Algorithms for Sensor-Based Robotics: Kalman Filters for Mapping and Localization

Sensors!

Robots' link to the external world

(obsession with depth)

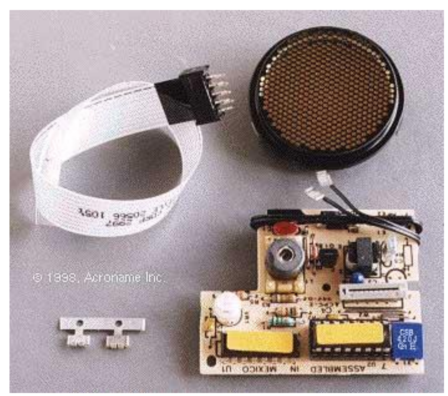
Laser

Sensors, sensors, sensors!

and tracking what is sensed: world models

Kinect

IR rangefinder



sonar rangefinder

Light-field camera

Sensors!

Robots' link to the external world...

Sensors, sensors, sensors!

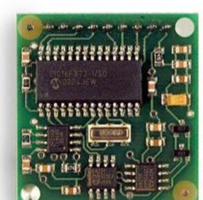
and tracking what is sensed: world models

gyro

GPS

Force/
Torque

Inertial
measurement unit
(gyro +
accelerometer)



compass

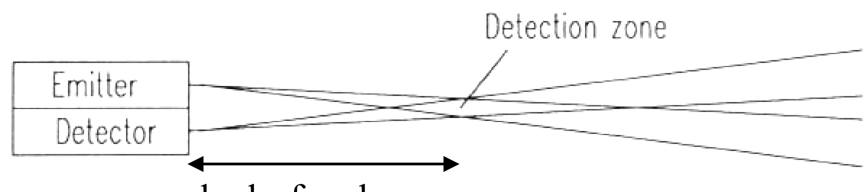
16-735, Howie Choset
with slides from G.D.
Hager and Z. Dodds

Infrared sensors

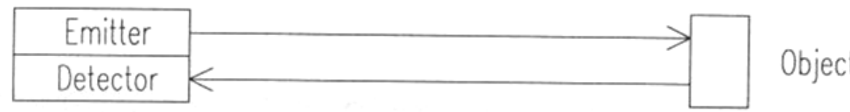
"Noncontact bump sensor"

(1) sensing is based on light intensity.

"object-sensing" IR

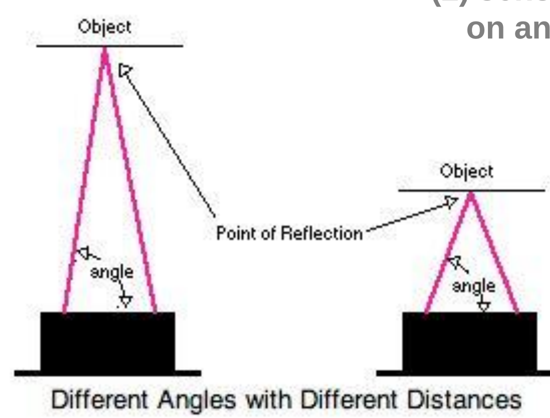


looks for changes
at this distance



diffuse distance-sensing IR

(2) sensing is based on angle receved.



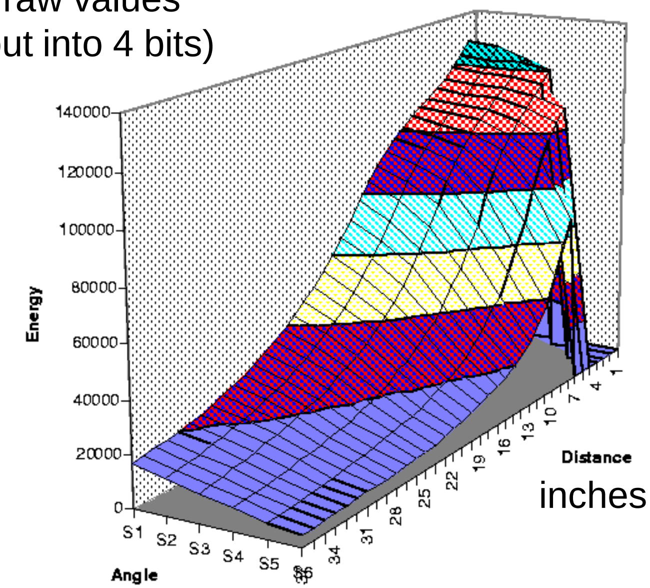
IR emitter/detector pair

IR detector

Infrared calibration

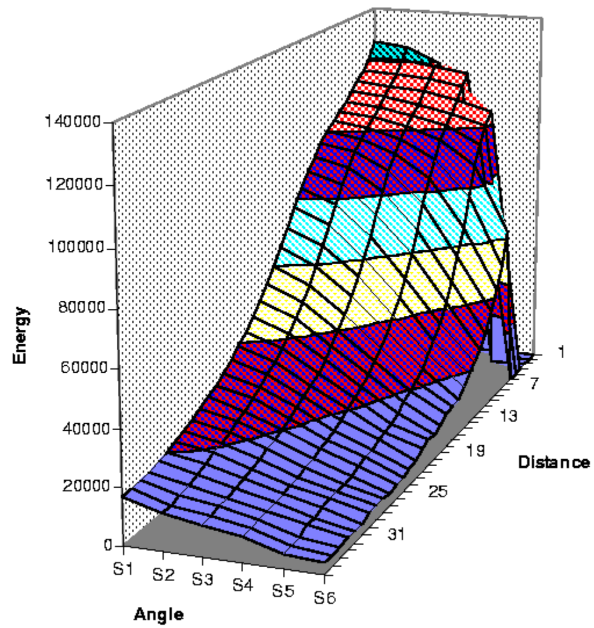
The response to white copy paper
(a dull, reflective surface)

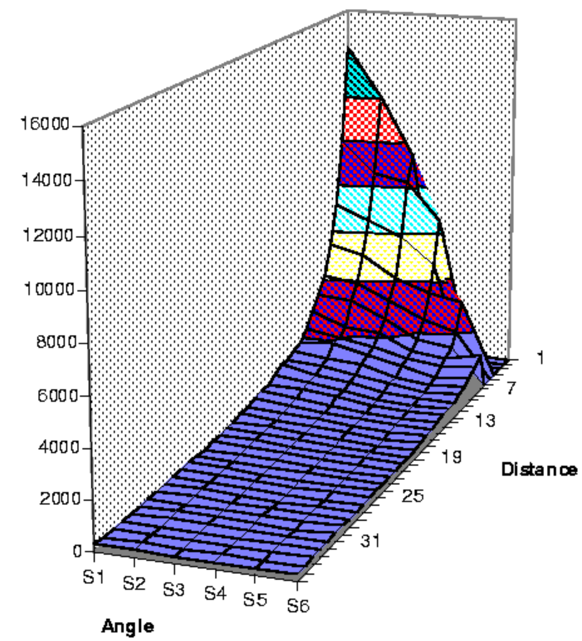
raw values
(put into 4 bits)



inches

15º increments



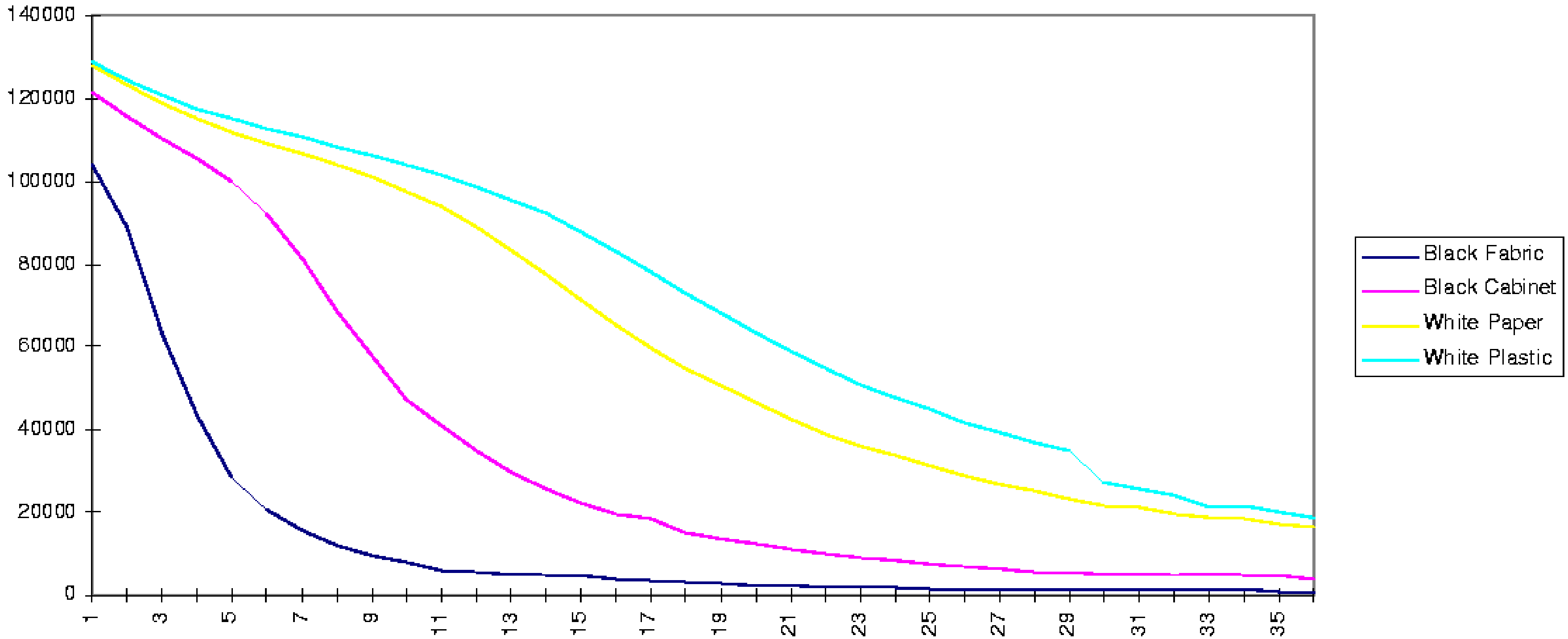


in the dark

fluorescent light

incandescent light

Infrared calibration



energy vs. distance for various materials

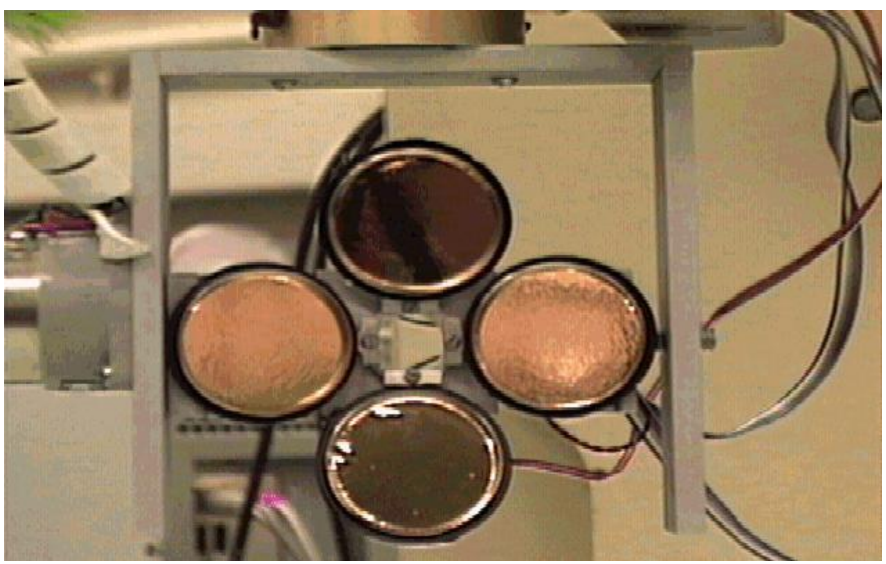
(the incident angle is 0^{o}, or head-on)

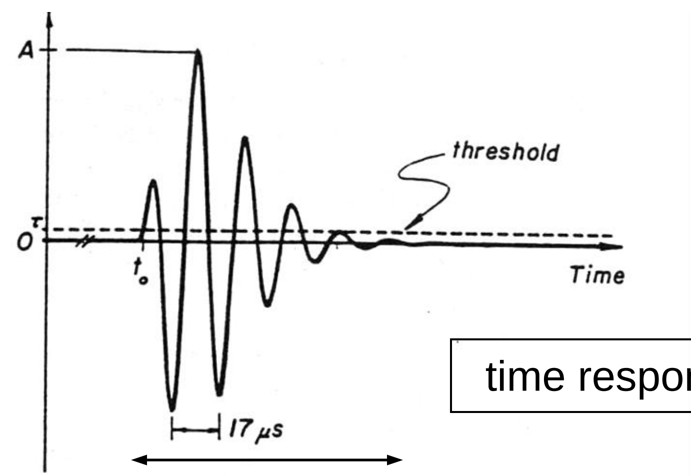
(with no ambient light)

