CSE397/497-013
Introduction to Mobile Robotics

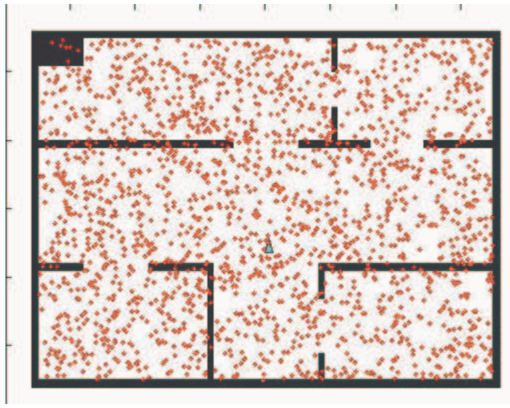
Localization & Mapping II

LEHIGH UNIVERSITY
Department of Computer Science & Engineering
P.C. Rossin College of Engineering and Applied Science

Today's Agenda

- Plagiarism
- Homework 4
  - *New due date is 10 Nov 05* ☹
  - *MPEGs required for submission (keep them small < 1 MB)*
  - *Questions*
- The plane Jane vanilla Kalman Filter

Monte Carlo Localization (Homework 5)



Bayesian Filters (1)

$$p(B \mid A) = \frac{p(A \mid B)\, p(B)}{p(A)}$$

- PFs and Kalman Filters (KF/EKF) are example of Bayesian Filters
- Bayesian filters do not *explicitly* estimate the state
- Instead, they propagate a *posterior* probability density function for the state from which it can be inferred
- In the KF, a gaussian distribution ***P*** is propagated at each timestep with mean μ and variance σ^2. The former is used as the state estimate
- In the PF, a (weighted) particle set corresponds to the posterior from which an estimate for the state can be inferred

The Particle Filter

- Unlike the KF, which represents the *pdf* parametrically as a gaussian, the PF approximates it as a sample set

$$Bel(\bar{x}) \approx \{x^i, w^i\} \;\; i \in [1..m]$$

  - m *denotes the number of particles in the sample set*
  - $\mathbf{x}^i$ *corresponds to a hypothetical state estimate*
  - w^i *corresponds to a weight reflecting a "confidence" in how well the particle* $\mathbf{x}^i$ *reflects the true state* $\mathbf{x}$

- $\sum_m w^i=1$, so that the sample set corresponds to a discrete probability density function

- It has been shown that as the number of samples approaches infinity, the sample set converges to the true posterior [Tanner, *Tools for Statistical Inference*, 1996]. However, no proofs for rates of convergence exist

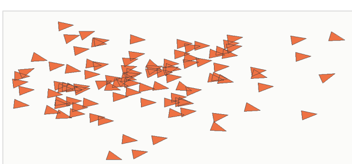
Predictor-Corrector Example

1) We have a *prior* of uniform weighted particle

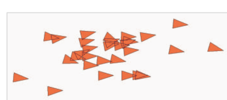
2) Particles are weighted based on the sensor measurement and resampled according to weight to generate our posterior.

3) Particles are passed through our motion model to generate a new posterior

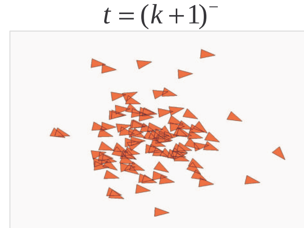
$t = k^-$



$t = k$



$t = (k+1)^-$



At this point, we have *m* unique samples

We still have *m* samples, but they are all *equally weighted* and not necessarily unique

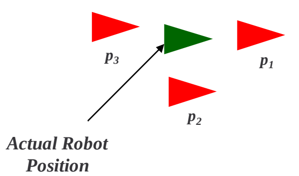
Particles are again unique and equally weighted

CSE397/497 Intro to Mobile Robotics

Predictor-Corrector Example (2)

1) We have a *prior* of uniform weighted particle

$t = k^-$



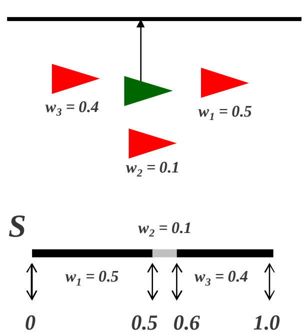
CSE397/497 Intro to Mobile Robotics

Predictor-Corrector Example (3)

2a) Particles are weighted based on the sensor measurement

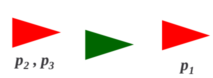
2b) Particles are resampled according to weight to generate our posterior.

