

Robert Collins
Penn State

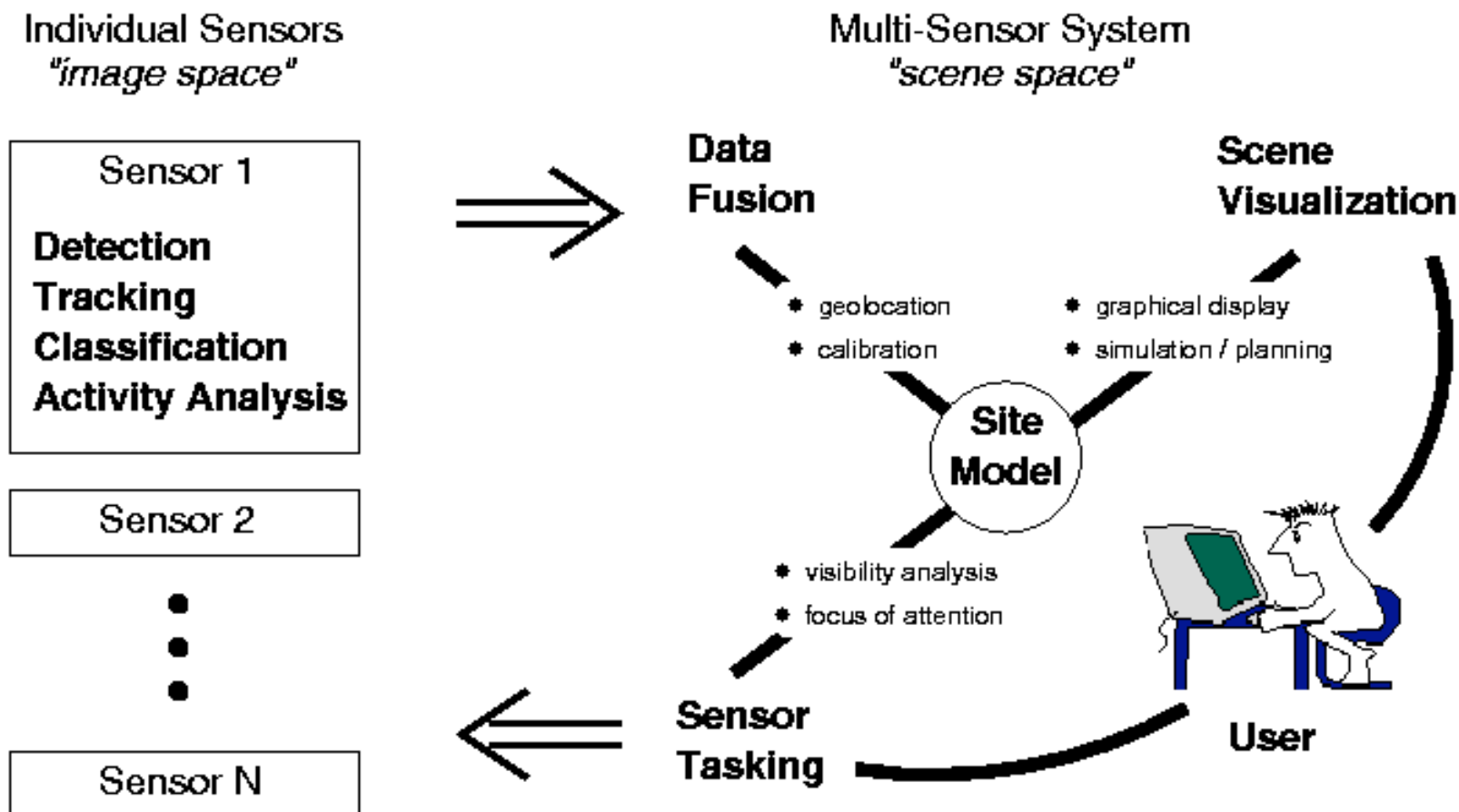
Fundamentals of Video Surveillance

Sino-USA Summer School
July 2010

Robert Collins
Penn State University
University Park, PA, USA

Automated Video Surveillance

Primary goal is situation awareness: fusing information from multiple sensors into a coherent model of actors, actions and events to help a remote user to understand what is happening (e.g. who is where; who is doing what to whom).

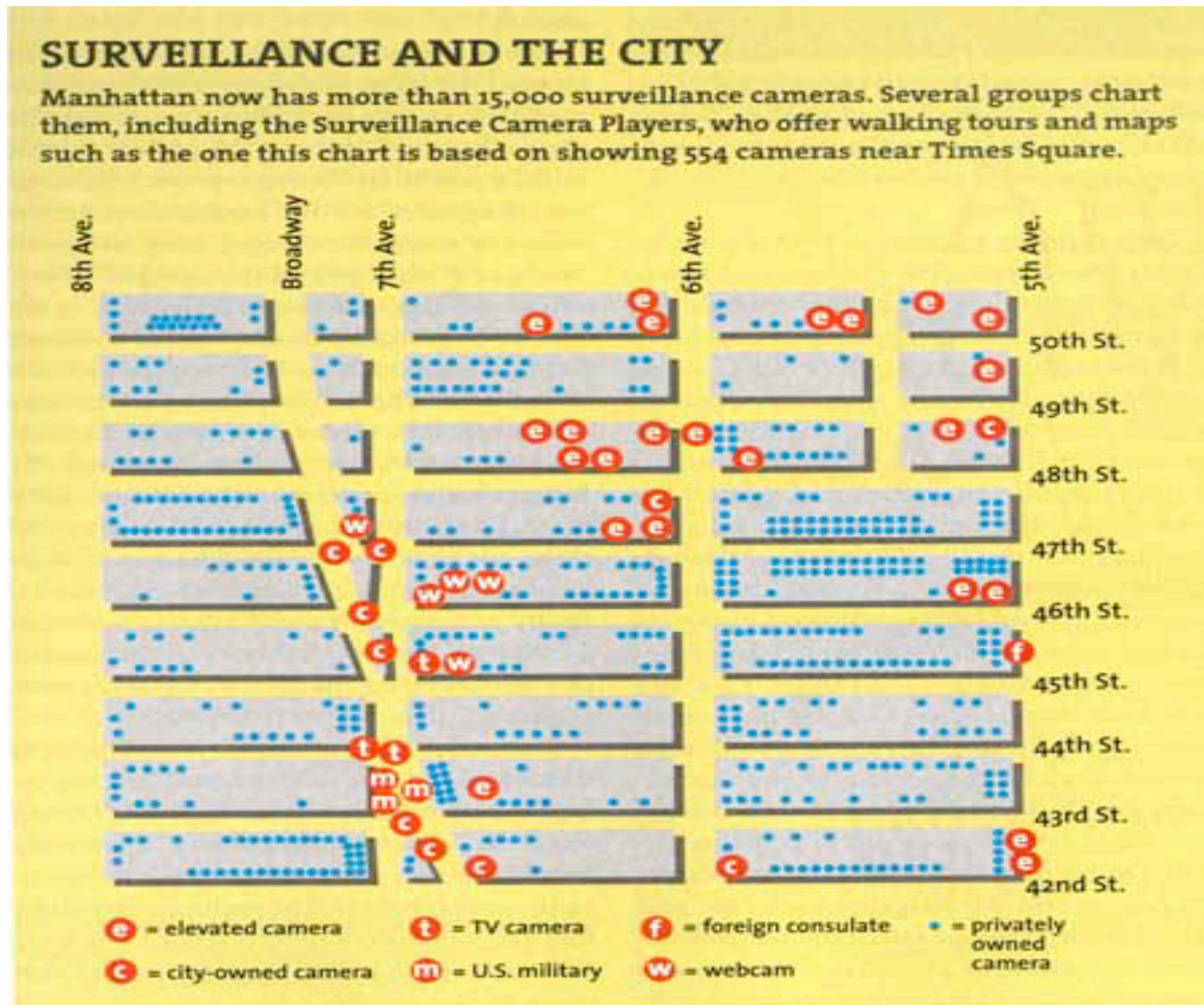


Robert Collins
Penn State

Motivation



Motivation



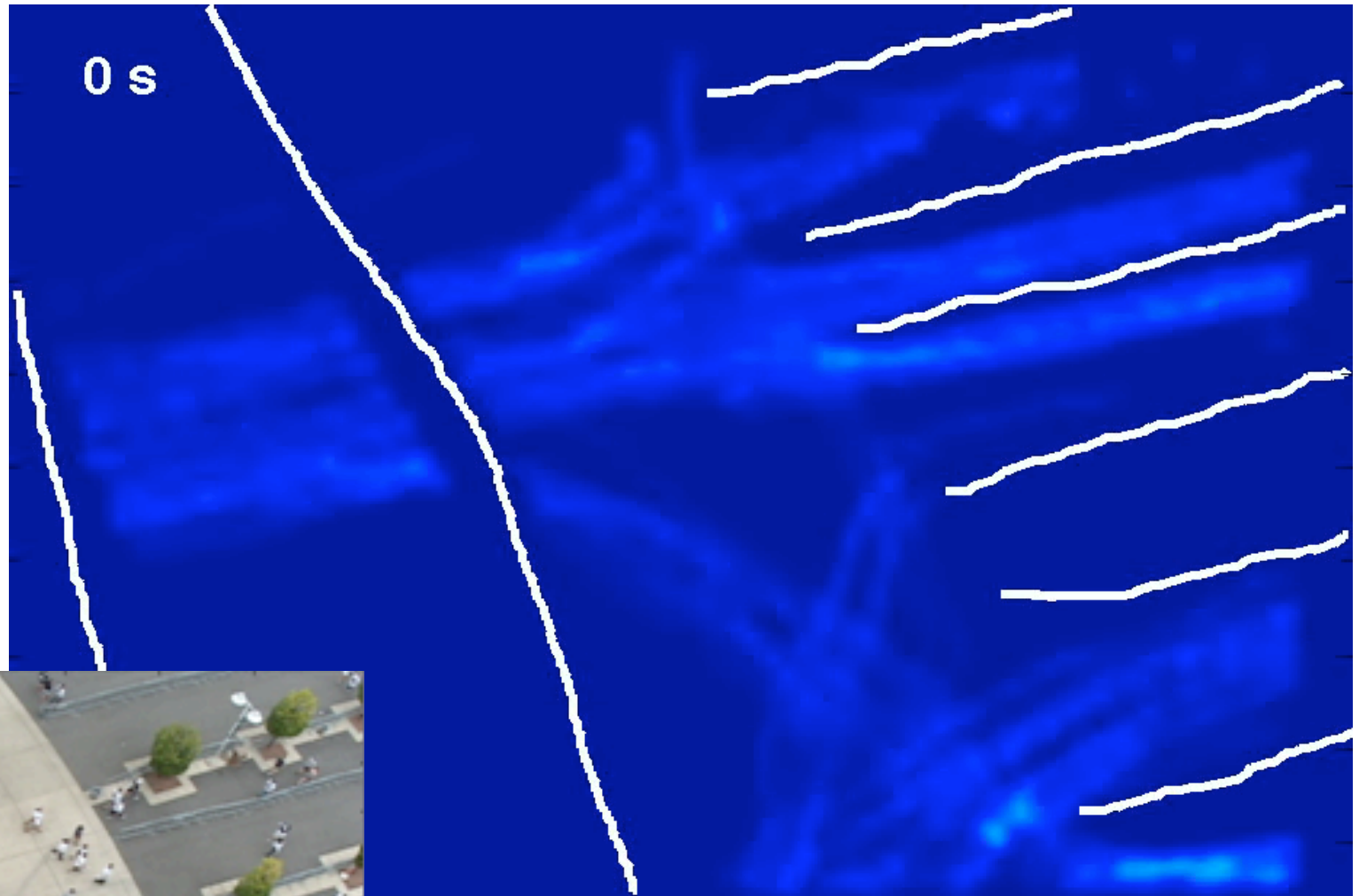
Example: Crowd Analysis

Automated video analysis of crowds in public spaces using computer vision tools

- Real-time monitoring
 - situation awareness
 - notification / alarms
- After-action review
 - trend analysis
 - analyze abnormal events

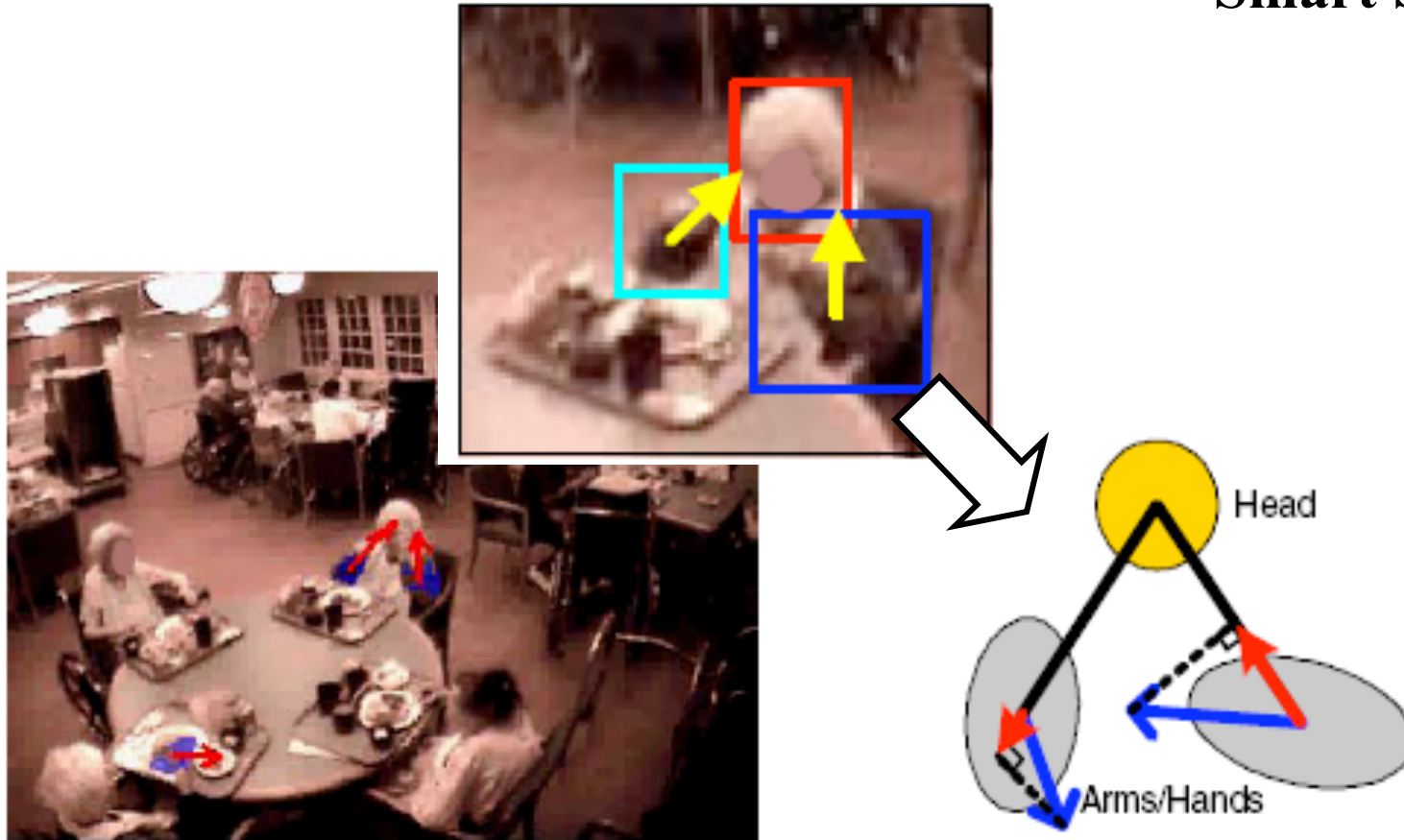


Measuring Crowd Flow/Density



Motivation

- Assisted Living
- “Smart Spaces”



Alzheimer's ward of
a nursing home

J.Gao, R.Collins, A.Hauptmann and H.Wactler,
"Articulated Motion Modeling for Activity Analysis,"
IEEE Workshop on Articulated and NonRigid Motion, in
conjunction with CVPR'04, Washington, DC, 2004.

Motivation



Overview

Part 1: Change/Motion Detection

Basics: BG subtraction; Frame Difference

Classification-based methods

Part 2: From Pixels to 2D Blobs

Detection via RJMCMC

Classifier Grids

Part 3: Data Association

Linear Assignment Problem

Murty K-best; PDAF; JPDAF

Part 4: Persistent Tracking

Adaptive Tracking

Tracking as Classification

Overview

Part 1: Change/Motion Detection

Basics: BG subtraction; Frame Difference

Classification-based methods

Part 2: From Pixels to 2D Blobs

Detection via RJMCMC

Classifier Grids

Part 3: Data Association

Linear Assignment Problem

Murty K-best; PDAF; JPDAF

Part 4: Persistent Tracking

Adaptive Tracking

Tracking as Classification

Overview

Part 1: Change/Motion Detection

Basics: BG subtraction; Frame Difference

Classification-based methods

Part 2: From Pixels to 2D Blobs

Detection via RJMCMC

Classifier Grids

Part 3: Data Association

Linear Assignment Problem

Murty K-best; PDAF; JPDAF

Part 4: Persistent Tracking

Adaptive Tracking

Tracking as Classification

Overview

Part 1: Change/Motion Detection

Basics: BG subtraction; Frame Difference

Classification-based methods

Part 2: From Pixels to 2D Blobs

Detection via RJMCMC

Classifier Grids

Part 3: Data Association

Linear Assignment Problem

Murty K-best; PDAF; JPDAF

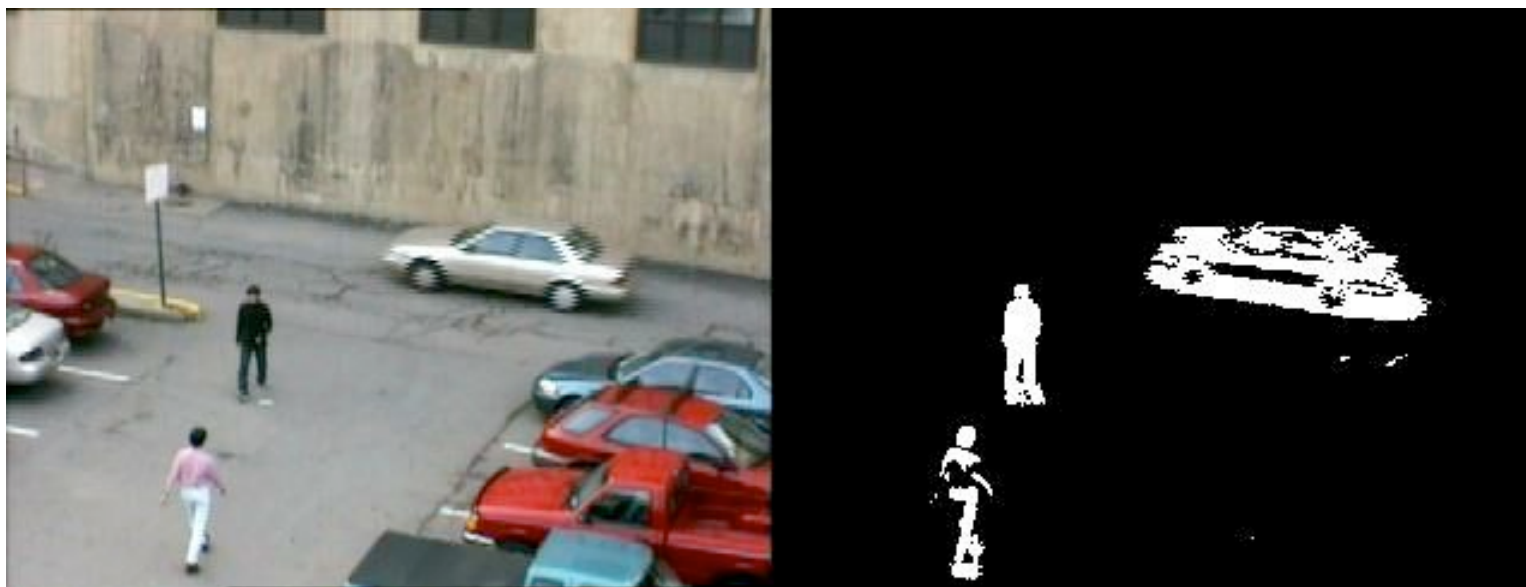
Part 4: Persistent Tracking

Adaptive Tracking

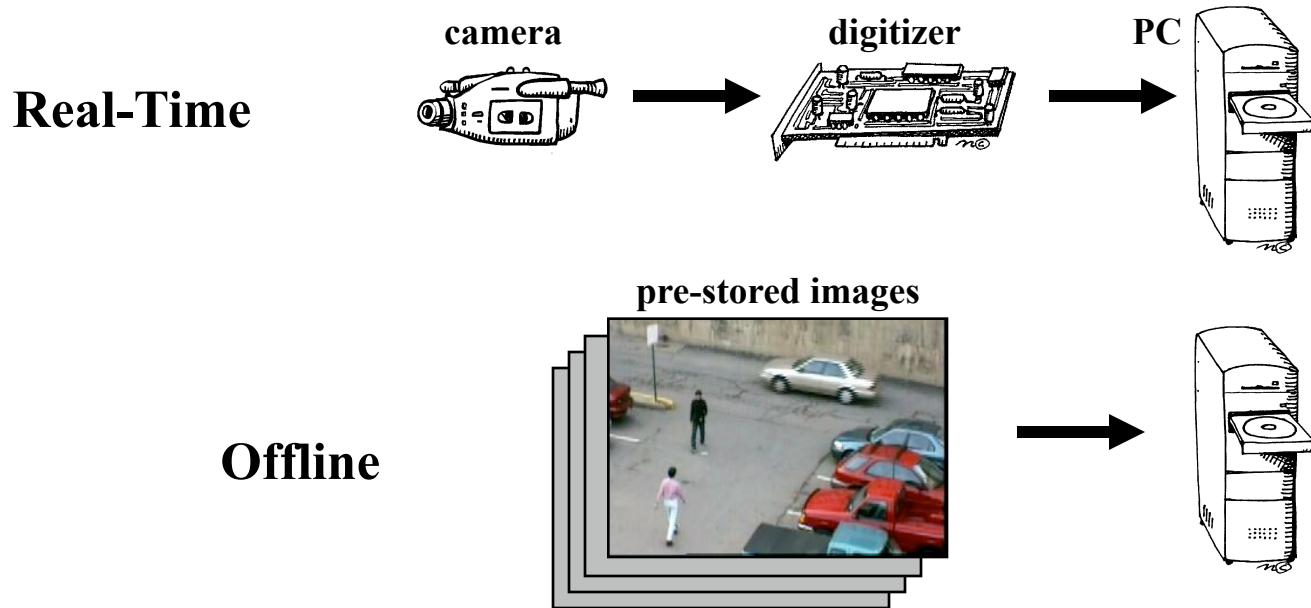
Tracking as Classification

Part1 : Change Detection

Goal: Learn methods for pixel-level motion / change detection
Understand pros and cons of basic approaches



Basics of Video



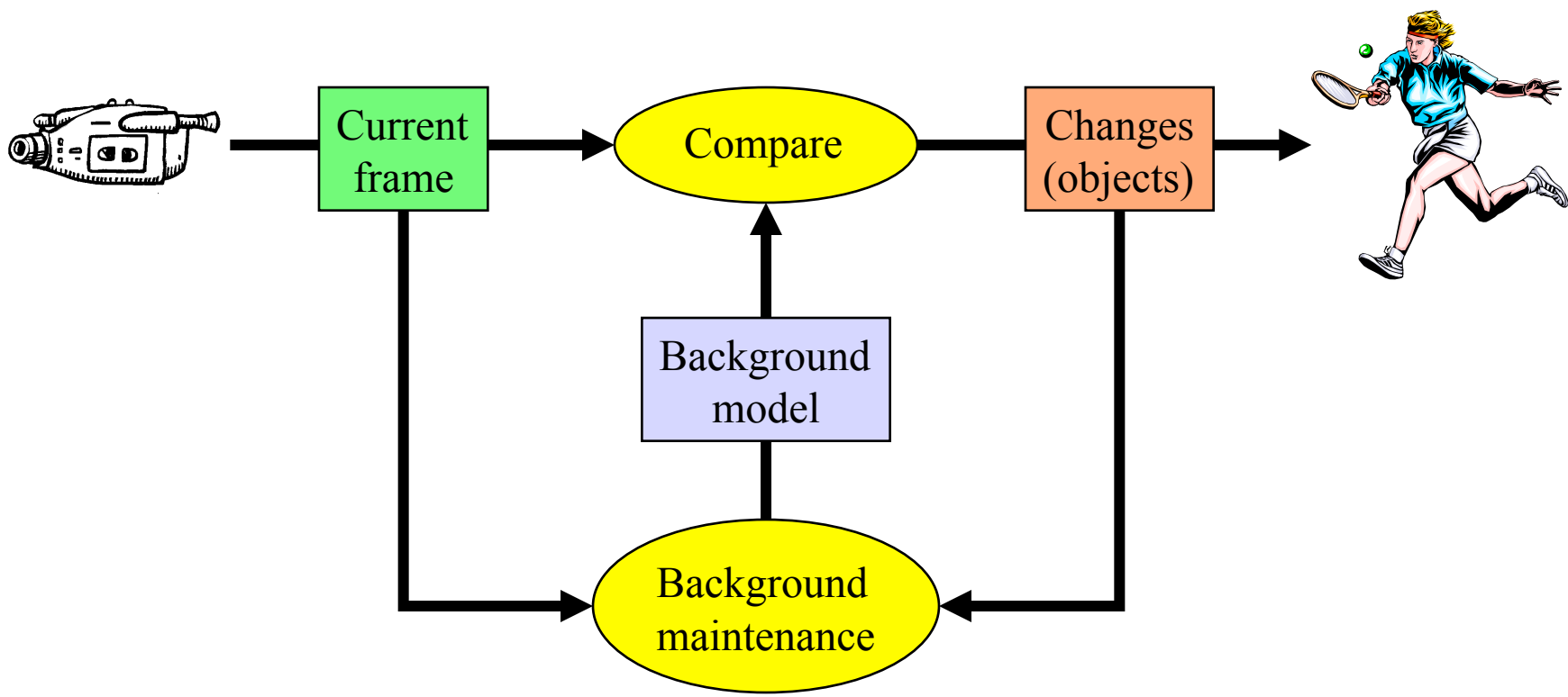
Frames come in 30 times per second. This is not much time to process each image. Real-time algorithms therefore tend to be very simple.

One of the main features of video imagery is the temporal consistency from frame to frame. Not much changes during $1/30$ of a second!

Detecting Moving Objects

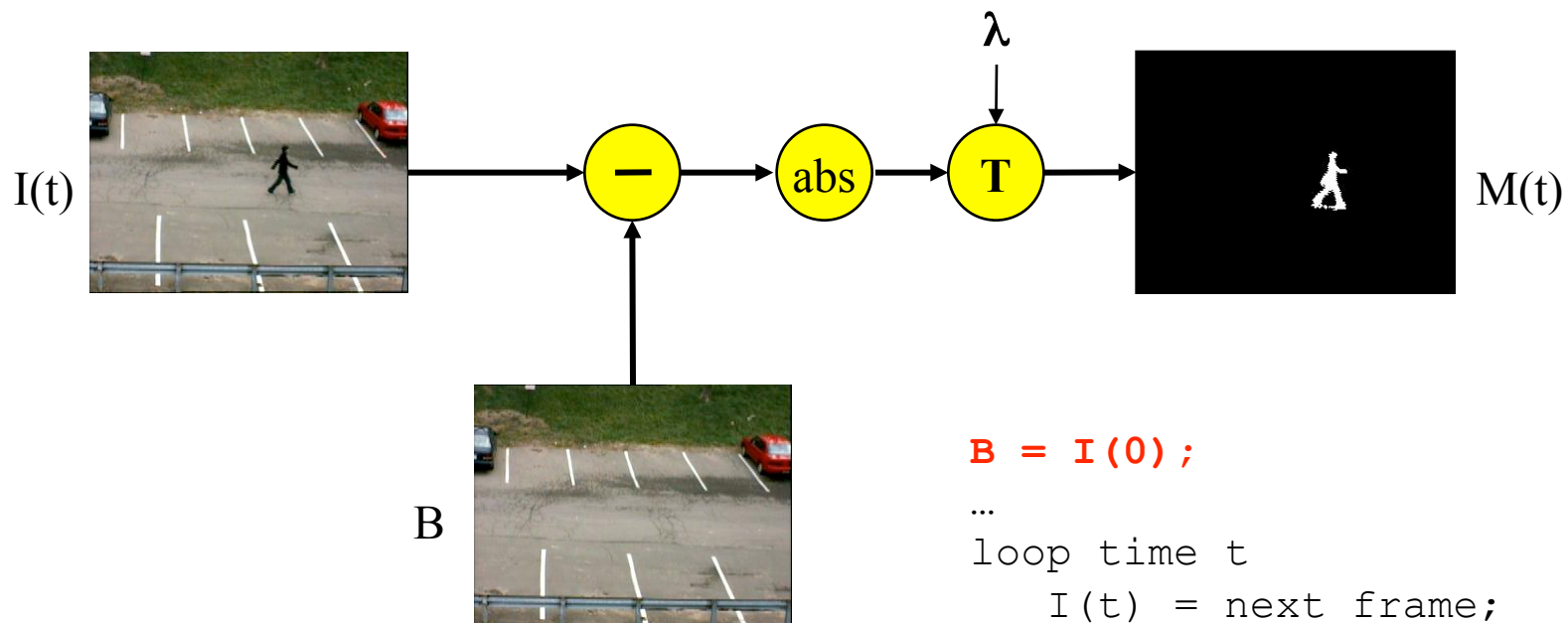
Assumption: objects that move are important (e.g. people and vehicles)

Basic approach: maintain a model of the static background. Compare the current frame with the background to locate moving foreground objects.



Simple Background Subtraction

- Background model is a static image (assumed to have no objects present).
- Pixels are labeled as object (1) or not object (0) based on thresholding the absolute intensity difference between current frame and background.



```
B = I(0);
```

```
...
```

```
loop time t
```

```
    I(t) = next frame;
```

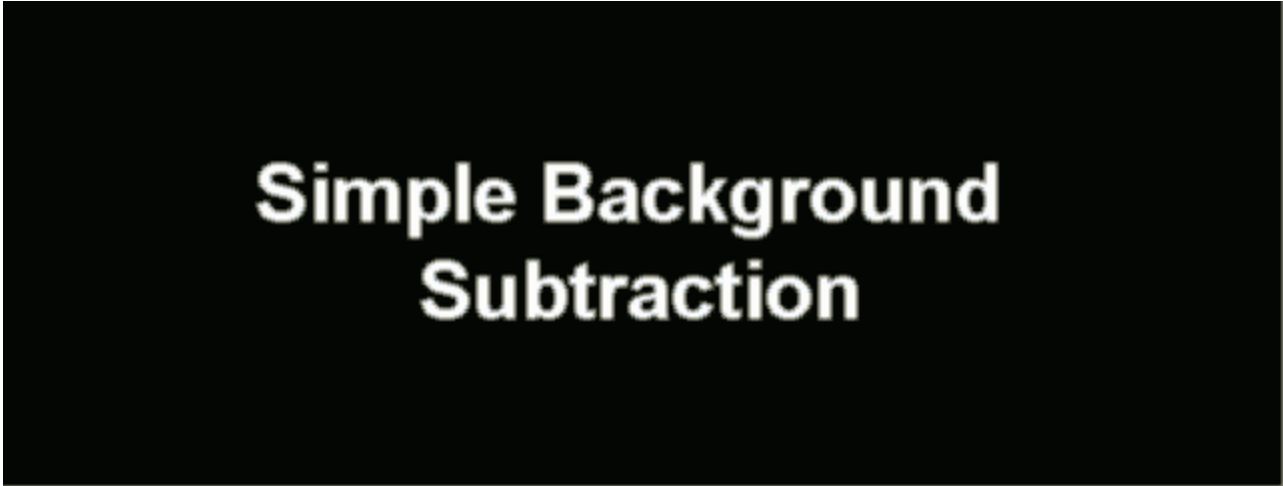
```
    diff = abs[B - I(t)];
```

```
    M(t) = threshold(diff,  $\lambda$ );
```

```
...
```

```
end
```


Background Subtraction Results



**Simple Background
Subtraction**

movie

BG Observations

Background subtraction does a reasonable job of extracting the shape of an object, provided the object intensity/color is sufficiently different from the background.



BG Observations



Objects that enter the scene and stop continue to be detected, making it difficult to detect new objects that pass in front of them.

If part of the assumed static background starts moving, both the object and its negative ghost (the revealed background) are detected



BG Observations



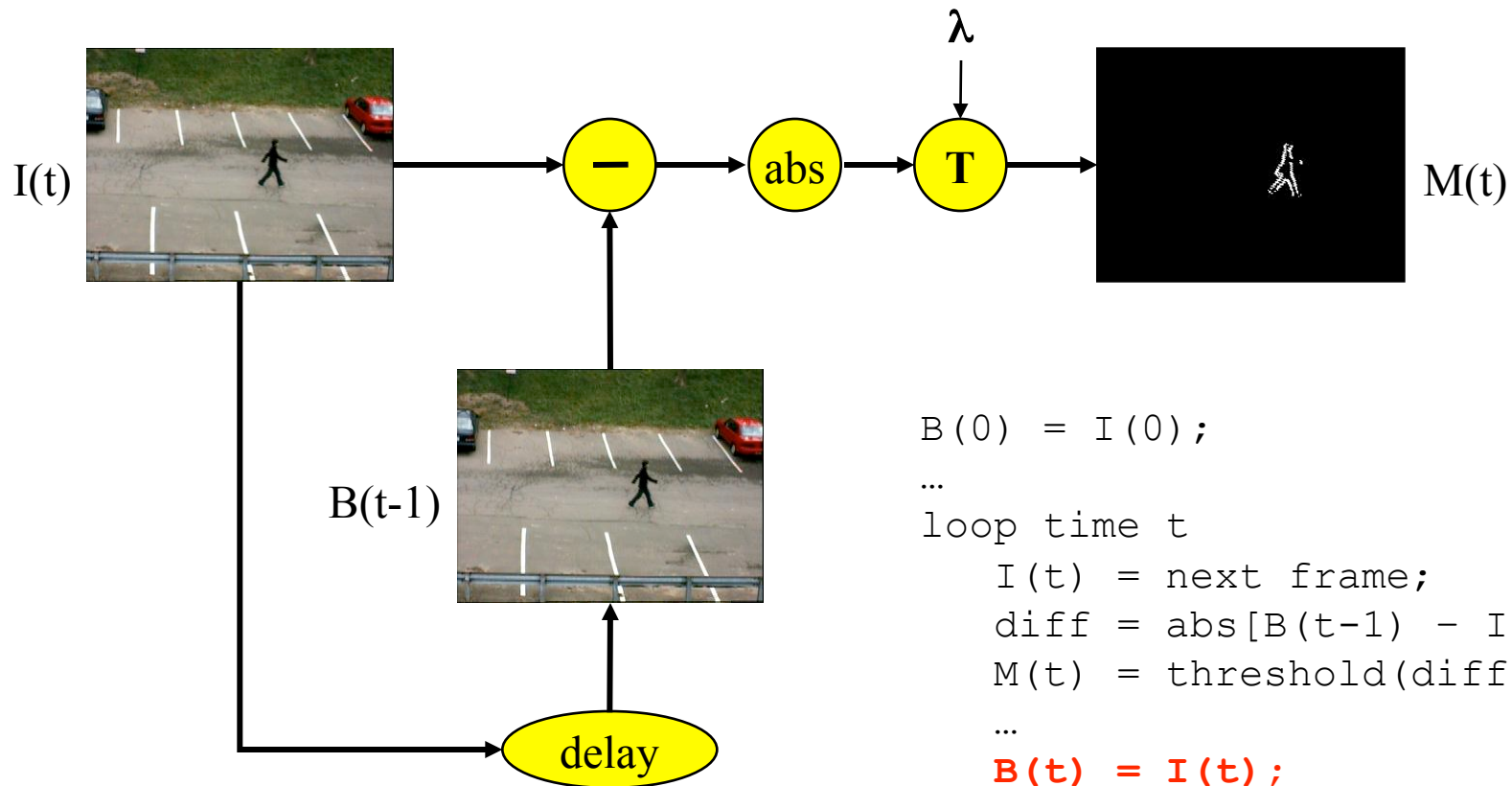
Background subtraction is sensitive to changing illumination and unimportant movement of the background (for example, trees blowing in the wind, reflections of sunlight off of cars or water).



Background subtraction cannot handle movement of the camera.


Simple Frame Differencing

- Background model is replaced with the previous image.



```
B(0) = I(0);  
...  
loop time t  
    I(t) = next frame;  
    diff = abs[B(t-1) - I(t)];  
    M(t) = threshold(diff,  $\lambda$ );  
    ...  
    B(t) = I(t);  
end
```

Frame Differencing Results



Simple Frame
Differencing

movie

FD Observations

Frame differencing is very quick to adapt to changes in lighting or camera motion.

Objects that stop are no longer detected. Objects that start up do not leave behind ghosts.

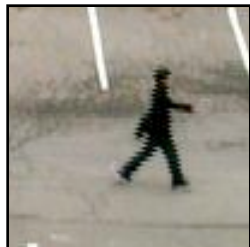
However, frame differencing only detects the leading and trailing edge of a uniformly colored object. As a result very few pixels on the object are labeled, and it is very hard to detect an object moving towards or away from the camera.



Differencing and Temporal Scale

Note what happens when we adjust the temporal scale (frame rate) at which we perform two-frame differencing ...

$$\text{Define } D(N) = \| I(t) - I(t+N) \|\$$



I(t)



D(-1)



D(-3)



D(-5)



D(-9)



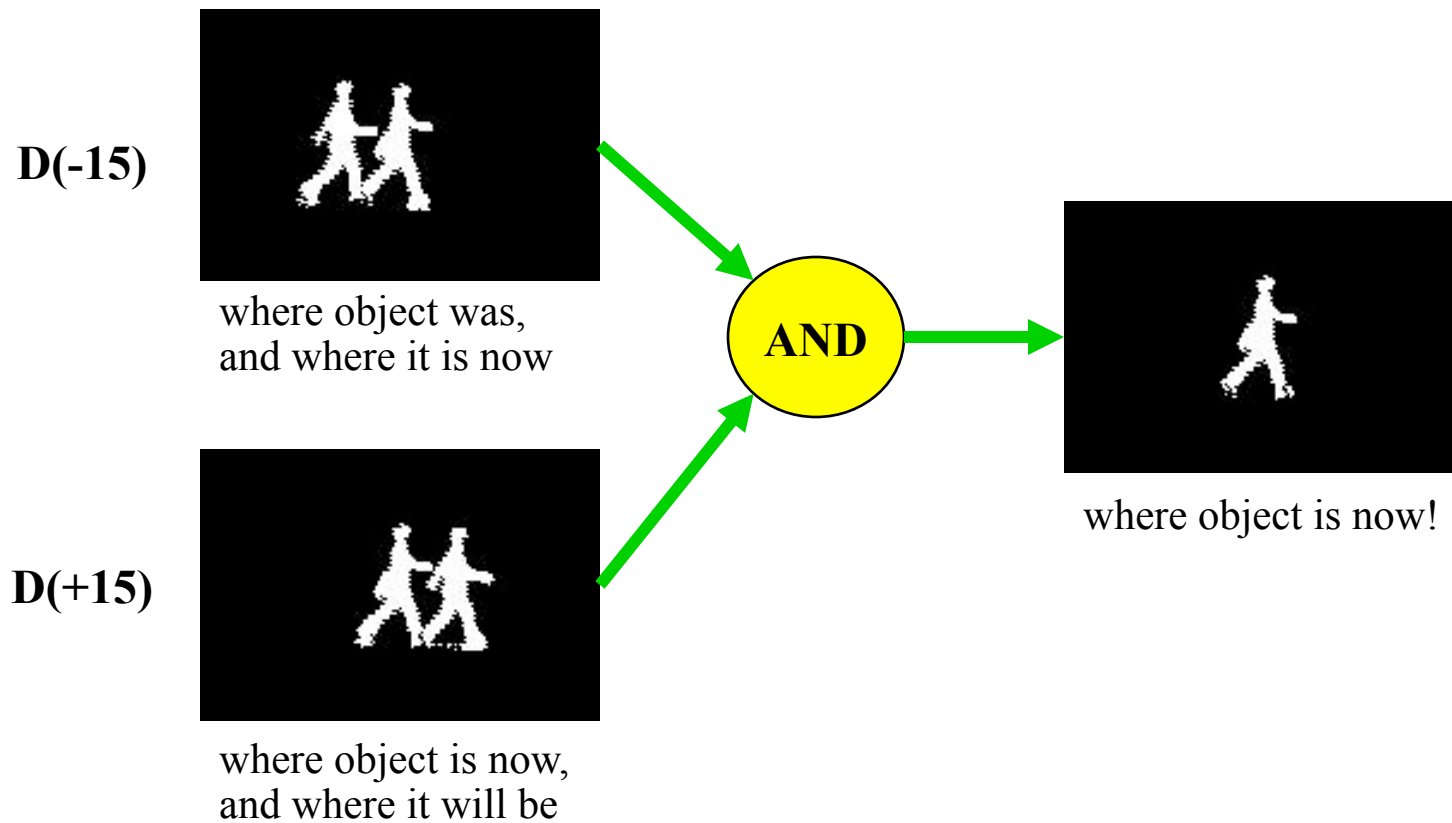
D(-15)



more complete object silhouette, but two copies
(one where object used to be, one where it is now).

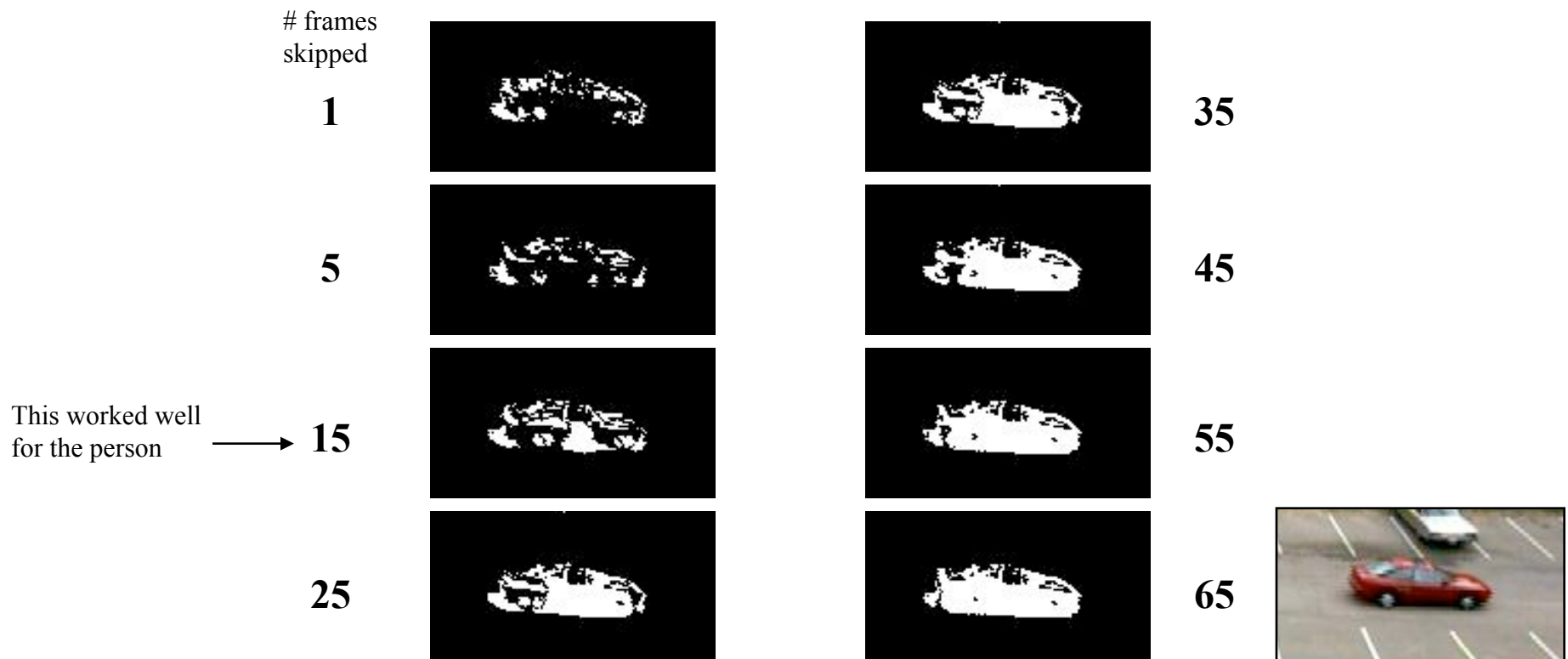
Three-Frame Differencing

The previous observation is the motivation behind three-frame differencing



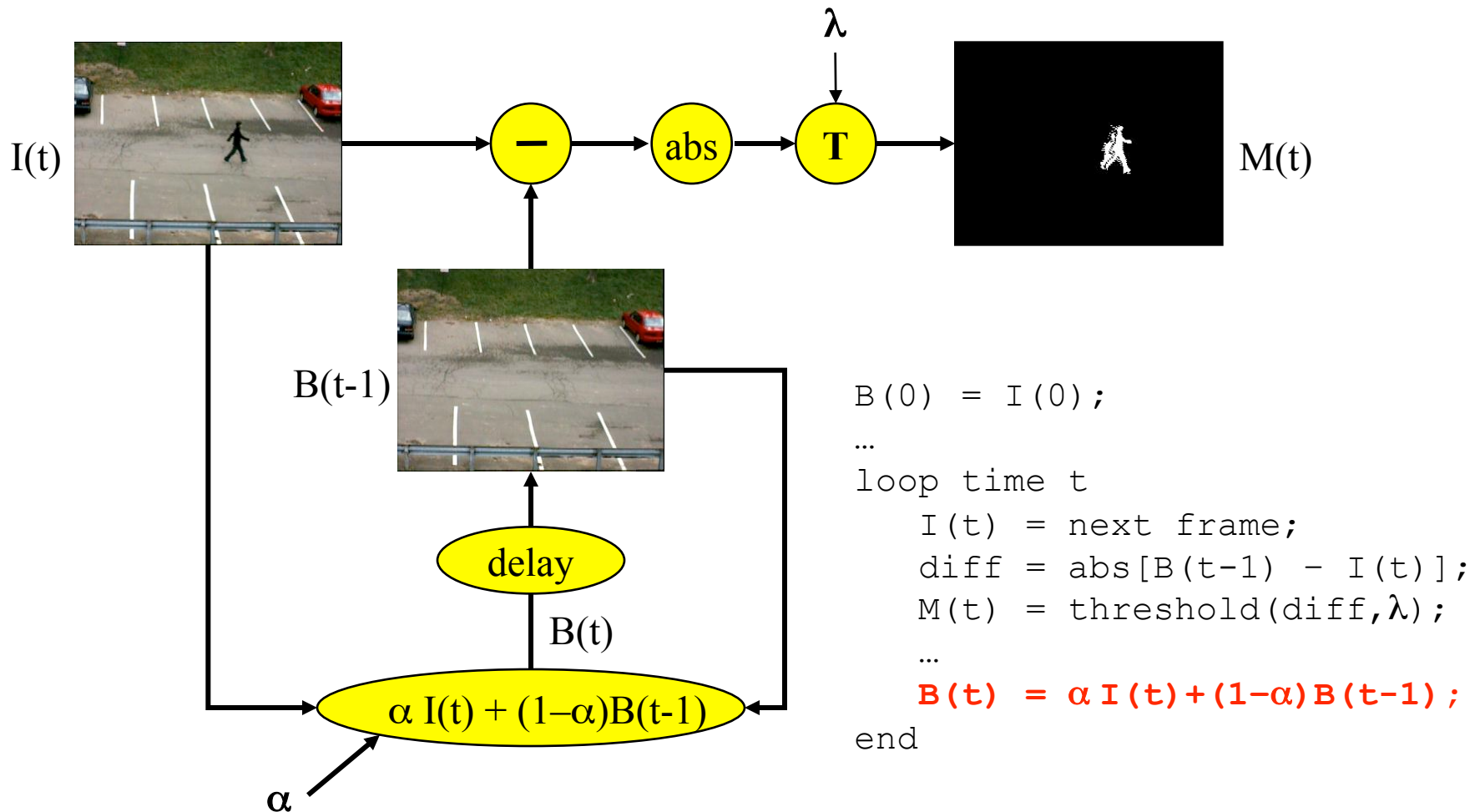
Three-Frame Differencing

Choice of good frame-rate for three-frame differencing depends on the size and speed of the object

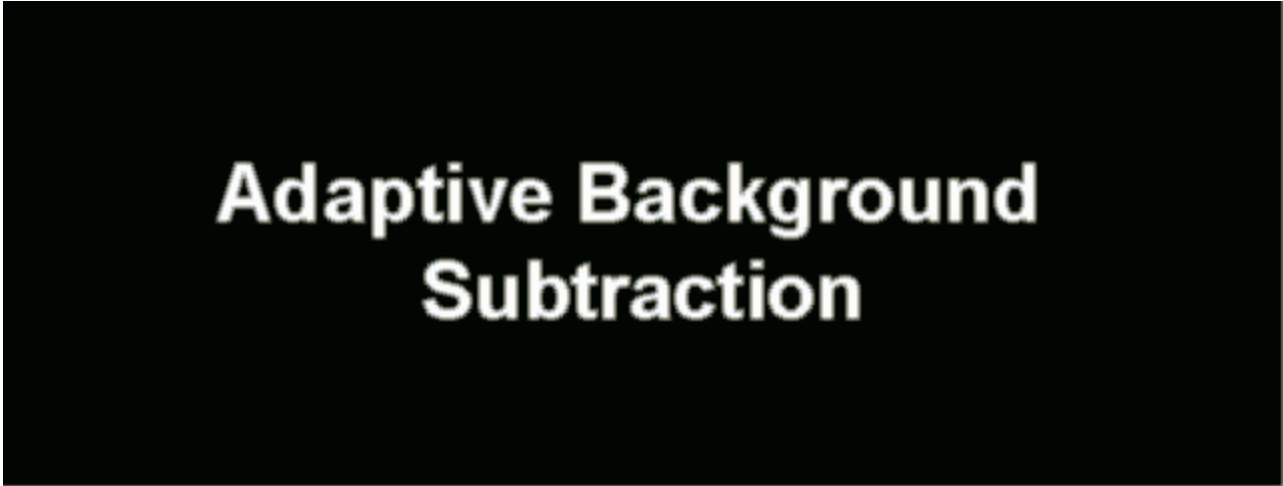


Adaptive Background Subtraction

- Current image is “blended” into the background model with parameter α
- $\alpha = 0$ yields simple background subtraction, $\alpha = 1$ yields frame differencing



Adaptive BG Subtraction Results



**Adaptive Background
Subtraction**

movie

Adaptive BG Observations

Adaptive background subtraction is more responsive to changes in illumination and camera motion.

Fast small moving objects are well segmented, but they leave behind short “trails” of pixels.

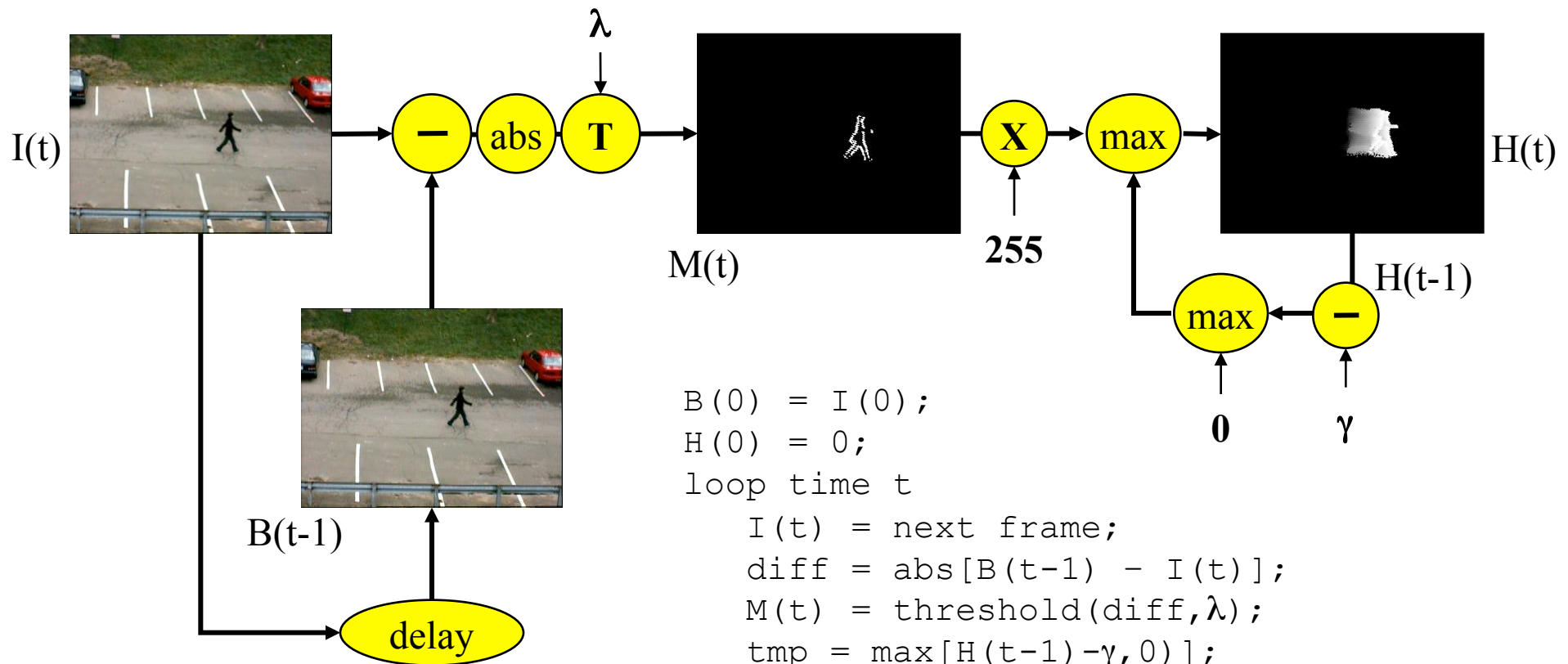
Objects that stop, and ghosts left behind by objects that start, gradually fade into the background.

The centers of large slow moving objects start to fade into the background too! This can be “fixed” by decreasing the blend parameter A , but then it takes longer for stopped/ghost objects to disappear.



Persistent Frame Differencing

- Motion images are combined with a linear decay term
- also known as motion history images (Davis and Bobick)



$B(0) = I(0);$

$H(0) = 0;$

loop time t

$I(t) = \text{next frame};$

$\text{diff} = \text{abs}[B(t-1) - I(t)];$

$M(t) = \text{threshold}(\text{diff}, \lambda);$

$\text{tmp} = \text{max}[H(t-1) - \gamma, 0];$


$H(t) = \text{max}[255 * M(t), \text{tmp}];$

...

$B(t) = I(t);$

end

Persistent FD Results



**Persistent Frame
Differencing**

movie

Persistent FD Observations

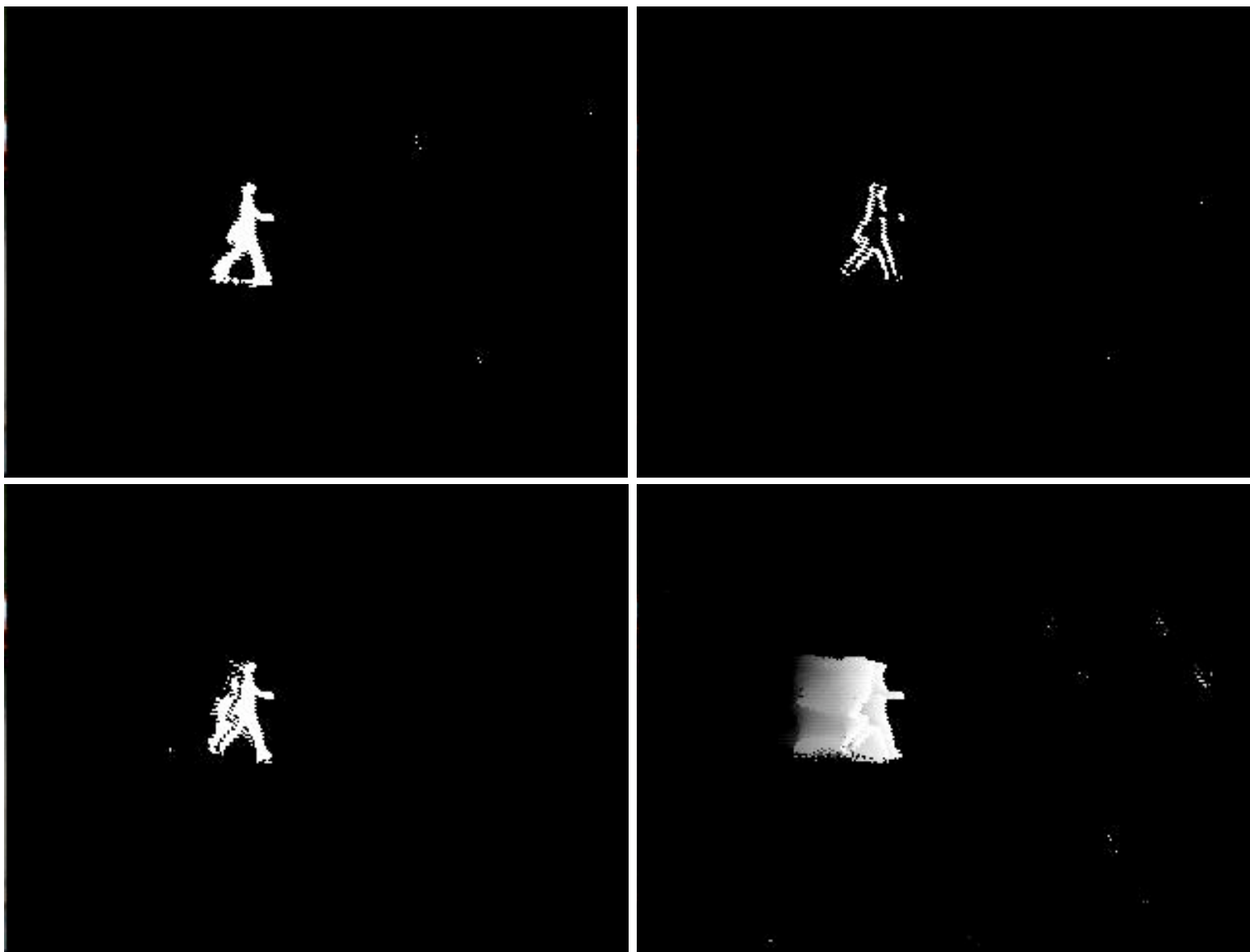
Persistent frame differencing is also responsive to changes in illumination and camera motion, and stopped objects / ghosts also fade away.

Objects leave behind gradually fading trails of pixels. The gradient of this trail indicates the apparent direction of object motion in the image.

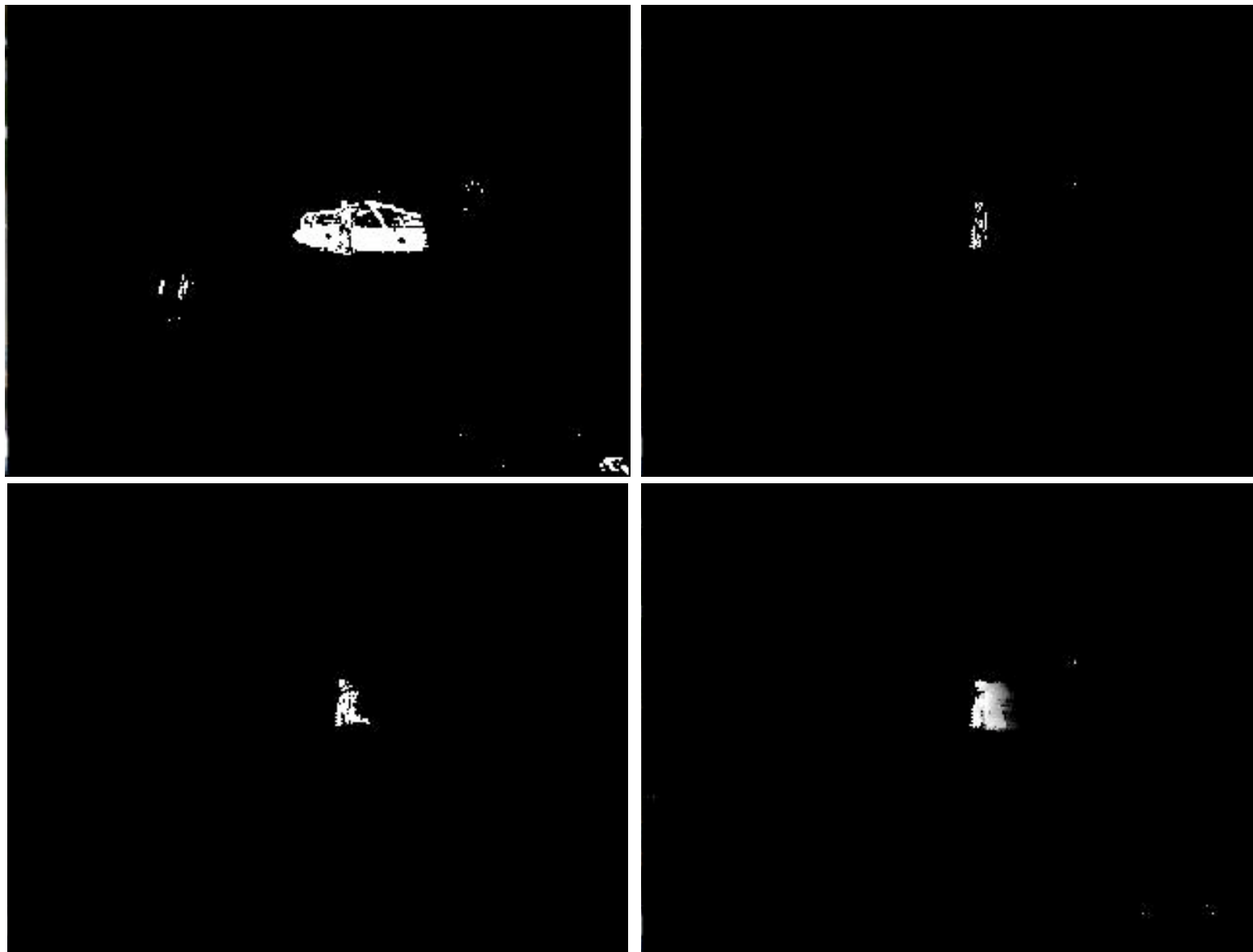
Although the centers of uniformly colored objects are still not detected, the leading and trailing edges are made wider by the linear decay, so that perceptually (to a person) it is easier to see the whole object.



Comparisons



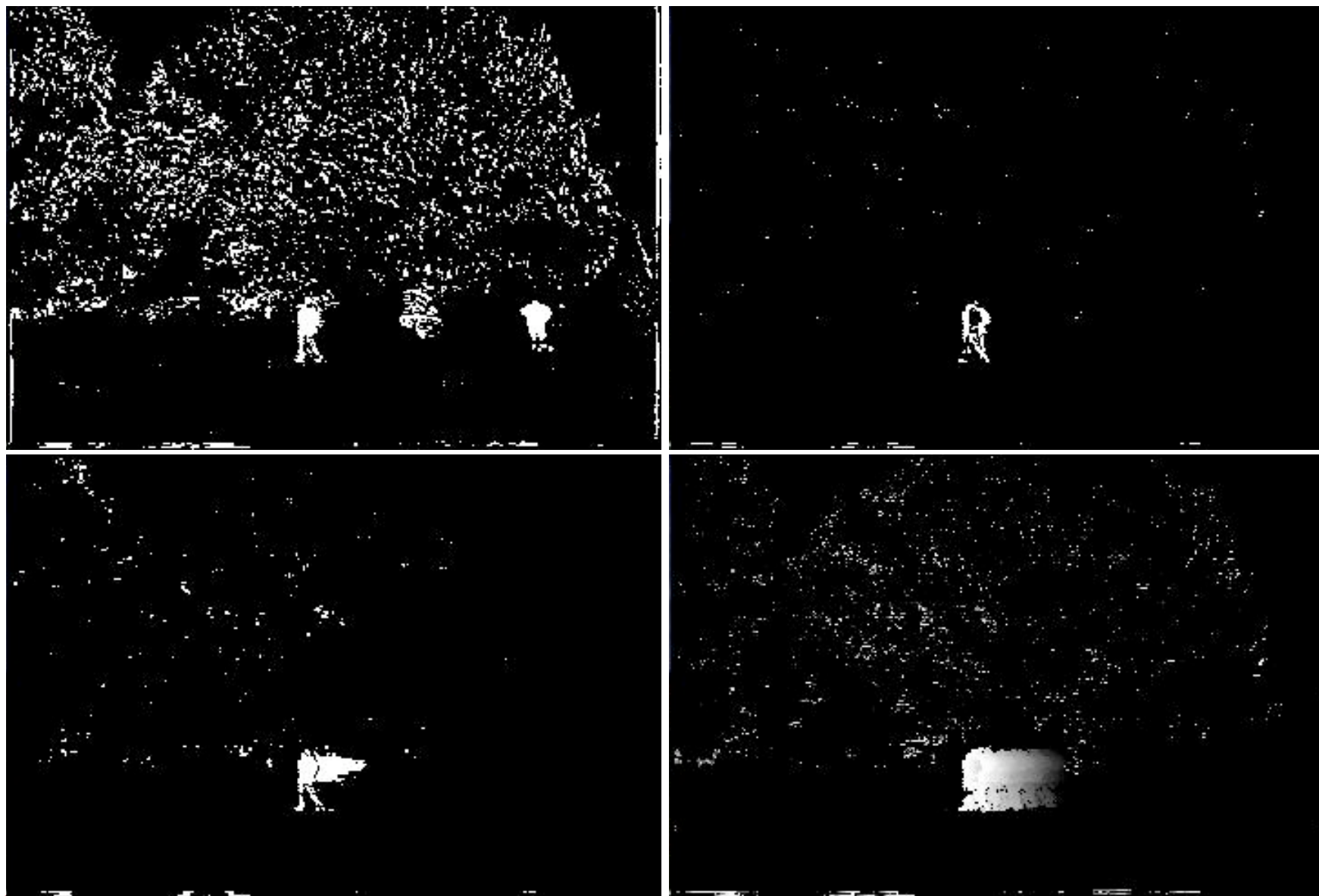
Comparisons



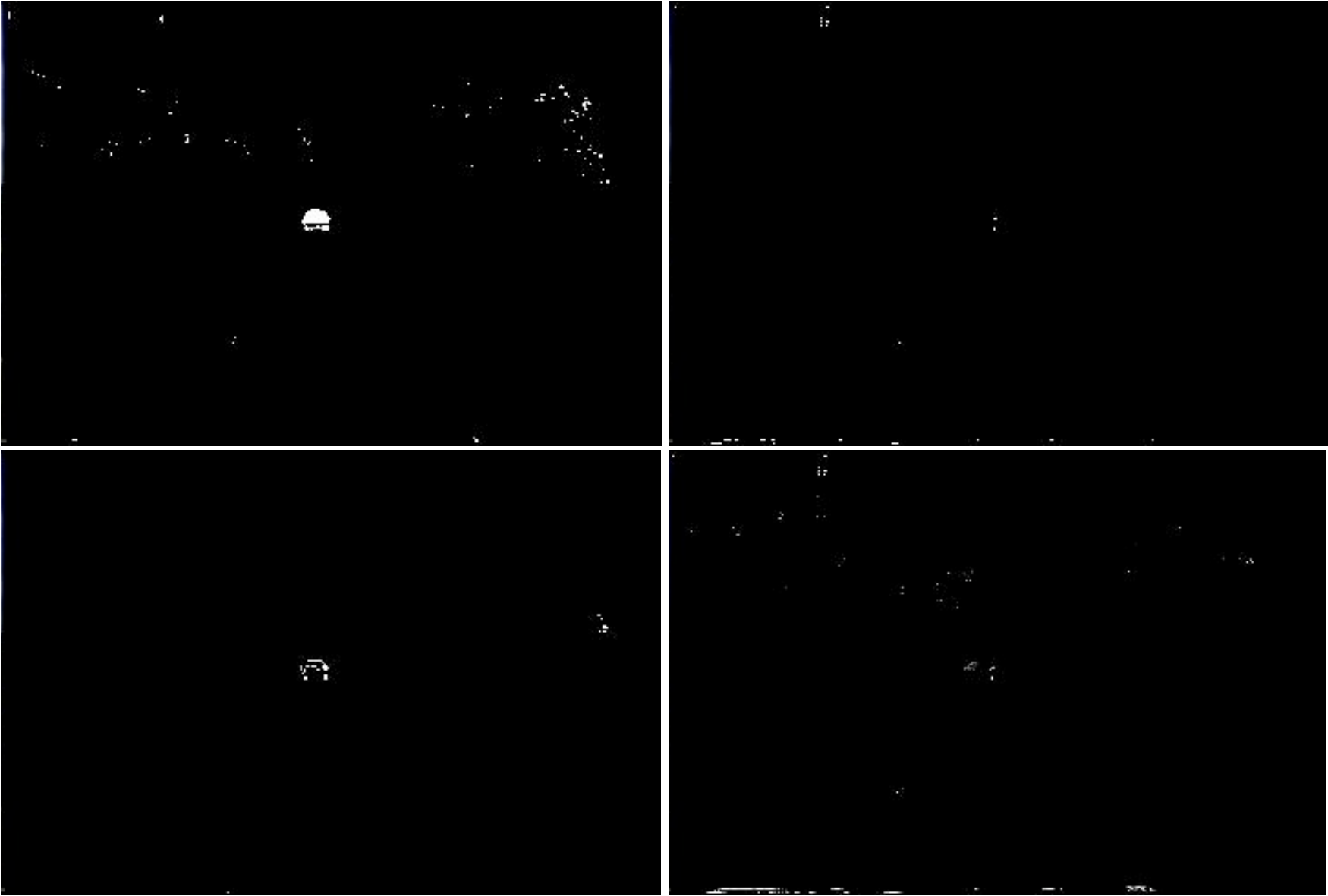
Comparisons



Comparisons

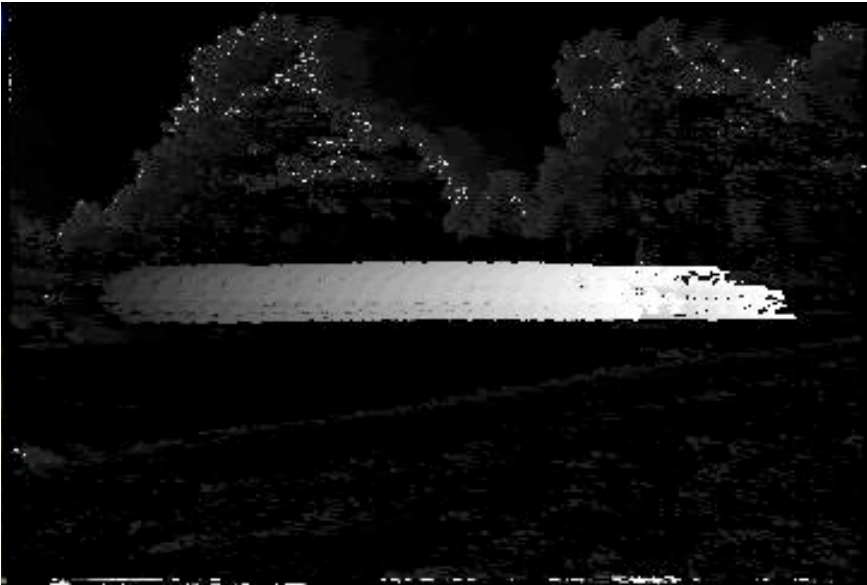
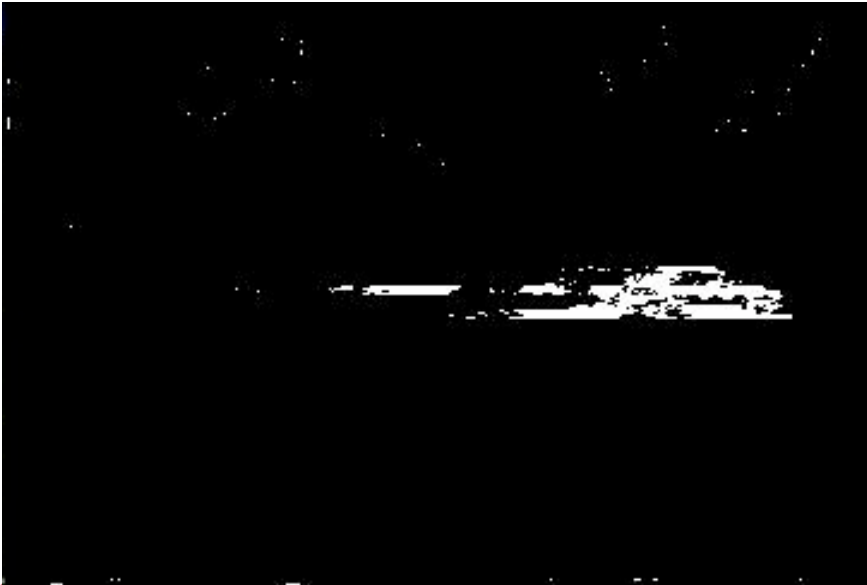


Comparisons



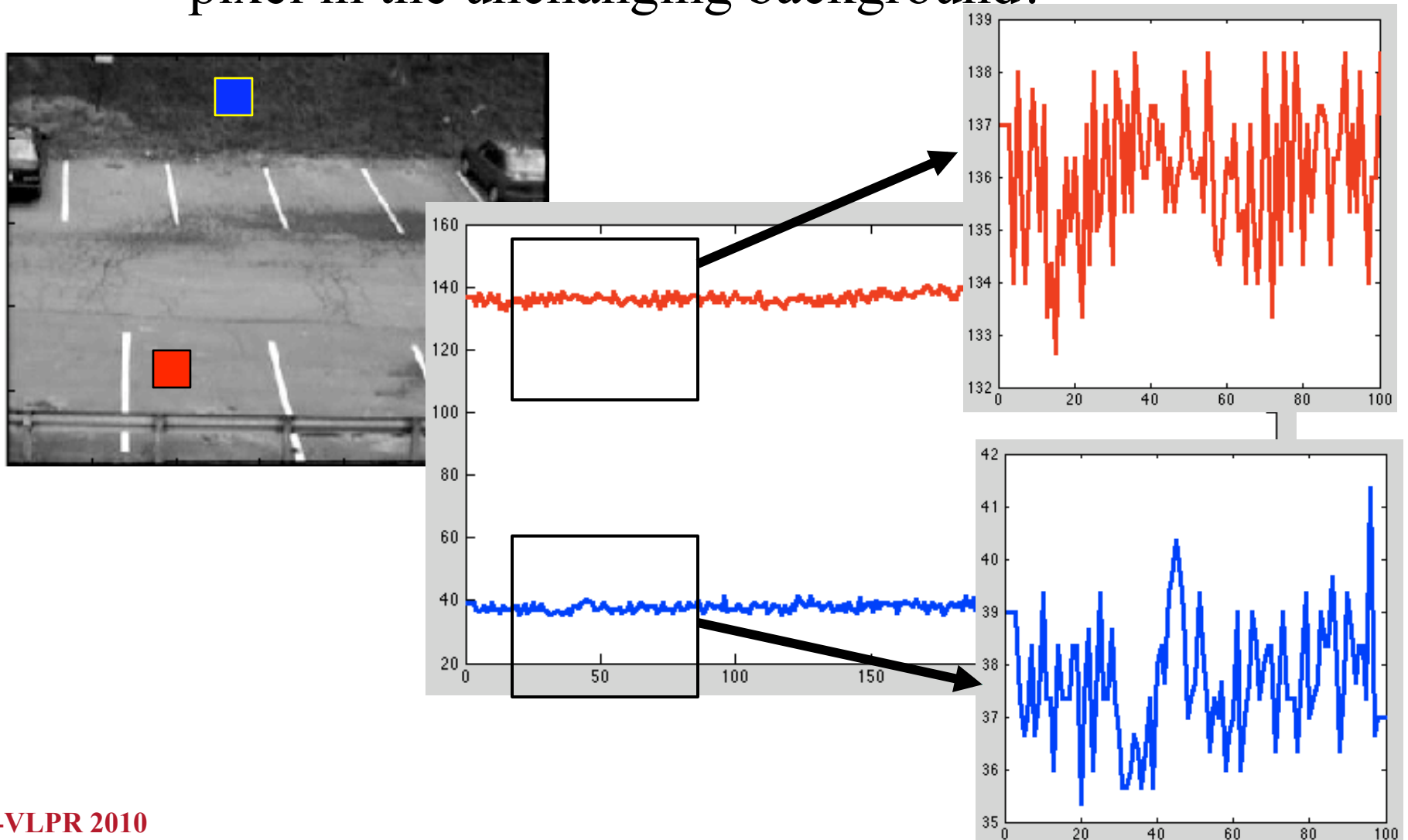
Robert Collins
Penn State

Comparisons



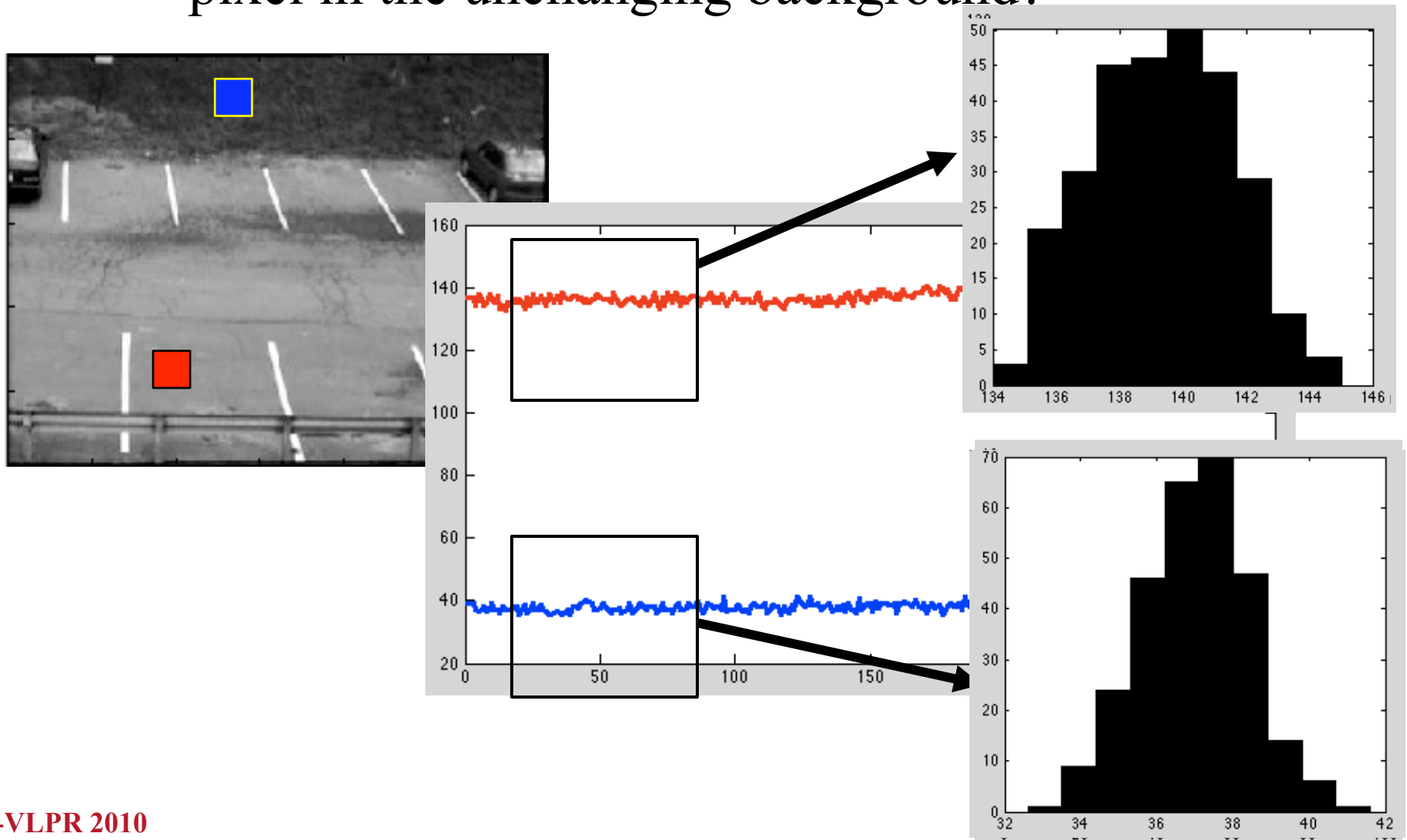
Interpreting Bg Subtraction

- What is a good statistical model of the value of a pixel in the unchanging background?



Interpreting Bg Subtraction

- What is a good statistical model of the value of a pixel in the unchanging background?

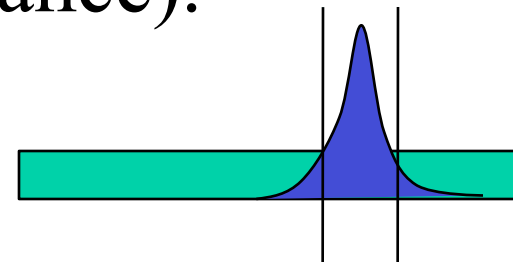


Interpreting Bg Subtraction

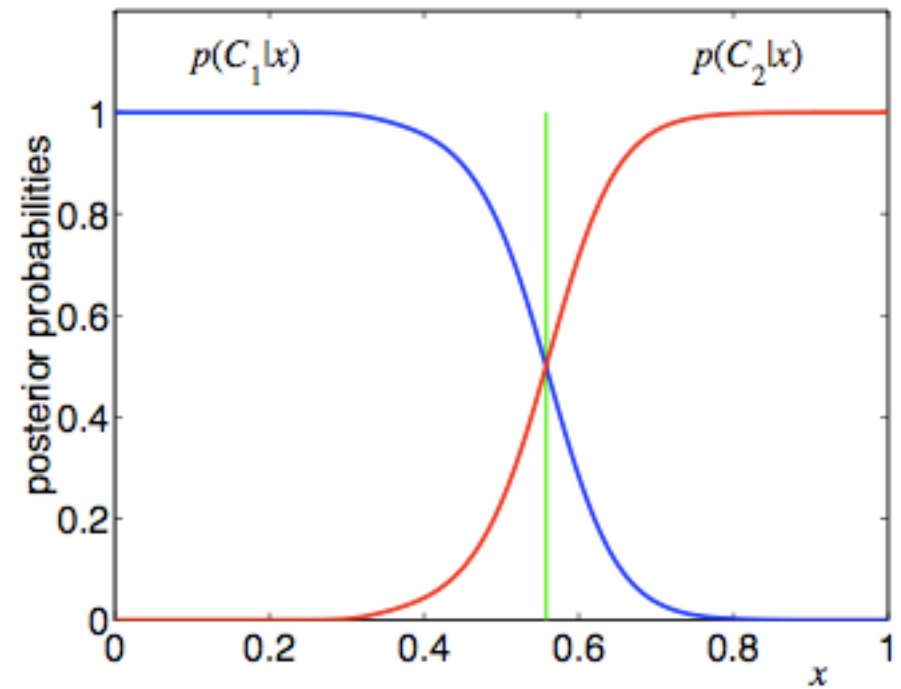
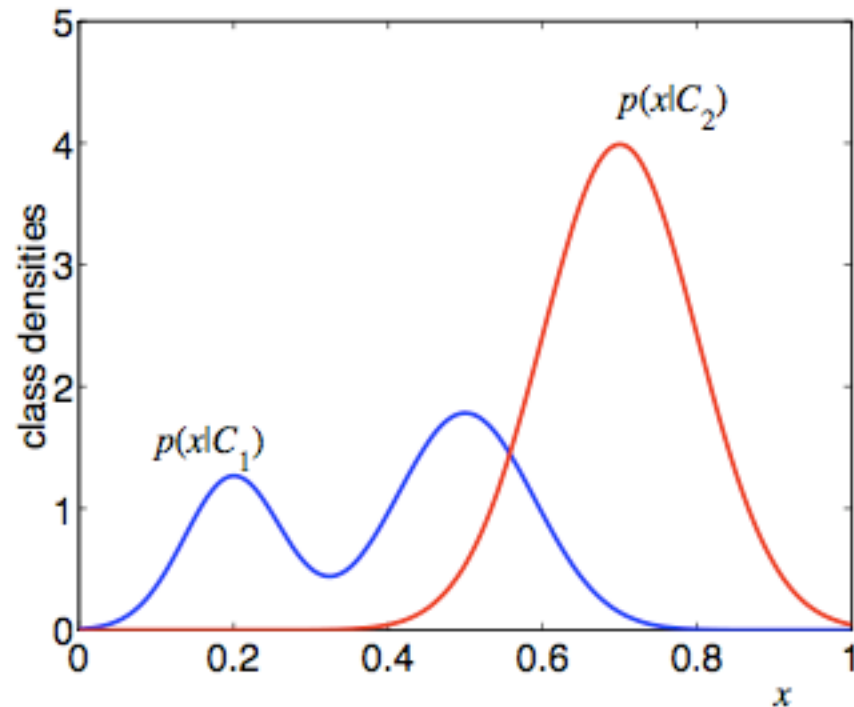
- Gaussian at each pixel. Adaptive mean (recursive estimator), constant variance.
- Detect outlier observations as foreground.

