Motivation

- Detection of interesting objects in videos is the first step in the process of automated surveillance.

- Focus of attention method greatly reduces the processing time required for tracking and activity recognition.
Introduction

- Objectives:
  - Given a sequence of images from a stationary camera identify pixels comprising ‘interesting’ objects.
  - All independently moving objects are interesting

- General Solution
  - Model properties of the scene (e.g. color, texture e.t.c) at each pixel
  - Significant change in the properties indicates an interesting change

Segmenting Background
Introduction

- Problems in Realistic situations:
  - Moving but uninteresting objects e.g. trees, flags or grass.
  - Long term illumination changes e.g. time of day.
  - Quick illumination changes e.g. cloudy weather
  - Shadows
  - Other Physical changes in the background
    - Dropping or picking up of objects
  - Initialization

Background Subtraction Method by Stauffer and Grimson

- In realistic scenarios multiple Processes are generating color ‘x’ at each pixel, where \( x = [R, G, B]^T \)

A method is required that can incorporate multiple colors in the background model
Solution

- For each pixel \((i,j)\) at time ‘\(t\)’ each process is modeled as a Gaussian distribution.
  - Gaussian distribution is described by a mean ‘\(m\)’ and a covariance matrix \(\Sigma\).
    \[
    N(x_{ij}^t | m_{ij}^t, \Sigma_{ij}^t) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2}(x_{ij}^t - m_{ij}^t)^T \Sigma_{ij}^{-1} (x_{ij}^t - m_{ij}^t)}
    \]
  - Weight \(\omega\) associated with each distribution signifying relevance in recent time.
- Thus each Pixel is modeled as a mixture of Gaussians.

Multimodality
**Mixture of Gaussians**

- Finding the mean and variance for one Gaussian is easy
- Much tougher for the Mixture of Gaussians case
- Find
  - Number of Gaussians
  - Mean and variance of each Gaussian

**Solution**

- At each frame
  - Calculate Mahalanobis distance of pixel's color value from each of the associated K Gaussian distributions

```
w1  
w2  
w3  

Distributions at t-1  Pixel at t  Distributions at t
```

- \( w_1 \rightarrow \)
- \( w_2 \rightarrow \)
- \( w_3 \rightarrow \)
Solution

- If a match is found with the $k$th Gaussian, update parameters

\[ m_{i,j}^{t,k} = (1 - \rho) m_{i,j}^{t-1,k} + \rho x_{i,j}^{t} \]

\[ \sum_{i,j}^{t,k} = (1 - \rho) \sum_{i,j}^{t-1,k} + \rho (x_{i,j}^{t} - m_{i,j}^{t}) (x_{i,j}^{t} - m_{i,j}^{t})^T \]

where $\rho$ is a learning parameter.

Solution

- If a match is not found
  - Replace lowest weight distribution with a new distribution such that
    \[ m_{i,j}^{t,new} = x_{i,j}^{t} \]
    \[ \sum_{i,j}^{t,new} = \sum_{i,j}^{initial} \]
  - The prior weights of $K$ distributions are adjusted as
    \[ \omega_{i,j}^{t-1} = (1 - \alpha) \omega_{i,j}^{t-1} + \alpha (M_{i,j}^{t-1}) \]
  - $M$ is 1 for model that matched and 0 for others
Background Subtraction

- Problem: Choosing a threshold
  - Pixel is foreground if $|l_1(x,y) - l_2(x,y)| \leq \lambda$
  - otherwise background?
  - What is the correct value of $\lambda$?

Setting a Threshold

$\lambda = 10$  $\lambda = 20$  $\lambda = 50$

$\lambda = 100$  $\lambda = 200$  $\lambda = 300$
Pros

- Handles slow changes in illumination conditions
- Can accommodate physical changes in the background after a certain time interval.
- Initialization with moving objects will correct itself after a certain time interval.

Cons

- Can’t handle quick changes in illumination conditions e.g. cloudy weather.
- Initialization with moving objects
- Physical changes in background
- Shadows
Results

[Image of an outdoor scene]

Results

[Image of an indoor scene]
Results

Summary of Algorithm

- **Learn** background model by watching 30 second video
- **Detect** moving object by measuring deviations from background model, and applying connected component to foreground pixels.
- **Update** background and blob statistics
Background Models
Camera Motion

- Mobile camera
- Nominal motion

Removing Camera Motion

- Affine Transformation (Anandan)
- Projective Transformation (Mann-Pickard)
Projective Flow (weighted)

\[ u f_x + v f_y + f_t = 0 \]  \quad \text{Optical Flow const. equation}

\[ u^T f_x + f_t = 0 \]

\[ x' = \frac{A x + b}{C^T x + 1} \]  \quad \text{Projective transform}

\[ u = x' - x = \frac{A x + b}{C^T x + 1} - x \]

