Friedrich-Alexander-Universität
Technische Fakultät

Lecture 11, Part 1

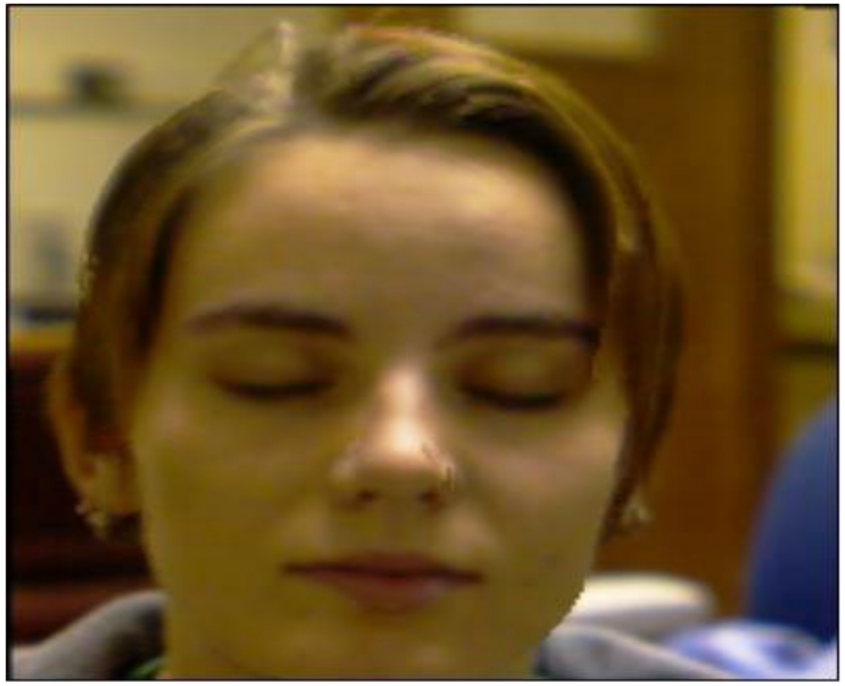
Surface Reconstruction 1/2

Computer Vision
Summer Semester 2023

Prof. Bernhard Egger, Prof. Tim Weyrich, Prof. Andreas Maier

Portable Laser Scanner

Minolta VIVID 910
3D Laser Scanner



Surface Reconstruction – Task Statement

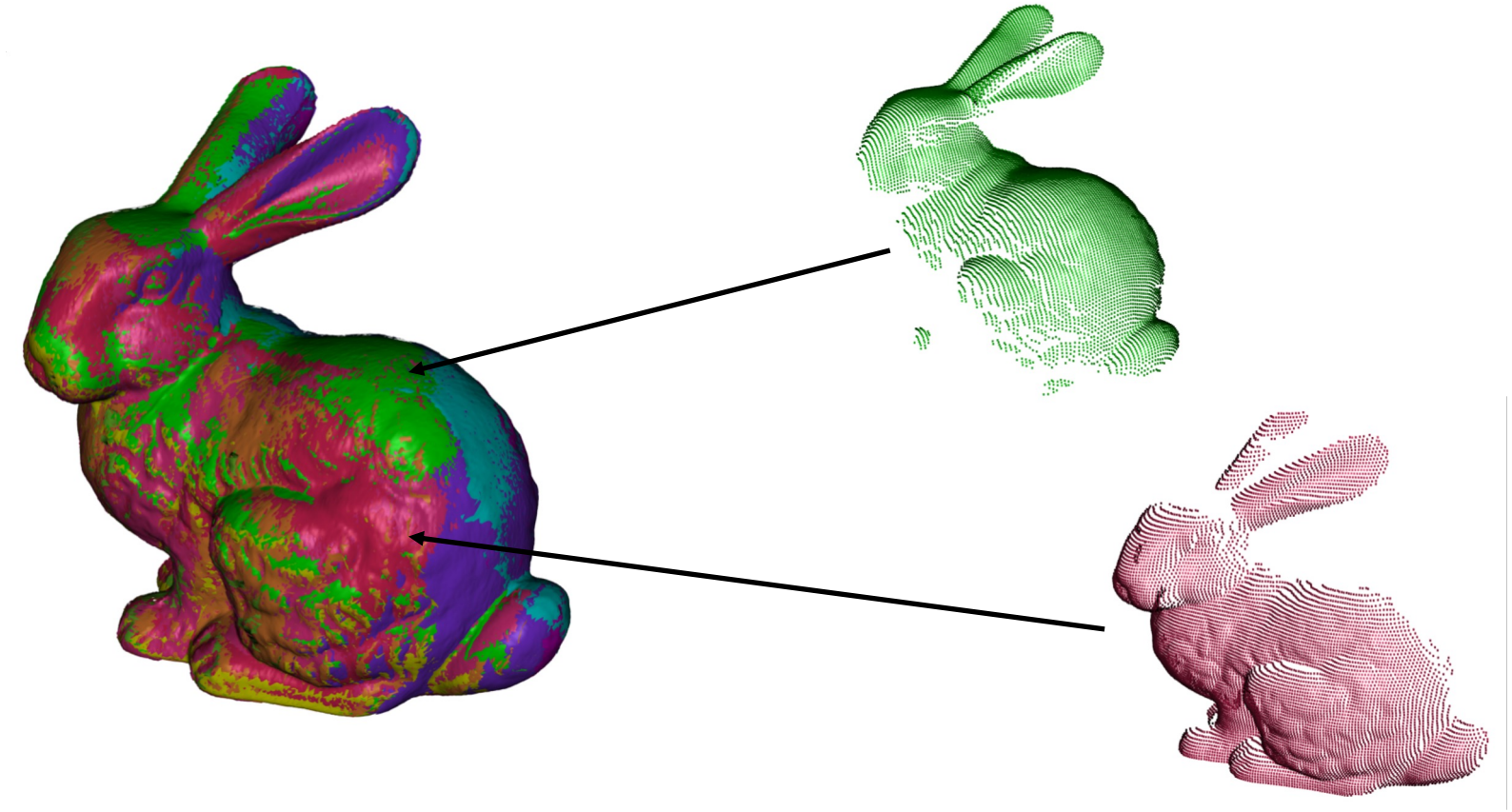
- Given: dense set of 3-D points
  - from multiple scans or
  - from a stream of depth images or
  - reconstructed from multi view stereo
  - or …

