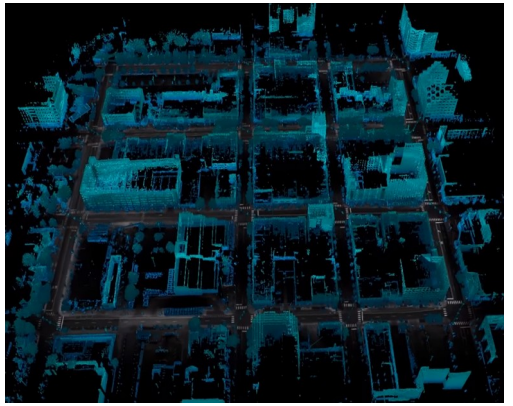
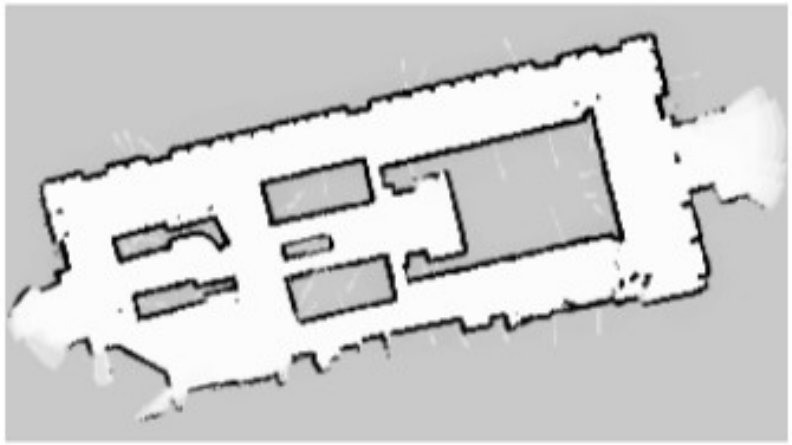
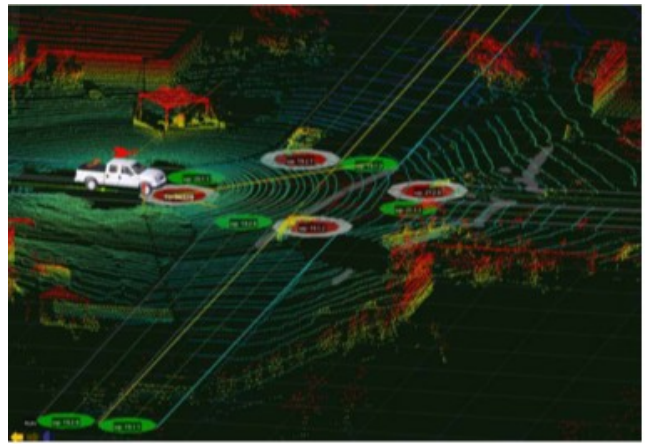
BYU Electrical & Computer Engineering





L: LIDAR AND SCAN MATCHING

ECEN 522R: Robotic Localization and Mapping (FALL 2021)

Some slides courtesy of Ryan Eustice

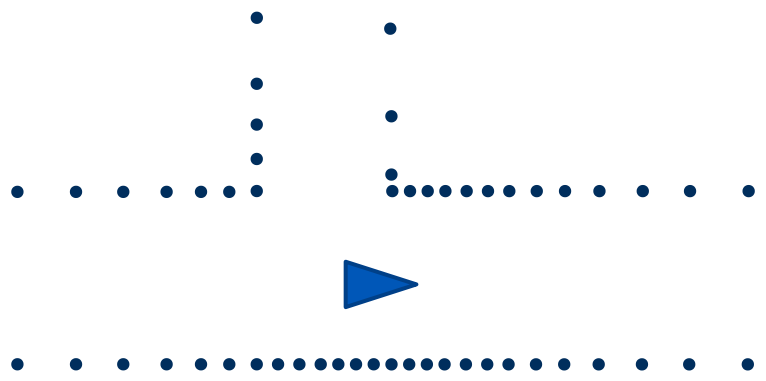
Today's Topic

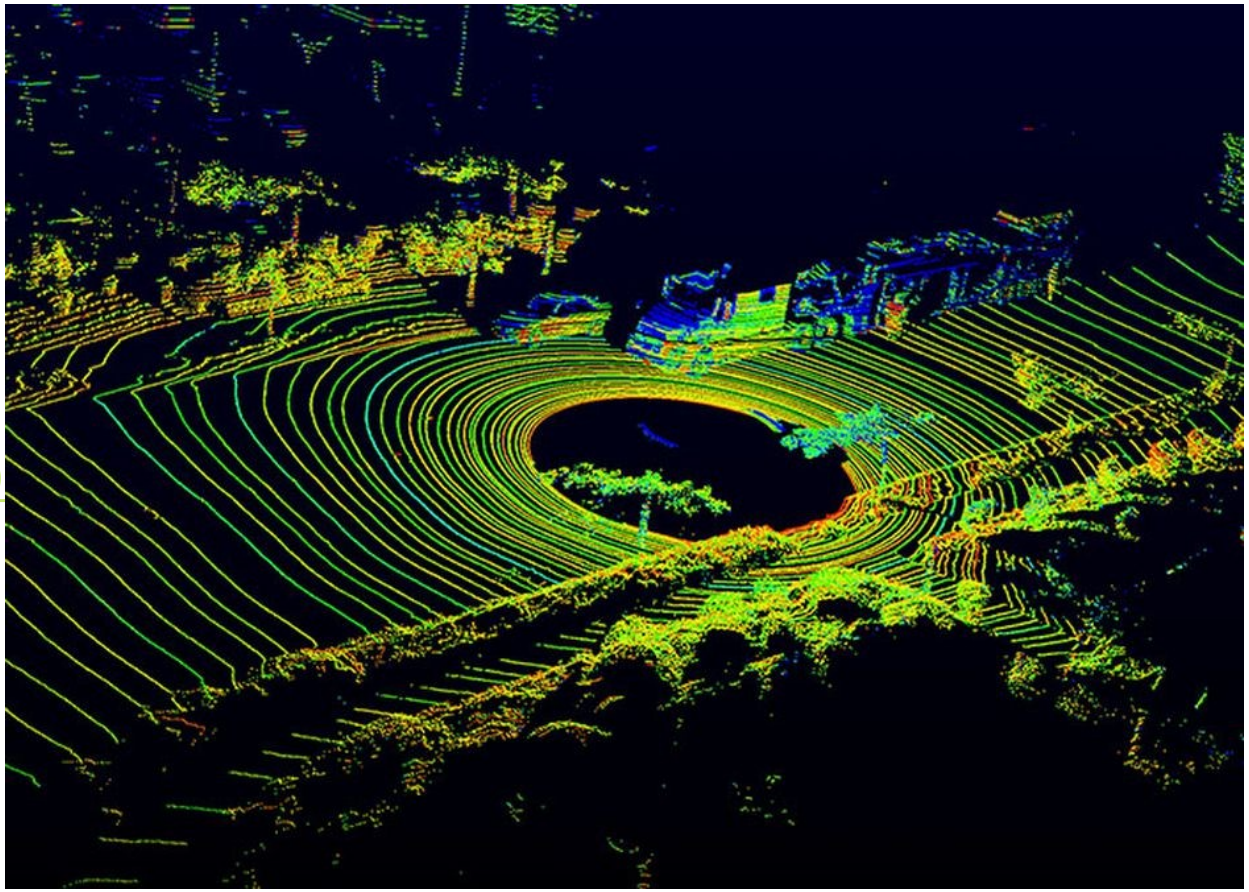
- Lasers / LiDAR
- Line Feature Extraction
  - Split-n-Fit
  - Agglomerative
  - RANSAC
- Scan Matching
  - Correlative
  - ICP / ICL

Laser Scanners / LiDAR

- Measures range via time-of-flight
  - Requires reflections from the surface (for example, glass will be ignored)

- LiDAR (Light Detection and Ranging)
  - E.g. Velodyne Lidar Products https://velodynelidar.com/products/
  - Datasheets on next two slides from https://www.mapix.com/wp-content/uploads/2018/07/63-9229_Rev-H_Puck-_D





Velodyne LiDAR®

Puck™

REAL-TIME 3D LiDAR SENSOR

Automotive | Robotics | Mapping | UAV | Security | Industrial

VLP-16

Velodyne LiDAR PUCK™

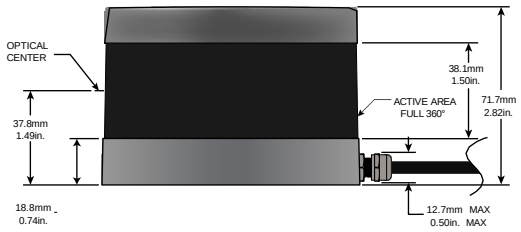
Velodyne's Puck (VLP-16) is the smallest, cost-optimized product in Velodyne's 3D LiDAR product range. Developed with mass production in mind, the Puck is far more cost-effective than comparable sensors, and it retains the key features of Velodyne's breakthroughs in LiDAR: Real-time, 360°, 3D distance and calibrated reflectivity measurements.

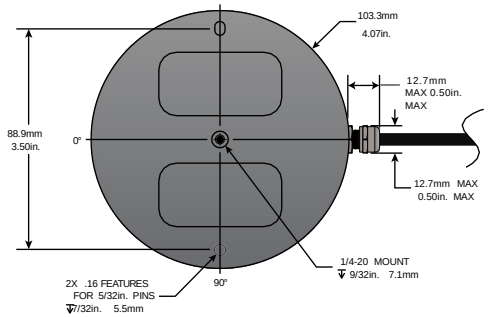
Real-Time 3D LiDAR

The VLP-16 has a range of 100 m, and the sensor's low power consumption, light weight, compact footprint and dual return capability make it ideal not only for autonomous vehicles but also for robotics, terrestrial 3D mapping and many other applications.

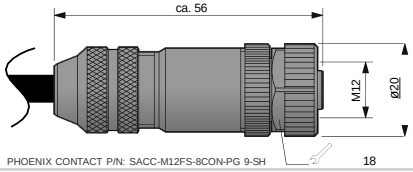
Velodyne's LiDAR Puck supports 16 channels, ~300,000 points/second, 360° horizontal field of view and a 30° vertical field of view, with ±15° up and down. The Puck does not have visible rotating parts, and is highly resilient in challenging environments while operating over a wide temperature range.

DIMENSIONS *(Subject to change)*

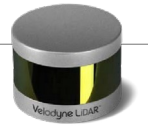




M12 CONNECTOR OPTION



For other connector options contact
Velodyne Sales (sales@velodyne.com)



Real-Time 3D LiDAR Sensor

The VLP-16 provides high definition 3-dimensional information about the surrounding environment.

	Specifications:
Sensor:	• 16 Channels • Measurement Range: 100 m • Range Accuracy: Up to ±3 cm (Typical)[1] • Field of View (Vertical): +15.0° to -15.0° (30°) • Angular Resolution (Vertical): 2.0° • Field of View (Horizontal): 360° • Angular Resolution (Horizontal/Azimuth): 0.1° – 0.4° • Rotation Rate: 5 Hz – 20 Hz • Integrated Web Server for Easy Monitoring and Configuration
Laser:	• Laser Product Classification: Class 1 Eye-safe per IEC 60825-1:2007 & 2014 • Wavelength: 903 nm
Mechanical/ Electrical/ Operational	• Power Consumption: 8 W (Typical)[2] • Operating Voltage: 9 V – 18 V (with Interface Box and Regulated Power Supply) • Weight: ~830 g (without Cabling and Interface Box) • Dimensions: See diagram on previous page • Environmental Protection: IP67 • Operating Temperature: -10°C to +60°C[3] • Storage Temperature: -40°C to +105°C
Output:	• 3D LiDAR Data Points Generated: - Single Return Mode: ~300,000 points per second - Dual Return Mode: ~600,000 points per second • 100 Mbps Ethernet Connection • UDP Packets Contain: - Time of Flight Distance Measurement - Calibrated Reflectivity Measurement - Rotation Angles - Synchronized Time Stamps (µs resolution) • GPS: $GPRMC and $GPGGA NMEA Sentences from GPS Receiver (GPS not included)

63-9229 Rev-H

For more details and ordering information, contact Velodyne Sales (sales@velodyne.com)

1. Typical accuracy refers to ambient wall test performance across most channels and may vary based on factors including but not limited to range, temperature and target reflectivity.
2. Operating power may be affected by factors including but not limited to range, reflectivity and environmental conditions.
3. Operating temperature may be affected by factors including but not limited to air flow and sun load.

CLASS 1 LASER PRODUCT

Aligning Scans (Measuring Motion)

- Two common cases:
  - Incremental motion, or using the LiDAR to augment or replace odometry
  - Loop closures, or identifying locations that have been previously visited and thereby reduce positional uncertainty

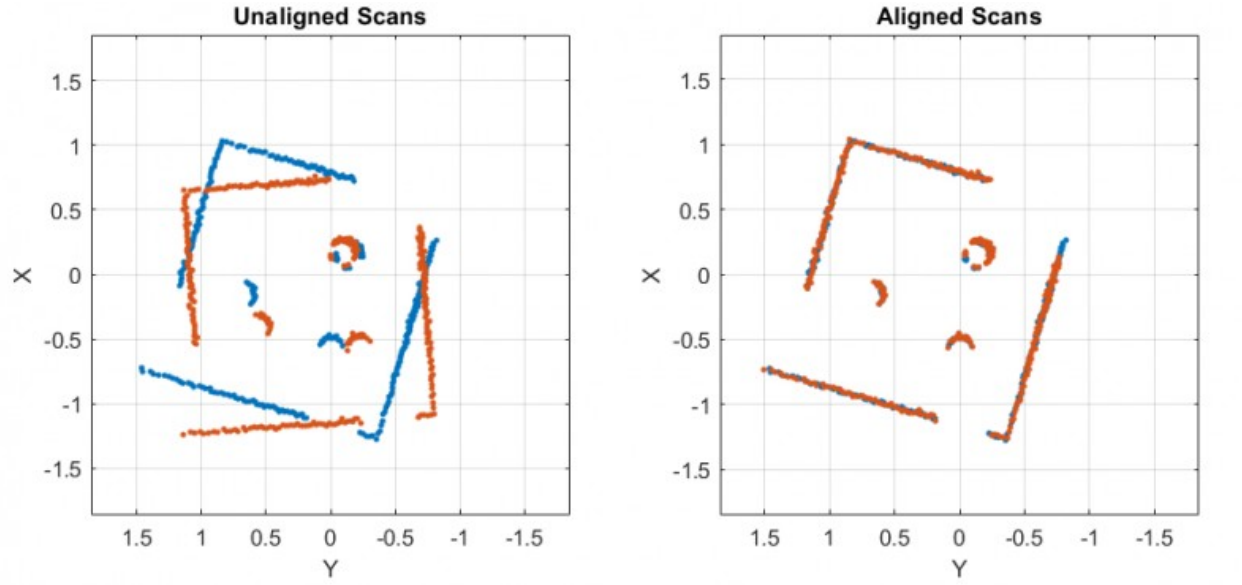


Image source: https://blogs.mathworks.com/racing-lounge/files/2018/09/autorace_scanmatch-e1536675342546.png

Feature Extraction

- The plan:
  - Extract features from two scans
  - Match features
  - Compute rigid-body transformation

- Possible features:
  - Lines
  - Corners
  - Occlusion boundaries/depth discontinuities
  - And more

Line Extraction

- Given laser scan, produce a set of 2D line segments
- Two basic approaches:
  - Divisive (“Split N Fit”)
  - Agglomerative
- The tasks at hand:
  - If we know which points belong to a line, how do we fit a line to those points?
  - How do we determine which points belong together?

Optimal Line Derivation

- Assume we know a set of points belong to a line. How do we compute parameters of the line?

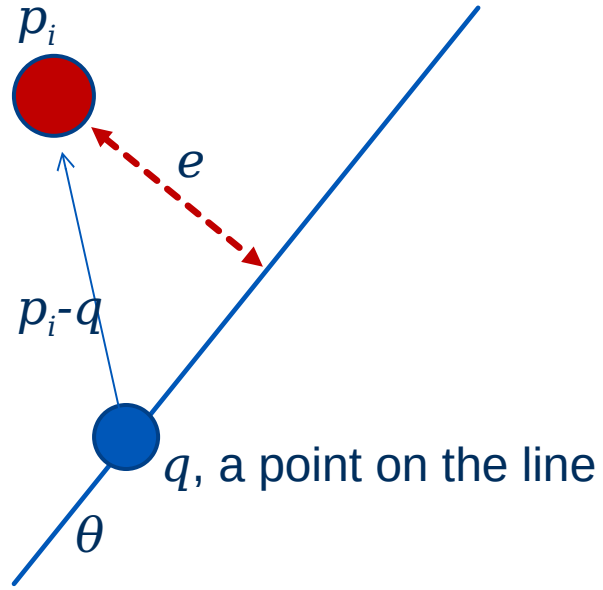
- How to parameterize the line?
  - Slope + y intercept?
  - Endpoints of line
  - A point on the line + unit vector
  - Perpendicular distance + orientation

Optimal Line Derivation

- Error for one point

$$|e| = (p_i - q) \cdot \hat{n}$$

$$\hat{n} = \begin{bmatrix} -sin(\theta) \\ cos(\theta) \end{bmatrix}$$



- Cost function
  - Solution found by minimizing

$$err^2 = \sum_i ((p_i - q) \cdot \hat{n})^2$$

Optimal Line Derivation

- Method: work in terms of moments:

$$err^2 = \sum_i \left((p_{i_x} - q_x)\hat{n}_x + (p_{i_y} - q_y)\hat{n}_y \right)^2$$

$$M_x = \sum p_x$$

$$M_y = \sum p_y$$

$$M_{xx} = \sum p_x^2$$

$$M_{xy} = \sum p_x p_y$$

$$M_{yy} = \sum p_y^2$$

$$C_{xx} = \frac{1}{N} M_{xx} - \left(\frac{M_x}{N} \right)^2$$

$$C_{xy} = \frac{1}{N} M_{xy} - \frac{M_x}{N} \frac{M_y}{N}$$

$$C_{yy} = \frac{1}{N} M_{yy} - \left(\frac{M_y}{N} \right)^2$$

Optimal Line Derivation

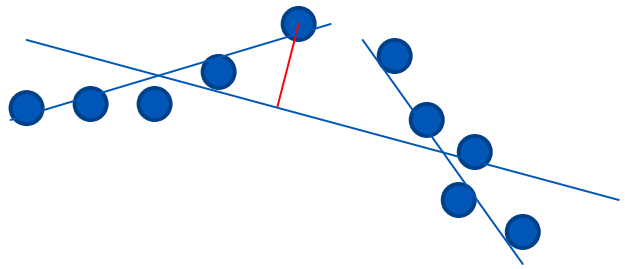
- Solution:

$$
\begin{aligned}
q_x &= \frac{1}{N} M_x \\
q_y &= \frac{1}{N} M_y \\
\theta &= \frac{\pi}{2} + \frac{1}{2} \text{atan2}(-2C_{xy}, C_{yy} - C_{xx})
\end{aligned}
$$

