Probabilistic visual learning for object representation

(B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696-710, 1997 (*on-line*)).

• The problem

- Learning and modeling the visual appearance of objects (i.e., faces and human hands) from a set of training imagery.

- Given a set of training images x^t, $t = 1, 2, \ldots, N$ from an object class Ω, estimate the likelihood function $p(x/\Omega)$.

• The approach

- Perform density estimation in high-dimensional spaces using an eigenspace decomposition.

- This is more efficient than performing the density estimation in the original high-dimensional space.

- Two types of density estimates are derived for modeling the training data:

  (1) a multivariate Gaussian

  (2) a mixture-of-Gaussians

- These densities are the used to compute likelihoods for object detection.

• Density estimation in eigenspace

- The goal is to estimate the *complete* probability density of the object's appearance using an eigenvector decomposition of the image space.

- The probability density is decomposed into two components:

(1) the density in the principal subspace (i.e., defined by the principal components).

(2) the density in its orthogonal complement (i.e., defined by the components that are thrown away).

• Quick review of eigenspace theory

- Given a set of image vectors x^t, t=1,2,..,N^T, we identify the largest components M of the covariance matrix Σ.

- Each image vector is then expressed in a new coordinate system (defined by the principal eigenvectors):

$$y = \Phi_M^T \tilde{x}$$

where $\tilde{x} = x - \bar{x}$ and the columns of Φ are the principal components.

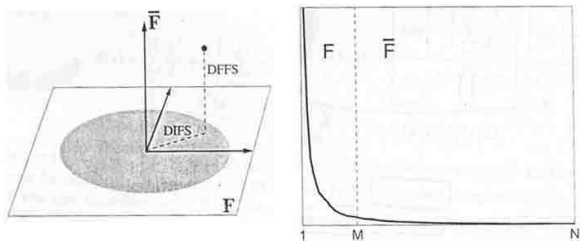
Principal subspace F: the space defined by the principal components Φ_i, $i = 1, 2, .., M$

Orthogonal complement $\bar{F}$: the space defined by the other components Φ_i, $i = M + 1, .., N$

Distance in face space (DIFS): distance computed in the space of principal components.

Distance from face space (DFFS): (1) project, (2) reconstruct, (3) subtract from original (essentially the distance computed in the space of the non-principal components.

$$\varepsilon^2(x) = \sum_{i=M+1}^{N} y_i^2 = \|\tilde{x}\| - \sum_{i=1}^{M} y_i^2 \text{ (residual)}$$



- **Modeling the density as a high-dimensional Gaussian**

  - Let's assume that the density of faces in class Ω is modeled by a high-dimensional Gaussian whose parameters $(\bar{x}, \Sigma)$ have been estimated:

$$p(x/\Omega) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} exp[-\frac{1}{2}(x-\bar{x})^t \Sigma^{-1}(x-\bar{x})]$$

  - The DIFS should be computed using the Mahalanobis distance:

$$d(x) = (x-\bar{x})^t \Sigma^{-1}(x-\bar{x})$$

  (this is essentially $P(\Omega/x)$ assuming equal priors)

  - In most applications, we simply discard the $\bar{F}$-space and work with $p_F(x/\Omega)$

  - To compute the likelihood of an observation x, however, it might be important to use the complete density $p(x/\Omega)$ (i.e., we need $p_{\bar{F}}(x/\Omega)$).

  - There are an infinity of vectors which are not members of Ω and can have similar F-space projections.

  - We may get many false positives without using $p_{\bar{F}}(x/\Omega)$

- **Computing $d(x)$ assuming a Gaussian density**

  - We can rewrite $d(x)$ using the decomposition of Σ (i.e., $\Sigma = \Phi\Lambda\Phi^T$)

$$d(x) = (x - \bar{x})^t \Sigma^{-1} (x - \bar{x}) = \tilde{x}^t \Phi \Lambda^{-1} \Phi^T \tilde{x} = y^T \Lambda^{-1} y = \sum_{i=1}^{N} \frac{y_i^2}{\lambda_i}$$

  - Evaluating the above expression explicitly is not feasible due to the high dimensionality (i.e., we do not compute all the eigenvalues in practice).

