Univerza v Ljubljani

Advanced computer vision methods
Tracking by Recursive Bayes Filters
Part II: Kalman filter

Matej Kristan

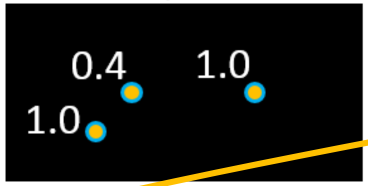
Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

Previously at ACVM…

- Tracking as state estimation

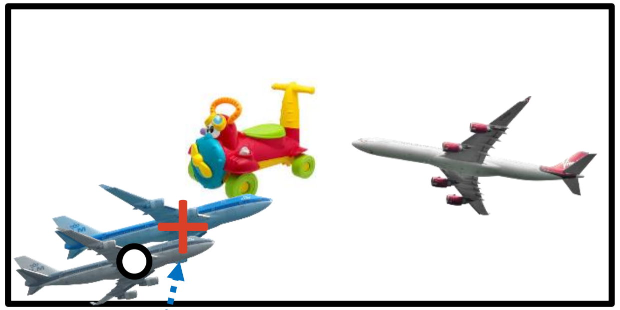
$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}$$

A noisy detector



Observe a scene at *k-1*

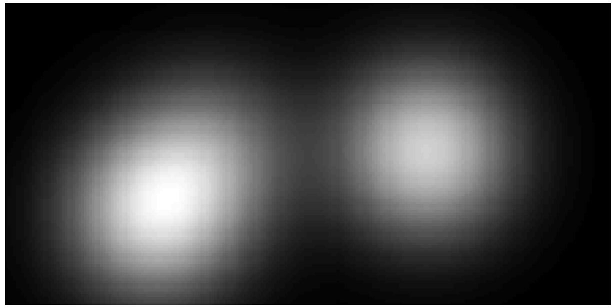
Observe a scene at *k*



Maximum
a posteriori state!

- Encode state info by pdfs

likelihood

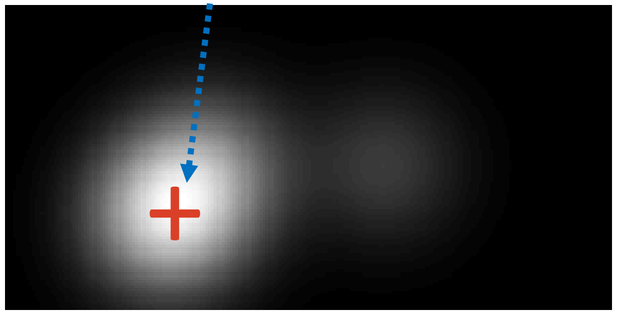


*

Predicted prior knowledge (pdf)

∝

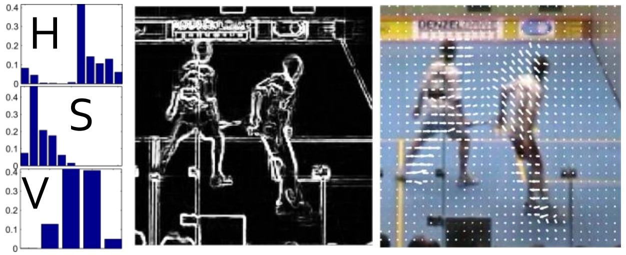
Posterior pdf



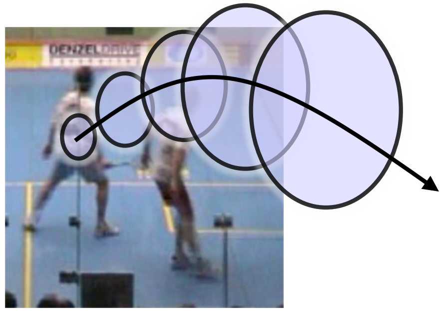
Previously at ACVM ...

State definition

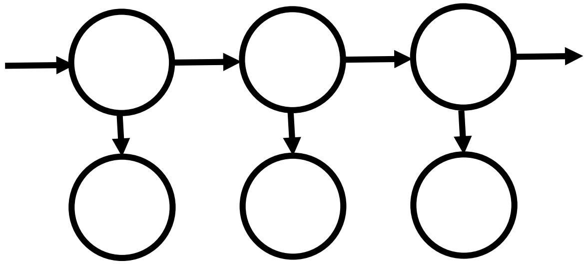
Observation model



Dynamic model

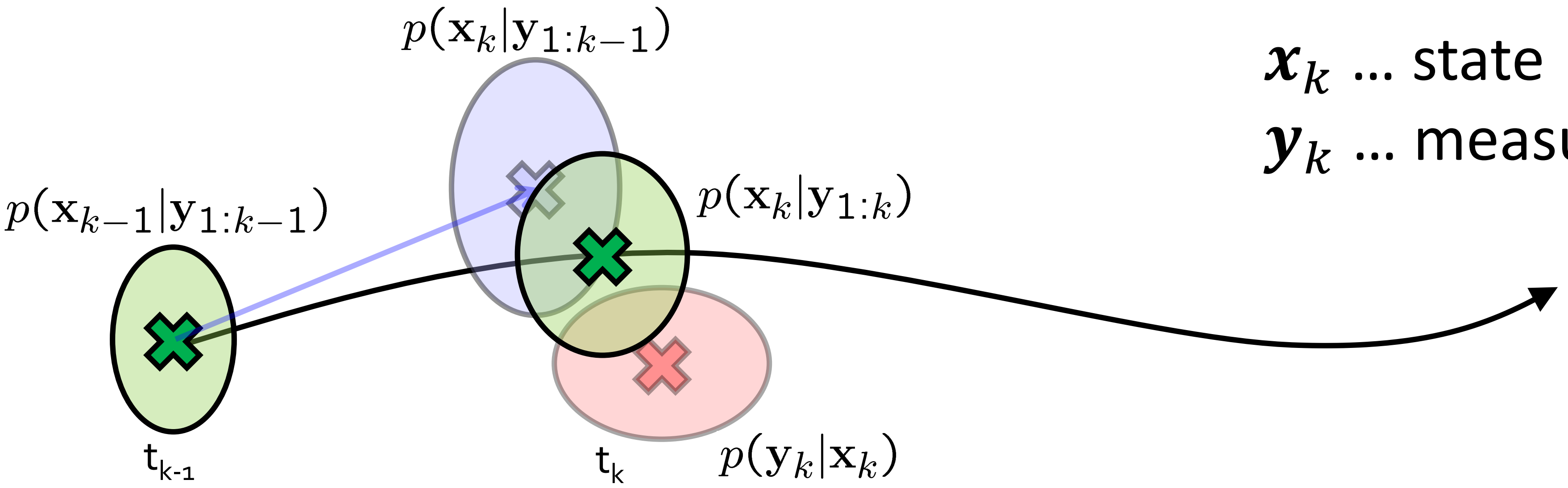


Inference



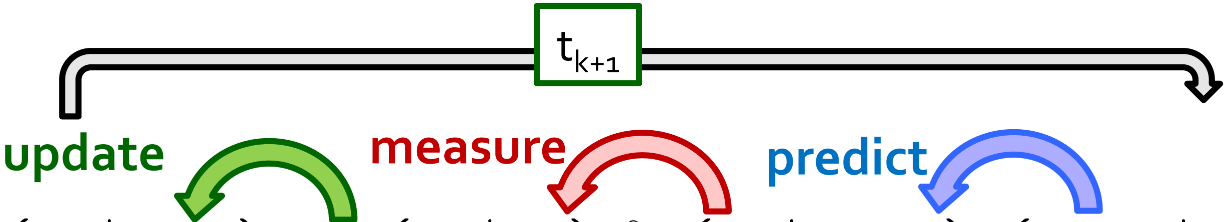
Previously at ACVM …

- At each time-step estimate the posterior:



$\boldsymbol{x}_k$ … state

$\boldsymbol{y}_k$ … measurement



$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})d\mathbf{x}_{k-1}$$

posterior estimate | Observation model | motion model | posterior at k-1

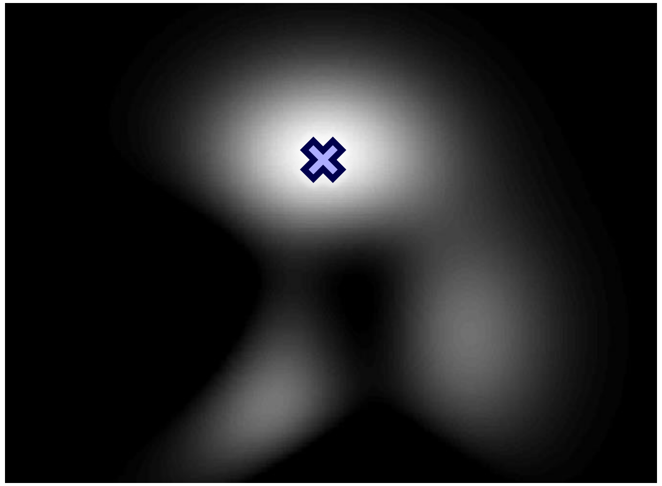
How to interpret the posterior?

- Face localization

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) d\mathbf{x}_{k-1}$$

current input image

current posterior



$p(\mathbf{x}_k|\mathbf{y}_{1:k})$

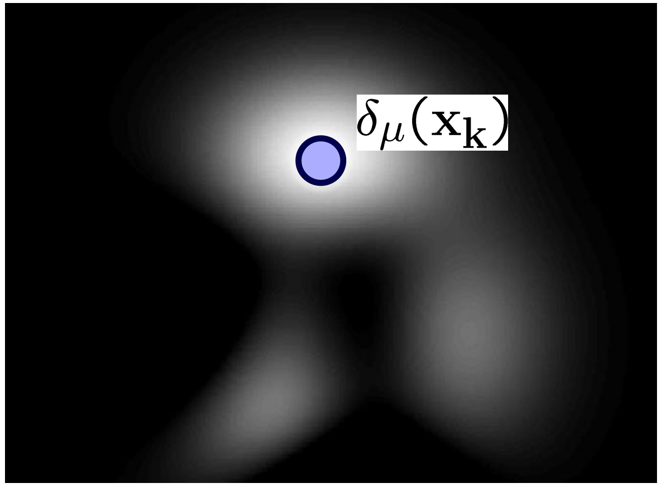
- Indicates likely and less likely positions of a tracked face.
- For example, get most probable position (maximum a posteriori)

Analytic representations

- A single point : Dirac-delta

current input image

current posterior



$p(\mathbf{x}_k|\mathbf{y}_{1:k})$

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \begin{cases} inf & \text{if } \mathbf{x_k} = \mu \\ 0 & \text{otherwise} \end{cases}$$

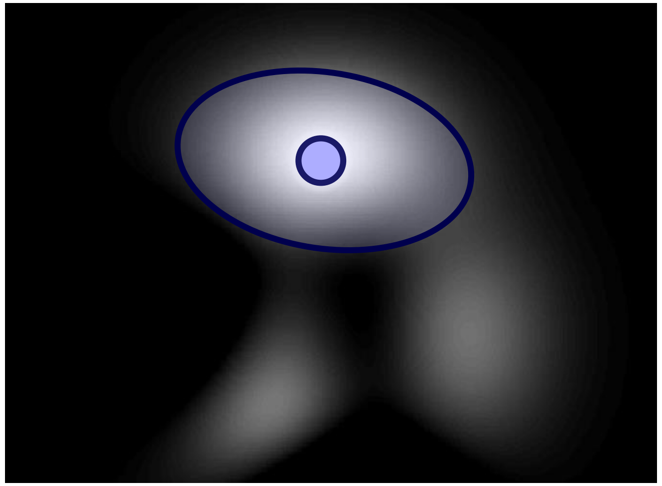
$$\int_{-\infty}^{\infty} p(\mathbf{x}_k|\mathbf{y}_{1:k}) d\mathbf{x}_k = 1$$

Analytic representations

- A single point + covariance: a Gaussian distribution

current input image

current posterior



$p(\mathbf{x}_k|\mathbf{y}_{1:k})$

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \mathcal{N}(\mathbf{x}|\mu;\boldsymbol{\Sigma})$$

$$\mathcal{N}(\mathbf{x}|\mu;\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}_k-\mu)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_k-\mu)}$$

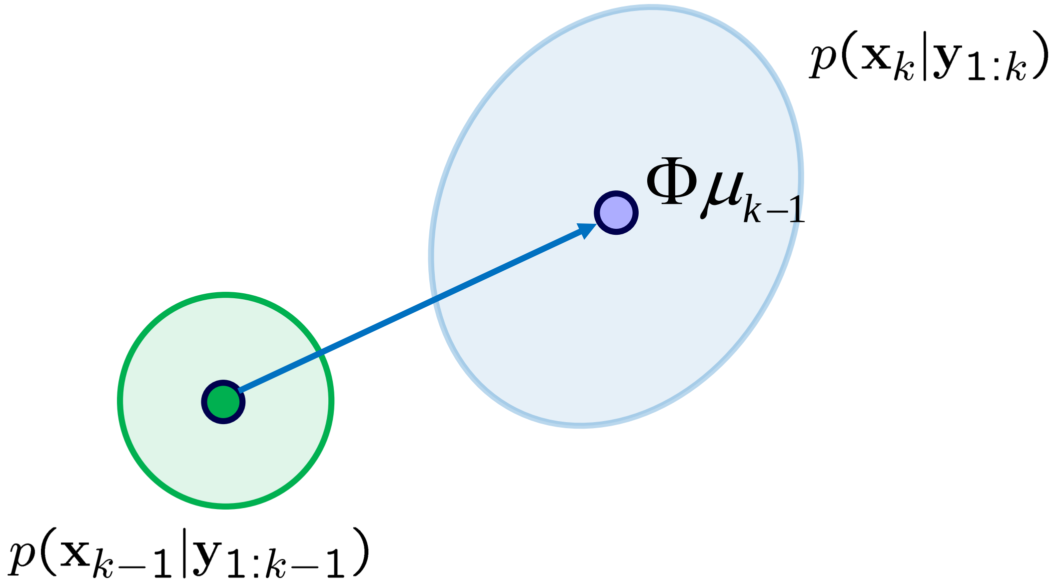
What if everything was Gaussian?

