



Automatic Tracking of Moving Objects in Video for Surveillance Applications

Manjunath Narayana

Committee:

Dr. Donna Haverkamp (Chair)

Dr. Arvin Agah

Dr. James Miller

*Department of Electrical Engineering and Computer Science
The University of Kansas, Lawrence, KS*



Main Objective

- Problem:

Tracking of moving objects in surveillance video

- Goals:

- Set up a system for automatic tracking at KU
- Add to existing research in the field by developing new algorithms





Outline

- Motivation
- Contributions of this thesis
- Background
- System description
- Shortcomings in basic system
- Improvements
- The Vicinity Factor – a new concept
- Bayesian algorithm for probabilistic track assignment
- Results
- Summary and Conclusions
- Future work





Motivation

- High interest in intelligent video applications
 - Surveillance – Identification of interesting/anomalous behavior
 - Airports, highways, offices
 - Remove or assist humans in these applications
 - Identification/Classification of targets
- Segmentation and tracking of targets are first tasks in all above applications
- Aim: Setting up a tracking framework at KU
 - First step that can result in great deal of research in the future



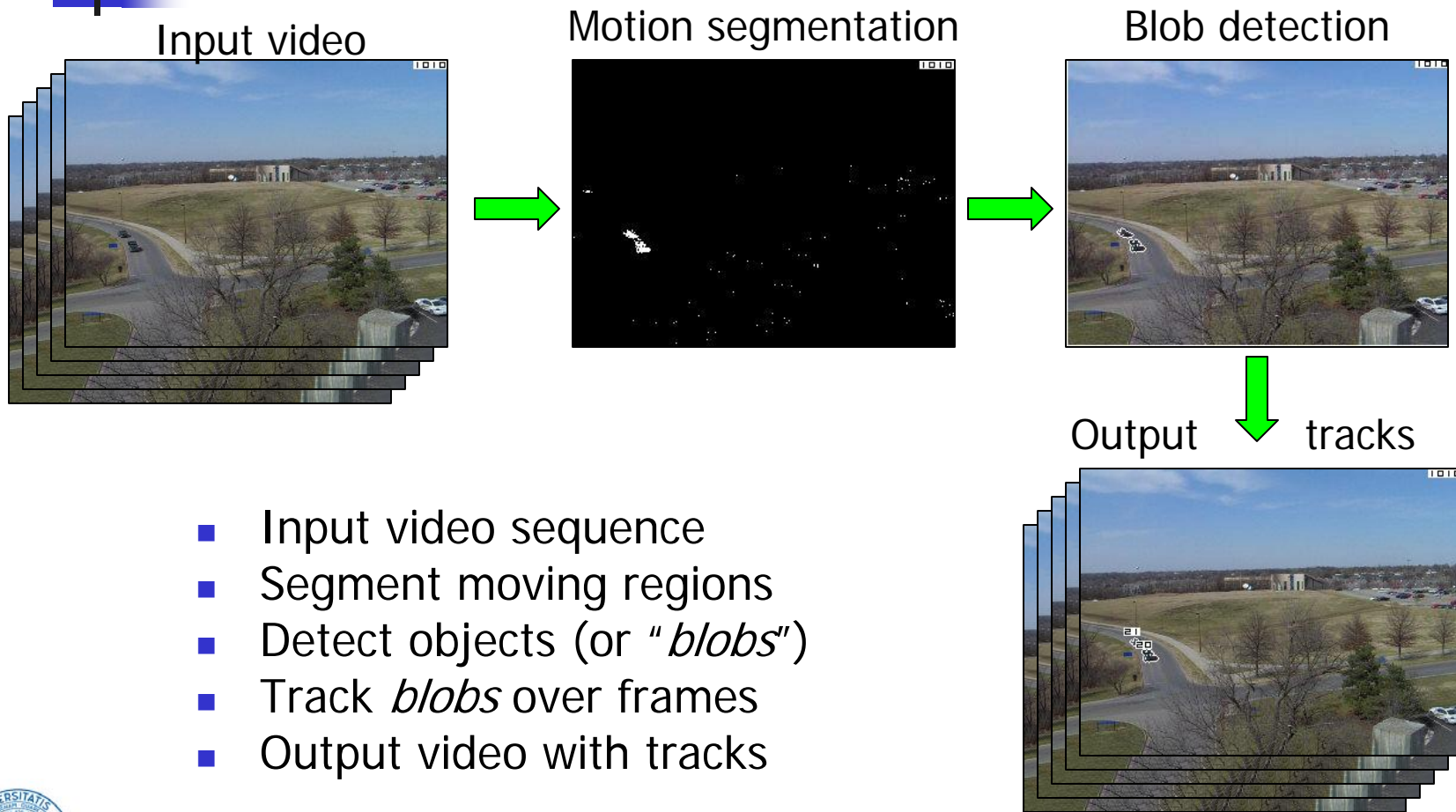


Contributions of this thesis

- Establishment of framework for Object segmentation and tracking
 - Object segmentation – distinguishing the object of interest in the video
 - Tracking – following the object's motion in successive frames
- Development of the Vicinity Factor – a new concept
- Design of a new Bayesian algorithm for tracking
 - paper accepted for CVPR 2007 workshop



System Overview



- Input video sequence
- Segment moving regions
- Detect objects (or "*blobs*")
- Track *blobs* over frames
- Output video with tracks



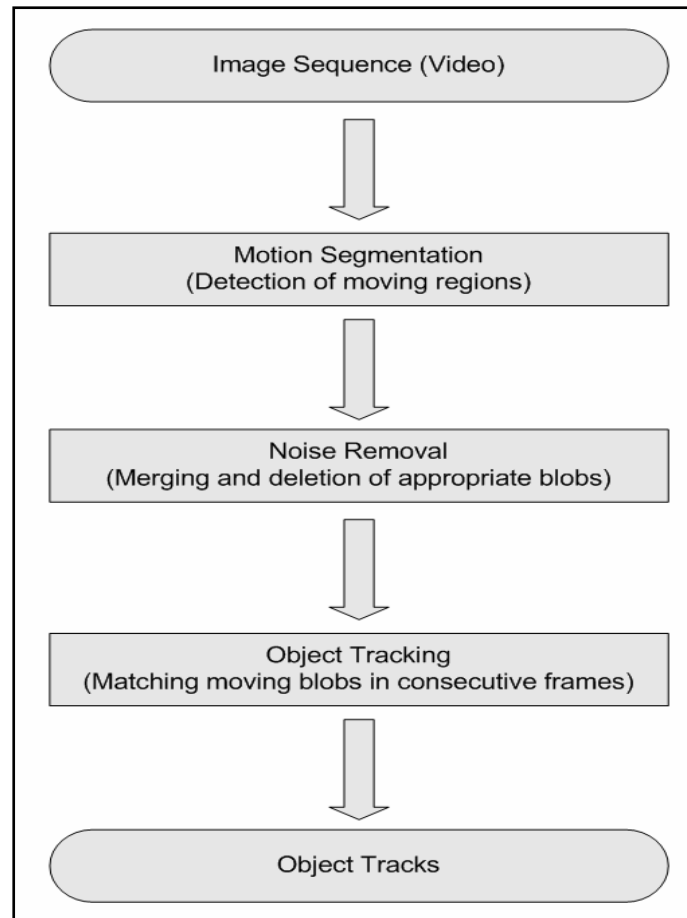


Background

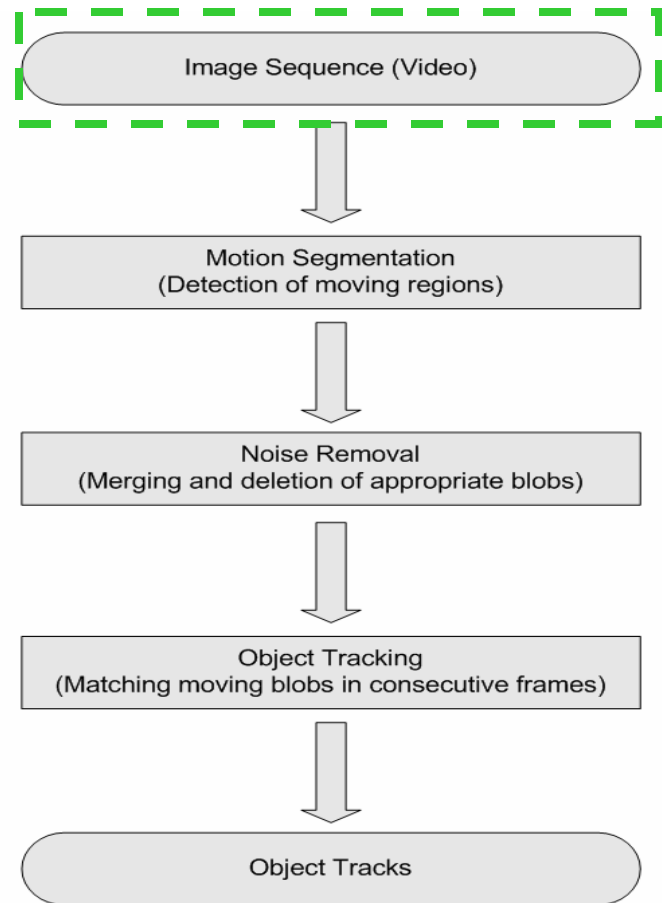
- Object segmentation
 - Color based
 - Edge based
 - Motion based
 - Background pixel modeling
 - Foreground pixel modeling
 - Optical flow methods
 - Gradient methods like moving edge detector
 - We use average background modeling with improvements
- Tracking
 - Once objects detected, find correspondence between objects of previous frame and objects of current frame
 - “Match matrix” of distances between objects is used to find correspondence



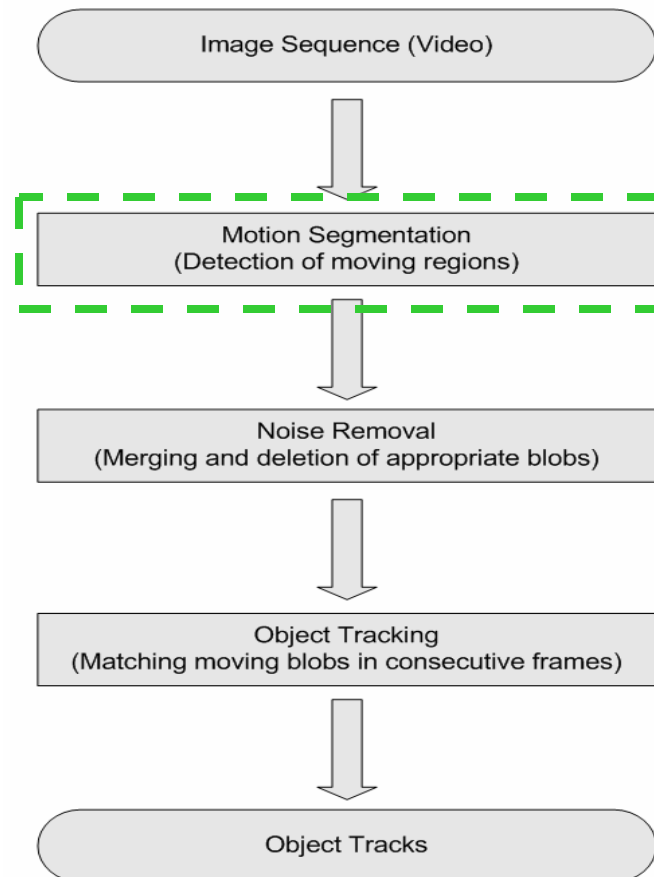
System Architecture



Input Image Sequence



Motion Segmentation



Segmentation method

- Develop a model for the background
- Subtract the current frame from background model
- Threshold the difference image
- Challenges:
 - Outdoor sequences are more difficult
 - Illumination changes
 - Camera jitter and noise

Background model



Current frame



Thresholded difference



Background model

- Cannot use static background (BG) model for outdoors
- Moving average BG model
- The average grayscale intensity value of each pixel over 150 frames



