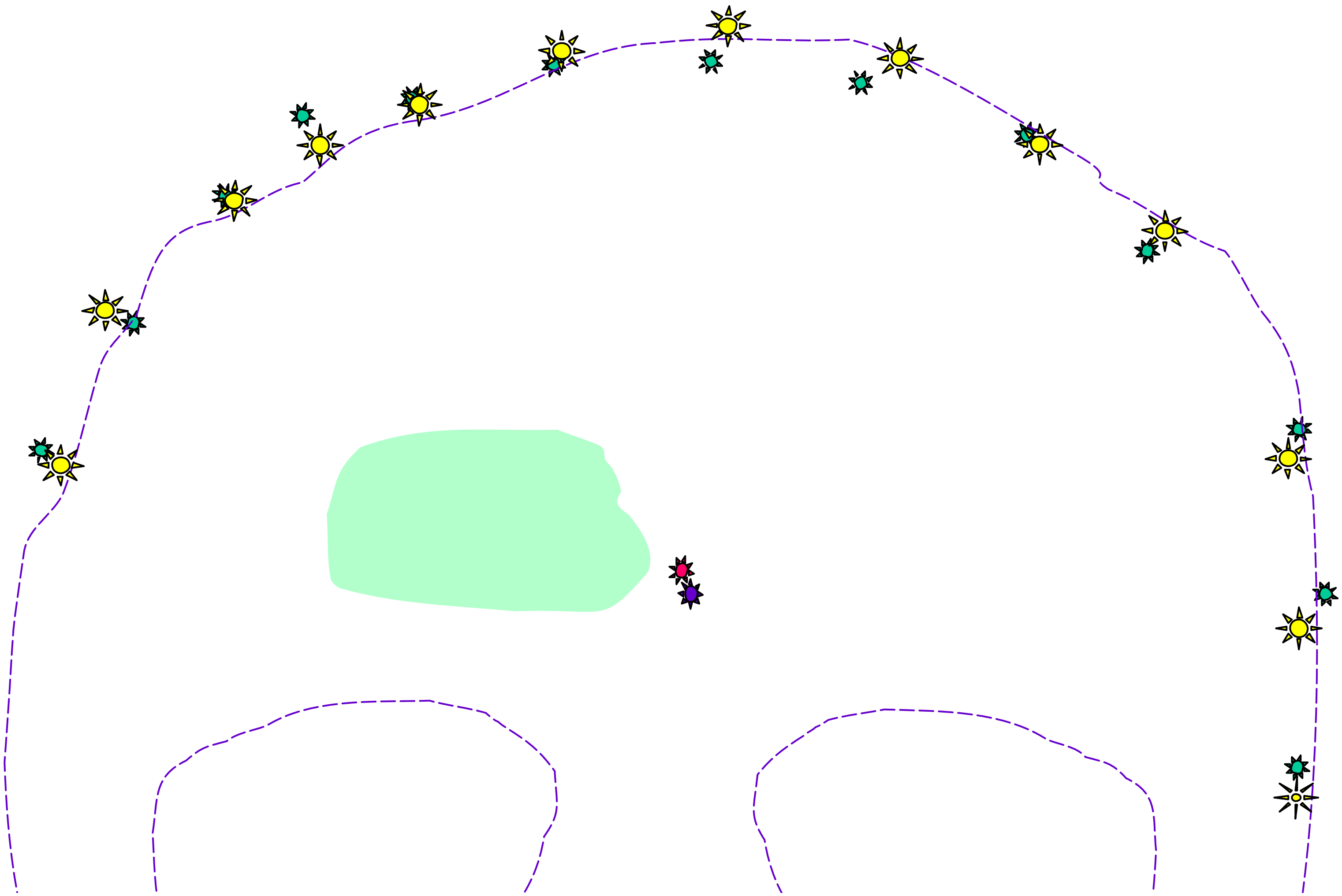
Iterative Closest Point: step 1



Iterative Closest Point: step 2



Iterative Closest Point: step 3



Iterative Closest Point: Discussion

- Minimization step can be fast
- Crucially requires fast finding of nearest points
- Local minima still an issue
- Data overlap still an issue

Distance Maps

- Many authors, e.g., Lavallee, Brunie, Malandain, Mangin
- Basic idea is to use different distance metric:

