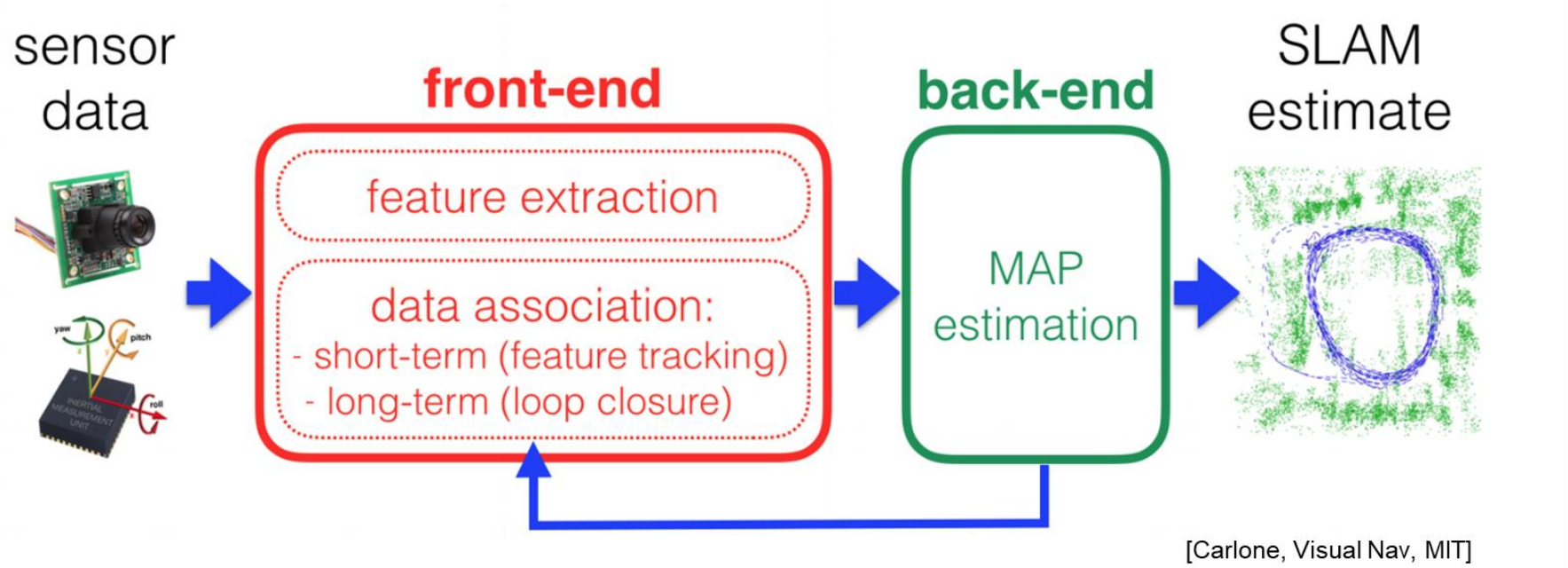
Robotic Mapping & Localization

Kaveh Fathian
Assistant Professor
Computer Science Department
Colorado School of Mines

Lec11: Lidar Odometry & ICP

*Courtesy of Maani Ghaffari, Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Kai Arras, Paolo Cignoni

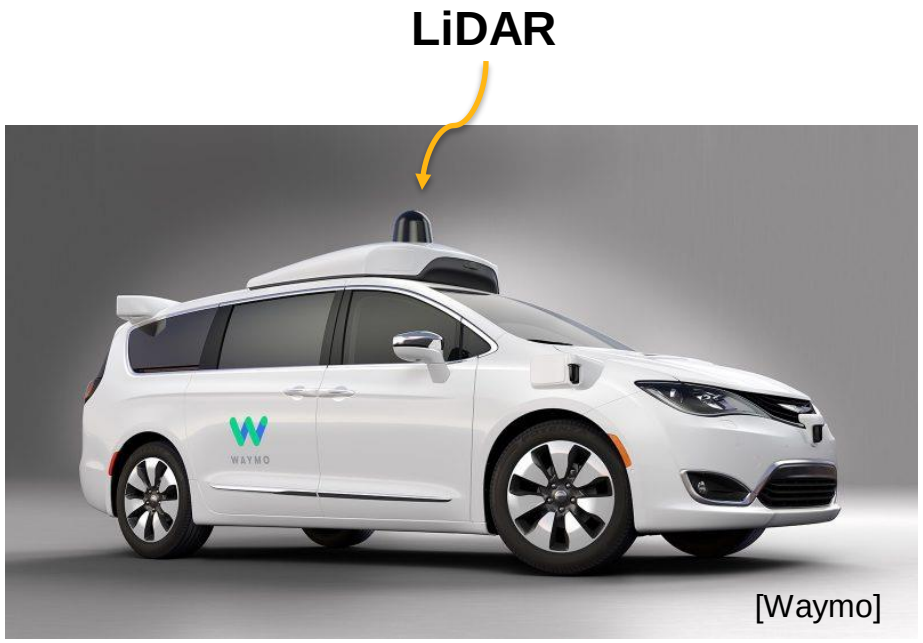
Review

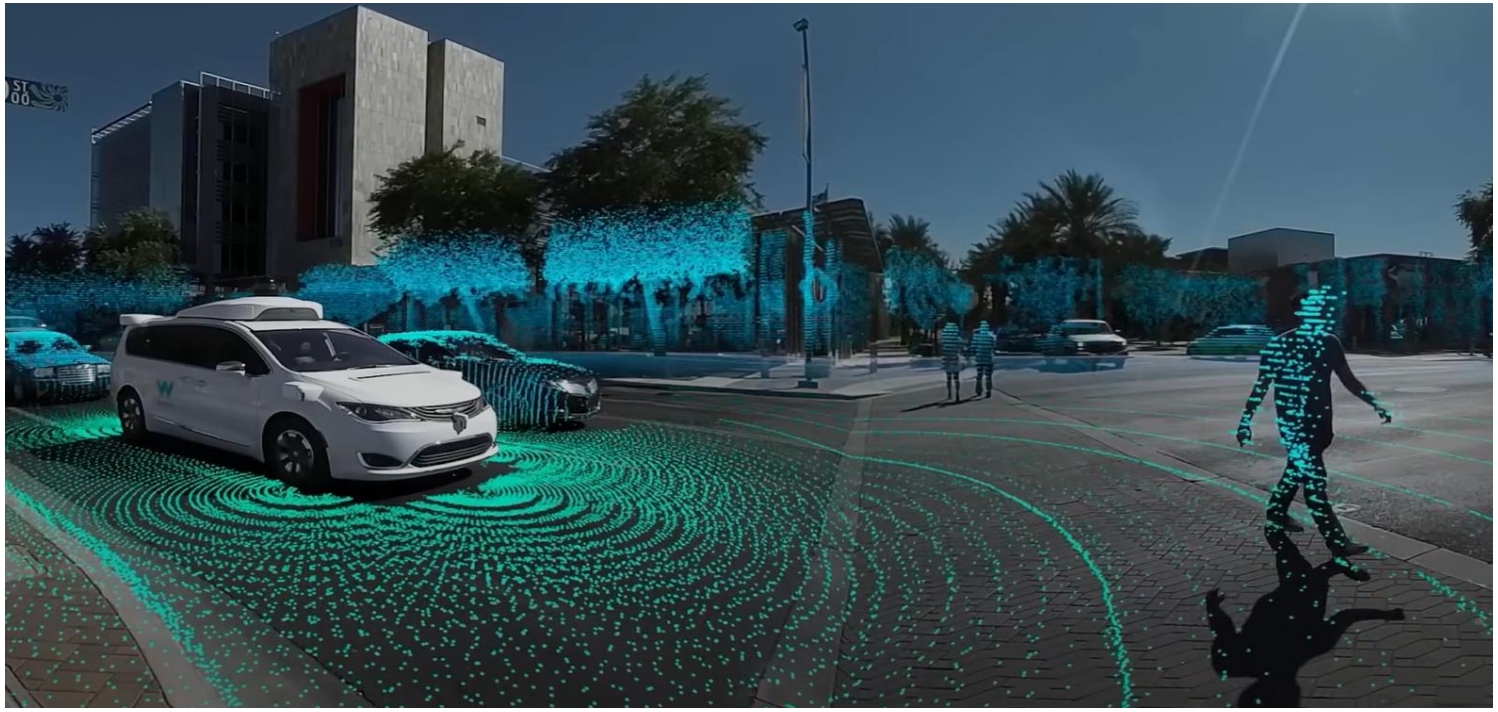


[Carlone, Visual Nav, MIT]

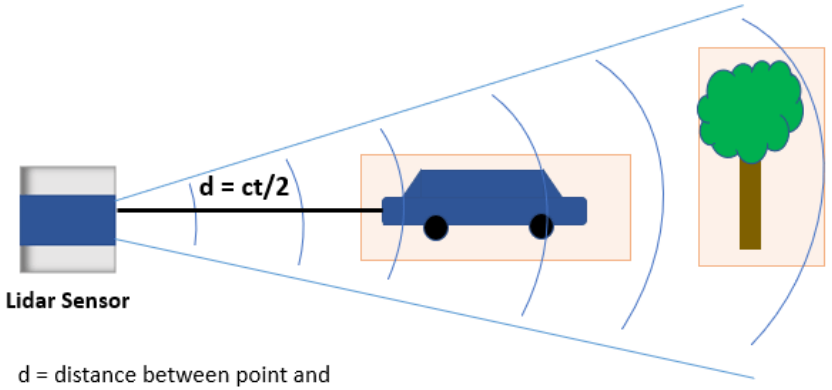
- A SLAM pipeline consists of
  - Sensors
  - Frontend algorithms
  - Backend algorithms

