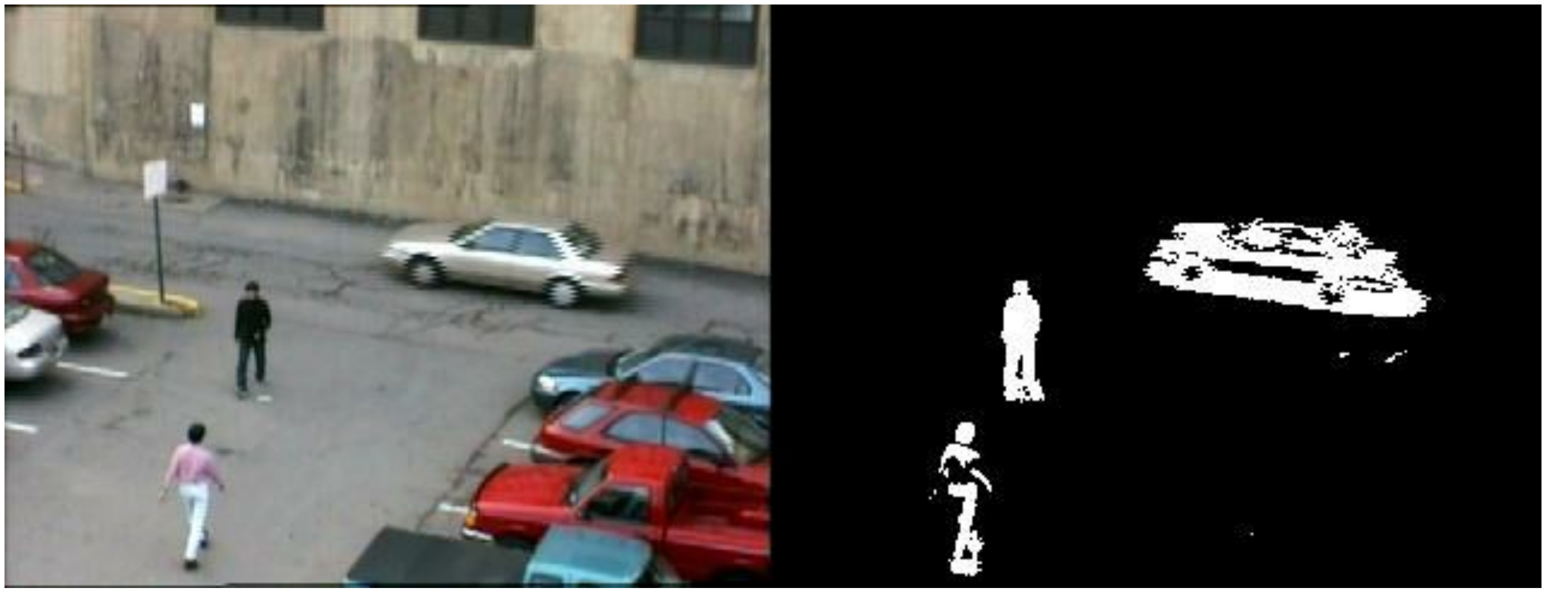
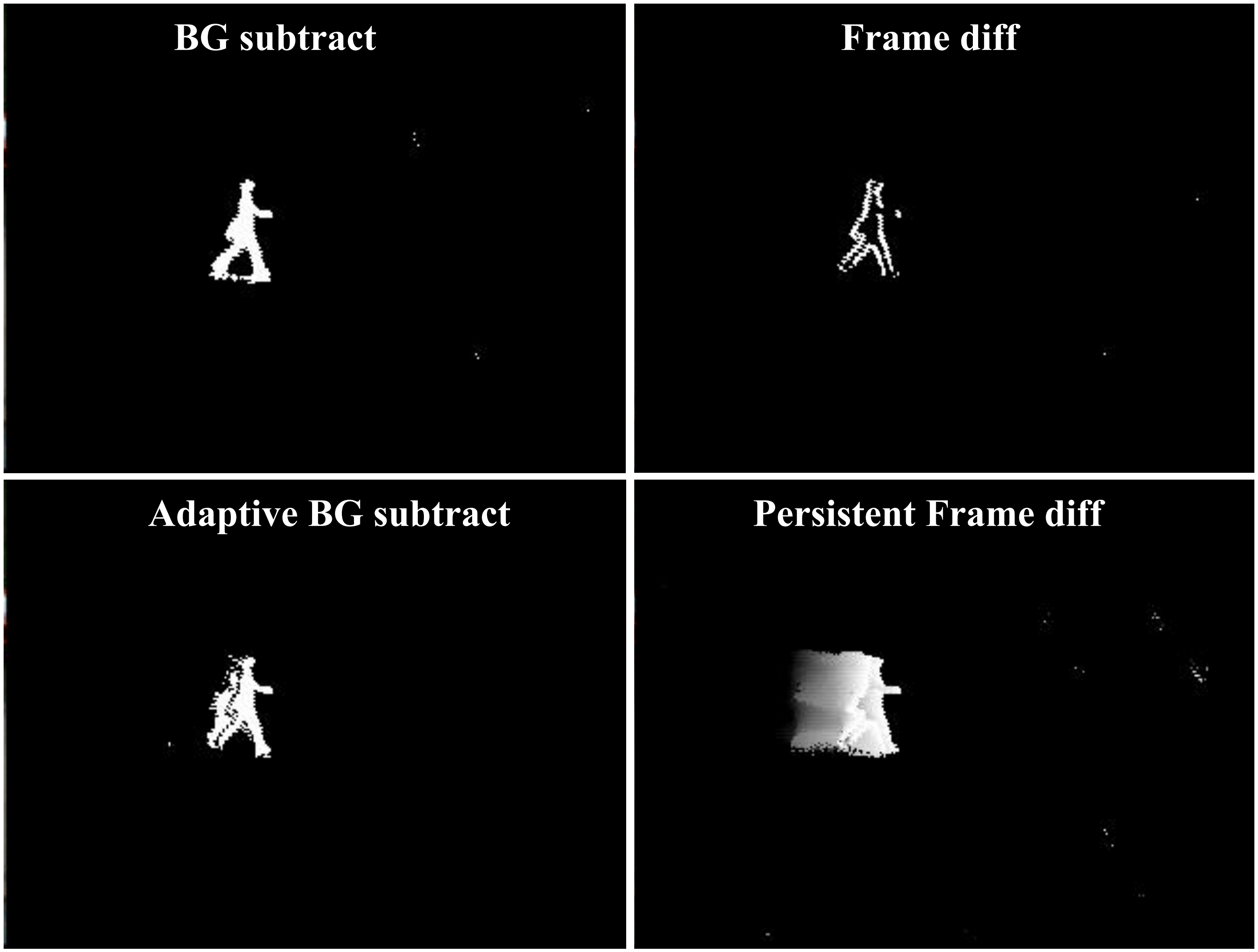
Robert Collins
CMPEN454

