Univerza v Ljubljani

# Advanced computer vision methods
# Tracking by Recursive Bayes Filters
# Part III: Particle filters

Matej Kristan

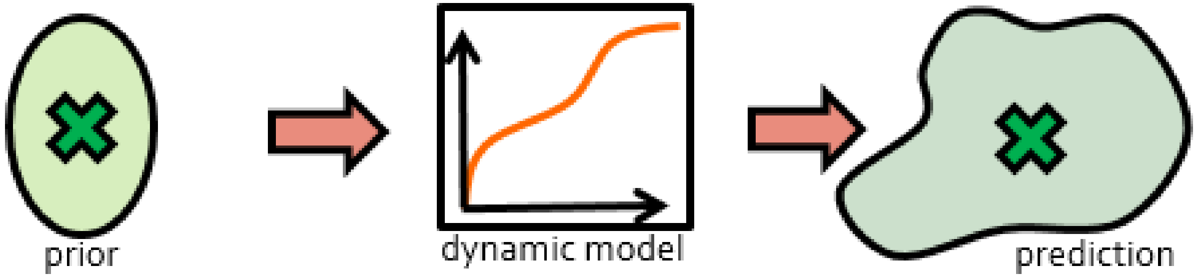
Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

# Previously at ACVM…

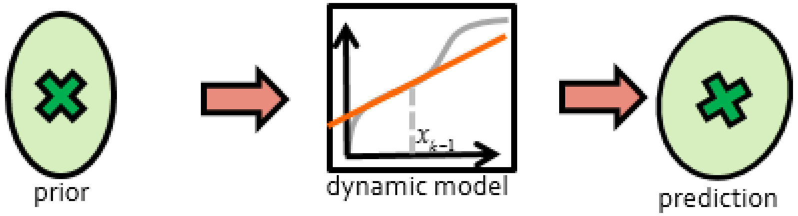
- Everything is Gaussian and linear → Kalman filter

$$\underbrace{p(\mathbf{x}_k|\mathbf{y}_{1:k})}_{\text{posterior estimate}} \propto \underbrace{p(\mathbf{y}_k|\mathbf{x}_k)}_{\text{Observation model}} \int \underbrace{p(\mathbf{x}_k|\mathbf{x}_{k-1})}_{\text{motion model}}\underbrace{p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})}_{\text{posterior at k-1}} d\mathbf{x}_{k-1}$$

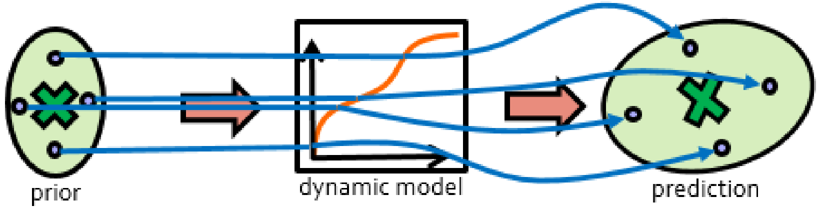
- But what if dynamics are non-linear?




Extended Kalman filter:



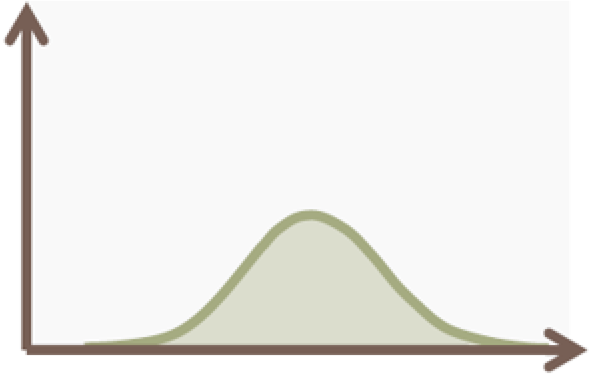

Unscented Kalman filter:



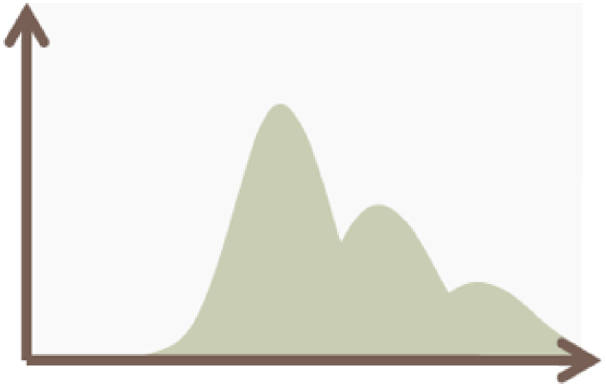
# Beyond the basic Kalman

Drawbacks of Kalman filter:

- Assumes a Gaussian posterior and a linear Gaussian dynamic model



vs



Numerical approaches:

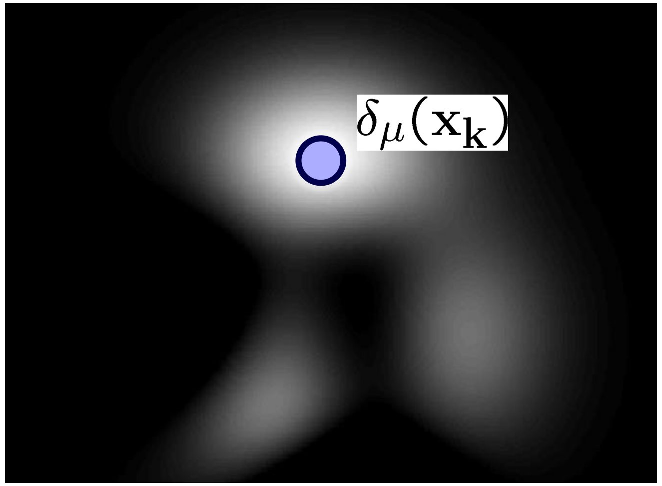
- Grid-based methods – discretize the posteriors
- The mid 80's has seen a rise of Monte Carlo approximation of the recursive Bayes filter
- Eventually got known under the common name: *Particle filters*

# Analytic representations of posteriors

- A single point : Dirac-delta

current input image


current posterior




$p(\mathbf{x}_k|\mathbf{y}_{1:k})$

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \begin{cases} inf & \text{if } \mathbf{x_k} = \mu \\ 0 & \text{otherwise} \end{cases}$$

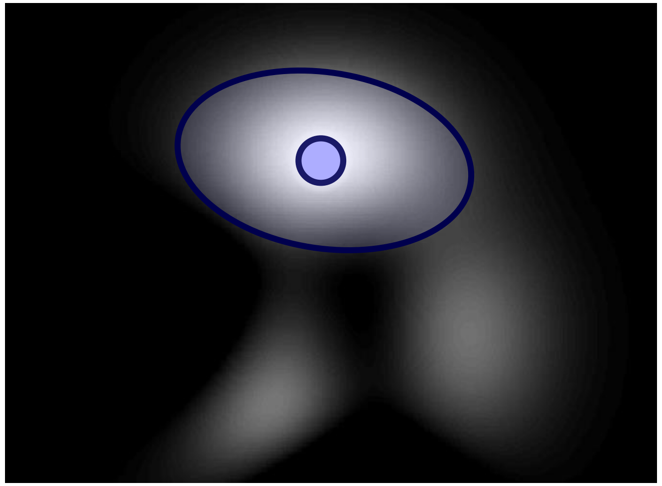
$$\int_{-\infty}^{\infty} p(\mathbf{x}_k|\mathbf{y}_{1:k}) d\mathbf{x}_k = 1$$

# Analytic representations of posteriors

- A single point + covariance: a Gaussian distribution

current input image

current posterior



$p(\mathbf{x}_k|\mathbf{y}_{1:k})$

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \mathcal{N}(\mathbf{x}|\mu;\boldsymbol{\Sigma})$$

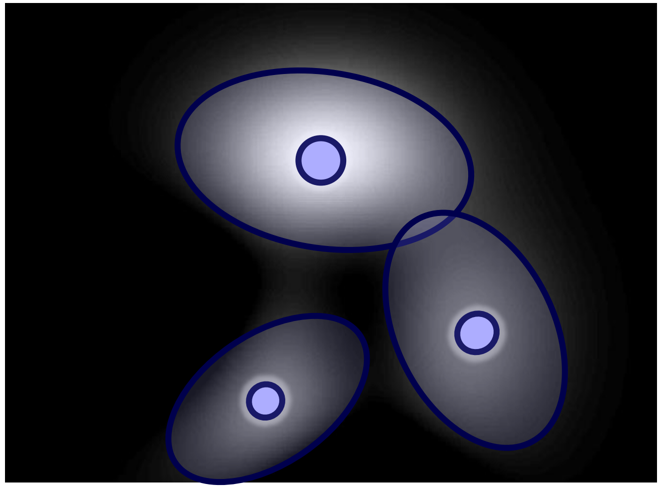
$$\mathcal{N}(\mathbf{x}|\mu;\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}_k-\mu)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_k-\mu)}$$

# Analytic representations of posteriors

- A weighted point + covariance: a Gaussian mixture

current input image

current posterior



$p(\mathbf{x}_k|\mathbf{y}_{1:k})$

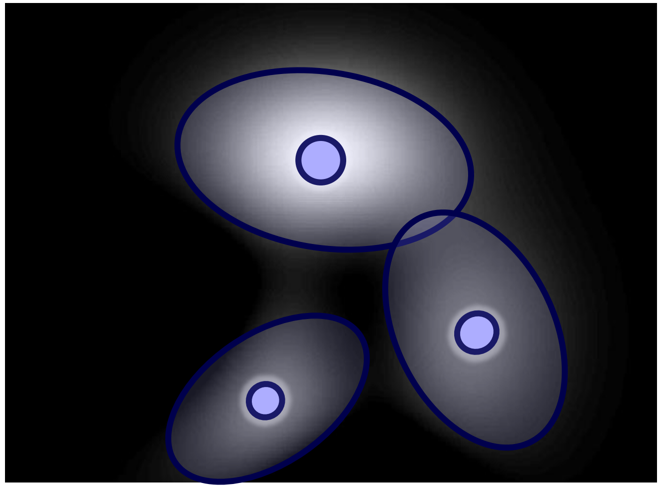
$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \sum_{i=1}^{N} w_i \mathcal{N}(\mathbf{x}|\mu_i; \mathbf{\Sigma}_i)$$

# Analytic representations of posteriors

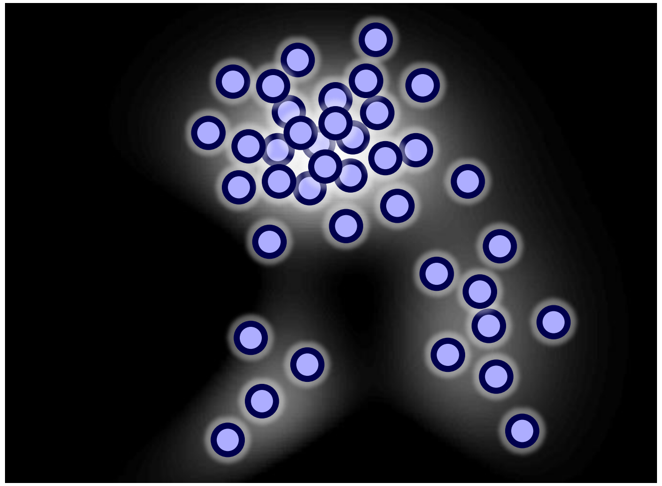
- A set of samples: $\{\mathbf{x}_k^{(i)}\}_{i=1:N}$

current input image

current posterior



$p(\mathbf{x}_k|\mathbf{y}_{1:k})$



Before we continue:

*How to sample from a mixture model?*

# The remainder of this lecture

1. Sampling from a mixture model

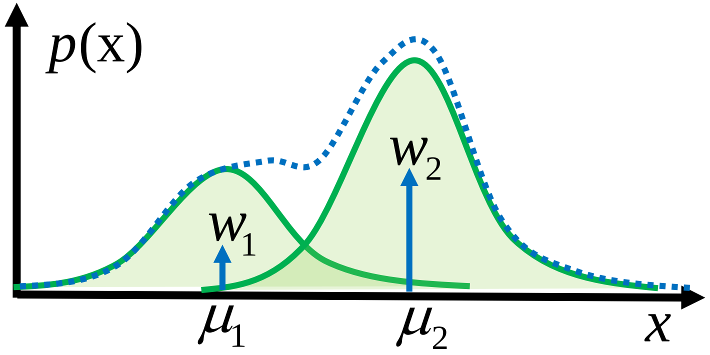
2. A gentle introduction to Monte Carlo integration

