Moving Object Tracking

Princeton University
COS 429 Lecture

Dec. 6, 2007

Harpreet S. Sawhney
hsawhney@sarnoff.com

vision technologies LAB

SARNOFF Corporation

Recapitulation : Last Lecture

- Moving object detection as robust regression with outlier detection

- Simultaneous multiple surface/moving object estimation

- Expectation-Maximization (EM) algorithm as a formal mechanism for multiple model estimation & its application to multiple motion detection and estimation

The Tracking Problem

- Maintain identity across time of designated or automatically detected objects

The Tracking Problem : Issues

- **Object Detection** : Designated or automatically detected

- **Object State Instantiation**: Object representation
  - Position, Velocity, Shape, Color, Appearance, Template…

- **State Prediction**:
  - Having seen $\{y_0, y_1, \ldots, y_{i-1}\}$ what state do these measurements predict for the next time instant *i*?
  - Need a representation for $P(X_i \mid Y_0 = y_0, \ldots, Y_{i-1} = y_{i-1})$

- **Data Association**:
  - Which of the measurements at the *i*-th instant correspond to the predicted state at that instant ?
  - Use $P(X_i \mid Y_0 = y_0, \ldots, Y_{i-1} = y_{i-1})$ to establish the correspondence

- **State Update**:
  - With the corresponding measurement y_i established for instant *i*, compute an estimate of the optimal new state through $P(X_i \mid Y_0 = y_0, \ldots, Y_i = y_i)$

Object Detection

Designated Object

- User specifies a template
- The system converts that template into an appropriate representation to be tracked

Object Detection

Fixed Cameras

- Model the background using a reference image or a reference distribution

- Detect objects as changes with respect to the reference

Object Detection

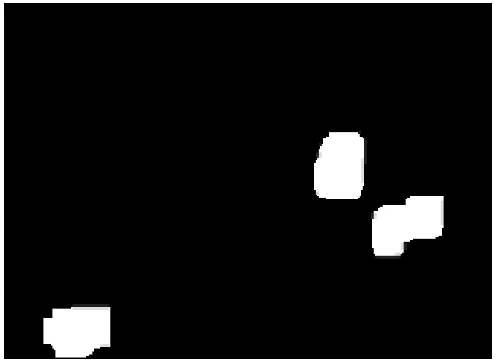
Moving Cameras

- Align consecutive frames using the now well-known techniques studied in this class

- Use frame differencing between aligned frames to detect changes designated as new objects

Simple Tracker : Blob tracker

- Change-based tracker:
  - Approach
    - Align video images
    - Detect regions of change
    - Track change blobs

  - Problem with this approach is that it uses <u>no appearance</u> information
    - difficult to deal with stalled or close-by objects

Moving Blobs

Simple Tracker - Correlation Based

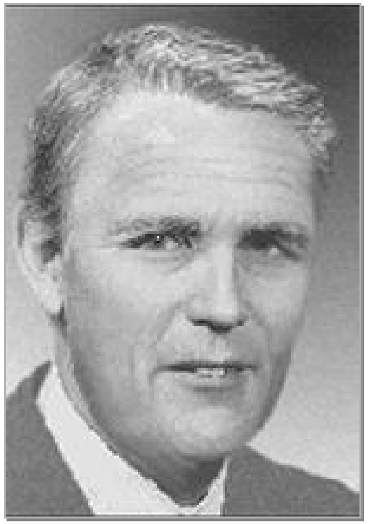
- Correlation-based tracker:
  - Approach
    - Initialize the templates and the supports of foreground objects
    - Estimate motion by correlation
  - The problem with this approach is that it does not simultaneously compute the segmentation and appearance
    - No accurate segmentation or region of support $\Rightarrow$ may drift over time.
    - Get confused by cluttered backgrounds

Problems with the simple trackers

- They lack the two key ingredients for optimal tracking:
  - State prediction
  - Optimal state updation
- Since measurements are never perfect --- each has some uncertainty associated with it --- optimal state prediction and updation need to take the uncertainties into account
- Furthermore, the object representation needs to be richer
  - Not just a change blob, or fixed template
  - Optimal method for updating the state

Kalman Filtering

- Assume that results of experiment (i.e., optical flow) are noisy measurements of system state
- Model of how system evolves
- Prediction / correction framework
- Optimal combination of system model and observations



Rudolf Emil Kalman

Acknowledgment: much of the following material is based on the SIGGRAPH 2001 course by Greg Welch and Gary Bishop (UNC)

Simple Example

- A point whose position remains constant : x
  - Say a temperature reading
- Noisy measurement of that single point z_1
- Variance σ_1^2 (uncertainty σ_1)
- Best estimate of true position $\hat{x}_1 = z_1$
- Uncertainty in best estimate $\hat{\sigma}_1^2 = \sigma_1^2$

Simple Example

- Second measurement z_2, variance σ_2^2

- Best estimate of true position

$$\hat{x}_2 = \frac{\frac{1}{\sigma_1^2} z_1 + \frac{1}{\sigma_2^2} z_2}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}$$

$$= \hat{x}_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\left(z_2 - \hat{x}_1\right)$$

- Uncertainty in best estimate

$$\hat{\sigma}_2^2 = \frac{1}{\frac{1}{\hat{\sigma}_1^2} + \frac{1}{\sigma_2^2}}$$

Online Weighted Average

- Combine successive measurements into constantly-improving estimate

- Uncertainty decreases over time

- Only need to keep current measurement,
  last estimate of state and uncertainty

We have essentially computed the Least Squares OR Minimum Variance OR Maximum Likelihood estimate of X given a number of noisy measurements Z through an incremental method

Terminology

- In this example, position is *state*
  - in general, any vector

- State evolves over time according to a *dynamic model* or *process model*
  - (in this example, “nothing changes”)

- Measurements are related to the state according to a *measurement model*
  - (possibly incomplete, possibly noisy)

- Best estimate of state $\hat{x}$ with covariance *P*

Tracking Framework

- Very general model:
  - We assume there are moving objects, which have an underlying state X
  - There are measurements Z, some of which are functions of this state
  - There is a clock
    - at each tick, the state changes
    - at each tick, we get a new observation

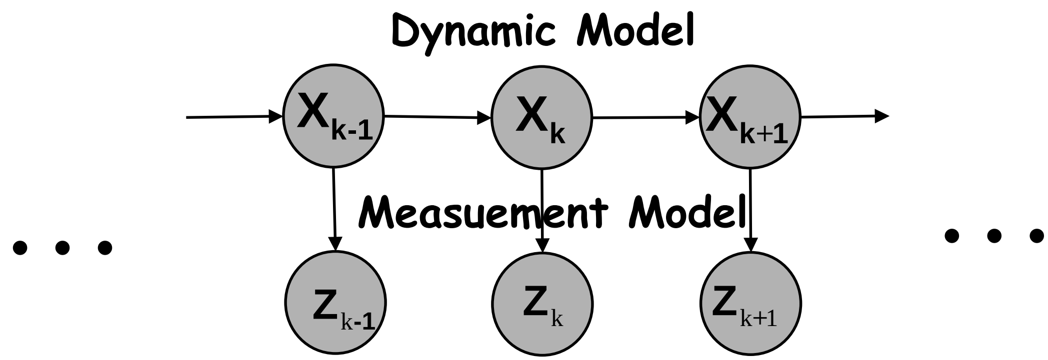
- Examples
  - object is ball, state is 3D position+velocity, measurements are stereo pairs
  - object is person, state is body configuration, measurements are frames, clock is in camera (30 fps)

Bayesian Graphical Model

State Variables:

Those that tell us about objects & their states

But they are hidden, cannot be directly observed



Measurements:

Can be directly observed

Are noisy, uncertain

Bayesian Formulation

$$p(x_k \mid z_k) = \kappa p(z_k \mid x_k) \int p(x_k \mid x_{k-1}) p(x_{k-1} \mid z_{k-1}) dx_{k-1}$$

$p(x_k \mid z_k)$ Posterior probability after latest measurement

$p(z_k \mid x_k)$ Likelihood of the current measurement

$p(x_k \mid x_{k-1})$ Temporal prior from the dynamic model

$p(x_{k-1} \mid z_{k-1})$ Posterior probability after previous measurement

κ Normalizing constant

The Kalman Filter

- Key ideas:
  - Linear models interact uniquely well with Gaussian noise
    - make the prior Gaussian,
    - everything else Gaussian and the calculations are easy
  - Gaussians are really easy to represent
    - once you know the mean and covariance, you're done

Linear Models

- For "standard" Kalman filtering, everything must be linear
- System / Dynamical model: $x_k = \Phi_{k-1} x_{k-1} + \xi_{k-1}$
- The matrix Φ_k is *state transition matrix*
- The vector ξ_k represents *additive noise*, assumed to have covariance *Q : N(0;Q)*

$$\mathbf{x_k \sim N(\Phi_k x_{k-1};Q_k)}$$

Linear Models

- Measurement model / Likelihood model:

$$\mathbf{z_k} = \mathbf{H_k x_k} + \boldsymbol{\mu}_\mathbf{k}$$

$$\mathbf{z_k} \sim \mathbf{N}(\mathbf{H_k x_k}; \mathbf{R_k})$$

- Matrix *H* is *measurement matrix*

- The vector μ is *measurement noise*, assumed to have covariance *R :* $\mathbf{N(0;\mu)}$

Position-Velocity Model

- Points moving with constant velocity
- We wish to estimate their PV state at every time instant

$$\mathbf{x}_k = \begin{bmatrix} x \\ {}^{dx}\!/_{dt} \end{bmatrix}$$

Position-Velocity State

$$\Phi_k = \begin{bmatrix} 1 & \Delta t_k \\ 0 & 1 \end{bmatrix}$$

Constant Velocity Dynamic Model Matrix

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Only position is directly observable

Prediction/Correction

- Predict new state

$$x'_k = \Phi_{k-1}\hat{x}_{k-1}$$

$$P'_k = \Phi_{k-1}P_{k-1}\Phi^{\mathrm{T}}_{k-1} + Q_{k-1}$$

- Correct to take new measurements into account

$$\hat{x}_k = x'_k + K_k\left(z_k - H_k x'_k\right)$$

$$P_k = \left(I - K_k H_k\right)P'_k$$

Kalman Gain

- Weighting of process model vs. measurements

$$K_k = P'_k H_k^{\mathrm{T}} \left(H_k P'_k H_k^{\mathrm{T}} + R_k \right)^{-1}$$

- Compare to what we saw earlier:

$$\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}$$

Optimal Linear Filter

Optimal Linear Estimate

Predicted state

Measurement

$$\hat{x}_k(+) = K'_k \hat{x}_k(-) + K_k z_k$$

Under Gaussian assumptions, linear estimate is the optimal

Estimation Error: $$\hat{x}_k(+) = x_k + \tilde{x}_k(+)$$

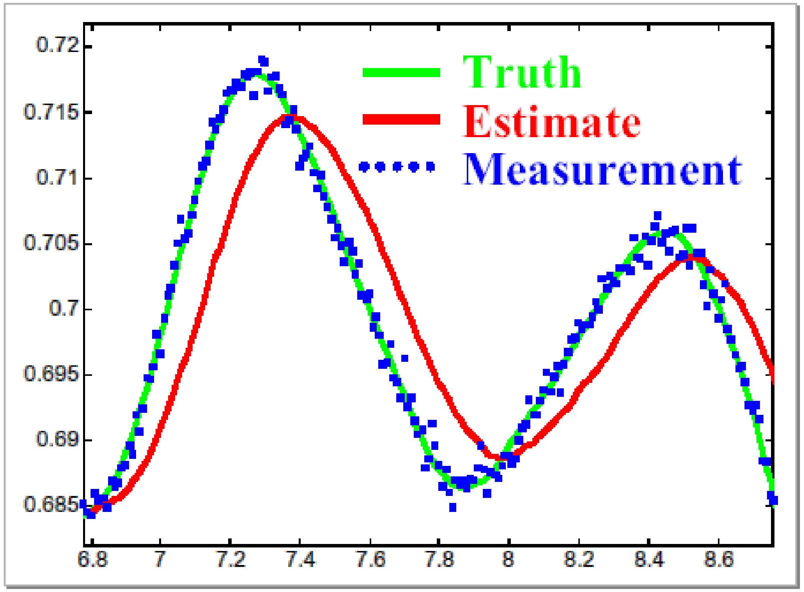
$$\tilde{x}_k(+) = [K'_k + K_k H_k - I]x_k + K'_k \tilde{x}_k(-) + K_k v_k$$

