Unknown correspondence and ICP

D.A. Forsyth

University of Illinois at Urbana Champaign

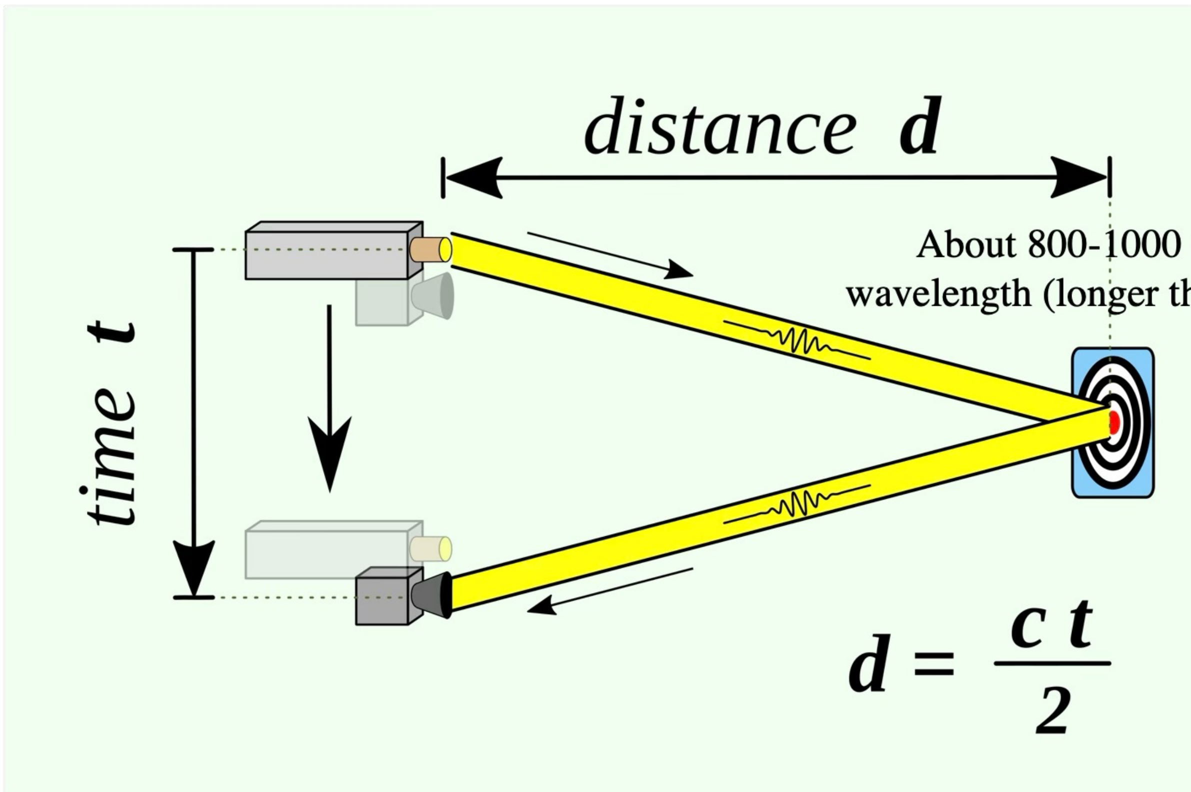
What if you don’t know correspondences?

- RANSAC isn’t usually enough
- Issue:
  - volume of outliers can be extremely high
  - eg N points in image 1, N points in image 2
    - N inliers, at most
    - N (N-1) outliers or more

A very important reason to care about interest points and descriptors

  - Much worse for LIDAR correspondence
    - because you mostly can’t do descriptors

