

# Online Learning for Tracking

Robert Collins

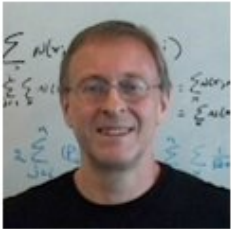
July 25, 2009

VLPR Summer School. Beijing, China.

# We Are...



## Faculty



Robert T. Collins



Yanxi Liu



David Capel

Penn State  
Lab for Perception, Action and Cognition

## Graduate Students



Weina Ge



Somesh Kashyap



SeungKyu Lee



Minwoo Park



Ingmar Rauschert



Zhaozheng Yin



Chen-Ping Yu



Timothy Smith



Kyle Brocklehurst

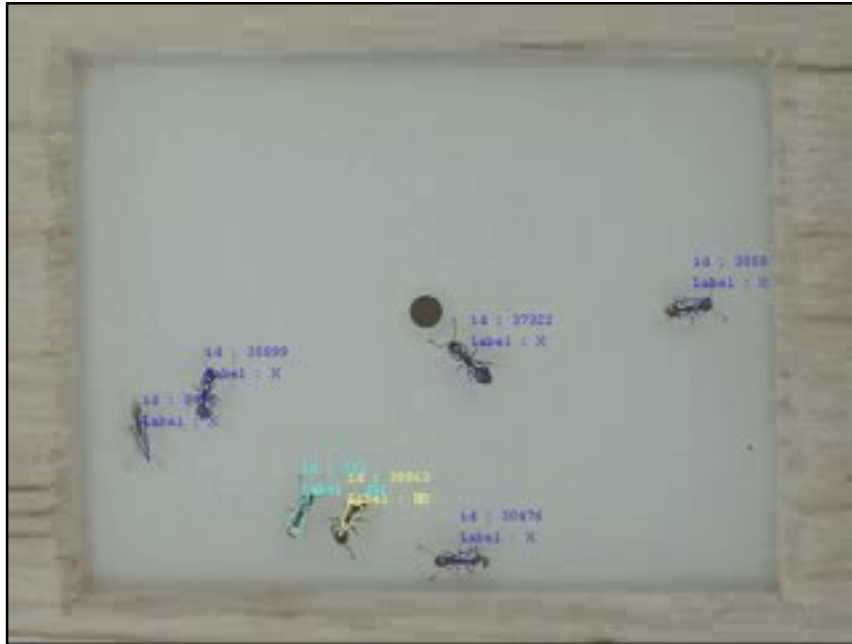
# What is Tracking?



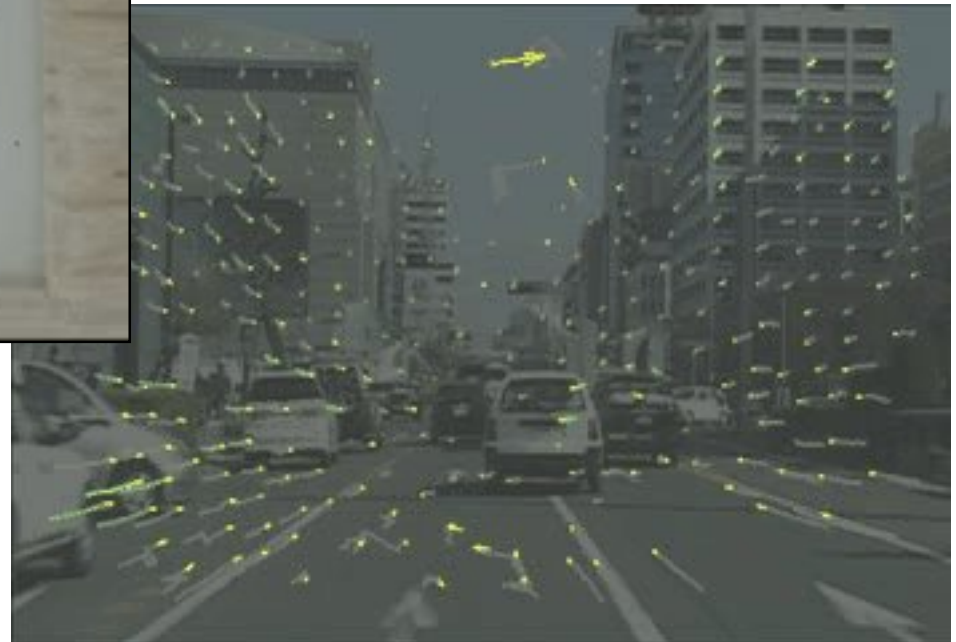
typical idea: tracking a single target in isolation.

# What is Tracking?

Multi-target tracking....



ant behavior, courtesy of  
Georgia Tech biotracking

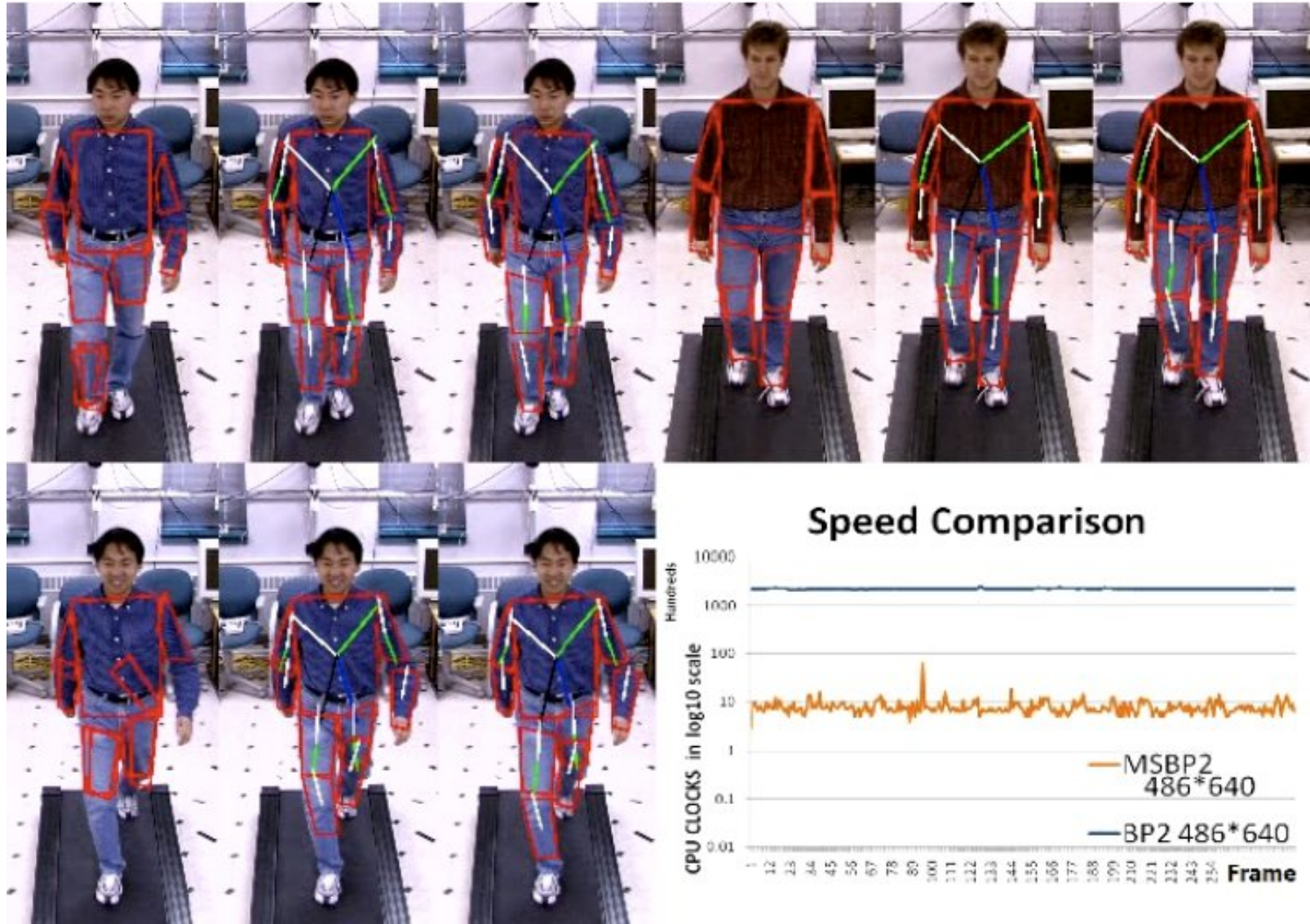


“targets” can be corners, and  
tracking gives us optic flow.



# What is Tracking?

articulated objects having multiple, coordinated parts



# What is Tracking?

Active tracking involves moving the sensor in response to motion of the target. Needs to be real-time!



# Lecture Outline

- Brief Intro to Tracking
- Appearance-based Tracking
- Online Adaptation (learning)

# State Space Approach

Two vectors of interest:

- 1) State vector: vector of variables  $x_k$  representing what we want to know about the target object  
*examples:  $[x,y]$ ;  $[x,y,dx,dy]$ ;  $[x,y,\theta,scale]$*
- 2) Measurement vector: noisy observations  $z_k$  related to the state vector.  
*examples: image intensity/color; motion blobs*

Because our observations will be noisy, estimating the state vector will be a statistical estimation problem.



# What is Tracking ?

What distinguishes tracking from “typical” statistical estimation (or machine learning) problems?

- We typically have a strong temporal component involved.
- estimating quantities that are expected to change over time (thus expectations of the dynamics play a role)
- interested in current state  $S_t$  for a given time step  $t$
- usually assume can only compute  $S_t$  from information seen at previous times steps  $1, 2, \dots, (t-1)$ . [can't see the future]
- usually want to be as efficient as possible, even “real-time”.

These concerns lead naturally to recursive estimators.

# Bayesian Filtering

Rigorous general framework for tracking. Estimates the values of a state vector based on a time series of uncertain observations.

Key idea: use a recursive estimator to construct the posterior density function (pdf) of the state vector at each time  $t$  based on all available data up to time  $t$ .

Bayesian hypothesis: All quantities of interest, such as MAP or marginal estimates, can be computed from the posterior pdf.

# Filtering Framework

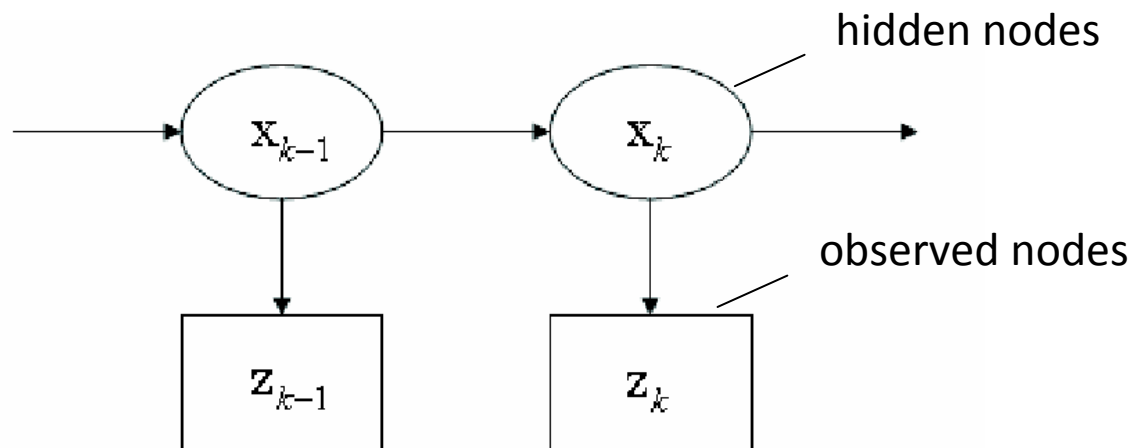
We want to recursively estimate the current target state vector each time a new observation is received.

Two step approach:

- 1) prediction: propagate current state forward in time, taking process noise into account (translate, deform, and spread the pdf)
- 2) update: use Bayes theorem to modify prediction pdf based on current observation

# Tracking as a Graphical Model

Graphical Model:



Markov assumptions

$$p(\mathbf{x}_k | \mathbf{x}_0, \dots, \mathbf{x}_{k-1}) = p(\mathbf{x}_k | \mathbf{x}_{k-1})$$

$$p(\mathbf{z}_k | \mathbf{x}_0, \dots, \mathbf{x}_k) = p(\mathbf{z}_k | \mathbf{x}_k)$$

Factored joint probability distribution

$$p(\mathbf{x}_0, \dots, \mathbf{x}_k, \mathbf{z}_1, \dots, \mathbf{z}_k) = p(\mathbf{x}_0) \prod_{i=1}^k p(\mathbf{z}_i | \mathbf{x}_i) p(\mathbf{x}_i | \mathbf{x}_{i-1})$$

# Recursive Bayes Filter

Motion Prediction Step:

predicted current state                      state transition                      previous estimated state

$$\underline{p(\mathbf{x}_k | \mathbf{z}_{1:k-1})} = \int \underline{p(\mathbf{x}_k | \mathbf{x}_{k-1})} \underline{p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1})} d\mathbf{x}_{k-1}.$$

Data Correction Step (Bayes rule):

estimated current state                      measurement                      predicted current state

$$\underline{p(\mathbf{x}_k | \mathbf{z}_{1:k})} = \frac{\underline{p(\mathbf{z}_k | \mathbf{x}_k)} \underline{p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}}{\underline{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})}}$$

↓ normalization term

$$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) d\mathbf{x}_k$$



# Problem

Except in special cases, these integrals are intractable.

Motion Prediction Step:

$$p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} | \mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1}.$$

Data Correction Step (Bayes rule):

$$p(\mathbf{x}_k | \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})}$$

$$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) d\mathbf{x}_k$$

# Practical Note

Often the two types of probabilities  $P(x_k|x_{k-1})$  and  $P(z_k|x_k)$  are not explicitly given to you. Instead, two functions are given:

- 1) System model - how current state is related to previous state (specifies evolution of state with time)

$$x_k = f_k(x_{k-1}, v_{k-1}) \quad v \text{ is process noise}$$

- 2) Measurement model - how noisy measurements are related to the current state

$$z_k = h_k(x_k, n_k) \quad n \text{ is measurement noise}$$

and you have to be able to propagate distributions through these equations, which can be very difficult analytically.

# Special Case 1: Kalman Filter

With suitable assumptions, we can derive Kalman filtering and particle filtering from the recursive Bayes filter equations.

For example, if:

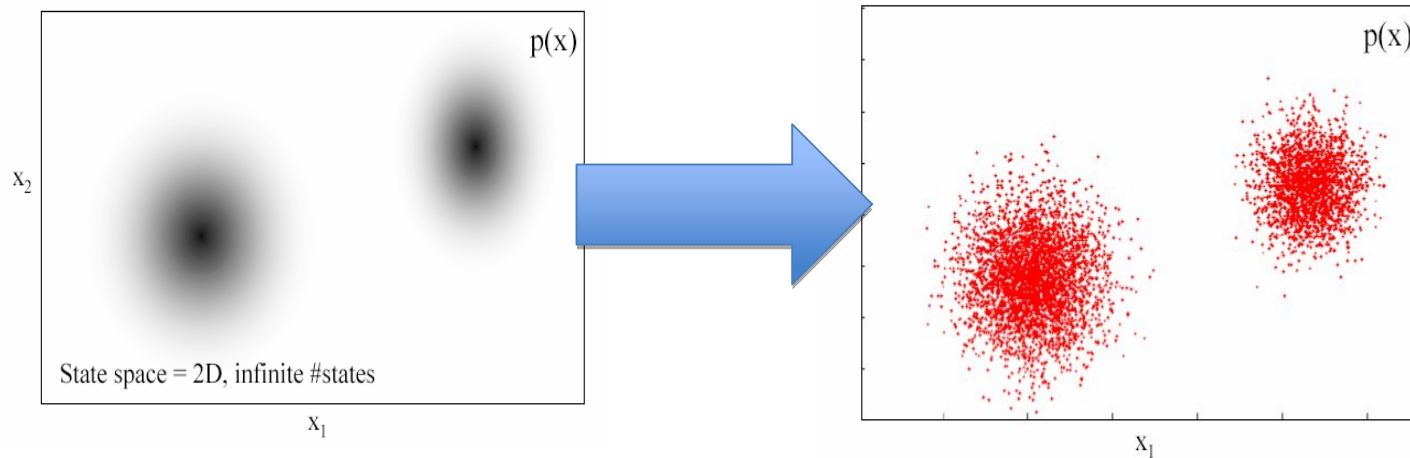
- Next state is a linear function of current state  
plus zero-mean Gaussian noise (process noise)
- Observation is linear function of current state  
plus zero-mean Gaussian noise (measurement noise)
- Initial prior distribution of first state is Gaussian

Then:

All distributions remain Gaussian, and we can solve the integrals analytically. The Kalman filter equations specify how to update the Gaussian mean and covariance parameters over time.

# Special Case 2: Particle Filter

Nonparametric representation of distributions with a discrete set of weighted samples (particles).



# Why Does This Help?

If we can represent a distribution  $P(x)$  by random samples  $x_i$  (particles), then we can compute marginal distributions and expected values by summation, rather than integration.

That is, we can approximate:

$$E(f(x)) = \int f(x)P(x)dx$$

by first generating  $N$  i.i.d. samples from  $P(x)$  and then forming the empirical estimate:

$$\hat{E}(f(x)) = \frac{1}{N} \sum_{i=1}^N f(x_i)$$



# Why Does This Help?

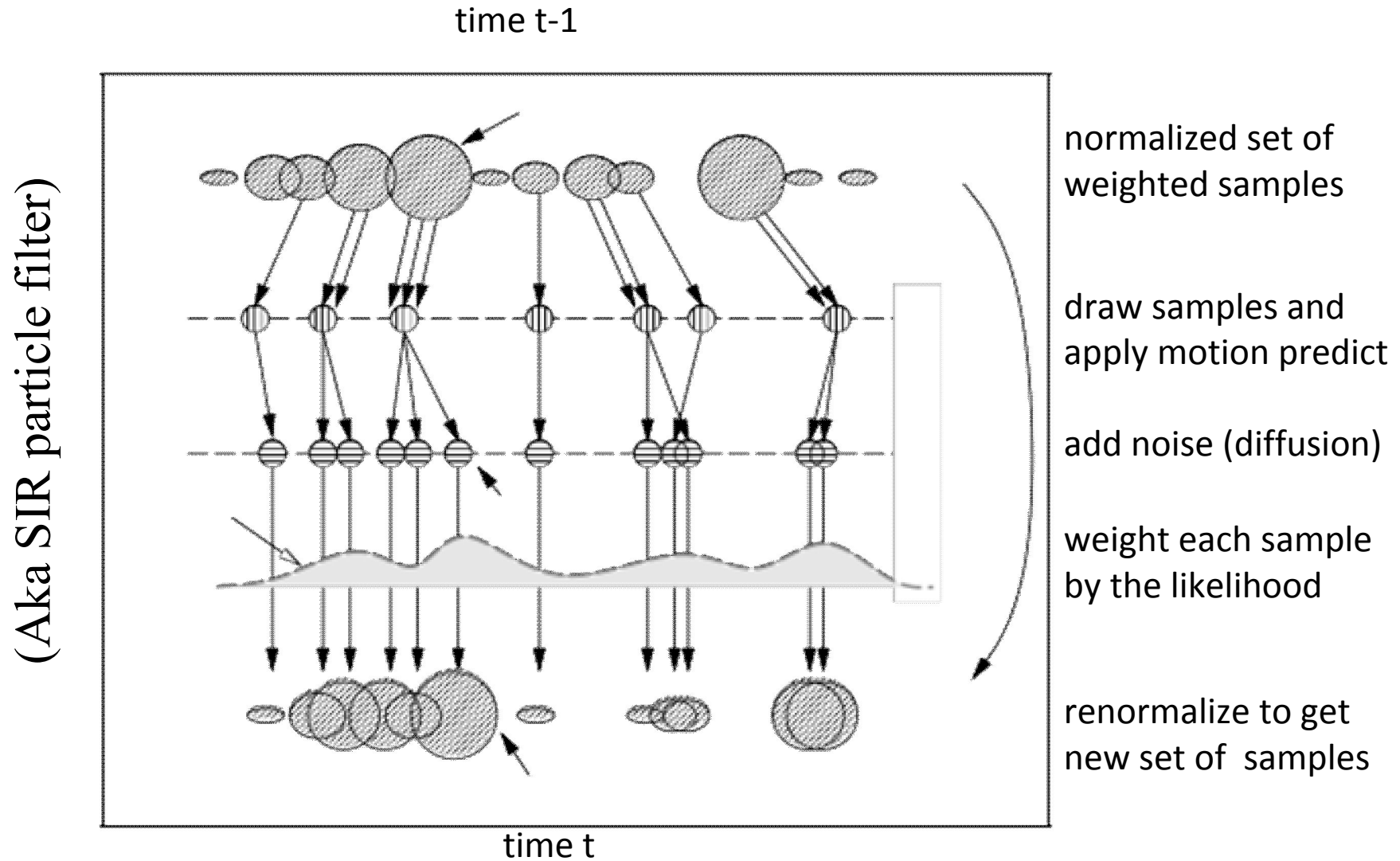
For example, the integral in the denominator of Bayes rule goes away for free, as a consequence of representing distributions by a weighted set of samples. Since we have only a finite number of samples, we can easily compute the normalization constant by summing the weights!

Data Correction Step (Bayes rule):

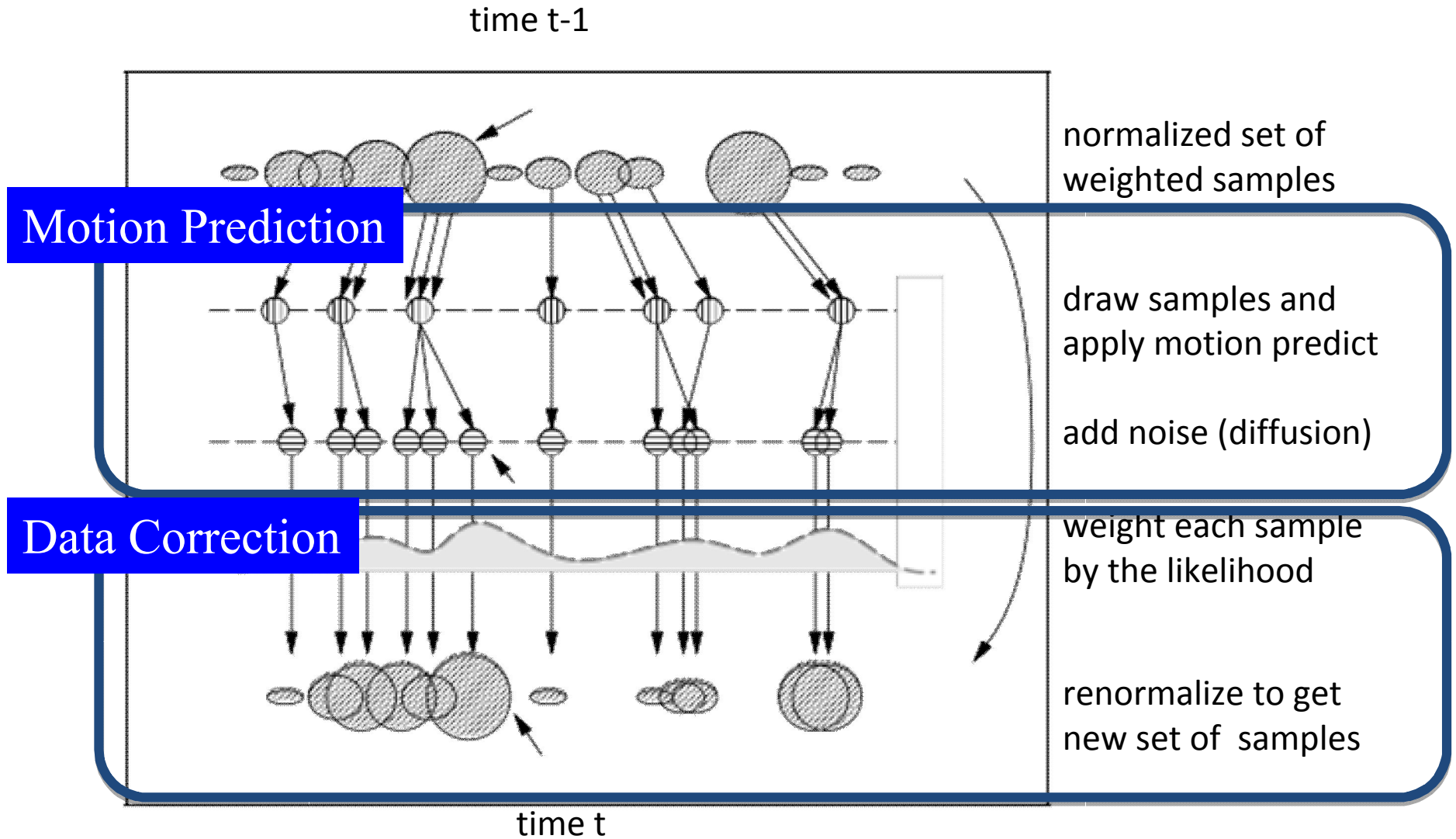
$$p(\mathbf{x}_k | \mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1})}{p(\mathbf{z}_k | \mathbf{z}_{1:k-1})}$$

$$p(\mathbf{z}_k | \mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) d\mathbf{x}_k$$

# Condensation (Isard&Blake)



# Condensation (Isard&Blake)



# Back to our Filtering Framework

Let's say we want to recursively estimate the current state at every time that a measurement is received.

Two step approach:

- 1) prediction: propagate state pdf forward in time, taking process noise into account (translate, deform, and spread the pdf)
- 2) update: use Bayes theorem to modify prediction pdf based on current measurement

But which observation should we update with?

# Filtering, Gating, Association

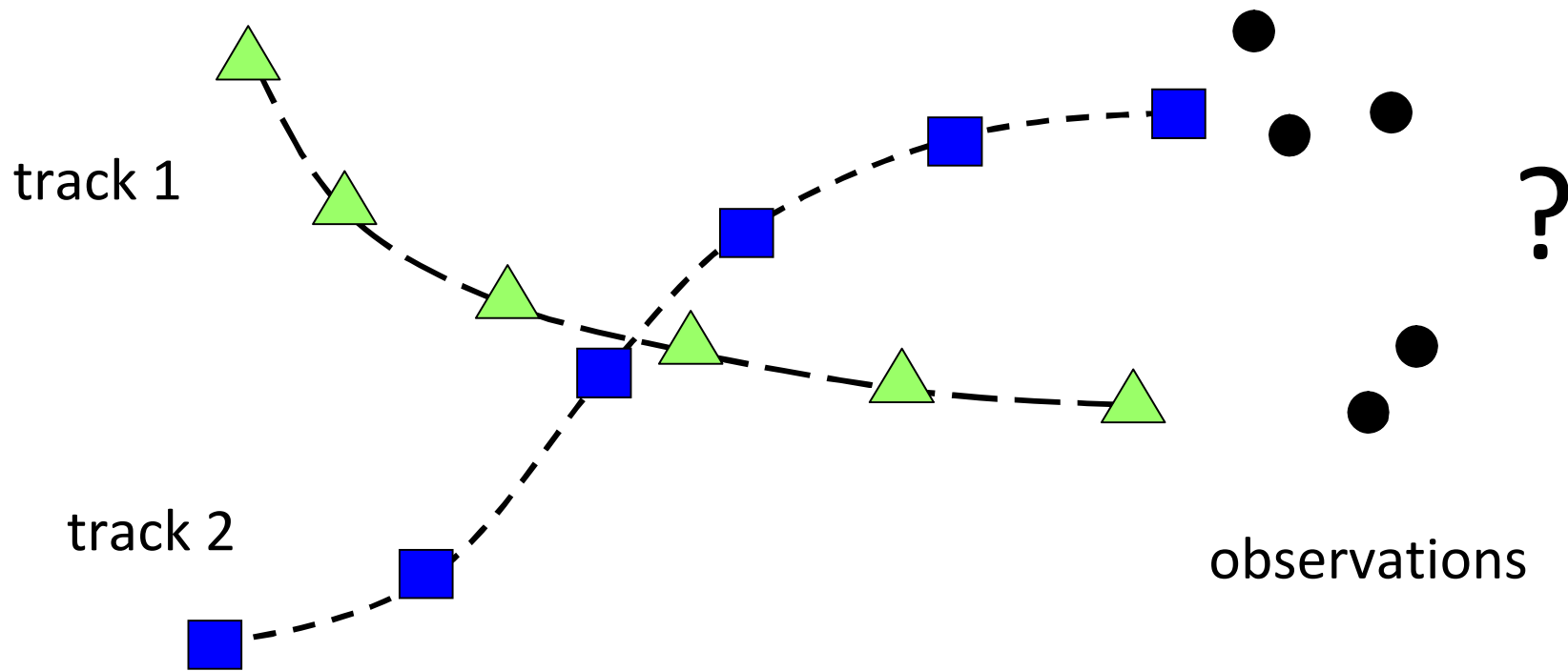
## Add Gating and Data Association

- 1) prediction: propagate state pdf forward in time, taking process noise into account (translate, deform, and spread the pdf)
- 1b) Gating to determine possible matching observations
- 1c) Data association to determine best match
- 2) update: use Bayes theorem to modify prediction pdf based on current measurement



# Data Association

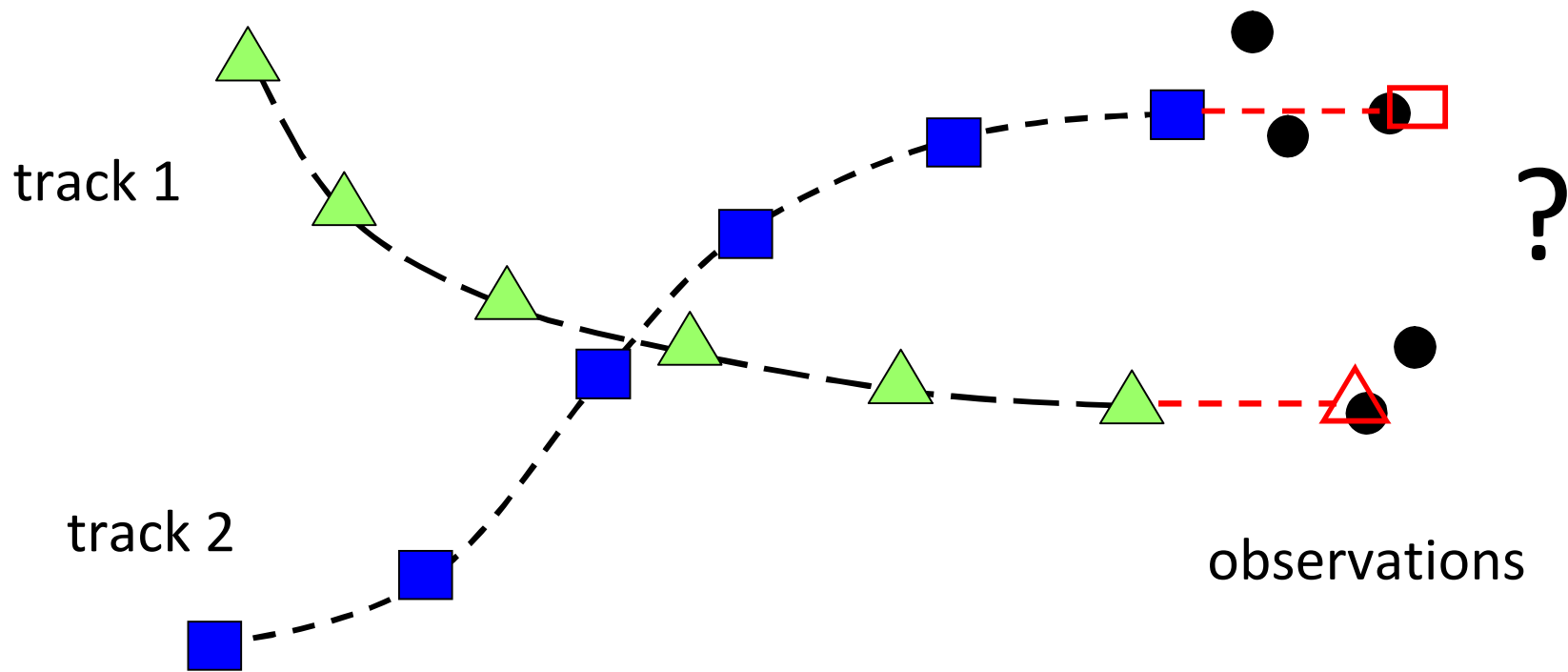
Occurs naturally in multi-frame matching tasks (matching observations in a new frame to a set of tracked trajectories)



How to determine which observations to add to which track?