3. Application to the Bayes recursive filter equation

# Generating samples from a mixture

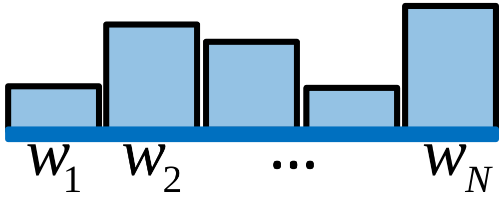
- Let us consider a mixture of two Gaussians (N=2):

$$p(\mathrm{x}) = \sum_{i=1}^{N} w_i N(\mathrm{x}; \mu_i, \Sigma_i)$$




- Weights form a discrete pdf over the components (N=2):

$$p(\mathrm{i}) = \{w_i\}_{i=1:N}$$

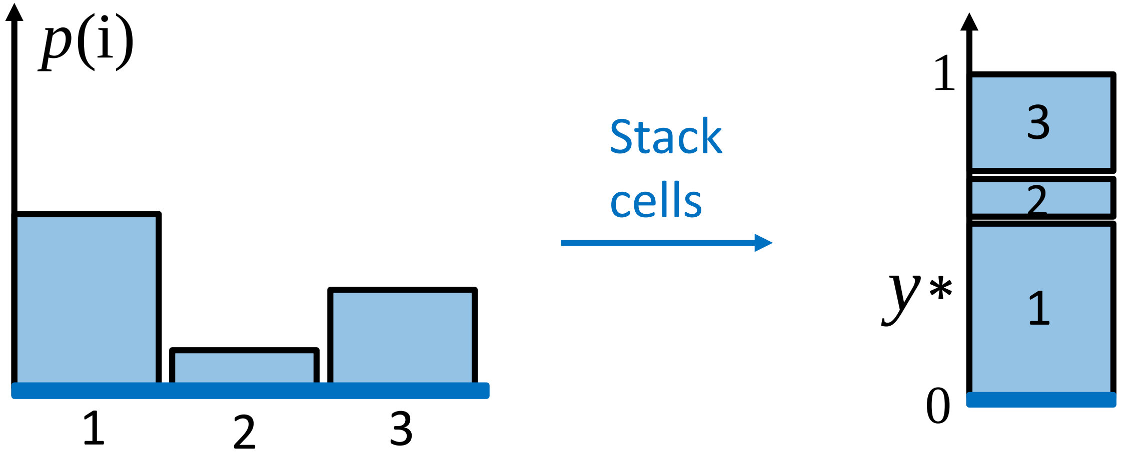



- Samples are generated in two stages:
  1. Sample identity of a component ($i \sim \mathrm{p(i)}$)
  2. Sample a point from *i*-th component ($x \sim N(x; \mu_i, \Sigma_i)$)

# Sampling from a discrete pdf

- Consider a discrete pdf $p(i)$ with domain $\{1,2,3\}$

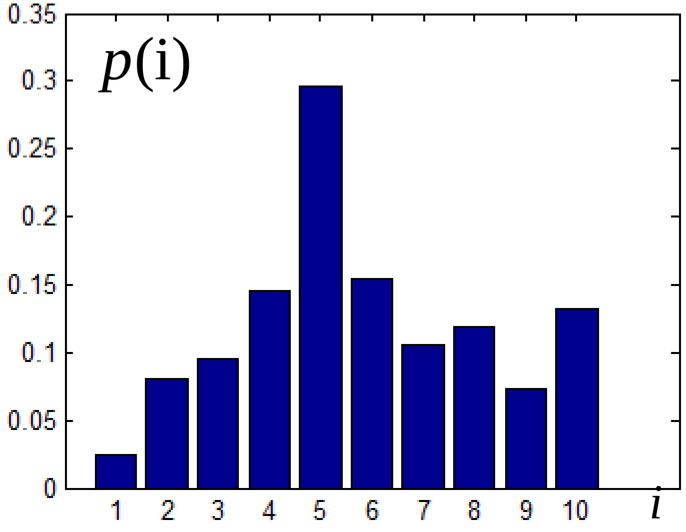
$$p(\mathrm{i}) = \begin{cases} 0.6 & i=1 \\ 0.1 & i=2 \\ 0.3 & i=3 \end{cases}$$

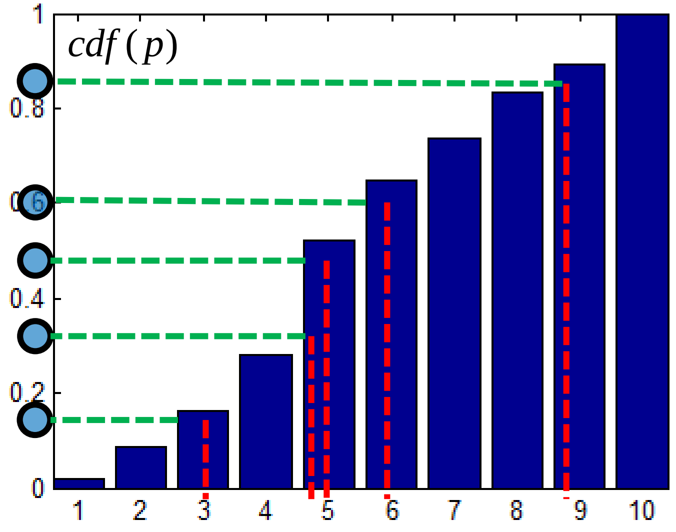



- Draw a number $y$ uniformly from (real) interval [0,1]
- The probability that $y$ will fall within the interval occupied by cell 1 equals to the probability 0.6!
- Thus uniform samplers may be used (e.g. rand())!

# Sampling from a discrete pdf

- To draw N samples from $p(i)$:
  - Calculate the cumulative pdf (Matlab: cumsum(p))
  - Draw N numbers $y_j$ uniformly from [0,1] (Matlab: rand(N,1))
  - For each $y_j$ determine the corresponding index $i$.

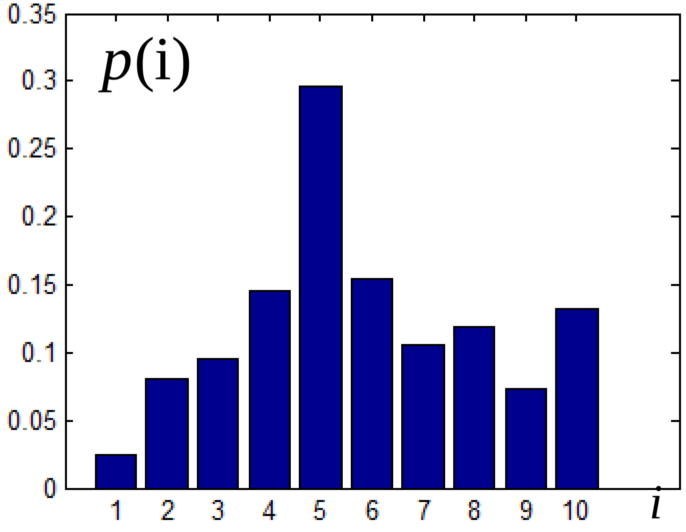


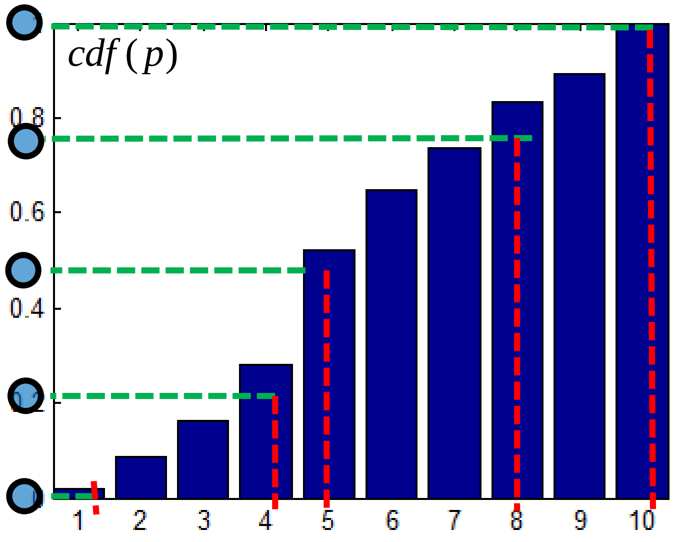




For example: we have generated $\{3,5,5,6,9\}$.

# Sampling from a discrete pdf

- “Deterministic” sampling: *(preferred in some applications)*
  - Calculate the cumulative pdf (Matlab: cumsum(p))
  - Spread N numbers $y_j$ evenly from [0,1] (Matlab: 0:1/(N-1):1)
  - For each $y_j$ determine the corresponding index $i$.

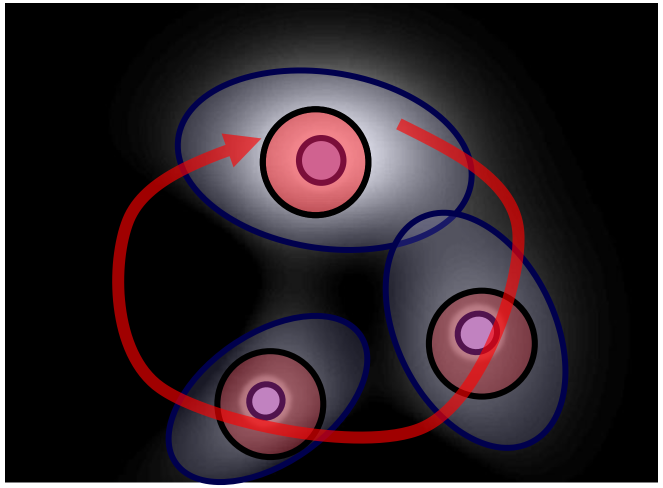


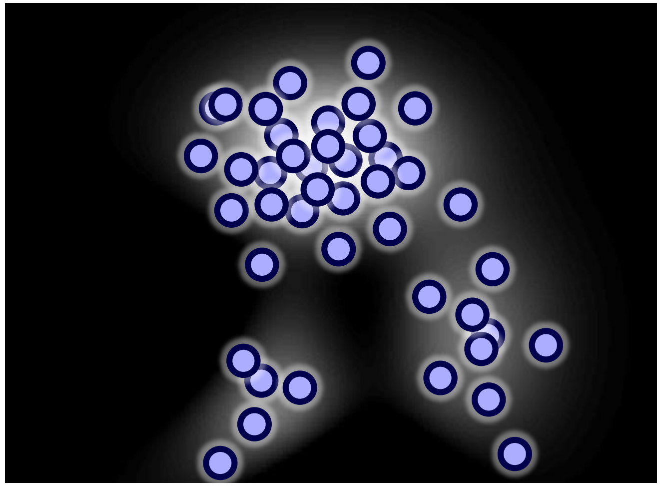

For example: we have generated $\{1,4,5,8,10\}$.

# Sampling example

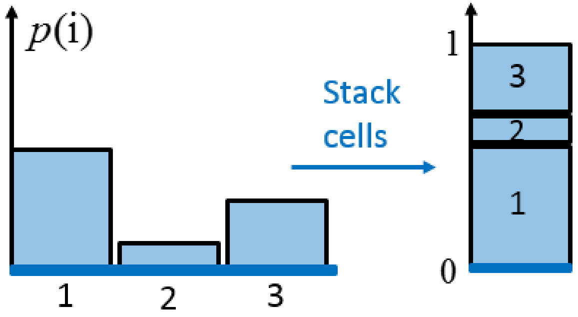
Some mixture model



Sampled points



- Samples are generated in two stages:
    1. Sample identity of a component ($i \sim \mathrm{p(i)}$)

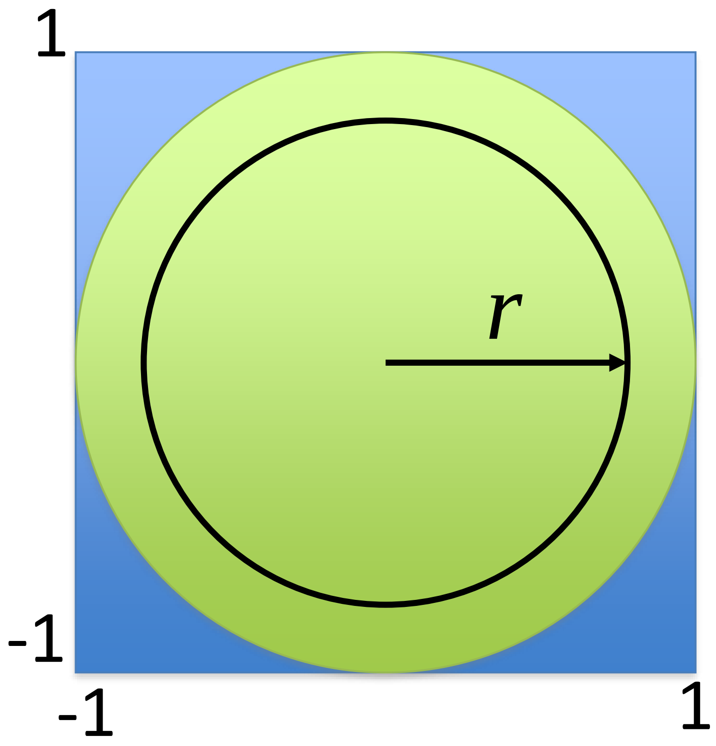



i=1 ,i=2 ,i=1 ,i=3 ,....

2. Sample a point from *i*-th component ($x \sim N(x; \mu_i, \Sigma_i)$)

# Monte Carlo integration

- The Recursive Bayes Filter is mainly about integration.
- What is the area of the unit circle?




