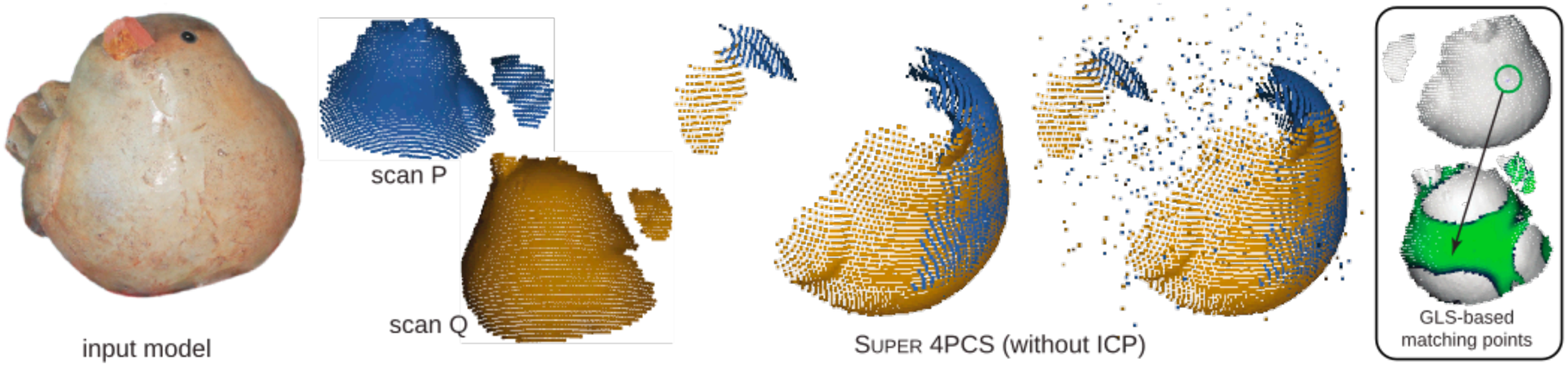
Winter 2023-2024 COMP0119

Acquisition and Processing of 3D Geometry

Niloy J. Mitra



Scanning and Registration

[Image source: Super4PCS]

Last Time

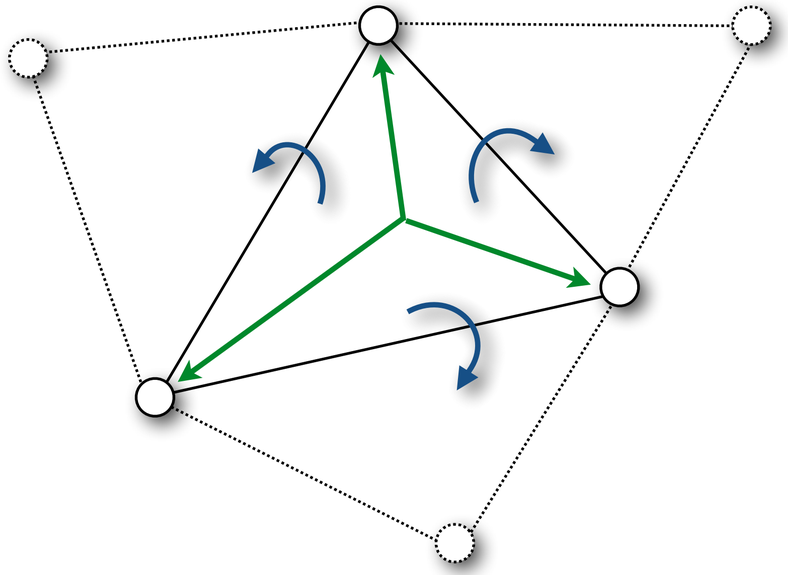
- Polygonal meshes are a good compromise
  - approximation $O(h^2)$
  - arbitrary topology
  - piecewise smooth surfaces
  - adaptive refinement
  - efficient rendering

Last Time

- Polygonal mesh as graph embedding
- Advantages of triangle meshes
- Mesh data structures
  - File formats: STL, OFF, OBJ
  - Connectivity: Face-, edge-, half-edge-based
  - Design criteria?

Face-Based Connectivity

- Vertex:
  - position
  - 1 face
- Face:
  - 3 vertices
  - 3 face neighbors

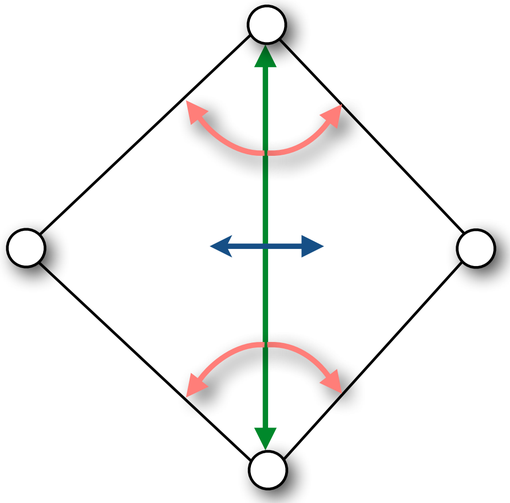


64 B/v

no edges!

Edge-Based Connectivity

- Vertex
  - position
  - 1 edge
- Edge
  - 2 vertices
  - 2 faces
  - 4 edges
- Face
  - 1 edge



120 B/v

edge orientation?

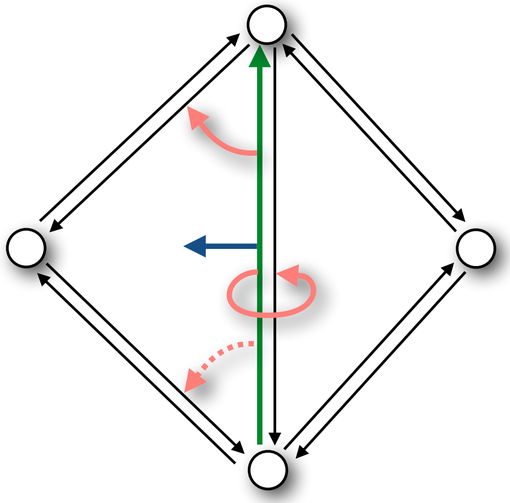
Halfedge-Based Connectivity

- Vertex
  - position
  - 1 halfedge
- Halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges
- Face
  - 1 halfedge

在半边数据结构中，每个半边通常会包含以下关系指针：

- **顶点指针**：指向半边起点的顶点。
- **对偶半边指针**：指向与当前半边共享同一条边的另一半边。
- **面指针**：指向当前半边所属的面。
- **下一个半边指针**：指向同一个面中的下一个半边。

```
struct HE_edge
  {

      HE_vert* vert;   // vertex at the end of the half-edge
      HE_edge* pair;   // oppositely oriented adjacent half-edge
      HE_face* face;   // face the half-edge borders
      HE_edge* next;   // next half-edge around the face

  };
```



```
struct HE_face
    {

        HE_edge* edge;  // one of the half-edges bordering the face

    };
```

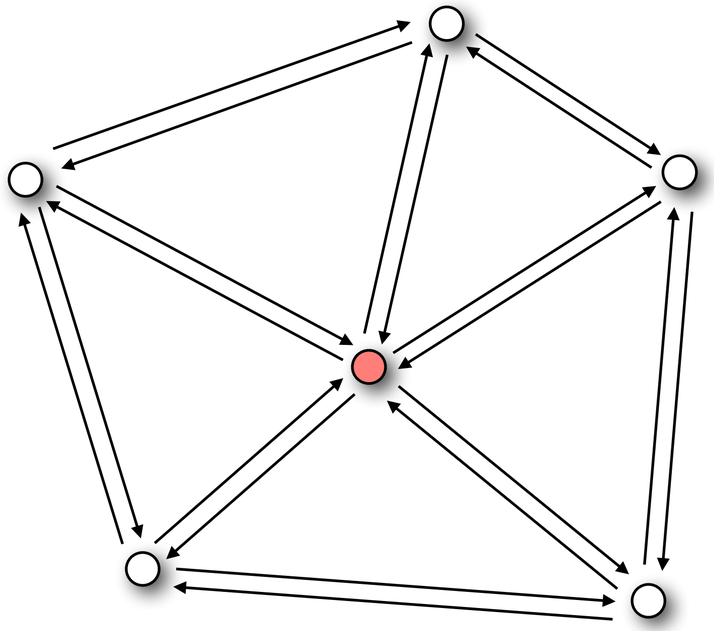
```
struct HE_vert
    {

        float x;
        float y;
        float z;

        HE_edge* edge;  // one of the half-edges emantating from the vertex

    };
```

96 to 144 B/v

no case distinctions during traversal

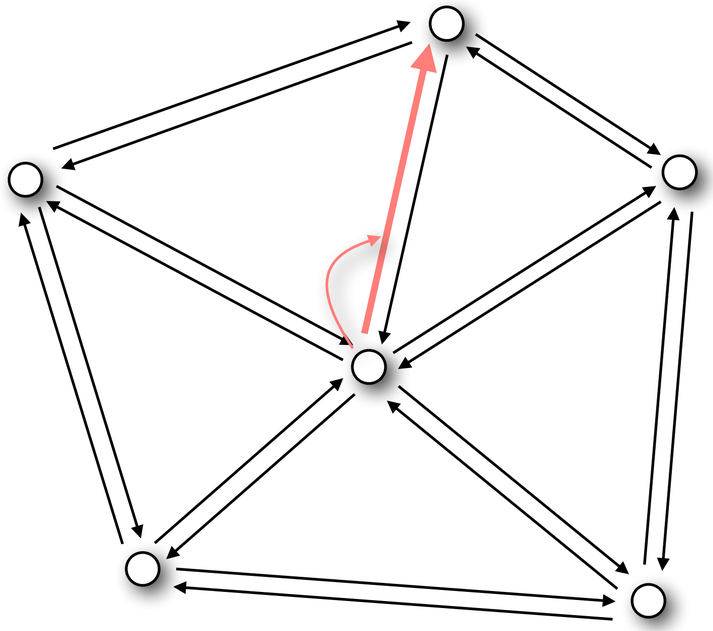
One-Ring Traversal

1. Start at vertex



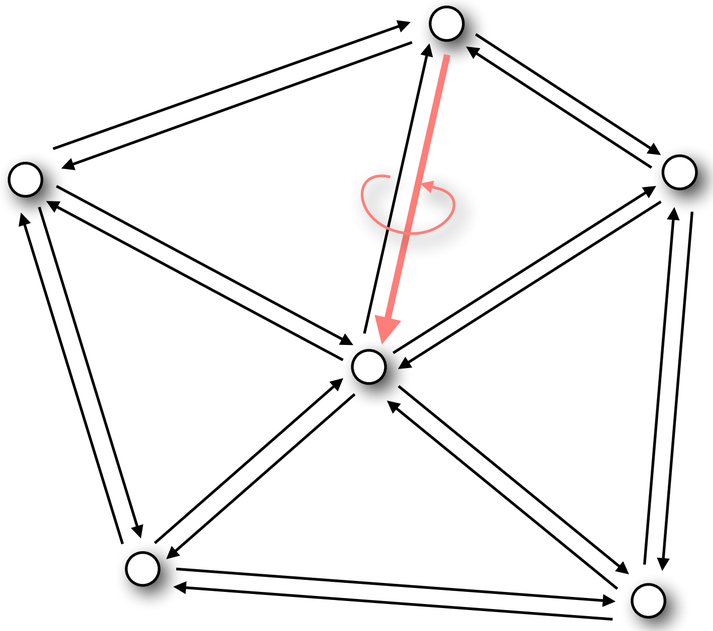
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



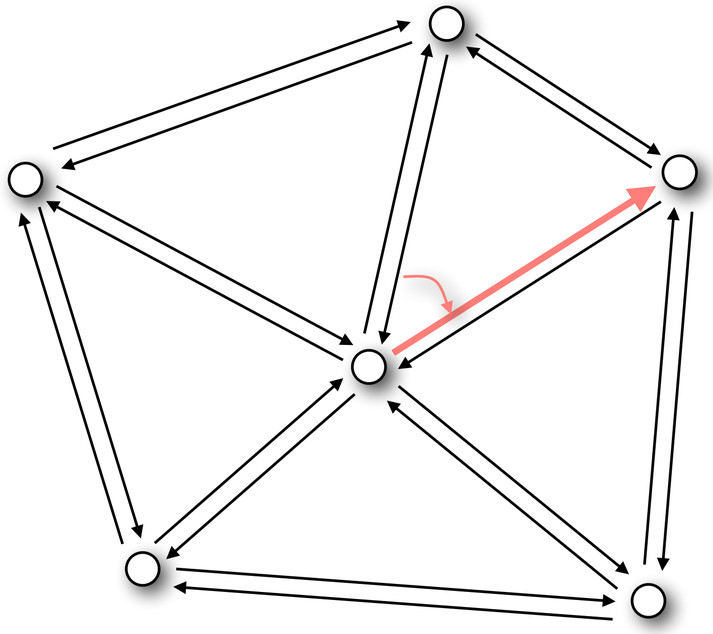
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge



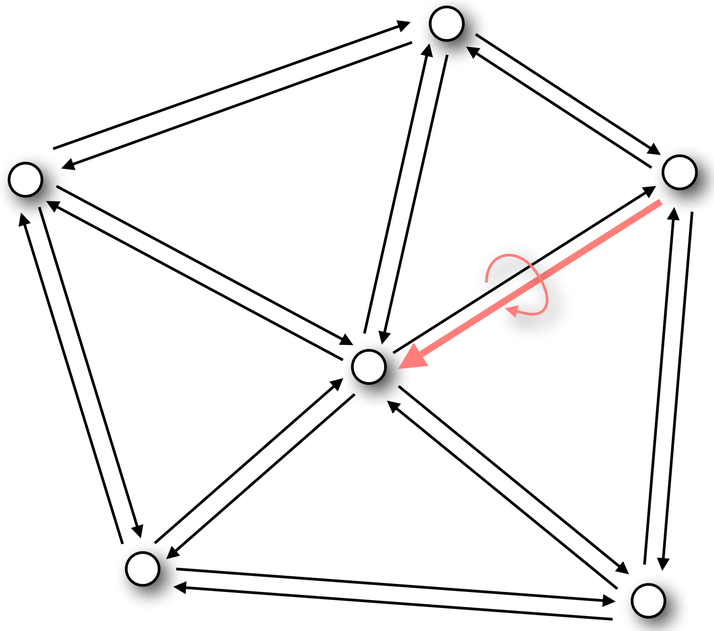
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge



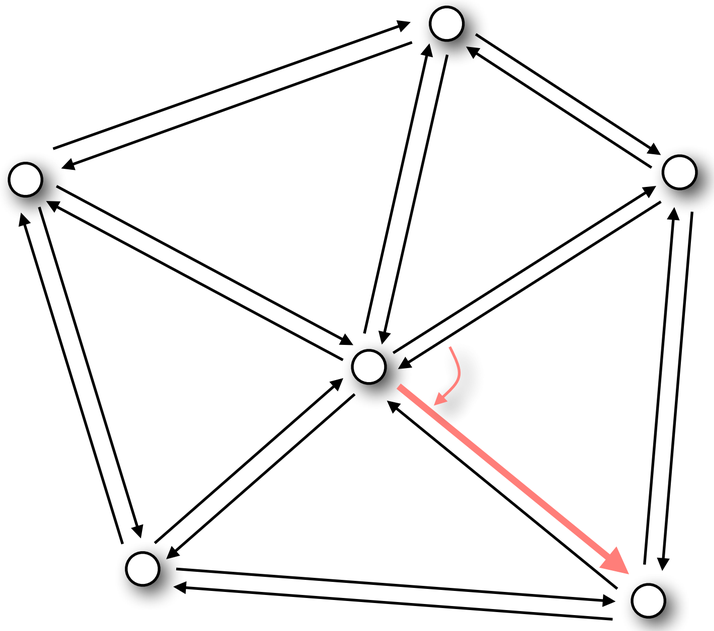
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite



One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
6. Next
7. ...

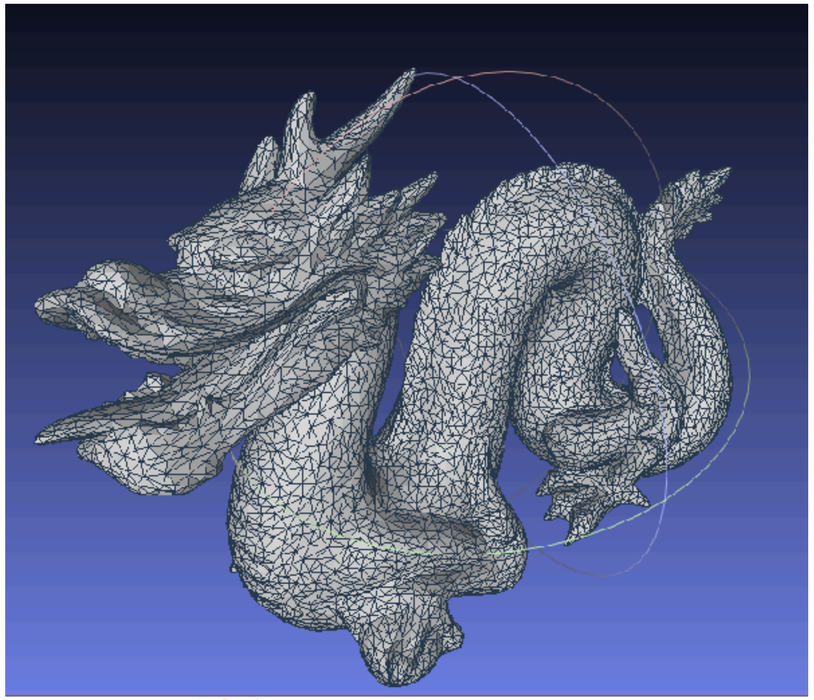


Halfedge-Based Libraries

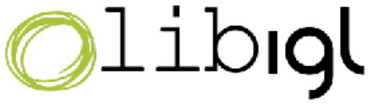
- CGAL
  - www.cgal.org
  - Computational geometry
  - Free for non-commercial use

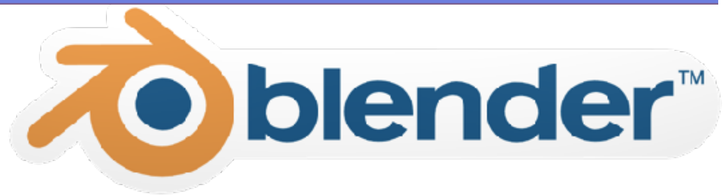
- OpenMesh
  - www.openmesh.org
  - Mesh processing
  - Free, LGPL licence

Looking at Meshes



OpenMesh





Literature

- Book #1: Chapters 2,3;
- Book #2: Chapters 1,2
- Kettner, *Using generic programming for designing a data structure for polyhedral surfaces*, Symp. on Comp. Geom., 1998
- Campagna et al., *Directed Edges - A Scalable Representation for Triangle Meshes*, Journal of Graphics Tools 4(3), 1998
- Botsch et al., *OpenMesh - A generic and efficient polygon mesh data structure*, OpenSG Symp. 2002

Representations

- Mathematical representations
  - parametric (explicit)

$$\mathbf{f} : t \mapsto \begin{pmatrix} r\cos(t) \\ r\sin(t) \end{pmatrix}, \quad \mathcal{S} = \mathbf{f}([0, 2\pi])$$

  - implicit

$$F(x, y) = x^2 + y^2 - r^2$$

$$\mathcal{S} = \{(x, y) : F(x, y) = 0\}$$

- Approximation properties
  - polynomials: approximation error

$$O(h^{p+1})$$

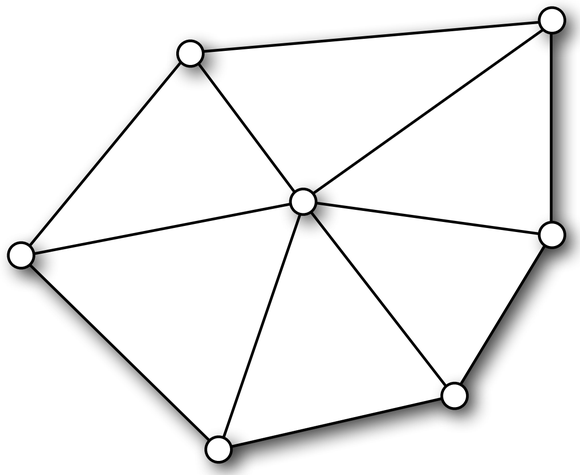
Triangle Meshes

- **Topology**: vertices, edges, triangles

$$\mathcal{V} = \{v_1, \ldots, v_n\}$$
$$\mathcal{E} = \{e_1, \ldots, e_k\} \ , \quad e_i \in \mathcal{V} \times \mathcal{V}$$
$$\mathcal{F} = \{f_1, \ldots, f_m\} \ , \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$$

- **Geometry**: vertex positions

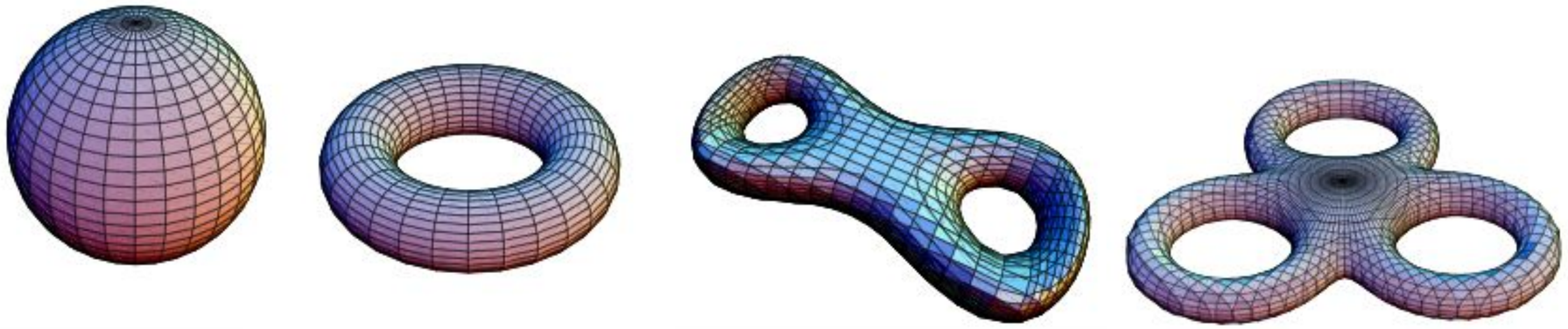
$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \ , \quad \mathbf{p}_i \in \mathbb{R}^3$$



Global Connectivity: Genus

Genus:
Maximal number of closed simple cutting curves that do not disconnect the graph into multiple components.

