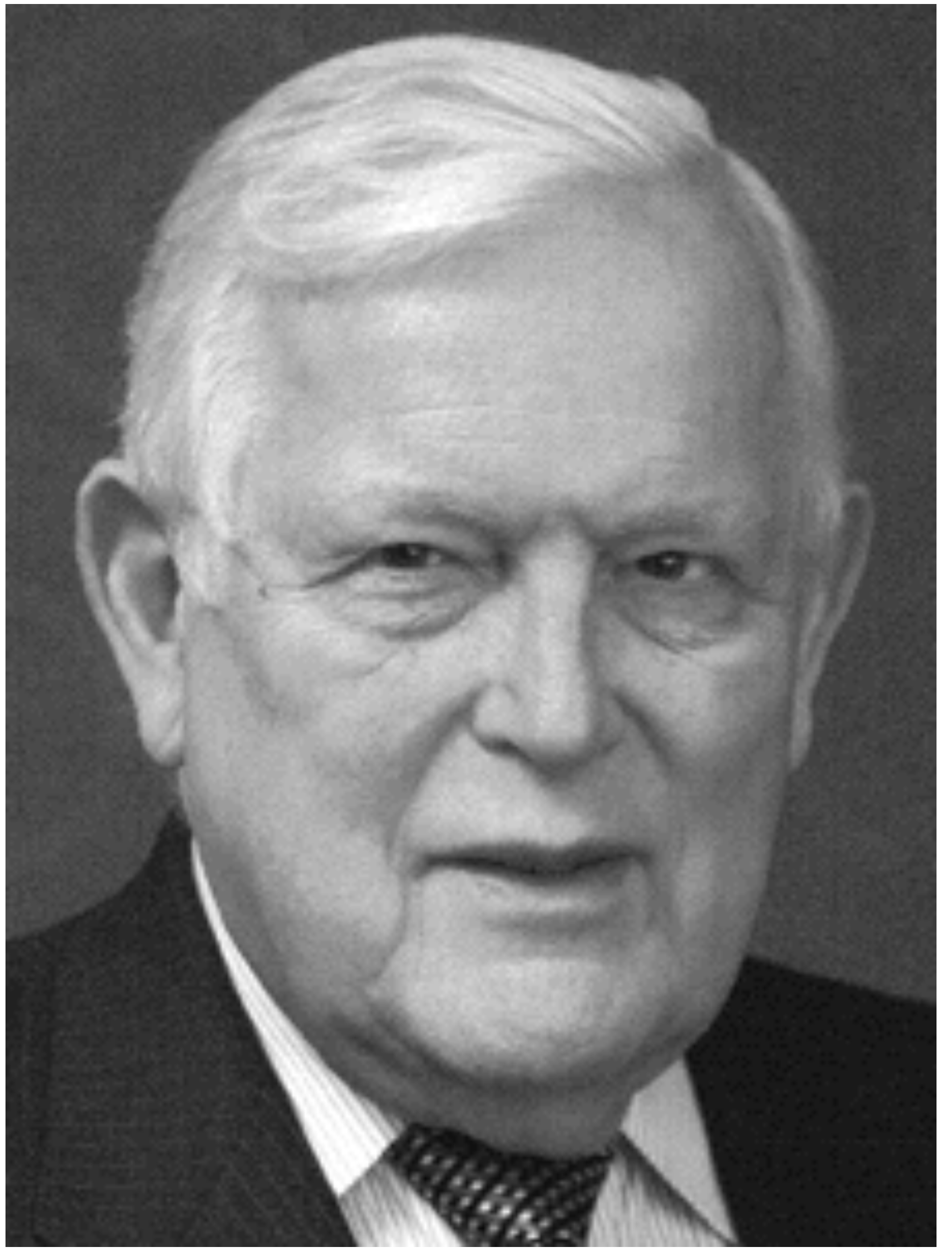


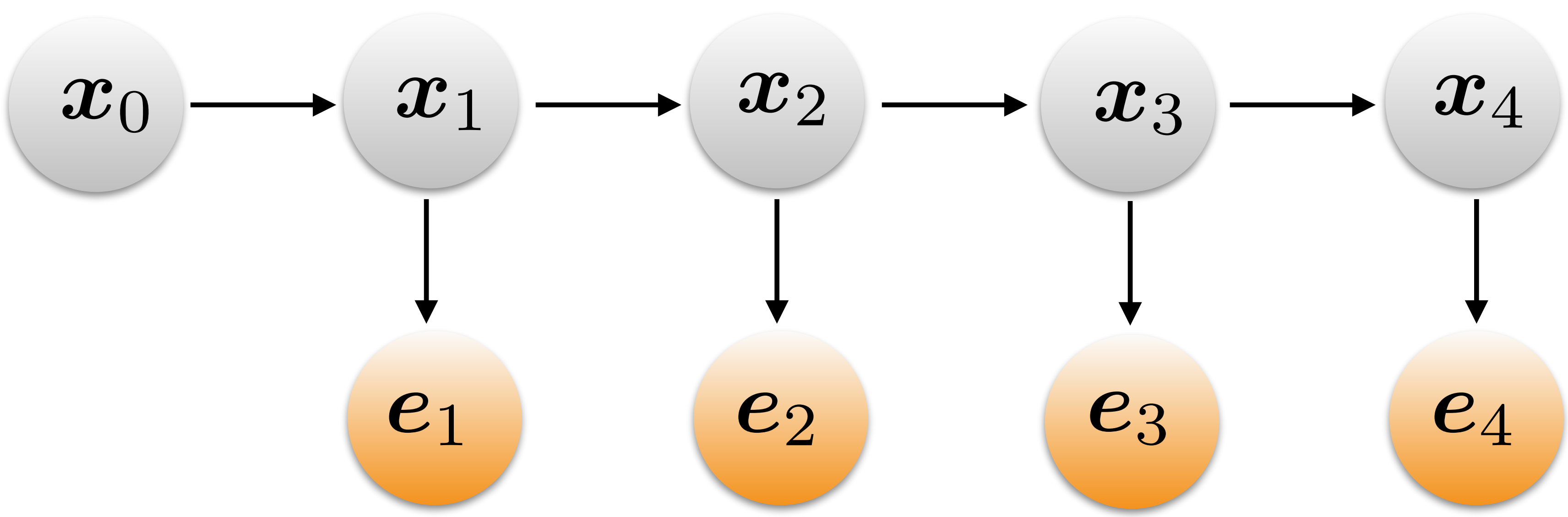
Kalman Filter

16-385 Computer Vision (Kris Kitani)

Carnegie Mellon University

Examples up to now have been **discrete** (binary) random variables

Kalman 'filtering' can be seen as a special case of a temporal inference with continuous random variables



Everything is continuous…

$$\boldsymbol{x} \qquad \boldsymbol{e} \qquad P(\boldsymbol{x}_0) \qquad P(\boldsymbol{e}|\boldsymbol{x}) \qquad P(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

probability distributions are no longer tables but functions

Making the connection to the 'filtering' equations

(Discrete) Filtering

Tables | Tables | Tables

$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \sum_{\boldsymbol{X}_t} P(\boldsymbol{X}_{t+1}|\boldsymbol{X}_t) P(\boldsymbol{X}_t|\boldsymbol{e}_{1:t})$$

Kalman Filtering

Gaussian | Gaussian | Gaussian

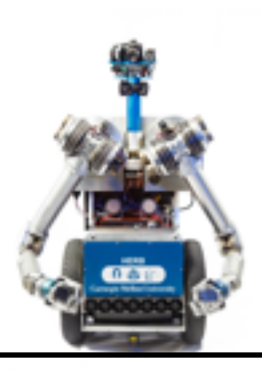
$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \int_{\boldsymbol{x}_t} P(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t) P(\boldsymbol{x}_t|\boldsymbol{e}_{1:t}) d\boldsymbol{x}_t$$

observation model | motion model | belief

integral because
continuous PDFs

Simple, 1D example…

x



x_1

$$\xrightarrow{\;s\;} \cdot \xrightarrow{\;r_2\;}$$

know velocity

noise

x_2

$$x_t = x_{t-1} + s + r_t$$

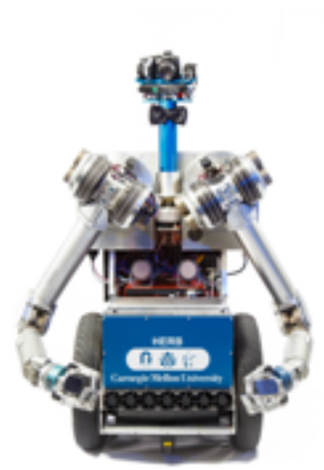
$$r_t \sim \mathcal{N}(0, \sigma_R)$$

'sampled from'

System (motion) model

$$x_t = x_{t-1} + s + r_t$$

$$r_t \sim \mathcal{N}(0, \sigma_R)$$



s r_2

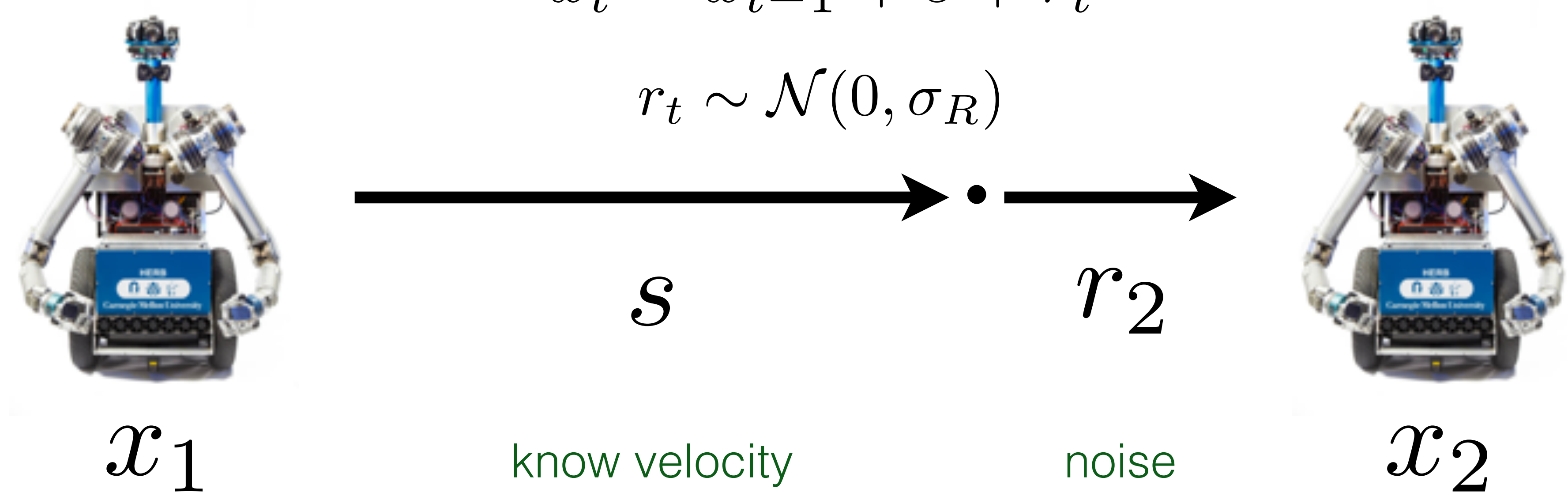
x_1 know velocity noise x_2

How do you represent the motion model?