Calculus gives:

$$A = \int_0^1 2\pi r dr$$

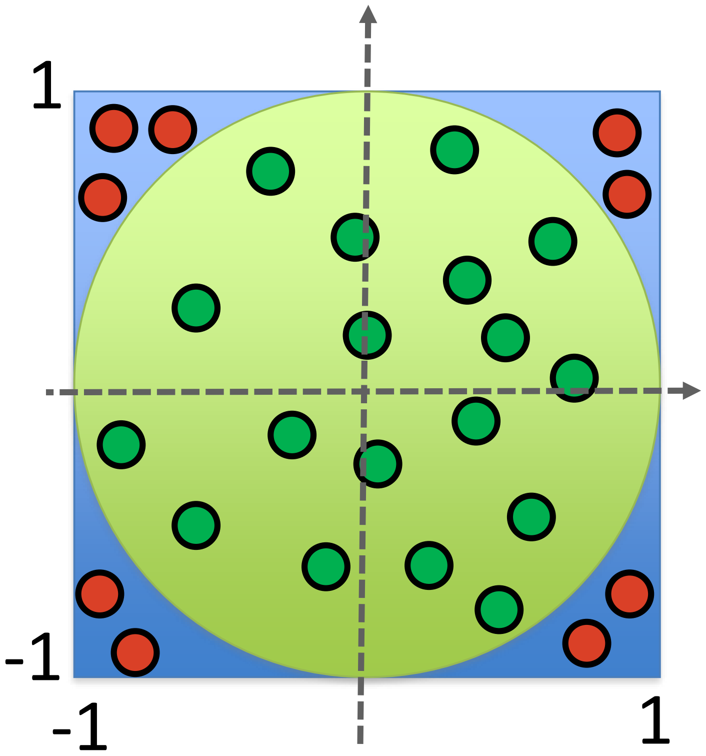
*What if you wanted to avoid calculus?*

# Apply sampling

- Sample $x$ and $y$ uniformly from interval [-1,1]
  $(x, y) \sim U(x, y)$ … a uniform distribution
- Check if a sample is within a circle:

$$c(\mathrm{x},\mathrm{y}) = \begin{cases} 1 & ; x^2 + y^2 \le 1 \\ 0 & ; \text{otherwise} \end{cases}$$

- Repeat N times
- Count the proportion of samples that fall into the circle $\alpha = N_{in}/N$.
- Multiply by the area of the square: $A \approx 4 \cdot N_{in}/N$

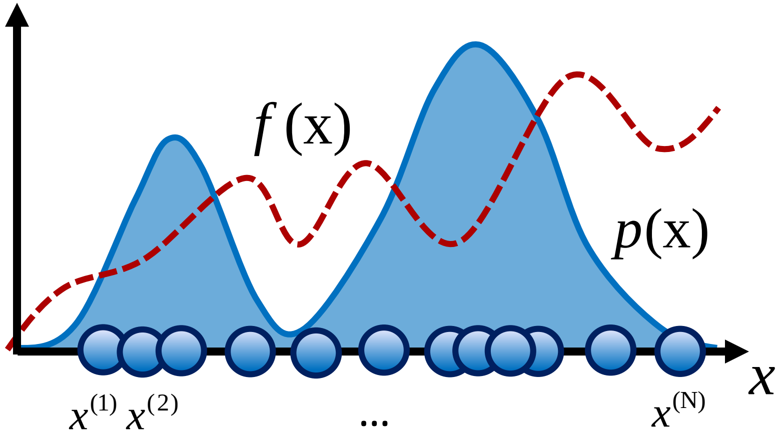



$$\int_{-1}^{1} U(\mathrm{x},\mathrm{y})\, \mathrm{c}(\mathrm{x},\mathrm{y})\, \mathrm{dxdy}$$

# Monte Carlo integration

- The expected value:

$$I = \int f(\mathrm{x})\,\mathrm{p}(\mathrm{x})\,\mathrm{dx} = \langle f(\mathrm{x}) \rangle_{p(\mathrm{x})}$$




Approximate $p(x)$:

- Generate N samples from $p(x)$:

$$x^{(\mathrm{i})} \sim p(\mathrm{x}) \longrightarrow \left\{x^{(\mathrm{i})}\right\}_{i=1:N}$$

- A Monte Carlo approximation of a pdf:

$$p(\mathrm{x}) \approx \sum_{i=1}^{N} \frac{1}{N} \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

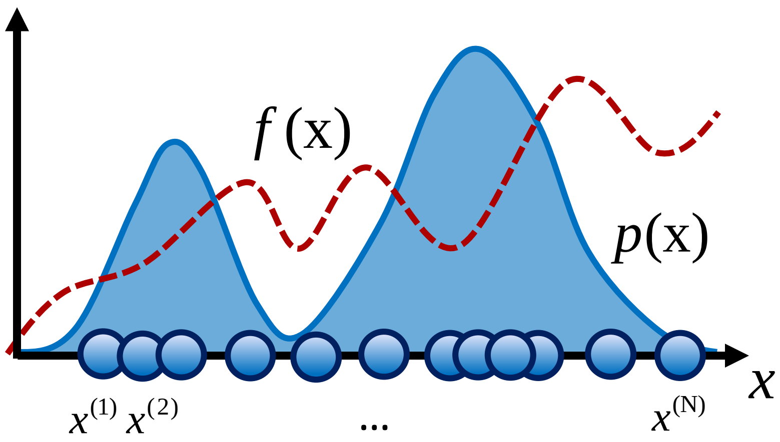
Dirac-delta centered at $x^{(i)}$

# Monte Carlo integration

- A MC approximation of a pdf:

$$p_N(\mathrm{x}) = \sum_{i=1}^{N} \frac{1}{N} \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

$$p(\mathrm{x}) \approx p_N(x)$$

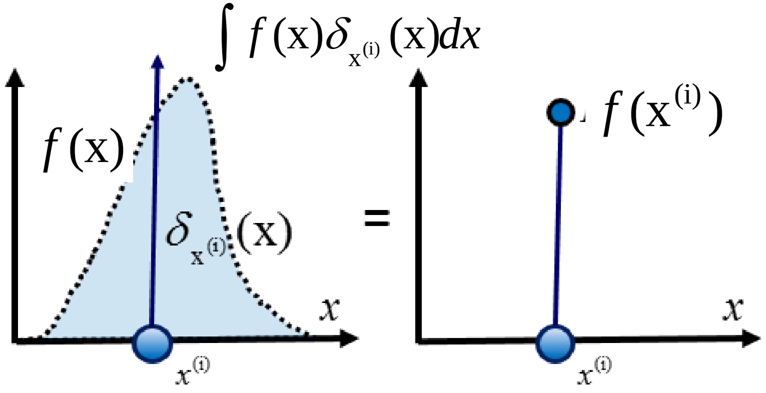



- A MC approximation of the integral:

$$I = \int f(\mathrm{x})\,\mathrm{p}(\mathrm{x})\,\mathrm{dx}$$

$$I_N = \int f(\mathrm{x}) \left[ \sum_{i=1}^{N} \frac{1}{N} \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x}) \right] \mathrm{dx}$$

$$= \sum_{i=1}^{N} \frac{1}{N} \int f(\mathrm{x}) \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})\,\mathrm{dx} = \sum_{i=1}^{N} \frac{1}{N} f(\mathrm{x}^{(\mathrm{i})})$$



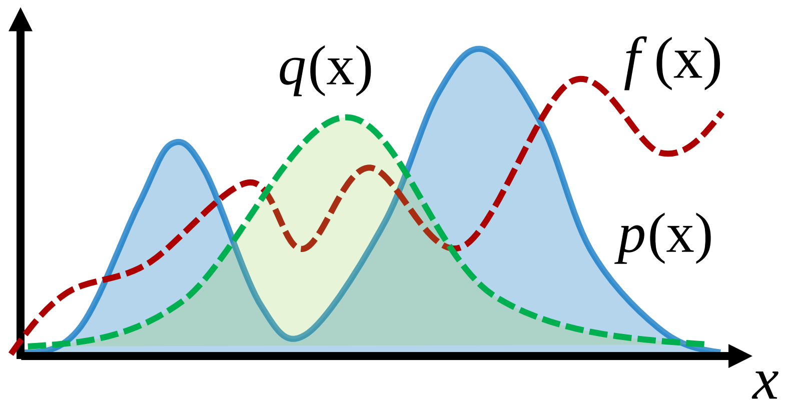

$$\lim_{N \to \infty} I_N(\mathrm{f}) = I(\mathrm{f})$$

"with probability one"

# Monte Carlo integration

- Sampling from $p(x)$ may be difficult, but we can evaluate it up to a prop. constant:

  $$\tilde{p}(\mathrm{x}) = \mathrm{C}\mathrm{p}(\mathrm{x})$$




- Let $q(x)$ be another pdf that is easy to sample from
- The integral can be rewritten into

$$I = \int f(\mathrm{x})\,\mathrm{p}(\mathrm{x})\,\mathrm{dx} = \int f(\mathrm{x}) \frac{1}{C} \tilde{\mathrm{p}}(\mathrm{x}) \frac{q(\mathrm{x})}{q(\mathrm{x})}\,\mathrm{dx}$$

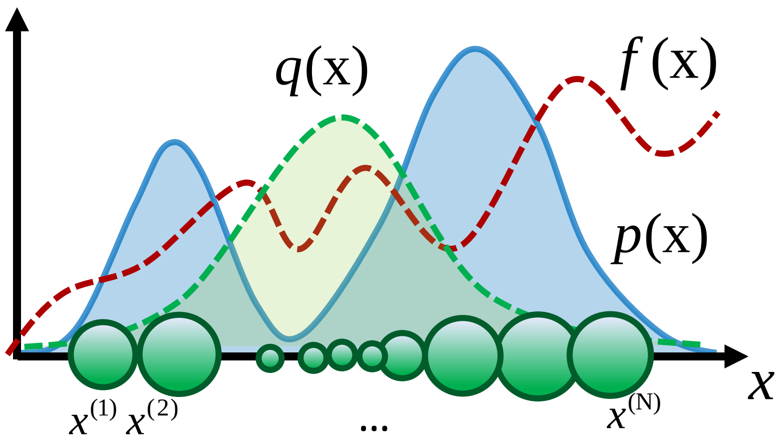
$$= \int f(\mathrm{x})\,\mathrm{q}(\mathrm{x}) \frac{1}{C} \tilde{\mathrm{w}}(\mathrm{x})\,\mathrm{dx} \qquad , \qquad \tilde{w}(\mathrm{x}) = \frac{\tilde{p}(\mathrm{x})}{q(\mathrm{x})}$$

# Monte Carlo integration

- Now sample from $q(x)$…

$$q(\mathrm{x}) \approx \sum_{i=1}^{N} \frac{1}{N} \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

- And the integral becomes:




$$I = \int f(\mathrm{x})\, p(\mathrm{x})\, \mathrm{dx} = \int f(\mathrm{x}) q(\mathrm{x}) \frac{1}{C} \tilde{\mathrm{w}}(\mathrm{x})\, \mathrm{dx} \quad , \quad \tilde{w}(\mathrm{x}) = \frac{\tilde{p}(\mathrm{x})}{q(\mathrm{x})}$$

$$= \frac{1}{N} \sum_{i} f(\mathrm{x}^{(\mathrm{i})}) \frac{1}{C} \tilde{\mathrm{w}}(\mathrm{x}^{(\mathrm{i})})$$

$$= \sum_{i} f(\mathrm{x}^{(\mathrm{i})})\, \mathrm{w}^{(\mathrm{i})}$$

$$\mathrm{w}^{(\mathrm{i})} = \frac{\tilde{w}(\mathrm{x}^{(i)})}{\sum_{i=1}^{N} \tilde{w}(\mathrm{x}^{(i)})}$$

$$C = ?$$

$$\int \tilde{p}(\mathrm{x}) = \int \mathrm{Cp}(\mathrm{x})$$

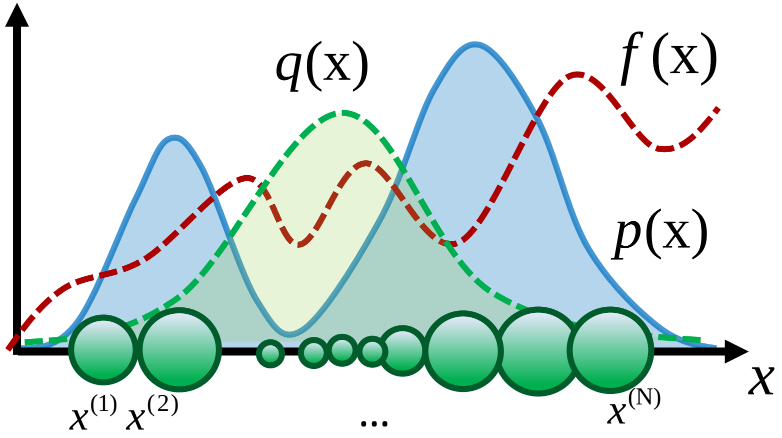
$$\Downarrow$$

$$C = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}(\mathrm{x}^{(i)})$$

# Monte Carlo integration

- So to approximate an integral over a complicated function $f(x)$:

$$I = \int f(\mathrm{x})\, p(\mathrm{x})\, \mathrm{dx}$$




- Sample from $q(x)$,
- Calculate weights at samples:

$$\tilde{w}(\mathrm{x}^{(\mathrm{i})}) = \frac{\tilde{p}(\mathrm{x}^{(\mathrm{i})})}{q(\mathrm{x}^{(\mathrm{i})})}$$

- Normalize the weights: $\mathrm{w}^{(\mathrm{i})} = \dfrac{\tilde{w}(\mathrm{x}^{(i)})}{\sum_{i=1}^{N} \tilde{w}(\mathrm{x}^{(i)})}$

