Registration

600.445 Computer-Integrated Surgery

Russell H. Taylor

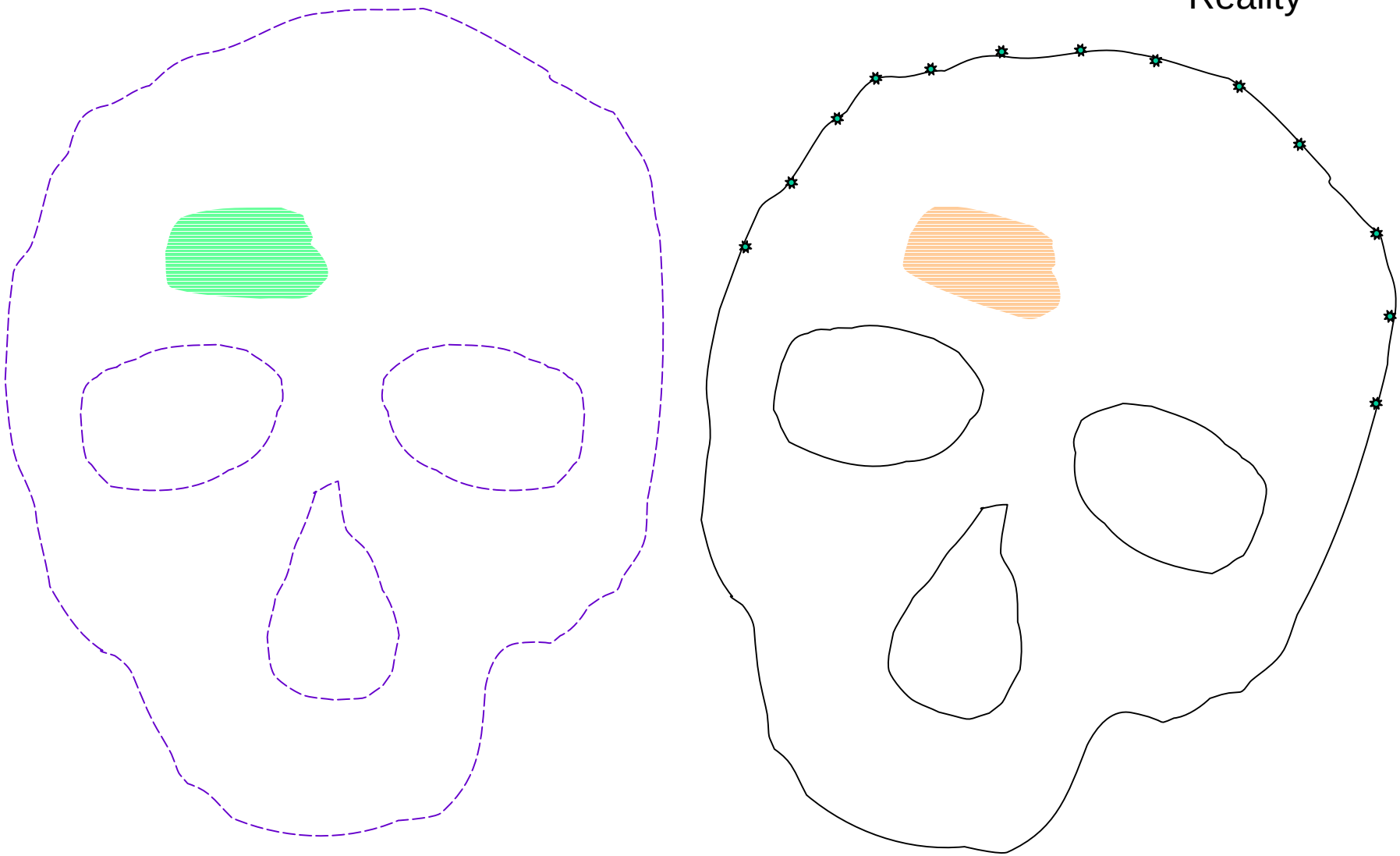
What needs registering?

- **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

Preoperative
Model

Intraoperative
Reality



Framework

- Definition of coordinate system relations
- Segmentation of reference features
- Definition of disparity function between features
- Optimization of disparity function

Definitions

Overall Goal: Given two coordinate systems,

$$\mathbf{Ref}_A \ \& \ \mathbf{Ref}_B$$

and coordinates

$$\mathbf{x}_A \ \& \ \mathbf{x}_B$$

associated with homologous 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:

$$\mathbf{x}_A = \mathbf{T}(\mathbf{x}_B)$$

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

Distance Functions

Given two (possibly distributed) features *Fi* and *Fj*, need to define a distance metric distance (Fi, Fj) between them. Some choices include:

- Minimum distance between points
- Maximum of minimum distances
- Area between line features
- Volume between surface features
- Area between point and line
- etc.

Disparity Functions Between Feature Sets

Let $\mathcal{F}_A = \{\ldots F_{Ai} \ldots\}$ and $\mathcal{F}_B = \{\ldots F_{Bi} \ldots\}$ be corresponding sets of features in $\mathbf{Ref}_A$and $\mathbf{Ref}_B$, respectively. We need to define an appropriate disparity function $D(\mathcal{F}_A, \mathcal{F}_B)$ between feature sets. Some typical choices include:

$$D = \sum_i w_i [distance(F_{Ai}, \mathbf{T}(F_{Bi}))]^2$$

$$D = \max_i distance(F_{Ai}, \mathbf{T}(F_{Bi}))$$

$$D = \underset{i}{\text{median}}\ distance(F_{Ai}, \mathbf{T}(F_{Bi}))$$

$$D = Cardinality\{i | distance(F_{Ai}, \mathbf{T}(F_{Bi})) > threshold\}$$

Optimization

- Global *vs* Local
- Numerical *vs* Direct Solution
- Local Minima

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.

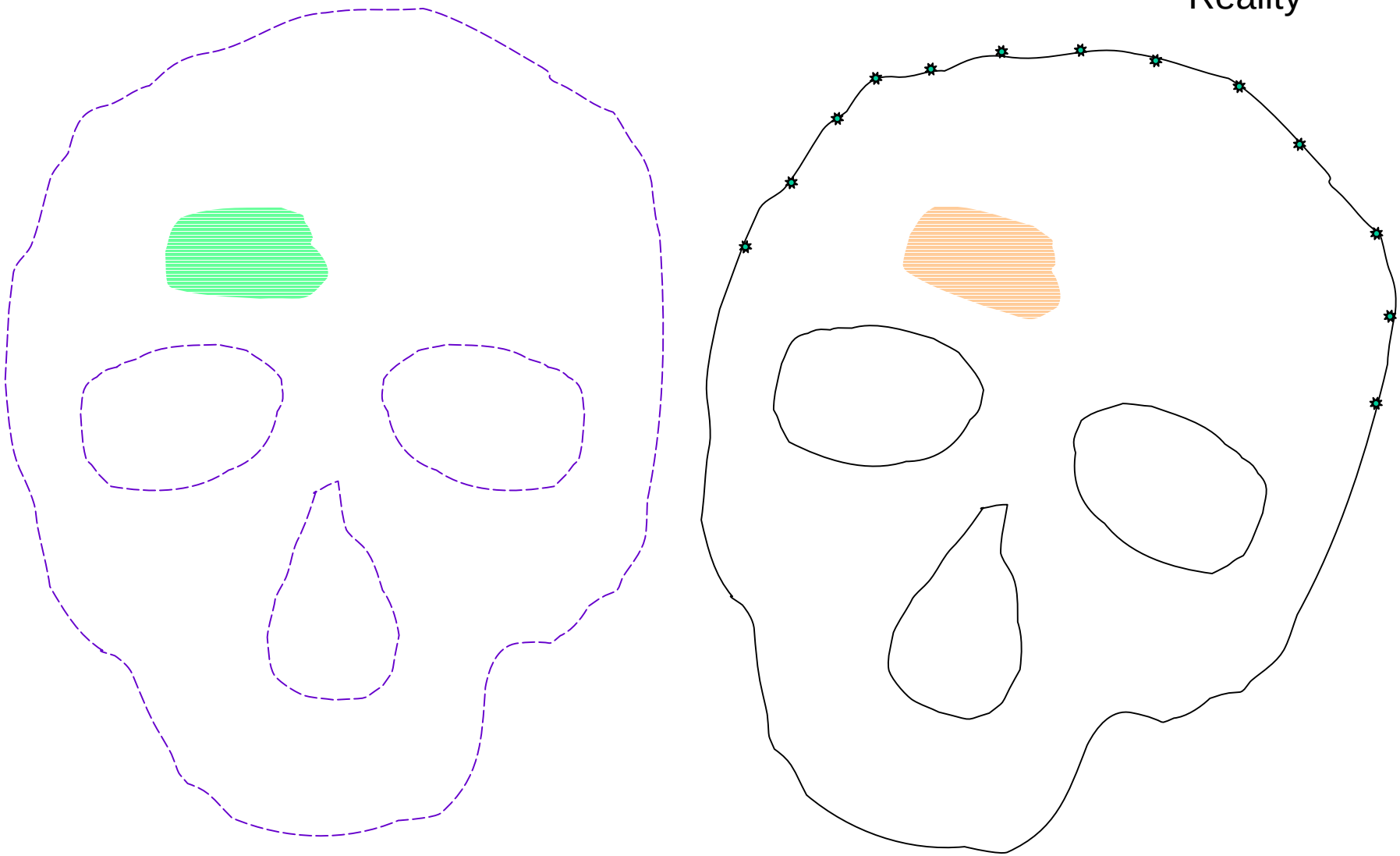
Examples

- 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]

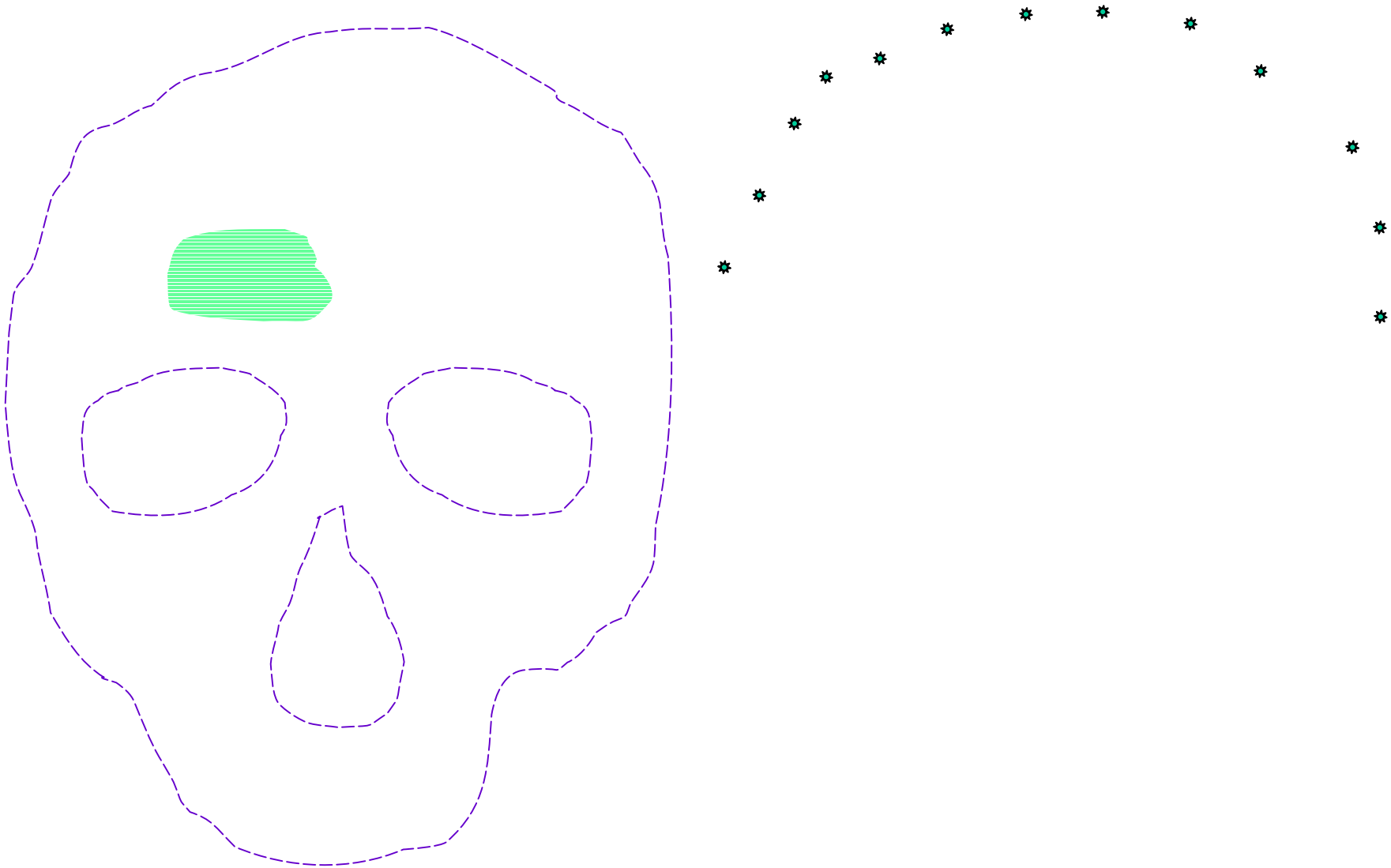
A typical registration problem

Preoperative
Model

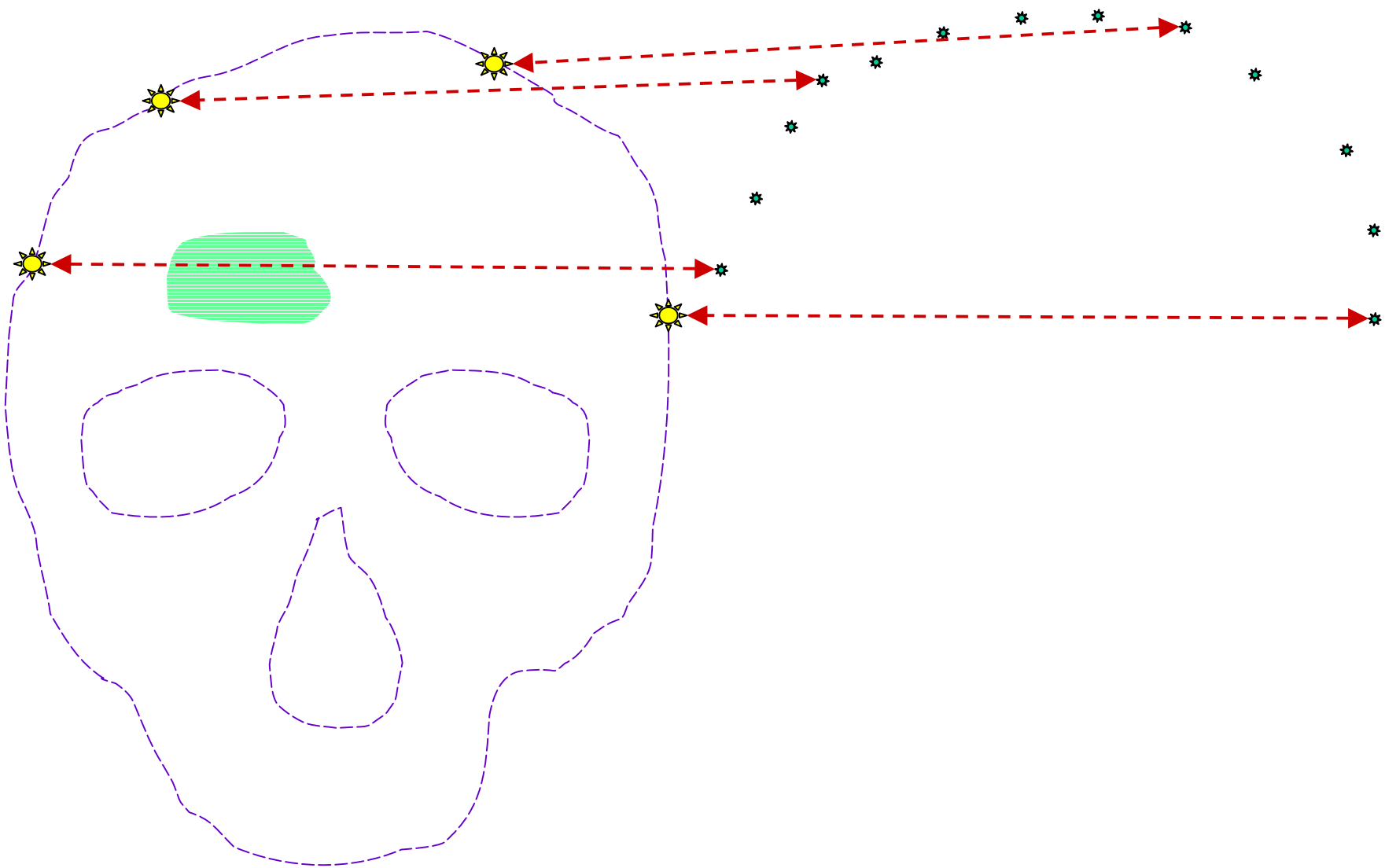
Intraoperative
Reality



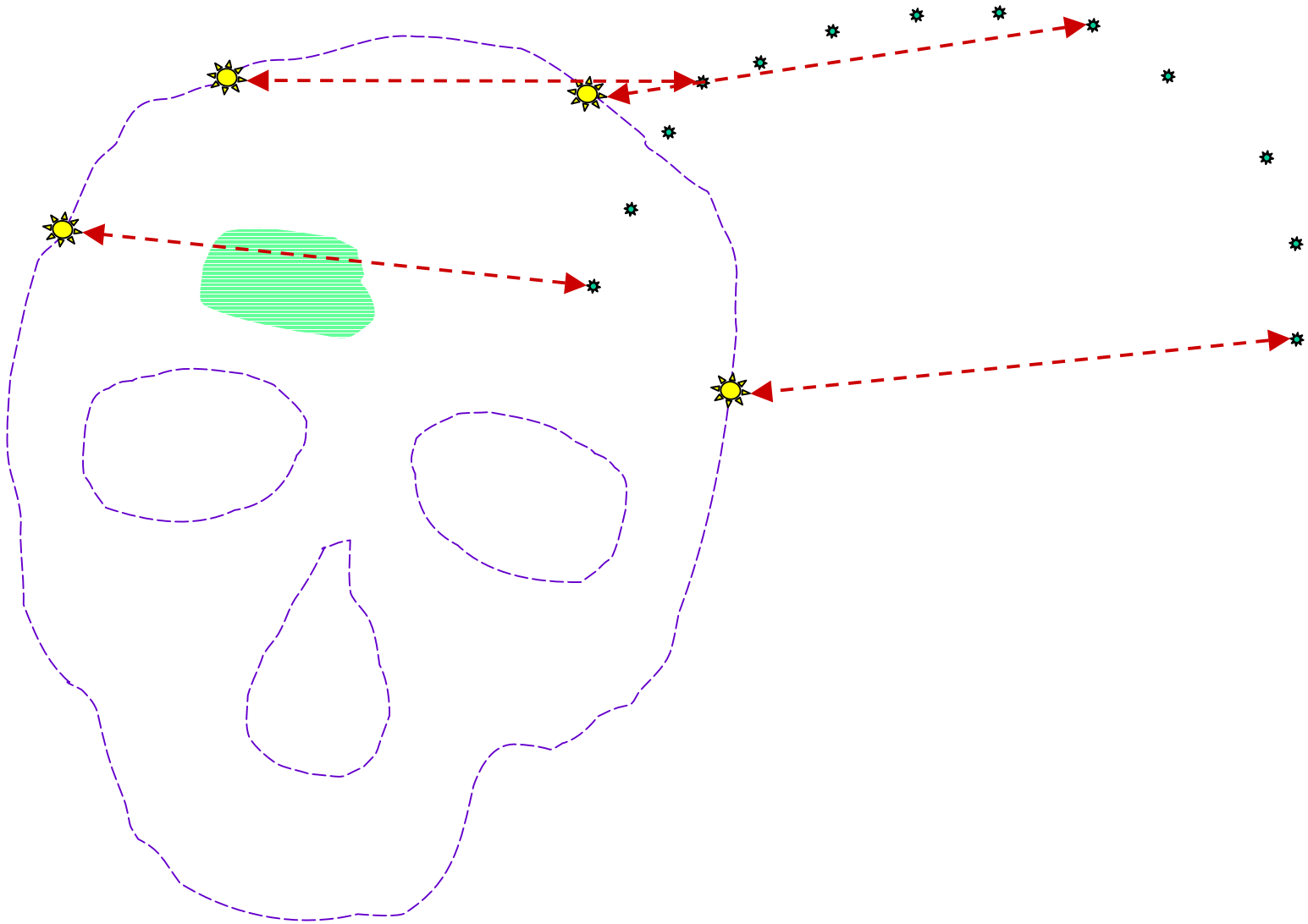
What the computer knows



Find homologous points & pull!



Find homologous points & pull!

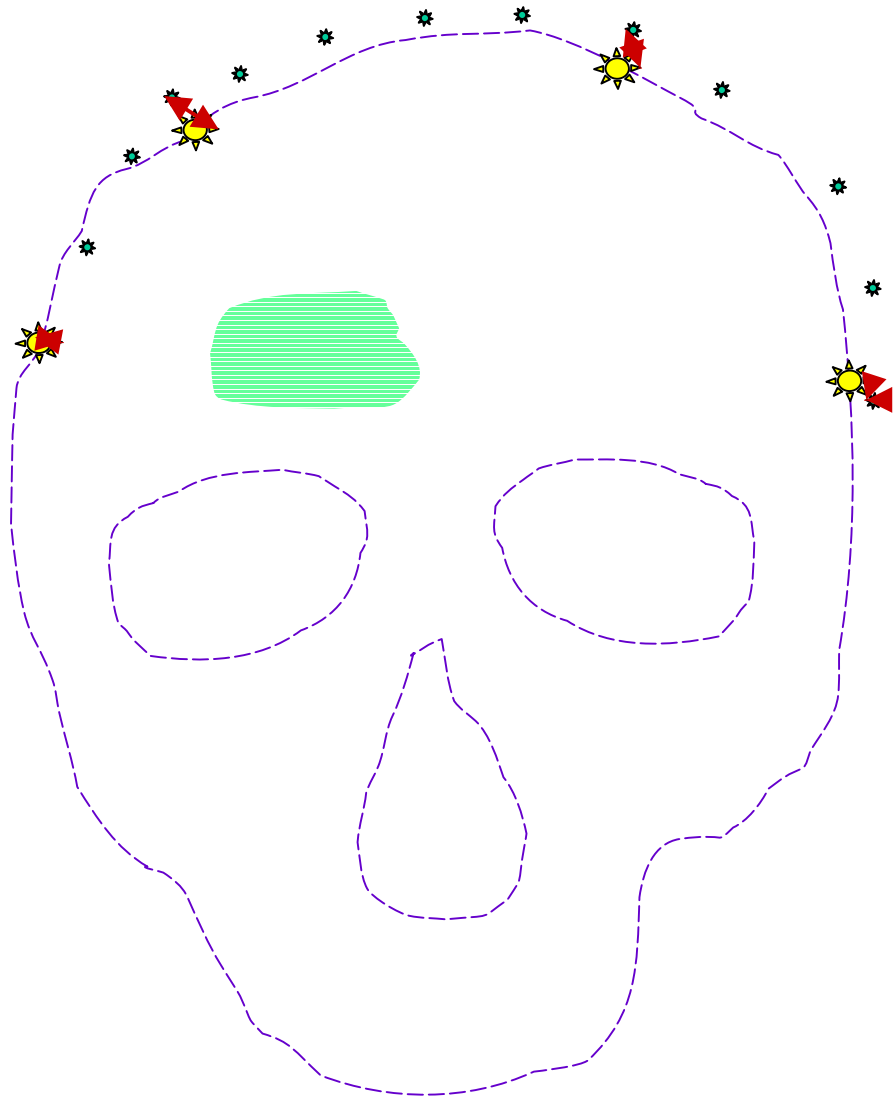


Find homologous points & pull!

