The Kalman Filter

COSC450

Pose Tracking

Pose Estimation

- Camera pose at each frame
- Computed fresh each time

Motion tracking

- Model of an object's motion
- Computed through a sequence
- Able to make predictions

Measurement vs modelling

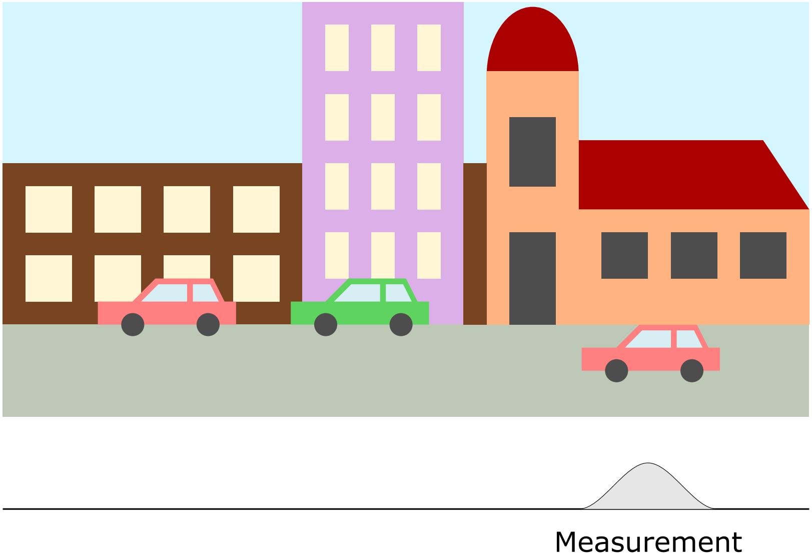
The typical model consists of:

- A measurement model
- A motion model
- A combining filter

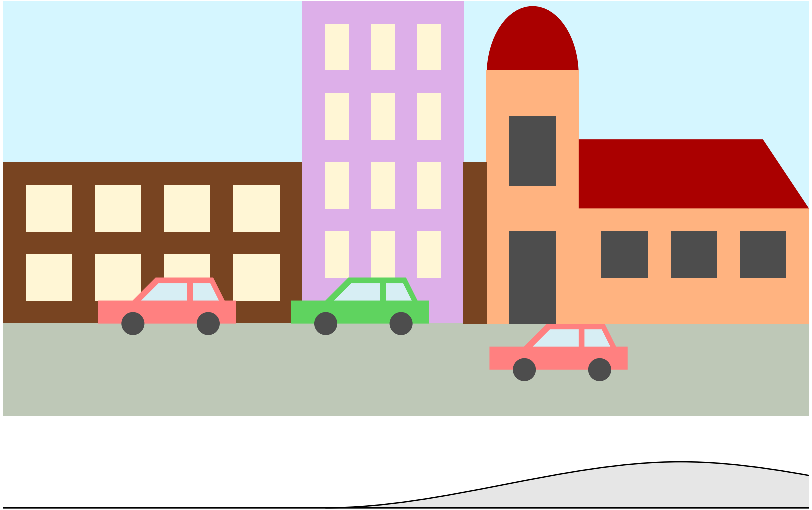
These are used to

- Make measurements
- Predict the future
- Weight these two parts

Example – Follow the Red Car

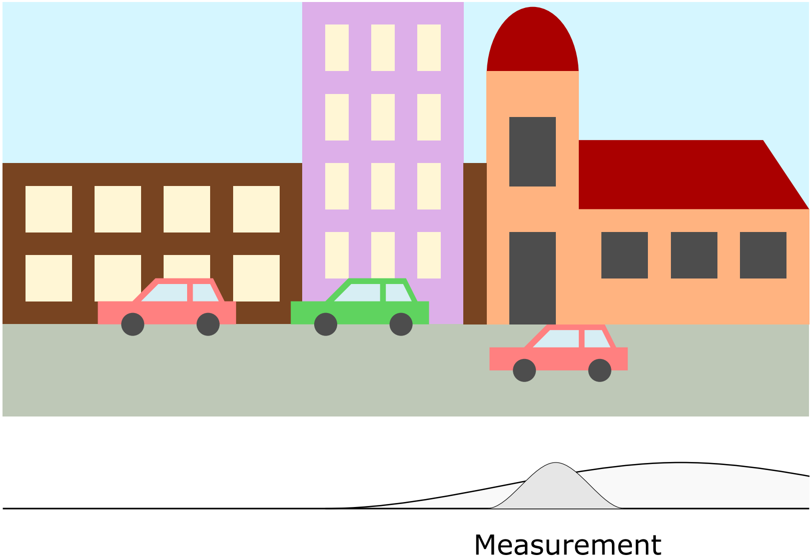


Example – Follow the Red Car

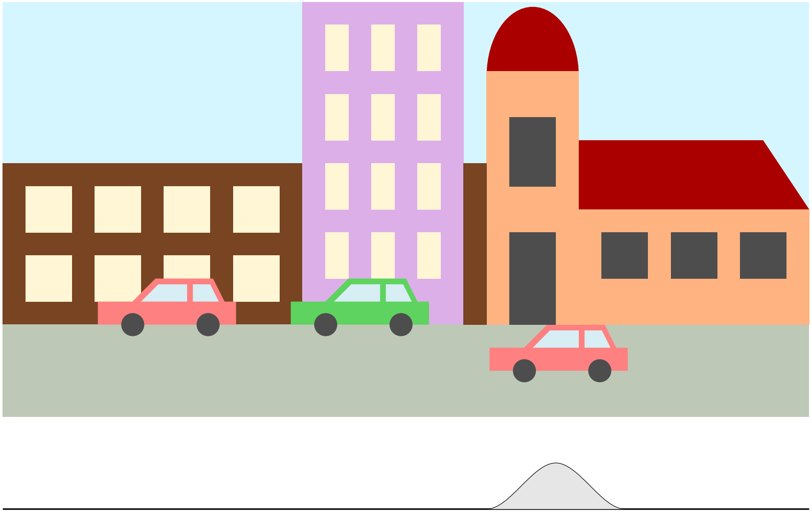


Prediction

Example – Follow the Red Car

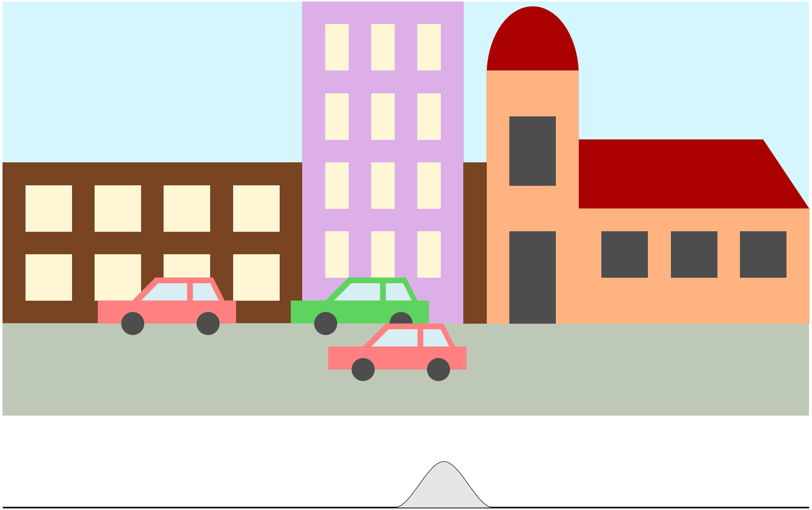


Example – Follow the Red Car



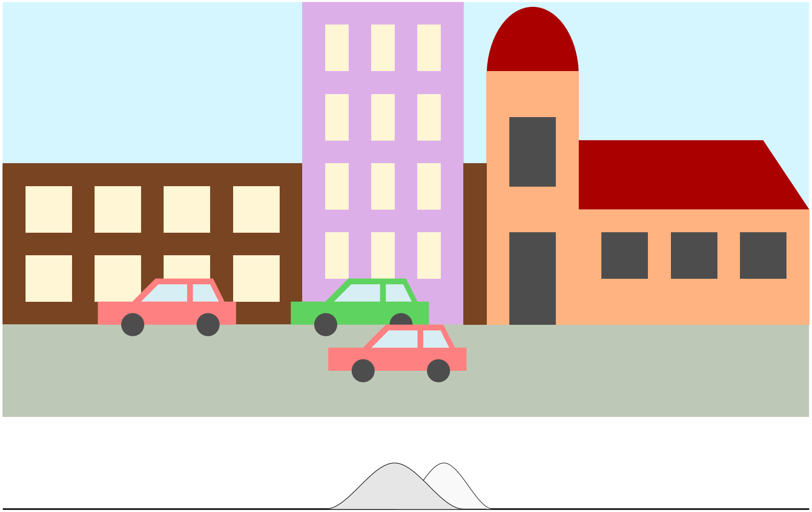
Combined

Example – Follow the Red Car



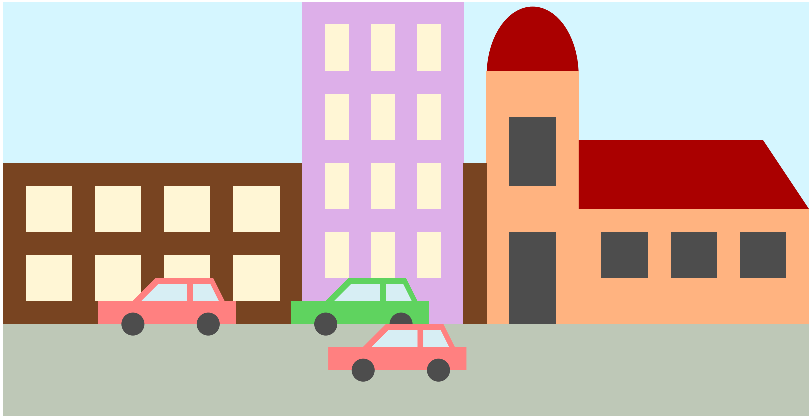
Prediction

Example – Follow the Red Car



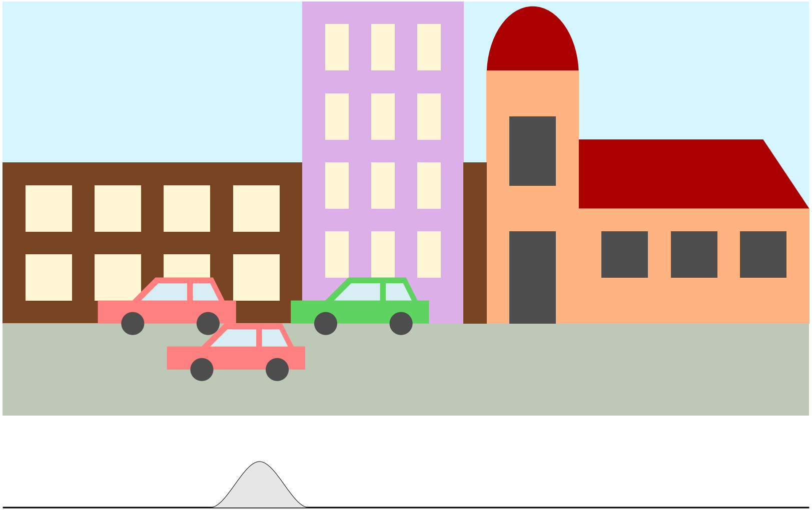
Measurement

Example – Follow the Red Car



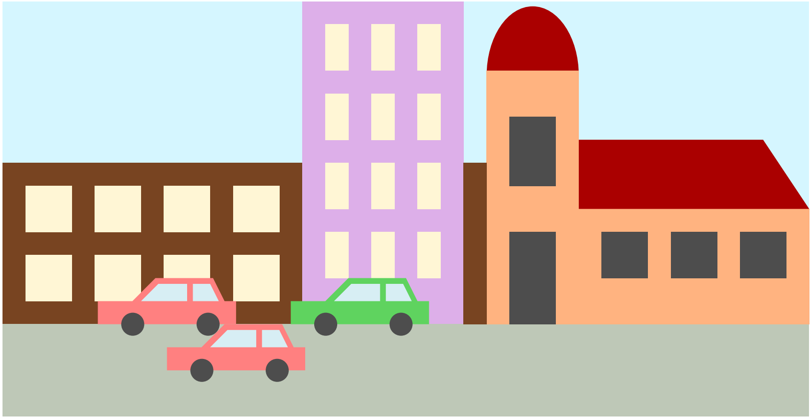
Combined

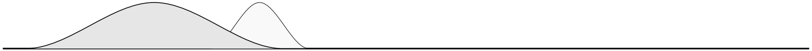
Example – Follow the Red Car



Prediction

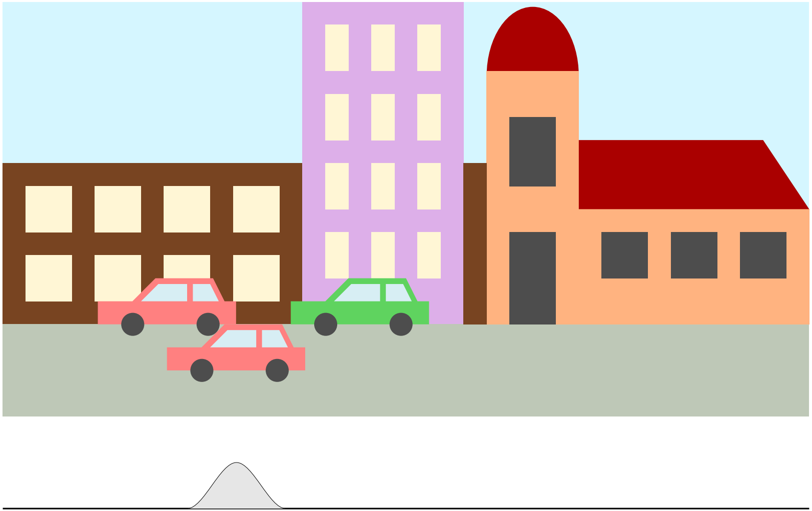
Example – Follow the Red Car





Measurement

Example – Follow the Red Car



Combined

A Simple Example

We have

- A state we want to estimate

