Localization

KOM4520 Fundamentals of Robotic Vision

Today's lecture

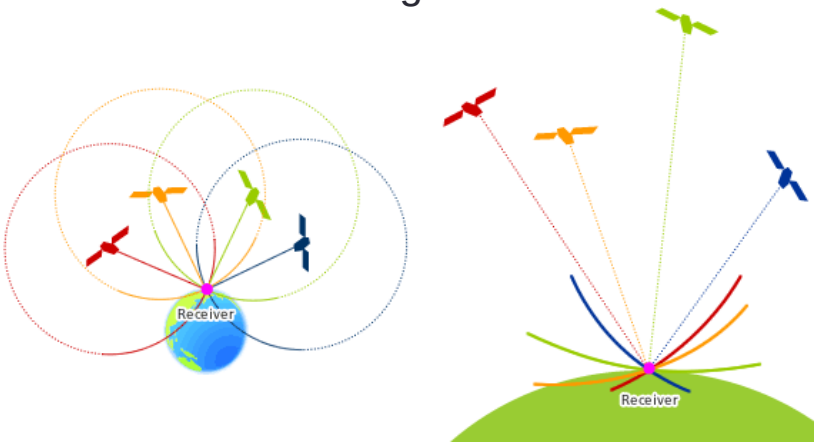
- The Need for Localization
- Primitive Localization Skills
- Deadreckoning of a Vehicle
- Localization as an estimation problem
  - Bayes Filters
  - Deadreckoning using the EKF

The Need for Localization

- During the discussion of map-based navigation we assumed that the robot was aware of its position.
- We will now discuss some of the common techniques used to estimate the location of a robot in the world – a process known as localization.

"in order to get somewhere we need to know where we are"

- Today GPS makes outdoor localization so easy that we often take this capability for granted. Unfortunately GPS is a far from perfect sensor since it relies on very weak radio signals received from distant orbiting satellites.



The Need for Localization

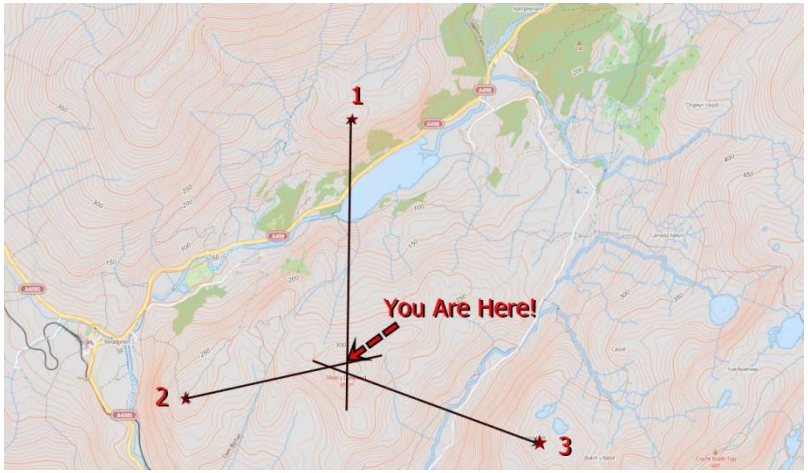
- This means that GPS cannot work where there is no *line of sight* radio reception, for instance indoors.
- GPS signals are also extremely weak and can be easily jammed and this is not acceptable for some applications.
- We will introduce the *classical* navigation principles such as deadreckoning and the use of landmarks on which modern robotic navigation is founded.

  Deadreckoning : the estimation of location based on estimated speed, direction and time of travel with respect to a previous estimate

  The recursive nature of the process, each estimate is based on the previous one, means that errors will accumulate over time and for sea voyages of many-years this approach was quite inadequate.

Primitive Localization Skills

- The earliest mariners there were no maps, or lighthouses or even compasses. They had to create maps as they navigated by incrementally adding new non-manmade features to their maps just beyond the boundaries of what was already known.
- A landmark is a visible feature in the environment whose location is known with respect to some coordinate frame.
- **Re-sectioning** is the estimation of position by measuring the bearing angles to known landmarks. Triangulation is the estimation of position by measuring the bearing angles to the unknown point from each of the landmarks.



