Localization and Navigation

Where Are We?

METR 4202: Advanced Control & Robotics
Dr Surya Singh
Lecture # 8
September 13, 2013

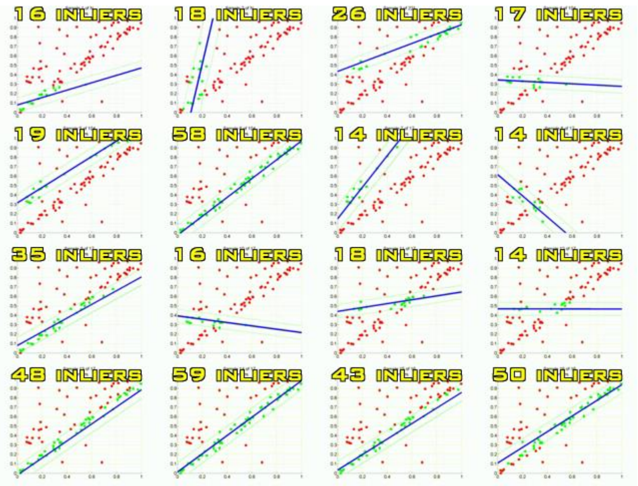
metr4202@itee.uq.edu.au
http://itee.uq.edu.au/~metr4202/

RoboticsCourseWare.org
RoboticsCourseWare Contributor

Schedule

Week	Date	Lecture (F: 9-10:30, 42-212)
1	26-Jul	Introduction
2	2-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	9-Aug	Robot Kinematics
4	16-Aug	Robot Dynamics & Control
5	23-Aug	Robot Trajectories & Motion
6	30-Aug	Sensors & Measurement
7	6-Sep	Perception / Computer Vision
8	**13-Sep**	**Localization and Navigation**
9	20-Sep	State-Space Modelling
	27-Sep	State-Space Control
10	4-Oct	*Study break*
11	11-Oct	Motion Planning
12	18-Oct	Vision-based control (+ Prof. P. Corke or Prof. M. Srinivasan)
13	25-Oct	Applications in Industry (+ Prof. S. LaValle) & Course Review

Cool Robotics Share (this week)



D. Wedge, *The RANSAC Song*

RANdom SAmple Consensus

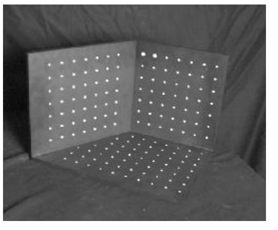
1. Repeatedly select a small (minimal) subset of correspondences
2. Estimate a solution (in this case a the line)
3. Count the number of "inliers", $|e|<\Theta$ (for LMS, estimate med($|e|$)
4. Pick the *best* subset of inliers
5. Find a complete least-squares solution

- Related to least median squares
- See also: MAPSAC (Maximum *A Posteriori* SAmple Consensus)

From Szeliski, *Computer Vision: Algorithms and Applications*

Camera calibration

- Determine camera parameters from known 3D points or calibration object(s)
- internal or intrinsic parameters such as focal length, optical center, aspect ratio:
  what kind of camera?
- external or extrinsic (pose) parameters:
  where is the camera?
- How can we do this?



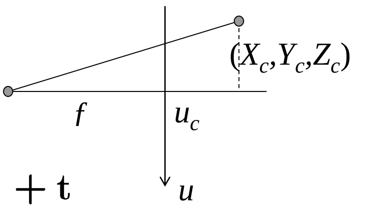
From Szeliski, *Computer Vision: Algorithms and Applications*

Camera calibration – approaches

- Possible approaches:
  – linear regression (least squares)
  – non-linear optimization
  – vanishing points
  – multiple planar patterns
  – panoramas (rotational motion)

From Szeliski, *Computer Vision: Algorithms and Applications*

Image formation equations



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Calibration matrix

- Is this form of K good enough?
- non-square pixels (digital video)
- skew
- radial distortion

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{K} \; \mathbf{X}_c$$

$$\begin{bmatrix} fa & s & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{K}$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Levenberg-Marquardt