  - We need to estimate $d(x)$ using only M eigenvalues (i.e., those corresponding to the principal components).

$$d(x) = \sum_{i=1}^{N} \frac{y_i^2}{\lambda_i} = \sum_{i=1}^{M} \frac{y_i^2}{\lambda_i} + \sum_{i=M+1}^{N} \frac{y_i^2}{\lambda_i}$$

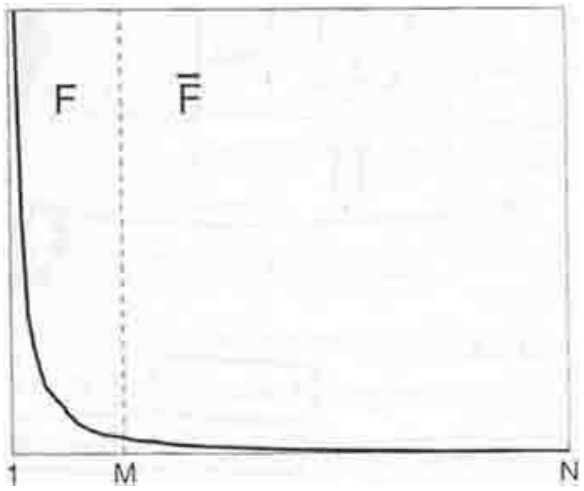
  - We will approximate $d(x)$ with the following expression:

$$\hat{d}(x) = \sum_{i=1}^{M} \frac{y_i^2}{\lambda_i} + \frac{1}{\rho} \sum_{i=M+1}^{N} y_i^2 = \sum_{i=1}^{M} \frac{y_i^2}{\lambda_i} + \frac{\varepsilon^2(x)}{\rho}$$

  - It can be shown that the optimal value ρ^* of ρ is given by:

$$\rho^* = \frac{1}{N - M} \sum_{i=M+1}^{N} \lambda_i$$

  - The value of ρ^* can be computed by extrapolating the eigenspace spectrum (e.g., fi t a function like $1/f$ to the available eigenvalues and the fact that the last eigenvalue is the estimated pixel noise variance).



- Substituting $\hat{d}(x)$ in $p(x/\Omega)$ we get $\hat{p}(x/\Omega)$:

$$\hat{p}(x/\Omega) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} exp[-\frac{1}{2}(\sum_{i=1}^{M}\frac{y_i^2}{\lambda_i} + \frac{\varepsilon^2(x)}{\rho})] =$$

$$\frac{1}{(2\pi)^{M/2}\prod_{i=1}^{M}\lambda_i^{1/2}} exp[-\frac{1}{2}(\sum_{i=1}^{M}\frac{y_i^2}{\lambda_i})] \cdot \frac{1}{(2\pi)^{(N-M)/2}\prod_{i=M+1}^{N}\lambda_i^{1/2}} exp[-\frac{1}{2}(\frac{\varepsilon^2(x)}{\rho})] =$$

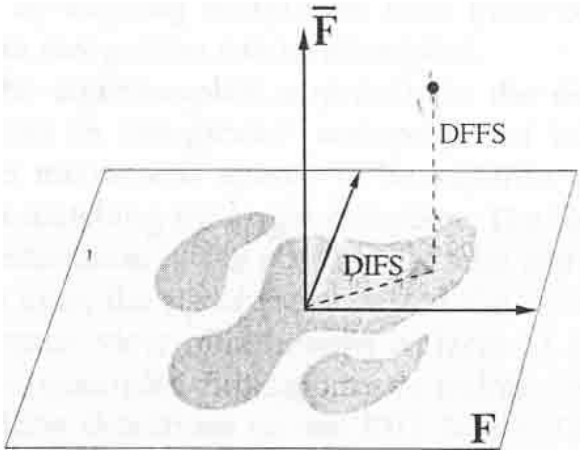
$$\frac{1}{(2\pi)^{M/2}\prod_{i=1}^{M}\lambda_i^{1/2}} exp[-\frac{1}{2}(\sum_{i=1}^{M}\frac{y_i^2}{\lambda_i})] \cdot \frac{1}{(2\pi\rho)^{(N-M)/2}} exp[-\frac{1}{2}(\frac{\varepsilon^2(x)}{\rho})] =$$

$$p_F(x/\Omega) \cdot \hat{p}_{\bar{F}}(x/\Omega)$$

- **Modeling the density as a mixture of Gaussians**

- When the training set represents multiple views or multiple objects under varying illumination conditions, the density of the training views in the F-space is no longer unimodal.

- The training data tends to lie on complex and non separable low-dimensional manifolds in image space.



- To deal with this issue, we need to assume that $p_F(x/\Omega)$ is an arbitrary density $p(y/\theta)$ ($\hat{p}_F(x/\Omega)$ is still assumed to be Gaussian).

$$\hat{p}(x/\Omega) = p(y/\theta)\ \hat{p}_F(x/\Omega)$$

- We can model $p(y/\theta)$ as a mixture of Gaussians:

$$p(y/\theta) = \sum_{k=1}^{K} p(y/\theta_k)\pi_k$$

where θ are the parameters of the mixture (each $p(y/\theta_k)$ is a Gaussian).

- The parameters of the mixture can be estimated using the EM algorithm.

- **Face detection using** $\hat{p}(x/\Omega)$

  - Compute the posterior probabilities $P(\Omega/x)$ at each location of the image using $\hat{p}(x/\Omega)$ (*saliency map*).

  - Use different window sizes for handling scale (*multiscale saliency maps*).

  - Local maxima in the saliency map correspond to faces.

