# Tracking I

Vorlesung
„Visuelle Perzeption für Mensch-Maschine Interaktion“

Rainer Stiefelhagen

2013-01-13

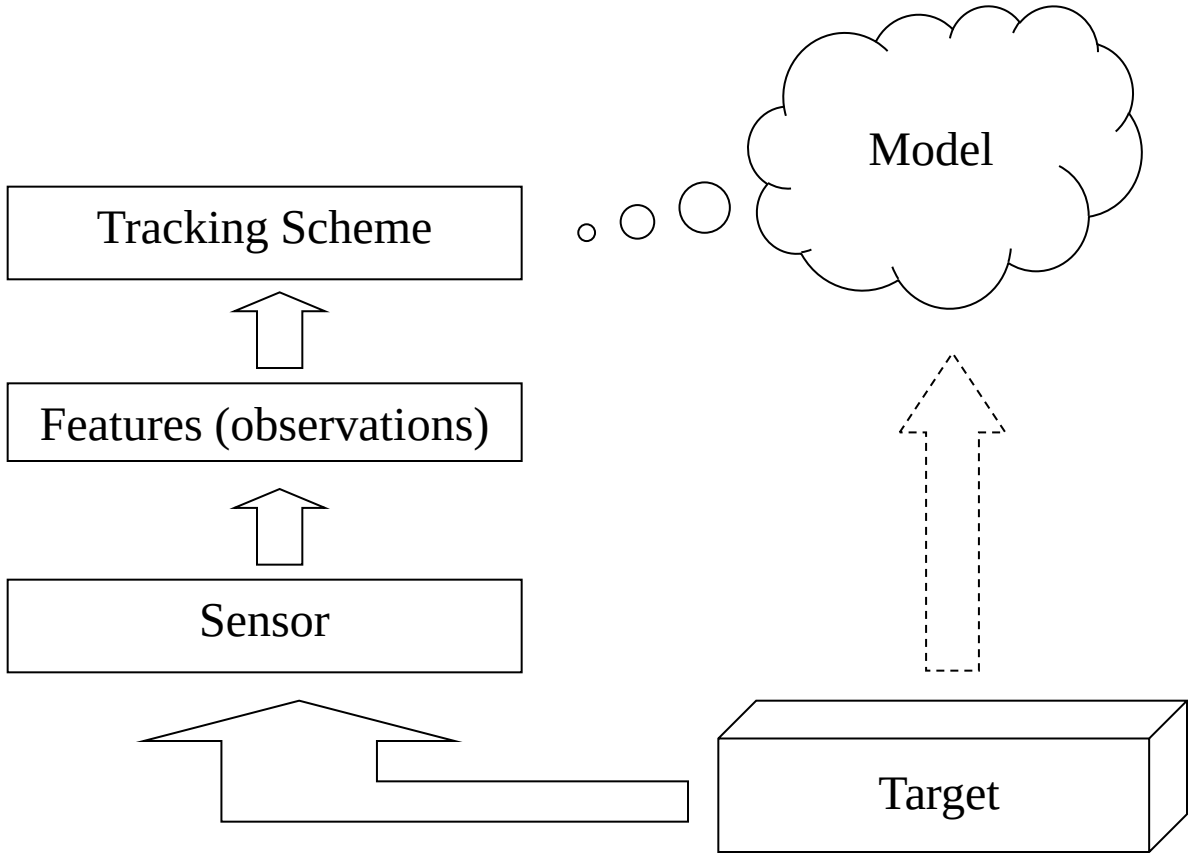
# Overview

- Introduction
- Features
  - Templates
  - Color Models
  - Background models
- Tracking Schemes
  - Meanshift
  - Kalman Filter
  - Particle Filter
- Examples
- Summary

# Definition

- **Tracking**:
  - determine a target's location (and/or rotation, deformation, pose, …) over a sequence of images
  - i.e.: determine a target's ***state*** (location and/or rotation, deformation, pose, …) over a sequence of ***observations*** derived from images
- **Tracking ≠ Detection**!

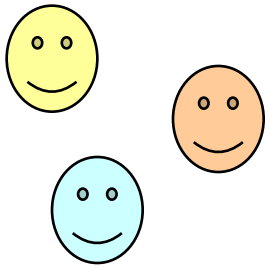
# Tracking


Model
Tracking Scheme
Features (observations)
Sensor
Target

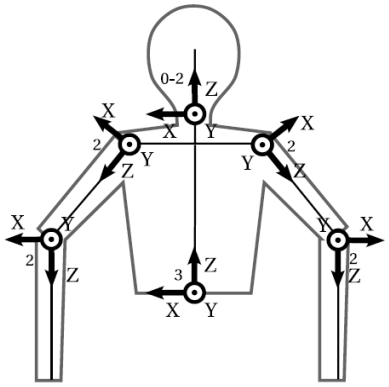
# Target Types

- Single object
  - face, person, …

- Multiple objects
  - group of people, head and hands, …

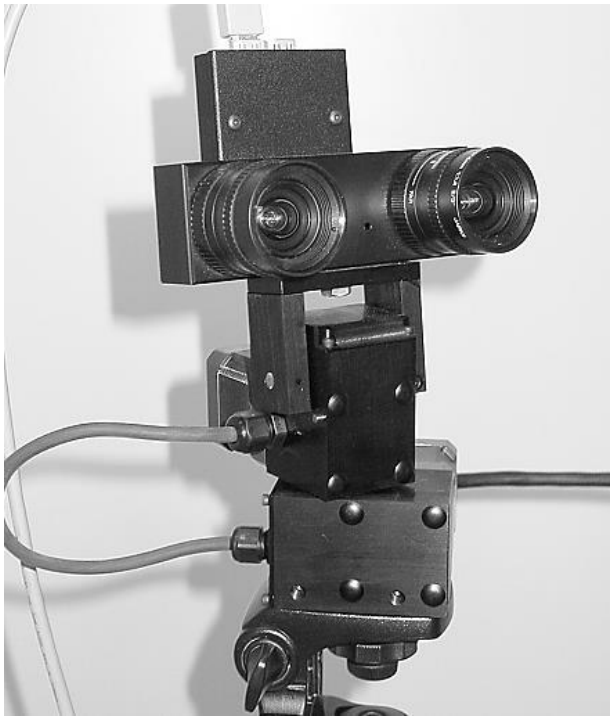


- Articulated body
  - full body, hand



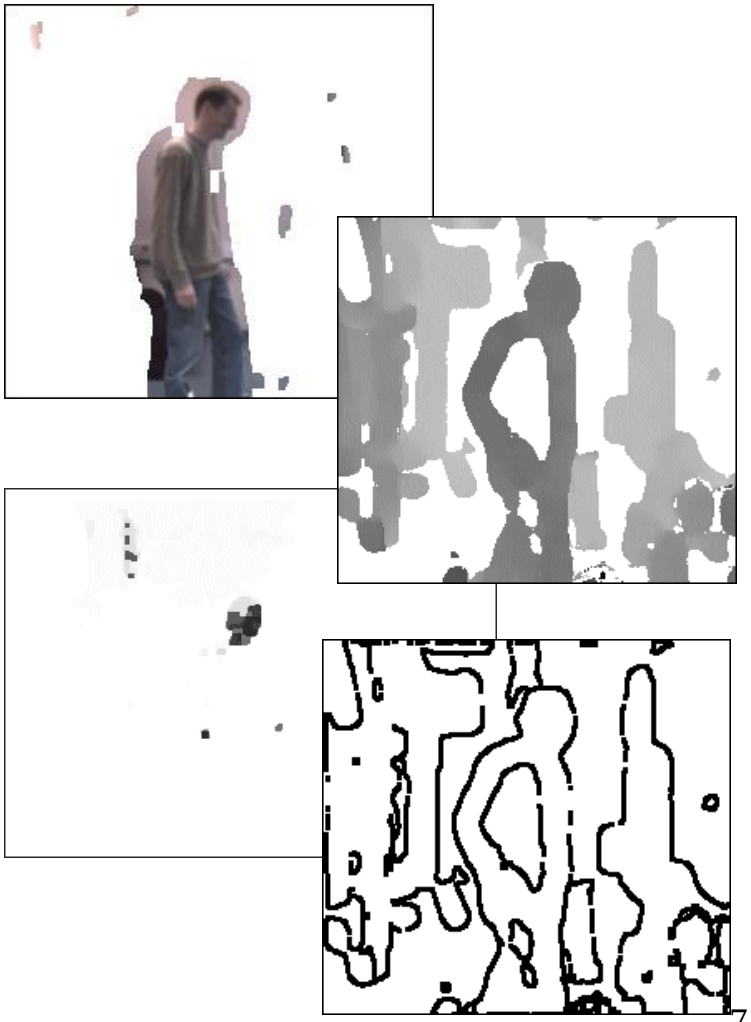
# Sensor Setup

- Single Camera
- Multiple Cameras
  - Wide baseline
  - Narrow baseline (dense stereo)
- Active Cameras
  - Pan, Tilt, Zoom
  - Moving cameras (robot)
- Omnidirectional Cameras
- Cameras + Microphones