- Iterative non-linear least squares [Press'92]
  - Linearize measurement equations

$$\begin{aligned}\hat{u}_i &= f(\mathbf{m}, \mathbf{x}_i) + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m} \\ \hat{v}_i &= g(\mathbf{m}, \mathbf{x}_i) + \frac{\partial g}{\partial \mathbf{m}} \Delta \mathbf{m}\end{aligned}$$

  - Substitute into log-likelihood equation: quadratic cost function in Dm

$$\sum_i \sigma_i^{-2} (\hat{u}_i - u_i + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m})^2 + \cdots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Levenberg-Marquardt

- What if it doesn't converge?
  - Multiply diagonal by (1 + l), increase l until it does
  - Halve the step size Dm (my favorite)
  - Use line search
  - Other ideas?
- Uncertainty analysis: covariance S = A-1
- Is maximum likelihood the best idea?
- How to start in vicinity of global minimum?

From Szeliski, *Computer Vision: Algorithms and Applications*

Camera matrix calibration

- Advantages:
  - very simple to formulate and solve
  - can recover K [R | t] from M using QR decomposition [Golub & VanLoan 96]

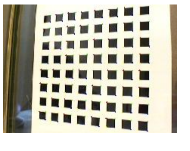
- Disadvantages:
  - doesn't compute internal parameters
  - more unknowns than true degrees of freedom
  - need a separate camera matrix for each new view

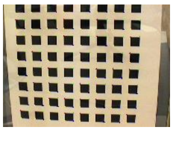
From Szeliski, *Computer Vision: Algorithms and Applications*

Multi-plane calibration

- Use several images of planar target held at unknown orientations [Zhang 99]
  - Compute plane homographies

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim \mathbf{HX}$$





  - Solve for K-TK-1 from Hk's
    - 1plane if only f unknown
    - 2 planes if (f,uc,vc) unknown
    - 3+ planes for full K
  - Code available from Zhang and OpenCV

From Szeliski, *Computer Vision: Algorithms and Applications*

Rotational motion

- Use pure rotation (large scene) to estimate f
  - estimate f from pairwise homographies
  - re-estimate f from 360º "gap"
  - optimize over all {K,Rj} parameters
    [Stein 95; Hartley '97; Shum & Szeliski '00; Kang & Weiss '99]

- Most accurate way to get f, short of surveying distant points

From Szeliski, *Computer Vision: Algorithms and Applications*

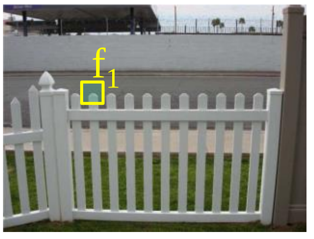
Feature matching

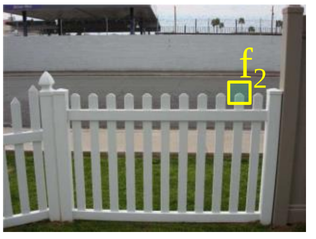
- Given a feature in I_1, how to find the best match in I_2?
  1. Define distance function that compares two descriptors
  2. Test all the features in I_2, find the one with min distance

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature distance

- How to define the difference between two features f_1, f_2?
  - Simple approach is $SSD(f_1, f_2)$
    - sum of square differences between entries of the two descriptors
    - can give good scores to very ambiguous (bad) matches



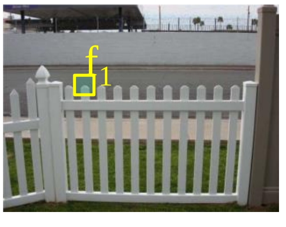


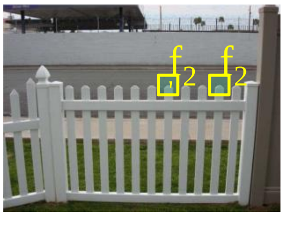
I_1 I_2

From Szeliski, *Computer Vision: Algorithms and Applications*

Feature distance

- How to define the difference between two features f_1, f_2?
  - Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
    - f_2 is best SSD match to f_1 in I_2
    - f_2' is 2nd best SSD match to f_1 in I_2
    - gives small values for ambiguous matches