Sonar sensing

single-transducer sonar timeline

0	75µs	the transducer goes into "receiving" mode and awaits a signal...	.5s
a "chirp" is emitted into the environment	typically when reverberations from the initial chirp have stopped	limiting range sensing	after a short time, the signal will be too weak to be detected





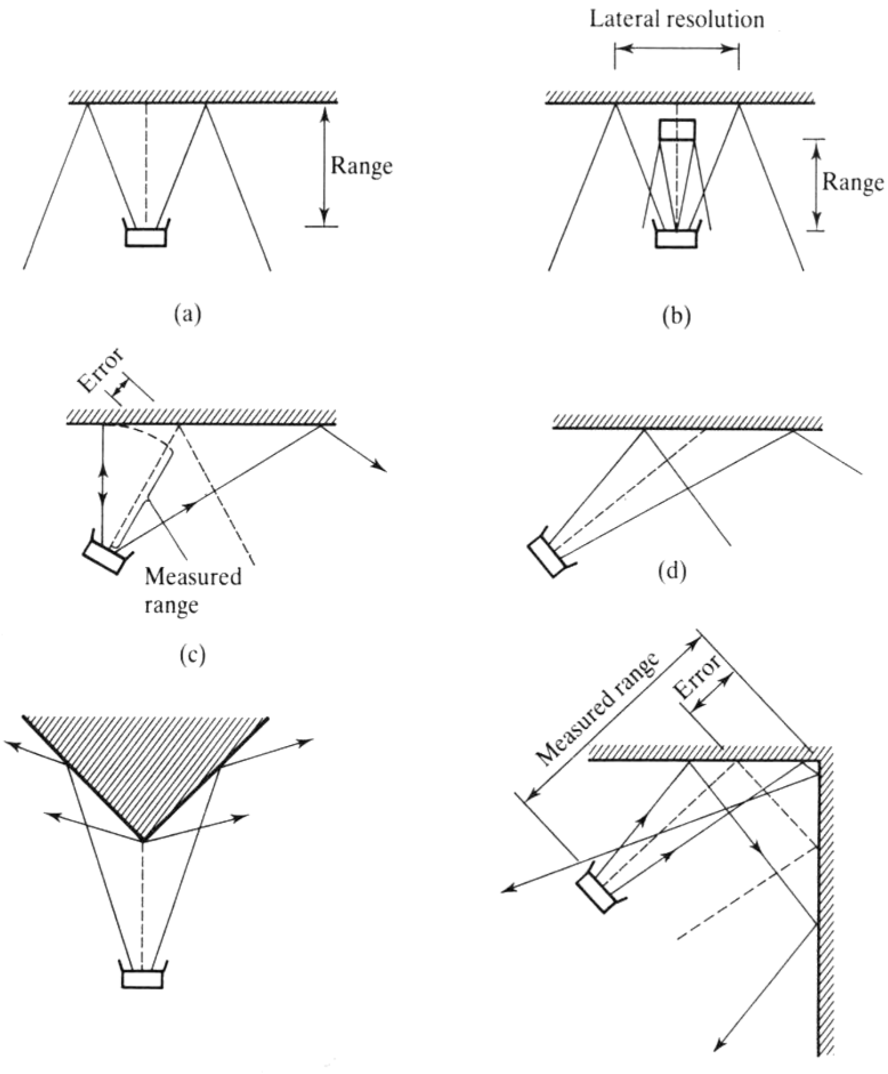
time response

blanking time

Polaroid sonar emitter/receivers

No lower range limit for *paired* sonars...

Sonar effects



(a) (b) (c) (d) (e) (f)

(a) Sonar providing an accurate range measurement

(b-c) Lateral resolution is not very precise; the closest object in the beam's cone provides the response

(d) Specular reflections cause walls to disappear

(e) Open corners produce a weak spherical wavefront

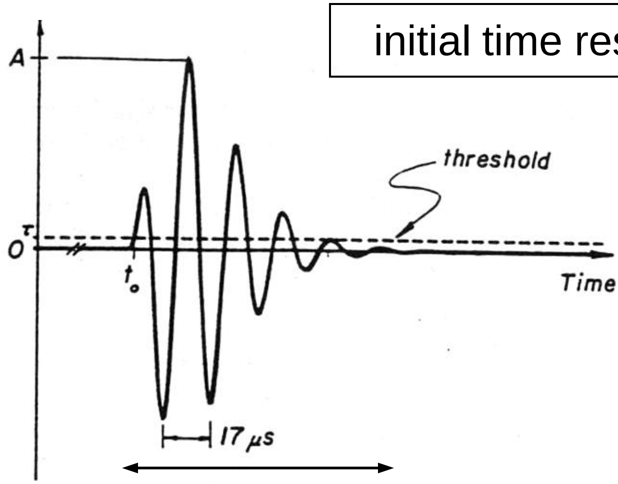
(f) Closed corners measure to the corner itself because of multiple reflections --> sonar ray tracing

resolution: time / space

with slides from G.D.

Sonar modeling

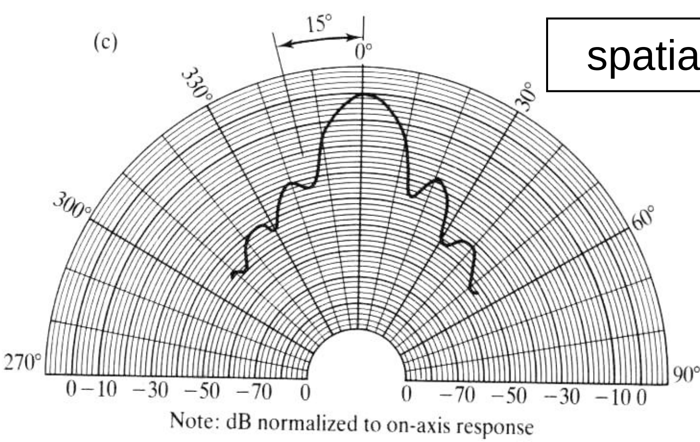
initial time response



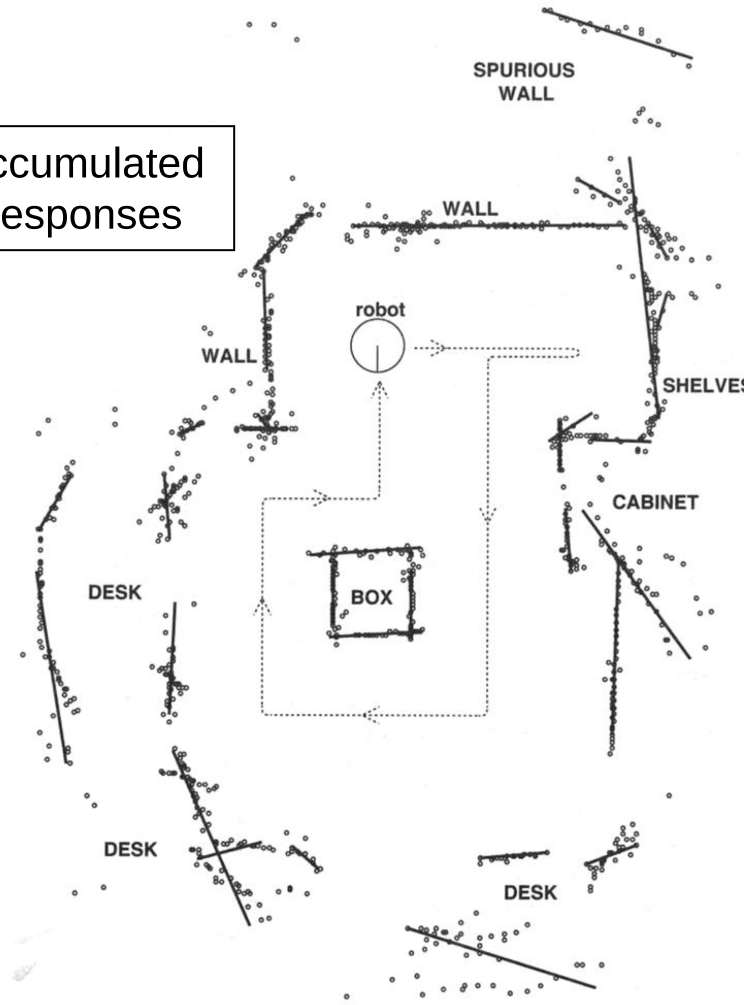
blanking time

cone width

spatial response



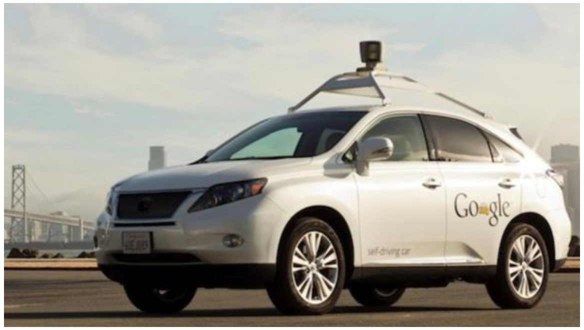
accumulated responses

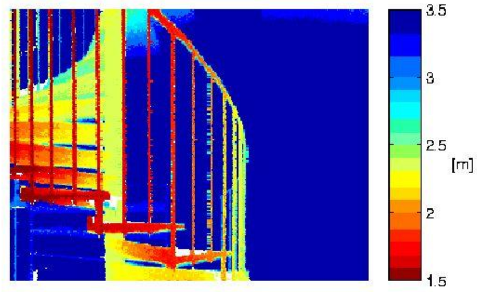


16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

Laser Ranging

LIDAR/Laser range finder





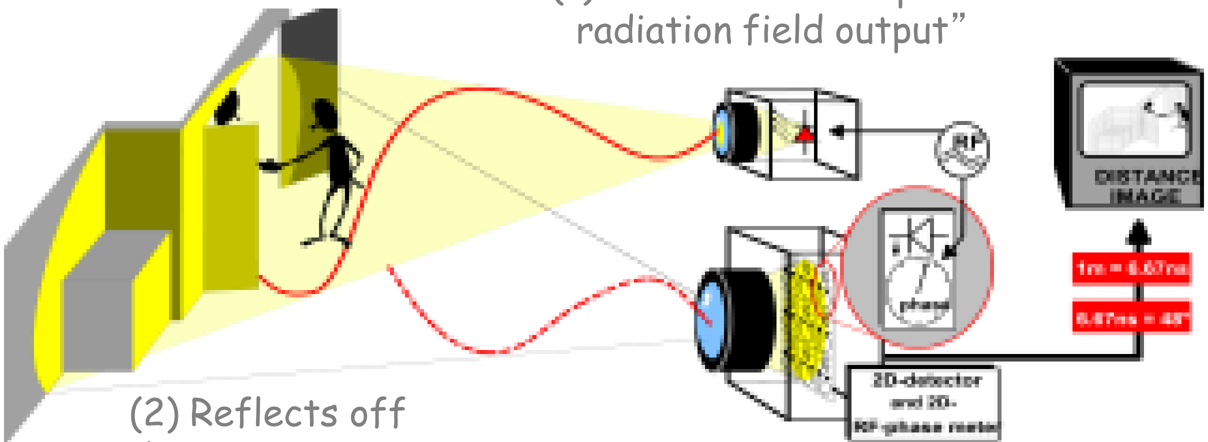
LIDAR map

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

Recent, Cool...