- Challenge:
  - reconstruct (closed) surface of the object

- Slides based on
  - Lecture Slides "Geometry Processing", Summer term 2017
  - Hugues Hoppe: "Poisson surface reconstruction"

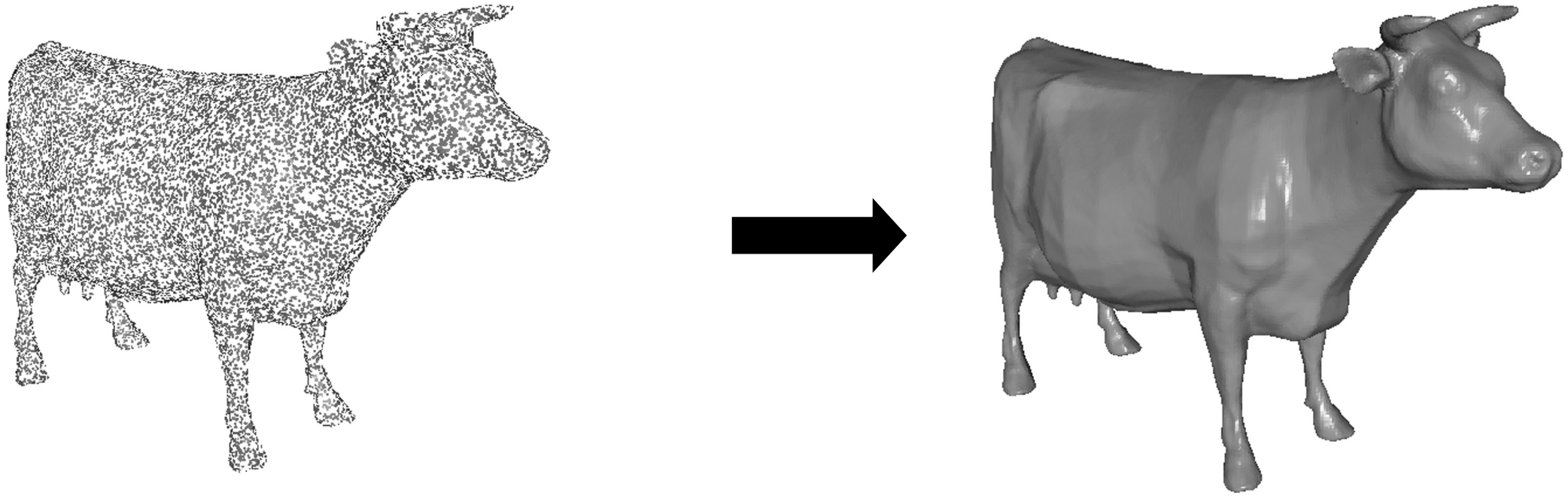
Surface Reconstruction

Problem #1: multiple sub-scans need to be aligned



Surface Reconstruction

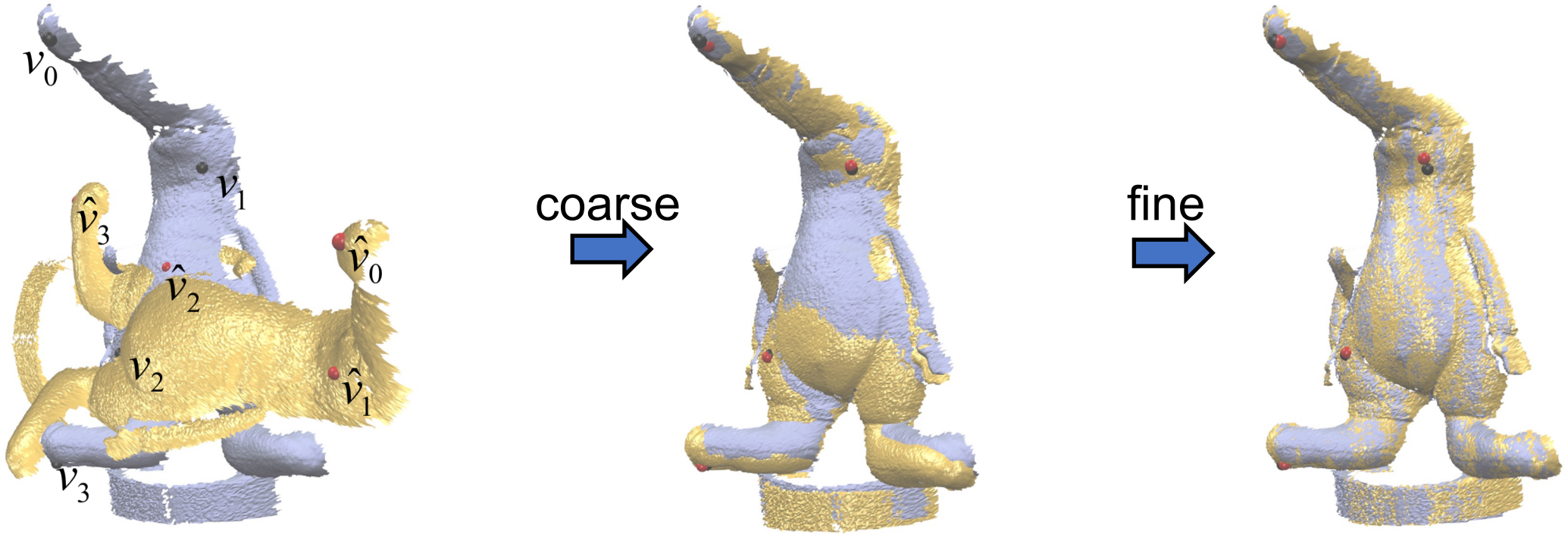
Problem #2: transform (merged) point clouds into surface



