Recursive Bayesian Filtering

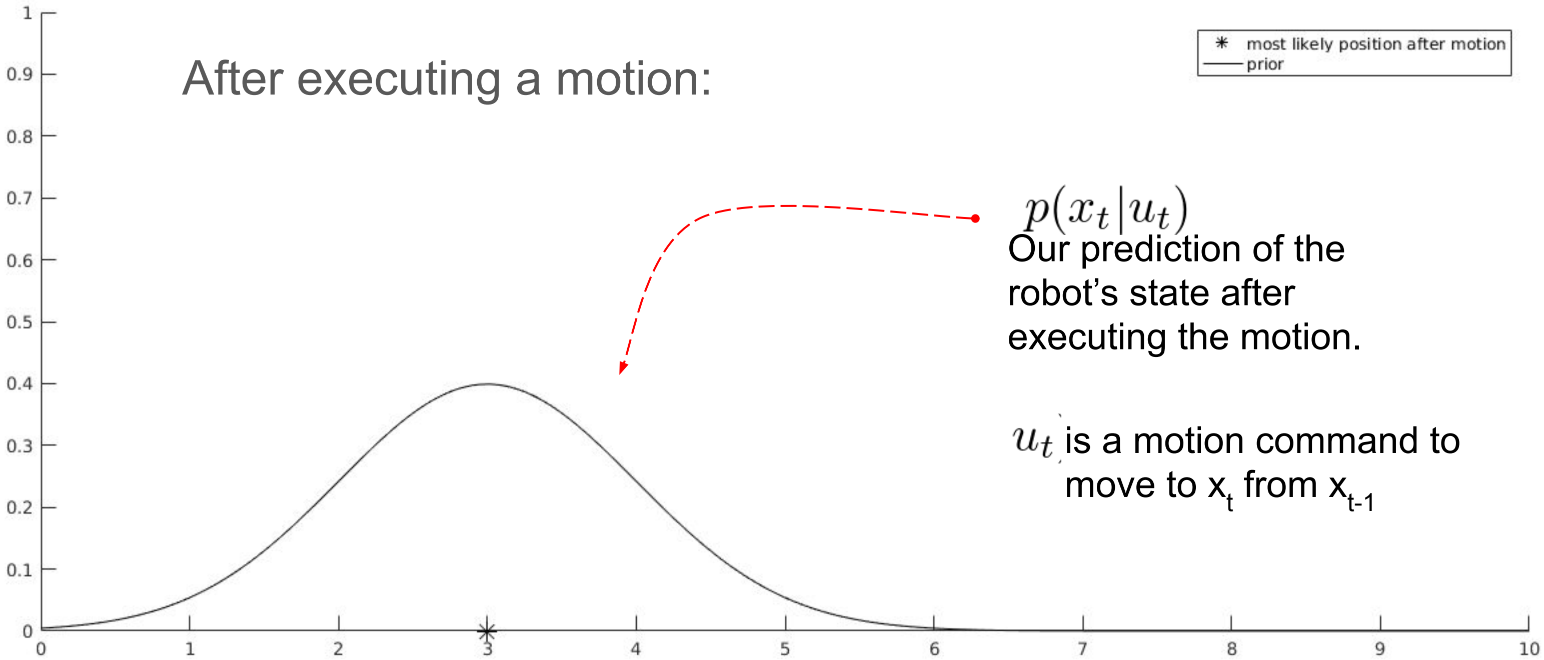
Feras Dayoub

Learning objectives

- Multivariate Normal Distribution.
- Recursive Bayesian filtering.
  - Discrete Bayesian filter
  - Linear Kalman filter.

Recap

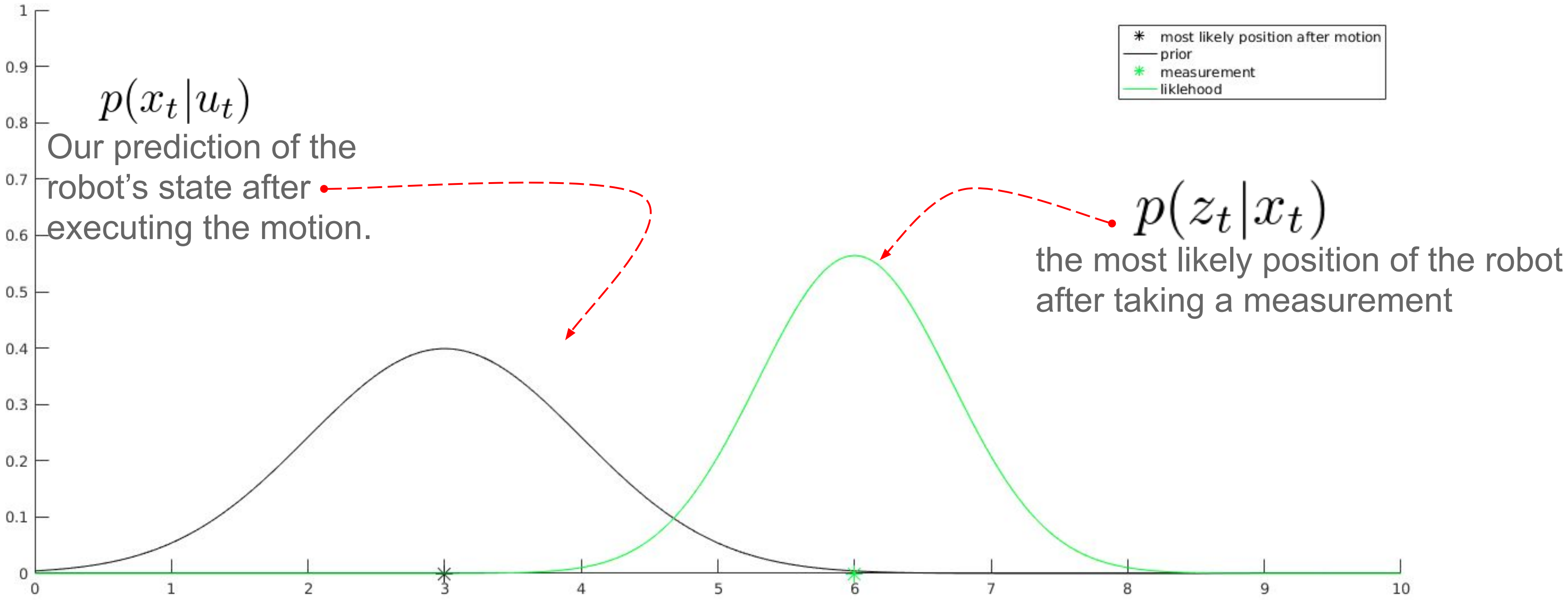
After executing a motion:



$p(x_t|u_t)$
Our prediction of the robot's state after executing the motion.