Intro to Tracking

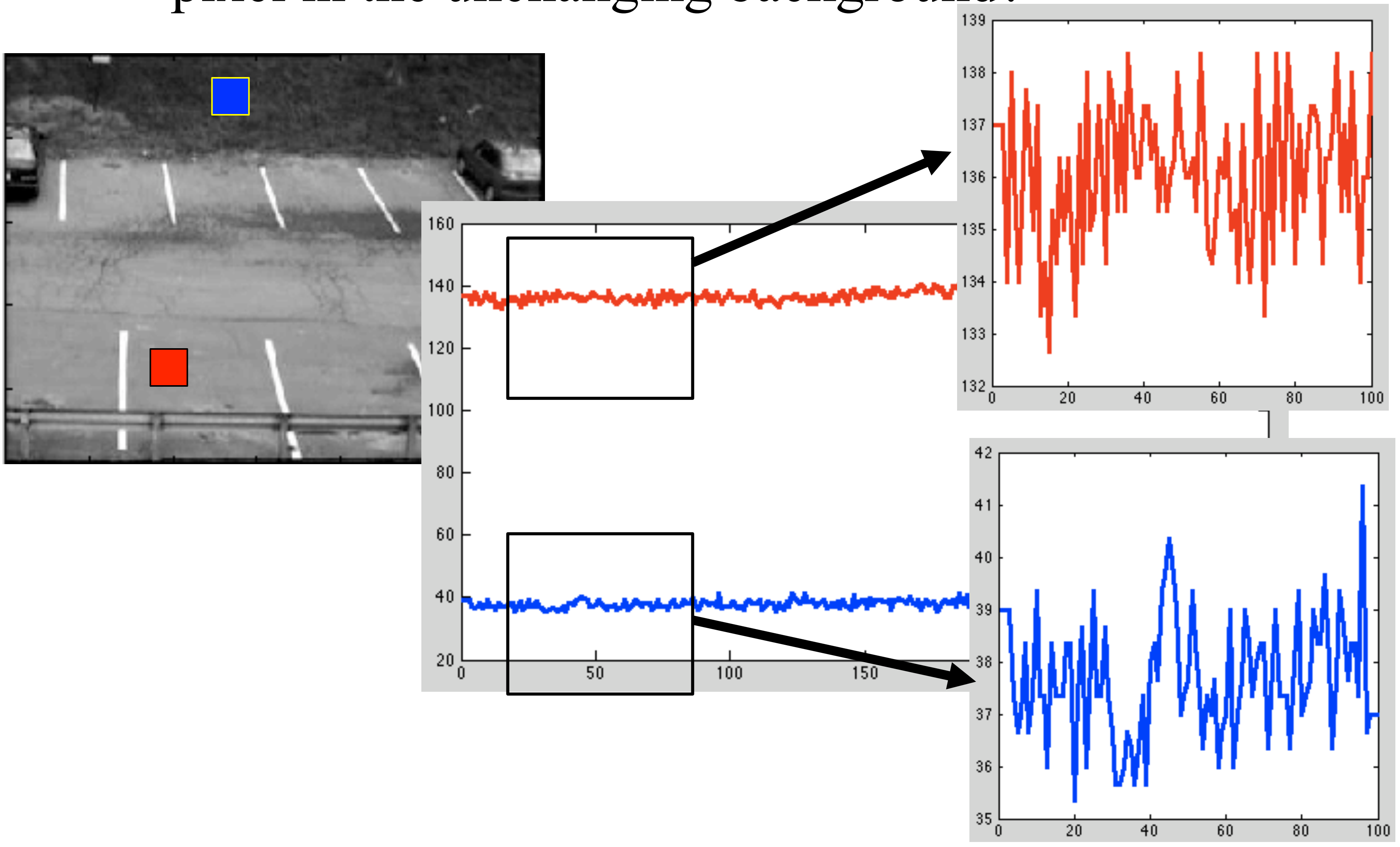


Recall from Before



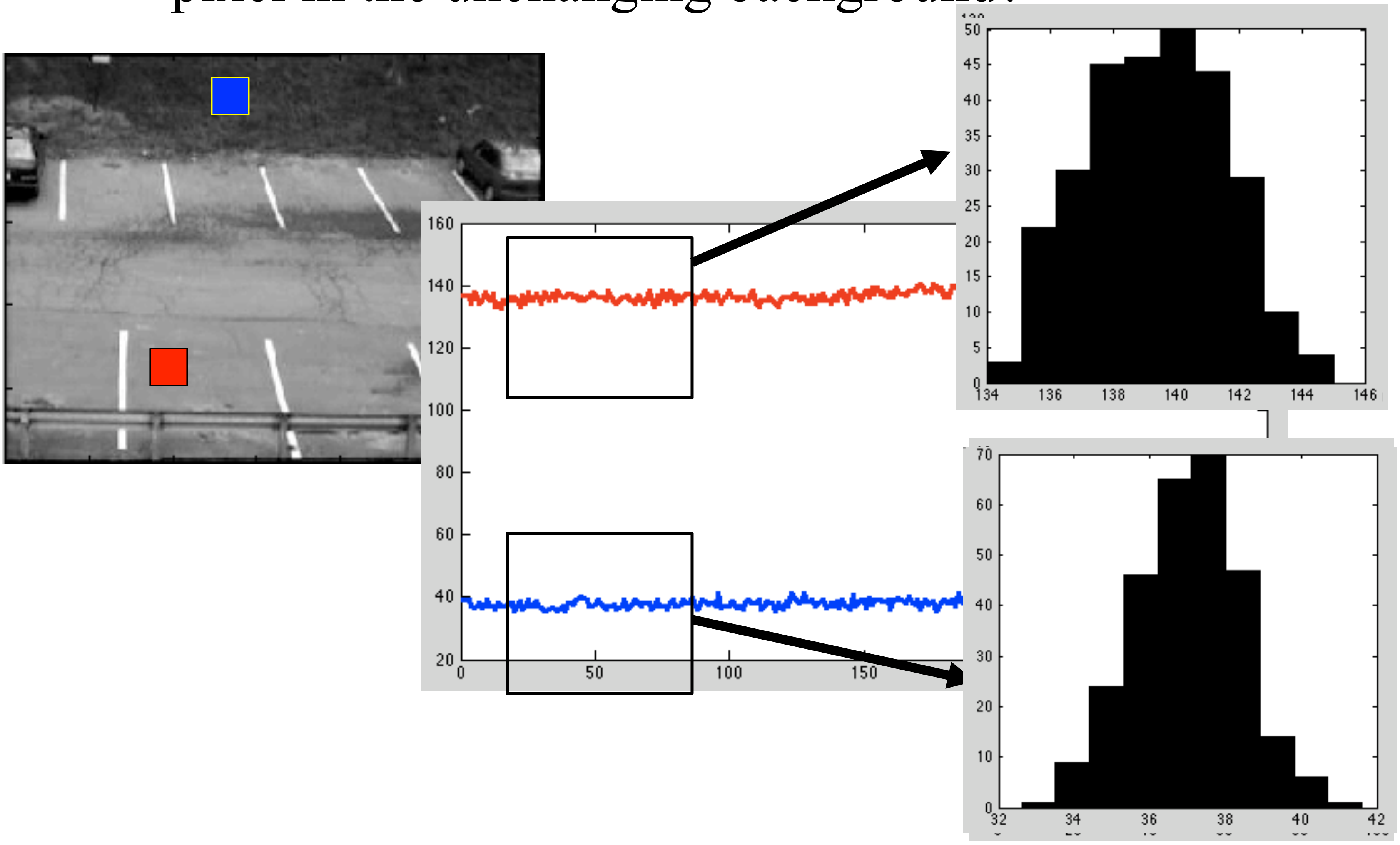
Understanding Bg Subtraction

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



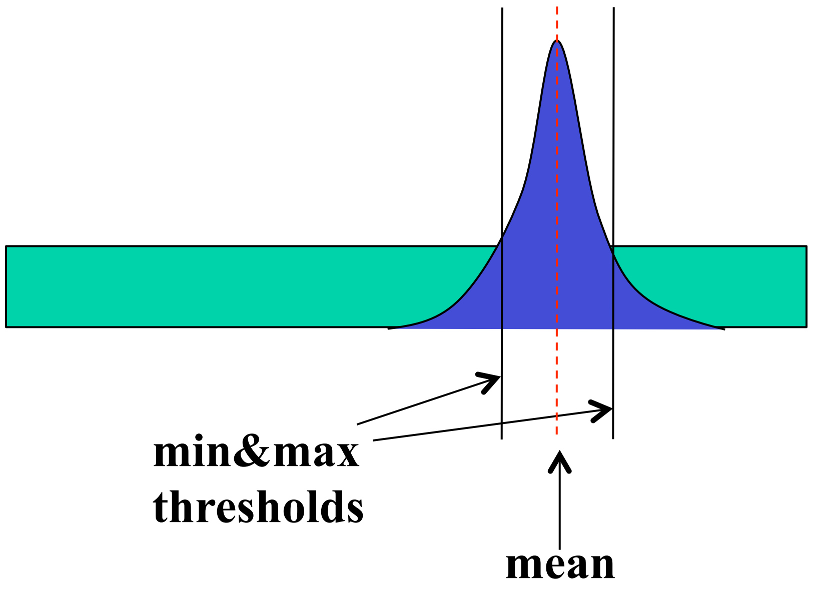
Understanding Bg Subtraction

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



Understanding Bg Subtraction

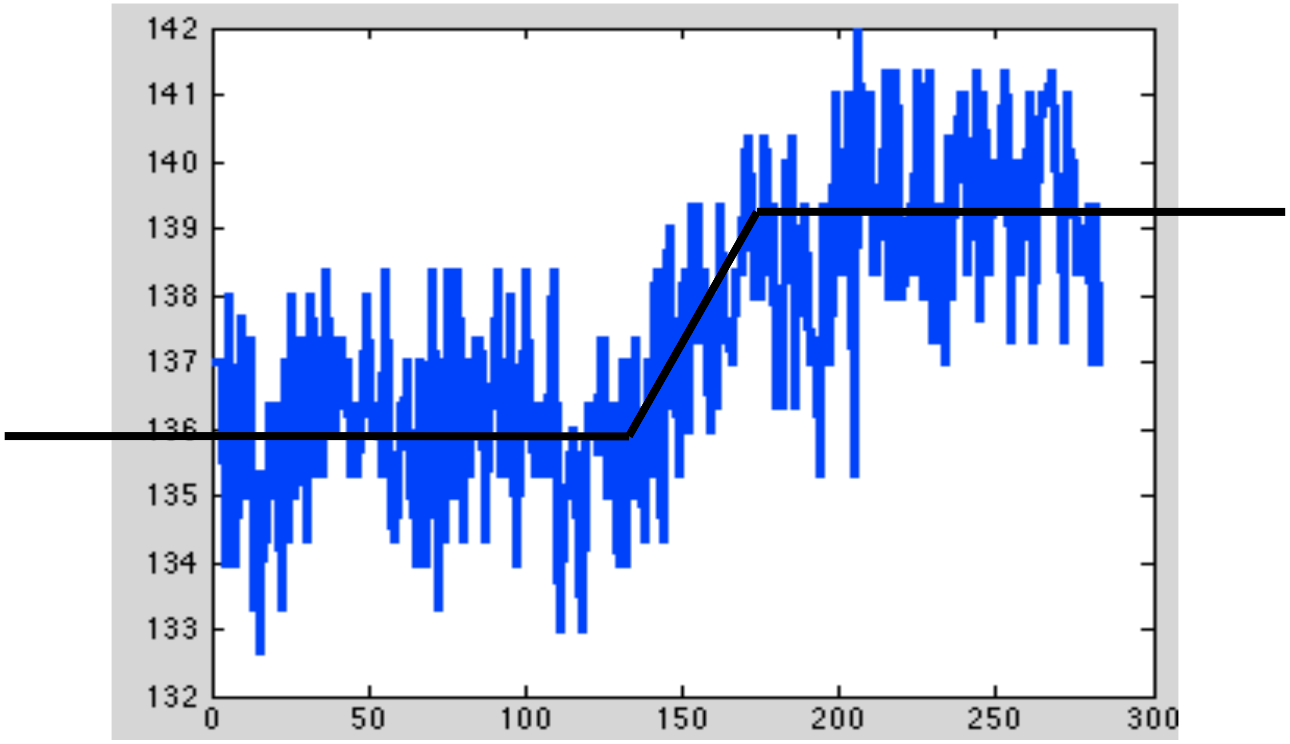
- Basic idea: background model is Gaussian with adaptive mean and constant variance. Foreground model is uniform distribution (or Gaussian with large variance).



