

COURSERA
ROBOTICS SPECIALIZATION

ROBOTICS: ESTIMATION AND LEARNING

Notes are taken by Jelena Kocić

April 2019

Contents

1. Gaussian Model Learning.....	4
1.1. Introduction	4
1.2. Single Dimensional Gaussian	4
1.2.1. 1.2.1 Gaussian Distribution	4
1.2.2. 1.2.2. Maximum Likelihood Estimate of Gaussian Model Parameters.....	6
1.3. Multivariate Gaussian	10
1.3.1. Multivariate Gaussian Distribution	10
1.3.2. MLE of Multivariate Gaussian	13
1.4. Mixture of Gaussian	15
1.4.1. Gaussian Mixture Model (GMM)	15
1.4.2. GMM Parameter Estimation via EM	17
1.4.3. Expectation-Maximization (EM).....	22
PROGRAMMING ASSIGNMENT: Color Learning and Target Detection	27
2. Bayesian Estimation – Target Tacking.....	28
2.1. Kalman Filter Motivation	28
2.2. Kalman Filter Model – System and Measurement Models	29
2.3. MAP Estimate of KF – Maximum-A-Posterior Estimation.....	32
2.4. Non-Linear Variations – Extended Kalman and Unscented Kalman Filter.....	38
PROGRAMMING ASSIGNMENT: Kalman Filter Tracking	41
3. Mapping.....	42
3.1. Robotic Mapping.....	42
3.1.1. Metric Map	42
3.1.2. Topological Map.....	43
3.1.3. Semantic Map	44
3.2. Occupancy Grid Mapping.....	45
3.2.1. Occupancy Grid Map.....	45
3.2.2. Log-odd Update	50
3.2.3. Handling Range Sensor	54
3.3. 3D Mapping.....	60
PROGRAMMING ASSIGNMENT: 2D Occupancy Grid Mapping.....	64

4. Bayesian Estimation – Localization	65
4.1. Odometry Modeling.....	65
4.2. Map Registration.....	67
4.3. Particle Filters	71
4.4. Iterative Closest Point Algorithm – ICP.....	77
PROGRAMMING ASSIGNMENT: Particle Filter Based Localization.....	83

1. Gaussian Model Learning

1.1. Introduction

Sources of uncertainty in robotics:

- Sensor noise,
- Lack of knowledge about the world,
- Dynamic changes in motion and environment.

Methods for dealing with uncertainty:

- Probabilistic modeling,
- Machine learning.

1.2. Single Dimensional Gaussian

1.2.1. 1.2.1 Gaussian Distribution

Gaussian distribution is a widely used continuous probabilistic representation, and it provides a useful way to estimate uncertainty in the world.

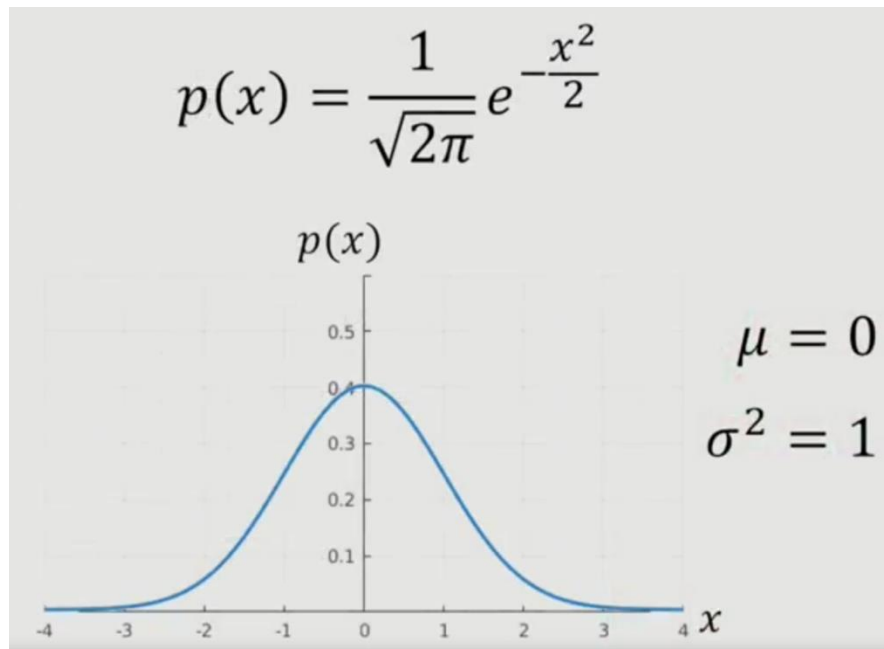
Why Gaussian distribution?

- only two parameters are needed to specify the Gaussian (mean and variance),
- mathematically the distribution has good properties (e.g. product of Gaussian distributions forms another Gaussian),
- the central limit theorem tells us that the expectation of the mean of any random variable converges to the Gaussian distribution.

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\}$$

x	Variable
μ	Mean
σ^2	Variance
σ	Standard deviation

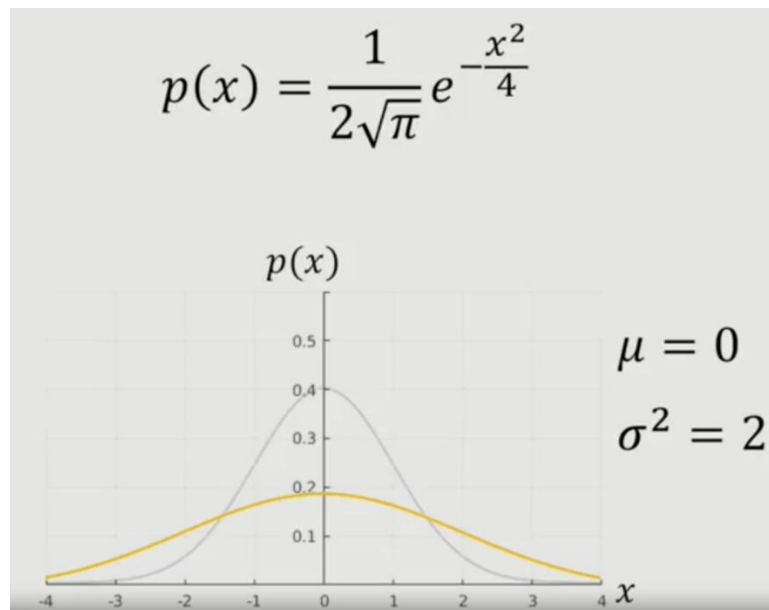
Standard Normal Distribution is when $\mu = 0$ and $\sigma^2 = 1$.



The mean value determines the center of the distribution. We can also say that the peak location of the distribution changes.

The variance changes the spread of the distribution. If the variance increases, the curve spreads out as compared to the standard Gaussian curve. Also, the peak value decreases so that the integral is still 1, which fulfills the properties of a probability density function. Conversely, a smaller

variance tightens the curve and the peak value becomes bigger as well. So that the integral remains 1.



1.2.2. 1.2.2. Maximum Likelihood Estimate of Gaussian Model Parameters

Goal is to estimate the mean and the variance given observed data.

Likelihood is the probability of an observation given the model parameters.

Likelihood: $p(\{x_i\}|\mu, \sigma)$

Observed data Unknown parameters

Quiz:

Complete the paragraph using a correct match of terms.

Likelihood is defined as the of the given values. It can be viewed as a function of given . By maximum likelihood estimation of , we mean a process to compute the that are most likely to give the .

- ☐ A. model parameter(s), B. probability, C. observed data
- ☐ A. observed data, B. probability, C. model parameter(s)
- ☒ A. probability, B. observed data, C. model parameter(s)

Correct

Objective is to estimate the mean and the variance given observed data:

$$\hat{\mu}, \hat{\sigma} = \arg \max_{\mu, \sigma} p(\{x_i\} | \mu, \sigma)$$

Assuming independence of observations,

$$p(\{x_i\} | \mu, \sigma) = \prod_{i=1}^N p(x_i | \mu, \sigma)$$

$$\hat{\mu}, \hat{\sigma} = \arg \max_{\mu, \sigma} \prod_{i=1}^N p(x_i | \mu, \sigma)$$

It will turn out that maximizing log likelihood is simpler in many cases.

=>

$$\begin{aligned}
 (1) \quad \arg \max_{\mu, \sigma} \prod_{i=1}^N p(x_i | \mu, \sigma) &= \arg \max_{\mu, \sigma} \ln \left\{ \prod_{i=1}^N p(x_i | \mu, \sigma) \right\} \\
 &= \arg \max_{\mu, \sigma} \sum_{i=1}^N \ln p(x_i | \mu, \sigma)
 \end{aligned}$$

NOTE 2:

$$\log(x_1 \times x_2 \times \cdots \times x_k) = \log(x_1) + \log(x_2) + \cdots + \log(x_k)$$

$$\hat{\mu}, \hat{\sigma} = \arg \max_{\mu, \sigma} \sum_{i=1}^N \ln p(x_i | \mu, \sigma)$$

(2) Gaussian! $\frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{(x_i - \mu)^2}{2\sigma^2} \right\}$

$$\hat{\mu}, \hat{\sigma} = \arg \max_{\mu, \sigma} \sum_{i=1}^N \left\{ -\frac{(x_i - \mu)^2}{2\sigma^2} - \ln \sigma \right\}$$

, where the constant is ignored.

Changing into minimization problem:

$$\hat{\mu}, \hat{\sigma} = \arg \min_{\mu, \sigma} \sum_{i=1}^N \left\{ \frac{(x_i - \mu)^2}{2\sigma^2} + \ln \sigma \right\}$$

$$\hat{\mu}, \hat{\sigma} = \arg \min_{\mu, \sigma} \underbrace{\sum_{i=1}^N \left\{ \frac{(x_i - \mu)^2}{2\sigma^2} + \ln \sigma \right\}}_{J(\mu, \sigma)}$$

• At optimum,

$$\frac{\partial J}{\partial \mu} = 0 \longrightarrow \hat{\mu}$$

$$\frac{\partial J(\hat{\mu}, \sigma)}{\partial \sigma} = 0 \longrightarrow \hat{\sigma}$$

If we apply the optimality condition for convex optimization, the first order derivative of J with respect to mu should be zero. From this, we can compute the maximum likelihood estimate of mu and we are going to write it as mu hat.

We apply the same optimality condition to compute the estimate of sigma. For this, we can use the value of mu hat in place of mu as a parameter.

The MLE solution:

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^N x_i$$

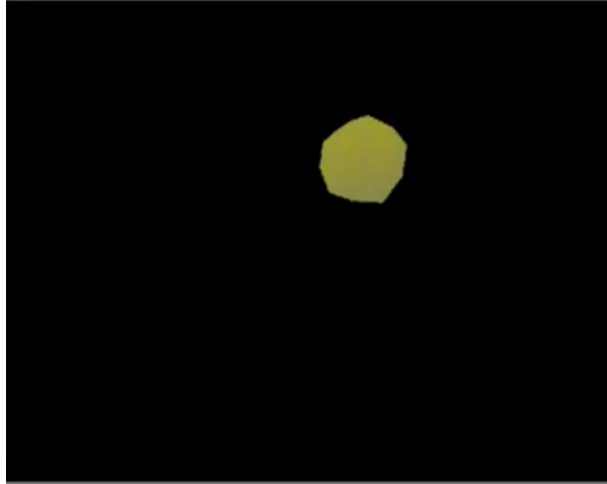
$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{\mu})^2$$

Mu hat is exactly the sample mean the average of the data, which is a natural estimate of data. Also, sigma hat square is just a sample variance.

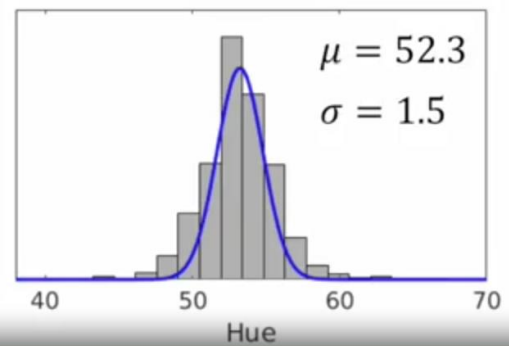
An example:

Ball color distribution

Segmented Ball Image

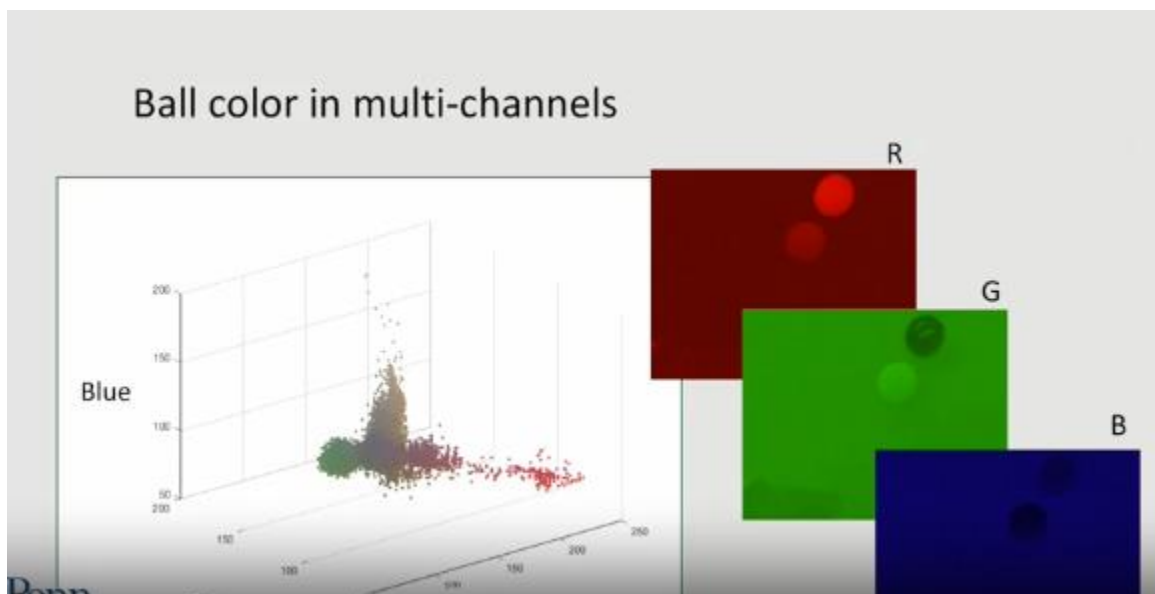


$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^N x_i$$
$$\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{\mu})^2$$



1.3. Multivariate Gaussian

1.3.1. Multivariate Gaussian Distribution



Mathematically, the multivariate Gaussian is expressed as an exponential coupled with a scalar vector.

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

D	Number of Dimensions	(Dimension = 1)
\mathbf{x}	Variable	$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\}$
$\boldsymbol{\mu}$	Mean vector	
Σ	Covariance matrix	

D is the number of dimensions we are going to use. \mathbf{x} is the vector of variables whose probability we are attempting to quantify. To signify that x is now a vector, we make x bold.

In contrast to the 1D case μ our mean is now a vector and σ , our covariance matrix is a square matrix.

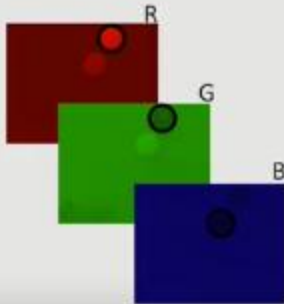
In our covariance matrix, there are two key components: diagonal terms: variance and off-diagonal terms: correlation.

(Dimension = 2)

$$\Sigma = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1} \sigma_{x_2} \\ \sigma_{x_2} \sigma_{x_1} & \sigma_{x_2}^2 \end{bmatrix} \quad (\sigma_{x_1} \sigma_{x_2} = \sigma_{x_2} \sigma_{x_1})$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

Ball color in multi-channels



$$D = 3$$

$$\mathbf{x} = [x_R \quad x_G \quad x_B]$$

$$D = 3$$

$$\mathbf{x} = [x_R \quad x_G \quad x_B]$$

$$\boldsymbol{\mu} = [\mu_R \quad \mu_G \quad \mu_B]$$

$$\Sigma = \begin{bmatrix} \sigma_{x_R}^2 & \sigma_{x_R} \sigma_{x_G} & \sigma_{x_R} \sigma_{x_B} \\ \sigma_{x_R} \sigma_{x_G} & \sigma_{x_G}^2 & \sigma_{x_G} \sigma_{x_B} \\ \sigma_{x_R} \sigma_{x_B} & \sigma_{x_G} \sigma_{x_B} & \sigma_{x_B}^2 \end{bmatrix}$$

Special case:

$$p(\mathbf{x}) = \frac{1}{2\pi} \exp \left\{ -\frac{x^2 + y^2}{2} \right\}$$

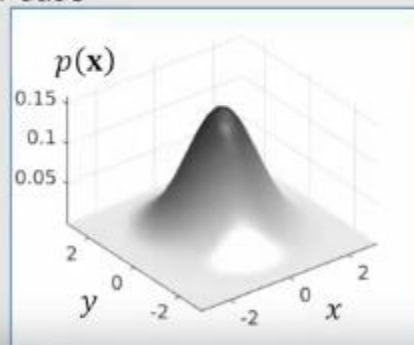
• 2D Zero-mean Spherical Case

$$D = 2$$

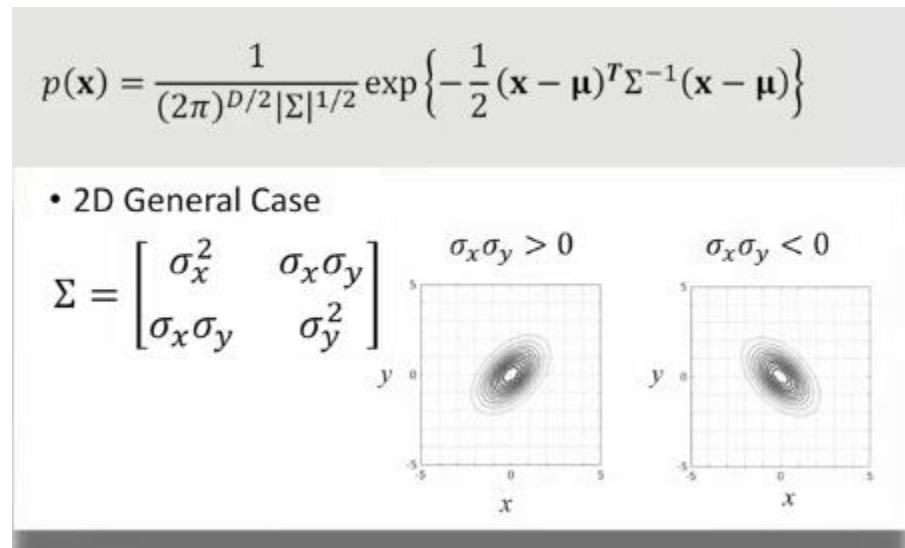
$$\mathbf{x} = [x \quad y]^T$$

$$\boldsymbol{\mu} = [0 \quad 0]^T$$

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



General case:



There are two other properties I'm going to mention about sigma.

First, the covariance metrics must remain symmetric and positive definite. Meaning, the elements of sigma are symmetric about its diagonal line. The Eigenvalues of sigma must be positive.