- We measure the first bearing from the point (1) that was literally at 0°.
- Another measurement for (2) and a final one for (3) may be required.
- No doubt that we're somewhere within the small triangle created by the intersecting lines.
- These three landmarks are also pretty far apart on the compass, which makes the result more accurate.

www.mudandroutes.com

Primitive Localization Skills

- However this process is critically reliant on correctly associating the observed landmark with the feature on the map. If we mistake one landmark for another significant error in estimated position will be the result.

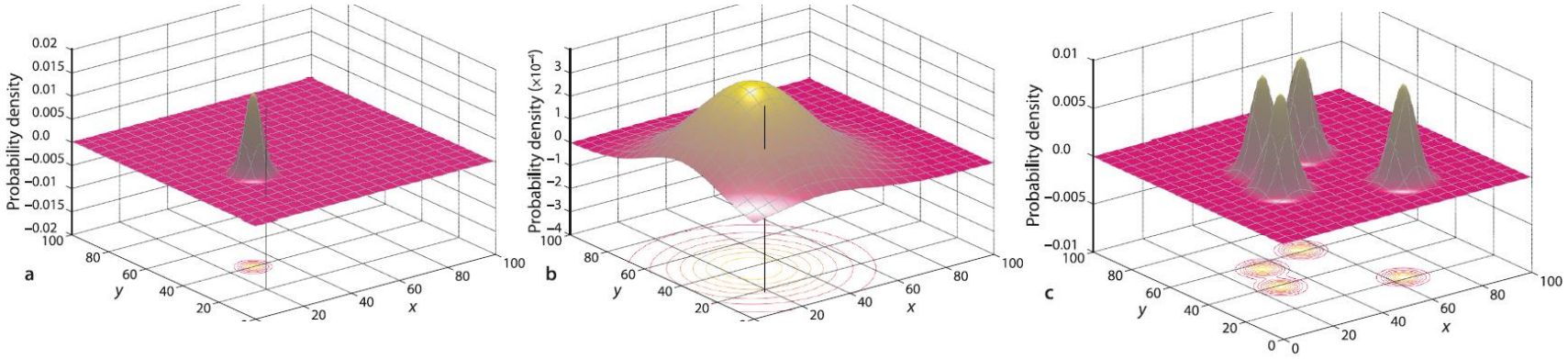
- We would believe we were at **q** instead of **p**.

- Robots operating today in environments without GPS face exactly the same problems
- A robot's estimate of distance traveled will be imperfect and it may have no map, or perhaps an imperfect or incomplete map.
- Additional information from observed features in the world is critical to minimizing a robot's localization error but the possibility of data association error remains.

- If x is the true, but unknown, position of the robot then x' will be our best estimate of that position.
- We also wish to know the uncertainty of the estimate which we can consider in statistical terms as the standard deviation associated with the position.

Uncertainty of the Localization Estimates

- A probability density function (PDF) over all possible positions of the robot can be used



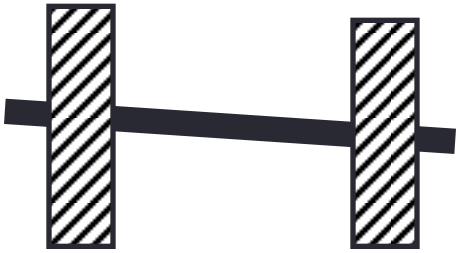
- Likelihood of the vehicle being at that position, sometimes referred to as belief or bel(x).
- **a** The vehicle has low position uncertainty, σ =1; **b** the vehicle has much higher position uncertainty, σ = 20; **c** the vehicle has multiple hypotheses for its position, each σ = 1
- Consider an indoor robot that has observed a door and there are four doors marked on the map.
- The value of a PDF is *not* the probability of being at that location. Consider a small region of the *xy*-plane, the volume under that region of the PDF is the probability of being in that region.