or equivalently:
Abs(pixval-mean) < threshold

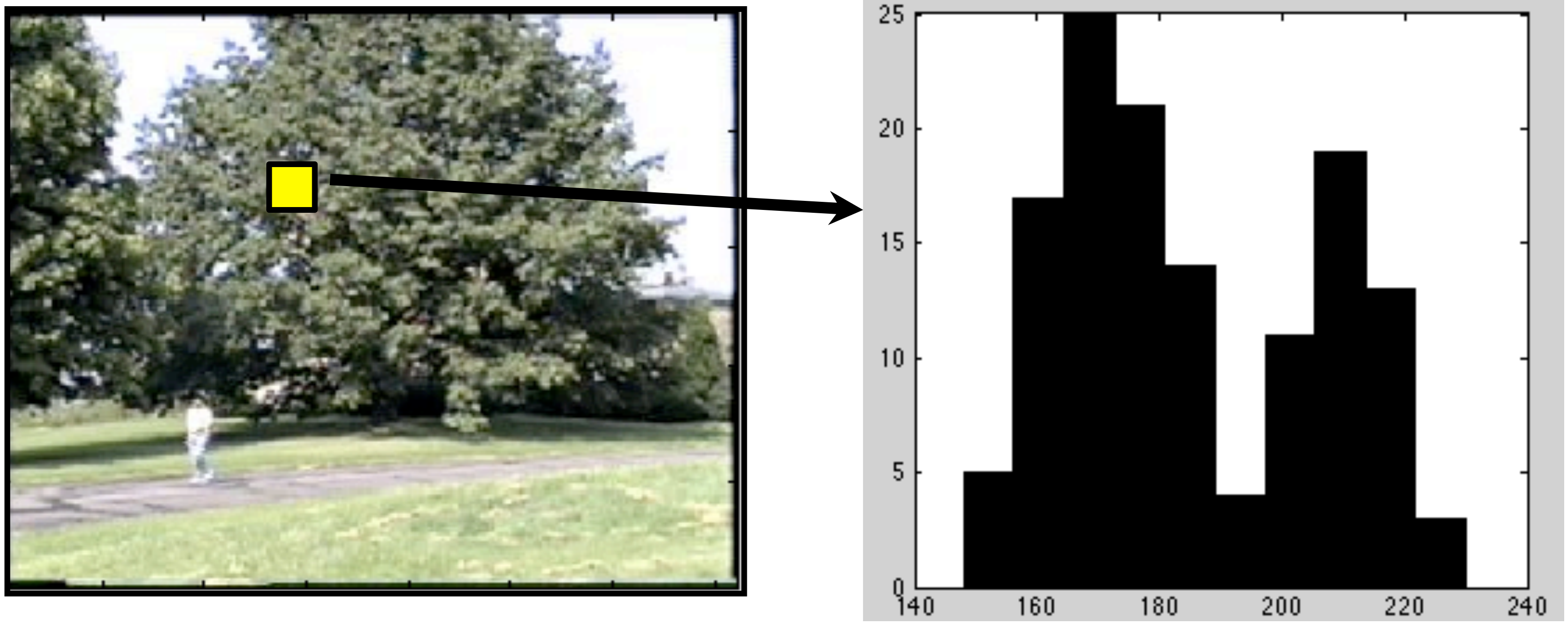
Understanding Bg Subtraction

- 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



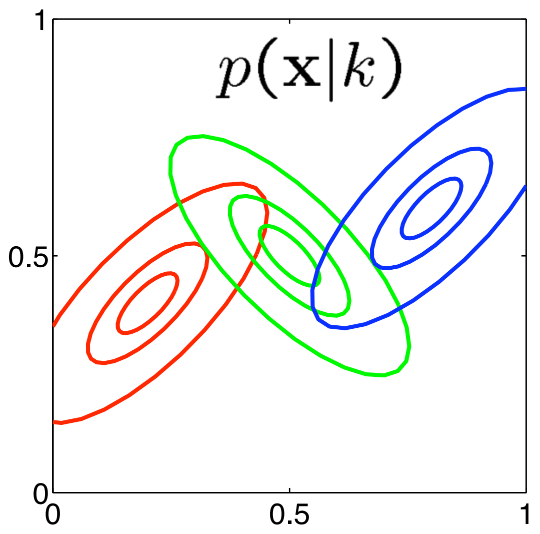
Idea: Use a Mixture of Gaussians!

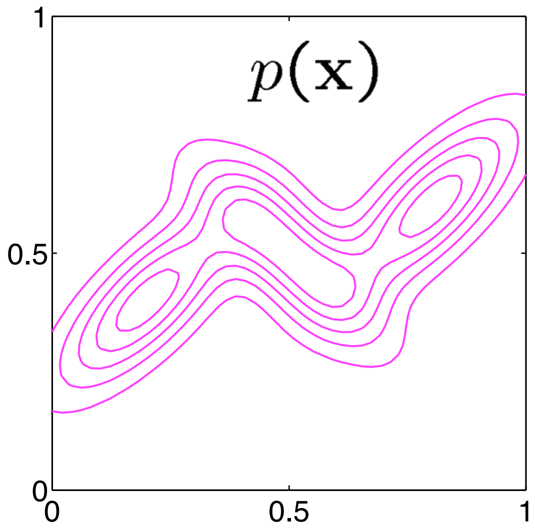
- Linear combination of Gaussians

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

- Normalization and positivity requirements

$$\sum_{k=1}^{K} \pi_k = 1 \qquad 0 \leqslant \pi_k \leqslant 1$$

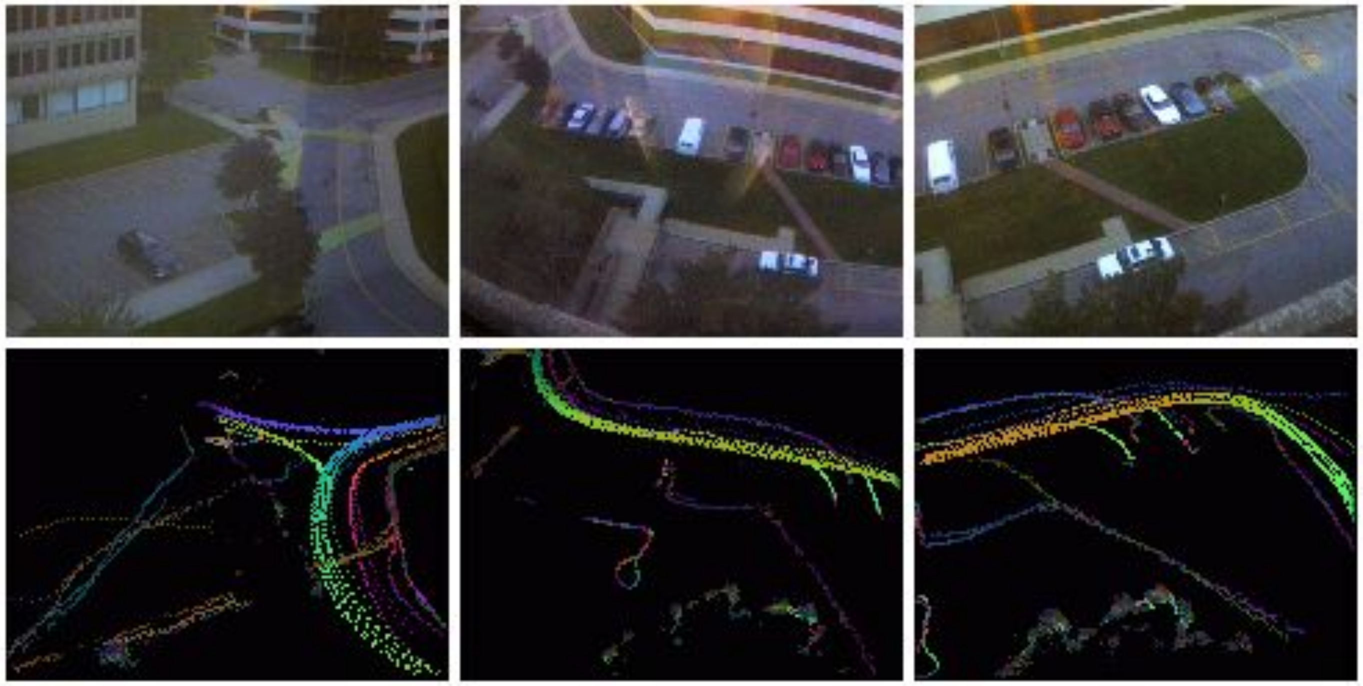




Statistical Background Modeling

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.