- Calculate the weighted average: $I \approx \sum_i f(\mathrm{x}^{(\mathrm{i})})\, w(\mathrm{x}^{(\mathrm{i})})$

# Monte Carlo pdf representation

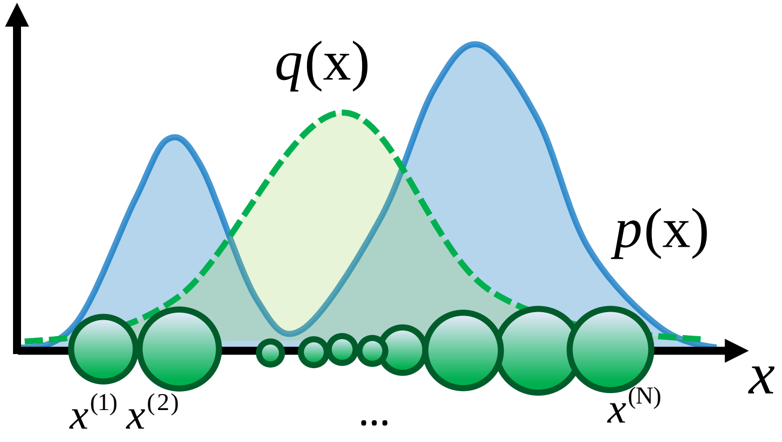
- Represent $p(x)$ by a weighted set of samples:

$$p(\mathrm{x}) \approx \sum_{i=1}^{N} \mathrm{w}^{(\mathrm{i})}\, \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

- ...by sampling from $q(x)$
- and calculate the correction weights:

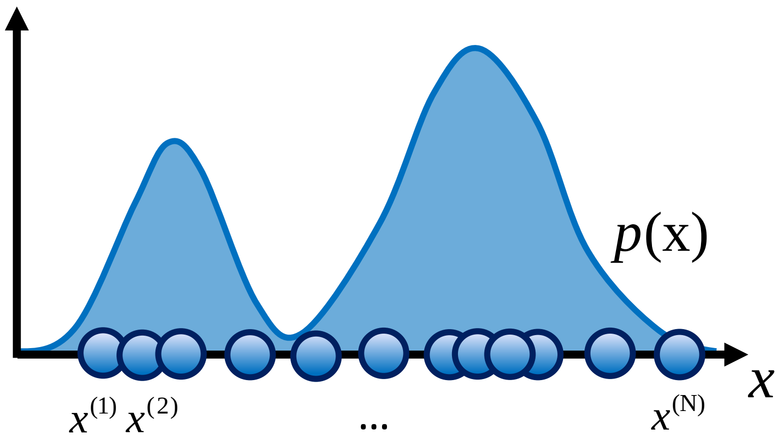
$$\mathrm{w}^{(\mathrm{i})} = \frac{\tilde{w}(\mathrm{x}^{(i)})}{\sum_{i=1}^{N} \tilde{w}(\mathrm{x}^{(i)})}$$

$$\tilde{w}(\mathrm{x}^{(\mathrm{i})}) = \frac{p(\mathrm{x}^{(\mathrm{i})})}{q(\mathrm{x}^{(\mathrm{i})})}$$



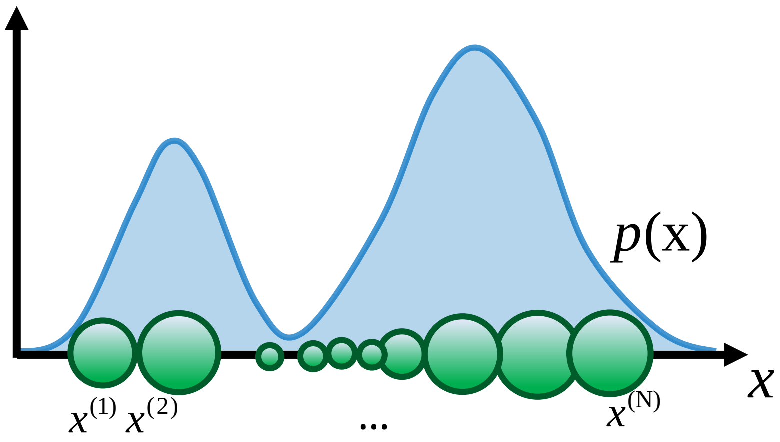
# Monte Carlo pdf representation

- Note the different (but equivalent) Monte Carlo representations of the same pdf:




$$p(\mathrm{x}) \approx \sum_{i=1}^{N} \frac{1}{N} \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

$$\left\{ \mathrm{x}^{(\mathrm{i})}, \frac{1}{N} \right\}_{i=1:N}$$




$$p(\mathrm{x}) \approx \sum_{i=1}^{N} \mathrm{w}^{(\mathrm{i})} \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

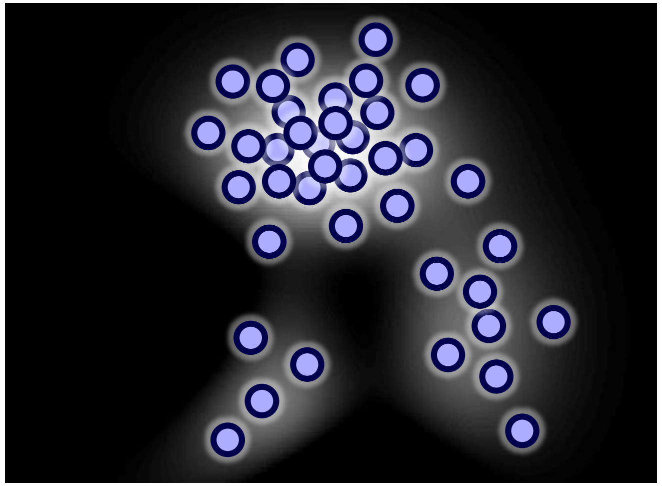
$$\left\{ \mathrm{x}^{(\mathrm{i})}, w^{(\mathrm{i})} \right\}_{i=1:N}$$

# Analytic representations of posteriors

- Monte Carlo samples: a mixture of weighted Dirac deltas

current input image

current posterior

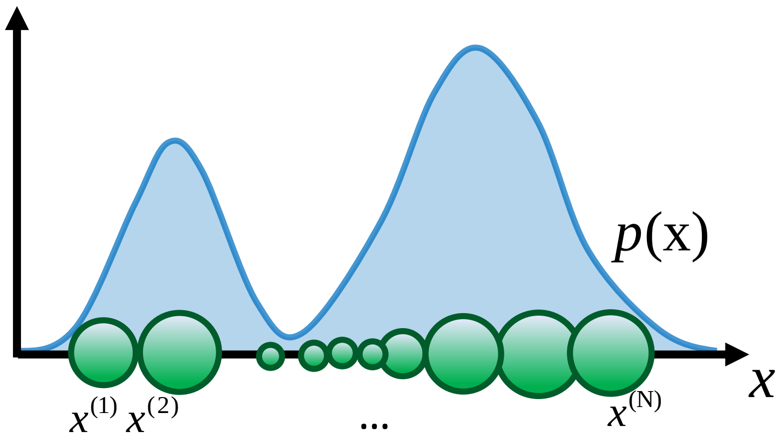


$p(\mathbf{x}_k|\mathbf{y}_{1:k})$

$$p(\mathrm{x}) \approx \sum_{i=1}^{N} \mathrm{w}^{(\mathrm{i})}\, \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

# Computing expectations by MC

- Assume we have a MC representation of $p(x)$




$$p(\mathrm{x}) \approx \sum_{i=1}^{N} \mathrm{w}^{(\mathrm{i})}\, \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})$$

$$\left\{\mathrm{x}^{(\mathrm{i})}, w^{(\mathrm{i})}\right\}_{i=1:N} \longleftarrow$$ This set is called “particles”

- What is the expected value of $x$ under $p(x)$?

$$\langle x \rangle_{p(\mathrm{x})} = \int x p(\mathrm{x})\, \mathrm{dx} = \int x \left[\sum_{i=1}^{N} \mathrm{w}^{(\mathrm{i})}\, \delta_{\mathrm{x}^{(\mathrm{i})}}(\mathrm{x})\right] \mathrm{dx} = \sum_{i=1}^{N} \mathrm{w}^{(\mathrm{i})} \mathrm{x}^{(\mathrm{i})}$$

# Particle filters

- Recursive Bayes Filters
- Approximate the posterior by weighted samples – particles
- Instead of integration, sample and apply summation
- Originate from fields like:
  - Statistical machine learning and pattern recognition
  - Statistical mechanics
  - Signal processing
  - Econometrics

# Particle filters

- Go by names:
  - Sequential Monte Carlo Methods
  - Sequential Importance Sampling with Resampling
- We will consider the simplest particle filter called "the bootstrap particle filter"
- Despite simplicity, still widely used
- Introduced to computer vision in the paper from Isard and Blake[1].

[1]Isard, M. and Blake, A. "CONDENSATION --conditional density propagation for visual tracking" IJCV, 29, 1, 5--28, (1998)

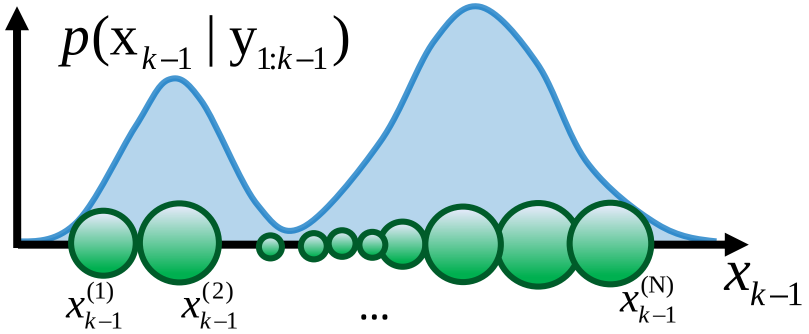
# Particle filters

- Recall the recursive Bayes filter equation

$$p(\mathrm{x}_k \mid y_{1:k}) \propto p(\mathrm{y}_k \mid x_k) \int p(\mathrm{x}_k \mid \mathrm{x}_{k-1})\, \mathrm{p}(\mathrm{x}_{k-1} \mid \mathrm{y}_{1:k-1})\, \mathrm{dx}_{k-1}$$

- Approximate the prior by MC

$$p(\mathrm{x}_{k-1} \mid \mathrm{y}_{1:k-1}) \approx \sum_{i=1}^{N} \mathrm{w}_{k-1}^{(\mathrm{i})} \delta_{\mathrm{x}_{k-1}^{(\mathrm{i})}}(\mathrm{x}_{k-1})$$




- The integral in the posterior vanishes:

$$p(\mathrm{x}_k \mid y_{1:k}) \propto p(\mathrm{y}_k \mid x_k) \sum_{i=1}^{N} \mathrm{w}_{k-1}^{(\mathrm{i})}\, p(\mathrm{x}_k \mid \mathrm{x}_{k-1}^{(\mathrm{i})})$$

But we need to represent the posterior by MC samples as well.
Apply importance sampling!

# Particle filters

- Sample $N$ samples from the proposal: $x_k^{(j)} \sim q(x)$

The posterior distribution becomes:

$$p(\mathrm{x}_k \mid \mathrm{y}_{1:k}) \propto \sum_{j=1}^{N} \mathrm{w}_k^{(j)} \delta_{\mathrm{x}_k^{(j)}}(\mathrm{x}_k) \;, \quad \mathrm{w}_k^{(j)} = \frac{p(\mathrm{y}_k \mid x_k^{(j)}) \sum_{i=1}^{N} \mathrm{w}_{k-1}^{(\mathrm{i})} \, p(\mathrm{x}_k^{(j)} \mid \mathrm{x}_{k-1}^{(\mathrm{i})})}{q(\mathrm{x}_k^{(j)})}$$

- Using $q(\mathrm{x}) = \sum_{i=1}^{N} \mathrm{w}_{k-1}^{(\mathrm{i})} \, p(\mathrm{x}_k \mid \mathrm{x}_{k-1}^{(\mathrm{i})})$ , the weights further simplify into

$$\mathrm{w}_k^{(j)} = p(\mathrm{y}_k \mid x_k^{(j)})$$

- Note that $q(x)$ is a mixture model (dynamic model).
- This is the “Bootstrap particle filter”.