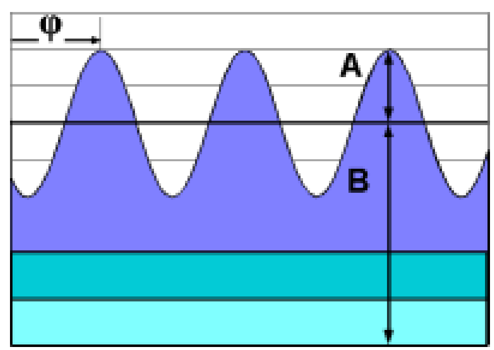
The 3d "time-of-flight" camera

(1) "RF-modulated optical radiation field output"

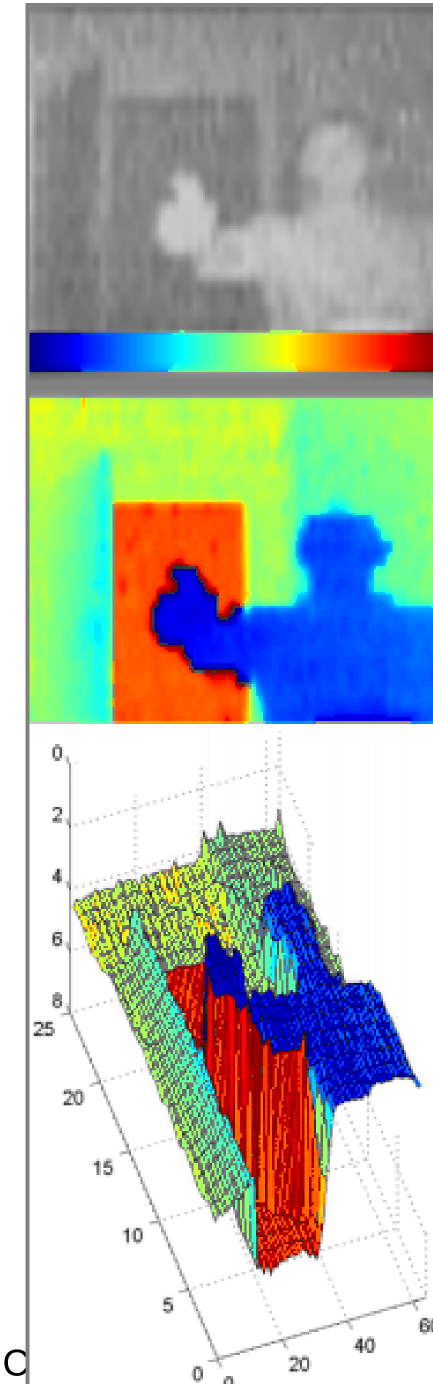


(2) Reflects off the environment

(3) Time-interval input measurements provides data
(not time-integrated!)



How can we get an image from this information?

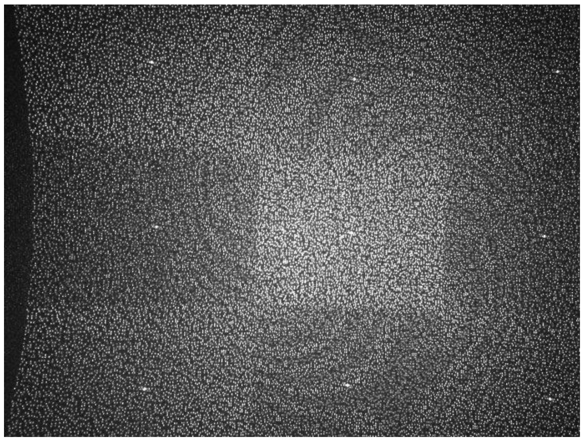


http://www.csem.ch /detailed/p_531_3d_cam.htm

C

(w/slides from Z.

More recent, Cooler...



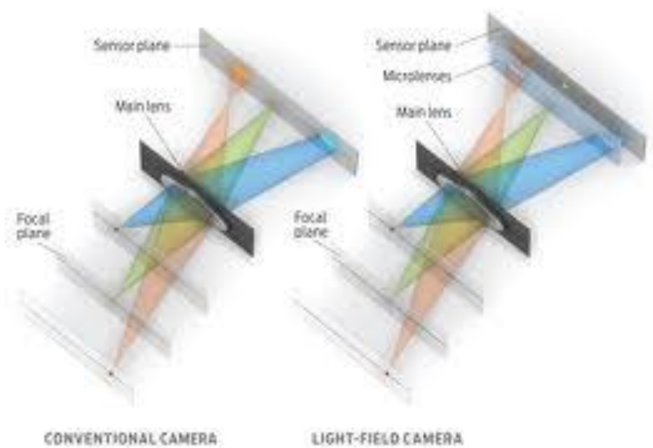
Structured light:
Project a known dot pattern with an IR transmitter (invisible to humans)

Infer depth from deformation to that pattern
depth from focus: Points far away are blurry
depth from stereo: Closer points are shifted

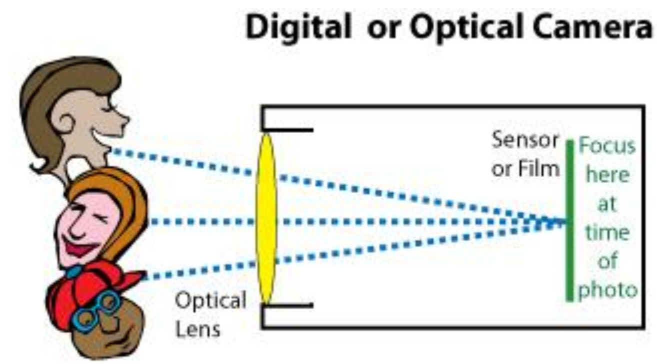
The Latest, Coolest...

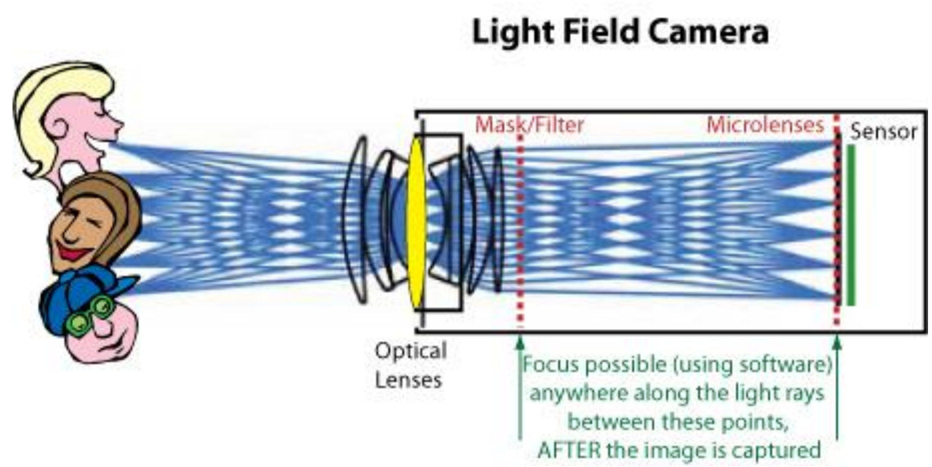
Light field camera (passive)

"Capture" the light going in every direction at every 3D point



ieee.org
4/12/2015





http://excelmathmike.blogspot.com

The Problem

- Mapping: What is the world around me (geometry, landmarks)
  - sense from various positions
  - integrate measurements to produce map
  - assumes perfect knowledge of position
- Localization: Where am I in the world (position wrt landmarks)
  - sense
  - relate sensor readings to a world model
  - compute location relative to model
  - assumes a perfect world model
- Together, these are SLAM (Simultaneous Localization and Mapping)
  - How can you localize without a map?
  - How can you map without localization?
- All localization, mapping or SLAM methods are based on updating a state:
  - What makes a state? Localization? Map? Both?
  - How certain is the state?

Representations for Bayesian Robot Localization

<u>Discrete approaches (’ 95)</u>
- Topological representation (’ 95)
  - uncertainty handling (POMDPs)
  - occas. global localization, recovery
- Grid-based, metric representation (’ 96)
  - global localization, recovery

<u>Kalman filters (late-80s?)</u>
- Gaussians
- approximately linear models
- position tracking

Robotics

AI

<u>Particle filters (’ 99)</u>
- sample-based representation
- global localization, recovery

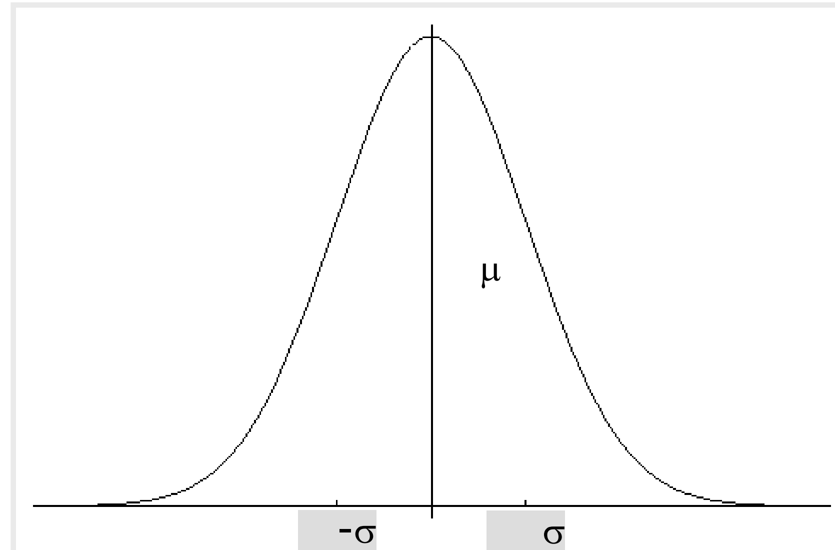
<u>Multi-hypothesis (’ 00)</u>
- multiple Kalman filters
- global localization, recovery

Background
Gaussian (or Normal) Distribution

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

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

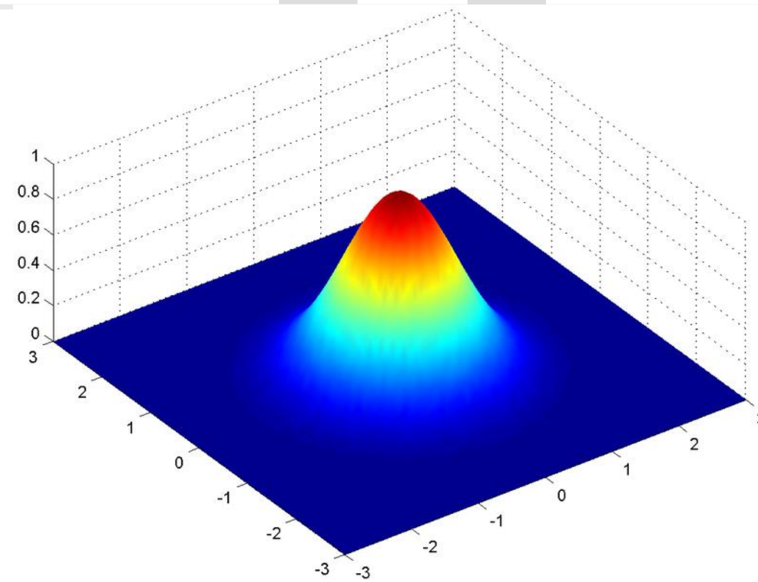
Univariate



$$p(\mathbf{x}) \sim N(u, \Sigma):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-u)^t \Sigma^{-1}(\mathbf{x}-u)}$$

Multivariate



Properties of Gaussians

$$\left.\begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow \quad Y \sim N(a\mu + b, a^2\sigma^2)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \sigma_1^2) \\ X_2 \sim N(\mu_2, \sigma_2^2) \end{array}\right\} \Rightarrow X_1 + X_2 \sim N\left(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\right)$$

- We stay in the “Gaussian world” as long as we start with Gaussians and perform only linear transformations.

- Same holds for multivariate Gaussians

Kalman Filter

- Seminal paper published in 1960
- Great web page at http://www.cs.unc.edu/~welch/kalman/
- Recursive solution for discrete linear filtering problems
  - A state $\boldsymbol{x} \in R^n$
  - A measurement $\boldsymbol{z} \in R^m$
  - Discrete (i.e. for time $t = 1, 2, 3, \ldots$)
  - Recursive (i.e. $\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1})$)
  - Linear system (i.e $\boldsymbol{x}_t = A\,\boldsymbol{x}_{t-1}$)
- The problem is defined by a *linear* process model

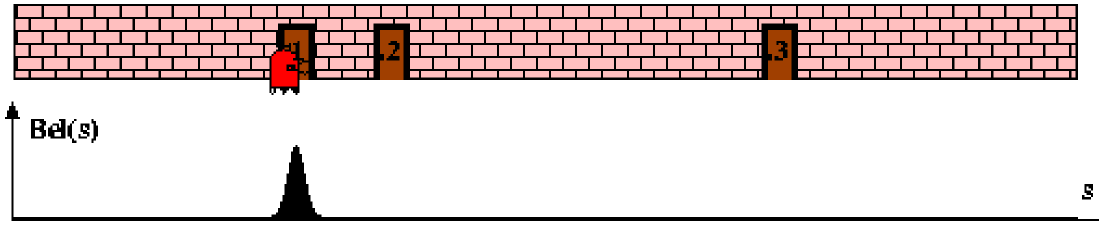
$$\boldsymbol{x}_t = \underbrace{A}_{\text{state transition}}\,\boldsymbol{x}_{t-1} + B\,\underbrace{\boldsymbol{u}_{t-1}}_{\text{control Input (optional)}} + \underbrace{\boldsymbol{w}_{t-1}}_{\text{Gaussian white noise}}$$

and a measurement *linear* model (with white Gaussian noise)

$$\boldsymbol{z}_t = \underbrace{H}_{\text{observation model}}\,\boldsymbol{x}_t + \underbrace{\boldsymbol{v}_t}_{\text{Gaussian white noise}}$$

