Point sets, Maps and Navigation

D.A. Forsyth

Issues

- Where am I?
  - Simplest: register observations and motion to a map
    - correspondence and robustness
- Build a map
  - Register observations to one another
    - global consistency
- Incorporating motion models
  - Registration should benefit from knowledge of motion
    - Filtering

Simplest case

- Registration with known correspondence
  - No motion model
  - 3D observations of known beacons at known 3D locations
    - beacons y_i; observations x_i
    - (for generality) weights w_i
- Problem:
  - choose rotation R, translation t to minimize

$$C(R, \mathbf{t}) = \sum_i w_i \left(R\mathbf{x}_i + \mathbf{t} - \mathbf{y_i}\right)^T \left(R\mathbf{x}_i + \mathbf{t} - \mathbf{y_i}\right)$$

  - THIS CAN BE DONE IN CLOSED FORM!

The translation

- Solve for translation as function of R

$$\nabla_{\mathbf{t}} C = \mathbf{0} = R(\sum_i w_i \mathbf{x}_i) + \mathbf{t}(\sum_i w_i) - (\sum_i w_i \mathbf{y}_i)$$

- So

Weighted centroids

$$\mathbf{t} = \overline{\mathbf{y}} - R\overline{\mathbf{x}}$$

- Plug this into cost function to get

$$G(R) = \sum_i w_i (R(\mathbf{x}_i - \overline{\mathbf{x}}) - (\mathbf{y}_i - \overline{\mathbf{y}}))^T (R(\mathbf{x}_i - \overline{\mathbf{x}}) - (\mathbf{y}_i - \overline{\mathbf{y}}))$$

The rotation

$$G(R) = \sum_i w_i (R(\mathbf{x}_i - \overline{\mathbf{x}}) - (\mathbf{y}_i - \overline{\mathbf{y}}))^T (R(\mathbf{x}_i - \overline{\mathbf{x}}) - (\mathbf{y}_i - \overline{\mathbf{y}}))$$

- Substitute

$$G(R) = \sum_i w_i (R(\mathbf{u}_i) - (\mathbf{v}_i))^T (R(\mathbf{u}_i) - (\mathbf{v}_i))$$

- Expand

$$G(R) = \sum_i w_i \left[\mathbf{u}_i^T \mathbf{u}_i - 2\mathbf{v}_i R \mathbf{u}_i + \mathbf{v}_i^T \mathbf{v}_i \right]$$

- So MAXIMIZE

$$H(R) = \sum_i w_i \mathbf{v}_i R \mathbf{u}_i$$

The rotation

- Rewrite using

$$H(R) = \sum_i w_i \mathbf{v}_i R \mathbf{u}_i$$

$$U = [\mathbf{u}_1, \mathbf{u_2}, \ldots]$$

- To get:

$$H(R) = \mathrm{Trace}\left[W V^T R U\right]$$

- Rotate through Trace to get:

$$H(R) = \mathrm{Trace}\left[R \underline{U W V^T}\right]$$

- Rewrite

This is data

$$H(R) = \mathrm{Trace}\left[R D\right]$$

The SVD (in case you don't recall!)

$$D = A \Sigma B^T$$

- For any D
- A is orthonormal, B is orthonormal, Sigma is diagonal
  - by convention, diagonal values are sorted by magnitude
  - we drop zero diagonals, and corresponding columns of B, A^T
    - they don't do anything
- A staple of numerical analysis
  - stable, well-behaved, etc. algorithms easily available
  - partial SVDs available
  - works fine at very large scales
  - generally, a good thing

The rotation

$$H(R) = \mathrm{Trace}\left[RD\right]$$

- SVD data

$$D = A\Sigma B^T$$

- Substitute, and rotate:

$$H(R) = \mathrm{Trace}\left[RA\Sigma B^T\right] = \mathrm{Trace}\left[\Sigma \underline{B^T R A}\right]$$

-

This must be orthonormal!

The rotation

- We must maximise:

$$H(R) = \mathrm{Trace}\left[\Sigma M(R)\right]$$

  - (where M(R) is orthonormal)

- But this means that M(R) has 1 or -1 on the diagonal!

- So if

$$H(R) = \mathrm{Trace}\left[RA\Sigma B^T\right] = \mathrm{Trace}\left[\Sigma B^T RA\right]$$

- the orthonormal matrix we're looking for is:

$$R = BA^T$$

Final details

- Careful:

$$R = BA^T$$

  - could be a reflection (ie det=-1; a flip; etc.)

