Recap

Dimensionality Reduction

Here’s how the EVecs may look like:

<table>
<thead>
<tr>
<th>EV 1</th>
<th>EV 2</th>
<th>EV 3</th>
<th>EV 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV 5</td>
<td>EV 6</td>
<td>EV 7</td>
<td>EV 8</td>
</tr>
<tr>
<td>EV 9</td>
<td>EV 10</td>
<td>EV 11</td>
<td>EV 12</td>
</tr>
<tr>
<td>EV 13</td>
<td>EV 14</td>
<td>EV 15</td>
<td>EV 16</td>
</tr>
</tbody>
</table>

Feature Selection

Treat each feature as a binary classifier.

Pick a threshold $\theta$ that minimizes the FP rate.
Today

Tracking: Intro
Mean-shift and CAMShift
Kalman Filtering, Particle Filtering
Optical Flow
HW3! Yay!
So far we’ve examined just discrete images.

Now let’s think about video: Images + time

We’re adding the time domain.

\[ I(x,y,t) \]
Tracking

Pixel (brightness) tracking:
- Track a pixel in time: \( I(x,y,t) \rightarrow I(x',y',t+1) \)
- Track all the pixels in the image
- Compare brightness / windows

Feature tracking:
- Close to image alignment that we saw
- Sparse tracking, only selected key-points
- Compare descriptors, not intensity

Contour / Shape tracking:
- Higher level information, no longer pixels
- Lines, edges

Tracking: Find the position of the object in the scene across time.

Path of objects in 3D coordinates: “sausages”
Human Intuition

Where is he going next?
Human Intuition

Where is he going next?
Human Intuition

Where is he going next?

Did you guess correctly?

Why did you make that guess?

What made you think that?

What happened in the image?
Human Intuition

How about the objects in this scene?
Human Intuition

What is moving?

How does that affect the pixel brightness change?
Changing Intensity

Change in intensity is not always from the object-of-interest motion, can be:

- camera motion,
- lights motion,
- other objects motion,
- object deformation (but not motion),
- optics,
- noise,
- even thermal blur effect!
- ...

Bottom line: Tracking is a hard problem.

Look at the marked region. What’s the motion within it?
The Barber Shop Pole Illusion

Aperture Illusion

When we look at a very small region and try to estimate motion.

What’s the direction of motion?
Aperture Illusion

When we look at a very small region and try to estimate motion.
Aperture Illusion

When we look at a very small region and try to estimate motion.
The Barber Shop Pole Illusion

Aperture Illusion

When we look at a very small region and try to estimate motion.

How about now?

So is it moving right or down?
The Barber Shop Pole Illusion

Aperture Illusion

When we look at a very small region and try to estimate motion.

How about now?

Insight: The window of sampling around our point makes a big difference...
Tracking-by-Detection

Simplest idea:

- We can detect objects in images, right?
- So we can detect the object in every frame and thus track it.

Sounds dead simple and foolproof, what could go wrong? **What are the problems?**

Think about the challenges of detection...
(also look at the video to the right :)
Probabilistic Framework for Tracking

**Measurement model**: (generative, model the data)

\[ Pr(x_t|w_t) \]

The observed data at time \( t \) depends on the (hidden) world state at time \( t \).

**Temporal model**:

\[ Pr(w_t|w_{t-1}) \]

Markovianity: The state depends only on the previous state.

Our goal is to infer the world state given the (noisy?) measurements up until now. The marginal posterior:

\[ Pr(w_t|x_{1..t}) \]
Probabilistic Framework for Tracking

We can expand the posterior with Bayes rule:

\[ Pr(w_t | x_1...t) = \frac{Pr(x_t | w_t)Pr(w_t | x_1...t-1)}{\int Pr(x_t | w_t)Pr(w_t | x_1...t-1)dw_t} \]

denormalization

This: \( Pr(w_t | x_1...t-1) \) is a prediction, since we try to model something “in the future” based on data that we haven’t seen yet.

We can also think of it as a prior: what we know / expect about the upcoming behaviour of \( w_t \)

So in essence we have a two step process:

- Predict (prior), based on data up to now
- Estimate, combine prior and likelihood of current data
Mean-shift Tracking

Let’s take a simple way to estimate the posterior:

- Simple temporal model, only looks at last data point: $Pr(w_t|x_{1\ldots t-1}) = Pr(w_t|x_{t-1})$
- Simple measurement model: Maximum Likelihood of a gaussian = the mean $\mu$.

