

Content Based Retrieval: Similarity and Classification

Dr. Michael S. Lew

Leiden Institute for Advanced Computer Science
Leiden University, Netherlands

Multimedia Information Retrieval - Dr. Michael Lew, LIACS, Leiden University

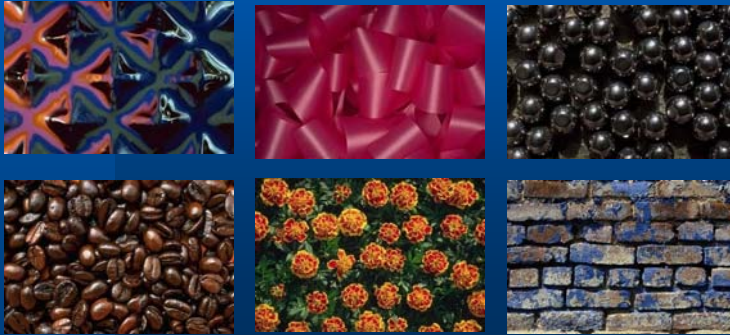
Similarity

- Which of these images are most similar?



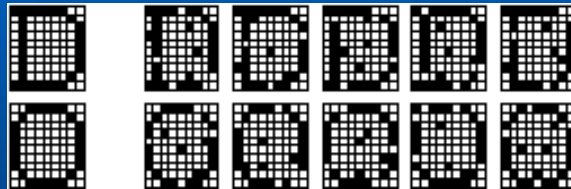
Similarity

- Which of these textures is most similar?



Similarity

- Which of these Shapes is most similar?



- Is the character below a D or an O?



Histograms - Regularization Properties

- Regularization properties of your space
 - some feature spaces are naturally smooth so you can use smoothing kernel functions to interpolate the space
 - i.e. Suppose we just have a “Blue” color histogram of 256 components. The “Blue” space is naturally smooth so we can interpolate on fewer examples
 - i.e. However, suppose our histogram is composed of 5 components: 1 red, 1 green, 1 blue, 1 texture, 1 shape. In this case, we could not interpolate the frequency of texture based on the frequency of blue.

Histograms - Discriminatory Properties

- Discriminatory properties of your space
 - Suppose we are in a “blue screen” studio and we want to detect all of the blue background so we can change the background using computer graphics
 - Assume we are using the RGB color space and we are limiting ourselves to 256 components or 8 bits.
 - Allocating 0 bits for red; 0 bits for green; and 8 bits for blue would be a possibility because in this case the color blue would be the main discriminator.
 - The components with greater discriminatory power typically receive more bits.

Histograms: Practical Notes

- For general image collections such as the Corel Stock Photography database it has been experimentally found that for 8 bits or 256 components:
- In the RGB color space, 3:3:2 is best (where 3:3:2 means 3 bits for R; 3 bits for G; 2 bits for B. Also, the number of bits, N , are used to uniformly divide the color space into 2^N regions)
- In the HSI color space, 4:2:2 is best
- However, there is no single correct answer - depends on your particular application

Improving Classes of Algorithms?

- Goal: How can we improve wide classes of computer vision algorithms?
 - Object Recognition
 - Stereo Matching
 - Texture Classification

Motivation

- Many computer vision algorithms can be viewed as
 - Features
 - Inter-Feature Distance or metric
- Examples
 - Color Indexing of Images and Video
 - Texture Classification
 - Stereo Matching
 - Motion Tracking
 - etc.
- Perhaps we can improve the distance measure

What is Distance? What is a Metric? (according to computational geometry)

| | |
|--|-----------------------|
| $P_1 : d(I, I) = d(J, J)$ | self-similarity |
| $P_2 : d(I, J) \geq d(I, I)$ | minimality |
| $P_3 : d(I, J) = d(J, I)$ | symmetry |
| $P_4 : d(I, K) + d(K, J) \geq d(I, J)$ | triangular inequality |

- Any function satisfying P_1 - P_4 is a “distance”
- Any function satisfying P_1 , P_2 , and P_4 is a “metric”
- Do we care if its a “distance” or “metric”?
 - Several studies have shown that P_1 and P_3 are NOT validated by “human perception”

How do you choose a feature metric?