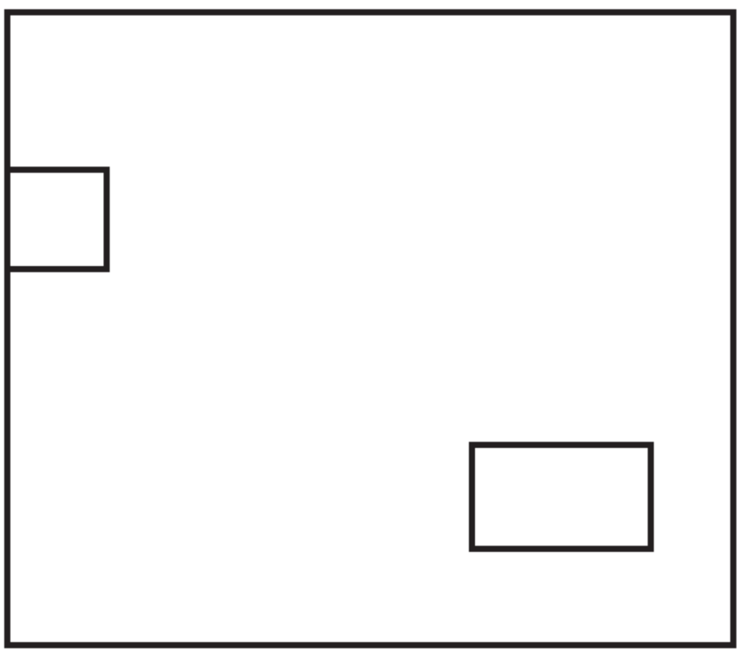
LIDAR produces point clouds



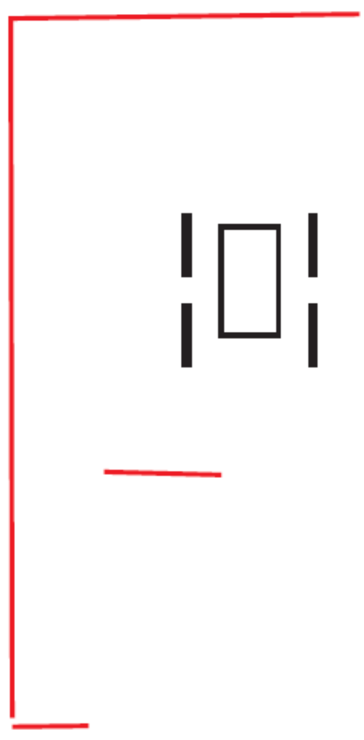
Wikipedia

LIDAR registered to map

Map

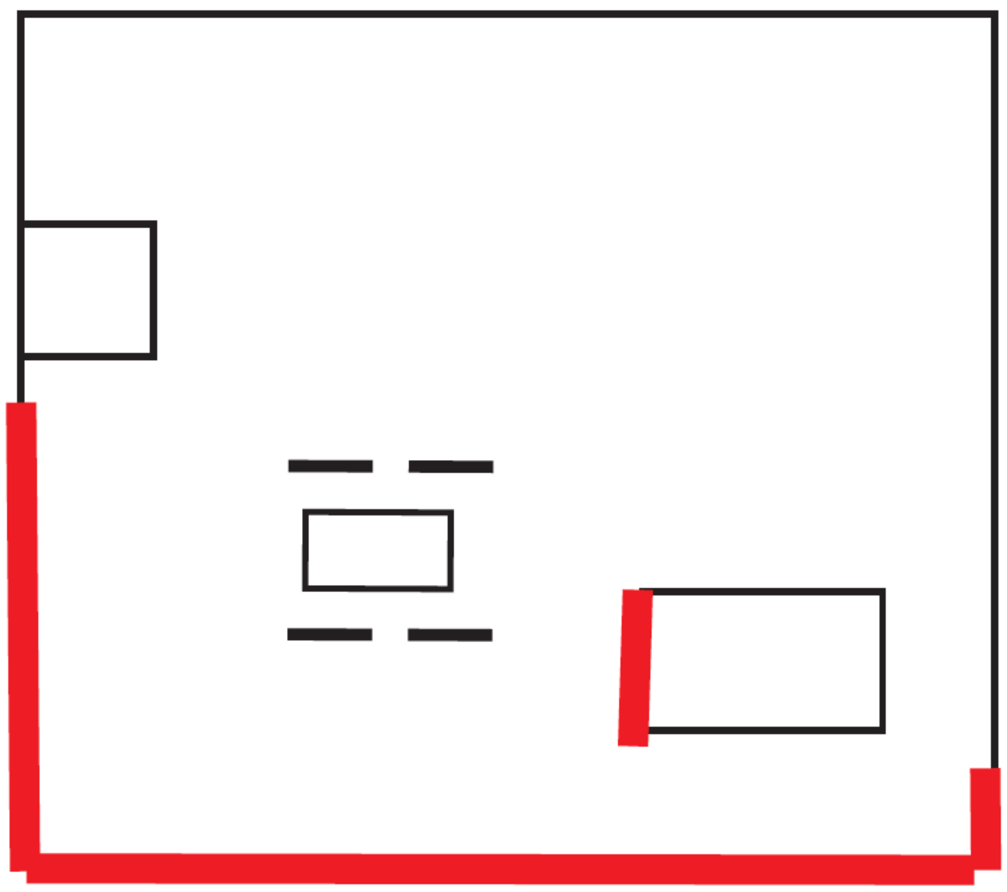


LIDAR



Yield your location

Map



Registering meshes to LIDAR

- I have a CAD model of a car
    - (large triangular mesh)
- Where is this car in LIDAR data?
    - registration problem