Review - LiDAR





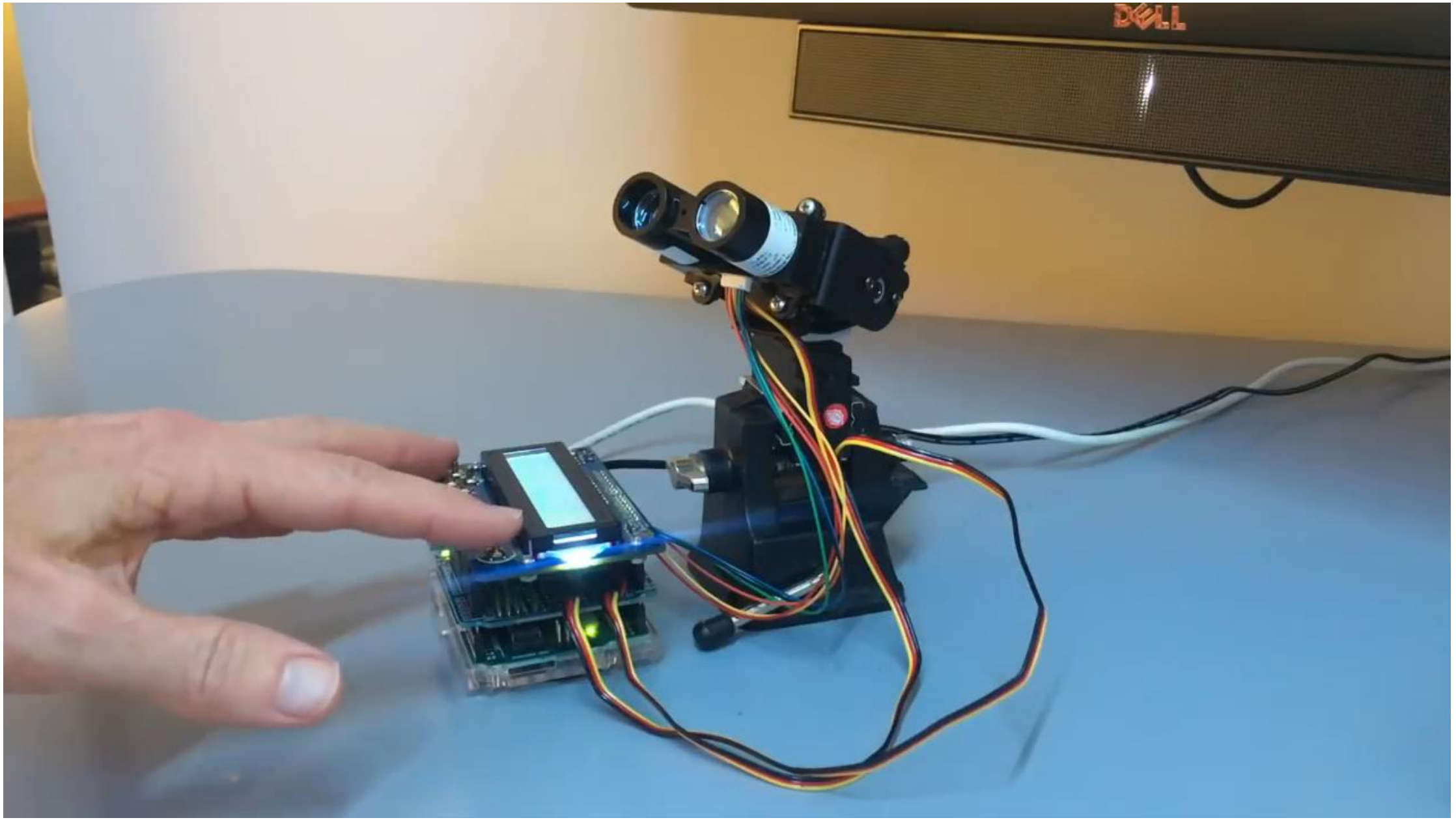
- Light Detection and Ranging (**LiDAR**) is an exteroceptive sensor
- LiDAR calculates how long it takes for beams of light to hit an object & reflect back to the sensor ➔ distance to the object



d = distance between point and the sensor
c = speed of light
t = time of flight

Review - LiDAR

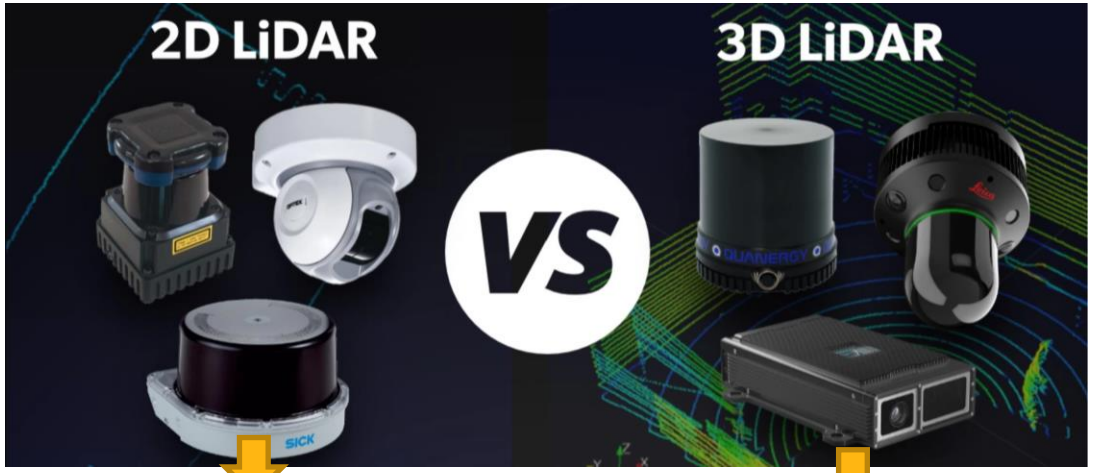
https://www.youtube.com/watch?v=xkut3yRL61U

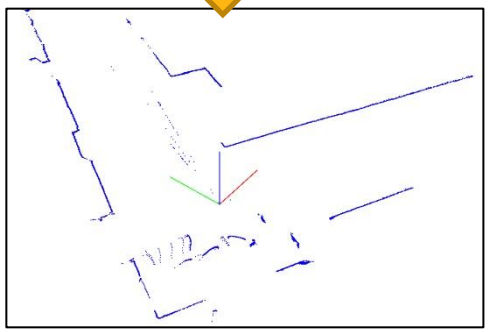


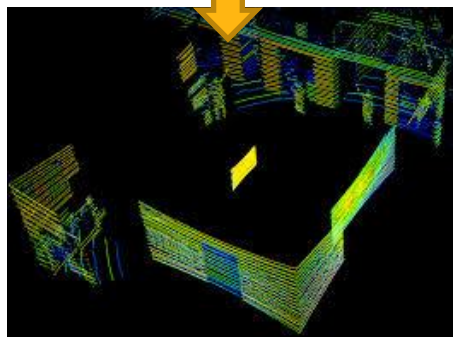
Review - LiDAR

- Common LiDAR sensors provide 2D or 3D scans

- Lidars can be rotating vs. solid state
- In a rotating/spinning LiDAR, a laser beam is steered at 360° by a mechanical system
- In a solid-state LiDAR, a MEMS-based mirror vibrates to scan the environment







Rotating

Solid-state

Review - LiDAR

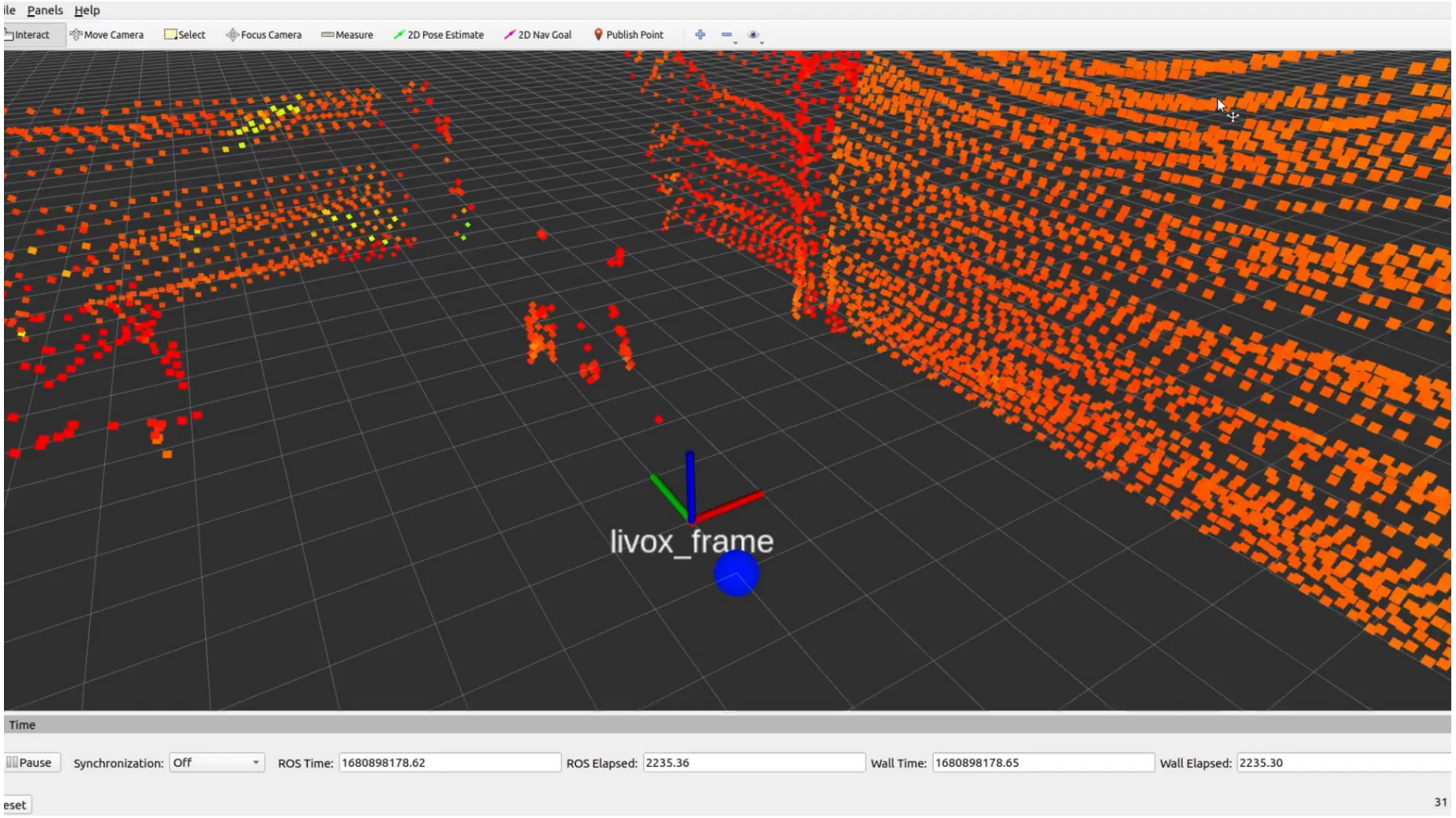
- **Advantages**:
  - Works day/night
  - Very high accuracy
  - Provides 3D/depth information; detailed 3D maps
  - Operates effectively over long distances

Ford tests Lidar-equipped car in pitch darkness [techcrunch.com]

- **Challenges**:
  - Poor performance in (heavy) rain, snow, fog, dusty conditions (due to spurious beam returns)
  - Surfaces with low reflectivity (e.g., black/dark surfaces) may be challenging to detect
  - Traditional (spinning) LiDAR sensors are heavy; high power consumption
  - More expensive than alternative sensors (e.g., RGBD cameras)
  - Crosstalk from other LiDAR devices