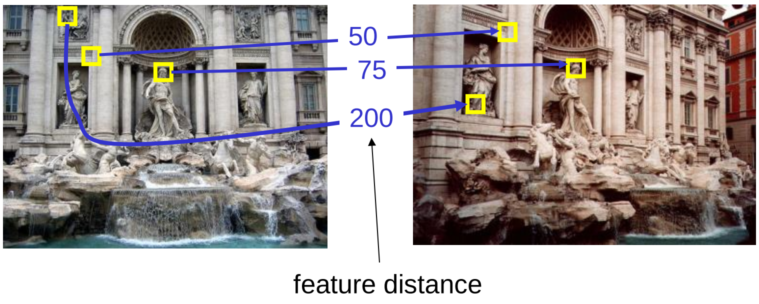


I_1 I_2

From Szeliski, *Computer Vision: Algorithms and Applications*

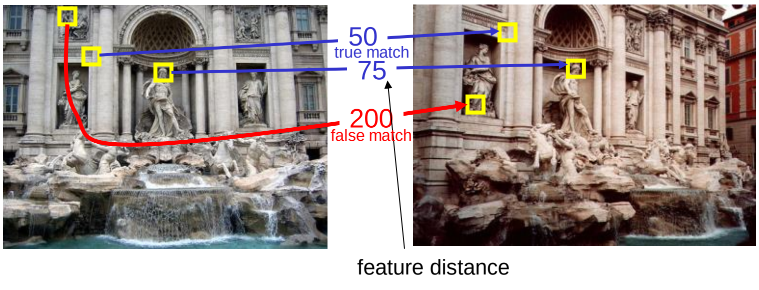
Evaluating the results

- How can we measure the performance of a feature matcher?



From Szeliski, *Computer Vision: Algorithms and Applications*

True/false positives



- The distance threshold affects performance
  - True positives = # of detected matches that are correct
    - Suppose we want to maximize these—how to choose threshold?
  - False positives = # of detected matches that are incorrect
    - Suppose we want to minimize these—how to choose threshold?

From Szeliski, *Computer Vision: Algorithms and Applications*

SFM: Structure from Motion
(& Cool Robotics Share (this week))

Structure [from] Motion

- Given a set of feature tracks,
  estimate the 3D structure and 3D (camera) motion.
- Assumption: orthographic projection
- Tracks: (u_{fp},v_{fp}), f: frame, p: point
- Subtract out **mean** 2D position…

  $\mathbf{i}_f$: rotation, $\mathbf{s}_p$: position

$$u_{fp} = i_f^T s_p, v_{fp} = j_f^T s_p$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Structure from motion

- How many points do we need to match?
- 2 frames:
  - (R,t): 5 dof + 3n point locations ≤
  - 4n point measurements ⇒
  - n ≥ 5
- k frames:
  - 6(k–1)-1 + 3n ≤ 2kn
- always want to use many more

$$\hat{u}_{ij} = f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)$$
$$\hat{v}_{ij} = g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Measurement equations

- Measurement equations

$$u_{fp} = \boldsymbol{i}_f^T \boldsymbol{s}_p \qquad \boldsymbol{i}_f\text{: rotation, } \boldsymbol{s}_p\text{: position}$$
$$v_{fp} = \boldsymbol{j}_f^T \boldsymbol{s}_p$$

- Stack them up…

$$\boldsymbol{W} = \boldsymbol{R}\,\boldsymbol{S}$$
$$\boldsymbol{R} = (\boldsymbol{i}_1,\ldots,\boldsymbol{i}_F, \boldsymbol{j}_1,\ldots,\boldsymbol{j}_F)^T$$
$$\boldsymbol{S} = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_P)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Factorization

$$\boldsymbol{W} = \boldsymbol{R}_{2F\times 3}\,\boldsymbol{S}_{3\times P}$$

SVD

$$\boldsymbol{W} = \boldsymbol{U}\,\boldsymbol{\Lambda}\,\boldsymbol{V} \quad \boldsymbol{\Lambda} \text{ must be rank 3}$$

$$\boldsymbol{W'} = (\boldsymbol{U}\,\boldsymbol{\Lambda}^{1/2})(\boldsymbol{\Lambda}^{1/2}\,\boldsymbol{V}) \quad = \boldsymbol{U'}\,\boldsymbol{V'}$$