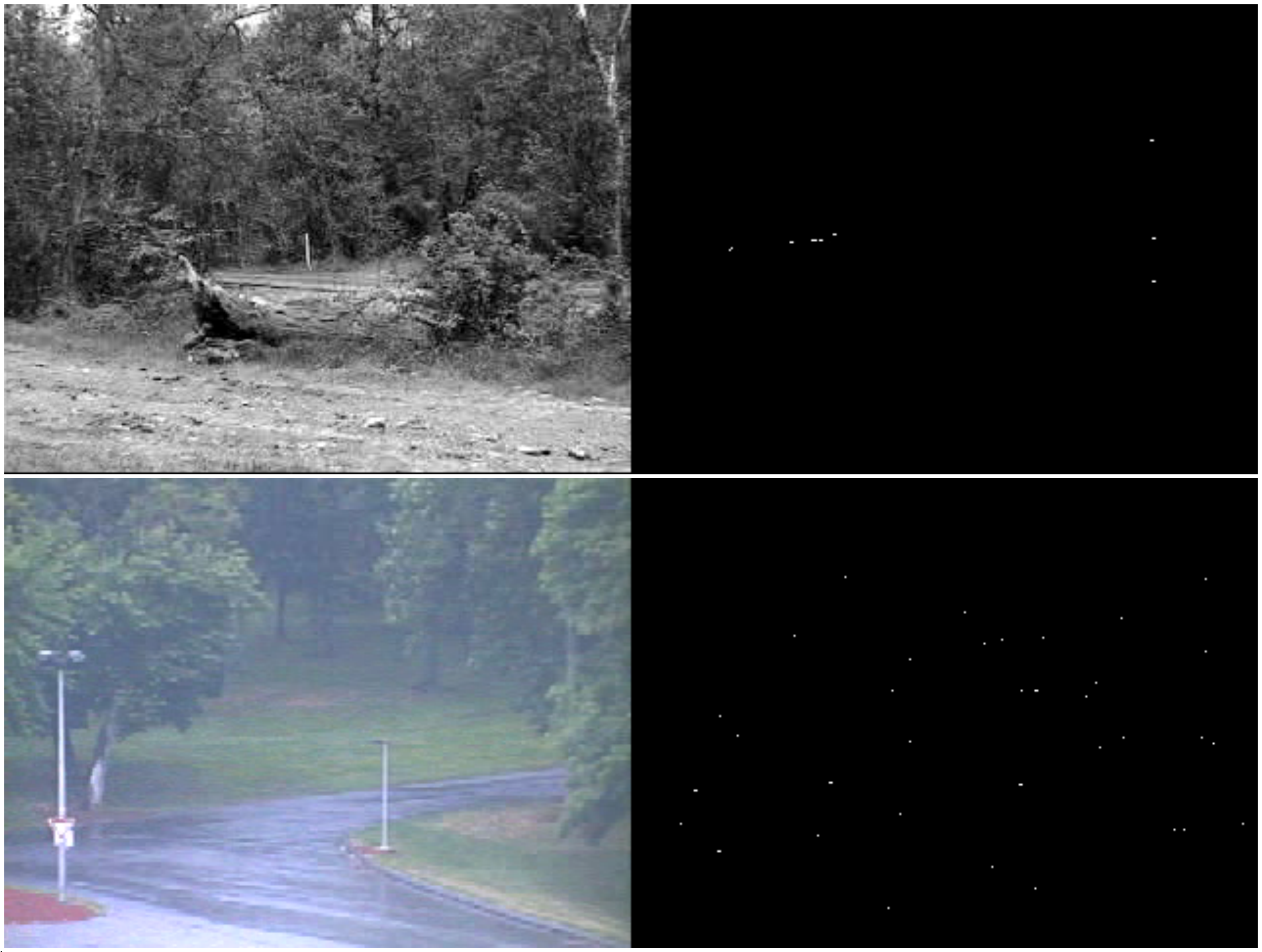
Code available from Zivkovic.

http://www.zoranz.net/DOWNLOAD.html

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.

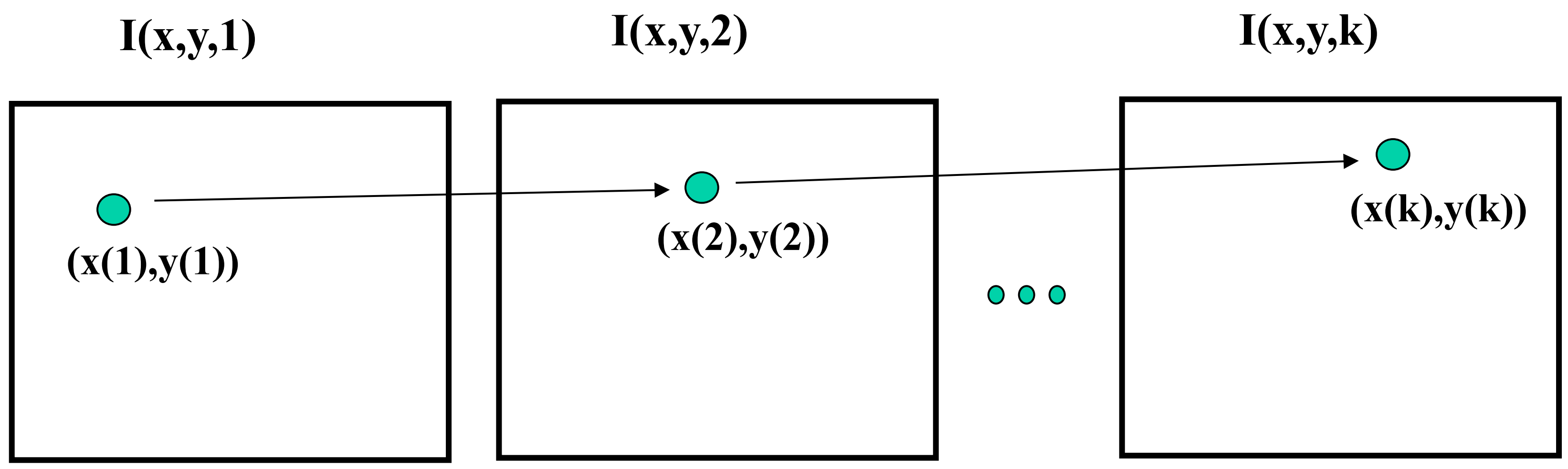


movies

Videos by Ahmed Elgammal, Rutgers

Understanding Frame Differencing

consider a scene point moving through an image sequence



assumption: its brightness/color will remain the same
(that's how we can recognize that it IS the same point)

$$\boxed{I(x(t), y(t), t) = Constant}$$

Brightness Constancy Equation

$$\boldsymbol{I}\,(x(t), y(t), t) = Constant$$

Take derivative of both sides wrt time:

$$\frac{d\,\boldsymbol{I}\,(x(t), y(t), t)}{dt} = 0$$

(using chain rule)

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

Understanding Frame Differencing

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

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

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

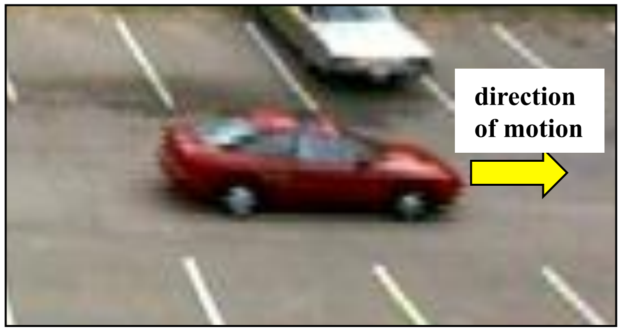
$\frac{\partial I}{\partial t}$ (temporal derivative = frame difference!)

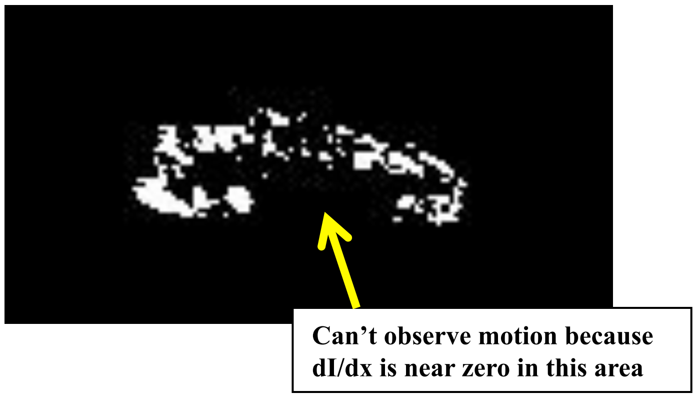
Understanding Frame Differencing

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

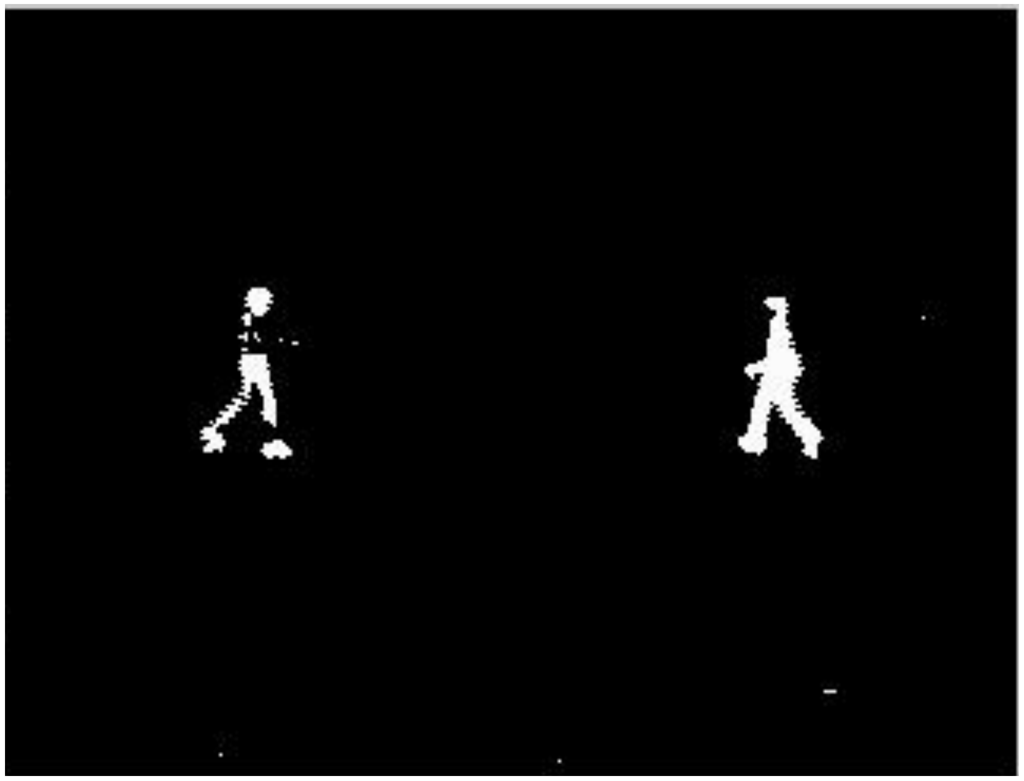
Observations:

- 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. [if brightness constancy assumption is true]
- 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.





Grouping Pixels into Blobs



Motivation: change detection is a pixel-level process.
We want to raise our description to a higher level of abstraction.

