

Computer Vision Week 11

Tracking

Tracking

- 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
- Examples
 - object is ball, state is 3D position+velocity, measurements are stereo pairs
 - object is person, state is body configuration, measurements are frames, clock is in camera (30 fps)

Problem formulation

- X : state; not directly observable
- Y : output (measurement); we observe this
- $p(X)$ Prior probability of state vector; summarizes prior domain knowledge by independent measurements
- $p(Y)$ Probability of measuring Y ; fixed for any given image
- $p(Y|X)$ Probability of measuring Y given that the state is X ; compares image to expectation based on state
- $p(X|Y)$ Probability of X given that measurement Y has occurred; called state posterior
- X_t, Y_t discrete model; measurements taken at discrete instants of time (clock ticks)

Motion and measurement model

- $X_{t+1} = A X_t + B W_t$
- $Y_{t+1} = C X_{t+1} + D N_{t+1}$

Three main steps

- **Prediction:** we have seen y_0, \dots, 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 $P(X_i | Y_0 = y_0, \dots, Y_{i-1} = y_{i-1})$.
- **Data association:** Some of the measurements obtained from the i -th frame may tell us about the object's state. Typically, we use $P(X_i | Y_0 = y_0, \dots, Y_{i-1} = y_{i-1})$ to identify these measurements.
- **Correction:** now that we have y_i — the relevant measurements — we need to compute a representation of $P(X_i | Y_0 = y_0, \dots, Y_i = y_i)$.

Simplifying Assumptions

- **Only the immediate past matters:** formally, we require

$$P(X_i | X_1, \dots, X_{i-1}) = P(X_i | X_{i-1})$$

This assumption hugely simplifies the design of algorithms, as we shall see; furthermore, it isn't terribly restrictive if we're clever about interpreting X_i as we shall show in the next section.

- **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

$$P(Y_i, Y_j, \dots, Y_k | X_i) = P(Y_i | X_i) P(Y_j, \dots, Y_k | X_i)$$

Again, this isn't a particularly restrictive or controversial assumption, but it yields important simplifications.

Tracking as induction

- Assume data association is done
 - we'll talk about this later; a dangerous assumption
- Do correction for the 0'th frame
- Assume we have corrected estimate for i'th frame
 - show we can do prediction for i+1, correction for i+1

Base case

Firstly, we assume that we have $P(\mathbf{X}_0)$.

$$\begin{aligned} P(\mathbf{X}_0 | \mathbf{Y}_0 = \mathbf{y}_0) &= \frac{P(\mathbf{y}_0 | \mathbf{X}_0) P(\mathbf{X}_0)}{P(\mathbf{y}_0)} \\ &= \frac{P(\mathbf{y}_0 | \mathbf{X}_0) P(\mathbf{X}_0)}{\int P(\mathbf{y}_0 | \mathbf{X}_0) P(\mathbf{X}_0) d\mathbf{X}_0} \\ &\propto P(\mathbf{y}_0 | \mathbf{X}_0) P(\mathbf{X}_0) \end{aligned}$$

Induction step

Given $P(\mathbf{X}_{i-1} | \mathbf{y}_0, \dots, \mathbf{y}_{i-1})$.

Prediction

Prediction involves representing

$$P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_{i-1})$$

Our independence assumptions make it possible to write

$$\begin{aligned} P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) &= \int P(\mathbf{X}_i, \mathbf{X}_{i-1} | \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) d\mathbf{X}_{i-1} \\ &= \int P(\mathbf{X}_i | \mathbf{X}_{i-1}, \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) P(\mathbf{X}_{i-1} | \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) d\mathbf{X}_{i-1} \\ &= \int P(\mathbf{X}_i | \mathbf{X}_{i-1}) P(\mathbf{X}_{i-1} | \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) d\mathbf{X}_{i-1} \end{aligned}$$

