SLAM Tutorial

Slides by Marios Xanthidis, C. Stachniss, P. Allen, C. Fermuller, Paul Furgale, Margarita Chli, Marco Hutter, Martin Rufli, Davide Scaramuzza, Roland Siegwart
What is the SLAM problem?

- The problem could be described in the following question:
  “If we leave a robot in an unknown location in an unknown environment can the robot make a satisfactory map while simultaneously being able to find its pose in that map?”

- The solution to this problem was the “Holy Grail” of the field of mobile robotics.
The SLAM Problem

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

- **Localization**: inferring location given a map
- **Mapping**: inferring a map given a location
- **SLAM**: learning a map and locating the robot simultaneously
The SLAM Problem

• SLAM is a **chicken-or-egg problem**:  
  → A map is needed for localizing a robot  
  → A pose estimate is needed to build a map

• Thus, SLAM is (regarded as) a **hard problem** in robotics
The SLAM Problem

**Given:**
- The robot’s controls
  \[ U_{0:k} = \{u_1, u_2, \cdots, u_k\} \]
- Relative observations

**Wanted:**
- Map of features
  \[ m = \{m_1, m_2, \cdots, m_n\} \]
- Path of the robot
  \[ X_{0:k} = \{x_0, x_1, \cdots, x_k\} \]
Structure of the Landmark-based SLAM-Problem
SLAM Applications

Indoors

Undersea

Space

Underground
Representations

• Grid maps or scans

[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;...]

• Landmark-based

[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;...]
Why is SLAM a hard problem?

**SLAM**: robot path and map are both *unknown*

Robot path error correlates errors in the map.
Why is SLAM a hard problem?

- In the real world, the mapping between observations and landmarks is unknown.
- Picking wrong data associations can have catastrophic consequences.
- Pose error correlates data associations.
The SLAM Problem (Difficulties)

• The bad news:
  – Pretty difficult problem because it combines the difficulties of both the localization and mapping problem without the essential assumptions of the known map or the known pose. Classic chicken or egg problem. Data Association problem also key.

• The good news:
  – The problem is considered solved but there are still some issues on having more general SLAM solutions and creating better maps.
Fig. 37.1 Graphical model of the SLAM problem. Arrows indicate causal relationships, and shaded nodes are directly observable to the robot. In SLAM, the robot seeks to recover the unobservable variables.
The SLAM Problem - Preliminaries(1)

- $X_k$: the state vector describing the location and orientation of the vehicle at time $k$.
- $u_k$: the control vector applied the time $k-1$.
- $m_i$: a vector describing the location of the $i^{th}$ landmark. The landmarks are motionless.
- $z_{ik}$: an observation taken from the vehicle of the location of the $i^{th}$ landmark at time $k$. 
The SLAM Problem- Preliminaries(2)

- Also the following sets are defined:
  - \( X_{0:k} = \{x_0, x_1, \ldots, x_k\} \) Position estimates
  - \( U_{0:k} = \{u_1, u_2, \ldots, u_k\} \) Motion update controls
  - \( Z_{0:k} = \{z_1, z_2, \ldots, z_k\} \) Sensor measurements
  - \( m = \{m_1, m_2, \ldots, m_n\} \) Landmark locations
SLAM: Simultaneous Localization and Mapping

• Full SLAM: Estimates entire path and map!

\[ p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \]

• Online SLAM: Integrations (marginalization) typically done one at a time

\[ p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \int \ldots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \, dx_1 \, dx_2 \ldots dx_{t-1} \]

Estimates most recent pose and map!
Full SLAM Problem

• We need to compute for all times $k$ the probabilistic distribution of the joint posterior density of the landmarks and the state:

$$P(x_k, m | Z_{0:k}, U_{0:k}, x_0)$$

Why do we need the $x_0$ element?

• In order to compute that probability we need to compute previously:

  – $P(z_k | x_k, m)$ (measurement model)
  – $P(x_k | x_{k-1}, u_k)$ (motion model)

• Online SLAM just calculates the most recent state based upon the previous states
Probabilistic SLAM - Updates

- **Time update (prediction):**
  \[
P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \int P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1}
\]

- **Measurement update (correction):**
  \[
P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m)P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})}
\]

- We can solve the localization problem with the assumption that we know the map:
  \[
P(x_k|Z_{0:k}, U_{0:k}, m)
\]

- And the mapping problem with the assumption we know the location:
  \[
P(m|X_{0:k}, Z_{0:k}, U_{0:k})
\]
Structure of Probabilistic SLAM(1)

• The landmark locations estimates are highly correlated. We may know with high accuracy the relation between the landmarks even if the absolute location is uncertain!

