NSF Engineering Research Center for Computer Integrated Surgical Systems and Technology

LABORATORY FOR Computational Sensing + Robotics THE JOHNS HOPKINS UNIVERSITY

Registration – Part 1

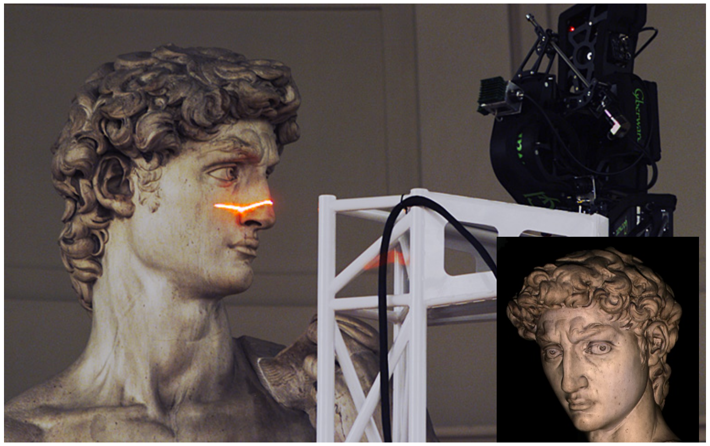
600.455/655 Computer Integrated Surgery

100 YEARS JOHNS HOPKINS ENGINEERING

WHITING SCHOOL OF ENGINEERING
THE JOHNS HOPKINS UNIVERSITY

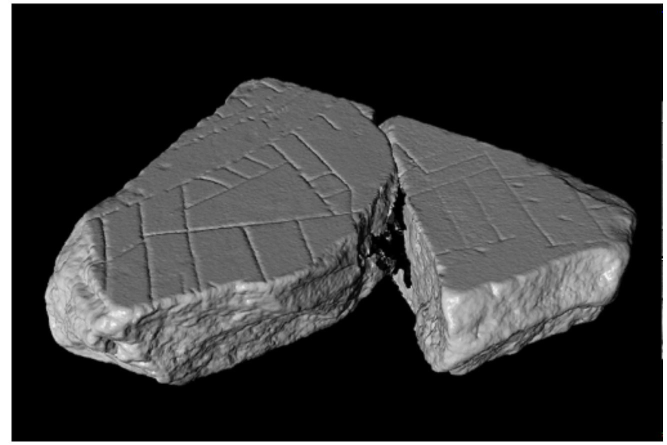
Russell H. Taylor
John C. Malone Professor of Computer Science,
with joint appointments in Mechanical Engineering, Radiology & Surgery
Director, Laboratory for Computational Sensing and Robotics
The Johns Hopkins University
rht@jhu.edu

Why is Registration Important?



Digitize important cultural artifacts

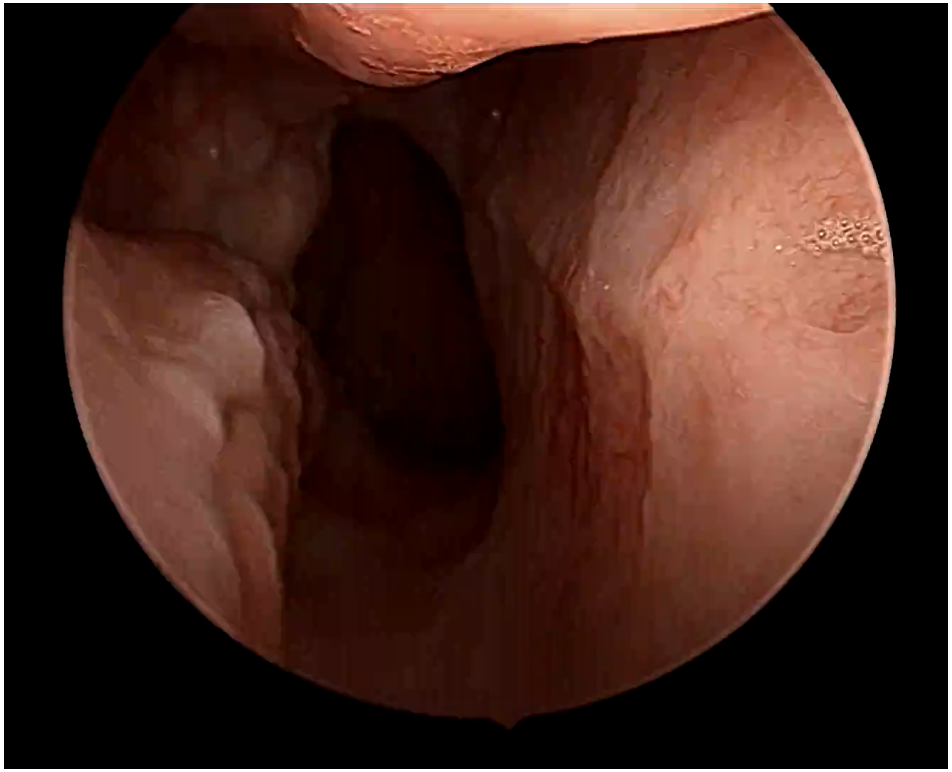
Medical interventions

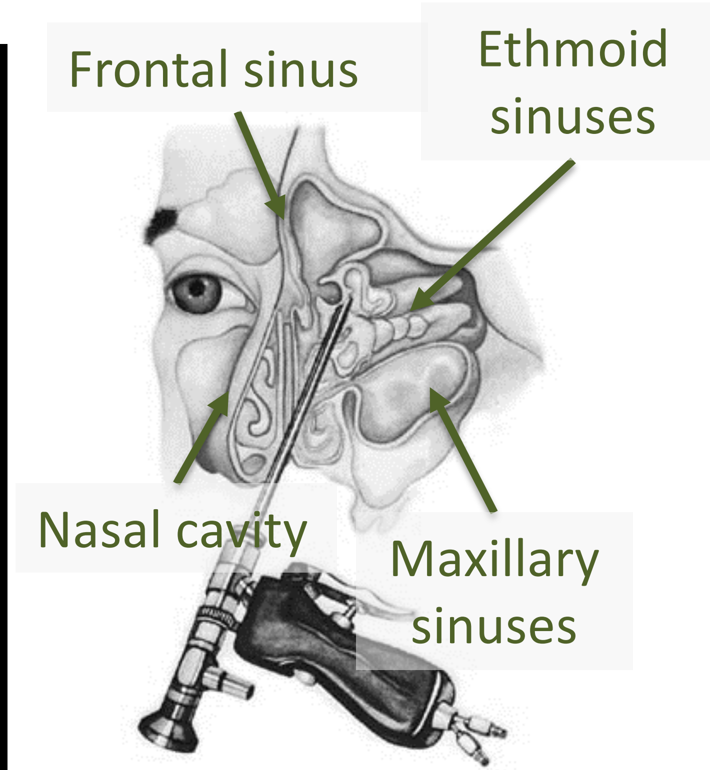


Archeology

And many more applications…

Why is Registration Important?

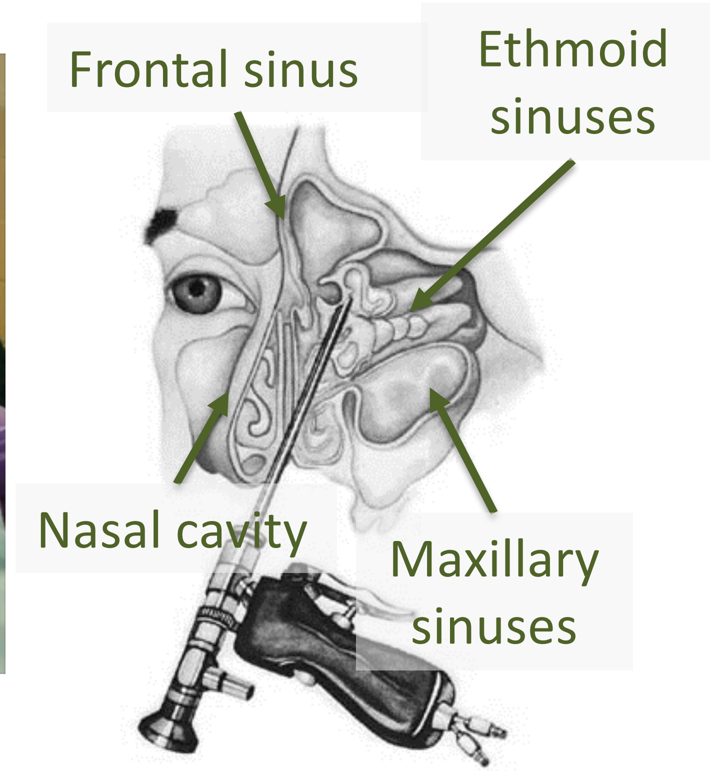




Typical Example: Sinus Endoscopy. The surgeon can only see video from the endoscope. But crucial data is in the CT about structures that cannot be seen.

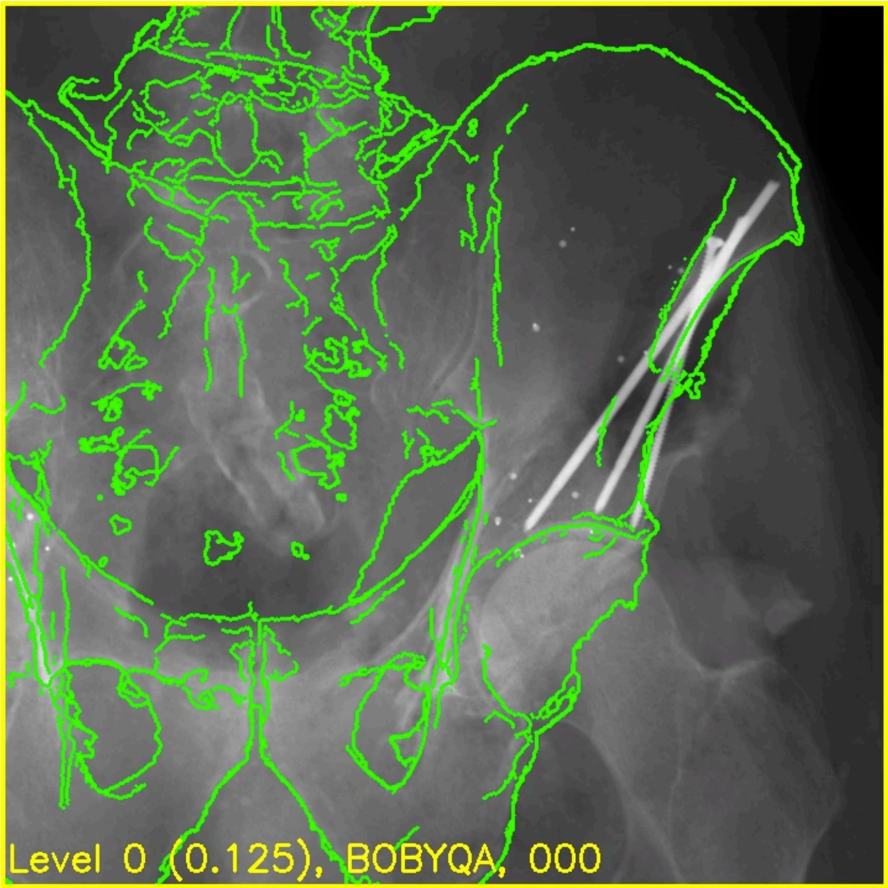
Slide Credit: Ayushi Sinha

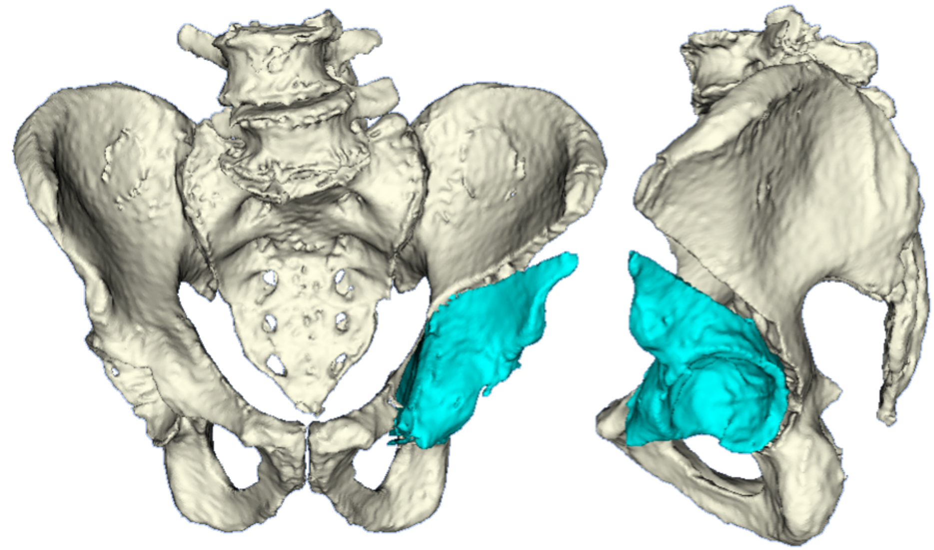
Why is Registration Important?



Typical Example: Sinus Endoscopy. After registration, the computed can create video overlays, help guide a robot, or provide other assistance.

Why is Registration Important?



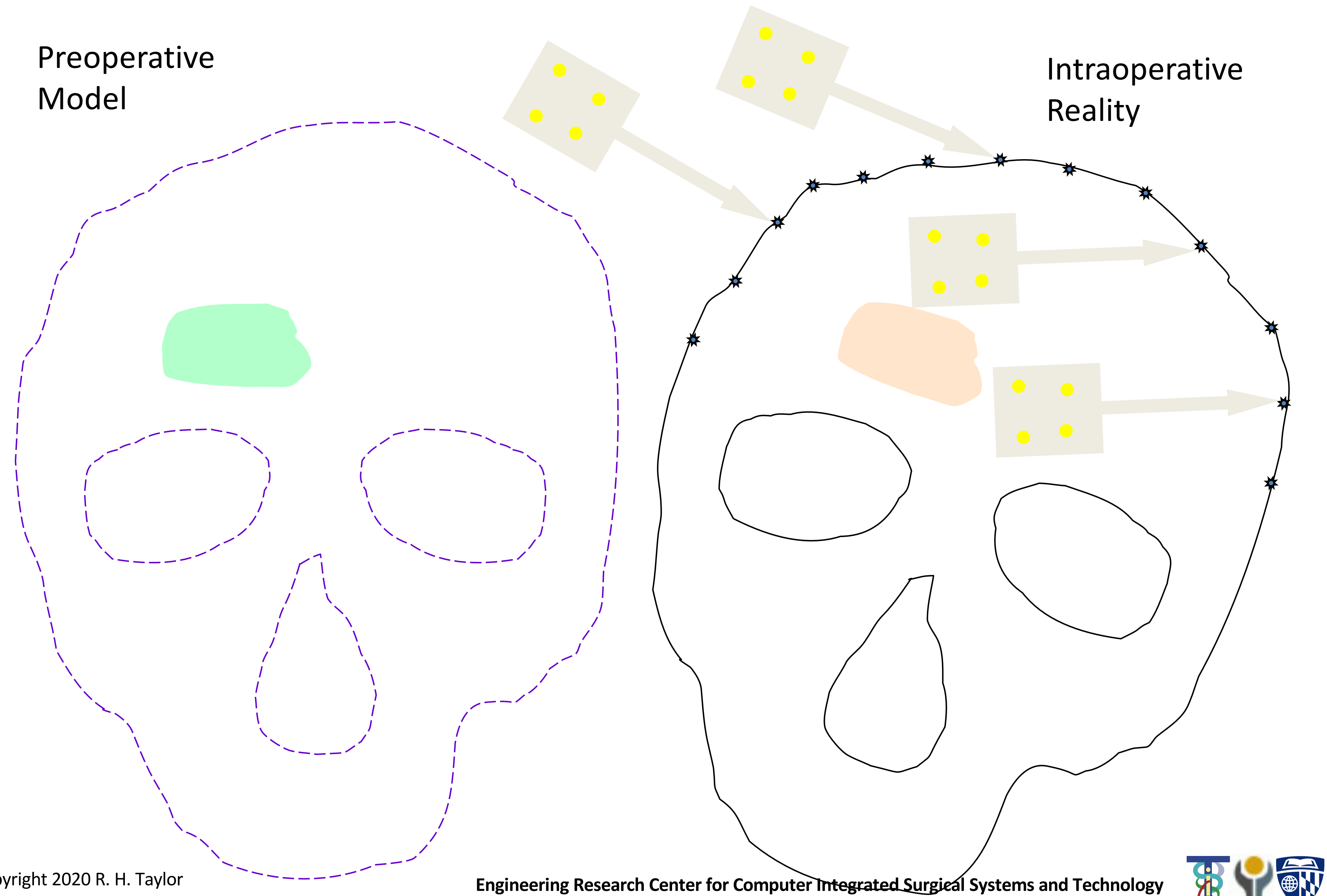


Typical Example: Osteotomies. Surgeon needs to know the position and orientation of bone fragment relative to pelvis, based on x-ray images.

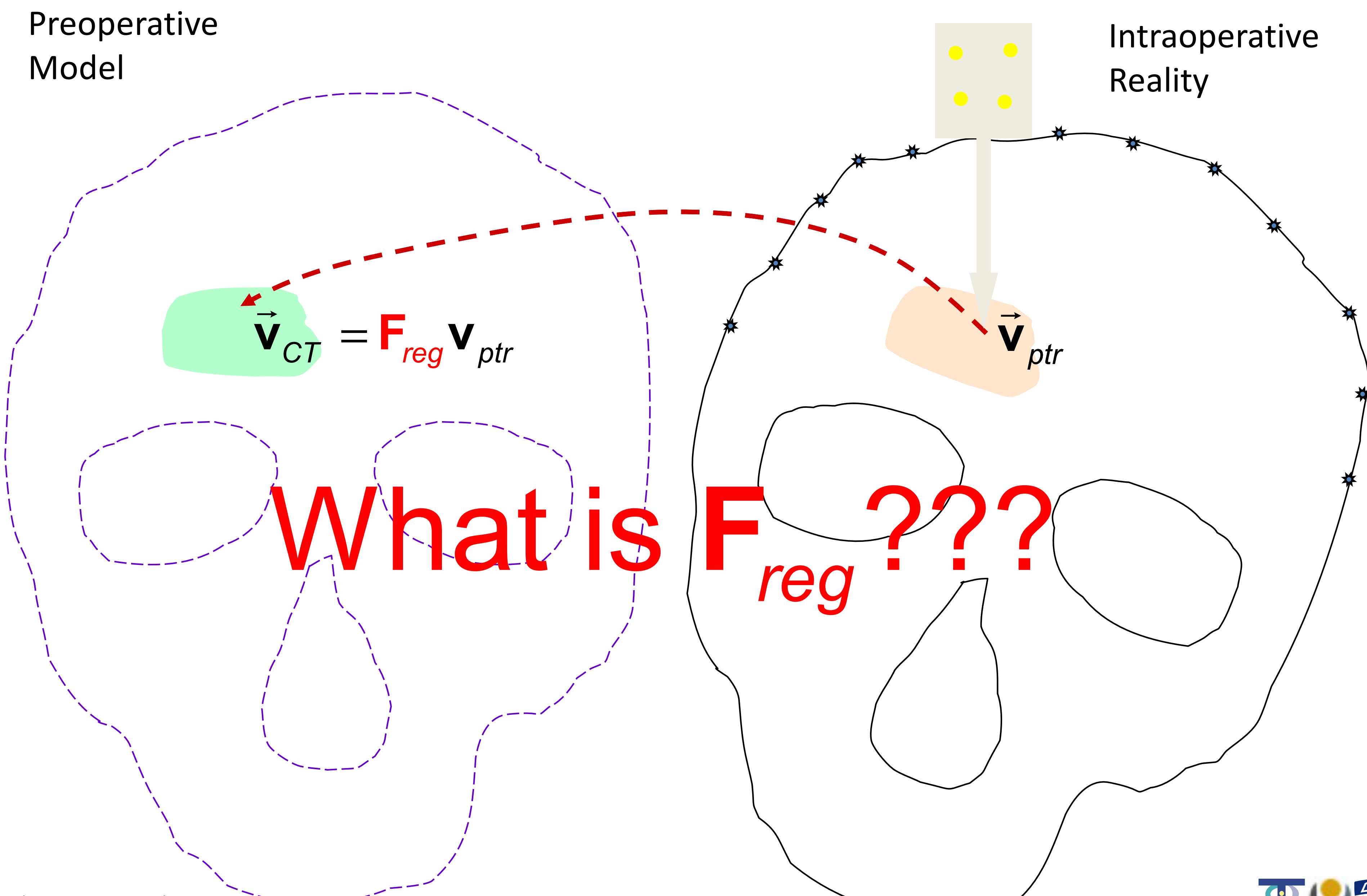
What needs registering?