$$\mathbf{s} = \begin{bmatrix} \text{location} \\ \text{velocity} \end{bmatrix} = \begin{bmatrix} x \\ v \end{bmatrix}$$

- A measurement model at each time, t,

$$m_t = x_t + r_t$$

- A motion model from time $t-1$ to t

$$\begin{aligned} x_t &= x_{t-1} + v_{t-1} + q_{x,t} \\ v_t &= v_{t-1} + q_{v,t} \end{aligned}$$

- q and r are error (noise) terms

The Kalman filter assumes

- A linear state update model

$$\mathbf{s}_t = \mathrm{A}\mathbf{s}_{t-1} + \mathbf{q}_t$$

- A linear measurement model

$$\mathbf{m}_t = \mathrm{B}\mathbf{s}_t + \mathbf{r}_t$$

- Random errors $\mathbf{q}_t$, $\mathbf{r}_t$
- Errors have Gaussian distributions
- Known (co)variance, Q_t, R_t
- Predicts the state and covariance, P_t
- Designed to minimise P_t

Kalman Filter Equations

Previous state estimate and covariance	$\tilde{\mathbf{s}}_{t-1}$	P_{t-1}
Predict next state and covariance	$\tilde{\mathbf{s}}_t^- = \mathrm{A}\tilde{\mathbf{s}}_{t-1}$	$\mathrm{P}_t^- = \mathrm{A}\mathrm{P}_{t-1}\mathrm{A}^\mathsf{T} + \mathrm{Q}_t$
Predict the measurement	$\tilde{\mathbf{m}}_t = \mathrm{B}\tilde{\mathbf{s}}_t^- = \mathrm{B}\mathrm{A}\tilde{\mathbf{s}}_{t-1}$	