Deadreckoning of a Vehicle

- Deadreckoning is the estimation of a robot's pose based on its estimated speed, direction and time of travel with respect to a previous estimate.
- An odometer is a sensor that measures distance traveled and sometimes also change in heading direction.
- For wheeled vehicles this can be determined by measuring the angular rotation of the wheels.
- These sensors are imperfect.
- There are systematic errors : incorrect wheel radius or gyroscope bias, and random errors such as slip between wheels and the ground.

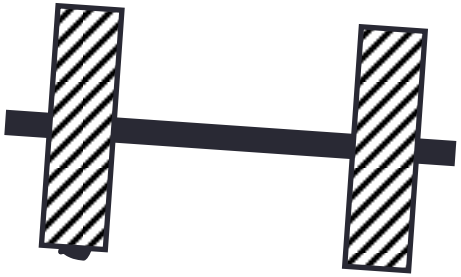
Possible sources of noise

Ideal case



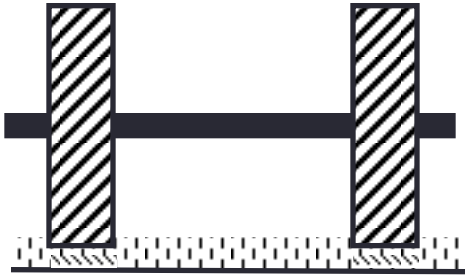
Different wheel diameters

(systematic source of noise)



Bump

(casual source of noise)



Carpet

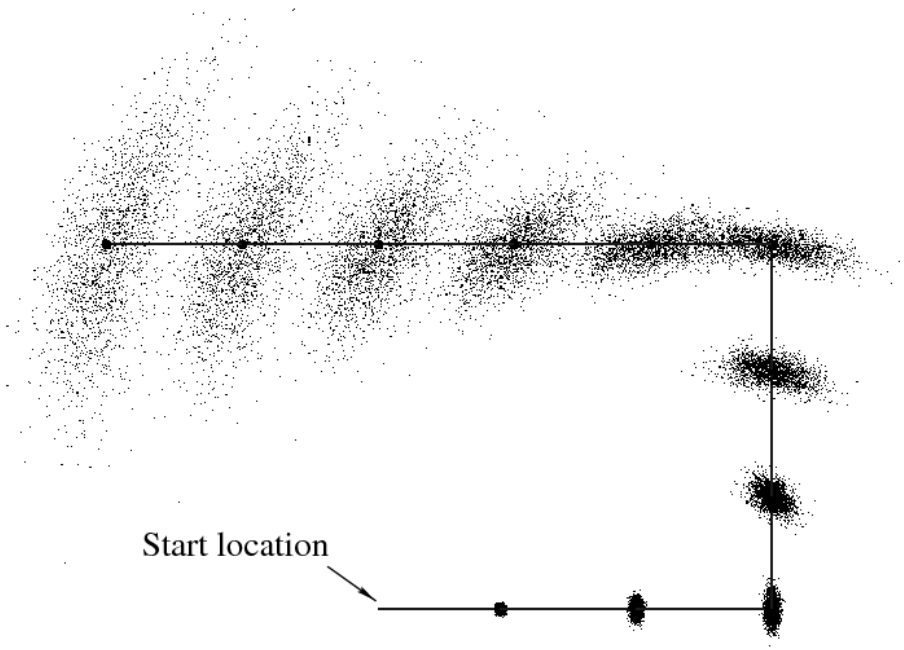
(casual source of noise)

and many more

Thrun Burgard Fox, Probabilistic Robotics, MIT Press 2006

Possible sources of noise

Accumulation of the pose estimation error under the robot motion
(only proprioceptive measurements - e.g. encoders, accelerometers, etc.)