Second, even when the covariance matrix has none zero correlation terms we can always find the coordinate transformation which makes the shape appear symmetric.

1.3.2. MLE of Multivariate Gaussian

Objective: Estimate the mean and the covariance matrix given observed data.

Likelihood: $p(\{\mathbf{x}_i\} | \boldsymbol{\mu}, \Sigma)$

\swarrow \searrow
 Observed data Unknown parameters

• Objective

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} p(\{\mathbf{x}_i\} | \boldsymbol{\mu}, \Sigma)$$

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} \prod_{i=1}^N p(\mathbf{x}_i | \boldsymbol{\mu}, \Sigma)$$

(1) Take the log!

$$\arg \max \text{likelihood} \leftrightarrow \arg \max \ln(\text{likelihood})$$

$$\log(x_1 \times x_2 \times \dots \times x_k) = \log(x_1) + \log(x_2) + \dots + \log(x_k)$$

$$\arg \max_{\boldsymbol{\mu}, \Sigma} \prod_{i=1}^N p(\mathbf{x}_i | \boldsymbol{\mu}, \Sigma) \rightarrow \arg \max_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \ln p(\mathbf{x}_i | \boldsymbol{\mu}, \Sigma)$$

The statement

$$\log(x_1 * x_2 * \dots * x_k) = \log(x_1) + \log(x_2) + \dots + \log(x_k)$$

should be

$$\log(x_1 * x_2 * \dots * x_k) = \log(x_1) + \log(x_2) + \dots + \log(x_k)$$

(2) Gaussian!

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \left\{ -\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) - \frac{1}{2} \ln |\Sigma| + c \right\}$$



$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \min_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \left\{ \frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) + \frac{1}{2} \ln |\Sigma| \right\}$$

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \min_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \left\{ \frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) + \frac{1}{2} \ln |\Sigma| \right\}$$

$J(\boldsymbol{\mu}, \Sigma)$

• At optimum,

$$\textcircled{1} \quad \frac{\partial J}{\partial \boldsymbol{\mu}} = \mathbf{0} \longrightarrow \hat{\boldsymbol{\mu}} \quad \textcircled{2} \quad \frac{\partial J(\hat{\boldsymbol{\mu}}, \Sigma)}{\partial \Sigma} = 0 \longrightarrow \hat{\Sigma}$$

• In summary, we have

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$$

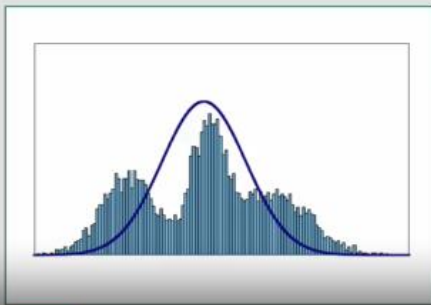
$$\hat{\Sigma} = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T$$

1.4. Mixture of Gaussian

1.4.1. Gaussian Mixture Model (GMM)

Limitations of Single Gaussian

- Single Mode
- Symmetric



A single Gaussian cannot properly model a distribution if it has multiple modes or there is a lack of symmetry.

GMM is the sum of Gaussians.

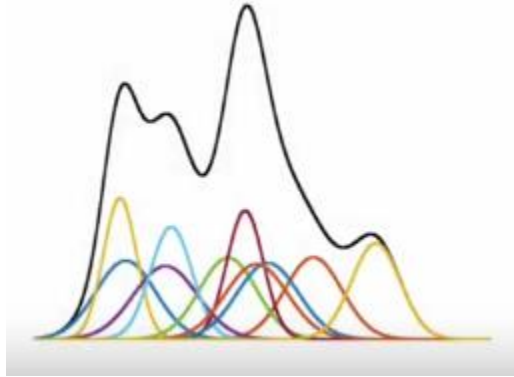


Figure. The colorful lines are ten random Gaussian curves. The black line is the sum of all the Gaussians.

If we choose the right Gaussian elements, then we can express any unusual distribution.

- Mixture of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^K w_k g_k(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k)$$

g_k : Gaussian with $\boldsymbol{\mu}_k$ and Σ_k

w_k : mixing coefficient (weight)

$$w_k > 0, \sum_{k=1}^K w_k = 1$$

The weights, W s should be all positive and they must sum to 1. This make sure that the distribution of GMM is a probability density that integrals to 1.

Using GMM



Flexibility



Parameters \uparrow

$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$$

$$\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \dots, \boldsymbol{\Sigma}_K\}$$

$$\boldsymbol{w} = \{w_1, w_2, \dots, w_K\}$$

K : Number of Components

Bigger number of parameters. \rightarrow No analytic solution. Overfitting.

1.4.2. GMM Parameter Estimation via EM

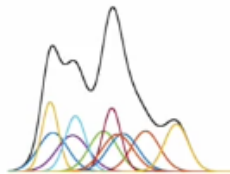
• Single Gaussian



$\boldsymbol{\mu}$

$\boldsymbol{\Sigma}$

• Mixture of Gaussians



$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$$

$$\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \dots, \boldsymbol{\Sigma}_K\}$$

$$\boldsymbol{w} = \{w_1, w_2, \dots, w_K\}$$

K : Number of Components

$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K\}$$

$$\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \dots, \boldsymbol{\Sigma}_K\}$$

$$w = 1/K$$

K : Given

Learning GMM Parameters

Likelihood: $p(\{\mathbf{x}_i\}|\boldsymbol{\mu}, \Sigma)$

\swarrow \searrow
Observed data Unknown parameters

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} p(\{\mathbf{x}_i\}|\boldsymbol{\mu}, \Sigma)$$

$$\boldsymbol{\mu} = \{\boldsymbol{\mu}_k\}$$

$$\Sigma = \{\Sigma_k\} \quad k = 1, 2, \dots, K$$

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} \prod_{i=1}^N p(\mathbf{x}_i|\boldsymbol{\mu}, \Sigma)$$

(1) Take the log!

$$\arg \max \text{likelihood} \leftrightarrow \arg \max \ln(\text{likelihood})$$

$$\log(x_1 \times x_2 \times \dots \times x_k) = \log(x_1) + \log(x_2) + \dots + \log(x_k)$$

$$\arg \max_{\boldsymbol{\mu}, \Sigma} \prod_{i=1}^N p(\mathbf{x}_i|\boldsymbol{\mu}, \Sigma) \rightarrow \arg \max_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \ln p(\mathbf{x}_i|\boldsymbol{\mu}, \Sigma)$$

Learning GMM Parameters

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \ln p(\mathbf{x}_i | \boldsymbol{\mu}, \Sigma)$$

(2) Gaussian Mixture Model!

$$p(\mathbf{x}) = \sum_{k=1}^K w_k g_k(\mathbf{x} | \boldsymbol{\mu}_k, \Sigma_k) \quad g_k: \text{Gaussian with } \boldsymbol{\mu}_k \text{ and } \Sigma_k$$



$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \ln \left\{ \frac{1}{K} \sum_{k=1}^K g_k(\mathbf{x}_i | \boldsymbol{\mu}_k, \Sigma_k) \right\}$$

$$\hat{\boldsymbol{\mu}}, \hat{\Sigma} = \arg \max_{\boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \ln \left\{ \frac{1}{K} \sum_{k=1}^K g_k(\mathbf{x}_i | \boldsymbol{\mu}_k, \Sigma_k) \right\}$$

where

$$g_k(\mathbf{x}) = \frac{1}{(2\pi)^{D/2} |\Sigma_k|^{1/2}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right\}$$

→ No closed form solution exist.

But when we apply the specific probability model of the GMM into the equation, which is a sum of Gaussians, we have this. It turns out that we cannot further simplify this formula analytically, because there appears a summation of Gaussians inside the log function. This implies we can estimate the parameters only via iterative computations.

EM for GMM:

1. Special case: EM for GMM Parameter Estimation
2. General EM Algorithm

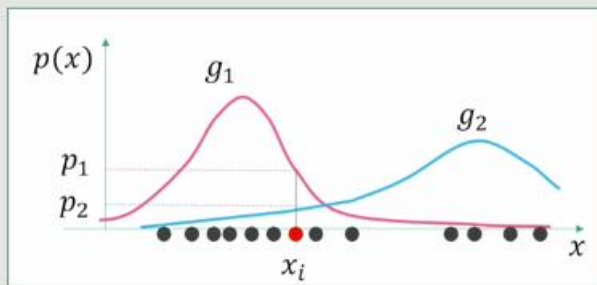
Initial μ and Σ

Latent variable z

- Latent Variable

$$z_k^i = \frac{g_k(\mathbf{x}_i | \mu_k, \Sigma_k)}{\sum_{k=1}^K g_k(\mathbf{x}_i | \mu_k, \Sigma_k)}$$

$$z_k^i = \frac{g_k(\mathbf{x}_i | \mu_k, \Sigma_k)}{g_1(\mathbf{x}_i | \mu_1, \Sigma_1) + g_2(\mathbf{x}_i | \mu_2, \Sigma_2)}$$



$$z_1^i = \frac{p_1}{p_1 + p_2}$$

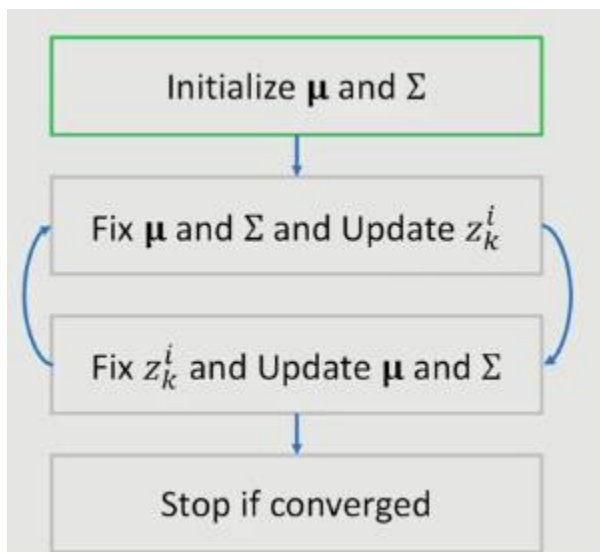
$$z_2^i = \frac{p_2}{p_1 + p_2}$$

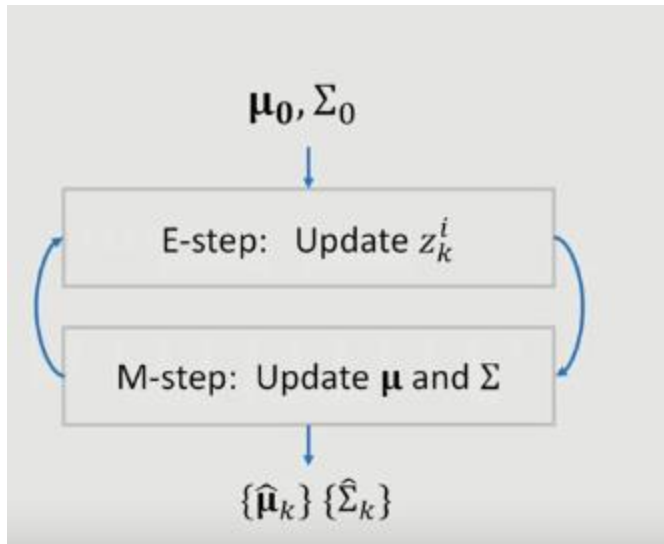
- Mean and Covariance Matrix

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{z_k} \sum_{i=1}^N z_k^i \mathbf{x}_i$$

$$\hat{\Sigma}_k = \frac{1}{z_k} \sum_{i=1}^N z_k^i (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

$$z_k = \sum_{i=1}^N z_k^i$$





1.4.3. Expectation-Maximization (EM)

Let consider the EM Algorithm as a maximization of a lower bound of an object function.

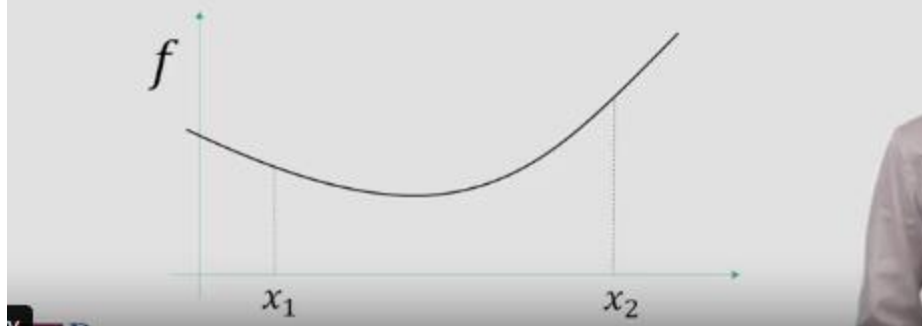
- EM as lower-bound maximization

$$\arg \max_{\theta} \sum_i \ln p(x_i | \theta) \quad \theta : \text{All parameters}$$

- (1) Jensen's inequality
- (2) Latent variable and marginal probability
- (3) Procedure : E-step and M-step.

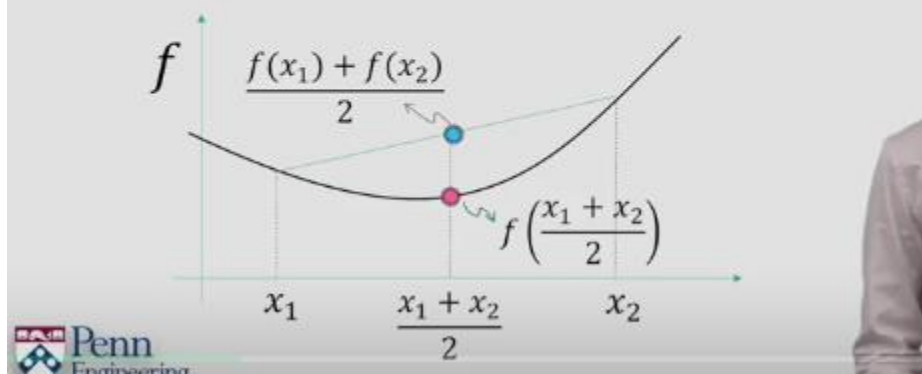
(1) Jensen's inequality

f : **convex** function



(1) Jensen's inequality

f : **convex** function



(1) Jensen's inequality

f : **convex** function

$$f(a_1x_1 + a_2x_2) \leq a_1f(x_1) + a_2f(x_2)$$

$$a_1 + a_2 = 1$$

$$a_1 \geq 0$$

$$a_2 \geq 0$$

n

(1) Jensen's inequality

f : **convex** function

$$f\left(\sum a_i x_i\right) \leq \sum a_i f(x_i)$$

$$\sum a_i = 1$$

$$a_i \geq 0$$

m
eering

Penn
Engineering

\ln is concave

$$\sum a_i = 1$$

$$a_i \geq 0$$

Penn
Engineering

(From definition of marginal probability)

Penn
Engineering

Log-likelihood

Lower bound

Penn
Engineering

(2) latent variable

$$\ln p(X|\theta) = \ln \sum_Z p(X, Z|\theta) \geq \sum_Z q(Z) \ln \frac{p(X, Z|\theta)}{q(Z)}$$

Log-likelihood

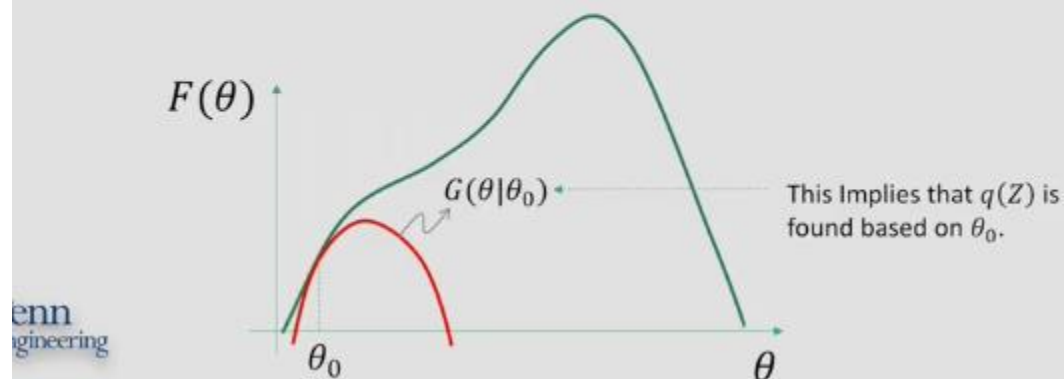
Lower bound



Find q !