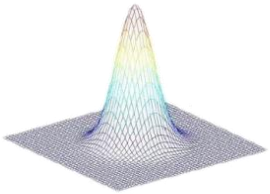
- Assume that:
  - The **posterior** is a single **Gaussian**.

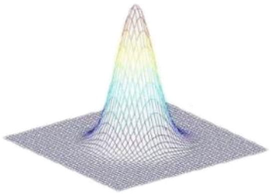
    $$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \mathcal{N}(\mathbf{x}_k|\mu_k; \mathbf{P}_k)$$

  - **Dynamic model** is linear with **Gaussian** noise.

    $$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k|\Phi\mathbf{x}_{k-1}; \mathbf{Q}_k)$$







$$\underbrace{p(\mathbf{x}_k|\mathbf{y}_{1:k})}_{\text{posterior estimate}} \propto \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\text{Observation model}} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1})}_{\text{motion model}} \underbrace{p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})}_{\text{posterior at k-1}} d\mathbf{x}_{k-1}$$

What if everything was Gaussian?

- Assume that:
  - The **posterior** is a single **Gaussian**.

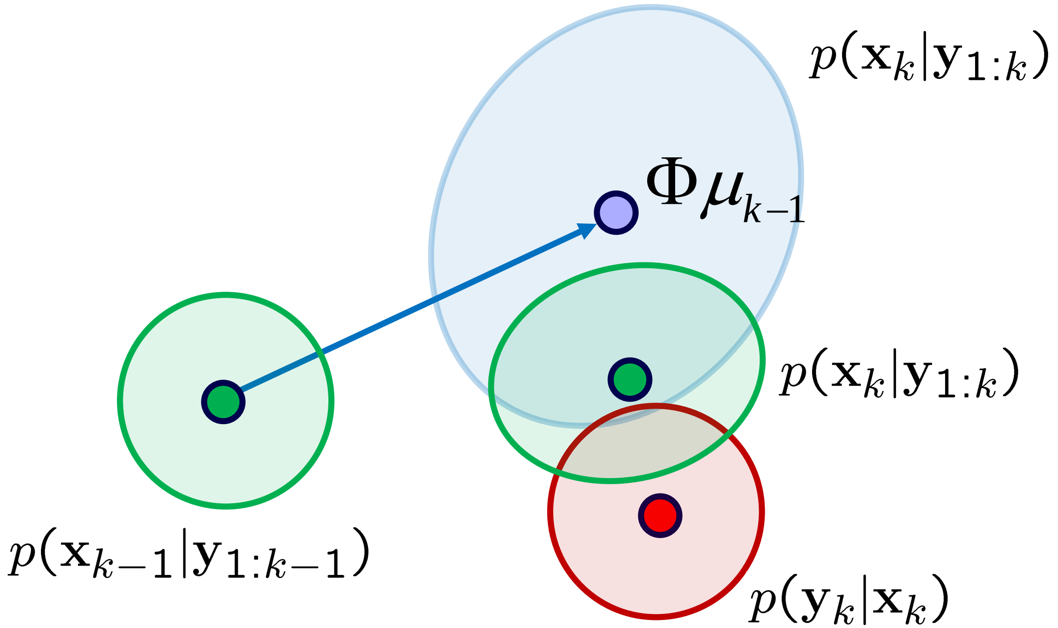
    $$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \mathcal{N}(\mathbf{x}_k|\mu_k; \mathbf{P}_k)$$

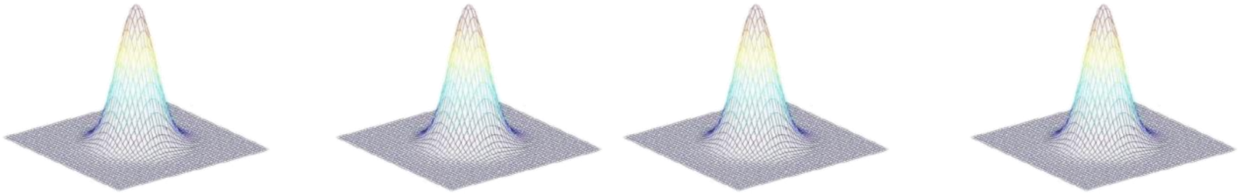
  - **Dynamic model** is linear with **Gaussian** noise.

    $$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k|\Phi\mathbf{x}_{k-1}; \mathbf{Q}_k)$$

  - **Observation** model is a **Gaussian**.

    $$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k|\mathbf{H}\mathbf{x}_\mathbf{k}; \mathbf{R}_k)$$





$$\underbrace{p(\mathbf{x}_k|\mathbf{y}_{1:k})}_{\text{posterior estimate}} \propto \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\text{Observation model}} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1})}_{\text{motion model}} \underbrace{p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})}_{\text{posterior at k-1}} d\mathbf{x}_{k-1}$$

The Kalman filter

- Assume that all distributions as Gaussians
- And assume linear dynamics
- A well-known filter emerges

The Kalman filter!

- Originally presented as a recursive Least Squares method, not as a Recursive Bayes Filter

Kalman, R. E. 1960-2016. A New Approach to Linear Filtering and Prediction Problems,‖ Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).

Recall some basic statistics

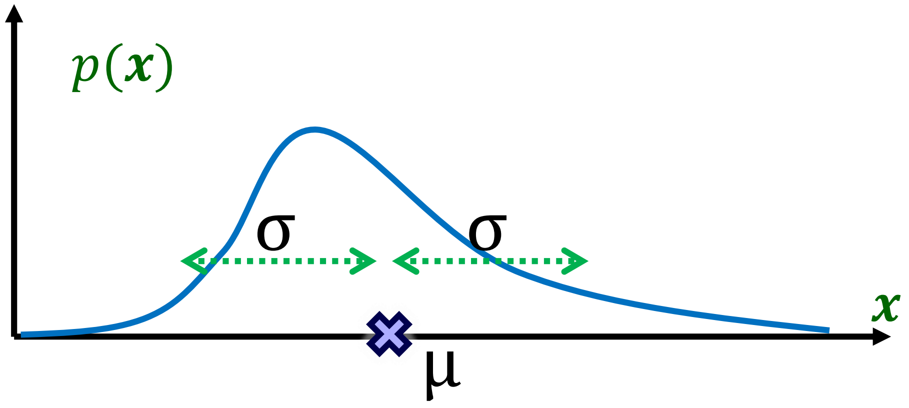
- Recall expected values
  - Expected value (weighted average of x)

$$\mu = \langle x \rangle_{p(\mathrm{x})} = \int_{-\infty}^{\infty} x p(\mathrm{x})\, \mathrm{dx}$$

For short: $\langle x \rangle = \langle x \rangle_{p(\mathrm{x})}$

  - Variance=expected squared change *(i.e., weighted average of* $(x-\mu)^2$*)*

$$\sigma^2 = \left\langle (x-\mu)^2 \right\rangle_{p(\mathrm{x})} = \int_{-\infty}^{\infty} (x-\mu)^2\, p(\mathrm{x})\, \mathrm{dx}$$



[1]See Barber's "Bayesian reasoning and machine learning", Section 8.2. (page 166)

Recall some basic statistics

- Same for vectors…
  - Expected value (weighted average of $\boldsymbol{x}$)

$$\boldsymbol{\mu} = \langle \mathbf{x} \rangle = \int_{-\infty}^{\infty} \mathbf{x} p(\mathbf{x}) \, \mathrm{d}\, \mathbf{x}$$

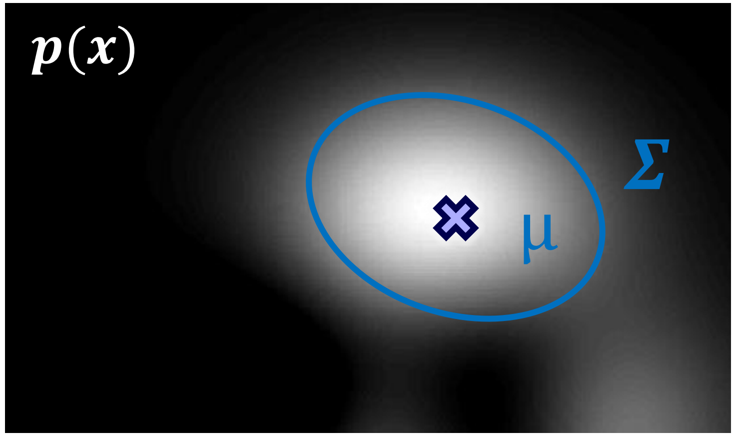
  - Variance = expected sq. change,

$$\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{\mu}$$

$$\boldsymbol{\Sigma} = \left\langle \Delta \mathbf{x} \Delta \mathbf{x}^T \right\rangle$$

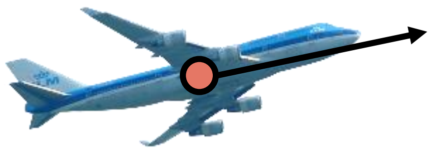
$$= \left\langle (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \right\rangle$$

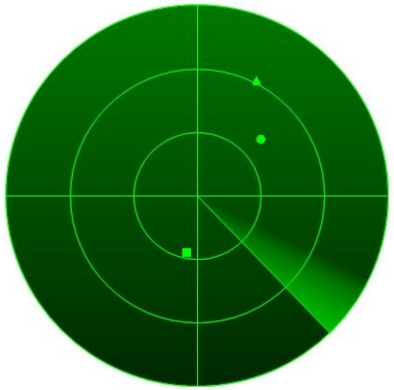
$$= \int_{-\infty}^{\infty} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \, p(\mathbf{x}) \, \mathrm{dx}$$



A working example

- Track an airplane





- State: position and velocity

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}$$

- Observe: only position

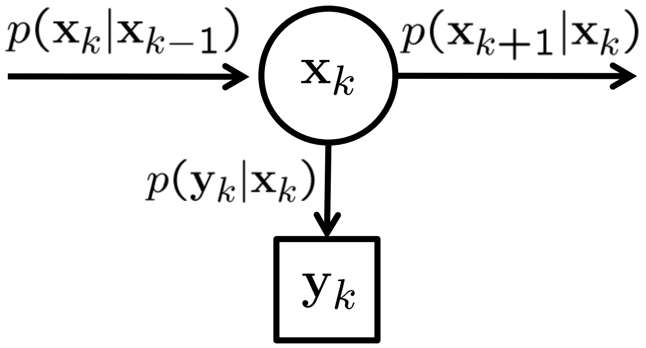
$$\mathbf{y}_k = \begin{bmatrix} x_k^{(\text{measured})} \\ y_k^{(\text{measured})} \end{bmatrix}$$

- Dynamics: Assume a NCV model

Slightly more formally

- State and observation

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} \qquad \mathbf{y}_k = \begin{bmatrix} x_k^{(\mathrm{m})} \\ y_k^{(\mathrm{m})} \end{bmatrix}$$



- Dynamic model

$$\mathbf{x}_k = \Phi \mathbf{x}_{k-1} + \mathbf{w}_k$$

noise $\mathbf{Q}_k$

$$\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + w_k$$

- Observation model

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$$

noise $\mathbf{R}_k$

$$\begin{bmatrix} x_k^{(m)} \\ y_k^{(m)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} + \mathbf{v}_k$$