- **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



A typical registration problem



Taxonomy of methods

- Feature-based
- Intensity-based

Framework for feature-based methods

- 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,

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

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:

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

Definitions

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

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

  The key assumption is that deformations may be neglected.
- **Similarity Transformation:** Essentially, rigid+scale change. Preserves angles and shape, but not size

  $$\mathbf{T}(\mathbf{x}) = s\mathbf{R} \bullet \mathbf{x} + \mathbf{p}$$
- **Elastic Transformation:** Cases where must take more general 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.

Distance 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:

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

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

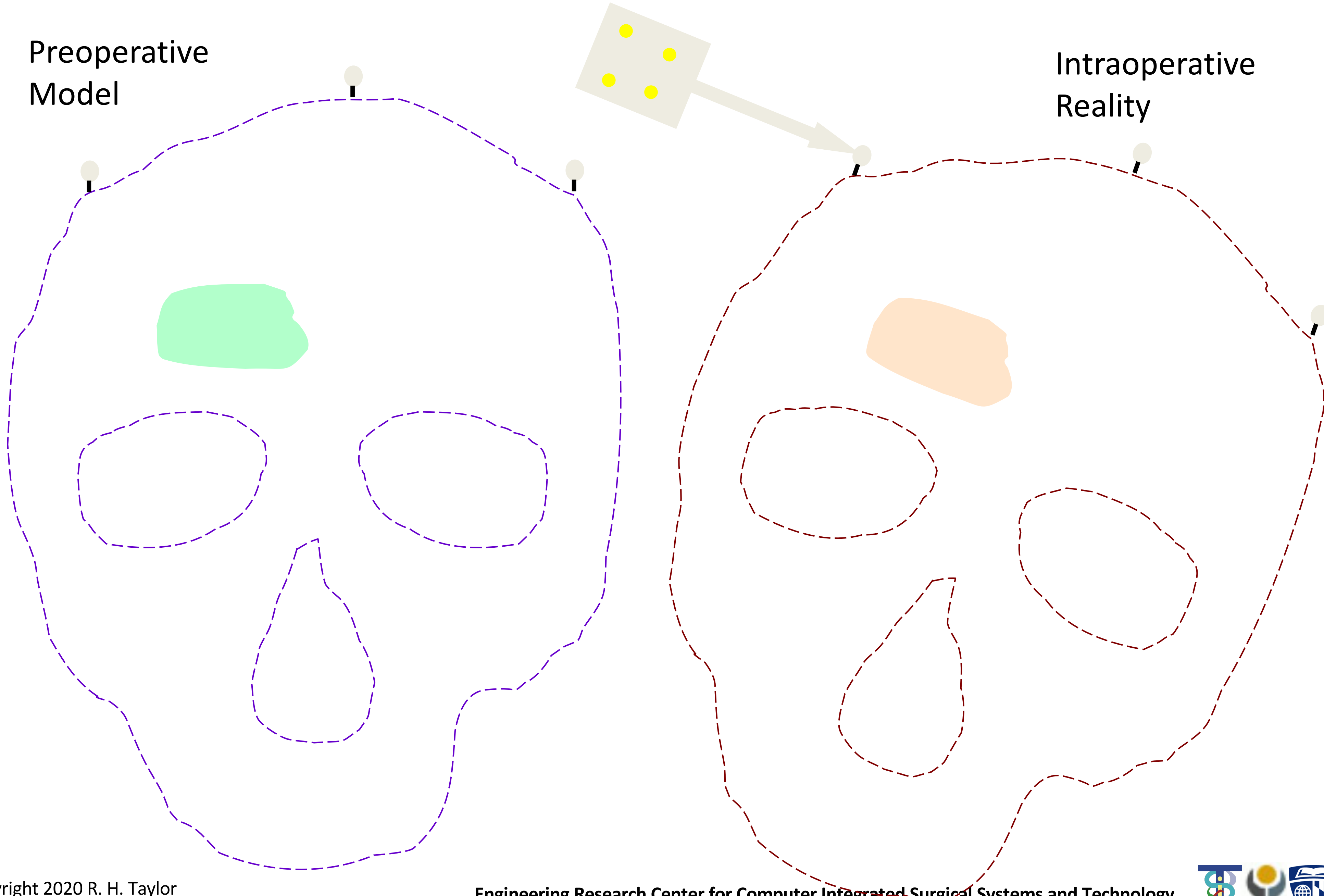
$$D = \underset{i}{\text{median}}\ distance(F_{Ai}, \mathbf{T}(F_{Bi}))$$

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

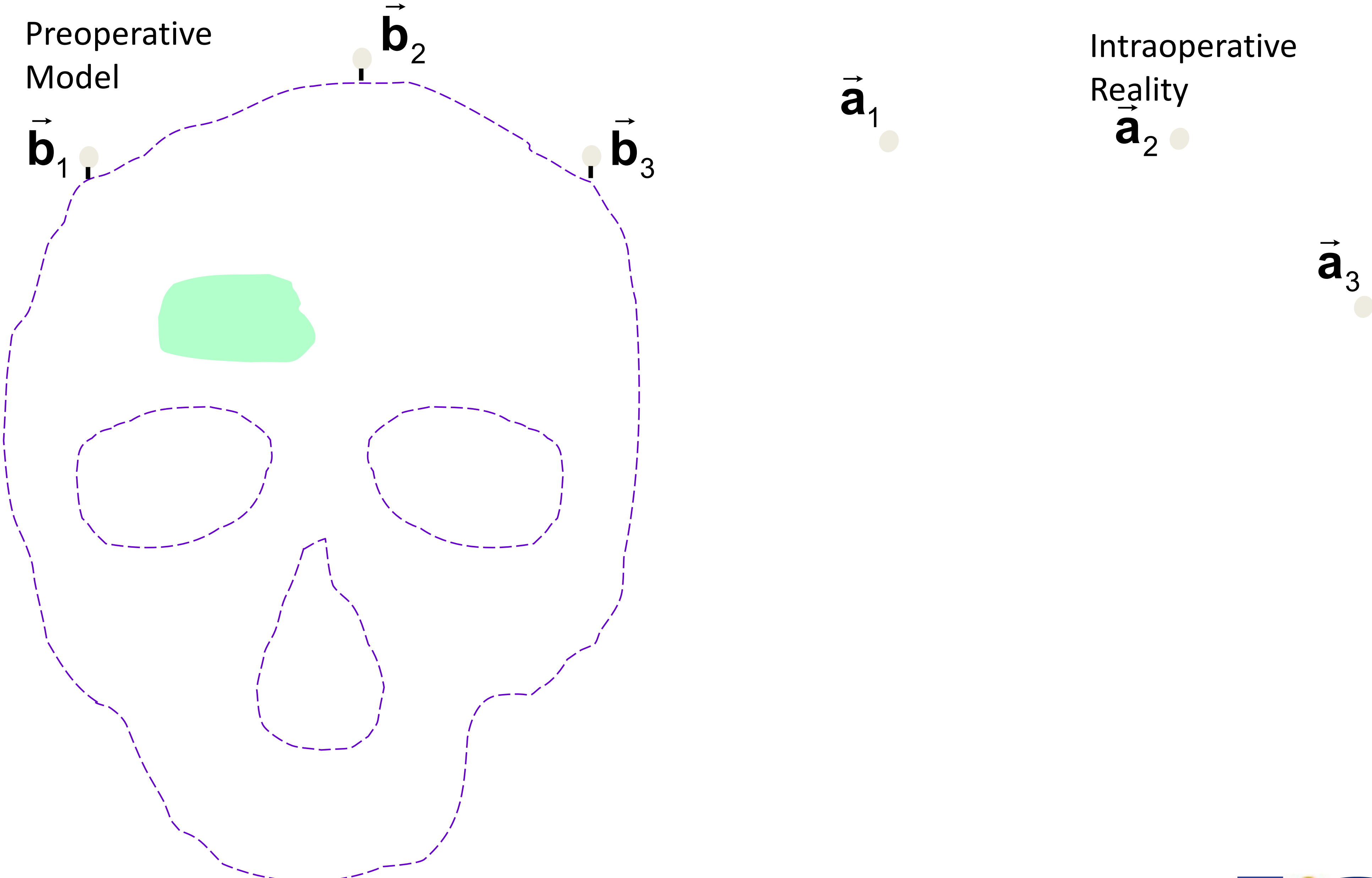
Optimization

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

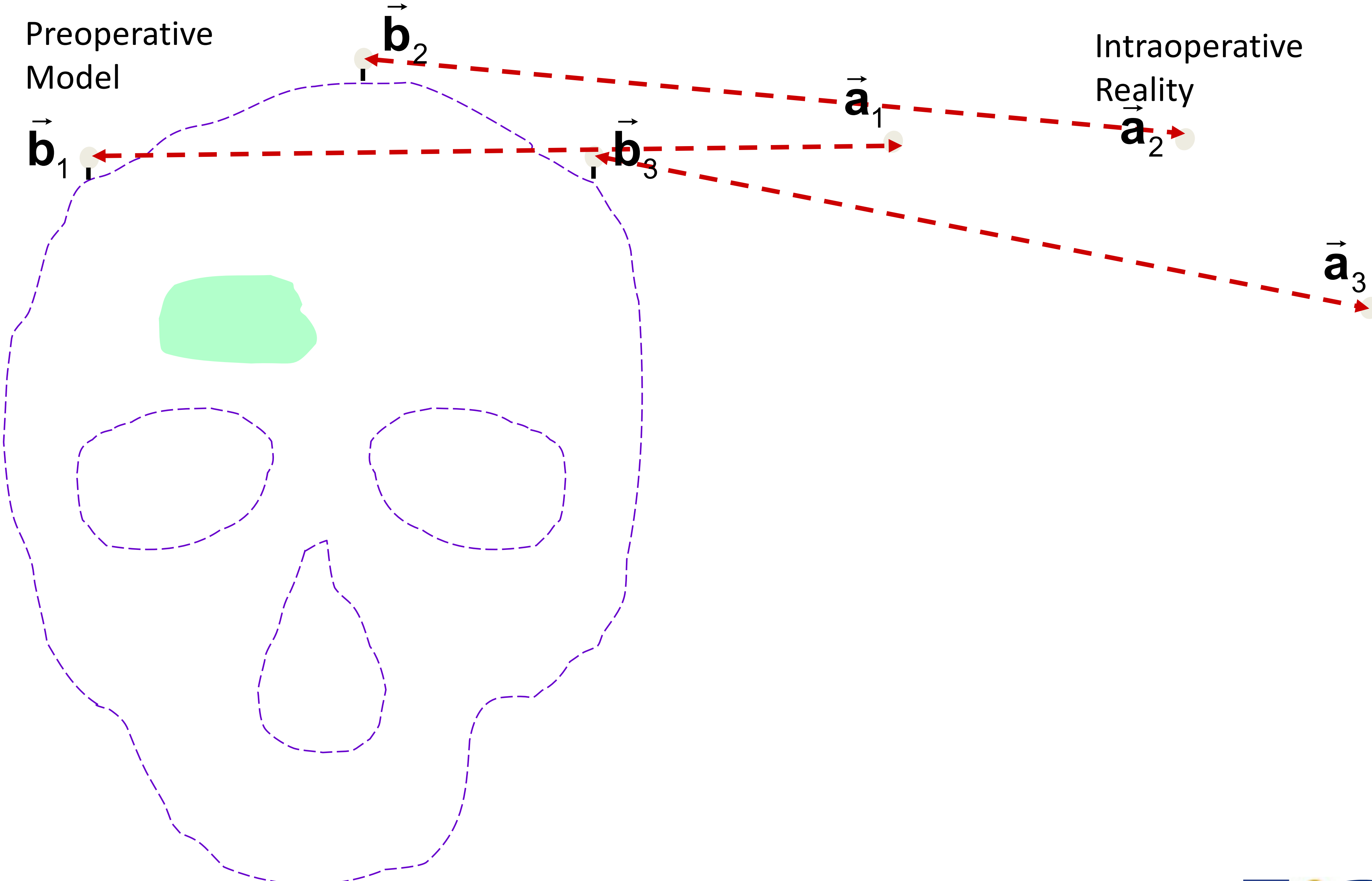
A typical fiducial-based registration problem



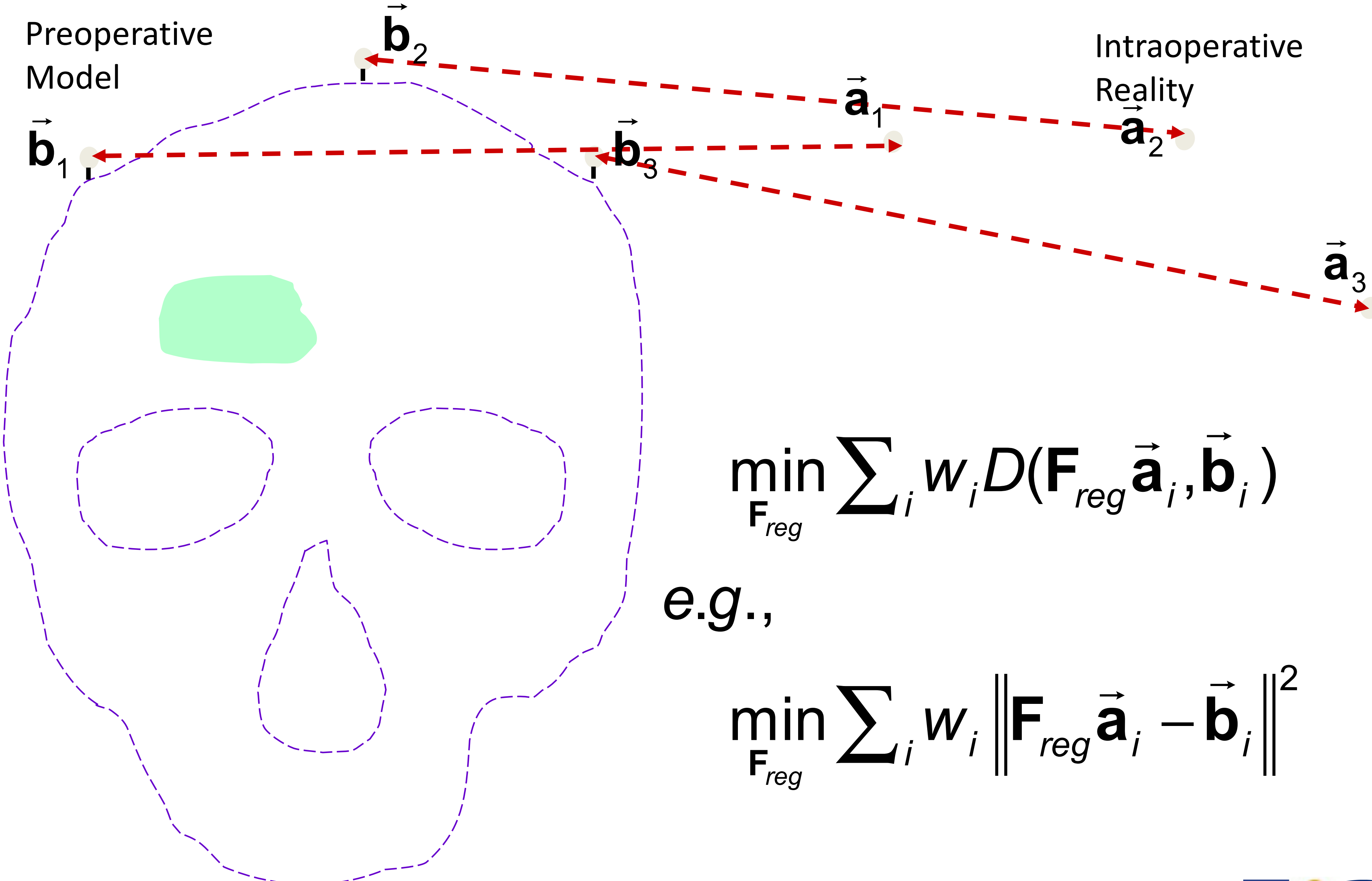
What the computer knows



Identify corresponding points



Find best rigid transformation!

