Tracking
Definition of Tracking

• Tracking:
  – Generate some conclusions about the motion of the scene, objects, or the camera, given a sequence of images.
  – Knowing this motion, predict where things are going to project in the next image, so that we don’t have so much work looking for them.
Why Track?

- Detection and recognition are expensive.
- If we get an idea of where an object is in the image because we have an idea of the motion from previous images, we need less work detecting or recognizing the object.
Tracking a Silhouette by Measuring Edge Positions

- Observations are positions of edges along normals to tracked contour
Why not Wait and Process the Set of Images as a Batch?

- In a car system, detecting and tracking pedestrians in real time is important.
- Recursive methods require less computing
Implicit Assumptions of Tracking

- Physical cameras do not move instantly from a viewpoint to another
- Objects do not teleport between places around the scene
- Relative position between camera and scene changes incrementally
- We can model motion
Related Fields

- Signal Detection and Estimation
- Radar technology
The Problem: Signal Estimation

• We have a system with parameters
  – Scene structure, camera motion, automatic zoom
  – System state is unknown (“hidden”)

• We have measurements
  – Components of stable “feature points” in the images.
  – “Observations”, projections of the state.

• We want to recover the state components from the observations
Necessary Models

• We use models to describe a priori knowledge about
  • the world (including external parameters of camera)
  • the imaging projection process

System Dynamics Model

Measurement Model (projection)

(u, v)

(u, v)

Previous State

Next State

State

Measurement
A Simple Example of Estimation by Least Square Method

- Goal: Find estimate $\hat{a}$ of state $a$ such that the least square error between measurements and the state is minimum.

\[ C = \frac{1}{2} \sum_{i=1}^{n} (x_i - a)^2 \]

\[ \frac{\partial C}{\partial a} = 0 = \sum_{i=1}^{n} (x_i - \hat{a}) = \sum_{i=1}^{n} x_i - n \hat{a} \]

\[ \hat{a} = \frac{1}{n} \sum_{i=1}^{n} x_i \]
Recursive Least Square Estimation

• We don’t want to wait until all data have been collected to get an estimate $\hat{a}$ of the depth
• We don’t want to reprocess old data when we make a new measurement
• Recursive method: data at step $i$ are obtained from data at step $i-1$
Recursive Least Square Estimation 2

- Recursive method: data at step \( i \) are obtained from data at step \( i - 1 \)

\[
\hat{a}_i = -\frac{1}{i} \sum_{k=1}^{i} x_k = -\frac{1}{i} \sum_{k=1}^{i-1} x_k + \frac{1}{i} x_i
\]

\[
\hat{a}_i = \frac{i-1}{i} \hat{a}_{i-1} + \frac{1}{i} x_i
\]

\[
\hat{a}_i = \hat{a}_{i-1} + \frac{1}{i} (x_i - \hat{a}_{i-1})
\]
Recursive Least Square Estimation 3

\[ \hat{a}_i = \hat{a}_{i-1} + \frac{1}{i} (x_i - \hat{a}_{i-1}) \]

- **Gain**: Specifies how much do we pay attention to the difference between what we expected and what we actually get.
- **Actual measure**
- **Predicted measure**
- **Innovation**

Estimate at step \( i \)
Least Square Estimation of the State Vector of a Static System

1. Batch method

\[ x_i = H_i \cdot a \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_n \end{bmatrix} \]

⇒ \( x = H \hat{a} \)

Find estimate \( \hat{a} \) that minimizes

\[
C = \frac{1}{2} (X - H \hat{a})^T (X - H \hat{a})
\]

We find \( \hat{a} = (H^T H)^{-1} H^T X \)
Least Square Estimation of the State Vector of a Static System 2

2. Recursive method

Calculation is similar to calculation of running average
We had:
\[ \hat{a}_i = \hat{a}_{i-1} + \frac{1}{i} (x_i - \hat{a}_{i-1}) \]

Now we find:
\[ \hat{a}_i = \hat{a}_{i-1} + K_i (X_i - H_i \hat{a}_{i-1}) \]

with
\[ K_i = P_i H_i^T \]
\[ P_i = (H_i H_i^T)^{-1} \]
Dynamic System

\[
\begin{align*}
X_i &= X_{i-1} + V_{i-1} \Delta t \\
V_i &= V_{i-1} + A_{i-1} \Delta t \\
A_i &= A_{i-1} + w
\end{align*}
\]

\[
\begin{bmatrix}
X_i \\
V_i \\
A_i
\end{bmatrix} = 
\begin{bmatrix}
1 & \Delta t & 0 \\
0 & 1 & \Delta t \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X_{i-1} \\
V_{i-1} \\
A_{i-1}
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
w
\end{bmatrix}
\]

\[
\Rightarrow a_i = \Phi a_{i-1} + w
\]

\[
x_i =
\begin{bmatrix}
1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
X_i \\
V_i \\
A_i
\end{bmatrix} + V
\]

\[
\Rightarrow x_i = H a_i + V
\]
Recursive Least Square Estimation for a Dynamic System (Kalman Filter)