$$P(x_t | x_{t-1})$$

$$x_t = x_{t-1} + s + r_t$$

$$r_t \sim \mathcal{N}(0, \sigma_R)$$

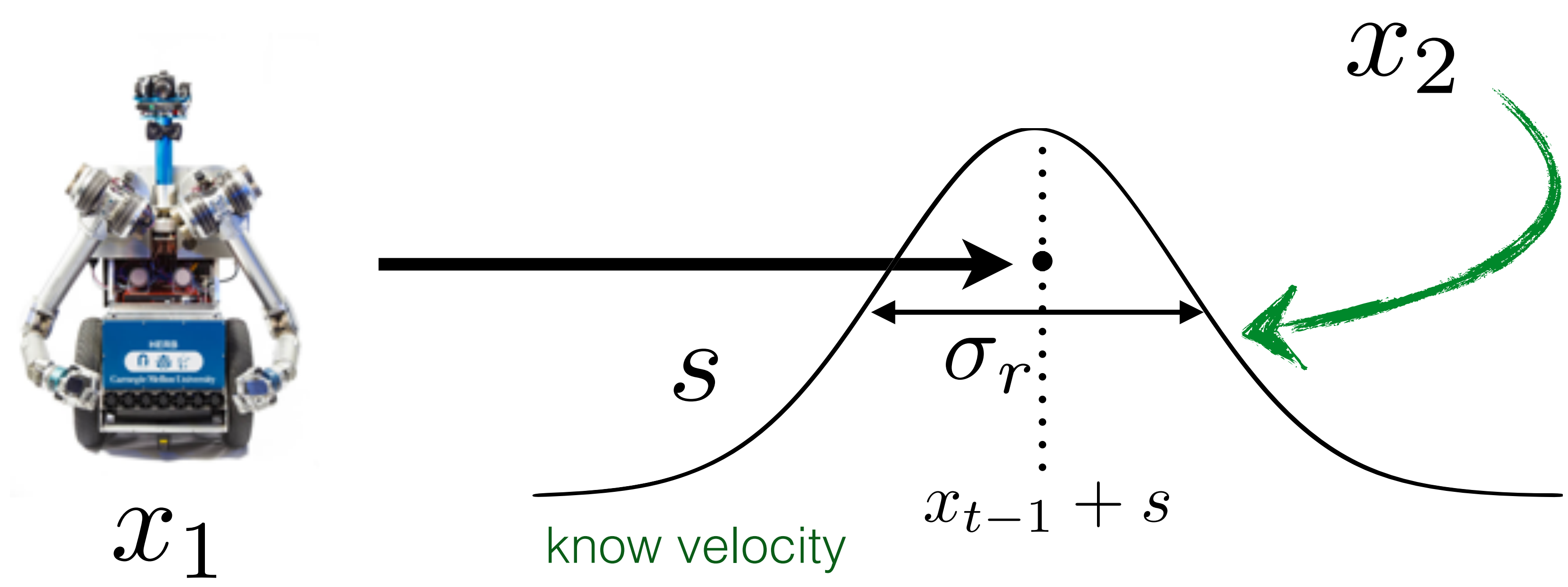


How do you represent the motion model?

A linear Gaussian (continuous) transition model

$$P(x_t | x_{t-1}) = \mathcal{N}(x_t; \underbrace{x_{t-1} + s}_{\text{mean}}, \underbrace{\sigma_r}_{\text{standard deviation}})$$

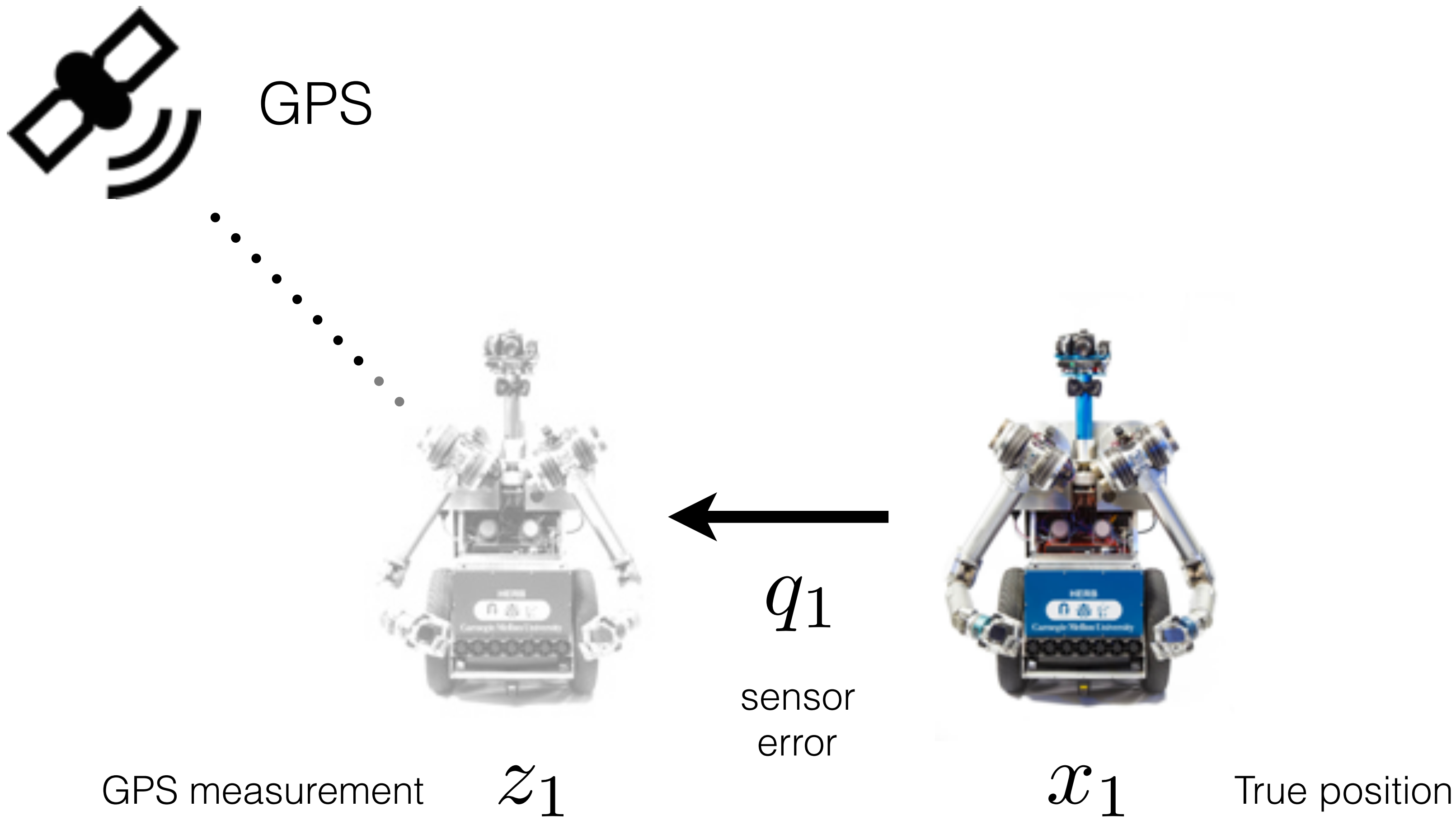
How can you visualize this distribution?



A linear Gaussian (continuous) transition model

$$P(x_t | x_{t-1}) = \mathcal{N}(x_t; x_{t-1} + s, \sigma_r)$$

Why don't we just use a table as before?