Previous 75 frames

Current frame

Next 75 frames



Moving average BG model

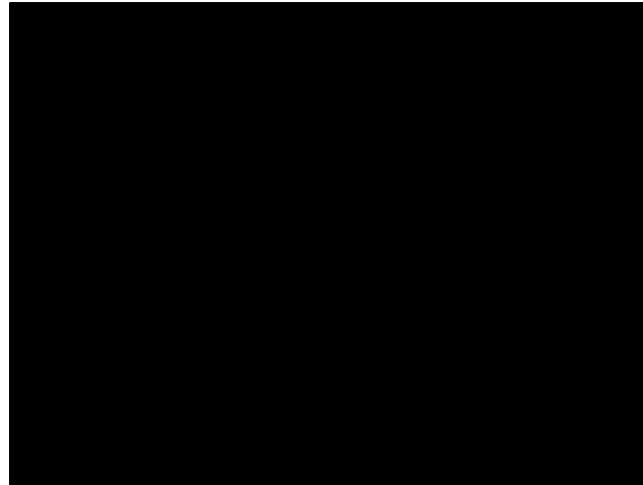
- Simple yet efficient
- Works because fast moving objects do not contribute much to the average value

$$bg^k(y, x) = \frac{1}{N} \sum_{j=k-(N/2)}^{k+(N/2)} I^j(y, x)$$



Thresholding the difference image

$$seg1^k(y, x) = \begin{cases} 1 & , \text{ if } |bg^k(y, x) - I^k(y, x)| > T1 \\ 0 & , \text{ otherwise} \end{cases}$$



Shortcomings of average BG model

- Tails visible in front and behind of objects





Improvement in BG model

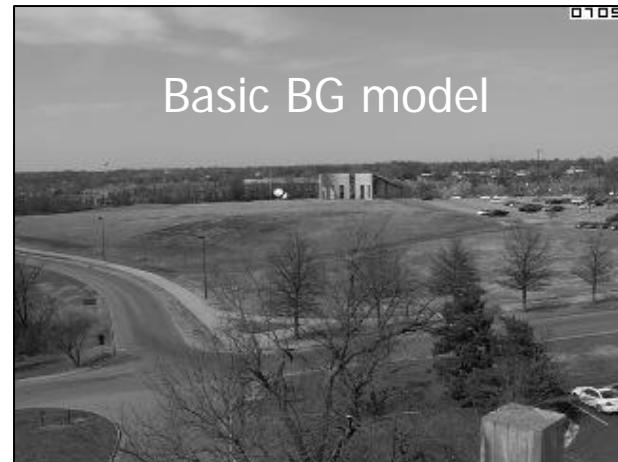
- Intensity of BG does not change much from one frame to next
- The intensity of BG pixel in previous frame is better estimate of BG in current frame than is the average BG
- Secondary BG model (SBG)
- Use of secondary model to refine segmentation



SBG model illustration



Original frame





SBG model calculation

Initialization:

$$sbg^1 = bg^1(y, x)$$

Update:

$$sbg^{k+1}(y, x) = \begin{cases} I^k & \text{If } (y, x) \text{ is a BG} \\ & \text{pixel in frame } k \\ sbg^k & , \text{ otherwise} \end{cases}$$

Segmentation:

$$seg2^k(y, x) = \begin{cases} 1 & , \text{ if } (|sbg^k(y, x) - I^k(y, x)| > T2 \text{ and } seg1^k(y, x) = 1) \\ 0 & , \text{ otherwise} \end{cases}$$



SBG model

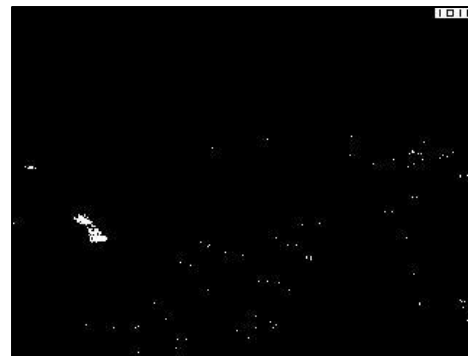
- Sharper picture
- Averaging and blurring effect of basic BG model is not present in SBG model



SBG results



Basic Average BG model results

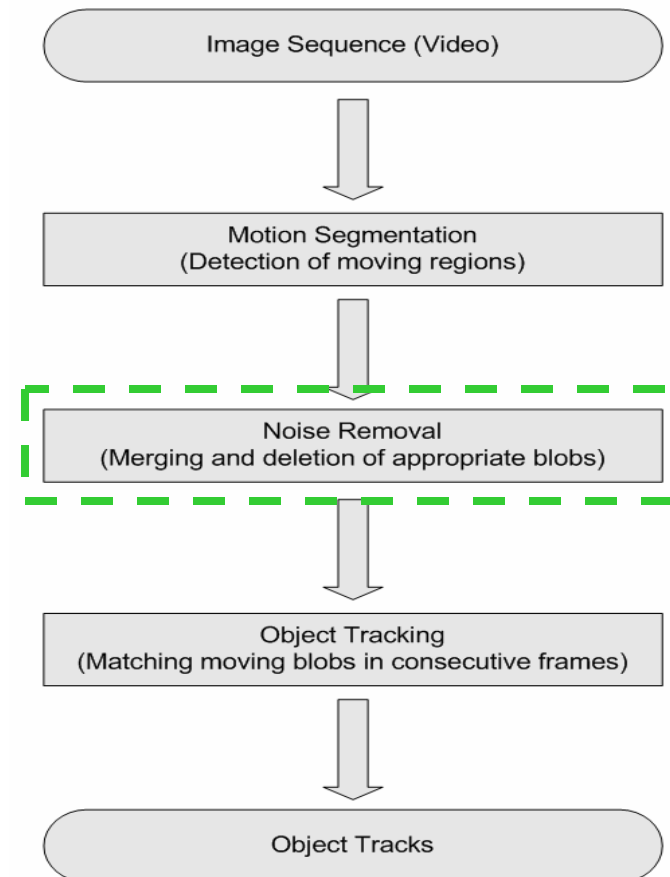


Secondary BG model refined results

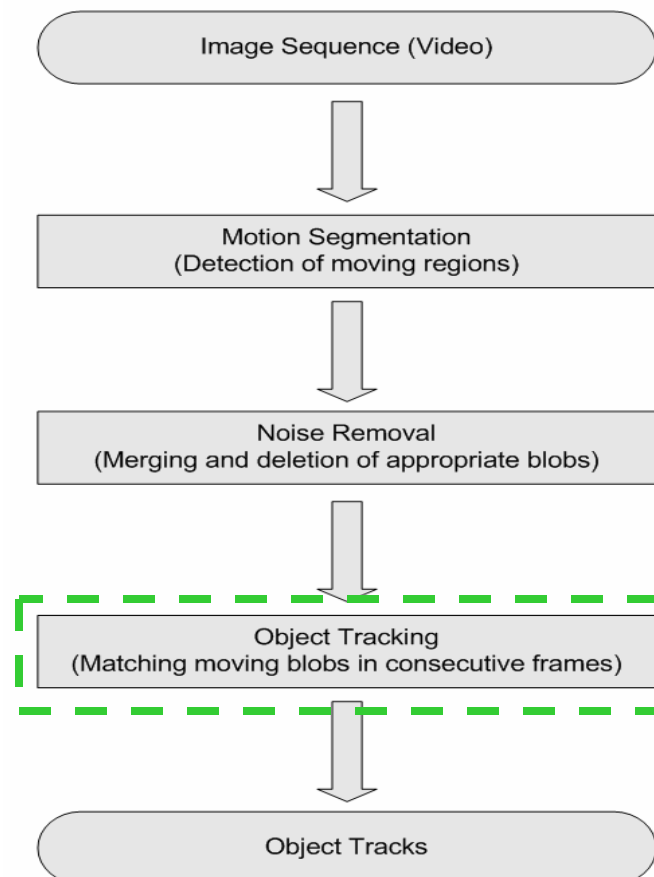


Noise removal

- Remove blobs that are very small
- Group all connected pixels as one object
- Perform Dilation and Erosion morphology operations
 - Small blobs get deleted
 - Envelope is established around each detected object in the color image
- Calculate statistics for each blob (color, position, size)



Object Tracking





Correspondence algorithm

- Once moving objects (*blobs*) detected, find correspondence between tracks of previous frame and *blobs* of current frame
- Most common method: a Match matrix used to determine correspondences
- Euclidean distance between *blobs* commonly used as the measure for a match
- Non-trivial because data is noisy and objects are not predictable
- Objects
 - Appear in the scene
 - Disappear due to exit from scene or occlusion
 - Merge with other objects or the background
 - Break up into two or more objects due to occlusion



Example of tracking problem

Fig 1



previous frame

Fig 2



current frame





Correspondence problem

Tracks: $T^{k-1} = \{t_1^{k-1}, t_2^{k-1}, t_3^{k-1}, \dots, t_{u^{k-1}}^{k-1}\}$

Blobs: $O^k = \{o_1^k, o_2^k, o_3^k, \dots, o_{v^k}^k\}$

- Given track set of previous frame and blob set of current frame, calculate the **Match matrix of Euclidean distances between them** in color space (R,G,B) and Position (Y, X) values





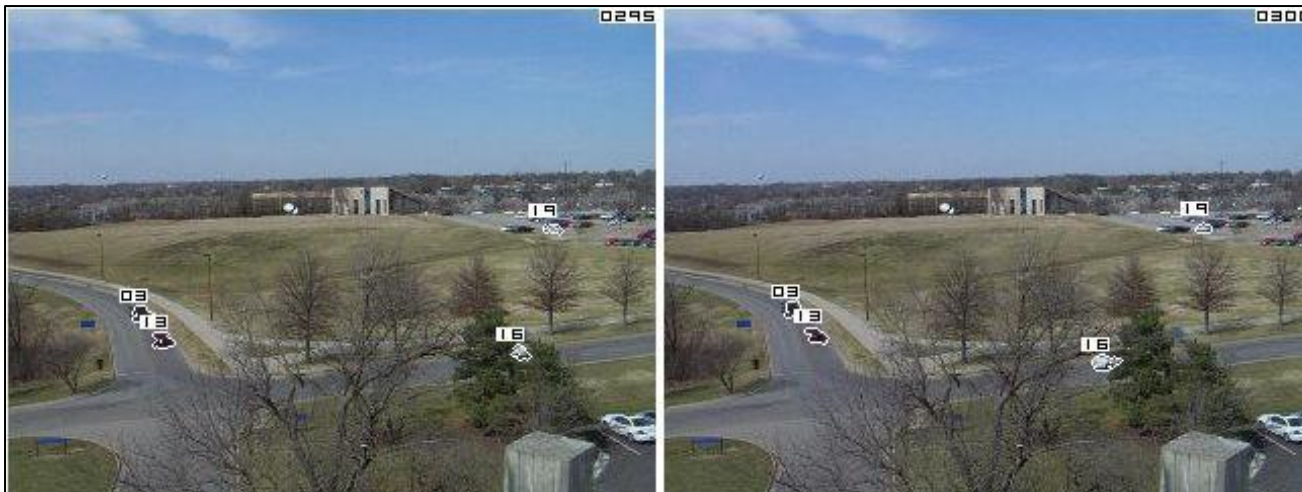
Match matrix

	Blob O ₁	Blob O ₂	Blob O ₃
t ₁	?	?	?
t ₂	?	?	?
t ₃	?	?	?

$$MM_{j,i} = \sqrt{(\Delta Y / Ydim)^2 + (\Delta X / Xdim)^2} \\ + \sqrt{(\Delta R / 255)^2 + (\Delta G / 255)^2 + (\Delta B / 255)^2}$$