# Track Matching

Intuition: predict next position along each track.

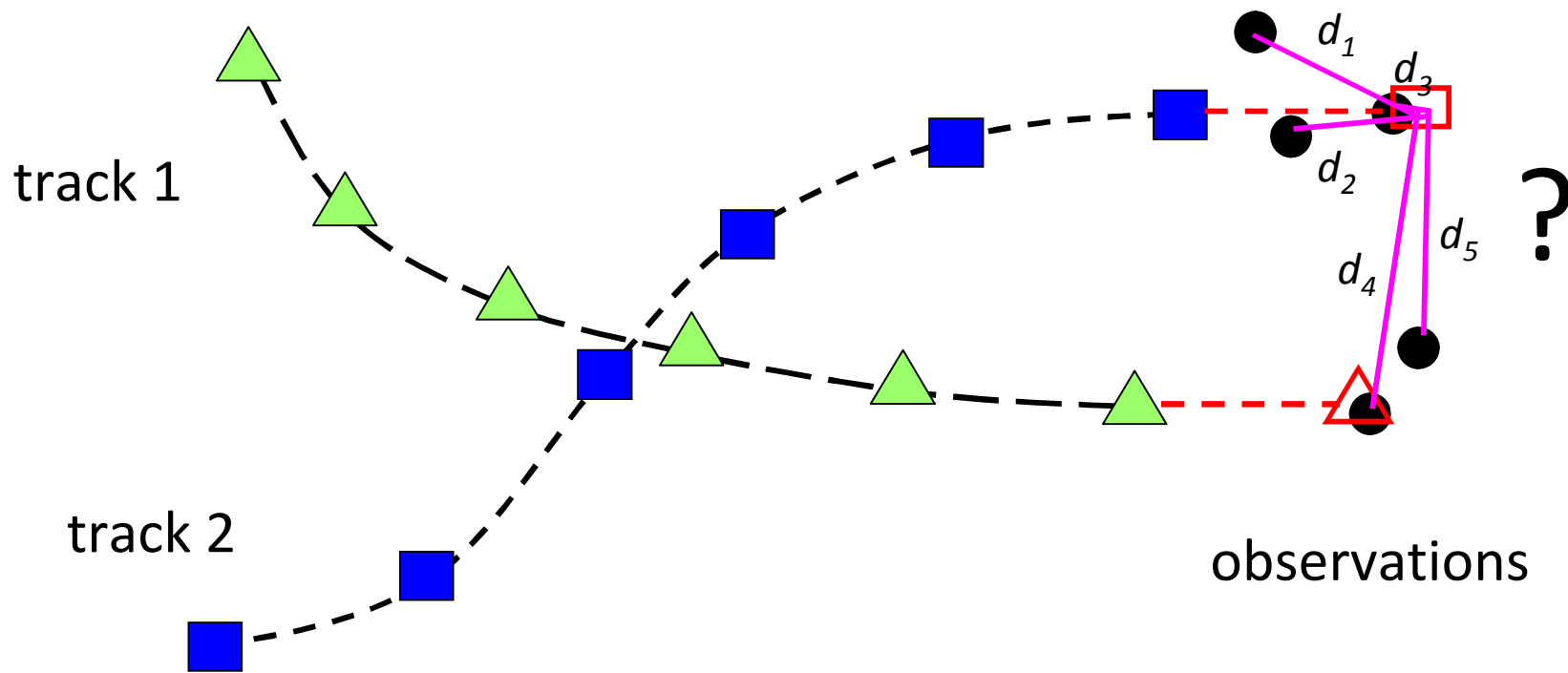


How to determine which observations to add to which track?

# Track Matching

Intuition: predict next position along each track.

Intuition: match should be close to predicted position.



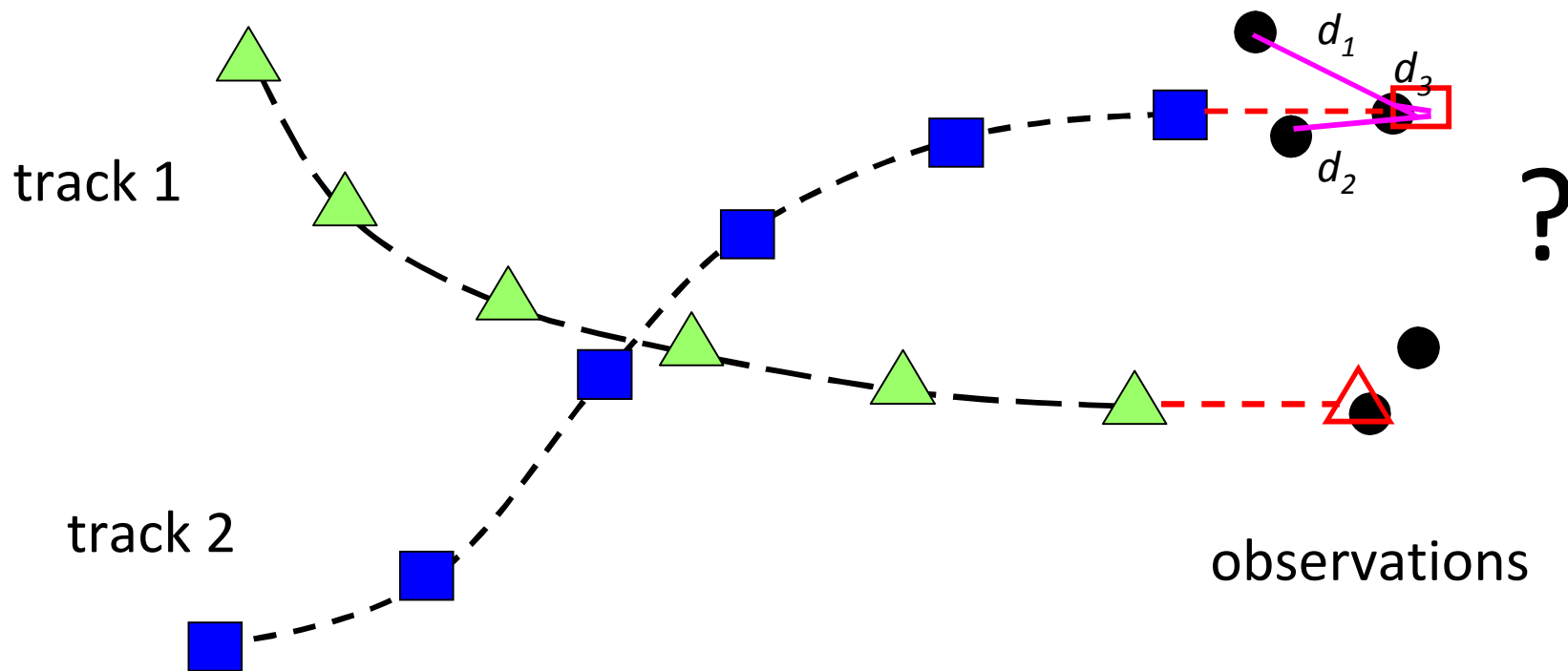
How to determine which observations to add to which track?

# Track Matching

Intuition: predict next position along each track.

Intuition: match should be close to predicted position.

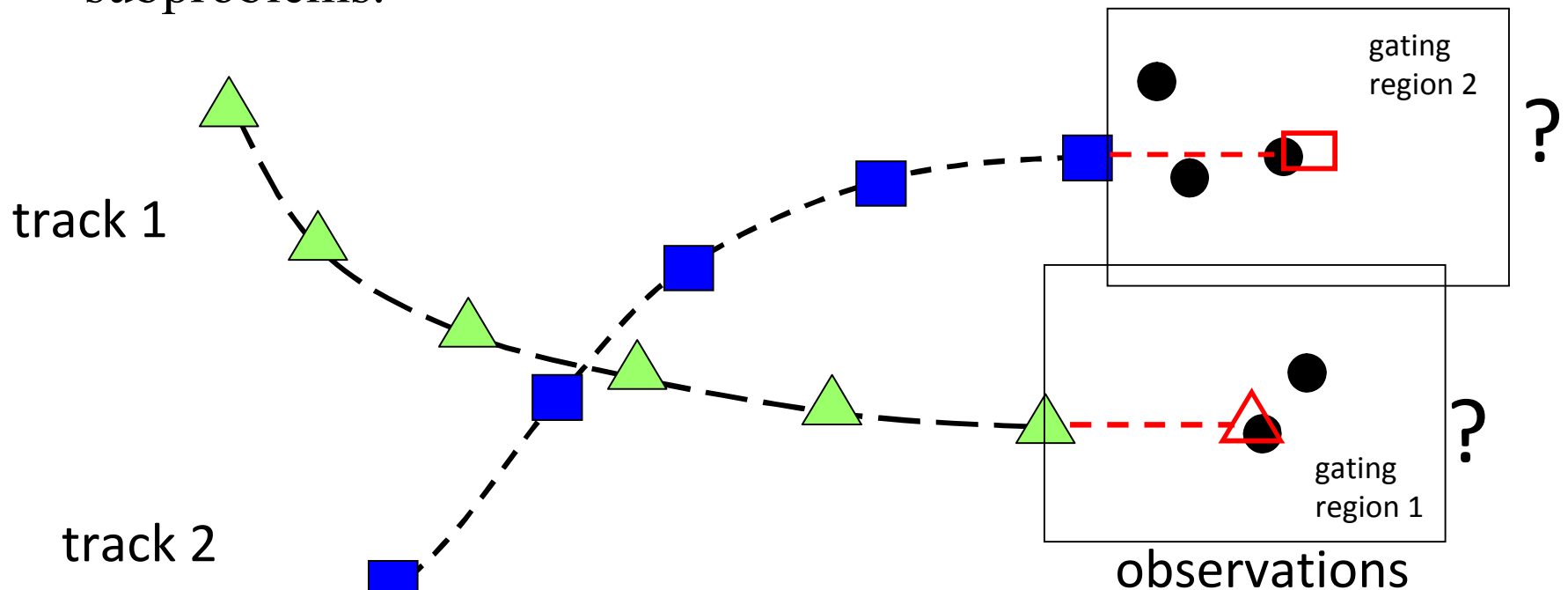
Intuition: some matches are highly unlikely.



How to determine which observations to add to which track?

# Gating

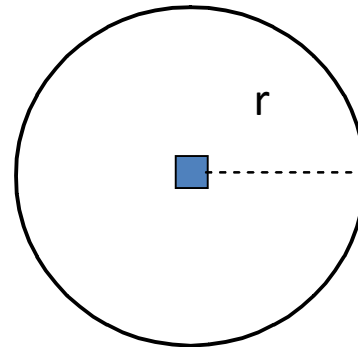
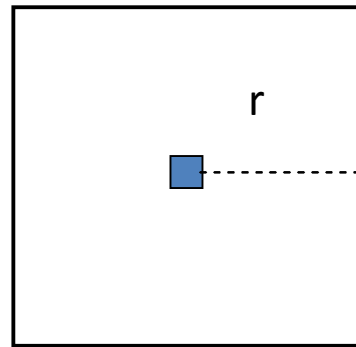
A method for pruning matches that are geometrically unlikely from the start. Allows us to decompose matching into smaller subproblems.



How to determine which observations to add to which track?

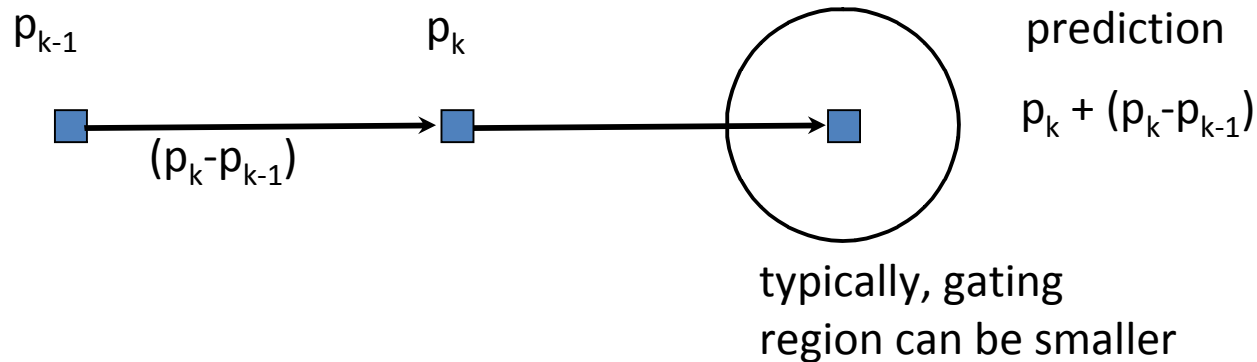
# Simple Prediction/Gating

Constant position + bound on maximum interframe motion



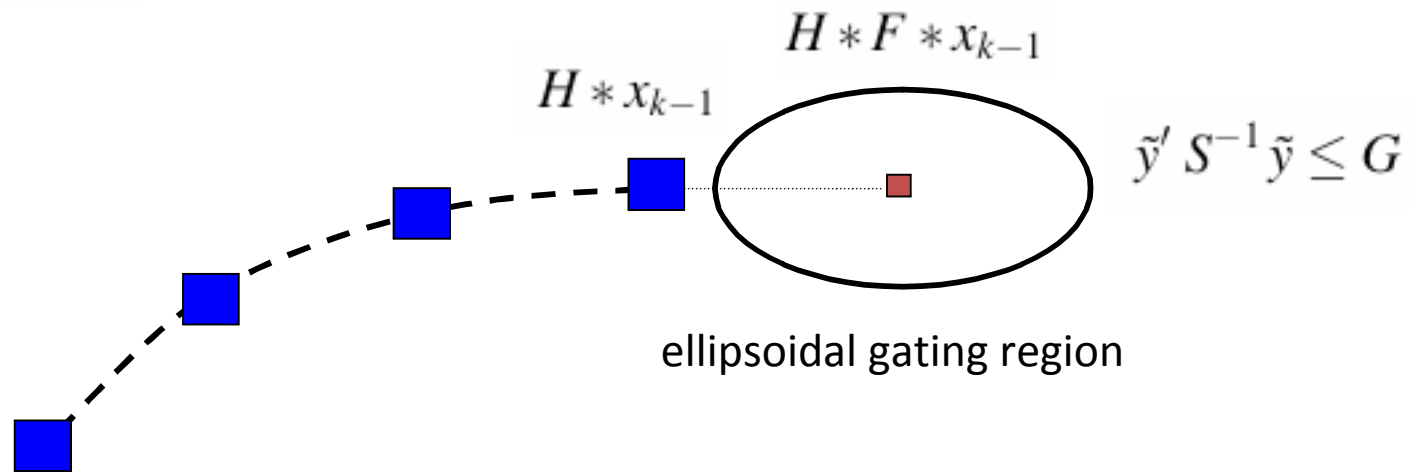
constant position  
prediction

Three-frame constant velocity prediction



# Kalman Filter Prediction/Gating

$$x = \begin{bmatrix} x \\ y \\ u \\ v \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \quad (\text{predicted state})$$

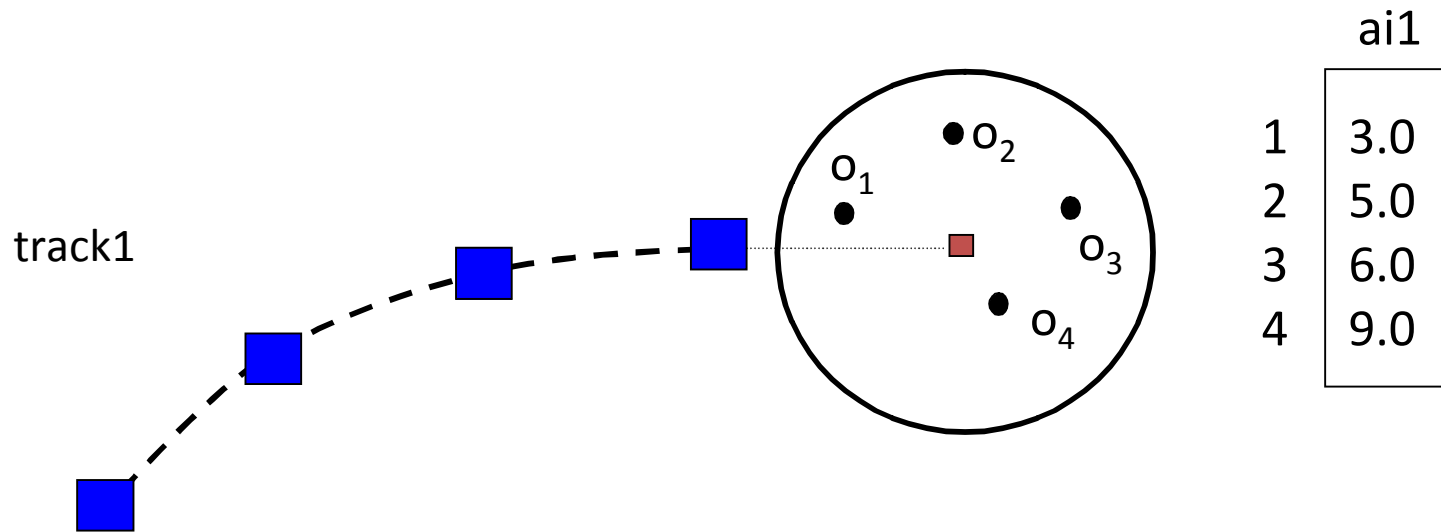
$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad (\text{predicted estimate covariance})$$

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad (\text{innovation or measurement residual})$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \quad (\text{innovation (or residual) covariance})$$

# Global Nearest Neighbor (GNN)

Evaluate each observation in track gating region. Choose “best” one to incorporate into track.



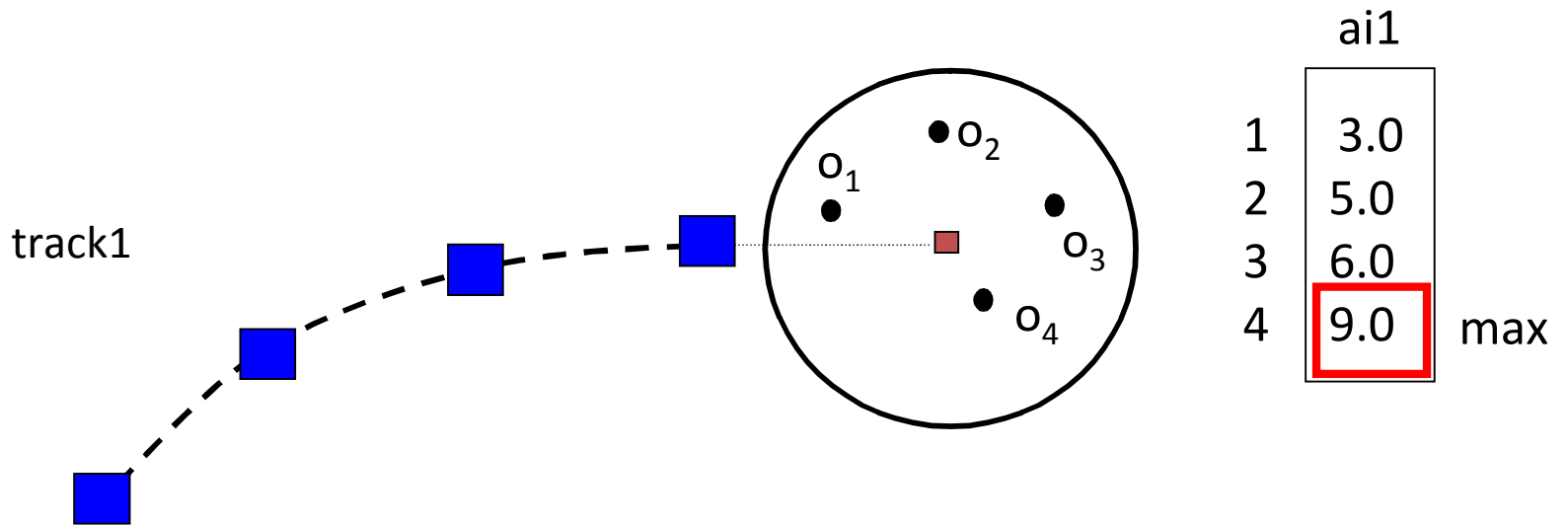
$a_{1j}$  = score for matching observation  $j$  to track 1

Could be based on Euclidean or Mahalanobis distance to predicted location (e.g.  $\exp\{-d^2\}$ ). Could be based on similarity of appearance (e.g. appearance template correlation score)



# Global Nearest Neighbor (GNN)

Evaluate each observation in track gating region. Choose “best” one to incorporate into track.

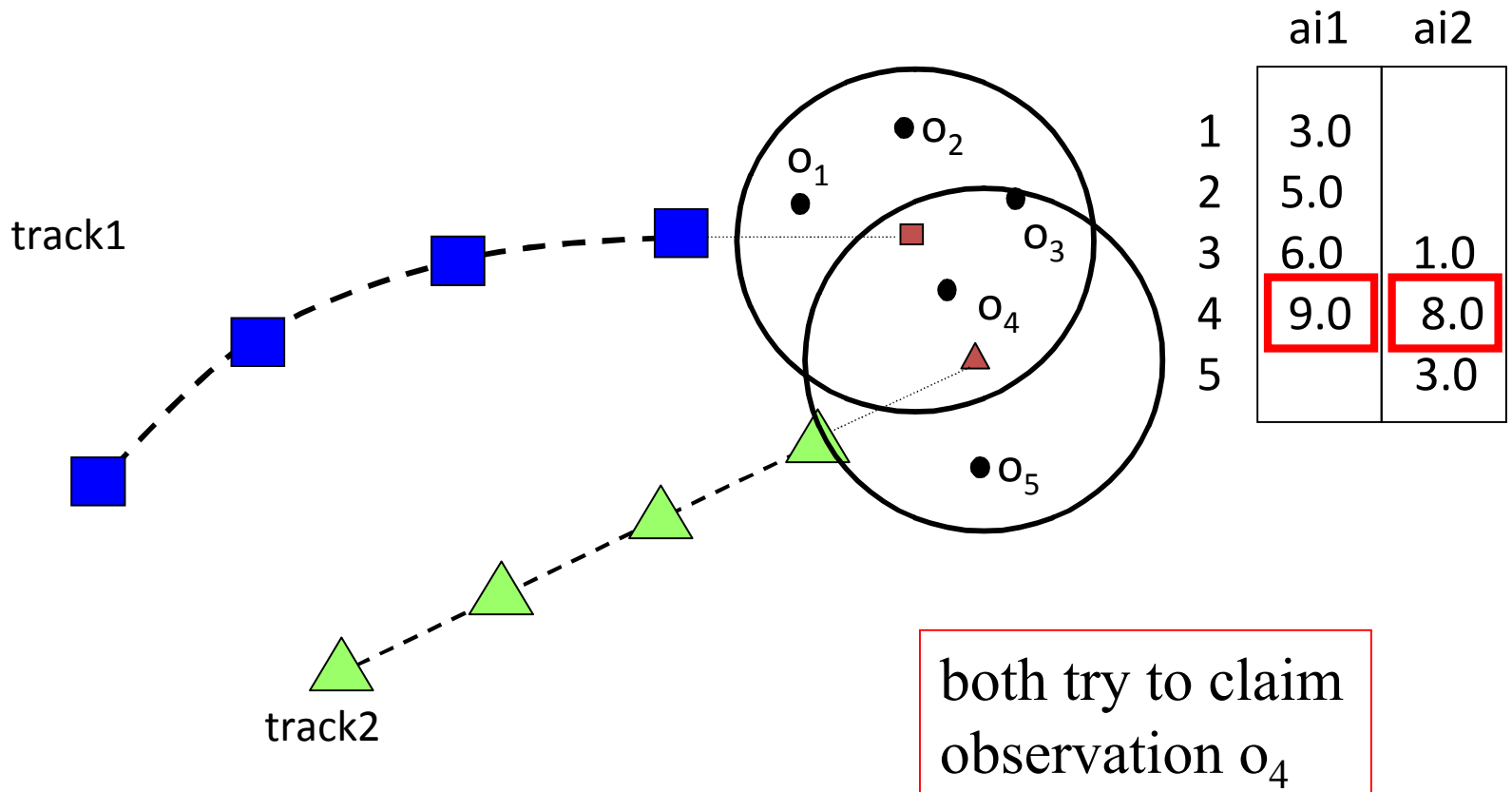


$a_{i1}$  = score for matching observation  $i$  to track 1

Choose best match  $a_{m1} = \max \{a_{11}, a_{21}, a_{31}, a_{41}\}$

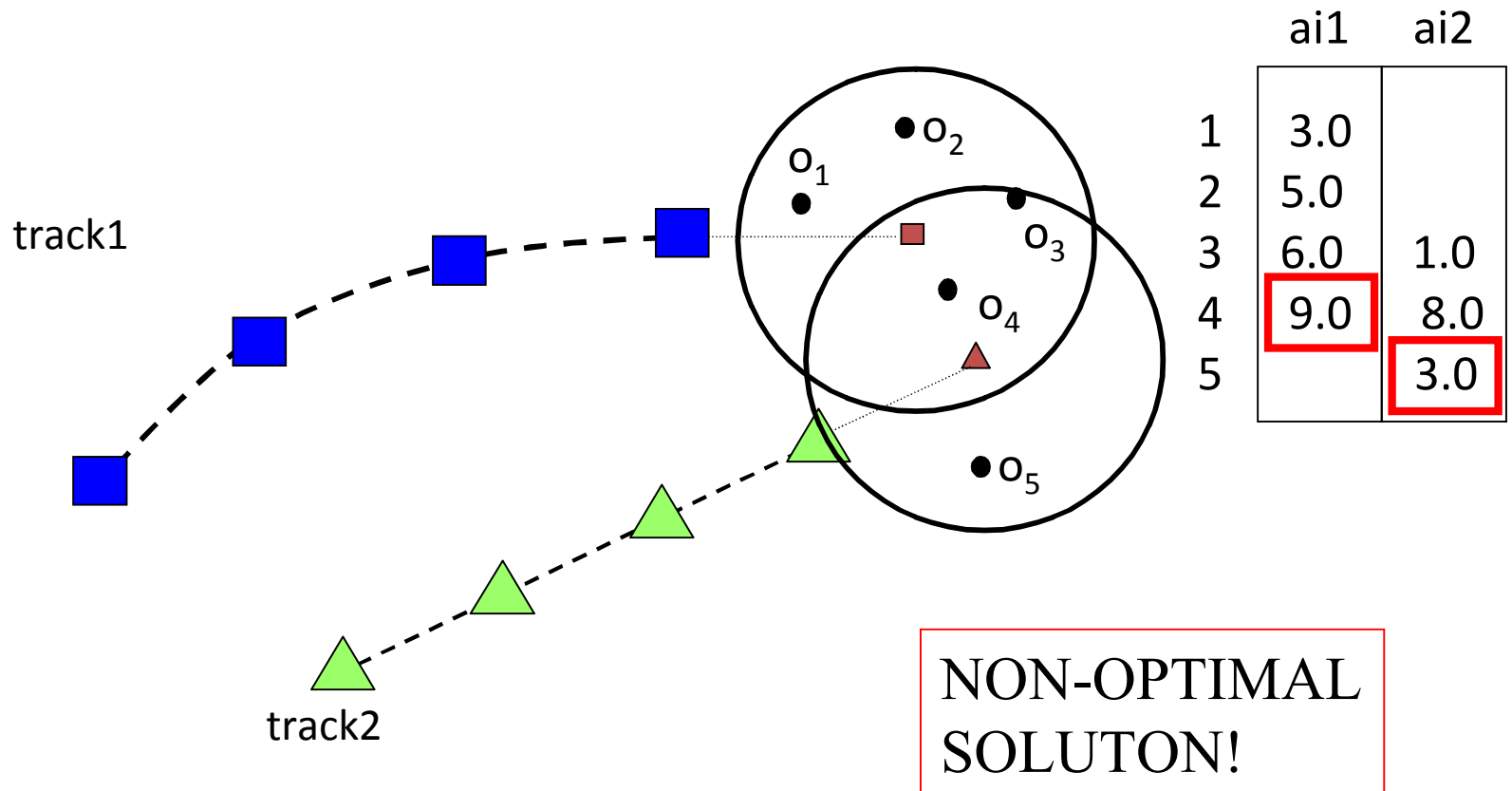
# Global Nearest Neighbor (GNN)

Problem: if do independently for each track, could end up with contention for the same observations.



# Greedy (Best First) Strategy

Assign observations to trajectories in decreasing order of goodness, making sure to not reuse an observation twice.



# Assignment Problem

Mathematical definition. Given an  $N \times N$  array of benefits  $\{X_{ai}\}$ , determine an  $N \times N$  permutation matrix  $M_{ai}$  that maximizes the total score:

$$\text{maximize: } E = \sum_{a=1}^N \sum_{i=1}^N M_{ai} X_{ai}$$

$$\text{subject to: } \left. \begin{array}{l} \forall i \sum_{a=1}^A M_{ai} = 1 \\ \forall a \sum_{i=1}^I M_{ai} = 1 \\ M_{ai} \in \{0, 1\} \end{array} \right\} \begin{array}{l} \text{constraints that say} \\ \text{M is a permutation matrix} \end{array}$$

The permutation matrix ensures that we can only choose one number from each row and from each column. (like assigning one worker to each job)

# Hungarian Algorithm

## Hungarian algorithm

---

From Wikipedia, the free encyclopedia

The **Hungarian algorithm** is a [combinatorial optimization algorithm](#) which solves [assignment problems](#) in [polynomial time](#) ( $O(n^3)$ ). The first version, known as the **Hungarian method**, was invented and published by [Harold Kuhn](#) in 1955. This was revised by [James Munkres](#) in 1957, and has been known since as the **Hungarian algorithm**, the **Munkres assignment algorithm**, or the **Kuhn-Munkres algorithm**. In 2006, it was discovered that [Carl Gustav Jacobi](#) had solved the assignment problem in the early 19th century, and published posthumously in 1890 in the Latin language.<sup>[1]</sup>

The algorithm developed by Kuhn was largely based on the earlier works of two [Hungarian](#) mathematicians: [Dénes König](#) and [Jenő Egerváry](#). The great advantage of Kuhn's method is that it is strongly [polynomial](#) (see [Computational complexity theory](#) for details). The main innovation of the algorithm was to combine two separate parts in Egerváry's proof into one.