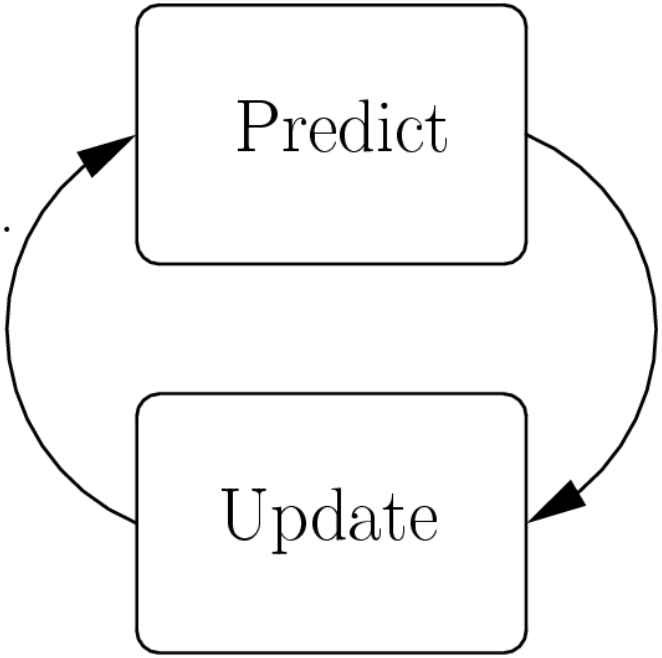
# Features

- Templates
- Color
- Background-Foreground
- Edges
- Dense Disparity
- Detectors (body, body parts)
- …



# Tracking Schemes

- Meanshift
- Kalman Filter
- Particle Filter
- …



# Features

# Template Matching


Image I and template T

SSD scores

Illustration: Murray/Reid, University of Oxford, Computer Vision Course

(SSD: Sum of Squared Differences)

# Template Matching

- Exhaustive search for the best match is slow!
  → use gradient descent and search for local minimum

- Template matching can be extended to handle rotation, scaling and general affine transformation

# Template Matching

- Simple template matching fails for deformable/articulated objects, i.e. hands, mouth, the human body

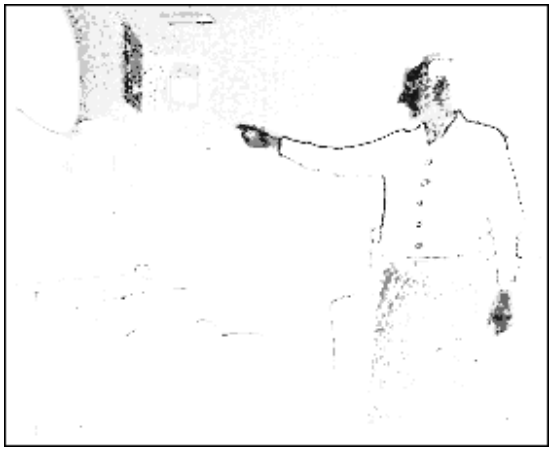
- Are there appearance models that are inherently invariant against scale, rotation, deformation, …?
  - Partial Models
  - Color Models
  - Foreground-Background Models

# Color Models

- *Parametric* color models: Gaussian-Mixture-Model, Linear Decision Boundaries, …
- *Non-parametric* color model: Histogram
- Different Color Spaces:
  RGB, RG, HSV/HSI, YUV, …

# Histogram Backprojection

- The simplest (and fastest) way to utilize histogram information is the histogram backprojection



- Each pixel in the backprojection is set to the value of the histogram bin indexed by the color of the respective pixel

# Histogram Matching

- Backprojection is good, when the color distribution of the target is monomodal.
- Backprojection is not optimal, when the target is multi colored!

→ Build a histogram of the image within the search window, and compare it to the target histogram.

# Histogram Matching

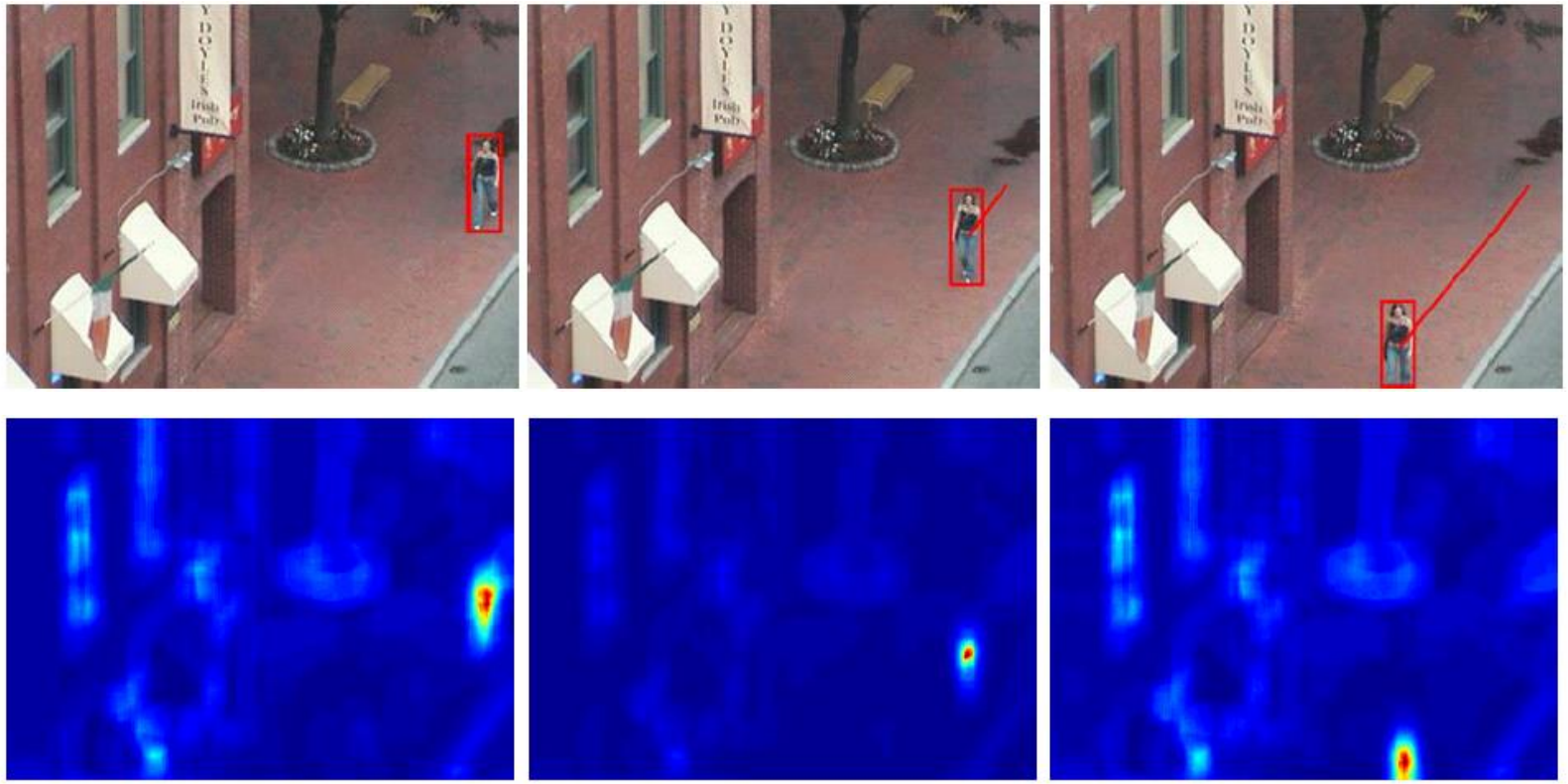


Illustration: F. Porikli, Integral Histogram, CVPR'05.

# Bhattacharyya coefficients

- Distance between two dense histograms *a* and *b*, where *n* is the number of histogram bins:

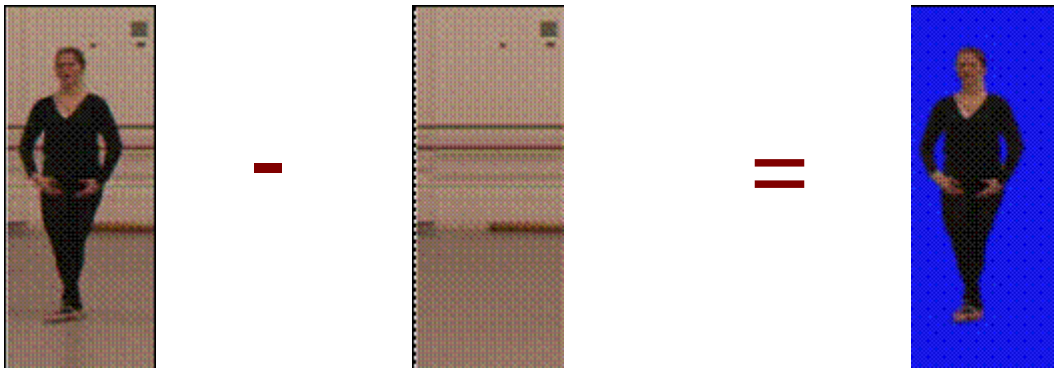
$$d(a,b) = \sqrt{1 - \sum_{i=1}^{n} \sqrt{a_i b_i}}$$

- The histograms have to be normalized: $\sum_{i=1}^{n} a_i = 1$

- Multi-dimensional histograms can be *flattened*.