Match matrix example



(a) Previous frame - 0295

(b) Current frame - 0300

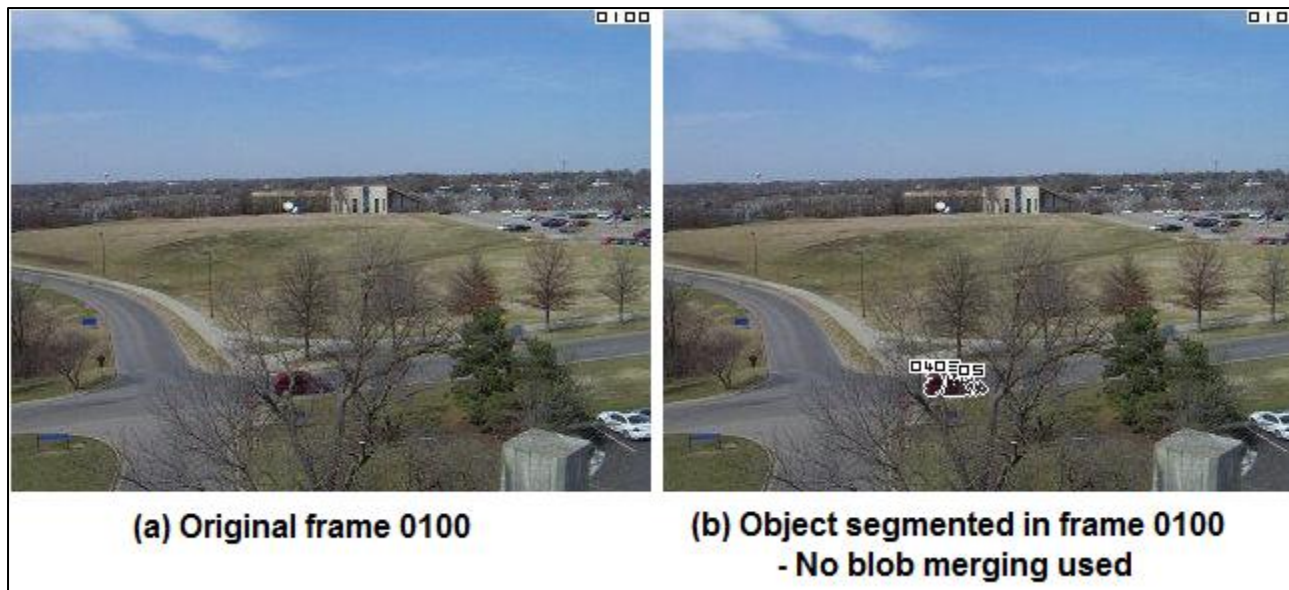
	Blob 1	Blob 2	Blob 3	Blob 4
Track 3	1.18	0.02	0.13	1.10
Track 13	1.21	0.12	0.03	1.10
Track 16	0.36	1.03	1.00	0.21
Track 19	0.11	1.28	1.28	0.33

(c) Match matrix for current frame 0300



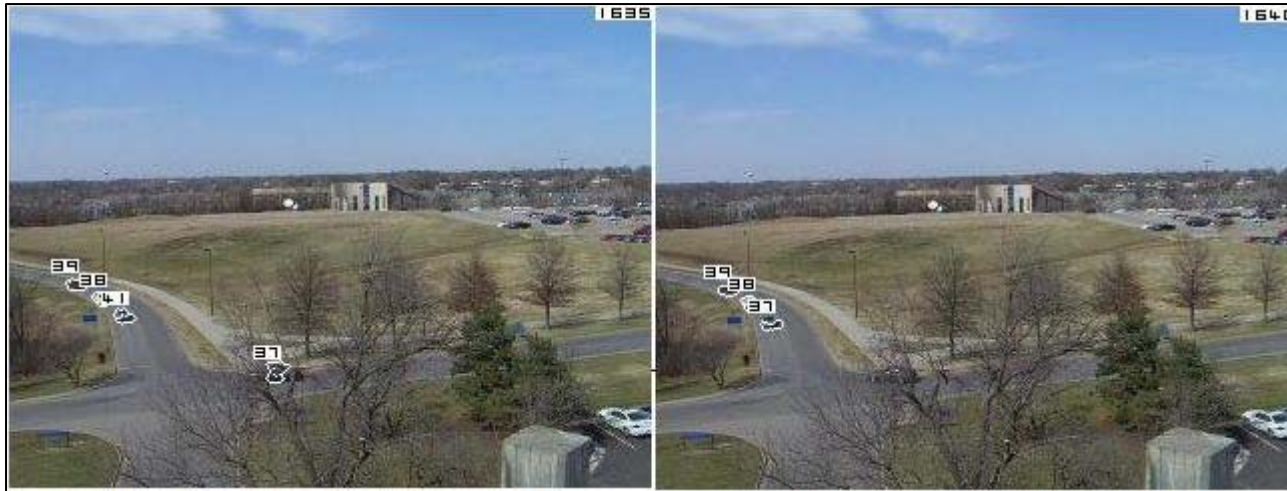
Shortcomings in basic algorithm – broken objects

- Broken objects



Shortcomings in basic algorithm – velocity not used

- Assignment against flow of velocity



Merge module for broken objects

For any two blobs a and b in current frame,

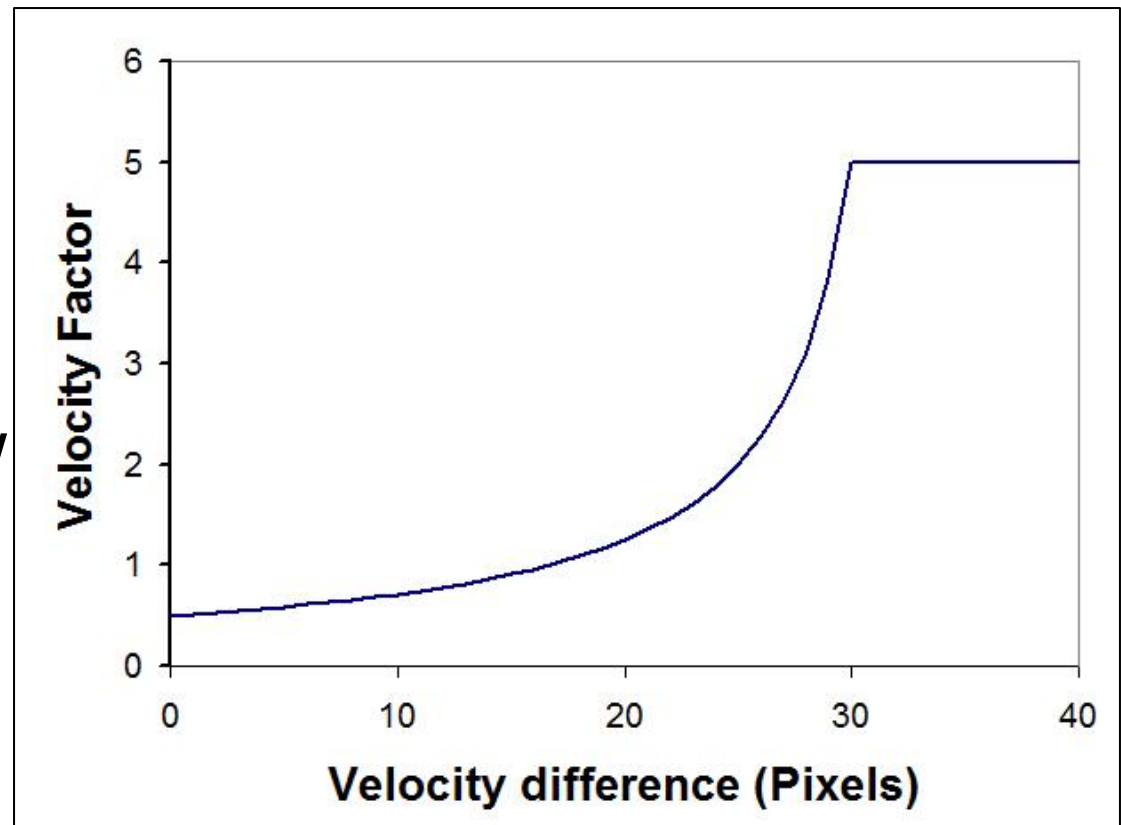
If { $\Delta Y \leq Ythreshold$
and $\Delta X \leq Xthreshold$
and $\Delta R \leq Rthreshold$
and $\Delta G \leq Gthreshold$
and $\Delta B \leq Bthreshold$ }

then, relabel all pixels of blob a with number b



Velocity factor

- Multiply all entries in the Match matrix with a scale factor that reflects velocity



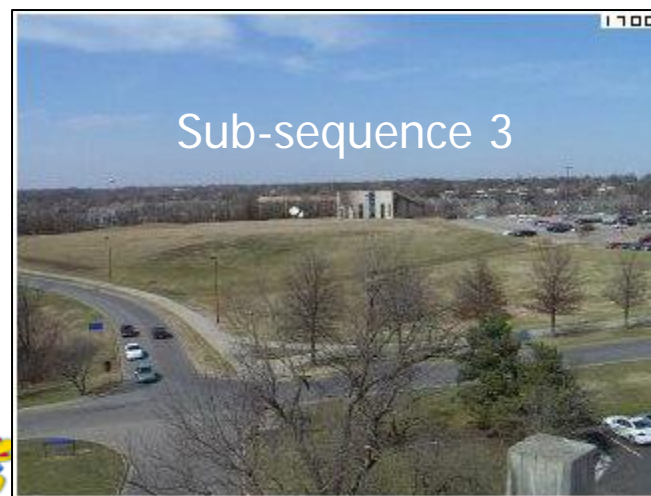
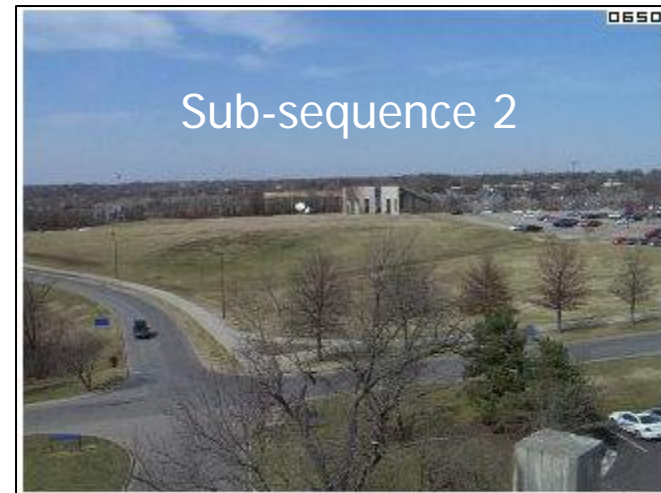
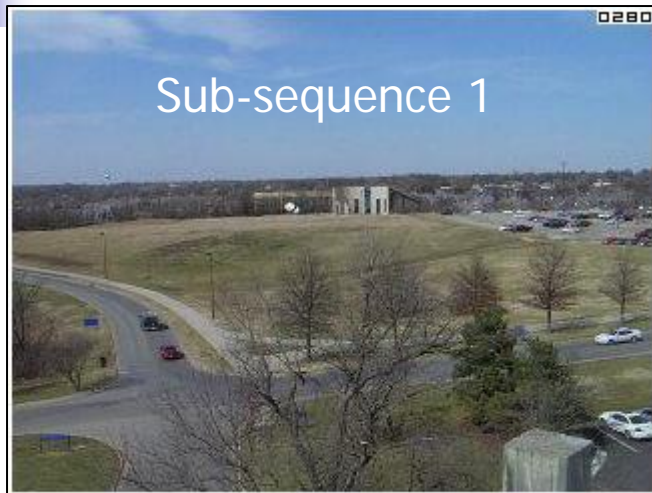


Tracking accuracy

- Accuracy difficult to calculate
- Data is real-life surveillance video
- No ground truth
- Manual observation and analysis used to determine accuracy
- 3 sub-sequences chosen in video
 - Easy, medium, and difficult situations
 - About 700 frames in all



Sub-sequences for analysis





Comparison metrics

- "*number-of-blobs*" error
 - To find accuracy of segmentation
- "*index*" error and "*swap*" error
 - To calculate errors in tracking
 - "*index*" error – number of absolute errors in track numbers
 - "swap" error – number of times a track number for an object changes



Error calculation

Frame	Car # 1			
	Track number assigned	Errors		
		number of blobs	index	swap
0275	6	0	1	0
0280	0	1	0	0
0285	7	0	0	1
0290	7	0	0	0
0295	6,7	1	0	0
0300	7	0	0	0
0305	6	0	1	1
0310	7	0	0	1
0315	7,8,9	2	0	0
Total errors		4	2	3



Segmentation errors

- Compares segmentation results with and without use of blob merging

number-of-blobs errors in segmentation and merging				
[Frames]	Subsequence 1 [0200 – 0400]	Subsequence 2 [0550 – 0830]	Subsequence 3 [1600 – 1750]	Total 1 + 2 + 3
(Number of instances of cars)	(116)	(55)	(121)	(292)
No Blob merging	24	5	29	58
Blob merging (threshold= 10)	23	5	23	51
Blob merging (threshold= 15)	27	2	19	48
Blob merging (threshold= 20)	34	2	20	56



Tracking errors - *index*

- Compares tracking index errors with and without use of Velocity factor

Errors in index assignment				
[Frames] (Number of instances of cars)	Subsequence 1 [0200 – 0400] (116)	Subsequence 2 [0550 – 0830] (55)	Subsequence 3 [1600 – 1750] (121)	Total 1 + 2 + 3 (292)
Without Velocity factor	13	0	32	45
With Velocity factor	11	1	13	25