$t = k$



$t = k$

$s_1 = 0.323$
$s_2 = 0.677$
$s_3 = 0.900$

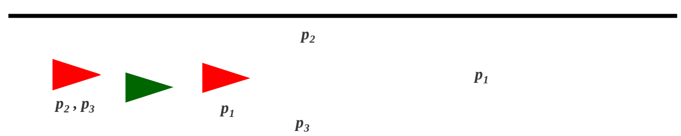


We still have *m* samples, but they are all *equally weighted* and not necessarily unique

Predictor-Corrector Example (3)

3) Particles are passed through our motion model to generate a new posterior

$$t = k + 1^{-}$$



Particles are again unique and equally weighted

4) Iterate…

The Particle Filter Algorithm

$function\ X_{k+1} = runFilter(X_k, z_k, u_k)$

$X_{k+1} = \emptyset$

$\text{for i} = 1:\text{m}$

  $\text{generate random } x \text{ from } X \text{ based on sample weights;}$

  $\text{generate random } x' \sim p(x' \mid u_k, x);$

  $w = p(z_k \mid x');$

  $\text{Insert } (x', w) \in X_{k+1};$

end

$\text{Normalize weight factors } \forall\, w_i \in X_{k+1};$

$return\ X_{k+1};$

The Particle Filter Algorithm (ver. 2.0)

$function\ X_{k+1} = runFilter(X_k, z_k, u_k)$
$X_{k+1} = \emptyset;\ X_{temp} = \emptyset;$
for i = 1 : m
  generate random x from X_k based on sample weights;
  Insert $x \in X_{temp}$
end
for all $x \in X_{temp}$
  $x = x' \sim p(x' \mid u_k, x);$
end
for all $x \in X_{temp}$
  $w = p(z_k \mid x');$
  Insert $(x,w) \in X_{k+1};$
end
Normalize weight factors $\forall\ w_i \in X_{k+1};$
$return\ X_{k+1};$

This version will be better for your Matlab implementation on the PF assignment!

$$p(B \mid A) = \frac{p(A \mid B)\,p(B)}{p(A)}$$

The MCL Problem

- In the MCL problem, our objective is to estimate position and orientation in a workspace
- We assume the availability of a map *m*
- This allows us to condition our belief to not only the current pose, but constrained to lie within a map. Thus, our belief equation becomes

$$Bel(\vec{x}_t) = \eta\, p(z_t \mid x_t, m) \int p(\vec{x}_t \mid u_t, x_{t-1}, m) Bel(\vec{x}_{t-1}) dx_{t-1}$$

- Thus, we can infer expected measurements from a given pose through:
  - Ray tracing if we are doing an occupancy grid
  - Line intersection if we are representing the map as a set of lines
- We can also combine map information with our motion model to exploit constraints in the workspace

Generating the Sensor Model

- Operation of the particle filter hinges upon associating a probability with each sensor measurement given a state so that a proper weight can be associated with each sample

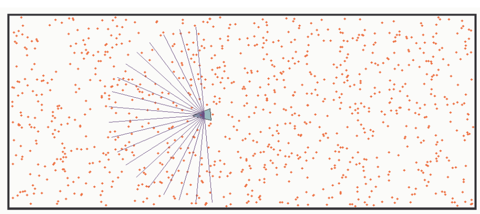
$$Bel(\vec{x}_t) = \eta p(z_t \mid x_t, m) \int p(\vec{x}_t \mid u_t, x_{t-1}, m) Bel(\vec{x}_{t-1}) dx_{t-1}$$

- This is NOT the same as sampling the probability density function of ***z***
- For a continuous distribution, the probability of measuring a specific value is zero
- Normally, sensors have a resolution which a given measurement is rounded to (*e.g.* a LRF may have a cm level resolution)
- Probabilities can then be determined by integrating the sensor *pdf* over this resolution range

CSE397/497 Intro to Mobile Robotics

Generating a MCL Specific Sensor Model

- MCL is typically performed with range sensors at known bearing angles to the robot (although cameras have also been used)
- As such, a single scan consists of numerous sensor measurements (*e.g.* from laser or sonar pulses)



- If we assume that these *n* measurements are independent, the conditional probability can then be expressed as

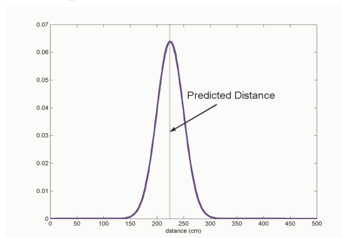
$$p(z_t \mid x_t, m) = \prod_{i=1}^{n} p(z_t^i \mid x_t, m)$$

© JR Spletzer

CSE397/497 Intro to Mobile Robotics

MCL Sensor Model Issues (1)