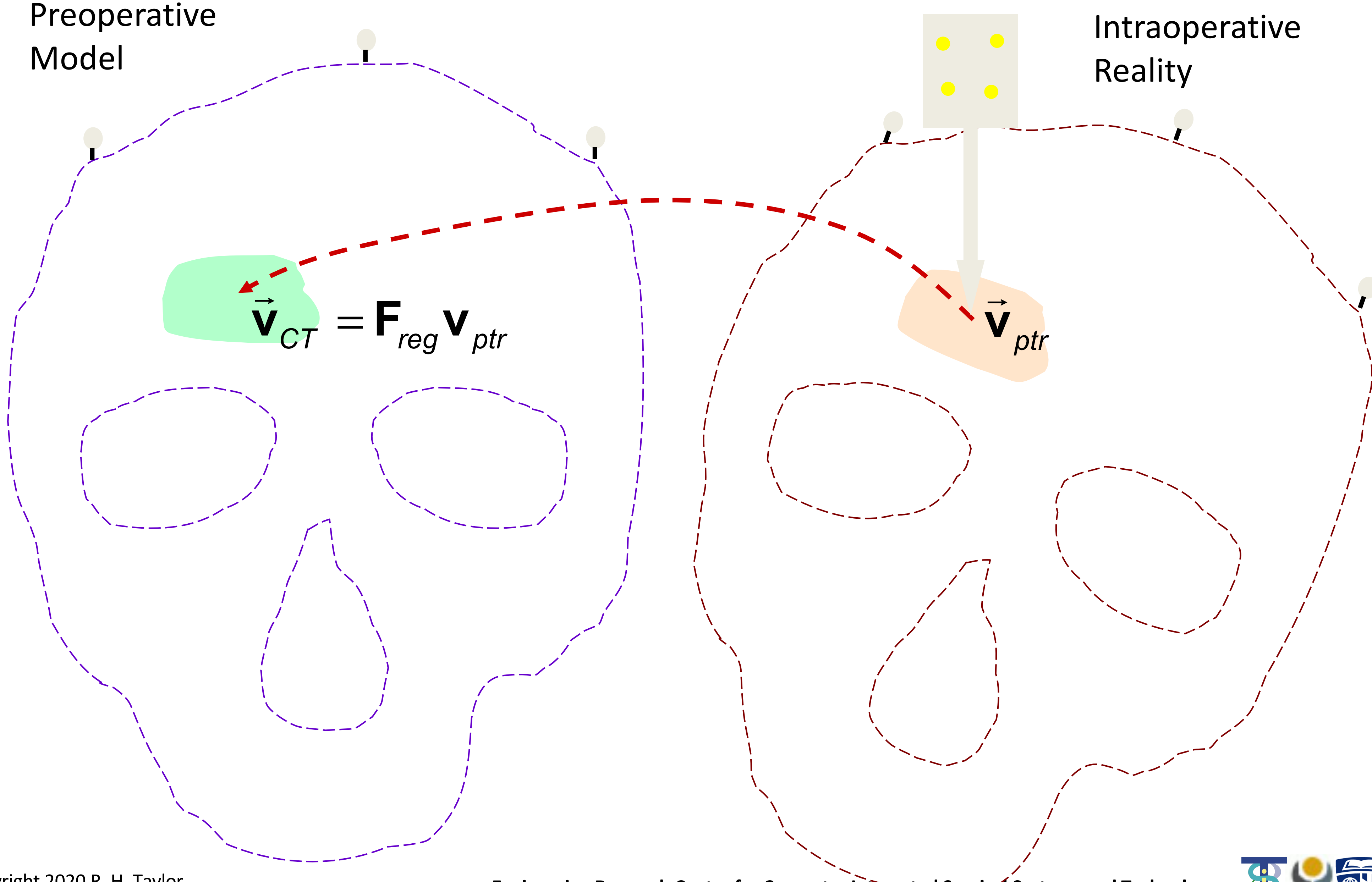


$$\min_{\mathbf{F}_{reg}} \sum_i w_i D(\mathbf{F}_{reg}\vec{\mathbf{a}}_i, \vec{\mathbf{b}}_i)$$

e.g.,

$$\min_{\mathbf{F}_{reg}} \sum_i w_i \left\| \mathbf{F}_{reg}\vec{\mathbf{a}}_i - \vec{\mathbf{b}}_i \right\|^2$$

Navigate


Preoperative Model
Intraoperative Reality
$\vec{v}_{CT} = F_{reg} v_{ptr}$
$\vec{v}_{ptr}$

Sampled 3D data to surface models

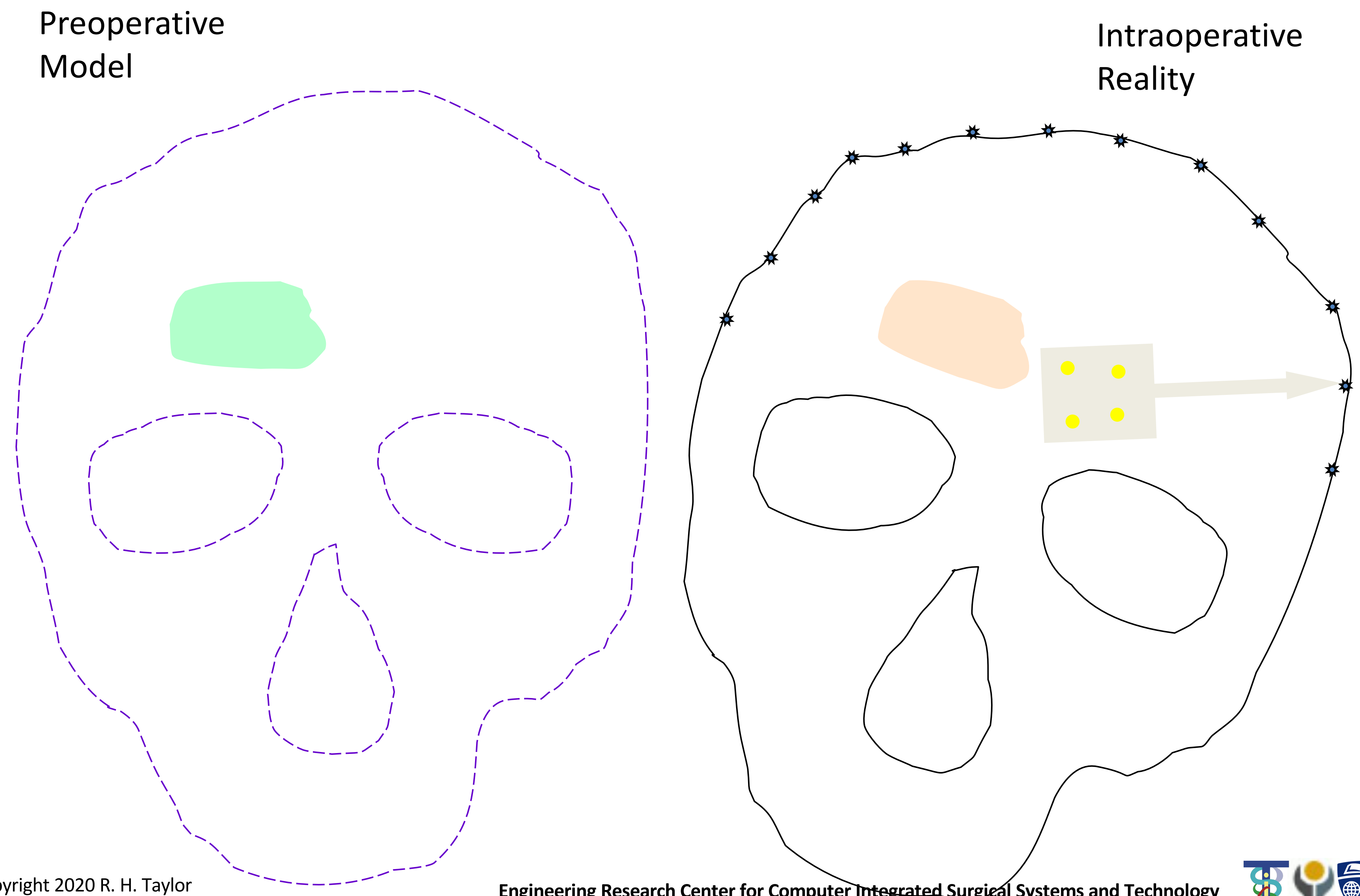
Outline:

- 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.

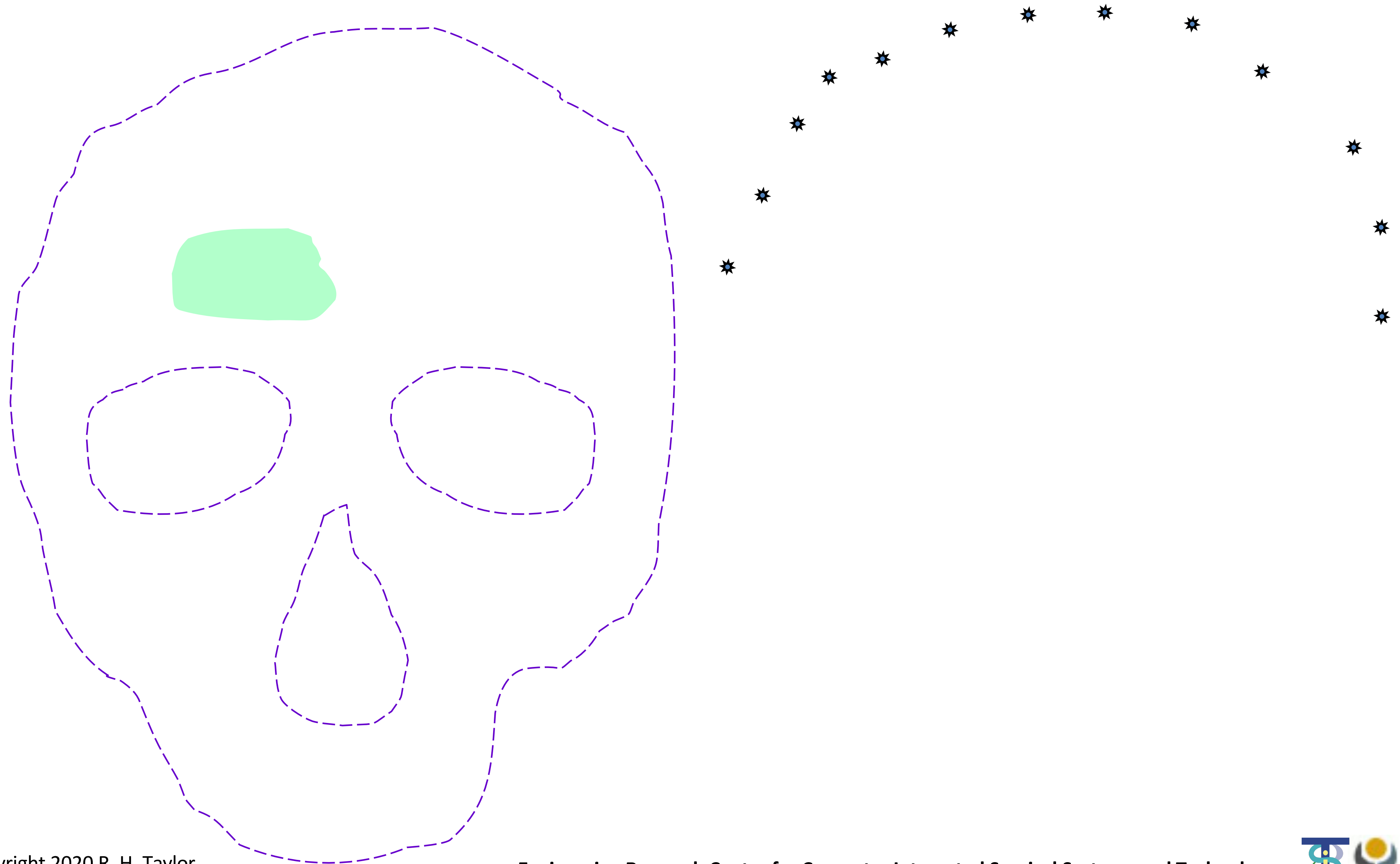
Examples

- 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]

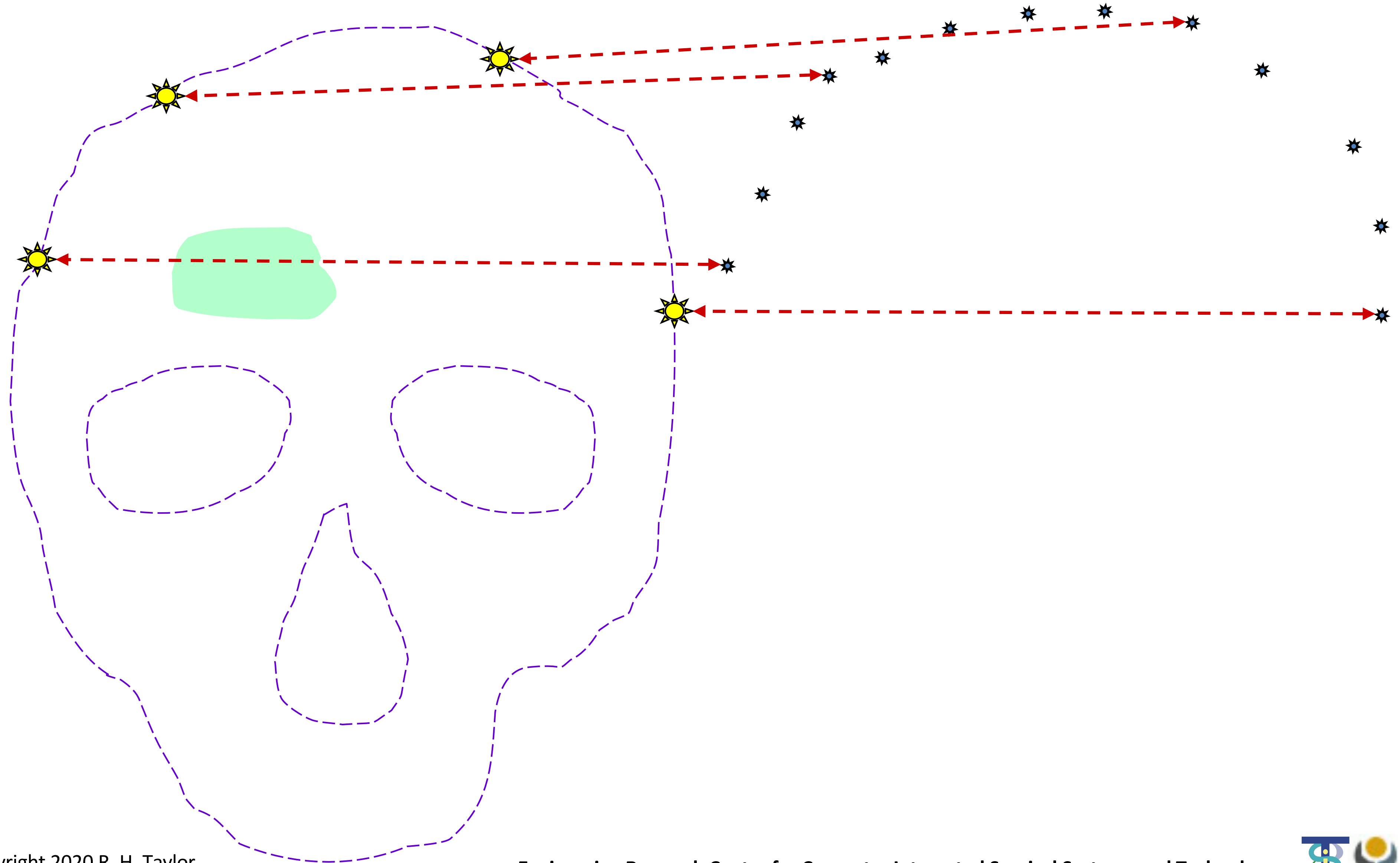
A typical surface registration problem



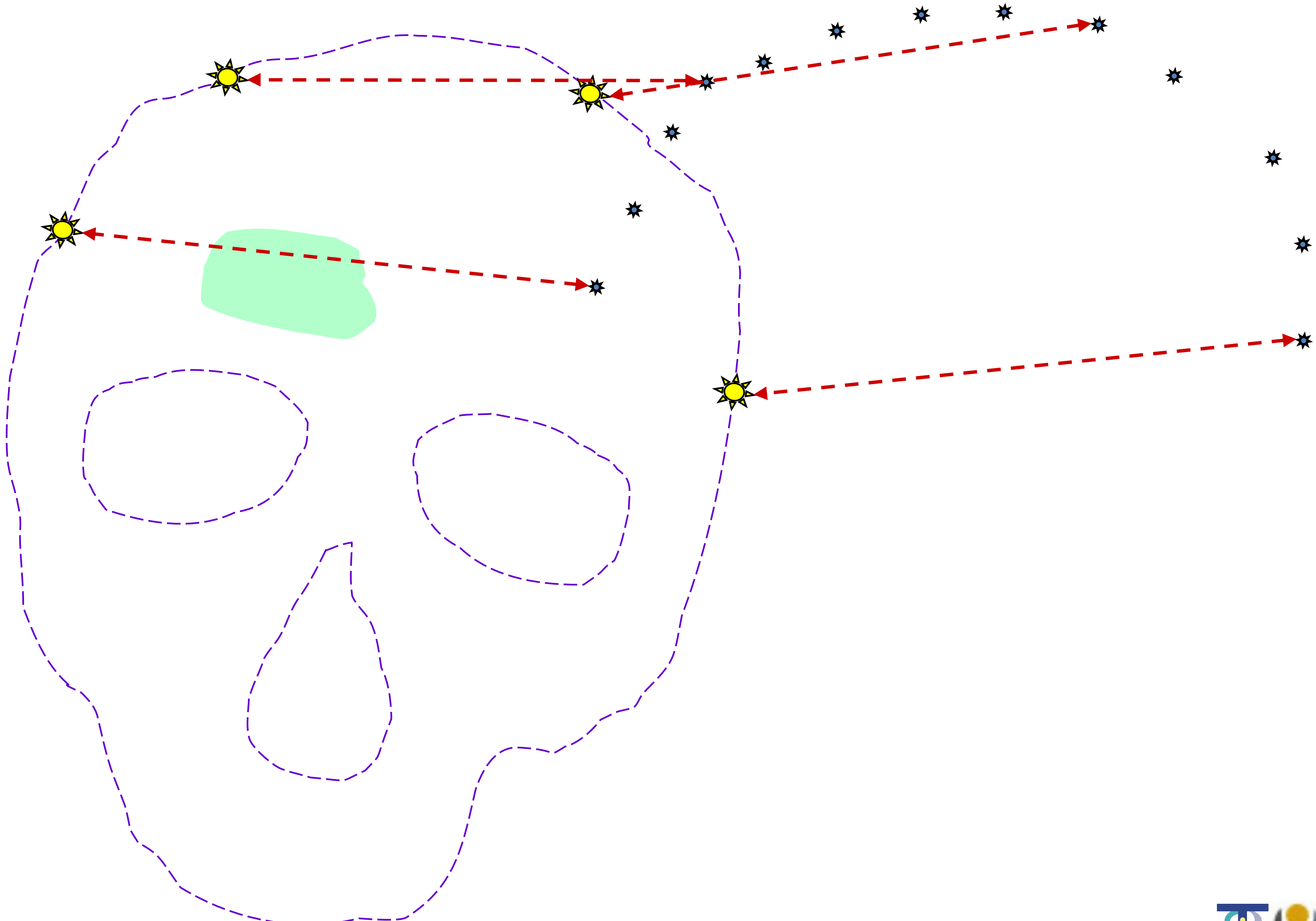
What the computer knows



Find corresponding points & pull!



Find corresponding points & pull!

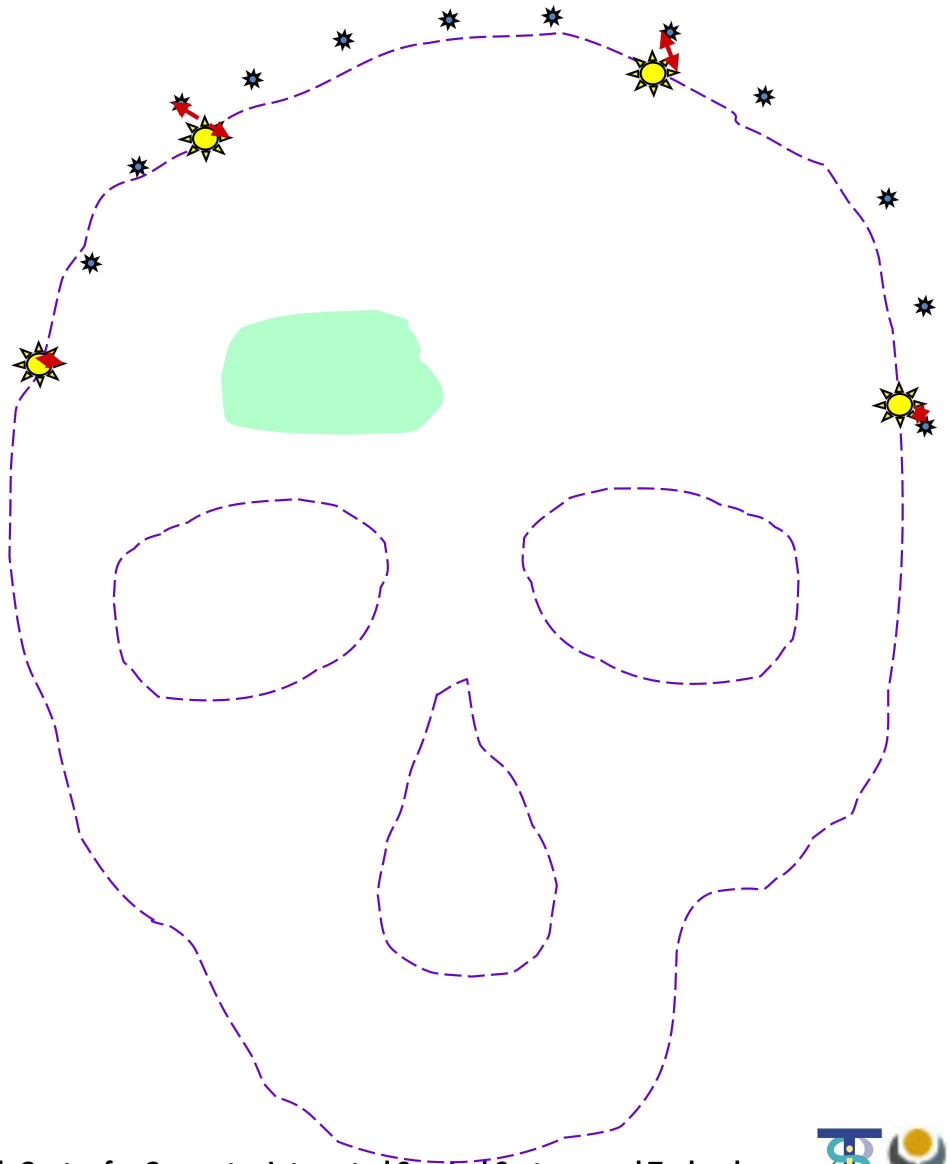


Find corresponding points & pull!

