

CONDENSATION

Conditional Density Propagation for Visual Tracking

Michael Isard and Andrew Blake

Presented by Neil Alldrin

Department of Computer Science & Engineering

University of California, San Diego

La Jolla, CA 92093, USA

Outline

- The problem / motivation
- Previous methods - Kalman filters
- CONDENSATION
 - Modeling shape and motion
 - Factored sampling
 - The CONDENSATION algorithm
 - Benefits / limitations
- Demonstration

The Goal

What are we trying to do?

- Track an object throughout a video sequence in real-time
- Object can move and change shape



Motivation

- Tracking has many potential applications
 - Surveillance
 - Missile guidance systems
 - Virtual mouse pointer
- Tracking is also useful for more advanced techniques such as behavior recognition

Difficulties

Some difficulties specific to tracking include

- Object representation
- Background clutter
- Computational efficiency
- Initialization
- Occlusion

Existing Methods

- Tracking via repeated recognition (Sullivan & Carlson; Mori & Malik)
- Articulated Tracking (Breglar; Blark)
- *Kalman Filters* (Kalman - 1960)

Kalman Filters

Basic Idea

- Model the object being tracked
- At each time-frame,
 - *Prediction* - Predict where object should be
 - *Measurement* - Observe where object went
 - *Assimilation* - Update object model by combining the two

Modeling shape and motion

System Model

- Object modelled by a *state vector*, \mathbf{x} , and a set of equations, the *system model*
- The state is a time-dependent vector \mathbf{x}_t of system variables
- The system model is a vector equation describing the evolution of the state in time

Modeling shape and motion

Assumptions

- The state does not change much between consecutive time instants
- Linear system model
 - $\mathbf{x}_t = \Phi_{t-1}\mathbf{x}_{t-1} + \xi_{t-1}$
 - Φ_{t-1} is the *state transition matrix* at time $t - 1$
 - ξ_{t-1} is a random vector modelling additive system noise

Modeling shape and motion

Example

- Want to model a basketball
- Parameterize as a circle, $\mathbf{x}_0 = (x, y, r, v_x, v_y, v_r)^T$



Modeling shape and motion

Example

- The state transition matrix Φ_0 is used to predict the next object state

- $$\Phi_0 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

- $$\mathbf{x}_1 = \Phi_0 \mathbf{x}_0 + \xi_0$$

- $$\mathbf{x}_1 = (x + v_x + \xi^x, y + v_y + \xi^y, r + v_r + \xi^r, \dots)^T$$

Modeling shape and motion

Measurement Model

- At each time step a measurement, \mathbf{z}_t , of the state vector is taken
- Assume a linear relationship between measurements and the system state
 - $\mathbf{z}_t = H_t \mathbf{x}_t + \mu_t$
 - H_t is the *measurement matrix*
 - μ_t is a random vector modelling additive noise

Modeling shape and motion

Uncertainty

- Problem : The system model and feature measurements are imprecise
- Solution : Quantify the uncertainty with a probability density function, specifically a Gaussian
- Define covariance matrix P_t corresponding to state vector \mathbf{x}_t

The Kalman Filter Algorithm

- Want to estimate the system's state, $\hat{\mathbf{x}}_t$, at time t using the system model prediction from time $t - 1$ and the measurements, \mathbf{z}_t , taken at time t
- Three steps
 - Prediction
 - Measurement
 - Assimilation

The Algorithm

Symbols

- $\hat{\mathbf{x}}'_t$ is the state estimate predicted *before* obtaining the measurement \mathbf{z}_t
- $\hat{\mathbf{x}}_t$ is the state estimate computed *after* integrating the measurement \mathbf{z}_t with the prediction $\hat{\mathbf{x}}'_t$
- P'_t and P_t are the *state covariance matrices* for $\hat{\mathbf{x}}'_t$ and $\hat{\mathbf{x}}_t$ respectively
- K_t is the *gain matrix*, specifying the relative importance of the prediction $\hat{\mathbf{x}}'_t$ and the measurement $\hat{\mathbf{x}}_t$
- Q_t and R_t are the covariance matrices of the noise processes ξ_t and μ_t

The Algorithm

Prediction

- Based on object state and system model from time $t - 1$, predict the object state at time t
- $\hat{\mathbf{x}}'_t = \Phi_{t-1} \hat{\mathbf{x}}_{t-1}$
- $P'_t = \Phi_{t-1} P_{t-1} \Phi_{t-1}^T + Q_{t-1}$

