CS 3630!

Lecture 17: Iterated Closest Point

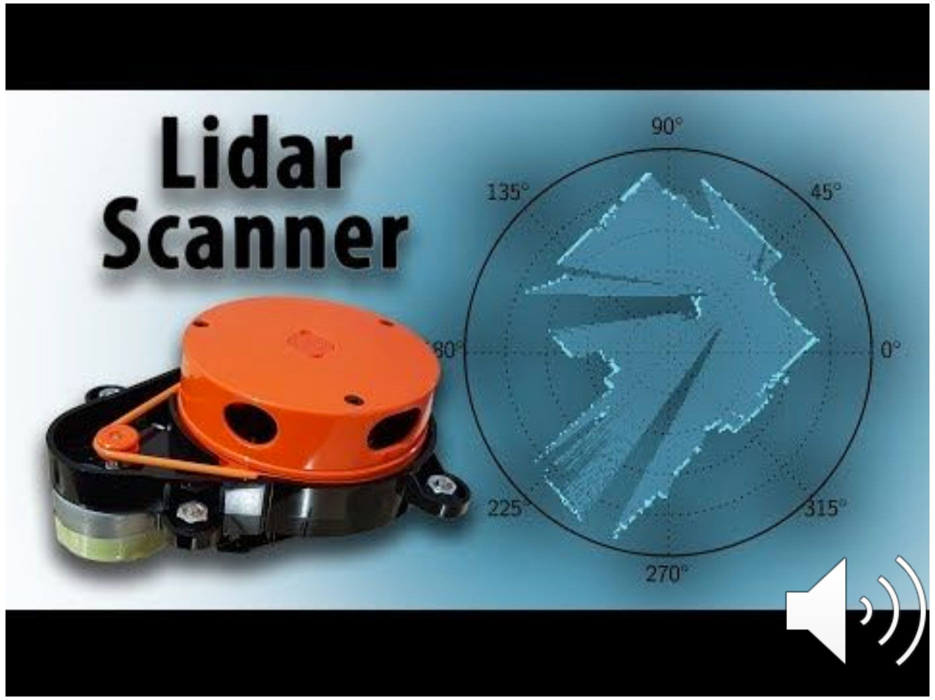
Topics

1. **Applications**
2. **Basic ICP Algorithm**
3. **ICP Variants**

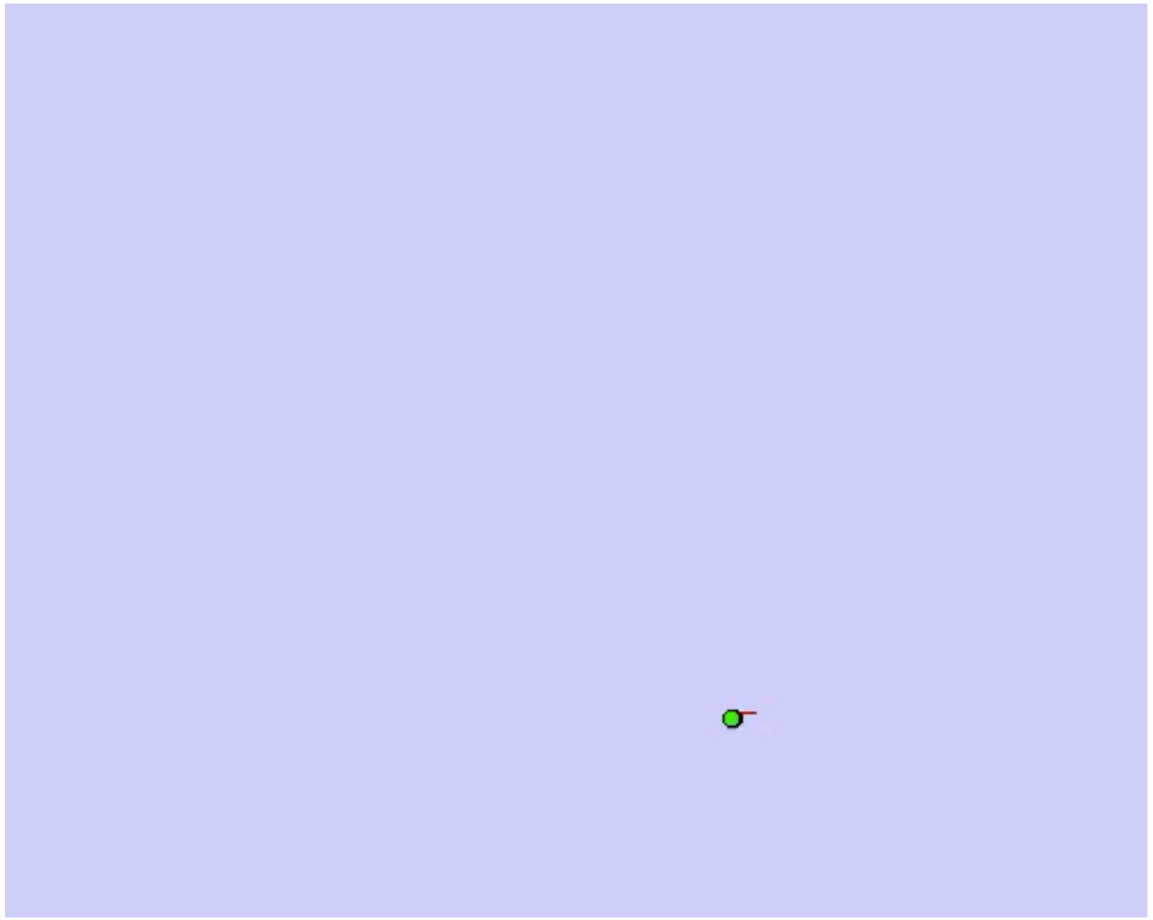
- Includes slides adapted from Marc Pollefeys and James Hayes.