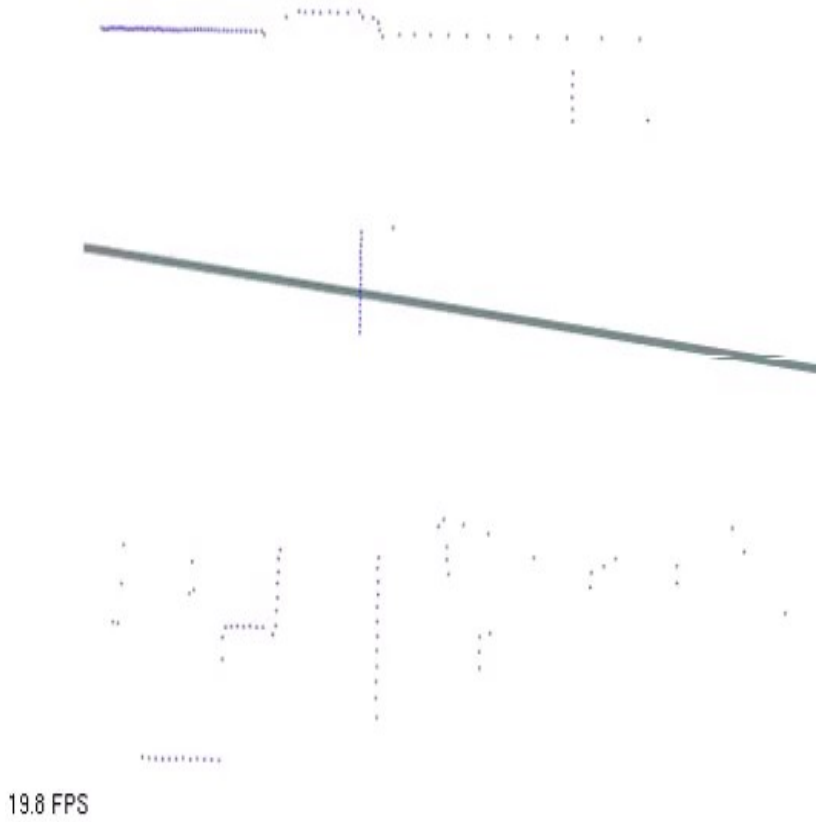
- *Because all quantities are written in terms of moments, we can compute this quantity incrementally and without storing all the points!*

Line Fitting: Divisive

- Init: All points belong to a single line
- Split-N-Fit(points)
  - Fit line to points
  - **if** line error < thresh **then**
    - **return** the line
  - **else**
    - Which point fits the worst?
    - Split the line into two lines at that point
    - **return** Split-N-Fit(leftpoints) U Split-N-Fit(rightpoints)

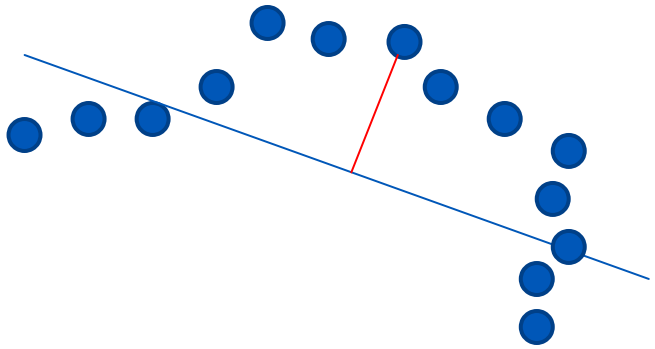


Line Fitting: Divisive



Line Fitting: Divisive

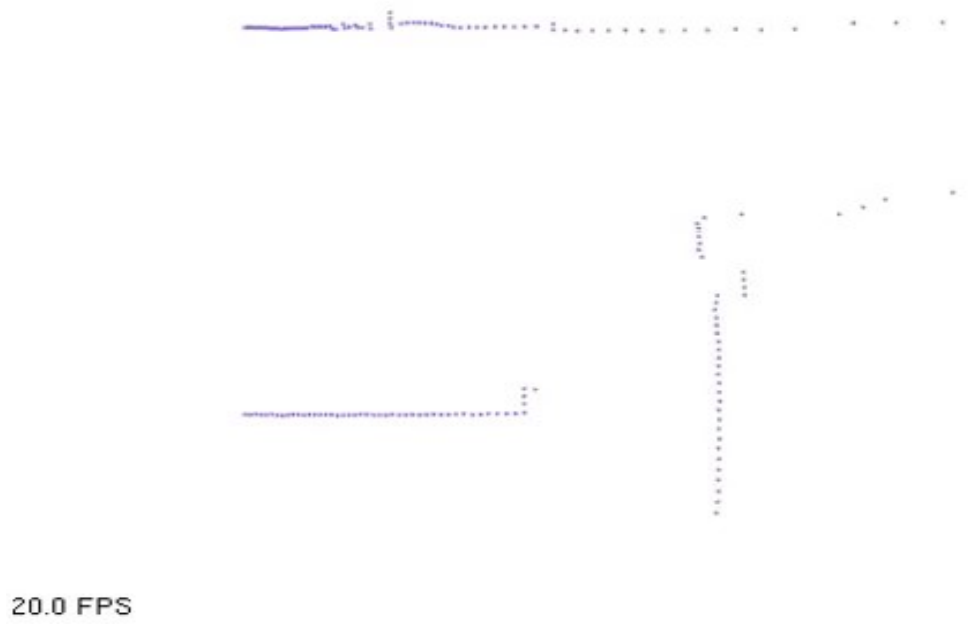
- Pros:
  - Easy to implement
- Cons:
  - Assumes points are in scan order
  - Doesn't always generate good answers
    - Can split points that should belong together
    - "Split-N-Fit-N-Merge"



Line Fitting: Agglomerative

- Agglomerative-Fit(points)
  - Init: create N-1 lines for each pair of adjacent points
  - **do forever**
    - **for** each pair of adjacent lines i, i+1
      - Compute error for line that merges those two lines
      - **if** $\text{minimum error} > \text{thresh}$ **then**
        - **return** lines
      - **else**
        - merge lines with minimum error

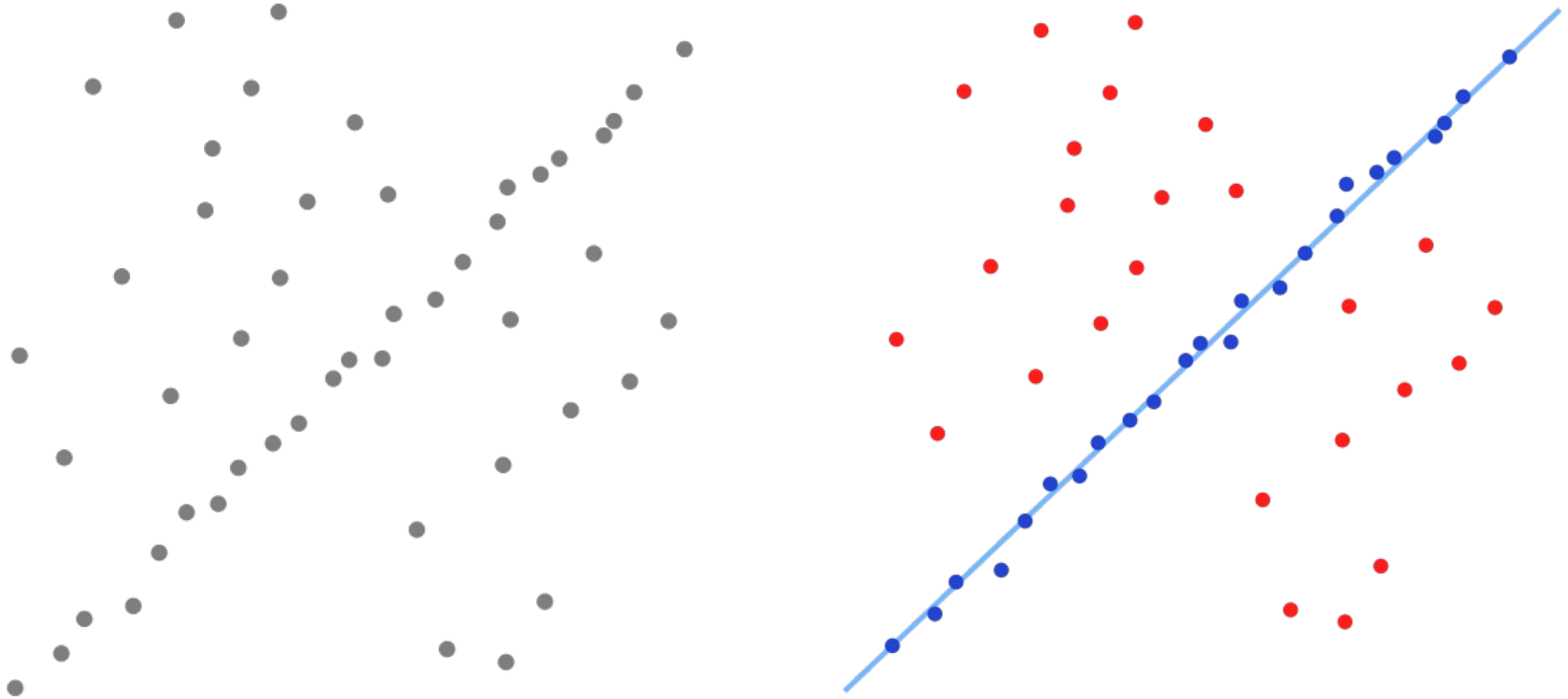
Line Fitting: Agglomerative



Line Fitting: Agglomerative

- Pros:
  - Good quality, matches human expectations
- Cons:
  - Assumes points are in scan order

RANSAC: Line Fitting



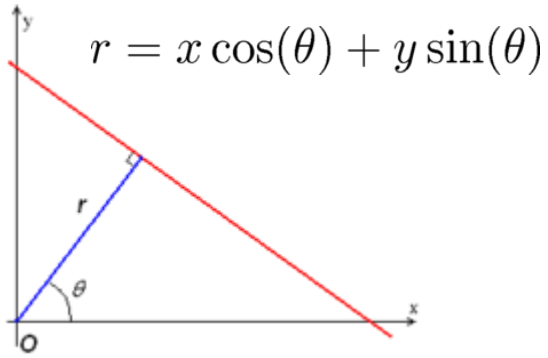
Line Fitting: RANSAC

- Pros:
  - Easy
  - Does not require points to be in scan order

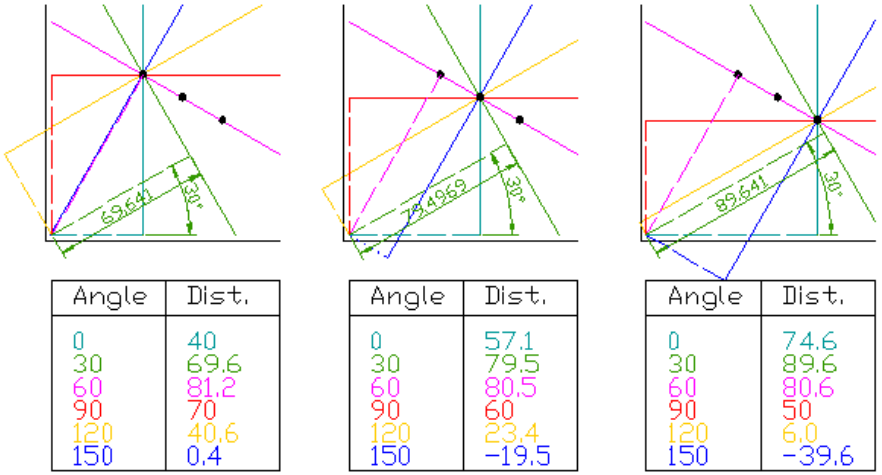
- Cons:
  - Hard to exploit points being in scan order
  - Not obvious how to extract more than one line

Line Fitting: Hough Transform

- r-theta line parameterization



- Let every point vote for all lines passing through it



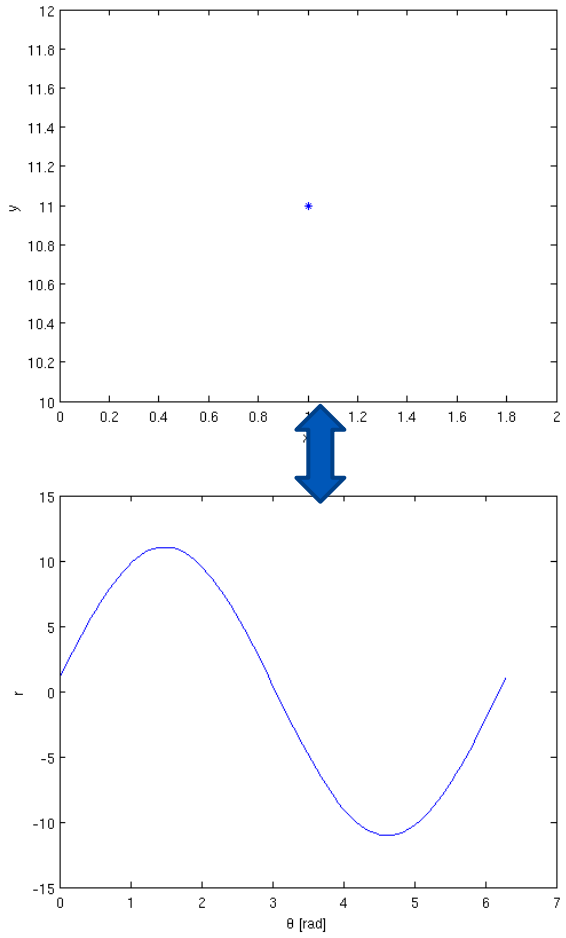
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

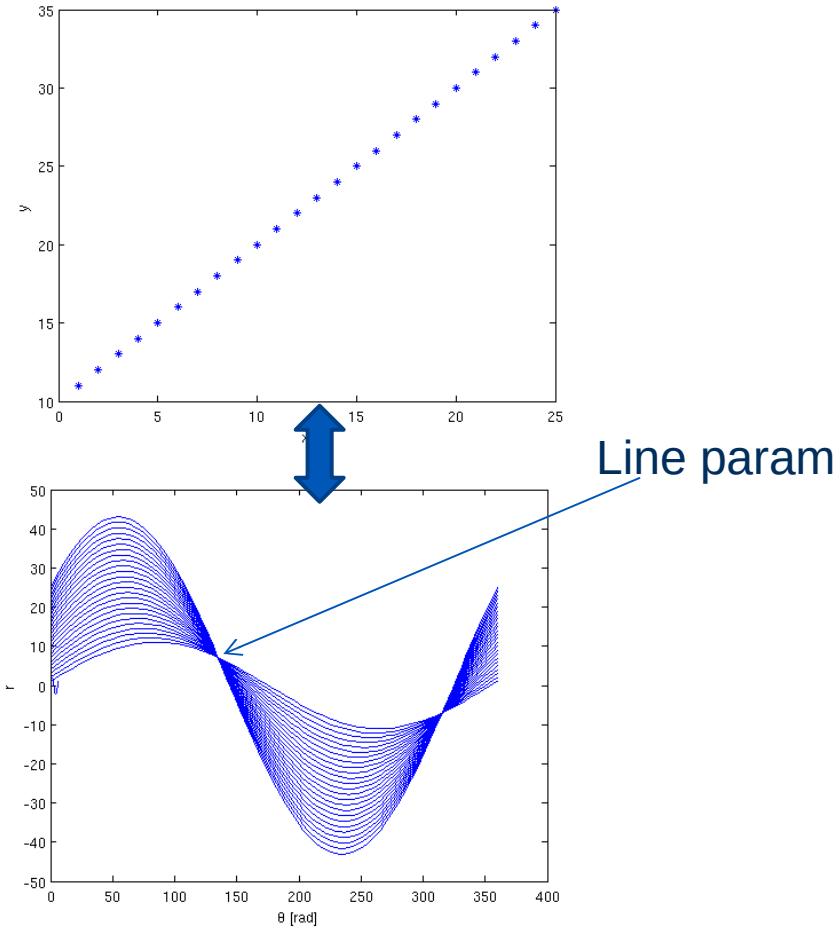
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

Line Fitting: Hough Transform

- Single-point mapping

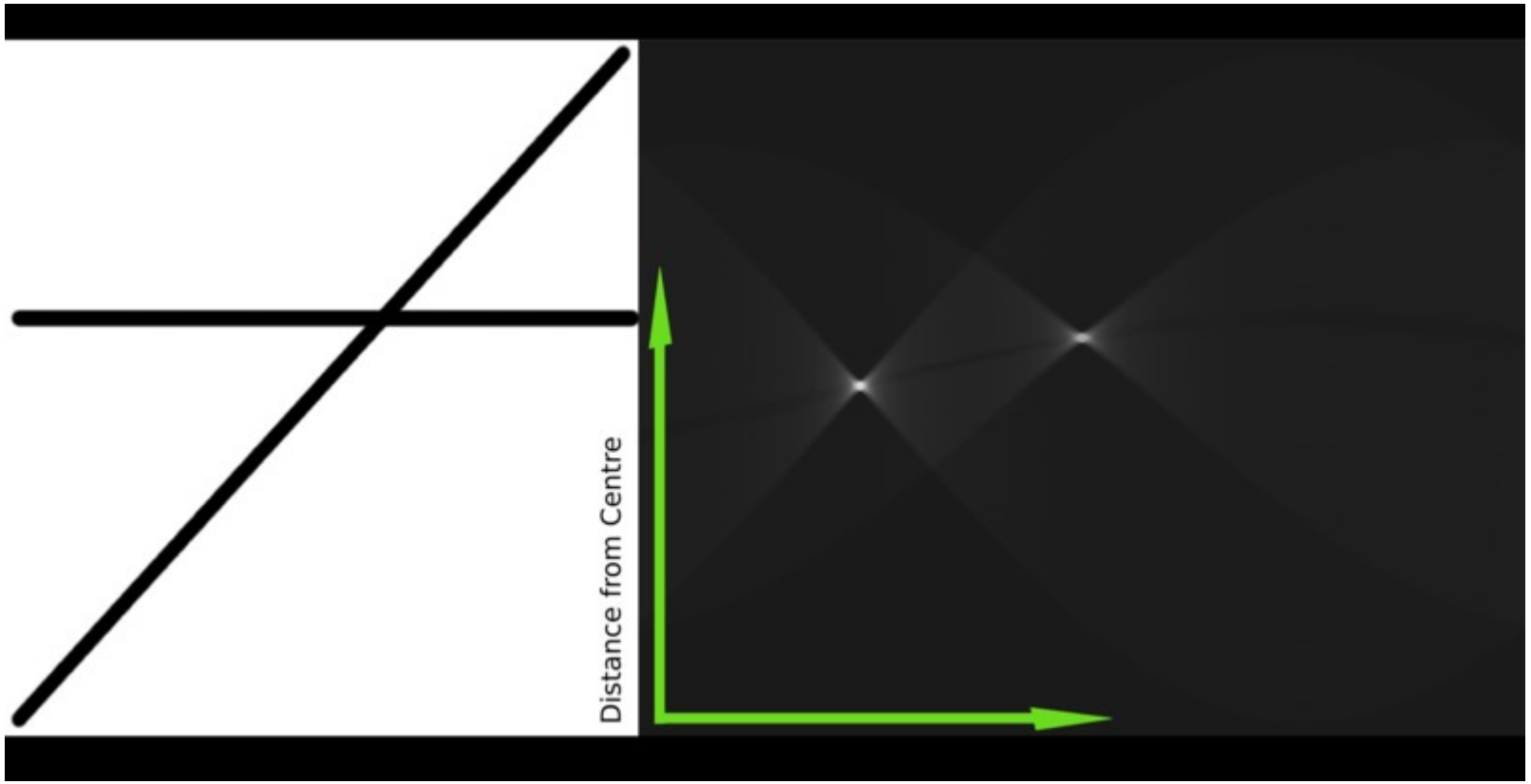


- 25-point mapping



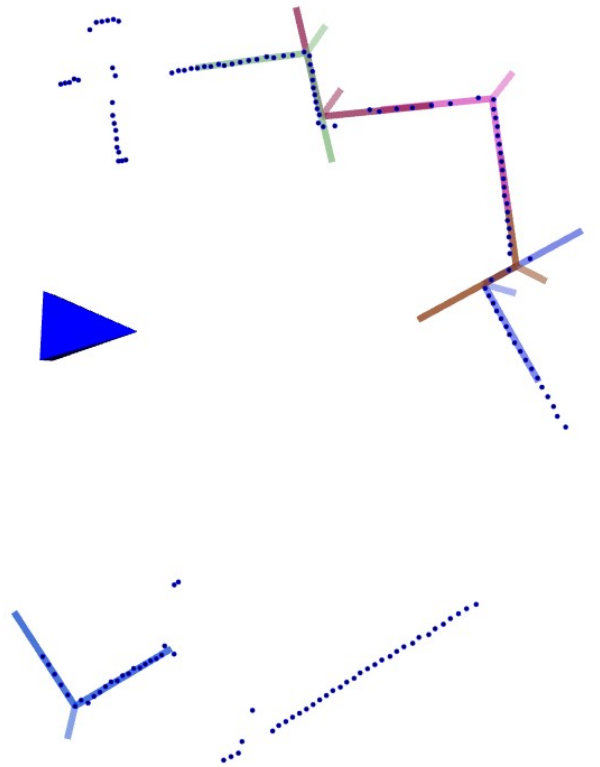
Line Fitting: Hough Transform

- Each point is evidence for a set of lines
  - Vote for each of the possible lines.
- Lines with lots of evidence get many votes.