Iterate this until converge

Find new point pairs every iteration

Key challenge is finding point pairs efficiently.



Head in Hat Algortithm

- 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 $\mathbf{T}$.

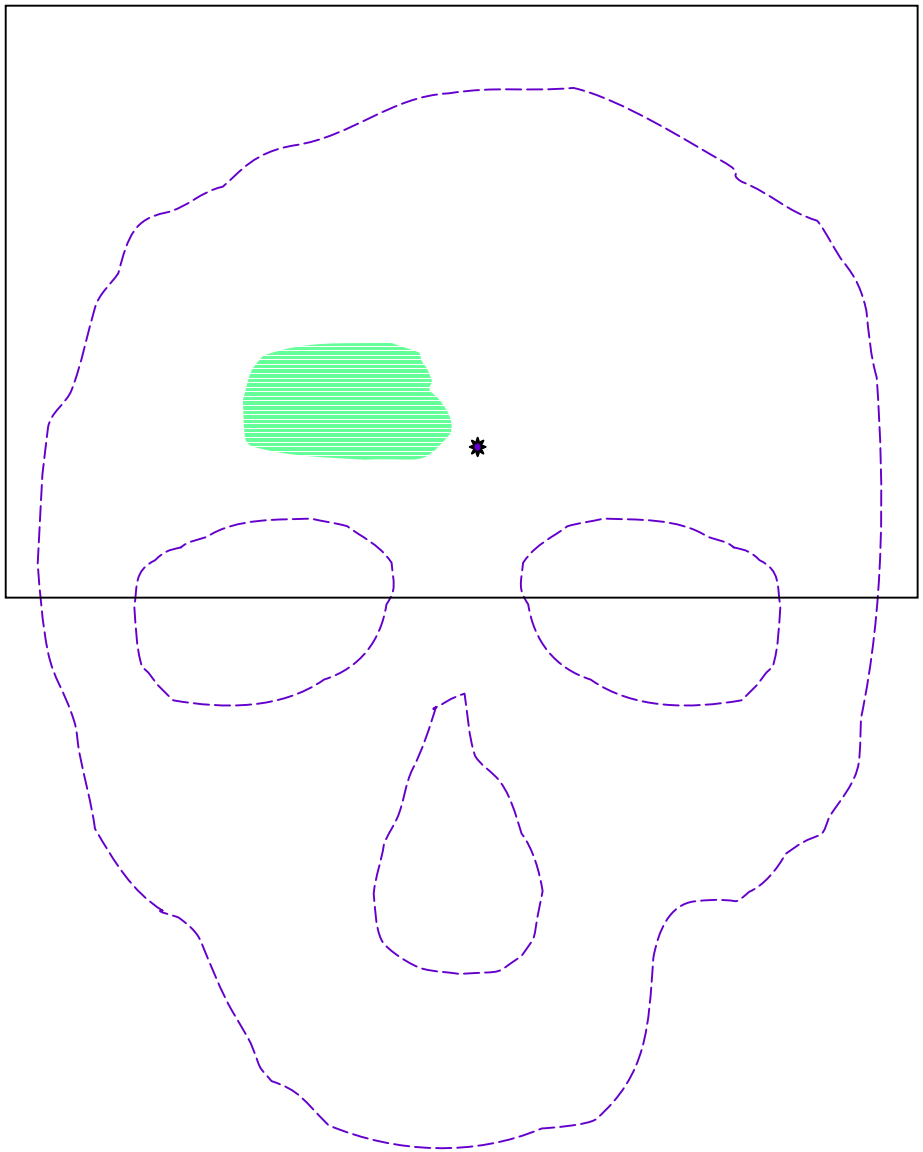
- Optimization uses standard numerical method (steepest gradient descent [Powell]) to find six parameters (3 rotations, 3 translations) defining $\mathbf{T}$.

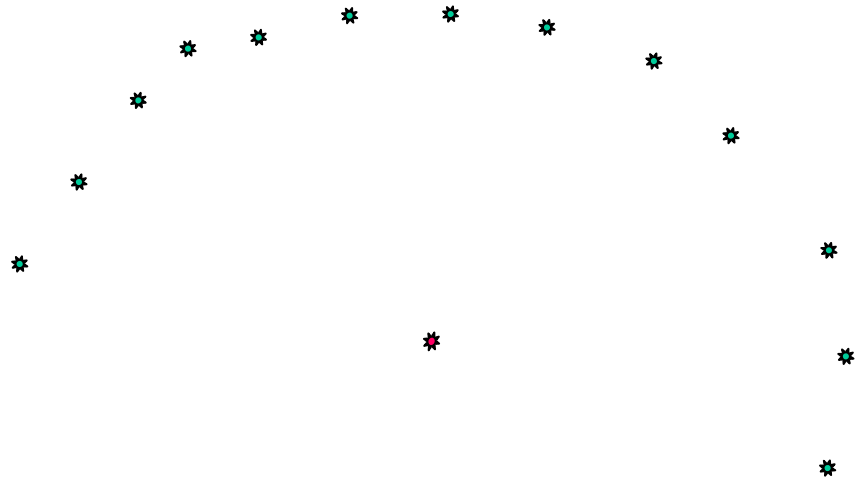
Head in Hat Algorithm

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

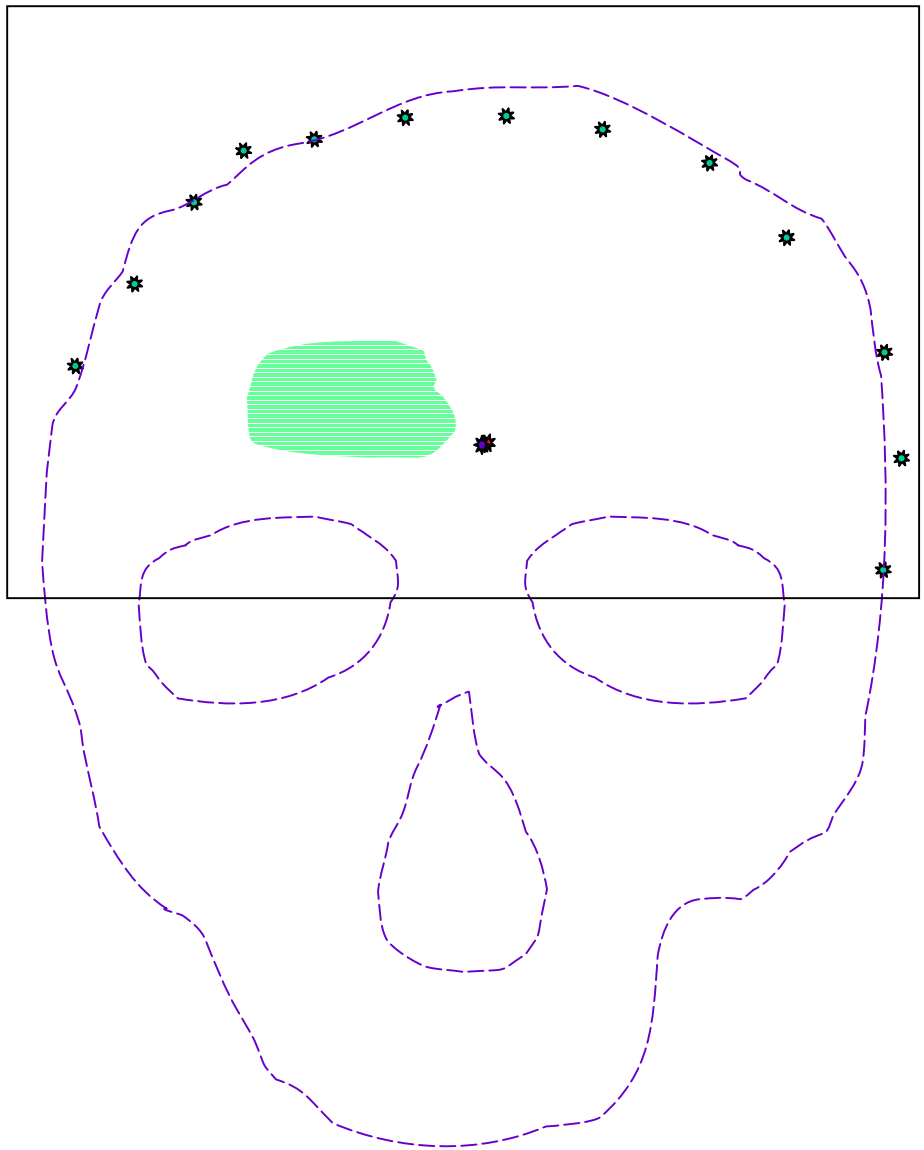
1. Compute centroid $\mathbf{g}_B$ of surface $\mathcal{F}_B$.
2. Determine a point $\mathbf{q}_i$ that lies on the intersection of the line $\mathbf{g}_B - \mathbf{f}_i$ and $\mathcal{F}_B$.
3. Then, $d_S(\mathcal{F}_B, \mathbf{f}_i) = \|\mathbf{q}_i - \mathbf{f}_i\|$

Head-in-hat algorithm: step 0

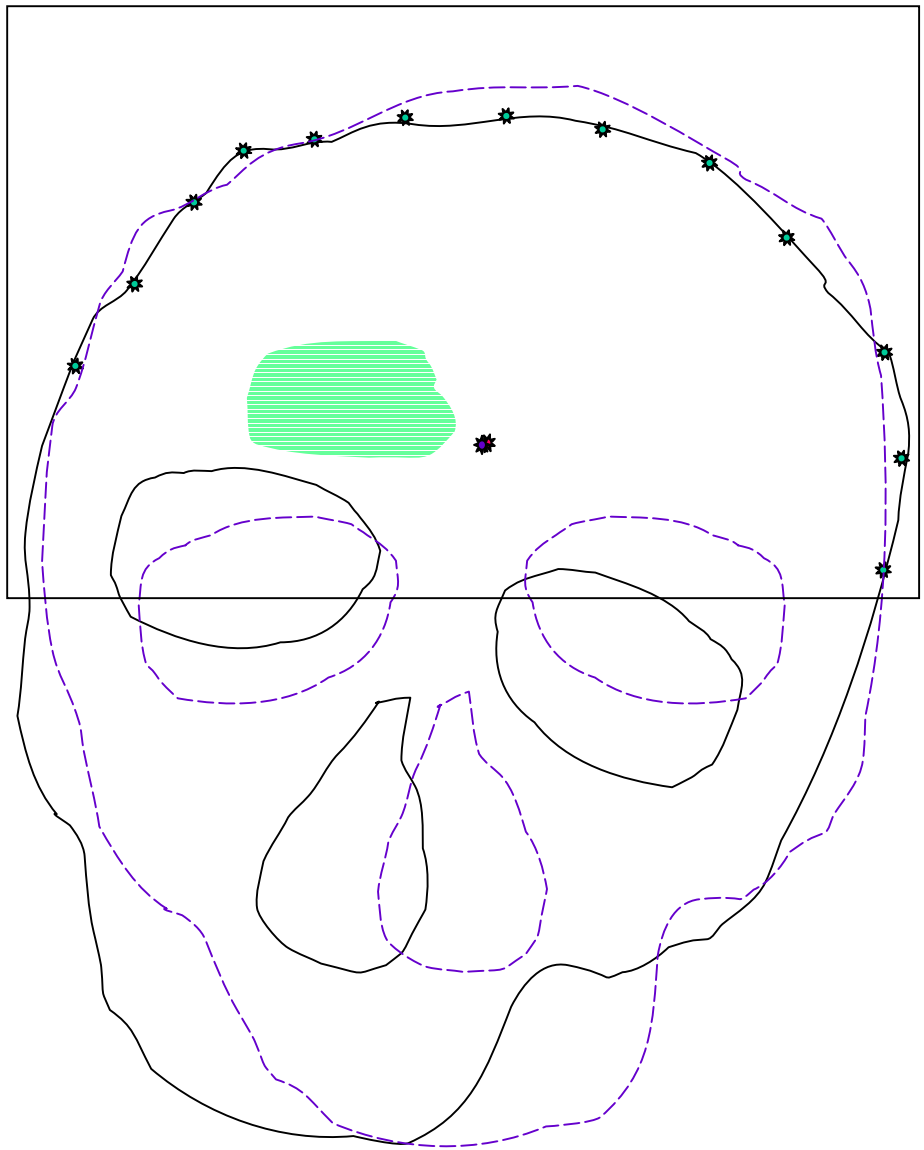




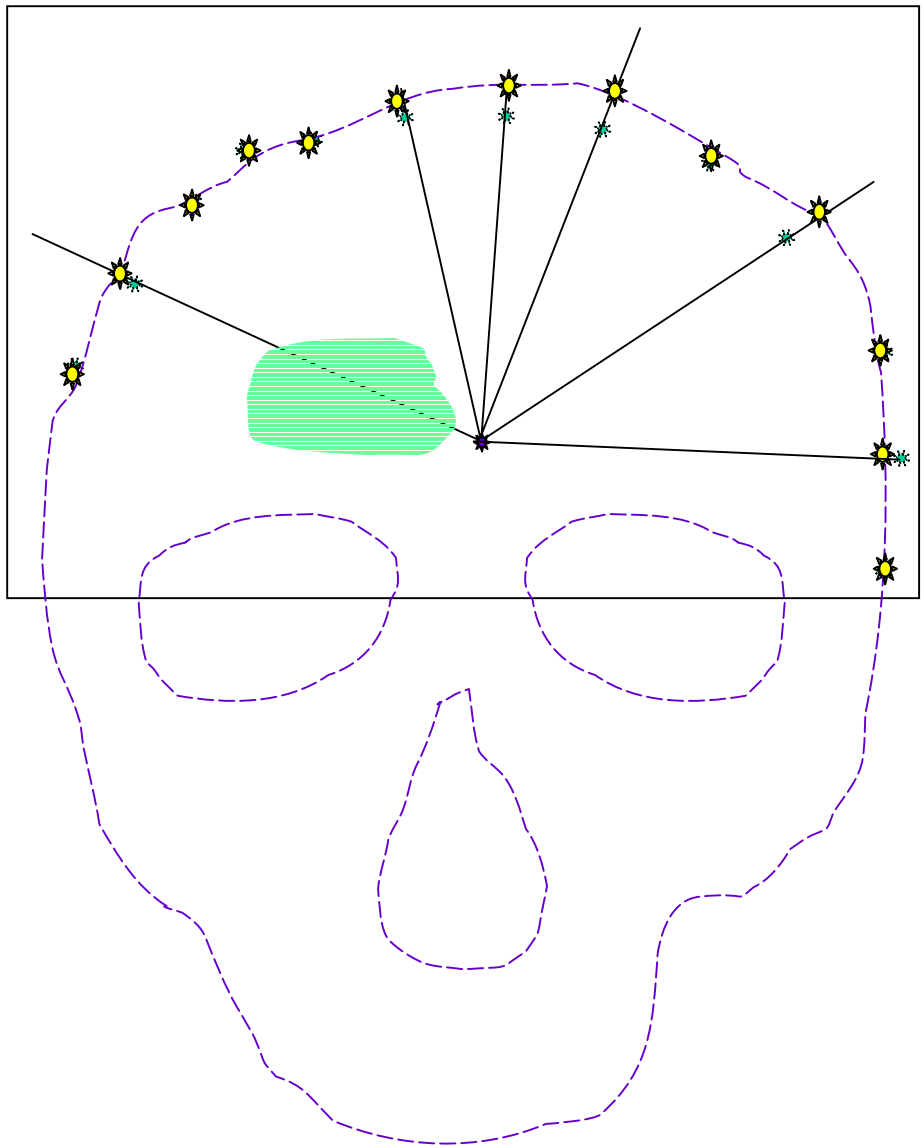
Head-in-hat algorithm: step1



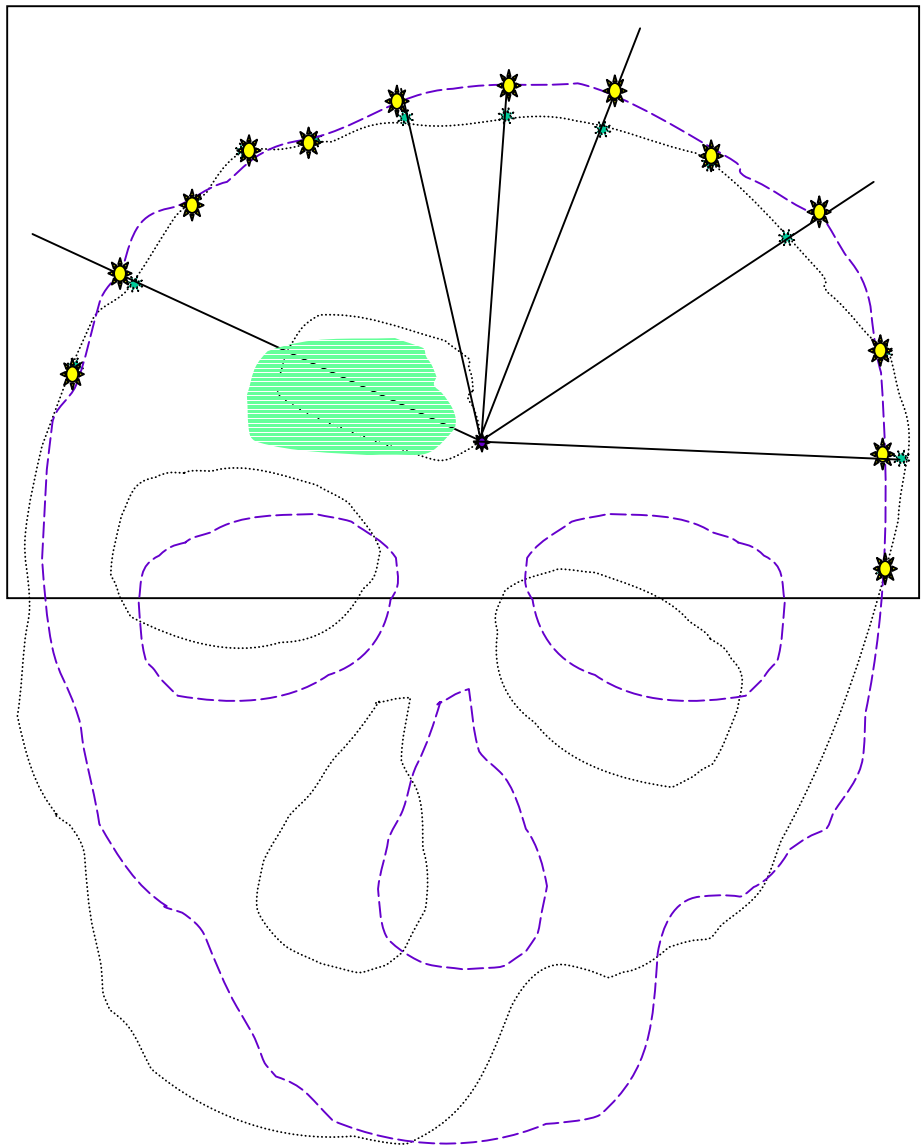
Head-in-hat algorithm: step1



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

Minimizing Rigid Registration Errors

Typically, given a set of points $\{\mathbf{a}_i\}$ in one coordinate system and another set of points $\{\mathbf{b}_i\}$ in a second coordinate system

Goal is to find $[\mathbf{R},\mathbf{p}]$ that minimizes

$$\eta = \sum_i \mathbf{e}_i \bullet \mathbf{e}_i$$

where

$$\mathbf{e}_i = (\mathbf{R} \bullet \mathbf{a}_i + \mathbf{p}) - \mathbf{b}_i$$

This is tricky, because of $\mathbf{R}$.

Minimizing Rigid Registration Errors

Step 1: Compute

$$\bar{\mathbf{a}} = \frac{1}{N}\sum_{i=1}^{N}\vec{\mathbf{a}}_i \qquad \bar{\mathbf{b}} = \frac{1}{N}\sum_{i=1}^{N}\vec{\mathbf{b}}_i$$

$$\tilde{\mathbf{a}}_i = \vec{\mathbf{a}}_i - \bar{\mathbf{a}} \qquad \tilde{\mathbf{b}}_i = \vec{\mathbf{b}}_i - \bar{\mathbf{b}}$$

Step 2: Find $\mathbf{R}$ that minimizes

$$\sum_i (\mathbf{R}\cdot\tilde{\mathbf{a}}_i - \tilde{\mathbf{b}}_i)^2$$

Step 3: Find $\vec{\mathbf{p}}$

$$\vec{\mathbf{p}} = \bar{\mathbf{b}} - \mathbf{R}\cdot\bar{\mathbf{a}}$$

Step 4: Desired transformation is

$$\mathbf{F} = Frame(\mathbf{R}, \vec{\mathbf{p}})$$

Solving for R: iteration method

Given $\left\{\cdots,\left(\tilde{\mathbf{a}}_i,\tilde{\mathbf{b}}_i\right),\cdots\right\}$, want to find $\mathbf{R}=\arg\min\sum_i\left(\mathbf{R}\tilde{\mathbf{a}}_i-\tilde{\mathbf{b}}_i\right)$

Step 0: Make an initial guess $\mathbf{R}_0$

Step 1: Given $\mathbf{R}_k$, compute $\breve{\mathbf{b}}_i=\mathbf{R}_k{}^{-1}\tilde{\mathbf{b}}_i$

Step 2: Compute $\Delta\mathbf{R}$ that minimizes

$$\sum_i(\Delta\mathbf{R}\,\tilde{\mathbf{a}}_i-\breve{\mathbf{b}}_i)^2$$

Step 3: Set $\mathbf{R}_{k+1}=\mathbf{R}_k\Delta\mathbf{R}$

Step 4: Iterate Steps 1-3 until residual error is sufficiently small (or other termination condition)

Iterative method: Solving for ΔR

Approximate $\Delta\mathbf{R}$ as $(\mathbf{I} + skew(\bar{\alpha}))$. I.e.,

$$\Delta\mathbf{R} \bullet \mathbf{v} \approx \mathbf{v} + \bar{\alpha} \times \mathbf{v}$$

for any vector $\mathbf{v}$. Then, our least squares problem becomes

$$\min_{\Delta\mathbf{R}} \sum_i (\Delta\mathbf{R} \bullet \tilde{\mathbf{a}}_i - \breve{\mathbf{b}}_i)^2 \approx \min_{\bar{\alpha}} \sum_i (\tilde{\mathbf{a}}_i - \breve{\mathbf{b}}_i + \bar{\alpha} \times \tilde{\mathbf{a}}_i)^2$$

This is linear least squares problem in $\bar{\alpha}$.

Then compute $\Delta\mathbf{R}(\bar{\alpha})$.

Direct Techniques to solve for R

- Method due to K. Arun, et. al., IEEE PAMI, Vol 9, no 5, pp 698-700, Sept 1987

Step 1: Compute

$$\mathbf{H} = \sum_i \begin{bmatrix} \bar{a}_{i,x}\bar{b}_{i,x} & \bar{a}_{i,x}\bar{b}_{i,y} & \bar{a}_{i,x}\bar{b}_{i,z} \\ \bar{a}_{i,y}\bar{b}_{i,x} & \bar{a}_{i,y}\bar{b}_{i,y} & \bar{a}_{i,y}\bar{b}_{i,z} \\ \bar{a}_{i,z}\bar{b}_{i,x} & \bar{a}_{i,z}\bar{b}_{i,y} & \bar{a}_{i,z}\bar{b}_{i,z} \end{bmatrix}$$

Step 2: Compute the SVD of $\mathbf{H} = \mathbf{USV^t}$

Step 3: $\mathbf{R} = \mathbf{VU^t}$

Step 4: Verify $Det(\mathbf{R}) = 1$. If not, then algorithm may fail.

- Failure is rare, and mostly fixable. The paper has details.

