Visual Navigation for Flying Robots

Simultaneous Localization and Mapping

Dr. Jürgen Sturm

Agenda for Today

- Outlier rejection using RANSAC
- Laser-based motion estimation
- The SLAM problem
- Pose graph SLAM
- Map optimization

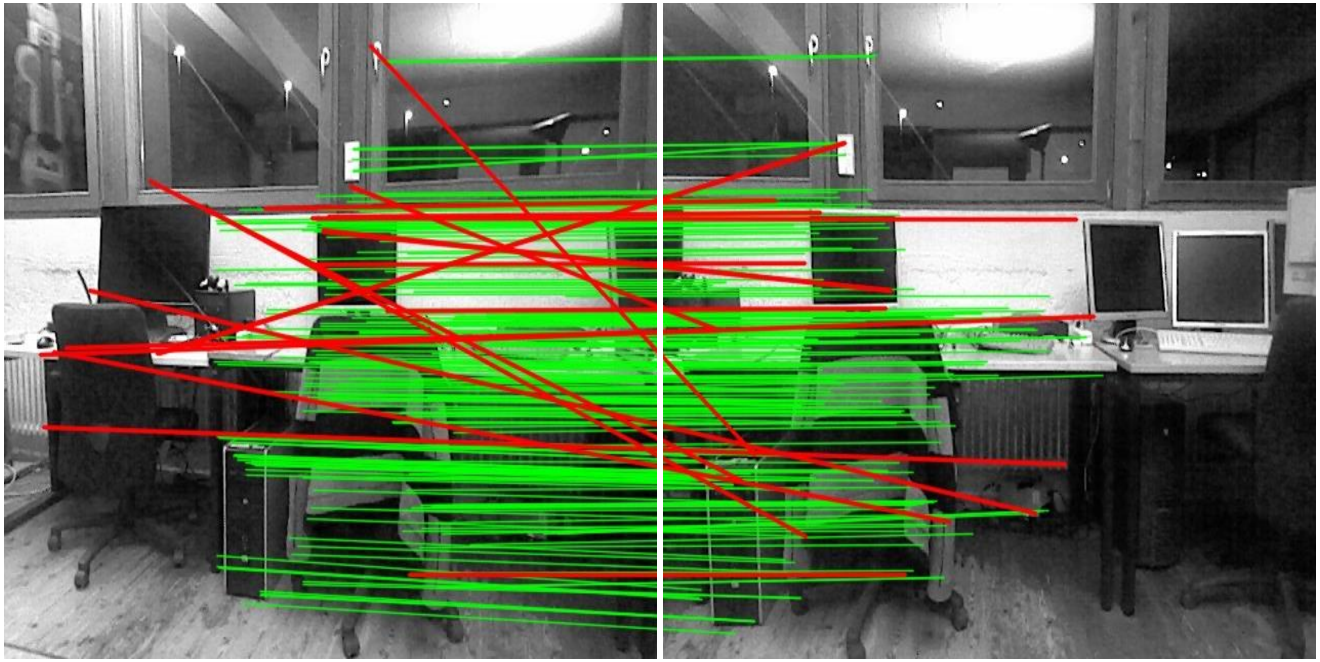
Remember: 8-Point Algorithm

Given: Image pair


Find: Camera motion R,t (up to scale)

- Compute correspondences
- Compute essential matrix
- Extract camera motion

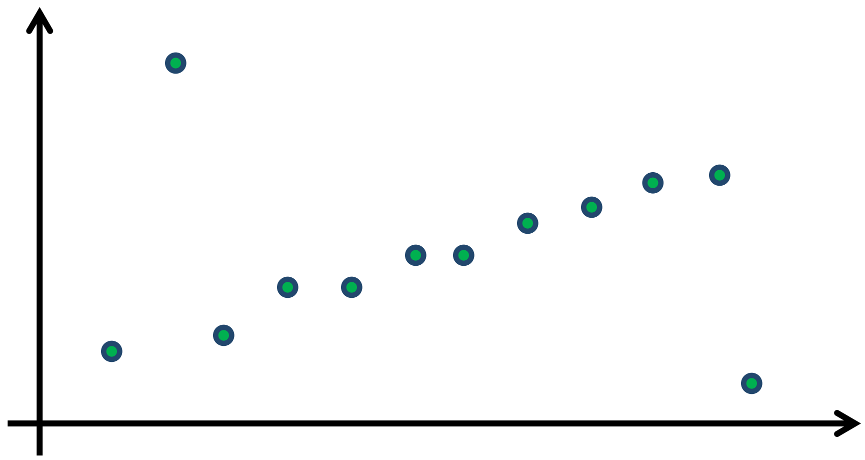
How To Deal With Outliers?



Problem: No matter how good the feature descriptor/matcher is, there is always a chance for bad point correspondences (=outliers)

Robust Estimation

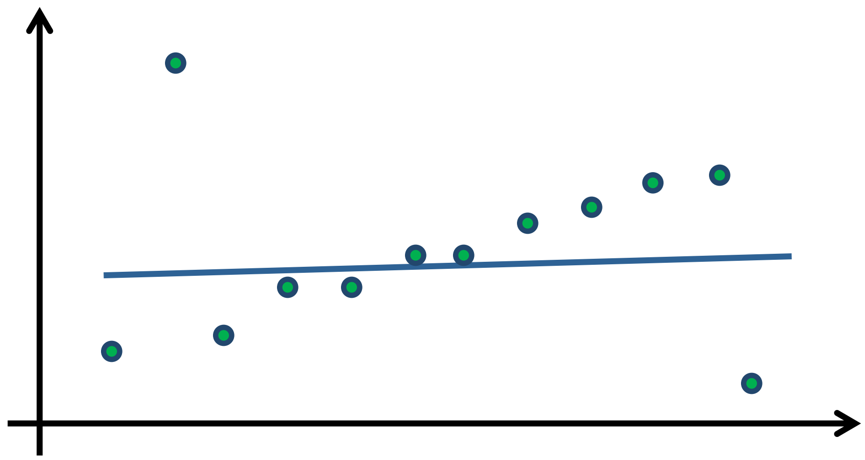
Example: Fit a line to 2D data containing outliers



- Input data is a mixture of
  - Inliers (perturbed by Gaussian noise)
  - Outliers (unknown distribution)
- Let's fit a line using least squares…

Robust Estimation

Example: Fit a line to 2D data containing outliers



- Input data is a mixture of
  - Inliers (perturbed by Gaussian noise)
  - Outliers (unknown distribution)
- Least squares fit gives poor results!

RANdom SAmple Consensus (RANSAC)

[Fischler and Bolles, 1981]

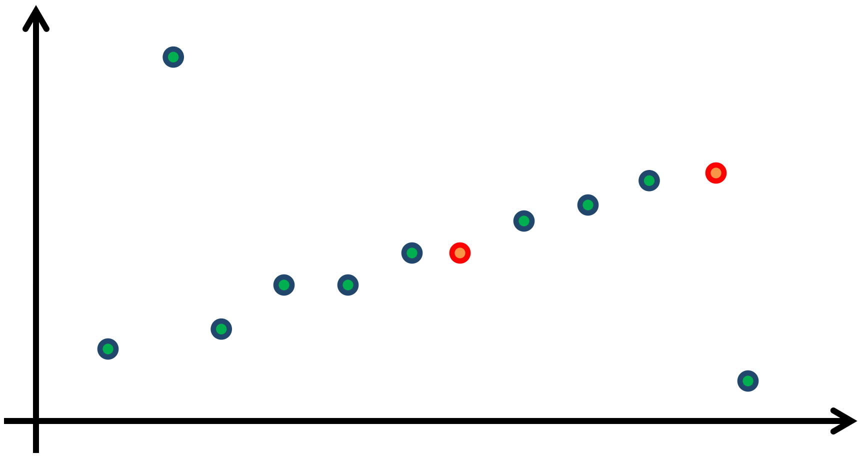
Goal: Robustly fit a model to a data set S which contains outliers

Algorithm:

1. Randomly select a (minimal) subset
2. Instantiate the model from it
3. Using this model, classify the all data points as inliers or outliers
4. Repeat 1-3 for N iterations
5. Select the largest inlier set, and re-estimate the model from all points in this set

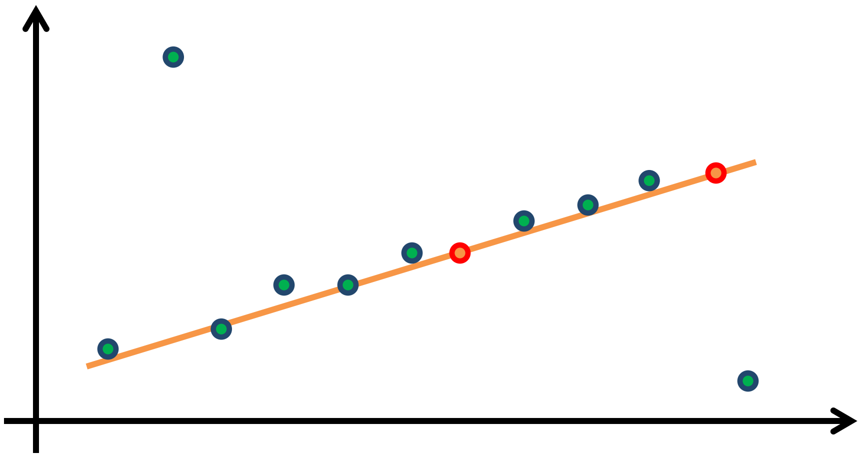
Example

- Step 1: Sample a random subset



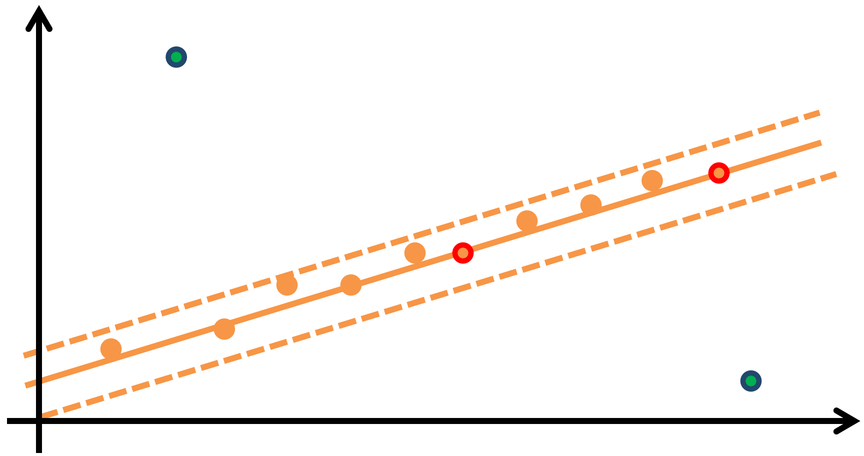
Example

- Step 2: Fit a model to this subset



Example

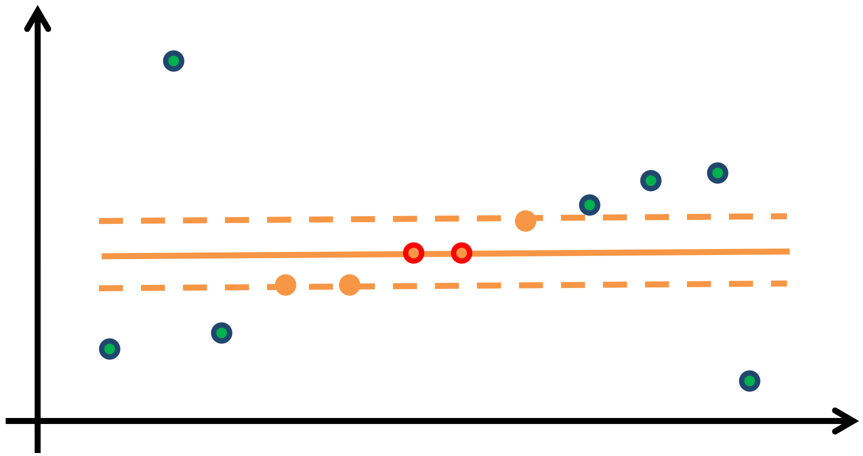
- Step 3: Classify points as inliers and outliers (e.g., using a threshold distance)



→ 10 inliers, 2 outliers

Example

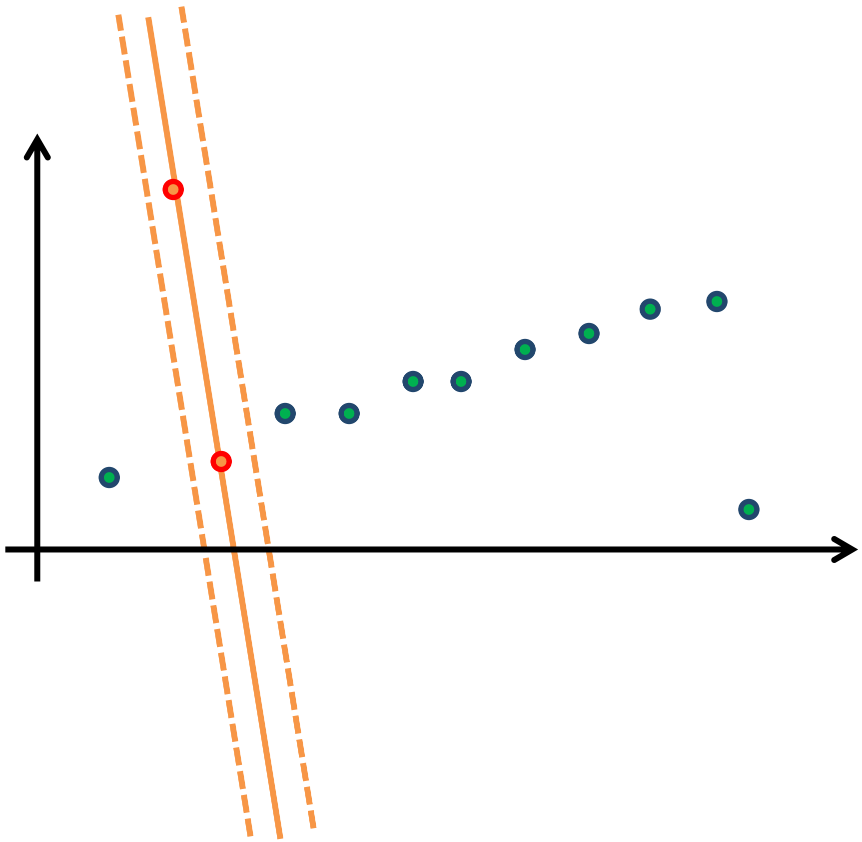
- Step 4: Repeat steps 1-3 for N iterations



Iteration 2:
→ 5 inliers, 7 outliers

Example

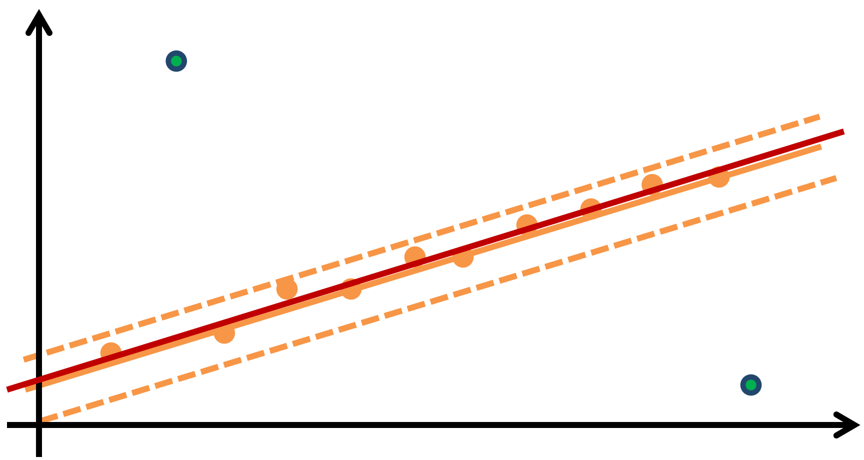
- Step 4: Repeat steps 1-3 for N iterations



Iteration 3:
→ 2 inliers, 10 outliers

Example

- Step 5: Select the best model (most inliers), the re-fit model using all inliers



Best model:
Iteration 1
(10 inliers, 2 outliers)

How Many Iterations Do We Need?

