Face Recognition Using a Line Edge Map

Yongsheng Gao and *Maylor K.H. Leung* at
Nanyang Tech. University

IEEE Pattern Analysis and Machine Intelligence 2002

Slides by David Anthony Torres
Computer Science and Engineering — University of California at San Diego

Interest Points Vs. Edge Maps

- Interest point detectors are popular
  - SIFT, Harris/Forstner
- What about edge information?
  - Can carry distinguishing info too.
  - Interest points don't capture this info

Line Edge Map

- Humans recognize line drawings well.
  - ➢ Maybe computer algorithms can too.
- Benefits of using edge information:
  - ➢ Advantages of template matching and geometrical feature matching:
    - Partially illumination-invariant
    - Low memory requirement
    - Recognition performance of template matching

Line Edge Map

- Takács (1998) used edge maps for face recognition.
  - Apply edge-detector to get a binary input image I
  - I is a set of edge points.
  - Use Hausdorff distance to measure the similarity between two sets of points I_1 and I_2.

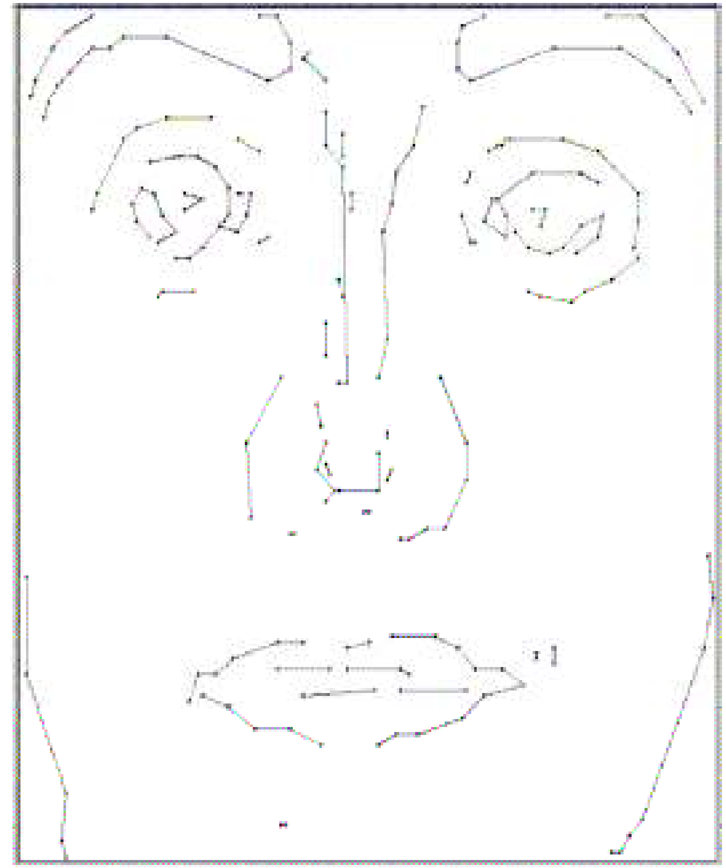
Hausdorff Distance

$$h(I_1, I_2) = \frac{1}{|I_1|} \sum_{i \in I_1} \min_{j \in I_2} \| i - j \|$$

- *i* and *j* are edge pixel positions (x,y).
- For each pixel *i* in I_1
  Find the closest corresponding pixel *j* in I_2
  Take the average of all these distances $\|i-j\|$.
- Calculated without explicitly pairing the sets of points.
- Achieved a 92% accuracy in their experiments.

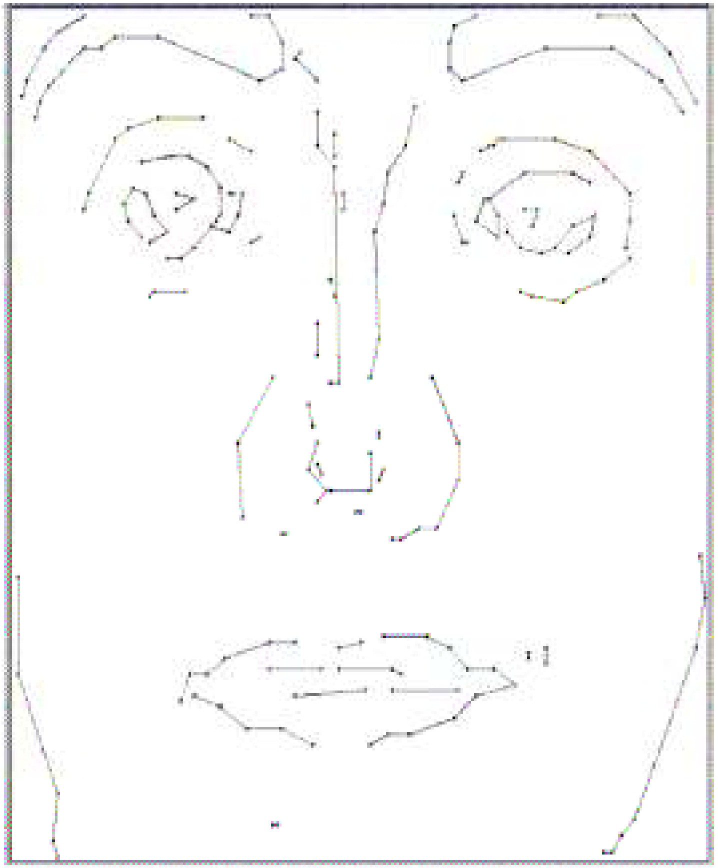
Line Edge Map

- Takács Edge Map doesn't consider local structure.
- Authors introduce the Line Edge Map (LEM)
- Groups edge pixels into line segments.
  - Apply polygonal line fitting to a thinned edge map



Line Edge Map

- LEM is a series of line segments.
  - ➢ LEM records only the endpoints of lines.
  - ➢ Further reduces storage requirements.