The recursive Bayes filter

- Recall the recursive equation

$$\underbrace{p(\mathbf{x}_k|\mathbf{y}_{1:k})}_{\text{posterior estimate}} \propto \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\text{Observation model}} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1})}_{\text{motion model}} \underbrace{p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})}_{\text{posterior at k-1}} d\mathbf{x}_{k-1}$$

- The next few slides:
  1. Solve the integral: $\mathrm{p}(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} \mid y_{1:k-1}) \,\mathrm{d}\,\mathbf{x}_{k-1}$
  2. Solve the posterior update: $\mathrm{p}(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \propto \mathrm{p}(\mathbf{y}_k \mid \mathbf{x}_k)\, \mathrm{p}(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$
  3. We will use a few tricks that apply to Gaussians

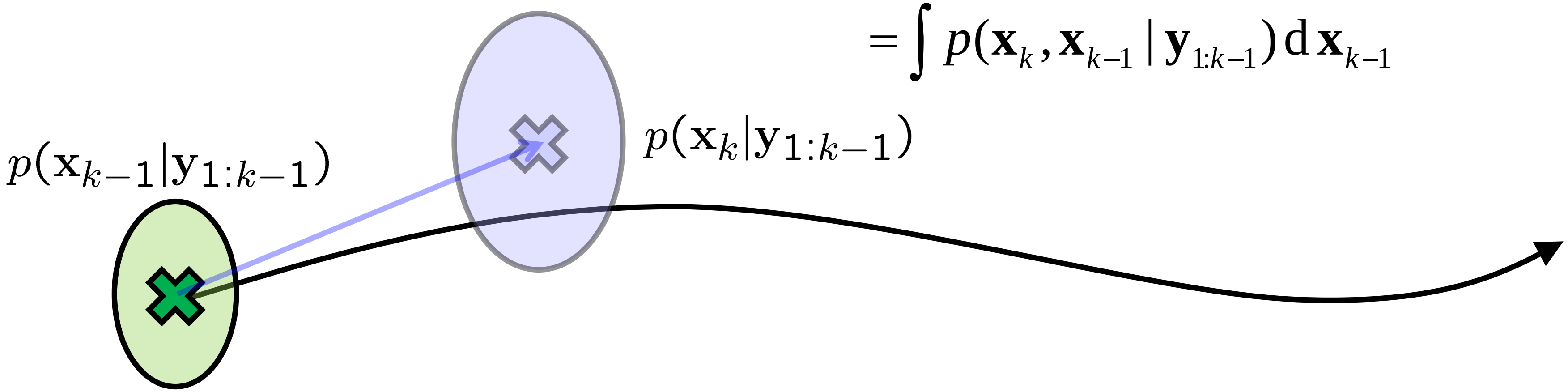
You may want to review the properties of Gaussian, integrals, marginal, etc., in Barber's "Bayesian reasoning and machine learning", Section 8.4.

Solving the prediction

- Solve the integral:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\mathbf{x}_{k-1}$$

$$= \int p(\mathbf{x}_k, \mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\mathbf{x}_{k-1}$$

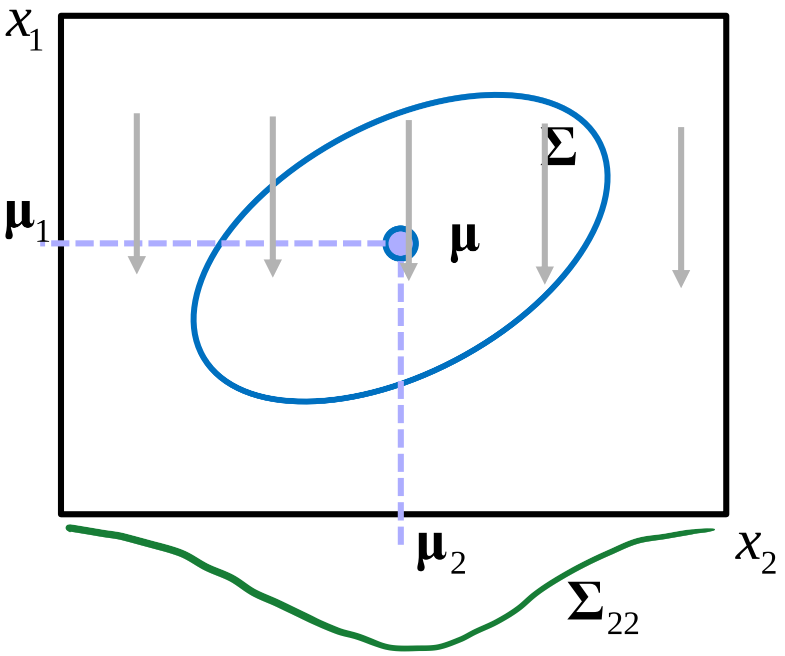


- Since $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ and $p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})$ are both Gaussians, their product will be a Gaussian as well.
- Note that we want to solve the following form:

$$p(\mathbf{x}_2) = \int p(\mathbf{x}_2 \mid \mathbf{x}_1)\, p(\mathbf{x}_1)\, \mathrm{d}\mathbf{x}_1$$

A note on marginalization

- Marginalization: $\int p(\mathbf{x}_2 \mid \mathbf{x}_1)\, p(\mathbf{x}_1)\, \mathrm{d}\,\mathbf{x}_1 = \int p(\mathbf{x}_2, \mathbf{x}_1)\, \mathrm{d}\,\mathbf{x}_1 = \mathrm{p}(\mathbf{x}_2)$



$$p(\mathbf{x}_1, \mathbf{x}_2) = N(\mu, \Sigma)$$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}, \boldsymbol{\Sigma}_{12} = \boldsymbol{\Sigma}_{21}^T$$

$$p(\mathrm{x}_2) = \int p(\mathrm{x}_1, x_2) dx_1 = N(\mathrm{x}_2; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22})$$

- One could solve this integral by "completing the squares"[1], but we can take a shortcut.

[1]See Barber's "Bayesian reasoning and machine learning", Section 8.4.1.

Solving the prediction

$$\mathrm{p}(\mathbf{x}_2) = \int p(\mathbf{x}_2 \mid \mathbf{x}_1)\, p(\mathbf{x}_1)\, \mathrm{d}\,\mathbf{x}_1$$

- The prior is a Gaussian $p(\mathbf{x}_1) = N(\mathbf{x}_1; \mu_1, \Sigma_1)$
- The dynamic model takes the prior and "pushes" it through a linear model and adds noise:

$$\mathbf{x}_2 = \mathbf{\Phi}\mathbf{x}_1 + \mathbf{W} \;,\; \mathbf{W} \sim N(\boldsymbol{\mu} = 0, \mathbf{Q}) \longrightarrow \mathbf{x}_2 \sim N(\mathbf{x}_2; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

$$\boldsymbol{\mu}_2 = \langle \mathbf{x}_2 \rangle = \langle \mathbf{\Phi}\mathbf{x}_1 + \mathbf{W} \rangle = \mathbf{\Phi}\langle \mathbf{x}_1 \rangle + \langle \mathbf{W} \rangle = \mathbf{\Phi}\boldsymbol{\mu}_1$$

$$\Delta\mathbf{x}_2 = \mathbf{\Phi}\Delta\mathbf{x}_1 + \Delta\mathbf{W}$$

$$\begin{aligned}
\boldsymbol{\Sigma}_2 &= \langle \Delta\mathbf{x}_2 \Delta\mathbf{x}_2^T \rangle = \langle (\mathbf{\Phi}\Delta\mathbf{x}_1 + \Delta\mathbf{W})(\mathbf{\Phi}\Delta\mathbf{x}_1 + \Delta\mathbf{W})^T \rangle \\
&= \langle \mathbf{\Phi}\Delta\mathbf{x}_1\Delta\mathbf{x}_1^T\mathbf{\Phi}^T + \mathbf{\Phi}\Delta\mathbf{x}_1\Delta\mathbf{W}^T + \Delta\mathbf{W}\Delta\mathbf{x}_1^T\mathbf{\Phi}^T + \Delta\mathbf{W}\Delta\mathbf{W}^T \rangle \\
&= \mathbf{\Phi}\langle \Delta\mathbf{x}_1\Delta\mathbf{x}_1^T \rangle \mathbf{\Phi}^T + \mathbf{\Phi}\langle \Delta\mathbf{x}_1\Delta\mathbf{W}^T \rangle + \langle \Delta\mathbf{W}\Delta\mathbf{x}_1^T \rangle \mathbf{\Phi}^T + \langle \Delta\mathbf{W}\Delta\mathbf{W}^T \rangle \\
&= \mathbf{\Phi}\boldsymbol{\Sigma}_1\mathbf{\Phi}^T + \mathbf{Q}
\end{aligned}$$

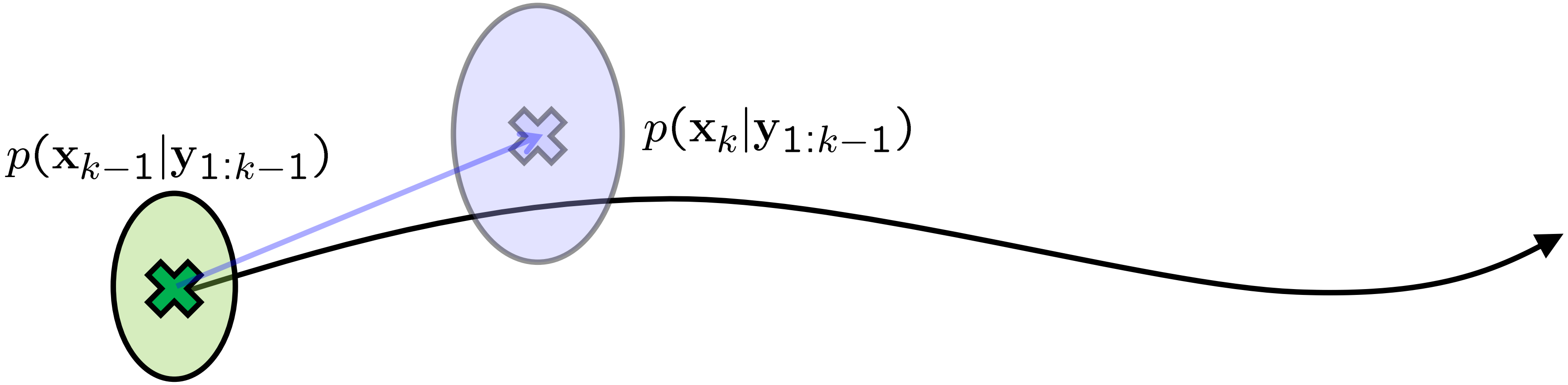
Solving the prediction

- To summarize:

$$p(\mathbf{x}_1) = \mathrm{N}(\mu_1, \Sigma_1)$$

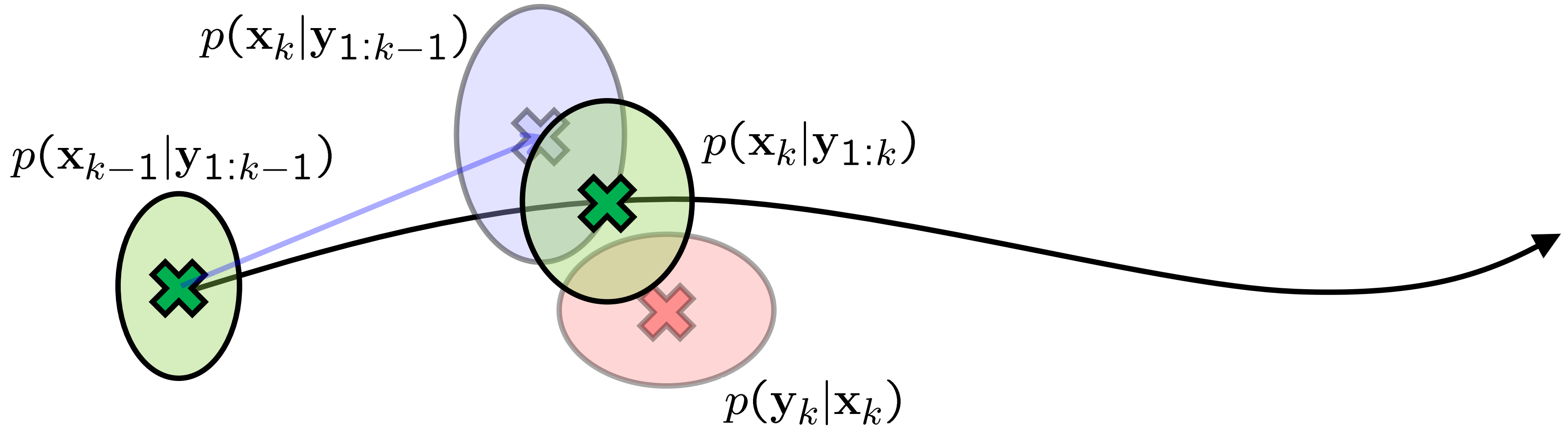
$$\mathbf{x}_2 = \mathbf{\Phi}\mathbf{x}_1 + \mathbf{W} \quad , \quad \mathbf{W} \sim N(\boldsymbol{\mu} = 0, \mathbf{Q}) \quad , \quad p(\mathbf{x}_2 \mid \mathbf{x}_1) \sim N(\mathbf{x}_2; \mathbf{\Phi}\mathbf{x}_1, \mathbf{Q})$$

$$\mathrm{p}(\mathbf{x}_2) = \int p(\mathbf{x}_2 \mid \mathbf{x}_1)\, p(\mathbf{x}_1)\, \mathrm{d}\,\mathbf{x}_1 = N(\mathbf{x}_2; \mathbf{\Phi}\boldsymbol{\mu}_1, \mathbf{\Phi}\mathbf{\Sigma}_1\mathbf{\Phi}^T + \mathbf{Q})$$