- For a probability of success p, we need

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$$ iterations

for subset size s and outlier ratio ϵ

- E.g., for p=0.99:

	Required points s	Outlier ratio ε						
		10 %	20 %	30 %	40 %	50 %	60 %	70 %
Line	2	3	5	7	11	17	27	49
Plane	3	4	7	11	19	35	70	169
Essential matrix	8	9	26	78	272	1177	7025	70188

Summary on RANSAC

- Efficient algorithm to estimate a model from noisy and outlier-contaminated data
- RANSAC is used today very widely
- Often used in feature matching / visual motion estimation
- Many improvements/variants (e.g., PROSAC, MLESAC, …)

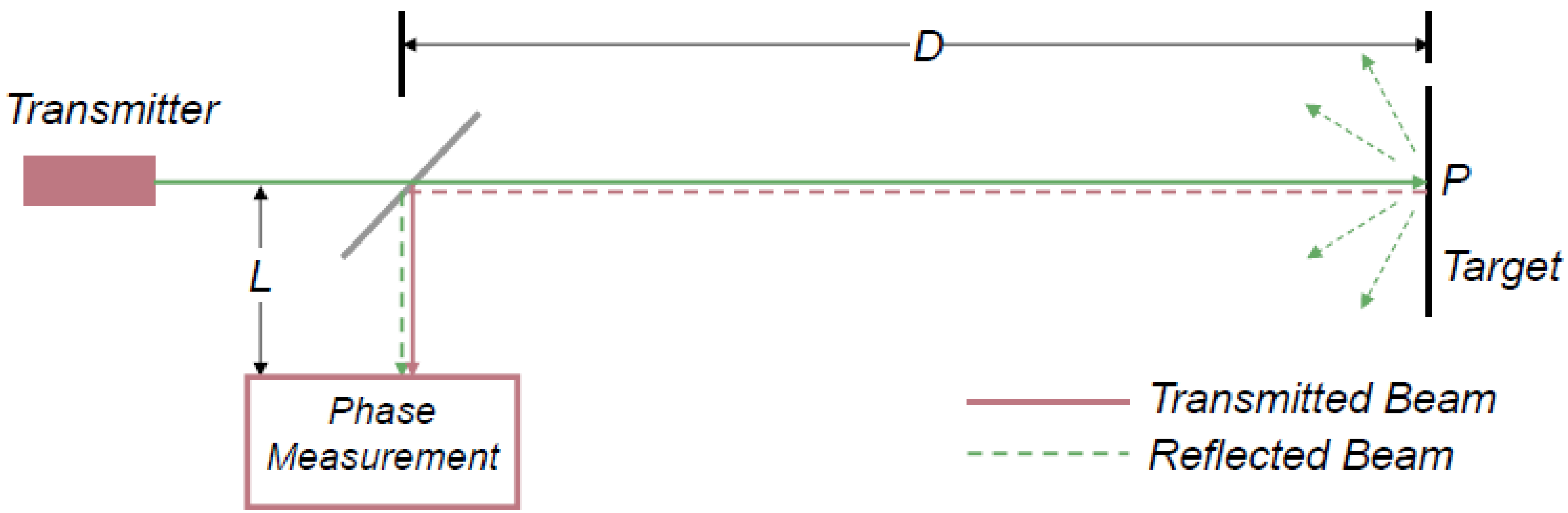
Laser-based Motion Estimation

- So far, we looked at motion estimation (and place recognition) from **visual** sensors
- Today, we cover motion estimation from **range** sensors
  - Laser scanner (laser range finder, ultrasound)
  - Depth cameras (time-of-flight, Kinect …)



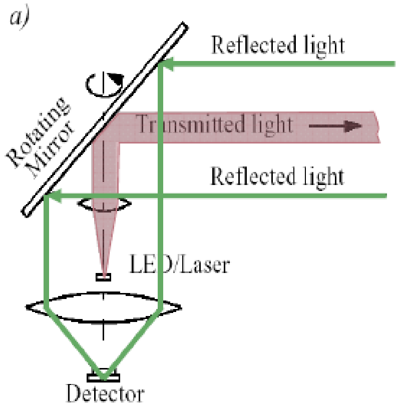
Laser Scanner

- Measures phase shift or time-of-flight
- Pro: High precision, wide field of view, safety approved for collision detection
- Con: Relatively expensive + heavy

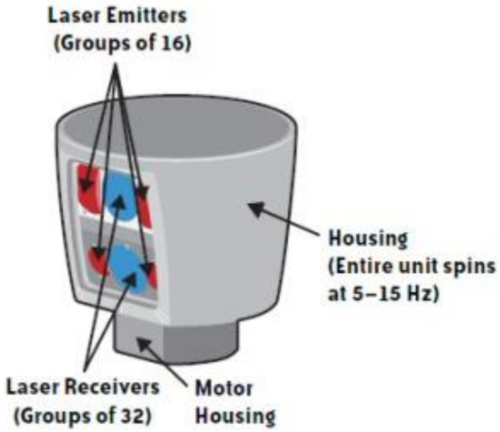


Laser Scanner

- 2D scanners



- 3D scanners



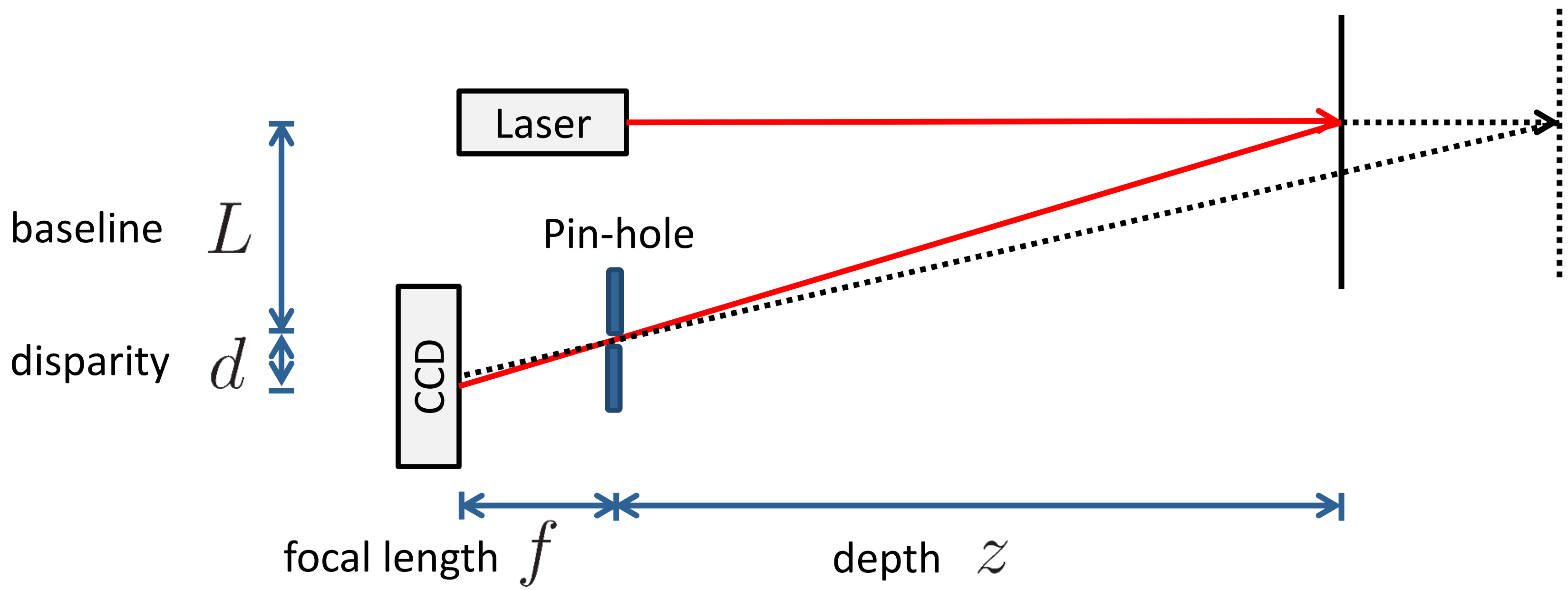
Laser Triangulation

Idea:

- Well-defined light pattern (e.g., point or line) projected on scene
- Observed by a line/matrix camera or a position-sensitive device (PSD)
- Simple triangulation to compute distance

Laser Triangulation

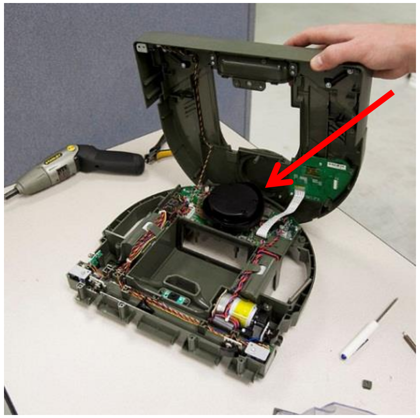
- Function principle

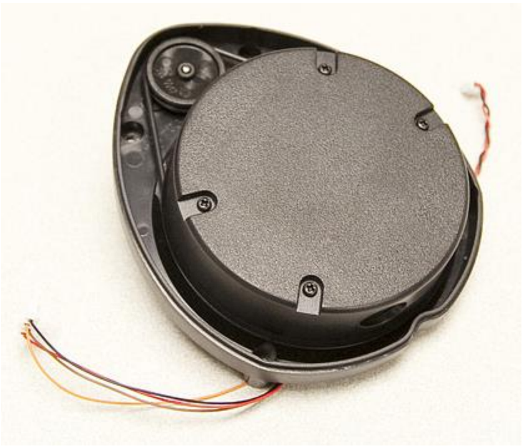


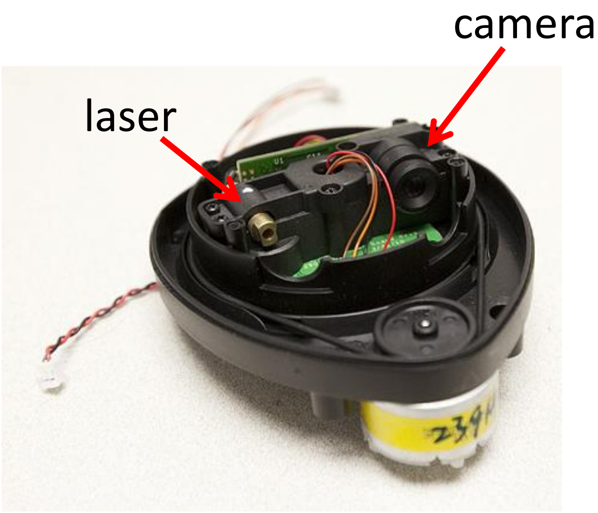
- Depth triangulation $z = f\frac{L}{d}$

Example: Neato XV-11

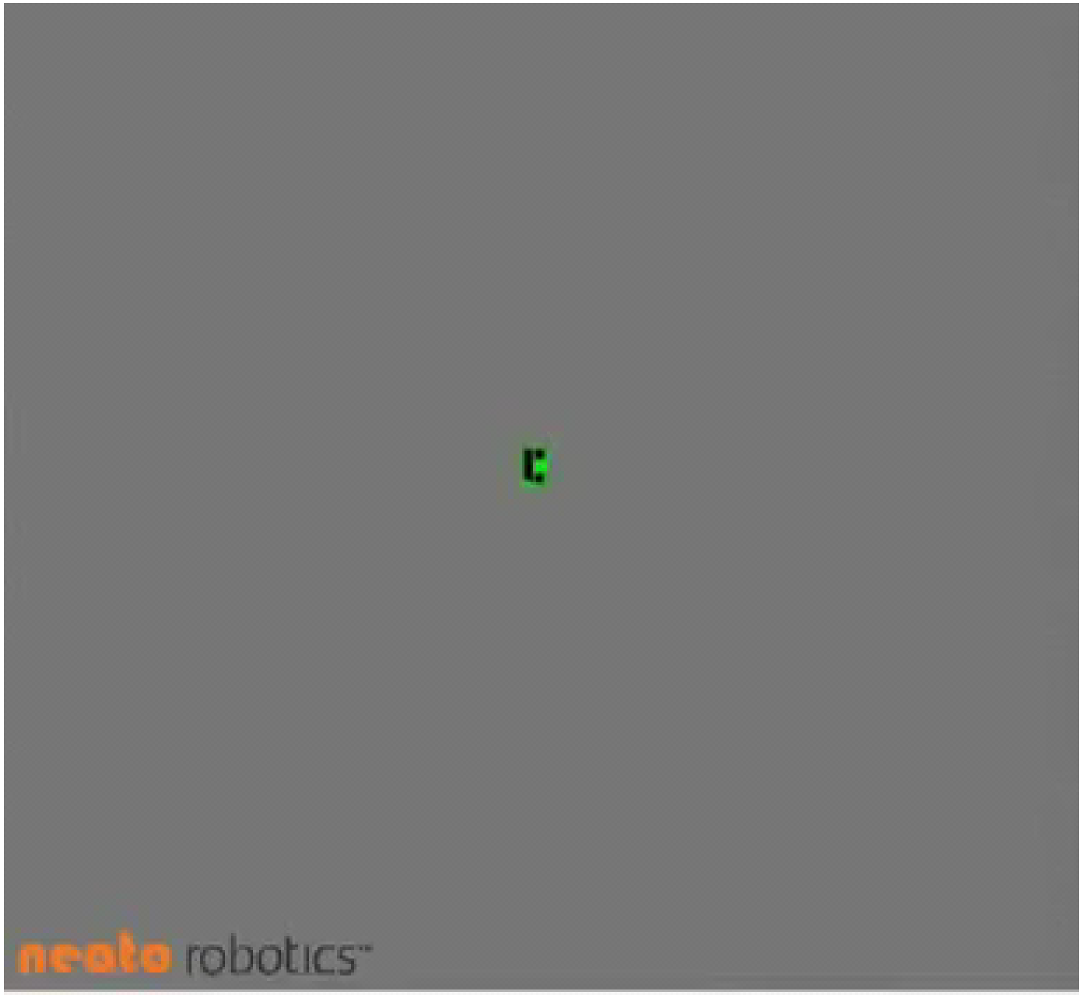
- K. Konolige, “A low-cost laser distance sensor”, ICRA 2008
- Specs: 360deg, 10Hz, 30 USD







How Does the Data Look Like?