Interpreting Bg Subtraction

- Interpret change detection as two-class classification
- On the board: Work out the math of simple bg subtraction as classification.
- Basic ideas: background model is Gaussian with adaptive mean and constant variance. Foreground model is uniform distribution (or Gaussian with large variance).



Generative vs. Discriminative



Credit: C. Bishop, ICPR 2004

Generative vs. Discriminative Models

- **Generative approach:** separately model class-conditional densities and priors

$$p(\mathbf{x}|\mathcal{C}_k), \quad p(\mathcal{C}_k)$$

- then evaluate posterior probabilities using Bayes' theorem

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

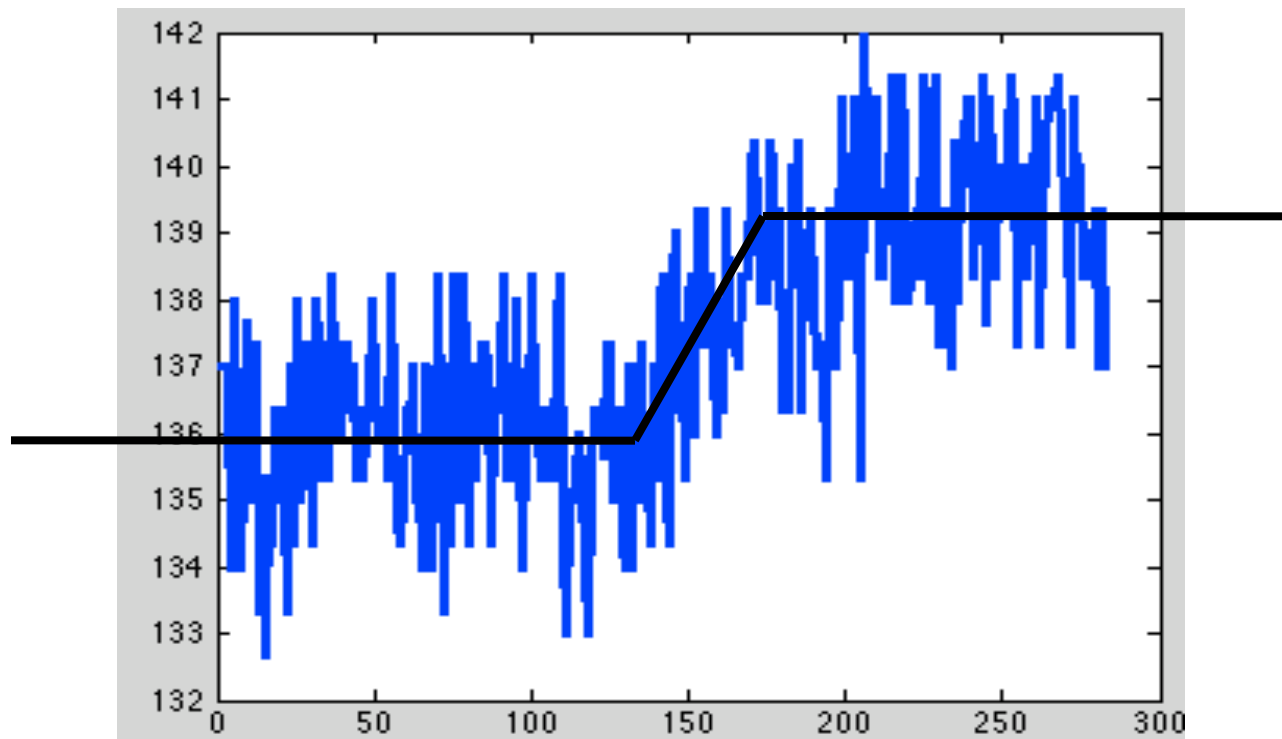
- **Discriminative approach:** directly model posterior probabilities

$$p(\mathcal{C}_k|\mathbf{x})$$

Credit: C. Bishop, ICPR 2004

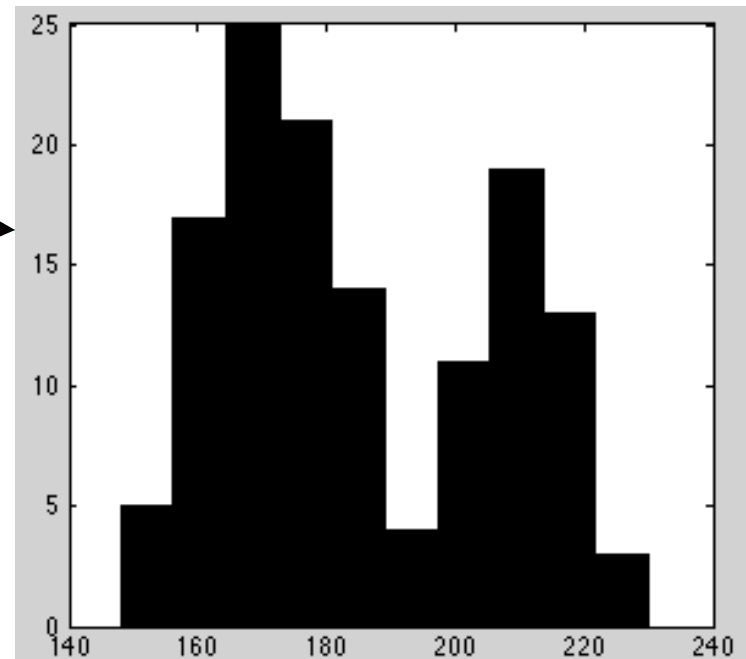
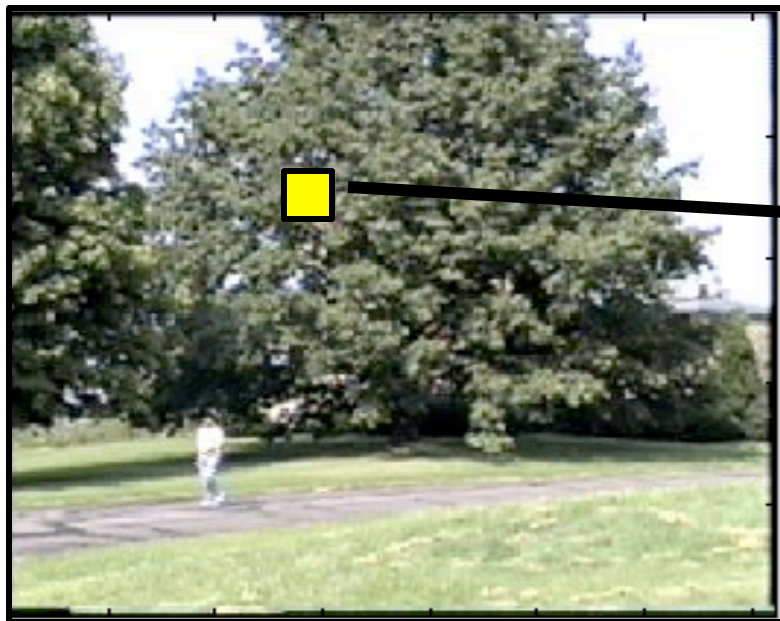
Interpreting Bg Subtraction

- Recursively estimating mean of Gaussian model of background appearance at each pixel (independently)
- Adaptive update of background values automatically takes care of slow appearance changes (e.g. lighting)



Limitation of Gaussian Assumption

- There is a problem with multimodal pixels
- Examples: trees in the wind; rippling water



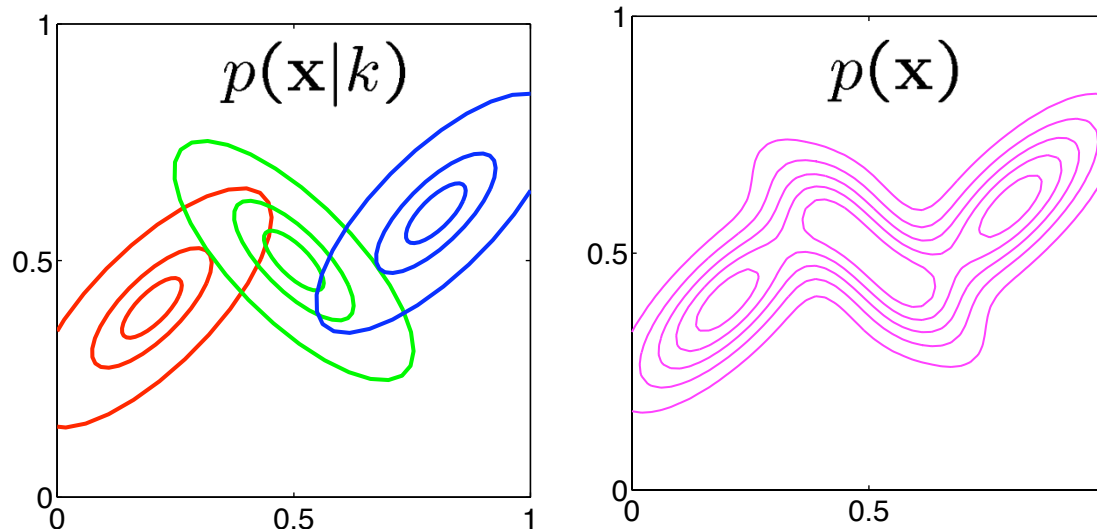
Idea: Use a Mixture of Gaussians!

- Linear combination of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Normalization and positivity requirements

$$\sum_{k=1}^K \pi_k = 1 \quad 0 \leq \pi_k \leq 1$$



Statistical Background Modeling

Mixture of Gaussians in RGB space for background modeling.

Chris Stauffer and Eric Grimson, "Learning Patterns of Activity using Real-time Tracking," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22(8), August 2000, pp. 747-757.



Recent (improved) code available from Zivkovic.

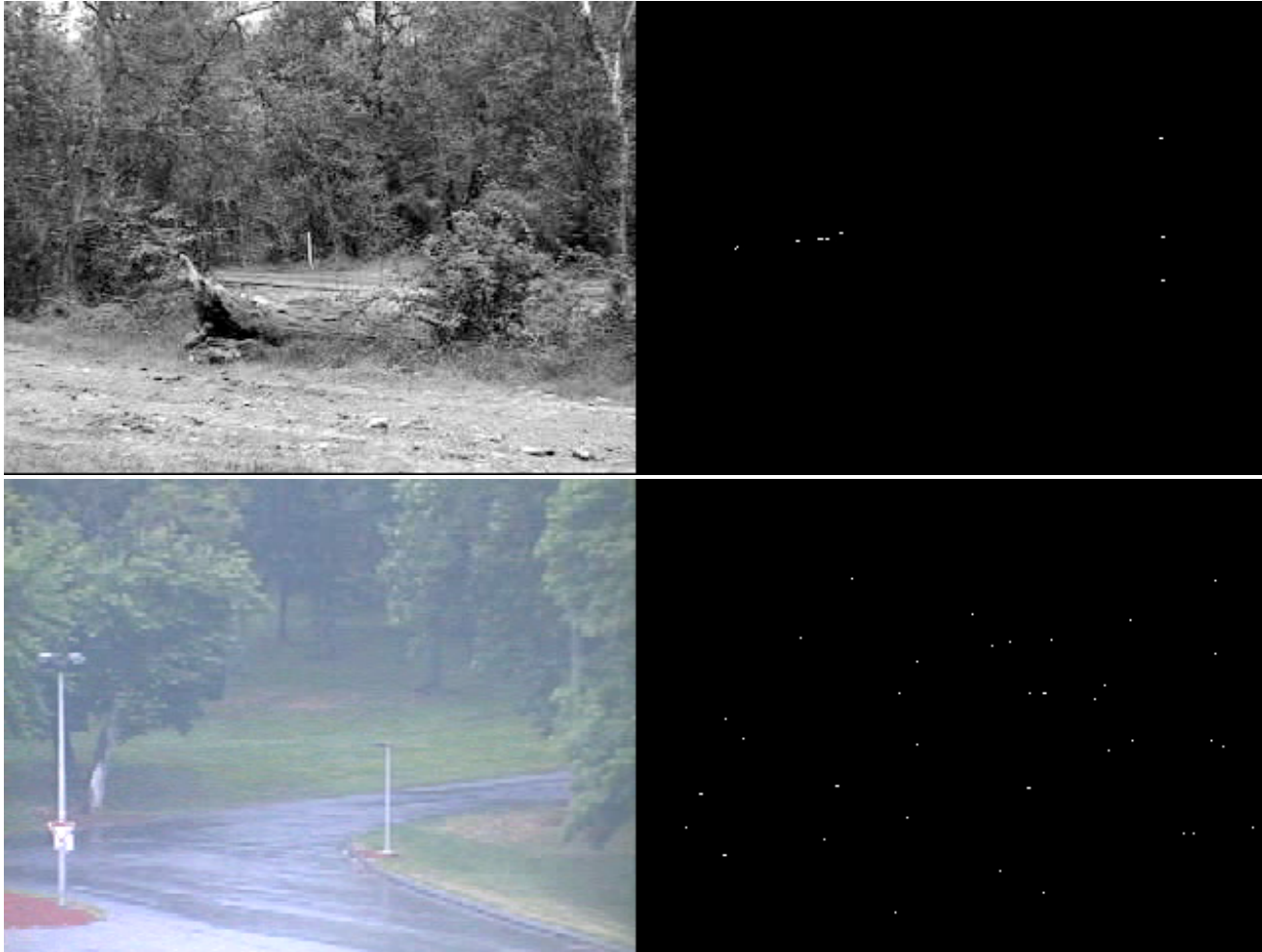
<http://staff.science.uva.nl/~zivkovic/DOWNLOAD.html>

Robert Collins
Penn State

Statistical Background Modeling

Nonparametric statistical model using kernel density estimation (KDE)

Ahmed Elgammal, David Harwood, Larry Davis “Non-parametric Model for Background Subtraction”, 6th European Conference on Computer Vision. Dublin, Ireland, June 2000.



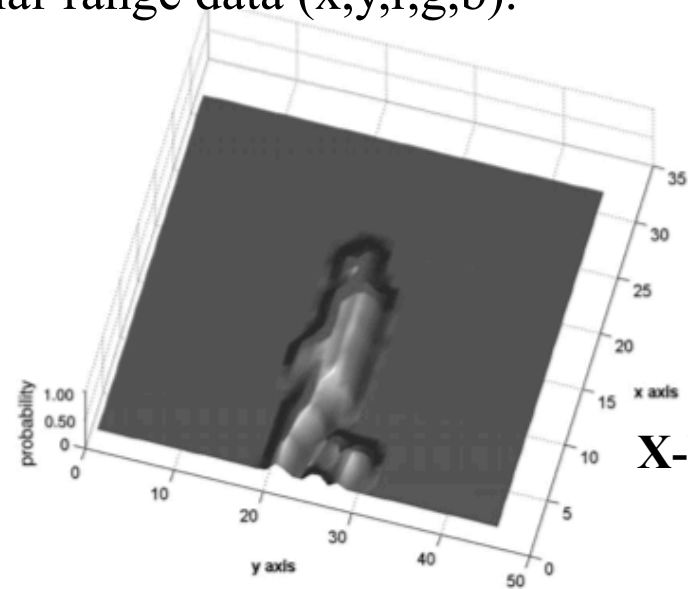
Sheikh and Shah

Model background **AND** foreground using kernel density estimator on 5-dimensional space of joint spatial-range data (x,y,r,g,b).

Example of Foreground Model



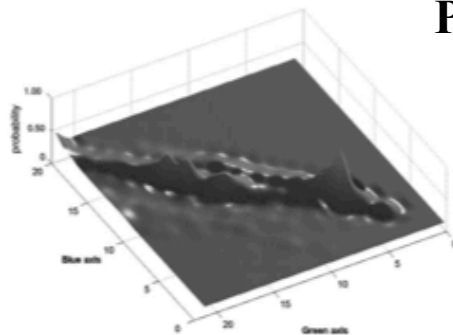
(a)



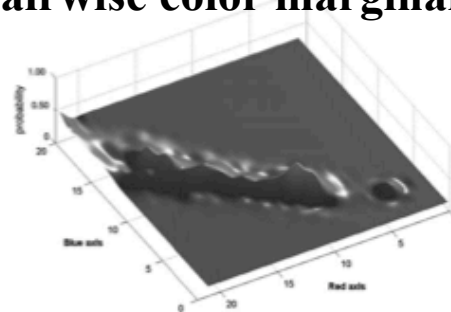
X-Y marginal

(b)

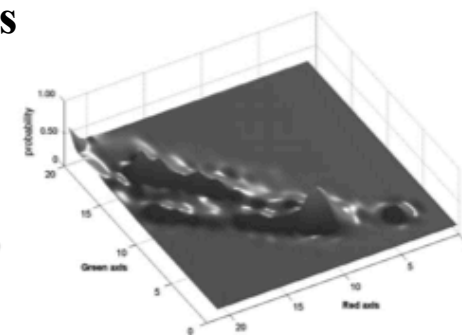
Pairwise color marginals



(c)



(d)



(e)

Sheikh and Shah

- Model background AND foreground appearance

$$P(\mathbf{x}|\psi_b) = n^{-1} \sum_{i=1}^n \varphi_H(\mathbf{x} - \mathbf{y}_i), \quad \text{KDE}$$

$$P(\mathbf{x}|\psi_f) = \alpha\gamma + (1 - \alpha)m^{-1} \sum_{i=1}^m \varphi_H(\mathbf{x} - \mathbf{z}_i) \quad \text{Uniform+KDE}$$

- Classification uses likelihood ratio

$$\tau = -\ln \frac{P(\mathbf{x}|\psi_b)}{P(\mathbf{x}|\psi_f)} = -\ln \frac{n^{-1} \sum_{i=1}^n \varphi_H(\mathbf{x} - \mathbf{y}_i)}{\alpha\gamma + (1 - \alpha)m^{-1} \sum_{i=1}^m \varphi_H(\mathbf{x} - \mathbf{z}_i)}.$$

Thus, the classifier δ is,

$$\delta(\mathbf{x}) = \begin{cases} -1 & \text{if } -\ln \frac{P(\mathbf{x}|\psi_b)}{P(\mathbf{x}|\psi_f)} > \kappa \\ 1 & \text{otherwise,} \end{cases}$$

Other Feature Spaces

Some examples include:

Optic flow. Robert Pless, Spatio-temporal background models for outdoor surveillance. *Journal on Applied Signal. Processing*, 14:2281–2291, 2005

Texture measures (local binary patterns). Heikkilä, M. and Pietikäinen, M. (2006), A Texture-Based Method for Modeling the Background and Detecting Moving Objects. *IEEE Trans. Pattern Analysis and Machine Intelligence* 28(4):657-662.

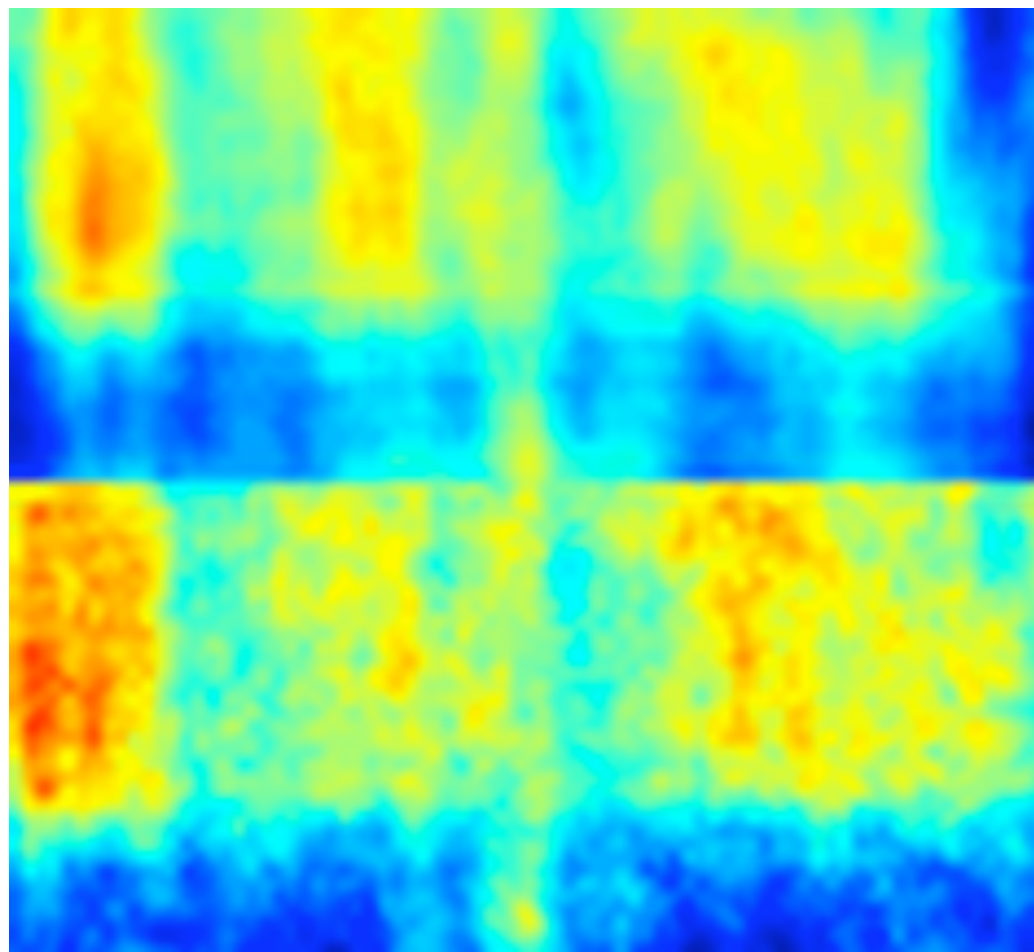
Detector confidence scores. Stalder et.al. “Cascaded Confidence Filtering for Improved Tracking-by-Detection,” to appear, ECCV 2010.

Other Feature Spaces

Example: Detector Confidence Scores



From M Hebert at CMU



Stabilizing Camera Motion



Video in

Reference view

Warped video

Subtraction

Apparent motion of a panning/tilting camera can be removed by warping images into alignment with a collection of background reference views.

Tends not to work well for background subtraction. The background changes while you are not looking at it, causing a false positive detection when you do finally look.

Stabilization works better for frame differencing.

Frank Dellaert and Robert Collins,
“Fast Image-Based Tracking by
Selective Pixel Integration,” ICCV
Workshop on Frame-Rate Vision,
Corfu, Greece, Sept 1999.

Understanding Frame Differencing

Recall the brightness constancy equation for computing optic flow

$$\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

$$\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \quad (\text{spatial derivatives})$$

$$\frac{dx}{dt}, \frac{dy}{dt} = (\mathbf{u}, \mathbf{v}) \quad (\text{optical flow; motion vector})$$

$$\frac{\partial I}{\partial t} \quad (\text{temporal derivative} = \text{frame difference!})$$

Understanding Frame Differencing

Recall the brightness constancy equation for computing optic flow

$$\frac{\partial I}{\partial x} \frac{dx}{dt} + \frac{\partial I}{\partial y} \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$$

Observations:

If there is no optical flow (motionless pixels), then frame difference should be zero (or very small due to random noise).

Conversely, if frame difference is large enough magnitude, then that implies motion at that pixel. [changing brightness breaks this implication]

If no spatial gradient at a pixel (uniform region) then frame difference will be zero even if there IS motion... so motion state is undefined in those areas.

If brightness constancy does not hold, frame differencing can fail.

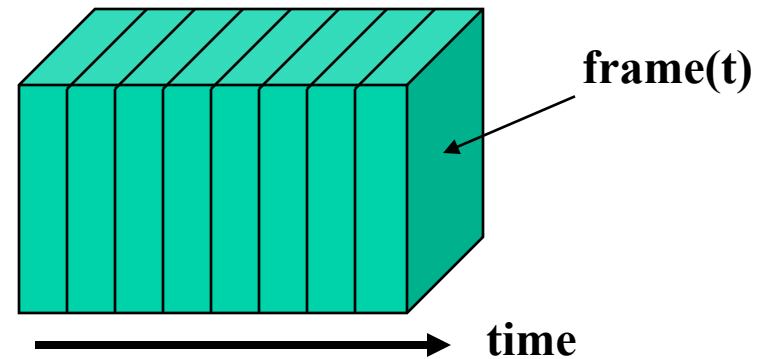
Our Work in this Area

Basic Idea: Generalize persistent frame differencing to include both spatial and temporal smoothing, and to use temporal information from frames both forwards and backwards in time from the current frame.

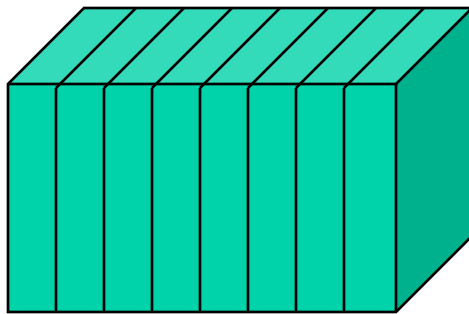
- **Z.Yin and R.Collins, “Belief Propagation in a 3D Spatio-temporal MRF for Moving Object Detection, IEEE Computer Vision and Pattern Recognition (CVPR), 2007.**
- **Z.Yin and R.Collins, “Moving Object Localization in Thermal Imagery by Forward-Backwards MHI, IEEE Workshop on Object Tracking and Classification in and Beyond the Visible Spectrum (OTCBVS), 2006.**
- **Z.Yin and R.Collins, paper in preparation on Spatially Tuned Message Passing.**

MRF-based Motion Detection

consider a spatio-temporal
sequence of video frames

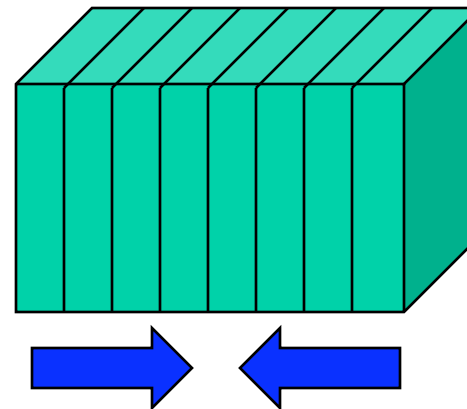


Filtering



compute motion based
on previous frames
(recursive)

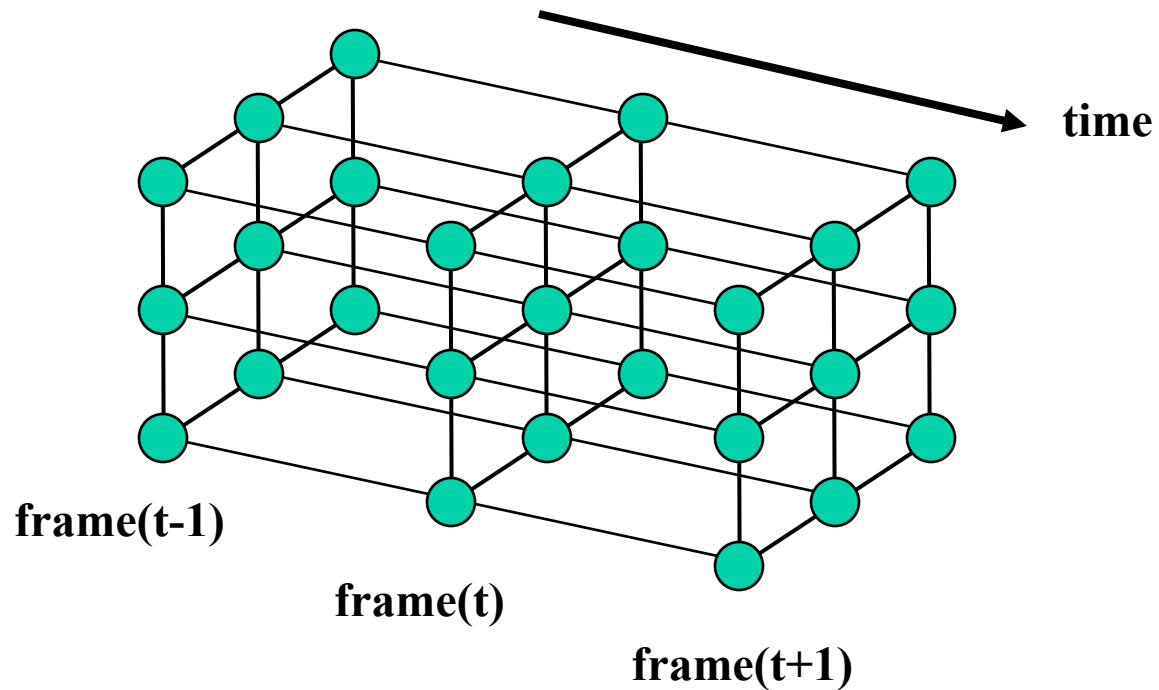
Smoothing



compute motion based on
previous and future frames
(sliding window – fixed lag)

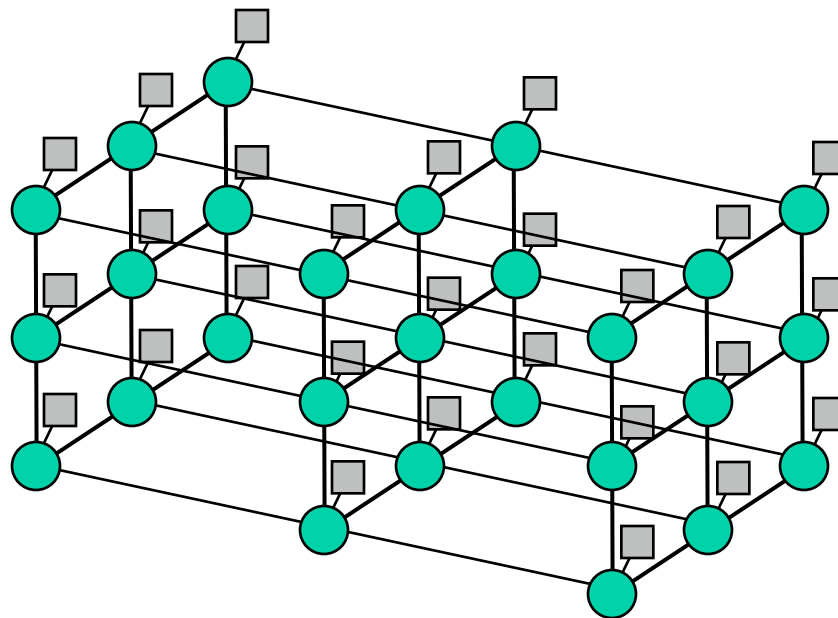
MRF-based Motion Detection

Consider grid where each pixel has is 6-connected
(4 spatial neighbors, 2 temporal neighbors)



MRF-based Motion Detection

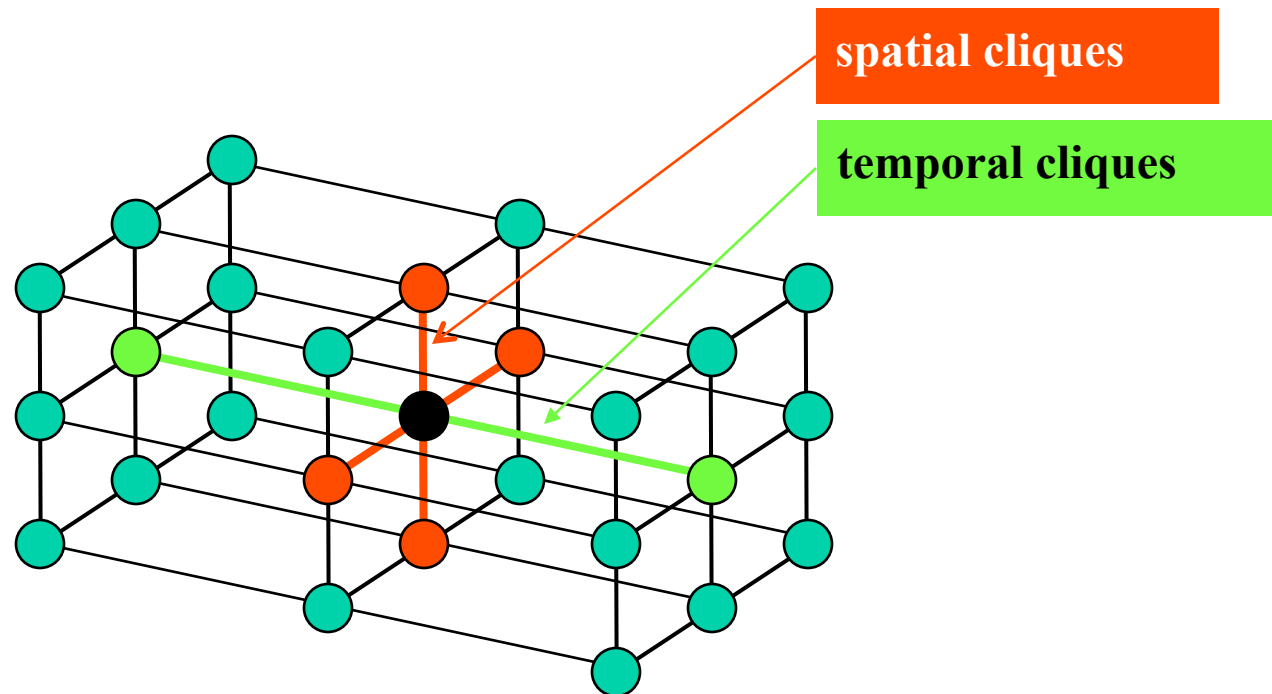
These nodes will be our “hidden” states representing motion / no-motion.



Each hidden state is also connected to an observed state (in our case a pixel difference)

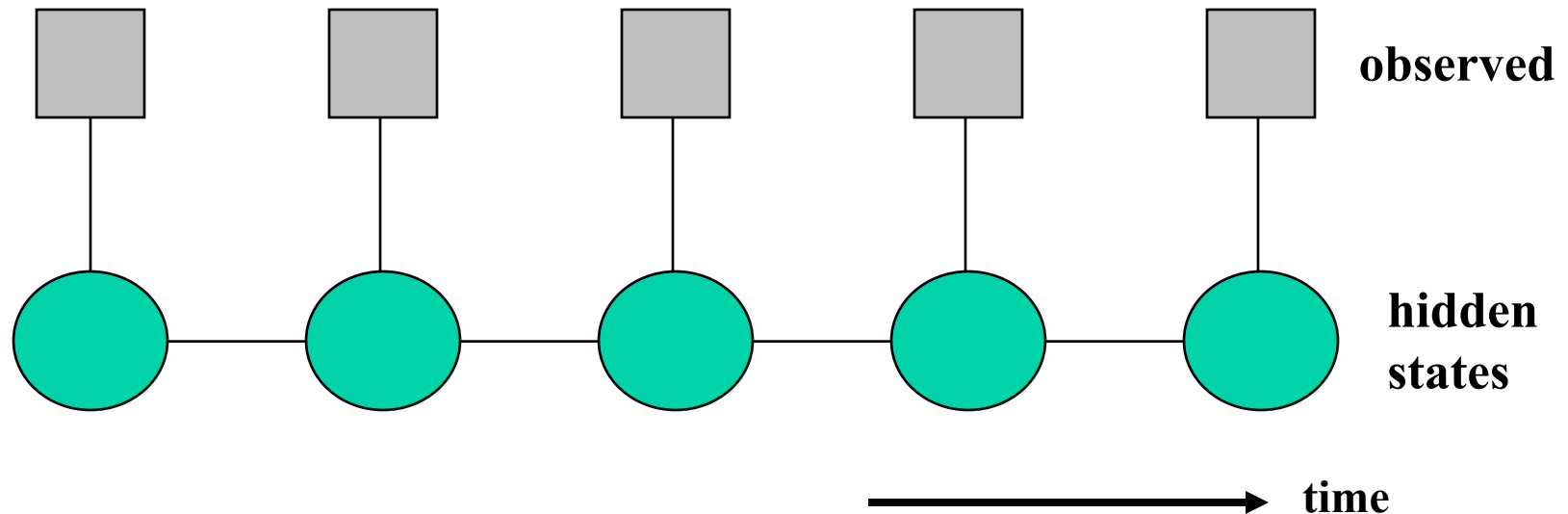
MRF-based Motion Detection

This setup has only pairwise cliques.



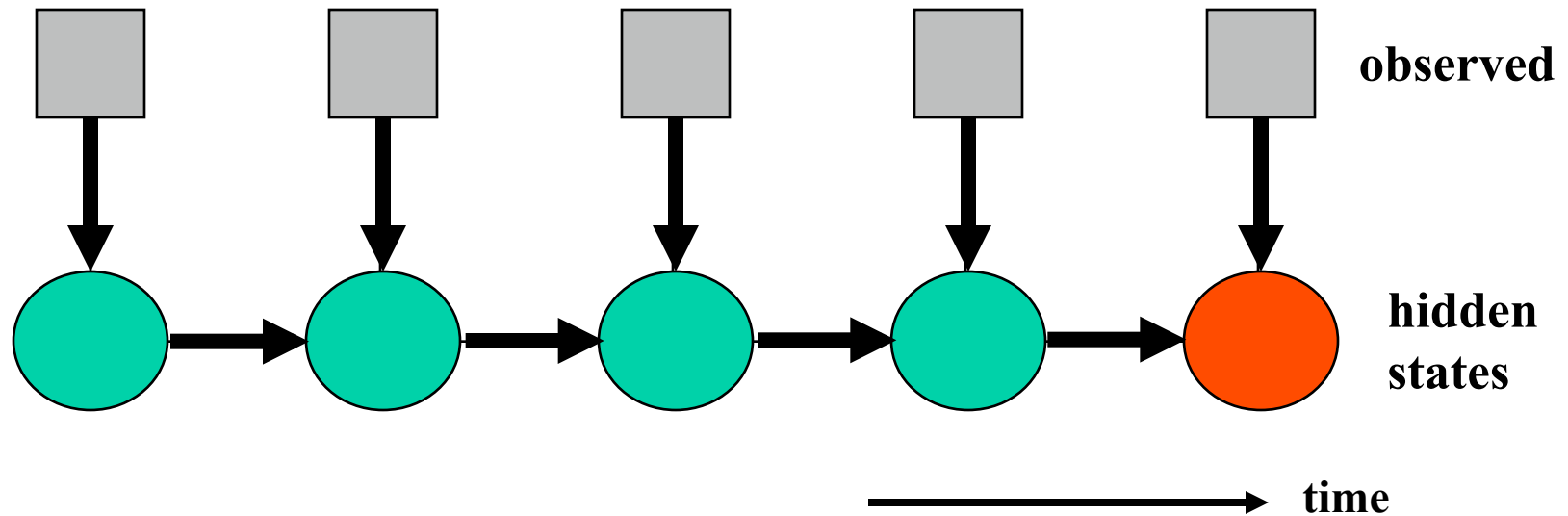
We define a simple compatibility function on each clique via the “Potts model” (each pixel encouraged to have same state value as its neighbors).

MRF-based Motion Detection



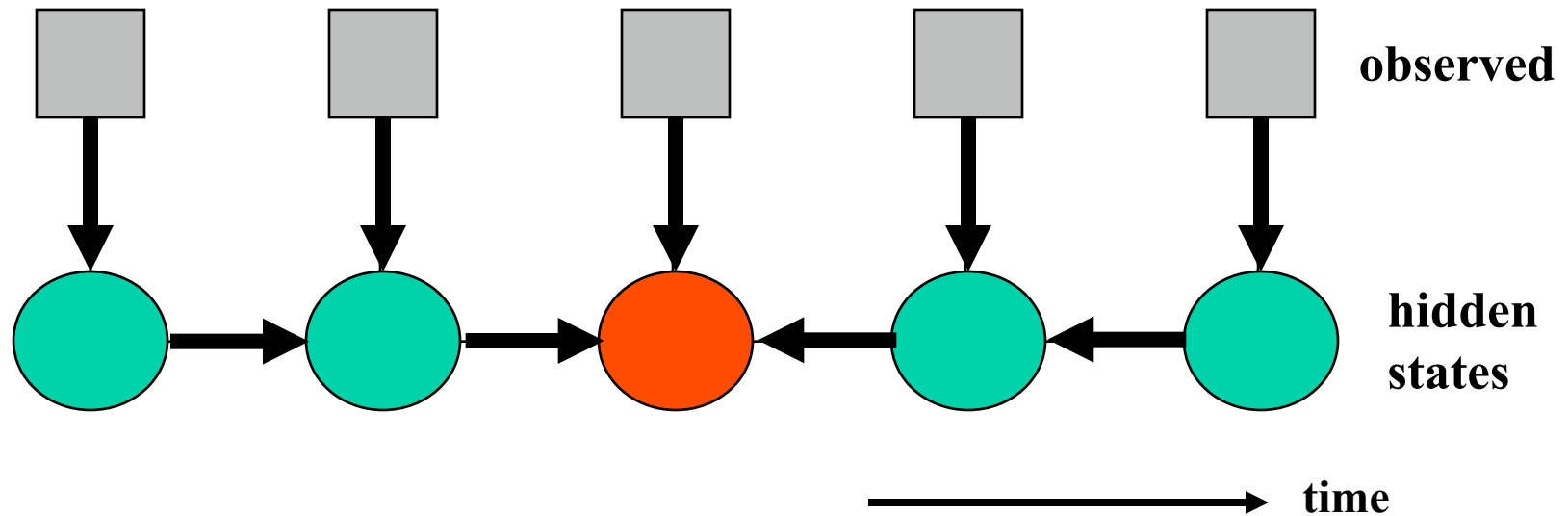
Consider temporal cliques, processing one pixel through time.
Note, this graphical model looks like a HMM. To compute optimal state value (or distribution over state values) at any node, we can do message passing.

MRF-based Motion Detection



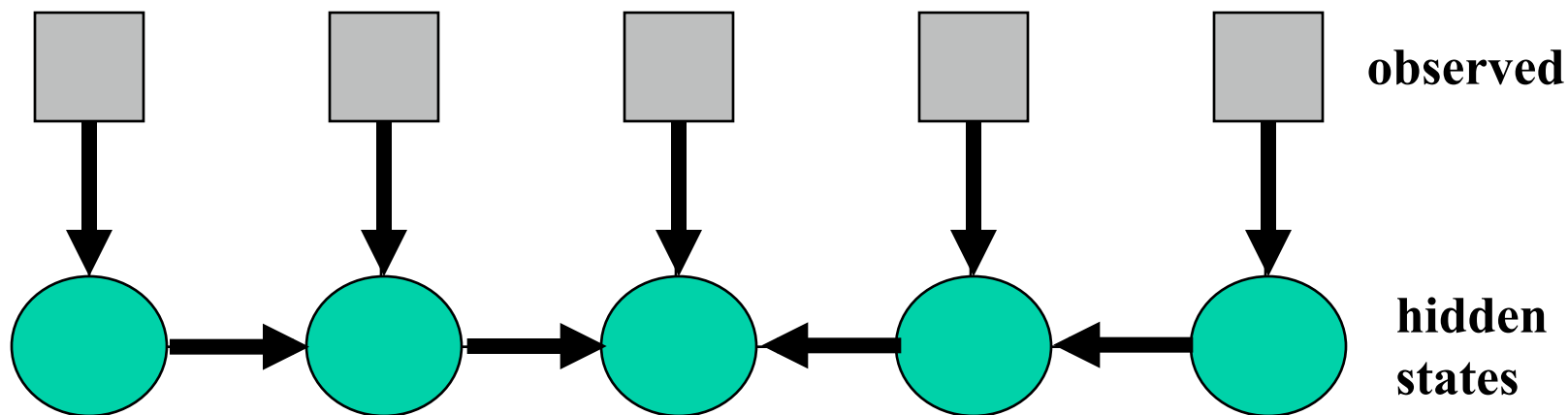
Message passing for filtering (Forward only)

MRF-based Motion Detection



**Message passing for smoothing (fixed-lag smoothing).
Forward-Backward message passing.**

MRF-based Motion Detection



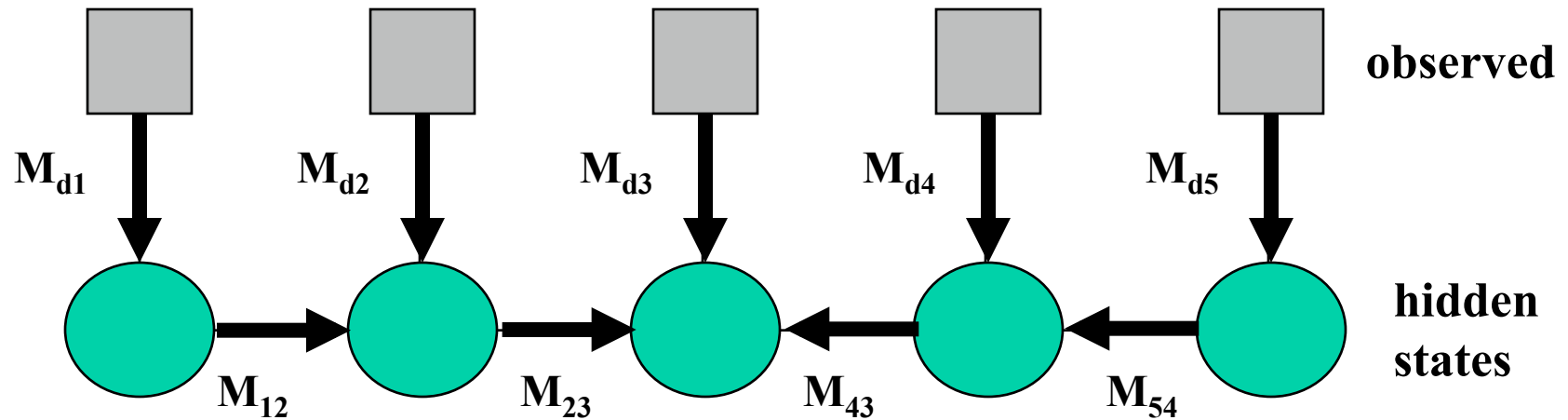
Specific instantiation.

Binary state: (no-motion , motion)

Each hidden node contains distribution $v = (a , 1-a)$.

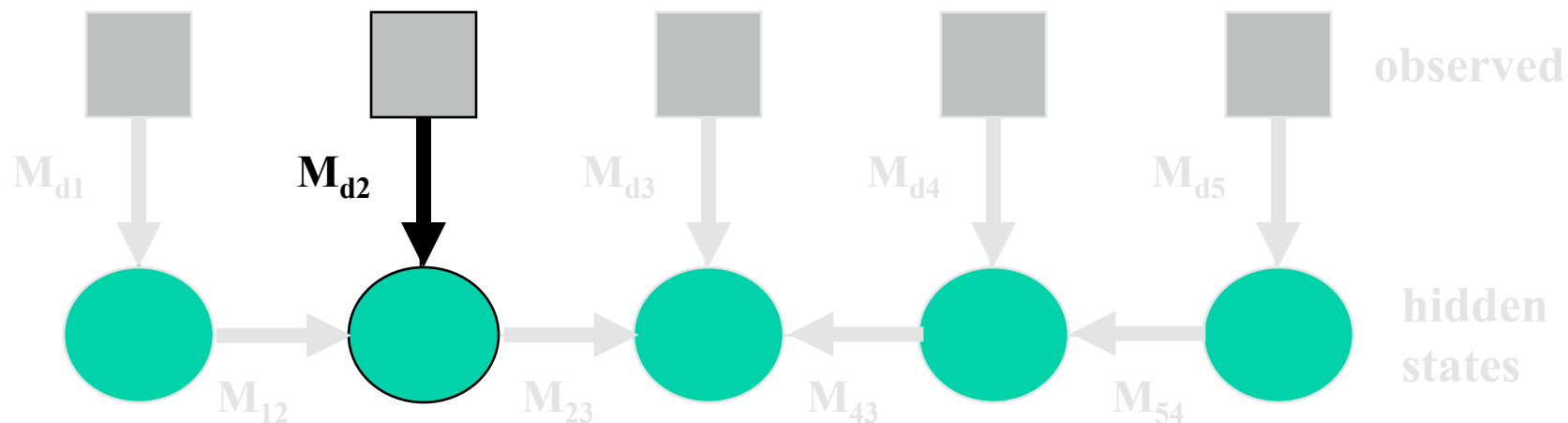
Compatibility represented by Potts model $P = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$

MRF-based Motion Detection



Messages specify what distribution each node thinks its neighbor should have.

MRF-based Motion Detection



Data messages:

If temporal difference magnitude at pixel is over threshold:

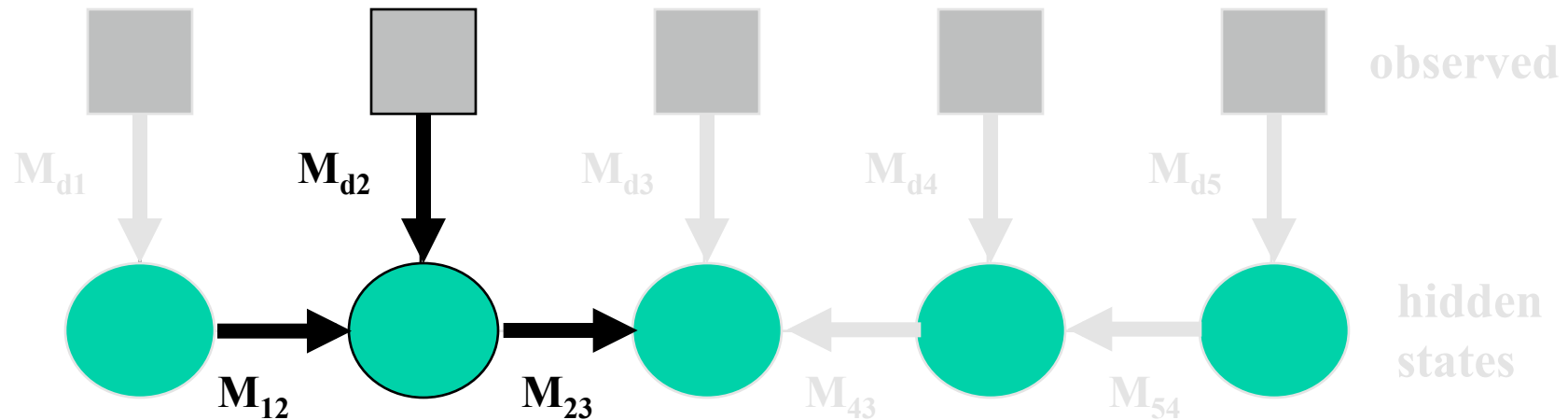
[0 , 1] confident of motion

otherwise

[.5 , .5] equally uncertain about motion / no motion

(alternatively could choose to use [1, 0])

MRF-based Motion Detection



Internal message passing:

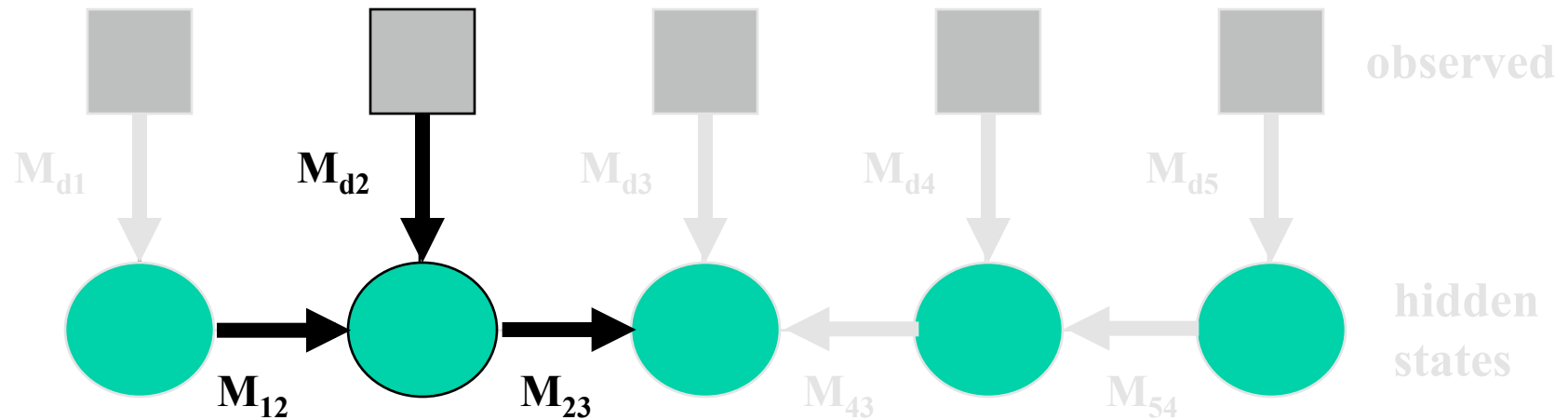
compute “belief” at node from incoming messages

$$\text{bvector} = (M_{d2} .* M_{12}) / \text{dotprod}(M_{d2}, M_{12})$$

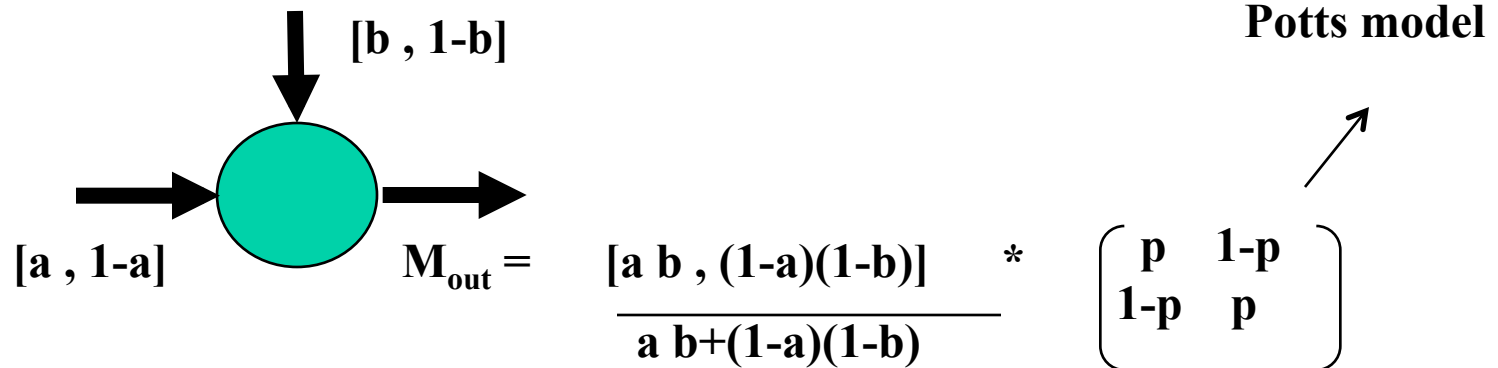
marginalize pairwise compatibility wrt belief

$$M_{12} = \text{bvector} * P$$

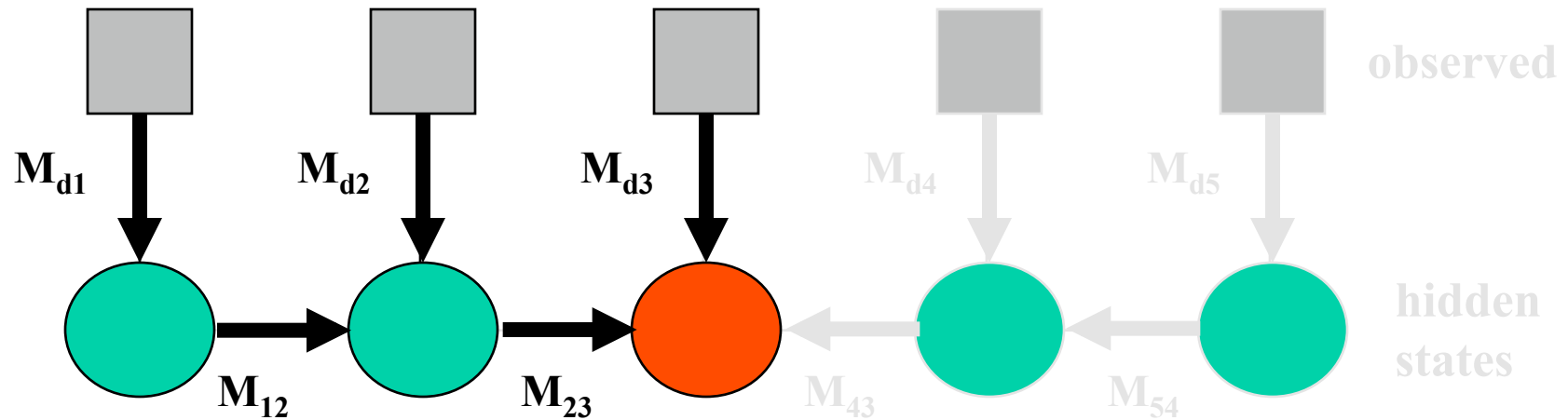
MRF-based Motion Detection



Example:

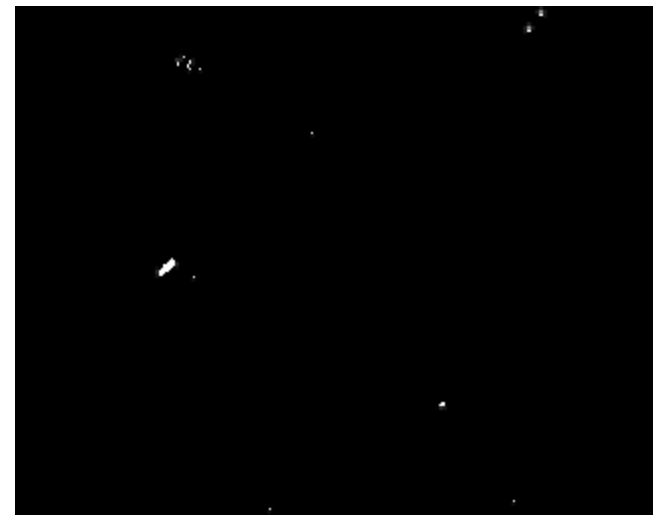


MRF-based Motion Detection

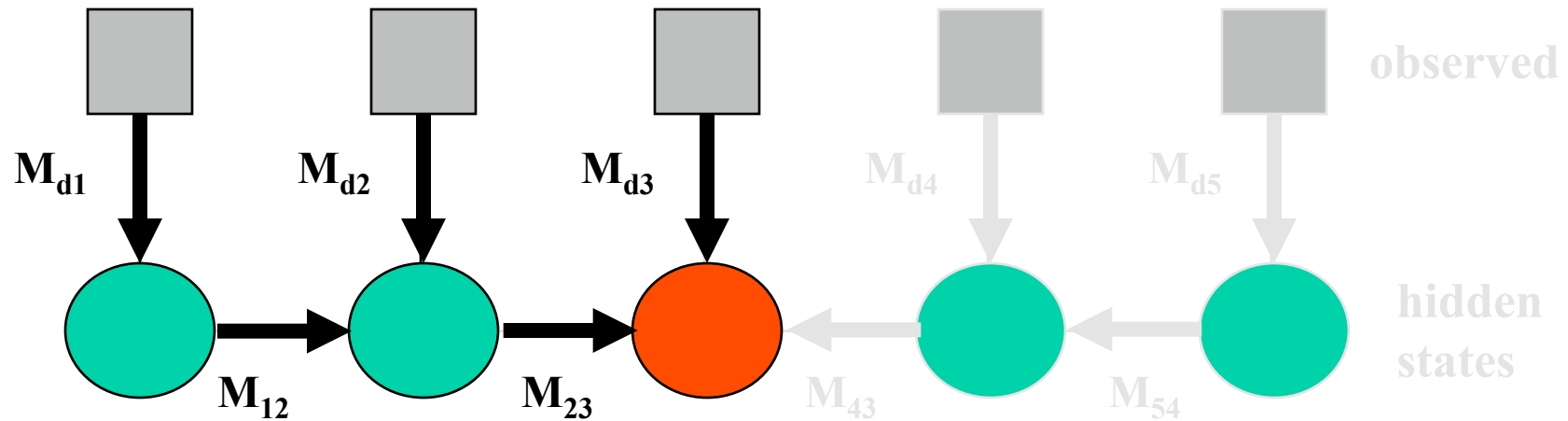


Forward only temporal filtering

**Showing MMSE at each pixel.
This is expected value of the
belief at each pixel.**

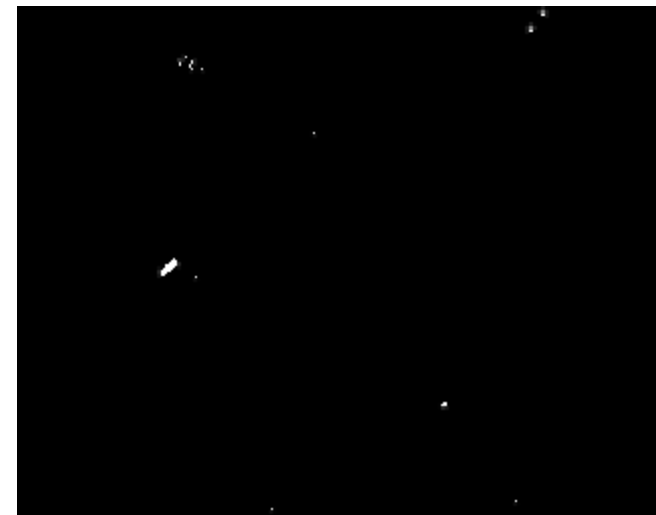


MRF-based Motion Detection

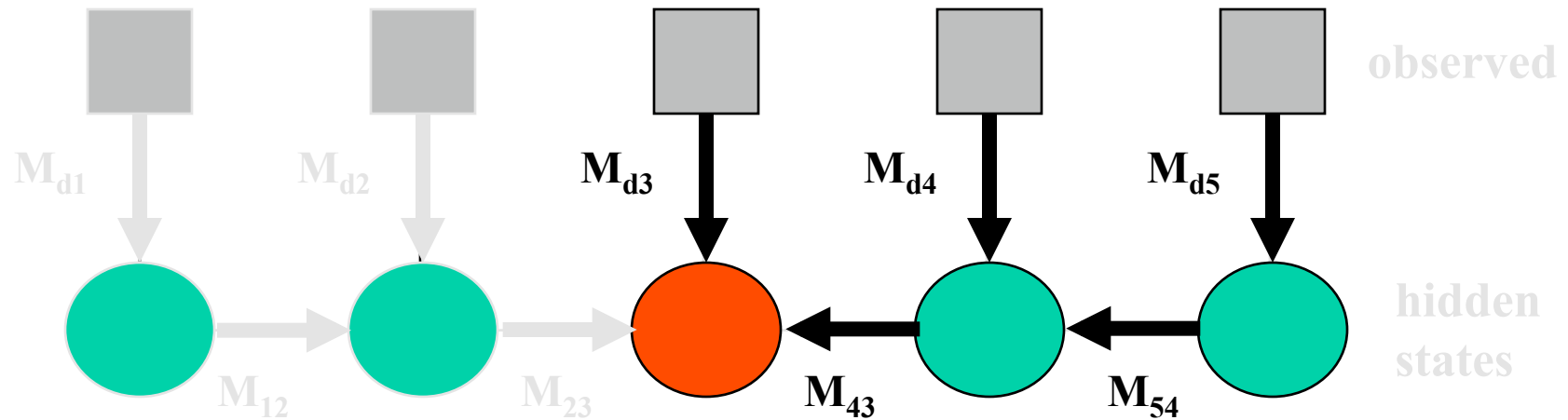


Forward only temporal filtering

Note: this looks a *lot* like motion history image (MHI) processing. In fact, you can show it *is* an MHI, with an exponential decay function.



MRF-based Motion Detection

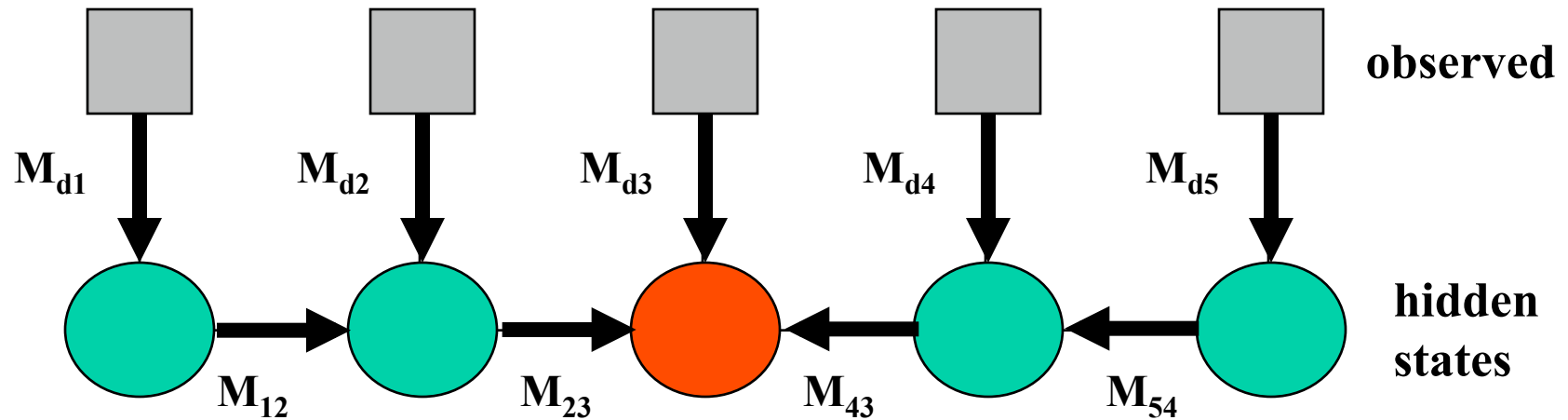


Backward only temporal filtering

MHI in the opposite direction.

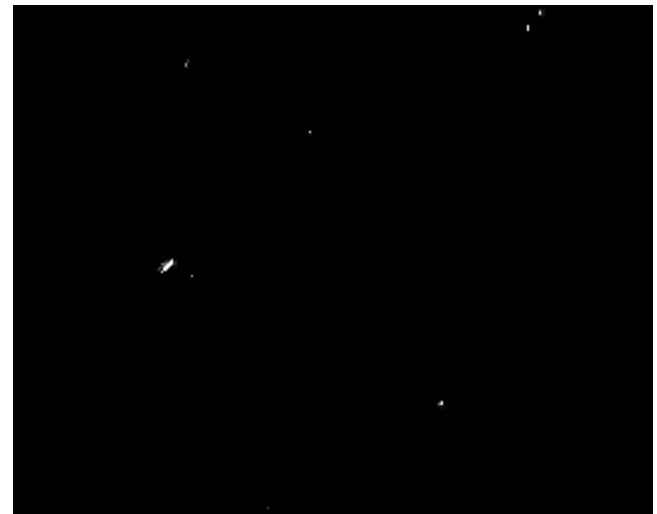


MRF-based Motion Detection



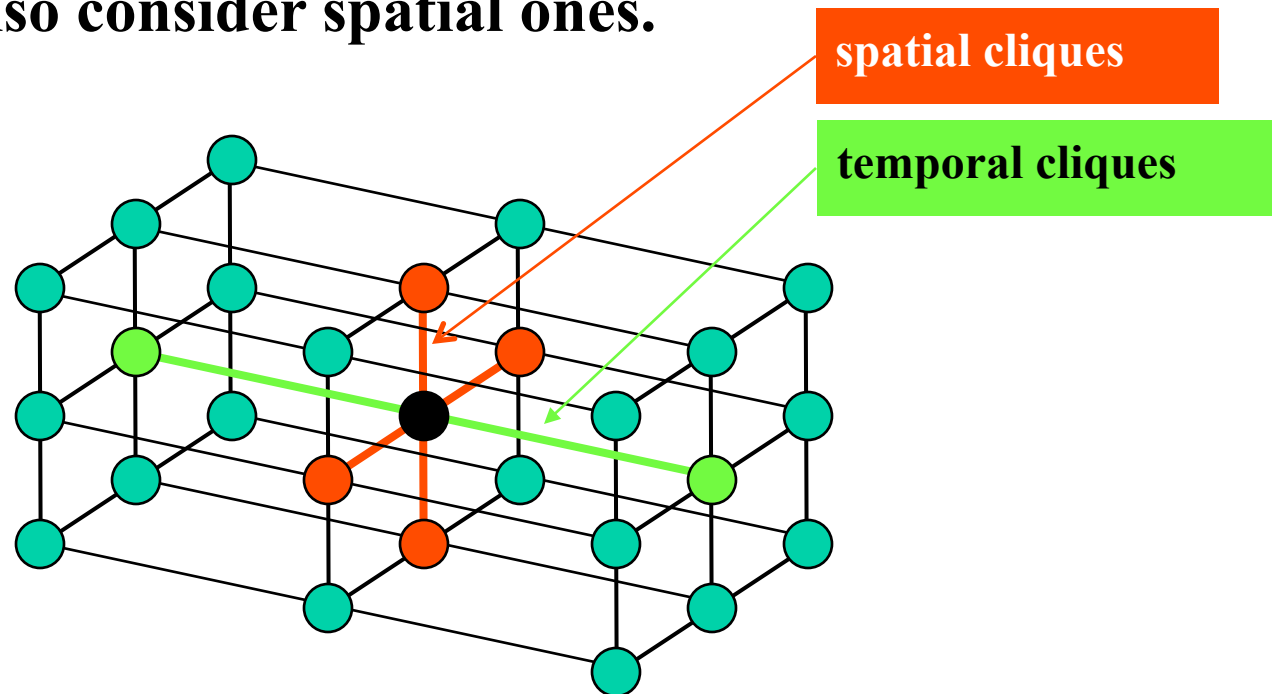
Forward/Backward

**Better delineation. No “trails”.
Reduced noise.**



MRF-based Motion Detection

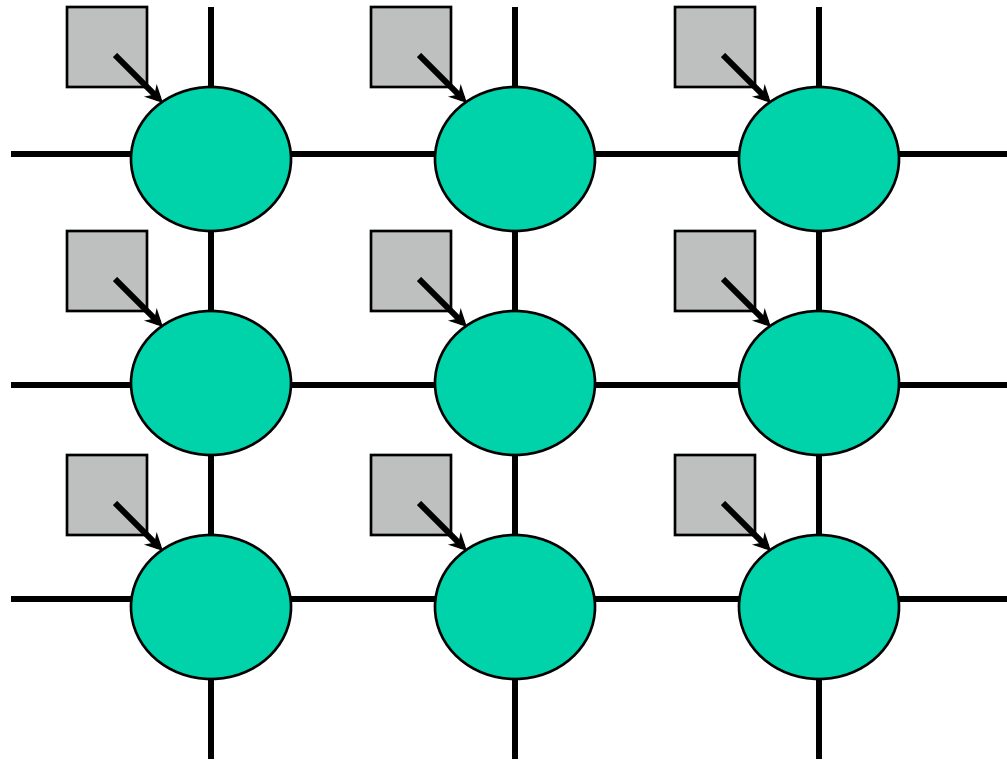
So far we only looked at temporal cliques.
Must also consider spatial ones.



We use the same Potts model compatibility function
(each pixel encouraged to be in same state as neighbors).

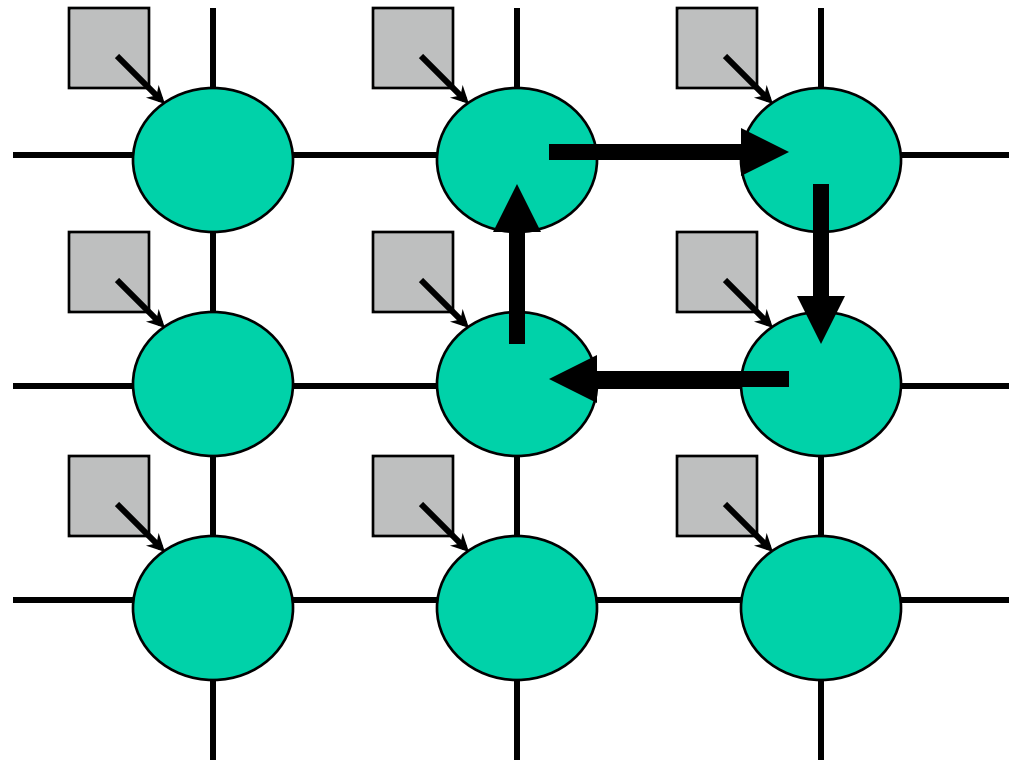
MRF-based Motion Detection

Consider message passing on the spatial grid...



MRF-based Motion Detection

Consider message passing on the spatial grid...



**Problem, there are loops (cycles) in the graph.
Message passing is not strictly correct when used
on loopy graphs (may not even converge).**

MRF-based Motion Detection

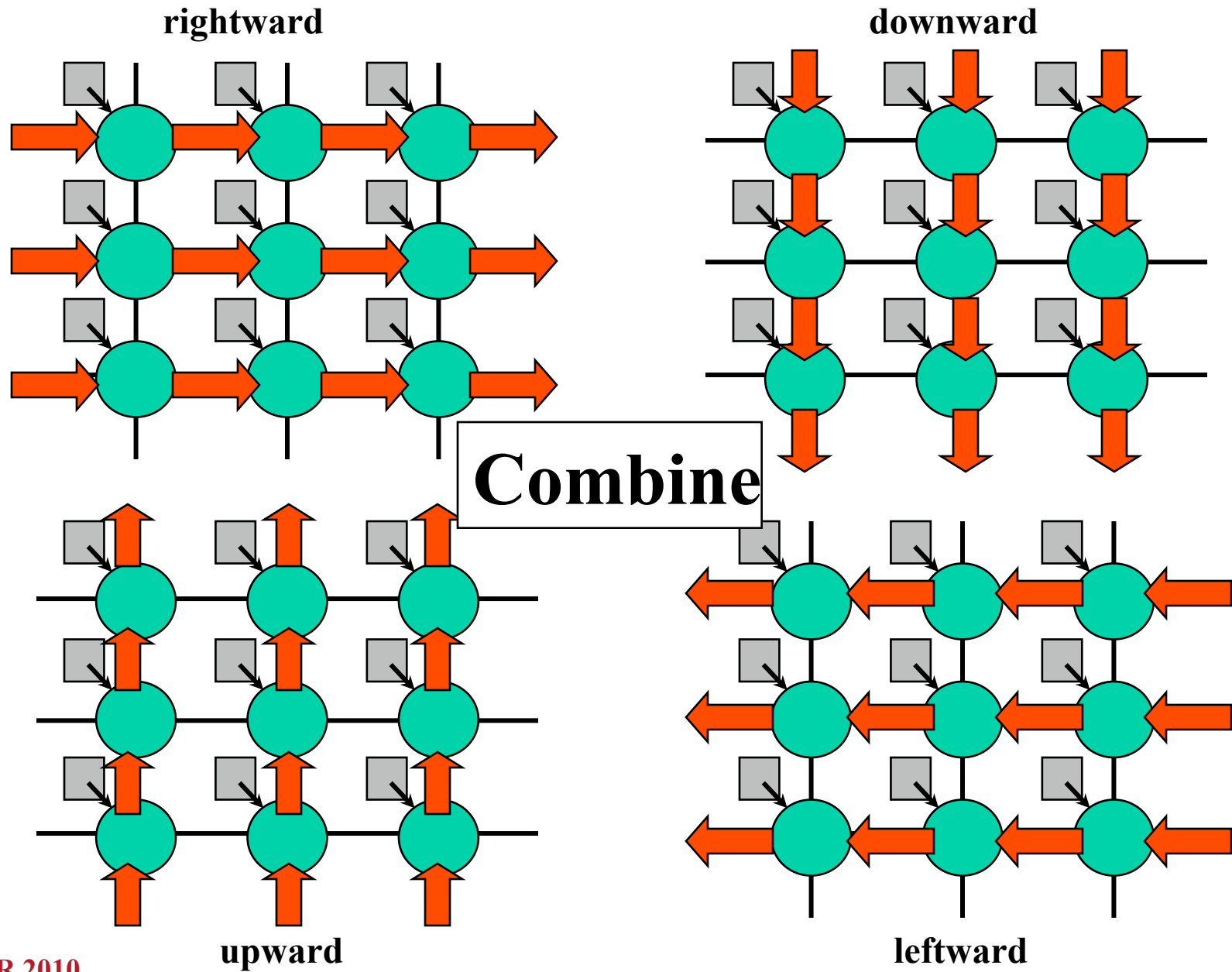
Solution:

**Go ahead and use message passing anyways!
(also called loopy belief propagation)**

It often works very well in practice.

**There is a lot of work going on currently to
characterize when and why this works.**

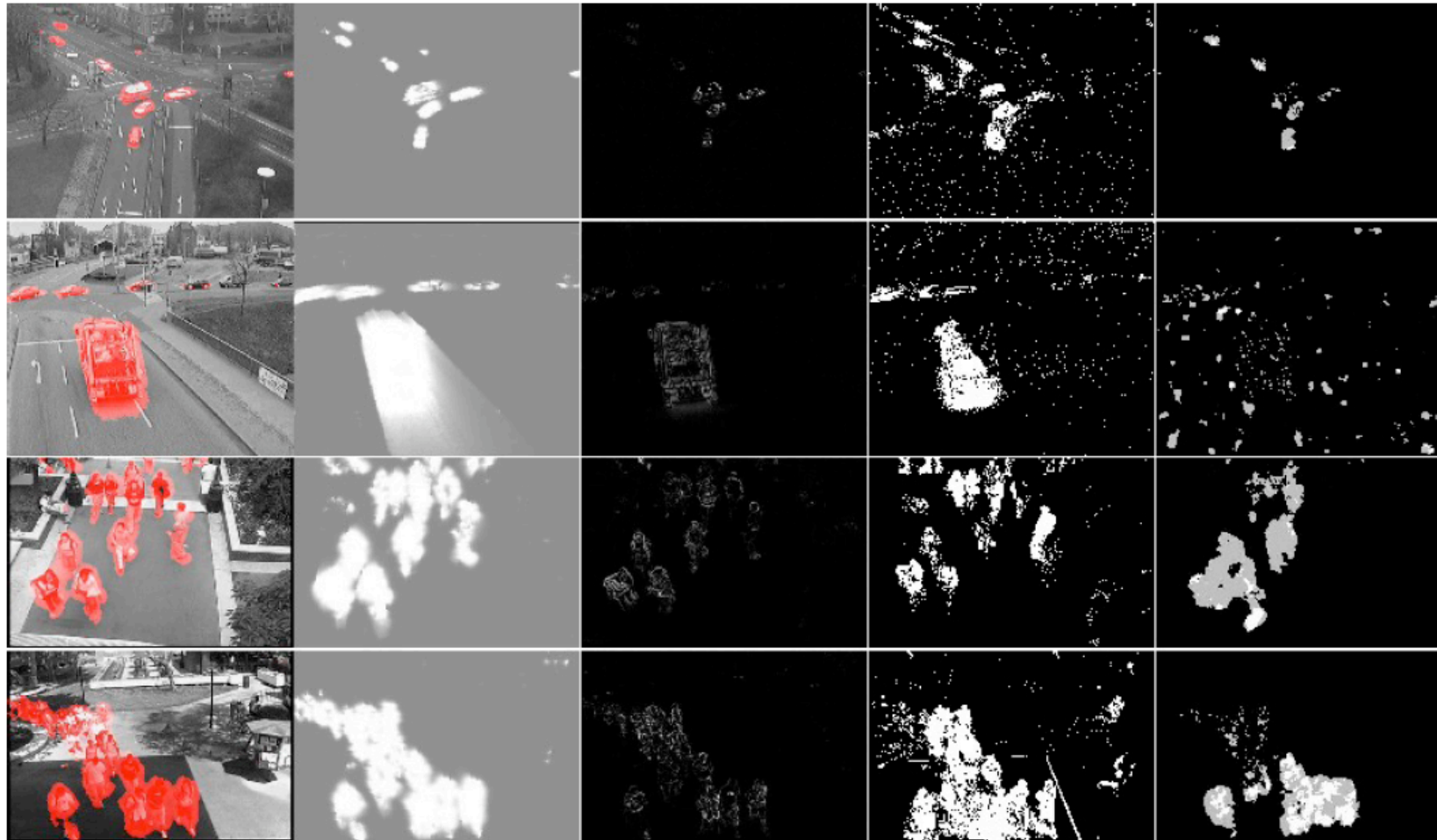
Accelerated Message Passing



Current State of the Art

Yin and Collins: Add directional message passing based on considering the aperture problem (what components of “flow” are observable).