Solving for the posterior update

- We are ultimately after $p(\mathrm{x}_K \mid y_{1:K}) \propto p(\mathbf{y}_K \mid \mathbf{x}_K)\, p(\mathrm{x}_K \mid y_{1:K-1})$

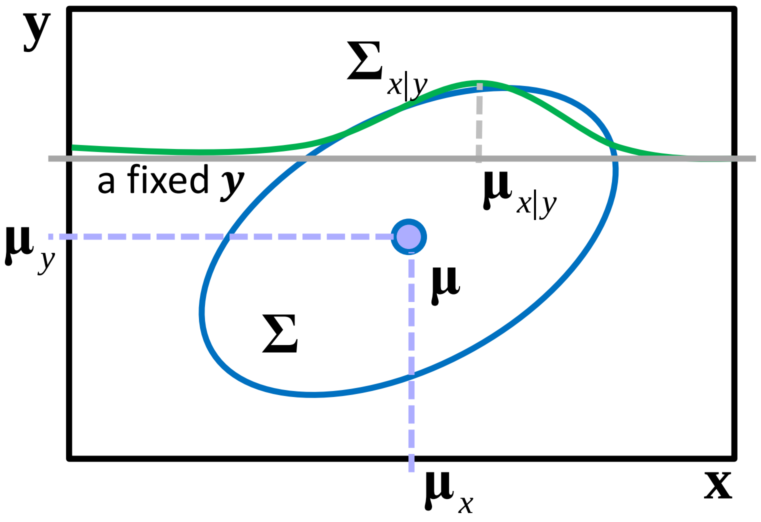


- We make use of the fact that the product of two Gaussians is a Gaussian as well.
- Note that we are considering the following problem:

$$p(\mathbf{x} \mid \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{x})\, \mathrm{p}(\mathbf{x})$$

Solving for the posterior update

- Will take the following shortcut: $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})\,\mathrm{p}(\mathbf{x})$
  - Compute the joint pdf $p(\mathbf{x},\mathbf{y})$
  - Condition on y: $p(\mathbf{x}|\mathbf{y})$



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

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix},\ \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{yx} & \boldsymbol{\Sigma}_{yy} \end{bmatrix},\ \boldsymbol{\Sigma}_{xy} = \boldsymbol{\Sigma}_{yx}^T$$

$$p(\mathbf{x}|\mathbf{y}) = N(\mathrm{x};\mu_{x|y},\Sigma_{x|y})$$

- Established result states that:

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y)$$

$$\boldsymbol{\Sigma}_{x|y} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{yx}$$

Solving for the posterior update

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

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{xx} & \boldsymbol{\Sigma}_{xy} \\ \boldsymbol{\Sigma}_{xy} & \boldsymbol{\Sigma}_{yy} \end{bmatrix}$$

- Recall: $p(\mathbf{x}\,|\,\mathbf{y}) \propto p(\mathbf{y}\,|\,\mathbf{x})\,\mathrm{p}(\mathbf{x})$

$\mathbf{x} \sim N(\mu_x, \Sigma_{xx})$ … the prior on **x**

$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{V}$, $\mathbf{V} \sim N(\boldsymbol{\mu} = 0, \mathbf{R})$ … the observation model

- From the results of conditioning we have:

$$p(\mathbf{x}\,|\,\mathbf{y}) = N(\mathbf{x}; \boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y})$$

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}(\mathbf{y} - \boldsymbol{\mu}_y)$$

$$\boldsymbol{\Sigma}_{x|y} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xy}\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\Sigma}_{xy}^{T}$$

$\Rightarrow$ The following values are required:

$\boldsymbol{\mu}_y = ?$, $\boldsymbol{\Sigma}_{xy} = ?$, $\boldsymbol{\Sigma}_{yy} = ?$

Solving for the posterior update

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{V}$$
$$\mathbf{V} \sim N(\boldsymbol{\mu} = 0, \mathbf{R})$$
$$\mathbf{x} \sim N(\mu_x, \Sigma_{xx})$$

- We are after: $\boldsymbol{\mu}_y = ?\ ,\ \boldsymbol{\Sigma}_{xy} = ?\ ,\ \boldsymbol{\Sigma}_{yy} = ?$

$$\boldsymbol{\mu}_y = \langle \mathbf{y} \rangle = \langle \mathbf{H}\mathbf{x} + \mathbf{V} \rangle = \mathbf{H}\boldsymbol{\mu}_x$$

$$\Delta\mathbf{y} = \mathbf{H}\Delta\mathbf{x} + \Delta\mathbf{V}$$

$$\boldsymbol{\Sigma}_{yy} = \left\langle \Delta\mathbf{y}\Delta\mathbf{y}^T \right\rangle = \left\langle (\mathbf{H}\Delta\mathbf{x} + \Delta\mathbf{V})(\mathbf{H}\Delta\mathbf{x} + \Delta\mathbf{V})^T \right\rangle$$
$$= \mathbf{H}\langle \Delta\mathbf{x}\Delta\mathbf{x}^T \rangle \mathbf{H}^T + 0 + 0 + \langle \Delta\mathbf{V}\Delta\mathbf{V}^T \rangle = \mathbf{H}\boldsymbol{\Sigma}_{xx}\mathbf{H}^T + \mathbf{R}$$

$$\boldsymbol{\Sigma}_{xy} = \langle \Delta\mathbf{x}\Delta\mathbf{y}^T \rangle = \langle \Delta\mathbf{x}(\mathbf{H}\Delta\mathbf{x} + \Delta\mathbf{V})^T \rangle = \langle \Delta\mathbf{x}\Delta\mathbf{x}^T \rangle \mathbf{H}^T + \langle \Delta\mathbf{x}\Delta\mathbf{V}^T \rangle = \boldsymbol{\Sigma}_{xx}\mathbf{H}^T$$

- And finally: $p(\mathbf{x} \mid \mathbf{y}) = \mathrm{N}(\mathbf{x}; \boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y})$

$$\boldsymbol{\mu}_{x|y} = \boldsymbol{\mu}_x + \boldsymbol{\Sigma}_{xx}\mathbf{H}^T(\mathbf{H}\boldsymbol{\Sigma}_{xx}\mathbf{H}^T + \mathbf{R})^{-1}(\mathbf{y} - \boldsymbol{\mu}_y)$$
$$\boldsymbol{\Sigma}_{x|y} = \boldsymbol{\Sigma}_{xx} - \boldsymbol{\Sigma}_{xx}\mathbf{H}^T(\mathbf{H}\boldsymbol{\Sigma}_{xx}\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}\boldsymbol{\Sigma}_{xx}$$

Putting it all together: Kalman filter

External input (if available)

- Dynamic model: $\mathbf{x}_k = \mathbf{\Phi x}_{k-1} + \boxed{\Gamma \mathbf{u}_k} + \mathbf{W}_k \qquad \mathbf{W}_k \sim \mathrm{N}(\mu = 0, \mathbf{Q})$
- Observation model: $\mathbf{y}_k = \mathbf{H x}_k + \mathbf{V}_k \qquad \mathbf{V}_k \sim \mathrm{N}(\mu = 0, \mathbf{R})$
- Initial posterior: $p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1}) = \mathrm{N}(\mathbf{x}_{k-1}; \mu = \hat{\mathbf{x}}_{k-1}, \Sigma = \mathbf{P}_{k-1})$

1. Prediction: $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \mathrm{N}(\mathbf{x}_k; \mu = \tilde{\mathbf{x}}_k, \Sigma = \tilde{\mathbf{P}}_k)$

$$\tilde{\mathbf{x}}_k = \mathbf{\Phi} \hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_k$$

$$\tilde{\mathbf{P}}_k = \mathbf{\Phi P}_{k-1} \mathbf{\Phi}^T + \mathbf{Q}$$

2. Update by measurement $\boldsymbol{y}_k$: $p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \mathrm{N}(\mathbf{x}_k; \mu = \hat{\mathbf{x}}_k, \Sigma = \hat{\mathbf{P}}_k)$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}(\mathbf{y}_k - \mathbf{H} \tilde{\mathbf{x}}_k)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{KH}) \tilde{\mathbf{P}}_k$$

This is called the "Kalman gain":

$$\mathbf{K} = \tilde{\mathbf{P}}_k \mathbf{H}^T (\mathbf{H} \tilde{\mathbf{P}}_k \mathbf{H}^T + \mathbf{R})^{-1}$$

See J.D. Prince "Computer vision: models, learning and inference", Section 19.2.3

Making more sense of these equations…

- Prediction: $\tilde{\mathbf{x}}_k = \mathbf{\Phi}\hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_k$

  $\tilde{\mathbf{P}}_k = \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{Q}$

- Update: $\mathbf{K} = \tilde{\mathbf{P}}_k \mathbf{H}^T (\mathbf{H}\tilde{\mathbf{P}}_k \mathbf{H}^T + \mathbf{R})^{-1}$

  $\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}(\mathbf{y}_k - \mathbf{H}\tilde{\mathbf{x}}_k)$

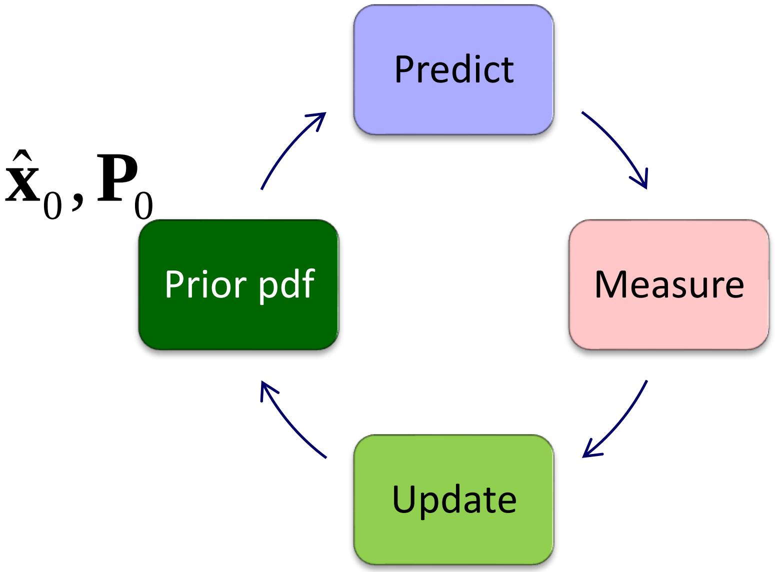
  $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\tilde{\mathbf{P}}_k$

