# Lecture05b: Kalman Filters

CS540 2/15/18

## Announcements

On-campus students:
- Make sure I am wearing the microphone

All students:
- How are the projects coming?
  - Due one week from today
  - No more extensions
- One week from today will be project presentations
  - Send me Powerpoint or pdf files by Wednesday night

***Are there any questions?***

## Where were we?

We were talking about evolutionary algorithms
- Population based
- Problem components
  - State representation
  - Evaluation function
- Three algorithm components
  - Selection
  - Cross-over
  - Mutation
  - Not all versions use all three…
- Issues
  - Diversity / exploration
  - Memory (history)
  - Initial population
  - Efficiency

## Yes, but where were we?

We started talking about dynamic problems
- Where the evaluation function is a function of time
- For example tracking (objects move)

This led to particle filters
- Partially observable states
- Prediction model added to selection
- Crossover optional
  - Often used when tracking a single target (or the camera motion)
  - Not commonly used for tracking multiple targets

But what is the accepted best practice for predicting future states?

## Kalman Filters

Assume constant velocity
- Constant acceleration models are also possible

Then the (motion) *state* of a target is:

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

By definition, the state contains enough information to predict $x_{t+1}$ from $x_t$.
- Subscripts represent time
- Linear system → predict new state from previous state linearly
  - Note: linear in the state representation
  - State may contain x, x′, x′′, x′′′….



## Prediction

In general, future states are stochastically drawn from

$$x_{t+1} = Fx_t + Q$$

F is the motion model
- For constant velocity, $F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Q is a covariance matrix representing the noise
- Because the motion model is wrong
- Because observations contain noise
- Q is diagonal if errors are independent
- Q is 4x4 in our example (outer product of states)

## Prediction (II)

But we don't know the noise, so we estimate:

$$x_{t+1|t} = Fx_t$$

$x_{t+1|t}$ tells us where to look for the target



## Predicting Uncertainty

Let $P_t$ be a matrix representing our uncertainty at time t
- Initialize: $P_1 = Q$

Then we estimate as follows:

$$P_{t+1|t} = FP_tF^T + Q$$

- F propagates uncertainty across terms
  - If you were unsure about Δx, you become unsure of x
- Q adds in new uncertainty (randomness)

$P_{t+1|t}$ tells us how big a search window to use



## Updating based on observations

Now we get an observation $z_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix}$

Problems
- Observations are partial
- observations have noise, too

Let $z_t = Hx_t + R$

Where $H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ — H selects the observable part of the state

And where R is another noise matrix
- 2x2 in our example (outer product of z's)
- Possibly diagonal
- Similar to Q

— R models observation error



## Update (II)

The measurement residual is

$$y_{t+1} = z_{t+1} - Hx_{t+1|t}$$

*Predicted next state*

*Residual: observation - prediction*

*Observed next state*

*Select observable part*

The residual covariance is

$$S_{t+1} = HP_{t+1|t}H^T + R$$

*Measurement uncertainty: old + new*

Observable part of Previous uncertainty

*Uncertainty in new measurement*

The optimal Kalman Gain is

$$K_{t+1} = P_{t+1|t}H^T S_{t+1}^{-1}$$

Observable part of Previous uncertainty

*Measurement uncertainty*



## A word about Kalman Gain

Remember this is all about linear dynamics

Our new state estimate should be a weighted average of
- The predicted state and
- The observation

The Kalman gain calculates this weight
- Using observation noise estimates
- And motion noise estimates
- Assuming noise is Gaussian
- And initial estimates of Q and R

K is 4x2



## Update (III)

Now we can update our state estimates!

$$x_{t+1} = x_{t+1|t} + K_{t+1}y_{t+1}$$

$$P_{t+1} = (I - K_{t+1}H)P_{t+1|t}$$

# Tracking with Kalman Filters

1. Set $Q = \begin{bmatrix} \sigma_x & 0 & 0 & 0 \\ 0 & \sigma_y & 0 & 0 \\ 0 & 0 & \sigma_{\Delta x} & 0 \\ 0 & 0 & 0 & \sigma_{\Delta y} \end{bmatrix}$
   - Pixel sigmas represent random positional error
   - Delta sigmas represent variations from constant velocity
2. Set $R = \begin{bmatrix} \sigma_m & 0 \\ 0 & \sigma_m \end{bmatrix}$
   - Sigmas represent positional matching error
3. Find target in 2 consecutive frames
4. Estimate $x_1 = \begin{bmatrix} x_1 \\ y_1 \\ \Delta x = x_1 - x_0 \\ \Delta y = y_1 - y_0 \end{bmatrix}$
5. Set $P_1 = Q$
6. Set $S_1 = R$



# Tracking with Kalman Filters (II)

Now, for frames t >= 1

- Predict new states
  - $x_{t+1|t} = Fx_t$
  - $P_{t+1|t} = FP_tF^T + Q$
- Look for match in frame t+1, creating $z_{t+1}$
- Update state predictions
  - $y_{t+1} = z_{t+1} - Hx_{t+1|t}$
  - $S_{t+1} = HP_{t+1|t}H^T + R$
  - $K_{t+1} = P_{t+1|t}H^TS_{t+1}^{-1}$
  - $x_{t+1} = x_{t+1|t} + K_{t+1}y_{t+1}$
  - $P_{t+1} = (I - K_{t+1}H)P_{t+1|t}$



# Particle Filters

In practice, many particle filters use a population of Kalman filters

- Kalman filters come with a prediction model
- Kalman filters smooth out observation noise

Issues

- Cost
  - not too bad, if carefully implemented
  - Only a few real matrix multiplies (and they're small)
- Initialization of new tracks
  - Requires detections in back to back frames
- Appropriate mutation operators