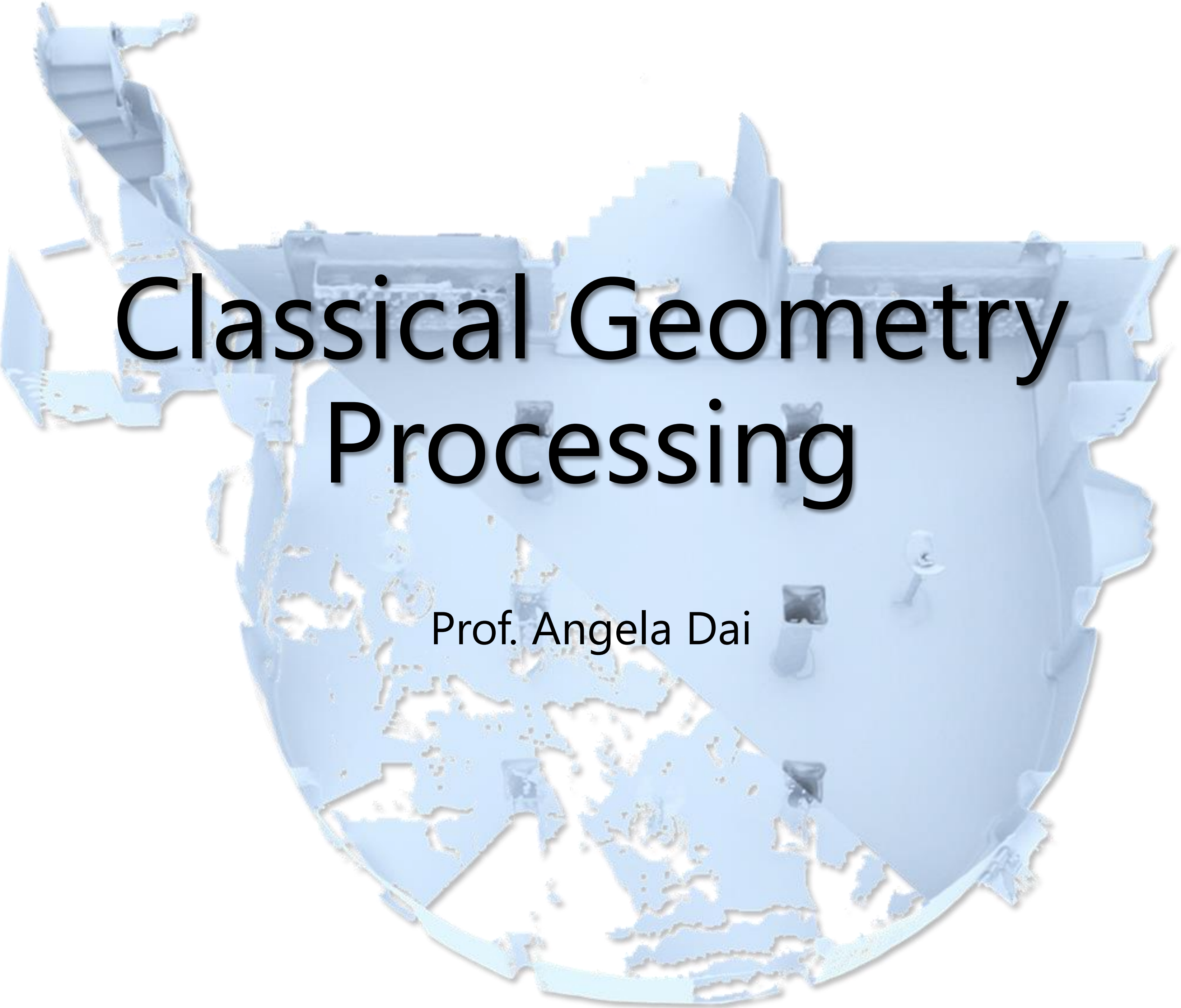
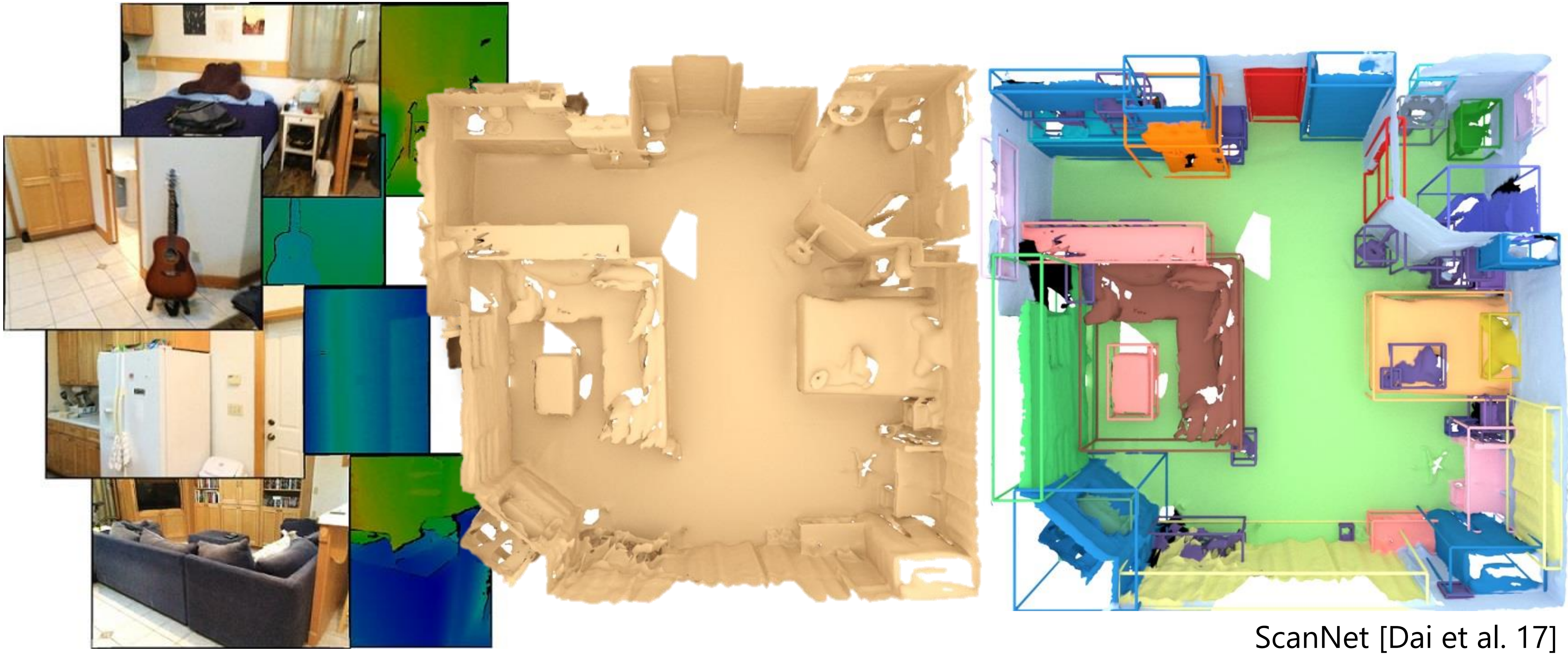


Classical Geometry Processing

Prof. Angela Dai

Brief Recap

Machine Perception of Real-World Environments



ScanNet [Dai et al. 17]

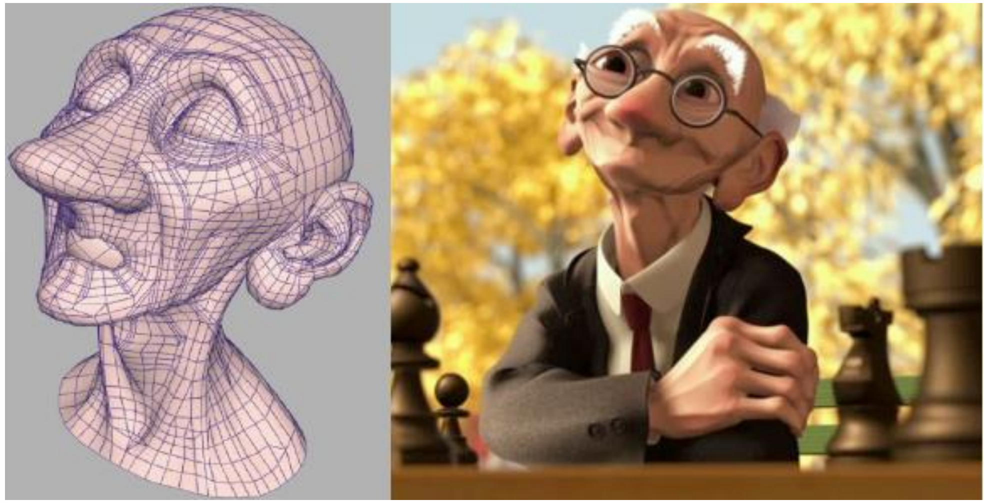
We *perceive* and *interact* with a 3D world

ASIMO, Honda

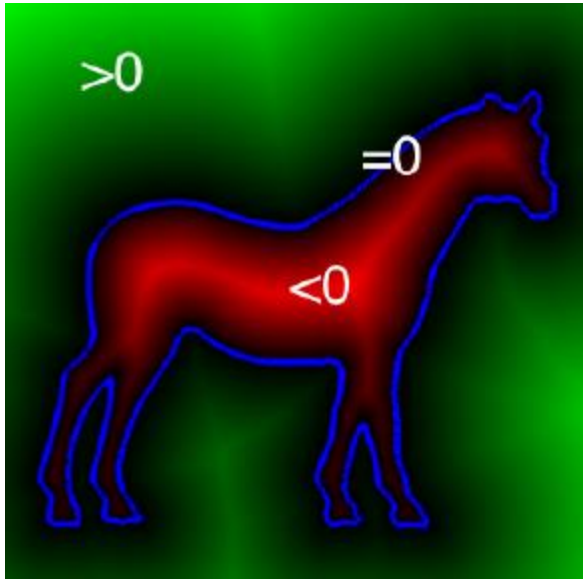
Star Trek TNG (Phantasms)

How to represent 3D?

Discrete:
Meshes,
Point Samples



Parametric



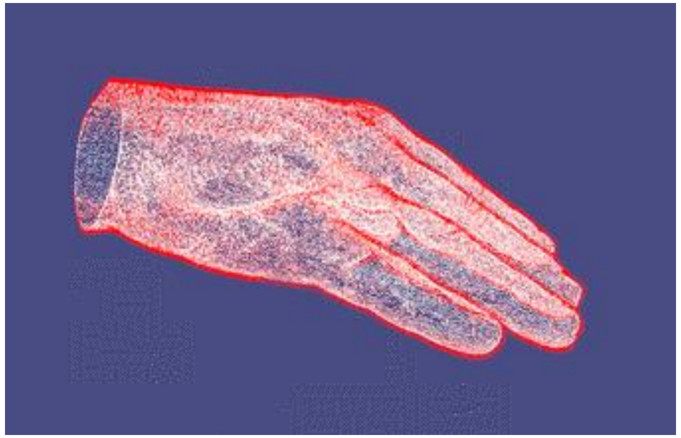
Implicit:
Distance Fields

Poisson Surface Reconstruction

- State-of-the-art reconstruction
- Oriented points to implicit function
- Formulates reconstruction as a variational problem

$$\min_{f} ||\nabla f - V||$$

where V is the vector field defined by the point samples



[Kazhdan et al. '06, Kazhdan et al. '13]

Poisson Surface Reconstruction

- No machine learning
- No data-driven priors

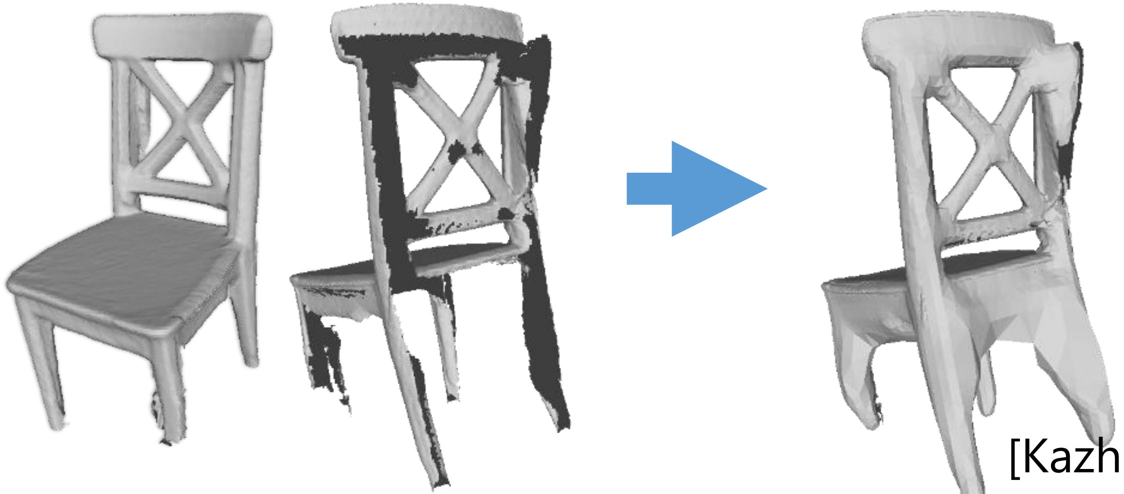
- How to fill in large holes?