Induction step

Correction

Correction involves obtaining a representation of

$$P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_i)$$

Our independence assumptions make it possible to write

$$\begin{aligned} P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_i) &= \frac{P(\mathbf{X}_i, \mathbf{y}_0, \dots, \mathbf{y}_i)}{P(\mathbf{y}_0, \dots, \mathbf{y}_i)} \\ &= \frac{P(\mathbf{y}_i | \mathbf{X}_i, \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) P(\mathbf{y}_0, \dots, \mathbf{y}_{i-1})}{P(\mathbf{y}_0, \dots, \mathbf{y}_i)} \\ &= P(\mathbf{y}_i | \mathbf{X}_i) P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) \frac{P(\mathbf{y}_0, \dots, \mathbf{y}_{i-1})}{P(\mathbf{y}_0, \dots, \mathbf{y}_i)} \\ &= \frac{P(\mathbf{y}_i | \mathbf{X}_i) P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_{i-1})}{\int P(\mathbf{y}_i | \mathbf{X}_i) P(\mathbf{X}_i | \mathbf{y}_0, \dots, \mathbf{y}_{i-1}) d\mathbf{X}_i} \end{aligned}$$

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 .
- Then a linear dynamic model has the form

$$\mathbf{x}_i = N(\mathbf{D}_i \mathbf{x}_{i-1}; \Sigma_{d_i})$$

$$\mathbf{y}_i = N(\mathbf{M}_i \mathbf{x}_i; \Sigma_{m_i})$$

- This is much, much more general than it looks, and extremely powerful

Examples

- Drifting points
 - we assume that the new position of the point is the old one, plus noise.
 - For the measurement model, we may not need to observe the whole state of the object
 - e.g. a point moving in 3D, at the 3k'th tick we see x , 3k+1'th tick we see y , 3k+2'th tick we see z
 - in this case, we can still make decent estimates of **all three** coordinates at each tick.
 - This property, which does not apply to every model, is called **Observability**

Examples

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

Points moving with constant velocity

– We have

$$u_i = u_{i-1} + \Delta t v_{i-1} + \varepsilon_i$$

$$v_i = v_{i-1} + \zeta_i$$

- (the Greek letters denote noise terms)
- Stack (u, v) into a single state vector

$$\begin{pmatrix} u \\ v \end{pmatrix}_i = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_{i-1} + \text{noise}$$

- which is the form we had above

Points moving with constant acceleration

– We have

$$u_i = u_{i-1} + \Delta t v_{i-1} + \varepsilon_i$$

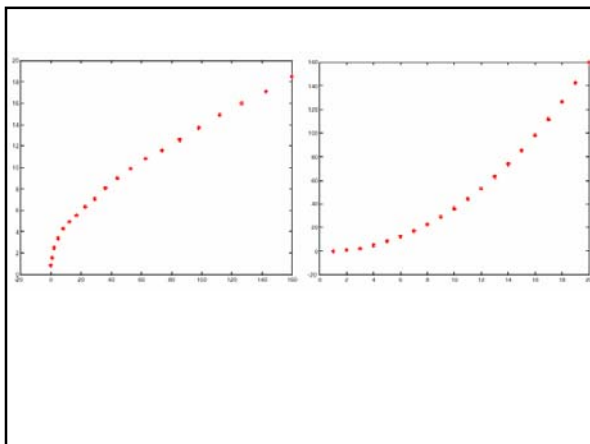
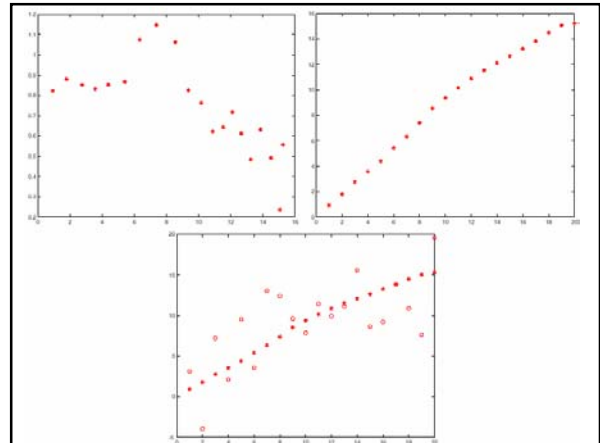
$$v_i = v_{i-1} + \Delta t a_{i-1} + \zeta_i$$