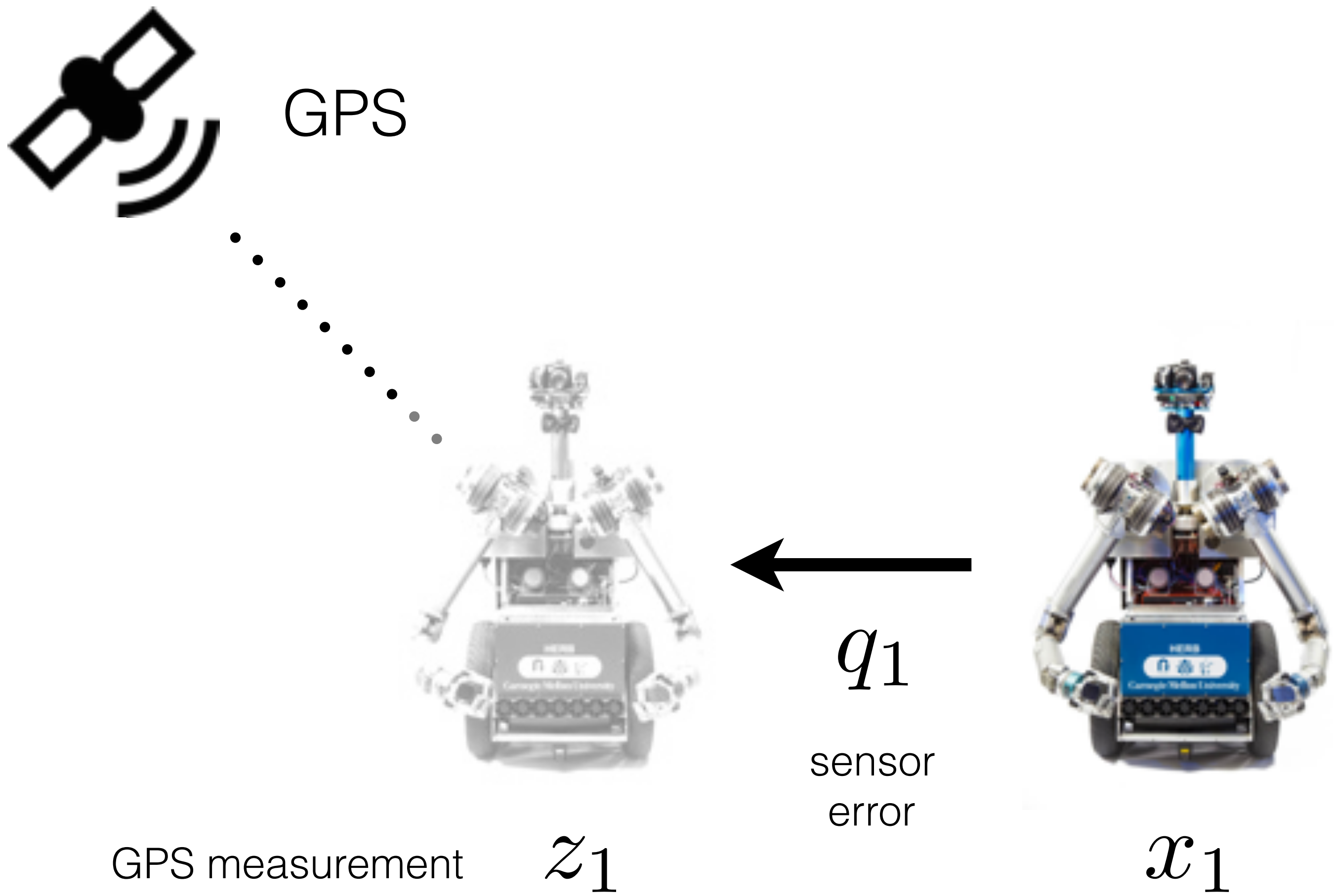


$$z_t = x_t + q_t$$

$$q_t \sim \mathcal{N}(0, \sigma_Q)$$

sampled from a Gaussian

Observation (measurement) model

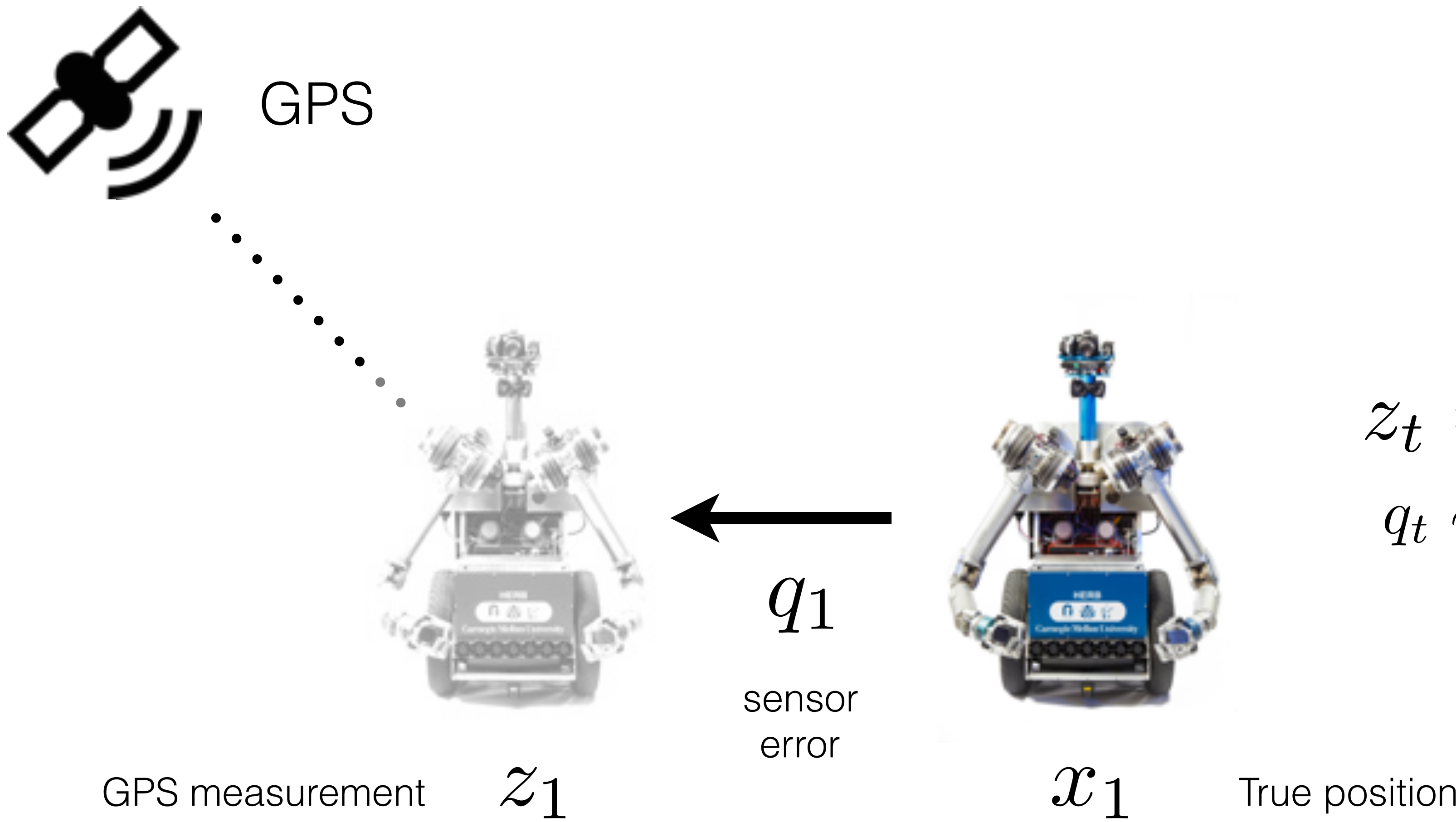


$$z_t = x_t + q_t$$
$$q_t \sim \mathcal{N}(0, \sigma_Q)$$

How do you represent the observation (measurement) model?

$$P(\boldsymbol{e}|\boldsymbol{x})$$

e represents z

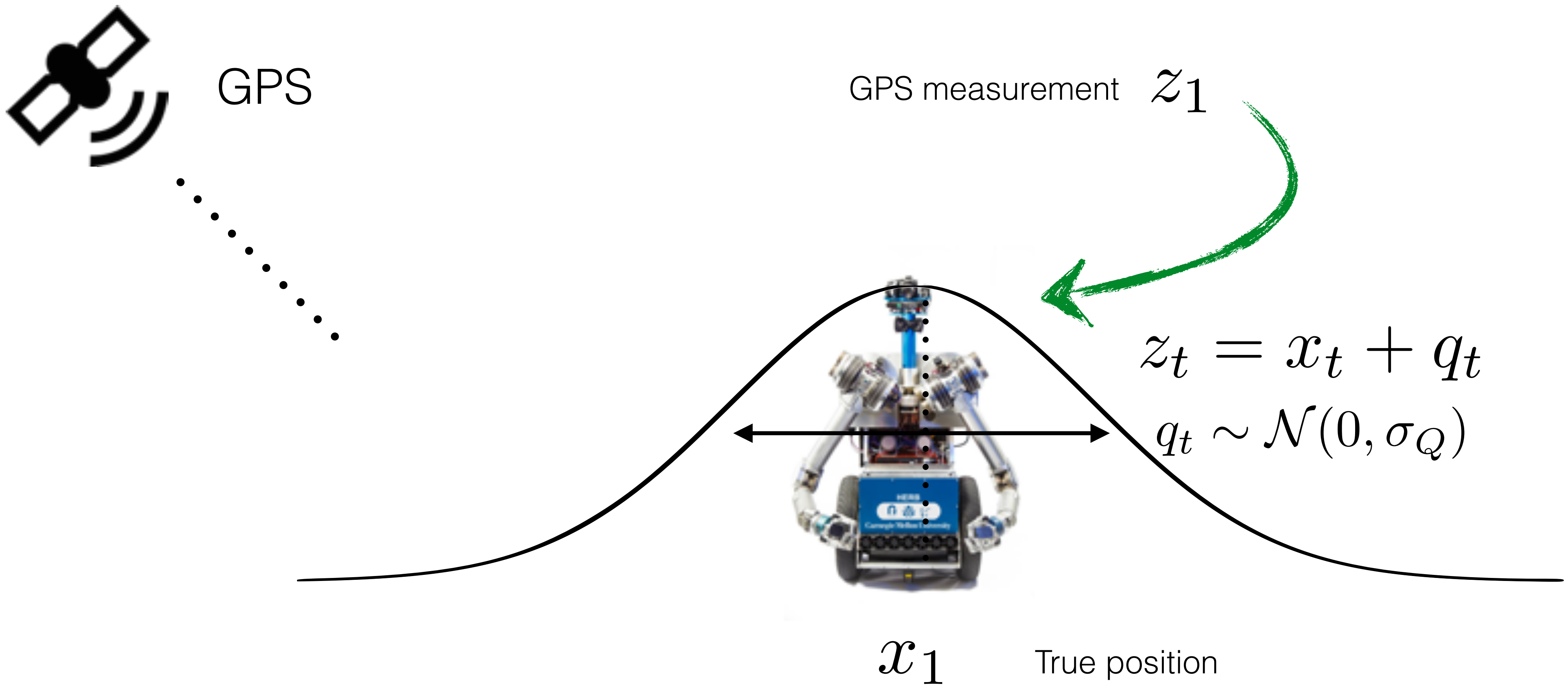


$$z_t = x_t + q_t$$
$$q_t \sim \mathcal{N}(0, \sigma_Q)$$

How do you represent the observation (measurement) model?

Also a linear Gaussian model

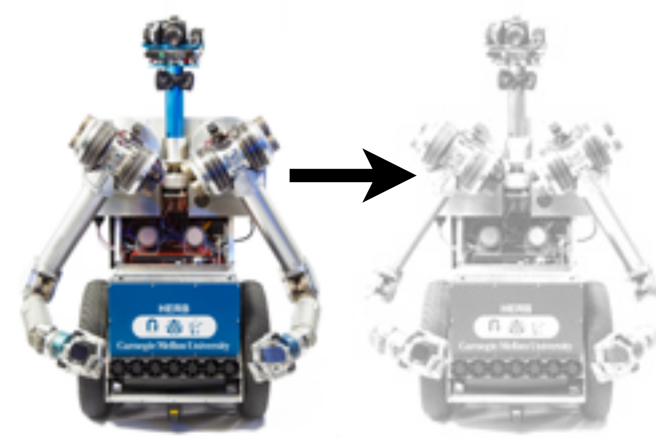
$$P(z_t | x_t) = \mathcal{N}(z_t; x_t, \sigma_Q)$$



How do you represent the observation (measurement) model?

Also a linear Gaussian model

$$P(z_t|x_t) = \mathcal{N}(z_t; x_t, \sigma_Q)$$

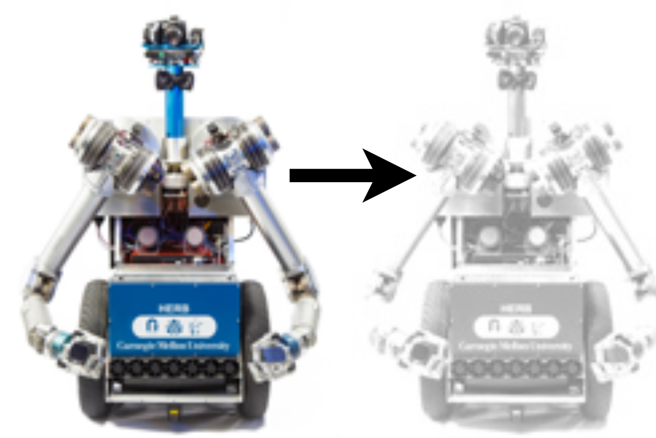


x_0 $\hat{x}_0$

true position initial estimate

initial estimate uncertainty σ_0

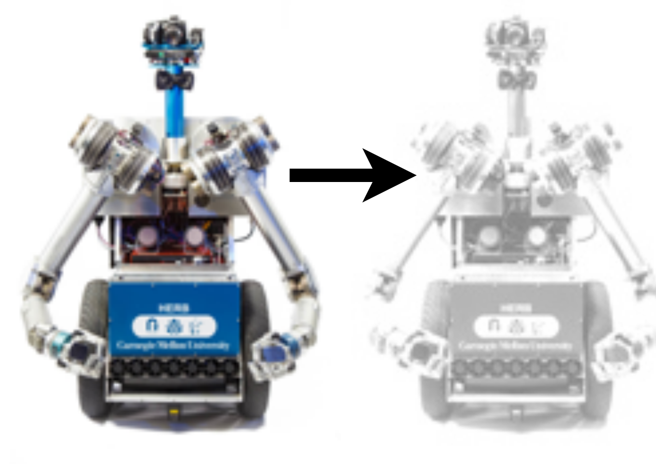
Prior (initial) State



x_0 $\hat{x}_0$

true position initial estimate

How do you represent the prior state probability?