Laser Scanner

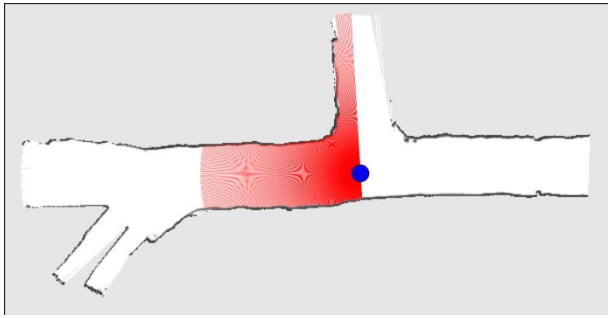
- Measures angles and distances to closest obstacles

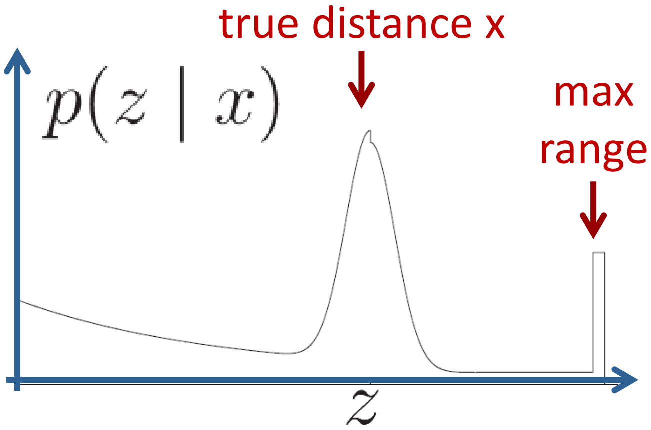
$$\mathbf{z} = (\theta_1, z_1, \dots, \theta_n, z_n) \in \mathbb{R}^{2n}$$

- Alternative representation: 2D point set (cloud)

$$\mathbf{z} = (x_1, y_1, \dots, x_n, y_n)^\top \in \mathbb{R}^{2n}$$

- Probabilistic sensor model $p(z \mid x)$





Laser-based Motion Estimation

How can we best align two laser scans?

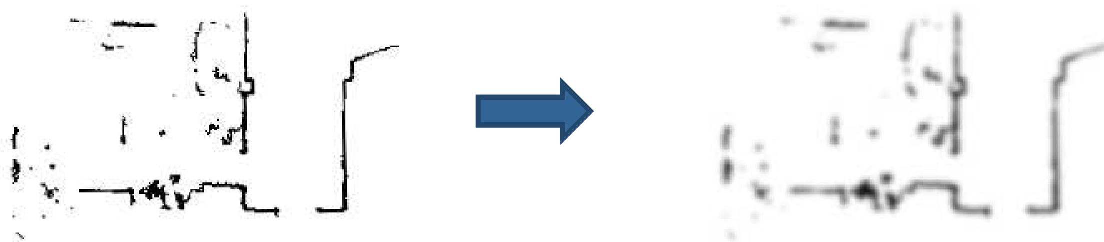
Laser-based Motion Estimation

How can we best align two laser scans?

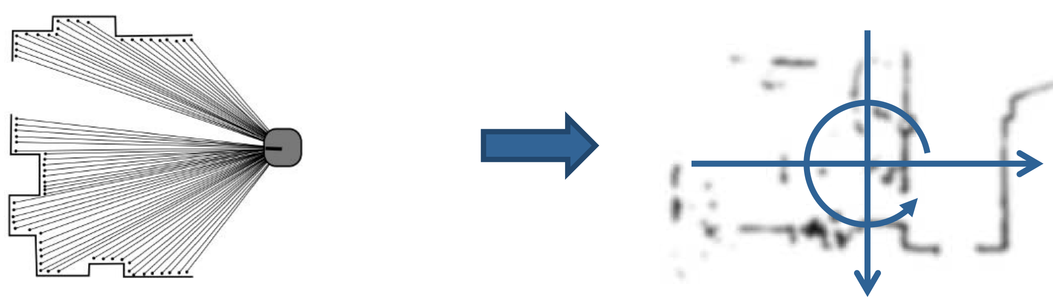
- Exhaustive search
- Iterative minimization (ICP)

Exhaustive Search

- Estimate a map using first scan and sensor model

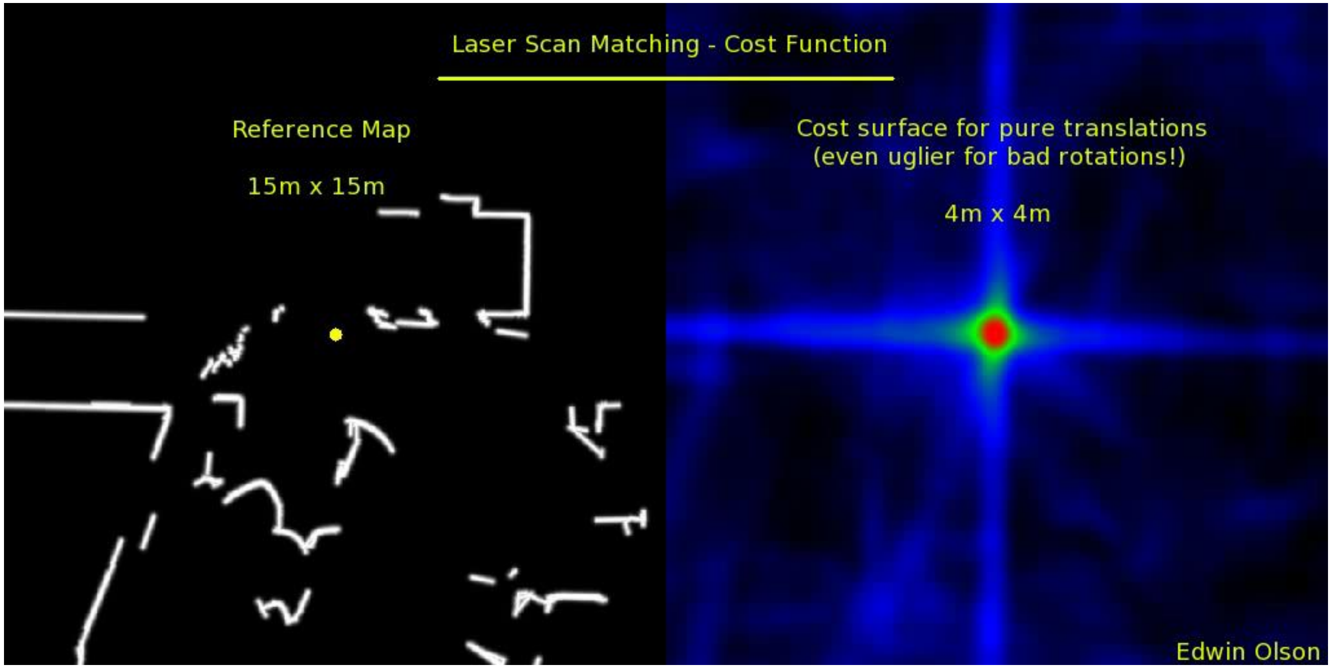


- Sweep second scan over map, compute correlation and select best pose

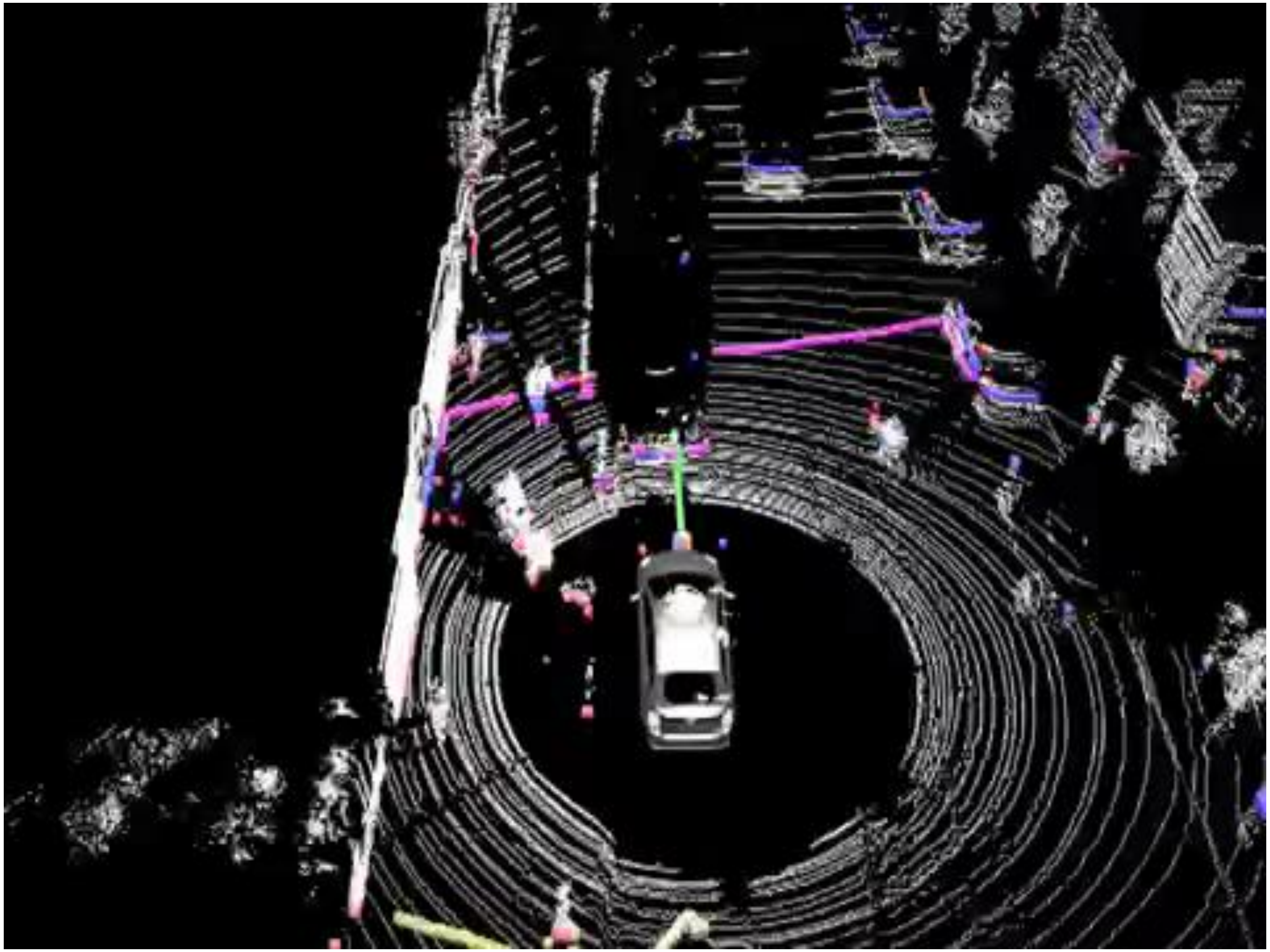


Example: Exhaustive Search [Olson, ICRA '09]

- Multi-resolution correlative scan matching
- Real-time by using GPU
- Remember: SE(2) has 3 DOFs



Does Exhaustive Search Generalize To 3D As Well?



Iterative Closest Point (ICP)

- **Given:** Two corresponding point sets (clouds)

$$P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$$
$$Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$$

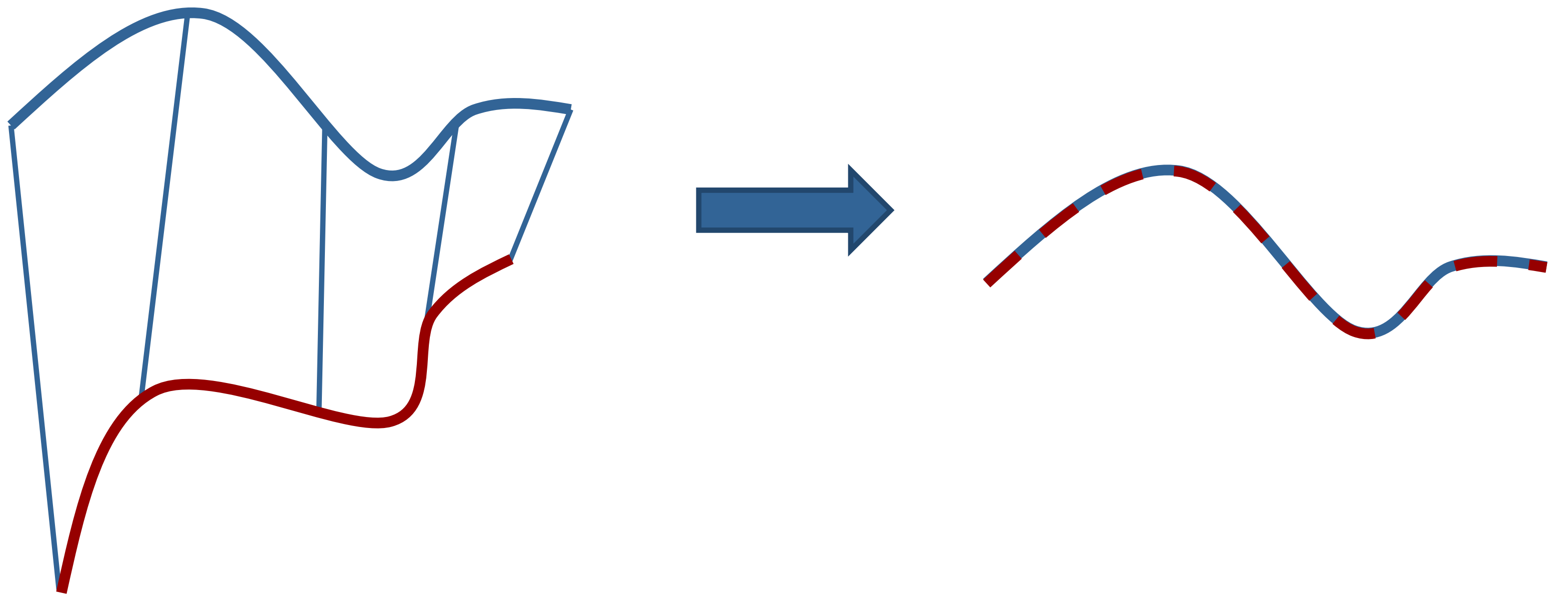
- **Wanted:** Translation $\mathbf{t}$ and rotation R that minimize the sum of the squared error

$$E(R, \mathbf{t}) = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{p}_i - R\mathbf{q}_i - \mathbf{t}||^2$$

where $\mathbf{p}_i$ and $\mathbf{q}_i$ are corresponding points

Known Correspondences

Note: If the correct correspondences are known, both rotation and translation can be calculated in **closed form**.