Tracking errors - *swap*

- Compares tracking swap errors with and without use of Velocity factor

Number of Swap errors in assigning track numbers to blobs in consecutive frames				
[Frames] (Number of instances of cars)	Subsequence 1 [0200 – 0400] (116)	Subsequence 2 [0550 – 0830] (55)	Subsequence 3 [1600 – 1750] (121)	Total 1 + 2 + 3 (292)
Without Velocity factor	5	0	13	18
With Velocity factor	5	2	7	14





Results - summary

- Blob merging is useful
 - Improvement by 17.2%
- Accuracy of blob merging depends on the threshold used
- Velocity factor is very useful
 - Improvement by 44.4%



Depth variation in scenes

- In videos, there is significant change in object distance from camera ("Object Depth", OD) from one part of the image to another
- This affects the position thresholds that are used in the algorithm
- Ex: Car 01 far away from camera compared to Car 02. Need to use different value for threshold for each car
- For instance, the threshold used for blob merging





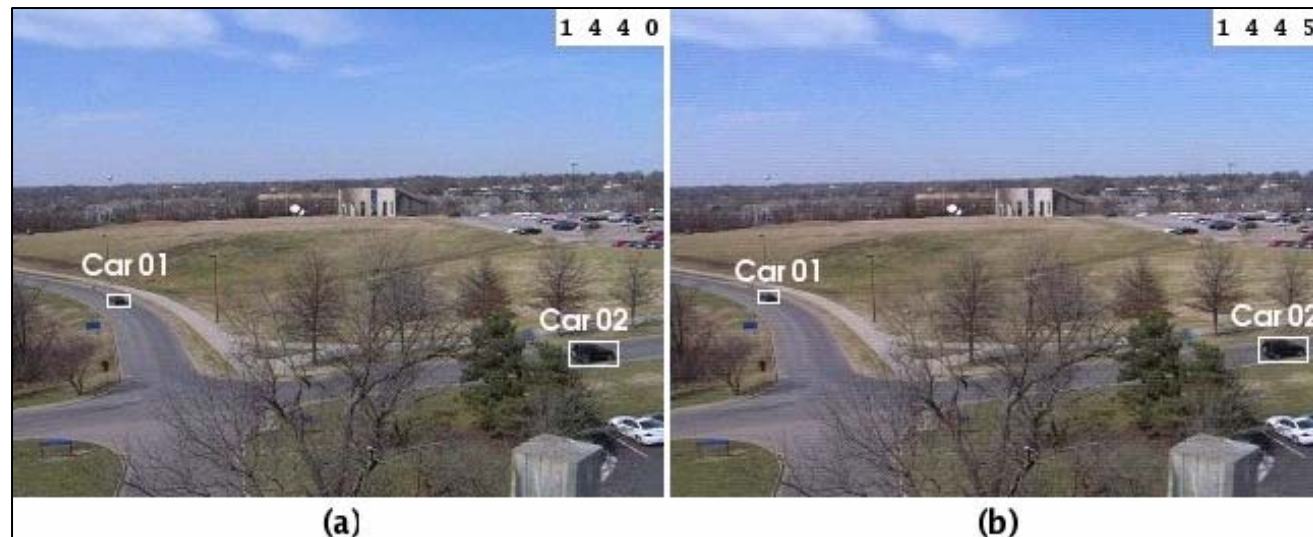
Object Depth variation

- Need to compensate for change in OD
- Can use 3-D modeling or stereo-based approaches
 - Complicated methods
- Alternative Learning approach to solve this problem
 - The Vicinity Factor



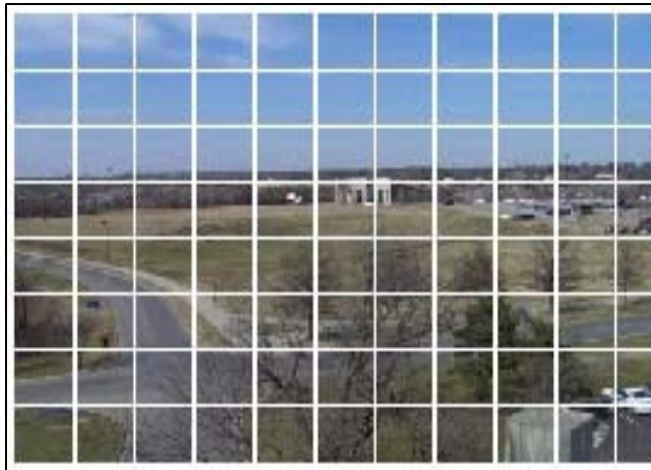
The Vicinity Factor

- Objects closer to camera move more in pixel space than objects far away do
- By observing this variation during tracking, we can learn the relative distances of objects in different parts of the scene



Vicinity Factor (VF) concept

- Break the scene into grids of 30 by 30 pixels
- Set an initial value for VF
- Each time a track is seen in a grid location, observe how much it moves
- If motion is large, then VF for that grid location is increased
- If motion is small, then VF for that grid location is decreased





VF Calculation (1)

Initialize:

$$vicinityY[g_y][g_x] = vicinityX[g_y][g_x] = \frac{1 + maxFactor}{2}$$

where

$vicinityY[g_y][g_x]$ is the VF in Y direction
 $vicinityX[g_y][g_x]$ is the VF in X direction

Update:

For every successfully updated track $t_j^k \in T^k$,

$$\Delta y = y(t_j^k) - y(t_j^{k-1})$$

$$\Delta x = x(t_j^k) - x(t_j^{k-1})$$





VF Calculation (2)

$$\begin{aligned} \text{vicinityY}[g_y][g_x] &= \text{vicinityY}[g_y][g_x]_{old} + \\ &\quad ((\Delta y - 1) \times (\text{maxFactor} - \text{minFactor}) \times \text{constant}) \\ &= \text{vicinityY}[g_y][g_x]_{old} + ((\Delta y - 1) \times (\text{maxFactor} - 1) \times \alpha) \\ &\approx \text{vicinityY}[g_y][g_x]_{old} + ((\Delta y - 1) \times \text{maxFactor} \times \alpha) \end{aligned}$$

$$\text{vicinityX}[g_y][g_x] = \text{vicinityX}[g_y][g_x]_{old} + ((\Delta x - 1) \times \text{maxFactor} \times \alpha)$$

Note that VF Y and VF X will not have the same values



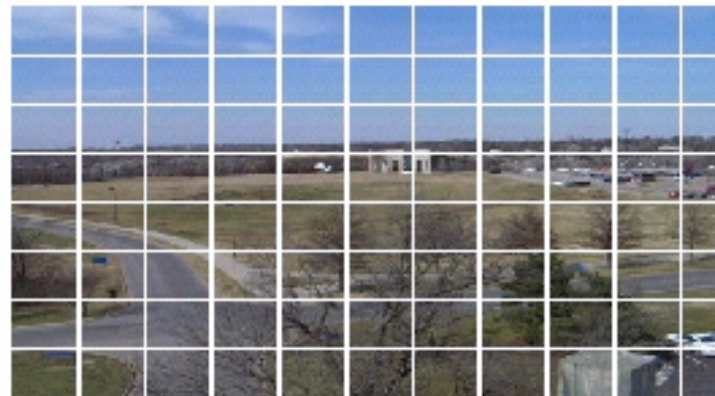
VF matrix example

3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	2	3	2
1	1	1	3	3	3	3	3	3	3	3
3	1	1	3	3	3	3	3	5	5	5
3	3	1	5	5	5	4	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3

(a) Vicinity Factor X

3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	2	3	1
1	1	1	3	3	3	3	3	3	3	3
3	1	1	1	2	2	1	1	1	1	1
3	3	1	1	1	1	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3

(b) Vicinity Factor Y



(c) Scene and grid layout for video sequence 1





Use of VF in blob merging

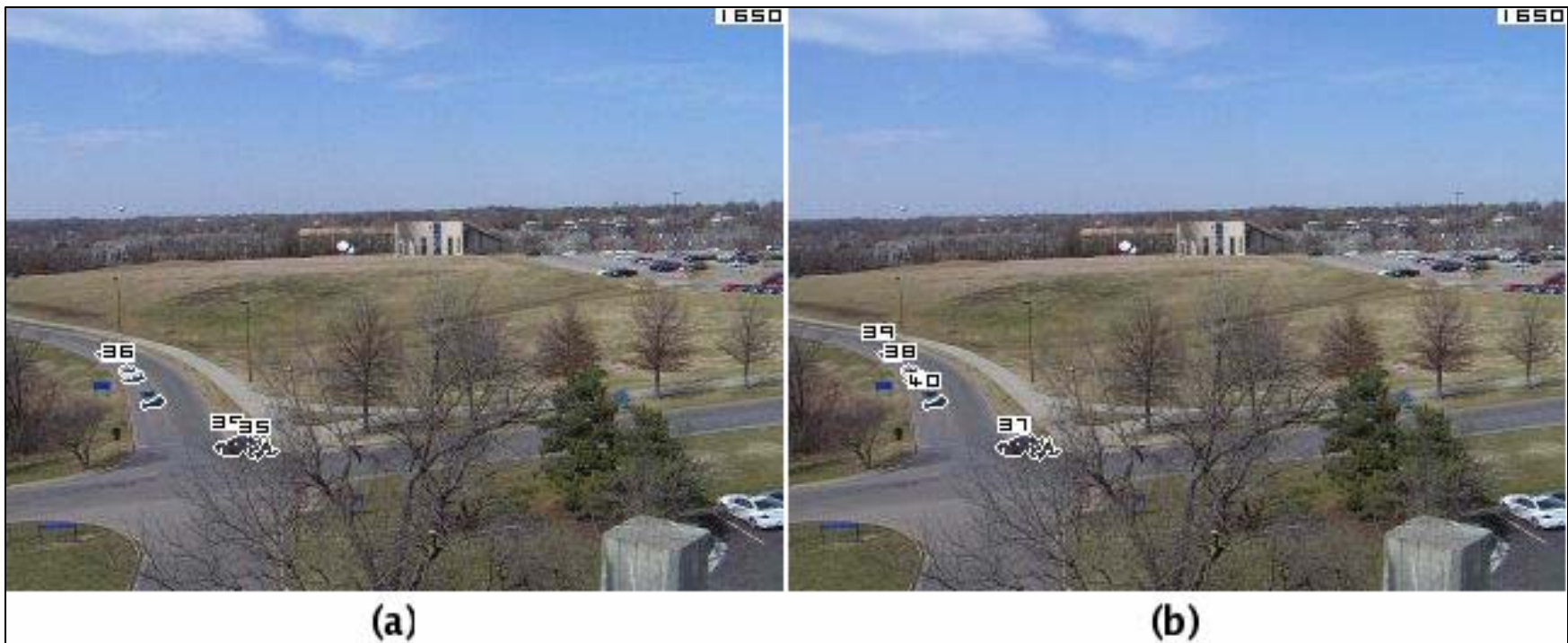
- Used VF as basis for variable thresholds in different parts of the scene for blob merging

Errors in segmentation and merging				
[Frames]	Subsequence 1 [0200 – 0400]	Subsequence 2 [0550 – 0830]	Subsequence 3 [1600 – 1750]	Total 1 + 2 + 3
(Number of instances of cars)	(116)	(55)	(121)	(292)
No Blob merging	24	5	29	58
Blob merging (threshold= 10)	23	5	23	51
Blob merging (threshold= 15)	27	2	19	48
Blob merging (threshold= 20)	34	2	20	56
Blob merging (Vicinity Factor)	23	2	20	45



Use of VF in blob merging (2)

- Example of improvement





Probabilistic track assignment

- Thus far, tracks assigned based on Euclidean distance matrix
- Probabilistic track assignment can lead to better inference in higher level applications
- Bayesian algorithm to track objects was designed
- Work accepted and presented at CVPR 2007 workshop, Minneapolis, 18-23rd June, 2007



Need for Probabilistic track assignment

- Major issue: object data is noisy
- Objects
 - Appear in the scene
 - Disappear due to exit from scene or occlusion
 - Merge with other objects or the background
 - Break up into two or more objects due to occlusion
- A probabilistic algorithm to assign track numbers to objects may be very useful





Our Bayesian Approach

- We propose a Bayesian approach to determine probabilities of match between *blobs*.
- Bayesian approach results in a Belief matrix (of probabilities) instead of a Match matrix (of distances)





Method

- Basic principle:
 - Given a track in previous frame, we expect a *blob* of similar color and position in current frame with some probability
 - Provides basis for Bayesian method
 - Upon observing a *blob* of given color and position, what is the posterior probability that this *blob* belongs on one of the tracks from the previous frame?