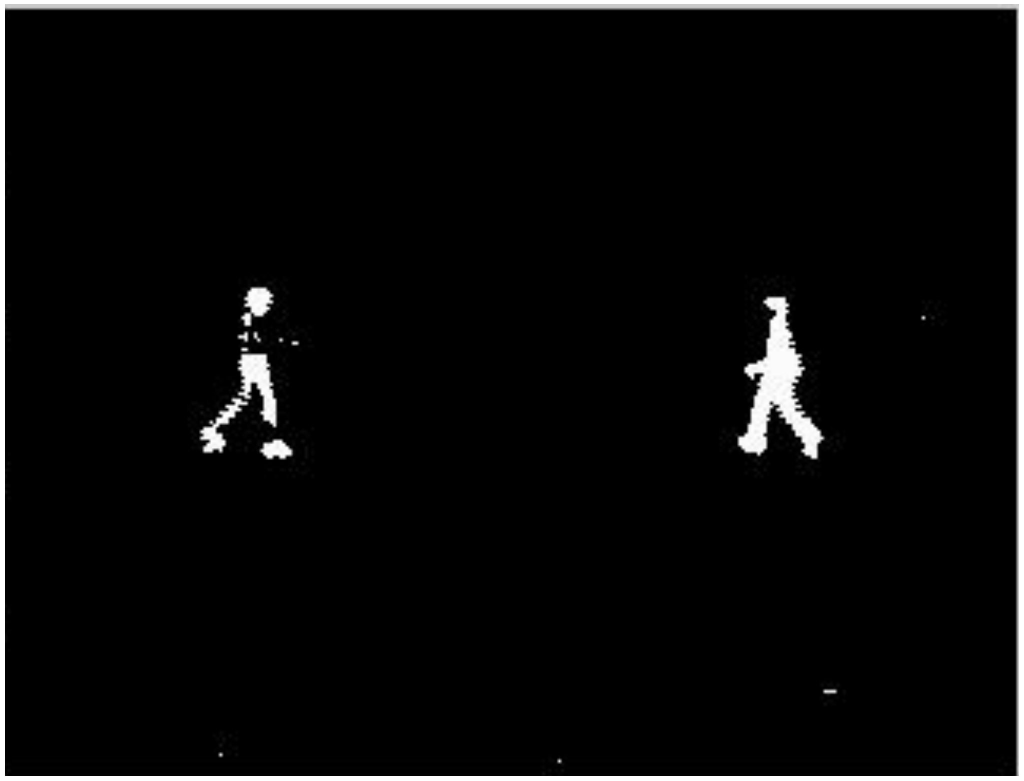
Motivation

If we have an object-level blob description, AND we can maintain a model of the scene background, then we can artificially remove objects from the video sequence.

movie

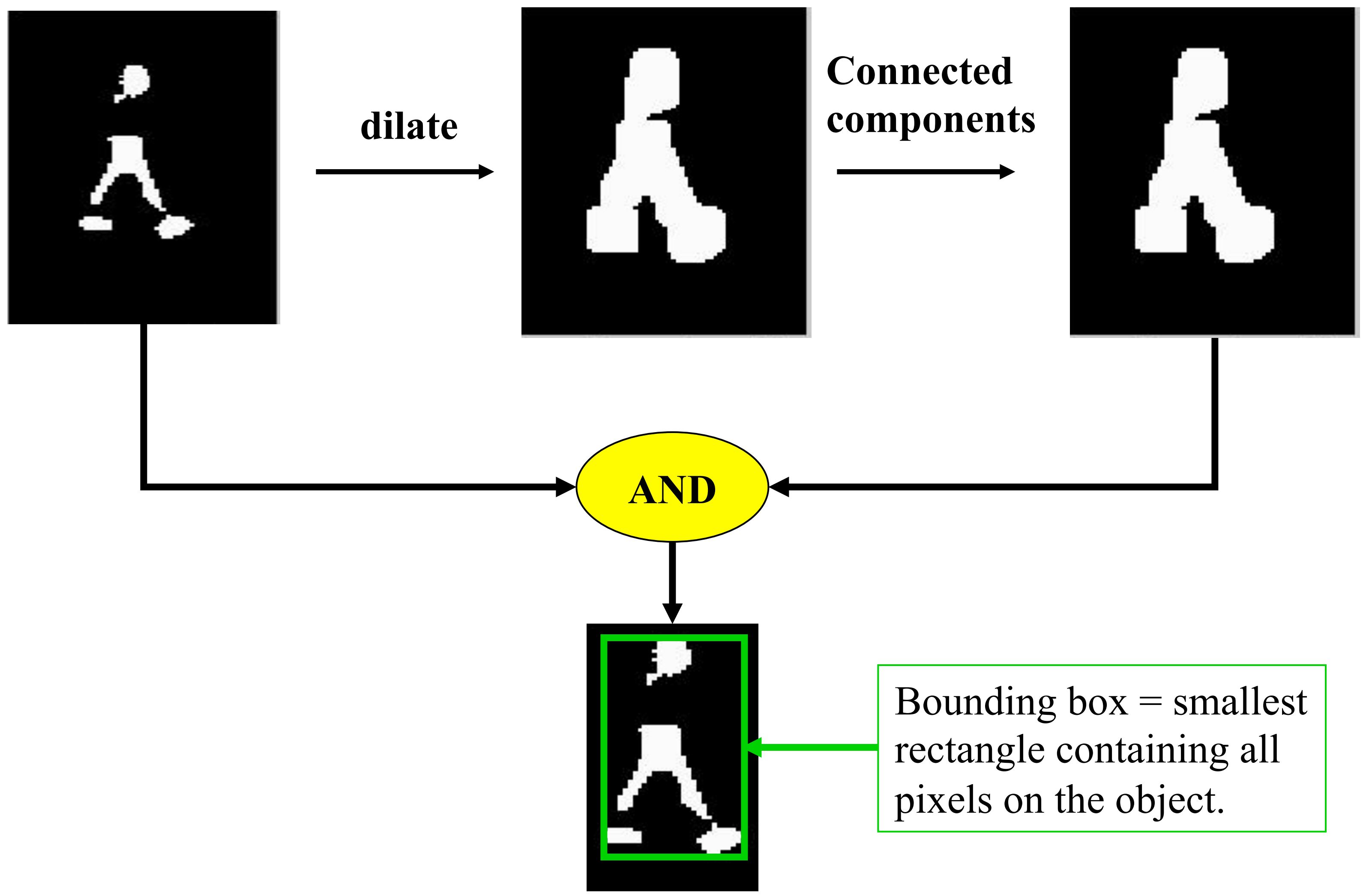
Alan Lipton, "Virtual Postman – Real-time, Interactive Virtual Video," IASTED CGIM, Palm Springs, CA, Octover 1999.

Grouping Pixels into Blobs



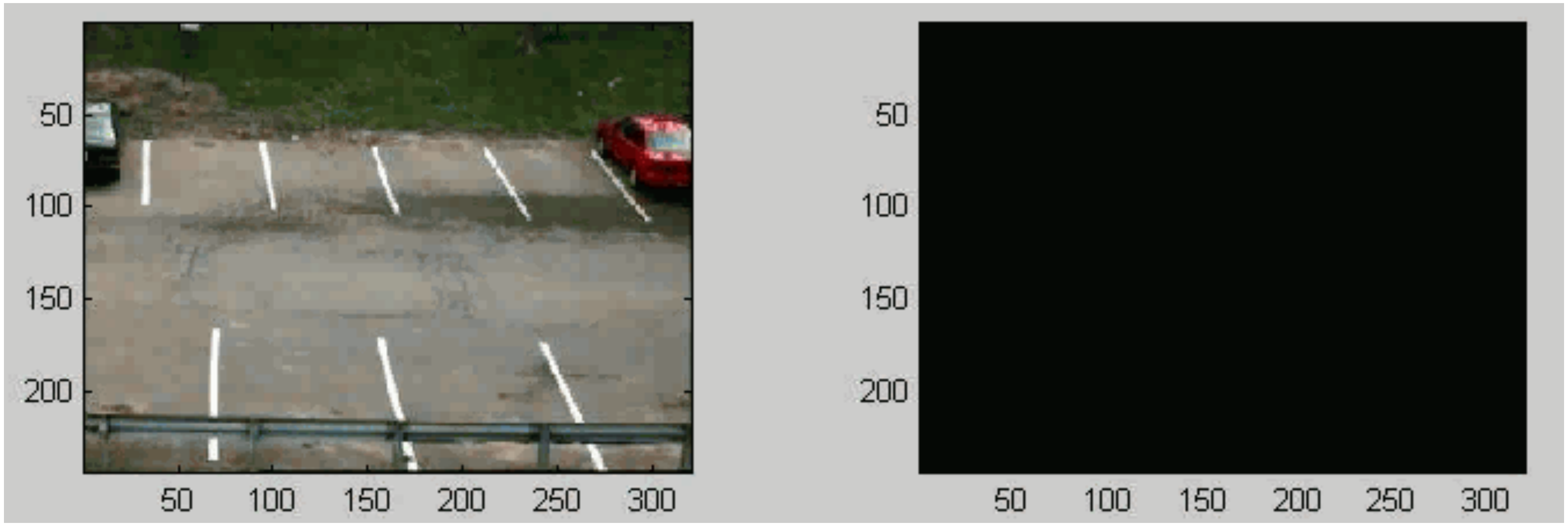
- median filter to remove noisy pixels
- connected components (with gap spanning)
- Size filter to remove small regions