For an unbiased estimate: $E[\tilde{x}_k(+)] = 0$ $\quad\quad \therefore K'_k = I - K_k H_k$

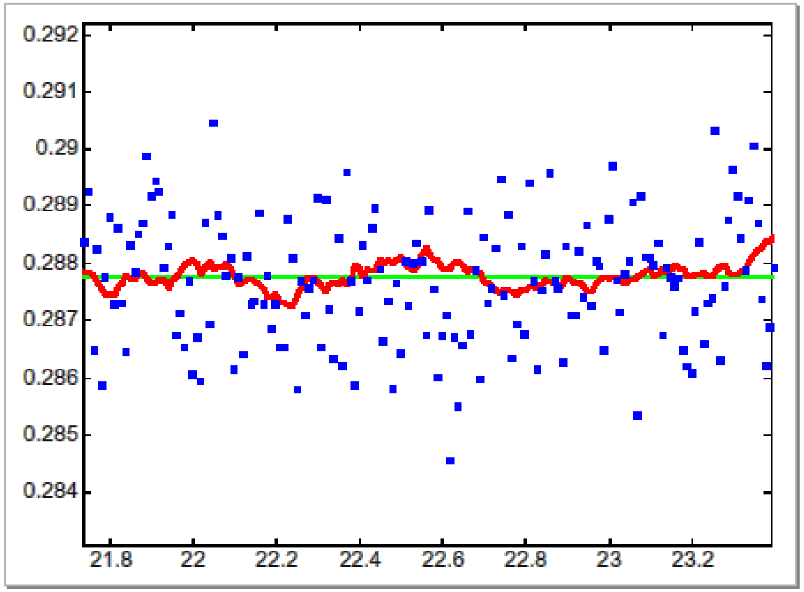
$$\therefore \hat{x}_k(+) = \hat{x}_k(-) + K_k[z_k - H_k \hat{x}_k(-)]$$

K_k Is obtained by minimizing the variance of the state estimate

Results: Position-Only Model



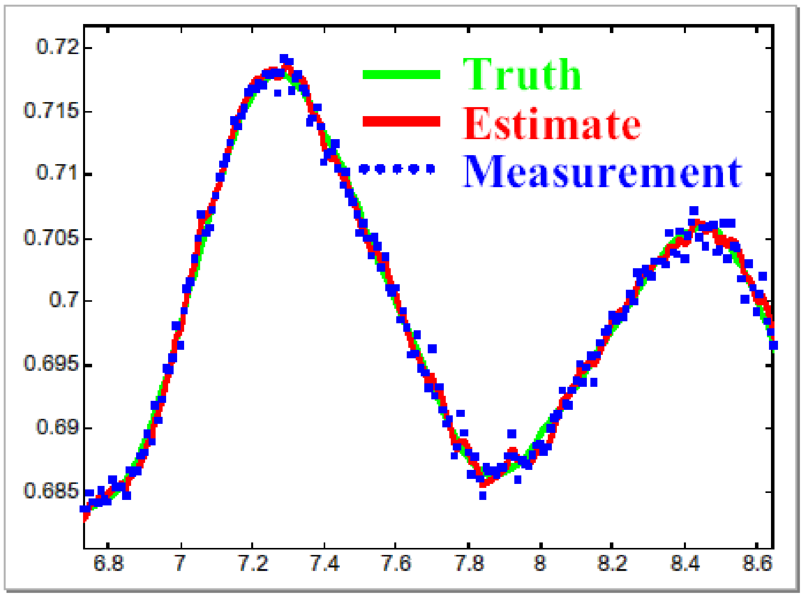
Moving



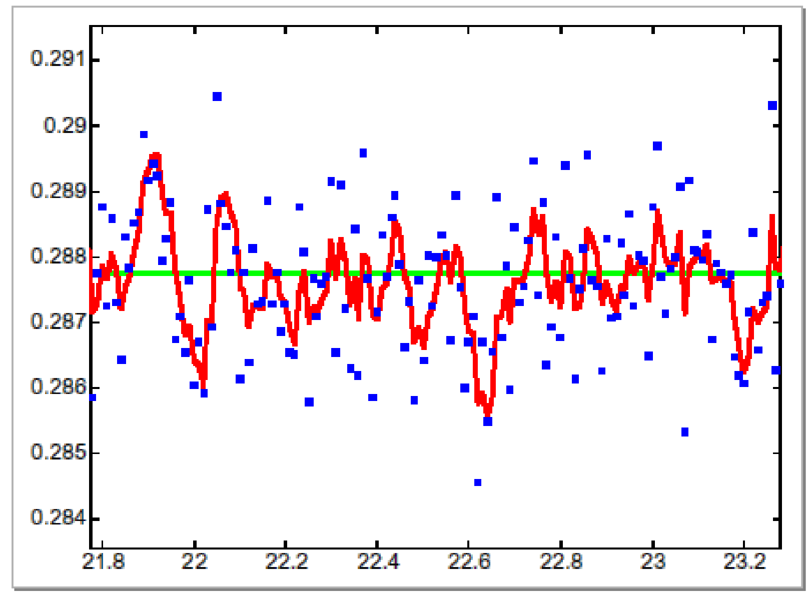
Still

[Welch & Bishop]

Results: Position-Velocity Model



Moving



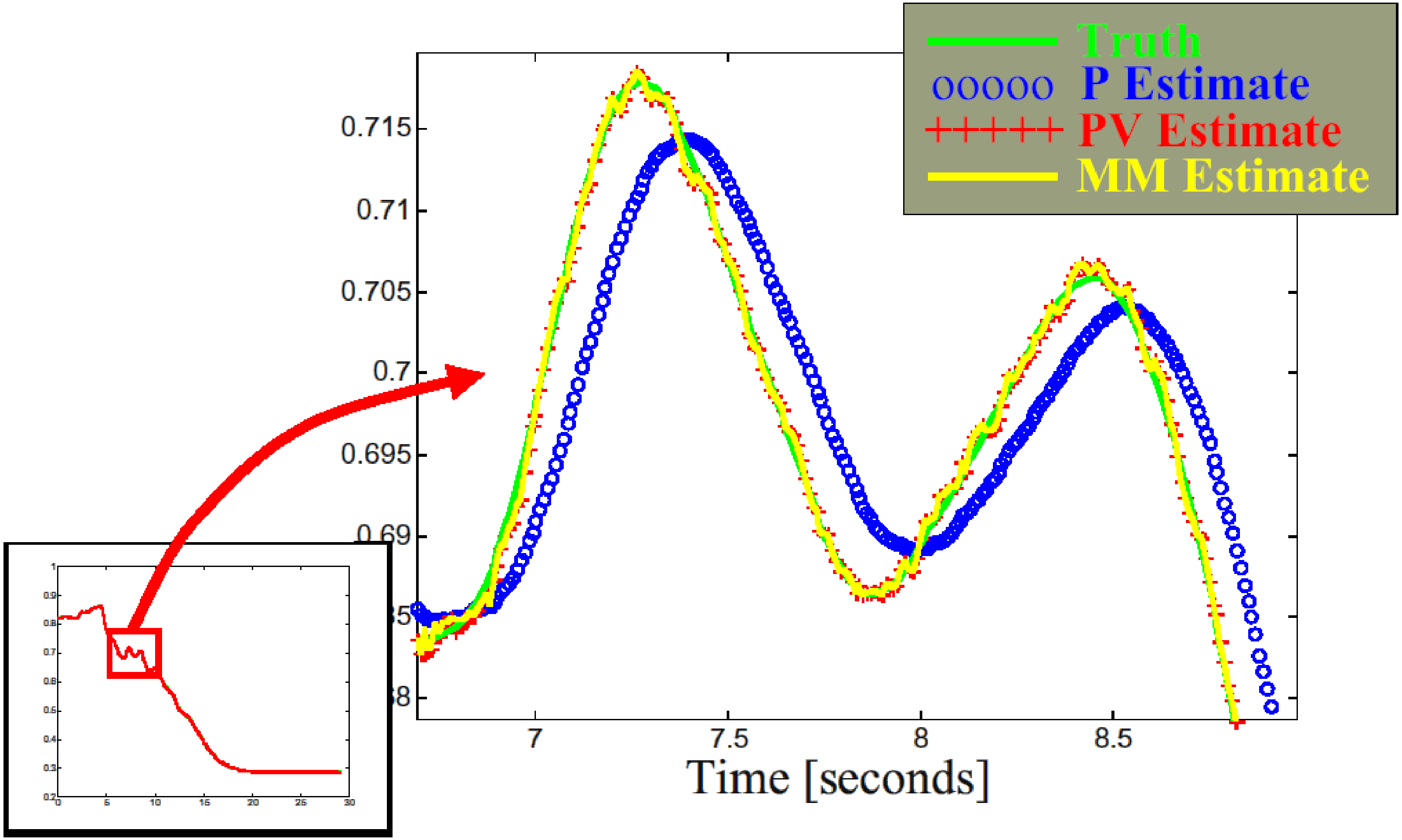
Still

[Welch & Bishop]

Extension: Multiple Models

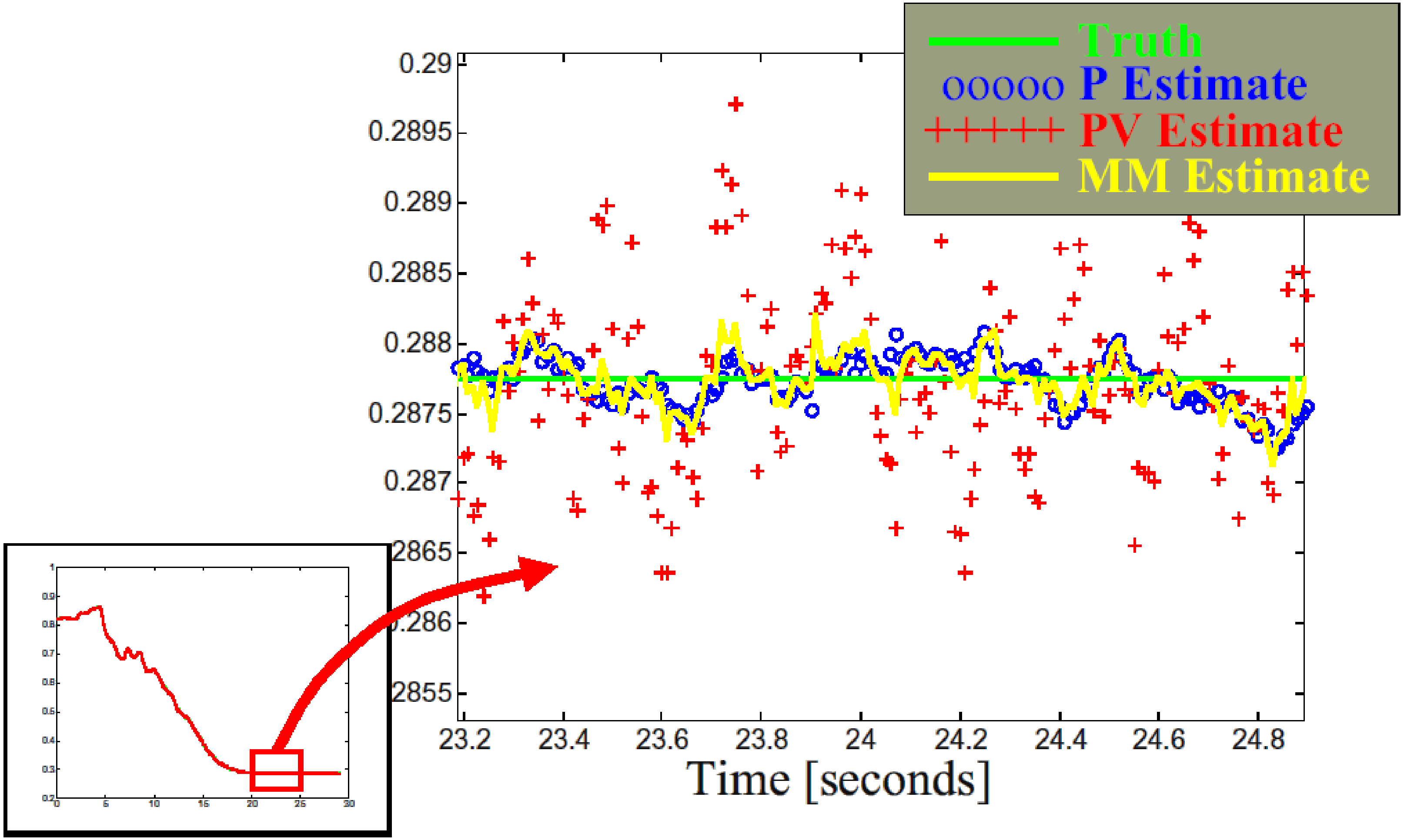
- Simultaneously run many KFs with different system models
- Estimate probability each KF is correct
- Final estimate: weighted average

