SLAM for dummies

About SLAM

DEFINITION: SLAM is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment using the map.

SLAM consists of multiple parts: Landmark extraction, data association, state estimation, state update and landmark update etc. It is more like a concept instead of an algorithm. Each single part of it can be implemented by different methods.

Hardware

To do SLAM, there is a need for the hardware composed of two parts:

- a mobile robot;
- range measurement devices.

There are several options for the range measurement devices. Here 3 tpyes of measurement devices are introduces.

1. Laser scanner.

   Advantages: precise, efficcient and less compution.

   Disadvantages: expensive. Bad reading when looking at certain surface like glass.

2. Sonar.

   Advantages: cheap. Able to be used under water.

   Disadvantages: less precious.

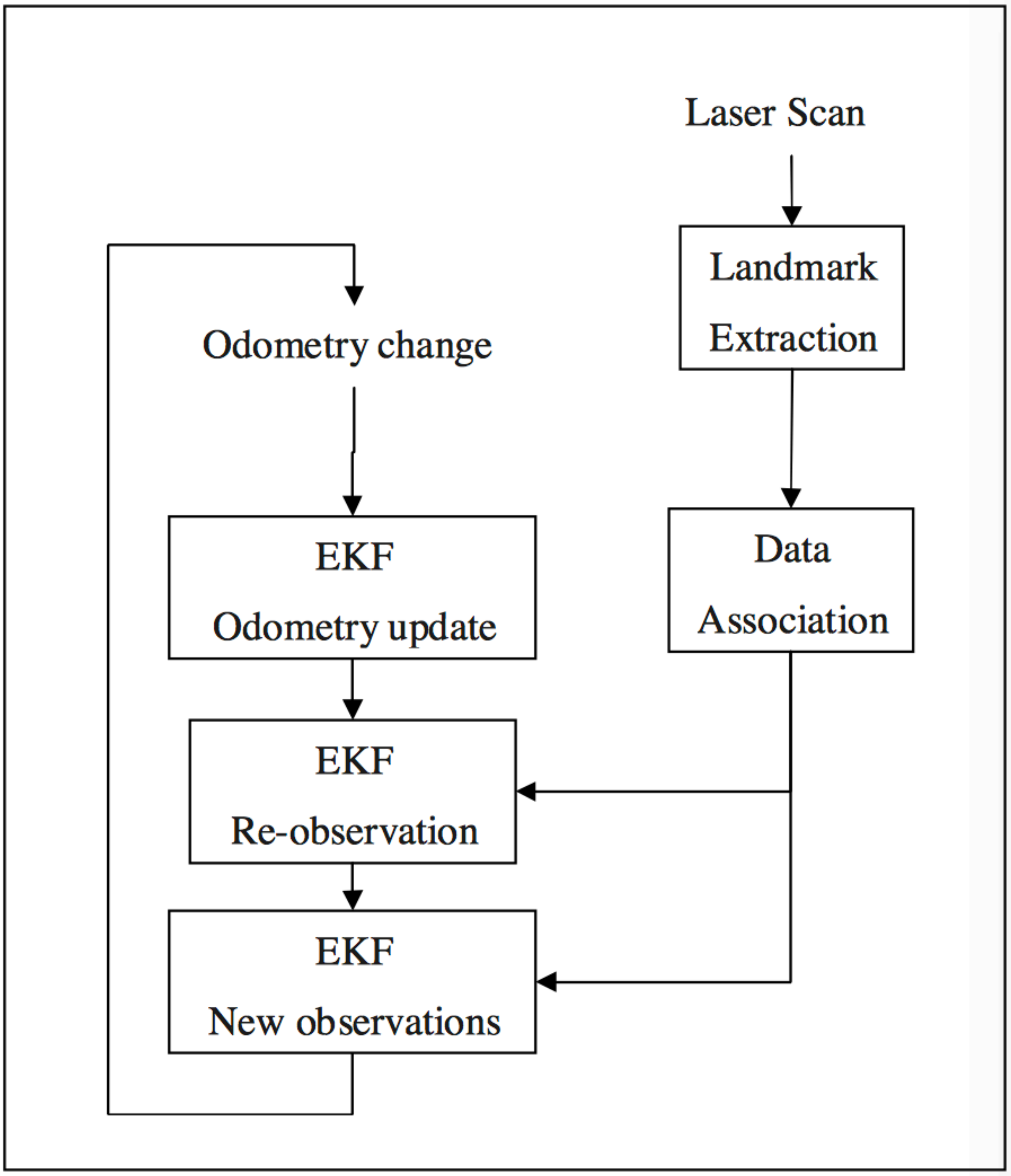
3. Visual camera.