$$a_i = a_{i-1} + \xi_i$$

- (the Greek letters denote noise terms)
- Stack (u, v) into a single state vector

$$\begin{pmatrix} u \\ v \\ a \end{pmatrix}_i = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ a \end{pmatrix}_{i-1} + \text{noise}$$

- 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 --- once you know the mean and covariance, you're done

The Kalman Filter in 1D

- Dynamic Model

$$x_i \sim N(d_i x_{i-1}, \sigma_d^2)$$

$$y_i \sim N(m_i x_i, \sigma_m^2)$$

- Notation

mean of $P(X_i|y_0, \dots, y_{i-1})$ as \bar{x}_i^- — Predicted mean

Corrected mean — mean of $P(X_i|y_0, \dots, y_i)$ as \bar{x}_i^+

the standard deviation of $P(X_i|y_0, \dots, y_{i-1})$ as σ_i^-
of $P(X_i|y_0, \dots, y_i)$ as σ_i^+

Prediction for 1D 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 constant² 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 by square of constant, adding zero mean noise adds zero to the mean, adding rv's adds variance

Dynamic Model:

$$x_i \sim N(d_i x_{i-1}, \sigma_d)$$

$$y_i \sim N(m_i x_i, \sigma_m)$$

Start Assumptions: \bar{x}_0 and σ_0 are known

Update Equations: Prediction

$$\bar{x}_i^- = d_i \bar{x}_{i-1}^+$$

$$\sigma_i^- = \sqrt{\sigma_d^2 + (d_i \sigma_{i-1}^+)^2}$$

Update Equations: Correction

$$\bar{x}_i^+ = \left(\frac{\bar{x}_i^- \sigma_m^2 + m_i y_i (\sigma_i^-)^2}{\sigma_m^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_m^2 (\sigma_i^-)^2}{\sigma_m^2 + m_i^2 (\sigma_i^-)^2} \right)}$$

Correction for 1D Kalman filter

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

$$\bar{x}_i^+ = \left(\frac{\bar{x}_i^- \sigma_m^2 + m_i y_i (\sigma_i^-)^2}{\sigma_m^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_m^2 (\sigma_i^-)^2}{\sigma_m^2 + m_i^2 (\sigma_i^-)^2} \right)}$$

- Notice:

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

In higher dimensions, derivation follows the same lines, but isn't as easy.

Dynamic Model:

$$x_i \sim N(\mathcal{D}x_{i-1}, \Sigma_d)$$

$$y_i \sim N(\mathcal{M}_i x_i, \Sigma_m)$$

Start Assumptions: \bar{x}_0 and Σ_0 are known

Update Equations: Prediction

$$\bar{x}_i^- = \mathcal{D} \bar{x}_{i-1}^+$$

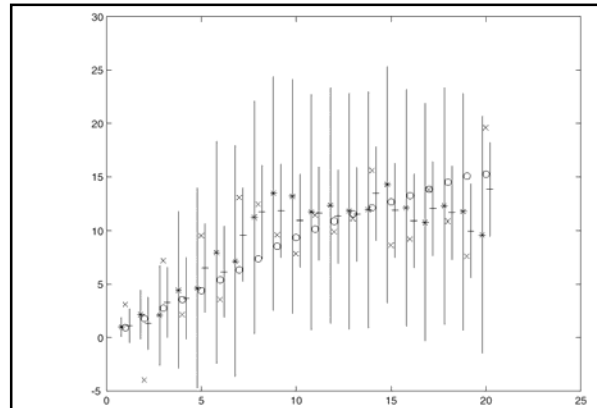
$$\Sigma_i^- = \Sigma_d + \mathcal{D} \Sigma_{i-1}^+ \mathcal{D}^T$$