LiDAR odometry

- Lidar odometry is to estimate the motion/pose from consecutive Lidar scans



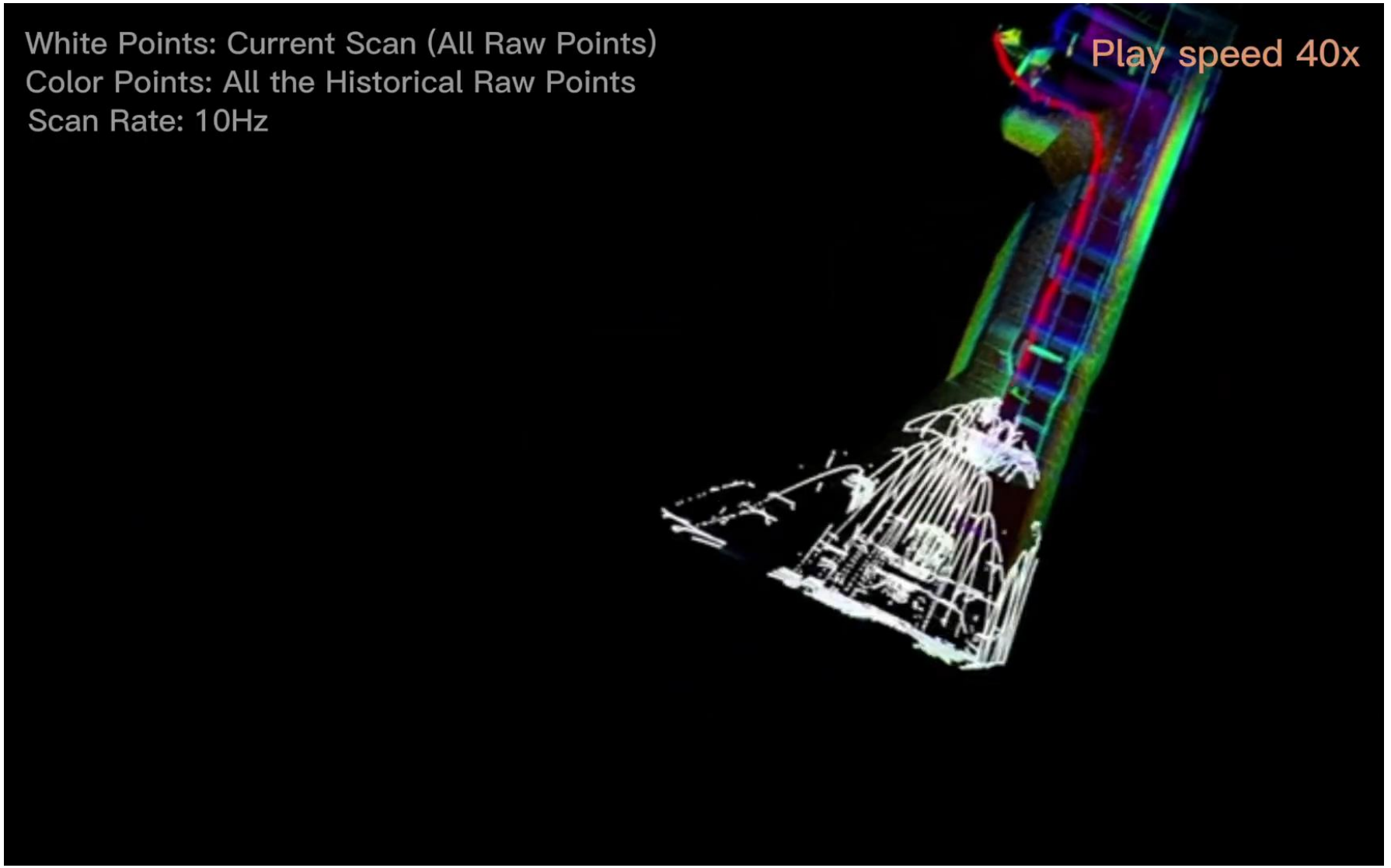
https://www.youtube.com/watch?v=UcaE_PMEbJ8

LiDAR odometry

- Lidar odometry is to estimate the motion/pose from consecutive Lidar scans

LiDAR SLAM

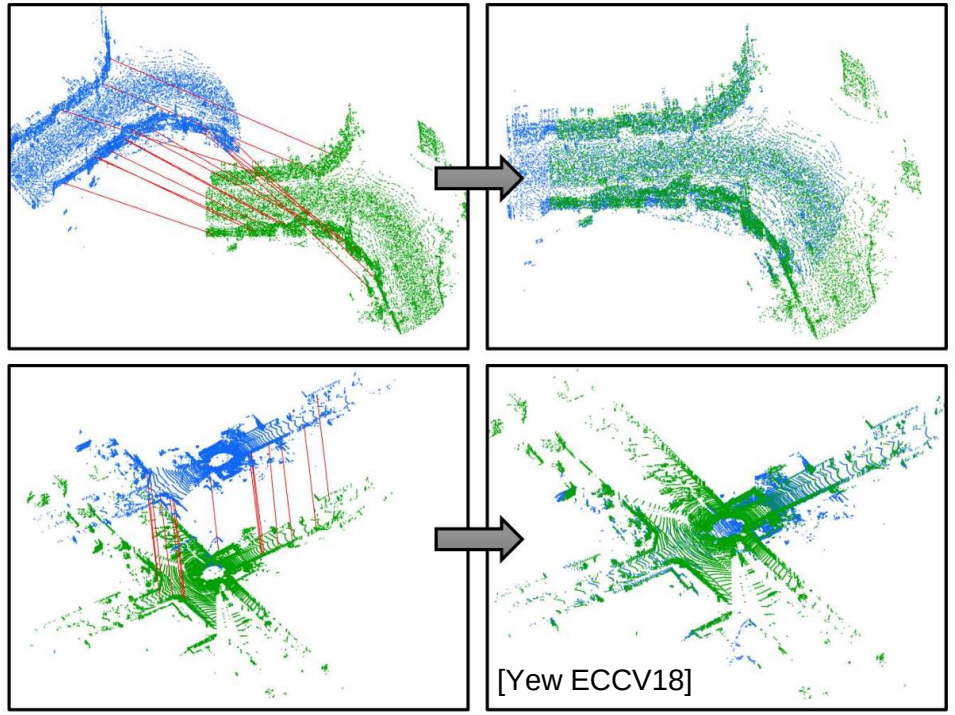
- Lidar **SLAM** includes *mapping* (& localization/loop closure) in addition to odometry

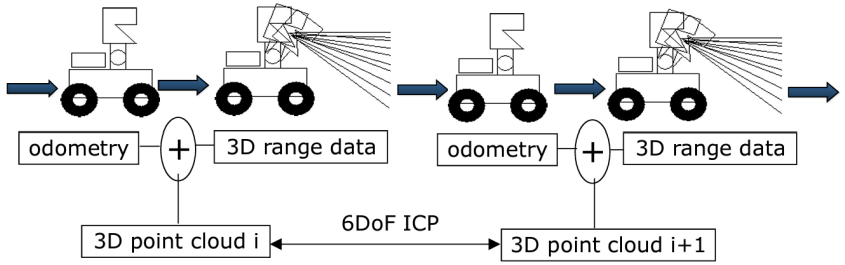


https://www.youtube.com/watch?v=Cr78ZeV-Rx4

LiDAR odometry

- Lidar odometry requires solving the **point cloud registration** problem
- Registration problem: Find the geometric relationship between sensor data and some prior (such as previous sensor data or map).
- Approaches can vary by:
  - Resolution
    - Coarse algorithms look for crude alignments, such as for place recognition.
    - Fine algorithms try to achieve better alignments, but generally need more accurate initial guesses.
  - Data
    - Direct algorithms use the raw observations.
    - Indirect (feature-based) algorithms abstract dense raw data into sparse features.
  - Local vs. global solvers.
  - Real-time vs. offline solvers.





Point cloud registration problem

- A set of 3D points: $\mathrm{X} := \{x_i\}$, $\mathrm{X} \subset \mathbb{R}^3$
- A rigid body transformation on 3D points: $T \cdot x := R\,x + p$
- $R \in \mathrm{SO}(3)$, $p \in \mathbb{R}^3$

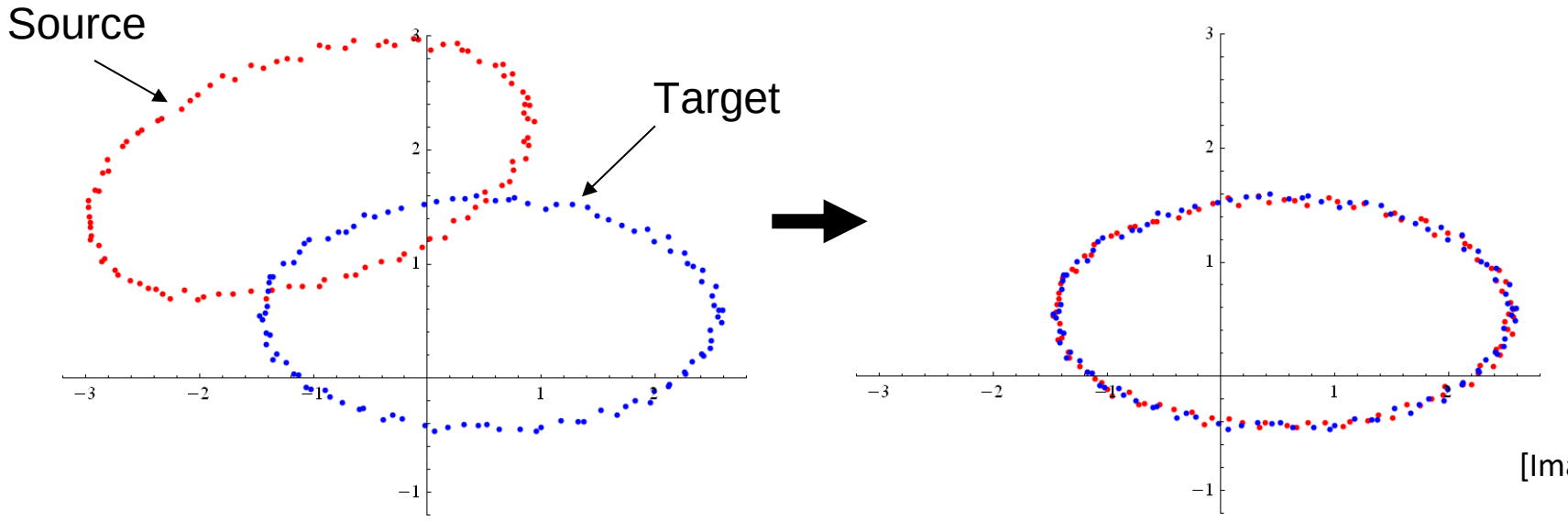
Point cloud registration problem

Definition (Target Point Cloud)

The point cloud X_t which is considered to be in a fixed reference frame is called the target point cloud, i.e., the point cloud we are trying to align to.

Definition (Source Point Cloud)

The point cloud X_s that we transform by $T \in \mathrm{SE}(3)$ to align to the target point cloud.

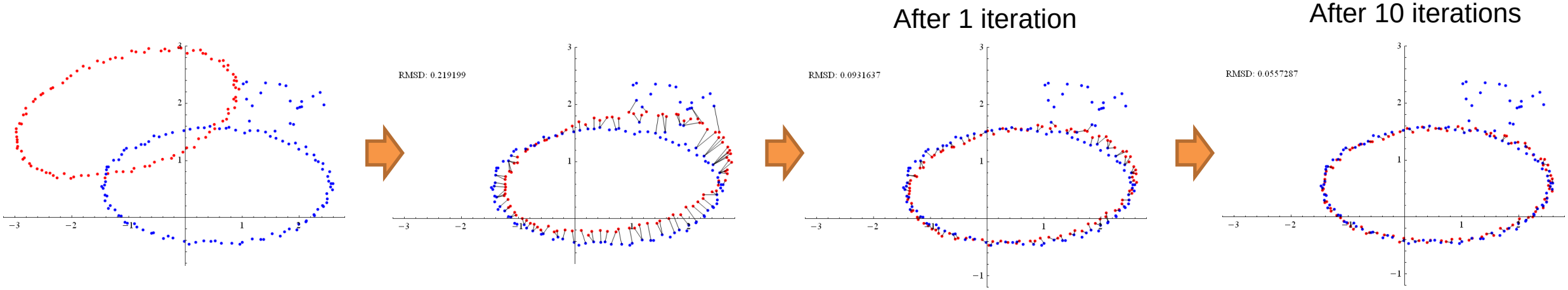


[Image courtesy: Ju]

Iterative Closest Point (ICP) algorithm - overview

- **Step 1:** Determine associations, $\mathcal{I}$, between target and source by finding the nearest neighbor.

- **Step 2:** Given those associations, minimize the residual between points.