Informally, the number of holes or handles.



Genus 0 Genus 1 Genus 2 Genus 3

Euler-Poincare Formula

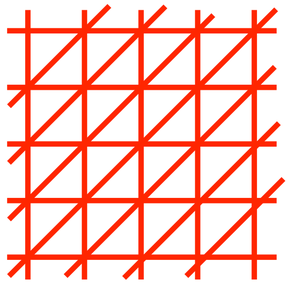
- For a closed polygonal mesh of genus *g*, the relation of the number *V* of vertices, *E* of edges, and *F* of faces is given by *Euler's formula* χ

$$V - E + F = 2(1 - g)$$

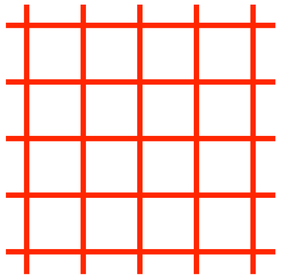
- The term $2(1 - g)$ is called the ***Euler characteristic***

Consequences of Euler Characteristics

- Triangle meshes
  - $F \approx 2V$
  - $E \approx 3V$
  - Average valence = 6



- Quad meshes
  - $F \approx V$
  - $E \approx 2V$
  - Average valence = 4

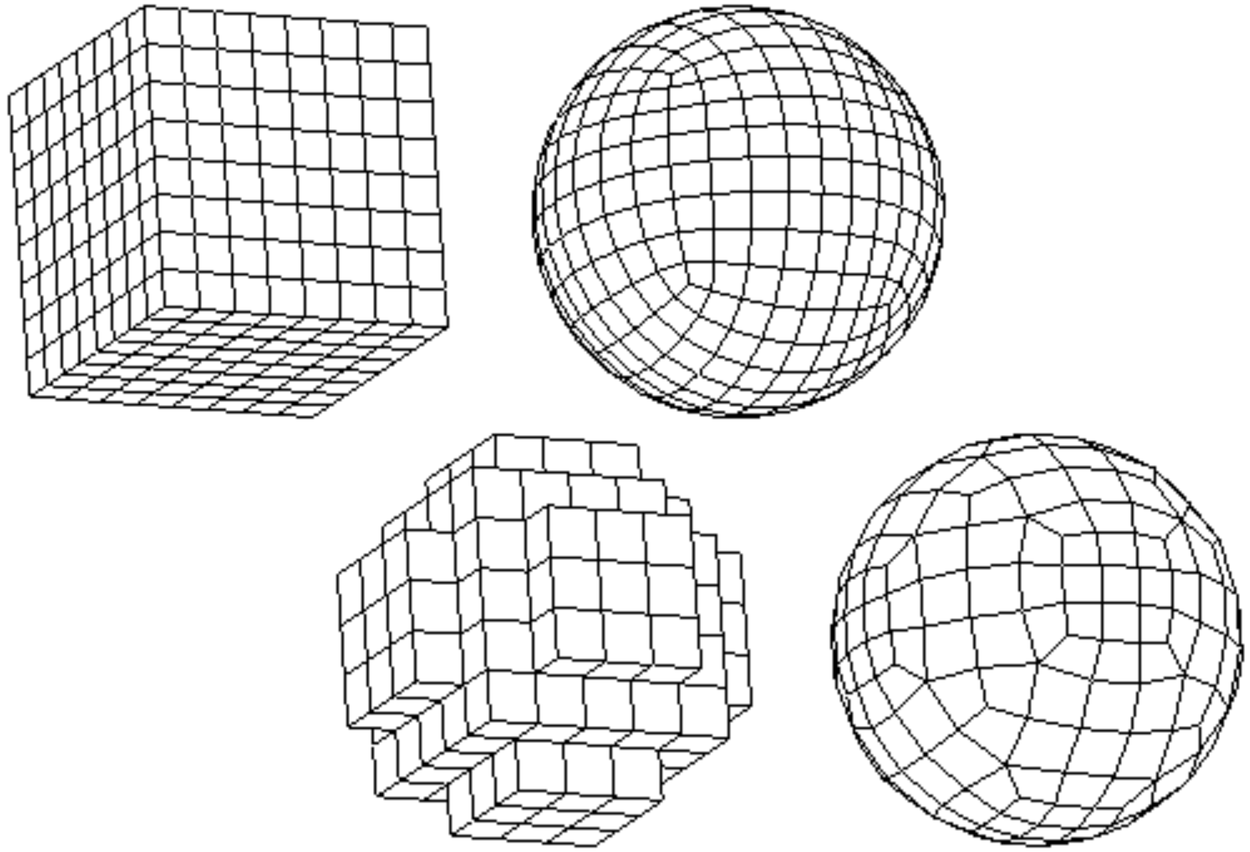


Soccer Ball

- How many Pentagons are in a soccer ball?

Any closed surface of genus 0 consisting only of hexagons and pentagons and where every vertex has valence 3 must have exactly 12 pentagons!

Consequences



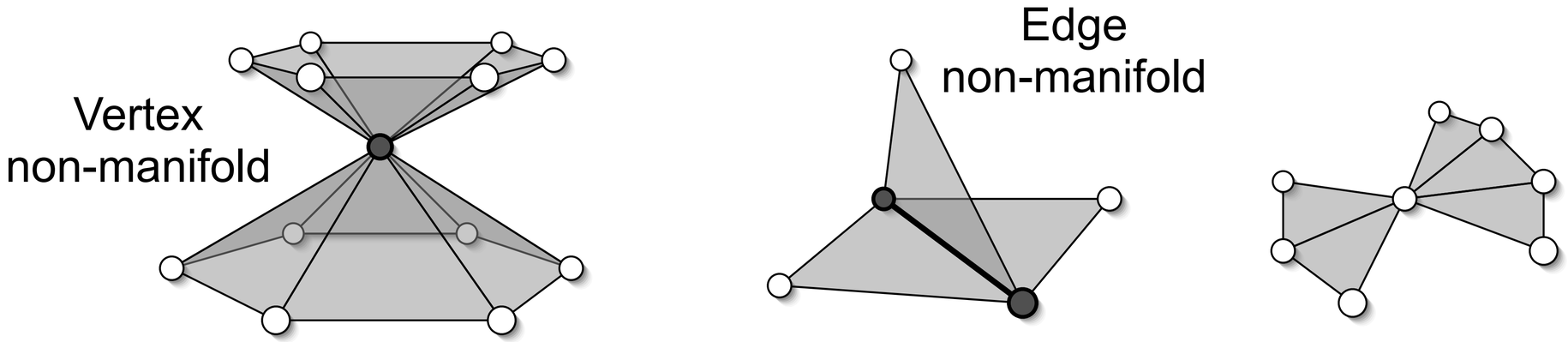
2-Manifold Surfaces

- Local neighborhoods are disk-shaped

$$\mathbf{f}(D_\varepsilon[u,v]) = D_\delta[\mathbf{f}(u,v)]$$

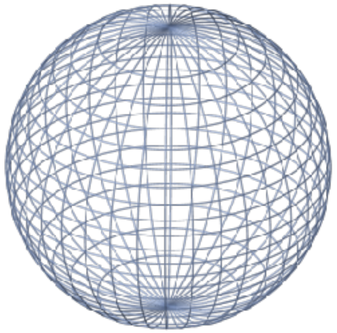
- Guarantees meaningful neighbor enumeration
  - required by most algorithms

- **Non-manifold** examples:



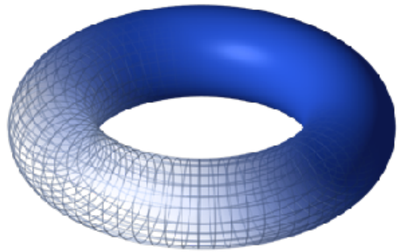
Euler Characteristic

Sphere



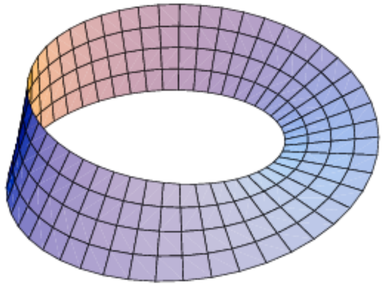
$\chi = 2$

Torus



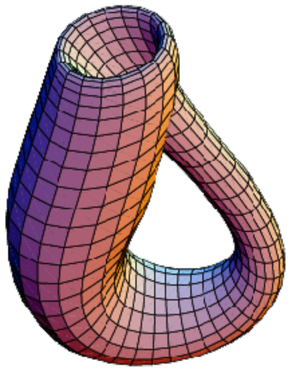
$\chi = 0$

Moebius strip



$\chi = 0$

Klein bottle



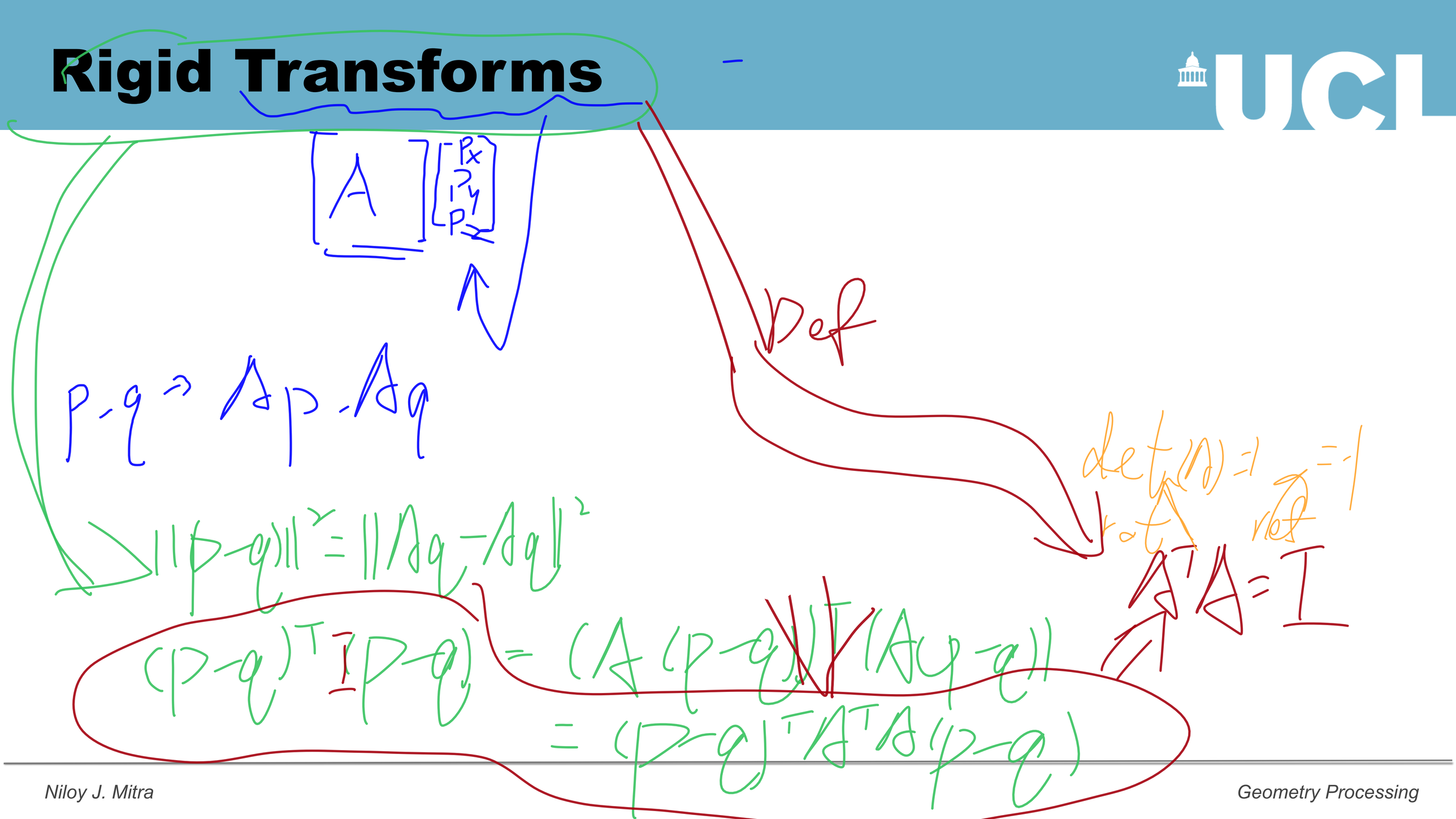
$\chi = 0$

images from Wikipedia

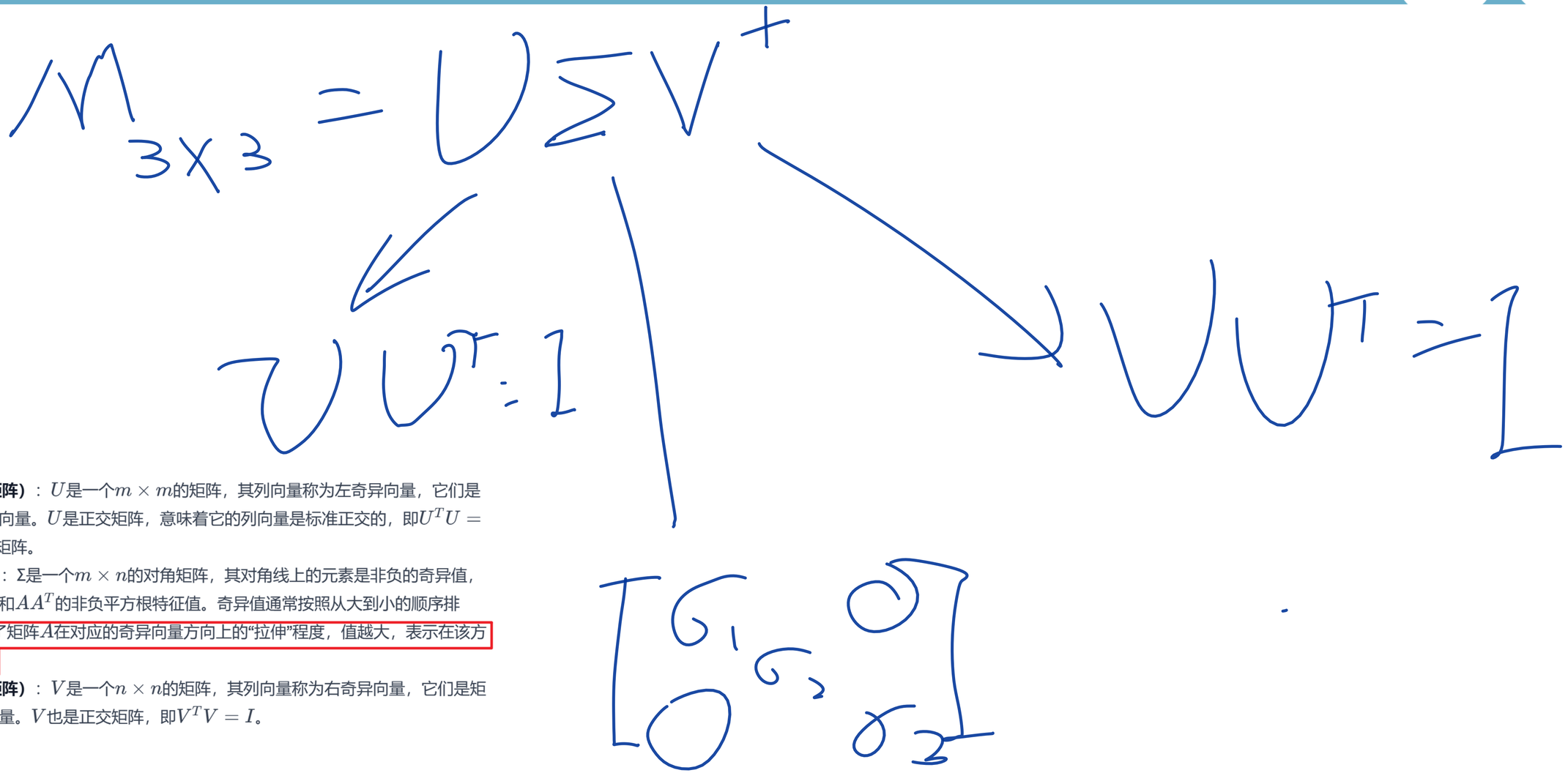
Normal Estimation for Triangle Meshes

Normal Estimation for Point Clouds

Rigid Transforms



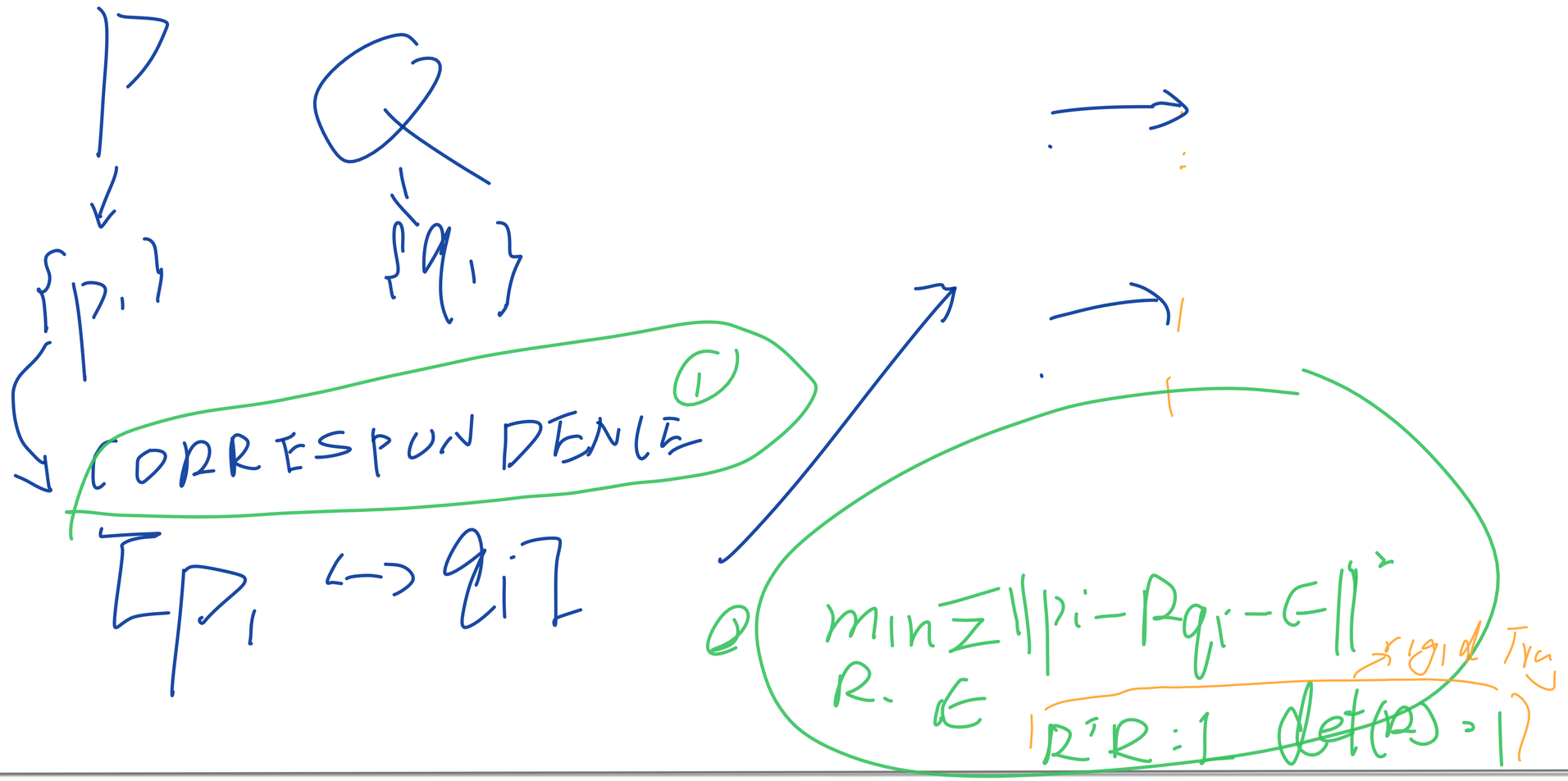
Singular Value Decomposition (SVD)