- How is a mesh a point cloud?
    - sample points on the mesh
    - vertices

Iterated closest points or ICP

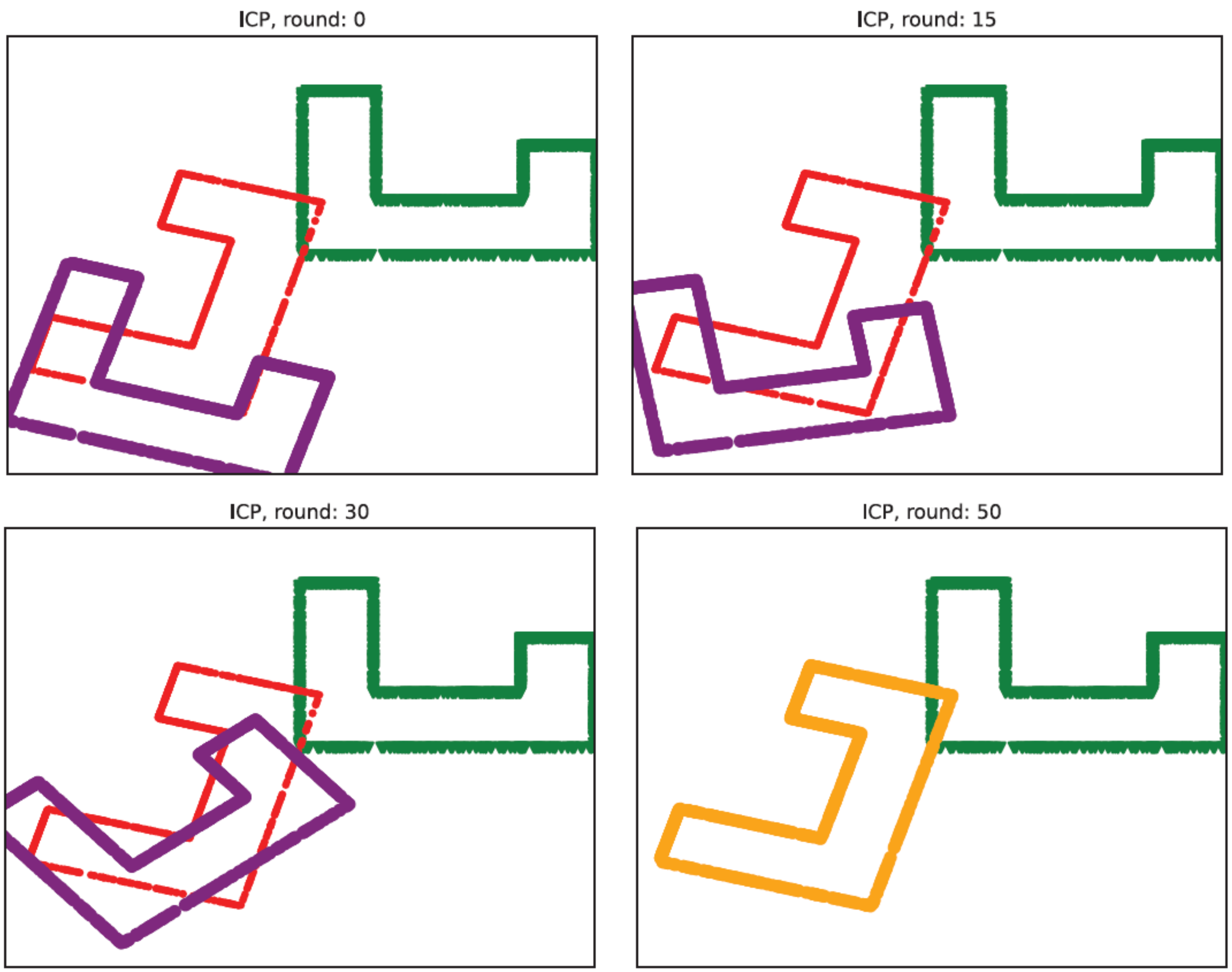
- Idea:
  - If the transformation is nearly the identity,
  - then nearest point is likely correspondence
- Strategy:
  - Start with good transformation
  - Iterate:
    - Estimate correspondences assuming tx is right
    - Re-estimate tx