Assume $Pr(x_t|w_t) \sim \text{Norm}(\mu, \sigma)$. We want to find the point where it is maximal.

We estimate the PDF of $Pr(x_t|w_t)$ by iteratively looking at the weighted mean in a search window, and march towards a convergence point.
Mean-shift Tracking

Start with a good initial guess (our prior $Pr(w_t|x_{t-1})$), and calculate the weighted mean based on a confidence map.

When the march converges around a point of maximal likelihood, set the new $w_t$ to be that point.

Since we shift towards the mean, the method is called “mean-shift”.

Skin color model: 
32x32x32 histogram in HSV built offline
Mean-shift Tracking

Start with a good initial guess (our prior $Pr(w_t|x_{t-1})$), and calculate the weighted mean based on a confidence map.

When the march converges around a point of maximal likelihood, set the new $w_t$ to be that point.

Since we shift towards the mean, the method is called “mean-shift”.
One confidence map we could use is based on (skin) color.

If the first frame we calculate the color histogram in the H domain (from HSV), and keep it.

In subsequent frames we make a prediction for each image pixel value based on the histogram, essentially taking the histogram bin value.

The histogram essentially is an explicit PDF for:

$$Pr(x \mid y = \text{skin})$$

(generative, models the data)
Mean-shift Tracking

What are the problems with such a mean-shift tracker?

- Window size is constant.
- Chosen probabilistic model (color histogram) not adaptive. Relies on a calculation made in the first frame (*gasp*).
- Must be able to “march” from a good initial guess, otherwise if too far - will not converge.

What can we do? (to tackle each problem)
CAMShift

An improvement for mean-shift:

- Perform mean-shift as usual
- Recalculate the color histogram in the final region and update the confidence map, and run mean-shift again until convergence
- Use an elliptical model to recalculate the size and position of the square search region.
CAMshift

Problems:

- Underlying is still mean-shift.
- Needs good initial guess
- Hard to know when lost tracking, because of adaptive calculation.

Still - a very fast tracker.
Questions?
Kalman Filter

Let’s take a different model for the measurement and temporal models.

**Measurement:**

\[ Pr(x_t|w_t) \sim \text{Norm}(\mu_m + \Phi w_t, \Sigma_m) \]

The observed data is linearly related to the world state, with normally distributed additive noise.

**Temporal:**

\[ Pr(w_t|w_{t-1}) \sim \text{Norm}(\mu_p + \Psi w_{t-1}, \Sigma_p) \]

The world state is linearly dependant on the previous world state with normally distributed additive noise.

**Insight:** We can represent the transitions in data and world state with some matrix multiplications.

**Note:** The KF is a heavy concept. We’re only going to see it in high-level. The details are in your readings.
In the Kalman model:

The new world state can be calculated by multiplying the old world state with the state transition matrix (and adding some Gaussian noise).

The data can be calculated from the world state by multiplying with the relationship matrix (and adding some Gaussian noise).
Kalman Filter

Remember our goal:

\[
Pr(w_t|x_{1\ldots t}) = \frac{Pr(x_t|w_t)Pr(w_t|x_{1\ldots t-1})}{\int Pr(x_t|w_t)Pr(w_t|x_{1\ldots t-1})dw_t}
\]

- we can calculate the prior (prediction) with the state transition matrix
- and then calculate the likelihood of the data

[Credit: Simon Prince]
Kalman Filter

Inference

State Prediction:  \[ \mu_+ = \mu_p + \Psi \mu_{t-1} \]

Covariance Prediction:  \[ \Sigma_+ = \Sigma_p + \Psi \Sigma_{t-1} \Psi^T \]

State Update:  \[ \mu_t = \mu_+ + K(x_t - \mu_m - \Phi \mu_+) \]

Covariance Update:  \[ \Sigma_t = (I - K\Phi) \Sigma_+ , \]

\[ K = \Sigma_+ \Phi^T (\Sigma_m + \Phi \Sigma_+ \Phi^T)^{-1} \]

The Kalman Gain: determines how much the measurements contribute to the state estimate. If it’s small - measurements are unreliable, if larger - measurements are more reliable than prior.

[Credit: Simon Prince]
Kalman Filter | Example

No Kalman Filter: Posterior based only on last data point and prior.