Update Equations: Correction

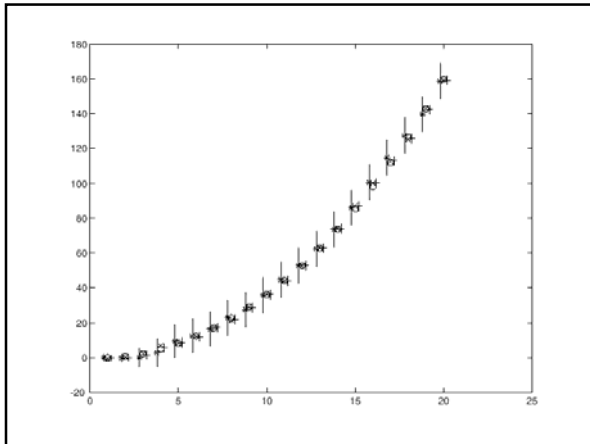
$$K_i = \Sigma_i^- \mathcal{M}_i^T [\mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_m]^{-1}$$

$$\bar{x}_i^+ = \bar{x}_i^- + K_i [y_i - \mathcal{M}_i \bar{x}_i^-]$$

$$\Sigma_i^+ = [I - K_i \mathcal{M}_i] \Sigma_i^-$$

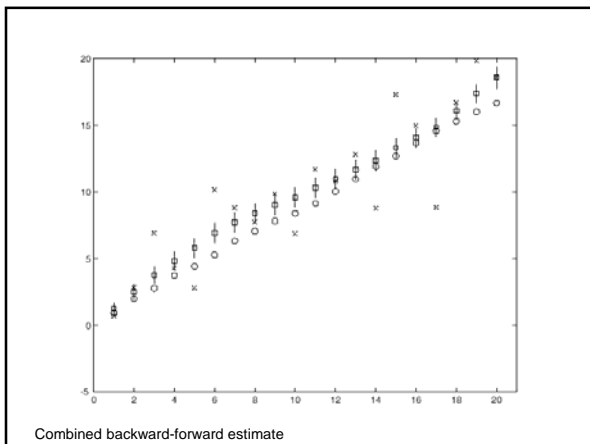
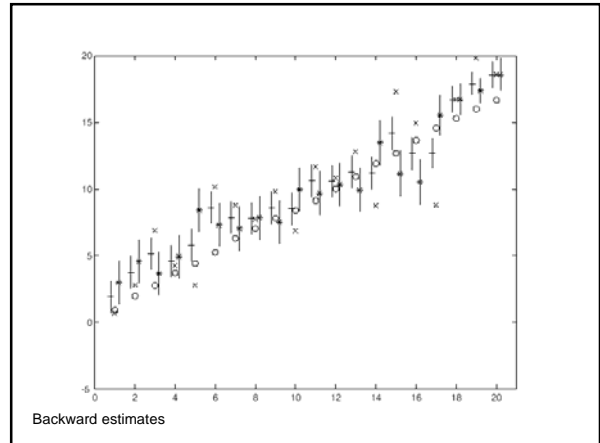
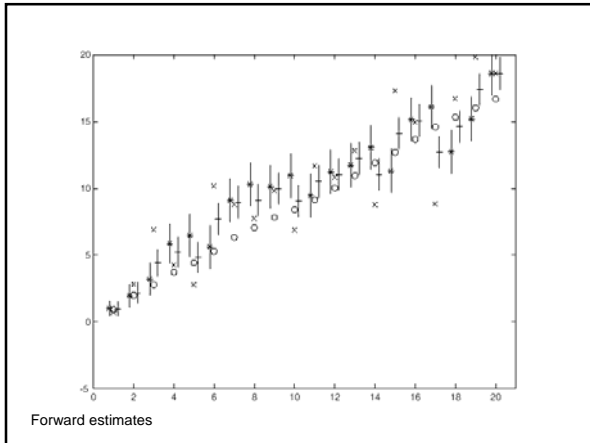