- Maximum Likelihood Perspective:
- It is well known:
 - If the noise distribution is Gaussian, then the appropriate metric is L_2
- What if the noise distribution is not Gaussian?

Similarity Theory for Gaussian

- If the distributions are independent, we can multiply them to get the similarity probability which we are trying to maximize

$$P \propto \prod_{x=1}^N \left[e^{-\frac{1}{2} \left(\frac{F(x)}{\sigma} \right)^2} \right] (\Delta z),$$

- Multiplying exponentials reduces to summing the exponents

$$P \propto e^{\sum_{x=1}^N \left[-\frac{1}{2} \left(\frac{F(x)}{\sigma} \right)^2 \right]} (\Delta z)^N$$

- Taking the log of both sides gives

$$\log P \propto \sum_{x=1}^N \left[-\frac{1}{2} \left(\frac{F(x)}{\sigma} \right)^2 \right] + N \log(\Delta z)$$

Similarity Theory

- Maximizing $\log P$ is equivalent to minimizing the negative and so we arrive at

$$\frac{1}{2\sigma^2} \sum_{z=1}^N [F(z)]^2 + N \log(\Delta z)$$

- Since N , Δz , and σ are constants, the functional reduces to minimizing

$$\sum_{z=1}^N [F(z)]^2$$

- So, the Sum of Squared Differences (SSD) is justified when the difference distribution is Gaussian

How do you choose a feature metric?

- What if the difference distribution is not Gaussian?
- For the rest of the talk, we describe the distribution of the differences as the “noise”

Maximum Likelihood

If ρ is the negative logarithm of the “noise” between x and y then the similarity probability is $P(G)$ as shown as Equation 1:

$$P(G) = \prod_{i=1}^M \{\exp[-\rho(x_i, y_i)]\}$$

which is equivalent to minimizing Equation 2:

$$\sum_{i=1}^M \rho(n_i)$$

Linking Maximum Likelihood and Metrics

- Equation 2 is significant because it directly links the distribution (pdf) to a metric.
- If the pdf of $(x-y)$ is represented by a normalized histogram $H(x-y)$, then the metric which maximizes the similarity probability is:
 - metric = $\sum [-\ln H(x-y)]$
 - which we call the *maximum likelihood metric (M)*

Why Use Other Metrics?

- If the Maximum Likelihood metric is mathematically perfect then why should one use other metrics?
- Two assumptions were made
 - Basing the distribution on the difference $x[k] - y[k]$
 - Independence of the distributions
- In general, neither of these assumptions is generally valid

Other Feature Metrics

- Cauchy (L_C) $\rightarrow p(z) = \log(a^2 + z^2)$
 - “a” allows adjustment to the weight of the tails
- Known as the “robust” distribution because the influence of outliers (the tails) can be directly controlled.

Other Feature Metrics

- Kullback Relative Information
 - not symmetric
 - $J = \sum x[k] \log(x[k]/y[k])$
- Jeffrey Distance
 - based on the Kullback, but symmetric
 - does not perform well when one histogram is a shifted version of another.

$$d_J(H_0, H_1) = \sum_{i=1}^i \left[H_0(i) \log \left(\frac{H_0(i)}{m_i} \right) + H_1(i) \log \left(\frac{H_1(i)}{m_i} \right) \right]$$

Other Feature Metrics

- Bhattacharyya Distance (where functions are normalized to have area equal to 1)

$$d(H_1, H_2) = -\log \sum_{\alpha} \sqrt{H_1(\alpha) H_2(\alpha)}$$

- Signal to Noise Ratio (SNR)

$$d_{snr}(H_0, H_1) = 10 \log_{10} \left(\frac{Var[H_0]}{Var[H_0 - H_1]} \right)$$

Other Feature Metrics

- Normalized Correlation Coefficient
 - Noise resistant because it measures the shape of the histogram.