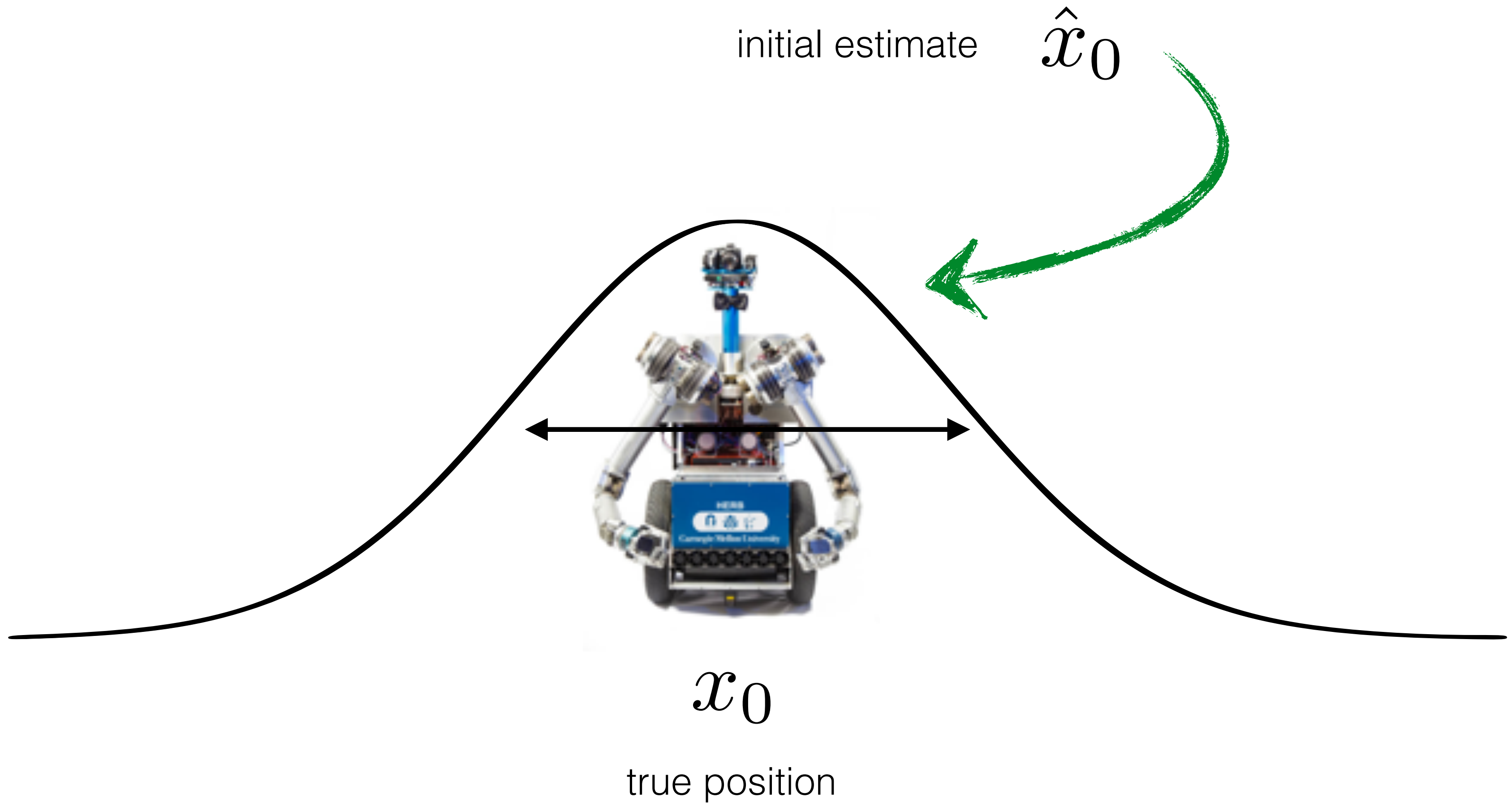
x_0 true position

$\hat{x}_0$ initial estimate

How do you represent the prior state probability?

Also a linear Gaussian model!

$$P(\hat{x}_0) = \mathcal{N}(\hat{x}_0; x_0, \sigma_0)$$



How do you represent the prior state probability?

Also a linear Gaussian model

$$P(\hat{x}_0) = \mathcal{N}(\hat{x}_0; x_0, \sigma_0)$$

Inference

So how do you do temporal filtering with the KL?

Recall: the first step of filtering was the 'prediction step'

$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \overbrace{\boxed{\int_{\boldsymbol{x}_t} \underbrace{P(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)}_{\text{motion model}} \underbrace{P(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})}_{\text{belief}} d\boldsymbol{x}_t}}^{\text{prediction step}}$$

compute this!
It's just another Gaussian

need to compute the 'prediction' mean and variance…

Prediction

(Using the motion model)

How would you predict $\hat{x}_1$ *given* $\hat{x}_0$ *?*

using this 'cap' notation to denote 'estimate'

$$\hat{x}_1 = \hat{x}_0 + s$$

(This is the mean)

$$\sigma_1^2 = \sigma_0^2 + \sigma_r^2$$

(This is the variance)

$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \overbrace{\boxed{\int_{\boldsymbol{x}_t} \underbrace{P(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)}_{\text{motion model}} \underbrace{P(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})}_{\text{belief}} d\boldsymbol{x}_t}}^{\text{prediction step}}$$

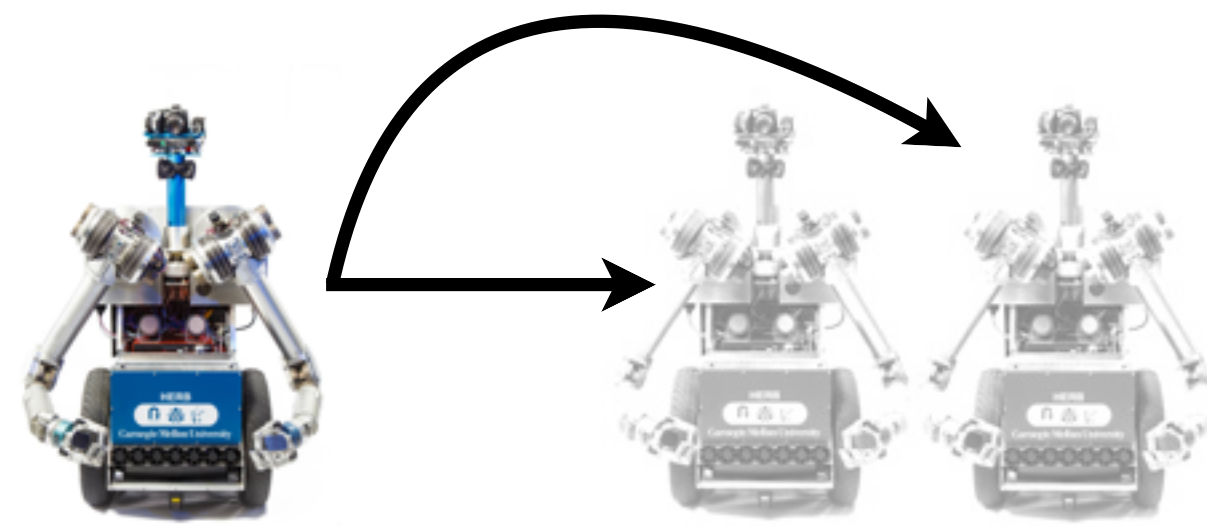
the second step after prediction is …

… update step!

$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto \boxed{\underbrace{P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1})}_{\text{observation}} \int_{\boldsymbol{x}_t} \underbrace{P(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)P(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})d\boldsymbol{x}_t}_{\text{prediction}}}$$

compute this
(using results of the prediction step)

In the **update step,** the **sensor measurement** **corrects** the system **prediction**



initial estimate	system prediction	sensor estimate
$\hat{x}_0$	$\hat{x}_1$	z_1
	σ_1^2	σ_q^2
	uncertainty	uncertainty

Which estimate is correct? Is there a way to know?

Is there a way to merge this information?

Intuitively, the smaller variance mean less uncertainty.

system prediction σ_1^2

sensor estimate σ_q^2

So we want a <u>weighted state estimate correction</u>

something like this…

$$\hat{x}_1^+ = \frac{\sigma_q^2}{\sigma_1^2 + \sigma_q^2}\hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_q^2} z_1$$

This happens naturally in the Bayesian filtering (with Gaussians) framework!

Recall the filtering equation:

observation

one step motion prediction

$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto \boxed{P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1})} \boxed{\int_{\boldsymbol{x}_t} P(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)P(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})d\boldsymbol{x}_t}$$

Gaussian

Gaussian

What is the product of two Gaussians?

Recall …

When we multiply the prediction (Gaussian) with the observation model (Gaussian) we get …

… a product of two Gaussians

$$\mu = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_2^2 + \sigma_1^2} \qquad \sigma = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

applied to the filtering equation…

$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \int_{\boldsymbol{x}_t} P(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)P(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})d\boldsymbol{x}_t$$

mean: z_1

variance: σ_q

mean: $\hat{x}_1$

variance: σ_1

new mean:

$$\hat{x}_1^+ = \frac{\hat{x}_1\sigma_q^2 + z_1\sigma_1^2}{\sigma_q^2 + \sigma_1^2}$$

'plus' sign means post 'update' estimate

new variance:

$$\hat{\sigma}_1^{2+} = \frac{\sigma_q^2\sigma_1^2}{\sigma_q^2 + \sigma_1^2}$$

system prediction σ_1^2

sensor estimate σ_q^2

With a little algebra…

$$\hat{x}_1^+ = \frac{\hat{x}_1 \sigma_q^2 + z_1 \sigma_1^2}{\sigma_q^2 + \sigma_1^2} = \hat{x}_1 \frac{\sigma_q^2}{\sigma_q^2 + \sigma_1^2} + z_1 \frac{\sigma_1^2}{\sigma_q^2 + \sigma_1^2}$$

We get a <u>weighted state estimate correction!</u>

Kalman gain notation

With a little algebra…

$$\hat{x}_1^+ = \hat{x}_1 + \frac{\sigma_1^2}{\sigma_q^2 + \sigma_1^2}(z_1 - \hat{x}_1) = \hat{x}_1 + K(z_1 - \hat{x}_1)$$

'Kalman gain' 'Innovation'

With a little algebra…

$$\sigma_1^+ = \frac{\sigma_1^2 \sigma_q^2}{\sigma_1^2 + \sigma_q^2} = \left(1 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_q^2}\right)\sigma_1^2 = (1 - \mathbf{K})\,\sigma_1^2$$

Summary (1D Kalman Filtering)

To solve this…

$$P(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) \propto P(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \int_{\boldsymbol{x}_t} P(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t) P(\boldsymbol{x}_t|\boldsymbol{e}_{1:t}) d\boldsymbol{x}_t$$

Compute this…

$$\hat{x}_1^+ = \hat{x}_1 + \boxed{\frac{\sigma_1^2}{\sigma_1^2 + \sigma_q^2}}(z_1 - \hat{x}_1) \qquad \sigma_1^{2+} = \sigma_1^2 - \boxed{\frac{\sigma_1^2}{\sigma_1^2 + \sigma_q^2}}\sigma_1^2$$

$$\boxed{K = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_q^2}}$$

'Kalman gain'

$$\hat{x}_1^+ = \hat{x}_1 + K(z_1 - \hat{x}_1) \qquad \sigma_1^{2+} = \sigma_1^2 - K\sigma_1^2$$

mean of the new Gaussian — variance of the new Gaussian

Simple 1D Implementation

```
[x p] = KF(x,v,z)

  x = x + s;
  v = v + q;

  K = v/(v + r);

  x   = x + K * (z - x);
  p   = v - K * v;
```

Just 5 lines of code!

or just 2 lines

```
[x P] = KF(x,v,z)
  x   = (x+s)+(v+q)/((v+q)+r)*(z-(x+s));
  p   = (v+q)-(v+q)/((v+q)+r)*v;
```

Bare computations (algorithm) of Bayesian filtering:

$$\mathrm{KalmanFilter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$$

motion control

prediction mean

$$\bar{\mu}_t = A_t \mu_{t-1} + B u_t$$

'old' mean

Prediction

'old' covariance

prediction covariance

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R$$

Gaussian noise

$$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$$

Gain

observation model

update mean

$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

Update

update covariance

$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Simple Multi-dimensional Implementation
(also 5 lines of code!)

