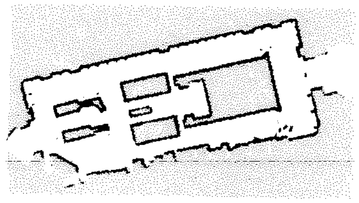
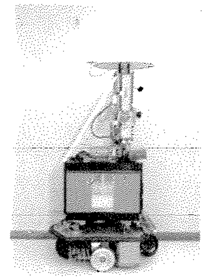
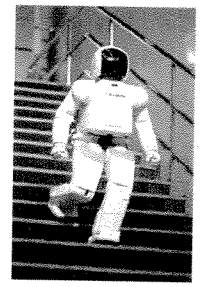
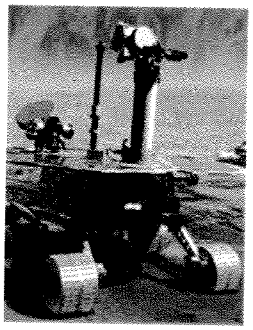
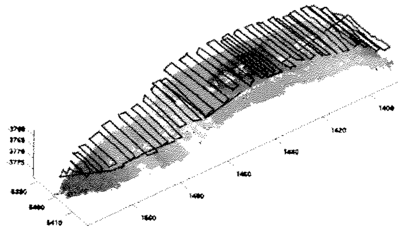
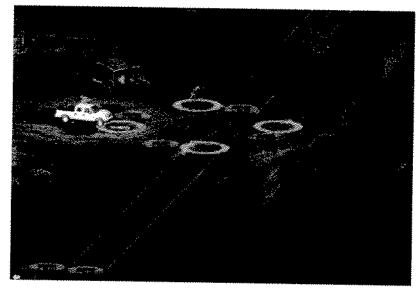












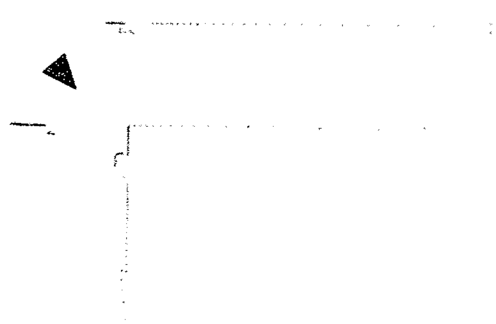
EECS568 Mobile Robotics: Methods and Principles

Prof. Edwin Olson

L15. Laser Range Finders

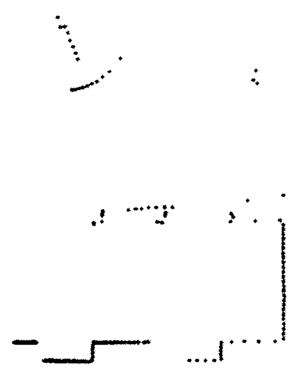
Laser Scanners

- Measure range via time-of-flight
- SICK
  - Industrial safety
  - 180 samples, 1 degree spacing, 75Hz
  - Resolution ~1cm, ~0.25 deg.
  - Interlacing
  - Max range: "80m" 30m fairly reliable
  - Intensity
  - $4500