Quarternion Technique to solve for R

- B.K.P. Horn, “Closed form solution of absolute orientation using unit quaternions”, *J. Opt. Soc. America*, A vol. 4, no. 4, pp 629-642, Apr. 1987.
- Method described as reported in Besl and McKay, “A method for registration of 3D shapes”, *IEEE Trans. on Pattern Analysis and Machine* Intelligence, vol. 14, no. 2, February 1992.
- Solves a 4x4 eigenvalue problem to find a unit quaternion corresponding to the rotation
- This quaternion may be converted in closed form to get a more conventional rotation matrix

Digression: quaternions

Invented by Hamilton as a way to express the ratio of vectors. Can be thought of as

4 elements: $\mathbf{q} = [q_0, q_1, q_2, q_3]$

scalar & vector: $\mathbf{q} = s + \vec{\mathbf{v}} = [s, \vec{\mathbf{v}}]$

$$\mathbf{q} = q_0 + q_1\vec{\mathbf{i}} + q_2\vec{\mathbf{j}} + q_3\vec{\mathbf{k}}$$

Properties:

Linearity: $\lambda\mathbf{q}_1 + \mu\vec{\mathbf{q}}_2 = [\lambda s_1 + \mu s_2, \lambda\vec{\mathbf{v}}_1 + \mu\vec{\mathbf{v}}_2]$

Conjugate: $\mathbf{q}^* = s - \vec{\mathbf{v}} = [s, -\vec{\mathbf{v}}]$

Product: $\mathbf{q}_1 \circ \mathbf{q}_2 = [s_1 s_2 - \vec{\mathbf{v}}_1 \cdot \vec{\mathbf{v}}_2, s_1\vec{\mathbf{v}}_2 + s_2\vec{\mathbf{v}}_1 + \vec{\mathbf{v}}_1 \times \vec{\mathbf{v}}_2]$

Transform vector: $\mathbf{q} \circ \vec{\mathbf{p}} = \mathbf{q} \circ [0, \vec{\mathbf{p}}] \circ \mathbf{q}^*$

Norm: $\|\mathbf{q}\| = \sqrt{s^2 + \vec{\mathbf{v}} \cdot \vec{\mathbf{v}}} = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2}$

Digression continued: unit quaternions

We can associate a rotation by angle θ about an axis $\vec{\mathbf{n}}$ with the unit quaternion:

$$Rot(\vec{\mathbf{n}},\theta) \Leftrightarrow \left[\cos\frac{\theta}{2},\ \sin\frac{\theta}{2}\vec{\mathbf{n}}\right]$$

Exercise: Demonstrate this relationship. I.e., show

$$Rot((\vec{\mathbf{n}},\theta)\vec{\mathbf{p}} = \left[\cos\frac{\theta}{2},\ \sin\frac{\theta}{2}\vec{\mathbf{n}}\right] \circ [0,\vec{\mathbf{p}}] \circ \left[\cos\frac{\theta}{2},\ -\sin\frac{\theta}{2}\vec{\mathbf{n}}\right]$$

Rotation matrix from unit quaternion

$$\mathbf{q} = [q_0, q_1, q_2, q_3]; \quad \|\mathbf{q}\| = 1$$

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^3 - {q_3}^3 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & {q_0}^2 - {q_1}^2 + {q_2}^3 - {q_3}^3 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & {q_0}^2 - {q_1}^2 - {q_2}^3 + {q_3}^3 \end{bmatrix}$$

Unit quaternion from rotation matrix

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} r_{xx} & r_{yx} & r_{zx} \\ r_{xy} & r_{yy} & r_{zy} \\ r_{xz} & r_{yz} & r_{zz} \end{bmatrix};$$

$$a_0 = 1 + r_{xx} + r_{yy} + r_{zz};\ a_1 = 1 + r_{xx} - r_{yy} - r_{zz}$$
$$a_2 = 1 - r_{xx} + r_{yy} - r_{zz};\ a_3 = 1 - r_{xx} - r_{yy} + r_{zz}$$

$a_0 = \max\{a_k\}$	$a_1 = \max\{a_k\}$	$a_2 = \max\{a_k\}$	$a_3 = \max\{a_k\}$
$q_0 = \frac{\sqrt{a_0}}{2}$	$q_0 = \frac{r_{yz} - r_{zy}}{4q_1}$	$q_0 = \frac{r_{zx} - r_{xz}}{4q_2}$	$q_0 = \frac{r_{xy} - r_{yx}}{4q_3}$
$q_1 = \frac{r_{xy} - r_{yx}}{4q_0}$	$q_1 = \frac{\sqrt{a_1}}{2}$	$q_1 = \frac{r_{xy} + r_{yx}}{4q_2}$	$q_1 = \frac{r_{xz} + r_{zx}}{4q_3}$
$q_2 = \frac{r_{zx} - r_{xz}}{4q_0}$	$q_2 = \frac{r_{xy} + r_{yx}}{4q_1}$	$q_2 = \frac{\sqrt{a_2}}{2}$	$q_2 = \frac{r_{yz} + r_{zy}}{4q_3}$
$q_3 = \frac{r_{yz} - r_{zy}}{4q_0}$	$q_3 = \frac{r_{xz} + r_{zx}}{4q_1}$	$q_3 = \frac{r_{yx} + r_{xy}}{4q_2}$	$q_3 = \frac{\sqrt{a_3}}{2}$

Quaternion method for R

Step 1: Compute

$$\mathbf{H} = \sum_i \begin{bmatrix} \bar{a}_{i,x}\bar{b}_{i,x} & \bar{a}_{i,x}\bar{b}_{i,y} & \bar{a}_{i,x}\bar{b}_{i,z} \\ \bar{a}_{i,y}\bar{b}_{i,x} & \bar{a}_{i,y}\bar{b}_{i,y} & \bar{a}_{i,z}\bar{b}_{i,z} \\ \bar{a}_{i,z}\bar{b}_{i,x} & \bar{a}_{i,z}\bar{b}_{i,y} & \bar{a}_{i,z}\bar{b}_{i,z} \end{bmatrix}$$

Step 2: Compute

$$\mathbf{G} = \begin{bmatrix} trace(\mathbf{H}) & \Delta^T \\ \Delta & \mathbf{H} + \mathbf{H}^T - trace(\mathbf{H})\mathbf{I} \end{bmatrix}$$

where $\Delta^T = \begin{bmatrix} \mathbf{H}_{2,3} - \mathbf{H}_{3,2} & \mathbf{H}_{3,1} - \mathbf{H}_{1,3} & \mathbf{H}_{1,2} - \mathbf{H}_{2,1} \end{bmatrix}$

Step 3: Compute eigen value decomposition of $\mathbf{G}$

$\text{diag}(\bar{\lambda}) = \mathbf{Q}^T\mathbf{G}\mathbf{Q}$

Step 4: The eigenvector $\mathbf{Q}_k = [q_0, q_1, q_2, q_3]$ corresponding to the largest eigenvalue λ_k is a unit quaternion corresponding to the rotation.

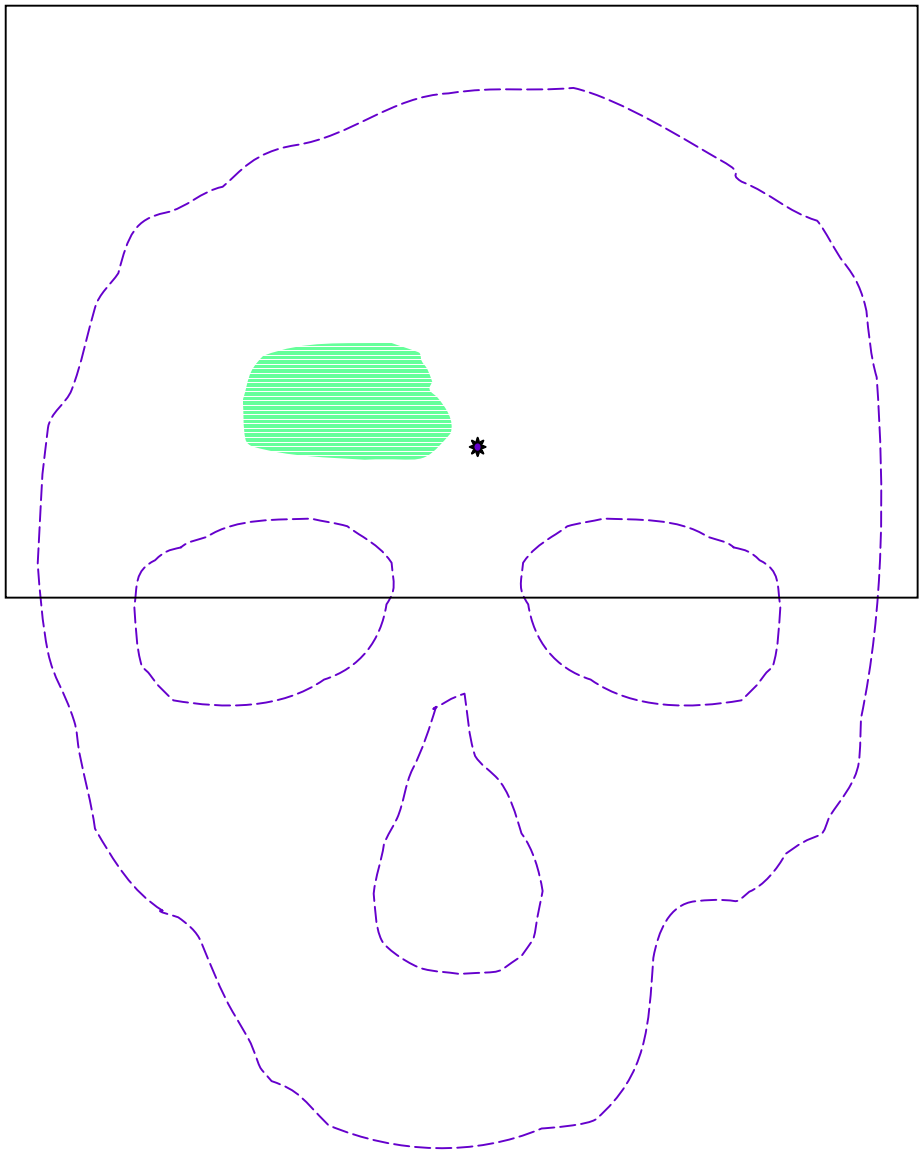
Iterative Closest Point

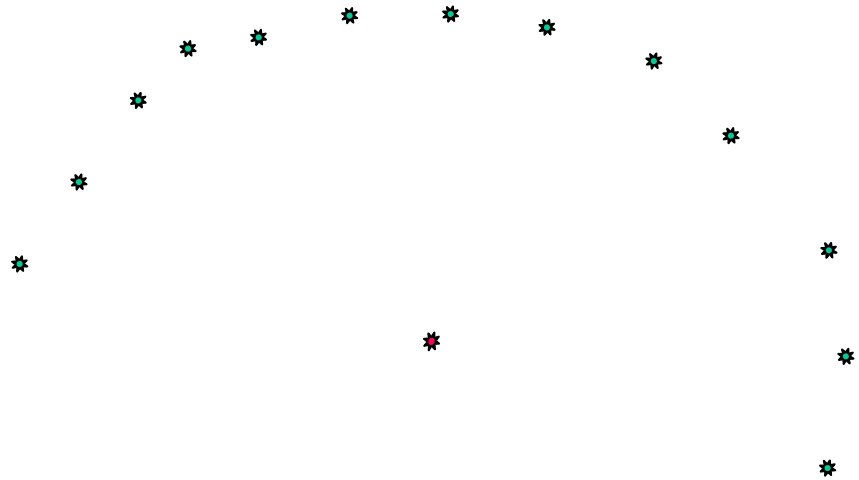
- Besl and McKay, 1992
- Start with an initial guess, $\mathbf{T}_0$, for $\mathbf{T}$.
- At iteration k
  1. For each sampled point $\mathbf{f}_i \in \mathcal{F}_A$. find the point $\mathbf{v}_i \in \mathcal{F}_B$ that is closest to $\mathbf{T}_k \cdot \mathbf{f}_i$.
  2. Then compute $\mathbf{T}_{k+1}$ as the transformation that minimizes