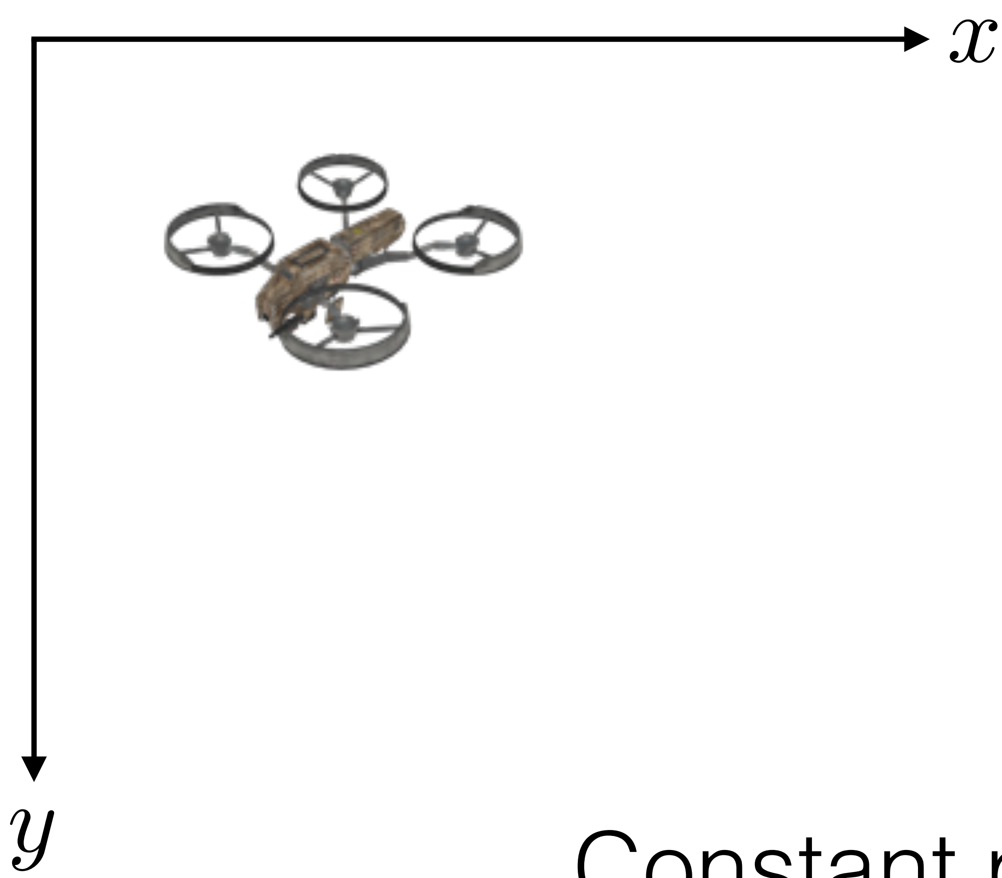
```
[x P] = KF(x,P,z)

  x = A*x;
  P = A*P*A' + Q;

  K = P*C'/(C*P*C' + R);

  x   = x + K*(z - C*x);
  P   = (eye(size(K,1))-K*C)*P;
```

2D Example



state

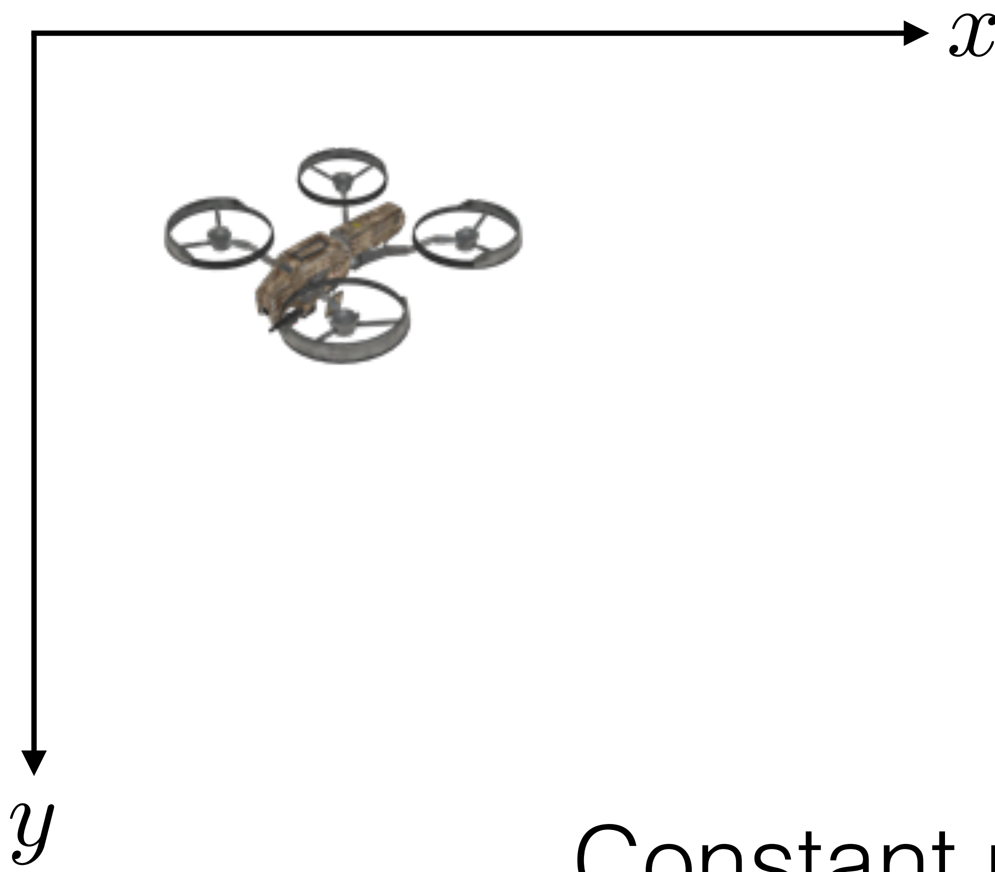
$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

measurement

$$\boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Constant position Motion Model

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_t + \epsilon_t$$



state

$$\boldsymbol{x} = \left[\begin{array}{c} x \\ y \end{array} \right]$$

measurement

$$\boldsymbol{z} = \left[\begin{array}{c} x \\ y \end{array} \right]$$

Constant position Motion Model

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_t + \epsilon_t$$

Constant position

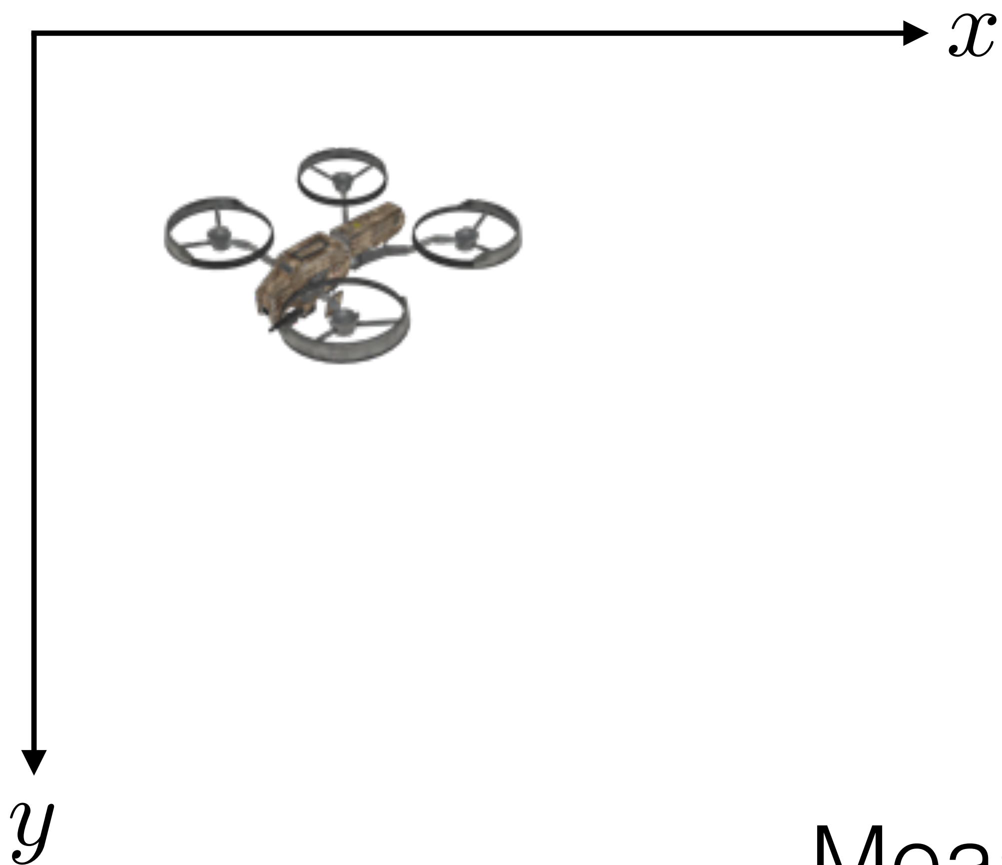
$$A = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right]$$

$$B\boldsymbol{u} = \left[\begin{array}{c} 0 \\ 0 \end{array} \right]$$

system noise

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

$$R = \left[\begin{array}{cc} \sigma_r^2 & 0 \\ 0 & \sigma_r^2 \end{array} \right]$$



state

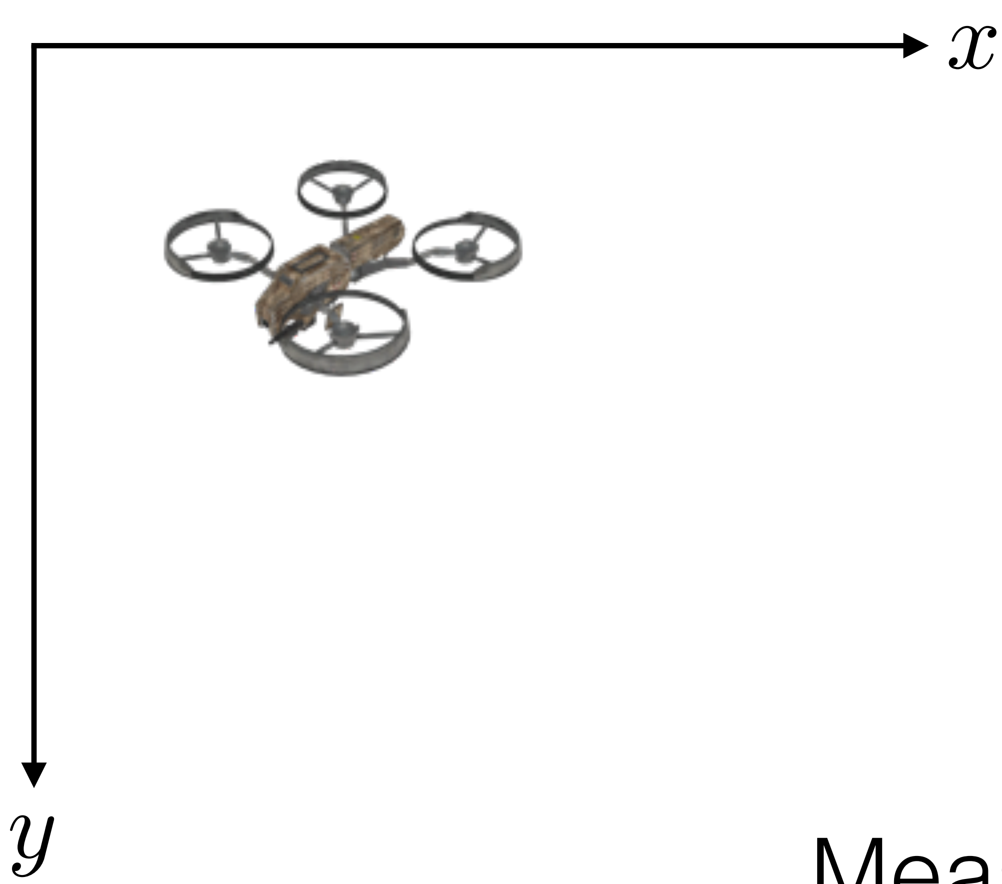
$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

measurement

$$\boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Measurement Model

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t + \delta_t$$



state

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

measurement

$$\boldsymbol{z} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Measurement Model

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t + \delta_t$$

zero-mean measurement noise

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \delta_t \sim \mathcal{N}(\mathbf{0}, Q) \qquad Q = \begin{bmatrix} \sigma_q^2 & 0 \\ 0 & \sigma_q^2 \end{bmatrix}$$

Algorithm for the 2D object tracking example

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

motion model observation model

General Case

$$\bar{\mu}_t = A_t \mu_{t-1} + B u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R$$
$$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Constant position Model

$$\bar{\boldsymbol{x}}_t = \boldsymbol{x}_{t-1}$$
$$\bar{\Sigma}_t = \Sigma_{t-1} + R$$
$$K_t = \bar{\Sigma}_t (\bar{\Sigma}_t + Q)^{-1}$$
$$\boldsymbol{x}_t = \bar{\boldsymbol{x}}_t + K_t (z_t - \bar{\boldsymbol{x}}_t)$$
$$\Sigma_t = (I - K_t) \bar{\Sigma}_t$$

Just 4 lines of code

```
[x P] = KF_constPos(x,P,z)

  P = P + Q;

  K = P / (P + R);

  x   = x + K * (z - x);
  P   = (eye(size(K,1))-K) * P;
```

Where did the 5th line go?