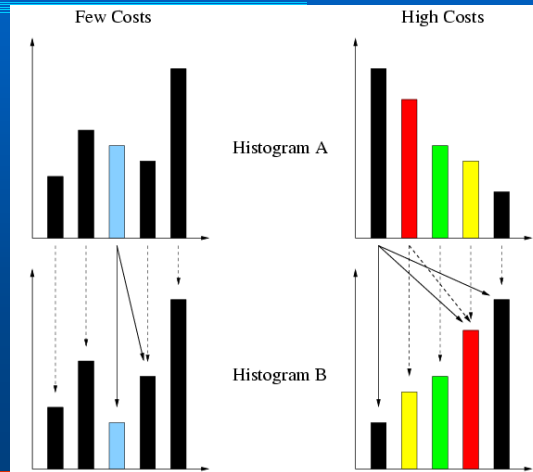
$$\rho(H_0, H_1) = \frac{E[H_0 H_1] - E[H_0]E[H_1]}{\sqrt{\text{Var}[H_0]\text{Var}[H_1]}}$$

Other Feature Metrics

- Earth Mover's Distance
 - What is the minimum cost to make one histogram the same as the other?

$$EMD(H_0, H_1) = \frac{\text{minimum cost}}{\min(\text{area}(H_0), \text{area}(H_1))}$$

Earth Mover's Distance Example



Mahalanobis Distance

Tries to account for feature correlation

$$r^2 = (\mathbf{x} - \mathbf{m}_x)' \mathbf{C}_x^{-1} (\mathbf{x} - \mathbf{m}_x)$$

where \mathbf{C}_x is the Covariance matrix of \mathbf{x}
and \mathbf{m}_x is the mean vector of \mathbf{x}

It also accounts for scaling of coordinate axis
and can provide curved decision boundaries

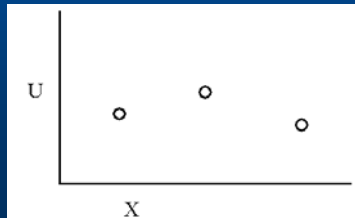
In the special case where features are independent and
variances are same, it becomes the SSD

Classification

- A *classification algorithm* is a system for classifying a dense set of observations or data given a sparse set of data.
- Classification algorithms usually involve a concept learning stage called the *training stage*, and a *classification stage* which takes as input the concept from the first stage as well as new data.
- Intuitively, concept learning is a process which takes preclassified empirical observations or the *training set*, and produces a representation of the data which is effective at correctly classifying the data.

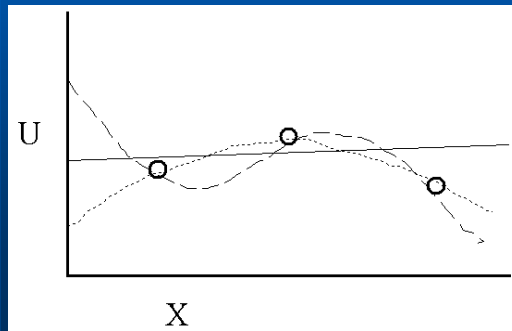
Classification

- An interesting mathematically oriented approach toward understanding learning algorithms is to think of them as approximating a function from a finite set of data points in N dimensional space, where N is the number of available feature classes. In one dimension, we have the class membership axis, U , along the y axis, and the value from the feature class, X , along the x axis.



Classification

- The *learning bias* is the model which decides between the various possible curves.



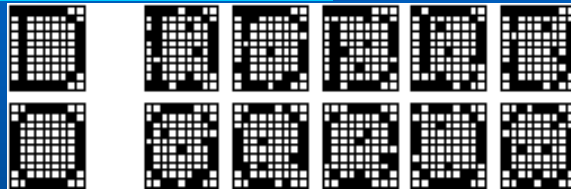
Classification System Properties

- Concept accuracy: probability of correct classification
- Concept conciseness: minimize storage space
- Auxiliary information:
 - Salience - importance of an attribute
 - Confidence - How confident is the classification? i.e. medical diagnosis
 - General applicability – Is the algorithm only useful in one application?
 - Efficiency – Is the algorithm too slow to use?
 - Ease of use – Is the algorithm very difficult to use in practice?
 - Transparency - Is the learned concept a “black box” or understandable?