(3a) Find a lower bound G with an initial guess

$$F \underbrace{\frac{\ln p(X|\theta)}{\sum_Z q(Z) \ln \frac{p(X, Z|\theta)}{q(Z)}}}_{G}$$



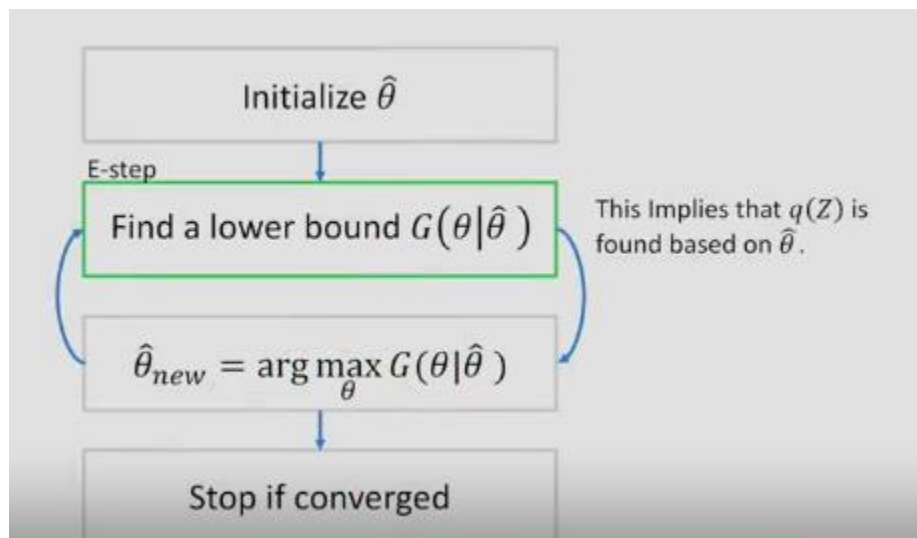
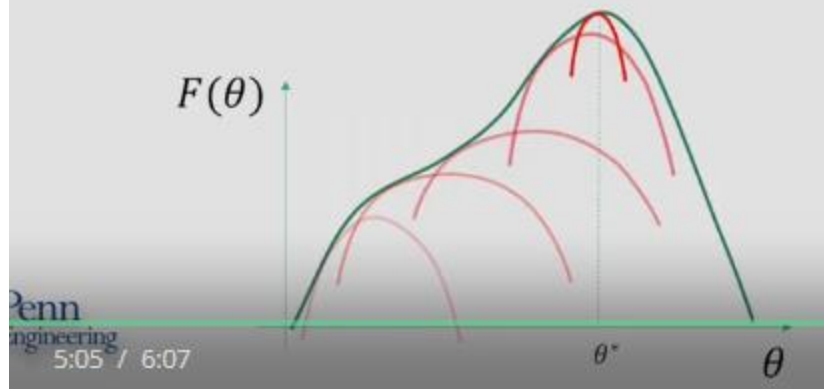
(3b) Find $\theta^* = \arg \max_{\theta} G(\theta|\theta_0)$

(3c) Find a new lower bound G with $\theta_1 \leftarrow \theta^*$

(3d) Find $\theta^* = \arg \max_{\theta} G(\theta|\theta_1)$

(4e) Repeat (until converged)

$$F \underbrace{\ln p(X|\theta)} \geq \underbrace{\sum_Z q(Z) \ln \frac{p(X, Z|\theta)}{q(Z)}}_G$$

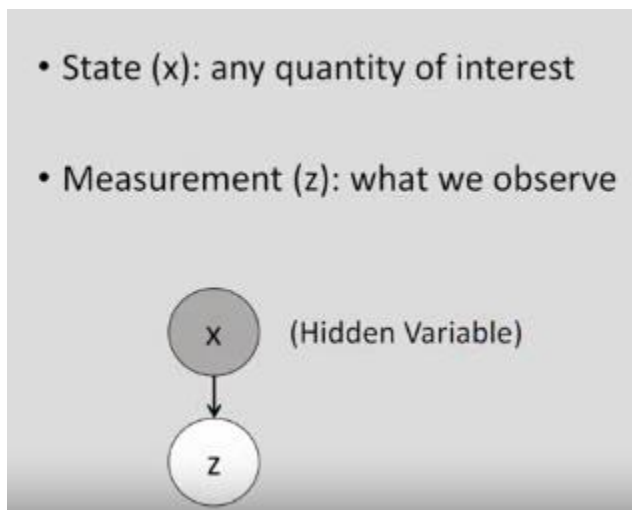
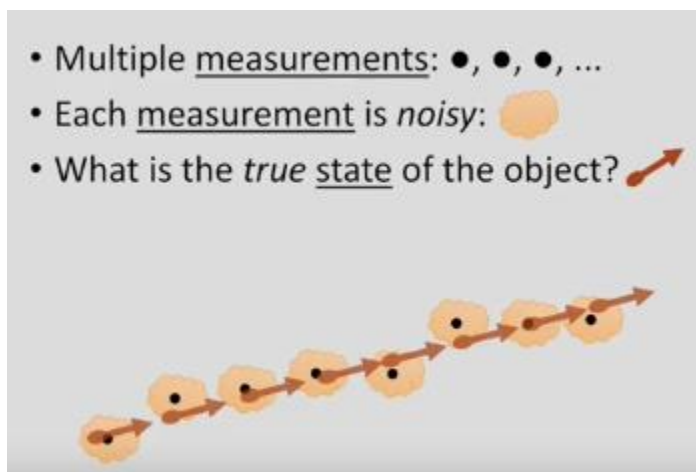


PROGRAMMING ASSIGNMENT: Color Learning and Target Detection

2. Bayesian Estimation – Target Tacking

2.1. Kalman Filter Motivation

The Kalman Filter is an optimal tracking algorithm for linear systems that is widely used in many applications. Examples of tracking includes pedestrian and vehicle tracking for self-driving cars or items traveling along a conveyor belt on an assembly line.



The concepts of measurements should be disambiguated from the concepts of state.

There's a true state of the world, but the robots can only observe a shadow of that world. For instance, the true position of the soccer ball may be 11 meters away from the robot, but the robot thinks that it is 11.32567 meters away. The robot observes this position of the ball through its camera, and this measurement through the camera gives a noisy estimate of the state. One source of the noise is the collection of pixels that can be misclassified between the ball and the surrounding area. We saw this kind of noise in module one.

- Example: “What characterizes the **state** of a ball?”

- Position, Velocity, Acceleration
- Rotation
- Color
- Size
- Weight
- Temperature
- Elasticity
- ...

- Example: What do we observe or **measure**?

- Distance
- Angle
- Inertia change
- Color
- ...

2.2. Kalman Filter Model – System and Measurement Models

Bayesian Kalman Filtering

- Modeling motion and noise
- Mathematical underpinnings of Kalman filters
- Position tracking example

Linear Modeling

- Discrete Linear dynamical system of motion

$$\bullet \quad x_{t+1} = A x_t + B u_t \quad z_t = C x_t$$

- Simple state vector, x , is position and velocity

$$\bullet \quad x_{t+1} := \begin{bmatrix} v & dv/dt \end{bmatrix}$$

- Description of Dynamics

$$\bullet \quad A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$$

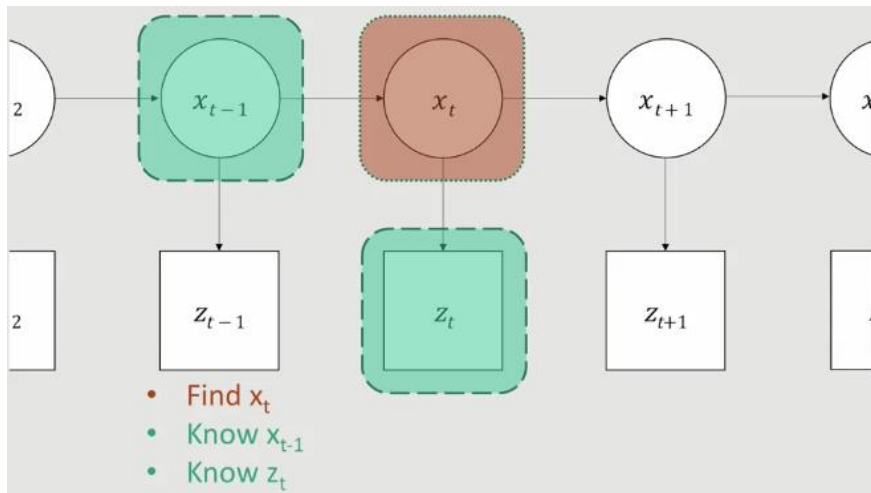
In a simple example, the state, x , will be indexed by time steps, t . The state will be comprised of position v , in meters, and velocity dv/dt measured in meters per second. Due to dynamics, the state changes in each time step. Going from the current time step t to the next time step $t+1$. This change is captured by A , the state transition matrix. Sometimes notated as π . The state transition matrix combines state information to describe the state at the next step in time.

The transition simplifies the current state to depend only on the previous state making our mathematical lives easier.

With the state being in position in velocity, we know that the position must change in time based on the velocity. The state transition matrix captures this with the given formulation.

The robot will not directly measure x unfortunately, but the robot may observe portions of x through it's sensors. This portion is labeled z , where the relationship between the state and measurement is given by the mixing matrix, c .

For completeness, the term u is included. Which represents any external input not dependent on the state, x . We will not explore this extra term in this module, instead, it is set to zero.



Bayesian modeling

- Prediction using state dynamics model
 $p(x_{t+1}|x_t)$
- Inference from noisy measurements
 $p(z_t|x_t)$
- Model x_t with a Gaussian (mean and covariance)
 $p(x_t) = \mathcal{N}(x_t, P_t)$

- Apply linear dynamics

$$p(x_{t+1}|x_t) = Ap(x_t)$$

$$p(z_t|x_t) = Cp(x_t)$$
- Add noise for motion and observations

$$p(x_{t+1}|x_t) = Ap(x_t) + v_m$$

$$p(z_t|x_t) = Cp(x_t) + v_o$$
- Introduce Gaussian model of x_t

$$p(x_{t+1}|x_t) = A\mathcal{N}(x_t, P_t) + \mathcal{N}(0, \Sigma_m)$$

$$p(z_t|x_t) = C\mathcal{N}(x_t, P_t) + \mathcal{N}(0, \Sigma_o)$$

- Consolidate expression using special properties

$$p(x_{t+1}|x_t) = \mathcal{N}(Ax_t, P_t) + \mathcal{N}(0, \Sigma_m)$$

$$p(z_t|x_t) = \mathcal{N}(Cx_t, P_t) + \mathcal{N}(0, \Sigma_o)$$

- Apply *linear* transform to Gaussian distributions

$$p(x_{t+1}|x_t) = \mathcal{N}(Ax_t, AP_tA^T) + \mathcal{N}(0, \Sigma_m)$$

$$p(z_t|x_t) = \mathcal{N}(Cx_t, CP_tC^T) + \mathcal{N}(0, \Sigma_o)$$

- Apply summation

$$p(x_{t+1}|x_t) = \mathcal{N}(Ax_t, AP_tA^T + \Sigma_m)$$

$$p(z_t|x_t) = \mathcal{N}(Cx_t, CP_tC^T + \Sigma_o)$$

2.3. MAP Estimate of KF – Maximum-A-Posterior Estimation

- Bayes' Rule

$$p(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Given from Kalman model:

$$p(x_t|x_{t-1}) = \mathcal{N}(Ax_{t-1}, AP_{t-1}A^T + \Sigma_m)$$

$$p(z_t|x_t) = \mathcal{N}(Cx_t, CP_tC^T + \Sigma_o)$$

- Bayes' Rule

$$p(\alpha|\beta) = \frac{P(\beta|\alpha)P(\alpha)}{P(\beta)}$$

- Given from Kalman model:

$$p(x_t|x_{t-1}) = \mathcal{N}(Ax_{t-1}, AP_{t-1}A^T + \Sigma_m)$$

$$p(z_t|x_t) = \mathcal{N}(Cx_t, \Sigma_o)$$

From the dynamical system, the probability of the state given only the previous state can be represented with the prior information alpha.

Representing the information from our measurement model, beta provides observational evidence. Conditioned on a state, this evidence presents a constrained probability distribution known as the likelihood. Altogether, Bayes' Rule helps us to formulate an expression for the posterior probability.

Bayesian filtering

- Apply Bayes' Rule $p(\alpha|\beta) = \frac{P(\beta|\alpha)P(\alpha)}{P(\beta)}$

$$p(x_t|x_{t-1}) = \mathcal{N}(Ax_{t-1}, AP_{t-1}A^T + \Sigma_m) \rightarrow \boxed{\alpha}$$

Prior

$$p(z_t|x_t) = \mathcal{N}(Cx_t, \Sigma_o) \rightarrow \boxed{\beta|\alpha}$$

Likelihood

$$\boxed{p(x_t|z_t, x_{t-1})} = \frac{p(z_t|x_t, x_{t-1})p(x_t|x_{t-1})}{P(z_t)}$$

Posterior

Note :

The equation:

$$p(z_t|x_t) = N(Cx_t, \Sigma_0)$$

should actually be :

$$p(z_t|x_t) = N(Cx_t, CP_tC^T + \Sigma_0)$$

as you will observe in the following slides.

The posterior probability represents our best estimate of the state x of t , given information from both the previous state, x of $t-1$, and observation z_t . This estimate will provide a basis for the new mean over Gaussian distribution representing our state x of t .

The Maximum A-Posterior estimation technique can provide optimal best estimate of the distribution.

The map estimate is formed as an optimization problem over all values in the posterior distribution.

We drop the probabilities that are independent of the state such as the distribution of all measurements z of t unconditioned on the state x of t . Fully expanded, we see a maximization over the product of Gaussians.

A trick to calculate the MAP estimate is to take the logarithm of the product. The logarithm represents a monotonic function. So the optimal value of x sub t in a logarithmic function remains the optimal value of x sub t in the original function.

- Posterior distribution is another Gaussian
- MAP Estimates “optimal” x_t value
- Use MAP estimates to form a new mean and variance for the state

- Calculate the Maximum A Posteriori Estimate

$$\hat{x}_t = \operatorname{argmax}_{x_t} p(x_t | z_t, x_{t-1})$$

$$\hat{x}_t = \operatorname{argmax}_{x_t} \frac{p(z_t | x_t) p(x_t | x_{t-1})}{P(z_t)}$$

$$\hat{x}_t = \operatorname{argmax}_{x_t} p(z_t | x_t) p(x_t | x_{t-1})$$

$$\hat{x}_t = \operatorname{argmax}_{x_t} \mathcal{N}(Cx_t, CP_t C^T + \Sigma_o) \mathcal{N}(Ax_{t-1}, AP_{t-1} A^T + \Sigma_m)$$

- Calculate the Maximum A Posteriori Estimate

$$\hat{x}_t = \operatorname{argmax}_{x_t} \mathcal{N}(Cx_t, CP_t C^T + \Sigma_o) \mathcal{N}(Ax_t, AP_{t-1} A^T + \Sigma_m)$$

- Simplify with these substitutions

$$P = P_t = AP_{t-1} A^T + \Sigma_m$$

$$R = CP_t C^T + \Sigma_o$$

- Simplify the exponential form of \mathcal{N} via logarithms

$$\hat{x}_t = \operatorname{argmin}_{x_t} \frac{(z_t - Cx_t)R^{-1}(z_t - Cx_t)}{+ (x_t - Ax_{t-1})P^{-1}(x_t - Ax_{t-1})}$$

- Solve optimization by setting the derivative to zero

$$\hat{x}_t = \underset{x_t}{\operatorname{argmin}} \left((z_t - Cx_t)R^{-1}(z_t - Cx_t) + (x_t - Ax_{t-1})P^{-1}(x_t - Ax_{t-1}) \right)$$

$$0 = \frac{d}{dx_t} \left((z_t - Cx_t)R^{-1}(z_t - Cx_t) + (x_t - Ax_{t-1})P^{-1}(x_t - Ax_{t-1}) \right)$$

All values inside the argmin function of the form,

$AR^{-1}A$ are actually of the form $AR^{-1}A^T$

where $A = (z_t - Cx_t), (x_t - Ax_{t-1})$

- Collect terms in the derivative

$$(C^T R^{-1} C + P^{-1})x_t = z_t^T R^{-1} C + P^{-1} A x_{t-1}$$

$$x_t = (C^T R^{-1} C + P^{-1})^{-1} (z_t^T R^{-1} C + P^{-1} A x_{t-1})$$

- Apply the *Matrix Inversion Lemma*

$$(C^T R^{-1} C + P^{-1})^{-1} = P - PC^T(R + CPC^T)^{-1}CP$$

- Define *Kalman Gain*: $K = PC^T(R + CPC^T)^{-1}$

- Expand the terms

$$x_t = (C^T R^{-1} C + P^{-1})^{-1} (C^T R^{-1} y_t + P^{-1} A x_{t-1})$$

$$x_t = (P - K C P) (C^T R^{-1} z_t + P^{-1} A x_{t-1})$$

$$x_t = A x_{t-1} + P C^T R^{-1} z_t - K C A x_{t-1} - K C P C^T R^{-1} y_t$$

$$x_t = A x_{t-1} - K C A x_{t-1} + (P C^T R^{-1} - K C P C^T R^{-1}) y_t$$

$$x_t = A x_{t-1} - K C A x_{t-1} + K y_t$$

$$\hat{x}_t = A x_{t-1} + K (z_t - C A x_{t-1})$$

- Convince yourself that $K = P C^T R^{-1} - K C P C^T R^{-1}$

The first equation mentioned on the slide should be:

$$x_t = (C^T R^{-1} C + P^{-1})^{-1} (C^T R^{-1} z_t + P^{-1} A x_{t-1})$$

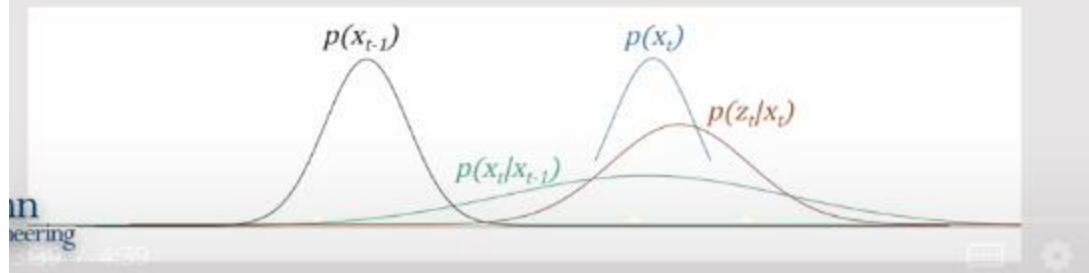
The z_t should replace the y_t in this and all the subsequent equations.

- Must update the covariance of the state

$$\hat{P}_t = P - K C P$$

1D Visualization

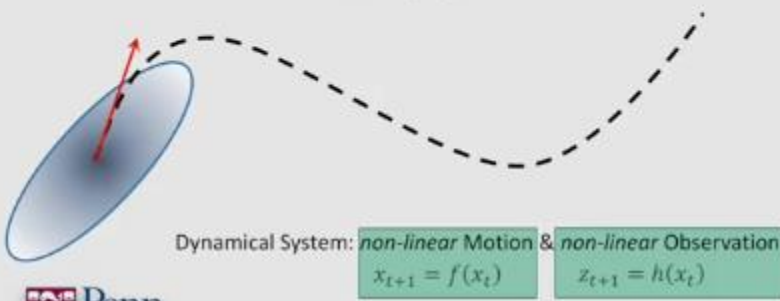
- The position of x is moving forward
 - Uncertain motion model increases the spread
- We observe a noisy position estimate, z_t
- The corrected position has less spread than both the observation and motion adjusted state



2.4. Non-Linear Variations – Extended Kalman and Unscented Kalman Filter

Extended Kalman Filter

- Linearize around the transition function
 - Jacobian represents the derivative in matrix form
- Calculate the uncertainty update via linearization



- Covariance prediction $p(x_{t+1}|x_t) = \mathcal{N}(Ax_t, AP_tA^T + \Sigma_m)$

$$p(x_{t+1}|x_t) = \mathcal{N}\left(f(x_t), \frac{\partial f}{\partial x} P_t \frac{\partial f^T}{\partial x} + \Sigma_m\right)$$

- Kalman Gain

$$K = PC^T(\Sigma_o + CPC^T)^{-1}$$

$$K = P \frac{\partial h^T}{\partial x} \left(\Sigma_o + \frac{\partial h}{\partial x} P_t \frac{\partial h^T}{\partial x} \right)^{-1}$$

- Overall update $\hat{x}_t = f(x_{t-1}) + K(z_t - h(f(x_t)))$

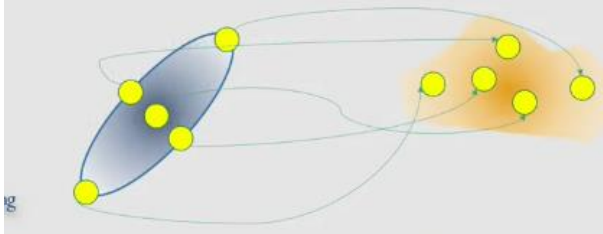
$$\hat{P}_t = P - K \frac{\partial h}{\partial x} P$$

Unscented Kalman Filter

- Noise distribution characterized by a set of points
- Transform these points in non-linear fashion
- Re-estimate mean and covariance for recalculating Gaussian distribution to characterize uncertainty

Unscented Kalman Filter

- Model the distribution based on a set of *sigma points*
 - Statistics of these points captures the distribution



- Approximate new distribution by computing the statistics of the *sigma points* \mathcal{X}_i
- Approximate a Gaussian with first two moments
 - Recalculate sigma points

- State prediction
 - Mean of Sigma points run through dynamic system

$$x_{t+1|x_t} = \frac{1}{N_X} \sum_i f(\mathcal{X}_i)$$

- Uncertainty prediction
 - Covariance of sigma points run through dynamic system

$$P_{t+1|x_t} = \frac{1}{N_X} \sum_i (f(\mathcal{X}_i) - x_{t+1|x_t})(f(\mathcal{X}_i) - x_{t+1|x_t})^T$$

- Expected Observation

- Mean of Sigma points' expected observation

$$z_{t+1|x_t} = \frac{1}{N_X} \sum_i h(f(\mathcal{X}_i))$$

Recall Linear System
 $K = PC^T(\Sigma_o + CPC^T)^{-1}$

- Kalman Gain

- Utilize sigma points, not observation model

$$K = \frac{1}{N_X} \sum_i (f(\mathcal{X}_i) - x_{t+1|x_t})(h(f(\mathcal{X}_i)) - z_{t+1|x_t})^T \cdot \left(\frac{1}{N_X} \sum_i (h(f(\mathcal{X}_i)) - z_{t+1|x_t})(h(f(\mathcal{X}_i)) - z_{t+1|x_t})^T \right)^{-1}$$

- Update just as the linear filter

- The covariance update will be slightly different
- See notes for good resources on further details

PROGRAMMING ASSIGNMENT:

Kalman Filter Tracking

3. Mapping

3.1. Robotic Mapping

Map is a spatial model of a robot's environment.