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 $\mathbf{T}$.

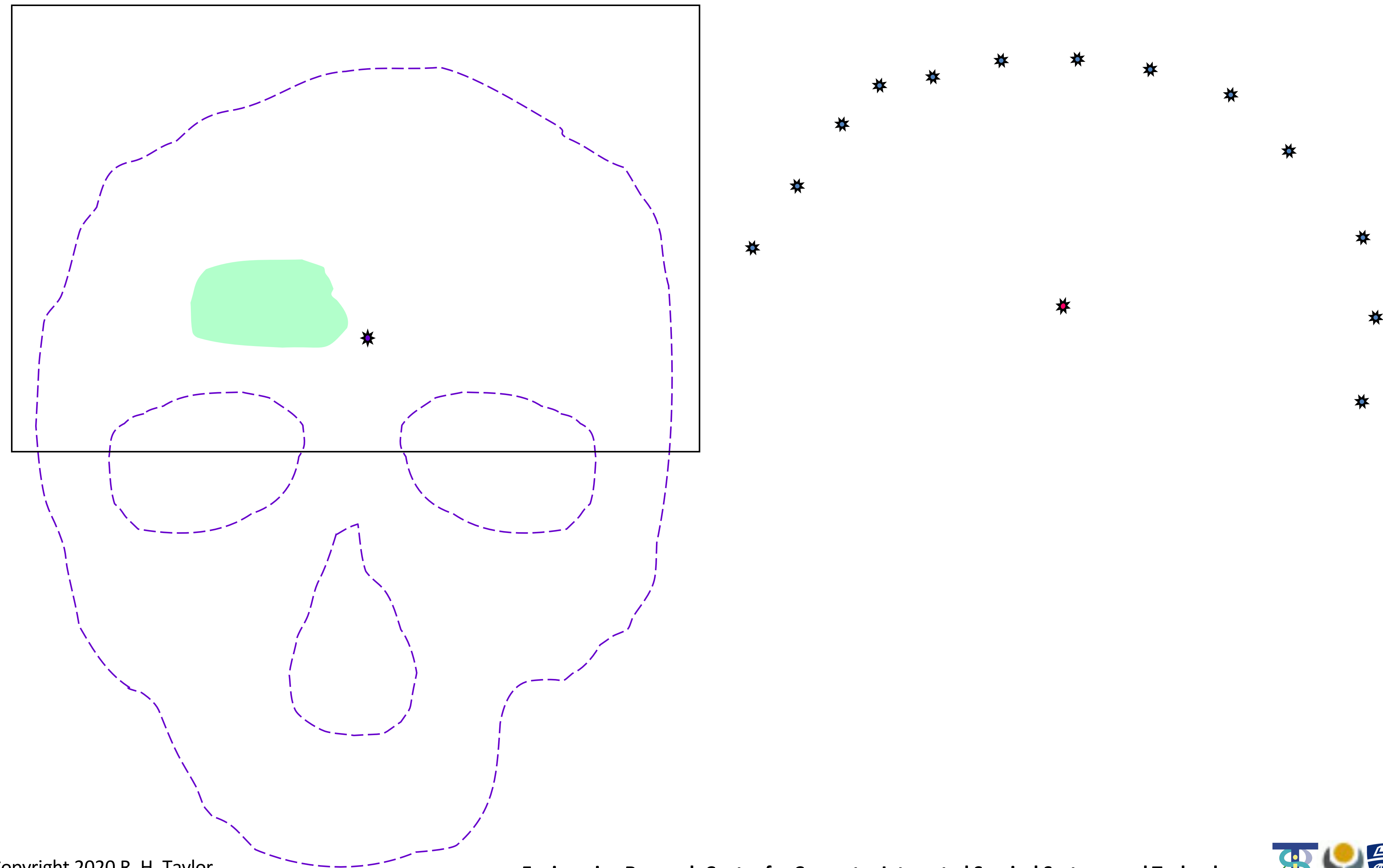
- Optimization uses standard numerical method (steepest gradient descent [Powell]) to find six parameters (3 rotations, 3 translations) defining $\mathbf{T}$.

Head in Hat Algorithm

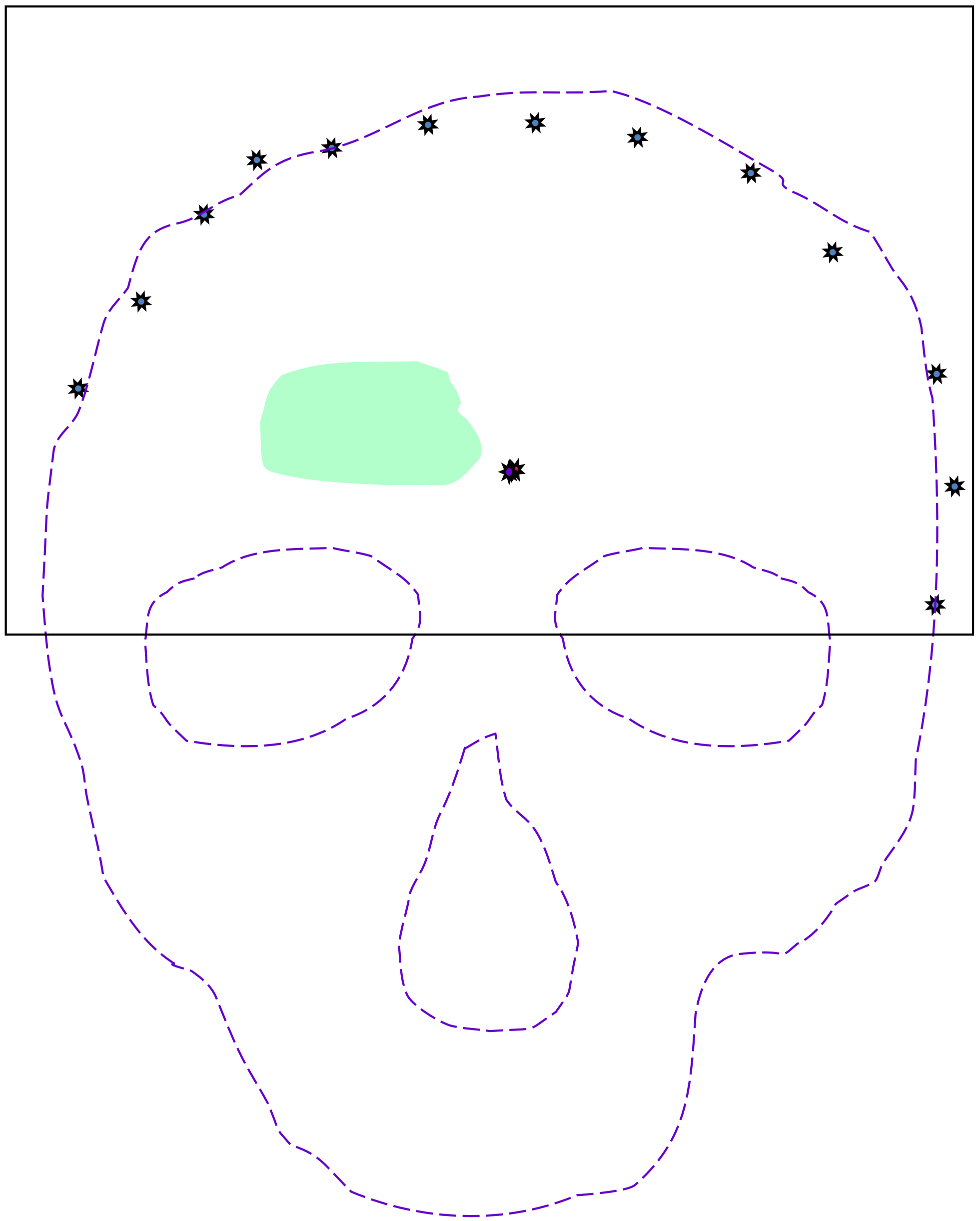
Definition of $d_S(\mathcal{F}_B, \mathbf{f}_i)$

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, $d_S(\mathcal{F}_B, \mathbf{f}_i) = \|\mathbf{q}_i - \mathbf{f}_i\|$