ICP overview:

- **Step 1:**

$$i_k = \arg\min_k \|x_k^t - T \cdot x_i^s\|$$

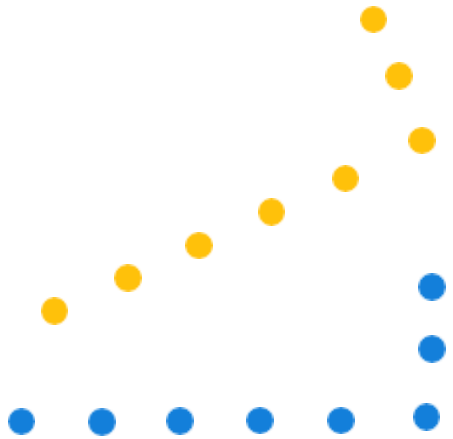
$$\mathcal{I} := \{i_k\}$$

- **Step 2:**

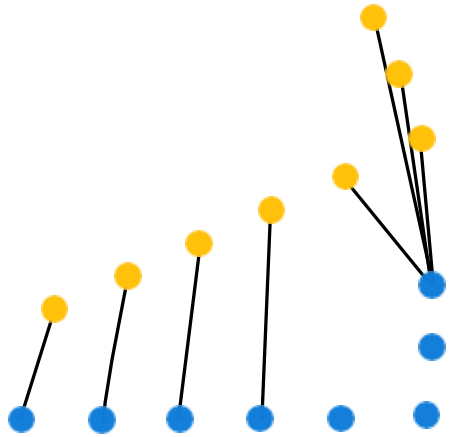
$$r_k(T) := x_k^t - T \cdot x_k^s$$

$$T^{\mathsf{OPT}} = \arg\min_{T \in \mathrm{SE}(3)} \sum_{k \in \mathcal{I}} \|r_k(T)\|^2$$

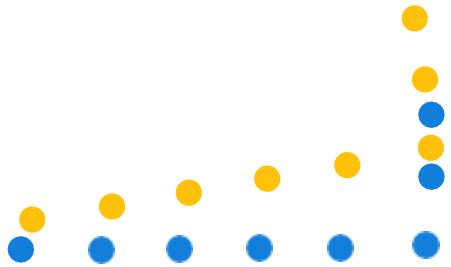
ICP



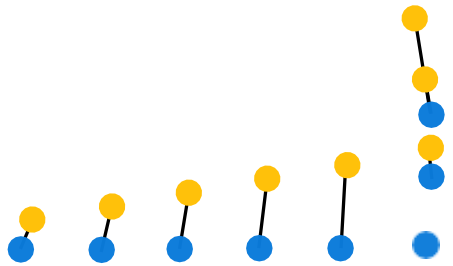
ICP



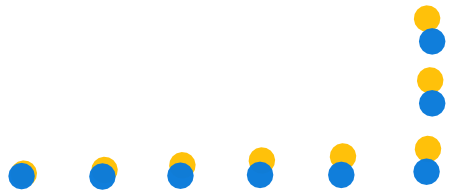
ICP



ICP

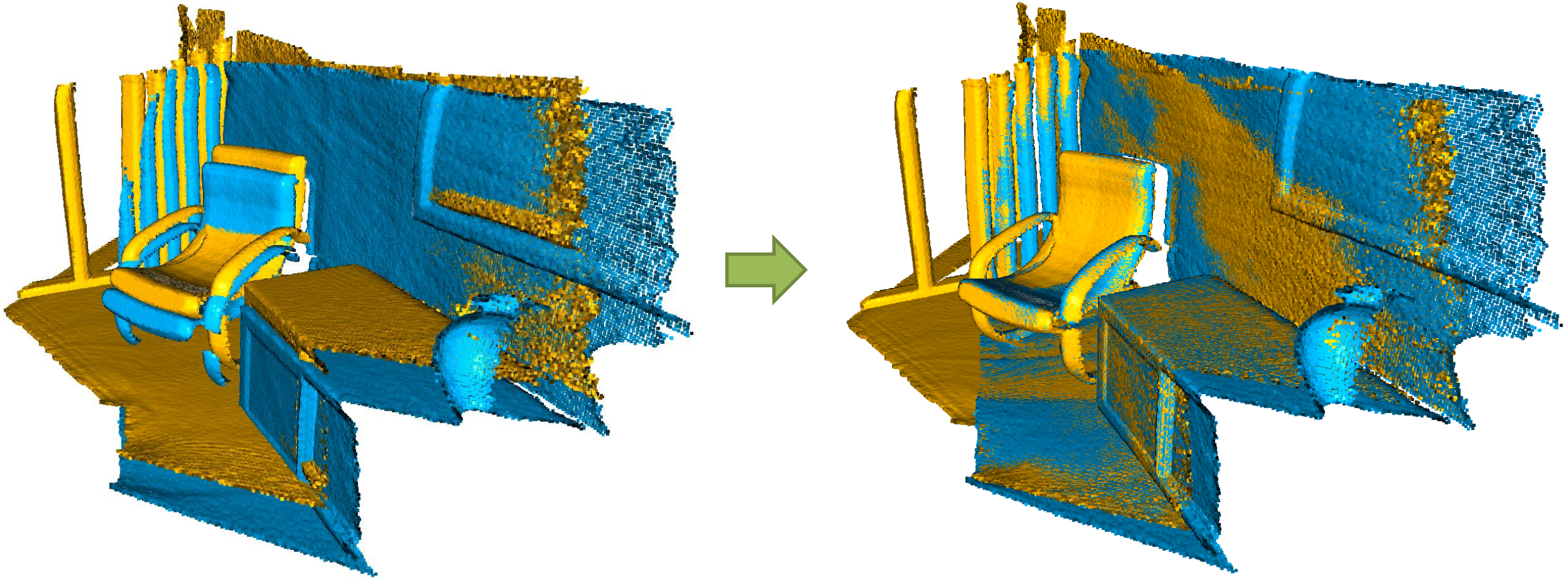


ICP



Point cloud registration example

- Registered scene solved using ICP
  (https://www.open3d.org/docs/release/tutorial/pipelines/icp_registration.html)



Least-squares fitting of two 3D point sets (Arun's Method)

- Given: two **corresponding** point sets:

$$X = \{x_1, \ldots, x_n\}$$

$$P = \{p_1, \ldots, p_n\}$$

- Wanted: translation t and rotation R that minimizes the sum of the squared error:

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} ||x_i - Rp_i - t||^2$$

Where x_i and p_i are corresponding points.

Center of Mass

$$\mu_x = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i \quad \text{and} \quad \mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

are the centers of mass of the two point sets.

Idea:

- Subtract the corresponding center of mass from every point in the two point sets before calculating the transformation.
- The resulting point sets are:

$$X' = \{x_i - \mu_x\} = \{x'_i\}$$

and

$$P' = \{p_i - \mu_p\} = \{p'_i\}$$

SVD

Let $W = \sum_{i=1}^{N_p} x'_i p'^T_i$

denote the singular value decomposition (SVD) of W by:

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

where $U, V \in \mathbb{R}^{3\times3}$ are unitary, and

$\sigma_1 \geq \sigma_2 \geq \sigma_3$ are the singular values of W.

SVD

Theorem (without proof):

If rank(W) = 3, the optimal solution of $E(R,t)$ is unique and is given by:

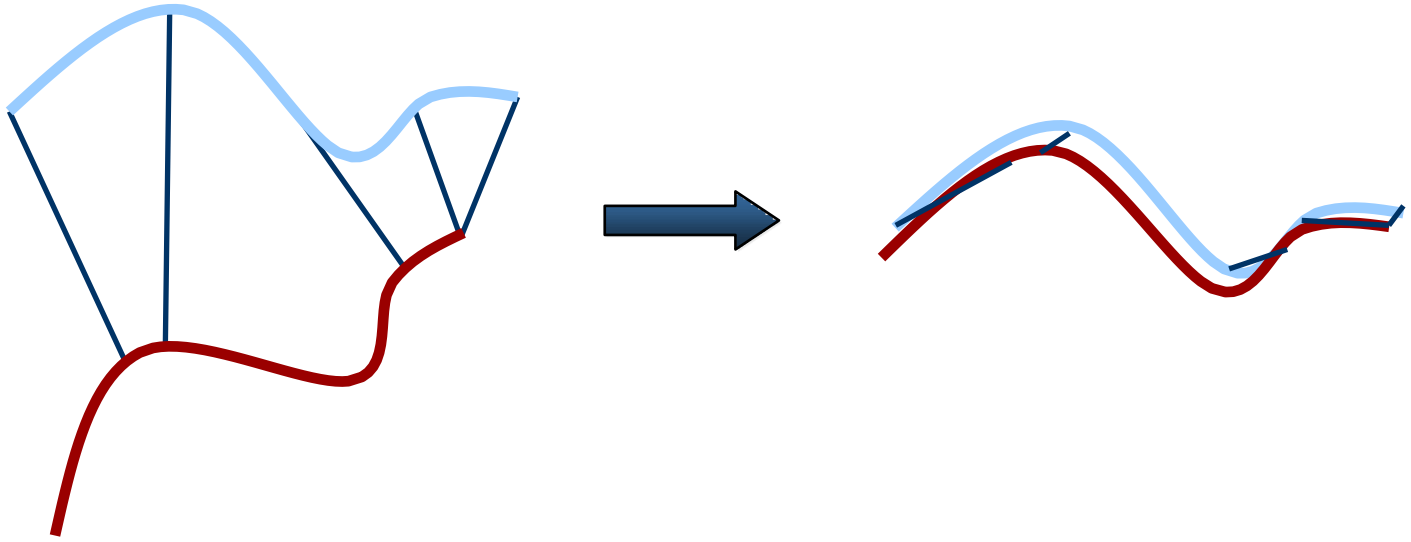
$$R = UV^T$$
$$t = \mu_x - R\mu_p$$

The minimal value of error function at (R,t) is:

$$E(R,t) = \sum_{i=1}^{N_p} (||x_i'||^2 + ||y_i'||^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

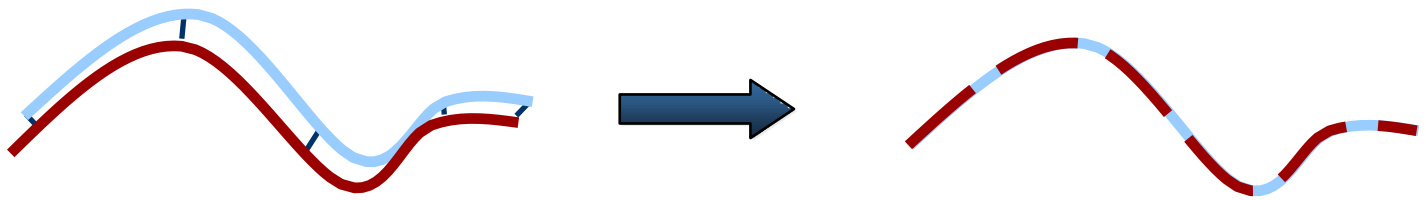
ICP with Unknown Data Association

- If correct correspondences are not known, it is generally impossible to determine the optimal relative rotation/translation in one step



ICP-Algorithm

- Idea: iterate to find alignment
- Iterated Closest Points (ICP) [Besl & McKay 92]
- **Converges** if starting positions are "**close enough**"



ICP-Variants

- Many variants on the following stages of ICP:

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Performance of Variants

- Various aspects of performance:
  - Speed
  - Stability (local minima)
  - Tolerance wrt. noise and/or outliers
  - Basin of convergence (maximum initial misalignment)