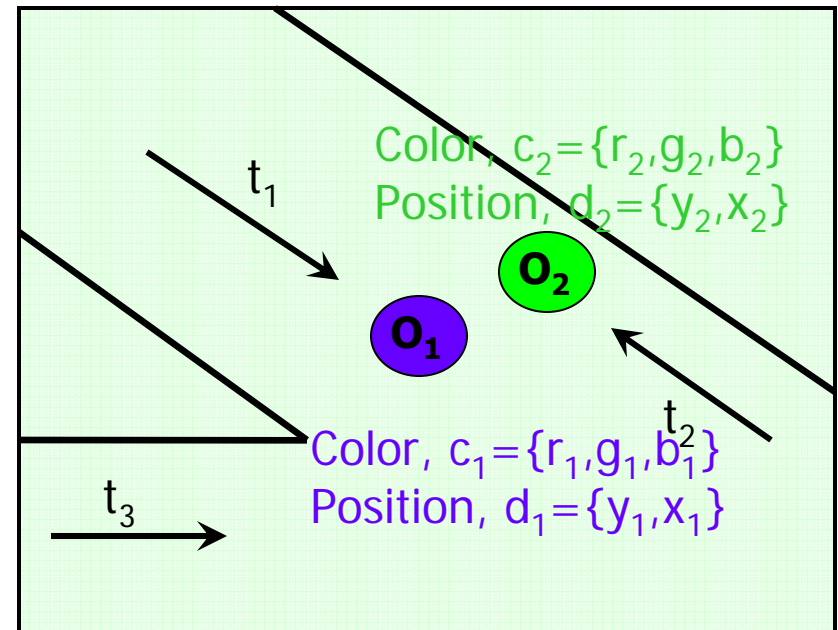


Illustration (1)

- Example:

Blobs O_1 and O_2 seen in current frame

What is the probability that each of these blobs belongs to the tracks in the previous frame?



Probabilistic network for track assignment

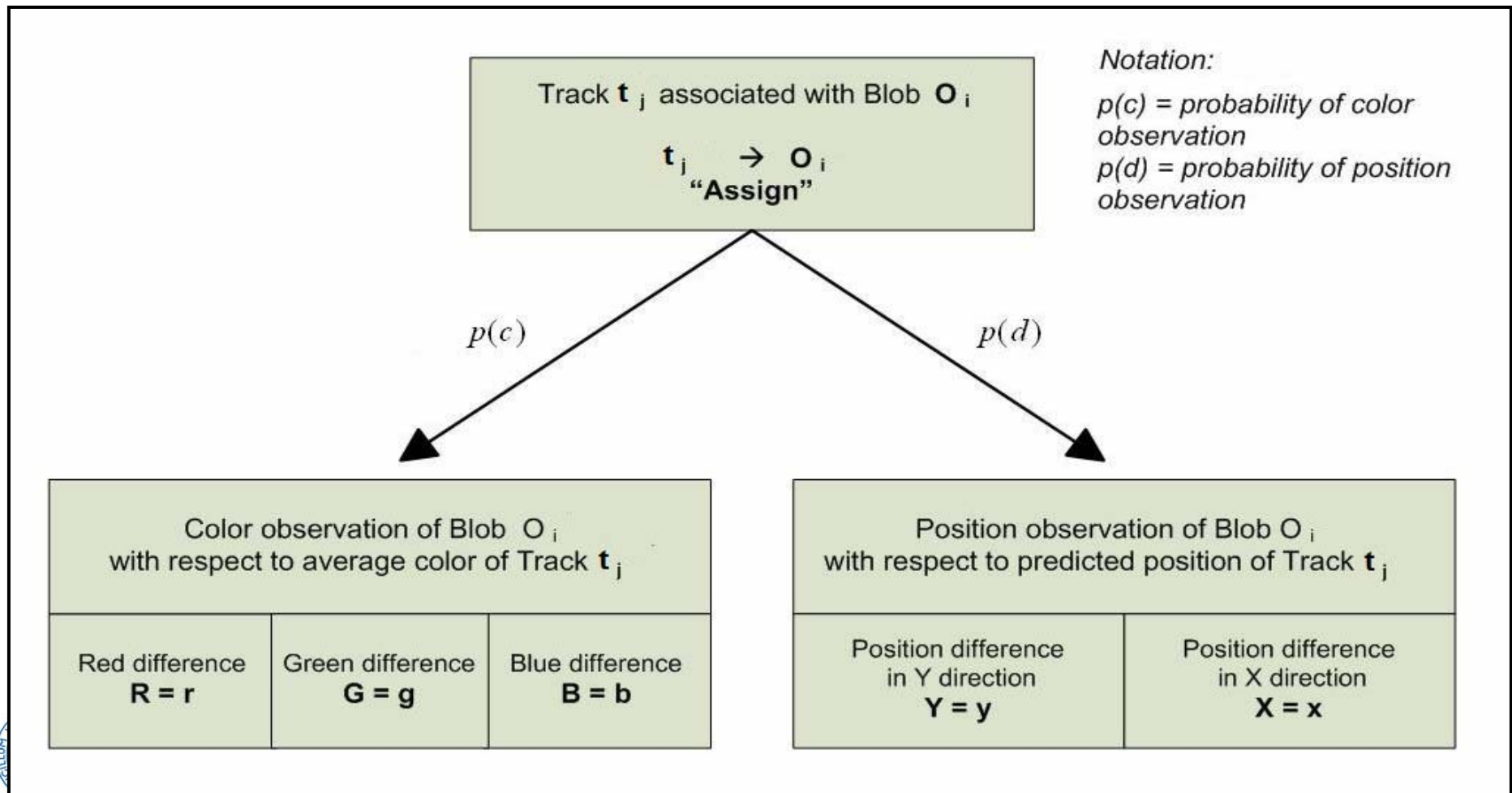
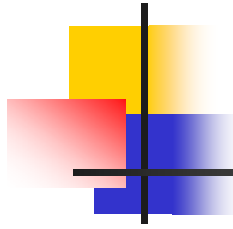


Illustration (2)

- Given three tracks t_1 , t_2 , and t_3 in the previous frame

- There are six probabilities:

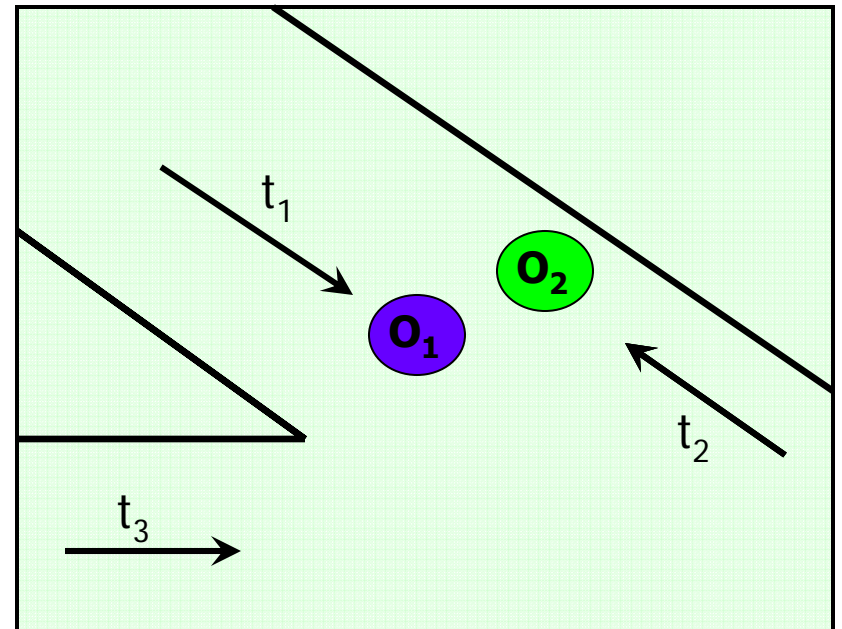
$$p(\text{Assign } t_1 \rightarrow O_1) \quad p(\text{Assign } t_1 \rightarrow O_2)$$

$$p(\text{Assign } t_2 \rightarrow O_1) \quad p(\text{Assign } t_2 \rightarrow O_2)$$

$$p(\text{Assign } t_3 \rightarrow O_1) \quad p(\text{Assign } t_3 \rightarrow O_2)$$

- Each track can either be assigned to blob O_1 or O_2 , or may be lost in the frame

- Produces initial Belief matrix with equal likelihood for all cases



	Blob O_1	Blob O_2	"lost"
t_1	0.33	0.33	0.33
t_2	0.33	0.33	0.33
t_3	0.33	0.33	0.33



Illustration (3)

- Consider track t_1 and update first element in matrix

$$p(\text{Assign } t_1 \rightarrow O_1)$$

- Observations:
color (c_1), position(d_1)

To find:

$$p(\text{Assign } t_1 \rightarrow O_1 | c_1, d_1)$$

- Assumption - color and position observations are independent:

$$p(\text{Assign } t_1 \rightarrow O_1 | c_1, d_1) =$$

$$p(\text{Assign } t_1 \rightarrow O_1 | c_1) \times p(\text{Assign } t_1 \rightarrow O_1 | d_1)$$

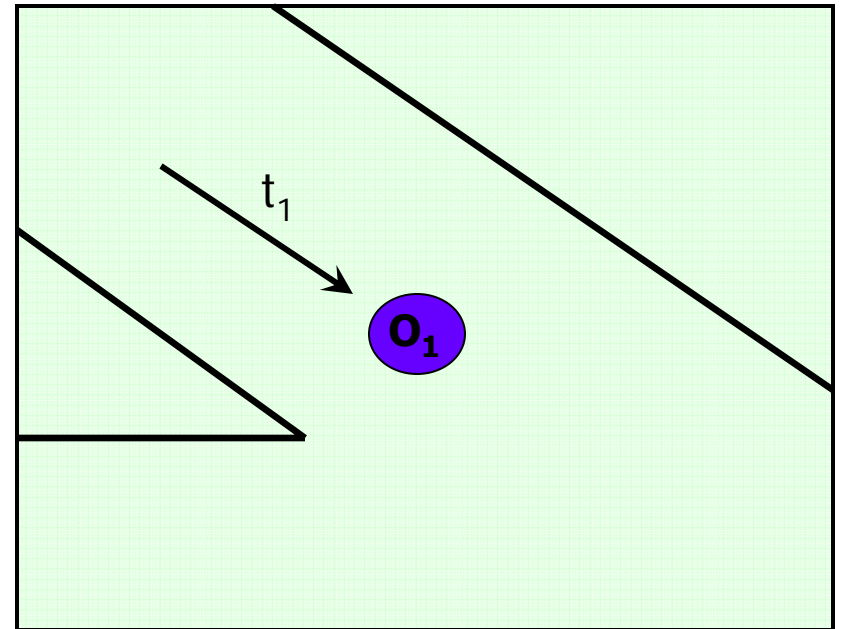
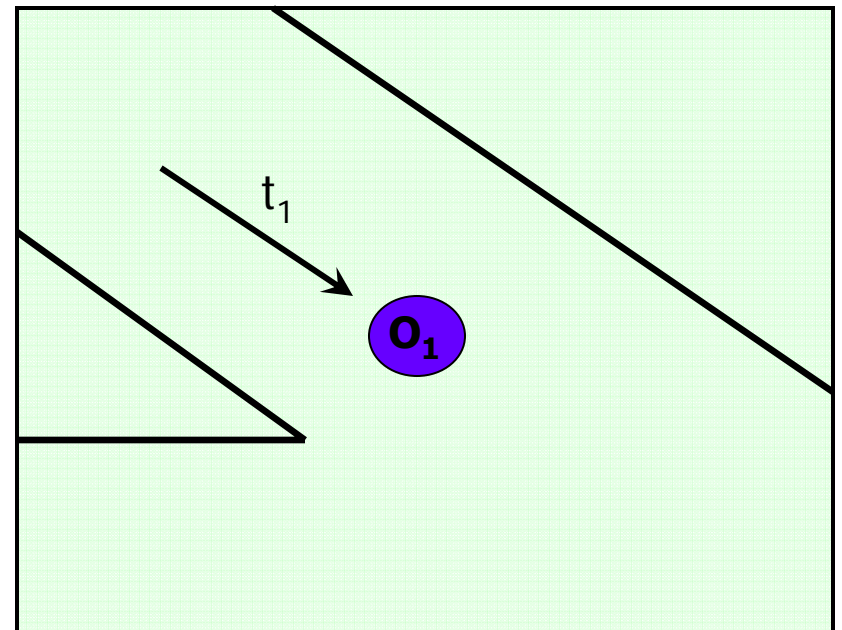