Initial

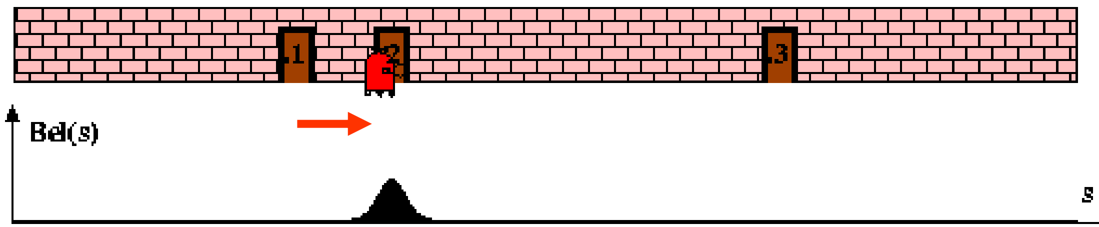
$$N(\hat{x}_t, \sigma_t)$$



Prediction

$$\hat{x}'_{t+1} = A\hat{x}_t + Bu_t$$

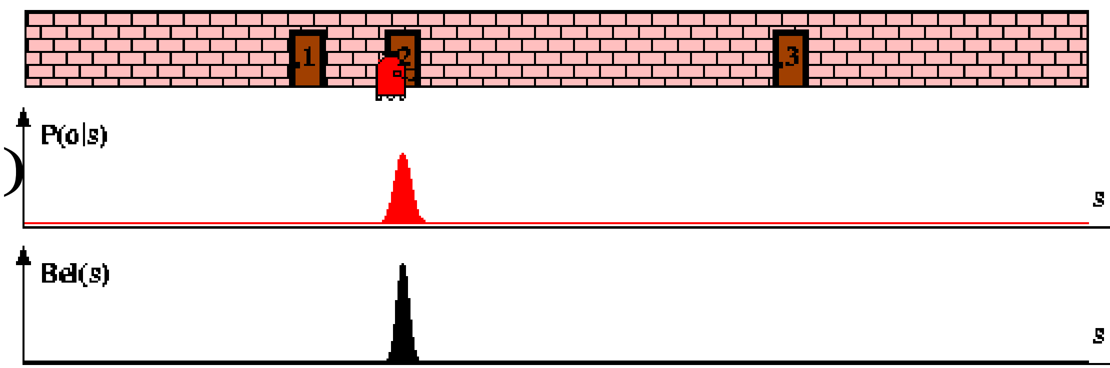
$$\sigma'_{t+1} = A\sigma_t A + Q$$



Correction

$$\hat{x}_{t+1} = \hat{x}'_{t+1} + K_{t+1}(z_{t+1} - H\hat{x}'_{t+1})$$

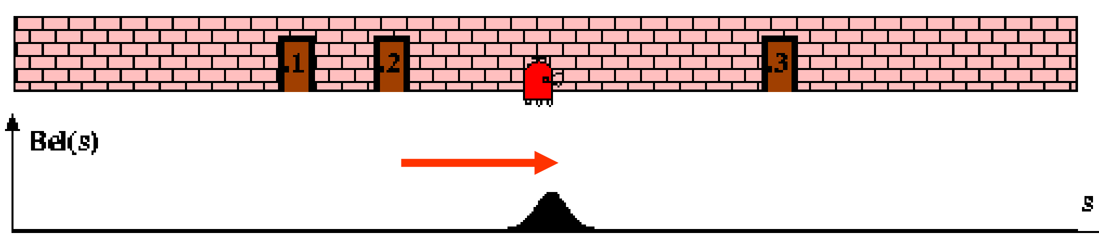
$$\sigma_{t+1} = (1 - K_{t+1}H)\sigma'_{t+1}$$



Prediction

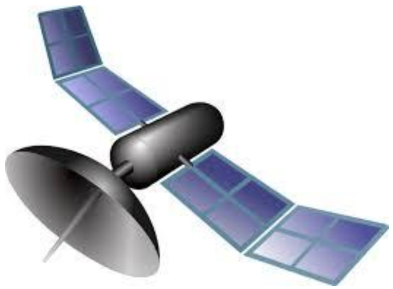
$$\hat{x}'_{t+2} = A\hat{x}_{t+1} + Bu_{t+1}$$

$$\sigma'_{t+2} = A\sigma_{t+1} A + Q$$



Example

- If we are given all the ingredients: $\boldsymbol{x}_{t\text{-}1}$, $\boldsymbol{z}_t$, A, H $\boldsymbol{w}_{t\text{-}1}$, $\boldsymbol{v}_{t\text{-}1}$ (and B and $\boldsymbol{u}_{t\text{-}1}$) what is the "optimal" $\boldsymbol{x}_t$?



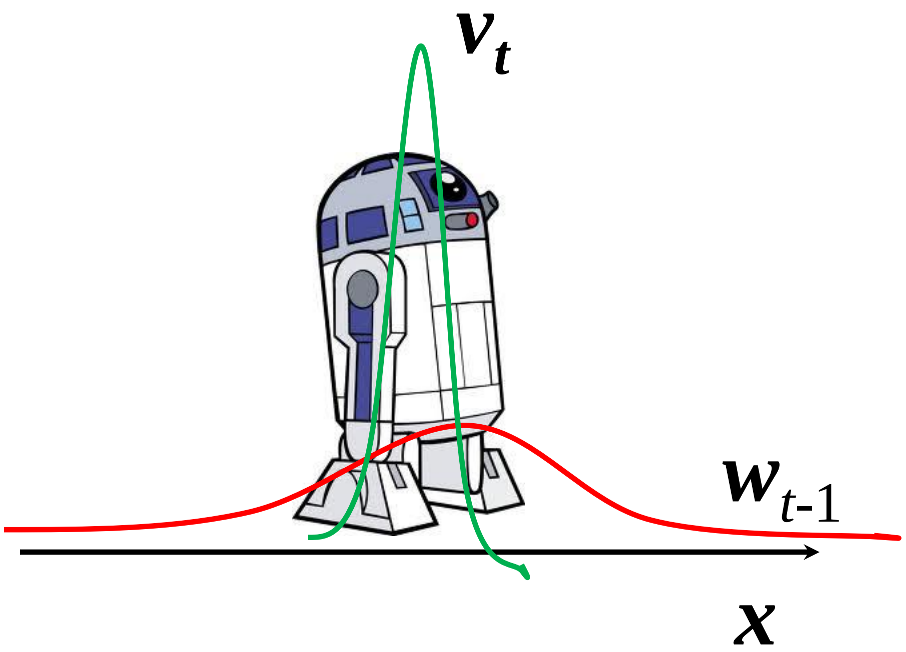
$$\boldsymbol{x}_t = A\,\boldsymbol{x}_{t\text{-}1} + B\,\boldsymbol{u}_{t\text{-}1} + \boldsymbol{w}_{t\text{-}1}$$

$$\boldsymbol{z}_t = H\,\boldsymbol{x}_t + \boldsymbol{v}_t$$

With

$$\boldsymbol{w}_{t\text{-}1} \sim N(\,0, Q\,)$$

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



What if

- Q is tiny and R is large?
- R is large and Q is tiny?
- Q and R are large?
- Q and R are tiny?

Kalman Filter

- *A priori* estimate $\hat{\mathbf{x}}'_t$ (prediction using process model) at step t
- *A posteriori* estimate $\hat{\mathbf{x}}_t$ (correction using measurement model) at step t

Compute *a posteriori* estimate as a linear combination of an *a priori* estimate and difference between the actual measurement and expected measurement

$$\hat{x}_t = \hat{x}'_t + K_t(z_t - H\hat{x}'_t)$$

What is K?

Gain or " blending factor" that adds a measurement innovation

Kalman Filter

Define *a priori* error between true state and *a priori* estimate

$$\boldsymbol{e}'_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}'_t$$

and its covariance as

$$\Sigma'_t = E(\boldsymbol{e}'_t \boldsymbol{e}'^T_t)$$

Define *a posteriori* error between true state and posterior estimate

$$\boldsymbol{e}_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t$$

and its covariance as

$$\Sigma_t = E(\boldsymbol{e}_t \boldsymbol{e}^T_t)$$

Then K that minimizes the a posteriori covariance is defined by

$$K_t = \Sigma'_t H^T (H \Sigma'_t H^T + R)^{-1}$$

Note that

- If $R \rightarrow 0$ then $K_t = H^{-1}$ (increase residual weight)
- If $\Sigma_t \rightarrow 0$ then $K_t = 0$ (decrease residual weight)

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}'_t + K \underbrace{(\boldsymbol{z}_t - H \hat{\boldsymbol{x}}'_t)}_{\text{residual}}$$

Kalman Filter

- Recipe:
  - Given

    $$\hat{\boldsymbol{x}}_0, \Sigma_0$$

  - Time update

    $$\hat{\boldsymbol{x}}_t' = A\hat{\boldsymbol{x}}_{t-1} + B\boldsymbol{u}_{t-1}$$

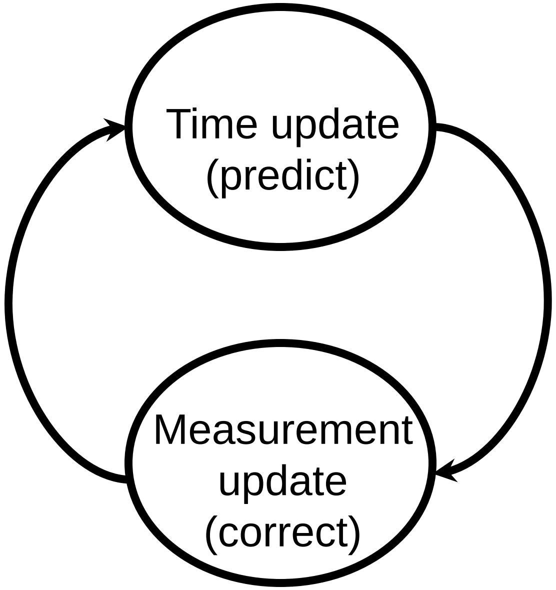
    $$\Sigma_t' = A\Sigma_{t-1}A^T + Q$$

  - Measurement update

    $$K_t = \Sigma_t' H^T (H\Sigma_t' H^T + R)^{-1}$$

    $$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_t' + K_t(\boldsymbol{z}_t - H\hat{\boldsymbol{x}}_t')$$

    $$\Sigma_t = (1 - K_t H)\Sigma_t'$$

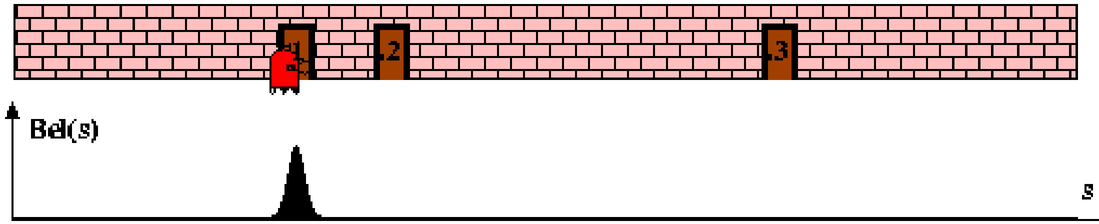


Some Examples

- Point moving on the line according to $f = m\,a$
  - state is position *and* velocity
  - input is force
  - sensing is position
- Point in the plane under Newtonian laws
- Non-holonomic kinematic system (no dynamics)
  - state is workspace configuration
  - input is velocity command
  - sensing could be direction and/or distance to beacons
- All dynamic systems are "open-loop" integration
  - Force → acceleration → velocity → position
- Role of sensing is to "close the loop" and pin down state