o: state; x: measurements; *: predicted means; +: corrected means



Smoothing

- Idea
 - We don't have the best estimate of state - what about the future?
 - Run two filters, one moving forward, the other backward in time.
 - Now combine state estimates
 - The crucial point here is that we can obtain a smoothed estimate by viewing the backward filter's prediction as yet another measurement for the forward filter
 - so we've already done the equations



Hand Gesture Recognition System for HCI and Sign Language Interfaces

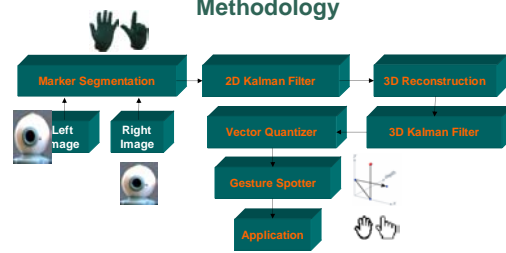
Cem Keskin
Ayşe Naz Erkan
Furkan Kırış
Özge Güler
Lale Akarun

The System Overview

- Real-time gesture recognition system
- Marker based hand segmentation
- 3D from stereo vision using two web cams
- Support for hand posture recognition
- Modules to
 - register the markers
 - train the system,
 - calibrate the cameras
 - choose HCI commands to trigger gestures
 - support tracking of two hands



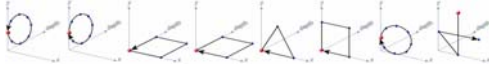
Methodology



- **Camera Calibration:** Special calibration object – using least squares approach
- **Marker Segmentation:** Hue based connected components using double thresholding
- **3D Reconstruction:** 3D reconstruction from stereo vision – least squares approach
- **Vector Quantization:** 15 codewords symbolizing 3D spatial motion
- **Gesture Modeling:** Left-Right HMMs for gesture modeling
- **Gesture Spotting:** A dynamic threshold HMM to spot meaningful gestures
- **Gesture Training:** Baum-Welch algorithm to estimate HMM parameters

Test Results

- We train the system with 8 3D gestures



- We bind the gestures to commands of a third party Windows painting application and test the recognition rates

Gesture	# States	# Training	# Trials	# Correct	# Wrong	Rate %	Tool
1	8	65	20	20	0	100	Zoom
2	4	56	20	20	0	100	Arrow
3	8	59	20	20	0	100	Freehand
4	4	59	20	20	0	100	Eraser
5	3	60	20	20	0	100	Brush
6	4	56	20	19	1	95	Selection
7	8	57	20	19	1	95	Draw Object
9	3	57	20	20	0	100	Draw Line

- With 2 misclassifications out of 160 trials, the system yields a recognition performance of 98.75%

Improved Hand Tracking and Gesture Recognition with Posture Information

- **Advanced Hand Tracking:**

- **Aim:** Robust hand tracking without using markers

- **Considerations:**

- Different color spaces for connected components algorithm
 - RGB, Normalized RGB, RGB Ratios, HSI, TSL, LUX, CIE L*a*b and CIE L*u*v
- Mean-Shift segmentation
 - A kernel-based density estimation technique for detection and clustering of the skin color
- Particle Filters
 - A method that tries to solve the generalized tracking problem by approximation
 - Similar to genetic algorithm
 - Overcomes restrictions of the Kalman filter



Hand Shape Recognition Module

- A module to identify the static pose of the hand
- Required for sign language or similar applications
- Also useful for HCI applications