Known Correspondences

- **Idea:** The center of mass of both point sets has to match

$$\bar{\mathbf{p}} = \frac{1}{n} \sum_i \mathbf{p}_i \qquad \bar{\mathbf{q}} = \frac{1}{n} \sum_i \mathbf{q}_i$$

- Subtract the corresponding center of mass from every point
- Afterwards, the point sets are zero-centered, i.e., we only need to recover the rotation…

Known Correspondences

- Decompose the matrix

$$W = \sum_i (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{q}_i - \bar{\mathbf{q}})^\top = USV^\top$$

  using singular value decomposition (SVD)

- **Theorem**

  If $\operatorname{rank} W = 3$, the optimal solution of $E(R, \mathbf{t})$ is unique and given by

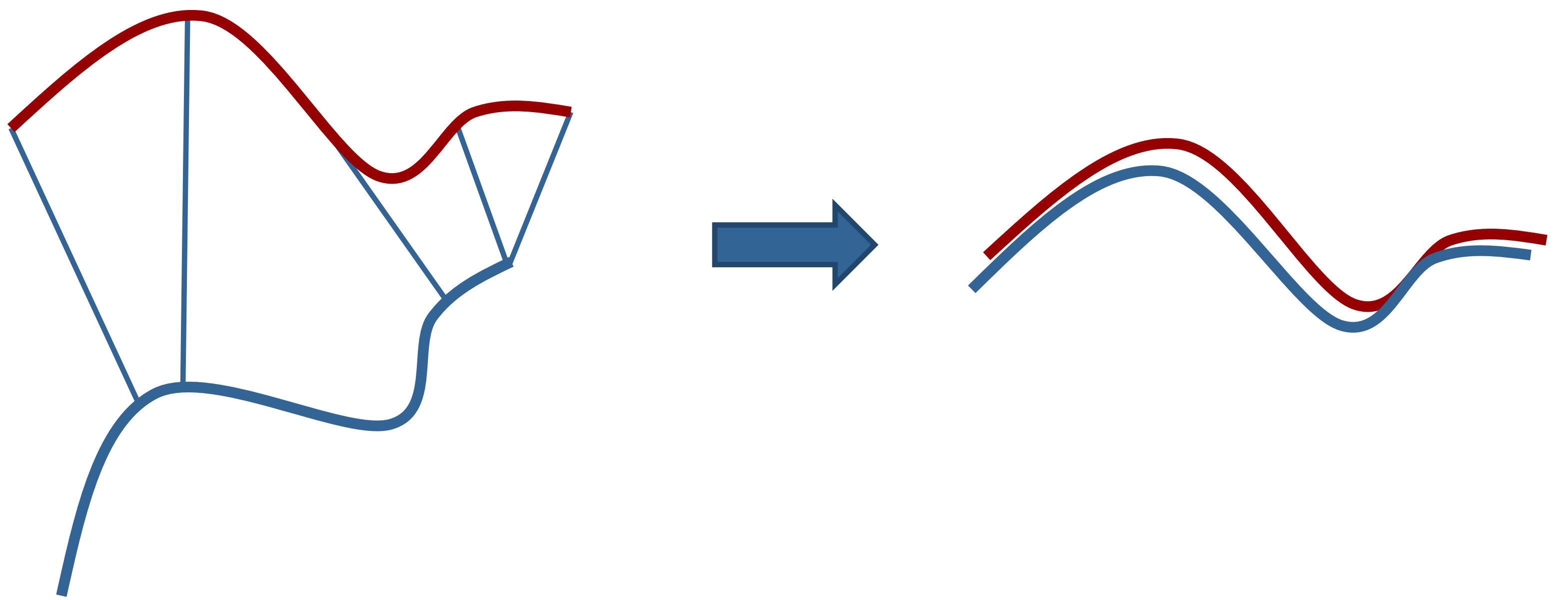
$$R = UV^\top$$

$$\mathbf{t} = \bar{\mathbf{p}} - R\bar{\mathbf{q}}$$

(for proof, see http://hss.ulb.uni-bonn.de/2006/0912/0912.pdf, p.34/35)

Unknown Correspondences

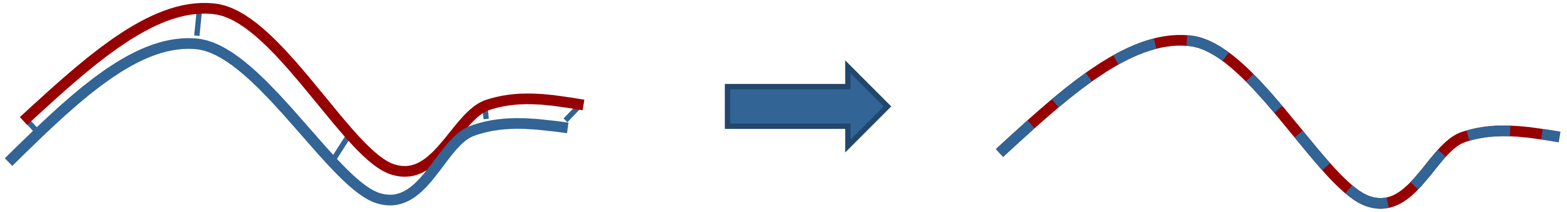
- If the correct correspondences are not known, it is generally impossible to determine the optimal transformation in one step



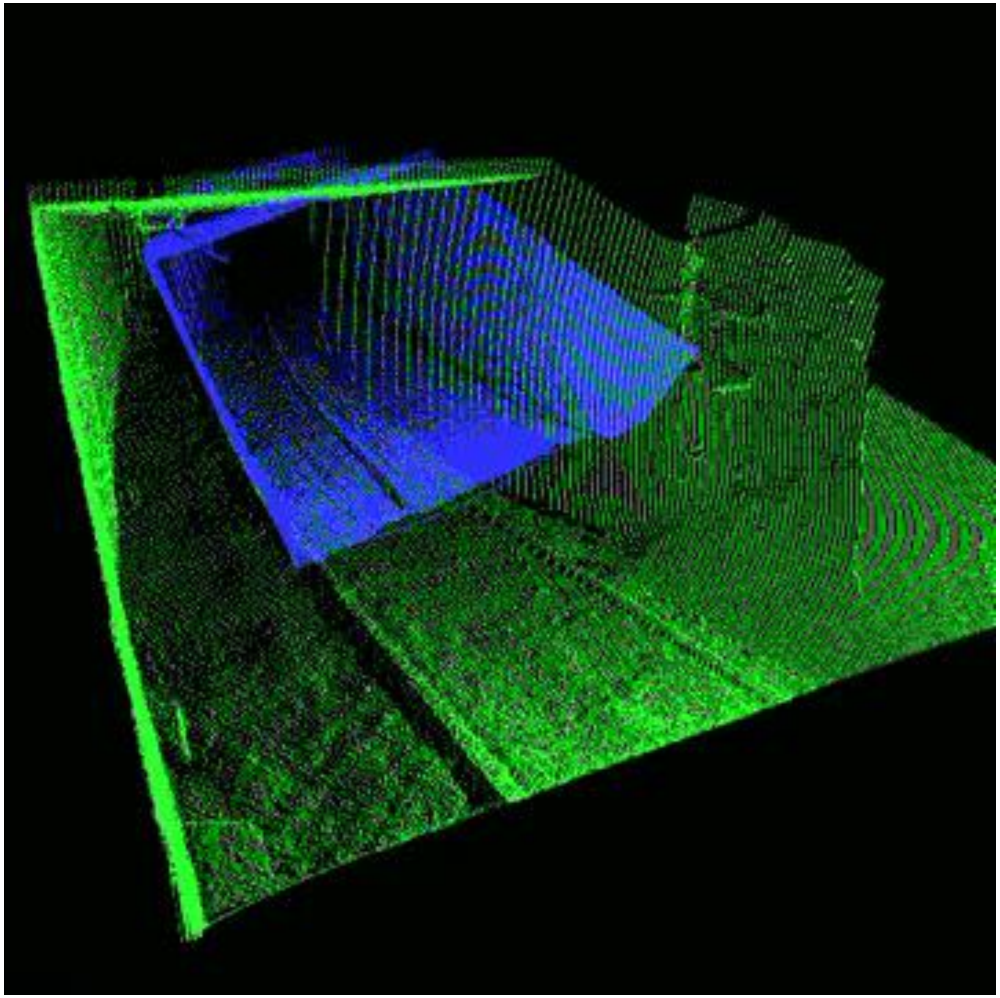
ICP Algorithm

[Besl & McKay, 92]

- **Algorithm:** Iterate until convergence
  - Find correspondences
  - Solve for R,t
- Converges if starting position is "close enough"



Example: ICP



ICP Variants

Many variants on all stages of ICP have been proposed:

- **Selecting** and **weighting** source points
- **Finding** corresponding points
- Rejecting certain (outlier) correspondences
- Choosing an **error metric**
- **Minimization**

Performance Criteria

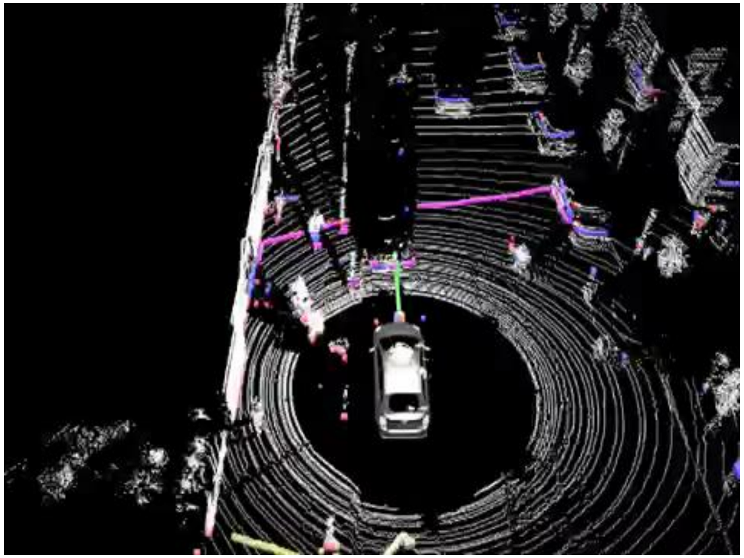
- Various aspects of performance
  - Speed
  - Stability (local minima)
  - Tolerance w.r.t. noise and/or outliers
  - Basin of convergence (maximum initial misalignment)
- Choice depends on data and application

Selecting Source Points

- Use all points
- Random sampling
- Spatially uniform sub-sampling
- Feature-based sampling

Spatially Uniform Sampling

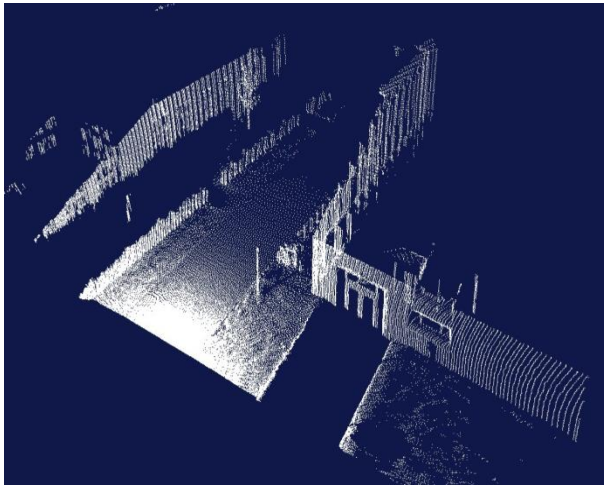
- Density of points usually depends on the distance to the sensor → no uniform distribution
- Can lead to a bias in ICP



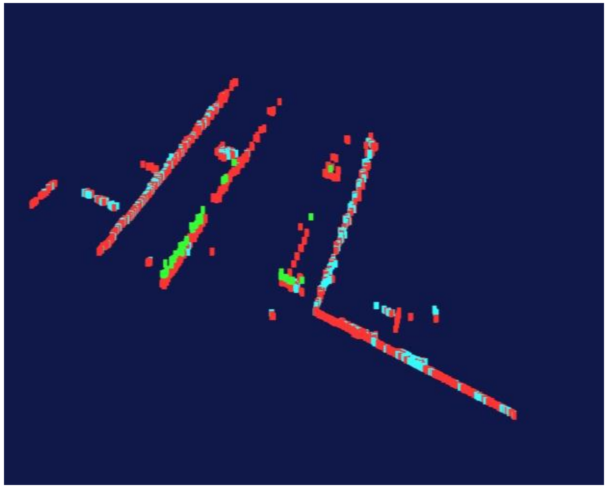
Feature-based Sampling

Detect interest points (same as with images)

- Decrease the number of correspondences
- Increase efficiency and accuracy
- Requires pre-processing



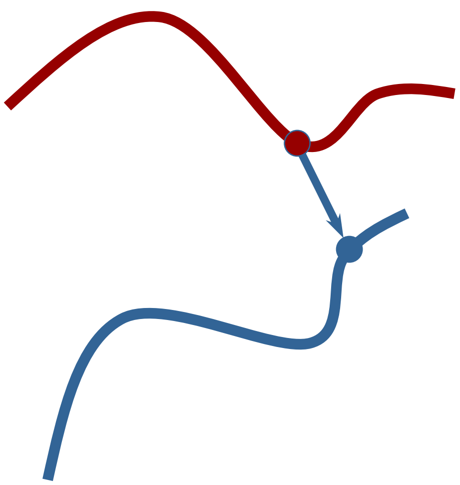
3D Scan (~200.000 Points)



Extracted Features (~5.000 Points)