1. **U（左奇异向量矩阵）**：U是一个$m \times m$的矩阵，其列向量称为左奇异向量，它们是矩阵AA^T的特征向量。U是正交矩阵，意味着它的列向量是标准正交的，即$U^TU = I$，其中I是单位矩阵。
2. **Σ（奇异值矩阵）**：Σ是一个$m \times n$的对角矩阵，其对角线上的元素是非负的奇异值，它们是矩阵A^TA和AA^T的非负平方根特征值。奇异值通常按照从大到小的顺序排列。奇异值反映了矩阵A在对应的奇异向量方向上的“拉伸”程度，值越大，表示在该方向上的拉伸越大。
3. **V（右奇异向量矩阵）**：V是一个$n \times n$的矩阵，其列向量称为右奇异向量，它们是矩阵A^TA的特征向量。V也是正交矩阵，即$V^TV = I$。

Local Registration

- Problem Setup



Local Registration

- Objective function and Optimization

POINT TO POINT

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}$$

$$det(\mathbf{R}) = 1$$

Local Registration

- Closed form solution

FINAL SOLUTION.

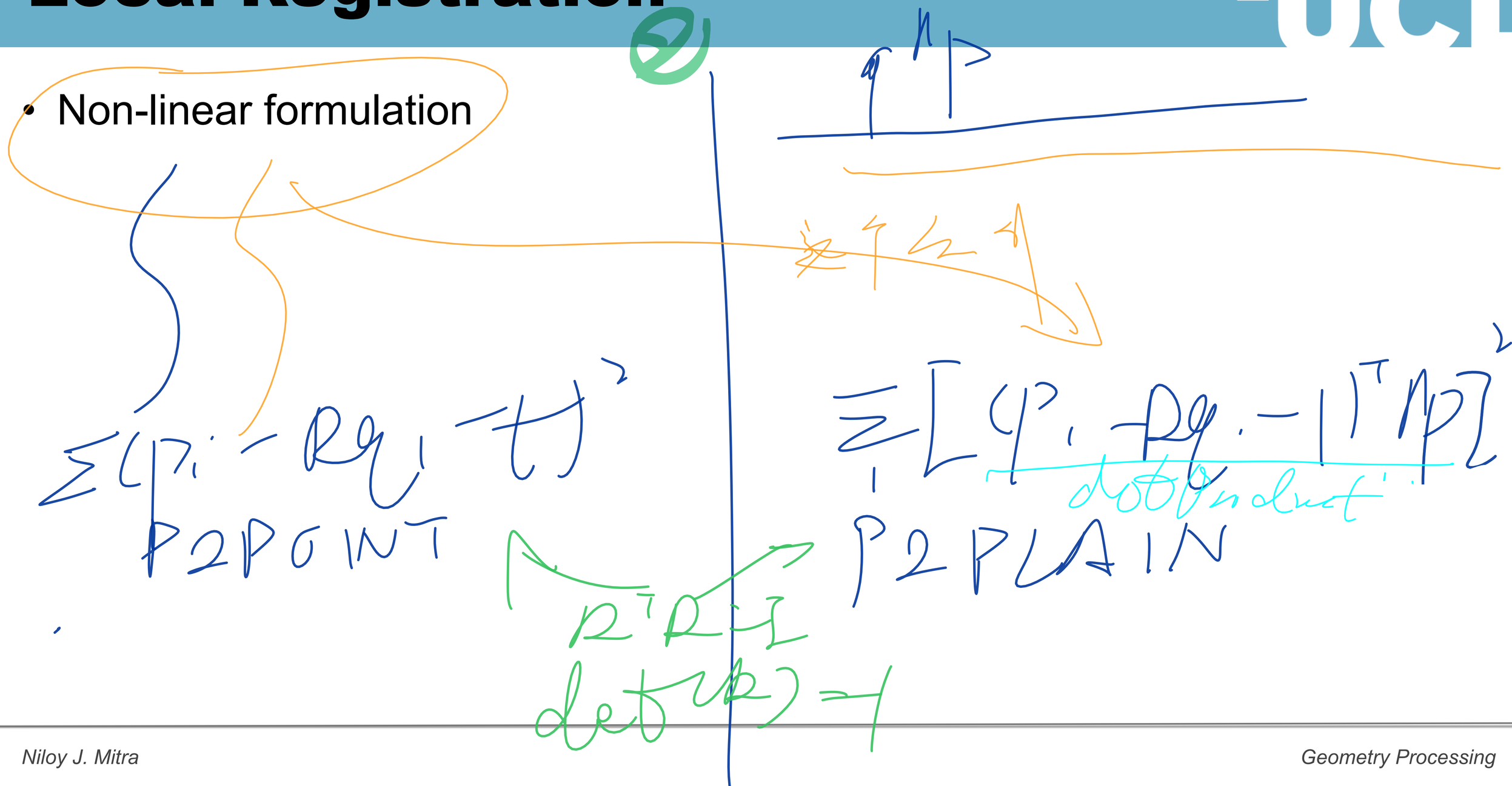
$$\rightarrow \begin{cases} \bar{p} = \dfrac{\Sigma p_i}{n} \\ \bar{q} = \dfrac{\Sigma q_i}{n} \end{cases}$$

$$\rightarrow \text{SVD}\left(\sum_i \tilde{q}_i \tilde{p}_i^T\right) = \text{SVD}\left(\tilde{Q}\tilde{P}^T\right) = U \Sigma V^T$$

$$\rightarrow \boxed{\begin{aligned} R &= V \begin{bmatrix} 1 & & \\ & 1 & \\ & & \det(VU^T) \end{bmatrix} U^T \\ t &= \bar{p} - R\bar{q} \end{aligned}}$$

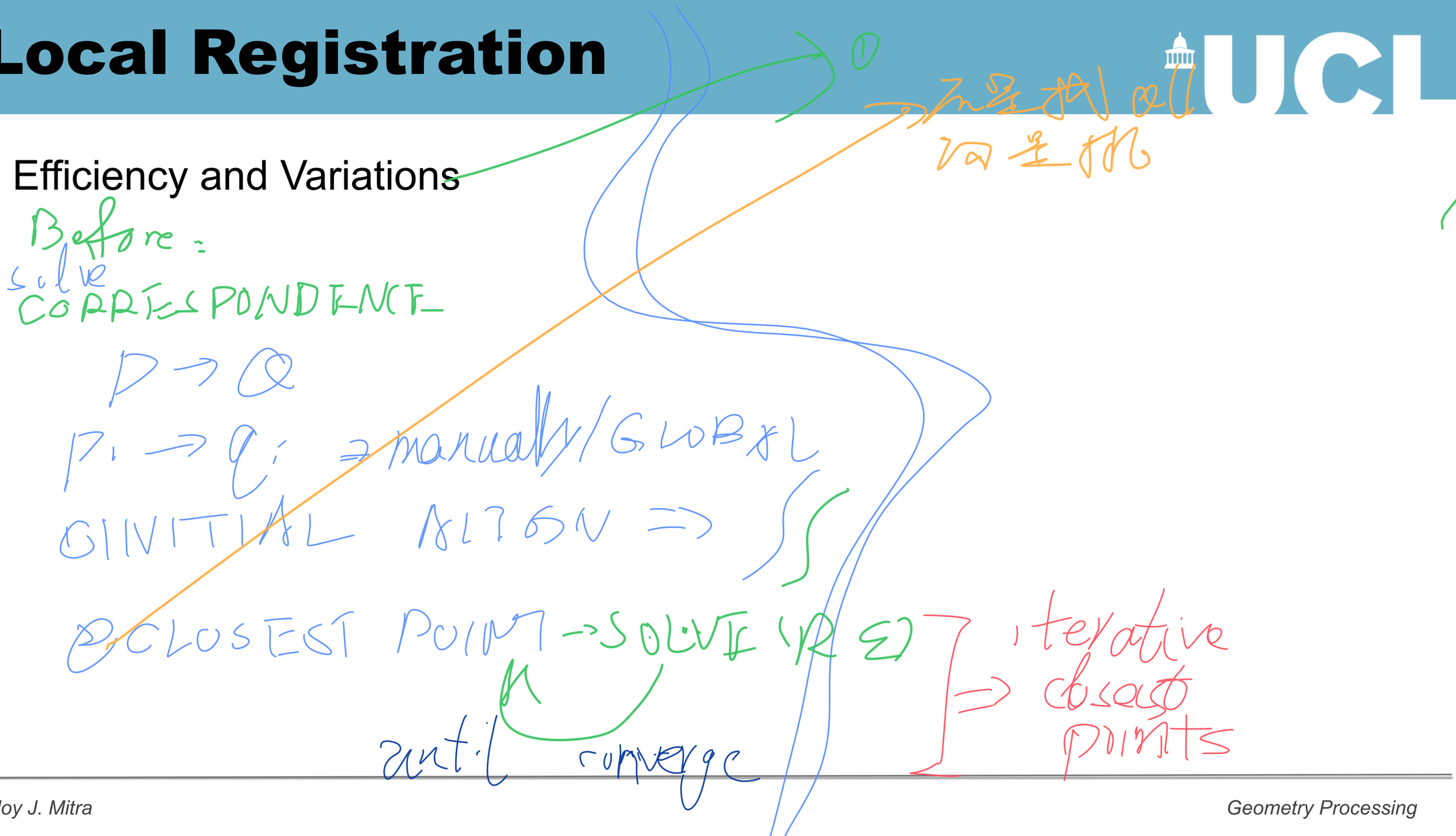
Local Registration

- Non-linear formulation

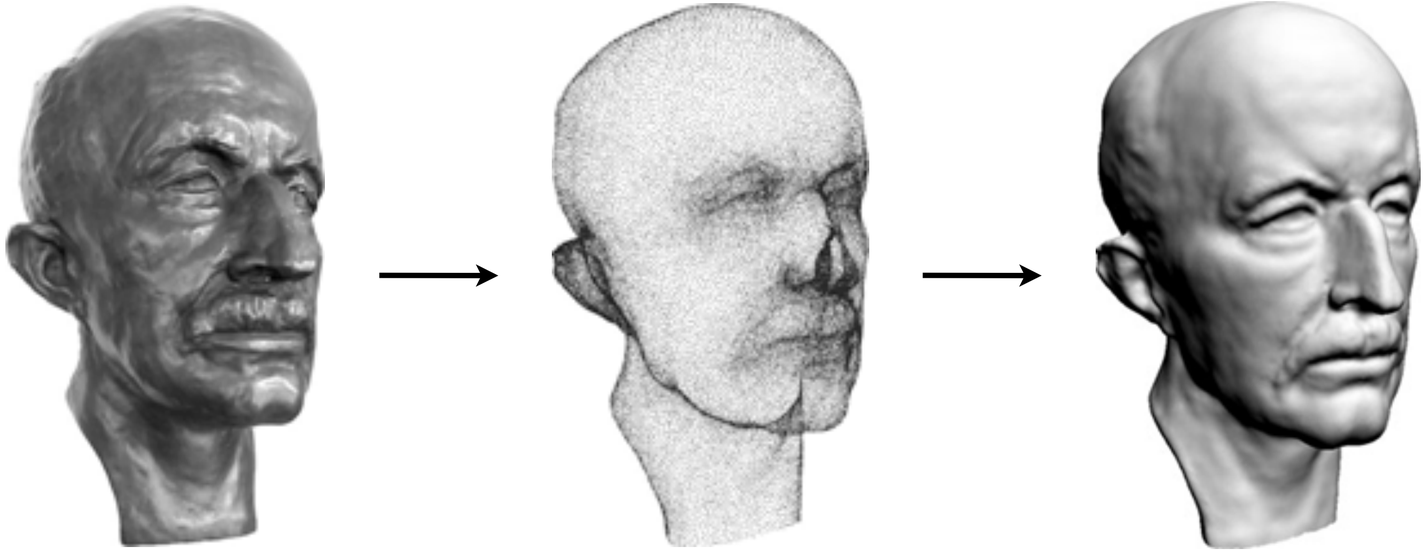


Local Registration

- Efficiency and Variations



Surface Reconstruction

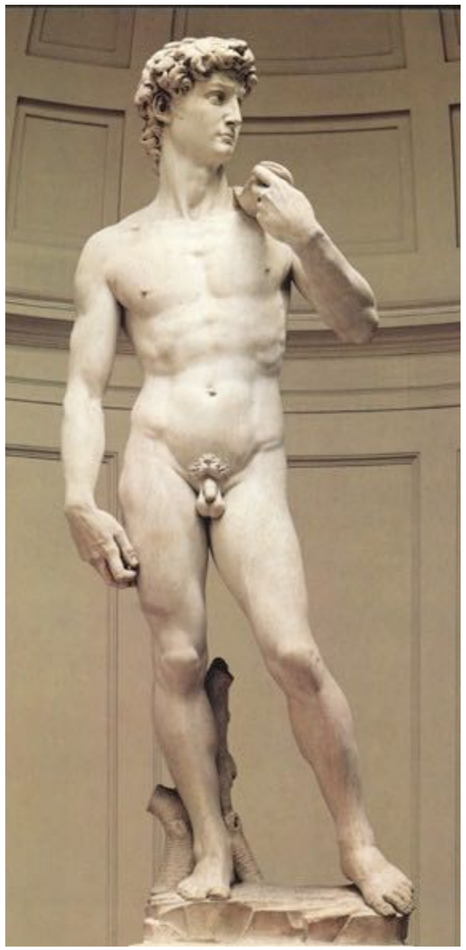


physical model

acquired point cloud

reconstructed model

Digital Michelangelo




1G sample points → 8M triangles


4G sample points → 8M triangles

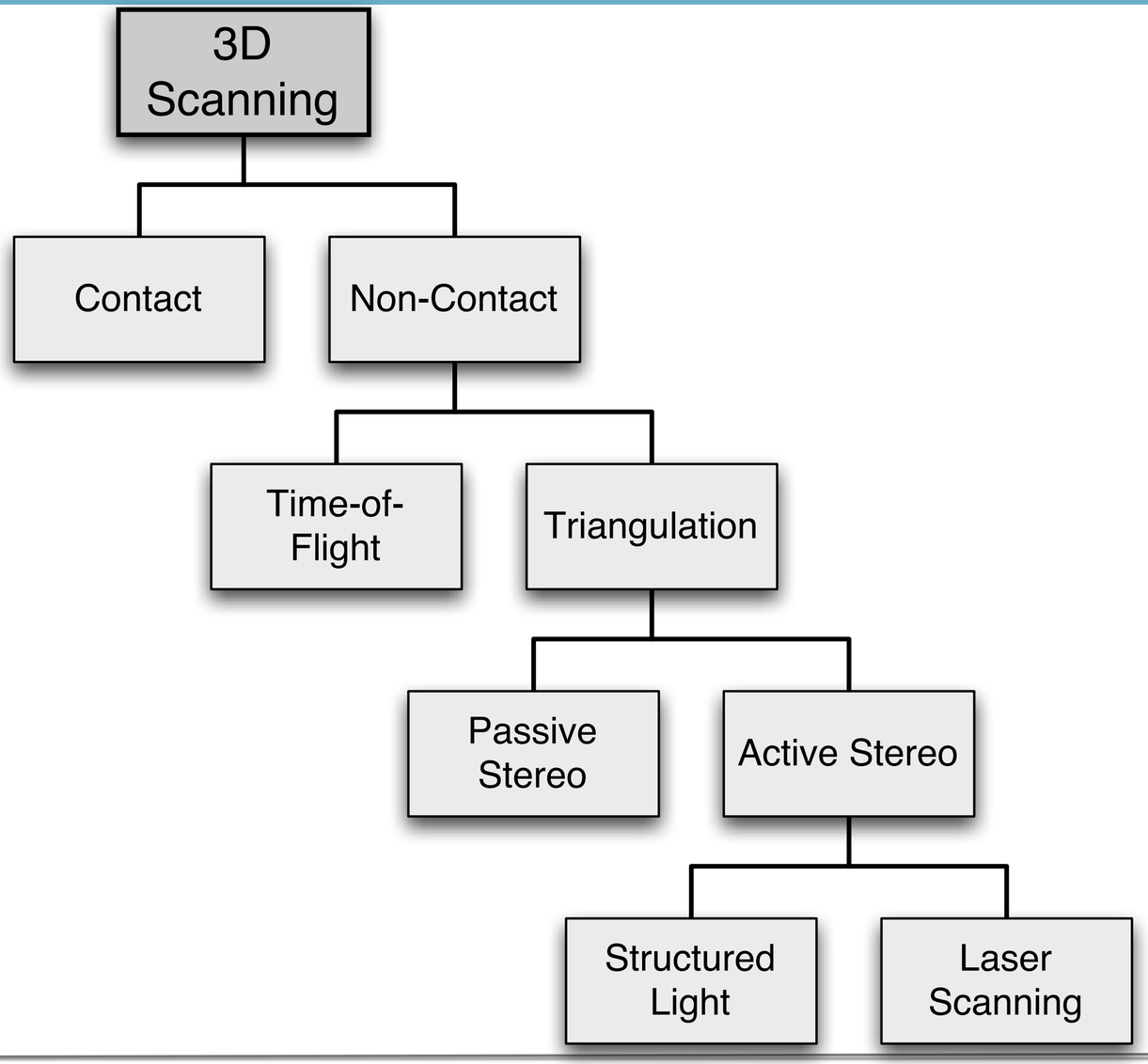
Outline

- 3D Scanning Techniques
- Scan Registration
- Explicit Reconstruction
  - from point clouds
  - from range scans
- Implicit Reconstruction
  - from point clouds
  - from range scans

