Application – Traffic Tracking

- We want to track vehicles on a road
  - Eg: The truck in the images to the left
  - They are moving with a (fairly) constant velocity
  - In each frame we can measure the position of a feature on the vehicle we want to track

t=0

t=10

State Update Equation

- We assume the truck is moving with constant velocity
  - Our state is the truck position *(x,y)* and velocity *(u,v)*
    $s=[x,y,u,v]^T$
  - At each time the velocity adds on to the position

$$x_t = x_{t-1} + u_{t-1}$$
$$y_t = y_{t-1} + v_{t-1}$$
$$u_t = u_{t-1}$$
$$v_t = v_{t-1}$$

$$\begin{bmatrix} x_t \\ y_t \\ u_t \\ v_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ u_{t-1} \\ v_{t-1} \end{bmatrix}$$

$$s_t = As_{t-1}$$

Measurement Equation

- At each time we can detect features in the image
  - These make our measurements, m_t
  - We can directly measure the position of the truck, but not its velocity
- $m_t = [x,y]^T$

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ u_{t-1} \\ v_{t-1} \end{bmatrix}$$

$$m_t = Hs_t$$

An Initial Estimate

- The initial estimate of the state
  - We give a rough value of *x* and *y* to say which feature we are tracking
  - We probably won't have any idea about *u* and *v*
  - So we will use
    $s_0=[100,170,0,0]^T$
- We also need to give the (un)certainty
  - Our estimate of the position is good to within a few pixels
  - Our motion estimate is not good, but we expect the motion to be small
  - We represent this as a covariance matrix

Covariance Matrices

- So what is a covariance matrix?
  - It gives the relationships between sets of variables
  - The variance of a variable, *x*, is
    $var(x) = E((x-\bar{x})^2)$
  - The covariance of two variables, *x* and *y*, is
    $cov(x,y) = E((x-\bar{x})(y-\bar{y}))$
- Given a vector of variables
  $x=[x1,x2,\ldots,xk]$
  - The covariance , *C*, is a *k*×*k* matrix
  - The i,j^{th} entry of *C* is:
    $C_{i,j}=cov(x,y)$
  - A diagonal entry, $C_{i,i}$, gives the variance in the variable x_i
  - *C* is symmetric

Covariance in Noise

- The noise terms *v* and *w* need to be estimated
  - They have zero mean, and covariance *Q* and *R* respectively
  - We need an estimate of these matrices
  - *Q* and *R* say how certain we are about our model equations
- To estimate *Q*
  - Our initial estimate will be within a few pixels, say σ=3
  - The velocity is a bit less certain, but won't be large, say σ=5
  - There is no reason to think that the errors are related, so the covariance terms will be zero

Initial Covariance

$$P_0 = \begin{bmatrix} 9 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix}$$

- The variances of x and y are $3^2 = 9$
- The variances of u and v are $5^2 = 25$
- Since we assume independence the off-diagonal entries are all 0

Uncertainty in the Model

- Our model equations have noise terms
  - v represents the fact that our state update model may not be accurate
  - w represents the fact that measurements will always be noisy
  - We need to estimate their covariances
- In general
  - Often the terms will be independent. If this is the case the off-diagonal entries will be zero
  - Choosing the diagonal entries (varainces) is often more difficult

State Update Covariance

- The state update equation is not perfect
  - It assumes that the motion is constant but u and v might change over time
  - It assumes that all the motion is represented by u and v but other factors might affect x and y
- These errors will probably be small
  - The motion is slow and quite smooth
  - So the variance in these terms is probably a pixel or less, say σ = ½

$$Q = \begin{bmatrix} 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix}$$

State Update Covariance

- The measurements we make will be noisy
  - The features are located only to the nearest pixel
  - Because of image noise, aliasing, etc, they might be off by a pixel or so
- These errors are a bit easier to estimate
  - The feature is probably in the right place, or a pixel off
  - So the variance in these terms is probably $\sigma^2 = 1$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Predict the State

- We can now run the filter
  - First we make a prediction of the state at $t=1$ based on our initial estimate at $t=0$

$$s_1^- = As_0$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix}$$

Prediction Covariance

$$P_1^- = AP_0A^T + Q$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 & 0 & 0 & 0 \\ 0 & 9 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix}$$

$$= \begin{bmatrix} 34.25 & 0 & 25 & 0 \\ 0 & 34.35 & 0 & 25 \\ 25 & 0 & 35.25 & 0 \\ 0 & 25 & 0 & 25.25 \end{bmatrix}$$

Making a Measurement

- The state prediction gives us a guide to where the feature will be
  - We expect it to be near (100,170)
  - The variance in the *x* position is 34.25
  - The variance in the *y* position is 34.25 also
- We can use this to restrict our search for a feature
  - We are 95% certain that the feature lies in a circle of radius 2σ of the prediction

$$\sigma = \sqrt{34.25} \approx 5.85$$

  - We look for a feature in this region

Making a Measurement

- Within the search region
  - We compute a value that tells us how likely each point is to be a feature (Harris)
  - We find the point with the largest value within this region
  - This is $m_1 = [103,163]^T$