- A shortcoming of particle filters is that they tend to fail if the sensor models are too accurate
- This can result from a not generating an initial sample close enough to the true state estimate
- One potential solution is to inflate the sensor model error. For example, the standard deviation for the SICK LRF is modeled as σ≈25cm when in reality it is closer 1cm.
- This violates the basis from which the PF was derived, but has basis in actual measurements and works well in practice

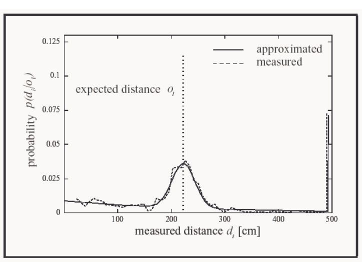


Autonomous Mobile Robots, Chapter 5

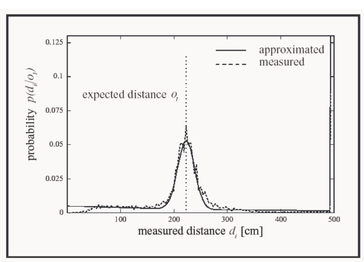
5.6.2

Markov Localization

- The critical challenge is the calculation of $p(i|l)$ $\quad p(l|i) = \frac{p(i|l)p(l)}{p(i)}$
  - *The number of possible sensor readings and geometric contexts is extremely large*
  - *$p(i|l)$ is computed using a model of the robot's sensor behavior, its position l, and the local environment metric map around l.*
  - *Assumptions*
    - *Measurement error can be described by a distribution with a mean*
    - *Non-zero chance for any measurement*



Ultrasound.

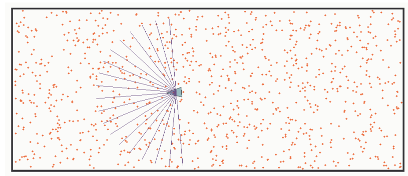


Laser range-finder.

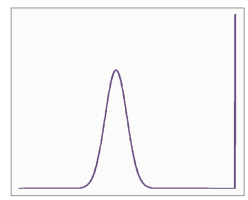
Courtesy of W. Burgard

MCL Sensor Model Issues (2)

- A second issue using the LRF is that for many scans, there will be no sensor data available



- This typically results from wall features being outside the maximum range of the sensor as above, but can also arise when the laser scan is absorbed, multi-path error, etc.



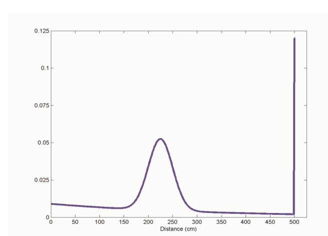
- To address this, the probability of obtaining such a reading is explicitly modeled. The weighting of this is probability is a function of the range and the environment being explored

MCL Sensor Model Issues (3)

- Recall that the conditional probability for the sensor measurement is expressed as the product of the individual probabilities.

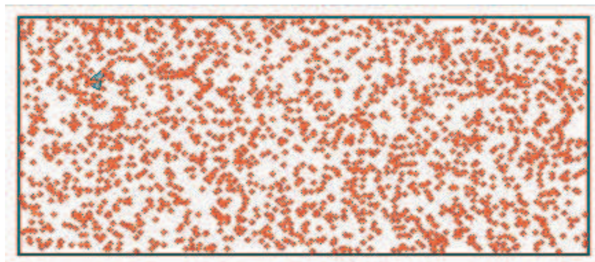
$$p(z_t \mid x_t, m) = \prod_{i=1}^{n} p(z_t^i \mid x_t, m)$$

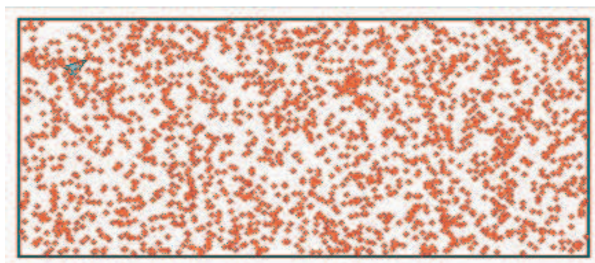
- As a consequence, a single “outlier” can cause the probability to approach zero
- Such errors can readily be caused by errors in our map, furniture, persons/robots moving throughout the environment, etc.



- This is handled by introducing an exponential based probability density into the sensor model for unmodeled “obstacles”

Simple MCL Examples

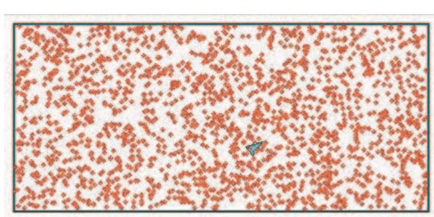




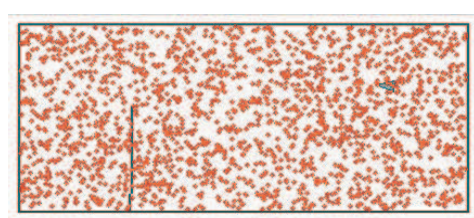
CSE397/497 Intro to Mobile Robotics

When can MCL Fail?