Mapping is a process of building a map.

Consideration for mapping:

- Map representation,
- Available sensors,
- Purpose of mapping.

Mapping results can be different according to the reason why the robot wants to have a map:

- to build a fine globally consistent map, or
- to avoid local collisions while navigating.

The level of precision and accuracy of the map should be decided based on the motivation for the mapping.

Types of map:

- Metric map,
- Topological map,
- Semantic map.

3.1.1. Metric Map

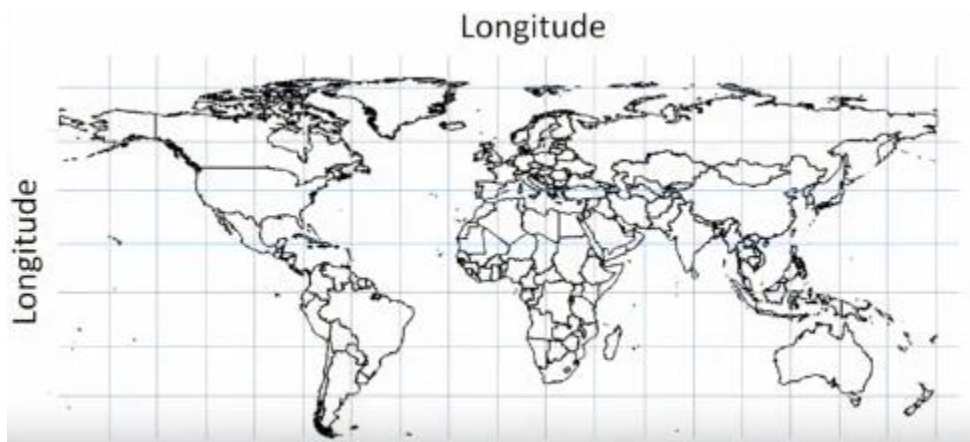


Figure 3.1. Metric map

Generally, in a metric map, a location is represented as a coordinate. This serves as the most basic form of maps since most mobile robots use some coordinate frame for self-localization.

3.1.2. Topological Map

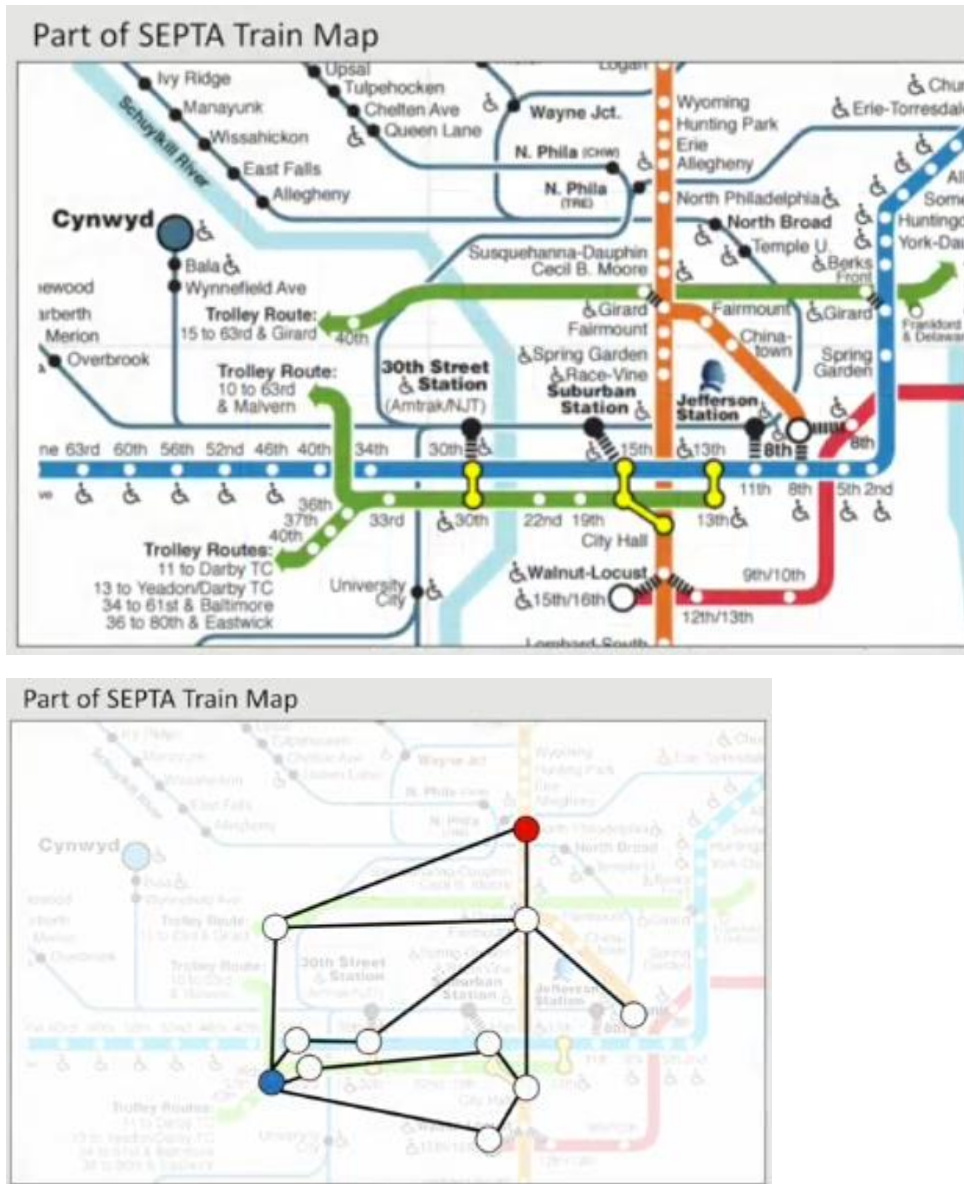


Figure 3.2. Topological map

In topological map the locations are represented as nodes, and their connections as arcs. The exact coordinate is not important in this representation, but the connections among nodes matter.

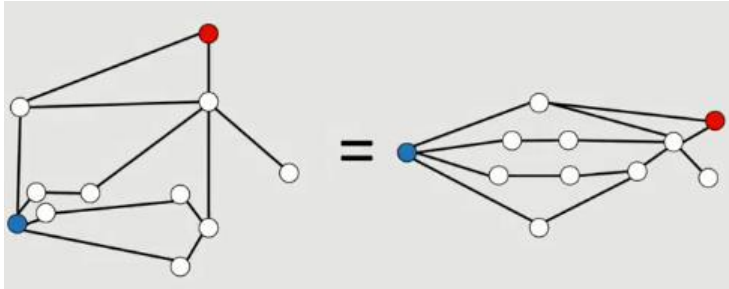


Figure 3.3. In topological map only the connectivity between nodes matter.

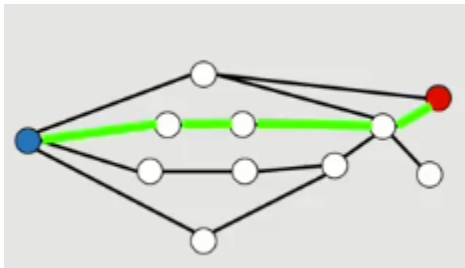


Figure 3.4. Graph representation of the map is useful for path planning.

3.1.3. Semantic Map



Figure 3.5. Semantic map is map with labels.

In figure 3.5, the example of the semantic map is given, with distinct labels and relative locations of labeled objects.

The semantic map is useful for high level planning or human robot interaction. Building a semantic map requires advanced object recognition techniques.

Challenges in mapping:

- Noisy measurement in local coordinate,
- Motion involved,
- Change over time.

3.2. Occupancy Grid Mapping

3.2.1. Occupancy Grid Map

Problem definition: A real mobile robot is running on the ground. The data is collected from the same robot. The robot has many on-board sensors. As the robot collects this information over time, while moving around, we can build a map of the objects detected by sensors. The goal is to build an occupancy grid maps from laser readings.

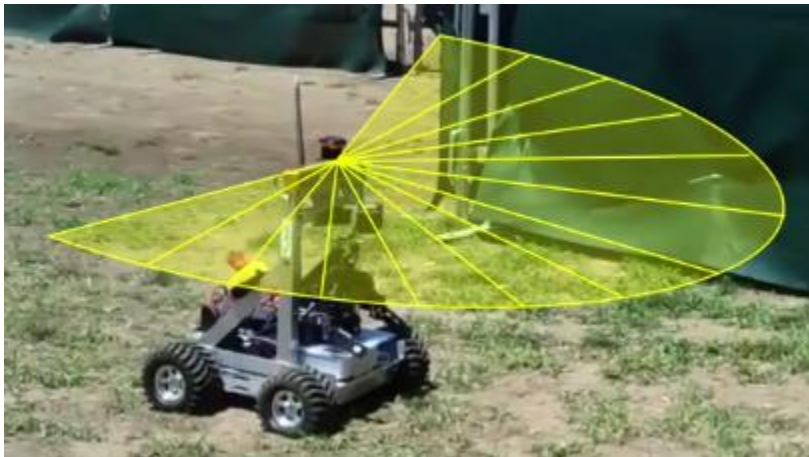


Figure 3.6. The real mobile robot with on-board sensors.



Figure 3.7. The results of Indore mapping.

Goal: build an occupancy grid maps from laser readings.

The term **Occupancy** is defined as a binary random variable.

$$m_{x,y}: \{free, occupied\} \rightarrow \{0, 1\}$$

This case **Occupancy** is defined in the probability space that has two possible states, free and occupied.

Given some probability space (Ω, P) ,
a **random variable** $X: \Omega \rightarrow R$ is a *function* that
maps the sample space to the reals.

An **Occupancy grid map** is just an array of occupancy variables.

- **Occupancy grid map**

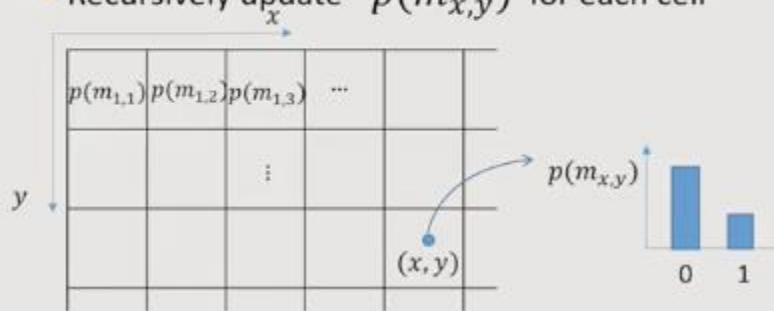
: fine-grained grid map where an occupancy variable associated with each cell



- **Occupancy grid mapping**

: A Bayesian filtering to maintain an occupancy grid map.

Recursively update $p(m_{x,y})$ for each cell



Occupancy grid mapping requires, a Bayesian filtering algorithm to maintain an Occupancy grid map. Bayesian filtering implies a recursive update to the map. A robot can never be certain about the world so we use the probabilistic notion of occupancy instead of the occupancy itself.

Sensor Measurements:



Figure 3.7. Range sensor.

Occupancy grid mapping algorithms usually incorporate a range sensor. This sensor provides distance information. However in our map cell's point of view there are two possible measurements. A cell could be passed through by the ray, which means it is free empty space. Also it is possible that a cell is hit by the ray, which means a cell is occupied by something.

We will use **0**, for the **Free** measurements. **1**, where the **Occupied** measurement for each cell.

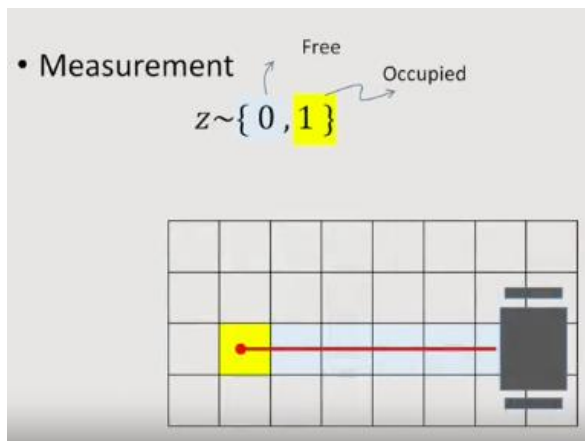


Figure 3.8. Free and Occupied cells

• Measurement model

$$p(z|m_{x,y})$$

$$p(z = 1|m_{x,y} = 1) \quad : \text{ True } \mathbf{occupied} \text{ measurement}$$

$$p(z = 0|m_{x,y} = 1) \quad : \text{ False } \mathbf{free} \text{ measurement}$$

$$p(z = 1|m_{x,y} = 0) \quad : \text{ False } \mathbf{occupied} \text{ measurement}$$

$$p(z = 0|m_{x,y} = 0) \quad : \text{ True } \mathbf{free} \text{ measurement}$$

Figure 3.9. Probabilistic model of the measurements.

There are only four possible conditional probabilities of measurements (because the variables z and m are all binary).

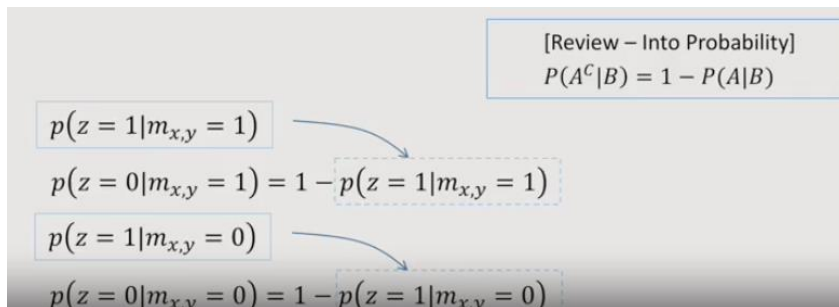
Probability that z is 1 given m is 1 is the probability that we have **occupied measurements** for an **occupied cell**.

Probability that z is 0 given m is 1 is the probability that we have **free measurement** for an **occupied cell**.

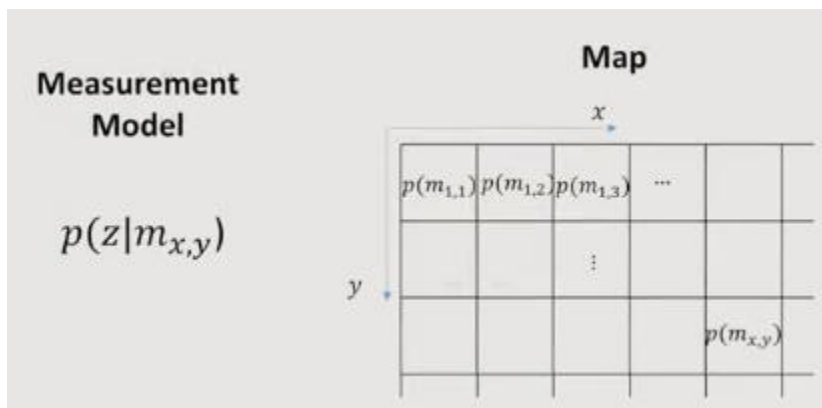
We can define a probabilities of observation given m is 0, in the same way.

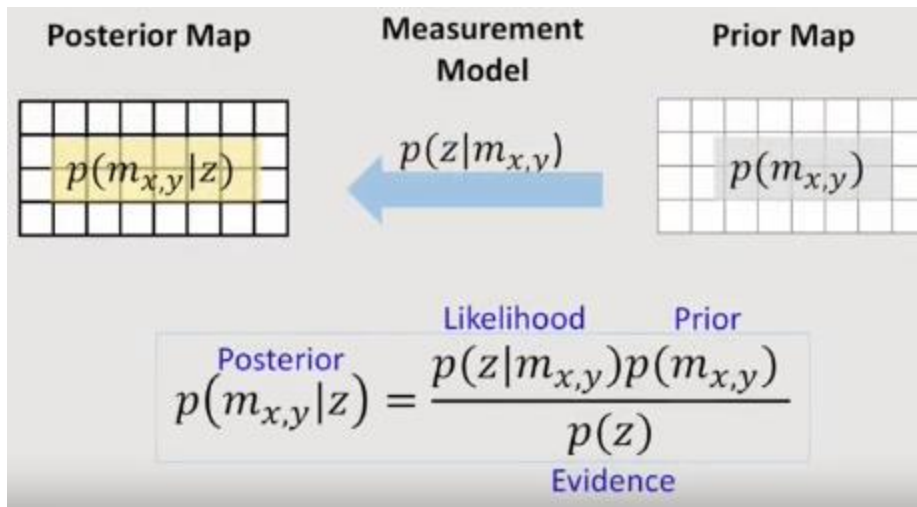
False measurement stem from:

- sensor noise,
- discretized space representation,
- moving objects, and
- uncertain knowledge of the robot motion.



There is four parameters. However, we actually have two parameters for our measurement model, because of conditional probability.





If we had some prior information of the cell, and we may take that into consideration, according to Bayes' rule.

3.2.2. Log-odd Update

If there is a probability of something happening, written as $p(X)$ and the odds can be considered as a ratio.

$$Odd: = \frac{(X \text{ happens})}{(X \text{ not happens})} = \frac{p(X)}{p(X^c)}$$

Applying Bayes' rule, we can rewrite the odds to include the sensor model term, the prior term, or both, the numerator and the denominator.