ICP Variants

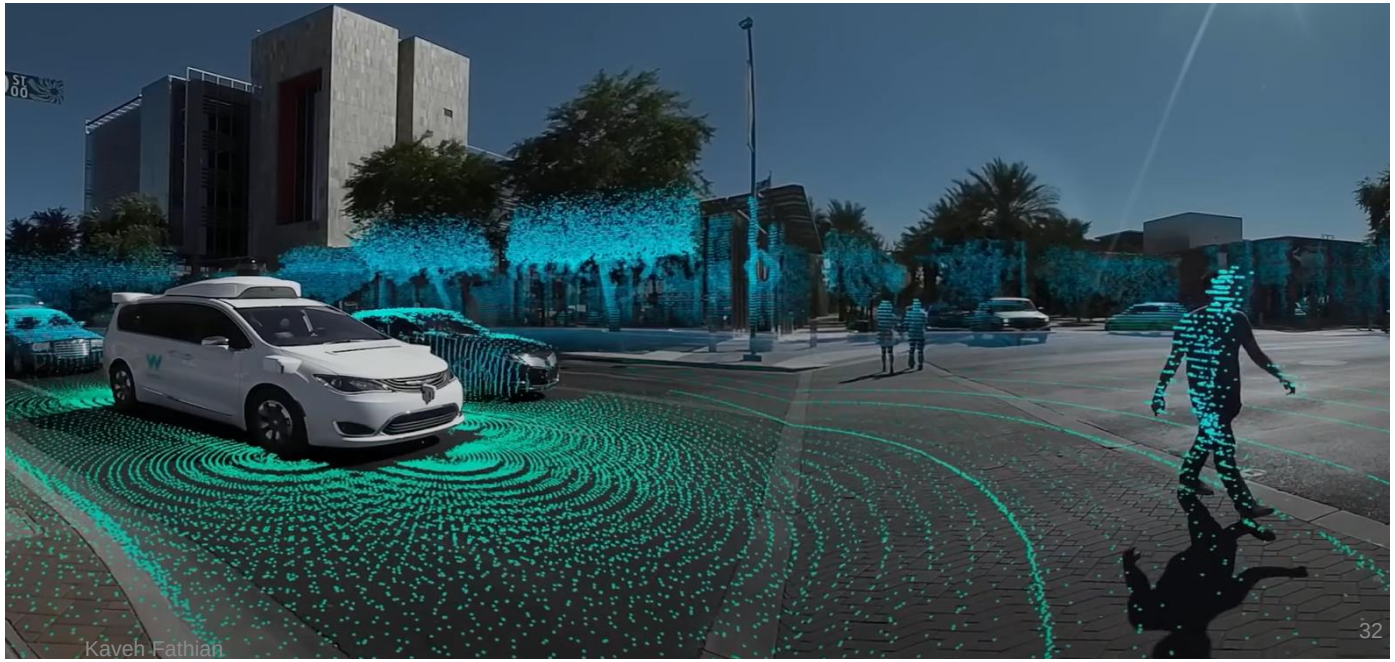
1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Selecting Source Points

- Use all points
- Uniform sub-sampling
- Random sampling
- Feature based Sampling
- Normal-space sampling
  - Ensure that samples have normals distributed as uniformly as possible

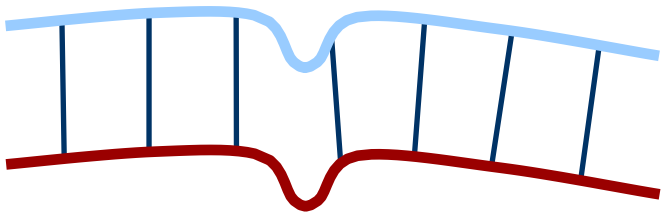
Feature Extraction

- LiDAR point cloud scans typically contain thousands of points
- They must be downsampled to obtain real-time performance for SLAM
- Some basic downsampling techniques:
  - Voxel Grid downsampling
  - Random uniform/non-uniform downsampling
  - Octree downsampling
- Alternatively, feature points can be extracted

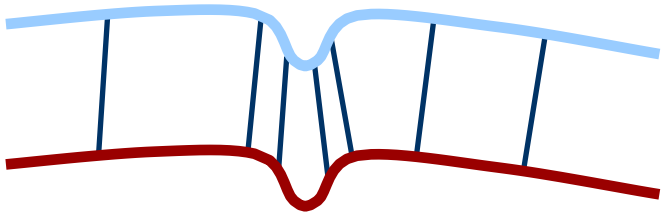


Normal-Space Sampling

- Samples the point cloud in the space of normal directions computed at every point



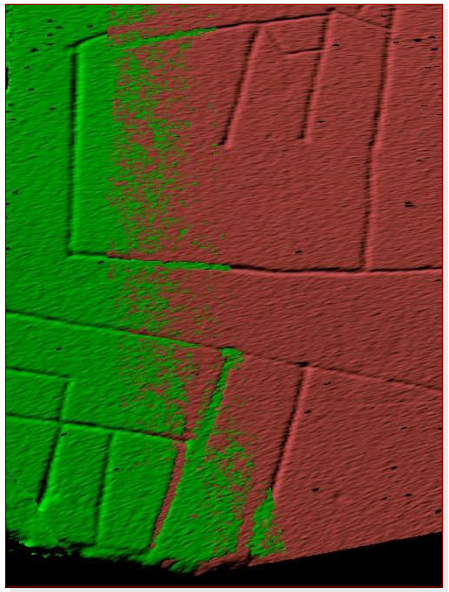
uniform sampling



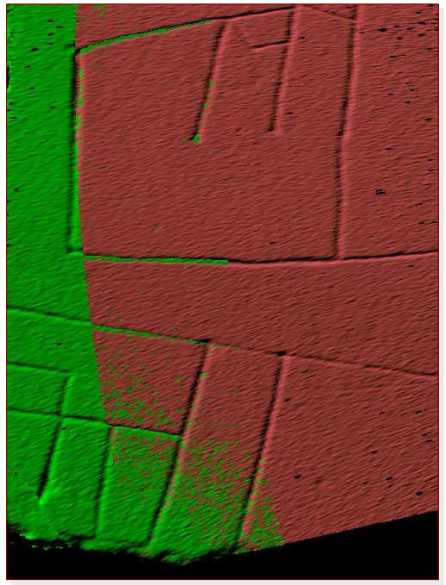
normal-space sampling

Comparison

- Normal-space sampling is better for mostly-smooth areas with sparse features [Rusinkiewicz et al.]



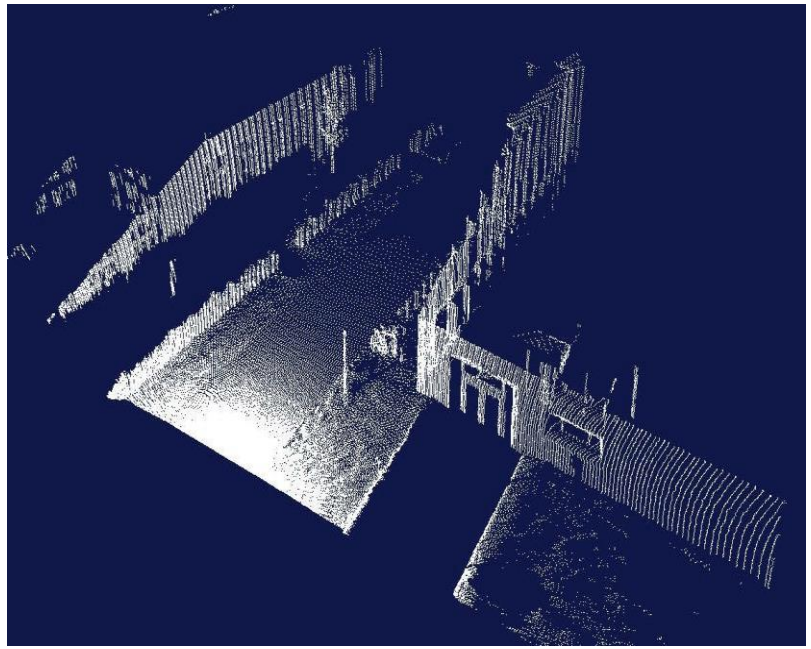
Random sampling



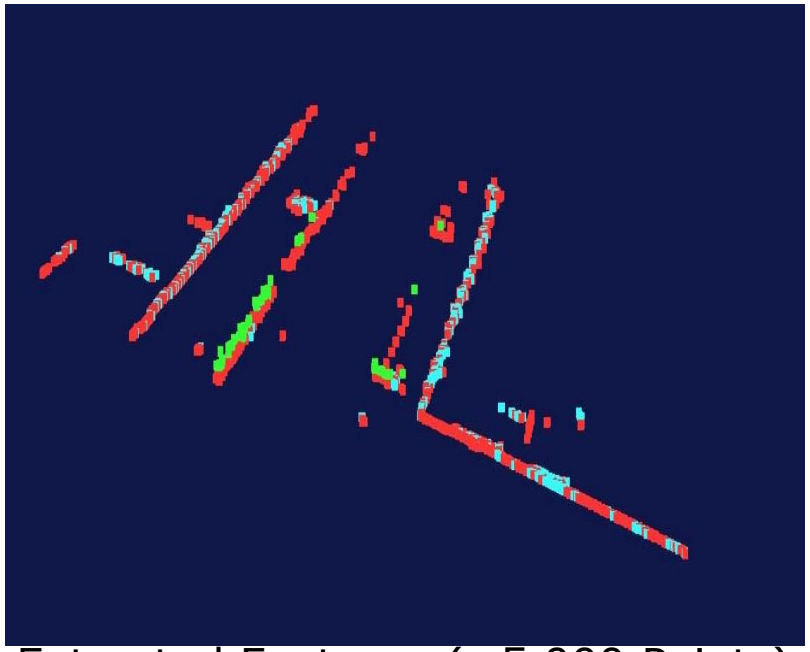
Normal-space sampling

Feature-Based Sampling

- try to find “important” points
- decrease the number of correspondences
- higher efficiency and higher accuracy
- requires preprocessing



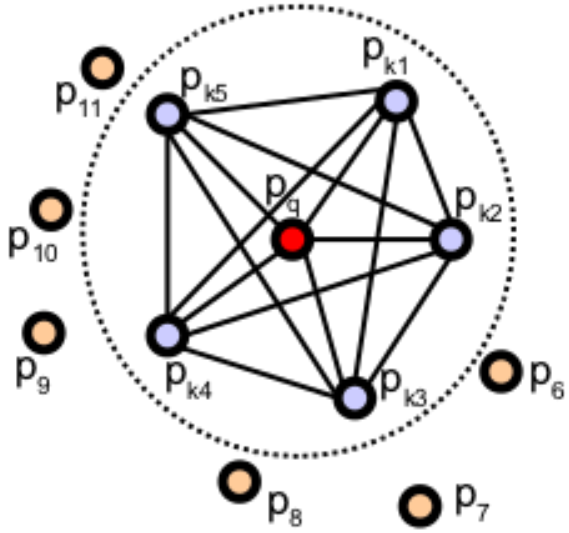
3D Scan (~200.000 Points)

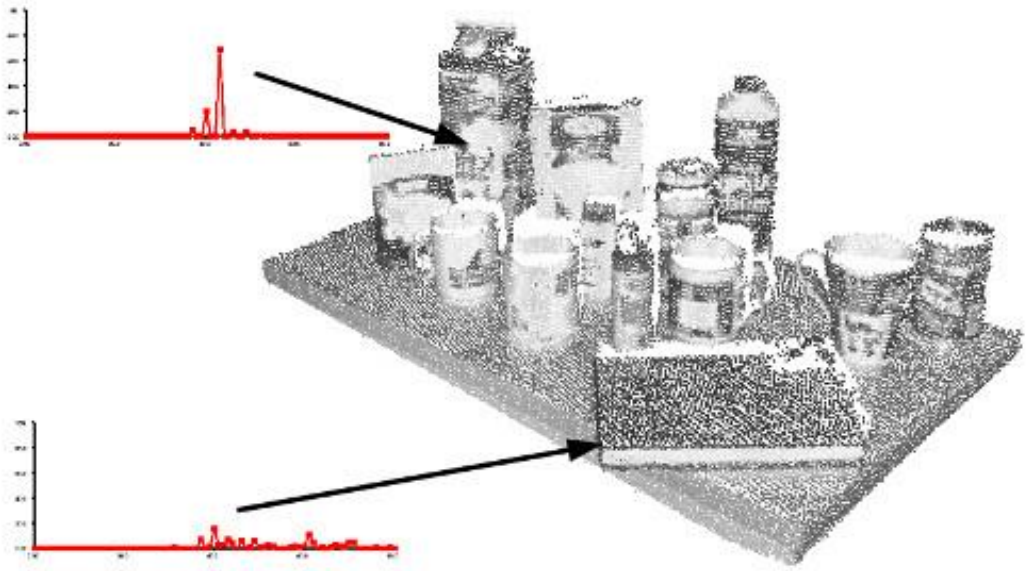


Extracted Features (~5.000 Points)