u_t is a motion command to move to x_t from x_{t-1}

Recap

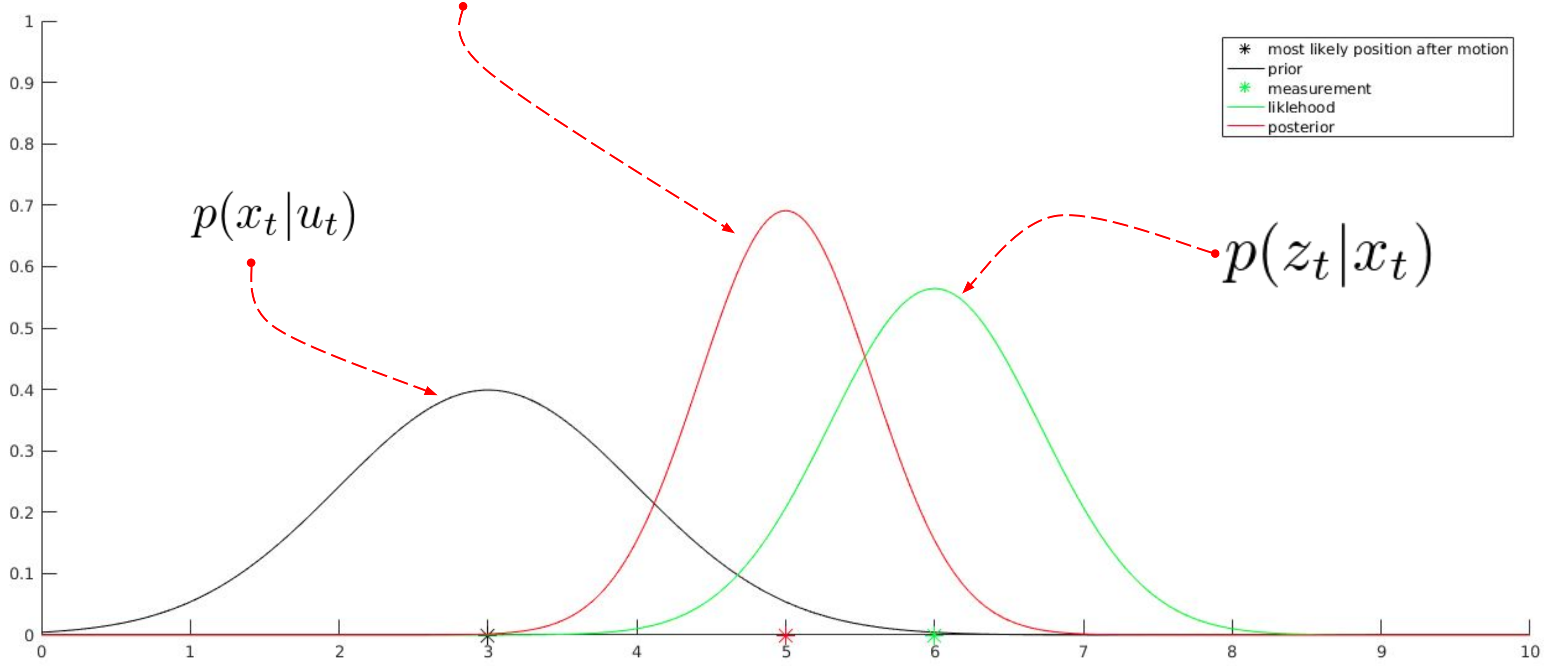


most likely position after motion
prior
measurement
liklehood
$p(x_t|u_t)$
Our prediction of the robot's state after executing the motion.
$p(z_t|x_t)$
the most likely position of the robot after taking a measurement
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6
7
8
9
10

Recap

posterior = likelihood * prior

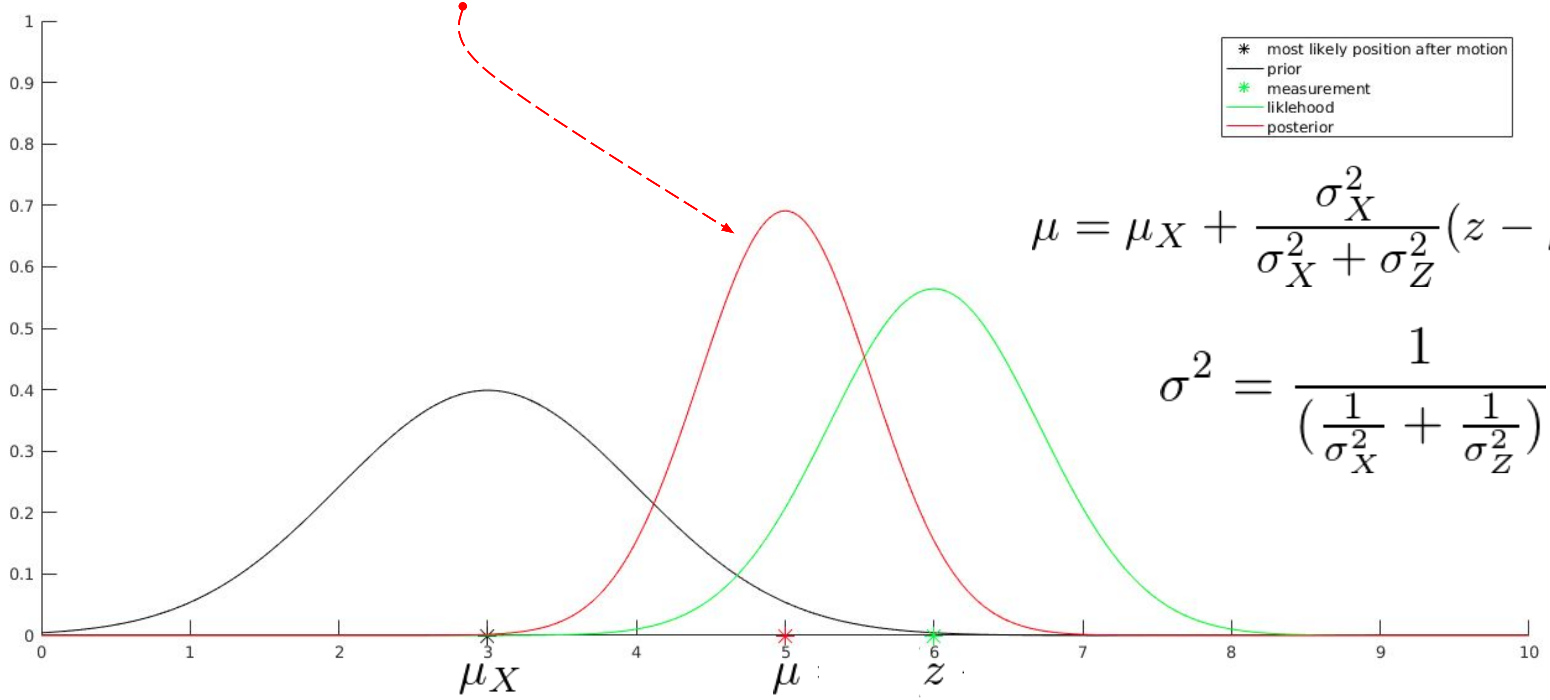
$$p(x_t|z_t,u_t) = \eta \times p(z_t|x_t) \times p(x_t|u_t)$$