Surface Reconstruction

Problem #1: Alignment = Registration

- Step 1: Coarse Alignment
- Step 2: Fine Alignment



Coarse Alignment

- Step 1: Coarse Alignment
  - Often, camera path is known (roughly)
  - Or rough alignment can be determined from feature matching
  - Or is made by user
    - e.g. by clicking three corresponding points and finding rigid transformation that matches these
  - General problem:
    Given 2 sets of corresponding points $\{p_i\}$ and $\{q_i\}$, find rotation R and translation t (= rigid transformation) such that

$$\sum_i \|Rp_i + t - q_i\|^2$$

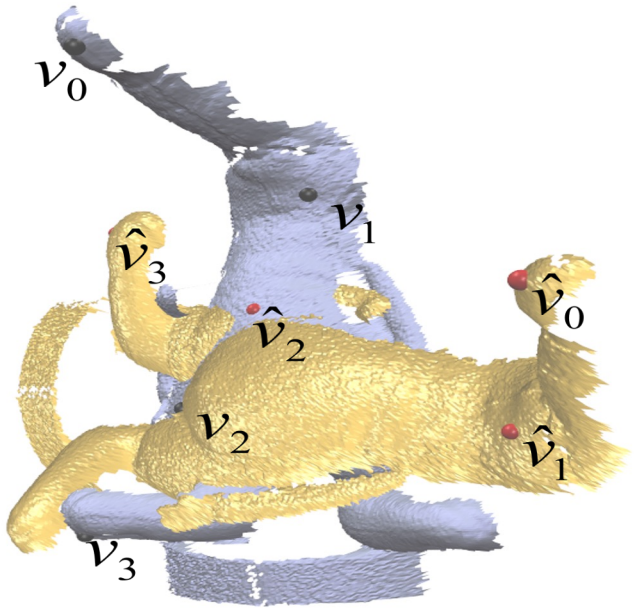
    gets minimized
    → **Procrustes** problem, can be solved using SVD