Closest Point Matching

- Find closest point in the other point set
- Distance threshold



- Closest-point matching generally stable, but slow

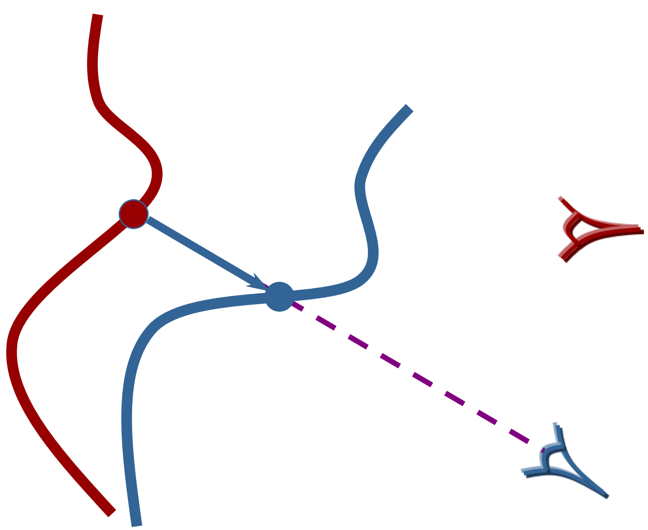
Speeding Up Correspondence Search

Finding closest point is most expensive stage of the ICP algorithm

- Build index for one point set (kd-tree)
- Use simpler algorithm (e.g., projection-based matching)

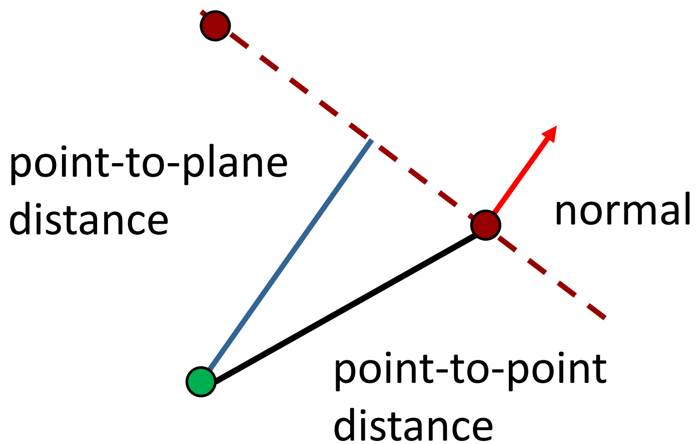
Projection-based Matching

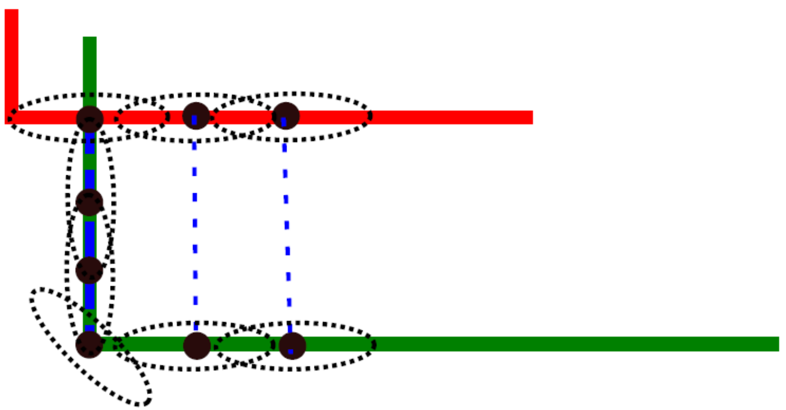
- Slightly worse performance per iteration
- Each iteration is one to two orders of magnitude faster than closest-point
- Requires point-to-plane error metric



Error Metrics

- Point-to-point
- Point-to-plane lets flat regions slide along each other





- Generalized ICP: Assign individual covariance to each data point [Segal, RSS 2009]

Minimization

- Only point-to-point metric has closed form solution(s)
- Other error metrics require non-linear minimization methods

Example: Real-Time ICP on Range Images

[Rusinkiewicz and Levoy, 2001]

- Real-time scan alignment
- Range images from structure light system (projector and camera, temporal coding)

ICP: Summary

- ICP is a powerful algorithm for calculating the displacement between point clouds
- The overall speed depends most on the choice of matching algorithm
- ICP is (in general) only locally optimal → can get stuck in local minima

The SLAM Problem

SLAM is the process by which a robot **builds a map** of the environment and, at the same time, uses the map to **compute its location**:

- Localization: inferring location given a map
- Mapping: inferring a map given a location

The acronym SLAM stands for "simultaneous localization and mapping".

The SLAM Problem

Given:

- The robot's controls $\mathbf{u}_{1:t} =< \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_t >$
- (Relative) observations $\mathbf{z}_{1:t} =< \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_t >$

Wanted:

- Map of features $\mathbf{m} =< \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k >$
- Trajectory of the robot $\mathbf{x}_{1:t} =< \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t >$

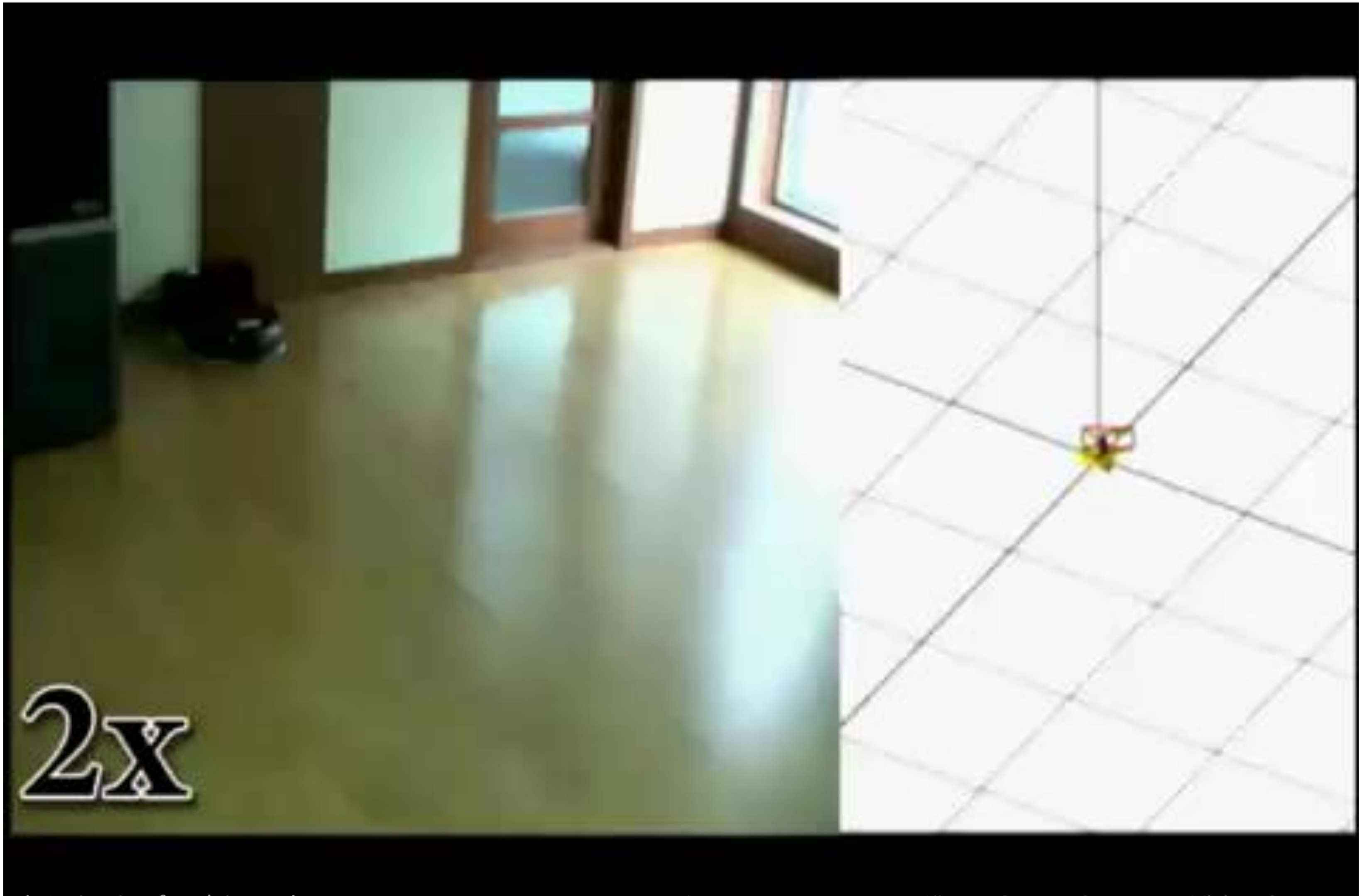
SLAM Applications

SLAM is central to a range of indoor, outdoor, in-air and underwater applications for both unmanned and autonomous vehicles.

Examples

- At home: vacuum cleaner, lawn mower
- Air: inspection, transportation, surveillance
- Underwater: reef/environmental monitoring
- Underground: search and rescue
- Space: terrain mapping, navigation

SLAM with Ceiling Camera (Samsung Hauzen RE70V, 2008)



Localization, Path planning, Coverage (Neato XV11, $300)

SfM vs. SLAM

- Structure from Motion (SfM)
  - Monocular/stereo camera
  - Sometimes uncalibrated sensors (e.g., Flickr images)
- Simultaneous Localization and Mapping (SLAM)
  - Multiple sensors: Laser scanner, ultrasound, monocular/stereo camera, GPS, …
  - Typically in combination with an odometry sensor
  - Typically pre-calibrated sensors

Remember: 3D Transformations

- Representation as a homogeneous matrix

$$M = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \in \mathrm{SE}(3) \subset \mathbb{R}^{4\times 4}$$

Pro: easy to concatenate and invert
Con: not minimal

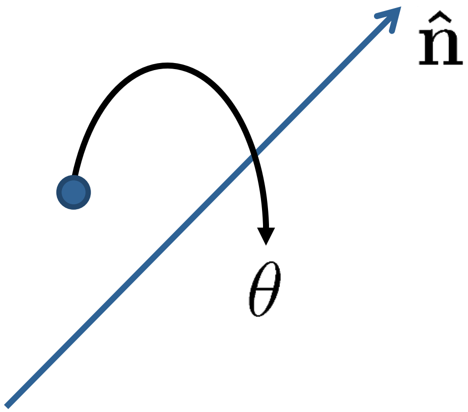
- Representation as a twist coordinates

$$\boldsymbol{\xi} = (\underbrace{\omega_x \quad \omega_y \quad \omega_z}_{\text{angular velocity}} \quad \underbrace{v_x \quad v_y \quad v_z}_{\text{linear velocity}})^\top \in \mathbf{R}^6$$

Pro: minimal
Con: need to convert to matrix for concatenation and inversion

Remember: 3D Rotation as Axis/Angle

- Represent rotation by
  - rotation axis $\hat{\mathbf{n}}$ and
  - rotation angle θ
- 4 parameters $(\hat{\mathbf{n}}, \theta)$
- 3 parameters $\boldsymbol{\omega} = \theta \hat{\mathbf{n}}$
  - length is rotation angle
  - also called the angular velocity
  - minimal representation



Remember: 3D Transformations

- From twist coordinates to twist

$$\hat{\boldsymbol{\xi}} = \begin{pmatrix} 0 & -\omega_z & \omega_y & v_x \\ \omega_z & 0 & -\omega_x & v_y \\ -\omega_y & \omega_x & 0 & v_z \\ 0 & 0 & 0 & 0 \end{pmatrix} \in \mathrm{se}(3)$$

- Exponential map between se(3) and SE(3)

$$M = \exp \hat{\boldsymbol{\xi}} \qquad \hat{\boldsymbol{\xi}} = \log M$$

(or compute using Rodriguez' formula)

Notation

- Camera poses in a minimal representation (e.g., twists)

$$\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n$$

- … as transformation matrices

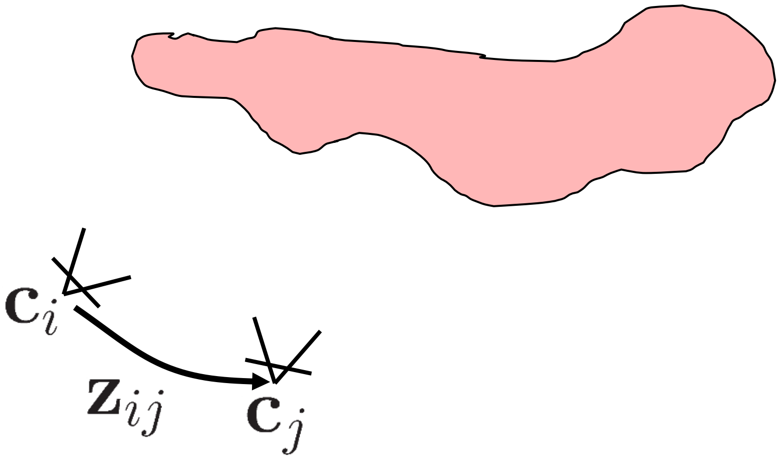
$$M_1, M_2, \ldots, M_n$$

- … as rotation matrices and translation vectors

$$(R_1, \mathbf{t}_1), (R_2, \mathbf{t}_2), \ldots, (R_n, \mathbf{t}_n)$$

Incremental Motion Estimation

- **Idea:** Estimate camera motion from frame to frame

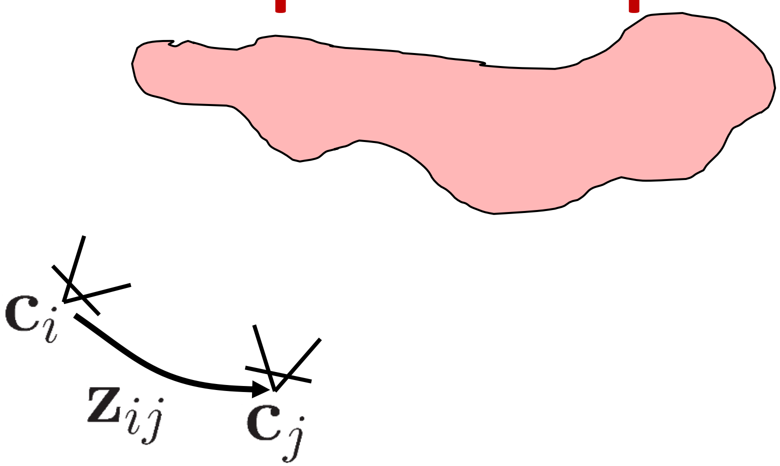


Incremental Motion Estimation

- **Idea:** Estimate camera motion from frame to frame
- **Motion concatenation (for twists)**

$$\mathbf{c}_j = \mathbf{c}_i \oplus \mathbf{z}_{ij} = \log\left(\exp \hat{\mathbf{c}}_i \exp \hat{\mathbf{z}}_{ij}\right)$$