Motivation

- 3D Scanners are becoming more and more prevalent
- 2D lasers now built into vacuuming robots
- Sensor of choice in autonomous vehicles
- Simple way to do SLAM

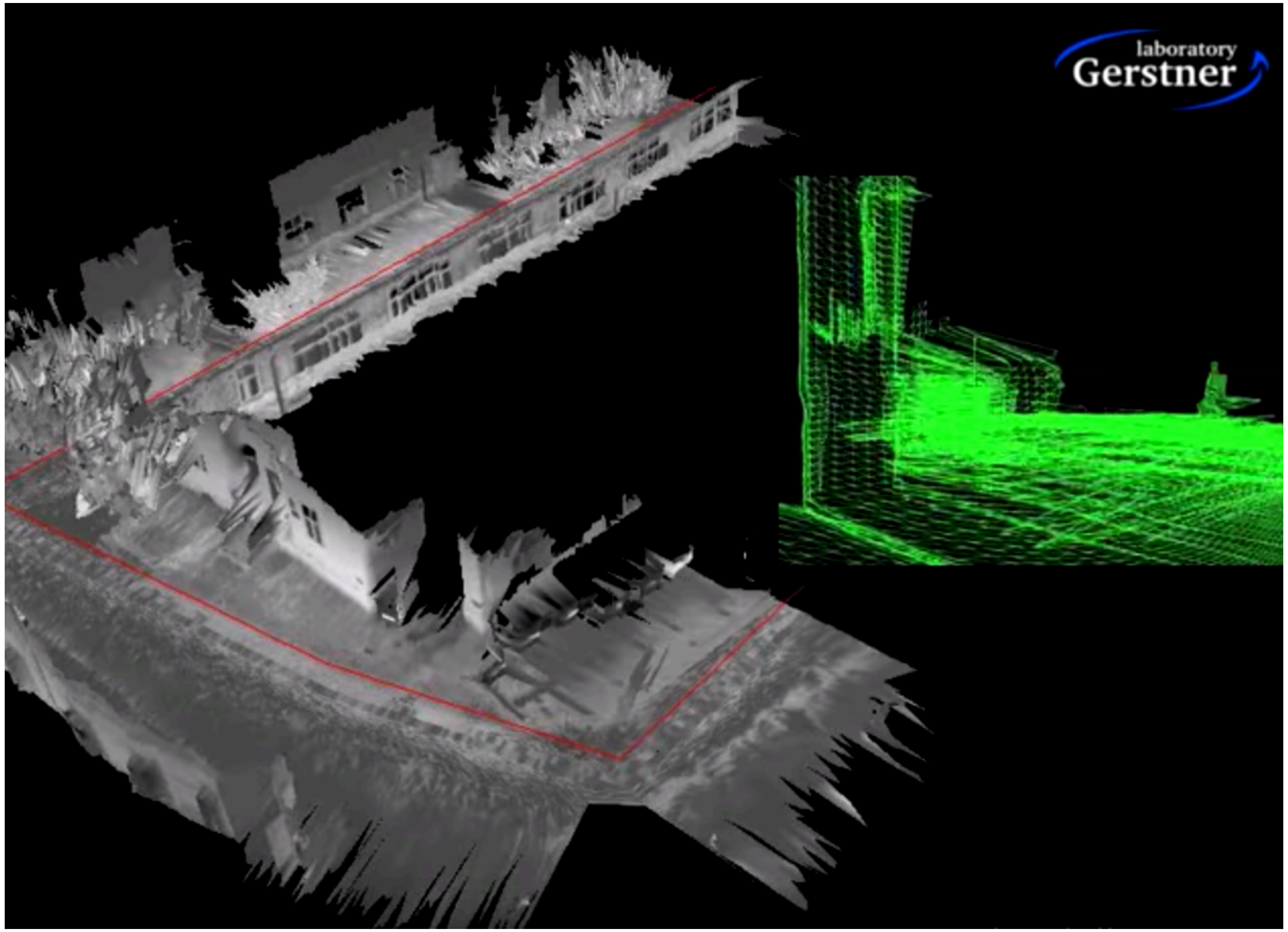


ICP in Robotics



https://www.youtube.com/watch?v=Ni8OFNyC5RY
https://www.youtube.com/watch?v=9rTkUZ7HV_o