ICP

Formally, start with a transformation estimate $\mathcal{T}_1$, a set of $\mathbf{m}_i^{(1)} = \mathcal{T}^{(1)}(\mathbf{y}_i)$ (the *running points*) and then repeat three steps:

- **Estimate correspondences** using the transformation estimate. Then, for each $\mathbf{x}_i$, we find the closest $\mathbf{m}^{(n)}$ (say $\mathbf{m}_c^{(n)}$); then $\mathbf{x}_i$ corresponds to $\mathbf{m}_{c(i)}^{(n)}$.
- **Estimate a transformation** $\mathcal{T}^{(n+1)}$ using the corresponding pairs.
- **Update the running points** by mapping $\mathbf{m}_i^{(n)}$ to $\mathcal{T}^{(n+1)}(\mathbf{m}_i^{(n)})$ and

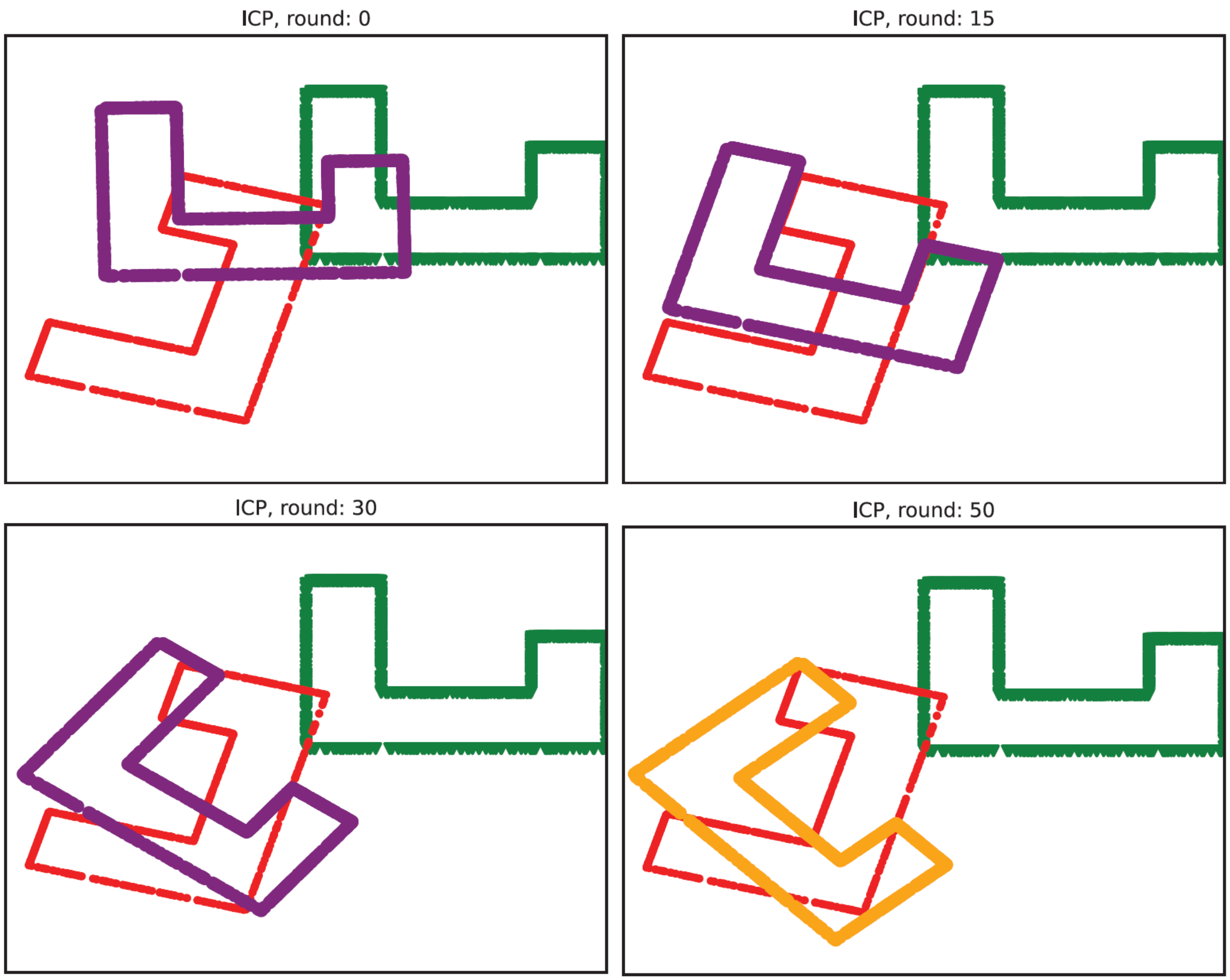
Can converge quite fast



- Red – target
- Green – source
- Purple – running points

- Green at 0 -> Purple at 0 – initial transformation

ICP doesn’t always converge



- Red – target
- Green – source
- Purple – running points