Results: Multiple Models



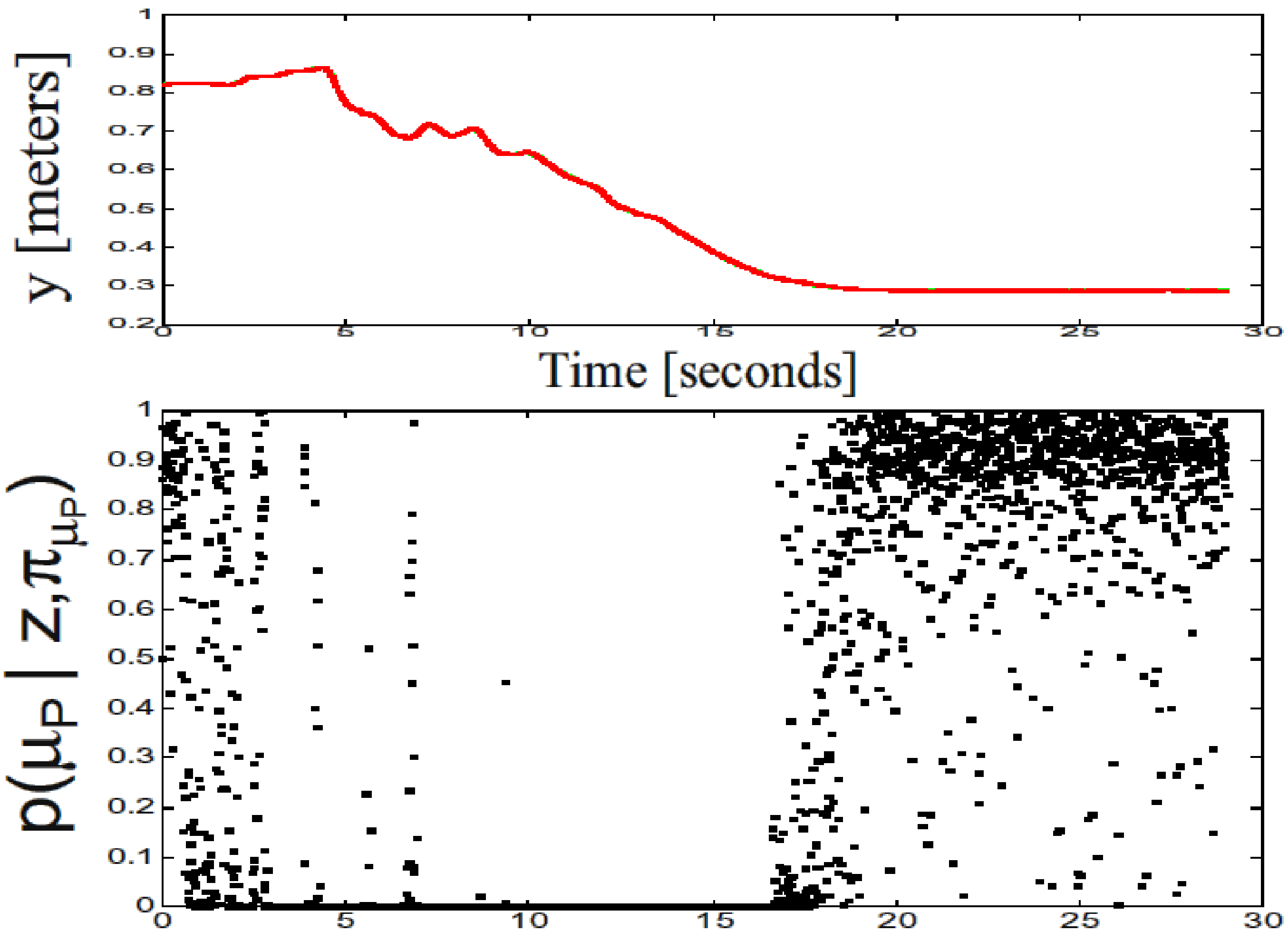
[Welch & Bishop]

Results: Multiple Models



[Welch & Bishop]

Results: Multiple Models

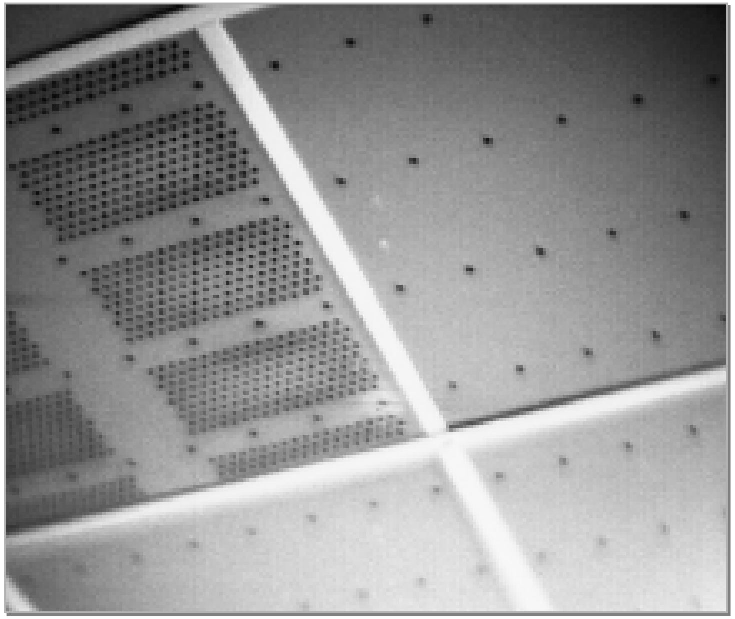


[Welch & Bishop]

Extension: SCAAT

- *H* be different at different times
  - Different sensors, types of measurements
  - Sometimes measure only part of state
- Single Constraint At A Time (SCAAT)
  - Incorporate results from one sensor at once
  - Alternative: wait until you have measurements from enough sensors to know complete state (MCAAT)
  - MCAAT equations often more complex, but sometimes necessary for initialization

UNC HiBall



- 6 cameras, looking at LEDs on ceiling
- LEDs flash over time

[Welch & Bishop]

Extension: Nonlinearity (EKF)

- HiBall state model has nonlinear degrees of freedom (rotations)

- Extended Kalman Filter allows nonlinearities by:
  - Using general functions instead of matrices
  - Linearizing functions to project forward
  - Like 1st order Taylor series expansion
  - Only have to evaluate Jacobians (partial derivatives), not invert process/measurement functions

Other Extensions

- On-line noise estimation
- Using known system input (e.g. actuators)
- Using information from both past and future
- Non-Gaussian noise and particle filtering

Data Association

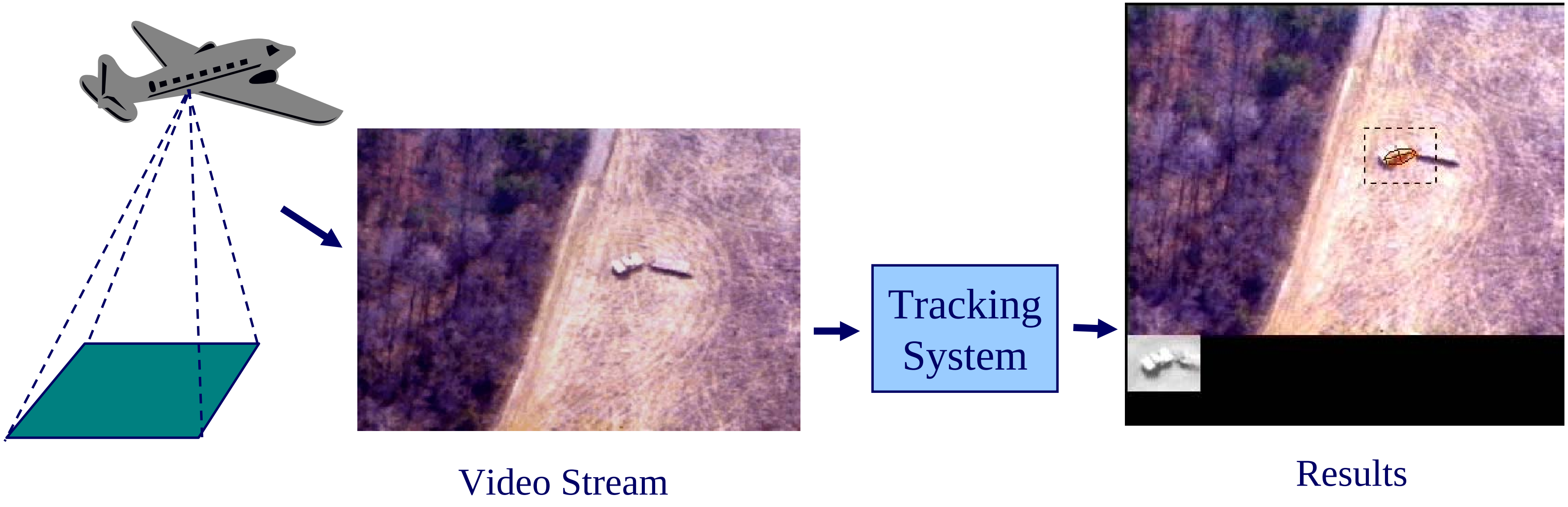
- Nearest Neighbors
  - choose the measurement with highest probability given predicted state
  - popular, but can lead to catastrophe
- Probabilistic Data Association
  - combine measurements, weighting by probability given predicted state
  - gate using predicted state

Video based Tracking : Complexities

- In addition to position and velocity, object state may include:
  - Appearance, shape, specific object models : people, vehicles, etc.
- Camera may move in addition to the object
  - Track background as well as the foreground
- Measurement model and the associated likelihood computation is more complex:
  - Compute the likelihood of the presence of a head-n-shoulders person model at a given location in the image
- Multiple objects need to be tracked simultaneously
  - Measurements need to be optimally associated with a set of models rather than a single model as in the previous examples

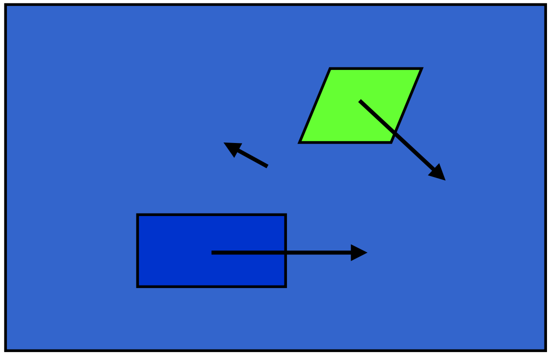
Application - Tracking vehicles in aerial videos

- The goals of a tracking system are to
  - detect new moving objects
  - maintain identity of objects, handle multiple objects and interactions between them. e.g. passing, stopped, etc.
  - provide information regarding the objects, e.g. shape, appearance and motion.