Make ***R*** orthogonal

$$\boldsymbol{R} = \boldsymbol{QU'},\ \ \boldsymbol{S} = \boldsymbol{Q}^{-1}\boldsymbol{V'}$$

$$\boldsymbol{i}_f^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{i}_f = 1 \ \ldots$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Results

- Look at paper figures…

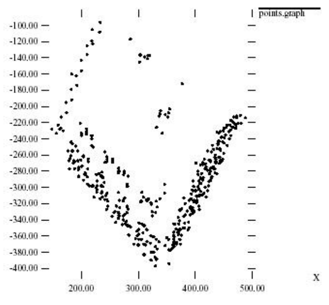


Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

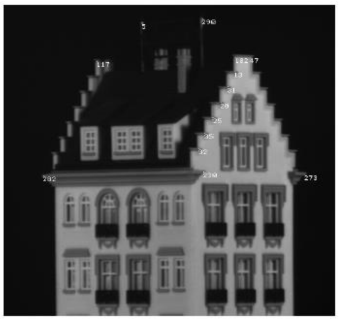


Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

From Szeliski, *Computer Vision: Algorithms and Applications*

Bundle Adjustment

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

$$\begin{aligned}\hat{u}_{ij} &= f(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})\\ \hat{v}_{ij} &= g(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})\end{aligned}$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Two-frame methods

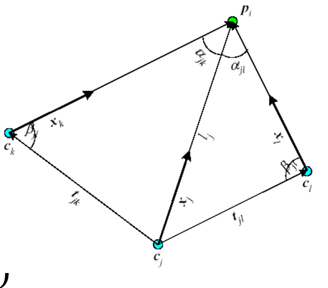
- Two main variants:
- Calibrated: "Essential matrix" E
  use ray directions (xi, xi')
- Uncalibrated: "Fundamental matrix" F

- [Hartley & Zisserman 2000]

From Szeliski, *Computer Vision: Algorithms and Applications*

Essential matrix

- Co-planarity constraint:
- $x' \approx R\,x + t$
- $[t]_\times x' \approx [t]_\times R\,x$
- $x'\,[t]_\times x' \approx x'\,[t]_\times R\,x$
- $x'\,E\,x = 0$ with $E = [t]_\times R$
- Solve for E using least squares (SVD)
- t is the least singular vector of E
- R obtained from the other two s.v.s



From Szeliski, *Computer Vision: Algorithms and Applications*

Fundamental matrix

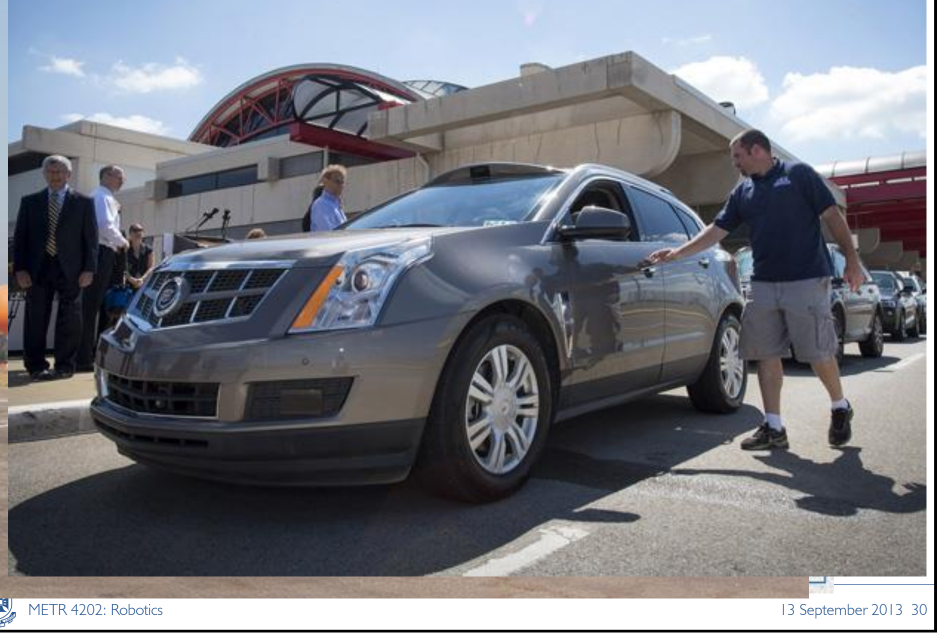
- Camera calibrations are unknown
- $x'\,F\,x = 0$ with $F = [e]_\times H = K'[t]_\times R\,K\text{-}1$
- Solve for F using least squares (SVD)
  - re-scale (xi, xi') so that |xi|≈1/2 [Hartley]
- e (epipole) is still the least singular vector of F
- H obtained from the other two s.v.s
- "plane + parallax" (projective) reconstruction
- use self-calibration to determine K [Pollefeys]