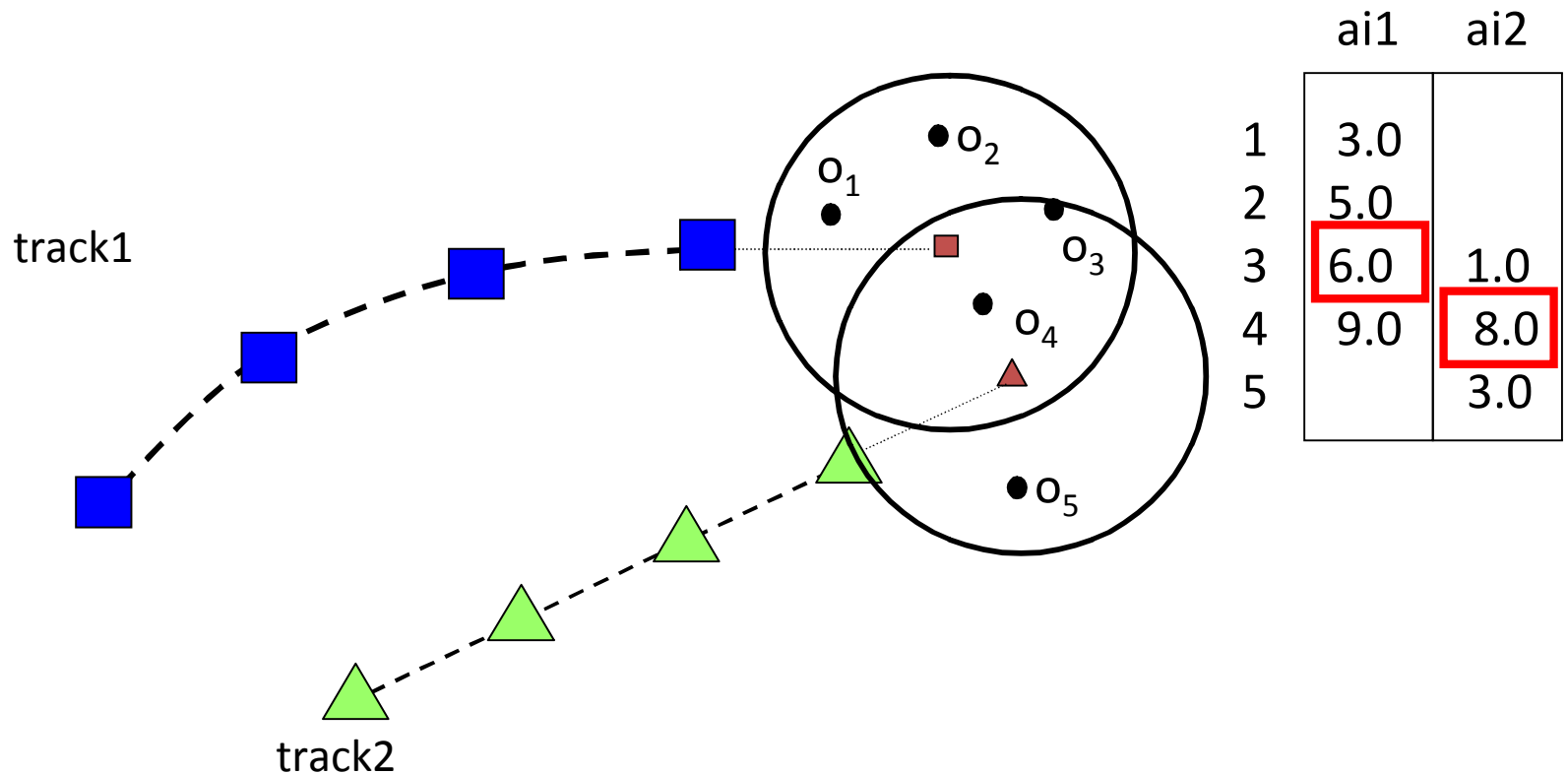


hence the name

# Result From Hungarian Algorithm

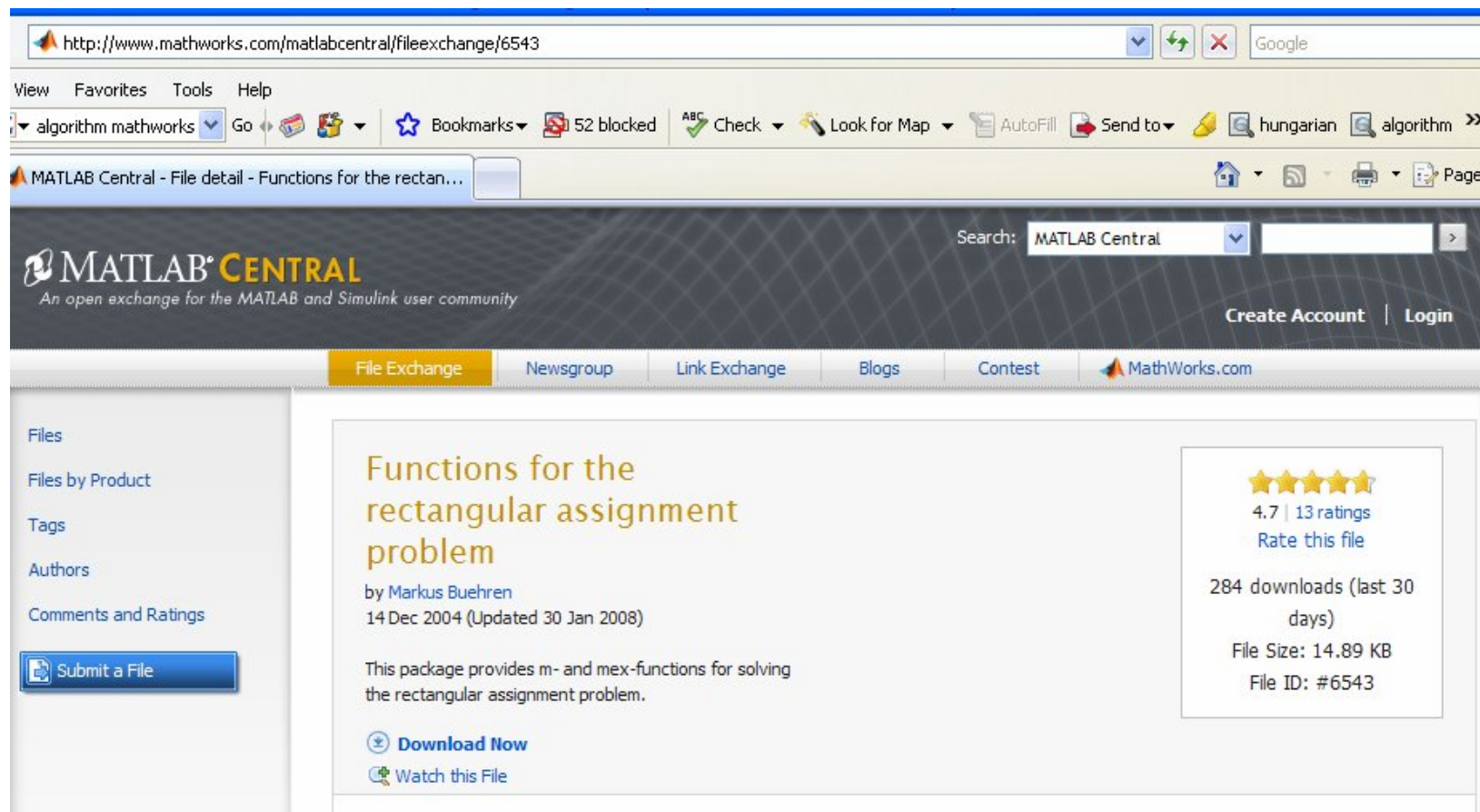
Each track is now forced to claim a different observation.

And we get the optimal assignment in this case.



# Handling Missing Matches

Typically, there will be a different number of tracks than observations. Some observations may not match any track. Some tracks may not have observations. That's OK. Most implementations of Hungarian Algorithm allow you to use a rectangular matrix, rather than a square matrix. See for example:



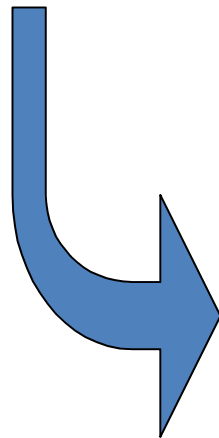
The screenshot shows a web browser window displaying the MATLAB Central website. The address bar shows the URL <http://www.mathworks.com/matlabcentral/fileexchange/6543>. The page title is "MATLAB Central - File detail - Functions for the rectan...". The main content area features the MATLAB Central logo and the text "An open exchange for the MATLAB and Simulink user community". A search bar contains the text "MATLAB Central". The page is categorized under "File Exchange". The main heading is "Functions for the rectangular assignment problem" by Markus Buehren, dated 14 Dec 2004 (Updated 30 Jan 2008). The description states: "This package provides m- and mex-functions for solving the rectangular assignment problem." The file has a rating of 4.7 (13 ratings) and 284 downloads (last 30 days). The file size is 14.89 KB and the file ID is #6543. There are buttons for "Download Now" and "Watch this File".

# If Square Matrix is Required...

	track1	track2	
1	3.0	0	5x3
2	5.0	0	
3	6.0	1.0	
4	9.0	8.0	
5	0	3.0	

pad with array of small random numbers to get a square score matrix.

Square-matrix  
assignment



	track1	track2	
1	0	0	5x3
2	0	0	
3	1	0	
4	0	1	
5	0	0	

ignore whatever happens in here



# More Sophisticated DA Approaches

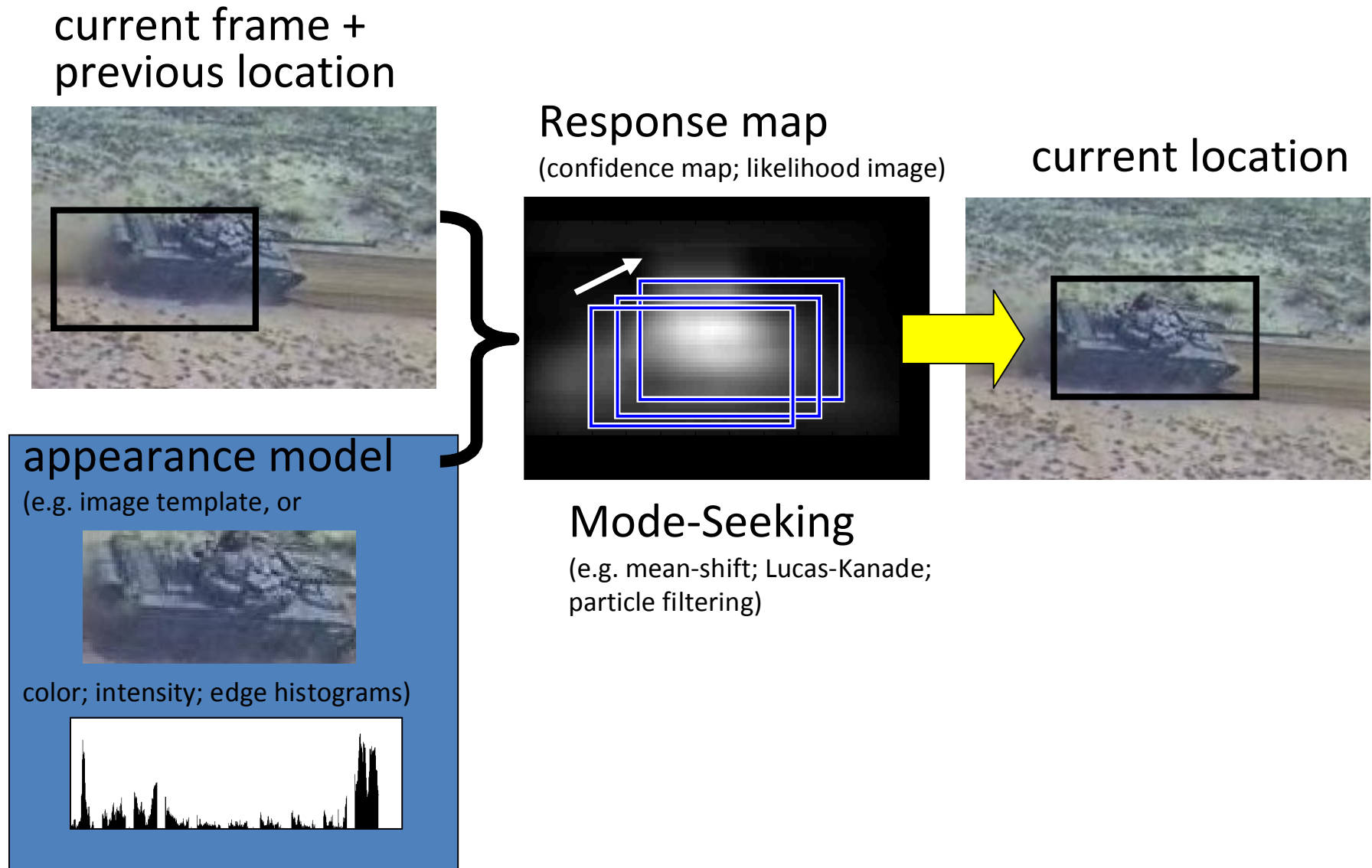
(that we won't be covering)

- Probabilistic Data Association (PDAF)
- Joint Probabilistic Data Assoc (JPDAF)
- Multi-Hypothesis Tracking (MHT)
- Markov Chain Monte Carlo DA (MCMCDA)

# Lecture Outline

- Brief Intro to Tracking
- Appearance-based Tracking
- Online Adaptation (learning)

# Appearance-Based Tracking



# Relation to Bayesian Filtering

In appearance-based tracking, data association tends to be reduced to gradient ascent (hill-climbing) on an appearance similarity response function.

Motion prediction model tends to be simplified to assume constant position + noise (so assumes previous bounding box significantly overlaps object in the new frame).

# Appearance Models

want to be invariant, or at least resilient, to changes in  
photometry (e.g. brightness; color shifts)  
geometry (e.g. distance; viewpoint; object deformation)

Simple Examples:

histograms or parzen estimators.

photometry

coarsening of bins in histogram

widening of kernel in parzen estimator

geometry

invariant to rigid and nonrigid deformations;

resilient to blur, resolution.

invariant to arbitrary permutation of pixels! (drawback)

# Appearance Models

Simple Examples (continued):

Intensity Templates

- photometry

  - normalization (e.g. NCC)

  - use gradients instead of raw intensities

- geometry

  - couple with estimation of geometric warp parameters

Other “flexible” representations are possible, e.g. spatial constellations of templates or color patches.

Actually, any representation used for object detection can be adapted for tracking. Run time is important, though.

# Template Methods

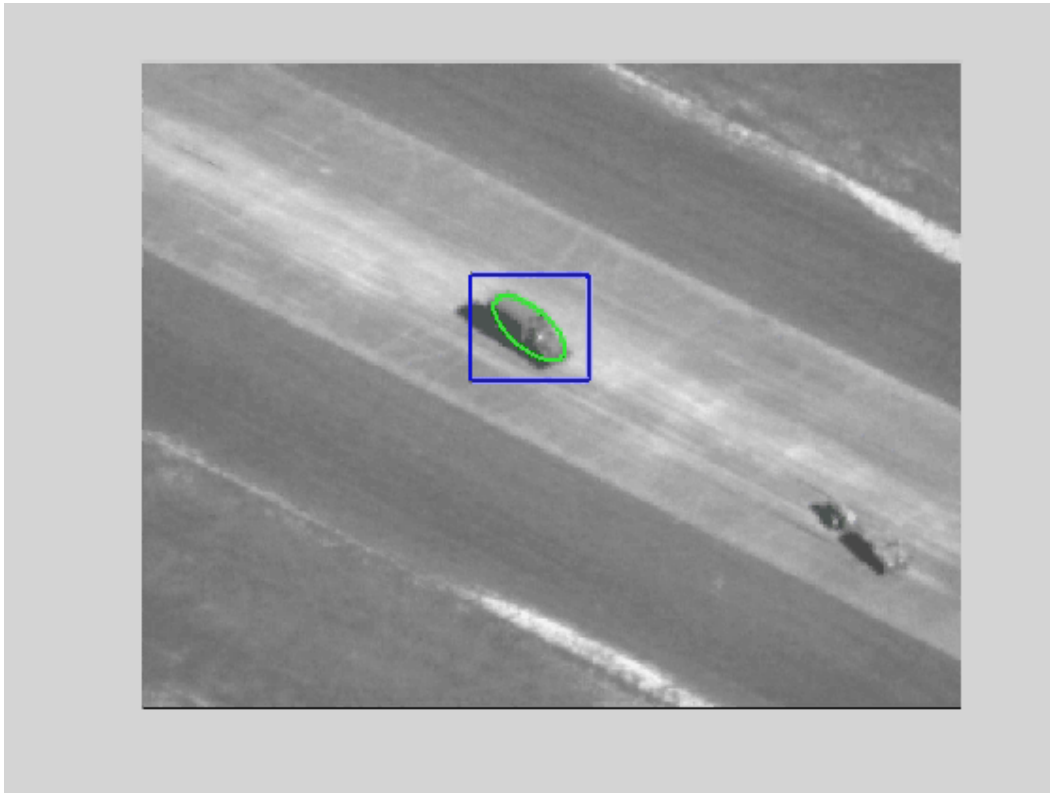
Simplest example is correlation-based template tracking.

Assumptions:

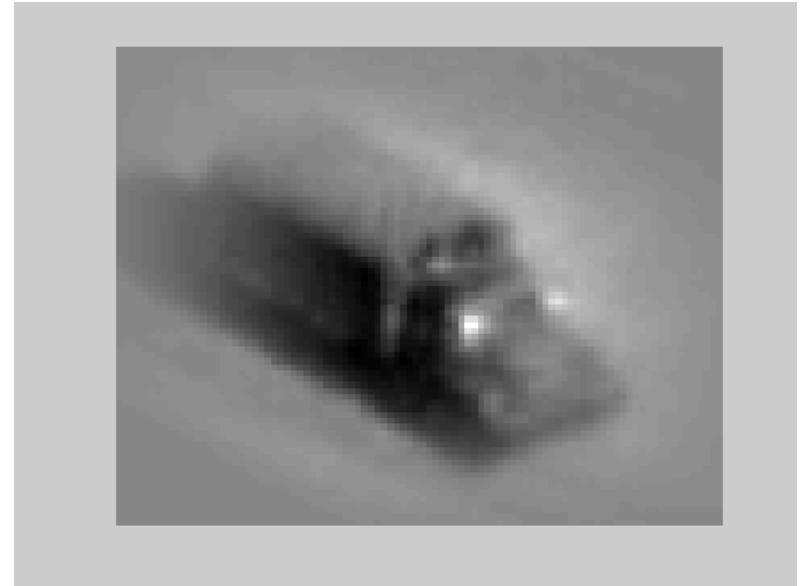
- a cropped image of the object from the first frame can be used to describe appearance
- object will look nearly identical in each new image (note: we can use normalized cross correlation to add some resilience to lighting changes.
- movement is nearly pure 2D translation

# Normalized Correlation, Fixed Template

Current tracked location



Fixed template



Failure mode: Unmodeled Appearance Change



# Naive Approach to Handle Change

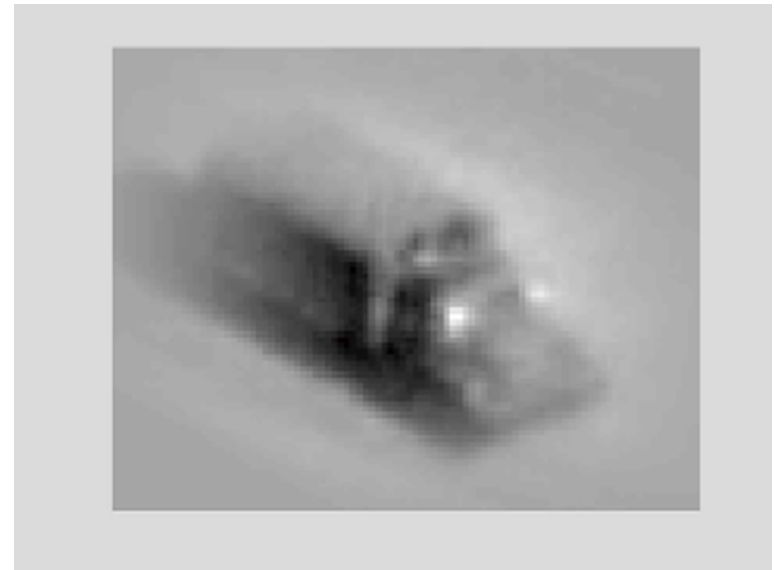
- One approach to handle changing appearance over time is adaptive template update
- One you find location of object in a new frame, just extract a new template, centered at that location
- What is the potential problem?

# Normalized Correlation, Adaptive Template

Current tracked location

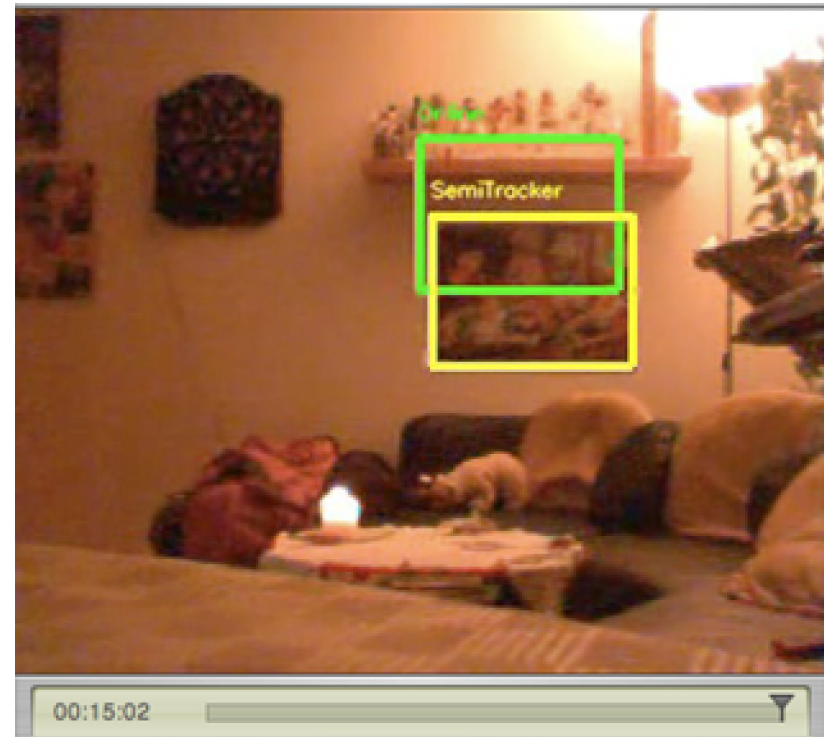
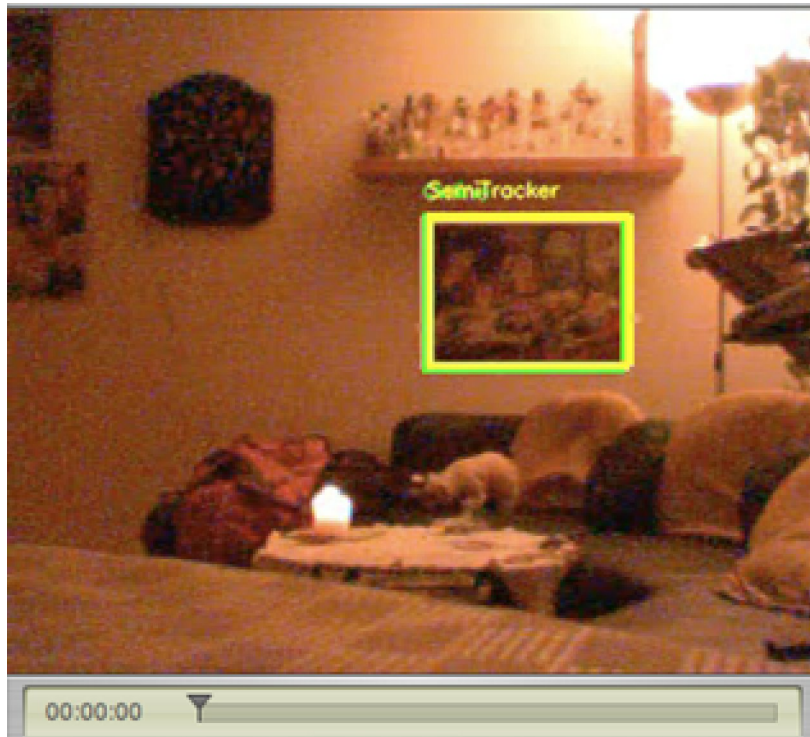


Current template



The result is even worse than before!

# Drift is a Universal Problem!



1 hour

Example courtesy of Horst Bischof. Green: online boosting tracker; yellow: drift-avoiding “semisupervised boosting” tracker (we will discuss it later today).

# Template Drift

- If your estimate of template location is slightly off, you are now looking for a matching position that is similarly off center.
- Over time, this offset error builds up until the template starts to “slide” off the object.
- The problem of drift is a major issue with methods that adapt to changing object appearance.

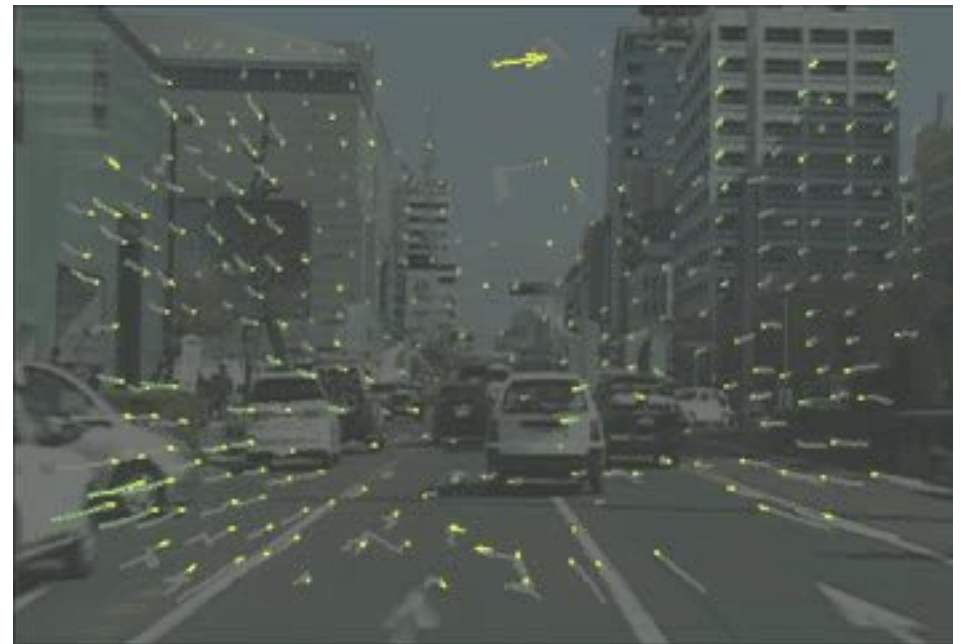
# Lucas-Kanade Tracking

The Lucas-Kanade algorithm is a template tracker that works by gradient ascent (hill-climbing).

Originally developed to compute translation of small image patches (e.g. 5x5) to measure optical flow.

KLT algorithm is a good (and free) implementation for tracking corner features.

Over short time periods (a few frames), drift isn't really an issue.



# Lucas-Kanade Tracking

Assumption of constant flow (pure translation) for all pixels in a large template is unreasonable.



However, the Lucas-Kanade approach easily generalizes to other 2D parametric motion models (like affine or projective).

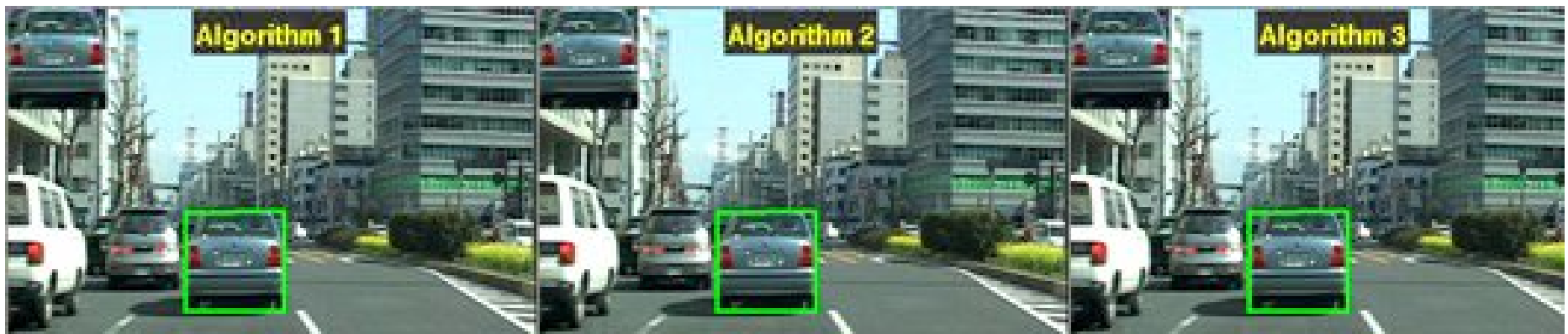
$$\mathbf{p}_n = \arg \min_{\mathbf{p}} \sum_{\mathbf{x} \in T_n} [I_n(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T_n(\mathbf{x})]^2$$

See a series of papers called “Lucas-Kanade 20 Years On”, by Baker and Matthews.

# Lucas-Kanade Tracking

As with correlation tracking, if you use fixed appearance templates or naively update them, you run into problems.

Matthews, Ishikawa and Baker, The Template Update Problem, PAMI 2004, propose a template update scheme.