[Kazhdan et al. ’06, Kazhdan et al. ’13]

Poisson Surface Reconstruction

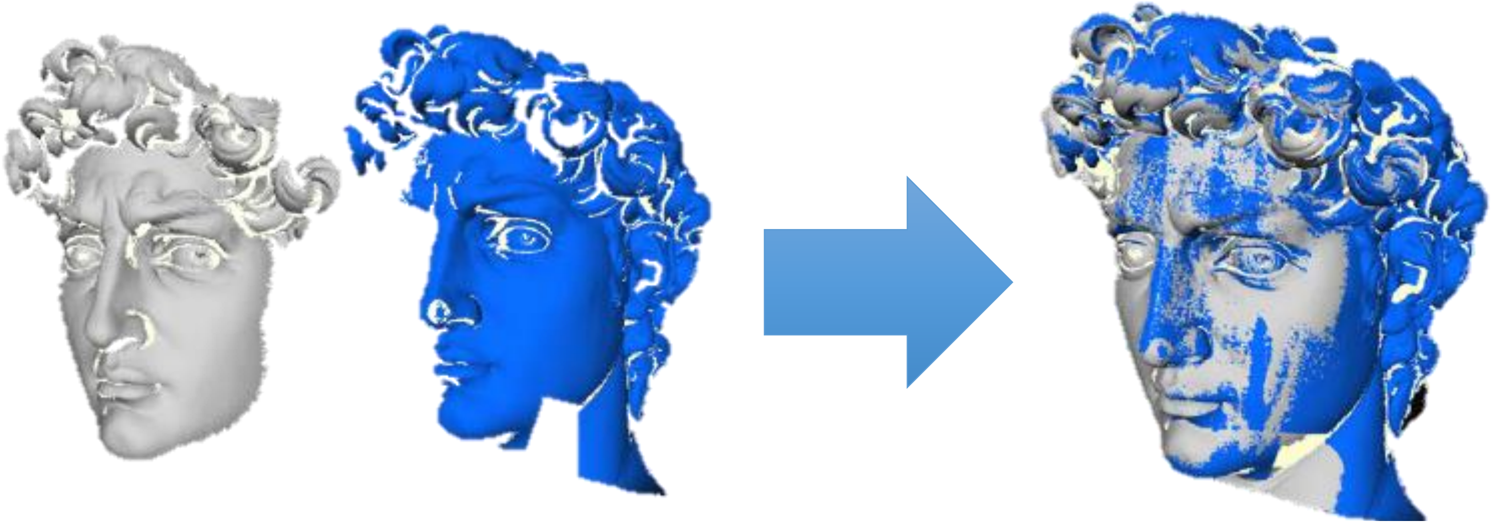
- Great with dense point samples
- Can be sensitive to V
  - Normal estimates need to be reliable
  - If multiple sensor measurements are taken, alignment must be good

- No general or semantic priors



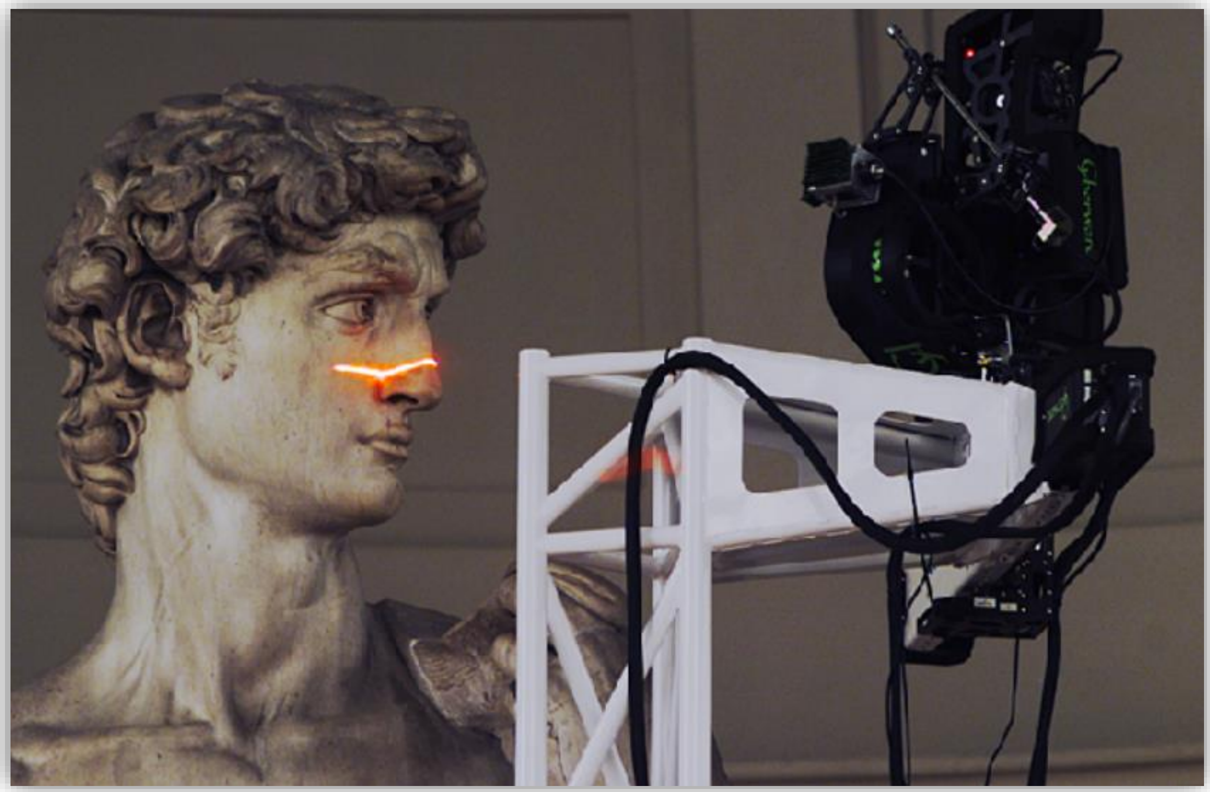
[Kazhdan et al. '06, Kazhdan et al. '13]

Registration and Alignment



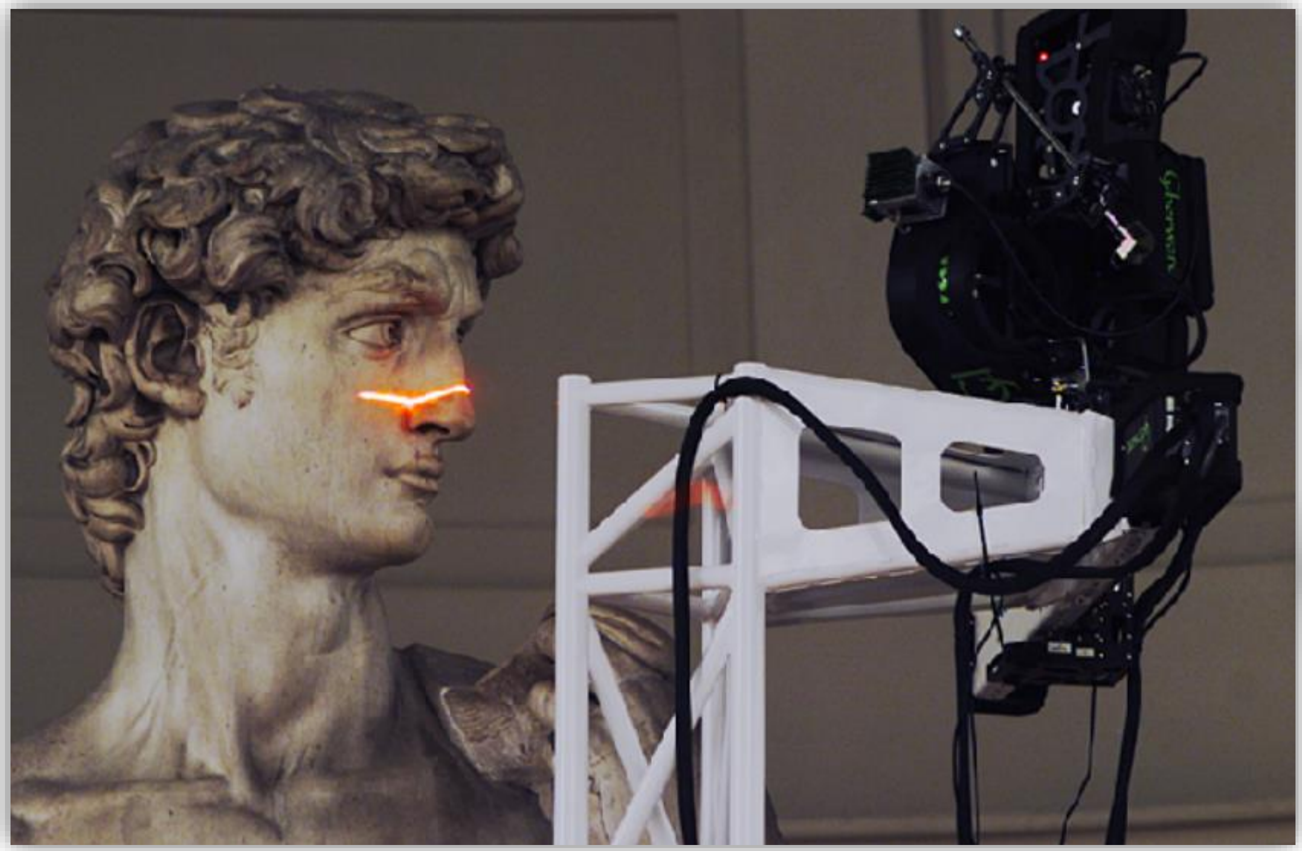
Geometry Acquisition

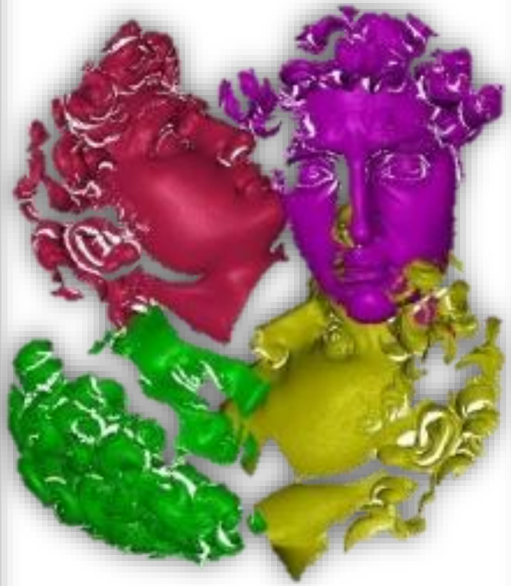
- 3D Scanning and Motion Capture (IN2354)