- Fix:

$$R = B\left(\mathrm{diag}\left[1, 1, \det(BA^T)\right]\right)A^T$$

So far

- Given two sets of points
  - with known correspondences
  - weights
- We can find optimal rotation, translation to register
  - easily
  - in closed form

- We now know where we are
  - for (say) x_i 3D measurements, y_i beacons
- Missing:
  - what happens if we *don't* have correspondences?
  - robustness

ICP = Iterated closest points

- What if we *don’t* have correspondences?
- Idea:
  - Repeat until convergence:
    - each x corresponds to “closest” y
    - register
- Big simple idea, lots of variants
  - What is “closest”?
  - What if you have lots of points?

Introduction to Mobile Robotics

Iterative Closest Point Algorithm

Wolfram Burgard, Cyrill Stachniss,

Maren Bennewitz, Kai Arras

- Full set of slides is on web page
  - I'm going to show some to make major points

ICP-Variants

- Variants on the following stages of ICP have been proposed:

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

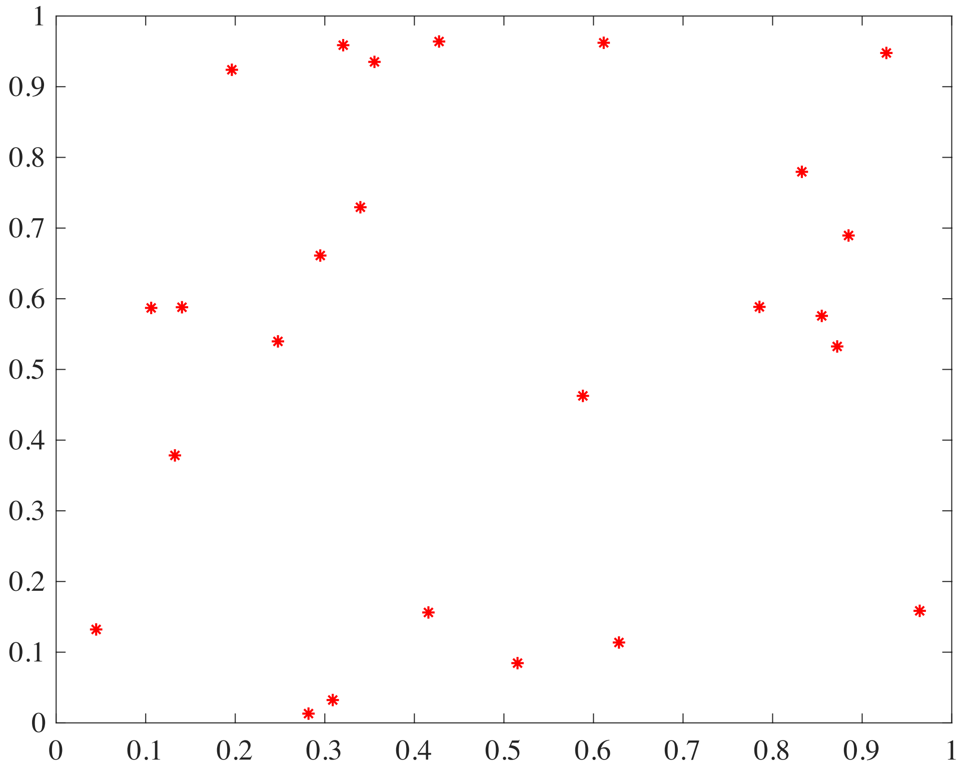
The issue here is efficiency - also, some points are more helpful than others (think corners)