Classification and Evaluation

- Voting
- Sequential Probability Ratio Test (SPRT)
- Minimum Distance Classifier (MDC)
 - also known as Nearest Neighbor Classifier
- Image Keys - Fisher's Optimal Linear Discriminant
- Principal Component Methods - Karhunen-Loeve Transform
- Scientific Validation, Benchmarking, Evaluation

Voting



- Every known letter casts a single vote for its class or the class of undecided. i.e. If the distance between the unknown letter and the known letter is less than a threshold then it votes for its class.
- Classify the unknown letter as the class which gets the most votes.

Sequential Probability Ratio Test (SPRT)

Let \mathbf{y}_m be a vector of m measurements y_1, y_2, \dots, y_m on an object to be classified as either w_1 or w_2 . SPRT is defined such that we classify \mathbf{y}_m according to

$$L(\mathbf{y}_m) = p(\mathbf{y}_m|w_1)/p(\mathbf{y}_m|w_2) \quad (4.1)$$

$$\text{If } L(\mathbf{y}_m) > A \text{ then choose } w_1 \quad (4.2)$$

$$\text{If } L(\mathbf{y}_m) < B \text{ then choose } w_2 \quad (4.3)$$

If $L(\mathbf{y}_m)$ falls between A and B then take another measurement.

where A and B are thresholds with $A > B$.

Minimum Distance Classifier (MDC) (perhaps the most popular classifier)

Suppose we have two feature classes, U (furriness) and V (pinkness).

(U,V) -> Classification

(0.2, 1.0) -> Pig

(0.2, 0.2) -> Fish

(1.0, 0.2) -> Cat

New Query

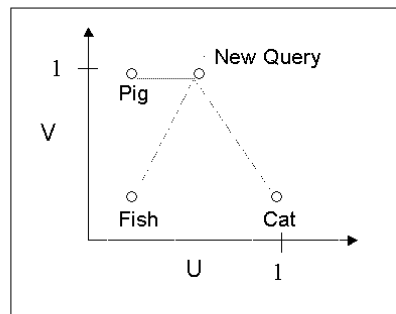
(0.5, 1.0)

Pig Distance = 0.3

Fish Distance = 1.3

Cat Distance = 1.3

Conclusion: New Query is a Pig



Example of Classification with Mahalanobis Distance with MDC

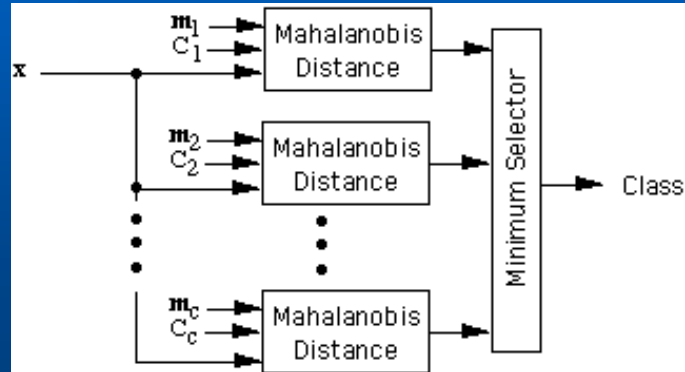


Image Keys

- Instead of computing the SSD of two images, X and Y ,

$$\sum_j \sum_i (X(i,j) - Y(i,j))^2$$

- we only need to compute the key distance

$$\sum_m (k(X,m) - k(Y,m))^2$$

Image Key Framework

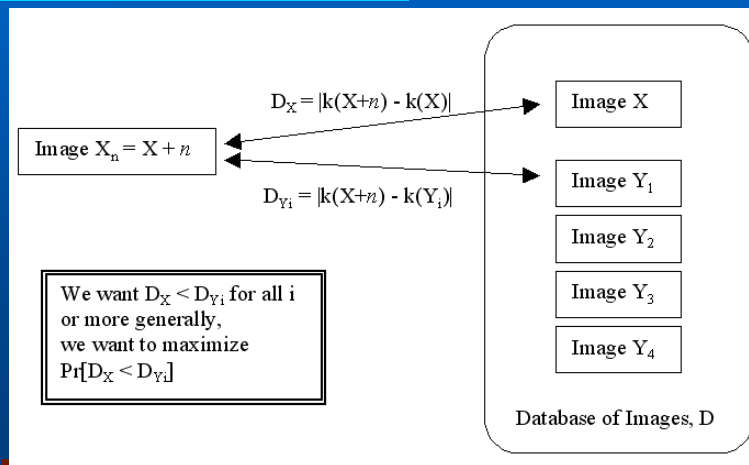
- Suppose we have an original image, X , distort it with noise to get $X_n = X + n$, and put X into the database. Then assuming Y is never X , we want

$$|k(X_n) - k(X)| < |k(X_n) - k(Y)| \text{ for all } Y \text{ where } |*| \text{ is the norm operator}$$