• Odd

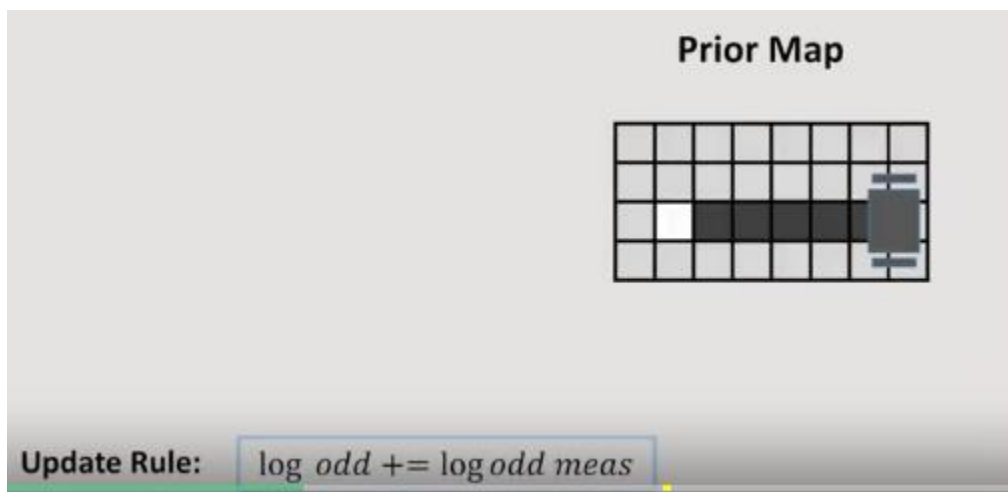
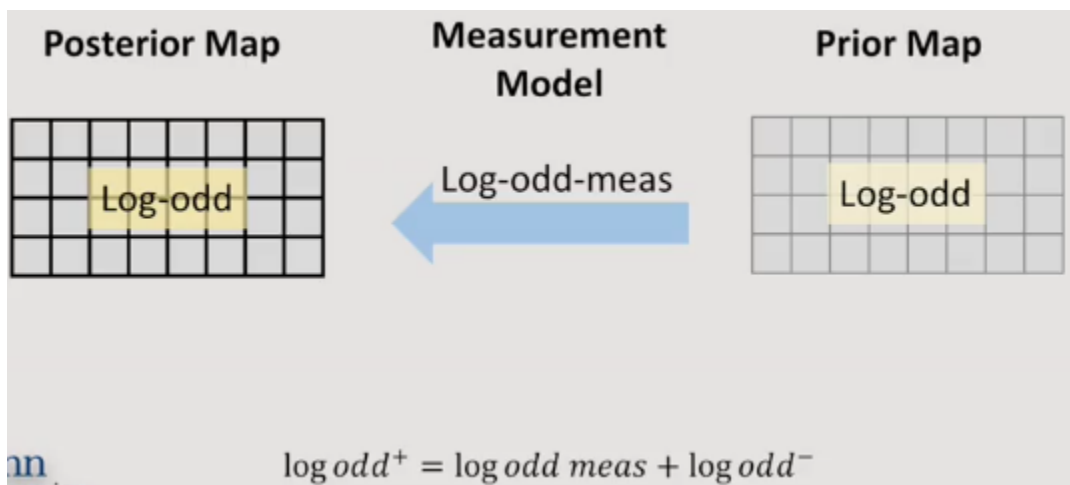
(Bayes' Rule)

$$p(m_{x,y} = 1|z) = \frac{p(z|m_{x,y} = 1)p(m_{x,y} = 1)}{p(z)}$$

$$Odd = \frac{p(m_{x,y} = 1|z)}{p(m_{x,y} = 0|z)} = \frac{p(z|m_{x,y} = 1)p(m_{x,y} = 1)/p(z)}{p(m_{x,y} = 0|z)}$$

Log-Odd: $\log \frac{p(m_{x,y} = 1|z)}{p(m_{x,y} = 0|z)} = \log \frac{p(z|m_{x,y} = 1)p(m_{x,y} = 1)}{p(z|m_{x,y} = 0)p(m_{x,y} = 0)}$

$$= \log \frac{p(z|m_{x,y} = 1)}{p(z|m_{x,y} = 0)} + \log \frac{p(m_{x,y} = 1)}{p(m_{x,y} = 0)}$$



Applying the update rule:

- the update is done only for observed cells,
- the updated values become priors when we receive new measurements in the future time steps.

- Measurement model in log-odd form

$$\log \frac{p(z|m_{x,y} = 1)}{p(z|m_{x,y} = 0)}$$

- Two possible measurement:

Case I : cells with z=1 $\log odd_{occ} := \log \frac{p(z = 1|m_{x,y} = 1)}{p(z = 1|m_{x,y} = 0)}$

Case II : cells with z=0 $\log odd_{free} := \log \frac{p(z = 0|m_{x,y} = 0)}{p(z = 0|m_{x,y} = 1)}$

(Trivial Case : cells not measured)

- Example

Constant Measurement Model

$$\log odd_{occ} := 0.9$$

$$\log odd_{free} := 0.7$$

Initial Map:

$$\log odd = 0 \quad \text{for all } (x,y)$$

$$\begin{array}{c} \updownarrow \\ p(m_{x,y} = 1) = p(m_{x,y} = 0) = 0.5 \end{array}$$

Update Rule:

$$\log odd += \log odd_{meas}$$

t0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

- Example

Constant Measurement Model

$$\log \text{odd}_{\text{occ}} := 0.9$$

$$\log \text{odd}_{\text{free}} := 0.7$$

Update

- Case I : cells with $z=1$

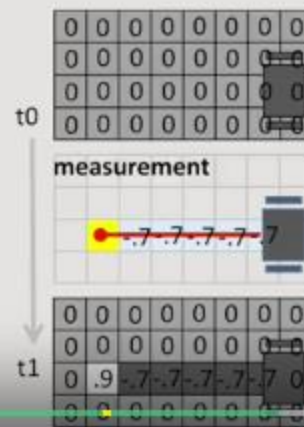
$$\log \text{odd} \leftarrow 0 + \log \text{odd}_{\text{occ}}$$

- Case II : cells with $z=0$

$$\log \text{odd} \leftarrow 0 - \log \text{odd}_{\text{free}}$$

Update Rule:

$$\log \text{odd} += \log \text{odd}_{\text{meas}}$$



- Example

Constant Measurement Model

$$\log \text{odd}_{\text{occ}} := 0.9$$

$$\log \text{odd}_{\text{free}} := 0.7$$

Update

- Case I : cells with $z=1$

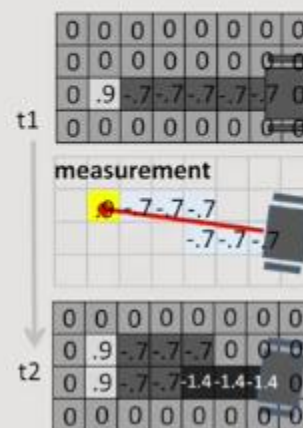
$$\log \text{odd} \leftarrow 0 + \log \text{odd}_{\text{occ}}$$

- Case II : cells with $z=0$

$$\log \text{odd} \leftarrow 0 - \log \text{odd}_{\text{free}}$$

Update Rule:

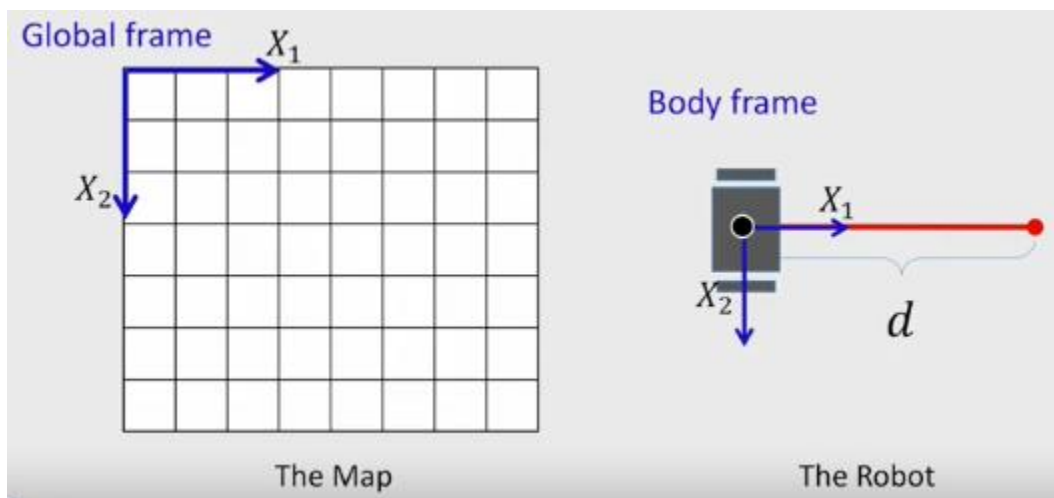
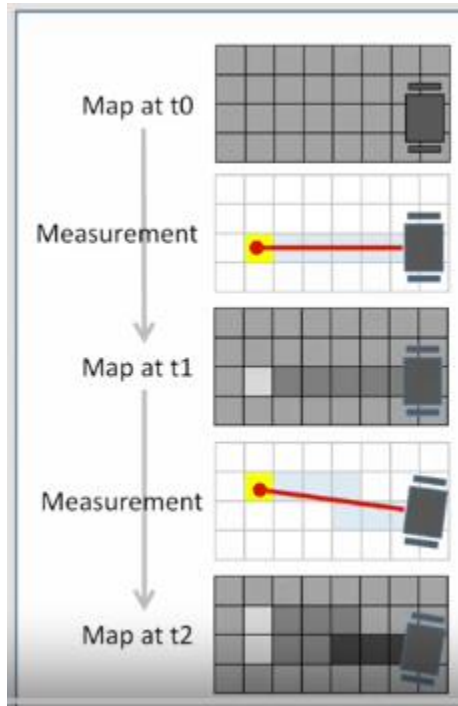
$$\log \text{odd} += \log \text{odd}_{\text{meas}}$$



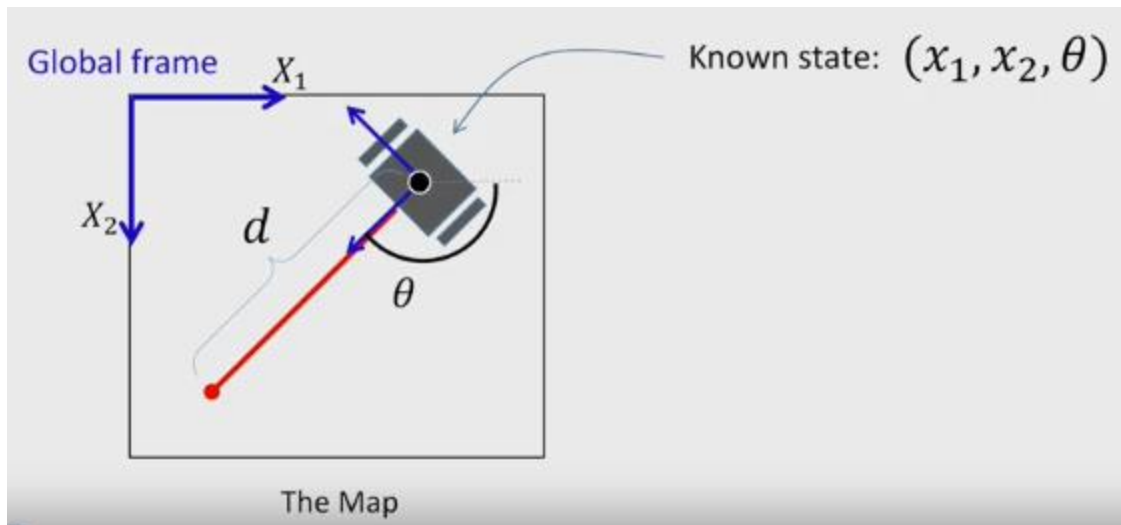
When we receive a new measurement, we update the observed cells in the same way as the previous moment.

You can see that as the cells are observed multiple times being free, they start to get darker. This means that our belief of those cells being occupied gets lower.

3.2.3. Handling Range Sensor

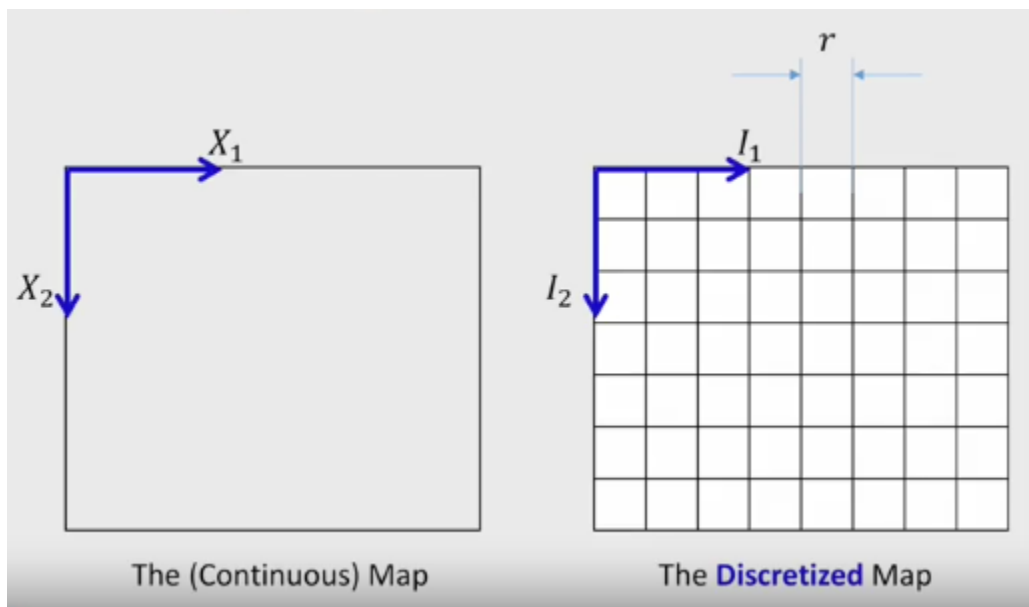


If we know where the robot is located on the map, and which direction it is heading, then we can place the robot on the map like this:

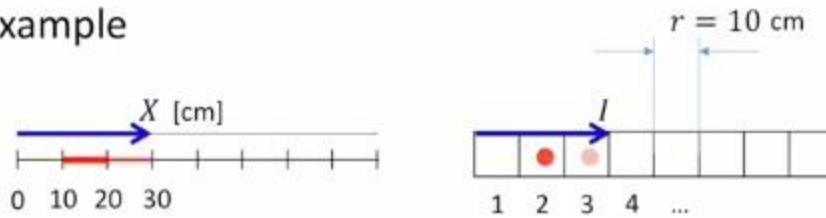


To find the coordinates of the obstructing point, we use the distance measurements in the known pose of the robot.

Up to this point, we use the continuous representation for the position. However, the map we are going to build should be represented as discretized cells of a certain resolution.

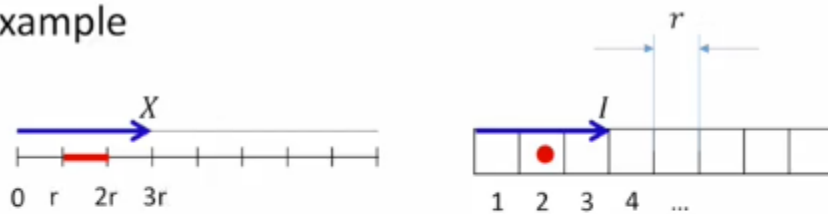


Example



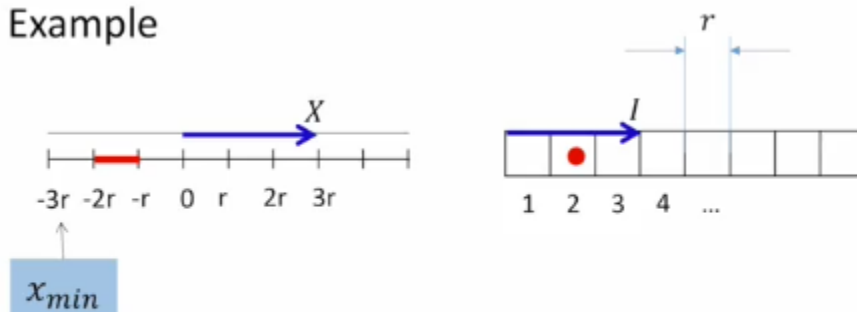
[cm]	$0 < x \leq 10$	\Rightarrow	$i = 1$	[index]
	$10 < x \leq 20$	\Rightarrow	$i = 2$	
	$20 < x \leq 30$	\Rightarrow	$i = 3$	

Example

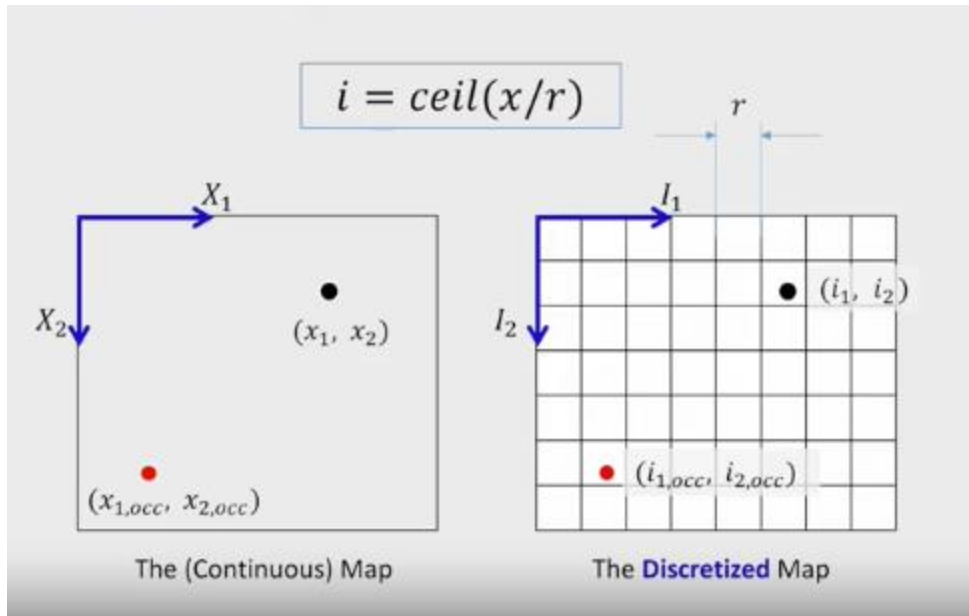


$$i = \text{ceil}(x/r)$$

Example



$$i = \text{ceil}((x - x_{min})/r)$$



Each component, X_1 and X_2 , can be treated independently for the computation of the index pair, i_1 and i_2 .