Illustration (4)

- First, color observation for first element in matrix, c_1
- By Bayes formula:

$$p(\text{Assign } t_1 \rightarrow O_1 | c_1) = \frac{p(c_1 | \text{Assign } t_1 \rightarrow O_1) \times p(\text{Assign } t_1 \rightarrow O_1)}{p(c_1)}$$

- The Belief matrix is updated



	Blob O_1	Blob O_2	"lost"
t_1	0.60	0.20	0.20
t_2	0.33	0.33	0.33
t_3	0.33	0.33	0.33



Illustration (5)

- Next, position observation for first element in matrix, d_1 , is considered

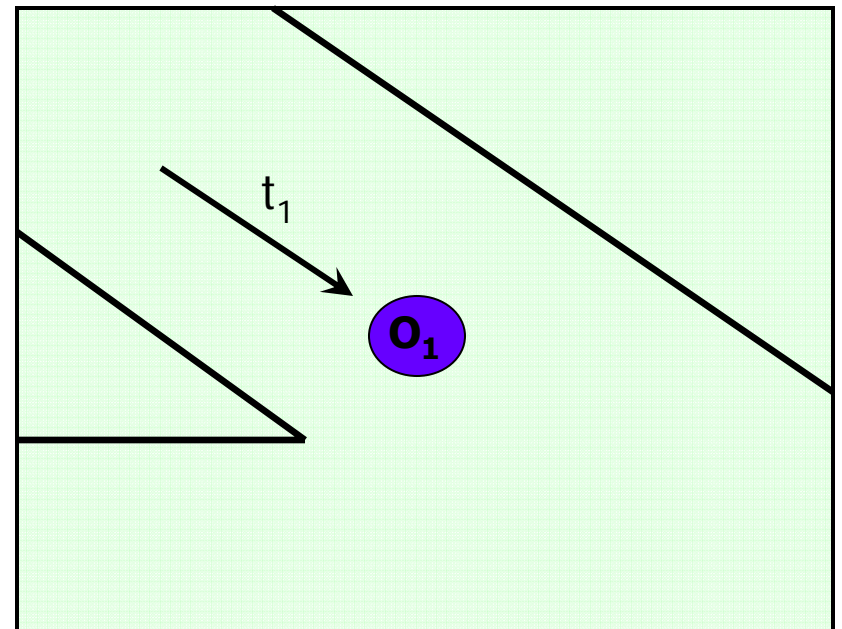
$$p(\text{Assign } t_1 \rightarrow O_1 | d_1)$$

- By Bayes formula:

$$p(\text{Assign } t_1 \rightarrow O_1 | d_1) =$$

$$\frac{p(d_1 | \text{Assign } t_1 \rightarrow O_1) \times p(\text{Assign } t_1 \rightarrow O_1)}{p(d_1)}$$

- Calculation and row normalization:



	Blob O_1	Blob O_2	"lost"
t_1	0.90	0.05	0.05
t_2	0.33	0.33	0.33
t_3	0.33	0.33	0.33

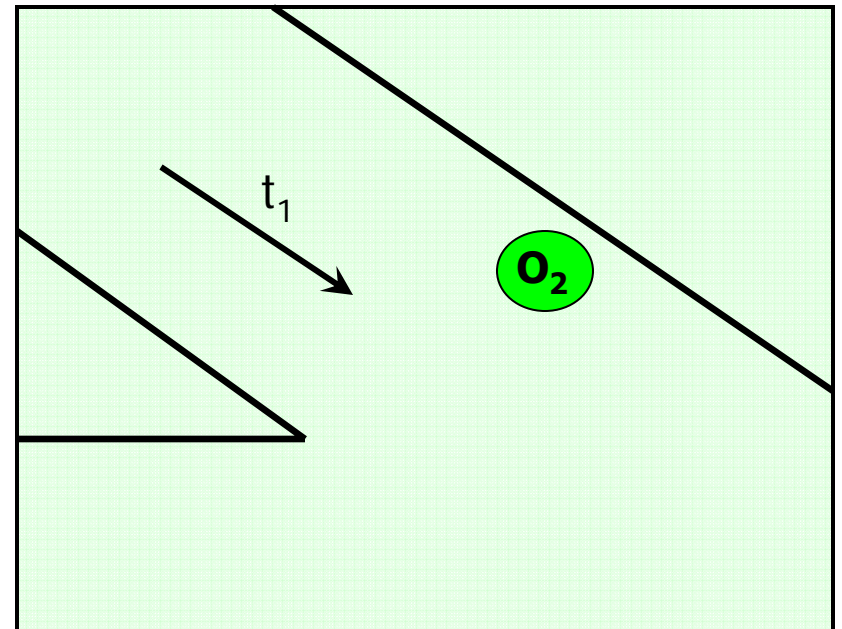


Illustration (6)

- After the first element update, we move to second element

$$p(\text{Assign } t_1 \rightarrow O_2)$$

- Similar calculation and update:
- Row 1 processing - complete



	Blob O_1	Blob O_2	"lost"
t_1	0.75	0.22	0.08
t_2	0.33	0.33	0.33
t_3	0.33	0.33	0.33



Illustration (7)

- Similarly, processing track t_2 and row 2:

	Blob O_1	Blob O_2	"lost"
t_1	0.75	0.22	0.08
t_2	0.25	0.60	0.15
t_3	0.33	0.33	0.33

- Processing track t_3 and row 3:

	Blob O_1	Blob O_2	"lost"
t_1	0.75	0.22	0.08
t_2	0.25	0.60	0.15
t_3	0.20	0.30	0.50

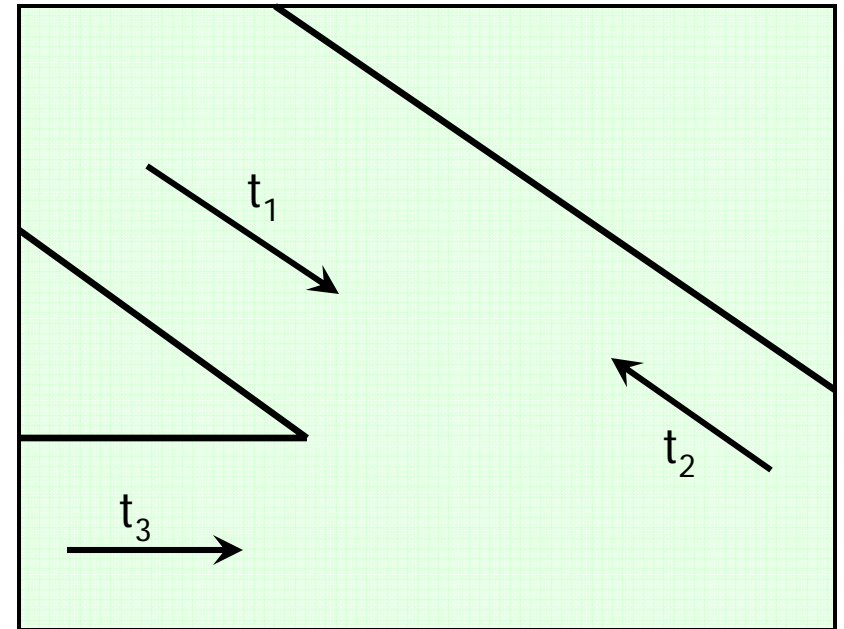
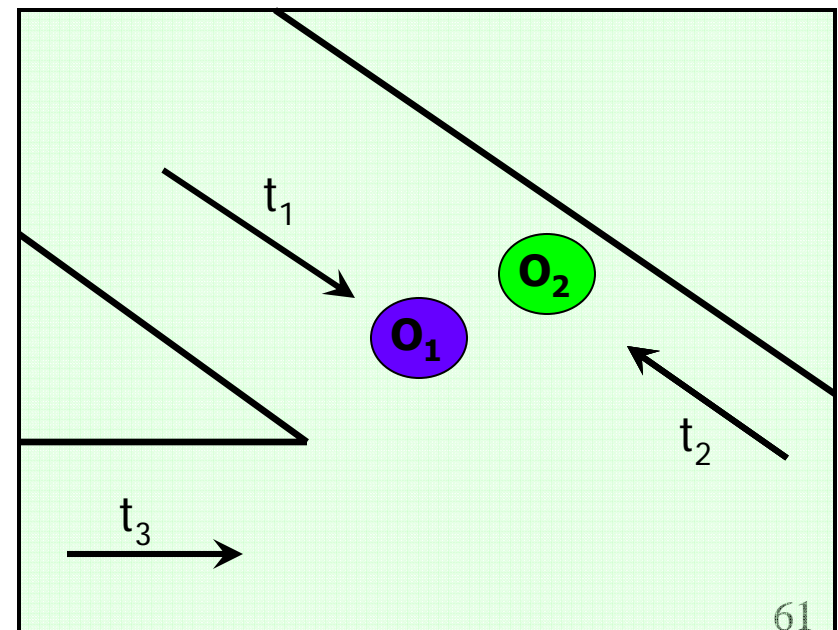


Illustration (8)

■ Based on the Belief matrix, the following assignments may be made

- $t_1 \rightarrow O_1$
- $t_2 \rightarrow O_2$
- $t_3 \rightarrow \text{"lost"}$

	Blob O_1	Blob O_2	"lost"
t_1	0.75	0.22	0.08
t_2	0.25	0.60	0.15
t_3	0.20	0.30	0.50

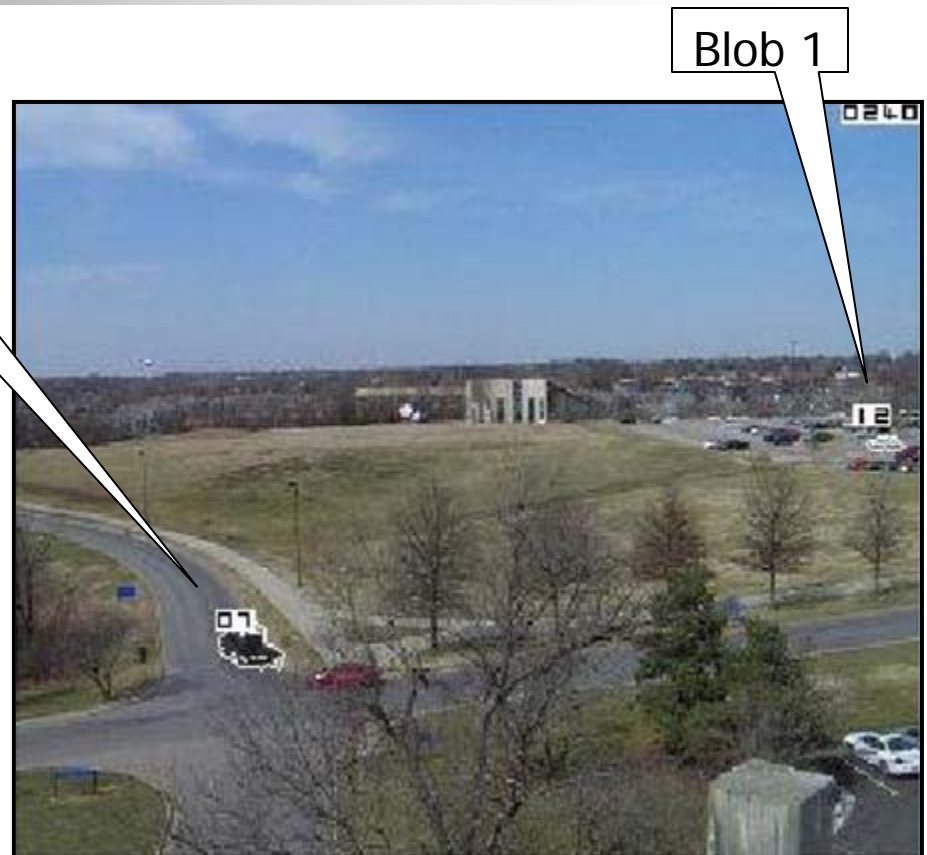


Results

Real example - frame 0240

	Blob 1	Blob 2	"lost"
track 03	0.00	0.10	0.90
track 07	0.00	1.00	0.00
track 11	0.00	0.39	0.61
track 12	1.00	0.00	0.00