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

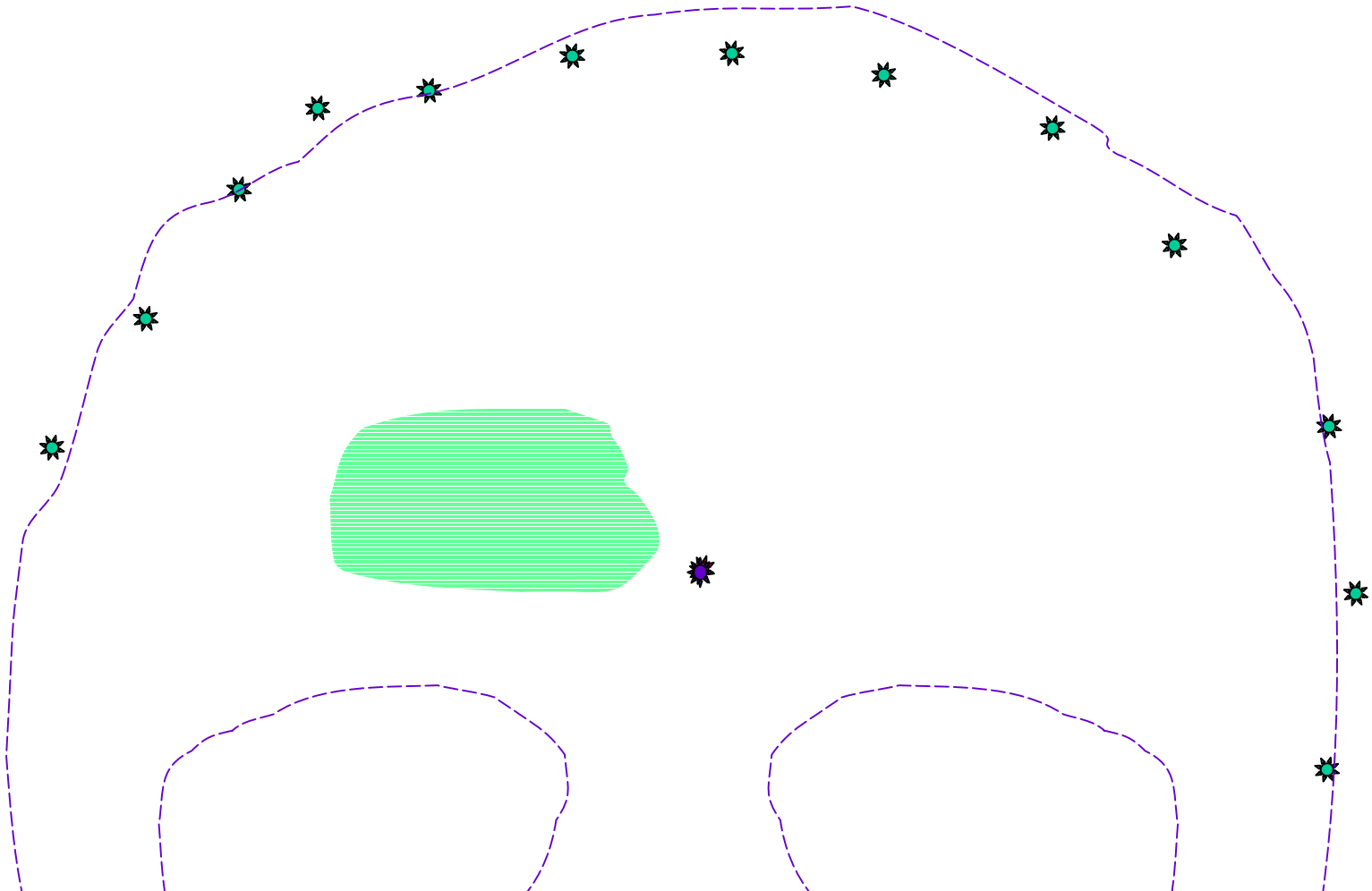
- Physical Analogy

Iterative Closest Point: step 0

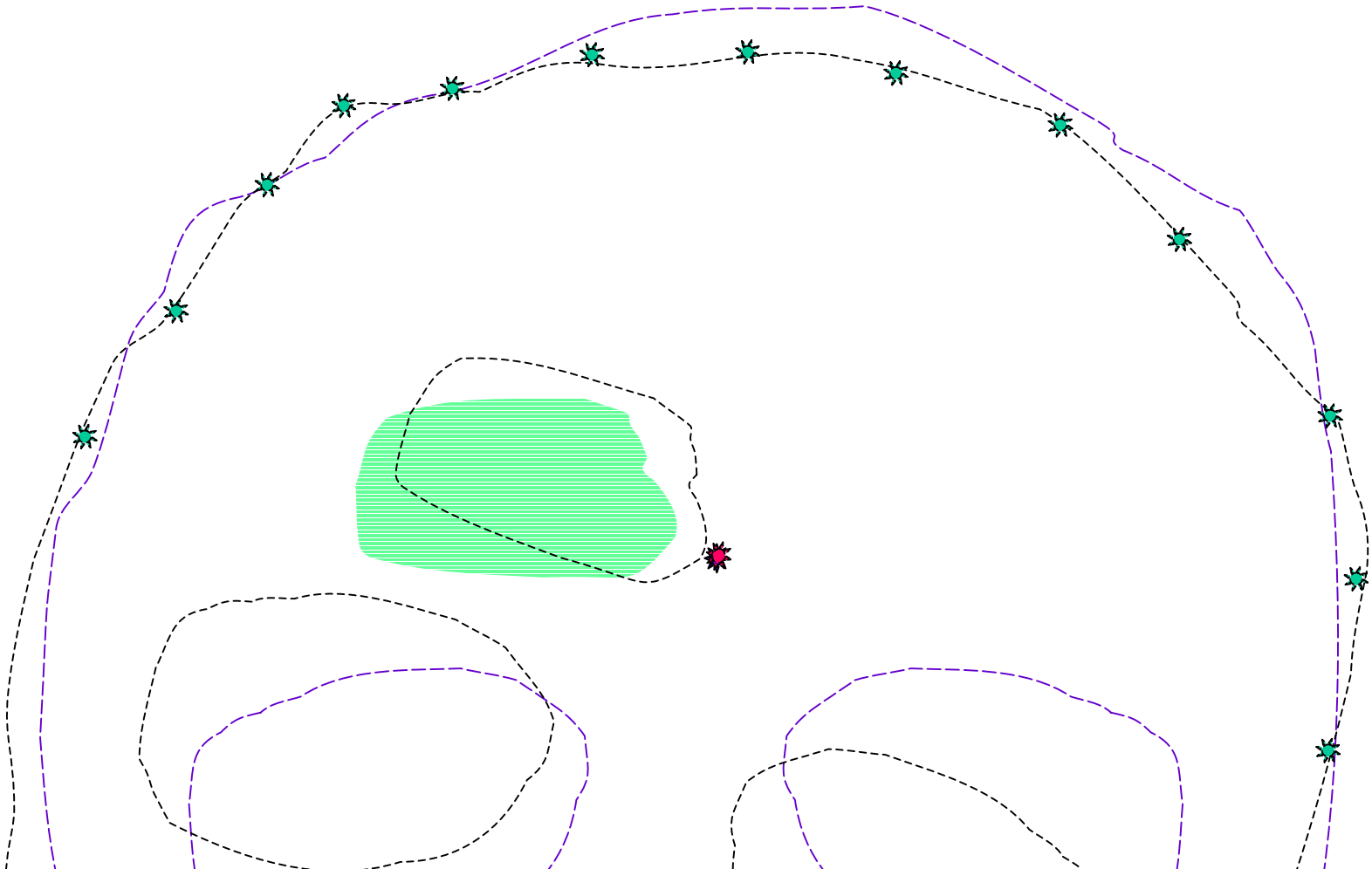




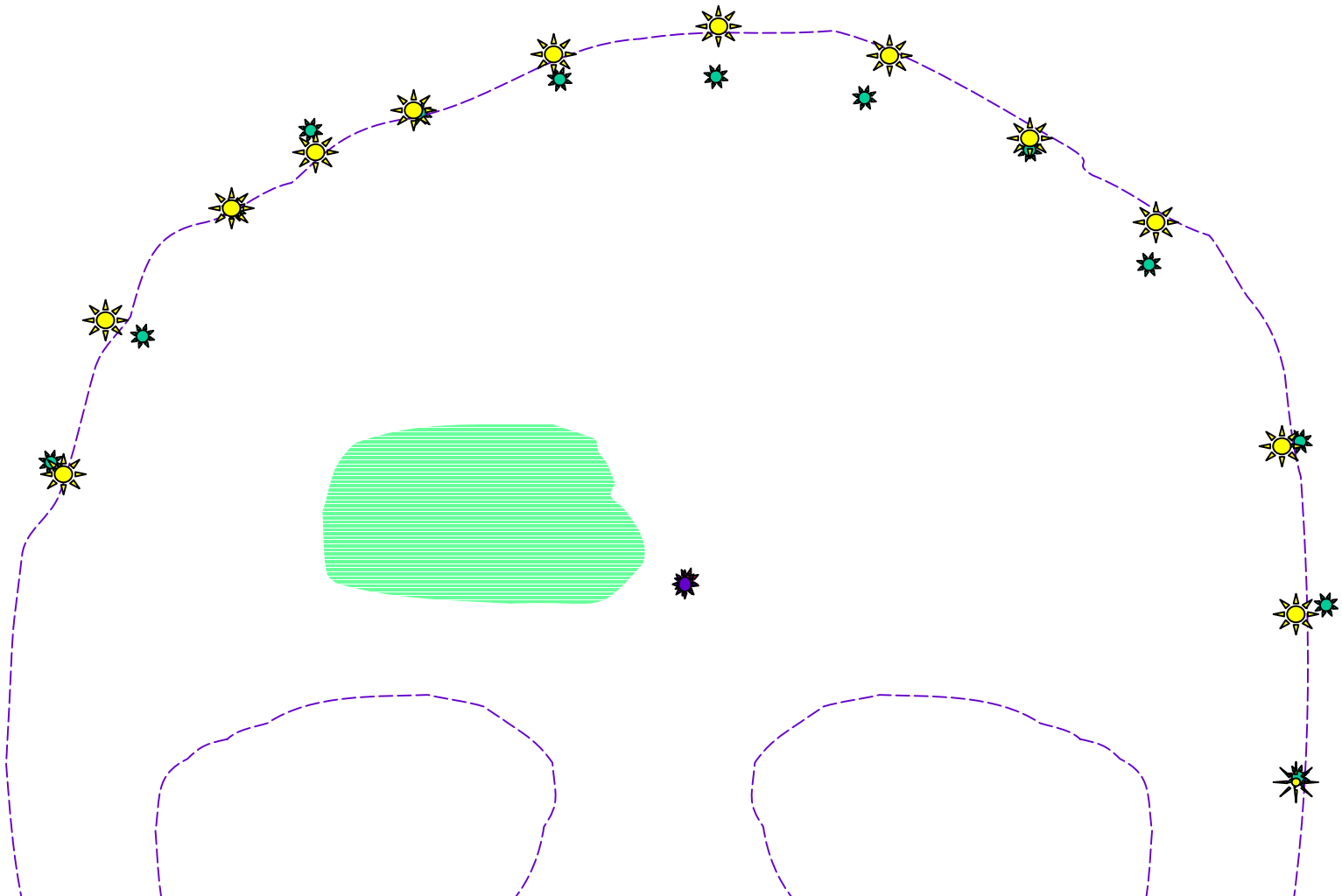
Iterative Closest Point: step1



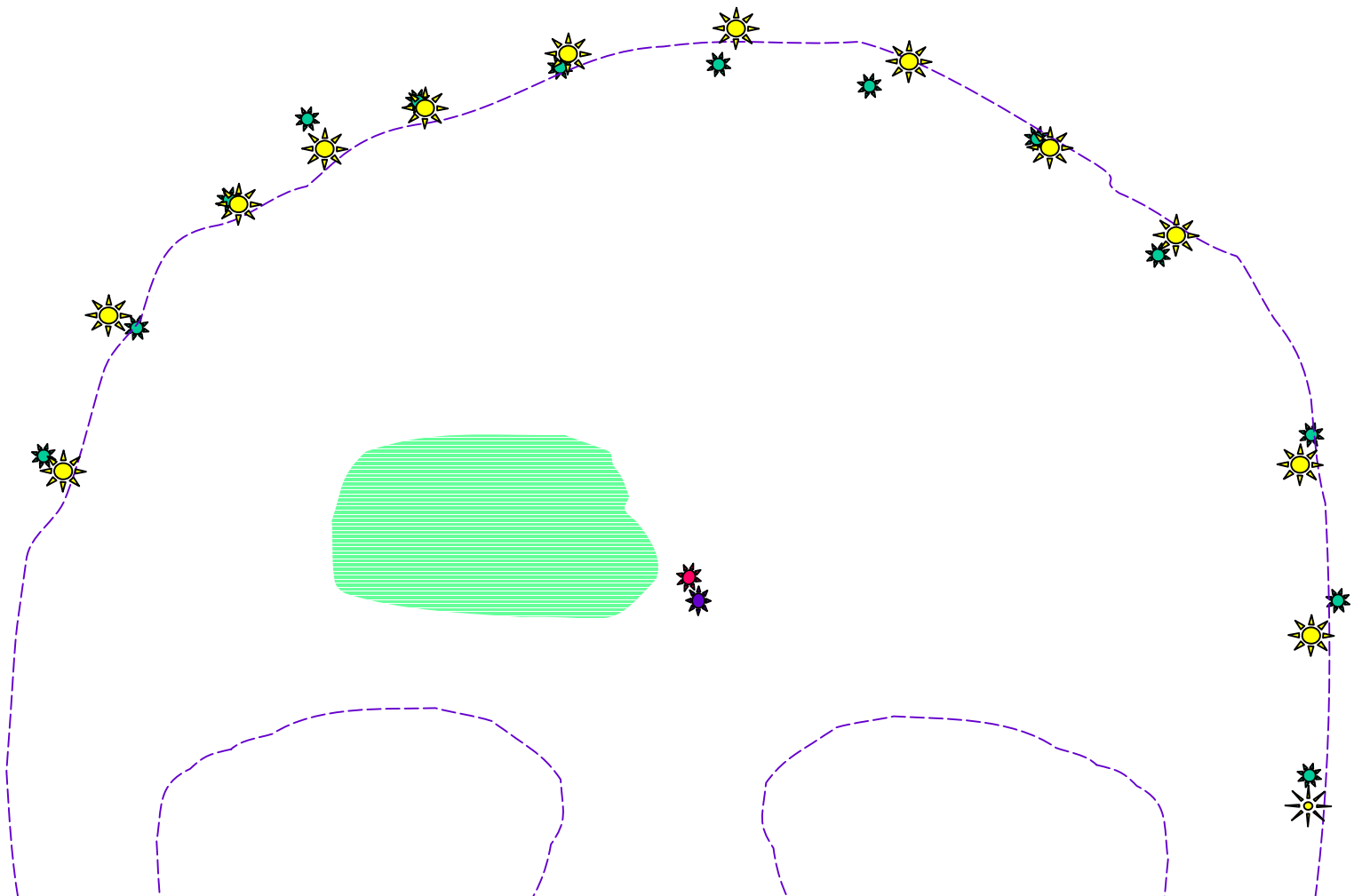
Iterative Closest Point: step1



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

Outline of a practical ICP code

Given

1. Surface model $\boldsymbol{M}$ consisting of triangles $\{\boldsymbol{m}_i\}$
2. Set of points $\boldsymbol{Q} = \{\vec{\mathbf{q}}_1, \cdots, \vec{\mathbf{q}}_N\}$ known to be on $\boldsymbol{M}$.
3. Initial guess $\mathbf{F}_0$ for transformation $\mathbf{F}_0$ such that the points $\mathbf{F}\vec{\mathbf{q}}_k$ lie on $\boldsymbol{M}$.
4. Initial threshold η_0 for match closeness

Outline of a practical ICP code

Temporary variables

n	Iteration number
$\mathbf{F}_n = [\mathbf{R}, \vec{\mathbf{p}}]$	Current estimate of transformation
η_n	Current match distance threshold
$\boldsymbol{C} = \{\cdots, \vec{\mathbf{c}}_k, \cdots\}$	Closest points on $\boldsymbol{M}$ to $\boldsymbol{Q}$
$\boldsymbol{D} = \{\cdots, d_k, \cdots\}$	Distances $d_k = \lVert \vec{\mathbf{c}}_k - \mathbf{F}_n \cdot \vec{\mathbf{q}}_k \rVert$
$\boldsymbol{I} = \{\cdots, i_k, \cdots\}$	Indices of triangles $\boldsymbol{m}_{i_k}$ corresp. to $\vec{\mathbf{c}}_k$
$\boldsymbol{A} = \{\cdots, \vec{\mathbf{a}}_k, \cdots\}$	Subset of $\boldsymbol{Q}$ with valid matches
$\boldsymbol{B} = \{\cdots, \vec{\mathbf{b}}_k, \cdots\}$	Points on $\boldsymbol{M}$ corresponding to $\boldsymbol{A}$
$\boldsymbol{E} = \{\cdots, \vec{\mathbf{e}}_k, \cdots\}$	Residual errors $\vec{\mathbf{b}}_k - \mathbf{F} \cdot \vec{\mathbf{a}}_k$
$\sigma_n,\ (\varepsilon_{\max})_n, \overline{\varepsilon}_n$	$\frac{\sum_k \vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}{NumElts(\boldsymbol{E})}$; $\max_k \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}$; $\frac{\sum_k \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}}{NumElts(\boldsymbol{E})}$

Outline of a practical ICP code

Step 0 : (initialization)

Input surface model $\boldsymbol{M}$ and points $\boldsymbol{Q}$.

Build an appropriate data structure (e.g., octree, kD tree) $\boldsymbol{T}$ to facilitate finding the closest point matching search.

$n \leftarrow 0$

$\boldsymbol{I} \leftarrow \{\cdots, 1, \cdots\}$

$\boldsymbol{C} \leftarrow \{\cdots, \text{point on } \boldsymbol{m}_1, \cdots\}$

$\boldsymbol{D} \leftarrow \{\cdots, \|\vec{\mathbf{c}}_k - \mathbf{F}_0 \vec{\mathbf{q}}_k\|, \cdots\}$

Outline of a practical ICP code

Step 1: (matching)

$\boldsymbol{A} \leftarrow \varnothing$; $\boldsymbol{B} \leftarrow \varnothing$

For $k \leftarrow 1$ step 1 to N do

begin

$[\vec{\mathbf{c}}_k, i_k, d_k] \leftarrow$ FindClosestPoint$(\mathbf{F}_n \vec{\mathbf{q}}_k, i_k, d_k, \boldsymbol{T})$;

// Note: develop first with simple

// search. Later make more

// sophisticated, using $\boldsymbol{T}$

if ($d_k < \eta_n$) then { put $\vec{\mathbf{q}}_k$ into $\boldsymbol{A}$; put $\vec{\mathbf{c}}_k$ into $\boldsymbol{B}$; };

// See also subsequent notes

end

Outline of a practical ICP code

Step 2 : (transformation update)

$n \leftarrow n+1$

$\mathbf{F}_n \leftarrow \text{FindBestRigidTransformation}(\boldsymbol{A}, \boldsymbol{B})$

$$\sigma_n \leftarrow \frac{\sqrt{\sum_{\mathrm{k}} \vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}}{NumElts(\boldsymbol{E})}; \quad (\varepsilon_{\max})_n \leftarrow \max_k \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}; \; \bar{\varepsilon}_n \leftarrow \frac{\sum_{\mathrm{k}} \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}}{NumElts(\boldsymbol{E})}$$

Step 3 : (adjustment)

Compute η_n from $\{\eta_0, \cdots, \eta_{n-1}\}$ // see notes next page

// May also update $\mathbf{F}_n$ from $\{\mathbf{F}_0, \cdots, \mathbf{F}_n\}$ (see Besl & McKay)

Step 4 : (iteration)

if TerminationTest($\{\sigma_0, \cdots, \sigma_n\}, \{(\varepsilon_{\max})_0, \cdots, (\varepsilon_{\max})_n, \{\bar{\varepsilon}_0, \cdots, \bar{\varepsilon}_n\}\}$)

then stop. Otherwise, go back to step 1 // see notes

Outline of practical ICP code

Threshold η_n update

The threshold η_n can be used to restrict the influence of clearly wrong matches on the computation of $\mathbf{F}_n$. Generally, it should start at a fairly large value and then decrease after a few iterations. One not unreasonable value might be something like $3(\varepsilon_{\max})_n$. If the number of valid matches begins to fall significantly, one can increase it adaptively.

Also, if the mesh is incomplete, it may be advantageous to excluse any matches with triangles at the edge of the mesh.

Outline of practical ICP code

Termination test

There are no hard and fast rules for deciding when to terminate the procedure. One criterion might be to stop when $\sigma_n, \bar{\varepsilon}_n$ and/or $(\varepsilon_{max})_n$ are less than desired thresholds and $1 \leq \bar{\varepsilon}_n / \bar{\varepsilon}_{n-1} \leq \gamma$ for some value γ (e.g., $\gamma \cong .95$) for several iterations.

Homework & Programming Assignments

- HW#3 (2002) covers analysis of a registration method similar to ICP
- PA#2 (2002) requires implementation of point cloud to point cloud registration
- PA#3 (2002) will require implementation of closest point pairing and performing one step of ICP
- PA#4 (2003) will require putting it all together, together with some possible refinements

Distance Maps

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

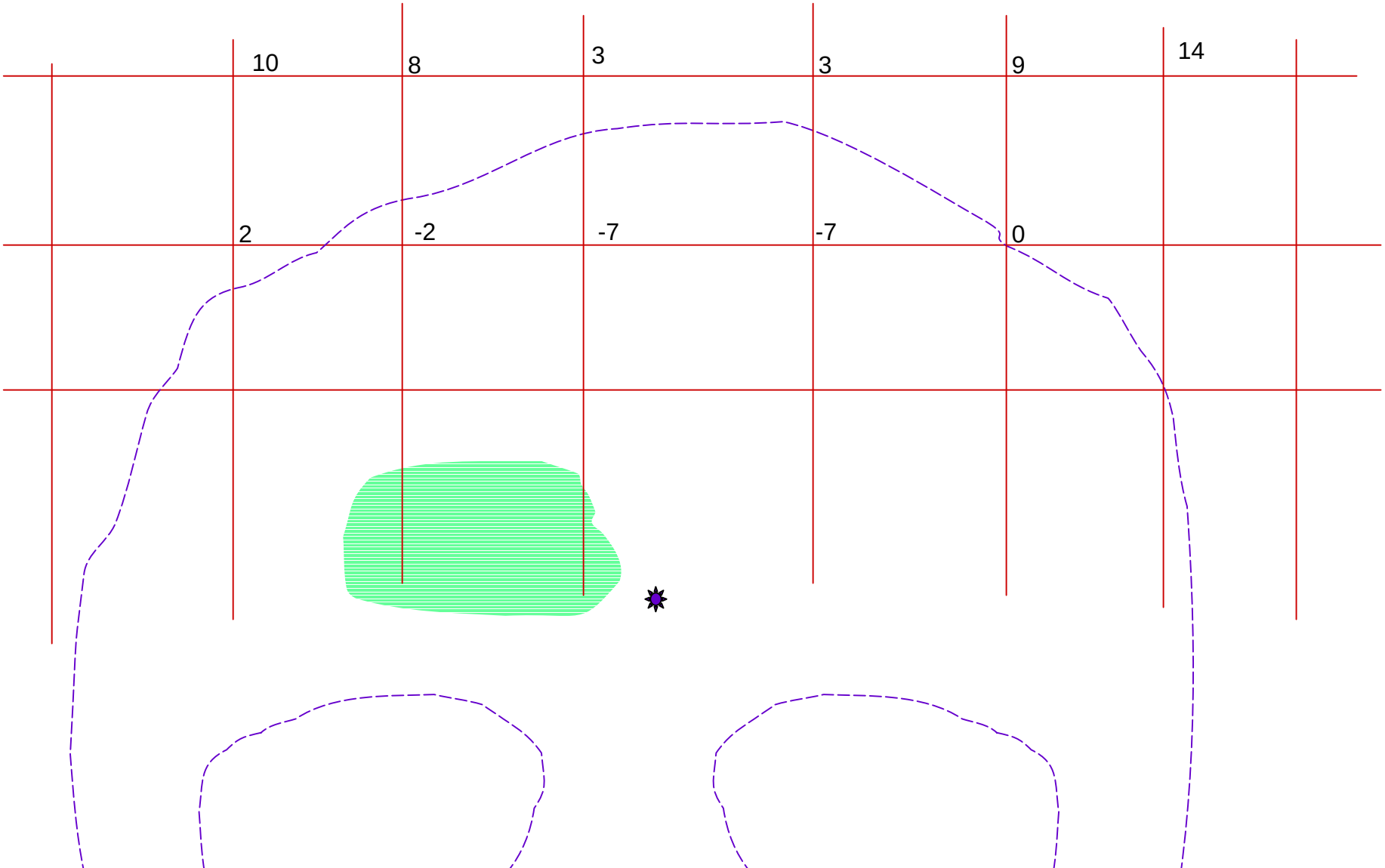
$$d_S(\mathcal{F}, \mathbf{f}) = \min_{\mathbf{p} \in \mathcal{F}} \|\mathbf{p} - \mathbf{f}\|$$

- But the problem is how to compute this quickly

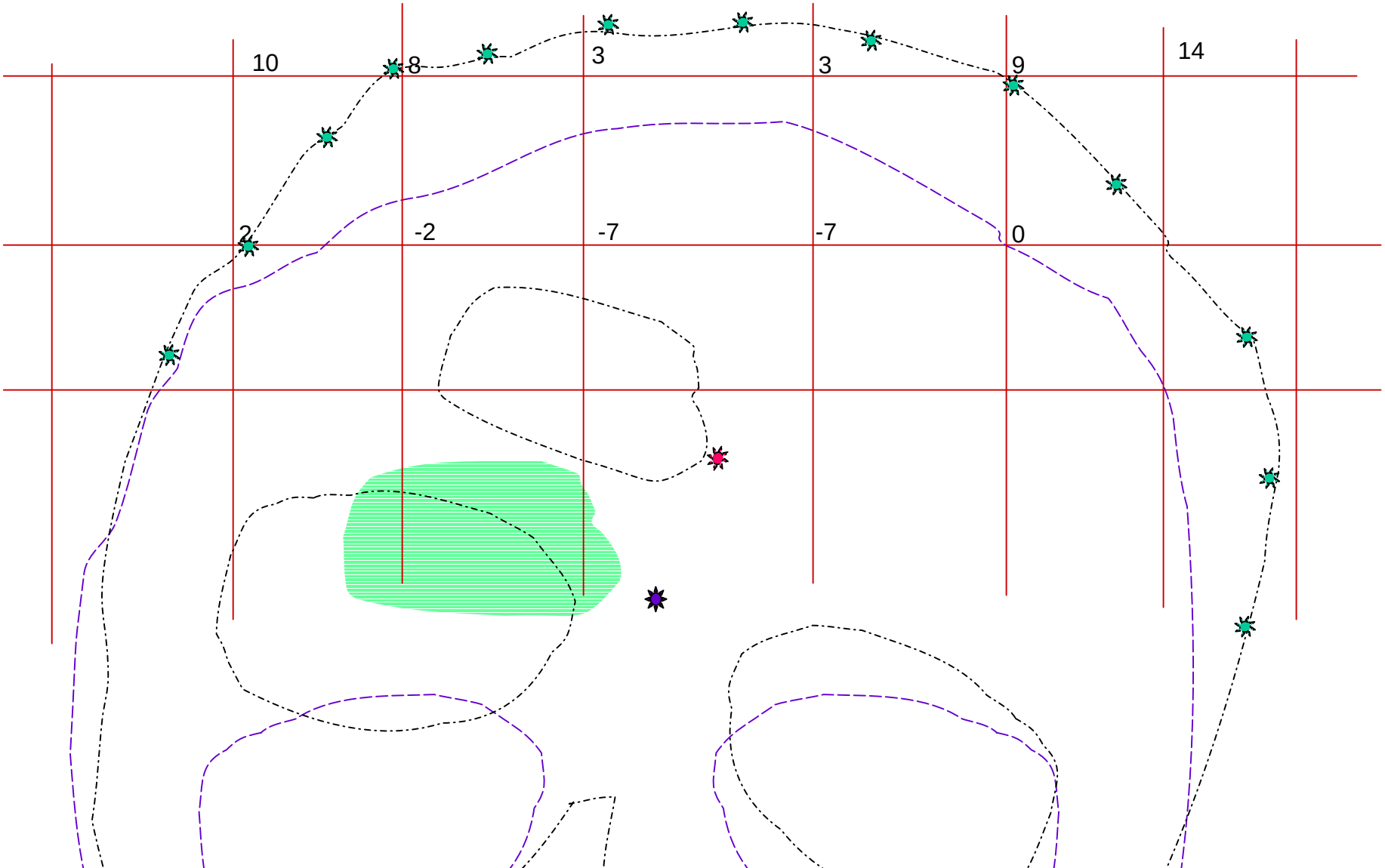
Distance Maps (Continued)