Other Features

- Corners
  - Intersections of nearby (?) lines.
  - Easy to extract from lines.
  - Only need ONE corner correspondence to compute RBT.
- Trees (Victoria Park)
- Depth Discontinuities
  - Beware of viewpoint effects



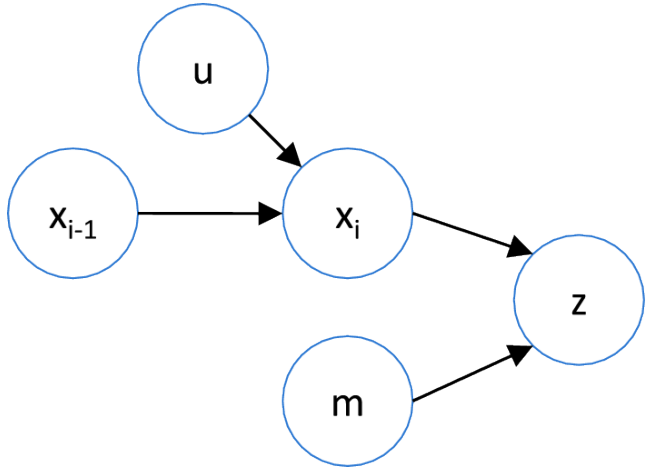
Aligning scans without features

- Correlation-based scan-matching
- ICP/ICL
- Matching scans without features
  - How well do the scans “line up”?
  - Find the RBT so that the two scans “line up” as well as possible.

Correlation-based scan matching

E. Olson, **Real-Time Correlative Scan Matching;** Proc. ICRA, 2009

- Probabilistically motivated



- We want:

$$p(x_i|x_{i-1}, u, z, m)$$

- Bayes rule, remove irrelevant conditionals

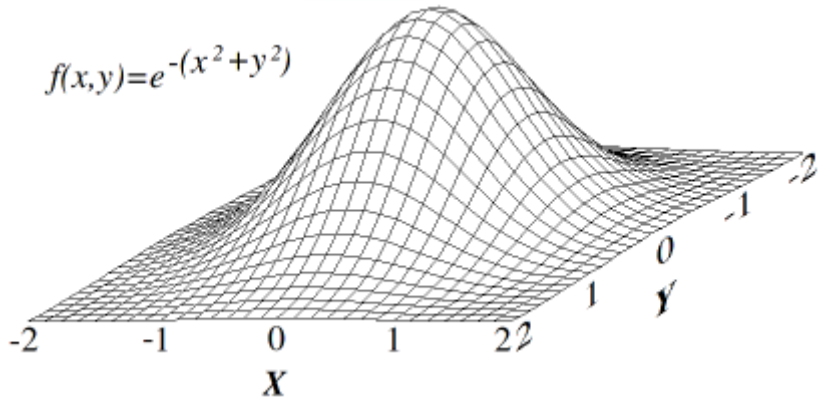
$$p(x_i|x_{i-1}, u, z, m) \propto p(z|x_i, m)p(x_i|x_{i-1}, u)$$

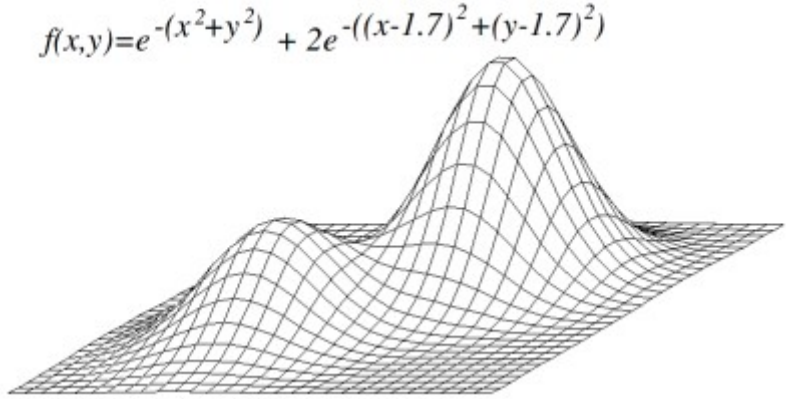
- Independent laser returns

$$p(z|x_i, m) = \prod_j p(z_j|x_i, m)$$

Correlation-based scan matching

- Which RBT is most likely?
- Local search (“hill climbing”)
  - Fast but not very robust
- “Exhaustive” search
  - Slow but super robust



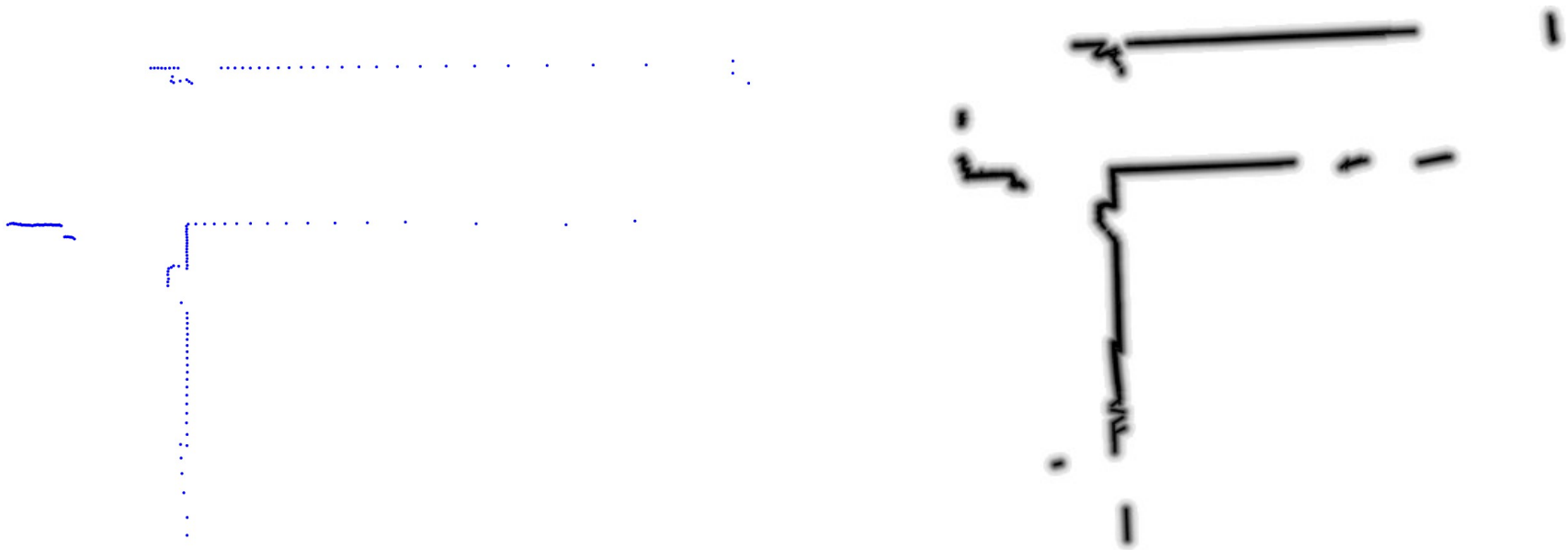


Correlation-based scan matching

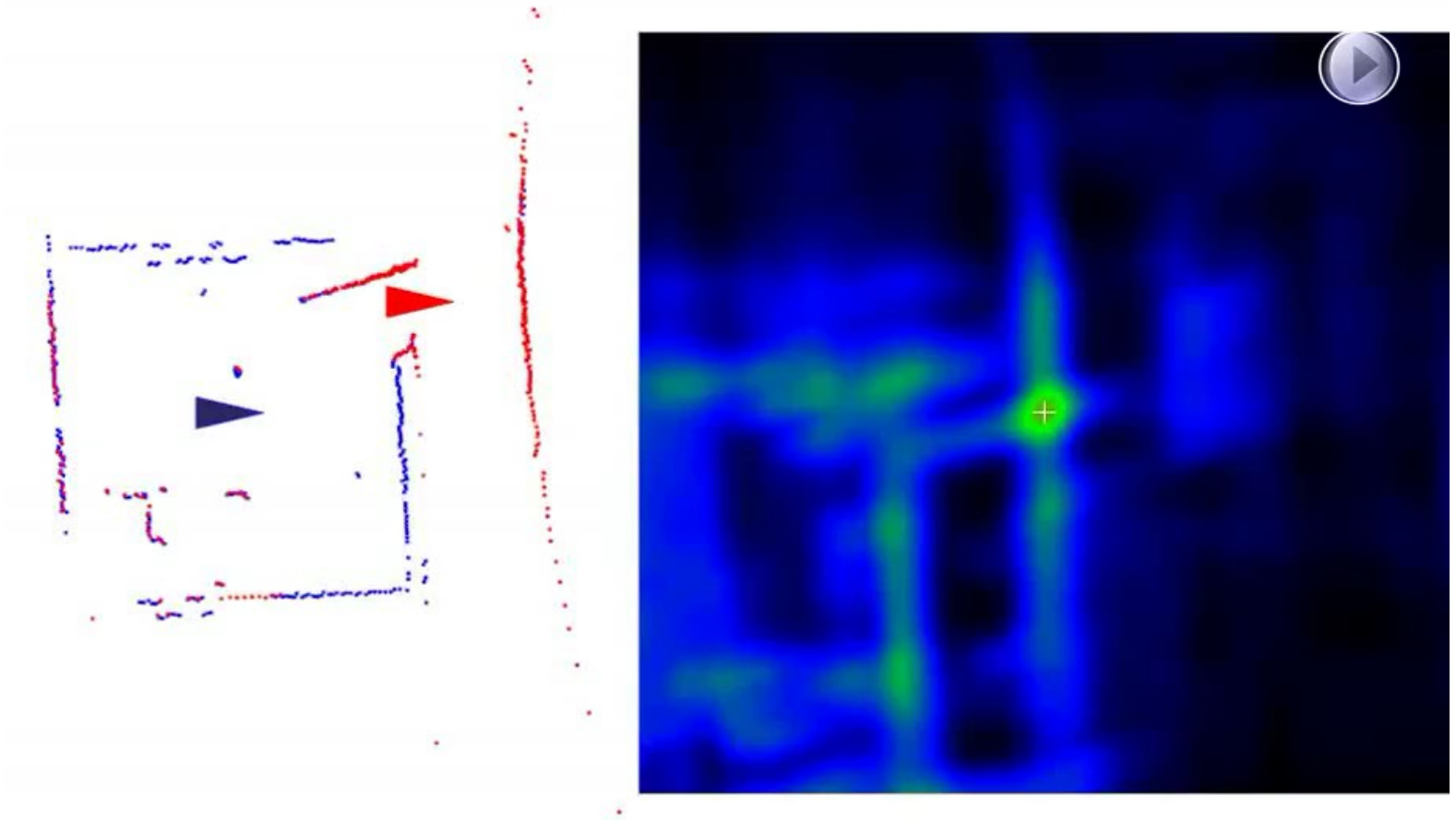
- scanmatch(pointsa, pointsb)
  - for T: all candidate RBTs
    - pointsaT = transform pointsa by T
    - logprob = 0
    - for each point p_a in pointsaT
      - logprob += log(probability of observing p_a | pointsb)
  - return T with maximum logprob.

Faster scan-matching: lookup table

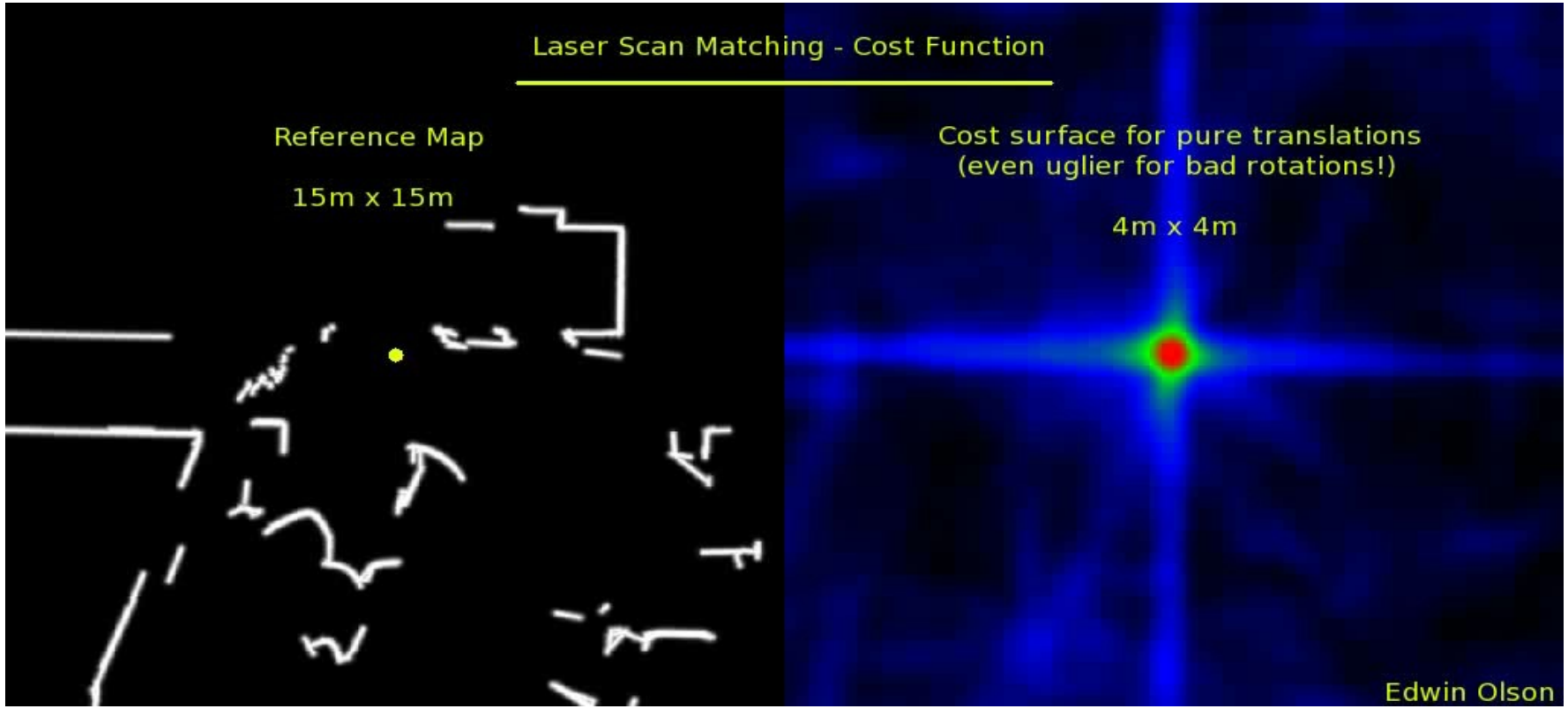
- Pre-compute log probability of observing a point
- Ignore visibility constraints



Correlation-based matching

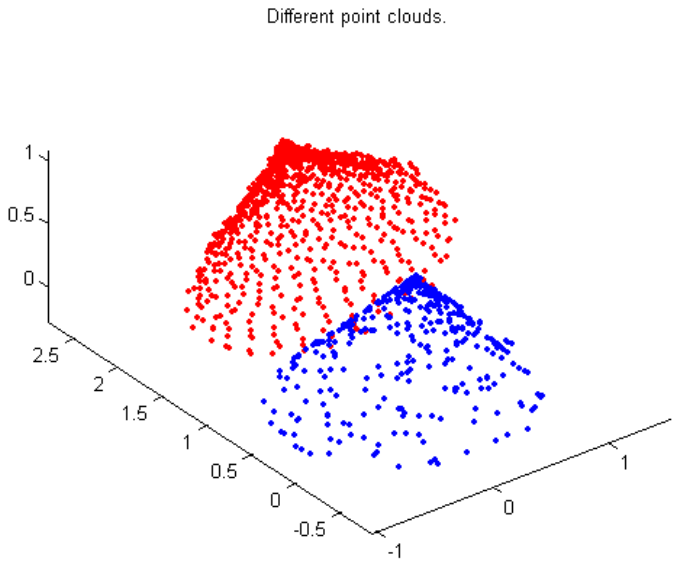


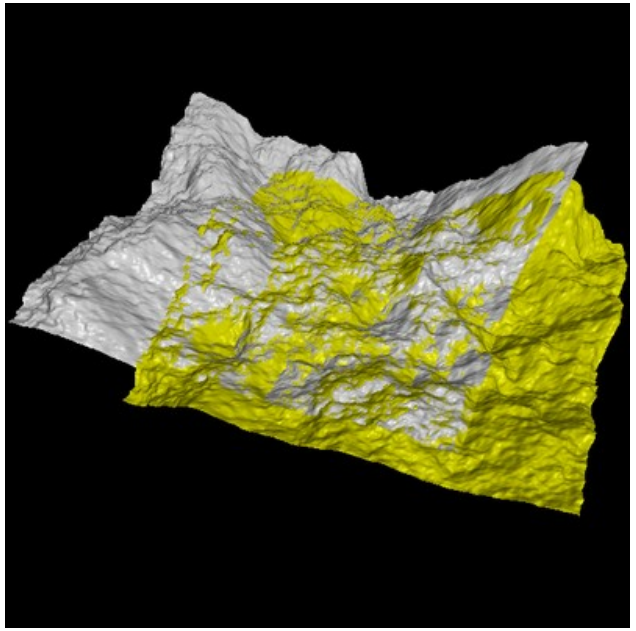
Incremental scan matching



Point based Rigid Registration

- Align two partially overlapping point clouds / meshes given initial guess for relative transform.





Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.
P. J. Besl and N. D. McKay, “A method for registration of 3D shapes,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, pp. 239–255, 1992.
Z. Zhang, “Iterative point matching for registration of free-form curves and surfaces,” *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

Problem Definition

- Given the coordinates of a set of points measured in two Cartesian coordinate systems (left, right) find the rigid transformation, T , between the two systems so that for corresponding homogenous points $\tilde{\mathbf{p}}_r$, $\tilde{\mathbf{p}}_l$ we have:

$$\tilde{\mathbf{p}}_r = T\tilde{\mathbf{p}}_l$$

1. Pairing between points is known, closed-form (analytic) solutions.

2. Pairing between points is unknown, iterative solutions (require initialization and only guarantee convergence to local optimum).

Known pairing

Adapted from: Z. Yaniv

Minimal Number of Points

Given corresponding non-collinear points $\mathbf{p}_{r,1}, \mathbf{p}_{r,2}, \mathbf{p}_{r,3}$ and $\mathbf{p}_{l,1}, \mathbf{p}_{l,2}, \mathbf{p}_{l,3}$

1. Arbitrarily choose $\mathbf{p}_1$ as the origin.
2. Construct the x-axis: $$\hat{\mathbf{x}} = \frac{\mathbf{p}_2 - \mathbf{p}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|}$$
3. Construct the y-axis: $$\mathbf{y} = (\mathbf{p}_3 - \mathbf{p}_1) - [(\mathbf{p}_3 - \mathbf{p}_1) \cdot \hat{\mathbf{x}}]\hat{\mathbf{x}}$$ $$\hat{\mathbf{y}} = \frac{\mathbf{y}}{\|\mathbf{y}\|}$$
4. Construct the z-axis: $$\hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}}$$

Minimal Number of Points (cont.)

5. Construct the rotation matrices for both point sets:

$$R_t^l = [\hat{\mathbf{x}}_l, \hat{\mathbf{y}}_l, \hat{\mathbf{z}}_l] \quad R_t^r = [\hat{\mathbf{x}}_r, \hat{\mathbf{y}}_r, \hat{\mathbf{z}}_r]$$

where subscript t denotes the triad coordinate system.

6. Construct the rotation matrix between coordinate systems:

$$R = R_l^r = R_t^r R_t^{l\top}$$

7. Construct the translation vector between coordinate systems:

$$\mathbf{t} = \mathbf{t}_{rl}^r = \mathbf{p}_{r,1} - R\mathbf{p}_{l,1}$$

Least-Squares Solution

- Analytic least squares solutions have been discovered again and again.
- The main difference between methods is the choice of rotation representation, in practice doesn't seem to really matter [Eggert et al. 1997].
- The two most popular rotation representations are:

1. Rotation matrix ([Schönemann 1966], [Arun et al. 1987], [Horn et al. 1988], [Umeyama 1991]).
2. Unit quaternion ([Faugeras and Hebert 1986], **[Horn 1987]**.

Least-Squares Solution (1)

- Given two points $\mathbf{p}_{l,i},\ \mathbf{p}_{r,i}$ and the transformation $[R, \mathbf{t}]$, the residual error is:

$$\mathbf{e}_i = \mathbf{p}_{r,i} - R\mathbf{p}_{l,i} - \mathbf{t}$$

- We want to minimize the sum of squared errors:

$$\sum_{i=1}^{n} \|\mathbf{e}_i\|^2$$

Least-Squares Solution (2)

- Refer all points to a coordinate system relative to the centroid:

$$\mathbf{p}'_{l,i} = \mathbf{p}_{l,i} - \boldsymbol{\mu}_l \qquad \mathbf{p}'_{r,i} = \mathbf{p}_{r,i} - \boldsymbol{\mu}_r$$

where

$$\boldsymbol{\mu}_l = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_{l,i} \qquad \boldsymbol{\mu}_r = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_{r,i}$$

- and rewrite the error term:

$$\mathbf{e}_i = \mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i} - \mathbf{t}'$$

where

$$\mathbf{t}' = \mathbf{t} - \boldsymbol{\mu}_r + R\boldsymbol{\mu}_l$$

Least-Squares Solution (3)

$$\sum_{i=1}^{n} \|\mathbf{e}_i\|^2 = \sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i} - \mathbf{t}'\|^2$$

$$= \sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}\|^2 - 2\mathbf{t}' \cdot \sum_{i=1}^{n} (\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}) + n\|\mathbf{t}'\|^2$$

Least-Squares Solution (3)

$$\sum_{i=1}^{n} \|\mathbf{e}_i\|^2 = \sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i} - \mathbf{t}'\|^2$$

$$= \sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}\|^2 - 2\mathbf{t}' \cdot \sum_{i=1}^{n} (\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}) + n\|\mathbf{t}'\|^2$$

This term is equal to zero as

$$\sum_{i=1}^{n} \mathbf{p}'_{r,i} = \sum_{i=1}^{n} \mathbf{p}'_{l,i} = \mathbf{0}$$

Least-Squares Solution (3)

$$\sum_{i=1}^{n} \|\mathbf{e}_i\|^2 = \sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i} - \mathbf{t}'\|^2$$

$$= \sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}\|^2 - 2\mathbf{t}' \cdot \sum_{i=1}^{n} (\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}) + n\|\mathbf{t}'\|^2$$

This term is dependent only on $\mathbf{t}'$, and is non negative.

$$\mathbf{t}' = \mathbf{0}$$

$$\Downarrow$$

$$\boxed{\mathbf{t} = \boldsymbol{\mu}_r - R\boldsymbol{\mu}_l}$$

Least-Squares Solution (3)

$$\sum_{i=1}^{n} \|\mathbf{e}_i\|^2 = \sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i} - \mathbf{t}'\|^2$$

$$= \underbrace{\sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}\|^2}_{} - 2\mathbf{t}' \cdot \sum_{i=1}^{n} (\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}) + n\|\mathbf{t}'\|^2$$

Minimization is reduced to minimizing this term with respect to the rotation operator, which will define the optimal translation.

Least-Squares Solution (4)

$$\sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}\|^2 = \underbrace{\sum_{i=1}^{n} \|\mathbf{p}'_{r,i}\|^2}_{\text{Constant not dependent on rotation.}} - 2\sum_{i=1}^{n} \mathbf{p}'_{r,i} \cdot R\mathbf{p}'_{l,i} + \sum_{i=1}^{n} \|R\mathbf{p}'_{l,i}\|^2$$

Least-Squares Solution (4)

$$\sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}\|^2 = \sum_{i=1}^{n} \|\mathbf{p}'_{r,i}\|^2 - 2\sum_{i=1}^{n} \mathbf{p}'_{r,i} \cdot R\mathbf{p}'_{l,i} + \underbrace{\sum_{i=1}^{n} \|R\mathbf{p}'_{l,i}\|^2}$$

Constant not dependent on rotation.

Least-Squares Solution (4)

$$\sum_{i=1}^{n} \|\mathbf{p}'_{r,i} - R\mathbf{p}'_{l,i}\|^2 = \sum_{i=1}^{n} \|\mathbf{p}'_{r,i}\|^2 - 2\sum_{i=1}^{n} \mathbf{p}'_{r,i} \cdot R\mathbf{p}'_{l,i} + \sum_{i=1}^{n} \|R\mathbf{p}'_{l,i}\|^2$$

Maximizing this term, minimizes the error

Up to this point we have not specified the rotation operator. Horn chose the unit quaternion as the rotation operator.

Quaternions (1)

- Quaternions are mathematical objects of the form:
  $\mathring{\mathbf{q}} = q_0 + iq_x + jq_y + kq_z = [s, \mathbf{v}]$ where $q_0, q_x, q_y, q_z \in \mathbb{R}$, and i, j, k are mutually orthogonal imaginary units:

	i	j	k
i	-1	k	-j
j	-k	-1	i
k	j	-i	-1

- Conjugate: $\mathring{\mathbf{q}}^* = [s, -\mathbf{v}]$
- Dot Product: $\mathring{\mathbf{p}} \cdot \mathring{\mathbf{q}} = p_0 q_0 + p_x q_x + p_y q_y + p_z q_z$
- Norm: $\|\mathring{\mathbf{q}}\| = \sqrt{\mathring{\mathbf{q}} \cdot \mathring{\mathbf{q}}} = \sqrt{\mathring{\mathbf{q}}\mathring{\mathbf{q}}^*} = \sqrt{s^2 + \|\mathbf{v}\|^2} = \sqrt{q_0^2 + q_x^2 + q_y^2 + q_z^2}$
- $\mathring{\mathbf{q}}^{-1} = (\frac{1}{\mathring{\mathbf{q}} \cdot \mathring{\mathbf{q}}})\mathring{\mathbf{q}}^*$ $\|\mathring{\mathbf{q}}\| = 1$ $\mathring{\mathbf{q}}^{-1} = \mathring{\mathbf{q}}^*$
- Computer

Quaternions (2)

- Addition: $\mathring{\mathbf{q}} = \mathring{\mathbf{q}}_1 \pm \mathring{\mathbf{q}}_2 = [s_1 \pm s_2, \mathbf{v}_1 \pm \mathbf{v}_2]$

- Multiplication: $\mathring{\mathbf{q}} = \mathring{\mathbf{q}}_1 \mathring{\mathbf{q}}_2 = [(s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2), (s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)]$

$$\begin{aligned}\mathring{\mathbf{p}}\mathring{\mathbf{q}} = &(p_0 q_0 - p_x q_x - p_y q_y - p_z q_z) \\ &+ i(p_0 q_x + p_x q_0 + p_y q_z - p_z q_y) \\ &+ j(p_0 q_y - p_x q_z + p_y q_0 + p_z q_x) \\ &+ k(p_0 q_z + p_x q_y - p_y q_x + p_z q_0)\end{aligned}$$

$$\begin{aligned}\mathring{\mathbf{q}}\mathring{\mathbf{p}} = &(q_0 p_0 - q_x p_x - q_y p_y - q_z p_z) \\ &+ i(q_0 p_x + q_x p_0 + q_y p_z - q_z p_y) \\ &+ j(q_0 p_y - q_x p_z + q_y p_0 + q_z p_x) \\ &+ k(q_0 p_z + q_x p_y - q_y p_x + q_z p_0)\end{aligned}$$

Quaternions (2)

- Addition: $\mathring{\mathbf{q}} = \mathring{\mathbf{q}}_1 \pm \mathring{\mathbf{q}}_2 = [s_1 \pm s_2, \mathbf{v}_1 \pm \mathbf{v}_2]$

- Multiplication: (as an orthogonal $4£4$ matrix)

$$\mathring{\mathbf{q}} = \mathring{\mathbf{q}}_1 \mathring{\mathbf{q}}_2 = [(s_1 s_2 - \mathbf{v}_1 \cdot \mathbf{v}_2), (s_1 \mathbf{v}_2 + s_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)]$$

$$\mathring{\mathbf{p}}\mathring{\mathbf{q}} = \begin{bmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & -p_z & p_y \\ p_y & p_z & p_0 & -p_x \\ p_z & -p_y & p_x & p_0 \end{bmatrix} \mathring{\mathbf{q}} = P\mathring{\mathbf{q}}, \qquad \mathring{\mathbf{q}}\mathring{\mathbf{p}} = \begin{bmatrix} p_0 & -p_x & -p_y & -p_z \\ p_x & p_0 & p_z & -p_y \\ p_y & -p_z & p_0 & p_x \\ p_z & p_y & -p_x & p_0 \end{bmatrix} \mathring{\mathbf{q}} = \bar{P}\mathring{\mathbf{q}}$$

- Matrices associated with the conjugate quaternion are just the transposes of the above:

$$\mathring{\mathbf{p}}^*\mathring{\mathbf{q}} = P^\top \mathring{\mathbf{q}} \qquad \mathring{\mathbf{q}}\mathring{\mathbf{p}}^* = \bar{P}^\top \mathring{\mathbf{q}}$$

Example

- Prove that

$$\mathring{\mathbf{p}} \cdot (\mathring{\mathbf{r}}\mathring{\mathbf{q}}^*) = (\mathring{\mathbf{p}}\mathring{\mathbf{q}}) \cdot \mathring{\mathbf{r}}$$

Quaternions (3)

- A unit quaternion can represent a rotation by θ radians around an axis $\hat{\mathbf{n}}$ as $\mathring{\mathbf{q}} = [\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{\mathbf{n}}]$

- Vectors can be represented by purely imaginary quaternions. If $\mathbf{r} = (x, y, z)^\top$ then we can use

$$\mathring{\mathbf{r}} = [0, \mathbf{r}] = 0 + ix + jy + kz$$

- Rotating a given vector using the unit quaternion:

$$\mathbf{r}' = R\mathbf{r} \iff \mathring{\mathbf{r}}' = \mathring{\mathbf{q}}\mathring{\mathbf{r}}\mathring{\mathbf{q}}^*$$

Note $(-\mathring{\mathbf{q}})\mathring{\mathbf{r}}(-\mathring{\mathbf{q}}^*) = \mathring{\mathbf{q}}\mathring{\mathbf{r}}\mathring{\mathbf{q}}^*$, so $\mathring{\mathbf{q}}$ and $-\mathring{\mathbf{q}}$ describe the same rotation.

Quaternions (3)

- Rotating a given vector using the unit quaternion:

$$\mathbf{r}' = R\mathbf{r} \iff \mathring{\mathbf{r}}' = \mathring{\mathbf{q}}\mathring{\mathbf{r}}\mathring{\mathbf{q}}^*$$

$$\mathring{\mathbf{q}}\mathring{\mathbf{r}}\mathring{\mathbf{q}}^* = (Q\mathring{\mathbf{r}})\mathring{\mathbf{q}}^* = \bar{Q}^\top(Q\mathring{\mathbf{r}}) = (\bar{Q}^\top Q)\mathring{\mathbf{r}}$$

$$\bar{Q}^\top Q = \begin{bmatrix} \mathring{\mathbf{q}}\cdot\mathring{\mathbf{q}} & 0 & 0 & 0 \\ 0 & (q_0^2+q_x^2-q_y^2-q_z^2) & 2(q_xq_y-q_0q_z) & 2(q_xq_z+q_0q_y) \\ 0 & 2(q_yq_x+q_0q_z) & (q_0^2-q_x^2+q_y^2-q_z^2) & 2(q_yq_z-q_0q_x) \\ 0 & 2(q_zq_x-q_0q_y) & 2(q_zq_y+q_0q_x) & (q_0^2-q_x^2-q_y^2+q_z^2) \end{bmatrix}$$

R

- so $\mathring{\mathbf{r}}'$ is purely imaginary if $\mathring{\mathbf{r}}$ is.

Least-Squares Solution (5)

We want to maximize:

$$\sum_{i=1}^{n} \mathbf{p}'_{r,i} \cdot R\mathbf{p}'_{l,i} = \sum_{i=1}^{n} (\mathring{\mathbf{q}}\mathring{\mathbf{p}}'_{l,i}\mathring{\mathbf{q}}^{*}) \cdot \mathring{\mathbf{p}}'_{r,i} = \sum_{i=1}^{n} (\mathring{\mathbf{q}}\mathring{\mathbf{p}}'_{l,i}) \cdot (\mathring{\mathbf{p}}'_{r,i}\mathring{\mathbf{q}})$$

In matrix form we get $\sum_{i=1}^{n} (\bar{P}_{l,i}\mathring{\mathbf{q}}) \cdot (P_{r,i}\mathring{\mathbf{q}})$ where

$$\mathring{\mathbf{q}}\mathring{\mathbf{p}}'_{l,i} = \begin{bmatrix} 0 & -x'_{l,i} & -y'_{l,i} & -z'_{l,i} \\ x'_{l,i} & 0 & z'_{l,i} & -y'_{l,i} \\ y'_{l,i} & -z'_{l,i} & 0 & x'_{l,i} \\ z'_{l,i} & y'_{l,i} & -x'_{l,i} & 0 \end{bmatrix} \mathring{\mathbf{q}} = \bar{P}_{l,i}\mathring{\mathbf{q}}$$

$$\mathring{\mathbf{p}}'_{r,i}\mathring{\mathbf{q}} = \begin{bmatrix} 0 & -x'_{r,i} & -y'_{r,i} & -z'_{r,i} \\ x'_{r,i} & 0 & -z'_{r,i} & y'_{r,i} \\ y'_{r,i} & z'_{r,i} & 0 & -x'_{r,i} \\ z'_{r,i} & -y'_{r,i} & x'_{r,i} & 0 \end{bmatrix} \mathring{\mathbf{q}} = P_{r,i}\mathring{\mathbf{q}}$$

Least-Squares Solution (6)

- With a bit of manipulation we get:

$$\sum_{i=1}^{n}(\bar{P}_{l,i}\mathring{\mathbf{q}})\cdot(P_{r,i}\mathring{\mathbf{q}}) \Rightarrow \sum_{i=1}^{n}\mathring{\mathbf{q}}^{\top}\bar{P}_{l,i}^{\top}P_{r,i}\mathring{\mathbf{q}} \Rightarrow$$

$$\mathring{\mathbf{q}}^{\top}\left(\sum_{i=1}^{n}\bar{P}_{l,i}^{\top}P_{r,i}\right)\mathring{\mathbf{q}} \Rightarrow \mathring{\mathbf{q}}^{\top}(\sum_{i=1}^{n}N_i)\mathring{\mathbf{q}} \Rightarrow \mathring{\mathbf{q}}^{\top}N\mathring{\mathbf{q}}$$

where N is a $4\text{£}4$ symmetric matrix (sum of symmetric matrices).

$$N=\begin{bmatrix}(S_{xx}+S_{yy}+S_{zz}) & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx}\\ S_{yz}-S_{zy} & (S_{xx}-S_{yy}-S_{zz}) & S_{xy}+S_{yx} & S_{zx}+S_{xz}\\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & (-S_{xx}+S_{yy}-S_{zz}) & S_{yz}+S_{zy}\\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & (-S_{xx}-S_{yy}+S_{zz})\end{bmatrix}$$

Where $S_{xx}=\sum_{i=1}^{n}x'_{l,i}x'_{r,i}$,$S_{xy}=\sum_{i=1}^{n}x'_{l,i}y'_{r,i}$, etc

Least-Squares Solution (6)

- With a bit of manipulation we get:

$$\sum_{i=1}^{n}(\bar{P}_{l,i}\mathring{\mathbf{q}}) \cdot (P_{r,i}\mathring{\mathbf{q}}) \Rightarrow \sum_{i=1}^{n}\mathring{\mathbf{q}}^{\top}\bar{P}_{l,i}^{\top}P_{r,i}\mathring{\mathbf{q}} \Rightarrow$$

$$\mathring{\mathbf{q}}^{\top}\left(\sum_{i=1}^{n}\bar{P}_{l,i}^{\top}P_{r,i}\right)\mathring{\mathbf{q}} \Rightarrow \mathring{\mathbf{q}}^{\top}(\sum_{i=1}^{n}N_i)\mathring{\mathbf{q}} \Rightarrow \mathring{\mathbf{q}}^{\top}N\mathring{\mathbf{q}}$$

where N is a $4 \pounds 4$ symmetric matrix (sum of symmetric matrices).