Line-Segment Hausdorff Distance (LHD)

- Need a new distance measure between sets of line segments.
- Expect it to be better because it uses line-orientation.
- First we'll see an initial model…
- Add to the model to make it more robust
  - Encourage one-one mapping of lines
  - Encourage mapping of "similar" lines.

Line-Segment Hausdorff Distance

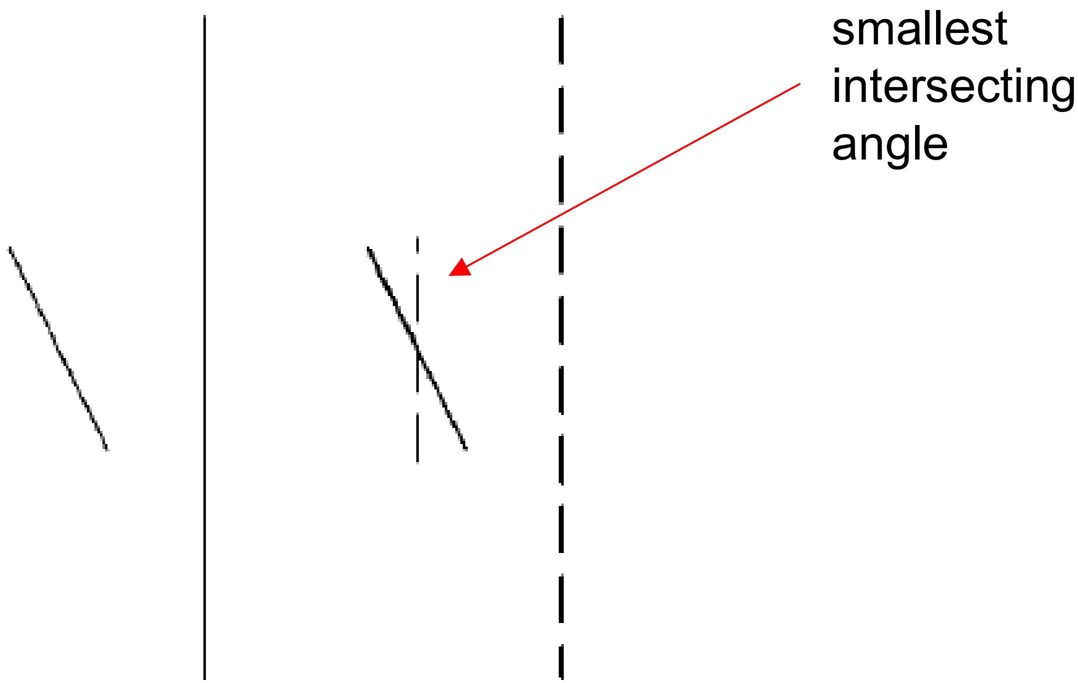
- Given two LEMs $S=(s_1,s_2,\ldots s_p)$ and $T=(t_1,t_2,\ldots s_q)$
- The LHD is built on the vector $d(s_i,t_j)$
  - ➢ *d()* represents the distance between two lines segments

$$\vec{d}(m_i^l, t_j^l) = \begin{bmatrix} d_\theta\left(m_i^l, t_j^l\right) \\ d_{//}\left(m_i^l, t_j^l\right) \\ d_\perp\left(m_i^l, t_j^l\right) \end{bmatrix}$$

Line-Segment Hausdorff Distance

$$\vec{d}(m_i^l, t_j^l) = \begin{bmatrix} d_\theta\left(m_i^l, t_j^l\right) \\ d_{//}\left(m_i^l, t_j^l\right) \\ d_\perp\left(m_i^l, t_j^l\right) \end{bmatrix}$$

$$d_\theta\left(m_i^l, t_j^l\right)\Big|$$



(a) (b)