Recap

posterior = likelihood * prior

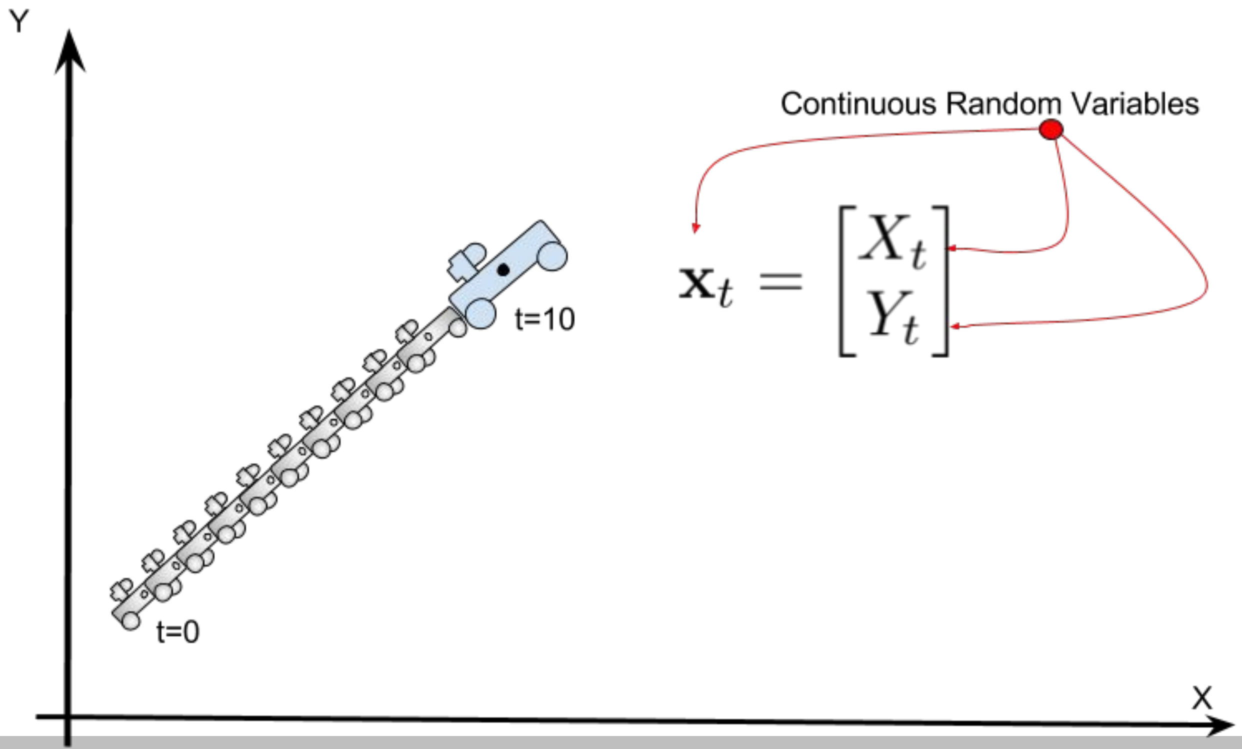
$$\mathrm{p}(\mathrm{x}_t | z_t, u_t) = \eta \times p(z_t | x_t) \times p(x_t | u_t)$$



$$\mu = \mu_X + \frac{\sigma_X^2}{\sigma_X^2 + \sigma_Z^2}(z - \mu_X)$$

$$\sigma^2 = \frac{1}{\left(\frac{1}{\sigma_X^2} + \frac{1}{\sigma_Z^2}\right)}$$

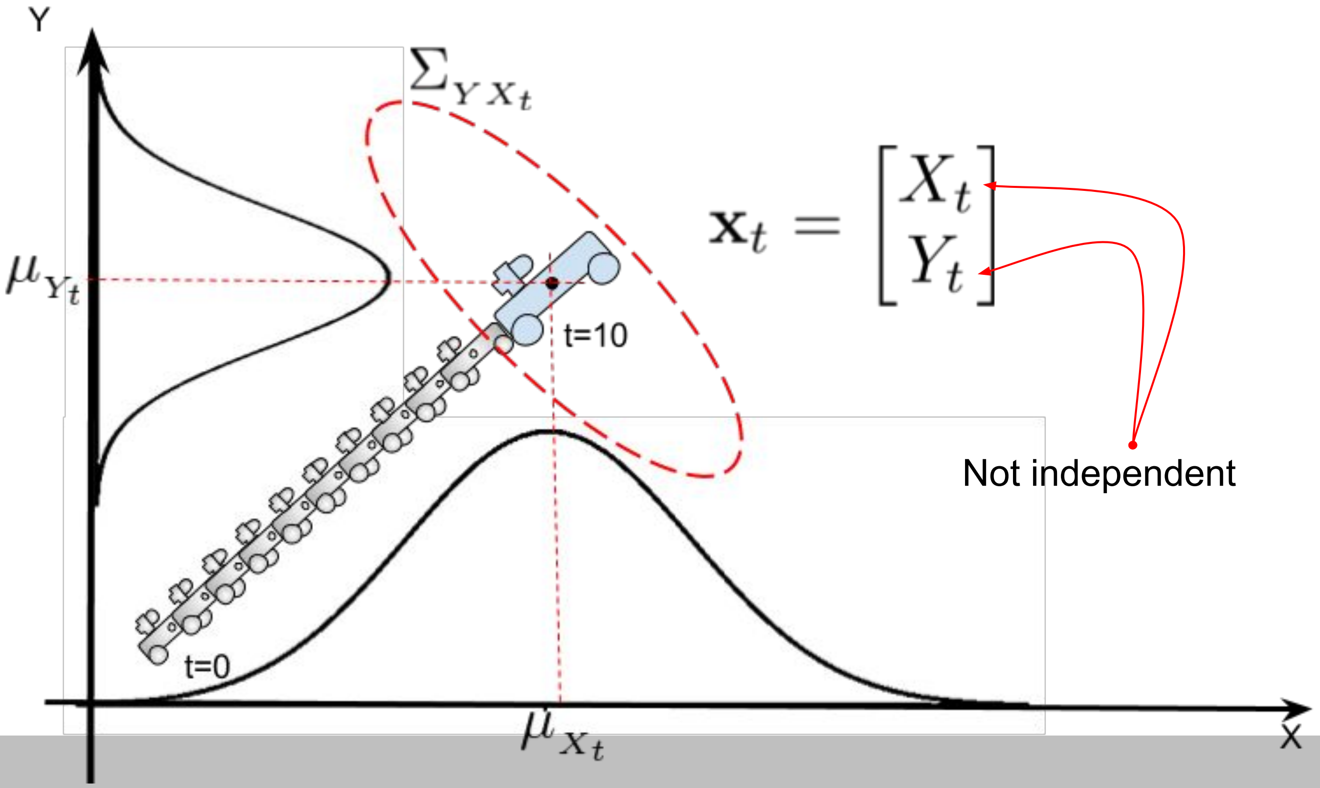
Our robot lives in 2D now!



Continuous Random Variables

$$\mathbf{x}_t = \begin{bmatrix} X_t \\ Y_t \end{bmatrix}$$

Multivariate Gaussian (bivariate for this case)



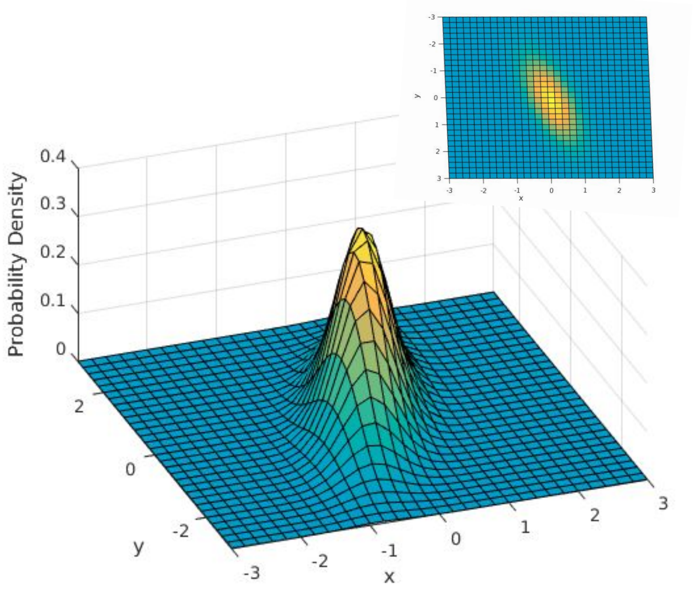
$$\mathbf{x}_t = \begin{bmatrix} X_t \\ Y_t \end{bmatrix}$$