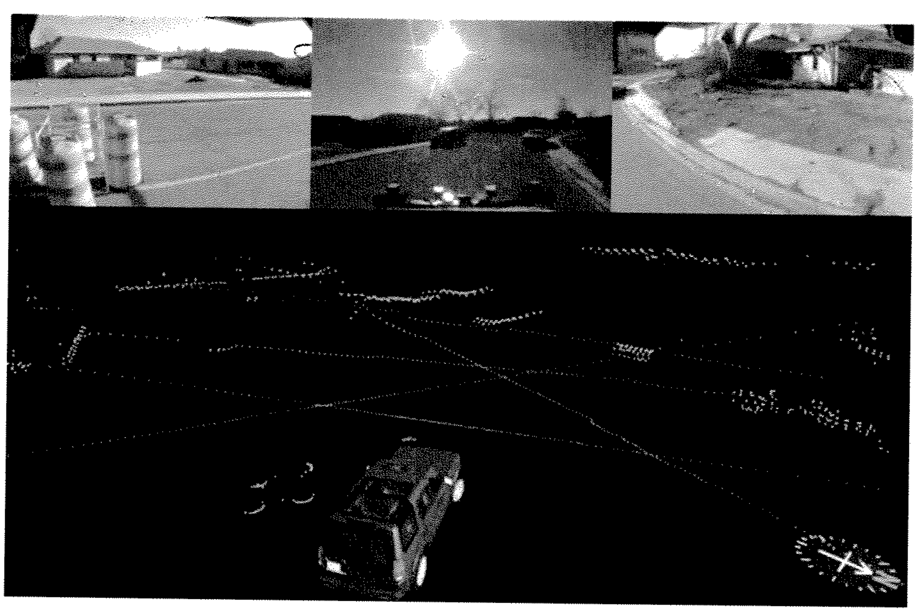


- Hokuyo
  - ~1080samples, 0.25deg spacing, 270 degree FOV, 10-40Hz
  - $1100 - $5000

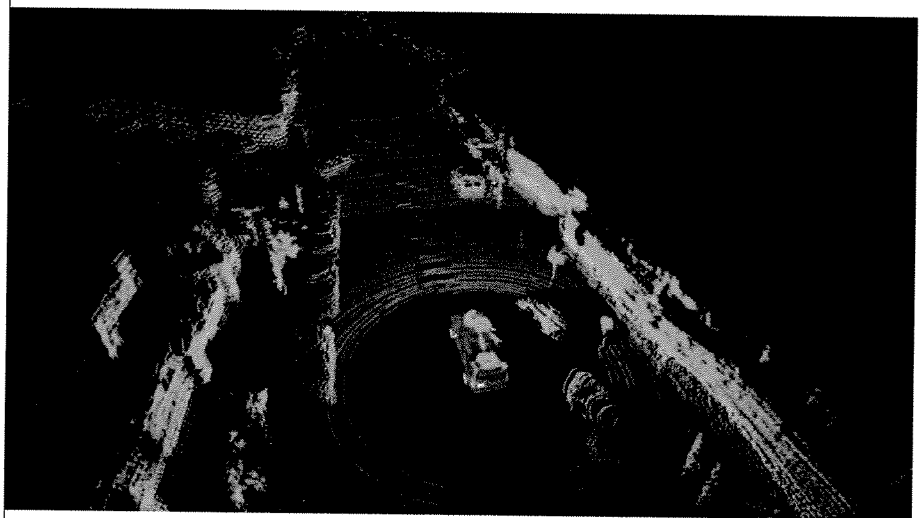
Laser Scanners: Planar Environments

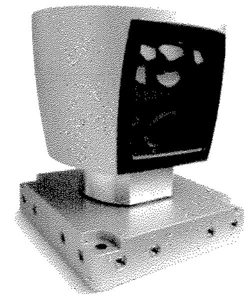


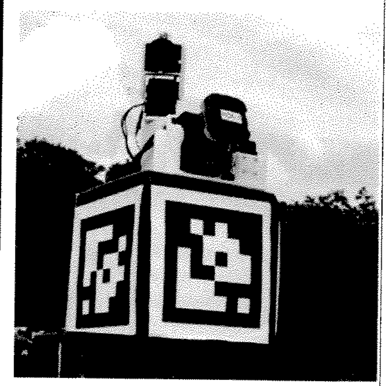
Planar LIDARS in 3D



3D LIDAR Sensors







Aligning scans = Measuring motion

- Two important cases:
  - Incremental motion. Use lidar to augment (or replace!) odometry.
  - Loop closing. Identify places we've been before to over-constrain robot trajectory.

Feature Extraction

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

- Possible features:
  - Lines
  - Corners
  - Occlusion boundaries/depth discontinuities
  - Trees (!)

Line Extraction

- Given laser scan, produce a set of 2D line segments

- Two basic approaches:
  - Divisive ("Split N Fit")
  - Agglomerative

- Our plan:
  - 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?