- 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

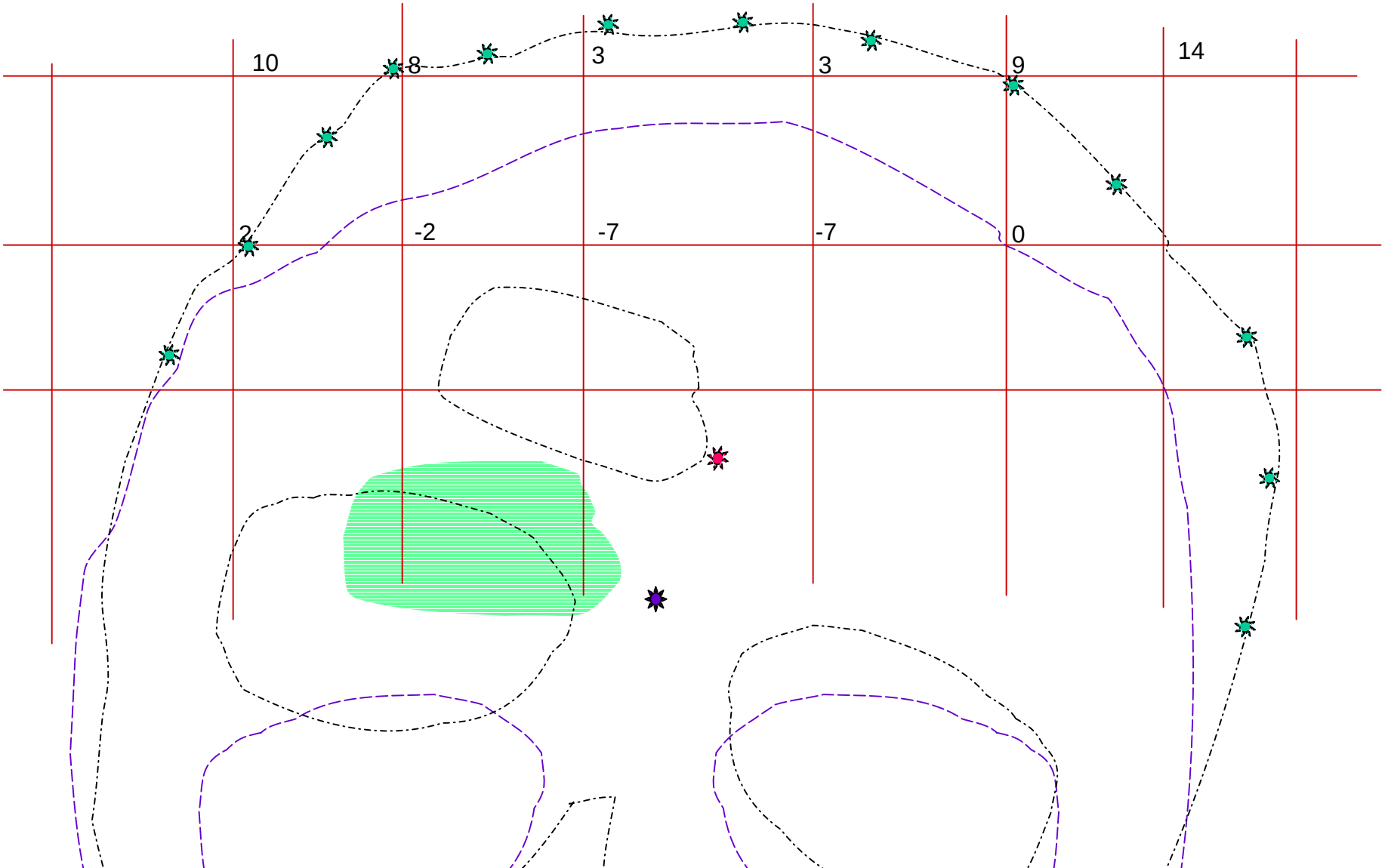
Distance Maps: step 0



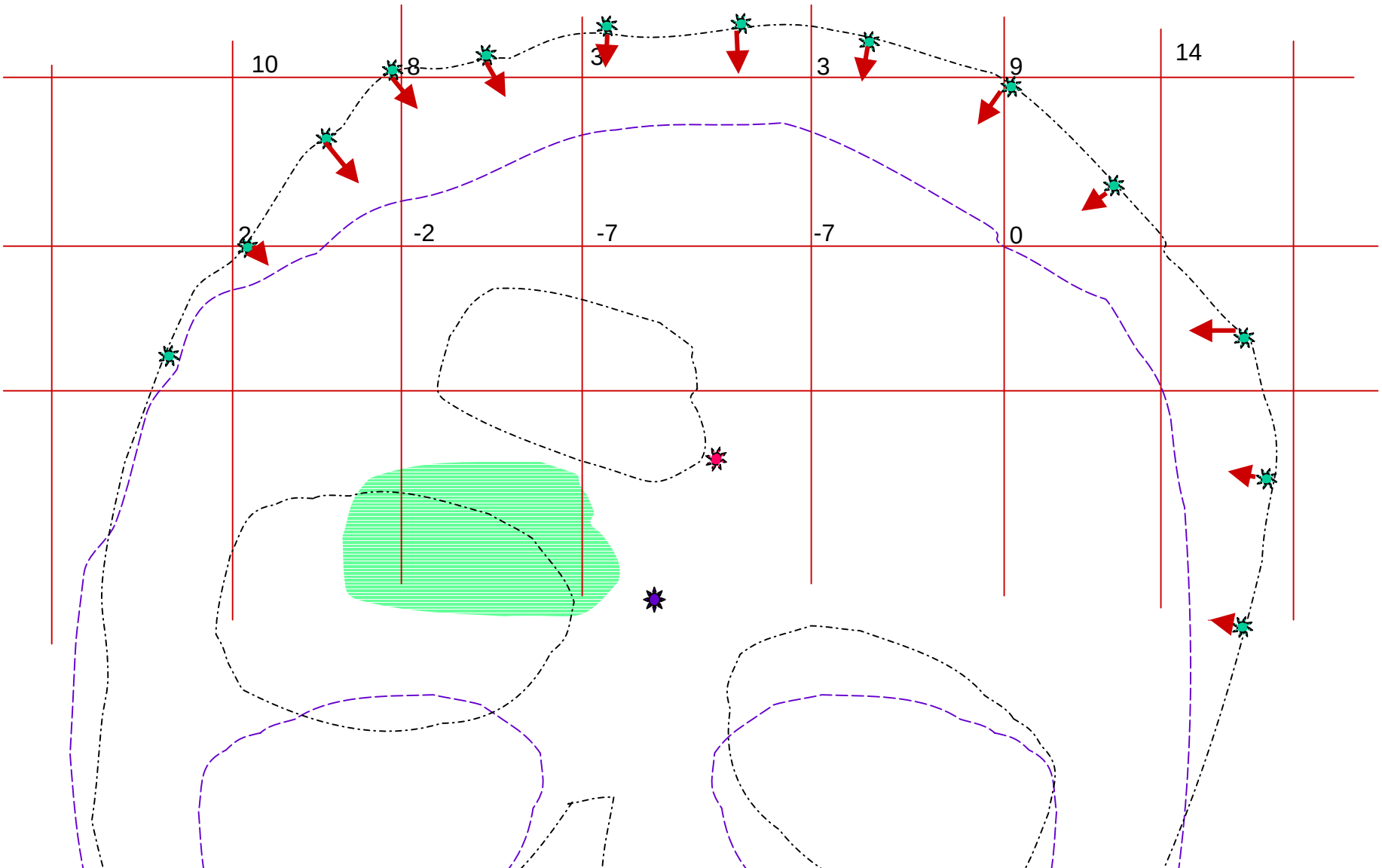
Distance Maps: step 1



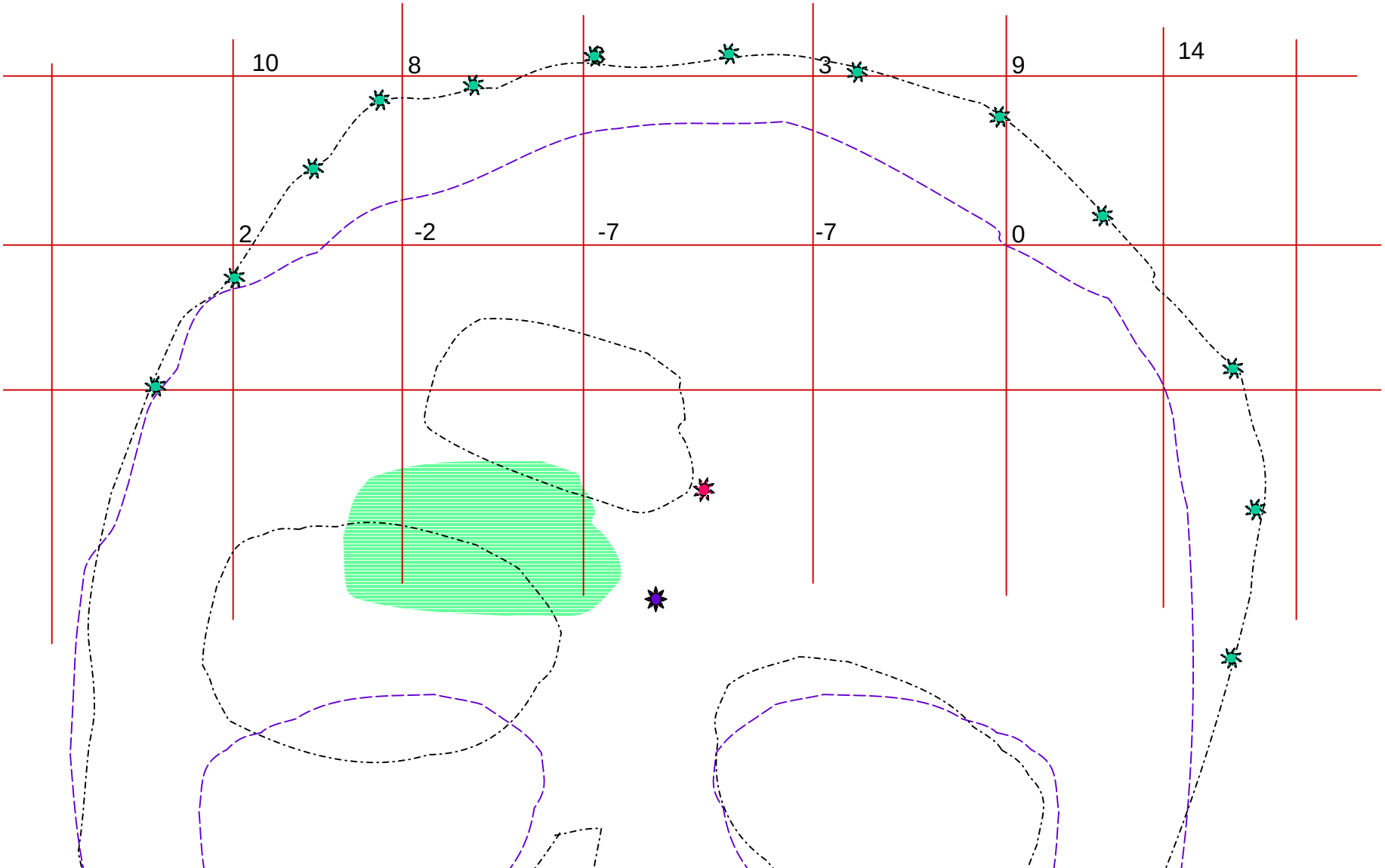
Distance Maps: step 1



Distance Maps: step 2



Distance Maps: step 3



Interpolating Distance Maps (2D Example)

$$d_S = \mu \ [\lambda d_{22} + (1-\lambda) d_{21}] + (1-\mu) \ [\lambda d_{12} + (1-\lambda) d_{11}]$$

$$\nabla d_S = \begin{bmatrix} \mu(d_{22} - d_{21}) + (1-\mu)(d_{12} - d_{11}) \\ \lambda(d_{22} - d_{12} + (1-\lambda)(d_{21} - d12) \end{bmatrix}$$

Distance Maps: Iteration Step

1. Determine cell $\mathcal{V}_i$ for each $\mathbf{p}_i = \mathbf{T} \cdot \mathbf{f}_i$. Let $\overline{\lambda_i}$ be the corresponding interpolation parameters for $\mathbf{p}_i$ within cell.

2. Determine small motion $\Delta\mathbf{T}$ that minimizes

$$\sum_i [(\Delta\mathbf{T}\mathbf{p}_i - \mathbf{p}_i) \cdot \nabla d_S(\overline{\lambda_i}, \mathcal{V}_i)]$$

   or

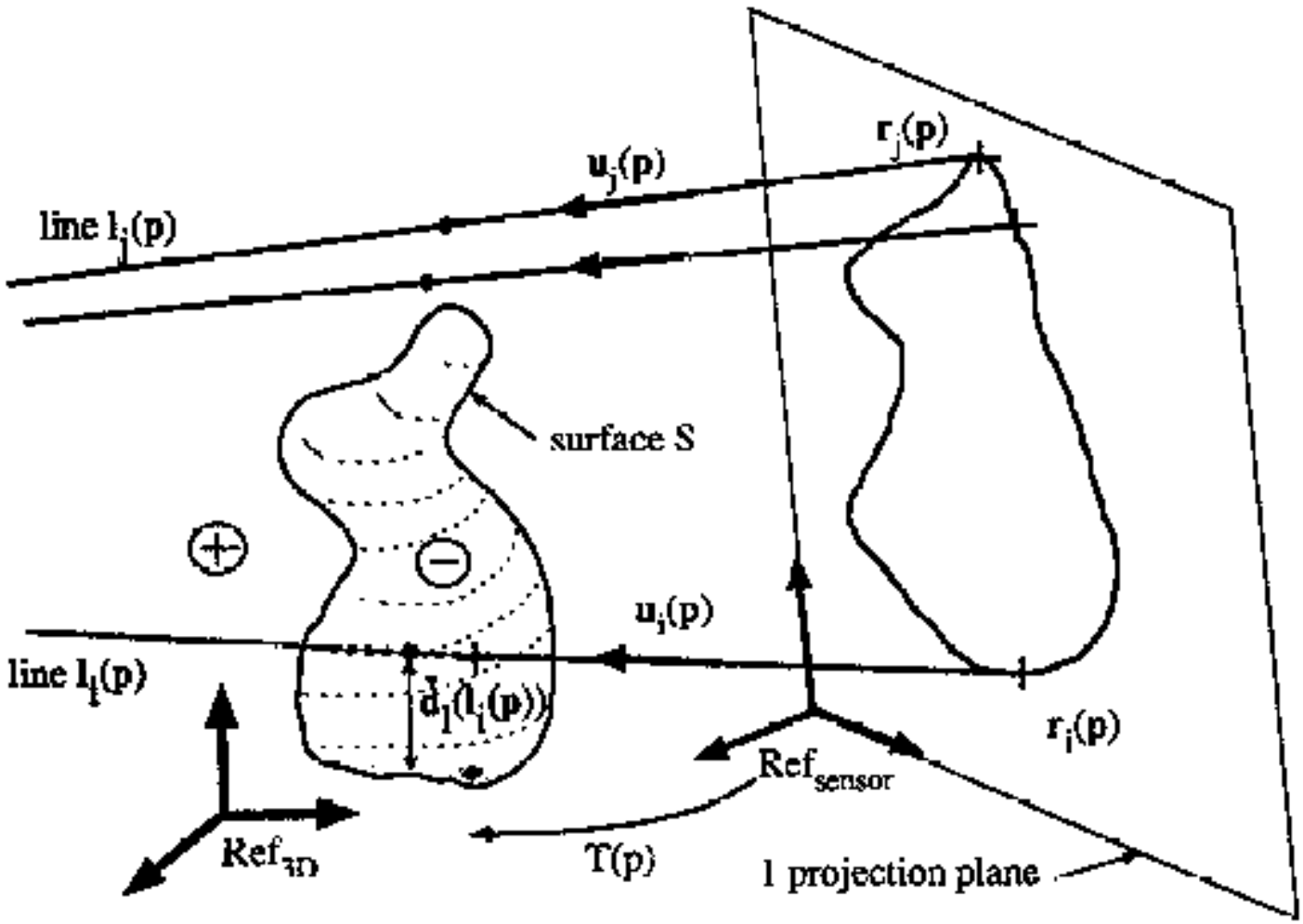
$$\sum_i -[(\Delta\mathbf{T}\mathbf{p}_i - \mathbf{p}_i) \cdot \nabla d_S(\overline{\lambda_i}, \mathcal{V}_i)]$$

3. Update $\mathbf{T} \leftarrow \Delta\mathbf{T} \bullet \mathbf{T}$

2D-to-3D using Octree Splines

- Lavallee, et. al
- Registers 2D images (e.g., x-ray) to 3D models
- Select 2D sample points (defines 3D rays) on contours
- Distance map method
  - Store distance of line to surface
  - Use Octree to store distance map
- Least squares minimization
- When converge, 3D surface is tangent to irregularly shaped cone defined by rays

2D-to-3D

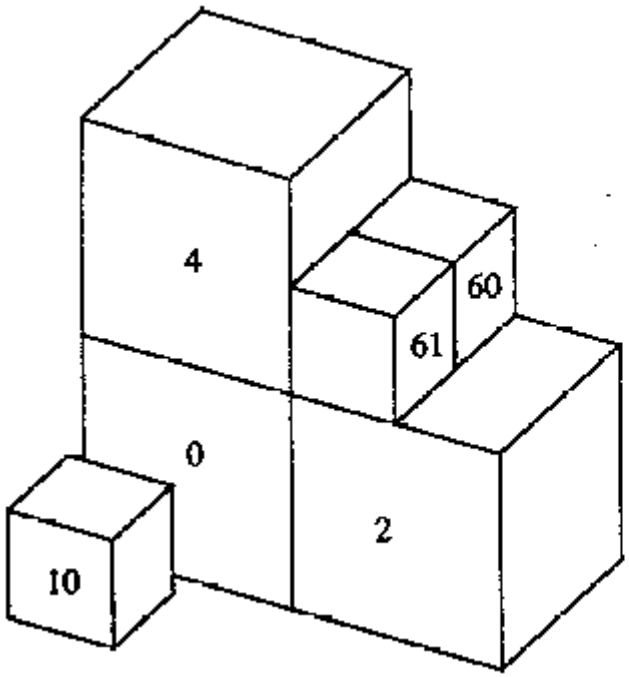


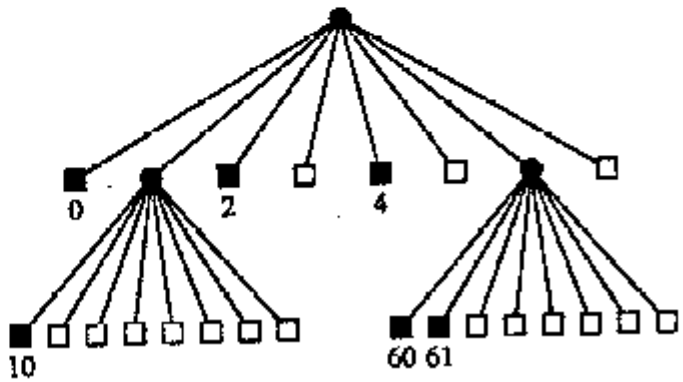
Source: Lavallee, CIS book

Octree Spline Distance Maps

- Select large number of points, $\mathbf{s}_j$ on the surface.
- Construct standard octree subdivision of space to encompass the $\mathbf{s}_j$
- Subdivide unoccupied cells near surface to be sure that adjacent cells differ in size from occupied cells by at most a factor of two.
- For each corner **c** of each octree cell, compute the euclidean distance $\mathbf{d_e(c)}$ to the closest point on the surface.
  - **Note**: Can do this with efficient tree walk of parents & neighbors of each cell

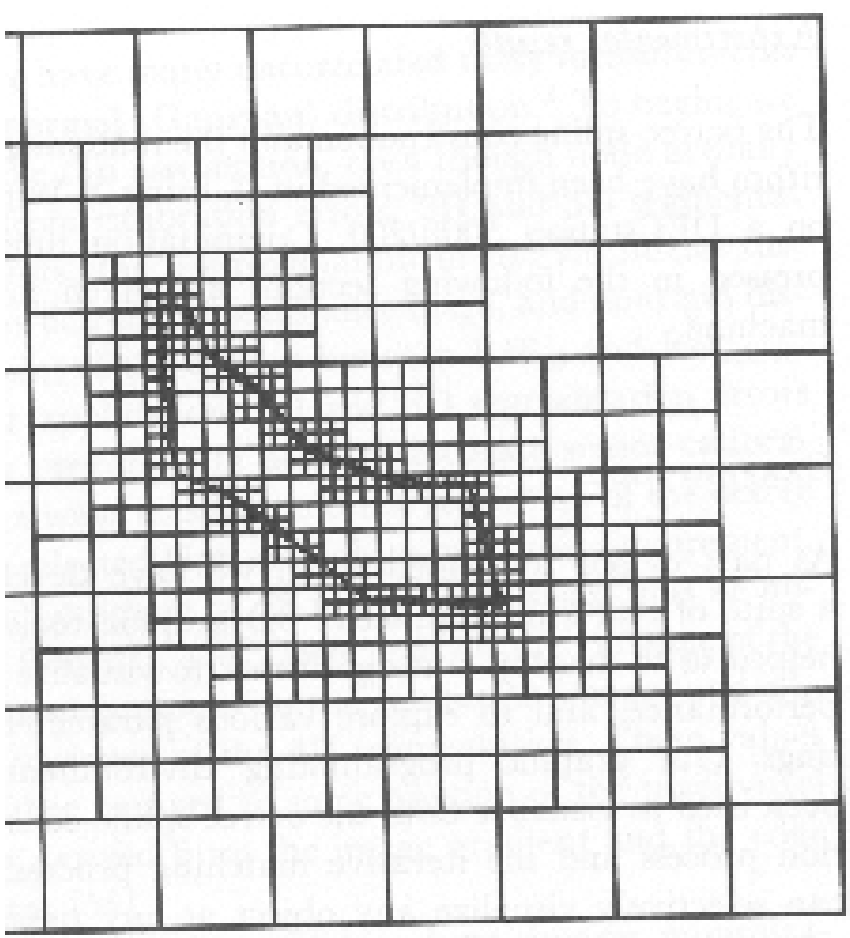
Octrees

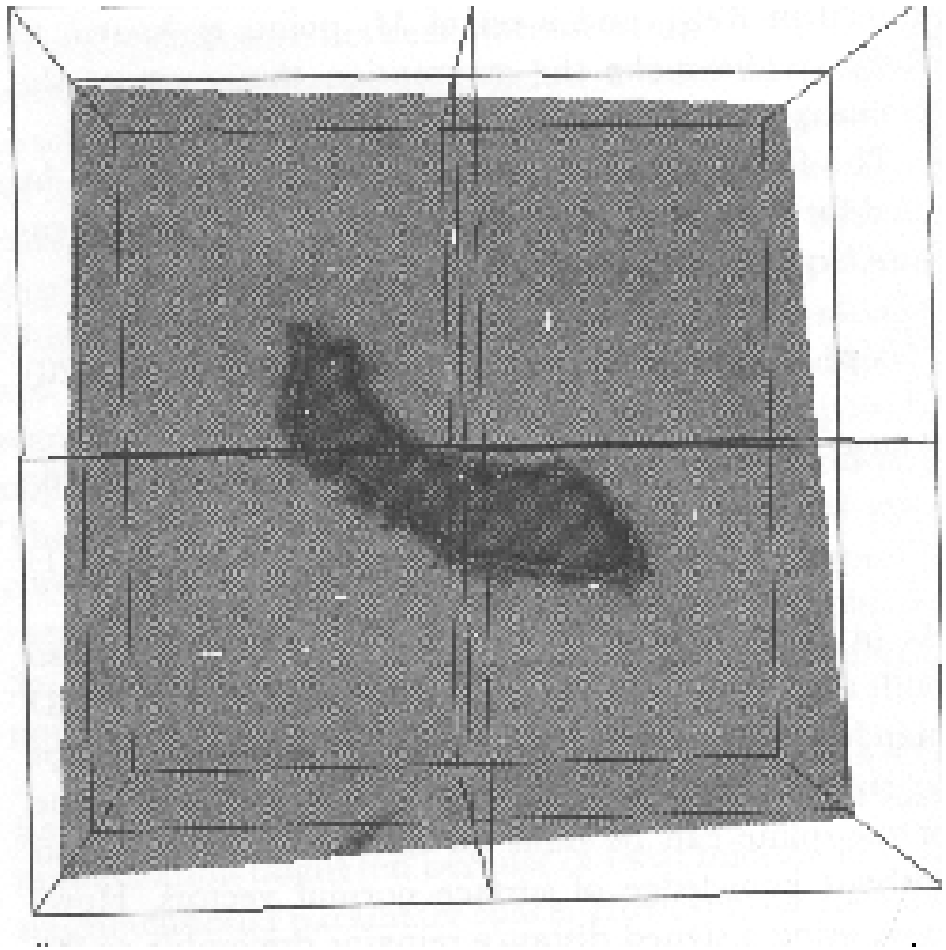




Source: Lavallee, CIS book

Octree-spline





Source: Lavallee, CIS book

Octree Spline Distance Maps (con'd)

- Compute signed distances **d'(c)** for each **c**
  - **Note:** assumes know something about the surface normals or can otherwise tell inside from outside
- Tweak the **d'(c)** to assure continuity of interpolations (a rather technical correction)
- Compute a lower bound on interpolation function value. I.e., for each node **N** of the octree, find a function $\mathbf{B_N(q)}$ such that $\mathbf{B_N(q)} \leq \mathbf{d'(q)}$ for all **q** in **N**.
  - This is done by a linear programming method defining $\mathbf{B_N(q)}=a\bullet\mathbf{q}+\mathbf{b}$, and then finding **a** and **b** such that $a\bullet\mathbf{q}+\mathbf{b}\leq\mathbf{d'(q)}$ for all **q** in $\mathbf{B_N(q)}$.

Line-to-surface distances

- Represent lines by

$$\mathbf{p}_i(\overline{\eta}, \lambda) = \mathbf{T}_{reg}(\overline{\eta})[\mathbf{q}_i + \lambda \mathbf{v}_i]$$

  where $\mathbf{q}$ is a point [on the contour], $\mathbf{v}$ is a direction, and λ is a free scalar variable. $\mathbf{T}_{reg}(\overline{\eta})$, is a transformation we are trying to find.

- $\tilde{d}_i(\mathbf{p}_i(\overline{\eta}, \lambda)$ can have local minima along λ, even if $\overline{\eta}$ is fixed.

  Caution Lavallee uses $\mathbf{p}$ for parameters of $\mathbf{T}$i find this confusing, so I have chosen to use $\overline{\eta}$.

  - Global search (e.g.. simulated annealing)
  - Exhaustive search of all cells line intersects
  - Tree-based search

Line-to-surface distances

Let $\mathbf{u}(\mathbf{p})$ be the normalized coordinates of a point $\mathbf{p}$ within a cell with corners $\{\mathbf{p}_{000}, \mathbf{p}_{001}, \ldots, \mathbf{p}_{111}\}$:

$$\mathbf{u}(\mathbf{p}) = \frac{\mathbf{p} - \mathbf{p}_{000}}{\mathbf{p}_{111} - \mathbf{p}_{000}}$$

where $\mathbf{p}_{000}$ and $\mathbf{p}_{111}$ are opposite corners of the cell. Let $\mathbf{a}$ and $\mathbf{b}$ be points where a line enters and leaves cell. Then, the equation of the line has the form

$$\begin{aligned} \mathbf{p}(\lambda) &= \lambda\mathbf{b} + (1-\lambda)\mathbf{a} \\ &= interpolate(\lambda, \mathbf{a}, \mathbf{b}) \end{aligned}$$

Note : $\tilde{d}$ is just **d'** of the earlier slides

Problem is to find

$$\min_{\lambda} \tilde{d}(\mathbf{p}(\lambda))$$

where

$$\begin{aligned} d(\mathbf{p}(\lambda)) &= interpolate(\mathbf{u}(\mathbf{p}(\lambda)), d_{000}, \ldots, d_{111}) \\ d_{ijk} &= \text{distance at } ijk'\text{th corner} \end{aligned}$$

As a practical matter, can just take $\min(\tilde{d}(\mathbf{a}), \tilde{d}(\mathbf{b}))$.