From Szeliski, *Computer Vision: Algorithms and Applications*

Cool Robotics Share

D. Wedge, *The Fundamental Matrix Song*

Localization: SFM → SLAM



What is SLAM?

- SLAM asks the following question:

  Is it possible for an autonomous vehicle to start at an unknown location in an unknown environment and then to incrementally build a map of this environment while simultaneously using this map to compute vehicle location?

- SLAM has many indoor, outdoor, in-air and underwater applications for both manned and autonomous vehicles.

- Examples
  - Explore and return to starting point (Newman)
  - Learn trained paths to different goal locations
  - Traverse a region with complete coverage (eg, mine fields, lawns, reef monitoring)
  - …

Components of SLAM

- Localisation
  - Determine pose given a priori map
- Mapping
  - Generate map when pose is accurately known from auxiliary source.
- SLAM
  - Define some arbitrary coordinate origin
  - Generate a map from on-board sensors
  - Compute pose from this map
  - Errors in map and in pose estimate are dependent.

History of SLAM

- It all started about 20 years ago at ICRA86 in San Francisco.
  - Probabilistic methods were new to robotics and AI
  - Several researchers were looking at applying estimation-theoretic methods to mapping and localisation problems
- They saw that:
  - Consistent probabilistic mapping was a fundamental problem
  - Major conceptual and computational issues needed to be addressed
- Key papers were written on geometric uncertainty (Smith and Cheeseman, HDW).
  - They showed that estimates exhibit a high degree of correlation between geometric features (ie, landmark locations in a map).

History of SLAM

- Landmark paper by Smith, Self and Cheeseman
  - Landmark estimates correlated via vehicle pose estimate
- Important implication
  - A consistent full solution requires a joint state composed of the vehicle pose and every landmark position
  - Many landmarks means huge state vector
  - Estimate to be updated following each landmark observation
- At the time, estimation meant Kalman filters
  - Computation and storage costs scale quadratically with the number of landmarks

History of SLAM

- Researchers initially thought SLAM would not converge
  - Assumed estimated map errors would exhibit a random walk behaviour
  - Unbounded map uncertainty
  - Leonard (1991): Simultaneous mapping and localisation, which came first, the chicken or the egg ?
- Researchers tried to *minimise* correlations between landmarks
  - Applied approximations to minimise or eliminate correlations, or simply assumed they were zero
  - Reduced the full filter to a series of decoupled landmark to vehicle filters.
- Theoretical work on SLAM came to a temporary halt
  - Work focused on either mapping or localisation as separate problems.