• The correlations are increased for every observations. Also the estimates for the relative location for every landmark are improved monotonically as more observations are made!
Structure of Probabilistic SLAM(2)
Structure of Probabilistic SLAM(3)

• The observation made by the robot regarding the relative location of the landmarks can be considered nearly independent, because the relative location of the landmarks is independent from the robot’s coordinate frame.
• The observation made by the robot regarding the absolute location of the landmarks is more uncertain because the absolute location of each landmark is strongly related to the robots coordinate frame.
• Because of the correlations of the landmarks we can update the location of landmarks even we cannot observe. So the correlations are increased for every observation we make.
• Thus, the robot ‘s accuracy on building the relative map of the environment increased for more observations.
Solutions to SLAM Problem

• The goal is to find an appropriate representation for the observation and motion problem.

• Three different methods:
  – Graph Slam
  – EKF-SLAM: Using the Extended Kalman Filter.
  – Using particle filters:
    • Rao-Blackwellized particle filter (FastSLAM)
Graph-Based SLAM ??

SLAM = simultaneous localization and mapping

graph = representation of a set of objects where pairs of objects are connected by links encoding relations between the objects
Graph-Based SLAM

- Nodes represent poses or locations
- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain

Robot pose

Constraint
Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses
Idea of Graph-Based SLAM

- Use a **graph** to represent the problem
- 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 a node configuration that minimize the error introduced by the constraints
Graph-SLAM and Least Squares

- The nodes represent the state
- Given a state, we can compute what we expect to perceive

Find a configuration of the nodes so that the real and predicted observations are as similar as possible
Create an Edge If... (1)

- ...the robot moves from $x_i$ to $x_{i+1}$
- Edge corresponds to odometry

The edge represents the **odometry** measurement
Create an Edge If... (2)

- the robot observes the same part of the environment from $x_i$ and from $x_j$.
The Graph with Landmarks

- **Nodes** can represent:
  - Robot poses
  - Landmark locations

- **Edges** can represent:
  - Landmark observations
  - Odometry measurements

- The minimization optimizes the **landmark locations and robot poses**
Graph Slam

- Treat constraints (motion and measurement) as “soft” elastic springs
- Want to minimize energy in the springs
SLAM overview

- Let us assume that the robot uncertainty at its initial location is zero.
- From this position, the robot observes a feature which is mapped with an uncertainty related to the exteroceptive sensor error model.
SLAM overview

- As the robot moves, its pose uncertainty increases under the effect of the errors introduced by the odometry.
SLAM overview

- At this point, the robot observes two features and maps them with an uncertainty which results from the combination of the measurement error with the robot pose uncertainty.
- From this, we can notice that the map becomes correlated with the robot position estimate. Similarly, if the robot updates its position based on an observation of an imprecisely known feature in the map, the resulting position estimate becomes correlated with the feature location estimate.
SLAM overview

- The robot moves again and its uncertainty increases under the effect of the errors introduced by the odometry.
In order to reduce its uncertainty, the robot must observe features whose location is relatively well known. These features can for instance be landmarks that the robot has already observed before.

In this case, the observation is called loop closure detection.

When a loop closure is detected, the robot pose uncertainty shrinks.

At the same time, the map is updated and the uncertainty of other observed features and all previous robot poses also reduce.
Linear 1-D Graph Slam

- Solution: List all constraints in a linear system
- Absolute constraint: \( X(0) = Q \) - starting position
- Movement Constraints: \( X(t) = X(t-1) + D_x(t) \)
- Measurement Constraints: Use distance measure to landmark:
  \[
  L(k) = X(t) + N
  \]
- Each constraint can be an estimate based upon a probability distribution
Linear 1-D Graph Slam