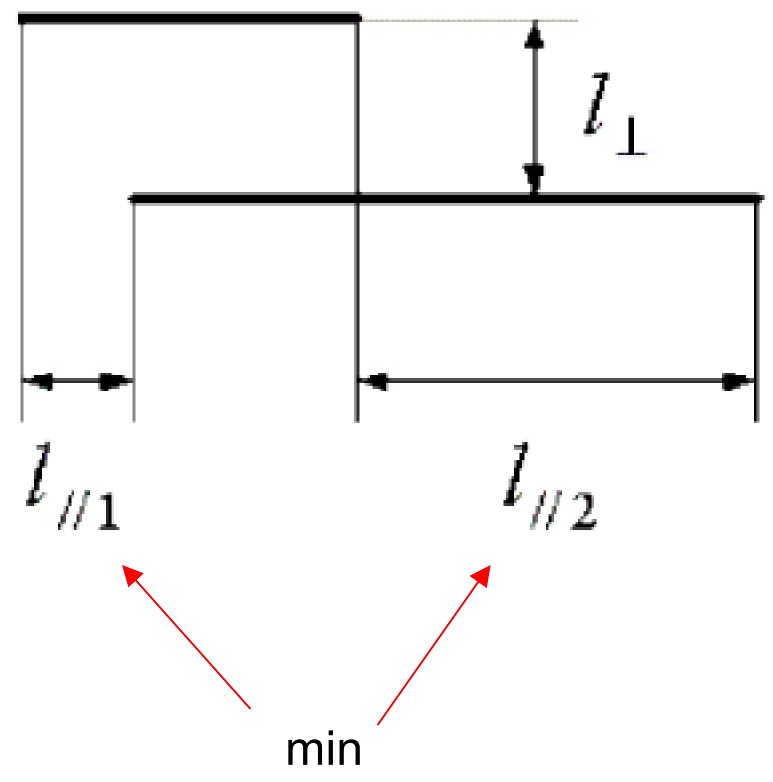
Line-Segment Hausdorff Distance

$$d_\theta\left(m_i^l, t_j^l\right) = f\left(\theta\left(m_i^l, t_j^l\right)\right)$$

- f() is a penalty function: $f(\theta) = \theta^2/W$
  - ➢Higher penalty on large deviation
- W is determined in training.

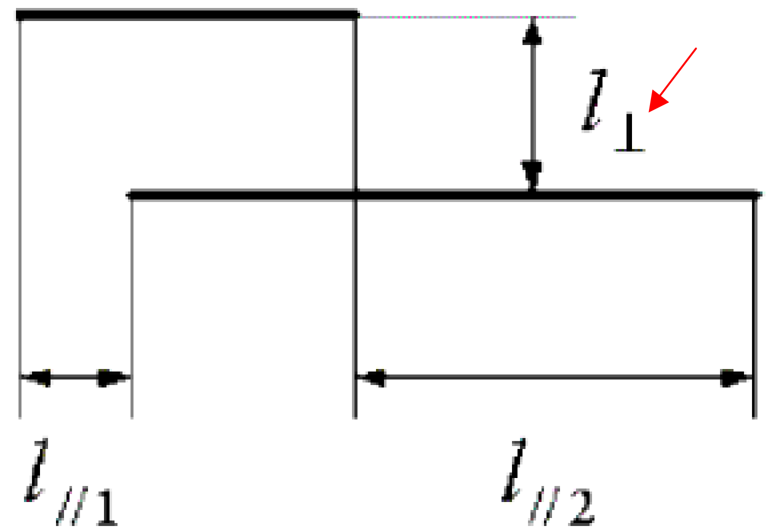
Line-Segment Hausdorff Distance

$$d_{//}\left(m_i^l, t_j^l\right) = \min(l_{//1}, l_{//2})$$



Line-Segment Hausdorff Distance

$$d_{\perp}\left(m_i^l, t_j^l\right) = l_{\perp}$$



- In general lines will not be parallel
- So rotate the shortest line

Line-Segment Hausdorff Distance

- Finally,

$$d\left(m_i^l, t_j^l\right) = \sqrt{d_\theta^2\left(m_i^l, t_j^l\right) + d_{//}^2\left(m_i^l, t_j^l\right) + d_\perp^2\left(m_i^l, t_j^l\right)}$$

- Primary line-segment Hausdorff Distance (LHD)

$$H(I,J) = \max(h(I,J), h(J,I))$$

where

$$h(I,J) = \frac{1}{\sum_{i \in I} \| i \|} \sum_{i \in I} \| i \| \cdot \min_{j \in J} d(i,j)$$

Some Problems…

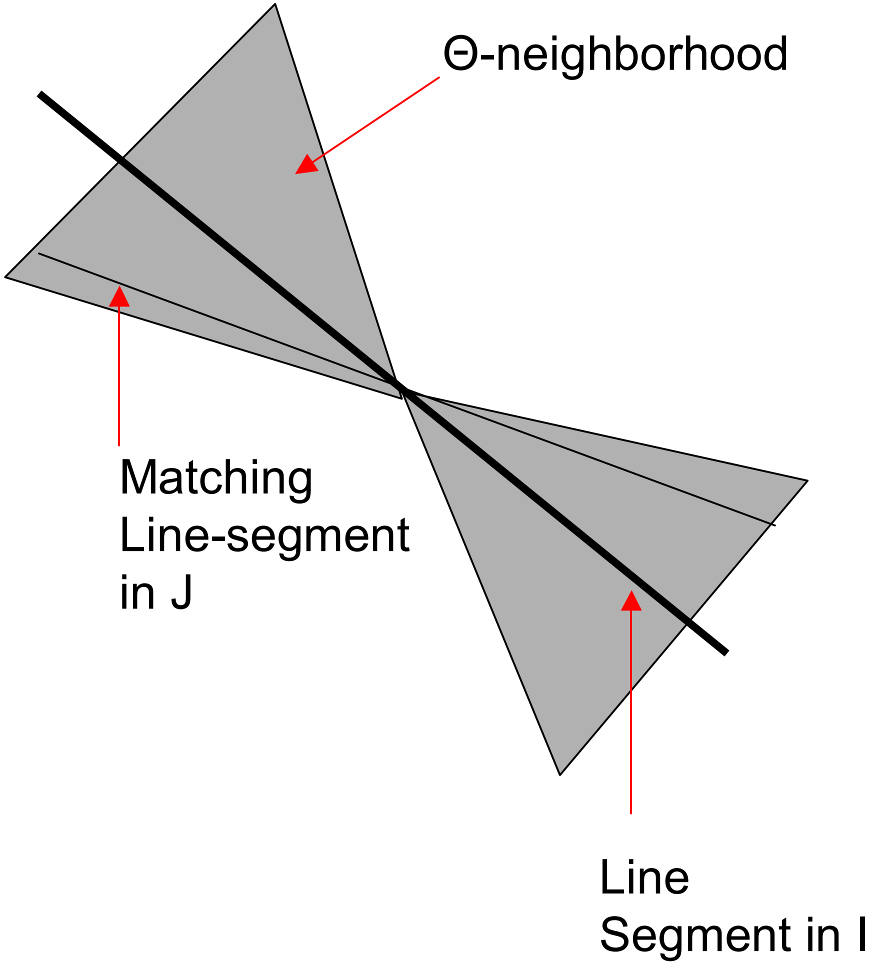
- Say T is an input LEM, M is its matching model LEM, and N is some other non-matching model.
- Due to segmentation problems it could be the case that