Qualitative evaluation



Red masks
from ours

Ours

Frame
differencing

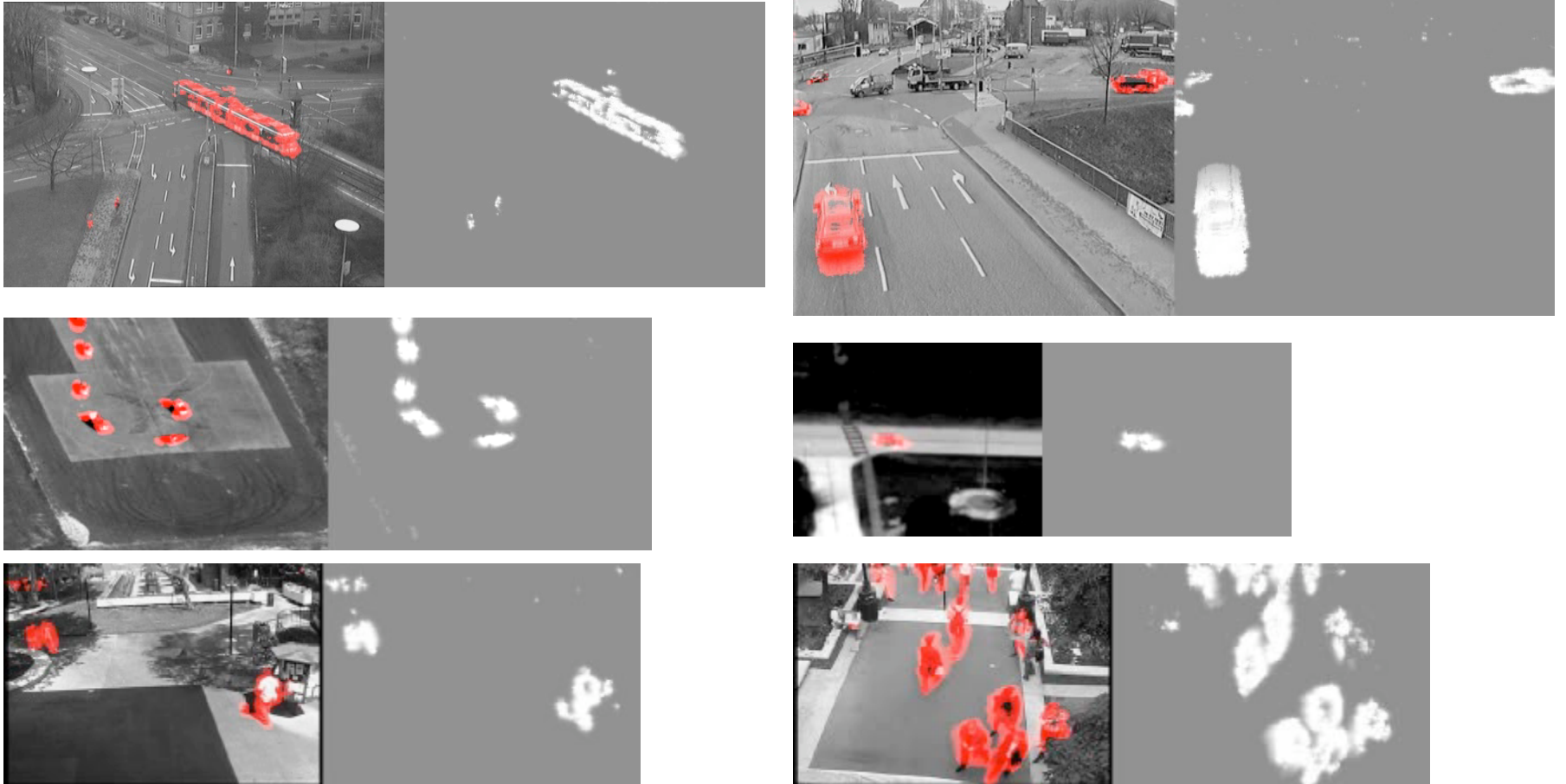
OpenCV
adaptive bg

Dense
optical flow

Current State of the Art

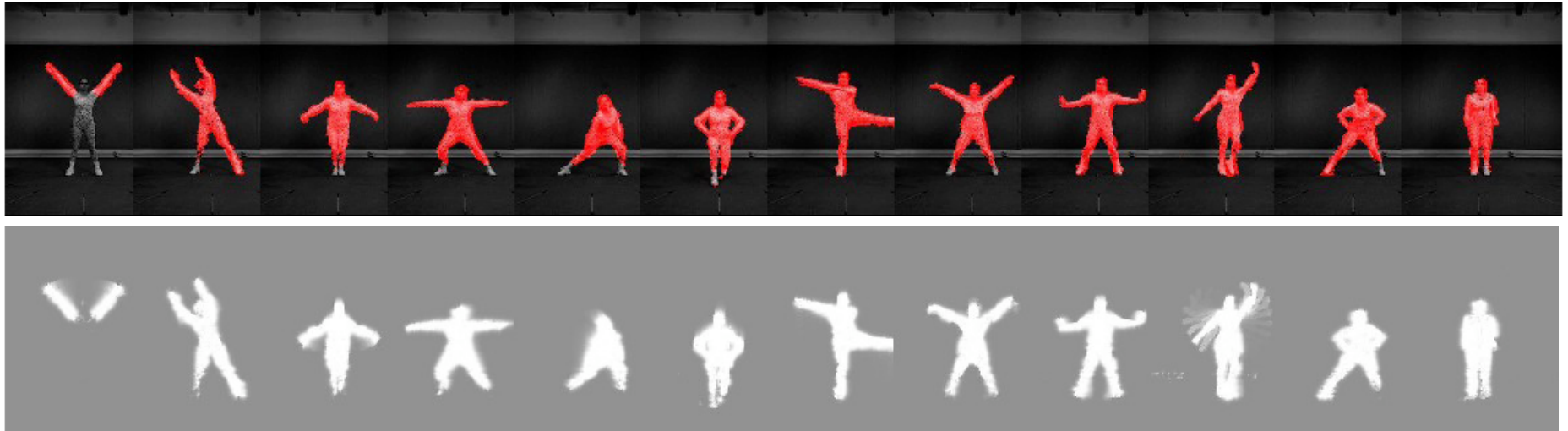
Yin and Collins Results

Video demos

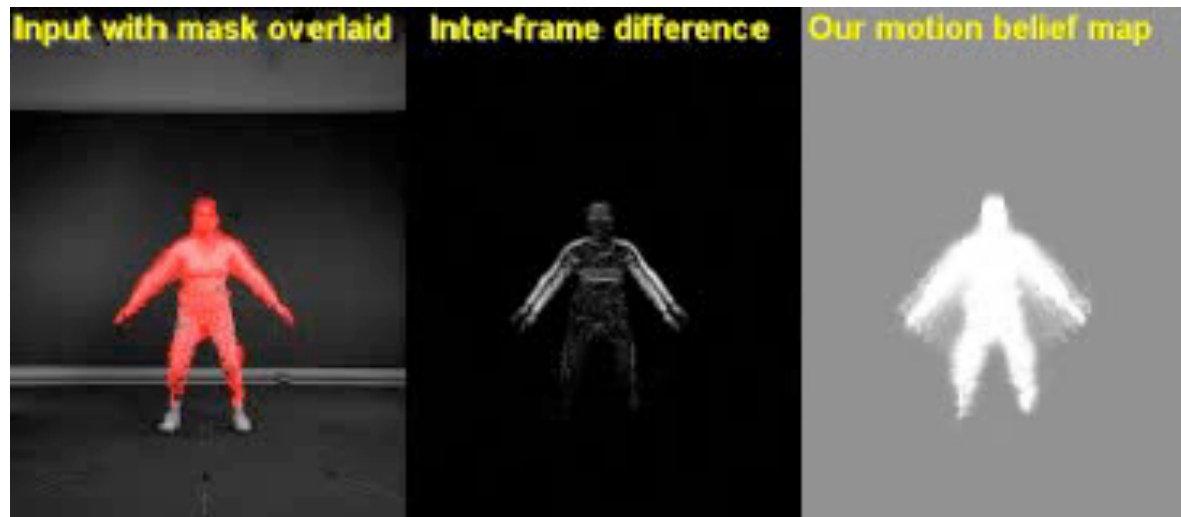


Current State of the Art

1. Motion segmentation



Video demo

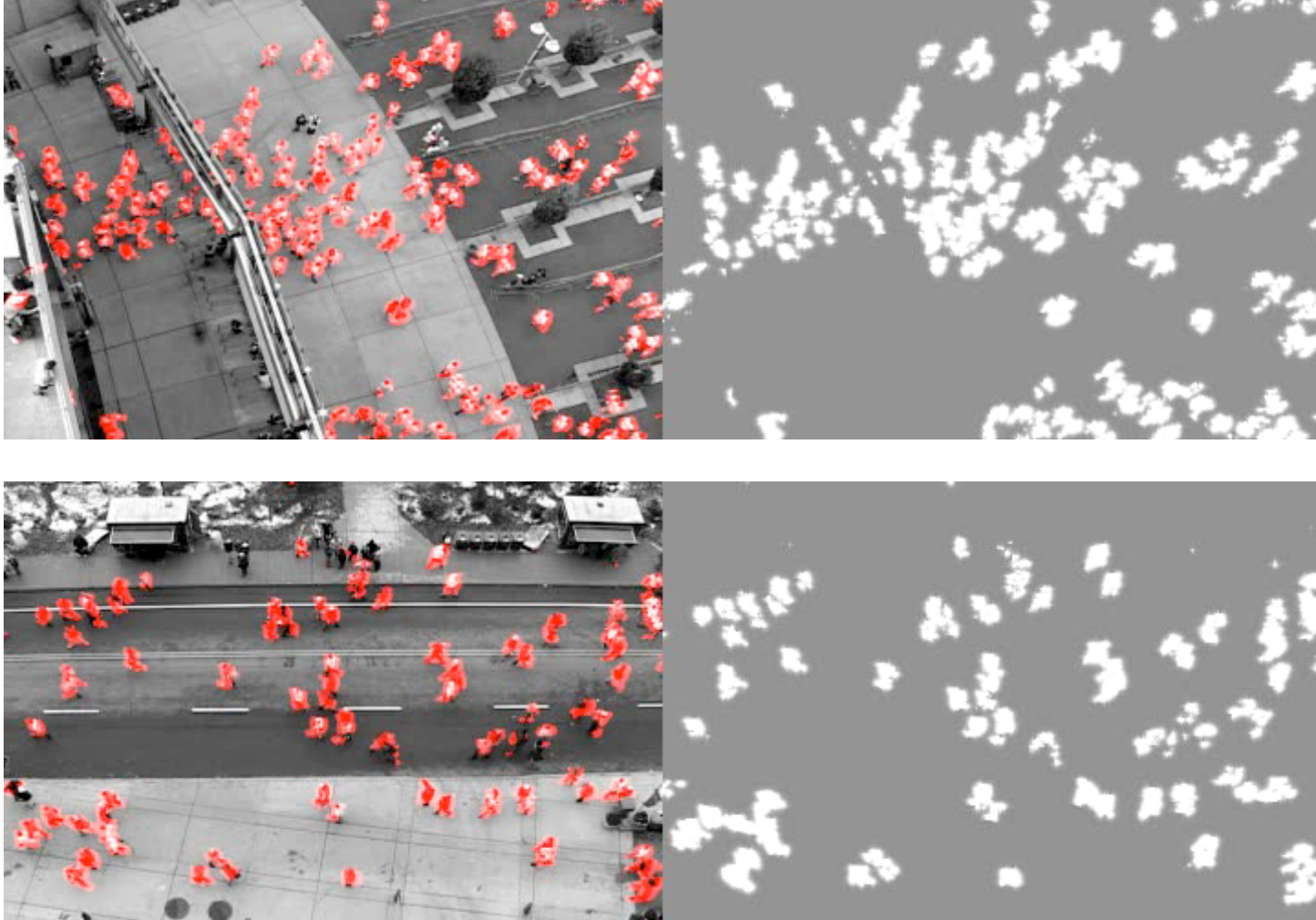


Robert Collins
Penn State

Current State of the Art

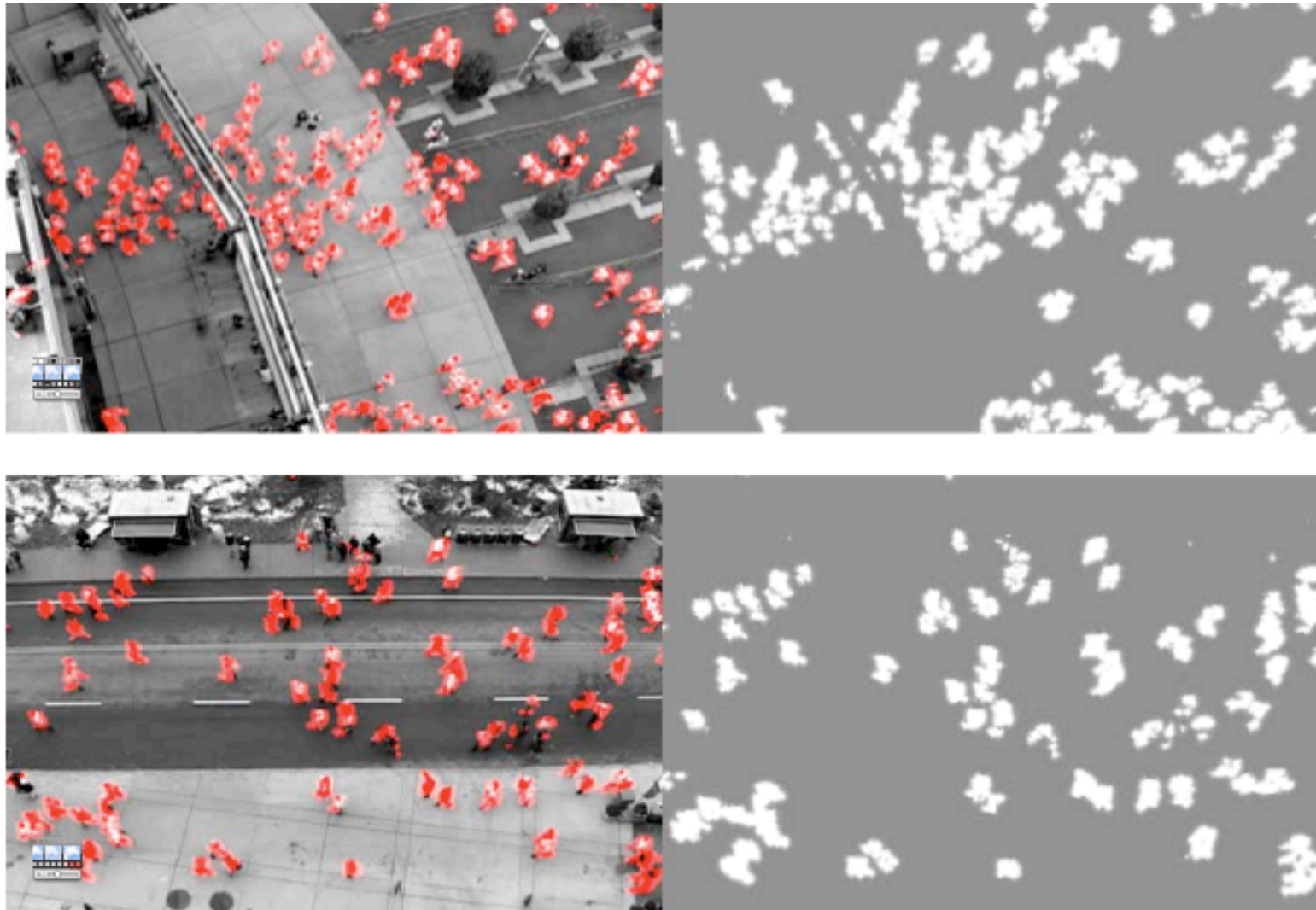
Yin and Collins results

Video demos



Next Step

Group foreground pixels into “blobs” that we can count or track.



Overview

Part 1: Change/Motion Detection

Basics: BG subtraction; Frame Difference

Classification-based methods

Part 2: From Pixels to 2D Blobs

Detection via RJMCMC

~~Classifier Grids~~

Part 3: Data Association

Linear Assignment Problem

Murty K-best; PDAF; JPDAF

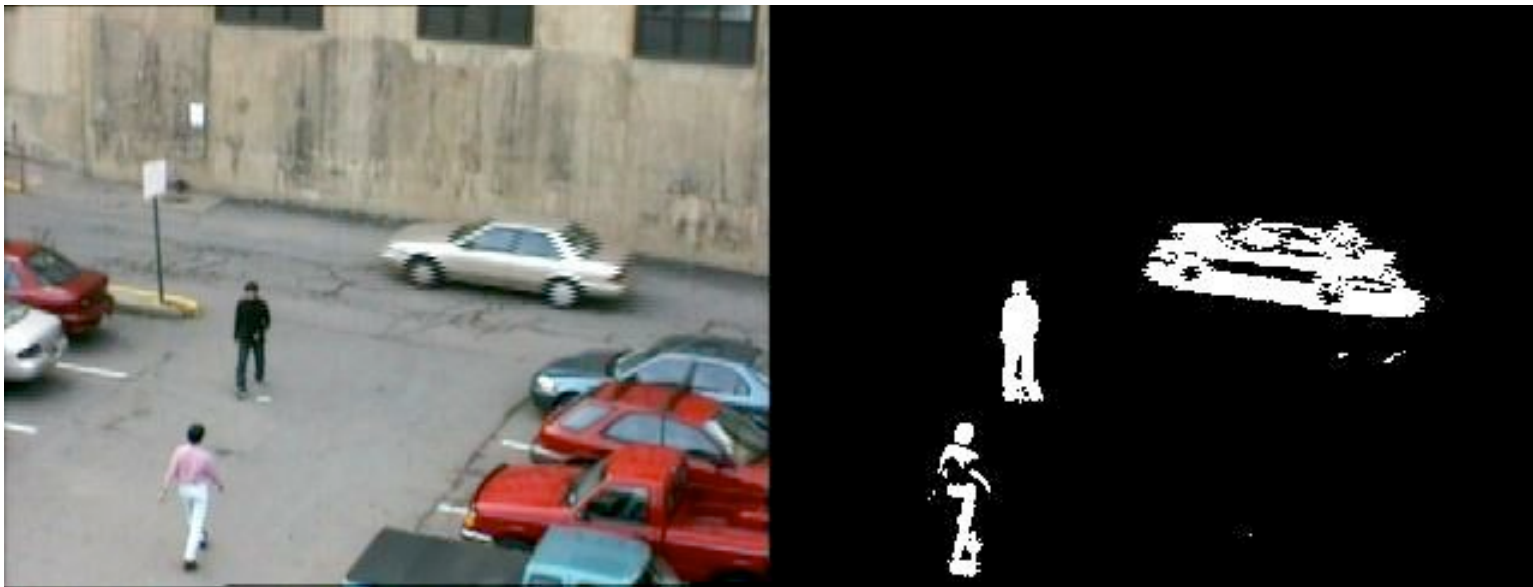
Part 4: Persistent Tracking

Adaptive Tracking

Tracking as Classification

The Story so Far...

We can classify foreground pixels based on background subtraction or frame differencing.

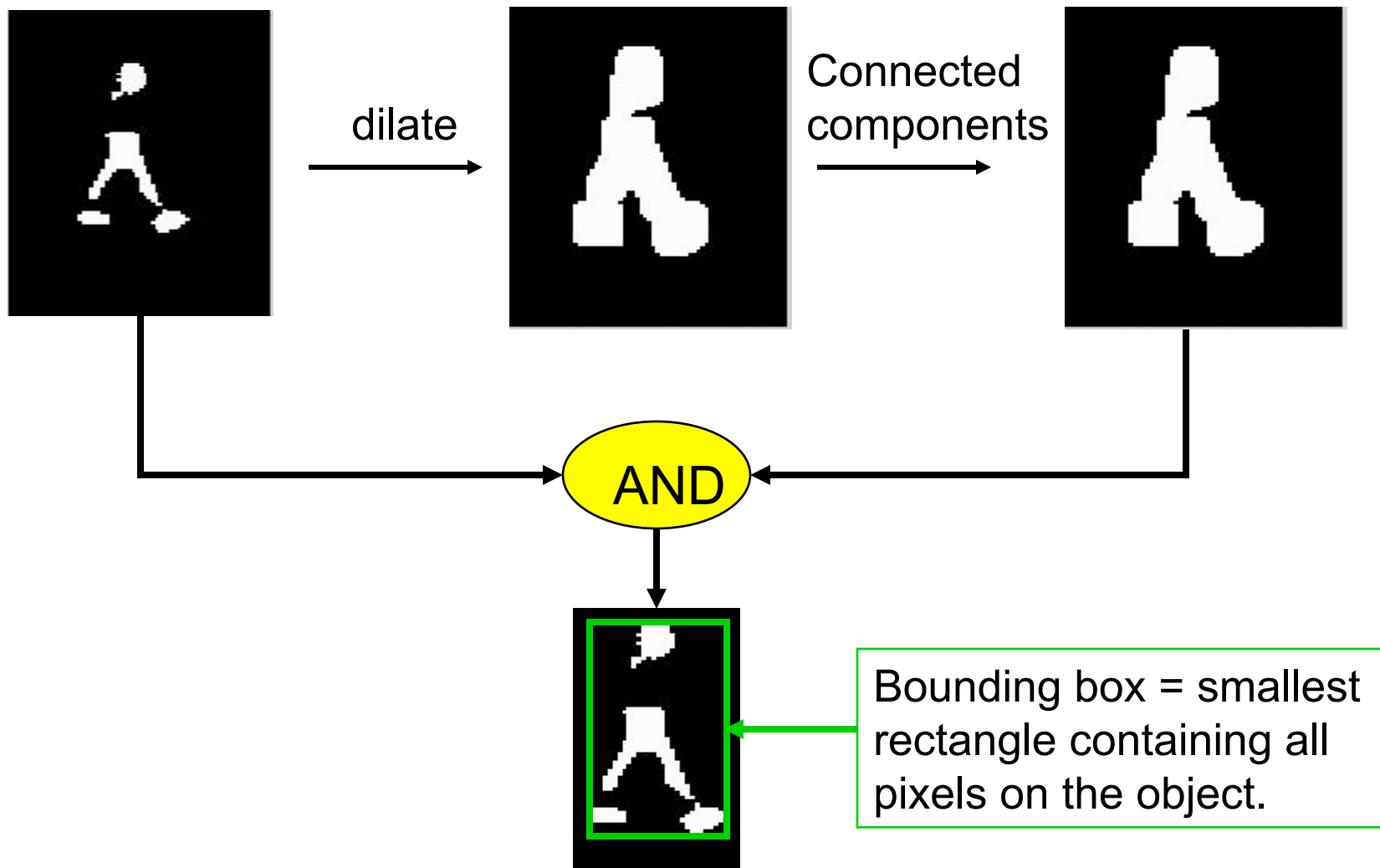


Next Task

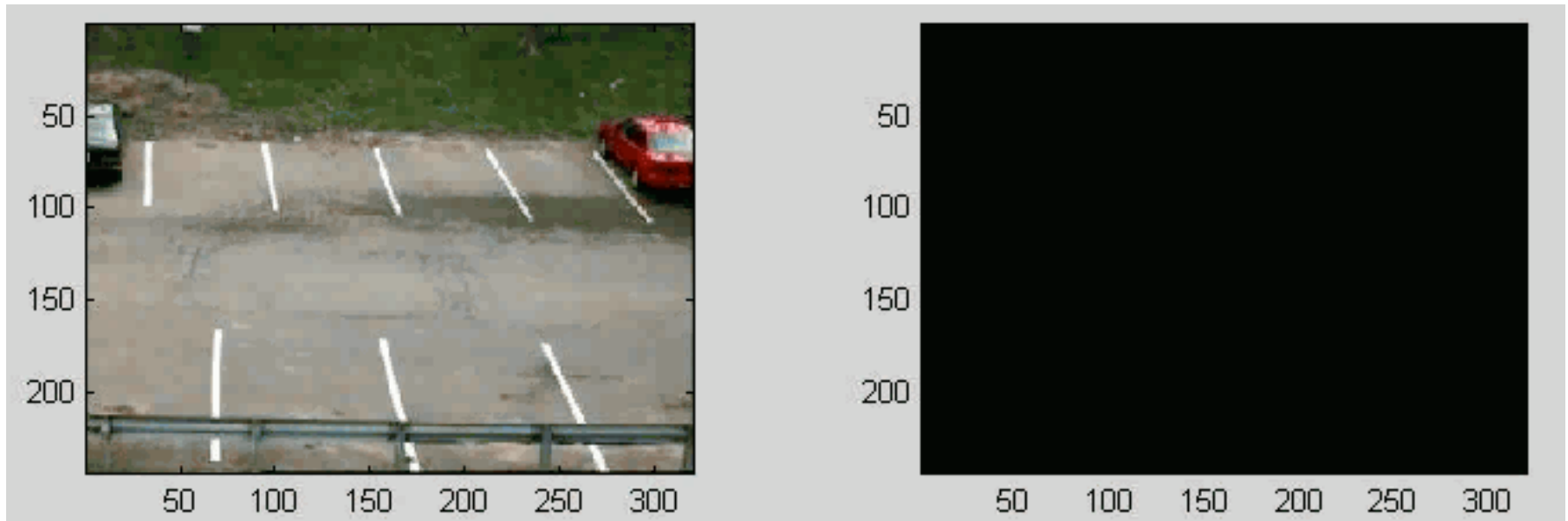
Grouping pixels... into blobs



Can we use Connected Components?

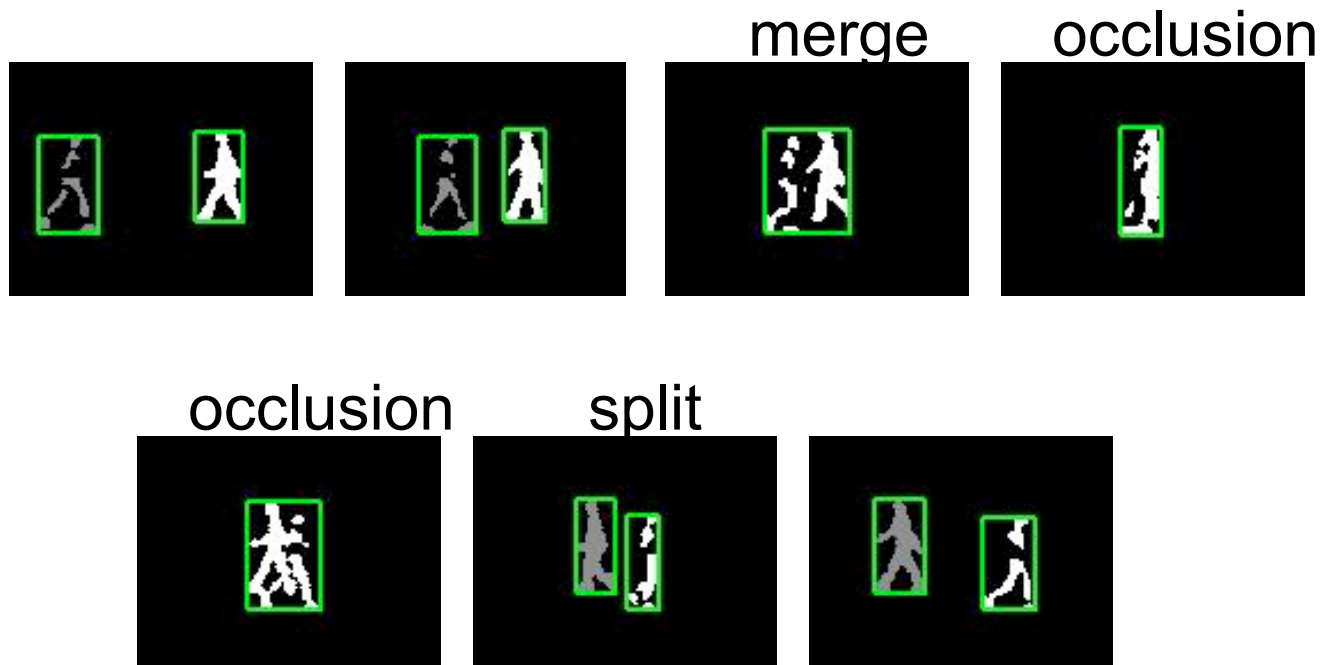


Simple Surveillance Results



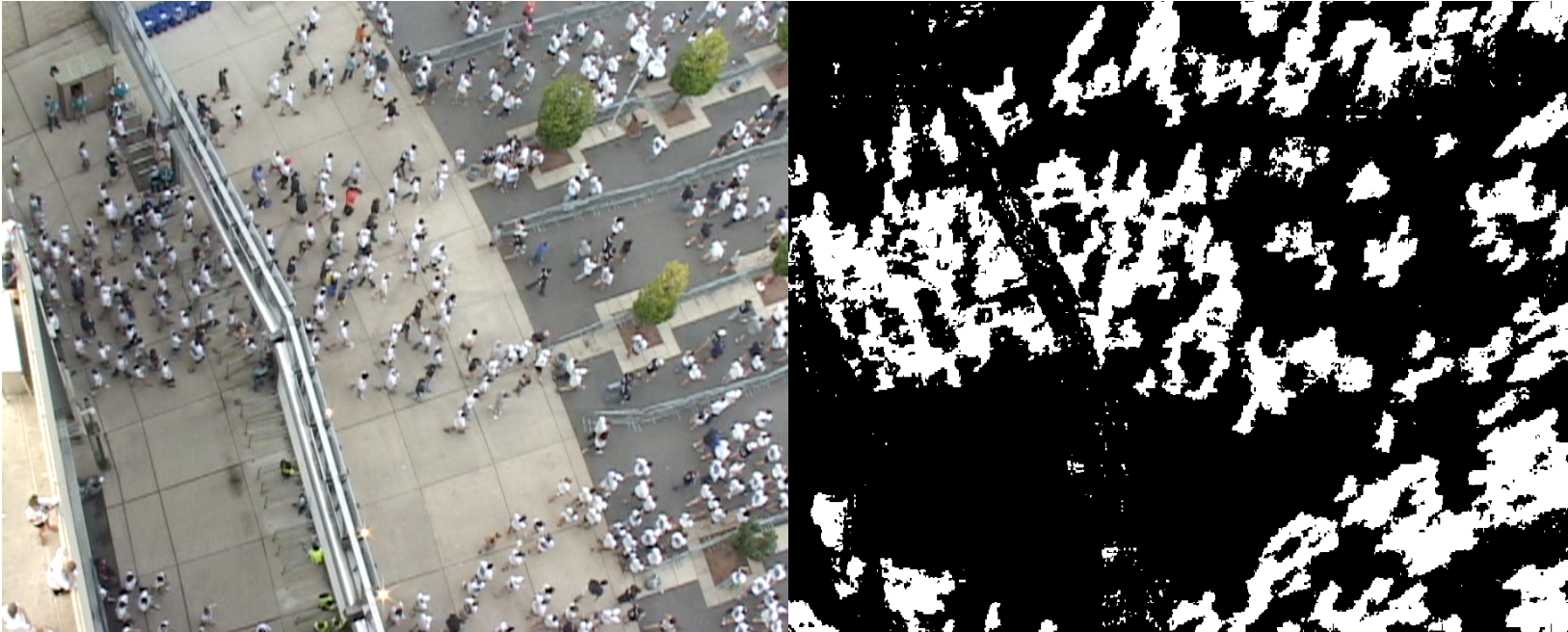
movie

Problem: Blob Merge/Split



When two objects pass close to each other, they are grouped as a single blob. Often, one object will become occluded by the other one. One of the challenging problems is to maintain correct labeling of each object after they split again.

Difficulty

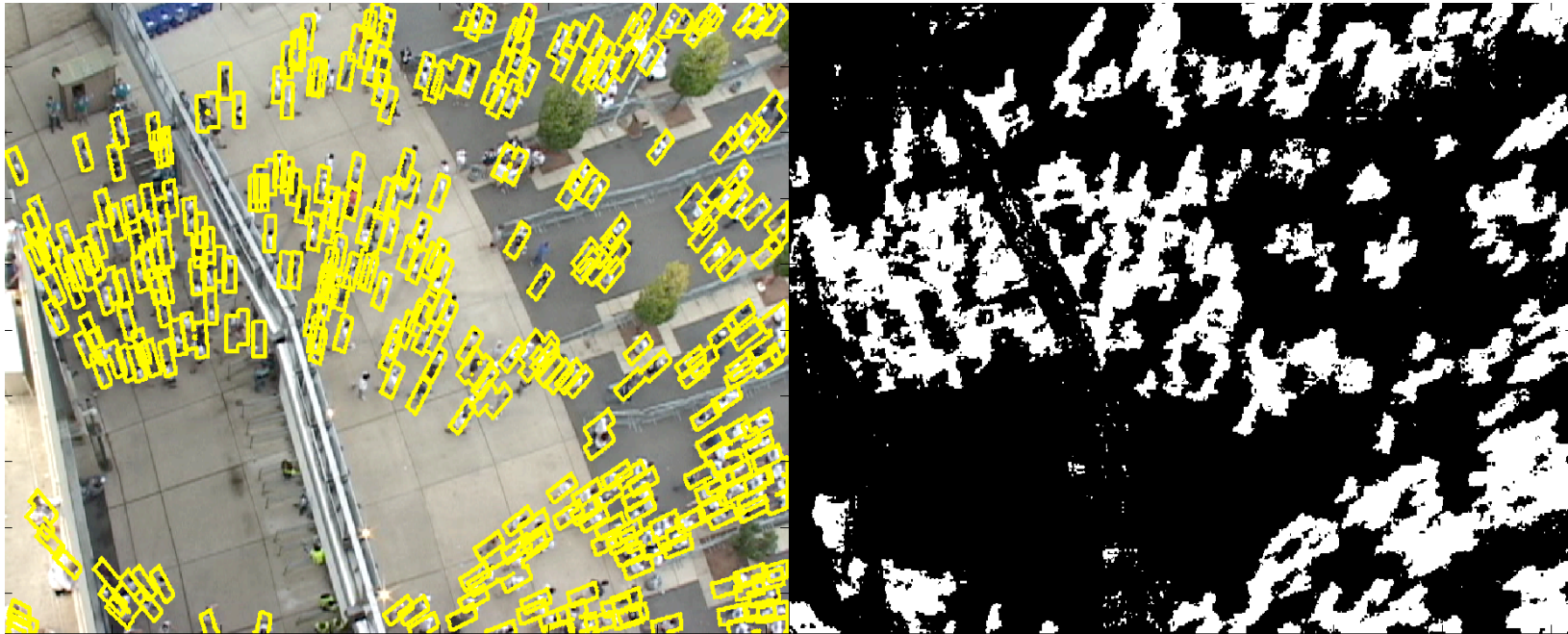


Each of these gets harder as the crowd gets denser!

Main components of a video surveillance system:

- Background subtraction
- Connected components to get blobs
- Data association to get trajectories

Our Approach*



Bayesian Marked Point Process

- the prior models expected size/shape of people, indexed by location
- the likelihood measures how well a proposed configuration explains the data
- MAP estimate of number/configuration of people is found using RJ-MCMC

*Weina Ge and R.Collins, "Marked Point Processes for Crowd Counting,"
IEEE Computer Vision and Pattern Recognition, Miami, FL, June 2009.

Foreground Object Detection

Bayesian Approaches

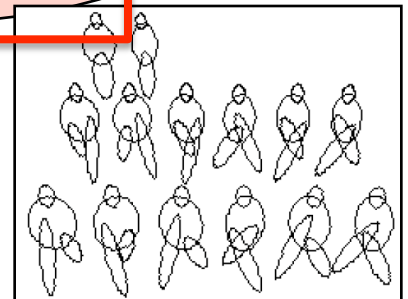
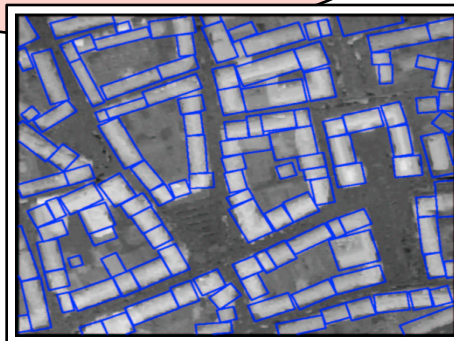
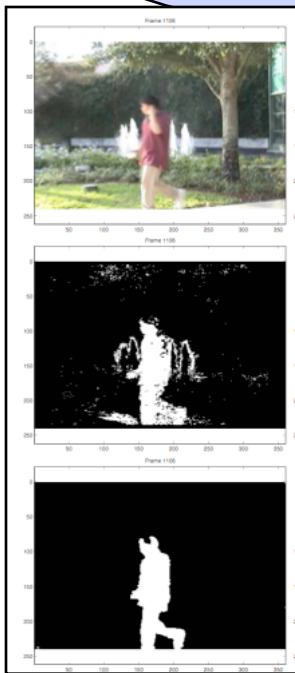
Non-Bayesian
[Dalal05; Rabaud06]

Pixel-wise MRF models
[Sheikh05]

Object-level models
(MPP)

Gibbs Point Process
[Ortner08]

Conditional Bayesian Models
[Rue99; Zhao03]



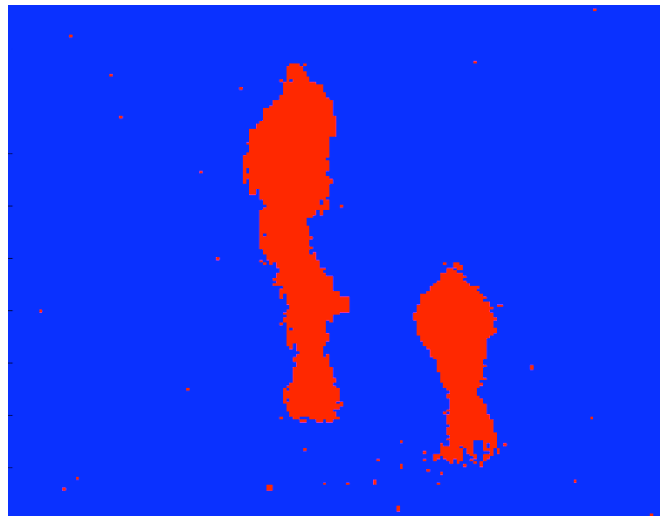
Marked Point Process

- Spatial Point Process
 - Distribution of a set of points in a bounded space
- Marked Point Process (MPP)
 - A spatial point process + a “mark” process
- Examples
 - Spatial process could model tree locations in a forest and the mark could be tree height
 - Spatial process could model cell locations on a microscope slide and the mark could be a polygonal representation of the cell boundary [Rue and Hurn, 1999]

Simplified Problem Statement

Given a foreground image, find a configuration of bounding boxes* that cover a majority of foreground pixels while leaving a majority of background pixels uncovered.

foreground
image



person-sized
bounding box

As an MPP:

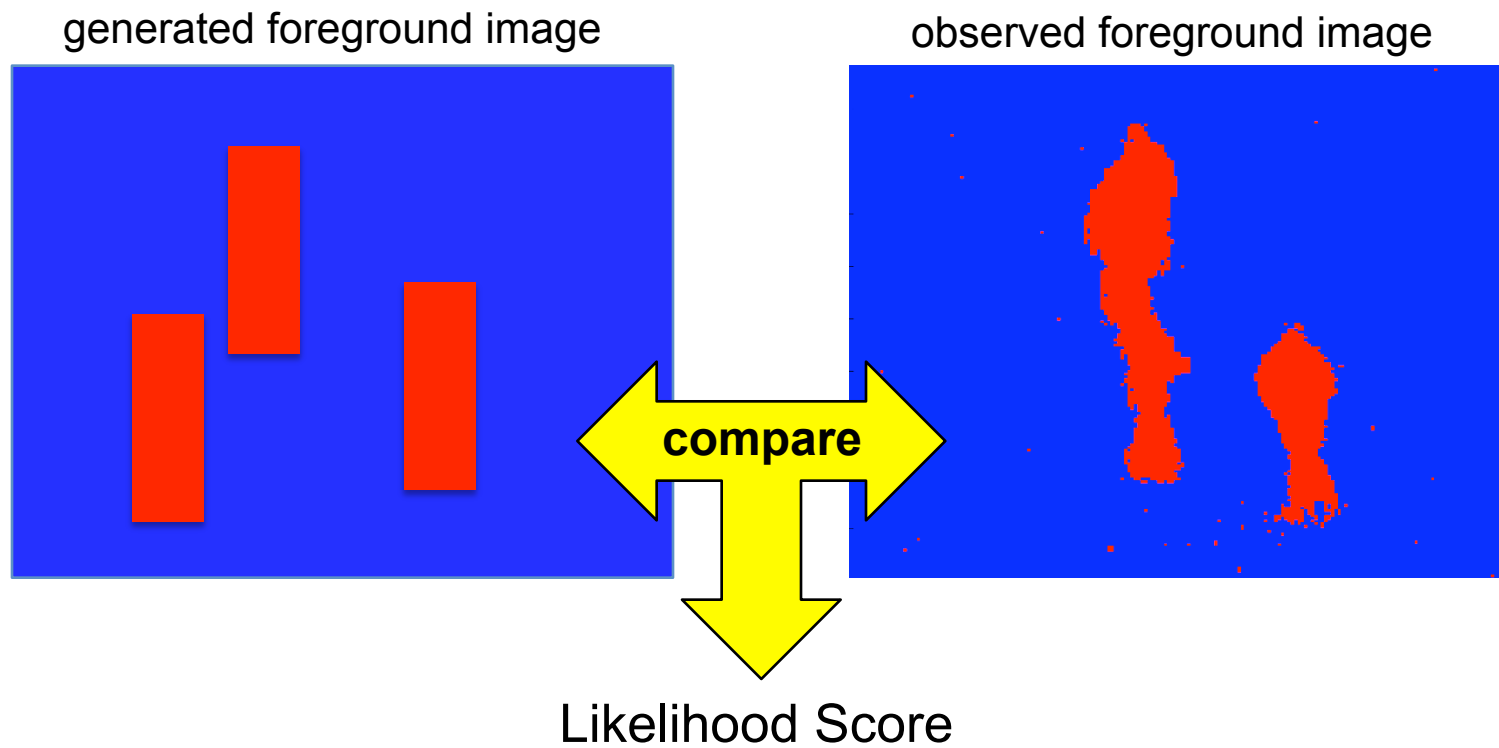
- Spatial process models number and (x,y) locations of bounding boxes
- Mark process models (height,width,orientation) of each bounding box.

*Footnote: We will add more realistic “shape” models in a moment

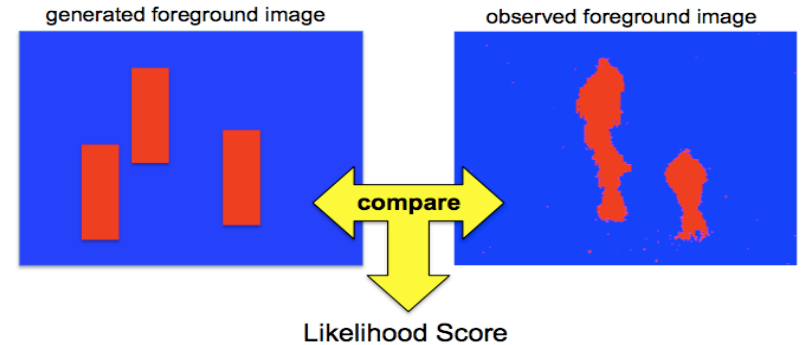
Likelihood Score

To measure how “good” a proposed configuration is, we generate a foreground image from it and compare with the observed foreground image to get a likelihood score.

$$\text{config} = \{\{x_1, y_1, w_1, h_1, \theta_1\}, \{x_2, y_2, w_2, h_2, \theta_2\}, \{x_3, y_3, w_3, h_3, \theta_3\}\}$$



Likelihood Score



Bernoulli distribution
model

$$\begin{aligned} p_{00} &= p(y_i = 0|x_i = 0) = \text{prob of observing background given a label of background} \\ p_{01} &= p(y_i = 0|x_i = 1) = \text{prob of observing background given a label of foreground} \\ p_{10} &= p(y_i = 1|x_i = 0) = \text{prob of observing foreground given a label of background} \\ p_{11} &= p(y_i = 1|x_i = 1) = \text{prob of observing foreground given a label of foreground} \end{aligned}$$

$$\begin{aligned} c_{00} &= \text{count of pixels where observation is background and label is background} \\ c_{01} &= \text{count of pixels where observation is background and label is foreground} \\ c_{10} &= \text{count of pixels where observation is foreground and label is background} \\ c_{11} &= \text{count of pixels where observation is foreground and label is foreground} \end{aligned}$$

likelihood

$$L(Y|X) = \prod_{i=1}^N p(y_i|x_i) = p_{00}^{c_{00}} p_{01}^{c_{01}} p_{10}^{c_{10}} p_{11}^{c_{11}}$$

simplify, by assuming

$$p_{00} = p_{11} = \mu \quad \text{and} \quad p_{01} = p_{10} = 1 - \mu$$

log likelihood

$$\begin{aligned} \log L(Y|X) &= (c_{00} + c_{11}) \log \mu + (c_{01} + c_{10}) \log(1 - \mu) \\ &= [N - (c_{01} + c_{10})] \log \mu + (c_{01} + c_{10}) \log(1 - \mu) \\ &= N \log \mu - (c_{01} + c_{10}) [\log \mu - \log(1 - \mu)] \end{aligned}$$

Number of pixels
that disagree!

Prior Model

We use a prior to model our expectations about bounding box configurations in the image

$$\underbrace{\pi(\mathbf{o}_i)} = \underbrace{\pi(p_i)} \underbrace{\pi(w_i, h_i, \theta_i | p_i)}$$

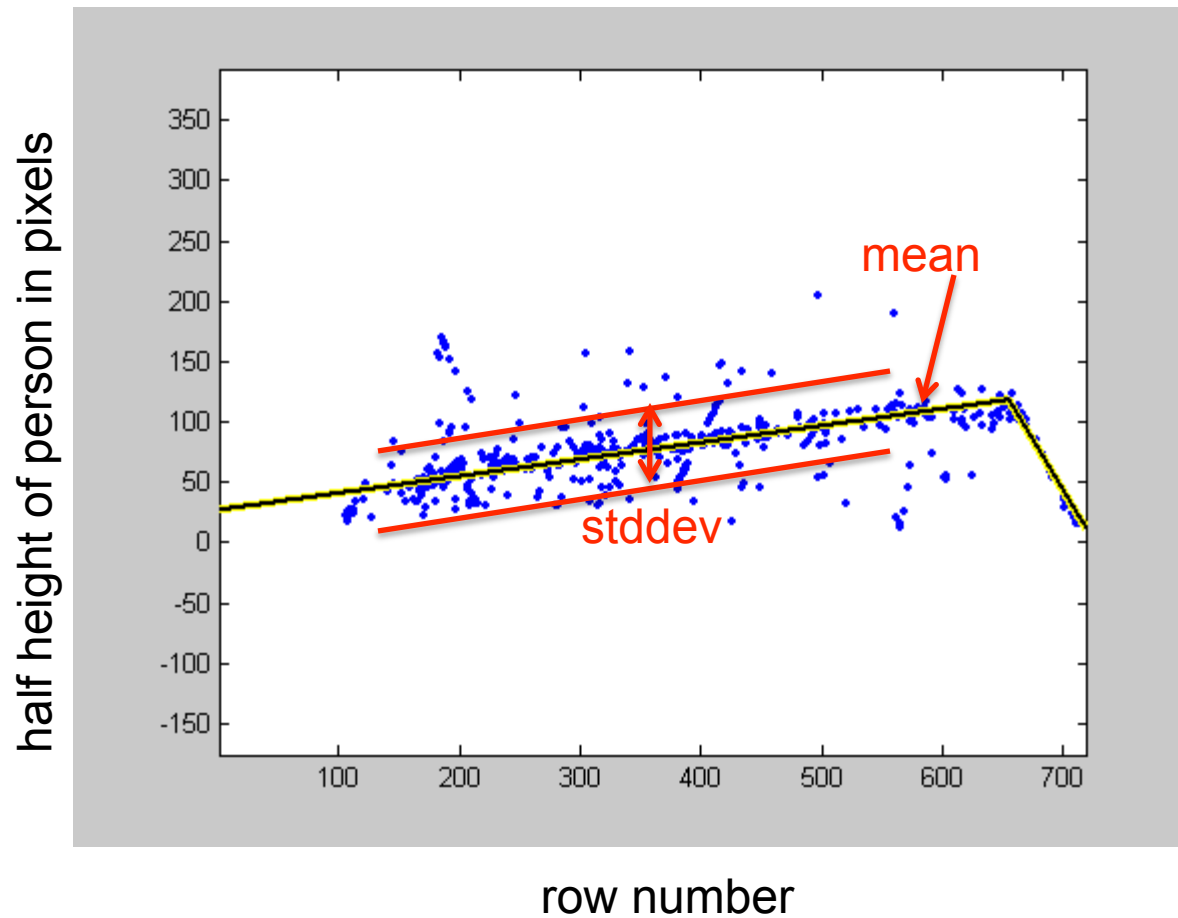
Prior for bounding box i

Prior on location (center point)

Prior on bounding box height, width and orientation, conditioned on center location.

Estimating Priors

Example: learning height distribution as a function of image row

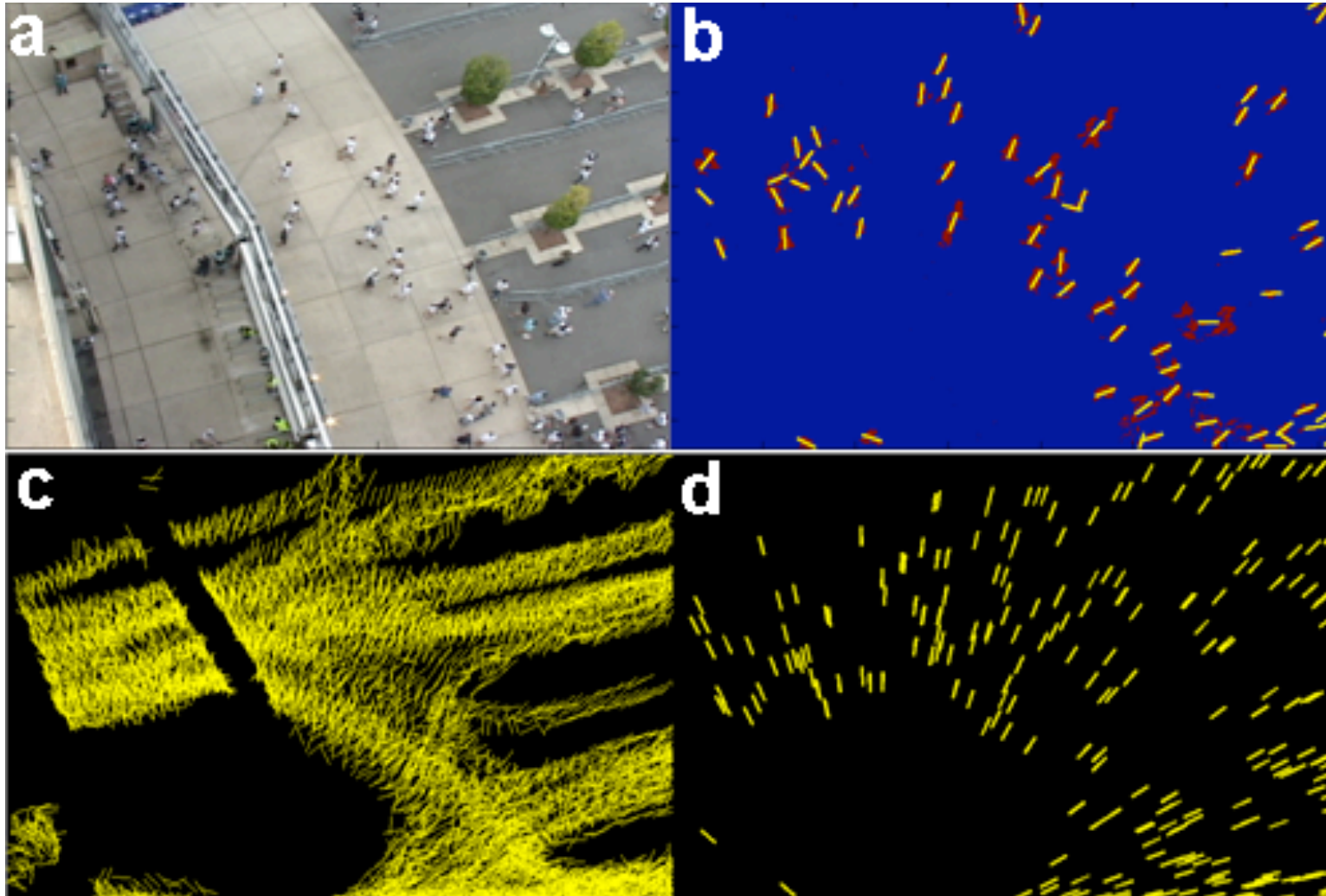


Estimating Priors

Example: learning orientation as a function of image location

sample frame

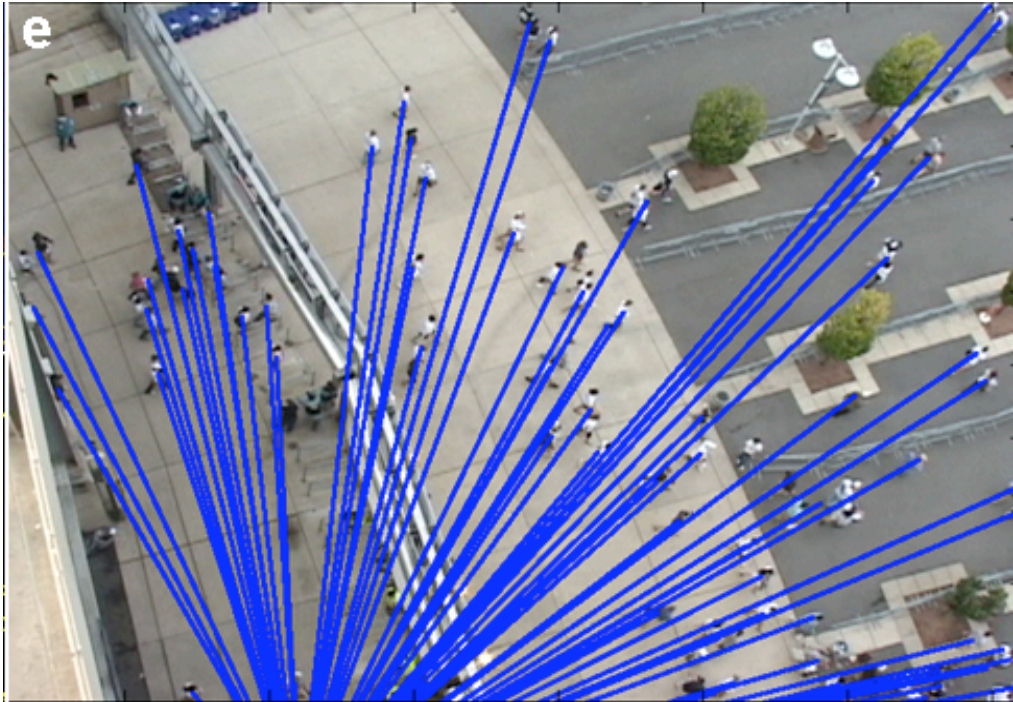
extracted blobs / axes



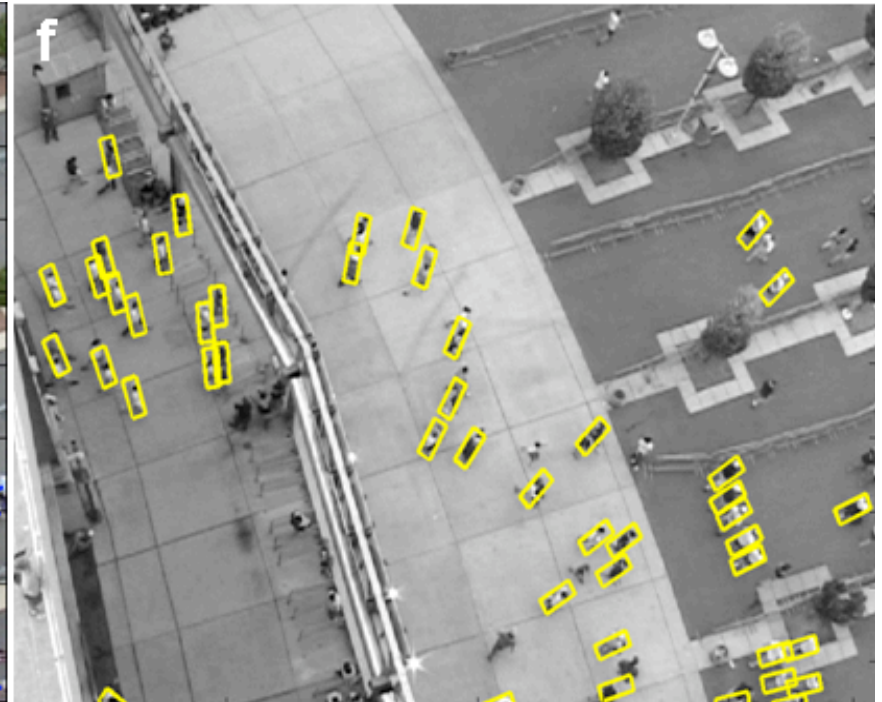
SU-VLPR2010 extracted from training sequence

inliers for VP estimation

Estimating Priors



estimated vanishing point



Scaled, oriented rectangles

Bottom line: it is not difficult to estimate priors on location, size and orientation of people as seen from a specific viewpoint.

Searching for the Max

The space of configurations is very large. We can't exhaustively search for the max likelihood configuration. We can't even really uniformly sample the space to a reasonable degree of accuracy.

$$\text{config}_k = \{\{x_1, y_1, w_1, h_1, \text{theta}_1\}, \{x_2, y_2, w_2, h_2, \text{theta}_2\}, \dots, \{x_k, y_k, w_k, h_k, \text{theta}_k\}\}$$

Let N = number of possible locations for (x_i, y_i) in a k -person configuration.

Size of $\text{config}_k = N^k$

And we don't even know how many people there are...

Size of config space = $N^0 + N^1 + N^2 + N^3 + \dots$

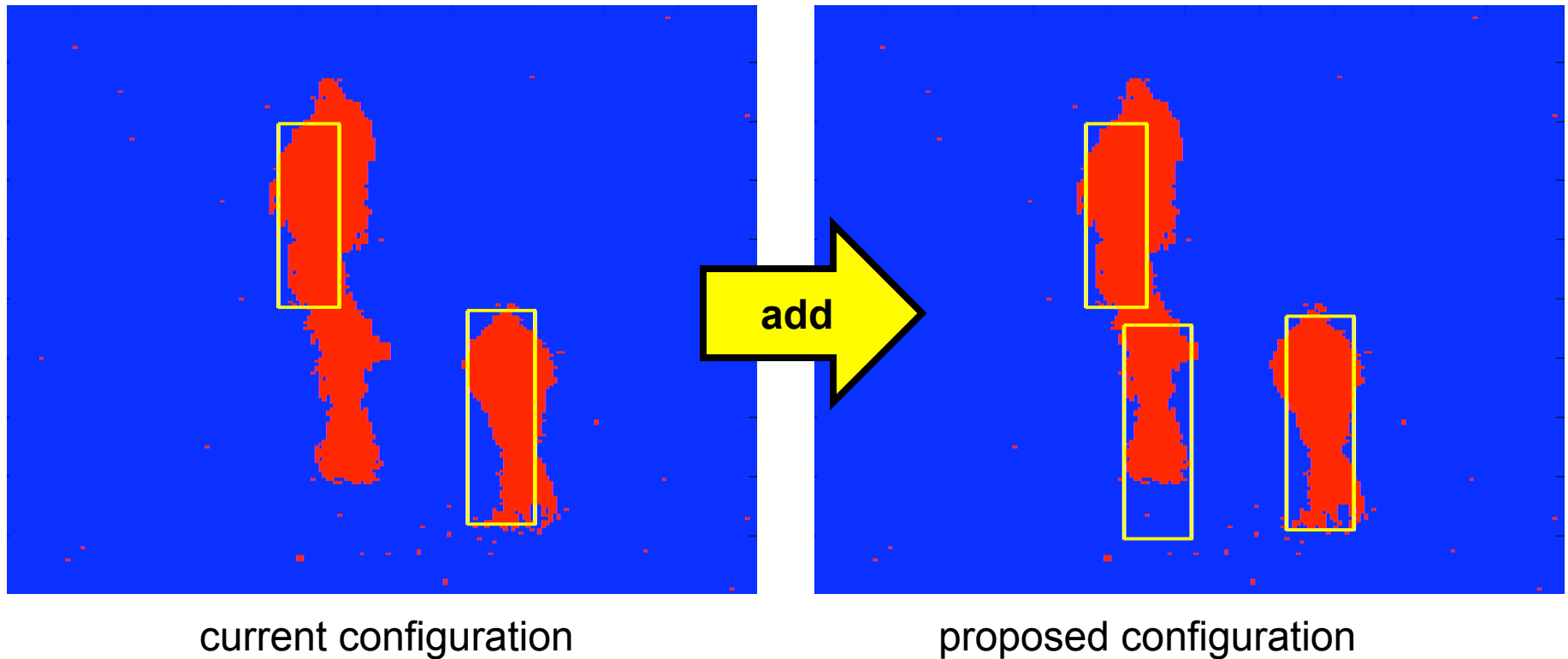
If we also wanted to search for width, height and orientation, this space would be even more huge.

Searching for the Max

- Local Search Approach
 - Given a current configuration, propose a small change to it
 - Compare likelihood of proposed config with likelihood of the current config
 - Decide whether to accept the change

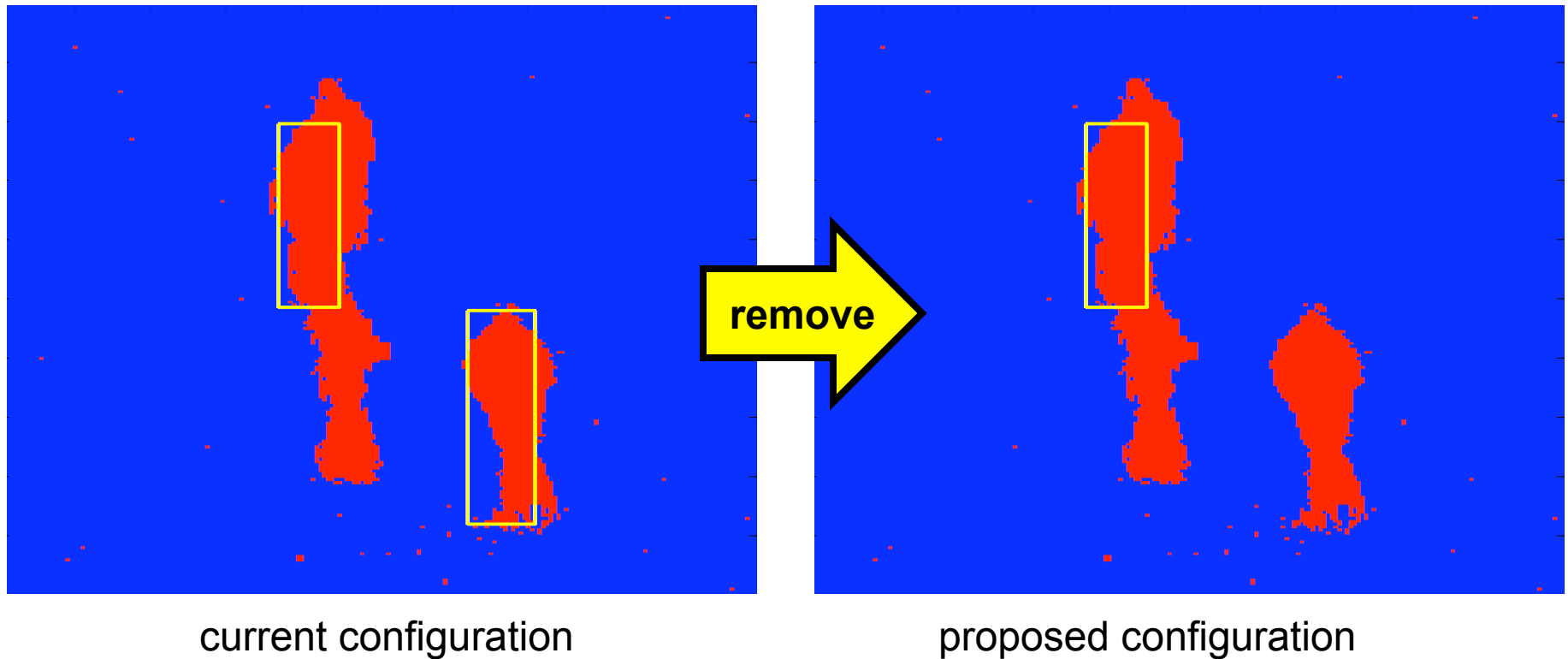
Proposals

- Add a rectangle (birth)



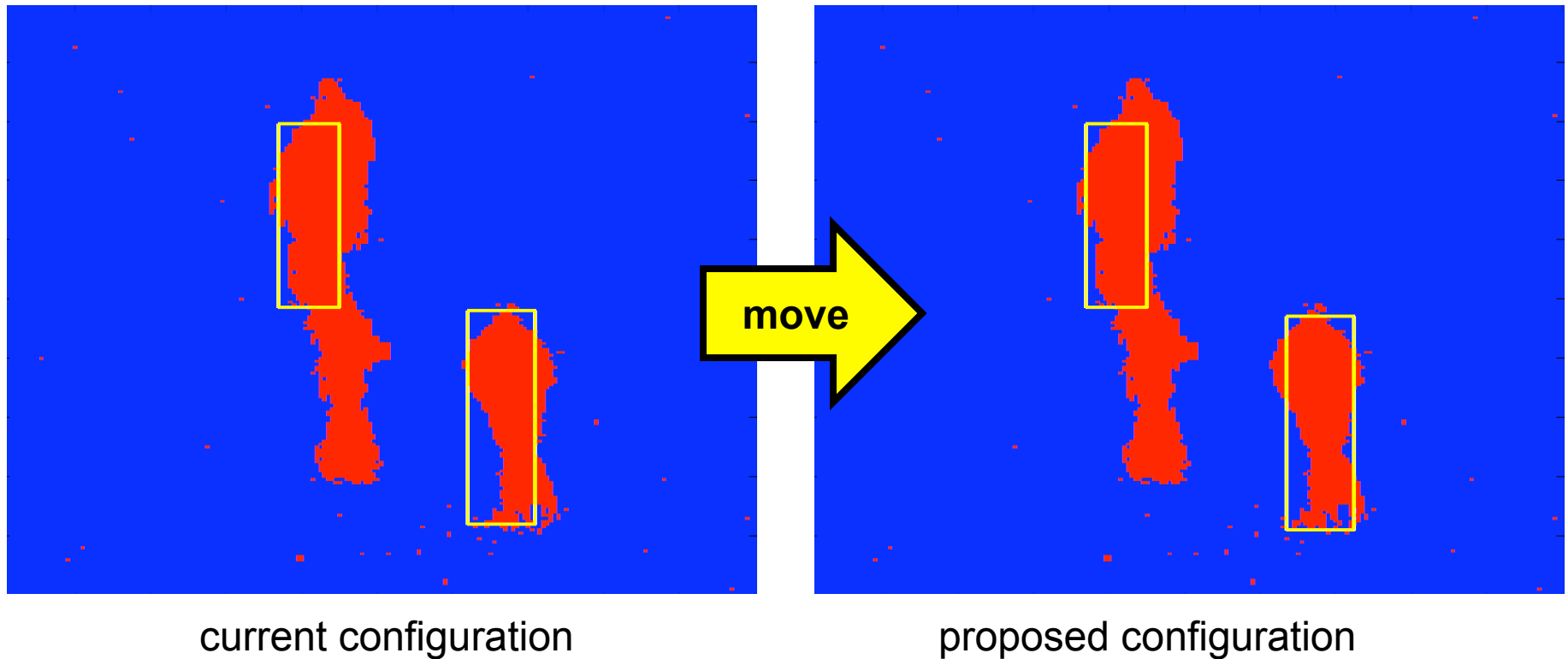
Proposals

- Remove a rectangle (death)



Proposals

- Move a rectangle



Searching for the Max

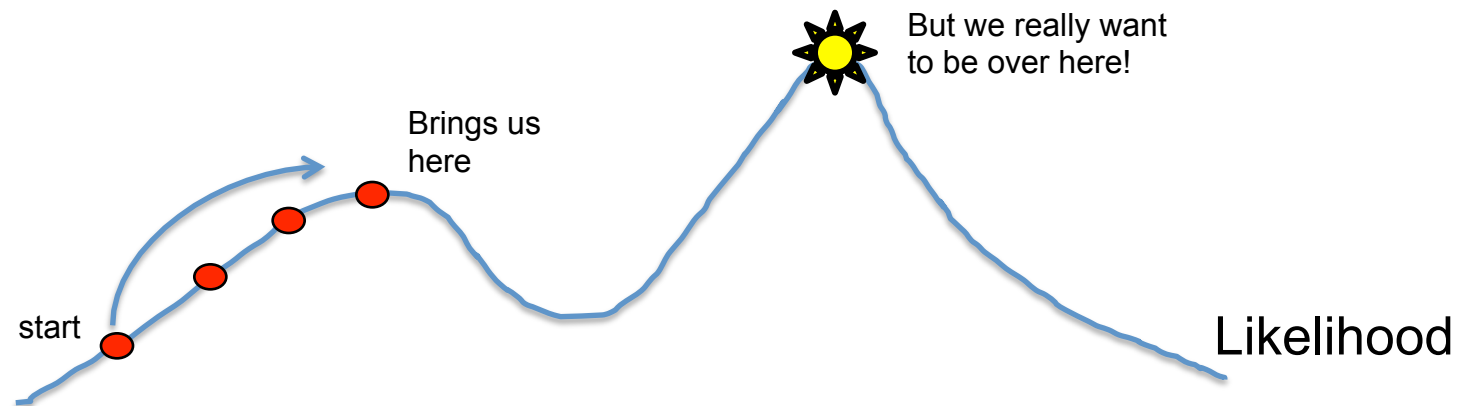
- Naïve Acceptance

- Accept proposed configuration if it has a larger likelihood score, i.e.

$$\text{Compute } a = \frac{L(\text{proposed})}{L(\text{current})}$$

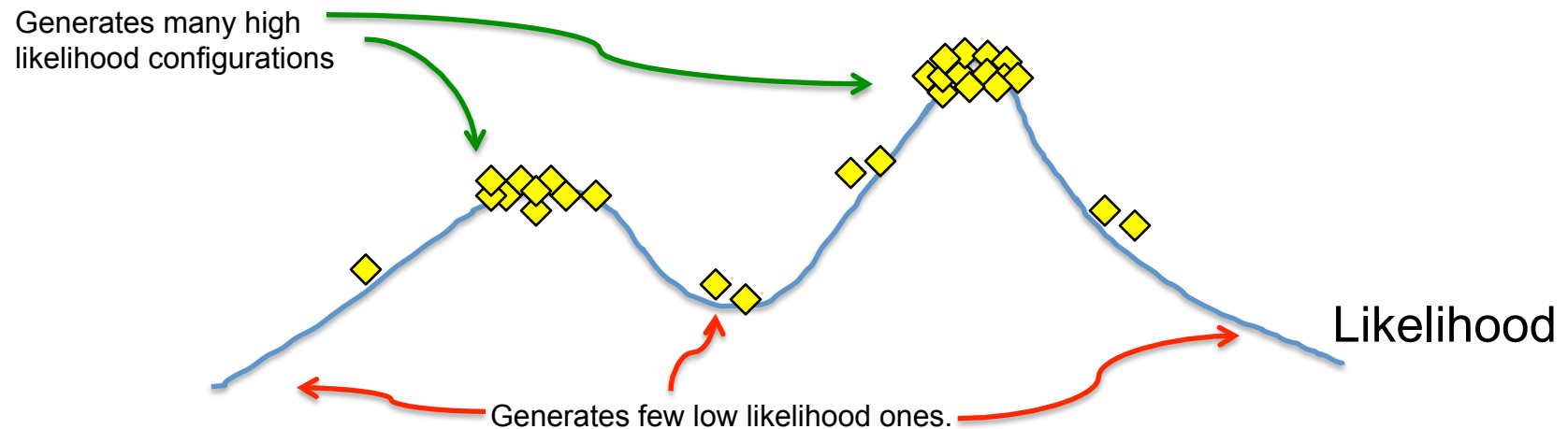
Accept if $a > 1$

- Problem: leads to hill-climbing behavior that gets stuck in local maxima



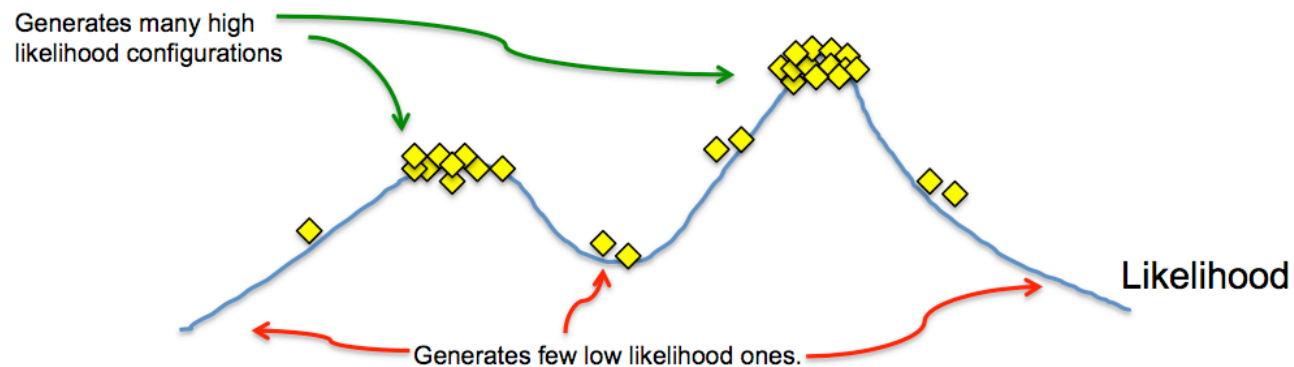
Searching for the Max

- The MCMC approach
 - Generate random configurations from a distribution proportional to the likelihood!



Searching for the Max

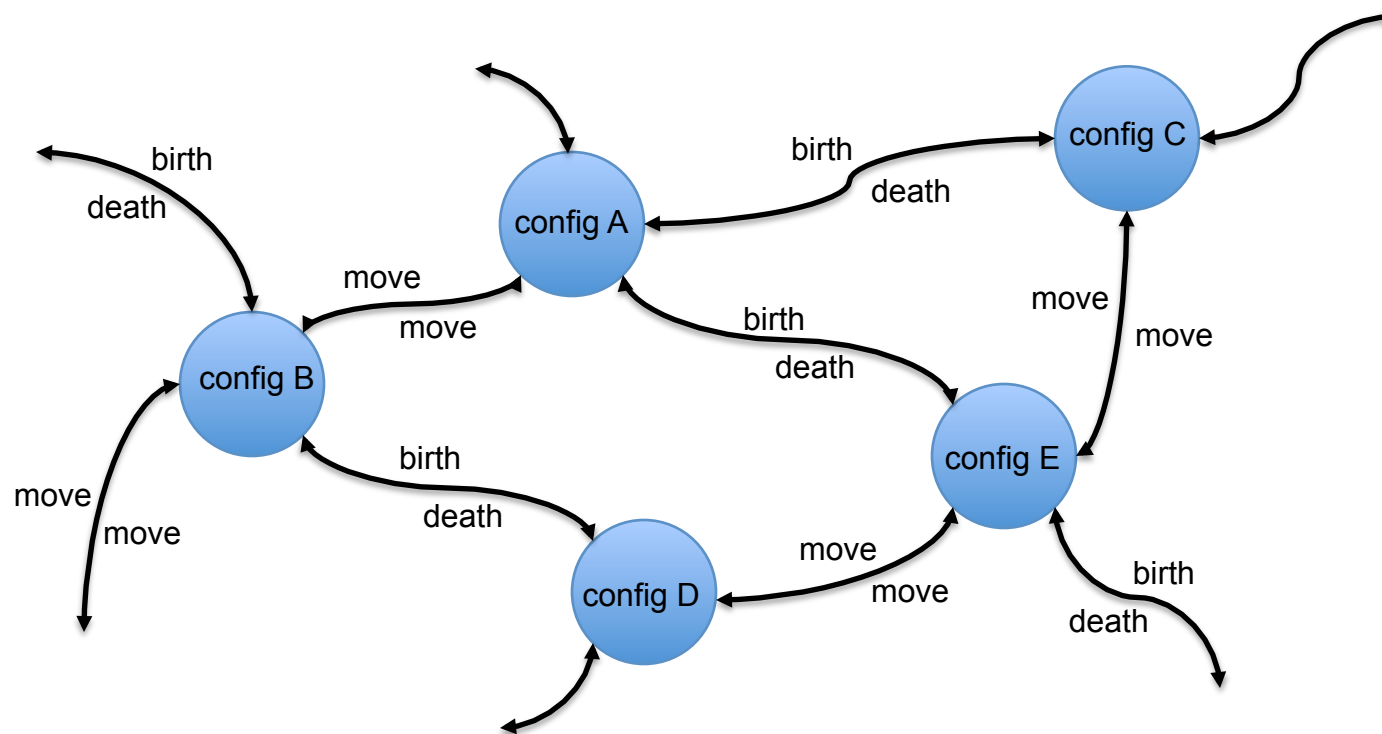
- The MCMC approach
 - Generate random configurations from a distribution proportional to the likelihood!



- This searches the space of configurations in an efficient way.
- Now just remember the generated configuration with the highest likelihood.

Sounds good, but how to do it?

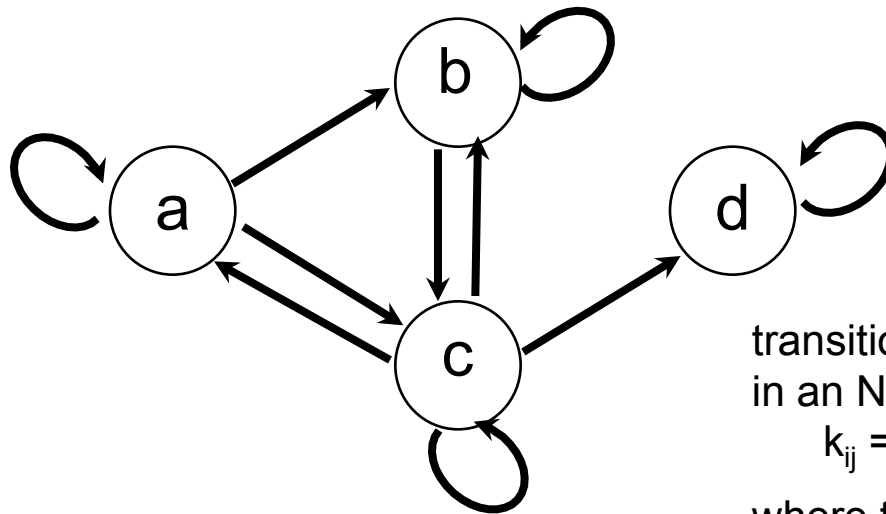
- Think of configurations as nodes in a graph.
- Put a link between nodes if you can get from one config to the other in one step (birth, death, move)



Recall: Markov Chains

Markov Chain:

- A sequence of random variables X_1, X_2, X_3, \dots
- Each variable is a distribution over a set of states (a,b,c...)
- Transition probability of going to next state only depends on the current state. e.g. $P(X_{n+1} = a \mid X_n = b)$

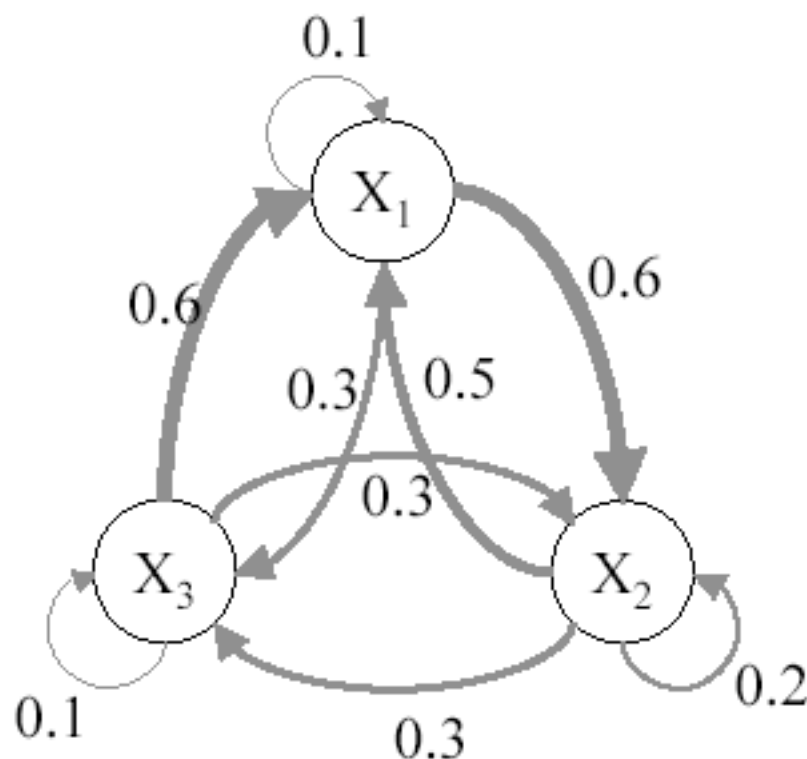


transition probs can be arranged
in an $N \times N$ table of elements

$$k_{ij} = P(X_{n+1}=j \mid X_n = i)$$

where the rows sum to one

A simple Markov chain

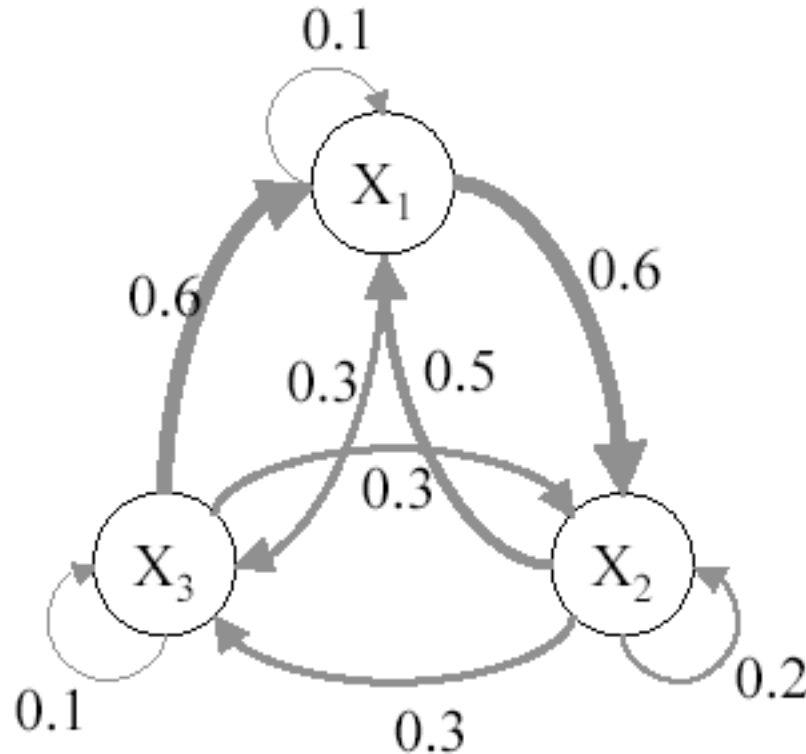


$$K = \begin{bmatrix} 0.1 & 0.5 & 0.6 \\ 0.6 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.1 \end{bmatrix}$$

K = transpose of transition prob table $\{k_{ij}\}$ (columns sum to one. We do this for computational convenience.

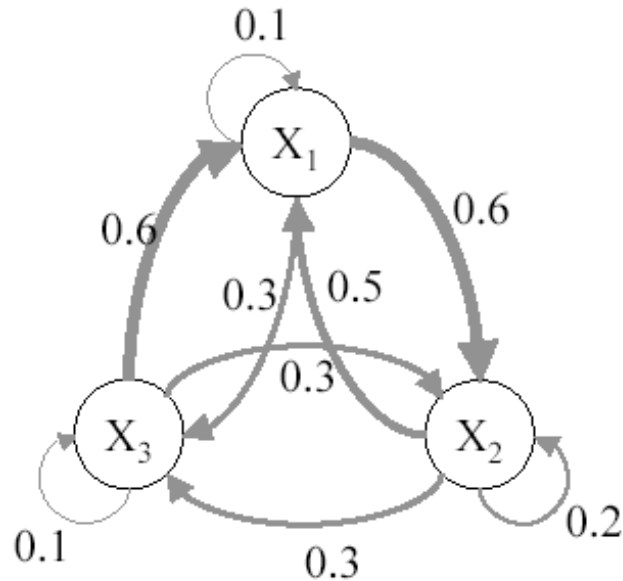
Question:

Assume you start in some state, and then run the simulation for a large number of time steps. What percentage of time do you spend at X_1 , X_2 and X_3 ?



$$K = \begin{bmatrix} 0.1 & 0.5 & 0.6 \\ 0.6 & 0.2 & 0.3 \\ 0.3 & 0.3 & 0.1 \end{bmatrix}$$

Experimental Approach



Start in some state, and then run the simulation for some number of time steps. After you have run it “long enough” start keeping track of the states you visit.

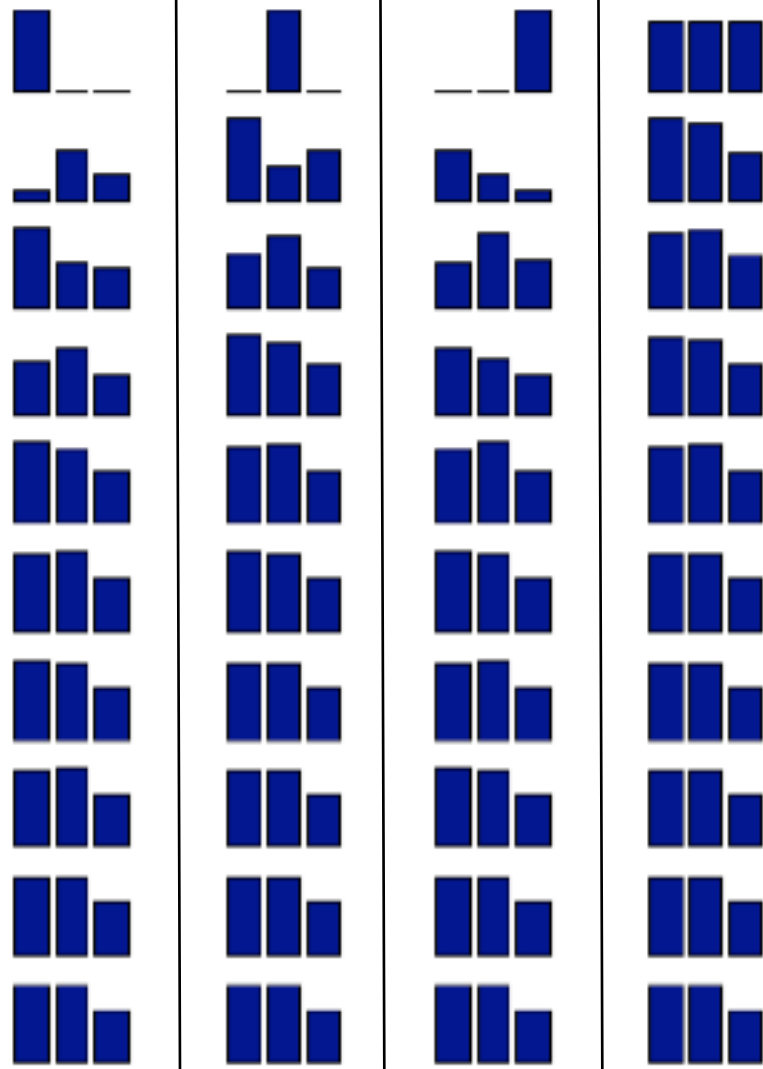
{... X1 X2 X1 X3 X3 X2 X1 X2 X1 X1 X3 X3 X2 ...}

These are samples from the distribution you want, so you can now compute any expected values with respect to that distribution empirically.