- **Motion composition operator (in general)**

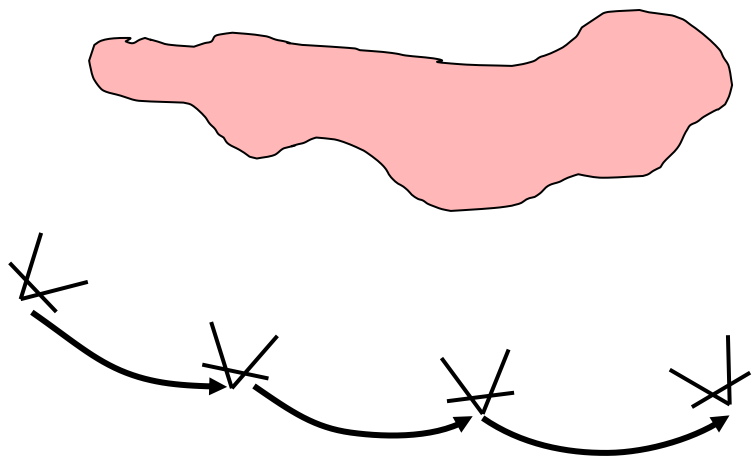


$$\mathbf{c}_j = \mathbf{c}_i \oplus \mathbf{z}_{ij}$$

$$\mathbf{z}_{ij} = \mathbf{c}_j \ominus \mathbf{c}_i$$

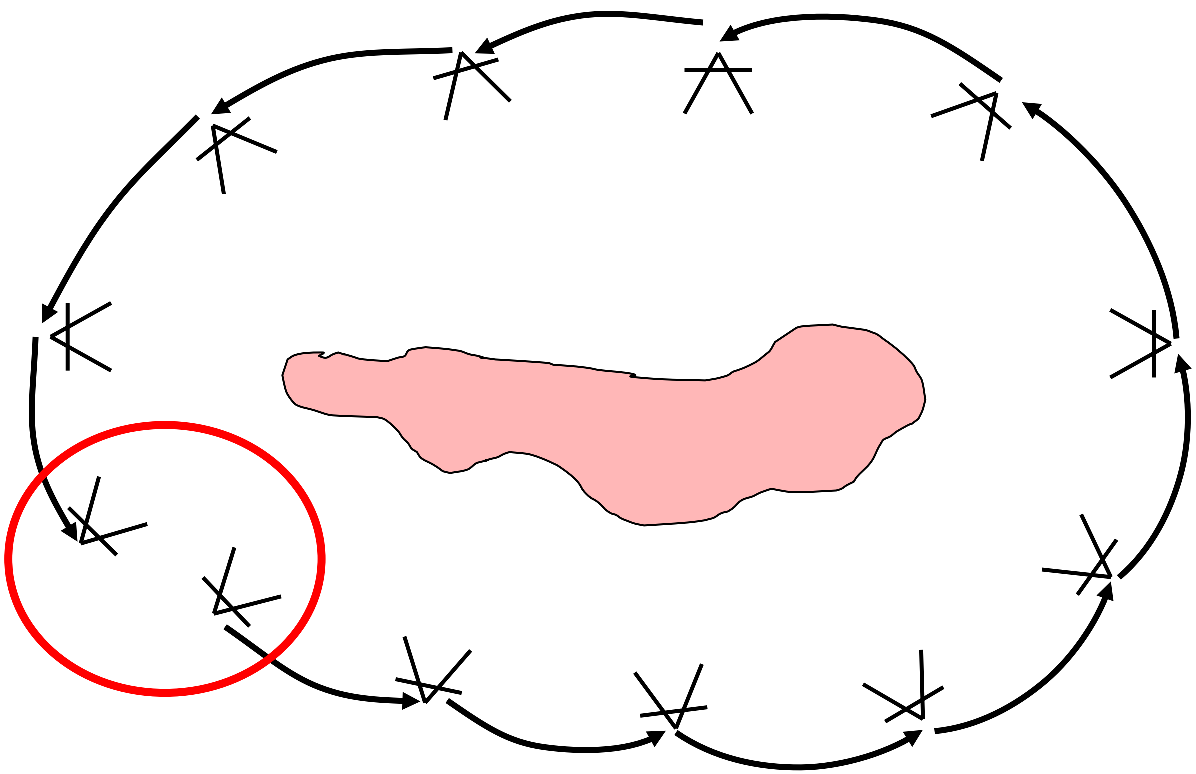
Incremental Motion Estimation

- **Idea:** Estimate camera motion from frame to frame



Incremental Motion Estimation

- **Idea:** Estimate camera motion from frame to frame

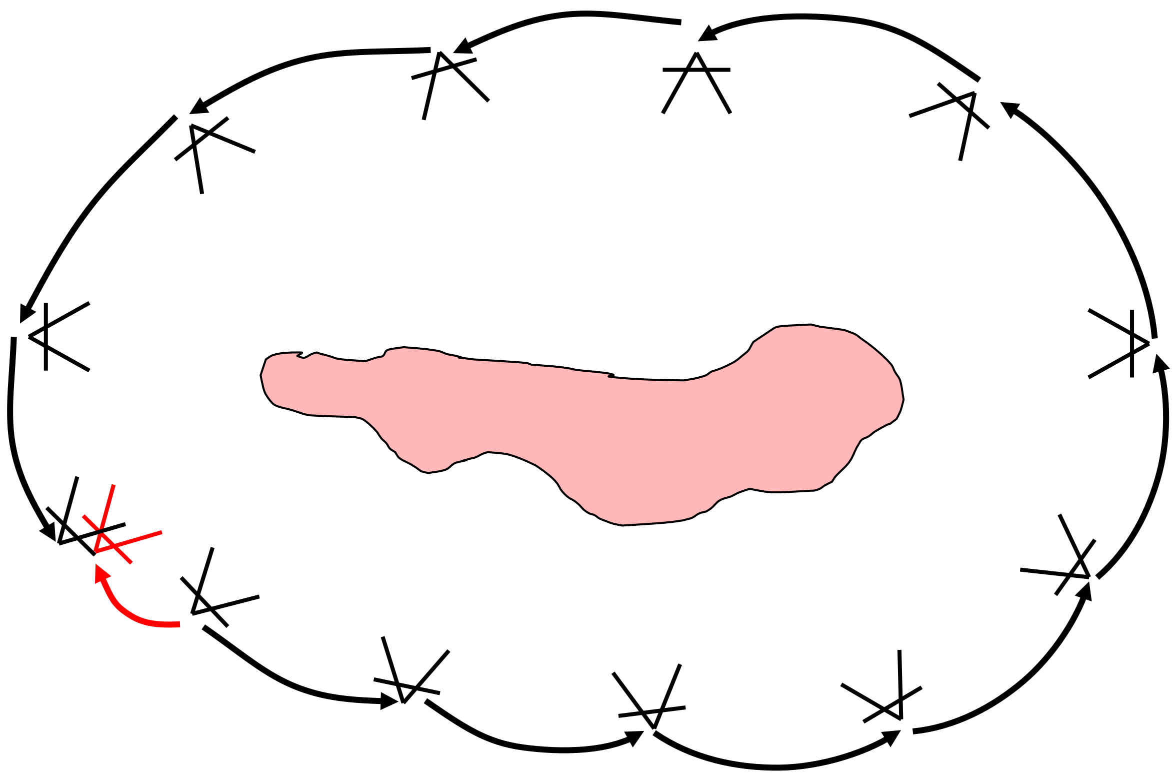


Loop Closures

- **Idea:** Estimate camera motion from frame to frame
- **Problem:**
  - Estimates are inherently noisy
  - Error accumulates over time → drift

Incremental Motion Estimation

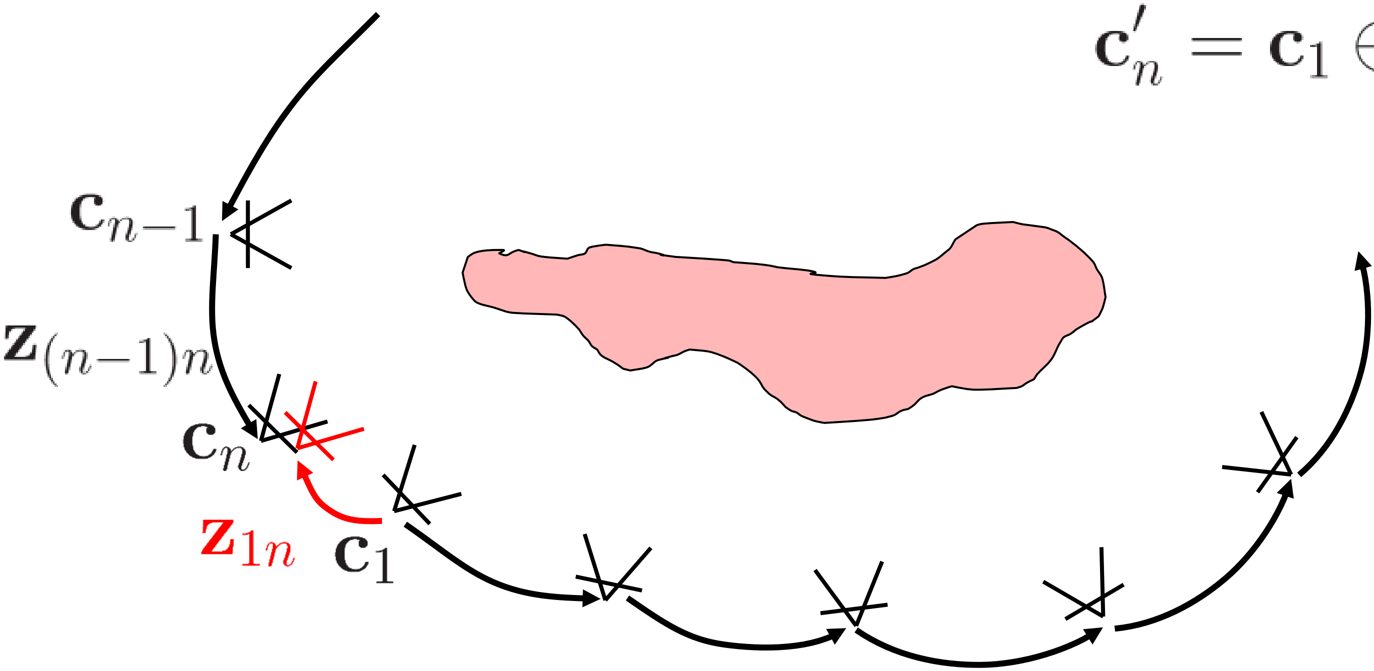
- **Idea:** Estimate camera motion from frame to frame



Incremental Motion Estimation

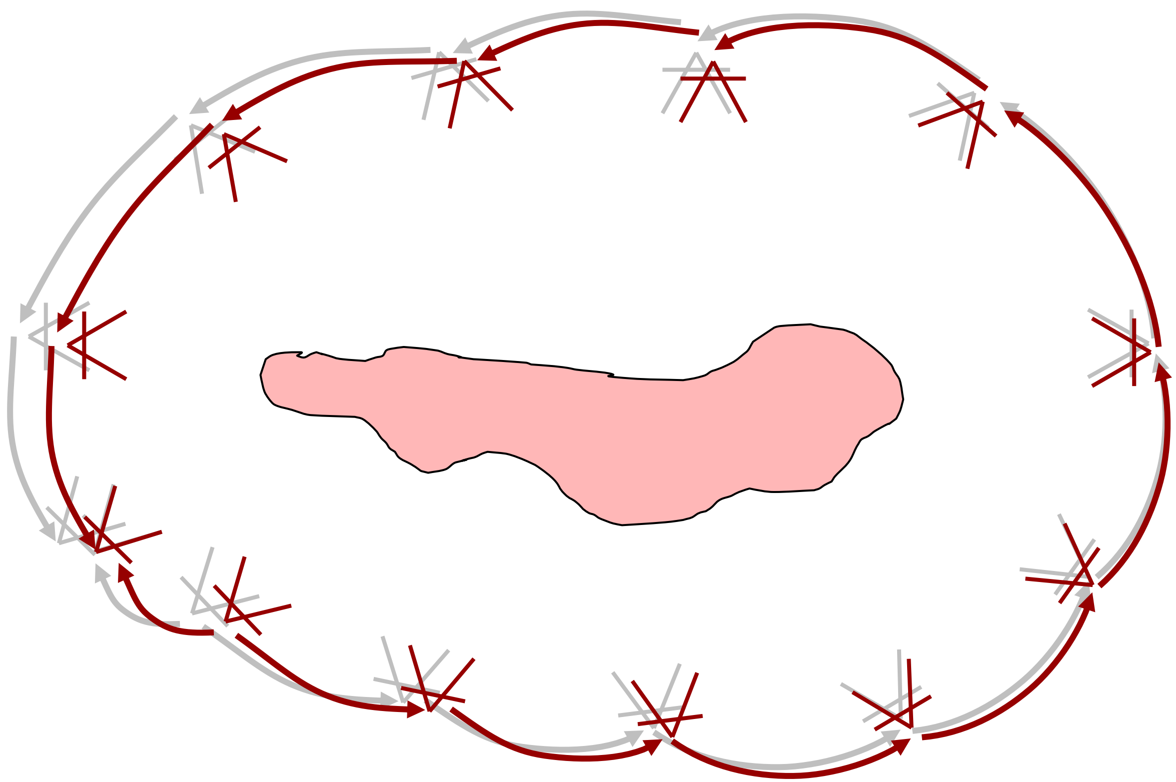
- **Idea:** Estimate camera motion from frame to frame
- Two ways to compute $\mathbf{c}_n$: $\mathbf{c}_n = \mathbf{c}_{n-1} \oplus \mathbf{z}_{(n-1)n}$