Update with real − predicted measurement	$\tilde{\mathbf{s}}_t = \tilde{\mathbf{s}}_t^- + \mathrm{K}_t(\mathbf{m}_t - \tilde{\mathbf{m}}_t)$	$\mathrm{P}_t = \mathrm{P}_t^- - \mathrm{K}_t\mathrm{B}\mathrm{P}_t^-$

K_t is the Kalman gain – designed to minimise P_t:

$$\mathrm{K}_t = \mathrm{P}_t^-\mathrm{B}^\mathsf{T}\left(\mathrm{B}\mathrm{P}_t^-\mathrm{B}^\mathsf{T} + \mathrm{R}_t\right)^{-1}$$

Note: Many presentations include a control term (e.g. steering a robot)

$$\mathbf{s}_t = \mathrm{A}\mathbf{s}_{t-1} + \mathrm{C}\mathbf{u}_t + \mathbf{q}_t$$

Example: 2D Motion Constant Acceleration

- State: $\begin{bmatrix} x & y & u & v & a & b \end{bmatrix}^{\mathsf{T}}$
- Update equation:

$$\underbrace{\begin{bmatrix} x_t \\ y_t \\ u_t \\ v_t \\ a_t \\ b_t \end{bmatrix}}_{\mathbf{s}_t} = \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathrm{A}} \underbrace{\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ u_{t-1} \\ v_{t-1} \\ a_{t-1} \\ b_{t-1} \end{bmatrix}}_{\mathbf{s}_{t-1}}$$