Energy Minimization

- Use standard technique (Lavenberg-Marquardt).
- Energy is

$$E(\overline{\eta}) = \sum_i \frac{1}{\sigma_i^2} \left[e_i(\overline{\eta}) \right]^2$$

I.e.

$$E(\overline{\eta}) = \sum_i \frac{1}{\sigma_i^2} \left[\min_\lambda \tilde{d}(\mathbf{T}(\overline{\eta}) \bullet (\mathbf{q}_i + \lambda \mathbf{v}_i)) \right]^2$$

- Gradient is

$$\frac{\partial e_i}{\partial \eta_i} = \left[\nabla \tilde{d}(\mathbf{T}(\overline{\eta}) \bullet (\mathbf{q}_i + \lambda_{min} \mathbf{v}_i)) \right] \cdot \left[\frac{\partial \mathbf{T}(\overline{\eta})}{\partial \eta_j} (\mathbf{q}_i + \lambda_{min} \mathbf{v}_i) \right]$$

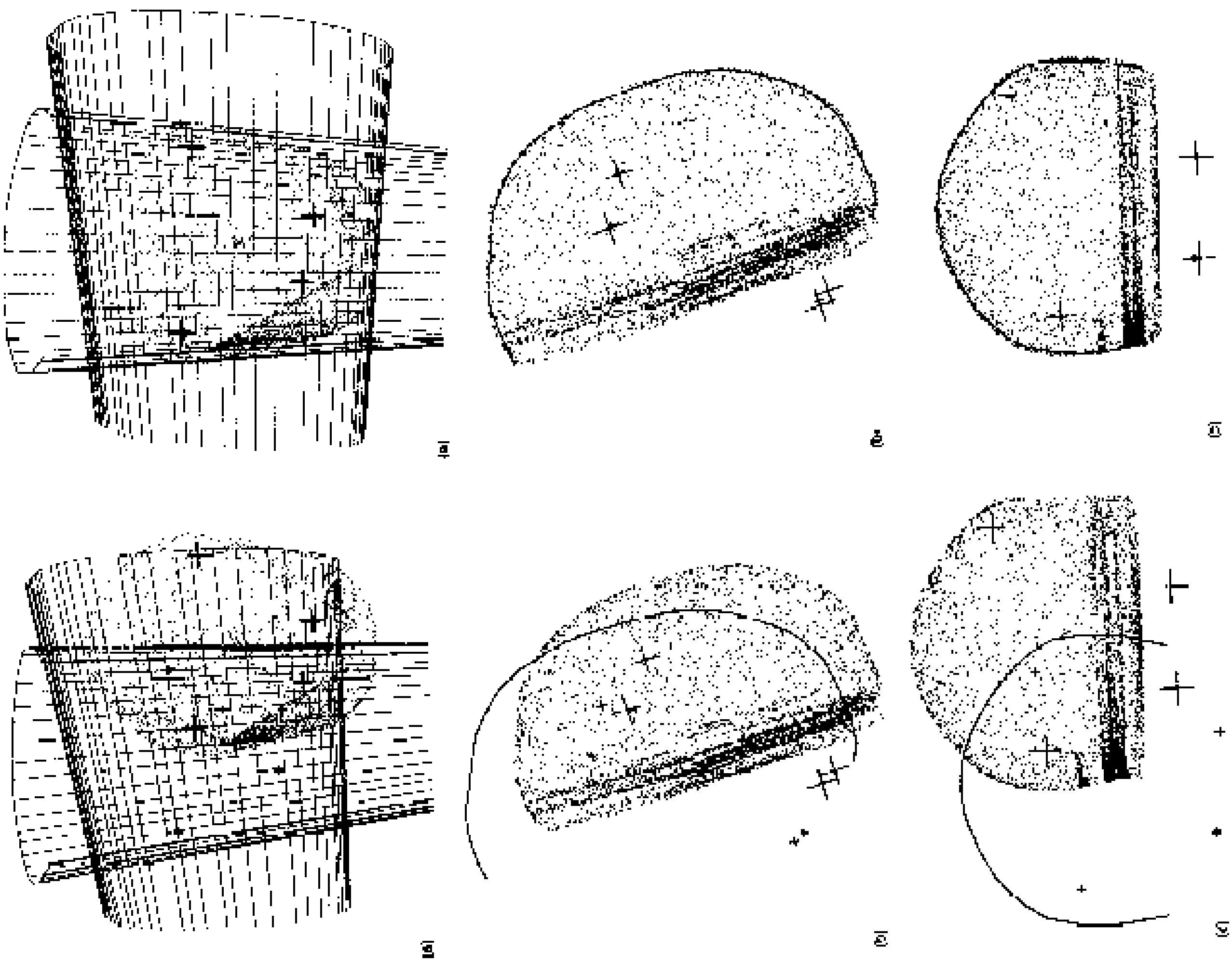
3D-to-3D with Octree Map

- Easy to adapt the mechanism of lines to points
- Use standard (unsigned) distance – saves step in computing map
- Energy function, given points $\mathbf{q}_i$, is

$$E(\overline{\eta}) = \sum_i \frac{1}{\sigma_i^2} \left[e_i(\overline{\eta})\right]^2 = \sum_i \frac{1}{\sigma_i^2} \left[\tilde{d}(\mathbf{T}(\overline{\eta})\mathbf{q}_i\right]^2$$

- Gradient is

$$\frac{\partial e_i}{\partial \eta_i} = \left[\nabla \tilde{d}(\mathbf{T}(\overline{\eta})\mathbf{q}_i)\right] \cdot \left[\frac{\partial \mathbf{T}(\overline{\eta})}{\partial \eta_j}\mathbf{q}_i\right]$$



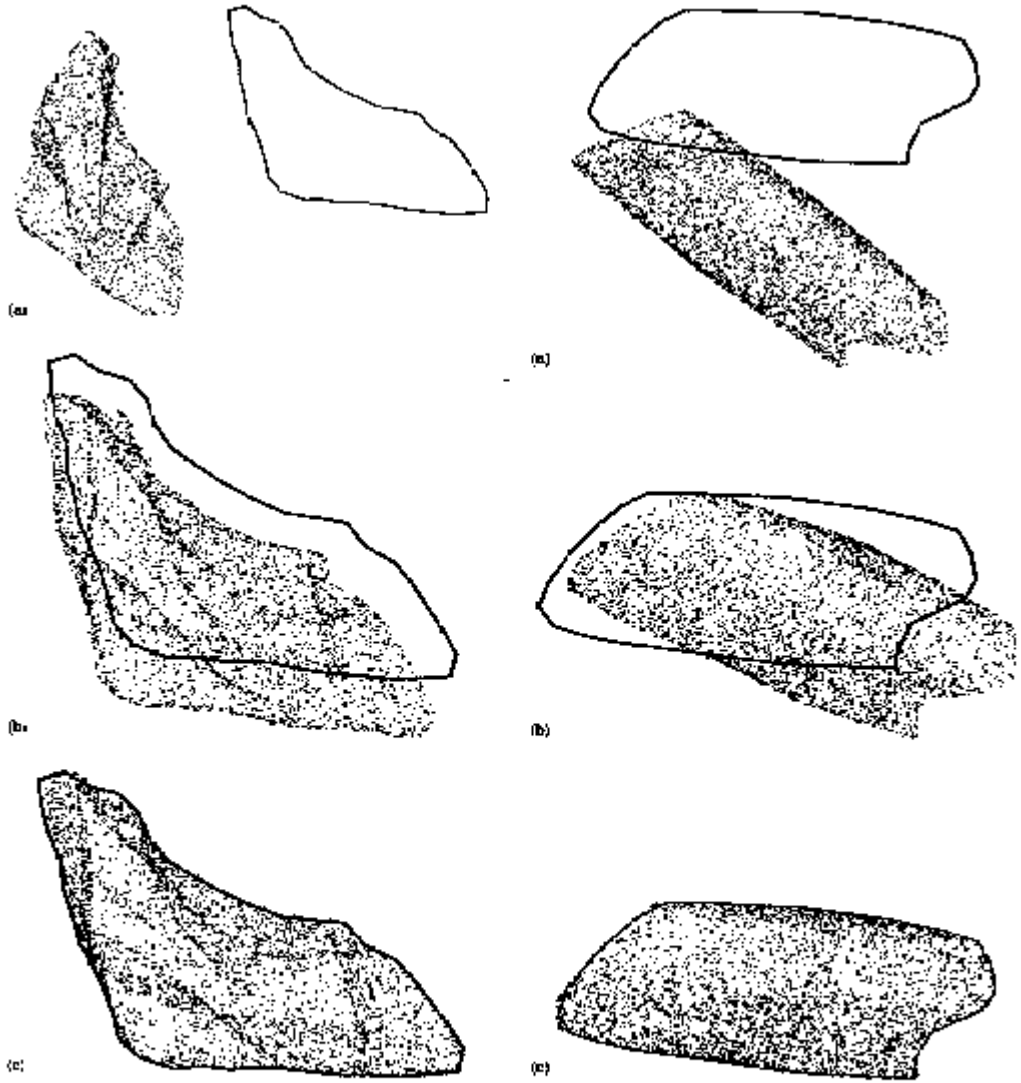


FIGURE 7.9 Convergence of algorithm for surface S_1 observed from the two projection viewpoints. The external contours of the projected surface end up fitting the real contours. (a) Initial configuration. (b) After two iterations. (c) After six iterations.

Sample Set Analysis

- **Question:** How good is a particular set of 3D sample points for the purpose of registration to a 3D surface?

- Long line of authors have looked at this question
- Next few slides are based on the work of David Simon, et al (1995)

Sample Set Analysis: Distance Estimates

Let

$$F(\mathbf{x}) = 0$$

be the implicit equation of a surface, then one good estimate of the distance of a point $\mathbf{x}$ to the surface is

$$D(\mathbf{x}) = \frac{F(\mathbf{x})}{\|\nabla F(\mathbf{x})\|}$$

Sample set analysis: sensitivity

Let $\mathbf{x}_s$ be a point on the surface, and let $T(\overline{\eta})$ represent a small perturbation with parameters $\overline{\eta}$ with respect to the surface of point $\mathbf{x}_s$:

$$\mathbf{x}'_s = T(\overline{\eta})\mathbf{x}_s$$

Then we define $\mathbf{V}(\mathbf{x}_s)$ to be

$$\mathbf{V}(\mathbf{x}_s) = \frac{\partial D(T(\overline{\eta})\mathbf{x}_s)}{\partial \overline{\eta}} = \begin{bmatrix} \mathbf{n}_s \\ \mathbf{x}_s \times \mathbf{n}_s \end{bmatrix}$$

where $\mathbf{n}_s$ is the unit normal to the surface at $\mathbf{x}_s$. So,

$$D(\mathbf{T}(\overline{\eta})\mathbf{x}_s)) \simeq \mathbf{V}^T(\mathbf{x}_s)\overline{\eta}$$

Squaring this gives

$$\begin{aligned} D^2(\mathbf{T}(\overline{\eta})\mathbf{x}_s)) &\simeq \overline{\eta}_T \mathbf{V}(\mathbf{x}_s)\mathbf{V}^T(\mathbf{x}_s)\overline{\eta} \\ &= \overline{\eta}^T \mathbf{M}(\mathbf{x}_s)\overline{\eta} \end{aligned}$$

Note that $\mathbf{M}$ is 6×6 positive, semi-definite, symmetric matrix.

Sample set analysis: sensitivity

For a region $\mathcal{R}$, define

$$\begin{aligned} E_R(\overline{\eta}) &= \overline{\eta}^T \left[\sum_{\mathbf{x}_s \in \mathcal{R}} \mathbf{M}(\mathbf{x}_s) \right] \overline{\eta} \\ &= \overline{\eta}^T \boldsymbol{\Psi}_{\mathcal{R}} \overline{\eta} \\ &= \overline{\eta}^T \mathbf{Q} \boldsymbol{\Lambda} \mathbf{Q}^T \overline{\eta} \\ &= \sum_{1 \le i \le 6} \lambda_i \left(\overline{\eta}^T \cdot \mathbf{q}_i \right)^2 \end{aligned}$$

- Note that the eigenvectors $\mathbf{q}_i$ correspond to small differential transformations $\mathbf{T}(\mathbf{q}_i)$. and can sort eigenvalues so that

$$\lambda_1 \ge \lambda_2 \ge \ldots \ge \lambda_6$$

- Note that eigenvector $\mathbf{q}_1$ corresponds to direction of greatest constraint.
- Similarly, can also think of $\mathbf{q}_6$ as the least constrained direction.

Sample Set Analysis: Goodness Measures

- Magnitude of smallest eigenvalue (Simon)
- (Kim and Khosla)

$$\frac{\sqrt[6]{\lambda_1 \cdot \ldots \cdot \lambda_6}}{\lambda_1 + \ldots + \lambda_6}$$

- Nahvi

$$\frac{\lambda_6^2}{\lambda_1}$$

Sample Set Selection

- One blind search method (similar to Simon, 1995) is:
  - Randomly select sample points on surface
  - (prune for reachability)
  - evaluate goodness of sample set using some criterion
  - repeat many times and choose the best one found

Sample Set Selection

- Refinement of blind search (hill climbing):
  - Randomly select sample points on surface
  - (prune for reachability)
  - evaluate goodness of sample set using some criterion
  - replace a point from sample set with a randomly selected point
  - evaluate goodness
  - if better, keep it
  - else revert to original point and try again
- Variations include simulated annealing, “genetic” algorithms

Sample Set Selection: Another Alternative

- Select large number of random points $\mathbf{x}_s$
- Prune for reachability
- For each point, compute constraint direction $\mathbf{V}_s = \mathbf{V}(\mathbf{x}_s)$. To a first approximation, a measurement at $\mathbf{x}_s$ with accuracy ϵ_s constrains $\overline{\eta}$ by

$$|\mathbf{V}_s \overline{\eta}| \leq \epsilon_s$$

- Now select subset of the $\mathbf{x}_s$ that minimizes, e.g.,

$$\min_{\delta_s} \max \overline{\eta}^T \mathbf{S} \overline{\eta}$$

subject to

$$\begin{aligned} \{\delta_s &\in \{0,1\} \\ |\delta_s \mathbf{V}_s \overline{\eta}| &\leq \epsilon_s \\ \sum_s \delta_s &\leq \text{subsetsize} \end{aligned}$$

There are various ways to do this.

Sample Set Selection: Another Alternative (con'd)

- One can also minimize other forms, e.g.,

$$\min_s \max_i |\sigma_i \eta_i|$$

subject to similar constraints

- An alternative is to minimize the number of sample points required to ensure that some constraints on $\overline{\eta}$ are guaranteed to be met. E.g.,

$$\min_{\delta_s} \sum \delta_s$$

such that

$$\begin{aligned} \delta_s &\in \{0, 1\} \\ \xi &\leq \xi_{limit} \end{aligned}$$

where

$$\xi = \max_{\overline{\eta}} \overline{\eta}^T \mathbf{S} \overline{\eta}$$

or some other form subject to

$$|\delta_s \mathbf{V}_s \overline{\eta}| \leq \epsilon_s$$

Deformable Altas-based Registration

- Material that follows is derived from Jianhua Yao's Ph.D. thesis work:
  - J. Yao, "Statistical bone density atlases and deformable medical image registrations", Ph. D. Thesis, Computer Science, The Johns Hopkins University, Baltimore, 2001.
- A number of other authors, including
  - Cootes et al. 1999
  - Feldmar and Ayache 1994
  - Ferrant et al. 1999
  - Fleute and Lavallee 1999
  - Lowe 1991
  - Maurer et al. 1996
  - Shen and Davatzikos 2000

Deformable Registration Between Density Atlas and Patient CT

- Goal: Register and Deform the statistical density atlas to match patient anatomy
- Significance:
  - Building patient specific model with same topology (mesh structure) as the atlas
  - Automatic segmentation
  - Accumulatively building models for training set
  - Pathological diagnosis

Deformable Registration Scheme

- Affine Transformation
  - Translation T=(t_x, t_y, t_z)
  - Rotation R=(r_x, r_y, r_z)
  - Scale S=(s_x, s_y, s_z)
- Global Deformation
  - Statistical deformation mode (M_i)
- Local Deformation
  - Adjustment of every vertex

Optimization Algorithm

- Direction Set (Powell's) methods in multi-dimensions
  - Search the parameter space to minimize the cost functions
  - Advantage
    - Don't need to compute derivative of cost functions
    - Much fewer evaluations than downhill simplex methods

Energy Function

- To measure the density and shape difference between model and image

$$E(mdl, img) = w_s E^{(s)}(mdl, img) + w_d E^{(d)}(mdl, img)$$

$$E^{(s)}(mdl, img) = \sum_{i=1}^{N(v)} \left(\vec{g}^{(mdl)}(v_i) \cdot \vec{g}^{(img)}(v_i) \right)$$

$$E^{(d)}(mdl, img) = \sum_{i=1}^{N(t)} \left(\oint_{\mu} \left(\frac{d^{(mdl)}(t_i, \mu) - d^{(img)}(t_i, \mu)}{d^{(mdl)}(t_i, \mu)} \right)^2 \right)$$

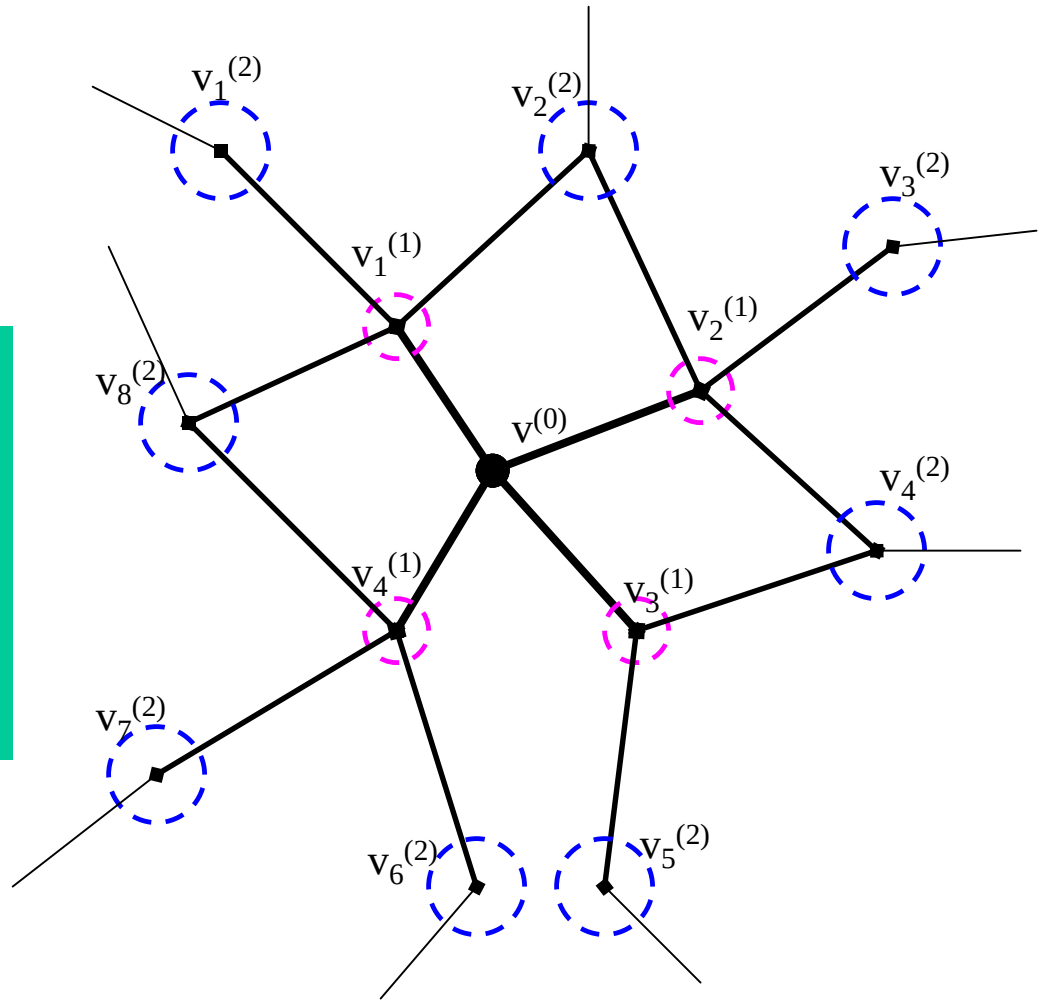
Local Deformation

- Motivation: Statistical deformation can’t capture all the variability due to the limited number of models in the training set
- Locally adjust the location of vertices to match the boundary of the bone and the interior density property
- Use multiple-layer flexible mesh template matching to find the correspondence between model vertices and image voxels

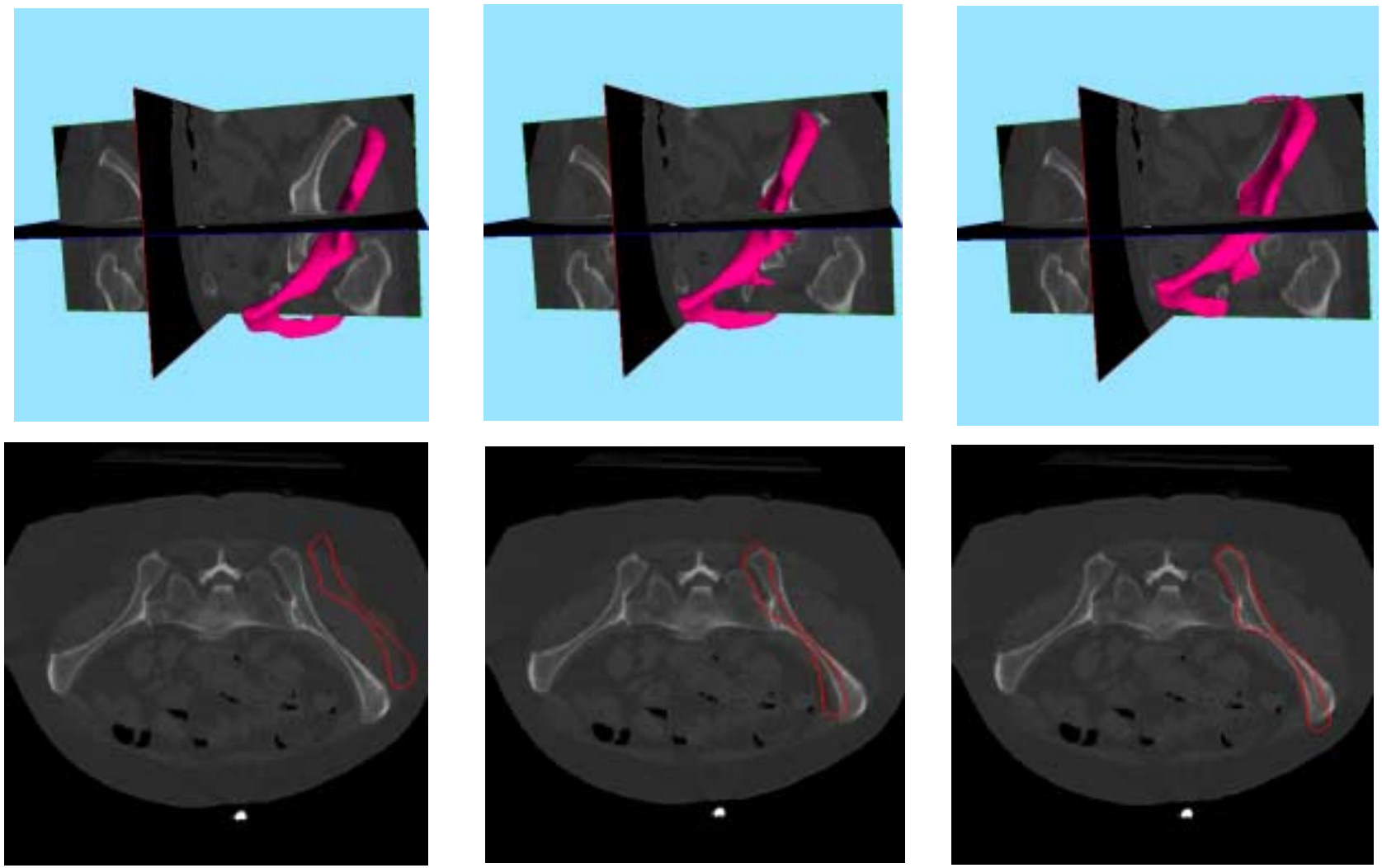
Multiple-layer Flexible Mesh Template

- Each vertex on the model defines a mesh template
- Template is in the form

$$(0, Sphere\ (v_1^{(1)} - v^{(0)}, r_1),$$
$$Sphere\ (v_2^{(1)} - v^{(0)}, r_1), \cdots,$$
$$Sphere\ (v_1^{(2)} - v^{(0)}, r_2),$$
$$Sphere\ (v_1^{(2)} - v^{(0)}, r_2), \cdots)$$



Results (Affine Transformation)



