Introduction to Kalman Filters
Overview

• The Problem – Why do we need Kalman Filters?
• What is a Kalman Filter?
• Conceptual Overview
• The Theory of Kalman Filter
• Simple Example
The Problem

- System state cannot be measured directly
- Need to estimate “optimally” from measurements
What is a Kalman Filter?

• **Recursive** data processing algorithm
• Generates **optimal** estimate of desired quantities given the set of measurements

• Optimal?
  – For linear system and white Gaussian errors, Kalman filter is “best” estimate based on all previous measurements
  – For non-linear system optimality is ‘qualified’

• Recursive?
  – Doesn’t need to store all previous measurements and reprocess all data each time step
Conceptual Overview

• Simple example to motivate the workings of the Kalman Filter
• Theoretical Justification to come later – for now just focus on the concept
• Important: Prediction and Correction
Conceptual Overview

- Lost on the 1-dimensional line
- Position – $y(t)$
- Assume Gaussian distributed measurements
Conceptual Overview

- Sextant Measurement at $t_1$: Mean = $z_1$ and Variance = $\sigma_{z1}$
- Optimal estimate of position is: $\hat{y}(t_1) = z_1$
- Variance of error in estimate: $\sigma^2_x(t_1) = \sigma^2_{z1}$
- Boat in same position at time $t_2$ - **Predicted** position is $z_1$
So we have the prediction $\hat{y}(t_2)$

GPS Measurement at $t_2$: Mean = $z_2$ and Variance = $\sigma_{z_2}$

Need to correct the prediction due to measurement to get $\hat{y}(t_2)$

Closer to more trusted measurement – linear interpolation?
Corrected mean is the new optimal estimate of position
New variance is smaller than either of the previous two variances
Conceptual Overview

• Lessons so far:

Make prediction based on previous data - $\hat{y}^{-}, \sigma^{-}$

Take measurement – $z_k, \sigma_z$

Optimal estimate ($\hat{y}$) = Prediction + (Kalman Gain) * (Measurement - Prediction)

Variance of estimate = Variance of prediction * (1 – Kalman Gain)
Conceptual Overview

- At time $t_3$, boat moves with velocity $\frac{dy}{dt}=u$
- Naïve approach: Shift probability to the right to predict
- This would work if we knew the velocity exactly (perfect model)
Conceptual Overview

- Better to assume imperfect model by adding Gaussian noise
- \( \frac{dy}{dt} = u + w \)
- Distribution for prediction moves and spreads out
• Now we take a measurement at $t_3$
• Need to once again correct the prediction
• Same as before
Conceptual Overview

• Lessons learnt from conceptual overview:
  – Initial conditions ($\hat{y}_{k-1}$ and $\sigma_{k-1}$)
  – Prediction ($\hat{y}^{-}_{k}$, $\sigma^{-}_{k}$)
    • Use initial conditions and model (eg. constant velocity) to make prediction
  – Measurement ($z_{k}$)
    • Take measurement
  – Correction ($\hat{y}_{k}$, $\sigma_{k}$)
    • Use measurement to correct prediction by ‘blending’ prediction and residual – always a case of merging only two Gaussians
    • Optimal estimate with smaller variance
Kalman Filter Model

Area under the curve sums to 1

\[ g(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \]

Gaussian

(in 1D for now)
Measurement Example

Prior Position Estimate $\mu, \sigma^2$
Measurement Estimate $v, r^2$
Measurement Example

Prior Position Estimate \( \mu, \sigma^2 \)
Measurement Estimate \( \nu, r^2 \)

Where is the new mean \( \mu' \)?
Measurement Example

Prior Position Estimate $\mu, \sigma^2$

Measurement Estimate $\nu, r^2$

What is the new covariance $\sigma^2'$?

**Definition.** The **variance** of a random variable $X$ with expected value $\mathbb{E}X = \mu_X$ is defined as $\text{var}(X) = \mathbb{E}((X - \mu_X)^2)$. The **covariance** between random variables $Y$ and $Z$, with expected values $\mu_Y$ and $\mu_Z$, is defined as $\text{cov}(Y, Z) = \mathbb{E}((Y - \mu_Y)(Z - \mu_Z))$. The **correlation** between $Y$ and $Z$ is defined as

$$\text{corr}(Y, Z) = \frac{\text{cov}(Y, Z)}{\sqrt{\text{var}(Y)\text{var}(Z)}}$$

The square root of the variance of a random variable is called its **standard deviation**. $\Box$
What is the new estimate?