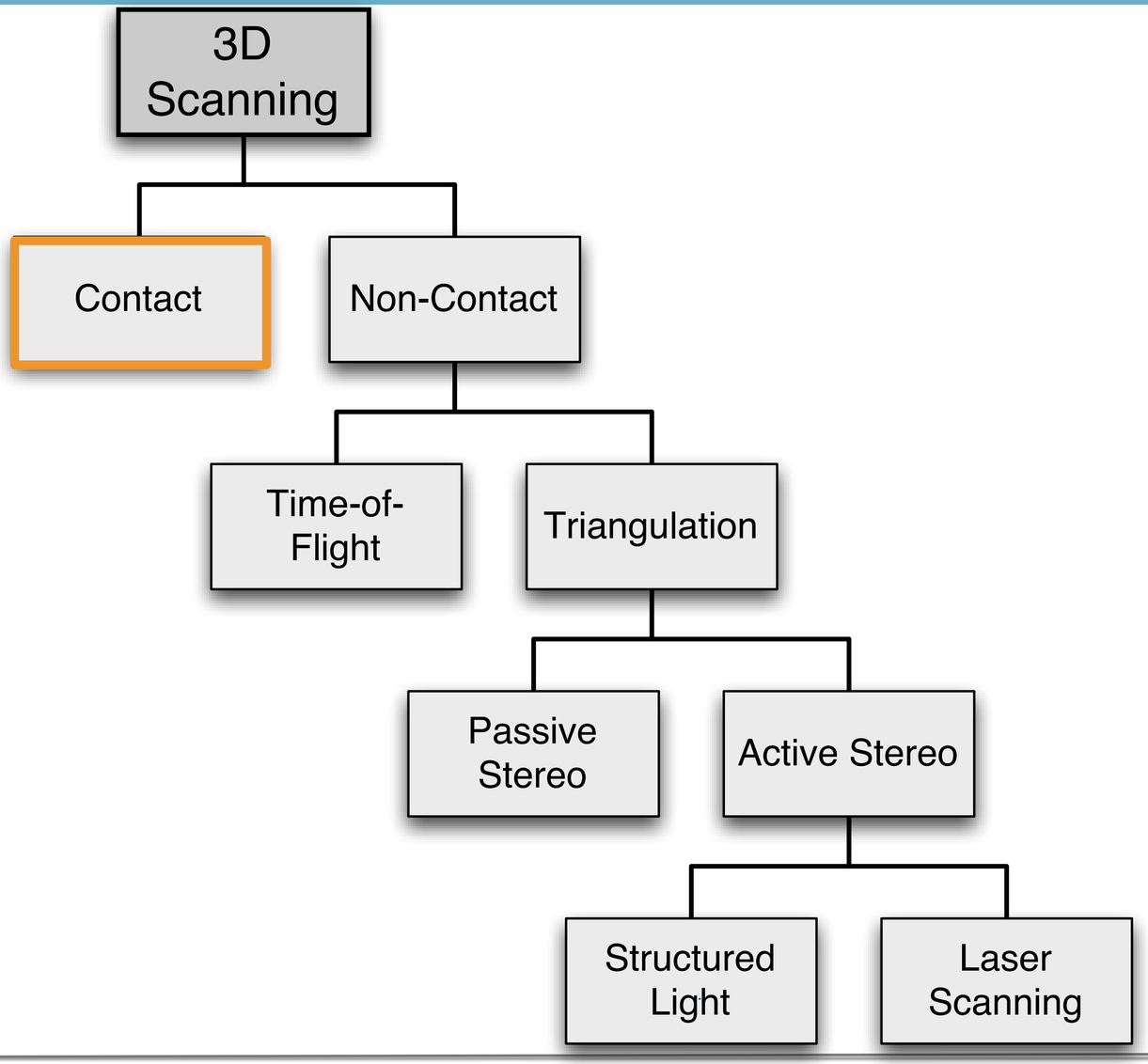
Outline

- **3D Scanning Techniques**
- Scan Registration
- Explicit Reconstruction
  - from point clouds
  - from range scans
- Implicit Reconstruction
  - from point clouds
  - from range scans

3D Scanning Techniques



3D Scanning Techniques



Contact Scanners

- Probe object by physical touch
  - Can be highly accurate
  - Often used in manufacturing control
  - Slow scanning, sparse set of samples

[Zeiss]

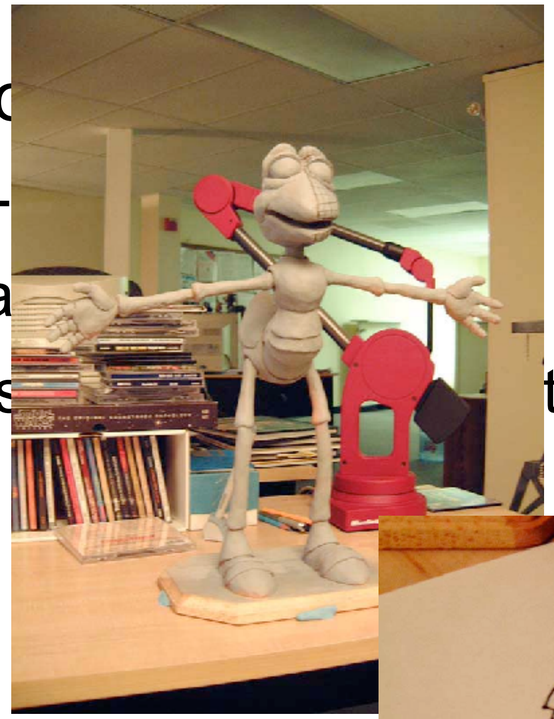
Contact Scanners

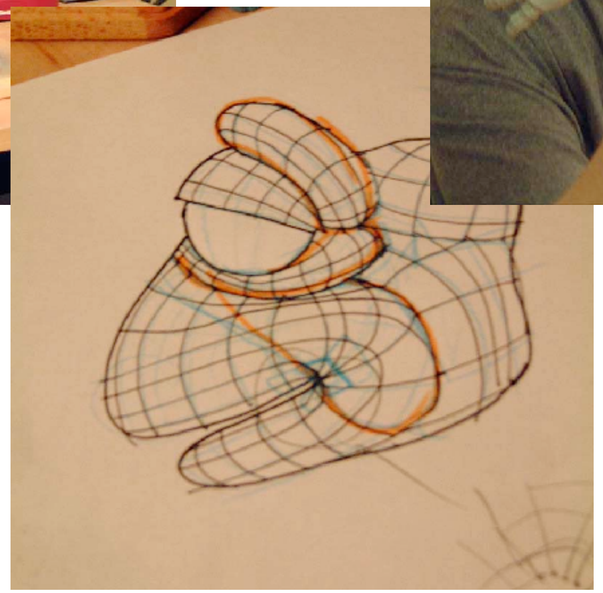
- Probe object by physical touch
  - Hand-held scanners
  - Less accurate
  - Slow scanning, sparse set of samples

[Immersion Microscribe]

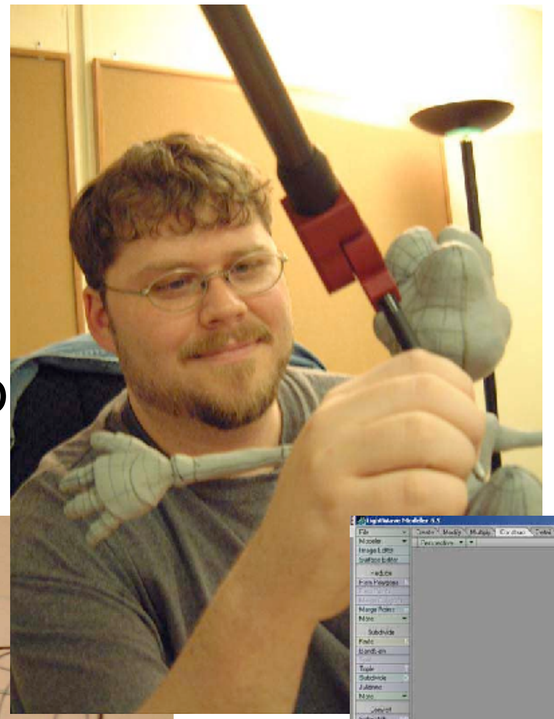
Contact Scanners

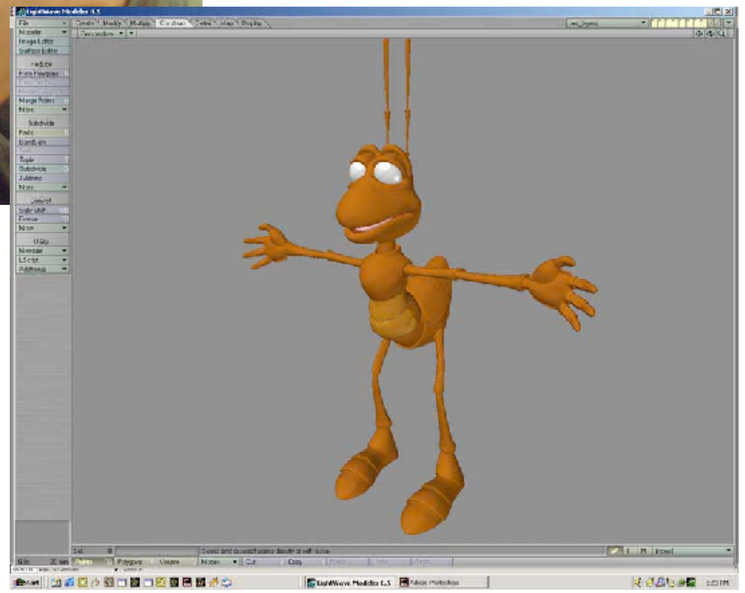
- Probe c touch
  - hand-
  - less a
  - slow s t of samp



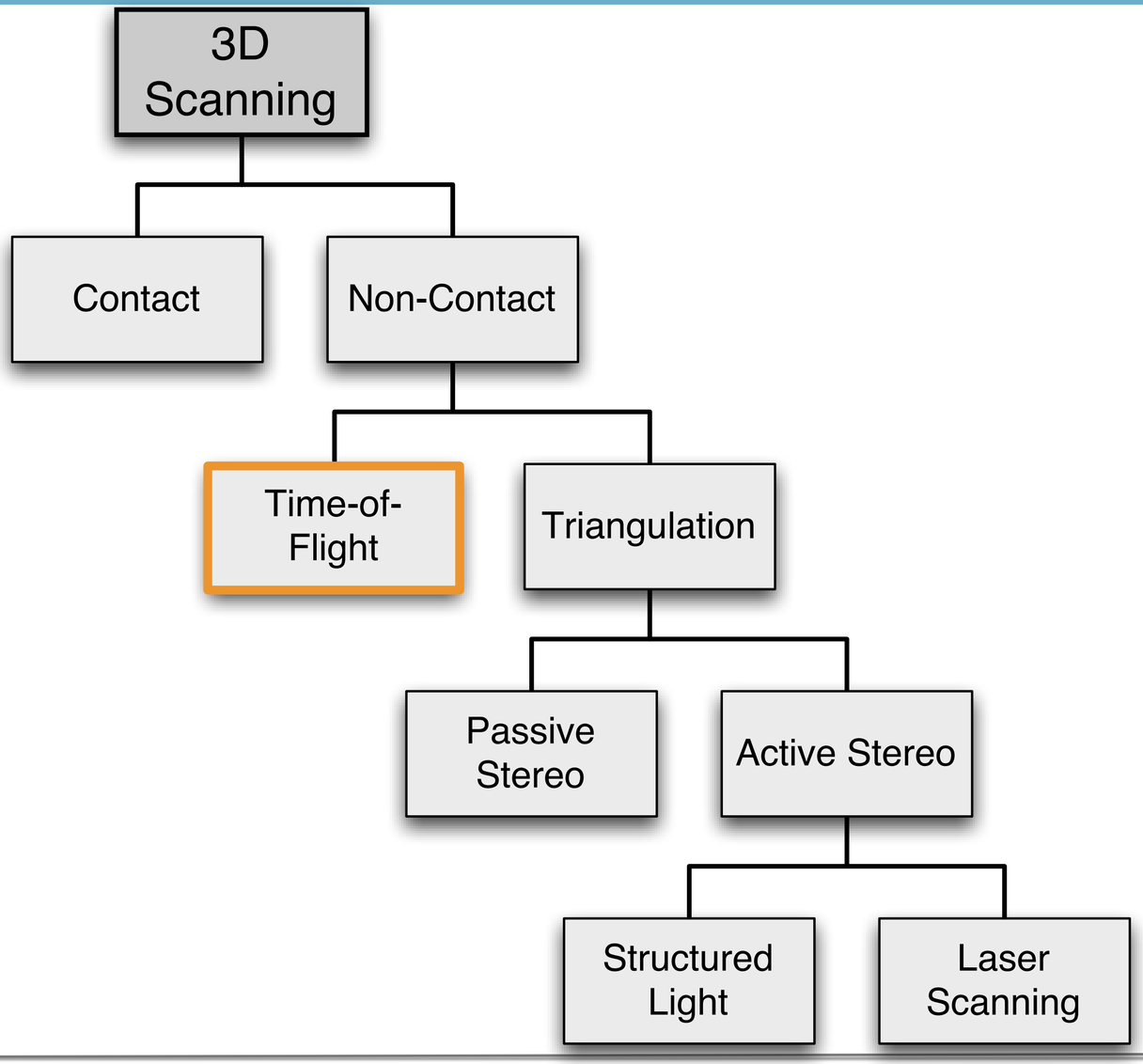


[Immersion, Magnetic Dreams]





3D Scanning Techniques



Time-of-Flight Scanners

- Probe object by laser or infrared light
  - Emit pulse of light, measure time till reflection from surface is “seen” by a detector
  - Known speed of light and round-trip time allows to compute distance to surface
- **LiDAR**
  - **Li**ght **D**etection **a**nd **R**anging
  - Good for long distance scans
  - 6mm accuracy at 50m distance

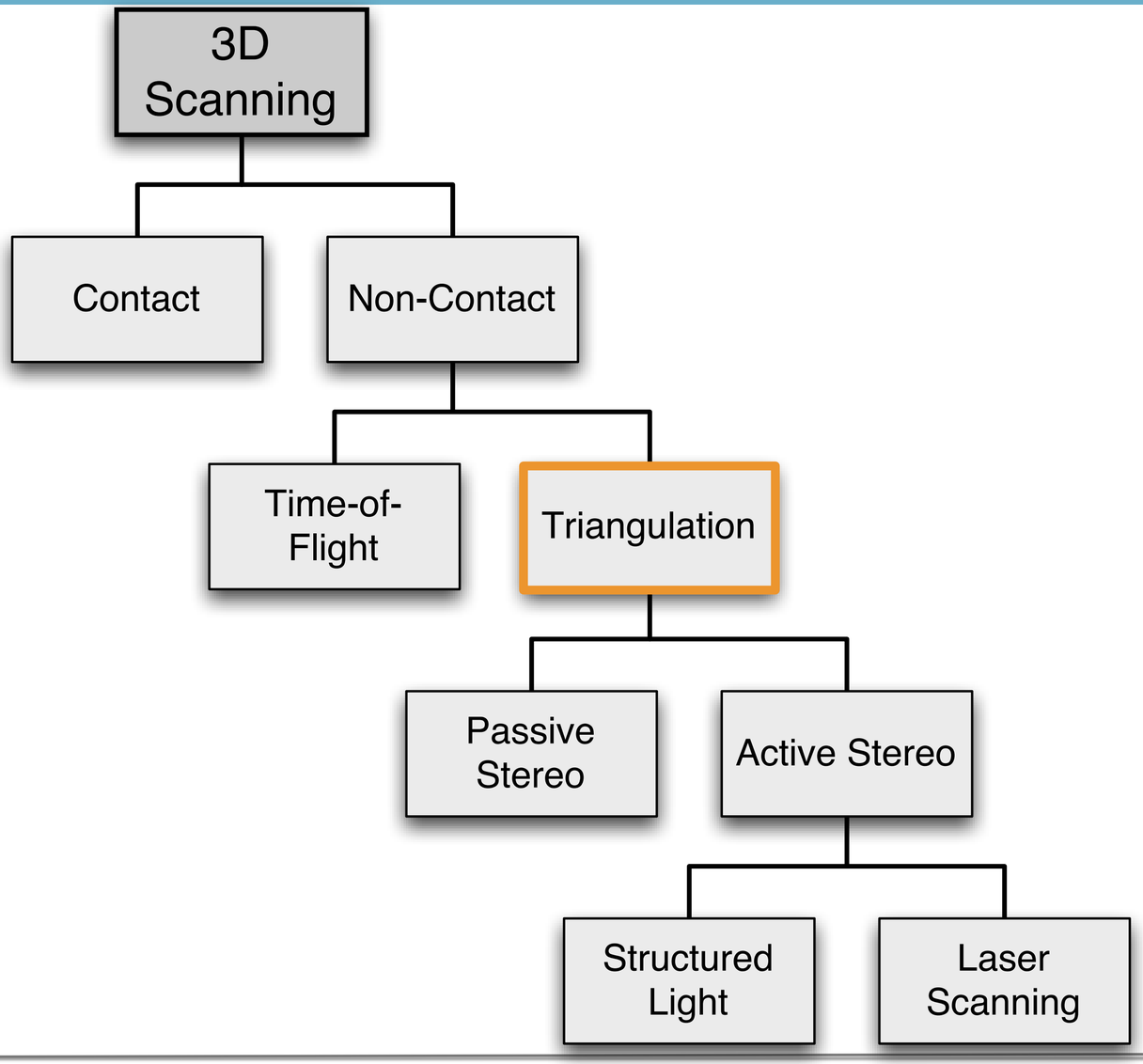
[Leica]

Time-of-Flight Scanners

- Probe object by laser or infrared light
  - Emit pulse of light, measure time till reflection from surface is “seen” by a detector
  - Known speed of light & round-trip time allows to compute distance to surface
- **Infrared light**
  - 176×144 pixels, up to 50fps
  - 30cm - 5m distance
  - 1cm accuracy

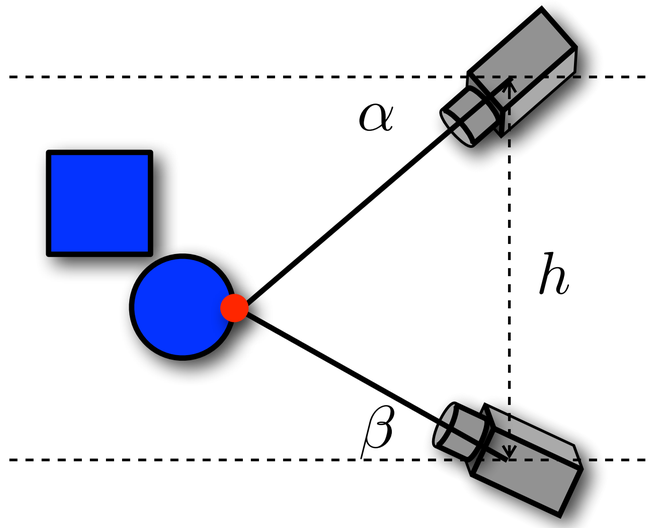
[Mesa Imaging]

3D Scanning Techniques



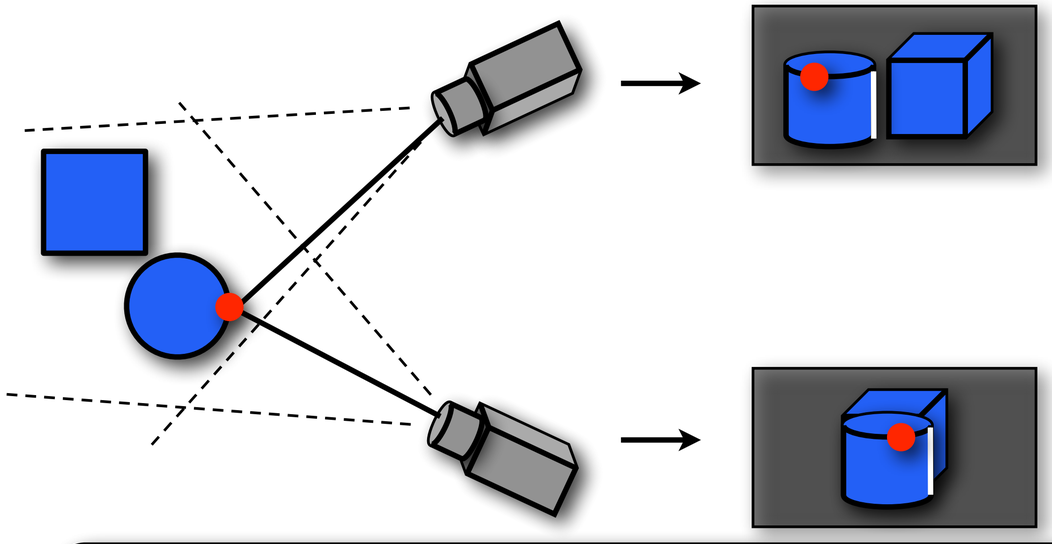
Triangulation Principle

- Two “cameras” identify the same 3D point
  - Compute depth from angles and baseline