FPFH

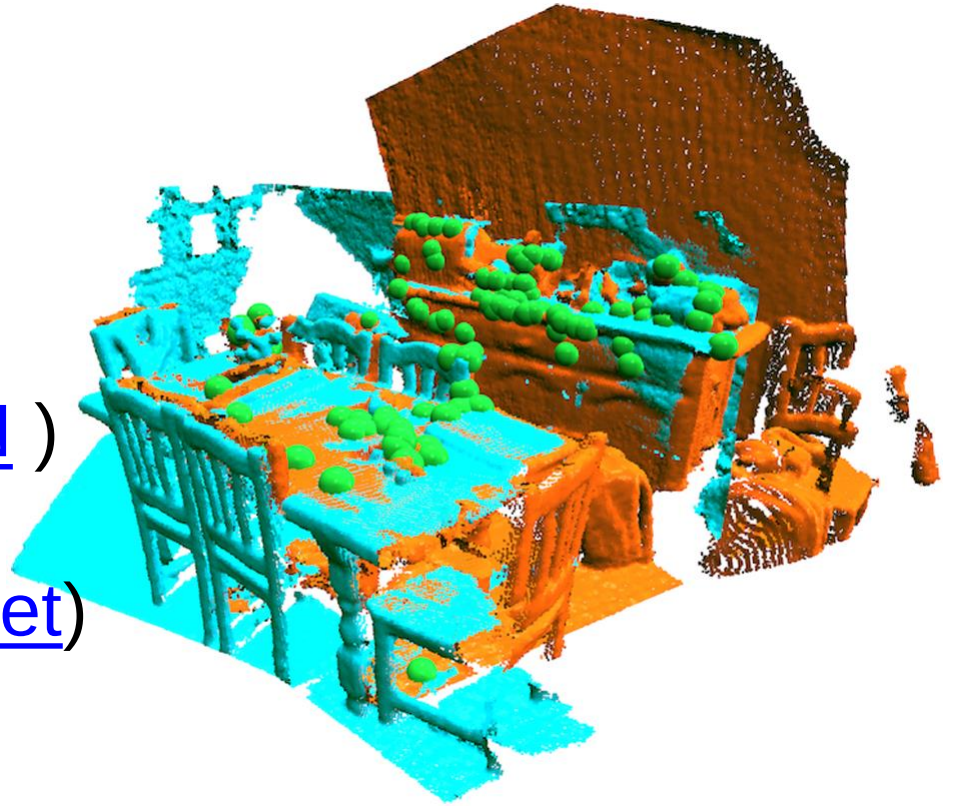
- **FPFH** (Fast Point Feature Histograms) was introduced as an extension of Point Feature Histograms (**PFH**) to be computationally more efficient
- Paper "3D is here: Point Cloud Library (PCL)"
- FPFH features are computed for ***each*** point in point cloud
- They represent local geometry of points within a specified neighborhood
- **PFH** captures the relative geometry between pairs of points as histograms based on their pairwise distances & angles between their normals





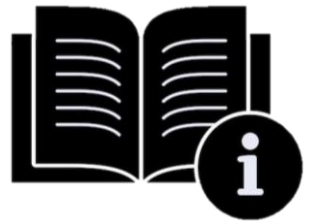
Deep-Learning-Based Features

- There are many deep-learning-based (recent) techniques for feature extraction
- Some popular DL-based methods:
  - PointNet, 2017 (https://arxiv.org/abs/1612.00593)
  - PointNet++, 2017 (https://arxiv.org/abs/1706.02413)
  - PointCNN, 2018 (https://github.com/yangyanli/PointCNN)
  - 3DSmoothNet, 2019 (https://github.com/zgojcic/3DSmoothNet)



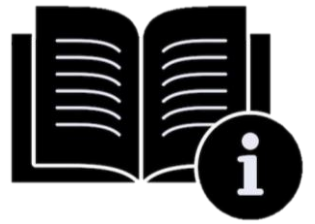
https://github.com/zgojcic/3DSmoothNet

ICP Variants



1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Selection vs. Weighting



- Could achieve same effect with weighting
- Hard to guarantee that enough samples of important features except at high sampling rates
- Weighting strategies turned out to be dependent on the data.
- Preprocessing / run-time cost tradeoff (how to find the correct weights?)

ICP Variants

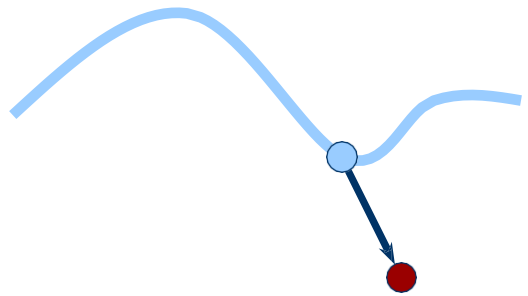
1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Data association
4. Rejecting certain (outlier) point pairs

Data Association

- Has the **greatest** effect on convergence & speed
- Data association techniques used for ICP:
  - Closest point
  - Normal shooting
  - Closest compatible point
  - Projection
  - Using kd-trees or oc-trees

Closest-Point Matching

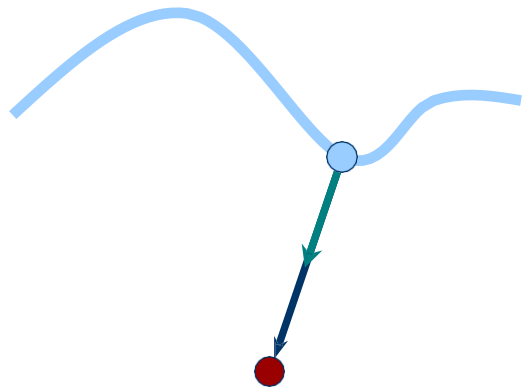
- Find closest point in other the point set



Closest-point matching generally stable, but slow and requires preprocessing

Normal Shooting

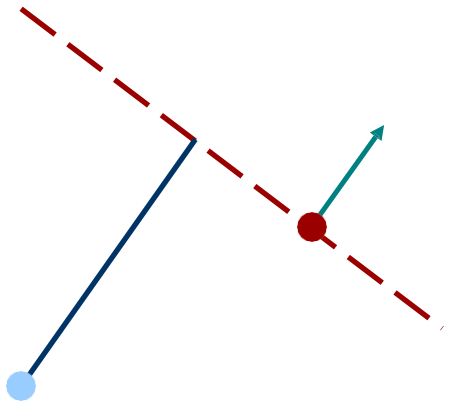
- Project along normal, intersect other point set

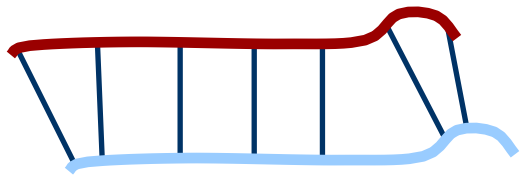


Slightly better than closest point for smooth structures, worse for noisy or complex structures

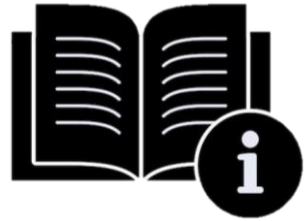
Point-to-Plane Error Metric

- Instead of measuring the distance between corresponding points directly, point-to-plane considers the distance from each point to the plane defined by its corresponding point in the other point cloud
- Using point-to-plane distance (instead of point-to-point) lets flat regions slide along each other [Chen & Medioni 91]

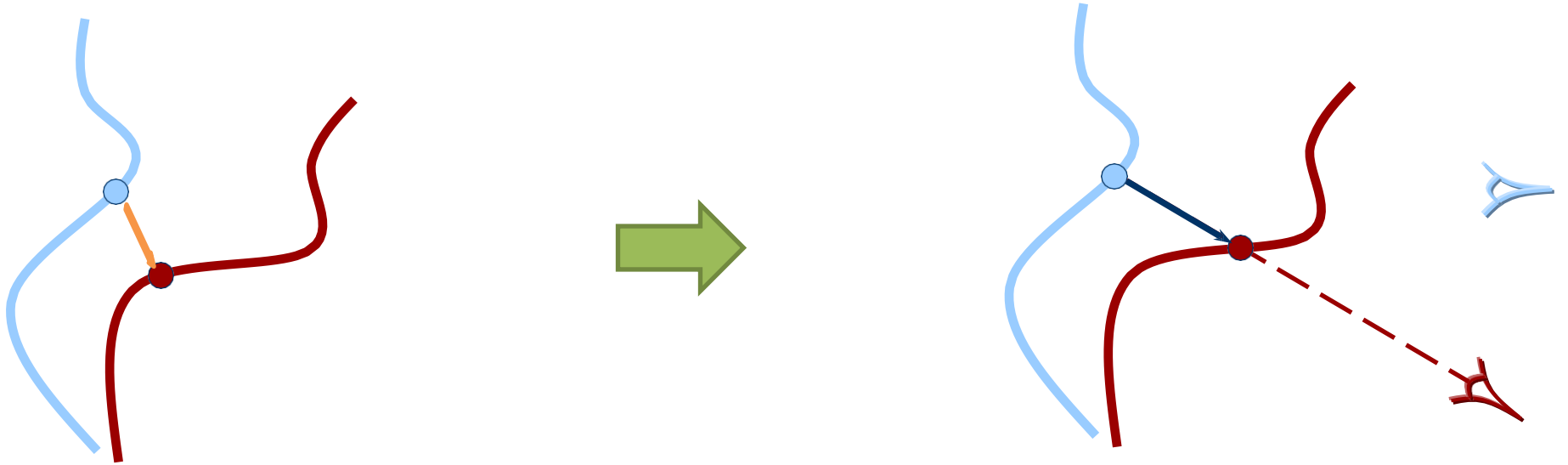




Projection

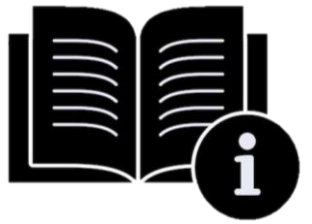


- Finding the closest point is the most expensive stage of the ICP algorithm
- Idea: simplified nearest neighbor search
- For range images, one can project the points according to the viewpoint [Blais 95]



https://ieeexplore.ieee.org/abstract/document/400574

Projection-Based Matching



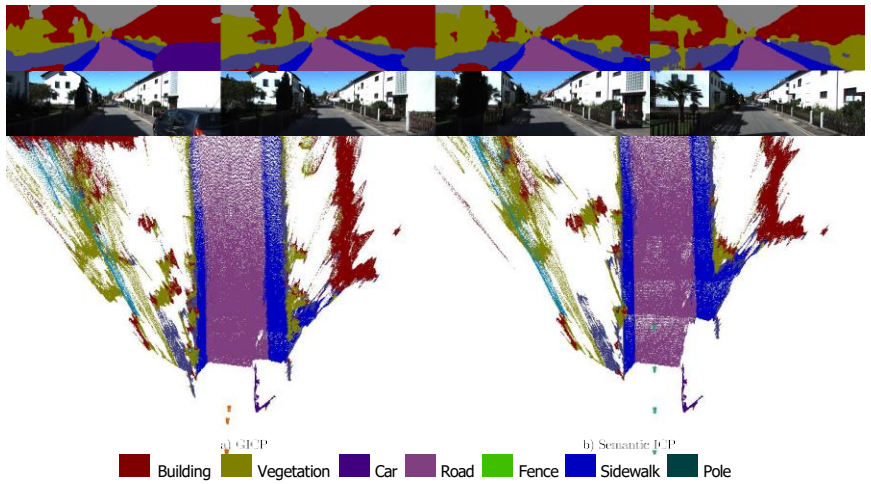
- Slightly worse alignments per iteration
- Each iteration is one to two orders of magnitude faster than closest-point
- Requires point-to-plane error metric