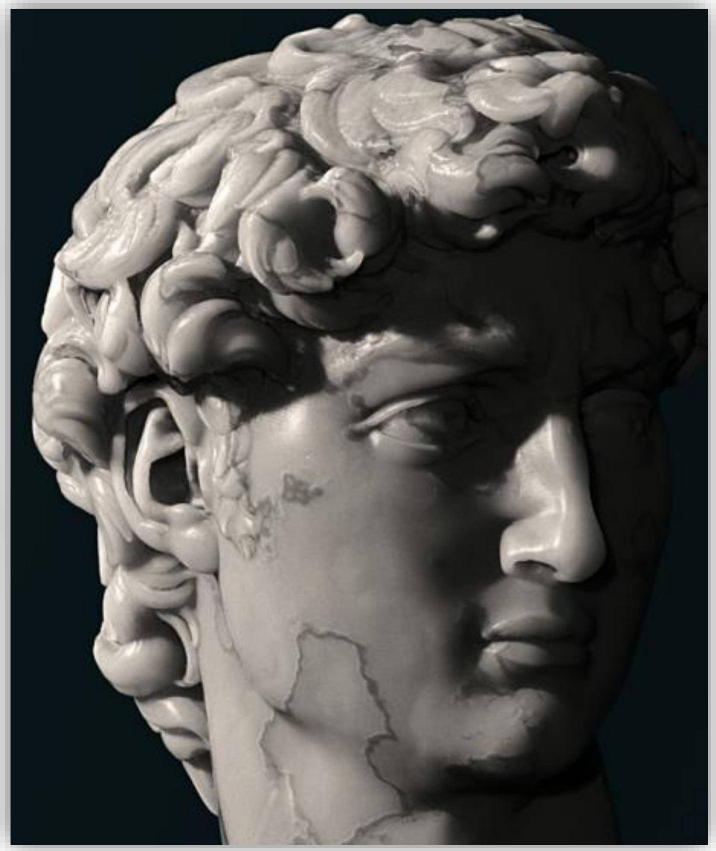


3D Alignment

- Many applications, e.g., 3D scanning, SLAM





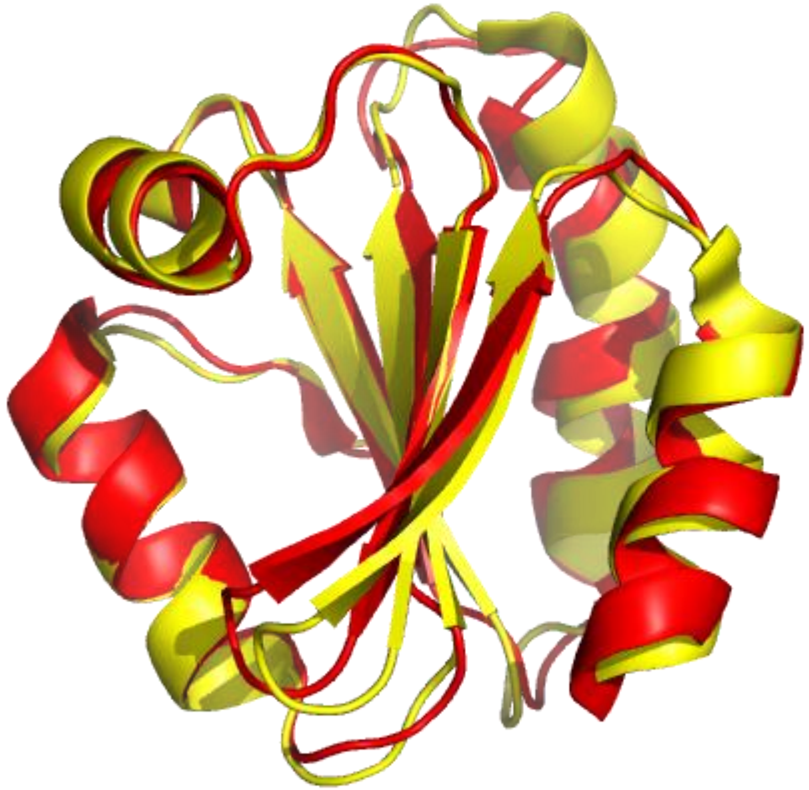


3D Alignment

- Many applications, e.g., 3D scanning, SLAM

3D Alignment

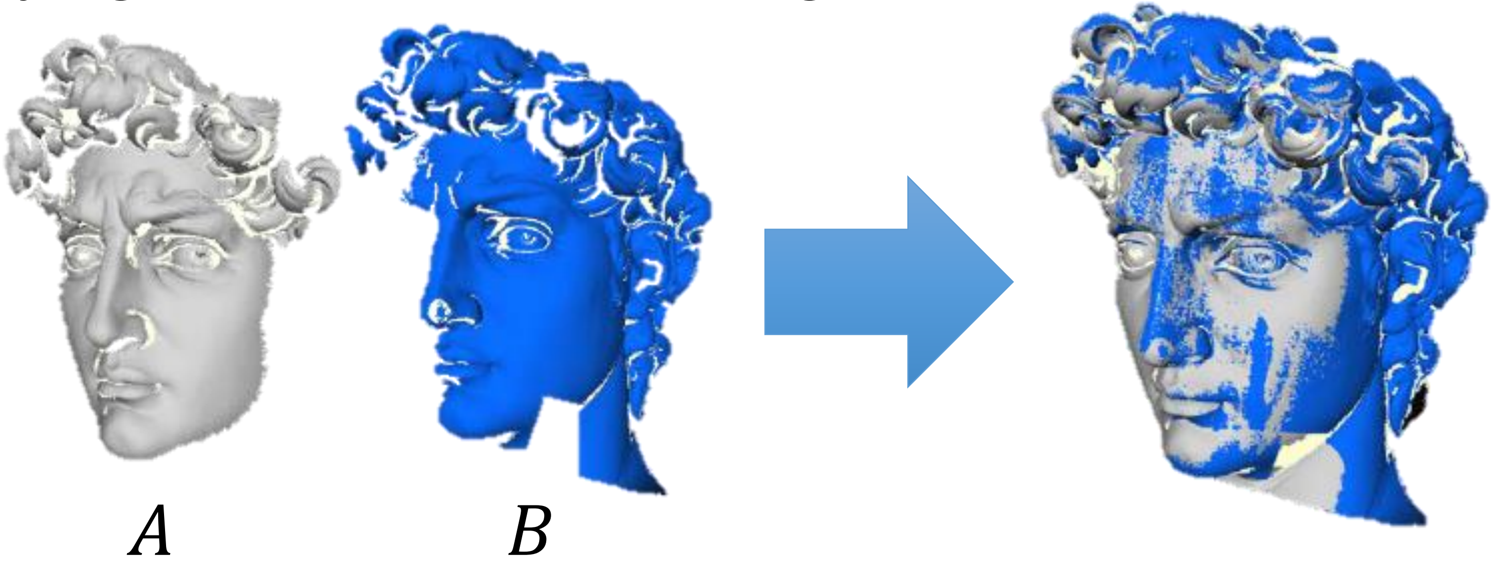
- Many applications



Protein Structure Alignment:
- (red) from humans
- (yellow) from fly Drosophila melanogaster.

3D Alignment (Registration)

- Input:
  - 2 shapes A and B with partial overlap
- Problem:
  - Register B to A by rigid transform, minimizing distance between A and B



How to measure distance?

- Measure of success for registration problem

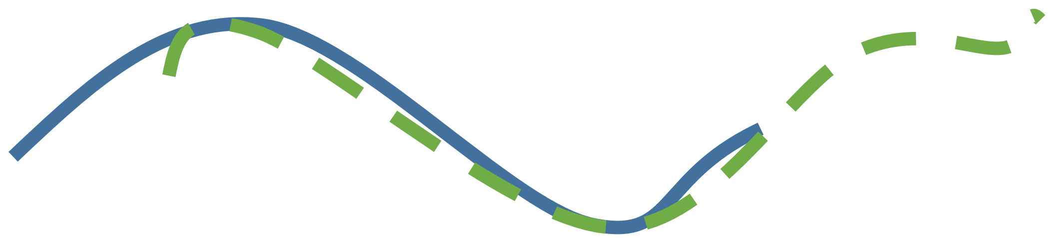
$$\min_{T} \delta(A, T(B))$$

T: rigid transform to bring B to A

- Fundamental for geometric similarity, classification, general machine learning losses

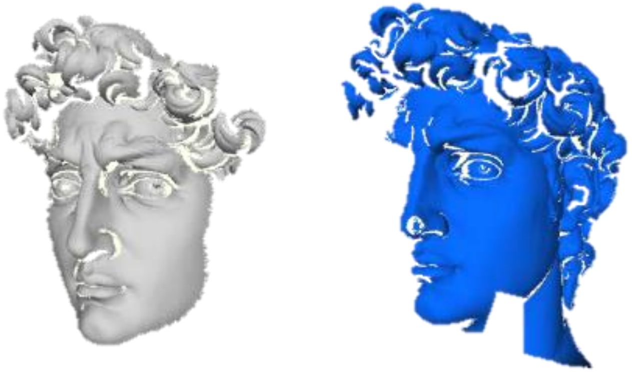
How to evaluate 3D distance?

- What about common function norms, e.g., ℓ_2?
  - We don't have correspondences across 3D structures, shapes
- Should support partial matches

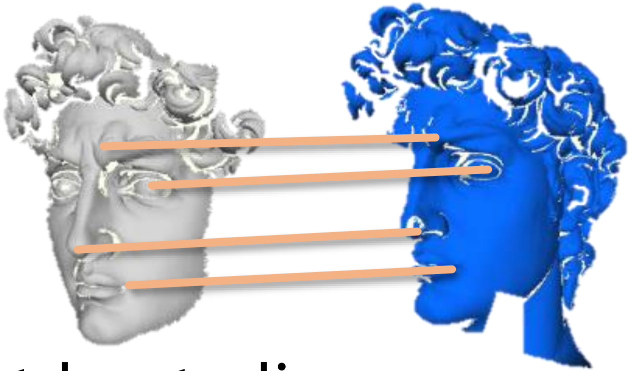


  - Trade-off between support size and aggregated distance
  - Distance for partial matches not a metric

Alignment Estimation



- Given shapes A and B
- Establish correspondences between A and B



- Find optimal transform that best aligns correspondences together, based on a distance measure

Transform Estimation

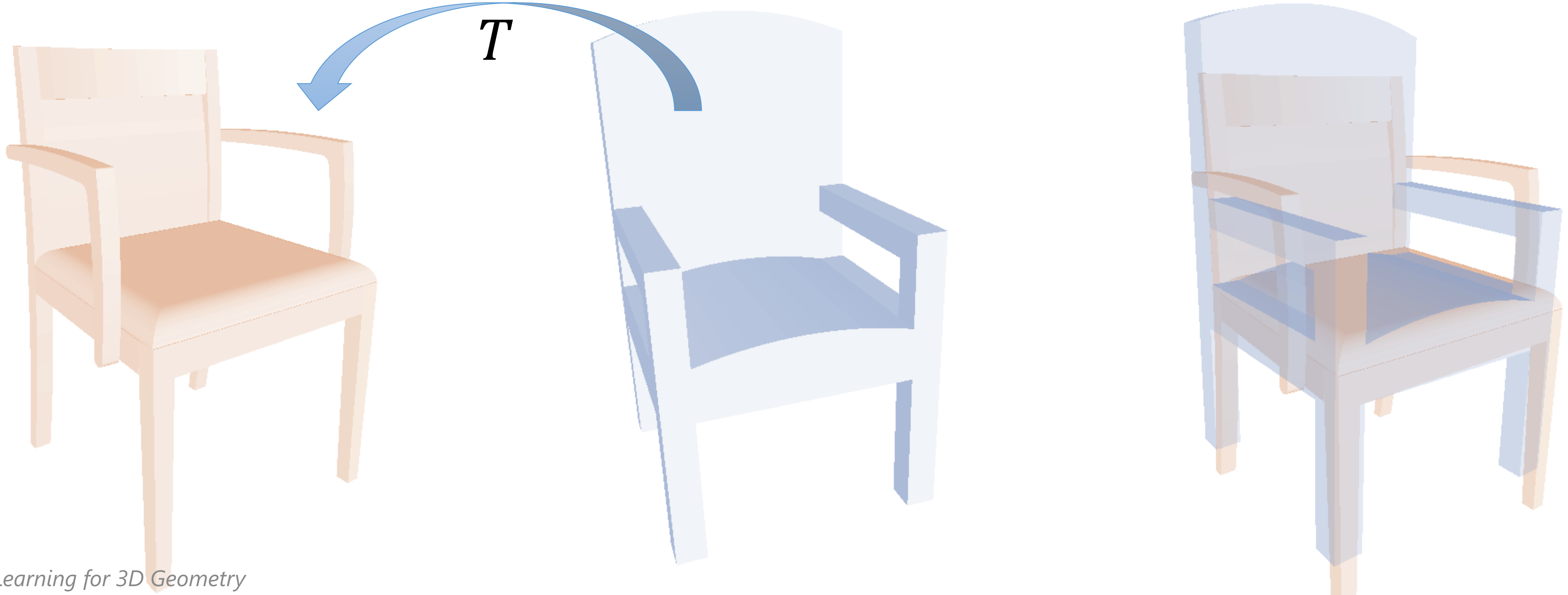
- Degrees of freedom
- Rigid motion has 6 degrees of freedom (3 rotation, 3 translation)
- Typically estimate with more correspondences -> overdetermined problem
- More general transforms -> more degrees of freedom, e.g., nonrigid deformations

Alignment Challenges

- Correspondence Estimation
  - Combinatorial search
- Transform Estimation
  - Transforms can be non-linear
- Difficult optimization -> look for good features, low-dimensional transforms