Analytic Approach

four possible initial distributions

[1 0 0] [0 1 0] [0 0 1] [.33 .33 .33]



q_0 initial distribution

$q_1 = K q_0$ distribution after one time step

$q_2 = K q_1 = K^2 q_0$

$q_3 = K q_2 = K^2 q_1 = K^3 q_0$

$q_{10} = K q_9 = \dots K^{10} q_0$

all eventually end up with same distribution -- this is the stationary distribution!

Eigen-analysis

K =

0.1000	0.5000	0.6000
0.6000	0.2000	0.3000
0.3000	0.3000	0.1000

$$KE = ED$$

in matlab:
 $[E,D] = \text{eigs}(K)$

E =

0.6396	0.7071	-0.2673
0.6396	-0.7071	0.8018
0.4264	0.0000	-0.5345

Eigenvalue v_1 always 1

Stationary distribution

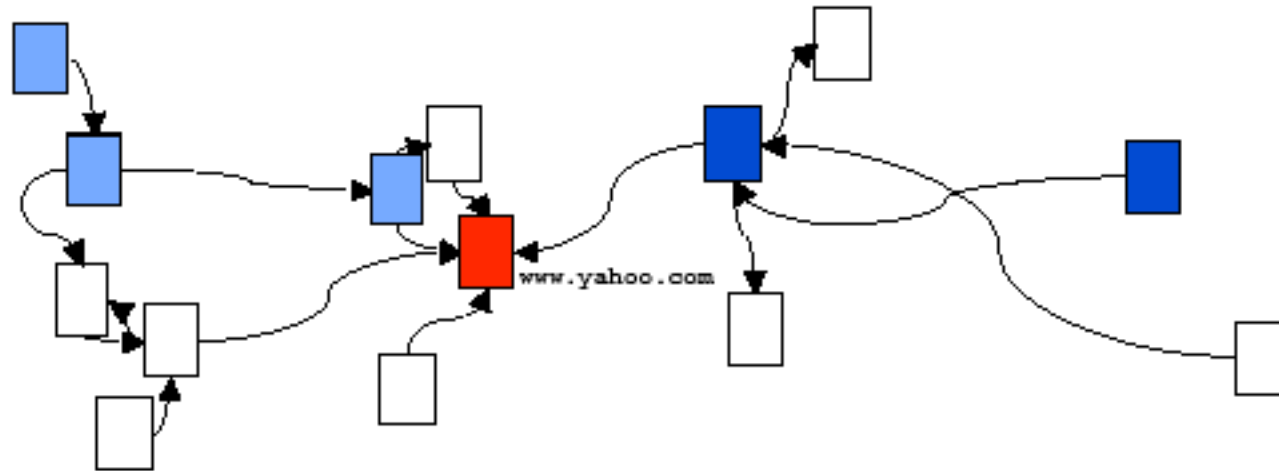
$$\pi = e_1 / \text{sum}(e_1)$$

$$\text{i.e. } K \pi = \pi$$

D =

1.0000	0	0
0	-0.4000	0
0	0	-0.2000

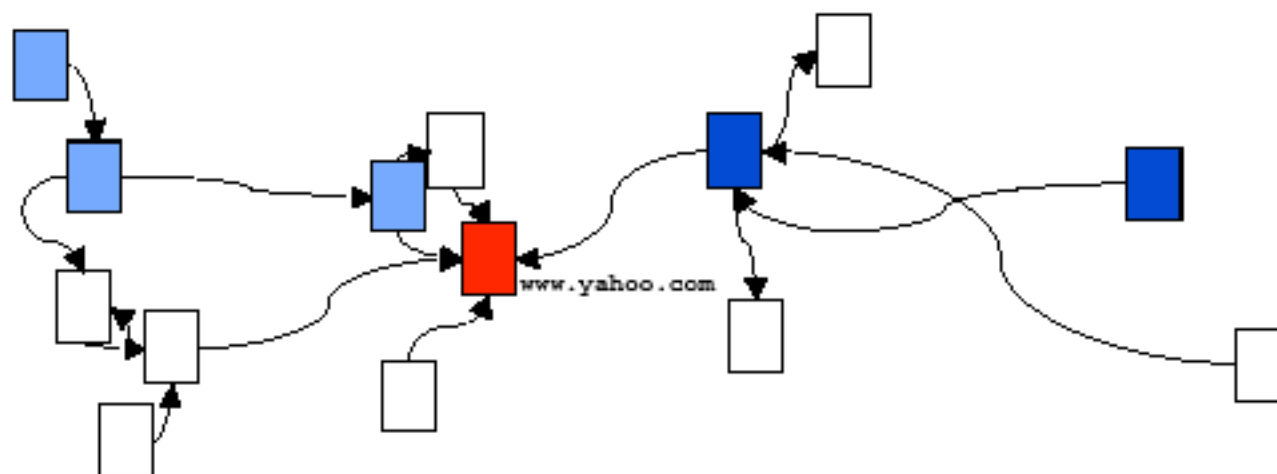
The Web as a Markov Chain



The PageRank of a webpage as used by Google is defined by a Markov chain. It is the probability to be at page i in the stationary distribution on the following Markov chain on all (known) webpages. If N is the number of known webpages, and a page i has k_i links then it has transition probability $(1-q)/k_i + q/N$ for all pages that are linked to and q/N for all pages that are not linked to. The parameter q is taken to be about 0.15.

Google Pagerank

Pagerank == First Eigenvector of the Web Graph !



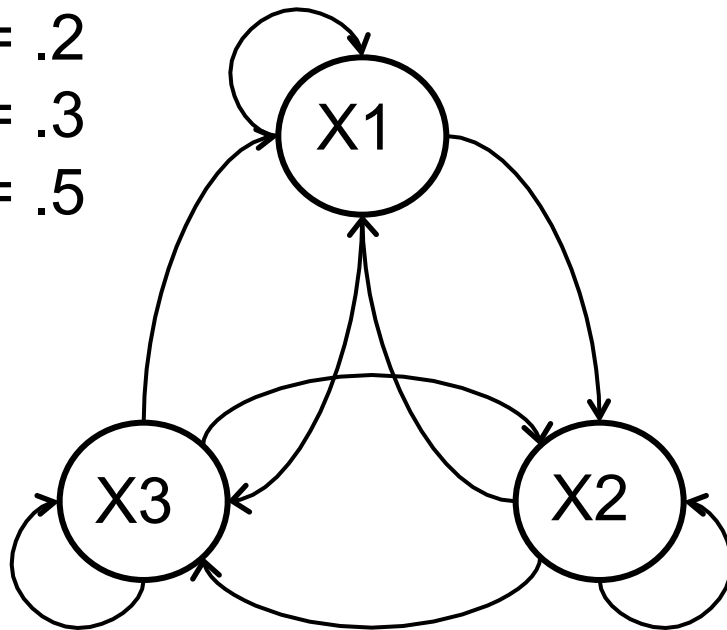
Computation assumes a 15% "random restart" probability

Sergey Brin and Lawrence Page , The anatomy of a large-scale hypertextual
{Web} search engine, Computer Networks and ISDN Systems, 1998

But How to Design a Chain?

Assume you want to spend a particular percentage of time at X1, X2 and X3. What should the transition probabilities be?

$$\begin{aligned} P(x1) &= .2 \\ P(x2) &= .3 \\ P(x3) &= .5 \end{aligned}$$



$$K = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

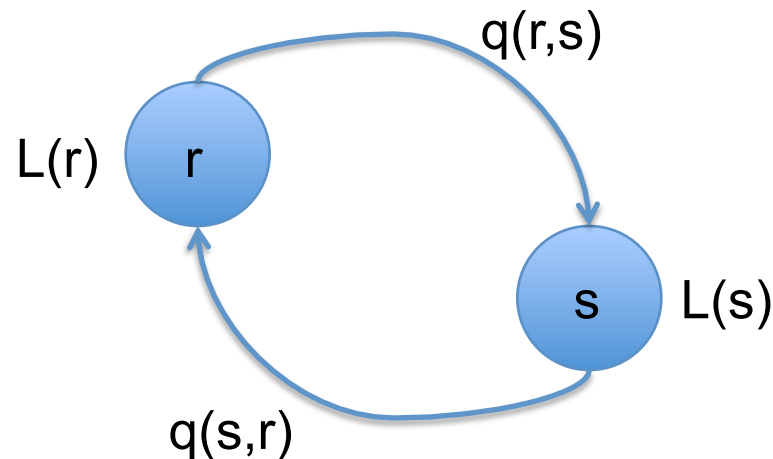
Detailed Balance

- Consider a pair of configuration nodes r, s
- Want to generate them with frequency relative to their likelihoods $L(r)$ and $L(s)$
- Let $q(r, s)$ be relative frequency of proposing configuration s when the current state is r (and vice versa)

A sufficient condition to generate r, s with the desired frequency is

$$L(r) q(r, s) = L(s) q(s, r)$$

“detailed balance”



Detailed Balance

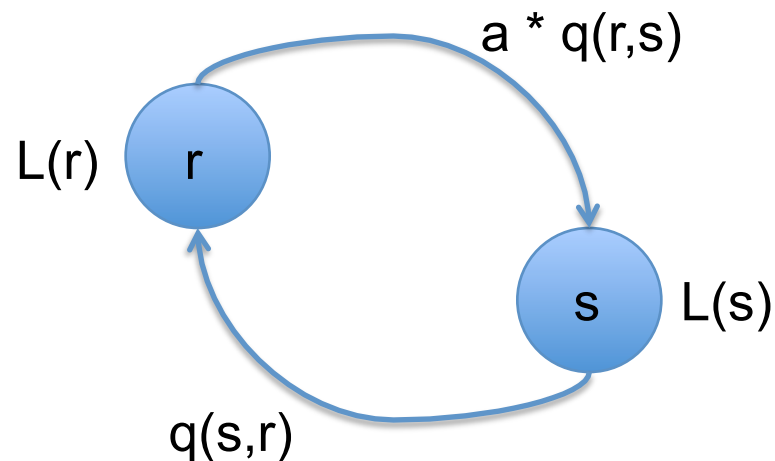
- Typically, your proposal frequencies do NOT satisfy detailed balance (unless you are extremely lucky).
- To “fix this”, we introduce a computational fudge factor a

Detailed balance:

$$a * L(r) q(r,s) = L(s) q(s,r)$$

Solve for a :

$$a = \frac{L(s) q(s,r)}{L(r) q(r,s)}$$



MCMC Sampling

- Metropolis Hastings algorithm

Propose a new configuration

$$\text{Compute } a = \frac{L(\text{proposed})}{L(\text{current})} \frac{q(\text{proposed}, \text{current})}{q(\text{current}, \text{proposed})}$$

Accept if $a > 1$

Else accept anyways with probability a

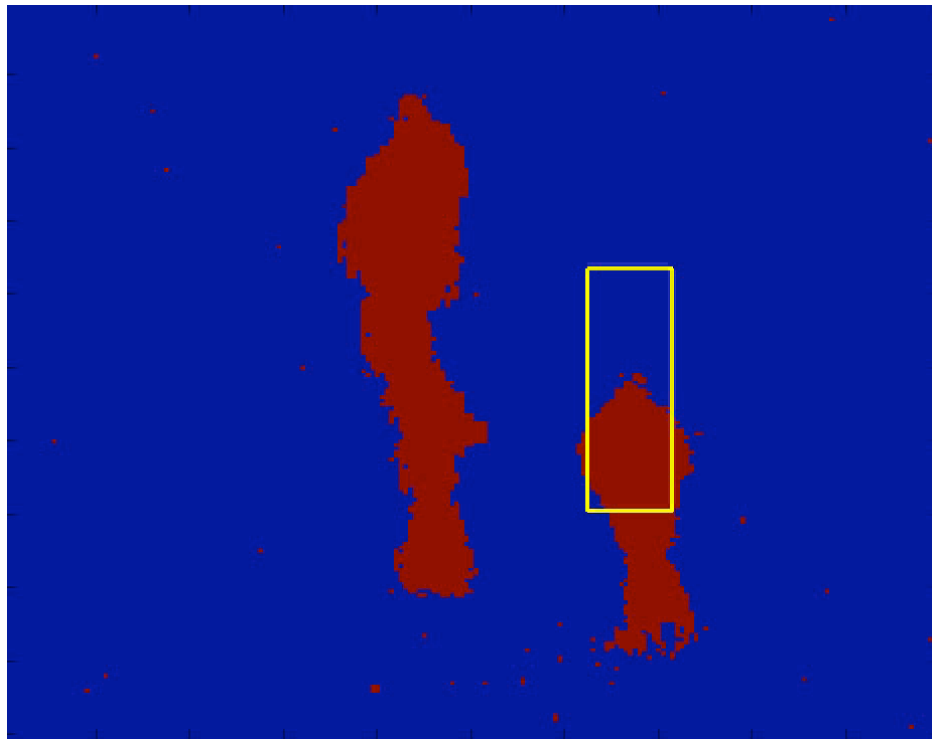
Difference from
Naïve algorithm

Trans-dimensional MCMC

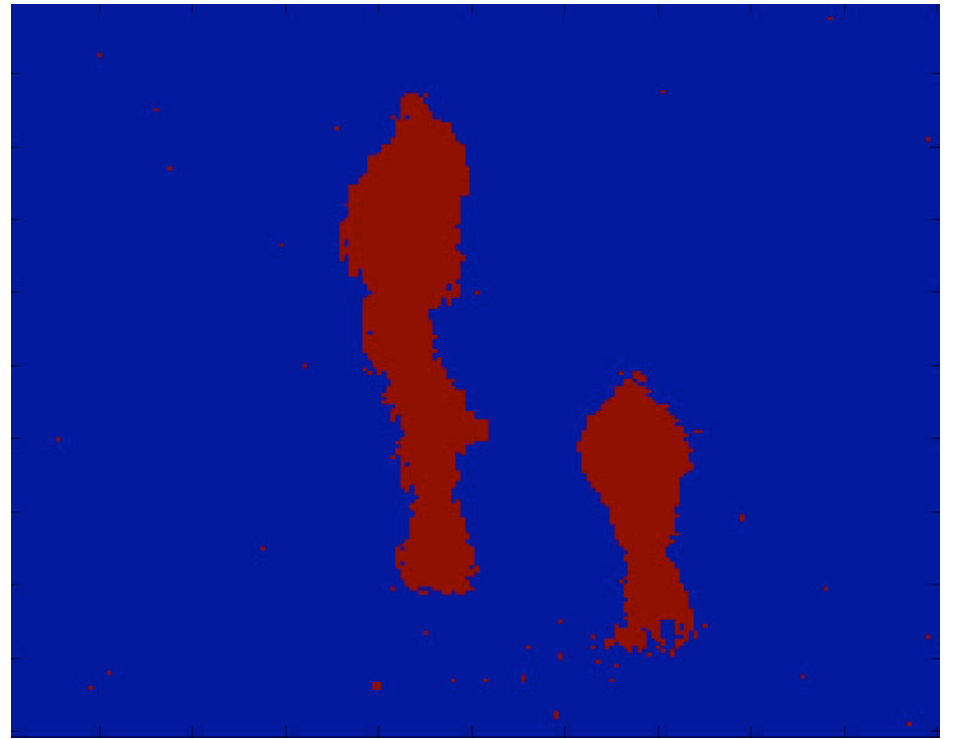
- Green's reversible-jump approach (RJMCMC) gives a general template for exploring and comparing states of differing dimension (diff numbers of rectangles in our case).
- Proposals come in reversible pairs: birth/death and move/move.
- We should add another term to the acceptance ratio for pairs that jump across dimensions. However, that term is 1 for our simple proposals.

MCMC in Action

Sequence of proposed configurations



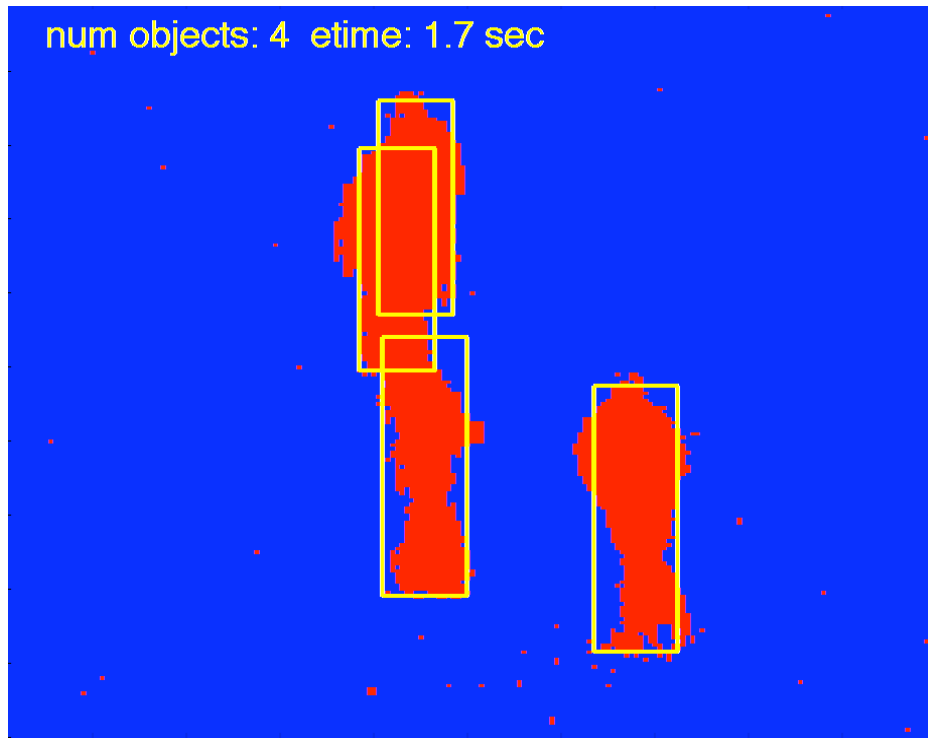
Sequence of "best" configurations



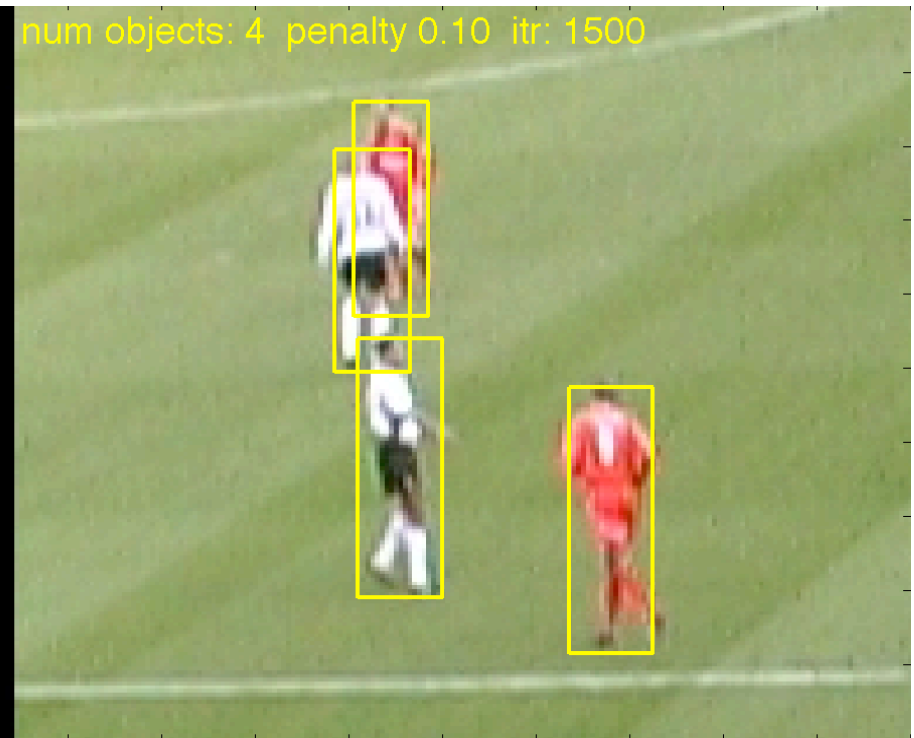
movies

MCMC in Action

MAP configuration

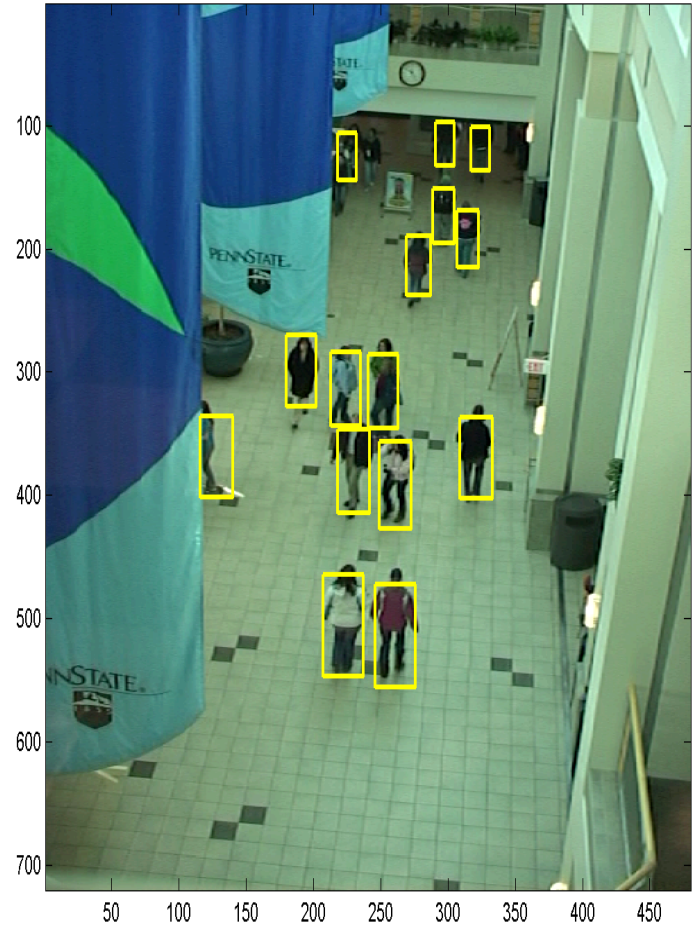
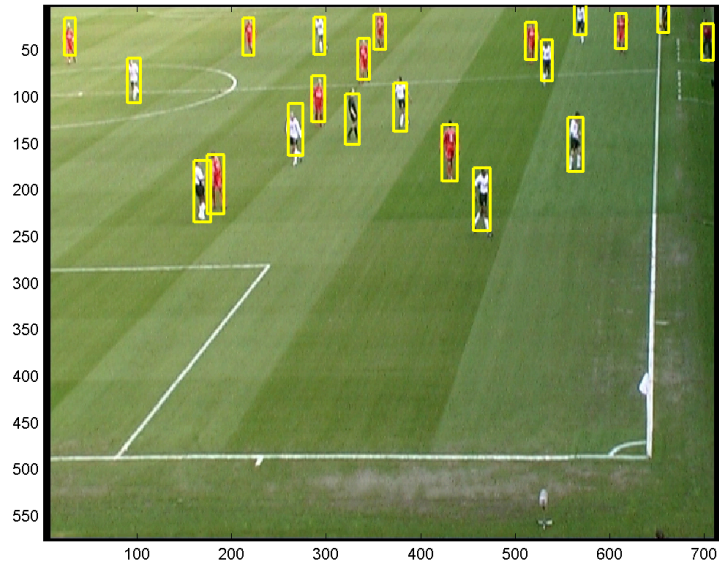


Looking good!



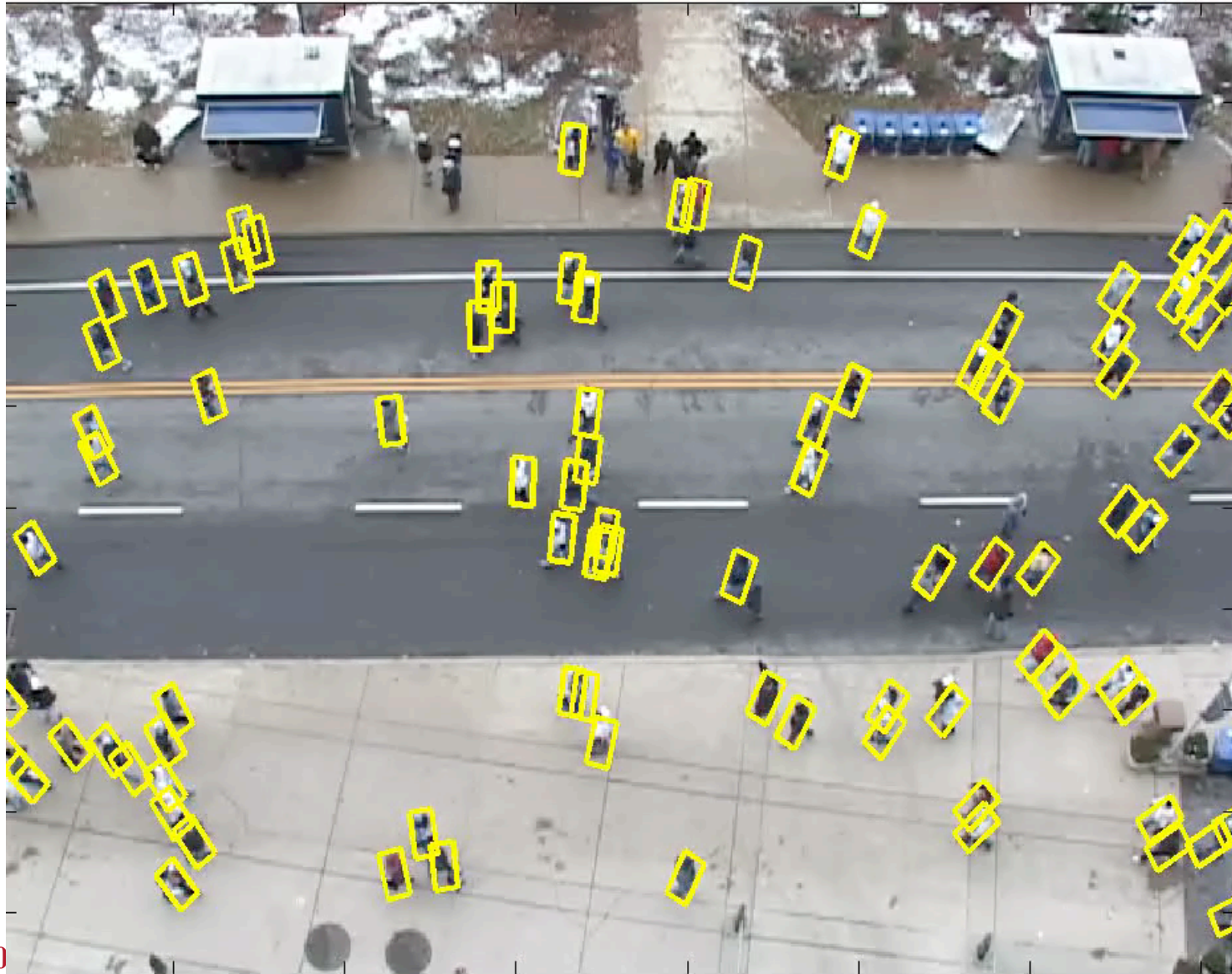
Robert Collins
Penn State

Examples



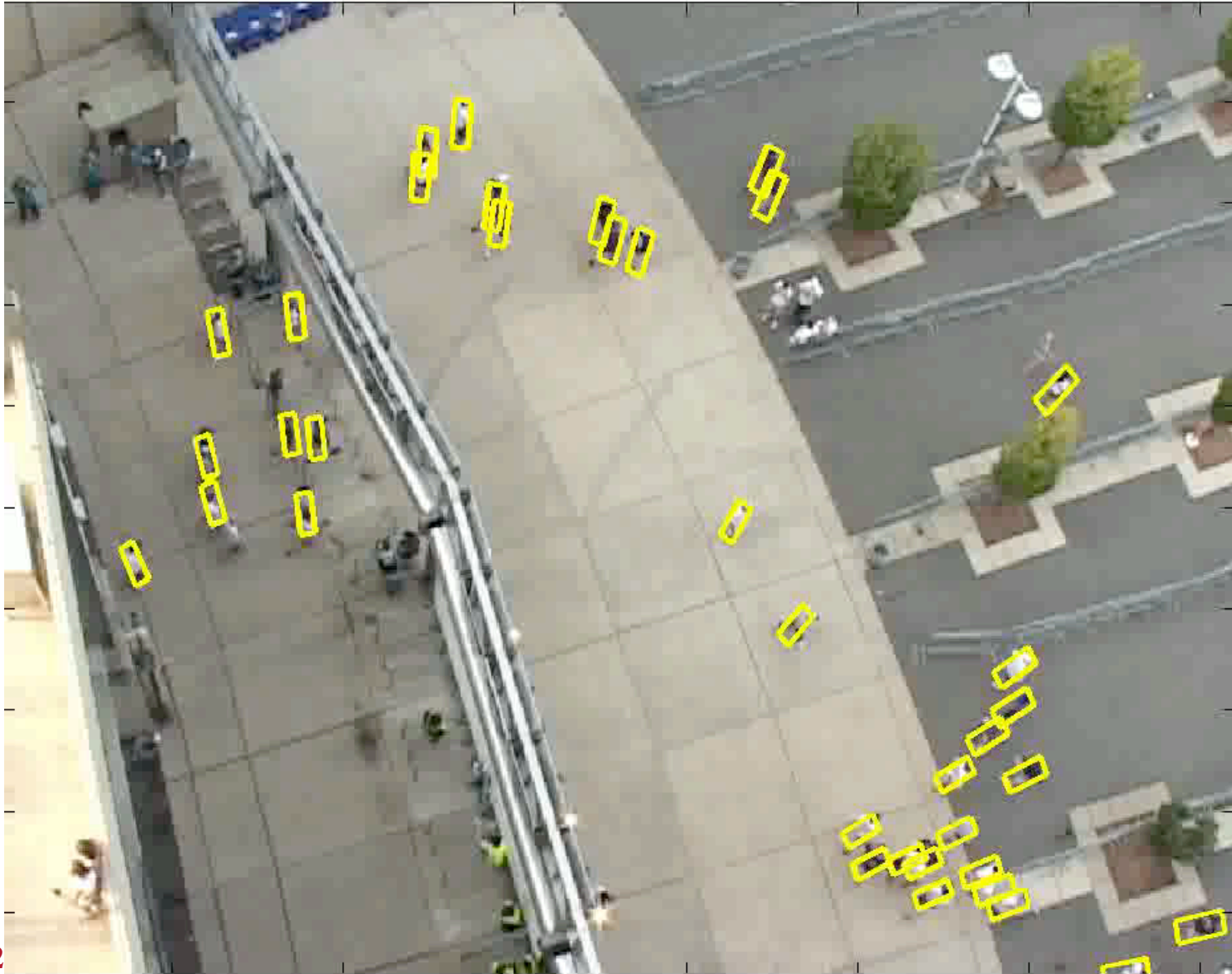
Example: Nov 22, Curtin Road

count 94

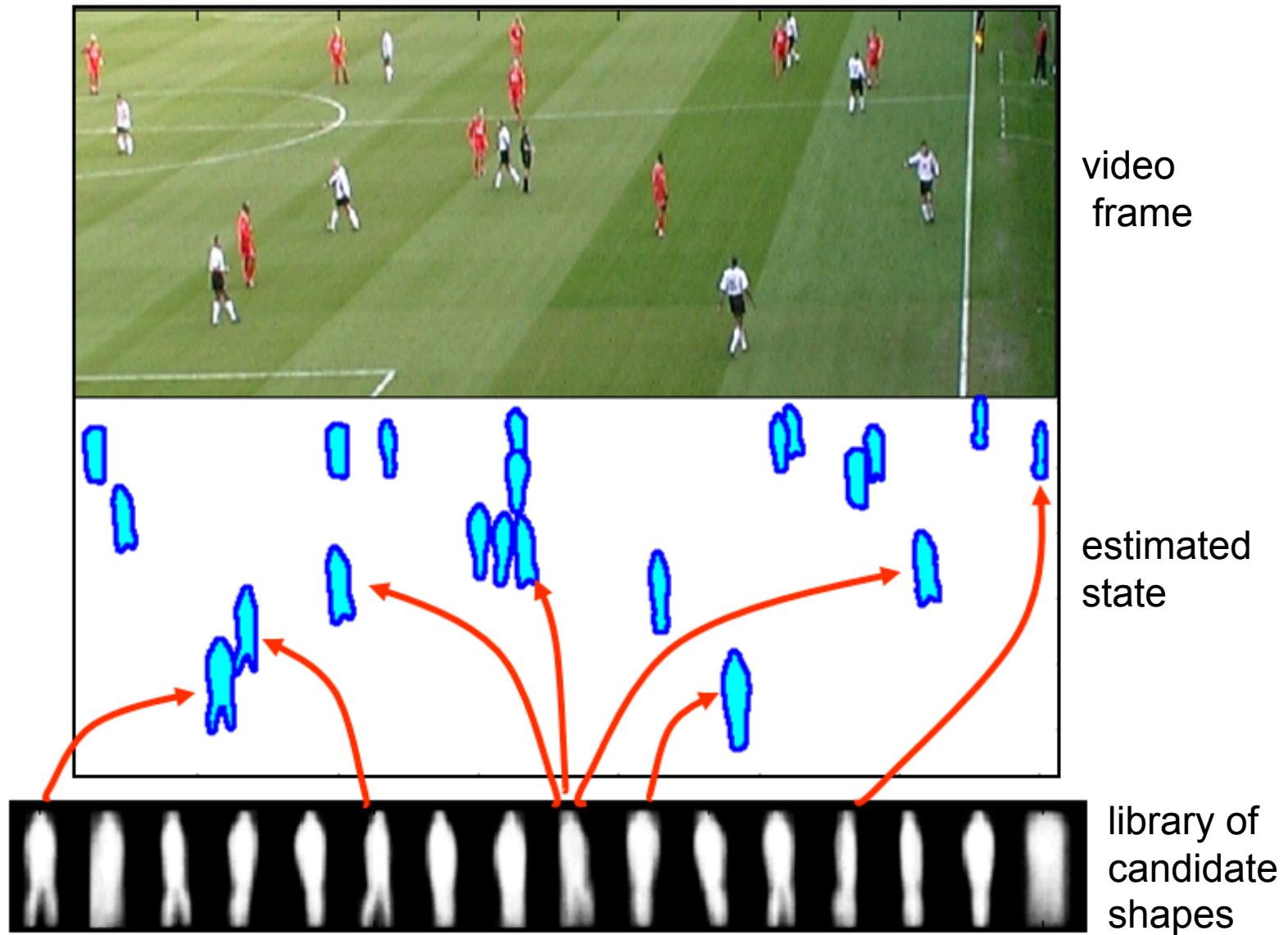


Example: Sep 6, Gate A

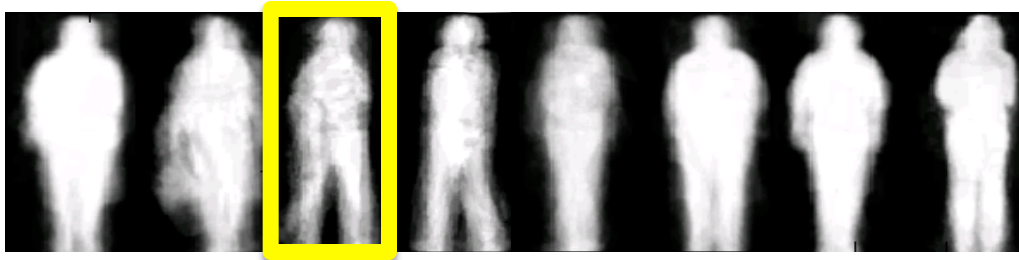
count 37



Adding Shape to the Estimation



Intrinsic vs Extrinsic Shape




Intrinsic shapes:
(silhouettes, aligned
and normalized)

Extrinsic shape
(how bounding box of
silhouette maps into
the image)



Revised Prior Model

$$\pi(\mathbf{o}_i) = \pi(p_i) \pi(w_i, h_i, \theta_i | p_i) \pi(s_i)$$


Prior for
object i

Prior on extrinsic shape
(location + bounding box
height, width and orientation)

Prior on intrinsic
shape selection

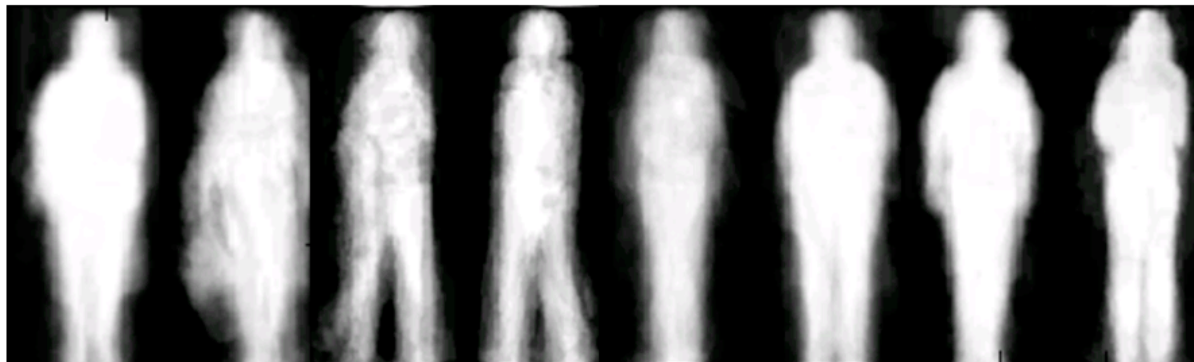
Learning Intrinsic Shapes

Silhouette shape represented as a Bernoulli mixture model

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\pi}) = \sum_{k=1}^K \pi_k p(x|\boldsymbol{\mu}_k)$$



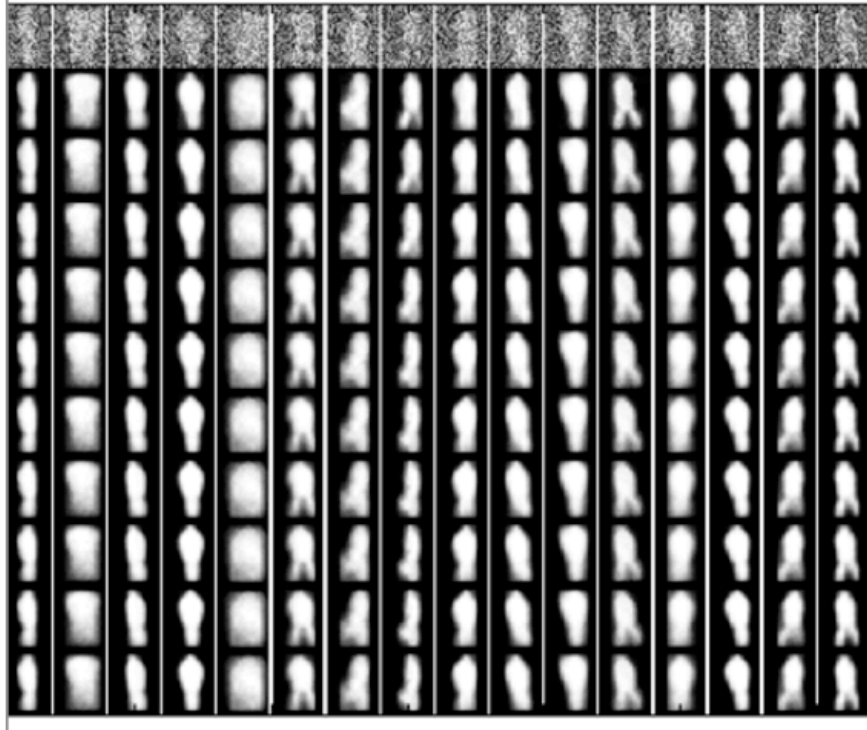
Training shapes
(foreground masks)



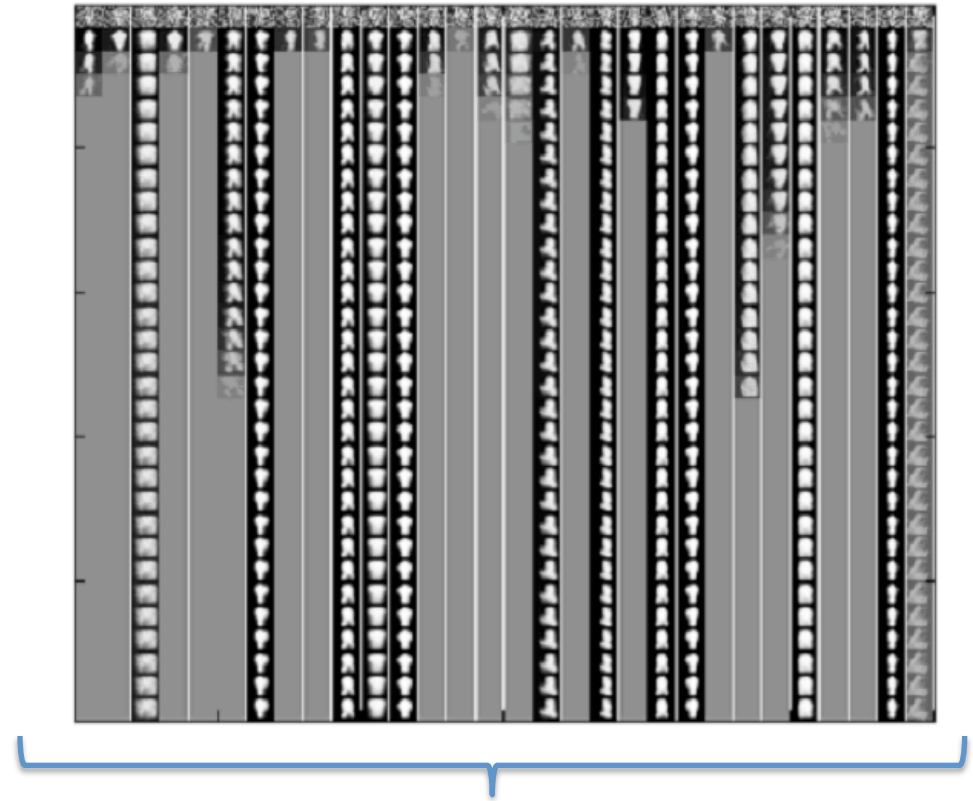
Learned Bernoulli
mixture model
("soft" silhouettes)

Learning Intrinsic Shapes

“standard” EM



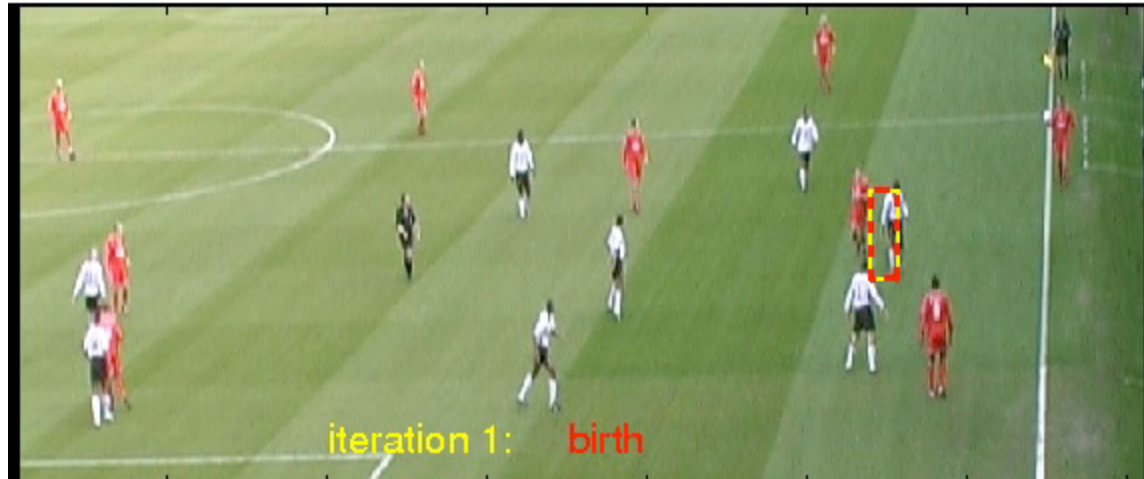
Bayesian EM with Dirichlet prior



Provides automatic selection of number of mixture components

Adding Shape to the Estimation

MCMC iterations



movie

MCMC proposes changes to current configuration

- add/remove a person
- shift location of person
- change their shape**

Robert Collins
Penn State

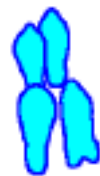
VSPETS Soccer Dataset

Evaluation run on every 10th frame
green contours: true positives
red boxes: false negatives
pink contours: false positives



movie

detailed view



SI

count=4

count=4

count=4

count=4

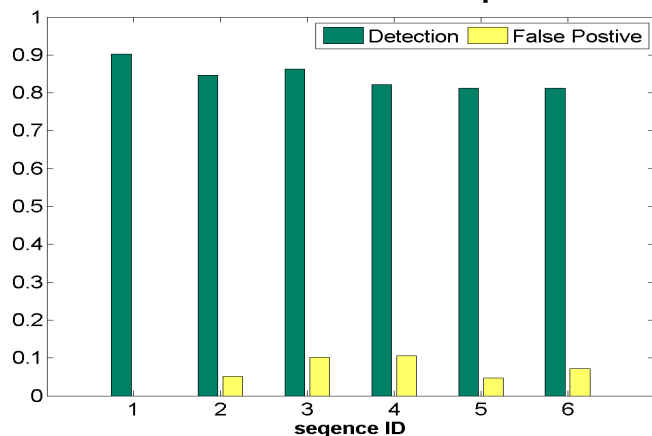
count=3

count=4

count=4

Caviar Dataset

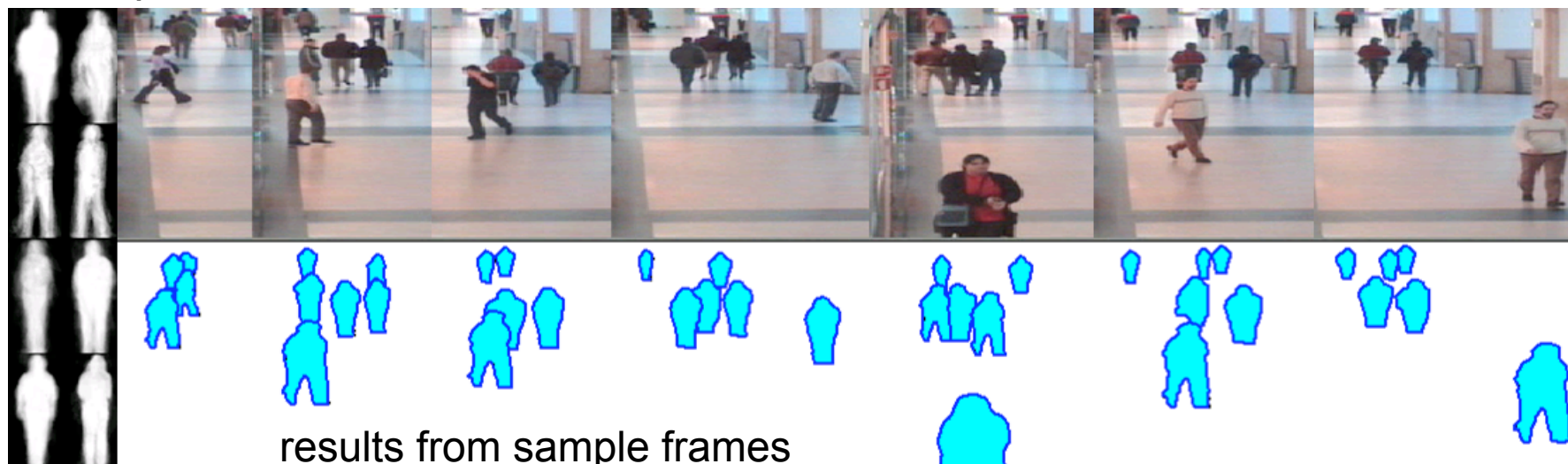
evaluation on six sequences



Dataset	Total # People	Detection Rate	False Positive Rate
CAVIAR	1258	.84	.06
SOCCER	3728	.92	.02

CAVIAR sequence #1
average detection rate: 0.90
average false positive rate: 0.00

Learned shapes



results from sample frames

Contributions

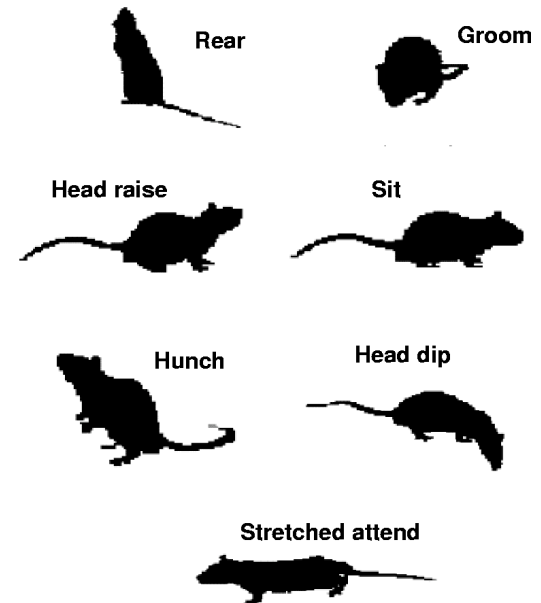
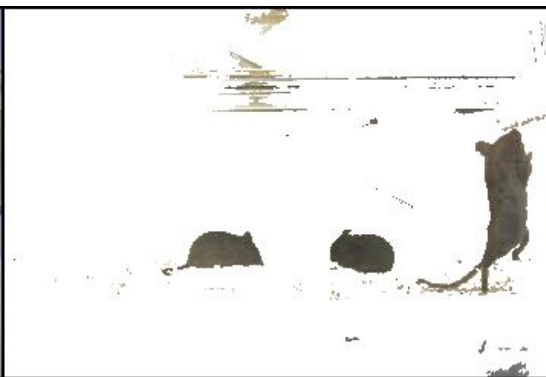
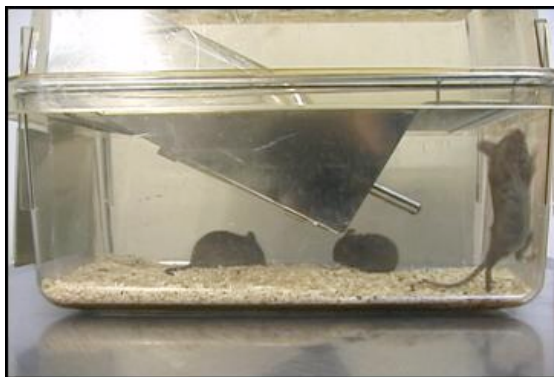
- We introduce a Marked Point Process framework for detecting people in crowds
- Conditional mark process models known correlations between bounding box size/orientation and image location
- Extrinsic and intrinsic shape models are learned automatically from training data
- Bayesian EM is used to automatically determine the number of components in the mixture of Bernoulli model for intrinsic shape

Research Directions

Idea: might be good to try other human shapes (activities)



or other animals/objects.



Overview

Part 1: Change/Motion Detection

Basics: BG subtraction; Frame Difference

Classification-based methods

Part 2: From Pixels to 2D Blobs

Detection via RJMCMC

Classifier Grids

Part 3: Data Association

Linear Assignment Problem

Murty K-best; PDAF; JPDAF

Part 4: Persistent Tracking

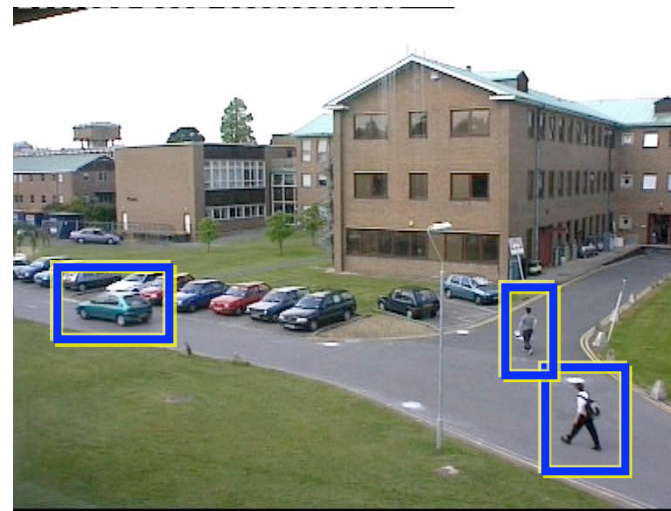
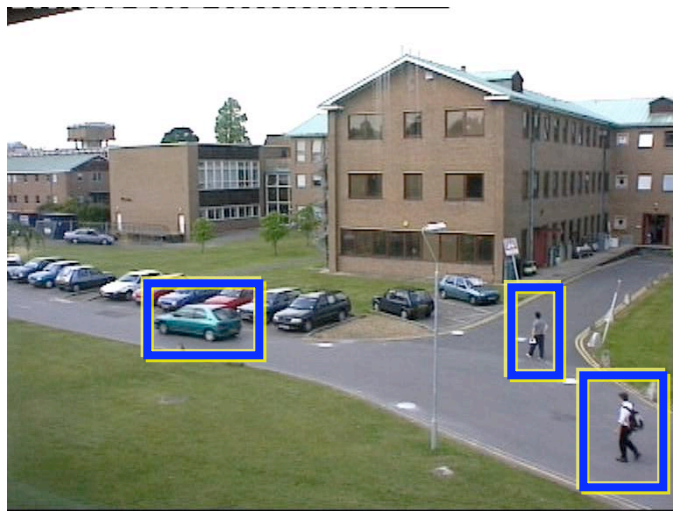
Adaptive Tracking

Tracking as Classification

Data Association Scenario

Two-frame Matching (Correspondence Problem)

Match up detected blobs across video frames

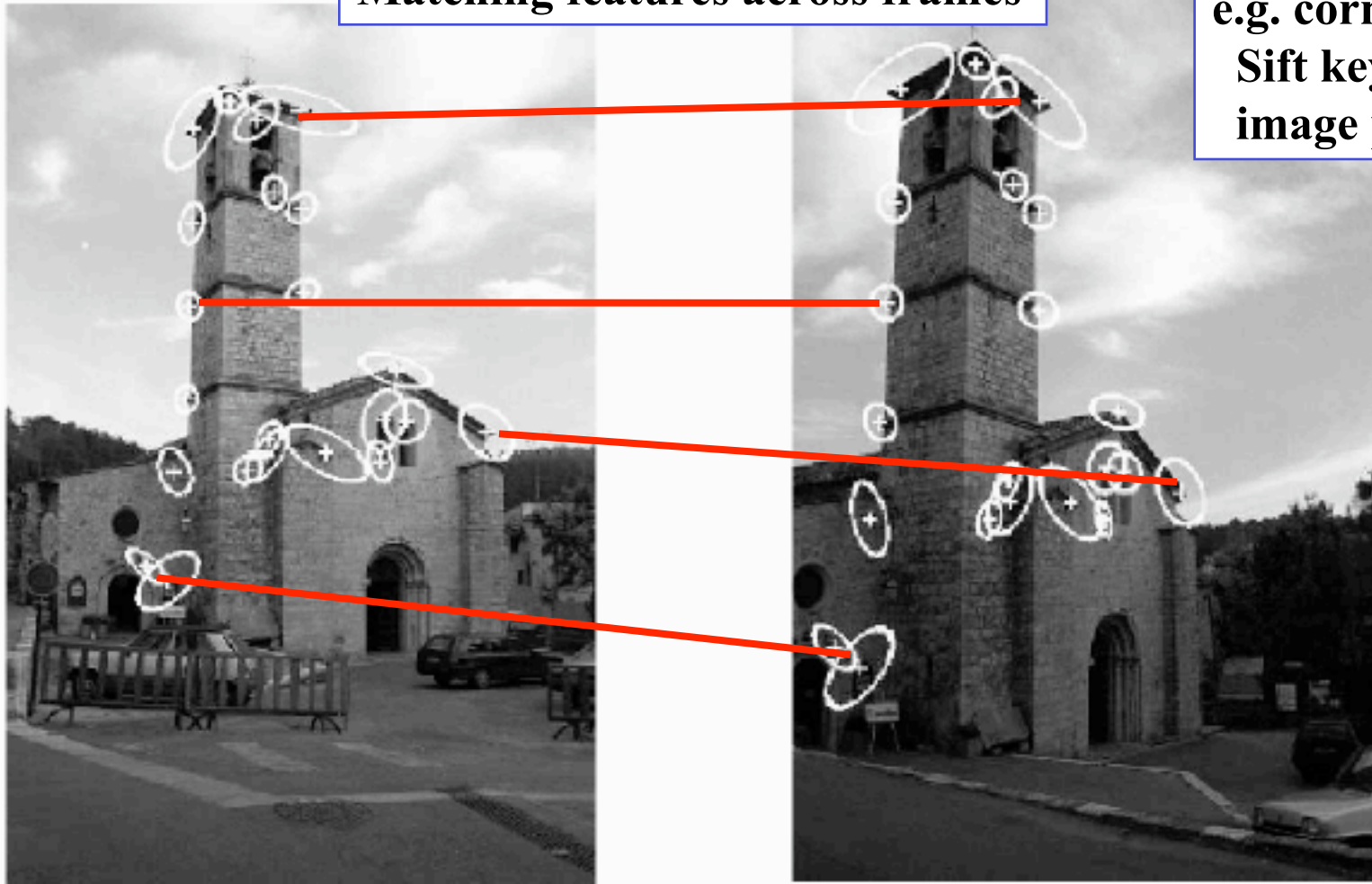


Data Association Scenario

Two-frame Matching (Correspondence Problem)

Matching features across frames

e.g. corners,
Sift keys,
image patches



Data Association Scenarios

In general, data association of blobs in video is easier than general correspondence problem associating features across two image frames.

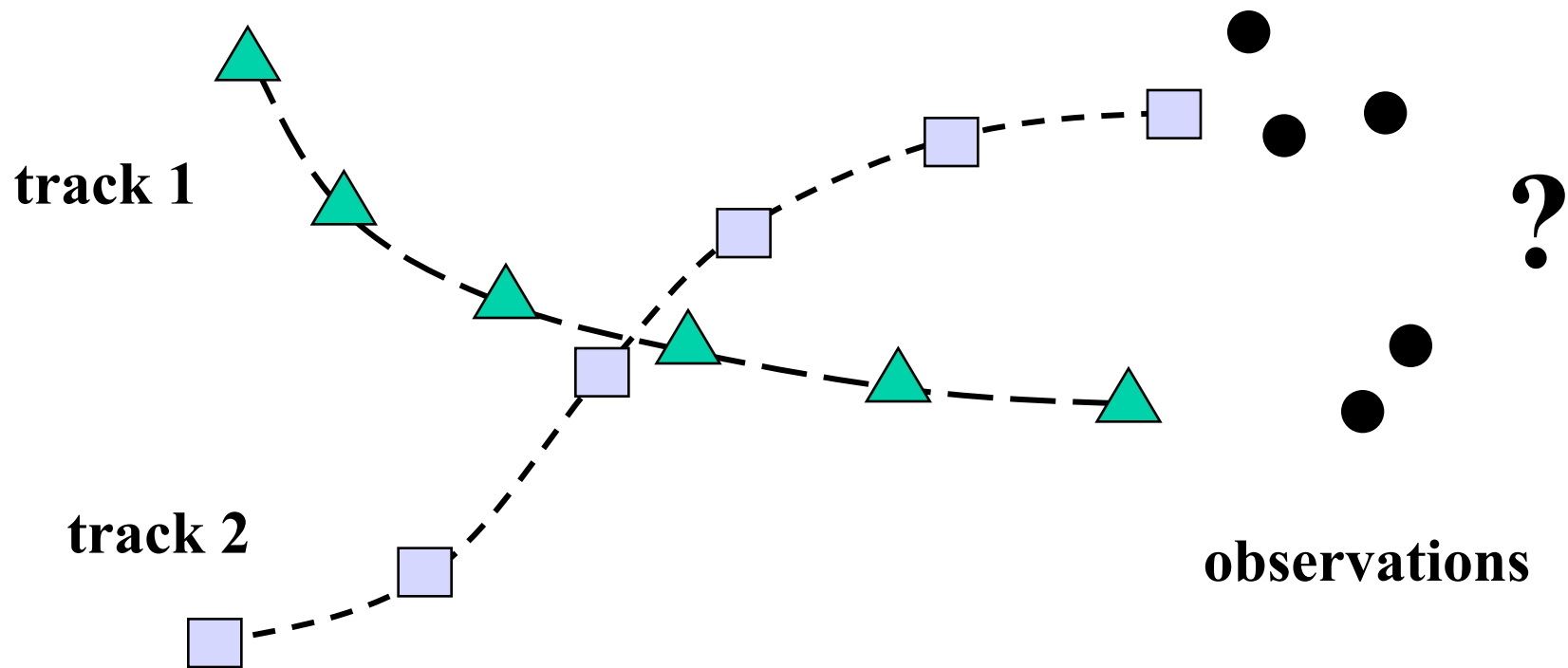
Why? Because of temporal coherence of object motion, which spatially constrains candidate matches.

Outline

- Track Prediction and Gating
- Global Nearest Neighbor (GNN)
- Linear Assignment Problem
- Murthy's k-best Assignments Algorithm
- Probabilistic Data Association (PDAF)
- Joint Probabilistic Data Assoc (JPDAF)
- Multi-Hypothesis Tracking (MHT)

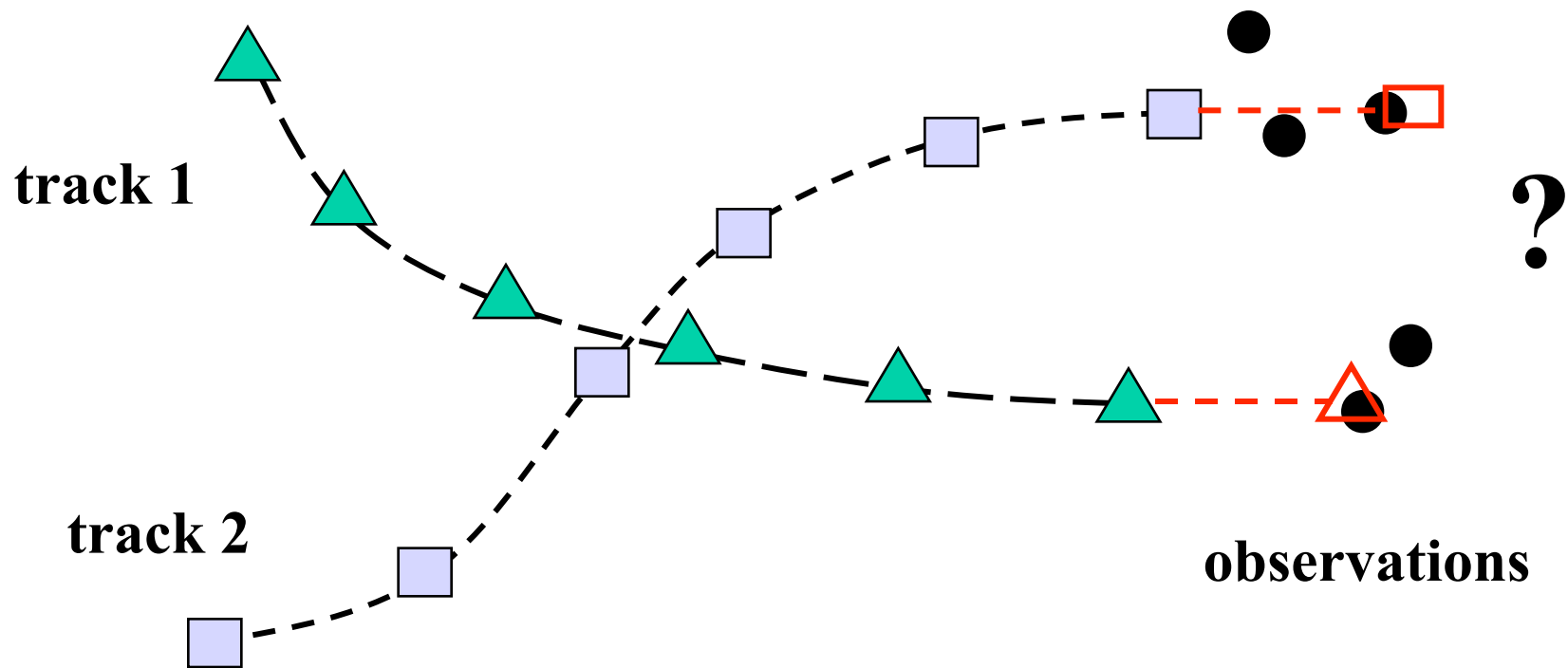
Track Matching

How do we match observations in a new frame to a set of tracked trajectories?



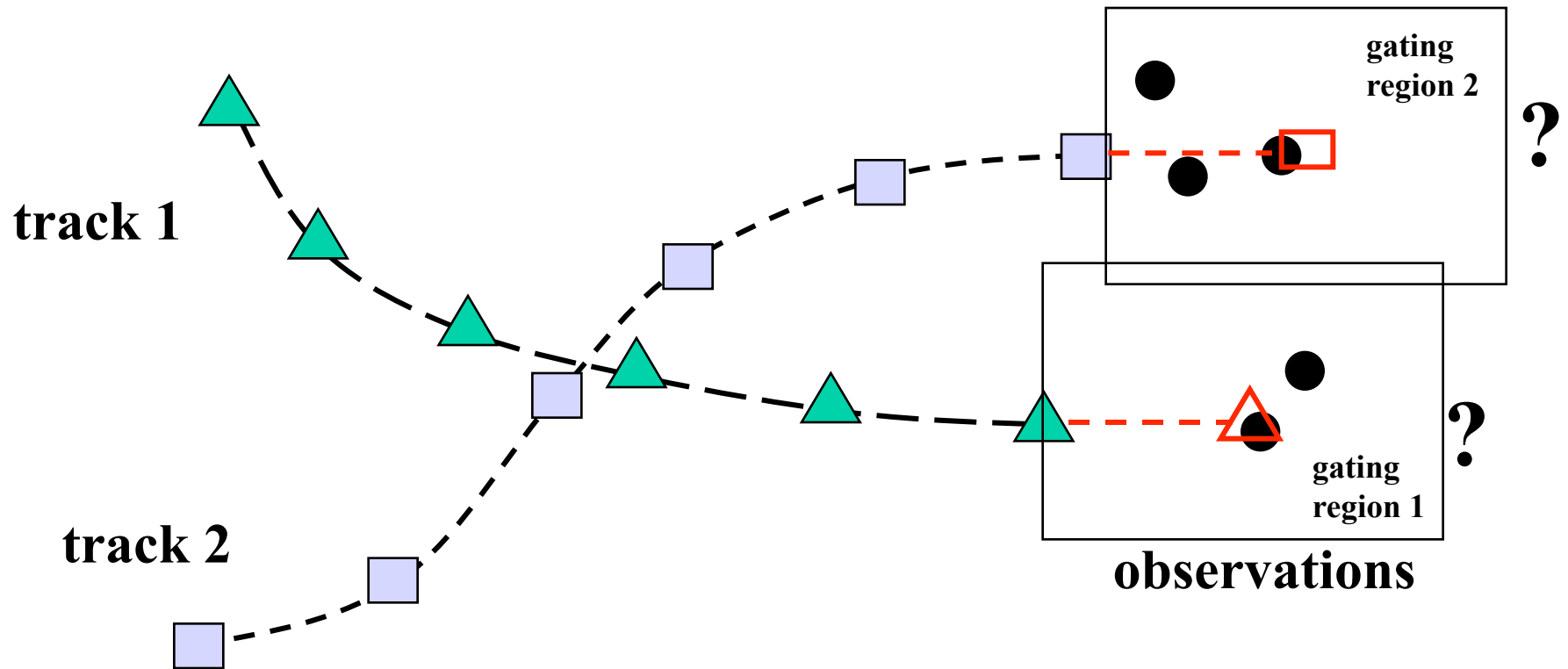
Track Matching

First, predict next target position along each track.



Track Matching

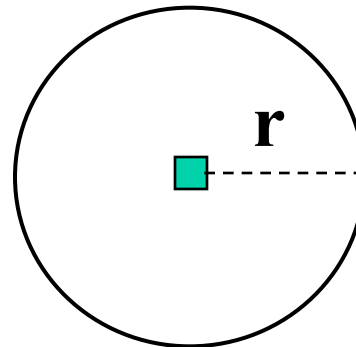
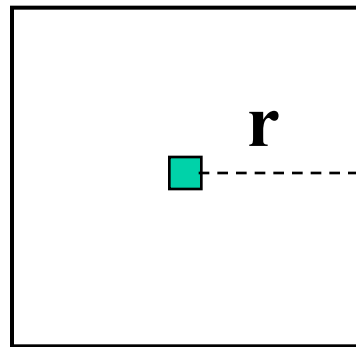
Form a “gating” region around each predicted target location to filter out unlikely matches that are too far away.



Note, this decomposes the full N^2 problem into a sparse set of smaller subproblems.

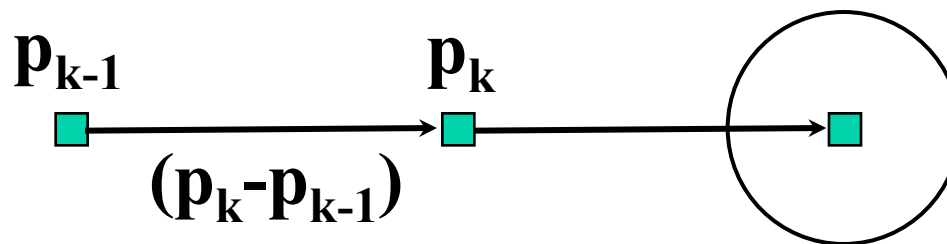
Simple Prediction/Gating

Constant position + bound on maximum interframe motion



constant position
prediction

Three-frame constant velocity prediction

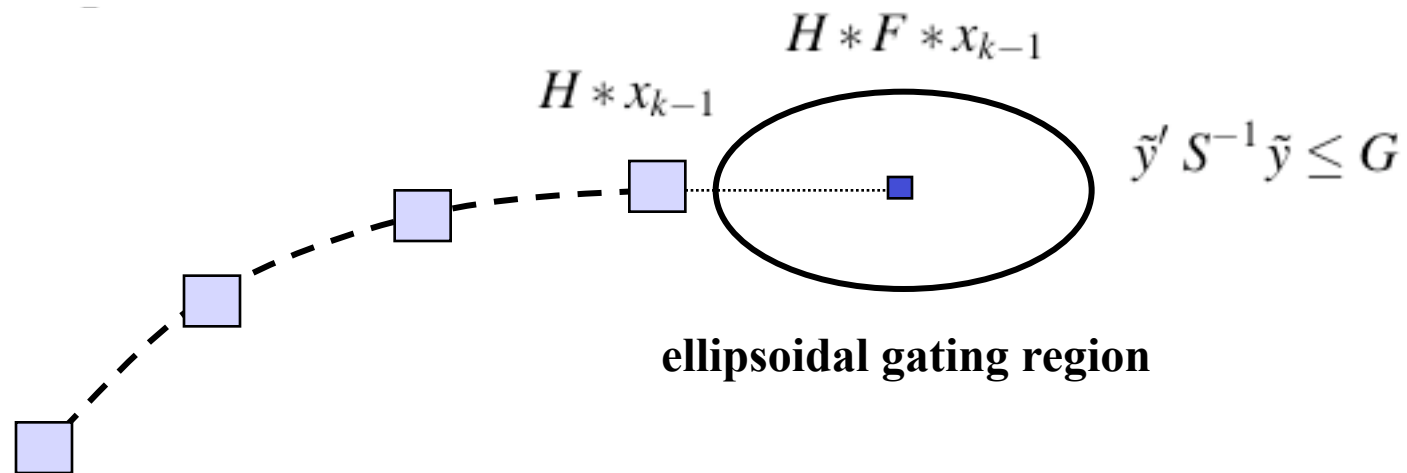


prediction
 $p_k + (p_k - p_{k-1})$

typically, gating
region can be smaller

Kalman Filter Prediction/Gating

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



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

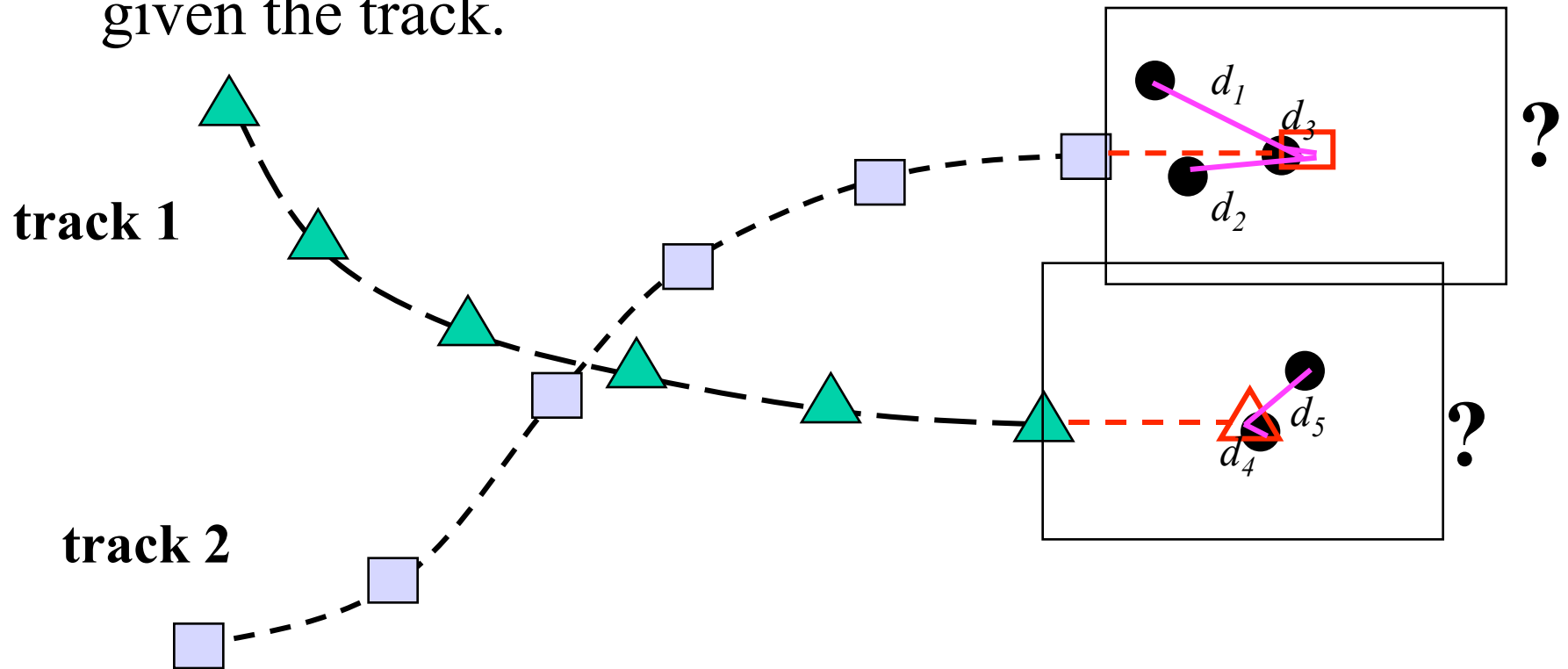
$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad (\text{predicted estimate covariance})$$

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

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

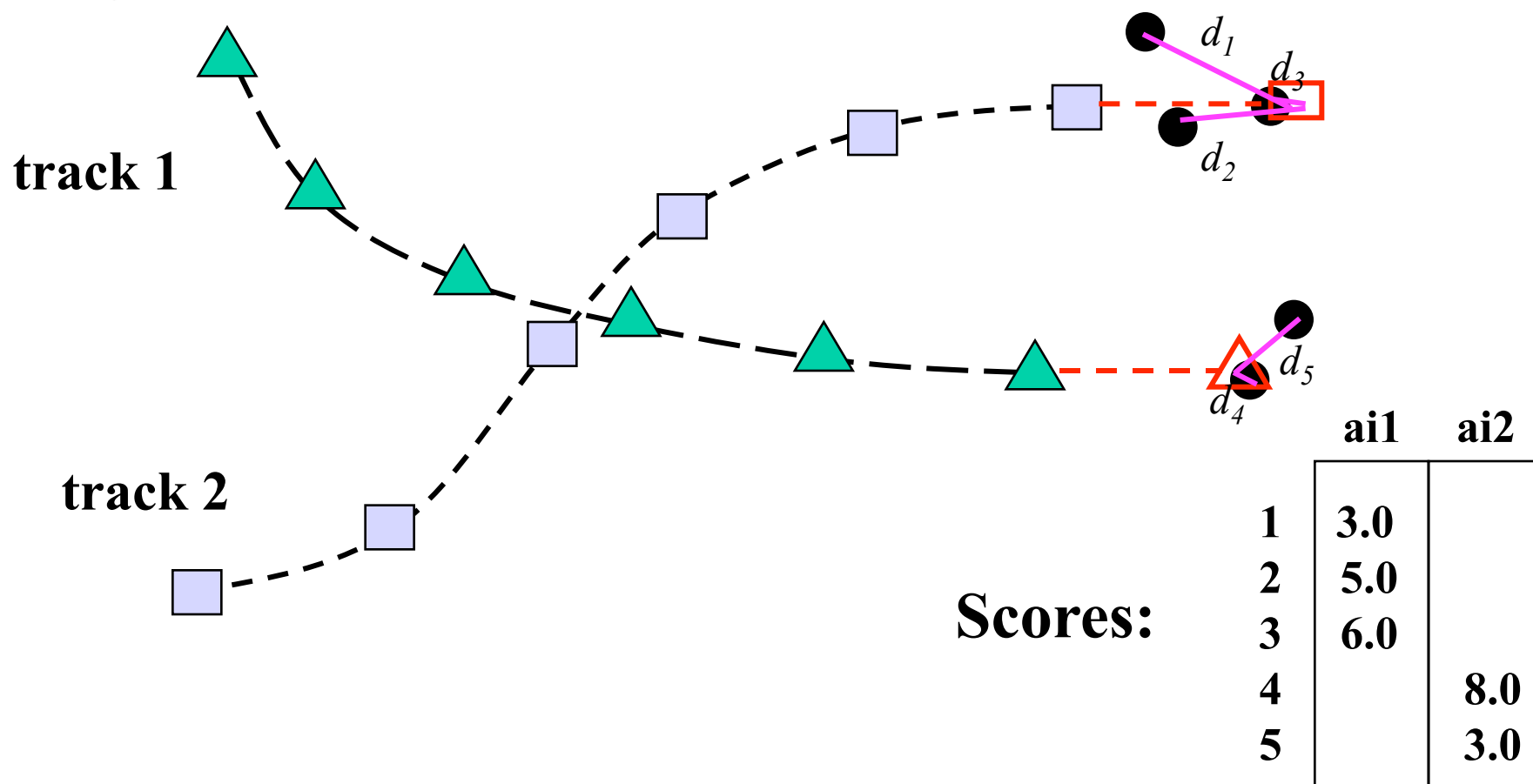
Track Matching

For each candidate match (a track-to-data pairing),
compute match score based on likelihood of the data
given the track.



Track Matching

For each candidate match (a track-to-data pairing), compute match score based on likelihood of the data given the track.

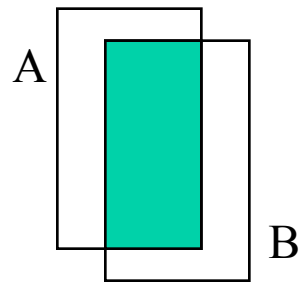


Association Likelihood Score

Determining the correspondence of blobs across frames is based on feature similarity between blobs.

Sample features: location , size / shape, velocity, appearance

For example: location, size and shape similarity can be measured based on bounding box overlap:

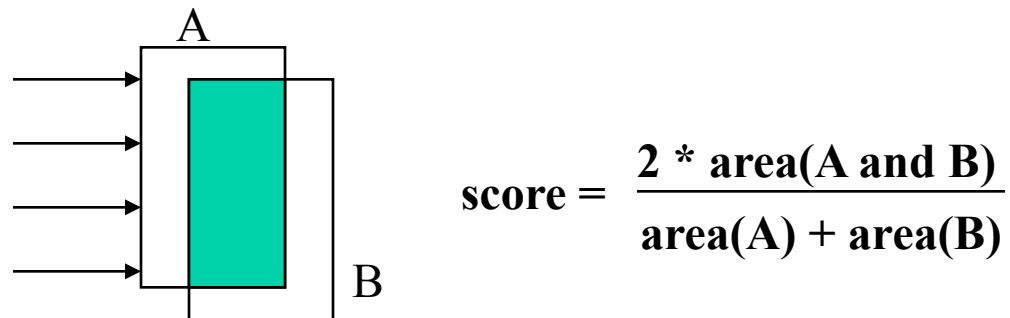
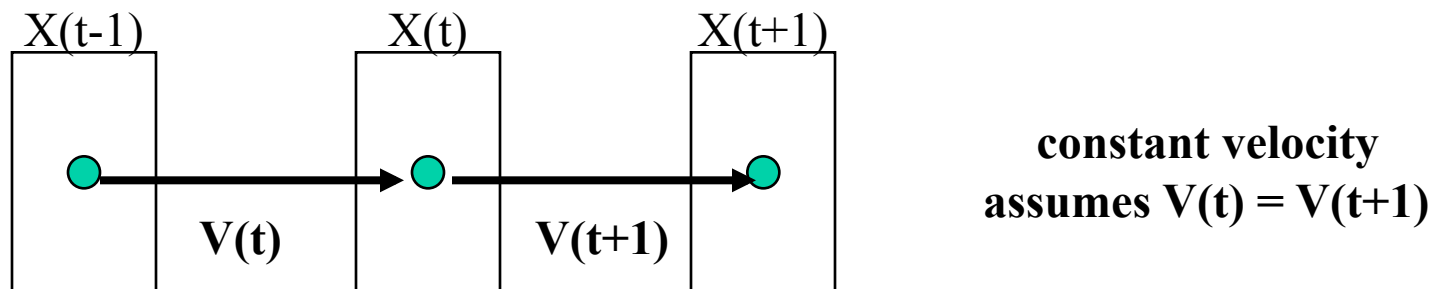


$$\text{score} = \frac{2 * \text{area}(\text{A and B})}{\text{area}(\text{A}) + \text{area}(\text{B})}$$

A = predicted bounding box
B = observed bounding box

Association Likelihood Score

It is common to assume that objects move with constant velocity

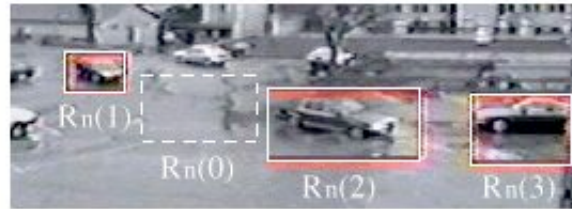


A = bounding box at time t, adjusted by velocity $V(t)$

B = bounding box at time t+1

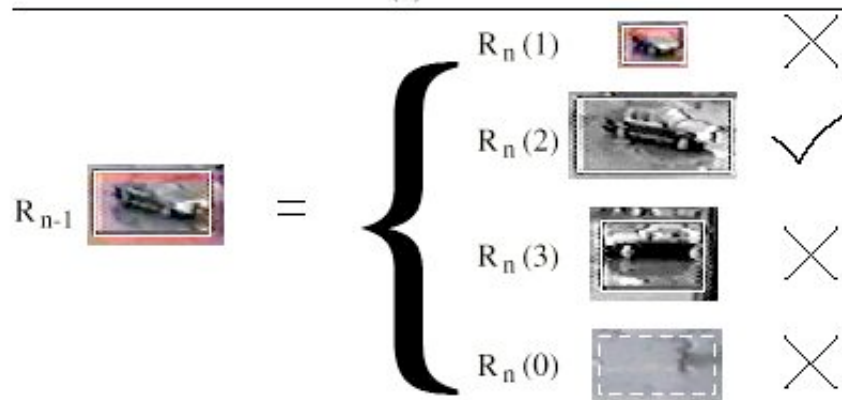
Using Appearance Scores

Correlation of image templates is an obvious choice (between frames)



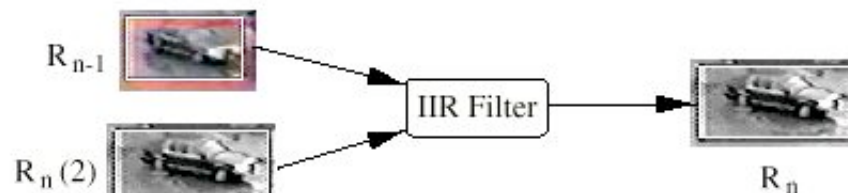
(a)

Extract motion blobs



(b)

For object in previous frame, compute correlation score with all blobs in current frame. Pick one with highest score (suboptimal strategy).