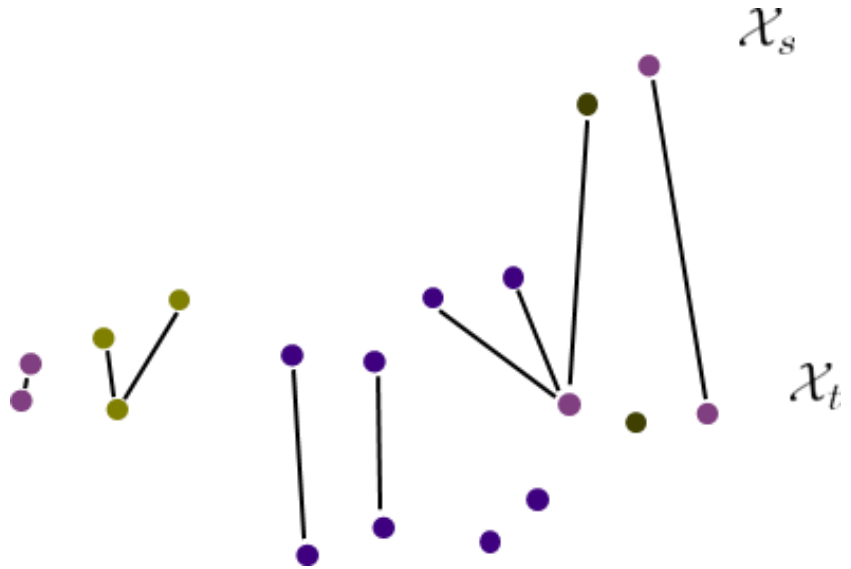
Closest Compatible Point

- Improves the previous two variants by considering the **compatibility** of the points
- Compatibility can be based on normals, colors, etc.
- In the limit, degenerates to feature matching

Example: Semantic ICP

http://robots.engin.umich.edu/publications/sparkison-2018a.pdf





Nearest Neighbor search using k-d tree

- FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces:
  https://www.cs.ubc.ca/research/flann/

- nanoflann: a C++11 header-only library for Nearest Neighbor (NN) search with KD-trees
  https://github.com/jlblancoc/nanoflann

- How to use a k-d tree to search using the Point Cloud Library (PCL):
  https://pcl.readthedocs.io/projects/tutorials/en/latest/kdtree_search.html

ICP Variants

1. Point subsets (from one or both point sets)
2. Weighting the correspondences
3. Nearest neighbor search
4. Rejecting certain (outlier) point pairs

Rejecting (outlier) point pairs

- sorting all correspondences with respect to the error and deleting the worst t%, Trimmed ICP (TrICP) [Chetverikov et al. 2002]

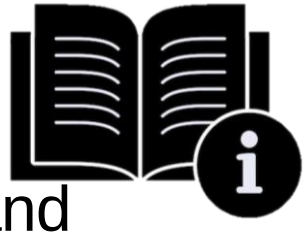
- t is to be estimated with respect to the overlap

  → Problem: Knowledge about the overlap is necessary or has to be estimated

ICP-Summary

- ICP is a powerful algorithm for calculating the displacement between scans.
- The major problem is to determine the correct **data associations**.
- Given the correct data associations, the transformation can be computed efficiently using SVD.
- In practice, ICP does *not* always converge to the correct solution

ICP & other registration algorithms:

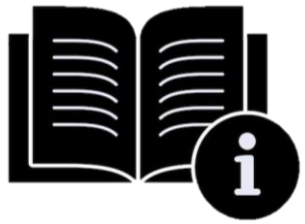


- Iterative Closet Point using point-to-point distance (previous slides; Besl and McKay (1992)).
- Iterative Closet Point using point-to-plane distance (see Y. Chen and G. Medioni (1991)).
- Generalized Iterative Closet Point; probabilistic formulation and maximum likelihood estimation (see A. Segal, D. Haehnel, and S. Thrun (2009)).
- Normal Distributions Transform (NDT) and its extensions.
- Extensions are available by introducing extra measurements or avoiding hard data associations using, for example, Expectation Maximization algorithm. For instance, see “Point Clouds Registration with Probabilistic Data Association”.

Challenges in point cloud registration

- **Unknown correspondences** between two point clouds.
- Finding local minima (instead of global) and the need for a "good" initial guess.
- Partial and unknown overlap between two point clouds.
- Variable density of the overlapping area, i.e., sparse to dense registration.
- Sensors with limited field of view or highly noisy sensory data make the registration problem harder to solve
- In practice, there are **outliers** in the data association

Small-scale problems & offline processing



If the size of the problem is not too large or we have unlimited time for processing data:

- We can exhaustively use multiple initializations to find the best solution.
- We can simultaneously solve for a permutation matrix to find the best correspondences and the optimal rigid body transformation.
- An example is Go-ICP, which uses Branch-and-Bound (BnB) method for global optimization.

RANSAC

Ransac: Basis

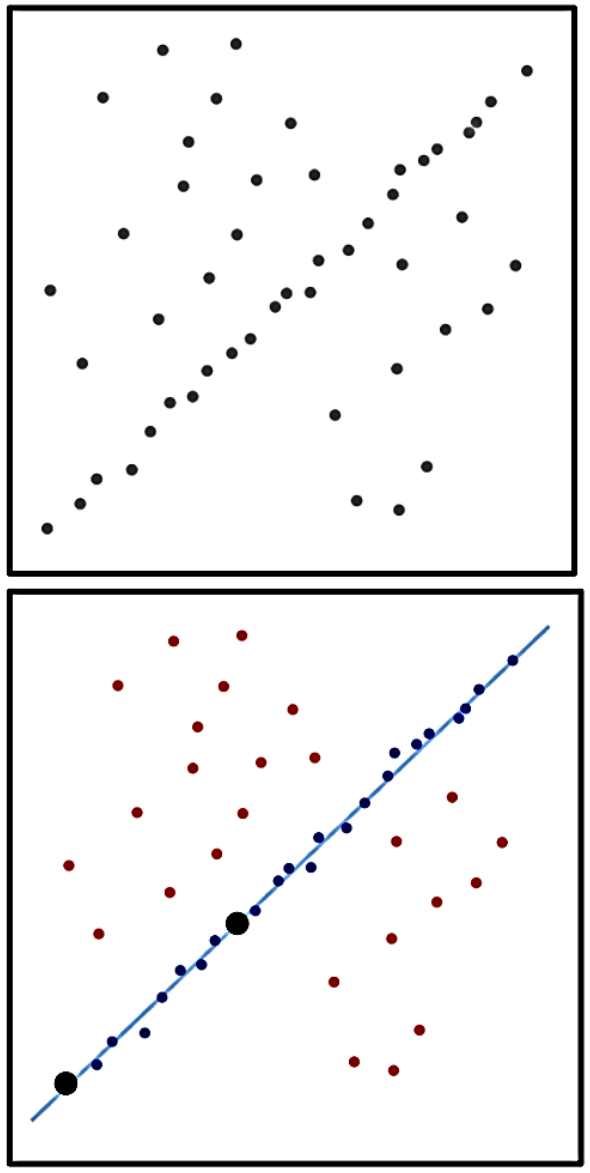
- **Ran**dom **Sa**mple **C**onsensus
  - Hypothesize and test.

- Used for Parametric Matching
  - Want to match two things.
  - Hypothesized match can be described by parameters (eg., translation, affine.)

- Match enough features to determine a hypothesis.
  - See if it is good.
  - Repeat.

Ransac Example: Grouping Points into Lines

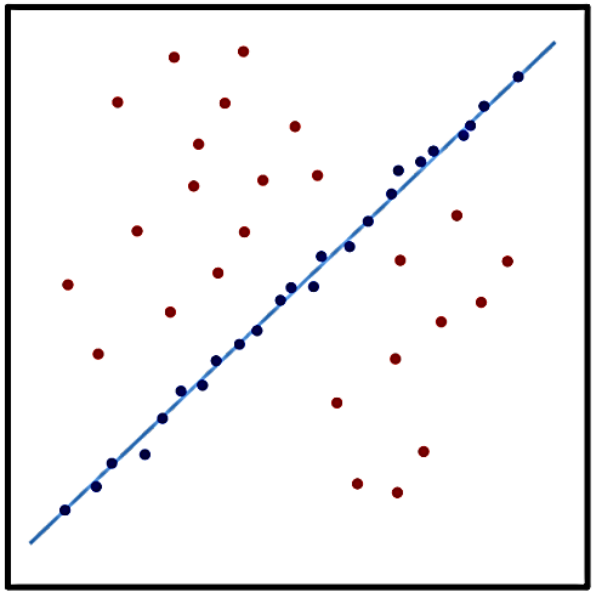
- Select a random subset of the original data. Call this subset the hypothetical inliers.
- A model is fitted to the set of hypothetical inliers.
- All other data are then tested against the fitted model.
- The estimated model is reasonably good if enough points have been classified as part of the consensus set.
- Afterwards, the model may be improved by re-estimating it using all members of the consensus set.



Ransac Complexity

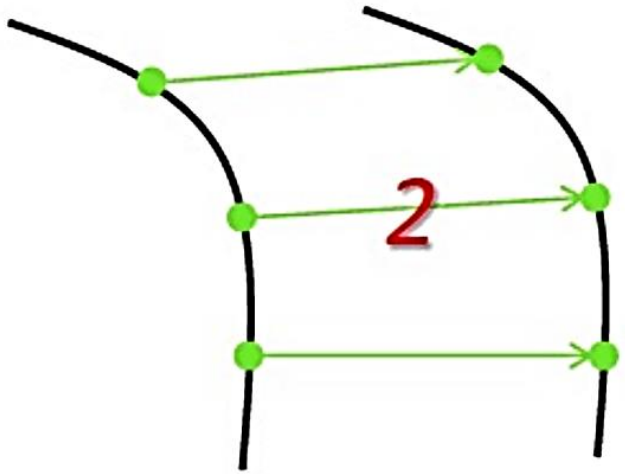
- Complexity?
- How many samples?
  - **p** is fraction of points on the line.
    - Fraction of inlayer/total
  - **n** points needed to define hypothesis (2 for lines)
  - **k** number of trials.
- Probability that after **N** trials I've the correct solution is:

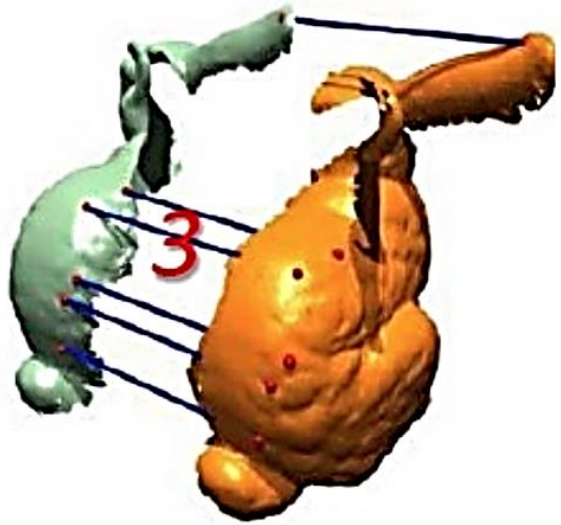
$$1-(1-p^n)^N$$



Ransac

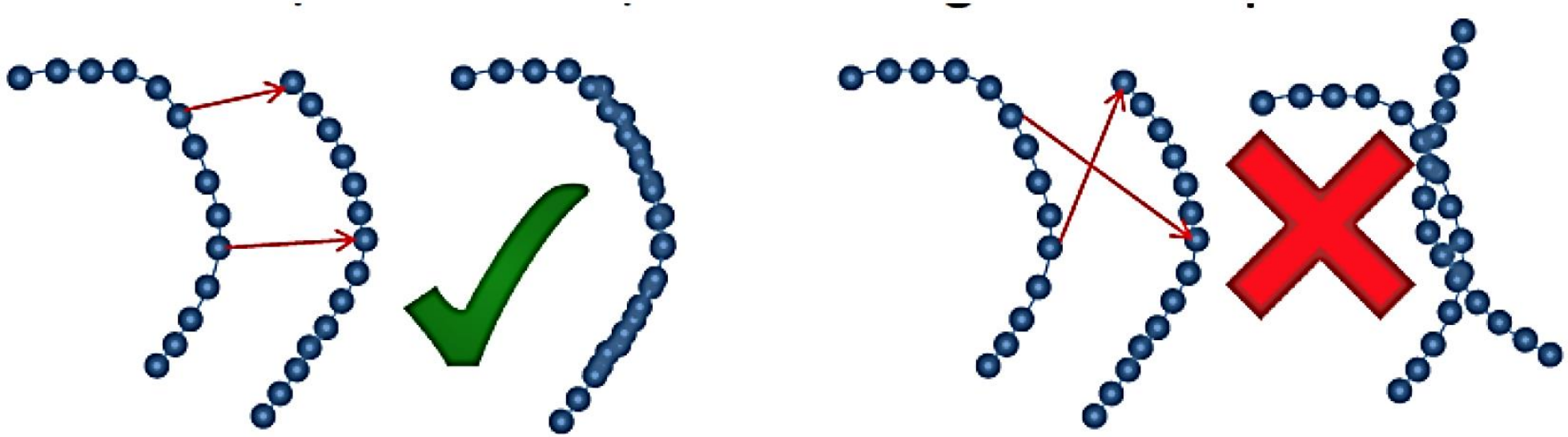
- How many point-pairs specify a rigid transform?
  - In R2?
  - In R3?
- Additional constraints?
  - Distance preserving
  - Stability?





Ransac

- Preprocessing: sample each object
- Iterate
  - Step I: Sample three (two) pairs, check distance constraints
  - Step II: Fit a rigid transform
  - Step III: Check how many point pairs agree. If above threshold, terminates; otherwise goes to Step I

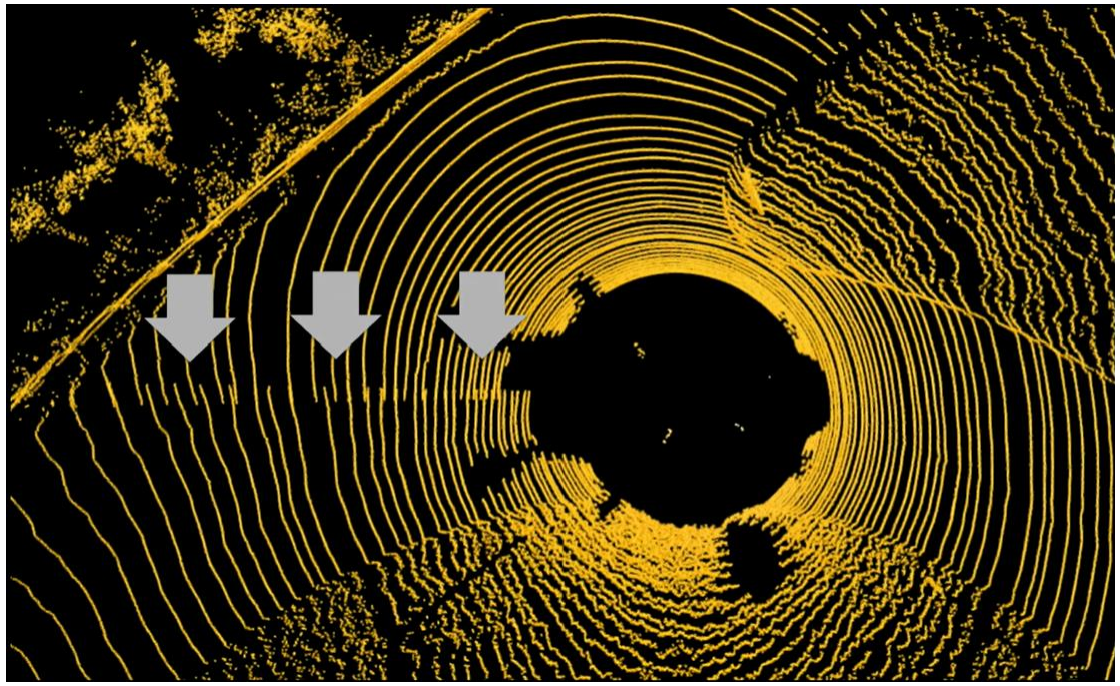


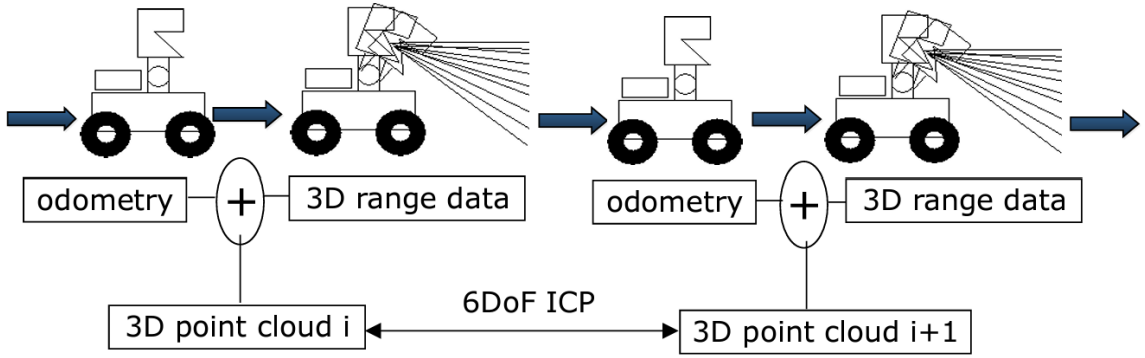
Ransac

- Sampling
  - Feature point detection
- Correspondences
  - Use feature descriptors
  - Denote a larger success rate p
  - Probability a descriptor identifies the correct match
  - Try only candidated made by pair of samplese with similar descriptor.
- Basic analysis
  - The probability of having a valid triplet $\mathbf{p^3}$
  - The probability of having a valid triplet in **N** trials is $\mathbf{1-(1-p^3)^N}$

LiDAR Odometry - Challenges

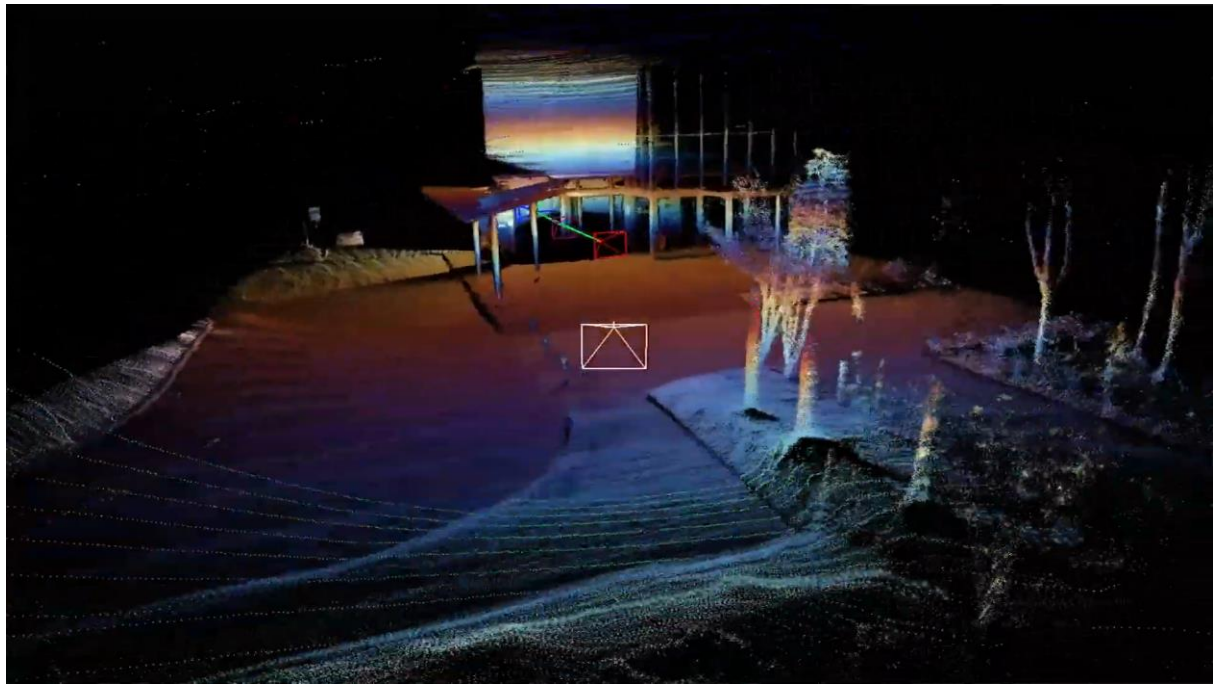
- Corresponding points in consecutive Lidar scans do not necessarily correspond to the same point in the environment!
- Motion distortion - motion affects Lidar scan
- Accurate Lidar odometry/SLAM requires motion compensation (or scan de-skewing)
- Data and/or associations always have outliers – require robust registration methods





LiDAR Odometry - Summary

- Alignment of 2D/3D data points is needed for LiDAR odometry (is in general an important task in perception)
- ICP is the standard algorithm for calculating the transform between scans
- ICP estimates translation and rotation (i.e., pose) between the scans
- The major problem is to determine the correct data associations (and remove outliers)



https://www.youtube.com/watch?v=bWmjo9zONLE

References

- LOAM: Lidar Odometry and Mapping in Real-time (https://www.ri.cmu.edu/pub_files/2014/7/Ji_LidarMapping_RSS2014_v8.pdf)
- Least-squares fitting of two 3D point sets (https://ieeexplore.ieee.org/document/4767965)
- Fast Point Feature Histograms (FPFH) for 3D Registration (https://www.cvl.iis.u-tokyo.ac.jp/class2016/2016w/papers/6.3DdataProcessing/Rusu_FPFH_ICRA2009.pdf)
- Generalized-ICP (http://www.roboticsproceedings.org/rss05/p21.pdf)
- Point Cloud Library (PCL): https://github.com/PointCloudLibrary/pcl
- Open3D: https://github.com/IntelVCL/Open3D
- University of Michigan's Mobile Robotics: methods & algorithms https://github.com/UMich-CURLY-teaching/UMich-ROB-530-public
- Iterative Closest Point - Cyrill Stachniss https://www.ipb.uni-bonn.de/html/teaching/msr2-2020/sse2-03-icp.pdf
- Ransac and ICP: https://vcg.isti.cnr.it/~cignoni/GMP2324/PDF/GP2324_13_RansacICP.pdf