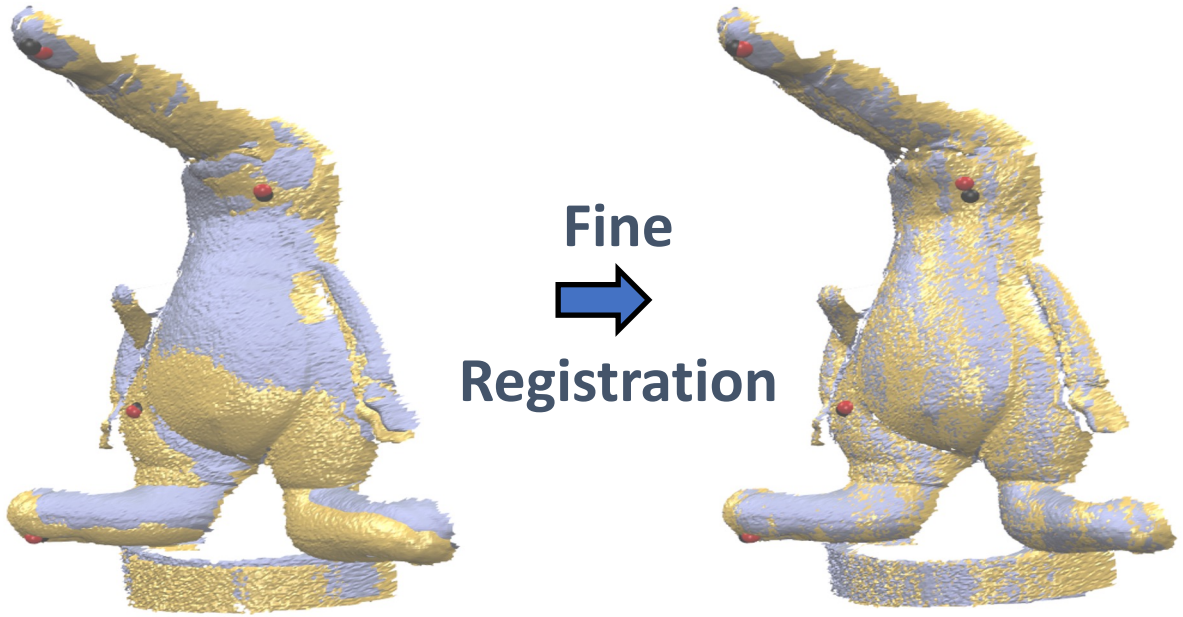
Fine Alignment

Step #2: Fine Alignment

Coarse alignment results are not perfect

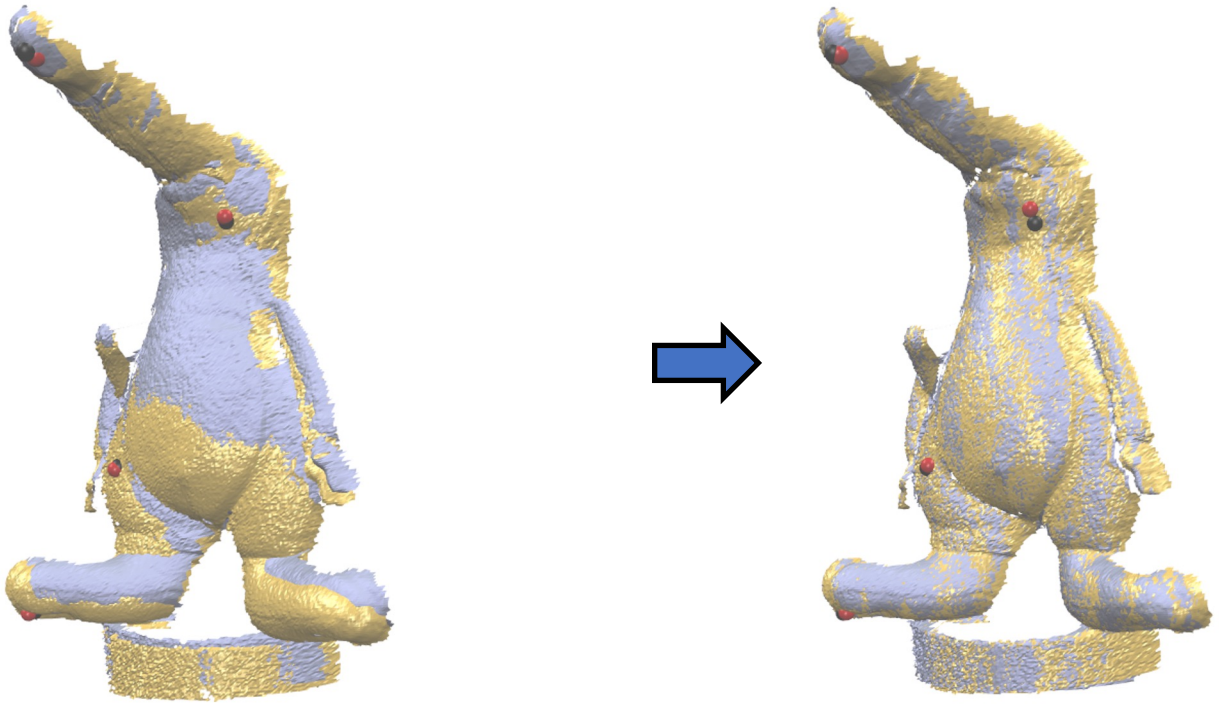
- User has not picked the exactly same points
- Automatically detected features do not perfectly match





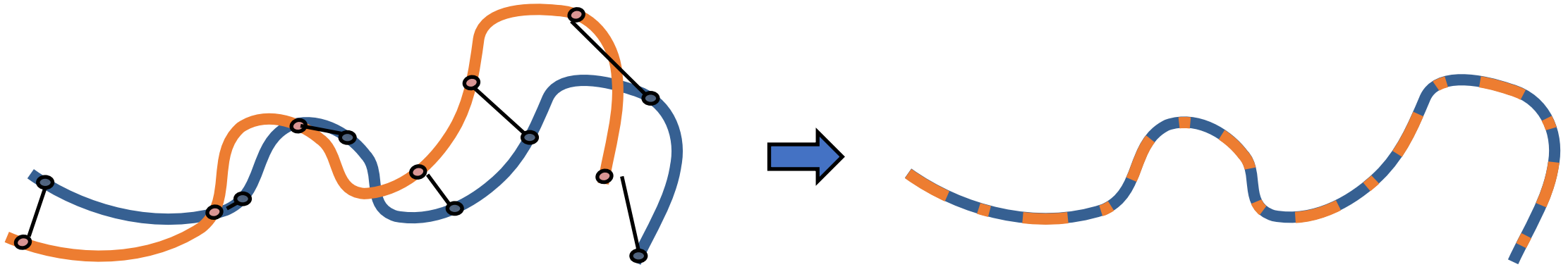
Fine Registration – Goal

- Given two partially overlapping point clouds that are roughly aligned
- Find the next best **near-optimal** alignment



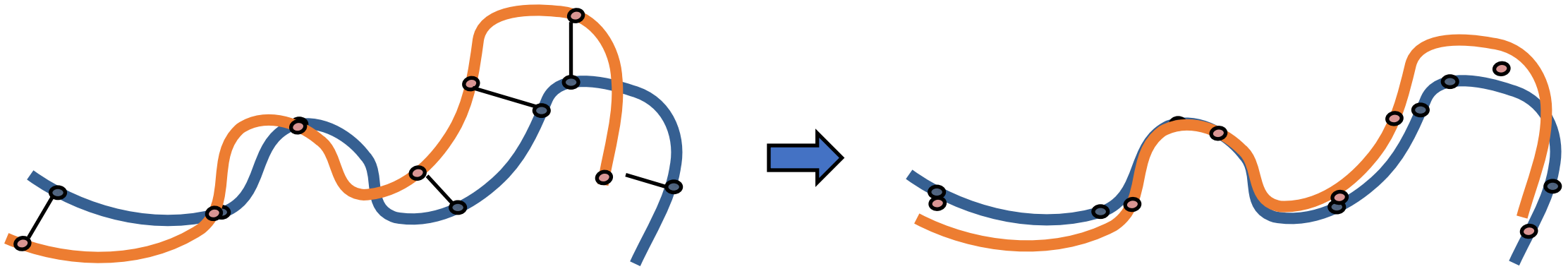
Fine Registration

- If correct correspondences are known, alignment can be found using **orthogonal Procrustes** analysis
- **Problem:** How to find correspondences?
  - User input?
  - Features?



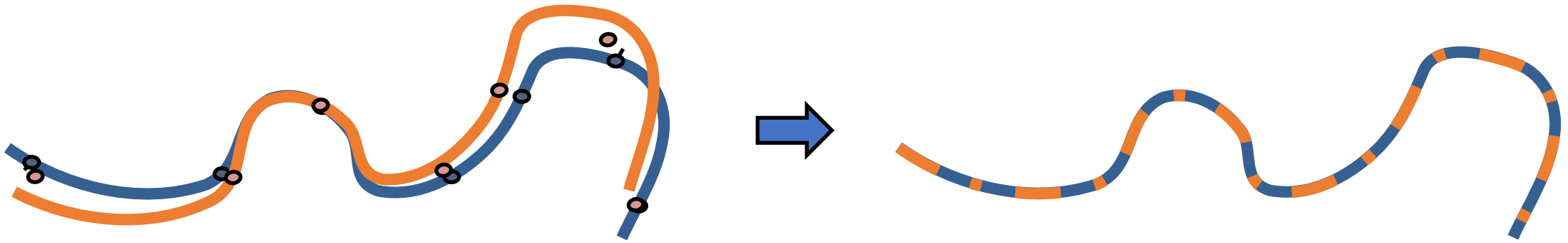
Fine Registration

- If correct correspondences are known, alignment can be found using orthogonal Procrustes analysis
- Problem: How to find correspondences?
  - User input?
  - Features?
- Alternative: assume **closest points** correspond