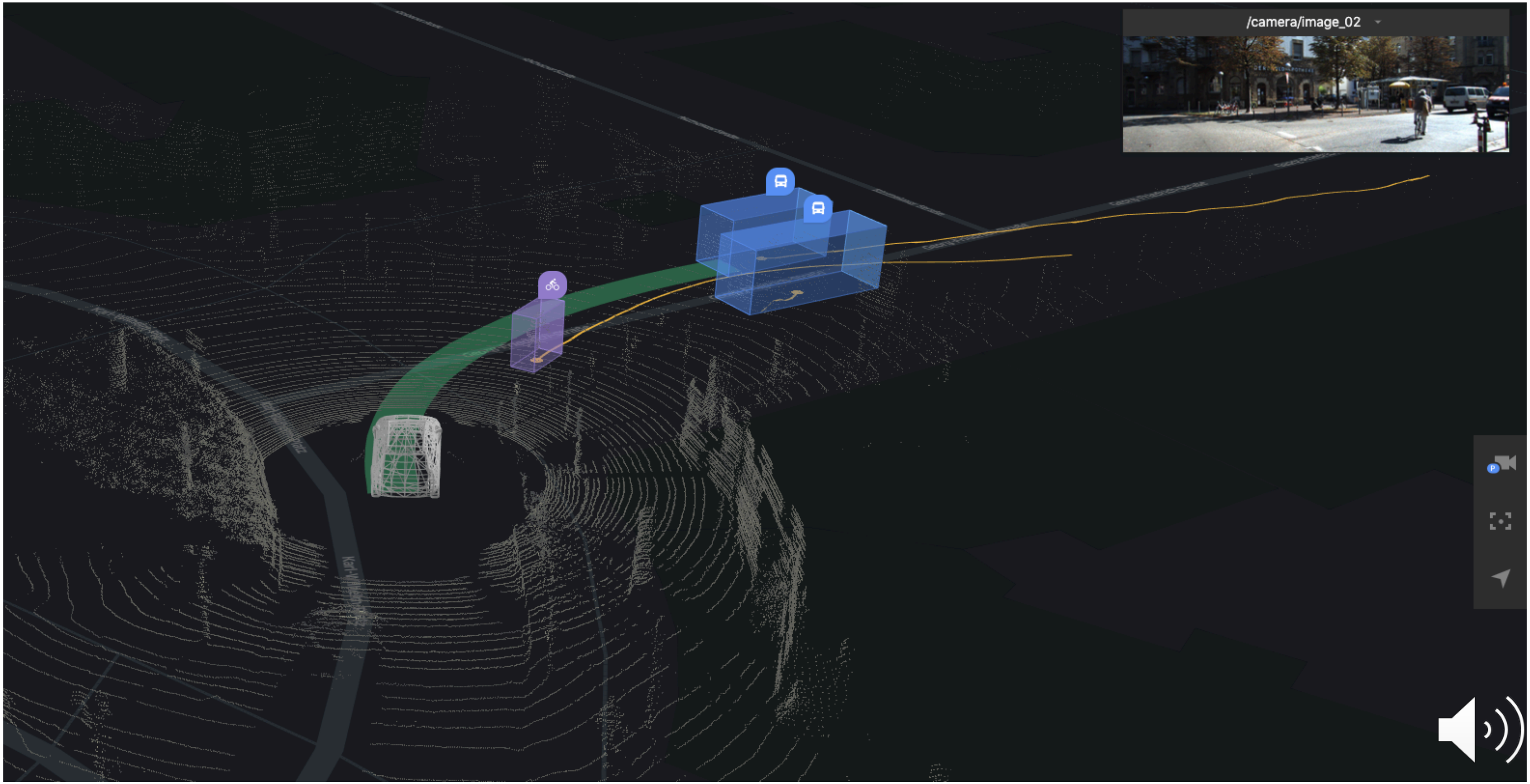
ICP Outdoors



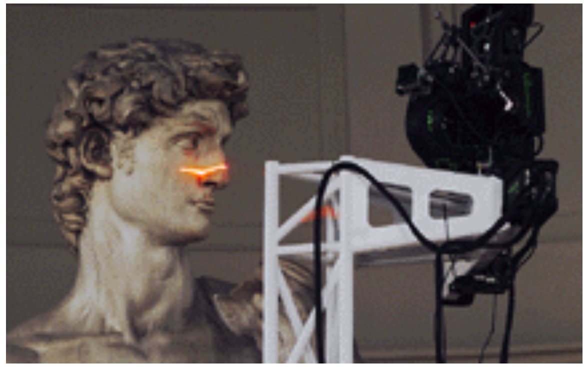
Gerstner Lab, Prague

Uber AVS

Try it live: https://avs.auto/demo/index.html



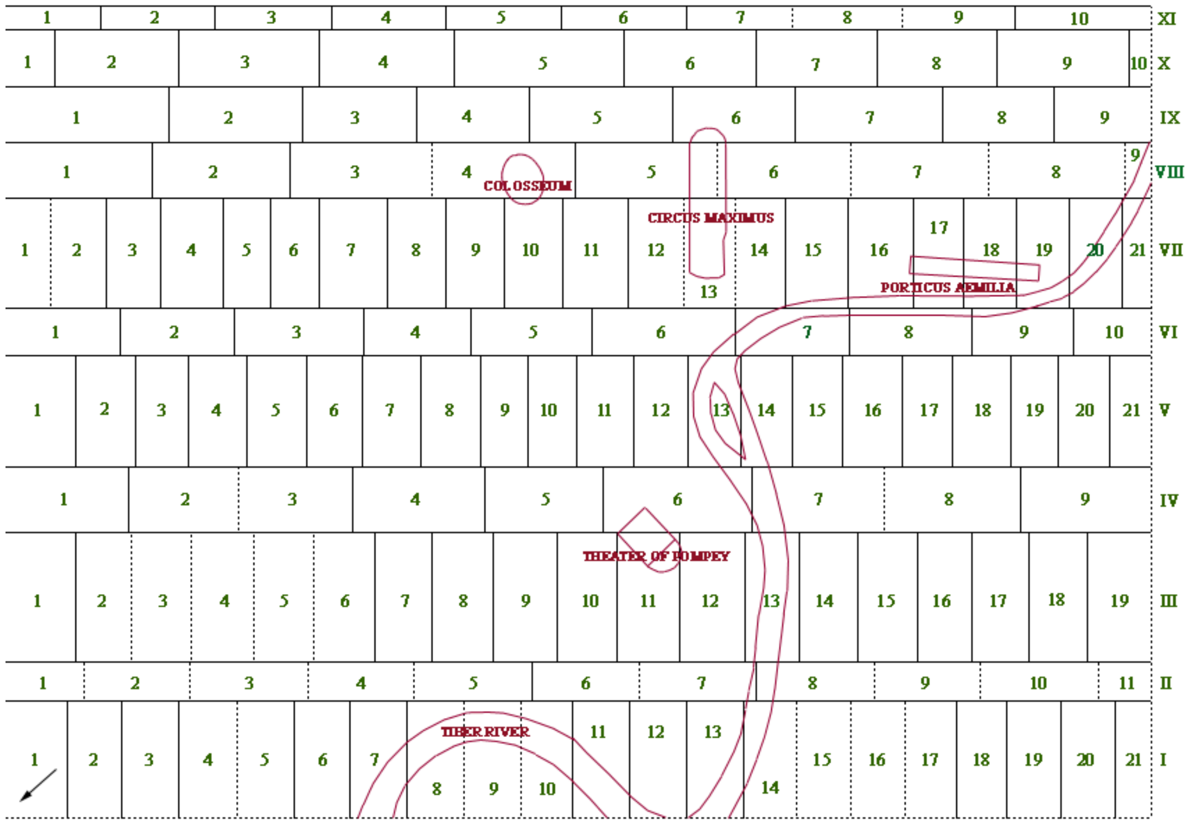
Digital Michelangelo



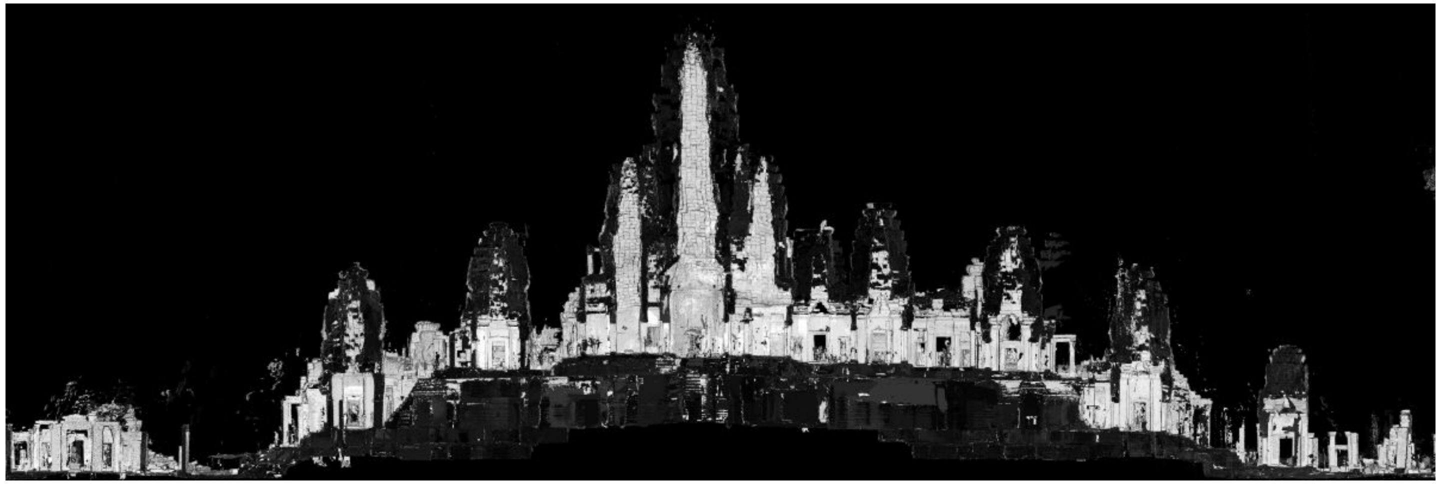
- http://graphics.stanford.edu/projects/mich/