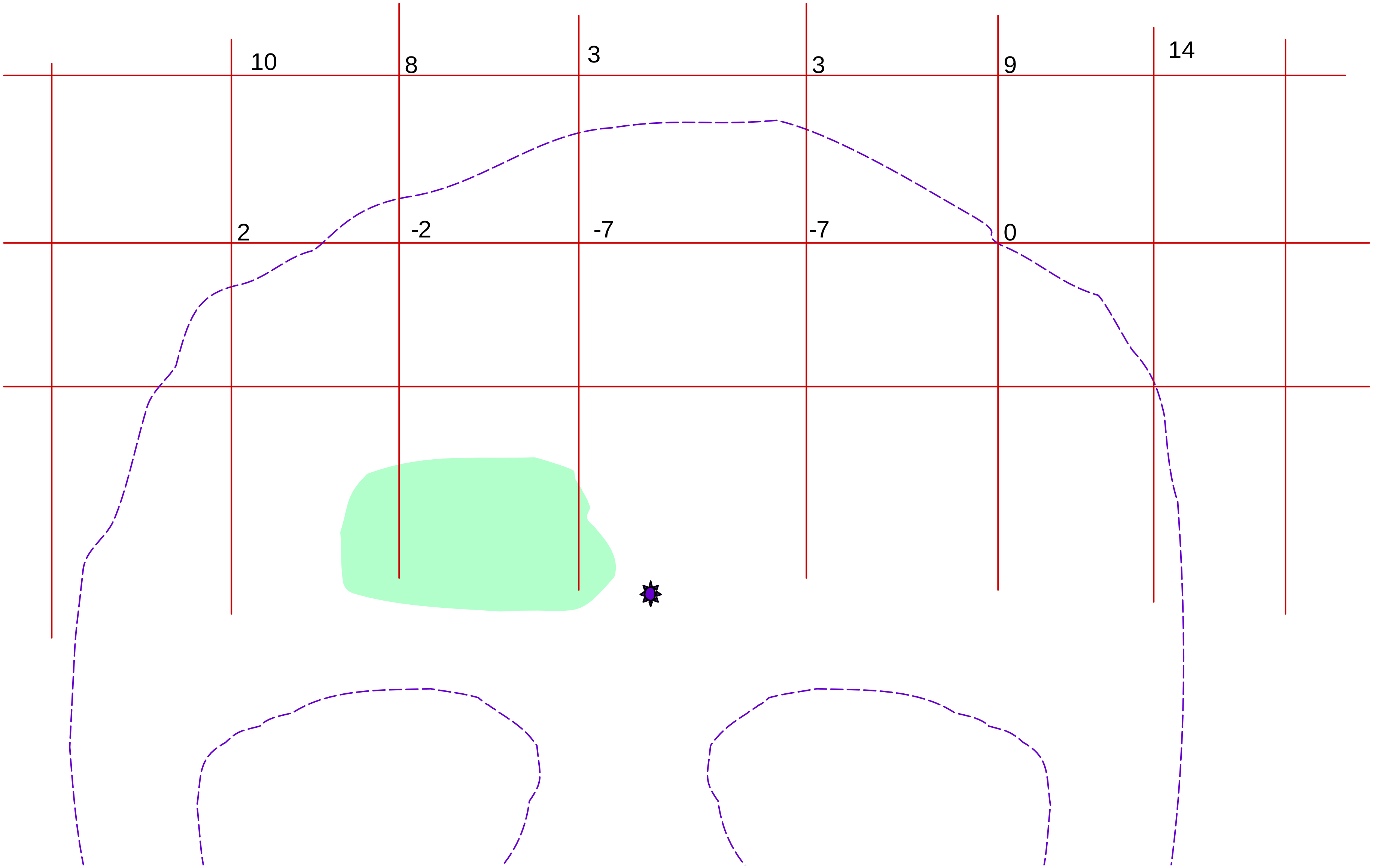
$$d_S(\mathcal{F}, \mathbf{f}) = \min_{\mathbf{p} \in \mathcal{F}} \|\mathbf{p} - \mathbf{f}\|$$