Tracking as a continuous motion segmentation problem

- Tracking problem $\Leftrightarrow$ continuous motion segmentation problem: estimation of a *complete* representation of foreground and background objects over time.
- Complete representation (Layer) includes:
  - motion of objects and background
  - shape of objects and support
  - appearance of objects
- Key: constraints



Layer based motion analysis method

- Simultaneously achieve motion and segmentation estimation (EM algorithm)
  - Estimate segmentation based on motion consistency
  - Estimate motion based on segmentation

Motion layer representations - models/constraints

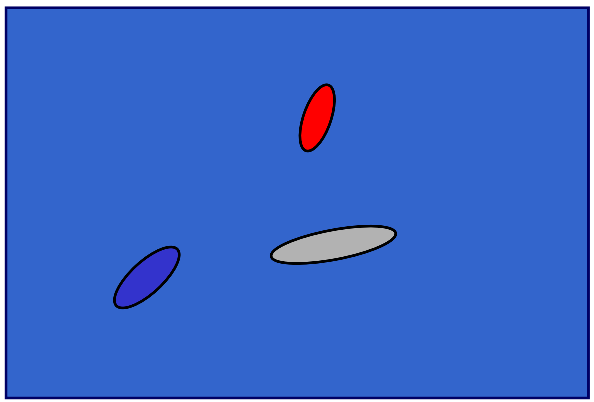
	Local constraints	Global constraints	Multi-frame consistency
Motion	Smooth dense flow: *Weiss 97*	2D affine: *Darrell91, Wang93, Hsu94, Sawhney96, Weiss 96, Vasconcelos97* 3D planar: *Torr99*	2D rotation and translation & constant velocity: *This paper*
Segmentation	MRF segmentation prior: *Weiss96, Vasconcelos97*	Background+Gaussian segmentation prior: *This paper - Section 2.1*	Constant segmentation prior: *This paper - Elliptical shape prior*
Appearance			Constant appearance: *This paper*

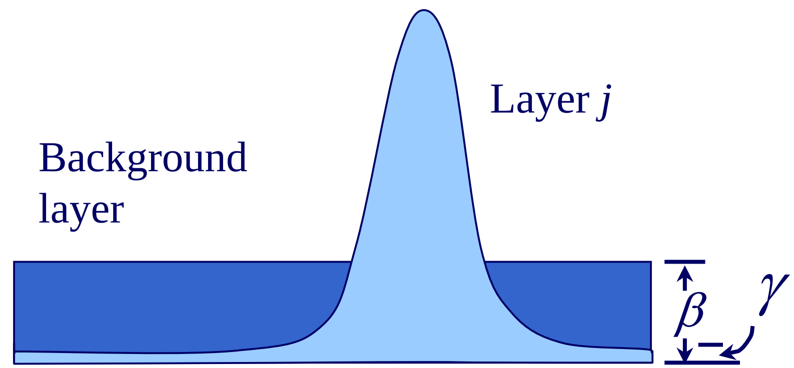
Dynamic Layer Representation

- Spatial and temporal constraints on the layer segmentation, motion, and appearance
- EM algorithm for maximum *a posteriori* estimation
- Layer ownership is constrained by a parametric shape distribution, instead of a local smoothness constraint. It prevents the layer evolving into arbitrary shapes, and enables tractable estimation over time.

Representation and constraints - segmentation and appearance

- Segmentation prior model $\Phi_t = \{l_t, s_t\}$
  - background + elliptical shapes
  - constant value over time



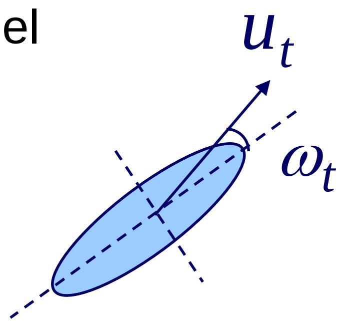


- Appearance model - A_t
  - constant value over time

Representation and constraints - motion

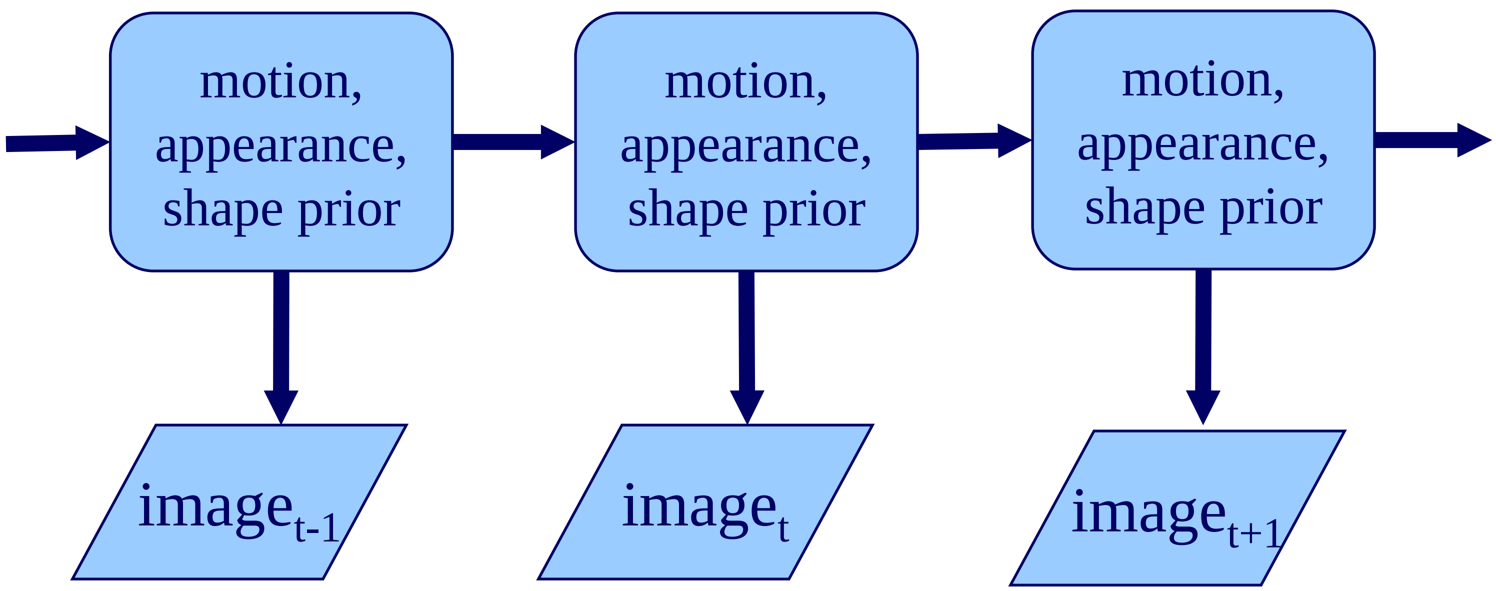
- Motion model
  - motion
    - foreground
      - translation + rotation
      - constant velocity model
    - background
      - planar surface

$$\Theta_t = (u_t, \omega_t)$$



MAP estimation

$$P(motion_t, appearance_t, shape\_prior_t \mid image_t, image_{t-1}, motion_{t-1}, appearance_{t-1}, shape\_prior_{t-1})$$



MAP estimation - formulation

- Notation
  - current image is I_t. Current state is $\Lambda_t = [\Theta_t, \Phi_t, \mathrm{A}_t]$
    .

- Estimation

$$
\begin{aligned}
&\max_{\Lambda_t} \arg P(\Lambda_t \mid \mathbf{I}_t, \Lambda_{t-1}, \mathbf{I}_{t-1}) \\
&= \max_{\Lambda_t} \arg \underbrace{P(\mathbf{I}_t \mid \Lambda_t, \Lambda_{t-1}, \mathbf{I}_{t-1})}_{\text{likelihood}} \underbrace{P(\Lambda_t \mid \Lambda_{t-1}, \mathbf{I}_{t-1})}_{\text{prior}}
\end{aligned}
$$

Optimization using EM algorithm

- The general Expectation Maximization algorithm
  - observation y and parameter θ
  - objective function:

$$\max_{\theta} \arg P(y \mid \theta) P(\theta)$$

  - equivalent to iteratively improving conditional expectation

$$Q(\theta \mid \theta') = E[\log P(x, y \mid \theta) \mid \theta', y] + \log P(\theta)$$

- For the dynamic layer tracker:

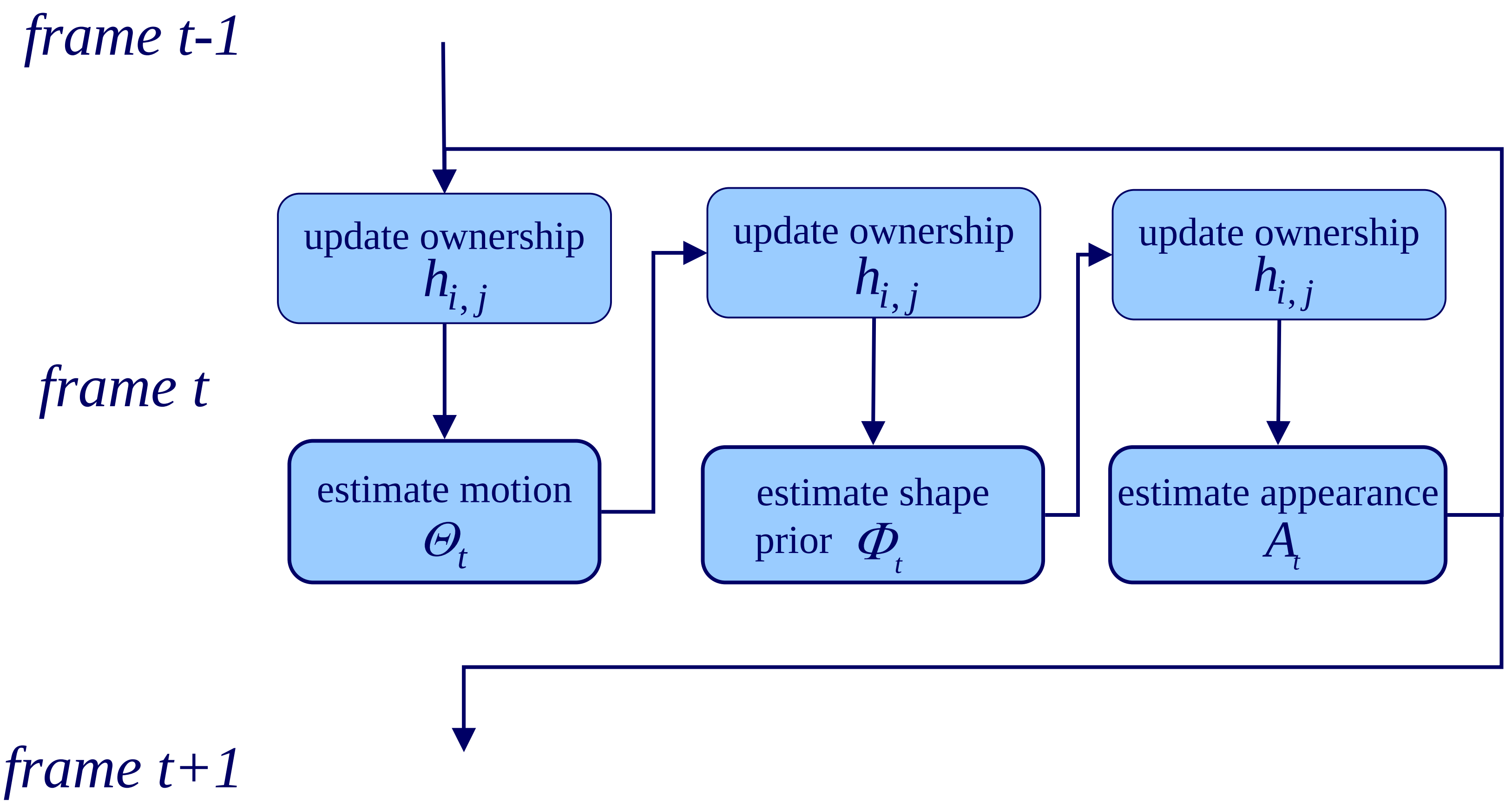
$$Q = E[\log P(I_t, z_t \mid \Lambda_t, \Lambda_{t-1}, I_{t-1}) \mid I_t, \Lambda_t', \Lambda_{t-1}, I_{t-1}] + \log P(\Lambda_t \mid \Lambda_{t-1})$$