- For exact solution, we need 1 constraint per unknown
- Example: \( X(0) = -3; \ X(1) = X(0) + 5; \ X(2) = X(1) + 3 \)
- Each constraint is a linear equation in the unknowns

\[
\begin{bmatrix}
1 & 0 & 0 \\
-1 & 1 & 0 \\
0 & -1 & 1
\end{bmatrix}
\begin{bmatrix}
X(0) \\
X(1) \\
X(2)
\end{bmatrix} =
\begin{bmatrix}
-3 \\
5 \\
3
\end{bmatrix}
\]

\[
A \times X = B
\]

\[
X = A^{-1} \times B
\]

\[
X = [-3 \ 2 \ 5]
\]

Robot starts at \( X(0) \), moves 5 units to \( X(1) \), moves 3 units to \( X(2) \)
Now add Landmark (L0) constraints from sensing
Linear 1-D Graph Slam

- Now add landmark constraints, linking position with map:
- At $X(0)$ see $L(0)$ at distance 10; At $X(1)$ see $L(0)$ at distance 5; at $X(2)$ see $L(0)$ at distance 2

- Solution:

$$\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 \\
0 & 1 & -1 & 0 \\
1 & 0 & 0 & -1 \\
0 & 1 & 0 & -1 \\
0 & 0 & 1 & -1 \\
\end{bmatrix} \begin{bmatrix}
X(0) \\
X(1) \\
X(2) \\
L(0) \\
\end{bmatrix} = \begin{bmatrix}
-3 \\
-5 \\
-3 \\
-10 \\
-5 \\
-2 \\
\end{bmatrix}$$

$A \times X = B$

Cannot Invert $A$!

$X = A^{-1} \times B$

$X = [-3 \ 2 \ 5 \ 7]$
Linear 1-D Graph Slam

For overdetermined system, compute PseudoInverse Matrix:

\[ A \cdot X = B \]

\[ A^T \cdot A \cdot X = A^T \cdot B; \quad A^T = \text{transpose of } A \]

\[ X = (A^T \cdot A)^{-1} \cdot A^T \cdot B; \quad A^T A \text{ guaranteed to be square} \]

\( (A^T \cdot A)^{-1} \cdot A^T \) is PseudoInverse

Solution:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 \\
0 & 1 & -1 & 0 \\
1 & 0 & 0 & -1 \\
0 & 1 & 0 & -1 \\
0 & 0 & 1 & -1 \\
\end{bmatrix}
\begin{bmatrix}
X(0) \\
X(1) \\
X(2) \\
L(0) \\
\end{bmatrix}
=
\begin{bmatrix}
-3 \\
-5 \\
-3 \\
-10 \\
-5 \\
-2 \\
\end{bmatrix}
\]

\[ X = [-3 \ 2 \ 5 \ 7] \]
Matlab Code: Least Squares

A=[1 0 0 0; -1 1 0 0; 0 -1 1 0; 1 0 0 -1; 0 1 0 -1; 0 0 1 -1]
B=[-3; 5; 3; -10; -5; -2]

ATA = transpose(A)*A
ATB = transpose(A)*B
X= inv(ATA)*ATB;
disp('Solution:')
disp(X)

X(0) = -3; X(1) = 2; X(2) = 5; L(0) = 7
Consistent Solution: If L0 is at 7, each loop is consistent
Include an inconsistent measurement, find BEST solution

A=[1 0 0 0; -1 1 0 0; 0 -1 1 0; 1 0 0 -1; 0 1 0 -1; 0 0 1 -1]
B=[-3; 5; 3; -10; -5; -1]

Inconsistent measurement
Was X(2) +2 =L(0)
Now (X2) +1 =L(0)

ATA = transpose(A)*A
ATB = transpose(A)*B
X= inv(ATA)*ATB;
disp('Solution:')
disp(X)

Least squares shifts the robot locations and landmark location to reduce the overall error. Notice the locations and the landmark are adjusted! However X(0) Remains unchanged as it has an absolute constraint (X(0) = fixed location)
Inconsistent Measurement Solution: Adjust L0, X(1), X(2) to minimize error in measurements and movement. Now L0 = X(2) + 1.375, closer to measured value = 1. Note ALL relative values are shifted to reduce global error.
Adding Confidence Measures