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

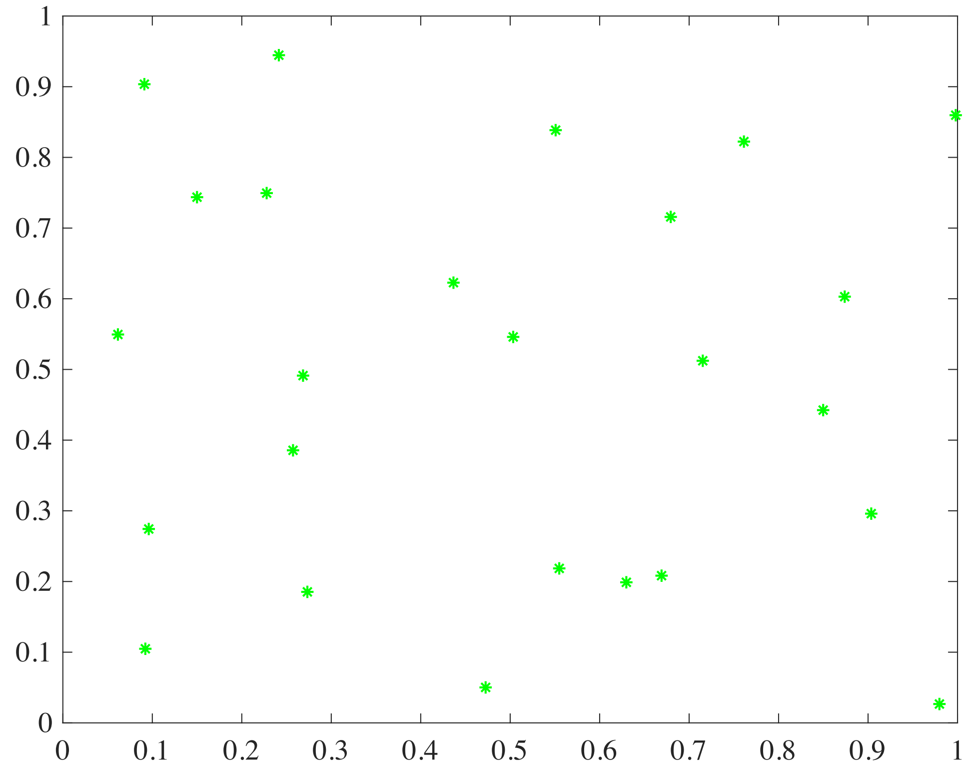
Selecting Source Points

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

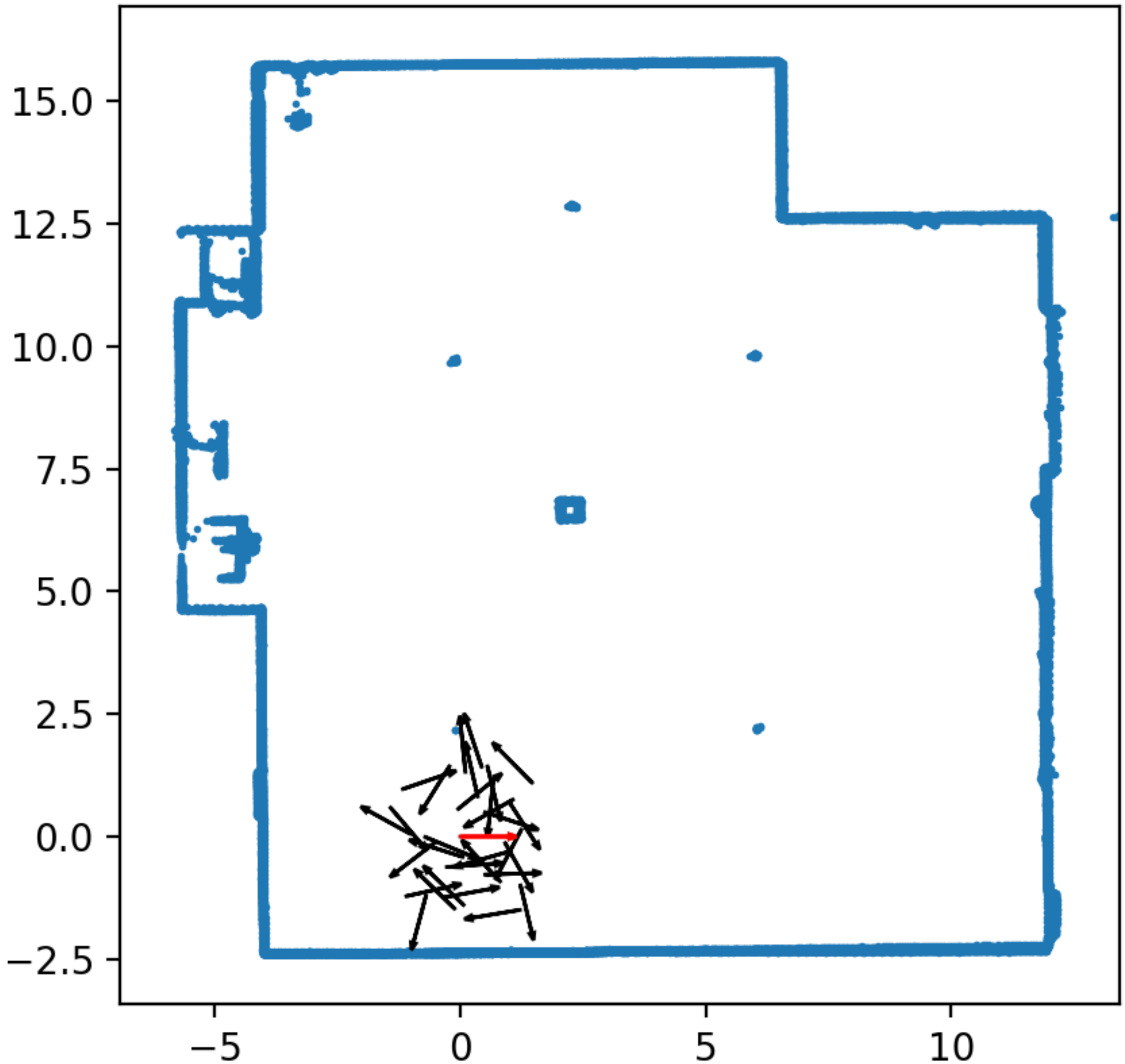
Uniform samples are shakey - stratify



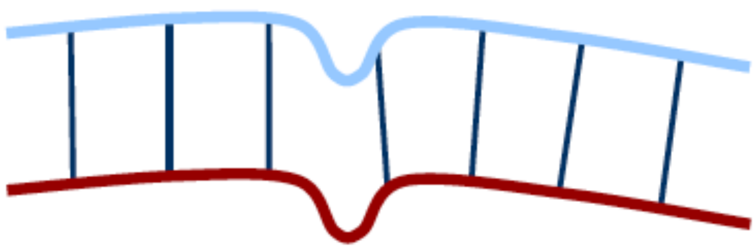
Uniform



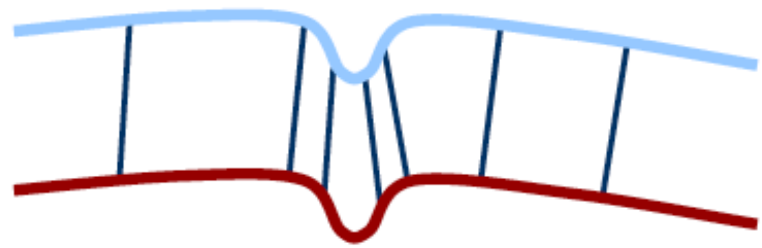
Block stratified


15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5
0
5
10

Normal-Space Sampling



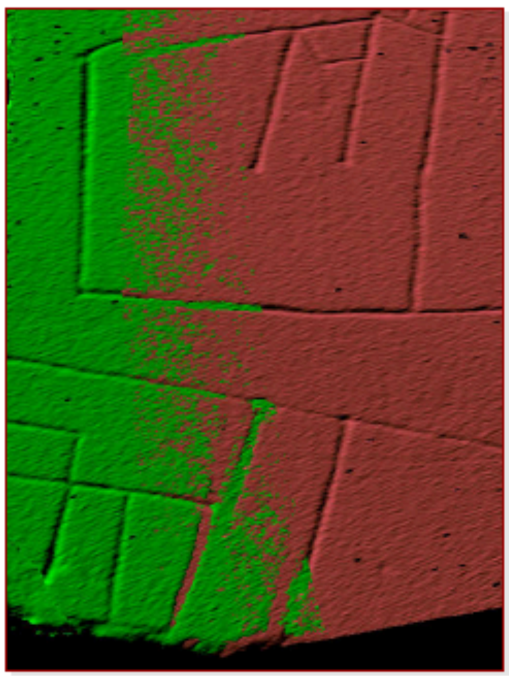
uniform sampling