Line Fitting

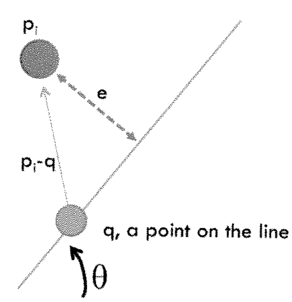
- Assume we know a set of points belong to a line. How do we compute parameters of the line?
- How to parameterize the line?
  - Slope + y intercept?
  - Endpoints of line
  - A point on the line + unit vector
  - R + theta

Line Fitting: 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:

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

$$e^2 = \sum (x^T v)^2 \qquad v = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

$$= \sum_i (x_i \cos\theta + y_i \sin\theta)^2$$

$$= \sum_i x_i^2 \cos^2\theta + 2x_i y_i \cos\theta \sin\theta + y_i^2 \sin^2\theta$$

$$\frac{\partial e^2}{\partial \theta} = \sum_i -2x_i^2 \cos\theta \sin\theta + 2x_i y_i (-\sin^2\theta + \cos^2\theta) + 2y_i^2 \sin\theta \cos\theta = 0$$

$$= \sum_i -x_i^2 \sin 2\theta + 2x_i y_i \cos 2\theta + y_i^2 \sin 2\theta = 0$$

$$= -M_{xx} \sin 2\theta + 2M_{xy} \cos 2\theta + M_{yy} \sin 2\theta = 0$$

$$= (-M_{xx} + M_{yy}) \sin 2\theta + 2M_{xy} \cos 2\theta = 0 \Rightarrow (-M_{xx} + M_{yy}) \sin 2\theta = -2M_{xy} \cos 2\theta$$

$$\Rightarrow \frac{\sin 2\theta}{\cos 2\theta} = \frac{2M_{xy}}{M_{yy} - M_{xx}} \qquad \frac{(\Delta y)}{(\Delta x)}$$

$$\Rightarrow \theta = \frac{1}{2} \operatorname{atan2}(2M_{xy}, M_{yy} - M_{xx})$$

N.B. There are 2 solutions due to the $\frac{1}{2}$...

$$\theta = \frac{1}{2}\operatorname{atan}() \quad \text{and} \quad \theta = \frac{1}{2}(2\pi + \operatorname{atan}())$$

but both give the same line!

Is this a max or a min? (We want max!)

actually, $\theta = \frac{1}{2} \operatorname{atan}\left(\frac{2M_{xy}}{M_{yy} - M_{xx}}\right)$. Using atan2 destroyed 2 more solutions! (tan is periodic with π) with factor $\frac{1}{2}$, this will lead to $\{\theta, \theta + \frac{\pi}{2}, \theta + \pi, \theta + \frac{3\pi}{2}\}$

Line Fitting: Strategy

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

- Method: work in terms of moments

$$M_x = \sum p_x$$
$$M_y = \sum p_y$$
$$M_{xx} = \sum p_x^2$$
$$M_{xy} = \sum p_x p_y$$
$$M_{yy} = \sum p_y^2$$

$$\begin{aligned} C_{xx} &= \frac{1}{N} M_{xx} - \left(\frac{M_x}{N}\right)^2 \\ C_{xy} &= \frac{1}{N} M_{xy} - \frac{M_x}{N}\frac{M_y}{N} \\ C_{yy} &= \frac{1}{N} M_{yy} - \left(\frac{M_y}{N}\right)^2 \end{aligned}$$

Line Fitting

- Solution:

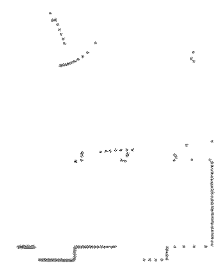
$$\begin{aligned} q_x &= \frac{1}{N} M_x \\ q_y &= \frac{1}{N} M_y \\ \theta &= \frac{\pi}{2} + \frac{1}{2}\mathrm{atan2}(-2C_{xy}, C_{yy} - C_{xx}) \end{aligned}$$

- How does this answer compare to SVD-based line fitting?
  - In fact, \theta gives the dominant eigenvector!

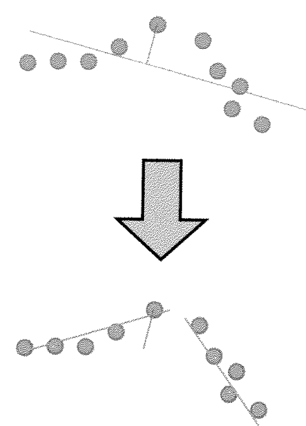
- *Because all quantities are written in terms of moments, we can compute this quantity incrementally and without storing all the points!*

Which points belong together?