Kalman Filter recursion

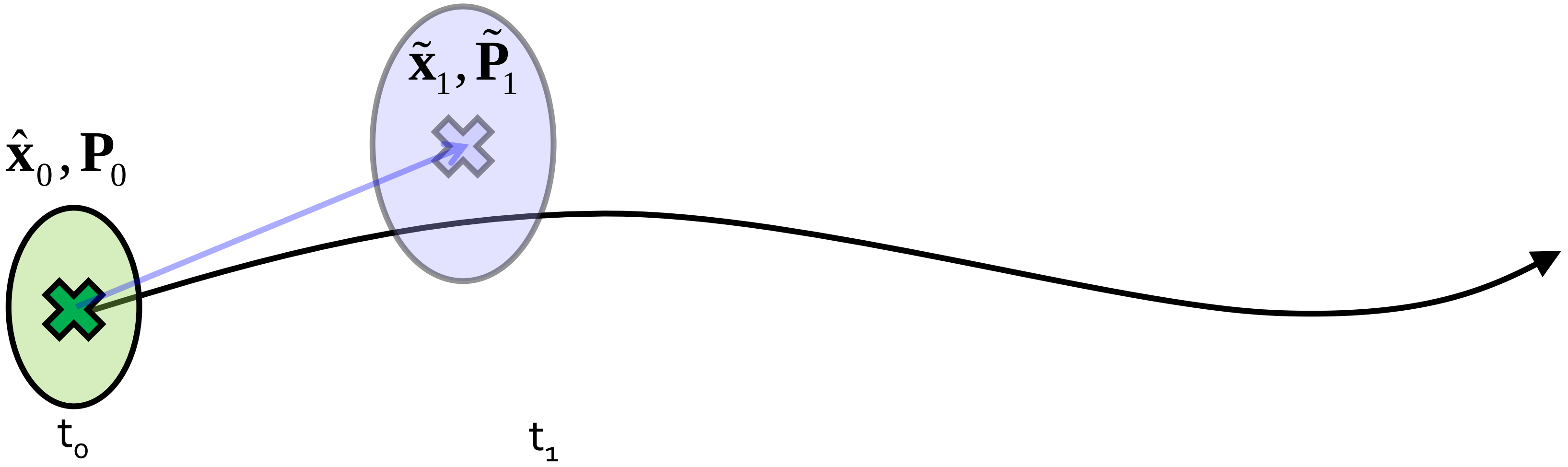
- Initialize

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} x_0 \\ y_0 \\ \dot{x}_0 \\ \dot{y}_0 \end{bmatrix}$$

$$\mathrm{P}_0 = \begin{bmatrix} L & 0 & 0 & 0 \\ 0 & L & 0 & 0 \\ 0 & 0 & L & 0 \\ 0 & 0 & 0 & L \end{bmatrix}$$



Kalman Filter recursion

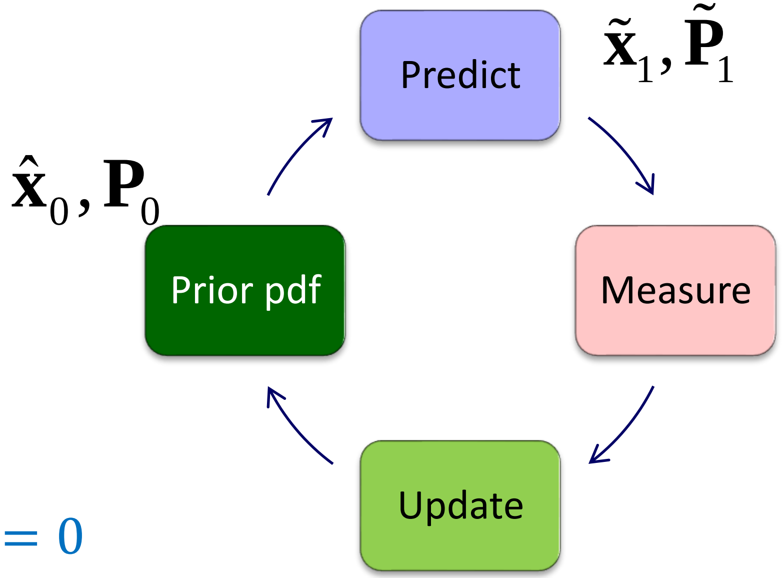


- Predict:

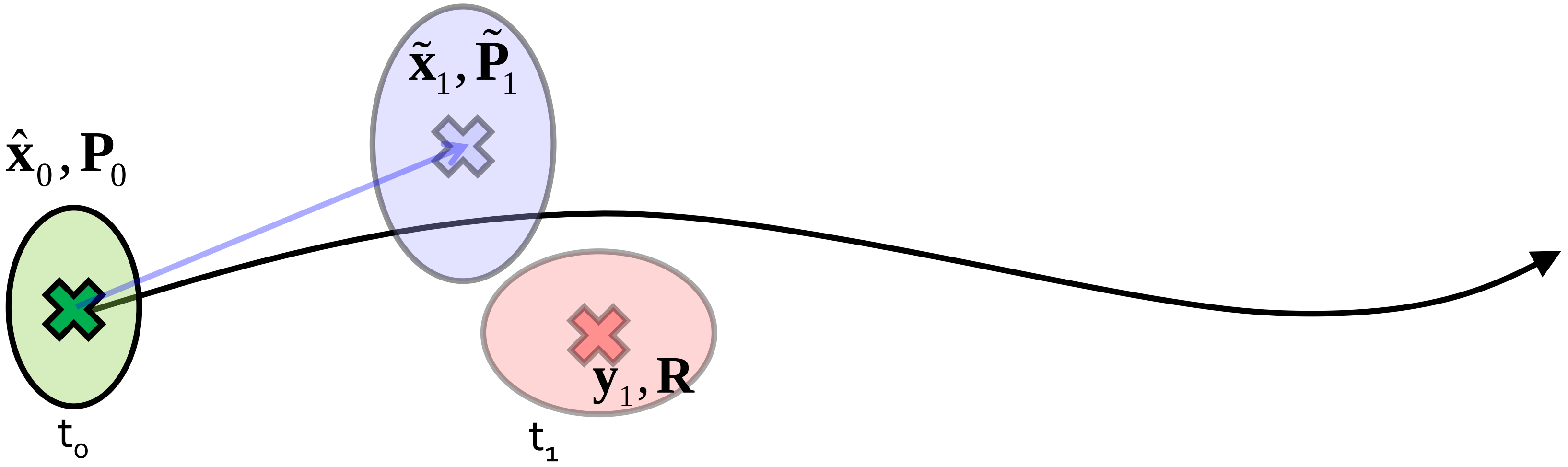
$$\tilde{\mathbf{x}}_1 = \boldsymbol{\Phi}\hat{\mathbf{x}}_0 + \Gamma \mathbf{u}_1$$

$$\tilde{\mathbf{P}}_1 = \boldsymbol{\Phi}\mathbf{P}_0\boldsymbol{\Phi}^T + \mathbf{Q}$$

External input – for most your applications it will be unavailable, i.e.,$\boldsymbol{u}_k = 0$



Kalman Filter recursion

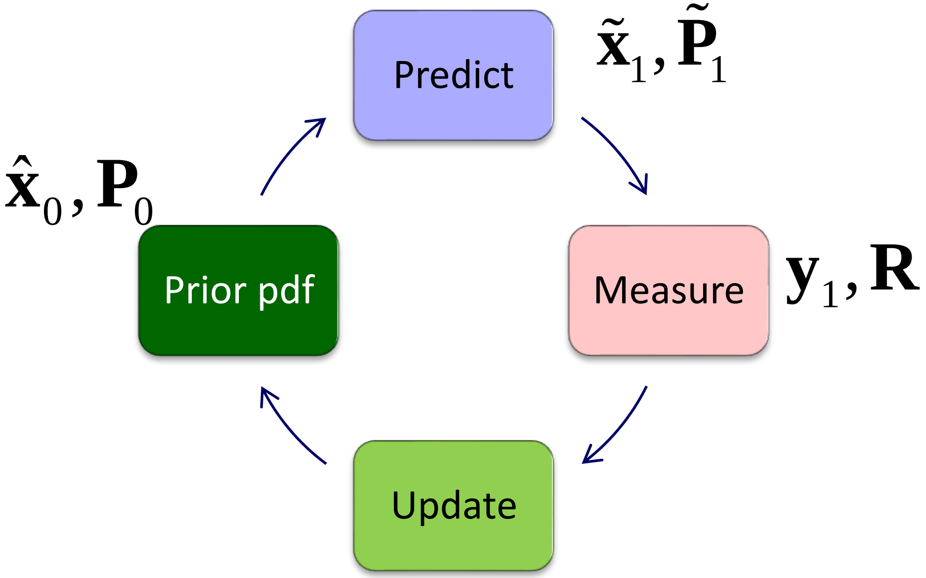


- Receive a noisy measurement

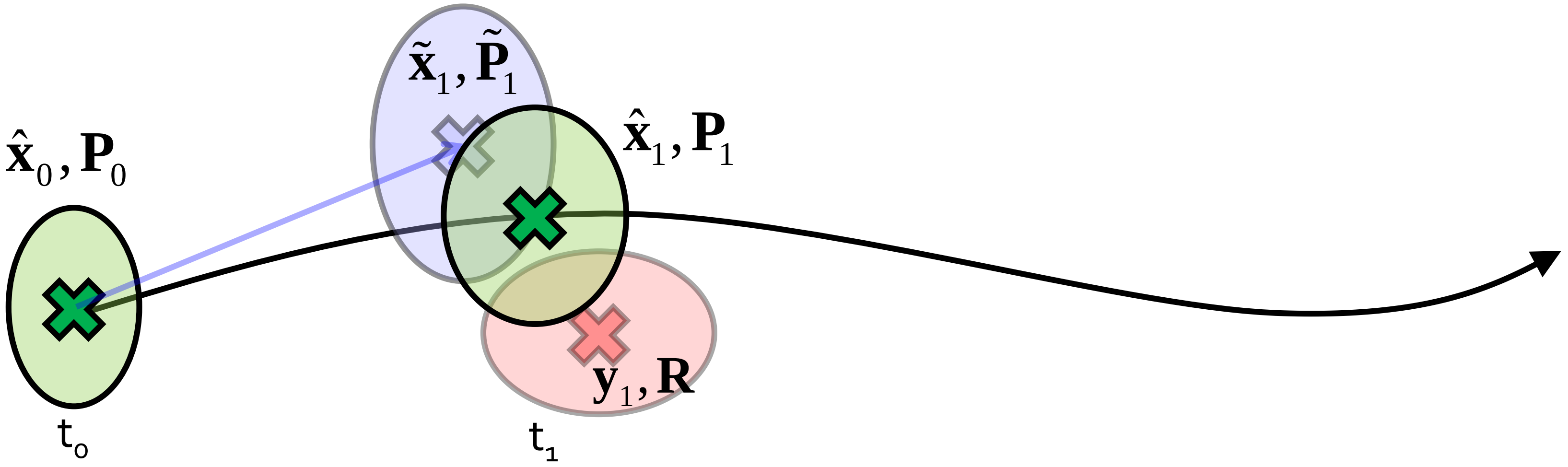
$$\mathbf{y}_1, \mathbf{R}$$

Compute the "Kalman gain":

$$\mathbf{K} = \tilde{\mathbf{P}}_1 \mathbf{H}^T (\mathbf{H}\tilde{\mathbf{P}}_1\mathbf{H}^T + \mathbf{R})^{-1}$$