General Case

$$\bar{\mu}_t = A_t \mu_{t-1} + B u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R$$
$$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

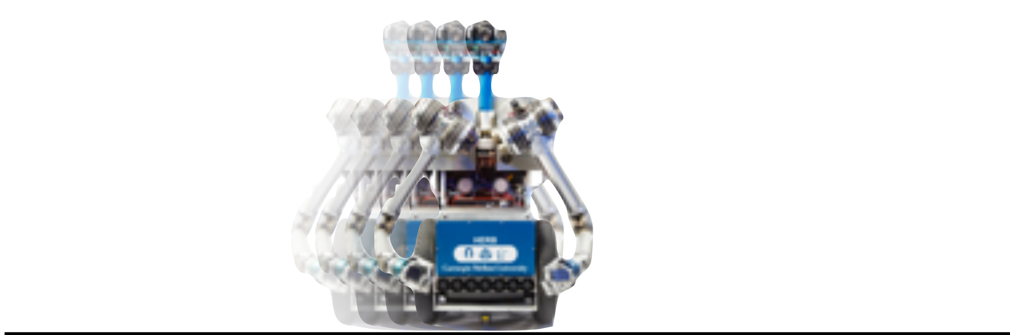
Constant position Model

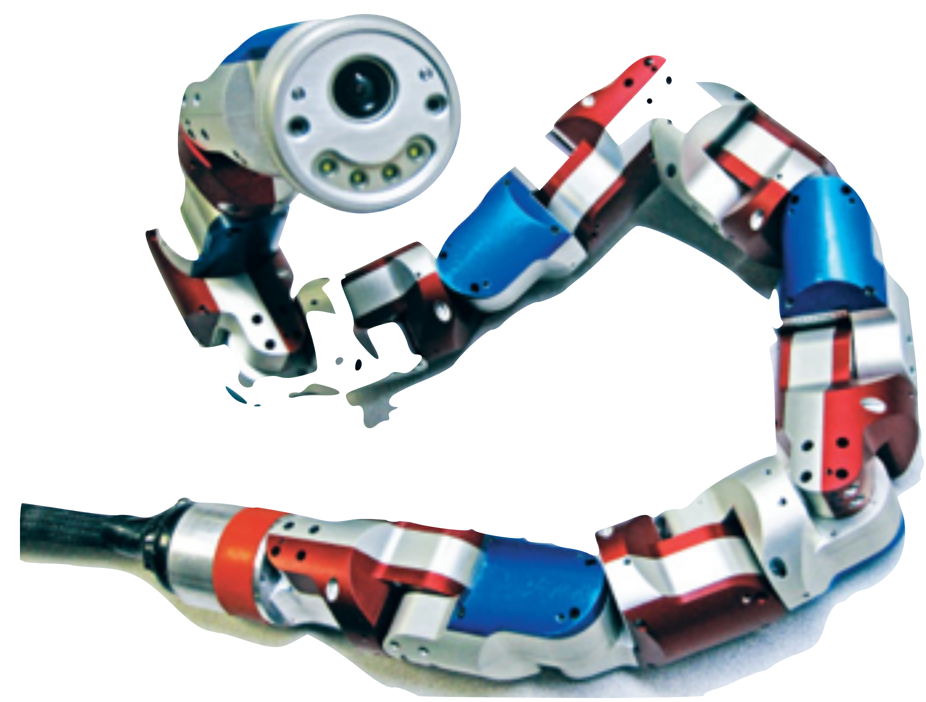
$$\bar{\boldsymbol{x}}_t = \boldsymbol{x}_{t-1}$$
$$\bar{\Sigma}_t = \Sigma_{t-1} + R$$
$$K_t = \bar{\Sigma}_t (\bar{\Sigma}_t + Q)^{-1}$$
$$\boldsymbol{x}_t = \bar{\boldsymbol{x}}_t + K_t (z_t - \bar{\boldsymbol{x}}_t)$$
$$\Sigma_t = (I - K_t) \bar{\Sigma}_t$$

Motion model of the Kalman filter is linear

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

but motion is not always linear

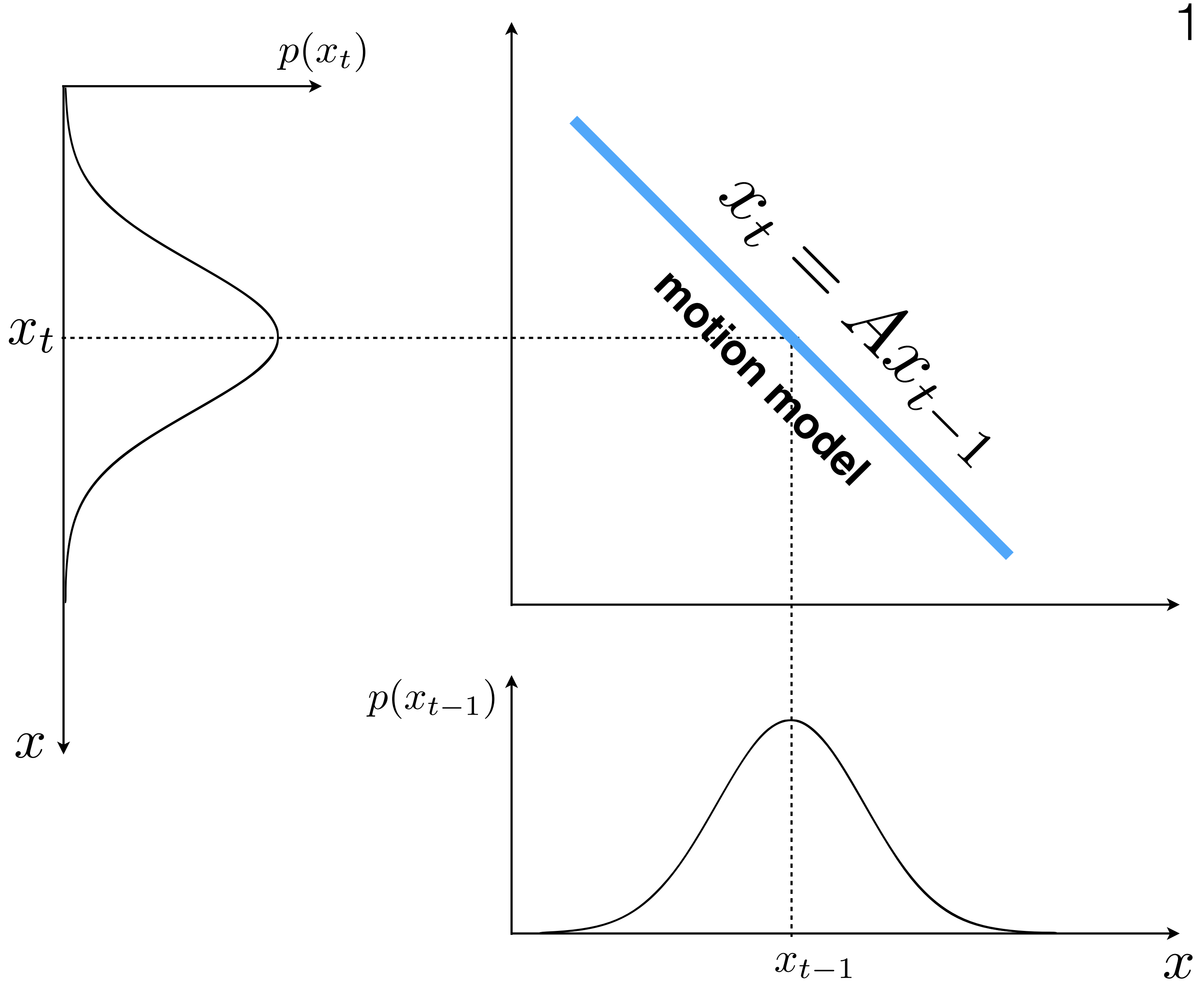




Visualizing **linear** models

1D motion model example

Output:
Gaussian **(Prediction)**



Input:
Gaussian **(Belief)**

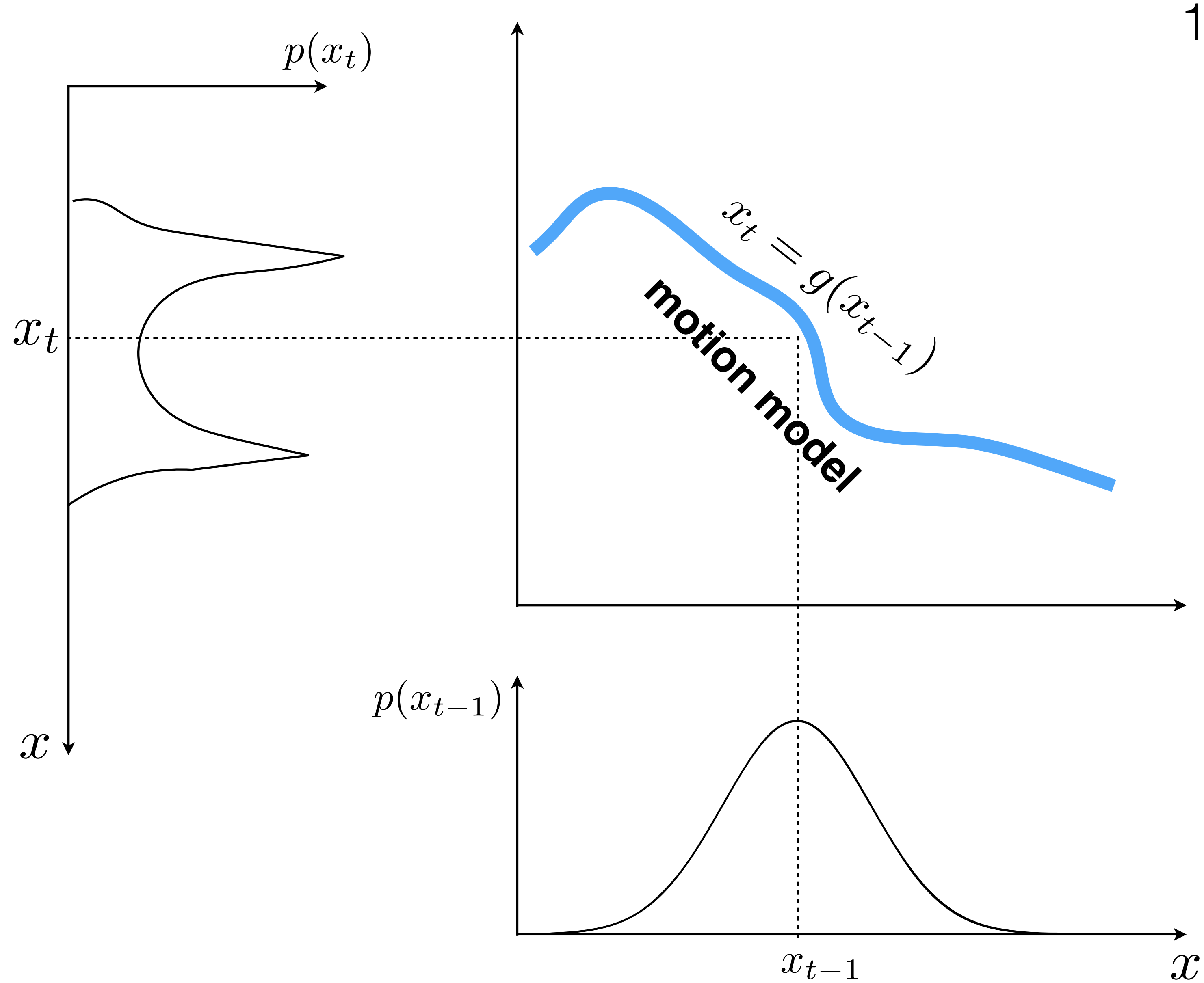
Can we use the Kalman Filter?