Kalman Filter: Posterior based on state transition and data likelihood.

Data points are just world state + some noise:

\[ Pr(w_t|w_{t-1}) = \text{Norm}_{w_t}[w_{t-1}, \sigma_p^2 I] \]

\[ Pr(x_t|w_t) = \text{Norm}_{x_t}[w_t, \Sigma_m] \]

[Credit: Simon Prince]
The Kalman Filter can fill-in the gaps in measurements using its predictive step.

It also smoothes the jitter by combining noisy measurements with prior internal state.
Problems:

- Linear model. Requires all state and measurement transitions to be linear.
- Models the system in a unimodal gaussian. Oftentimes noise is not normally distributed.
- Assumes a-priori motion model, can’t adapt to new movement behaviour.
- Has no “memory” of past states
- Always looks in the past and not the future (in case the future is known, it can’t be used..)

Some “patches” the KF available:

- Non-linear models: Extended KF and Unscented KF
- Memory: Lagged state models
- Lookahead
Questions?
Particle Filter

The KF modeled our state PDF \textit{explicitly} with a Gaussian. However PDF may be much more complex!

Key idea:

The PF suggests an \textit{implicit} model of the PDF, by way of sampling the state parametric space, using a multitude of “particles”.

\[ Pr(w_{t-1}|x_{1...t-1}) = \sum_{j=1}^{J} a_j \delta[w_{t-1} - \hat{w}_{t-1}^j] \]

Parties: Hypotheses of the state

Advantage:

- Can model virtually any PDF, not just unimodal normal.
- Support multiple state hypotheses at the same time, unlike the KF.
CONDENSATION

CONditional DENsity propagaTION
[Blake, Isard 1998]

**Step 1:** Sample the parameter space.

Initialize with some distribution (uniform, normal)
CONDENSATION

CONditional DENsity propagaTION

**Step 2:** Apply temporal model and add process noise.

Similar to KF state transition.
Step 3: Evaluate the samples ("particles") and establish state posterior $Pr(w_t|x_1...t)$

Use posterior as prior for next round of sampling.
How does that look like in practice?

[Credit: Simon Prince]

[Blake, Isard 1998]
CONDENSATION

How does that look like in practice?

“My” particle filter.

200 particles.

Particles evaluate the color histogram backpropagation.

**How do we move the particles?** (the state transition) **What’s a good strategy for motion?**
CONDENSATION

How does that look like in practice?

```python
def resample(weights):
    n = len(weights)
    indices = []
    C = [0. + sum(weights[:i+1]) for i in range(n)]
    u0, j = random(), 0
    for u in [(u0+i)/n for i in range(n)]:
        while u > C[j]:
            j+=1
        indices.append(j-1)
    return indices

def particlefilter(sequence, pos, stepsize, n):
    seq = iter(sequence)
    x = ones((n, 2), int) * pos  # Initial position
    f0 = seq.next()[tuple(pos)] * ones(n)  # Target colour model
    yield pos, x, ones(n)/n  # Return expected position, particles and weights
    for im in seq:
        np.add(x, uniform(-stepsize, stepsize, x.shape), out=x, casting="unsafe")  # Particle motion model: uniform step
        f = im[tuple(x.T)]  # Measure particle colours
        w = 1./(1. + (f0-f)**2)  # Weight- inverse quadratic colour distance
        w /= sum(w)  # Normalize w
        yield sum(x.T*w, axis=1), x, w  # Return expected position, particles and weights
    if 1./sum(w)**2 < n/2.:
        x = x[resample(w), :]
        # Resample particles according to weights
```

10-20 LoPC
Particle Filter | Importance (Re)Sampling

When particles are no longer representing the underlying PD - resample.

When does that happen?

Number of effective particles:

\[ \hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2} \]

Pick a threshold.
Questions?
HW3: Detection and Tracking

Your goal is to:

- Detect the face in the first frame of the movie
  - Using pre-trained Viola-Jones detector
- Track the face for 20 frames using:
  - CAMShift
  - Particle Filter
  - Face detector + Kalman Filter
  - Mean Optical Flow (next class)

For each frame write out the tracking results to a text file, in a particular format.

Due: Tue 10/17 9am (day of the midterm)

Skeleton and bootstrap code will be provided asap.
Wrap Up

Next:
- Segmentation
- Mid term