Motion Tracking

CS6240 Multimedia Analysis

Leow Wee Kheng

Department of Computer Science
School of Computing
National University of Singapore

Introduction

Video contains motion information which can be used for

- detecting the presence of moving objects
- tracking and analyzing the motion of the objects
- tracking and analyzing the motion of camera

Basic tracking methods:

- Gradient-based Image Flow:
  - Track points based on intensity gradient.
  - Example: Lucas-Kanade method [LK81, TK91].
- Feature-based Image Flow:
  - Track points based on template matching of features at points.
- Mean Shift Tracking:
  - Track image patches based on feature distributions, e.g., color histograms [CRM00].

Strengths and Weaknesses

- Image flow approach:
  - Very general and easy to use.
  - If track correctly, can obtain precise trajectory with sub-pixel accuracy.
  - Easily confused by points with similar features.
  - Cannot handle occlusion.
  - Cannot differentiate between planner motion and motion in depth.
  - Demo: lk-elephant.mpg.

- Mean shift tracking:
  - Very general and easy to use.
  - Can track objects that change size & orientation.
  - Can handle occlusion, size change.
  - Track trajectory not as precise.
  - Can't track object boundaries accurately.
  - Demo: ms-football1.avi, ms-football2.avi.

Basic methods can be easily confused in complex situations:

frame 1

frame 2

- In frame 1, which hand is going which way?
- Which hand in frame 1 corresponds to which hand in frame 2?

Notes:

- The chances of making wrong association is reduced if we can correctly predict where the objects will be in frame 2.
- To predict ahead of time, need to estimate the velocities and the positions of the objects in frame 1.

To overcome these problems, need more sophisticated tracking algorithms:

- Kalman filtering: for linear dynamic systems, unimodal probability distributions
- Extended Kalman filtering: for nonlinear dynamic systems, unimodal probability distributions
- Condensation algorithm: for multi-modal probability distributions

g-h Filter

Consider 1-D case and suppose object travels at constant speed.

Let x_n and $\dot{x}_n$ denote position and speed of object at time step n. Then, at time step $n+1$, we have

$$\begin{aligned} x_{n+1} &= x_n + \dot{x}_n T &(1) \\ \dot{x}_{n+1} &= \dot{x}_n &(2) \end{aligned}$$

where T is the time interval between time steps.

These equations are called the system dynamic model.

Suppose at time step n, measured position $y_n \neq$ estimated position x_n. Then, update speed $\dot{x}_n$ of object as follows:

$$\dot{x}_n \leftarrow \dot{x}_n + h_n \frac{y_n - x_n}{T} \tag{3}$$

where h_n is a small parameter.

Notes:

- If $x_n < y_n$, then estimated speed < actual speed.
  Algorithm 3 increases the estimated speed.
- If $x_n > y_n$, then estimated speed > actual speed.
  Algorithm 3 decreases the estimated speed.
- After updating for several times, the estimated speed will become closer and closer to the actual speed.

Another way of writing Algorithm 3 is as the following equation:

$$\dot{x}^*_{n,n} = \dot{x}^*_{n,n-1} + h_n \frac{y_n - x^*_{n,n-1}}{T} \,. \tag{4}$$

- $\dot{x}^*_{n,n-1}$ = predicted estimate: the estimation of $\dot{x}$ at time step n based on past measurements made up to time step $n-1$.
- $\dot{x}^*_{n,n}$ = filtered estimate: the estimation of $\dot{x}$ at time step n based on past measurements made up to time step n.

Some books use this notation:

$$\dot{x}^*_{n|n} = \dot{x}^*_{n|n-1} + h_n \frac{y_n - x^*_{n|n-1}}{T} \,. \tag{5}$$

The estimated position can be updated in a similar way:

$$x^*_{n,n} = x^*_{n,n-1} + g_n(y_n - x^*_{n,n-1}) \tag{6}$$

where g_n is a small parameter.

Taken together, the two estimation equations form the g-h track update or filtering equations [Bro98]:

$$\dot{x}^*_{n,n} = \dot{x}^*_{n,n-1} + \frac{h_n}{T}(y_n - x^*_{n,n-1}) \tag{7}$$

$$x^*_{n,n} = x^*_{n,n-1} + g_n(y_n - x^*_{n,n-1})\,. \tag{8}$$

Now, we can use the system dynamic equations to predict the object's position and speed at time step $n+1$.