Not independent

Multivariate Gaussian

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

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



Multivariate Gaussian $p(\mathbf{x}_t) \sim \mathcal{N}(\mu_t, \Sigma_t)$

The mean of a multivariate Gaussian is a vector of the means.

$$E[\mathbf{x}_t] = \mu_t = \begin{bmatrix} \mu_{X_t} \\ \mu_{Y_t} \end{bmatrix}$$

The covariance is a matrix. In our bivariate case, it is a 2 by 2 matrix:

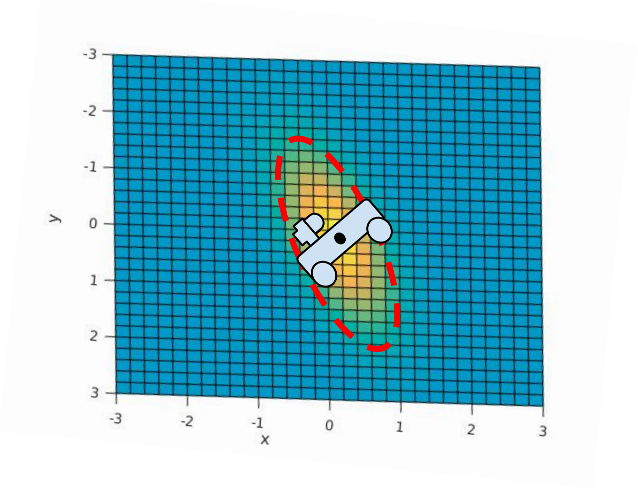
$$\mathrm{Cov}(\mathbf{x_t}) = \Sigma_t = \begin{bmatrix} \sigma^2_{XX} & \sigma^2_{XY} \\ \sigma^2_{XY} & \sigma^2_{YY} \end{bmatrix}$$

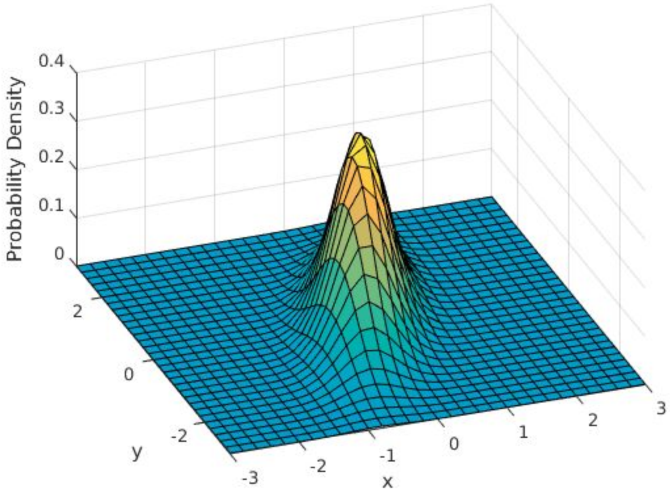
$$\begin{aligned} \sigma^2_{XX} &= Var(X) = E[(X - \mu_X)(X - \mu_X)] \\ \sigma^2_{YY} &= Var(Y) = E[(Y - \mu_Y)(Y - \mu_Y)] \\ \sigma^2_{XY} &= Cov(XY) = E[(X - \mu_X)(Y - \mu_Y)] \end{aligned}$$

$$\begin{aligned} \mathrm{Cov}(a\mathbf{x}) &= a^2\mathrm{Cov}(\mathbf{x}) \\ \mathrm{Cov}(\mathbf{A}\mathbf{x}) &= \mathbf{A}\Sigma\mathbf{A}^T \end{aligned}$$

The covariance matrix

In 2D and 3D, we can use the the covariance matrix to plot a confidence ellipse and ellipsoid (in 3D) that represent the shape of our confidence region:





https://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/

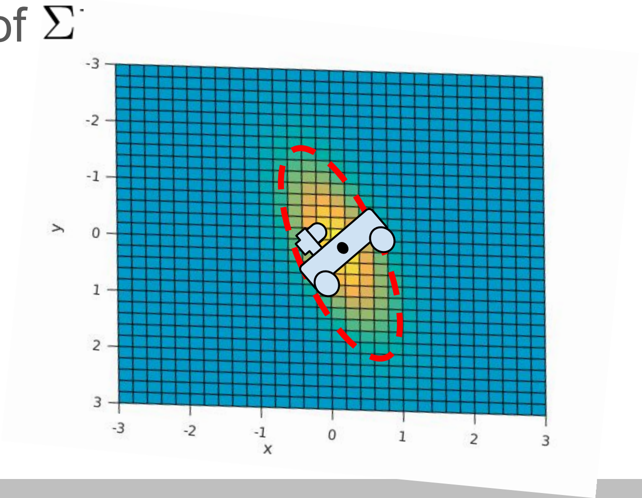
The covariance matrix

The link between the covariance matrix and the ellipse equation:

$$(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)=1$$

The orientation of the ellipse given by the eigenvectors of Σ

The size of the ellipse (area/volume) is given by $\sqrt{|\Sigma|}$



Use the function plot_ellipse(sigma,mu) from Peter's MATLAB toolbox

State estimation

- The state we are trying to estimate in the context of the localization problem is the pose of the robot at time **t**: $\mathbf{x}_t$

- The information available to us are the stream of sensor measurements: $\mathbf{z}_{1:t}$. This can be the range and bearing to landmarks with known position.

- And the control commands or the information about the motion between consecutive time steps: $\mathbf{u}_{1:t}$

Measurement probability

Bayes rule as we saw it at the end of the last lecture:

$$p(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \times p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) p(\mathbf{x}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$$

The current measurement is independent of the previous measurements and the motion given the current state of the robot

$$p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{z}_t | \mathbf{x}_t)$$

Motion probability

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \times p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$$

Using the law of total probability:

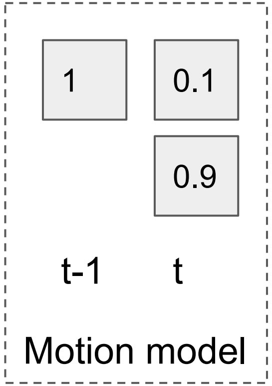
$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})d\mathbf{x}_{t-1}$$

The current state is independent of the past measurements and controls given the previous state

What about this term?