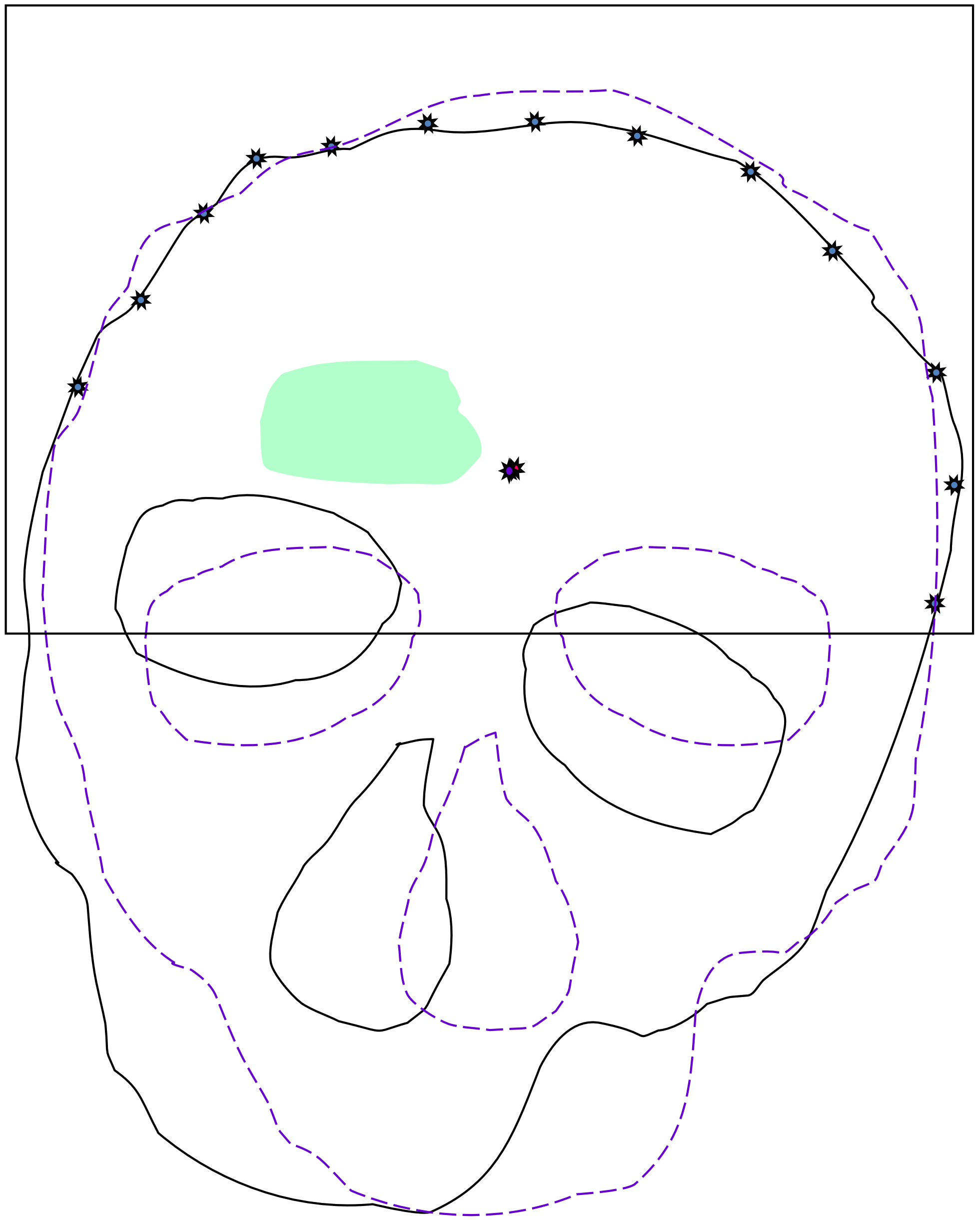
Head-in-hat algorithm: step 0



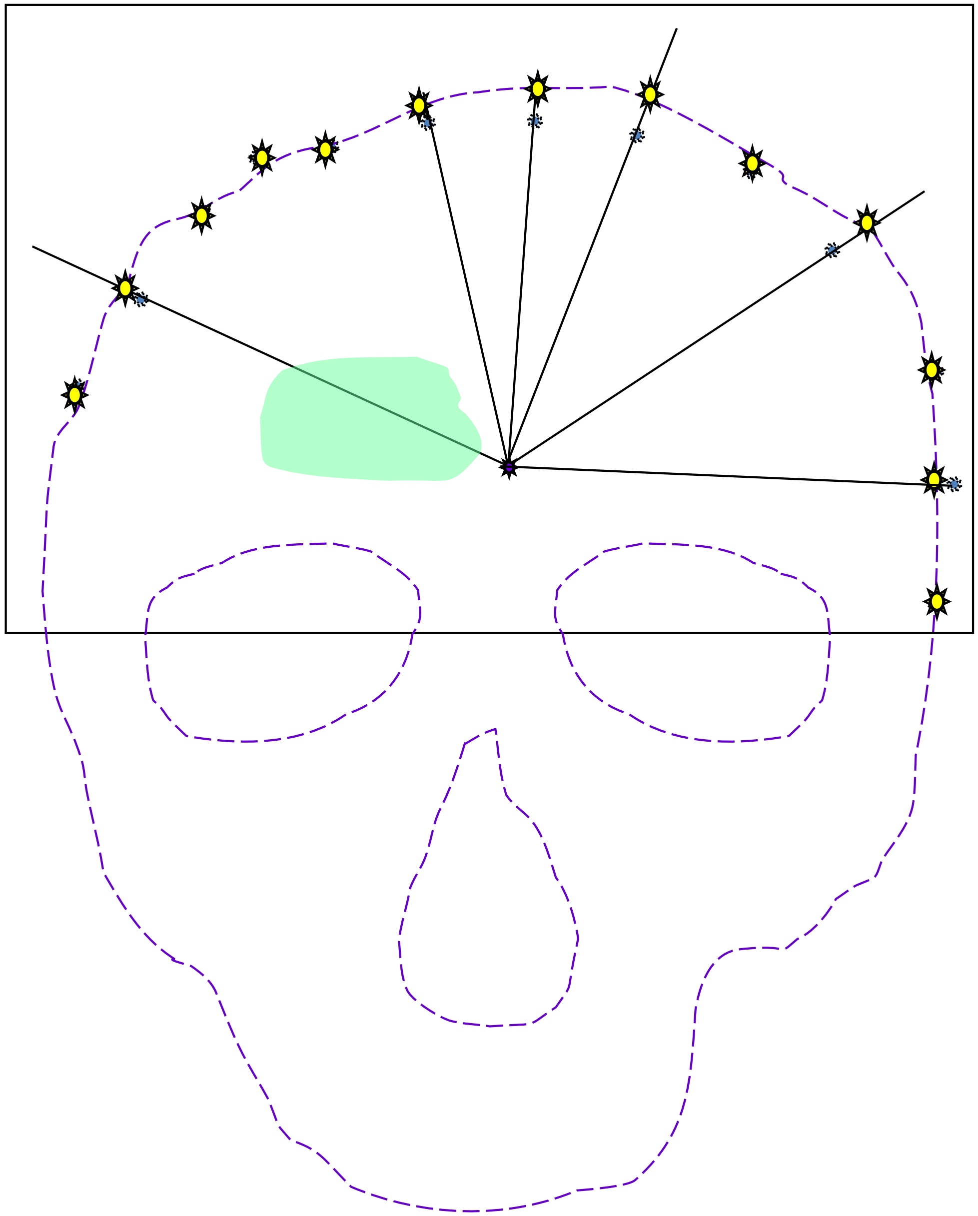
Head-in-hat algorithm: step1



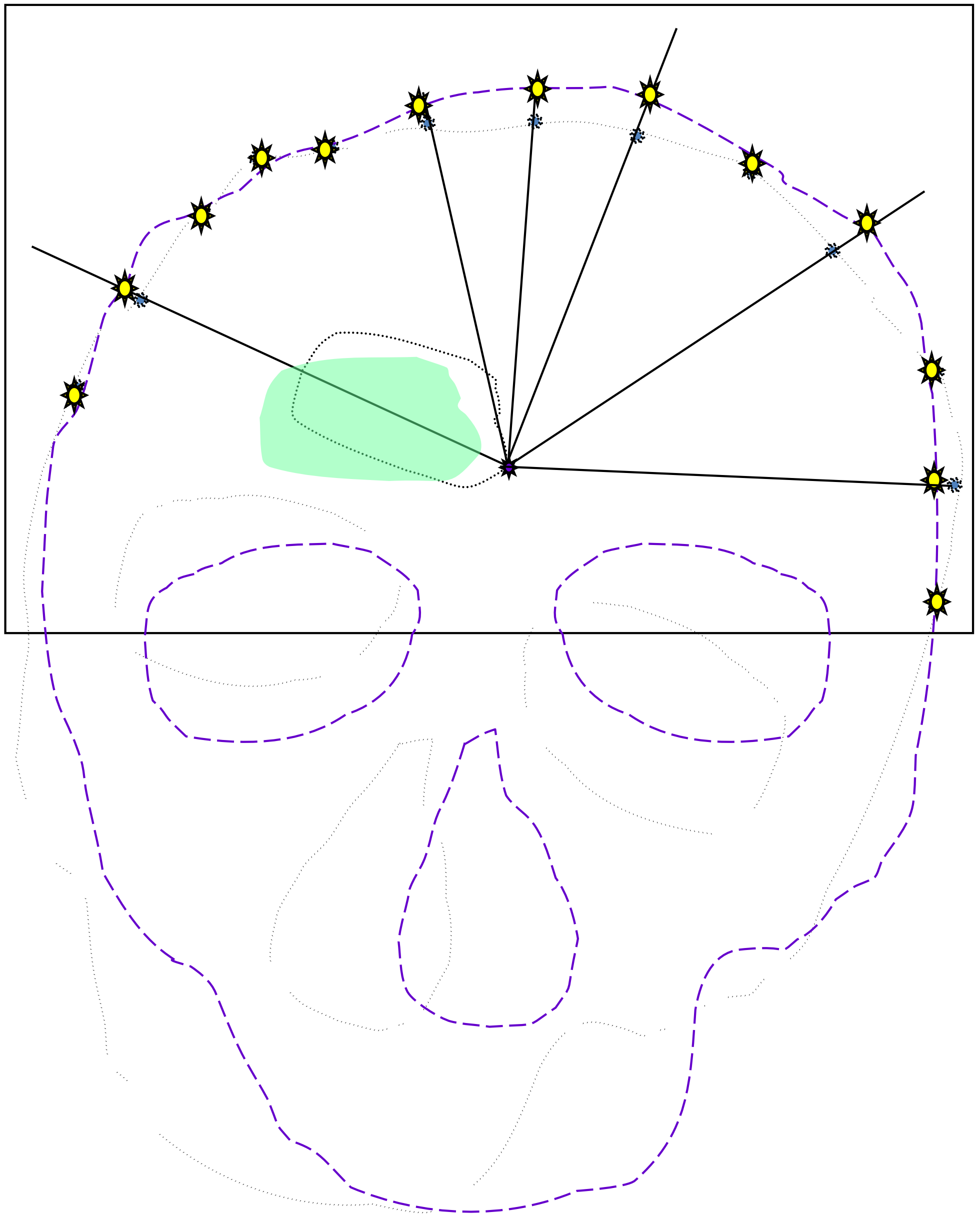
Head-in-hat algorithm: step1



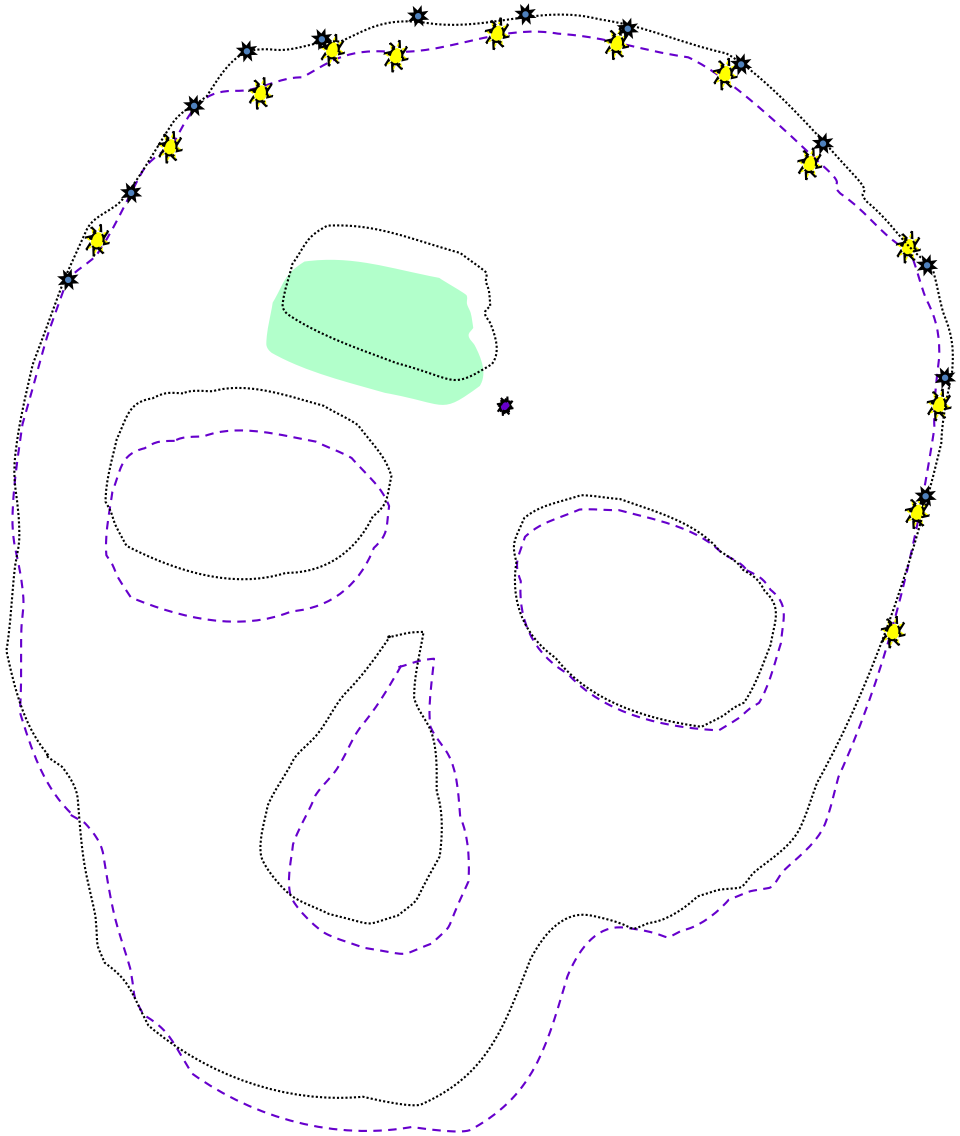
Head-in-hat algorithm: step 2



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

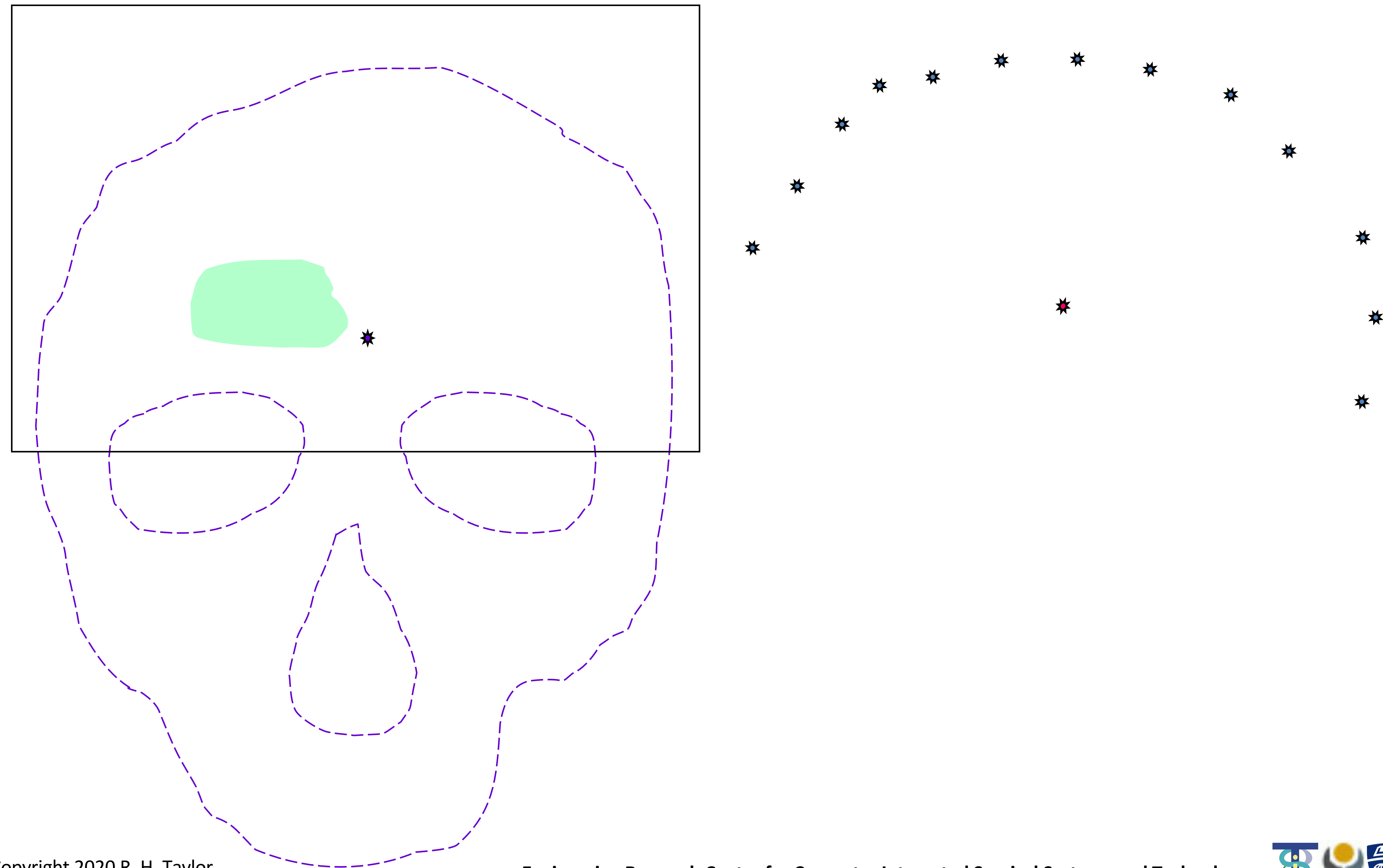
Iterative Closest Point

- 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

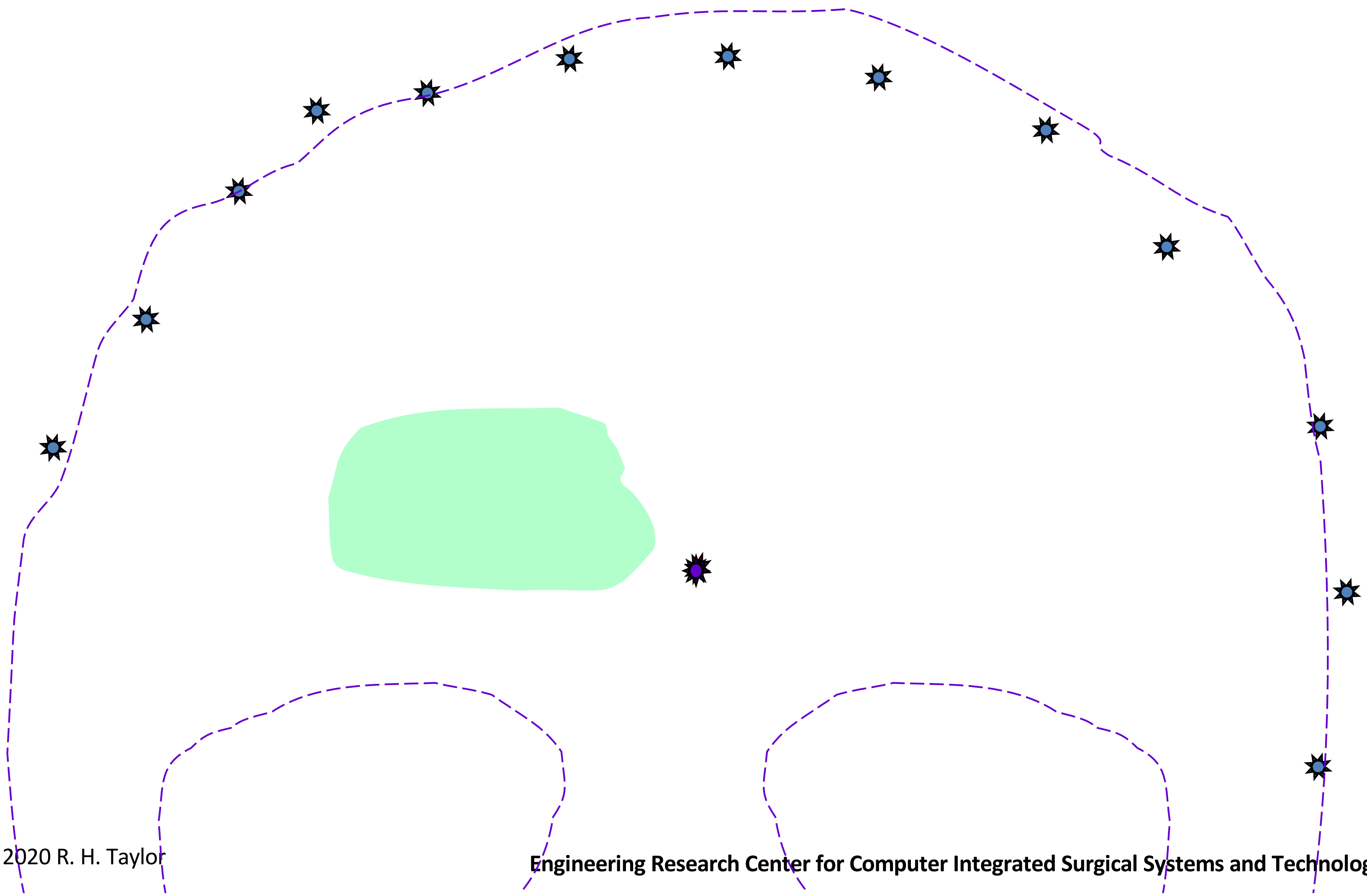
$$D_{k+1} = \sum_i \|\mathbf{v}_i - \mathbf{T}_{k+1} \cdot \mathbf{f}_i)\|^2$$