- If we know which points belong together, we can compute the line.
- But we don't know which points belong to a single line!
- For now: make use of fact that lidar points arrive "in order"

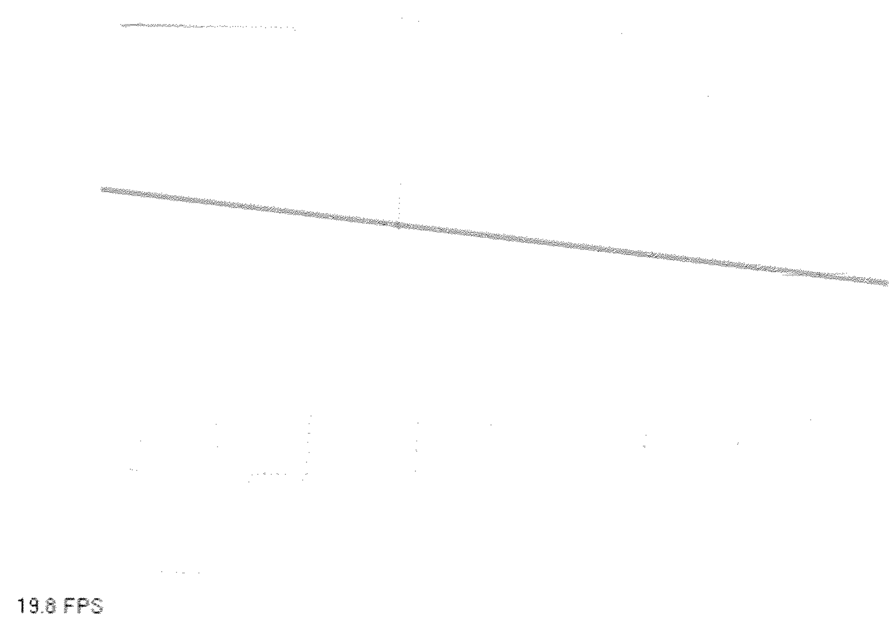


Divisive Line Fitting



- 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)

Divisive Line Fitting



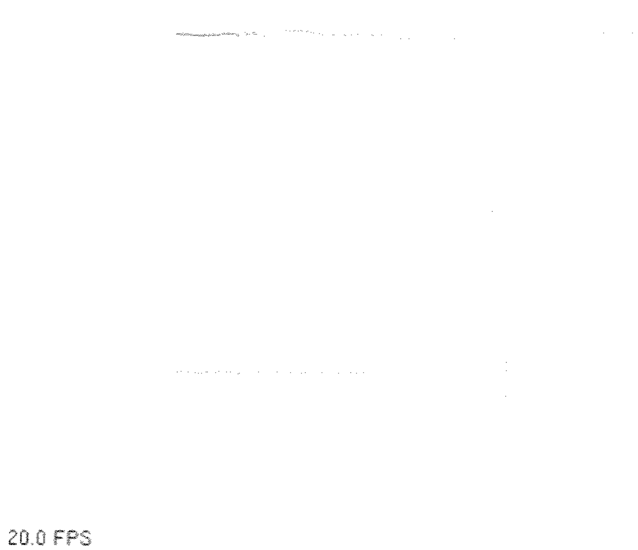
Divisive Line Fitting

- 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"
- Complexity?
  - As many as N divisions, each requiring up to N points. O(N^2)

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 minimum error > thresh then
      - return lines
    - else
      - merge lines with minimum error

Agglomerative Line Fitting



Agglomerative Line Fitting

- Pros:
  - Good quality, matches human expectations
- Cons:
  - Assumes points are in scan order
  - A little bit tricky to implement *quickly* (use a min-heap)
- Complexity?

RANSAC

- RANdom SAmple Consensus
- best_model = null
- best_consensus = -1
- do forever
  - Select enough points at random to fit a model M
  - Compute consensus = # of points that agree with M
  - if consensus > best_consensus
    - best_model = M
    - best_consensus = c

RANSAC: details

- What is the model for line fitting?
  - How many points required?
  - What if we wanted to fit parabolas instead?

- How many iterations do we need?
  - What is the probability that we pick p inliers?