To calculate, go through and multiply the two Gaussians and renormalize to sum to 1
Also, the multiplication of two Gaussian random variables is itself a Gaussian

Prior Position Estimate $\mu, \sigma^2$ : $p(x)$
Measurement Estimate $\nu, r^2$ : $p(z|x)$
New Estimate $\mu', \sigma'^2$ : $p(x|z)$
Example

Prior Position Estimate (10,4)
Measurement Estimate (12,4)

\[ \mu' = \frac{xr^2 + v\sigma^2}{\sigma^2 + r^2} \]

\[ \sigma^2' = \left( \frac{\sigma^2 + r^2}{\sigma^2 r^2} \right) \]
Example

Prior Position Estimate \((10,4)\)
Measurement Estimate \((12,4)\)

\[
\mu' = \left( \frac{x r^2 + v \sigma^2}{\sigma^2 + r^2} \right) = 11
\]

\[
\sigma'^2 = \left( \frac{\sigma^2 + r^2}{\sigma^2 r^2} \right) = 8
\]
Example

Prior Position Estimate (10,8)
Measurement Estimate (13,2)

$\mu' = 12.4$
$\sigma'^2 = 1.6$
Kalman Filter

\[ \mu' = \left( \frac{xr^2 + v\sigma^2}{\sigma^2 + r^2} \right) \]

\[ \sigma'^2 = \left( \frac{\sigma^2 + r^2}{\sigma^2 r^2} \right) \]
Motion Update

• For motion

\[ \mu' = \mu + u \]

\[ \sigma^2' = \sigma^2 + r^2 \]
Motion Update

• For motion

\[ \mu' = \mu + u \]

\[ \sigma'^2 = \sigma^2 + r^2 \]

Prior Position Estimate \((8,4)\)

Movement Estimate \((10, 6)\)

Model of motion noise

\(u, r^2\)
Motion Update

- For motion

\[ \mu' = \mu + u = 18 \]

\[ \sigma'^2 = \sigma^2 + r^2 = 10 \]

Prior Position Estimate (8, 4)

Movement Estimate (10, 6)

Model of motion noise

\[ u, r^2 \]
Kalman Filter

\[
\mu' = \left( \frac{x r^2 + v \sigma^2}{\sigma^2 + r^2} \right)
\]

\[
\sigma'^2 = \left( \frac{\sigma^2 + r^2}{\sigma^2 r^2} \right)
\]

Initial Belief

Sense

Gaussian: \( \mu, \sigma^2 \)

Move

\( \mu' = \mu + u = 18 \)

\( \sigma'^2 = \sigma^2 + r^2 = 10 \)
Theoretical Basis

• Process to be estimated:

\[ y_k = Ay_{k-1} + Bu_k + w_{k-1} \]

Process Noise \((w)\) with covariance \(Q\)

\[ z_k = Hy_k + v_k \]

Measurement Noise \((v)\) with covariance \(R\)

• Kalman Filter

Predicted: \(\hat{y}^-_k\) is estimate based on measurements at previous time-steps

\[
\hat{y}^-_k = Ay_{k-1} + Bu_k \\
P^-_k = AP_{k-1}A^T + Q
\]

Corrected: \(\hat{y}_k\) has additional information – the measurement at time \(k\)

\[
\hat{y}_k = \hat{y}^-_k + K(z_k - H \hat{y}^-_k) \\
K = P^-_kH^T(HP^-_kH^T + R)^{-1} \\
P_k = (I - KH)P^-_k
\]
Blending Factor

• If we are sure about measurements:
  – Measurement error covariance (R) decreases to zero
  – $K$ increase and weights residual(measurement) more heavily than prediction

• If we are sure about prediction
  – Prediction error covariance $P^{-1}_k$ decreases to zero
  – $K$ decreases and weights prediction more heavily than residual
Theoretical Basis

### Prediction (Time Update)

1. Project the state ahead
   \[ \hat{y}^-_k = A\hat{y}_{k-1} + B u_k \]
2. Project the error covariance ahead
   \[ P^-_k = A P_{k-1} A^T + Q \]

### Correction (Measurement Update)

1. Compute the Kalman Gain
   \[ K = P^-_k H^T (HP^-_k H^T + R)^{-1} \]
2. Update estimate with measurement \( z_k \)
   \[ \hat{y}_k = \hat{y}^-_k + K (z_k - H \hat{y}^-_k) \]
3. Update Error Covariance
   \[ P_k = (I - KH)P^-_k \]
Quick Example – Constant Model

- **External Controls**
  - System Error Sources
  - System
  - System State
  - Measuring Devices
  - Measurement Error Sources
  - Observed Measurements
  - Estimator
  - Optimal Estimate of System State
  - Black Box
Quick Example – Constant Model