# Bootstrap (SIR) particle filter

Resample:

1. Draw N samples from $$p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) \approx \sum_{i=1}^{N} w_{k-1}^{(i)} \delta_{\mathbf{x}_{k-1}^{(i)}}(\mathbf{x}_{k-1})$$

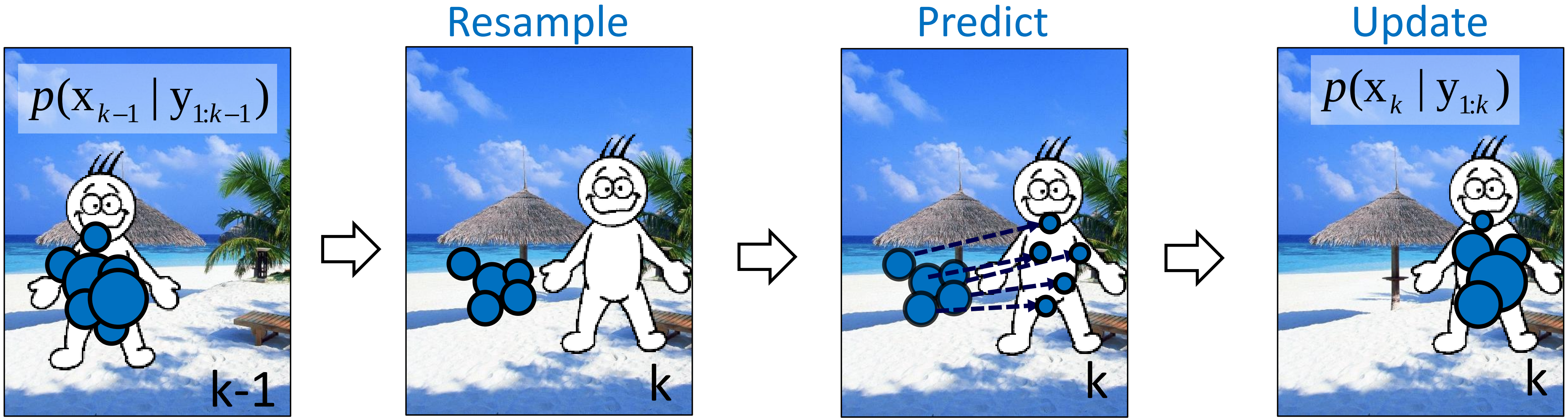
Predict:

2. Move each sample by dynamic model: $$\mathrm{x}_k^{(i)} \sim p(\mathrm{x}_k \mid \mathrm{x}_{k-1}^{(i)})$$

Update:

3. Evaluate the weight of each sample: $$\tilde{\mathrm{w}}_k^{(i)} = p(\mathrm{y}_k \mid x_k^{(i)})$$

- Normalize weights: $$p(\mathrm{x}_k \mid \mathrm{y}_{1:k}) \approx \sum_{i=1}^{N} \mathrm{w}_k^{(\mathrm{i})} \delta_{\mathrm{x}_k^{(\mathrm{i})}}(\mathrm{x}_k)$$



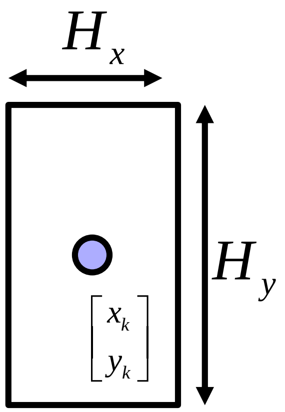
# Bootstrap particle filter by example

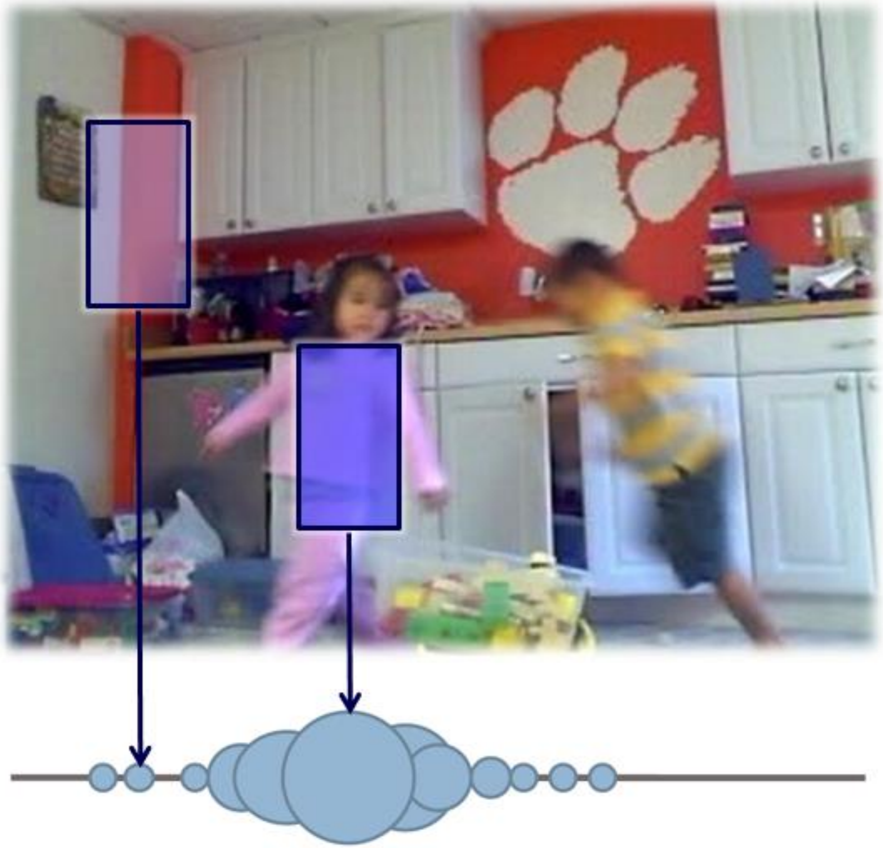
- Will be considering the following example:
- A particle is state + weight:

$$\mathbf{s}_k^{(i)} = \{\mathbf{x}_k^{(i)}, w_k^{(i)}\}$$

$$\left\{ \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \\ H_{xk} \\ H_{yk} \end{bmatrix}^{(i)}, w_k^{(i)} \right\}$$

- State: A rectangle



# Bootstrap particle filter recursion (step1)

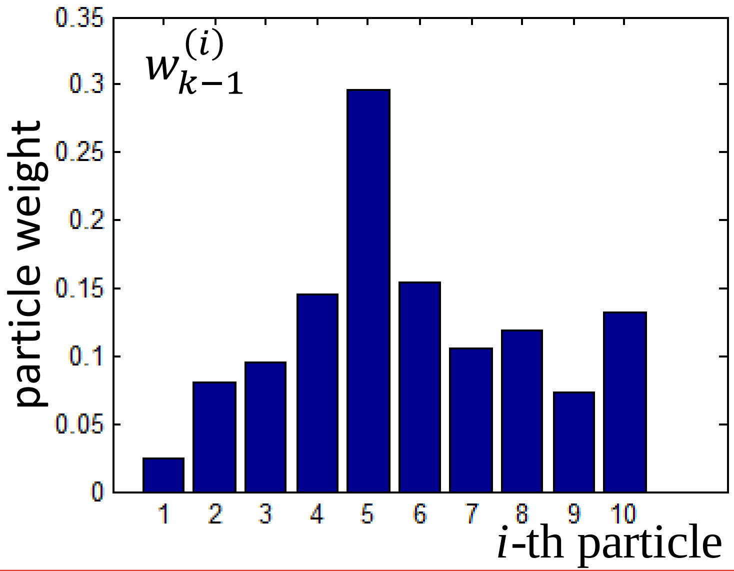
- Start with the posterior from the previous time-step.
- Recall, the posterior is a set of weighted samples.

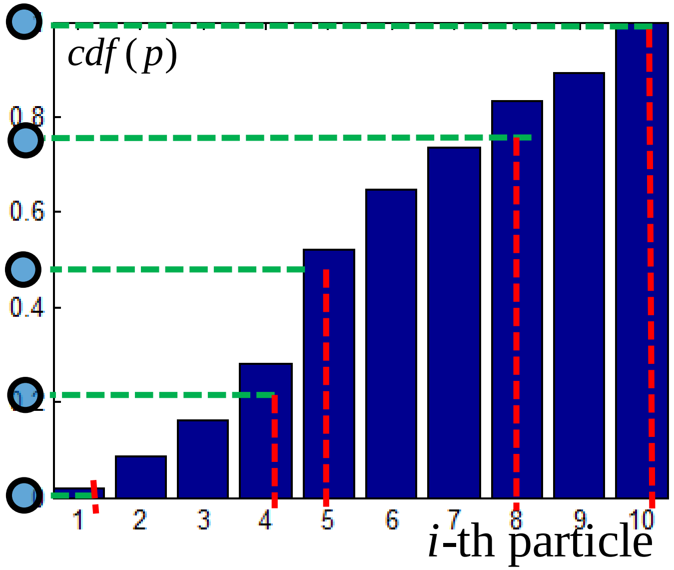
$$\{\mathbf{x}_{k-1}^{(i)}, w_{k-1}^{(i)}\}$$


$$p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) \approx \sum_{i=1}^{N} w_{k-1}^{(i)} \delta_{\mathbf{x}_{k-1}^{(i)}}(\mathbf{x}_{k-1})$$

# Resampling

- How to draw samples from $p(\mathrm{x}_{k-1} \mid \mathrm{y}_{1:k-1}) \approx \sum_{i=1}^{N} \mathrm{w}_{k-1}^{(\mathrm{i})} \, \delta_{\mathrm{x}_{k-1}^{(\mathrm{i})}}(\mathrm{x}_{k-1})$ ?
- Weighted particle set is a discrete distribution!
- For example, ten particles: $\left\{ \mathrm{x}^{(1)}, \mathrm{x}^{(4)}, \mathrm{x}^{(5)}, \mathrm{x}^{(8)}, \mathrm{x}^{(10)} \right\}$
  (weight of each new particle is 1/5)
- Draw *M*=5 samples:

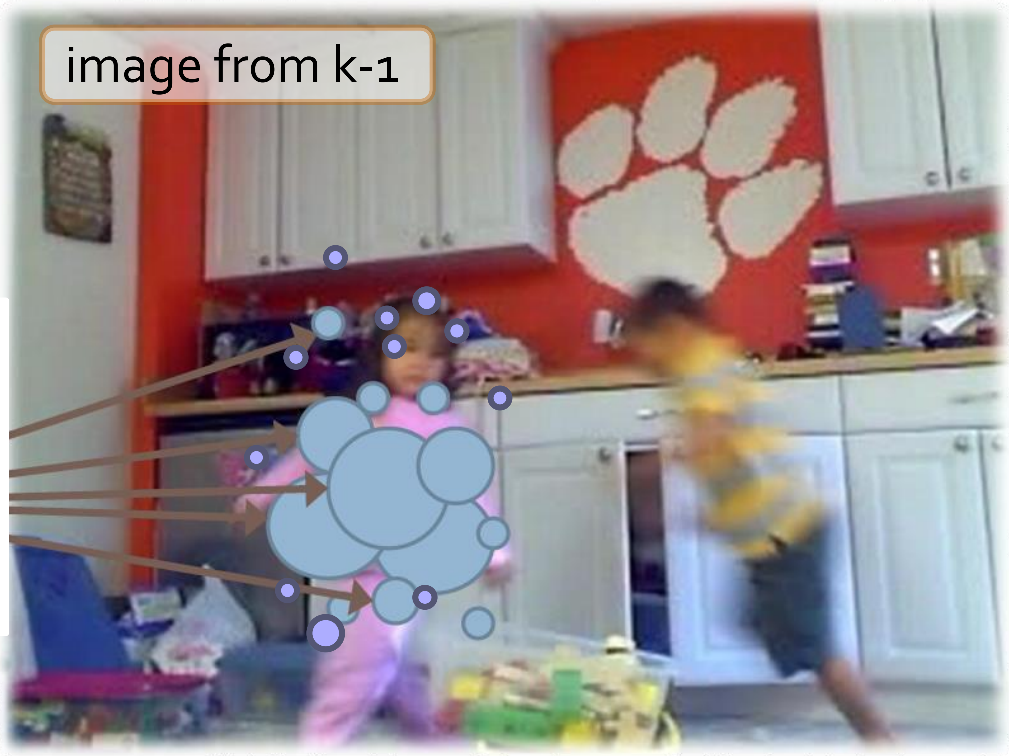


# Bootstrap particle filter recursion (step2)

- N new samples are drawn from the previous set.
- Some samples are repeated multiple times, while other are never selected.

$$p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) \approx \{\tilde{\mathbf{x}}_{k-1}^{(i)}, \frac{1}{N}\}$$

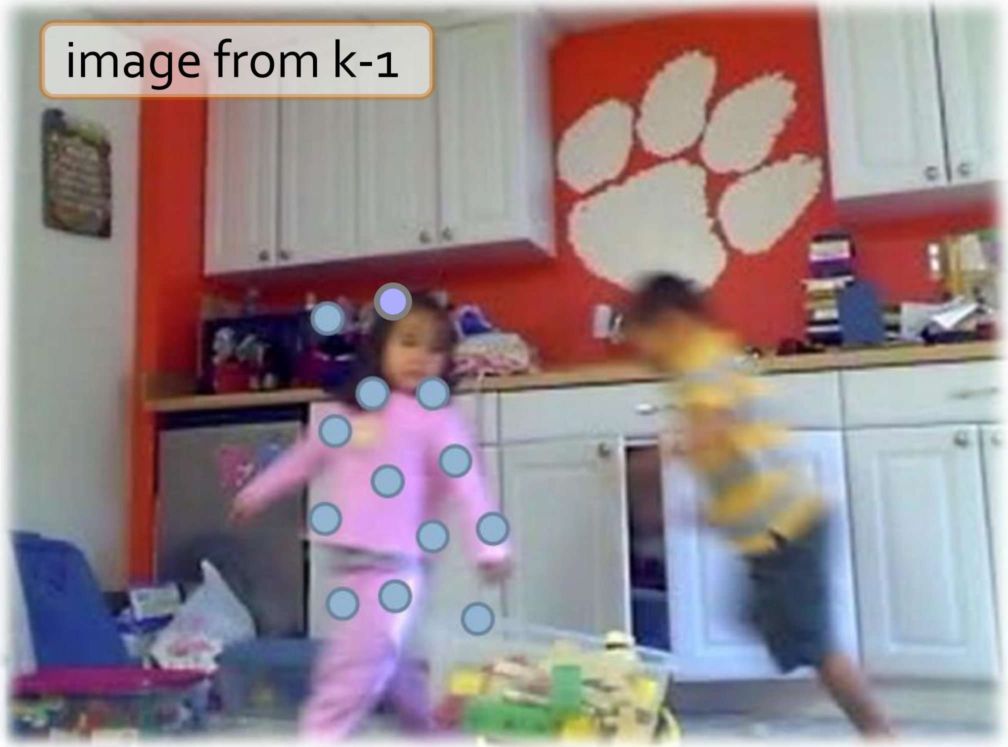



Before resampling

# Bootstrap particle filter recursion (step2)

- N new samples are drawn from the previous set.
- Some samples are repeated multiple times, while other are never selected.