Map of Rome

http://graphics.stanford.edu/projects/forma-urbis/database.html



Ikeuchi Lab
Bayon Project



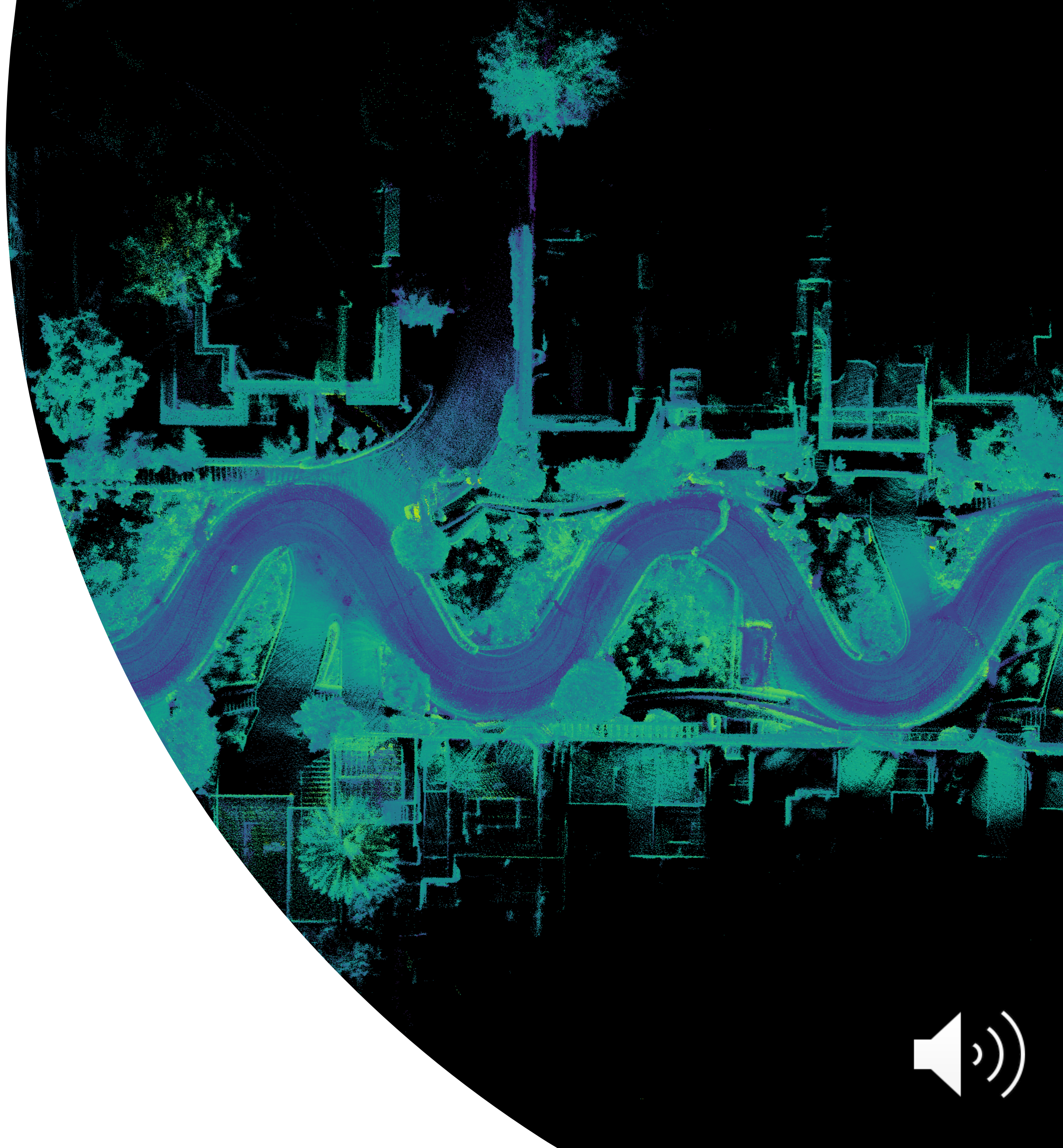
- http://www.cvl.iis.u-tokyo.ac.jp/research/bayon/

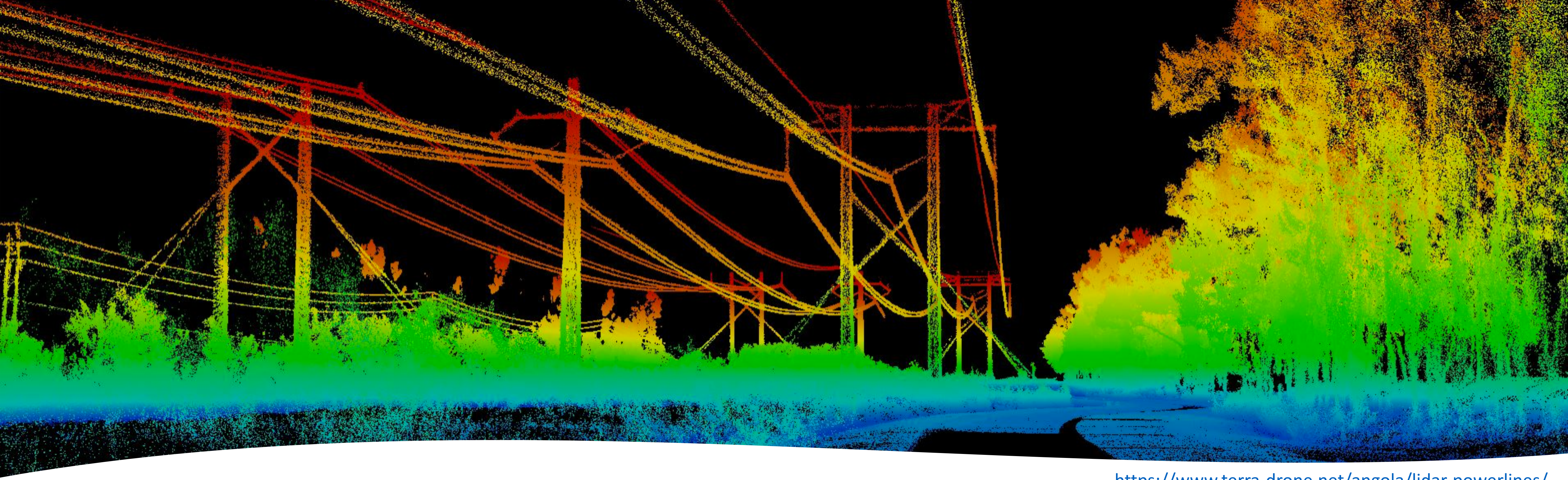
Aligning 3D Data

- How to find corresponding points?
- How to calculate a transform between two point clouds?

Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods
- Hypothesize and test
  - Generalized Hough transform
  - RANSAC
- Iterative Closest Points (ICP)





https://www.terra-drone.net/angola/lidar-powerlines/

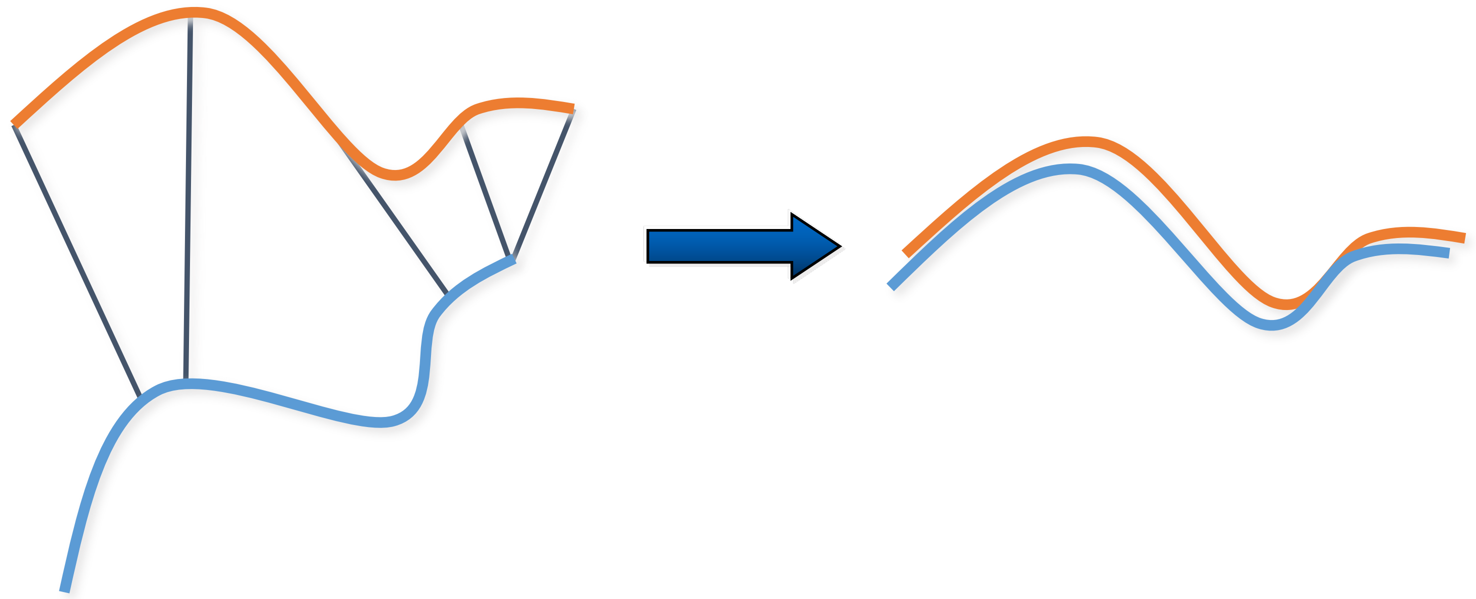
Iterative Closest Points (ICP) Algorithm

Goal: estimate transform between two dense sets of points

1. **Initialize** transformation (e.g., compute difference in means and scale)
2. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
3. **Estimate** transformation parameters using least squares
4. **Transform** the points in {Set 1} using estimated parameters
5. **Repeat** steps 2-4 until change is very small