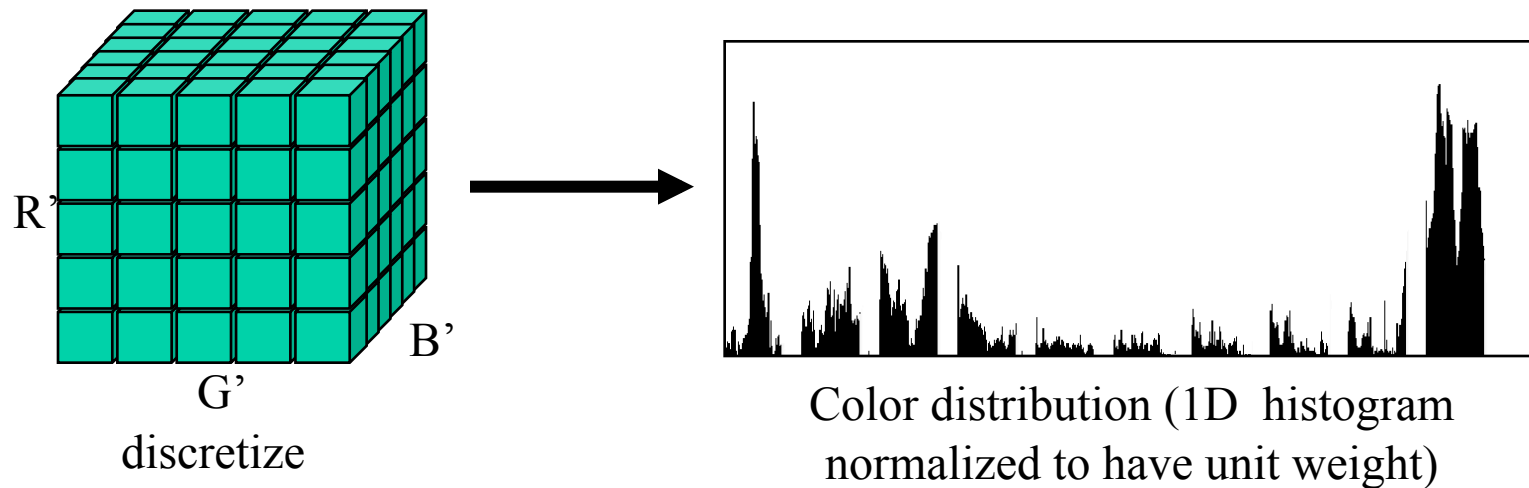


(c)

Update appearance template of blobs

However, cross correlation is computationally expensive.

Appearance via Color Histograms



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

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

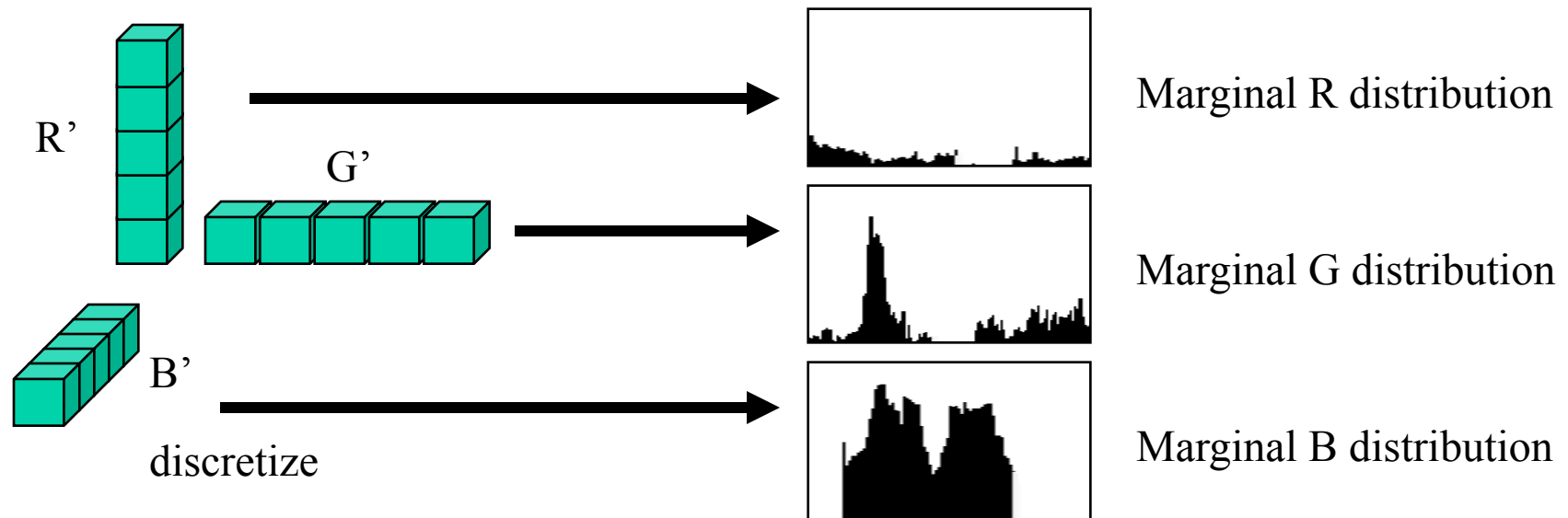
$$B' = B \ll (8 - \text{nbits})$$

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.

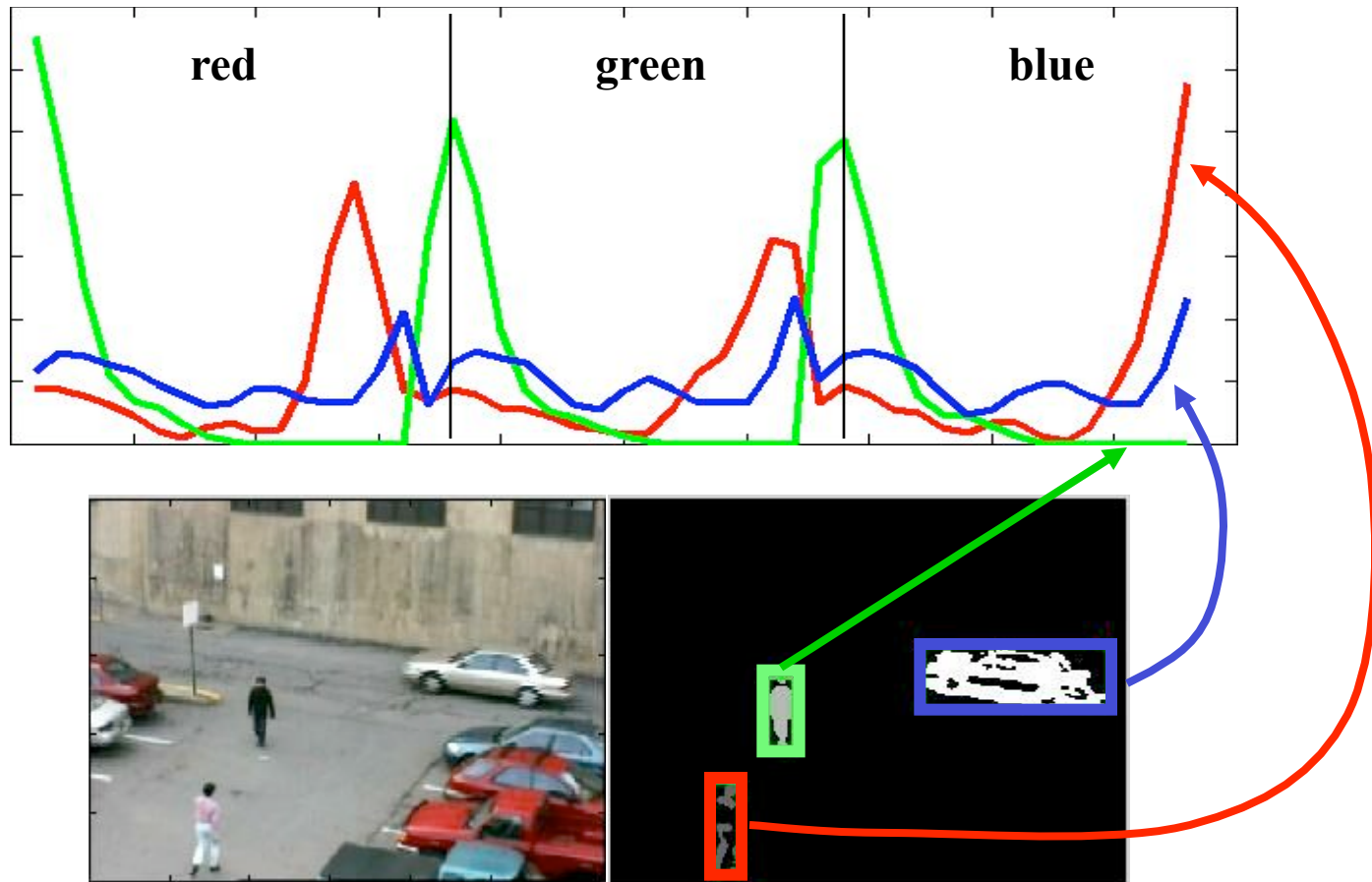


$$\begin{aligned} R' &= R \ll (8 - \text{nbits}) \\ G' &= G \ll (8 - \text{nbits}) \\ B' &= B \ll (8 - \text{nbits}) \end{aligned}$$

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 $3 \cdot 16 = 48$.

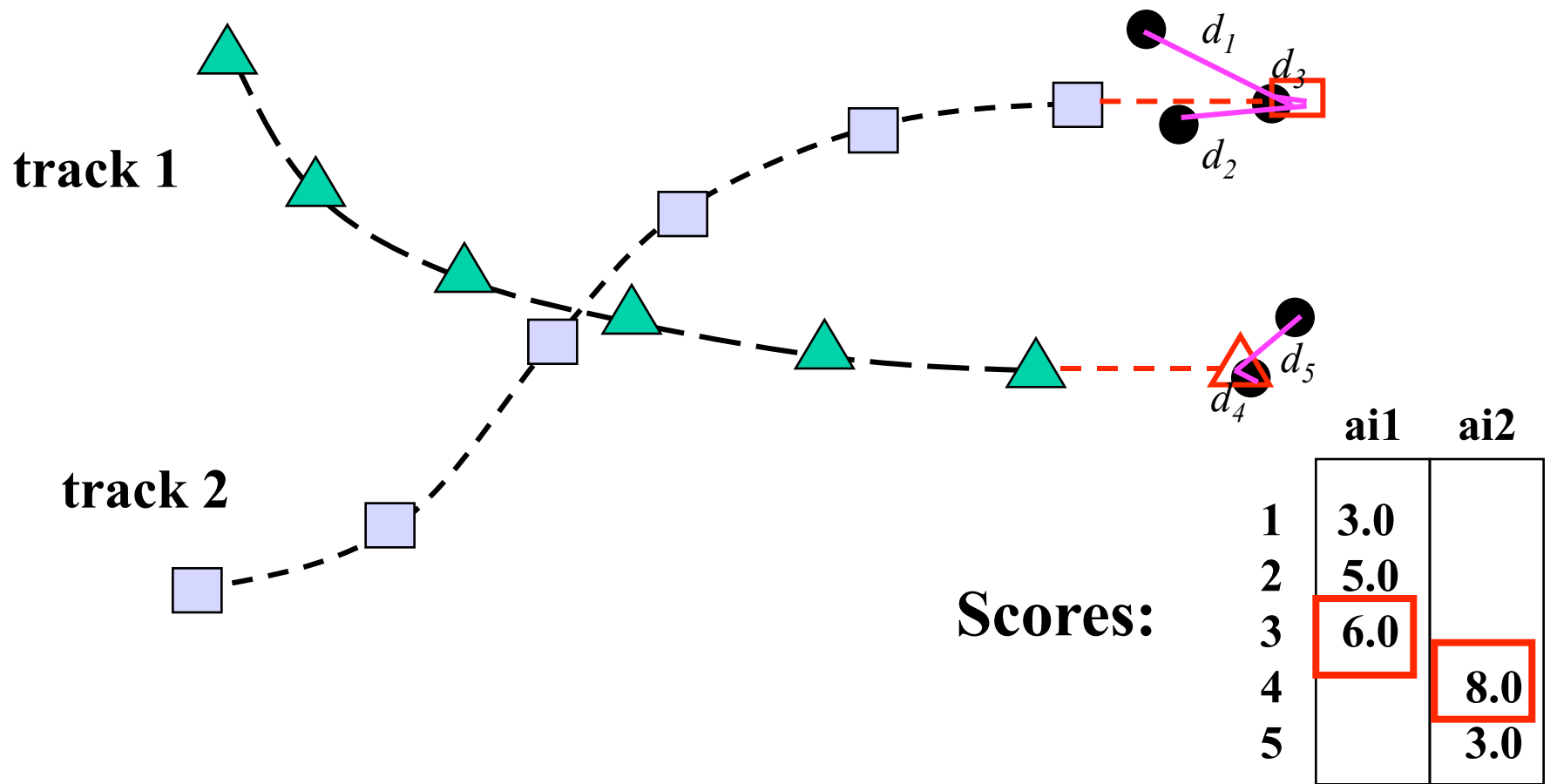
Color Histogram Example



Could measure similarity using chi-square, histogram intersection, EMD, etc.

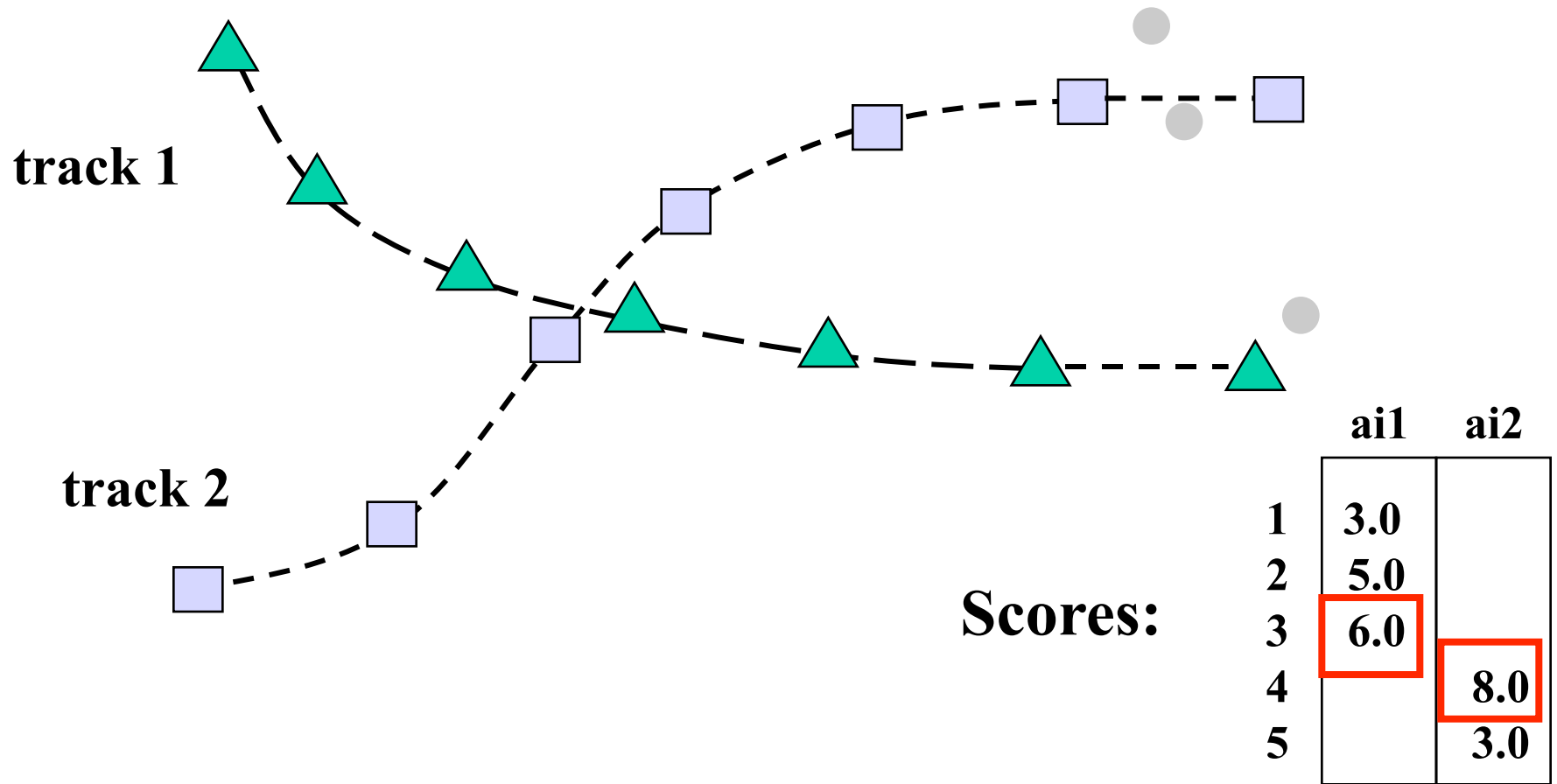
Track Matching

Determine best match and extend the trajectory.



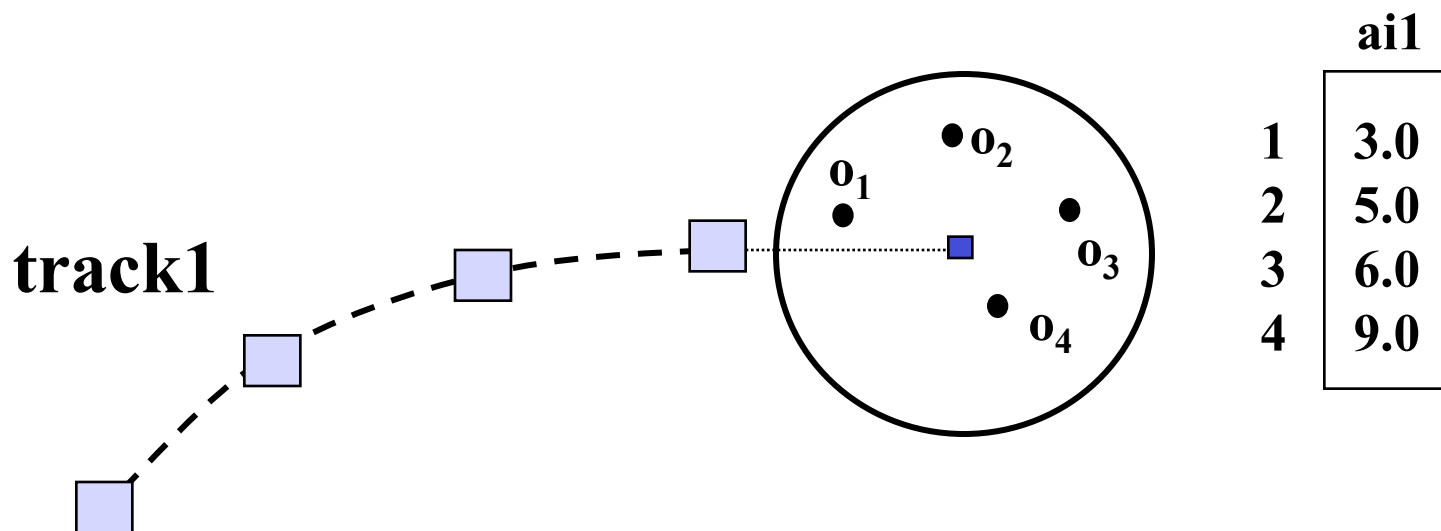
Track Matching

Determine best match and extend the trajectory.
(e.g. using these observations in a Kalman filter update)



Global Nearest Neighbor (GNN)

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

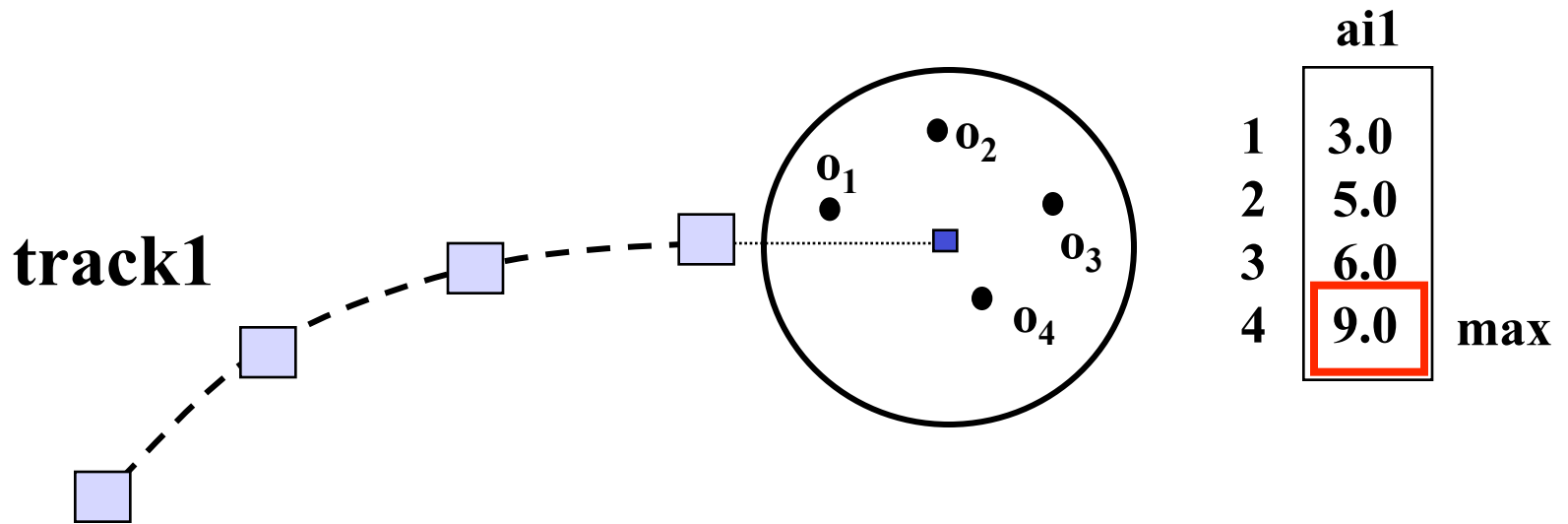


a_{1j} = score for matching observation j to track 1

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.

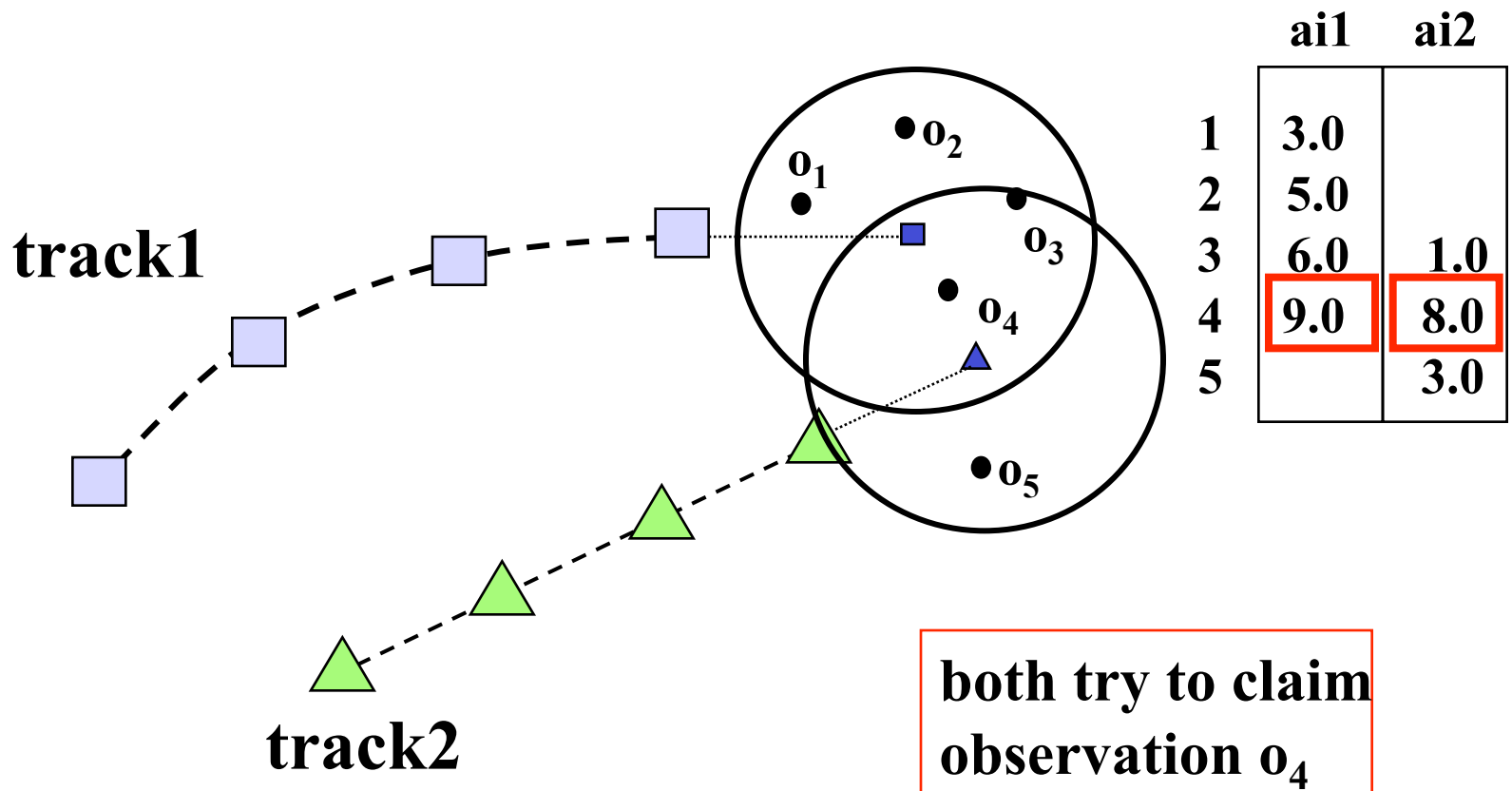


a_{i1} = score for matching observation i to track 1

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

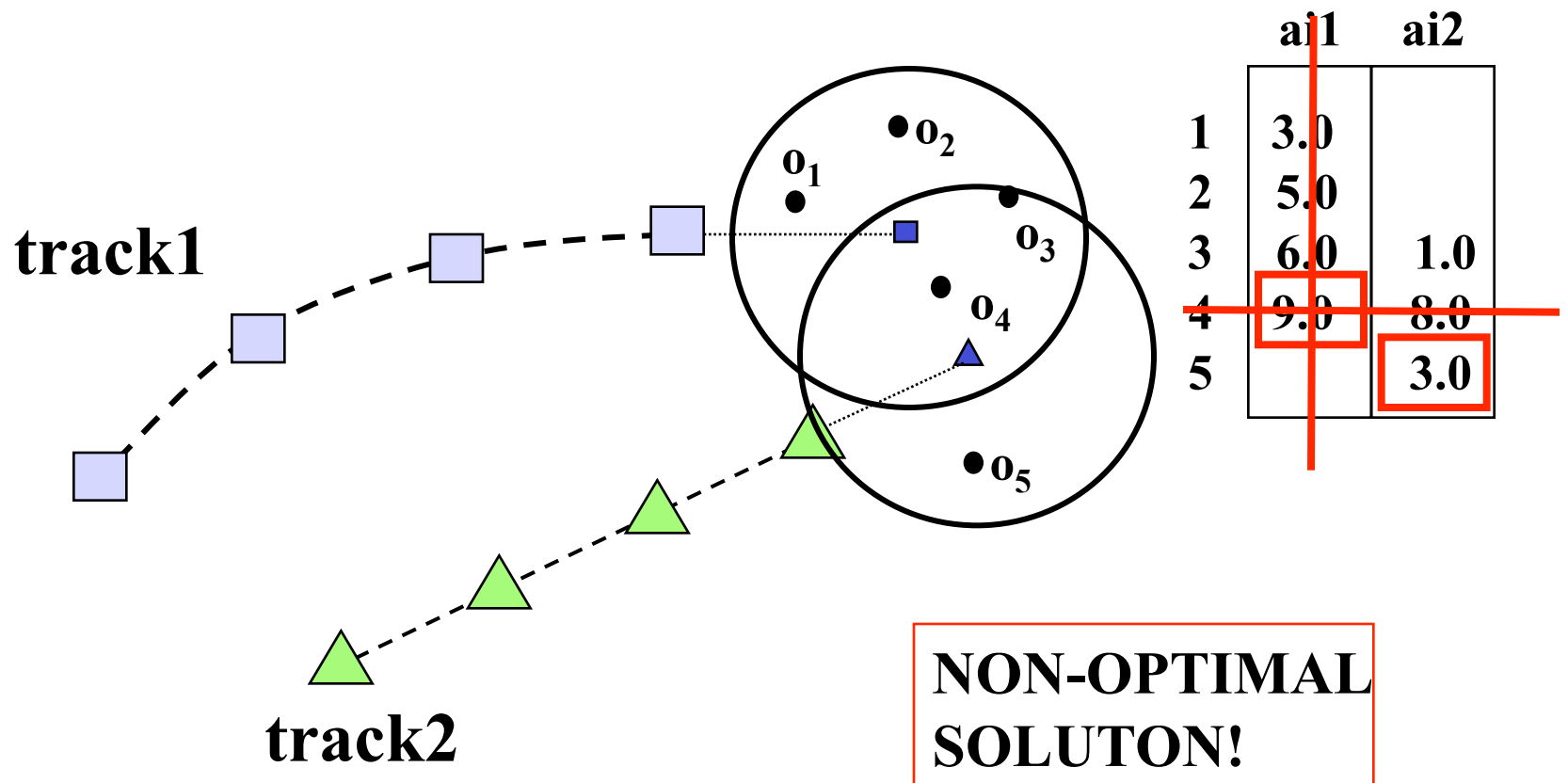
Global Nearest Neighbor (GNN)

Problem: if we do that independently for each track, we 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.



Linear Assignment Problem

We have N objects in previous frame and M objects in current frame. We can build a table of match scores $m(i,j)$ for $i=1\dots N$ and $j=1\dots M$. For now, assume $M=N$.

	1	2	3	4	5
1	0.95	0.76	0.62	0.41	0.06
2	0.23	0.46	0.79	0.94	0.35
3	0.61	0.02	0.92	0.92	0.81
4	0.49	0.82	0.74	0.41	0.01
5	0.89	0.44	0.18	0.89	0.14

problem: choose a 1-1 correspondence that maximizes sum of match scores.

Example:

5x5 matrix of match scores

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

working from left to right, choose one number from each column, making sure you don't choose a number from a row that already has a number chosen in it.

How many ways can we do this?

$$5 \times 4 \times 3 \times 2 \times 1 = 120 \quad (\text{N factorial})$$

Examples

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 2.88

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 2.52

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 4.14

A Greedy Strategy

Choose largest value and mark it

For $i = 1$ to $N-1$

Choose next largest remaining value that isn't in a row/col already marked

End

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 3.77

not as good as our current best guess!

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 4.14

Is this the best we can do?

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:

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

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

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)

Linear Programming

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

$$\begin{aligned} \text{subject to: } & \forall i \sum_{a=1}^A M_{ai} = 1 \\ & \forall a \sum_{i=1}^I M_{ai} = 1 \\ & M_{ai} \in \{0, 1\} \end{aligned}$$

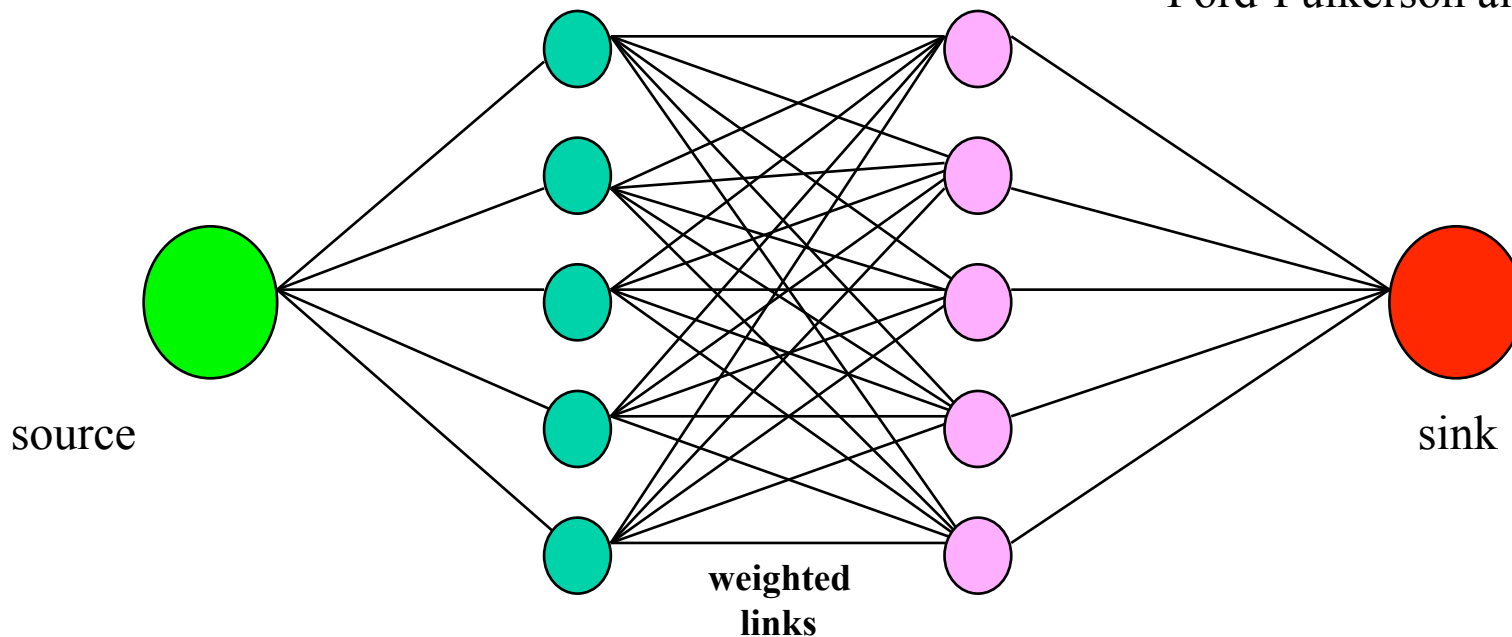
This has the form of a 0-1 integer linear program. Could solve using the simplex method. However, bad (exponential) worst-case complexity (0-1 integer programming is NP-hard)

More Efficient Solution

Can also be viewed as a maximal matching in a weighted bipartite graph, which in turn can be characterized as a max-flow problem.

Polynomial time algorithms available!

Possible solution methods:
Hungarian algorithm
Ford-Fulkerson algorithm



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.^[1]

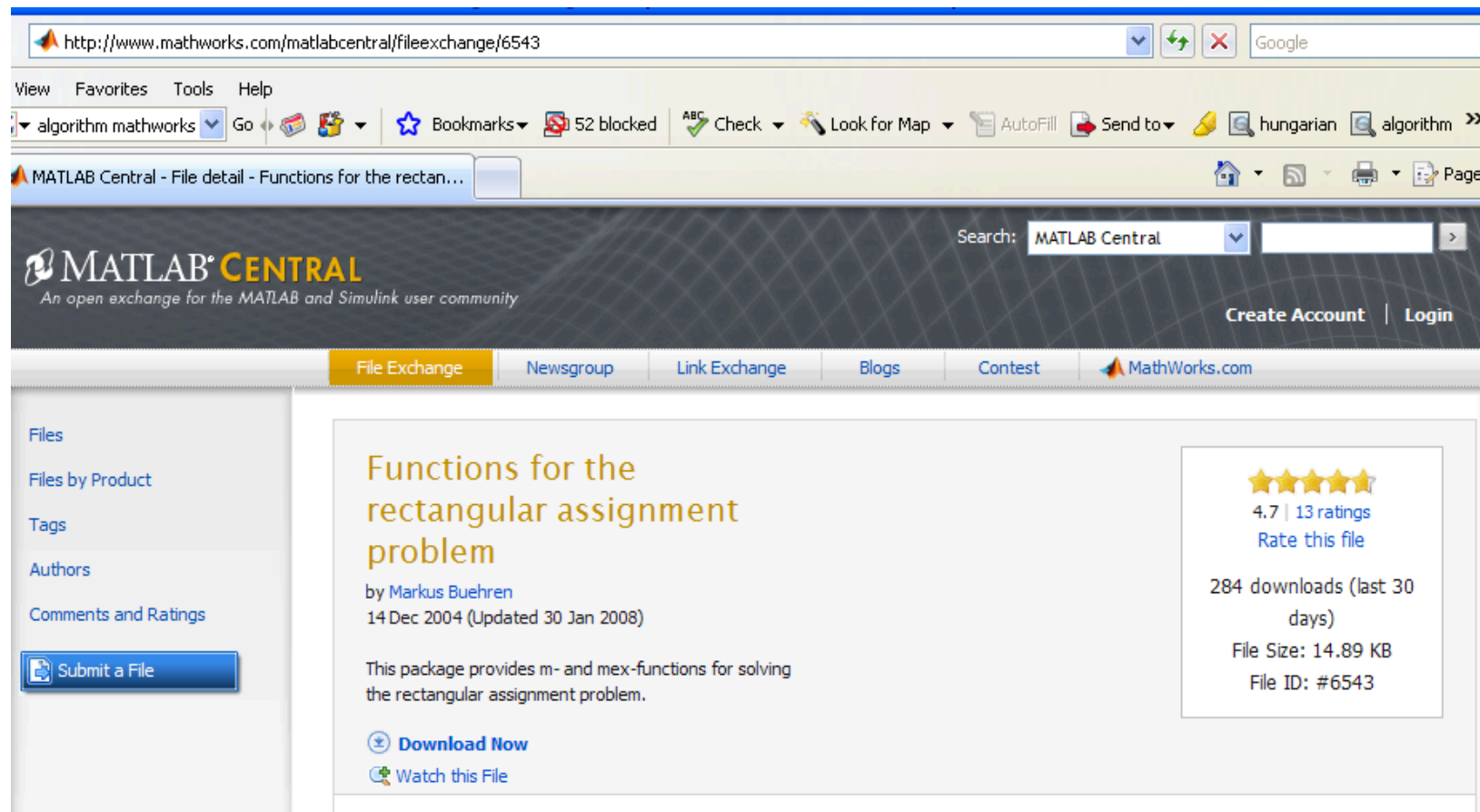
The algorithm developed by Kuhn was largely based on the earlier works of two [Hungarian](#) mathematicians: [Dénes König](#) and [Jenő Egerváry](#). The great advantage of Kuhn's method is that it is strongly [polynomial](#) (see [Computational complexity theory](#) for details). The main innovation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



hence the name

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:



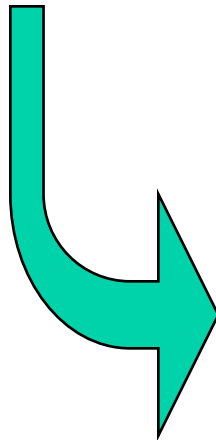
The screenshot shows a web browser window displaying the MATLAB Central file page for 'Functions for the rectangular assignment problem'. The browser address bar shows the URL: <http://www.mathworks.com/matlabcentral/fileexchange/6543>. The page header includes the MATLAB Central logo and navigation links like 'File Exchange', 'Newsgroup', 'Link Exchange', 'Blogs', and 'Contest'. The main content area features the title 'Functions for the rectangular assignment problem' by Markus Buehren, dated 14 Dec 2004 (Updated 30 Jan 2008). A description states: 'This package provides m- and mex-functions for solving the rectangular assignment problem.' On the right, a rating box shows 4.7 stars from 13 ratings, 284 downloads (last 30 days), a file size of 14.89 KB, and a file ID of #6543. A 'Download Now' button is visible at the bottom of the main content area.

If Square Matrix is Required...

	<i>track1</i>	<i>track2</i>	
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



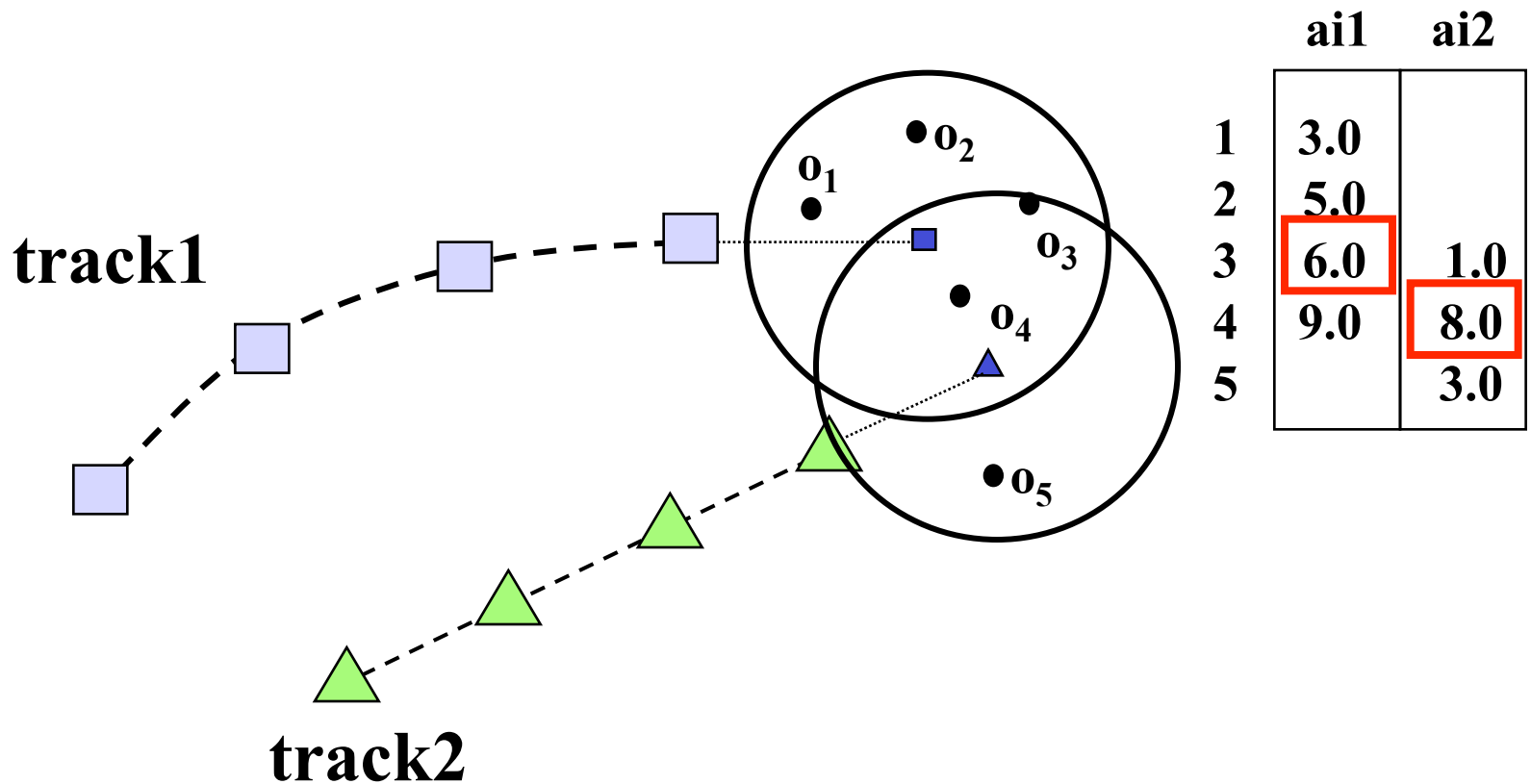
	<i>track1</i>	<i>track2</i>	
1	0	0	5x3
2	0	0	
3	1	0	
4	0	1	
5	0	0	

ignore whatever happens in here

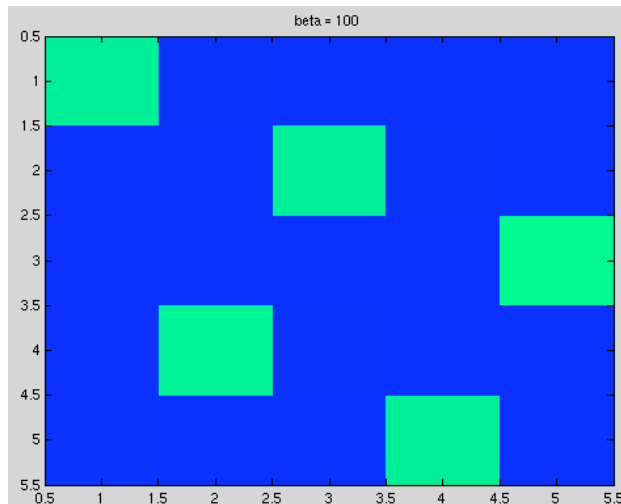
Result From Hungarian Algorithm

Each track is now forced to claim a different observation.

And we get the optimal assignment.



K-Best Assignment



permutation matrix computed
by Hungarian algorithm

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

score: 4.26

So far we know how to find the best assignment (max sum scores). But what if we also want to know the second best? Or maybe the top 10 best assignments?

Murty's K-Best Assignments

General Idea.

Start with best assignment.

Start methodically “tweaking” it by toggling matches in and out of the assignment

Maintain a sorted list of best assignments so far

During each iterative “sweep”, toggle the matches in the next best assignment

The K best assignments are found in decreasing order, one per sweep

1st sweep

<u>0.95</u>	0.76	0.62	0.41	0.06
0.23	0.46	<u>0.79</u>	0.94	0.35
0.61	0.02	0.92	0.92	<u>0.81</u>
0.49	<u>0.82</u>	0.74	0.41	0.01
0.89	0.44	0.18	<u>0.89</u>	0.14

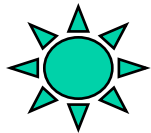
solution: (1,1)(4,2),(2,3),(5,4),(3,5)
constraints: none

constraints

~(1,1)

0.95	<u>0.76</u>	0.62	0.41	0.06
0.23	0.46	0.79	<u>0.94</u>	0.35
0.61	0.02	0.92	0.92	<u>0.81</u>
0.49	0.82	<u>0.74</u>	0.41	0.01
<u>0.89</u>	0.44	0.18	0.89	0.14

(5,1)(1,2),(4,3),(2,4),(3,5)
score 4.14



second best solution

(1,1),~(4,2)

<u>0.95</u>	0.76	0.62	0.41	0.06
0.23	0.46	0.79	<u>0.94</u>	0.35
0.61	0.02	0.92	0.92	<u>0.81</u>
0.49	0.82	<u>0.74</u>	0.41	0.01
0.89	<u>0.44</u>	0.18	0.89	0.14

(1,1)(5,2),(4,3),(2,4),(3,5)
score 3.88

(1,1)(4,2),~(2,3)

<u>0.95</u>	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	<u>0.35</u>
0.61	0.02	<u>0.92</u>	0.92	0.81
0.49	<u>0.82</u>	0.74	0.41	0.01
0.89	0.44	0.18	<u>0.89</u>	0.14

(1,1)(4,2),(3,3),(5,4),(2,5)
score 3.93

(1,1)(4,2),(2,3),~(5,4)

<u>0.95</u>	0.76	0.62	0.41	0.06
0.23	0.46	<u>0.79</u>	0.94	0.35
0.61	0.02	0.92	<u>0.92</u>	0.81
0.49	<u>0.82</u>	0.74	0.41	0.01
0.89	0.44	0.18	0.89	<u>0.14</u>

(1,1)(4,2),(2,3),(3,4),(5,5)
score 3.62

2nd sweep

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

solution: (5,1)(1,2),(4,3),(2,4),(3,5)
constraints: ~(1,1)

constraints

~(1,1),~(5,1)

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(4,1)(1,2),(2,3),(5,4),(3,5)
score 3.74

~(1,1),(5,1),~(1,2)

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(4,2),(1,3),(2,4),(3,5)
score 4.08

~(1,1)(5,1),(1,2),~(4,3)

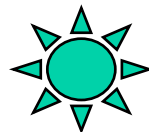
0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(1,2),(2,3),(4,4),(3,5)
score 3.66

~(1,1)(5,1),(1,2),(4,3),~(2,4)

0.95	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	0.35
0.61	0.02	0.92	0.92	0.81
0.49	0.82	0.74	0.41	0.01
0.89	0.44	0.18	0.89	0.14

(5,1)(1,2),(4,3),(3,4),(2,5)
score 3.66



third best solution

1st scan, different order

<u>0.95</u>	0.76	0.62	0.41	0.06
0.23	0.46	<u>0.79</u>	0.94	0.35
0.61	0.02	0.92	0.92	<u>0.81</u>
0.49	<u>0.82</u>	0.74	0.41	0.01
0.89	0.44	0.18	<u>0.89</u>	0.14

solution: (1,1)(4,2),(2,3),(5,4),(3,5)
constraints: none

constraints

~(3,5)

<u>0.95</u>	0.76	0.62	0.41	0.06
0.23	0.46	0.79	0.94	<u>0.35</u>
0.61	0.02	<u>0.92</u>	0.92	0.81
0.49	<u>0.82</u>	0.74	0.41	0.01
0.89	0.44	0.18	<u>0.89</u>	0.14

(1,1)(4,2),(3,3),(5,4),(2,5)
score 3.93

(3,5),~(5,4)

0.95	<u>0.76</u>	0.62	0.41	0.06
0.23	0.46	0.79	<u>0.94</u>	0.35
0.61	0.02	0.92	0.92	<u>0.81</u>
0.49	0.82	<u>0.74</u>	0.41	0.01
<u>0.89</u>	0.44	0.18	0.89	0.14

(5,1)(1,2),(4,3),(2,4),(3,5)
score 4.14

(3,5)(5,4),~(2,3)

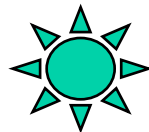
<u>0.95</u>	0.76	0.62	0.41	0.06
0.23	<u>0.46</u>	0.79	0.94	0.35
0.61	0.02	0.92	0.92	<u>0.81</u>
0.49	0.82	<u>0.74</u>	0.41	0.01
0.89	0.44	0.18	<u>0.89</u>	0.14

(1,1)(2,2),(4,3),(5,4),(3,5)
score 3.85

(3,5)(5,4),(2,3),~(4,2)

0.95	<u>0.76</u>	0.62	0.41	0.06
0.23	0.46	<u>0.79</u>	0.94	0.35
0.61	0.02	0.92	0.92	<u>0.81</u>
<u>0.49</u>	0.82	0.74	<u>0.41</u>	0.01
0.89	0.44	0.18	0.89	0.14

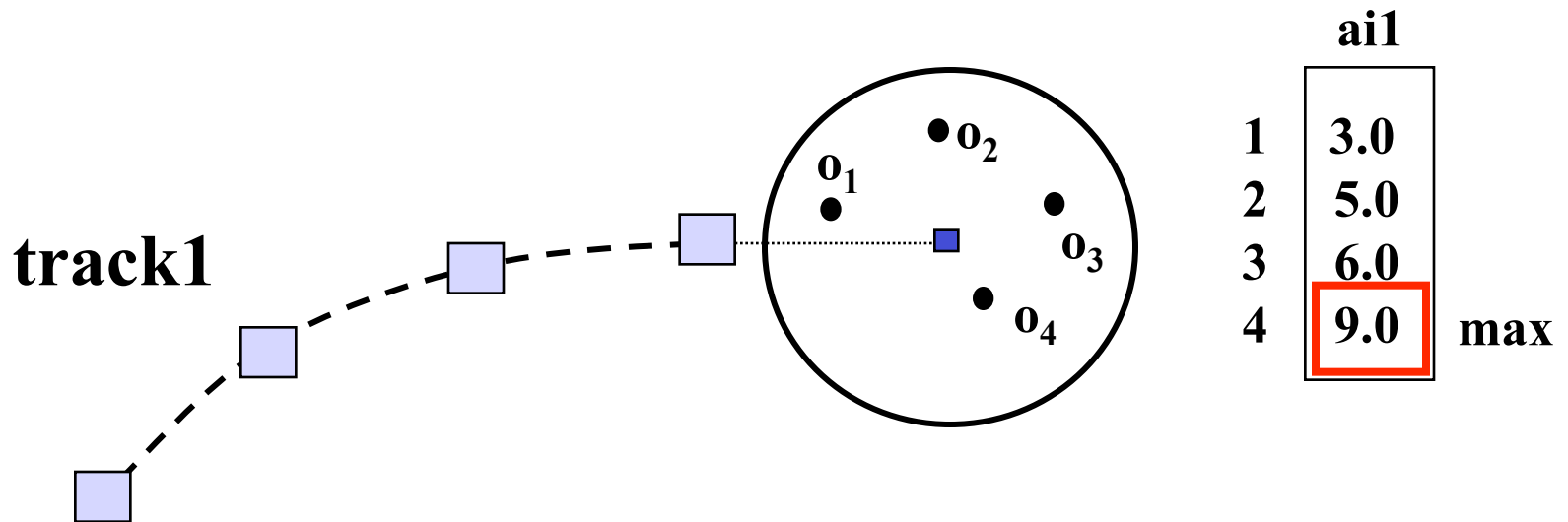
(4,1)(1,2),(2,3),(5,4),(3,5)
score 3.74



second best solution is found again.

Recall: Global Nearest Neighbor

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

PDAF

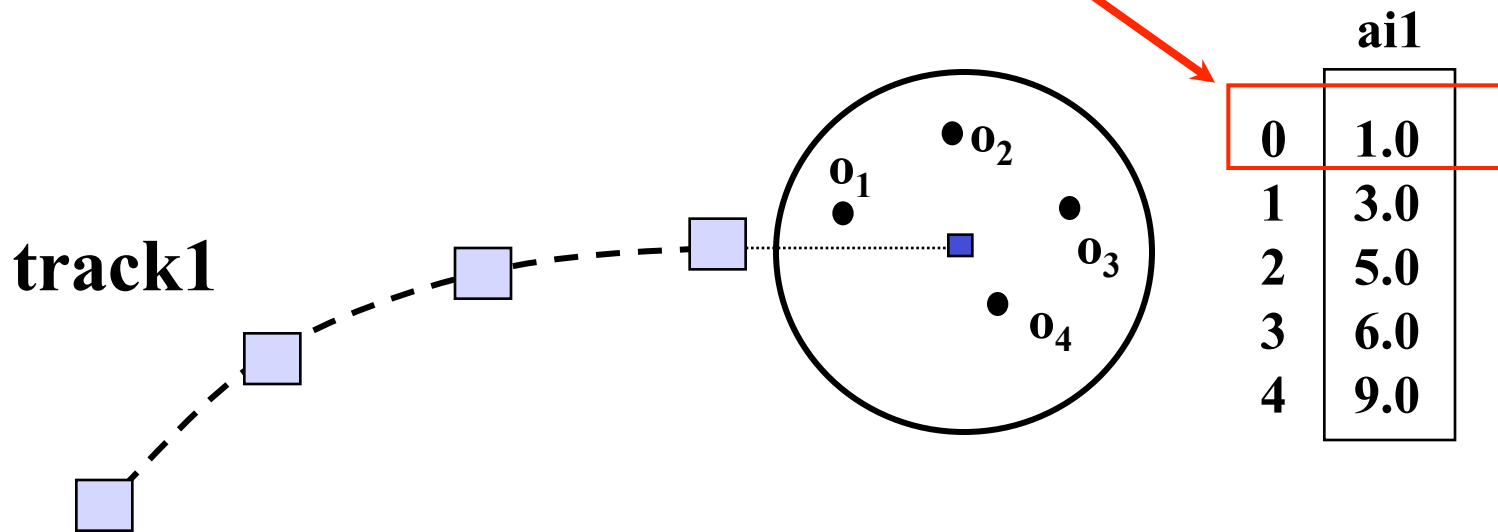
Probabilistic Data Association Filter

Updating single track based on new observations.

General idea: Instead of matching a single best observation to the track, we update based on all observations (in gating window), weighted by their likelihoods.

PDAF

Consider all points in gating window. Also consider the additional possibility that no observations match.

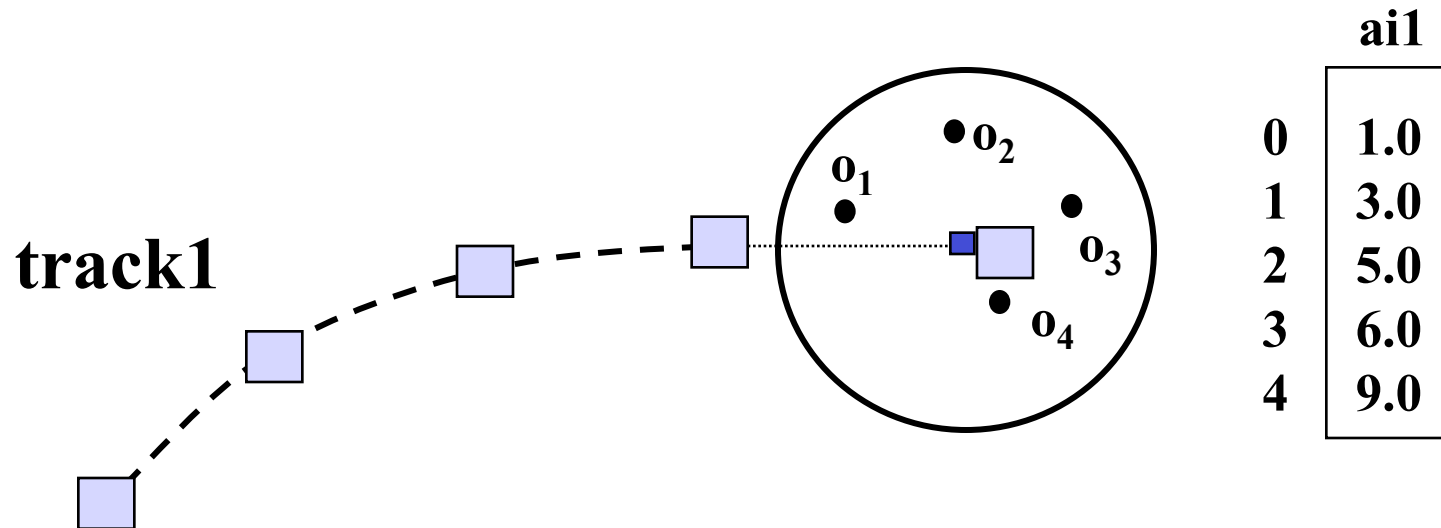


p_{i1} = “probability” of matching observation i to track 1

$$p_{i1} = \frac{a_{i1}}{\sum_{i=0}^n a_{i1}}$$

PDAF

The best matching “observation” is now computed as a weighted combination of the predicted locations and all candidate observations...



$$\text{New location} = 1/24 * \text{predicted location} + 3/24 * o1 + 5/24 * o2 + 6/24 * o3 + 9/24 * o4$$

Can also compute a measure of uncertainty from the spread of the candidate observations.

PDAF

Kalman filter update is based on residual vector (diff between predicted location and observed location)

When using single best observation

$$\tilde{y}_k = o_{\max} - H_k \hat{x}_{k|k-1}$$

PDAF uses weighted combination of observations

$$\tilde{y}_k = \sum_{i=1}^n p_{i|1} (o_i - H_k \hat{x}_{k|k-1})$$

note: if we weren't consider the possibility of no match, this would exactly be the diff between the weighted center of mass of observations and the predicted location

PDAF

Computation of Kalman posterior covariance must change too, to incorporate weighted matches and possibility of no match.

Typical computation when single match is used:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$$

PDAF computation:

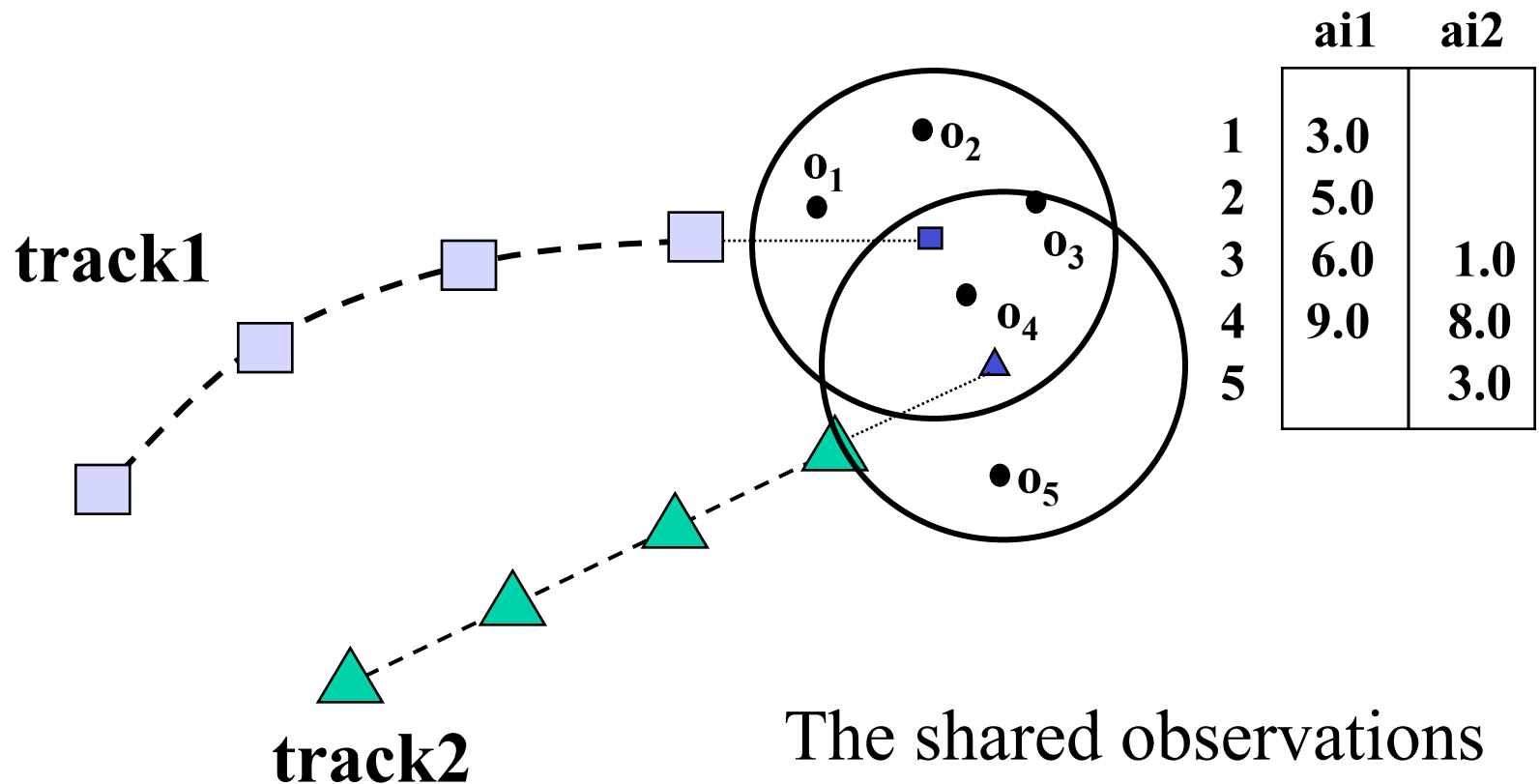
$$P_{k|k} = p_{01} P_{k|k-1} \quad \text{no match, no update}$$

$$+ (1 - p_{01}) [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k] P_{k|k-1} \quad \text{update if any match}$$

$$+ \mathbf{K}_k \text{Cov}(o_i - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) \mathbf{K}_k' \quad \text{correction term to reflect uncertain association (spread of possible matches or no match)}$$

Problem

When gating regions overlap, the same observations can contribute to updating both trajectories.



The shared observations introduce a coupling into the decision process.

JPDAF

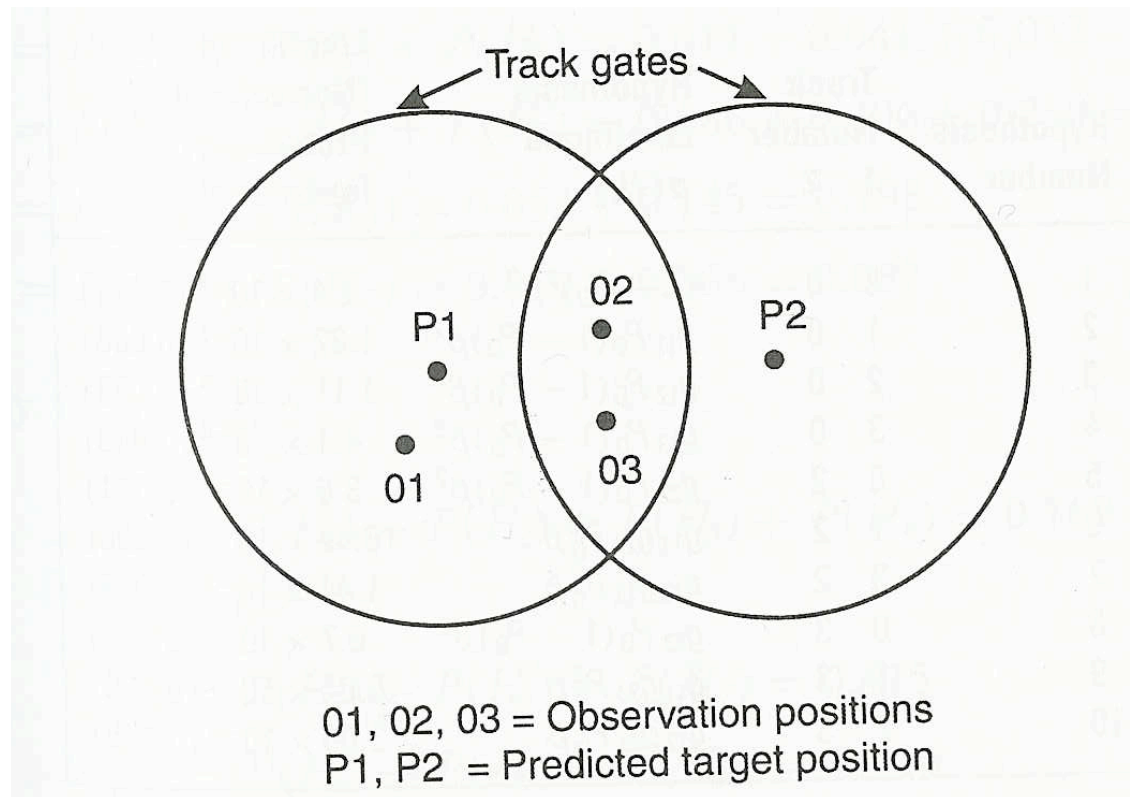
Joint Probabilistic Data Association Filter

If maintaining multiple tracks, doing PDAF on each one independently is nonoptimal, since observations in overlapping gate regions will be counted more than once (contribute to more than one track).

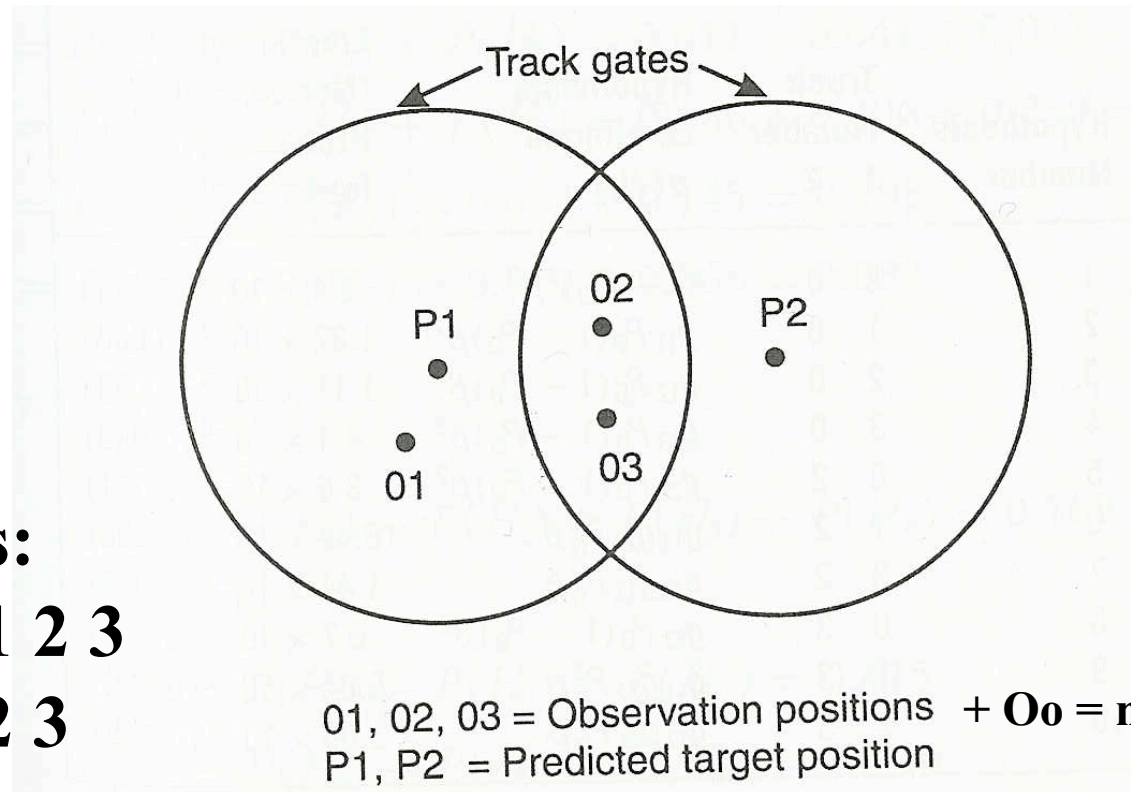
JPDAF reasons over possible combinations of matches, in a principled way.

JPDAF

Example (from Blackman and Popoli).



JPDAF



Candidates:

track1: 0 1 2 3

track2: 0 2 3

Possible assignments: (i,j) = assign i to track1, j to track2

(0, 0) (1, 0) (2, 0) (3, 0)

(0, 2) (1, 2) ~~(2, 2)~~ (3, 2)

(0, 3) (1, 3) (2, 3) ~~(3, 3)~~

**don't assign same
observation twice**

JPDAF

Each possible (non-conflicting) assignment becomes a hypothesis with an associated probability.

Table 6.8
Hypothesis Matrix for Example of Figure 6.3

Hypothesis Number	Track Number		Hypothesis Likelihood $p(H_i)$	Likelihood (Normalized Probability) for Example
	1	2		
1	0	0	$(1 - P_D)^2 \beta^3$	2.4×10^{-6} (0.011)
2	1	0	$g_{11} P_D (1 - P_D) \beta^2$	1.82×10^{-5} (0.086)
3	2	0	$g_{12} P_D (1 - P_D) \beta^2$	1.11×10^{-5} (0.053)
4	3	0	$g_{13} P_D (1 - P_D) \beta^2$	4.1×10^{-6} (0.019)
5	0	2	$g_{22} P_D (1 - P_D) \beta^2$	8.6×10^{-6} (0.041)
6	1	2	$g_{11} g_{22} P_D^2 \beta$	6.47×10^{-5} (0.306)
7	3	2	$g_{13} g_{22} P_D^2 \beta$	1.44×10^{-5} (0.068)
8	0	3	$g_{23} P_D (1 - P_D) \beta^2$	6.7×10^{-6} (0.032)
9	1	3	$g_{11} g_{23} P_D^2 \beta$	5.04×10^{-5} (0.239)
10	2	3	$g_{12} g_{23} P_D^2 \beta$	3.06×10^{-5} (0.145)

These likelihoods are based on appearance, prob of detection, prob of false alarm

JPDAF

Now compute probability p_{ij} that each observation i should be assigned to track j , by adding probabilities of assignments where that is so. Example: p_{11} = prob that observation should be assigned to track 1.

Table 6.8
Hypothesis Matrix for Example of Figure 6.3

Hypothesis Number	Track Number		Hypothesis Likelihood $p(H_i)$	Likelihood (Normalized Probability) for Example
	1	2		
1	0	0	$(1 - P_D)^2 \beta^3$	2.4×10^{-6} (0.011)
2	1	0	$g_{11} P_D (1 - P_D) \beta^2$	1.82×10^{-5} (0.086)
3	2	0	$g_{12} P_D (1 - P_D) \beta^2$	1.11×10^{-5} (0.053)
4	3	0	$g_{13} P_D (1 - P_D) \beta^2$	4.1×10^{-6} (0.019)
5	0	2	$g_{22} P_D (1 - P_D) \beta^2$	8.6×10^{-6} (0.041)
6	1	2	$g_{11} g_{22} P_D^2 \beta$	6.47×10^{-5} (0.306)
7	3	2	$g_{13} g_{22} P_D^2 \beta$	1.44×10^{-5} (0.068)
8	0	3	$g_{23} P_D (1 - P_D) \beta^2$	6.7×10^{-6} (0.032)
9	1	3	$g_{11} g_{23} P_D^2 \beta$	5.04×10^{-5} (0.239)
10	2	3	$g_{12} g_{23} P_D^2 \beta$	3.06×10^{-5} (0.145)

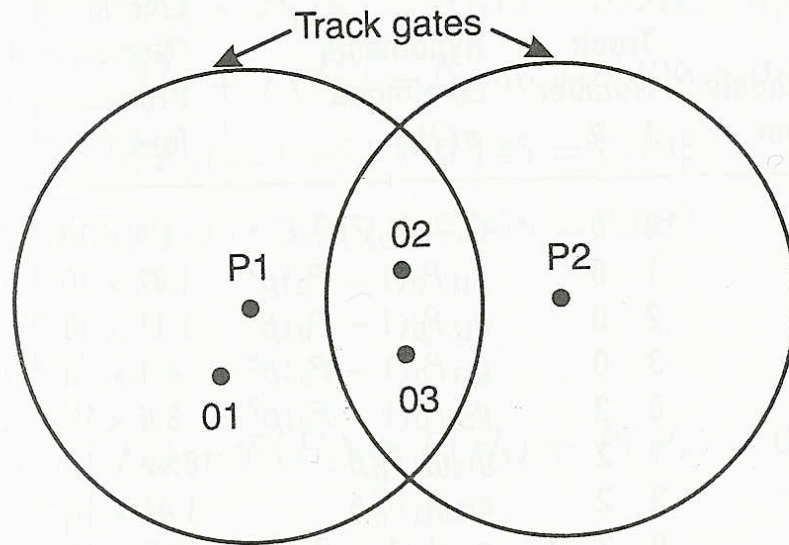
.086

+ .306

+ .239

.631

JPDAF



01, 02, 03 = Observation positions
P1, P2 = Predicted target position

Combined probabilities

Track1:

$$p_{10} = .084$$

$$p_{11} = .631$$

$$p_{12} = .198$$

$$p_{13} = .087$$

Track2:

$$p_{20} = .169$$

$$p_{21} = .0$$

$$p_{22} = .415$$

$$p_{23} = .416$$

JPDAF

Continuing the example:

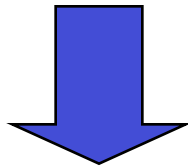
Track 1 probabilities

$$p_{01} = .084$$

$$p_{11} = .631$$

$$p_{21} = .198$$

$$p_{31} = .087$$



**PDAF filter
for track 1**

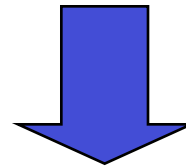
Track 2 probabilities

$$p_{02} = .169$$

$$p_{12} = 0.0$$

$$p_{22} = .415$$

$$p_{32} = .416$$



**PDAF filter
for track 2**

Note: running PDAF filters on each track independently is now OK because any inconsistency (double counting) has been removed.

Hypothesis Likelihoods

Table 6.8
Hypothesis Matrix for Example of Figure 6.3

Hypothesis Number	Track Number		Hypothesis Likelihood $p(H_i)$	Likelihood (Normalized Probability) for Example
	1	2		
1	0	0	$(1 - P_D)^2 \beta^3$	2.4×10^{-6} (0.011)
2	1	0	$g_{11} P_D (1 - P_D) \beta^2$	1.82×10^{-5} (0.086)
3	2	0	$g_{12} P_D (1 - P_D) \beta^2$	1.11×10^{-5} (0.053)
4	3	0	$g_{13} P_D (1 - P_D) \beta^2$	4.1×10^{-6} (0.019)
5	0	2	$g_{22} P_D (1 - P_D) \beta^2$	8.6×10^{-6} (0.041)
6	1	2	$g_{11} g_{22} P_D^2 \beta$	6.47×10^{-5} (0.306)
7	3	2	$g_{13} g_{22} P_D^2 \beta$	1.44×10^{-5} (0.068)
8	0	3	$g_{23} P_D (1 - P_D) \beta^2$	6.7×10^{-6} (0.032)
9	1	3	$g_{11} g_{23} P_D^2 \beta$	5.04×10^{-5} (0.239)
10	2	3	$g_{12} g_{23} P_D^2 \beta$	3.06×10^{-5} (0.145)

P_D is prob of detect
 B is prob of false alarm
 g_{ij} likelihood of observe j
 given track i

$$P(H) = \prod_{\text{Track } i \text{ assigned to observation } j} g_{ij} P_D \prod_{\text{Tracks assigned to no match } (0)} (1 - P_D) \prod_{\text{Unassigned observations}} B$$

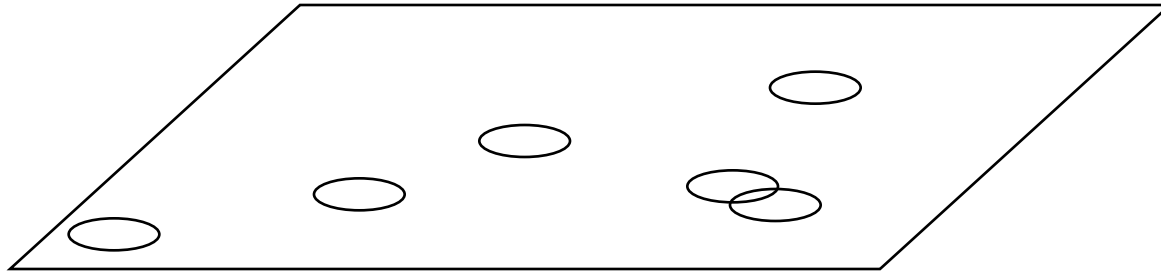
Multi-Hypothesis Tracking

Basic idea: instead of collapsing the 10 hypotheses from the last example into two trajectory updates, maintain and propagate a subset of them, as each is a possible explanation for the current state of the world.

This is a delayed decision approach. The hope is that future data will disambiguate difficult decisions at this time step.

Multi-Hypothesis Tracking

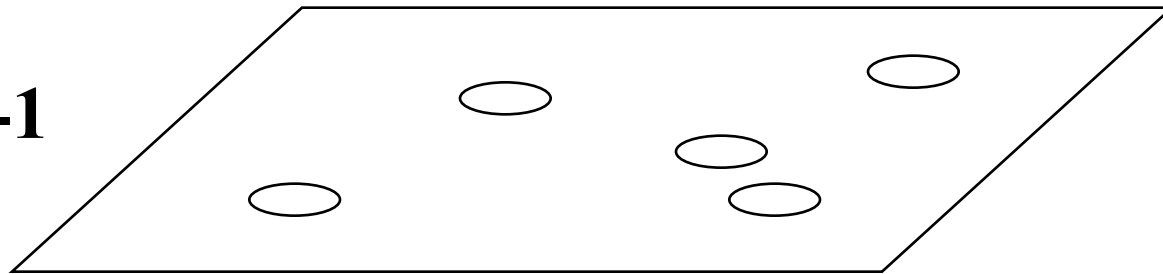
Frame t



6 detections

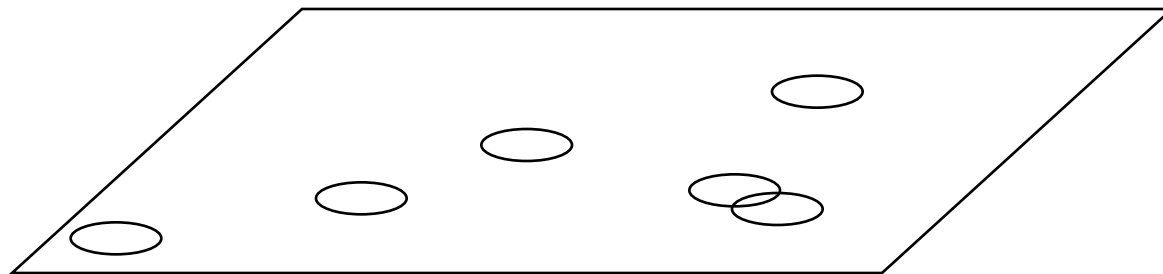
Multi-Hypothesis Tracking

Frame t-1



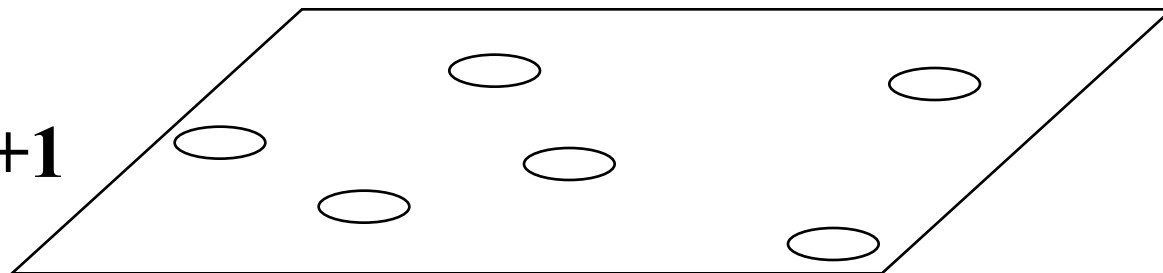
5 detections

Frame t



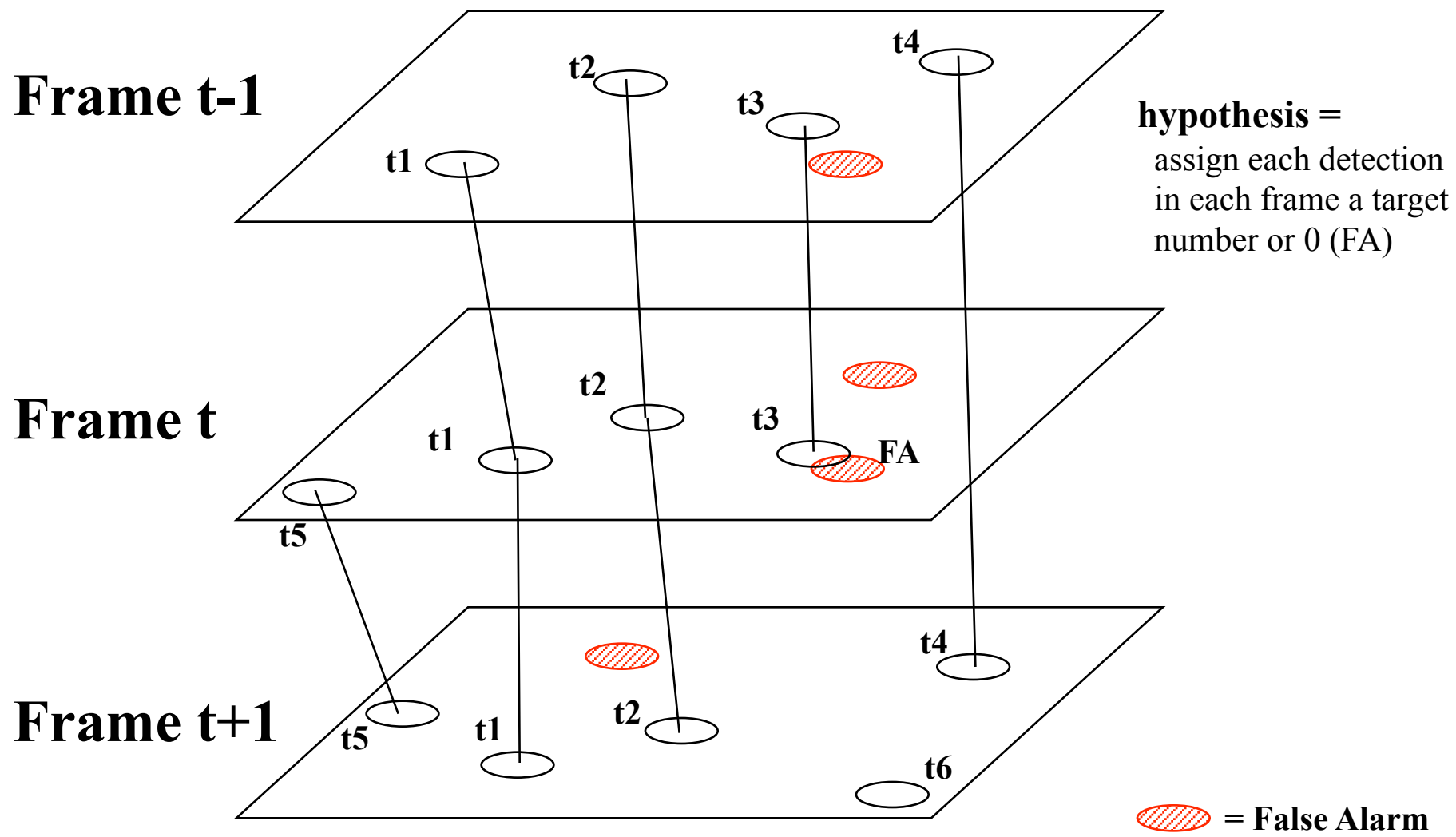
6 detections

Frame t+1

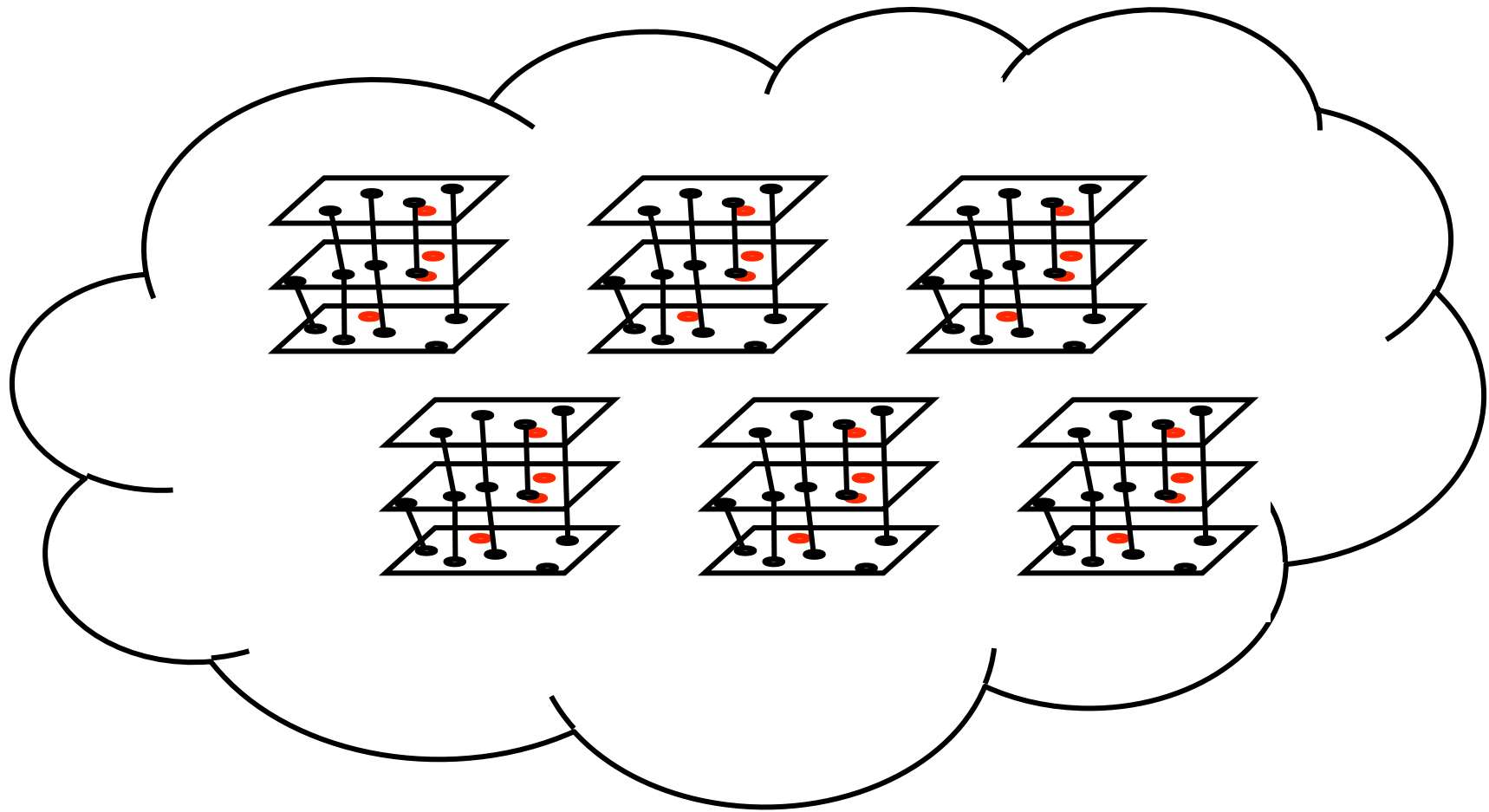


6 detections

Multi-Hypothesis Tracking



Multi-Hypothesis Tracking

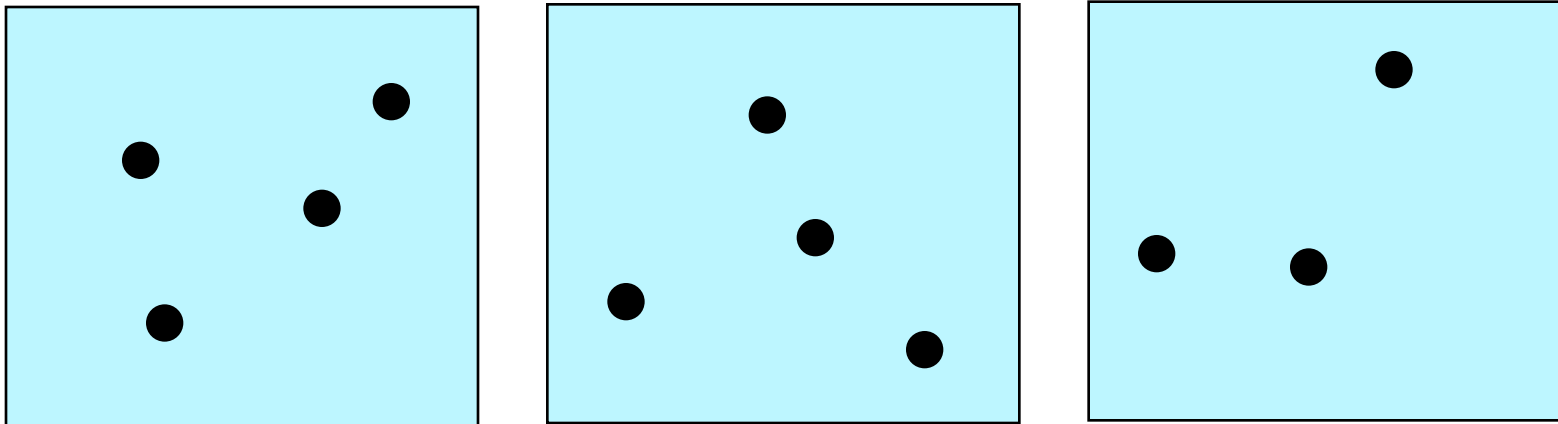


MHT maintains a set of such hypotheses. Each is one possible set of assignments of observations to targets or false alarms.