- **Our Approach:**

- **Model based analysis of hand shapes:** minimize the difference between a predefined model and the input images
- We use a genetic algorithm (GA) for global search and the downhill-simplex method (DS) for local search
- The similarity measure for the model-input matching is the non-overlapping areas of the model silhouette and the hand region in the input image

- **Hand Model:**

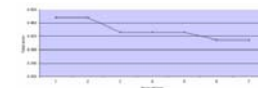
- A complete geometric hand model constructed with simple quadrics, namely cylinders and spheres, with 22 DOFs



Test Results



POPULATION SIZE: 600
CROSSOVER % 70
MUTATION % 8
ELITICISM % 8
DS TOLERANCE 0.005



10 Generations of GA 25.13s
DS 15.47s
Population Size: 600
DS Tolerance: 0.005

8 Generations of GA 20.84s
DS 15.81s
Population Size: 600
DS Tolerance: 0.005

4 Generations of GA 5.28s
DS 14.84s
Population Size: 400
DS Tolerance: 0.005

- For the general case, where all parameters are estimated, this module doesn't work in real time
- We will test the module for classification problems with restricted subsets

Fusion of Hand Gesture and Posture Information

- We will convert the HMMs to Input/Output HMMs
 - Take gesture codebook sequence as the output sequence
 - Take pose information sequence as the conditional input sequence
- Several reasons to use IOHMMs instead of HMMs
- Disadvantages of HMMs:
 - Weak incorporation of context
 - Ineffective coding of actual duration of gestures and gesture parts
 - Not good for prediction
 - Not good for synthesis – for a visualization module
- IOHMMs overcome these problems:
 - Better learning of long term dependencies
 - Effective modeling of duration
 - Represent data with richer, non-linear models
 - More discriminant training
- Current research area: A dynamic threshold model for IOHMMs

CONDENSATION

CONDITIONAL DENSITY PROPAGATION

Presented By FURKAN KIRAÇ
For CmpE 699 – Guided Research Course

Goal

- Model-based visual tracking in dense clutter at real time.

Introduction

- The problem of tracking curves in dense visual clutter is **challenging**.
- **Kalman filtering is inadequate** because it is based on **Gaussian densities** which, being **unimodal**, cannot represent simultaneous alternative hypotheses.
- The Condensation algorithm uses '**factored sampling**'.
- The probability distribution of possible interpretations is represented by a **randomly generated set**.
- Condensation uses **learned dynamical models**, together with visual observations, to propagate the random set over time.
- **The result is highly robust tracking of agile motion**.
- Despite the use of stochastic methods, the algorithm runs in **real-time**.

Kalman filters and data-association

- The tracking of shape and position over time, has been dealt with thoroughly by **Kalman filtering**, in the relatively clutter-free case in which **$p(x_t)$** can satisfactorily be **modeled as Gaussian** and can be applied to curves.
- These solutions work relatively poorly in clutter which causes the density for **x_t** to be multi-modal and therefore **non-Gaussian**.
- With simple, discrete features such as points or corners combinatorial data association methods can be effective with clutter but combinatorial methods do not apply naturally to curves.
- **There remains a need for an appropriately general probabilistic mechanism to handle multi-modal density functions.**

Approach

- Probabilistic framework for **tracking curves in clutter** using an iterative sampling algorithm.
- **Model motion and shape of target.**
- Top-down approach.
- Simulation instead of analytic solution.

Tracking curves in clutter

- Establish a **stochastic framework for tracking curves in visual clutter** using a **sampling algorithm**.
- The problem is to **track outlines and features of foreground objects, modeled as curves**, as they move in substantial clutter.
- This is challenging because elements in the background clutter may mimic parts of foreground features.
- In the most severe case of camouflage, the background may consist of objects similar to the foreground object, for instance when a person is moving past a crowd.
- Approach aims to dissolve the resulting ambiguity by applying probabilistic models of object shape and motion to analyze the video stream.
- The degree of generality of these models is pitched carefully: sufficiently specific for effective disambiguation but sufficiently general to be broadly applicable over entire classes of foreground objects.

