Announcements

• HW4 assigned, due on 12/3
• Course evaluations today
• Will discuss final exam next lecture

Main tracking notions

• State: usually a finite number of parameters (a vector) that characterizes the “state” (e.g., location, size, pose, deformation of thing being tracked).
• Dynamics: How does the state change over time? How is that changed constrained?
• Representation: How do you represent appearance of the thing being tracked?
• Prediction: Given the state at time \( t-1 \), what is an estimate of the state at time \( t \)?
• Correction: Given the predicted state at time \( t \), and a measurement at time \( t \), update the state.
• Initialization – what is the state at time \( t=0 \)?

What is state?

• 2-D image location, \( \Phi=(u,v) \)
• Image location + scale \( \Phi=(u,v,s) \)
• Image location + scale + orientation \( \Phi=(u,v,s,\theta) \)
• Affine transformation
• 3-D pose
• 3-D pose plus internal shape parameters (some may be discrete).
  – e.g., for a face, 3-D pose + facial expression using FACS + eye state (open/closed).
• Collections of control points specifying a spline
• Above, but for multiple objects (e.g. tracking a formation of airplanes).
• Augment above with temporal derivatives \( (\dot{\Phi}, \ddot{\Phi}) \)

State Examples:

– object is a ball: state is 3D position+velocity, measurements are stereo pairs
– object is person: state is body configuration, measurements are image frames
– What is the state here?

Example: Blob Tracker

• From input image \( I(u,v) \) (color?) at time \( t \), create a binary image by applying a function \( f(I(u,v)) \).
• Clean up binary image using morphological operators
• Perform connected component exploration to find “blobs.” – connected regions.
• Compute their moments (mean and covariance of coordinates of region), and use as state
• Using state estimate from \( t-1 \) and perform “data association” to identify state in from \( t \).
**Blob Tracking in IR Images**
- Threshold about body temperature
- Connected component analysis
- Position, scale, orientation of regions
- Temporal coherence

**Tracking: Probabilistic framework**
- Very general model:
  - We assume there are moving objects, which have an underlying state $X$
  - There are measurements $Y$, some of which are functions of this state
  - There is a clock
    - at each tick, the state changes
    - at each tick, we get a new observation

**Tracking State**

**Three main steps**
- **Prediction**: we have seen $y_0, \ldots, y_{i-1}$ — what state does this set of measurements predict for the $i$th frame? to solve this problem, we need to obtain a representation of $P(X_i | Y_0 = y_0, \ldots, Y_{i-1} = y_{i-1})$.
- **Data association**: Some of the measurements obtained from the $i$-th frame may tell us about the object’s state. Typically, we use $P(X_i | Y_0 = y_0, \ldots, Y_{i-1} = y_{i-1})$ to identify these measurements.
- **Correction**: now that we have $y_i$, the relevant measurements — we need to compute a representation of $P(X_i | Y_0 = y_0, \ldots, Y_{i-1} = y_{i-1}, Y_i = y_i)$.

We can try to express these conditional distributions parametrically, sample the distribution, or estimate the mode.

**Simplifying Assumptions**
- Only the immediate past matters: formally, we require $P(X_i | X_1, \ldots, X_{i-1}) = P(X_i | X_{i-1})$
- Measurements depend only on the current state: we assume that $Y_i$ is conditionally independent of all other measurements given $X_i$. This means that $P(Y_i | Y_1, \ldots, Y_{i-1}, X_i) = P(Y_i | X_i)$

**Tracking as induction**
- Assume data association is done
  - Sometimes challenging in cluttered scenes. See work by Christopher Rasmussen on Joint Probabilistic Data Association Filters (JPDAF).
- Do correction for the 0'th frame
- Assume we have corrected estimate for i’th frame
  - show we can do prediction for i+1 frame, correction for i+1 frame
Base case

$P(y | x)$ is our observation model. Given the state $x$, what is the observation. For example, $P(y | x)$ might be a Gaussian with mean $x$.

Firstly, we assume that we have $P(X_0)$

Prior distribution of initial state

$$P(X_0 | Y_0 = y_0) = \frac{P(y_0 | X_0)P(X_0)}{P(y_0)}$$

By Bayes rule

$$= \int P(y_0 | X_0)P(X_0) dX_0$$

$$\propto P(y_0 | X_0)P(X_0)$$

Induction step

Induction step

Induction step

Induction step

Integration involves obtaining a representation of

Our independence assumptions make it possible to write

$$P(X_i | Y_0, \ldots, Y_{i-1}) = \int P(X_i | X_{i-1}, X_{i-2}, \ldots, X_0, Y_0, \ldots, Y_{i-1})P(X_{i-1} | X_{i-2}, \ldots, X_0, Y_0, \ldots, Y_{i-2}) dX_{i-1}$$