The Algorithm

Measurement

- Extract features, \mathbf{z}_t , from the image at time t
- \mathbf{z}_t is assumed to be a linear transformation of the object state, $\mathbf{z}_t = H_t \mathbf{x}_t$

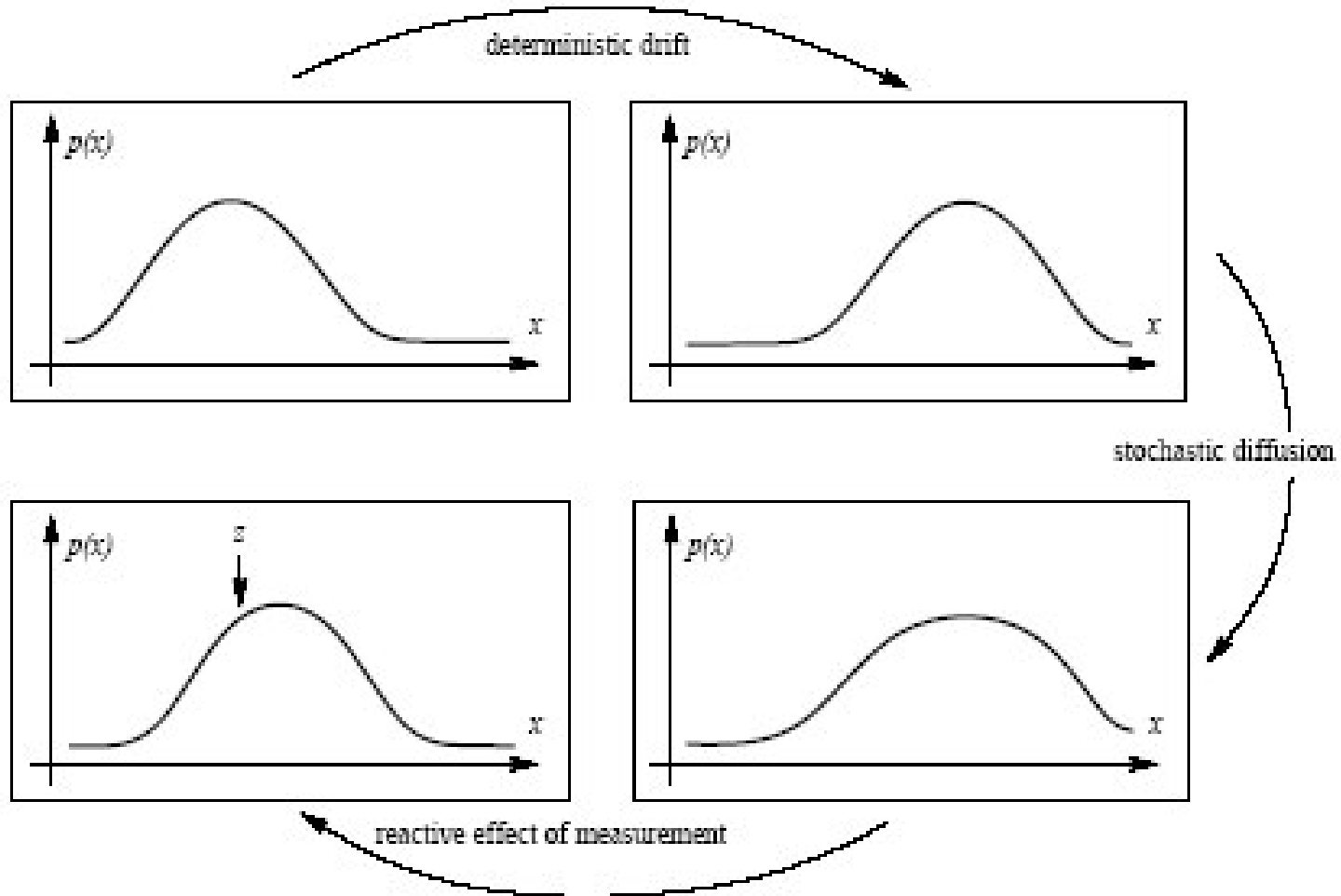
The Algorithm

Assimilation

- Combine the predicted state with the measured state to generate the new object state
 - $\hat{\mathbf{x}}_t = \hat{\mathbf{x}}'_t + K_t(\mathbf{z}_t - H_t\hat{\mathbf{x}}'_t)$
 - $\mathbf{z}_t - H_t\hat{\mathbf{x}}'_t$ is the difference between the measured features and the predicted features
 - $K_t = P'_t H_t^T (H_t P'_t H_t^T + R_t)^{-1}$
- Update the covariance matrix, P_t
 - $P_t = (I - K_t H_t) P'_t$