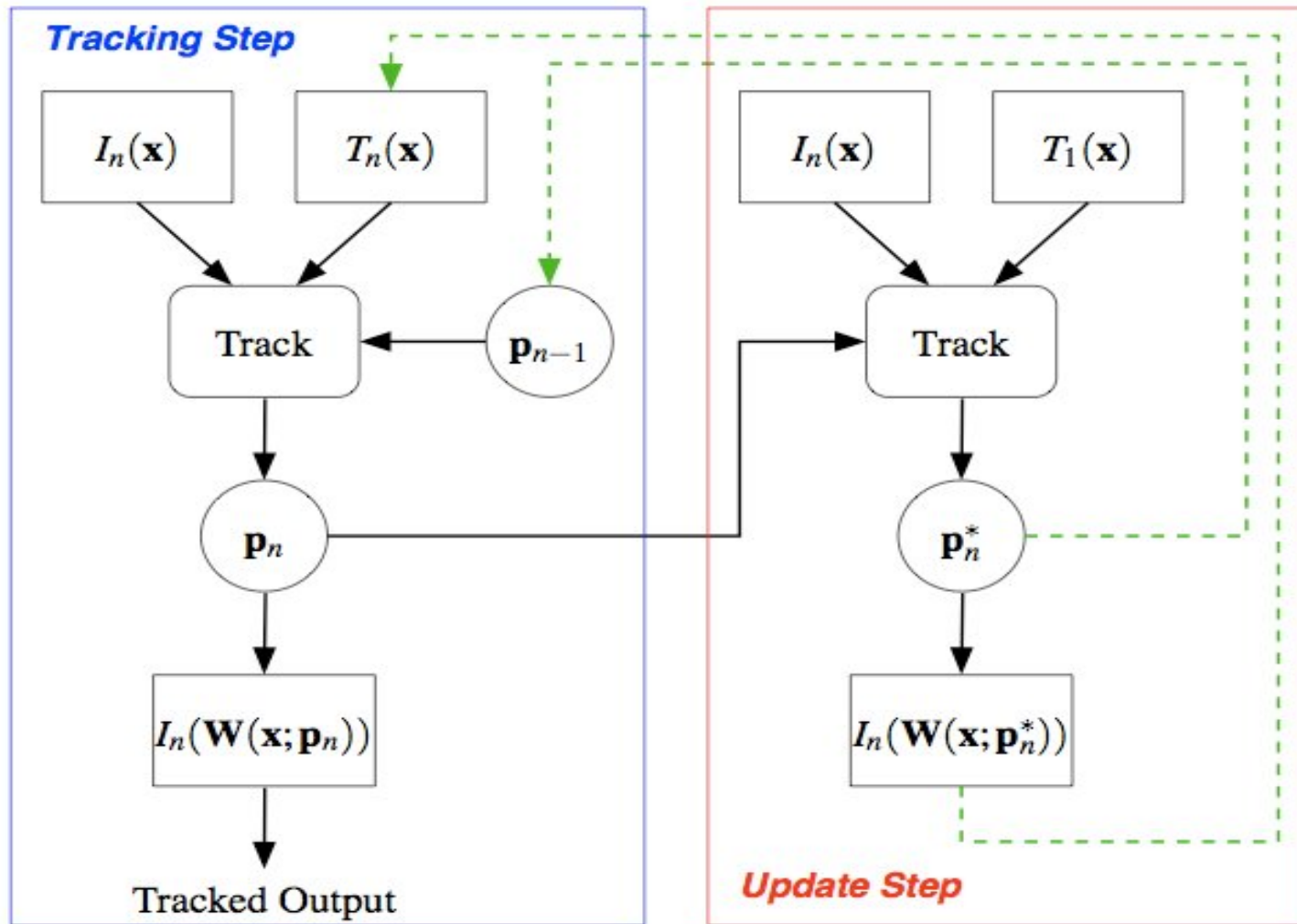
Fixed template

Naïve update

Their update



# Template Update with Drift Correction



If  $\|\mathbf{p}_n^* - \mathbf{p}_n\| \leq \varepsilon$  then  $T_{n+1}(\mathbf{x}) = I_n(\mathbf{W}(\mathbf{x}; \mathbf{p}_n^*))$

else  $T_{n+1}(\mathbf{x}) = T_n(\mathbf{x})$

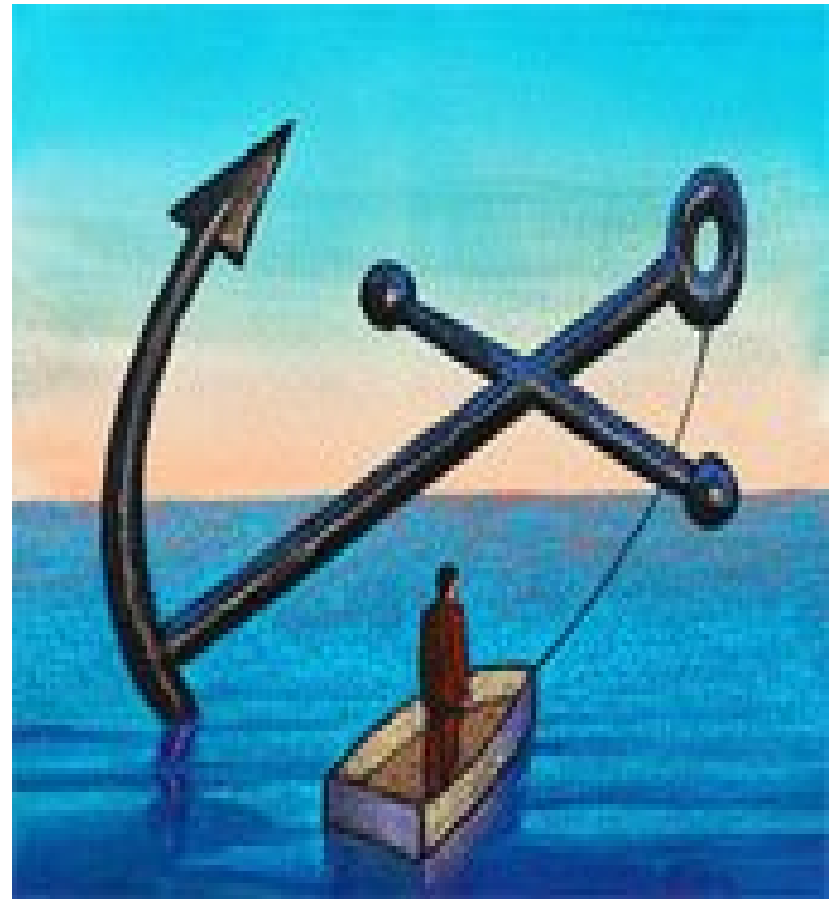


# Anchoring Avoids Drift

This is an example of a general strategy for drift avoidance that we'll call "anchoring".

The key idea is to make sure you don't stray too far from your initial appearance model.

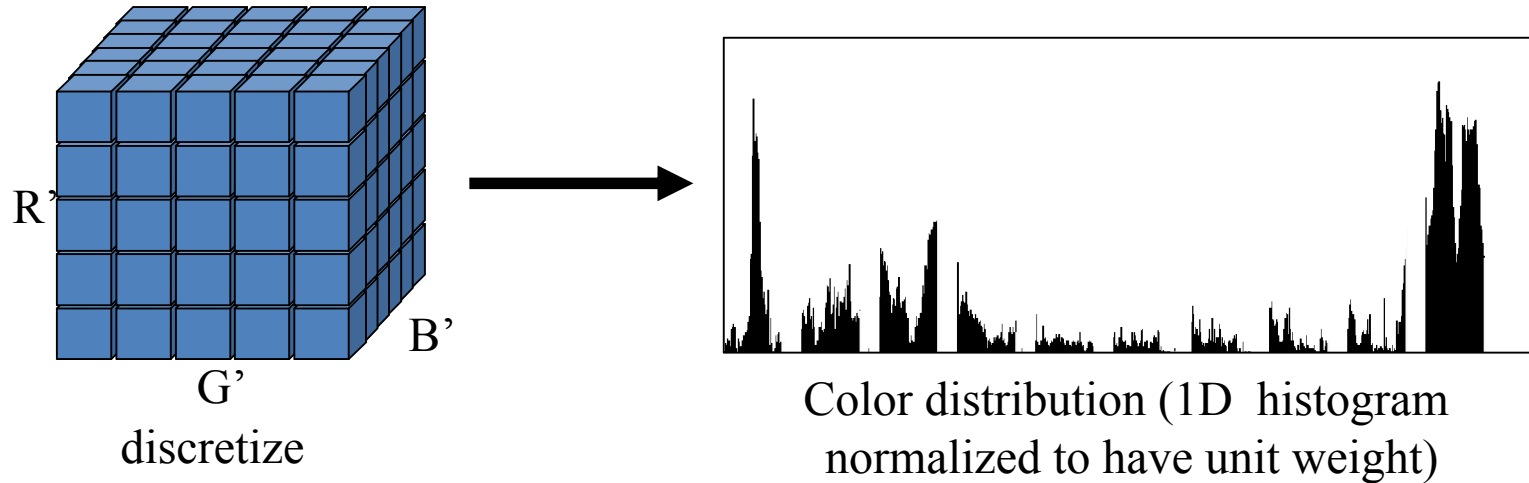
Potential drawbacks?  
[answer: You cannot accommodate very LARGE changes in appearance.]



# Histogram Appearance Models

- Motivation – to track non-rigid objects, (like a walking person), it is hard to specify an explicit 2D parametric motion model.
- Appearances of non-rigid objects can sometimes be modeled with color distributions
- NOT limited to only color. Could also use edge orientations, texture, motion...

# Appearance via Color Histograms



$$R' = R \ll (8 - \text{nbits})$$

$$G' = G \ll (8 - \text{nbits})$$

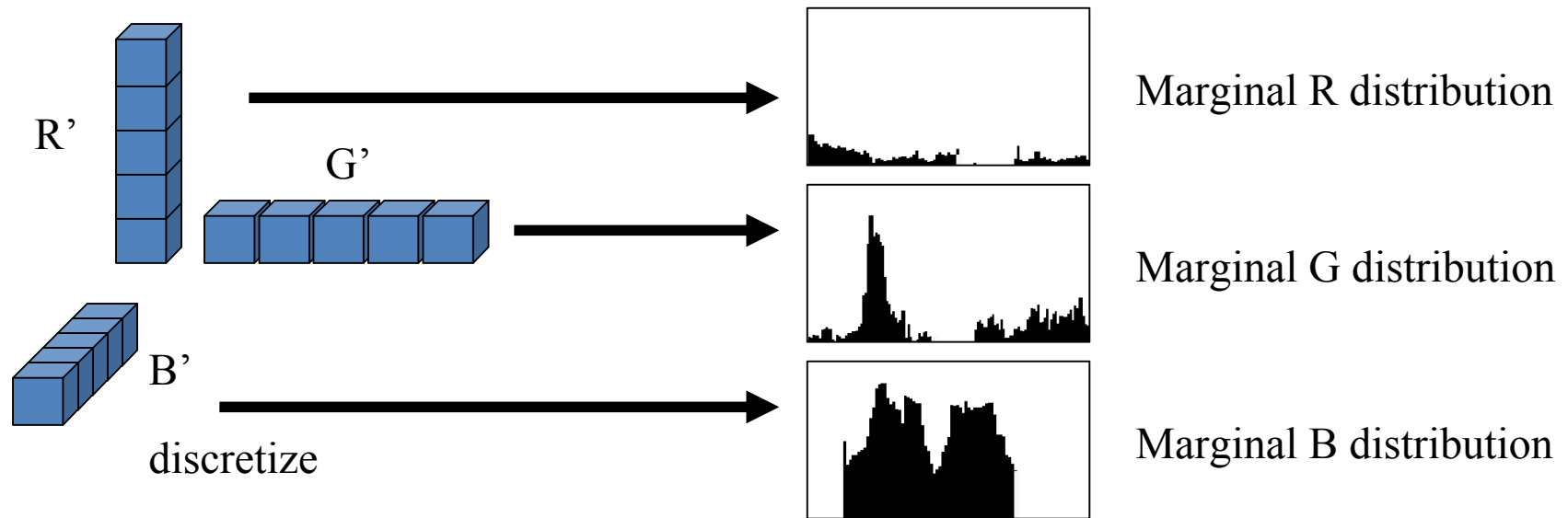
$$B' = B \ll (8 - \text{nbits})$$

Total histogram size is  $(2^{(8-\text{nbits})})^3$

example, 4-bit encoding of R,G and B channels yields a histogram of size  $16*16*16 = 4096$ .

# Smaller Color Histograms

Histogram information can be much much smaller if we are willing to accept a loss in color resolvability.



$$R' = R \ll (8 - \text{nbits})$$

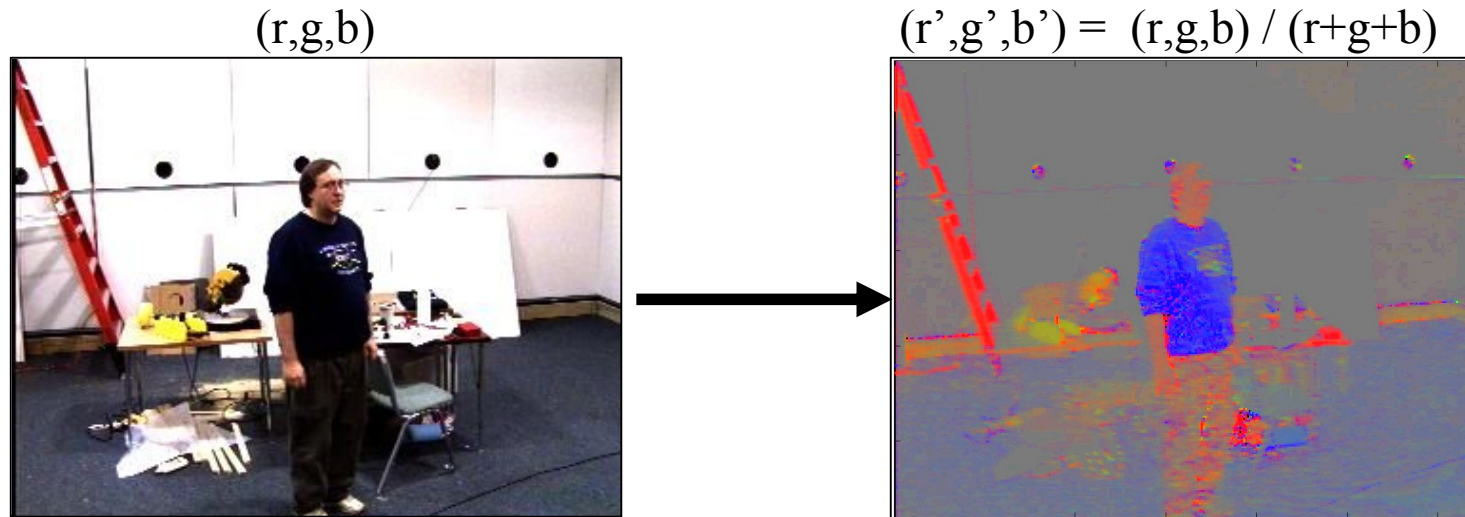
$$G' = G \ll (8 - \text{nbits})$$

$$B' = B \ll (8 - \text{nbits})$$

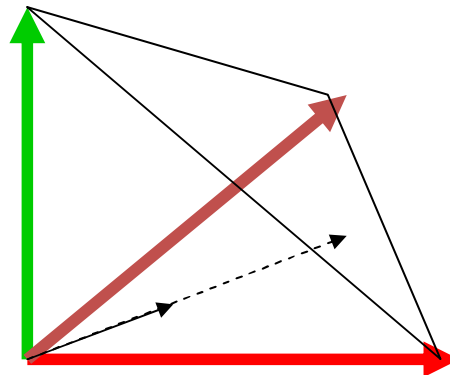
Total histogram size is  $3 \cdot (2^{(8-\text{nbits})})$

example, 4-bit encoding of R,G and B channels yields a histogram of size  $3 \cdot 16 = 48$ .

# Normalized Color



Normalized color divides out pixel luminance (brightness), leaving behind only chromaticity (color) information. The result is less sensitive to variations due to illumination/shading.

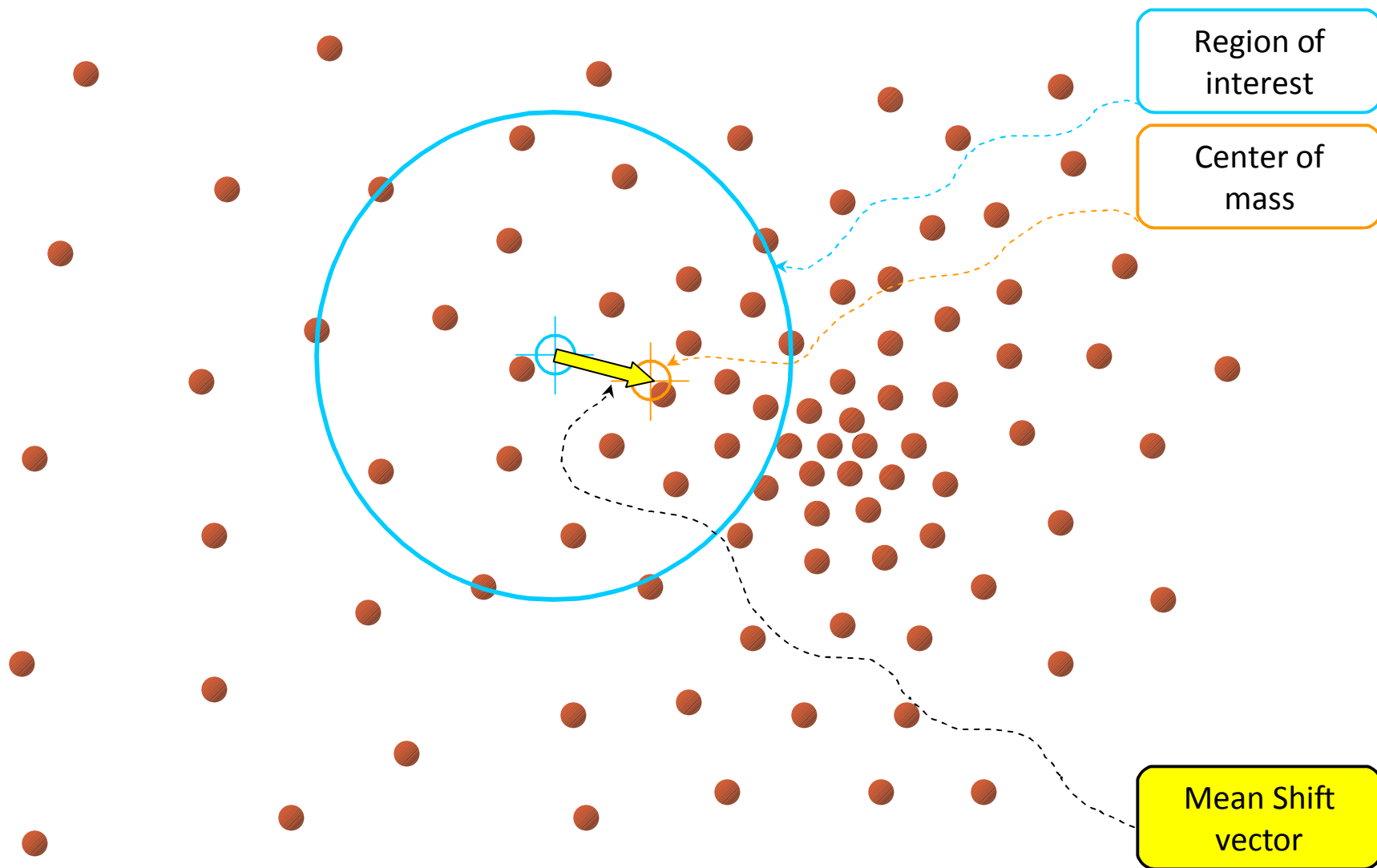


# Mean-Shift

Mean-shift is a hill-climbing algorithm that seeks modes of a nonparametric density represented by samples and a kernel function.

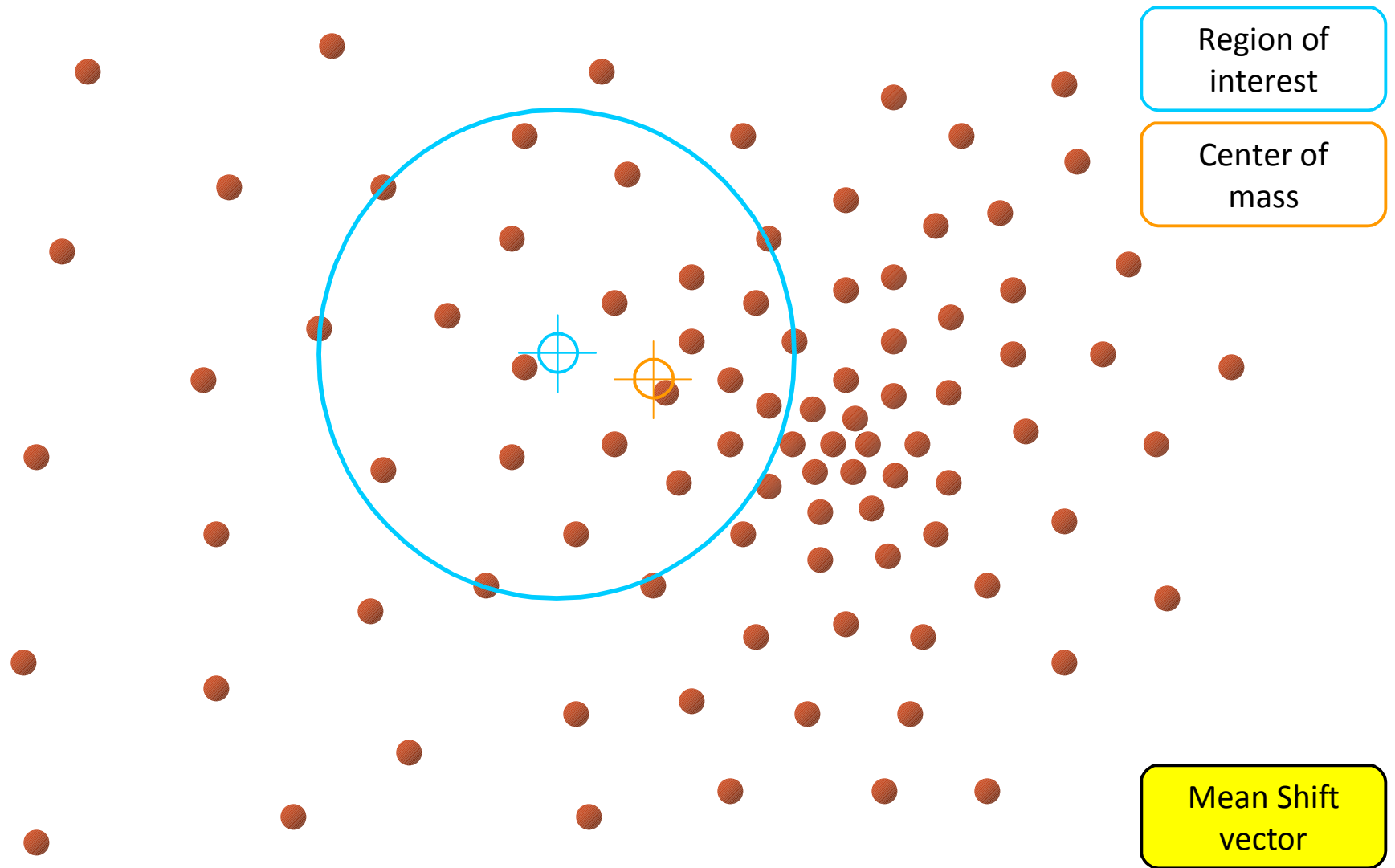
It is often used for tracking when a histogram-based appearance model is used. But it could be used just as well to search for modes in a template correlation surface.