- Optimize Q over Λ_t

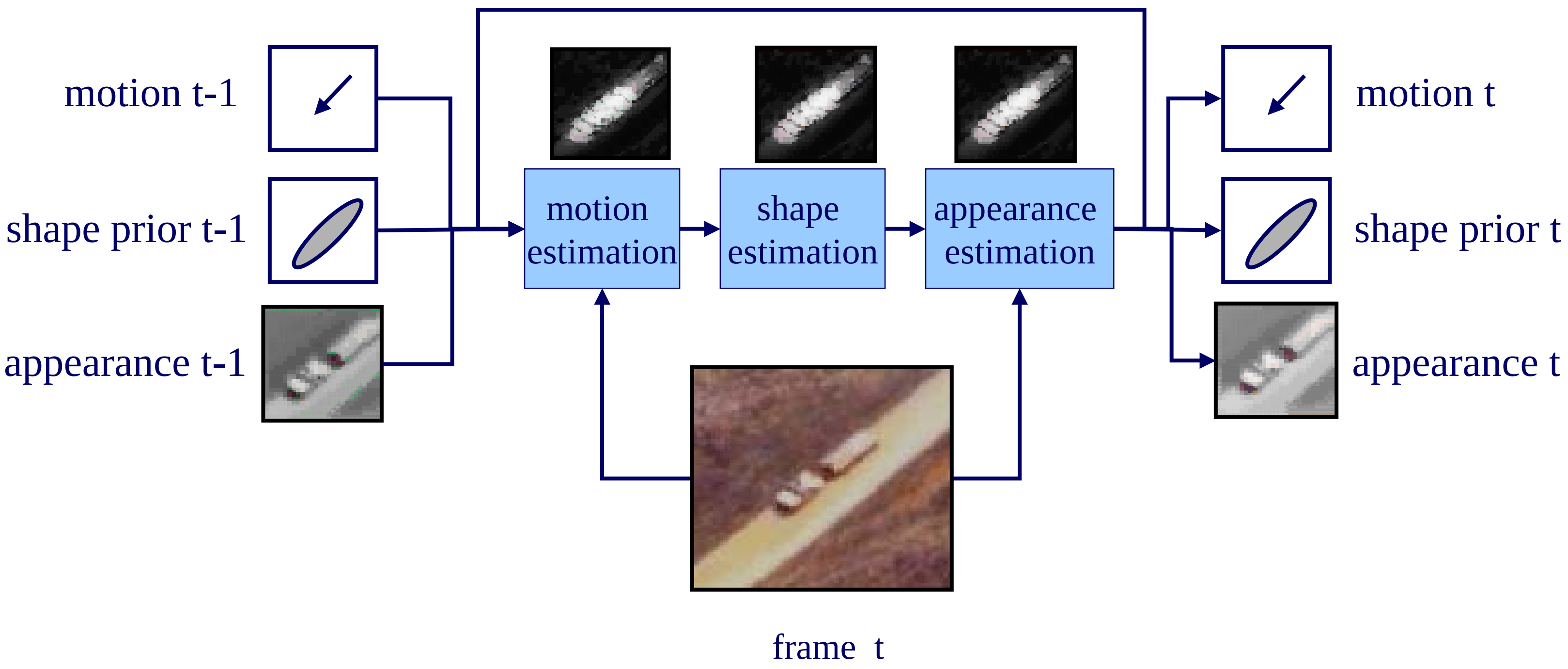
Optimization - 3 steps

- Optimization over motion, segmentation, and appearance correspond to the following three steps:
  - layer motion estimation based on current segmentation and appearance $\Rightarrow$ weighted correlation or direct method
  - layer segmentation estimation $\Rightarrow$ competition between motion layers
  - layer appearance estimation $\Rightarrow$ Kalman filtering of appearance

Optimization - flow chart



Optimization - illustration



Optimization - equations

- Motion estimation
  - weighted SSD
- Ownership estimation - gradient method

$$\frac{\partial f}{\partial s_{t,j}} = \sum_{i=0}^{n-1} \frac{h_{i,j}(D(x_i) - L_{t,j}(x_i))}{L_{t,j}(x_i) D(x_i)} (L_{t,j}(x_i) - \gamma) y_{i,j.y}^2 / s_{t,j}^3 - (s_{t,j} - s_{t-1,j}) / \sigma_{ls}^2$$

$$\frac{\partial f}{\partial l_{t,j}} = \sum_{i=0}^{n-1} \frac{h_{i,j}(D(x_i) - L_{t,j}(x_i))}{L_{t,j}(x_i) D(x_i)} (L_{t,j}(x_i) - \gamma) y_{i,j.x}^2 / l_{t,j}^3 - (l_{t,j} - l_{t-1,j}) / \sigma_{ls}^2$$

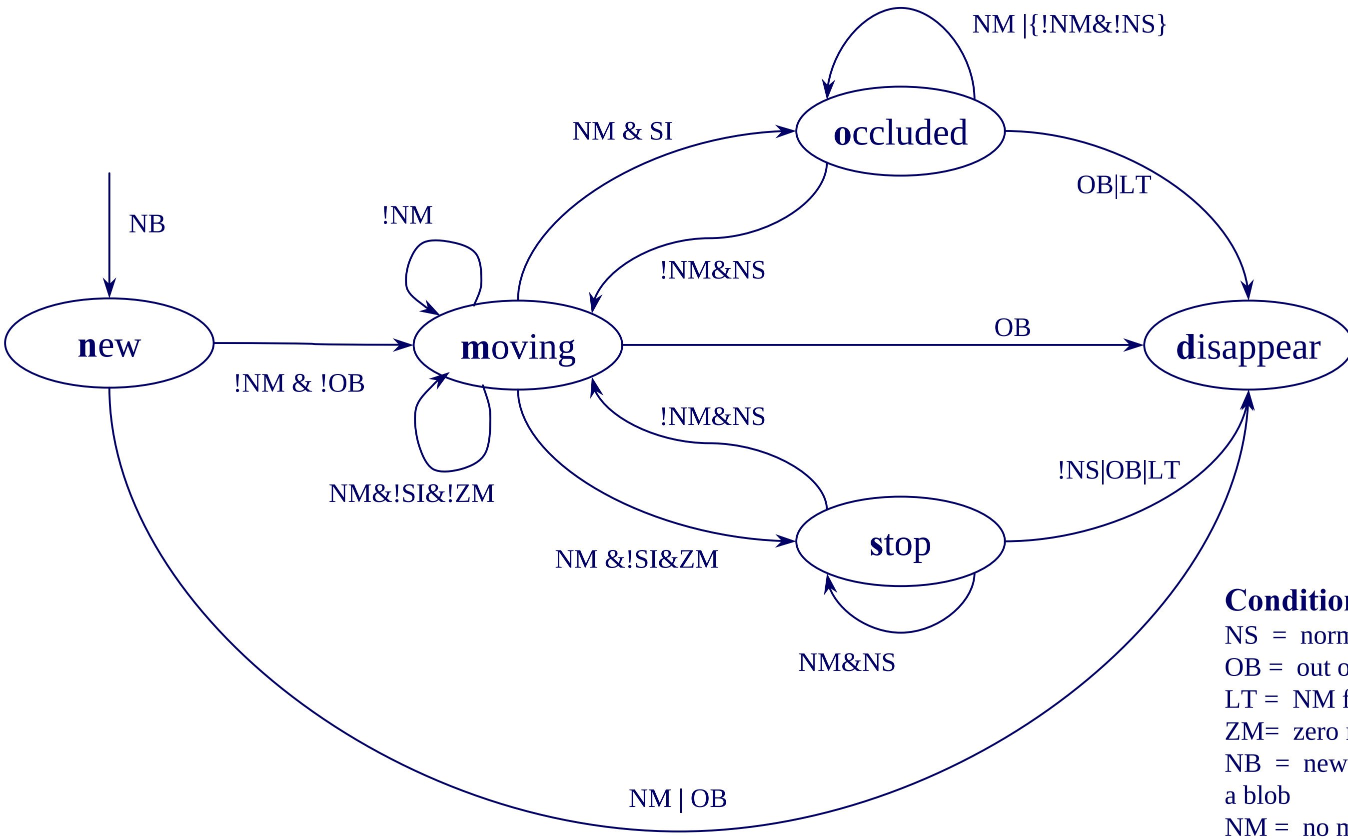
- Appearance estimation

$$A_{t,j}(T_j(x_i)) = \frac{A_{t,j}(T_j(x_i)) / \sigma_A^2 + h_{i,j} I_t(x_i) / \sigma_I^2}{(1/\sigma_A^2 + h_{i,j} / \sigma_I^2)}$$

Inference of object status

- A state transition graph is designed to
  - trigger events such as object initialization, object elimination
  - infer object states such as moving, stopping, two objects that are close to each other, etc.

Inference of Object Status

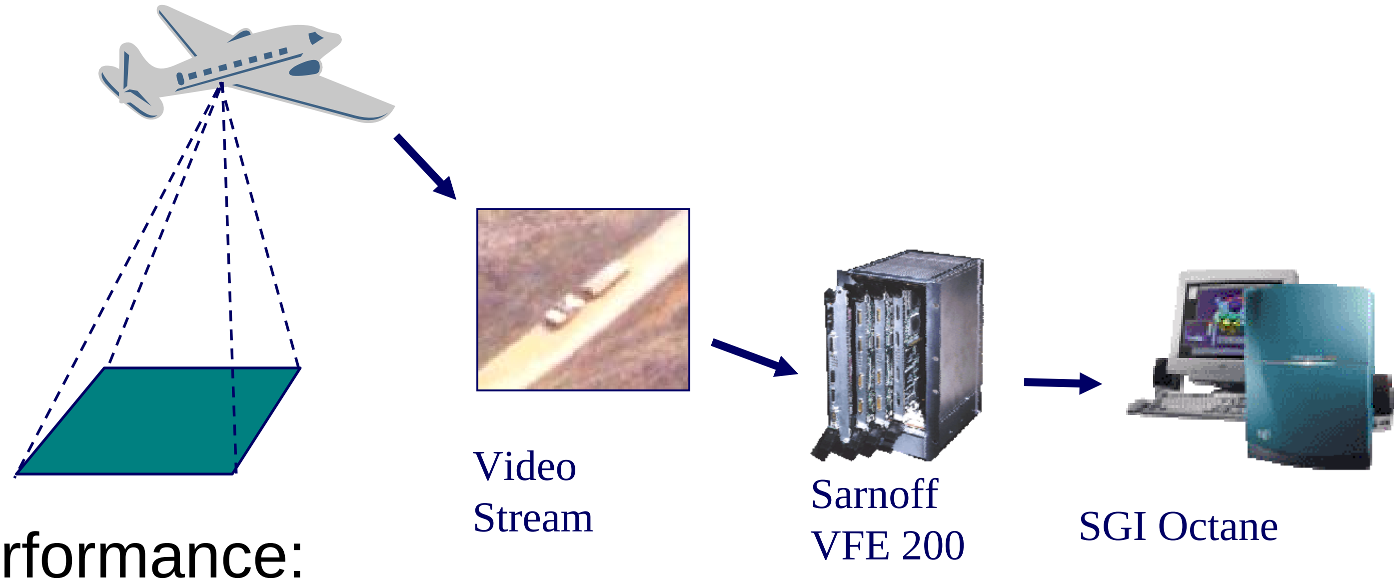


Conditions

NS = normal SSD score
OB = out of scope
LT = NM for a long time
ZM= zero motion estimation
NB = new blob, no object covering a blob
NM = no motion blob covering the object
SI = significant increase of SSD

Implementation - *Sarnoff Layer Tracker*

Airborne Video Surveillance System (tracking component)