The Algorithm

Probability density propagation



The Algorithm

Recap : The Kalman filter equations

$$\hat{\mathbf{x}}'_t = \Phi_{t-1} \hat{\mathbf{x}}_{t-1} \quad (1)$$

$$P'_t = \Phi_{t-1} P_{t-1} \Phi_{t-1}^T + Q_{t-1} \quad (2)$$

$$K_t = P'_t H_t^T (H_t P'_t H_t^T + R_t)^{-1} \quad (3)$$

$$\hat{\mathbf{x}}_t = \Phi_{t-1} \hat{\mathbf{x}}_{t-1} + K_t (\mathbf{z}_t - H_t \Phi_{t-1} \hat{\mathbf{x}}_{t-1}) \quad (4)$$

$$P_t = (I - K_t) P'_t (I - K_t)^T + K_t R_t K_t^T \quad (5)$$

Limitations

- Unimodal, Gaussian probability distribution
 - Object state often non-Gaussian
 - For example, what if the object is likely to be in one of two locations, but not inbetween?
- Sensitive to “background clutter”

CONDENSATION

Motivation

- Overcome limitations of Kalman filters; Specifically, allow multimodal probability distributions
- Increase robustness to background clutter

CONDENSATION

Overview

- The primary difference between Kalman filters and the CONDENSATION algorithm is how the state probability density is represented.
 - Kalman filters use a Gaussian
 - CONDENSATION uses factored sampling to approximate arbitrary pdf's
- Other names for this algorithm
 - Particle filters
 - Monte-Carlo methods

CONDENSATION

Outline

- Modelling shape and motion
- Factored sampling
- The algorithm
- Benefits
- Limitations