Kalman Filter recursion

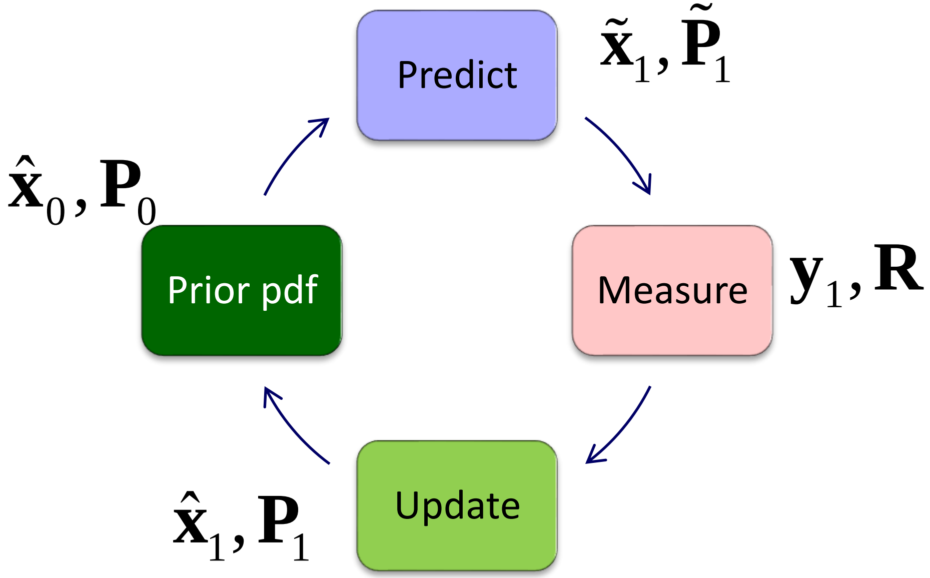


- Update the posterior

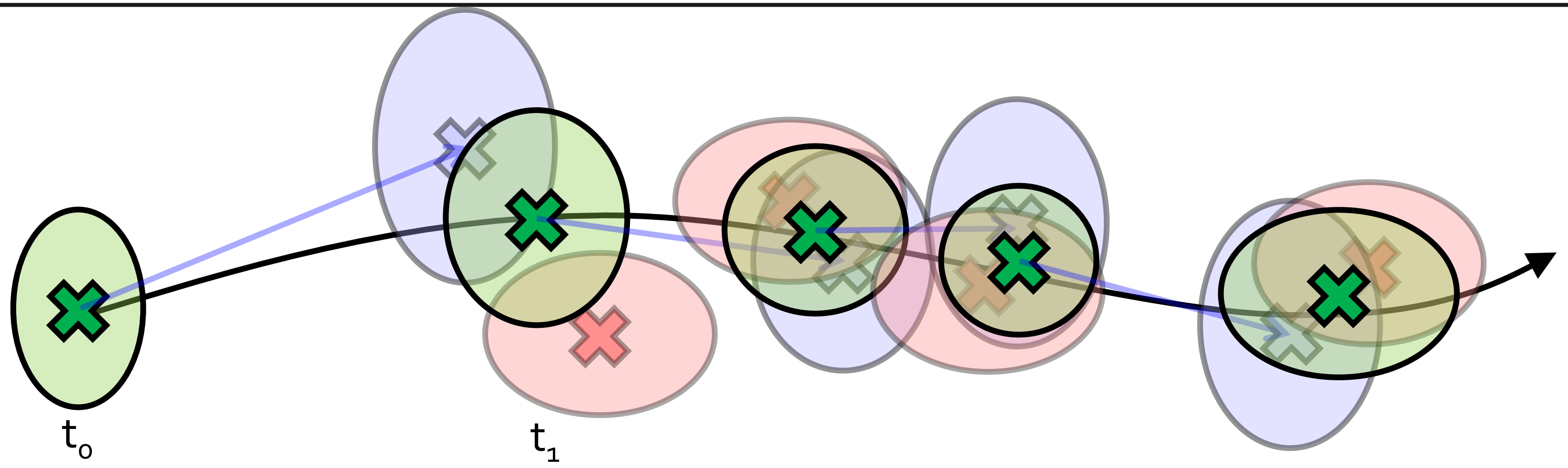
$$\mathbf{K} = \tilde{\mathbf{P}}_1 \mathbf{H}^T (\mathbf{H}\tilde{\mathbf{P}}_1 \mathbf{H}^T + \mathbf{R})^{-1}$$

$$\hat{\mathbf{x}}_1 = \tilde{\mathbf{x}}_1 + \mathbf{K}(\mathbf{y}_1 - \mathbf{H}\tilde{\mathbf{x}}_1)$$

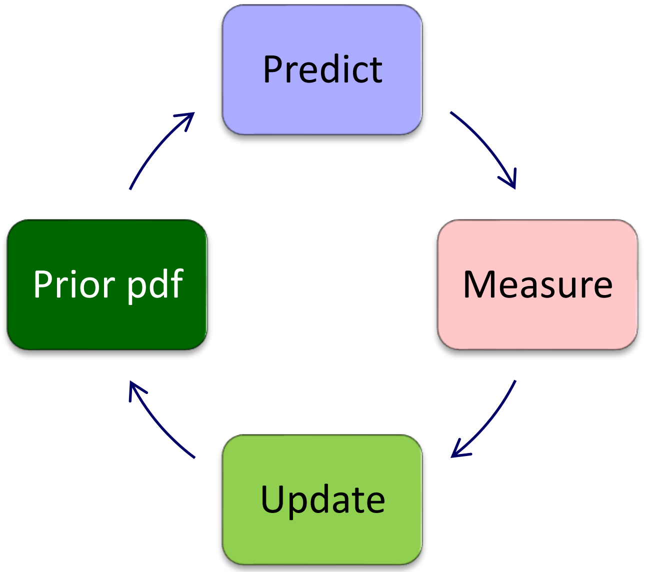
$$\mathbf{P}_1 = (\mathbf{I} - \mathbf{K}\mathbf{H})\tilde{\mathbf{P}}_1$$



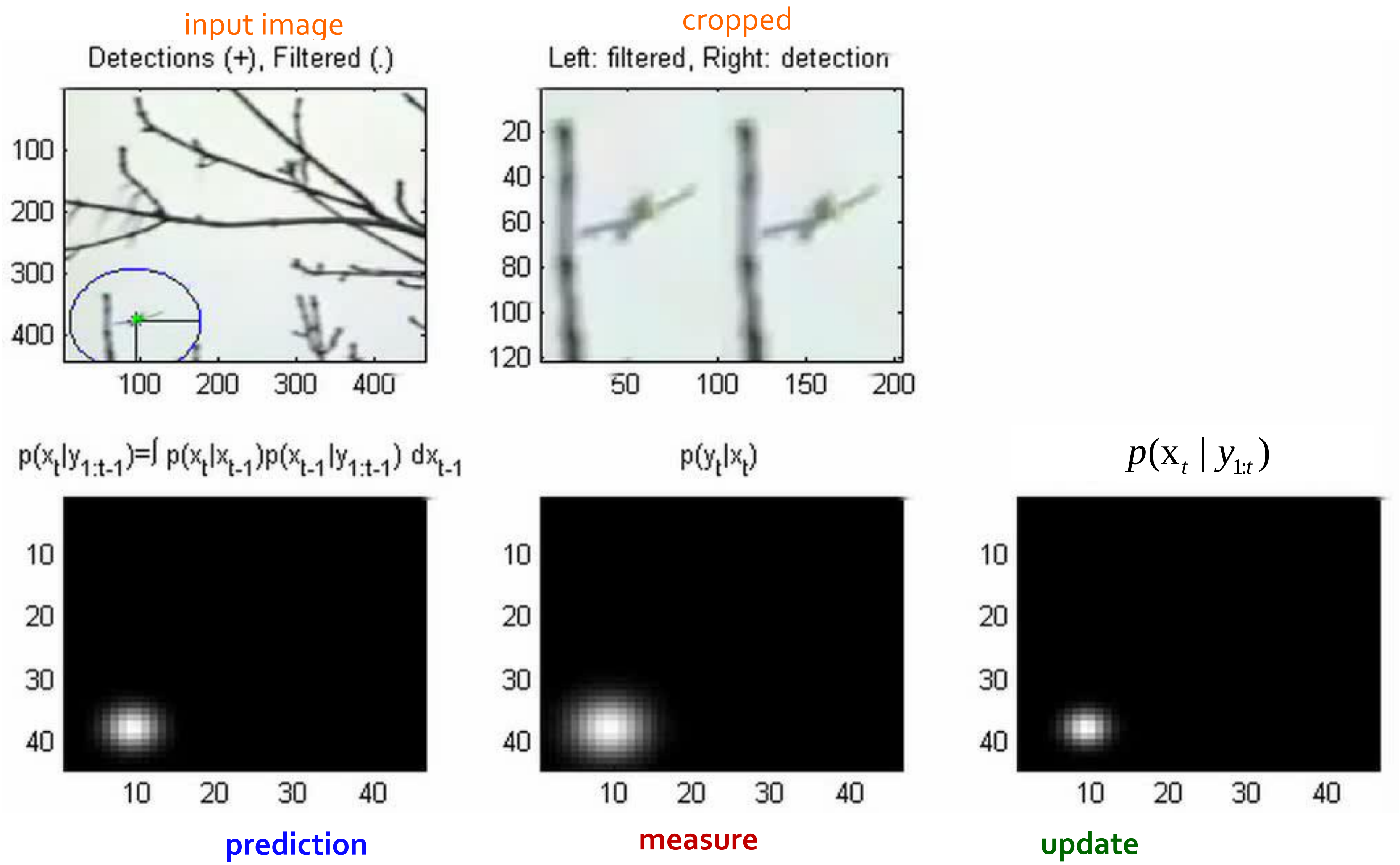
Kalman Filter recursion



1. Prediction from the motion model
2. Receive a noisy measurement
3. Update the posterior



Kalman filter in action

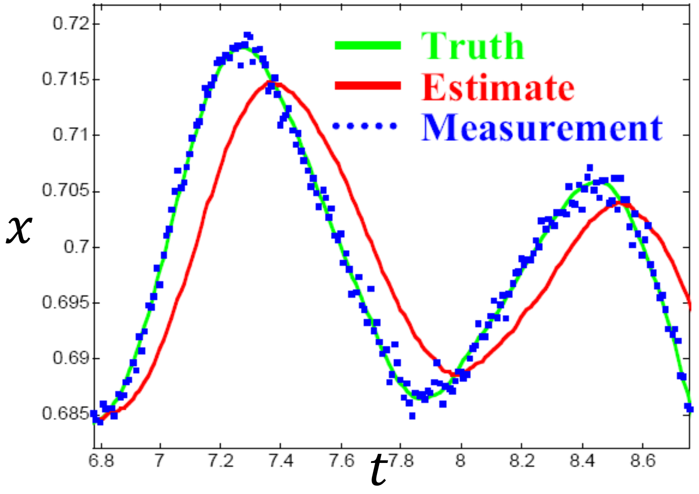


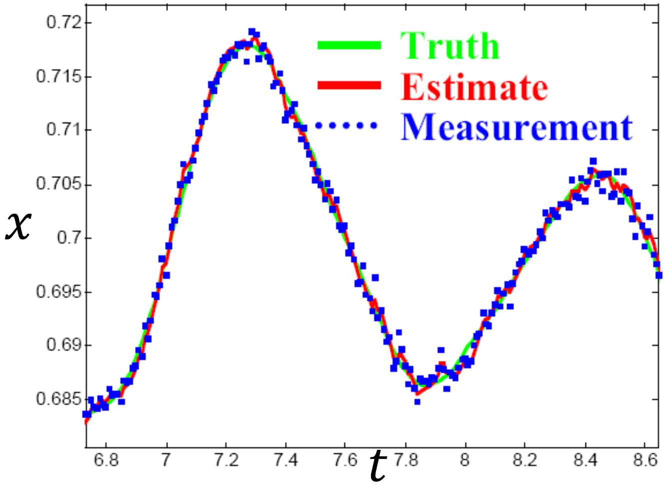
Kalman filter: Dynamics

Without velocity (RW): $\mathbf{x}_t = \begin{pmatrix} x \\ y \end{pmatrix}$

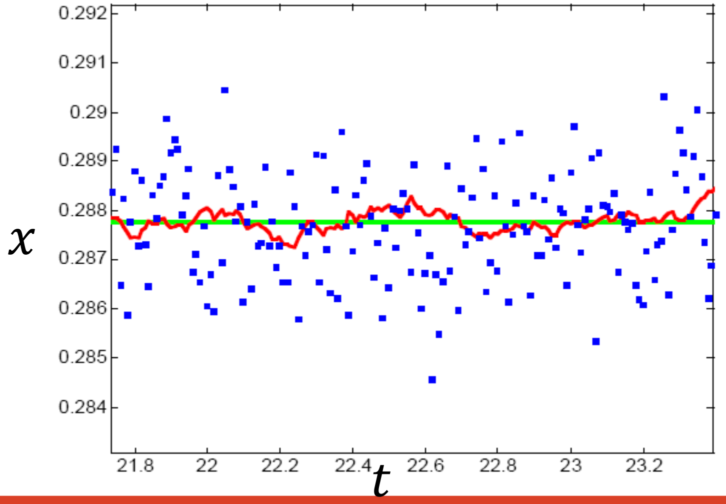
With velocity (NCV): $\mathbf{x}_t = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix}$

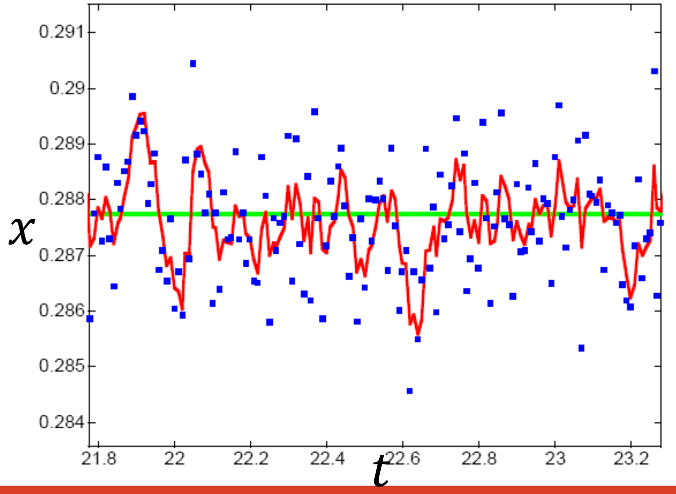
Target moving





Target not moving





Another example

detection

prediction

update

$$\tilde{\mathbf{x}}_k = \mathbf{\Phi}\hat{\mathbf{x}}_{k-1}$$

$$\tilde{\mathbf{P}}_k = \mathbf{\Phi}\mathbf{P}_{k-1}\mathbf{\Phi}^T + \mathbf{Q}$$

$$\mathbf{K} = \tilde{\mathbf{P}}_k\mathbf{H}^T(\mathbf{H}\tilde{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R})^{-1}$$

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{K}(\mathbf{y}_k - \mathbf{H}\tilde{\mathbf{x}}_k)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}\mathbf{H})\tilde{\mathbf{P}}_k$$

Dynamic model takes over when target not detected!