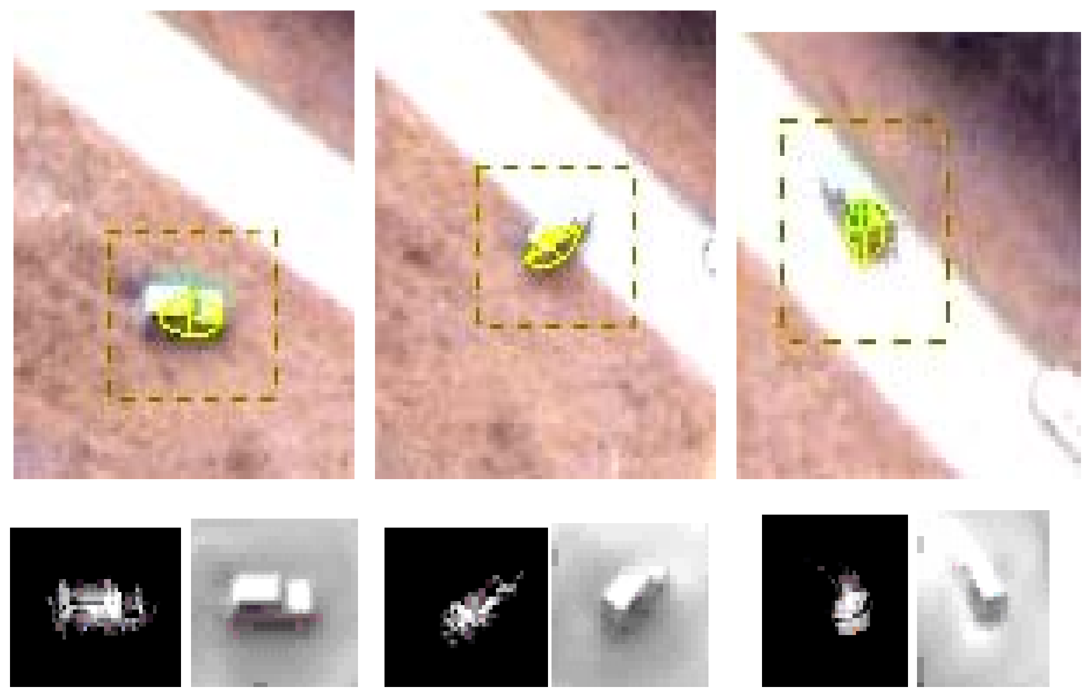
- Performance:
  - Originally developed on a PC, ported to SGI Octane. 20-25 Hz for one object over a single processor.

Results

- Turning

Results

- Turning



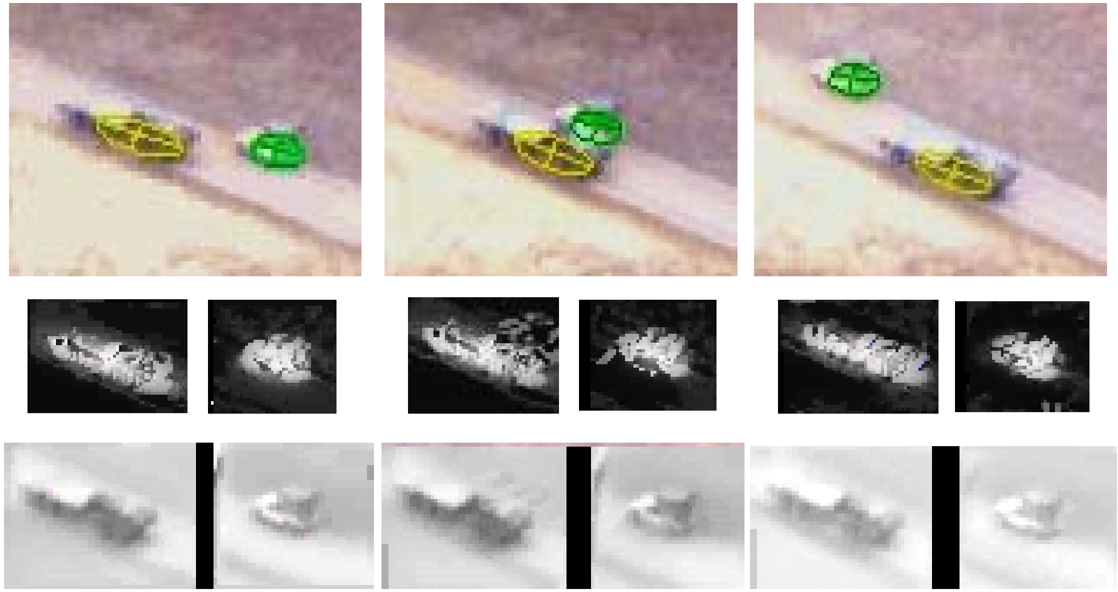
(a) (b) (c)

Results

- Passing - opposite directions

Results

- Passing - opposite directions



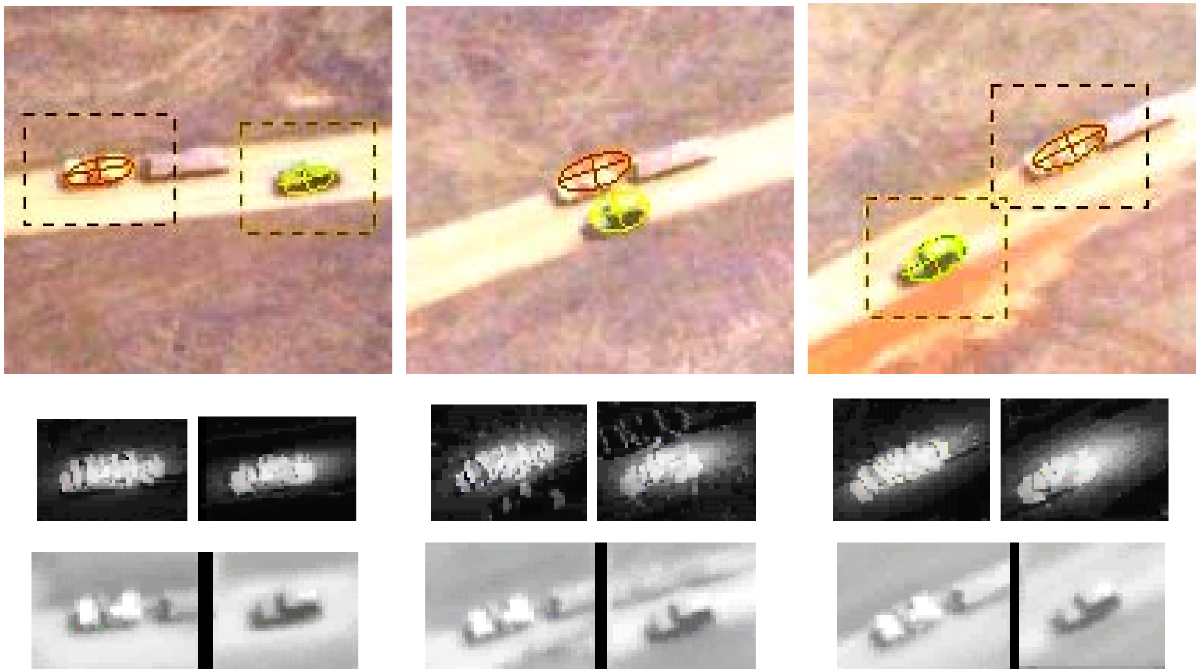
(a) (b) (c)

Results

- Passing - the same direction

Results

- Passing - the same direction



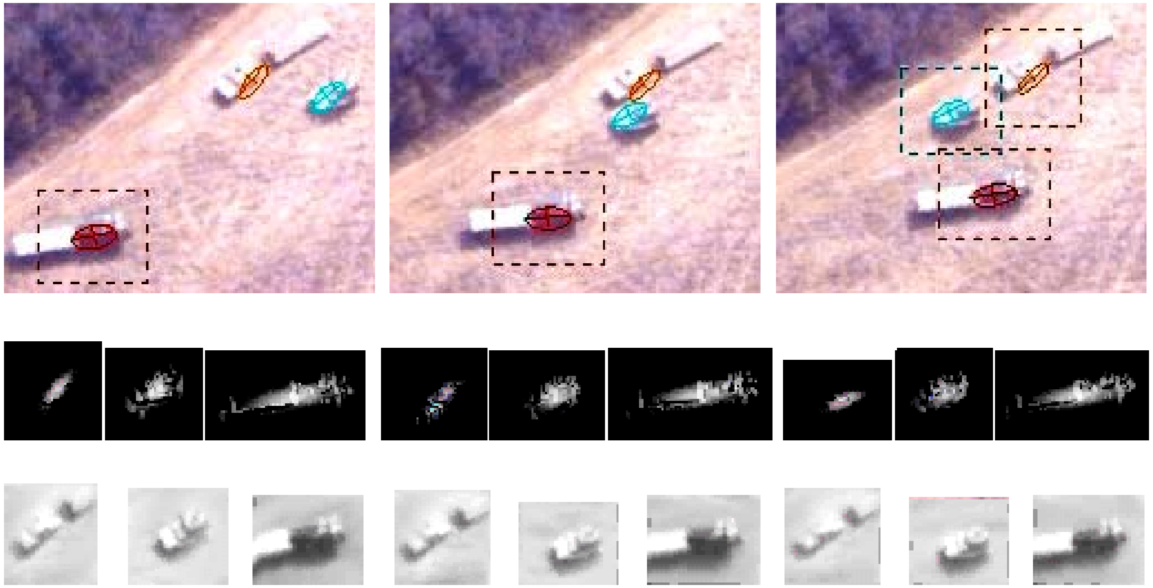
(a) (b) (c)

Results

- Stop, Passing

Results

- Stop, Passing



(a) (b) (c)

Implementation – *Sarnoff Layer Tracker*

- Motion estimation:
  - 95% of computation is for motion estimation. Currently, weighted SSD correlation is used. Searching in a 13x13 window at half resolution, for 3 different angles. The size of the object is around 40x40 pixels.
- Ownership estimation
  - change image is integrated into the formulation to further improve the robustness.
- Appearance estimation
  - appearance model for the background is not computed, instead, the previous image is used.

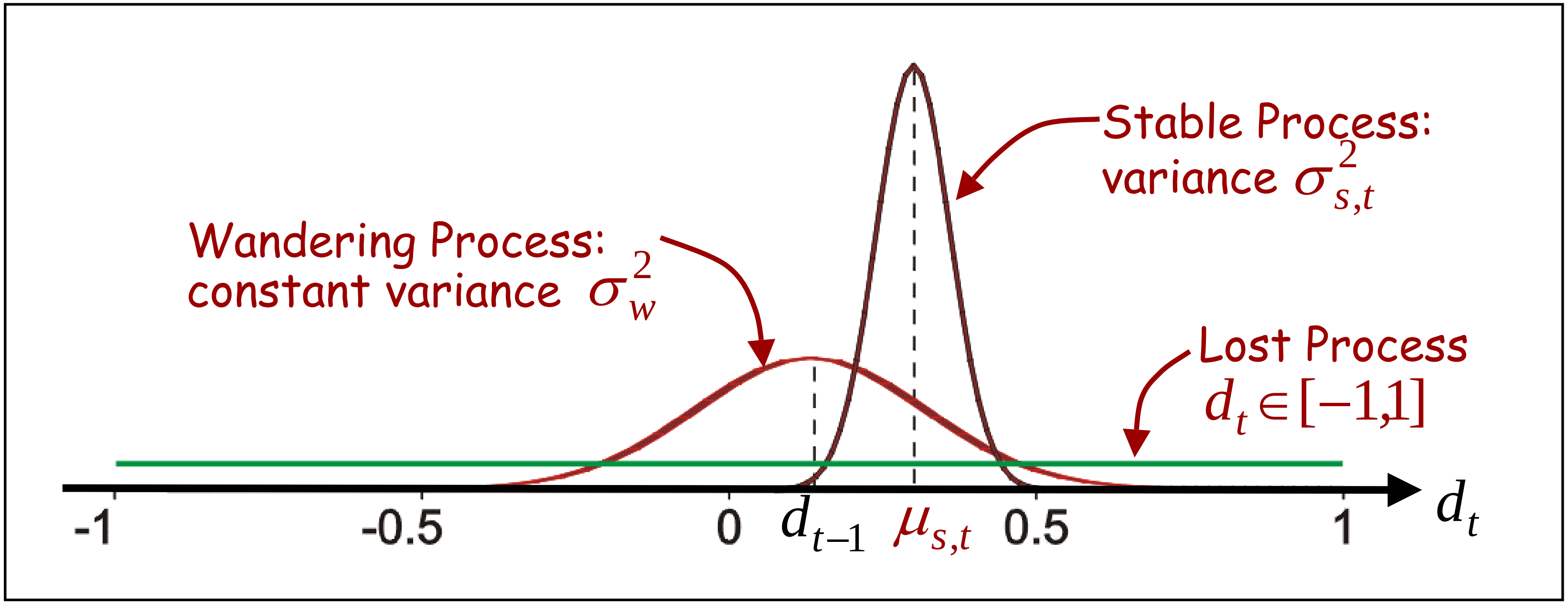
An Alternative Appearance Model

- In the previous model, appearance gets incrementally averaged over time since it is part of the state vector