Modeling shape and motion

- Once an object has been located approximately, tracking it in subsequent images becomes more efficient computationally especially if **motion is modeled as well as shape**.
- One important facility is the **modeling of curve segments with B-Splines which interact with images or image sequences**.
- **This is more general than modeling entire objects but more clutter resistant than applying signal processing to low level corners or edges**.
- The B-spline curves could, in theory, be parameterized by their control points. In practice this allows too many degrees of freedom for stable tracking and **it is necessary to restrict the curve to a low dimensional parameter x , for example over an affine space, or more generally allowing a 'shape space' of non rigid motion**.

Probabilistic Framework

- Object dynamics form a temporal Markov Chain

$$p(\mathbf{x}_t | \mathcal{X}_{t-1}) = p(\mathbf{x}_t | \mathbf{x}_{t-1})$$

- Observations are independent

$$p(\mathcal{Z}_{t-1}, \mathbf{x}_t | \mathcal{X}_{t-1}) = p(\mathbf{x}_t | \mathcal{X}_{t-1}) \prod_{i=1}^{t-1} p(\mathbf{z}_i | \mathbf{x}_i).$$

- Use Bayes' rule

Stochastic Dynamics

- **Second order models** are used in this work. The dynamics are entirely determined therefore by the form of the conditional density, for instance

$$p(x_t | x_{t-1}) \propto \exp -\frac{1}{2}(x_t - x_{t-1} - 1)^2$$

- represents a **one-dimensional random walk (discrete diffusion)** whose **step length is a standard normal variate, superimposed on a rightward drift at unit speed**.
- Of course, for realistic problems, the state \mathbf{x} is multi-dimensional and the density is more complex.

Notation

- X State vector (curve's position and orientation)
- Z Measurement Vector (image edge locations)
- $p(X)$ Prior probability of state vector; summarizes prior domain knowledge by independent measurements
- $p(Z)$ Possibility of measuring Z ; fixed for any given image
- $p(Z|X)$ Possibility of measuring Z given that the state is X ; compares image to expectation based on state
- $p(X|Z)$ Possibility of X given that measurement Z has occurred; called state posterior

Tracking as Estimation

- Compute state posterior, $p(X|Z)$, and select next state to be the one that maximizes this (Maximum a Posteriori estimate)
- Measurements are complex and noisy, so posterior cannot be evaluated in closed form
- Particle filter (iterative sampling) idea: Stochastically approximate the state posterior with a set of N weighted particles
- Use Bayes' rule to compute $p(X|Z)$

Bayes' Rule

This is what you can evaluate

$p(\mathbf{X} | \mathbf{Z})$

This is what you want. Knowing $p(\mathbf{X} | \mathbf{Z})$ will tell us what is the most likely state \mathbf{X} .

This is what you may know a priori, or what you can **predict**

$p(\mathbf{X})$

This is a constant for a given image

$$p(\mathbf{X} | \mathbf{Z}) = \frac{p(\mathbf{Z} | \mathbf{X}) p(\mathbf{X})}{p(\mathbf{Z})}$$

Temporal propagation of conditional densities

- The Kalman filter as a recursive linear estimator is a special case, applying only to Gaussian densities, of a more general probability density propagation process.
- In the simple Gaussian case, the diffusion is purely linear and the density function evolves as a Gaussian pulse that translates, spreads and is reinforced, remaining Gaussian throughout, as in figure, a **process that is described analytically and exactly by the Kalman filter.**

Temporal propagation of conditional densities

- The random component of the dynamical model leads to spreading - increasing uncertainty - while the deterministic component causes density function to drift.
- The effect of an external observation z_i is to superimpose a reactive effect on the diffusion in which the density tends to peak in the vicinity of observations.
- In clutter, there are typically several competing observations and these tend to encourage a non-Gaussian state-density.