- MCL relies upon difference in the environment to induce corresponding differences in sensor measurements
- Large open areas, long featureless corridors, symmetric environments, etc. can cause MCL to be slow to converge or to converge to the wrong pose
- MCL can exploit even minor differences to obtain a correct pose estimate



Inconsistent Convergence



Consistent Convergence

© JR Spletzer

MCL Extensions

- MCL provides a method for solving the kidnapped robot problem – previously a very difficult problem in mobile robotics
- This is accomplished by adding a small amount of random particles at each time step, and resampling to the original number of particles
- MCL has also been extended to solve the SLAM problem – Simultaneous Localization and Mapping
- This is accomplished by generating a map on the fly, and conditioning your measurements to the portion of the map currently available
- In structured 2D environments, particle filters (with SICK lasers) have effectively solved the SLAM problem

The Kalman Filter

Notation Review

1. Matrices are denoted by a capital letter. In text, they will be bold (e.g. ***A***)
2. Vectors are denoted by a lowercase letter. In text, they will be bold (e.g. ***x***). In Microsoft Equations, they will have an overscore

$$\text{e.g.} \quad \vec{x}_i$$

3. Scalars are lowercase letters without emphasis
4. $\boldsymbol{x}_k^-$ denotes the *a priori* estimate for the state vector ***x*** at time step ***k*** before the measurement update phase
5. $\boldsymbol{x}_k$ denotes the estimate for the state vector ***x*** at time step ***k*** after the measurement update phase

Bayesian Filters (1)

$$p(B \mid A) = \frac{p(A \mid B)\, p(B)}{p(A)}$$

- PFs and Kalman Filters (KF/EKF) are example of Bayesian Filters
- Bayesian filters do not *explicitly* estimate the state
- Instead, they propagate a *posterior* probability density function for the state from which it can be inferred
- In the KF, a gaussian distribution ***P*** is propagated at each timestep with mean μ and variance σ^2. The former is used as the state estimate
- In the PF, a (weighted) particle set corresponds to the posterior from which an estimate for the state can be inferred

CSE397/497 Intro to Mobile Robotics

First & Second Order Statistics/Moments: Expected Value & Variance of the State

- The *expected value* for a random variable X is (*i.e.* the mean) defined as

$$\mu = E(X) = \sum_{i=1}^{n} p_i x_i \quad \text{for discrete } X$$

$$\mu = E(X) = \int_{-\infty}^{\infty} x p_X(x) dx \quad \text{for continuous } X$$

- The *variance* of X about the mean is defined as

$$\sigma^2 = E[(x-\mu)^2] = \sum_{i=1}^{n} p_i (x_i - \mu)^2 \quad \text{for discrete } X$$

$$\sigma^2 = E[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p_X(x) dx \quad \text{for continuous } X$$

CSE397/497 Intro to Mobile Robotics

What is Covariance?

- When X is a vector, the variance is expressed in terms of a *covariance matrix* **C** where

$$c_{ij} = E[(\vec{x}_i - \vec{\mu}_i)^T (\vec{x}_j - \vec{\mu}_j)]$$

- The resulting matrix has the form

$$C = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \cdots & \rho_{2n}\sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1n}\sigma_1\sigma_n & \rho_{2n}\sigma_2\sigma_n & \cdots & \sigma_n^2 \end{bmatrix}$$

where ρ_{ij} corresponds to the degree of correlation between variables X_i and X_j

The Correlation Coefficient

- Correlation is a means to estimate how two functions/series are correlated. For a discrete series, it is defined as

$$\rho = \frac{\sum_i \left[(x_i-\mu_x)(y_i-\mu_y)\right]}{\sqrt{\sum_i (x_i-\mu_x)^2}\sqrt{\sum_i (y_i-\mu_y)^2}} \equiv \frac{C_{xy}}{\sqrt{C_{xx}}\sqrt{C_{yy}}} \equiv \frac{C_{xy}}{\sigma_x \sigma_y}$$

where ρ denotes the *correlation coefficient*

- The denominator normalizes the correlation coefficient such that

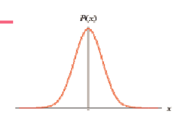
$$\rho \in [-1,1]$$

CSE397/497 Intro to Mobile Robotics

The Gaussian Distribution

- A 1-D Gaussian distribution is defined as

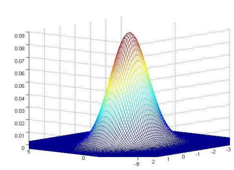
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Gaussian p(x)