- Green at 0 -> Purple at 0 – initial transformation

ICP Issues

- You have to do an awful lot of nearest neighbors
  - particularly if the point cloud is big
  - subsample the point cloud
  - approximate nearest neighbors

- There are still robustness issues
- Ideas:
  - drop correspondences for large distances
  - use IRLS at each round

Resampling

- If the two point clouds are big, resample
  - Choose some number N of points that is acceptable
  - Draw N points from point clouds
    - uniformly and at random
  - Do this with replacement, because its easier

Approximate nearest neighbors

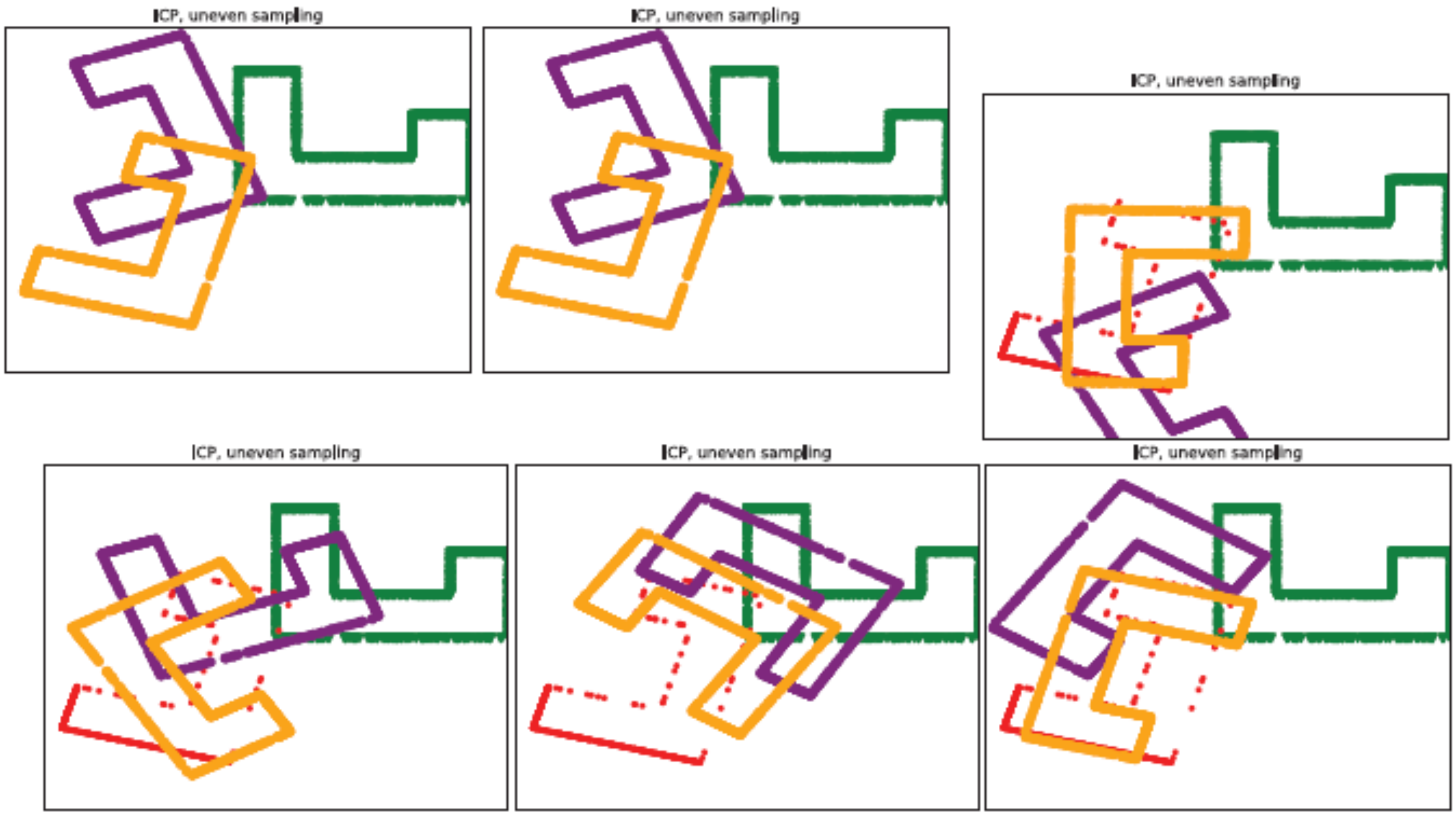
- ISSUE:
  - do you build a new tree for every iteration?
- Strategies:
  - Space is almost always 2D or 3D, so you can grid it
  - It is easy to test what grid bin a point falls in
    - (truncate, round)
  - Build a big enough grid and use that