$$H(T,M) >> H(T,N)$$

- Keeping track of matched line-pairs could help.

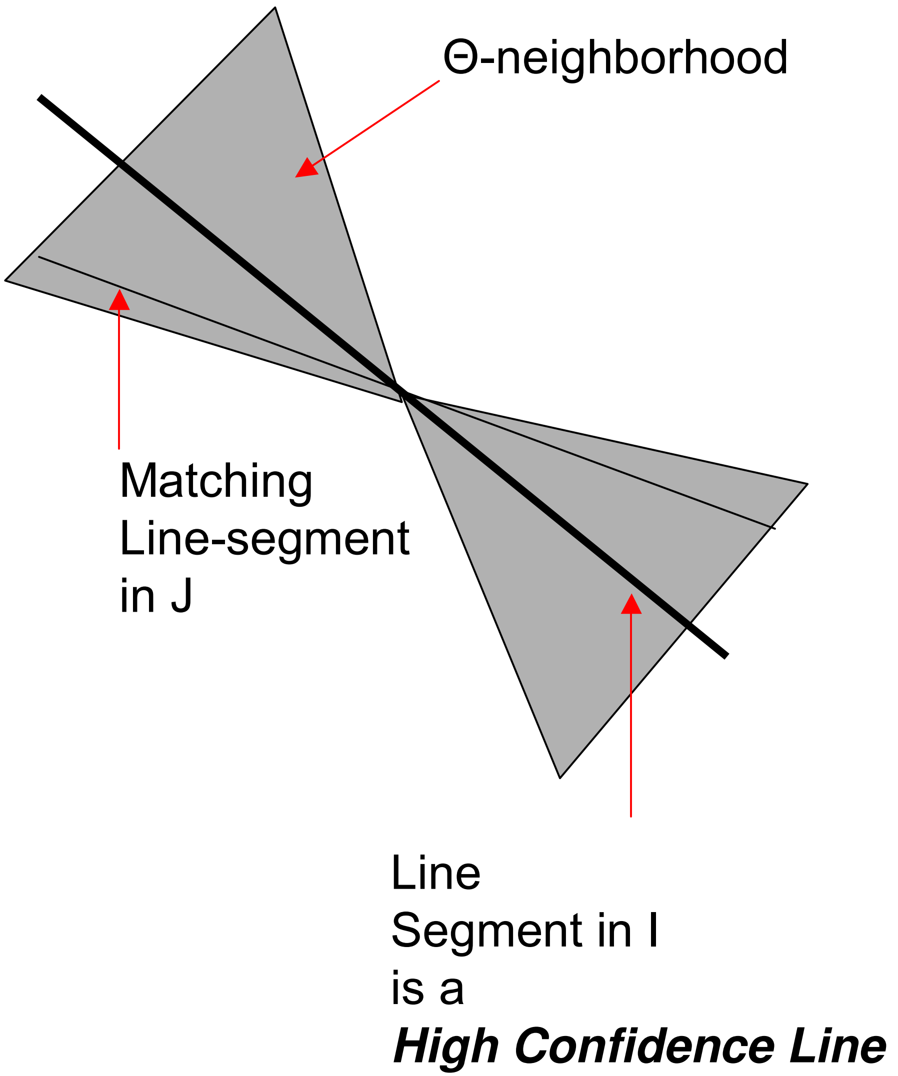
Neighborhoods

- Positional neighborhood N_p
- Angular neighborhood N_a
- Heuristic: lines that fall within the neighborhood are probably matches.



Neighborhoods

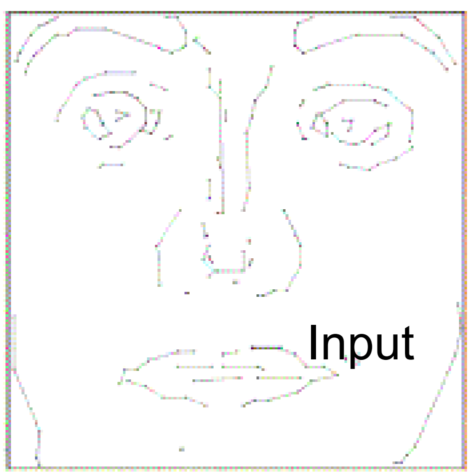
- If $\geq$1 line falls into the neighborhoods we call the original line segment *I,* a *high confidence line*.