# Intuitive Description



**Objective : Find the densest region**

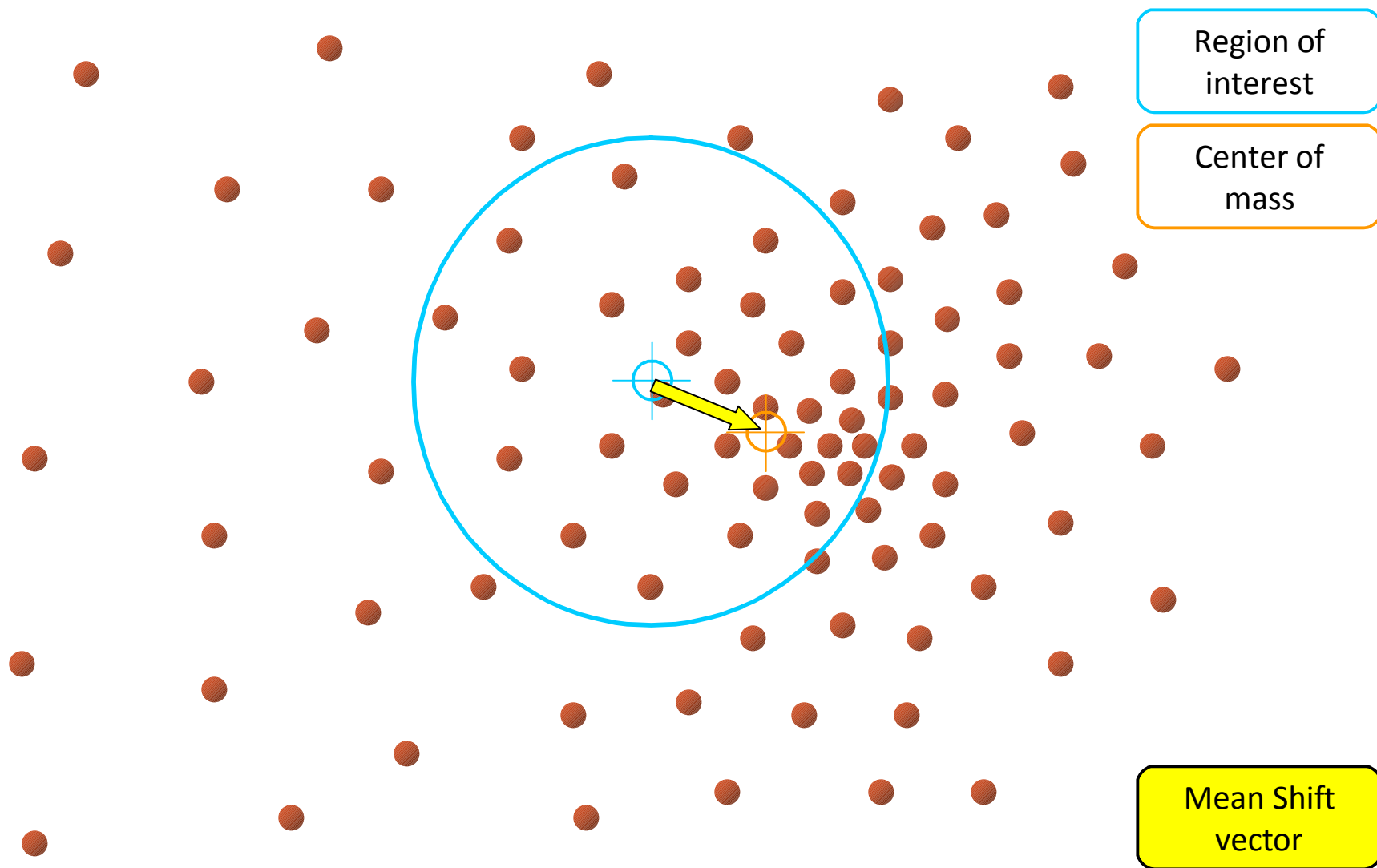
# Intuitive Description



Objective : Find the densest region

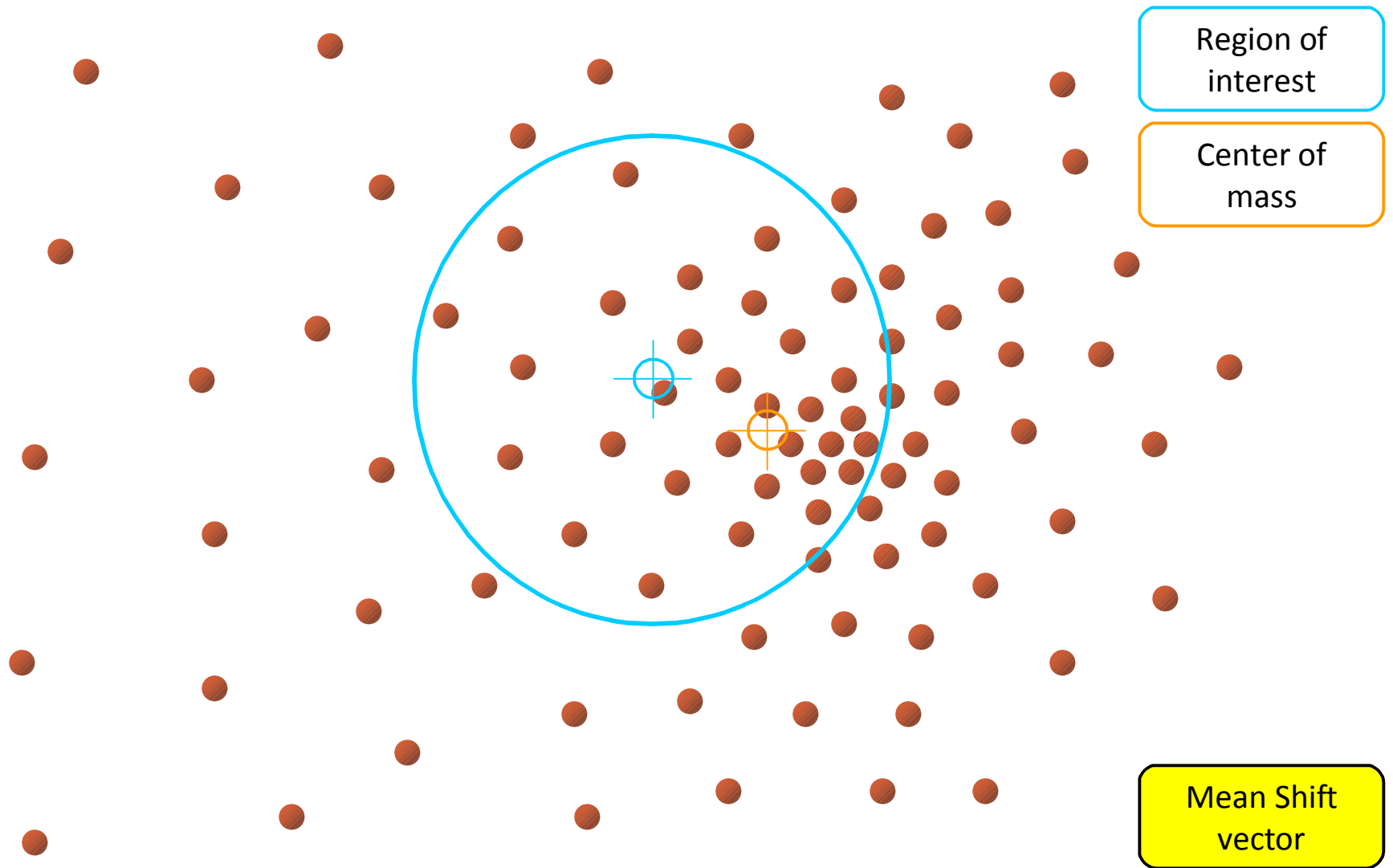


# Intuitive Description



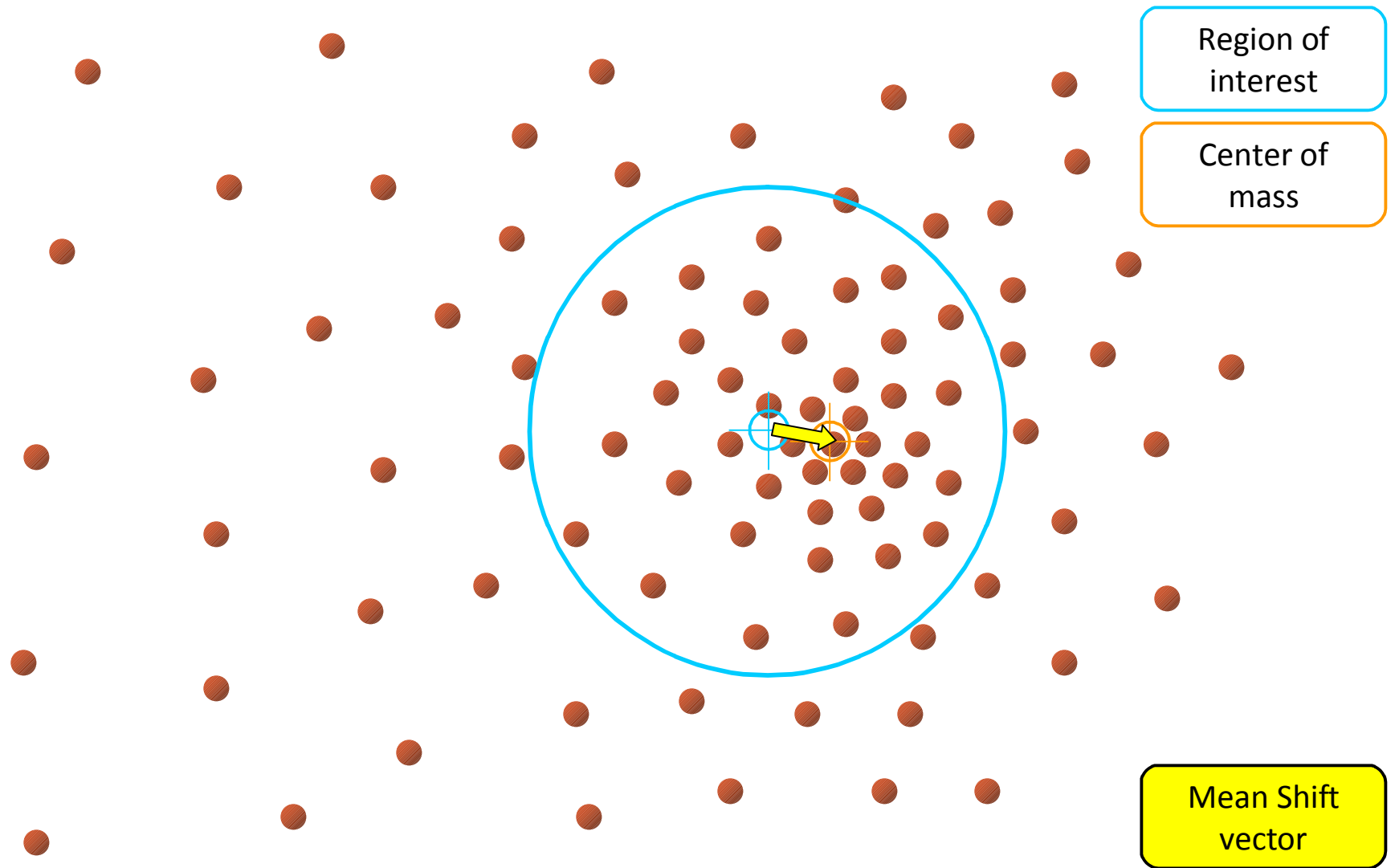
**Objective : Find the densest region**

# Intuitive Description



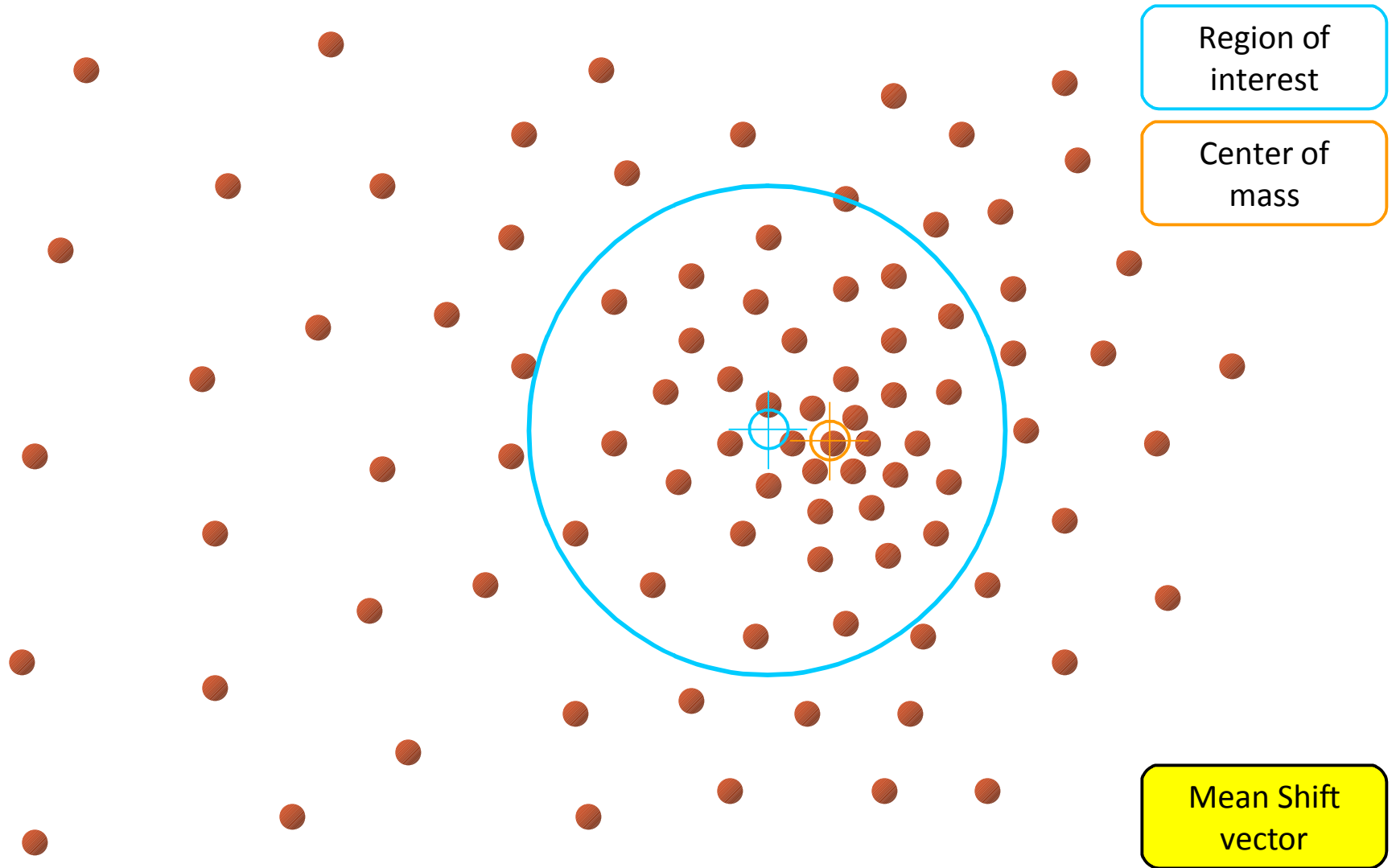
Objective : Find the densest region

# Intuitive Description



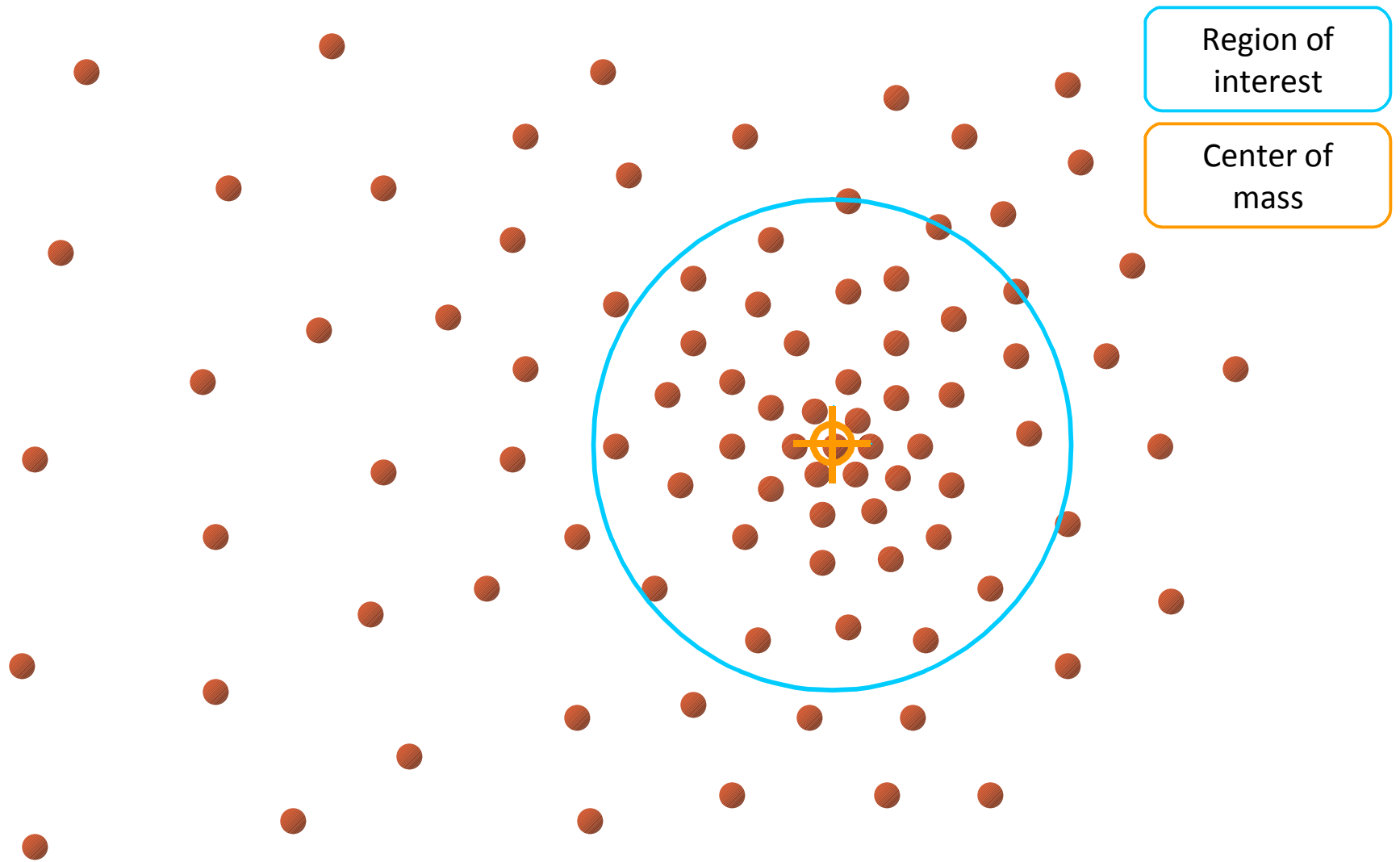
**Objective : Find the densest region**

# Intuitive Description



Objective : Find the densest region

# Intuitive Description



Objective : Find the densest region

# Mean-Shift Tracking

## Two predominant approaches:

- 1) Weight images: Create a response map with pixels weighted by “likelihood” that they belong to the object being tracked. Perform mean-shift on it.
- 2) Histogram comparison: Weight image is implicitly defined by a similarity measure (e.g. Bhattacharyya coefficient) comparing the model distribution with a histogram computed inside the current estimated bounding box. [Comaniciu, Ramesh and Meer]

# Mean-shift on Weight Images

Ideally, we want an indicator function that returns 1 for pixels on the object we are tracking, and 0 for all other pixels

In practice, we compute response maps where the value at a pixel is roughly proportional to the likelihood that the pixel comes from the object we are tracking.

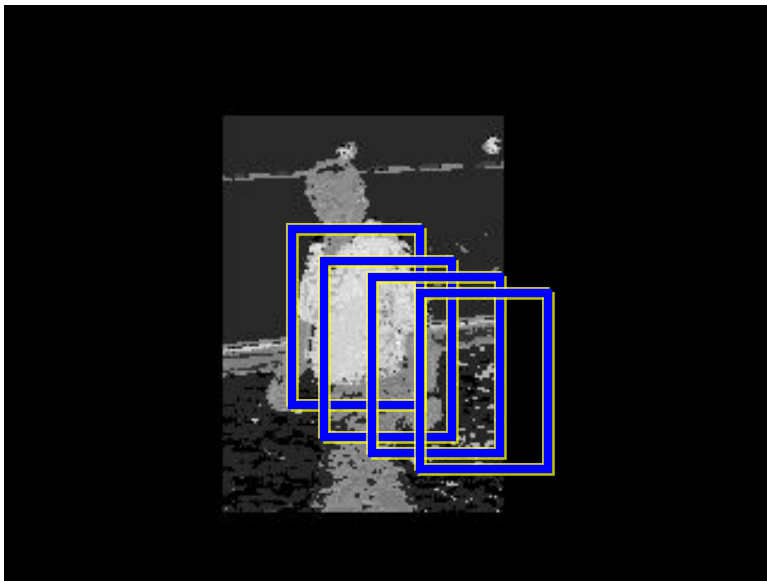
Computation of likelihood can be based on

- color
- texture
- shape (boundary)
- predicted location
- classifier outputs



# Mean-Shift on Weight Images

The pixels form a uniform grid of data points, each with a weight (pixel value). Perform standard mean-shift algorithm using this weighted set of points.



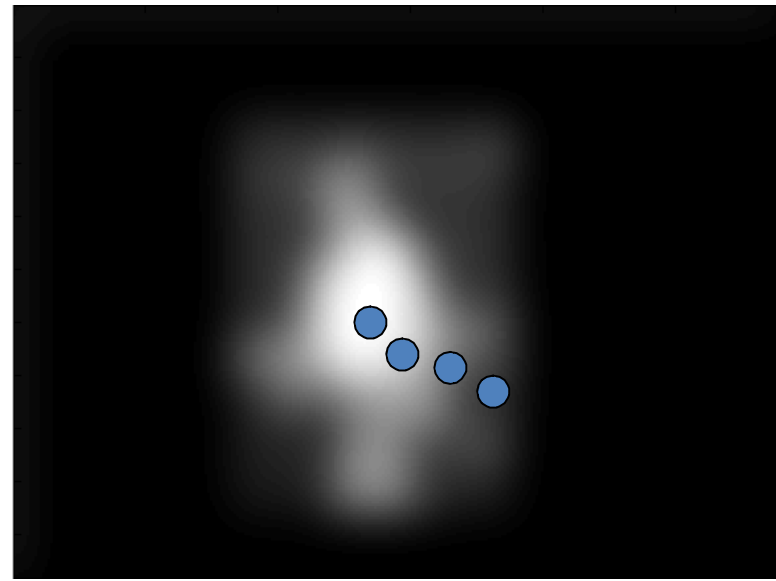
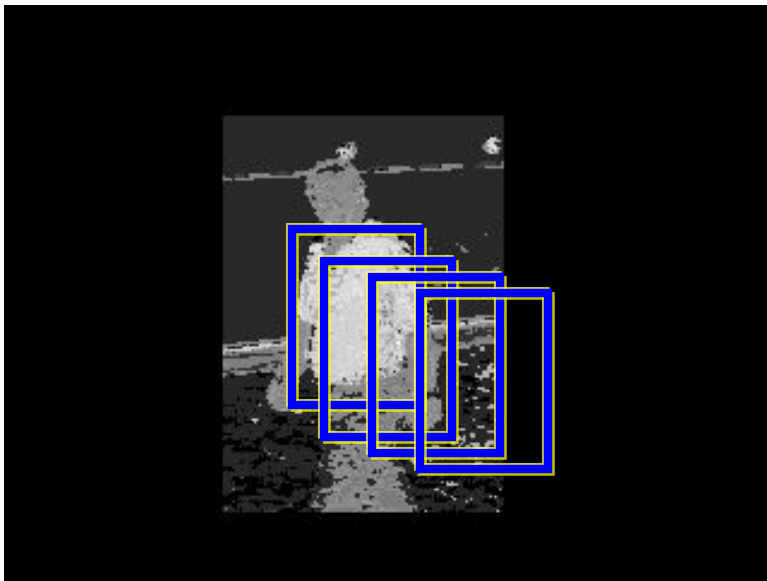
$$\Delta \mathbf{x} = \frac{\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x}) w(\mathbf{a}) (\mathbf{a}-\mathbf{x})}{\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x}) w(\mathbf{a})}$$

$\mathbf{K}$  is a smoothing kernel  
(e.g. uniform or Gaussian)



# Nice Property

Running mean-shift with kernel  $K$  on weight image  $w$  is equivalent to performing gradient ascent in a (virtual) image formed by convolving  $w$  with some “shadow” kernel  $H$ .



The algorithm is performing hill-climbing on an implicit density function determined by Parzen estimation with kernel  $H$ .

# Mean-Shift Tracking

Some examples.



Gary Bradski, CAMSHIFT



Comaniciu, Ramesh and  
Meer, CVPR 2000  
(Best paper award)

# Mean-Shift Tracking

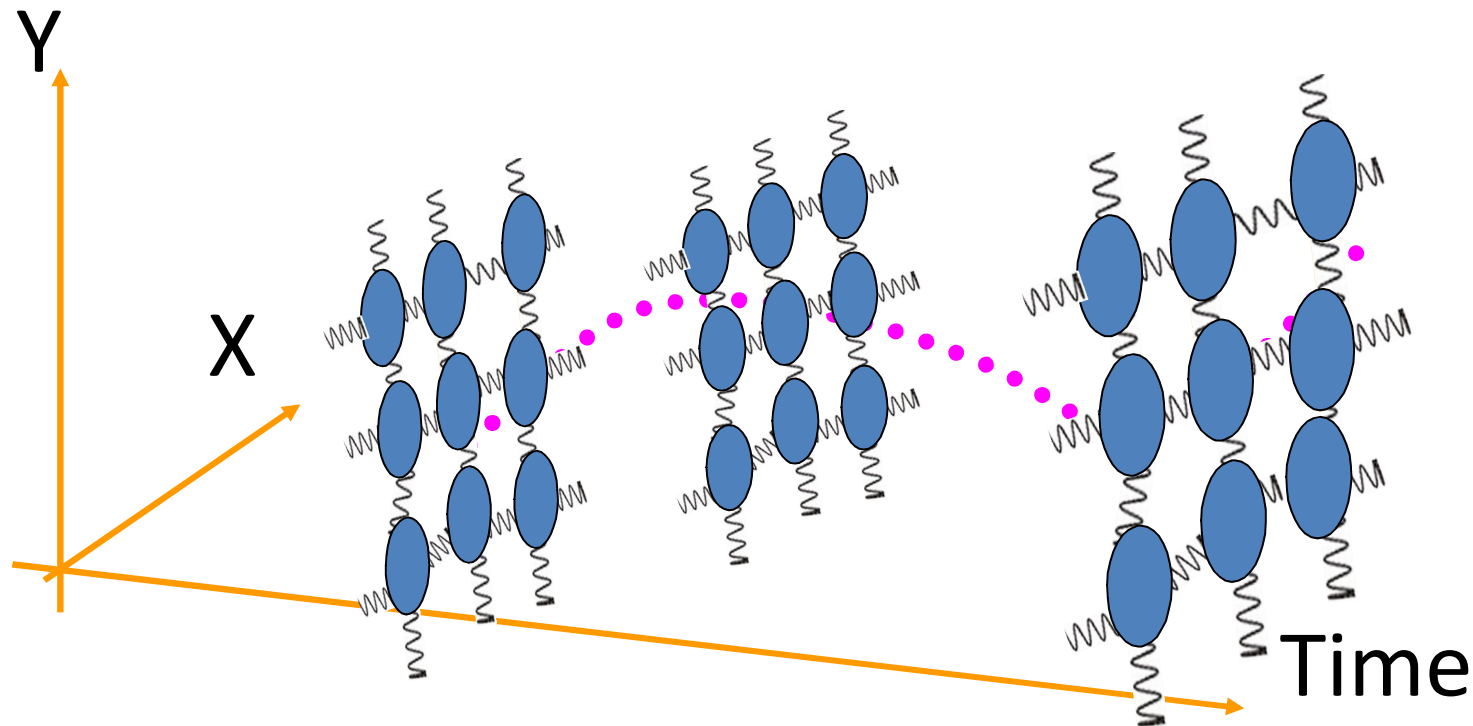
Using mean-shift in real-time to control a pan/tilt camera.



Collins, Amidi and Kanade, An Active Camera System for Acquiring Multi-View Video, ICIP 2002.

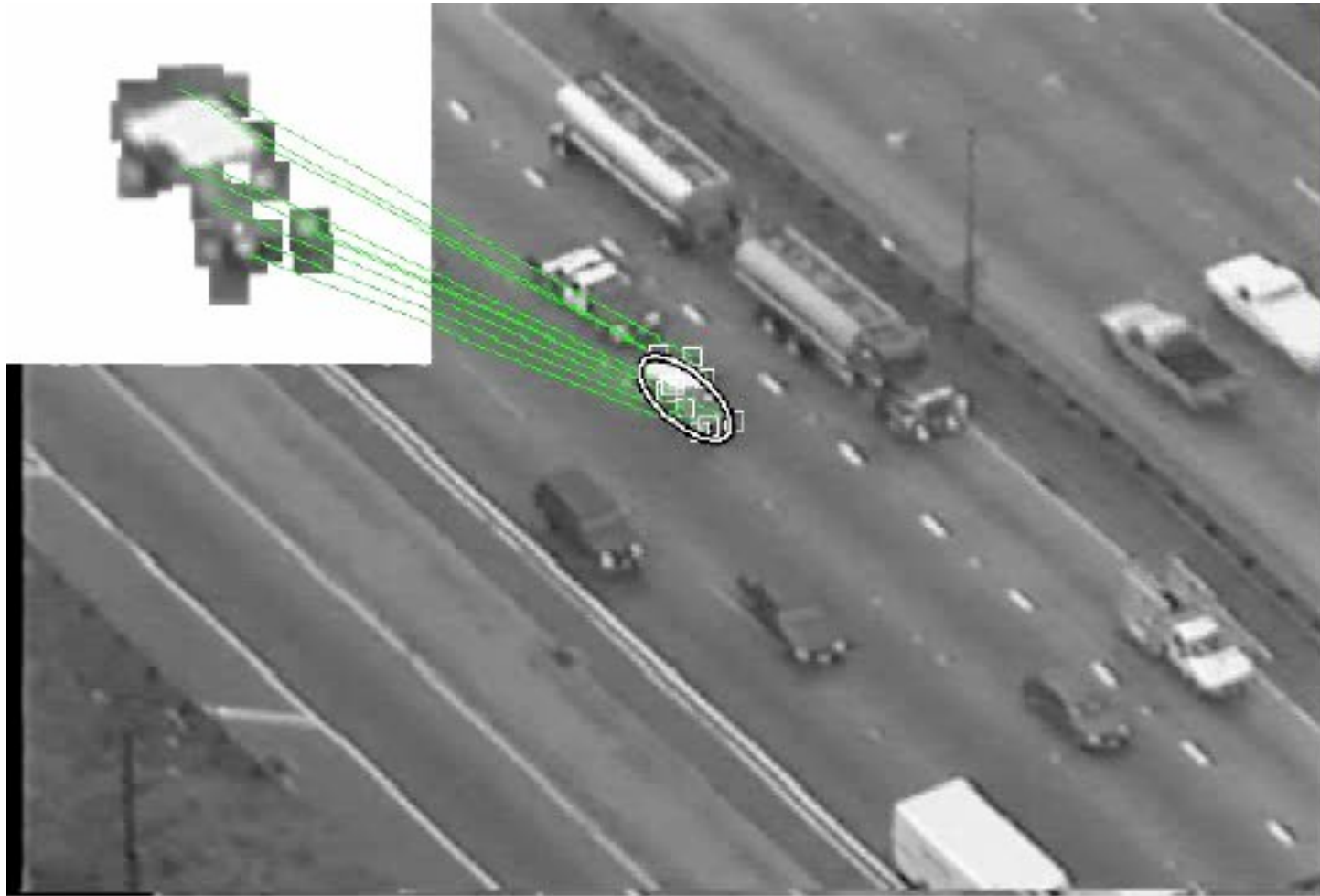
# Constellations of Patches

- Goal is to retain more spatial information than histograms, while remaining more flexible than single templates.



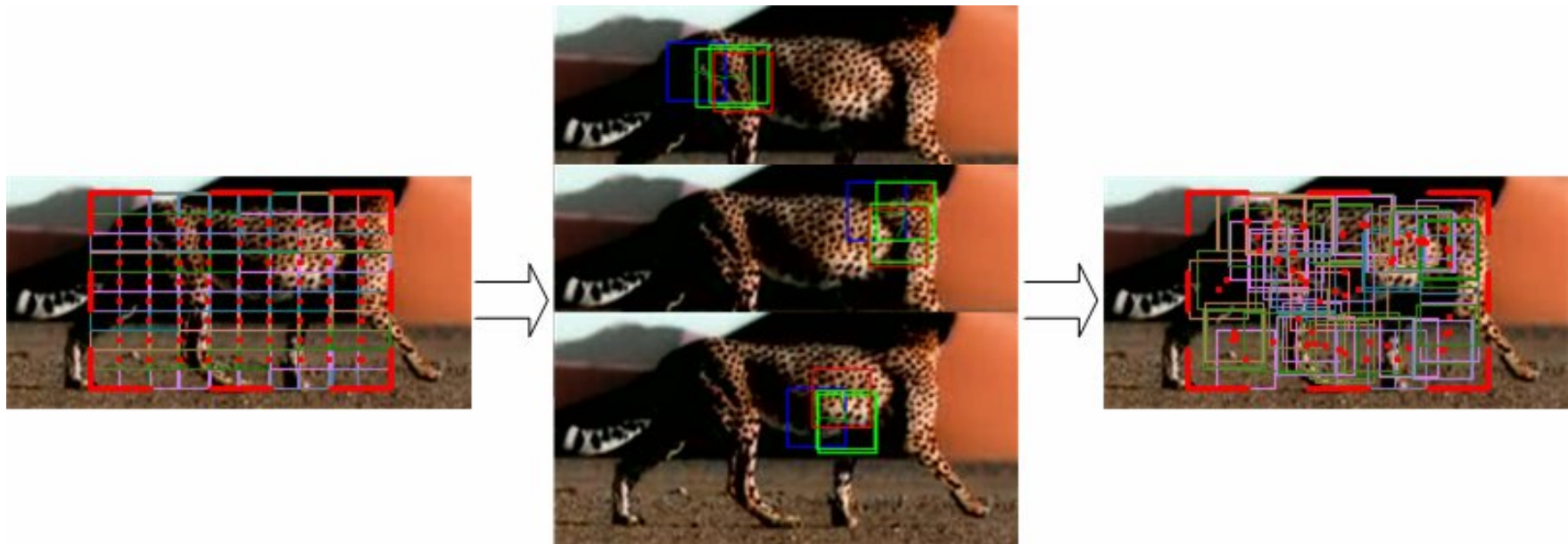
# Example: Corner Patch Model

Yin and Collins, "On-the-fly object modeling while tracking," CVPR 2007.



# Example: Attentional Regions

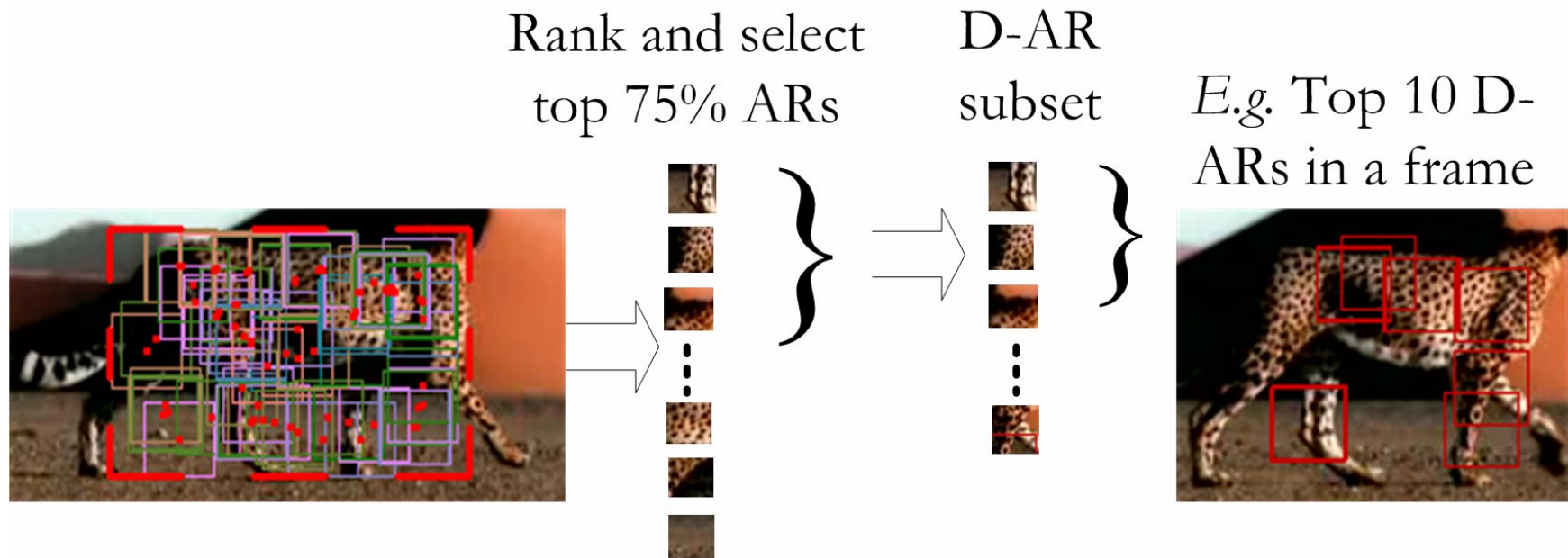
Yang, Yuan, and Wu, “Spatial Selection for Attentional Visual Tracking,” CVPR 2007.



ARs are patch features that are sensitive to motion (a generalization of corner features). AR matches in new frames collectively vote for object location.