Passive Stereo Matching

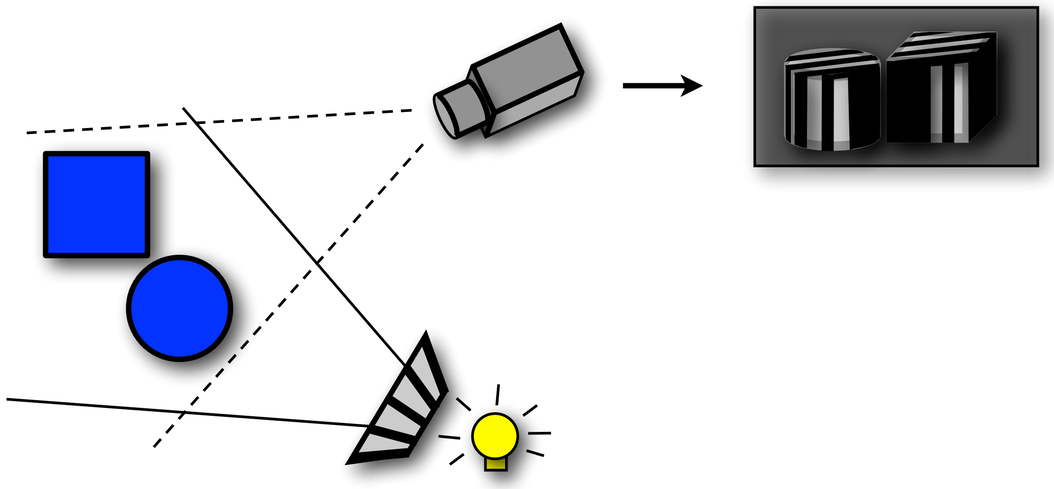
- Find and match features in both images



+ No need for active light/laser
- Needs color features to match
- Sparse and noisy samples

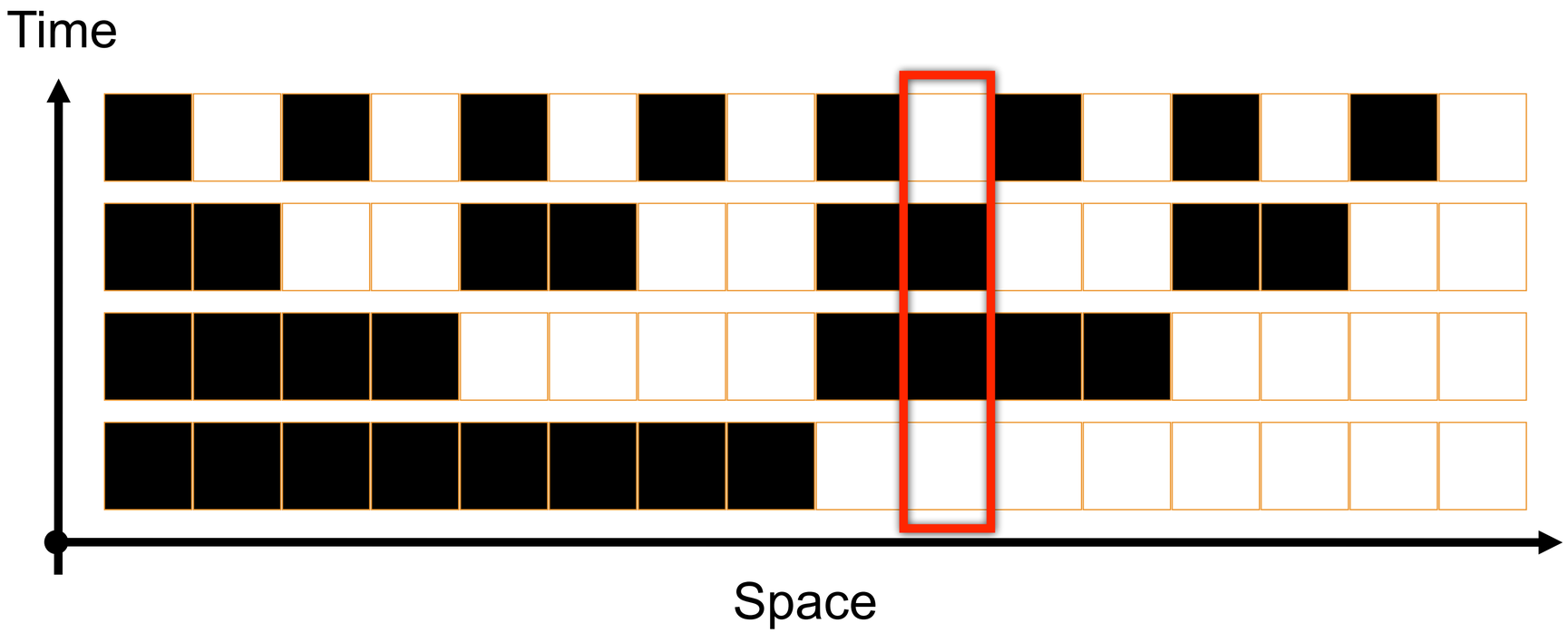
Structured Light Scanner

- Project special b/w patterns to identify pixels



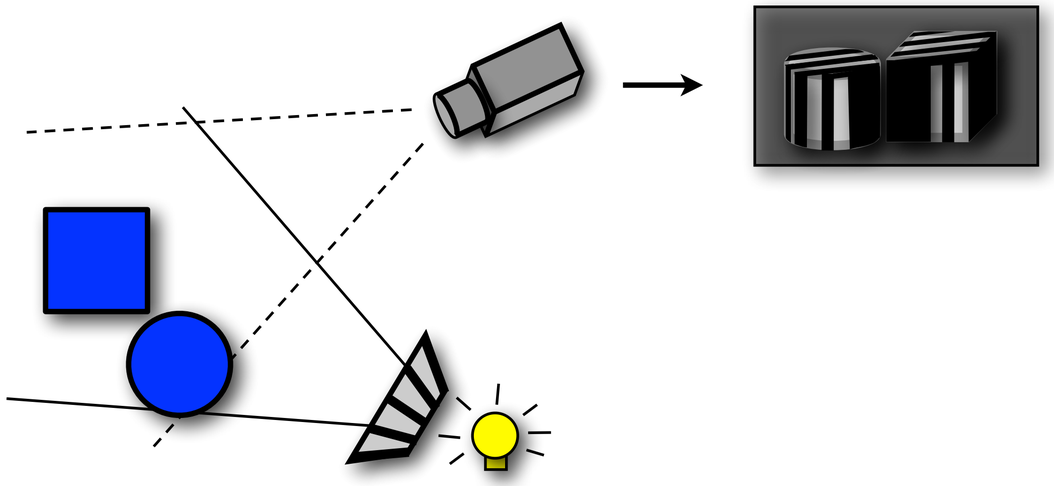
Time-Coded Light Patterns

- Assign each stripe a unique light code
  - Project several b/w patterns over time
  - Color pattern identifies row/column



Structured Light Scanner

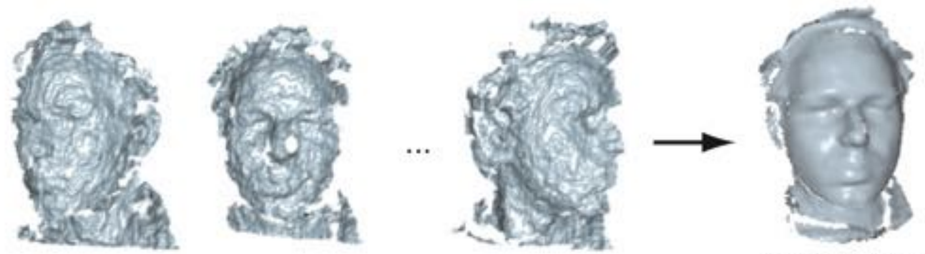
- Project special b/w patterns to identify pixels



+ Quite accurate (<1mm)
- Problematic for textured materials
- Needs several images (slow)

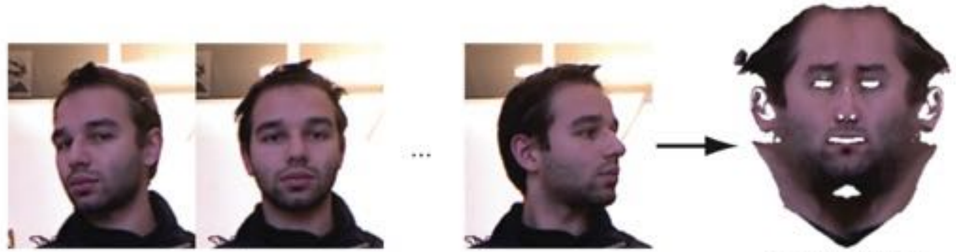
Microsoft Kinect

- Invisible infrared pattern



Kinect Raw Depth Maps

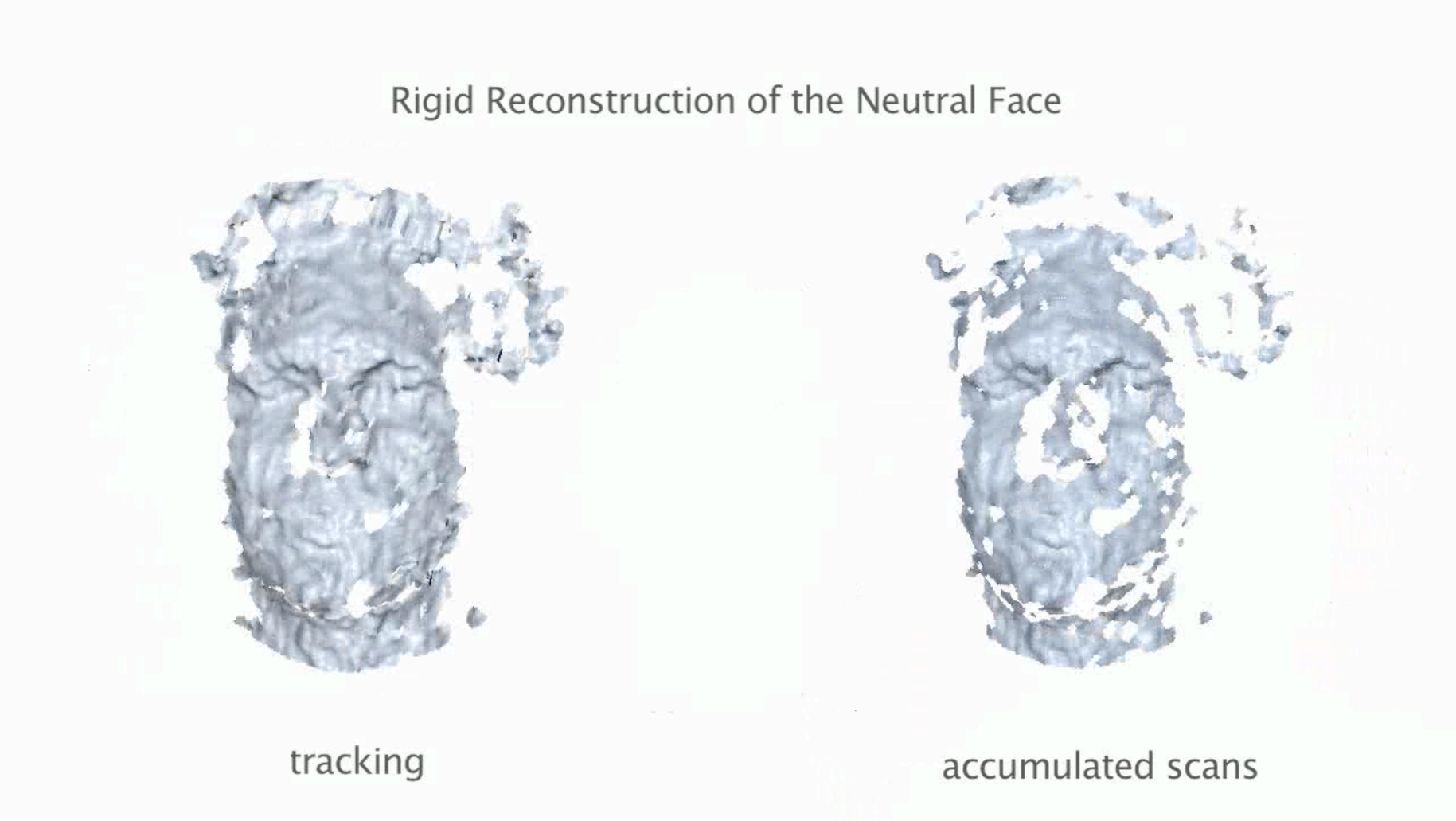
Accumulated
3D Model



Kinect Raw Images

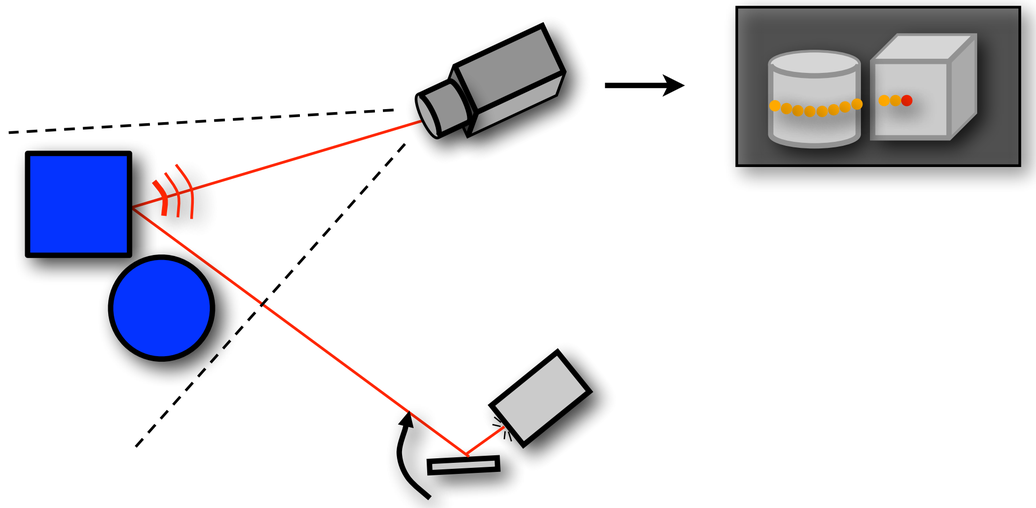
Accumulated
Texture

Scanning Faces



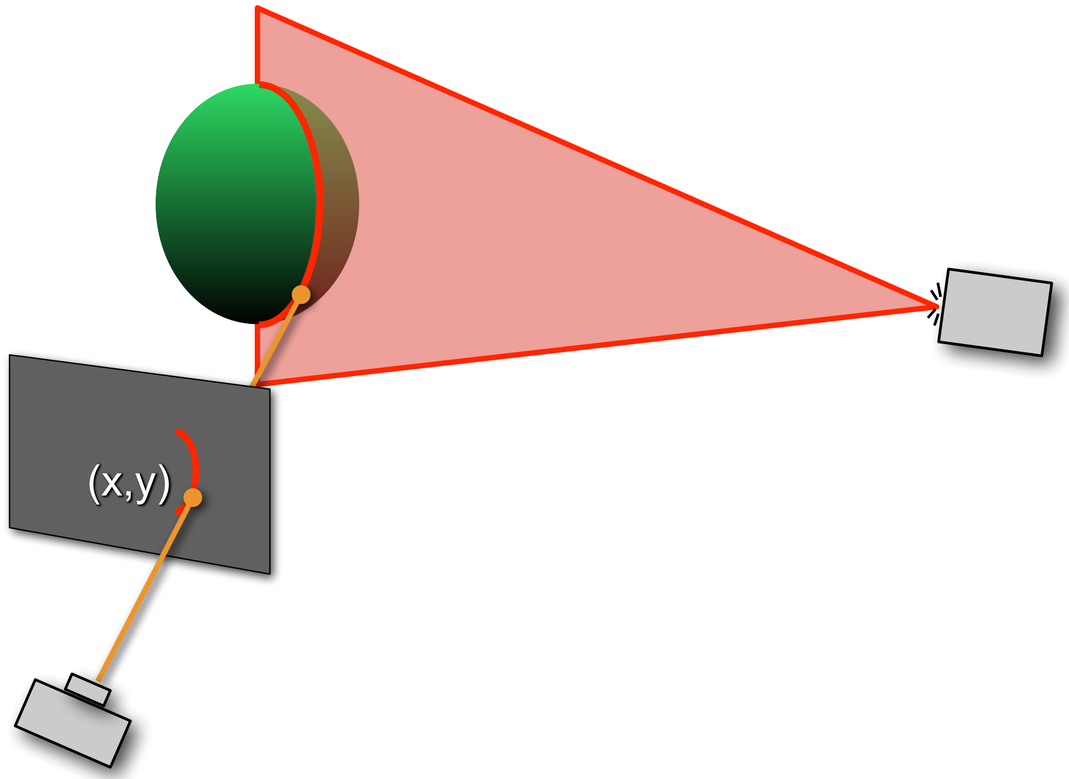
Laser Scanning

- Sweep laser, record when pixel intensity is maximum

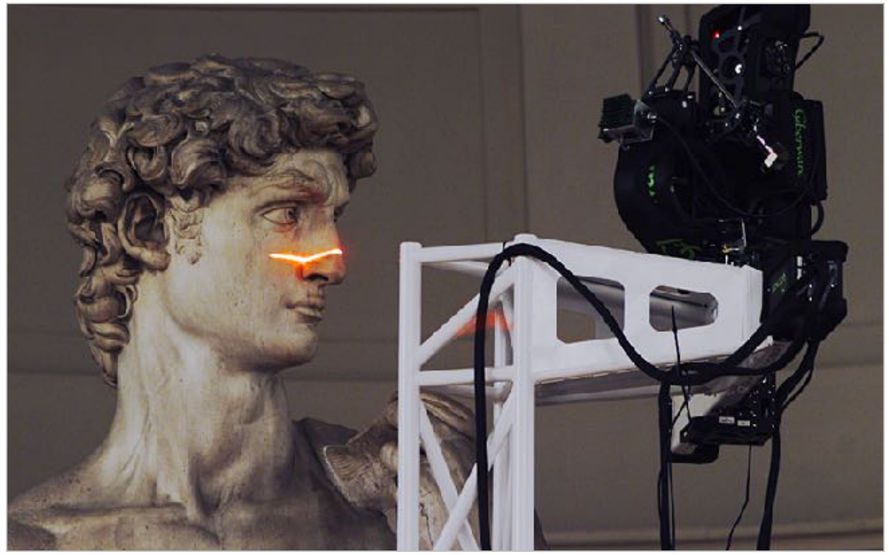


3D Laser Scanning

- Project laser stripe onto object
- Get depth by ray-plane intersection



Laser Scanning



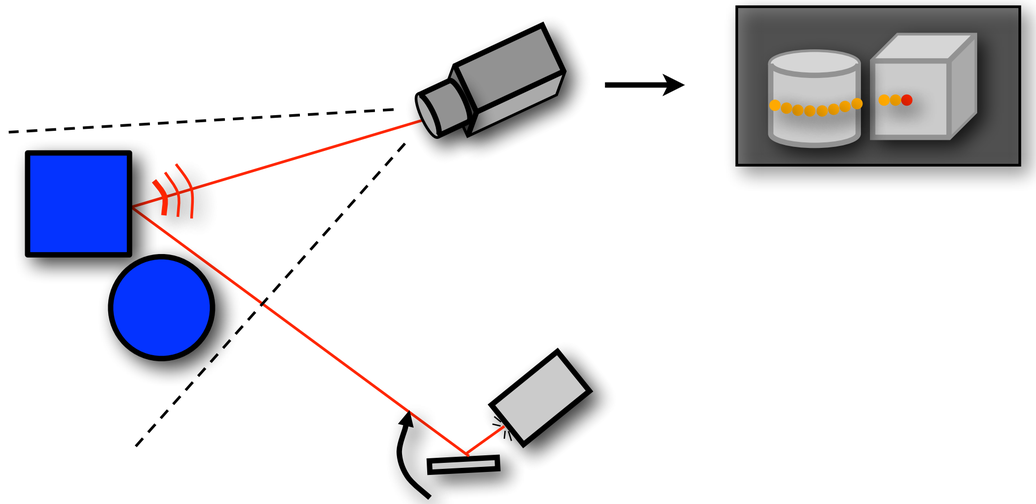
[Digital Michelangelo Project]

[Cyberware]

[Minolta]

Laser Scanning

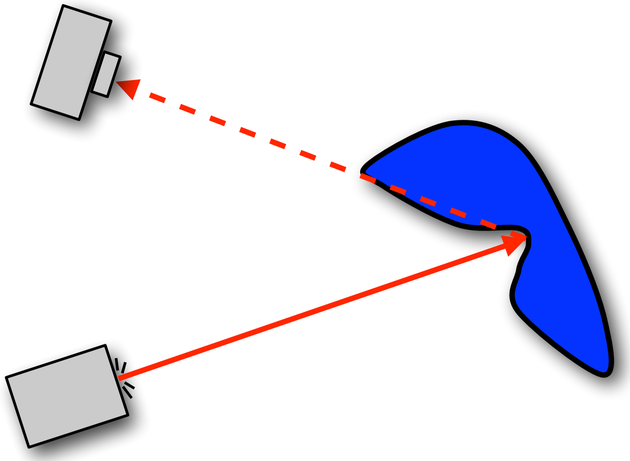
- Sweep laser, record when pixel intensity is max.