Aligning 3D Data

Assume closest points correspond to each other, compute the best transform…

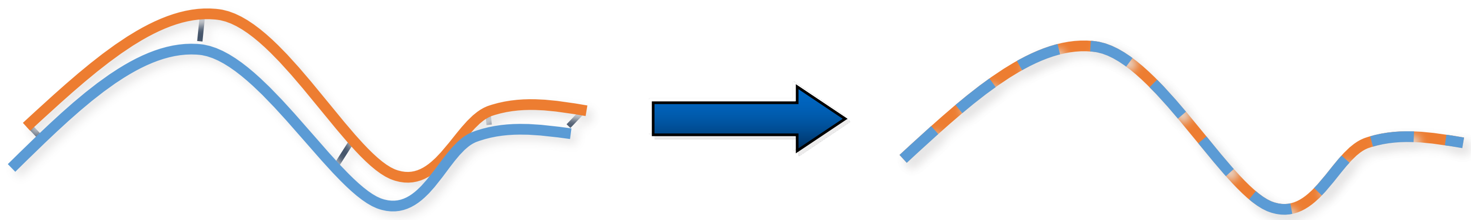


Aligning 3D Data

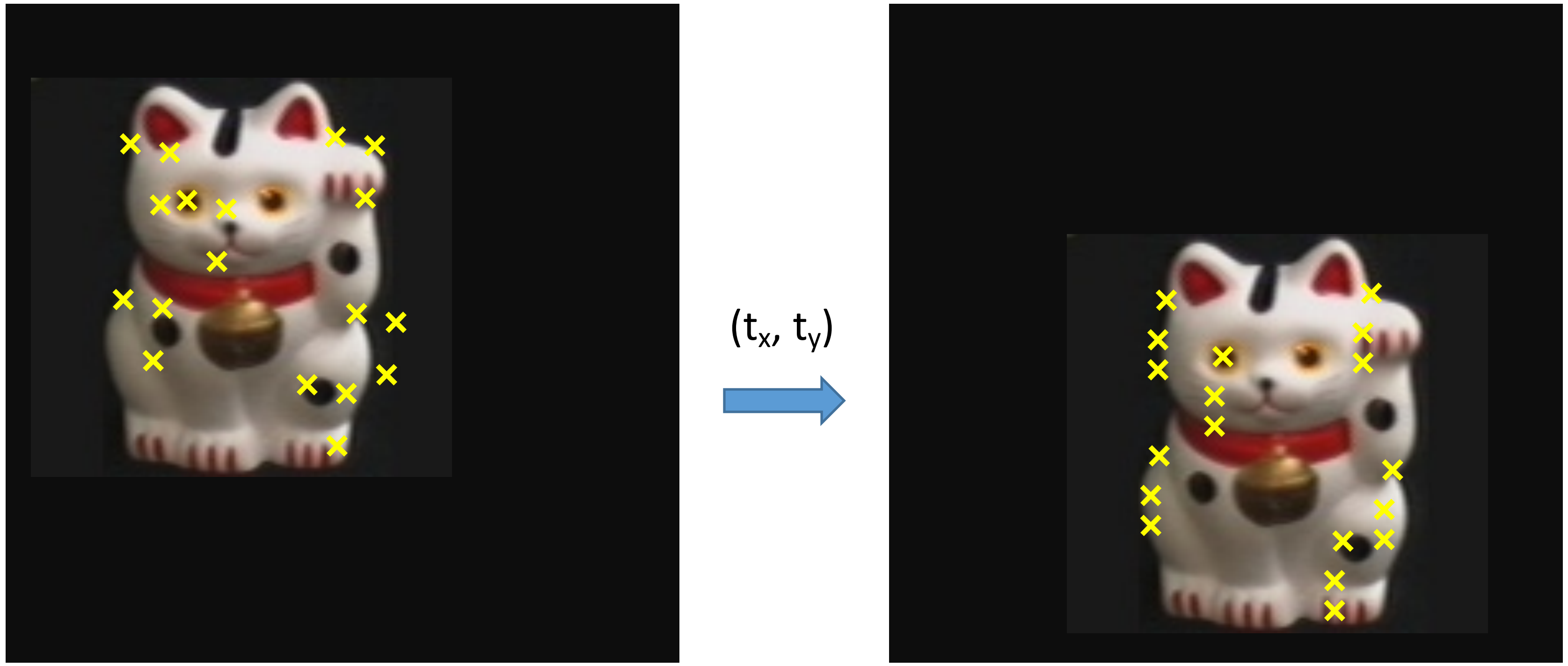
… and iterate to find alignment

Iterated Closest Points (ICP) [Besl & McKay 92]

Converges if starting position “close enough“



Example: solving for translation



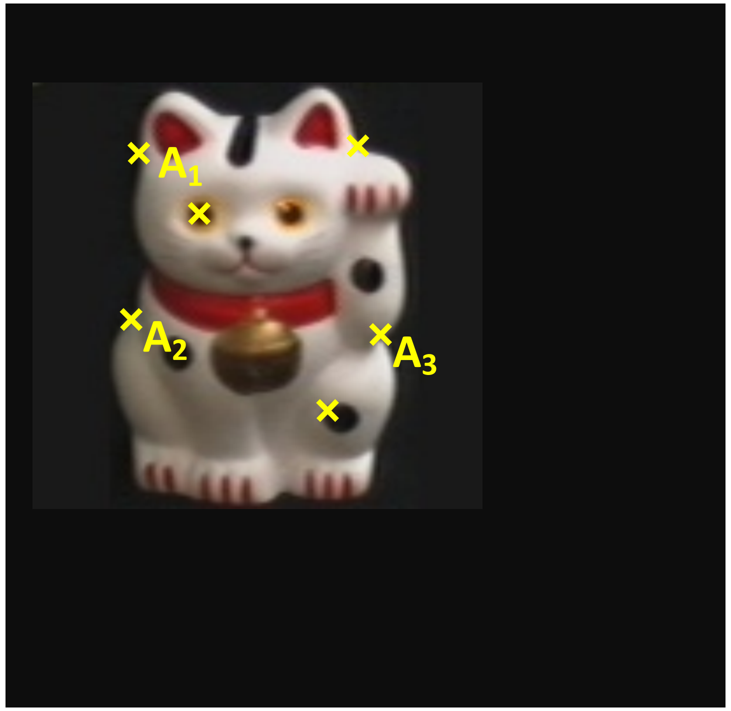
Problem: no initial guesses for correspondence

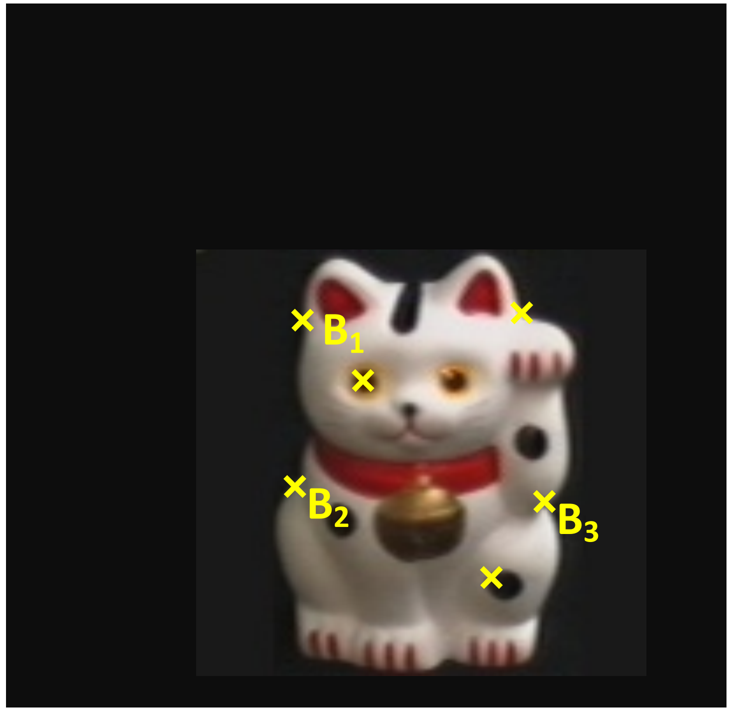
ICP solution

1. Find nearest neighbors for each point
2. Compute transform using matches
3. Move points using transform
4. Repeat steps 1-3 until convergence