Deadreckoning of a Vehicle

- A vehicle model describes the evolution of the robot's configuration as a function of its control inputs, However for real robots we rarely have access to these control inputs.
- Most robotic platforms have proprietary motion control systems that accept motion commands from the user (speed and direction) and report odometry information

- The first step in estimating the robot's pose is to write a function, f(·), that describes how the vehicle's configuration changes from one time step to the next.
- A discrete-time model for the evolution of configuration based on odometry where $\boldsymbol{\delta}\langle k\rangle = (\delta_d, \delta_\theta)$ is the distance traveled and change in heading over the preceding interval, and k is the time step

$$\boldsymbol{\xi}\langle k\rangle \cong \begin{bmatrix} cos\theta\langle k\rangle & -sin\theta\langle k\rangle & x\langle k\rangle \\ sin\theta\langle k\rangle & cos\theta\langle k\rangle & y\langle k\rangle \\ 0 & 0 & 1 \end{bmatrix}$$ initial pose

We choose to move forward in the vehicle *x*-direction by δ_d, and then rotate by δ_θ giving the new configuration

$$\boldsymbol{\xi}\langle k+1\rangle \cong \begin{bmatrix} cos\theta\langle k\rangle & -sin\theta\langle k\rangle & x\langle k\rangle \\ sin\theta\langle k\rangle & cos\theta\langle k\rangle & y\langle k\rangle \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \delta_d \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\delta_\theta & -sin\delta_\theta & 0 \\ sin\delta_\theta & cos\delta_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cong$$

$$\begin{bmatrix} cos(\theta\langle k\rangle + \delta_\theta) & -sin(\theta\langle k\rangle + \delta_\theta) & x\langle k\rangle + \delta_d cos\theta\langle k\rangle \\ sin(\theta\langle k\rangle + \delta_\theta) & cos(\theta\langle k\rangle + \delta_\theta) & y\langle k\rangle + \delta_d sin\theta\langle k\rangle \\ 0 & 0 & 1 \end{bmatrix}$$

Deadreckoning of a Vehicle

- This gives the new configuration in terms of the previous configuration and the odometry.

$$\boldsymbol{x}\langle k+1\rangle = \begin{bmatrix} x\langle k\rangle + \delta_d cos\theta\langle k\rangle \\ y\langle k\rangle + \delta_d sin\theta\langle k\rangle \\ \theta\langle k\rangle + \delta_\theta \end{bmatrix}$$

- In practice odometry is not perfect and we model the error by imagining a **random number generator** that corrupts the output of a perfect odometer.
- The measured output of the real odometer is the
  perfect but unknown odometry (δ_d, δ_θ) plus
  the output of the random number generator (v_d, v_θ).

Such random errors are often referred to as noise, more specifically as sensor noise.

The robot's configuration at the next time step, including the odometry error is

$$\boldsymbol{x}\langle k+1\rangle = f(\boldsymbol{x}\langle k\rangle, \boldsymbol{\delta}\langle k\rangle, \boldsymbol{v}\langle k\rangle) = \begin{bmatrix} x\langle k\rangle + (\delta_d + v_d) cos\theta\langle k\rangle \\ y\langle k\rangle + (\delta_d + v_d) sin\theta\langle k\rangle \\ \theta\langle k\rangle + \delta_\theta + v_d \end{bmatrix}$$

k is the time step
$\boldsymbol{\delta}\langle k\rangle = (\delta_d, \delta_\theta)^T \in \mathcal{R}^{2x1}$ is the odometry measurement
$\boldsymbol{v}\langle k\rangle = (v_d, v_\theta)^T \in \mathcal{R}^{2x1}$ is the random measurement noise over the preceding interval

- .