# Background Models

- Capture the scene *without* the object to be tracked
  → background model
- Each pixel that *differs significantly* from its counterpart in the background model belongs to the target
- Prerequisite: the scene is supposed to be *static*.



current image | background model | foreground (thresholded)

Illustration: A. Efros, Rendering and Image Processing Course, CMU

# Background Models

Problems:

- The static-scene assumption does not hold in general:
  - objects are being moved
  - the camera is being moved
  - illumination changes

  → *background model adaptation*

- The target casts shadows

# Simple Background Model Adaptation

- Background = mean of the past *n* frames

$$b_t = \frac{1}{n}\sum_{i=1}^{n} x_{t-i}$$

- Alpha process

$$b_t = (1-\alpha)\cdot b_{t-1} + \alpha \cdot x_t$$

- New problem: if the target does not move, it becomes part of the background

# Tracking Schemes

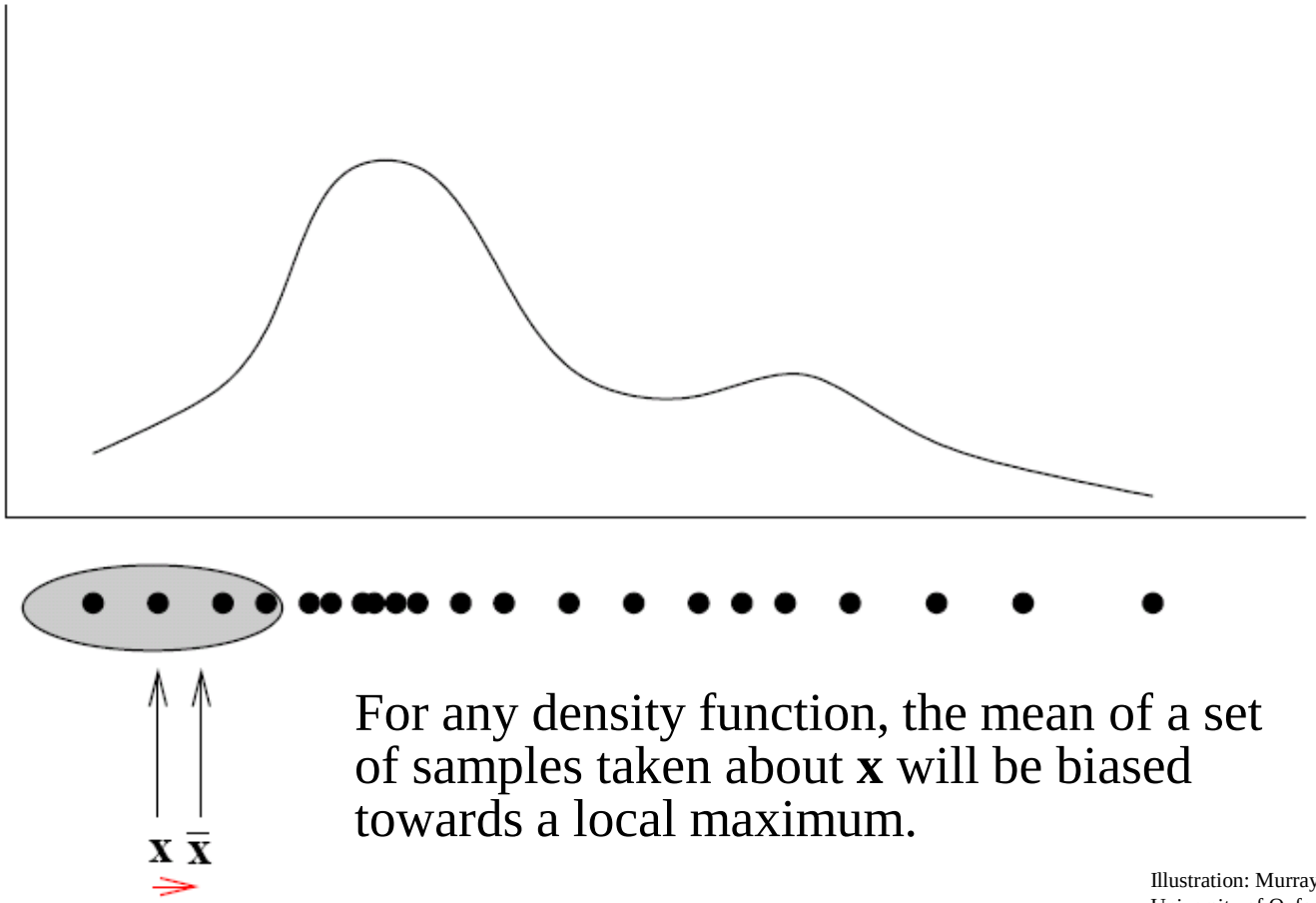
# Meanshift

How to shift the search window in order to maximize the similarity between model and search window?

- Exhaustive search too expensive
- Analytical gradient descent formulation hard to find

→ Use mean-shift estimate of gradient

# Meanshift

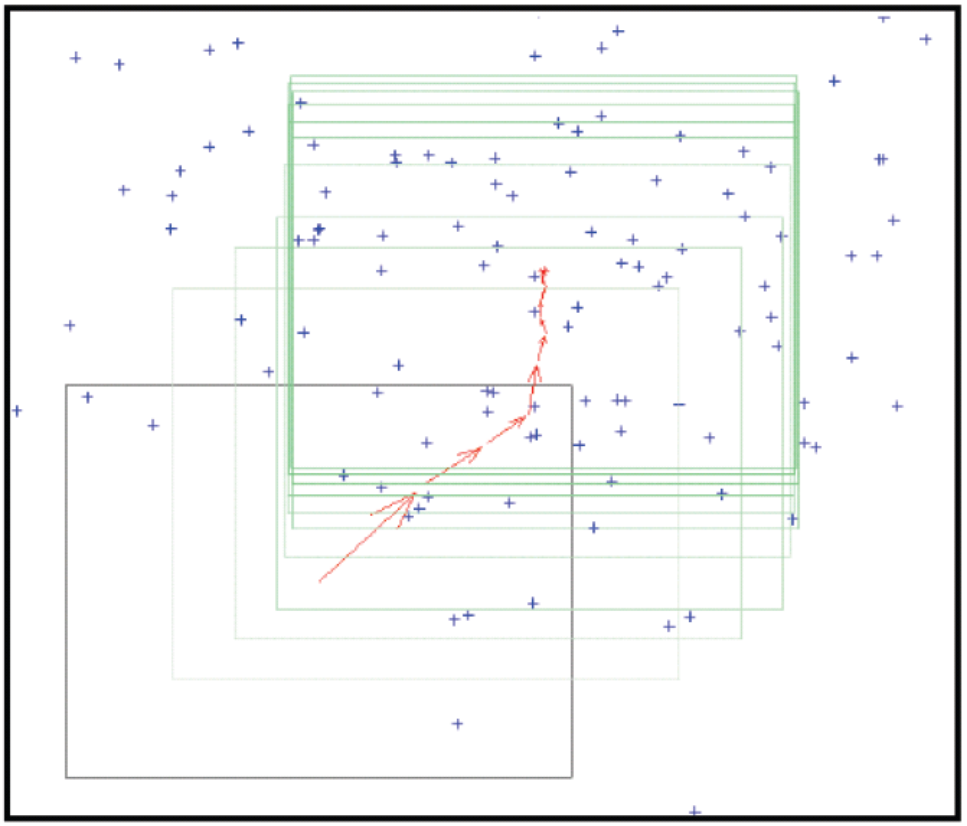



For any density function, the mean of a set of samples taken about **x** will be biased towards a local maximum.

Illustration: Murray/Reid, University of Oxford, Computer Vision Course

# Meanshift

1. Choose search window size
2. Choose initial location
3. Compute mean of samples within the window
4. Center search window to the mean location
5. Repeat 3. and 4. until convergence



# Tracking as State Estimation

- Want to predict state of the system (position, pose, …)
- State cannot directly be measured
- Only certain observations (measurements) can be made
- Observations are noisy !
  - Measurement errors

- What is the most likely state $x$ of the system at a given time, given a sequence of observations $Z_t$ ?

$$\arg\max\ p(x_t \mid Z_t)$$

# Notation

- State: $x_t$
  - State of the system at time *t*
- Observation / measurement: $z_t$
  - Observation / measurement about the certain aspects of the system at time *t*
- Observations up to time t: $z_{1:t}$ or $Z_t$

# Bayes Filter

- Assume state x to be Markov process

$$p(x_t \mid x_{t-1}, x_{t-2}, .., x_0) = p(x_t \mid x_{t-1})$$

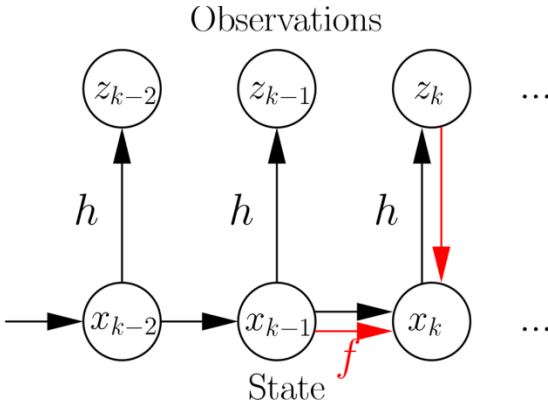
- States $x$ generate observations $z$

$$p(z_t \mid x_t, x_{t-1}, .., x_0) = p(z_t \mid x_t)$$

- Want to estimate most likely state $x_t$ given sequence $Z_t$:

$$\arg\max\ p(x_t \mid Z_t)$$

- Can be estimated recursively



# Bayes Filter

- Predict step:

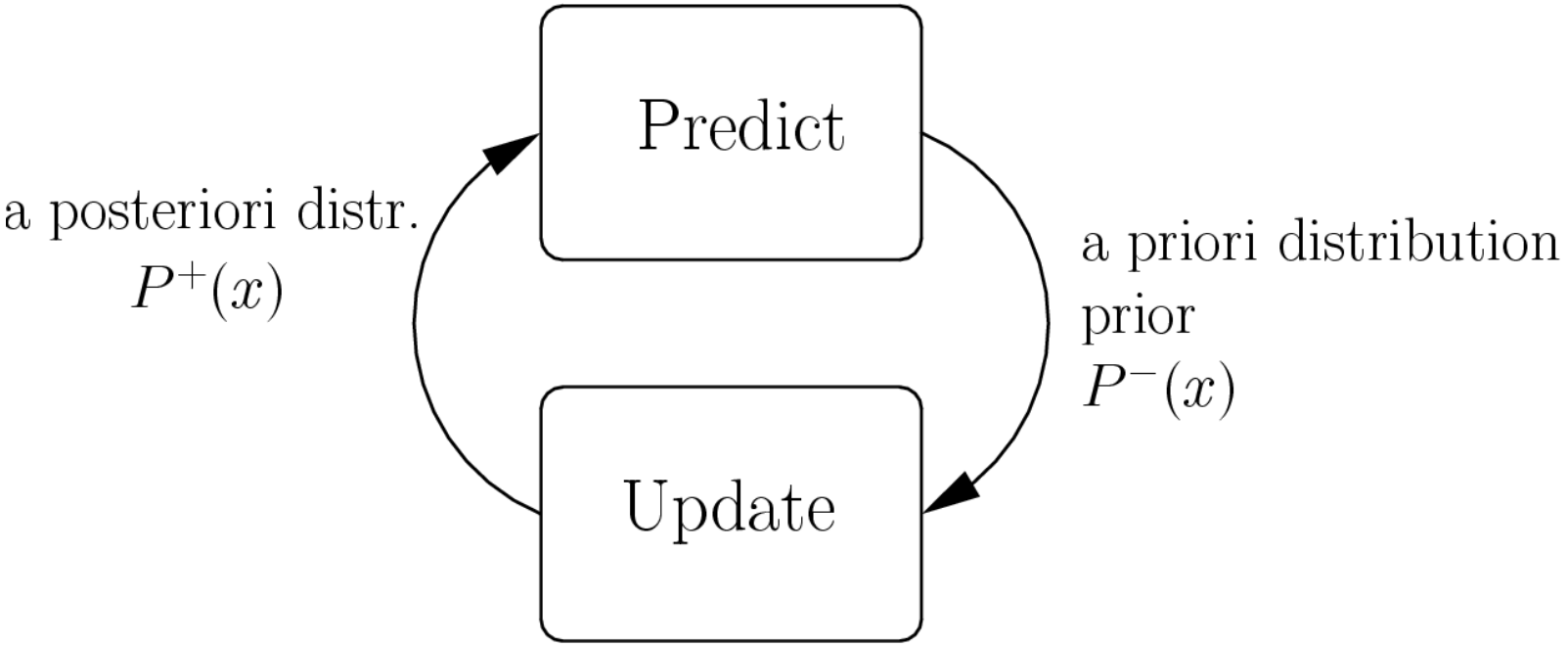
$$p(x_t \mid Z_{t-1}) = \int_{x_{t-1}} p(x_t \mid x_{t-1})\, p(x_{t-1} \mid Z_{t-1})\, dx_{t-1}$$

- Update step:

$$p(x_t \mid Z_t) = \alpha\ p(z_t \mid x_t)\, p(x_t \mid Z_{t-1})$$

- Needed
  - Process model: $p(x_t \mid x_{t-1})$
  - Measurement model: $p(z_t \mid x_t)$

# Bayes Filter



# Kalman Filter

- The Kalman Filter is an instance of a Bayes Filter
- Assumes
  - *Linear* state propagation and measurement model
  - *Gaussian* process and measurement noise

# Kalman Filter

The process to be estimated:

$$x_k = Ax_{k-1} + w_{k-1} \qquad p(w) \sim N(0,Q)$$

$$z_k = Hx_k + v_k \qquad p(v) \sim N(0,R)$$

| | |
|---|---|
| $x_k$: state at time k | $z_k$: observation at time k |
| $A$: transition matrix | $H$: measurement matrix |
| $p(w) \sim N(0,Q)$: process noise | $p(v) \sim N(0,R)$: measurement noise |

# Kalman Filter

Example: A position-velocity motion model

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{z} = \begin{pmatrix} x \\ y \end{pmatrix} \qquad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