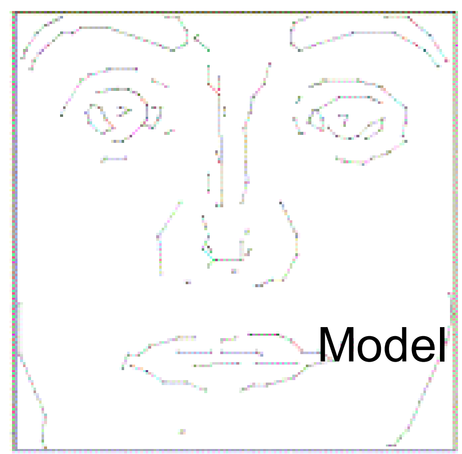


High Confidence Ratio

- N_{hc} is the num. of high confidence lines in a LEM.
- N_{total} is the total num. of lines in a LEM.

$$R = \frac{N_{hc}}{N_{total}}$$





New Hausdorff Distance

$$H'(T,M) = \sqrt{H^2(T,M) + (W_n D_n)^2}$$

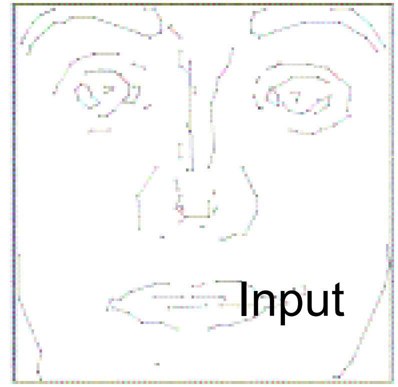
- W_n is a weight.
- D_n is the average number of lines (across input and model) that are not confidently-matched, i.e.

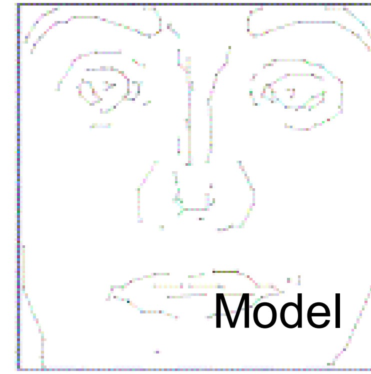
$$D_n = 1 - \frac{R_M + R_T}{2} = \frac{(1 - R_M) + (1 - R_T)}{2}$$

R_T and R_M are the high confidence ratios for input and model respectively

Summary

- Start with to LEM's





- Calculate Hausdorff Distance

$$H(I,J) = \max(h(I,J), h(J,I))$$

$$h(I,J) = \frac{1}{\sum_{i \in I} \| i \|} \sum_{i \in I} \| i \| \cdot \min_{j \in J} d(i,j)$$

Summary

- $$d\left(m_i^l, t_j^l\right) = \sqrt{d_\theta^2\left(m_i^l, t_j^l\right) + d_{//}^2\left(m_i^l, t_j^l\right) + d_\perp^2\left(m_i^l, t_j^l\right)}$$

- $$\vec{d}(m_i^l, t_j^l) = \begin{bmatrix} d_\theta\left(m_i^l, t_j^l\right) \\ d_{//}\left(m_i^l, t_j^l\right) \\ d_\perp\left(m_i^l, t_j^l\right) \end{bmatrix}$$

Summary

- Finally we take into account the effect of neighborhoods

$$H'(T,M)=\sqrt{H^2(T,M)+(W_nD_n)^2}$$

- $D_n = 1 - \frac{R_M + R_T}{2} = \frac{(1-R_M)+(1-R_T)}{2}$

Free Parameters

- We have four free parameters to fix
  - (W, W_n, N_p, N_a)
    - $\theta^2/W = f(\theta) = d_\theta$
    - $H'(T,M) = \sqrt{H^2(T,M) + (W_n D_n)^2}$
    - Neighborhoods N_p, N_a

- Use simulated annealing to estimate
  - With probability $p = e^{-\frac{\Delta Err}{t}}$

Results

Face Recognition under Controlled Conditions

Bern Database

AR Database

Face Recognition under Controlled Conditions

TABLE 1
Face Recognition Results of Edge Map (EM) [2], Eigenface (20-Eigenvectors), and LEM