- **Experiment1: detection of facial features**

  - Detect facial features like the eyes, the nose, and the mouth (e.g., needed to normalize faces for face recognition).

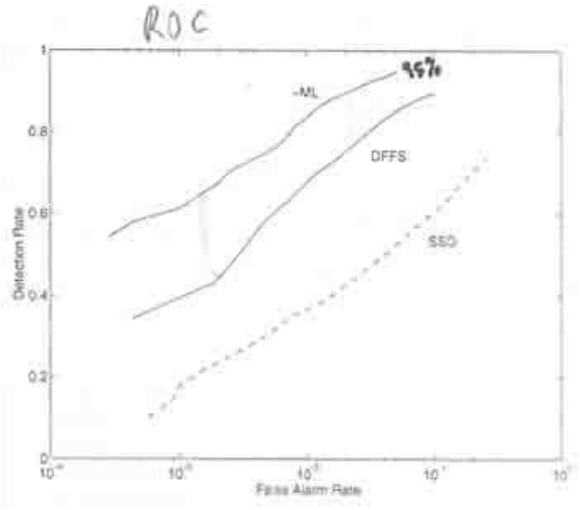
  - A database including 7,562 "mugshot" images were used in this experiment.

- Three methods have been compared using 7,000 images from that database:

(1) template matching using SSD (sum-of-squared differences) - the template was based on the average facial feature (e.g., left eye).

(2) DFFS template (Euclidean distance, 5 principal components)

(3) ML detector (a single Gaussian, 5 principal components)



• Experiment2: detection of faces

- The mutiscale detector was also tested on 2,000 images from the FERET database.

- A single Gaussian was used - 10 principal components.

- Correct detection rate was 97%.

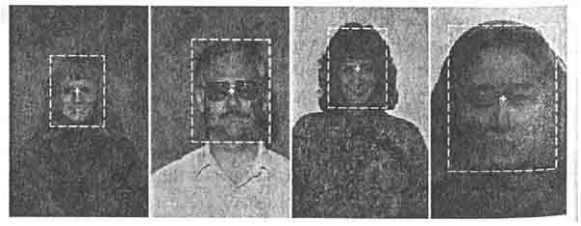
- **Experiment3: recognition of frontal faces**

  - A database with 7,562 images of 3,000 people (men, women, children, different races) was used.

  - A representative set of 128 faces were used for computing the eigenfaces.

  - A single Gaussian was used - 20 principal components.

  - The average recognition rate was assessed by performing several experiments (200 faces selected at random for testing) - the system achieved 95% recognition rate.



- **Experiment4: recognition under general viewing conditions**

- Given N individuals under M different views we could:

(1) Use Nayar's approach (parametric eigenspace)

(2) Build M distinct eigenspaces (one per view) - (view-based eigenspace)

- They argue that the "view-based" eigenspace approach models a complex distribution by the union of several simpler distributions.

- To determine the location and orientation of a face in an input image, we compute its likelihood in each eigenspace and select the maximum.

- The view-based approach was evaluated using 9 views of 21 people (189 images, -90 to 90 degrees).

Interpolation performance

* Trained with ±90 degrees, ±45 degrees, ±0 degrees
* Tested with ±68 degrees and ±23 degrees
* Achieved 90% average recognition performance.

Extrapolation performance

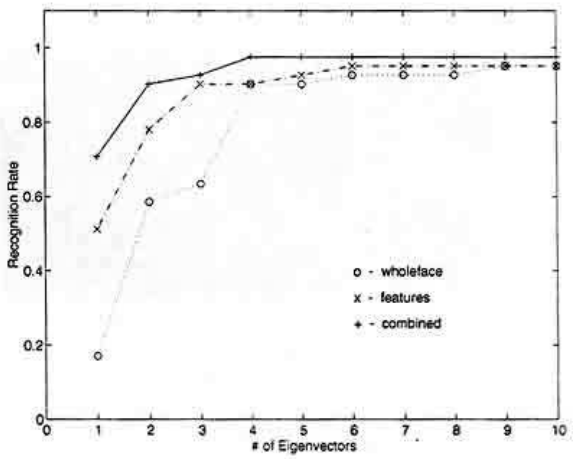
* Trained with images e.g., in the interval (-90 degrees to +45 degrees)
* Tested with view e.g., +68 degrees, and +90 degrees
* For testing views separated by ±23 degrees from the training images, the average recognition rate was 83%
* For testing views separated by ±45 degrees from the training images, the average recognition rate was 50%

- **Experiment5: modular recognition**

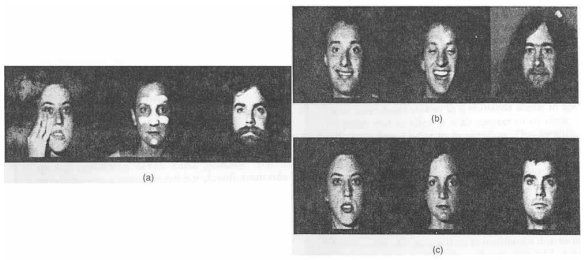
  - Build a separate eigenspace for each facial feature (i.e., eigeneyes, eigennoses, eigenmouths).