# Kalman Filter

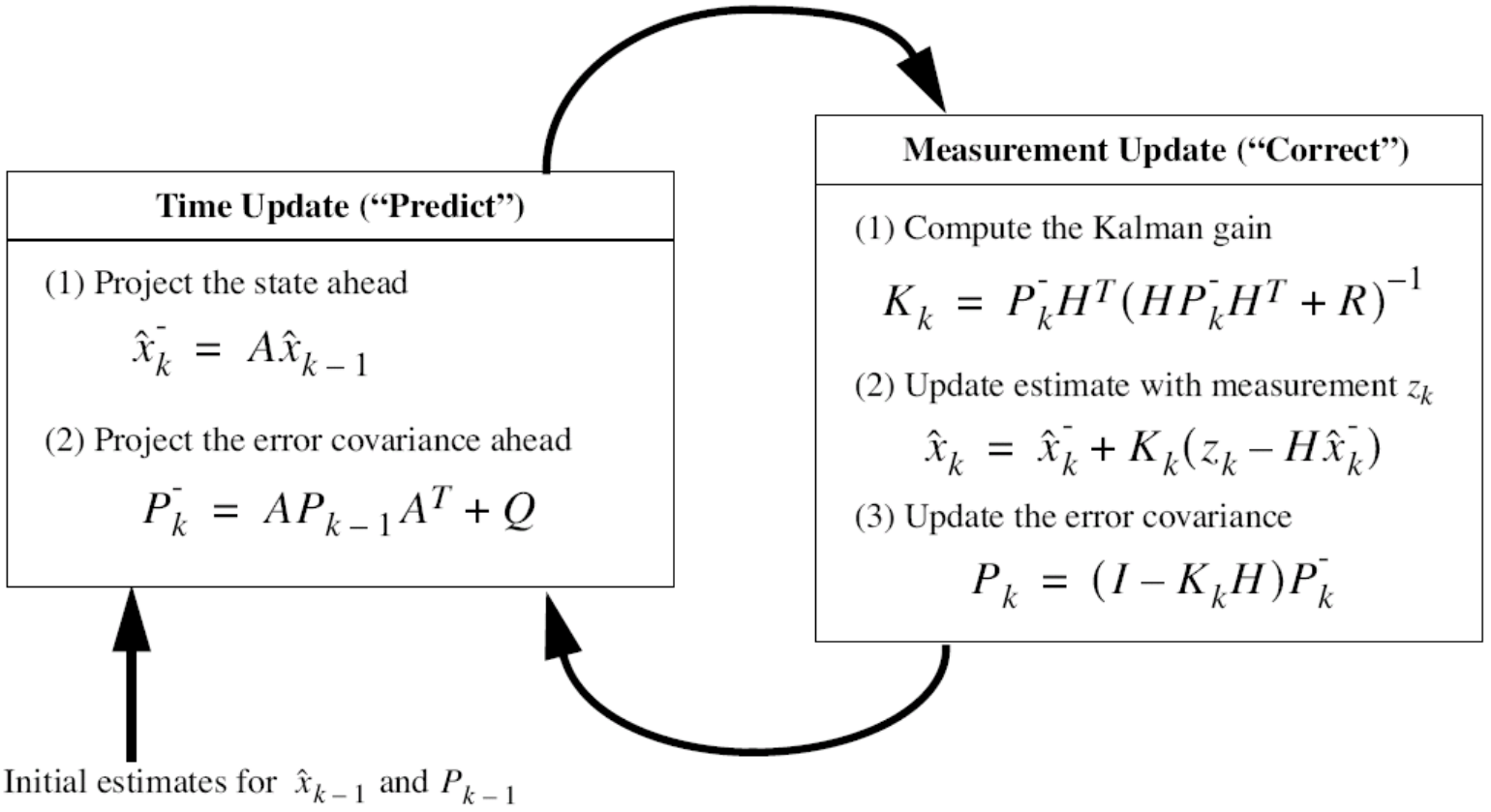



Illustration: Welch&Bishop,
*An Introduction to the Kalman Filter*

# Kalman Filter

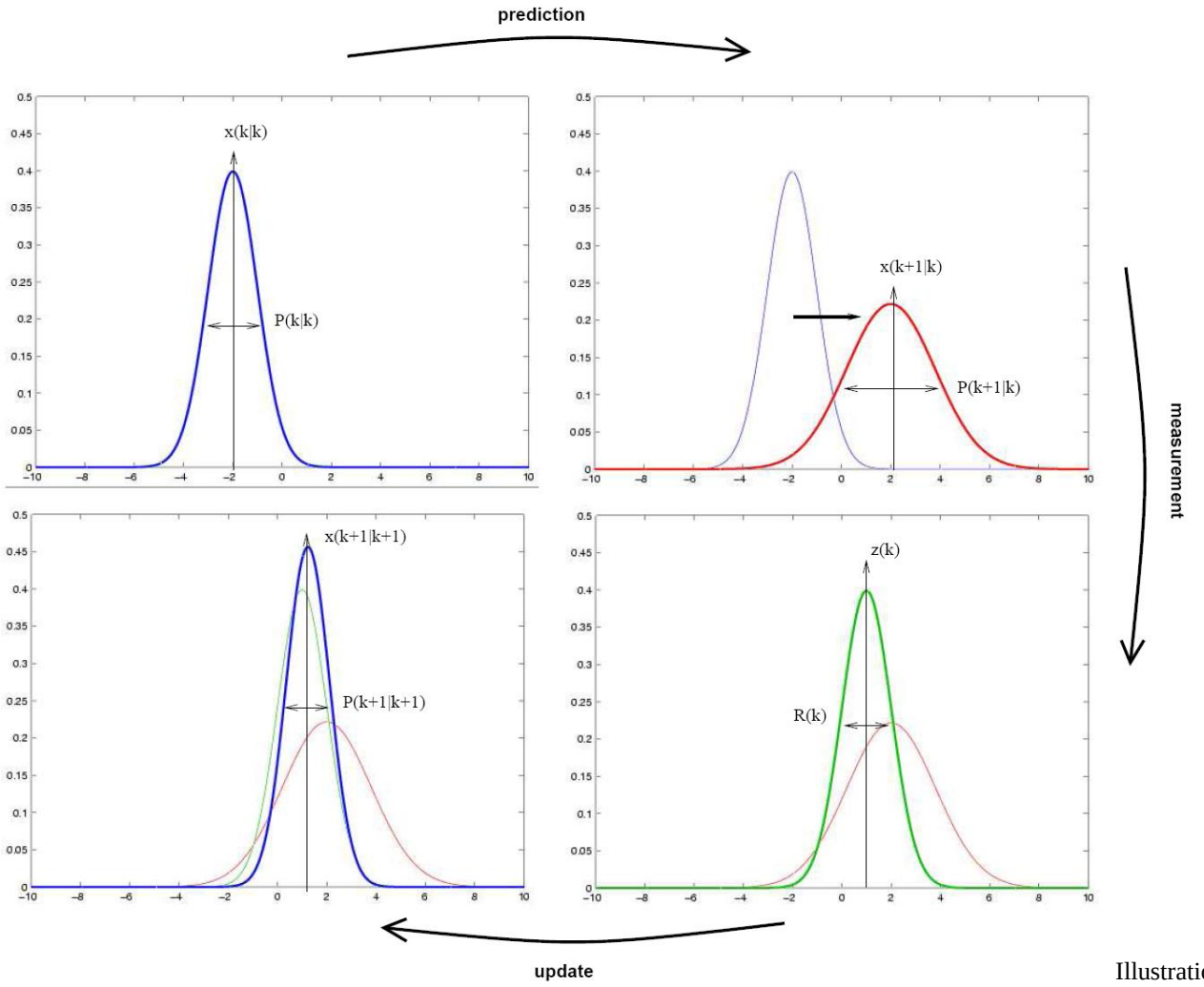



Illustration: Murray/Reid, University of Oxford, Computer Vision Course

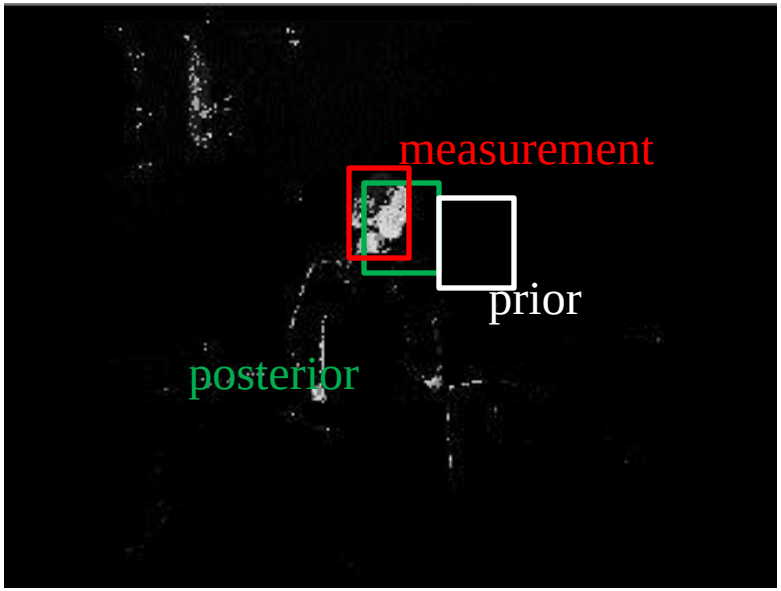
# Kalman Filter

The simple Kalman Filter is not applicable, when the process to be estimated is not linear or the measurement relationship to the process is not linear.

→ The Extended Kalman Filter (EKF) linearizes about the current mean and covariance

# Tracking one Face with a Kalman Filter

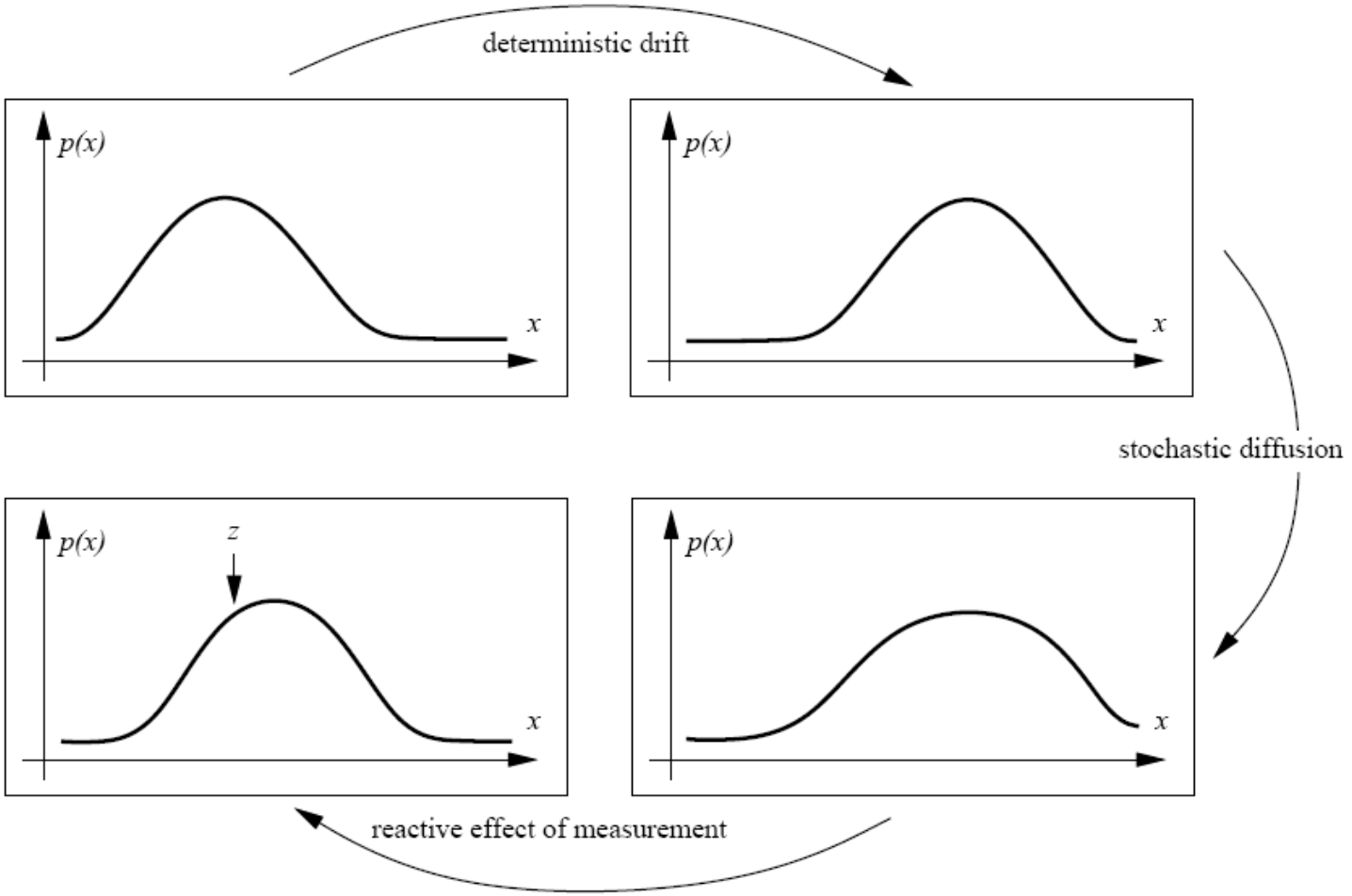
- State: (x,y,scale)
- Observations:
  - skin color
- Propagate old state
  - → prior
- Find best measurement
  - E.g. search for box with maximum skin color support around prior
- Compute update
  - → posterior