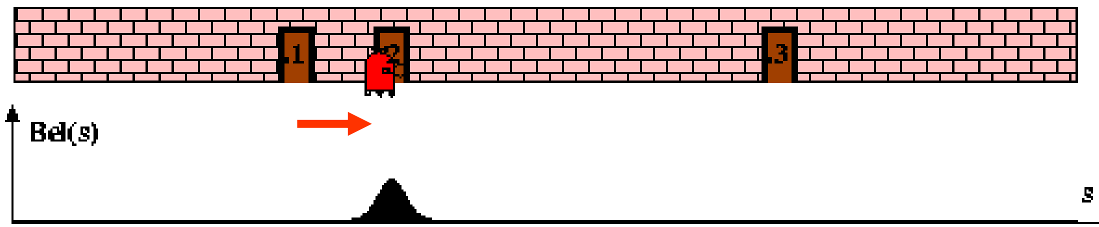
Initial

$$N(\hat{x}_t, \sigma_t)$$



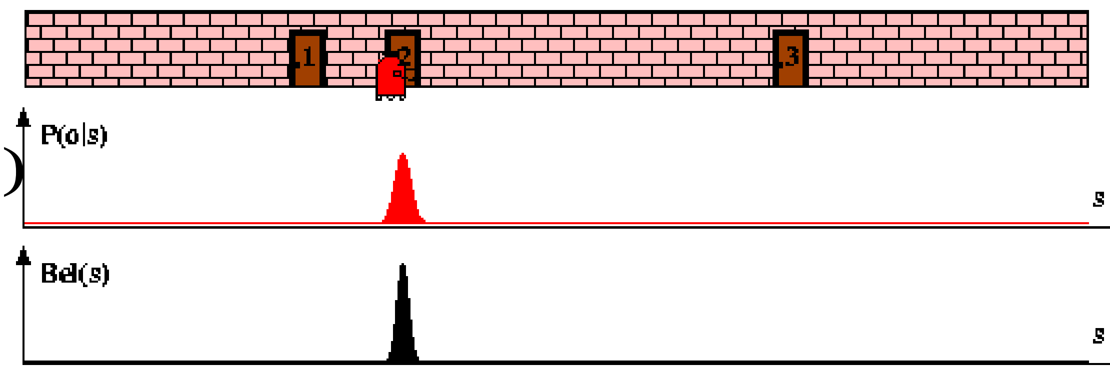
Prediction

$$\hat{x}'_{t+1} = A\hat{x}_t + Bu_t$$
$$\sigma'_{t+1} = A\sigma_t A + Q$$



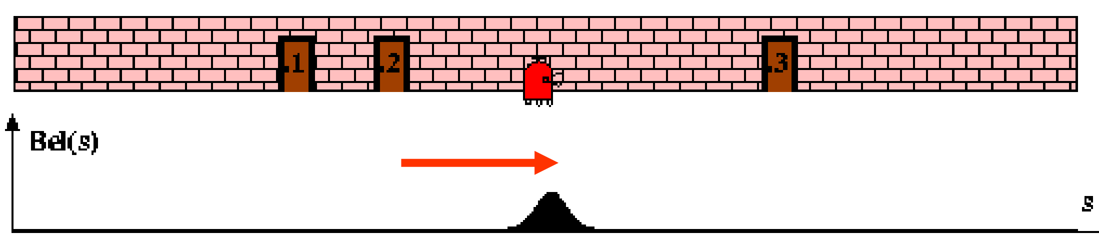
Correction

$$\hat{x}_{t+1} = \hat{x}'_{t+1} + K_{t+1}(z_{t+1} - H\hat{x}'_{t+1})$$
$$\sigma_{t+1} = (1 - K_{t+1}H)\sigma'_{t+1}$$

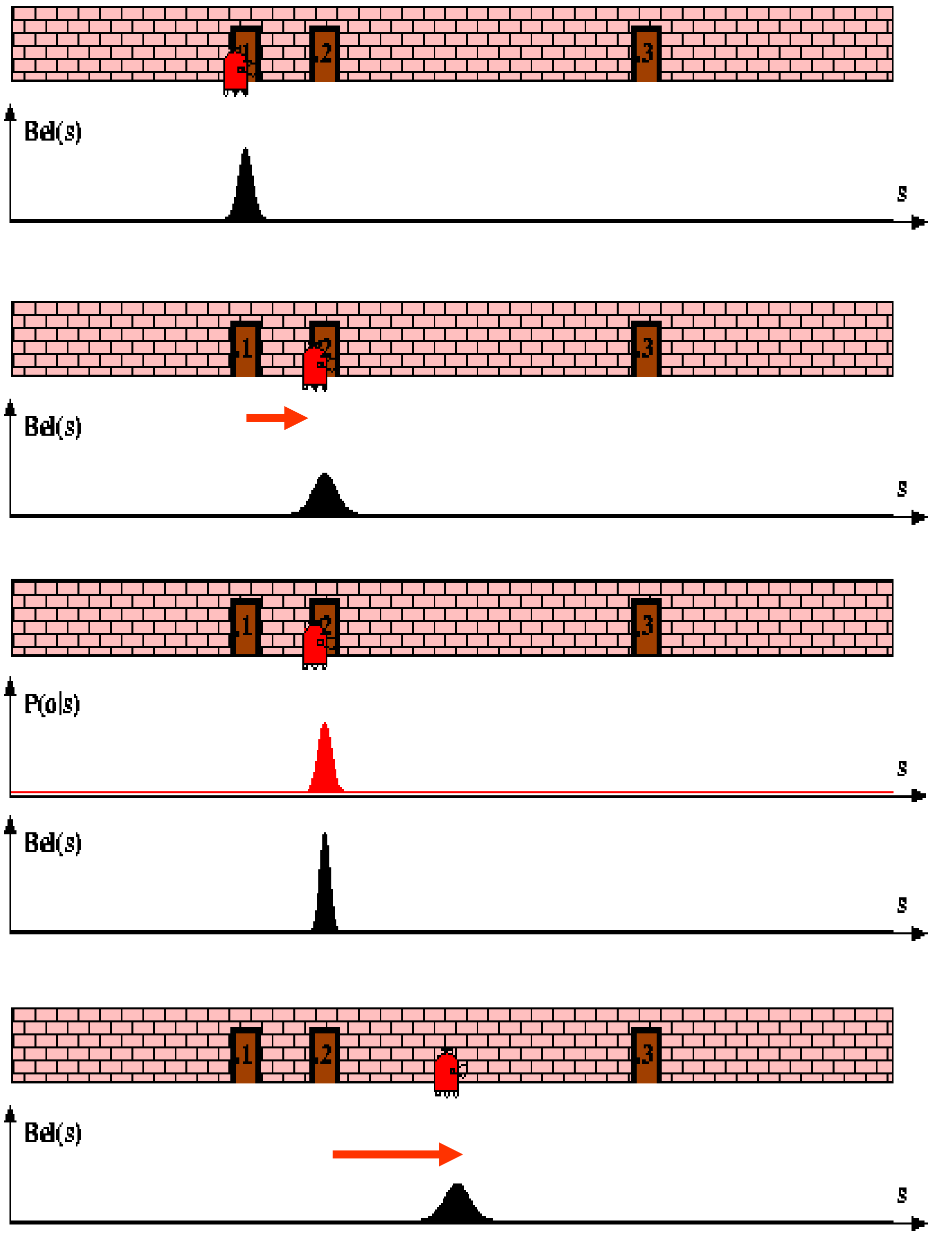


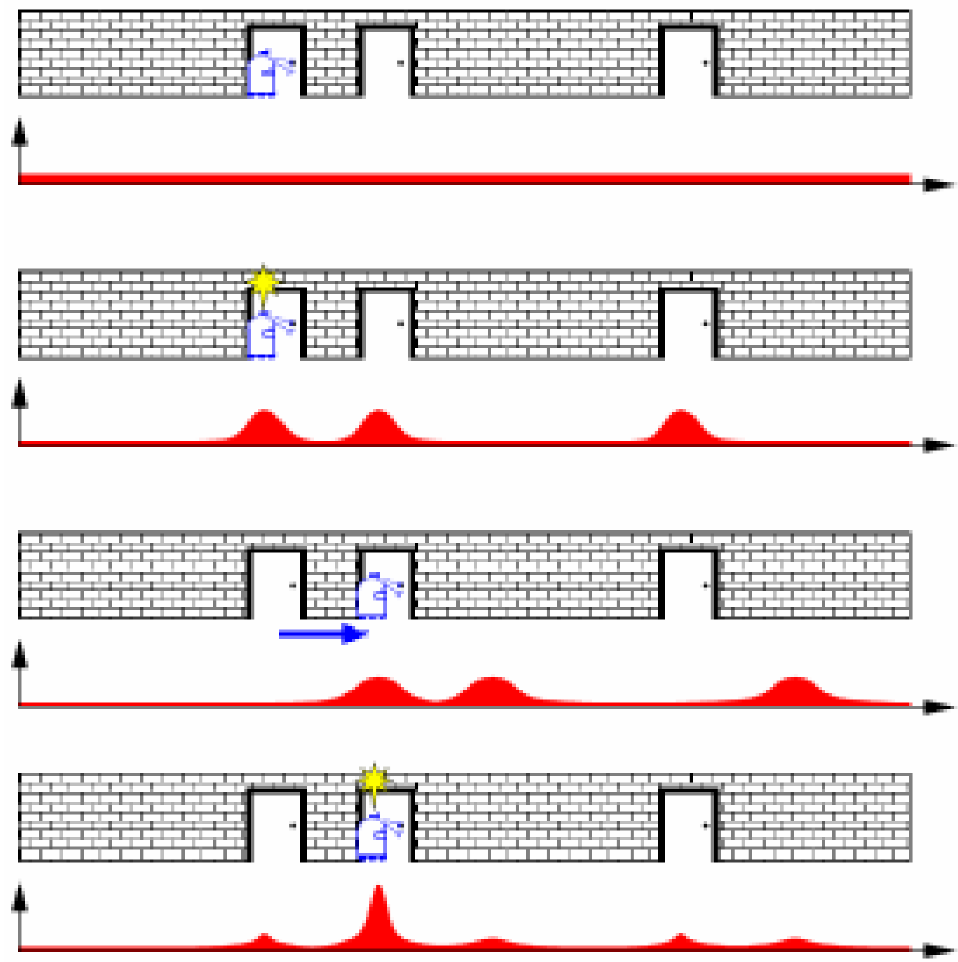
Prediction

$$\hat{x}'_{t+2} = A\hat{x}_{t+1} + Bu_{t+1}$$
$$\sigma'_{t+2} = A\sigma_{t+1} A + Q$$



Fully Observable vs Partially Observable





A concrete example

Process Model

$$\hat{x}'_t = \hat{x}_{t-1} + 1 \qquad \hat{x}'_t = A\hat{x}_{t-1} + Bu_{t-1}$$

$$\hat{x}_0 = 0$$

$$\sigma_0 = 1$$

$$v_t \sim N(0,1)$$

Observation Model

$$z_t = 2x_t \qquad z_t = Hx_t$$

$$w_t \sim N(0,2)$$

$$\hat{x}'_1 = 0 + 1 = 1 \qquad \hat{x}'_t = A\hat{x}_{t-1} + Bu_{t-1}$$

$$\sigma'_1 = 1 + 1 = 2 \qquad \Sigma'_t = A\Sigma_{t-1}A^T + Q$$

$$z_1 = 2.1$$

$$K_1 = 2 \times 2(2 \times 2 \times 2 + 2)^{-1} \qquad K_t = \Sigma'_t H^T (H\Sigma'_t H^T + R)^{-1}$$

$$K_1 = 0.4$$

$$\hat{x}_1 = 1 + 0.4(2.1 - 2 \times 1) \qquad \hat{x}_{t+1} = \hat{x}'_{t+1} + K_{t+1}(z_{t+1} - H\hat{x}'_{t+1})$$

$$\hat{x}_1 = 1.04$$

$$\sigma_1 = (1 - 0.4 \times 2)2 \qquad \Sigma_t = (I - K_t H)\Sigma'_t$$

$$\sigma_1 = 0.4$$

Kalman Filter for Dead Reckoning

- Robot moves along a straight line with state $\boldsymbol{x} = [\, p, v \,]^T$

  p: position

  v: velocity
- u is the input force applied to the robot

Newton's 2nd law $\dot{v} = \frac{u}{m}$ first order finite difference: $\frac{v_{t+1} - v_t}{\Delta t} = \frac{u_t}{m}$

$$\boldsymbol{x}_{t+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_t + \begin{bmatrix} 0 \\ \frac{\Delta t}{m} \end{bmatrix} u_t$$

$$x_{t+1} = \begin{bmatrix} p_{t+1} \\ v_{t+1} \end{bmatrix} = \begin{bmatrix} p_t + v_t \Delta t \\ v_t \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{u_t}{m} \Delta t \end{bmatrix}$$

Integrate velocity

Integrate acceleration

- Robot has velocity sensor

$$\boldsymbol{z}_t = H \boldsymbol{x}_t = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_t = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix}$$

The robot position is not measured

The measured velocity depends on the robot velocity (duh!)