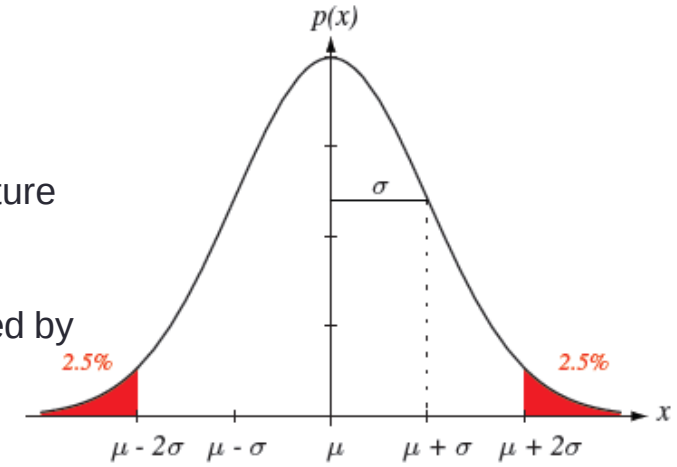
Deadreckoning of a Vehicle

- In the absence of any information to the contrary we model the odometry noise as $\boldsymbol{v} = (v_d, v_\theta)^T \sim N(0, \mathbf{V})$, a zero-mean multivariate Gaussian process with covariance matrix $\mathbf{V}$.
- $\mathbf{V} = \begin{bmatrix} (\sigma_d)^2 & 0 \\ 0 & (\sigma_\theta)^2 \end{bmatrix}$
- This constant matrix is diagonal which means that the errors in distance and heading are independent

---

The Univariate Normal Distribution

They represent many probability distributions in nature quite accurately.
In our case, when patterns can be represented as random variations of an ideal prototype (represented by the mean feature vector)
Everyday examples: height, weight of a population



A univariate normal distribution has roughly 95% of its area in the range $|x - \mu| \leq 2\sigma$, as shown. The peak of the distribution has value $p(\mu) = 1/\sqrt{2\pi}\sigma$. From:

The Univariate Normal Distribution

Definition of univariate normal distribution

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

or

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

Here x is a random variable, feature (not a random vector)

The expected value of x is $\mu = \mathcal{E}[x] = \int_{-\infty}^{\infty} xp(x)dx$

The variance or squared deviation of x is $\sigma^2 = \mathcal{E}[(x-\mu)^2] = \int_{-\infty}^{\infty}(x-\mu)^2 p(x)dx$

We can represent univariate normal distribution with two parameters

$$p(x) = N(\mu, \sigma^2)$$

The Multivariate Normal Distribution

Definition of multivariate normal distribution

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{V}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{V}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Here $\mathbf{x}$ is a random feature (d component)

$\boldsymbol{\mu}$ is the expected vector (mean vector) of $\mathbf{x}$

$\mathbf{V}$ is the dxd covariance matrix

T denotes transpose of $(\mathbf{x} - \boldsymbol{\mu})$

$\mathbf{V}^{-1}$ denotes inverse of covariance matrix

We can represent multivariate normal distribution with two parameters

$$p(\mathbf{x}) = N(\boldsymbol{\mu}, \mathbf{V})$$

Deadreckoning of a Vehicle

- Forming $\mathbf{V}$ is not always easy but we can conduct experiments or make some reasonable engineering assumptions.
- In the examples which follow, we choose $\sigma_d = 2\ cm$ and $\sigma_\theta = 0.5°$ per sample interval

```
S = diag([0.02, 0.5*pi/180].^2);
```

a bicycle object is created

```
veh= Bicycle('covar', V)
```

veh =
Bicycle object
L=1
Superclass: Vehicle
max speed=1, max steer input=0.5, dT=0.1, nhist=0
V=(0.0004, 7.61544e-05)
configuration: x=0, y=0, theta=0

These are default parameters such as the vehicle's length, speed, steering limit and the sample interval which defaults to 0.1 s.

The object provides a method to simulate motion over one time step

```
odo = veh.step(1, 0.3)
```

odo =
0.1108 0.0469
where we have specifi ed a speed of $1\ ms^{-1}$ and a steering angle of 0.3 rad.

With a sample interval of 0.1 the robot's true (but 'unknown') configuration can be seen by

```
>> veh.x'
```

ans =
0.1000 0 0.0309

Given the reported odometry estimation of the configuration after one time step using the eq for $\boldsymbol{x}\langle k+1\rangle$

```
>> veh.f([0 0 0], odo)
```