First, we rewrite the equations using the new notation to obtain the g-h state transition or prediction equations:

$$\begin{aligned} \dot{x}^*_{n+1,n} &= \dot{x}^*_{n,n} && (9) \\ x^*_{n+1,n} &= x^*_{n,n} + \dot{x}^*_{n,n}\,T && (10) \\ &= x^*_{n,n} + \dot{x}^*_{n+1,n}\,T\,. && (11) \end{aligned}$$

Substituting these equations into the filtering equations yield the g-h tracking-filter equations:

$$\begin{aligned} \dot{x}^*_{n+1,n} &= \dot{x}^*_{n,n-1} + \frac{h_n}{T}(y_n - x^*_{n,n-1}) && (12) \\ x^*_{n+1,n} &= x^*_{n,n-1} + T\,\dot{x}^*_{n+1,n} + g_n(y_n - x^*_{n,n-1})\,. && (13) \end{aligned}$$

These equations also describe many other filters, e.g.,

- Wiener filter
- Kalman filter
- Bayes filter
- Least-squares filter
- etc...

They differ in their choices of g_n and h_n.

g-h-k Filter

Consider the case in which the object travels with constant acceleration.

The equation of motion becomes:

$$
\begin{aligned}
x_{n+1} &= x_n + \dot{x}_n T + \ddot{x}_n \frac{T^2}{2} && (14)\\
\dot{x}_{n+1} &= \dot{x}_n + \ddot{x}_n T && (15)\\
\ddot{x}_{n+1} &= \ddot{x}_n \,. && (16)
\end{aligned}
$$

Following the same procedure used to develop the g-h filtering and prediction equations, we can develop the g-h-k filtering equations

$$\ddot{x}^*_{n,n} = \ddot{x}^*_{n,n-1} + \frac{2k_n}{T^2}(y_n - x^*_{n,n-1}) \tag{17}$$

$$\dot{x}^*_{n,n} = \dot{x}^*_{n,n-1} + \frac{h_n}{T}(y_n - x^*_{n,n-1}) \tag{18}$$

$$x^*_{n,n} = x^*_{n,n-1} + g_n(y_n - x^*_{n,n-1}) \tag{19}$$

and g-h-k state transition equations

$$\ddot{x}^*_{n+1,n} = \ddot{x}^*_{n,n} \tag{20}$$

$$\dot{x}^*_{n+1,n} = \dot{x}^*_{n,n} + \ddot{x}^*_{n,n}\, T \tag{21}$$

$$x^*_{n+1,n} = x^*_{n,n} + \dot{x}^*_{n,n}\, T + \ddot{x}^*_{n,n}\, \frac{T^2}{2}\,. \tag{22}$$

(Exercise)

1-D 2-State Kalman Filter

The system dynamic equations that we have considered previously

$$x_{n+1} = x_n + \dot{x}_n T \tag{23}$$

$$\dot{x}_{n+1} = \dot{x}_n \tag{24}$$

are deterministic description of object motion.

In the real world, the object will not have a constant speed for all time. There is uncertainty in the object's speed.

To model this, we add a random noise u_n to the object's speed. This gives rise to the following stochastic model [Bro98]:

$$x_{n+1} = x_n + \dot{x}_n T \tag{25}$$

$$\dot{x}_{n+1} = \dot{x}_n + u_n \,. \tag{26}$$

The equation that links the actual data x_n and the observed (or measured) data y_n is called the observation equation:

$$y_n = x_n + \nu_n \tag{27}$$

while ν_n is the observation or measurement noise.

The error $e_{n+1,n}$ of estimating x_{n+1} is

$$e_{n+1,n} = x_{n+1} - x^*_{n+1,n} \,. \tag{28}$$

Kalman looked for an optimum estimate that minimizes the mean squared error. After much effort, Kalman found that the optimum filter is given by the equations:

$$\dot{x}^*_{n+1,n} = \dot{x}^*_{n,n-1} + \frac{h_n}{T}(y_n - x^*_{n,n-1}) \tag{29}$$

$$x^*_{n+1,n} = x^*_{n,n-1} + T\,\dot{x}^*_{n+1,n} + g_n(y_n - x^*_{n,n-1}) \tag{30}$$

which are the same as for the g-h filter.

For the Kalman filter, g_n and h_n are

- dependent on n
- functions of the variance of the object position and speed
- functions of the accuracy of prior knowledge about the object's position and speed

In the steady state, g_n and h_n are constants g and h given by

$$h = \frac{g^2}{2 - g} \,. \tag{31}$$

Kalman Filter in Matrix Notation

The system dynamic equation in matrix form is [Bro98]:

$$\mathbf{X}_{n+1} = \mathbf{\Phi}\,\mathbf{X}_n + \mathbf{U}_n\,. \tag{32}$$

- $\mathbf{X}_n$ = state vector
- $\mathbf{\Phi}$ = state transition matrix
- $\mathbf{U}_n$ = system noise vector