- In 2-D (assuming uncorrelated variables) this becomes

$$p(\vec{x}) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left[\frac{(x_1-\mu_1)^2}{2\sigma_1^2}+\frac{(x_2-\mu_2)^2}{2\sigma_2^2}\right]}$$



- In *n* dimensions, it generalizes to

$$p(\vec{x}) = \frac{1}{\sqrt{(2\pi)^n |C|}} e^{-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)}$$

The Normal (Gaussian) distribution is completely parameterized by its first and second moments.

What is a Kalman Filter?

- *Optimal recursive* data fusion algorithm
- Predictor-Corrector style algorithm
- Processes all available sensor measurements in estimating the value of parameters of interest using
  - *Knowledge of system and sensor dynamics*
  - *Statistical models reflecting uncertainty in system noise and sensor dynamics*
  - *Any information regarding initial conditions*

What is a Kalman Filter (cont'd)?

- *Optimal* in the sense that for systems which can be described by a linear model – *e.g.*

$$\vec{x}_{k+1} = Ax_k + Bu_k + w_k$$
$$z_k = Hx_k + v_k$$

and for which the process and measurement noises w_k and v_k are *normally distributed*, the Kalman filter is the provably optimal estimator (estimate has minimum error variance)

- In our case, "process noise" corresponds to uncertainty in the motion model, measurement noise is from uncertainty in the sensing model, ***x*** denotes the state being estimated (the robot pose) and ***z*** the sensor measurements

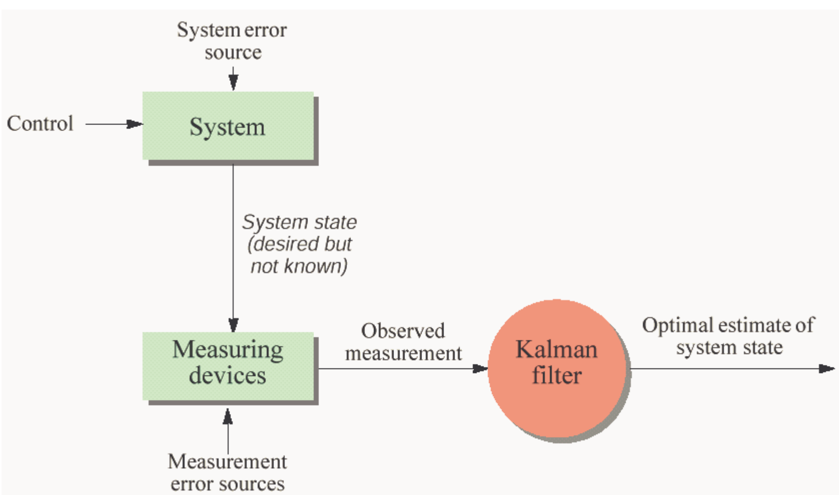
What is a Kalman Filter (cont'd)?

- *Recursive* in the sense that it is "memory-less"
  - *Does not require all previous data to be maintained in memory and reprocessed at each time step*
  - *Propagates first and second order statistics only* (i.e. *mean and variance/covariance)*
- The primary assumption for the KF is that noise in both our motion model and sensor measurements can approximated with unimodal, zero-mean Gaussian noise

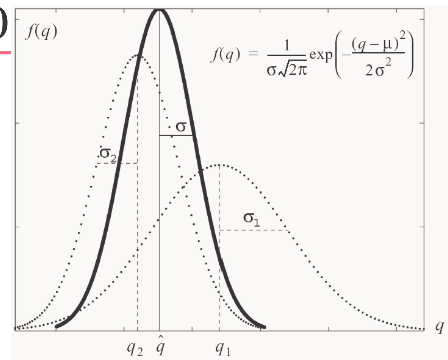
$$p(w) \sim N(0,Q) \qquad p(v) \sim N(0,R)$$

- With this assumption - and the linear process/measurement models – the uncertainty in the state estimate will also be normally distributed

Kalman Filter Localization



Introduction to Kalman Filter (1)



- Two measurements

$$\hat{q}_1 = q_1 \text{ with variance } \sigma_1^2$$
$$\hat{q}_2 = q_2 \text{ with variance } \sigma_2^2$$

- Weighted least-squares

$$S = \sum_{i=1}^{n} w_i(\hat{q} - q_i)^2$$

- Finding minimum error

$$\frac{\partial S}{\partial \hat{q}} = \frac{\partial}{\partial \hat{q}} \sum_{i=1}^{n} w_i(\hat{q} - q_i)^2 = 2\sum_{i=1}^{n} w_i(\hat{q} - q_i) = 0$$