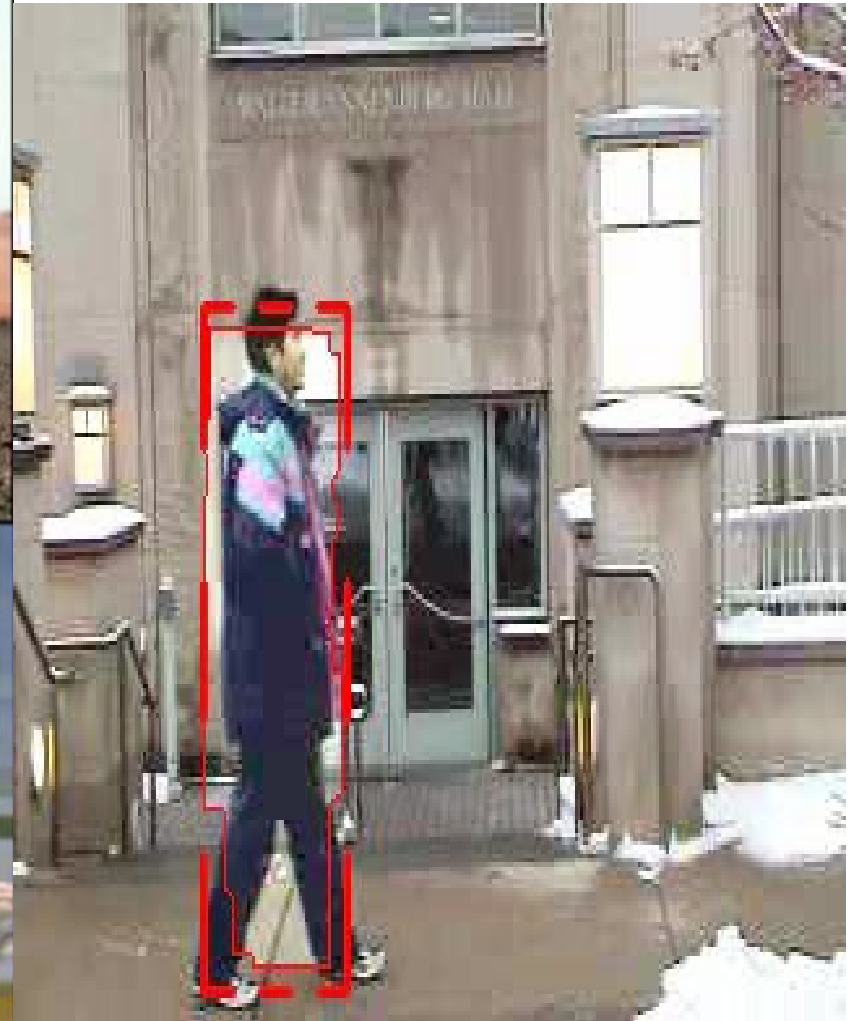
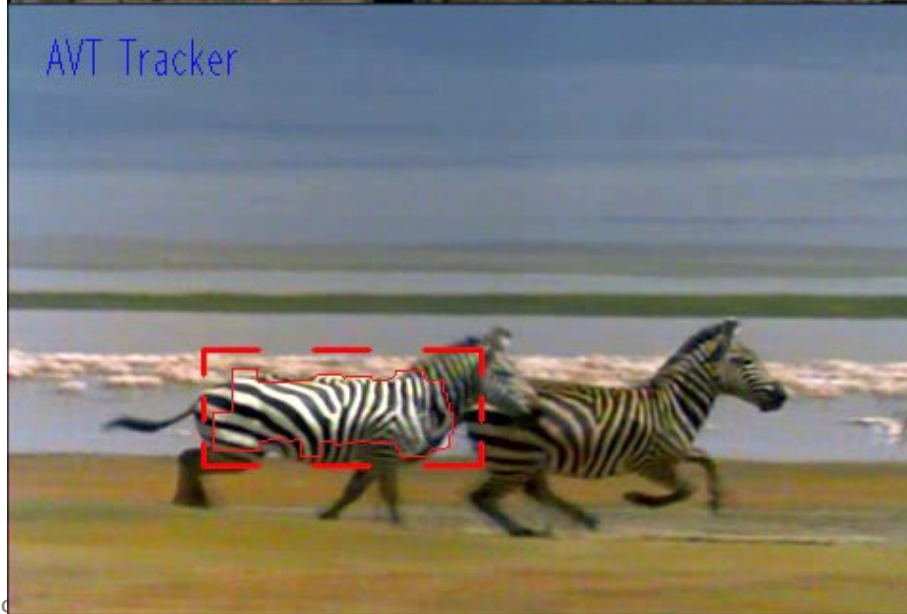
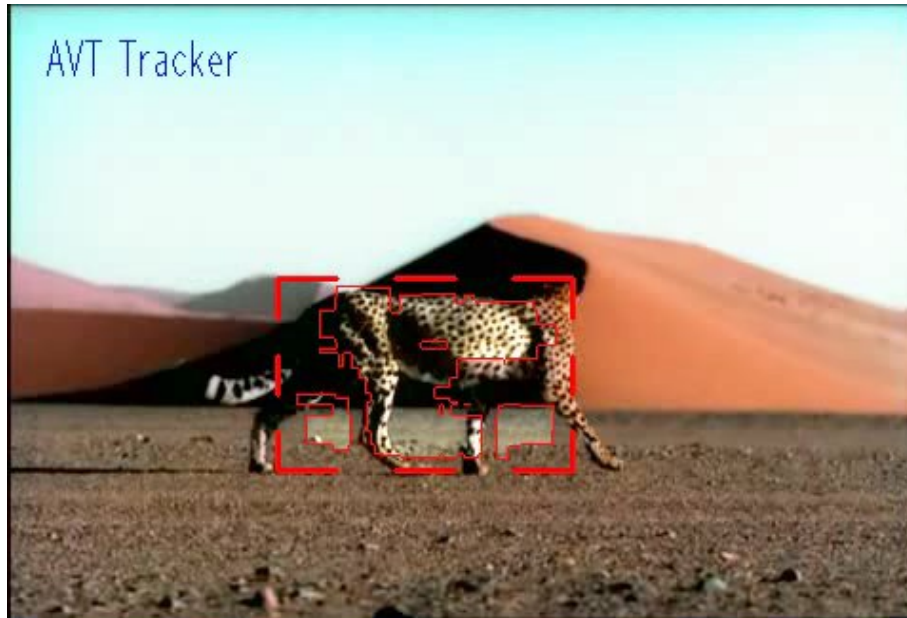
# Example: Attentional Regions



Discriminative ARs are chosen on-the-fly as those that best discriminate current object motion from background motion.

Drift is unlikely, since no on-line updates of ARs, and no new features are chosen after initialization in first frame. (but adaptation to extreme appearance change is this also limited)

# Example: Attentional Regions



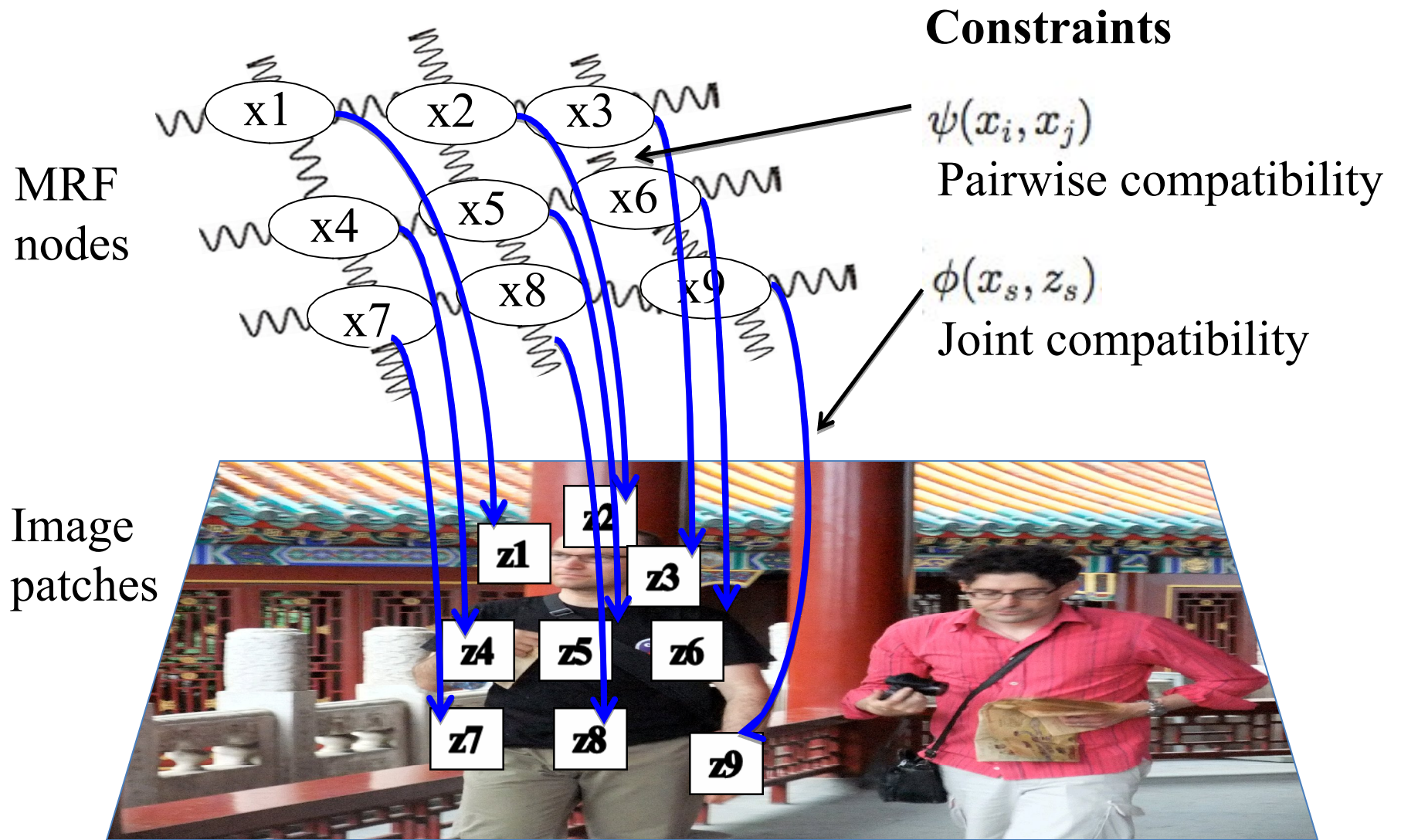
Movies courtesy of Ying Wu



# Tracking as MRF Inference

- Each patch becomes a node in a graphical model.
- Patches that influence each other (e.g. spatial neighbors) are connected by edges
- Infer hidden variables (e.g. location) of each node by Belief Propagation

# MRF Model Tracking



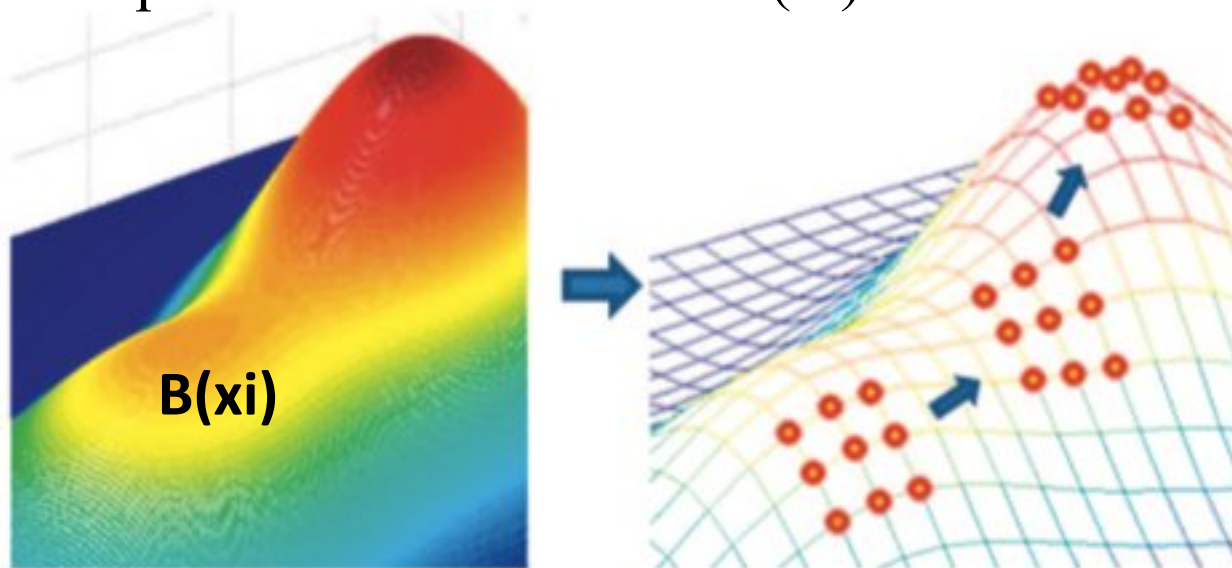
# Mean-Shift Belief Propagation

Park, Brocklehurst, Collins and Liu, “Deformed Lattice Detection in Real-World Images Using Mean-Shift Belief Propagation”, to appear, PAMI 2009.

Efficient inference in MRF models with particular applications to tracking.

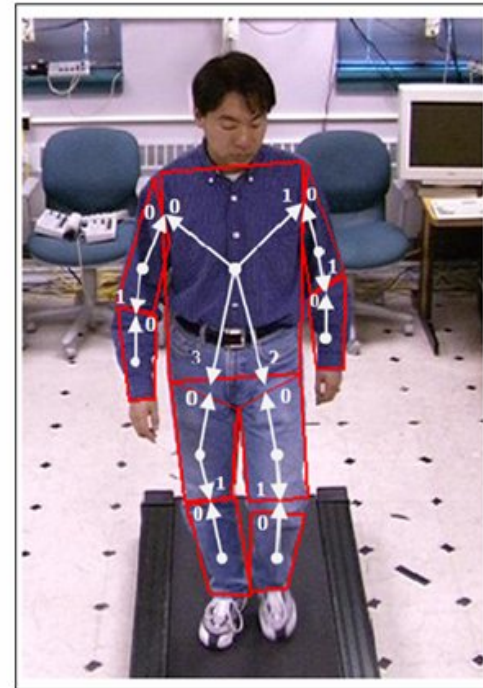
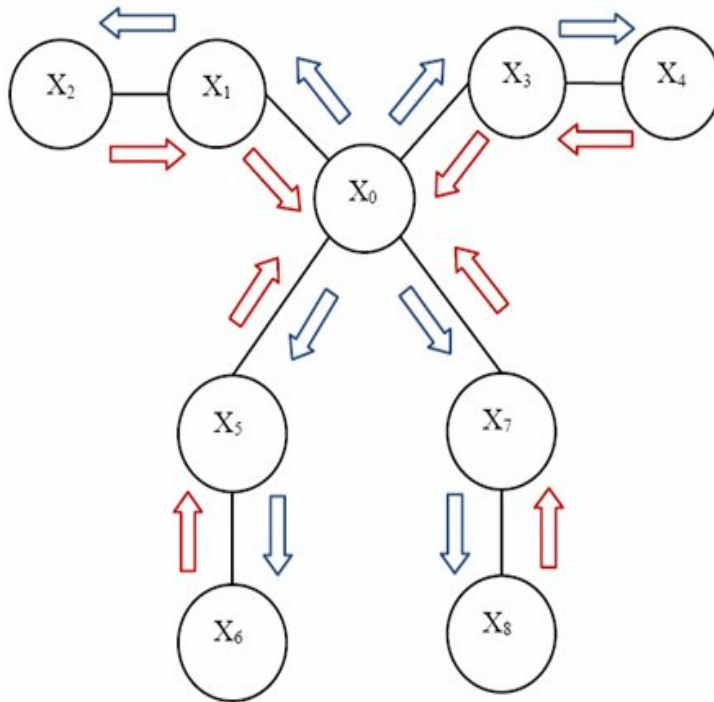
$$p(x_1, \dots, x_N, z_1, \dots, z_N) = k \prod_{(i,j)} \psi(x_i, x_j) \prod_s \phi(x_s, z_s)$$

General idea: Iteratively compute a belief surface  $B(x_i)$  for each node  $x_i$  and perform mean-shift on  $B(x_i)$ .



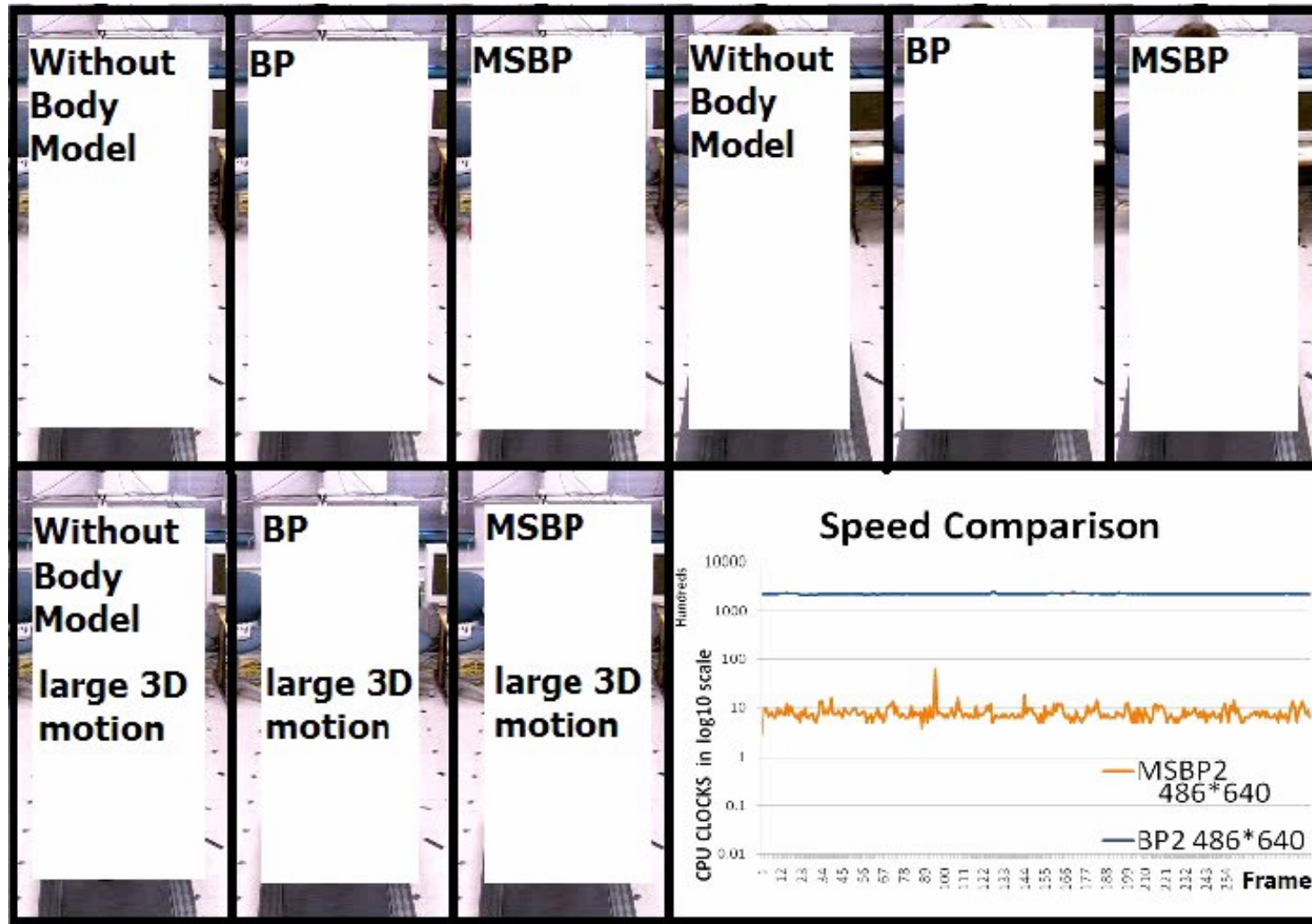
# Example: Articulated Body Tracking

- Loose-limbed body model. Each body part is represented by a node of an acyclic graph and the hidden variables we want to infer are 3 dimensional  $\mathbf{x}_i$  ( $x, y, \theta$ ), representing 2 dimensional translation ( $x, y$ ) and in-plane rotation  $\theta$





# Articulated Body Tracking



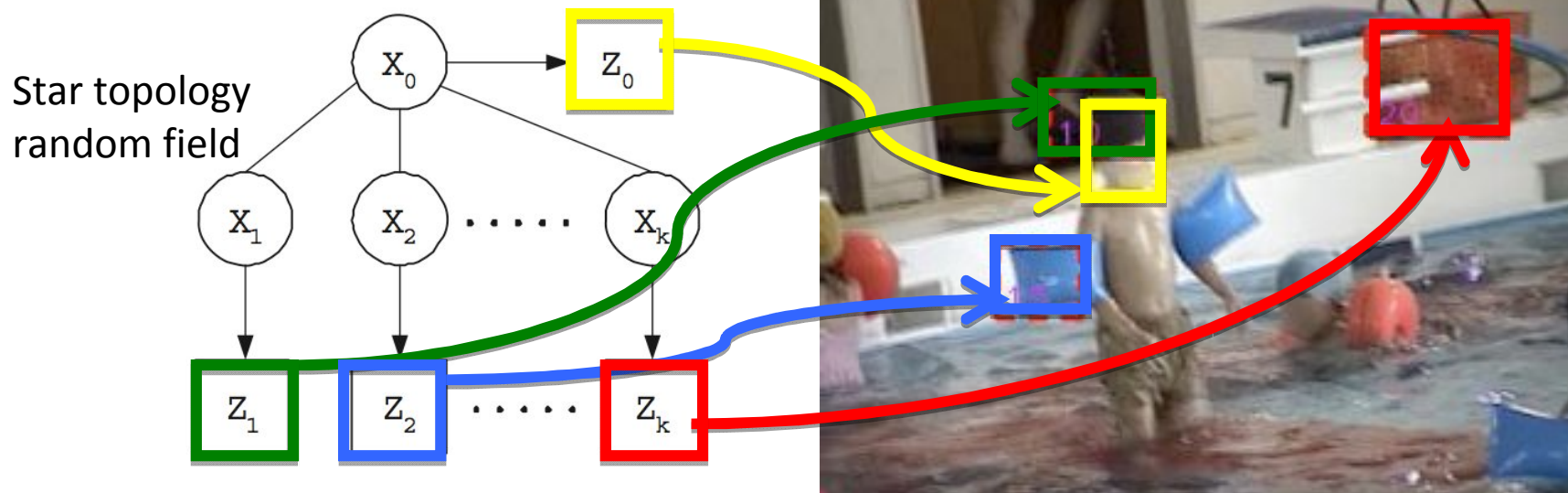
Limitations. If the viewpoint changes too much, this 2D graph tracker will fail. But the idea is that we also are running the body pose detector at the same time. The detector can this “guide” the tracker, and also reinitialize the tracker after failure.

# Example: Auxiliary Objects

Yang, Wu and Lao, “Intelligent Collaborative Tracking by Mining Auxiliary Objects,” CVPR 2006.

Look for auxiliary regions in the image that:

- frequently co-occur with the target
- have correlated motion with the target
- are easy to track

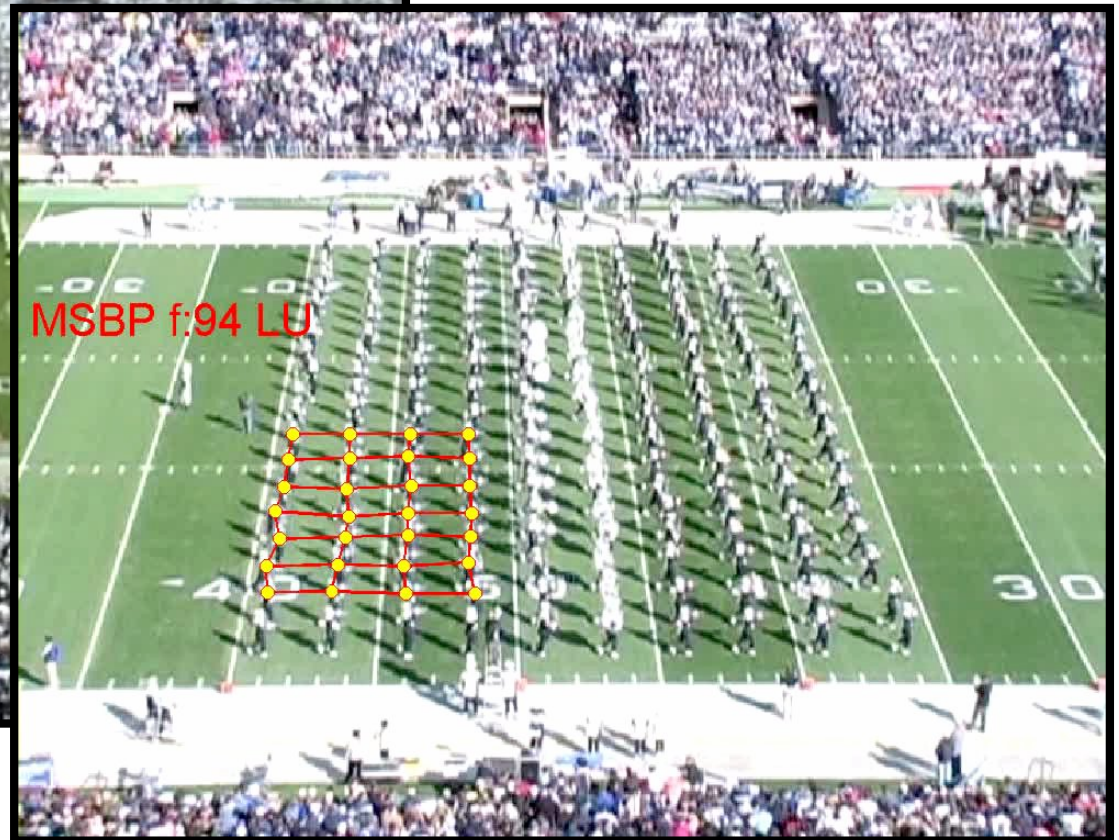


# Example: Formations of People

MSBP tracker can also track arbitrary graph-structured groups of people (including graphs that contain cycles).



examples of tracking the Penn State Blue Band





# Lecture Outline

- Brief Intro to Tracking
- Appearance-based Tracking
- Online Adaptation (learning)



# Motivation for Online Adaptation

First of all, we want succeed at persistent, long-term tracking!

The more invariant your appearance model is to variations in lighting and geometry, the less specific it is in representing a particular object. There is then a danger of getting confused with other objects or background clutter.

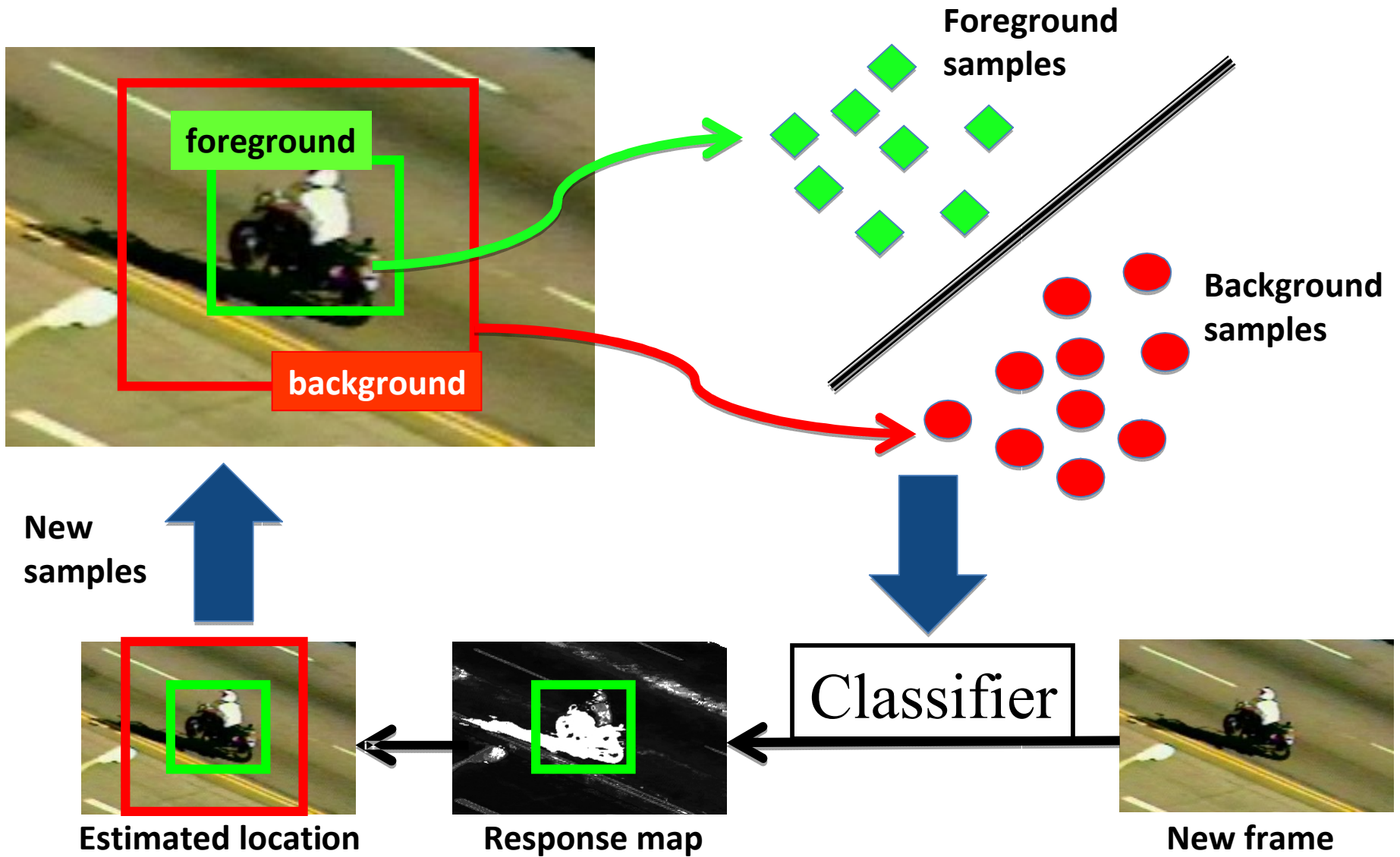
Online adaptation of the appearance model or the features used allows the representation to have retain good specificity at each time frame while evolving to have overall generality to large variations in object/background/lighting appearance.

# Tracking as Classification

Idea first introduced by Collins and Liu, “Online Selection of Discriminative Tracking Features”, ICCV 2003

- Target tracking can be treated as a binary classification problem that discriminates foreground object from scene background.
- This point of view opens up a wide range of classification and feature selection techniques that can be adapted for use in tracking.

# Overview:



# Observation

Tracking success/failure is highly correlated with our ability to distinguish object appearance from background.



Suggestion:

Explicitly seek features that best discriminate between object and background samples.

Continuously adapt feature used to deal with changing background, changes in object appearance, and changes in lighting conditions.

Collins and Liu, "Online Selection of Discriminative Tracking Features", ICCV 2003

# Feature Selection Prior Work

Feature Selection: choose  $M$  features from  $N$  candidates ( $M \ll N$ )

Traditional Feature Selection Strategies

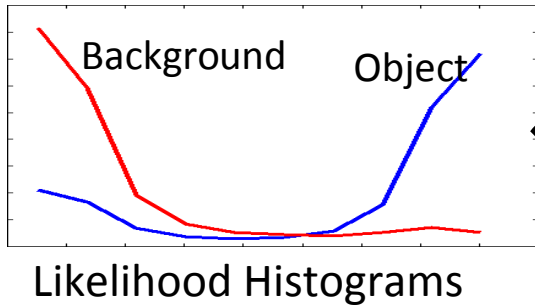
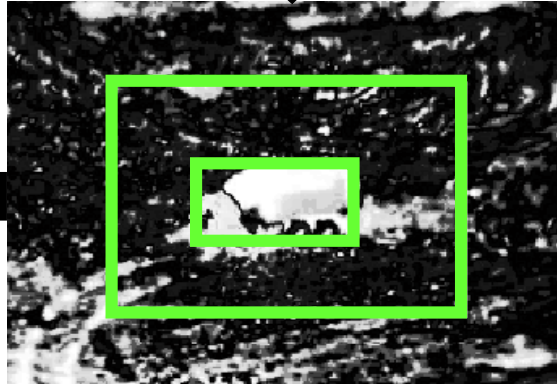
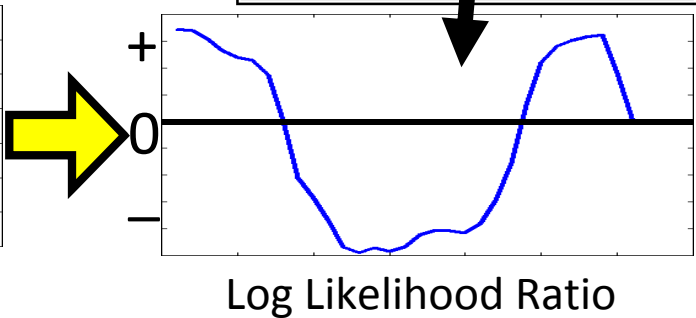
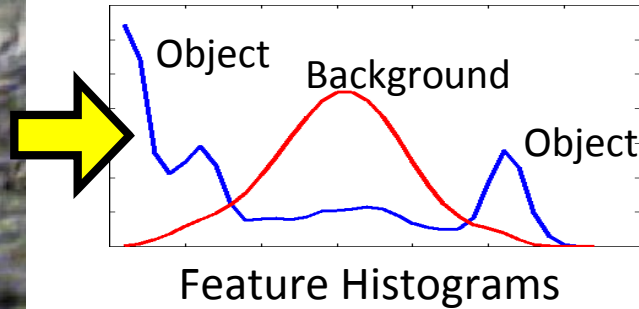
- Forward Selection
- Backward Selection
- Branch and Bound

Viola and Jones, Cascaded Feature Selection for Classification

Bottom Line: slow, off-line process

# Evaluation of Feature Discriminability

Can think of this as nonlinear, "tuned" feature, generated from a linear seed feature



Variance Ratio  
(feature score)

$$\frac{\text{Var between classes}}{\text{Var within classes}}$$

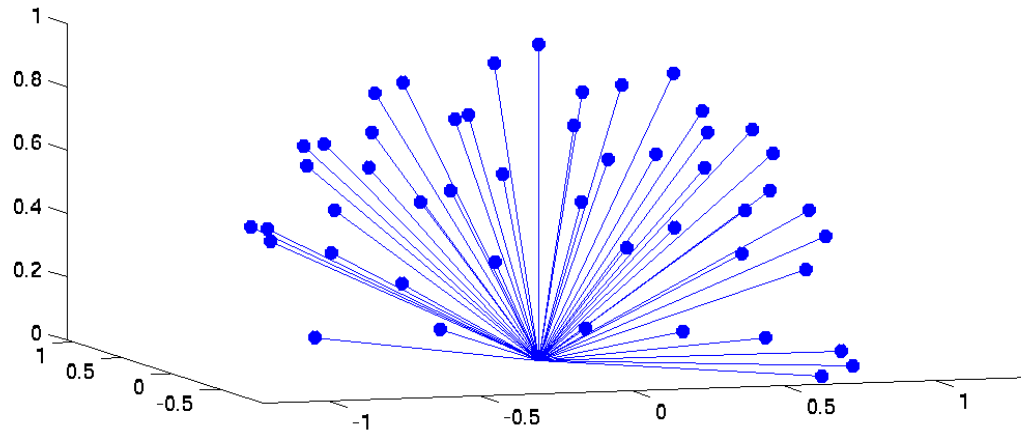
Note: this example also explains why we don't just use LDA

# Example: 1D Color Feature Spaces

Color features: integer linear combinations of R,G,B

$$\frac{(a R + b G + c B)}{(|a|+|b|+|c|)} + \text{offset}$$

where  $a,b,c$  are  $\{-2,-1,0,1,2\}$  and offset is chosen to bring result back to  $0,\dots,255$ .