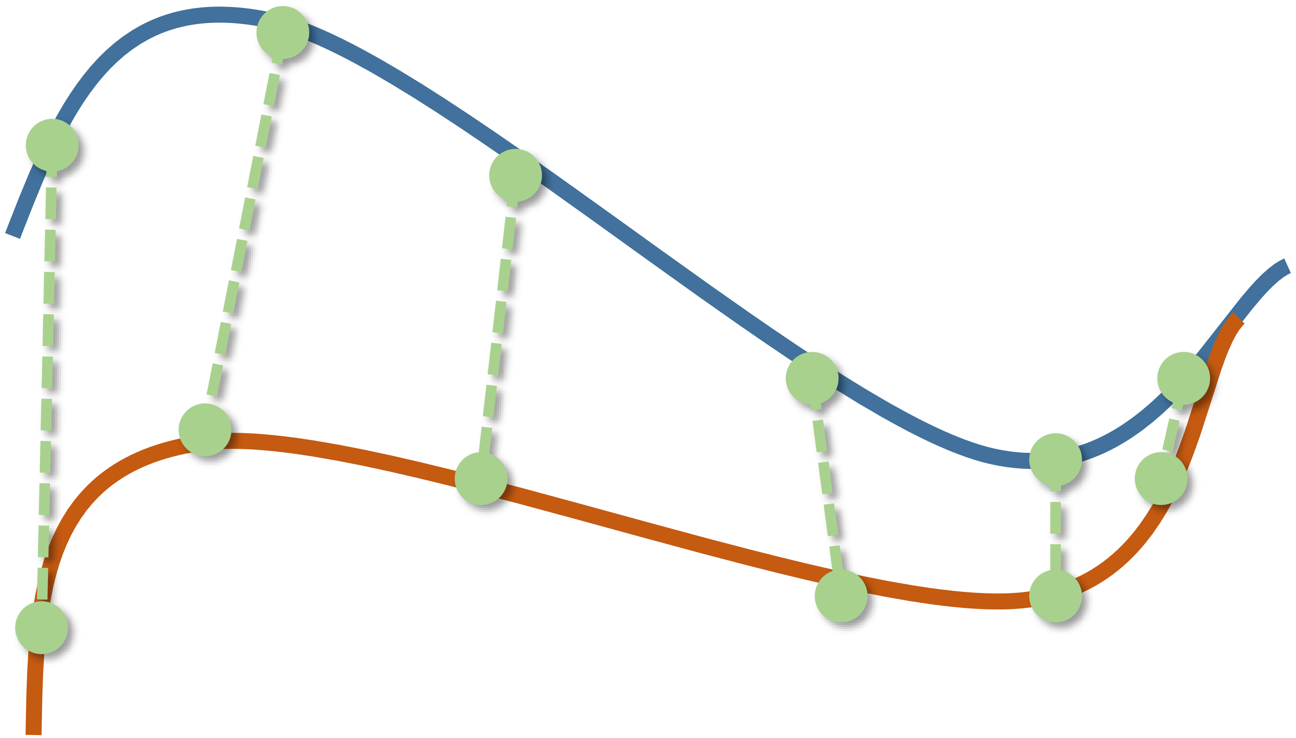
Rigid 3D Alignment

- Find 6DoF rigid transform that best aligns shapes, even if the shapes are different



Rigid 3D Alignment (Given Correspondences)

- Given correspondences $\{x_i\}, \{y_i\} \in \mathbb{R}^3$
- Find rigid transform $\boldsymbol{R}, t$ that minimizes $\sum_{i=1}^{N} \|\boldsymbol{R}x_i + t - y_i\|_2^2$



Solved as *orthogonal Procrustes problem* in 1966

Rigid 3D Alignment (Given Correspondences)

$$\min_{\boldsymbol{R},t} \sum_{i=1}^{N} \|\boldsymbol{R}x_i + t - y_i\|_2^2$$

- How to solve for $\boldsymbol{R}, t$?

- Consider coordinate system centered at the mean of the x_i

$$\min_{\boldsymbol{R},t} \sum_{i=1}^{N} \underbrace{\|t - y_i\|_2^2}_{\text{translation part}} - 2 \sum_{i=1}^{N} \underbrace{\langle \boldsymbol{R}x_i, y_i \rangle}_{\text{rotation part}}$$

Rigid 3D Alignment (Given Correspondences)

$$\min_{\boldsymbol{R},t} \sum_{i=1}^{N} \|t - y_i\|_2^2 - 2 \sum_{i=1}^{N} \langle \boldsymbol{R} x_i, y_i \rangle$$

- Translation: $t = \frac{1}{N} \sum_{i=1}^{N} y_i$ (align centroids)
- Remove translation by mean-centering:

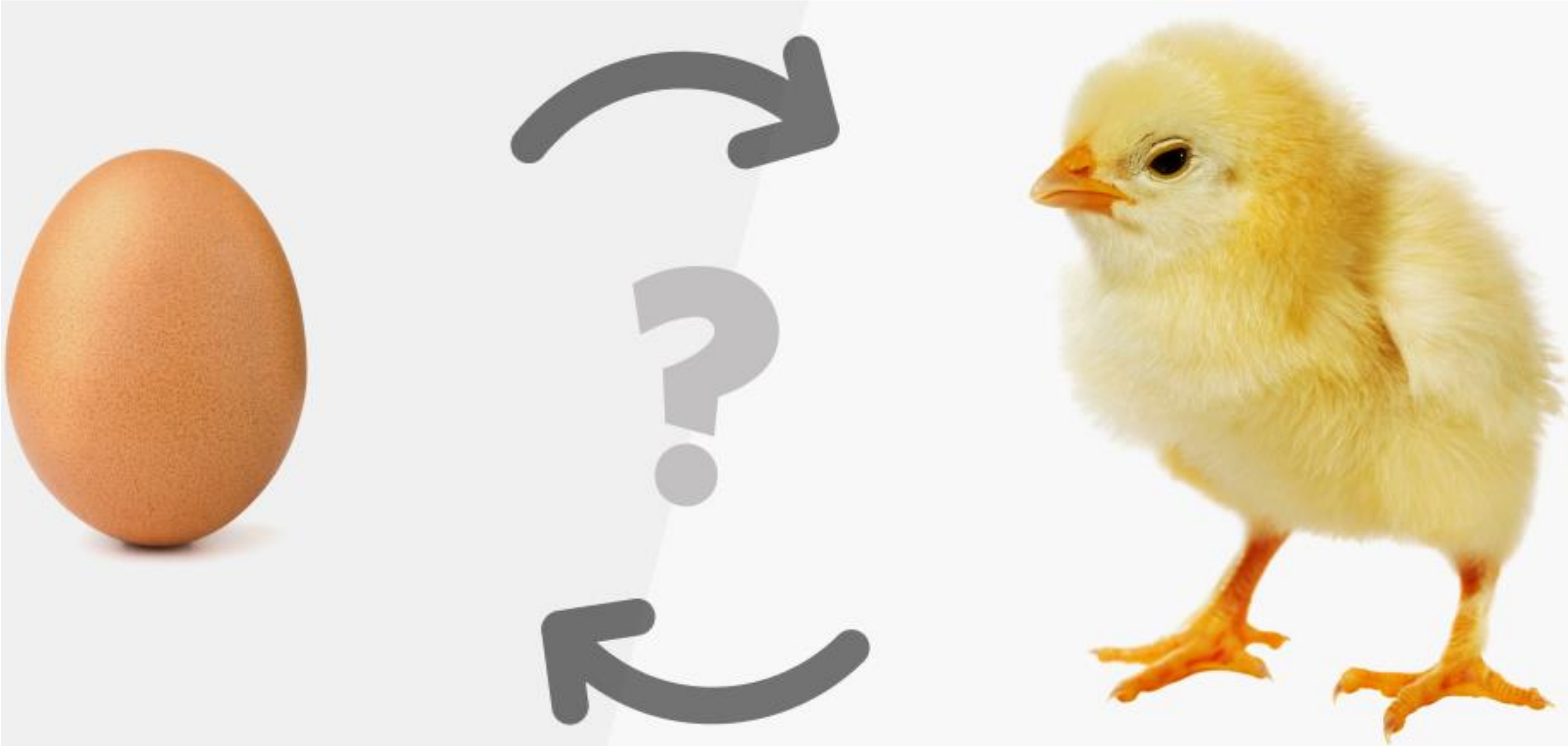
$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \qquad \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \qquad X = [x_0 - \bar{x}, \dots, x_n - \bar{x}]^T \qquad Y = [y_0 - \bar{y}, \dots, y_n - \bar{y}]^T$$

$$\begin{array}{cc} & N \\ 3 & \boxed{X} \\ & \boxed{Y} \end{array}$$

- Compute SVD: $XY^T = UDV^T$ ← 3×3 matrix
- Define $S = \begin{cases} I, & \text{if } \det(U)\det(V) = 1 \\ diag(1, \dots, 1, -1) & \text{otherwise} \end{cases}$

$$\boxed{R = USV^T}$$

How to get correspondences?



How to get correspondences?

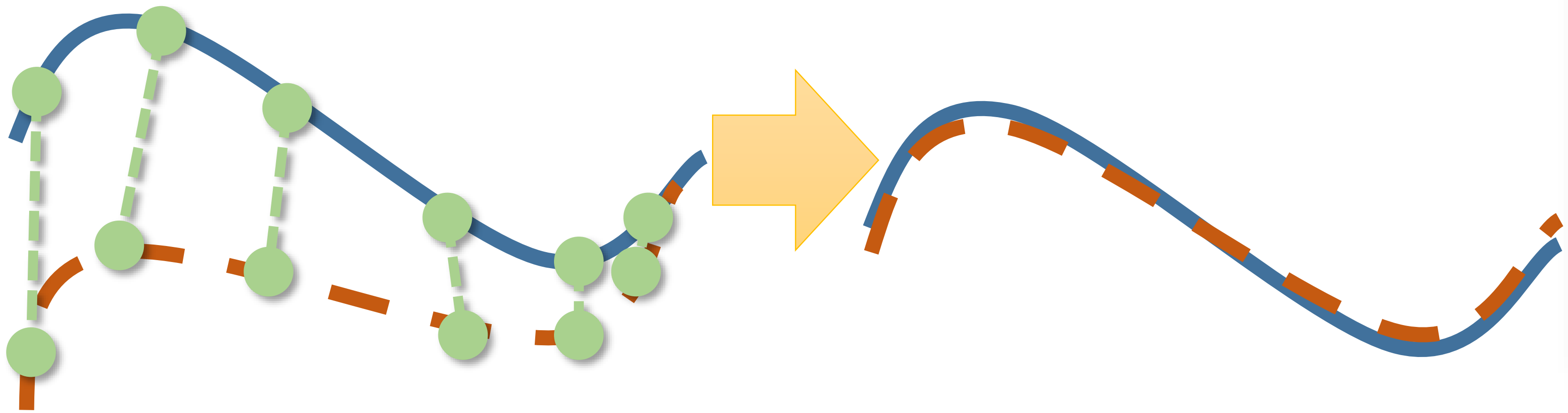
- Iterate between finding correspondences and solving for the best transform for those correspondences

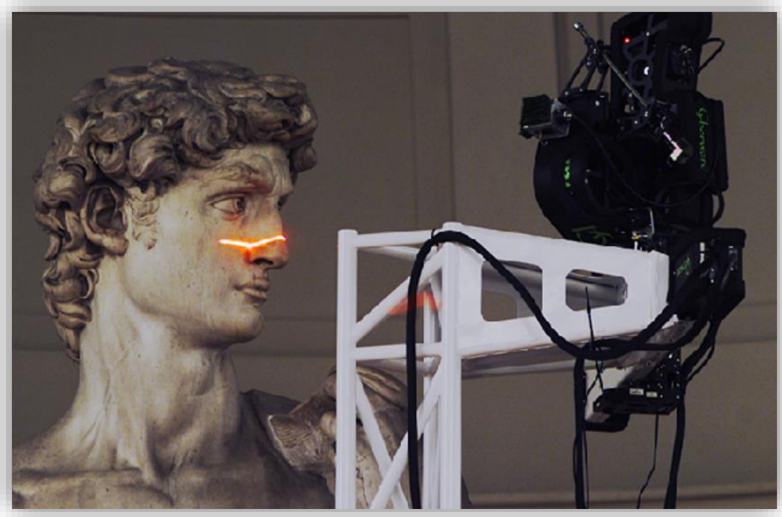
➢Iterative Closest Points (ICP)

- Various methods to explicitly match features (handcrafted or learned), which can also be refined with ICP

Iterative Closest Points (Besl and McKay '92)

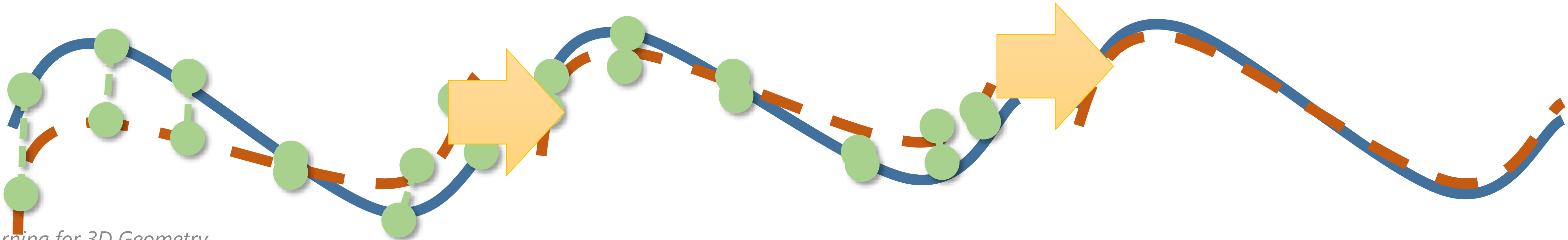
- Developed for aligning 3D shapes
- Nice analysis: *Efficient variants of the ICP algorithm* (Rusinkiewicz and Levoy 2001)