Fine Registration

- … and iterate
  → Iterative Closest Points (ICP) [Besl&McKay92]
- Each iteration decreases the sum of (squared) distances between the data sets
  - Movable scan is brought closer to target scan
- Converges to a locally optimal alignment
  - Globally optimal alignment if initial alignment is "good enough"



Iterative Closest Point (ICP) Algorithm

- Given two scans P and Q
- Iterate
  1. Find pairs of closest points (p_i, q_i)
     Use *k*D-tree to speed up search
  2. Find rotation R and translation t which minimize

$$\min_{R,t} \sum_i \|p_i - (Rq_i + t)\|^2$$

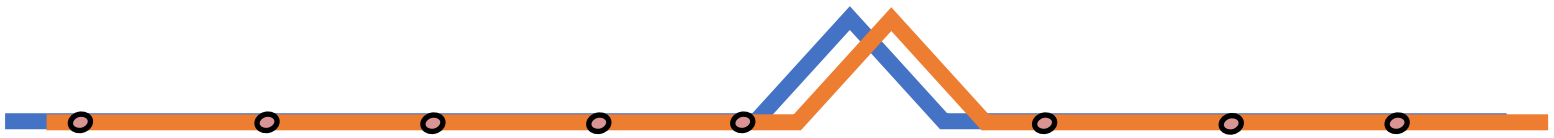
note: Rotation is minimized with the Procrustes method

- Example:

very common problem: given a set of points P, find for a new point p the nearest (or the k nearest) in P → efficient algorithms available, e.g. ANN

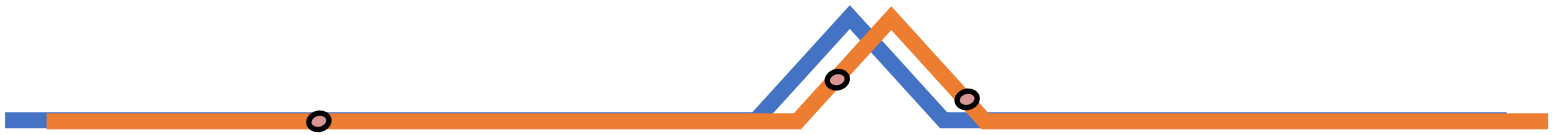
Iterative Closest Point (ICP) Algorithm

- How to pick correspondences
  - For all points in P find closest point in Q
    → Expensive
  - In practice:
    - For a subset of points in P find closest point in Q
    - How to choose the points?
      - Randomly / Uniformly (not so good)



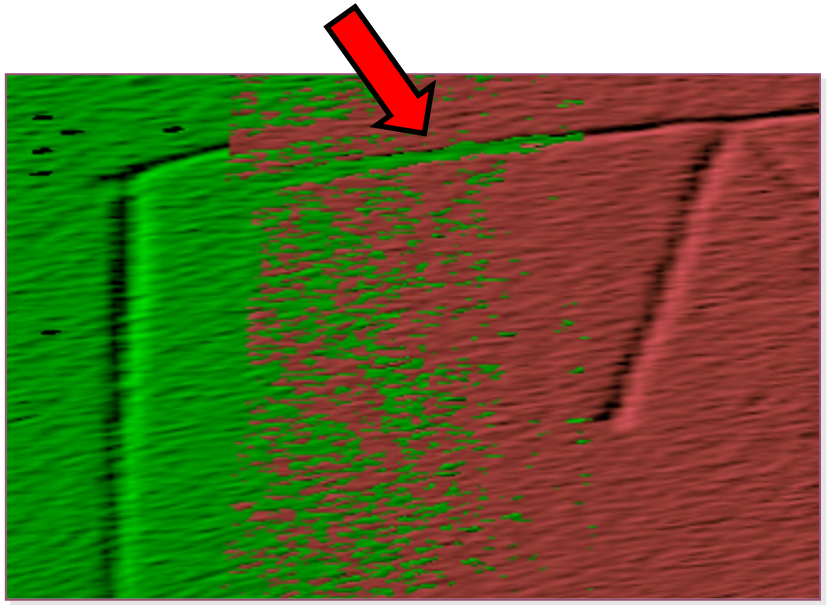
ICP Algorithm – Analysis

- How to pick correspondences
  - For all points in P find closest point in Q
    → Expensive
  - In practice:
    - For a subset of points in P find closest point in Q
    - How to choose the points?
      - Randomly / Uniformly (not so good)
      - Better: Normal space sampling
        (pick points such that points with different normals are selected)