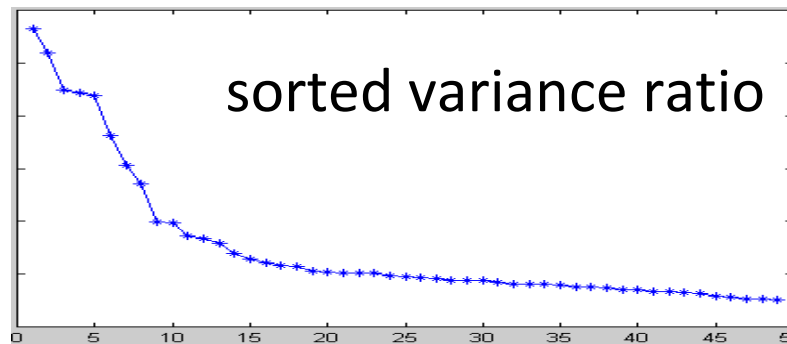
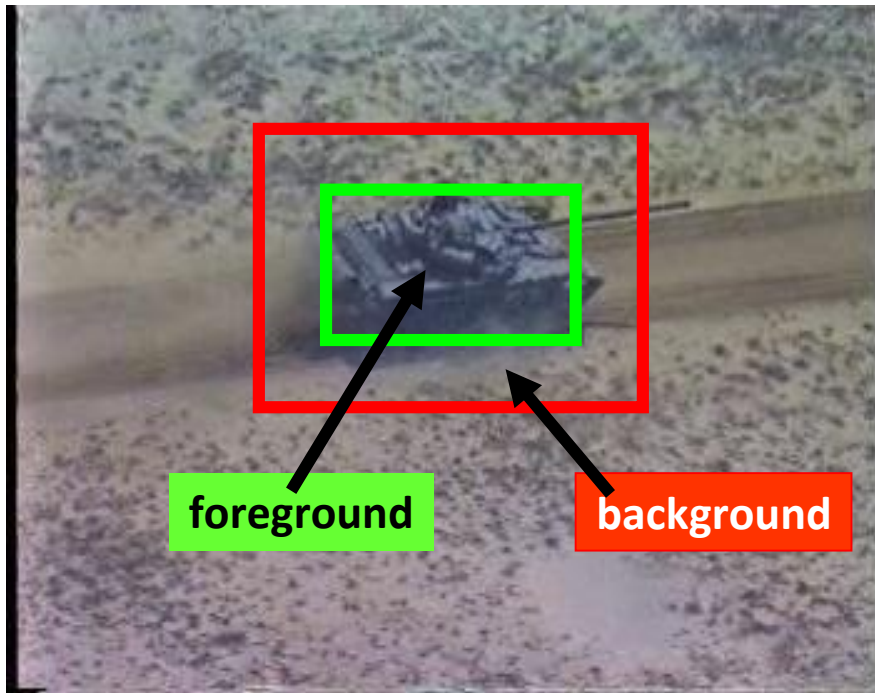


The 49 color feature candidates roughly uniformly sample the space of 1D marginal distributions of RGB.

# Example

training frame

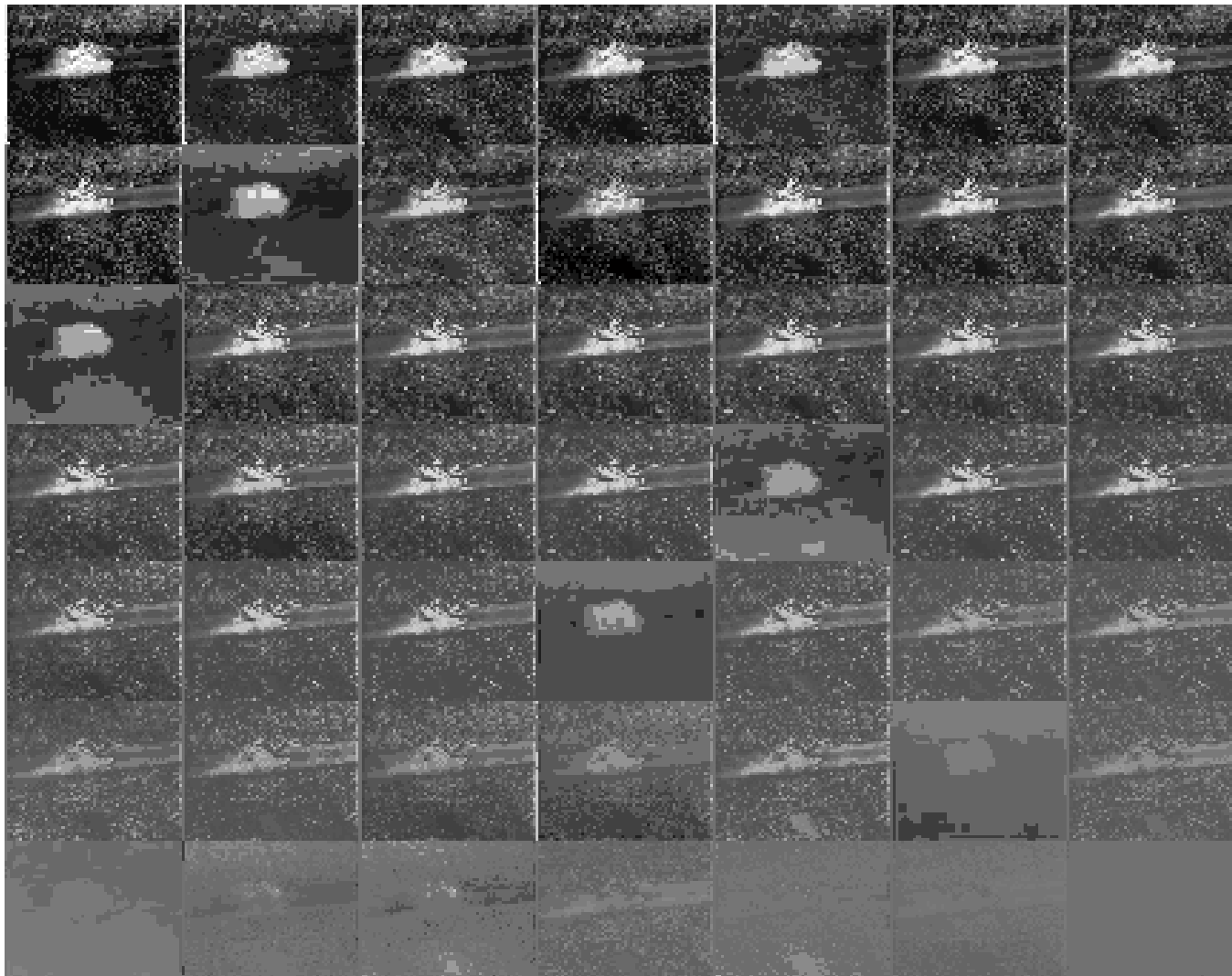
test frame





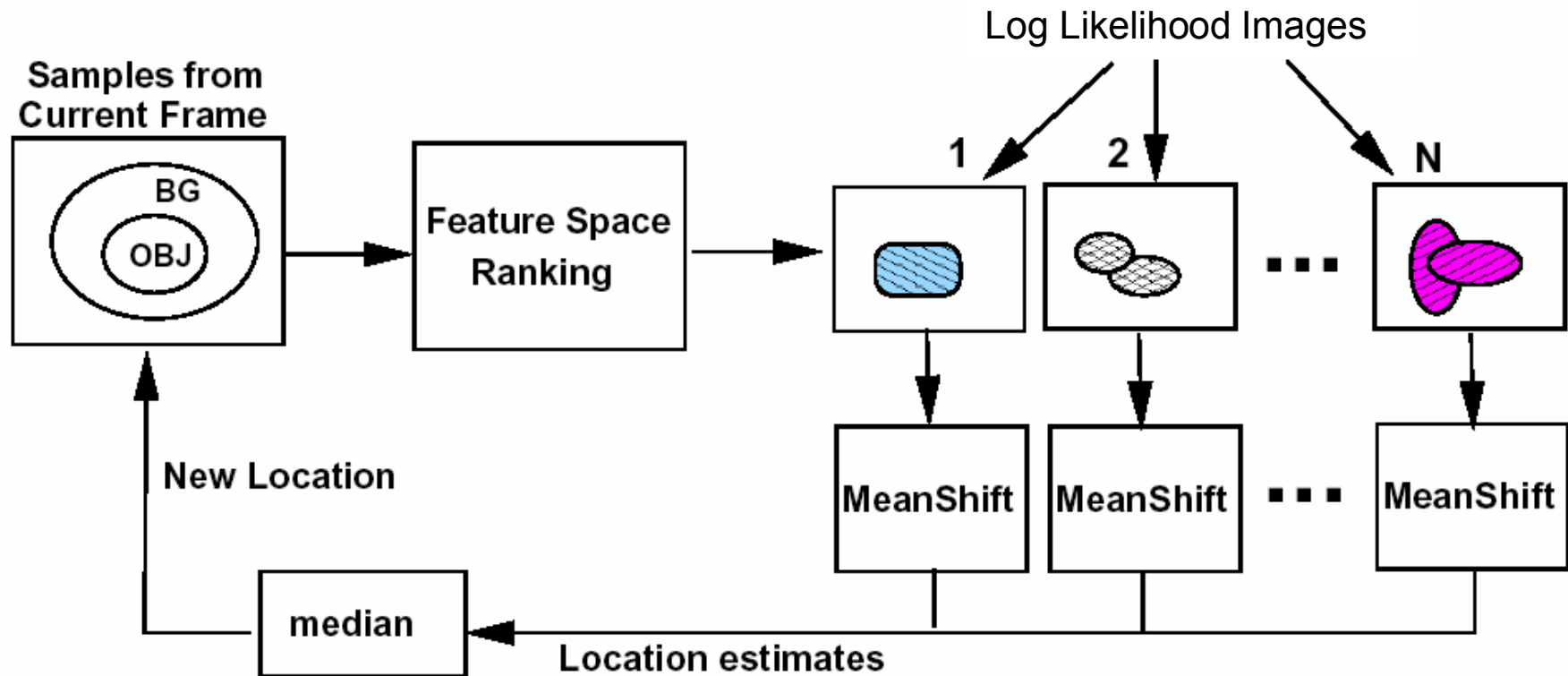
# Example: Feature Ranking

Best



Worst

# Overview of Tracking Algorithm



Note: since log likelihood images contain negative values, must use modified mean-shift algorithm as described in Collins, CVPR'03

# Avoiding Model Drift

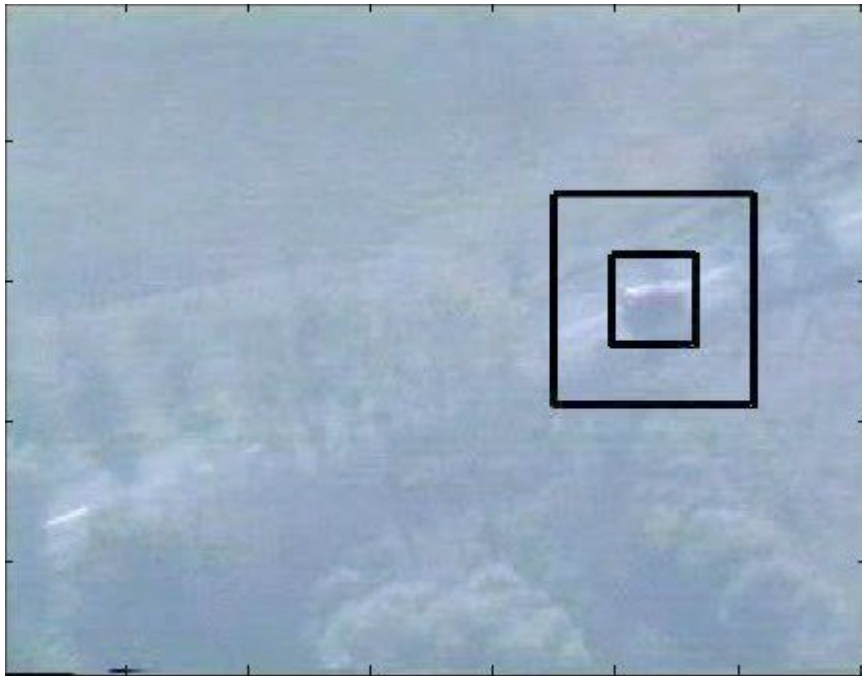
Drift: background pixels mistakenly incorporated into the object model pull the model off the correct location, leading to more misclassified background pixels, and so on.

Our solution: force foreground object distribution to be a combination of current appearance and original appearance (anchor distribution)

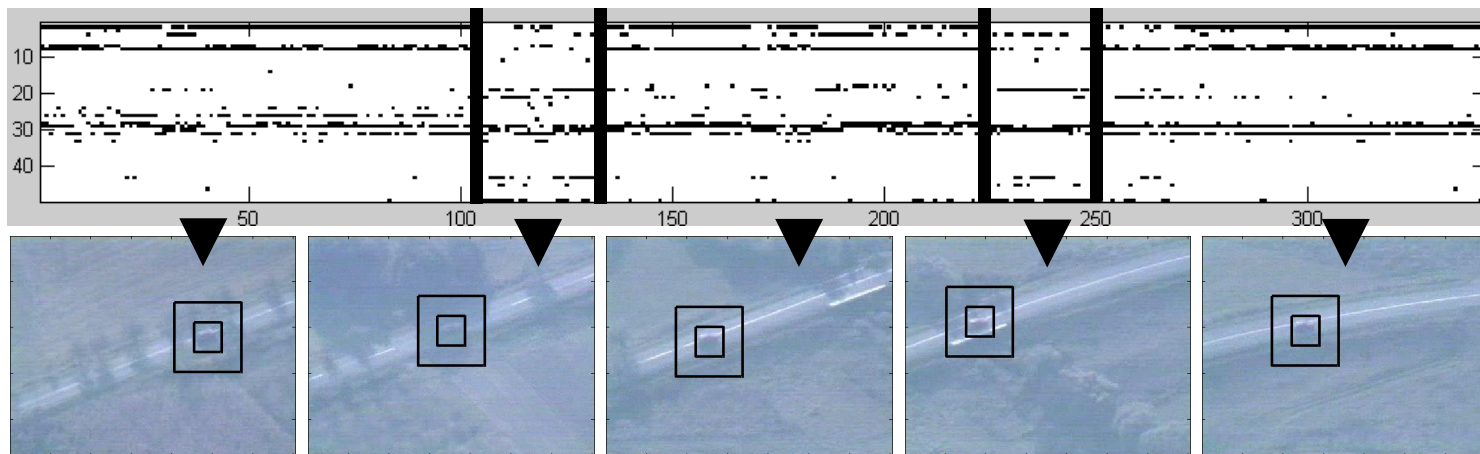
anchor distribution = object appearance histogram from first frame  
model distribution = (current distribution + anchor distribution) / 2

Note: this solves the drift problem, but limits the ability of the appearance model to adapt to large color changes

# Examples: Tracking Hard-to-See Objects



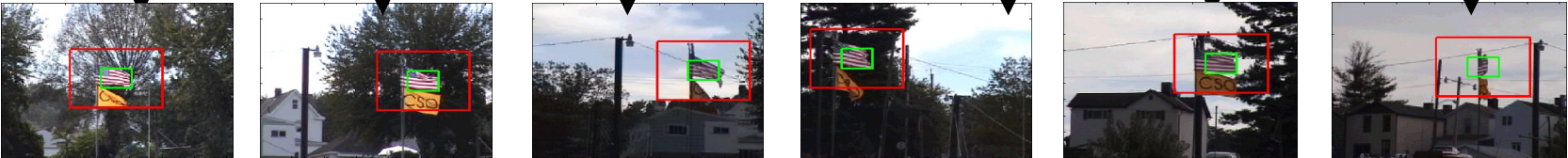
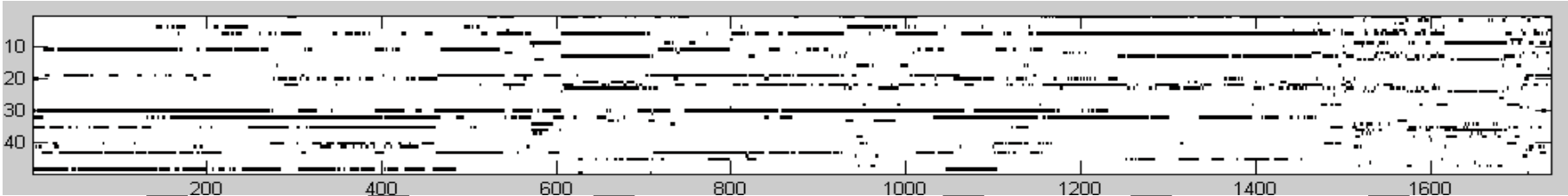
Trace of selected features



# Examples: Changing Illumination / Background



Trace of selected features



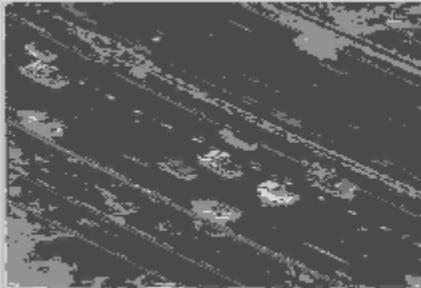
# Examples: Minimizing Distractions

Current location

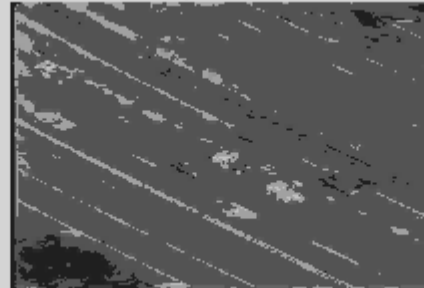
Feature scores



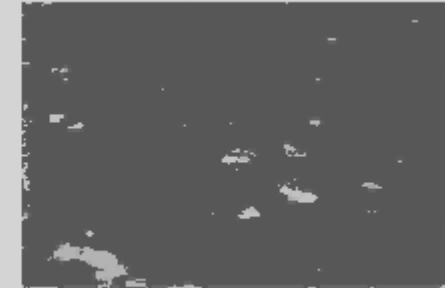
-R+2G



2R-2G+B



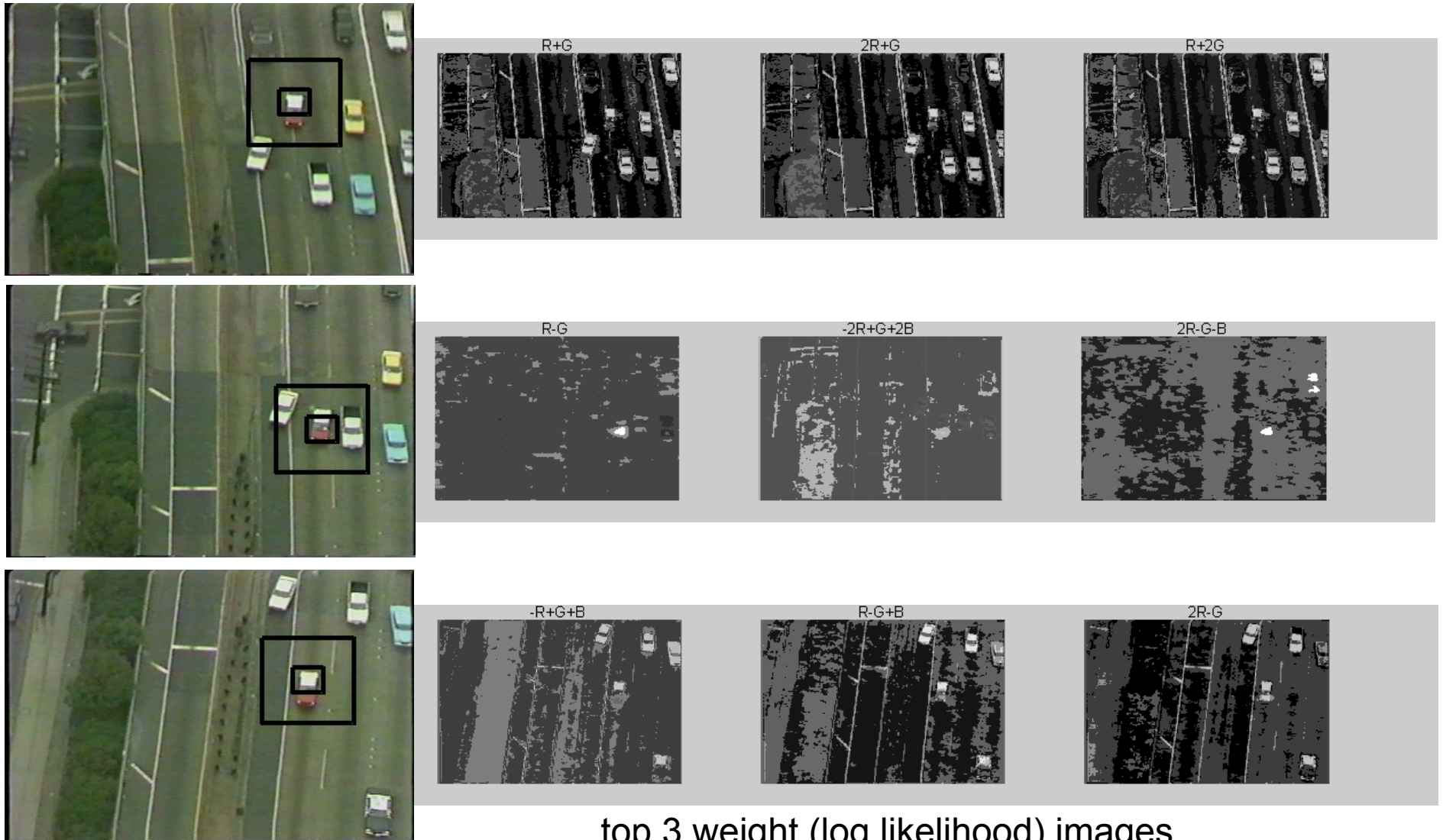
-R+2G-B



Top 3 weight (log likelihood) images



# More Detail



top 3 weight (log likelihood) images

# On-line Boosting for Feat Select

Grabner, Grabner, and Bischof, “Real-time tracking via on-line boosting.” BMVC 2006.

Use boosting to select and maintain the best discriminative features from a pool of feature candidates.

- Haar Wavelets
- Integral Orientation Histograms
- Simplified Version of Local Binary Patterns



# Boosting

- general method for improving the accuracy of any learning algorithm
- combine (weak) classifier (weighted vote of weak classifiers)

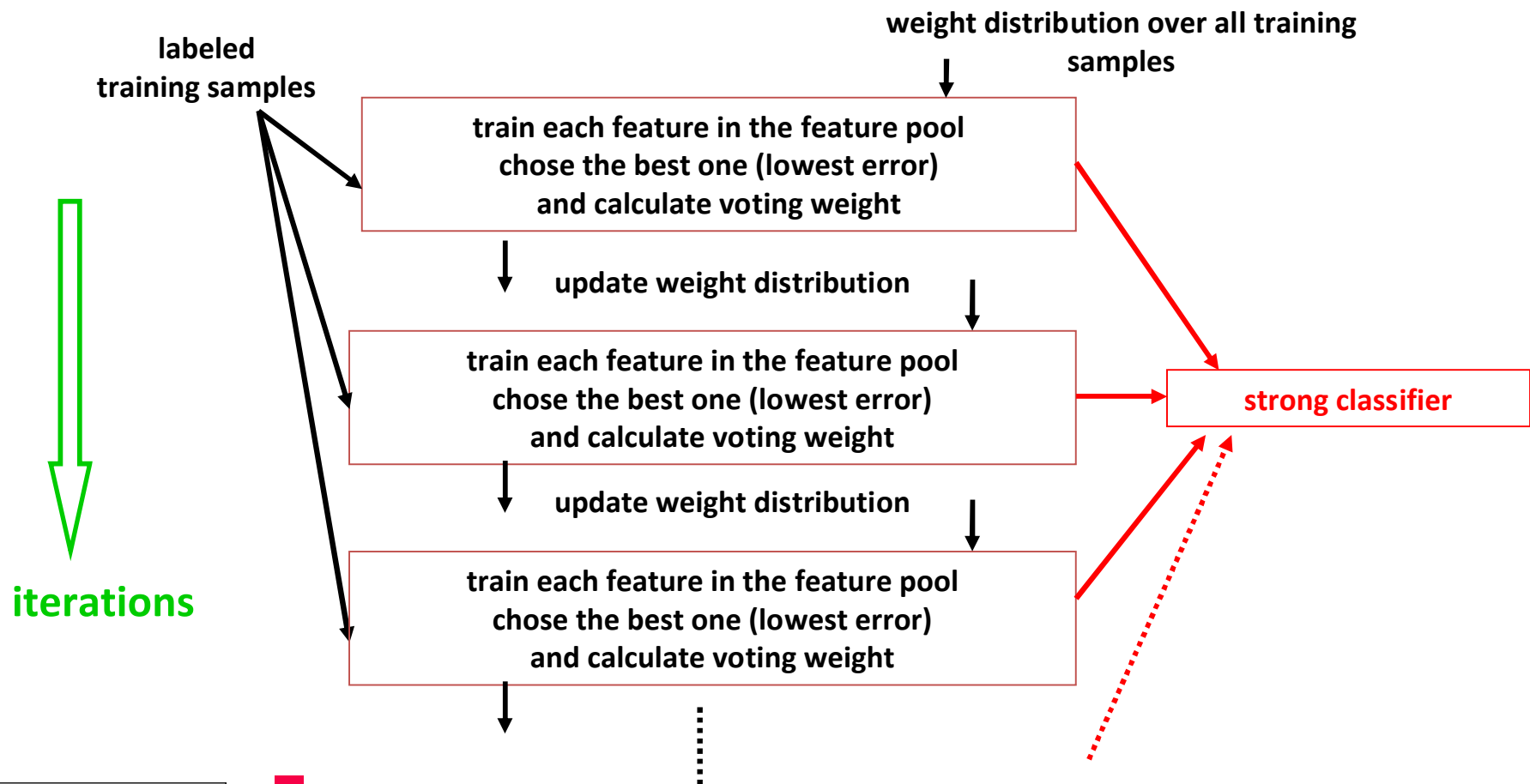
$$hStrong(x) = \text{sign}\left(\sum_{t=1}^N \alpha_n \cdot hWeak_n(x)\right)$$

## AdaBoost (adaptive boosting)

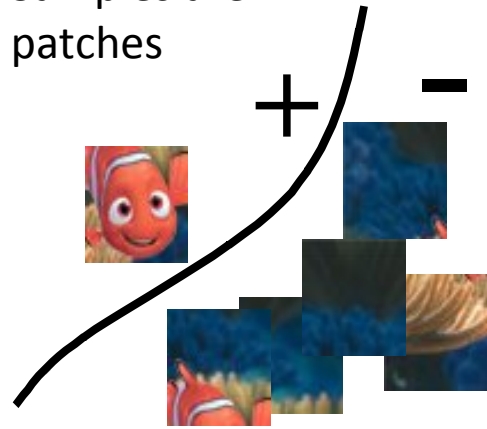
- instead of sampling, re-weight (**Y. Freund and R. Schapire**)
  - training error: decreases exponentially
  - generalization error: SVM – maximizes the margin
- widely used
  - text recognition, routing, learning problems in natural language processing,...
  - image retrieval, generic object detection and recognition, active shape model,...