   Advantages: simulate human vision. More information.

   Disadvantages: computationally intensive. Error prone due to changes in light.

The SLAM Process

One of the goal of the process is to use the environment to update the position of the robot. Since the odometry of the robot is no precise enough, we can use laser scans of the environment to correct the position of the robot. An EKF is responsible for updating where the robot thinks it is based on these features, which is called landmarks. **The EKF keeps track of an estimate of the uncertainty in 1) the robots position and in 2) these landmarks it has seen in the environment**.



Explaination of the steps in the figures:

1. When the odometry changes because of the robot moves, the uncertainty pertaining to the robot's new position is updated in the EKF using odometry update;
2. Landmarks are then extracted from the environment from the robot's new position;
3. the robot attempts to associate these landmarks to observations of landmarks it previously has observed;
4. Re-observed landmarks are used to update the robot's position in the EKF; landmarks which is newly seen are added to the EKF as new observations so that they can be re-observed later.

NOTE: At any point in these setps the EKF will have *an estimate* of the robot's current position.

Data

Laser Data

The first step in the SLAM process is to obtain data about the surroundings of the robot. The output from the laser scanner tells the ranges from right to left in terms of meters.

If the laser scanner for some reason cannot tell the exact length for a specific angle it will return a specific high value.

Odometry Data

The **goal** of the odometry data is to **provide an approximate position of the robot**, to serve as the **initial guess** of where the robot might be in the EKF.

The **difficult part** about the odometry data and the laser data is to get the timing right. To make sure they are valid at the same time one can **extrapolate the data**. It is easiest to extrapolate the odometry data since the controls are known.

Landmarks

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

Obviously, the type of landmarks a robot uses will often depend on the environment in which the robot is operating.

Requirements for the landmarks:

1. Landmarks should be re-observable by allowing them to be viewed (detected) from different positions and thus from different angles.
2. Landmarks should be unique enough so that they can be easily identified from one time-step to another without mixing them up.
3. Of course, the landmarks should be stationary.

Landmark Extraction

After the determination of what kinds of landmarks would be used, we need to be able to reliably extract them from input. Here, two basic landmark extraction algorithms using a laser scanner are introduced.

Spike landmarks

The spike landmark extraction uses extrema to find landmarks. They are identified by finding values in the range of a laser scan where two values differ by more than the threshold, e.g. 0.5 meters.

The spikes can also be found by having three values next to each other, A, B and C. Subtracting B from A and B from C and adding the two numbers will yield a value.

However, Spike landmarks rely on the landscape changing a lot between two laser beams. The algorithm will fail in smooth environments.

RANSAC

RANSAC (Random Sampling Consensus) is a method which can be used to extract **lines** from a laser scan. These lines can in turn be used as landmarks.

Step:

1. RANSAC randomly takes a sample of the laser readings and then using a least squares approximation to find the best fit line that runs through these readings.
2. Then it checks how many laser readings lie close to this best fit line. If the number is above some threshold(consensus) we can safely assume that we have seen a line.

Pseudo code:

```
While
    • there are still unassociated laser readings,
    • and the number of readings is larger than the consensus,
    • and we have done less than N trials.
do
    • Select a random laser data reading.
    • Randomly sample S data readings within D degrees of this laser data reading (for
      example, choose 5 sample readings that lie within 10 degrees of the randomly
      selected laser data reading).
    • Using these S samples and the original reading calculate a least squares best fit line.
    • Determine how many laser data readings lie within X centimeters of this best fit line.
    • If the number of laser data readings on the line is above some consensus C do the
      following:
        ◦ calculate new least squares best fit line based on all the laser readings
          determined to lie on the old best fit line.
        ◦ Add this best fit line to the lines we have extracted.
        ◦ Remove the number of readings lying on the line from the total set of
          unassociated readings.
```

So there are 5 parameters that can be tuned:

- *N* - Max iteration;
- *S* - Number of sample to compute initial line;
- *D* - Degree from initial reading to sample from;
- *X* - Threshold to determine whether a point belongs to the line;
- *C* - Consensus.

The EKF implementation assumes that landmarks come in as **a range and bearing** from the robots position.