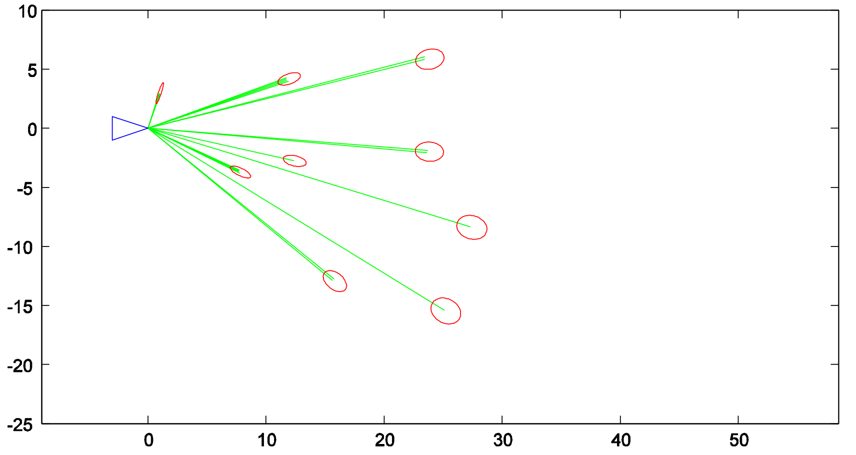
History of SLAM

- Conceptual break-through: SLAM converges (Csorba, Dissa)
  - Must correctly formulate as a joint state; correlations are essential
  - Correlations grow, not shrink
  - Stronger correlations means better relative map estimate
- The term `SLAM' was introduced in 1995.
- Groups started working again at SLAM (aka CML)
  - ACFR, Zaragoza, MIT
  - Key series of papers through 2000 showing convergence properties.
  - Initial work on computational efficiency and on loop-closure.
- Growth in SLAM interest.
  - First `SLAM' session held at ISRR 1999. Included also probabilistic mapping methods of Sebastian Thrun and Dieter Fox.
  - First ICRA SLAM workshop: 2000
  - First SLAM summer school: 2002
  - Massive increase in SLAM papers since 2002