- No system is perfect so we choose to maximize the probability of finding X given X_n

$$\Pr\{|k(X+n) - k(X)| < |k(X+n) - k(Y)| \text{ is maximized}$$

Optimal Image Keys



Optimal Image Keys

- If we make the assumption of linear keys, $k(X+n) = k(X) + k(n)$, and implement the magnitude of a stochastic variable by the magnitude of its variance, then if Σ_X and Σ_n are the covariance matrices of X and n , then the criterion function is

$$\text{maximize } \frac{k^t \Sigma_X k}{k^t \Sigma_n k}$$

or

$$\begin{aligned} &\text{maximize } k^t \Sigma_X k \\ &\text{subject to } k^t \Sigma_n k = 1 \end{aligned}$$

Optimal Image Keys

- Why does this work?
- We are maximizing the between-class separation by Σ_X
- and minimizing the within-class distance by Σ_n

$$\text{maximize } \frac{k^t \Sigma_X k}{k^t \Sigma_n k}$$

- This means that we are deriving a new space where the different classes are far apart and each class is as compact as possible.

Linear Keys Notes

$$\text{maximize } k^t \Sigma_X k$$

$$\text{subject to } k^t \Sigma_n k = 1$$

- If Σ_n is the identity matrix, then it reduces to the Karhunen-Loeve Transform (KLT) also known as principal components.

Karhunen-Loeve Transform (KLT)

- The KLT is well known for "de-correlating" data and for being optimal in the sense of data representation.
- However, it only completely de-correlates vector components from a Gaussian source.
- From a practical perspective, the KLT converts an $M \times M$ image into N coefficients. This brings a speed increase of roughly $M \times M / N$. For example, for a 1000×1000 image, only 50 coefficients may need to be stored \rightarrow speedup of 20,000
- 2.3 years is reduced to 1 hour by using the KLT
- 5.5 hours is reduced to 1 second

Karhunen-Loeve Transform (KLT)

- (1) For M example faces, normalize/filter, convert to 1D vectors.
- (2) Compute the Covariance Matrix

$$Cov = E\{(x^T - m_x) \cdot (x - m_x)\} = \frac{1}{N^2} \sum_{k=1}^{N^2} (x_k^T - m_k) \cdot (x_k - m_k)$$

- (3) Compute the eigenvectors of the covariance matrix.
 - These are your principal components
 - Only keep the most descriptive N components

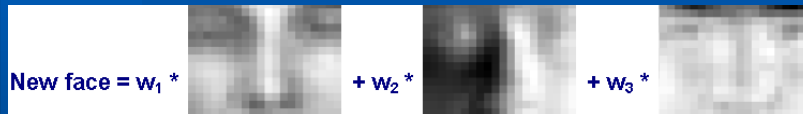
KLT Example

- What do eigenvectors look like - eigenvectors of faces:



KLT Example

- Approximate any new face by a linear combination of eigenfaces:



- Eigenfaces minimize the average truncation error where λ_j is the eigenvalue of the j^{th} eigenvector. "Optimal with respect to representation"

$$\sum_{j=m+1}^n \lambda_j$$

Why not always use the KLT? Isn't it good to always de-correlate your features?

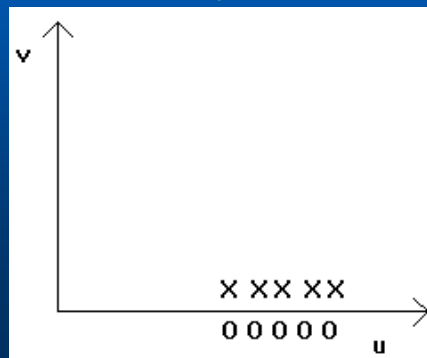
- Your source may not be Gaussian. In practice, many/most are provably not Gaussian.
- That said, using the KLT to try to de-correlate features might be a good step anyway - Its not clear if there are any significant problems with applying the KLT other than computational expense and the need for training data.
- In fact, this leads us back to the Mahalanobis distance which is essentially doing a KLT first.