Combinatorial Explosion

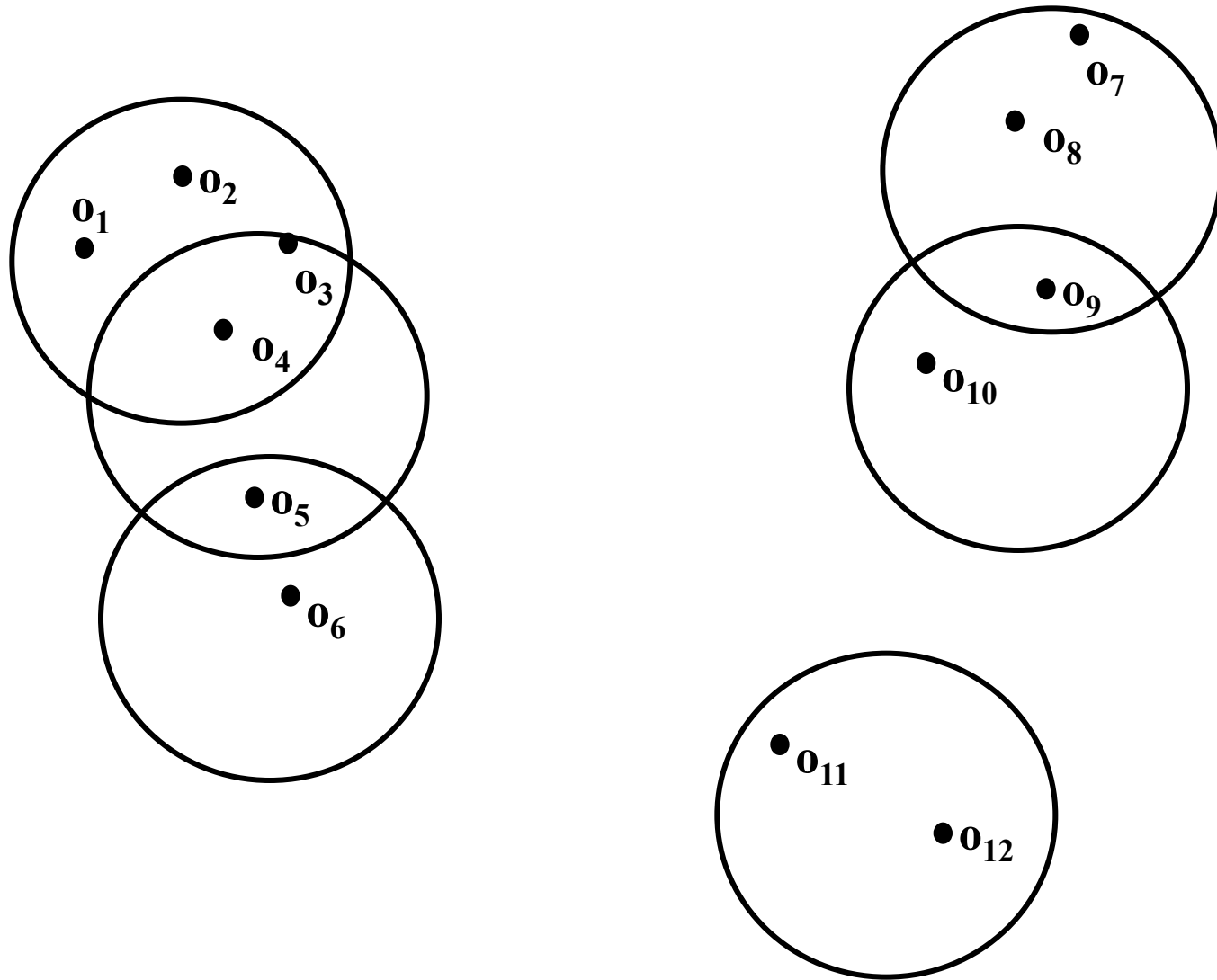
Rough order of magnitude on number of hypotheses:

Let's say we have an upper bound N on number of targets and we can associate each contact in each scan a number from 1 to N . (we are ignoring false alarms at the moment)



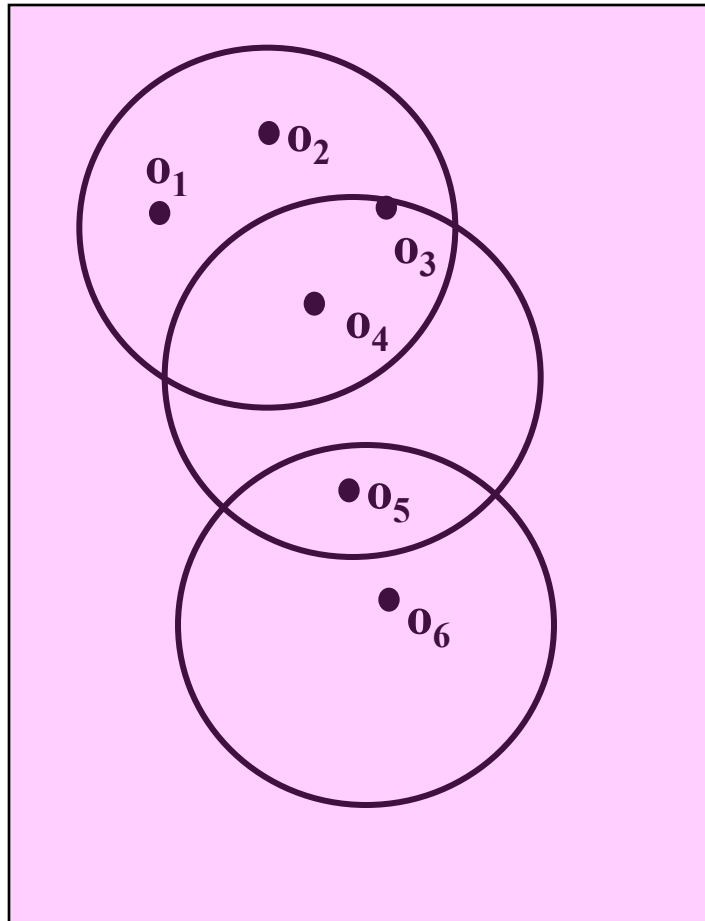
$$\frac{N!}{(N-4)!} * \frac{N!}{(N-5)!} * \frac{N!}{(N-3)!}$$

Mitigation Strategies

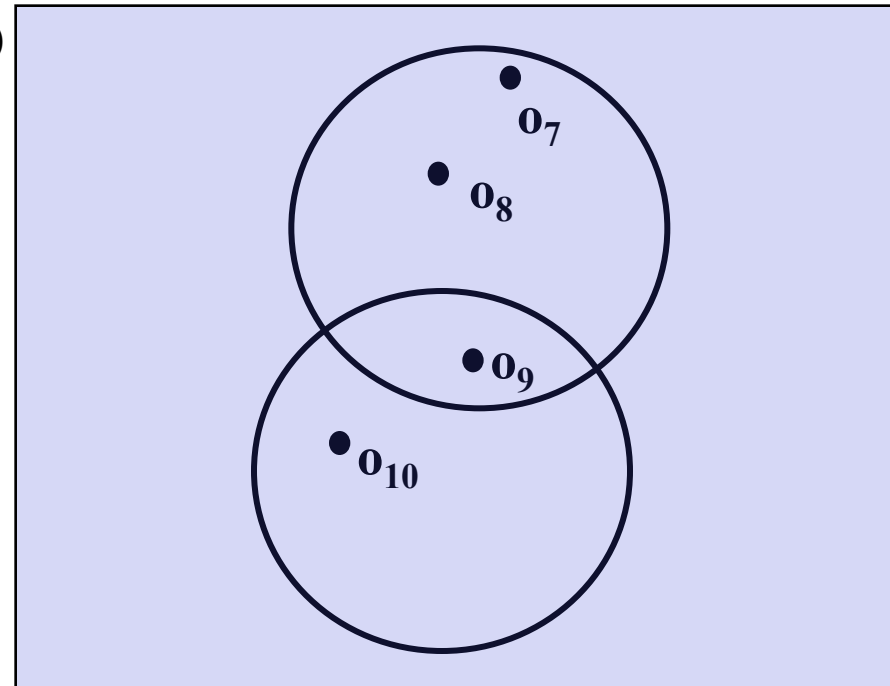


Mitigation Strategies

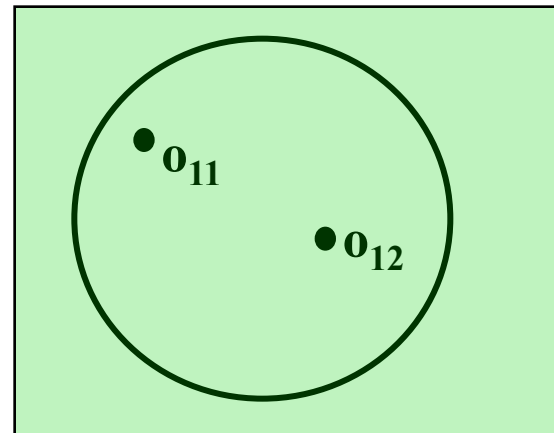
Clustering: can analyze each cluster independently (e.g. on a separate processor)



cluster1



cluster2



cluster3

Mitigation Strategies

Track Merging

merge similar trajectories (because this might allow you to merge hypotheses)

- **common observation history**
(e.g two tracks having the last N observations in common)
- **similar current state estimates**
(e.g same location and velocity in Kalman Filter)

Mitigation Strategies

Pruning: Discard low probability hypotheses

For example, one hypothesis that is always available is that every contact ever observed has been a false alarm! However, that is typically a very low probability event.

One of the most principled approaches to this is by Cox and Hingorani (PAMI'96). They combine MHT with Murty's k-best assignment algorithm to maintain a fixed set of k best hypotheses at each scan.

MHT with Murty's k-best

One way to avoid combinatorial explosion is to fix the number of hypotheses maintained at each frame, and use Murty's method for k-best assignments to find the k best hypotheses.

Example: let's say we want 5 hypotheses at each stage:

Given the 5 old hypotheses from time $t-1$, perform Hungarian algorithm to find best assignment of the observations in the current frame to each of them, forming new hypotheses at time t . Rank order them by hypothesis likelihood and put them in a priority queue:
 $\{H1, H2, H3, H4, H5\}$.

MHT with Murty's k-best

Now perform Murty's method using H1, to find the k highest variants of H1. Let's say $k=3$, and those variants are H_{11} , H_{12} , H_{13} .

By insertion sort, put them in with the original list of hypotheses, bumping out any hypotheses as necessary to keep a total list length of 5.

For example, we might now have $\{H1, H_{11}, H2, H3, H_{12}\}$, where H4, H5 and H_{13} have been discarded

Now perform Murty's method on the next item in the list, which is H_{11} , and so on. If H_{11} had been less than H2 in score, then Murty's would have been performed on H2 instead.

MCMCDA

Idea: use Markov Chain Monte Carlo (MCMC) to sample from / explore the huge combinatorial space of hypotheses.

S. Oh, S. Russell, and S. Sastry, 2004. Markov Chain Monte Carlo data association for general multiple-target tracking problems. In Proc. IEEE Int. Conf. on Decision and Control, pages 735-742, 2004.

Yu, G. Medioni, and I. Cohen, 2007. Multiple target tracking using spatio-temporal Markov Chain Monte Carlo data association. In Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition, pages 1-8, 2007.

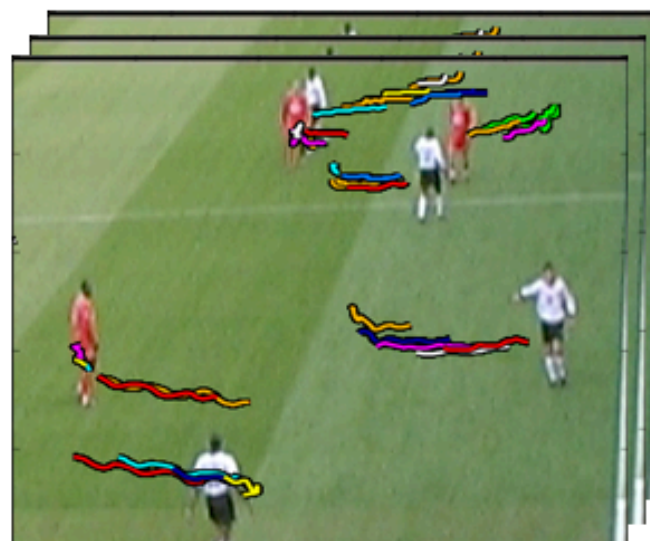


W.Ge and R.Collins, 2008, "Multi-target Data Association by Tracklets with Unsupervised Parameter Estimation," British Machine Vision Conference (BMVC'08), University of Leeds, September 2008, pp. 935-944.

MCMCDA

Rather than use detections in each frame, first extract a set of “tracklets” by tracking detections through short subsequences of the original video.

For example, detection “seeds” at every 10th frame are tracked through the next 30 frames (1 second) of video.



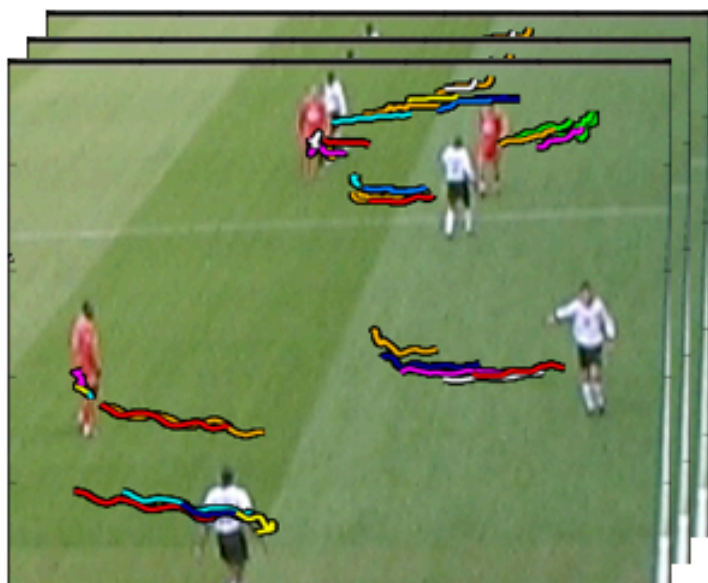
raw tracklets

Why?

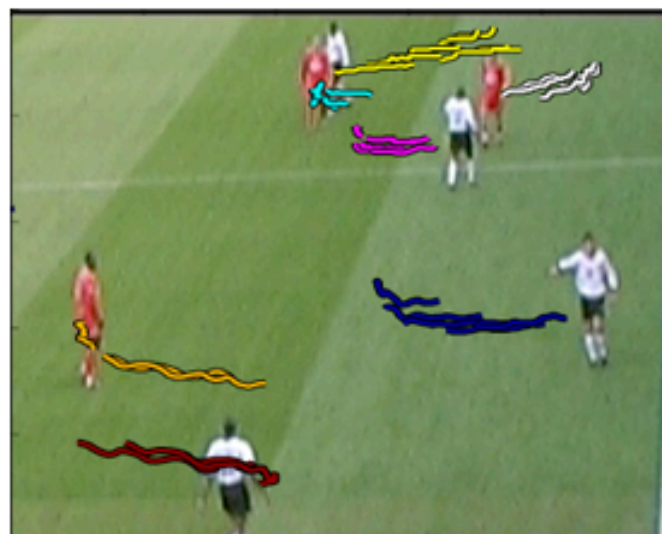
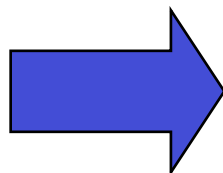
- tracklets provide more spatial/temporal context than raw detections
- short tracklets can be generated by simple (fast) trackers
- less prone to drift/occlusion than longer tracks

MCMCDA

Problem we are trying to solve: Find a partition of the set of overlapping tracklets such that tracklets belonging to the same object are grouped together. They could obviously be merged after that by a postprocessing stage.



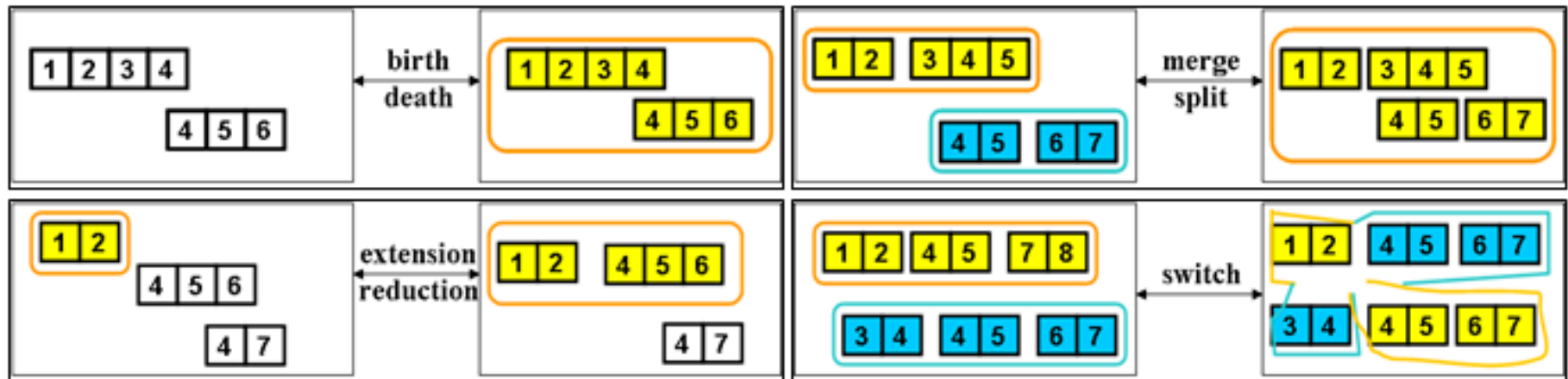
raw tracklets



estimated tracklet partition

MCMCDA

Recall that MCMC stochastically explores the search space (of tracklet partitions, in this example) by proposing a set of “moves” from the current state to a new state.



MCMC moves

MCMCDA

MCMC then decides whether or not to accept the proposal based, in part, on the ratio of likelihoods of the current state and the proposed state.

Z=observed tracklets

w=partition into trajectories

$$\omega^* = \arg \max_{\omega} (p(\omega|Z)) \xleftrightarrow{\text{Bayes' rule}} \arg \max_{\omega} (\underbrace{p(Z|\omega)}_{\text{likelihood}} \underbrace{p(\omega)}_{\text{prior}})$$

The likelihood and prior are calculated as functions of 8 features:

Likelihood features

Color Appearance

Object Size

Spatial Proximity

Velocity Coherence

Prior features

False Alarm Rate

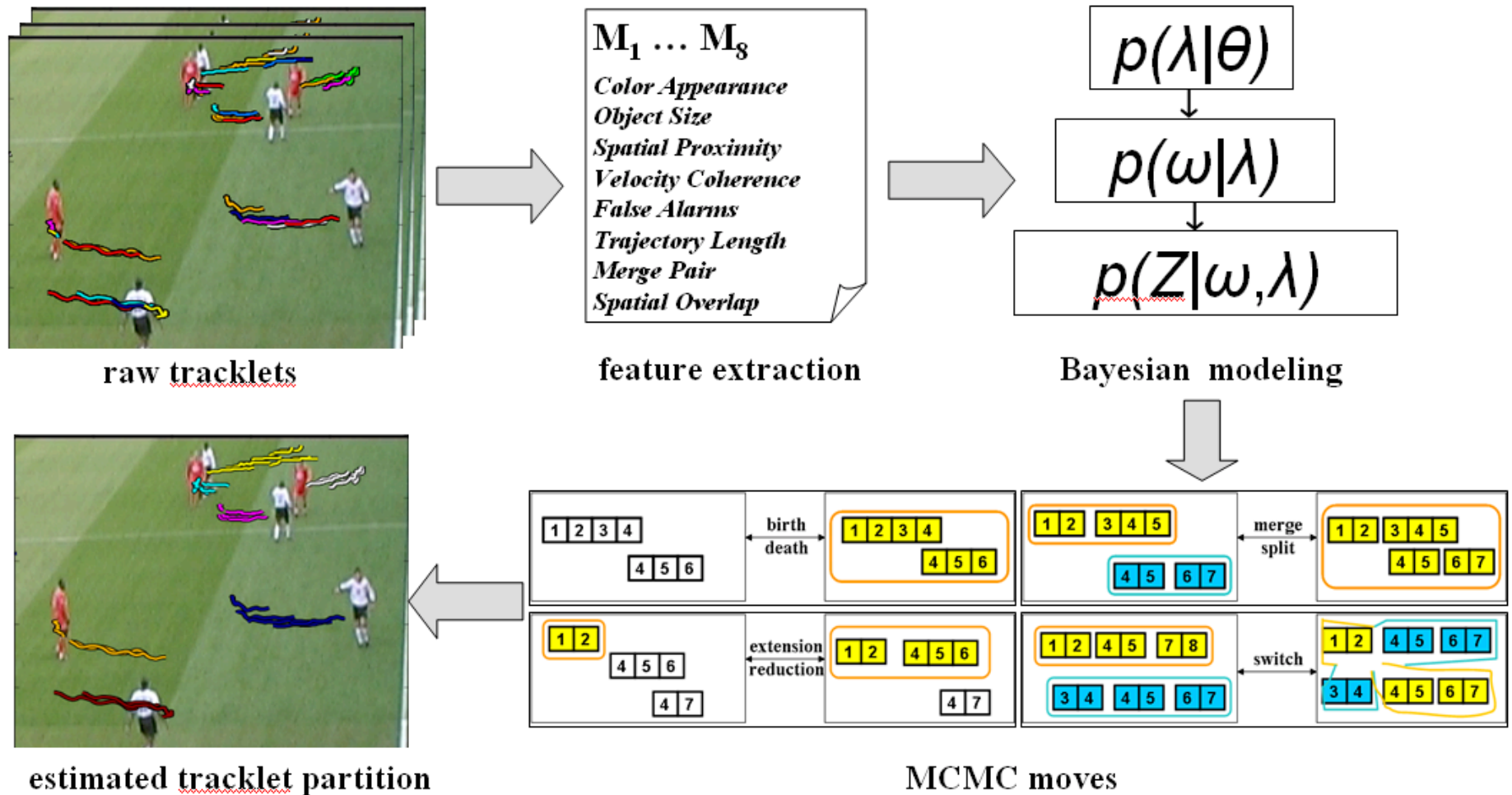
Trajectory Length

Merge Pair (encourage merging rather than starting new tracks)

Spatial Overlap (discourage overlaps between diff tracks)

MCMCDA

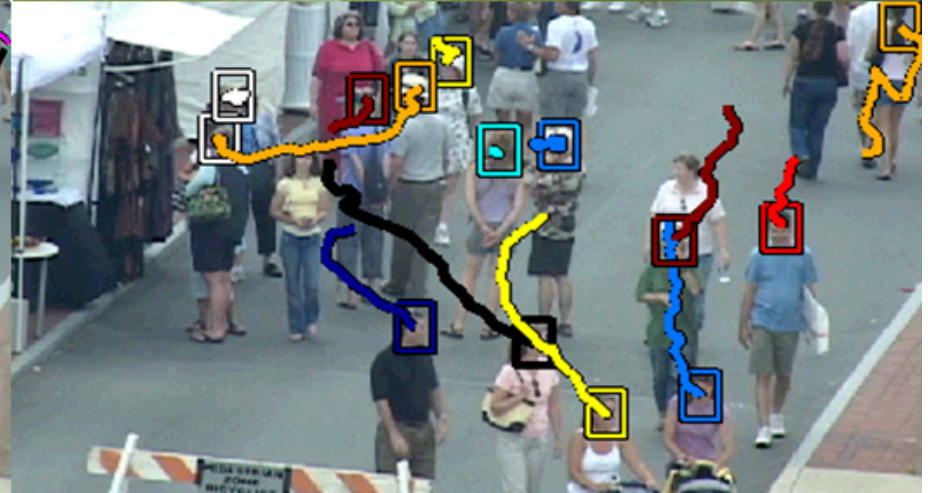
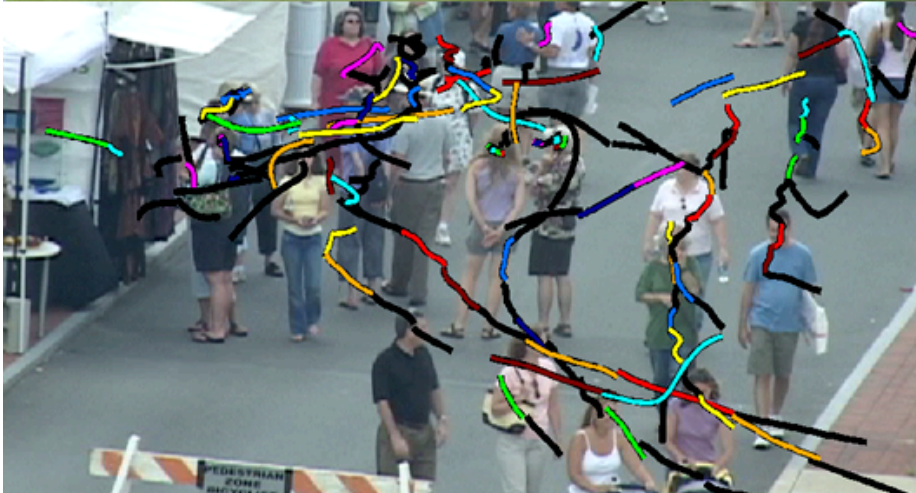
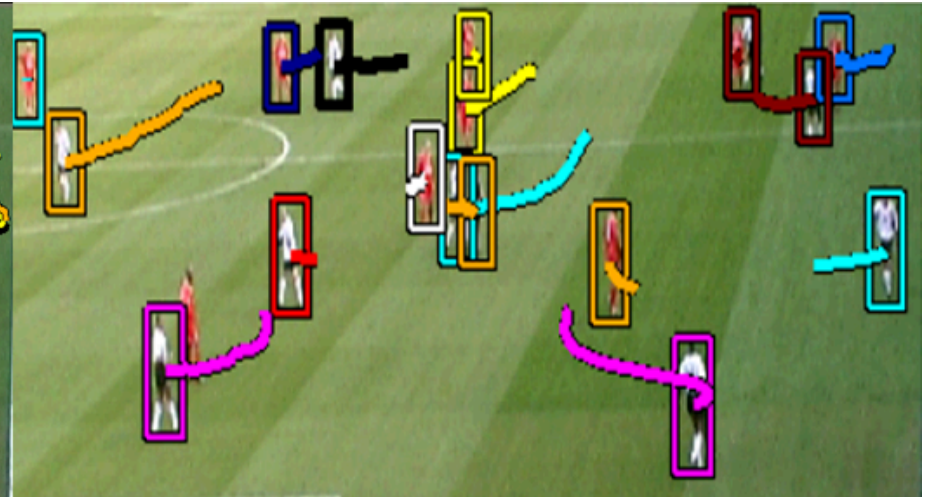
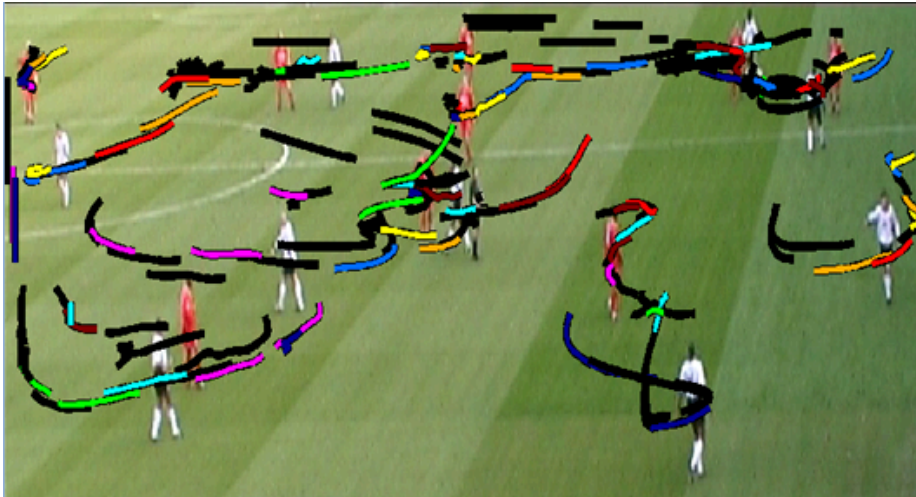
Putting it all together:



MCMCDA

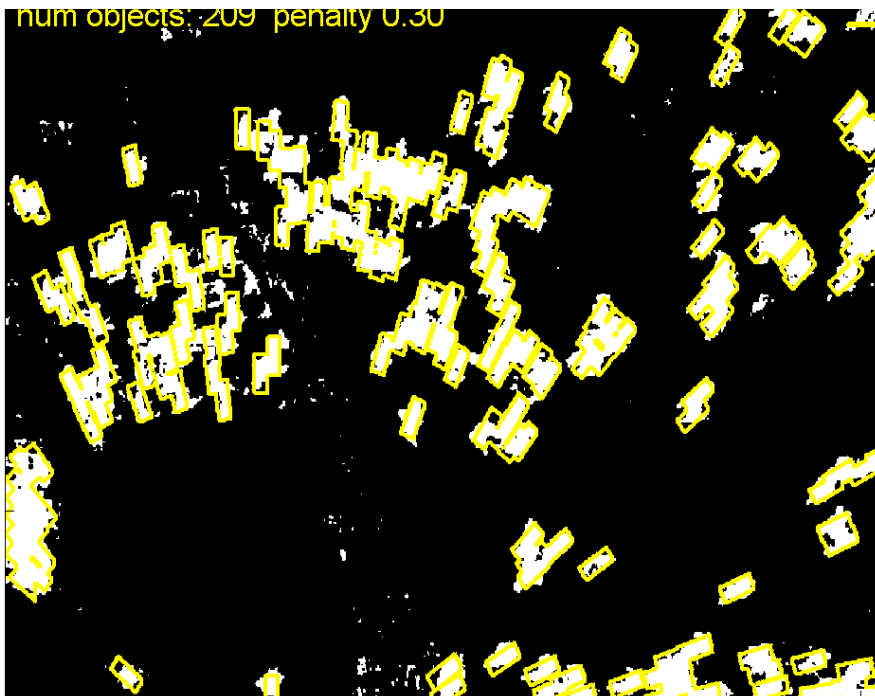
input tracklets

hypothesized tracks (at some time)



Application: Crowd Analysis

- Recall we had a method for blob/person detection in each frame.



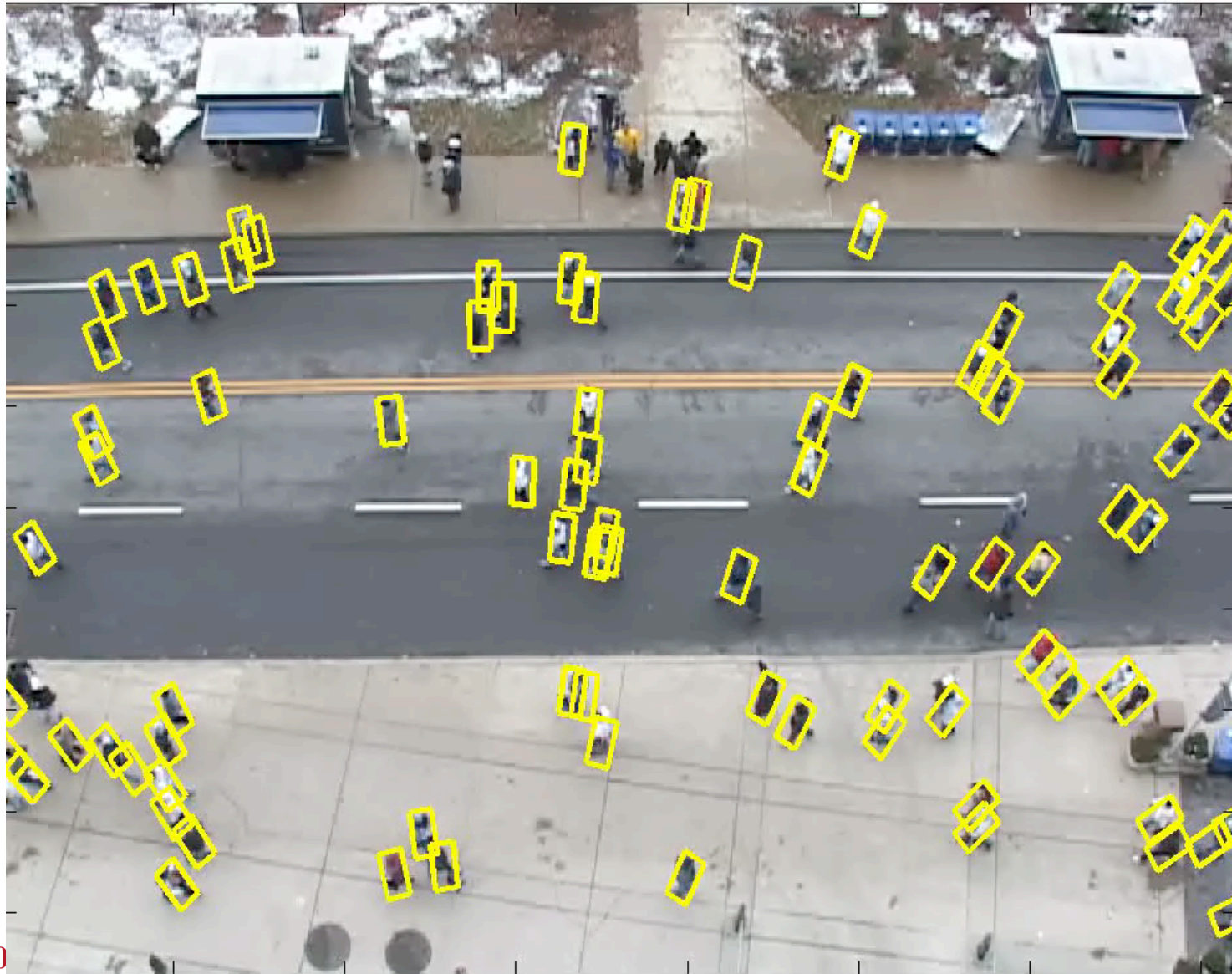
Good for low-resolution / wide-angle views.

Relies on foreground/background segmentation.

Not appropriate for very high crowd density or stationary people.

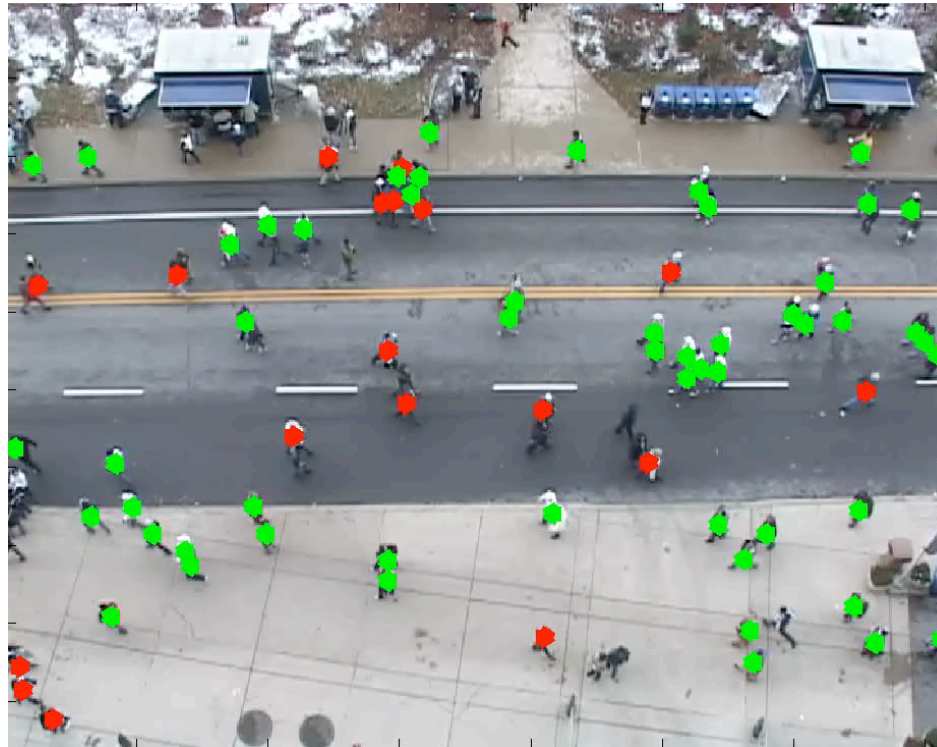
Detections, Nov 22, Curtin Road

count 94



Crowd Behavior

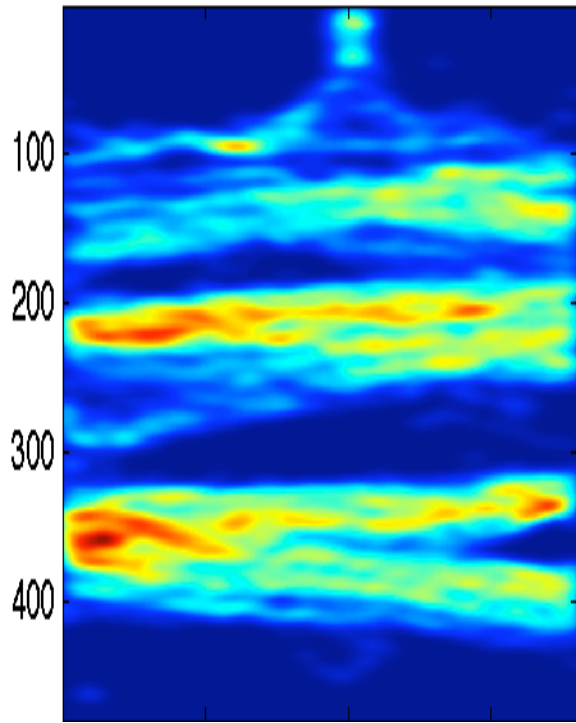
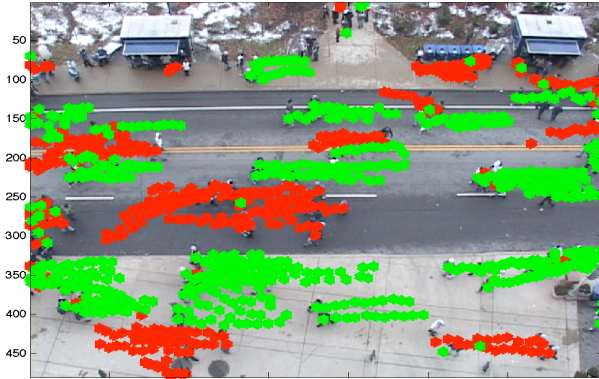
- In areas of bidirectional motion, people tend to follow others to minimize collisions (maximize throughput). Known as the “fingering” effect.



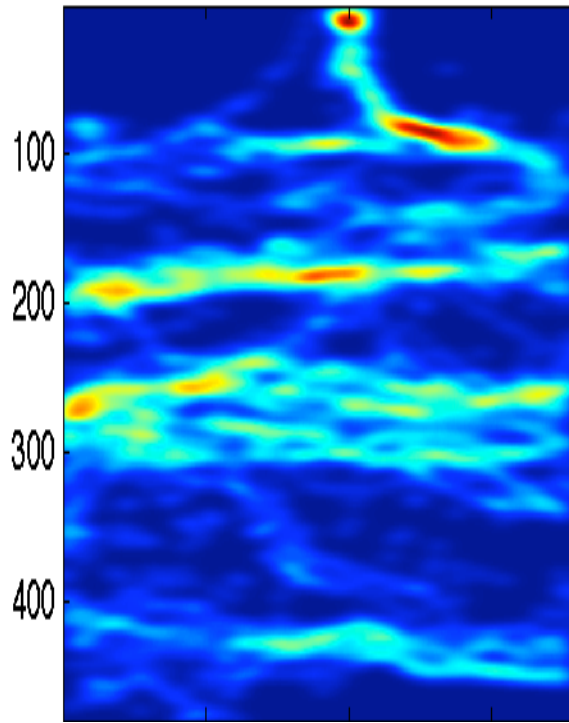
movie

Green: leftward moving. Red: rightward moving,

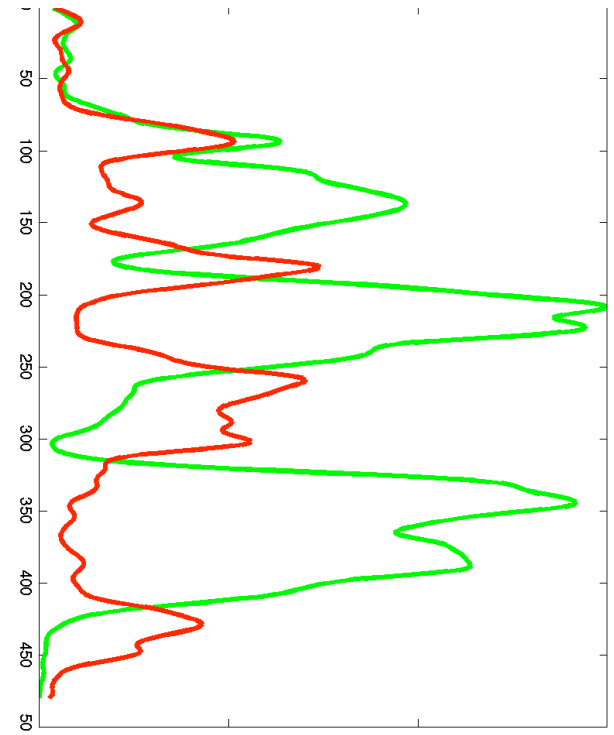
Fingering Effect



<-- Leftward



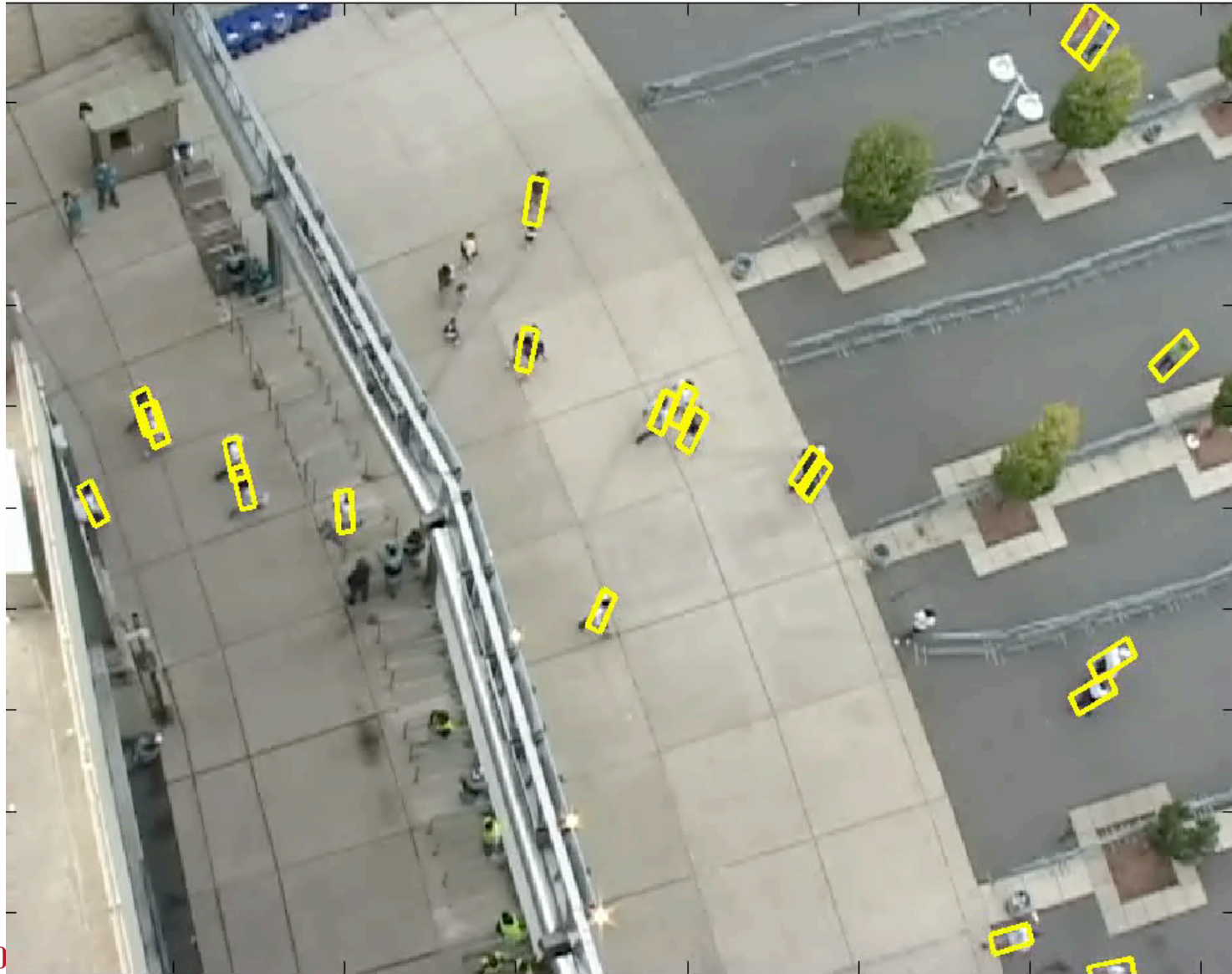
Rightward -->



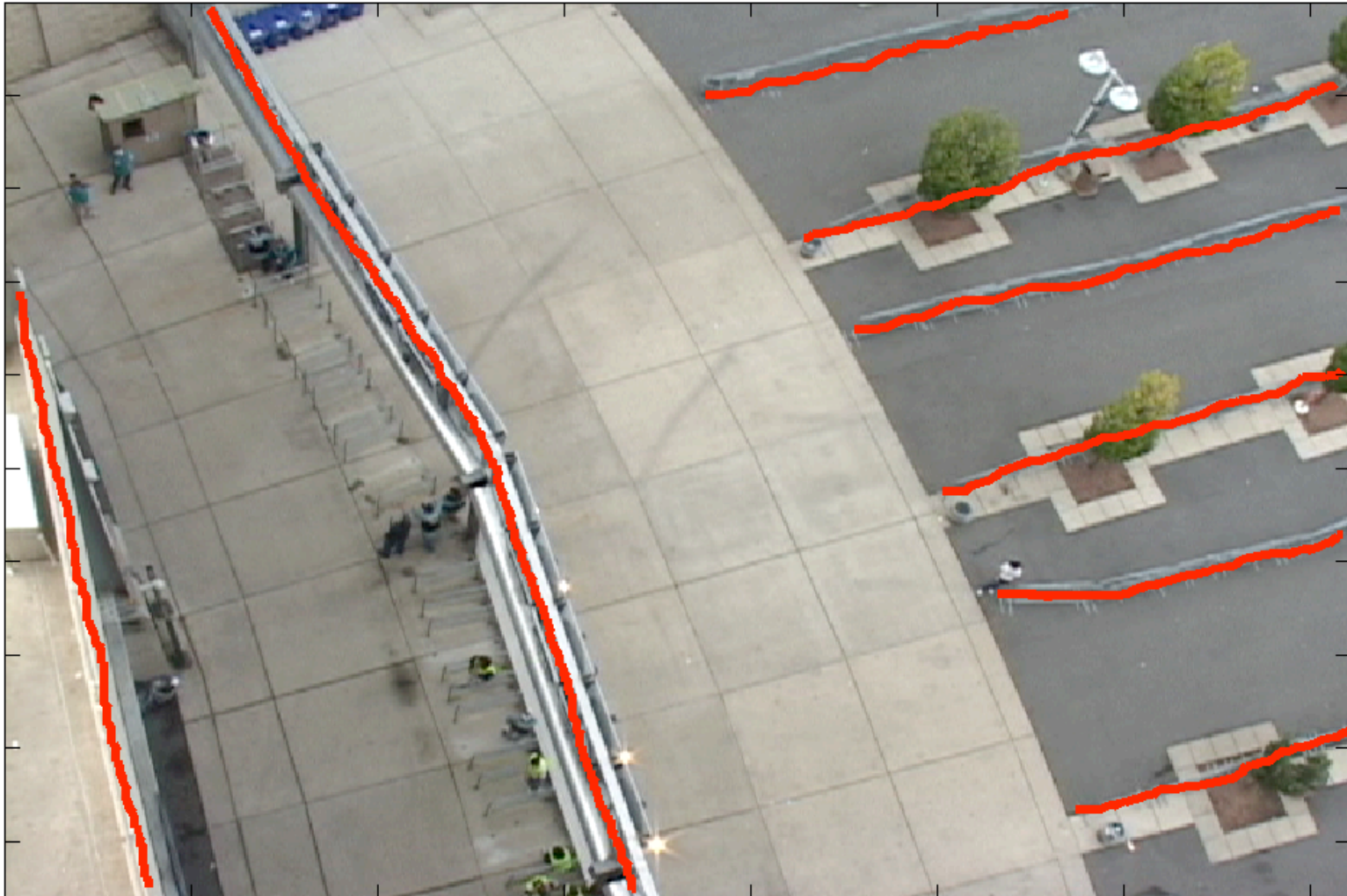
Density by image row
Green (leftward); Red (rightward)

Detections, Sep 6, Gate A

count 21

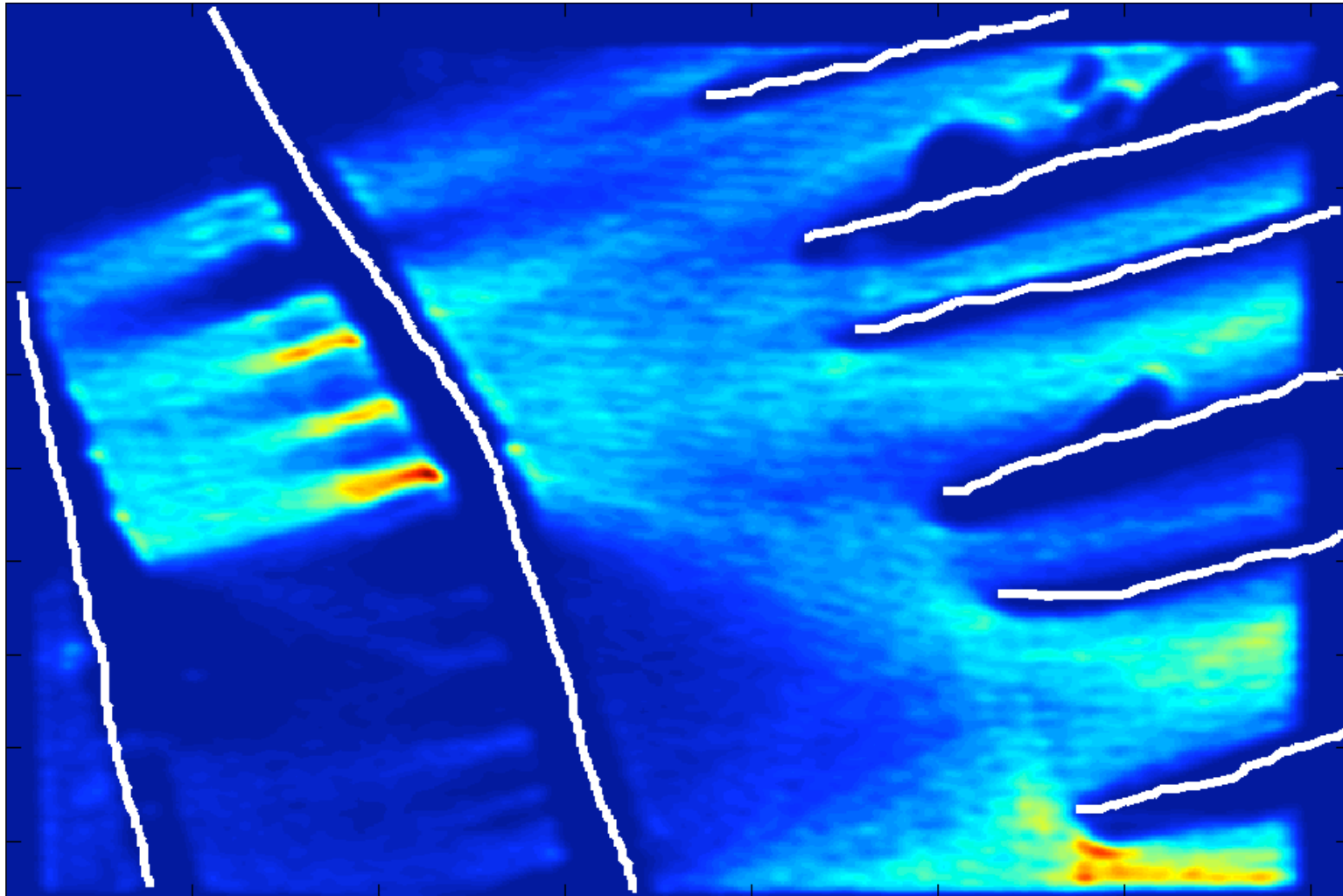


Crowd Flow/Density



Keep in mind this scene structure (as depicted by red lines)

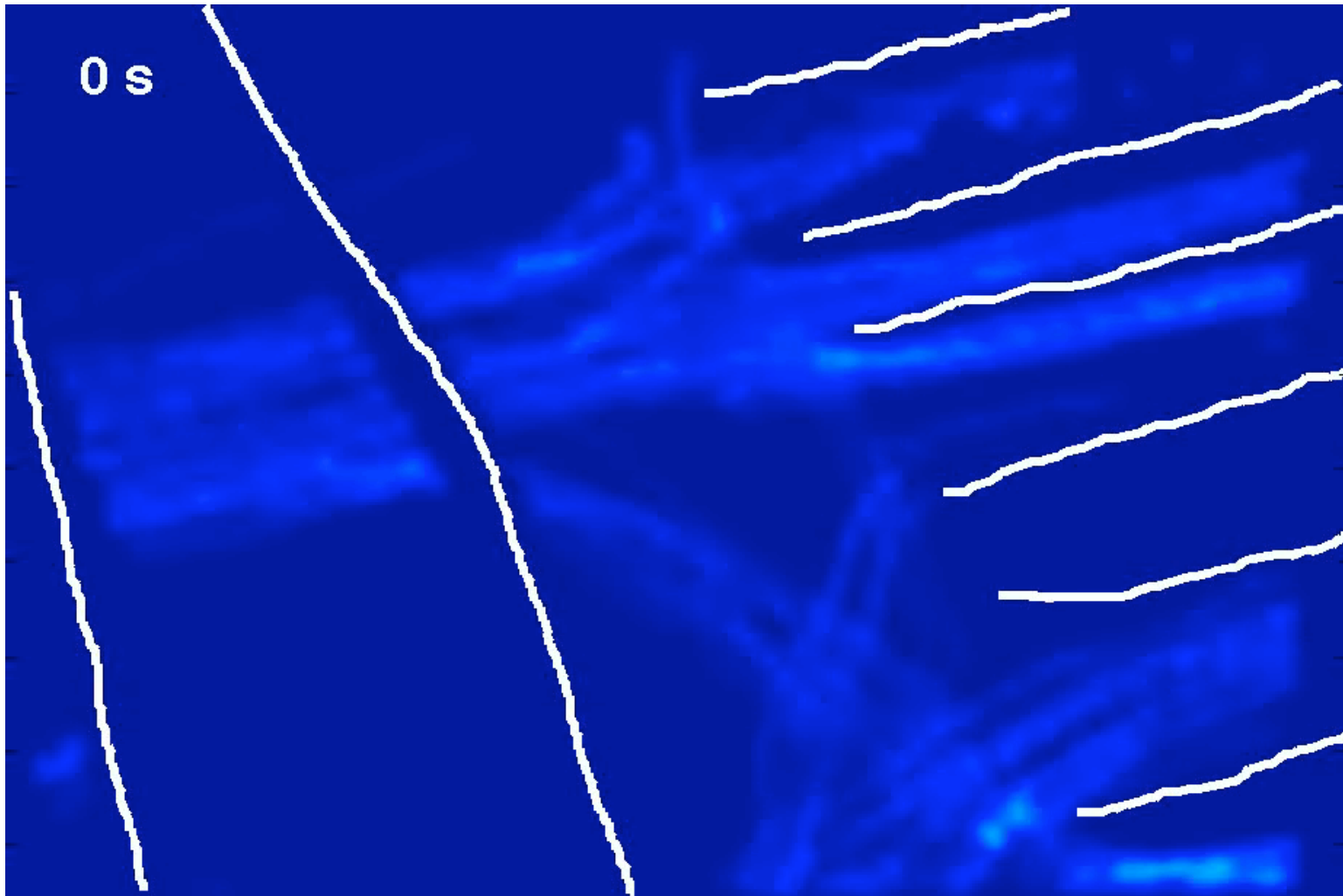
Crowd Flow/Density



30 minute period

Crowd Flow/Density

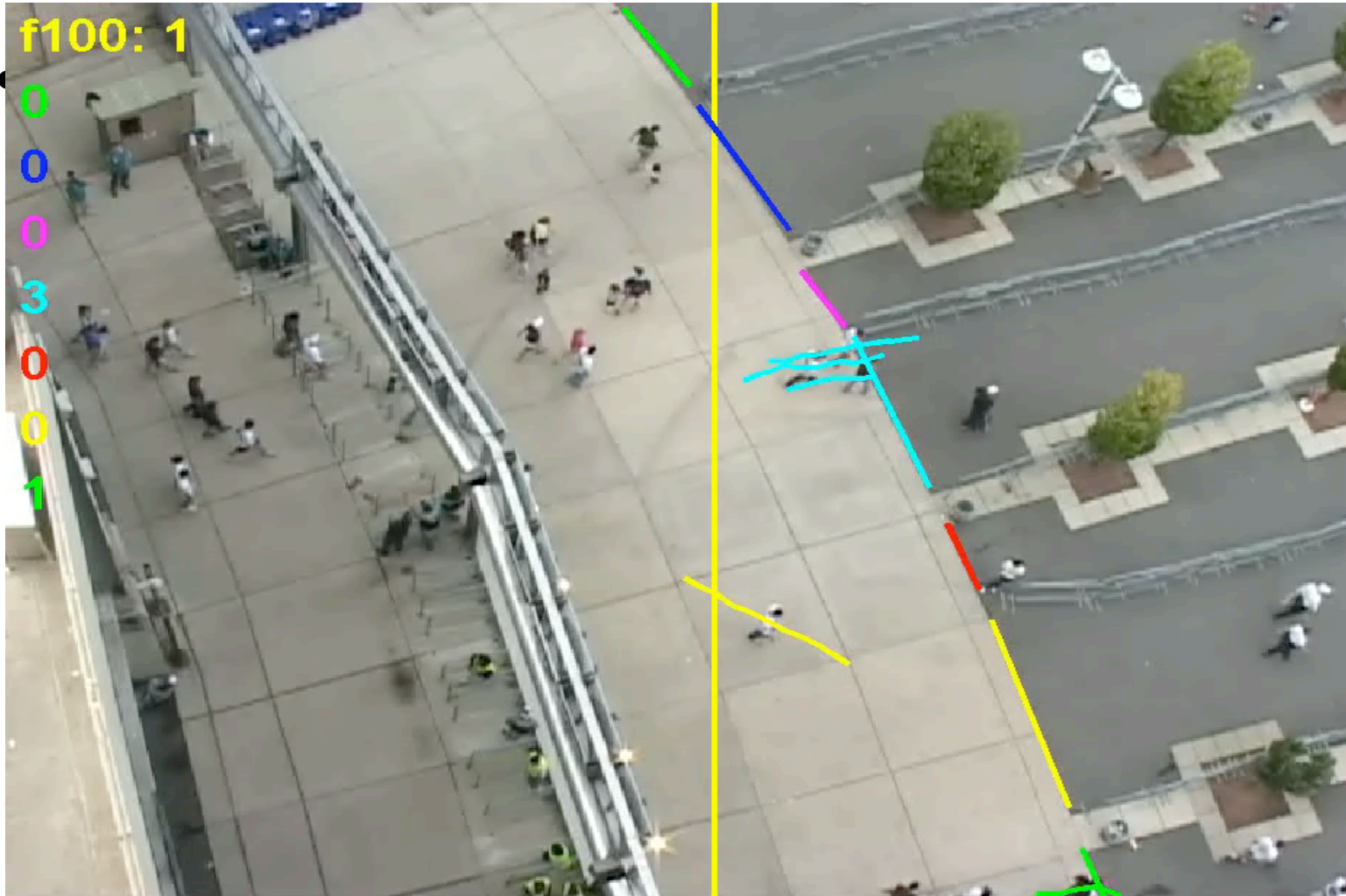
movie



Time Lapse. Integrated over spatial/temporal windows.

GateA Path Counts

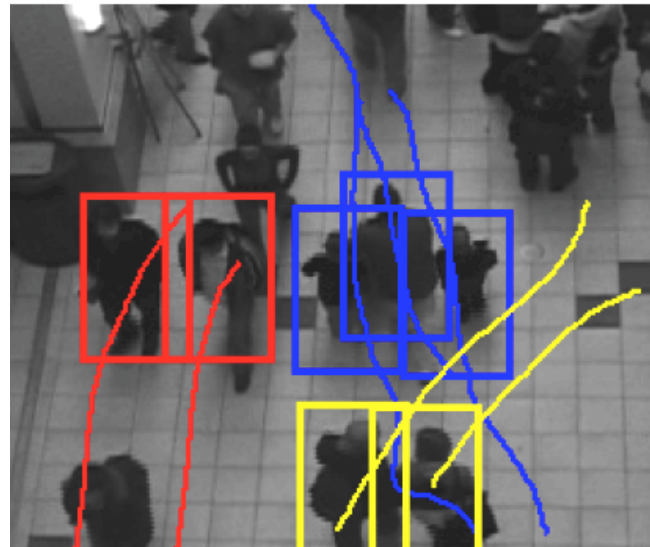
movie



Maintain a running count of number of people whose trajectories cross a set of user-specified lines (color-coded).

Collective Locomotion

- **Find small groups traveling together**
 - Sociological hypothesis: validating that the majority of people in the crowd cluster in small groups
 - Public safety: improving situation awareness and emergency response during public disturbances



Sample Grouping Results

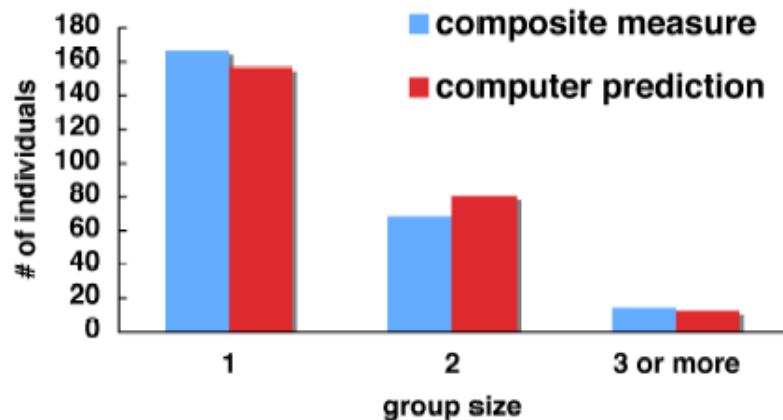


PSU
HUB

movie

note: computer only sees this view!

Evaluation reveals substantial agreement between computer-generated groupings and those found by human coders (ground truth)



	match rate	$\chi^2(4, 248)$	Cohen's κ
trichotomous	85%	219.98	.69
dichotomous	89%	138.26	.75

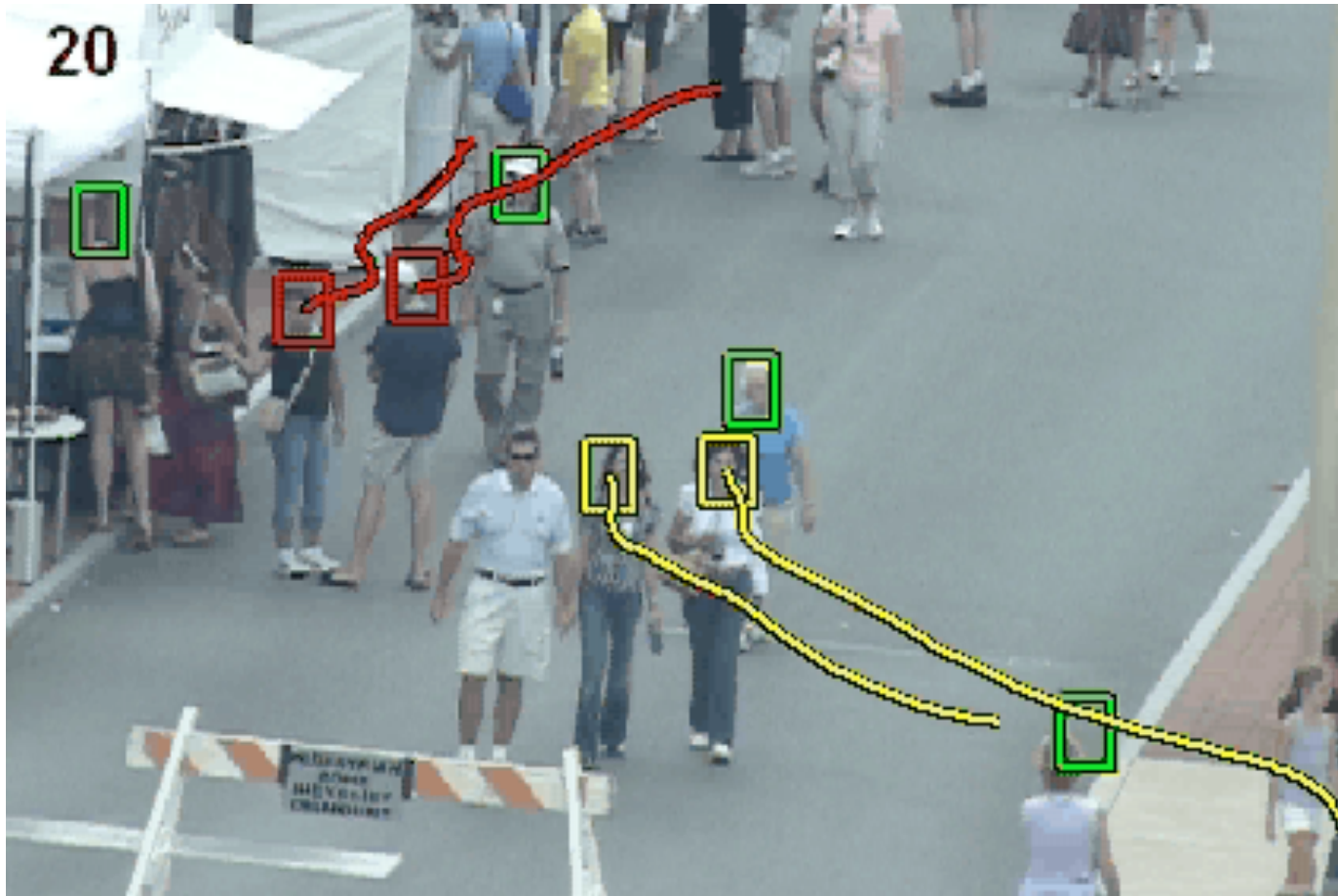
$p < .001$

Robert Collins
Penn State

Sample Grouping Results



Sample Grouping Results



movie

Arts Festival, PSU campus

Research Directions

- Validation; improve algorithm robustness.
- Detection of stationary people
- Tackle the HARD problems. Primarily high-density crowds.



Aside: Camera Motion

Hypothesis: constant velocity target motion model is adequate provided we first compensate for effects of any background camera motion.

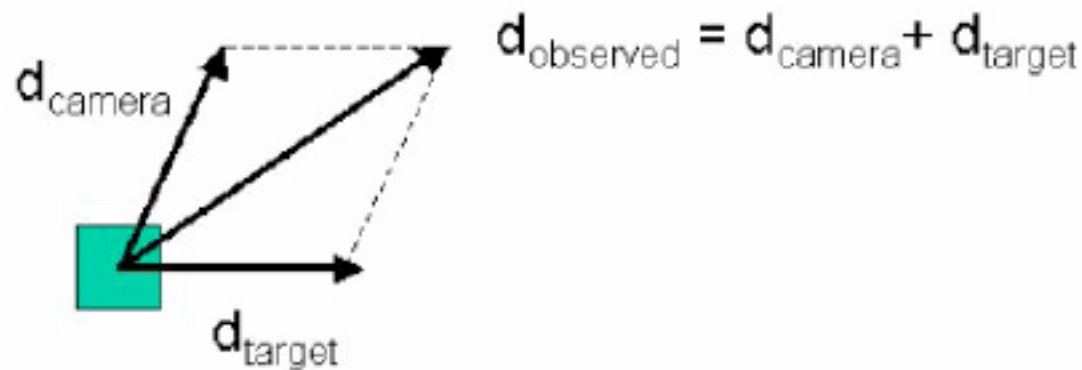


Figure 3: Decomposition of observed displacement of a target between two video frames into terms based only on motion of the camera and motion of the target.

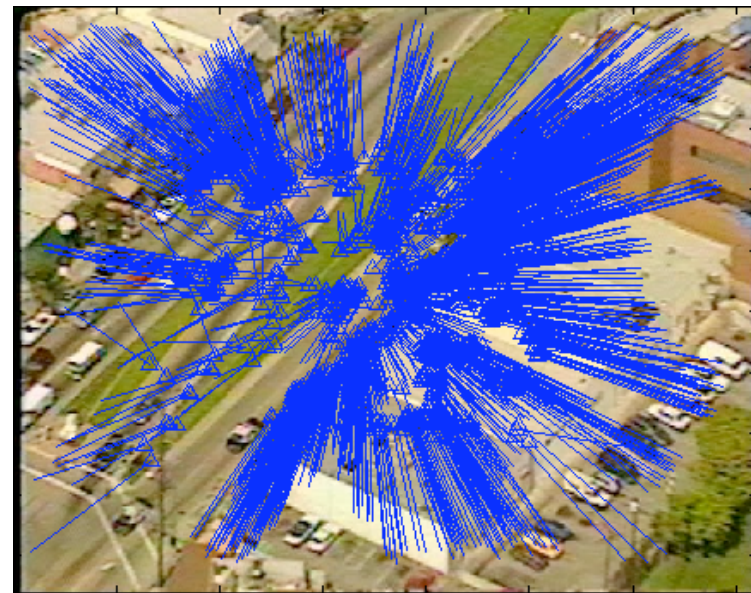
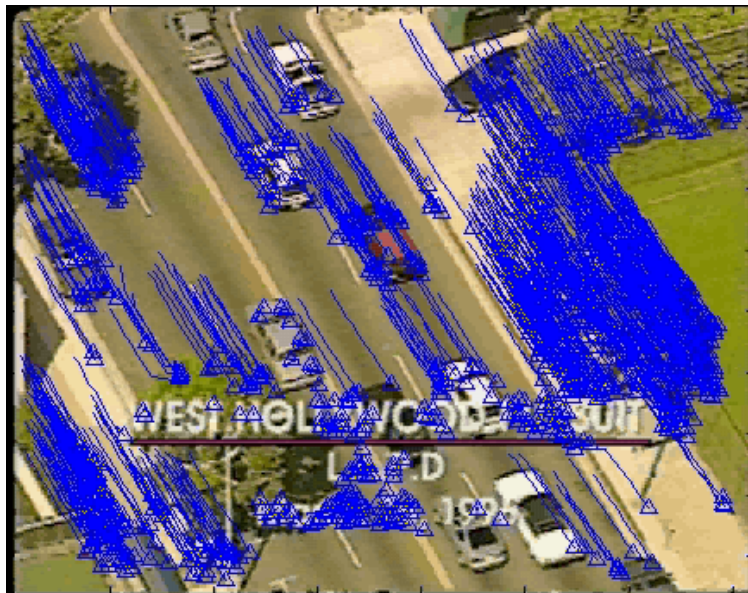
Camera Motion Estimation

Approach:

Estimate sparse optic flow using Lucas-Kanade algorithm (KLT)

Estimate parameteric model (affine) of scene image motion

Note: this offers a low computational cost alternative to image warping and frame differencing approaches.



used for motion prediction, and zoom detection

Parameteric Camera Motion Model

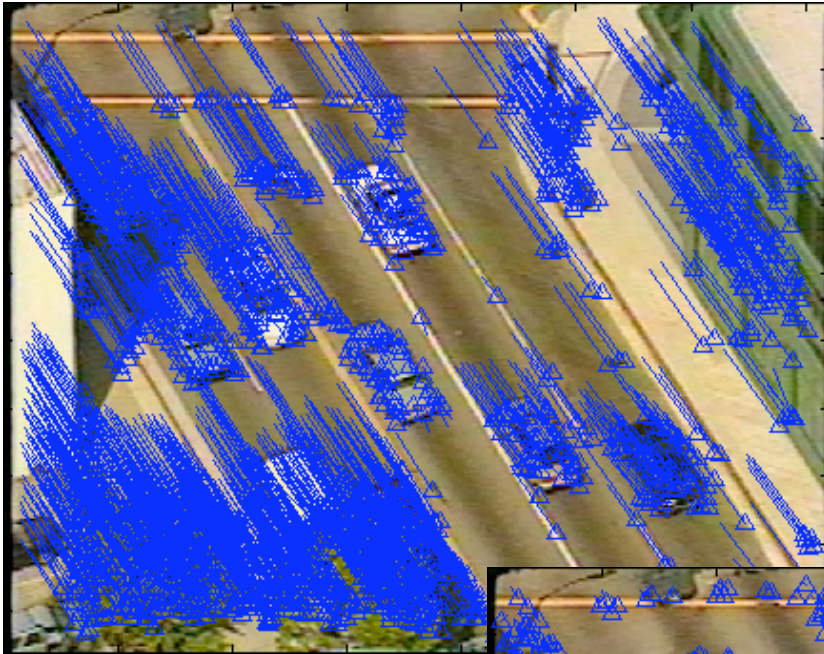
Apparent motion of stationary scene pixels in the image is a function of camera motion (R, T) and scene structure (depth at each pixel).

Assumption: for small field of view aerial camera, viewing a target on the ground, apparent scene motion in a subsequence can be modeled as low-parameter, global image transformation
e.g. 6 parameter affine or 8 parameter projective

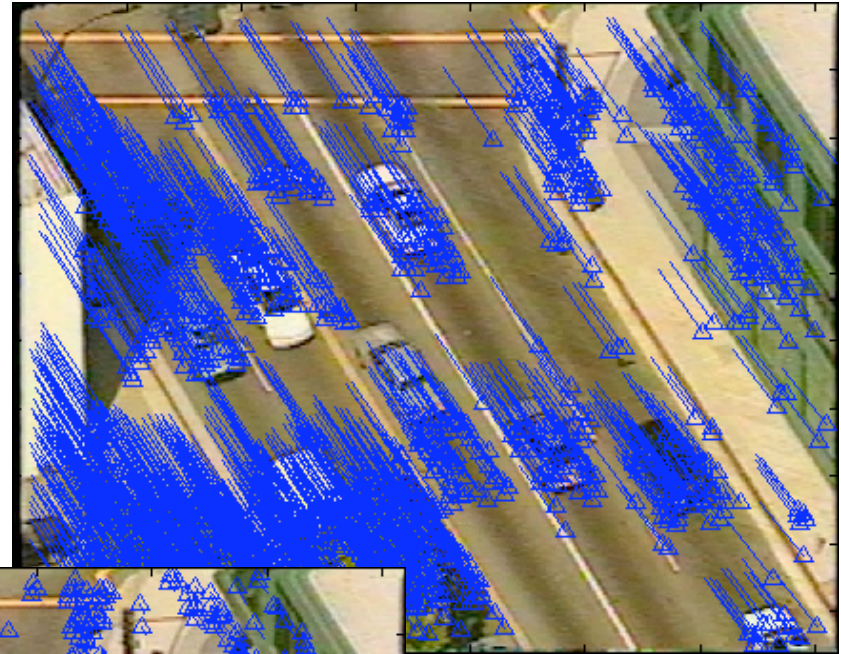
(note: this assumption has been demonstrated to be good based on the success of Sarnoff's image stabilization work. Main difference: they are estimating over each pixel to do explicit image warping. We will estimate from a sparse flow field, and do NOT do any warping).

To estimate a set of global flow parameters from possibly noisy flow vectors, we can use a robust sampling estimation method such as RANSAC or least median-of-squares.