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$$

Law of total probability

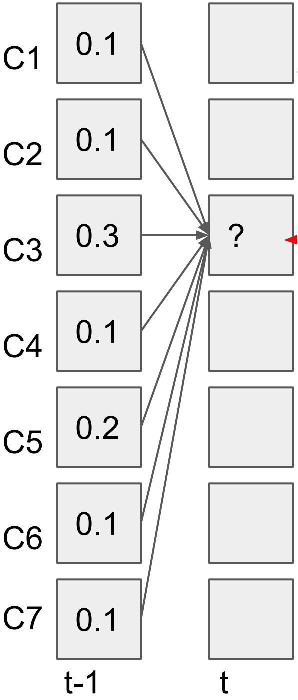


$$P(C1|C1, \text{move command}) = 0.1$$

the probability of staying in the same cell is 10%

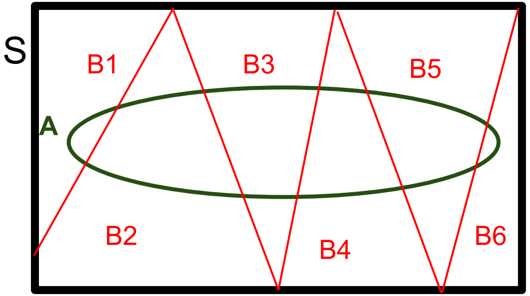
$$P(C2|C1, \text{move command}) = 0.9$$

the probability of moving to the next cell is 90%



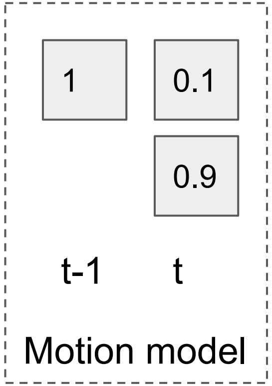
$$P(C_{i_t}|U_{1:t}) = \sum_{j=1}^{7} P(C_{i_t}|C_{j_{t-1}}, U_t) P(C_{j_{t-1}}|U_{1:t-1})$$

$$P(A) = \sum_{i}^{n} P(A|B_i) \times P(B_i)$$



Why integral?

$$P(C_{i_t}|U_{1:t}) = \sum_{j=1}^{7} P(C_{i_t}|C_{j_{t-1}}, U_t)P(C_{j_{t-1}}|U_{1:t-1})$$



$$P(C1|C1, \text{move command}) = 0.1$$

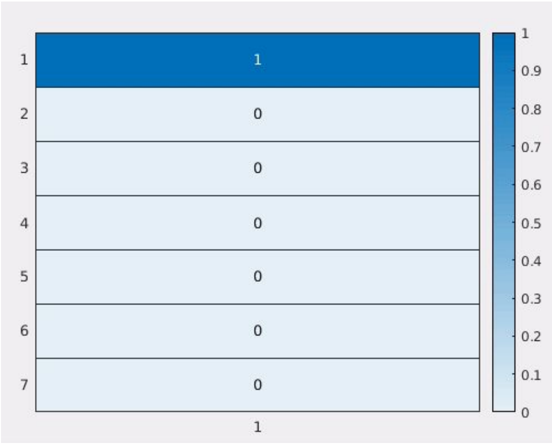
the probability of staying in the same cell is 10%

$$P(C2|C1, \text{move command}) = 0.9$$

the probability of moving to the next cell is 90%

motion

	t=0	t=1	t=2	t=3	t=4
C1	1	.1	.01	0	0
C2	0	.9	.18	.03	0
C3	0	0	.81	.24	.05
C4	0	0	0	.73	.30
C5	0	0	0	0	.65
C6	0	0	0	0	0
C7	0	0	0	0	0



Recursive Bayesian filtering

$$\underbrace{p(\mathbf{x}_t|\mathbf{z}_{1:t},\mathbf{u}_{1:t})}_{bel(\mathbf{x}_t)} = \eta \times p(\mathbf{z}_t|\mathbf{x}_t) \underbrace{\int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1})d\mathbf{x}_{t-1}}_{\bar{bel}(\mathbf{x}_t)}$$

The measurement probability

The motion probability

The distribution of the robot pose from the previous step

Discrete Recursive Bayesian Filter

For all i **do**

$$\bar{bel}(C_{i_t}) = \sum_j P(C_{i_t} | C_{j_{t-1}}, U_t)\, bel(C_{i_{t-1}})$$

$$bel(C_{i_t}) = \eta P(\mathbf{z}_t | C_{i_t}) . \bar{bel}(C_{i_t})$$

return

$$bel(C_{i_t}) = P(C_{i_t} | Z_{1:t}, U_{1:t})$$

Kalman filter

The Kalman filter is a Bayesian filter where the motion and the measurement probability distributions are Gaussians (as we saw in the previous lecture) and the state dynamics is linear.

State transition model: $\mathbf{x}_t = \mathbf{A}\mathbf{x}_{\mathbf{t-1}} + \mathbf{B}\mathbf{u}_t + \mathbf{v}_t$

Measurement model: $\mathbf{z}_t = \mathbf{H}\mathbf{x}_{\mathbf{t}} + \mathbf{w}_t$

Process noise (zero mean Gaussian)

Measurement noise (zero mean Gaussian)

These are random variables with multivariate Gaussians

(Toy example) point moving in 2D plane with constant velocity

$$\mathbf{x}_t = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \sim \mathcal{N}(\mu_{\mathbf{x_t}}, \mathbf{\Sigma}_{\mathbf{x_t}})$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We want to estimate the position and the speed.
But we only observe the position.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{z}_t = \begin{bmatrix} x \\ y \end{bmatrix}$$

$\mathbf{u}$?

(Toy example) The component of the motion model

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{\mathbf{t-1}} + \mathbf{B}\mathbf{u}_t + \mathbf{v}_t$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Describes how the state evolves between time steps without control or noise.

$$\mathbf{B} = \begin{bmatrix} \frac{(\delta t)^2}{2} & 0 \\ 0 & \frac{(\delta t)^2}{2} \\ \delta t & 0 \\ 0 & \delta t \end{bmatrix}$$

Describes how the control u change the the state between time steps. Our example consider a constant velocity therefore the control input (i.e the acceleration) is zero.

$$\mathbf{v}_t \sim \mathcal{N}(0, \mathbf{R})$$

Random variable to represent the process noise which is assumed to be Gaussian with mean 0 and covariance **R**

$$\mathcal{N}(\mu_{x_{t-1}}, \mathbf{\Sigma}_{t-1})$$

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{t:1}) = \eta \times p(\mathbf{z}_t|\mathbf{x}_t) \boxed{\int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)} p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) d\mathbf{x}_{t-1}$$

$$\mathbf{x}_t = \underbrace{\mathbf{A}\mathbf{x_{t-1}} + \mathbf{B}\mathbf{u}_t + \mathbf{v}_t}$$

For given values of $\mathbf{x_{t-1}}$ and $\mathbf{u}_t$

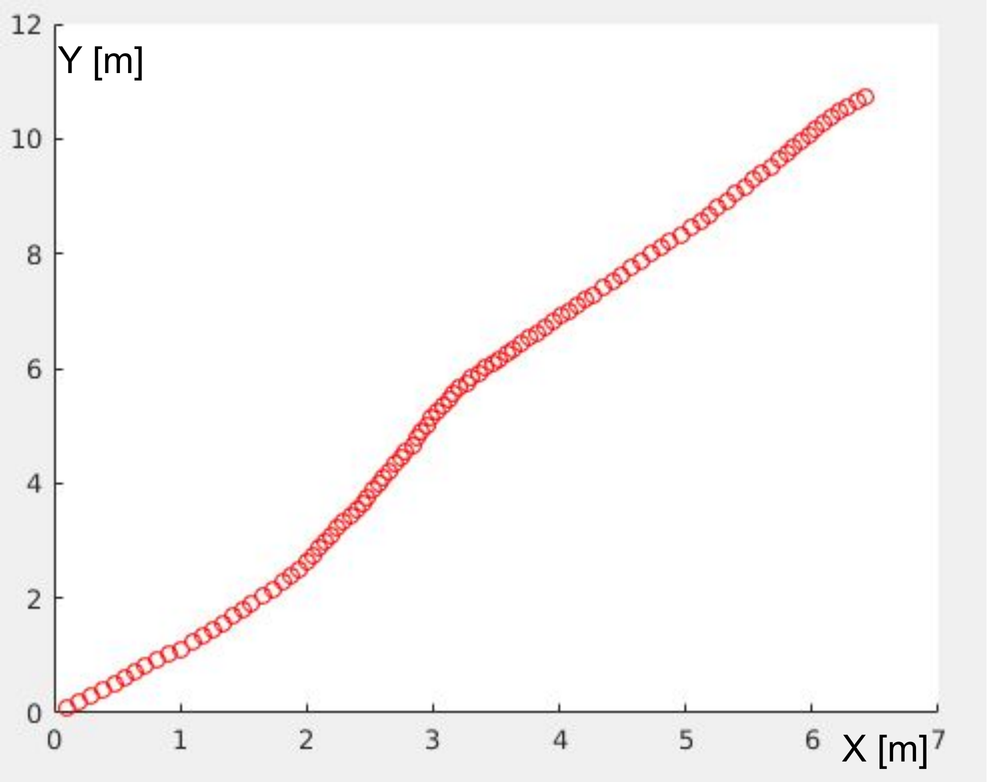
$$\mathcal{N}(\mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u_t}, \mathbf{R})$$