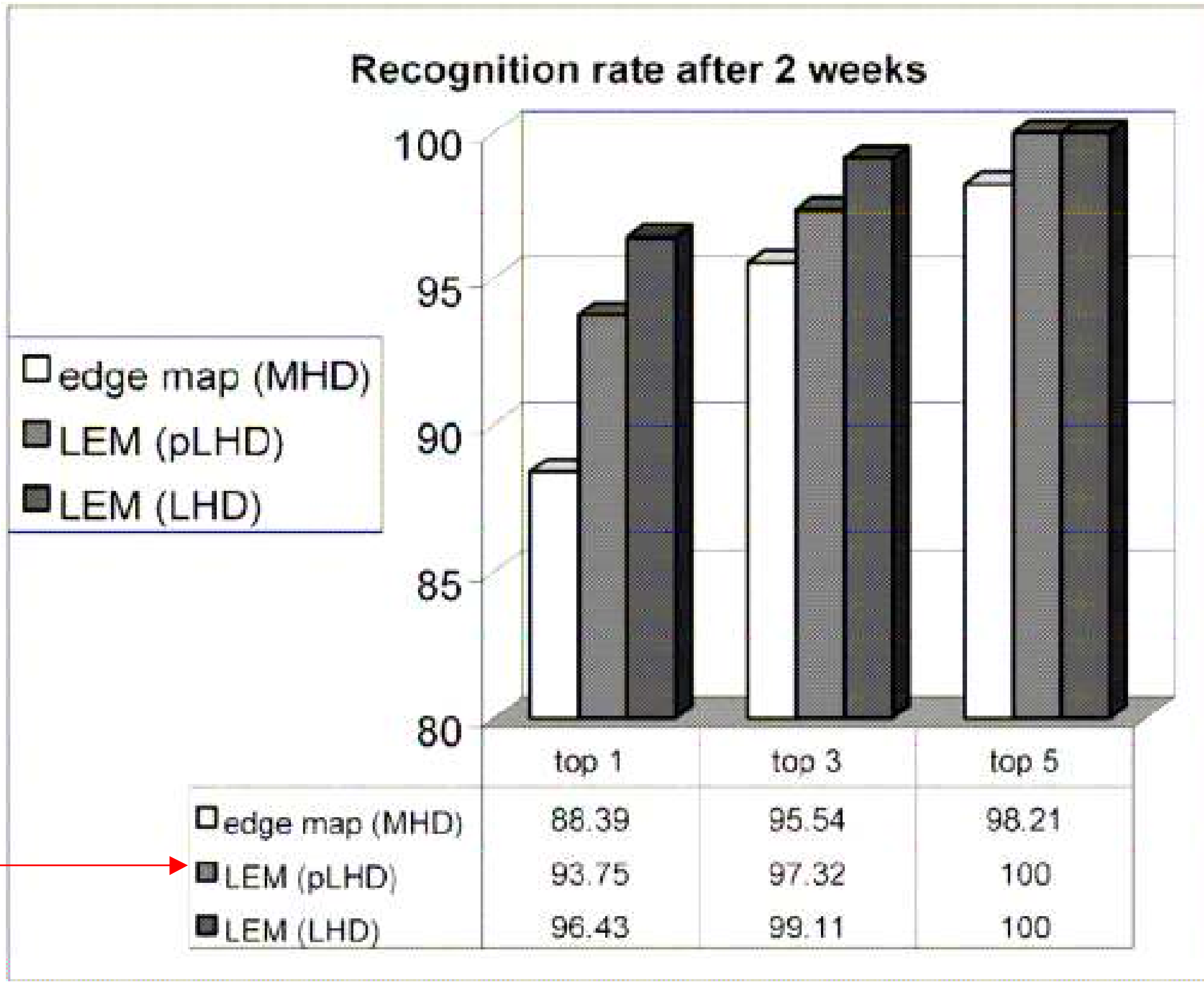
	Bern database			AR database		
Method	EM	Eigenface	LEM	EM	Eigenface	LEM
Recognition rate	96.7%	100%	100%	88.4%	55.4%	96.4%

Face Recognition under Controlled Conditions

TABLE 2
Performance Comparison on the AR Database

Method	Recognition rate
LEM	96.43%
Eigenface (20-eigenvectors)	55.36%
Eigenface (60-eigenvectors)	71.43%
Eigenface (112-eigenvectors)	78.57%

Face Recognition under Controlled Conditions



w/o neighborhood heuristic

Sensitivity to Size Variation

TABLE 3
Recognition Results with Size Variations

	Top 1	Top 5	Top 10
Edge map	43.3%	56.0%	64.7%
Eigenface (112-eigenvectors)	44.9%	68.8%	75.9%
LEM (pLHD)	53.8%	67.6%	71.9%
LEM (LHD)	66.5%	75.9%	79.7%

- Used the AR data base.
- Applied a random scaling factor of ±10%

Recognition Under Varying Lighting

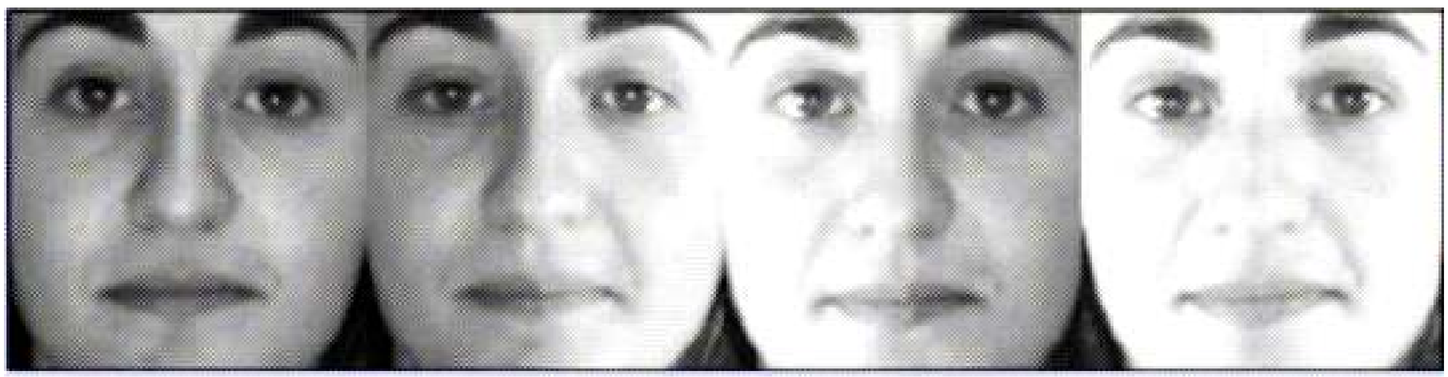


TABLE 4
Recognition Results under Varying Lighting

Testing faces	Eigenface		Edge map	LEM
Left light on	20-eigenvectors	6.25%	82.14%	92.86%
	60-eigenvectors	9.82%		
	112-eigenvectors	9.82%		
	112-eigenvectors w/o 1st 3	26.79%		
Right light on	20-eigenvectors	4.46%	73.21%	91.07%
	60-eigenvectors	7.14%		
	112-eigenvectors	7.14%		
	112-eigenvectors w/o 1st 3	49.11%		
Both lights on	20-eigenvectors	1.79%	54.46%	74.11%
	60-eigenvectors	2.68%		
	112-eigenvectors	2.68%		
	112-eigenvectors w/o 1st 3	64.29%		