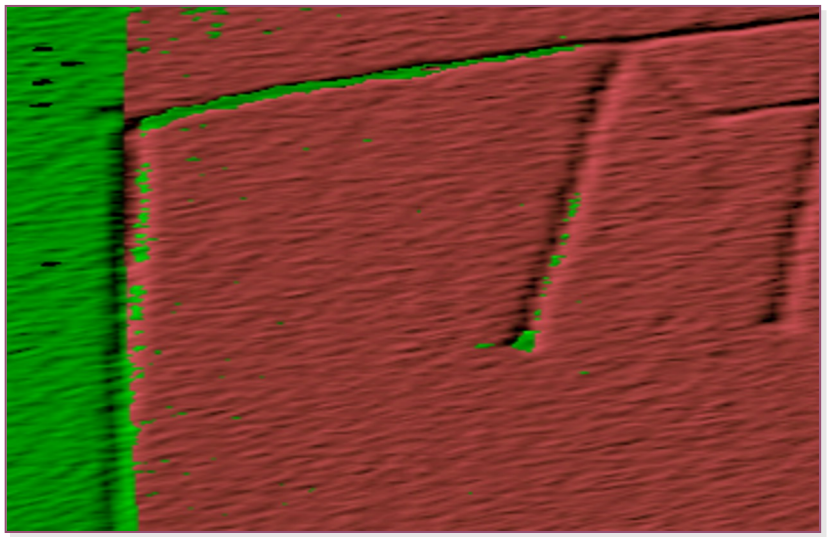


ICP Algorithm – Analysis

- How to pick correspondences
  - Normal sampling vs. random sampling



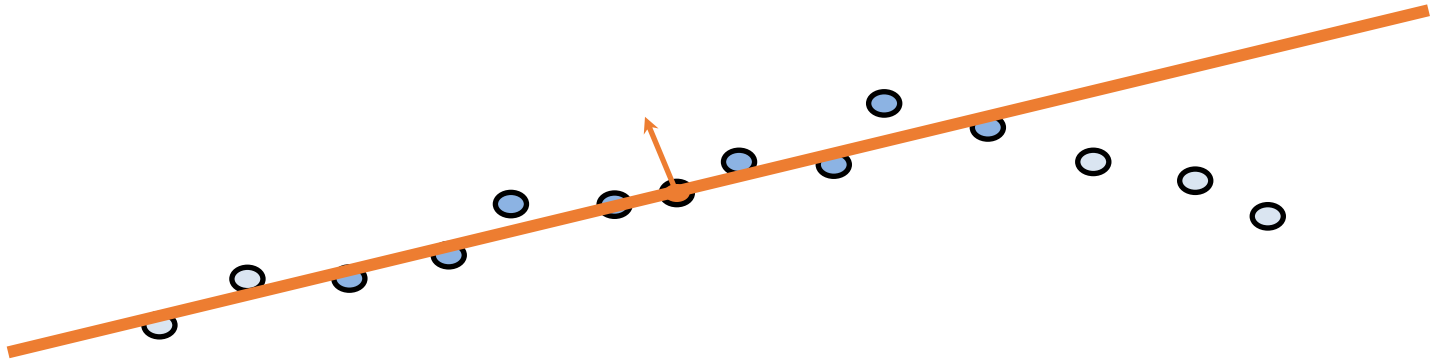
Random sampling



Normal sampling

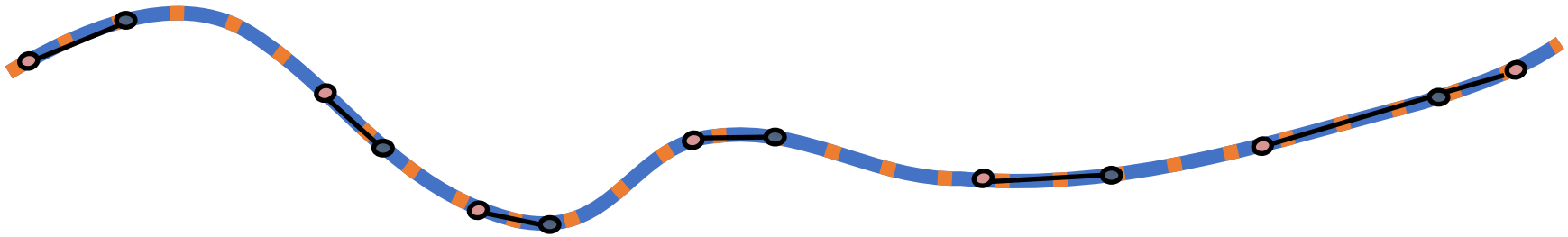
Interlude – Computing Point Normals

- Often we want to have normals with our points, how do we get these normal vectors?
- for each point
  - Compute k-nearest neighbors
  - Fit plane to neighborhood
  - Use plane normal as point normal vector



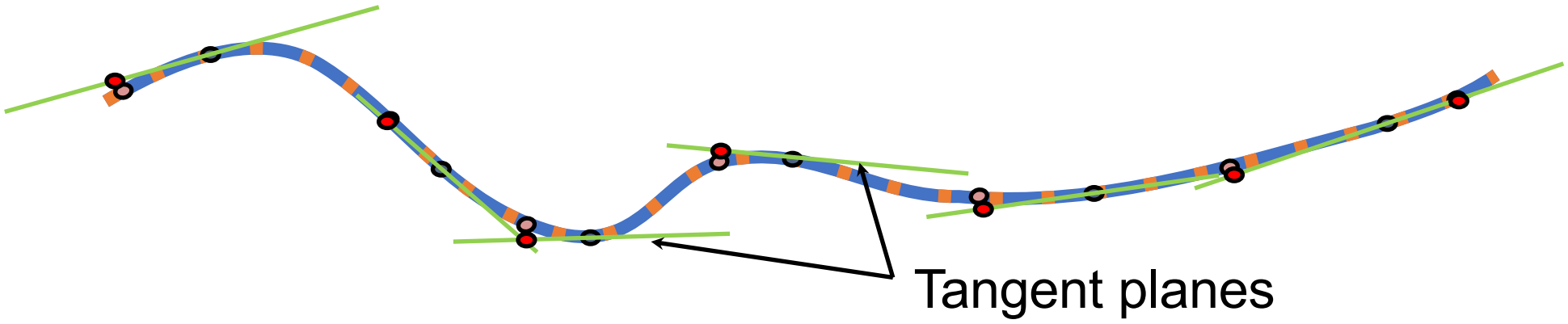
ICP Algorithm – Analysis

- In practice, datasets given by discrete points
- How to find corresponding points?
  - Closest point
    - Datasets are perfectly aligned, Error != 0