$$p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1}) \approx \{\tilde{\mathbf{x}}_{k-1}^{(i)}, \frac{1}{N}\}$$




After resampling

# Bootstrap particle filter recursion (step3)

- Apply the motion model $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ to every particle:

$$\mathbf{x}_k^{(i)} = \underbrace{\Phi \mathbf{x}_{k-1}^{(i)}}_{\text{Linear motion}} + \underbrace{\mathbf{w}_k^{(i)}}_{\text{Noise term}}$$

- For each particle simulate its own noise $\mathbf{w}_k^{(i)}$

$$\mathbf{w}_k^{(i)} \sim \mathcal{N}(\cdot|\mathbf{0};\mathbf{Q})$$

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \{\mathbf{x}_k^{(i)}, \tfrac{1}{N}\}$$

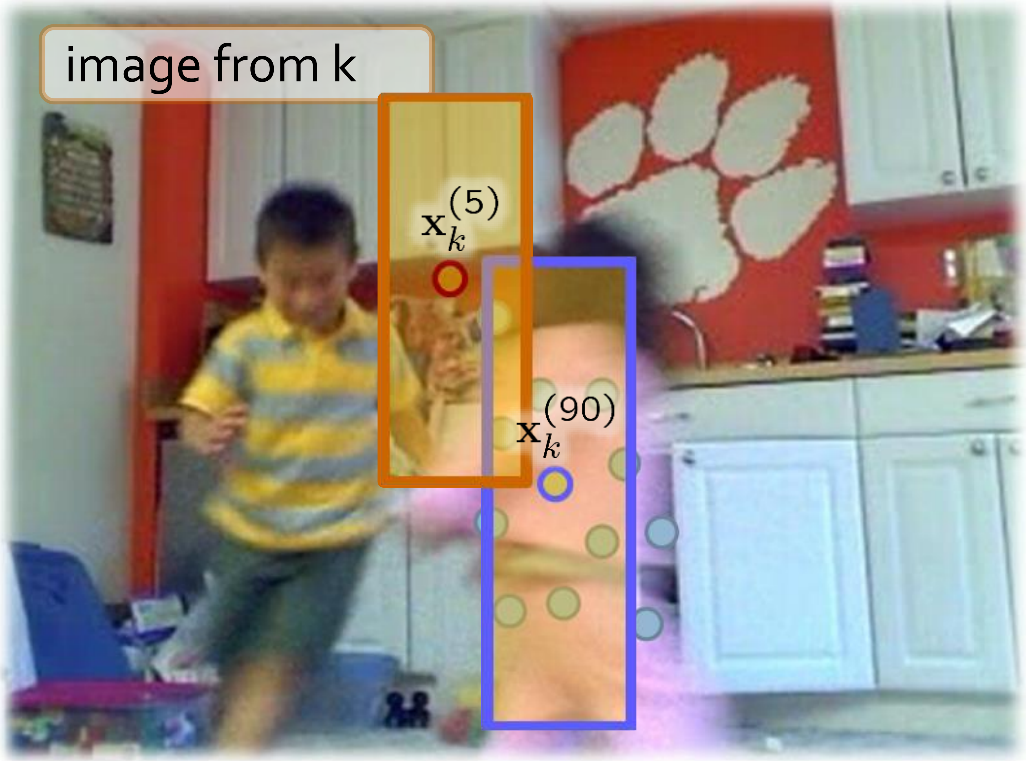
# Bootstrap particle filter recursion (step4)

- For each state $x_k^{(i)}$ obtain an observation $y_k^{(i)}$
- Evaluate likelihood:

  $p(\mathrm{y}_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})})$

- Set the weight to the likelihood value:

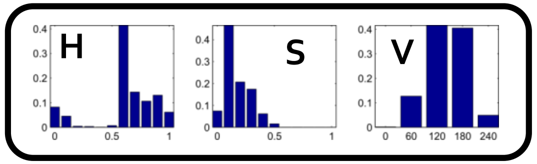
  $w^{(\mathrm{i})} = p(y_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})})$




- For example, consider the states indexed by 5 and 90.

# Bootstrap particle filter recursion (step4)

reference model: $h_{ref}$

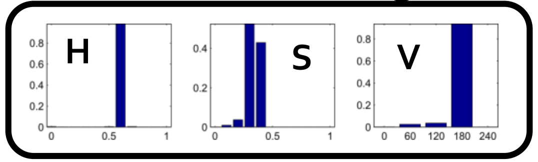


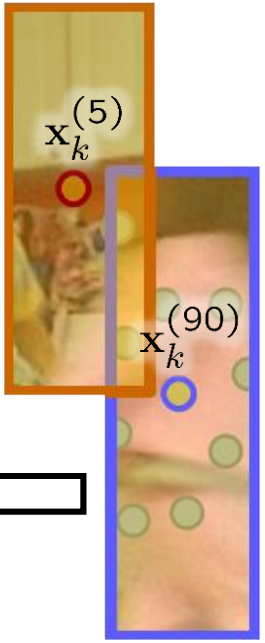

- Example of likelihood:

$$p(\mathrm{y}_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})})$$

$\mathbf{y}_k^{(5)}$ = color histogram

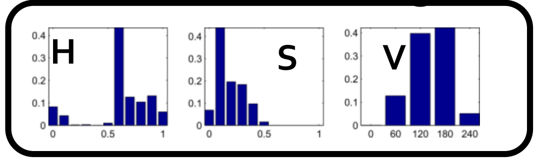
Poor match




$\mathbf{y}_k^{(90)}$ = color histogram

Good match




$dist(y_k^{(i)}, h_{ref})$

e.g., Hellinger distance

- Probability of observing a histogram $y_k^{(i)}$,

assuming that the target is at $x_k^{(i)}$: $\quad p(y_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})}) = \mathrm{e}^{(-\frac{1}{2}\mathrm{dist}(y_k^{(\mathrm{i})}, \mathrm{h}_{ref})^2/\sigma^2)}$

# Bootstrap particle filter recursion (step4)

- For each state $x_k^{(i)}$ obtain an observation $y_k^{(i)}$
- Evaluate likelihood:

  $p(\mathrm{y}_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})})$

- Set the weight to the likelihood:

  $w^{(\mathrm{i})} = p(y_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})})$

Before measurement update.

# Bootstrap particle filter recursion (step4)

- For each state $x_k^{(i)}$ obtain an observation $y_k^{(i)}$
- Evaluate likelihood:

  $p(\mathrm{y}_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})})$
- Set the weight to the likelihood:

  $w^{(\mathrm{i})} = p(y_k^{(\mathrm{i})} \mid \mathrm{x}_k^{(\mathrm{i})})$

After updating.

# Bootstrap particle filter steps in a nutshell

Posterior from previous time-step


Resample

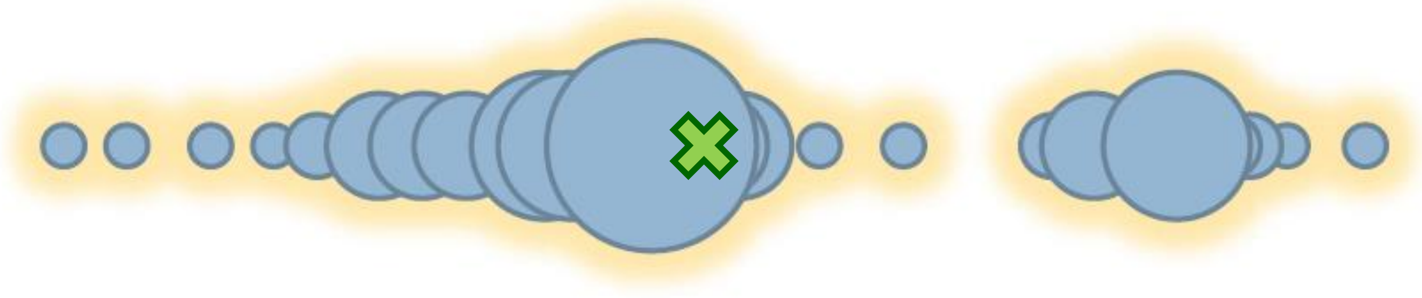

Predict particles


Evaluate weights


See J.D. Prince “Computer vision: models, learning and inference”, Section 19.5

# Making sense of the estimated pdf

- Estimate the current state of the target from the pdf.



- Interpretation of the pdf depends on application.
- A common approach is to calculate the mean particle.
- The mean particle (expected value over pdf):

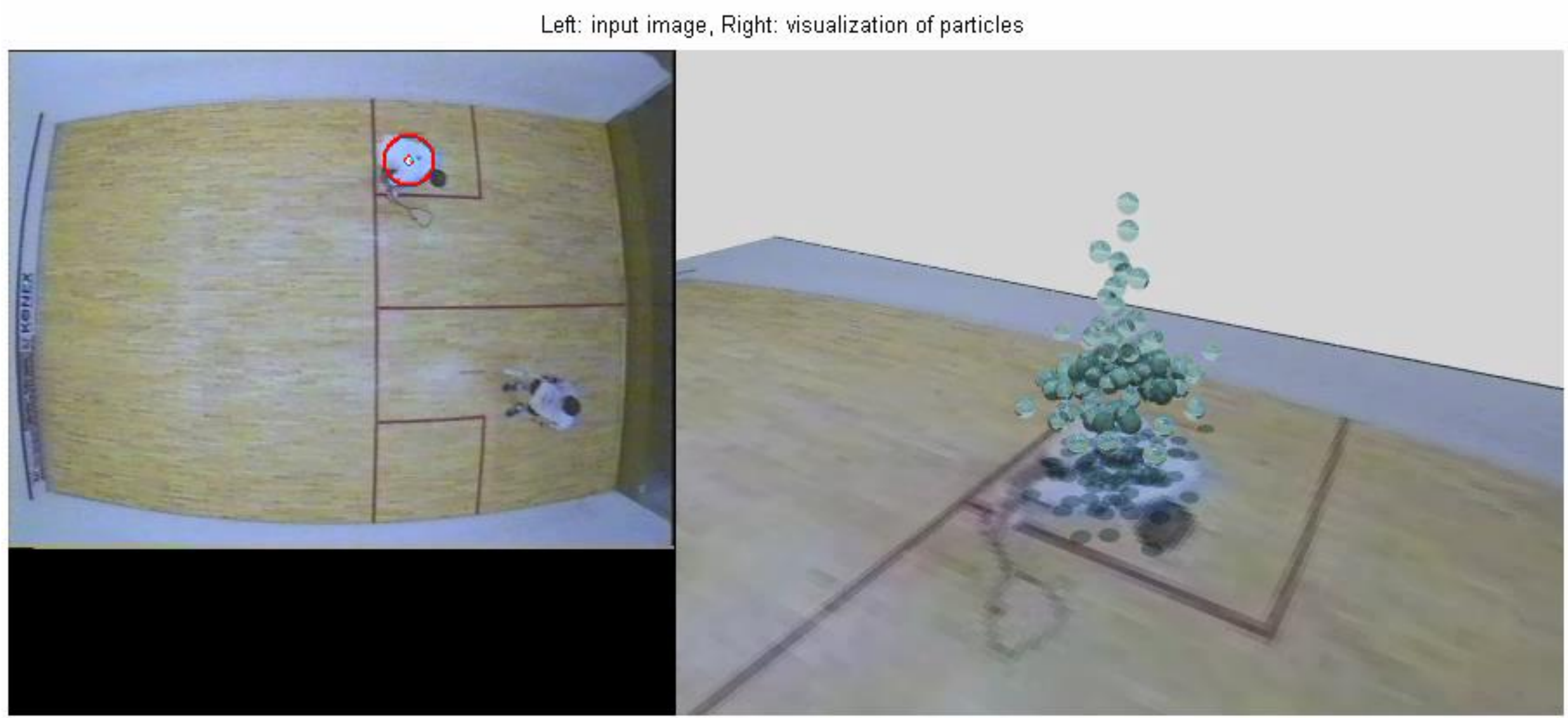
$$\tilde{x}_k = \langle x_k \rangle_{p(\mathrm{x}_k | y_{1:k})} = \int x p(\mathrm{x}_k \mid y_{1:k}) \, \mathrm{dx} = \sum_{i=1}^{N} w^{(\mathrm{i})} x_k^{(\mathrm{i})}$$