The observation equation in matrix form is

$$\mathbf{Y}_n = \mathbf{M}\,\mathbf{X}_n + \mathbf{V}_n\,. \tag{33}$$

- $\mathbf{Y}_n$ = measurement vector
- $\mathbf{M}$ = observation matrix
- $\mathbf{V}_n$ = observation noise vector

The state transition or prediction equation becomes

$$\mathbf{X}^*_{n+1,n} = \mathbf{\Phi}\,\mathbf{X}^*_{n,n} \tag{34}$$

The track update or filtering equation becomes

$$\mathbf{X}^*_{n,n} = \mathbf{X}^*_{n,n-1} + \mathbf{K}_n(\mathbf{Y}_n - \mathbf{M}\,\mathbf{X}^*_{n,n-1})\,. \tag{35}$$

The matrix $\mathbf{K}_n$ is called the Kalman gain.

The state transition equation and track update equation are used in the tracking process.

Example: For the stochastic model, the system dynamic equations are:

$$
\begin{aligned}
x_{n+1} &= x_n + \dot{x}_n T && (36) \\
\dot{x}_{n+1} &= \dot{x}_n + u_n && (37)
\end{aligned}
$$

and the observation equation is:

$$
y_n = x_n + \nu_n \,. \tag{38}
$$

These equations give rise to the following matrices:

$$
\mathbf{X}_n = \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix}, \quad \mathbf{\Phi} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad \mathbf{U}_n = \begin{bmatrix} 0 \\ u_n \end{bmatrix}. \tag{39}
$$

$$
\mathbf{Y}_n = \begin{bmatrix} y_n \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \mathbf{V}_n = \begin{bmatrix} \nu_n \end{bmatrix}. \tag{40}
$$

To apply Kalman filtering, we have

$$\mathbf{X}^*_{n,n} = \begin{bmatrix} x^*_{n,n} \\ \dot{x}^*_{n,n} \end{bmatrix}, \quad \mathbf{X}^*_{n+1,n} = \begin{bmatrix} x^*_{n+1,n} \\ \dot{x}^*_{n+1,n} \end{bmatrix}. \tag{41}$$

and

$$\mathbf{K}_n = \begin{bmatrix} g_n \\ \dfrac{h_n}{T} \end{bmatrix}. \tag{42}$$

The previous form of Kalman gain does not tell us how to compute g_n and h_n.

The following general form does (derivation omitted):

$$\mathbf{K}_n = \mathbf{S}^*_{n,n-1}\,\mathbf{M}^T\left[\mathbf{R}_n + \mathbf{M}\,\mathbf{S}^*_{n,n-1}\,\mathbf{M}^T\right]^{-1} \tag{43}$$

where

$$\begin{aligned}
\mathbf{S}^*_{n,n-1} &= \mathrm{COV}(\mathbf{X}^*_{n,n-1}) = E\{\mathbf{X}^*_{n,n-1}\,\mathbf{X}^{*T}_{n,n-1}\} && (44)\\
&= \mathbf{\Phi}\,\mathbf{S}^*_{n-1,n-1}\,\mathbf{\Phi}^T + \mathbf{Q}_n && (45)\\
\mathbf{S}^*_{n-1,n-1} &= \mathrm{COV}(\mathbf{X}^*_{n-1,n-1}) && (46)\\
&= [\mathbf{I} - \mathbf{K}_{n-1}\,\mathbf{M}]\,\mathbf{S}^*_{n-1,n-2} && (47)\\
\mathbf{Q}_n &= \mathrm{COV}(\mathbf{U}_n) && (48)\\
\mathbf{R}_n &= \mathrm{COV}(\mathbf{V}_n) && (49)
\end{aligned}$$

To use Kalman filter:

1. Write down system dynamic equation and observation equation.
2. Derive track update equation and state transition equation.
3. Given $\mathbf{\Phi}$, $\mathbf{M}$, $\mathbf{R}_n$, $\mathbf{Q}_n$, $n = 0, 1, \ldots$, $\mathbf{X}^*_{0,-1}$, and $\mathbf{S}^*_{0,-1}$.
4. Repeat for $n = 0, 1, \ldots$
    1. Compute Kalman gain:
    $$\mathbf{K}_n = \mathbf{S}^*_{n,n-1}\,\mathbf{M}^T \left[\mathbf{R}_n + \mathbf{M}\,\mathbf{S}^*_{n,n-1}\,\mathbf{M}^T\right]^{-1}$$
    2. Measure $\mathbf{Y}_n$ and update estimate using update equation:
    $$\mathbf{X}^*_{n,n} = \mathbf{X}^*_{n,n-1} + \mathbf{K}_n(\mathbf{Y}_n - \mathbf{M}\,\mathbf{X}^*_{n,n-1})\,.$$
    3. Compute covariance of smoothed estimate:
    $$\mathbf{S}^*_{n,n} = [\mathbf{I} - \mathbf{K}_n\,\mathbf{M}]\,\mathbf{S}^*_{n,n-1}$$
    4. Predict using state transition equation:
    $$\mathbf{X}^*_{n+1,n} = \mathbf{\Phi}\,\mathbf{X}^*_{n,n}$$
    5. Compute predictor covariance:
    $$\mathbf{S}^*_{n+1,n} = \mathbf{\Phi}\,\mathbf{S}^*_{n,n}\,\mathbf{\Phi}^T + \mathbf{Q}_{n+1}$$