- Linear Least Squares allows us to include a weighting of each linear constraint. If we know something about how confident a measure is, we can include that in the computation.
- We weight each constraint by a diagonal matrix where the weights are $1/variance$ for each constraint.
- Highly confident constraints have low variance; $1/variance$ is large weight.
- Unconfident constraints have high variance; $1/variance$ is small weight.
- Matrix Formulation for weighted Least Squares ($A*X=B$):

$$X = (A^T * W * A)^{-1} * A^T * W * B$$

Weights amplify/attenuate measurements based on measurements variance.
Matlab Code: Confidence Weights

Include a weight on a measurement, scaled by 1/variance

% x0= -3, x1= x0+5; x2=x1+3; xo sees L0 at 10; x1 sees L0 at 5, x2 sees L0 at 1
% we also have MUCH higher confidence (small variance) in % X(2) measurement of L0
% use weight matrix where diagonals are 1/sigma**2 – % variance in each measurement. Use variance of 0.2 (1/.2 =5)

A=[1 0 0 0; -1 1 0 0; 0 -1 1 0; 1 0 0 -1; 0 1 0 -1; 0 0 1 -1]
B=[-3; 5; 3; -10; -5; -1]
W=[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 0; 0 0 0 0 1 0; 0 0 0 0 0 5]

ATWA = transpose(A)*W*A
ATWB = transpose(A)*W*B
X= inv(ATWA)*ATWB;
disp('Solution:')
disp(X)

Note: Notice that L0 location from X(2) is now closer to the Measured value of 1 due to confidence weights
Confidence Measures Solution: Adjust $L_0$, $X(1)$, $X(2)$ to minimize error in measurements and movement, given greater confidence in measurement of $L_0$ from $X(2)$. Now $L_0 = X(2) + 1.107$, even closer to measured value = 1. Note ALL relative values are shifted to reduce global error.
Example: Multiple landmarks: Can incorporate multiple landmarks – each measurement is a constraint. In this example, X(0)=5, X(1)=X(0)+7, X(2)=X(1)+2; X(0) sees L0 at 2, X(1) sees L1 at 4 and X(2) sees L1 at 2.

Solution:
- X(0) = 5.0000
- X(1) = 12.0000
- X(2) = 14.0000
- L0 = 7.0000
- L1 = 16.0000

If X(1) sees L0 at 4 and X(2) sees L1 at 1.

Solution:
- X(0) = 5.0000
- X(1) = 12.0000
- X(2) = 14.3000
- L0 = 6.0000
- L1 = 15.6667
SLAM: EKF Slam

- Solution 1: Gaussian Filtering (EKF, UKF)
  - Track a Gaussian belief of the state/landmarks
  - Assume all noise is Gaussian
  - Follow the well-known “predict/correct” approach

- Pros
  - Runs online
  - Well understood
  - Works well for low-uncertainty problems

- Cons
  - Works poorly for high-uncertainty problems
  - Unimodal estimate
  - States must be well approximated by a Gaussian

Goal: estimate $p(x_{0:K}, m | u_{1:K}, z_{1:N})$
SLAM: The three main solutions

- **Solution 2: Particle Filtering**
  - Represent our belief by a series of samples
  - Follow the well-known “predict/correct” approach
- **Pros**
  - Noise densities can be from any distribution
  - Works for multi-modal distributions
  - Easy to implement
- **Cons**
  - Does not scale to high-dimensional problems
  - Requires many particles to have good convergence

Goal: estimate $p(x_{0:K}, m|u_{1:K}, z_{1:N})$
Particle Filter SLAM: FastSLAM

- **FastSLAM approach**
  - It solves the SLAM problem using particle filters.
  - Each particle $k$ : estimate of robot path and mean, and covariance of each of the $n$ features: $P[k] (X_t[k]; \mu[k]; \Sigma_1[k] \ldots \mu[k] \Sigma_n[k])$

- **Particle filter update**
  - In the update step a new particle distribution, given motion model and controls applied is generated.
    a) For each particle: 
      1. Compare particle’s prediction of measurements with actual measurements
      2. Particles whose predictions match the measurements are given a high weight
    b) Filter resample:
      - Resample particles based on weight
      - Filter resample
        - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements and randomly draw particles from previous distribution based on weights creating a new distribution.
Particle Filters

- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

- Sampling Importance Resampling (SIR) principle
  - Draw the new generation of particles
  - Assign an importance weight to each particle
  - Resampling