# Particle filter in action

All states in distribution

The mean state

# Particle filter in action

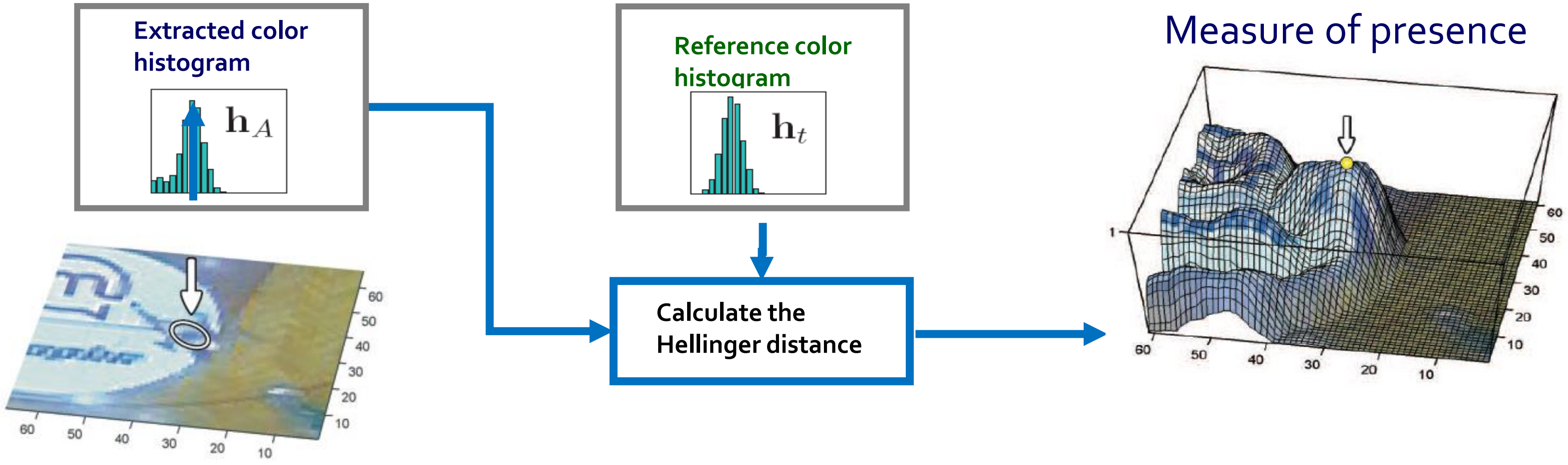



Input image and the mean particle.

Particle distribution. Only ellipse centers are shown, the height represents weight.

# Effects of the visual model

- A simple histogram-based visual model (joint RGB)



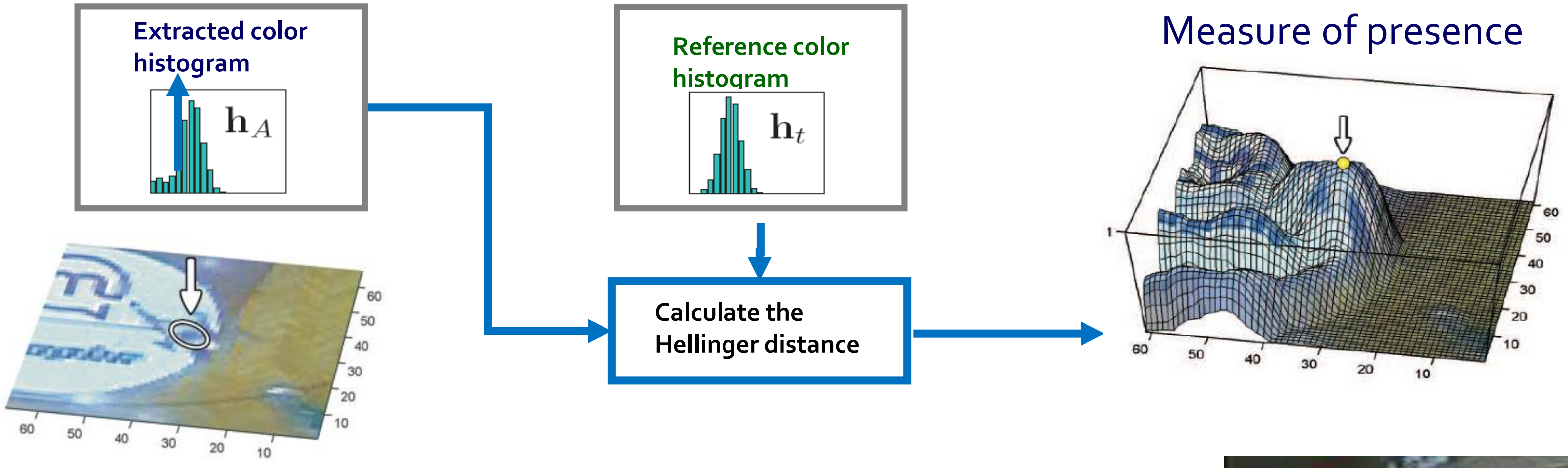

- After a tracking iteration update the reference histogram $h_t$ by the histogram $h_A$ sampled from the estimated position.

$$h_{t+1} = \alpha h_A + (1-\alpha) h_t$$

$\alpha = 0.05$ … adaptation constant

# Effects of the visual model

- A simple histogram-based visual model (joint RGB)

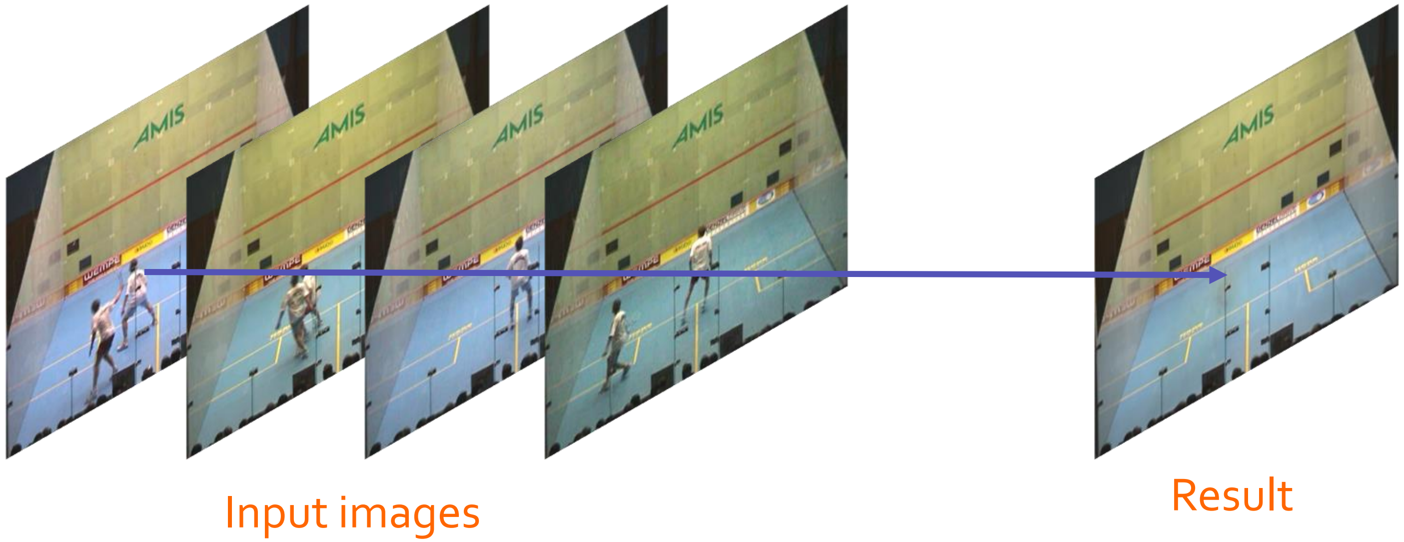



Tracking fails **14** times due to clutter.

Maybe accounting for the background would help….

# A simple background image

- CAUTION: *Assumes a static camera*.
- Many approaches exist, most popular are based on Mixture models e.g., [Stauffer and Grimson 1999]
- A simple background estimate by selecting random images and taking median intensity along each pixel.

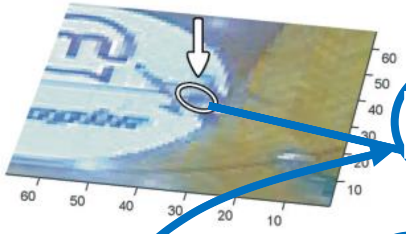



Stauffer and Grimson, "Adaptive background mixture models for real-time tracking," CVPR 1999.
Piccardi, "Background subtraction techniques: a review", IEEE SMC2004

# Effects of the visual model

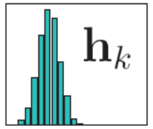
- Accounting for the background

**current image**



$\mathbf{h}_A$

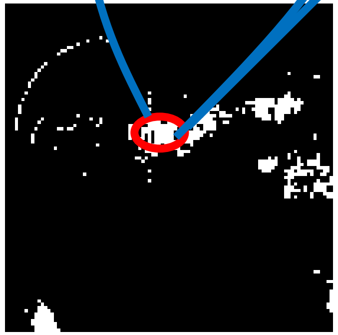
**reference**




**background image**


$\mathbf{h}_B$

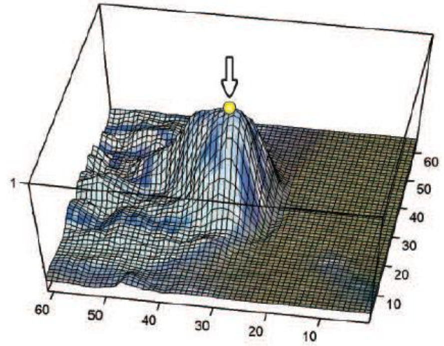
**mask function**



**Normalized Hellinger distance-based measure**

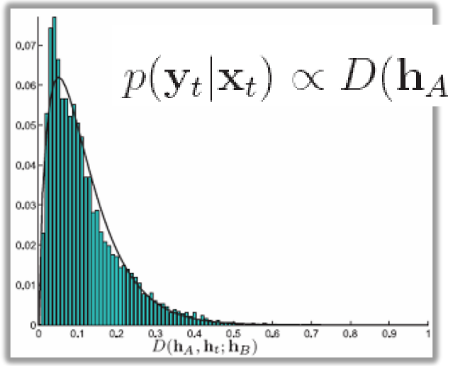
**Ratio between the number of visible and invisible pixels $\beta$ .**

$$1-\rho(\mathrm{h}_A, h_k; h_B)$$



**Measure of presence:**

$$D(\mathbf{h}_A, \mathbf{h}_k; \mathbf{h}_B) = \beta^{-1}\rho(\mathbf{h}_A, \mathbf{h}_k; \mathbf{h}_B)$$



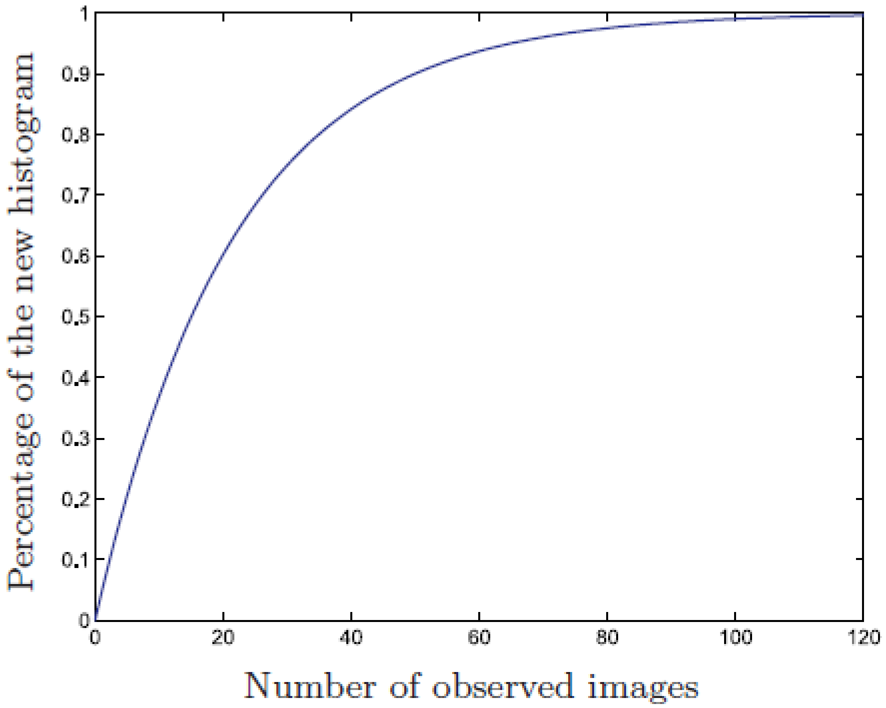
$$p(\mathbf{y}_t|\mathbf{x}_t) \propto D(\mathbf{h}_A, \mathbf{h}_t; \mathbf{h}_B)^{\gamma_1 - 1} e^{\frac{-D(\mathbf{h}_A, \mathbf{h}_t; \mathbf{h}_B)}{\gamma_2}}$$

Kristan et al.,"Closed-world tracking of multiple interacting targets for indoor-sports applications". CVIU2009.