Notes:

- $\mathbf{U}_n$ and $\mathbf{V}_n$ are assumed to be uncorrelated zero-mean Gaussian noise, i.e.,

$$\mathrm{COV}(\mathbf{U}_n) = E\{\mathbf{U}_n\,\mathbf{U}_k^T\} = \begin{cases} \mathbf{Q}_n & \text{if } n = k \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$\mathrm{COV}(\mathbf{V}_n) = E\{\mathbf{V}_n\,\mathbf{V}_k^T\} = \begin{cases} \mathbf{R}_n & \text{if } n = k \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$$E\{\mathbf{U}_n\,\mathbf{V}_k^T\} = 0 \quad \text{for all } n, k$$

- In [Bro98], $\mathbf{S}^*_{0,-1}$ is given as

$$\mathbf{S}^*_{0,-1} = \mathrm{COV}(\mathbf{X}^*_{0,-1})$$

and Step 4(e) is given as

$$\mathbf{S}^*_{n+1,n} = \mathbf{\Phi}\,\mathbf{S}^*_{n,n}\,\mathbf{\Phi}^T + \mathbf{Q}_{n+1}\,.$$

- In other books, e.g., [BH97], $\mathbf{S}^*_{0,-1}$ is given as

$$\mathbf{S}^*_{0,-1} = \mathrm{COV}(\mathbf{X}_0 - \mathbf{X}^*_{0,-1})$$

and Step 4(e) is given as

$$\mathbf{S}^*_{n+1,n} = \mathbf{\Phi}\, \mathbf{S}^*_{n,n}\, \mathbf{\Phi}^T + \mathbf{Q}_n\,.$$

- In general, $\mathbf{\Phi}$ and $\mathbf{M}$ may change over time, i.e., $\mathbf{\Phi}_n$, $\mathbf{M}_n$.
- The matrix form can be easily applied to multi-dimensional, multi-variate cases. For example, for 3-D space, we have

$$\begin{aligned}\mathbf{X}^*_{n,n-1} = \big[\; & x^*_{n,n-1} \quad \dot{x}^*_{n,n-1} \quad \ddot{x}^*_{n,n-1} \\ & y^*_{n,n-1} \quad \dot{y}^*_{n,n-1} \quad \ddot{y}^*_{n,n-1} \quad z^*_{n,n-1} \quad \dot{z}^*_{n,n-1} \quad \ddot{z}^*_{n,n-1} \;\big]^T \end{aligned} \tag{50}$$

Example

Use Kalman filter to track "random walk" [BH97].

The actual random walk is generated by the equation:

$$\dot{x} = u(t) \tag{51}$$

where $u(t)$ is a Gaussian white noise with variance $= 1$.

Measurements are sampled at time $t = 0, 1, 2, \ldots$

$$y_n = x_n + \nu_n \tag{52}$$

where ν_n is Gaussian white noise with variance $= 1$.

For the correct Kalman filter model, we choose the model

$$\mathbf{X}_{n+1} = [\ x_{n+1}\ ] \quad = \quad [\ x_n\ ] + [\ u_n\ ]$$

$$\mathbf{Y}_n = [\ y_n\ ] \quad = \quad [\ x_n\ ] + [\ \nu_n\ ]$$

That is, $\mathbf{\Phi} = [1]$ and $\mathbf{M} = [1]$.
Also choose $\mathbf{Q}_n = [1]$, $\mathbf{R}_n = [0.1]$, $\mathbf{X}^*_{0,-1} = [0]$, $\mathbf{S}^*_{0,-1} = [1]$.