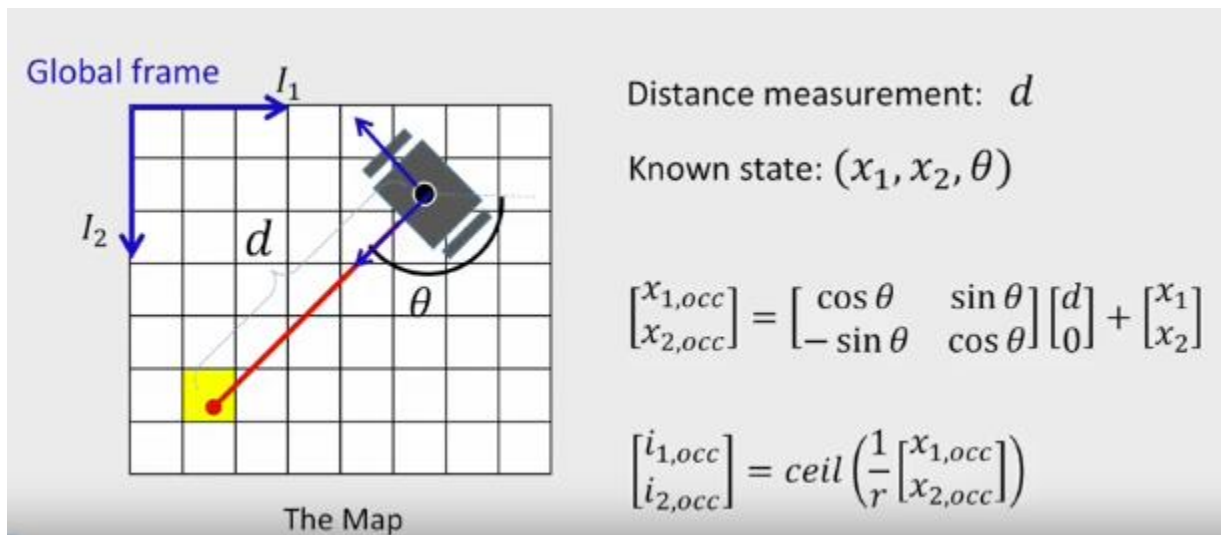
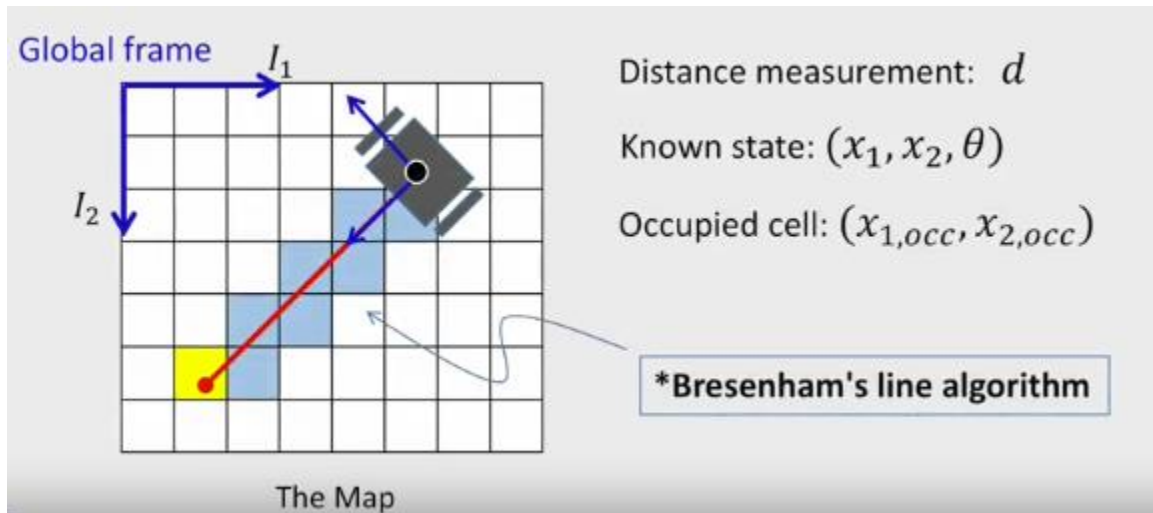


Figure 3.10. Obtaining the 2D grid indices of the occupied cell.

Given: the distance measurement d and the pose of the robot.

Compute:

1. the location of the obstructing point on the continuous domain,
2. the indices of the point on the discretized map of resolution R ,
3. the indices of free empty cells.



Bresenham's line algorithm: the algorithm takes two points as the input argument and returns a list of cells that forms an approximation for a line segment between the two points.

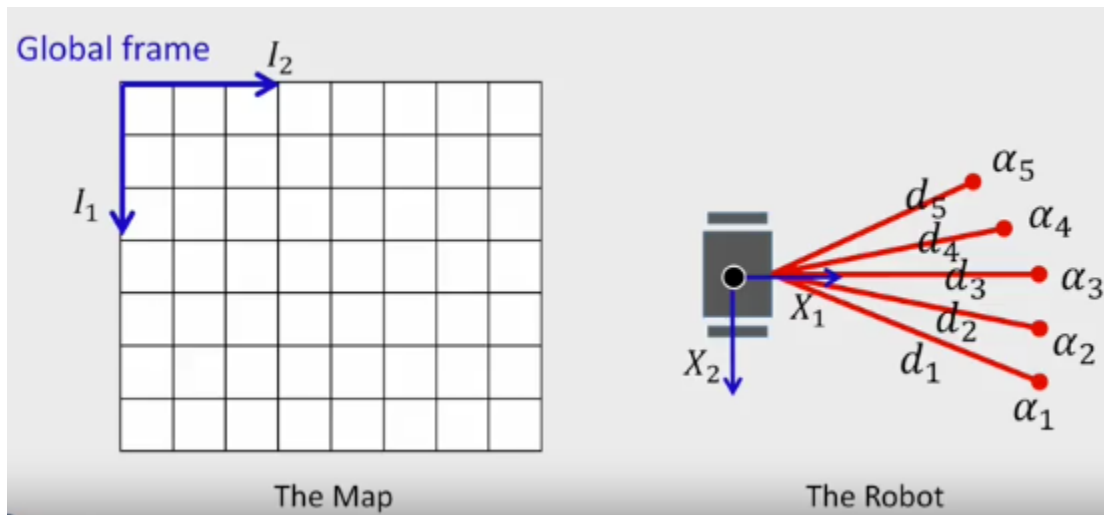
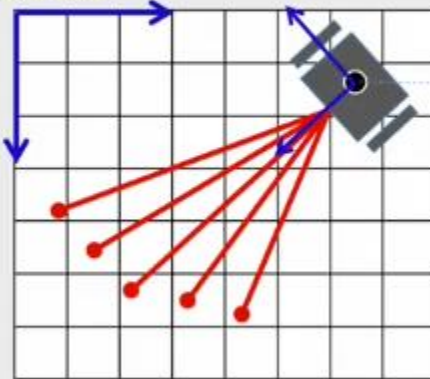


Figure 3.11. Sensor with multiple rays emitted in different directions.

Global frame



The Map

Distance measurement:

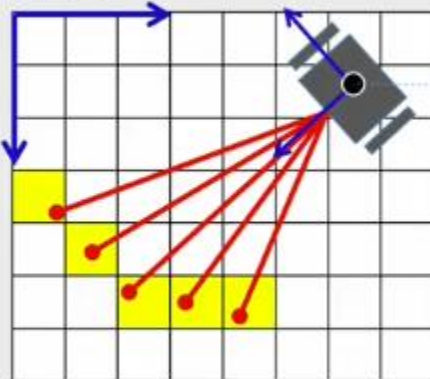
$$(d_1, d_2, d_3, d_4, d_5)$$

Directions of rays:

$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$$

Known state: (x_1, x_2, θ)

Global frame



The Map

Distance measurement:

$$(d_1, d_2, d_3, d_4, d_5)$$

Directions of rays:

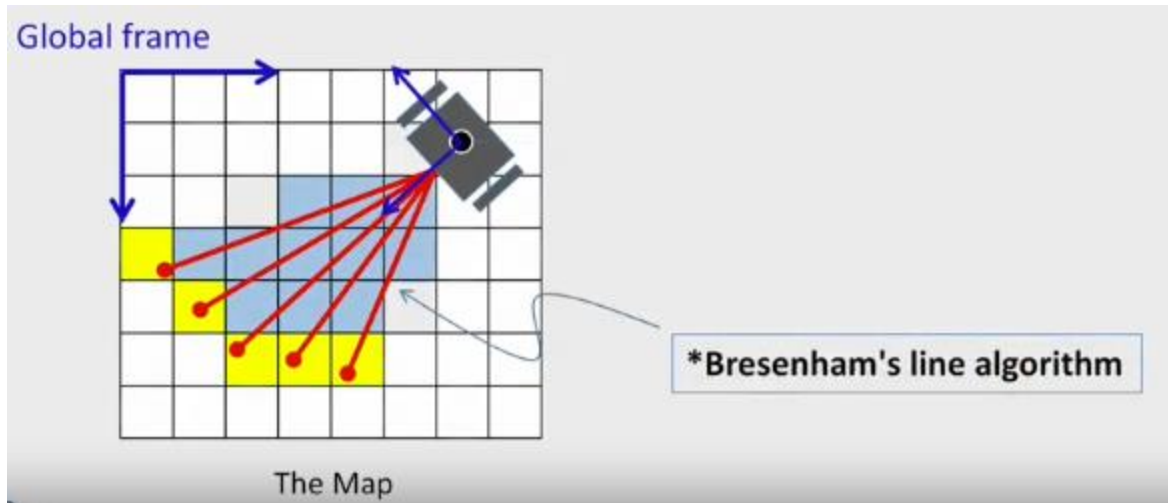
$$(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$$

Known state: (x_1, x_2, θ)

For k -th occupied cell:

$$\begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix} = \begin{bmatrix} d_k \cos(\theta + \alpha_k) \\ d_k \sin(\theta + \alpha_k) \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} i_{1k} \\ i_{2k} \end{bmatrix} = \text{ceil} \left(\frac{1}{r} \begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix} \right)$$

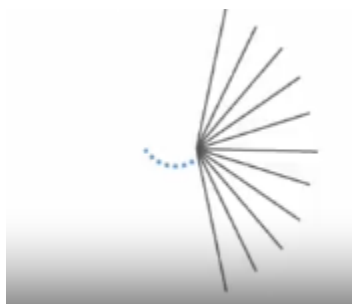


3.3. 3D Mapping

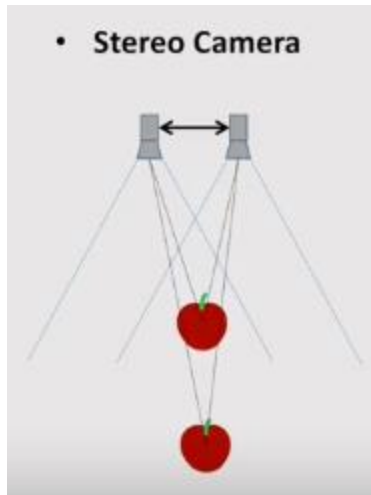
When robots interact with the real world, 2D maps can be very limiting. We eventually want to sense the 3D world and represent 3D spatial information in our map.

Sensors for 3D mapping:

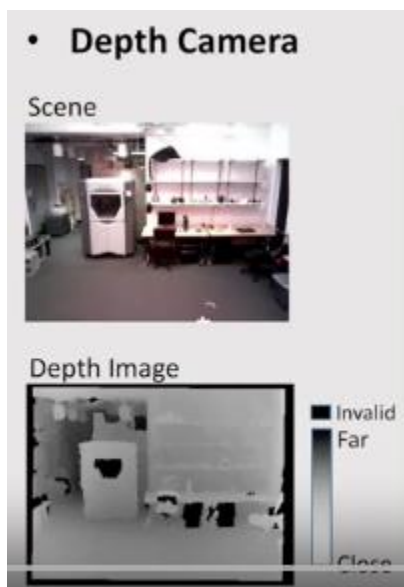
1. LIDAR range sensor,
2. stereo cameras,
3. depth camera.



LIDAR: If the sensors can see environments while rotating around an axis as illustrated, then we can generate a 3D point cloud.



Stereo cameras work like human eyes. Two cameras will observe the same scene while slightly apart from each other. The slight separation makes depth information computable.

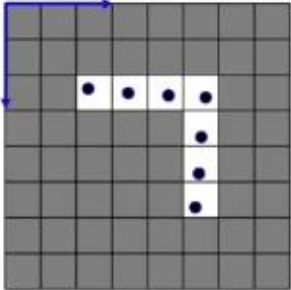


A depth camera gives depth information as just another image. Most depth cameras work in indoor environments. They are more applicable to near distance tasks than LIDAR sensors.

Map representations:

- Grid representation,
- List representation,
- Three representation:
 - kd-tree,
 - octree.

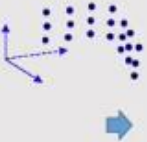
Grid Representation



- Immediate access to a cell

- Requires Large memory
(map size) $\sim (\text{map range}) / (\text{resolution})$
- Lose information from discretization

List Representation




2.3, 2.3, 1.0
2.5, 3.4, 1.1
2.6, 4.6, 1.0
2.7, 5.7, 1.1
3.8, 5.5, 1.2
4.8, 5.4, 1.0
5.8, 5.2, 1.1
...

- Takes long to search ($O(N)$)

- Requires less memory
(map size) $\sim (\# \text{ Occupied Points} = N)$
- No discretization

In 3D, N is usually very large.

Tree Representation



```

graph TD
    A["2.6, 4.6"] --> B["2.5, 3.4"]
    A --> C["3.8, 5.5"]
    B --> D["2.3, 2.3"]
    C --> E["4.8, 5.4"]
    C --> F["2.7, 5.7"]
    E --> G["5.8, 5.2"]
  
```

- Reasonable search time ($O(\log N)$)

- Requires less memory
(map size) $\sim (\# \text{ Points})$
- No discretization

To build a 3D map, from 3D point cloud we can use a special data structure called a **tree** that is organized for efficient maintenance of data points.

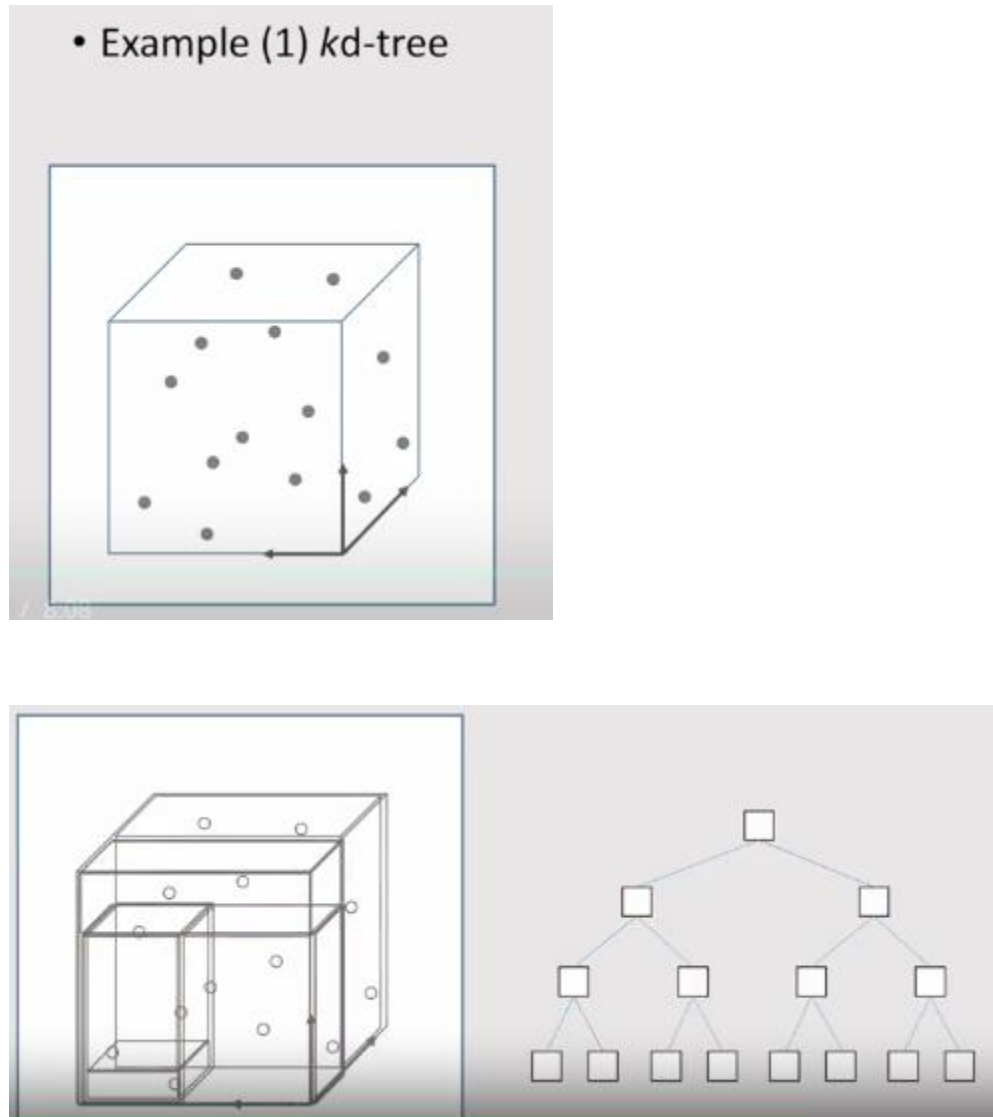


Figure. kd-tree, a binary tree.

- Example (2) Octree

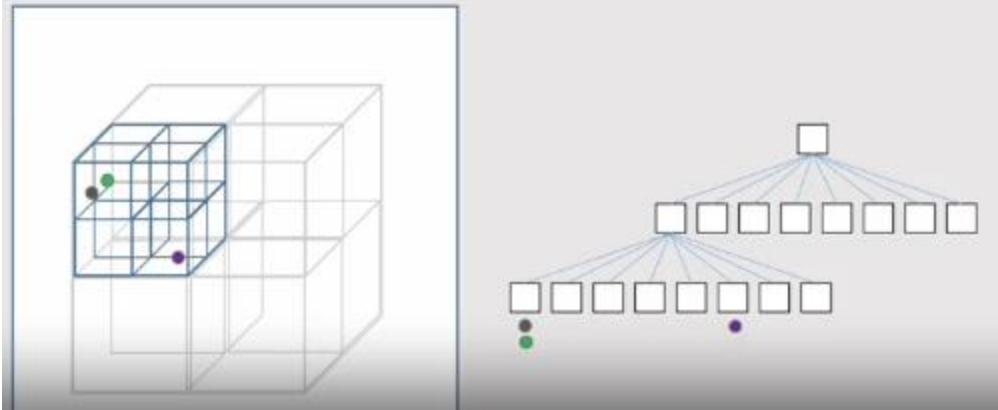


Figure. Octree.

As compared to the kD tree which is a binary tree, octree creates eight branches from a single node. Then that node which represents a box in 3D is divided into smaller octants. Thus Octree represents the whole world as recursive octants.

The nodes at the lowest level may contain the data points. And parents nodes contained a pointer to their children nodes.

Note that this representation is different from a full grid. Since our node stops expanding its branches when the corresponding box is empty.

PROGRAMMING ASSIGNMENT:

2D Occupancy Grid Mapping

4. Bayesian Estimation – Localization

4.1. Odometry Modeling

Odometry provides a measurement of how far the robot has moved. Odometry is just one method of finding the robot's location in the world.

Global car navigation:

- GPS
- Cell tower
- WiFi

These sources represent global knowledge of position, exact coordinates. Odometry and other sources of information can augment the global localization sources with local knowledge. “How have they changed coordinates?”

These sources of information are more precise, giving centimeter accuracy. However, integrating sources, like encoders and gyroscopes, over time can lead to drift. This is due to the accumulation of errors in time.

Other local sensors like laser scanners and color and depth cameras can help to correct these errors.