- But the problem is how to compute this quickly

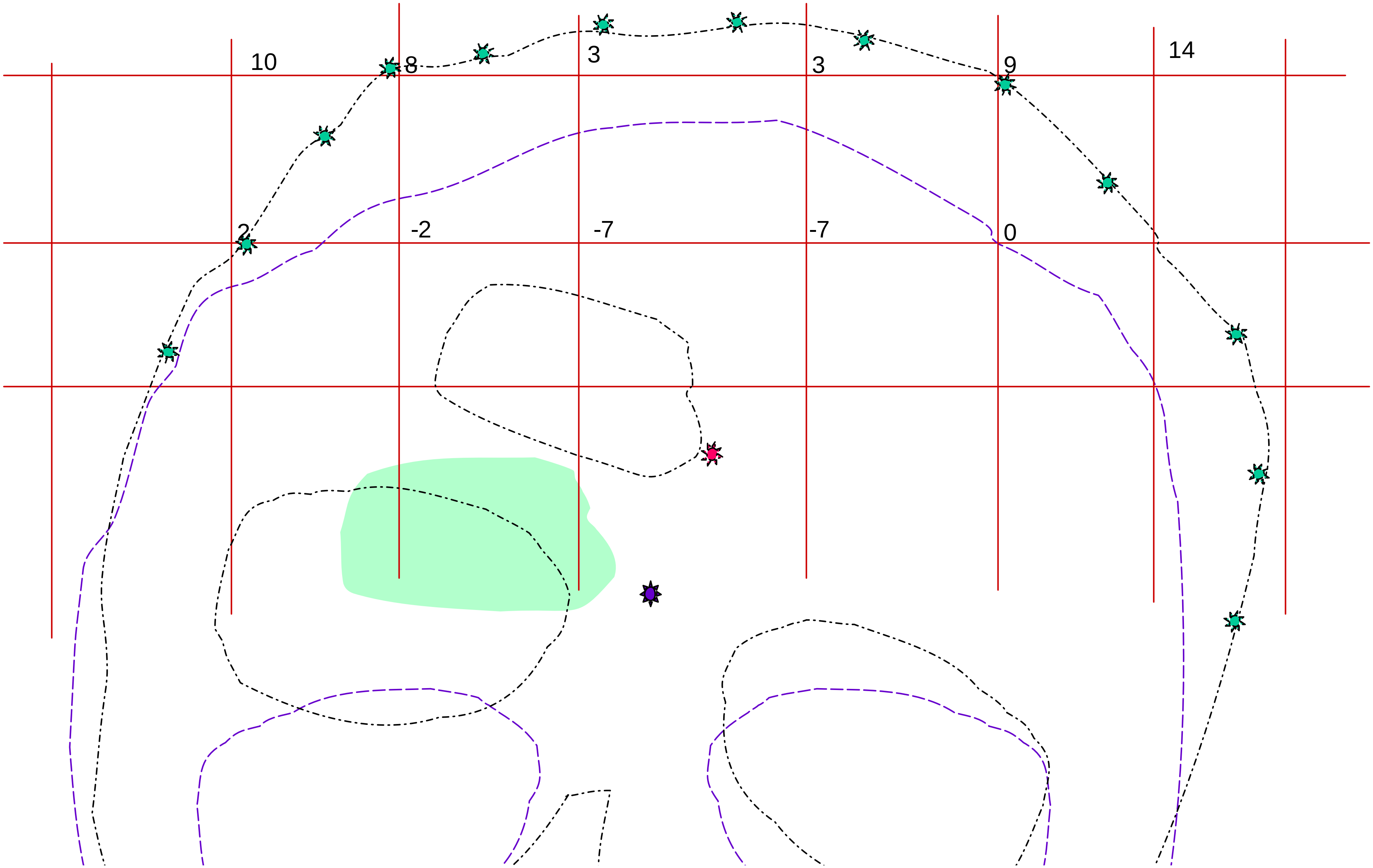
Distance Maps (Continued)

- Approach is to **precompute** $d_S(\mathcal{F}, \mathbf{v}_j)$ for a lattice of points $\mathbf{v}_j$.
- Then, to compute $d_S(\mathcal{F}, \mathbf{f}_i)$:
  1. Determine the set $\mathcal{V}$ of lattice points surrounding $\mathbf{f}_i$.
  2. Look up the distances $\{d_j = d_S(\mathcal{F}, \mathbf{v}_j)\}$ for $\mathbf{v}_j \in \mathcal{V}$.
  3. Estimate d_S from the d_j, e.g., by trilinear interpolation
- Various techniques to do the optimization