ans =
0.1106 0.0052 0.0469

Localization as an estimation problem

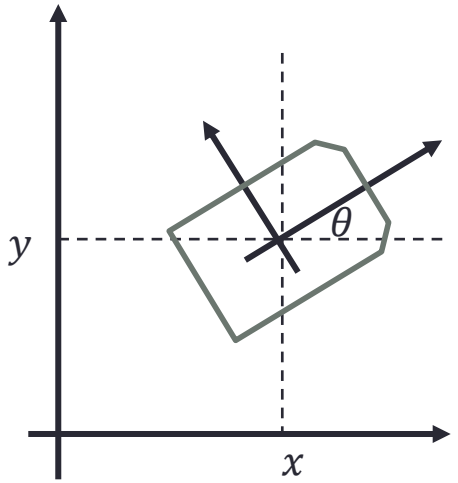
- The robot must infer its pose from available data.
- Motion information: Proprioceptive sensors (e.g. encoders, accelerometers, etc.)
- Environment Measurements: Exteroceptive sensors (e.g. laser, sonar, IR, GPS, camera, RFID, etc.)

- A filtering approach is required to fuse all information.

- How to estimate our new pose given the previous pose and noisy odometry. We want the best estimate of where we are and how certain we are about that. Most common mathematical tool is the Kalman filter.

Localization as an estimation problem

$$x_r = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$



$$x_{r,0:t} = \{x_{r,0}, x_{r,1}, \ldots, x_{r,t}\}$$

Robot poses from time 0 to time t

$$z_{1:t} = \{z_1, z_2, \ldots, z_t\}$$

Robot exteroceptive measurements from time 1 to time t

$$u_{0:t} = \{u_0, u_1, \ldots, u_t\}$$

Motion commands (or proprioceptive measurements) from time 0 to time t

Notes from: Francesco Martinelli, “Mobile Robot Localization” University of Rome, Italy

Localization as an estimation problem

Belief of the robot at time t: probability density function (pdf) describing the information the robot has regarding its pose at time t, based on all available data (exteroceptive measurements and motion commands):

$$bel_t(x_r) = p(x_{r,t} = x_r | z_{1:t}, u_{0:t-1})$$

Prior belief of the robot at time t: pdf before acquiring the last measurement z_t:

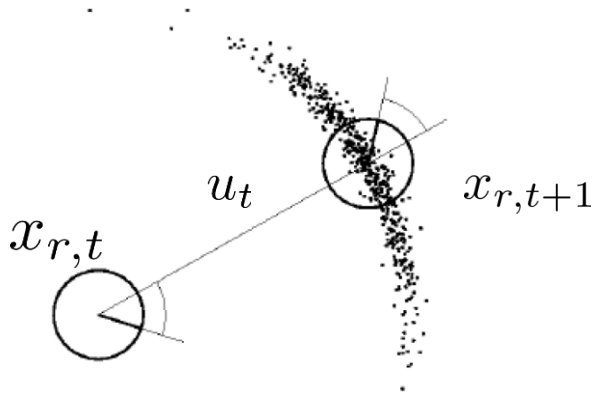
$$\overline{bel}_t(x_r) = p(x_{r,t} = x_r | z_{1:t-1}, u_{0:t-1})$$

Notes from: Francesco Martinelli, “Mobile Robot Localization” University of Rome, Italy

Localization as an estimation problem

The robot motion model is the pdf of the robot pose at time t+1 given the robot pose and the motion action at time t. It takes into account the noise characterizing the proprioceptive sensors:

$$p(x_{r,t+1}|x_{r,t}, u_t)$$



The measurement model describes the probability of observing at time t a given measurement z_t when the robot pose is $x_{r,t}$. It takes into account the noise characterizing the exteroceptive sensors:

$$p(z_t|x_{r,t})$$

Notes from: Francesco Martinelli, “Mobile Robot Localization” University of Rome, Italy

Localization as an estimation problem: Bayes Filter

Prediction

$$\overline{bel}_t(x_r) = \int_{\Omega} p(x_r | x_{r,t-1} = y, u_{t-1}) bel_{t-1}(y) dy$$

Robot pose space

Motion model

Correction or **Measurement Update**

$$\begin{aligned} bel_t(x_r) &= p(x_{r,t} = x_r | z_{1:t}, u_{0:t-1}) \\ &= \eta p(z_t | x_{r,t} = x_r) \overline{bel}_t(x_r) \end{aligned}$$

Normalizing factor

Measurement model

Notes from: Francesco Martinelli, “Mobile Robot Localization” University of Rome, Italy