(motion model and observation model are linear)

Visualizing **non-linear** models

1D motion model example

Output:
NOT
Gaussian



Input:
Gaussian **(Belief)**

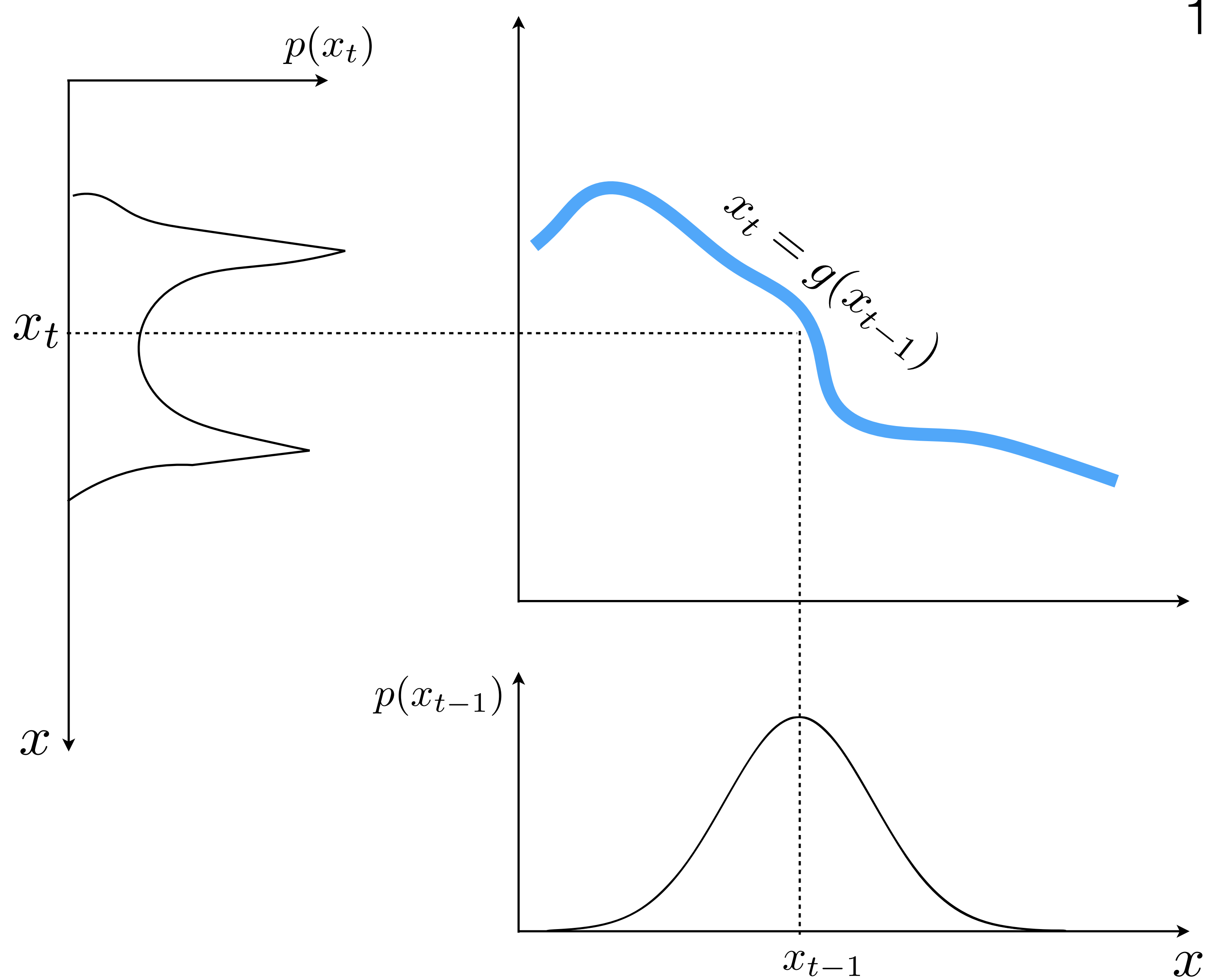
Can we use the Kalman Filter?

(motion model is not linear)

How do you deal with non-linear models?

1D motion model
example

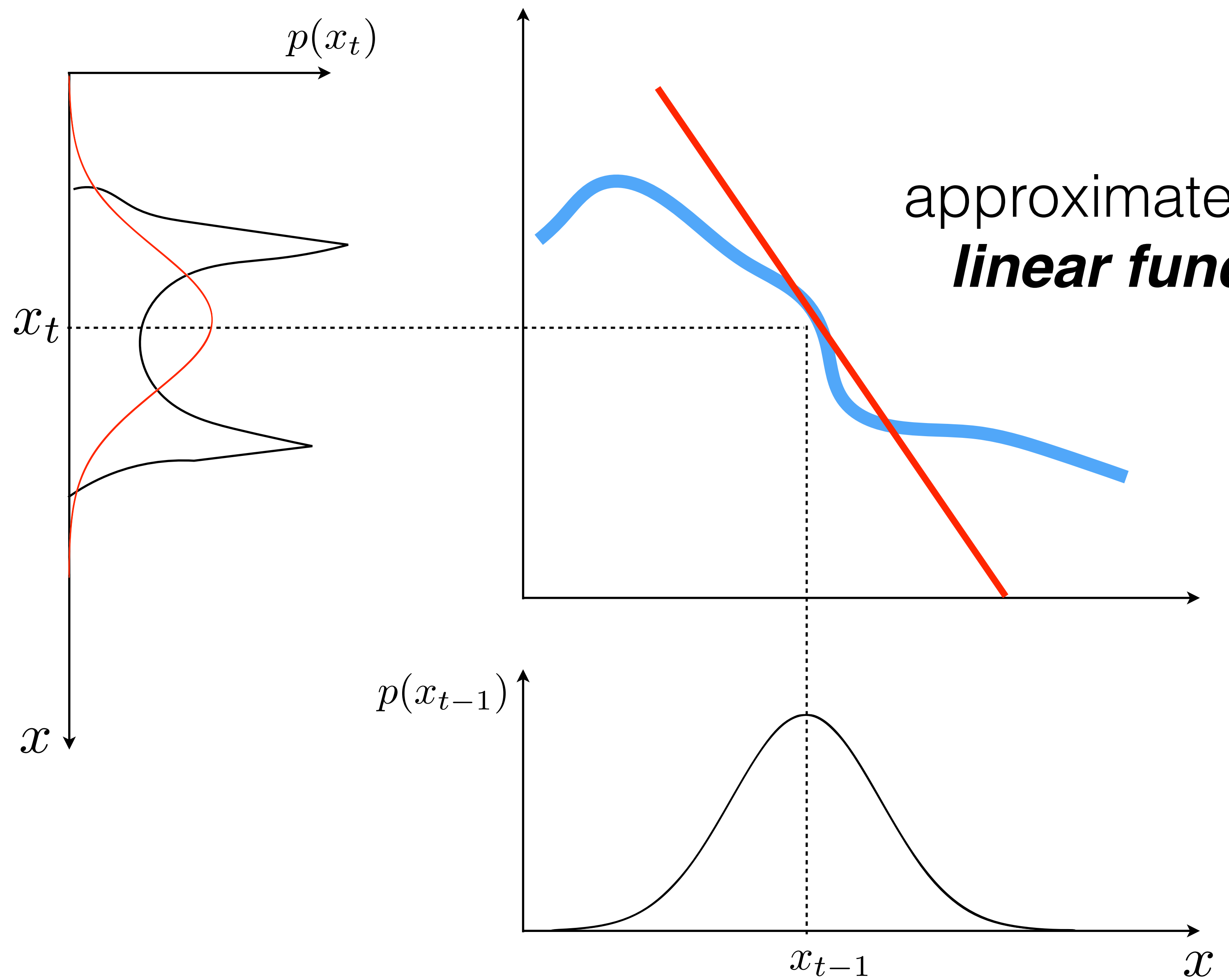
Output:
NOT
Gaussian



Input:
Gaussian

How do you deal with non-linear models?

Output:
Gaussian



approximate with a
linear function

When does this trick work?

Input:
Gaussian

Extended Kalman Filter

- Does not assume linear Gaussian models
- Assumes Gaussian noise
- Uses local linear approximations of model to keep the efficiency of the KF framework

Kalman Filter	Extended Kalman Filter
linear motion model	non-linear motion model
$x_t = Ax_{t-1} + Bu_t + \epsilon_t$	$x_t = g(x_{t-1}, u_t) + \epsilon_t$
linear sensor model	non-linear sensor model
$z_t = C_t x_t + \delta_t$	$z_t = H(x_t) + \delta_t$

Motion model linearization

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$

Taylor series expansion

Motion model linearization

$$
\begin{aligned}
g(x_{t-1}, u_t) &\approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1}) \\
&\approx g(\mu_{t-1}, u_t) + \qquad G_t \qquad (x_{t-1} - \mu_{t-1})
\end{aligned}
$$

What's this called?

Motion model linearization

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$

$$\approx g(\mu_{t-1}, u_t) + \qquad G_t \qquad (x_{t-1} - \mu_{t-1})$$

What's this called?

Jacobian Matrix

'the rate of change in x'
'slope of the function'

Motion model linearization

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$

$$\approx g(\mu_{t-1}, u_t) + \qquad G_t \qquad (x_{t-1} - \mu_{t-1})$$

Jacobian Matrix

'the rate of change in x'
'slope of the function'

Sensor model linearization

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t}(x_{t-1} - \bar{\mu}_t)$$

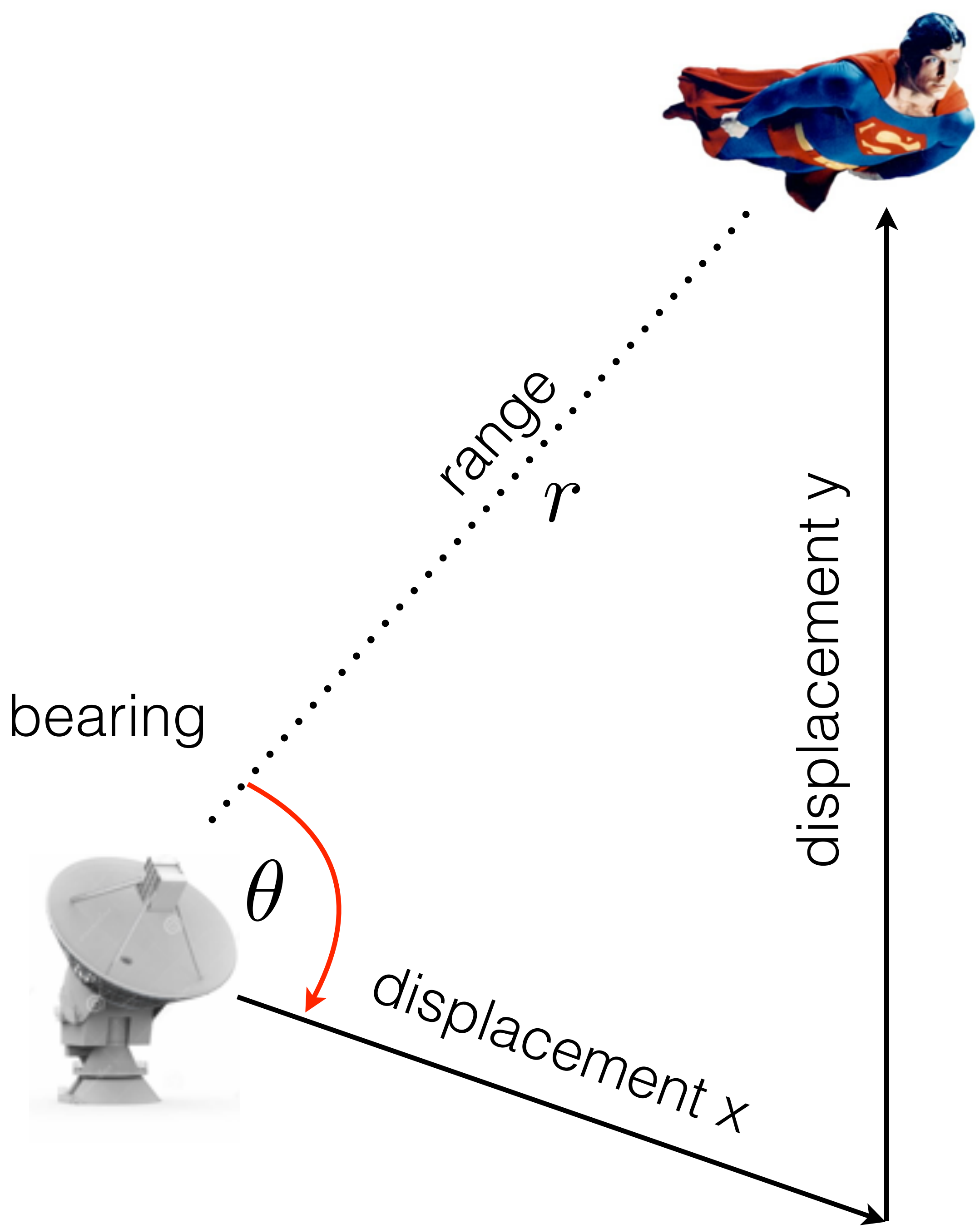
$$\approx h(\bar{\mu}_t) + \quad H_t \quad (x_t - \bar{\mu}_t)$$