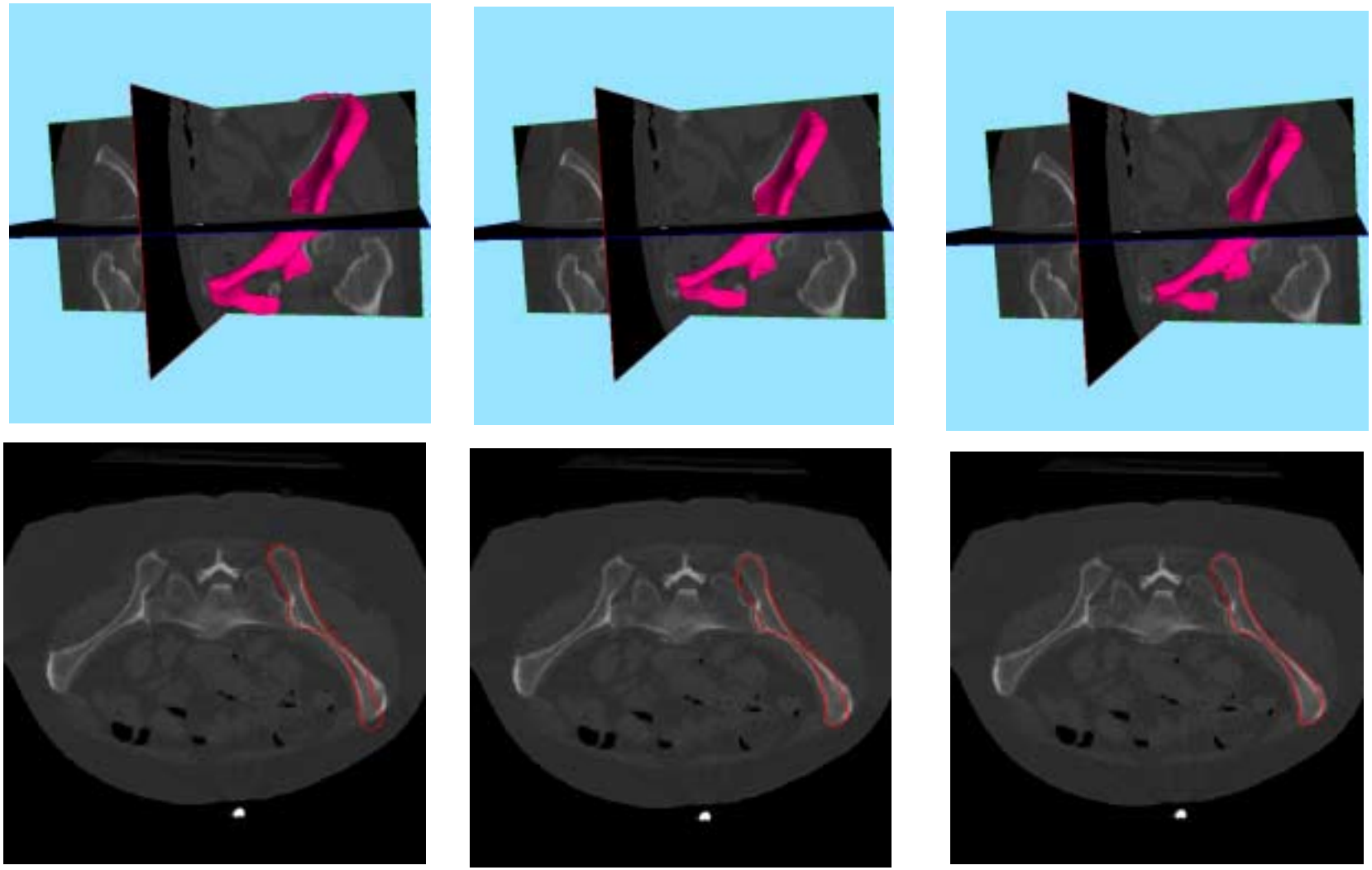
Initial

Intermediate

Final

Jianhua Yao

Results (Global Deformation)



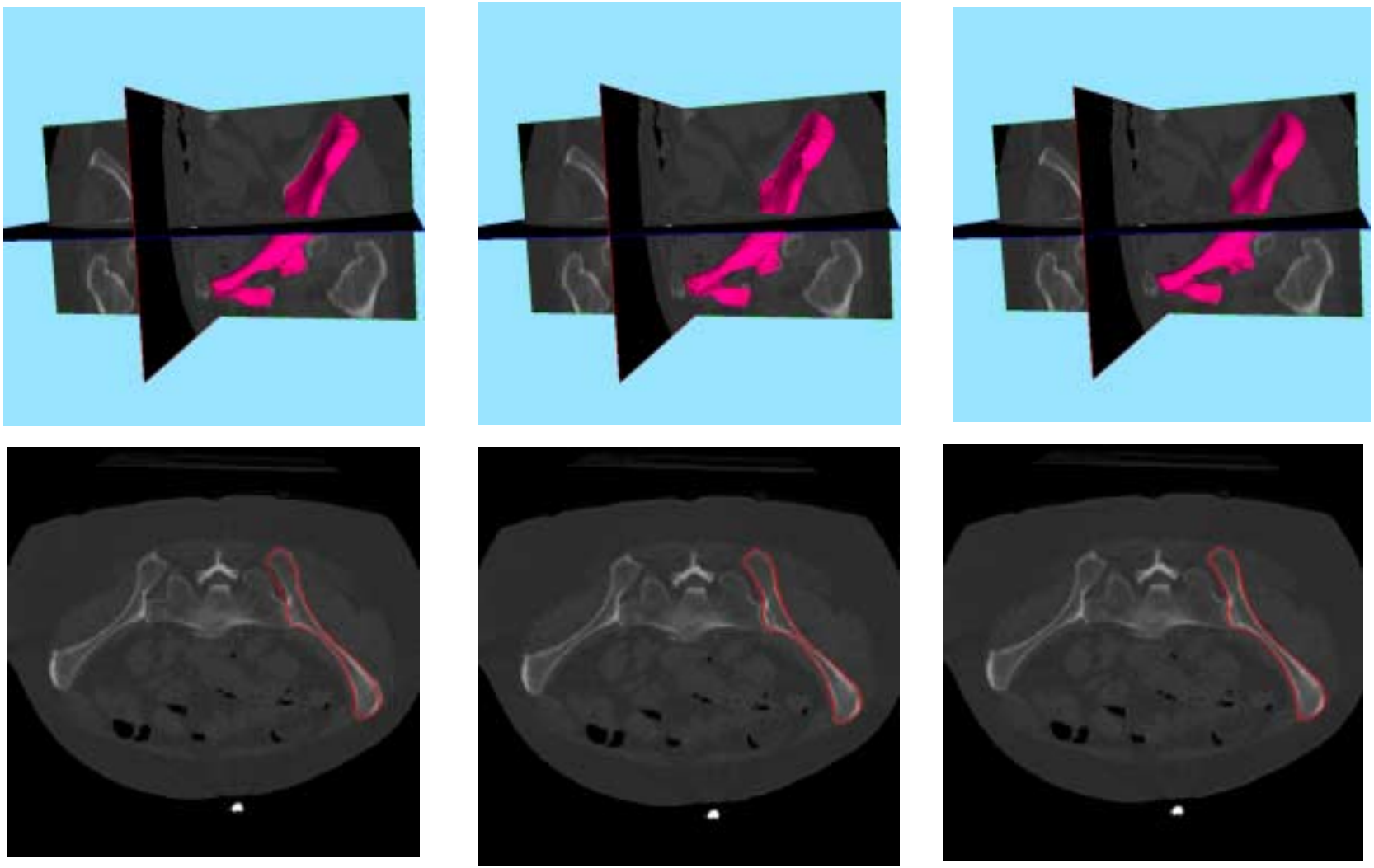
Initial

Intermediate

Final

Jianhua Yao

Results (Local Deformation)

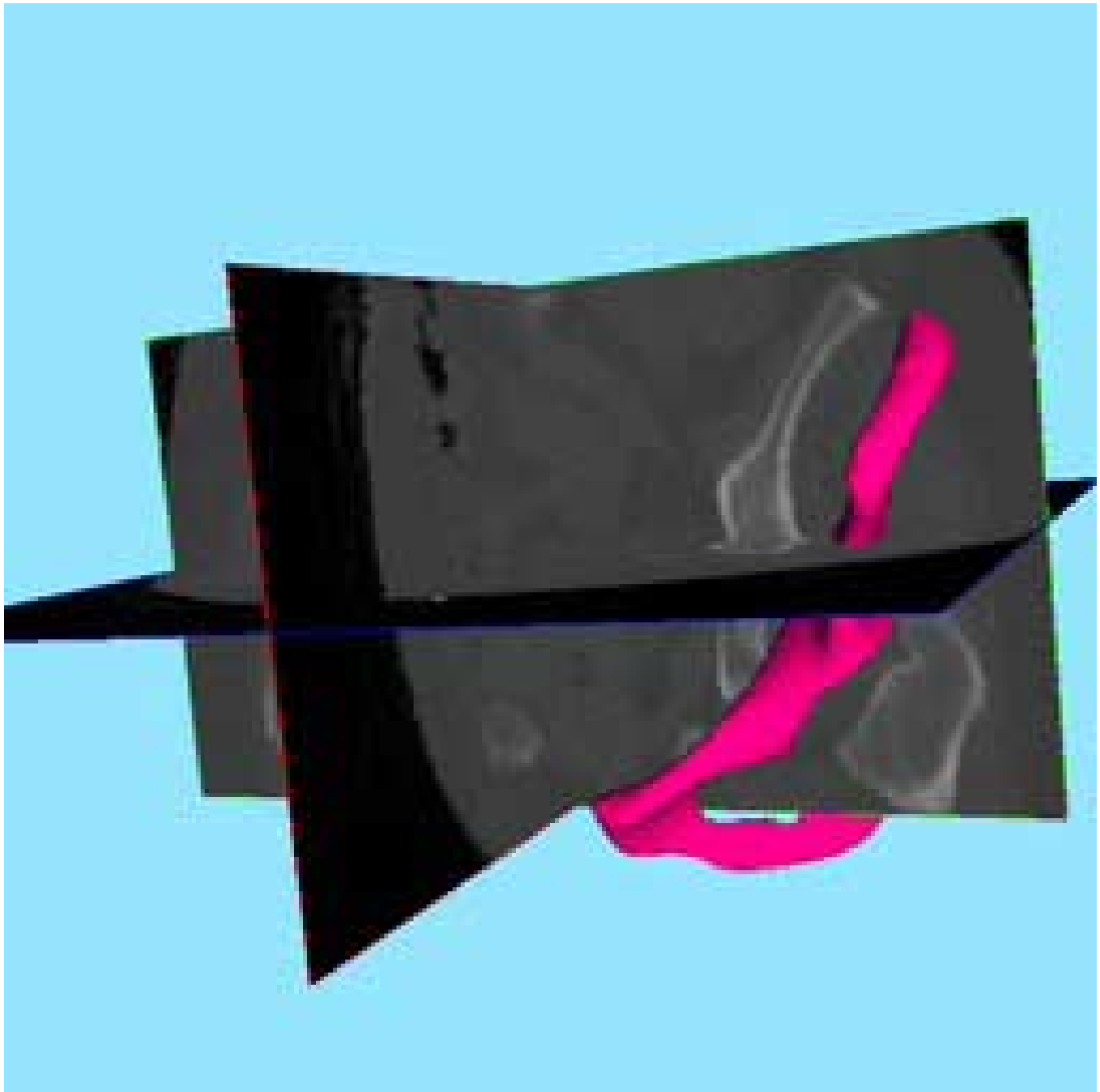


Initial

Intermediate

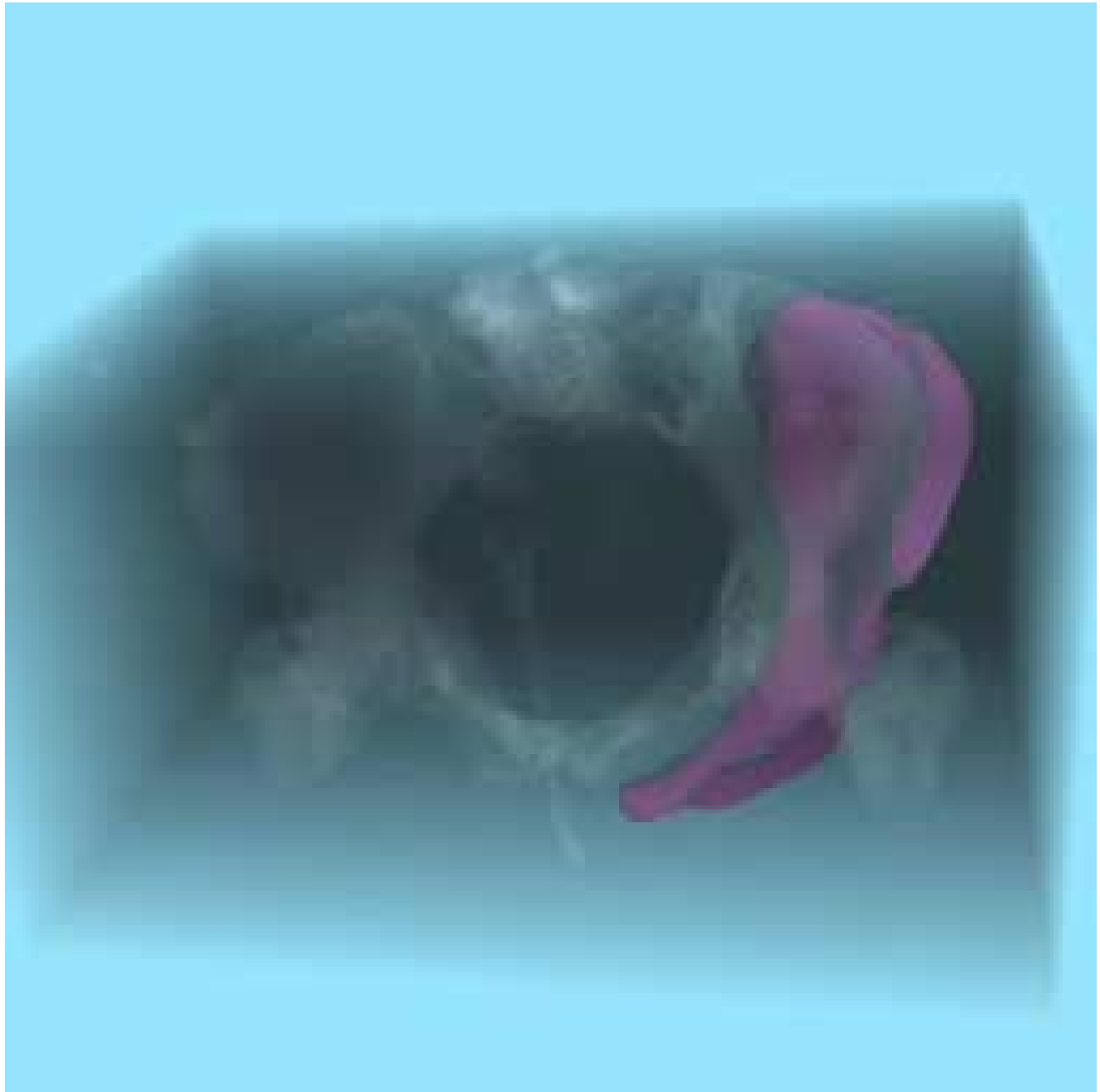
Final

Jianhua Yao



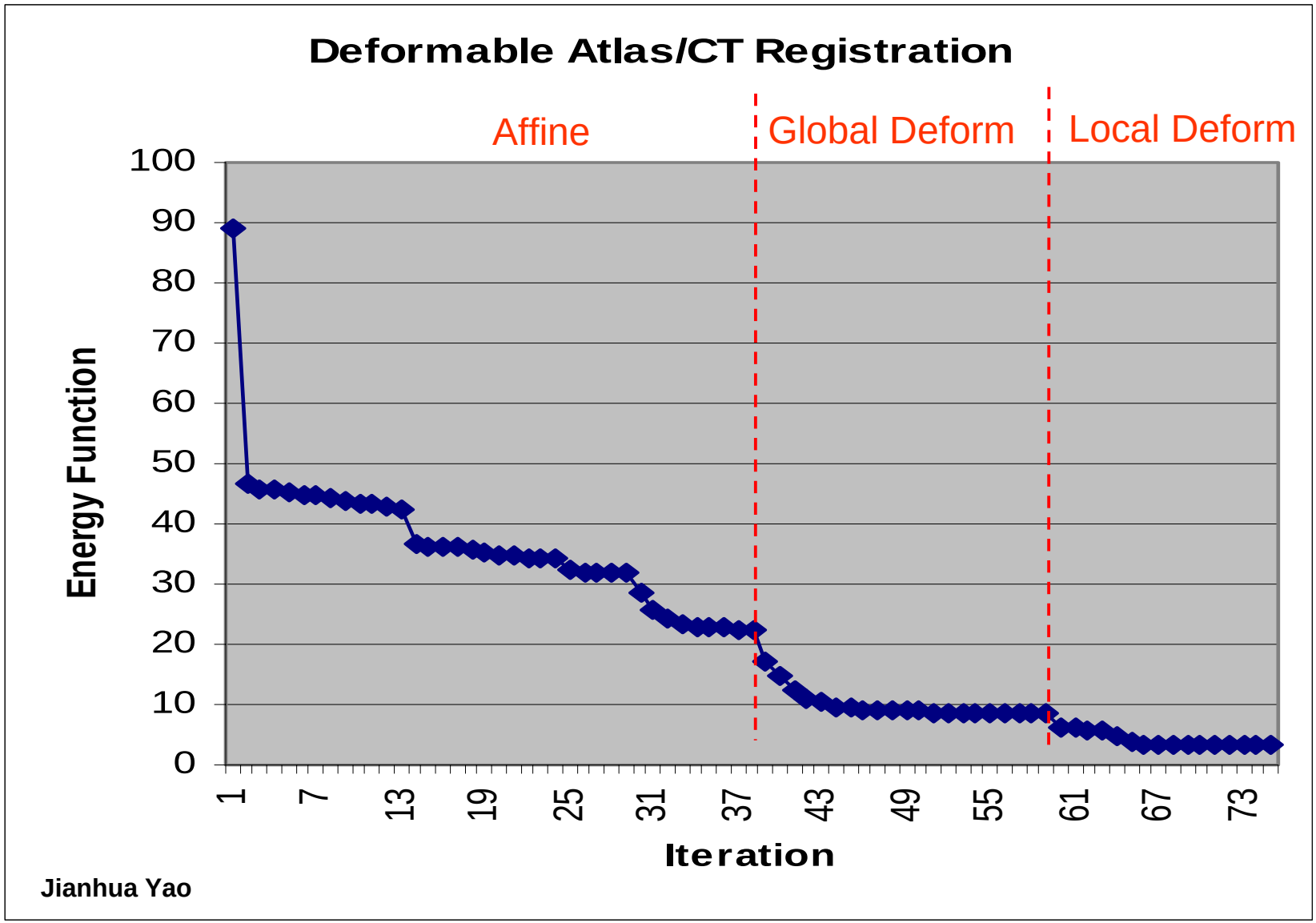
Jianhua Yao

Jianhua Yao

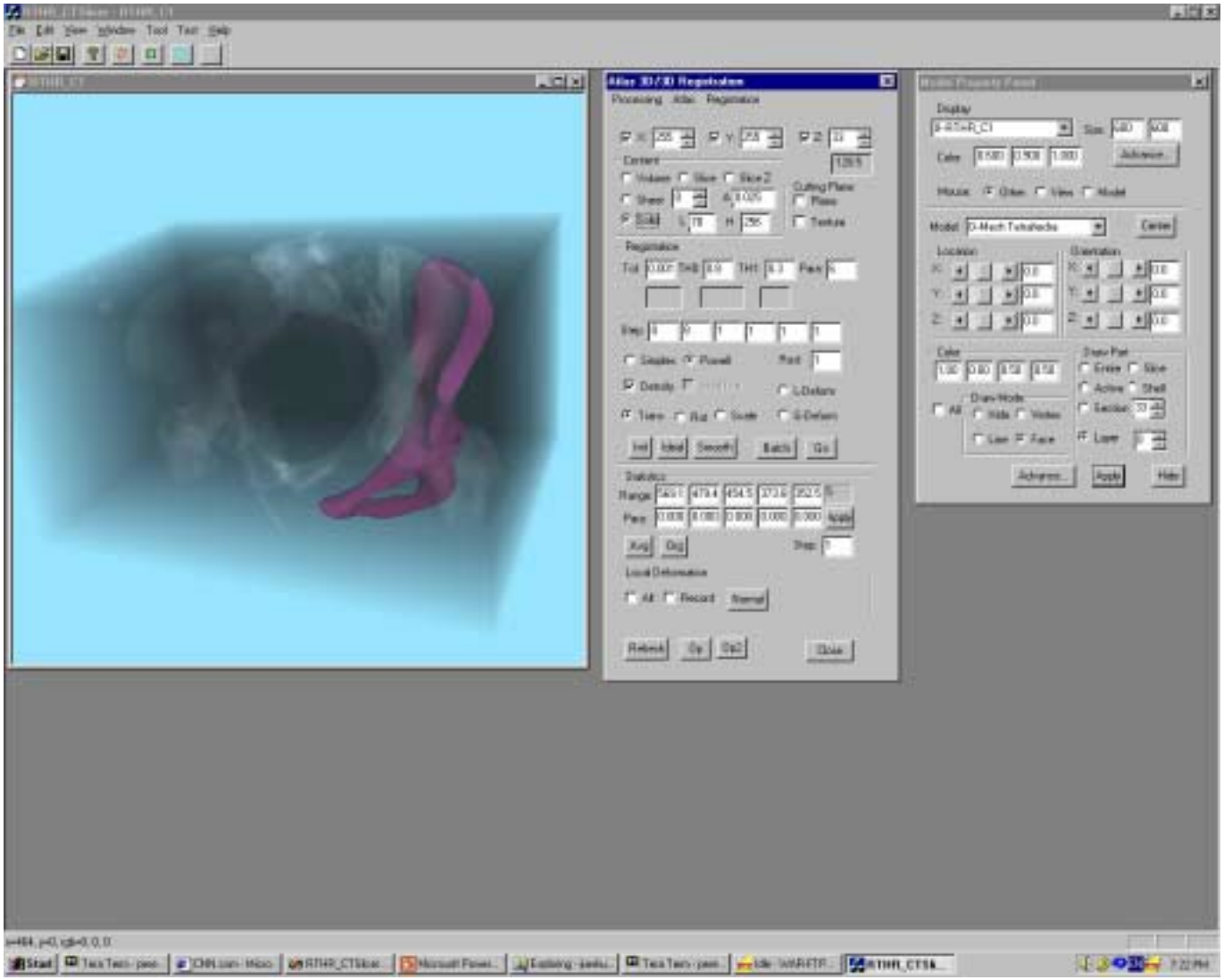


Jianhua Yao

Results (Deformable Registration)



Toolkits



Jianhua Yao

Deformable registration between density atlas and a set of 2D X-Rays

- Goal: Register and Deform the statistical density atlas to match intraoperative x-rays
- Significance:
  - Build virtual patient specific CT without real patient CT
  - Register pre-operative models and intra-operative images
  - Map predefined surgical procedure and anatomical landmarks into intra-operative images