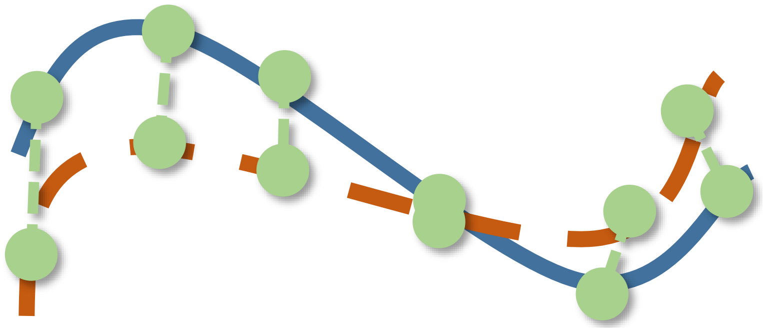


Iterative Closest Points

- How to find correspondences?
- Assume that closest points correspond
- Align the P_a points to their closest P_B neighbors; repeat
- Converges if starting positions are “close enough”



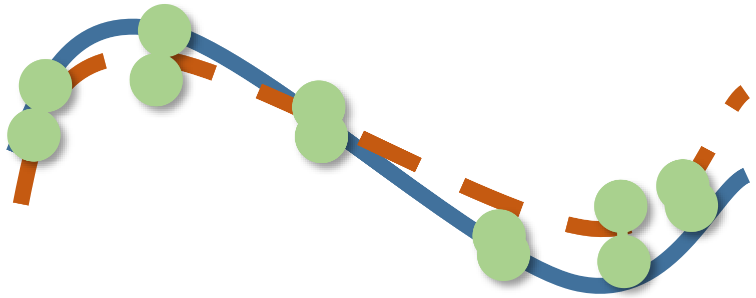
Iterative Closest Points



- Given a pair of shapes, A and B

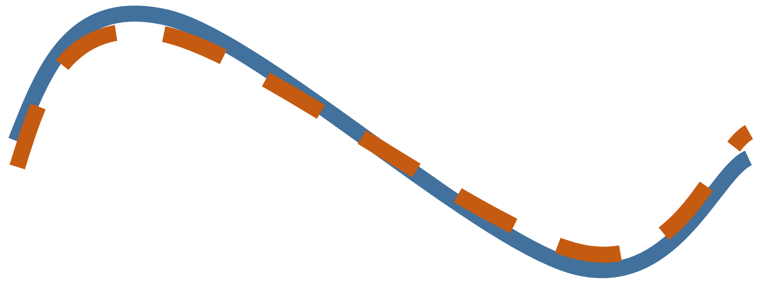
- Iterate:
  - Find corresponding points P_A and P_B based on proximity
  - Find optimal transform $\boldsymbol{R}, t$ minimizing $\underset{\boldsymbol{R},t}{\operatorname{argmin}} \sum_i ||\boldsymbol{R}x_i + t - y_i||_2^2$
  - Apply optimized $\boldsymbol{R}, t$

Iterative Closest Points



- Given a pair of shapes, A and B
- Iterate:
  - Find corresponding points P_A and P_B based on proximity
  - Find optimal transform $\boldsymbol{R}, t$ minimizing $\underset{\boldsymbol{R},t}{\operatorname{argmin}} \sum_i \|\boldsymbol{R} x_i + t - y_i\|_2^2$
  - Apply optimized $\boldsymbol{R}, t$

Iterative Closest Points



- Given a pair of shapes, A and B

- Iterate:
  - Find corresponding points P_A and P_B based on proximity
  - Find optimal transform $\boldsymbol{R}, t$ minimizing $\underset{\boldsymbol{R},t}{\operatorname{argmin}} \sum_i ||\boldsymbol{R}x_i + t - y_i||_2^2$
  - Apply optimized $\boldsymbol{R}, t$

ICP: Runtime Analysis

- Each iteration:
  - Find closest points:
    - $O(N_B)$ per point
    - $O(N_B * N_A)$ total
  - Compute optimal alignment: $O(N_A)$
  - Update scene $O(N_A)$

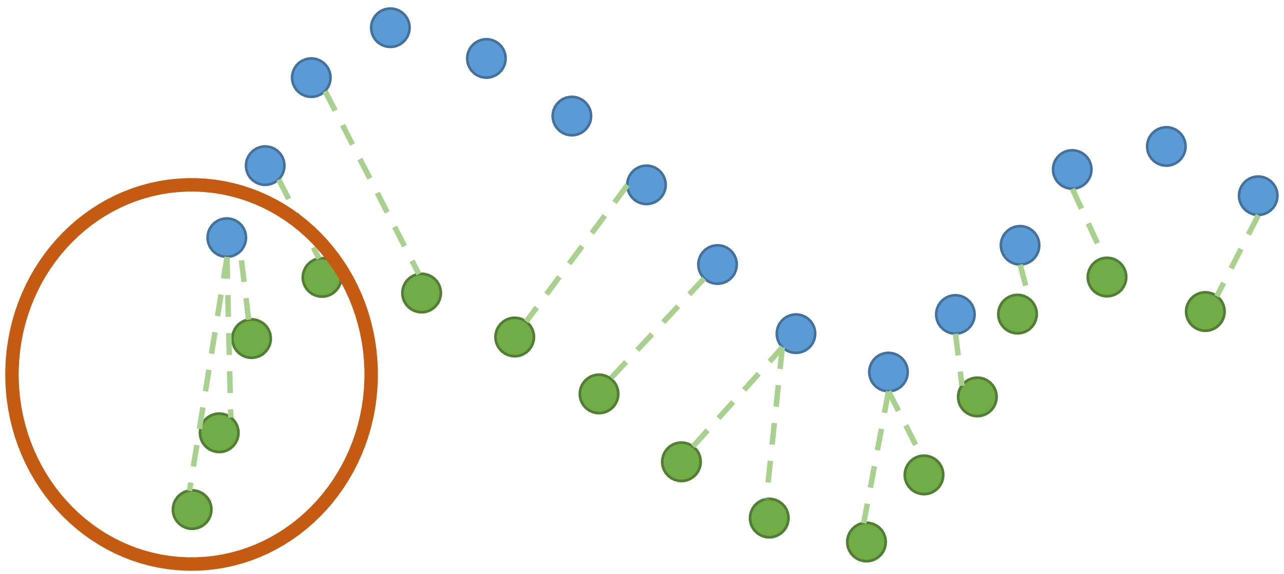
- Speed up with fast or approximate nearest-neighbor data structures, e.g., *kd*-tree

ICP Analysis

- Selection of points
- Matching correspondences
- Weighting correspondences
- Rejecting outlier correspondences
- Assigning error metric to the current transform
- Minimizing error metric w.r.t transform

ICP Analysis

- How to select correspondences?

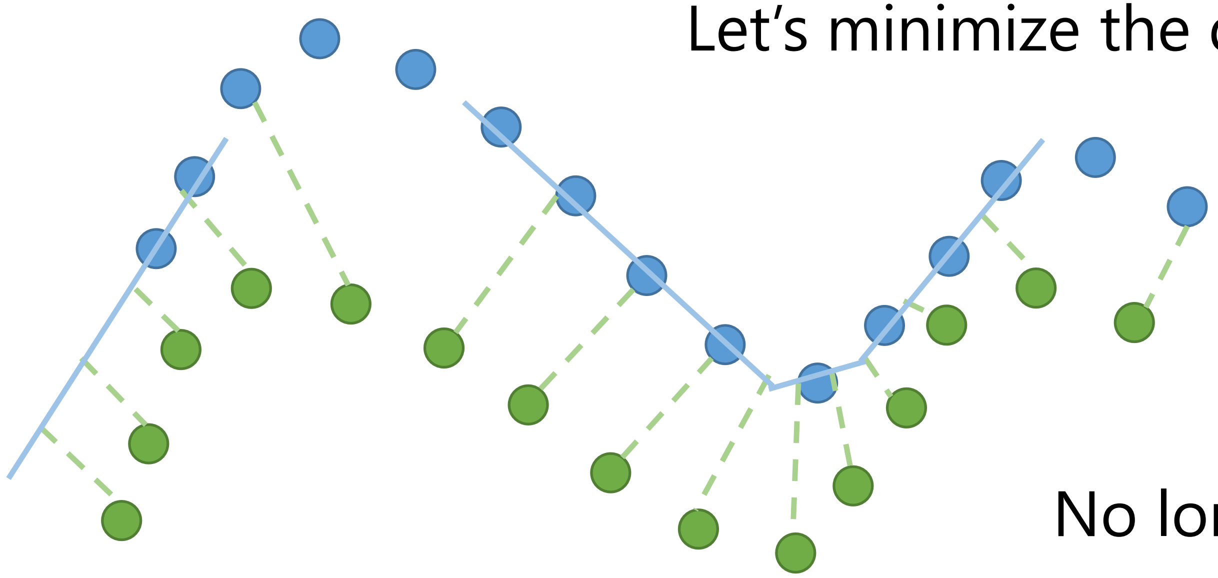


But: Uneven Sampling

Ideally, 1:1 correspondences

ICP Analysis

- How to select correspondences?



Let’s minimize the distance to the tangent plane!

No longer a closed form solution

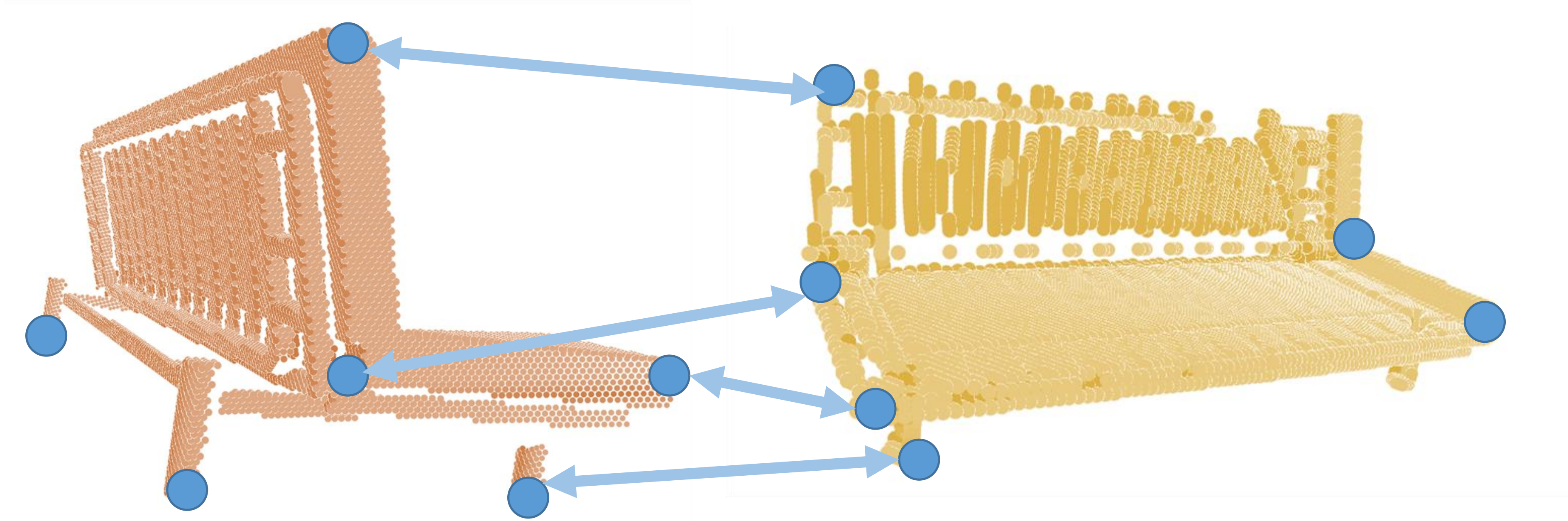
In practice: faster convergence than point-point ICP

ICP

- Solve for rigid transform between two surfaces
- Still used today

- Small basin of convergence
  - needs a good initialization
  - Hard to find correspondences
  - Need “sufficient” overlap