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation

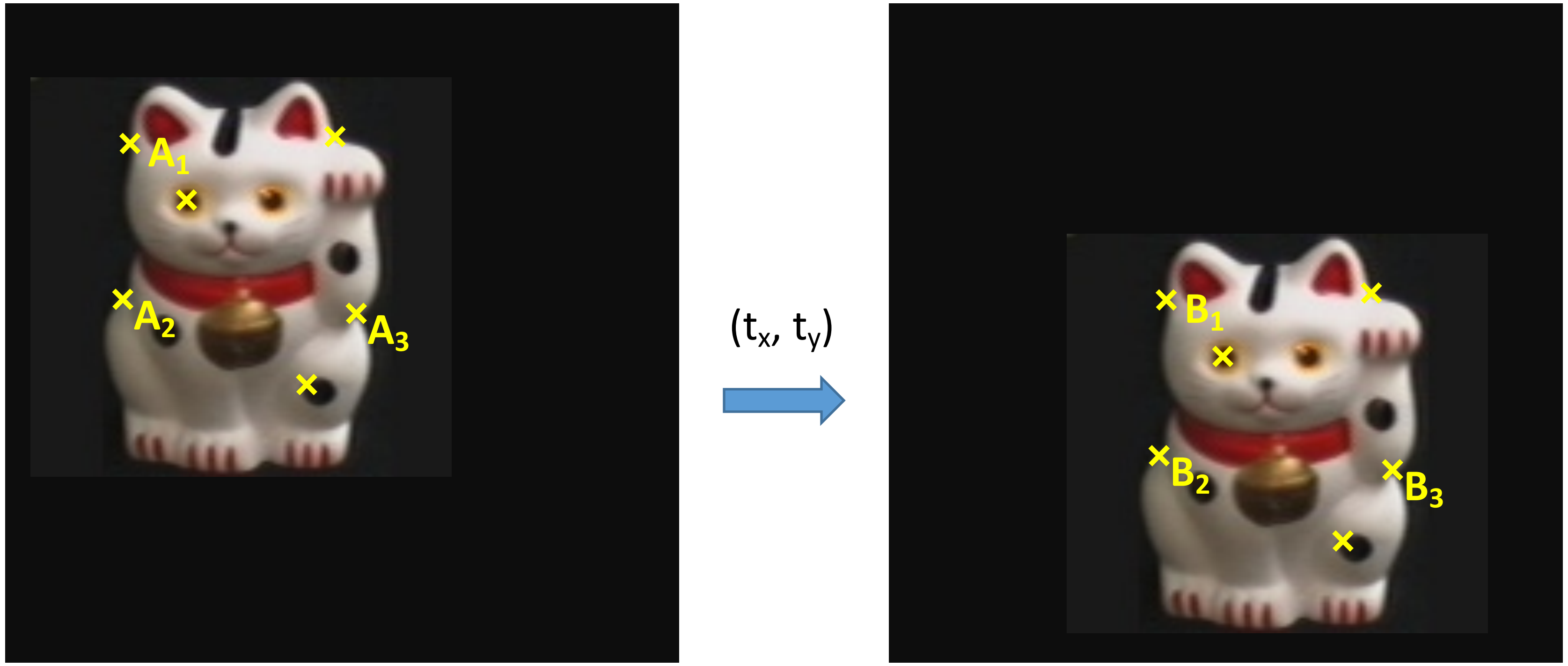




Given matched points in {A} and {B}, estimate the translation of the object

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



Least squares solution

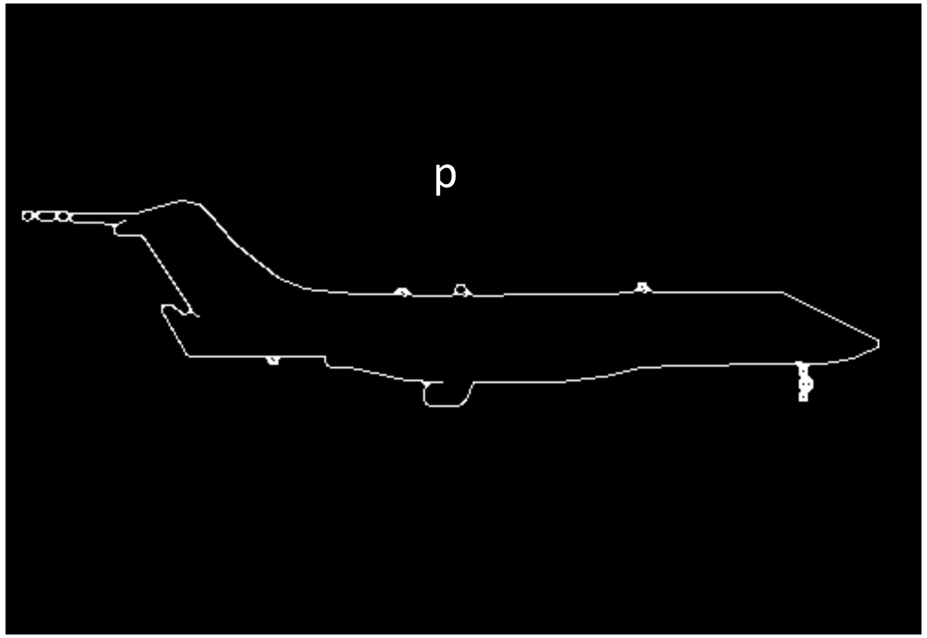
1. Write down objective function
2. Derived solution
    a) Compute derivative
    b) Compute solution
3. Computational solution
    a) Write in form Ht=b
    b) Solve using pseudo-inverse $t^* = H^+ b$

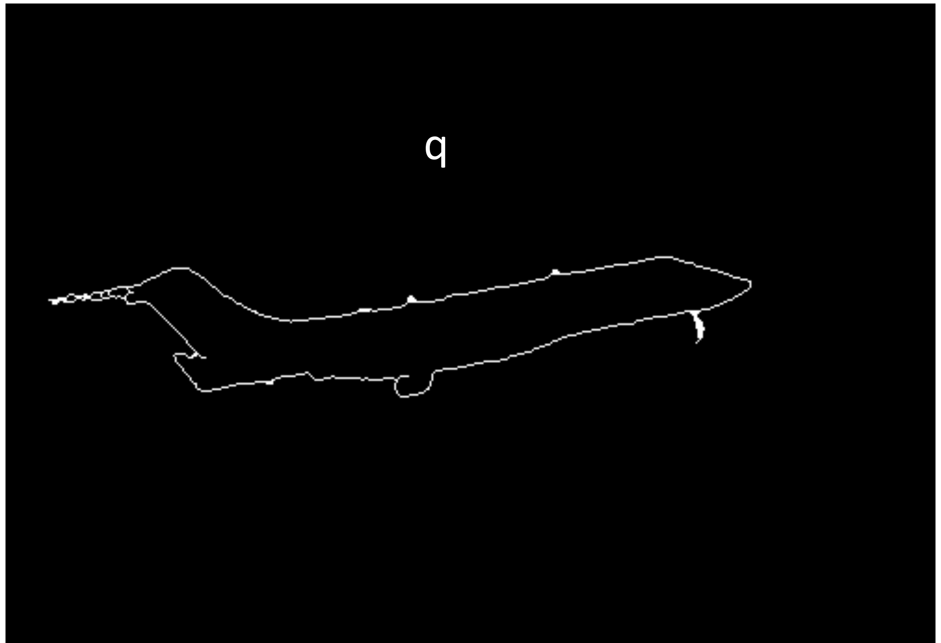
$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

(we can't use inverse $t=H^{-1}b$ as H is not square -> pseudo-inverse)