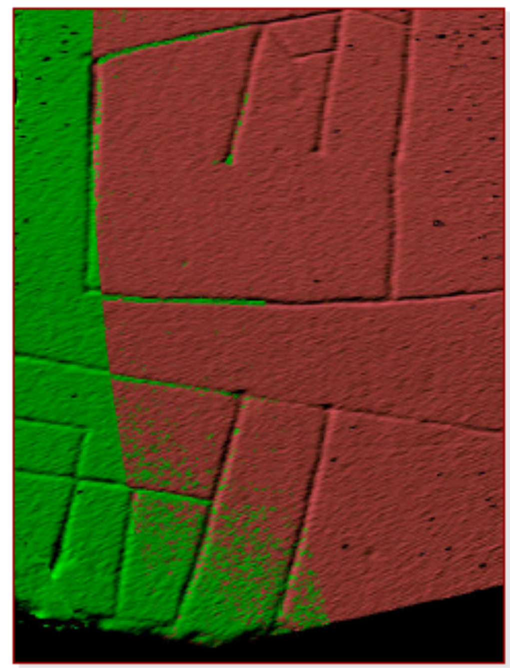
normal-space sampling

Comparison

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



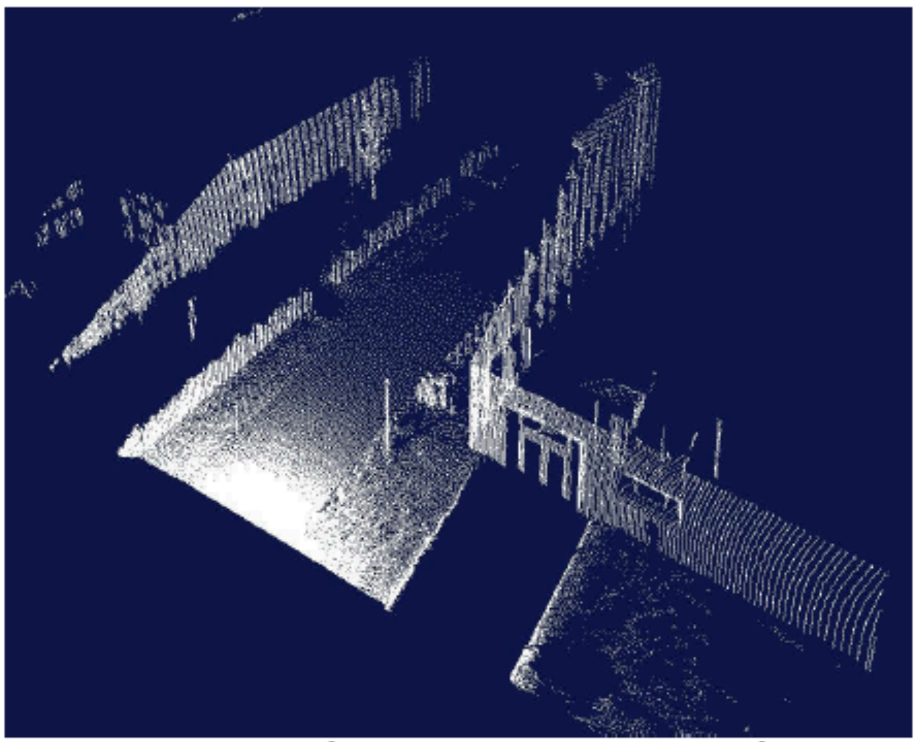
Random sampling



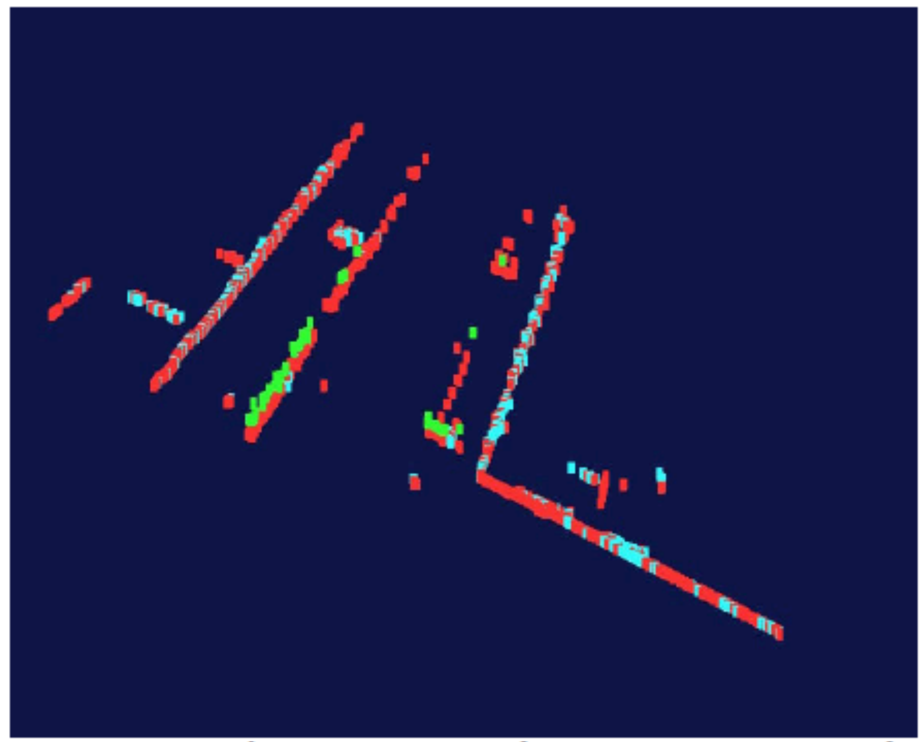
Normal-space sampling

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)

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

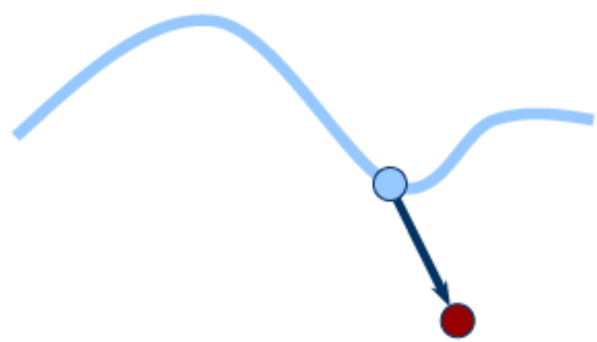
Data Association

- has greatest effect on convergence and speed
- Closest point
- Normal shooting
- Closest compatible point