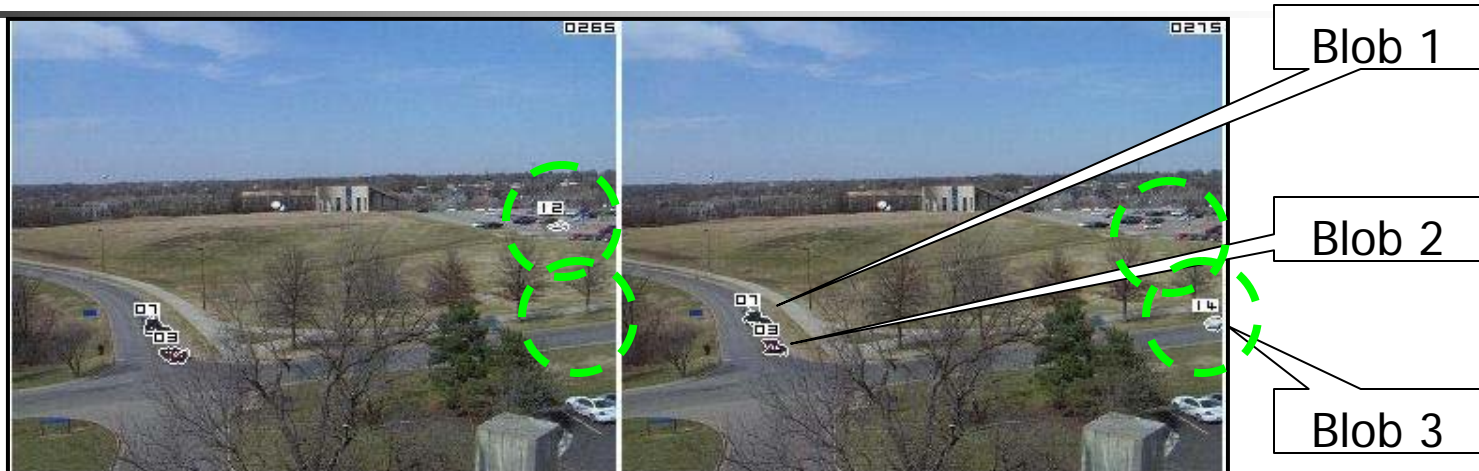
Belief matrix



Resulting track assignments



Results – comparison with Euclidean distance matrix



	Blob 1	Blob 2	Blob 3	"lost"
Track 3	0.35	0.55	0.00	0.10
Track 7	0.38	0.52	0.00	0.10
Track 12	0.00	0.00	0.49	0.51

(c) Bayes belief matrix - frame 0275

	Blob 1	Blob 2	Blob 3
Track 3	0.17	0.03	1.45
Track 7	0.03	0.15	1.44
Track 12	1.29	1.36	0.29

(d) Euclidean distance matrix - frame 0275

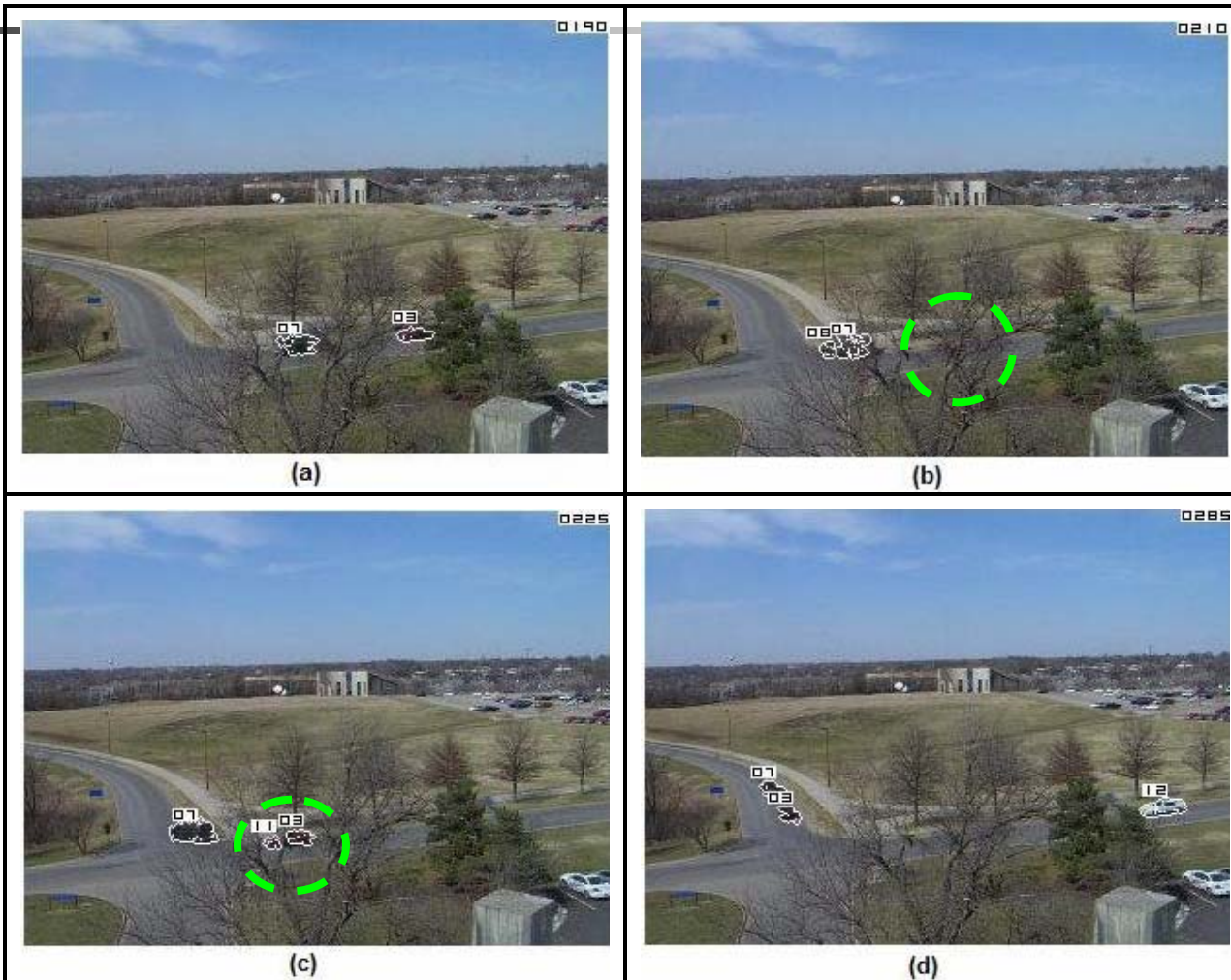


Results – comparison with Euclidean distance matrix

- “lost” probability can be useful
- Track 12 (lost in frame 0275) would be erroneously assigned to blob 3 if Euclidean distance matrix used
- Bayesian Belief matrix can be a useful alternative to distance-based match matrix
- More useful in inference tasks



Results - tracking



Results sequence 1



Results sequence 2



Results sequence 3





Summary

- Designed suitable data structures, classes, and methods for BG segmentation, blobs, and tracks
- Works on outdoor sequences
- Average Background model
- Secondary Background model
- Blob merging
- Velocity factor
- The Vicinity Factor
- Bayesian algorithm





Conclusions

- Automatic tracking can lead to interesting research and applications
- We have established a baseline system for future research
- Successfully met research objectives
 - Setting up of a base system
 - Developing new algorithms
- System is capable of tracking based on
 - Euclidean distance-based match matrix
 - Bayesian Belief matrix for probabilistic assignment





Future work (1)

- Improvements in current system
 - Object segmentation
 - Gaussian BG model
 - Template matching and search methods
 - Object tracking
 - Use of blob size as a feature for matching
 - Secondary analysis of tracks to establish longer tracks
 - Template matching and search based tracking
 - Useful in non-static camera applications





Future work (2)

- Improvements in current system (contd..)
 - Vicinity Factor
 - To identify anomalous behavior based on object motion
 - To estimate object size in different parts of grid
 - To detect “active” regions of the scene
 - In multi-class problems, use of separate VF matrix for each class of objects
 - Bayesian tracking algorithm
 - Automatically learn PDF's for the assignment network
 - Use size as an observation, along with color and position
 - Other color spaces like HSV



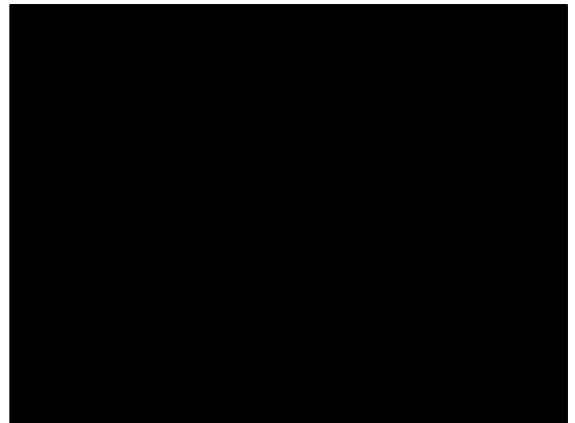
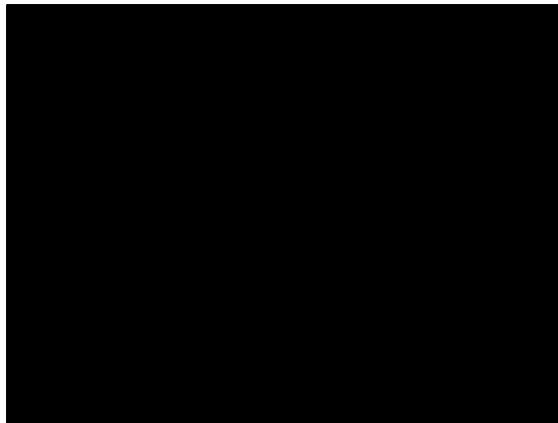
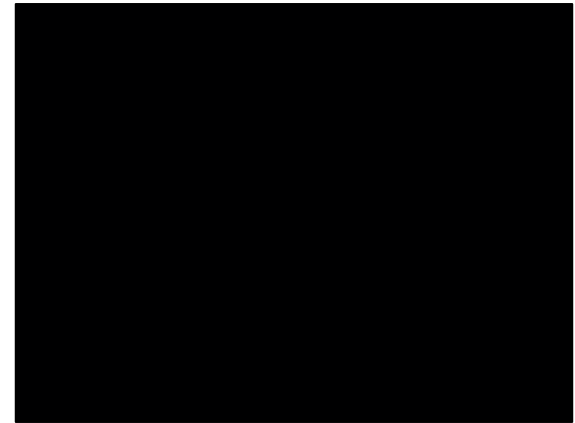
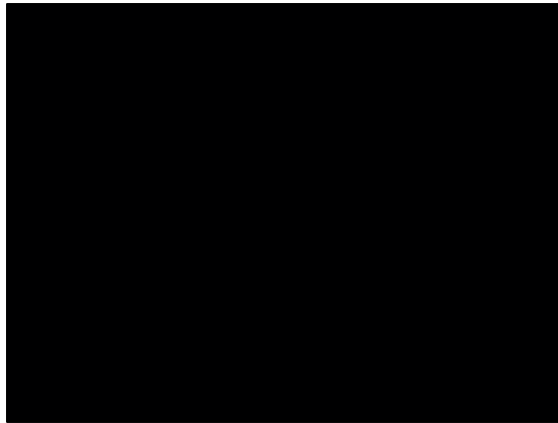


Future work (3)

- Extensions of current system
 - Learn patterns of activity from tracks
 - Gait analysis for human identification
 - Gesture recognition
 - Behavior analysis based on tracks
 - Detection of events
 - ...