- The expression $\mathring{\mathbf{q}}^{\top}N\mathring{\mathbf{q}}$ is maximized when $\mathring{\mathbf{q}}$ is set to the unit eigenvector corresponding to the largest eigenvalue of N.

Least-Squares Solution – Summary

1. Translate all points so that coordinates are relative to centroids.
2. Construct the matrix N and compute the unit eigenvector corresponding to the largest eigenvalue.
   - *This is the unit quaternion associated with the least-squares orientation.*
3. Compute the translation given the rotation computed in the previous stage.

$$\mathbf{t} = \boldsymbol{\mu}_r - R\boldsymbol{\mu}_l$$

Least-Squares Solution – Matlab

```
function [T] = absoluteOrientation(pointsInLeft,pointsInRight)
% Returns the 4x4 homogenous transform T
% Assumes the input points are [3xn] arrays where n is the # of points

meanLeft = mean(pointsInLeft')';
meanRight = mean(pointsInRight')';

n = size(pointsInLeft,2);
M = pointsInLeft*pointsInRight';
M = M - n*meanLeft*meanRight';

delta = [M(2,3) - M(3,2); M(3,1) - M(1,3); M(1,2) - M(2,1)];

N = [trace(M), delta'; delta, (M+M'-trace(M)*eye(3))];

[eigenVectors, eigenValues] = eig(N);
[dummy,index] = max(diag(eigenValues));

rotation = q2R(eigenVectors(:,index));
translation = meanRight - rotation*meanLeft;
T = [rotation, translation; [0, 0, 0, 1]];

%=========================================================================
function R = q2R(q)

q0 = q(1);
qx = q(2);
qy = q(3);
qz = q(4);

R = [(q0^2+qx^2-qy^2-qz^2),       2*(qx*qy-q0*qz), 2*(qx*qz+q0*qy); ...
     2*(qy*qx+q0*qz),       (q0^2-qx^2+qy^2-qz^2), 2*(qy*qz-q0*qx); ...
     2*(qz*qx-q0*qy),             2*(qz*qy+q0*qx), (q0^2-qx^2-qy^2+qz^2)];
```

Least-Squares Solution – Caveats

1. Least squares formulation assumes noise is isotropic, independent and identically distributed (i.e., ~$N(0, s^2)$).

   Solution: use iterative total least squares [Ohta and Kanatani 1998].

2. The number of outliers needed to throw our estimator outside of reasonable bounds (breakdown point) is one. Possible solutions:

   $\sum_{i=1}^{n} w_i ||e_i||^2 \quad , \quad w_i \geq 0$ -squares ([Maurer et al. 1998]), set and follow exactly the same derivation.

   RANSAC ([Fischler and Bolles 1981]).

unknown pairing

Adapted from: Z. Yaniv

Iterative Closest Point (ICP)

- Assume that the transformation is small, nearly identity.
- Given this assumption it is reasonable that the distance $\mathbf{e}_i = \mathbf{p}_{r,i} - R\mathbf{p}_{l,i} - \mathbf{t}$ is small. Match the closest point in $\{p_r\}$ to $T(p_l)$.
- As the transformation is not close to the identity the match is usually wrong.
- Overcome this by using an iterative scheme.

Iterative Closest Point (ICP)

- Iterations alternate between a matching step and a transformation computation step based on an analytic solution, hence Iterative Closest Point (ICP).

- This iterative framework was independently developed by several authors ([Chen and Medioni 1992], [Besl and McKay 1992], [Zhang 1994]).

- Euclidean distance is the most basic mechanism for establishing correspondences, other mechanisms provide better results.

Iterative **Corresponding** Point (ICP)

- Minimal distance between point descriptors replaces/ augments the minimal Euclidean distance.
- Point descriptors include: normal direction, surface curvature, texture, etc.
- The data is not necessarily two point sets (e.g., point set/surface mesh, point set/ray set).
- The cardinality (number of spatial entities) of both data sets is usually not equal, rather we have:

$$\{p\}_{i=1...n} \subset \{g\}_{i=1...m}$$

Iterative **Corresponding** Point (ICP)

Initialization:

1. Set cumulative transformation, and apply to points.
2. Pair *corresponding* points and compute similarity (e.g., root mean square distance).

Iterate:

3. Compute incremental transformation using the current correspondences (i.e., analytic least squares solution).
4. Update cumulative transformation, and apply to points.
5. Pair corresponding points and compute similarity.
6. If improvement in similarity is less than threshold t or number of iterations has reached threshold n terminate.

ICP – Establishing Correspondence

- Most ICP based methods still use Euclidean distance to establish correspondence.

- This step is the most computationally expensive step with a worst case cost of $O(MN)$ for two datasets of size M and N, respectively.

ICP – Establishing Correspondence (cont.)

- Use spatial data structures to speed up the search for nearest neighbor.
- The most popular data structure is the kd-tree (suggested in [Besl and McKay 1992]).
- Computational complexity:
  - Construction $O(N \log N)$.
  - Query $O(M \log N)$.

kd-tree

- Two versions of the kd-tree:

1. At each level i of the tree data is partitioned around the median value of coordinate ($i \bmod d$) where d is the point dimensionality.

[Friedman et al. 1977], textbooks: [de Berg et al. 1997], [Samet 1990].

2. At each level i compute the eigenvectors, eigenvalues of the covariance matrix. Partition plane is perpendicular to the direction of maximal variance (eigenvector corresponding to largest eigenvalue).

[Sproull 1991], [Williams et al. 1997], [McNames 2001].

- Second version yields more balanced trees

kd-tree: Construction

1. If cardinality of $\{P_i\}$ is less than bucket size create leaf node containing point data.

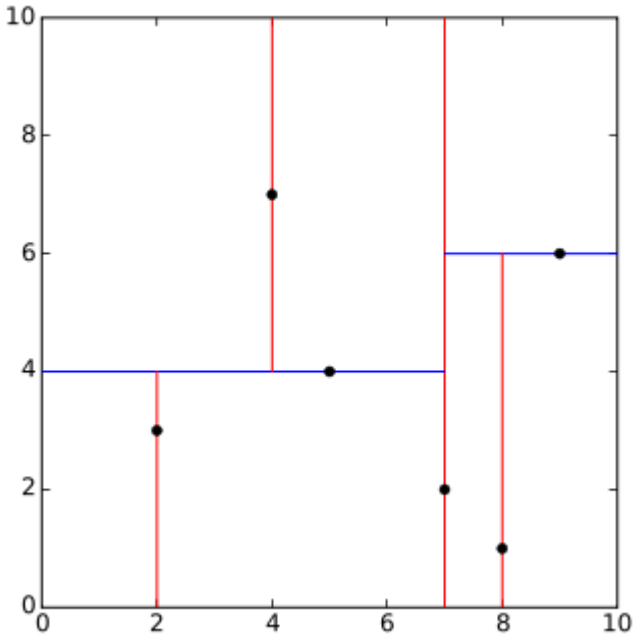
2. Else
   a) Choose key coordinate (two options):
      i. If at level i key is the $(i \bmod d)$ coordinate.
      ii. Find axis with maximal spread and choose this as the key (requires additional information held in internal tree nodes).
   b) Split data according to median of the 'key' coordinate and apply recursion with two point sets $\{P_i[key]\} \leq median$, left subtree, and $\{P_i[key]\} > median$, right subtree.

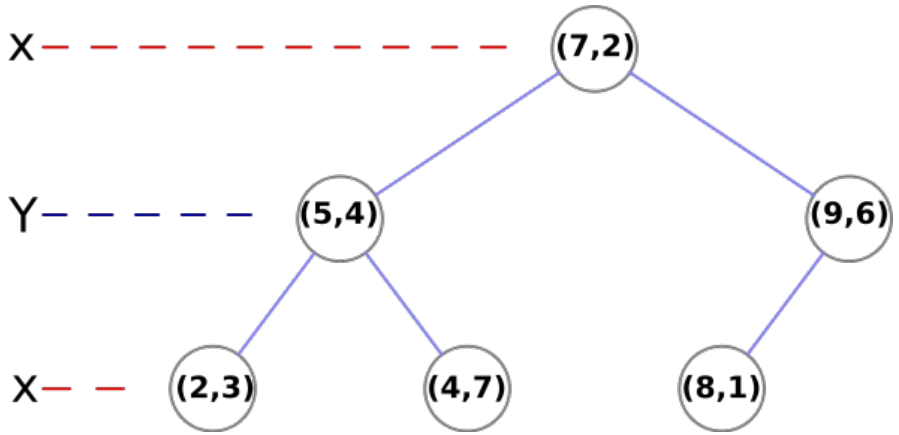
kd-tree: Query

1. If kd-tree node is a leaf perform an exhaustive search on points contained in the node.
2. Else
   a. Key is coordinate $(i \bmod d)$, where i is current level in the tree.
   b. If $point[key] > partition\ value$
      i. Recurse on right sub tree.
      ii. If $currentMinimalDistance > distance(point; partitionPlane)$ recurse on left sub tree.
   c. Else
      i. Recurse on left sub tree.
      ii. If $currentMinimalDistance > distance(point; partitionPlane)$ recurse on right sub tree.

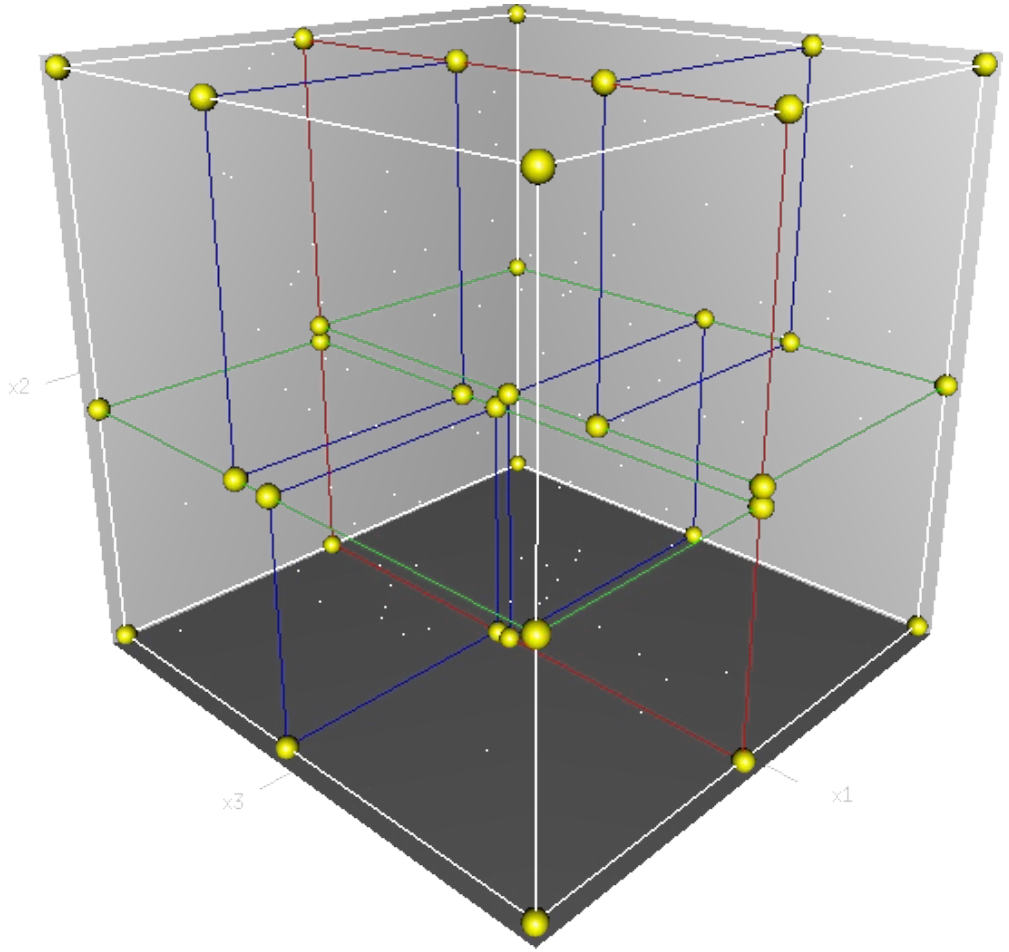
kd-trees:
To speed up closest point selection

- Let us take an example of a 2-D tree.
- Sample data [(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)]

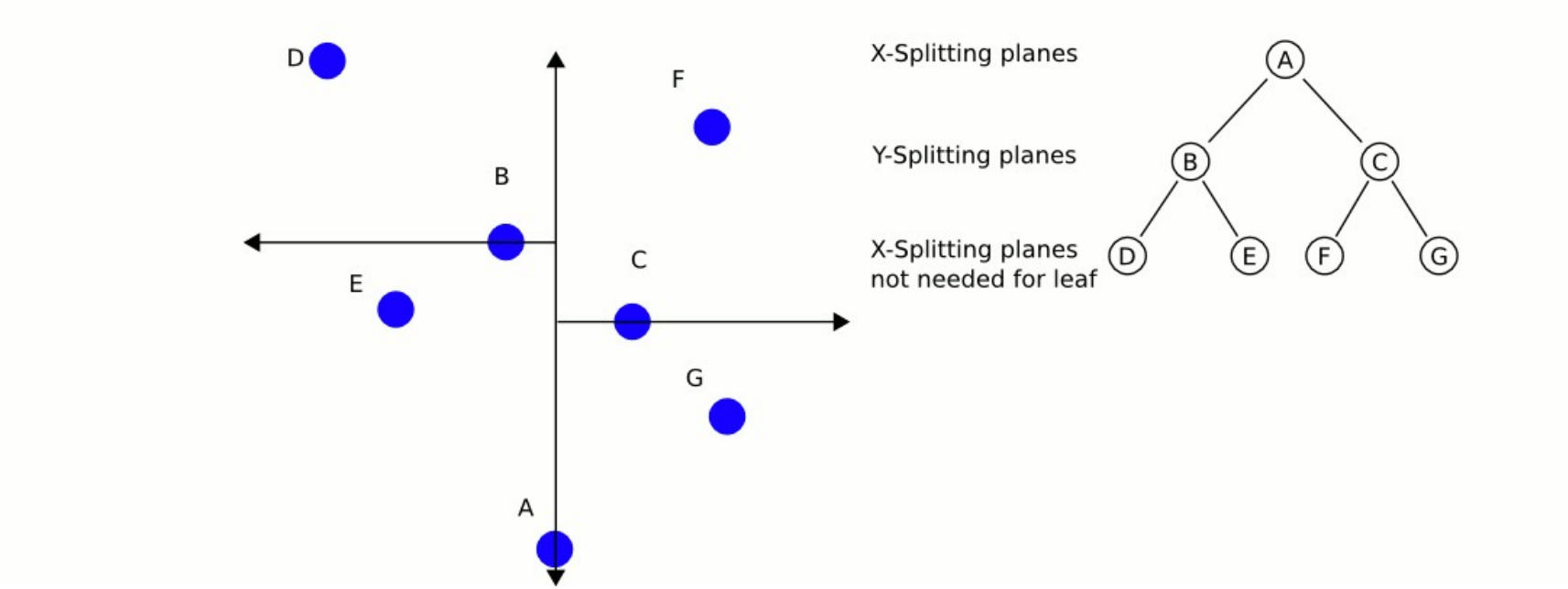




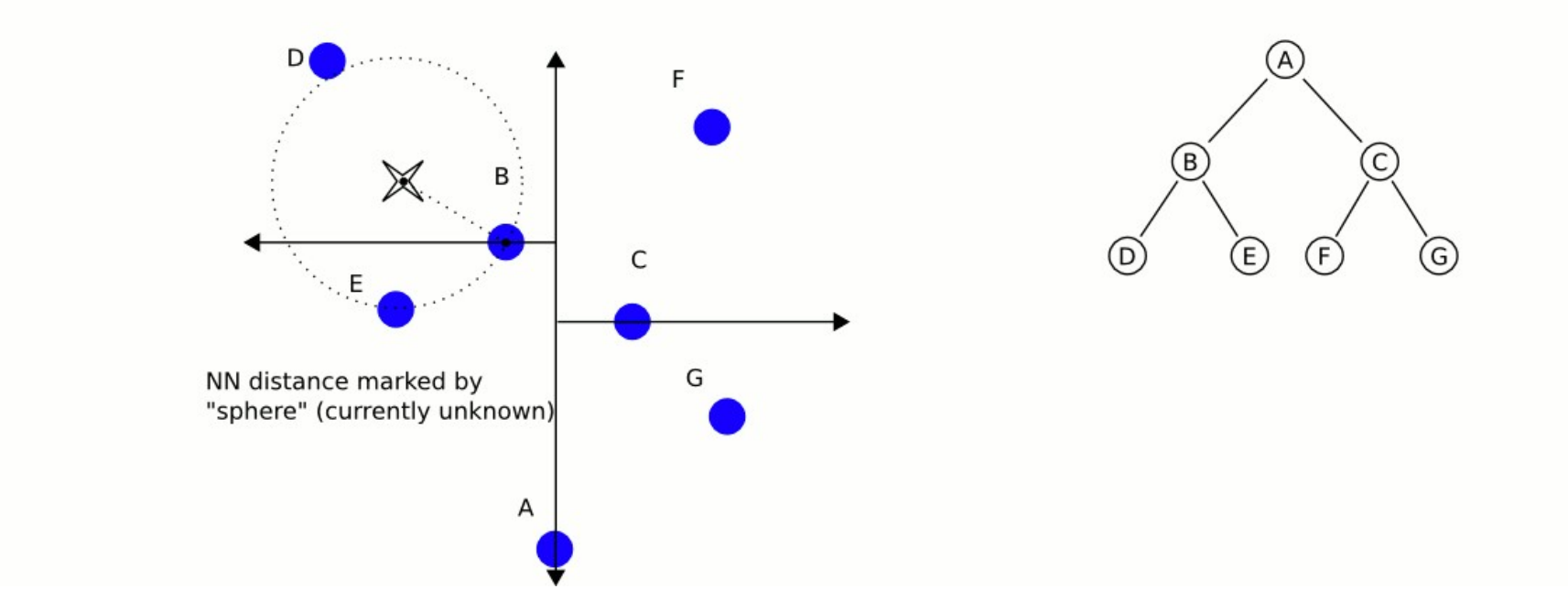
kd-trees (3 dimensional)