New EKF Algorithm

(pretty much the same)

Kalman Filter	Extended KF
$\bar{\mu}_t = A_t \mu_{t-1} + B u_t$	$\bar{\mu}_t = g(\mu_{t-1}, u_t)$
$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + R$	$\bar{\Sigma}_t = G_t \bar{\Sigma}_{t-1} G_t^\top + R$
$K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + Q_t)^{-1}$	$K_t = \bar{\Sigma}_t H_t^\top \left(H_t \bar{\Sigma}_t H^\top + Q \right)^{-1}$
$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$	$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$
$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$	$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$

2D example



state: position-velocity

$$\boldsymbol{x} = \begin{array}{cl} \left[\begin{array}{c} x \\ \dot{x} \\ y \\ \dot{y} \end{array}\right] & \begin{array}{l} \text{position} \\ \text{velocity} \\ \text{position} \\ \text{velocity} \end{array} \end{array}$$

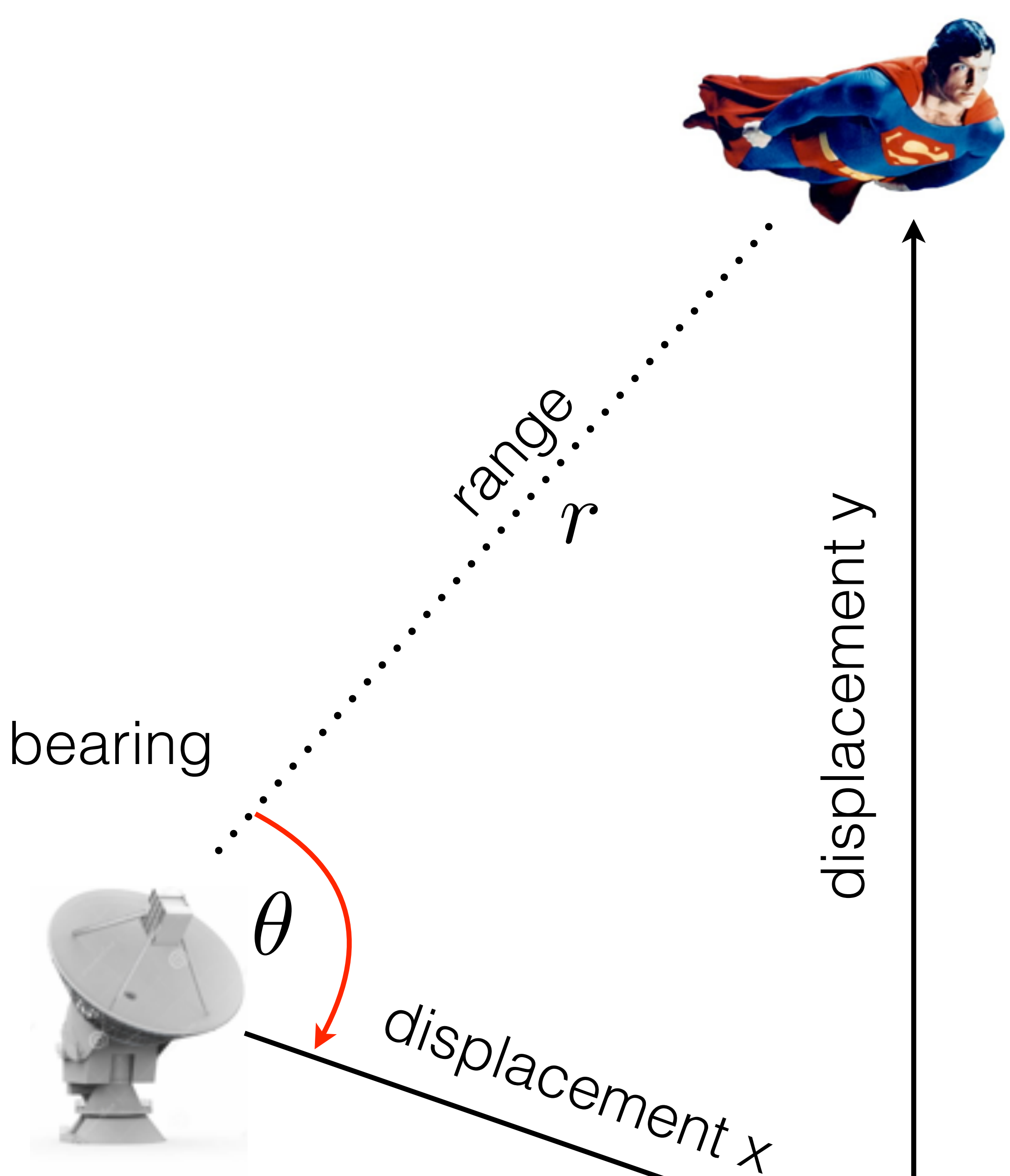
constant velocity motion model

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

with additive Gaussian noise

Motion model is linear but …

measurement: range-bearing



$$\boldsymbol{z} = \begin{bmatrix} r \\ \theta \end{bmatrix}$$

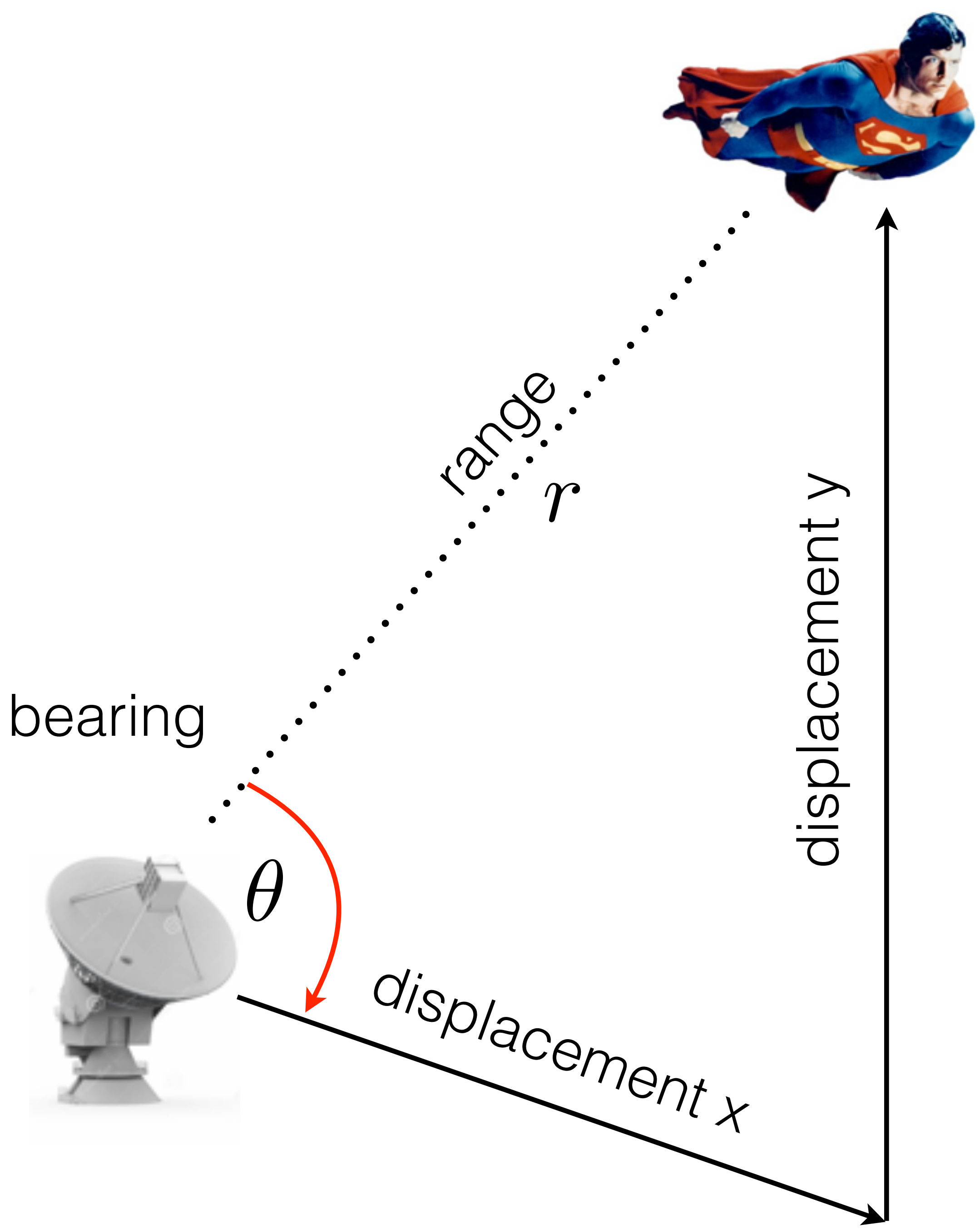
$$= \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

measurement model

Is the measurement model linear?

$$\boldsymbol{z} = h(r, \theta)$$

with additive Gaussian noise



measurement: range-bearing

$$\boldsymbol{z} = \begin{bmatrix} r \\ \theta \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

measurement model

Is the measurement model linear?

$$\boldsymbol{z} = h(r, \theta)$$

with additive Gaussian noise

non-linear!

What should we do?

linearize the observation/measurement model!

$$\boldsymbol{z} = \begin{bmatrix} r \\ \theta \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

$$H = \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = ?$$

What is the Jacobian?

$$H = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial \dot{x}} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial \dot{y}} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \dot{x}} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \dot{y}} \end{bmatrix} =$$

$$\boldsymbol{z} = \begin{bmatrix} r \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1}(y/x) \end{bmatrix}$$

$$H = \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = ?$$

What is the Jacobian?

Jacobian used in the Taylor series expansion looks like …

$$H = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial \dot{x}} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial \dot{y}} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \dot{x}} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \dot{y}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ -sin(\theta)/r & 0 & \cos(\theta)/r & 0 \end{bmatrix}$$

```
[x P] = EKF(x,P,z,dt)

  r    = sqrt (x(1)^2+x(3)^2);
  b    = atan2(x(3),x(1));
  y    = [r; b];

  H = [ cos(b)     0  sin(b)     0;
       -sin(b)/r 0  cos(b)/r   0];

  x = F*x;
  P = F*P*F' + Q;

  K = P*H'/(H*P*H' + R);

  x    = x + K*(z - y);
  P    = (eye(size(K,1))-K*H)*P;
```

extra computation for the EKF measurement model Jacobian

Parameters:

```
Q = diag([0 .1 0 .1]);
R = diag([50^2 0.005^2]);
F = [ 1 dt 0 0;
      0 1 0 0;
      0 0 1 dt;
      0 0 0 1];
```

Problems with EKFs

Taylor series expansion = poor approximation of non-linear functions
success of linearization depends on limited uncertainty and amount of local non-linearity

Computing partial derivatives is a pain

Drifts when linearization is a bad approximation

Cannot handle multi-modal (multi-hypothesis) distributions