Gap Spanning Connected Components



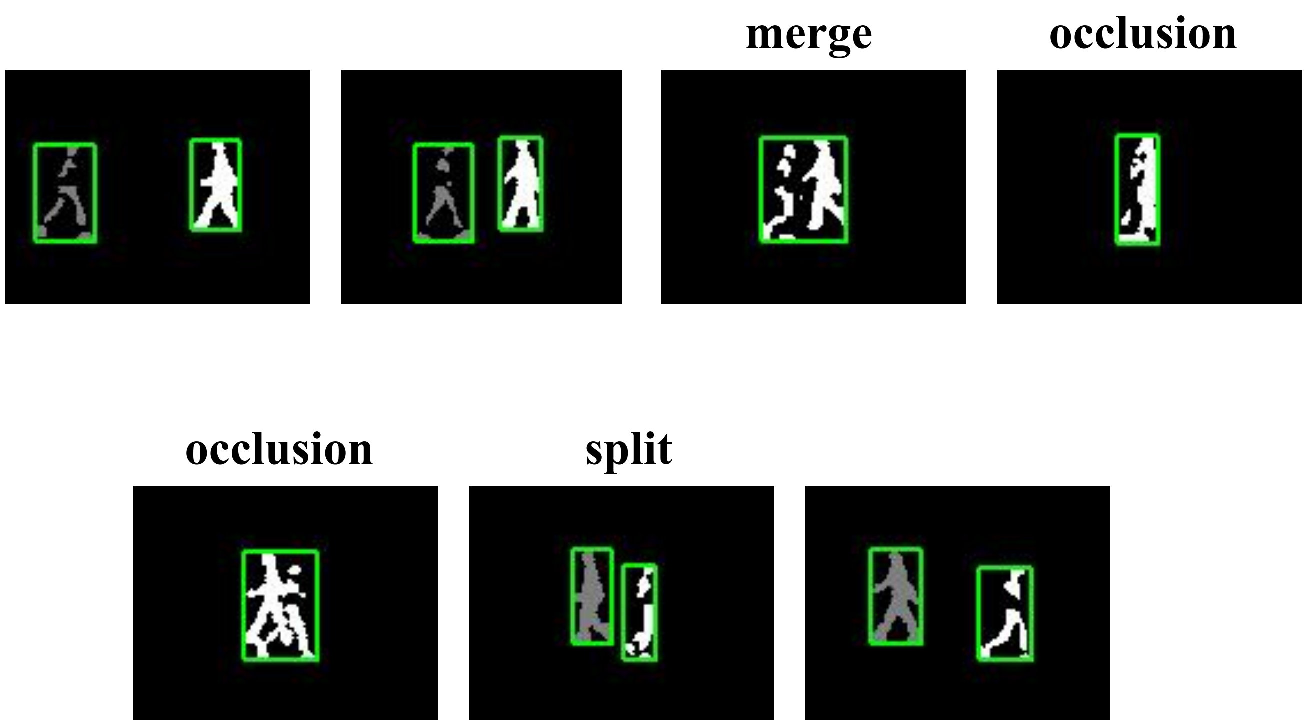
Bounding box = smallest rectangle containing all pixels on the object.

Detected Blobs



movie

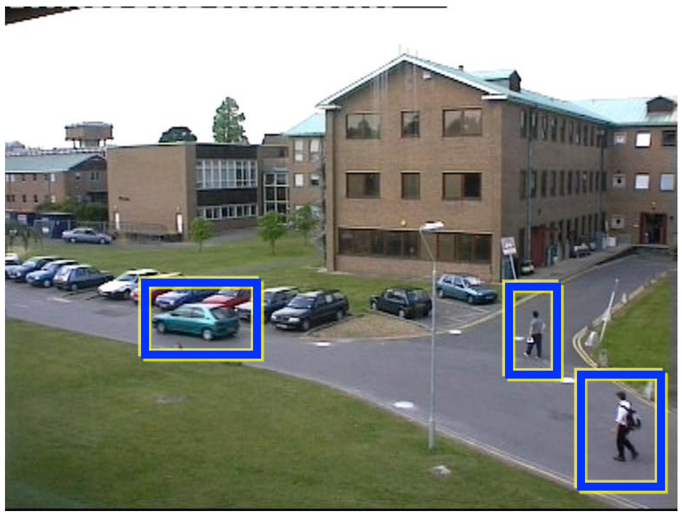
Blob Merge/Split

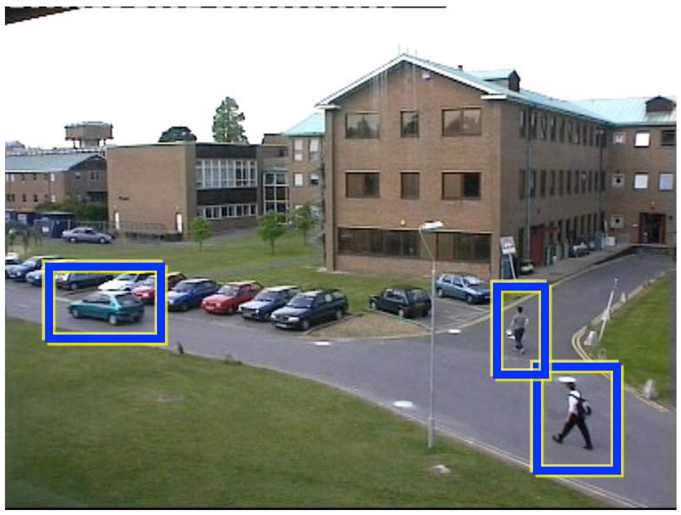


When two objects pass close to each other, they are detected as a single blob. Often, one object will become occluded by the other one. One of the challenging problems is to maintain correct labeling of each object after they split again.

Tracking/ Data Association

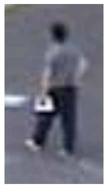
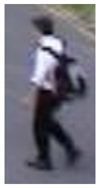
Match object detections across frames





Linear Assignment Formulation

Form a matrix of pairwise similarity scores

			Frame T+1	
		.11	$\boxed{\textbf{.95}}$	**.23**
Frame T		**.85**	**.25**	$\boxed{\textbf{.89}}$
		$\boxed{\textbf{.90}}$	**.12**	**.81**

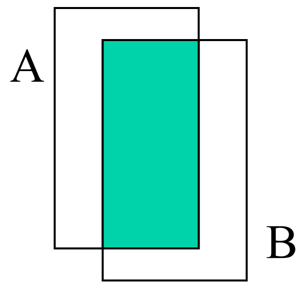
We want to choose one match from each row and column to maximize sum of scores

Affinity (Similarity) Scores

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

Commonly used features: location , size / shape, velocity, appearance

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

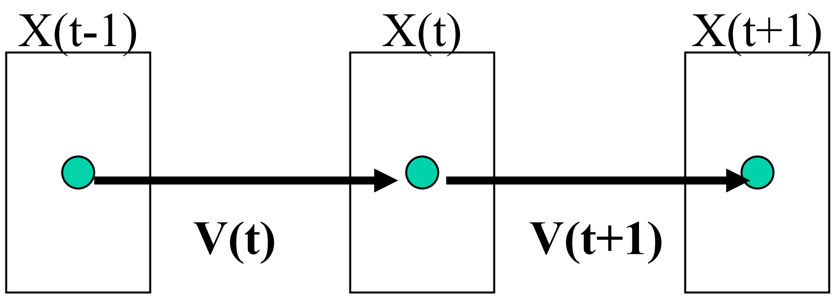


$$\textbf{score} = \frac{2 * \text{area}(A \text{ and } B)}{\text{area}(A) + \text{area}(B)}$$

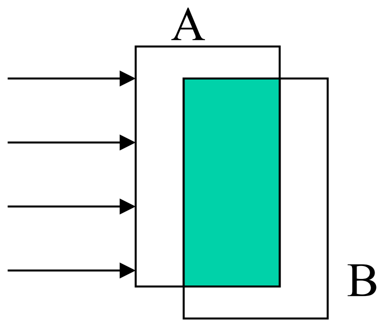
A = bounding box at time t
B = bounding box at time t+1

Smooth Motion

It is common to assume that objects move with constant velocity



constant velocity assumes V(t) = V(t+1)

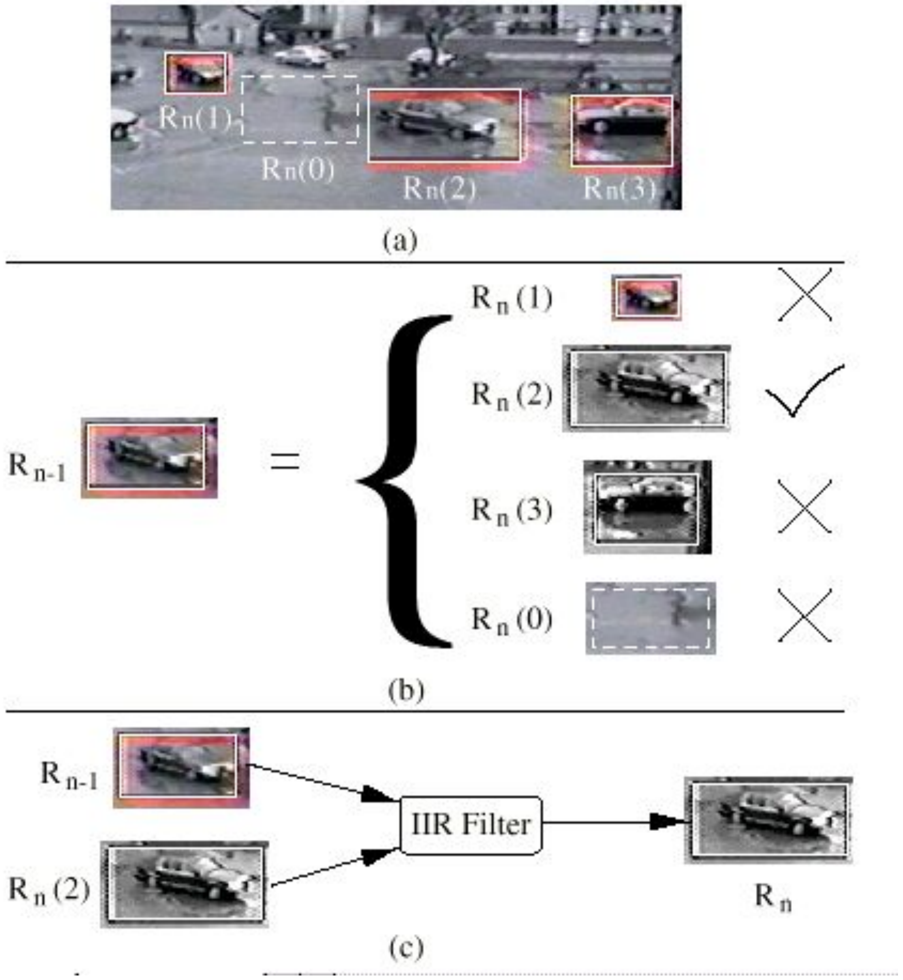


$$\textbf{score} = \frac{\textbf{2 * area(A and B)}}{\textbf{area(A) + area(B)}}$$