- Physical Analogy

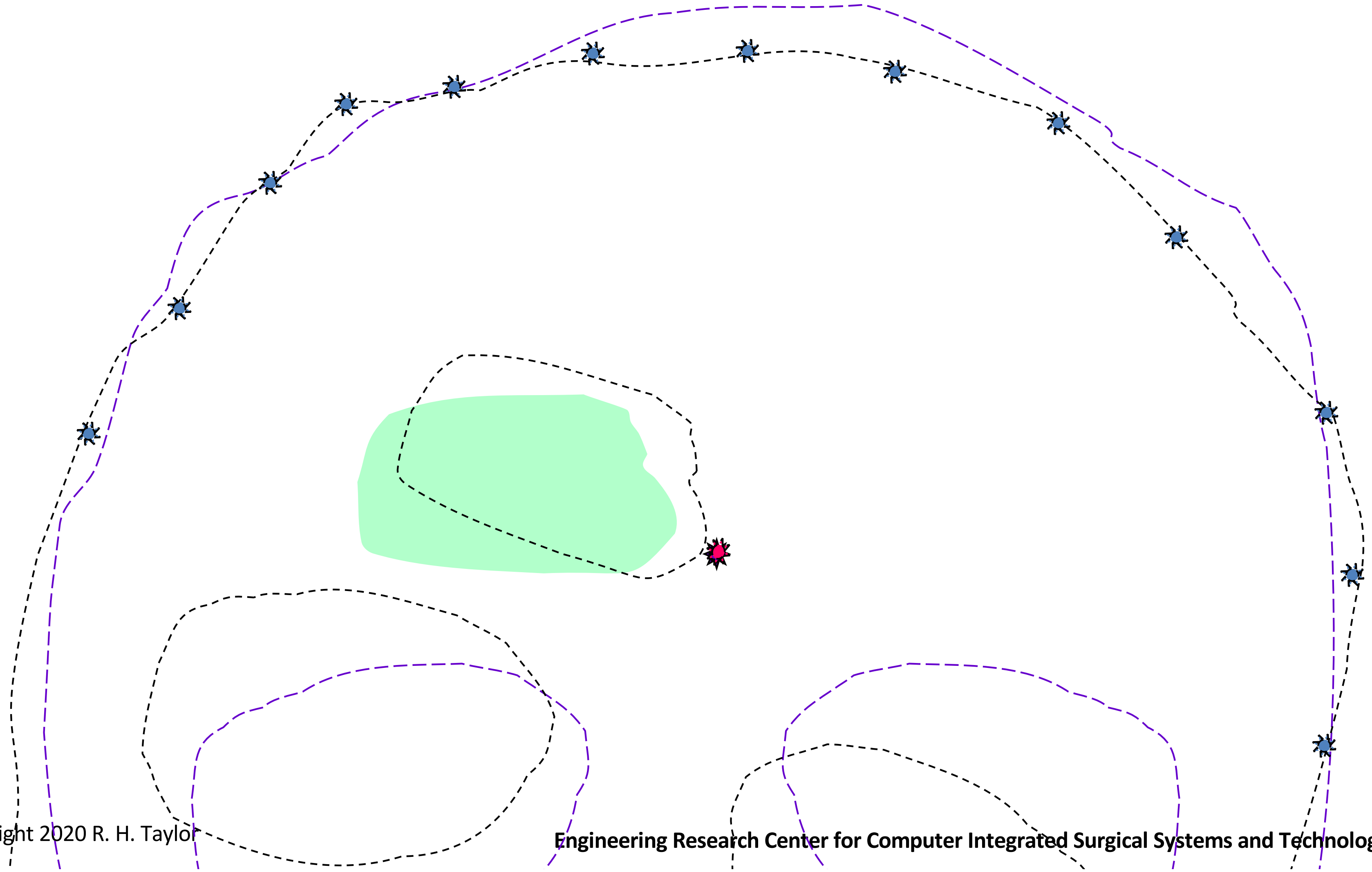
Iterative Closest Point: step 0



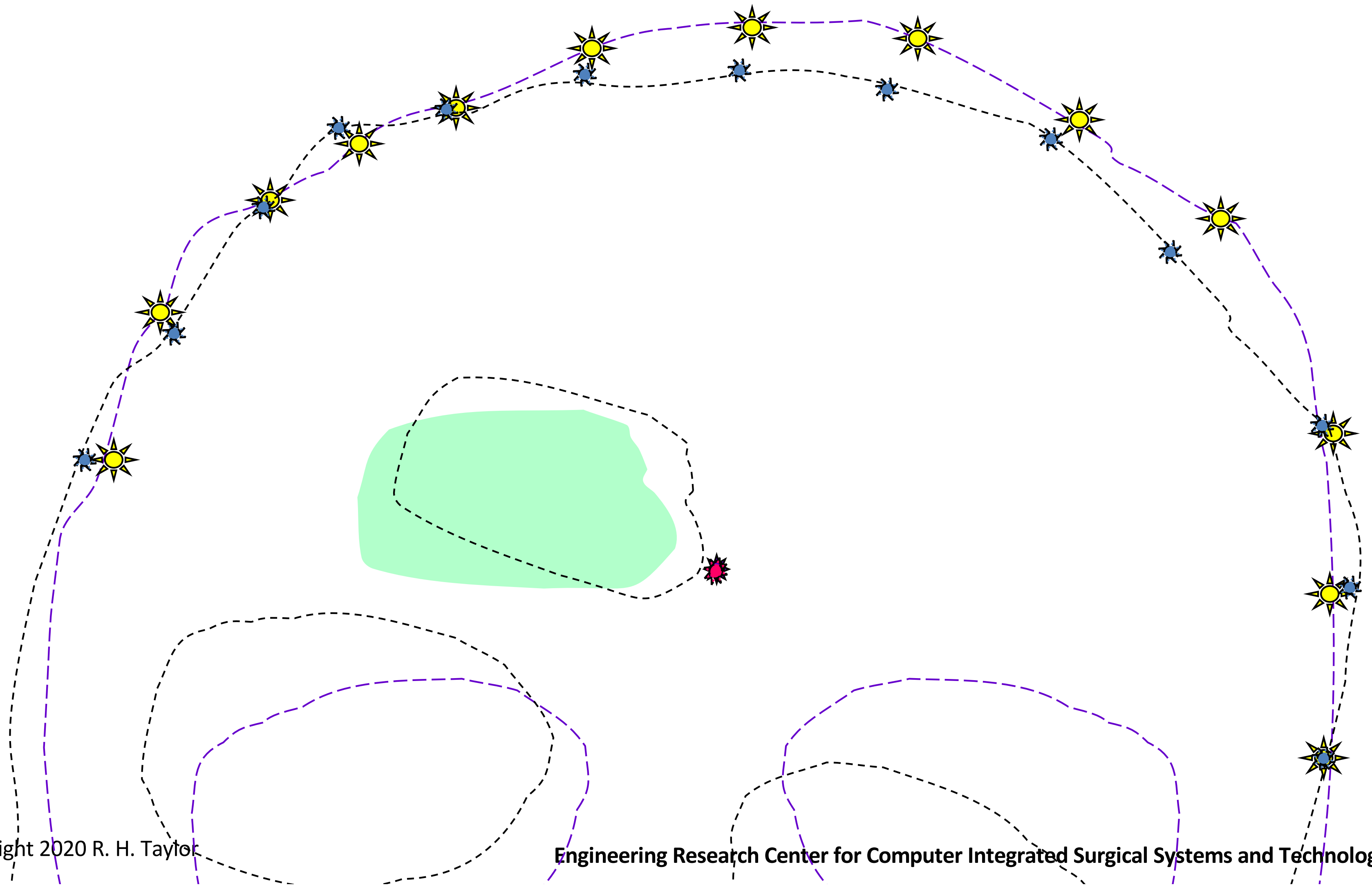
Iterative Closest Point: step1



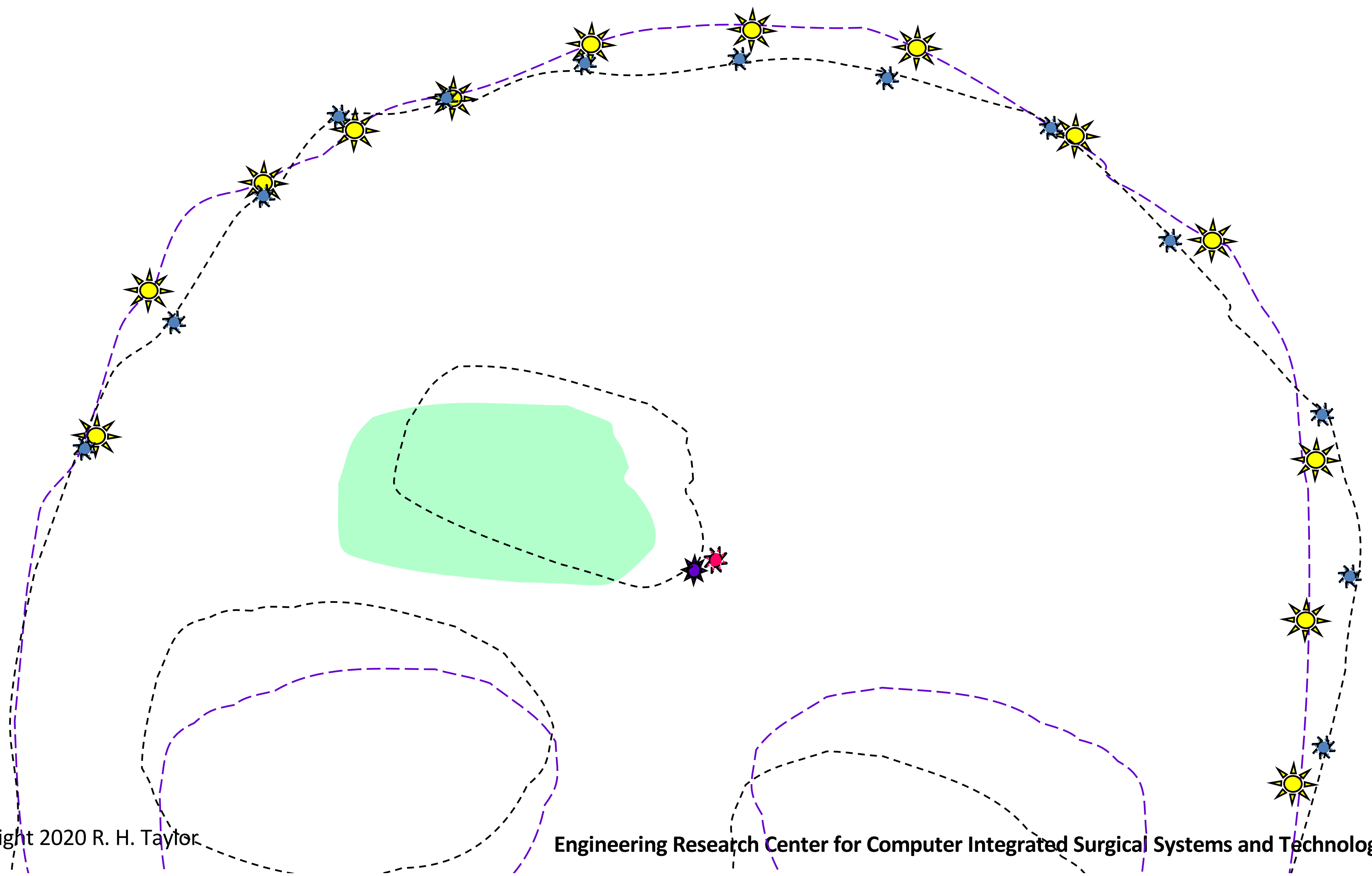
Iterative Closest Point: step1



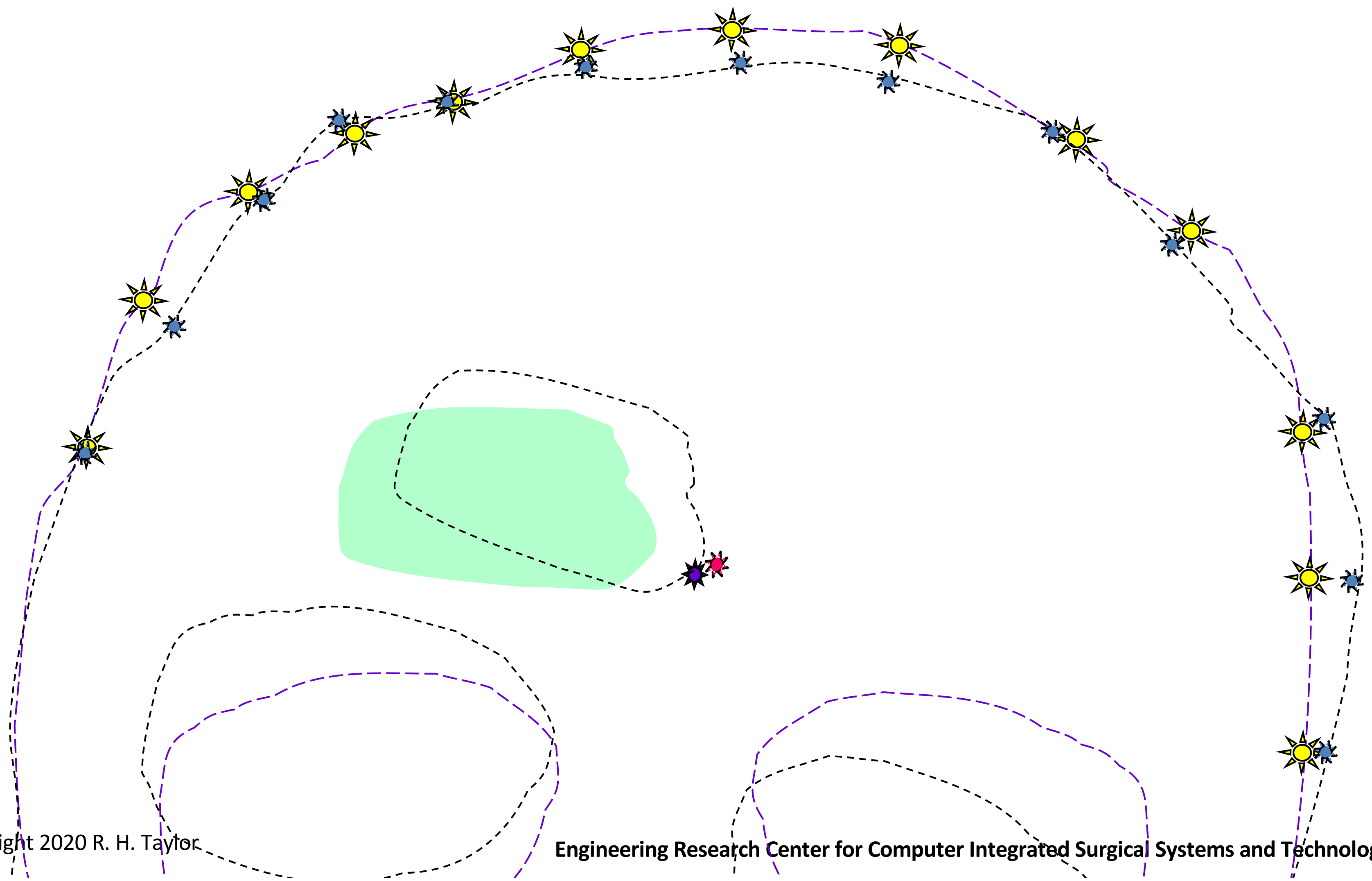
Iterative Closest Point: step2



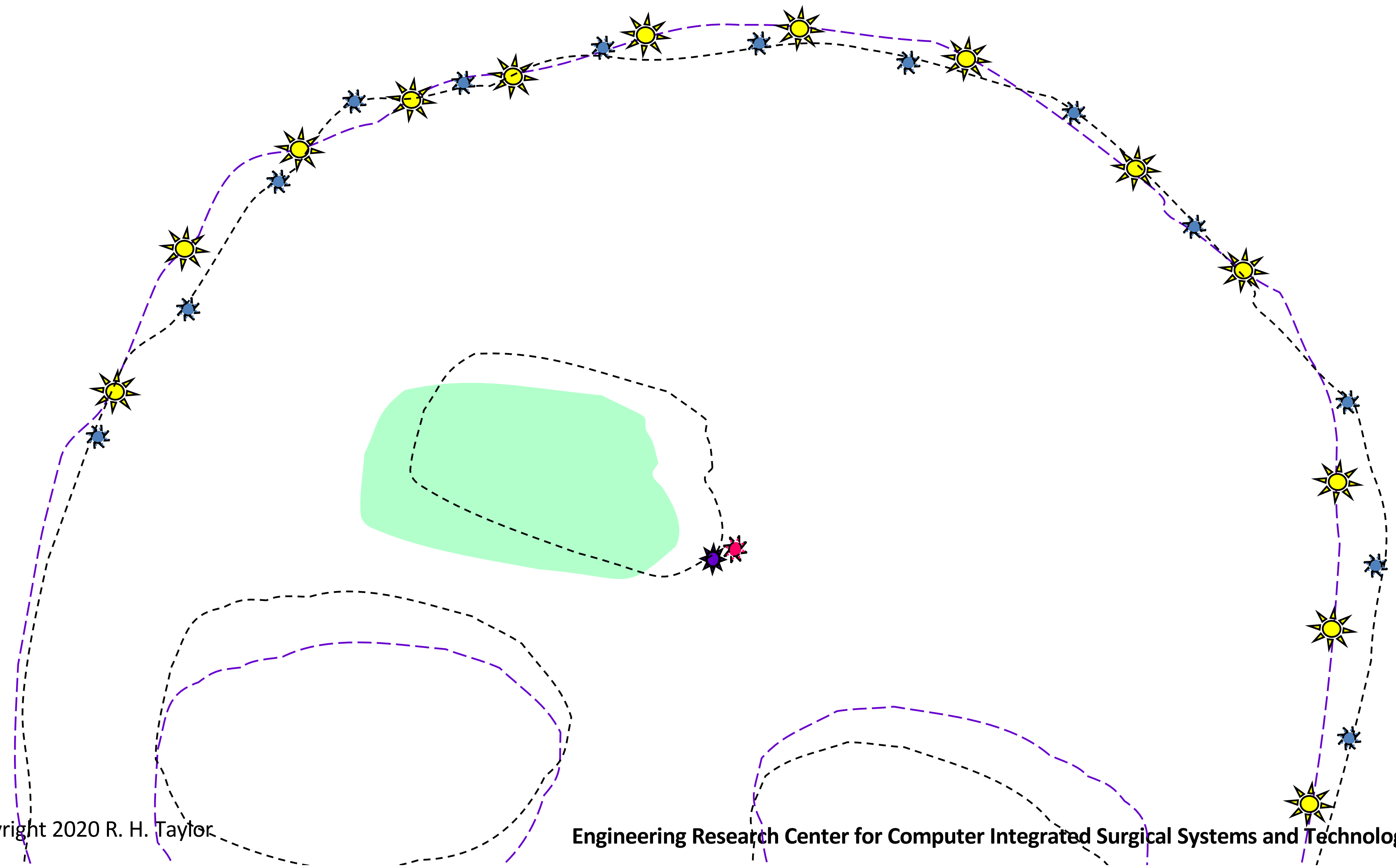
Iterative Closest Point: step 3



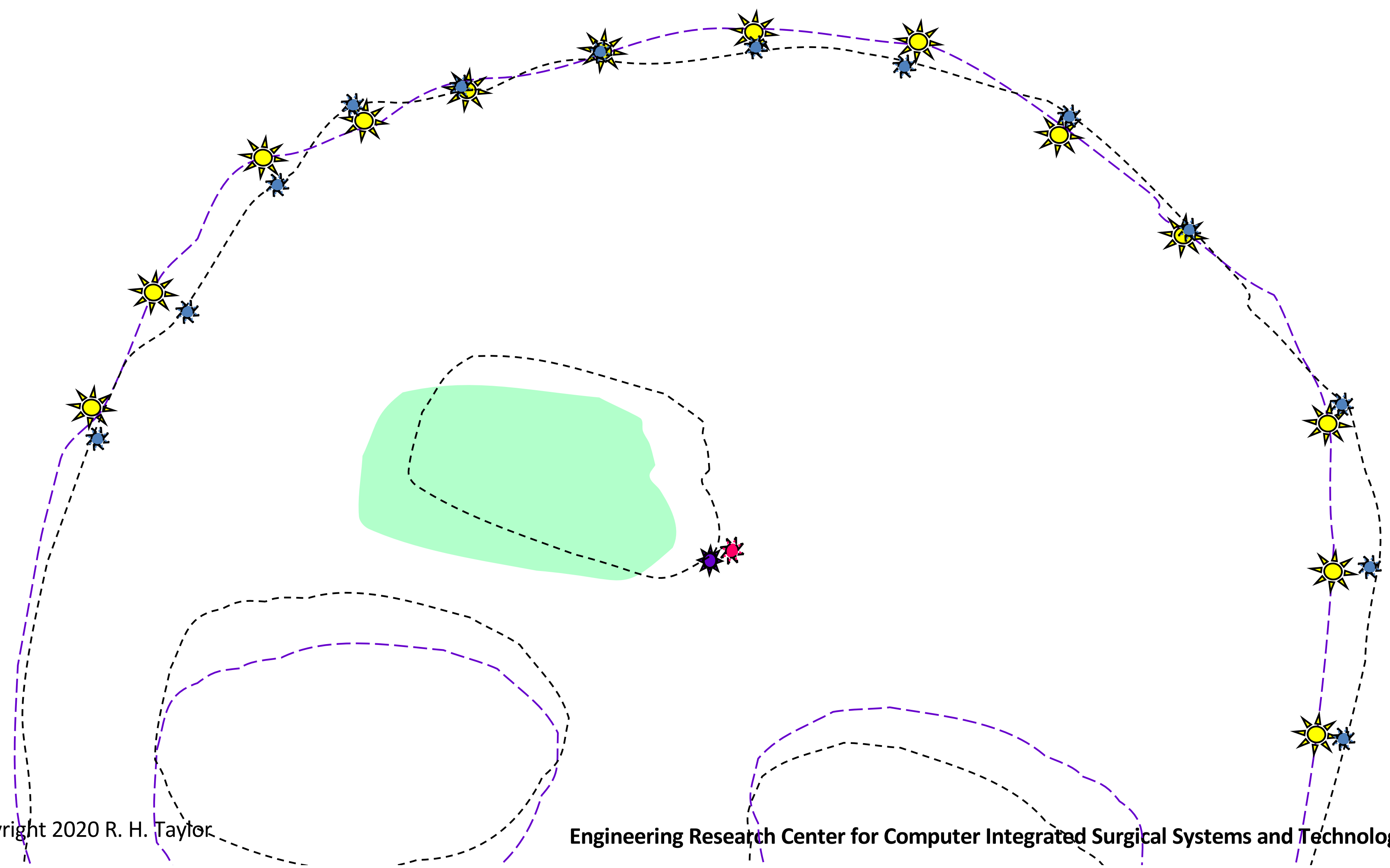
Iterative Closest Point: step 2 interation 2



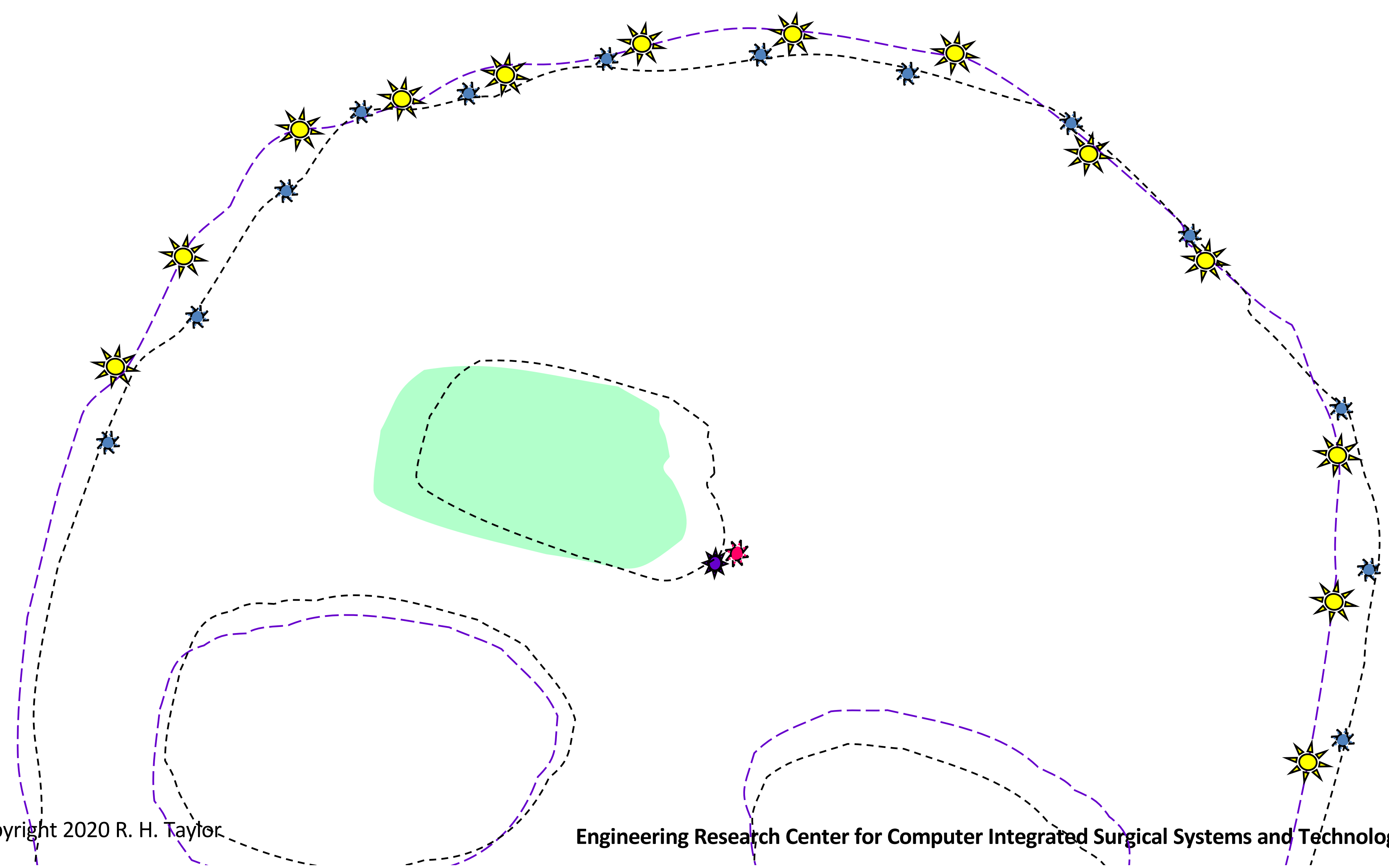
Iterative Closest Point: step 3 interation 2