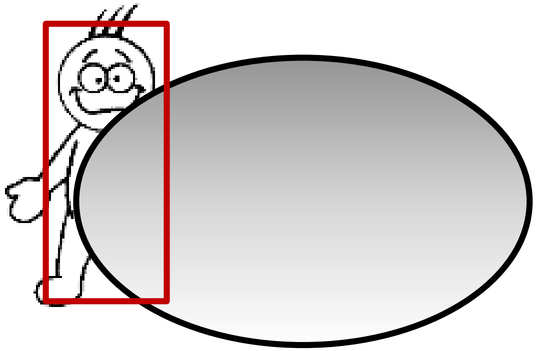
# Effects of the visual model

- Constant adaptation may not be always appropriate
- Consider the following constant adaptation

$$h_{t+1} = \alpha h_A + (1-\alpha) h_t$$




Percentage of the new histogram in the reference histogram when using $\alpha = 0.05$.



What does this mean for occlusion?

# Effects of the visual model

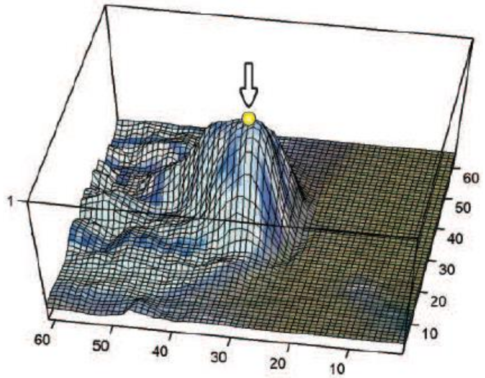
- Constant adaptation does not stop during occlusion!
- Require non-constant adaptation:

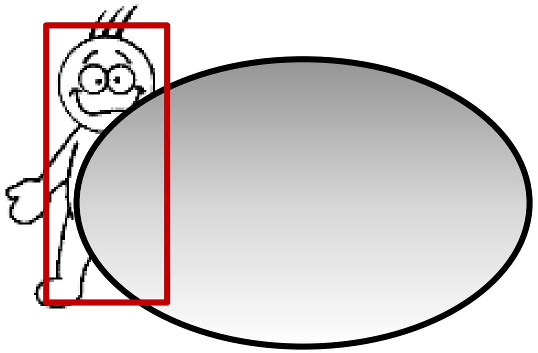
$$h_k = \alpha_k h_k + (1 - \alpha_k) h_{k-1}$$

$$\alpha_k = \Omega_{max} \cdot (1.0 - \rho(\mathbf{h}_A, \mathbf{h}_{k-1}; \mathbf{h}_B))$$

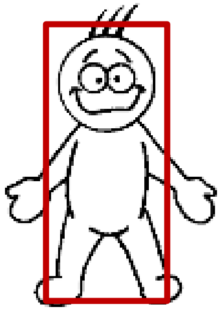
Maximal allowed adaptation ($\Omega_{max} = 0.05$)

$$1 - \rho(\mathrm{h}_A, h_k; h_B)$$



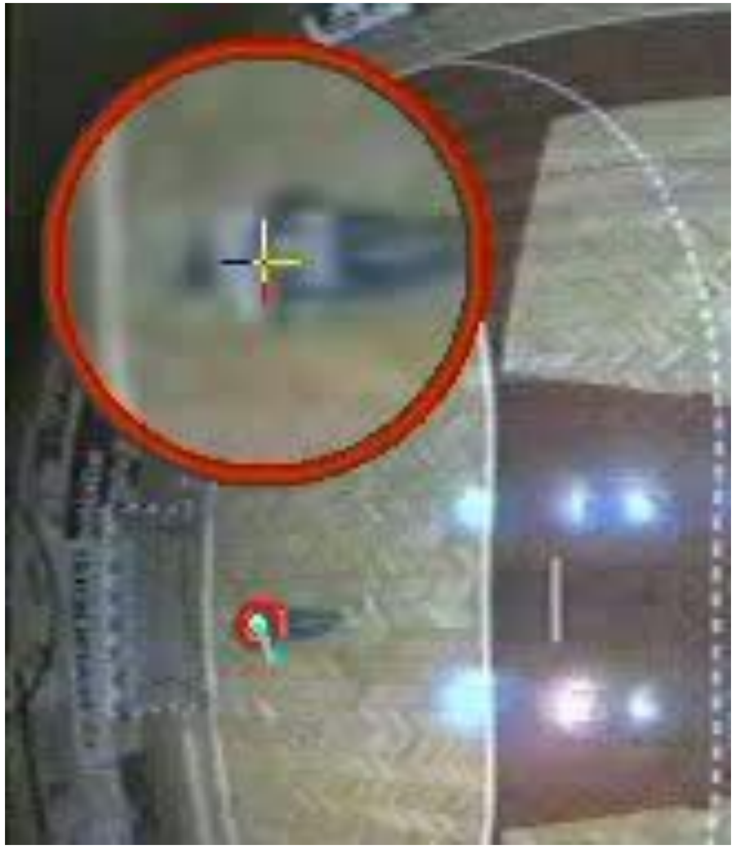
Low adaptation ($\alpha_k \approx 0$)



High adaptation ($\alpha_k \approx 0.05$)

# Effects of the visual model

Not accounting for the background



14 failures

The background image

Accounting for the background

1 failure

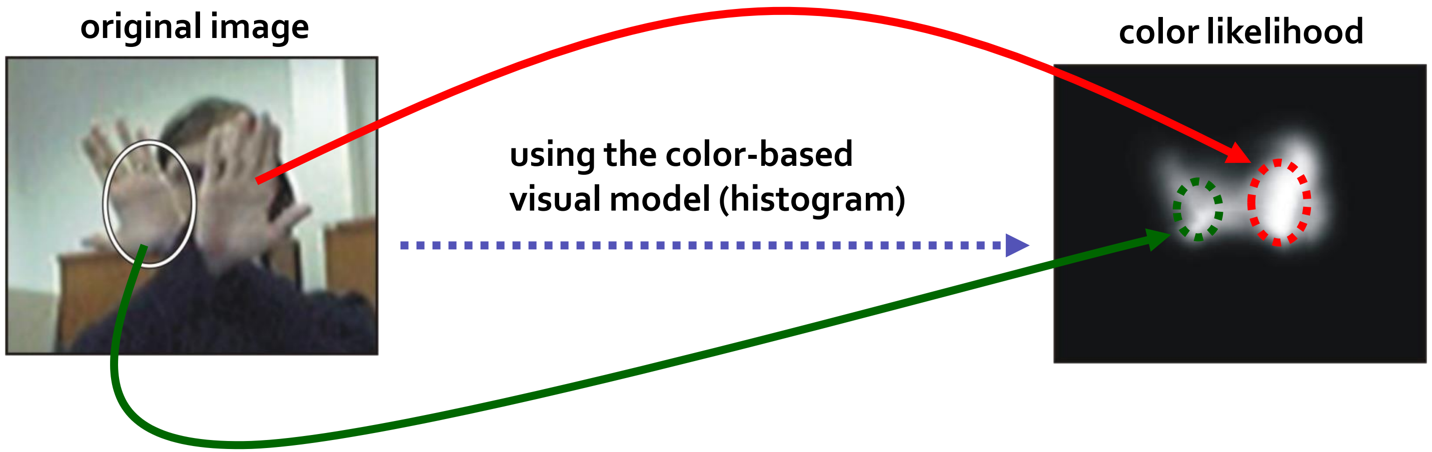
# Effects of the visual model

- Occlusion by a visually-similar object.

14 failures

# Color ambiguity

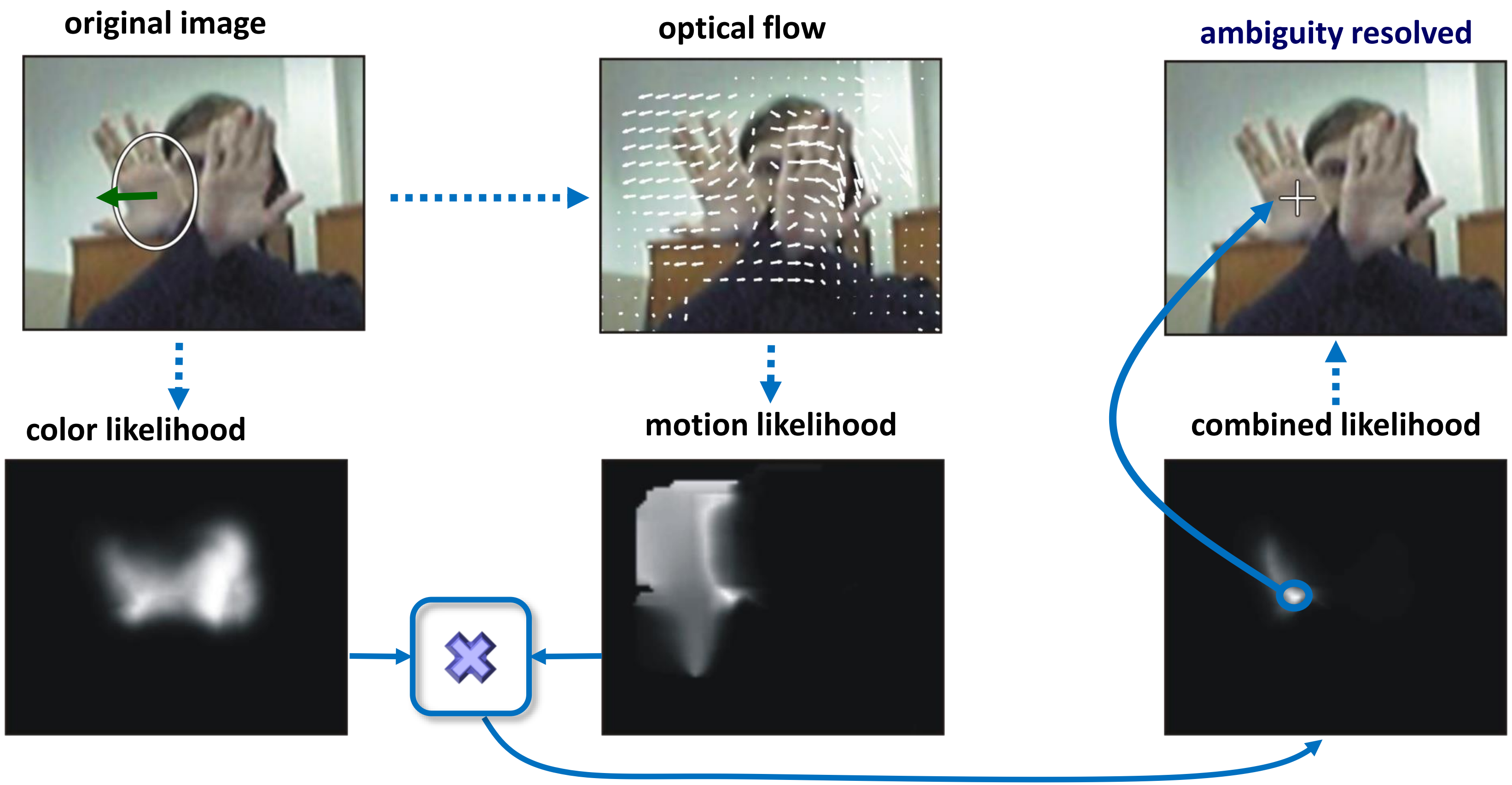
- The problem of the *color ambiguity*: Tracking hands




- Problem: The color likelihood function is ambiguous with respect to the location of the person’s hand.
- The mode stretches accross both hands, which usually causes a failure in tracking.

# The local motion visual model

- Use the motion information.



# The local motion visual model

- Hand tracking revisited:

color-based model 14 failures


combined model 2 failures


- The combined visual model significantly improves tracking in presence of occlusions.

M. Kristan et al., "A Local-motion-based probabilistic model for visual tracking". *Pattern Recognition*, 2009.

# The "take home" message

1. Uncertainties pose major challenge in tracking.
2. Recursive Bayes filters deal naturally with these uncertainties.
3. World is linear and Gaussian: Kalman filter
4. Relaxing these assumptions: Particle filters

# The “take home” message

5. Take care when implementing:
    - Probabilistic visual model.
    - Probabilistic dynamic model.

6. Many open challenges remain:
   *initialization, drift, target dynamics, efficient schemes for multiple targets, visual models, occlusion, online appearence learning, etc.*

# References


- Color-based bootstrap particle filter:
  - P. Pérez, C. Hue, J. Vermaak, M. Gangnet, "Color-Based Probabilistic Tracking", ECCV2002
  - K. Nummiaro, E. Koller-Meier and L. Van Gool, ”An adaptive color-based particle filter”,Image and Vision Computing, Vol. 21, No. 1, pp. 99-110, 2002
- A wider disposition of particle filters
  - Arulampalam et al., “A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking”. TSP2002.

# References

- Text book Kalman:
  - S. J.D. Prince, “Computer vision: models, learning and inference”, Section 19.2
- Text book Particle filter:
  - S. J.D. Prince, “Computer vision: models, learning and inference”, Section 19.5
- For additional info on probability see:
  - S. J.D. Prince, “Computer vision: models, learning and inference”, Chapter 1

# Acknowledgement


- Some images and parts of slides have been taken from the following talks:
  - Kevin Smith’s ”SELECTED TOPICS IN COMPUTER VISION – 2D tracking”

# Next week we'll talk about trainable trackers

- Please read a few slides from here on deep learning: http://cs231n.stanford.edu/
  - Module 2: Convolutional Neural Networks ->Convolutional Neural Networks: Architectures, Convolution / Pooling Layers
- Great if you also have a look at Transformers:
  - Eg., https://jalammar.github.io/illustrated-transformer/