$$\mathbf{c}'_n = \mathbf{c}_1 \oplus \mathbf{z}_{1n}$$



Loop Closures

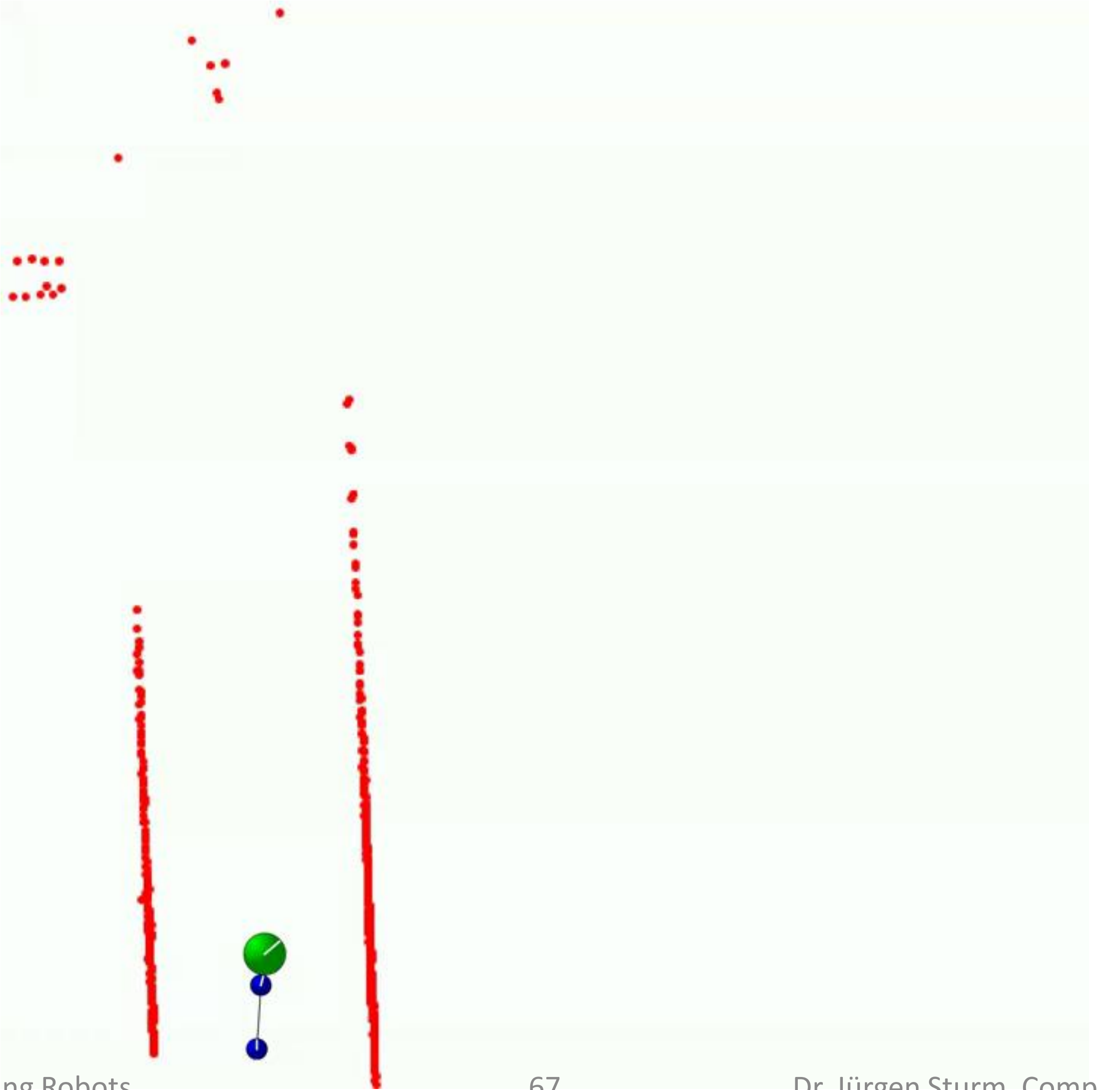
- **Solution:** Use loop-closures to minimize the drift / minimize the error over all constraints



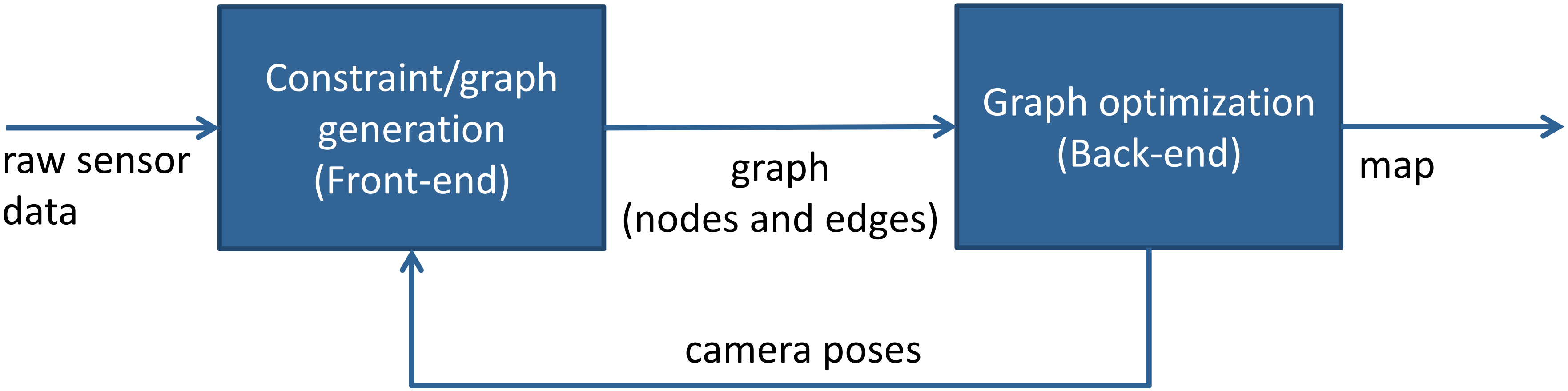
Graph SLAM
[Thrun and Montemerlo, 2006; Olson et al., 2006]

- Use a graph to represent the model
- Every **node** in the graph corresponds to a pose of the robot during mapping
- Every **edge** between two nodes corresponds to a spatial constraint between them
- **Graph-based SLAM:** Build the graph and find the robot poses that **minimize the error** introduced by the constraints

Example: Graph SLAM on Intel Dataset



Graph SLAM Architecture

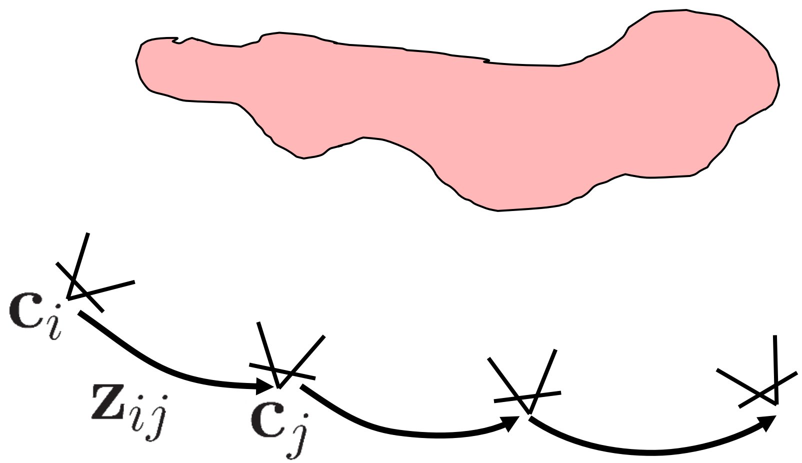


- Interleaving process of front-end and back-end
- A consistent map helps to determine new constraints by reducing the search space

Problem Definition

- **Given:** Set of relative pose observations $\mathbf{z}_{ij} \in \mathbb{R}^6$

- **Wanted:** Set of camera poses $\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathbb{R}^6$
  ➔ State vector $\mathbf{x} = (\mathbf{c}_1^\top, \ldots, \mathbf{c}_n^\top)^\top \in \mathbb{R}^{6n}$



Map Error

- Observation $\mathbf{z}_{ij}$
- Expected relative pose $\bar{\mathbf{z}}_{ij} = \mathbf{c}_j \ominus \mathbf{c}_i$

- Difference between observation and expectation

$$\mathbf{e}_{ij} = \mathbf{z}_{ij} \ominus \bar{\mathbf{z}}_{ij}$$

- Given the correct map $\mathbf{x}$, this difference is the result of observation/sensor noise…

Error Function

- **Assumption:** Observation noise is normally distributed

$$\mathbf{e}_{ij} \sim \mathcal{N}(\mathbf{0}, \Sigma_{ij})$$

- Error term for one observation (proportional to negative loglikelihood)

$$f_{ij}(\mathbf{x}) = -\log p(\mathbf{e}_{ij}) \propto \mathbf{e}_{ij}(\mathbf{x})^\top \Sigma_{ij}^{-1} \mathbf{e}_{ij}(\mathbf{x})$$

- Note: error is a scalar $f_{ij}(\mathbf{x}) \in \mathbb{R}$

Error Function

- Map error (over all observations)

$$f(\mathbf{x}) = \sum_{ij} f_{ij}(\mathbf{x}) = \sum_{ij} \mathbf{e}_{ij}(\mathbf{x})^\top \Sigma_{ij}^{-1} \mathbf{e}_{ij}(\mathbf{x})$$

- **Minimize this error** by optimizing the camera poses

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{ij} \mathbf{e}_{ij}(\mathbf{x})^\top \Sigma_{ij}^{-1} \mathbf{e}_{ij}(\mathbf{x})$$

- How can we solve this optimization problem?

Non-Linear Optimization Techniques

- Gradient descend
- Gauss-Newton
- Levenberg-Marquardt

Gauss-Newton Method

1. Linearize the error function
2. Compute its derivative
3. Set the derivative to zero
4. Solve the linear system
5. Iterate this procedure until convergence

Linearization and Derivation

- Error function

$$f(\mathbf{x}) = \sum_{ij} f_{ij}(\mathbf{x}) = \sum_{ij} \mathbf{e}_{ij}(\mathbf{x})^\top \Sigma_{ij}^{-1} \mathbf{e}_{ij}(\mathbf{x})$$

- Linearize the error function around the initial guess

$$f(\mathbf{x} + \Delta\mathbf{x}) = \sum_{ij} \mathbf{e}_{ij}(\mathbf{x} + \Delta\mathbf{x})^\top \Sigma_{ij}^{-1} \underline{\mathbf{e}_{ij}(\mathbf{x} + \Delta\mathbf{x})}$$

Let's look at this term first…

Linearizing the Error Function

- Approximate the error function around an initial guess $\mathbf{x} \in \mathbb{R}^{6n}$ using Taylor expansion

$$\mathbf{e}_{ij}(\mathbf{x} + \Delta\mathbf{x}) \simeq \mathbf{e}_{ij}(\mathbf{x}) + J_{ij}\Delta\mathbf{x} \qquad (\in \mathbb{R}^6)$$

  with increment

$$\Delta\mathbf{x} \in \mathbb{R}^{6n}$$

  and Jacobian

$$J_{ij}(\mathbf{x}) = \begin{pmatrix} \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{c}_1} & \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{c}_2} & \cdots & \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{c}_n} \end{pmatrix} \in \mathbb{R}^{6 \times 6n}$$

Derivatives of the Error Terms

- Does one error function $\mathbf{e}_{ij}(\mathbf{x})$ depend on all state variables in $\mathbf{x}$?

Derivatives of the Error Terms

- Does one error function $\mathbf{e}_{ij}(\mathbf{x})$ depend on all state variables in $\mathbf{x}$?
  - No, $\mathbf{e}_{ij}(\mathbf{x})$ depends only on $\mathbf{c}_i$ and $\mathbf{c}_j$

Derivatives of the Error Terms

- Does one error function $\mathbf{e}_{ij}(\mathbf{x})$ depend on all state variables in $\mathbf{x}$?
  - No, $\mathbf{e}_{ij}(\mathbf{x})$ depends only on $\mathbf{c}_i$ and $\mathbf{c}_j$
- Is there any consequence on the **structure** of the Jacobian?

Derivatives of the Error Terms

- Does one error function $\mathbf{e}_{ij}(\mathbf{x})$ depend on all state variables in $\mathbf{x}$?
  - No, $\mathbf{e}_{ij}(\mathbf{x})$ depends only on $\mathbf{c}_i$ and $\mathbf{c}_j$
- Is there any consequence on the **structure** of the Jacobian?
  - Yes, it will be non-zero only in the columns corresponding to $\mathbf{c}_i$ and $\mathbf{c}_j$
  - Jacobian is **sparse**

$$J_{ij}(\mathbf{x}) = \begin{pmatrix} \mathbf{0} & \cdots & \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{c}_i} & \cdots & \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{c}_j} & \cdots & \mathbf{0} \end{pmatrix}$$

Linearizing the Error Function

Linearize $f(\mathbf{x}) = \sum_{ij} \mathbf{e}_{ij}(\mathbf{x})^T \Sigma_{ij}^{-1} \mathbf{e}_{ij}(\mathbf{x})$

$$\simeq \mathbf{c} + 2\mathbf{b}^\top \Delta\mathbf{x} + \Delta\mathbf{x}^\top H \Delta\mathbf{x}$$

with $\mathbf{b}^\top = \sum_{ij} \mathbf{e}_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$

$$H = \sum_{ij} J_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$$

(Linear) Least Squares Minimization

1. Linearize error function

$$f(\mathbf{x} + \Delta\mathbf{x}) \simeq \mathbf{c} + 2\mathbf{b}^\top \Delta\mathbf{x} + \Delta\mathbf{x}^\top H \Delta\mathbf{x}$$

2. Compute the derivative

$$\frac{\mathrm{d}f(\mathbf{x} + \Delta\mathbf{x})}{\mathrm{d}\Delta\mathbf{x}} = 2\mathbf{b} + 2H\Delta\mathbf{x}$$

3. Set derivative to zero

$$H\Delta\mathbf{x} = -\mathbf{b}$$

4. Solve this linear system of equations, e.g.,

$$\Delta\mathbf{x} = -H^{-1}\mathbf{b}$$

Gauss-Newton Method

Problem: $f(\mathbf{x})$ is non-linear!

Algorithm: Repeat until convergence

1. Compute the terms of the linear system

$$\mathbf{b}^\top = \sum_{ij} \mathbf{e}_{ij}^\top \Sigma_{ij}^{-1} J_{ij} \qquad H = \sum_{ij} J_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$$

2. Solve the linear system to get new increment

$$H \Delta\mathbf{x} = -\mathbf{b}$$

3. Update previous estimate

$$\mathbf{x} \leftarrow \mathbf{x} + \Delta\mathbf{x}$$

Structure of the Minimization Problem

Linearize $f(\mathbf{x}) = \sum_{ij} \mathbf{e}_{ij}(\mathbf{x})^T \Sigma_{ij}^{-1} \mathbf{e}_{ij}(\mathbf{x})$

$$\simeq \mathbf{c} + 2\mathbf{b}^\top \Delta\mathbf{x} + \Delta\mathbf{x}^\top H \Delta\mathbf{x}$$

with $\mathbf{b}^\top = \sum_{ij} \mathbf{e}_{ij}^\top \Sigma_{ij}^{-1} J_{ij} \in \mathbb{R}^{6n}$

$$H = \sum_{ij} J_{ij}^\top \Sigma_{ij}^{-1} J_{ij} \in \mathbb{R}^{6n \times 6n}$$

this quickly gets huge!