- A more sophisticated appearance model allows for averaging as well as keeping up with frame-to-frame appearance changes:

  – Jepson et al.'s WSL model
  – A mixture model of appearance
  – Estimated incrementally using online EM

WSL Adaptive Model in 1D



Mixture model for current data (4 dof):

$$p(d_t \mid \mathbf{q}_t, \mathbf{m}_t, d_{t-1}) = m_s\, p_s(d_t|\mathbf{q}_t) + m_w\, p_w(d_t|d_{t-1}) + m_l\, p_l(d_t)$$

stable parameters
$\mathbf{q}_t = (\mu_{s,t}, \sigma^2_{s,t})$

mixing probabilities
$\mathbf{m}_t = (m_s, m_w, m_l)$

On-Line Approximate EM

One E-Step: Compute data ownerships only at current time

$$o_{j,t}(d_t) = \frac{m_{j,t-1}\, p_j(d_t \mid \mathbf{q}_{t-1}, d_{t-1})}{p(d_t \mid \mathbf{q}_{t-1}, \mathbf{m}_{t-1}, d_{t-1})} \qquad j \in \{w, s, l\}$$

One M-Step: Update weighted i^{th}-order data moments

$$M^{(i)}_{j,t} = \alpha\, o_{j,t}(d_t)\, d_t^{\,i} + (1-\alpha)\, M^{(i)}_{j,t-1}, \quad j \in \{w, s, l\}$$

Updated mixing probabilities (0^{th} order moments):

$$m_{j,t}(d_k) = M^{(0)}_{j,t}, \qquad j \in \{w, s, l\}$$

Updated mean and variance of stable process:

$$\mu_{s,t} = \frac{M^{(1)}_{s,t}}{M^{(0)}_{s,t}} \qquad \sigma^2_{s,t} = \frac{M^{(2)}_{s,t}}{M^{(0)}_{s,t}} - \mu^2_{s,t}$$

Estimation of Motion Parameters

To estimate the motion model parameters we maximize the sum of log likelihood and log prior :

$$O(\mathbf{u}_t) = \underbrace{\log L(D_t \mid \mathbf{u}_t, A_{t-1}, D_{t-1})}_{\text{likelihood}} + \underbrace{\log p(\mathbf{u}_t \mid \mathbf{u}_{t-1})}_{\text{prior}}$$

where:

warp parameters: $\mathbf{u}_t$

data at time t-1: $D_{t-1} = \{d(\mathbf{x}, t-1)\}_{\mathbf{x} \in \mathcal{R}_{t-1}}$

appearance model: $A_t = (\mathbf{q}_t, \mathbf{m}_t)$

parametric motion: $\mathbf{x}_t = \mathbf{w}(\mathbf{x}_{t-1}; \mathbf{u}_t)$

Optimization Details

Data Likelihood:

$$L(D_t \mid \mathbf{u}_t, A_{t-1}, D_{t-1}) = \prod_{\mathbf{x} \in \mathcal{R}_{t-1}} p(d(\mathbf{w}(\mathbf{x}; \mathbf{u}_t), t) \mid A_{t-1}, d(\mathbf{x}, t-1))$$

data from time t is warped back to t-1 and compared to predictions from the tracking region at time t-1.

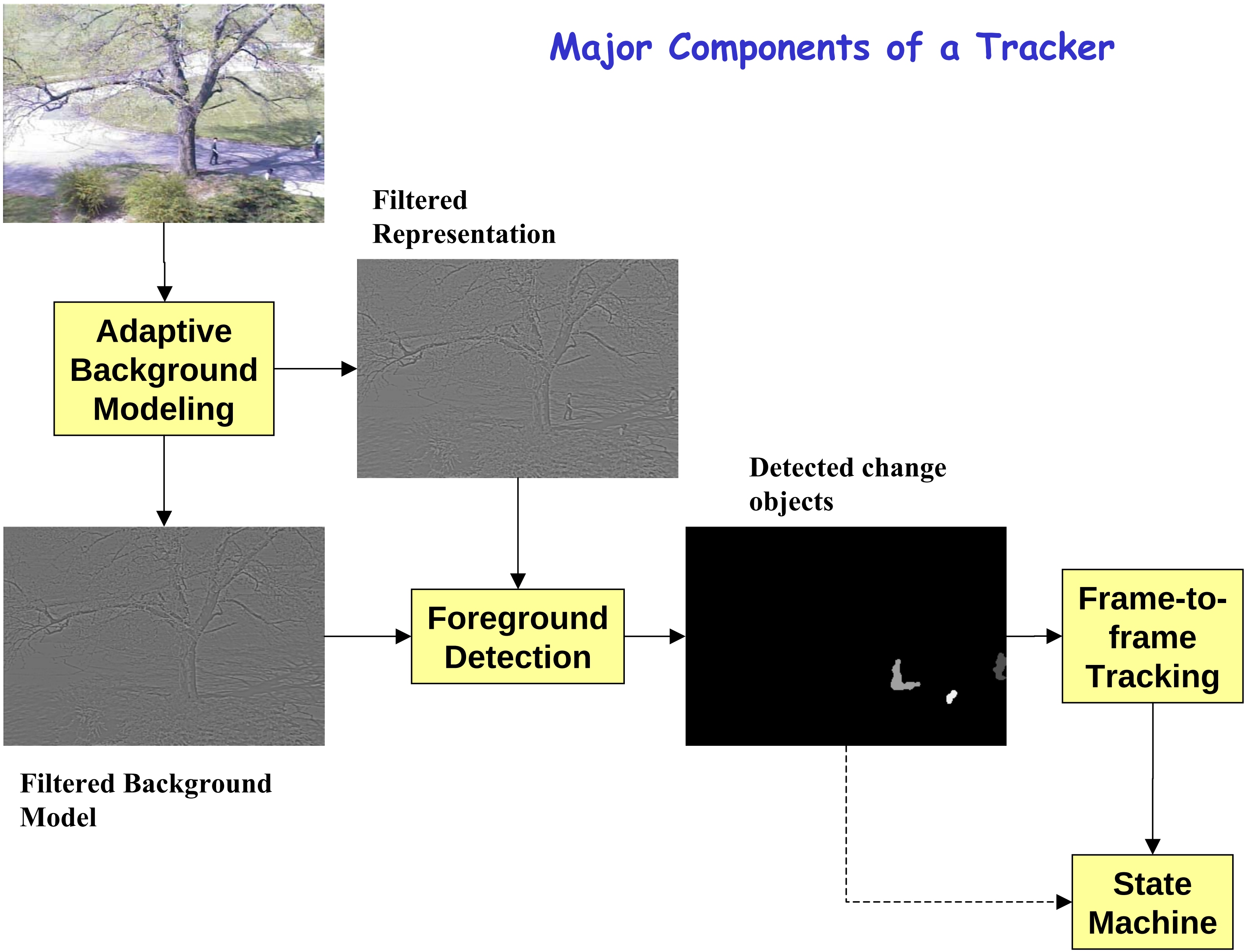
Motion Prior:

$$p(\mathbf{u}_t \mid \mathbf{u}_{t-1}) = \underbrace{G(\mathbf{u}_t, \sigma_0^2)}_{\text{slow}} \; \underbrace{G(\mathbf{u}_t - \mathbf{u}_{t-1}, \sigma_1^2)}_{\text{smooth}}$$

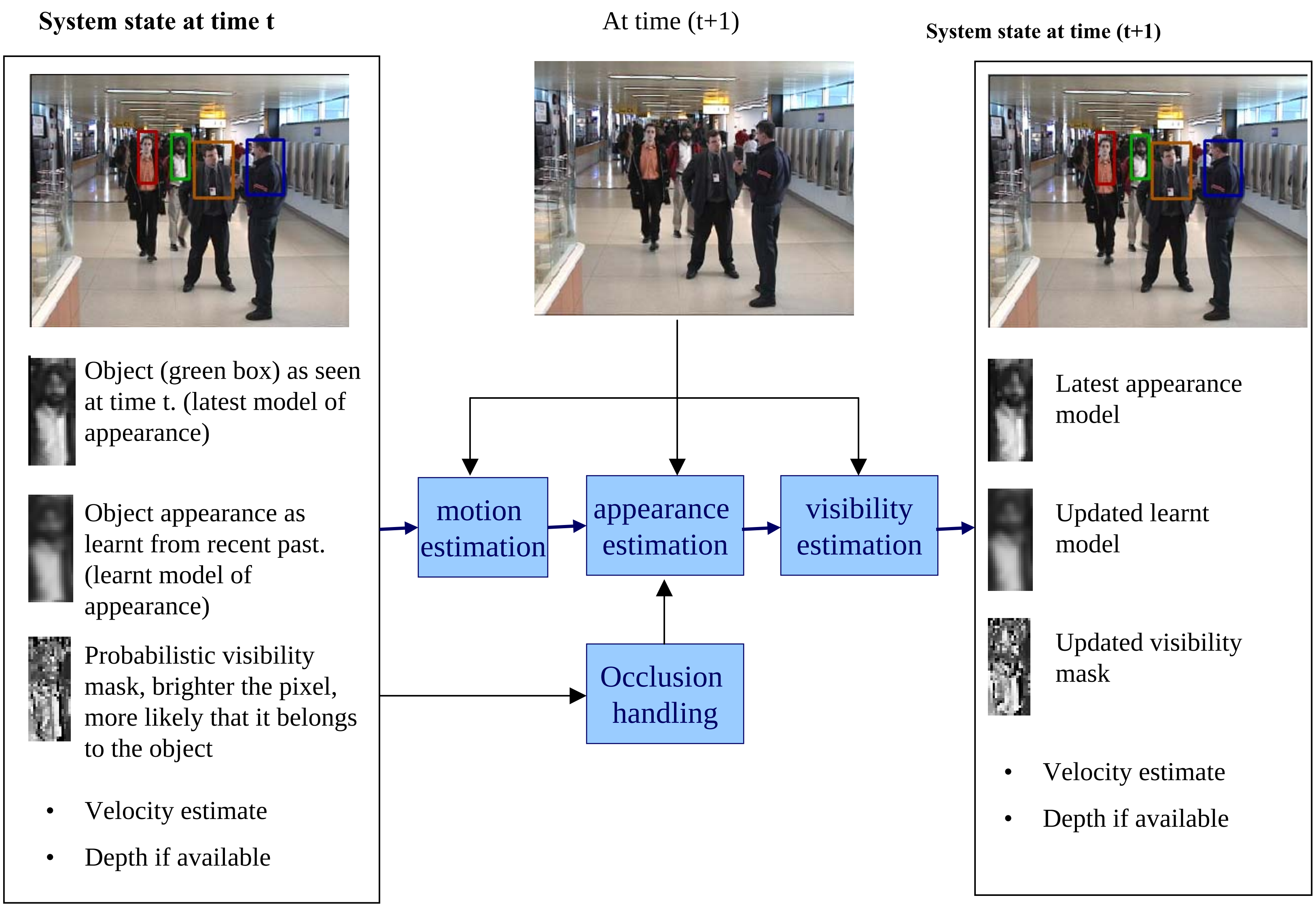
Fitting process for $\mathbf{u}_t$ is similar to fitting mixture models for flow (Jepson & Black, 1993).

Real-Time Tracking

Major Components of a Tracker



Tracker Block Diagram



Sample Progress of the Tracker

System state at time t=1

Occlusion is detected at this frame

Note learnt model is much more immune to occlusions than the latest model.