Video from: Kevin Smith's "SELECTED TOPICS IN COMPUTER VISION – 2D tracking"

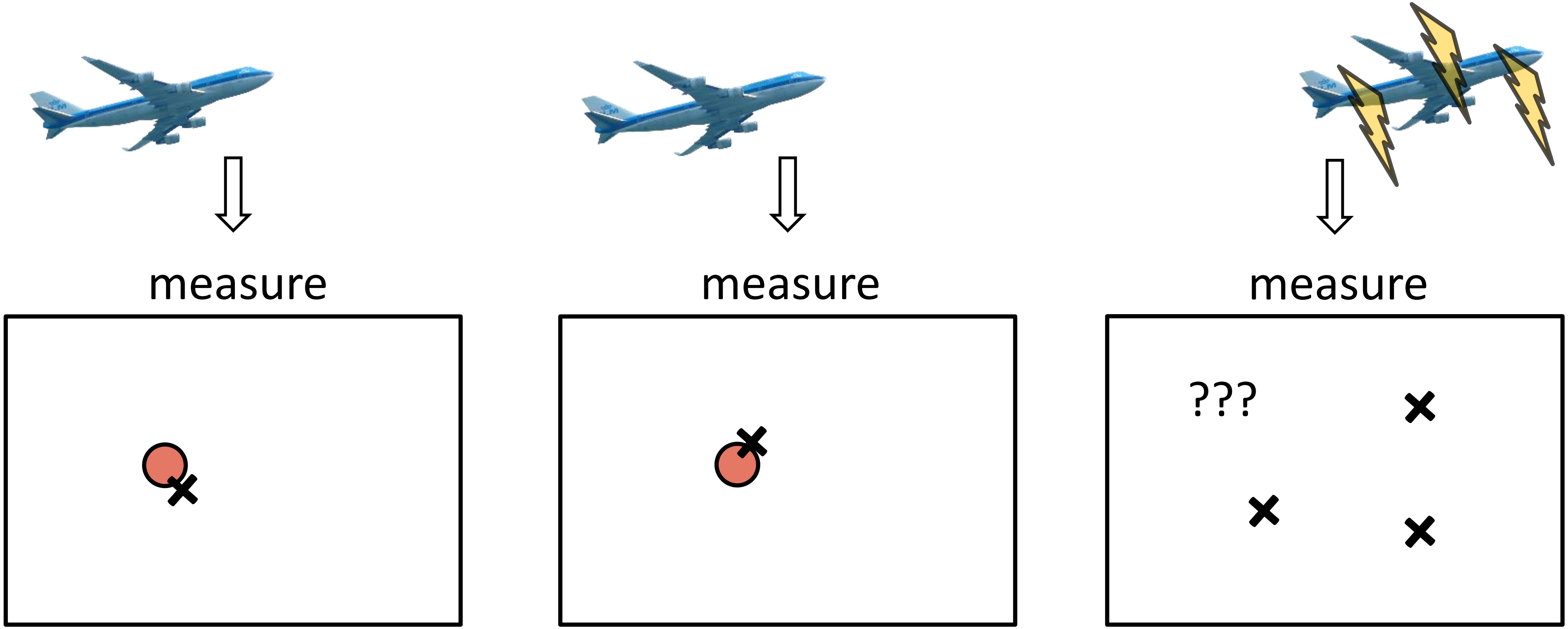
Multiple measurements (noise)?

- Assume a NCV model

$$\mathbf{x}_k = \mathbf{\Phi}\mathbf{x}_{k-1} + \mathbf{W}_k \qquad \mathbf{W}_k \sim \mathrm{N}(\mu = 0, \mathbf{Q})$$

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{V}_k$$

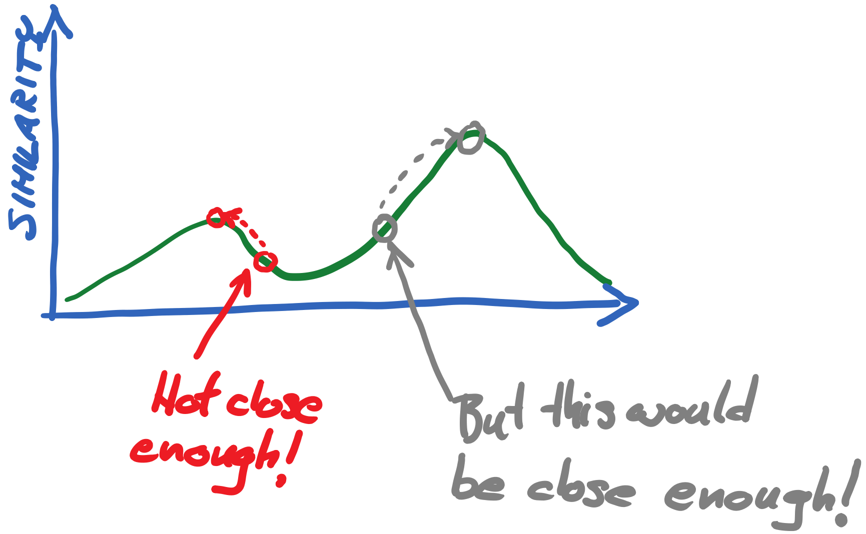
- How would you resolve noisy measurements?

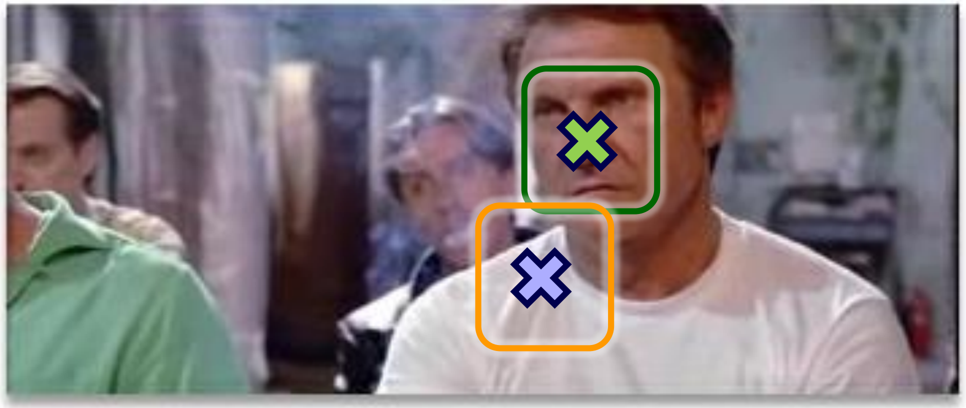


Combine Kalman with local optimization?

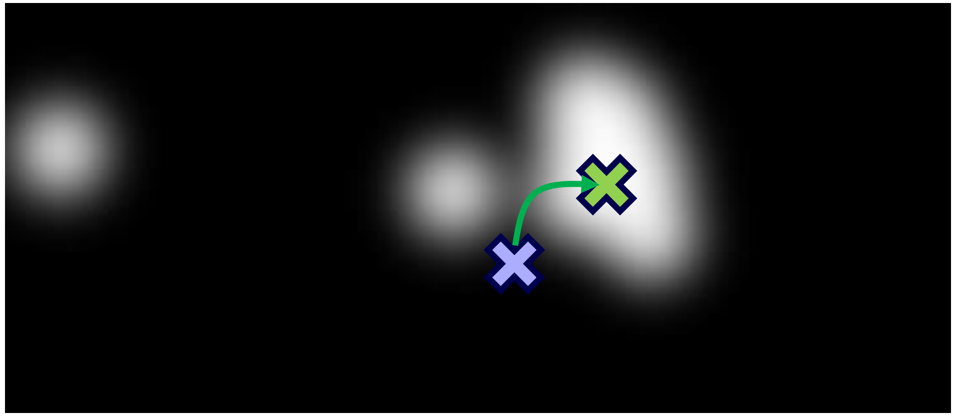
- Recall that Mean Shift converges well if initialized close to solution

Apply Kalman for prediction to better estimate the starting point for MS!





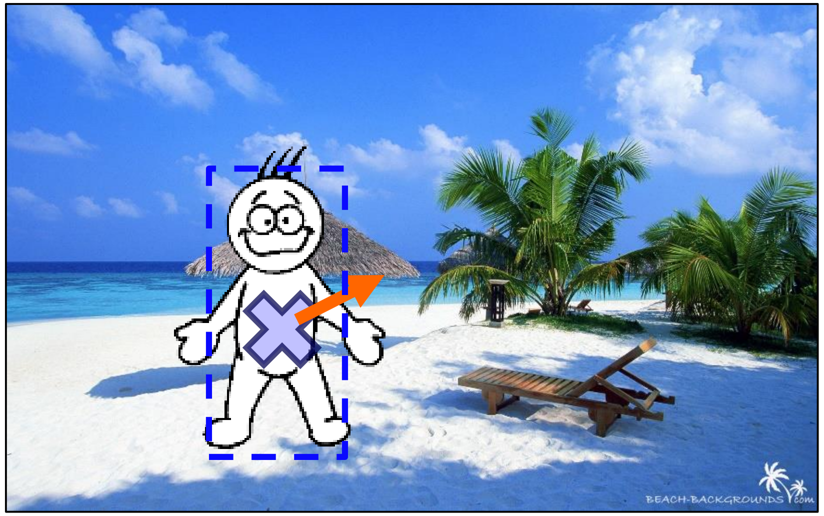
input image

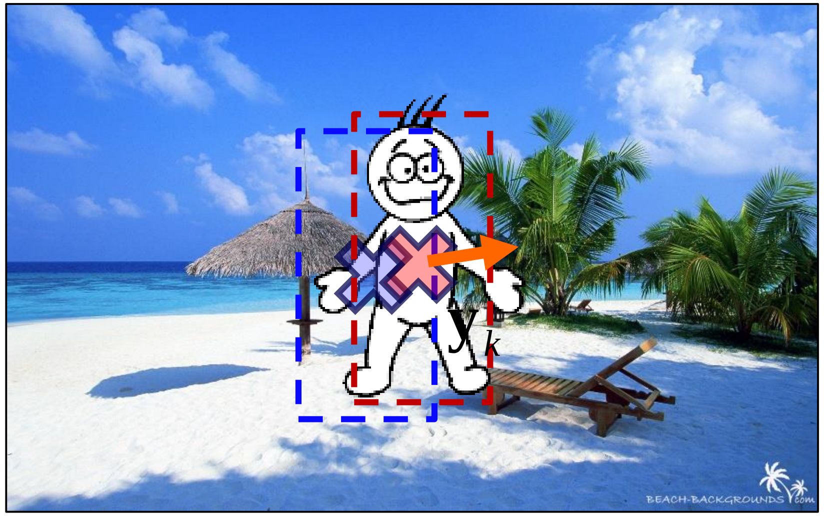


similarity/probability

Improve Mean Shift with the Kalman filter

- Example:
  - Predict initial position from Kalman filter
  - Run Mean Shift to find local optimum – This is measurement $\boldsymbol{y}_k$
  - Update Kalman filter by the measurement $\boldsymbol{y}_k$
  - Prediction improved for the next time-step





“Possible to improve any local optimization by dynamics in this way”

Setting parameters?