  - Used a database with 2 views of 45 people.

  - Used half for training and half for testing.



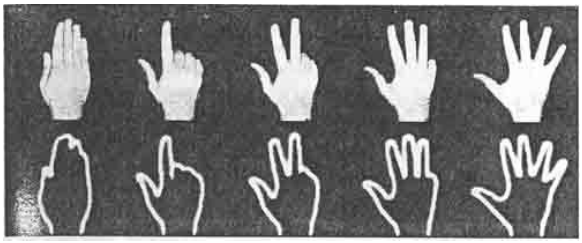
  - The eigenfeature approach is really superior when there are variations in the input image (e.g., hats, beards etc.)



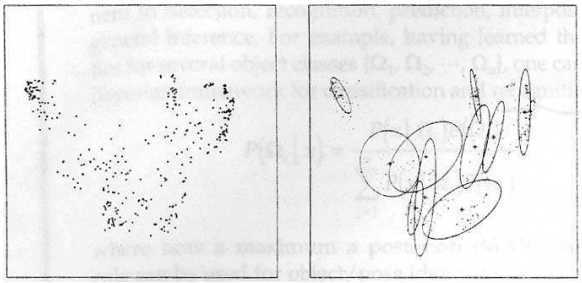
- **Experiment6: hand detection and recognition**

  - The human hand is a highly articulated, non-rigid object.

  - A shape representation (edge-based) is used to characterize its identity (i.e., texture not important).

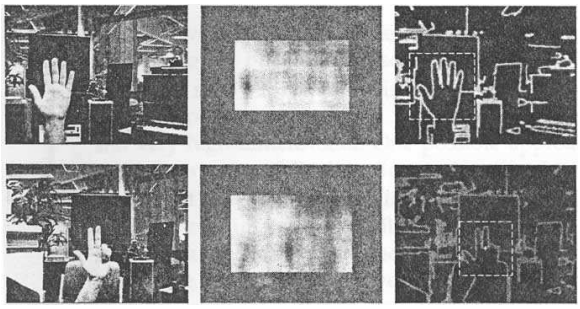


- A mixture of Gaussians is used in this case for hand detection (the distribution of hand gestures is multimodal).



- Used 5 mixture components and 10 principal components.

- Present some results where the hand is detected successfully assuming complex background.



- Once the hand has been detected, recognition is very very accurate even using a simple Euclidean distance (100% success rate assuming 375 frames from an image sequence containing 7 gestures).

Tracking color objects using adaptive mixture models

(S. McKenna, Y. Raja, and S. Gong, "Tracking color objects using adaptive mixture models", *Image and Vision Computing*, vol. 17, pp. 225-231, 1999 (*on-line*))

• The problem

- Tracking color objects in real-time under varying illumination, viewing geometry, and camera parameters.

• Current approaches

- Use non-parametric models based on histograms (requires lots of data).

- Use only one Gaussian whose parameters are adapted over time.

• The approach

- A statistical model is proposed for modeling color distributions over time.

- The model is based on adaptive Gaussian mixtures for real-time tracking color objects.

  (1) A set of predetermined generic object color models can be used to initialize (or re-initialize) the tracker.

  (2) During tracking, the model adapts and improves its performance by becoming specifi c to the observed conditions.

- The model performs satisfactory under varying illumination, viewing geometry, and camera parameters.

• Assumptions

- The number of components is kept fi xed (determined using a fi xed data set).

- This implies that the number of components needed to accurately model an object's color does not change signifi cantly with changing viewing conditions.

• Color mixture models

- The conditional density of a pixel x_i, belonging to an object, O ($i = 1, 2, \ldots, n$), is modeled as a Gaussian mixture with K components (x_i is two-dimensional):

$$p(x_i/O) \equiv p(x_i/\theta) = \sum_{k=1}^{K} p(x_i/\theta_k)\pi_k$$

- Each component $p(x_i/\theta_k)$ is modeled as a Gaussian:

$$p(x_i/\theta_k) = \frac{1}{(2\pi)|\Sigma_k|^{1/2}} exp[-\frac{1}{2}(x_i - \mu_k)^t \Sigma_k^{-1}(x_i - \mu_k)]$$

- The Bayes rule is used to decide whether pixel x_i, ($i = 1, 2, \ldots, n$) belongs to O:

$$P(O/x_i) = \frac{p(x_i/O)P(O)}{p(x_i)}$$

- **Using the EM algorithm to estimate the parameters of the mixture**

Expectation step (using paper's notation):

$$E(z_{ik}) = \frac{p(x_i/\theta_k^t)\pi_k^t}{\sum_{j=1}^{K} p(x_i/\theta_j^t)\pi_j^t} \quad \text{or} \quad P(k/x_i) = \frac{p(x_i/\theta_k^t)\pi_k^t}{p(x_i/O)}$$