- How do we compute consensus? How close is "close enough"?
  - Threshold based on sensor noise
  - "Soft" consensus

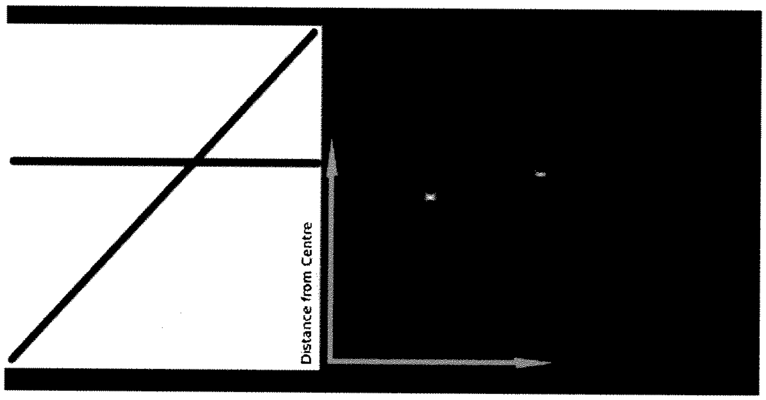
RANSAC

- 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

Hough Transform

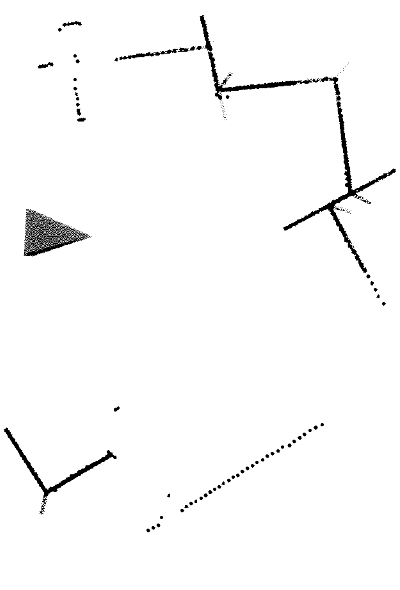
- Each point is evidence for a set of lines
  - Vote for each of the possible lines.
  - Lines with lots of evidence get many votes.



- Tends to be slow.

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 using features

- What method?
  - RANSAC!
  - Randomly guess enough correspondences to fit a model
  - Pick the model with the best consensus score.
- Can incorporate *negative* information
  - Penalize consensus if features are missing.

Aligning scans *without* features

- Can we align the individual lidar points?
  - Avoids potential for introducing error due to feature extractor problems
  - Works in any environment (?)
    - Though feature extraction can act as a filter
    - Tree detector rejects ground strikes in Victoria Park

Iterative Closest Point (ICP)

- until converged:
  - For each point in scan A, find the closest point in 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.

Iterative Closest Line (ICL)

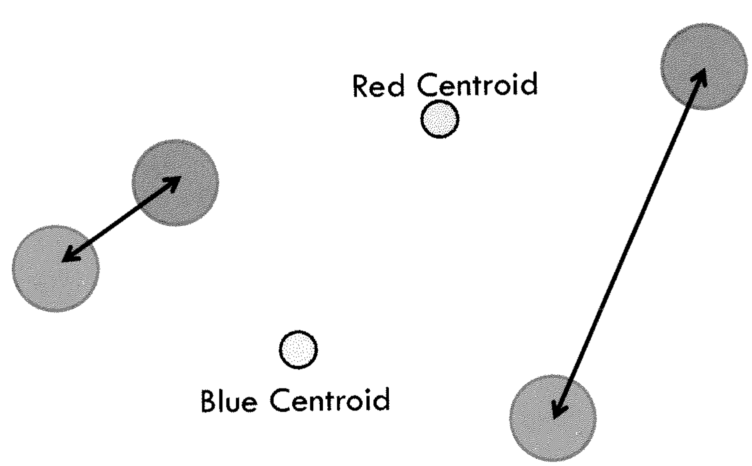
- LIDARs sample space at essentially arbitrary locations.
  - No reason to think that the exact points sampled in scan A were observed in scan B
- Solution: interpolate lines between points in scan B, find closest line instead of closest point.