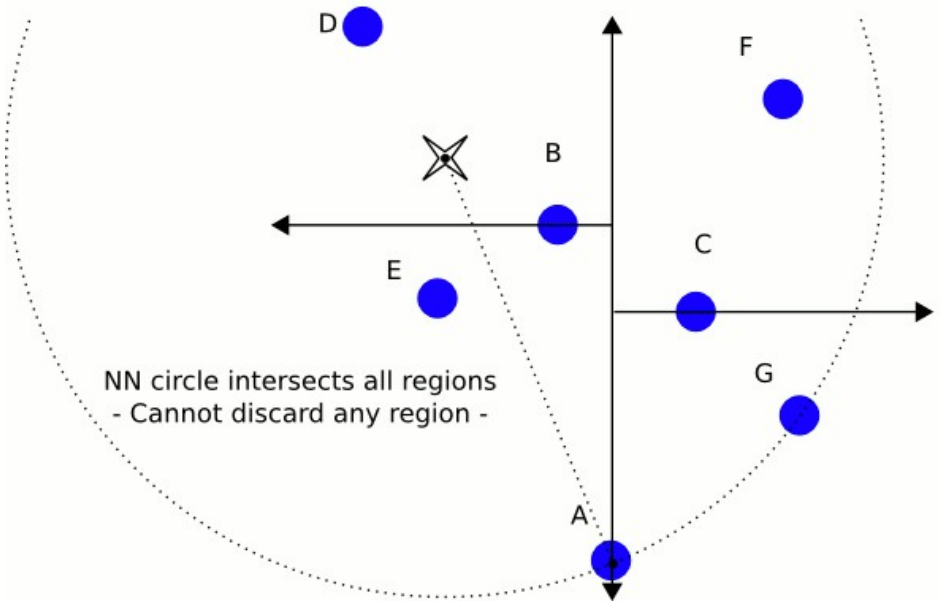
Nearest Neighbor Search

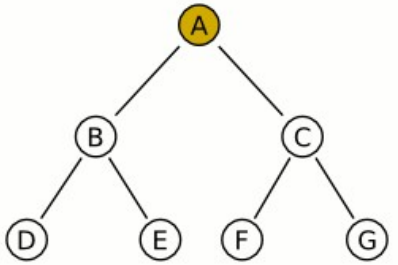


Nearest Neighbor Search



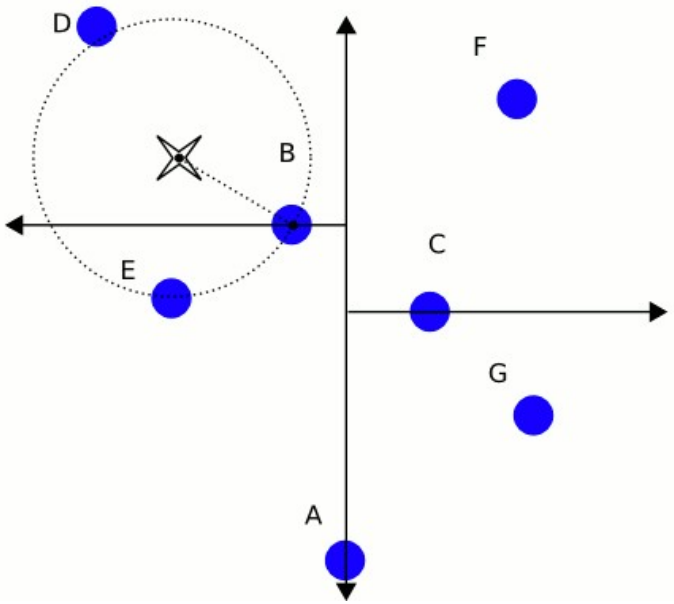
Nearest Neighbor Search

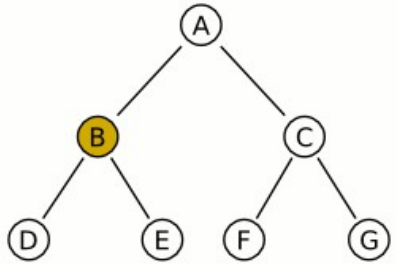




Start at A, then proceed in depth-first search (maintain a stack of parent-nodes if using a singly-linked tree). Set best estimate to A's distance
Then examine left child node

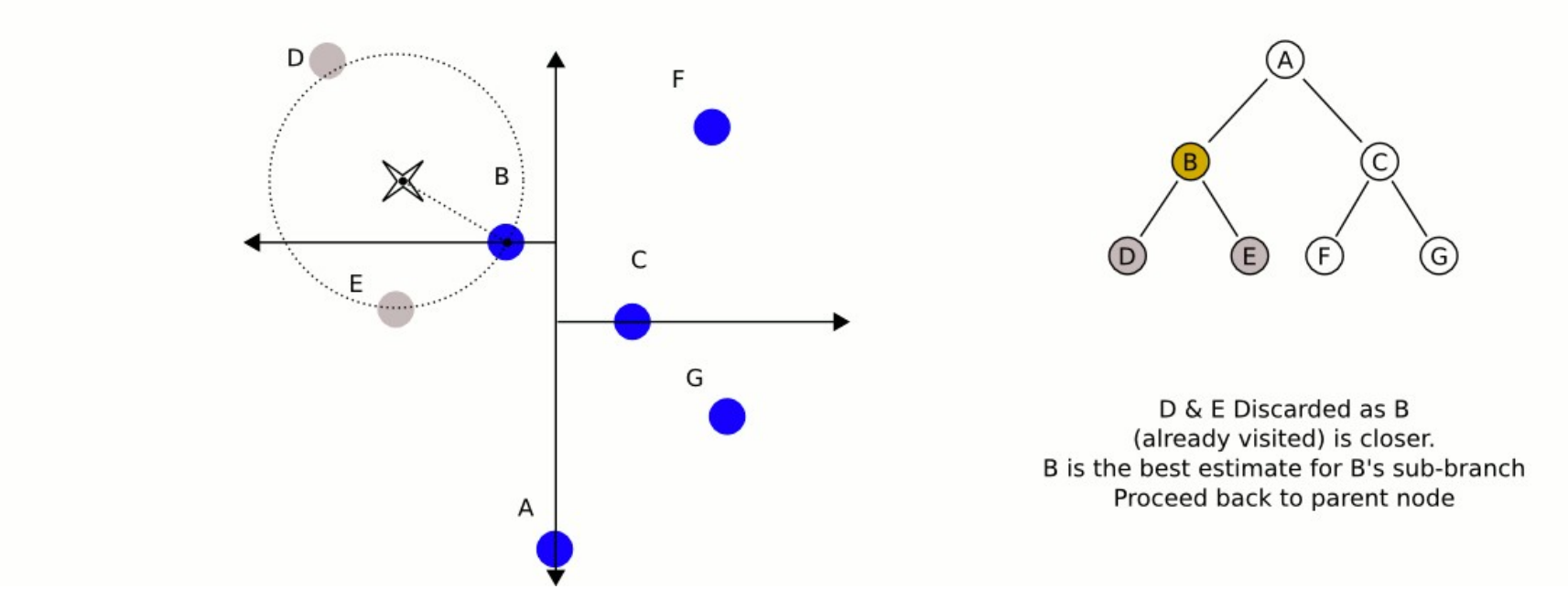
Nearest Neighbor Search



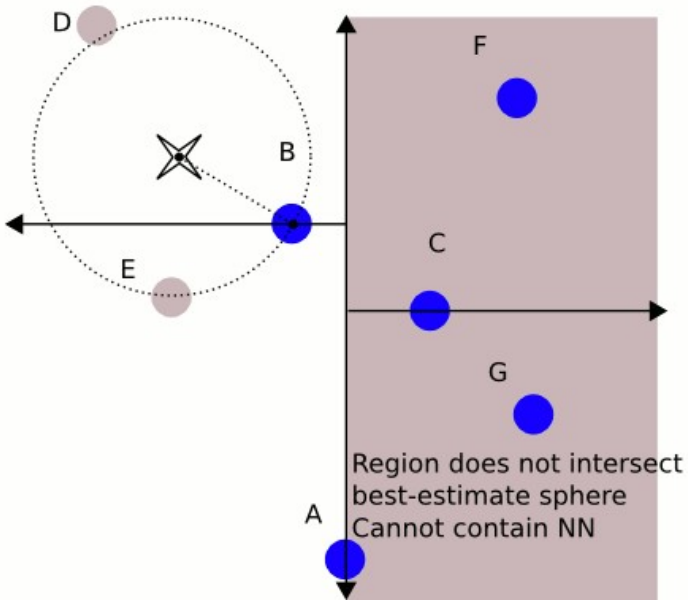


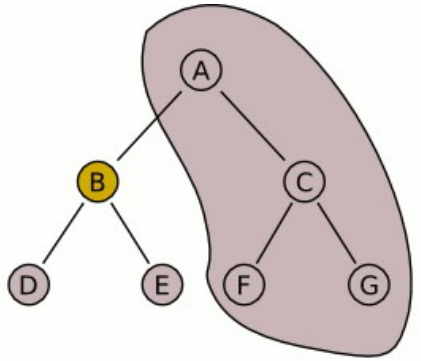
Calculate B's distance and
compare against best estimate
- It is smaller distance, so update
best estimate. Examine children (left then right)

Nearest Neighbor Search



Nearest Neighbor Search





A's children have all been searched,
B is the best estimate for entire tree

ICP- Caveats

- Convergence to the desired minimum is highly dependent upon initialization, as only local convergence is guaranteed.
- Sensitive to outliers, carried over with the analytic solution which is used as part of the iterative scheme.
- All classical solutions to these problems have been applied at one time or another.

  A very partial list:
  - simulated annealing [Gong et al. 1997], [Luck et al. 2000], [Penney et al. 2001]
  - M-estimators [Kaneko et al. 2003], [Ma and Ellis 2003]
  - least median of squares [Masuda and Yokoya 1995], [Trucco et al. 1999]
  - least trimmed squares [Chetverikov et al. 2005]
  - weighted least squares [Maurer et al. 1998], [Turk and Levoy 1994]

Iterative Closest Point (ICP)

- until converged:
  - For each point in $\mathrm{scan\ A}$, find the closest point in $\mathrm{scan\ B}$.
  - Compute the RBT that best aligns the correspondences from the previous step.

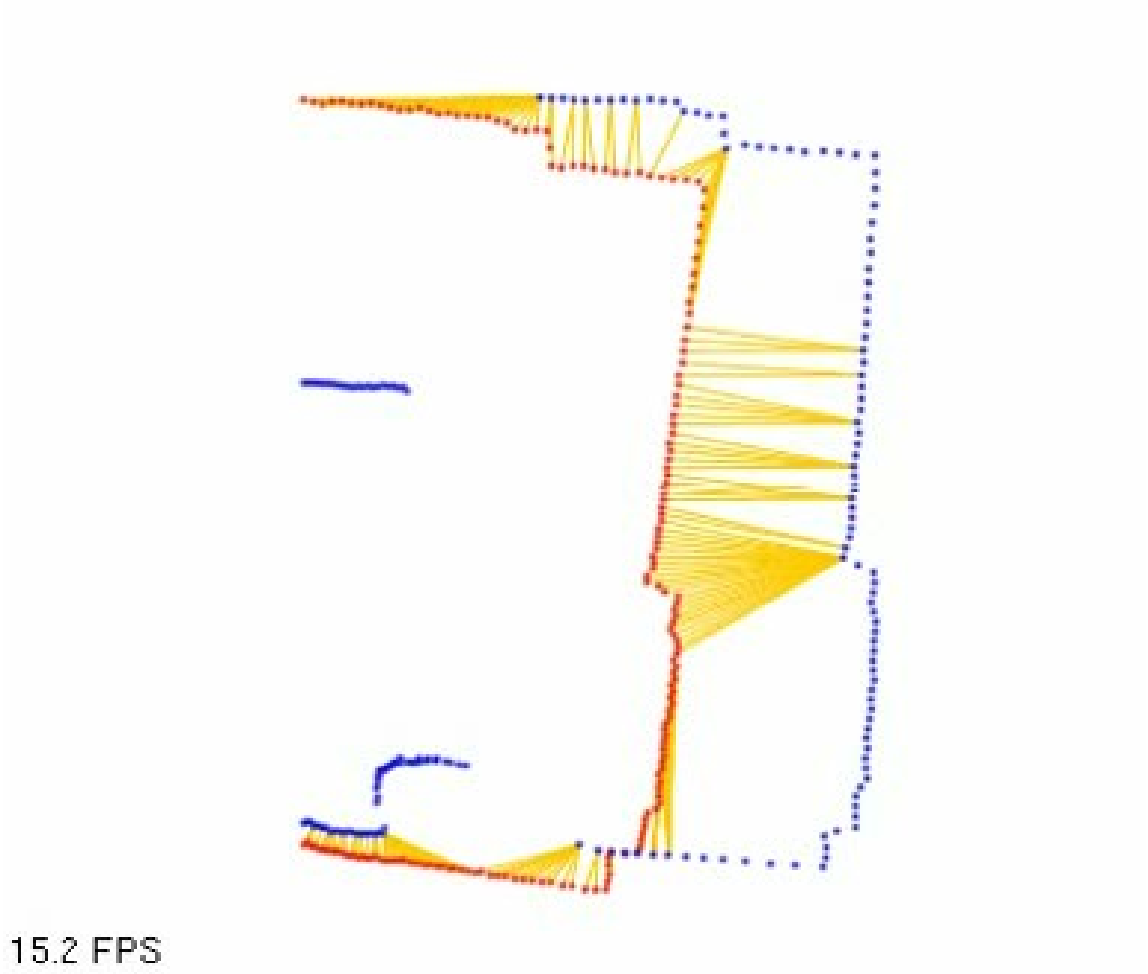
- Robustness tweaks:
  - Outliers: If distance between corresponding points is too large, delete the correspondence.
  - Also match from B to A: if the correspondences are very different in distance, delete the correspondence.

Standard ICP Algorithm

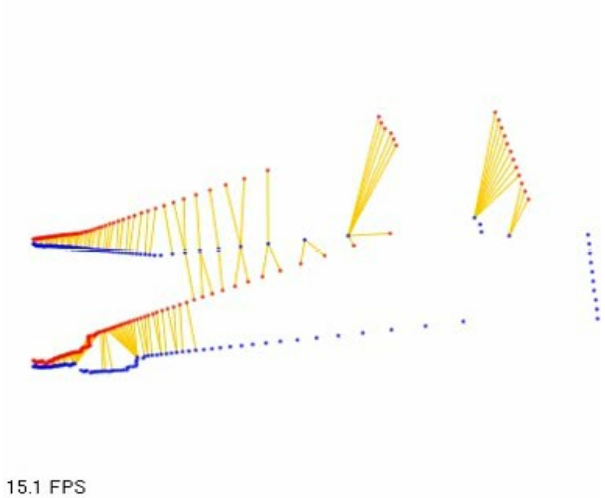
input : Two pointclouds: $A = \{a_i\}, B = \{b_i\}$
An initial transformation: T_0
output: The correct transformation, T, which aligns A and B

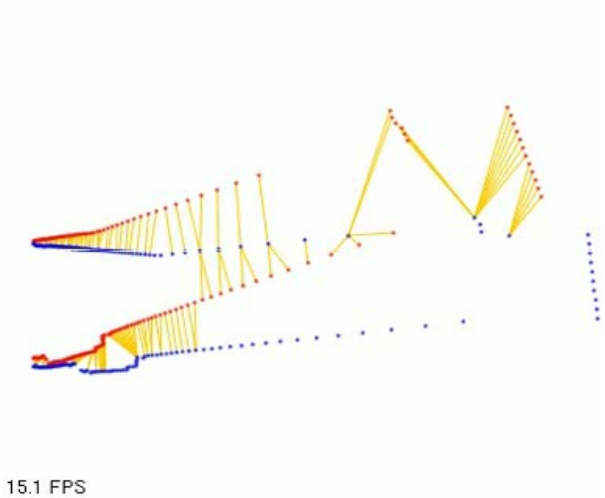
$T \leftarrow T_0$;
while *not converged* **do**
  for $i \leftarrow 1$ **to** N **do**
    $m_i \leftarrow \texttt{FindClosestPointInA}(T \cdot b_i)$;
    if $||m_i - T \cdot b_i|| \leq d_{max}$ **then**
      $w_i \leftarrow 1$;
    else
      $w_i \leftarrow 0$;
    end
  end
  $T \leftarrow \underset{T}{\text{argmin}} \{\sum_i w_i ||T \cdot b_i - m_i||^2\}$;
end

Iterative Closest Point (ICP)



ICP: Tricky case





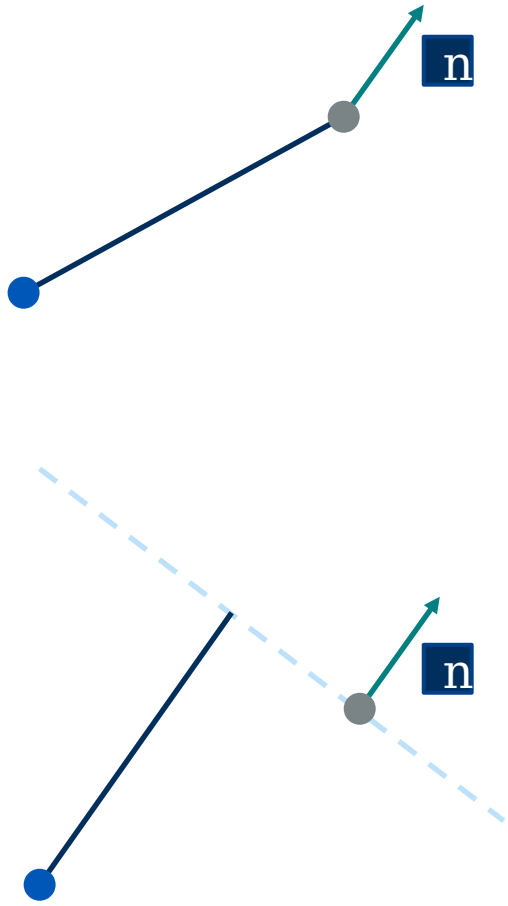
Iterative Closest Line (ICL)

- LIDARs sample space at essentially arbitrary locations.
  - No reason to think that the exact points sampled in $\mathrm{scan\ A}$ were observed in $\mathrm{scan\ B}$
  - Solution: interpolate lines between points in $\mathrm{scan\ B}$, find closest *line* instead of closest *point*.