Localization as an estimation problem: Bayes Filter

The Bayes filter is not a practical algorithm

Gaussian Filters:
Extended Kalman Filter (EKF),
Unscented Kalman Filter (UKF),
Extended Information Filter (EIF)

Non Parametric Filters:
Histogram Filter (HF),
Particle Filter (PF)

Deadreckoning using the EKF

- Kalman filter provides the optimal estimate of the system state, vehicle configuration in this case, assuming that the noise is zero-mean and Gaussian.
- The filter is a recursive algorithm that updates, at each time step, the optimal estimate of the unknown true configuration and the uncertainty associated with that estimate based on the previous estimate and noisy measurement data.
- The Kalman filter is formulated for linear systems but our nonlinear –the extended Kalman filter (EKF) will be used here.

- The vehicle configuration is

$$\boldsymbol{x} = (x_{\mathrm{v}}, y_{\mathrm{v}}, \theta_{\mathrm{v}})^T$$

- And the prediction equations will be

$$\hat{\boldsymbol{x}}^+\langle k+1\rangle = f(\hat{\boldsymbol{x}}\langle k\rangle, \hat{\boldsymbol{u}}\langle k\rangle)$$
$$\hat{\boldsymbol{P}}^+\langle k+1\rangle = \boldsymbol{F}_{\boldsymbol{x}}\hat{\boldsymbol{P}}\langle k\rangle\boldsymbol{F}_{\boldsymbol{x}}^T + \boldsymbol{F}_{\boldsymbol{v}}\hat{\boldsymbol{V}}\boldsymbol{F}_{\boldsymbol{v}}^T$$

describe how the state and covariance evolve with time.

- $\hat{\boldsymbol{x}}^+\langle k+1\rangle$ indicates an estimate of $\boldsymbol{x}$ at time $k+1$ based on information up to, and including, time k.
- $\hat{\boldsymbol{u}}\langle k\rangle$ is the input to the process which in this case is the measured odometry, $\hat{\boldsymbol{u}}\langle k\rangle = \boldsymbol{\delta}\langle k\rangle$
- $\hat{\boldsymbol{P}} \in \mathcal{R}^{3\mathrm{x}3}$ is a covariance matrix representing uncertainty in the estimated vehicle configuration
- $\hat{\boldsymbol{V}}$ is our estimate of the covariance of the odometry noise which in reality we do not know.

Deadreckoning using the EKF

- $\widehat{\mathbf{V}}$ is our estimate of the covariance of the odometry noise which in reality we do not know.
- $\boldsymbol{F_x}$ and $\boldsymbol{F_v}$ are Jacobian matrices which are functions of the current state and odometry.

- robot's configuration at the step $k+1$ was:

$$\boldsymbol{x}\langle k+1\rangle = \boldsymbol{f}(\boldsymbol{x}\langle k\rangle, \boldsymbol{\delta}\langle k\rangle, \boldsymbol{v}\langle k\rangle) = \begin{bmatrix} x\langle k\rangle + (\delta_d + v_d)cos\theta_v\langle k\rangle \\ y\langle k\rangle + (\delta_d + v_d)sin\theta_v\langle k\rangle \\ \theta\langle k\rangle + \delta_\theta + v_\theta \end{bmatrix}$$

$$\boldsymbol{F_x} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}|_{\boldsymbol{v}=0} = \begin{bmatrix} 1 & 0 & -\delta_d \sin\theta_v \\ 0 & 1 & \delta_d \cos\theta_v \\ 0 & 0 & 1 \end{bmatrix}$$

The time step notation is dropped

$$\boldsymbol{v} = (v_d, v_\theta)^T \quad \longrightarrow \quad \boldsymbol{F_v} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{v}}|_{\boldsymbol{v}=0} = \begin{bmatrix} \cos\theta_v & 0 \\ \sin\theta_v & 0 \\ 0 & 1 \end{bmatrix}$$