$$\mathcal{N}(\mathbf{A}\mu_{x_{t-1}} + \mathbf{B}\mathbf{u_t}, \underbrace{\mathbf{A}\mathbf{\Sigma}_{t-1}\mathbf{A}^T + \mathbf{R}})$$

Our uncertainty increases with each time step

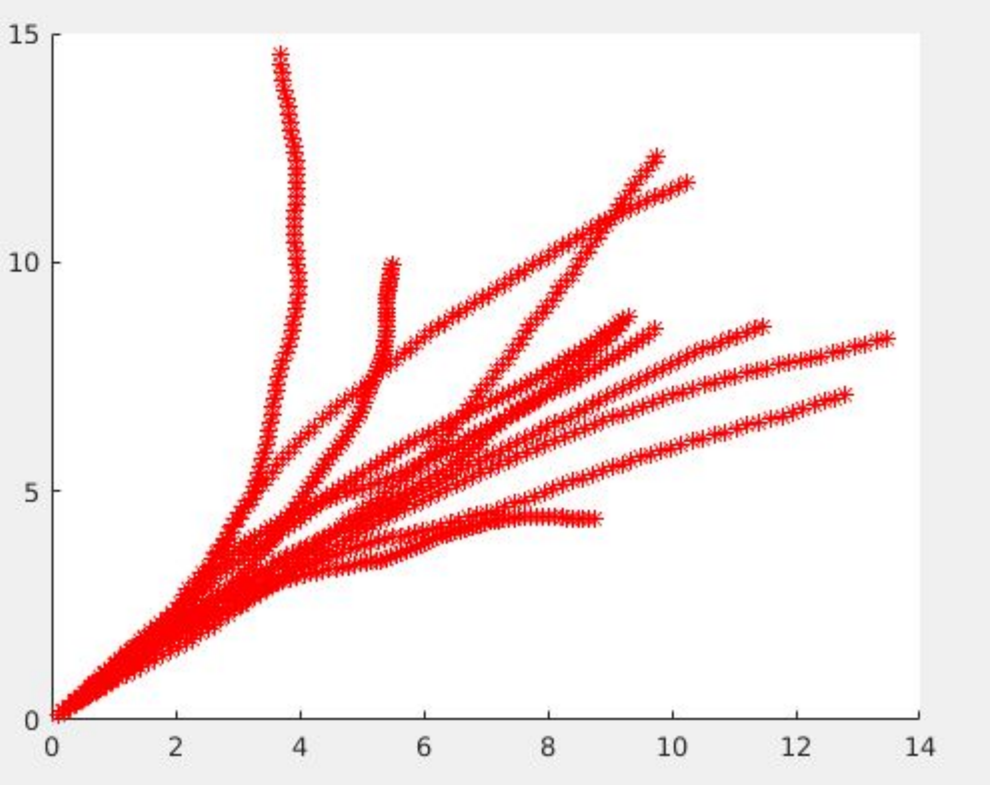
(Toy example) point moving in 2D plane with constant velocity

```
update_rate =5; %5 Hz
dT = 1/update_rate ; % time delta
run_time = 30; %  seconds
nSteps  = run_time * update_rate;

A = [1 0 dT 0;
      0 1 0 dT;
      0 0 1 0;
      0 0 0 1];
% The process noise in the syestm
sigmaV = 0.01;
R = [0.001 0 0 0;0 0.001 0 0;
      0 0 (sigmaV)^2 0 ;0 0 0 (sigmaV)^2];

figure(1)
hold on
% our initial position
% the point starts at [0 0] and move with vx = vy = 0.5 m/s
initX = [0 0 0.5 0.5]';
% lets generate some "real" data
x_true = zeros(4,nSteps);
x_true(:,1) = initX;
for i = 2:nSteps
   v = mvnrnd([0 0 0 0],R,1)';
   x_true(:,i) = A*x_true(:,i-1) + v;
end
scatter(x_true(1,:),x_true(2,:),'ro');
```



(Toy example) point moving in 2D plane with constant velocity

Due to the noise in the system, the trajectory of the point is not deterministic.



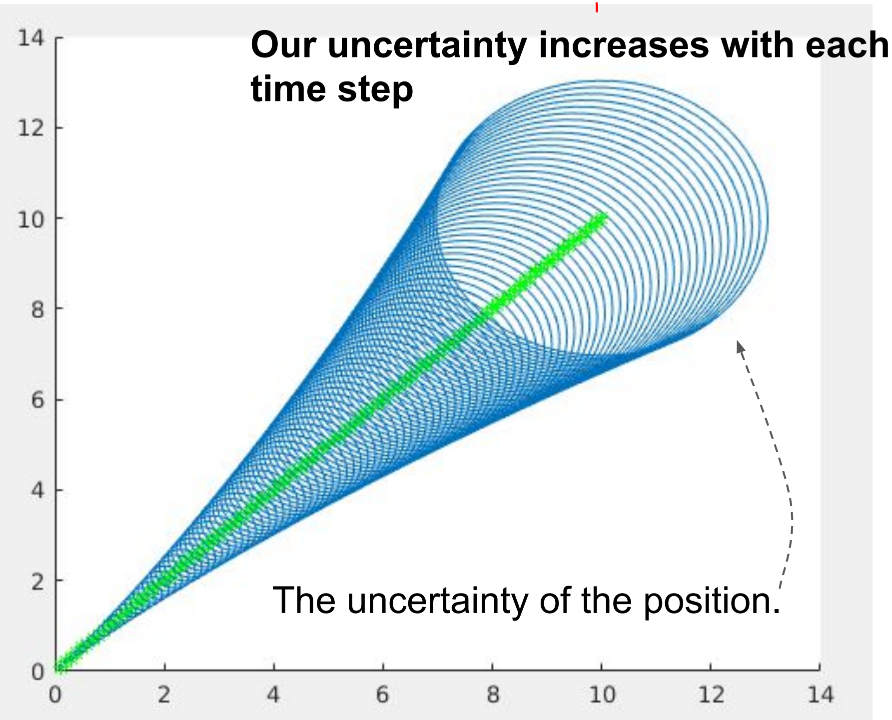
(Toy example) point moving in 2D plane with constant velocity

$$\mathcal{N}(\mathbf{A}\mu_{x_{t-1}} + \mathbf{B}\mathbf{u_t}, \underbrace{\mathbf{A}\Sigma_{t-1}\mathbf{A}^T + \mathbf{R}})$$

```
figure(1)
hold on
% let's represent our uncertainty
SigmaX = [0.001 0 0 0;0 0.001 0 0; 0 0 0.01 0;0 0 0 0.01];
muX = [0 0 1 1]';
for i = 1:nSteps
muX = A*muX;
SigmaX = A*SigmaX*A' + R;
scatter(muX(1),muX(2),'g*');
plot_cov(muX,SigmaX,1)
end
```

very certain initial position

Our uncertainty increases with each time step



The uncertainty of the position.