Example

- Let plug some numbers

$m = 1$

$\Delta t = 0.1$

$\boldsymbol{v}_t \sim N(0, \mathrm{Q})$

$\boldsymbol{w}_t \sim N(0, \mathrm{R})$

$$\mathrm{Q} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$\mathrm{R} = \begin{bmatrix} 0 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Start at rest from the current position

$$x_0 = [0 \quad 0]^T$$

$$\Sigma_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\boldsymbol{x}_1' = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.1 \\ \hline 1 \end{bmatrix} 1$$

$$\boldsymbol{x}_1' = \begin{bmatrix} 0 \\ 0.1 \end{bmatrix}$$

prediction at t =1

$$\Sigma_1' = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}^T + \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$\Sigma_1' = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

prediction covariance at t =1

$$K_t = \Sigma_t' H^T (H \Sigma_t' H^T + R)^{-1}$$

$$K_1 = \Sigma_1' H^T \begin{bmatrix} 0 & 0 \\ 0 & 0.2 \end{bmatrix}^{-1}$$

Uh oh!
Matrix is singular.
The reason is that an infinite number of states can generate the same observation

$$\boldsymbol{z}_t = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_t$$

Observability

- If H does not provide a one-to-one mapping between the state and the measurement, then the system is *unobservable*

$$\mathbf{z}_t = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_t$$

In this case H is singular, such that many states can generate the same observation

Kalman Filter Limitations

- Assumptions:
  - Linear state dynamics
  - Observations linear in state
  - White Gaussian noise

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_{t-1} + \boldsymbol{w}_{t-1}$$
$$\boldsymbol{z}_t = H\boldsymbol{x}_t + \boldsymbol{v}_t$$

- What can we do if system is not linear?
  - Non-linear state dynamics
  - Non-linear observations

$$\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}, \boldsymbol{w}_{t-1})$$
$$\boldsymbol{z}_t = \boldsymbol{h}(\boldsymbol{x}_t, \boldsymbol{v}_t)$$

Linearize it!

$$\boldsymbol{x}_t \approx \tilde{\boldsymbol{x}}_t + A(\boldsymbol{x}_{t-1} - \hat{\boldsymbol{x}}_{t-1}) + W\boldsymbol{w}_{t-1}$$
$$\boldsymbol{z}_t \approx \tilde{\boldsymbol{z}}_t + H(\boldsymbol{x}_t - \tilde{\boldsymbol{x}}_t) + V\boldsymbol{v}_t$$

$$\tilde{\boldsymbol{x}}_t = \boldsymbol{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}, 0)$$
$$\tilde{\boldsymbol{z}}_t = \boldsymbol{h}(\tilde{\boldsymbol{x}}_t, 0)$$

Extended Kalman Filter

- Where A, H, W and V are Jacobians defined by

$$A(\boldsymbol{x}_t) = \begin{bmatrix} \frac{\partial \boldsymbol{f}_1(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{f}_1(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}_n(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{f}_n(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_n} \end{bmatrix}$$

$$W(\boldsymbol{x}_t) = \begin{bmatrix} \frac{\partial \boldsymbol{f}_1(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{w}_1} & \cdots & \frac{\partial \boldsymbol{f}_1(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{w}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}_n(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{w}_1} & \cdots & \frac{\partial \boldsymbol{f}_n(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{w}_n} \end{bmatrix}$$

$$V(\boldsymbol{x}_t) = \begin{bmatrix} \frac{\partial \boldsymbol{h}_1(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{v}_1} & \cdots & \frac{\partial \boldsymbol{h}_1(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{v}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{h}_n(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{v}_1} & \cdots & \frac{\partial \boldsymbol{h}_n(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{v}_n} \end{bmatrix} \qquad H(\boldsymbol{x}_t) = \begin{bmatrix} \frac{\partial \boldsymbol{h}_1(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{h}_1(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{h}_n(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{h}_n(\boldsymbol{x}_t, 0)}{\partial \boldsymbol{x}_n} \end{bmatrix}$$

Extended Kalman Filter

- Kalman Filter Recipe:
  - Given
    $\hat{x}_0, \Sigma_0$
  - Prediction
    $$\hat{\boldsymbol{x}}'_t = A\hat{\boldsymbol{x}}_{t-1} + B\boldsymbol{u}_{t-1}$$
    $$\Sigma'_t = A\Sigma_{t-1}A^T + Q$$
  - Measurement correction
    $$K_t = \Sigma'_t H^T (H\Sigma'_t H^T + R)^{-1}$$
    $$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}'_t + K(\boldsymbol{z}_t - H\hat{\boldsymbol{x}}'_t)$$
    $$\Sigma_{\mathrm{t}} = (I - K_t H)\Sigma'_t$$

- Extended Kalman Filter Recipe:
  - Given
    $\hat{x}_0, \Sigma_0$
  - Prediction
    $$\hat{\boldsymbol{x}}'_t = \boldsymbol{f}(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_{t-1}, 0)$$
    $$\Sigma'_t = A_t\Sigma_{t-1}A_t^T + W_t Q W_t^T$$
  - Measurement correction
    $$K_t = \Sigma'_t H_t^T (H_t\Sigma'_t H_t^T + V_t R V_t^T)^{-1}$$
    $$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}'_t + K_t(\boldsymbol{z}_t - \boldsymbol{h}(\hat{\boldsymbol{x}}'_t, 0))$$
    $$\Sigma_{\mathrm{t}} = (I - K_t H_t)\Sigma'_t$$

EKF for Range-Bearing Localization

- State $\boldsymbol{s}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}$ position and orientation
- Input $\boldsymbol{u}_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}$ forward and rotational velocity
- Process model

$$\boldsymbol{f}(\boldsymbol{s}_{t-1}, \boldsymbol{u}_{t-1}\boldsymbol{w}_{t-1}) = \begin{bmatrix} x_{t-1} + \Delta t\, v_{t-1} \cos\theta_{t-1} \\ y_{t-1} + \Delta t\, v_{t-1} \sin\theta_{t-1} \\ \theta_{t-1} + \Delta t\, \omega_{t-1} \end{bmatrix} + \begin{bmatrix} w_{x_t} \\ w_{y_t} \\ w_{\theta_t} \end{bmatrix}$$

Se 2005

- Given a map, the robot sees N landmarks with coordinates

$$\boldsymbol{l}_1 = [x_{l_1} \quad y_{l_1}]^T, \cdots, \boldsymbol{l}_N = [x_{l_N} \quad y_{l_N}]^T$$

$$\boldsymbol{z}_t = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{s}_t, \boldsymbol{v}_1) \\ \vdots \\ \boldsymbol{h}_N(\boldsymbol{s}_t, \boldsymbol{v}_N) \end{bmatrix} \qquad \boldsymbol{h}_i(\boldsymbol{s}_t, \boldsymbol{v}_t) = \begin{bmatrix} \sqrt{\left(x_t - x_{l_i}\right)^2 + \left(y_t - y_{l_i}\right)^2} \\ \tan^{-1}\dfrac{y_t - y_{l_i}}{x_t - x_{l_i}} - \theta_t \end{bmatrix} + \begin{bmatrix} v_r \\ v_b \end{bmatrix}$$

Linearize Process Model

$$\boldsymbol{f}(\boldsymbol{s}_{t-1}, \boldsymbol{u}_{t-1} \boldsymbol{w}_{t-1}) = \begin{bmatrix} x_{t-1} + \Delta t\, v_{t-1} \cos\theta_{t-1} \\ y_{t-1} + \Delta t\, v_{t-1} \sin\theta_{t-1} \\ \theta_{t-1} + \Delta t\, \omega_{t-1} \end{bmatrix} + \begin{bmatrix} w_{x_t} \\ w_{y_t} \\ w_{\theta_t} \end{bmatrix}$$

$$A(\boldsymbol{x}_t) = \begin{bmatrix} \frac{\partial \boldsymbol{f}_1(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{f}_1(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{f}_n(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_1} & \cdots & \frac{\partial \boldsymbol{f}_n(\boldsymbol{x}_t, \boldsymbol{u}_t, 0)}{\partial \boldsymbol{x}_n} \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 & -\Delta t\, v_{t-1} \sin\theta_{t-1} \\ 0 & 1 & -\Delta t\, v_{t-1} \cos\theta_{t-1} \\ 0 & 0 & 1 \end{bmatrix}$$

Linearize Observation Model

$$\boldsymbol{z}_t = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{s}_t, \boldsymbol{v}_1) \\ \vdots \\ \boldsymbol{h}_N(\boldsymbol{s}_t, \boldsymbol{v}_N) \end{bmatrix} \qquad \boldsymbol{h}_i(\boldsymbol{s}_t, \boldsymbol{v}_t) = \begin{bmatrix} \sqrt{\left(x_t - x_{l_i}\right)^2 + \left(y_t - y_{l_i}\right)^2} \\ \tan^{-1}\dfrac{y_t - y_{l_i}}{x_t - x_{l_i}} - \theta_t \end{bmatrix} + \begin{bmatrix} v_r \\ v_b \end{bmatrix}$$

$$H(\boldsymbol{s}_t) = \begin{bmatrix} \dfrac{\partial \boldsymbol{h}_1(\boldsymbol{s}_t, 0)}{\partial \boldsymbol{s}_1} & \cdots & \dfrac{\partial \boldsymbol{h}_1(\boldsymbol{s}_t, 0)}{\partial \boldsymbol{s}_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \boldsymbol{h}_n(\boldsymbol{s}_t, 0)}{\partial \boldsymbol{s}_1} & \cdots & \dfrac{\partial \boldsymbol{h}_n(\boldsymbol{s}_t, 0)}{\partial \boldsymbol{s}_n} \end{bmatrix}$$

$$H_i(k+1, j) = \begin{bmatrix} \frac{(\hat{x}_r(k+1|k) - x_{\ell j})}{\sqrt{(\hat{x}_r(k+1|k) - x_{\ell j})^2 + (\hat{y}_r(k+1|k) - y_{\ell j})^2}} & \frac{(\hat{y}_r(k+1|k) - y_{\ell j})}{\sqrt{(\hat{x}_r(k+1|k) - x_{\ell j})^2 + (\hat{y}_r(k+1|k) - y_{\ell j})^2}} & 0 \\ \frac{-(\hat{y}_r(k+1|k) - y_{\ell j})}{1 + \left(\frac{\hat{y}_r(k+1|k) - y_{\ell j}}{\hat{x}_r(k+1|k) - x_{\ell j}}\right)^2 \left(\hat{x}_r(k+1|k) - x_{\ell j}\right)^2} & \frac{1}{1 + \left(\frac{\hat{y}_r(k+1|k) - y_{\ell j}}{\hat{x}_r(k+1|k) - x_{\ell j}}\right)^2 (\hat{x}_r(k+1|k) - x_{\ell j})} & -1 \end{bmatrix}$$

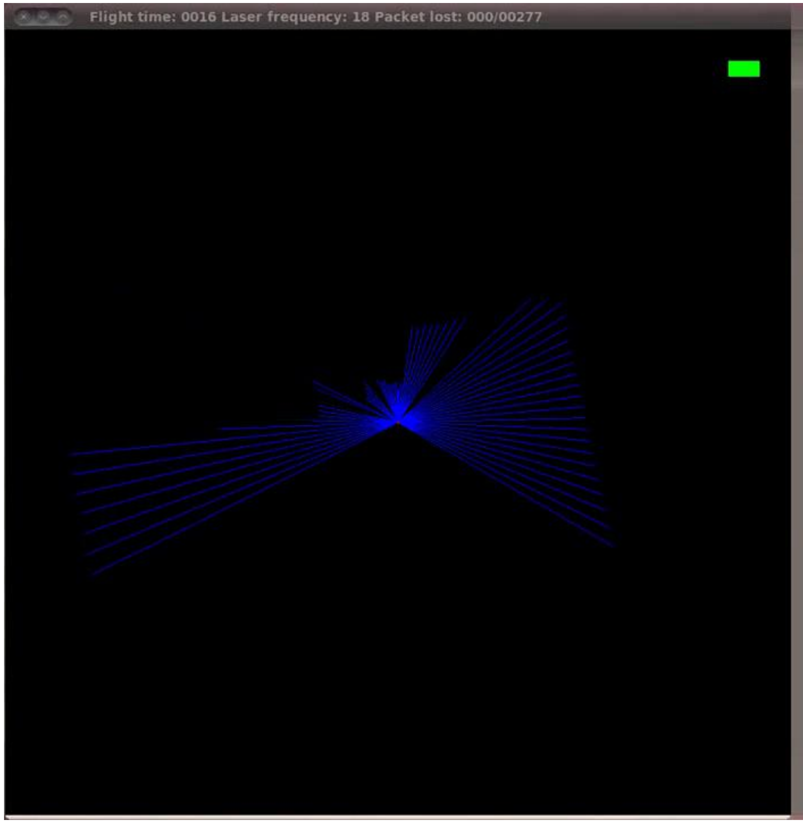
Data Association

- From observation model, we have an expected

$$z_t = \begin{bmatrix} h_1(s_t, v_1) \\ \vdots \\ h_N(s_t, v_N) \end{bmatrix}$$

$$\hat{x}_t = \hat{x}'_t + K_t(z_t - h(\hat{x}'_t, 0))$$

- So if we have N landmarks $\boldsymbol{l}_1, \ldots, \boldsymbol{l}_N$ and we are given a scan $\boldsymbol{z}_t$, how do associate each landmark to a scan observation?
- Given an observed landmark, we can do
  - Nearest neighbor
  - Mahalanobis distance
  - Probabilistic Data Association Filter (PDAF)



Xiaolei Hou

Pick the best landmark or, if it is too “different” create a new landmark $\boldsymbol{l}_{N+1}$

From Localization to Mapping

- For us, the landmarks have been a known quantity (we have a map with the coordinates of the landmarks), but landmarks are not part of the state