For comparison, consider an incorrect Kalman filter model:

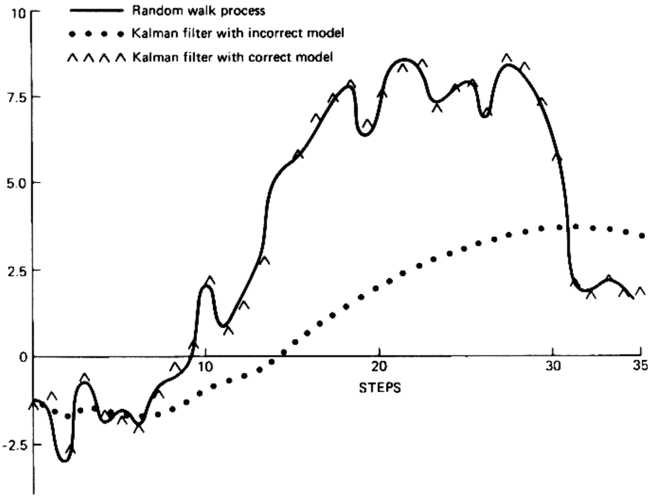
$$\mathbf{X}_{n+1} = [\ x_{n+1}\ ] \quad = \quad [\ x_n\ ]$$

$$\mathbf{Y}_n = [\ y_n\ ] \quad = \quad [\ x_n\ ] + [\ \nu_n\ ]$$

That is, $\mathbf{\Phi} = [1]$, $\mathbf{M} = [1]$, and $\mathbf{Q}_n = [0]$.

The other parameter values are the same as for the correct model:
$\mathbf{R}_n = [0.1]$, $\mathbf{X}^*_{0,-1} = [0]$, $\mathbf{S}^*_{0,-1} = [1]$.

Results:



Summary:

- Error in the model results in the tracking error.

Divergence Problems

- We want Kalman filter to converge to the correct track.
- But under certain conditions, divergence problems can arise.

Possible causes of divergence (see [BH97], Sect. 6.6 for details):

- Roundoff Errors:
  May become larger and larger as the number of steps increases.
  Prevention:
    - Use high-precision arithmetic.
    - Avoid deterministic model, i.e., include random noise variable.
    - Keep the $\mathbf{S}^*$ matrix symmetric because covariance matrix is symmetric.
    - Use the Joseph form to update $\mathbf{S}^*$:

$$\mathbf{S}^*_{n,n} = [\mathbf{I} - \mathbf{K}_n\,\mathbf{M}]\,\mathbf{S}^*_{n,n-1}\,[\mathbf{I} - \mathbf{K}_n\,\mathbf{M}]^T + \mathbf{K}_n\,\mathbf{R}_n\,\mathbf{K}_n^T \tag{53}$$

- Modeling Errors:
  - Errors in the system dynamic equation. We've seen the result of modeling errors in the "random walk" example.
  - In general, $\mathbf{\Phi}_n$ and $\mathbf{M}_n$ may change over time, i.e., vary for different n.
- Observability Problem:
  Some state variables may be hidden and not observable. If unobserved processes are unstable, then the estimation error will be unstable.

Data Association

During tracking, how to look for the next possible locations of the tracked objects?

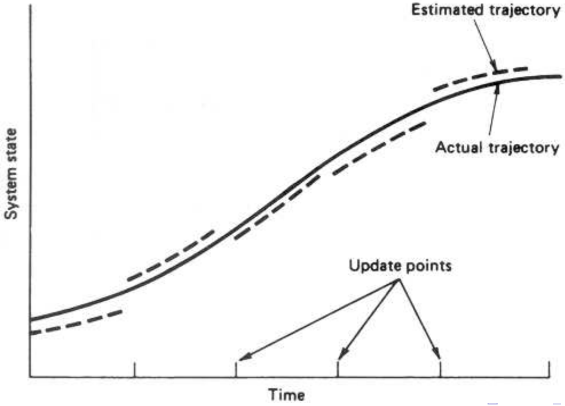
Possible approaches [Bro98]:

1. Nearest-Neighbor:
   Look for the nearest-neighboring object within a prediction window.
2. Branching or Track Splitting [Bla86]:
   Defer the decision for one or more time steps.
3. Probability Hypothesis Testing [Bla86]:
   Probabilistic data association, joint probabilistic data association.
4. Match features of the tracked objects.
5. Apply known constraints or knowledge about the tracked objects.

Extended Kalman Filter

Used when dynamics/measurement relationships are nonlinear [BH97].
Basic Idea:

- Approximate actual trajectory by piece-wise linear trajectories.
- Apply Kalman filter on estimated trajectories.



Assume the dynamic and measurement equations can be written as

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, t) + \mathbf{U}(t) \tag{54}$$

$$\mathbf{Y} = \mathbf{h}(\mathbf{X}, t) + \mathbf{V}(t) \tag{55}$$

where $\mathbf{f}$ and $\mathbf{h}$ are known functions.