Example: aligning boundaries in images

1. Extract edge pixels $p_1..p_n$ and $q_1..q_m$
2. Compute initial transformation (e.g., compute translation and scaling by center of mass, variance within each image)
3. Get nearest neighbors: for each point p_i find corresponding $\text{match(i)} = \underset{j}{\text{argmin}}\, dist(pi, qj)$
4. Compute transformation ***T*** based on matches
5. Warp points ***p*** according to ***T***
6. Repeat 3-5 until convergence





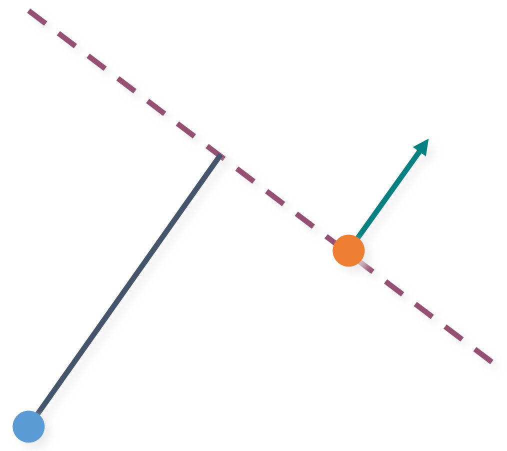
ICP Variants

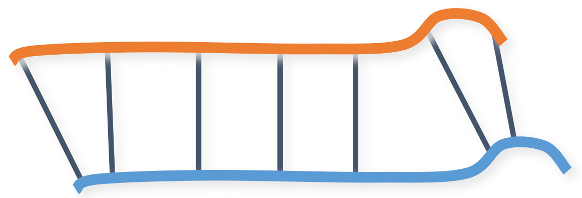
- Classic ICP algorithm not real-time
- To improve speed: examine stages of ICP and evaluate proposed variants
- [Rusinkiewicz & Levoy, 3DIM 2001]

1. Selecting source points (from one or both meshes)
2. Matching to points in the other mesh
3. Weighting the correspondences
4. Rejecting certain (outlier) point pairs
5. Assigning an error metric to the current transform
6. Minimizing the error metric

ICP Variant – Point-to-Plane Error Metric

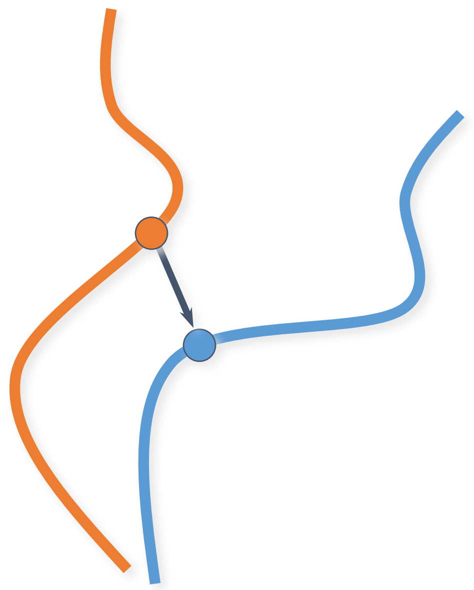
- Using point-to-plane distance instead of point-to-point lets flat regions slide along each other more easily [Chen & Medioni 91]





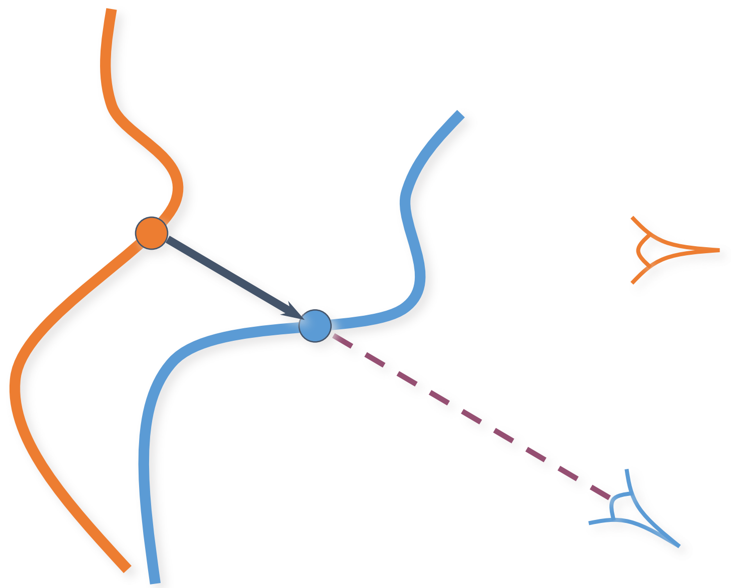
Finding Corresponding Points

- Finding closest point is most expensive stage of ICP
  - Brute force search – O(n)
  - Spatial data structure (e.g., k-d tree) – O(log n)
  - Voxel grid – O(1), but large constant, slow preprocessing



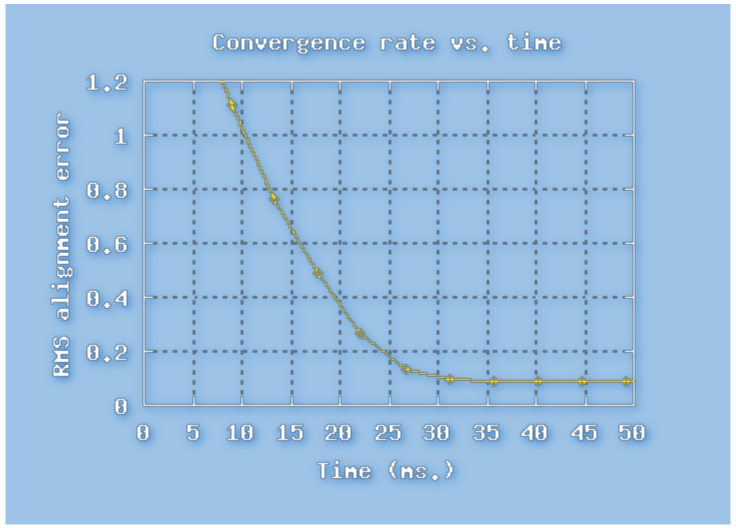
Finding Corresponding Points

- For range images, simply project point [Blais 95]
  - Constant-time, fast
  - Does not require precomputing a spatial data structure



High-Speed ICP Algorithm

- ICP algorithm with projection-based correspondences, point-to-plane matching can align meshes in a few tens of ms.
  (cf. over 1 sec. with closest-point)



[Rusinkiewicz & Levoy, 3DIM 2001]

Summary

1. Applications are plentiful
2. Basic ICP Algorithm is simple (but slow)
3. ICP Variants can speed up