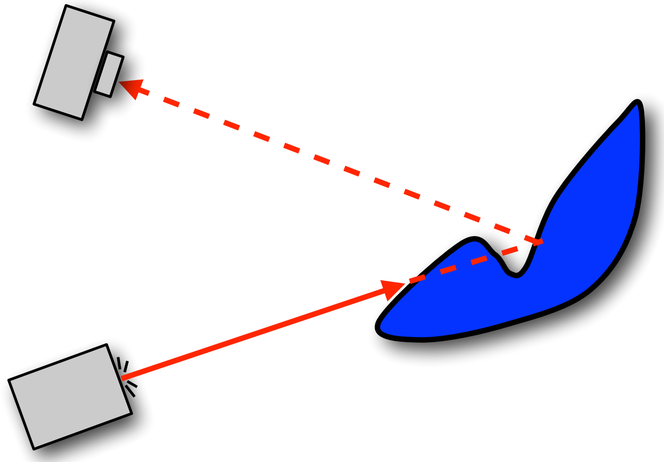
+ Quite accurate (<1mm)
- Problematic for difficult reflectance properties
- Laser sweeping is rather slow

Triangulation Problems

- Occlusion for concave regions



sensor cannot see lit point



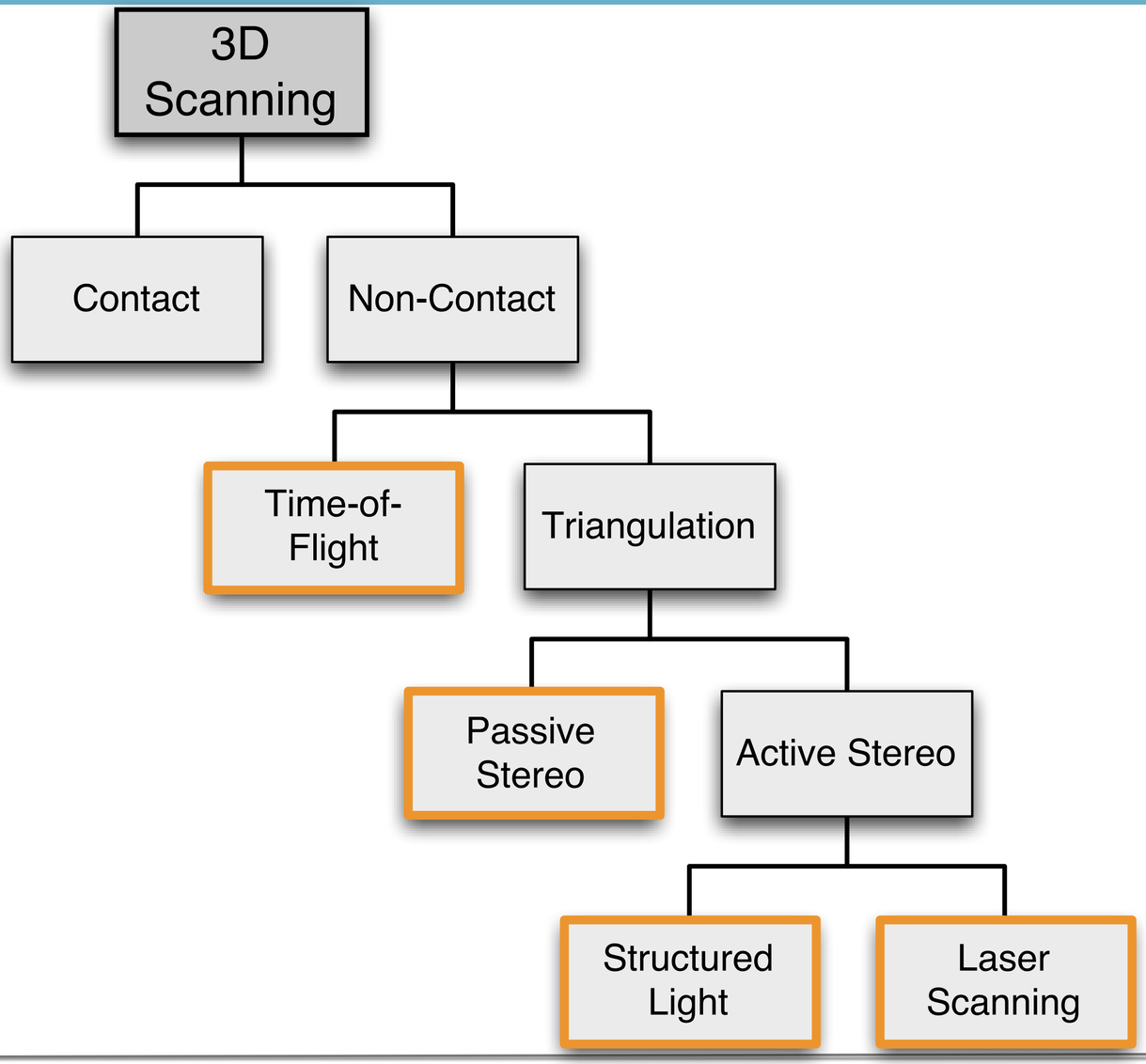
point in shadow

- General trade-off

longer baseline
more shadowing

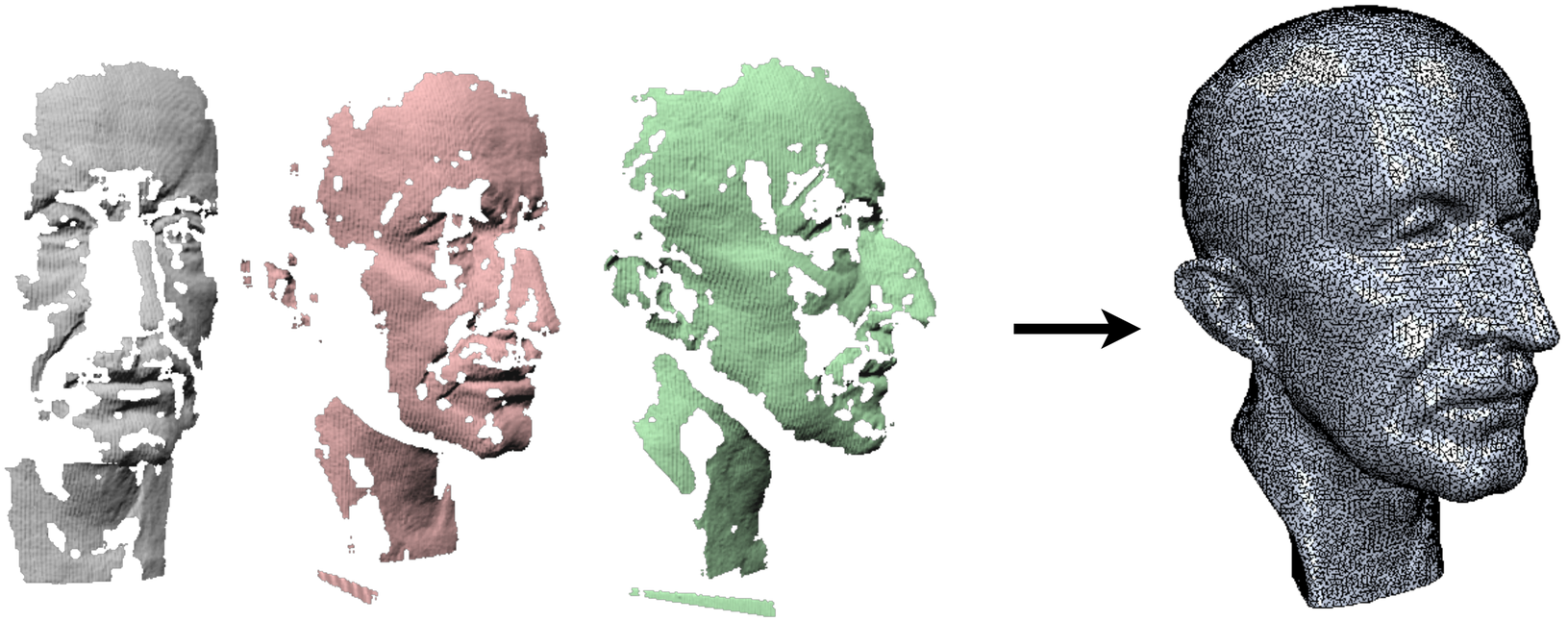
shorter baseline
less precision

3D Scanning Techniques



Multiple Scans Needed

- Take snapshots from different directions
  - Requires scan alignment / registration



set of raw scans

reconstructed model

Outline

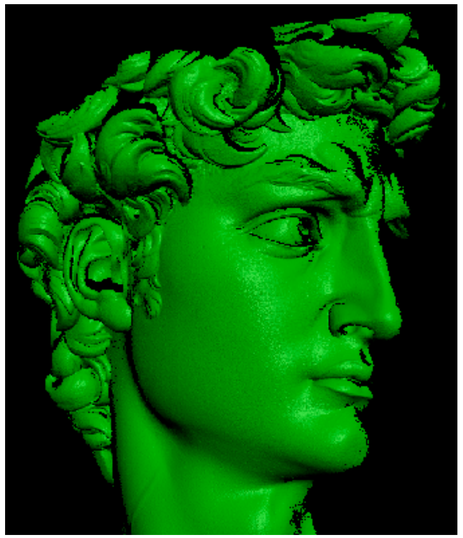
- 3D Scanning Techniques
- **Scan Registration**
- Explicit Reconstruction
  - from point clouds
  - from range scans
- Implicit Reconstruction
  - from point clouds
  - from range scans

Acknowledgements

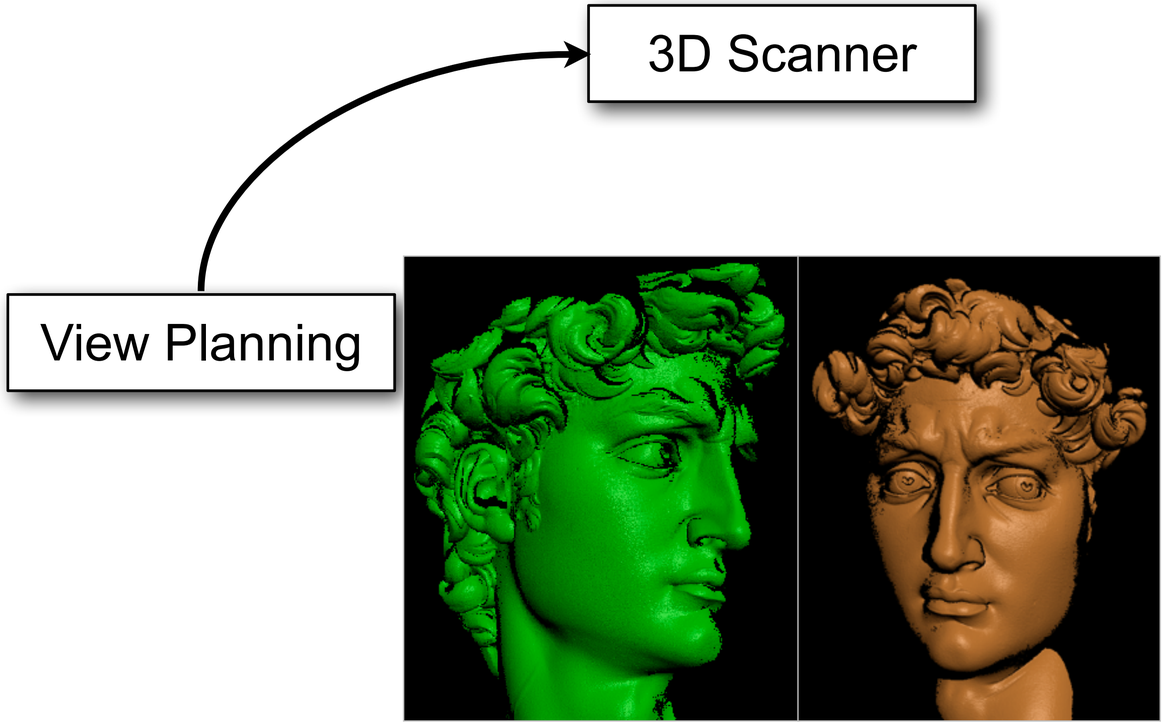
- Several images and slides are courtesy of Szymon Rusinkiewicz, Princeton University.
- See also tutorial (**ICP section**)
  Dynamic Geometry Processing
  Will Chang, Hao Li, Niloy J. Mitra, Mark Pauly, Michael Wand
  EUROGRAPHICS 2012
  http://www.mpi-inf.mpg.de/resources/deformableShapeMatching/EG2012_Tutorial/