Odometry updates start with modeling the robot. Different robots, such as humanoids or aerial vehicles, will require different models.

Tracking Angular Movement

- Encoder ticks (e) are observed at the inner and outer radii

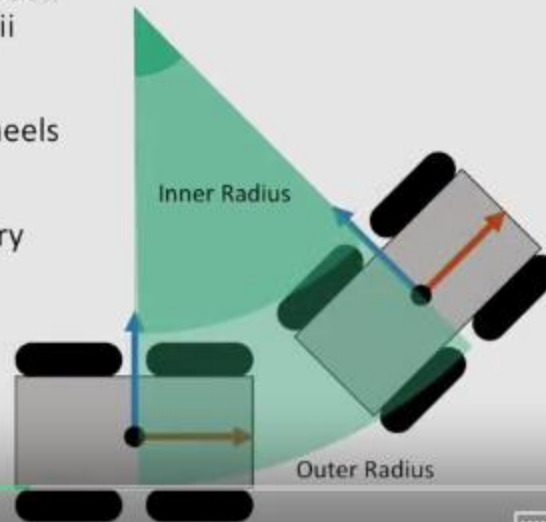
$$e_i = \theta r_i \quad e_o = \theta r_o$$

- Known width between wheels

$$w = r_o - r_i$$

- Calculate angular odometry

$$\theta = \frac{e_o - e_i}{w}$$



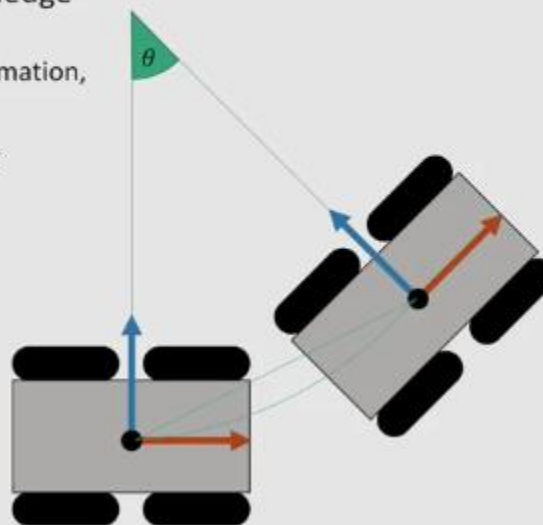
7 5:03

Tracking Translational Motion

- Translation requires knowledge of the angular movement
 - Use circular sector approximation, valid for small movements
- Quiz: Spinning in Place*

$$y = \frac{e_o + e_i}{2} \cos \theta$$

$$x = \frac{e_o + e_i}{2} \sin \theta$$



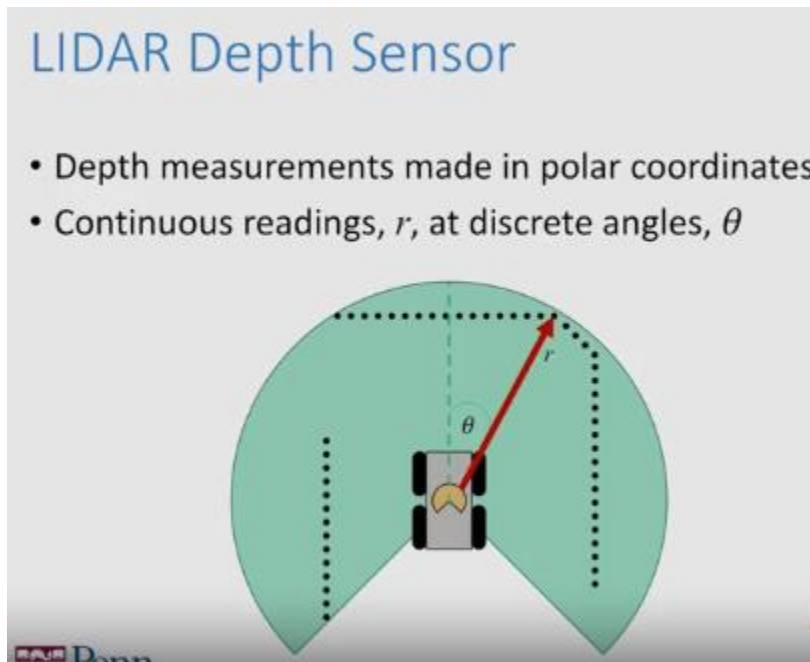
g

Simple approach characteristics:

- Local frame of reference of the robot starting point,
- Issue: Encoders suffer from slippage, missing counts,
- Issue: Gyroscope integration suffer from drift,
- Utilizing maps of the world can correct localization errors.

4.2. Map Registration

Goal: location a robot on a map given laser range data.

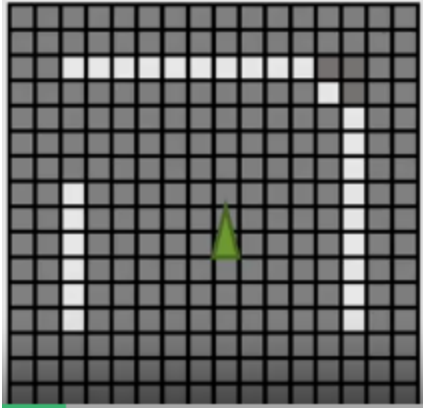


LIDAR stands for light detection and ranging and it provides distance measurements, often engineered in a laser scanner to provide two dimensional data.

Here we will model depth measurements in polar coordinates, where a continuous distance reading r is made at discrete angles θ . Here, θ encompasses 270 degrees, not a full circle. The laser scanner can only see 10 to 30 meters away.

In this range restriction, means that distance measurements showing here is black dots, can only be found within the area in green.

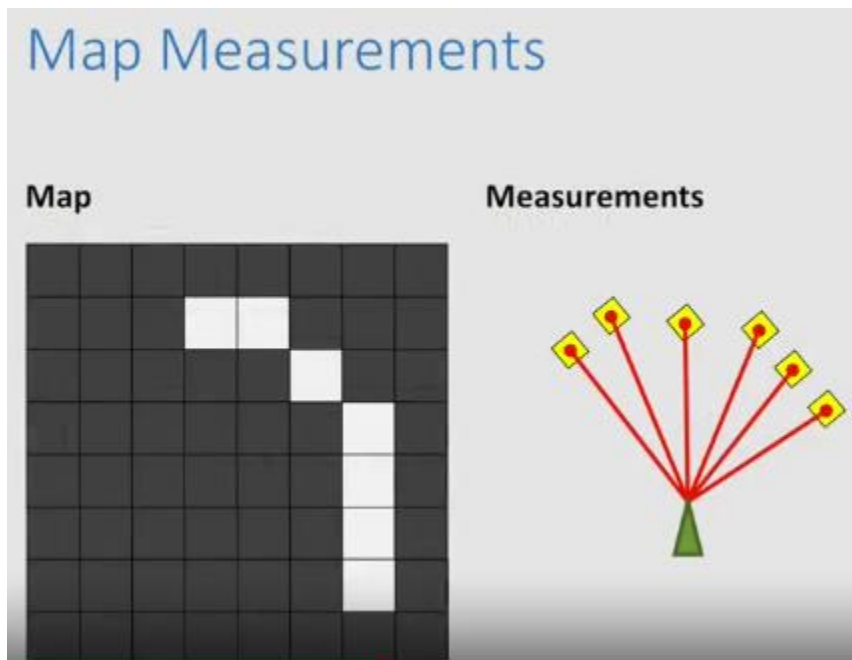
Discrete Grid representing 2D space, white cells represent the presence of an obstacle:



Because the robot lives in a finite grid world the grid must sometimes be expanded as the robot can escape the boundaries. In this case, the map representation should increase in size as the robot turns and travels on the corridor shown in the top left of this map or else information will be lost.

In addition to mapping the laser data, we can access map data and try to find the robot pose in the map given the laser data.

The complimentary stages of mapping and localization when performed together are known as SLAM, simultaneous localization and mapping, which is a major research topic in robotics.

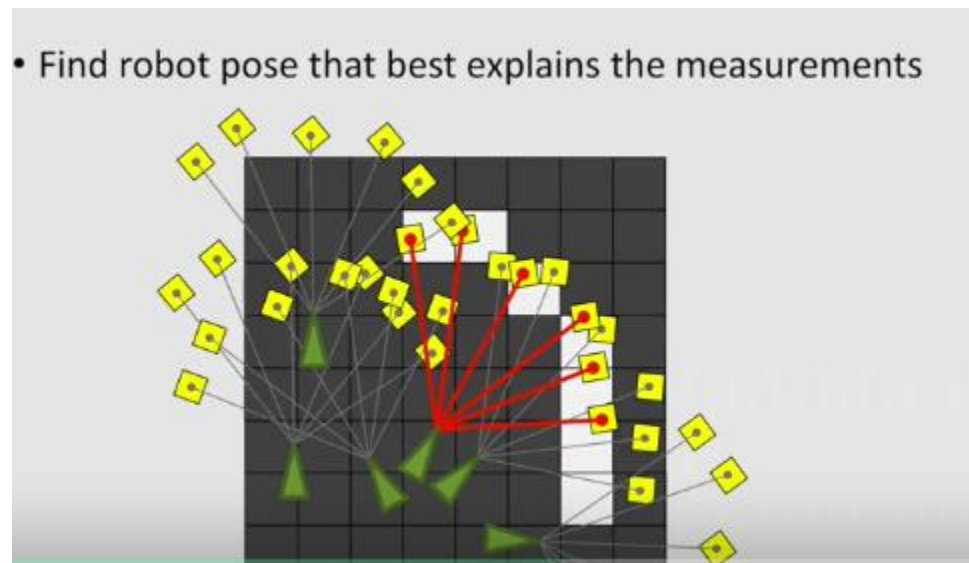


In the localization problem we have two sets of information:



- First, the occupancy grid map provides a grounds truth knowledge of what the robot should expect to observe in the world.
- Second, the set of lighter scan measurements provides information on what the robot is observing at the current time.

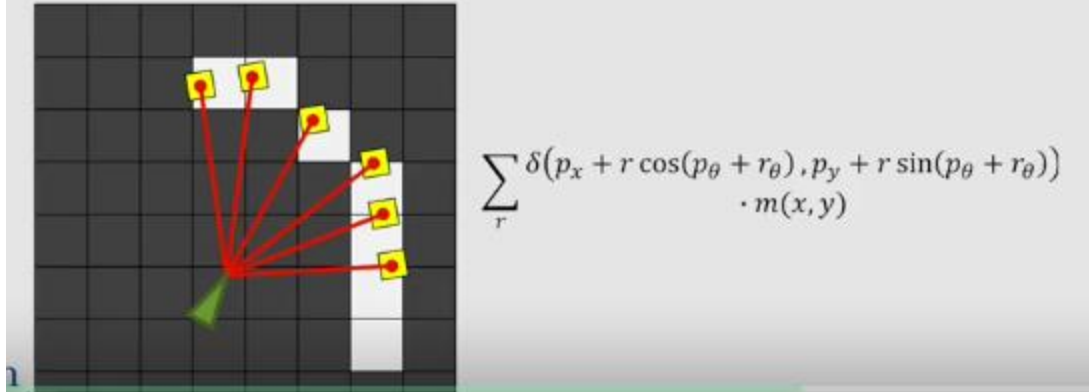
Goal: find the best robot pose on the map that explains the measured observations.

Searching over all possible poses of the robot can be difficult. But based on the odometry information, we have some tricks to make the search easier. We can constrain the search to a limited number of poses based on odometry information. Because we track the robot over time, we have the last known position of the robot and odometry information on how far the robot most likely moved. Thus, the most likely pose for the robot is now given a new set of laser data, is probably close to where the odometry predicts the robot to be. This prediction means that we can refine our search to poses near the prediction and be more confident in the validity of our search results.



Map Registration

- Correlate laser obstacles with map obstacles 
- Correlate laser free space with map free space 



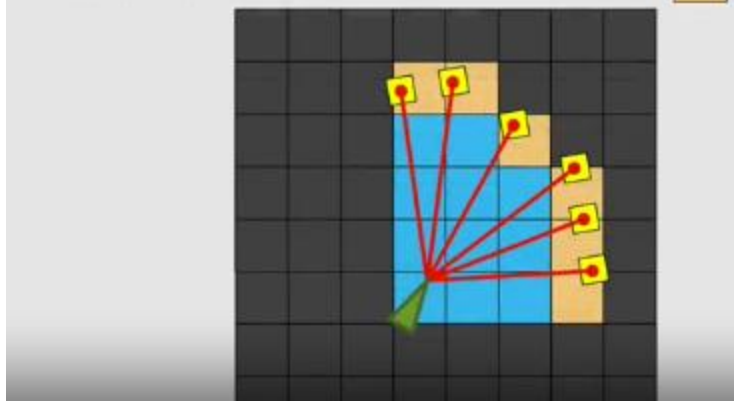
One metric is to consider the sum of the map values m , at coordinates x and y , where the laser returns r , hit.

This correlation metric can be modified to suit the application at hand. In our case, the value of our map cell will be a log odds ratio, so laser returns that are seen at a map location with high probability of occupancy will strongly increase the registration in the metric score. Laser returns with map locations known as free cells will decrease the metric score.

Additionally, the correlation can be scaled where returns from far distances affect the metric calculation less than nearby the laser returns.

We register the robot on the map, at the pose that maximizes the registration metric. Thus, when the odometry is calculated, it uses this pose to predict a new position of the robot, in time.

- Laser scans penetrate free space
- Laser scans return distance to obstacle



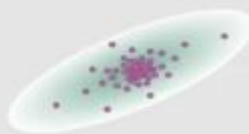
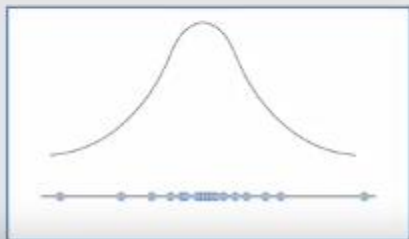
In addition to considering merely the laser returns, we can consider points for the laser returns penetrated. This calculation can further corroborate our map registration. It requires considerably more computation however.

To capture pose uncertainty using a simple Gaussian on position and angle may not provide a feasible approach.

4.3. Particle Filters

Particle Filter represents a distribution with a set of samples, referred to as particles. Instead of a fully defined function.

- Samples approximate a probability distribution
- Fast and efficient non-parametric model
- Ability to represent multimodal distributions
 - Mixtures of Gaussians, multi-hypothesis Kalman Filter



The statistics of the samples match the statistics of the distribution, such as the mean or standard deviation. However, they can be more complicated metrics as well.

In this way, there are no parameters as were seen in the mean and covariances of the Gaussian models. Instead, a full population is tracked.

In essence, the particle filter population represents a mixture of Gaussian distributions.

Here, the variance will go to 0. With 0 variance, the Gaussian distributions become Dirac Delta functions.

- Dirac Delta function
 - Sigma is going to zero, Gaussian distribution
- Particle Filter : Limit of Gaussian mixtures when $\sigma \rightarrow 0$ (variances shrink to zero)

Initial Population

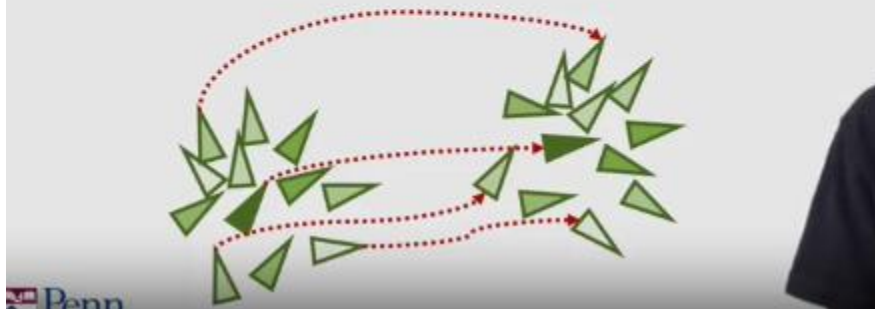
- Initial group of particles represents the underlying distribution of the belief state
- Particle is comprised of (pose, weight)
- Here, darker colors represents a higher *weight*
 - Represents probability, such that $weight = prob(pose)$



Initially, a set of particles represent the underlying belief state. Each particle is a pair of the pose and the weight of that pose. This is similar to representing a probability function where the weight is the probability of that pose in the underlying distribution.

Odometry Update

- Move the particles based on odometry information
- Each particle represents a possible pose, so individually must be moved via its local frame

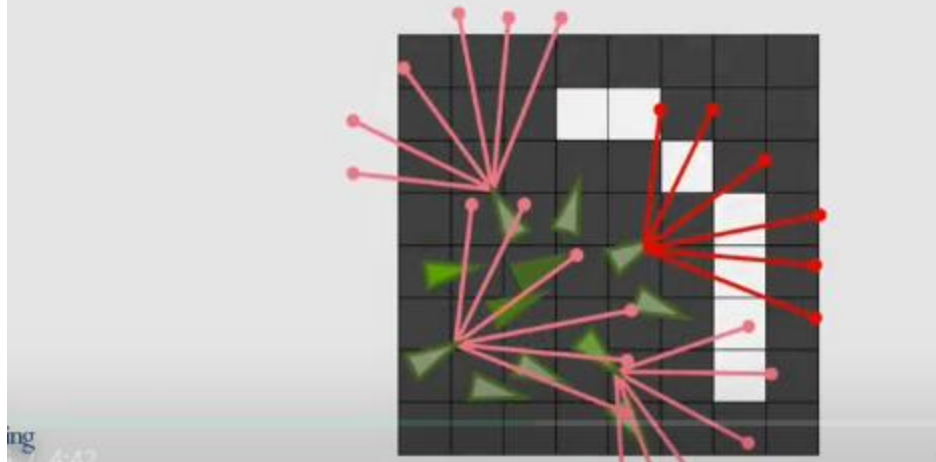


- Include odometry noise model
- Sampled for each particle from the odometry noise distribution $p'_i = p_i + \mathcal{N}(0, \Sigma)$

- Dispersion of particles represents the added uncertainty from moving

Correlation Update

- The weights of the particles can be updated based on LIDAR correlation data, $w'_i = w_i \cdot \text{corr}(p_i)$



- The new set of particles capture the distribution after odometry and sensor measurement
- However, this may not be the optimal set to represent the distribution

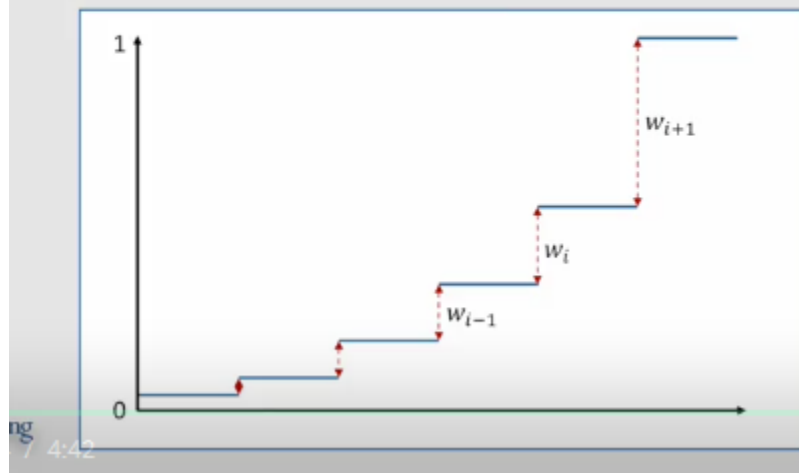


- Check a resampling criterion – the number of effective particles
- If the number of effective particles is too low, then *resample* to increase the effective number

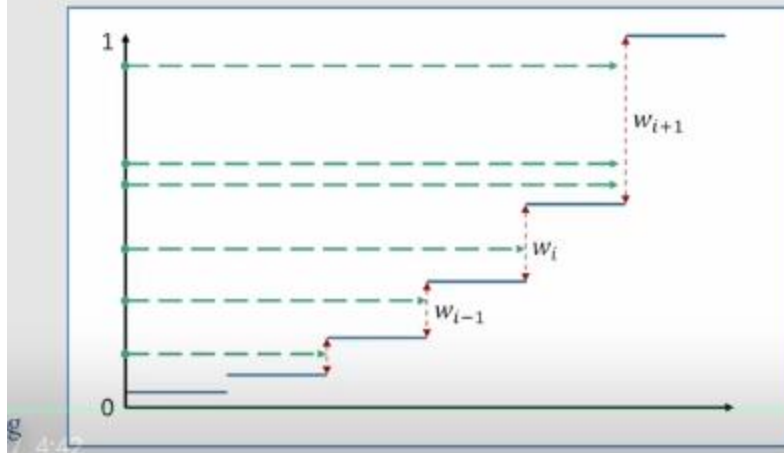


$$n_{effective} = \frac{(\sum_i w_i)^2}{\sum_i w_i^2}$$

- Use the cumulative probability to aid in resampling
- Sum of normalized weights is 1

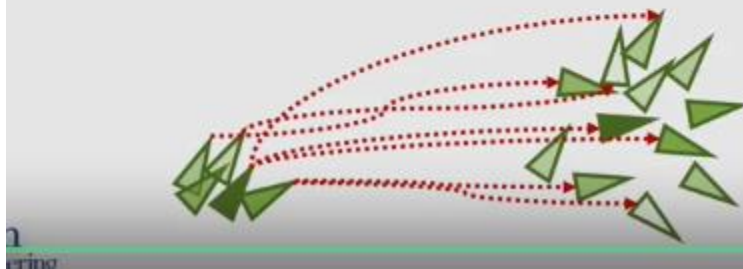


- Sample number uniformly between 0 and 1 of the cumulative range, and find which w_i includes that number



Particle Resampling

- The particles with the indices found in the resampling approach become the new set of particles to be fed into the next odometry update
- Particles may be duplicated, but the odometry noise will differentiate these particles.