Maximization step (using paper's notation):

$$\pi_k^{t+1} = \frac{1}{n}\sum_{x_i \in O} E(z_{ik}) \qquad \text{or} \qquad \pi_k^{t+1} = \frac{\psi_k^t}{n} \qquad \text{(i.e.,}$$

$$\psi_k^t = \sum_{x_i \in O} E(z_{ik}) = \sum_{x_i \in O} P(k/x_i))$$

$$\mu_k^{t+1} = \frac{1}{n\pi_k^{t+1}} \sum_{x_i \in O} E(z_{ik})x_i \quad \text{or} \quad \mu_k^{t+1} = \frac{\sum_{x_i \in O} P(k/x_i)x_i}{\psi_k^t}$$

$$\Sigma_k^{t+1} = \frac{1}{n\pi_k^{t+1}} \sum_{x_i \in O} E(z_{ik})(x_i - \mu_k^{t+1})(x_i - \mu_k^{t+1})^T \quad \text{or}$$

$$\Sigma_k^{t+1} = \frac{\sum_{x_i \in O} P(k/x_i)(x_i - \mu_k^{t+1})(x_i - \mu_k^{t+1})^T}{\psi_k^t}$$

• Initialization of mixture's parameters

- The mixture depends on the number of components and the choice initial parameters.

  * A suitable value for K was chosen based upon visual inspection of the object's color distribution.

  * The component means were initialized to a random subset of the training data points.

  * The covariance matrices were initialized to σI where σ was the Euclidean distance from the component's mean to its nearest neighboring component's mean.

  * The priors were initialized to $\pi=1/K$

- An alternative scheme based on cross-validation is also described (see paper pp. 226, 2nd column - start with one component, add more components incrementally).

• Color representation system

- RGB values are converted to the HSI (Hue-Saturation-Intensity) system.

- Only the H and S components are used (I is discarded to obtain invariance to the intensity of ambient illumination).

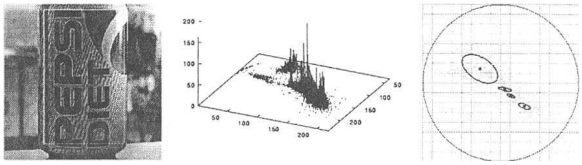
- Pixels corresponding to low S values were discarded (i.e., not reliable to measure H).

- Pixels with very high I value were also discarded.

• Adaptive color mixtures

- A non-adaptive model has given good results in the past under large rotations in depth, changes of scale, and partial occlusions.

- To deal with large changes in illumination conditions, an adaptive model is required.



- Let's denote the adaptive estimates of the parameters of the mixture components corresponding to frame $r-1$ as

$$[\pi_k(r-1), \mu_k(r-1), \Sigma_k(r-1)]$$

- In the next frame r, the parameters $[\pi_k(r), \mu_k(r), \Sigma_k(r)]$ will be adapted using a weighted sum of the following terms:

(1) adaptive estimates from frame $r-1$:

$$[\pi_k(r-1), \mu_k(r-1), \Sigma_k(r-1)]$$

(2) estimates using the data from frame r only:

$$[\pi_k^r, \mu_k^r, \Sigma_k^r]$$

(3) estimates using the data from frame $r-L-1$ only:

$$[\pi_k^{r-L-1}, \mu_k^{r-L-1}, \Sigma_k^{r-L-1}]$$ (L controls the adaptivity of the model)

Warning: superscripts r correspond to estimates using data from frame r only; indices r in parentheses correspond to adaptive estimates.

Compute the estimates using the data from frame r only

$$\pi_k^r = \frac{\psi_k^r}{n^r} \text{ where } \psi_k^r = \sum_{x_i \in O^r} P(k/x_i)$$

$$\mu_k^r = \frac{\sum_{x_i \in O^r} P(k/x_i)x_i}{\psi_k^r}$$

$$\Sigma_k^r = \frac{\sum_{x_i \in O^r} P(k/x_i)(x_i - \mu_k(r-1))(x_i - \mu_k(r-1))^T}{\psi_k^r}$$

Compute the adaptive parameters

* Will be estimated using the $L+1$ most recent frames (L controls the adaptivity of the model).

$$\mu_k(r) = \frac{\sum_{\tau=r-L}^{r} \sum_{x_i \in O^r} P(k/x_i)x_i}{\sum_{\tau=r-L}^{r} \psi_k^\tau}$$

$$\Sigma_k(r) = \frac{\sum_{\tau=r-L}^{r} \sum_{x_i \in O^r} P(k/x_i)(x_i - \mu_k(r-1))(x_i - \mu_k(r-1))^T}{\sum_{\tau=r-L}^{r} \psi_k^\tau}$$