Computing RBT from point correspondences

- We need to compute the rigid-body transformation
- 3DOFs:
    - Translation in x, y
    - Rotation
- Approaches:
    - Simple (bad) two point method
    - Optimal n point method

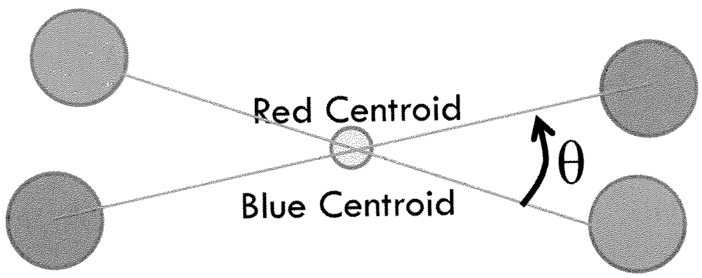
Naïve two point method (1/2)

- Compute translation component by aligning centroids



Naive two-point method (2/2)

- Now, rotate (so that lines between points have same angle)



General N-point method

- From Horn, "Closed-form solution of absolute orientation using unit quaternions"
- Actually a 3D method, but 2D special case is particularly easy.
- Formulation
  - Minimize mean squared error between corresponding points
  - Potential weakness: outliers

Optimal N-point method

- Assume that points x and points y are centered at zero. (This assumption can be patched up later

$$e_i = y_i - R(x_i) - t$$

$$y \approx R(x) + t \qquad\qquad R(x_i) = \begin{bmatrix} \cos(\theta)x_{i_0} - \sin(\theta)x_{i_1} \\ \sin(\theta)x_{i_0} + \cos(\theta)x_{i_1} \end{bmatrix}$$

$$\begin{aligned} \chi^2 &= \sum e_i^T e_i \\ &= \sum y_i^T y_i - R(x_i)^T R(x_i) + t^T t - 2y_i^T R(x_i) - 2y_i^T t + 2t^T R(x_i) \end{aligned}$$

Optimal Translation

- Let's compute the optimal Translation

$$\begin{aligned} \chi^2 &= \sum e_i^T e_i \\ &= \sum y_i^T y_i - R(x_i)^T R(x_i) + t^T t - 2y_i^T R(x_i) - 2y_i^T t + 2t^T R(x_i) \end{aligned}$$

$$\frac{\partial \chi^2}{\partial t} = \sum 2t - 2y_i + 2R(x_i) = 0$$

- Recall: we assumed points x & points y are centered at origin. i.e.,

$$\sum y_i = \sum R(x_i) = 0$$

- And thus:

$$t = 0$$

- (i.e., the optimal translation is the one that aligns their centroids.)

Optimal Rotation

- Let's compute the optimal rotation

$$\begin{aligned}\chi^2 &= \sum e_i^T e_i \\ &= \sum y_i^T y_i - R(x_i)^T R(x_i) + t^T t - 2y_i^T R(x_i) - 2y_i^T t + 2t^T R(x_i)\end{aligned}$$

- Letting t = 0 per previous slide:

$$\chi^2 = \sum y_i^T y_i - R(x_i)^T R(x_i) - 2y_i^T R(x_i)$$

- Note that y'y and R(x)'R(x) are the lengths of the vectors, and are independent of the rotation. We can thus drop those terms.

Optimal Rotation

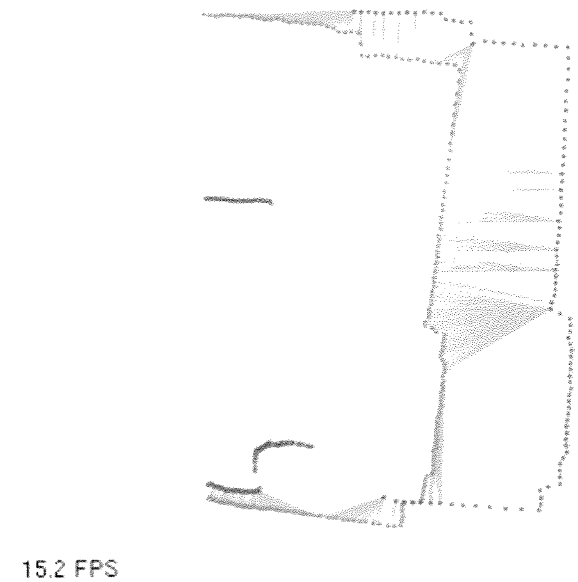
- Recall that: $R(x_i) = \begin{bmatrix} \cos(\theta)x_{i_0} - \sin(\theta)x_{i_1} \\ \sin(\theta)x_{i_0} + \cos(\theta)x_{i_1} \end{bmatrix}$

$$\begin{aligned}\chi^2 &= \sum y_i^T R(x_i) \\ &= \sum \cos(\theta)x_{i_0}y_{i_0} - \sin(\theta)x_{i_1}y_{i_0} + \sin(\theta)x_{i_0}y_{i_1} + \cos(\theta)x_{i_1}y_{i_1}\end{aligned}$$