$$= P(y_i | X_i)P(X_i | X_{i-1}, X_{i-2}, \ldots, X_0, Y_0, \ldots, Y_{i-1}) P(X_{i-1} | X_{i-2}, \ldots, X_0, Y_0, \ldots, Y_{i-2})$$

$$= P(y_i | X_i)P(X_i | X_{i-1}, X_{i-2}, \ldots, X_0, Y_0, \ldots, Y_{i-1})$$

$$= P(y_i | X_i)P(X_i | X_{i-1}, X_{i-2}, \ldots, X_0, Y_0, \ldots, Y_{i-1})$$

$$\int P(y_i | X_i)P(X_i | X_{i-1}, X_{i-2}, \ldots, X_0, Y_0, \ldots, Y_{i-1}) dX_i$$

Linear dynamic models

- Use notation $\sim$ to mean “has the pdf of”. $N(a, b)$ is a normal distribution with mean $a$ and covariance $b$.
- A linear dynamic model has the form
  $$x_i \sim N(D_{i-1} x_{i-1}, \Sigma_{i-1})$$
  $$y_i \sim N(M x_i, \Sigma_y)$$

Examples

- Points moving with constant velocity
- Periodic motion
- Etc.
- Points moving with constant acceleration
Points moving with constant velocity

- We have
  \[
  u_i = u_{i-1} + \Delta t v_{i-1} + \epsilon_i \quad \text{Position}
  \]
  \[
  v_i = v_{i-1} + \xi_i \quad \text{Velocity}
  \]
  – (the Greek letters denote noise terms)
- Stack \((u, v)\) into a single state vector
  \[
  \begin{pmatrix}
  u_i \\
  v_i \\
  \end{pmatrix} = \begin{pmatrix}
  1 & \Delta t \\
  0 & 1
  \end{pmatrix} \begin{pmatrix}
  u_{i-1} \\
  v_{i-1}
  \end{pmatrix} + \text{noise}
  \]
  which is the form we had above

Points moving with constant acceleration

- We have
  \[
  u_i = u_{i-1} + \Delta t v_{i-1} + \epsilon_i
  \]
  \[
  v_i = v_{i-1} + \Delta t a_{i-1} + \xi_i
  \]
  \[
  a_i = a_{i-1} + \xi_i
  \]
  – (the Greek letters denote noise terms)
- Stack \((u, v, a)\) into a single state vector
  \[
  \begin{pmatrix}
  u_i \\
  v_i \\
  a_i
  \end{pmatrix} = \begin{pmatrix}
  1 & \Delta t & 0 \\
  0 & 1 & \Delta t
  \end{pmatrix} \begin{pmatrix}
  u_{i-1} \\
  v_{i-1} \\
  a_{i-1}
  \end{pmatrix} + \text{noise}
  \]
  – which is the form we had above

The Kalman Filter

- Key ideas:
  - Linear models interact uniquely well with Gaussian noise - make the prior Gaussian, everything else Gaussian and the calculations are easy
  - Gaussians are really easy to represent – Just a mean vector mean and covariance matrix.

The Kalman Filter in 1D

- Dynamic Model
  \[
  x_i \sim N(m_i, \sigma_i^2)
  \]
  \[
  y_i \sim N(m_i, \sigma_i^2)
  \]
- Notation
  \[
  \text{mean of } P(X_i|y_0, \ldots, y_i) \text{ as } \bar{x}_i \quad \text{Predicted mean}
  \]
  \[
  \text{Corrected mean} \quad \text{mean of } P(X_i|y_0, \ldots, y_i) \text{ as } \bar{x}_i
  \]
  \[
  \text{the standard deviation of } P(X_i|y_0, \ldots, y_i) \text{ as } \sigma_i
  \]
  \[
  \text{of } P(X_i|y_0, \ldots, y_i) \text{ as } \sigma_i
  \]

Prediction for 1-D Kalman filter

- The new state is obtained by
  - multiplying old state by known constant
  - adding zero-mean noise
- Therefore, predicted mean for new state is
  - constant times mean for old state
- Predicted variance is
  - sum of constant squared times old state variance and noise variance

Because:
- Old state is normal random variable.
- Multiplying normal rv by constant implies
  - mean is multiplied by a constant
  - variance is multiplied by square of constant
- Adding zero mean noise adds zero to the mean, Adding rv’s adds variance
Correction for 1D Kalman filter

- Pattern match to identities given in book
  - basically, guess the integrals, get:

$$x_k = \left( \frac{\sigma_{x_k}^2 \sigma_{y_k}^2}{\sigma_{x_k}^2 + m_2 \sigma_{y_k}^2} \right)$$