Projective Flow (weighted)

\[ E_{flow} = \sum (u^T f_x + f_t)^2 \]

\[ = \sum \left( \frac{A x + b}{C x^T + 1} - x \right)^T f_x + f_t \right)^2 \]

\[ = \sum \left( (Ax + b - (C^T x + l)x)^T f_x + (C^T x + l)f_t \right)^2 \]

\[ \downarrow \text{minimize} \]
Projective Flow (weighted)

$$\left( \sum \phi \phi^T \right) a = \sum (x^T f_x - f_i) \phi$$

$$p = [a_1, a_2, b_1, a_3, a_4, b_2, c_1, c_2]^T$$

$$\phi' = [f_x, f_y, f_x, f_y, f_x f_y, f_y, f_y f_x - x^2 f_x - xy f_y, y f_x - y^2 f_y]$$

Projective Flow (unweighted)
Pseudo-Perspective

\[ x' = \frac{A x + b}{C^T x + 1} \]

Taylor Series

\[ x + u = a_1 + a_2 x + a_3 y + a_4 x^2 + a_5 xy \]
\[ y + v = a_6 + a_7 x + a_8 y + a_4 xy + a_5 y^2 \]

Bilinear

\[ x' = \frac{A x + b}{C^T x + 1} \]

Taylor Series & remove Square terms

\[ u + x = a_1 + a_2 x + a_3 y + a_4 xy \]
\[ v + y = a_5 + a_6 x + a_7 y + a_8 xy \]
Projective Flow (unweighted)

\[ \mathcal{E}_{\text{flow}} = \sum (u^T f_x + f_t)^2 \]

Minimize

Bilinear and Pseudo-Perspective

Homework: Derive this equation Due Feb 8

\[ \sum \Phi \Phi^T \mathbf{q} = -\sum f_t \Phi \]

\[ \Phi^T = \begin{bmatrix} f_x(xy, x, y, 1), f_y(xy, x, y, 1) \end{bmatrix} \]  \hspace{1cm} \text{bilinear} 

\[ \Phi^T = \begin{bmatrix} f_x(x, y, 1) & f_y(x, y, 1) & c_1 & c_2 \end{bmatrix} \]

\[ c_1 = x^2 f_x + xy f_y \]

\[ c_2 = xy f_x + y^2 f_y \]  \hspace{1cm} \text{Pseudo perspective}
Algorithm-1

- Estimate “q” (using approximate model, e.g. bilinear model).
- Relate “q” to “p”
  - select four points S1, S2, S3, S4
  - apply approximate model using “q” to \((x'_k, y'_k)\)
  - compute
  - estimate exact “p”:

Flow

\[ \text{Flow diagram} \]
True Projective

\[
x' = \frac{a_1 x + a_2 y + b_1}{c_1 x + c_2 y + 1}
\]

\[
y' = \frac{a_3 x + a_4 y + b_1}{c_1 x + c_2 y + 1}
\]

\[
\begin{bmatrix}
x'_k \\
y'_k
\end{bmatrix} =
\begin{bmatrix}
x_k & y_k & 1 & 0 & 0 & 0 & -x_k x'_k & -y_k y'_k \\
0 & 0 & 0 & x_k & y_k & 1 & -x_k y'_k & -y_k y'_k
\end{bmatrix}
\begin{bmatrix}
a_1 & a_2 & b_1 & a_3 & a_4 & b_2 & c_1 & c_1
\end{bmatrix}
\]

\[
a = [a_1 \ a_2 \ b_1 \ a_3 \ a_4 \ b_2 \ c_1 \ c_1]^
\]

\[
\begin{bmatrix}
x'_1 \\
y'_1
\end{bmatrix} =
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 x'_1 & -y_1 x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 y'_1 & -y_1 y'_1
\end{bmatrix}
\begin{bmatrix}
a
\end{bmatrix}
\]

\[
\begin{bmatrix}
x'_k \\
y'_k
\end{bmatrix} =
\begin{bmatrix}
x_k & y_k & 1 & 0 & 0 & 0 & -x_k x'_k & -y_k y'_k \\
0 & 0 & 0 & x_k & y_k & 1 & -x_k y'_k & -y_k y'_k
\end{bmatrix}
\begin{bmatrix}
a
\end{bmatrix}
\]

\[
P = Aa
\]

Perform least squares fit to compute a.
Final Algorithm

- A Gaussian pyramid of three or four levels is constructed for each frame in the sequence.
- The parameters “p” are estimated at the top level of the pyramid, between the two lowest resolution images, “g” and “h”, using algorithm-1.

The estimated “p” is applied to the next higher resolution image in the pyramid, to make images at that level nearly congruent.
- The process continues down the pyramid until the highest resolution image in the pyramid is reached.
Presentations

- Support Vector Machines: Vladimir (February 8, 2006)
- Graph-Cut: Paul (February 13, 2006)
- Adaboost: Alex (February 15, 2006)