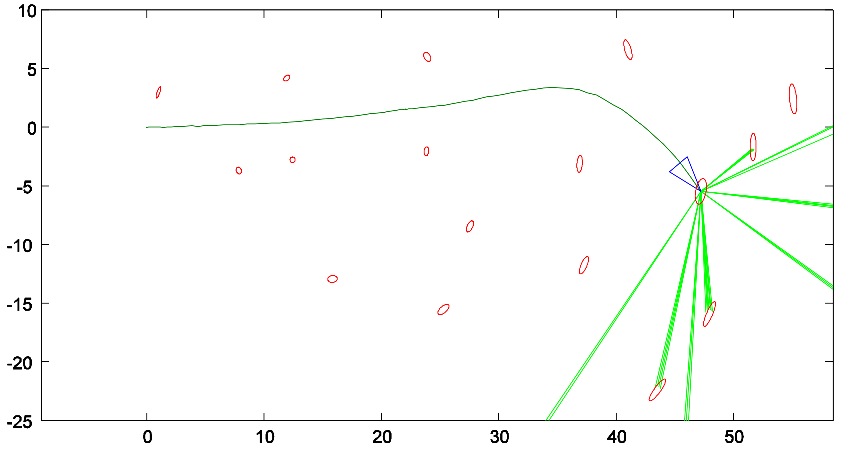
Basic SLAM Operation

Example: SLAM in Victoria Park

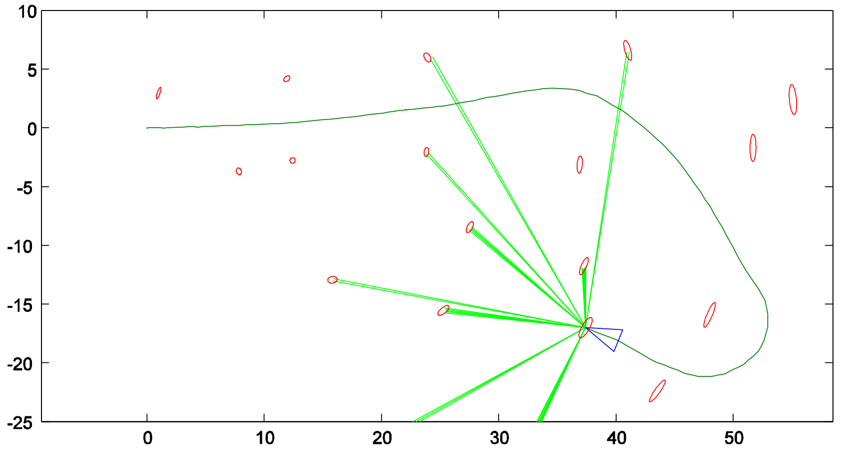
Basic SLAM Operation



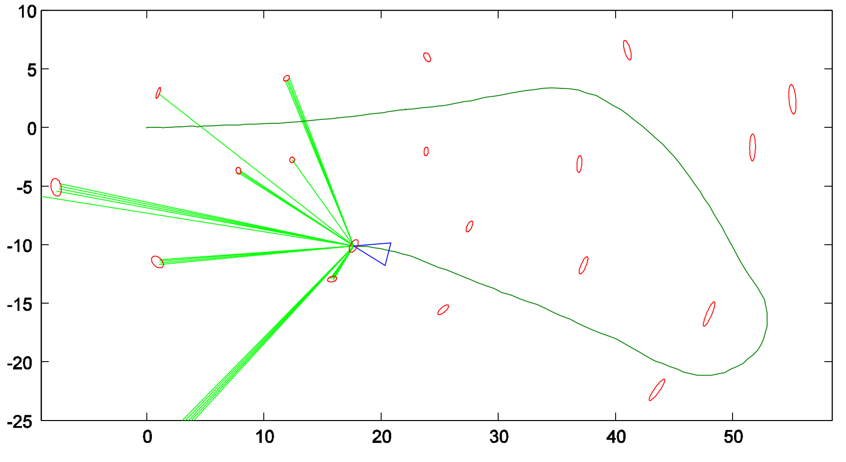
Basic SLAM Operation



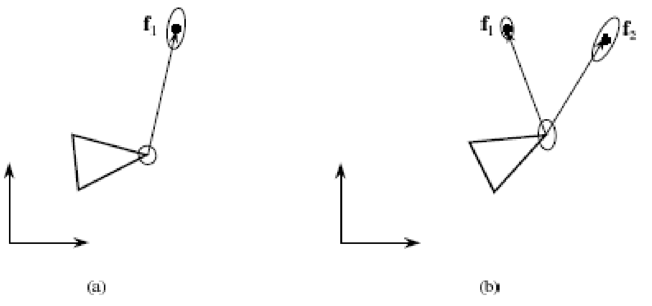
Basic SLAM Operation



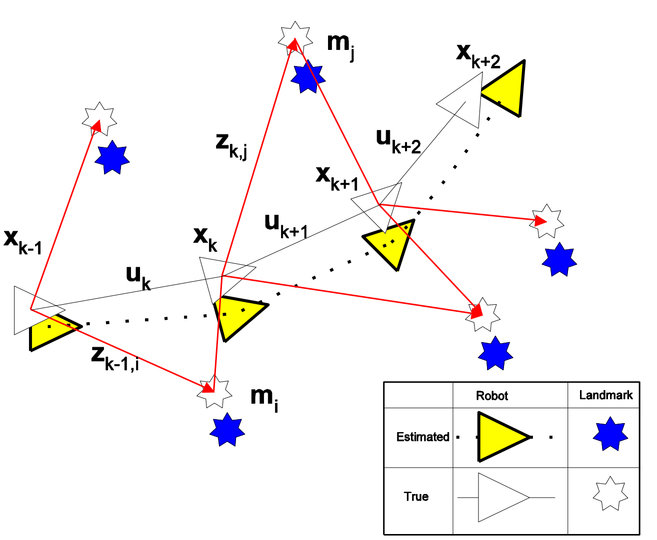
Basic SLAM Operation



Dependent Errors



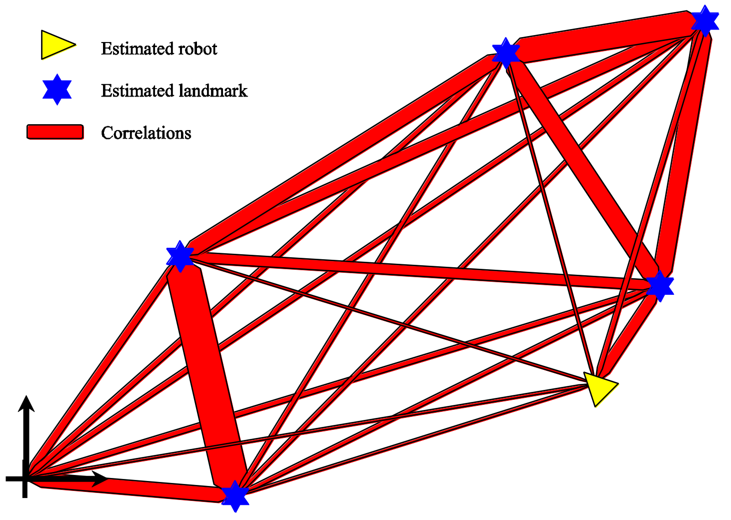
Correlated Estimates



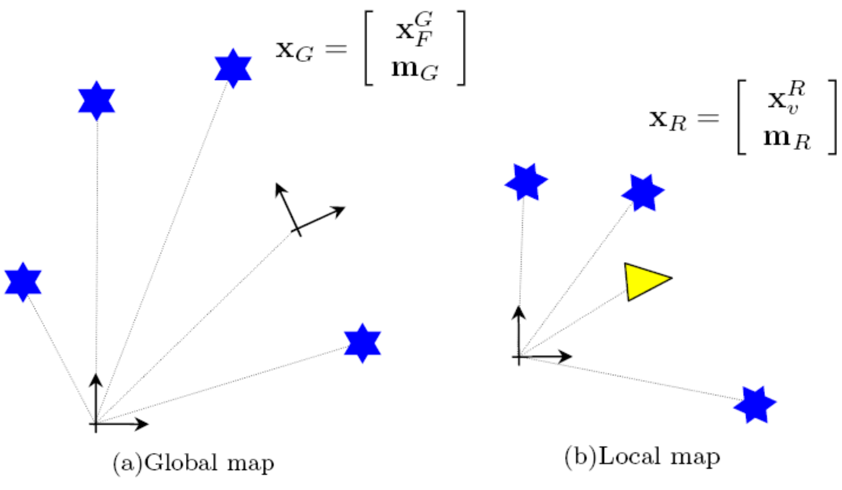
SLAM Convergence

- An observation acts like a displacement to a spring system
  - Effect is greatest in a close neighbourhood