* The above expressions can be written in the following form (see Appendix A):

$$\mu_k(r) = \mu_k(r-1) + \frac{\psi_k^r}{D_k(r)}(\mu_k^r - \mu_k(r-1)) - \frac{\psi_k^{r-L-1}}{D_k(r)}(\mu_k^{r-L-1} - \mu_k(r-1))$$

$$\Sigma_k(r) = \Sigma_k(r-1) + \frac{\psi_k^r}{D_k(r)}(\Sigma_k^r - \Sigma_k(r-1)) - \frac{\psi_k^{r-L-1}}{D_k(r)}(\Sigma_k^{r-L-1} - \Sigma_k(r-1))$$

where $D_k(r) = \sum_{\tau=r-L}^{r} \psi_k^\tau = D_k(r-1) + (\psi_k^r - \psi_k^{r-L-1})$

- Similarly, $\pi_k(r)$ is estimated as follows:

$$\pi_k(r) = \pi_k(r-1) + \frac{n^r}{\sum_{\tau=r-L}^{r} n^\tau}(\pi_k^r - \pi_k(r-1)) - \frac{n^{r-L-1}}{\sum_{\tau=r-L}^{r} n^\tau}(\pi_k^{r-L-1} - \pi_k(r-1))$$

- **Selecting** O^r

- Given a color mixture model for an object, the object is effectively located and tracked in the scene by computing a probability map for the pixels in the image within a search window.

- The size and position of the object are then estimated from the resulted distribution.

- The tracker estimates the centroid, height and width of the object.

- O^r is chosen to be a region of appropriate aspect ratio centered on the estimated object centroid.

- **Selective adaptation**

  - Any tracker can loose the object being tracked due, e.g., occlusion.

  - If such errors go undetected, the color model will adapt to image regions which do not correspond to the object being tracked.

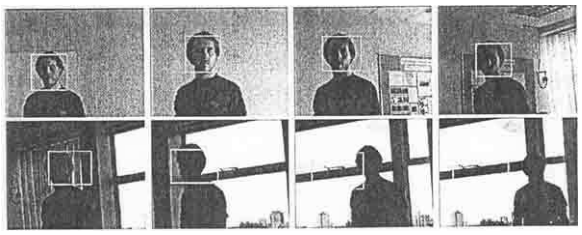
  - Keep track of the normalized log-likelihood L^r of the data O^r at each frame (the mixture model seeks to maximize L^r):

$$L^r = \frac{1}{n^r} \sum_{x \in O^r} \log\ p(x_i/O)$$

  - If the tracker loses the object, there is often a sudden, large drop in the value of L^r.

  - Adaptation is suspended until the object is again tracked with suffi ciently high likelihood (see paper for guidelines of how to choose a good likelihood thresh-old).

- **Experiments**



(non-adaptive model)

(adaptive model)

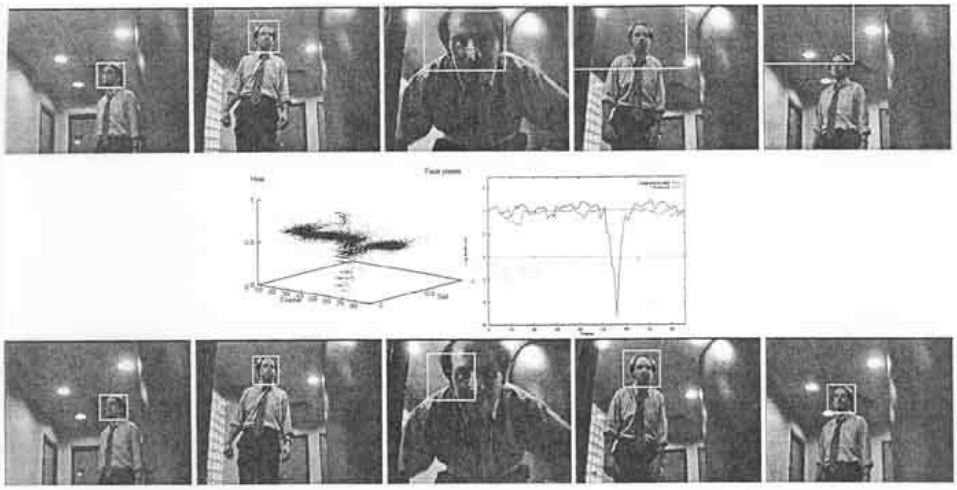
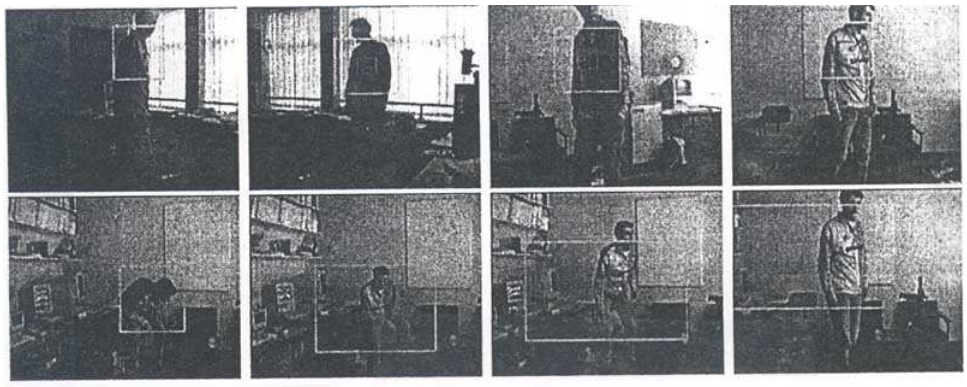
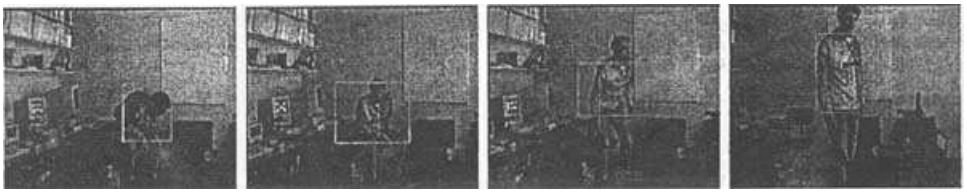