- Usually set only the covariances: Q and R

$$\mathbf{x}_k = \mathbf{\Phi x}_{k-1} + \mathbf{W}_k \qquad \mathbf{W}_k \sim \mathrm{N}(\mu = 0, \mathbf{Q})$$

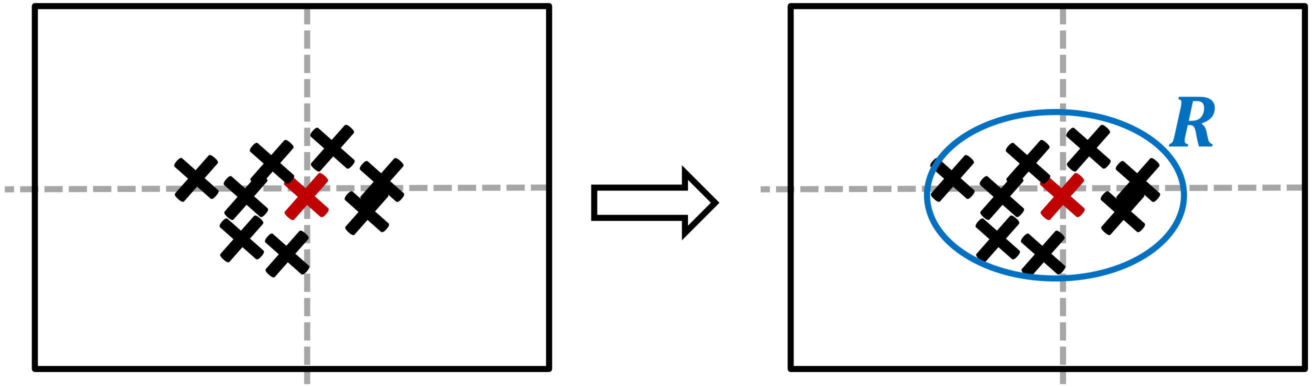
$$\mathbf{y}_k = \mathbf{H x}_k + \mathbf{V}_k \qquad \mathbf{V}_k \sim \mathrm{N}(\mu = 0, \mathbf{R})$$

- By trial and error
- By Expectation Maximization on a reference trajectory
- By rules of thumb – a good starting point

Setting the measurement cov. $\boldsymbol{R}$

- Perform detection with your detector on many examples.
- Manually annotate TRUE positions.

- Calculate *changes* from the true position
- Calculate covariance $\boldsymbol{R}$



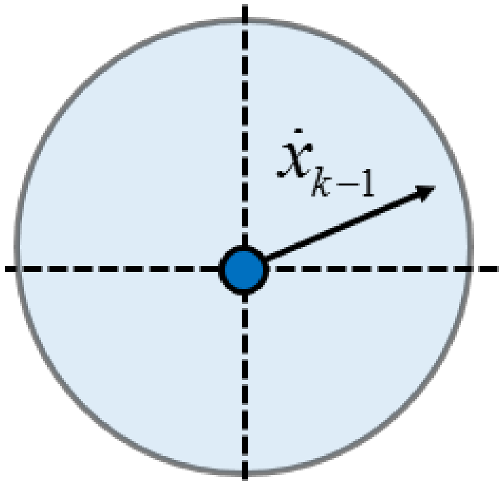
Rule of thumb for dynamics noise Q

- Example, NCV: $\boldsymbol{x}_k = \boldsymbol{\Phi}\boldsymbol{x}_{k-1} + \boldsymbol{W}$, $\Delta t = 1$
- $\boldsymbol{W} \sim N(0, \boldsymbol{Q})$

$$Q = q_c \begin{bmatrix} 1/3 & 1/2 \\ 1/2 & 1 \end{bmatrix} \qquad \Phi = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

- Assume we know the expected squared distance σ_m^2 a target can travel within Δt.
- Target is at origin and *starts moving*, i.e., velocity sampled from noise only:

$$\mathbf{x}_{k-1} = \begin{bmatrix} 0 \\ \dot{x}_{k-1} \end{bmatrix} \qquad \dot{x}_{k-1} \sim q_c$$



A general approach proposed in: M. Kristan et al." A Two-Stage Dynamic Model for Visual Tracking". IEEE SMCB, 2010.

Rule of thumb for dynamics noise Q

- Example, NCV: $\boldsymbol{x}_k = \boldsymbol{\Phi}\boldsymbol{x}_{k-1} + \boldsymbol{W}\,,\quad \boldsymbol{W} \sim N(0, \boldsymbol{Q})$

$$\Phi = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad Q = q_c \begin{bmatrix} 1/3 & 1/2 \\ 1/2 & 1 \end{bmatrix} \qquad \mathbf{x}_{k-1} = \begin{bmatrix} 0 \\ \dot{x}_{k-1} \end{bmatrix} \qquad \dot{x}_{k-1} \sim q_c$$

- The covariance of $\boldsymbol{x}_k$ (expected sq. change):

$$P = \langle (\mathbf{x}_k - 0)(\mathbf{x}_k - 0)^T \rangle = \langle \mathbf{x}_k {\mathbf{x}_k}^T \rangle = \boldsymbol{\Phi} \langle \mathbf{x}_{k-1} {\mathbf{x}_{k-1}}^T \rangle \boldsymbol{\Phi}^T + \mathbf{Q}$$

$$= \boldsymbol{\Phi} \left\langle \begin{bmatrix} 0 & 0\dot{x}_{k-1} \\ 0\dot{x}_{k-1} & \dot{x}_{k-1}\dot{x}_{k-1} \end{bmatrix} \right\rangle \boldsymbol{\Phi}^T + \mathbf{Q} \;\; = \boldsymbol{\Phi} \begin{bmatrix} 0 & 0 \\ 0 & \langle \dot{x}_{k-1}\dot{x}_{k-1} \rangle \end{bmatrix} \boldsymbol{\Phi}^T + \mathbf{Q}$$

$$\langle \dot{x}_{k-1}\dot{x}_{k-1} \rangle = q_c$$

$$P = \begin{bmatrix} \sigma_m^2 & p_{12} \\ p_{12} & p_{22} \end{bmatrix} = \begin{bmatrix} q_c & q_c \\ q_c & q_c \end{bmatrix} + \mathbf{Q} = q_c \begin{bmatrix} 1\frac{1}{3} & 1\frac{1}{2} \\ 1\frac{1}{2} & 2 \end{bmatrix} \longrightarrow q_c = \frac{3}{4}\sigma_m^2$$

Rule of thumb for dynamics noise Q

- Example of applying the rule of thumb
- Say: The expected change of target position is 10 pixels.
- Therefore the squared change is approximately:

  $\sigma_m^2 = 10^2$
- Then, applying the rule of thumb, the spectral density is $q_c = \frac{3}{4}10^2$
- So the dynamic model covariance is:

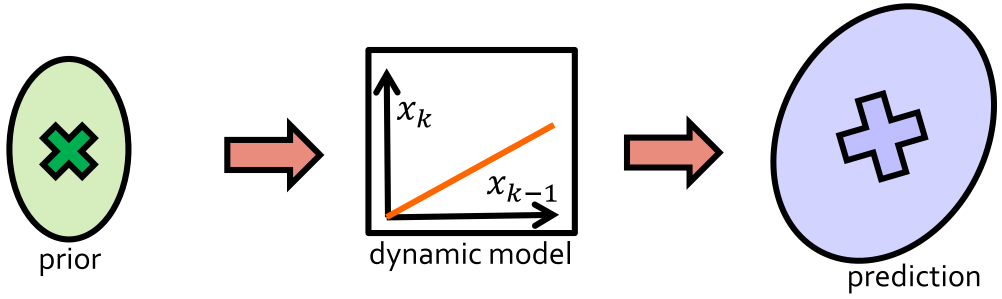
$$Q = q_c \begin{bmatrix} 1/3 & 1/2 \\ 1/2 & 1 \end{bmatrix} = \frac{3}{4}10^2 \begin{bmatrix} 1/3 & 1/2 \\ 1/2 & 1 \end{bmatrix}$$

Consider this an upper bound!

Beyond the basic Kalman

- Assumes linear dynamics with Gaussian noise
- + Simplifies the update equations
- - Cannot account for nonlinear dynamics

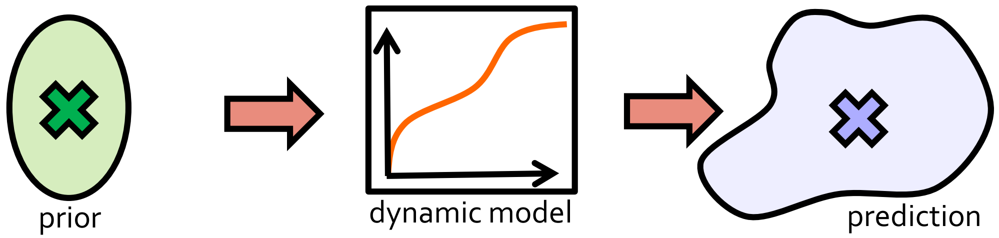
A linear dynamic model:



$$\int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\,\mathbf{x}_{k-1}$$

Integral easy to solve

A non-linear dynamic model:

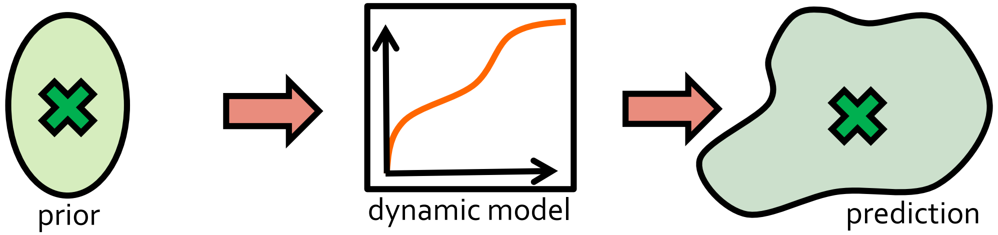


Integral not easy to solve

The result is not a Gaussian

Handling nonlinear dynamics

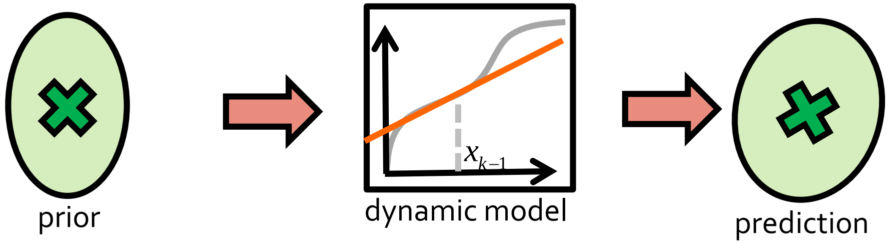
- Problem: prediction is no longer a Gaussian!



- Extended Kalman filter:

$$\int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\,\mathbf{x}_{k-1}$$

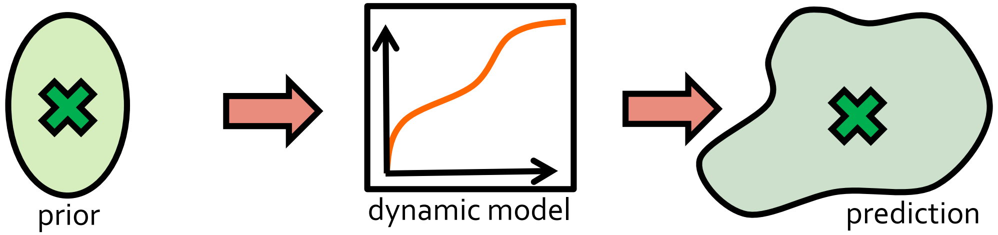
  - Linearize the dynamic model at x_{k-1}



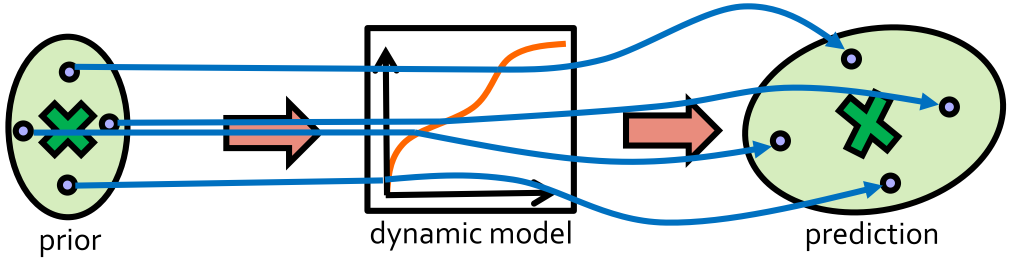
  - *Usually does not properly propagate the covariance*

Handling nonlinear dynamics

- Prediction no longer Gaussian!



- Unscented Kalman filter: $\int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, \mathrm{d}\,\mathbf{x}_{k-1}$
  - Numerically solve the integral using the Unscented transform



A nice summary of the main equations available here.

Wan, E.A.; Van Der Merwe, R., The unscented Kalman filter for nonlinear estimation, Proceedings of the IEEE ASSPCC 2000

See J.D. Prince "Computer vision: models, learning and inference", Section 19.4

References

- Text book Kalman:
  - S. J.D. Prince, “Computer vision: models, learning and inference”, Section 19.2

- For additional info on probability see:
  - S. J.D. Prince, “Computer vision: models, learning and inference”, Chapter 1

Acknowledgement

- Some images and parts of slides have been taken from the following talks:
  - Kevin Smith’s ”SELECTED TOPICS IN COMPUTER VISION – 2D tracking”