- Two choices:
  - Make the state the location of the landmarks relative to the robot (I also know exactly where I am …)
    - - No notion of location relative to past history
    - - No fixed reference for landmarks
  - Make the state the robot location now (relative to where we started) plus landmark locations
    - + Landmarks now have fixed location
    - - Knowledge of my location slowly degrades (but this is inevitable …)

Kalman Filters and SLAM

- Localization: state is the location of the robot
- Mapping: state is the location of 2D landmarks
- SLAM: state combines both
- If the state is $\boldsymbol{s}_t = \begin{bmatrix} x_t & y_t & \theta_t & l_{1_t}^T & \cdots & l_{N_t}^T \end{bmatrix}^T$
  then we can write a linear observation system
  - note that if we don't have some fixed landmarks, our system is *unobservable* (we can't fully determine all unknown quantities)
- Covariance Σ is represented by

http://ais.informatik.uni-freiburg.de

$$\underbrace{\begin{pmatrix}
\sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\
\sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\
\sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\
\sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\
\sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\
\sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}}
\end{pmatrix}}$$

Step 1: EKF Range Bearing SLAM State Update

- State $\boldsymbol{s}_t = \begin{bmatrix} x_t & y_t & \theta_t & l_{1_t}^T & \cdots & l_{N_t}^T \end{bmatrix}^T$ position and orientation and landmarks
- Input $\boldsymbol{u}_t = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix}$ forward and rotational velocity
- The process model for localization is

$$s_t^{'} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta t v_{t-1} \cos\theta_{t-1} \\ \Delta t v_{t-1} \sin\theta_{t-1} \\ \Delta t \omega_{t-1} \end{bmatrix}$$

This model is augmented for 2N+3 dimensions to accommodate landmarks. This results in the process equation

$$\begin{bmatrix} x'_{t-1} \\ y'_{t-1} \\ \theta'_{t-1} \\ l'_{1,t-1} \\ \vdots \\ l'_{N,t-1} \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \\ l_{1,t-1} \\ \vdots \\ l_{N,t-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta t v_{t-1} \cos\theta_{t-1} \\ \Delta t v_{t-1} \sin\theta_{t-1} \\ \Delta t \omega_{t-1} \end{bmatrix}$$

Step 2: EKF Range Bearing SLAM Covariance Update

Step 1b: Update the covariance matrix. The function **f**(***s***,***u***,***w***) only affects the robot's location and not the landmarks

$$\Sigma_t' = A_t \Sigma_{t-1} A_t^T + W_t Q W_t^T$$

Jacobian of the process model

$$A = \begin{bmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \theta} & \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial \theta} & 0 \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \theta} & \\ & 0 & & I \end{bmatrix}$$

Jacobian of the robot motion

2Nx2N identity

The motion of the robot does not affect the coordinates of the landmarks

Step 3: EKF Range Bearing SLAM Correction Gain

$$K_t = \Sigma_t' H_t^T (H_t \Sigma_t' H_t^T + V_t R V_t^T)^{-1}$$

$$\boldsymbol{h}_i(\boldsymbol{s}_t, \boldsymbol{v}_t) = \begin{bmatrix} \sqrt{(x_t - x_{l_i})^2 + (y_t - y_{l_i})^2} \\ \tan^{-1} \dfrac{y_t - y_{l_i}}{x_t - x_{l_i}} - \theta_t \end{bmatrix} + \begin{bmatrix} v_r \\ v_b \end{bmatrix}$$

Compute the Jacobian H_i of each $\boldsymbol{h}_i$ and then stack them into one big matrix H. Note that $\boldsymbol{h}_i$ only depends on 5 variables: x_t, y_t, θ_t, x_{li}, y_{li}

Need to be in the correct columns of H

$$H_i = \begin{bmatrix} \frac{\partial y_i}{\partial x_r} \\ \frac{\partial y_i}{\partial y_r} \end{bmatrix} =$$

$$\begin{bmatrix} \frac{-x_{\ell i}(k)+x_r(k)}{\rho_i} & \frac{-y_{\ell i}(k)+y_r(k)}{\rho_i} & 0 & \cdots & \frac{x_{\ell i}(k)-x_r(k)}{\rho_i} & \frac{y_{\ell i}(k)-y_r(k)}{\rho_i} & \cdots \\ \frac{y_{\ell i}(k)-y_r(k)}{\rho_i^2} & \frac{-x_{\ell i}(k)+x_r(k)}{\rho_i^2} & -1 & \cdots & \frac{-y_{\ell i}(k)+y_r(k)}{\rho_i^2} & \frac{x_{\ell i}(k)-x_r(k)}{\rho_i^2} & \cdots \end{bmatrix}$$

ρ_i is the range of the landmark

Step 4: EKF Range Bearing SLAM Measurement

$$\boldsymbol{z}_t = \begin{bmatrix} \boldsymbol{h}_1(\boldsymbol{s}_t, \boldsymbol{v}_1) \\ \vdots \\ \boldsymbol{h}_N(\boldsymbol{s}_t, \boldsymbol{v}_N) \end{bmatrix} \Longleftrightarrow \boldsymbol{h}_i(\boldsymbol{s}_t, \boldsymbol{v}_t) = \begin{bmatrix} \sqrt{\left(x_t - x_{l_i}\right)^2 + \left(y_t - y_{l_i}\right)^2} \\ \tan^{-1}\dfrac{y_t - y_{l_i}}{x_t - x_{l_i}} - \theta_t \end{bmatrix} + \begin{bmatrix} v_r \\ v_b \end{bmatrix}$$

- Observe N landmarks $z_t^i = \begin{bmatrix} r_t^i & \phi_t^i \end{bmatrix}$
- Must have data association

  Which measured landmark corresponds to $\boldsymbol{h}_i$?

  If $\boldsymbol{s}_t$ contains the coordinates of N landmarks in the map, $\boldsymbol{h}_i$ predicts the measurement of each landmark

  Must figure which measured landmark corresponds to $\boldsymbol{h}_i$.

  - Nearest neighbor
  - Probabilistic Data Association Filter (PDAF)
  - If using visual landmarks use visual descriptors to match landmarks

  If the measurement does not correspond to any predicted observation, then initialize and add the landmark to the map

$$\begin{bmatrix} l_x \\ l_y \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} r_t^i \cos(\phi_t^i + \theta_t) \\ r_t^i \sin(\phi_t^i + \theta_t) \end{bmatrix}$$

Step 5: EKF Range Bearing SLAM Correction Update

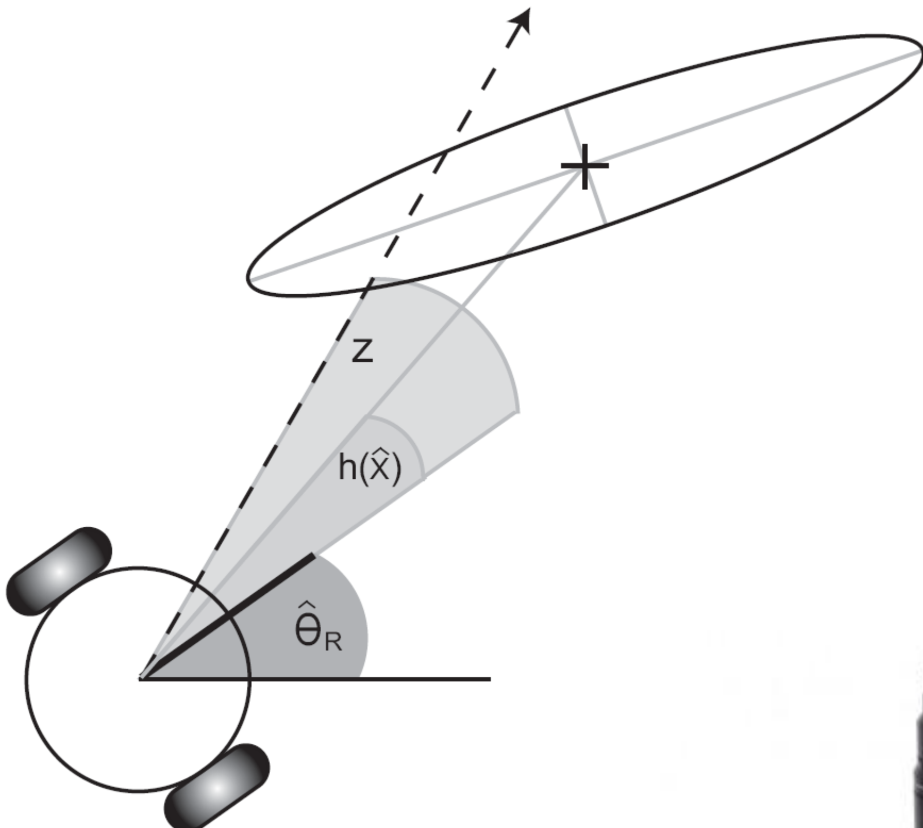
- From K and H update the posterior state estimate

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}'_t + K_t(\boldsymbol{z}_t - \boldsymbol{h}(\hat{\boldsymbol{x}}'_t, 0))$$

$$\Sigma_{\mathrm{t}} = (I - K_t H_t)\Sigma'_t$$

Tada! And we are done!

Bearing-Only SLAM



Tully 2008

Lets assume one landmark for now

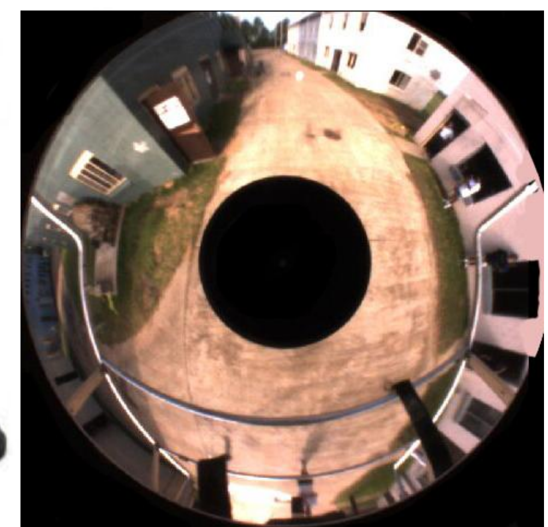
$$s = \begin{bmatrix} x & y & \theta & l_x & l_y \end{bmatrix}^T$$

$$h(s) = \tan^{-1}\left(\frac{l_y - y}{x_l - x}\right) - \theta$$

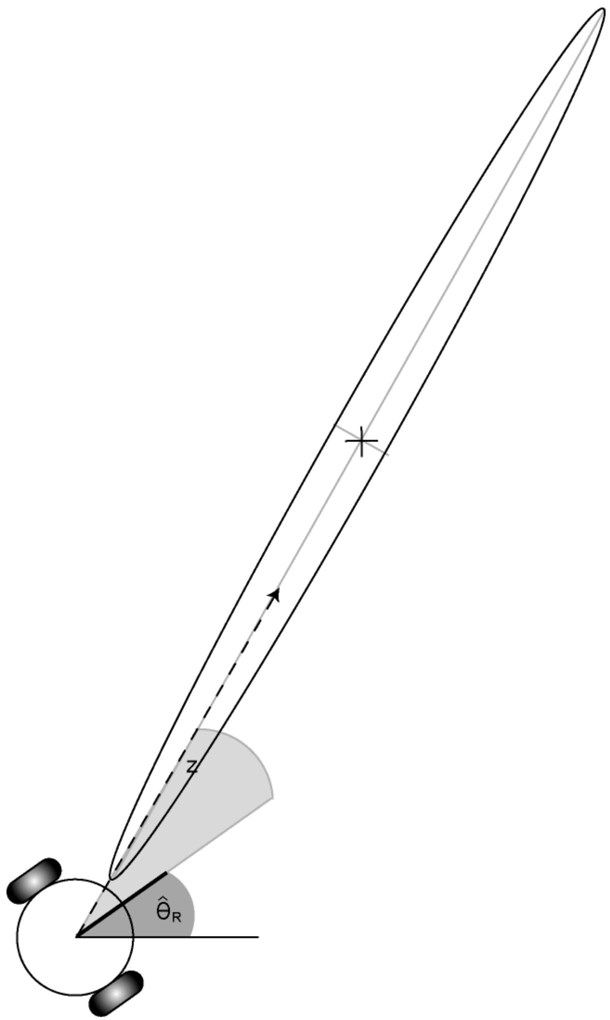
$$z = h(s)$$

www-robotics.usc.edu

Often use omni-directional sensor



Why Bearing-Only SLAM is Challenging



Tully 2008

• We cannot estimate the landmark location with one measurement

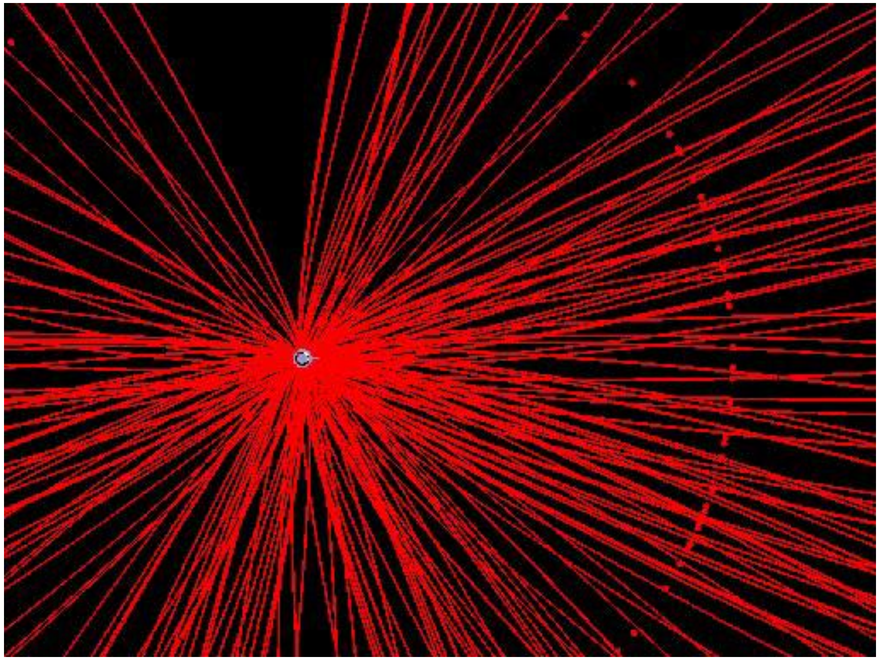
• We must guess the range and initialize with a large covariance due to the lack of range information