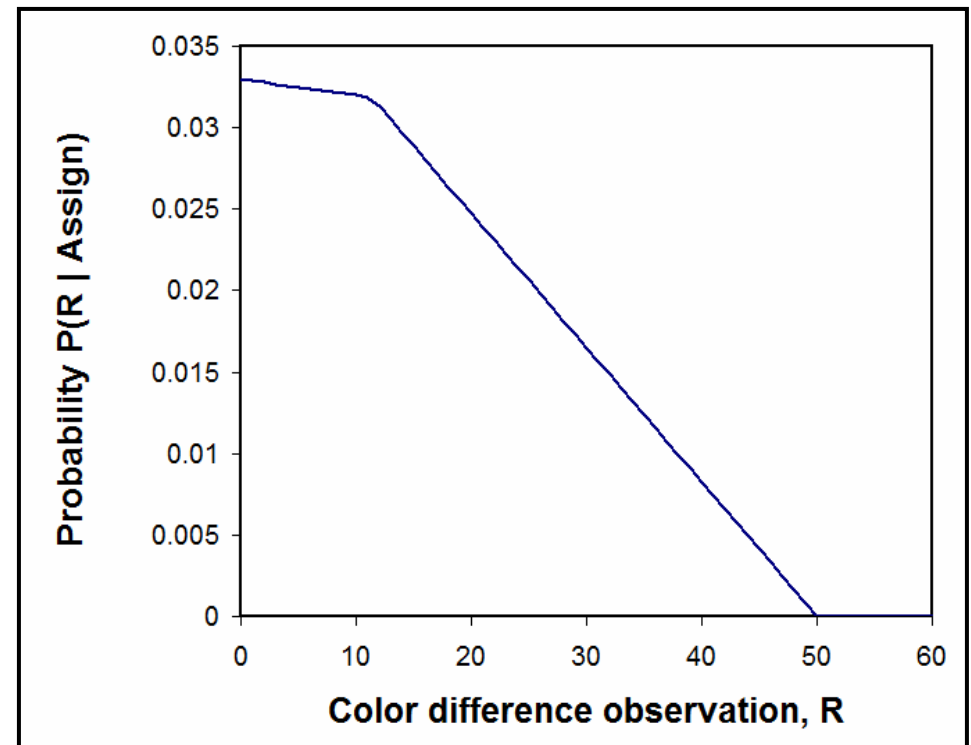


Questions ?



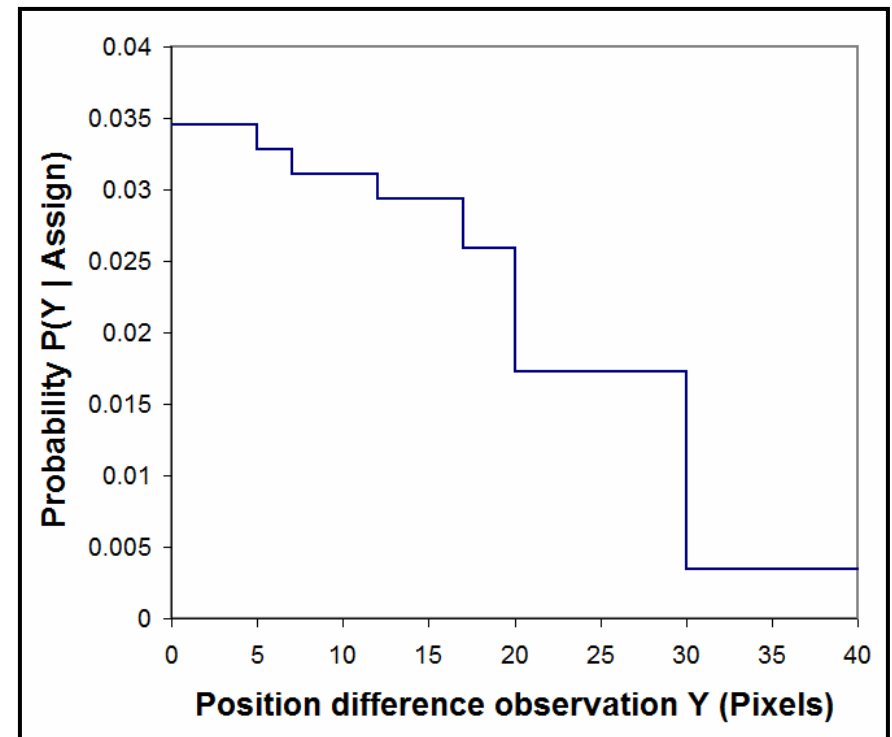
Probabilities – Color PDF

- 200 frames observed
- The color difference between the track and corresponding blob is observed
- PDF for Red color shown here
- Same PDF used for G and B



Probabilities – Position PDF

- 200 frames observed
- The position difference between the track and corresponding blob is noted
- PDF for Y position shown here
- Same PDF used for X position



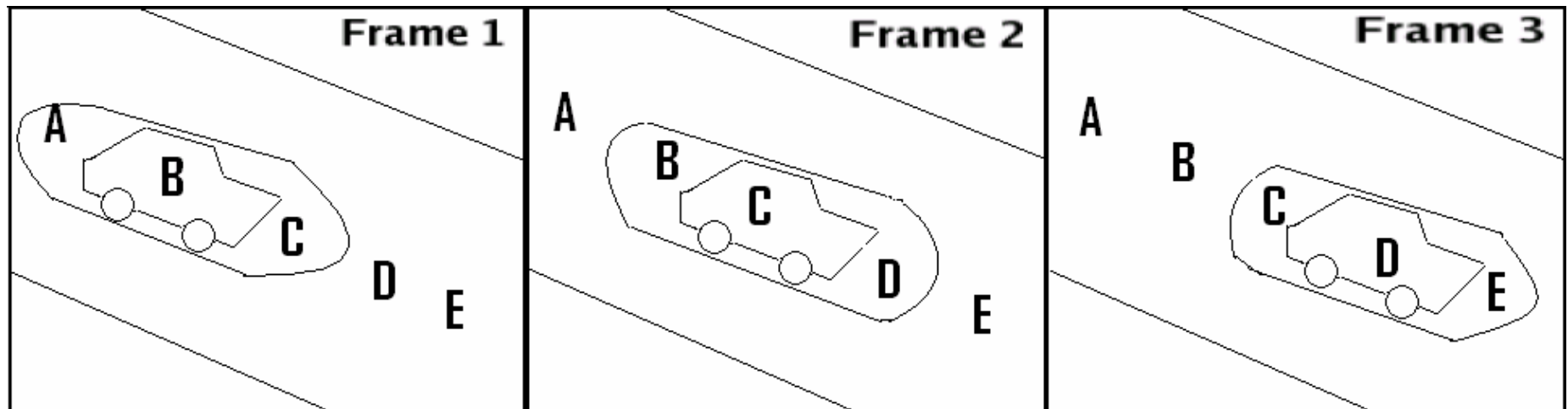


Background (2)

- Another common approach
 - Develop a model of the object being tracked
 - Use searching and statistical methods to locate the object in each frame
 - Computationally expensive method
 - Kalman filtering, Joint Probabilistic Data Association Filters, Condensation
- Commonly discussed tracking systems
 - VSAM – MIT
 - W4 – Univ of Maryland
 - Bramble – Compaq Research



SBG model illustration



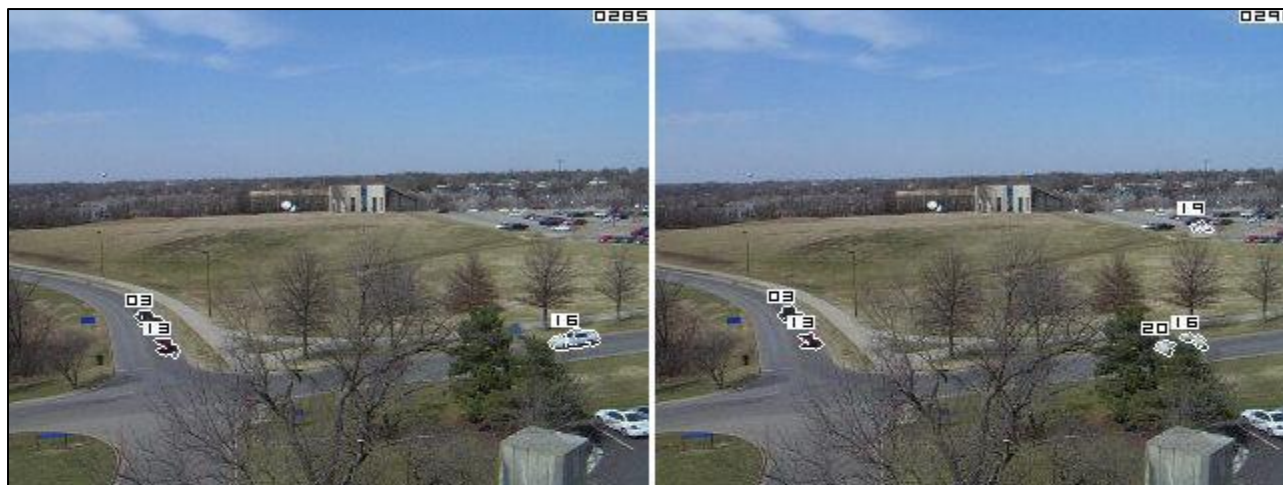


Track assignment procedure

- Based on the Match matrix
- Older tracks given preference
- If a match is not found for a track, it is not deleted immediately, but kept alive
 - Declared as “lost”
- In subsequent frames, if the track reemerges, it is reassigned



Basic results ("new track" case)



(a) Previous frame - 0285

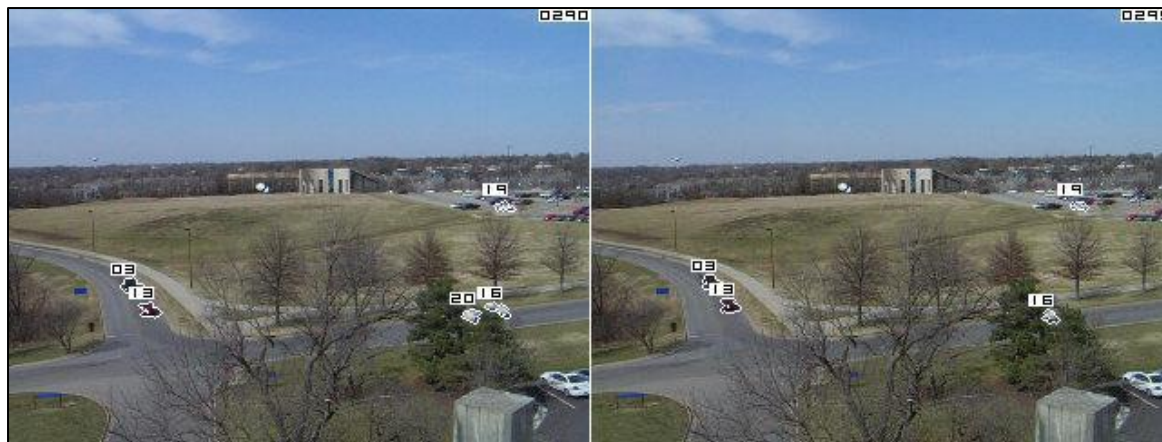
(b) Current frame - 0290

	Blob 1	Blob 2	Blob 3	Blob 4	Blob 5
Track 3	1.30	0.04	0.12	1.22	1.31
Track 13	1.28	0.12	0.03	1.18	1.26
Track 16	0.32	1.15	1.17	0.05	0.22

(c) Match matrix for current frame 0290



Basic results ("lost" case)



(a) Previous frame - 0290

(b) Current frame - 0295

	Blob 1	Blob 2	Blob 3	Blob 4
Track 3	1.28	0.01	0.13	1.05
Track 13	1.31	0.13	0.01	1.06
Track 16	0.31	1.16	1.16	0.11
Track 19	0.04	1.27	1.30	0.40
Track 20	0.32	1.29	1.29	0.23

(c) Match matrix for current frame 0295





How VF is useful?

- Automatically learns the distance variation that is caused due to OD change
- Gives basis for using variable thresholds in different parts of the image
- Applications
 - Blob merging – the thresholds used to decide whether or not to merge two blobs can be varied based on VF
 - Object size estimation – using VF Y and VF X values
 - Detecting active regions in scene
 - VF changes from its initialized value only in parts of the scene where real trackable motion was observed
 - Robust to noise blobs



Illustration (5)

- Row needs to be normalized so that sum of elements is 1

	Blob O_1	Blob O_2	"lost"
t_1	0.60	0.33	0.33
t_2	0.33	0.33	0.33
t_3	0.33	0.33	0.33

After row
normalization



	Blob O_1	Blob O_2	"lost"
t_1	0.60	0.20	0.20
t_2	0.33	0.33	0.33
t_3	0.33	0.33	0.33