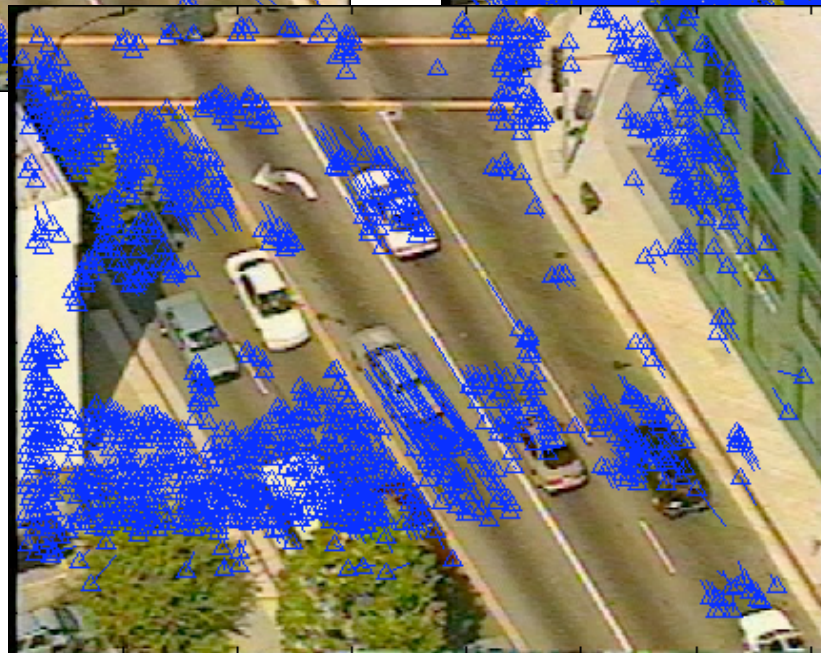
Samples of Affine Flow Fitting



original flow

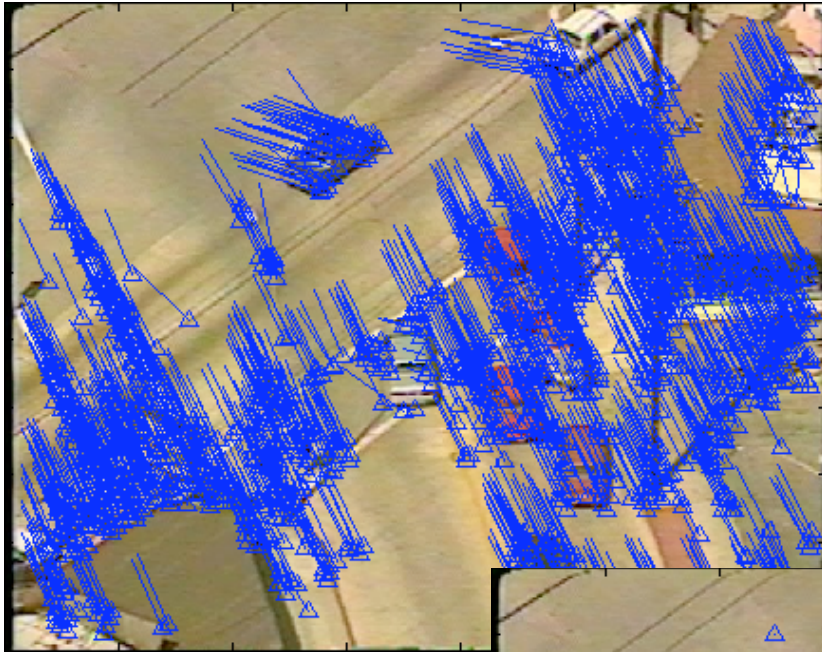


affine flow

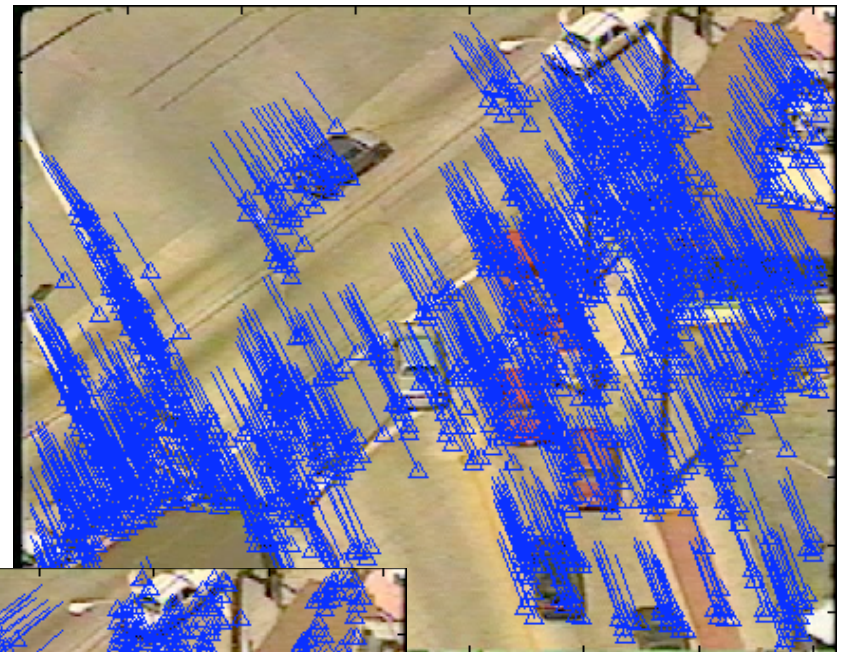


compensated flow

Samples of Affine Flow Fitting



original flow



affine flow



compensated flow

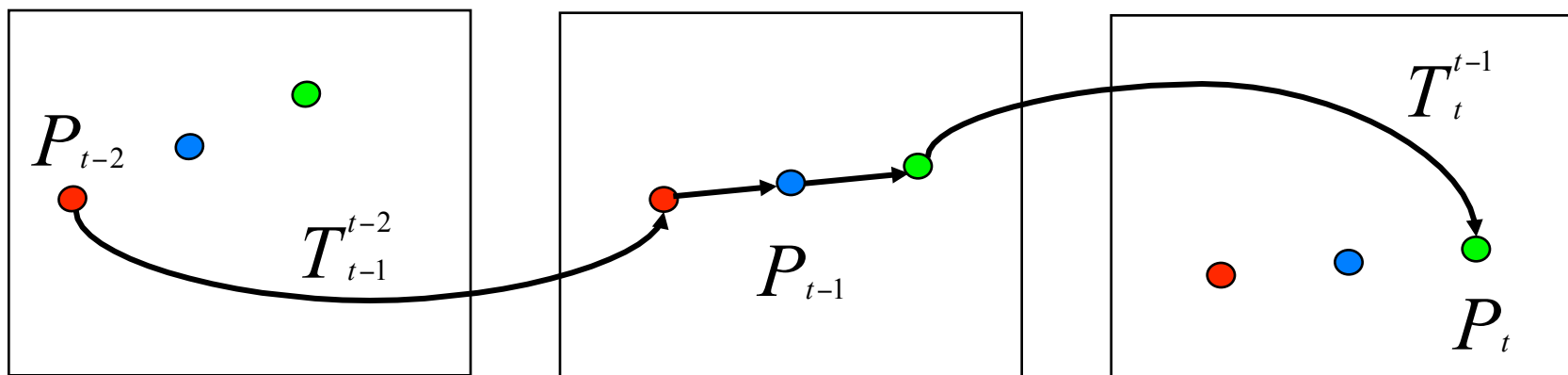
Target Motion Estimation

Approach: Constant velocity estimate, after compensating for camera motion

P_f = target position in frame f

T_g^f = camera motion from frame f to frame g

$$P_t = T_t^{t-1} * [P_{t-1} + (P_{t-1} - (T_{t-1}^{t-2} * P_{t-2}))]$$



Validation

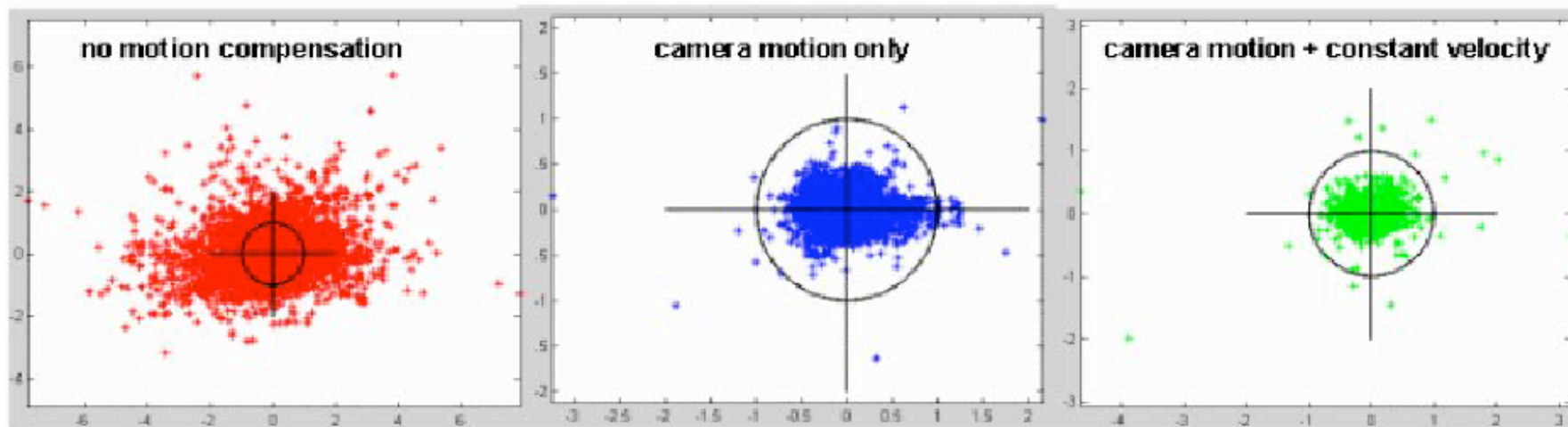


Figure 5: Validation of camera motion compensation and constant velocity location prediction. Units in the plots are centroid displacement / $\sqrt{\text{pixels on target}}$. Each plot is overlaid with a circle of unit radius. These plots result from 13,852 motion prediction tests.

Now back to data association...

Overview

Part 1: Change/Motion Detection

Basics: BG subtraction; Frame Difference

Classification-based methods

Part 2: From Pixels to 2D Blobs

Detection via RJMCMC

Classifier Grids

Part 3: Data Association

Linear Assignment Problem

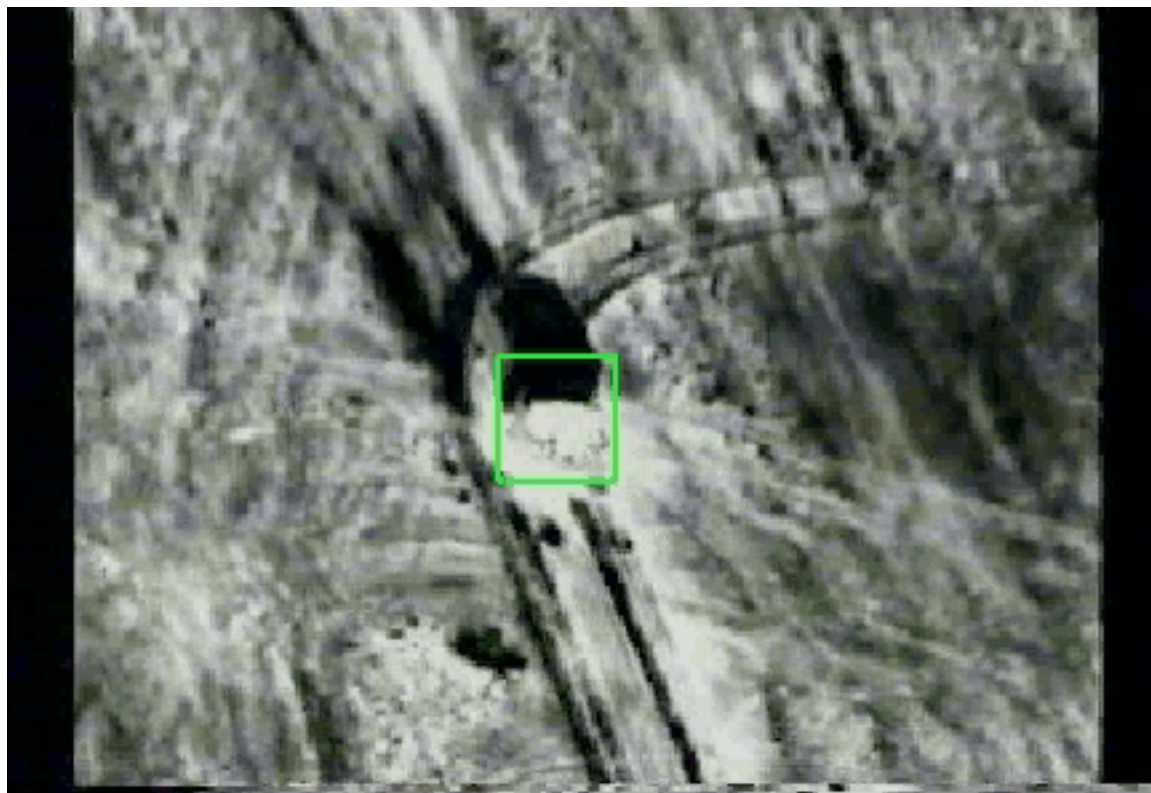
Murty K-best; PDAF; JPDAF

Part 4: Persistent Tracking

Adaptive Tracking

Tracking as Classification

What is Tracking?



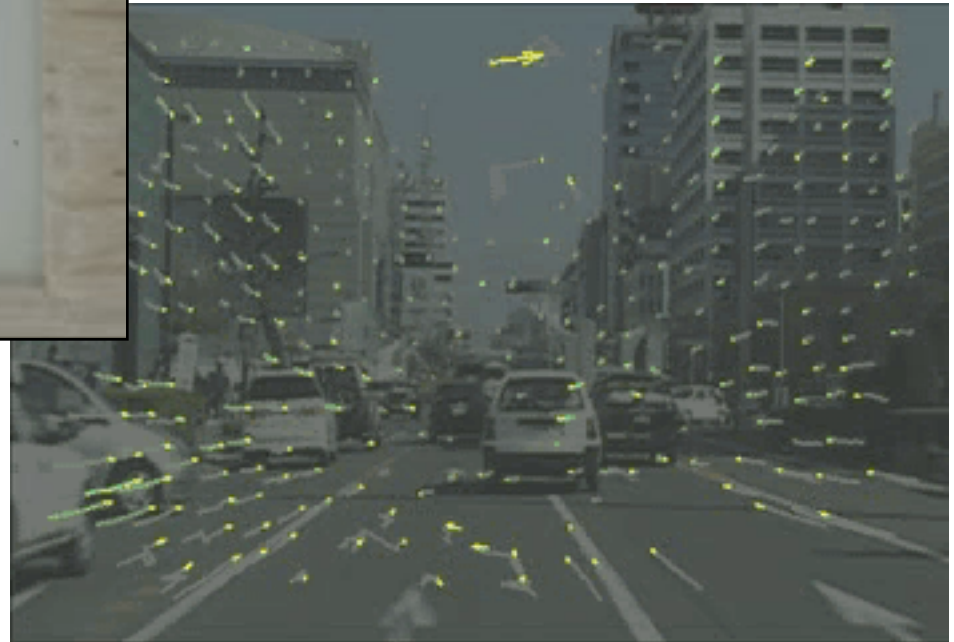
typical idea: tracking a single target in isolation.

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

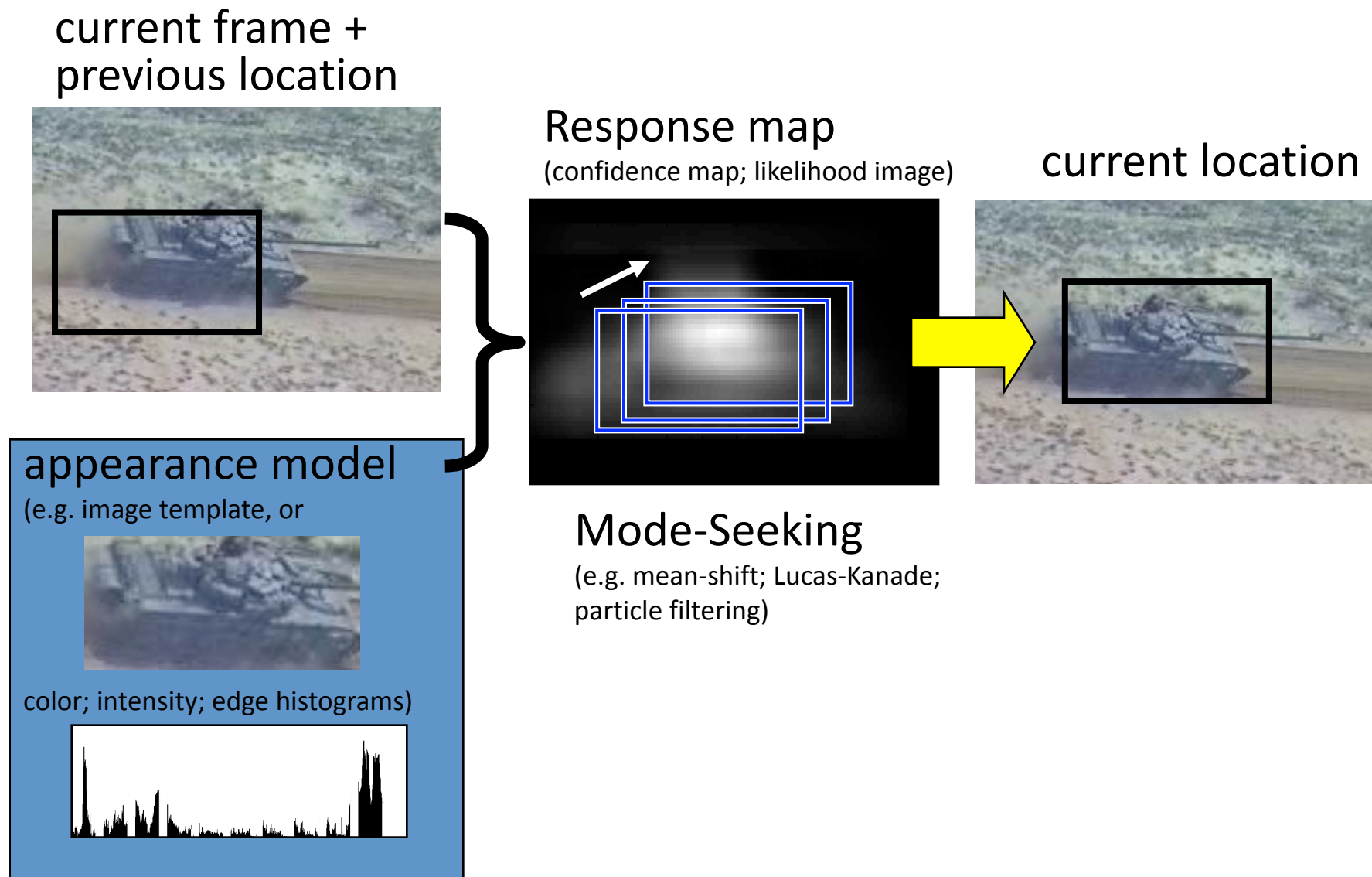


What is Tracking?

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



Appearance-Based Tracking



Relation to Data Association

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:

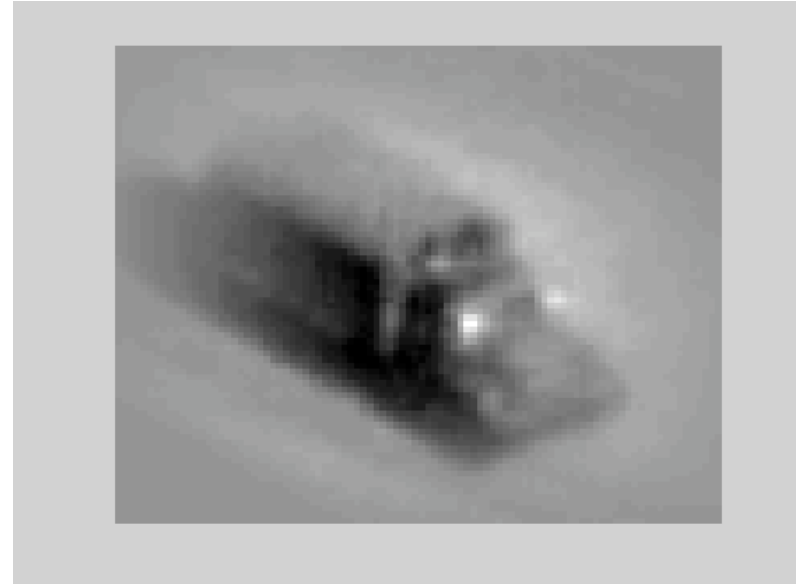
- a cropped image of the object from the first frame can be used to describe appearance
- object will look nearly identical in each new image (note: we can use normalized cross correlation to add some resilience to lighting changes.
- movement is nearly pure 2D translation

Normalized Correlation, Fixed Template

Current tracked location



Fixed template



Failure mode: Unmodeled Appearance Change

Naive Approach to Handle Change

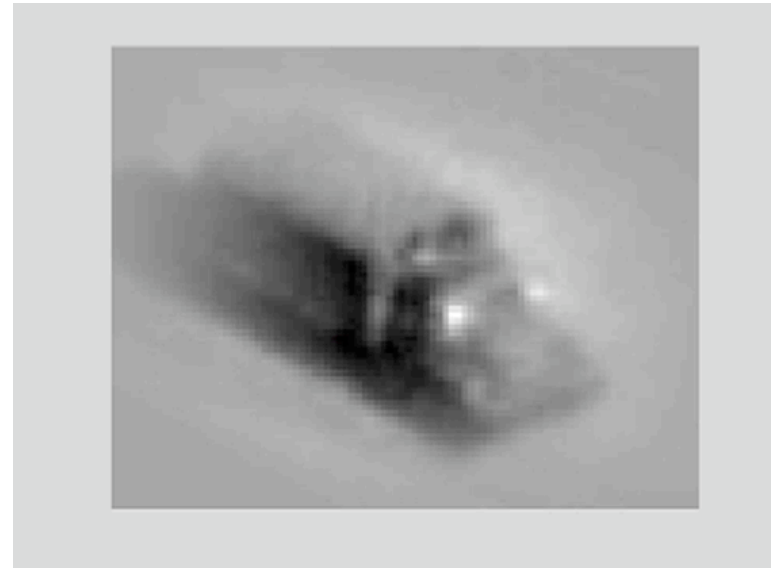
- One approach to handle changing appearance over time is adaptive template update
- One you find location of object in a new frame, just extract a new template, centered at that location
- What is the potential problem?

Normalized Correlation, Adaptive Template

Current tracked location

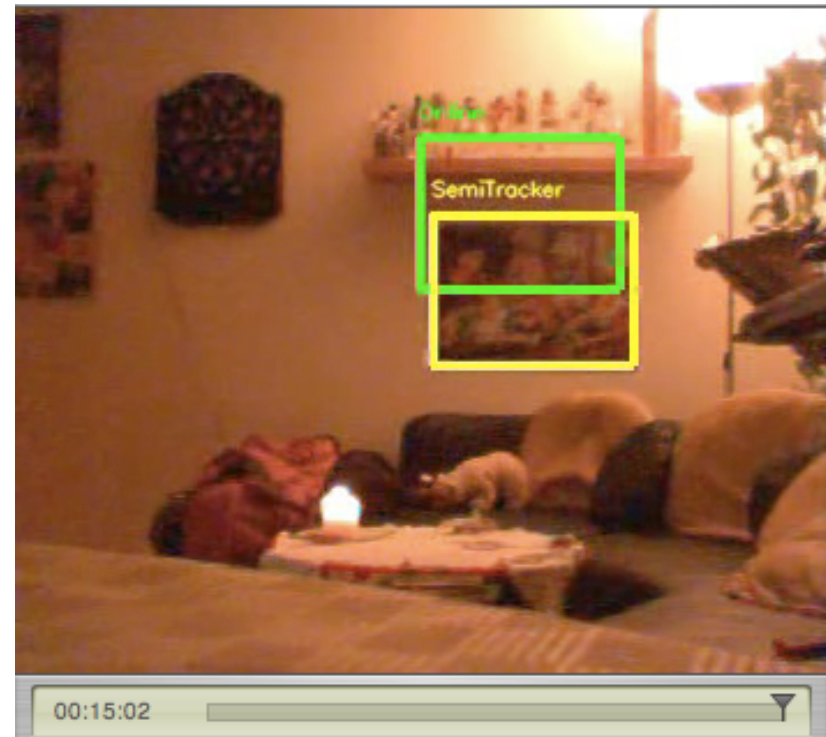
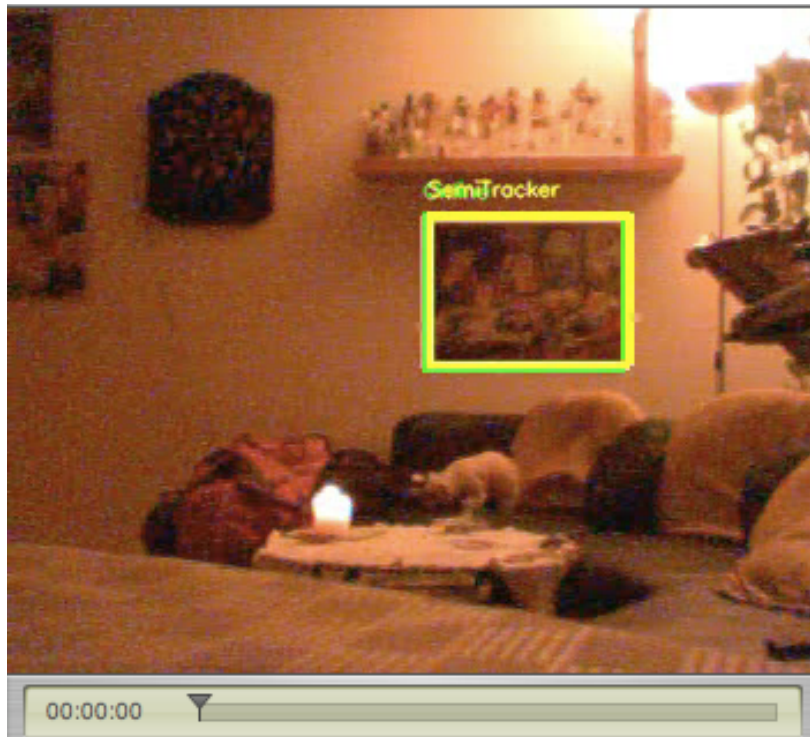


Current template



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.

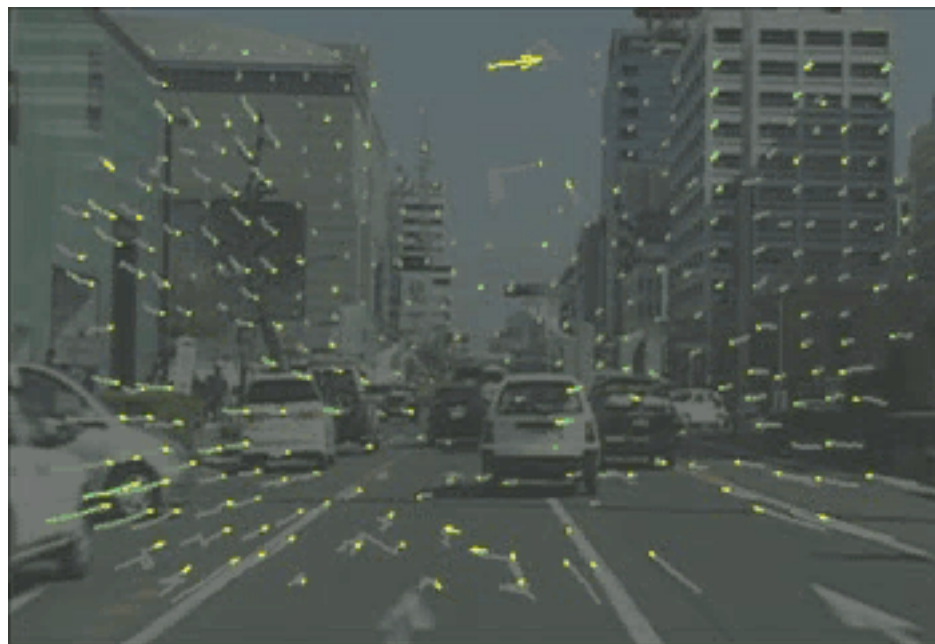
Lucas-Kanade Tracking

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.

movie

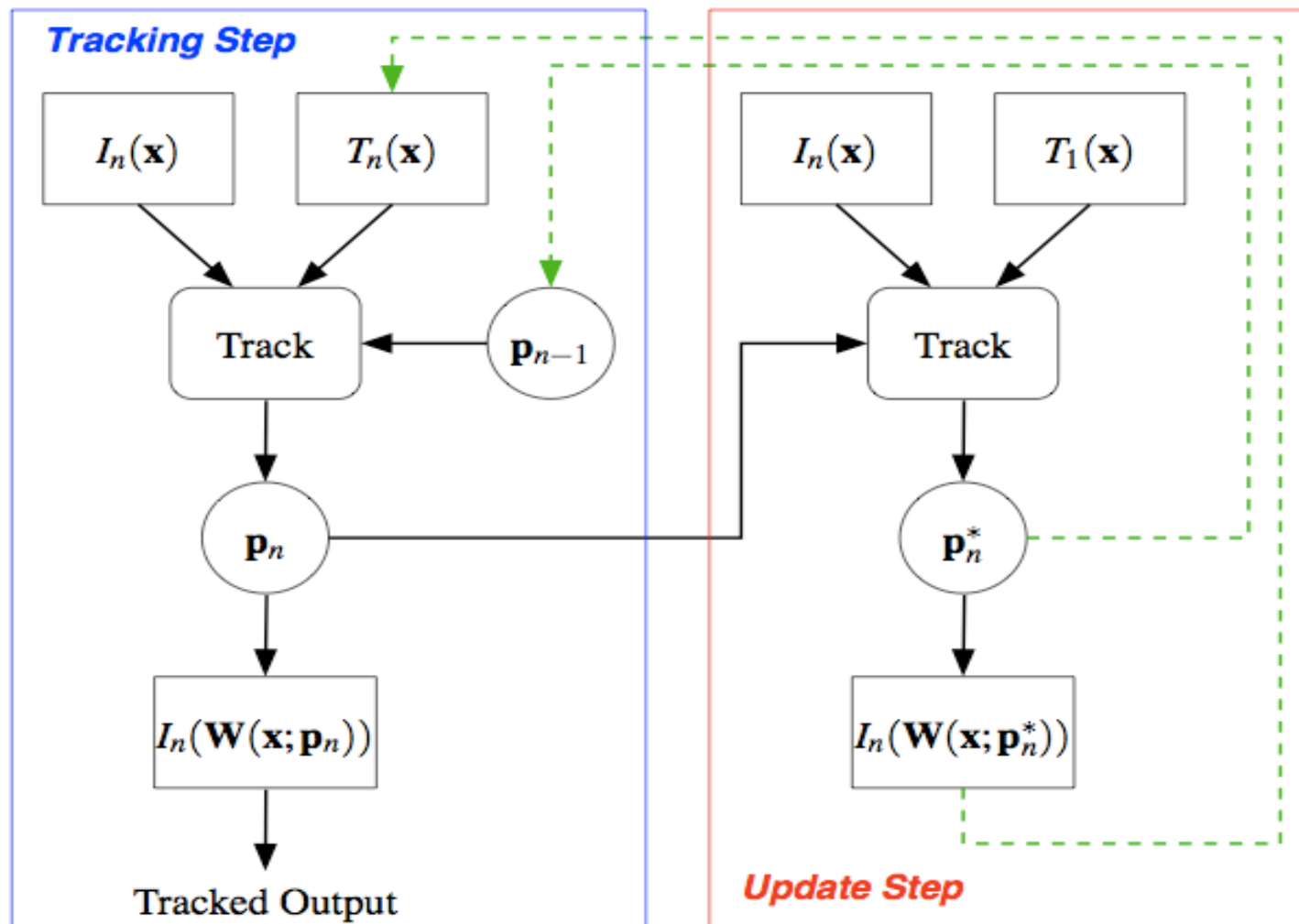


Fixed template

Naïve update

Their update

Template Update with Drift Correction



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

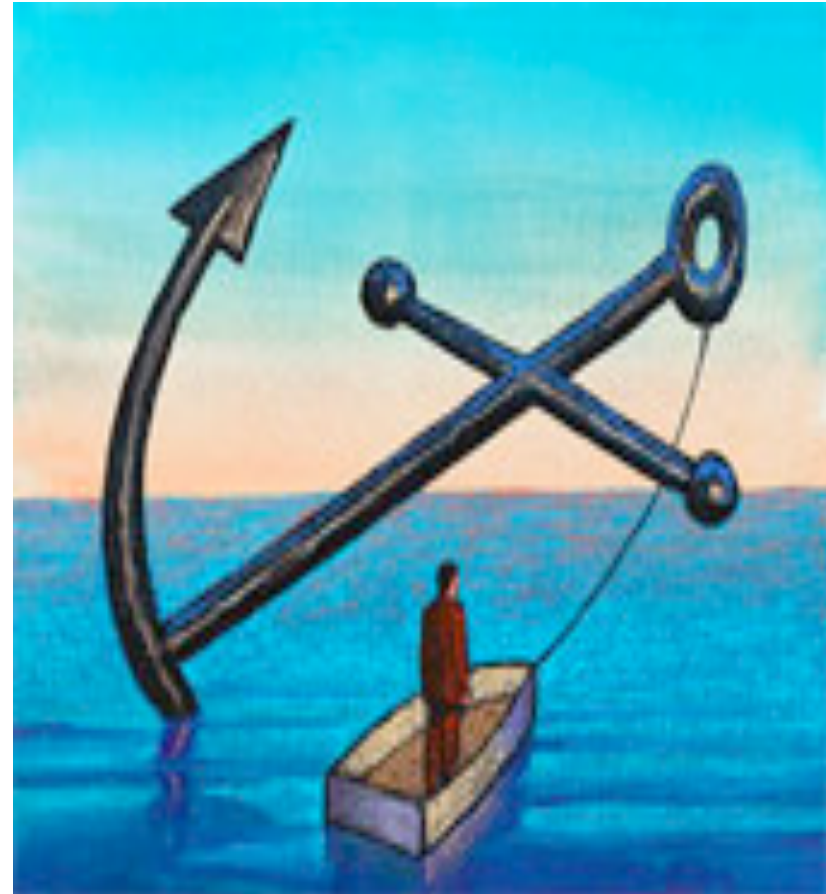
Anchoring Avoids Drift

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?

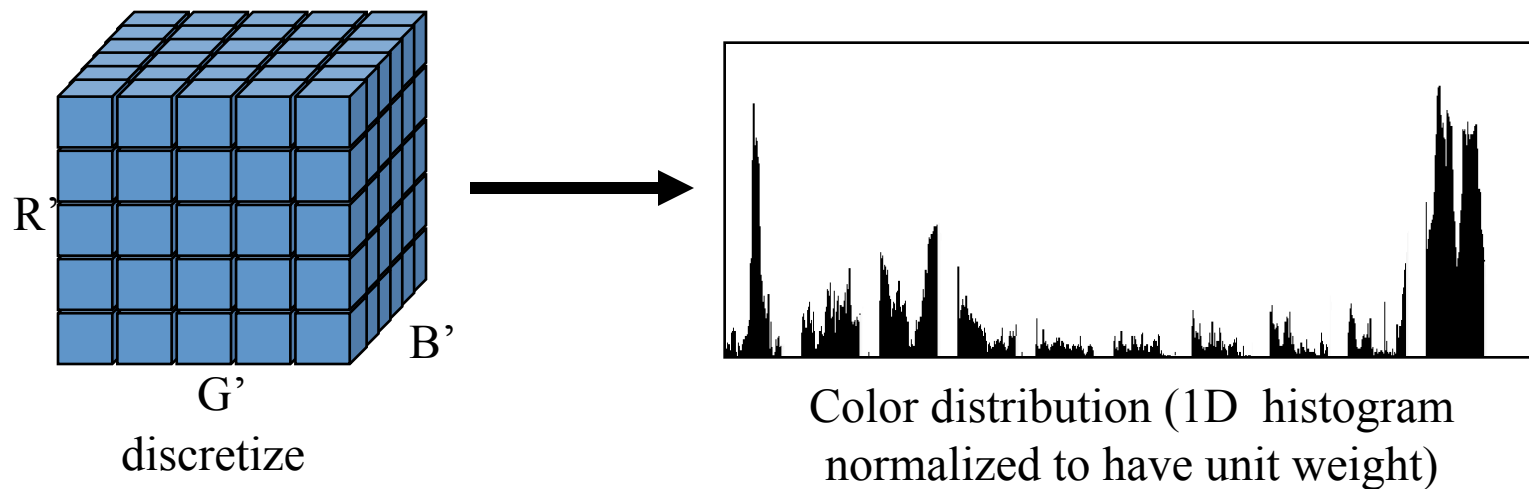
[answer: You cannot accommodate very LARGE changes in appearance.]



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



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

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

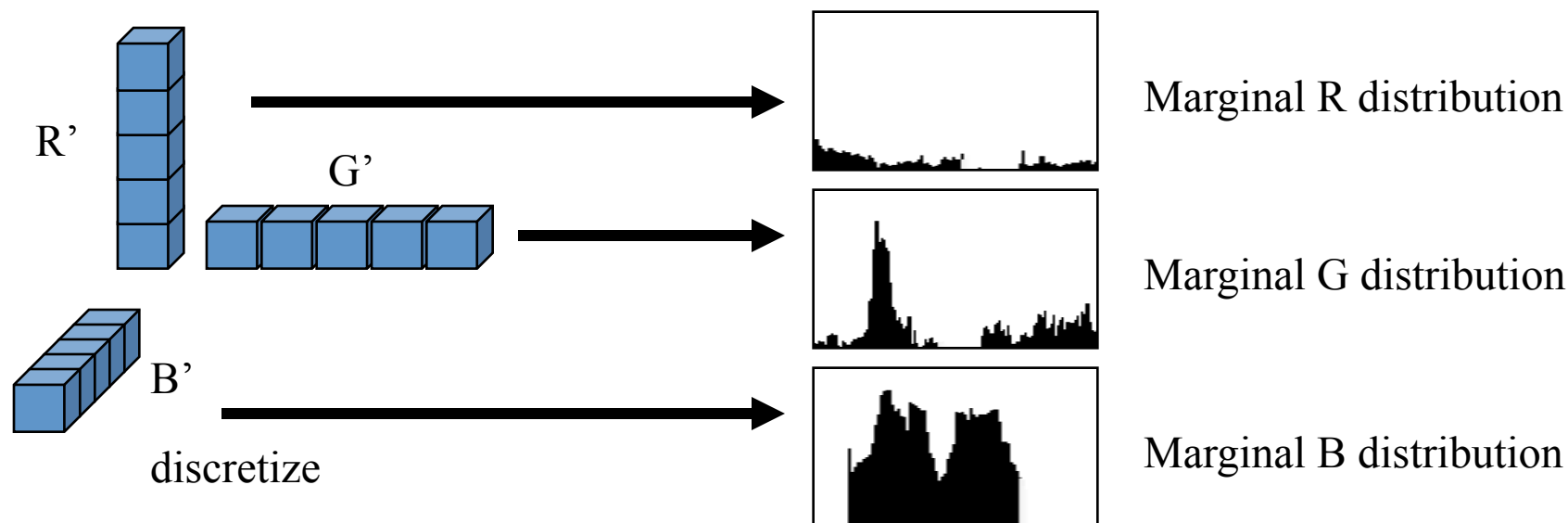
$$B' = B \ll (8 - \text{nbits})$$

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.



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

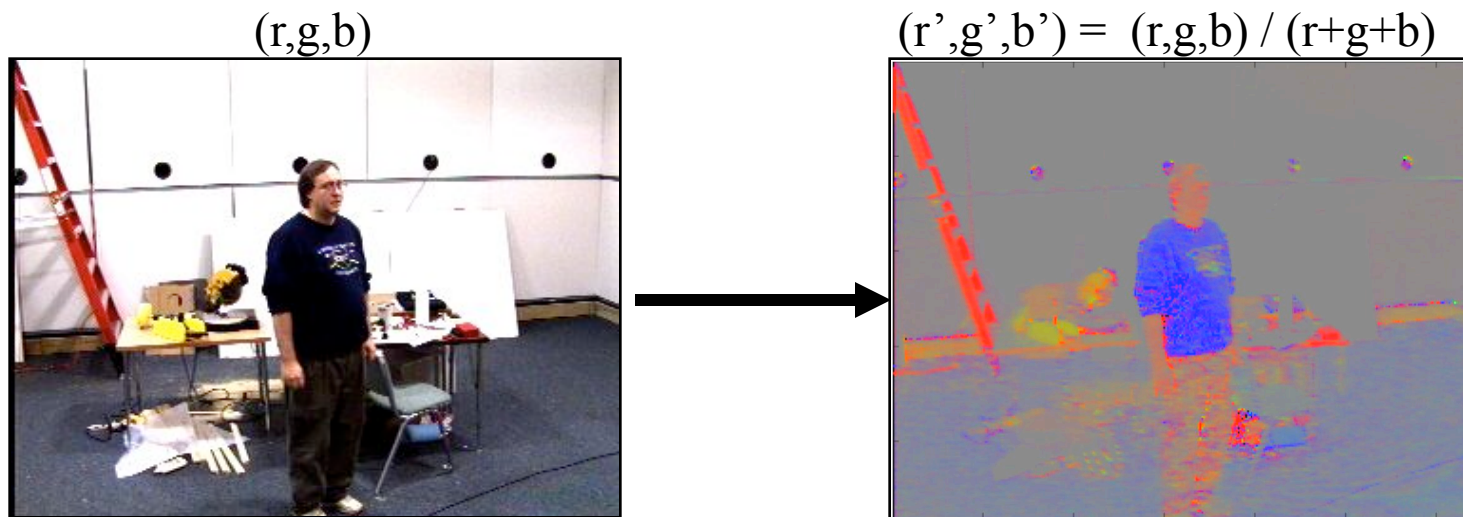
$$G' = G \ll (8 - \text{nbits})$$

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

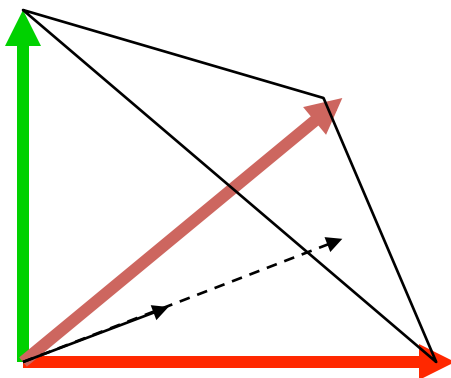
Total histogram size is $3 * (2^{(8-\text{nbits})})$

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

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.

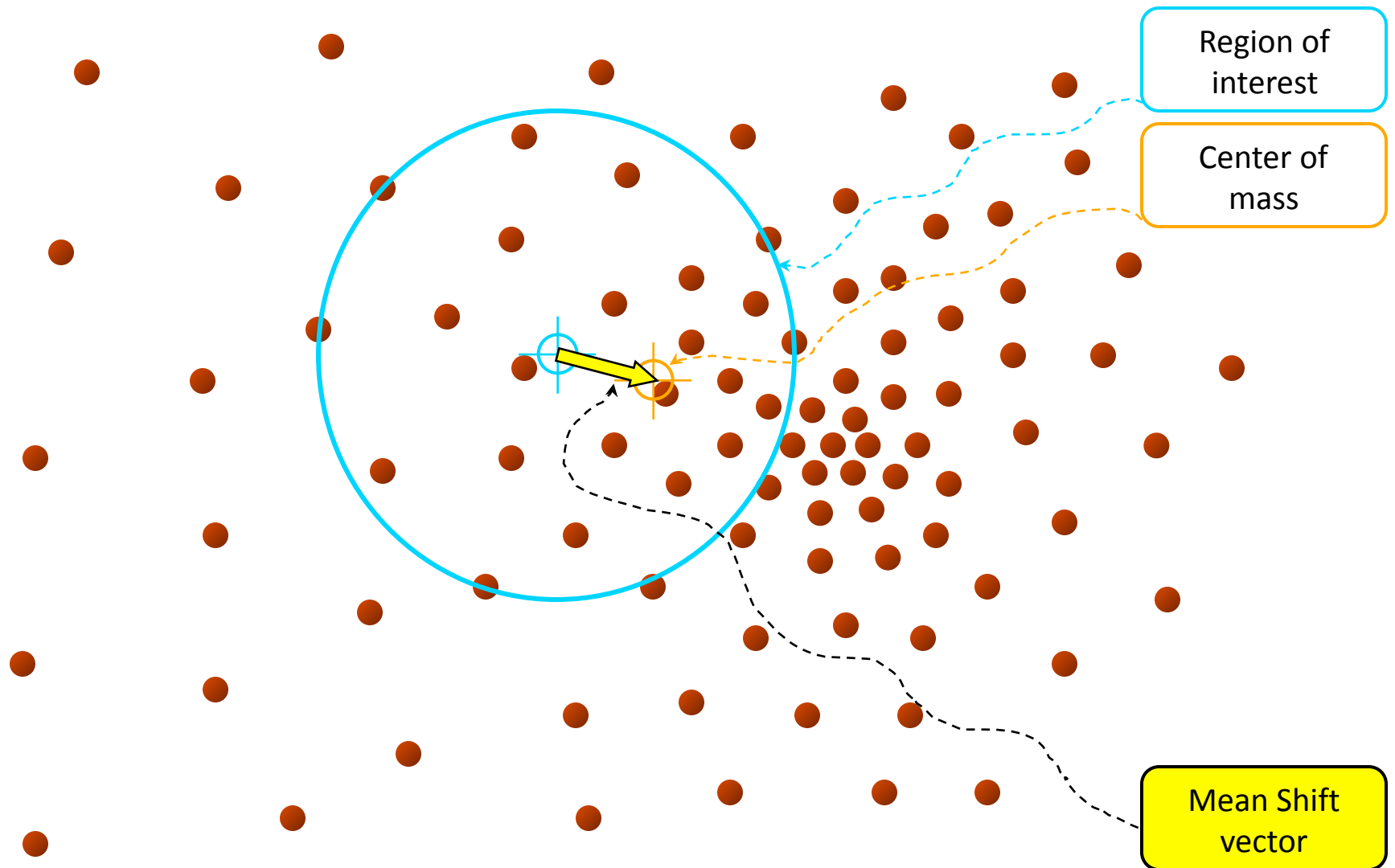


Mean-Shift

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

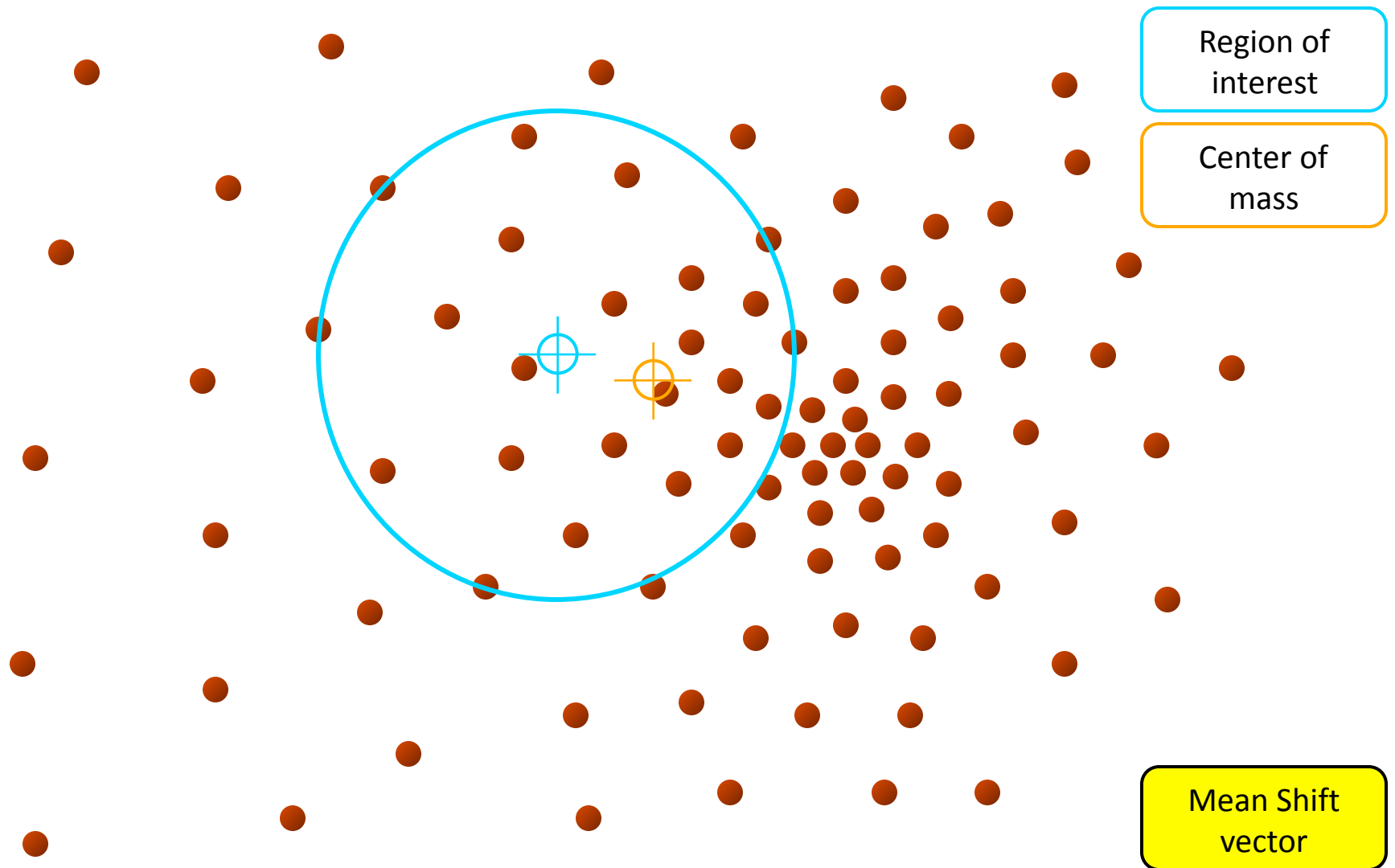
It is often used for tracking when a histogram-based appearance model is used. But it could be used just as well to search for modes in a template correlation surface.

Intuitive Description



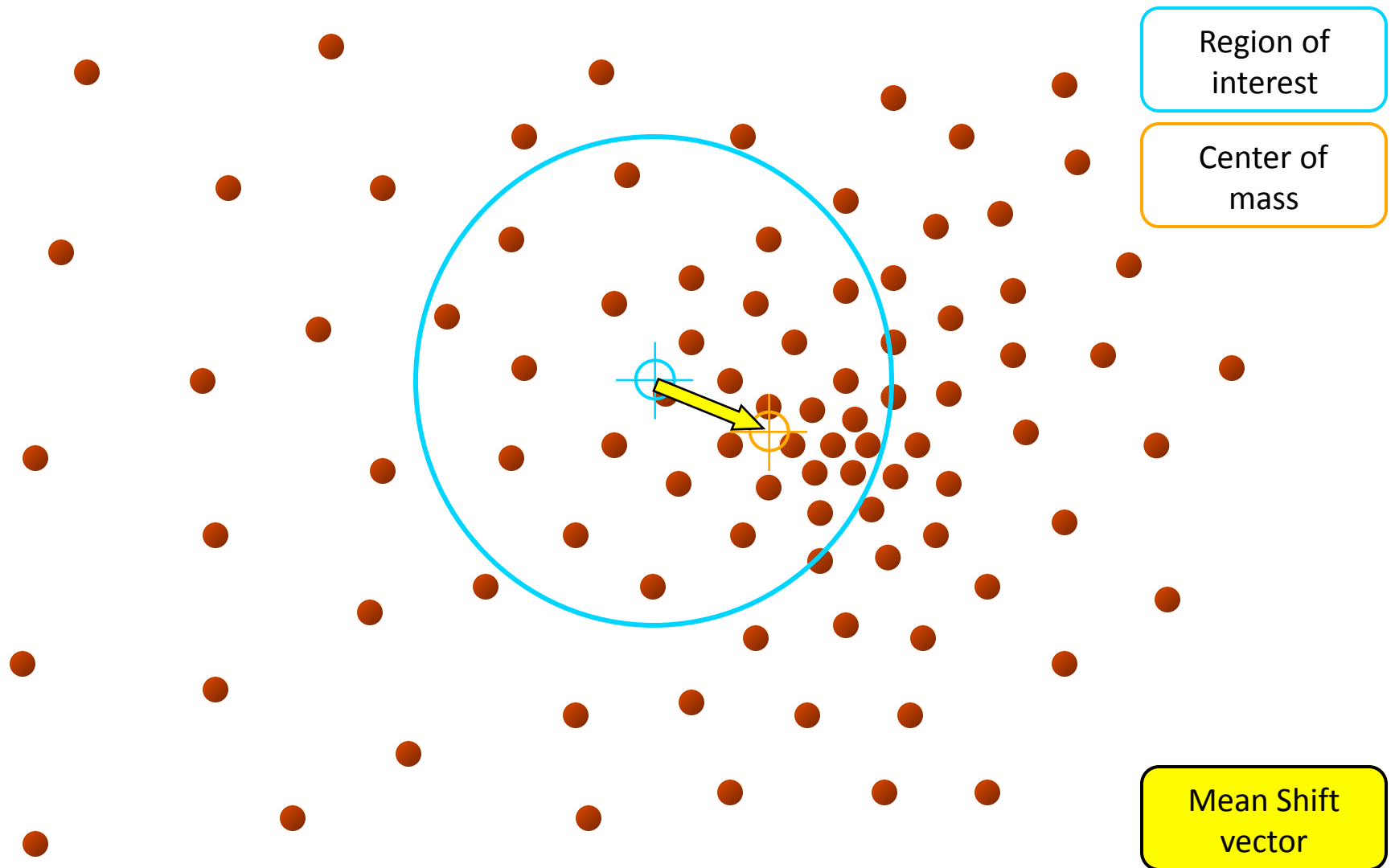
Objective : Find the densest region
Distribution of identical billiard balls

Intuitive Description



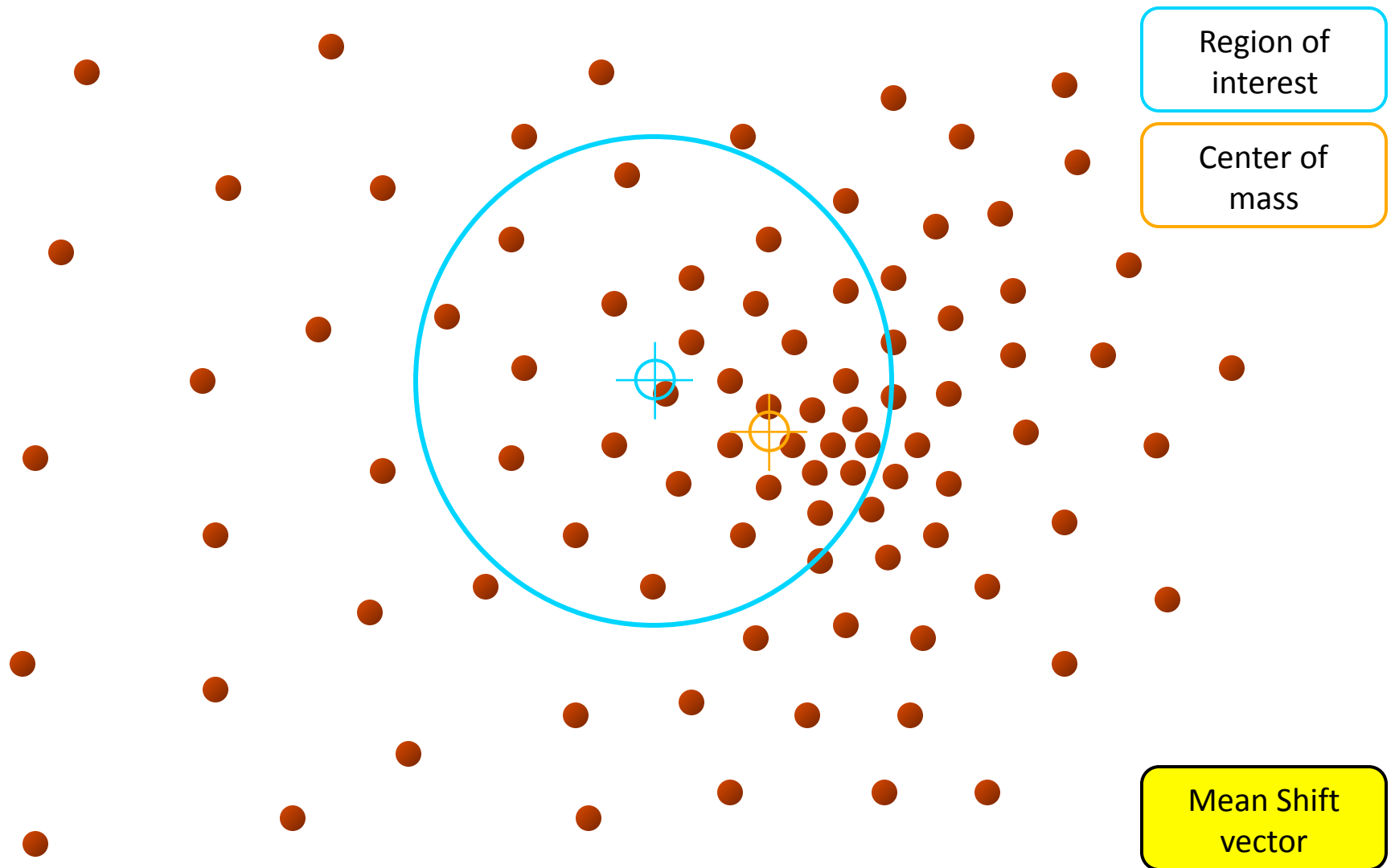
Objective : Find the densest region
Distribution of identical billiard balls

Intuitive Description



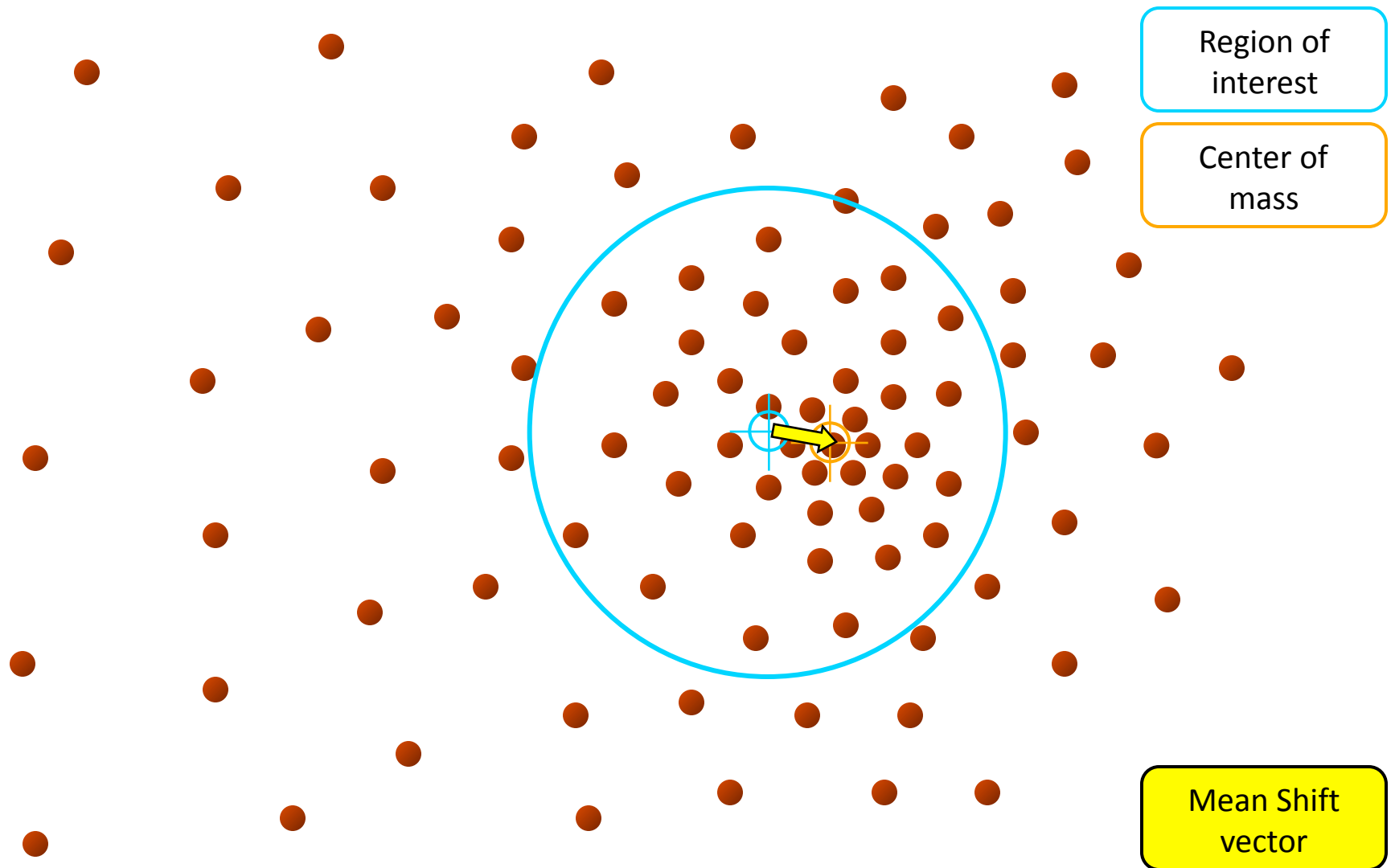
Objective : Find the densest region
Distribution of identical billiard balls

Intuitive Description



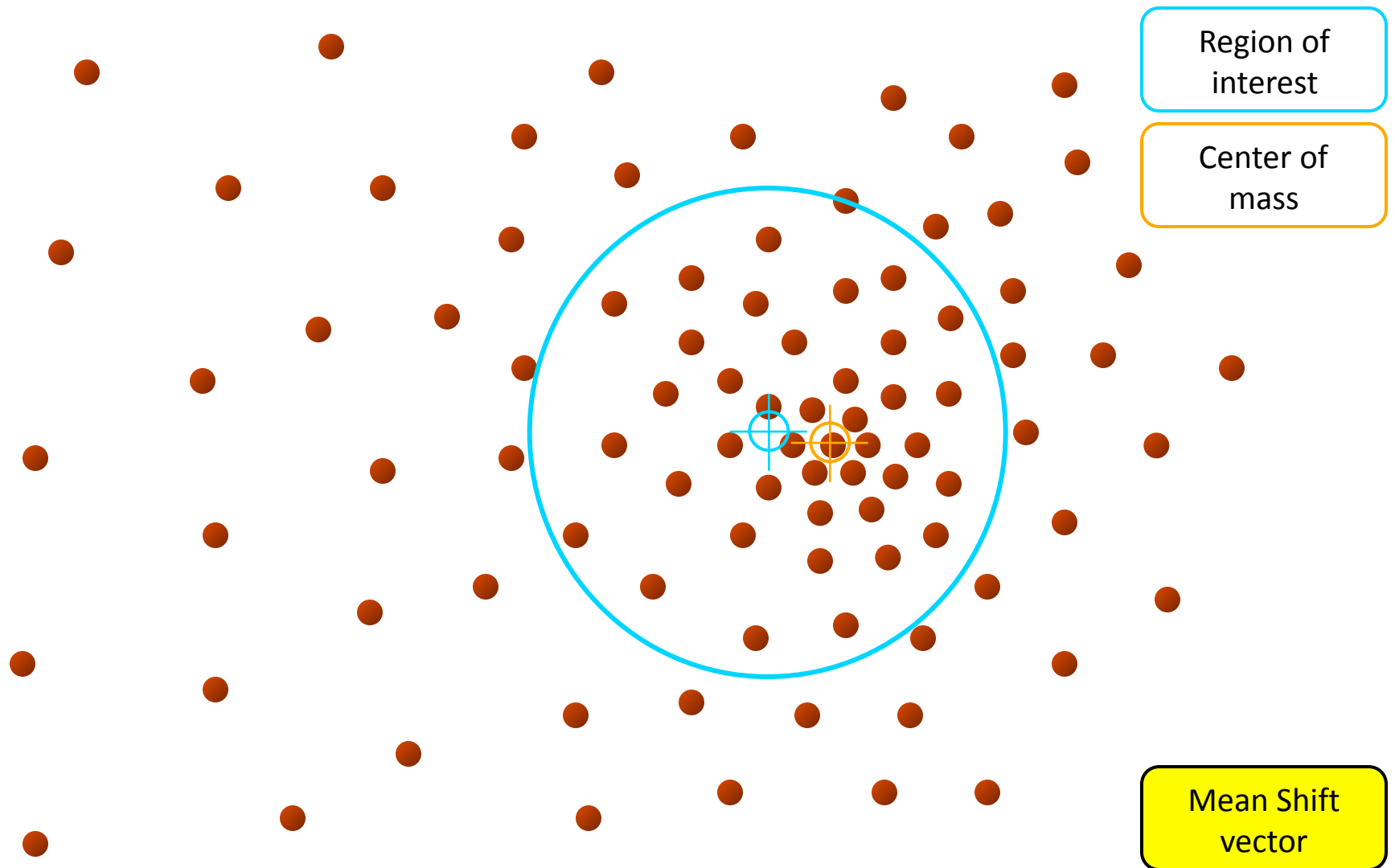
Objective : Find the densest region
Distribution of identical billiard balls

Intuitive Description



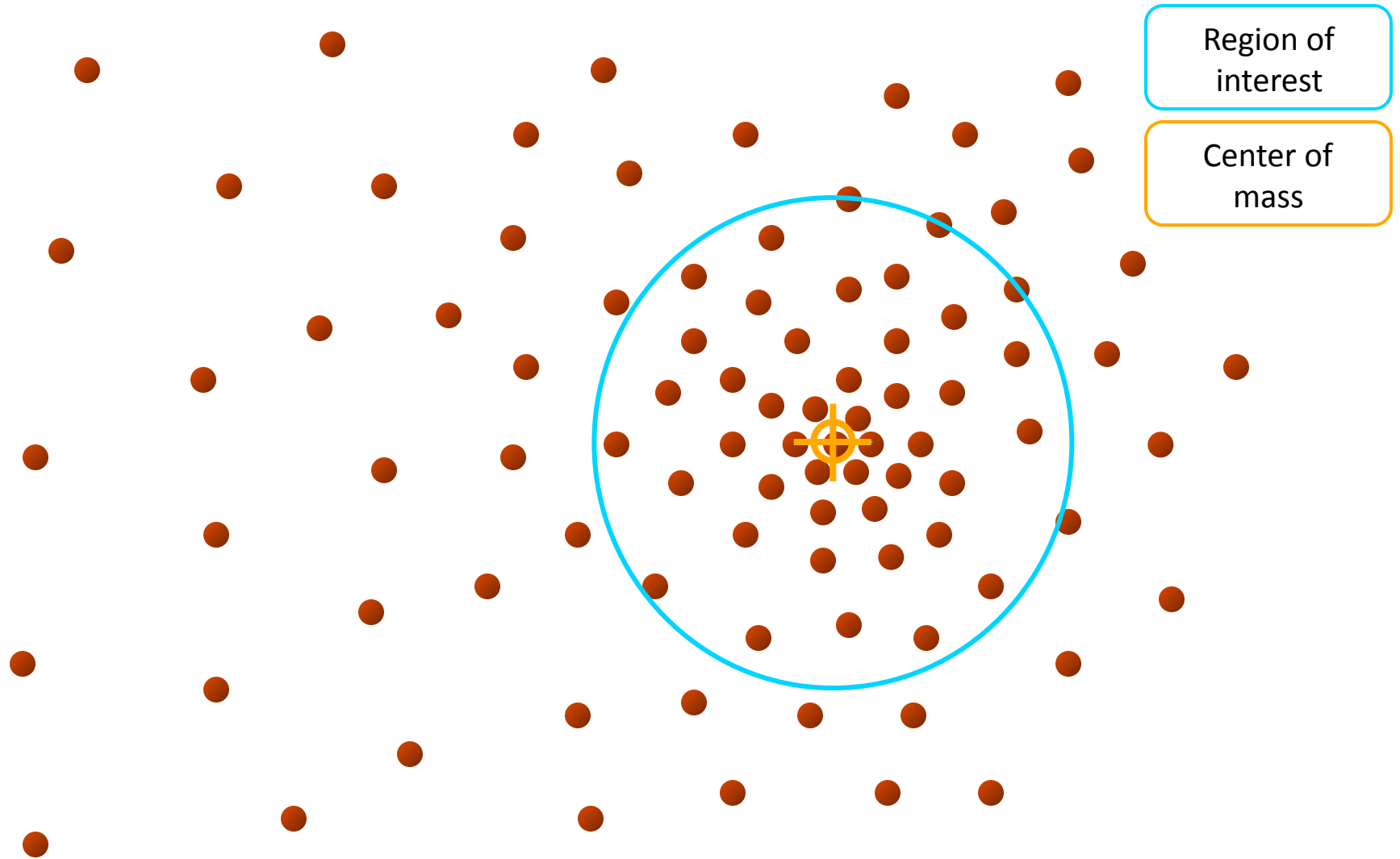
Objective : Find the densest region
Distribution of identical billiard balls

Intuitive Description



Objective : Find the densest region
Distribution of identical billiard balls

Intuitive Description



Objective : Find the densest region
Distribution of identical billiard balls

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.

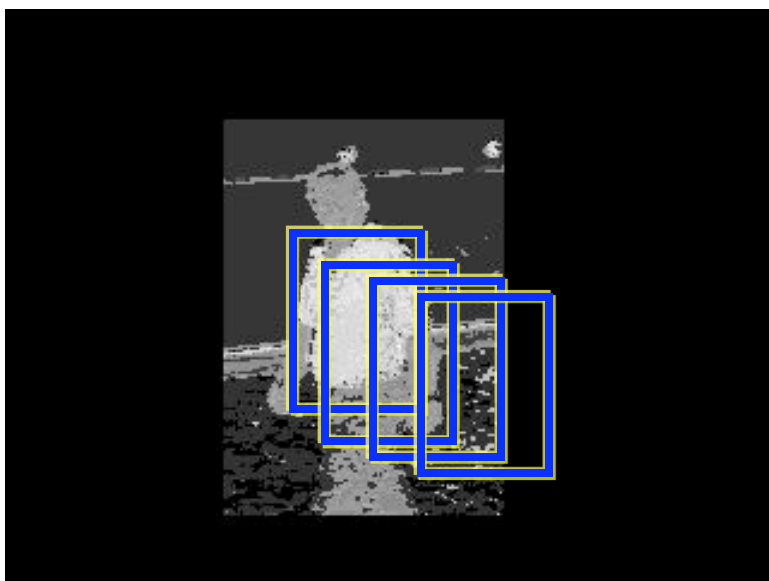
Computation of likelihood can be based on

- 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.

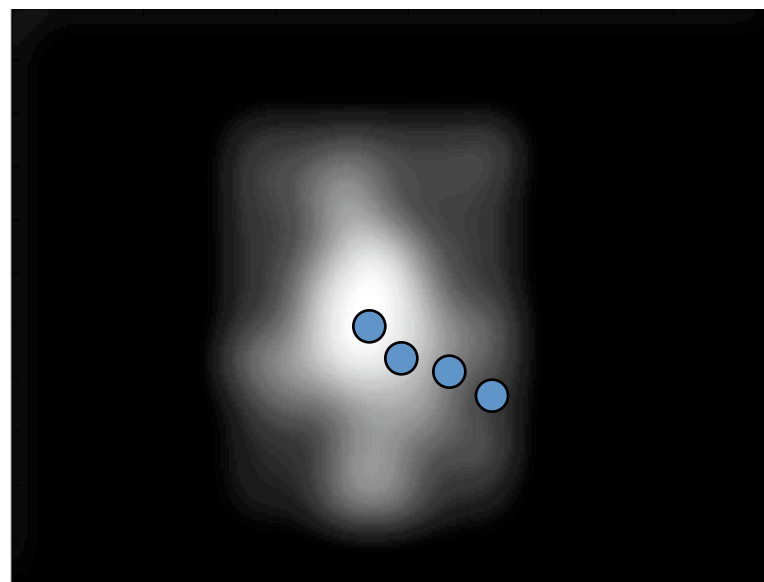
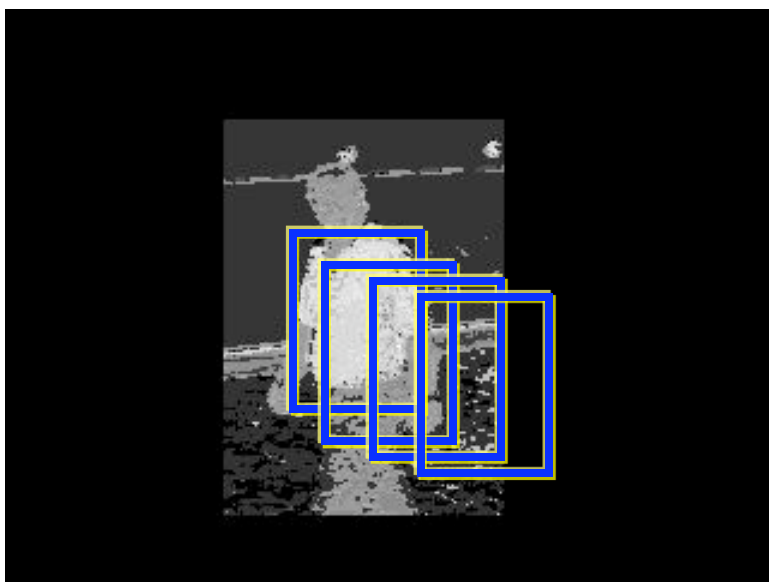


$$\Delta \mathbf{x} = \frac{\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x}) w(\mathbf{a}) (\mathbf{a}-\mathbf{x})}{\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x}) w(\mathbf{a})}$$

\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 .



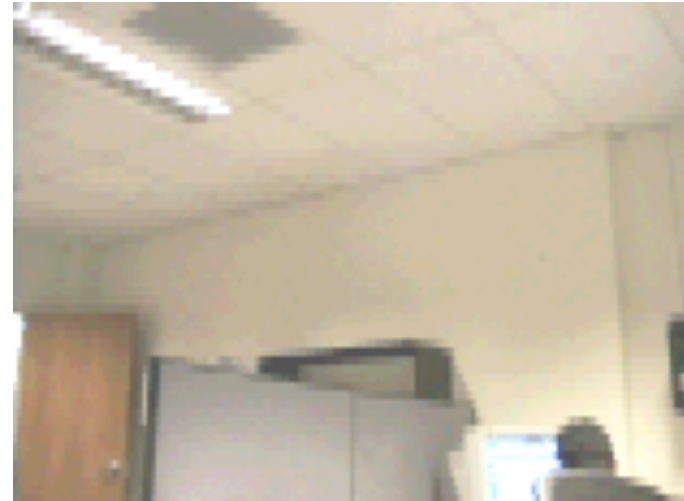
The algorithm is performing hill-climbing on an implicit density function determined by Parzen estimation with kernel H .

Mean-Shift Tracking

Some examples.



Gary Bradski, CAMSHIFT



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

Mean-Shift Tracking

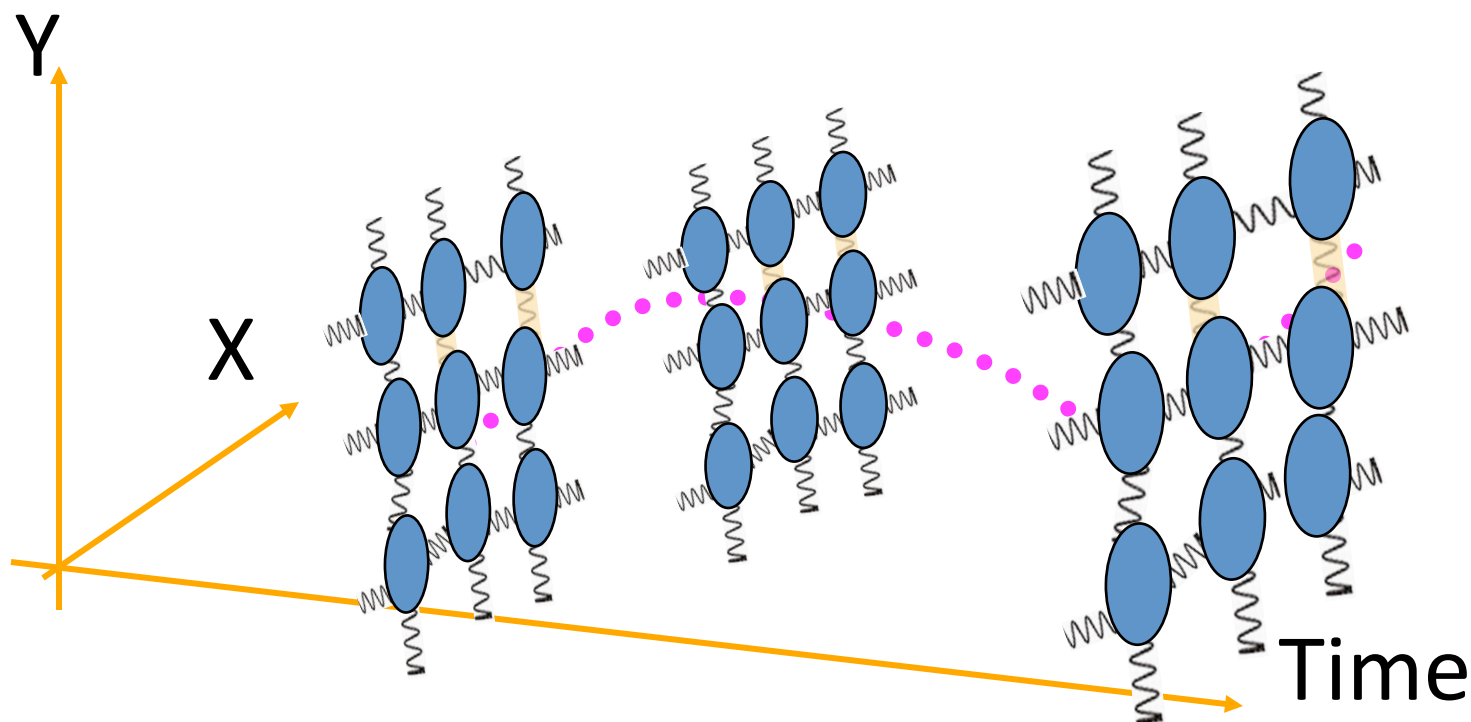
Using mean-shift in real-time to control a pan/tilt camera.



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

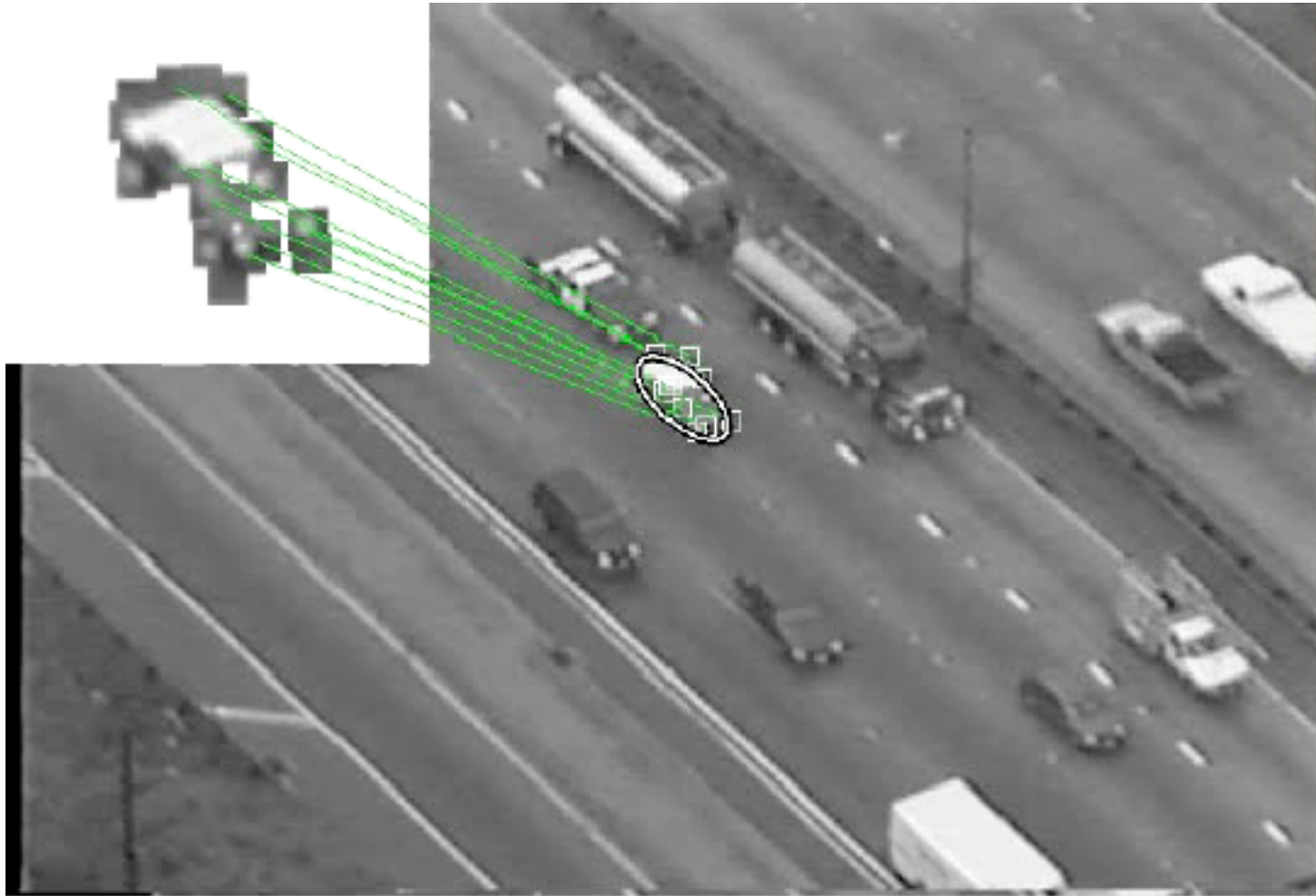
Constellations of Patches

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



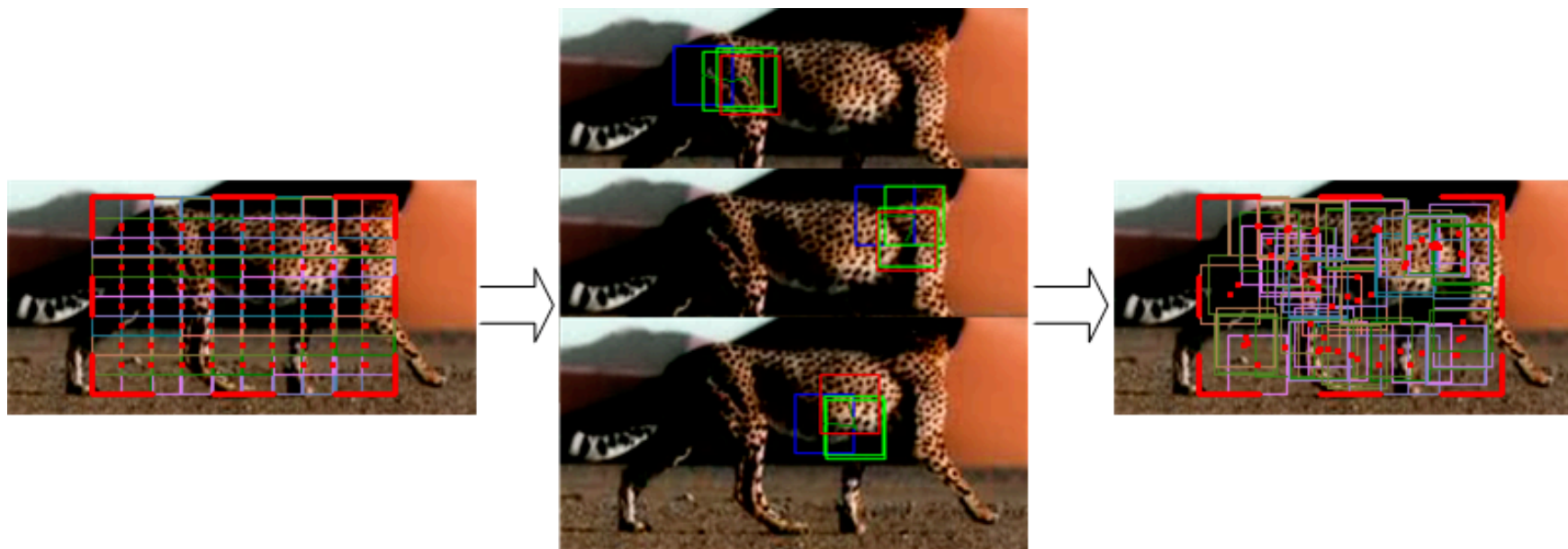
Example: Corner Patch Model

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



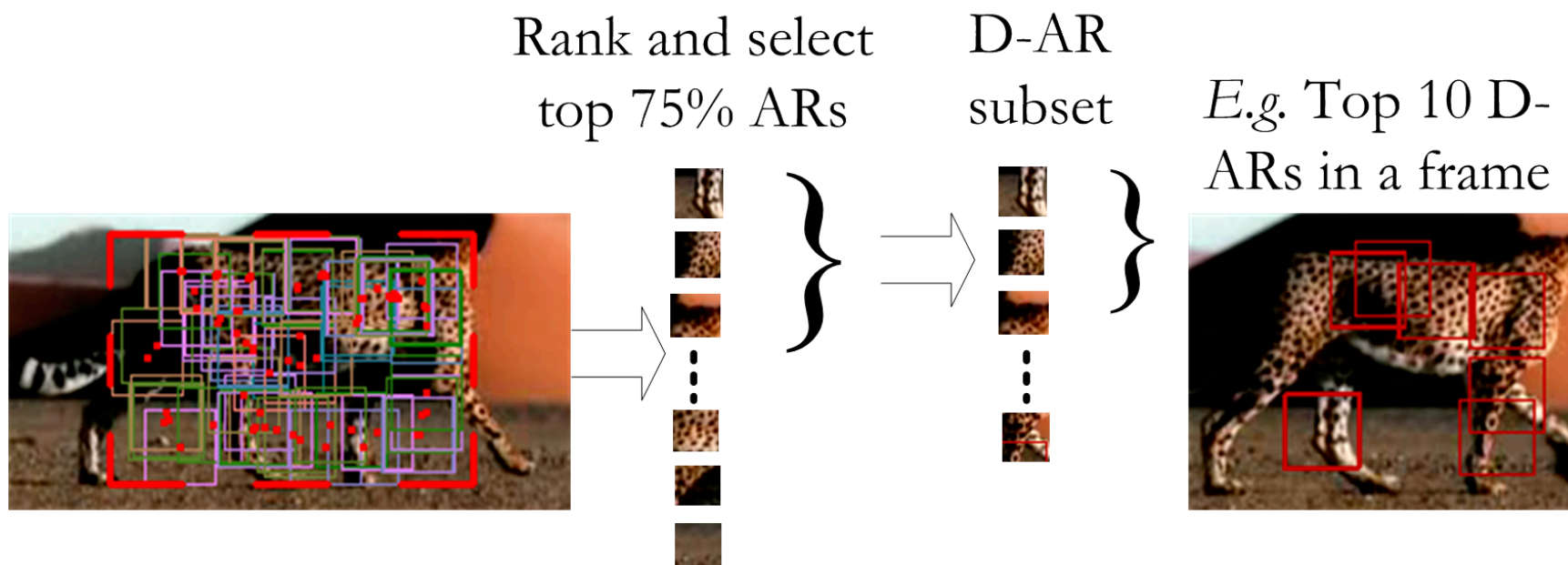
Example: Attentional Regions

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.