For Extended Kalman filter, the filter loop at Step 4 is similar except:

- In Step 4(b), the filtering equation is

$$\mathbf{X}^*_{n,n} = \mathbf{X}^*_{n,n-1} + \mathbf{K}_n(\mathbf{Y}_n - \mathbf{Y}^*_{n,n-1}) \,. \tag{56}$$

- In Step 4(d), compute $\mathbf{X}^*_{n+1,n}$ as the solution of Eqn. 54 at $t = t_{n+1}$, subject to the initial condition $\mathbf{X} = \mathbf{X}^*_{n,n}$ at t_n.

Once $\mathbf{X}^*_{n+1,n}$ is computed, can compute $\mathbf{Y}^*_{n+1,n}$ as

$$\mathbf{Y}^*_{n+1,n} = \mathbf{h}(\mathbf{X}^*_{n+1,n}\,, t) \,. \tag{57}$$

Then, the filter loop is repeated.

CONDENSATION

Conditional Density Propagation over time [IB96, IB98].
Also called particle filtering.

Main differences with Kalman filter:

1. Kalman filter:
    - Assumes uni-modal (Gaussian) distribution.
    - Predicts single new state for each object tracked.
    - Updates state based on error between predicted state and observed data.

2. CONDENSATION algorithm:
    - Can work for multi-modal distribution.
    - Predicts multiple possible states for each object tracked.
    - Each possible state has a different probability.
    - Estimates probabilities of predicted states based on observed data.

Probability Density Functions

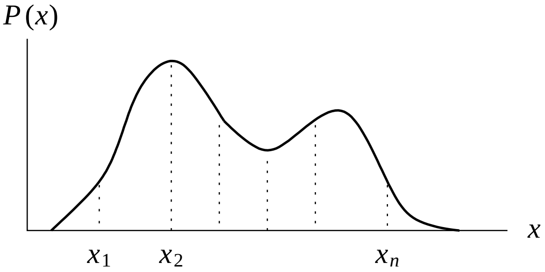
Two basic representations of probability density functions $P(x)$:

1. Explicit
   - Represent $P(x)$ by an explicit formula, e.g., Gaussian

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{58}$$

   - Given any x, can compute $P(x)$ using the formula.

2. Implicit
   - Represent $P(x)$ by a set of samples $x_1, x_2, \ldots, x_n$ and their estimated probabilities $P(x_i)$.
   - Given any $x' \neq x_i$, cannot compute $P(x')$ because there is no explicit formula.

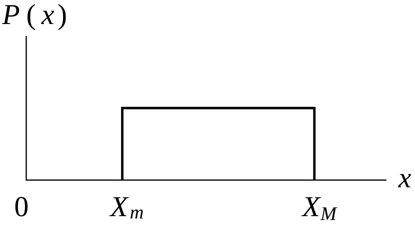


CONDENSATION algorithm predicts multiple possible next states.

- Achieved using sampling or drawing samples from the probability density functions.
- High probability samples should be drawn more frequently.
- Low probability samples should be drawn less frequently.

Sampling from Uniform Distribution

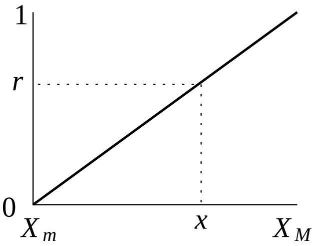
Uniform Distribution:



- Equal probability between X_m and X_M:

$$P(x) = \begin{cases} \dfrac{1}{X_M - X_m} & \text{if } X_m \leq x \leq X_M \\ 0 & \text{otherwise.} \end{cases} \tag{59}$$

Sampling Algorithm:



1. Generate a random number r from $[0, 1]$ (uniform distribution).
2. Map r to x:

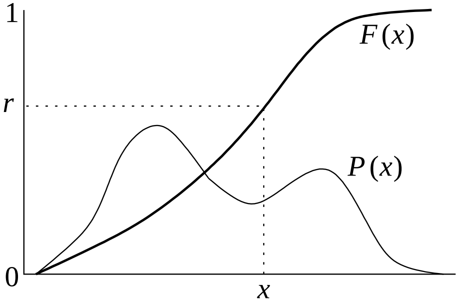
$$x = X_m + r(X_M - X_m) \tag{60}$$

The samples x drawn will have uniform distribution.

Sampling from Non-uniform Distribution

Let $P(x)$ denote the probability density function.
$F(x)$ is the indefinite integral of $P(x)$:

$$F(x) = \int_0^x P(x)dx \tag{61}$$



Sampling Algorithm:

1. Generate a random number r from $[0, 1]$ (uniform distribution).
2. Map r to x:
    - Find the x such that $F(x) = r$, i.e., $x = F^{-1}(r)$.
    - That is, find the x such that the area under $P(x)$ to the left of x equals r.

The samples x drawn will fit the probability distribution.