Recognition Under Facial Expression Changes

TABLE 5
Recognition Results under Different Facial Expressions

Testing faces	Eigenface		EM	LEM
Smiling expression	20-eigenvectors	87.85%	52.68%	78.57%
	60-eigenvectors	94.64%		
	112-eigenvectors	93.97%		
	112-eigenvectors w/o 1st 3	82.04%		
Angry expression	20-eigenvectors	78.57%	81.25%	92.86%
	60-eigenvectors	84.82%		
	112-eigenvectors	87.50%		
	112-eigenvectors w/o 1st 3	73.21%		
Screaming expression	20-eigenvectors	34.82%	20.54%	31.25%
	60-eigenvectors	41.96%		
	112-eigenvectors	45.54%		
	112-eigenvectors w/o 1st 3	32.14%		

View Based Identification — "Leave One Out" Experiment.

TABLE 6
"Leave-One-Out" Test of Yale Face Database

Method	Error Rate
Edge map	**26.06%**
Eigenface*	24.4%
Correlation*	23.9%
Linear Subspace*	21.6%
Eigenface w/o 1st 3*	15.3%
LEM	**14.55%**
Fisherface*	7.3%

Recognition Under Varying Pose

TABLE 7
Face Recognition Results under Pose Different Variations

	Recognition rate			
Method	Edge map	Eigenface (20-eigenvectors)	Eigenface (30-eigenvectors)	LEM
Looks left/right	50.00%	70.00%	75.00%	74.17%
Looks up	65.00%	51.67%	56.67%	70.00%
Looks down	67.67%	45.00%	55.00%	70.00%
Average	**58.17%**	**59.17%**	**65.12%**	**72.09%**

Additional Material…

Matching Time for LEM

- LEM takes longer than eigenface
  - Time $O(Nn) > O(Nm)$
    - N is # of faces
    - n is avg. # LEM-features
    - m is # eigenvectors
- Authors propose a face pre-filtering scheme
  - Idea: filter out faces before performing matching.

Face Prefiltering

- Quantize an LEM into : $\vec{S} = \begin{bmatrix} \Gamma \\ \Theta \end{bmatrix}$
- Where Γ is the sum of line segment lengths
- $$\Theta = \frac{1}{\sum_i l_i} \sum_i \vartheta_i l_i$$

where υ is the angle if the angle is <90 degrees.

Face Pre-filtering

$$\Delta\overrightarrow{S} \sim N_2\left(\overrightarrow{\mu}, \overrightarrow{\Sigma}\right),$$

where

$$\overrightarrow{\mu} = \begin{bmatrix} \mu_I \\ \mu_\theta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \overrightarrow{\Sigma} = \begin{bmatrix} \sigma_I^2 & \sigma_{I\theta} \\ \sigma_{\theta I} & \sigma_\theta^2 \end{bmatrix} = \begin{bmatrix} \sigma_I^2 & \sigma_I \sigma_\theta \rho \\ \sigma_\theta \sigma_I \rho & \sigma_\theta^2 \end{bmatrix}.$$

and the correlation coefficient

$$\rho = \frac{\sigma_{I\theta}}{\sigma_I \sigma_\theta}.$$

Face Pre-filtering

Then, the density function of the error vector can be represented as

$$f(\Delta\vec{S}) = \frac{1}{2\pi|\vec{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}\left(\Delta\vec{S} - \vec{\mu}\right)^T \vec{\Sigma}^{-1}\left(\Delta\vec{S} - \vec{\mu}\right)\right\}, \Delta\vec{S} \in \Re^2.$$

Face Pre-filtering

Since $|\vec{\Sigma}| = \sigma_l^2\sigma_\theta^2(1-\rho^2)$, the inverse of $\vec{\Sigma}$ exists if and only if $|\rho| < 1$. Straightforward calculation shows that

$$\vec{\Sigma}^{-1} = \frac{1}{\sigma_l^2\sigma_\theta^2(1-\rho^2)} \begin{bmatrix} \sigma_l^2 & -\sigma_l\sigma_\theta\rho \\ -\sigma_\theta\sigma_l\rho & \sigma_\theta^2 \end{bmatrix}. \qquad (15)$$

Face Pre-filtering

Thus, the density function of $\Delta \overline{S}$ becomes

$$
\begin{aligned}
& f(\Delta \vec{S}) = \\
& \quad \frac{1}{2\pi\sigma_l\sigma_\theta\sqrt{1-\rho^2}} \exp\left\{ -\frac{1}{2(1-\rho^2)} \left[\left(\frac{\Delta\Gamma - \mu_l}{\sigma_l} \right)^2 \right.\right. \\
& \quad \left.\left. - 2\rho \left(\frac{\Delta\Gamma - \mu_l}{\sigma_l} \right) \left(\frac{\Delta\Theta - \mu_\theta}{\sigma_\theta} \right) + \left(\frac{\Delta\Theta - \mu_\theta}{\sigma_\theta} \right)^2 \right] \right\}
\end{aligned}
$$