Prediction

\[ \hat{y}_{k} = y_{k-1} \]
\[ P^{-}_k = P^{-}_{k-1} \]

Correction

\[ K = P^{-}_k(P^{-}_k + R)^{-1} \]
\[ \hat{y}_k = \hat{y}^{-}_k + K(z_k - H \hat{y}^{-}_k) \]
\[ P_k = (I - K)P^{-}_k \]
Quick Example – Constant Model
Quick Example – Constant Model

Convergence of Error Covariance - $P_k$
Quick Example – Constant Model

Larger value of $R$ – the measurement error covariance (indicates poorer quality of measurements)

Filter slower to ‘believe’ measurements – slower convergence
SLAM

• Simultaneous localization and mapping:

  *Is it possible for a mobile robot to be placed at an unknown location in an unknown environment and for the robot to incrementally build a consistent map of this environment while simultaneously determining its location within this map?*
Figure 2 The robot is represented by the triangle. The stars represent landmarks. The robot initially measures using its sensors the location of the landmarks (sensor measurements illustrated with lightning).
Figure 3  The robot moves so it now thinks it is here. The distance moved is given by the robots odometry.
Figure 4 The robot once again measures the location of the landmarks using its sensors but finds out they don’t match with where the robot thinks they should be (given the robot’s location). Thus the robot is not where it thinks it is.
Figure 5 As the robot believes more its sensors than its odometry it now uses the information gained about where the landmarks actually are to determine where it is (the location the robot originally thought it was at is illustrated by the dashed triangle).
Figure 6 In actual fact the robot is here. The sensors are not perfect so the robot will not precisely know where it is. However this estimate is better than relying on odometry alone. The dotted triangle represents where it thinks it is; the dashed triangle where odometry told it it was; and the last triangle where it actually is.
Intuition: SLAM without Landmarks

Using only dead reckoning, vehicle pose uncertainty (and thus the uncertainty of map features) grows without bound.
With Landmark Measurements

- First position: two features observed
Illustration of SLAM with Landmarks

- Second position: two new features observed
Illustration of SLAM with Landmarks

- Re-observation of first two features results in improved estimates of both vehicle pose and features
Illustration of SLAM with Landmarks

- Third measurement: two additional features are added to the map
Illustration of SLAM with Landmarks

- Re-observation of first four features results in improved location estimates for vehicle poses and all map features
Illustration of SLAM with Landmarks

- Process continues as the vehicle moves through the environment
Three Basic Steps

• The robot moves
  – increases the uncertainty on robot pose
  – need a mathematical model for the motion
  – called *motion model*
Three Basic Steps

• The robot discovers interesting features in the environment
  – called landmarks
  – uncertainty in the location of landmarks
  – need a mathematical model to determine the position of the landmarks from sensor data
  – called inverse observation model
Three Basic Steps

• The robot *observes previously mapped landmarks*
  – uses them to correct both self localization and the localization of all landmarks in space
  – uncertainties decrease
  – need a model to predict the measurement from predicted landmark location and robot localization
  – called *direct observation model*
How to do SLAM

- Use internal representations for
  - the positions of landmarks (map)
  - the camera parameters

- Assumption: Robot's uncertainty at starting position is zero

Start: robot has zero uncertainty
How to do SLAM

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

First measurement of feature A

© R. Siegwart, D. Scaramuzza and M. Chli, ETH Zurich - ASL
How to do SLAM

- The robot observes a feature which is mapped with an uncertainty related to the measurement model

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations
How to do SLAM

- As the robot moves, its pose uncertainty increases, obeying the robot’s **motion model**.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot moves forwards: uncertainty grows

© R. Siegwart, D. Scaramuzza and M. Chli, ETH Zurich - ASL
How to do SLAM

- Robot observes two new features.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot makes first measurements of B & C
How to do SLAM

- Their position uncertainty results from the combination of the measurement error with the robot pose uncertainty.
- Map becomes correlated with the robot pose estimate.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot makes first measurements of B & C

© R. Siegwart, D. Scaramuzza and M. Chli, ETH Zurich - ASL
How to do SLAM

- Robot moves again and its uncertainty increases (motion model)

On every frame:
- **Predict** how the robot has moved
- Measure
- **Update** the internal representations

Robot moves again: uncertainty grows more
How to do SLAM

- Robot re-observes an old feature
  ⇒ Loop closure detection

On every frame:
- Predict how the robot has moved
- Measure
- Update the internal representations

Robot re-measures A: “loop closure”
How to do SLAM

- Robot updates its position: the resulting position estimate becomes correlated with the feature location estimates.
- Robot’s uncertainty shrinks and so does the uncertainty in the rest of the map.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

Robot re-measures A: “loop closure” uncertainty shrinks

© R. Siegwart, D. Scaramuzza and M. Chli, ETH Zurich - ASL
The Essential SLAM Problem
SLAM – Multiple parts

• Landmark extraction
• data association
• State estimation
• state update
• landmark update

There are many ways to solve each of the smaller parts
Hardware

- Mobile Robot
- Range Measurement Device
  - Laser scanner – CANNOT be used underwater
  - Sonar – NOT accurate
  - Vision – Cannot be used in a room with NO light
The goal of the process

The SLAM process consists of number of steps.