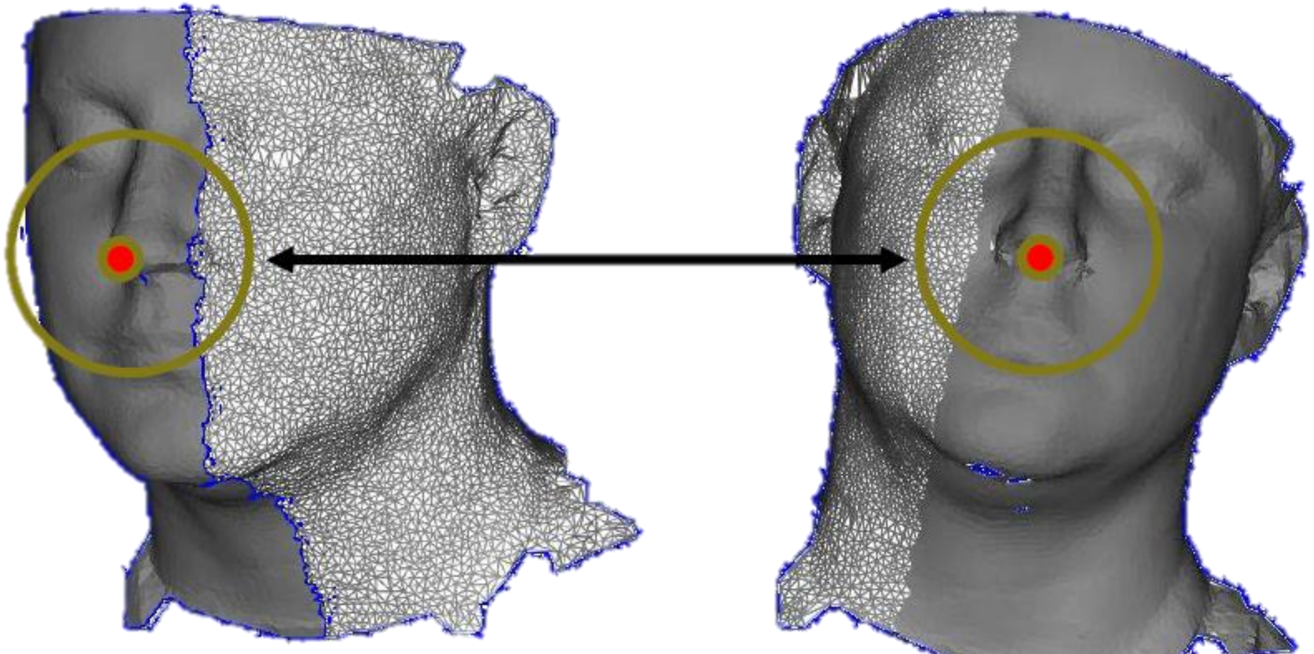
How to establish correspondences?



How to establish correspondence?

- When do two points on different shapes/scans represent the same feature?

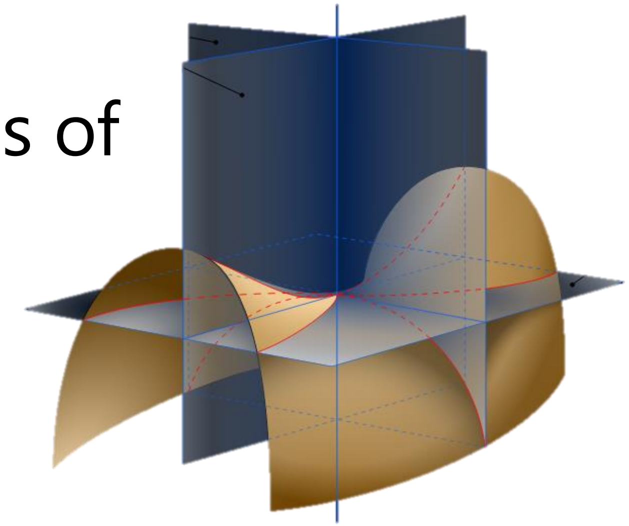
- Are the surrounding regions similar?



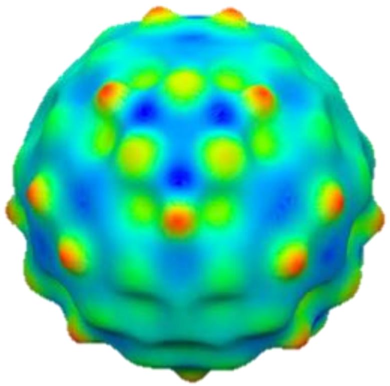
Feature descriptors summarize surrounding regions

Classical Descriptors

- Curvature
- Differential features describe characteristics of surrounding surface



- Differential features can be noisy on meshes and real-world captured data

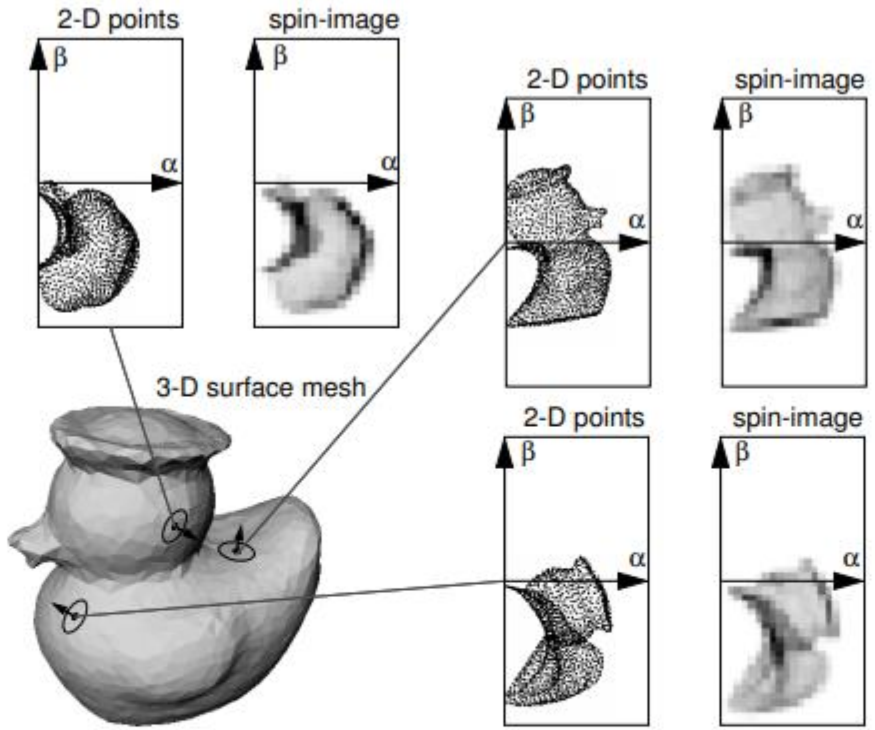


Classical Descriptors: Spin Images

- Create image associated with neighborhood of a feature point
- "Spin" image along point normal

- Collect contributions of each other point by their distance to tangent and distance to normal

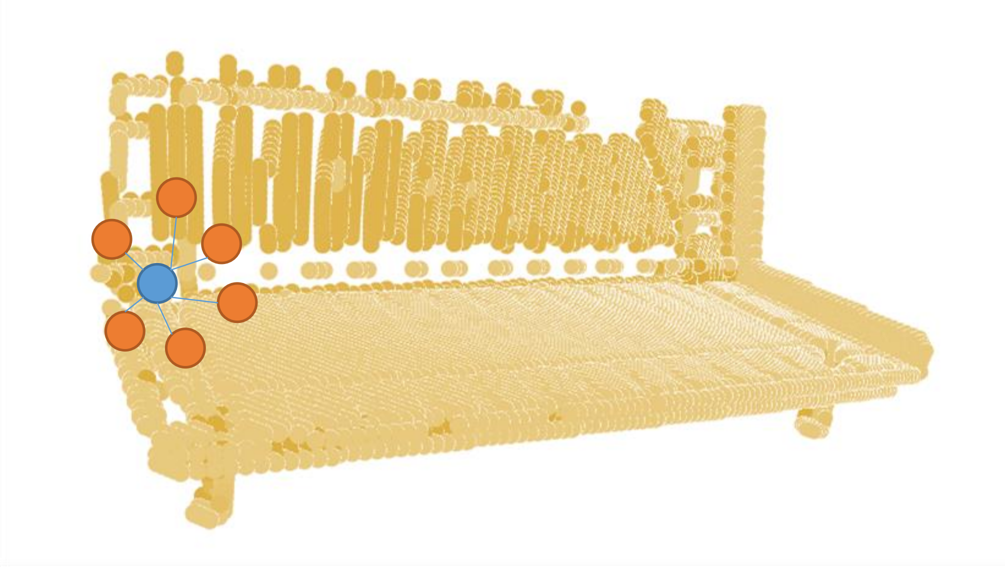
- 2D spin image comparison

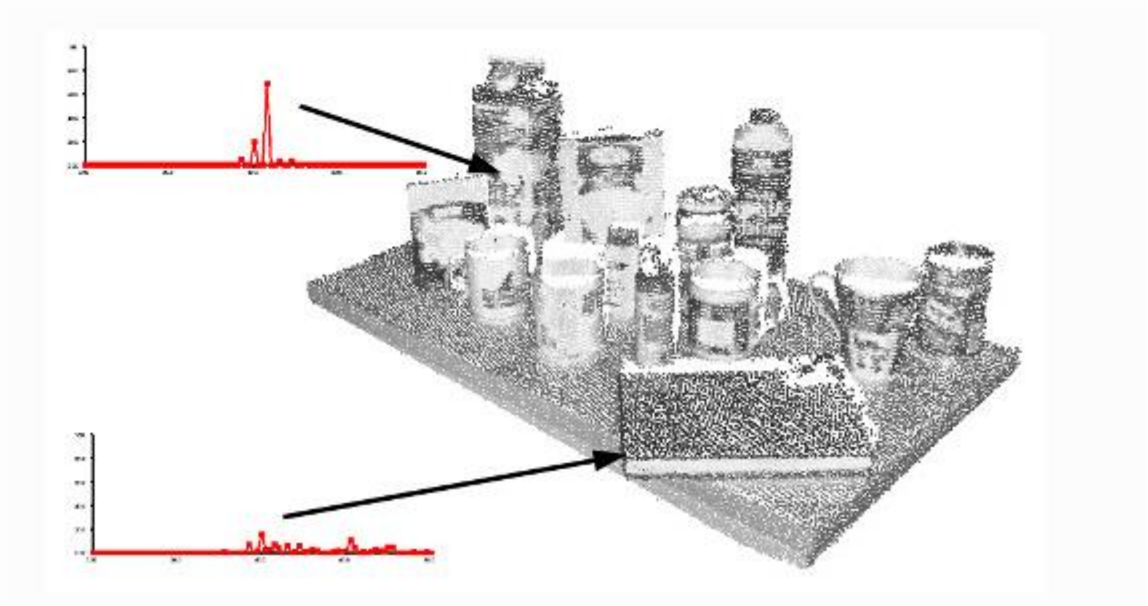


Spin Images [Johnson and Hebert '99]

Classical Descriptors: Point Feature Histograms

- For a point p find its k neighbors $\{q_i\}$
- Compute histogram from tuples of $\{(p, q_i)\}$ based on distances, normal, optionally curvature etc.



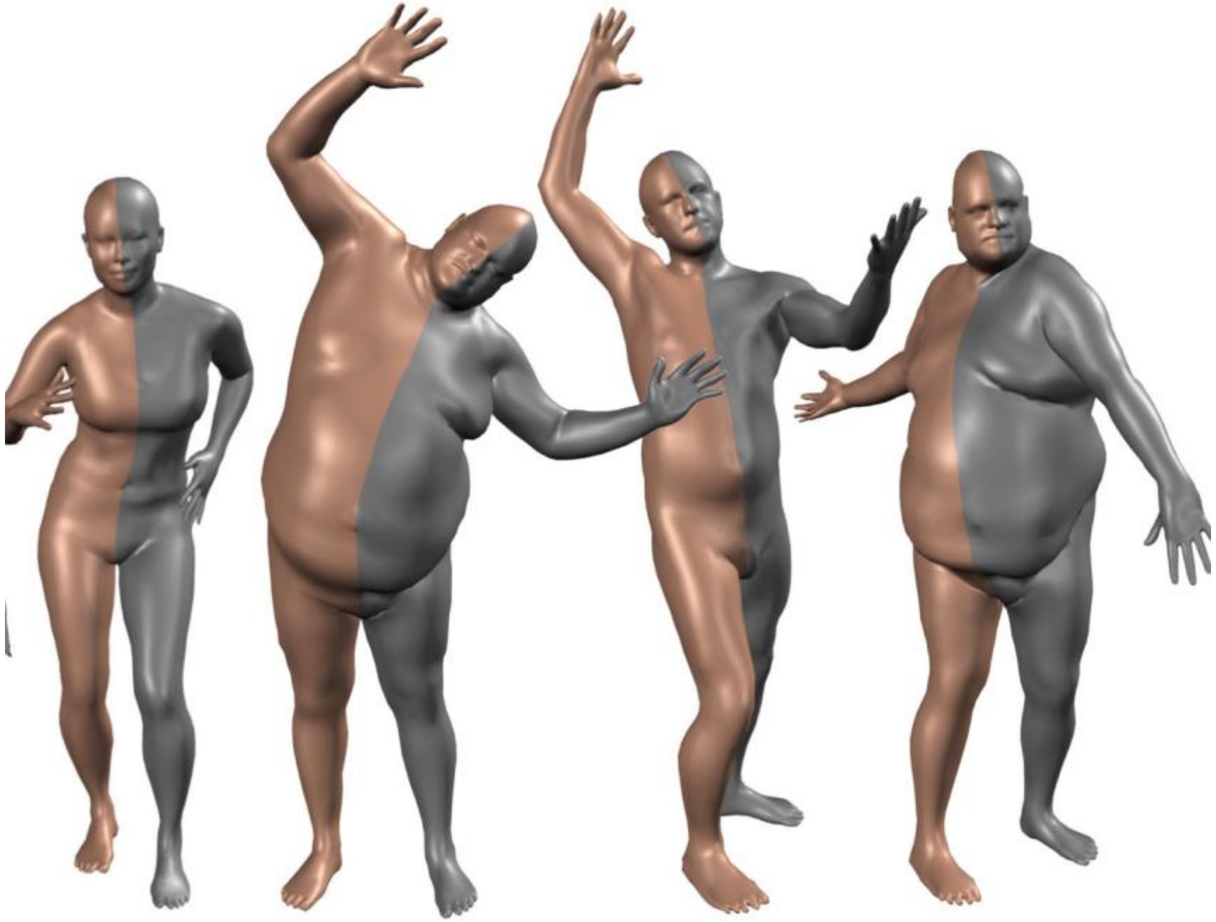


Classical Descriptors

- Often fast to compute
- Designed to be used with Euclidean distances, can use accelerated search structures for fast matching

- In practice: often have RGB images
    - Use classical RGB features (e.g., SIFT, SURF, etc.)