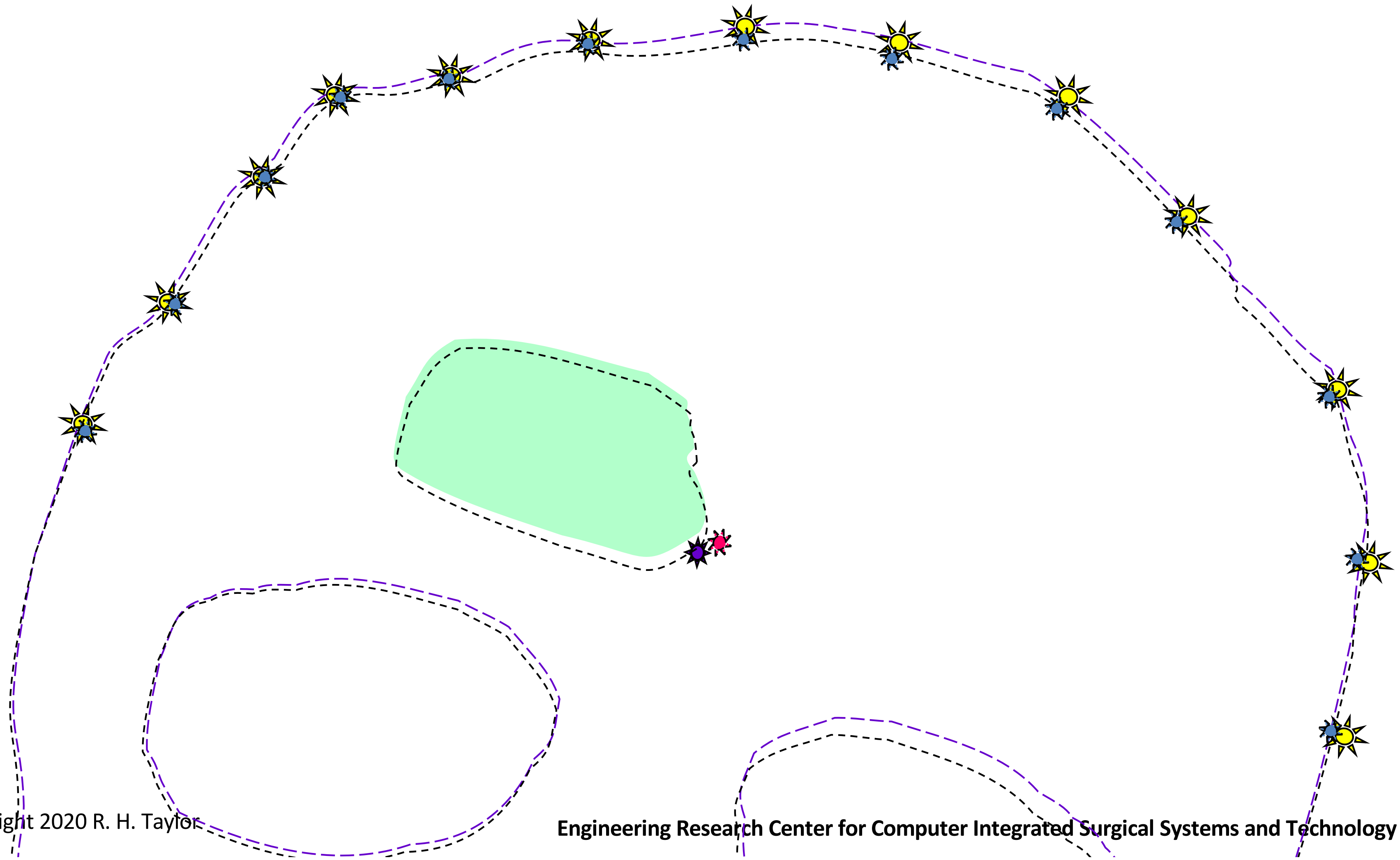
Iterative Closest Point: step 2 interation 3



Iterative Closest Point: step 3 interation 3



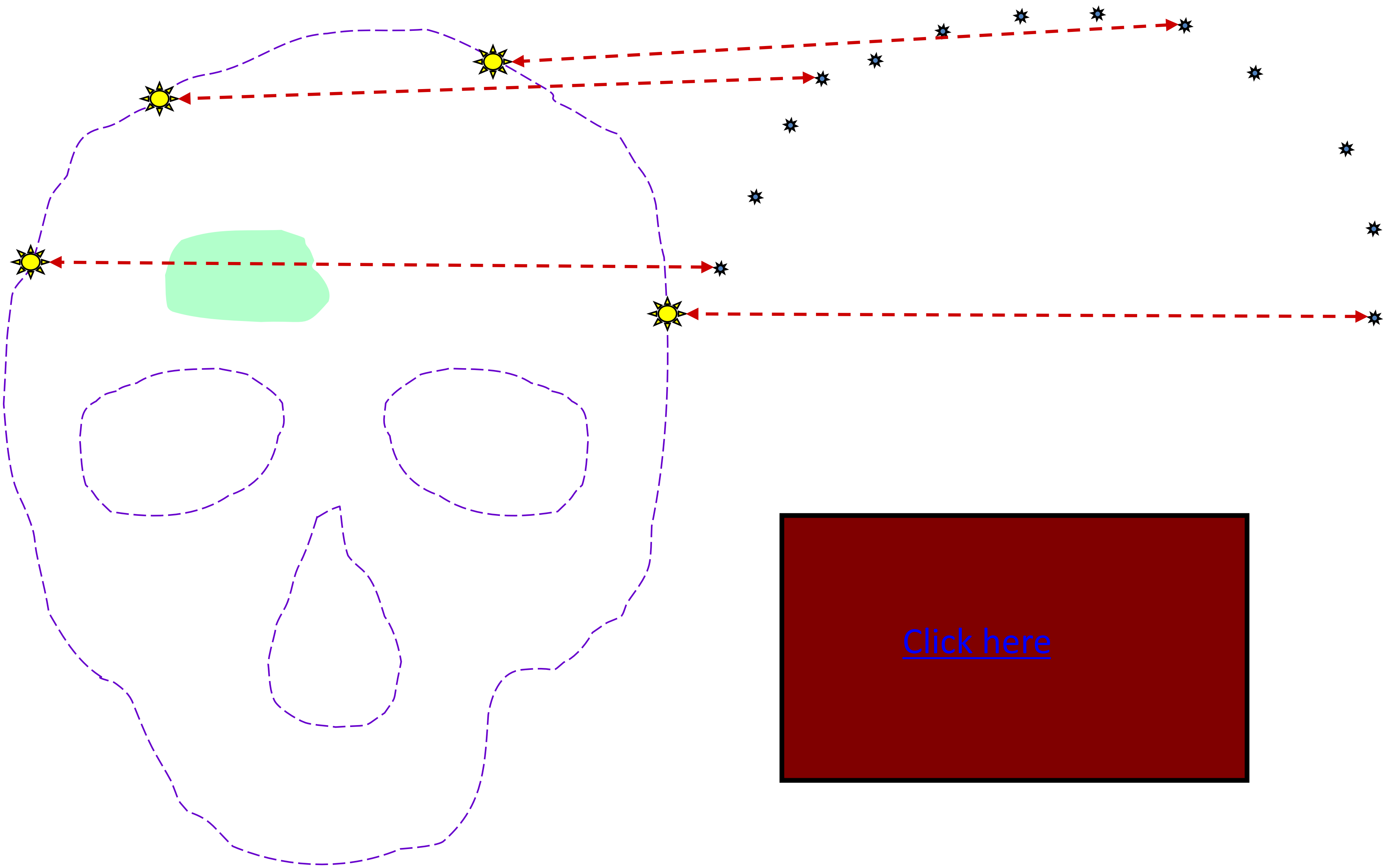
Iterative Closest Point: step 3 interation N



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

Digression: Finding Point Pairs



[Click here]

Outline of a practical ICP code

Given

1. Surface model $\mathtt{M}$ consisting of triangles $\{\mathtt{m}_i\}$
2. Set of points $\mathtt{Q} = \{\vec{\mathbf{q}}_1, \cdots, \vec{\mathbf{q}}_N\}$ known to be on $\mathtt{M}$.
3. Initial guess $\mathbf{F}_0$ for transformation $\mathbf{F}_0$ such that the points $\mathbf{F} \bullet \vec{\mathbf{q}}_k$ lie on $\mathtt{M}$.
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
$\mathtt{C} = \{\cdots, \vec{\mathbf{c}}_k, \cdots\}$	Closest points on $\mathtt{M}$ to $\mathtt{Q}$
$\mathtt{D} = \{\cdots, d_k, \cdots\}$	Distances $\mathrm{d_k} = \left\| \vec{\mathbf{c}}_k - \mathbf{F}_n \cdot \vec{\mathbf{q}}_k \right\|$
$\mathtt{I} = \{\cdots, i_k, \cdots\}$	Indices of triangles $\mathtt{m}_{i_k}$ corresp. to $\vec{\mathbf{c}}_k$
$\mathtt{A} = \{\cdots, \vec{\mathbf{a}}_k, \cdots\}$	Subset of $\mathtt{Q}$ with valid matches
$\mathtt{B} = \{\cdots, \vec{\mathbf{b}}_k, \cdots\}$	Points on $\mathtt{M}$ corresponding to $\mathtt{A}$
$\mathtt{E} = \{\cdots, \vec{\mathbf{e}}_k, \cdots\}$	Residual errors $\vec{\mathbf{b}}_k - \mathbf{F} \cdot \vec{\mathbf{a}}_k$

$$\left[\sigma_n, \ (\varepsilon_{\max})_n, \bar{\varepsilon}_n\right] = \left[\frac{\sum_{\mathrm{k}} \vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}{NumElts(\mathtt{E})}; \quad \max_{\mathrm{k}} \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}; \ \frac{\sum_{\mathrm{k}} \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}}{NumElts(\mathtt{E})}\right]$$

Outline of a practical ICP code

Step 0 : (initialization)

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

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

$$n \leftarrow 0; \quad \eta_n \leftarrow \text{large number}$$

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

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

$$\mathtt{D} \leftarrow \{\cdots, \left\| \vec{\mathbf{c}}_k - \mathbf{F}_0 \bullet \vec{\mathbf{q}}_k \right\|, \cdots\}$$

Outline of a practical ICP code

Step 1: (matching)

$\mathtt{A} \leftarrow \emptyset; \ \ \mathtt{B} \leftarrow \emptyset$

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

begin

$bnd_k = \left\| \mathbf{F}_n \bullet \vec{\mathbf{q}}_k - \vec{\mathbf{c}}_k \right\|$

$\left[\vec{\mathbf{c}}_k, i, d_k \right] \leftarrow \text{FindClosestPoint}\left(\mathbf{F}_n \bullet \vec{\mathbf{q}}_k, \vec{\mathbf{c}}_k, i_k, bnd_k, \mathtt{T} \right);$

// Note: develop first with simple
// search. Later make more
// sophisticated, using $\mathtt{T}$

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

// See also subsequent notes

end

Outline of a practical ICP code

Step 1: (matching)

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

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

begin

$bnd_k = \left\| \mathbf{F}_n \bullet \vec{\mathbf{q}}_k - \vec{\mathbf{c}}_k \right\|$

$\left[\vec{\mathbf{c}}_k, i, d_k \right] \leftarrow$ Fin

//

//

//

if ($d_k < \eta_n$) then

//

end

Note: If using a tree search, you can use previous match to get a reasonable initial bound. E.g.,

$$bnd_k = \left\| \vec{\mathbf{c}}_k - \mathbf{F}_n \bullet \vec{\mathbf{q}}_k \right\|$$

and then pass that to the tree search. Alternatively, you can find the closest point on triangle i_k and use that to get an initial bound bnd_k for the search

Outline of a practical ICP code

Step 2 : (transformation update)

$$n \leftarrow n+1$$

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

$$\sigma_n \leftarrow \frac{\sqrt{\sum_k \vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}}{NumElts(\mathtt{E})}; \quad (\varepsilon_{\max})_n \leftarrow \max_k \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}\,; \; \bar{\varepsilon}_n \leftarrow \frac{\sum_k \sqrt{\vec{\mathbf{e}}_k \cdot \vec{\mathbf{e}}_k}}{NumElts(\mathtt{E})}$$

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 $3\bar{\varepsilon}_n$. If the number of valid matches begins to fall significantly, one can increase it adaptively. Too tight a bound may encourage false minima

Also, if the mesh is incomplete, it may be advantageous to exclude 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 $\gamma \le \bar{\varepsilon}_n / \bar{\varepsilon}_{n-1} \le 1$ for some value γ (e.g., $\gamma \cong .95$) for several iterations.

Short further note: ICP related methods

- There is an extensive literature on methods based on ideas similar to ICP. Surveys and tutorials describing some of them may be found at
  - http://www.cs.princeton.edu/~smr/papers/fasticp/fasticp_paper.pdf
  - http://www.mrpt.org/Iterative_Closest_Point_%28ICP%29_and_other_matching_algorithms
- There are also a number of methods that incorporate a probabilistic framework. One example is the "Generalized-ICP" method of Segal, Haehnel, and Thrun
  - Aleksandr V. Segal, Dirk Haehnel, and Sebastian Thrun, "Generalized-ICP", in *Robotics: Science and Systems,* 2009.
  - http://www.robots.ox.ac.uk/~avsegal/resources/papers/Generalized_ICP.pdf

Typical Generalized ICP Algorithm

Outline below is based mostly on from paper by A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP", in *Robotics: Science and Systems*, 2009.

$n \leftarrow 0$;initialize $\mathbf{F}_0$, threshold value η_0, distribution parameters Φ

Step 1: (matching)

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

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

begin

$\left[\vec{\mathbf{c}}_k, i_k, d_k\right] \leftarrow \text{FindClosestPoint}\left(\mathbf{F}_n \bullet \vec{\mathbf{q}}_k, \vec{\mathbf{c}}_k, i_k, \mathtt{T}\right);$

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

\\ alternative: test if $prob(\vec{\mathbf{q}}_k \sim \vec{\mathbf{c}}_k) > \eta_n$

end

Step 2: (transformation update)

$n \leftarrow n+1$

$$\mathbf{F}_n \leftarrow \underset{\mathbf{F}}{\arg\max}\ prob(\mathbf{F} \bullet A \sim B; \Phi) = \underset{\mathbf{F}}{\arg\max} \prod_i prob(\mathbf{F} \bullet \vec{\mathbf{a}}_i \sim \vec{\mathbf{b}}_i; \Phi)$$

$$= \underset{\mathbf{F}}{\arg\min} \sum_i -\log prob(\mathbf{F} \bullet \vec{\mathbf{a}}_i \sim \vec{\mathbf{b}}_i; \Phi)$$

Step 3: (adjustment)

update threshold η_n and distribution parameters Φ

Step 4: (iteration)

if TerminationTest($\cdots$) then stop. Otherwise, go back to step 1 // see notes

Related concept: Estimation with Uncertainty

Suppose you know something about the uncertainty of the sample data at each point pair (e.g., from sensor noise and/or model error). I.e.,

$$\vec{\mathbf{a}}_k \in \boldsymbol{A}_k;\ \ \vec{\mathbf{b}}_k \in \boldsymbol{B}_k;\ \ \mathrm{cov}(\boldsymbol{A}_k, \boldsymbol{B}_k) = \mathbf{C}_k = \mathbf{Q}_k \Lambda_k \mathbf{Q}_k^{\,T}$$

Then an appropriate distance metric is the Mahalabonis distance

$$\mathrm{D}(\vec{\mathbf{a}}_k, \vec{\mathbf{b}}_k) = (\vec{\mathbf{a}}_k - \vec{\mathbf{b}}_k)^T \mathbf{C}_k^{\,-1} (\vec{\mathbf{a}}_k - \vec{\mathbf{b}}_k) = \vec{\mathbf{d}}_k^{\,T} \Lambda_k^{\,-1} \vec{\mathbf{d}}_k$$

where

$$\vec{\mathbf{d}}_k = \mathbf{Q}_k^{\,T} (\vec{\mathbf{a}}_k - \vec{\mathbf{b}}_k)$$

This approach is readily extended to the case where the samples are not independent.

Distance Maps

- Many authors
- Somewhat related to ICP and also to level sets
- Basic idea is to precompute the distance to the surface for a dense sampling of the volume.
- Then use the gradient of the distance map to compute an incremental motion that reduces the sum of the distances of all the moving points to the surface.
- Then iterate

Distance Maps

There are a number of very fast algorithms for computing the Euclidean Distance Transform (distance to surface of each point in an image at each point in a 3D volume grid). One example is:

> J. C. Torelli, R. Fabbri, G. Travieso, and O. Bruno, "A High Performance 3D Exact Eeuclidean Distance Transform Algorithm for Distributed Computing", *International Journal of Pattern Recognition and Artificial Intelligence, vol. 24- 6, pp. 897-915, 2010.*

But a web search will disclose many others, together with open source code

Distance Maps

Given

a current registration transformation **F**

Euclidean distance map d($\vec{\mathbf{p}}$)

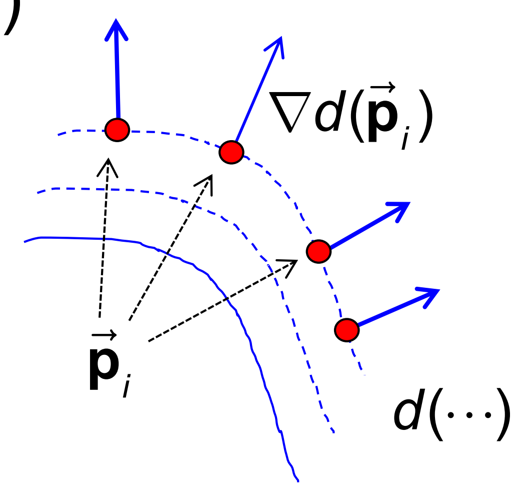
For each sample point $\vec{\mathbf{f}}_i$ compute $\vec{\mathbf{p}}_i = \mathbf{F} \bullet \vec{\mathbf{f}}_i$

Compute a small motion $\Delta\mathbf{F}$

$$\Delta\mathbf{F} = \underset{\Delta\mathbf{F}}{\arg\min} \sum_i (\Delta\mathbf{F} \bullet \vec{\mathbf{p}}_i - \vec{\mathbf{p}}_i) \bullet \nabla d(\vec{\mathbf{p}}_i)$$

Update $\mathbf{F} \leftarrow \Delta\mathbf{F} \bullet \mathbf{F}$

Iteate



Distance Maps

Given

a current registration transformation $\mathbf{F}$

Euclidean distance map $d(\vec{\mathbf{p}})$

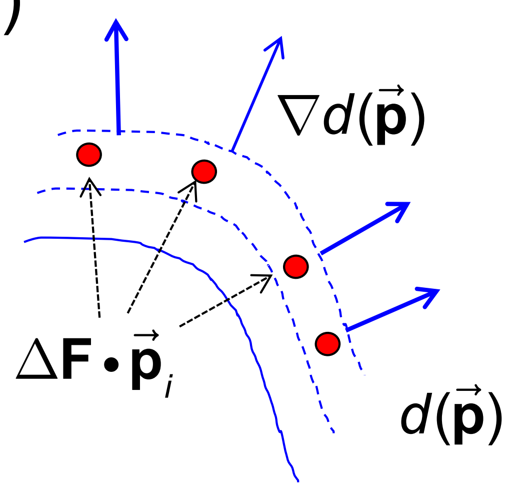
For each sample point $\vec{\mathbf{f}}_i$ compute $\vec{\mathbf{p}}_i = \mathbf{F} \bullet \vec{\mathbf{f}}_i$