Assigning Error Metric

- Point-to-point metric, taking into account distance and color difference

- Point-to-plane method



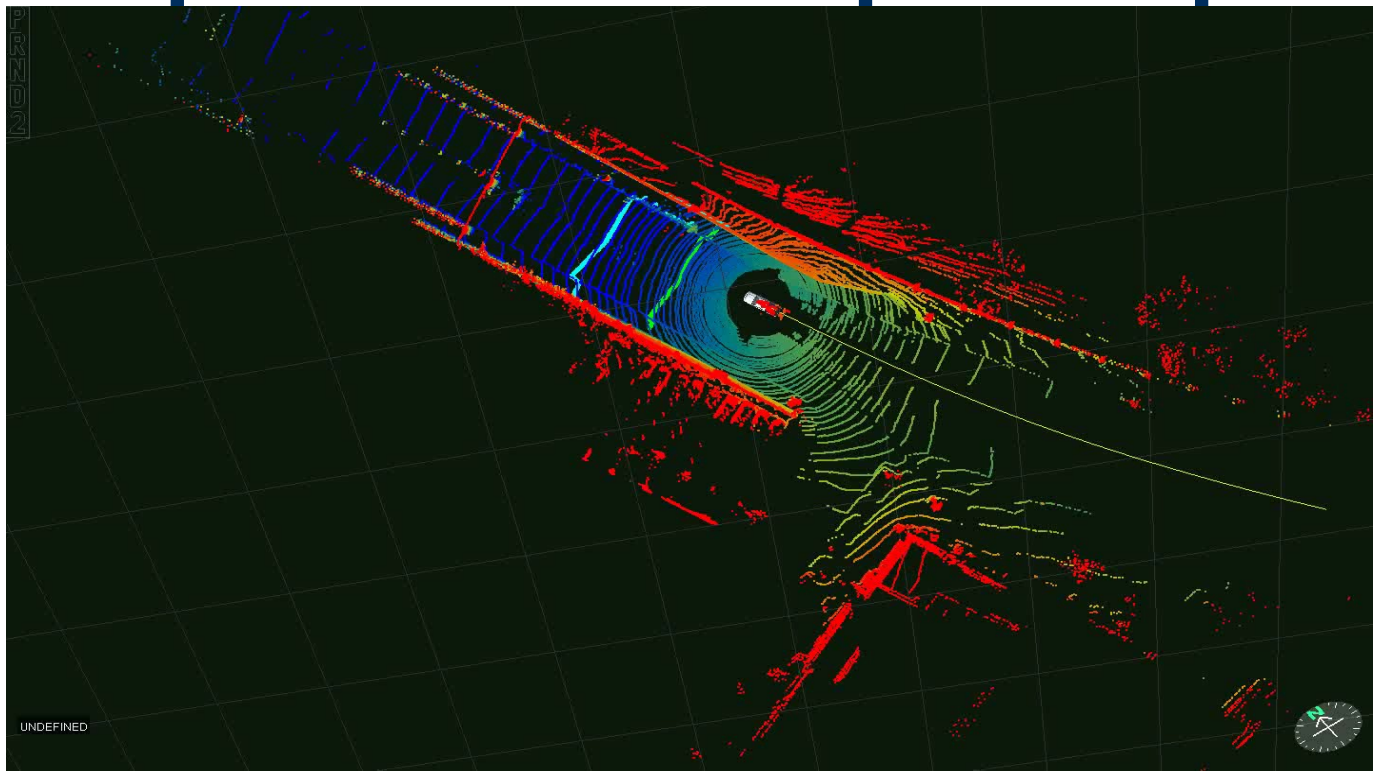
Summary

- Analytic solution
  - requires pairing
  - sensitive to outliers
  - guarantees optimality
  - assumes noise is isotropic and IID ($\sim N(0, s^2)$)

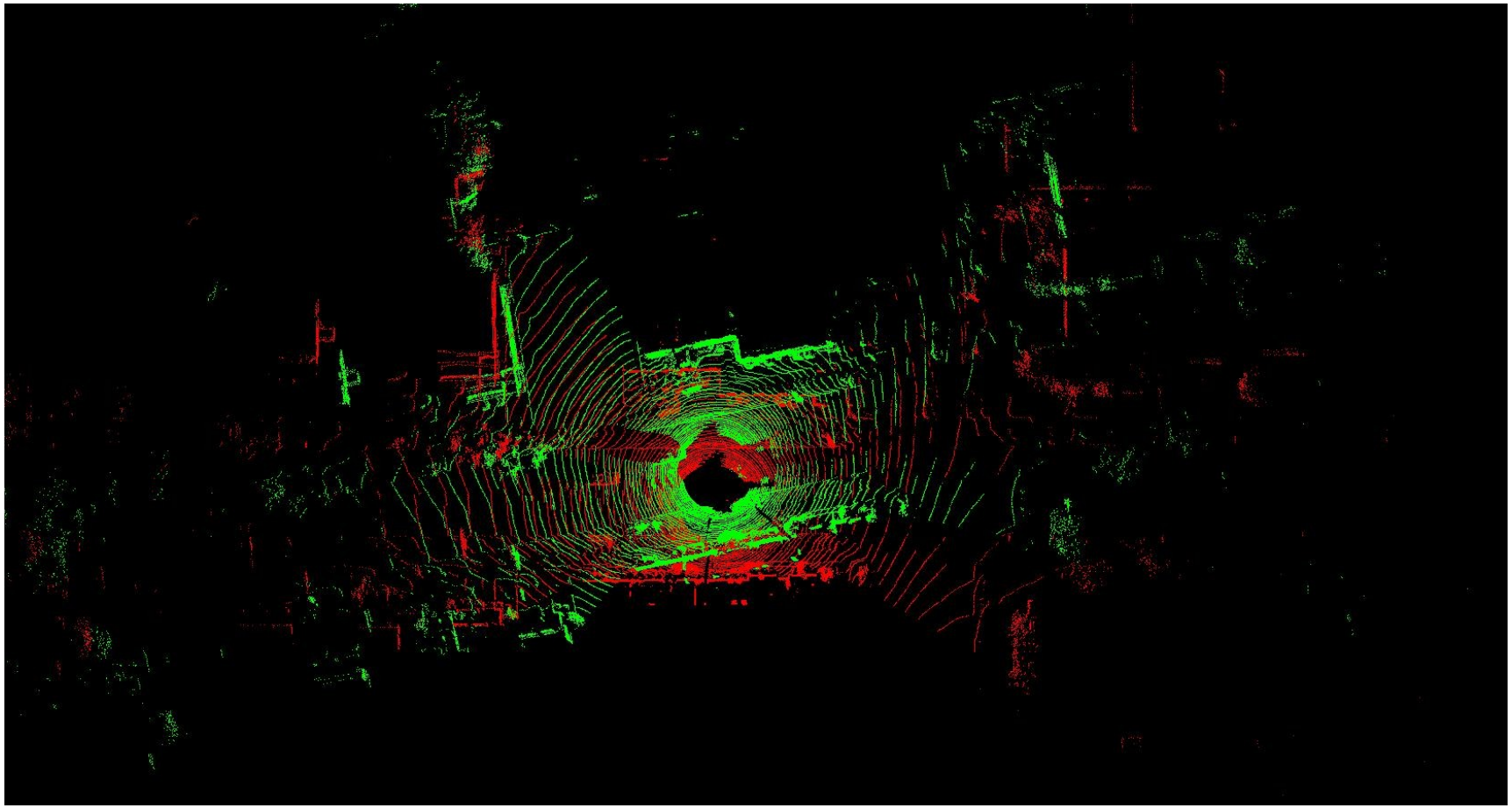
- Iterative solution
  - does not require a known pairing
  - sensitive to outliers
  - does not guarantee optimality
  - assumes noise is isotropic and IID ($\sim N(0, s^2)$)
  - requires initialization
  - requires use of spatial data structures to achieve reasonable running times on a reasonably large data set.

ICP Algorithm

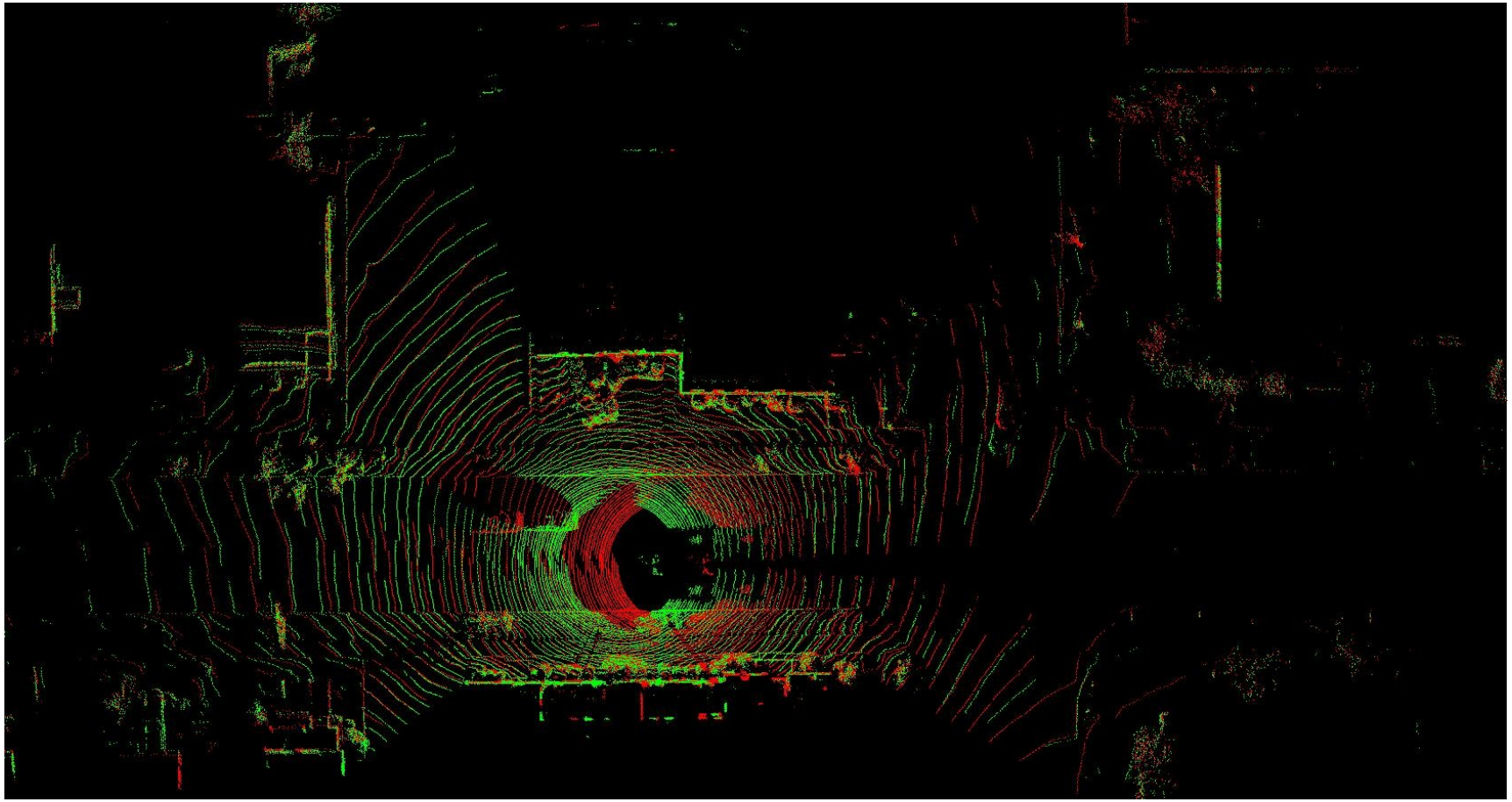
(Input: 3-D LIDAR point cloud)



Top down view of 3D data



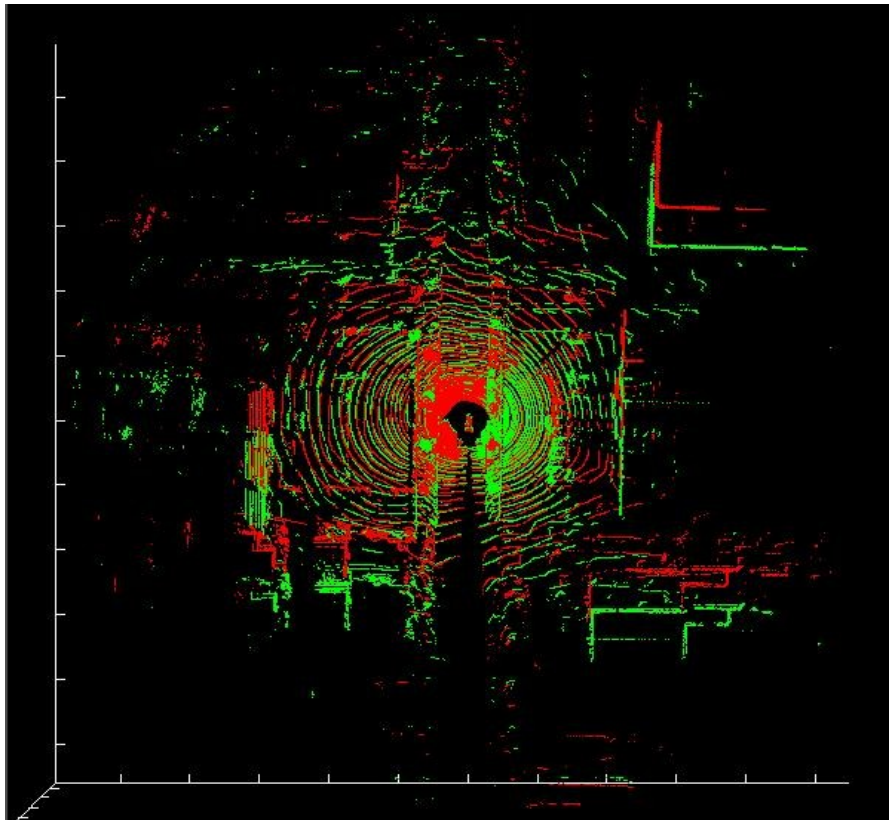
Before ICP



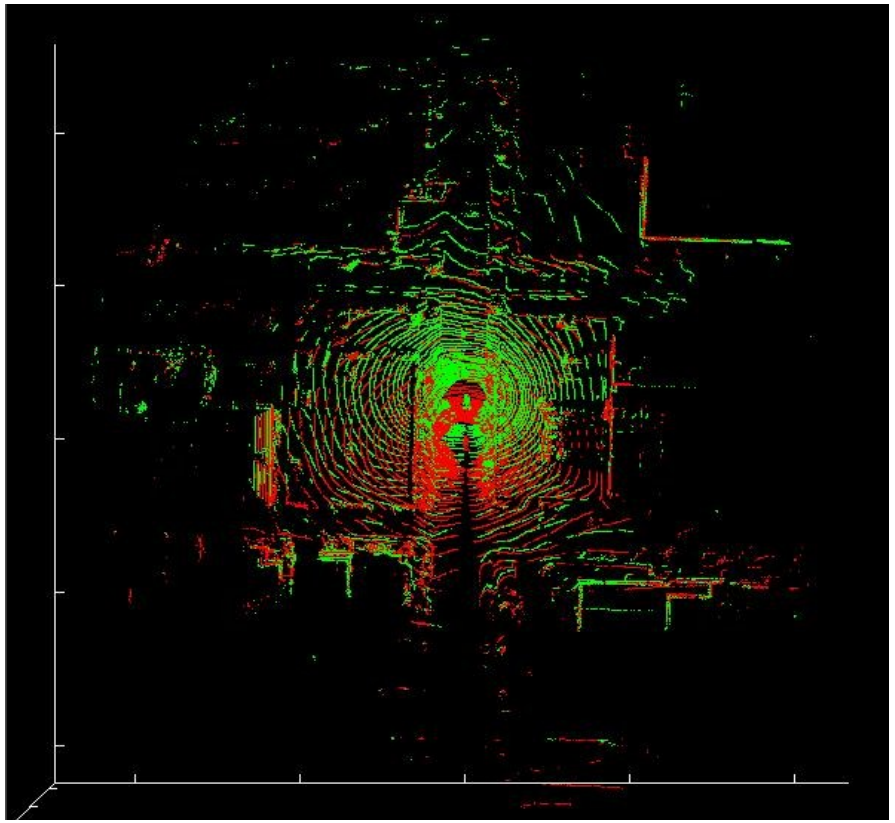
After ICP Alignment

Sometimes ICP doesn’t work

Improving ICP



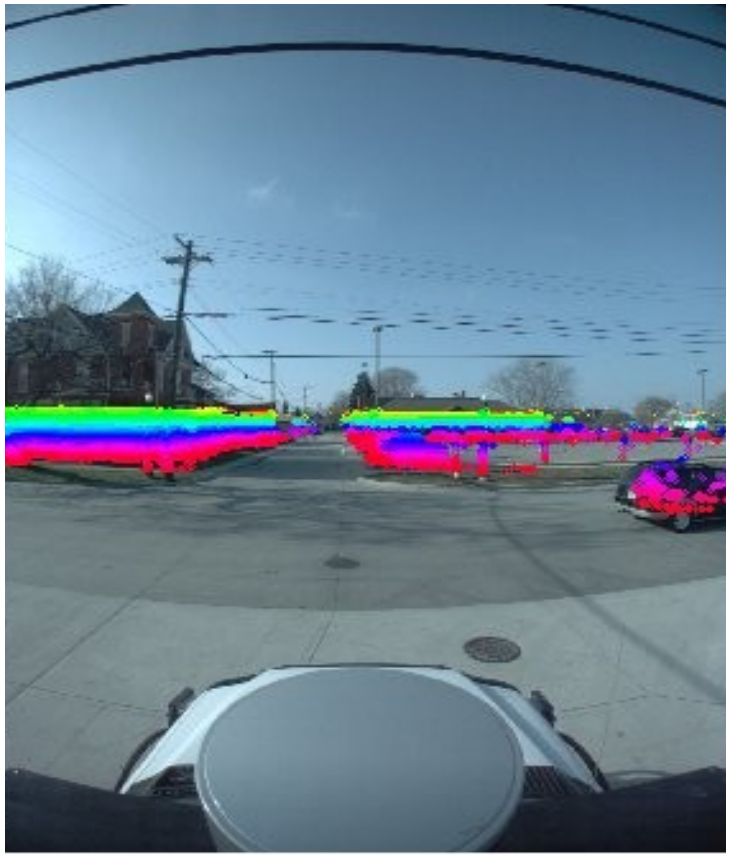
Failed scan registration when using only laser data and a naïve initial guess



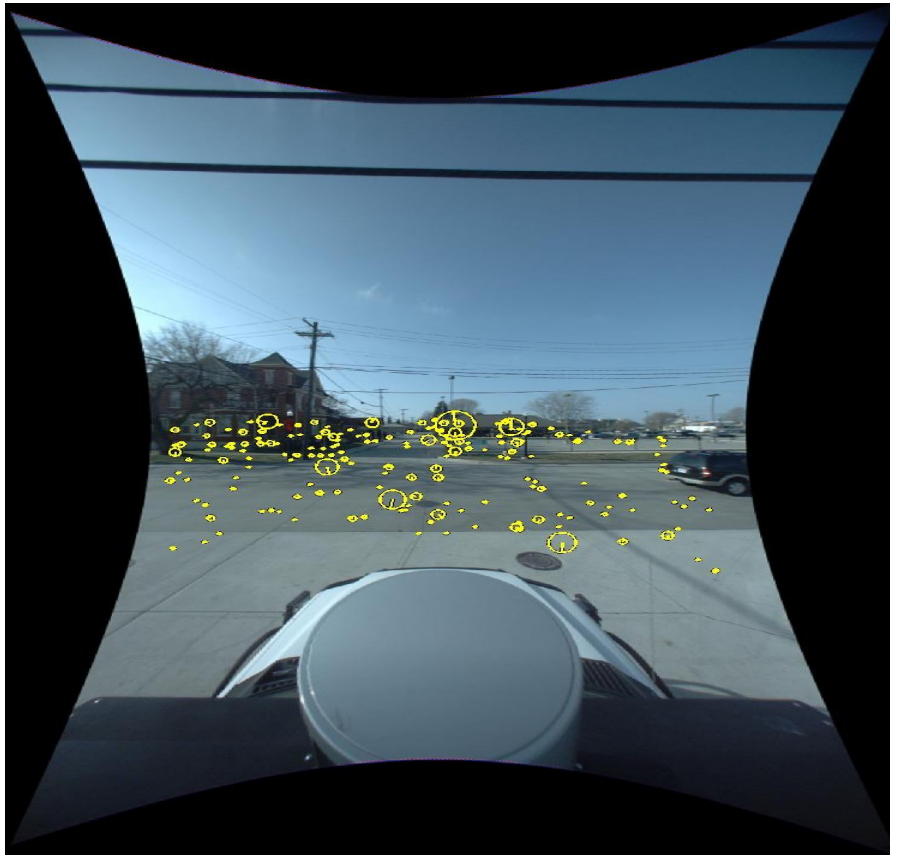
Correct scan registration obtained by using laser and camera data together (same naïve initial guess)