3D Model Acquisition Pipeline

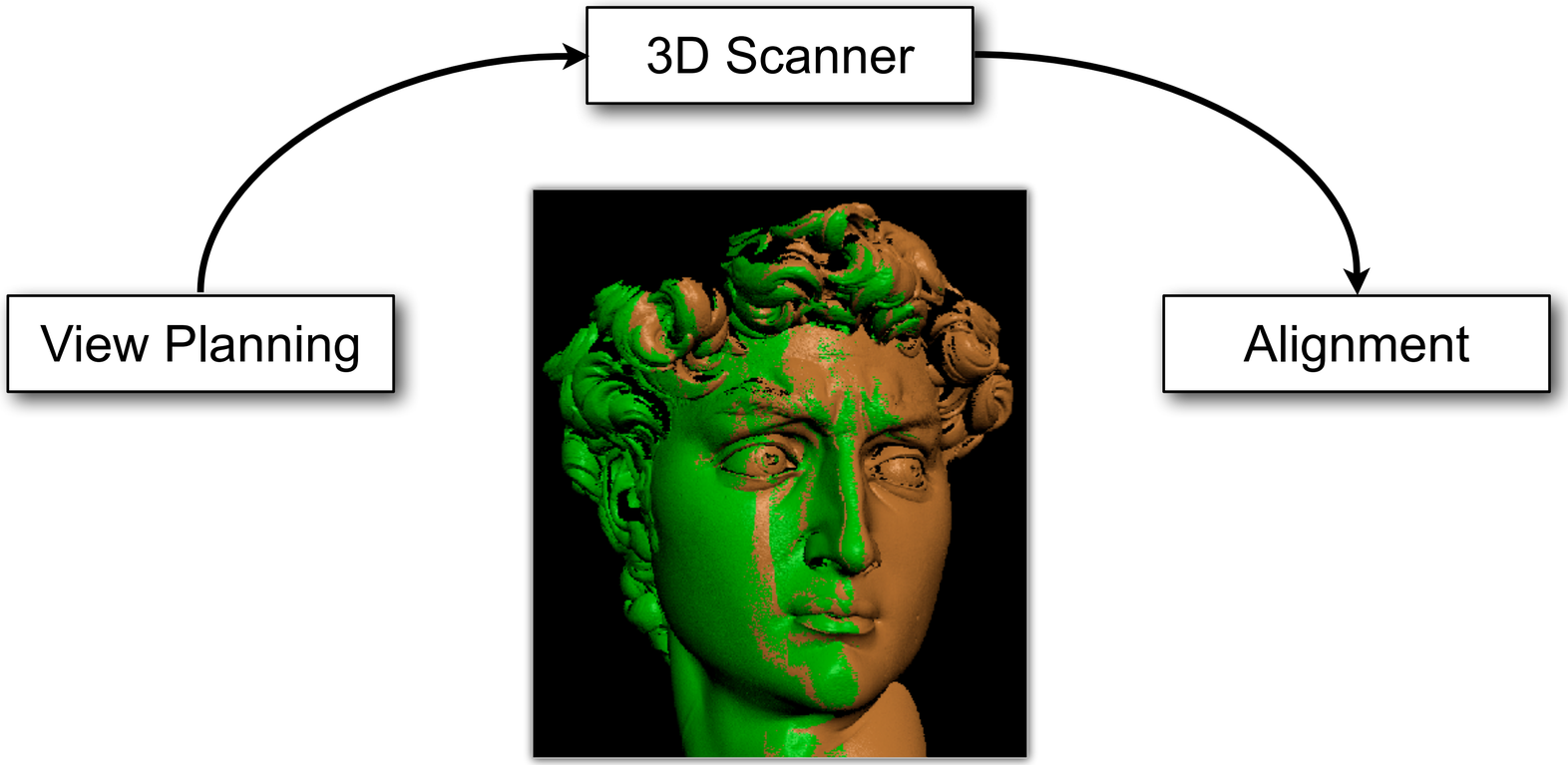
3D Scanner



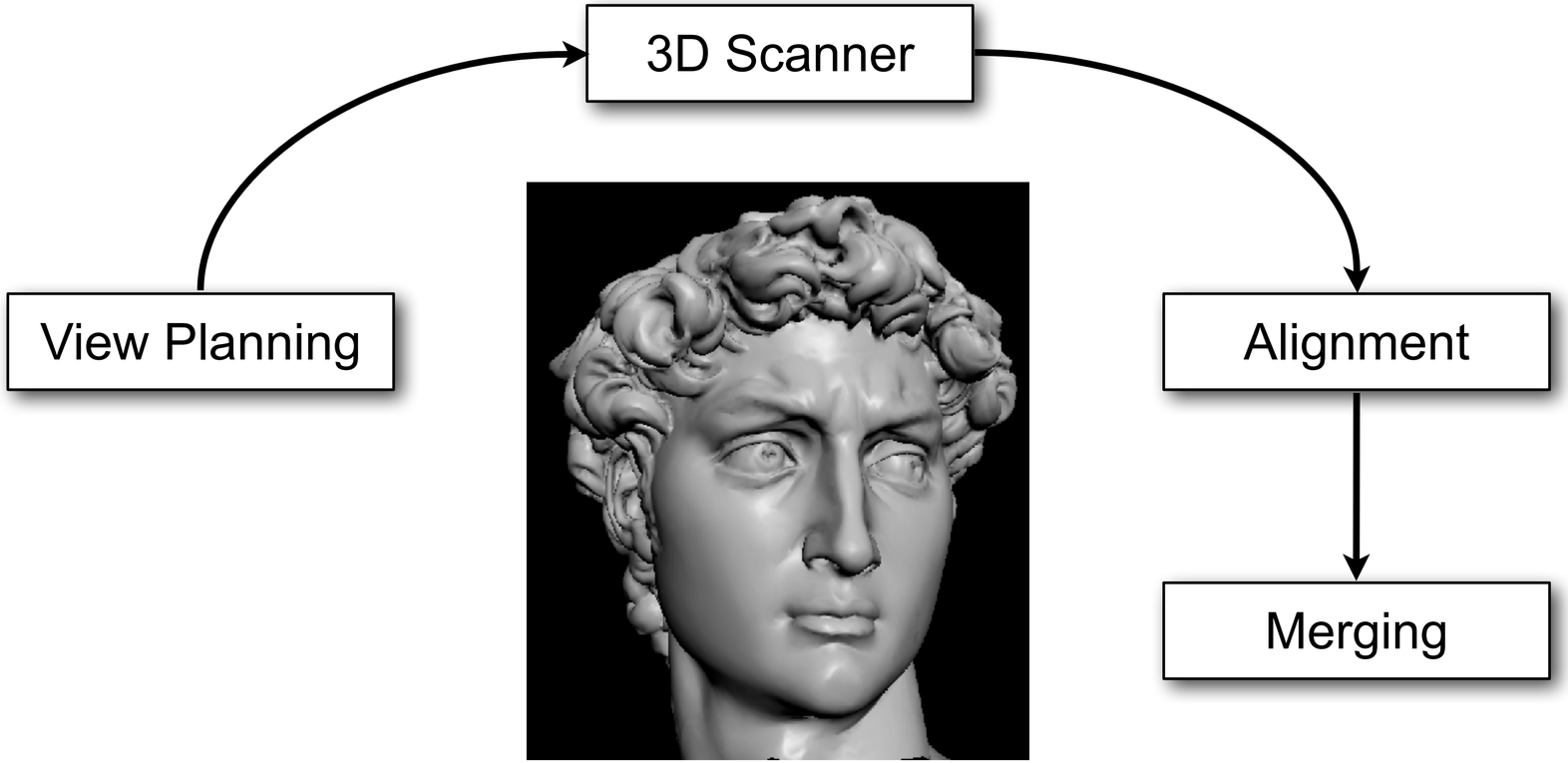
3D Model Acquisition Pipeline



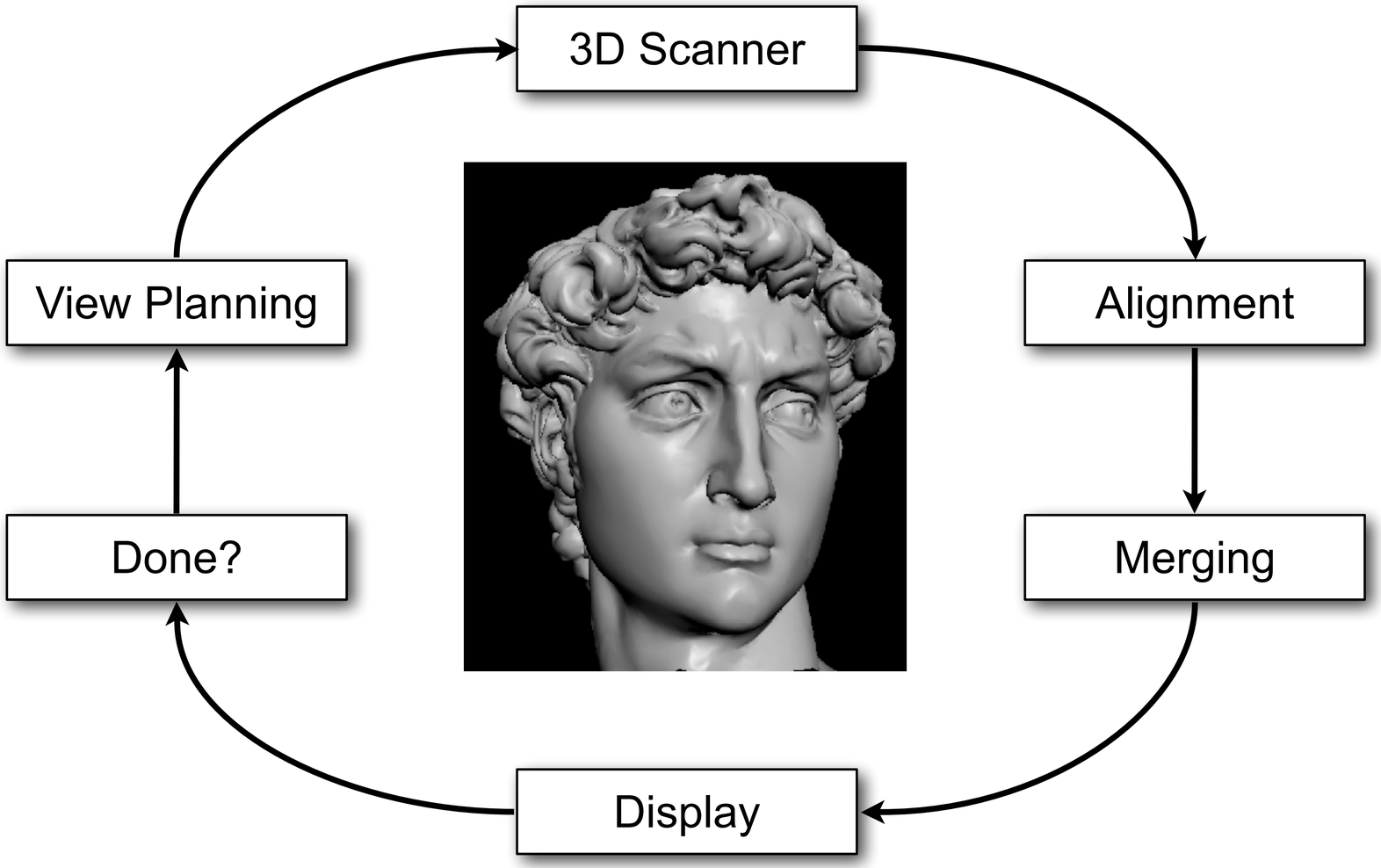
3D Model Acquisition Pipeline



3D Model Acquisition Pipeline

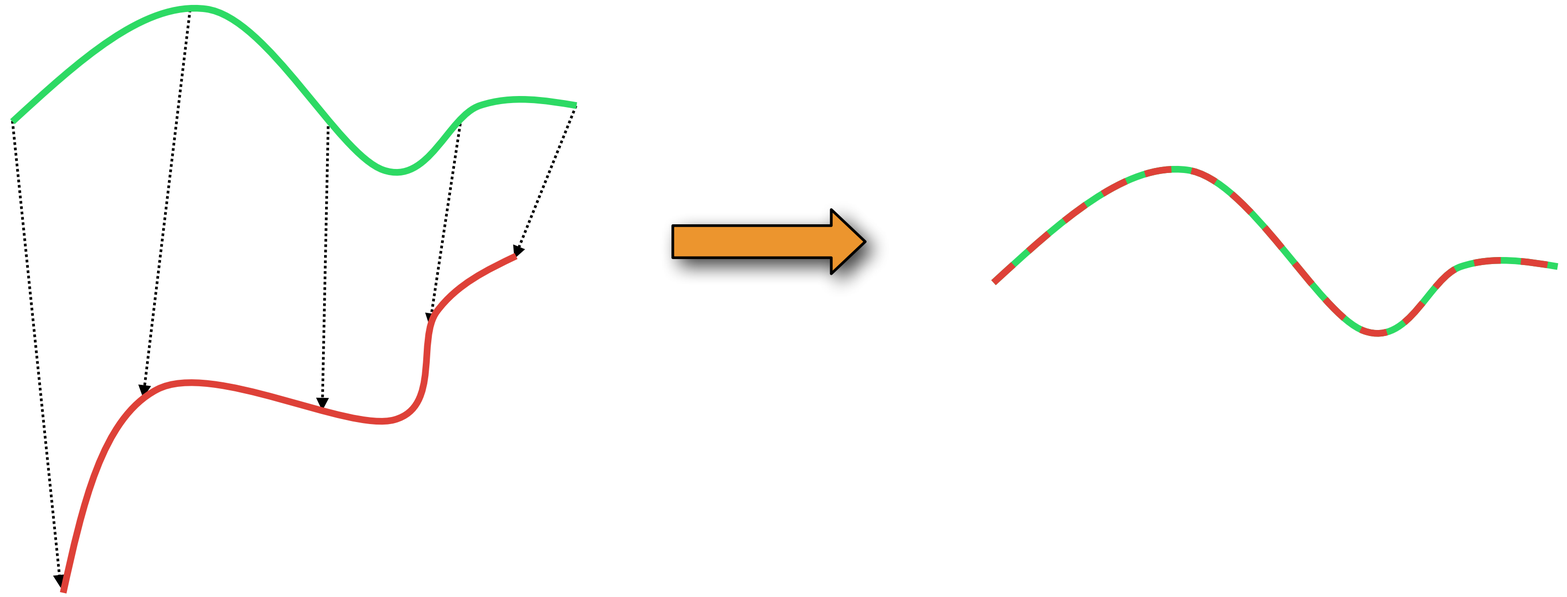


3D Model Acquisition Pipeline



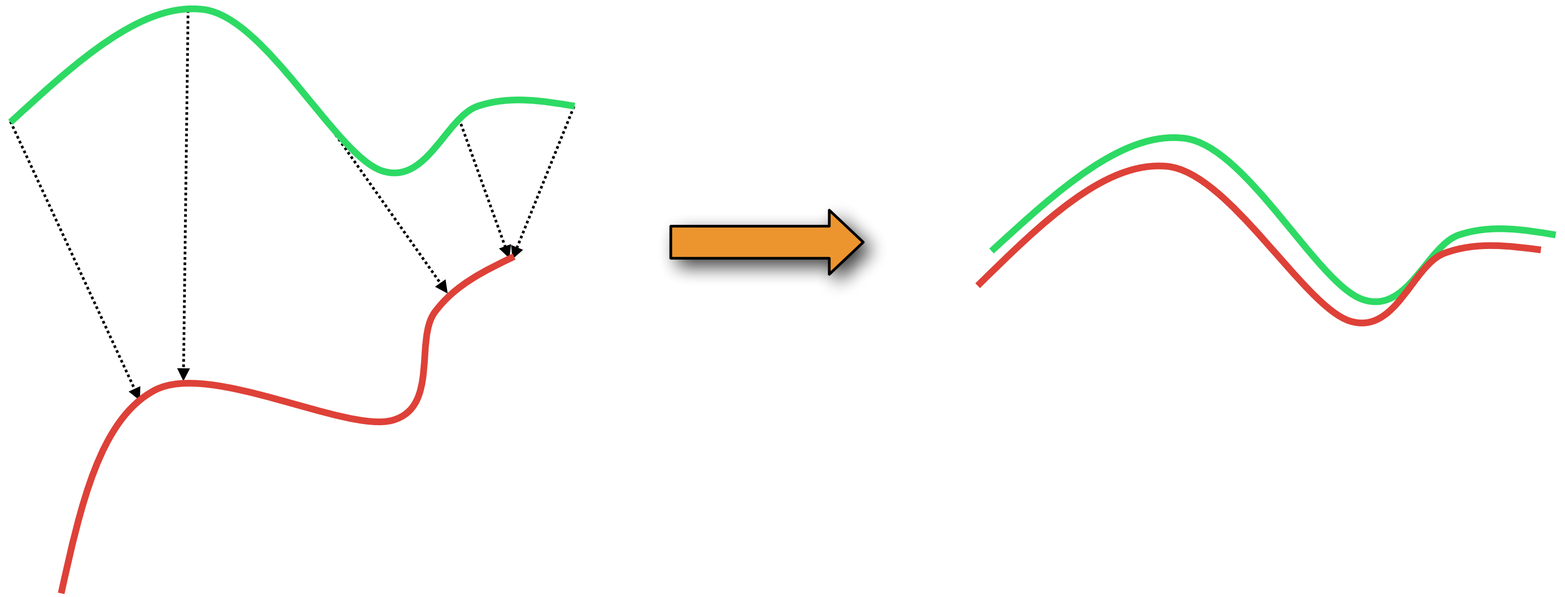
Aligning 3D Data

- If **correct correspondences** are known,
  can find correct relative orientation-preserving rigid transform
  (rotation and translation) — `see notes`



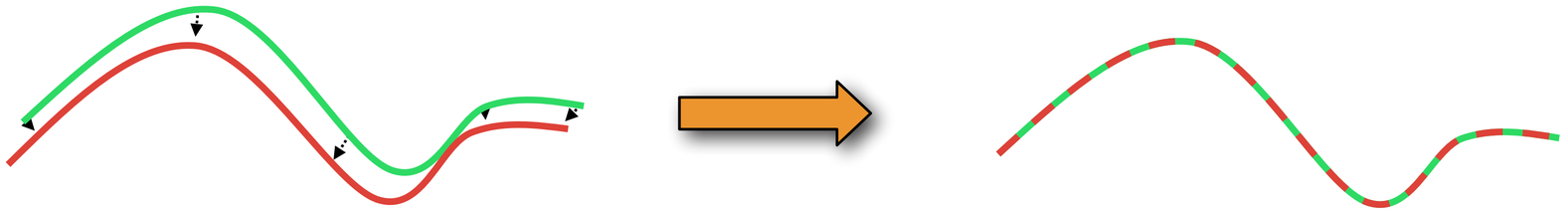
Aligning 3D Data

- How to **find correspondences**?
- Alternative: Assume **closest** points correspond



Aligning 3D Data

- … and iterate to find alignment
  - Iterative Closest Points (ICP) [Besl & McKay 92]
- Converges if starting position “sufficiently close”



Basic ICP Algorithm

Iterate until convergence:

1. **Select** a subset of points $\mathbf{p}_i$
2. **Match** each $\mathbf{p}_i$ to closest point $\mathbf{q}_i$ on other scan
3. **Reject** "bad" pairs ($\mathbf{p}_i$, $\mathbf{q}_i$)
4. **Compute** rotation **R** and translation **t** to minimize

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

Keep P fixed

5. **Iterate** after scan alignment: $\mathbf{q}_i \leftarrow \mathbf{R}\mathbf{q}_i + \mathbf{t}$

Basic ICP Algorithm

Iterate until convergence:

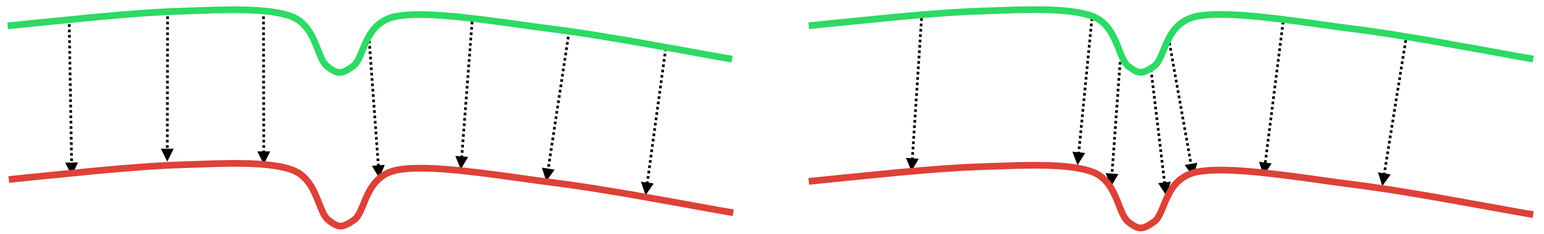
1. **Select a subset of points $\mathbf{p}_i$**
2. Match each $\mathbf{p}_i$ to closest point $\mathbf{q}_i$ on other scan
3. Reject "bad" pairs ($\mathbf{p}_i$, $\mathbf{q}_i$)
4. Computer rotation **R** and translation **t** to minimize

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

5. Iterate after scan alignment: $\mathbf{q}_i \leftarrow \mathbf{R}\mathbf{q}_i + \mathbf{t}$

Point Selection

- Use **all points** for matching
  - Too complex
- Uniform / random **subsampling**
  - Typically works well, but can miss features
- **Stable** sampling [Gelfand et al. 2003]
  - Normal-based sampling
  - Slippage analysis



Basic ICP Algorithm

Iterate until convergence:

1. Select a subset of points $\mathbf{p}_i$
2. **Match each p_i to closest point q_i on other scan**
3. Reject "bad" pairs ($\mathbf{p}_i$, $\mathbf{q}_i$)
4. Compute rotation **R** and translation **t** to minimize

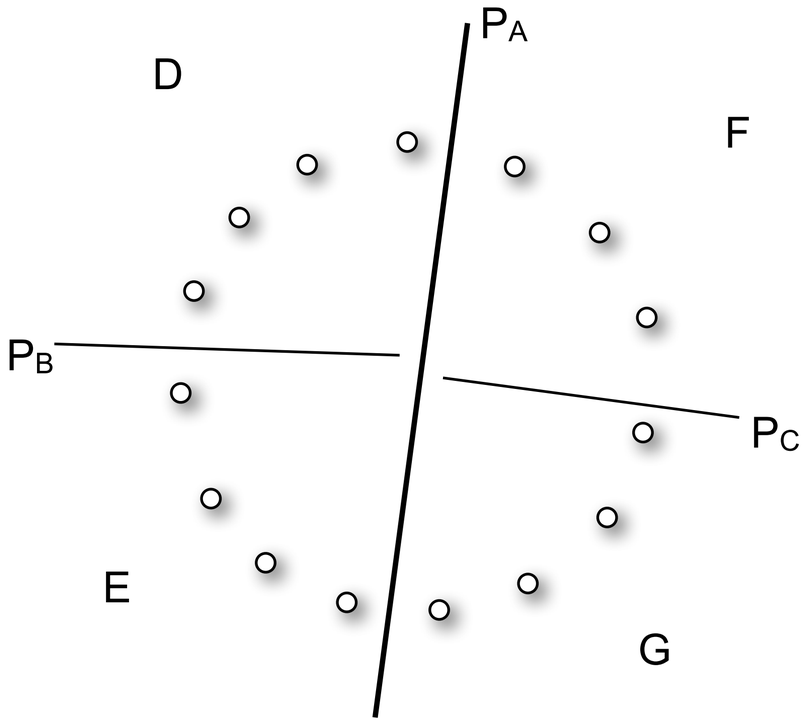
$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

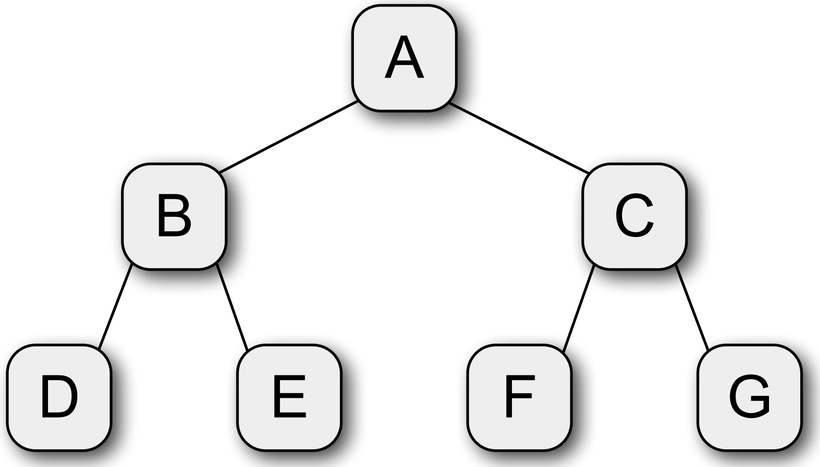
5. Iterate after scan alignment: $\mathbf{q}_i \leftarrow \mathbf{R}\mathbf{q}_i + \mathbf{t}$