  - Effect on other landmarks diminishes with distance
  - Propagation depends on local stiffness (correlation) properties
- With each new observation the springs become increasingly (and monotonically) stiffer.
- In the limit, a rigid map of landmarks is obtained.
  - A perfect *relative* map of the environment
- The location accuracy of the robot is bounded by
  - The current quality of the map
  - The relative sensor measurement

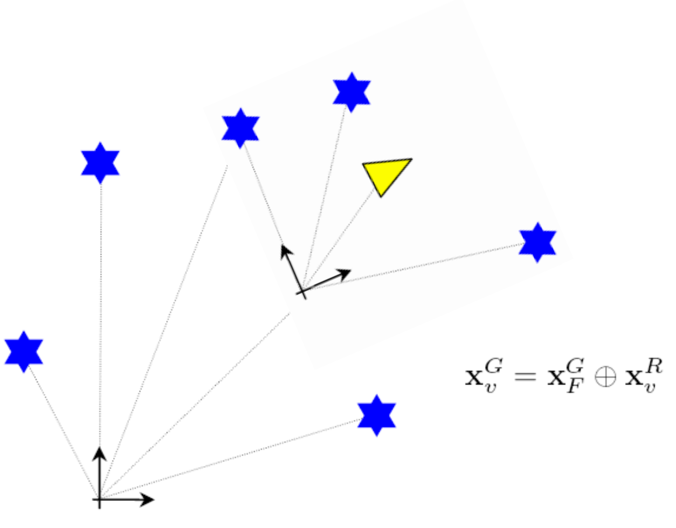
Spring Analogy



Constrained Local Submap Filter



CLSF Registration



CLSF Global Estimate