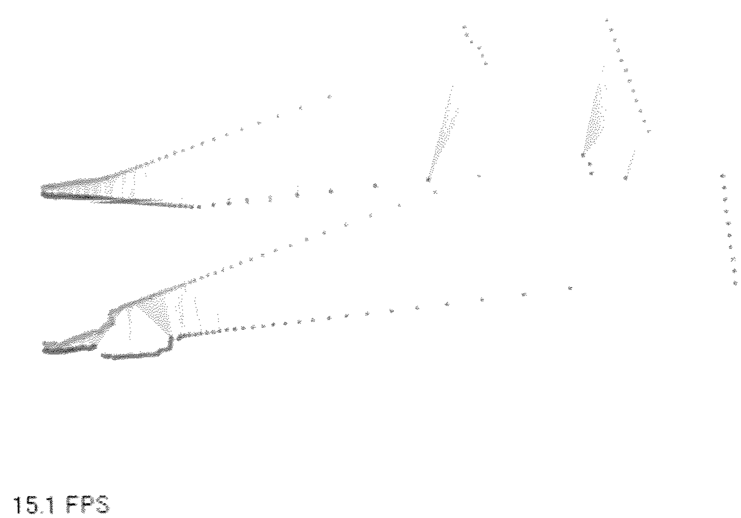
- Collect sin/cos terms, we can write:

$$\begin{aligned}\chi^2 &= M\sin(\theta) + N\cos(\theta) \\ \frac{\partial\chi^2}{\partial\theta} &= M\cos(\theta) - N\sin(\theta) = 0 \\ \theta &= \mathrm{atan2}(M, N)\end{aligned}$$

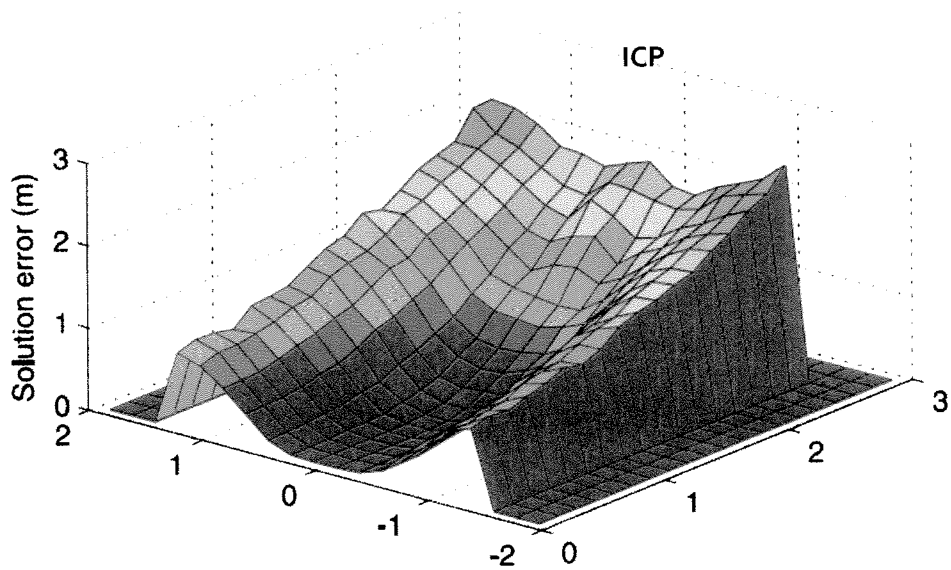
ICP In Action



ICP In Action (2)



ICP Performance



Better point-wise matching

- Next time!

Features vs. Points

- Features (pros)
  - Data reduction
  - Noise filtering
  - Extraction + Matching is often fast
- Non-Feature Matching (pros)
  - General purpose: don't require world to contain our features
  - Very robust