- Notice:
  - if measurement noise is small, we rely mainly on the measurement,
  - if it’s large, mainly on the prediction

Multi-variate Kalman Filter

Dynamic Model:
- $x_k \sim N(x_{k-1}, Q_k)$
- $w_k \sim N(0, Q_w)$

Start Assumptions: $\Sigma_0$ and $\Sigma_k$ are known

Update Equations:
- Prediction:
  $$\hat{x}_k = F x_{k-1}$$
  $$\Sigma_k = F \Sigma_{k-1} F^T + Q_k$$

- Correction:
  $$K_k = \Sigma_k M_k^T (M_k \Sigma_k M_k^T + R_k)^{-1}$$
  $$\hat{x}_k = \hat{x}_k + K_k (y_k - M_k \hat{x}_k)$$
  $$\Sigma_k = (I - K_k M_k) \Sigma_k$$

Tracking Modalities
(Define the features $Y$)

- Color
  - Histogram [Birchfield 1998; Bradski 1998]
  - Volume [Bliss et al., 1993; Bregler, 1997; Darrell, 1998]
- Shape
  - Deformable curve [Kass et al. 1988]
  - Template [Blake et al., 1993; Birchfield 1998]
  - Example-based [Comaniciu et al., 1993; Blumberg & Hogg, 1994]
- Appearance
  - Correlation [Lucas & Kanade, 1981; Shi & Tomasi, 1994]
  - Photometric variation [Hager & Belhumeur, 1998]
  - Outliers [Black et al., 1998; Hager & Belhumeur, 1998]
  - Nontextuality [Black et al., 1998; Sclaroff & Indelman, 1996]

- Motion
  - Background model [Wein et al., 1993; Rossides & Sclaroff, 1999; Stauffer & Grimson, 1999]
  - Optical flow [Cutler & Turk]
  - Epipolar motion [Hartley & Zisserman, 1996; Irani & Anandan, 1998]
- Stereo
  - Blob correlation [Azuayjani & Pentland, 1996]
  - Disparity map [Kanade et al., 1996; Konolige, 1997; Darrell et al., 1998]

Color Blob tracking

- Color-based tracker gets lost on white knight: Same Color

Snakes: Active Contours

- Contour C: continuous curve on smooth surface in $\mathbb{R}^3$
- Snake S: projection of C to image

- Curve types
  - Edge between regions on surface with contrasting properties
  - Line that contrasts with surface properties on both side
  - Silhouette of surface against contrasting background

- General Algorithm:
  - Perform edge detection
  - Fit parametric or non-parametric curve to data

Snakes: Basic Approach

- Parameterize a closed contour
- $r(s) = qB(s)$ or $r(s) = U(s)Q$

- Given a predicted state $q$, search radially for edges

- Solve a least squares problem for new state
Tracker Composition: Only Shape (Snakes)

• Geometry-based tracker gets lost on black pawn: Same shape

Tracker Composition: Color and Shape

• Combining Trackers => Robustness
• Trackers in video, IR and range

Visual Tracking using regions

\[ l_0 \xrightarrow{p_t} l_1 \]

Variability model: \[ l_1 = g(l_0, p_t) \]

Incremental Estimation: From \( l_0 \), \( l_{t+1} \) and \( p_t \) compute \( \Delta p_{t+1} \)

\[ \| l_0 - g(l_{t+1}, p_{t+1}) \| ^2 \Rightarrow \text{min} \]

Image Warping

• Warping is a change of coordinates: \( J(u,v) = l(I(u,v,p), g(u,v,p)) \)

• Always prefer to warp to destination to avoid gaps

• Two interpolation schemes
  - nearest neighbor
  - bilinear

• For much of tracking, nearest neighbor works well

Tracking using Textured Regions

• Mean intensity difference between \( I \) and affine warp of template image [Shi & Tomasi, 1994]

\[ \psi_{\text{mean}}(x,y) = \sum_{s,t} (I_s(x,y) - I_c(x,y))^2 \]

\[ \| I_s - I_c \| \]
**Template tracking: Planar Case**

Planar Object \( \Rightarrow \) Affine motion model: \( u'_i = Au_i + d \)

\[ l_i = g(p_i, \lambda_i) \]

**Hager/Toyama: Tracking Cycle**

- **Prediction**
  - Prior states predict new appearance

- **Image warping**
  - Generate a “normalized view”

- **Model inverse**
  - Compute error from nominal

- **State integration**
  - Apply correction to state

**SSD Tracking**

**XVision: A tracking System**

Composition of Primitive Trackers