A line can be translated into **a fixed point** by **1)** taking one fixed point in the world coordinates and **2)** calculating the point on the line closest to this fixed point. Using **the robots position** and **the position of this fixed point on the line** it is trivial to calculate a range and bearing from this.

Remarks

Spikes landmarks is fairly simple and is not robust against environments with people.

RANSAC uses line extraction and is robust against people. But it is also more complicate.

And there are other methods can be used, such as scan-matching that tries to match two successive laser scans.

Data Association

OBJECT: matching observed landmarks from different (laser) scans with each other.

Problems:

- might not re-observe landmarks every time step;
- might observe something as being a landmark but fail to ever see it again;
- might wrongly associate a landmark to a previously seen landmark.

The first two cases above are not acceptable for a landmark.

Data-association policy:

the first rule is that we don't actually consider a landmark worthwhile to be used in SLAM unless we have seen it *N* times. The technique is called the near-neighbor approach.

Pesudo code:

> 1. When you get a new laser scan, use landmark extraction to extract all visible landmarks.
> 2. Associate each extracted landmark to the **closest landmark** we have seen more than *N* times in the database.
> 3. Pass each of these pairs of associations (extracted landmark, landmark in database) through a validation gate.
>   1. 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.
>   2. 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.

The simplest way to calculate the nearest landmark is to calculate the **Euclidean distance**.

The validation gate uses the fact that our EKF implementation gives a bound on the uncertainty of an observation of a landmark. Thus we can determine if an observed landmark is a landmark in the database by checking if the landmark lies within the area of uncertainty.

By setting a constant an observed landmark is associated to a landmark if the following formula holds:

$$\nu_i^T \cdot S_i^{-1} \cdot \nu_i \leq \lambda \tag{1}$$

The EKF

The Extended Kalman Filter is used to estimate the state (position) of the robot from odometry data and landmark observations.

Overview

As soon as the landmark extraction and the data association is in place the SLAM process can be considered as **three steps**:

1. Update the current state estimate using the odometry data;
2. Update the estimated state from re-observing landmarks;
3. Add new landmarks to the current state;

The first step is an addition of the controls of the robot to the old state estimate.

In **the second step** the re-observed landmarks are considered. Here, the **innovation** is the difference between the estimated positions of the re-observed landmarks based on current states of the robot and the observed positions of them. (In the inverse way, the innovation is the difference between the estimated robot position and the "actual" robot position based on the re-observed landmarks.) In the second step the uncertainty of each landmark is updated to reflect recent changes.

In **the third step** new landmarks are added to the state, the robot map of the world.

The matrices

The System state: X

The system state vector **X** contains the position of the robot, x, y and θ. Furthermore, it contains the x and y position of each landmark.

$$X = [x_r, y_r, \theta_r, x_1, y_1, \ldots, x_n, y_n]^T \tag{2}$$

The size of **X** is $(3 + 2 \cdot n) \times 1$ where *n* is the number of landmarks.

The covariance matrix: P

DEFINITION: The covariance of two variates provides a measure of how strongly correlated these two variables are, measuring the degree of linear dependence between variables.

The covariance matrix **P** contains 1) the covariance on the robot position, 2) the covariance on the landmarks, 3) the covariance between robot position and landmarks and 4) the covariance between landmarks.

As shown in the figure above, the first cell, **A** contains the covariance of the robot position. **B** is the covariance of the first landmark. This continues down to **C**. The cell **D** contains the covariance between the robot state and the first landmarks; The cell **E** contains the covariance between the first landmark and the robot state, which can be deduced from **D** by transposing **D**. **F** contains the covariance between the last landmark and the first landmark, which is the transpose of **G**.

Build-up: Initially as the robot has not seen any landmarks, the covariance matrix **P** only includes the matrix **A**. The covariance matrix **A** must be initialized using some **default values** for the diagonal. This reflects uncertainty in the initial position.

The Kalman gain: K

DEFINITION: The Kalman gain **K** is computed to find out how much we will trust the observed landmarks and as such how much we want to gain from the new knowledge they provide.

This is done using **the uncertainty of the observed landmarks** along with a measure of the quality of the range measurement device and **the odometry performance** of the robot.

$$\begin{bmatrix} x_r & x_b \\ y_r & y_b \\ \theta_r & \theta_b \\ x_{1,r} & x_{1,b} \\ y_{1,r} & y_{1,b} \\ \vdots & \vdots \\ x_{n,r} & x_{n,b} \\ y_{n,r} & x_{n.b} \end{bmatrix} \tag{3}$$

The size **K** is $(3 + 2 \cdot n) \times 2$, where *n* is the number of landmarks.