Non-rigid Deformations



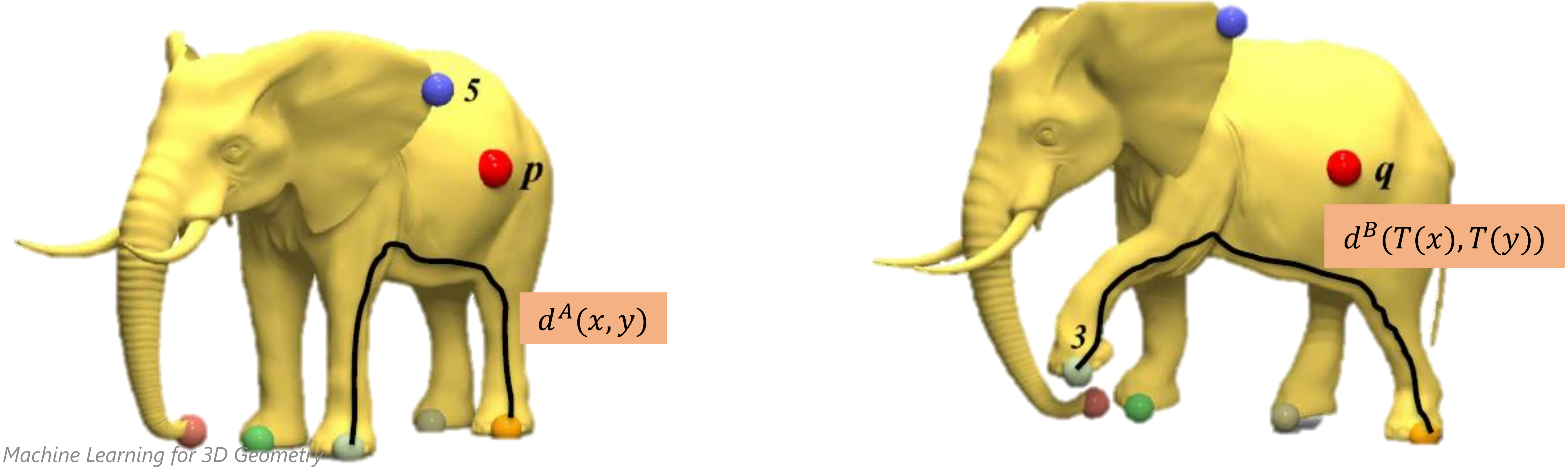
Rigid vs Deformable Registration

- Rigid: uniformly applied rigid transformation
  - Rotation, translation
  - Maintains same shape and size

- Deformable: allows non-uniform mapping between surfaces
  - Applications: to correct for small distortions in measurement
  - Modeling non-rigid shape movement over time

Non-Rigid Shape Matching

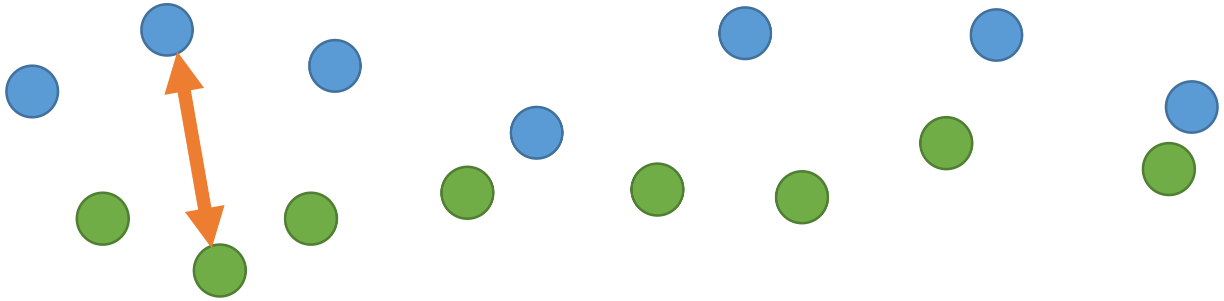
- Consider near-isometric cases
- Find correspondences that preserve intrinsic (geodesic) distances on the shapes



Measuring intrinsic shape similarity

- Gromov-Hausdorff distance

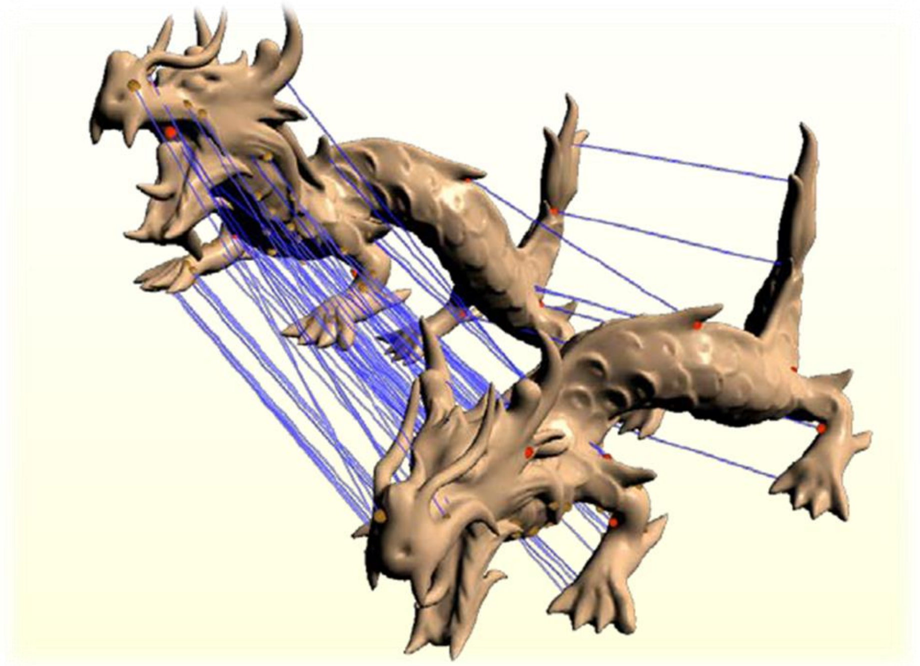
- Hausdorff distance: maximin
  - Maximum of all minimum distances between two sets of points

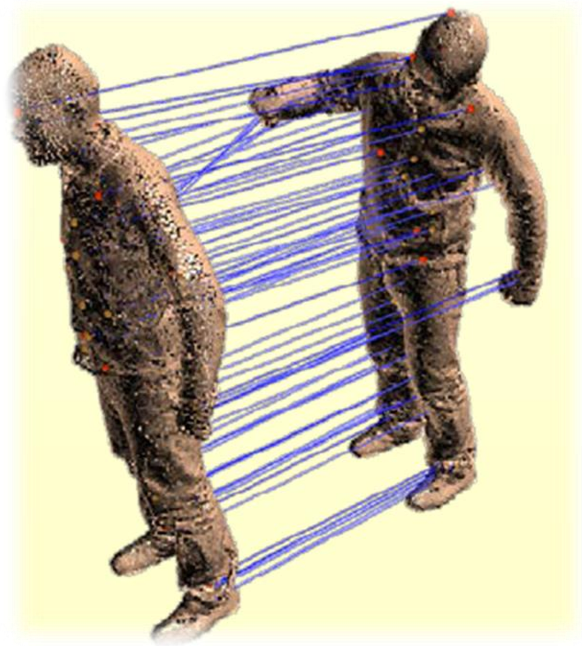


- Gromov-Hausdorff: infimum of all Hausdorff distances over mappings or correspondences
  - Over all correspondences -> difficult to compute!

Near isometries preserve local structure

- Optimal alignment can be defined, but difficult to compute
- Define descriptors of local regions -> establish good mappings

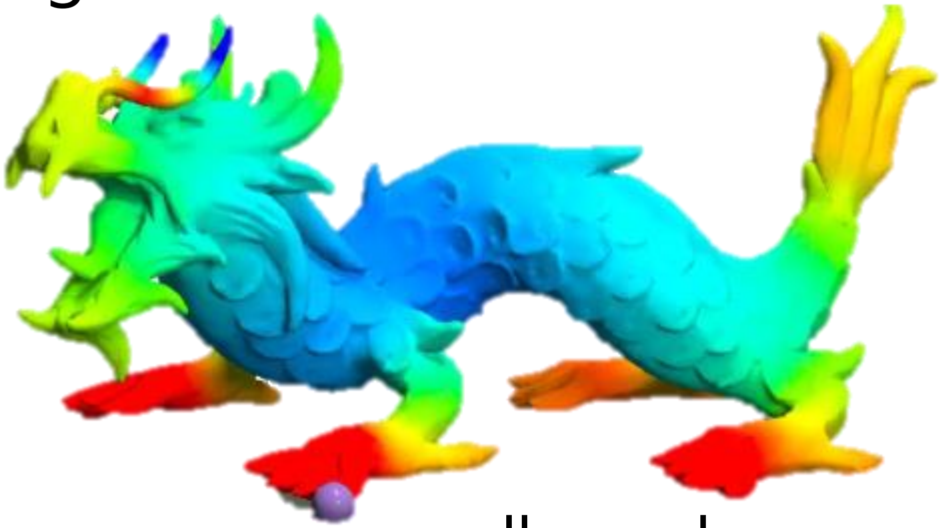




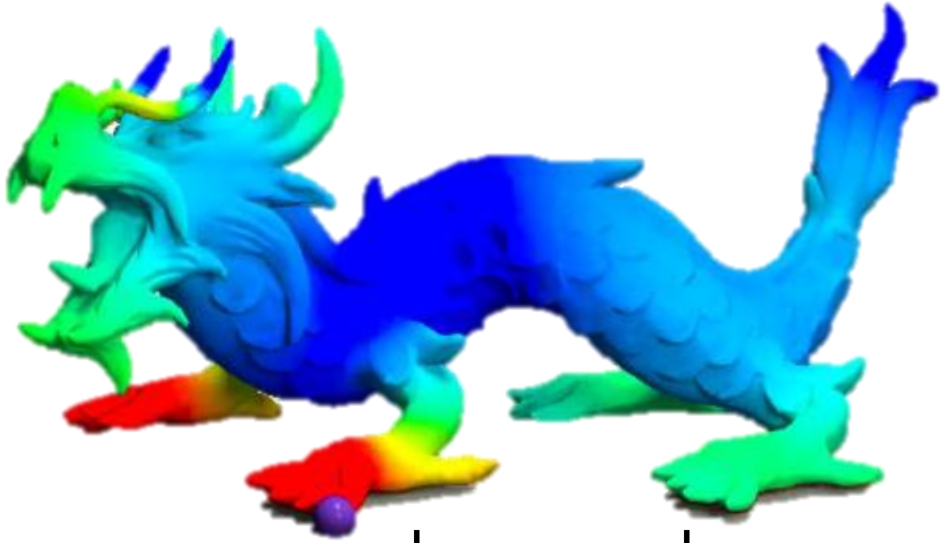
How large should a local region be?

Scale for local features?