**State equation**
\[ a_i = \Phi_i \, a_{i-1} + w_{i-1} \quad \quad w_i \sim N(0, Q_i) \]

**Measurement equation**
\[ x_i = H_i \, a_i + n_i \quad \quad n_i \sim N(0, R_i) \]

**Prediction for \( a_i \)**
\[ \hat{a}_i = \Phi_i \, \hat{a}_{i-1} + K_i \, (x_i - H_i \, \Phi_i \, \hat{a}_{i-1}) \]

**Gain**
\[ K_i = P'_i \, H_i^T \left( H_i \, P'_i \, H_i^T + R_i \right)^{-1} \]

**Prediction for \( x_i \)**
\[ P'_i = \Phi_i \, P_{i-1} \, \Phi_i^T + Q_i \]

**Prediction for \( x_i \)**
\[ P_{i-1} = (I - K_{i-1} \, H_{i-1}) \, P'_{i-1} \]

**Tweak factor for model**
**Covariance matrix for prediction error**
**Covariance for estimation error**
Estimation when System Model is Nonlinear
(Extended Kalman Filter)

State equation
\[
a_i = f(a_{i-1}) + w_{i-1}
\]

Measurement equation
\[
x_i = H_i a_i + v_i
\]

\[
\hat{a}_i = f(\hat{a}_{i-1}) + K_i (x_i - H_i f(\hat{a}_{i-1}))
\]

\[
K_i = P_i' H_i^T (H_i P_i' H_i + R_i)^{-1}
\]

\[
P_i' = \frac{\partial f}{\partial a_i} P_{i-1} \left( \frac{\partial f}{\partial a_i} \right)^T + Q_i
\]

\[
P_{i-1} = (I - K_{i-1} H_{i-1}) P_i'
\]

Differences compared to regular Kalman filter are circled in red

Jacobian Matrix
Tracking Steps

- Predict next state as $\Phi_i \hat{a}_{i-1}$ using previous step and dynamic model.
- Predict regions $N(H_i \Phi_i \hat{a}_{i-1}, P'_i)$ of next measurements using measurement model and uncertainties.
- Make new measurements $x_i$ in predicted regions.
- Compute best estimate of next state

$$\hat{a}_i = \Phi_i \hat{a}_{i-1} + K_i (x_i - H_i \Phi_i \hat{a}_{i-1})$$

“Correction” of predicted state
Recursive Least Square Estimation for a Dynamic System (Kalman Filter)
Tracking as a Probabilistic Inference Problem

- Find distributions for state vector $a_i$ and for measurement vector $x_i$. Then we are able to compute the expectations $\hat{a}_i$ and $\hat{x}_i$
- Simplifying assumptions (same as for HMM)

\[
P(a_i | a_1, a_2, \ldots, a_{i-1}) = P(a_i | a_{i-1})
\]
(Only immediate past matters)

\[
P(x_i, x_j, \ldots, | a_i) = P(x_i | a_i)P(x_j | a_i)\ldots P(x_k | a_i)
\]
(Conditional independence of measurements given a state)
Tracking as Inference

• Prediction

\[
P(a_i | x_1, \ldots, x_{i-1}) = \int P(a_i | a_{i-1})P(a_{i-1} | x_1, \ldots, x_{i-1})da_{i-1}
\]

• Correction

\[
P(a_i | x_1, \ldots, x_i) = \frac{P(x_i | a_i)P(a_i | x_1, \ldots, x_{i-1})}{\int P(x_i | a_i)P(a_i | x_1, \ldots, x_{i-1})da_i}
\]

• Produces same results as least square approach if distributions are Gaussians: Kalman filter

• See Forsyth and Ponce, Ch. 19
Kalman Filter for 1D Signals

State equation
\[ a_i = f \hat{a}_{i-1} + w_{i-1} \]

Tweak factor for model
\[ w_i \sim N(0,q) \]

Measurement equation
\[ x_i = h a_i + v_i \]

Measurement noise
\[ v_i \sim N(0,r) \]

Prediction for \( a_i \) (a priori estimate)
\[ \hat{a}_i = f \hat{a}_{i-1} + K_i (x_i - h f \hat{a}_{i-1}) \]

Gain
\[ K_i = p'_i h \left( h^2 p'_i + r \right)^{-1} \]

Standard deviation for prediction error
\[ p'_i = f^2 p_{i-1} + q \]

St.d. for estimation error
\[ p_{i-1} = (1 - K_{i-1} h) p'_{i-1} \]
Applications: Structure from Motion

- Measurement vector components:
  - Coordinates of corners, “salient points”

- State vector components:
  - Camera motion parameters
  - Scene structure

- Is there enough equations?
  - N corners, 2N measurements
  - N unknown state components from structure (distances from first center of projection to 3D points)
  - 6 unknown state components from motion (translation and rotation)
  - More measurements than unknowns for every frame if N>6 (2N > N + 6)

- Batch methods
- Recursive methods (Kalman filter)
Problems with Tracking

• Initial detection
  – If it is too slow we will never catch up
  – If it is fast, why not do detection at every frame?
    • Even if raw detection can be done in real time, tracking
      saves processing cycles compared to raw detection.
      The CPU has other things to do.

• Detection is needed again if you lose tracking

• Most vision tracking prototypes use initial
detection done by hand
  (see Forsyth and Ponce for discussion)
References

- O. Faugeras. Three-Dimensional Computer Vision. MIT Press. Ch. 8, “Tracking Tokens over Time”.