For the first row, the first column describes how much should be gained from the innovation in terms of range; the second column describes how much should be gained from the innovation in terms of the bearing.

The Jacobian of the measurement model: H

DEFINITION: The measurement model defines how to compute an expected range and bearing of the measurements (observed landmark positions).

$$\begin{bmatrix} range \\ bearing \end{bmatrix} = \begin{bmatrix} \sqrt{(\lambda_x - x)^2 + (\lambda_y - y)} + \nu_r \\ tan^{-1}(\frac{\lambda_y - y}{\lambda_x - x}) - \theta + \nu_\theta \end{bmatrix} \tag{4}$$

where λ_x is the x position of the landmark, x is the current estimated robot x position. θ is the robot rotation. This will give us the **predicted measurement** of the range and bearing to the landmark.

The Jacobian **H** of the measurement model (this matrix) is the derivative of if w.r.t. x, y and θ:

$$\partial \begin{bmatrix} range \\ bearing \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{x-\lambda_x}{r} & \frac{y-\lambda_y}{r} & 0 \\ \frac{\lambda_y-y}{r^2} & \frac{\lambda_x-x}{r^2} & -1 \end{bmatrix}}_{H} \cdot \partial \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5}$$

H shows us how much the range and bearing changes as x, y, and θ changes. But this is the contents of the usual **H** for regular EKF state estimateion. When doing SLAM we need some additional values for the landmarks:

$$\begin{bmatrix} (X_r & Y_r & \theta_r & X_1 & Y_1 & X_2 & Y_2 & X_3 & Y_3) \\ A & B & C & 0 & 0 & -A & -B & 0 & 0 \\ D & E & F & 0 & 0 & -D & -E & 0 & 0 \end{bmatrix} \tag{6}$$

For example, when using **H** for landmark No.2, the above matrix will be used.

The Jocabian of the prediction model: A

DEFINITION: The prediction model defines how to compute an expected position of the robot given the old position and the control input.

$$\begin{bmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{bmatrix} = \begin{bmatrix} x_{old} + \Delta t cos\theta + q\Delta t cos\theta \\ y_{old} + \Delta t sin\theta + q\Delta t sin\theta \\ \theta + \Delta\theta + q\Delta\theta \end{bmatrix} \tag{7}$$

Δt is the change in distance and q is the error term. The jocabian matrix **A** then can be written as

$$\begin{bmatrix} 1 & 0 & -\Delta t sin\theta \\ 0 & 1 & \Delta t cos\theta \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta y \\ 0 & 1 & \Delta x \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

The SLAM specific Jacobians: J_{xr} and J_z

There are some jacobians which are only used in SLAM.

The first is J_{xr}, which is ther Jacobian of the predication of the landmarks. It is basically the same as the jacobian of the prediction model, except that we start out without the rotation term.

$$\partial \begin{bmatrix} X_{landmark} \\ Y_{landmark} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & -\Delta t sin\theta \\ 0 & 1 & \Delta t cos\theta \end{bmatrix}}_{J_{xr}} \cdot \partial \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{9}$$

The jacobian J_z is also the jacobian of the prediction model for the landmarks, but this time with respect to [range, bearing]^T.

$$\partial \begin{bmatrix} X_{landmark} \\ Y_{landmark} \end{bmatrix} = \underbrace{\begin{bmatrix} cos(\theta + \Delta\theta) & -\Delta t sin(\theta + \Delta\theta) \\ sin(\theta + \Delta\theta) & -\Delta t cos(\theta + \Delta\theta) \end{bmatrix}}_{J_z} \cdot \partial \begin{bmatrix} \Delta t \\ \theta \end{bmatrix} \tag{10}$$

The process noise: Q and W

The process is assumed to have **a gaussian noise** proportional to the controls $\Delta x, \Delta y$ and $\Delta \theta$. The noise is denote as **Q**, which is a 3 by 3 matrix

$$W = [\Delta t cos\theta \quad \Delta t sin\theta \quad \Delta\theta]^T \tag{11}$$

$$Q = WCW^T = \begin{bmatrix} c\Delta x^2 & & \\ & c\Delta y^2 & \\ & & c\Delta\theta^2 \end{bmatrix} \tag{12}$$

where **C** is some gaussian sample representing how exact the odometry is.