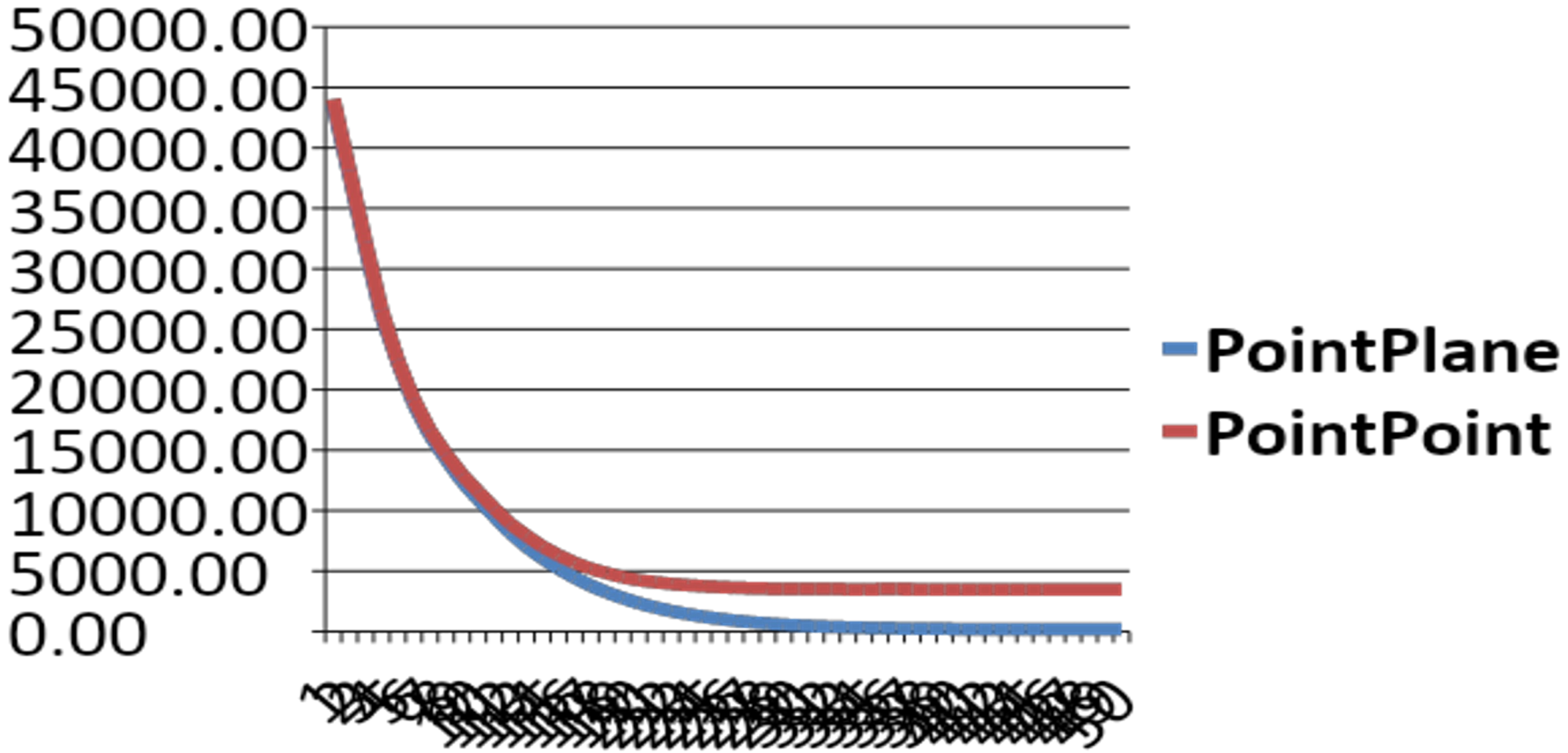
ICP Algorithm – Analysis

- In practice, datasets given by discrete points
- How to find corresponding points?
  - Closest point
  - Better: Closest point on tangent plane of closest point
    - Point normal vectors are given or can be computed
    - Convergence independent of sampling, faster convergence



ICP Algorithm – Analysis

– Comparing convergence

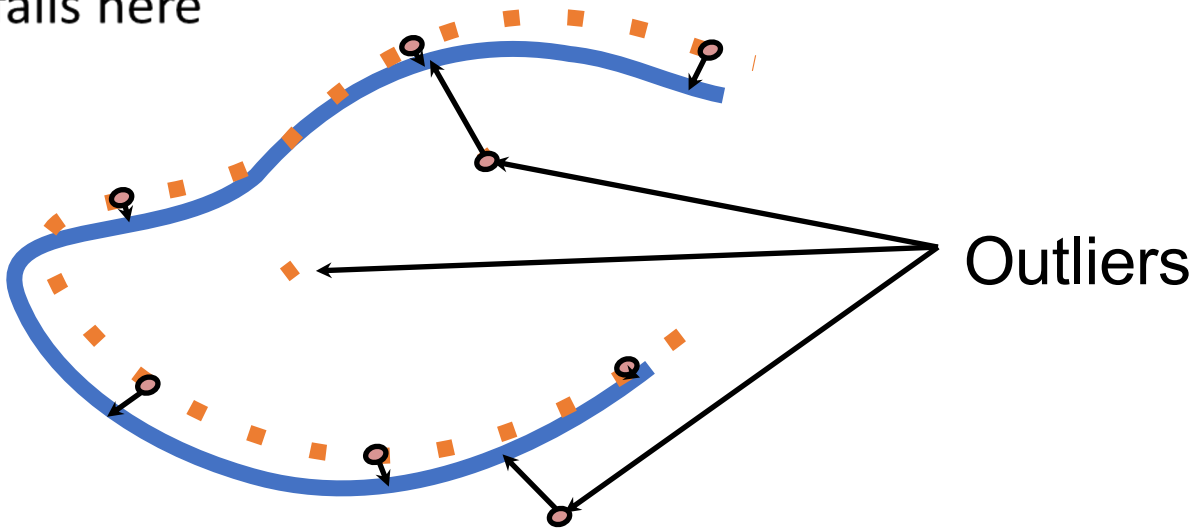


ICP Algorithm – Analysis

- Scanner data contains noise and outliers
  - Recall that we minimize the squared error!

$$\min_{R,t} \sum_i \|p_i - (Rq_i + t)\|^2$$

  - Large distances contribute extraordinary to total error
    ⟶ Least-squares fails here