Modelling Shape and Motion

Symbols

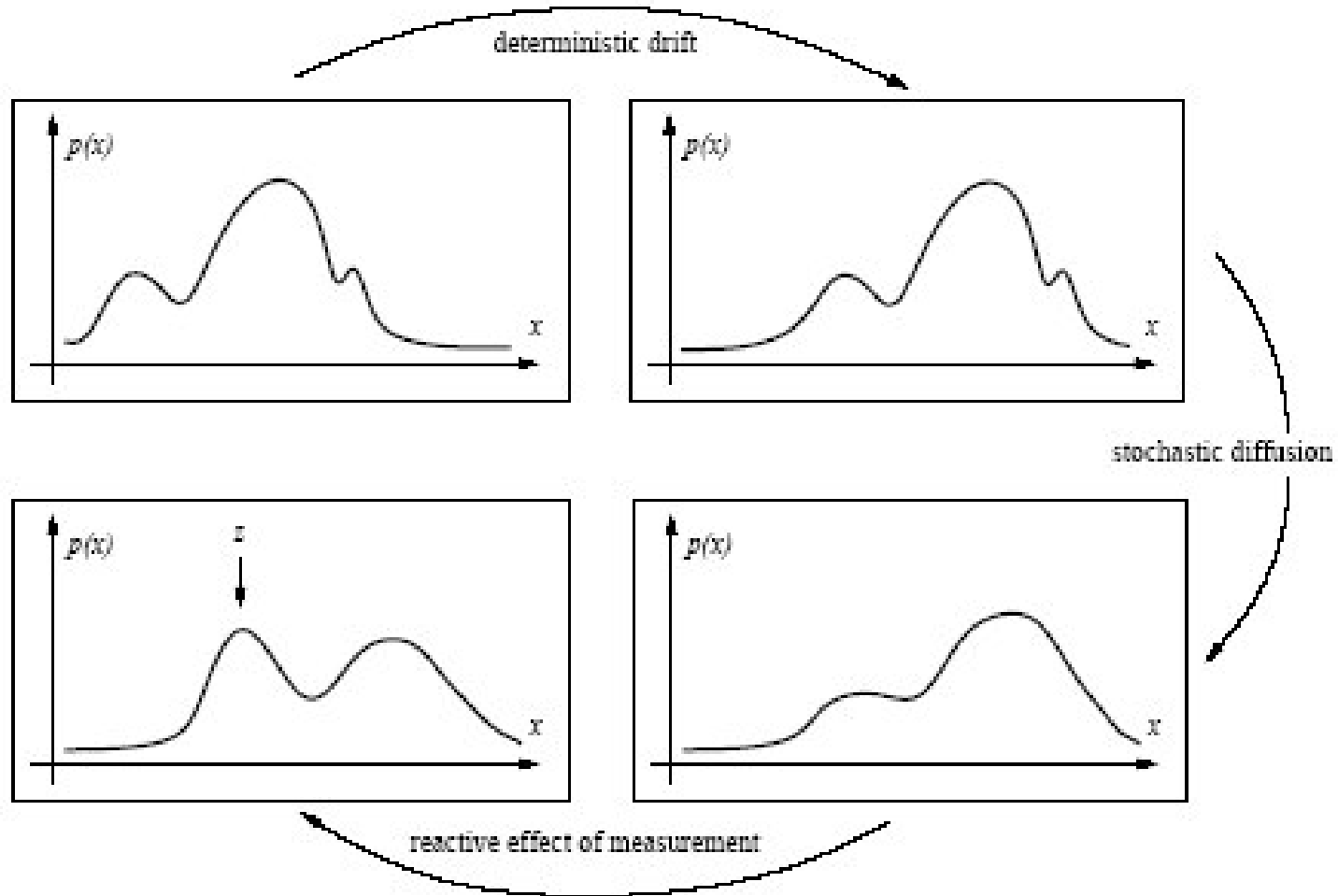
- \mathbf{x}_t is the object state vector at time t
- \mathbf{z}_t is the vector of measured features at time t
- $\chi_t = \{\mathbf{x}_0, \dots, \mathbf{x}_t\}$ is the state history
- $Z_t = \{\mathbf{z}_0, \dots, \mathbf{z}_t\}$ is the history of measured features
- $p(\mathbf{x})$ is the prior probability density for object \mathbf{x}
- $p(\mathbf{z}_t|\mathbf{x}_t)$ is the *observation density*

Modelling Shape and Motion

- Unlike Kalman filters, CONDENSATION allows for arbitrary probability density functions
- The goal is to calculate $p(\mathbf{x}_t|Z_t)$, the probability of object state \mathbf{x}_t given the history of feature measurements

Modelling Shape and Motion

Probability density propagation with CONDENSATION



Modelling Shape and Motion

Assumptions

- Markov assumption
 - Assume the current object state distribution is dependent only on the previous state
 - $p(\mathbf{x}_t | \chi_{t-1}) = p(\mathbf{x}_t | \mathbf{x}_{t-1})$
 - CONDENSATION actually uses a second-order model, but the idea is the same
- Time-independence of the observation density
 - $p(Z_t | \chi_t) = \prod_{i=1}^t p(\mathbf{z}_i | \mathbf{x}_i)$
 - $p(\mathbf{z}_t | \mathbf{x}_t) = p(\mathbf{z} | \mathbf{x})$
 - This allows the observation density to be a static function

Modelling Shape and Motion

Measurement

- Want to determine the observation density $p(\mathbf{z}|\mathbf{x})$ given a hypothesized object state \mathbf{x}
- After many assumptions,

$$p(\mathbf{z}|\mathbf{x}) \propto \exp\left\{-\frac{1}{2r} \int_0^L f(\mathbf{z}_1(s) - \mathbf{r}(s); \mu) ds\right\}$$

where $\mathbf{r}(s)$ is the curve represented by shape parameter \mathbf{x} , $\mathbf{z}_1(s)$ is the closest feature to $\mathbf{r}(s)$, r is a variance constant

- The observation process is ugly, but can be done

Factored Sampling

General Idea

- Factored sampling is a method for approximating probability densities
- The problem is to find $p(\mathbf{x}|\mathbf{z})$, which represents all knowledge about \mathbf{x} deducible from the data \mathbf{z}

Factored Sampling

General Idea

- Bayes' rule can be used in principle:

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

- In practice, however, this cannot be solved in closed form
- Factored sampling generates a random variate that approximates $p(\mathbf{x}|\mathbf{z})$
- Two parts
 - A sample set $S = \{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(N)}\}$
 - Weights, π_i , corresponding to each sample $\mathbf{s}^{(i)}$

Factored Sampling

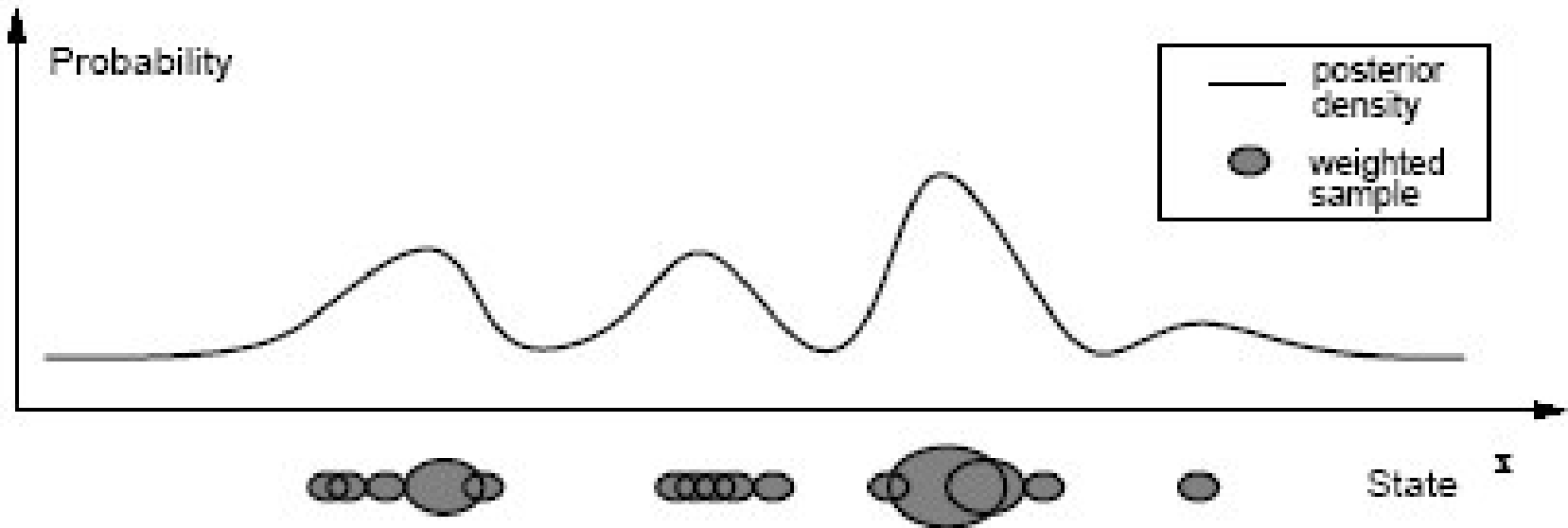
Details

- First, the set $S = \{\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(N)}\}$ is sampled randomly from the prior density $p(\mathbf{x})$
- Each element $\mathbf{s}^{(i)}$ is assigned a weight π_i proportional to the observation density $p(\mathbf{z}|\mathbf{x} = \mathbf{s}^{(i)})$