Camera+ICP Fused Algorithm

Assign SIFT descriptors to the 3-D LIDAR for robust point-cloud matching



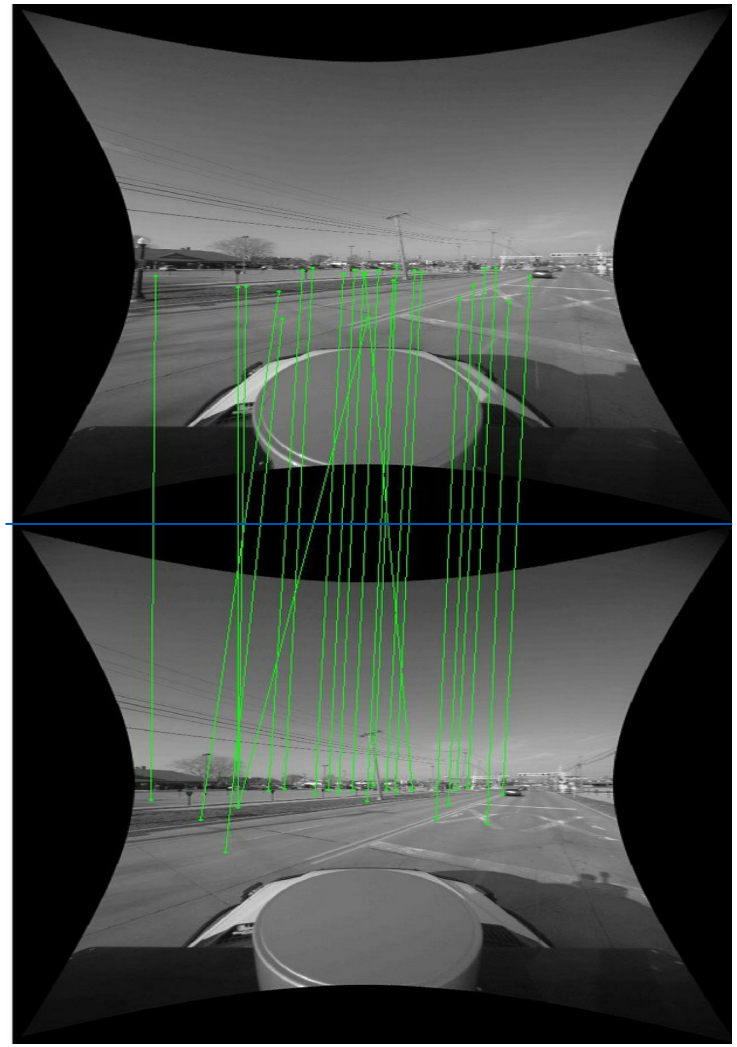
LIDAR projected into imagery
(ground-plane not rendered for visual clarity)



SIFT camera feature detections
(coincident with LIDAR data)

Camera+ICP Fused Algorithm

RANSAC algorithm that uses SIFT matches to seed ICP registration, resulting in faster and more robust point-cloud registration



Frame to frame SIFT matching

Visually bootstraped ICP Algorithm:

Extract SIFT features and re-project onto to lidar

Assign putative lidar correspondences based upon visual similarity

While (# trials)

- Draw 3 random putative correspondence pairs
- Fit rigid body transform, RBT
- Test for model consensus
- Keep RBT model with best consensus

endWhile

Refine RBT using generalized ICP

This algorithm generates more robust results with less spatial data

Visually Bootstrapped ICP Results

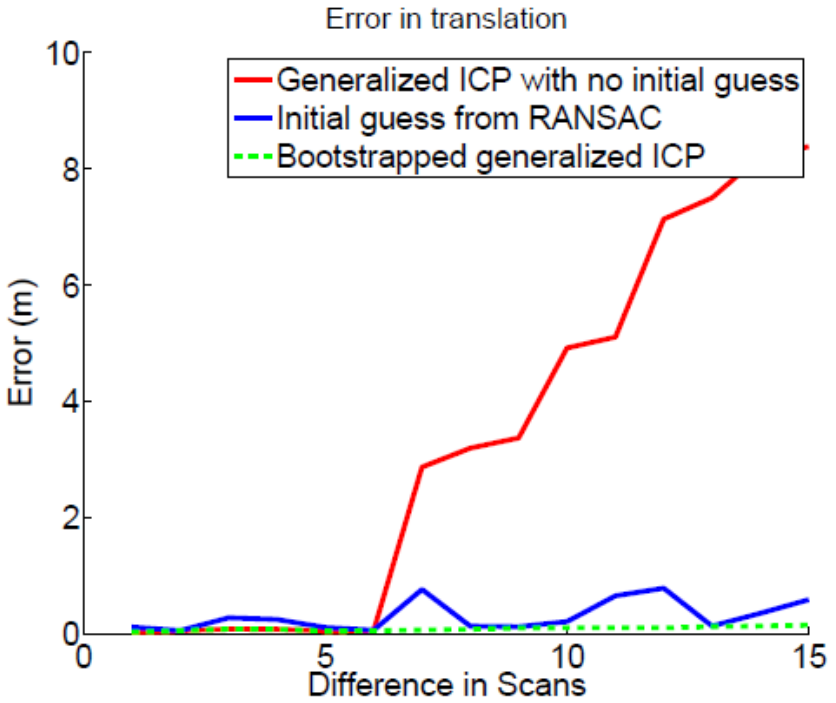


TABLE I

THIS TABLE SUMMARIZES THE ERROR IN SCAN ALIGNMENT. WE SHOW HERE THE TRANSLATION AND ROTATIONAL ERROR BETWEEN SCAN PAIRS {1-2, 1-5, 1-10, 1-15} OBTAINED AT DIFFERENT LOCATIONS. HERE WE HAVE USED THE POSE OF THE VEHICLE OBTAINED FROM A HIGH END IMU AS GROUND TRUTH TO CALCULATE ALL THE ERRORS.

	Generalized ICP with no initial guess						Initial guess from RANSAC						Bootstrapped generalized ICP					
Scans	T (m)		Ax (degrees)		An (degrees)		T (m)		Ax (degrees)		An (degrees)		T (m)		Ax (degrees)		An (degrees)	
	Err	Std	Err	Std	Err	Std	Err	Std	Err	Std	Err	Std	Err	Std	Err	Std	Err	Std
1-2	.047	.011	0	0	.05	.02	.15	.02	0	0	.223	.0003	.04	.010	0	0	.057	.110
1-5	.546	.173	.570	.20	1.15	.344	.20	.03	.43	.15	.230	.0001	.084	.010	.025	.090	.058	.006
1-10	6.37	.868	.710	.25	1.72	.573	.51	.09	.59	.01	.745	.0044	.145	.015	.030	.010	.057	.012
1-15	10.34	.834	1.86	.13	2.86	.057	1.02	.02	1.35	.54	1.15	.0021	.220	.008	.042	.015	.070	.017

T = Error in translation (meters); Ax = Error in rotation axis (degrees); An = Error in rotation angle (degrees)
Err = Average Error; Std = Standard Deviation

SLAM GPS-Denied Demonstration

OmniSTAR HP GPS ground-truth and Applanix POS-LV IMU
1.6 km loop around Dearborn

Research & Advanced Engineering

Dead Reckoning

No GPS Reception – Velocity Integration of MEMs-based XSENS MTi-G IMU Data

Research & Advanced Engineering

SLAM Demonstration

Ground Truth vs. Laser Odometry

No GPS Reception – Intra-frame Motion Compensated by XSENS MTi-G IMU

Research & Advanced Engineering

Summary:
Feature vs. non-feature matching

- Features (pros)
  - Data reduction
  - Noise filtering
  - Extraction + Matching is often fast
  - Can handle from large prior uncertainties.

- Non-Feature Matching (pros)
  - General purpose: don’t require world to contain our features
  - ICP/ICL:
    - Very fast, easy to implement. Local maxima problems.
  - Correlation
    - Fast enough, very robust

Bibliography

Rigid Registration with known point pairs:

Rotation matrix:

- P. H. Schönemann, “A generalized solution of the orthogonal procrustes problem”, Psychometrika, vol. 31, pp. 1-10, 1966.
- K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-D point sets,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, pp. 698–700, 1987.
- B. K. P. Horn, H. M. Hilden, and S. Negahdaripour, “Closed-form solution of absolute orientation using orthonormal matrices,” *Journal of the Optical Society of America A*, vol. 5, no. 7, pp. 1127–1135, 1988.
- S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 4, pp. 376–380, 1991.
  [This paper guarantees that the solution is a rotation, where the previous papers also admitted reflections].

Quaternion:

- O. D. Faugeras and M. Hebert, “The representation, recognition, and locating of 3-D objects,” *Int. J. Rob. Res.*, vol. 5, no. 3, pp. 27–52, 1986.
- B. K. P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, April 1987.

Bibliography

Improved least squares methods:

- N. Ohta, K. Kanatani, “Optimal estimation of three-dimensional rotation and reliability evaluation”, *IEEE Trans. Inf. Sys.*, vol. E82-D, no. 11, pp. 1247-1252, 1998.
  [This paper relaxes the assumption that the noise is isotropic and IID].
- C. R. Maurer Jr., R. J. Maciunas, J. M. Fitzpatrick, “Registration of head CT images to physical space using a weighted combination of points and surfaces”, *IEEE Trans. Med. Imag.*, vol. 17, no. 5, pp. 753-761, 1998.
- M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”, *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.

Comparison of analytic solutions:

- D. W. Eggert, A. Lorusso, R. B. Fisher, “Estimating 3-D rigid body transformations: a comparison of four major algorithms”, Mach. Vis. Appl., vol. 9, no. 5/6, pp. 272-290, 1997.

Bibliography

Rigid Registration with unknown point pairs:

Original ICP:

- Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.
- P. J. Besl and N. D. McKay, “A method for registration of 3D shapes,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, pp. 239–255, 1992.
- Z. Zhang, “Iterative point matching for registration of free-form curves and surfaces,” *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.

Improved (robust) ICP:

- J. Gong, R. Bachler, M. Sati, L. P. Nolte, “Restricted surface matching, a new approach to registration in computer assisted surgery”, *Computer Vision, Virtual Reality and Robotics in Medicine and Medical Robotics and Computer assisted Surgery*, pp. 597–605, 1997.
- J. P. Luck, C. Q. Little, W. Hoff, “Registration of range data using a hybrid simulated annealing and iterative closest point algorithm”, *International Conference on Robotics and Automation*, pp. 3739–3744, 2000.
- G. P. Penney, P. J. Edwards, A. P. King, J. M. Blackall, P. G. Batchelor, D. J. Hawkes, “A stochastic iterative closest point algorithm (stochastICP)”, *Medical Image Computing and Computer-Assisted Intervention*, pp. 762–769, 2001.

Bibliography

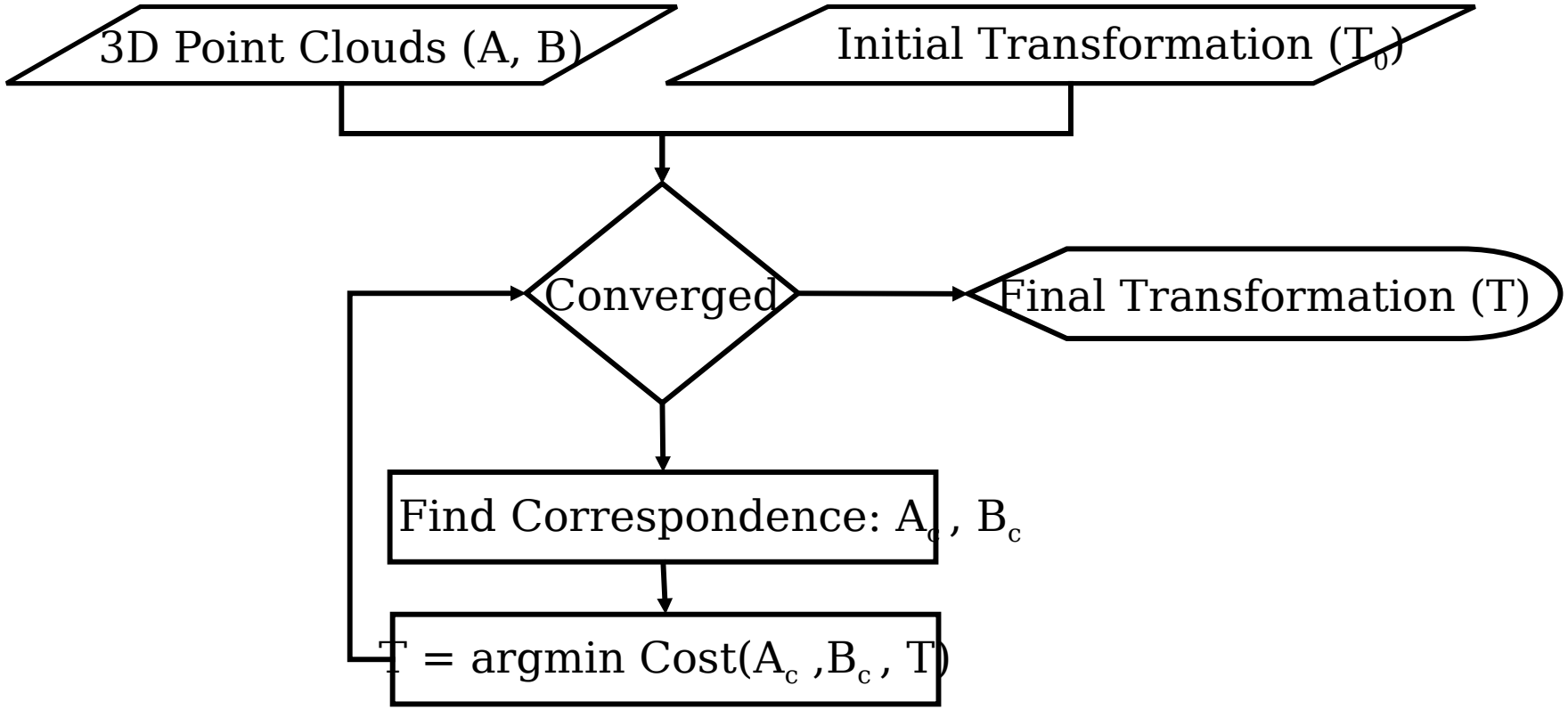
- S. Kaneko, T. Kondo, A. Miyamoto, “Robust matching of 3D contours using iterative closest point algorithm improved by M-estimation”, *Pattern Recognition*, vol. 36, no. 9, pp. 2041–2047, 2003.
- B. Ma and R. E. Ellis, “Robust registration for computer-integrated orthopedic surgery: Laboratory validation and clinical experience”, *Medical Image Analysis*, vol. 7, no. 3, pp. 237–250, 2003.
- T. Masuda and N. Yokoya, “A robust method for registration and segmentation of multiple range images”, *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 295–307, 1995.
- E. Trucco, A. Fusiello, and V. Roberto, “Robust motion and correspondence of noisy 3-D point sets with missing data”, *Pattern Recognition Letters*, vol. 20, no. 9, pp. 889–898, 1999.
- D. Chetverikov, D. Stepanov, and P. Krsek, “Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm”, *Image and Vision Computing*, vol. 23, no. 3, pp. 299–309, 2005.
- G. Turk and M. Levoy, “Zippered polygon meshes from range images”, *SIGGRAPH Computer graphics and interactive techniques*, pp. 311–318, 1994.

Bibliography

kd-tree:

- J. Friedman, J. Bentley, R. Finkel R, “An algorithm for finding best matches in logarithmic expected time", *ACM Transactions on Mathematical Software*, vol.3, pp. 209-226, 1977.
- M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer-Verlag, 1997.
- H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison Wesley, 1990.
- R. L. Sproull, “Refinements to nearest-neighbor searching”, *Algorithmica*, vol.6, pp. 579-589, 1991.
- J.P. Williams, R. H. Taylor, L. B. Wolf, “Augmented k-D techniques for accelerated registration and distance measurement of surfaces”, *Computer Aided Surgery: Computer-Integrated Surgery of the Head and Spine*, pp. 1-21, 1997.
- J. McNames, “A Fast Nearest-Neighbor Algorithm Based on a Principal Axis Search Tree”, *Pattern Anal. Machine Intell.*, vol. 23, no. 9, pp. 964-976, 2001.

Iterative Closest Point (ICP)



Besl, P. J. and McKay, N. D. (1992). A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239-255.

Generalized ICP

Generalized-ICP is based on attaching a probabilistic model to the minimization step of the standard ICP algorithm.

$$\mathrm{A} = \{\boldsymbol{a}_i\}_{i=1,2\ldots N} \quad \mathrm{B} = \{\boldsymbol{b}_i\}_{i=1,2\ldots N} \quad (\text{s.t. } \boldsymbol{a}_i == \boldsymbol{b}_i)$$

$$\boldsymbol{a}_i \sim N(\boldsymbol{\mu}_{ai}\ ,\ \Sigma_{Ai}) \ ;\ \boldsymbol{b}_i \sim N(\boldsymbol{\mu}_{bi}\ ,\ \Sigma_{Bi})$$

"Generalized ICP" by A. Segal, D. Haehnel, S. Thrun. RSS 2009

Generalized ICP

Since $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are assumed to be drawn from independent Gaussians, the reprojection error [$\boldsymbol{d}_i = \boldsymbol{b}_i$ - T$\boldsymbol{a}_i$] for each point correspondence should have the following distribution:

$$\boldsymbol{d}_i \sim N(\boldsymbol{\mu}_{bi} - \mathrm{T}\boldsymbol{\mu}_{ai},\ \Sigma_{Bi} + \mathrm{T}\,\Sigma_{Ai}\,\mathrm{T}')$$

$$\boldsymbol{\mu}_{bi} - \mathrm{T}\boldsymbol{\mu}_{ai} = 0 \quad \Rightarrow \quad \boldsymbol{\mu}_{bi} = \mathrm{T}\boldsymbol{\mu}_{ai}$$

OR

$$\boldsymbol{d}_i \sim N(\mathbf{0},\ \Sigma_{Bi} + \mathrm{T}\,\Sigma_{Ai}\,\mathrm{T}')$$

"So we want to maximize the likelihood that the error d_i is drawn from the zero-mean probability distribution mentioned above"

Generalized ICP

The MLE is given by:

$$\mathrm{T} = \operatorname{argmax} \prod_{i} p(\boldsymbol{d}_i) = \operatorname{argmax} \sum_{i} \log\left(p(\boldsymbol{d}_i)\right)$$

$$\mathrm{T} = \operatorname{argmin} \sum_{i} \boldsymbol{d}_i' \left(\Sigma_{Bi} + \mathrm{T}\,\Sigma_{Ai}\,\mathrm{T}'\right)^{-1} \boldsymbol{d}_i$$