4.4. Iterative Closest Point Algorithm – ICP

When not constrained to the ground, a point has six degrees of freedom which requires exponentially more points to be sampled in order to produce reasonable registration performance.

Instead of relying on particles, we can use a *direct optimization* to find the registration between our measurements and the map.

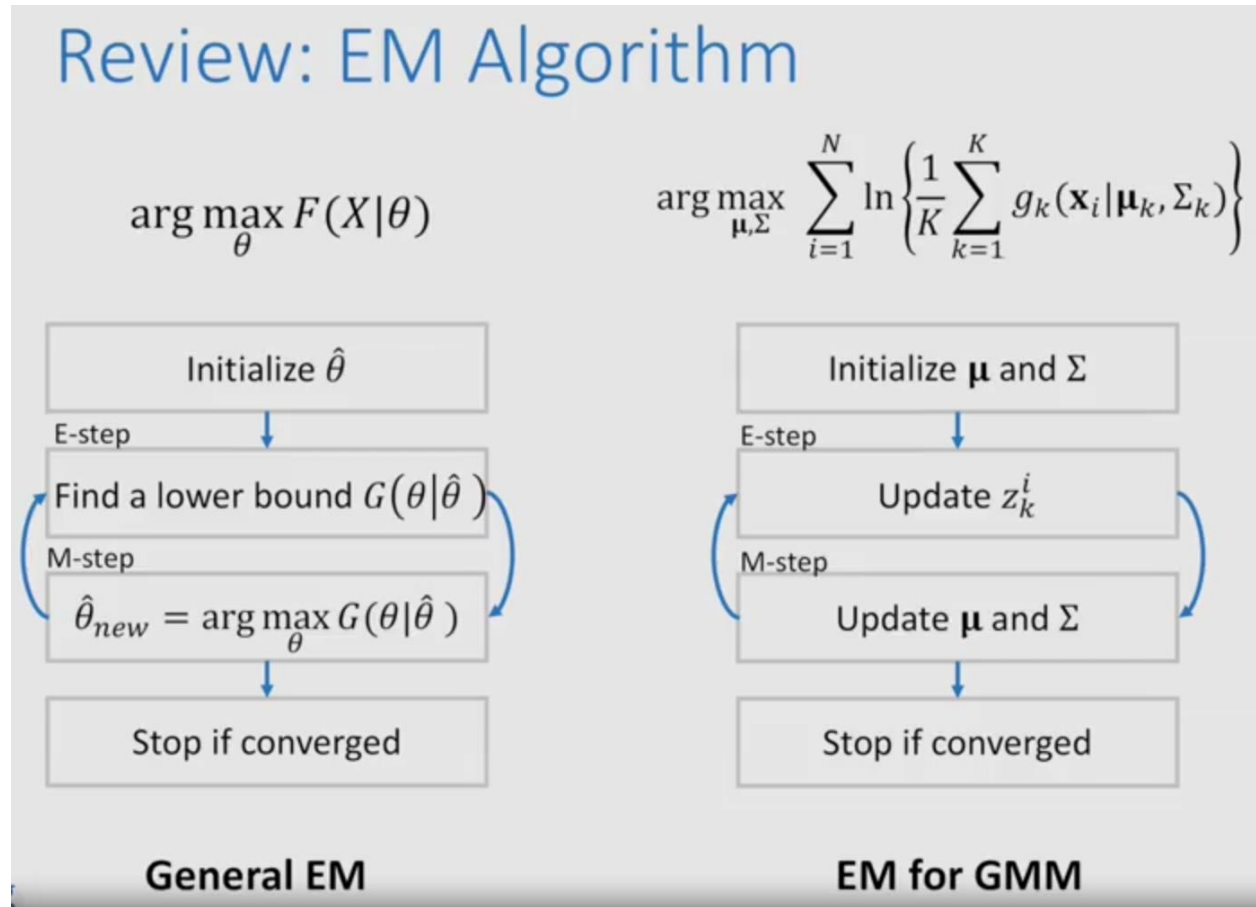
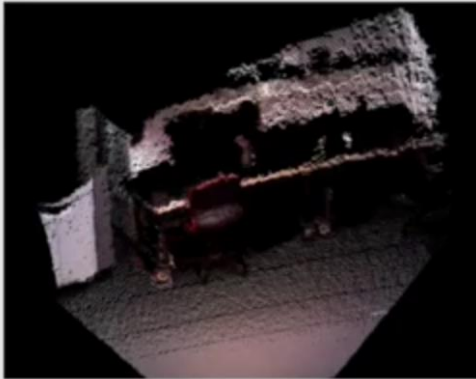


Figure 4.1. Expectation maximization (EM) algorithm.

Review: 3D Map Representation

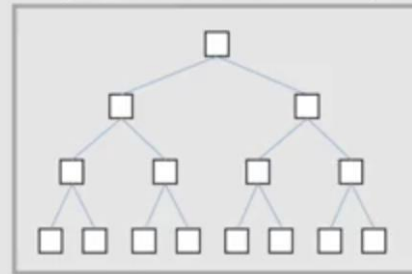
3D point cloud measurement



Map visualized in 3D



Implementation Example

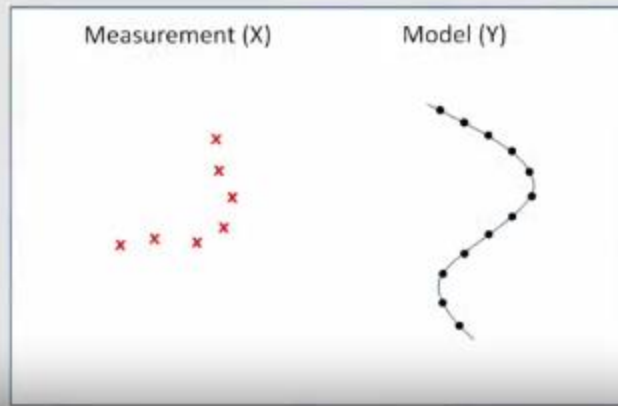


A 3D map is usually represented as a ***tree structure*** instead of as a full grid as done in two dimensions. This is in order to have efficient means.

The map can keep the full precision of point location. Due to the special organization, we can speed up the finding of the closest point to a given point in each map update.

ICP Algorithm

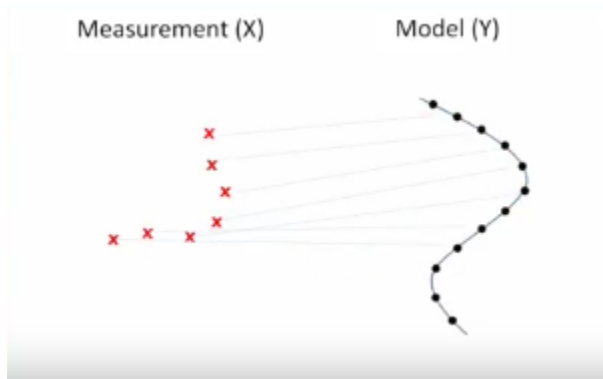
- Problem: Register two point sets X and Y.

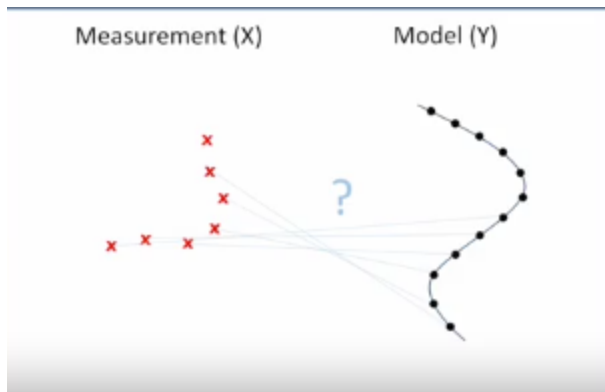


We have two sets of points, one of them is a point cloud as a measurement and the other is a point cloud of the map model.

Goal: put the newly measured points into the right place on the map model.

In order to do so, we need to find a rotation and translation that move the measured points to match the model points. Additionally, we need to know which measurement points correspond to which points in the model. We can visually detect the corresponding parts in this example.





However, for a robot with tens of thousands of noisy points it is not obvious to see the possible matching pattern.



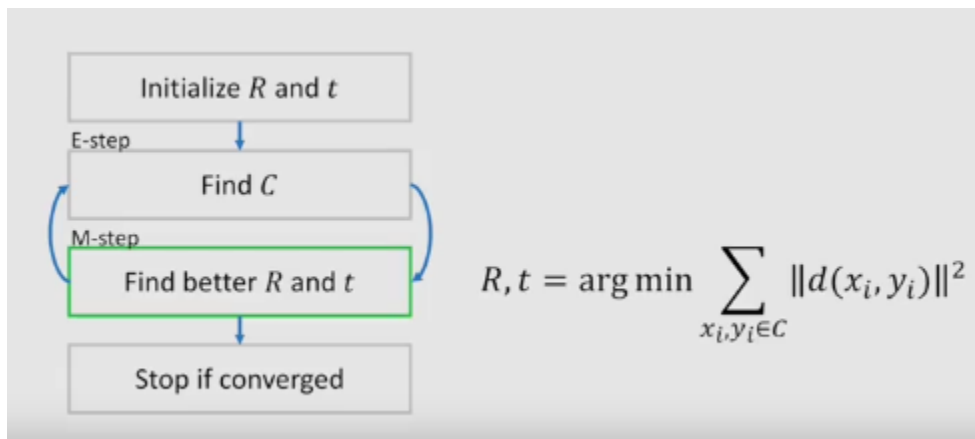
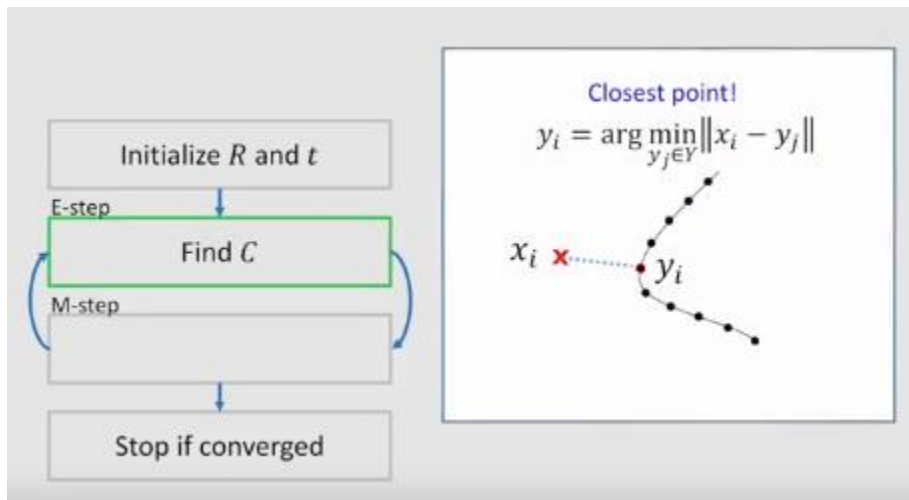
The strategy of the ICP algorithm takes an optimistic assumption that the point sets are close enough. In this way we have a good prior of the rotation r and translation, t .

Under this assumption, the correspondence of a point will be the closest point to that one. In this way, we will find closest points of all measured points corresponding to the map.

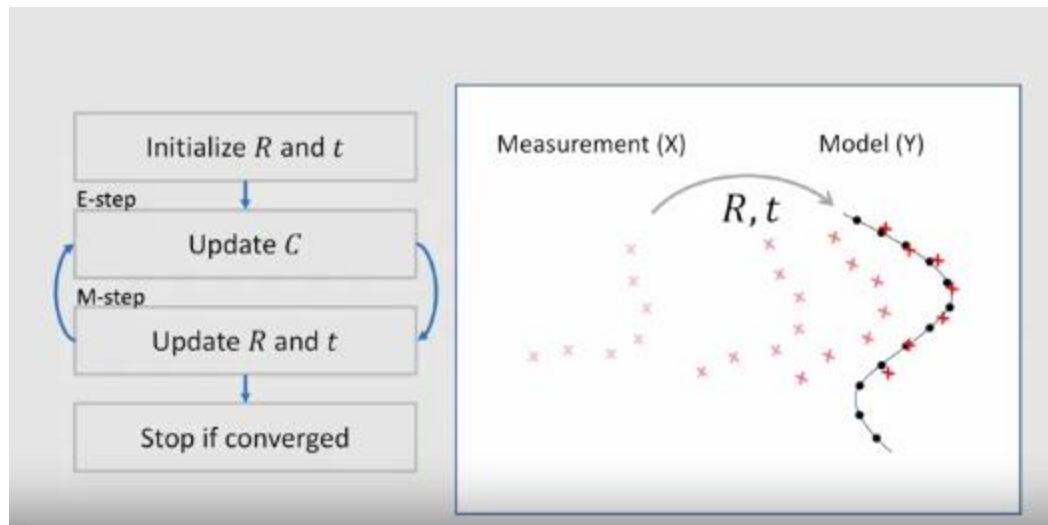
When a map has a tree structure, this process is much faster than a brute force search. Once we have our correspondences, we can enhance our estimate of R and t , by solving this optimization.

If you are interested, the cited paper gives details of the solution in order to obtain R and t . It's good practice.

[SOLUTION] K. Arun, T. Huang, and S. Blostein, "Least-squares fitting of two 3D point set", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5), pp. 698–700, 1987.



After each iteration, we will have better and better correspondences in addition to better registrations. We iterate this until it converges. Once it does, we can obtain the rotation and translation between the two sets of points.



ICP Example

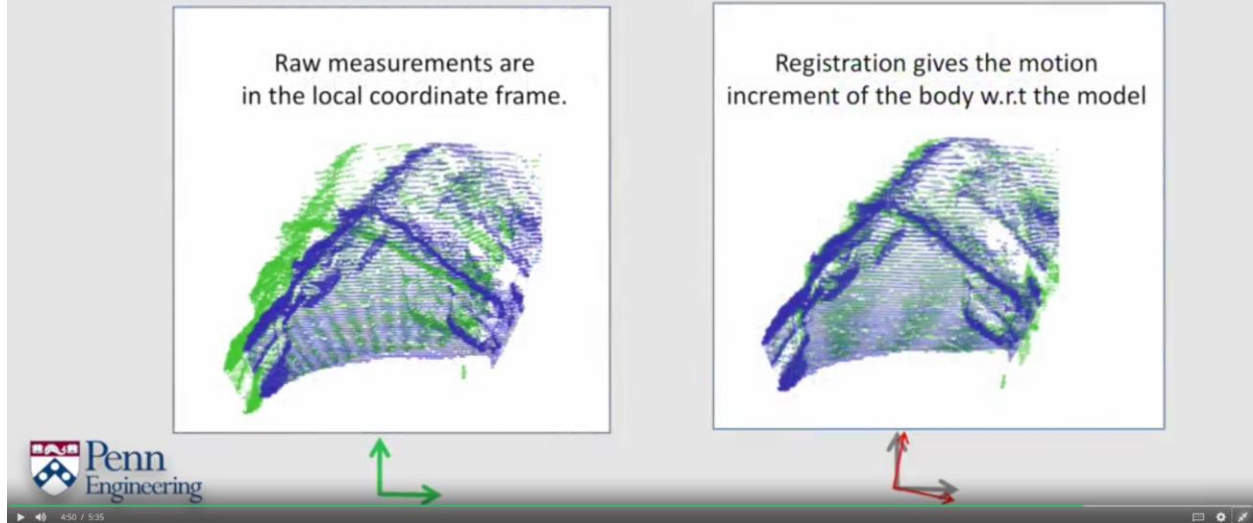
The 3D visualization shows a point cloud of a room scene, with points colored in green and blue. Below it are two 2D camera views of the same scene, showing a room with desks, chairs, and a person standing. The left view is the original image, and the right view is the transformed image, showing the result of the ICP alignment.

```

graph TD
    A[Initialize  $R$  and  $t$ ] --> B[E-step  
Update  $C$ ]
    B --> C[M-step  
Update  $R$  and  $t$ ]
    C --> B
    C --> D[Stop if converged]
  
```

1
ering

ICP: Motion Increment



Although we need a pretty close initial alignment ICP algorithm is widely used in many robotic applications, including *three dimensional simultaneous localization and mapping*.

The ICP algorithm could be used for localization with 3D sensors. There are many variants of this algorithm according to different optimization formulas different ways to choose correspondences and different ways to reject bad points. In general this has been a good overview of many different localization techniques.

PROGRAMMING ASSIGNMENT: Particle Filter Based Localization