What about deriving KLT for nonGaussian?

- That would be a good research topic.
- It would probably require iterative solutions for the basis vectors.

Does Representation Result in Detection?

- In this case, the main axis/eigenvector from the KLT will be u , but v is clearly better for classification.

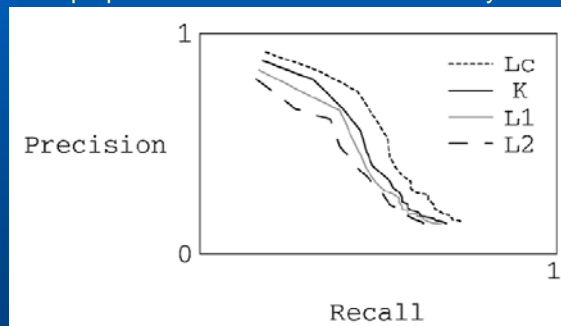


Evaluation

- Suppose we have two systems which are attempting to solve the "find the similar images" problem. How can we compare them regarding accuracy?

Precision-Recall Graphs

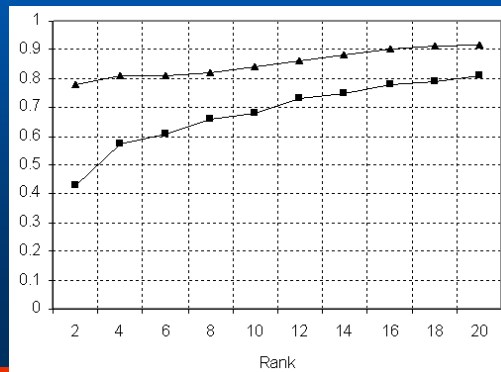
- Recall - proportion of relevant material actually retrieved
- Precision - proportion of retrieved material actually relevant



- Only relevant for a single window of results (i.e. top 10 ranks)

Detection/Rank Graphs

Gives the probability of finding at least one relevant result in the top N ranks. It does not say anything about finding ALL of the relevant results.



Experiments

Experimental evaluation of several distance measures in diverse vision areas.

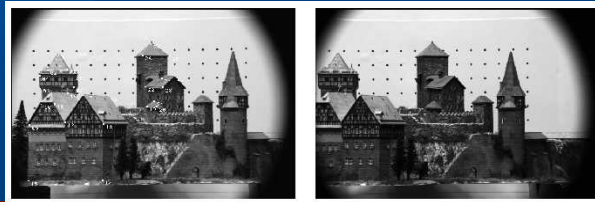
- Color image matching
- Color object recognition
- Texture classification
- Stereo Matching

Experiments: Sample Images

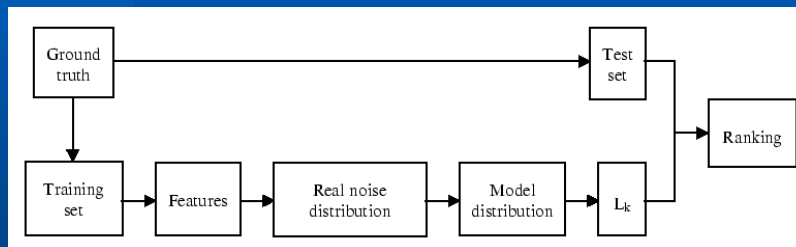
Color object:



Stereo Matching: Castle Stereo Pair - CMU:



Experimental Setup



Experiments: Promising Algorithms

Color Indexing: Perceptual Similarity
- Hafner, et al. [1995], PAMI

Color Object Recog: Color Invariants
- Gevers and Smeulders [1999], Patt. Recog.