- Typical application scenarios are tracking, localization, ...
Localization vs. SLAM

- A particle filter can be used to solve both problems

- Localization: state space $<x, y, \theta>$

- SLAM: state space $<x, y, \theta, \text{map}>$
  - for landmark maps $= <l_1, l_2, ..., l_m>$
  - for grid maps $= <c_{11}, c_{12}, ..., c_{1n}, c_{21}, ..., c_{nm}>$

- **Problem:** The number of particles needed to represent a posterior grows exponentially with the dimension of the state space!
Dependencies

- Is there a dependency between the dimensions of the state space?
- If so, can we use the dependency to solve the problem more efficiently?
Dependencies

- Is there a dependency between the dimensions of the state space?
- If so, can we use the dependency to solve the problem more efficiently?

- In the SLAM context
  - The map depends on the poses of the robot.
  - We know how to build a map given the position of the sensor is known.
Factored Posterior (Landmarks)

\[
p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) = p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t})
\]

poses, map, observations & movements

Does this help to solve the problem?

SLAM posterior

Robot path posterior

landmark positions

Factorization first introduced by Murphy in 1999
Knowledge of the robot’s true path renders landmark positions conditionally independent.
Factored Posterior

\[ p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) = p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot p(l_{1:m} \mid x_{1:t}, z_{1:t}) \]

\[ = p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_{i} \mid x_{1:t}, z_{1:t}) \]

Robot path posterior
(localization problem)

Conditionally independent
landmark positions
Rao-Blackwellization

\[ p(x_{1:t}, l_{1:m} \mid z_{1:t}, u_{0:t-1}) = \]
\[ p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \cdot \prod_{i=1}^{M} p(l_i \mid x_{1:t}, z_{1:t}) \]

- This factorization is also called Rao-Blackwellization
- Given that the second term can be computed efficiently, particle filtering becomes possible!
FastSLAM

- Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain $M$ EKFs
FastSLAM – Action Update

Particle #1

Particle #2

Particle #3

Landmark #1 Filter

Landmark #2 Filter
FastSLAM – Sensor Update

Particle #1

Particle #2

Particle #3

Landmark #1
Filter

Landmark #2
Filter
FastSLAM – Sensor Update

Particle #1

Particle #2

Particle #3

Weight = 0.8

Weight = 0.4

Weight = 0.1
FastSLAM - Video
FastSLAM Complexity

- Update robot particles based on control $u_{t-1}$

- Incorporate observation $z_t$ into Kalman filters

- Resample particle set

$N = \text{Number of particles}$
$M = \text{Number of map features}$
Results – Victoria Park

- 4 km traverse
- < 5 m RMS position error
- 100 particles

Blue = GPS
Yellow = FastSLAM

Dataset courtesy of University of Sydney
Data Association Problem

- Which observation belongs to which landmark?

- A robust SLAM must consider possible data associations

- Potential data associations depend also on the pose of the robot
Multi-Hypothesis Data Association

- Data association is done on a per-particle basis

- Robot pose error is factored out of data association decisions
Per-Particle Data Association

Was the observation generated by the red or the blue landmark?

\[ P(\text{observation}|\text{red}) = 0.3 \]
\[ P(\text{observation}|\text{blue}) = 0.7 \]

- Two options for per-particle data association
  - Pick the most probable match
  - Pick an random association weighted by the observation likelihoods
- If the probability is too low, generate a new landmark
Results – Data Association

Comparison of FastSLAM and EKF Given Motion Ambiguity

Robot RMS Position Error (m)

Error Added to Rotational Velocity (std.)
Results – Accuracy

Accuracy of FastSLAM vs. the EKF on Simulated Data

RMS Pose Error (meters)

Number of Particles
SLAM Challenges | components for scalable SLAM

1. Robust local motion estimation

2. Mapping & loop-closure detection

3. Map optimisation

4. Long-term map management & localization

[Chli, 2009, PhD thesis]
SLAM ++ - Real Time Object Level Slam
Moving camera same as moving robot
Use kinect Fusion to create 3D models
Detect semantic objects from Kinect 3D models
Objects are now "tracked" - higher level landmark with constraints
Can even add Virtual Objects to real scene!
see the video