A = bounding box at time t, adjusted by velocity V(t)
B = bounding box at time t+1

Appearance Information

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

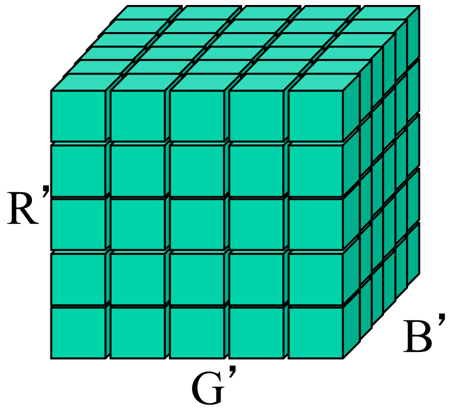


Extract blobs

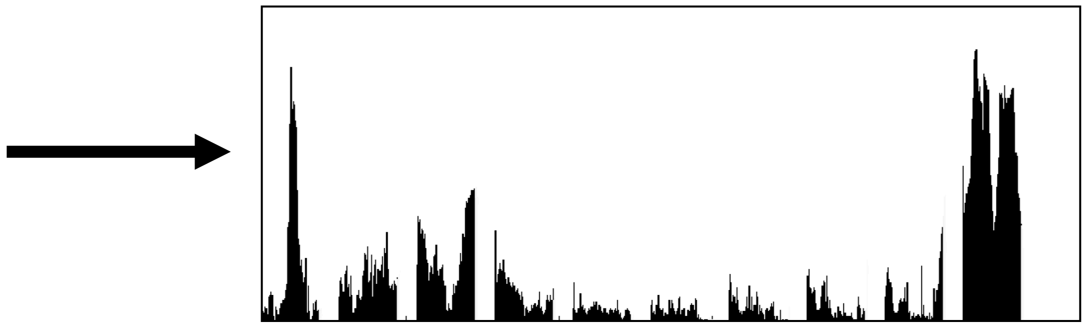
Data association via normalized correlation.

Update appearance template of blobs

Appearance via Color Histograms



discretize



Color distribution (1D histogram normalized to have unit weight)

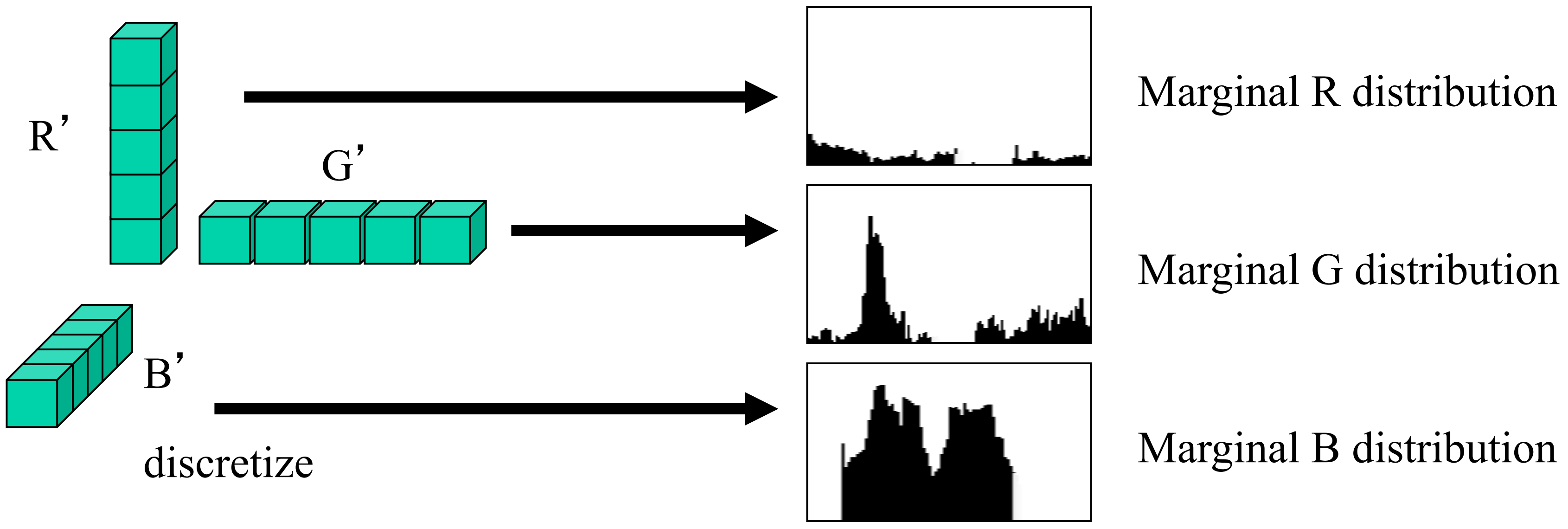
R' = R << (8 - nbits)
G' = G << (8 - nbits)
B' = B << (8 - nbits)

Total histogram size is (2^(8-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.

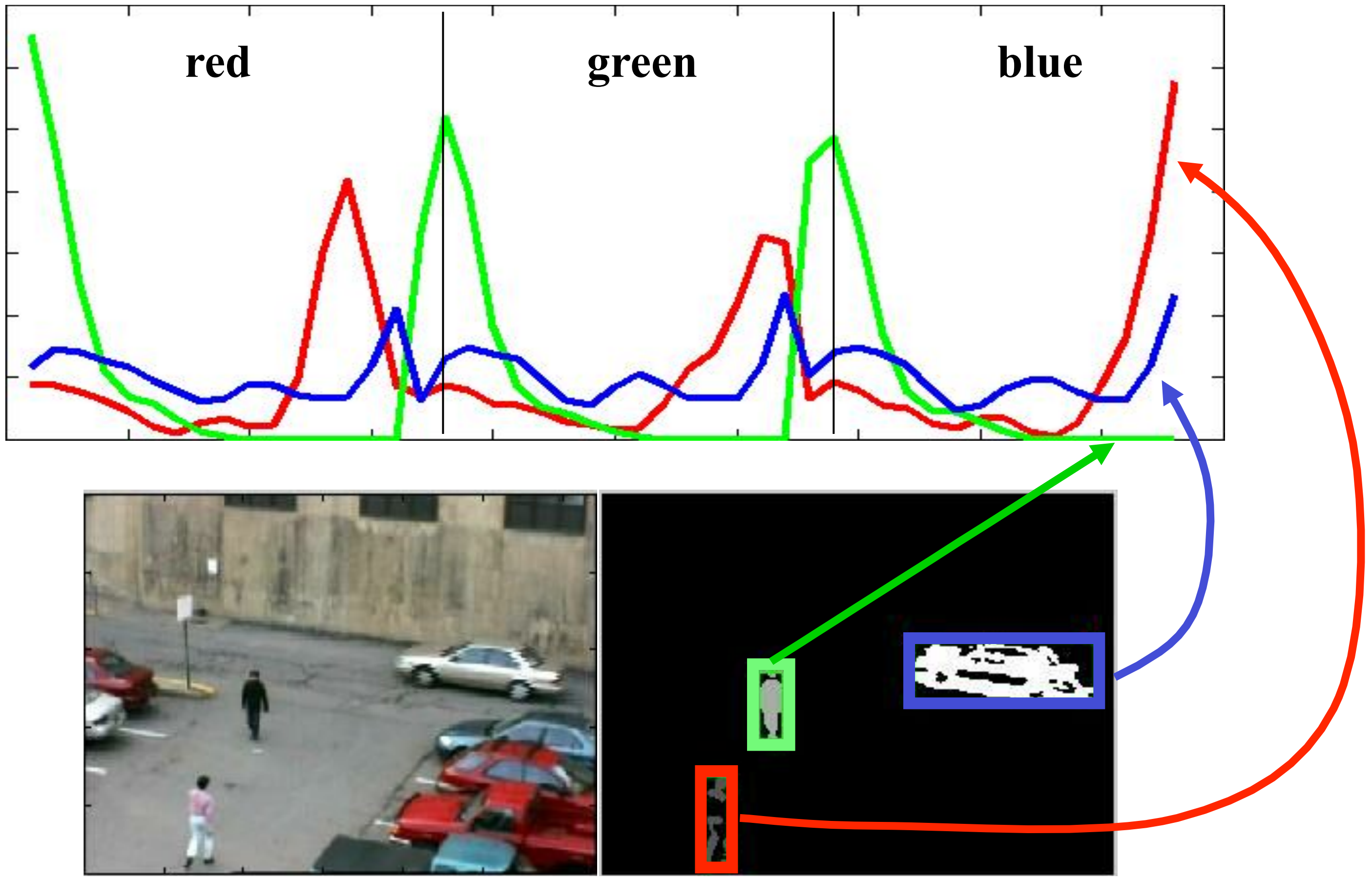


$$R' = R \ll (8 - \text{nbits})$$
$$G' = G \ll (8 - \text{nbits})$$
$$B' = B \ll (8 - \text{nbits})$$

Total histogram size is 3*(2^(8-nbits))