- Projection
- Using kd-trees or oc-trees

Q: who corresponds with who?
Doesn't have to be closest!

Closest-Point Matching

- Find closest point in other the point set

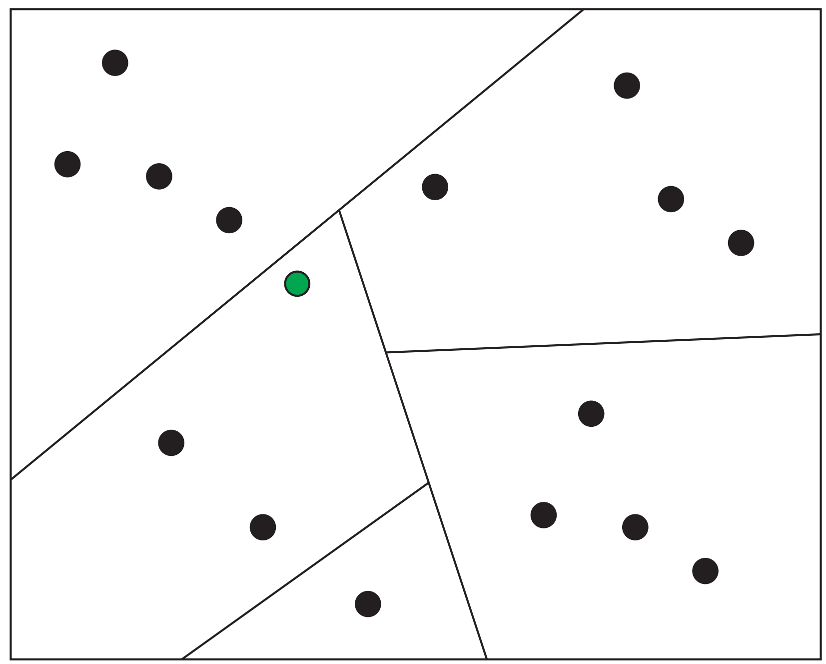


Closest-point matching generally stable, but slow and requires preprocessing

Speeding this up (in low D)

- We care about 2D, 3D
- Some correspondence errors may be tolerable.
  - We're making pooled estimates of rotation and translation
- Idea
  - target points into octree (kd tree, etc)
  - closest point *within tree cell*
    - which may not be the overall closest point!
    - whatever!
- Other hashing procedures could apply
  - but mostly more trouble than necessary in 2 or 3 D

Warning - KD trees aren't exact



This doesn't usually *matter* but…

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

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 there error and deleting the worst t%, Trimmed ICP (TrICP) [Chetverikov et al. 2002]
- t is to Estimate with respect to the Overlap

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