- After some calculation and rearrangements

$$\hat{q} = q_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}(q_2 - q_1)$$

Autonomous Mobile Robots, Chapter 5
5.6.3

Introduction to Kalman Filter (2)

- In Kalman Filter notation

$$\hat{x}_{k+1} = \hat{x}_k + K_{k+1}(z_{k+1} - \hat{x}_k)$$

$$K_{k+1} = \frac{\sigma_k^2}{\sigma_k^2 + \sigma_z^2} \quad ; \quad \sigma_k^2 = \sigma_1^2 \quad ; \quad \sigma_z^2 = \sigma_2^2$$

$$\sigma_{k+1}^2 = \sigma_k^2 - K_{k+1}\sigma_k^2$$

© R. Siegwart, I. Nourbakhsh

Introduction to Kalman Filter (3)

- Dynamic Prediction (robot moving)

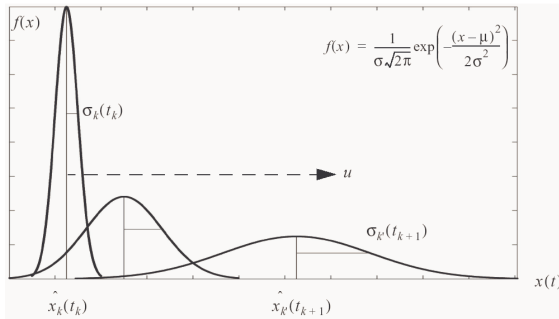
$$\frac{dx}{dt} = u + w$$

u = *velocity*
w = *noise*

- Motion

$$\hat{x}_{k'} = \hat{x}_k + u(t_{k+1} - t_k)$$

$$\sigma_{k'}^2 = \sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]$$



- Combining fusion and dynamic prediction

$$\hat{x}_{k+1} = \hat{x}_{k'} + K_{k+1}(z_{k+1} - \hat{x}_{k'})$$

$$= [\hat{x}_k + u(t_{k+1} - t_k)] + K_{k+1}[z_{k+1} - \hat{x}_k - u(t_{k+1} - t_k)]$$

$$K_{k+1} = \frac{\sigma_{k'}^2}{\sigma_{k'}^2 + \sigma_z^2} = \frac{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k]}{\sigma_k^2 + \sigma_w^2[t_{k+1} - t_k] + \sigma_z^2}$$

The Predictor-Corrector Approach

- In this example, prediction came from using knowledge of the vehicle dynamics to estimate its change in position
- The analogy would be integrating information from the vehicle odometry or to estimate changed in position
- The correction is accomplished through making exteroceptive observations and then fusing this with your current estimate
- This is akin to updating position estimates using landmark information, etc.
- In practice, the prediction rate is typically much higher than the correction

CSE397/497 Intro to Mobile Robotics

The Discrete Kalman Filter (1)

- The Kalman filter addresses the problem of estimating the state $\mathbf{x} \in R^n$ of a discrete-time controlled process governed by the linear difference equation

$$\vec{x}_{k+1} = A\vec{x}_k + B\vec{u}_k + \vec{w}_k$$

and with a measurement $z \in R^m$ that is

$$\vec{z}_k = H\vec{x}_k + \vec{v}_k$$

where $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ represent the process and measurement noise. They are assumed independent, white, and with Gaussian PDFs

$$p(w) \sim N(0,Q) \qquad p(v) \sim N(0,R)$$

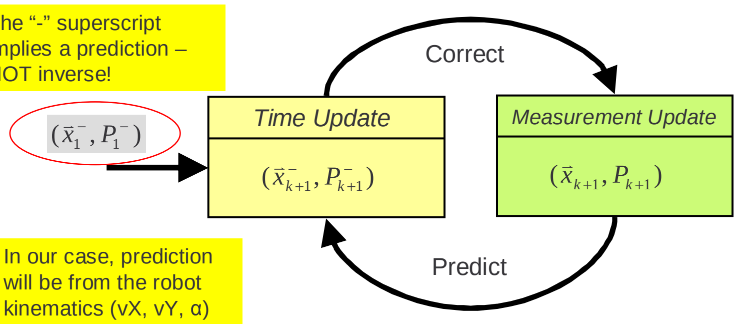
NOTE: The matrices A,B,H,Q & R may be time varying

CSE397/497 Intro to Mobile Robotics

The Discrete Kalman Filter (2)

- At each time step, the KF propagates both a *state estimate* $\boldsymbol{x_k}$ and an estimate for the *error covariance* $\boldsymbol{P_k}$. The latter provides an indication of the uncertainty associated with the state estimate
- As mentioned previously, the KF is a predictor-corrector algorithm. Prediction comes in the *time update* phase, and correction in the *measurement update* phase

The "-" superscript implies a prediction – NOT inverse!