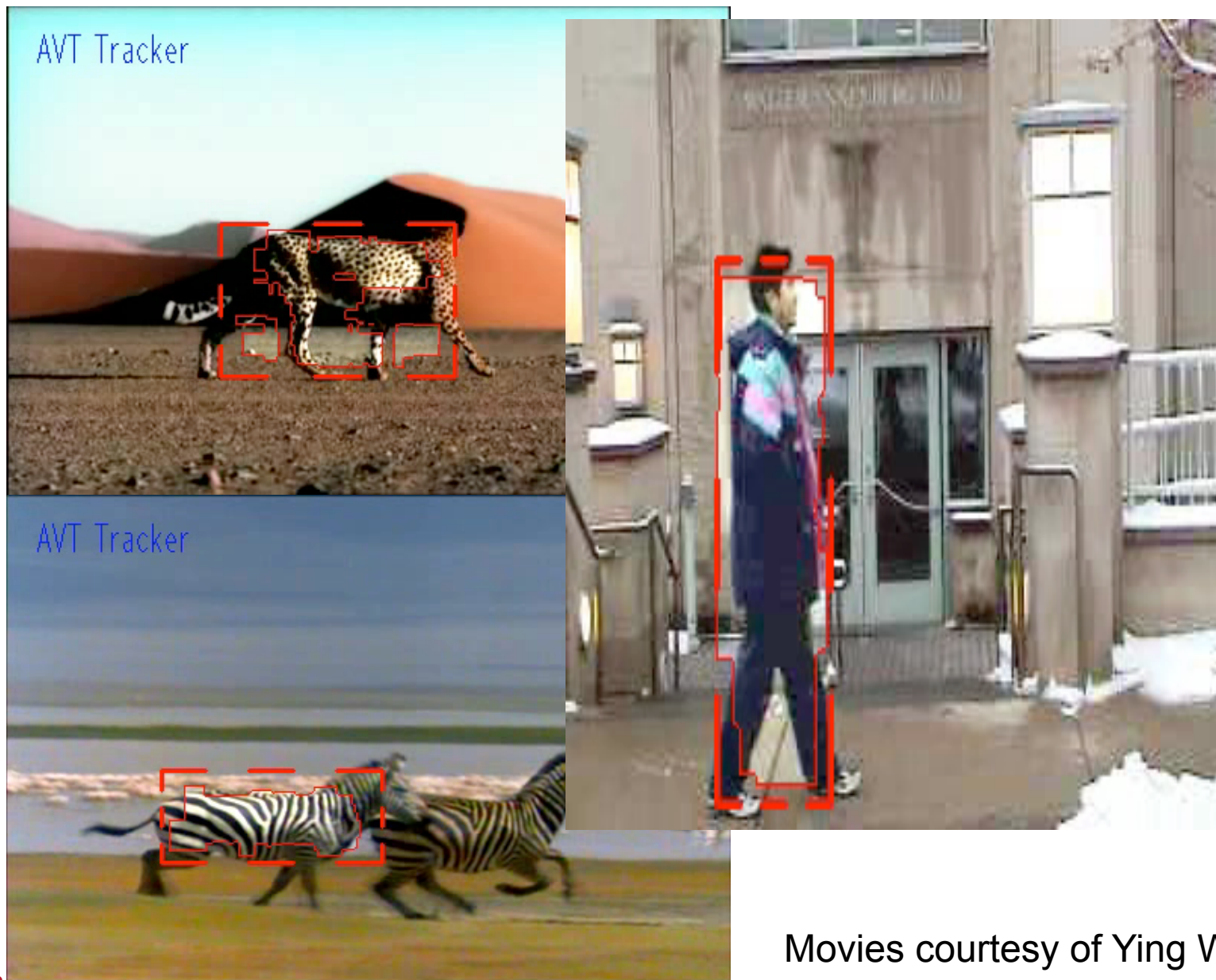
Example: Attentional Regions



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)

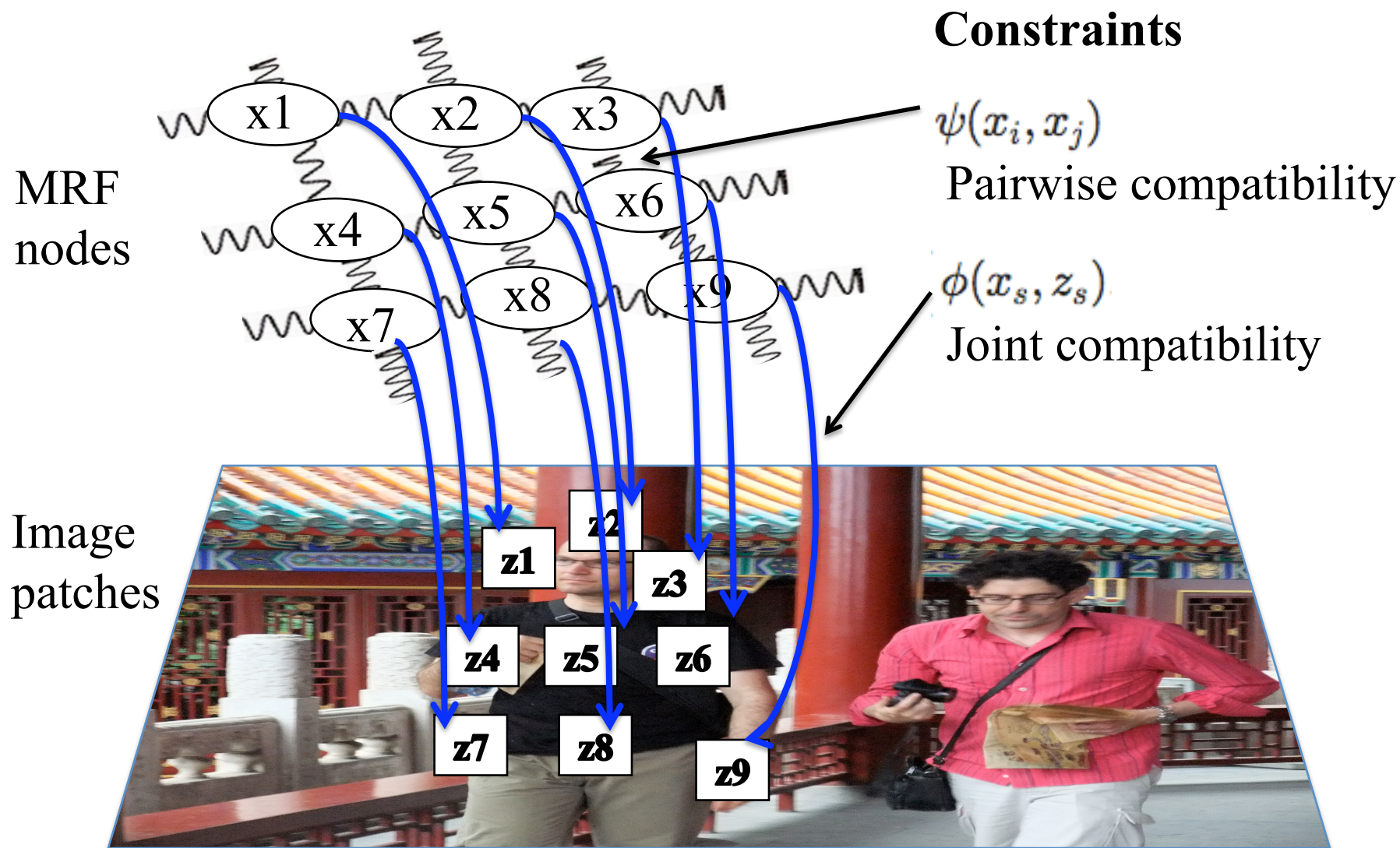
Example: Attentional Regions



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



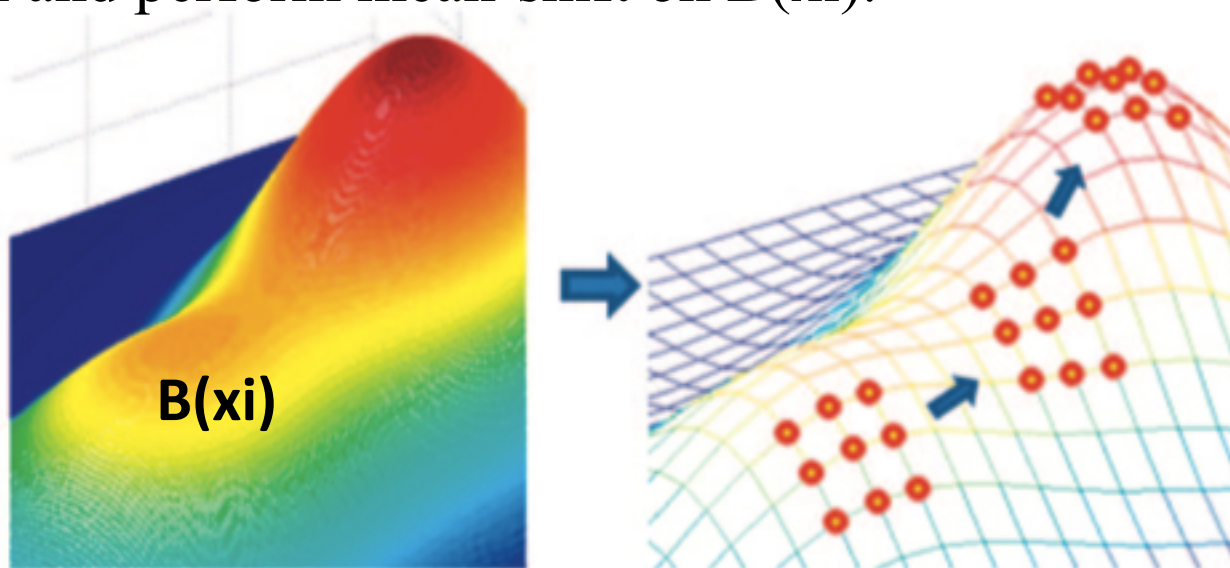
Mean-Shift Belief Propagation

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.

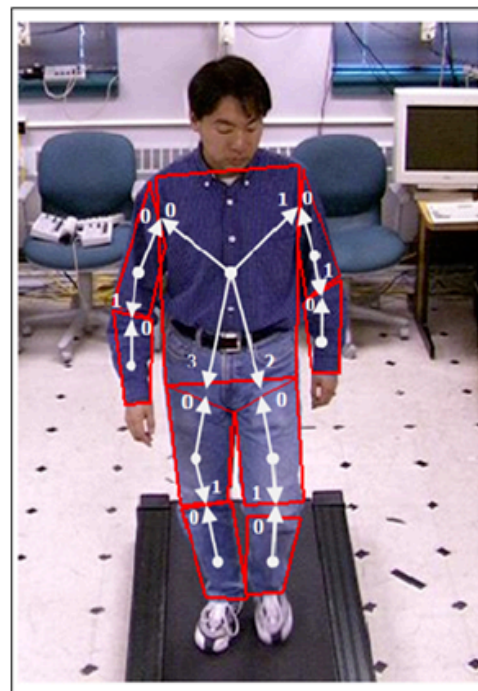
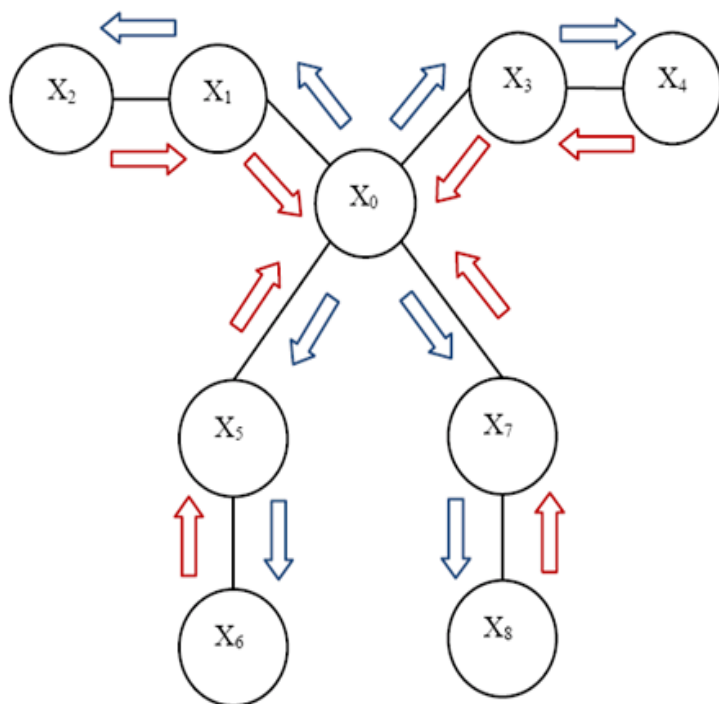
$$p(x_1, \dots, x_N, z_1, \dots, z_N) = k \prod_{(i,j)} \psi(x_i, x_j) \prod_s \phi(x_s, z_s)$$

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

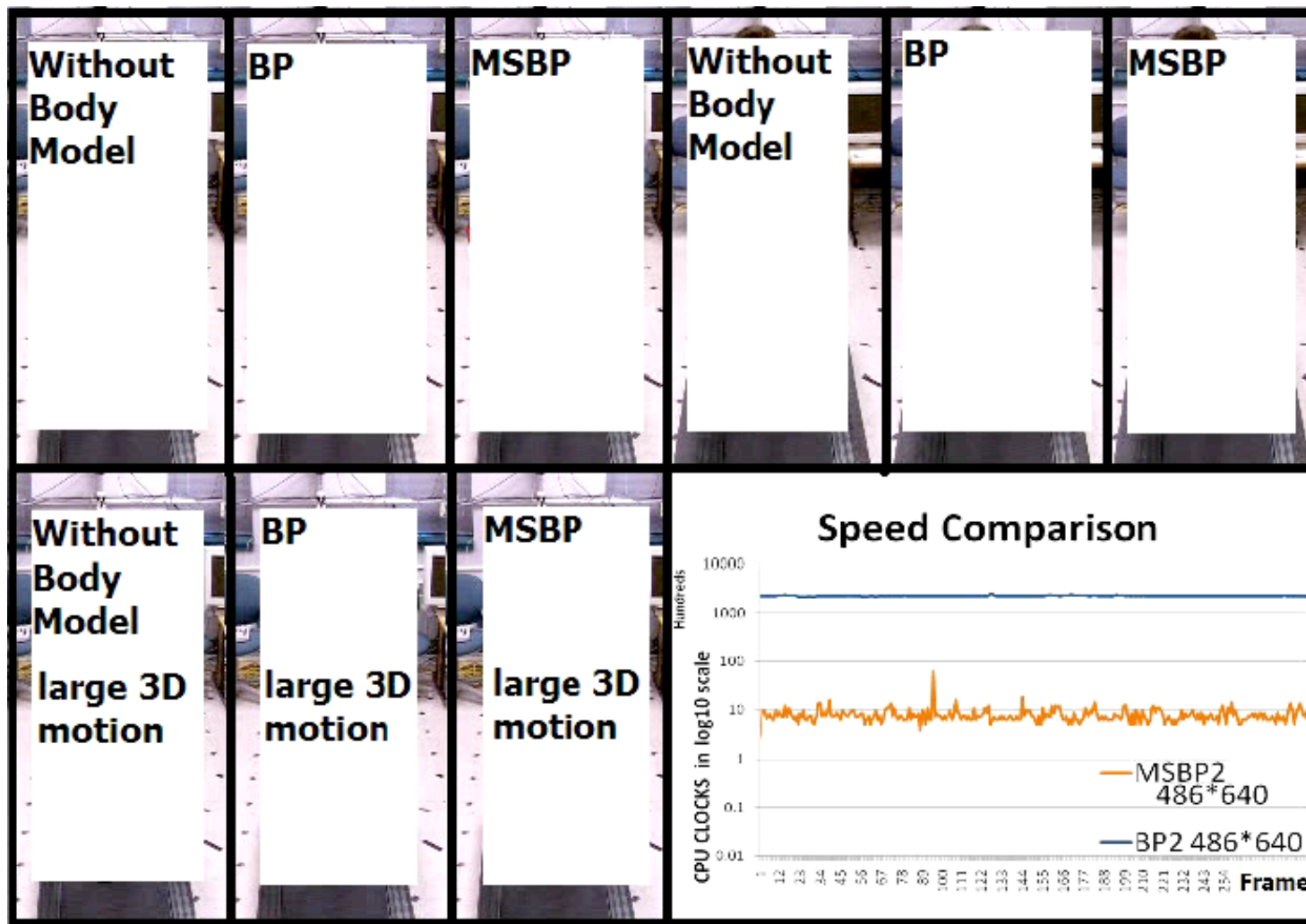


Example: Articulated Body Tracking

- 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, θ), representing 2 dimensional translation (x, y) and in-plane rotation θ



Articulated Body Tracking



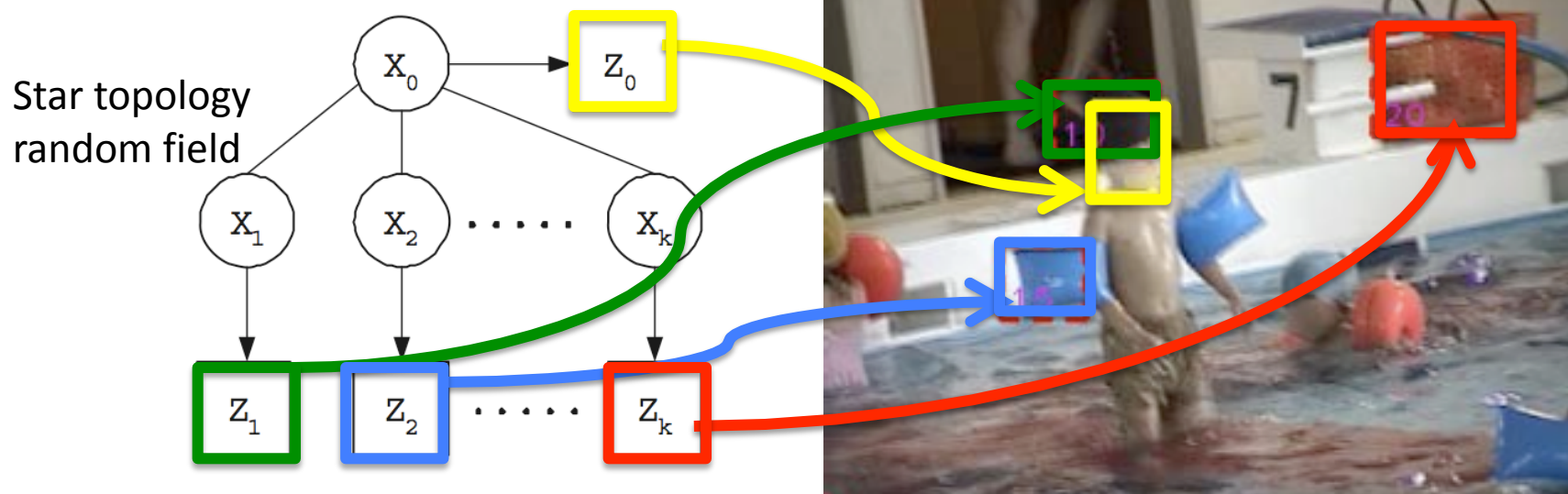
Limitations. If the viewpoint changes too much, this 2D graph tracker will fail. But the idea is that we also are running the body pose detector at the same time. The detector can this “guide” the tracker, and also reinitialize the tracker after failure.

Example: Auxiliary Objects

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

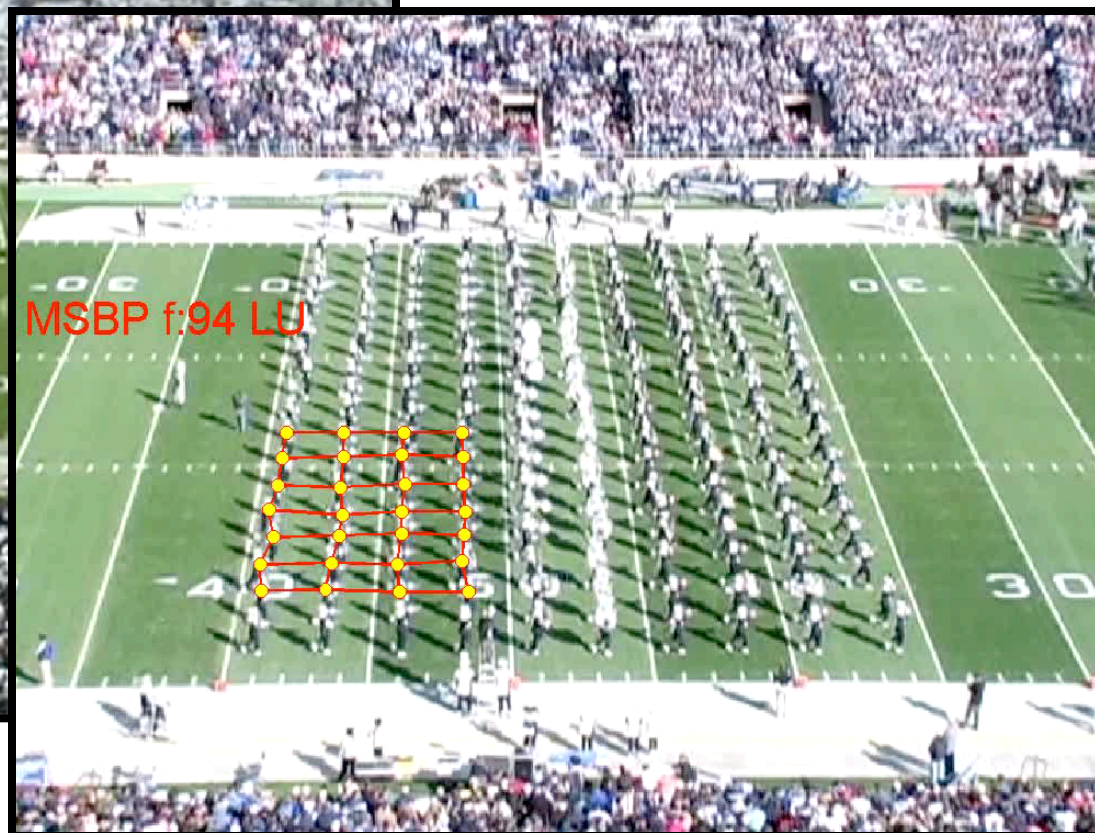
Look for auxiliary regions in the image that:

- 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.

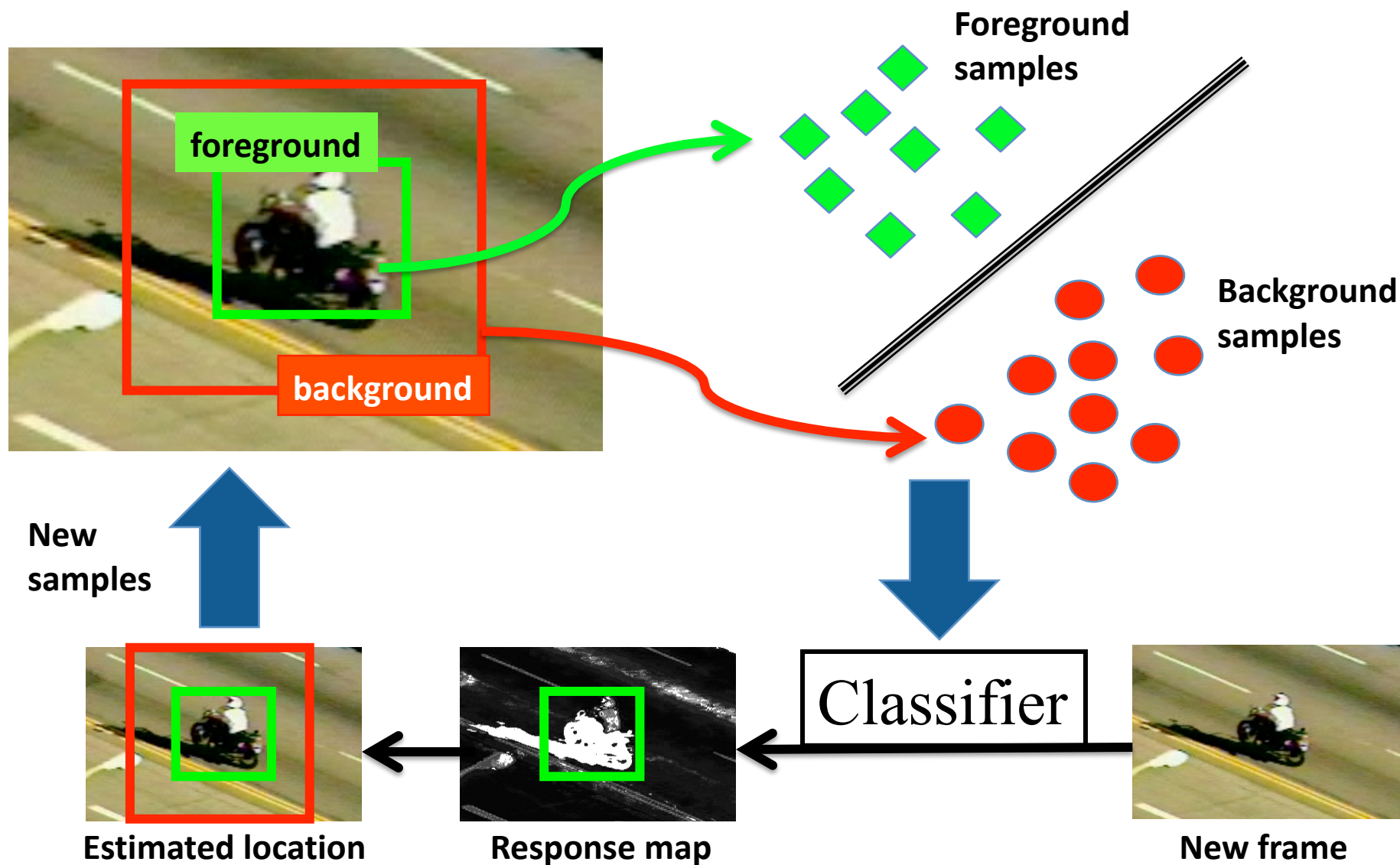
Online adaptation of the appearance model or the features used allows the representation to have retain good specificity at each time frame while evolving to have overall generality to large variations in object/background/lighting appearance.

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

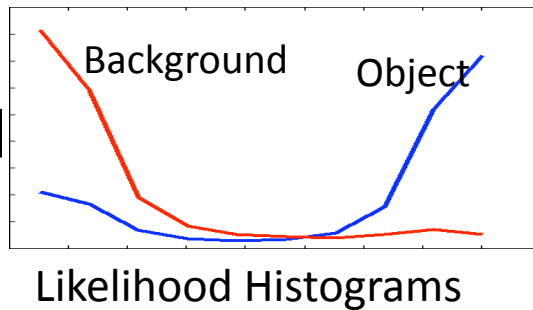
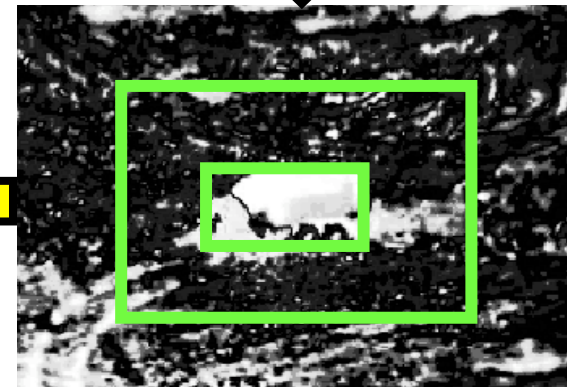
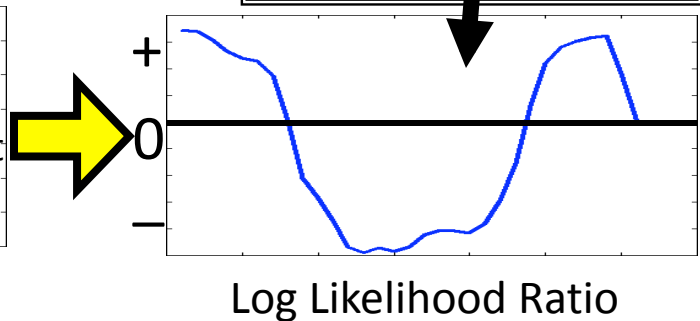
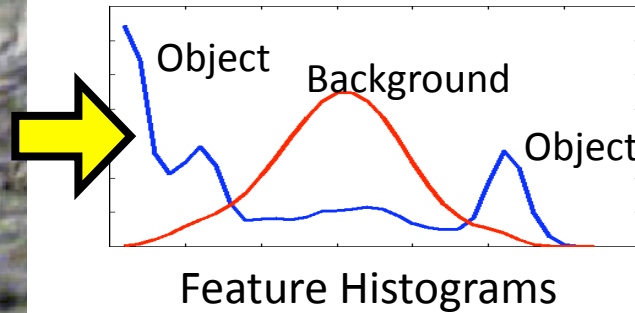
- Forward Selection
- Backward Selection
- Branch and Bound

Viola and Jones, Cascaded Feature Selection for Classification

Bottom Line: slow, off-line process

Evaluation of Feature Discrimination

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



Variance Ratio
(feature score)

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

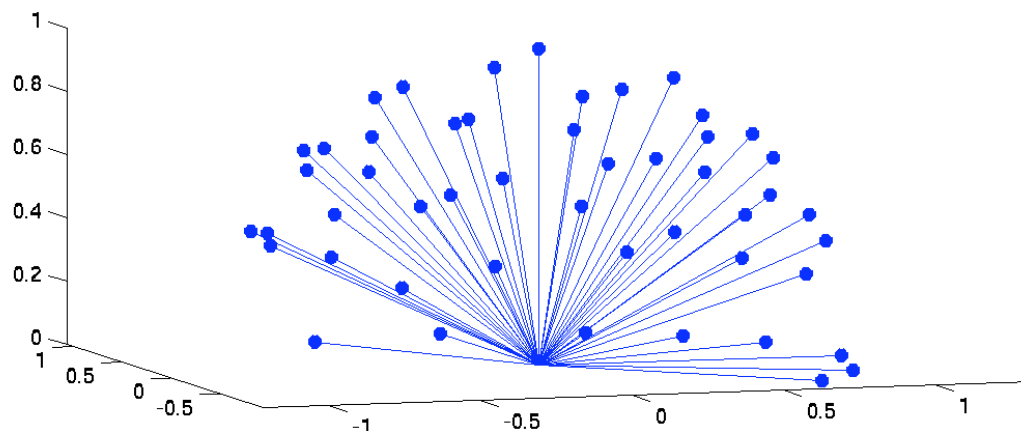
Note: this example also explains why we don't just use LDA

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 $0,\dots,255$.

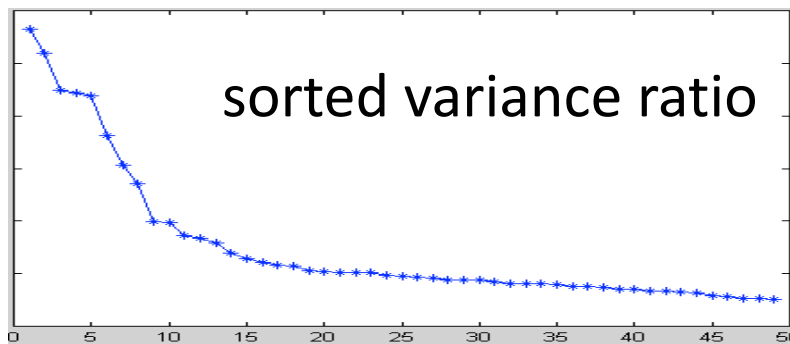
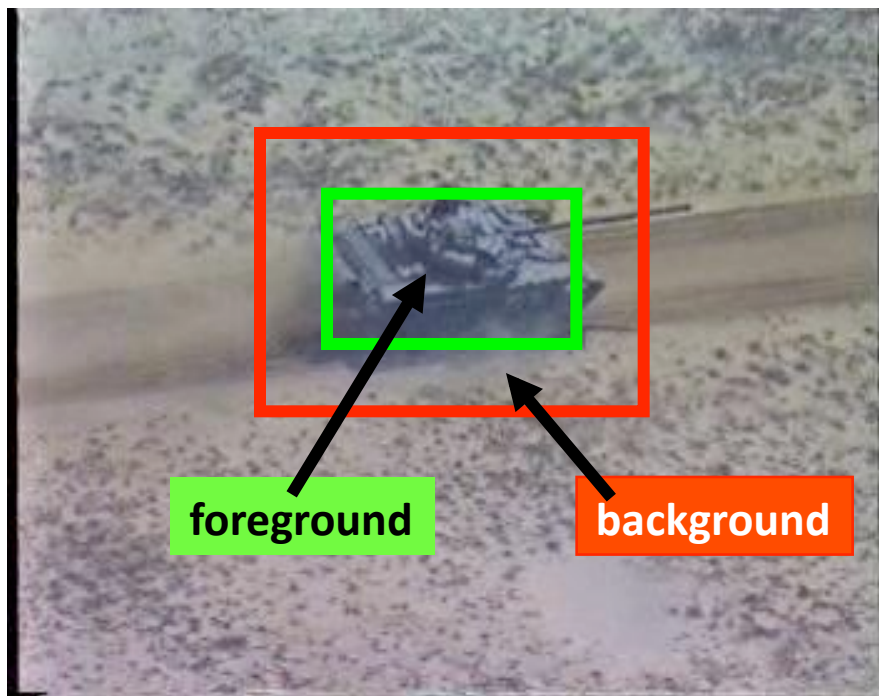


The 49 color feature candidates roughly uniformly sample the space of 1D marginal distributions of RGB.

Example

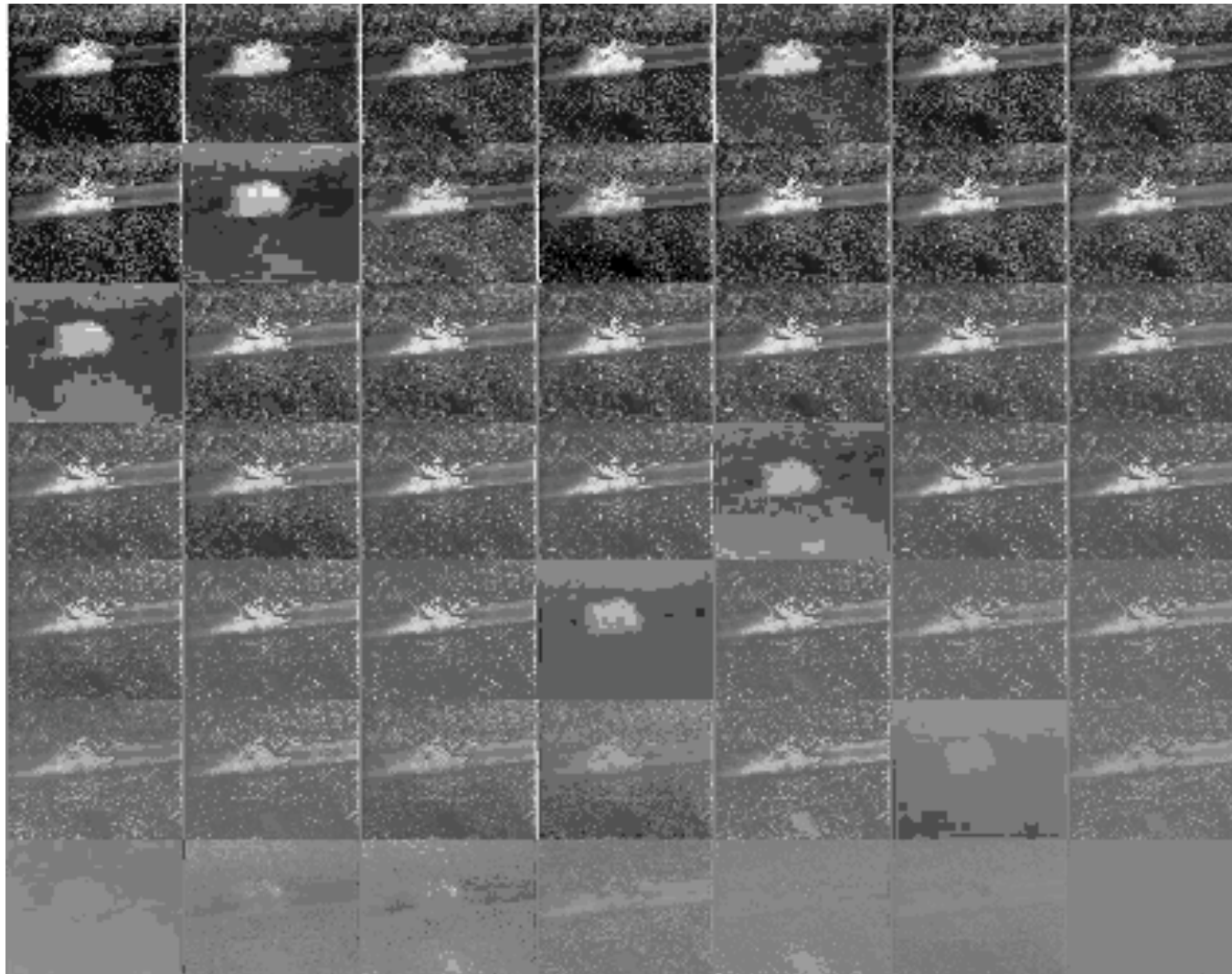
training frame

test frame



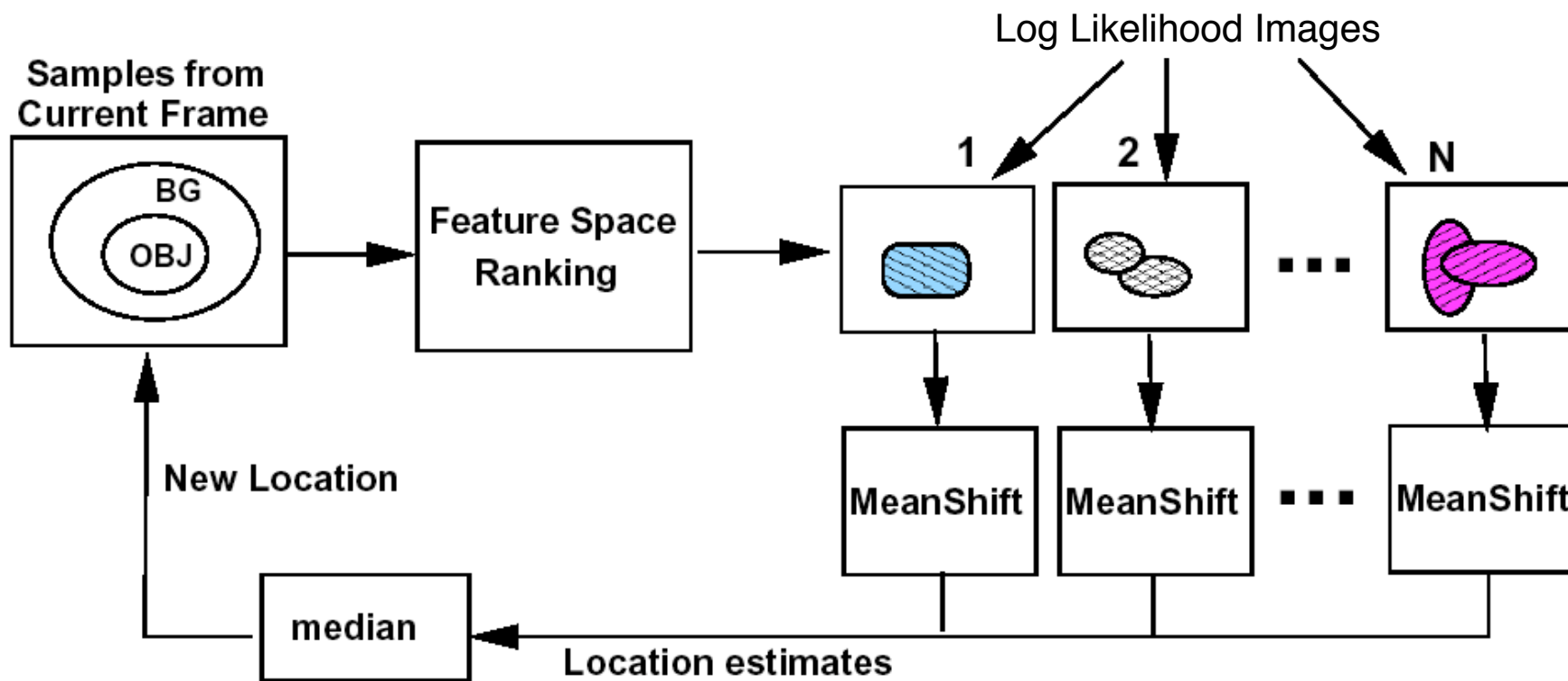
Example: Feature Ranking

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

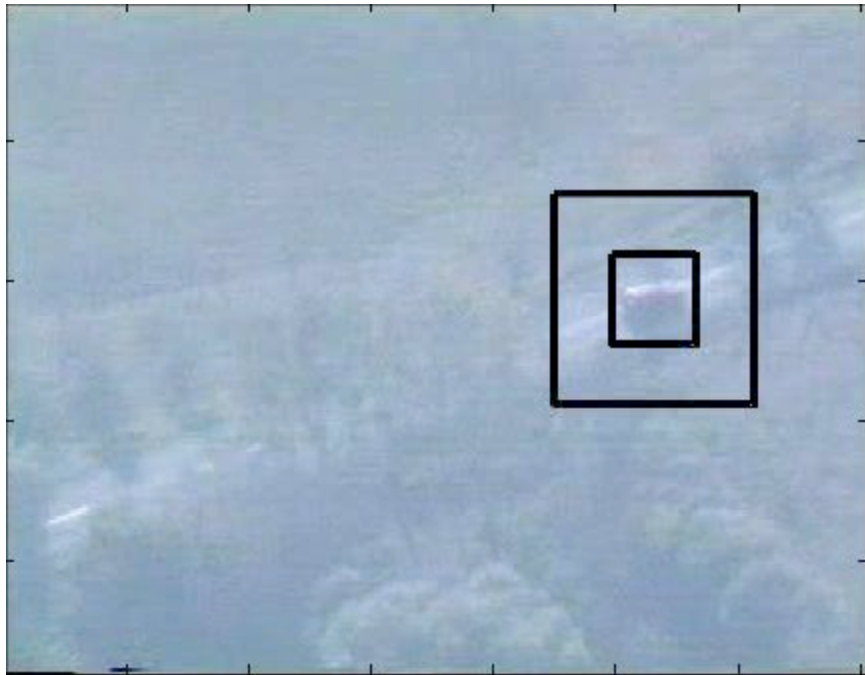
Drift: background pixels mistakenly incorporated into the object model pull the model off the correct location, leading to more misclassified background pixels, and so on.

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

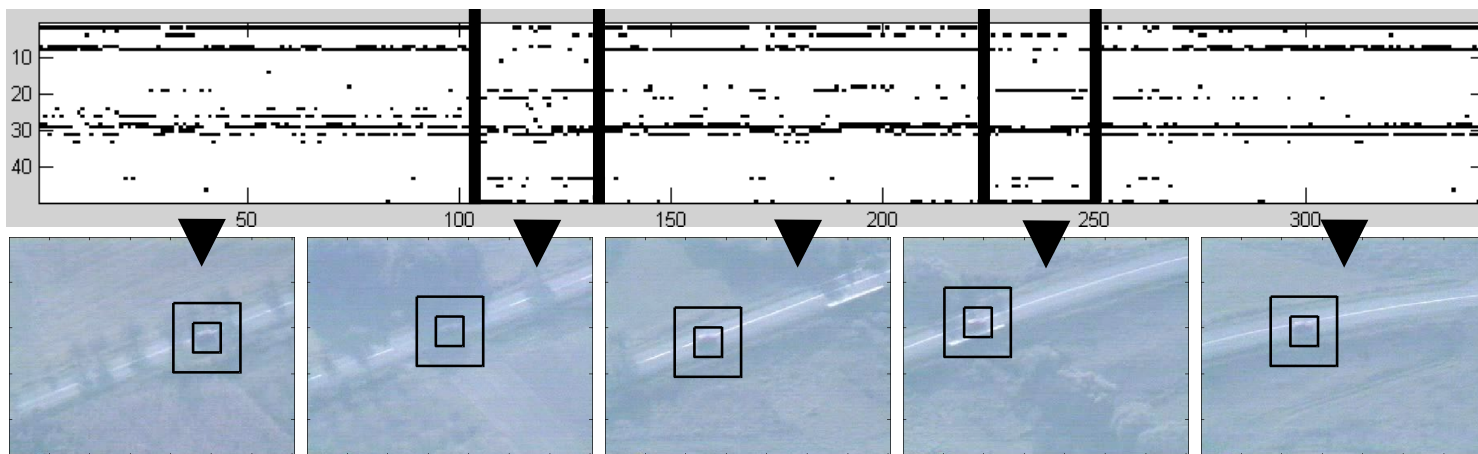
anchor distribution = object appearance histogram from first frame
model distribution = (current distribution + anchor distribution) / 2

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



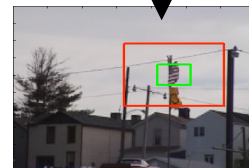
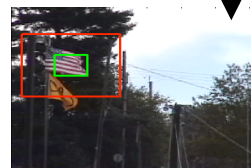
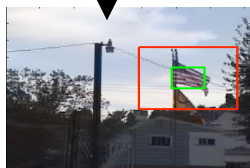
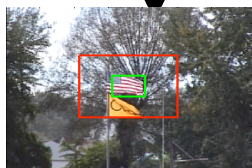
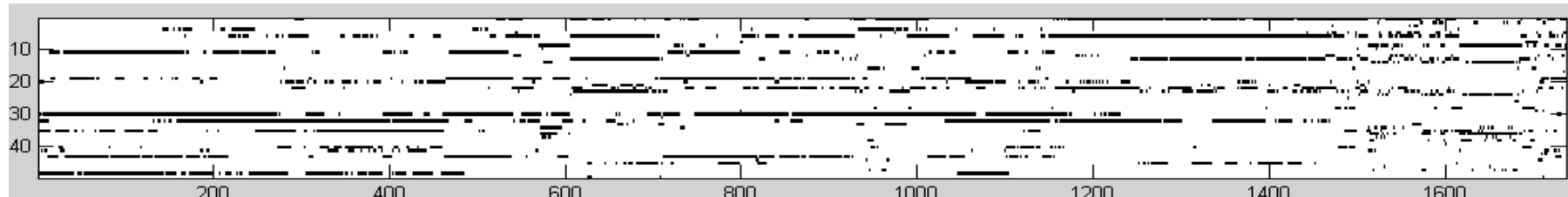
Trace of selected features



Examples: Changing Illumination / Background



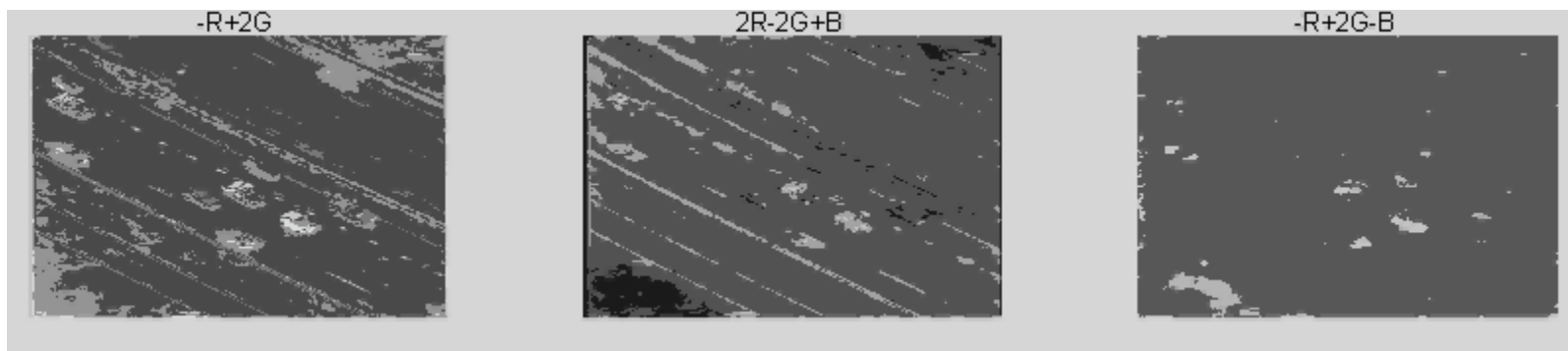
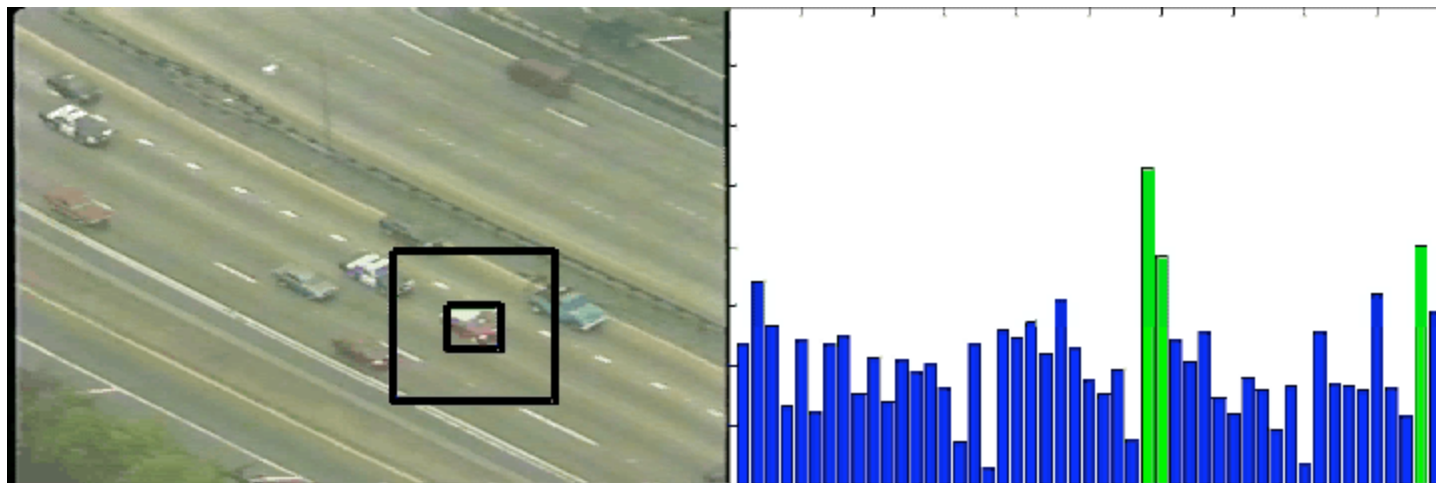
Trace of selected features



Examples: Minimizing Distractions

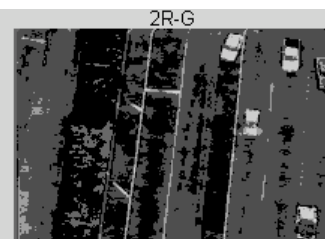
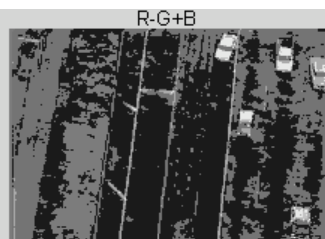
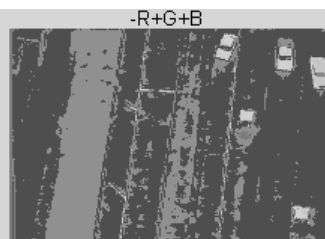
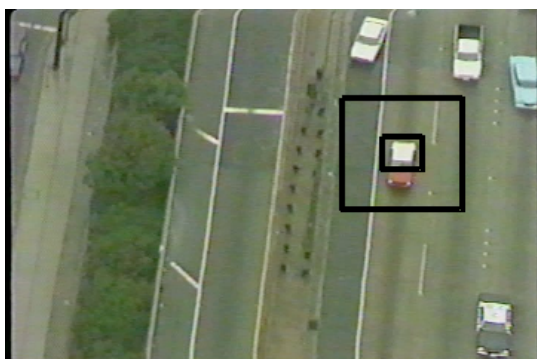
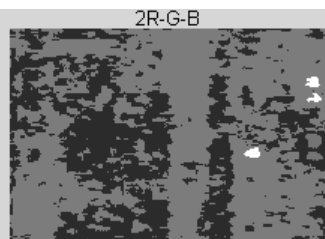
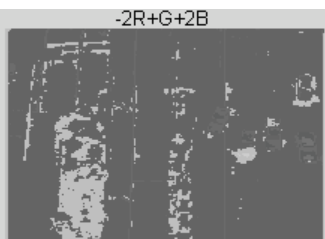
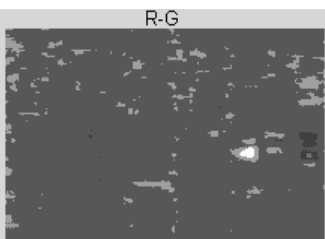
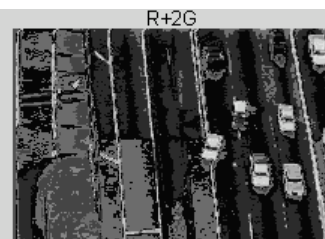
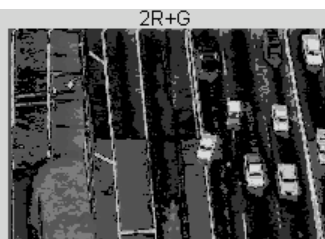
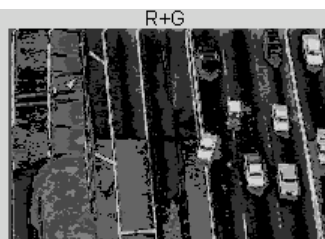
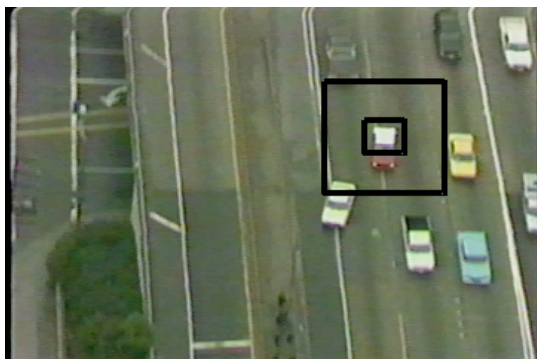
Current location

Feature scores



Top 3 weight (log likelihood) images

More Detail



top 3 weight (log likelihood) images

On-line Boosting for Tracking

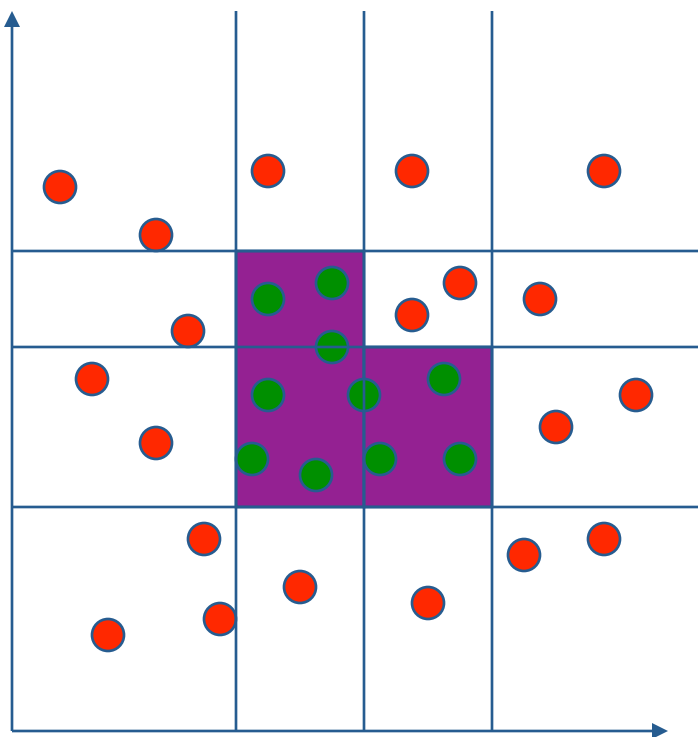
Grabner, Grabner, and Bischof, “Real-time tracking via on-line boosting.” BMVC 2006.

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

Adaboost learning

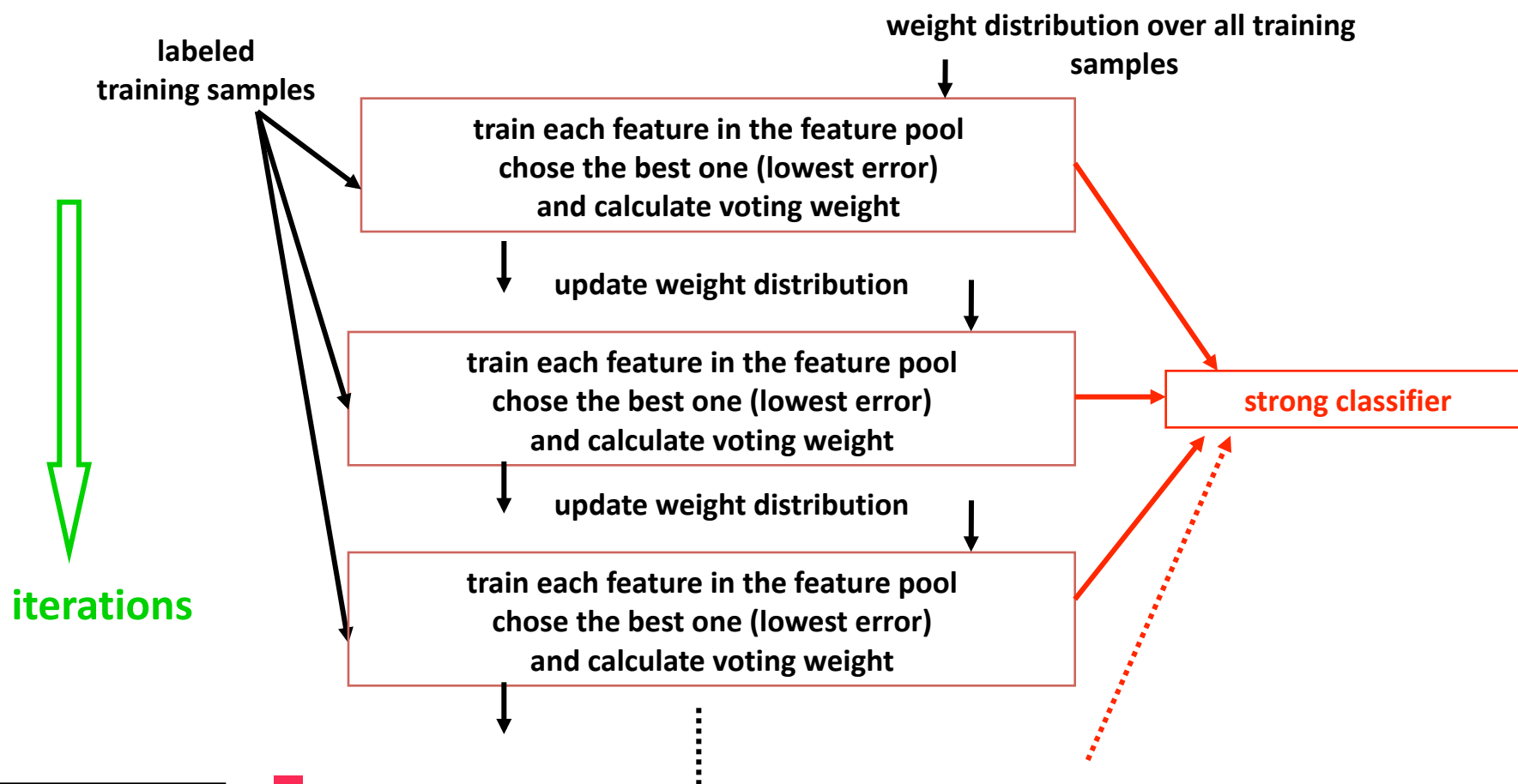
- Adaboost creates a single strong classifier from many weak classifiers



- Initialize sample weights
- For each cycle:
 - Find a classifier that performs well on the weighted sample
 - Increase weights of misclassified examples
- Return a weighted combination of classifiers

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

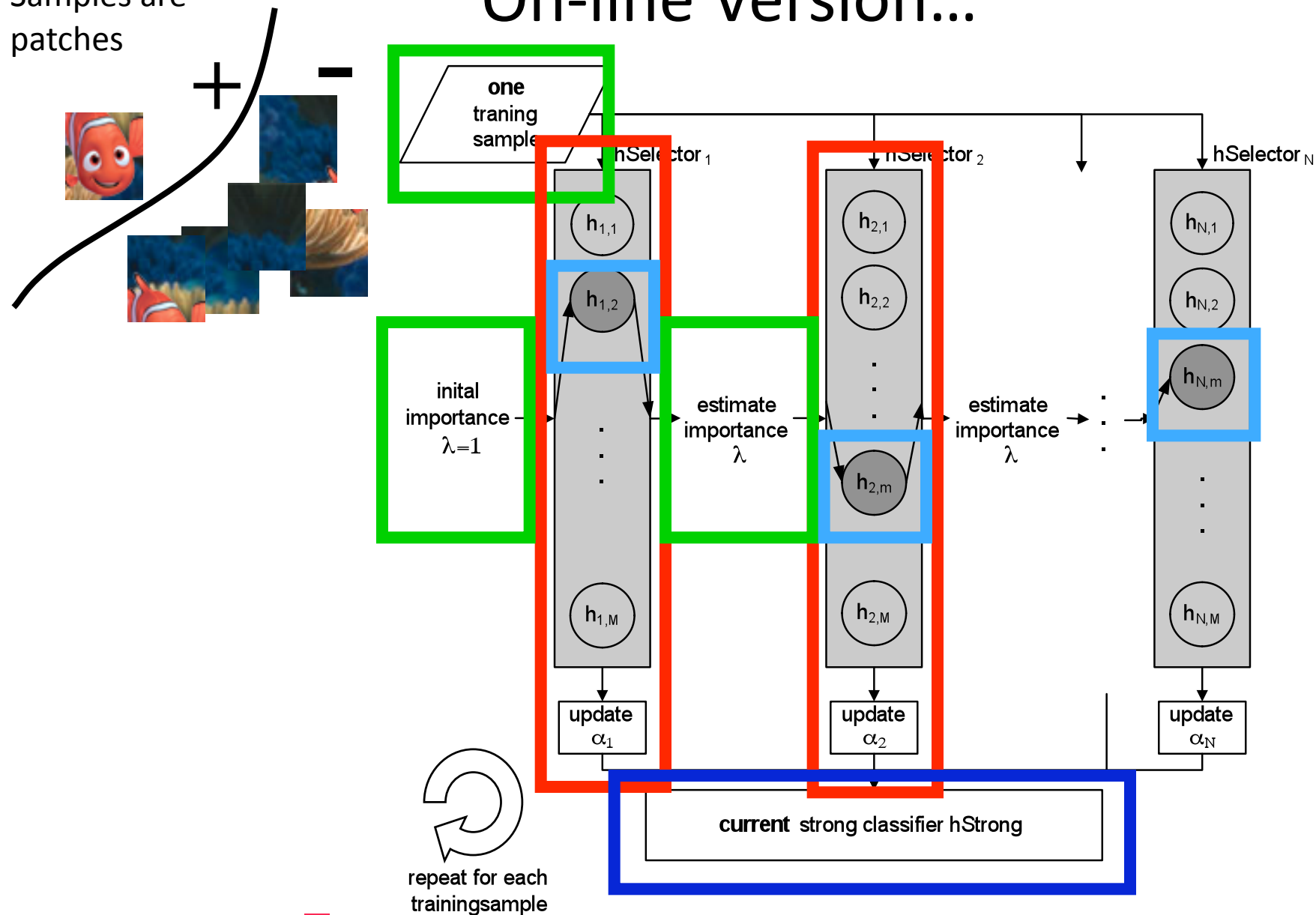


Robert Collins

Penn State

Samples are patches

On-line Version...



Horst Bischof



SU-VLPR 2010

Tracking Examples

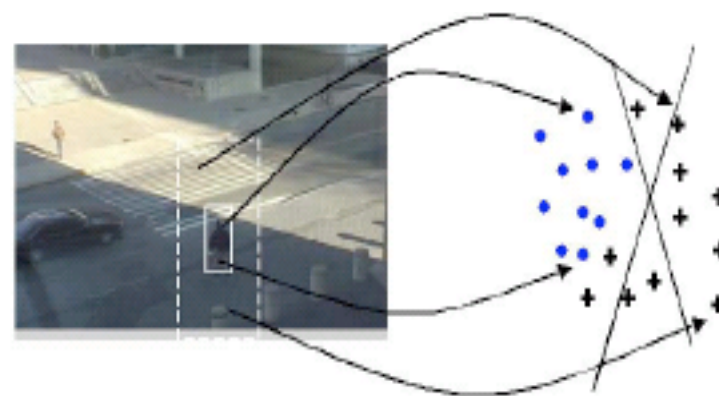


Ensemble Tracking

Avidan, “Ensemble Tracking,” PAMI 2007

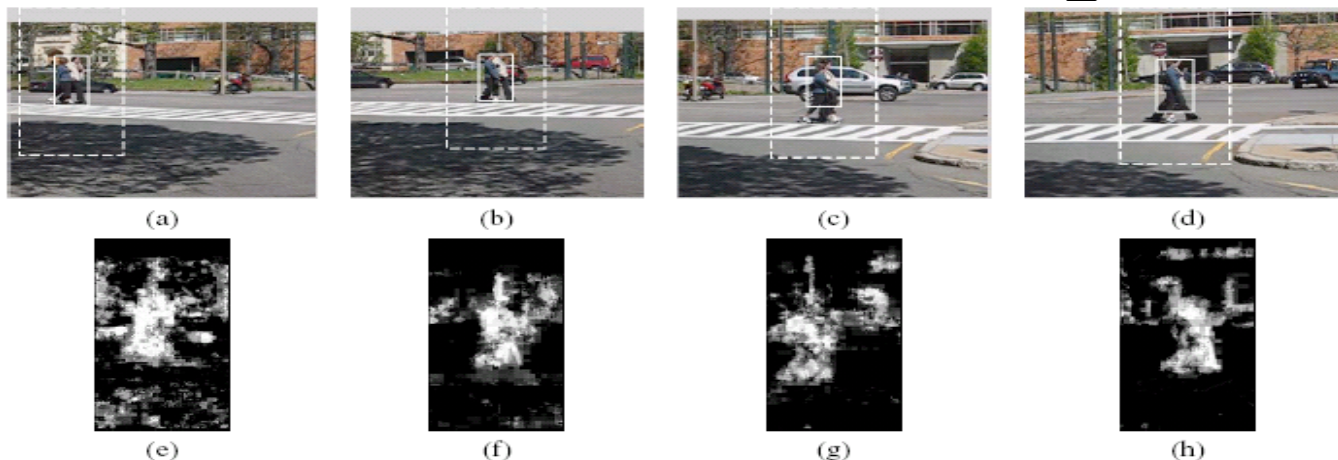
Use online boosting to select and maintain a set of weak classifiers (rather than single features), weighted to form a strong classifier. Samples are pixels.

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:

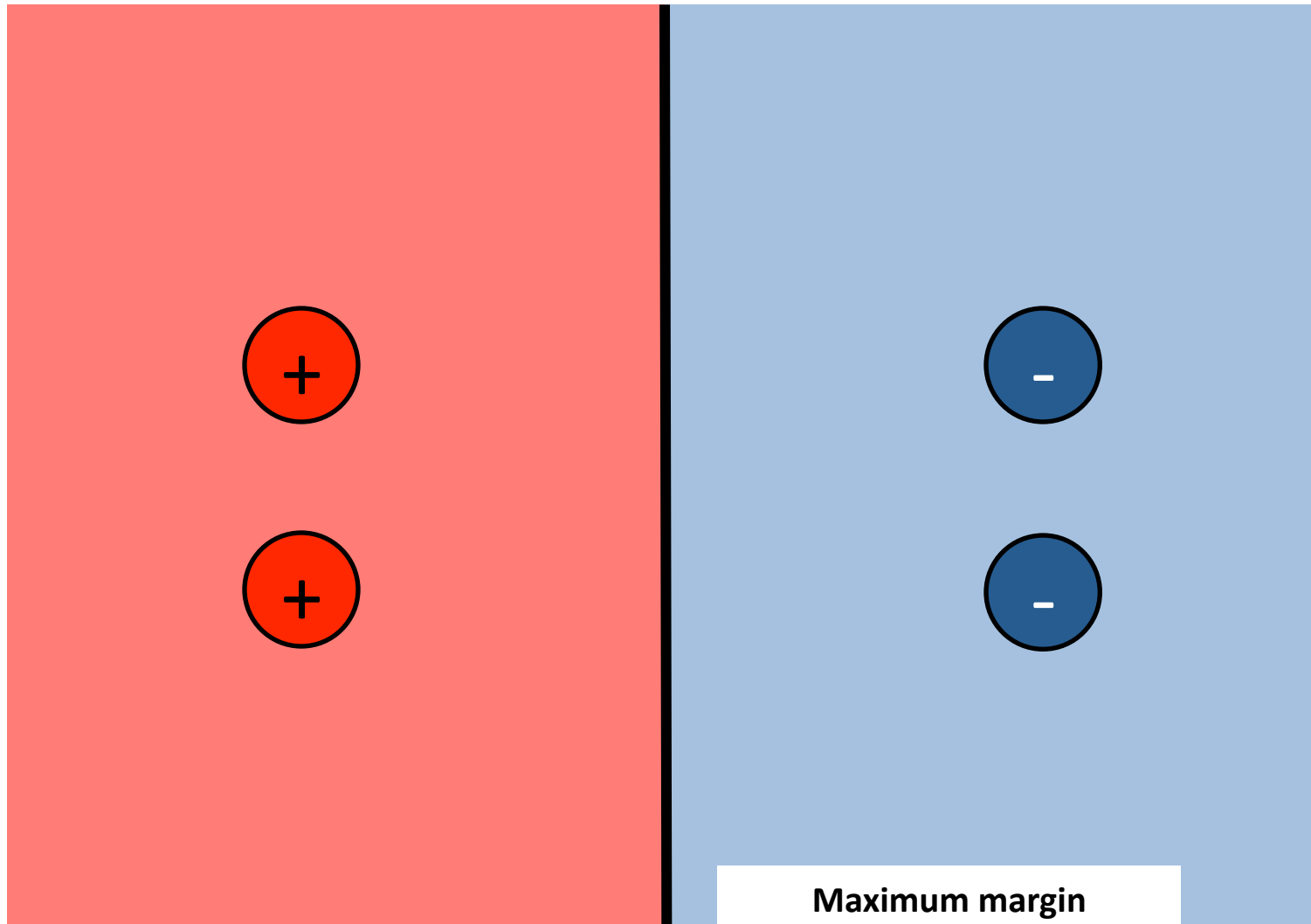
- Combine 2 classifiers

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

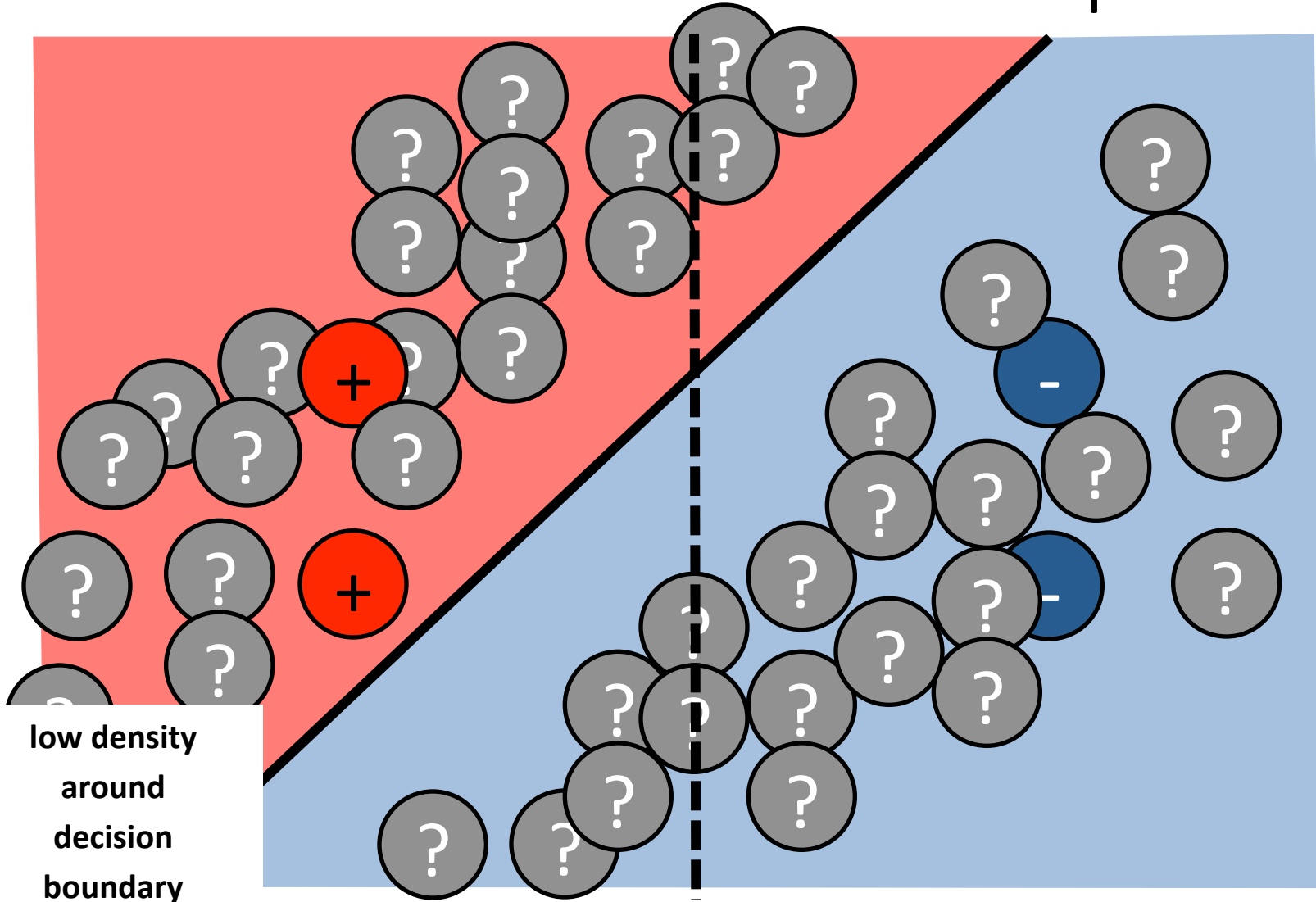
Classifier $H^{\text{off}} + H^{\text{on}}$ cannot deviate too much from H^{off}

- Semi-supervised learning framework

Supervised learning

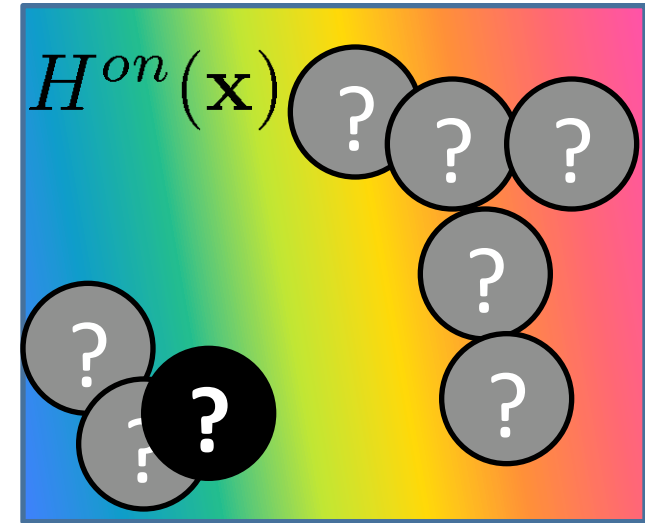
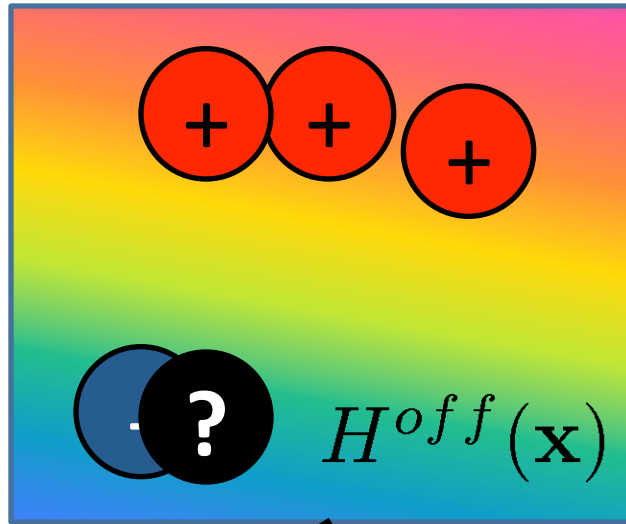


Can Unlabeled Data Help?



Drift Avoidance

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



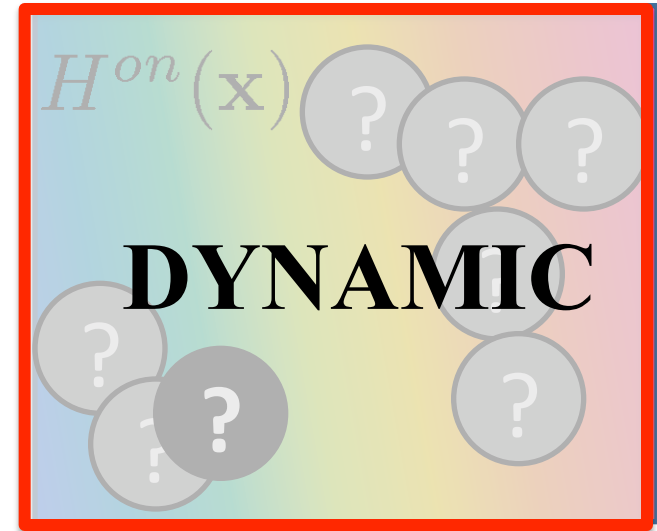
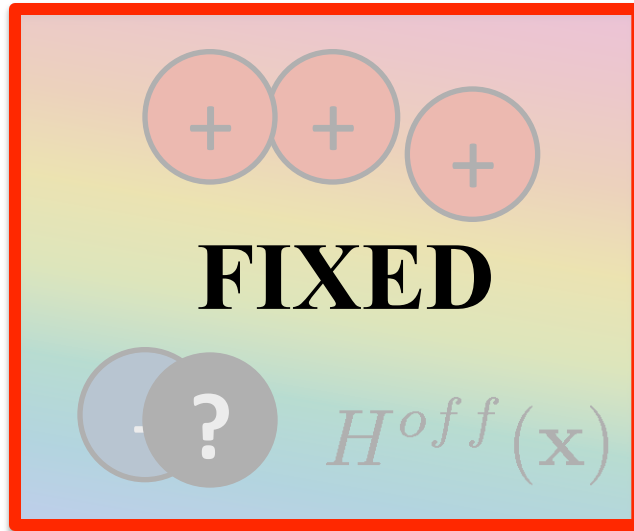
Labeled data comes from first frame

Combined classifier

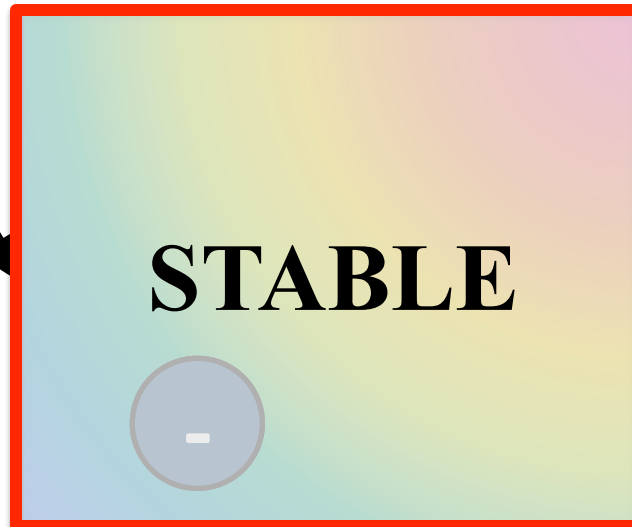
$$\text{sign} (H^{off}(\mathbf{x}) + H^{on}(\mathbf{x}))$$

Drift Avoidance

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



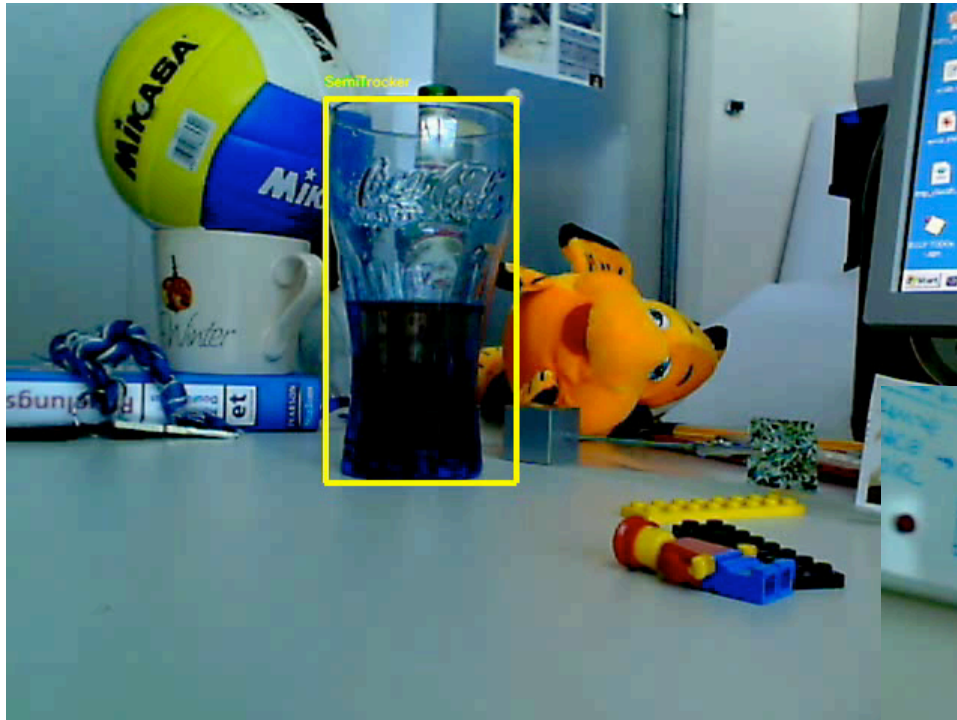
Labeled data
comes from
first frame



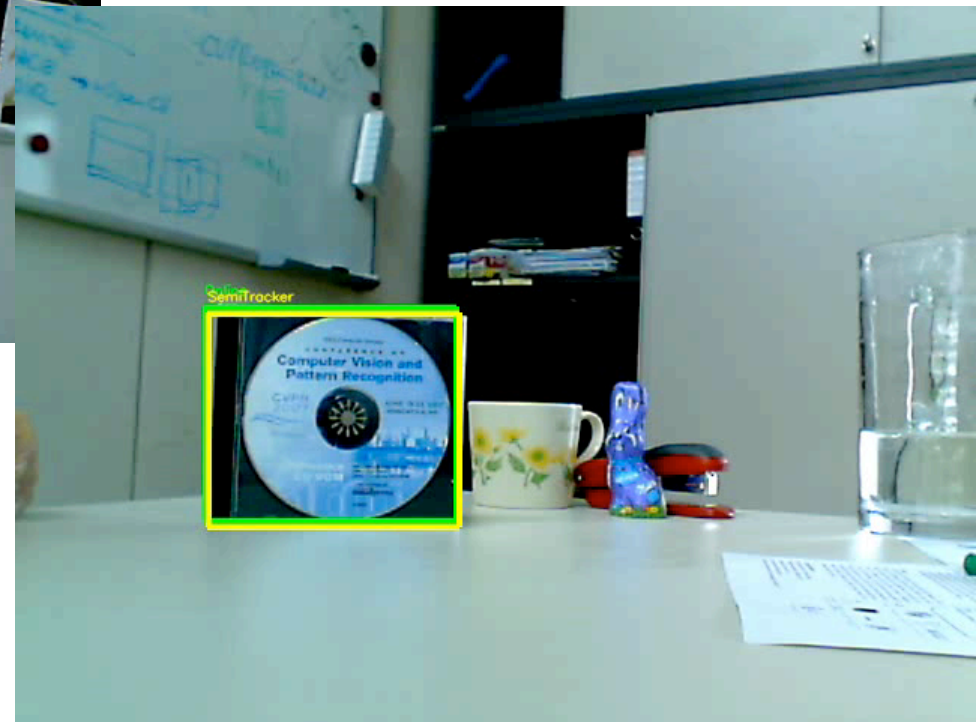
Combined
classifier

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

Examples



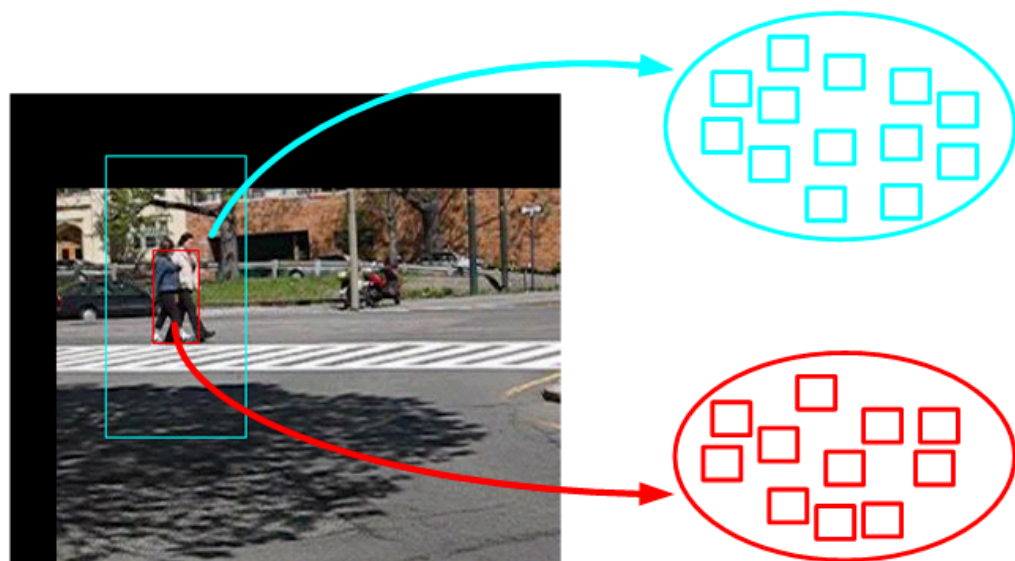
Green: online boosting
Yellow: semi-supervised



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.



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.