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

Factored Sampling

One dimensional case



The Algorithm

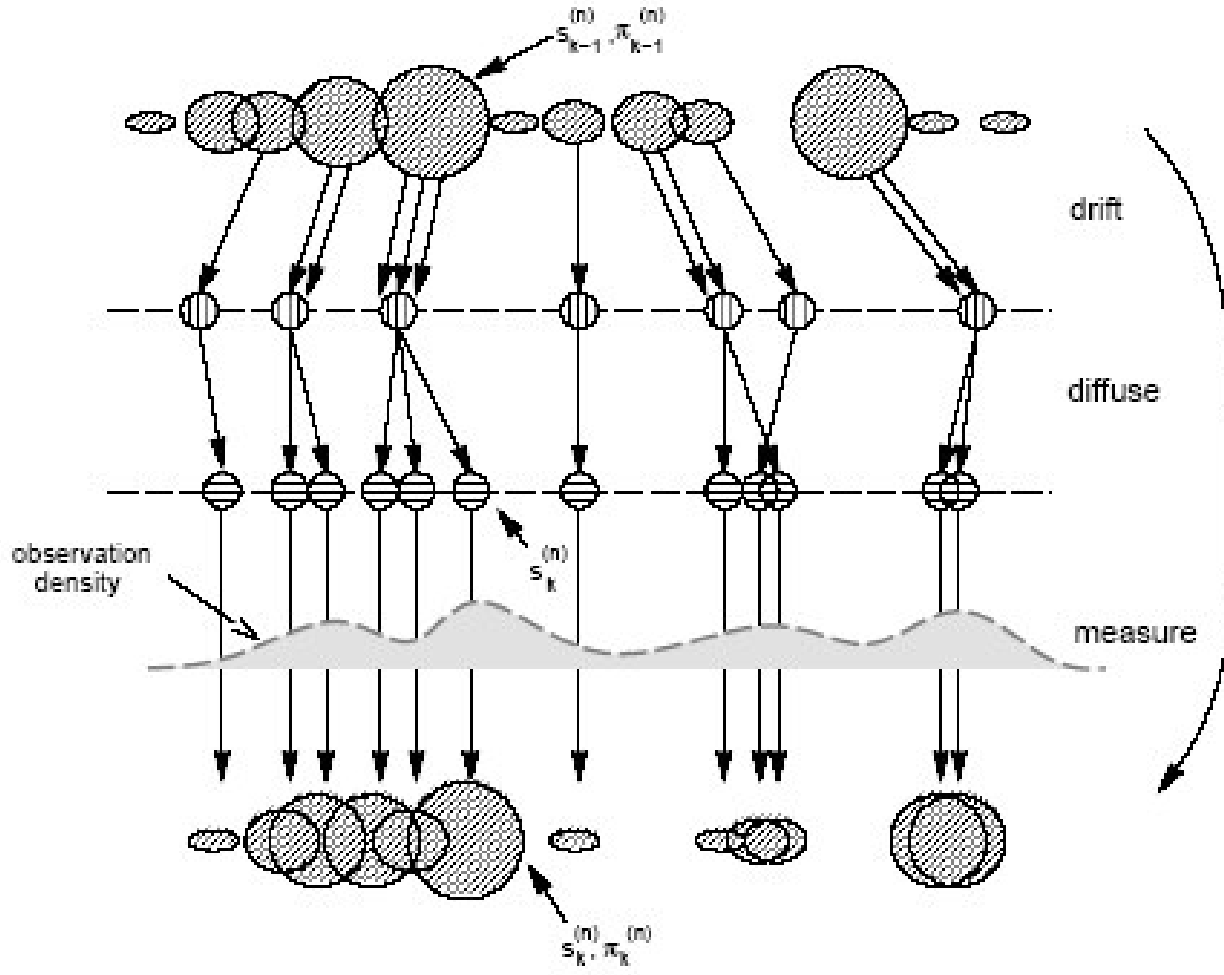
- The goal is to estimate the state probability density $p(\mathbf{x}_t|\mathbf{z}_t)$ given $p(\mathbf{x}_{t-1}|\mathbf{z}_{t-1})$ and z_t
- Since the state density is approximated as a sample set, the algorithm needs to generate a new sample set at each time t
- CONDENSATION follows the same basic iterative steps as a Kalman filter: prediction, measurement, and assimilation

The Algorithm

- Have N samples from time $t - 1$,
 $S_{t-1} = \{\mathbf{s}_{t-1}^{(i)}, \pi_{t-1}^{(i)}, i = 1, \dots, N\}$
- Want to construct S_t
- For $n = 1 : N$,
 1. Select a sample $\mathbf{s}_t^{(n)}$ from S_{t-1} according to probability π_{t-1}
 2. Predict $\mathbf{s}_t^{(n)}$ by sampling from $p(\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{s}_t^{(n)})$
 3. Measure and weight the new position according to image features \mathbf{z}_t ,
$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

The Algorithm

One time-step



Benefits

- Support for multi-modal probability distributions
 - Multi-object support, less likely to lose track of objects
- Robustness
- Computationally efficient
- Adjustable cost/performance ratio (by changing the number of samples used)
- Relatively simple

Limitations

- Initialization
- Object model needs to be known in advance
- Iterative - still prone to getting “stuck”

Demonstration

- Hand
- Leaf
- Pointing finger
- Dancing girl
- Drawing

Conclusion

- Got questions?
- That's all folks