Compute a small motion $\Delta\mathbf{F}$

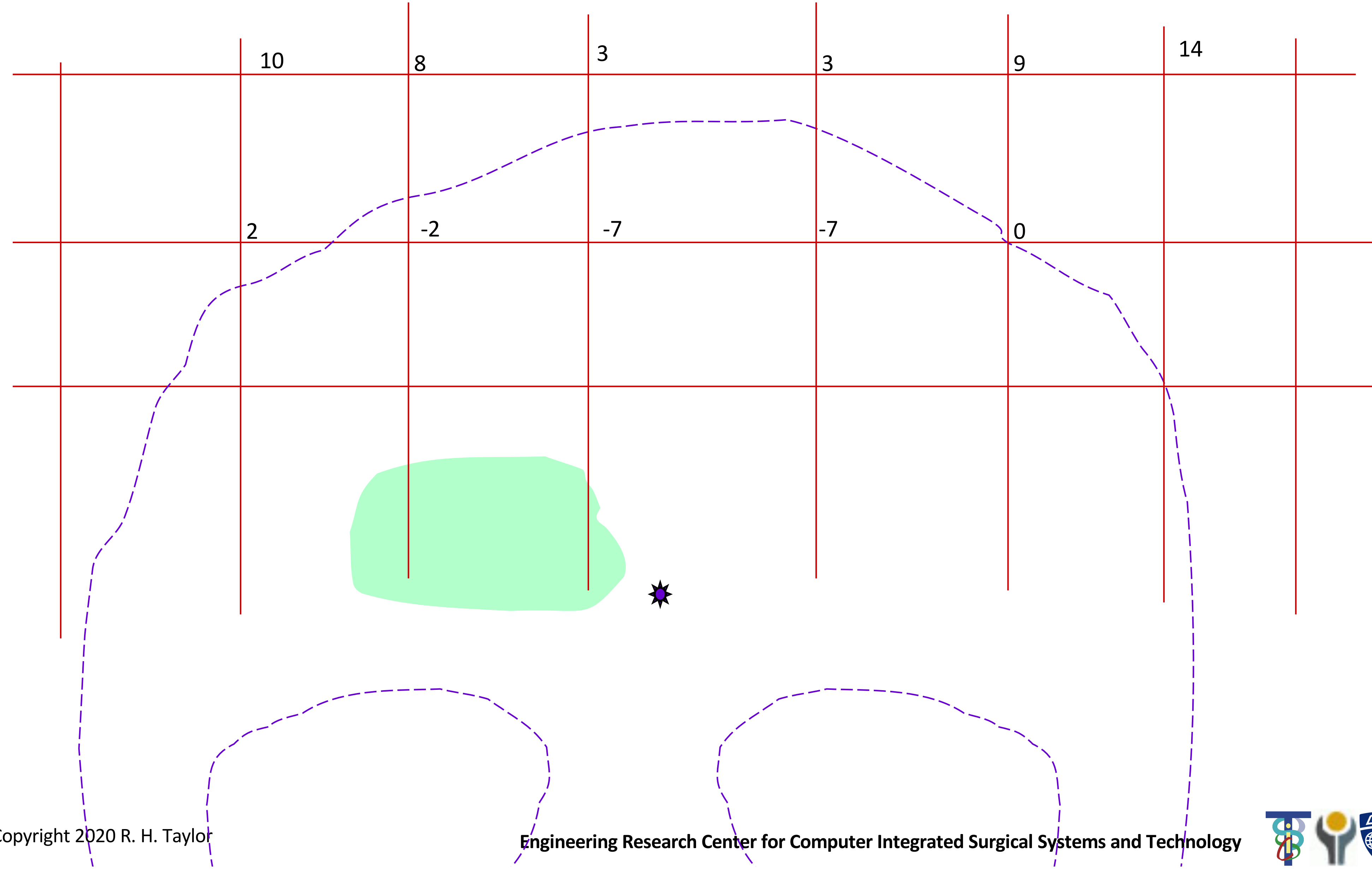
$$\Delta\mathbf{F} = \underset{\Delta\mathbf{F}}{\arg\min} \sum_i (\Delta\mathbf{F} \bullet \vec{\mathbf{p}}_i - \vec{\mathbf{p}}_i) \bullet \nabla d(\vec{\mathbf{p}}_i)$$

Update $\mathbf{F} \leftarrow \Delta\mathbf{F} \bullet \mathbf{F}$

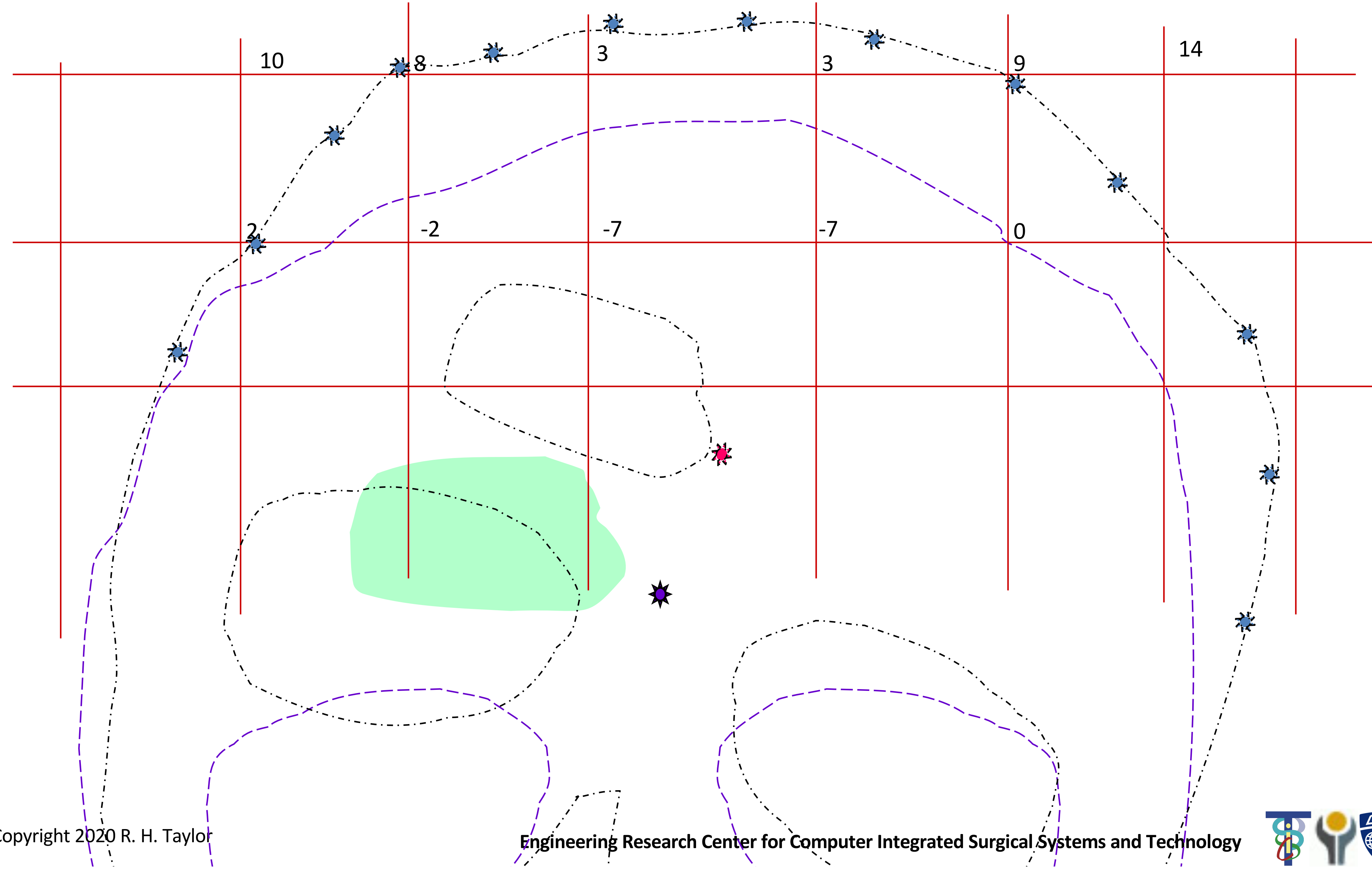
Iteate



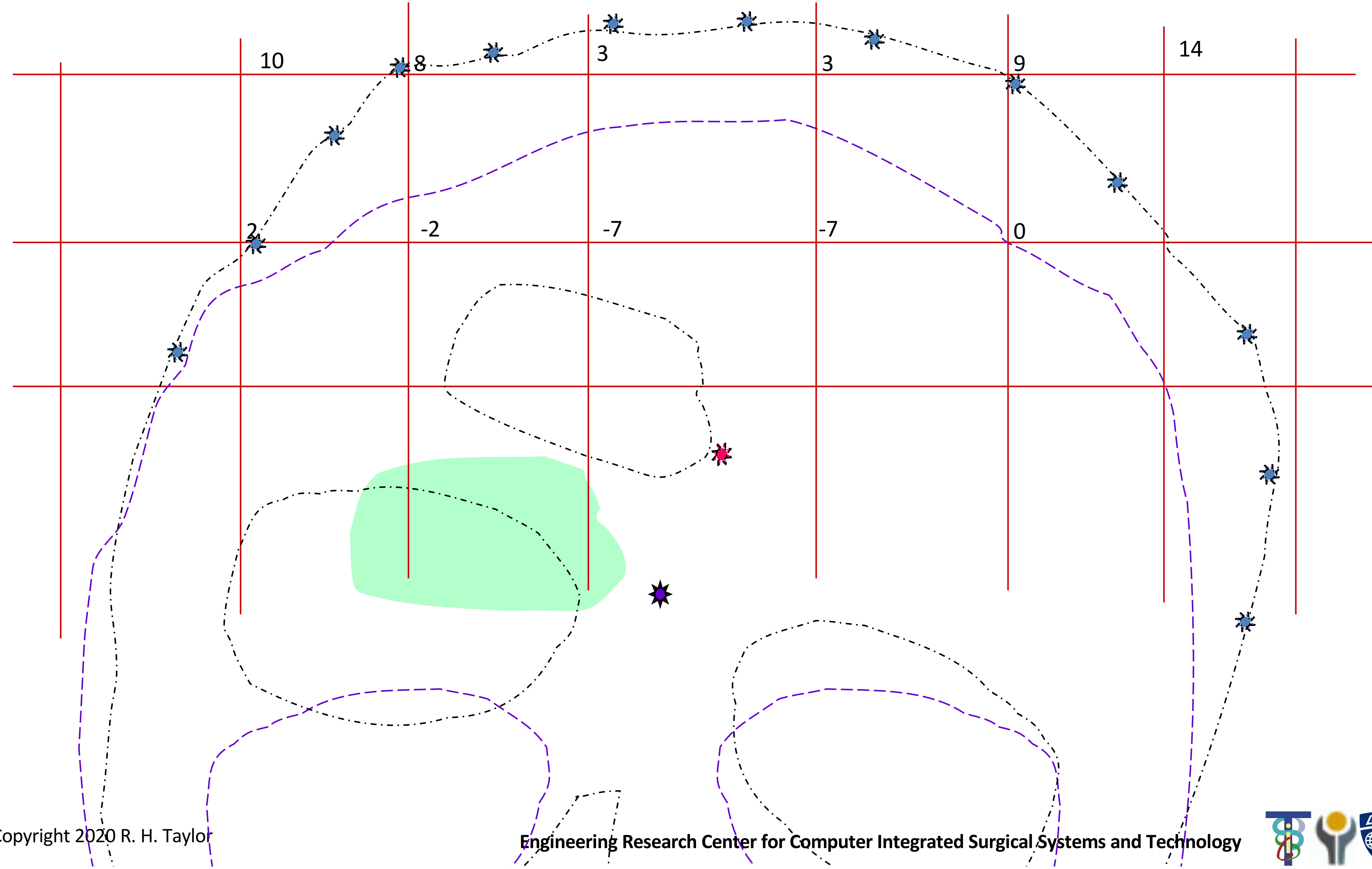
Distance Maps: step 0



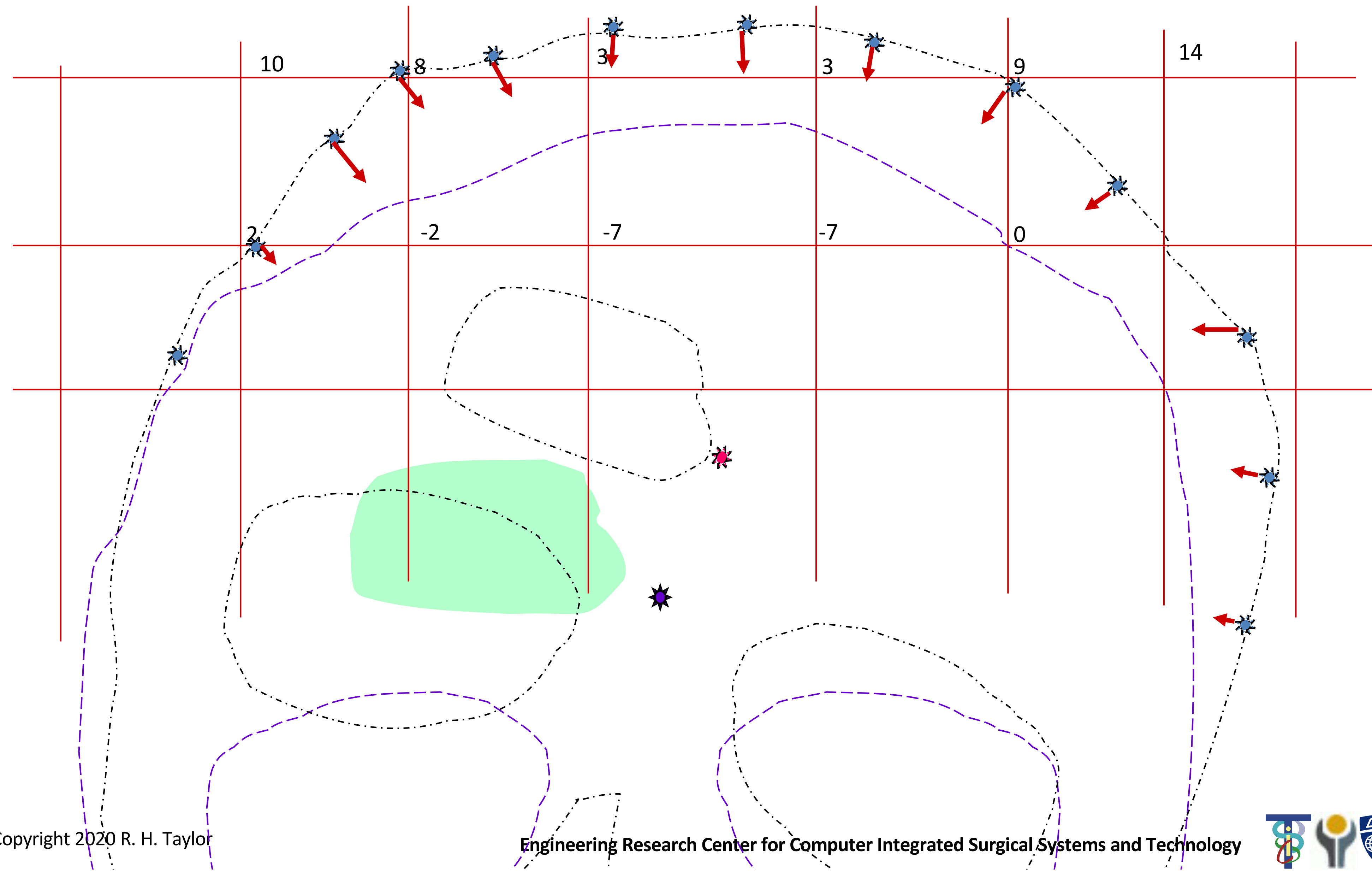
Distance Maps: step 1



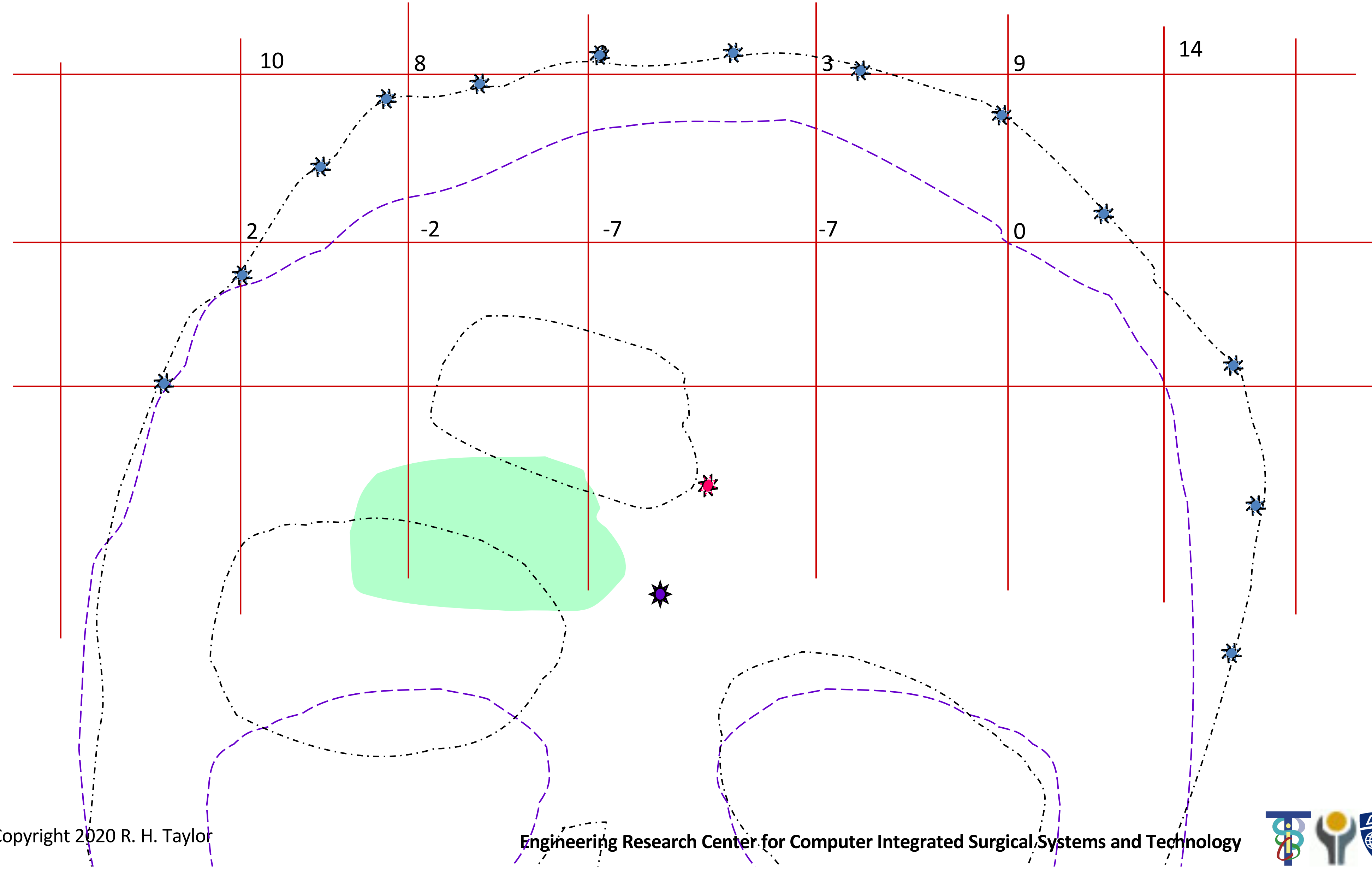
Distance Maps: step 1



Distance Maps: step 2



Distance Maps: step 3



Quiz

- The link will be in the chat
- You have 2 minutes