ICP Issues

- You have to do an awful lot of nearest neighbors
  - particularly if the point cloud is big
  - subsample the point cloud
  - approximate nearest neighbors

- There are still robustness issues
- Ideas:
  - drop correspondences for large distances
  - use IRLS at each round

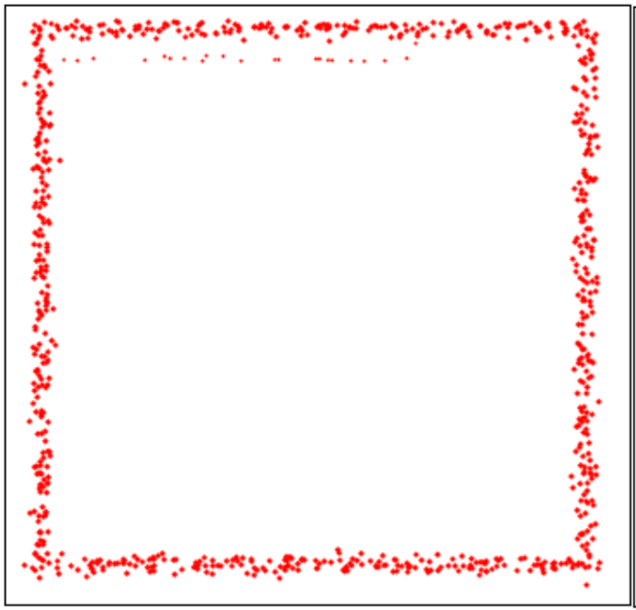
Uneven sampling -> ICP problems



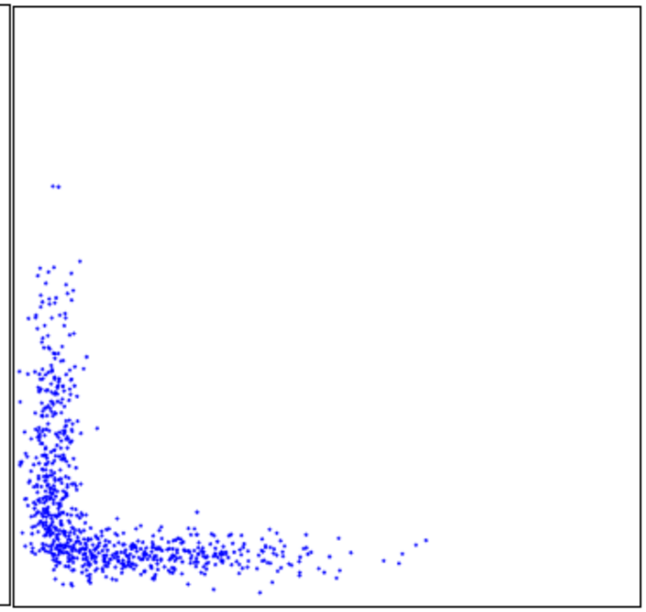
- Red – target
- Green – source
- Purple – running points

- Green at 0 -> Purple at 0 – initial transformation

Issue for LIDAR



Map



Vehicle observations

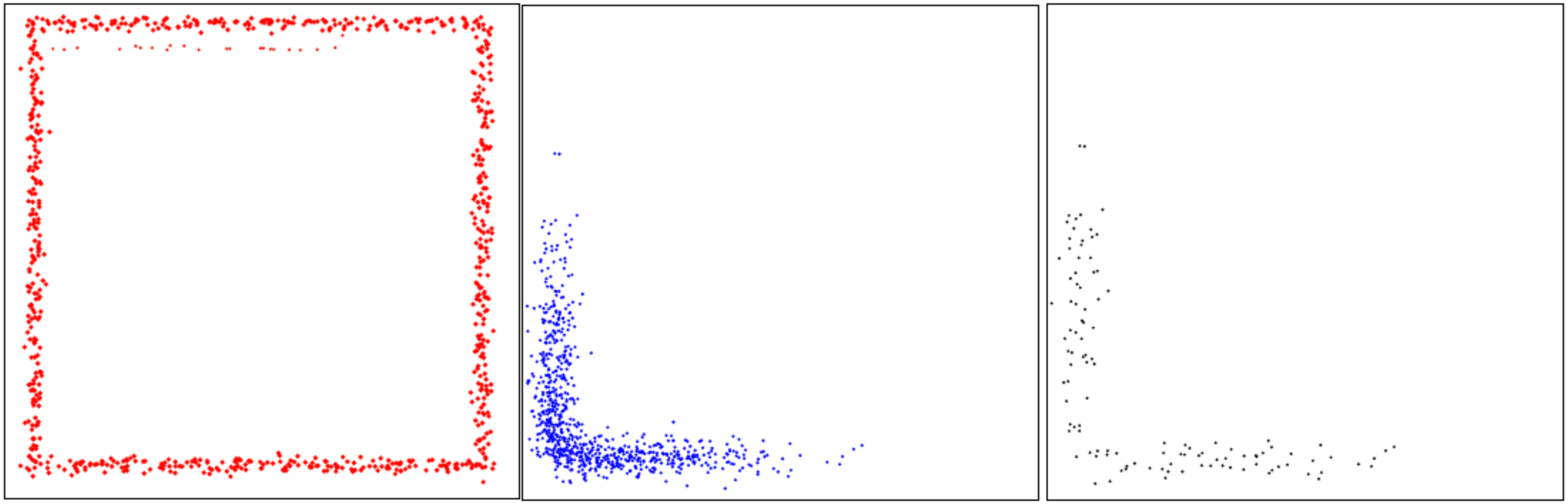
Stratifying a sample

- Cut the space into even bins
  - for example, a grid

- Read points into bins

- Resample by:
  - Iterate:
    - Choose a bin uniformly and at random
    - Choose a point from that bin uniformly and at random

Issue for LIDAR



Map

Vehicle observations

Stratified resample of observations

Biasing by normal can help

- Resample the point cloud so that there are about the same number of points with each normal

- How?
  - cut the unit sphere into bins,
  - read points into bins
  - resample by:
    - sample bin uniformly and at random
    - sample points in bin uniformly and at random

Think about this...

16.9. Imagine you obtain two LIDAR images of the same object from two different locations. Why do you not expect a near exact correspondence between the points in these two point clouds? (hint: this *isn't* about noise).