Deadreckoning using the EKF

All objects of the Vehicle superclass provide methods Fx and Fv to compute these Jacobians, for example

```
veh.Fx( [0,0,0], [0.5, 0.1] )
```

```
ans =
1.0000 0       -0.0499
0      1.0000 0.4975
0      0       1.0000
```

where the first argument is the state at which the Jacobian is computed and the second is the odometry.

```
P0 = diag([0.005, 0.005, 0.001].^2);
```

and we pass this to the constructor for an EKF object

```
ekf = EKF(veh, V, P0);
```

Running the filter for 1000 time steps

```
ekf.run(1000);
```

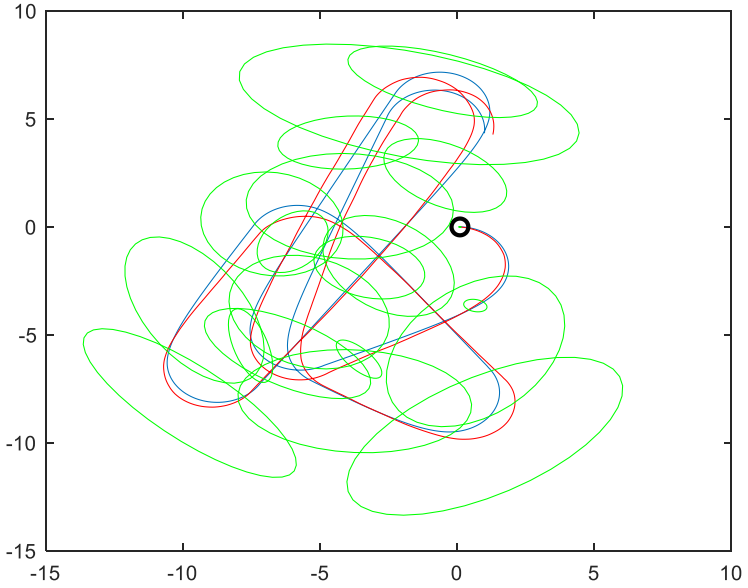
We can plot the true path taken by the vehicle, stored within the Vehicle superclass object, by

```
veh.plot_xy()
```

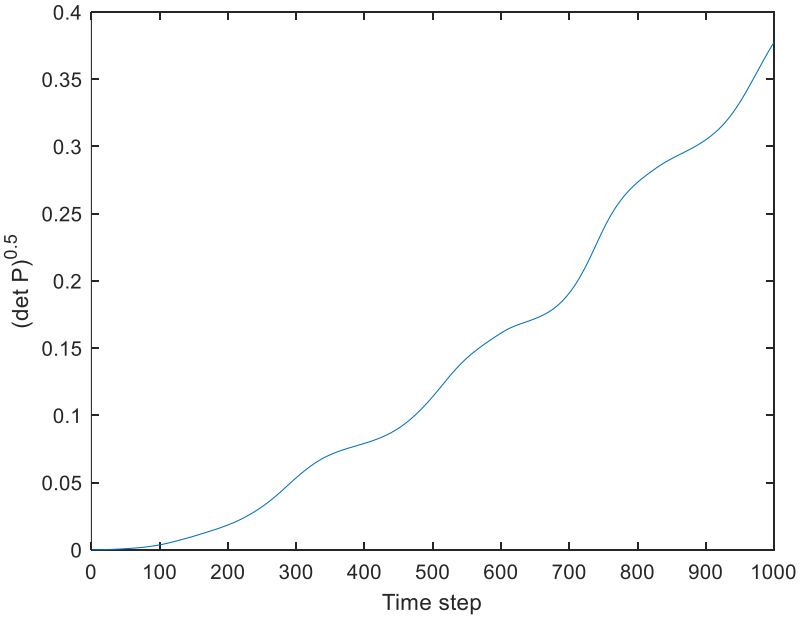
and the filter's estimate of the path stored within the EKF object

```
hold on
ekf.plot_xy('r')
ekf.plot_ellipse('g')
hold off
figure;
ekf.plot_P();
```

Deadreckoning using the EKF



Deadreckoning using the EKF. The true path of the robot, *blue*, and the path estimated from odometry in *red*. 95% confidence ellipses are indicated in *green*. The robot starts at the origin



Overall uncertainty is monotonically increasing