# OFF-line Boosting for Feature Selection

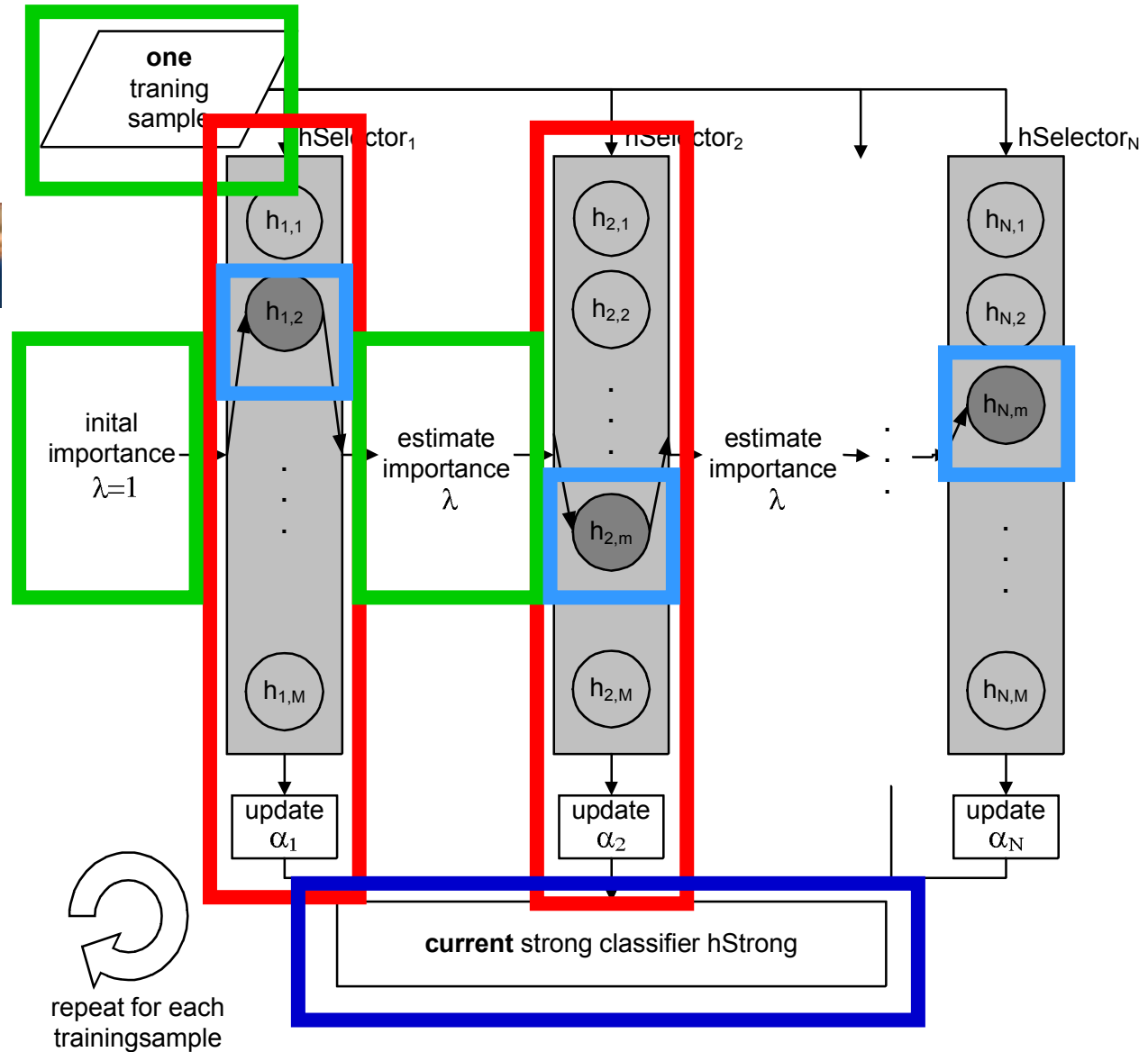
- Each weak classifier corresponds to a feature
- train all weak classifiers - choose best at each boosting iteration
- **add one** feature in each iteration



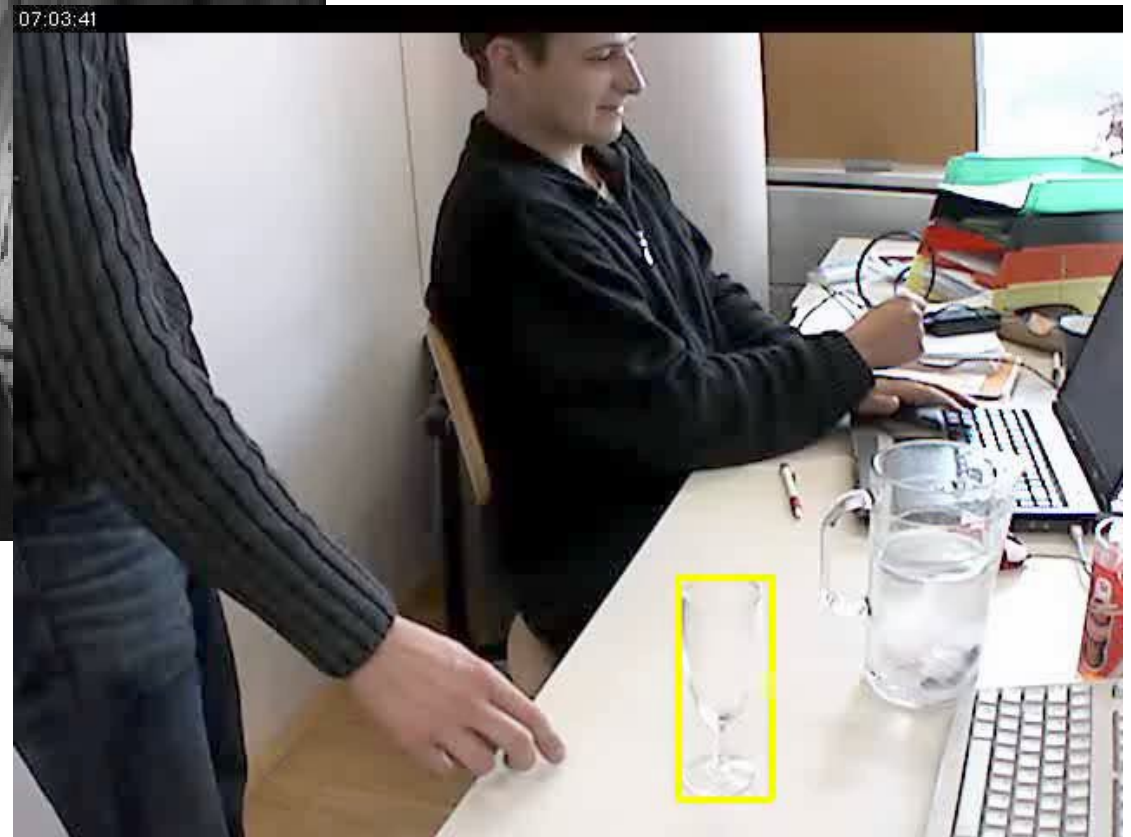
Samples are patches



# On-line Version...



# Tracking Examples



Horst Bischof

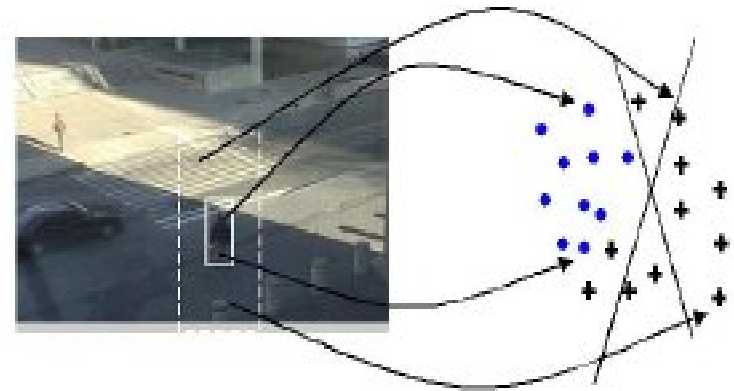


# Ensemble Tracking

Avidan, “Ensemble Tracking,” PAMI 2007

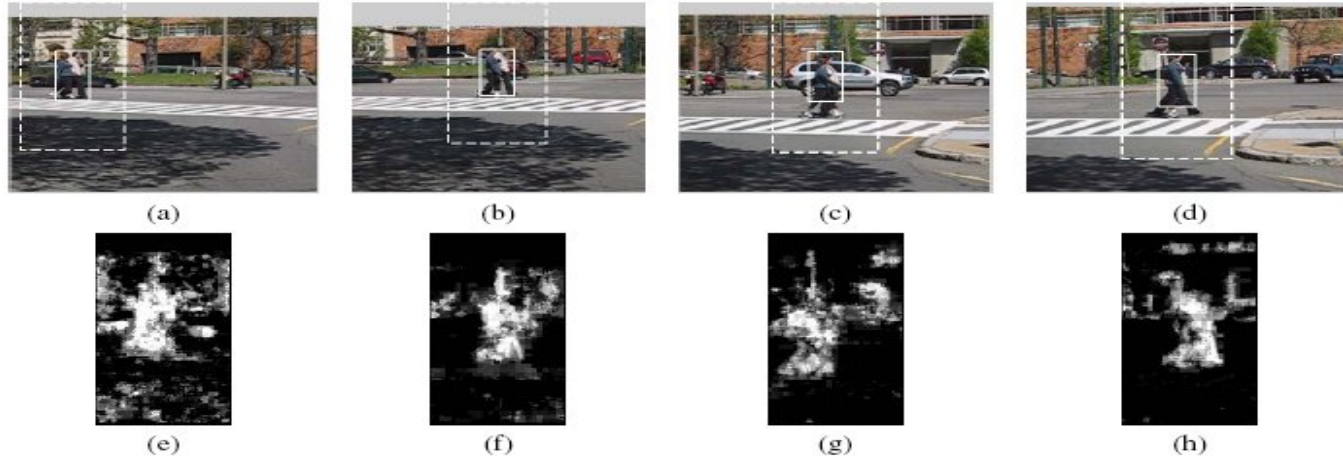
Use online boosting to select and maintain a set of weak classifiers (rather than single features), weighted to form a strong classifier. Samples are pixels.

Each weak classifier is a linear hyperplane in an 11D feature space composed of R,G,B color and a histogram of gradient orientations.



Classification is performed at each pixel, resulting in a dense confidence map for mean-shift tracking.

# Ensemble Tracking



During online updating:

- Perform mean-shift, and extract new pos/neg samples
- Remove worst performing classifier (highest error rate)
- Re-weight remaining classifiers and samples using AdaBoost
- Train a new classifier via AdaBoost and add it to the ensemble

Drift avoidance: paper suggests keeping some “prior” classifiers that can never be removed. (Anchor strategy).

# Semi-supervised Boosting

Grabner, Leistner and Bischof, "Semi-Supervised On-line Boosting for Robust Tracking," ECCV 2008.

Designed specifically to address the drift problem. It is another example of the Anchor Strategy.

Basic ideas:

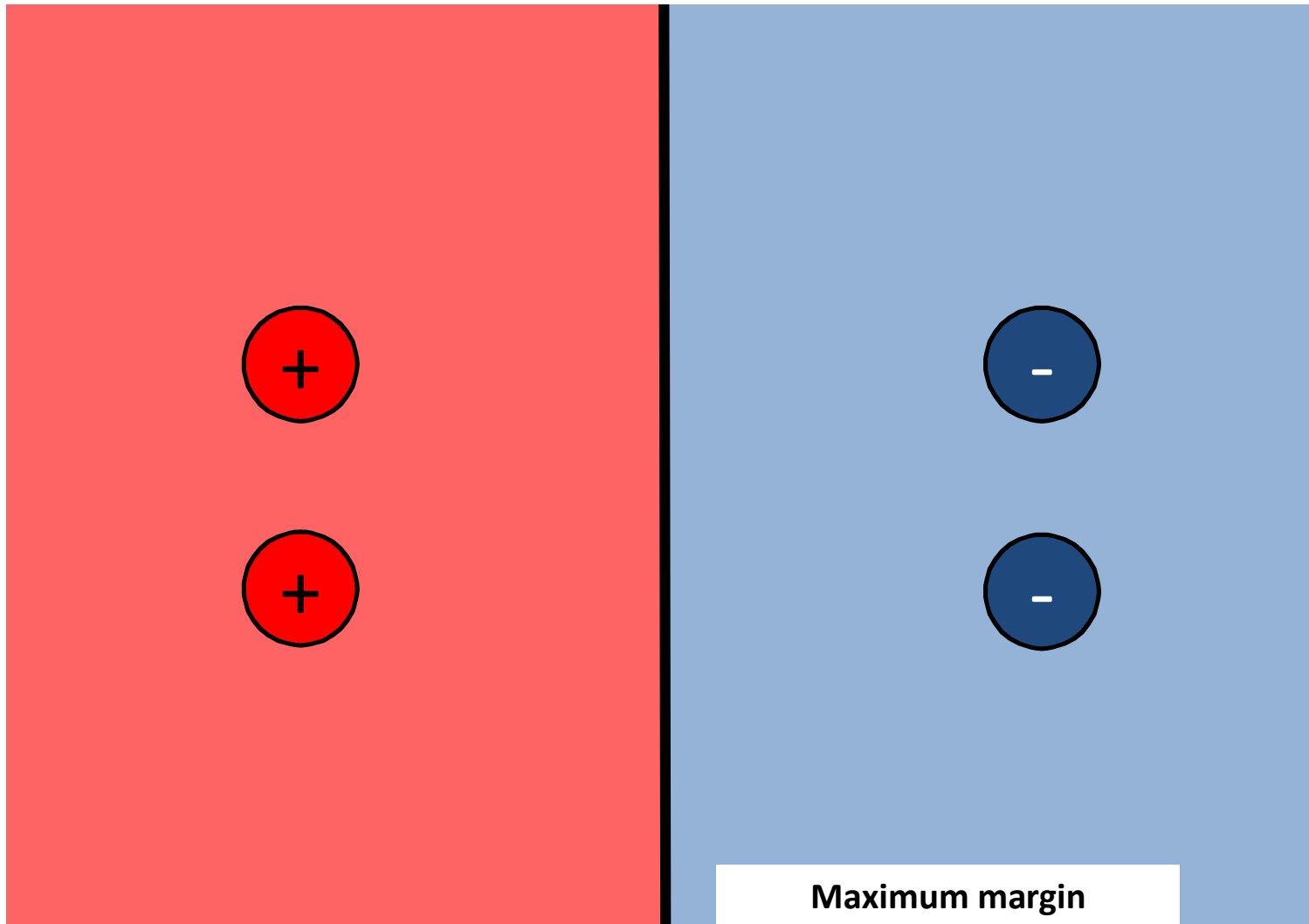
- Combine 2 classifiers

Prior (offline trained)  $H^{\text{off}}$  and online trained  $H^{\text{on}}$

Classifier  $H^{\text{off}} + H^{\text{on}}$  cannot deviate too much from  $H^{\text{off}}$

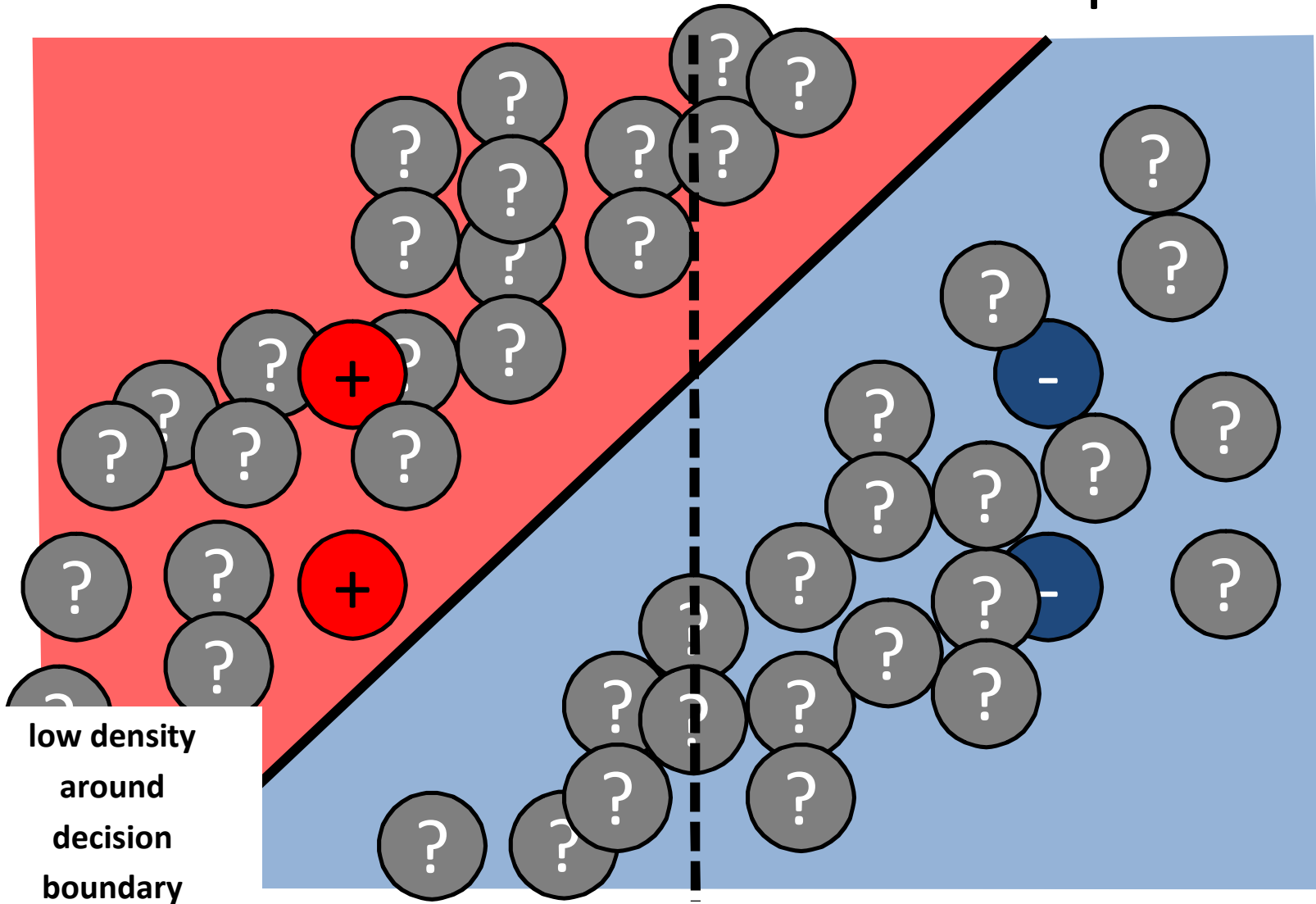
- Semi-supervised learning framework

# Supervised learning



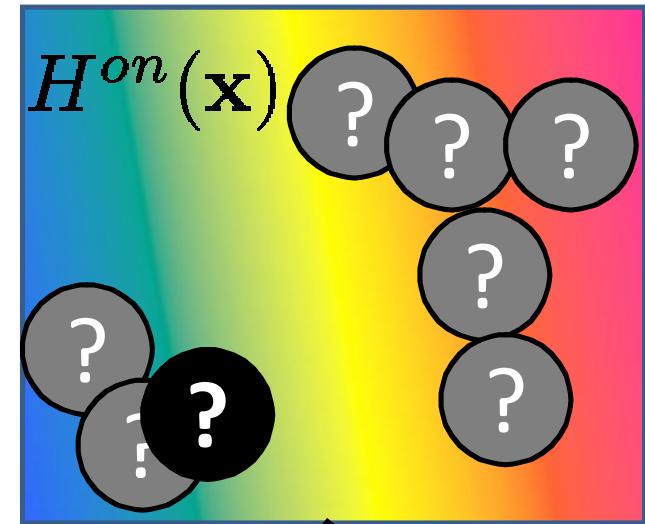
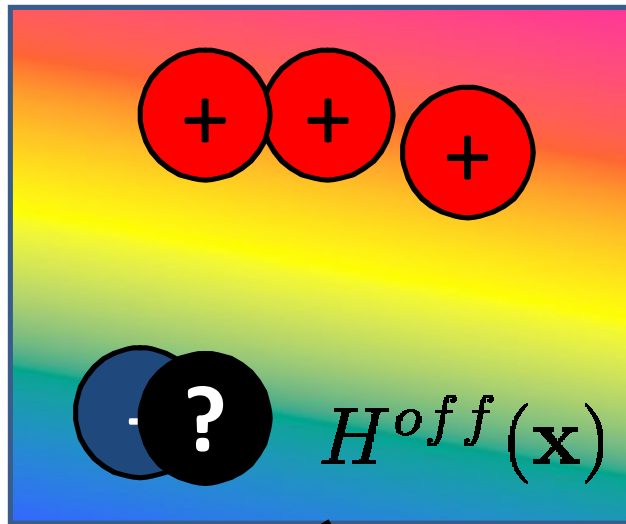


# Can Unlabeled Data Help?

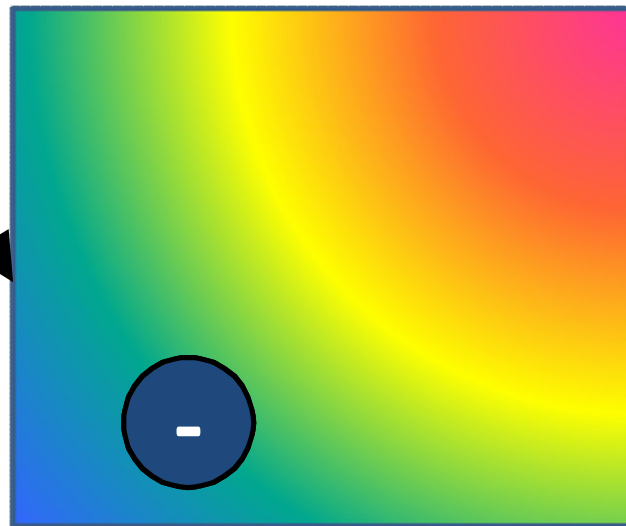


# Drift Avoidance

Key idea: samples from new frame are only used as unlabeled data!!!



Labeled data comes from first frame

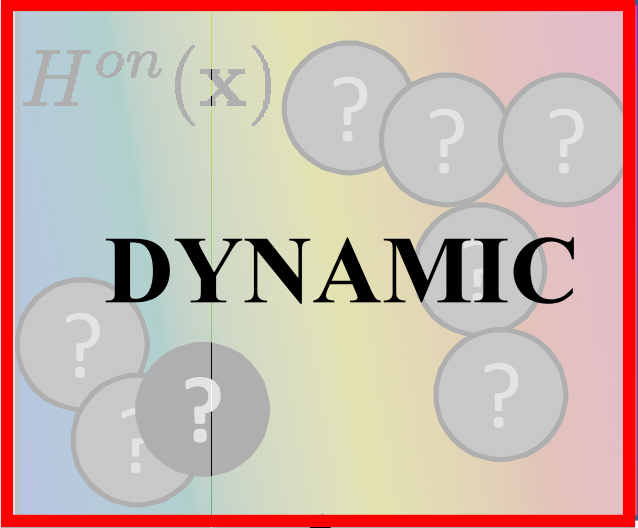
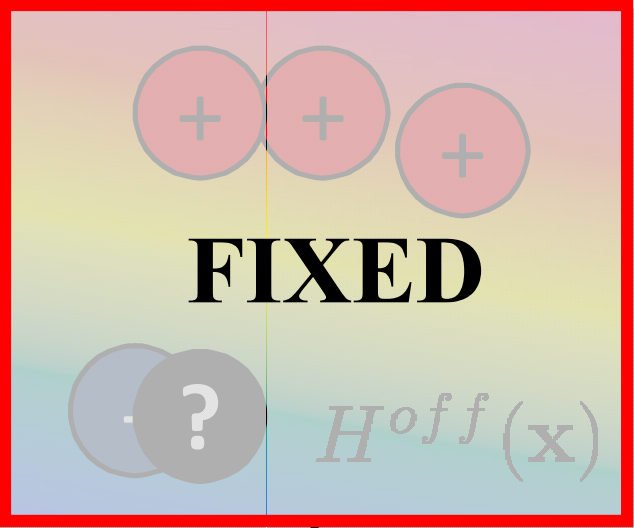


Combined classifier

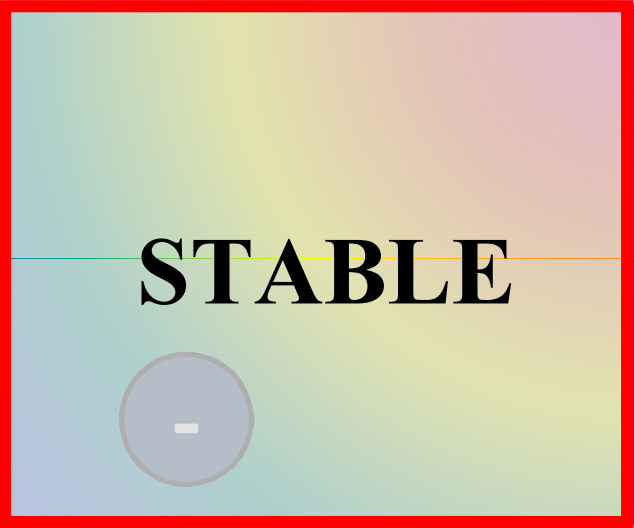
$$\text{sign} (H^{off}(\mathbf{x}) + H^{on}(\mathbf{x}))$$

# Drift Avoidance

Key idea: samples from new frame are only used as unlabeled data!!!



Labeled data comes from first frame



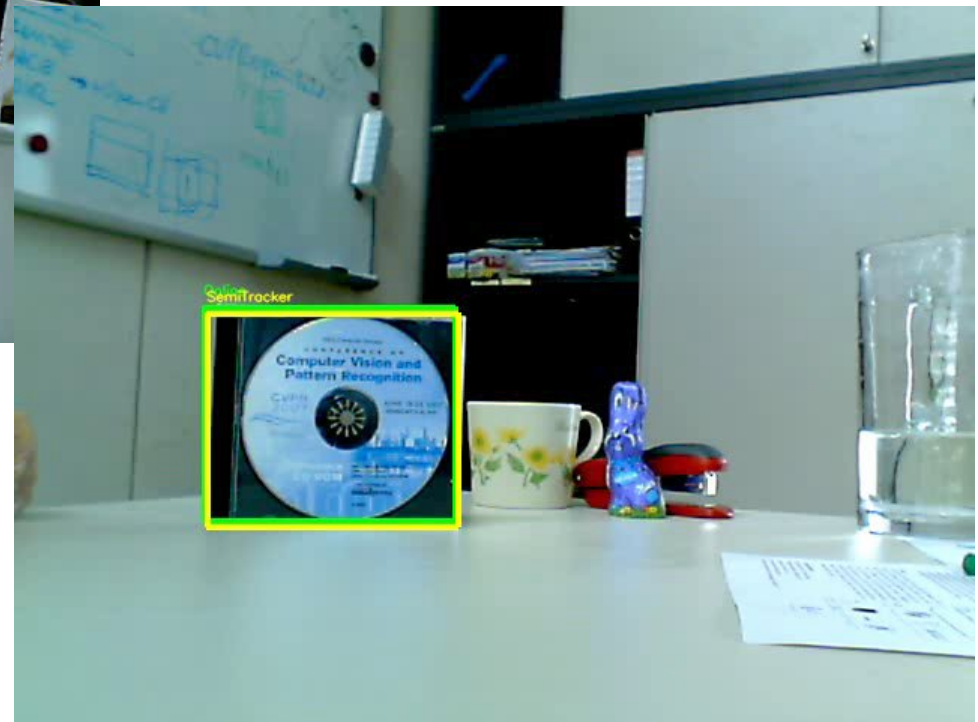
Combined classifier

$$\text{sign} (H^{\text{off}}(\mathbf{x}) + H^{\text{on}}(\mathbf{x}))$$

# Examples



Green: online boosting  
Yellow: semi-supervised

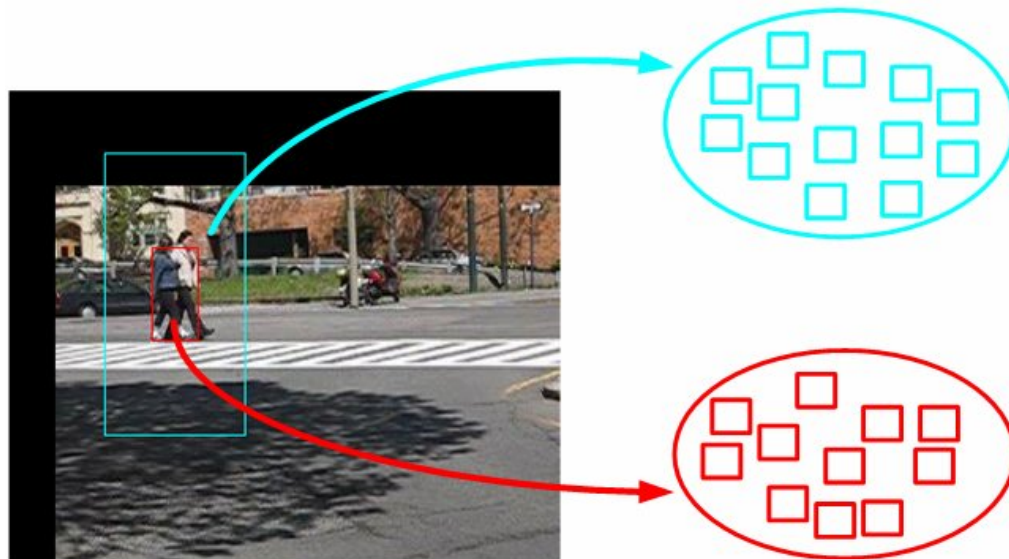


Horst Bischof

# Bag of Patches Model

Lu and Hager, “A Nonparametric Treatment for Location Segmentation based Visual Tracking,” CVPR 2007.

Key Idea: rather than try to maintain a set of features or set of classifiers, appearance of foreground and background is modeled directly by maintaining a set of sample patches.



KNN then determines the classification of new patches.

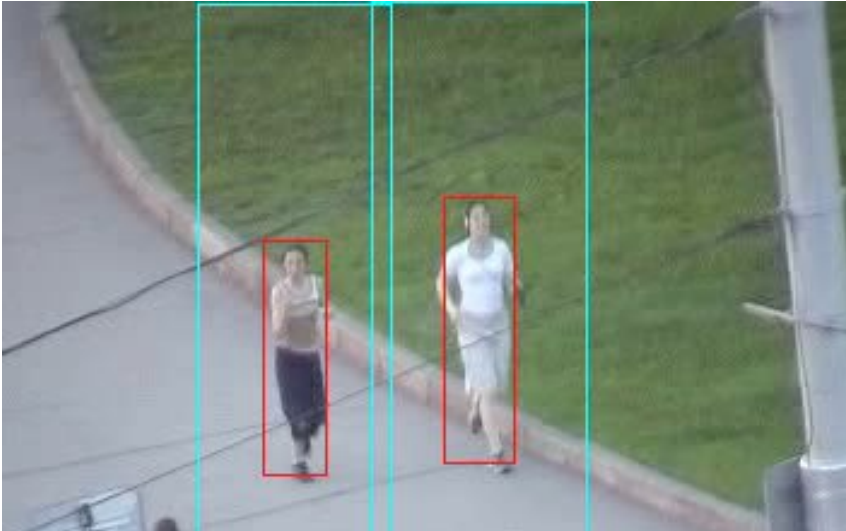
# Drift Avoidance (keep patch model clean)

Given new patch samples to add to foreground and background:

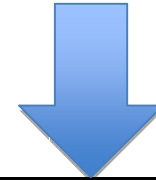
- Remove ambiguous patches (that match both fg and bg)
- Trim fg and bg patches based on sorted knn distances. Remove those with small distances (redundant) as well as large distances (outliers).
- Add clean patches to existing bag of patches.
- Resample patches, with probability of survival proportional to distance of a patch from any patch in current image (tends to keep patches that are currently relevant).



# Sample Results



Extension to video segmentation.  
See paper for the details.



SU-VLPR'09, Beijing

Collins, PSU

# Segmentation-based Tracking

This brings up a second general scheme for drift avoidance besides anchoring, which is to perform fg/bg segmentation.

In principle, it is could be a better solution, because your model is not constrained to stay near one spot, and can therefore handle arbitrarily large appearance change.

Simple examples of this strategy use motion segmentation (change detection) and data association.



# Segmentation-based Tracking



Yin and Collins. “Belief propagation in a 3d spatio-temporal MRF for moving object detection.” CVPR 2007.

Yin and Collins. “Online figure-ground segmentation with edge pixel classification.” BMVC 2008.

# Segmentation-based Tracking



Yin and Collins. "Shape constrained figure-ground segmentation and tracking." CVPR 2009.

# Tracking and Object Detection

Another way to avoid drift is to couple an object detector with the tracker.

Particularly for face tracking or pedestrian tracking, a detector is sometimes included in the tracking loop e.g. Yuan Li's Cascade Particle Filter (CVPR 2007) or K.Okuma's Boosted Particle Filter (ECCV 2004).

- If detector produces binary detections (I see three faces: here, and here, and here), use these as input to a data association algorithm.
- If detector produces a continuous response map, use that as input to a mean-shift tracker.

# Summary

Tracking is still an active research topic.

Topics of particular current interest include:

- Multi-object tracking (including multiple patches on one object)

- Synergies between

Classification and Tracking

Segmentation and Tracking

Detection and Tracking

All are aimed at achieving long-term persistent tracking in ever-changing environments.