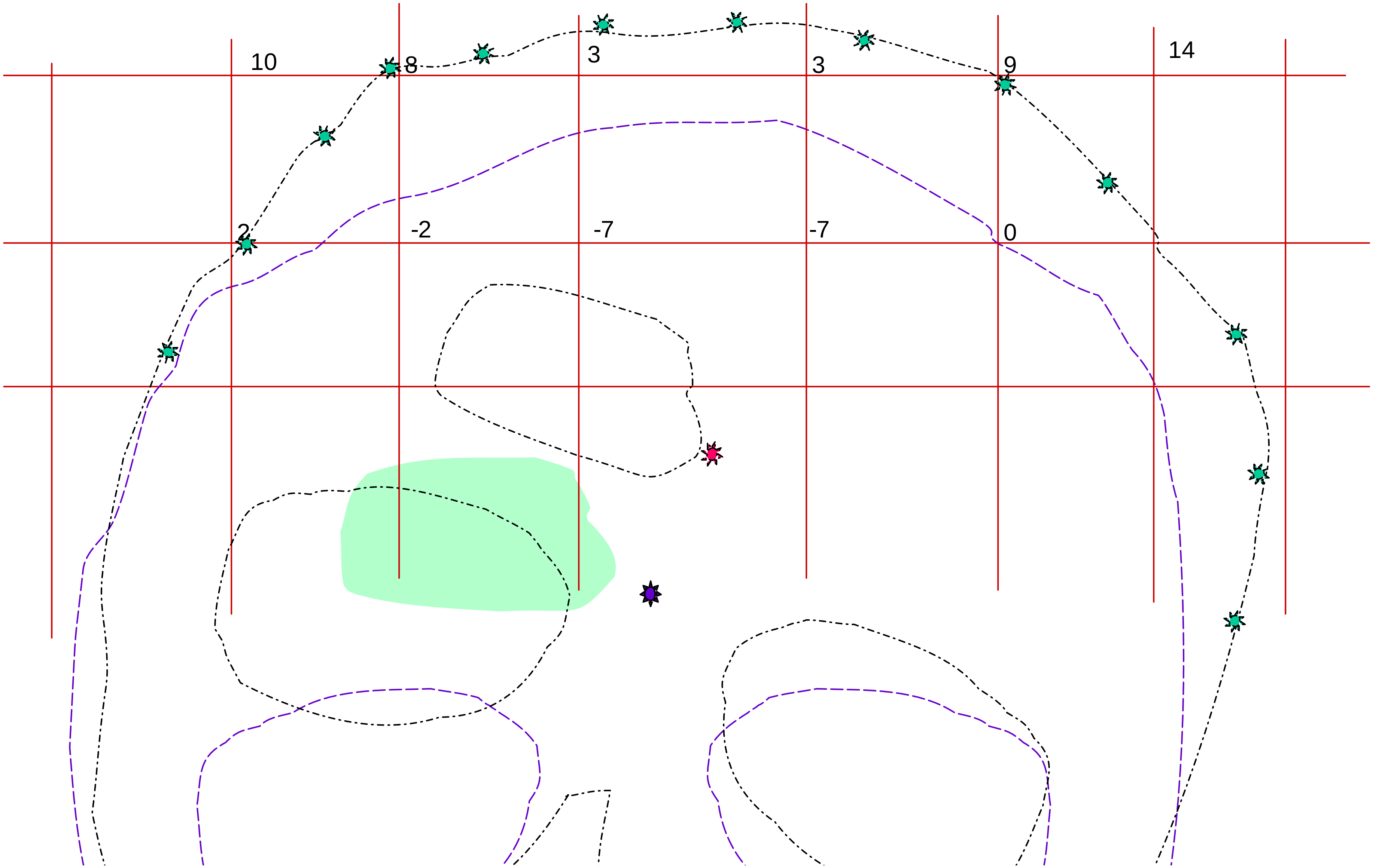
Distance Maps: step 0



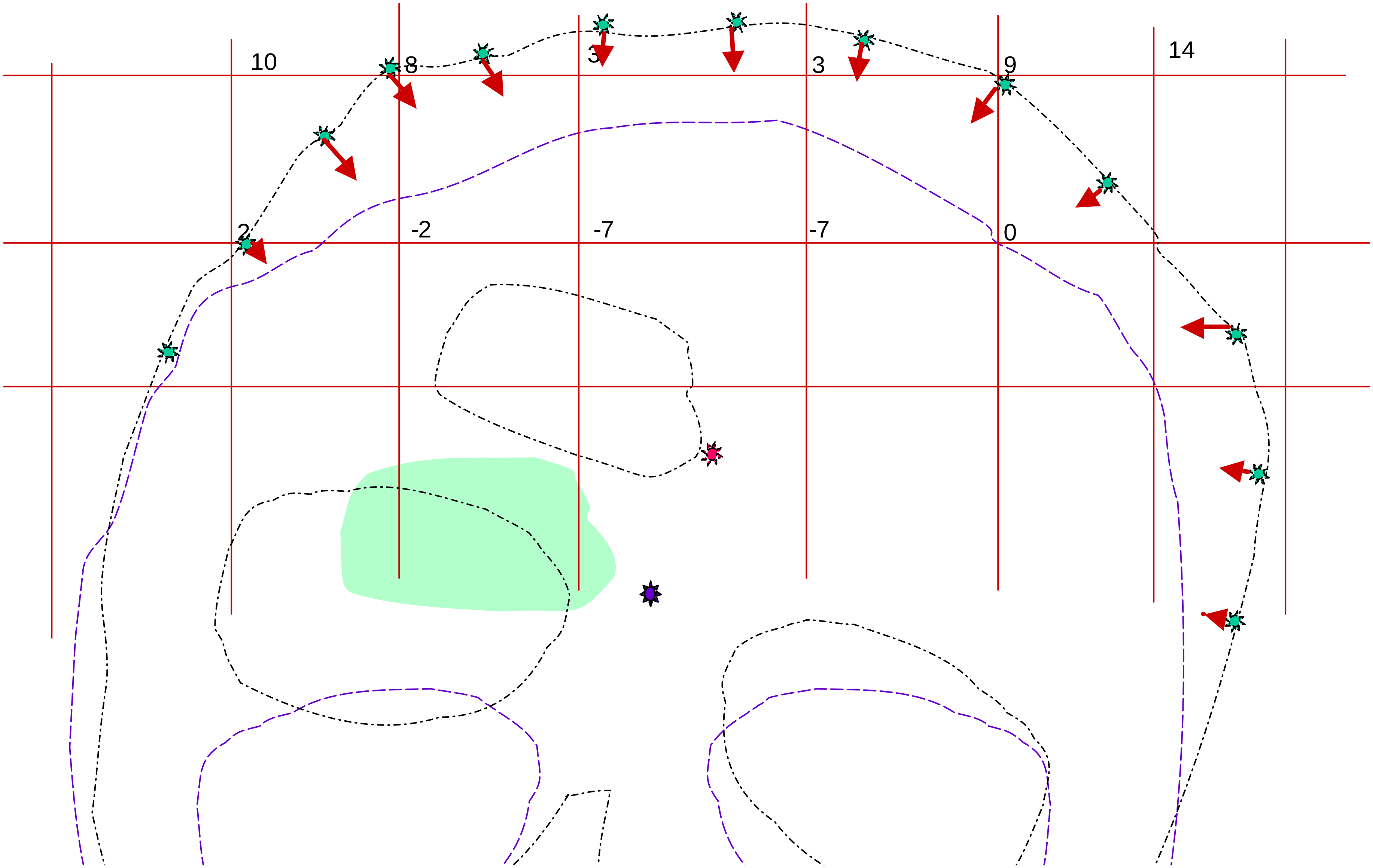
Distance Maps: step 1