(Toy example) The component of the measurement model

In our example, at each time step we receive a GPS measurement of our position.

These measurements are noisy so we model them as follows:

$$\mathbf{z}_t = \mathbf{H}\mathbf{x_t} + \mathbf{w}_t$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Describes how to map the state to the measurement.

$$\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q})$$

Random variable to represent the measurements noise which is assumed to be Gaussian with zero mean and covariance **Q**

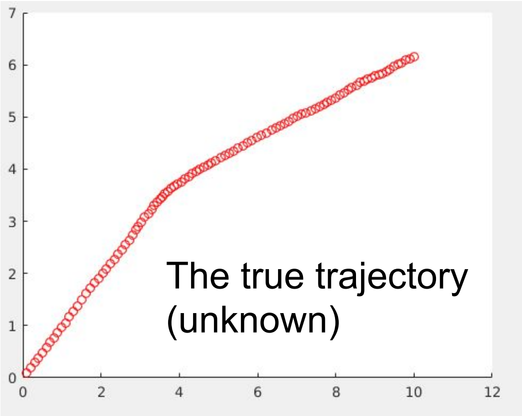
$$p(\mathbf{x}_t|\mathbf{u}_{1:t},\mathbf{z}_{t:1}) = \eta \times \boxed{p(\mathbf{z}_t|\mathbf{x}_t)} \overbrace{\int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1})d\mathbf{x}_{t-1}}^{\mathcal{N}(\mathbf{A}\mu_{x_{t-1}} + \mathbf{B}\mathbf{u_t},\, \mathbf{A}\boldsymbol{\Sigma}_{t-1}\mathbf{A}^T + \mathbf{R})}$$

$$\mathbf{z}_t = \mathbf{H}\mathbf{x_t} + \mathbf{w}_t$$

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{H}\mathbf{x}_t, \mathbf{Q})$$

When we get a measurement, we can reason about the true state of the system.

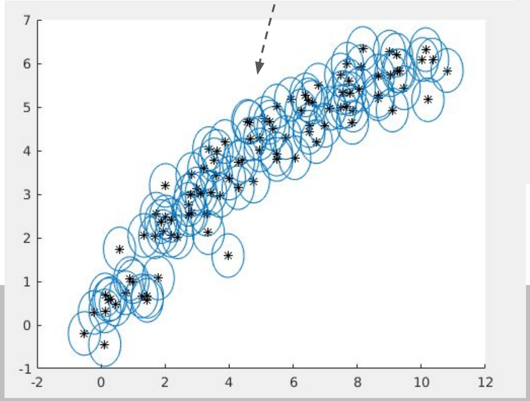
(Toy example) point moving in 2D plane with constant velocity

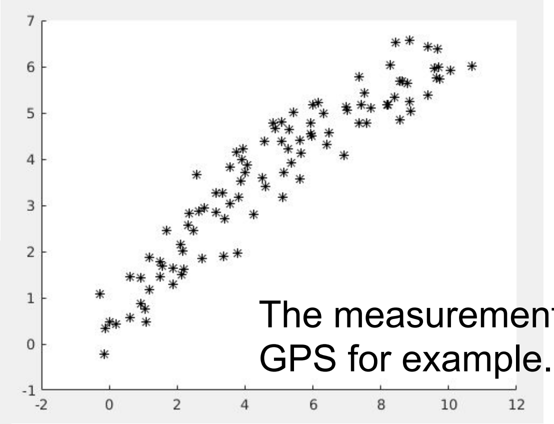


$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{H}\mathbf{x}_t, \mathbf{Q})$$

When we get a measurement, we can reason about the true state of the system.

```
% let's generate some measurements
H = [1 0 0 0;0 1 0 0];
% measurement noise
sigmaW= 0.5;
Q = [(sigmaW)^2 0;0 (sigmaW)^2];
sensor = [];
for i = 1:nSteps
w = mvnrnd([0 0 ],Q,1)';
z = H*x_true(:,i) + w;
sensor = [sensor,z];
scatter(z(1),z(2),'k*');
plot_cov(z,Q,1)
end
```





The Kalman filter's steps

$$p(\mathbf{x}_t|\mathbf{u}_{1:t},\mathbf{z}_{t:1}) = \eta \times \underbrace{p(\mathbf{z}_t|\mathbf{x}_t)}_{\text{sense}} \underbrace{\int p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t)p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1},\mathbf{u}_{1:t-1})d\mathbf{x}_{t-1}}_{\text{motion}}$$

$$\mathcal{N}(\mathbf{H}\mathbf{x}_t,\mathbf{Q}) \quad \mathcal{N}(\mathbf{A}\mu_{x_{t-1}} + \mathbf{B}\mathbf{u_t}, \mathbf{A}\Sigma_{t-1}\mathbf{A}^T + \mathbf{R})$$

What are the new mean and covariance after performing the two steps?

The Kalman filter's steps

$$p(\mathbf{x}_t|\mathbf{u}_{1:t},\mathbf{z}_{t:1}) = \eta \times \mathcal{N}(\underbrace{\mathbf{H}\mathbf{x}_t, \mathbf{Q}}_{\text{sense}}) \times \mathcal{N}(\underbrace{\mathbf{A}\mu_{x_{t-1}} + \mathbf{B}\mathbf{u}_t}_{\bar{\mu}_t}, \underbrace{\mathbf{A}\Sigma_{t-1}\mathbf{A}^T + \mathbf{R}}_{\bar{\Sigma}_t})$$

motion

$$p(\mathbf{x}_t|\mathbf{u}_{1:t},\mathbf{z}_{t:1}) = \mathcal{N}(\mu_t, \Sigma_t) \begin{cases} \mu_t = \bar{\mu}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}\bar{\mu}_t) \\ \Sigma_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\bar{\Sigma}_t \end{cases}$$

Kalman gain

The Kalman Gain

The real measurement
we get from the sensor

$$p(\mathbf{x}_t | \mathbf{u}_{1:t}, \mathbf{z}_{t:1}) = \mathcal{N}(\mu_t, \mathbf{\Sigma}_t) \left\{ \begin{array}{l} \mu_t = \bar{\mu}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}\bar{\mu}_t) \\ \mathbf{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\bar{\mathbf{\Sigma}}_t \end{array} \right.$$

$$\mathbf{K}_t = \bar{\mathbf{\Sigma}}_t\mathbf{H}^T(\mathbf{H}_t\bar{\mathbf{\Sigma}}_t\mathbf{H}^T + \mathbf{Q})^{-1}$$