The measurement noisy: R and V

The range measurement device is also assumed to have a **gaussian noise** proportional to the range and bearing. It is calculated as **VRV$^{\mathbf{T}}$** . **V** is just a 2 by 2 identity matrix. **R** is also a 2 by 2 matrix with numbers only in the diagonal.

$$R = \begin{bmatrix} r \cdot c & \\ & b * d \end{bmatrix} \tag{13}$$

In the upper left corner we have the range *r* and lower right bearing *b*, multiplied by some constants *c* and *d*. The constants should represent the **accuracy of the measurement device**. For example, the range error has 1 cm variance it should, *c* should be a gaussian with variance 0.01.

Processing step

Step 1: update current state using the odometry data

OBJECT: This step, called the **prediction step**, we update current state using the odometry data.

We use the controls given to the robot to calculate an estimate of the new position of the robot using **Eq** (7) and this should update the first three spaces in the state vector **X**.

We also need to update the **A** matrix **every iteration** using **Eq** (8).

Also **Q** should be updated to reflect the control terms $\Delta x, \Delta y$ and $\Delta\theta$.

$$Q = \begin{bmatrix} c\Delta x^2 & c\Delta x\Delta y & c\Delta x\Delta\theta \\ c\Delta y\Delta x & c\Delta y^2 & c\Delta y\Delta\theta \\ c\Delta\theta\Delta x & c\Delta\theta\Delta y & c\Delta\theta^2 \end{bmatrix} \tag{14}$$

Finally we can calculate the new covariance for the robot position and only the top left 3 by 3 matrix of P will be updated:

$$P^{rr} = AP^{rr}A + Q \tag{15}$$

And we also need to update the robot and feature cross correlation, which is done for the top 3 rows of the covariance matrix (except the first 3 columns):

$$P^{ri} = AP^{ri} \tag{16}$$

Step 2: Update state from re-observed landmarks

The estimate we obtained for the robot position is not completely exact due to the odometry errors from the robot.

Using the **associated landmarks** we can now calculate the **displacement** of the robot compared to what we think the robot position is. Using the **displacement** we can update the **robot position**. And this step is run for each re-observed landmark **seperately**.

We will try to **predict where the landmark is** using the current estimated robot position, (x, y) and the save it as (λ_x, λ_y). Then use **Eq** (4) we get the range and bearing to the landmarks, which is denoted as *h*.

h is compared to the range and bearing for the landmark we get from the data association, which is denoted as *z*.

But before this some more computations are required.

- Firstly we computed the Jacobian of the measurement model **H** using **Eq** (5) and (6).
- The error matrix **R** should also be updated to reflect the range and bearing in the current measurements.

Now we can compute the Kalman gain.

$$K = P \cdot H^T \cdot (HPH^T + VRV^T)^{-1} \tag{17}$$

Explain: The Kalman now contains a set of numbers indicating how much each of the landmark positions and the robot position should be updated according to the reobserved landmark. The term *(H * P * H T + V * R * V T)* is called the innovation covariance **S**, it is also used in the *Data Association* when calculating the validation gate for the landmark.

Finally we can compute a new state vector using the Kalman gain:

$$X = X + K \cdot (z - h) \tag{18}$$

This operation will update the robot position along with all the landmark positions, given the term of $(z - h)$ does not result in (0, 0). Note that $(z - h)$ yields a result of two numbers which is the displacement in range and bearing, denoted ν.

Step3: Add new landmarks t the current state

In this step we want to update the state vector **X** and the covariance matrix **P** with **new landmarks**.

First we add the new landmark to the state vector **X**.

$$X = \left[\begin{array}{c|cc} X & X_N & Y_N \end{array} \right] \tag{19}$$

Also we need to add a new row and column to the covariance matrix, shown in the figure below as the grey area.

First we add the covariance for the new landmark in the cell **C**, also called $\mathbf{P}^{\mathbf{N+1,N+1}}$.

$$P^{N+1,N+1} = J_{xr} P J_{xr}^T + J_z R J_z \tag{20}$$

After this we add the robot - landmark covariance for the new landmark, which is the lower-left corner of the covariance matrix.

$$P^{r,N+1} = P^{rr} \cdot J_{xr}^T \tag{21}$$

Finally, the (last) landmark - (i-th) landmark covariance need to be added (the lowest row).

$$P^{N+1,i} = J_{xr} \cdot (P^{ri})^T \tag{22}$$

Now, the robot is ready to move again, observing lanmarks, associating them and updating the system state etc.