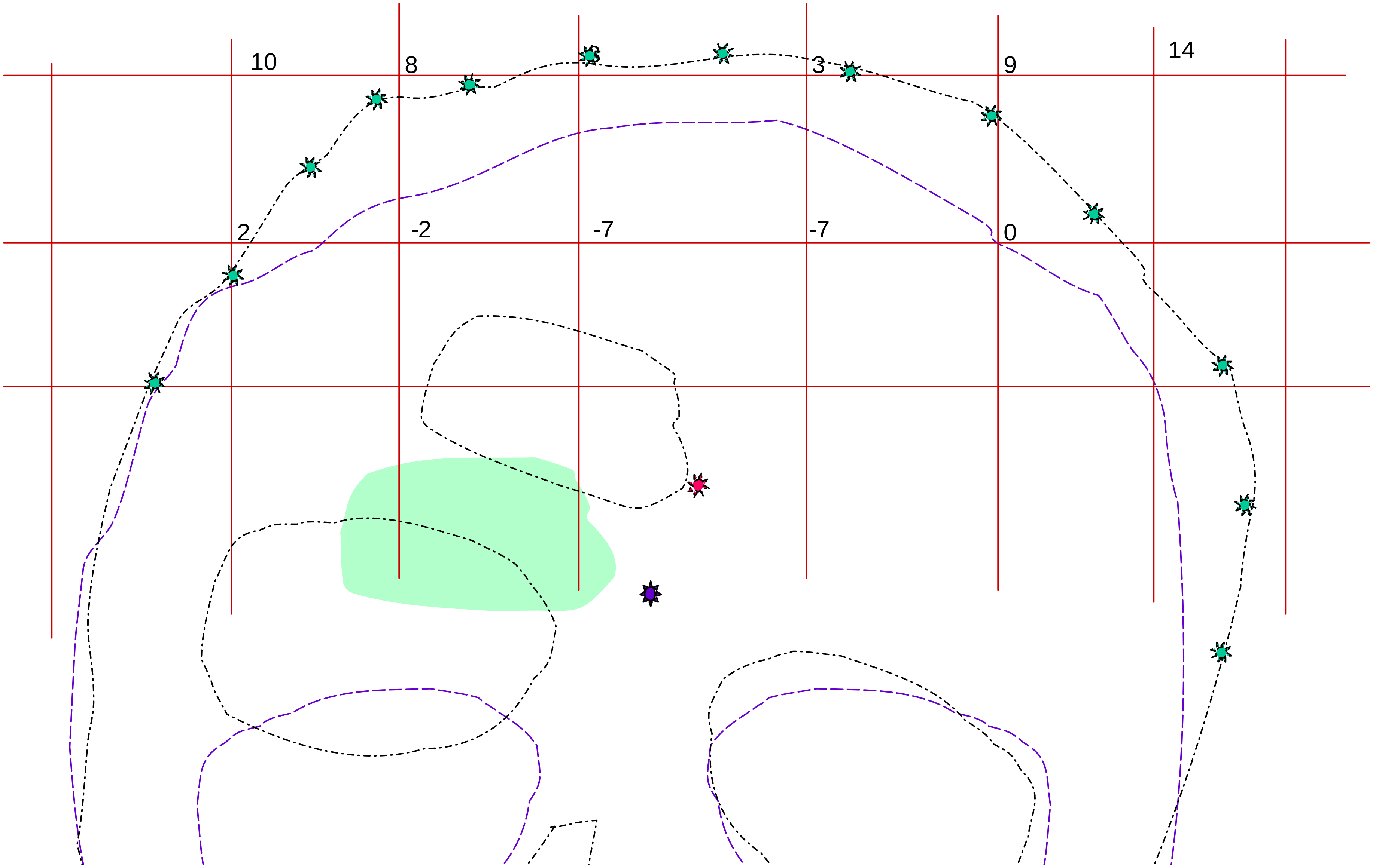
Distance Maps: step 1



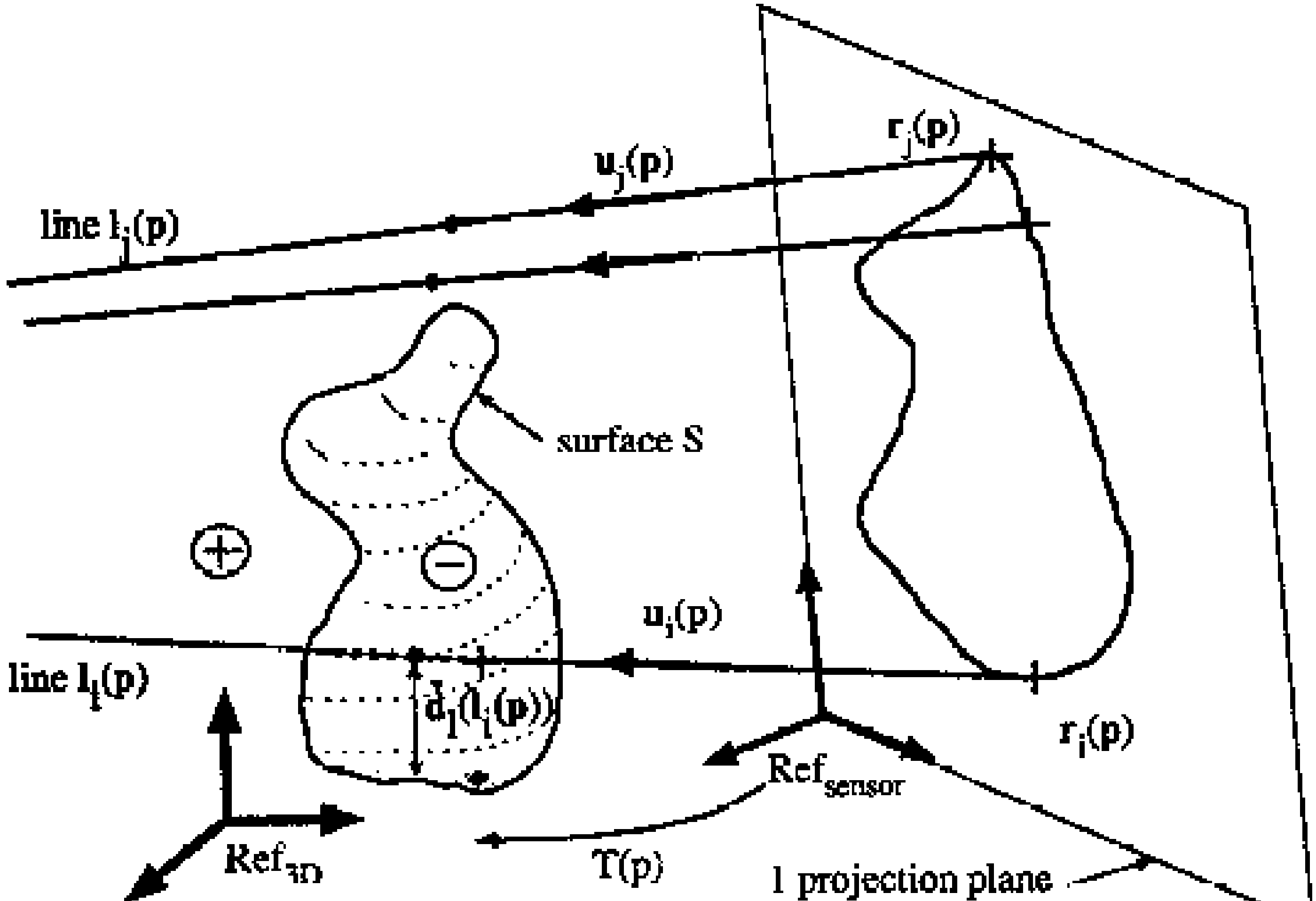
Distance Maps: step 2