# Particle Filter

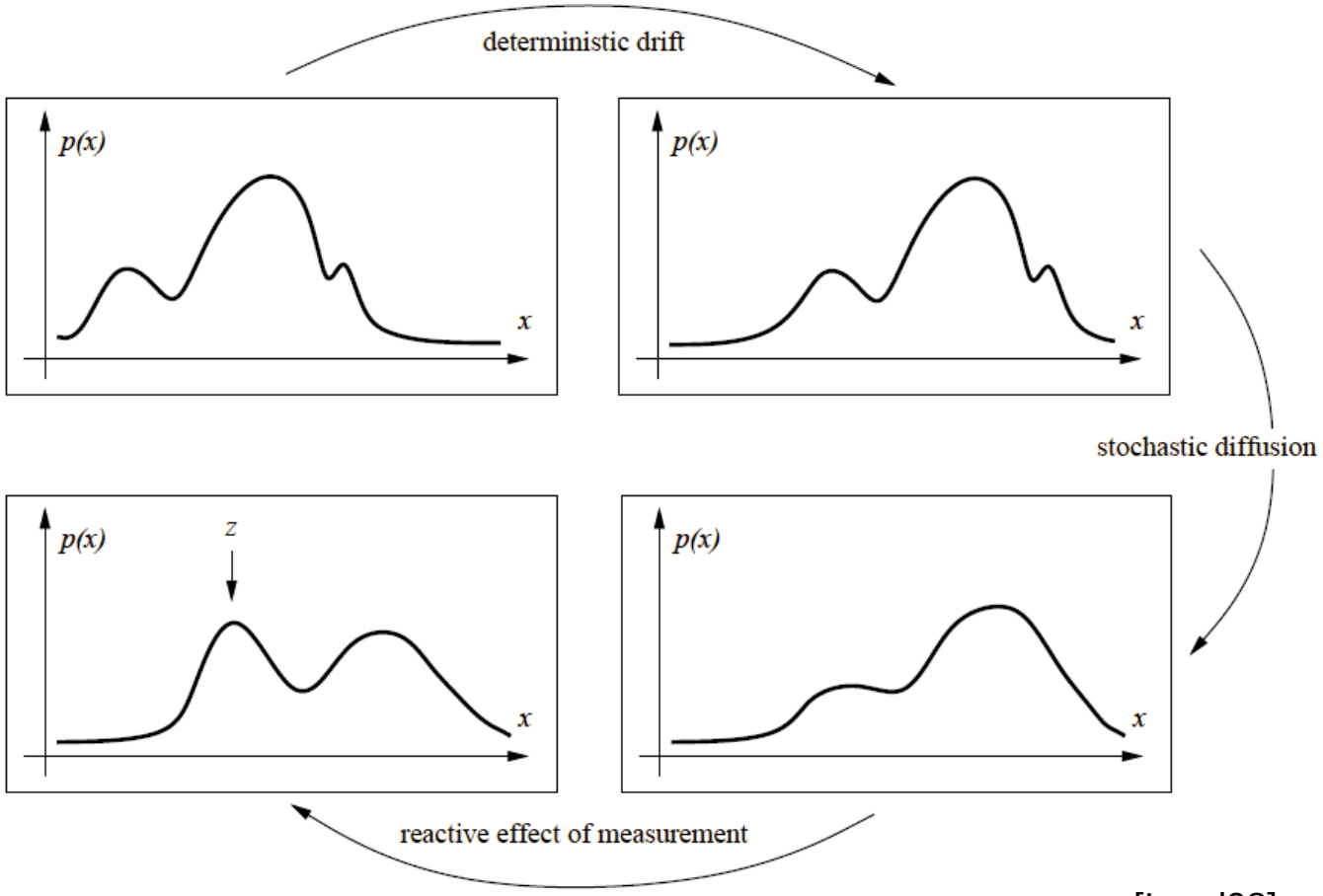
- The Kalman Filter often fails when the measurement density is *multimodal / non-Gaussian.*
- A Particle Filter represents and propagates arbitrary probability distributions. They are represented by a *set of weighted samples*.
- The Particle Filtering is a *numerical* technique (unlike the Kalman filter which is analytical).
- Like a Kalman Filter, a Particle Filter incorporates a *dynamic model* describing system dynamics

# Kalman Filter as Density Propagation



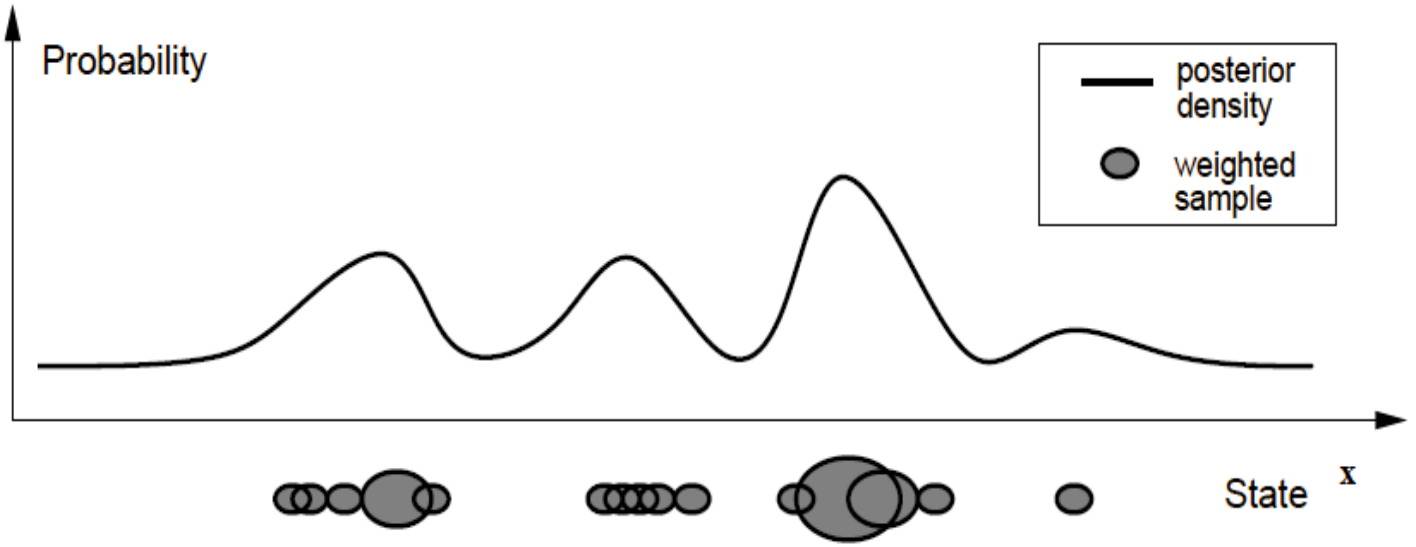

[Isard98]

# Probability Density Propagation




[Isard98]

# Factored Sampling




[Illustration: Isard98]

# Particle Filter

For a PF tracker, you need

- a set of ***N*** weighted samples (particle) at time ***k***

$$\left\{\left(s_k^{(i)}, \pi_k^{(i)}\right) \middle| \, i = 1..N\right\}$$

- the motion model

$$s_k^{(i)} \leftarrow s_{k-1}^{(i)}$$

- the observation model

$$\pi_k^{(i)} \leftarrow s_k^{(i)}$$

# The Condensation Algorithm

*A popular instance of a particle filter, often used in Computer Vision (Isard & Blake 1998)*

1. ***Select***

   Randomly select $N$ new samples $s_k^{(i)}$ from the old sample set $s_{k-1}^{(i)}$ according to their weights