In our case, prediction will be from the robot kinematics (vX, vY, α)

The Time Update Phase

1. Predict the state ahead

$$\vec{x}^-_{k+1} = A\vec{x}_k + B\vec{u}_k$$

2. Project the error covariance ahead

$$P^-_{k+1} = AP_k A^T + Q$$

CSE397/497 Intro to Mobile Robotics

The Measurement Update Phase

1. Compute the Kalman Gain K_k

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

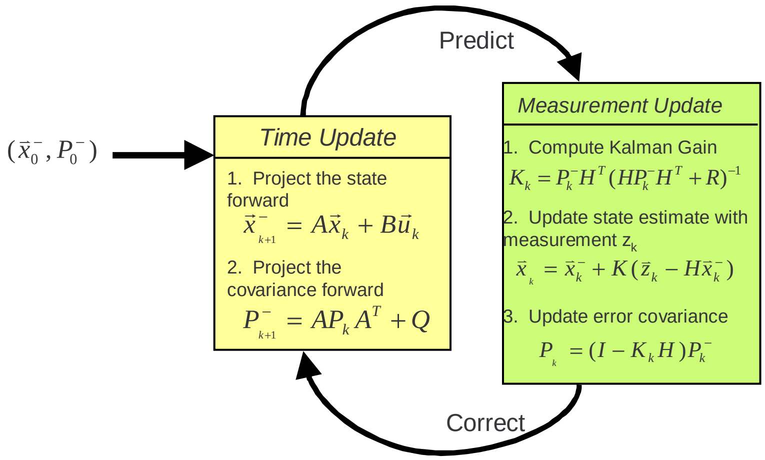
2. Update the estimate based on the new measurement z_k

$$\vec{x}_k = \vec{x}_k^- + K(\vec{z}_k - H\vec{x}_k^-)$$

3. Update the error covariance

$$P_k = (I - K_k H) P_k^-$$

The Discrete Kalman Filter (3)

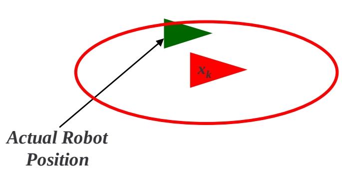


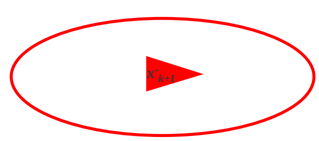
Predictor-Corrector KF Example (1)

1) We have a *covariance matrix P* with mean x_k. x_k is our pose estimate and the *P* is the uncertainty associated with that pose estimate.

2) We predict the next position from our motion model

$$\vec{x}_{k+1}^- = A\vec{x}_k + B\vec{u}_k$$
$$P_{k+1}^- = AP_k A^T + Q$$

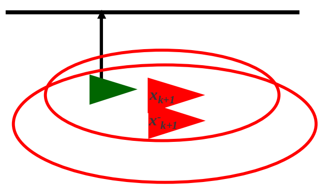




Predictor-Corrector KF Example (2)

3) We take a new measurement in the MU phase…

… and use this to estimate our new position x_k and covariance P_{k+1}



$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

$$\vec{x}_k = \vec{x}_k^- + K(\vec{z}_k - H\vec{x}_k^-)$$

$$P_k = (I - K_k H) P_k^-$$

1-D Example
Estimating a Random Constant

- Suppose we are trying to estimate the value of a 1D constant from corrupted sensor measurements. Our process model is then