Face Pre-filtering

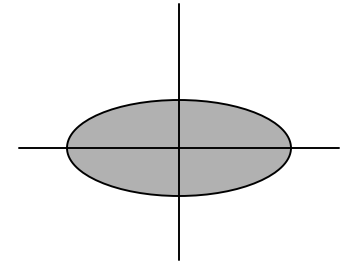
The constant density contours for a bivariate normal are a series of ellipses with different values of d as shown in the following equation:

$$\left(\Delta\vec{S}-\vec{\mu}\right)^{T}\vec{\Sigma}^{-1}\left(\Delta\vec{S}-\vec{\mu}\right)=d^{2}$$

or

$$\begin{aligned}\left(\frac{\Delta\Gamma-\mu_{l}}{\sigma_{l}}\right)^{2}-2\rho\left(\frac{\Delta\Gamma-\mu_{l}}{\sigma_{l}}\right)\left(\frac{\Delta\Theta-\mu_{\theta}}{\sigma_{\theta}}\right)\\+\left(\frac{\Delta\Theta-\mu_{\theta}}{\sigma_{\theta}}\right)^{2}=d^{2}\left(1-\rho^{2}\right).\end{aligned} \tag{17}$$

Face Pre-filtering

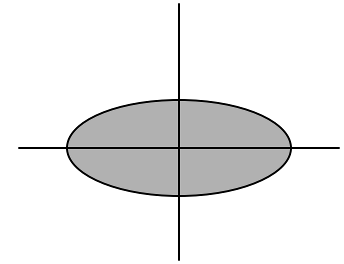


The probability that $\Delta\vec{S}$ falls in the elliptic region Ω of parameter d is given by

$$F(d) = \Pr\left(\Delta\vec{S} \in \Omega\right) = \iint_{\Omega} f\left(\Delta\vec{S}\right) d(\Delta\Gamma) d(\Delta\Theta)$$

$$= \iint_{\Omega} \frac{1}{2\pi\sigma_l\sigma_\theta\sqrt{1-\rho^2}} \exp\left\{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{\Delta\Gamma-\mu_l}{\sigma_l}\right)^2 - 2\rho\left(\frac{\Delta\Gamma-\mu_l}{\sigma_l}\right)\left(\frac{\Delta\Theta-\mu_\theta}{\sigma_\theta}\right) + \left(\frac{\Delta\Theta-\mu_\theta}{\sigma_\theta}\right)^2\right]\right\} d(\Delta\Gamma) d(\Delta\Theta).$$

Face Pre-filtering



Let

$$u = \frac{\Delta\Gamma - \mu_l}{\sigma_l}, \qquad v = \frac{\Delta\Theta - \mu_\theta}{\sigma_\theta}. \tag{19}$$

The equation of constant density contour can be rewritten as

$$u^2 - 2\rho uv + v^2 = d^2(1 - \rho^2). \tag{20}$$

Face -Prefiltering

$$F(d) =$$

$$\iint_{\Omega} \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left\{ -\frac{1}{2(1-\rho^2)} \left[u^2 - 2\rho uv + v^2 \right] \right\} du dv$$

Now for some hand-waving action...

- Rotate the Gaussian so that its axis aligned
- Perform a change of coordinates into a polar system

- $$F(d) = \int_0^d \int_0^{2\pi} \frac{1}{2\pi ab} \exp\left\{-\frac{1}{2}r^2\right\} |J| dr d\theta$$
$$= 1 - e^{-\frac{1}{2}d^2}.$$

- $$d = \sqrt{-2\ln[1 - F(d)]}$$

To summarize

- Given a probability F(d) we can obtain a constant density ellipse of the form:

$$\left(\frac{\Delta\Gamma - \mu_l}{\sigma_l}\right)^2 - 2\rho\left(\frac{\Delta\Gamma - \mu_l}{\sigma_l}\right)\left(\frac{\Delta\Theta - \mu_\theta}{\sigma_\theta}\right) + \left(\frac{\Delta\Theta - \mu_\theta}{\sigma_\theta}\right)^2 = d^2\left(1 - \rho^2\right)$$

- where

$$d = \sqrt{-2\ln[1 - F(d)]}$$

To summarize

- So if the error vector satisfies:

$$\left(\frac{\Delta\Gamma}{\sigma_l}\right)^2 - 2\rho\left(\frac{\Delta\Gamma}{\sigma_l}\right)\left(\frac{\Delta\Theta}{\sigma_\theta}\right) + \left(\frac{\Delta\Theta}{\sigma_\theta}\right)^2 < d^2\left(1-\rho^2\right)$$

- then the model is classified as a potential face.

Pre-Filtering Results

TABLE 11
AR Face Database Training Results

ρ	c_I	c_θ	μ_I	μ_θ
0.02	145.36	4.33	26.42	0.27

- Train to find parameter above.
- Small rho indicates vector components are nearly independent.

TABLE 12
Prefiltering Results on AR Face Database

F(d)	d^2	True acceptance rate	Filter out rate
90%	4.61	88.39%	50.31%
95%	5.99	92.86%	41.37%
99%	9.21	97.32%	26.58%
99.5%	10.60	99.11%	22.02%
99.7%	12.43	100%	17.06%

TABLE 13
Prefiltering Results on Bern University Face Database

$F(d)$	d^2	True acceptance rate	Filter out rate
90%	4.61	96.67%	61.55%
95%	5.99	96.67%	53.91%
96%	6.44	100%	51.95%