- Use environment to update the position of the robot. Since the odometry of the robot is often erroneous we cannot rely directly on the odometry.
- We can use laser scans of the environment to correct the position of the robot.
- This is accomplished by extracting features from the environment and re observing when the robot moves around.
Extended Kalman Filter

An EKF (Extended Kalman Filter) is the heart of the SLAM process.

- It is responsible for updating where the robot thinks it is based on the Landmarks (features).

- The EKF keeps track of an estimate of the uncertainty in the robots position and also the uncertainty in these landmarks it has seen in the environment.
Laser & Odometry data

• Laser data is the reading obtained from the scan
• The goal of the odometry data is to provide an approximate position of the robot
Landmarks

• Landmarks are features which can easily be re-observed and distinguished from the environment.

• These are used by the robot to find out where it is (to localize itself).
The key points about suitable Landmarks

- Landmarks should be easily re-observable.
- Individual landmarks should be distinguishable from each other.
- Landmarks should be plentiful in the environment.
- Landmarks should be stationary.
In an indoor environment such as that used by our robot there are many straight lines and well defined corners. These could all be used as landmarks.
Landmark Extraction

• Once we have decided on what landmarks a robot should utilize we need to be able to somehow reliably extract them from the robots sensory inputs.

• The 2 basic Landmark Extraction Algorithms used are Spikes and RANSAC
RANSAC (Random Sampling Consensus)

• This method can be used to extract lines from a laser scan that can in turn be used as landmarks.

• RANSAC finds these line landmarks by randomly taking a sample of the laser readings and then using a least squares approximation to find the best fit line that runs through these readings.
Data Association

• The problem of data association is that of matching observed landmarks from different (laser) scans with each other.

• Problems in Data Association
  – You might not re-observe landmarks every time.
  – You might observe something as being a landmark but fail to ever see it again.
  – You might wrongly associate a landmark to a previously seen landmark.
Algorithm – Nearest Neighbour Approach

• When you get a new laser scan use landmark extraction to extract all visible landmarks.
• Associate each extracted landmark to the closest landmark we have seen more than N times in the database.
• Pass each of these pairs of associations (extracted landmark, landmark in database) through a validation gate.
  – If the pair passes the validation gate it must be the same landmark we have re-observed so increment the number of times we have seen it in the database.
  – If the pair fails the validation gate add this landmark as a new landmark in the database and set the number of times we have seen it to 1.
Overview of the process

• Update the current state estimate using the odometry data
• Update the estimated state from re-observing landmarks.
• Add new landmarks to the current state.
Final Review – Open Areas

• There is the problem of closing the loop. This problem is concerned with the robot returning to a place it has seen before. The robot should recognize this and use the new found information to update the position.

• Furthermore the robot should update the landmarks found before the robot returned to a known place, propagating the correction back along the path.
SLAM Paradigms

• Some of the most important approaches to SLAM:
  – Extended Kalman Filter SLAM (EKF SLAM)
  – Particle Filter SLAM (FAST SLAM)
  – GraphSLAM
EKF Slam

- Keep track of combined state vector at time $t$: 
  - $x$, $y$, $\theta$
  - $m_{1,x}$, $m_{1,y}$, $s_1$
  - ...
  - $m_{N,x}$, $m_{N,y}$, $s_N$

- $m$ = estimated coordinates of a landmark
- $s$ = sensor’s signature for this landmark

- Very similar to EKF localization, starting at origin
EKF-SLAM

Grey: Robot Pose Estimate
White: Landmark Location Estimate
Visual Slam

• Single Camera
• What’s harder?
• How could it be possible?
GraphSLAM

- SLAM can be interpreted as a sparse graph of nodes and constraints between nodes.
GraphSLAM

- SLAM can be interpreted as a sparse graph of nodes and constraints between nodes.
- **nodes**: robot locations and map-feature locations
- **edges**: constraints between
  - consecutive robot poses (given by the odometry input $u$)
  - robot poses and the features observed from these poses.
GraphSLAM