- Measurement: $\begin{bmatrix} x & y \end{bmatrix}^{\mathsf{T}}$
- Measurement equation:

$$\underbrace{\begin{bmatrix} x_t \\ y_t \end{bmatrix}}_{\mathbf{m}_t} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathrm{B}} \underbrace{\begin{bmatrix} x_t \\ y_t \\ u_t \\ v_t \\ a_t \\ b_t \end{bmatrix}}_{\mathbf{s}_t}$$

Extended Kalman Filter

Kalman filter assumes linear models

- This is not always the case
- Non-linear state update measurement:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}) + \mathbf{q}_t \qquad \mathbf{m}_t = g(\mathbf{s}_t) + \mathbf{r}_t$$

- Replace A and B with Jacobians:

$$\mathrm{F}_{ij} = \frac{\partial f_i}{\partial s_j} \qquad \mathrm{G}_{ij} = \frac{\partial g_i}{\partial s_j}$$

- This makes a linear approximation

$$\begin{aligned} &\tilde{\mathbf{s}}_{t-1} && \mathrm{P}_{t-1} \\ \tilde{\mathbf{s}}_t^- &= f(\tilde{\mathbf{s}}_{t-1}) & \mathrm{P}_t^- &= \mathrm{F}\mathrm{P}_{t-1}\mathrm{F}^\mathsf{T} + \mathrm{Q}_t \\ \tilde{\mathbf{m}}_t &= g(\tilde{\mathbf{s}}_t^-) = g(f(\tilde{\mathbf{s}}_{t-1})) \\ \tilde{\mathbf{s}}_t &= \tilde{\mathbf{s}}_t^- + \mathrm{K}_t(\mathbf{m}_t - \tilde{\mathbf{m}}_t) & \mathrm{P}_t &= \mathrm{P}_t^- - \mathrm{K}_t\mathrm{G}\mathrm{P}_t^- \end{aligned}$$

$$\mathrm{K}_t = \mathrm{P}_t^-\mathrm{G}^\mathsf{T}\left(\mathrm{G}\mathrm{P}_t^-\mathrm{G}^\mathsf{T} + \mathrm{R}_t\right)^{-1}$$

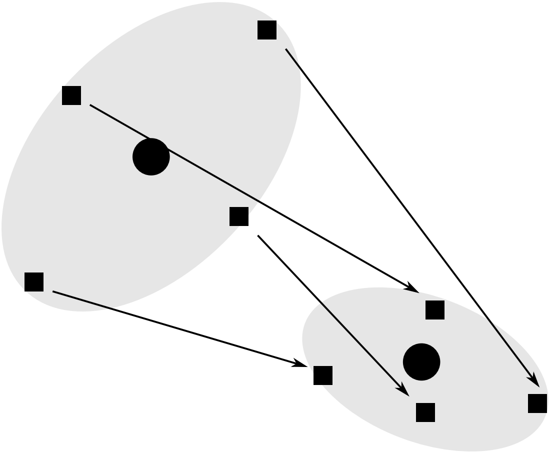
Unscented Kalman Filter

Kalman filter assumes known covariance

- This is often not true
- It is hard to estimate Q and R even in simple cases

UKF estimates covariance as it goes

- Samples points around the estimate
- Propagates them through state update and measurement
- Uses these to estimate covariance



Kalman Filter to Track Camera Pose

Even simple cases lead to lots of questions:

- State model – constant velocity

$$\mathbf{s} = \begin{bmatrix} r_1 & r_2 & r_3 & t_1 & t_2 & t_3 & \omega_1 & \omega_2 & \omega_3 & v_1 & v_2 & v_3 \end{bmatrix}^\mathsf{T}$$

  or constant acceleration

$$\mathbf{s} = \begin{bmatrix} r_1 & r_2 & r_3 & t_1 & t_2 & t_3 & \omega_1 & \omega_2 & \omega_3 & v_1 & v_2 & v_3 & \alpha_1 & \alpha_2 & \alpha_3 & a_1 & a_2 & a_3 \end{bmatrix}^\mathsf{T}$$

- Is a standard Kalman filter sufficient, or do you need an EKF?
- How best to represent rotations?
- How to determine the covariance matrices?
- How to tell if you have good choices?