We look for a feature near our predicted value, and the covariances tell us how widely to search

The Kalman Gain

- We now combine the prediction and measurement
  - We compute the Kalman gain matrix
  - This takes into account the relative certainty of the two pieces of information

$$K_1 = P_1^- H^T \left(H P_1^- H^T + R\right)^{-1} \approx \begin{bmatrix} 0.972 & 0 \\ 0 & 0.972 \\ 0.709 & 0 \\ 0 & 0.709 \end{bmatrix}$$

  - The first components are close to 1, which will give more trust to the measurement

The Final Estimate

- We can now make a final state estimate
  - We combine the prediction and the measurement

- We also compute the covariance in this estimate
  - This can be used to tell us how far we can trust the estimate
  - It is also used to make a prediction for the next frame

The State Estimate

$$s_1 = s_1^- + K_1(m_1 - H s_1^-)$$

$$\approx \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.972 & 0 \\ 0 & 0.972 \\ 0.709 & 0 \\ 0 & 0.709 \end{bmatrix} \left(\begin{bmatrix} 103 \\ 163 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 100 \\ 170 \\ 0 \\ 0 \end{bmatrix} \right)$$

$$\approx \begin{bmatrix} 102.9 \\ 163.2 \\ 2.13 \\ -4.96 \end{bmatrix}$$

The State Covariance

$$P_1 = P_1^- + K_1 H P_1^-$$

$$\approx \begin{bmatrix} 34.25 & 0 & 25 & 0 \\ 0 & 34.25 & 0 & 25 \\ 25 & 0 & 25.25 & 0 \\ 0 & 25 & 0 & 25.25 \end{bmatrix} + \begin{bmatrix} 0.972 & 0 \\ 0 & 0.972 \\ 0.709 & 0 \\ 0 & 0.709 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 34.25 & 0 & 25 & 0 \\ 0 & 34.25 & 0 & 25 \\ 25 & 0 & 25.25 & 0 \\ 0 & 25 & 0 & 25.25 \end{bmatrix}$$

$$\approx \begin{bmatrix} 0.971 & 0 & 0.71 & 0 \\ 0 & 0.971 & 0 & 0.71 \\ 0.71 & 0 & 7.52 & 0 \\ 0 & 0.71 & 0 & 7.52 \end{bmatrix}$$

Iteration

- We repeat this computation for each frame
  - Over time the state predictions become more accurate
  - The Kalman gain takes this into account and places more weight on the predictions
- To implement the Kalman filter
  - We need a lot of matrix routines
  - These are tiresome to code by hand, but there are several libraries available
  - Only need basic operations: +, −, ×, transpose, inverse

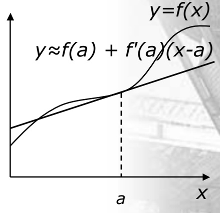
The Extended Kalman Filter

- The Kalman filter is limited by its assumptions
  - It assumes that all the noise/error terms are Gaussians with known (co)variance
  - It assumes that the model equations are linear
- Extended Kalman filters overcome the second assumption
  - They use a linear approximation to a non-linear function
  - They depend on the accuracy of this approximation
  - No proof, but they work well in practice

Linear Approximations

- If we have some function, $y = f(x)$
  - We can approximate this using

  $y \approx f(a) + f'(a)(x-a)$
  - a is any value we choose
  - This approximation is best when $x \approx a$



Linear Approximations

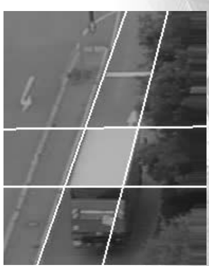
- If we have $z=f(x,y)$ we get

  $z \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b)$
  - f_x and f_y are the partial derivatives of f with respect to x and y
  - This approximates a 2D surface by a plane
- More generally, given $y = f(x_1, x_2, \dots, x_k)$ we have

  $y \approx f(a_1,a_2,\dots a_k)$
  $+ f_{x1}(a_1,a_2,\dots a_k)(x_1-a_1)$
  $+ f_{x2}(a_1,a_2,\dots a_k)(x_2-a_2)$
  …
  $+ f_{xk}(a_1,a_2,\dots a_k)(x_k-a_k)$

Example EKF

- Lines are detected with the Hough transform
  - The relationship between the states at subsequent times is non-linear
  - An extended Kalman filter allows us to track groups of lines with a common motion model



For More Information

- Videos of the truck example are available on the website
- Tracking lines with the EKF
  - *Tracking in a Hough Space with the Extended Kalman Filter*, Steven Mills, Tony Pridmore, and Mark Hills, Proceedings of the British Machine Vision Conference (BMVC2003), pages 173-182, 2003.
- A useful Java matrix library is JAMA

  http://math.nist.gov/javanumerics/jama/