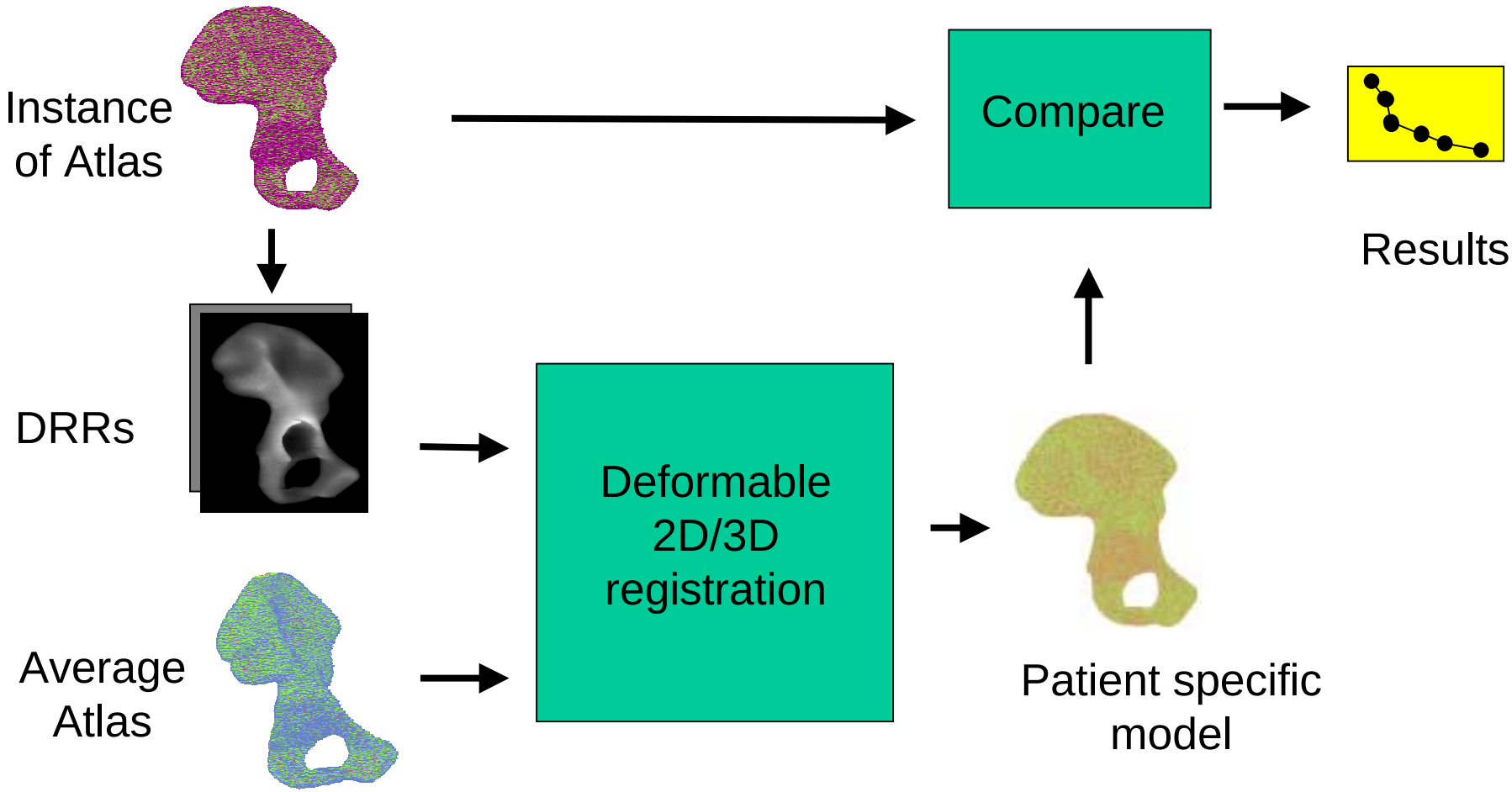
2D/3D Registration Scheme

- Solve a Powell optimization problem to minimize the cost function between DRRs of model and x-rays

$$\arg\min_{\Theta} \sum_{i=1}^{n} E(DRR_i(\Theta(mdl)), img_i)$$

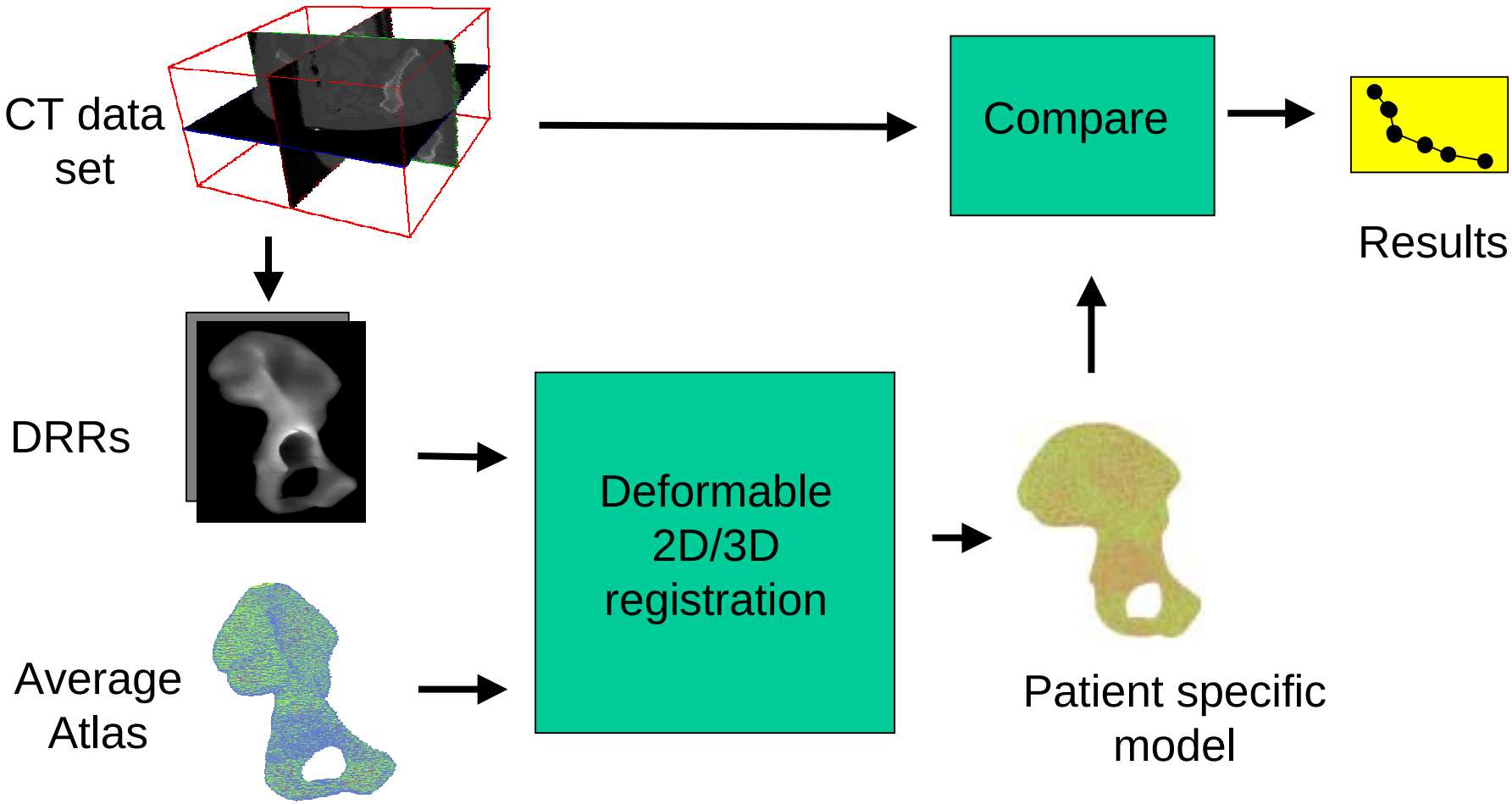
- Multiple resolutions and multiple step sizes
- E is the cost function
- Θ is the transformation of model
  - Affine Transformation
    - Translation T=(t_x, t_y, t_z)
    - Rotation R=(r_x, r_y, r_z)
    - Scale S=(s_x, s_y, s_z)
  - Global Deformation
    - Statistical deformation mode (M_i)

2D/3D Registration Experiment 1



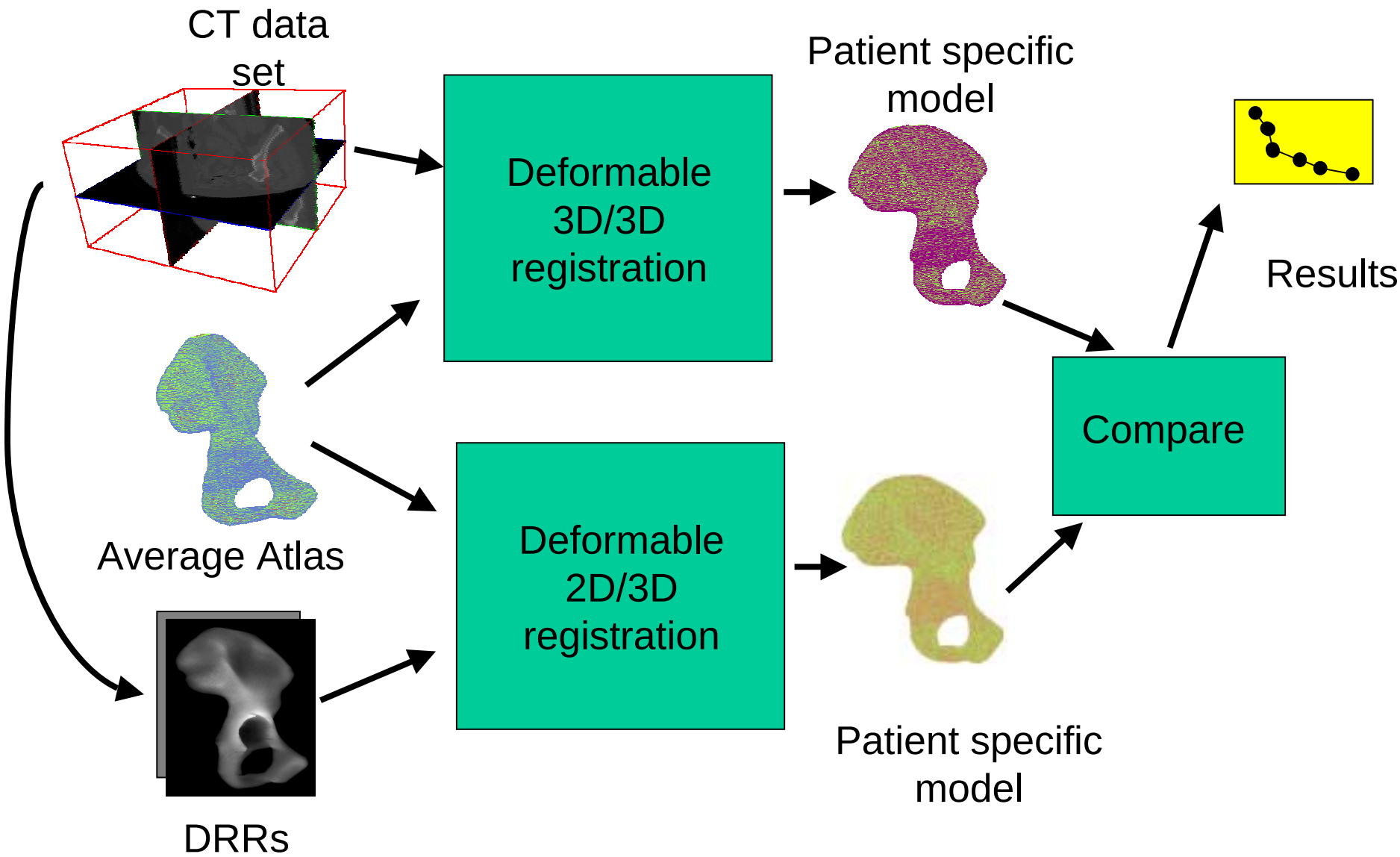
Jianhua Yao

2D/3D Registration Experiment 2



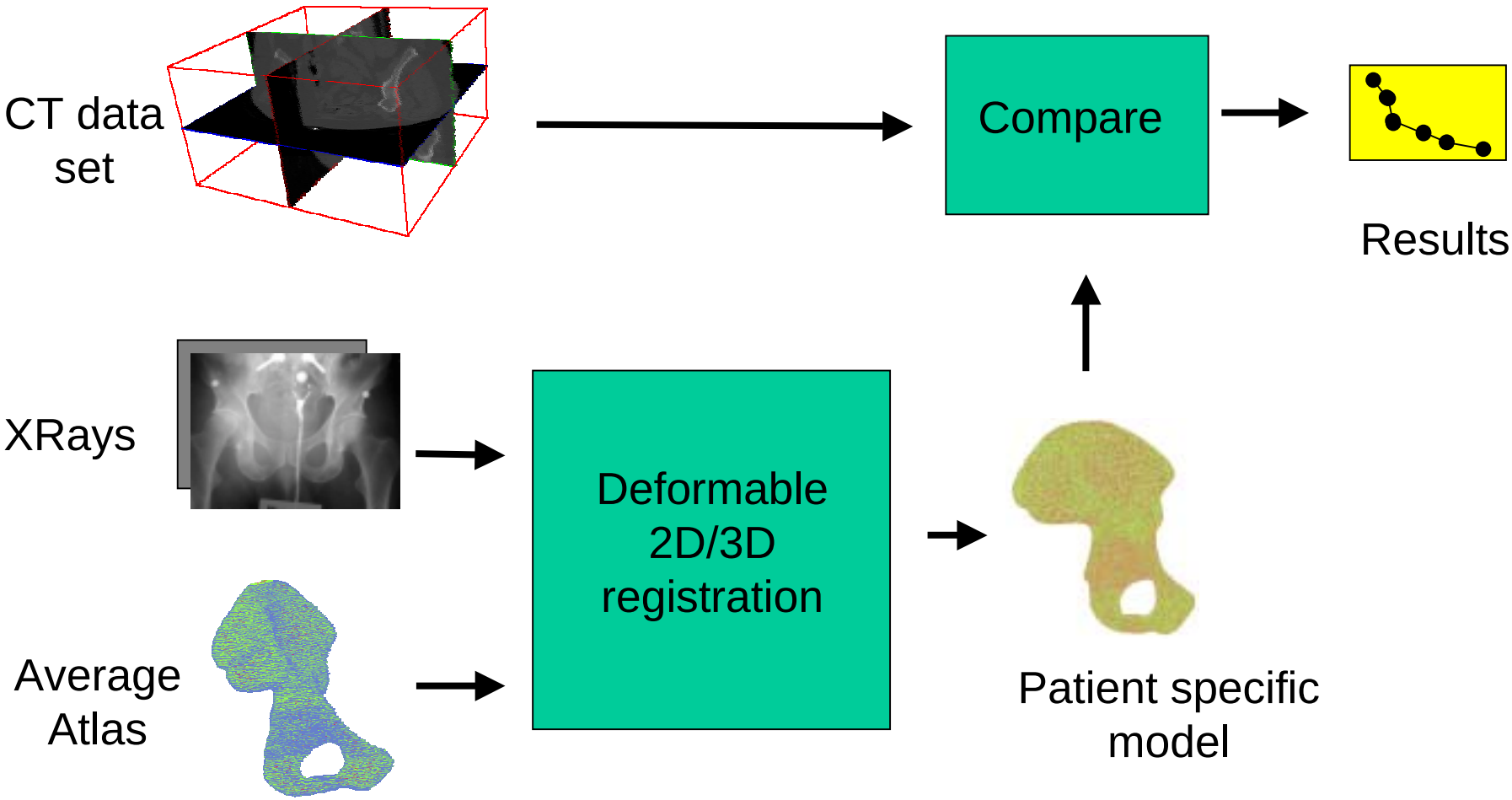
Jianhua Yao

2D/3D Registration Experiment 3



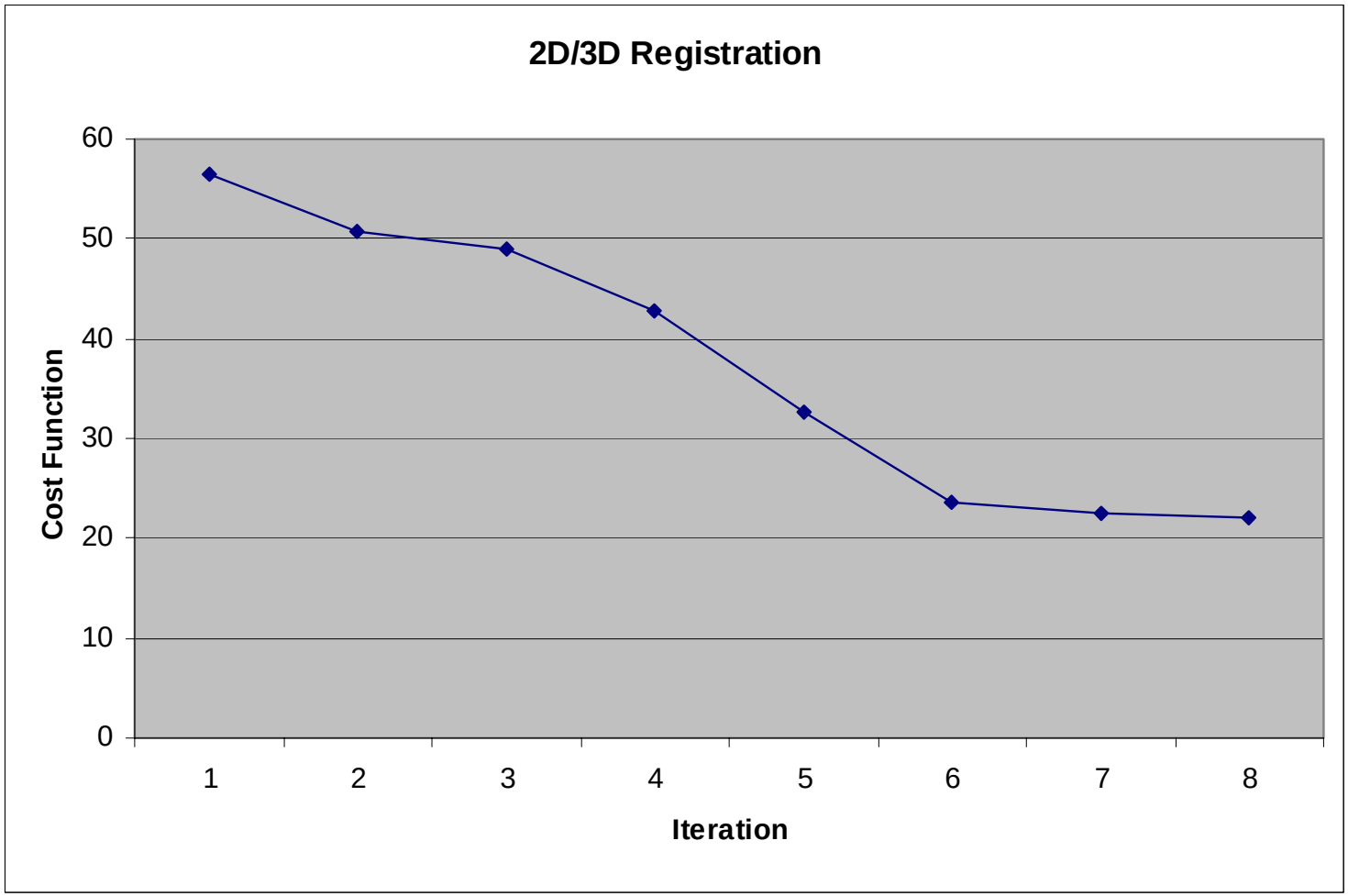
Jianhua Yao

2D/3D Registration Experiment 4

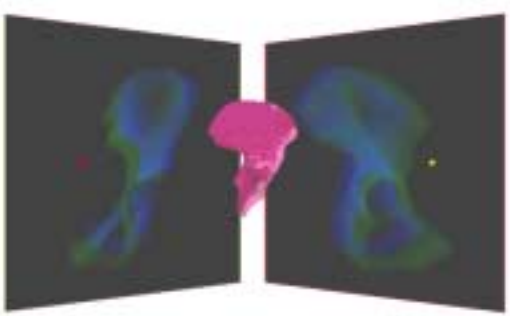


Jianhua Yao

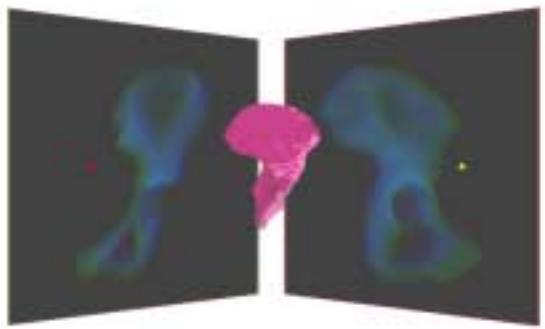
Results on Simulated Data



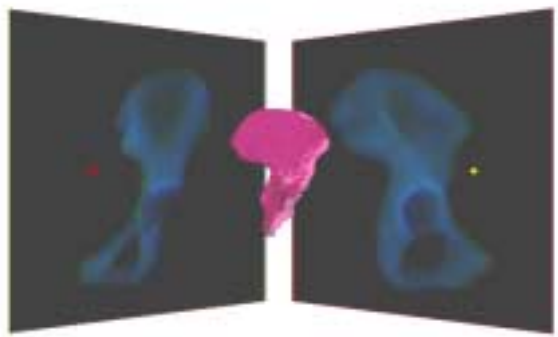
Visual Results



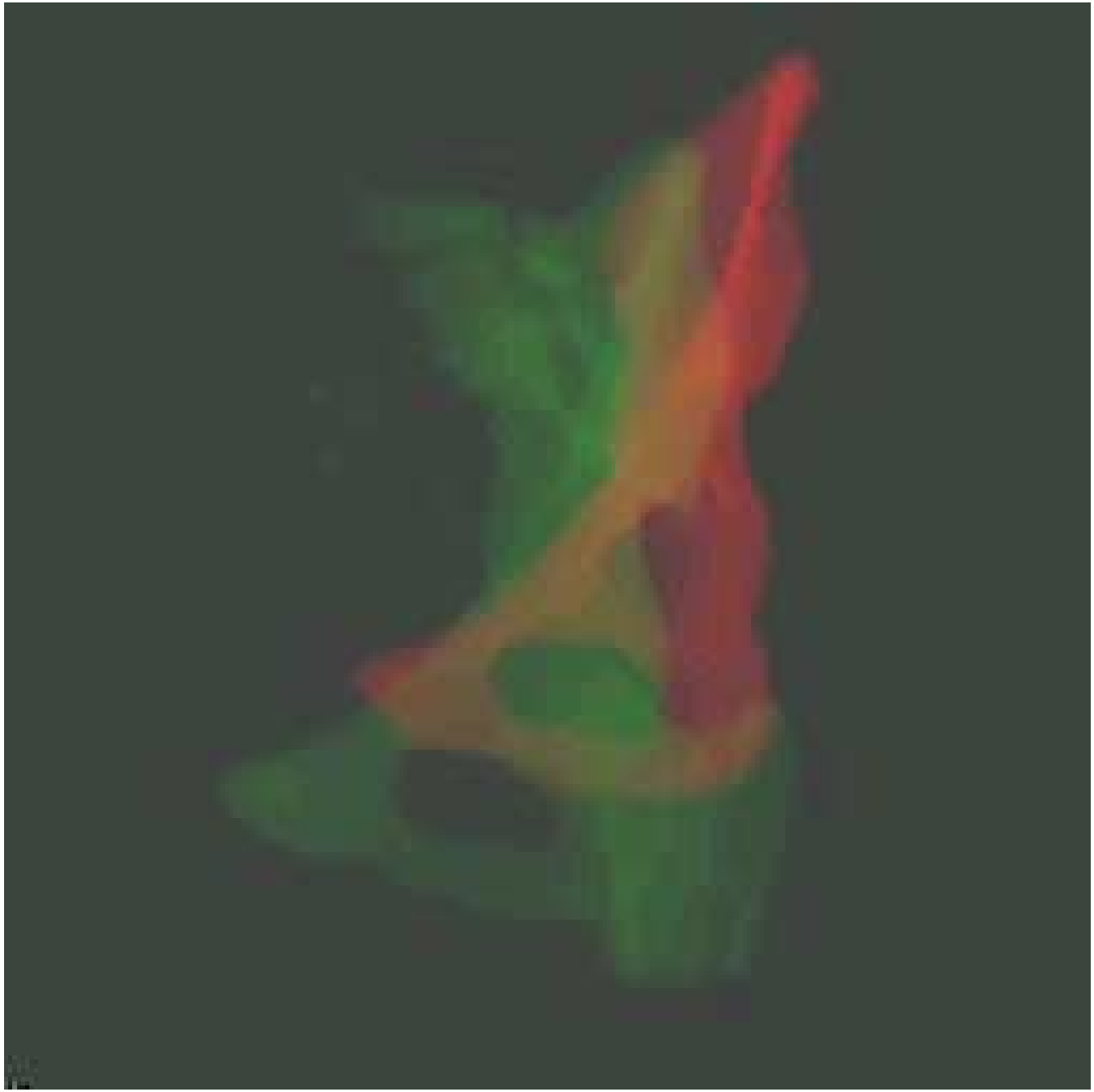
Initial



Intermediate



Final



Jianhua Yao

Future work

- Enlarge atlas
  - More patients
  - More parts of body (knee, spine, …)
- Registration & segmentation:
  - Atlas-driven reconstruction
  - Registration to incomplete data
  - Fractures & abnormal anatomy
- Enriched information
  - Biomechanics and biomedical
  - Surgical information