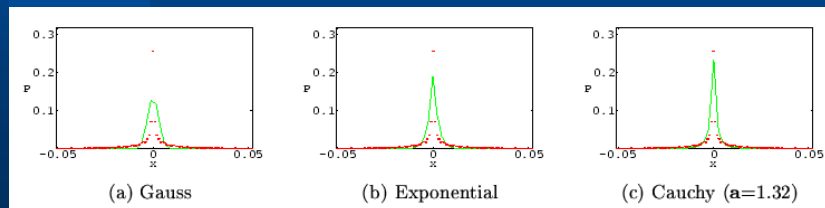
Stereo Matching: Cox, et al. [1996], CVIU
Fusiello, et al. [1997], CVPR

Texture Classification: Wavelet Methods
Smith and Chang [1994], ICIP
Ma and Manjunath [1996], CVPR

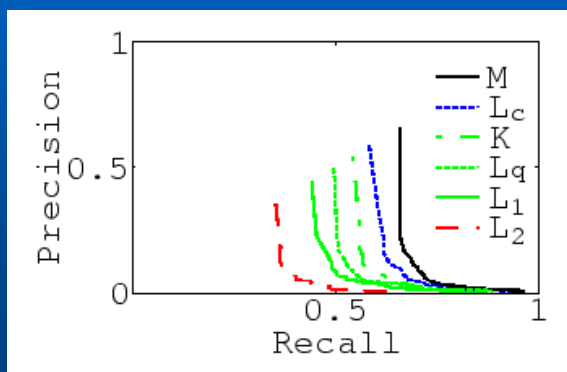
Color Image Matching

Database: 8,200 images from the Corel Professional photo database (www.corel.com); 82 in ground truth (GT); 50 in training set.

Distortion: Copies printed then scanned;
Varying Trans., Rot., Scale, Color fidelity.



Color Image Matching (*M is the Maximum Likelihood Metric*)

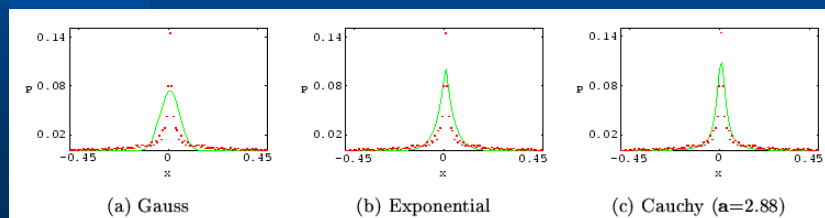


Color Object Recognition

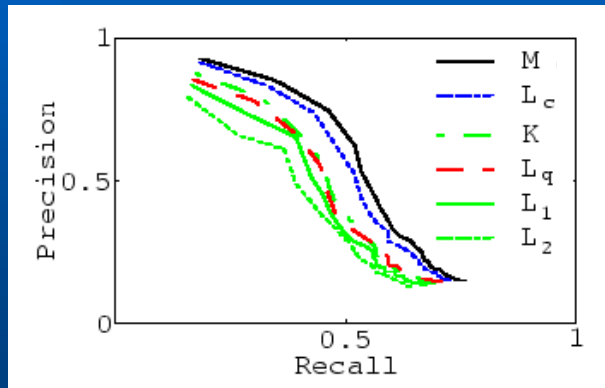
Database: 500 images from the object recog. database from Gevers and Smeulders[1999].
GT:48 images of 8 objects

Algorithm: Color invariants: I_1, I_2, I_3 ; with L_1 metric

Distortion: Different viewpoints of the same object



Color Object Recognition

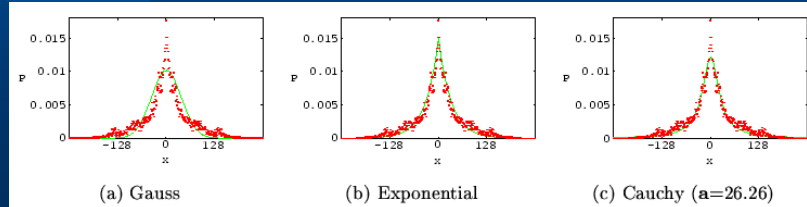


Stereo Matching

Algorithms: A Maximum Likelihood Stereo Algorithm
 - Cox, et al., *CVIU*, 1996

Efficient Stereo with Multiple Windowing
 - Fusiello, et al., *CVPR*, 1997
 (both algorithms used L_2)