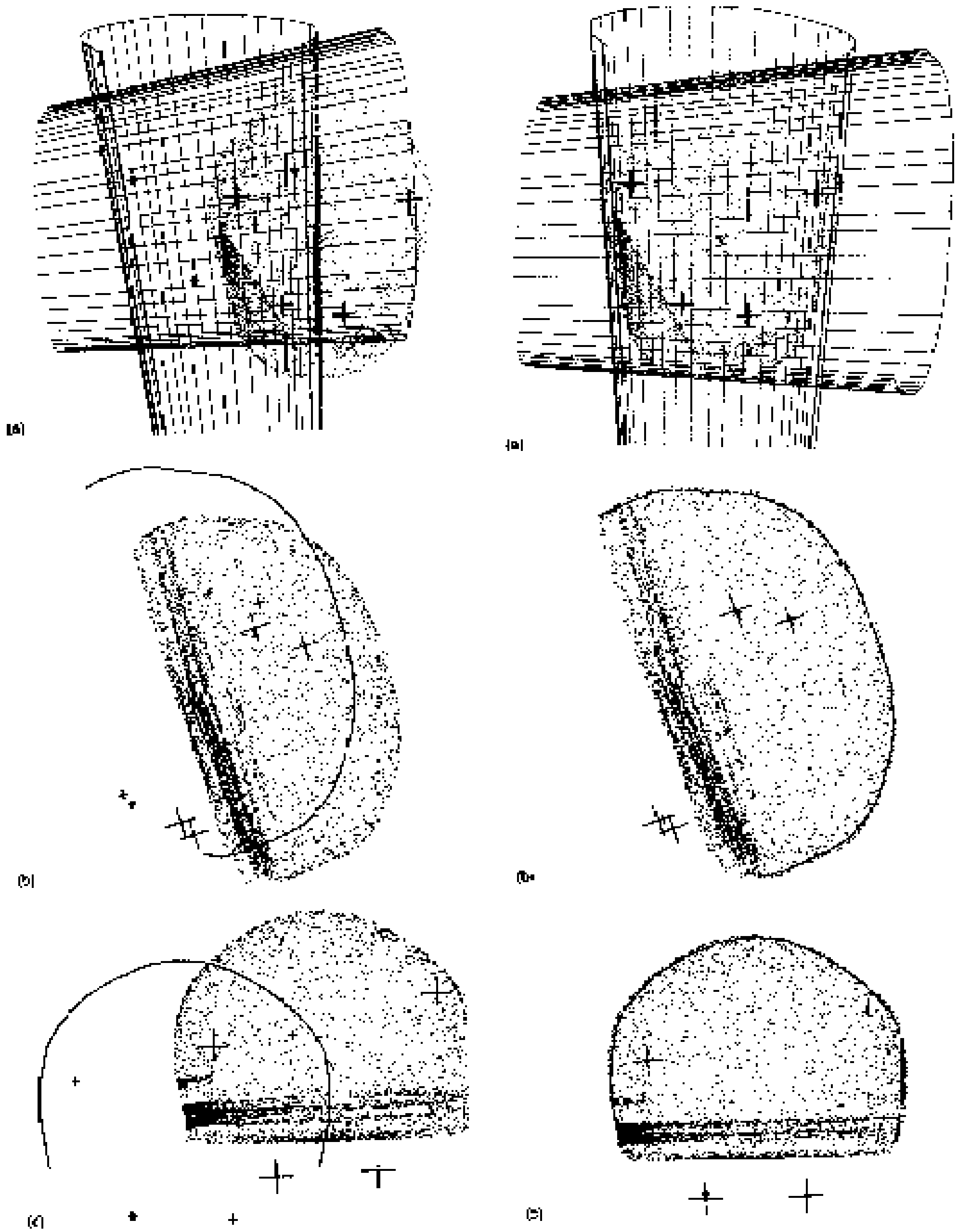
Distance Maps: step 3



2D-to-3D



Source: Lavallee, CIS book



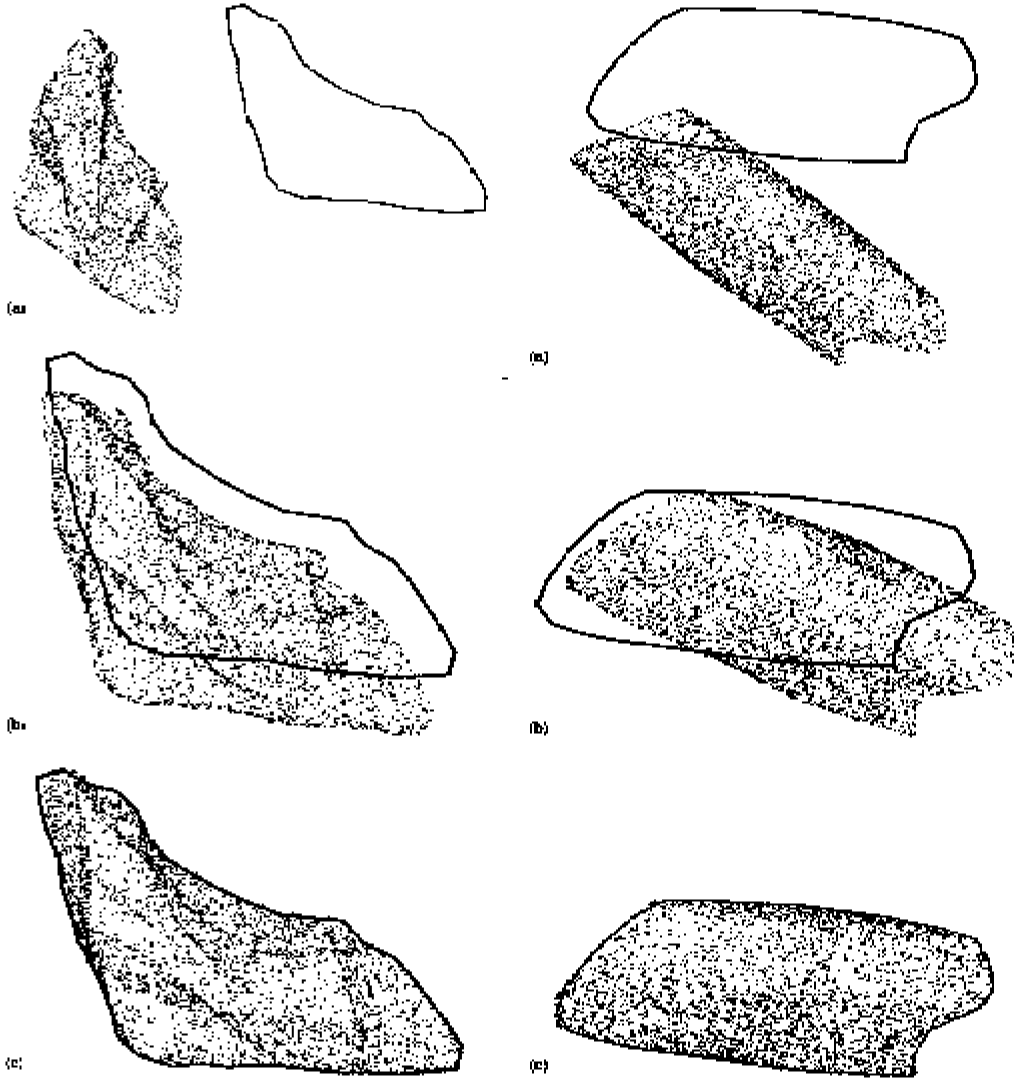


FIGURE 7.9 Convergence of algorithm for surface S_1 observed from the two projection viewpoints. The external contours of the projected surface end up fitting the real contours. (a) Initial configuration. (b) After two iterations. (c) After six iterations.