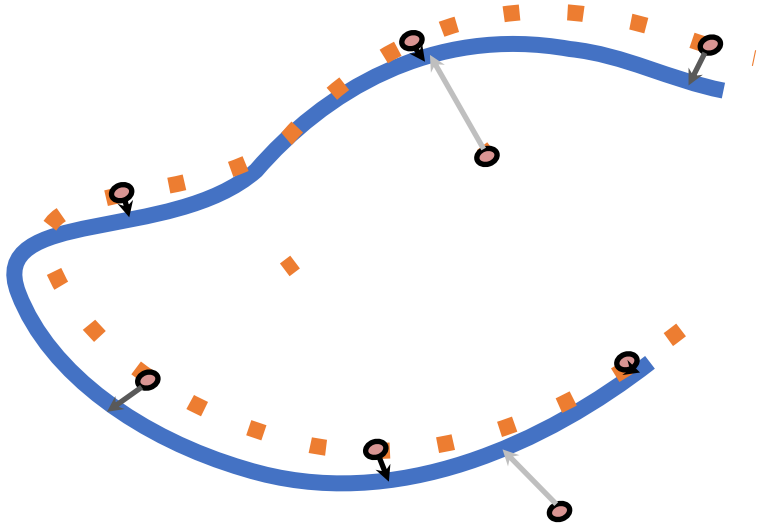
  - Prune outliers (e.g., only keep 50% closest correspondences)

ICP Algorithm – Analysis

- Scanner data contains noise and outliers
- Weight correspondences

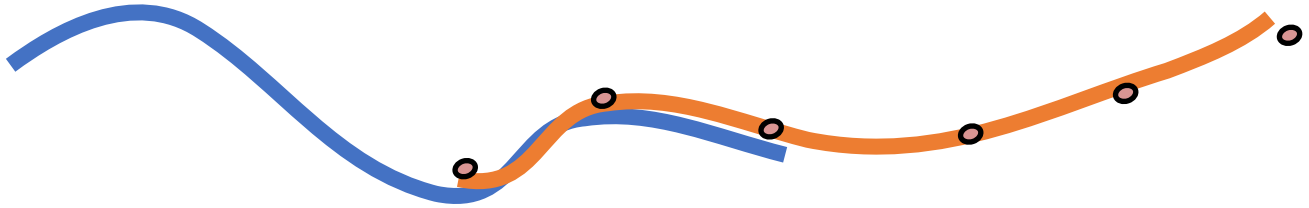
$$\min_{R,t} \sum_i w_i \|p_i - (Rq_i + t)\|^2$$

  - Good correspondences (close, same normals & curvature…) get large weights



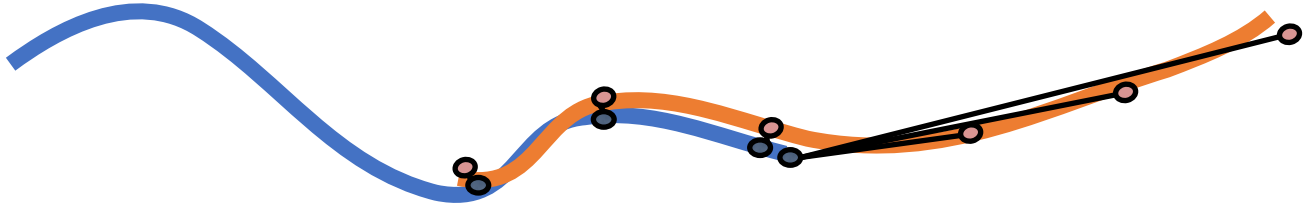
ICP Algorithm – Analysis

- In practice, scans overlap only **partially**
- Points outside the overlapping area have no meaningful correspondences!
- Example



ICP Algorithm – Analysis

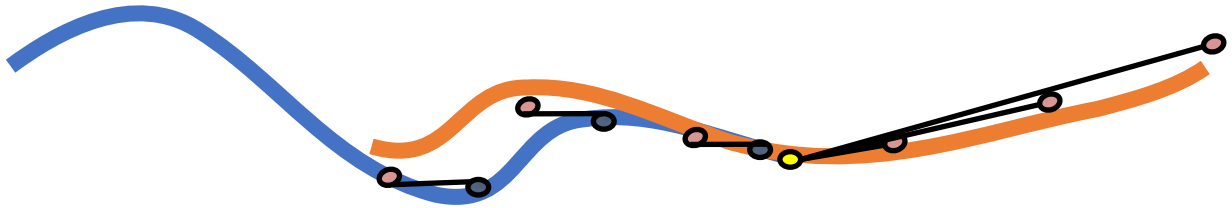
- In practice, scans overlap only **partially**
- Points outside the overlapping area have no meaningful correspondences!
- Example



- Solve for optimal rotation/translation

ICP Algorithm – Analysis

- In practice, scans overlap only **partially**
- Points outside the overlapping area have no meaningful correspondences!
- Example



- Solve for optimal rotation/translation
  - Obviously not an improvement
- Detect boundary
  - prune correspondence that map to boundary

ICP Algorithm – Wrap Up

- Iterate
  1. Select some points in source data set
  2. Find closest points in target data set (on tangent plane)
  3. Weight correspondences
  4. Reject “misleading” point pairs
     - Outliers
     - Points on the shape boundary
  5. compute R and t for the remaining correspondences
  6. Transform source data set
  7. If not converged, goto 1

References (Optional Read)

- A Method for Registration of 3-D Shapes
  - Paul J. Besl and Neil D. McKey – Transactions on Pattern Analysis and Machine Intelligence 14/2 1992
    (ICP paper, hard to read, uses Horn87 instead of Procrustes)
- Object Modeling by Registration of Multiple Range Images
  - Chen Y., Medioni G. – Image and Vision Computing 10, 3 (1992), 145–155.
- On the ICP Algorithm
  - Esther Ezra et al., SCG'06, June 2006