- Given a point on a shape, find other points with similar neighborhoods



smaller scale



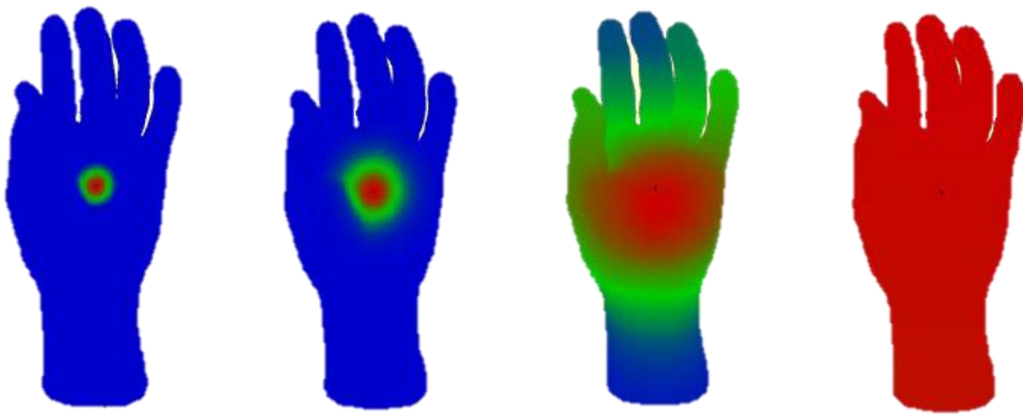
larger scale

- How to meaningfully compare neighborhoods at different scales?

Heat kernel signature

- Spectral shape analysis
- Heat kernel $k_t(x, y)$: amount of heat transferred from x to y in time t

$$f(x,t) = \int_M k_t(x,y) f(y) dy$$



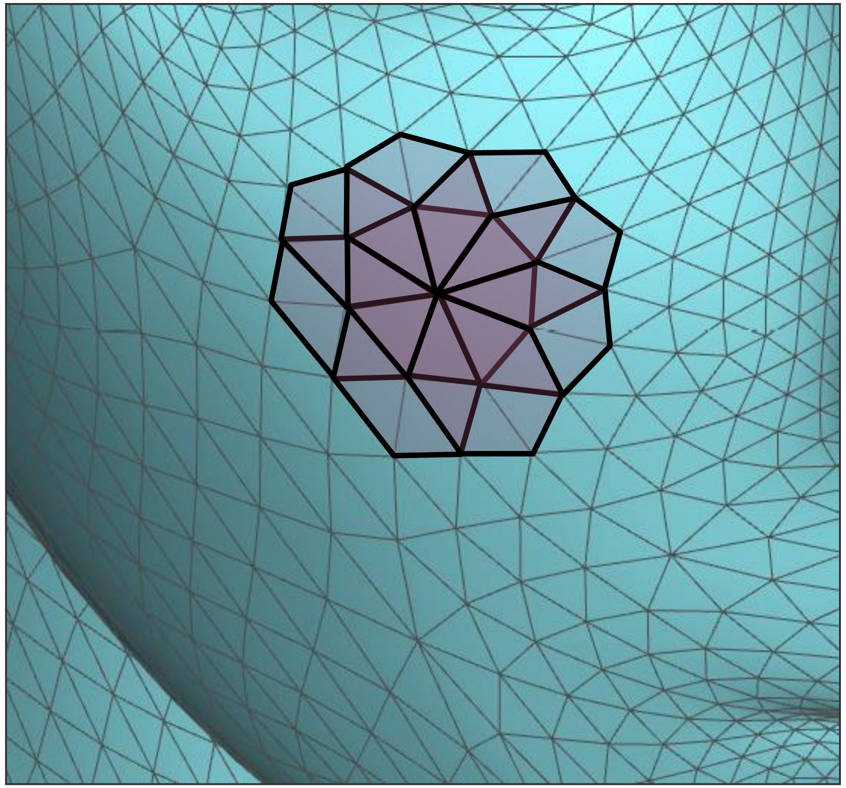
- Invariant under isometric deformations
- Multi-scale description

- In practice: can be difficult to apply to real scenarios (noisy, partial data)

[Sun et al. '09]

As-Rigid-As-Possible (ARAP)

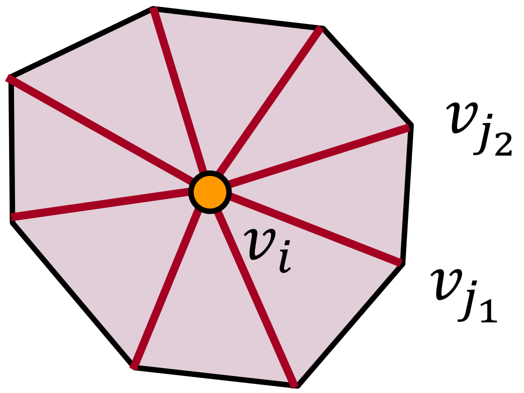
- Idea: maximize local rigidity



[Sorkine et al. ’07]

As-Rigid-As-Possible (ARAP)

- Idea: maximize local rigidity



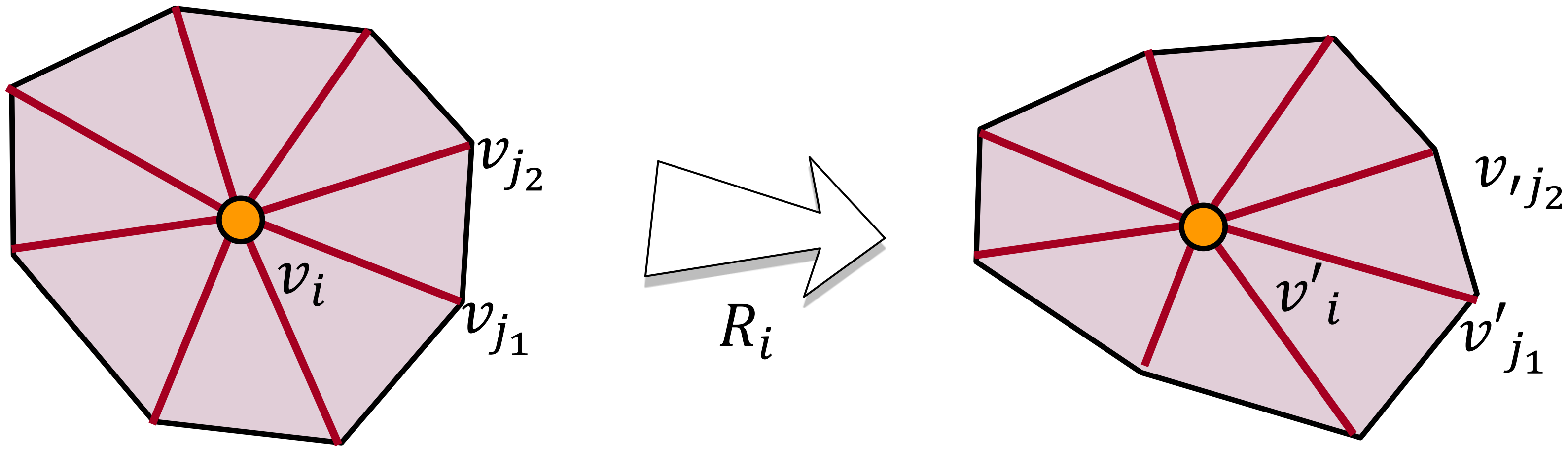
- Ask all star edges to transform rigidly by some rotation R
  - Shape of cell is preserved

$$\min \sum_{j \in N(i)} \left\| (v_i' - v_j') - R_i (v_i - v_j) \right\|^2$$

[Sorkine et al. '07]

As-Rigid-As-Possible (ARAP)

- If v, v' are known, then R_i is uniquely defined



- Build covariance matrix $S = VV'^T$
- SVD: $S = U\Sigma W^T, R_i = UW^T$

[Sorkine et al. '07]

ARAP Deformation Energy

$$\min \sum_{i=1}^{n} \sum_{j \in N(i)} \left\| \left(v_i' - v_j'\right) - R_i\left(v_i - v_j\right) \right\|^2$$

- Variables: v', R

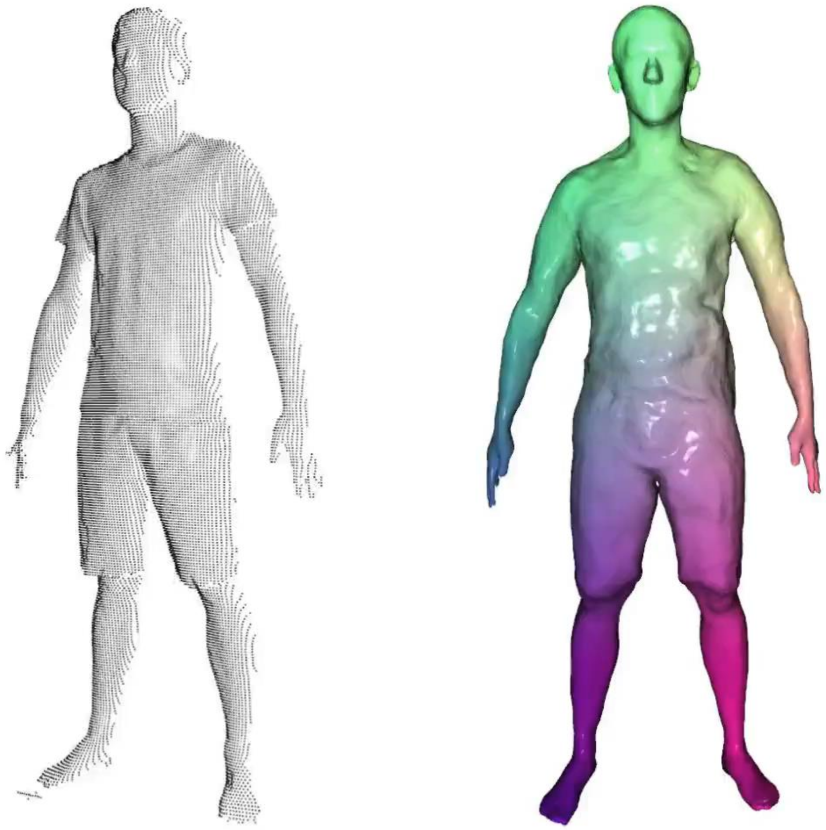
- Can iteratively optimize alternating solving for v', R
  - Solve for R_i with fixed v, v_i
  - With fixed R_i, minimize energy by finding $v'\colon Lv' = b$

ARAP

- Good at preserving edge lengths
- Fast on small meshes
- No understanding of volume
- Slow propagation on large meshes

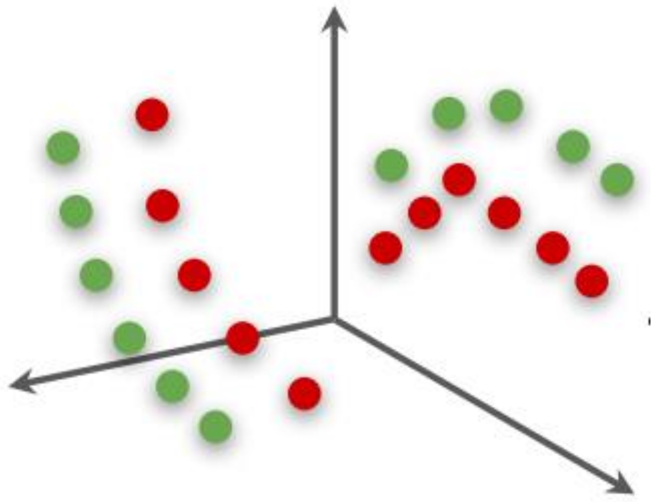
Scene Flow

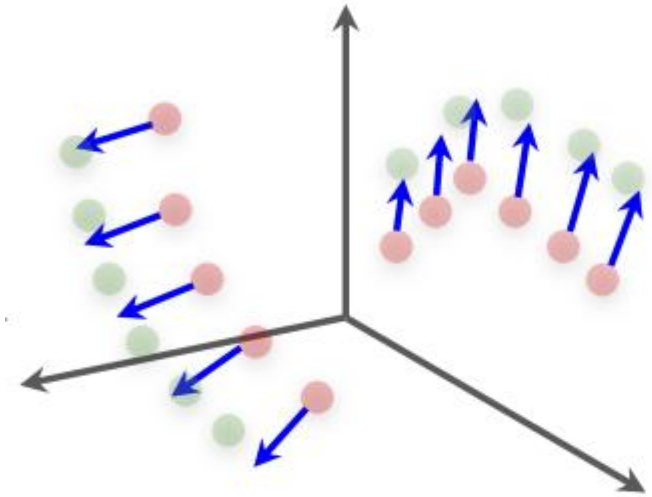
- 3D motion field of points in the world



Scene Flow

- Between two 3D observations P_t, P_{t+1}
- Flow is vector field Δ_{t}; for each point in P_t defining the motion to P_{t+1}
- $P_{t+1}(x, y, z) = P_t(x + \Delta x, y + \Delta y, z + \Delta z)$





Additional References

- Efficient variants of the ICP algorithm [Rusinkiewicz et al. ‘01]
  - http://www.pcl-users.org/file/n4037867/Rusinkiewicz_Effcient_Variants_of_ICP.pdf
- As-Rigid-As-Possible Surface Modeling [Sorkine et al ‘07]
  - https://igl.ethz.ch/projects/ARAP/