• The location is very uncertain and difficult to resolve with low parallax measurements

• The measurement model is very nonlinear, which breaks conventional filtering techniques

Bearing-Only SLAM with EKF

$$K_t = \Sigma_t' H_t^T (H_t \Sigma_t' H_t^T + V_t R V_t^T)^{-1}$$



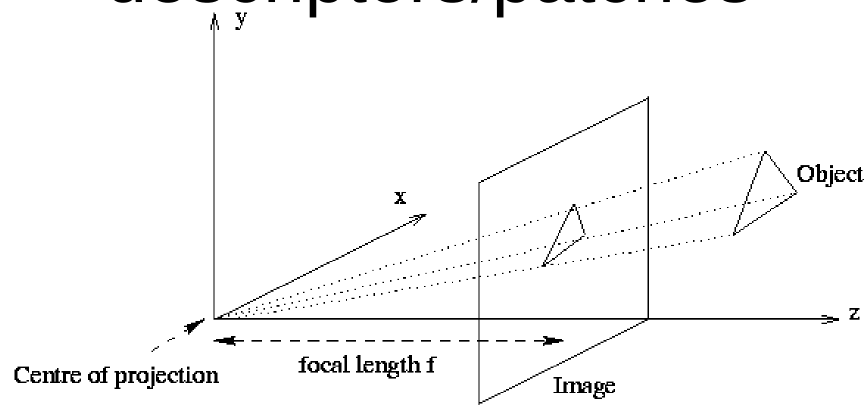
http://www.pracsyslab.org

- EKF uses the standard Kalman update
- The Kalman gain is computed through a linearization about the current estimate
- The result diverges
- Very dependent on the initialization “guess” of landmarks

Mono SLAM

Robot Vision, Imperial College

- A visual landmark with a single camera does not provide range
- Data association is given by tracking or matching visual descriptors/patches

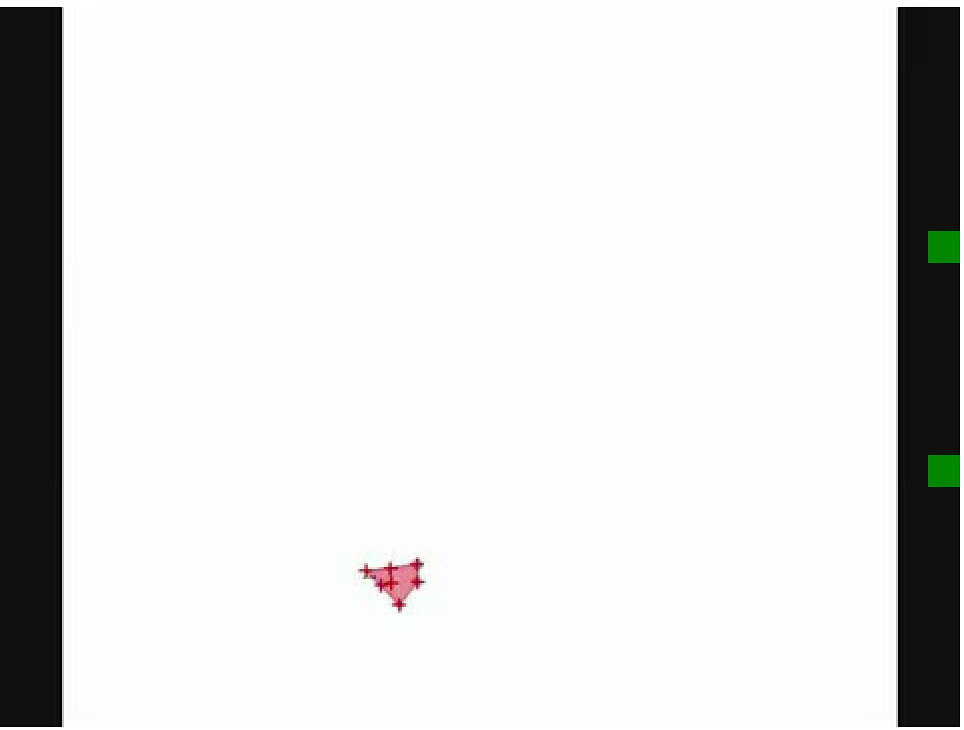


http://homepages.inf.ed.ac.uk/

Experimental Results – The Victoria Park Dataset

• A well studied benchmark dataset used in many other SLAM publications

• We simply ignored all of the range values provided with each landmark measurement

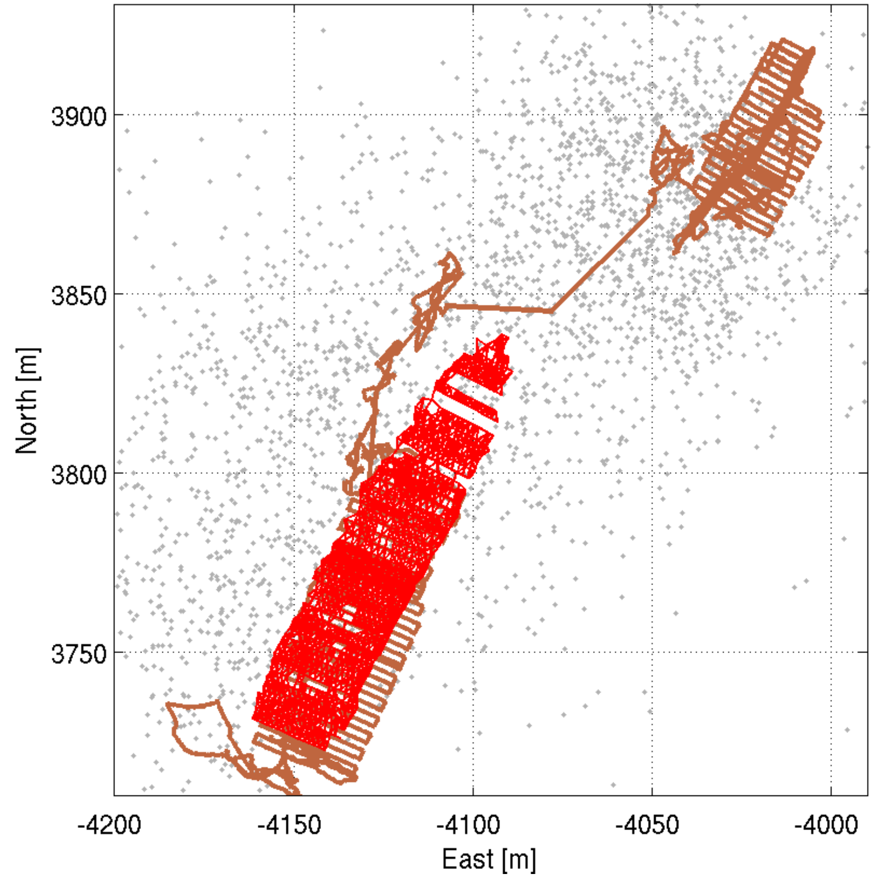


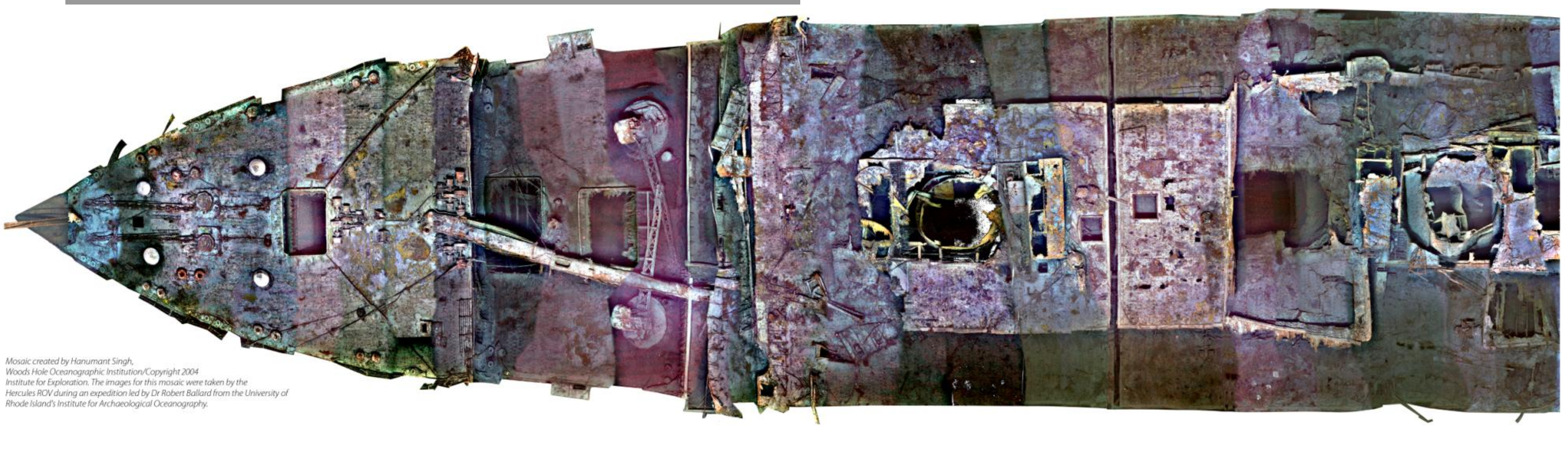
Navigation: RMS Titanic

Leonard & Eustice

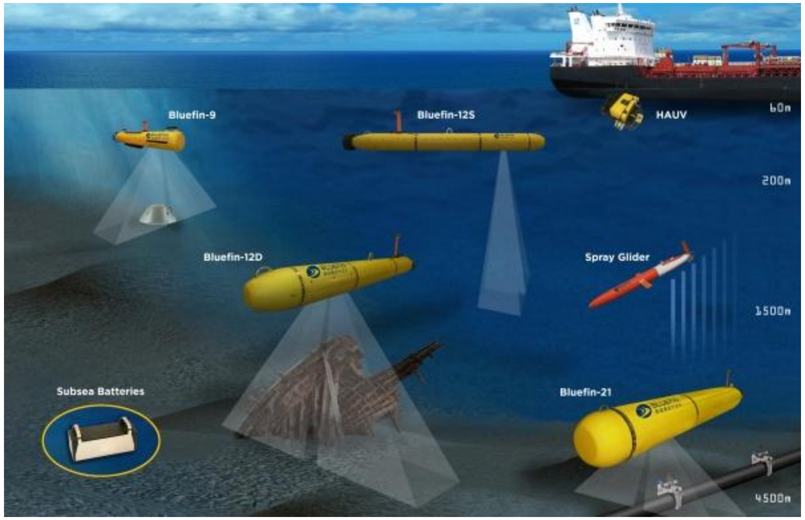
- EKF-based system
- 866 images
- 3494 camera constraints
- Path length 3.1km 2D / 3.4km 3D
- Convex hull > 3100m^2
- 344 min. data / 39 min. ESDF*

*excludes image registration time





Search of Flight 370



Summary

- Basic system modeling ideas
- Kalman filter as an estimation method from a system model
- Linearization as a way of attacking a wider variety of problems
- Mapping localization and mapping into EKF
- Extensions for managing landmark matching and not-well-constrained systems.