Sampling from Implicit Distribution

The method is useful when

- it is difficult to compute $F^{-1}(r)$, or
- the probability density is implicit.

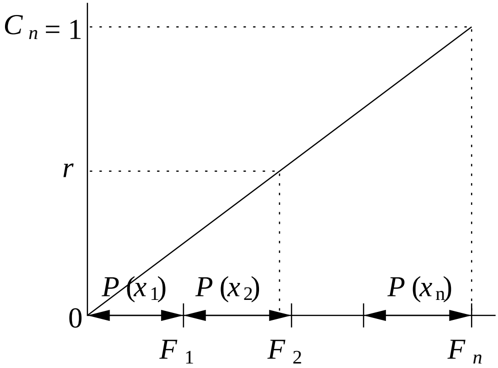
The basic idea is similar to the previous method:

- Given x_i and $P(x_i)$, $i = 1, \ldots, n$.
- Compute cumulative probability $F(x_i)$:

$$F(x_i) = \sum_{j=1}^{i} P(x_j) \tag{62}$$

- Compute normalized weight $C(x_i)$:

$$C(x_i) = F(x_i)/F(x_n) \tag{63}$$



Sampling Algorithm:

1. Generate a random number r from $[0, 1]$ (uniform distribution).
2. Map r to x_i:
   - Find the smallest i such that $C_i \geq r$.
   - Return x_i.

Samples $x = x_i$ drawn will follow probability density.

- The larger the n, the better the approximation.

Factored Sampling

- x: object model (e.g., a curve)
- z: observed or measured data in image
- $P(x)$: a priori (or prior) probability density of x occurring.
- $P(z|x)$: likelihood that object x gives rise to data z.
- $P(x|z)$: a posteriori (or posterior) probability density that the object is actually x given that z is observed in the image.
  So, want to estimate $P(x|z)$.

From Bayes' rule:

$$P(x|z) = k\, P(z|x)\, P(x) \tag{64}$$

where $k = P(z)$ is a normalizing term that does not depend on x.

Notes:

- In general, $P(z|x)$ is multi-modal.
- Cannot compute $P(x|z)$ using closed form equation.
  Has to use iterative sampling technique.
- Basic method: factored sampling [GCK91]. Useful when
  - $P(z|x)$ can be evaluated point-wise but sampling it is not feasible, and
  - $P(x)$ can be sampled but not evaluated.

Factored Sampling Algorithm [GCK91]:

1. Generate a set of samples $\{s_1, s_2, \ldots, s_n\}$ from $P(x)$.
2. Choose an index $i \in \{1, \ldots, n\}$ with probability π_i:

$$\pi_i = \frac{P(z|x = s_i)}{\sum_{j=1}^{n} P(z|x = s_j)} . \tag{65}$$

3. Return x_i.

The samples $x = x_i$ drawn will have a distribution that approximates $P(x|z)$.

- The larger the n, the better the approximation.
- So, no need to explicitly compute $P(x|z)$.

CONDENSATION Algorithm

Object Dynamics

- state of object model at time t: $\mathbf{x}(t)$
- history of object model: $\mathbf{X}(t) = (\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(t))$
- set of image features at time t: $\mathbf{z}(t)$
- history of features: $\mathbf{Z}(t) = (\mathbf{z}(1), \mathbf{z}(2), \ldots, \mathbf{z}(t))$

General assumption: object dynamic is a Markov process:

$$P(\mathbf{x}(t+1) \,|\, \mathbf{X}(t)) = P(\mathbf{x}(t+1) \,|\, \mathbf{x}(t)) \tag{66}$$

i.e., new state depends only on immediately preceding state.

- $P(\mathbf{x}(t+1) \,|\, \mathbf{X}(t))$ governs probability of state change.

Measurements

Measurements $\mathbf{z}(t)$ are assumed to be mutually independent, and also independent of object dynamics. So,

$$P(\mathbf{Z}(t) \,|\, \mathbf{X}(t)) = \prod_{i=1}^{t} P(\mathbf{z}(i) \,|\, \mathbf{x}(i)) \,. \tag{67}$$

CONDENSATION Algorithm

Iterate:

At time t, construct n samples $\{\mathbf{s}_i(t), \pi_i(t), c_i(t), i = 1, \ldots, n\}$ as follows:

The ith sample is constructed as follows:

1. Select a sample $\mathbf{s}'_j(t)$ as follows:
    - generate a random number $r \in [0, 1]$, uniformly distributed
    - find the smallest j such that $c_j(t-1) \geq r$
2. Predict by sampling from

$$P(\mathbf{x}(t) \,|\, \mathbf{x}(t-1) = \mathbf{s}'_j(t-1))$$

to choose $\mathbf{s}_i(t)$.