Figure 3. At the top are frames 35, 45, 55, 65 and 75 from a sequence with strong directional and exterior illumination. The walls have a fleshy tone. At around frame 55, the subject rapidly approaches the camera situated in a doorway, resulting in rapid changes in illumination, scale and auto-iris parameters. This can be seen in the 3D plot of the hue-saturation distribution over time. In the top sequence, the model was allowed to adapt in every frame, resulting in failure at around frame 60. The lower sequence illustrates the use of selective adaptation. The right-hand plot shows the normalised log-likelihood measurements and the adaptation threshold .





- **Interesting ideas for future research**

  - Use several cues, especially when color becomes unreliable.

  - Adaptive modeling of background scene colors:

$$P(O/x_i) = \frac{p(x_i/O)P(O)}{p(x_i/O)P(O) + p(x_i/B)P(B)}$$

  - Adaptive number of mixture components.

Adaptive background mixture models for real-time tracking

(C. Stauffer and E. Grimson, "Adaptive background mixture models for real-time tracking", *IEEE Computer Vision and Pattern Recognition Conference*, Vol. 2, pp. 246-252, 1998 (*on-line*))

• The problem

- Real-time segmentation of moving objects in image sequences.

- A robust moving object detection system should be able to handle:

(1) variations in lighting
(2) moving scene clutter
(3) multiple moving objects
(4) other arbitrary changes in the scene

• Traditional methods for background modeling

- The oldest technique for moving object detection is based on background subtraction:

(1) Subtract a model of the background from the current frame.

(2) Threshold the difference image

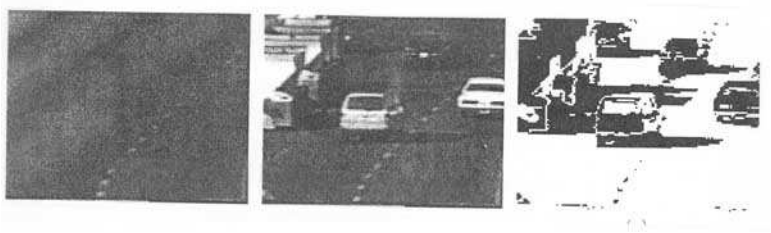
- We can assume a static or a time-varying background and a fixed or moving camera (fixed camera, varying background is assumed here).

- Non-adaptive background models have serious limitations and are rarely used (e.g., no significant changes are allowed in the scene except the moving objects).

- A standard method of adaptive backgrounding is based on averaging the images over time (i.e., approximates the current static scene except where motion occurs).

(1) Objects must move continuously.
(2) The background must be visible most of the time.
(3) Not robust when the scene contains multiple, slowly moving objects.

- Cannot distinguish shadows from moving objects.



• The approach

- Model the values of a particular pixel as a mixture of Gaussians.

- Evaluate the Gaussian distributions to determine which are most likely to result from a background process.

- Classify each pixel based on whether the Gaussian distribution which represents it most effectively is part of the background model.

- Adapt the model parameters over time to deal with:

(1) lighting changes
(2) repetitive motions of scene elements (e.g., swaying trees)
(3) slow-moving objects
(4) introducing or removing objects from the scene

- **Why modeling each pixel using a mixture?**

  - If each pixel resulted from a particular surface under particular lighting, a single Gaussian would be sufficient.

  - If only lighting changed over time, a single, adaptive Gaussian per pixel would be sufficient.