- Key property: constraints are not to be thought as rigid constraints but as soft constraints
  - Constraints acting like *springs*

- Solve full SLAM by relaxing these constraints
  - get the best estimate of the robot path and the environment map by computing the **state of minimal energy** of this spring mass network
GraphSLAM

a) Observation is landmark $m_1$
b) Robot motion from $x_1$ to $x_2$
GraphSLAM
GraphSLAM

1. Build graph
2. Inference: solve system of linear equations to get map and path
GraphSLAM

• The update time of the graph is constant.
• The required memory is linear in the number of features.
• Final graph optimization can become computationally costly if the robot path is long.
• Impressing results with even hundred million features.
SLAM Problem Statement

• Inputs:
  – No external coordinate reference
  – Time series of proprioceptive and exteroceptive measurements* made as robot moves through an *initially unknown* environment

• Outputs:
  • A map* of the environment
  • A robot *pose estimate* associated with each measurement, in the coordinate system in which the map is defined
What is a map?

• Collection of features with some relationship to one another
• What is a feature?
  – Occupancy grid cell
  – Line segment
  – Surface patch
• What is a feature relationship?
  – Rigid-body transform (metrical mapping)
  – Topological path (chain of co-visibility)
  – Semantics (label, function, contents)
Why is SLAM Hard?

• “Grand challenge”-level robotics problem
  – Autonomous, persistent, collaborative robots mapping multi-scale, generic environments

• Map-making = learning
  – Difficult even for humans
  – Even skilled humans make mapping mistakes

• Scaling issues
  – Space: Large extent (combinatorial growth)
  – Time: Persistent autonomous operation

• “Chicken and Egg” nature of problem
  – If robot had a map, localization would be easier
  – If robot could localize, mapping would be easier
  – ... But robot has neither; starts from blank slate
  – Must also execute an exploration strategy

• Uncertainty at every level of problem
### Uncertainty in Robotic Mapping

<table>
<thead>
<tr>
<th>Scale:</th>
<th>Uncertainty:</th>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>Sensor noise</td>
<td>Data association</td>
<td></td>
</tr>
<tr>
<td>Global</td>
<td>Navigation drift</td>
<td>Loop closing</td>
<td></td>
</tr>
</tbody>
</table>
Common range-and-bearing sensors

Polaroid sonar ring
12 range returns,
one per 30 degrees, at ~4 Hz

SICK laser scanner
180 range returns,
one per degree,
at 5-75 Hz

Other possibilities: Stereo/monocular vision; Robot itself (stall, bump sensing)
Tracking & long-baseline monocular vision

Track points, edges, texture patches from frame to frame; triangulate to recover local 3D structure. Also called "SFM," Structure From camera Motion, or object motion in the image.
Example

- SLAM with laser scanning
- Observations
- Local mapping
  - Iterated closest point
- Loop closing
  - Scan matching
  - Deferred validation
  - Search strategies
Observations
Observations

1. Max deviant

2. Max deviant

3. Max deviant
Scan Matching

- Robot scans, moves, scans again
- Short-term odometry/IMU error causes misregistration of scans
- *Scan matching* is the process of bringing scan data into alignment
Iterated Closest Point

- Find the transformation that best aligns the matching sets of points

What happens to the estimate of the relative vehicle pose between sensor frames 1 & 2?
Limitations / failure modes

• Computational cost (two scans of size $n$)
  – Naively, $O(n^2)$ plus cost of alignment step

• False minima
  – If ICP starts far from true alignment
  – If scans exhibit repeated local structure

• Bias
  – Anisotropic point sampling
  – Differing sensor fields of view (occlusion)

• Lots of research on improved ICP methods (see, e.g., Rusinkiewicz)
Loop Closing

- Naive ICP ruled out:
  - Too CPU-intensive
- Assume we have a pose uncertainty bound
- This limits the portion of the existing map that must be searched
- Still have to face the problem of matching two partial scans that are far from aligned
Loop Closing Ambiguity

• Consider SLAM state after ABC ... XY
  Large open-loop navigation uncertainty
  Y matches both A & B
  ... What to do?
Deferred Loop Validation

- Continue SLAM until Z matches C
- Examine graph for \textit{\sim identity cycle}
Summary

• SLAM is a hard robotics problem:
  – Requires sensor fusion over large areas
  – Scaling issues arise quickly with real data

• Key issue is managing *uncertainty*
  – At both low level and high level
  – Both continuous and discrete

• Saw several SLAM strategies
  – Local and global alignment
  – Randomization
  – Deferred validation

• SLAM is only part of the solution for most applications (need names, semantics)