Temporal propagation of conditional densities

- The Condensation algorithm** is designed to address this more general situation.
- It has the striking property that, despite its generality, it is a considerably simpler algorithm than the Kalman filter.
- Moreover, despite its use of random sampling which is often thought to be computationally inefficient, the Condensation algorithm **runs in near real-time.**
- This is because tracking over time maintains relatively tight distributions for shape at successive time-steps, and particularly when given the accurate, learned models of shape and motion.

Factored Sampling

- Generate a set of samples that approximates the posterior $p(\mathbf{X} | \mathbf{Z})$
- Sample set $\mathbf{s} = \{\mathbf{s}^{(0)}, \dots, \mathbf{s}^{(N)}\}$ generated from $p(\mathbf{X})$; each sample has a weight

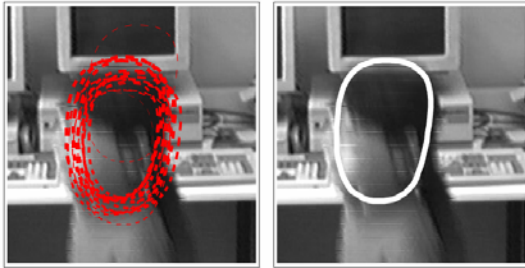
$$\pi_i = \frac{p_z(\mathbf{s}^{(i)})}{\sum_{j=1}^N p_z(\mathbf{s}^{(j)})}$$

$$p_z(\mathbf{x}) = p(\mathbf{z} | \mathbf{x}),$$

Factored Sampling

A set of points, the centers of the blobs in the figure, is sampled randomly from a prior density $p(\mathbf{x})$. Each sample is assigned a weight (depicted by blob area) in proportion to the value of the observation density. The weighted point set then serves as a representation of the posterior density, suitable for sampling.

Estimating Target States



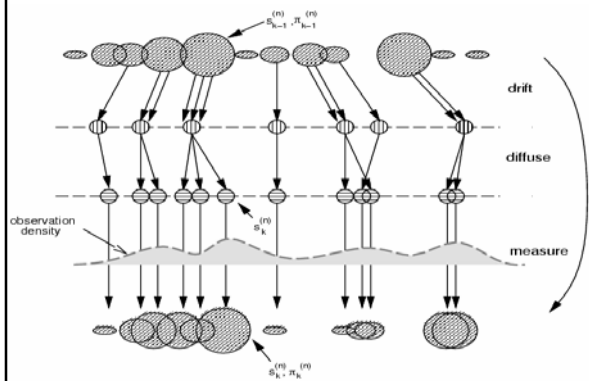
a) State Samples

b) Mean of Weighted State Samples

CONDENSATION Algorithm

1. **Select:** Randomly select N particles from $\{S_{t-1}^{(n)}\}$ based on weights $\pi_{t-1}^{(n)}$; same particle may be picked multiple times (factored sampling)
2. **Predict:** Move particles according to deterministic dynamics (drift), then perturb individually (diffuse)
3. **Measure:** Get a likelihood for each new sample by comparing it with the image's local appearance, i.e., based on $p(z_t|x_t)$; then update weight accordingly to obtain $\{(S_t^{(n)}, \pi_t^{(n)})\}$

One time step in Condensation



Notes on Updating

- Enforcing plausibility: Particles that represent impossible configurations are discarded.
- Diffusion is modeled with a Gaussian
- Likelihood function: Convert “goodness of prediction” score to pseudo-probability
 - More markings closer to predicted markings gives higher likelihood

Object Motion Model

- For video tracking we need a way to propagate probability densities, so we need a “motion model” such as
 - $X_{t+1} = A X_t + B W_t$ where W is a noise term and A and B are state transition matrices that can be learned from training sequences
- The state, X , of an object, e.g., a B-spline curve, can be represented as a point in 6D state space of possible 2D affine transformations of the object

Observation Process