Database: Castle and Tower sets from CMU; Stuttgart ISPRS
 Image Understanding Dataset - Flat & Suburb



Stereo Matching

Results Using Cox's algorithm:

| Set | L_2 | L_1 | K | L_c | M |
|--------|-------|-------|-------|-------------------------|-------|
| Castle | 93.45 | 94.72 | 94.53 | 95.72 $\mathbf{a}=7.47$ | 96.37 |
| Tower | 93.18 | 95.07 | 94.74 | 96.18 $\mathbf{a}=5.23$ | 97.04 |
| Robots | 74.81 | 76.76 | 78.15 | 82.51 $\mathbf{a}=26.2$ | 84.38 |

Results Using Fusiello's algorithm:

| Set | L_2 | L_1 | K | L_c | M |
|--------|-------|-------|-------|-------------------------|-------|
| Castle | 92.27 | 92.92 | 92.76 | 94.82 $\mathbf{a}=7.47$ | 95.73 |
| Tower | 91.79 | 93.67 | 93.14 | 95.28 $\mathbf{a}=5.23$ | 96.05 |
| Robots | 72.15 | 73.74 | 75.87 | 77.69 $\mathbf{a}=26.2$ | 79.54 |

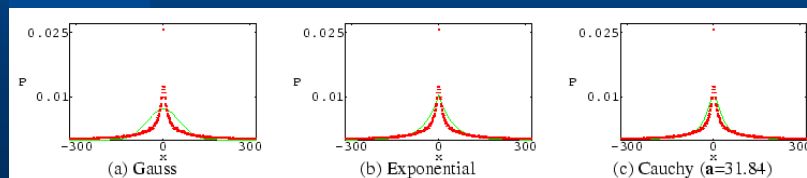
Texture Classification

(see Sebe and Lew, IEEE Multimedia'2000, ACCV'2000)

Algorithm: QMF Wavelets (also tried distribution based textures such as LBP, Laws,...);
32 dimensional feature vector (mean & var)

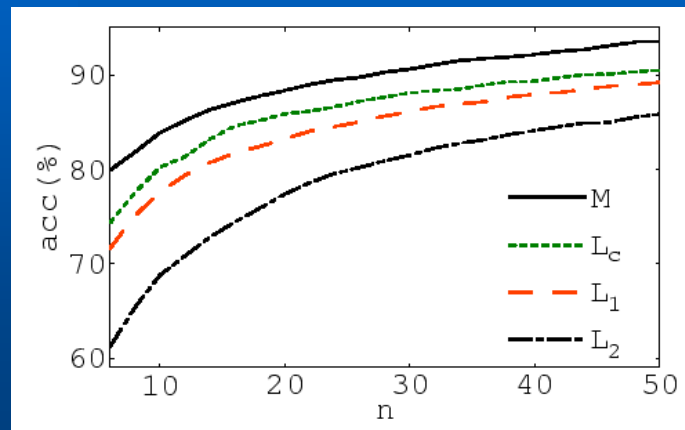
Database: GT:2240 samples from 112 Brodatz textures
TS: 1000 samples

Distortion: Different samples of the same texture



Texture Classification

(graph below is for QMF. Gabor has similar results)



Observations

- If your data set is not Gaussian, you have *at least* the following two choices:
 - (1) Transform the original features so that the new features have a Gaussian distribution and then use L_2 . This method was explored with respect to stereo matching by Cox, et al. [1996]
 - (2) Find the maximum likelihood metric directly from training data and use the original features

Observations

- If you have a data set which is Gaussian, then L_2 is appropriate.
- Observation: *We have not yet found a large data set in visual matching which exhibits a Gaussian noise distribution -> Robust methods/Max. Like. are important.*

Summary & Recommendations

- If you have a small training set, use and train the Cauchy metric
- If you have a large training set and the features are not correlated, use the maximum likelihood metric
- If you have a large training set and the features are highly correlated, strongly consider the Mahalanobis distance.
- No single evaluation graph is best. The precision-recall graphs and the detection-rank graphs are frequently used and accepted but known to have limitations