$$x_{k+1} = Ax_k + Bu_k + w_k = x_k + w_k$$

$$z_k = Hx_k + v_k = x_k + v_k$$

- The KF equations then are

Variance of our state estimate

Variance of our signal level

Variance of our measurement device

Time Update

$$x_{k+1}^- = x_k$$

$$P_{k+1}^- = P_k + Q$$

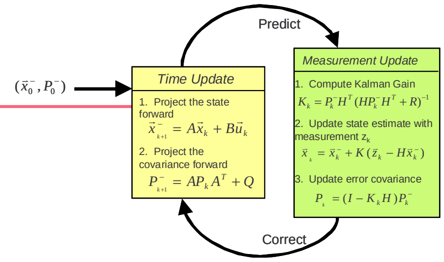
Measurement Update

$$K_k = P_k (P_k + R)^{-1}$$

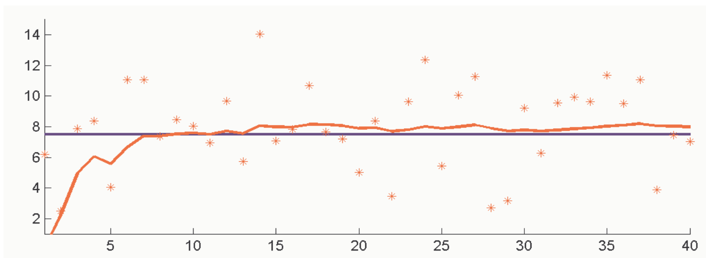
$$x_k = x_k^- + K(z_k - x_k^-)$$

$$P_k = (I - K_k) P_k^-$$

Simulation Results (1)

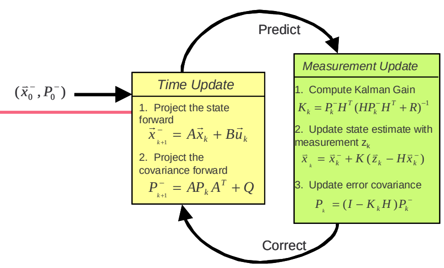


- Let us assume that x^*=7.5, **Q**=0.01, **R**=9
- With perfect knowledge of the process and sensor covariance model, we obtain

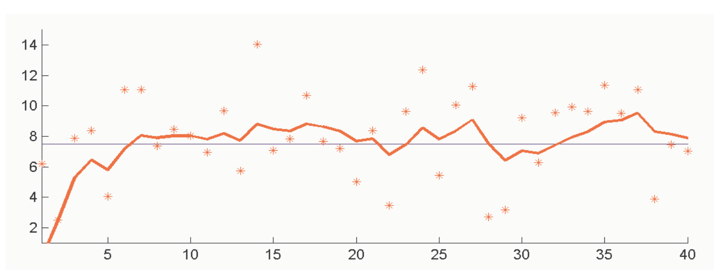


CSE397/497 Intro to Mobile Robotics

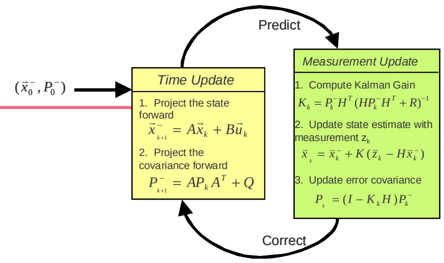
Simulation Results (2)



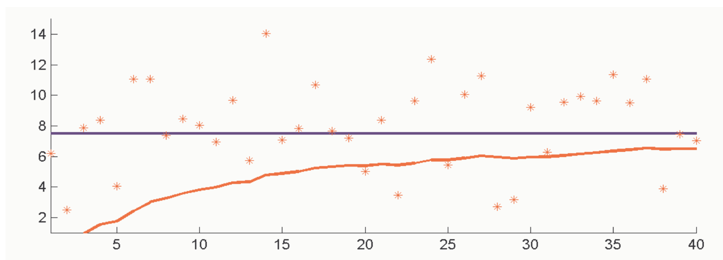
- Let us assume that x^*=7.5, **Q**=0.01, ***R***=9
- Let us further assume that the user believes that the sensor covariance ***R***= 0.09



Simulation Results (3)



- Let us assume that $\boldsymbol{x}^*$=7.5, $\boldsymbol{Q}$=0.01, $\boldsymbol{R}$=9
- Let us further assume that the user believes that the sensor covariance $\boldsymbol{R}$= 900



Kalman Filters vs. Particle Filters

KF	PF
Compact representation	*Memory-intensive representation*
Single state hypothesis	*n hypotheses (1 for each particle)*
Explicitly model Gaussian PDF for state / covariance estimation	*Implicitly Approximates any PDF for state/covariance*
Scales well computationally for higher dimensional representations	*Limited to ~3 dimensions on modern computers*
Diverge in the kidnapped robot problem	*Solves the kidnapped robot problem*
Limited to linear system models	*Works for any system model*
Optimal	*Sub-optimal*

Summary

- The primary limitation of a *PF* is the dimension of the state that can be represented (~3), as computational complexity scales exponentially with the dimension
- This often relegates *PF* to indoor localization problems
- *KFs* can represent much higher dimensional states in real time *O(1000)*
- *Hybrid* filters that integrate *PFs* and *KFs* are not uncommon
- The primary limitation of the *KF* is that it can only be used for linear models, but for these it is *the optimal data fusion algorithm*
- An *Extended Kalman Filter* (EKF) that approximates the *KF* through linearization techniques has greatly expanded *KF* applications and is one of the most widely used algorithms in mobile robotics