The Kalman filter steps

Prediction:

$$\bar{\mu}_t = \mathbf{A}\mu_{x_{t-1}} + \mathbf{B}\mathbf{u}_t$$

$$\bar{\Sigma}_t = \mathbf{A}\Sigma_{t-1}\mathbf{A}^T + \mathbf{R}$$

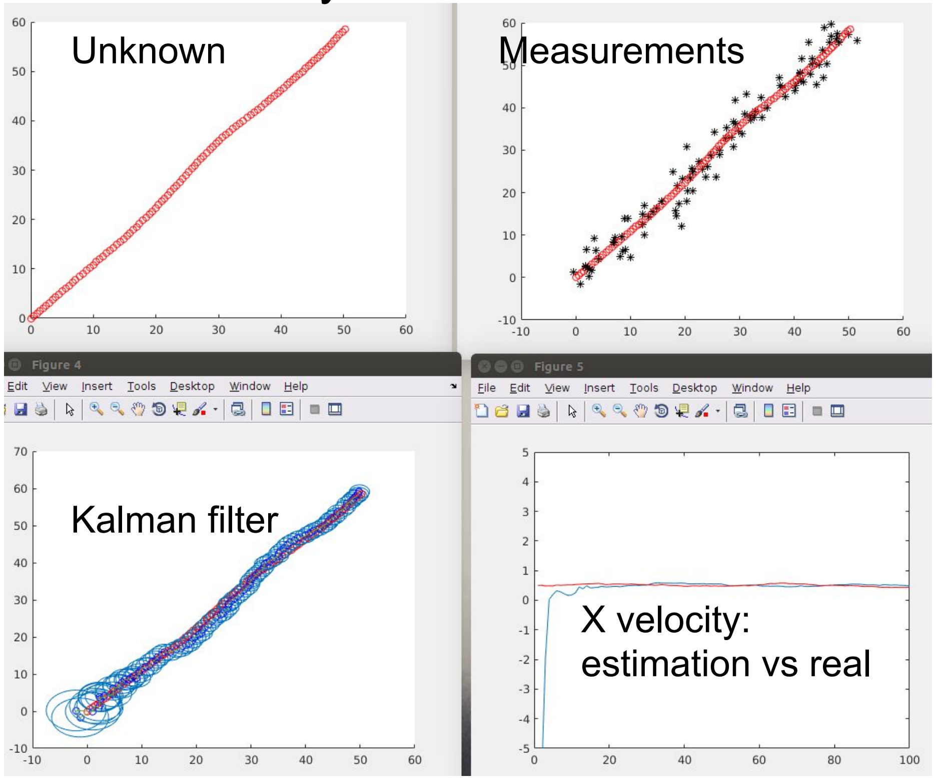
Update/Correction:

$$\mu_t = \bar{\mu}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}\bar{\mu}_t)$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\bar{\boldsymbol{\Sigma}}_t$$

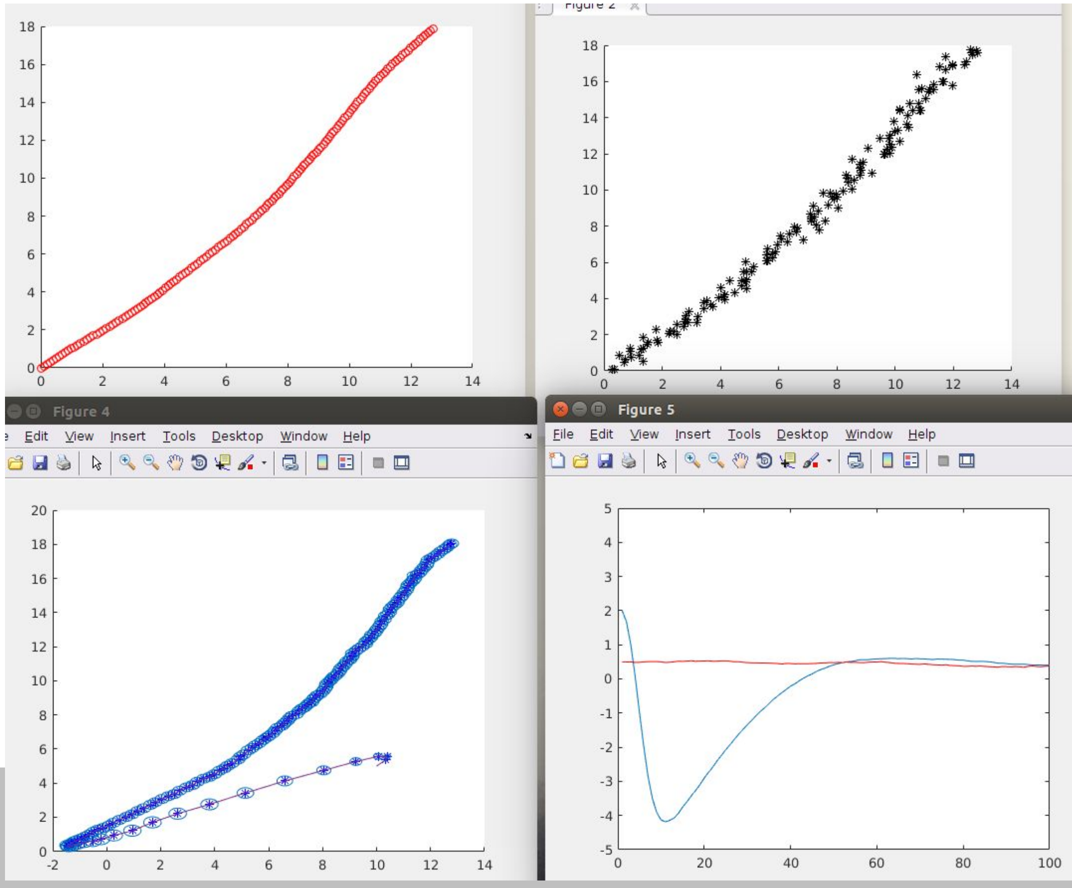
```
%initial guess
SigmaX = 10 * eye(4);
muX = [0 0 -10 -5]';

x_est = [];
x_est = [x_est,muX];
for i= 1:nSteps
    z = sensor(:,i);
    % predict
    muX = A*muX;
    SigmaX = A*SigmaX*A' + R;
    % correct
    K1 = SigmaX * H';
    K2 = H * SigmaX * H' + Q;
    K = K1 / K2;
    r = z - H * muX ;
    muX = muX + K * r;
    SigmaX = (eye(4) - K*H)*SigmaX;
    x_est = [x_est,muX];
    scatter(muX(1),muX(2),'b*');
    plot_cov(muX,SigmaX,3)
end
plot(x_est(1,:),x_est(2,:))
```

(Toy example) point moving in 2D plane with constant velocity



The power of Kalman filter! Our initial guess is very bad but the filter recover after few steps.

```
%initial guess
SigmaX = 0.01 * eye(4);
muX = [10 5 2 2]';

x_est = [];
x_est = [x_est,muX];
for i= 1:nSteps
    z = sensor(:,i);
    % predict
    muX = A*muX;
    SigmaX = A*SigmaX*A' + R;
    % correct
    K1 = SigmaX * H';
    K2 = H * SigmaX * H' + Q;
    K = K1 / K2;
    r = z - H * muX ;
    muX = muX + K * r;
    SigmaX = (eye(4) - K*H)*SigmaX;
    x_est = [x_est,muX];
    scatter(muX(1),muX(2),'b*');
    plot_cov(muX,SigmaX,3)
end
plot(x_est(1,:),x_est(2,:))
```



But ...
The motion model of the robot and the measurements model of the sensor are not linear.

Good news

Extended Kalman filter can solve this; Next Lecture.