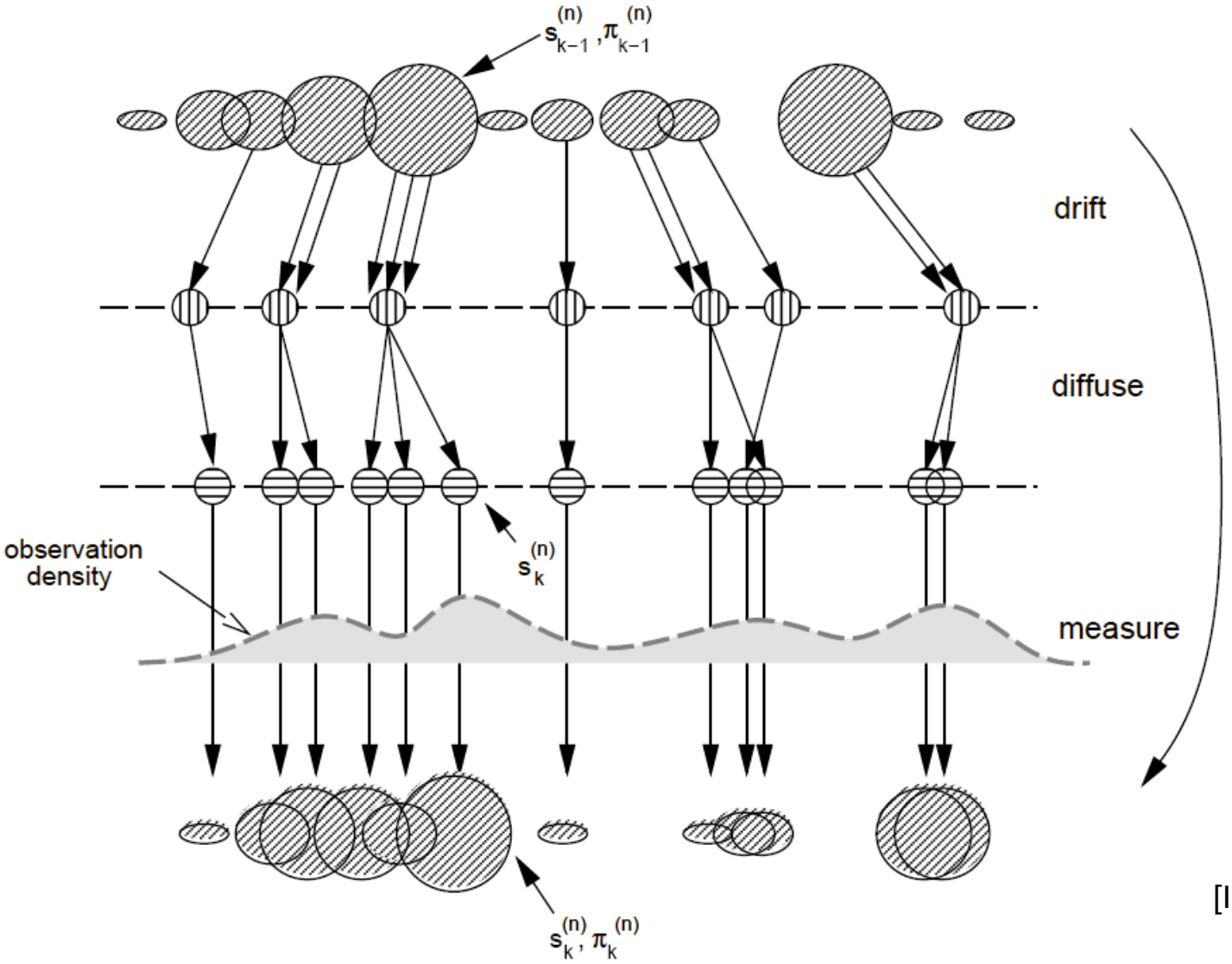
2. ***Predict***

   Propagate the new samples using the motion model

3. ***Measure***

   Calculate weights for the new samples using the observation model $\pi_k^{(i)} = p\left(z_k \mid x_k = s_k^{(i)}\right)$

# The Condensation Algorithm



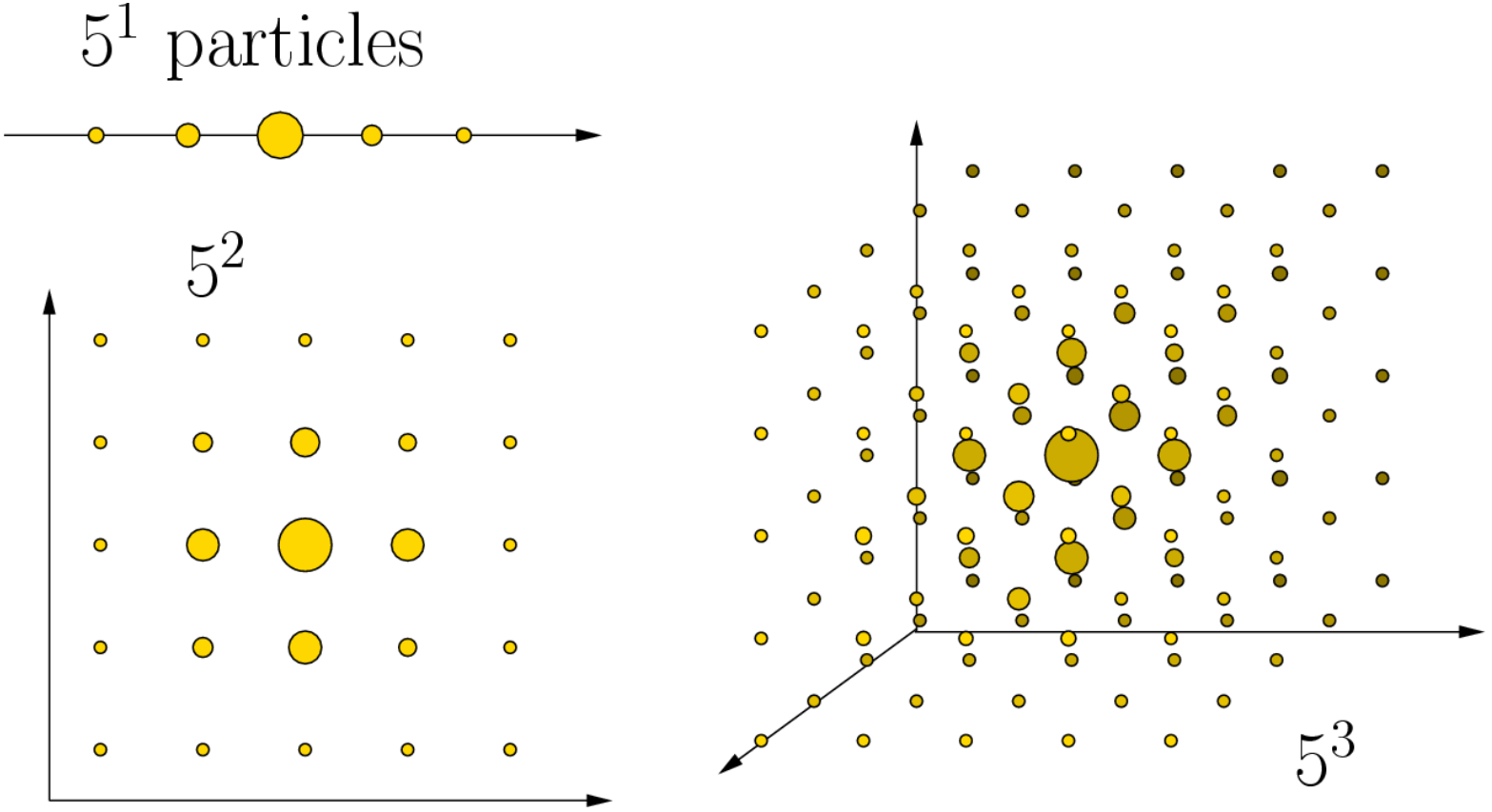

[Isard98]

# The Condensation Algorithm

- The PF tracks a probability distribution – how to get the target position?
  - Calculate the weighted mean of the particle set (sample set)
  - Cluster the particle set and search for the highest mode
  - Just take the strongest particle

- How many particles are needed?
  - *N* depends strongly on the dimension of the state space!
  - Tracking 1 object in the image plane typically requires 50-500 particles

# The Dimensionality Problem



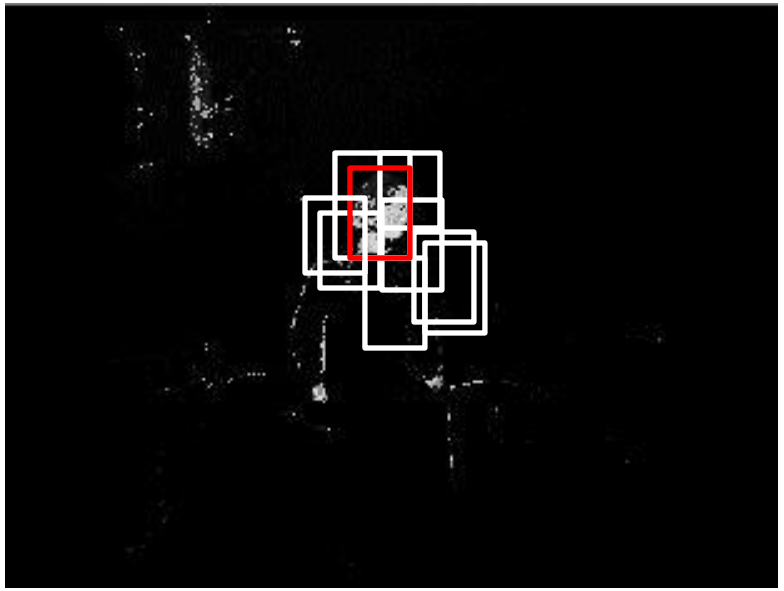
# Examples

# Tracking one Face with a Particle Filter



- State: (x,y,scale)
- Observations:
  - skin color

1) Select and predict samples



2) Measurement step:

- For each particle
  - Count supporting skin pixels in box defined by (x,y,scale)
  - Particle weights determined based on skin color support
- Particle with maximum weight choosen as best solution