3. Measure $\mathbf{z}(t)$ from image and weight new sample:

$$\pi_i(t) = P(\mathbf{z}(t) \,|\, \mathbf{x}(t) = \mathbf{s}_i(t))$$

- normalize $\pi_i(t)$ so that $\sum_i \pi_i(t) = 1$
- compute cumulative probability $c_i(t)$:

$$\begin{aligned} c_0(t) &= 0 \\ c_i(t) &= c_{i-1}(t) + \pi_i(t) \end{aligned}$$

Example

Track curves in input video [IB96].

Let

- $\mathbf{x}$ denote the parameters of a linear transformation of a B-spline curve, either affine deformation or some non-rigid motion,
- $\mathbf{p}_s$ denote points on the curve.

Notes:

- Instead of modeling the curve, model the transformation of curve.
- Curve can change shape drastically over time.
- But, changes of transformation parameters are smaller.

Model Dynamics

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\boldsymbol{\omega}(t) \tag{68}$$

- $\mathbf{A}$: state transition matrix
- $\boldsymbol{\omega}$: random noise
- $\mathbf{B}$: scaling matrix

Then, $P(\mathbf{x}(t+1) \,|\, \mathbf{x}(t))$ is given by

$$P(\mathbf{x}(t+1) \,|\, \mathbf{x}(t)) = \exp\left\{ -\frac{1}{2} \|\mathbf{B}^{-1}[\mathbf{x}(t+1) - \mathbf{A}\mathbf{x}(t)]\|^2 \right\} . \tag{69}$$

$P(\mathbf{x}(t+1) \,|\, \mathbf{x}(t))$ is a Gaussian.

Measurement

- $P(\mathbf{z}(t)\,|\,\mathbf{x}(t))$ is assumed to remain unchanged over time.
- $\mathbf{z}_s$ is nearest edge to point $\mathbf{p}_s$ on model curve, within a small neighborhood δ of $\mathbf{p}_s$.
- To allow for missing edge and noise, measurement density is modeled as a robust statistics, a truncated Gaussian:

$$P(\mathbf{z}|\mathbf{x}) = \exp\left\{-\frac{1}{2\sigma^2}\sum_s \phi_s\right\} \tag{70}$$

where

$$\phi_s = \begin{cases} \|\mathbf{p}_s - \mathbf{z}_s\|^2 & \text{if } \|\mathbf{p}_s - \mathbf{z}_s\| < \delta \\ \rho & \text{otherwise.} \end{cases} \tag{71}$$

ρ is a constant penalty.

Now, can apply CONDENSATION algorithm to track the curve.

Further Readings:

1. [BH97] Section 5.5: Kalman filter given in slightly different notations and slightly different estimate for $\mathbf{S}^*_{n+1,n}$.
2. [BH97] p. 346, 347: Extended Kalman filter.
3. [IB96, IB98]: Other application examples of CONDENSATION algorithm.

Exercises:

1. Derive the state transition equation and track update equations for *g*-*h*-*k* filter.
2. Derive the transition matrix $\mathbf{\Phi}$ for the dynamic system given by Equations 14, 15, 16.

Reference I

- R. G. Brown and P. Y. C. Hwang.
  Introduction to Random Signals and Applied Kalman Filtering.
  John Wiley & Sons, 3rd edition, 1997.

- S. S. Blackman.
  Multiple-Target Tracking with Radar Applications.
  Artech House, Norwood, M.A., 1986.

- E. Brookner.
  Tracking and Kalman Filtering Made Easy.
  John Wiley & Sons, 1998.

- D. Comaniciu, V. Ramesh, and P. Meer.
  Real-time tracking of non-rigid objects using mean shift.
  In *IEEE Proc. on Computer Vision and Pattern Recognition*, pages 673–678, 2000.

Reference II

U. Grenander, Y. Chow, and D. M. Keenan.
HANDS. A Pattern Theoretical Study of Biological Shapes.
Springer-Verlag, 1991.

M. Isard and A. Blake.
Contour tracking by stochastic propagation of conditional density.
In *Proc. European Conf. on Computer Vision*, volume 1, pages 343–356, 1996.

M. Isard and A. Blake.
CONDENSATION — conditional density propagation for visual tracking.
Int. J. Computer Vision, 29(1):5–28, 1998.

Reference III

- B. D. Lucas and T. Kanade.
  An iterative image registration technique with an application to stereo vision.
  In *Proceedings of 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.
  http://www.ri.cmu.edu/people/person_136_pubs.html.

- C. Tomasi and T. Kanade.
  Detection and tracking of point features.
  Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, 1991.
  http://citeseer.nj.nec.com/tomasi91detection.html,
  http://www.ri.cmu.edu/people/person_136_pubs.html.