Closest Point Search

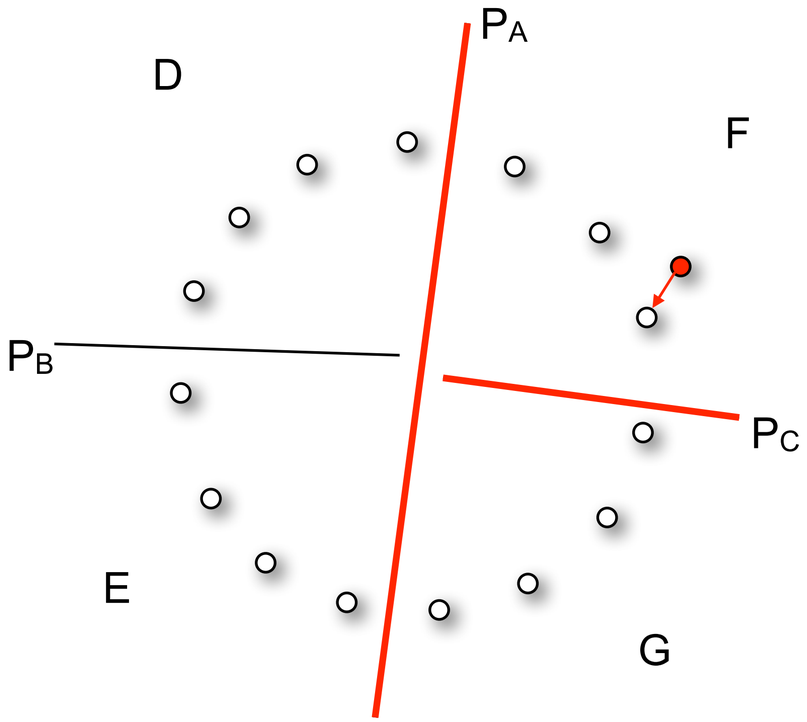
- Find closest point of a query point
  - Brute force: $O(n)$ complexity
- Use hierarchical **BSP tree**
  - Binary space partitioning tree (also kD-tree)
  - Recursively partition 3D space by planes
  - Tree should be balanced, put plane at median
  - log(n) tree levels, complexity $O(\log n)$

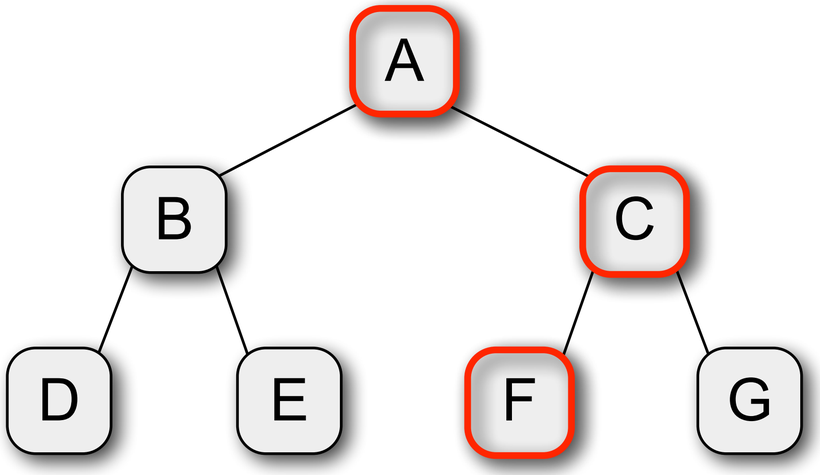
BSP Closest Points



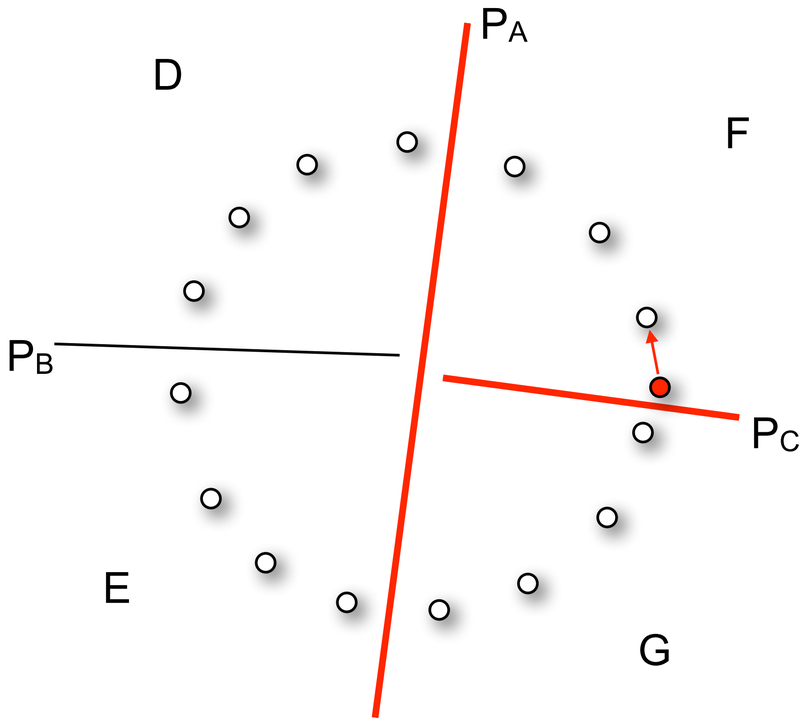


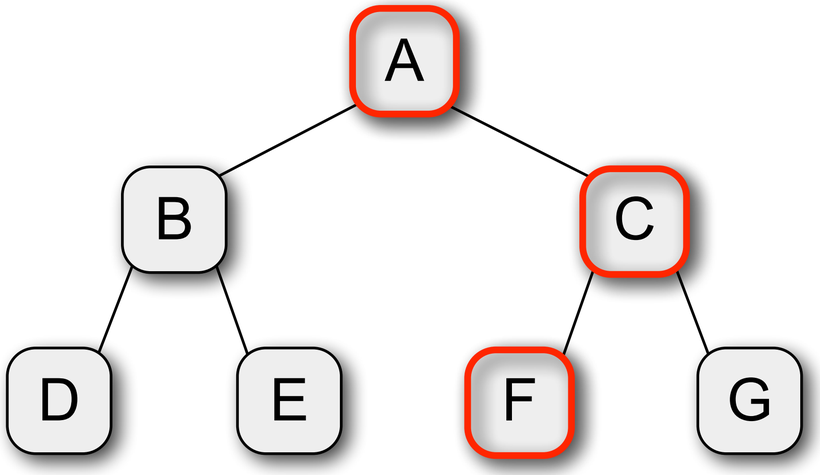
BSP Closest Points



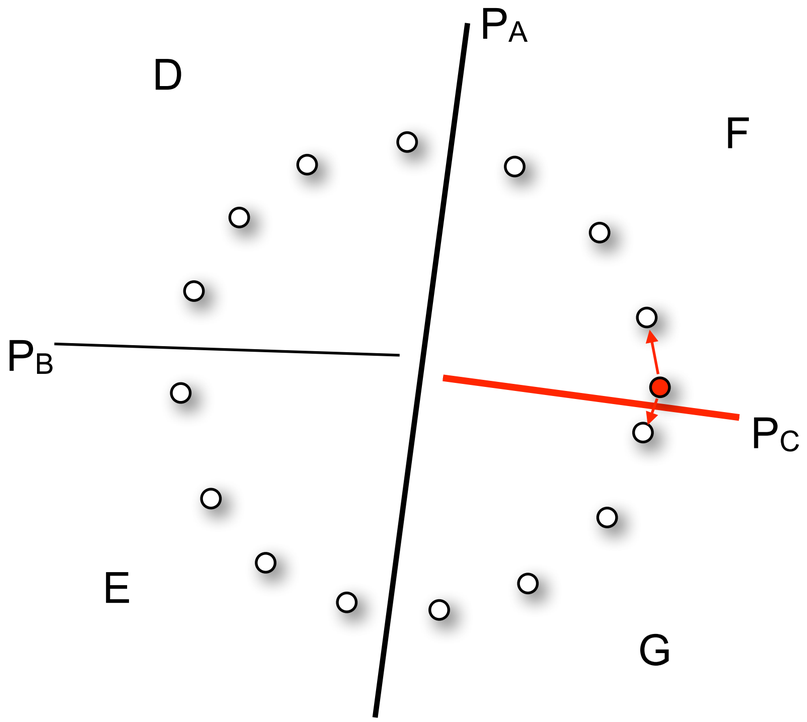


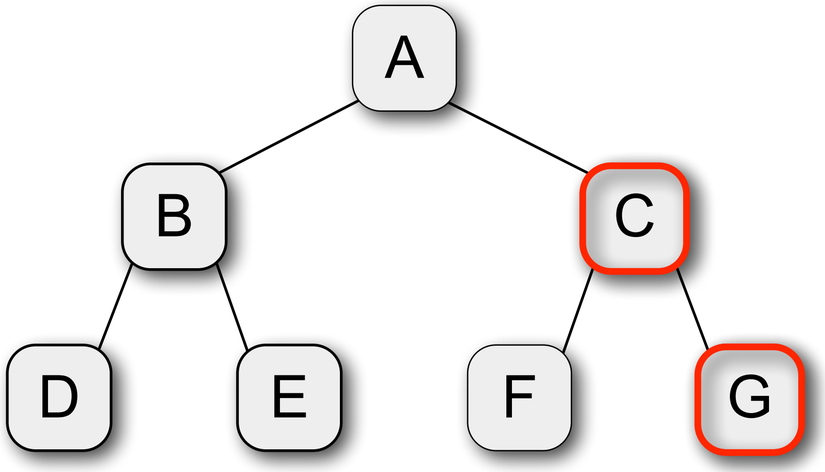
BSP Closest Points



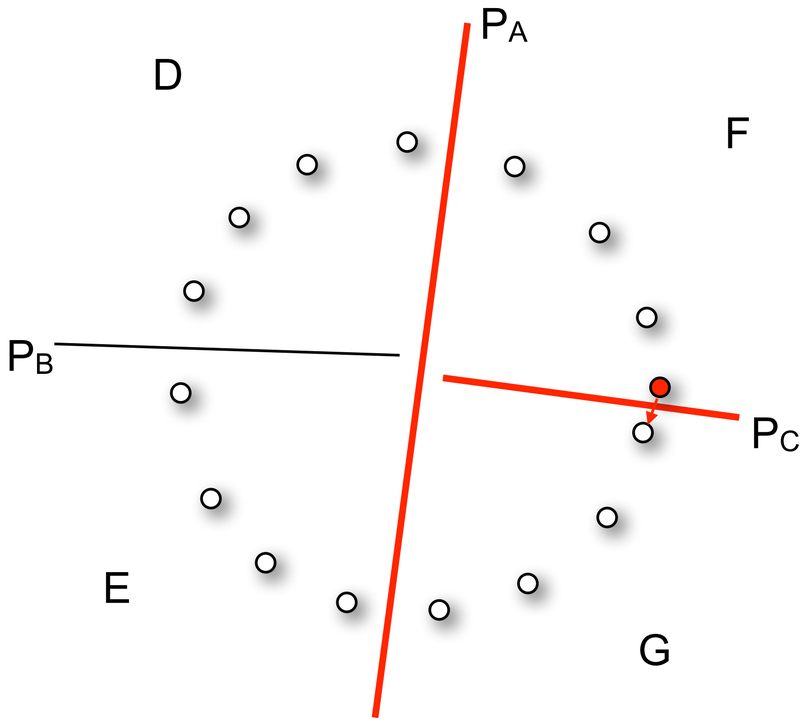


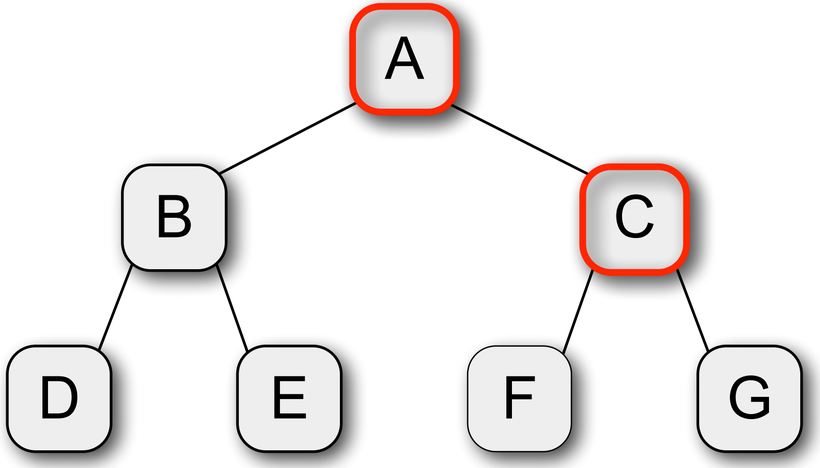
BSP Closest Points





BSP Closest Points





BSP Closest Points

```
BSPNode::dist(Point x, Scalar& dmin)
{
  if (leaf_node())
    for each sample point p[i]
      dmin = min(dmin, dist(x, p[i]));

  else
  {
    d = dist_to_plane(x);
    if (d < 0)
    {
      left_child->dist(x, dmin);
      if (|d| < dmin) right_child->dist(x, dmin);
    }
    else
    {
      right_child->dist(x, dmin);
      if (|d| < dmin) left_child->dist(x, dmin);
    }
  }
}
```

Basic ICP Algorithm

Iterate until convergence:

1. Select a subset of points $\mathbf{p}_i$
2. Match each $\mathbf{p}_i$ to closest point $\mathbf{q}_i$ on other scan
3. **Reject "bad" pairs $(\mathbf{p}_i, \mathbf{q}_i)$**
4. Compute rotation **R** and translation **t** to minimize

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

5. Iterate after scan alignment: $\mathbf{q}_i \leftarrow \mathbf{R}\mathbf{q}_i + \mathbf{t}$

Rejection of “Bad” Pairs

- Closest point often a bad approximation to corresponding point
- Only match ***compatible*** points
  - Do not match ***boundary points***
  - Do not match ***distant points***
  - Compatibility of ***colors***
  - Compatibility of ***normals***

Basic ICP Algorithm

Iterate until convergence:

1. Select a subset of points $\mathbf{p}_i$
2. Match each $\mathbf{p}_i$ to closest point $\mathbf{q}_i$ on other scan
3. Reject "bad" pairs (p_i, q_i)
4. **Compute rotation R and translation t to minimize**

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

5. Iterate after scan alignment: $\mathbf{q}_i \leftarrow \mathbf{R}\mathbf{q}_i + \mathbf{t}$

Shape Matching: Translation

- Define barycenters for each point set and recenter point sets

$$\bar{\mathbf{p}} := \frac{1}{m}\sum_{i=1}^{m}\mathbf{p}_i \qquad \bar{\mathbf{q}} := \frac{1}{m}\sum_{i=1}^{m}\mathbf{q}_i$$

$$\hat{\mathbf{p}}_i := \mathbf{p}_i - \bar{\mathbf{p}} \qquad \hat{\mathbf{q}}_i := \mathbf{q}_i - \bar{\mathbf{q}}$$

- Optimal translation vector **t** maps barycenters onto each other

$$\mathbf{t} = \bar{\mathbf{p}} - \mathbf{R}\bar{\mathbf{q}}$$

Shape Matching: Rotation

- Approximate nonlinear rotation by general matrix

$$\min_{\mathbf{R}} \sum_i \|\hat{\mathbf{p}}_i - \mathbf{R}\hat{\mathbf{q}}_i\|^2 \quad \rightarrow \quad \min_{\mathbf{A}} \sum_i \|\hat{\mathbf{p}}_i - \mathbf{A}\hat{\mathbf{q}}_i\|^2 \quad s.t. \quad \mathbf{R}\mathbf{R}^T = \mathbf{I}$$

- The least-squares solution reduces to:

$$\mathbf{A} = \sum_{i=1}^{m} (\mathbf{q}_i - \bar{\mathbf{q}})(\mathbf{p}_i - \bar{\mathbf{p}})^T = \sum_{i=1}^{m} \hat{\mathbf{q}}_i \hat{\mathbf{p}}_i^T$$

  - *Polar decomposition* extracts rotation from **A**

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \qquad \mathbf{R} = \mathbf{V}\mathbf{U}^T$$

$$\mathbf{t} = \bar{\mathbf{p}} - \mathbf{R}\bar{\mathbf{q}}$$

Shape Matching: Quaternions

- Alternative rotation optimization [Horn 1987]
  - Build the 4×4 matrix

$$\begin{pmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & -S_{xx}-S_{yy}+S_{zz} \end{pmatrix}$$

$$S_{xx} = \sum_i (\hat{\mathbf{p}}_i)_x (\hat{\mathbf{q}}_i)_x \ , \quad S_{xy} = \sum_i (\hat{\mathbf{p}}_i)_x (\hat{\mathbf{q}}_i)_y \ , \quad \dots$$

  - Its eigenvector $\mathbf{e}_{\max}$ w.r.t. largest eigenvalue is the rotation (axis $\mathbf{n}$, angle θ), represented as quaternion

$$\mathbf{e}_{\max} = \left(\cos\frac{\theta}{2}, \ \mathbf{n}_x \cdot \sin\frac{\theta}{2}, \ \mathbf{n}_y \cdot \sin\frac{\theta}{2}, \ \mathbf{n}_z \cdot \sin\frac{\theta}{2} \right)$$

Basic ICP Algorithm

Iterate until convergence:

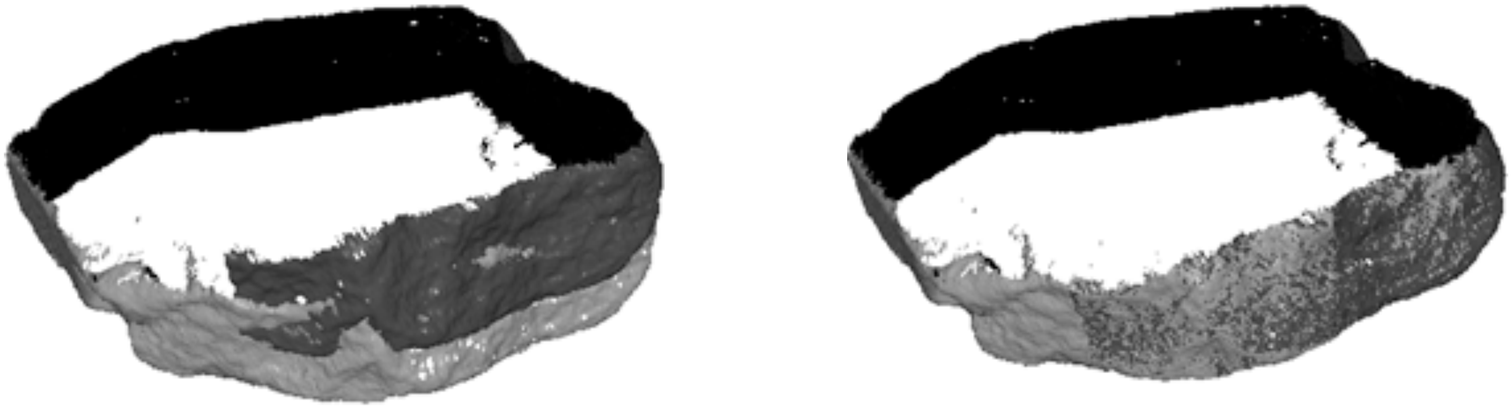
1. Select a subset of points $\mathbf{p}_i$
2. Match each $\mathbf{p}_i$ to closest point $\mathbf{q}_i$ on other scan
3. Reject "bad" pairs (p_i, q_i)
4. Compute rotation **R** and translation **t** to minimize

$$\min_{\mathbf{R},\mathbf{t}} \sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$$

5. **Iterate after scan alignment: $\mathbf{q}_i \leftarrow \mathbf{R}\mathbf{q}_i + \mathbf{t}$**

Global Registration

- What if we want to align *n*>2 scans to each other?
- Align each new scan to previous one?
  - No, leads to error accumulation
- Instead, distribute error between ***all*** scans



Global Registration

- Brute force approach #1:
  - For each scan
    - For each point
      - For each other scan
        - » Find closest point
  - Find optimal rotation and orientation of all scans based on collected point pairs

- Simultaneously optimizes all scans,
  but needs to solve a ($6n \times 6n$) nonlinear Newton optimization

Global Registration

- Brute force approach #2:
  - For each scan
    - For each point
      - For each other scan
        - » Find closest point
    - Find optimal rotation and orientation for this scan, while keeping all other scans fixed

- Distributes error slower,
  but can employ the simple [Horn87] optimization.

Literature

- **Horn: *Closed-form solution of absolute orientation using unit quaternions*, Journal Opt. Soc. Amer. 4(4), 1987.**
- **Besl & McKay: *A method for registration of 3D shapes*, Trans. on PAMI, 1992.**
- Pulli: *Multiview Registration for Large Data Sets*, 3DIM 1999.
- **Rusinkiewicz & Levoy: *Efficient Variants of the ICP Algorithm*, 3DIM 2001.**
- Gelfand et al.: *Geometrically Stable Sampling for the ICP Algorithm*, 3DIM 2003.
- Eggert et al.: *Estimating 3-D rigid body transformations: a comparison of four major algorithms*, Machine Vision and Applications, 1997.