System state at time t=8

The appearance models and visibility mask are still frozen to t=8 because of occlusion

System state at time t=16

The object reappears after occlusion, and the models and visibility mask are updated

System state at time t=27

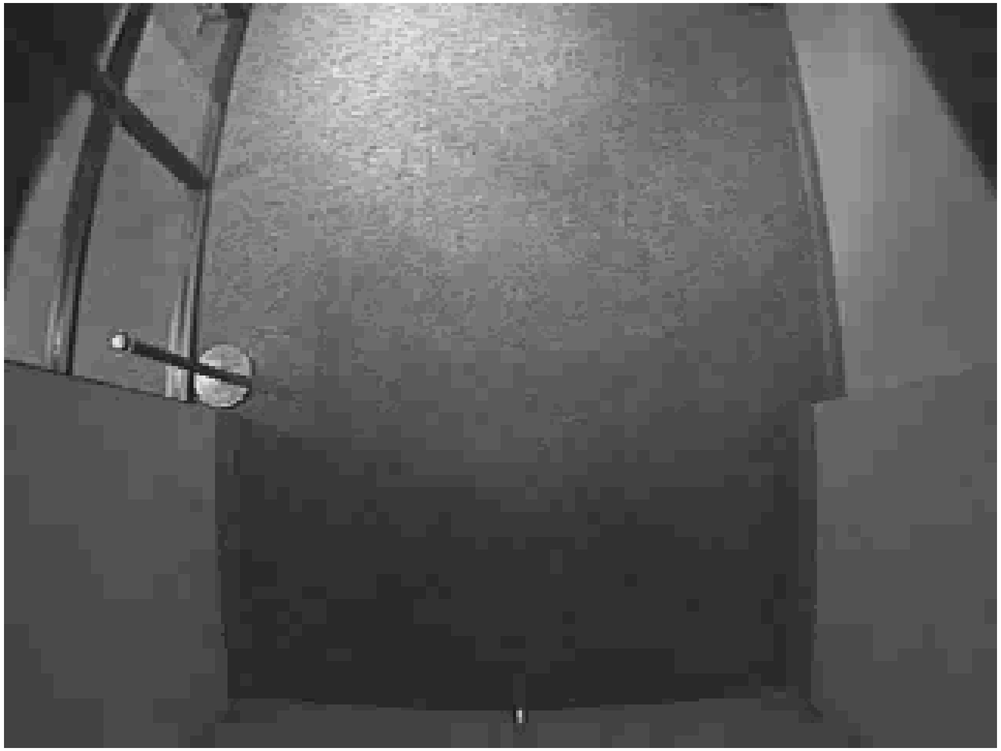
Tracker Features

- Non-parametric distribution based background representation.
  - Resilient to environmental effects like wind-induced motion, heat-induced scintillation etc.
- Foreground extraction based on pyramid filters and flow.
  - Tunable for different scenarios: outdoors, indoors.
- Comprehensive tracking based on appearance, motion and shape.
  - Automatically adapts to smooth and sudden changes of appearance.
  - Automatically weights appearance and shape matching.
  - Precise motion estimation based on optical flow.
- State machine that exploits appearance, motion and shape.
  - Handles occlusions, and confusing events with multiple objects.

Example: Outdoors

Example: Indoor Overhead

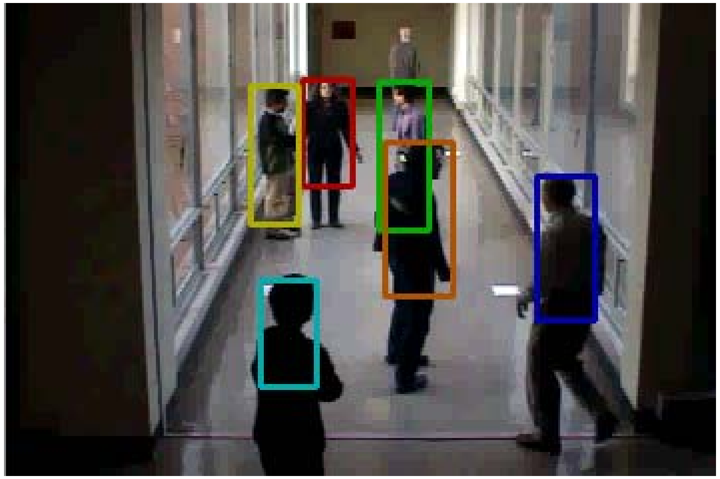
Example: Airport Overhead



Example: Airport (Light Traffic)

Example: Airport Sequence

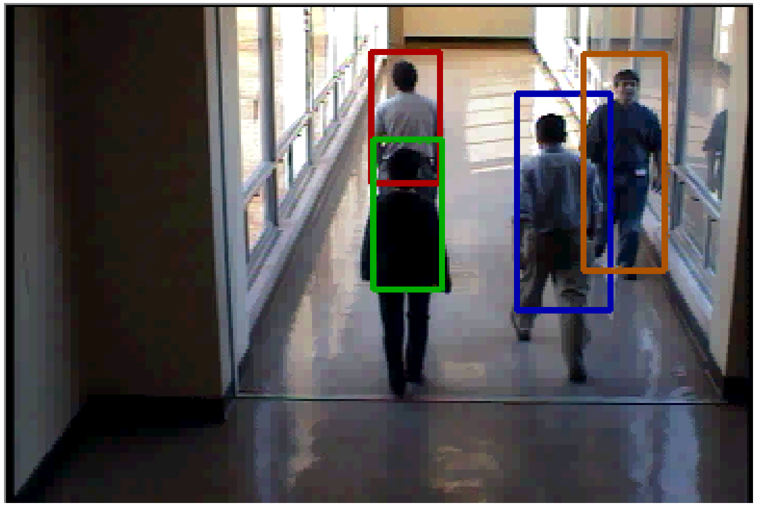
Example: Hallway Sequence



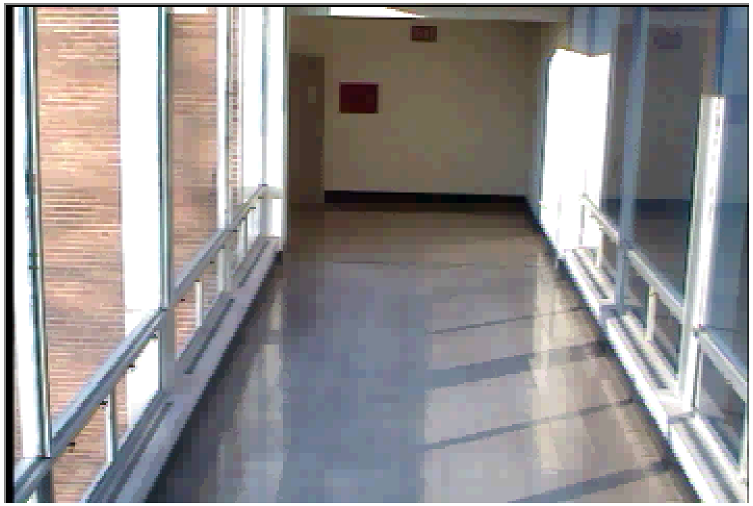
Example: Hallway Sequence

3D Tracking with Presence of Clutter and Multi-Camera Handoff

Camera 1



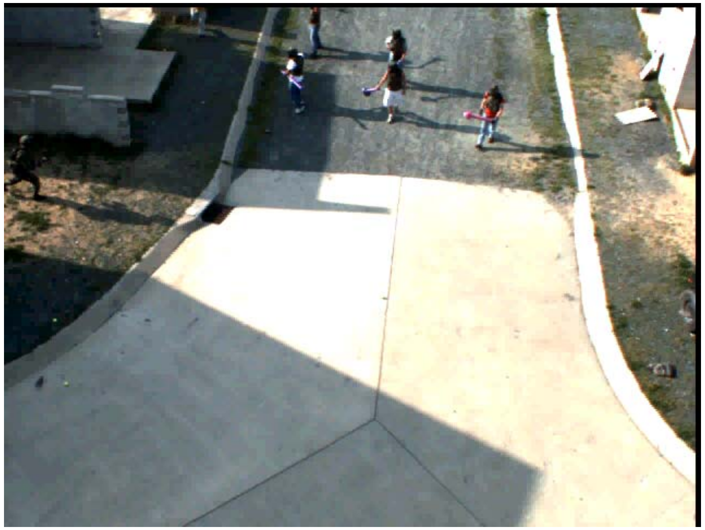
Camera 2



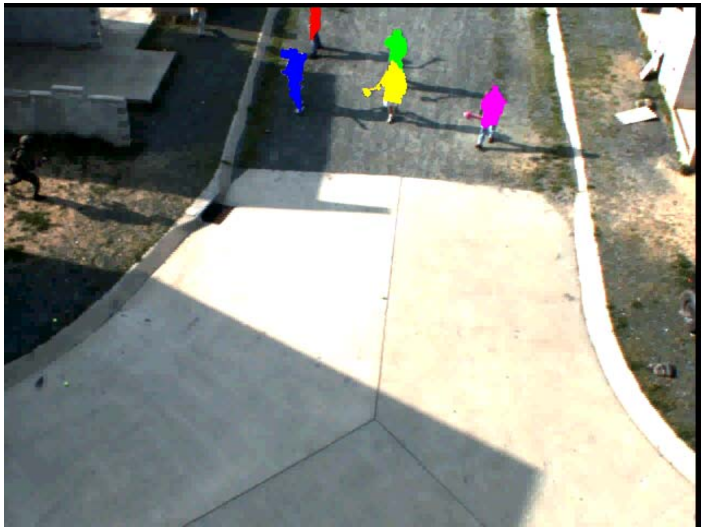
Video of Camera 1 and Camera 2

Handing-off from camera 1 to camera 2

3D Tracking in Outdoor Scenarios

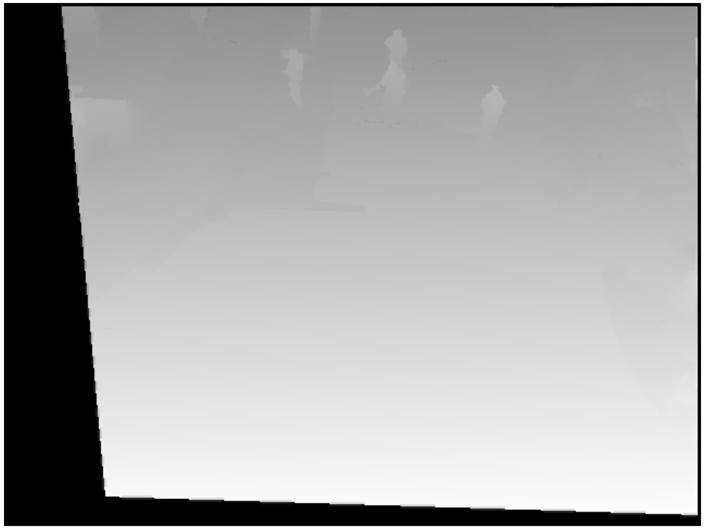


Original video



Video with entire mob being tracked simultaneously

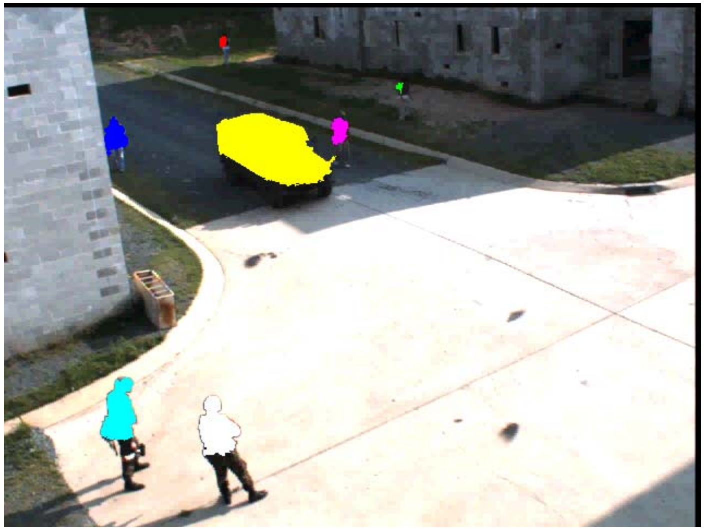
- Each color represents a different person in the image
- Note the 3D tracker can distinguish between people and their shadows



Depth Map Video

3D Tracking in Outdoor Scenarios

Original video



Video with people and vehicles being tracked simultaneously

Each color represents a different person/vehicle in the image

Depth Map Video

The End