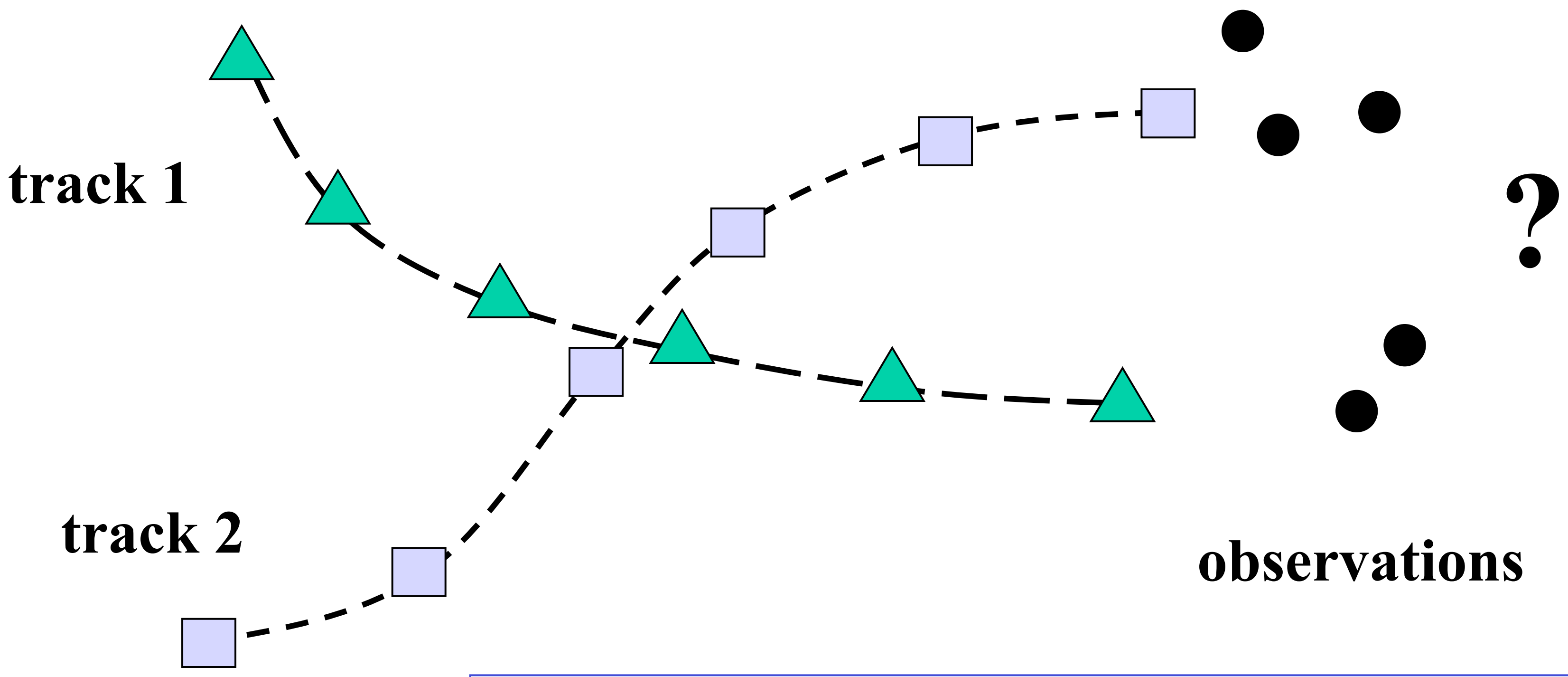
example, 4-bit encoding of R,G and B channels yields a histogram of size 3*16 = 48.

Color Histograms



Tracking / Data Association

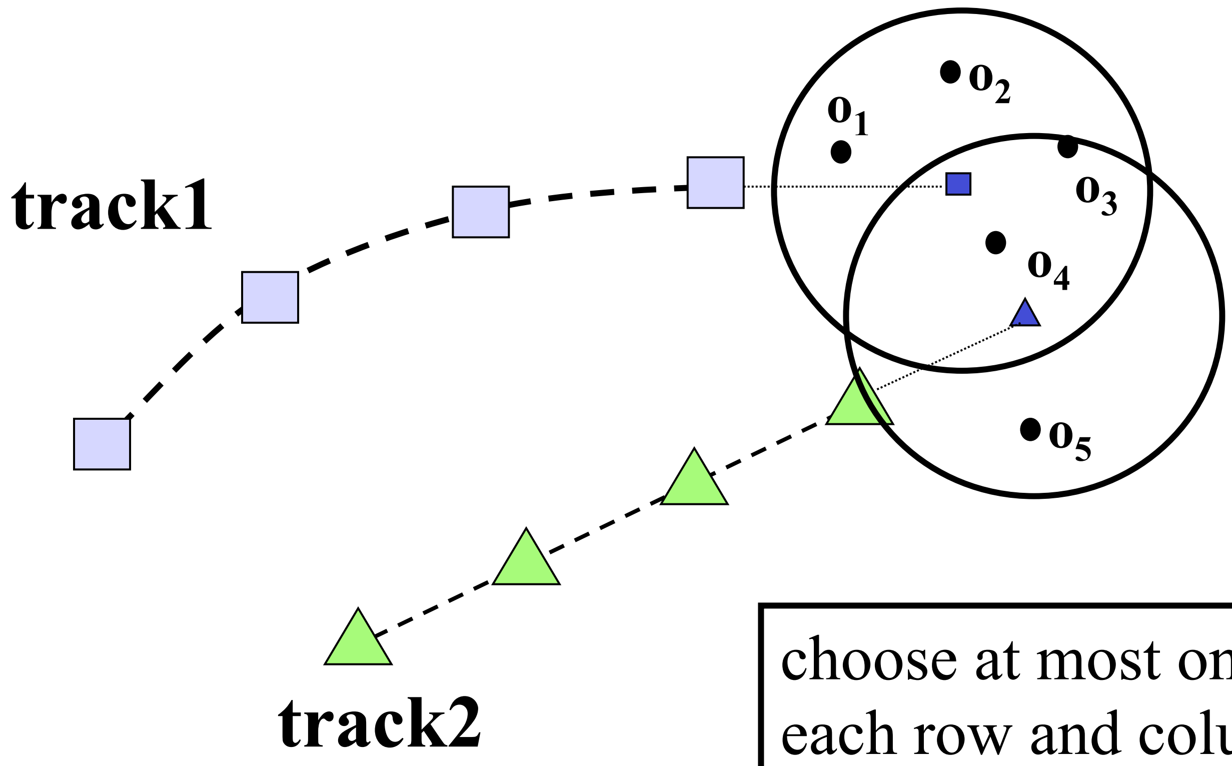
Match a current set of trajectories to object detections in the next frame



How to determine which observations to add to which track?

Linear Assignment Formulation

Extend each trajectory (motion prediction) and assign scores to each observation based on inverse distance such that closer observations get higher numbers.



	ai1	ai2
1	3.0	
2	5.0	
3	$\boxed{6.0}$	1.0
4	9.0	$\boxed{8.0}$
5		3.0

choose at most one match in each row and column to maximize sum of scores.

Linear Assignment Problem

Aka the "Marriage Problem"

Simplest form:
1) given k boys and k girls
2) ask each boy to rank the girls in order of desire
3) ask each girl to also rank order the boys
4) The marriage problem determines the pairing of boys and girls to maximize sum of overall pairwise rankings.

Note: in general you may not get your most desirable spouse, but you rarely get the least, and average bliss across society is maximized.

Linear Assignment Problem

Mathematical Definition

maximize: $$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_{ij}$$

subject to:
$$\left.\begin{aligned} &\textstyle\sum_j x_{ij} = 1; \quad i = 1,2,\ldots,n \\ &\textstyle\sum_i x_{ij} = 1; \quad j = 1,2,\ldots,n \\ &x_{ij} \in \{0,1\} \end{aligned}\right\}$$

constraints that say
X is a permutation matrix

The permutation matrix ensures that we only match up one object from each row and from each column.

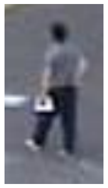
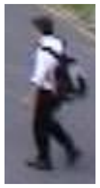
note: alternately, we can minimize costs rather than maximize weights

minimize: $$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

Emphasis

We are solving for a permutation matrix $\{ x_{ij} \}$

Frame T+1

		.11 **0**	$\boxed{.95}$ **1**	.23 **0**
Frame T		.85 **0**	.25 **0**	$\boxed{.89}$ **1**
		$\boxed{.90}$ **1**	.12 **0**	.81 **0**

How to Solve LAP

Try all possible combinations?

	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

Too slow for big problems.
There are N! of them.

Greedy Solution to LAP

Find the largest score.
Remove scores in same row and column from consideration
(that is, enforcing the 1-1 matching constraints)
Repeat

$$
\begin{array}{c|ccccc}
 & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\
1 & \boxed{0.95} & \cancel{0.76} & \cancel{0.62} & \cancel{0.41} & \cancel{0.06} \\
2 & \cancel{0.23} & \cancel{0.46} & \cancel{0.79} & \boxed{0.94} & \cancel{0.35} \\
3 & \cancel{0.61} & \cancel{0.02} & \boxed{0.92} & \cancel{0.92} & \cancel{0.81} \\
4 & \cancel{0.49} & \boxed{0.82} & \cancel{0.74} & \cancel{0.41} & \cancel{0.01} \\
5 & \cancel{0.89} & \cancel{0.44} & \cancel{0.18} & \cancel{0.89} & \boxed{0.14}
\end{array}
$$

Score = .95+.94+.92+.82+.14 = 3.77

Is this the best we can do?

Greedy Solution to LAP

No!

$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \boxed{0.95} & 0.76 & 0.62 & 0.41 & 0.06 \\
2 & 0.23 & 0.46 & 0.79 & \boxed{0.94} & 0.35 \\
3 & 0.61 & 0.02 & \boxed{0.92} & 0.92 & 0.81 \\
4 & 0.49 & \boxed{0.82} & 0.74 & 0.41 & 0.01 \\
5 & 0.89 & 0.44 & 0.18 & 0.89 & \boxed{0.14}
\end{array}$$

Greedy Solution

Score=3.77

$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & \boxed{0.95} & 0.76 & 0.62 & 0.41 & 0.06 \\
2 & 0.23 & 0.46 & \boxed{0.79} & 0.94 & 0.35 \\
3 & 0.61 & 0.02 & 0.92 & 0.92 & \boxed{0.81} \\
4 & 0.49 & \boxed{0.82} & 0.74 & 0.41 & 0.01 \\
5 & 0.89 & 0.44 & 0.18 & \boxed{0.89} & 0.14
\end{array}$$

Optimal Solution

Score=4.26

Greedy method is easy to program; quick to run; and yields “pretty good” solutions in practice.
But it often does not yield the optimal solution.

Hungarian Algorithm

There is an algorithm called Kuhn-Munkres or "Hungarian" algorithm specifically developed to efficiently solve the linear assignment problem.

If you need to solve LAP, you should use it.