# Face and Head Pose Tracking

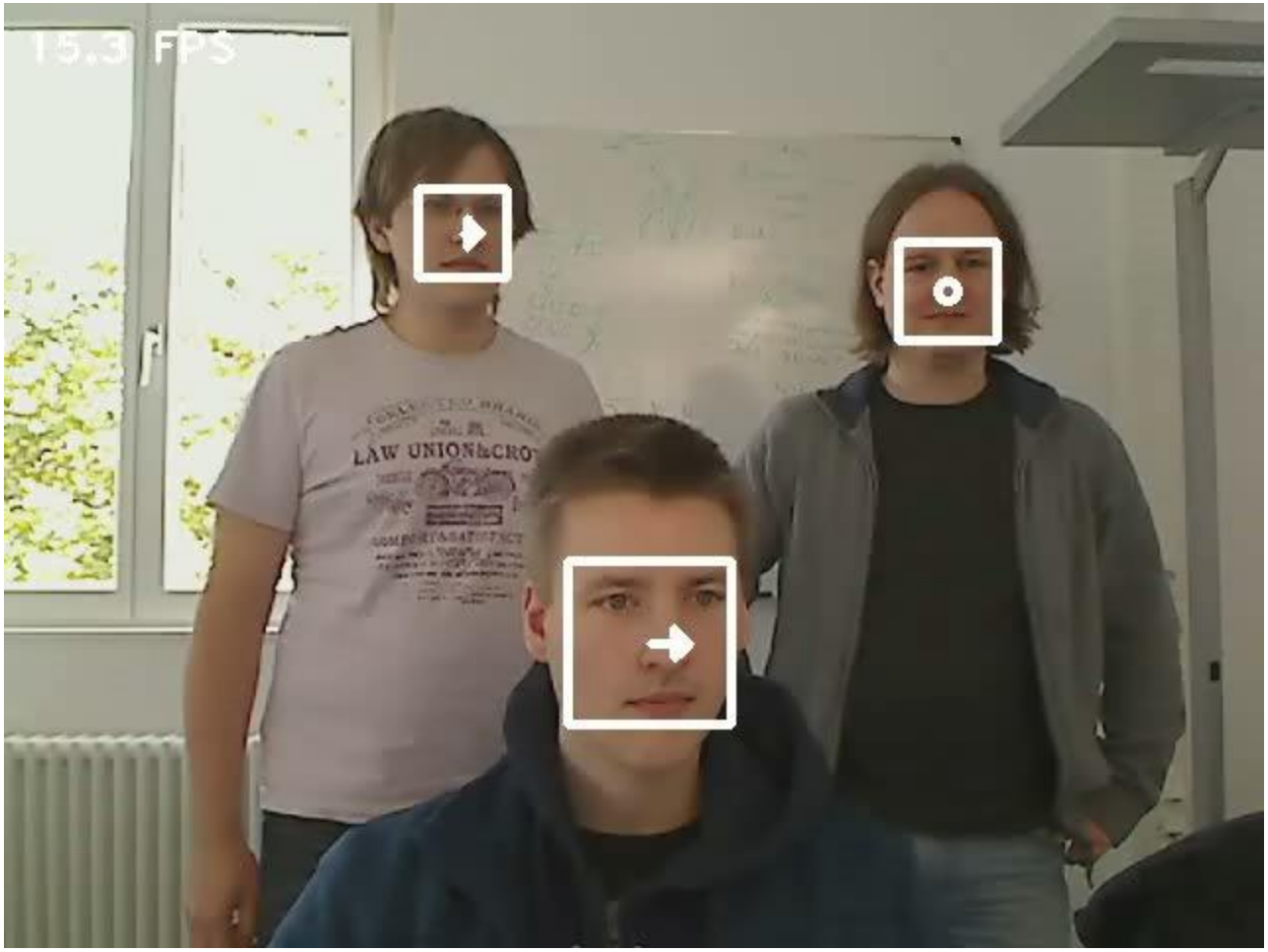
- Particle filter
  - Head-pose estimation integrated in the tracker

$$\boldsymbol{x} = \{x, y, s, \alpha\}$$

yaw angle

- Observation model
  - Use bank of face detectors for different poses
    - frontal, +/-15, +/-30, +/-45, +/-60
  - Update particle weights with score of matching detector, i.e. the detector with closest angle to hypothesis
- Dynamical model
  - Gaussian noise, no explicit velocity model
- Occlusion handling
  - Set particle weight to zero, if it is too close to another track's center

# Example video for head pose estimation




Computer Vision for Human-Computer Interaction
Research Group, Universität Karlsruhe (TH)
cv:hci

# Face and Pose Tracking in Camera Networks


Computer Vision for Human-Computer Interaction
Research Group, Universität Karlsruhe (TH)
cv:hci

# Summary

- Categorization: tracking scheme, features, sensor setup, target type
- Template matching
- Color models: parametric vs. non-parametric, back projection vs. histogram distance
- Background models: Gaussian mixture models, adaptation
- Meanshift
- Kalman Filter: transition and measurement matrix, process and measurement noise
- Particle Filter: motion model, observation model, Condensation algorithm, Particle vs. Kalman filter

# References

Kalman Filter Homepage

http://www.cs.unc.edu/~welch/kalman/

M. Isard and A. Blake, *Condensation – conditional density propagation for visual tracking*, International Journal of Computer Vision 29(1), pp. 5–28, 1998.

# A More Sophisticated Background Adaptation Scheme

- Motivation: some parts of the background exhibit higher variance than others → using the same classification threshold throughout the image is not optimal

- Idea: represent each pixel in the background model by a *mixture of Gaussians* in order to preserve the individual characteristics.

  [Stauffer, Grimson. *Adaptive Background Mixture Models for Real-time Tracking*. CVPR'98.]

# A More Sophisticated Background Adaptation Scheme

The probability of observing the current pixel value $X_t$ is represented by a mixture of $K$ Gaussians

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)}$$

# A More Sophisticated Background Adaptation Scheme

Updating procedure:

For each pixel, its value $X_t$ is checked against all $K$ distributions:

- If $X_t$ lies within a range of 2.5 standard deviations around one of the means $\mu_{i,t}$, then the parameters $\mu$ and $\Sigma$ of the matching distribution are updated using $X_t$. The mixture weights of all distributions are updated using a learning factor $\alpha$:

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$

($M_{k,t}$ = 1 for the matching distribution, zero otherwise)

- If none of the distributions matches $X_t$, then a new Gaussian distribution is created for $X_t$. It replaces the least probable distribution.

# A More Sophisticated Background Adaptation Scheme

Classification:

1. Determine the distributions which are most likely produced by the background process:
   - Distributions with a high weight and low variance are considered to be background distributions → order the list of distributions by the value of $\omega/\sigma$.
   - From the ordered list, choose the first $B$ distributions such that:

$$B = argmin_b \left( \sum_{k=1}^{b} \omega_k > T \right)$$

2. If a pixel's value $X_t$ matches none of the $B$ selected distributions, the pixel is classified as foreground

# A More Sophisticated Background Adaptation Scheme

Camera image

Learned background

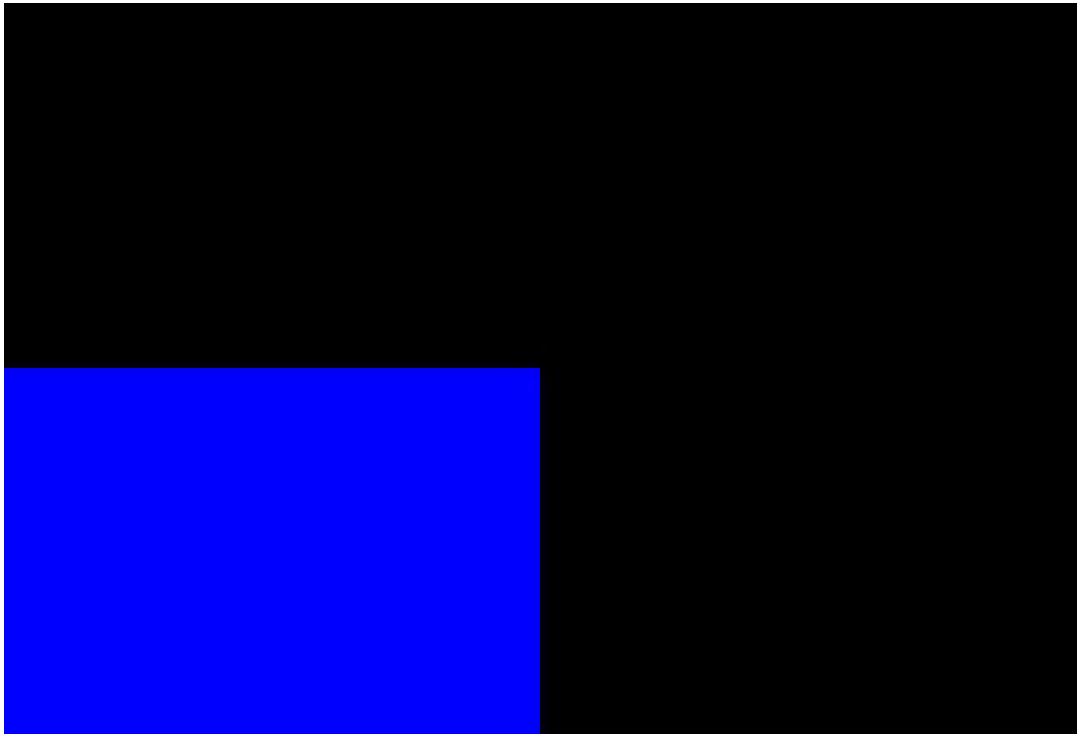
Foreground pixels

Tracking



Computer Vision for Human-Computer Interaction
Research Group, Universität Karlsruhe (TH)
cv:hci

# Background Subtraction



Video

Video: IBM Research
http://www.research.ibm.com/peoplevision/bgs.html

# References

Stauffer, Grimson. *Adaptive Background Mixture Models for Real-time Tracking*. CVPR'98