# Localization and Navigation

Where Are We?

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

Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture (F: 9-10:30, 42-212)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>26-Jul</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>2-Aug</td>
<td>Representing Position &amp; Orientation &amp; State (Frames, Transformation Matrices &amp; Affine Transformations)</td>
</tr>
<tr>
<td>3</td>
<td>9-Aug</td>
<td>Robot Kinematics</td>
</tr>
<tr>
<td>4</td>
<td>16-Aug</td>
<td>Robot Dynamics &amp; Control</td>
</tr>
<tr>
<td>5</td>
<td>23-Aug</td>
<td>Robot Trajectories &amp; Motion</td>
</tr>
<tr>
<td>6</td>
<td>30-Aug</td>
<td>Sensors &amp; Measurement</td>
</tr>
<tr>
<td>7</td>
<td>6-Sep</td>
<td>Perception / Computer Vision</td>
</tr>
<tr>
<td>8</td>
<td>13-Sep</td>
<td><strong>Localization and Navigation</strong></td>
</tr>
<tr>
<td>9</td>
<td>20-Sep</td>
<td>State-Space Modelling</td>
</tr>
<tr>
<td>10</td>
<td>27-Sep</td>
<td>State-Space Control</td>
</tr>
<tr>
<td></td>
<td>4-Oct</td>
<td>Study break</td>
</tr>
<tr>
<td>11</td>
<td>11-Oct</td>
<td>Motion Planning</td>
</tr>
<tr>
<td>12</td>
<td>18-Oct</td>
<td>Vision-based control (+ Prof. P. Corke or Prof. M. Srinivasan)</td>
</tr>
<tr>
<td>13</td>
<td>25-Oct</td>
<td>Applications in Industry (+ Prof. S. LaValle) &amp; Course Review</td>
</tr>
</tbody>
</table>
**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 $\text{med}(|e|)$)
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)
Image formation equations

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

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

From Szeliski, *Computer Vision: Algorithms and Applications*

---

Calibration matrix

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

\[
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix} \sim \begin{bmatrix}
f & 0 & u_c \\
0 & f & v_c \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix} = K \begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix}
\]

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

From Szeliski, *Computer Vision: Algorithms and Applications*
Levenberg-Marquardt

- Iterative non-linear least squares [Press’92]
  - Linearize measurement equations
    \[
    \hat{u}_i = f(m, x_i) + \frac{\partial f}{\partial m} \Delta m \\
    \hat{v}_i = g(m, x_i) + \frac{\partial g}{\partial m} \Delta m
    \]
  - Substitute into log-likelihood equation:
    quadratic cost function in \( \Delta m \)
    \[
    \sum_i \sigma_i^{-2}(\hat{u}_i - u_i + \frac{\partial f}{\partial m} \Delta m)^2 + \cdots
    \]

From Szeliski, *Computer Vision: Algorithms and Applications*
Camera matrix calibration

- **Advantages:**
  - very simple to formulate and solve
  - can recover $K[ R \vert t]$ 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

---

Multi-plane calibration

- **Use several images of planar target held at unknown orientations** [Zhang 99]
  - Compute plane homographies
    
    \[
    \begin{bmatrix}
    u_i \\
    v_i \\
    1 \\
    \end{bmatrix} \sim K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} x_i \\
    y_i \\
    1 \\
    \end{bmatrix} \sim HX
    \]
  - Solve for $K$-$TK^{-1}$ from $H_k$'s
    - 1 plane if only $f$ unknown
    - 2 planes if $(f, u_c, v_c)$ 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 360° “gap”
  – optimize over all \{K,R_j\} parameters
    [Stein 95; Hartley ’97; Shum & Szeliski ’00; Kang & Weiss ’99]

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

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 $\text{SSD}(f_1, f_2)$
    - sum of square differences between entries of the two descriptors
    - can give good scores to very ambiguous (bad) matches

From Szeliski, *Computer Vision: Algorithms and Applications*
Evaluating the results

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

![Feature distance diagram]

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...
  \[ u_{fp} = i_f^T s_p, \quad v_{fp} = j_f^T s_p \]

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 \( \leq \)
  – 4n point measurements \( \Rightarrow \)
  – \( n \geq 5 \)
• k frames:
  – 6(k−1)−1 + 3n \( \leq 2kn \)
• always want to use many more

Measurement equations

• Measurement equations
  \[ u_{fp} = i_f^T s_p \]
  \[ v_{fp} = j_f^T s_p \]

  \( i_f \): rotation, \( s_p \): position

  • Stack them up…
  \[ W = R S \]
  \[ R = (i_1, \ldots, i_F, j_1, \ldots, j_F)^T \]
  \[ S = (s_1, \ldots, s_p) \]
Factorization

\[ W = R_{2F \times 3} S_{3 \times P} \]

SVD

\[ W = U A V \quad A \text{ must be rank 3} \]

\[ W' = (U A^{1/2})(A^{1/2} V) = U' V' \]

Make \( R \) orthogonal

\[ R = QU' , \ S = Q'V' \]

\[ i_f^T Q^T Q i_f = 1 \dots \]

---

Results

- Look at paper figures…

---

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

\[
\begin{align*}
\hat{u}_{ij} &= f(K, R_j, t_j, x_i) \\
\hat{v}_{ij} &= g(K, R_j, t_j, x_i)
\end{align*}
\]

From Szeliski, *Computer Vision: Algorithms and Applications*

Two-frame methods

- Two main variants:
  - Calibrated: “Essential matrix” E
    use ray directions \((x_i, x'_i)\)
  - Uncalibrated: “Fundamental matrix” F

- [Hartley & Zisserman 2000]
Essential matrix

- Co-planarity constraint:
  - $x' \approx Rx + t$
  - $[t] \times x' \approx [t] \times Rx$
  - $x' [t] \times x' \approx x' [t] \times Rx$
  - $x' E x = 0$ 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 $F = [e] \times H = K' [t] \times R K^{-1}$

- Solve for $F$ using least squares (SVD)
  - re-scale $(x_i, x'_i)$ so that $|x_i| \approx 1/2$ [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*
D. Wedge, *The Fundamental Matrix Song*

**Cool Robotics Share**

![Image of a book titled "Multiple View Geometry"]

**Localization: SFM → SLAM**

![Image of a car with people inspecting it]
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 (e.g., 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

- Estimated robot
- Estimated landmark
- Correlations
Constrained Local Submap Filter

\[ x_G = \begin{bmatrix} x^G_F \\ m_G \end{bmatrix} \]

\[ x_R = \begin{bmatrix} x^R_v \\ m_R \end{bmatrix} \]

(a) Global map

(b) Local map

CLSF Registration

\[ x^G_v = x^G_F \oplus x^R_v \]
CLSF Global Estimate