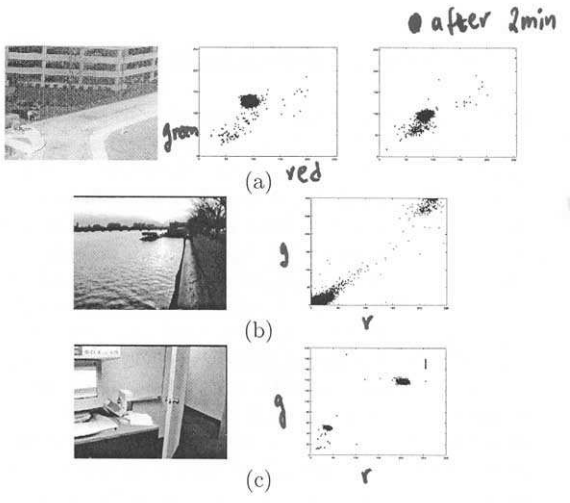
  - To model multiple surfaces and varying lighting conditions, a mixture of adaptive Gaussians are necessary.

- **Modeling pixel values using mixtures**

  - Consider the values of a particular pixel over time as a "pixel process" (i.e., time series).

  - At any time t, we know the history or each pixel at location (x, y):

$$\{X_1, X_2, ..., X_t\}=\{I(x, y, i), i = 1, 2, .., t\}$$



  - The recent history of each pixel, $\{X_1, X_2, ..., X_t\}$, is modeled by a mixture of K Gaussians.

- The probability of observing the current pixel value is

$$P(X_t) = \sum_{i=1}^{K} p(X_t/\mu_{i,t}\Sigma_{i,t})\pi_{i,t}$$

where $K = 2 - 5$ and $p()$ is given by

$$p(X_t/\mu_{i,t}\Sigma_{i,t}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{i,t}|^{1/2}} exp[-\frac{1}{2}(X_t - \mu_{i,t})^T \Sigma_{i,t}^{-1}(X_t - \mu_{i,t})]$$

- To simplify the calculations, $\Sigma_{i,t}$ is assumed to be diagonal:

$$\Sigma_{i,t} = \sigma_i^2 I$$

• Methodology

- Each Gaussian is evaluated (i.e., using its persistence and variance) to hypothesize which are most likely to be part of the "background process".

- Pixel values that do not fi t the background Gaussians are considered foreground and grouped using connected components.

- The connected components are tracked from frame to frame using a multiple hypothesis tracker.

- **Estimating/Updating the parameters of the model**

  - Each new observation is integrated into the model using standard learning rules (using the EM algorithm for every pixel would be costly).

  - Every pixel value, X_t, is checked against the existing K Gaussian distributions to find the one that represents it most.

  - A match is defined as a pixel value within 2.5σ of a distribution (.e., each pixel has essentially its own threshold).

  - If a match is found, the parameters of the mixture model as updated as follows:

$$\pi_{i,t} = (1 - \alpha)\pi_{i,t-1} + \alpha M_{i,t}$$

  where α is a learning constant and $M_{i,t}$ is 1 for the model which matched and 0 for the other models (re-normalize $\pi_{i,t}$ such that $\sum_{i=1}^{K} \pi_{i,t} = 1$)

  (only the parameters of the matched Gaussian i are updated below)

$$\mu_{i,t} = (1 - \rho)\mu_{i,t-1} + \rho X_t$$

$$\sigma_{i,t}^2 = (1 - \rho)\sigma_{i,t-1}^2 + \rho(X_t - \mu_{i,t})^T(X_t - \mu_{i,t})$$

  where $\rho = \alpha\, p(X_t/\mu_i, \sigma_i)$

  - If a match is not found, the least probable distribution is replaced with a distribution with the current pixel value as its mean value, an initial high variance, and a low prior weight.

- **Determining the background Gaussians**

- Determine which of the Gaussians of the mixture are most likely produce by background processes.

- The following observations are made:

(1) Moving objects are expected to produce more variance than a "static" (background) object.

(2) There should be more data supporting the background distributions because they are repeated, whereas pixel values from different objects are often not the same color.

- The following heuristic is used to determine the "background" Gaussians:

"choose the Gaussians which have most supporting evidence and the least variance"

- To implement this idea, the Gaussians are ordered by the value of π/σ

- The first B distributions are chosen as the background model, where

$$B = argmin_b(\sum_{i=1}^{b} \pi_i > T)$$

where T is a measure of the minimum portion of the data that should be accounted for by the background.

- **Connected components**

  - The mixture model allows the identifi cation of foreground pixels in each new frame.

  - These pixels are grouped into different regions using connected components.

- **Multiple hypothesis tracking**

  - Connected components are tracked from frame to frame.

  - A pool of Kalman fi lters are used to track the connected components (see paper for more details).

- **Experiments and results**

- The system has been run almost continuously for 16 months (24 hrs/day through rain and snow).

- The system can process 11-13 frames per second (160 x 120 pixels).

(see http://www.ai.mit.edu/projects/vsam)