- What is the structure of $\mathbf{b}^\top$ and H? (Remember: all J_{ij}'s are sparse)

Illustration of the Structure

$$\mathbf{b}_{ij}^\top = \mathbf{e}_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$$

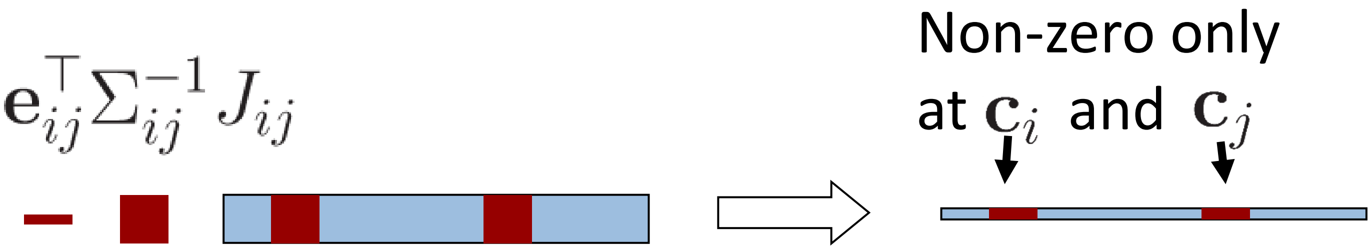
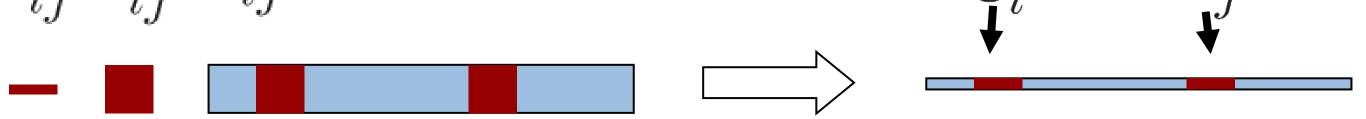


Illustration of the Structure

$$\mathbf{b}_{ij}^\top = \mathbf{e}_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$$

Non-zero only at $\mathbf{c}_i$ and $\mathbf{c}_j$



$$H_{ij} = J_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$$

Non-zero on the main diagonal at $\mathbf{c}_i$ and $\mathbf{c}_j$

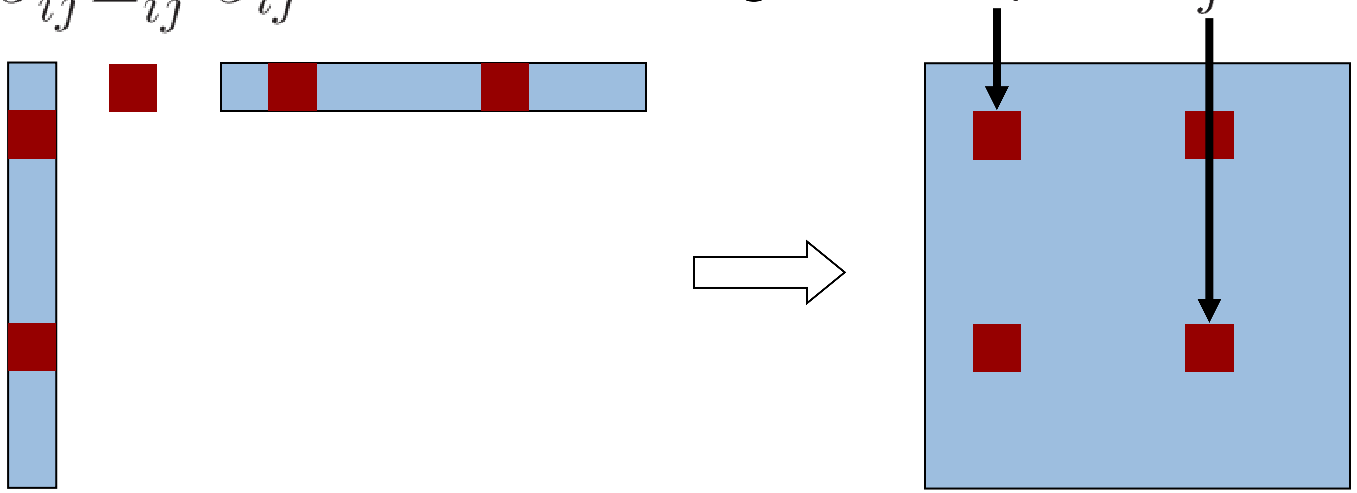
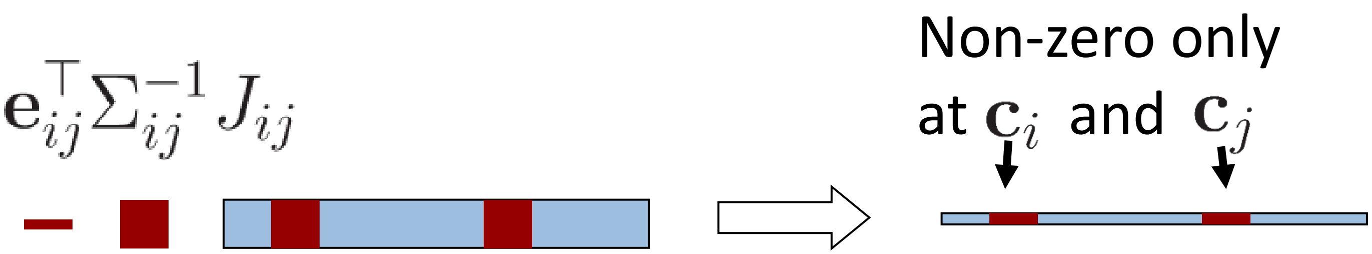


Illustration of the Structure

$$\mathbf{b}_{ij}^\top = \mathbf{e}_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$$



$$H_{ij} = J_{ij}^\top \Sigma_{ij}^{-1} J_{ij}$$

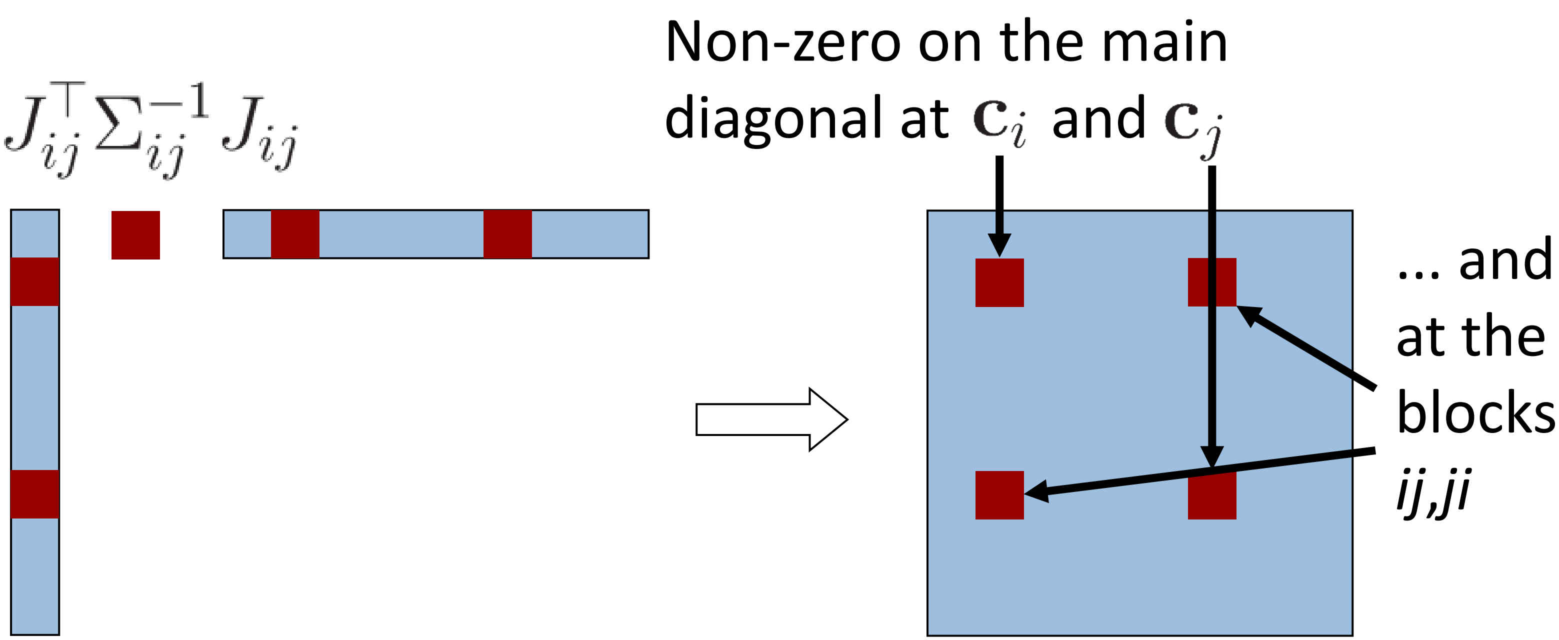
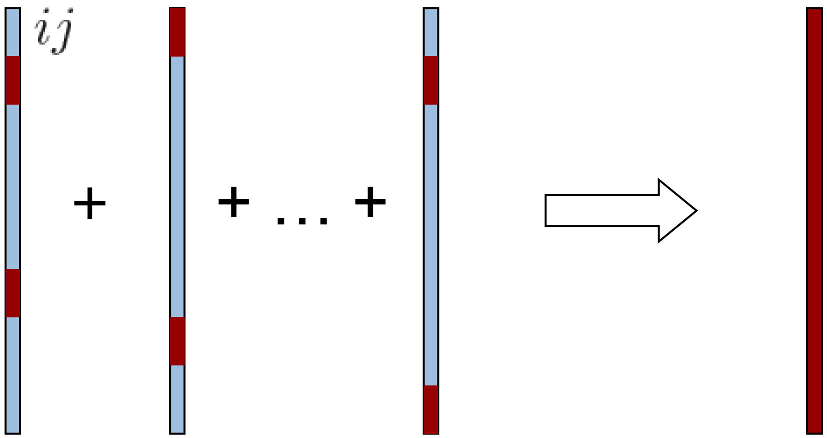


Illustration of the Structure

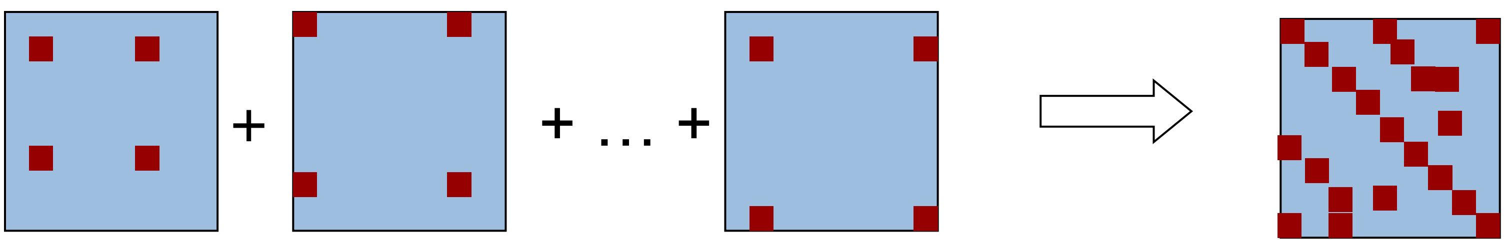
$$\mathbf{b} = \sum_{ij} \mathbf{b}_{ij}$$

b: dense vector



$$H = \sum_{ij} H_{ij}$$

H: sparse block structure
with main diagonal



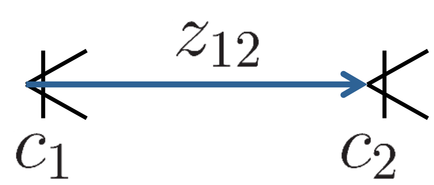
Sparsity of the Hessian

- Remember: We have to solve $H\Delta\mathbf{x} = -\mathbf{b}$
- The Hessian is
  - positive semi-definit
  - symmetric
  - sparse
- This allows the use of efficient solvers
  - Sparse Cholesky decomposition (~100M matrix elements)
  - Preconditioned conjugate gradients (~1.000M matrix elements)
  - ... many others

Example in 1D

- Two camera poses $c_1, c_2 \in \mathbb{R}$
- State vector $\mathbf{x} = (c_1, c_2)^\top \in \mathbb{R}^2$
- One (distance) observation $z_{12} \in \mathbb{R}$

- Initial guess $c_1 = c_2 = 0$
- Observation $z_{12} = 1$
- Sensor noise $\Sigma_{12} = 0.5$



Example in 1D

- Error $$e_{12} = z_{12} - \bar{z}_{12}$$ $$= z_{12} - (c_2 - c_1) = 1 - (0 - 0) = 1$$
- Jacobian $$J_{12} = \begin{pmatrix} \frac{\partial e_{12}}{\partial c_1} & \frac{\partial e_{12}}{\partial c_2} \end{pmatrix} = \begin{pmatrix} 1 & -1 \end{pmatrix}$$
- Build linear system of equations
$$b^\top = e_{12}^\top \Sigma^{-1} e_{12} = \begin{pmatrix} 2 & -2 \end{pmatrix}$$
$$H = J_{12}^\top \Sigma^{-1} J_{12} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix}$$
- Solve the system
$$\Delta x = -H^{-1} b$$ **but** $\det H = 0$ **???**

What Went Wrong?

- The constraint only specifies a **relative constraint** between two nodes
- Any poses for the nodes would be fine as long as their relative pose fits
- **One node needs to be fixed**
  - Option 1: Remove one row/column corresponding to the fixed pose
  - Option 2: Add to $H, \mathbf{b}$ a linear constraint $1 \cdot \Delta c_1 = 0$
  - Option 3: Add the identity matrix to H (Levenberg-Marquardt)

Fixing One Node

- The constraint only specifies a **relative constraint** between two nodes
- Any poses for the nodes would be fine as long as their relative pose fits
- **One node needs to be fixed (here: Option 2)**

$$H = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} + \boxed{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}}$$

additional constraint that sets $\Delta c_1 = 0$

$$\Delta x = -H^{-1} b$$

$$\Delta x = \begin{pmatrix} 0 & 1 \end{pmatrix}^\top$$

Levenberg-Marquardt Algorithm

- **Observations:**
  - Gauss-Newton method typically converges very quickly
  - Sometimes diverges when initial solution is far off
  - Gradient descent (with line search) never diverges
- **How can we combine the advantages of both minimization methods?**

Levenberg-Marquardt Algorithm

- **Idea:** Add a damping factor

$$(H + \lambda I)\Delta \mathbf{x} = -\mathbf{b}$$

$$(J^\top J + \lambda I)\Delta \mathbf{x} = -J^\top \mathbf{e}$$

- What is the effect of this damping factor?
  - Small λ → same as least squares
  - Large λ → steepest descent (with small step size)
- **Algorithm**
  - If error decreases, accept $\Delta \mathbf{x}$ and reduce λ
  - If error increases, reject $\Delta \mathbf{x}$ and increase λ

Non-Linear Minimization

- One of the state-of-the-art solution to compute the maximum likelihood estimate
- Various open-source implementations available
  - g2o [Kuemmerle et al., 2011]
  - sba [Lourakis and Argyros, 2009]
  - iSAM [Kaess et al., 2008]
  - Ceres [Google, 2012]
- Other extensions:
  - Robust error functions
  - Alternative parameterizations

Google Street View
Map Optimization with Ceres Solver
[Google, 2012]

Lessons Learned Today

- How to separate inliers from outliers using RANSAC
- How to align point clouds using ICP
- How to model the SLAM problem in a graph
- How to optimize the map using non-linear least squares