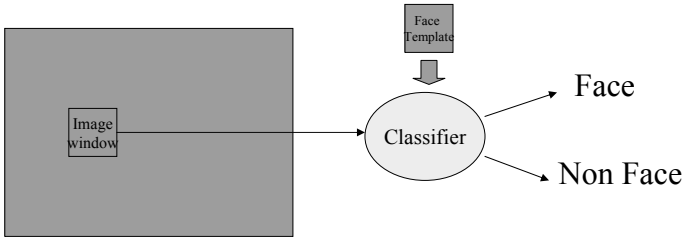
Object Detection and Recognition

Spring 2005
Ahmed Elgammal
Dept of Computer Science
Rutgers University

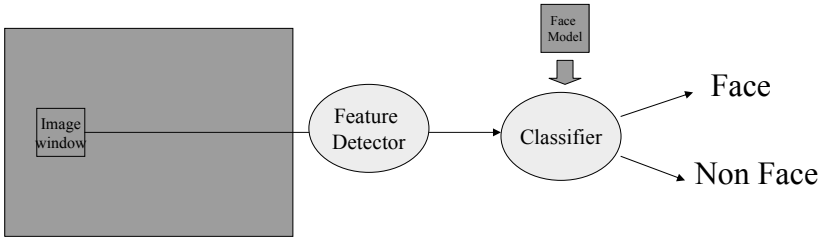
Finding Templates Using Classifiers

- Example: we want to find faces in a given image
- We can learn a model of the face appearance "template"
- Since we don't know where the face is in the image, we can test all image windows of a particular size and decide whether is contains a face template or not
- If we don't know how big is the object in the image: search over scale
- If we don't know the orientation: search over the orientation as well
- …



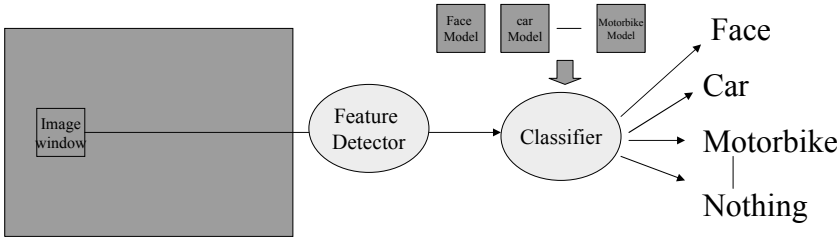
Framework

- Images: training data, testing data
- Feature selection and representation:
  - Input: Images
  - Methods: PCA, Linear discriminant analysis etc
  - Output: Features
    - One template for one object (template matching)
    - Multiple templates with constrains for one object
- Supply the features to a classifier:
  - Input: Features
  - Types: Probability model, determine decision boundary directly etc.
  - Output: class label



Framework

- The same framework applies for multiple object detection/recognition

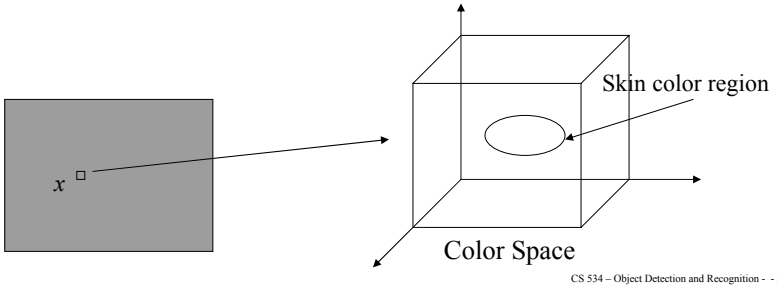


Histogram based classifiers

- Use a histogram to represent the class-conditional densities
  - (i.e. p(x|1), p(x|2), etc)
- Advantage: estimates become quite good with enough data!
- Disadvantage: Histogram becomes big with high dimension
  - but maybe we can assume feature independence? (Naïve Bayes)

Finding skin

- Skin has a very small range of (intensity independent) colors, and little texture
  - Compute an intensity-independent color measure, check if color is in this range, check if there is little texture (median filter)
  - See this as a classifier - we can set up the tests by hand, or learn them.
  - get class conditional densities (histograms), priors from data (counting)
- Classifier is
  - if $p(\text{skin}|\boldsymbol{x}) > \theta$, classify as skin
  - if $p(\text{skin}|\boldsymbol{x}) < \theta$, classify as not skin
  - if $p(\text{skin}|\boldsymbol{x}) = \theta$, choose classes uniformly and at random



Approach:
Construct a histogram of RGB values for skin pixels and another histogram for non-skin pixel.
The histograms represent: P(x|skin) and P(x|non-skin)

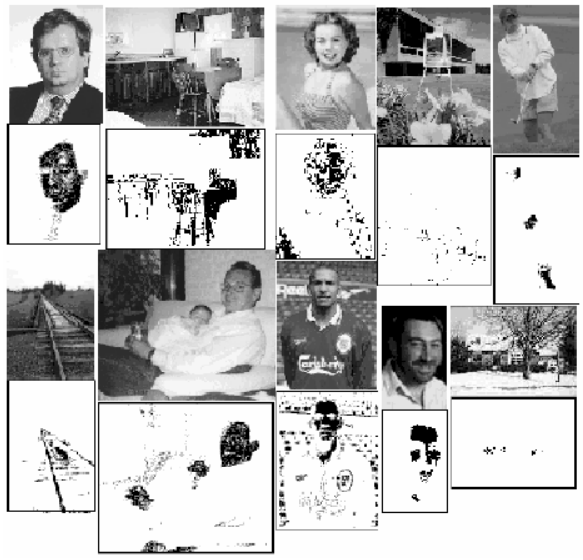
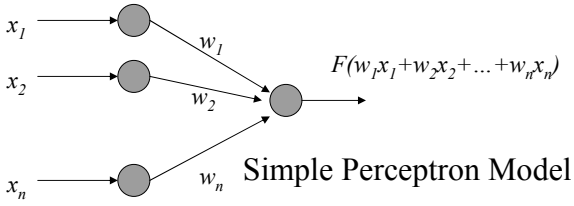


Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 copyright 1999, IEEE

Neural networks

- Linear decision boundaries are useful
  - but often not very powerful
  - we seek an easy way to get more complex boundaries
- Compose linear decision boundaries
  - i.e. have several linear classifiers, and apply a classifier to their output
  - a nuisance, because sign(ax+by+cz) etc. isn't differentiable.
  - use a smooth "squashing function" in place of sign.
- Neural network is a parametric approximation technique to build a model of posterior Pr(k|x) .



Simple Perceptron Model

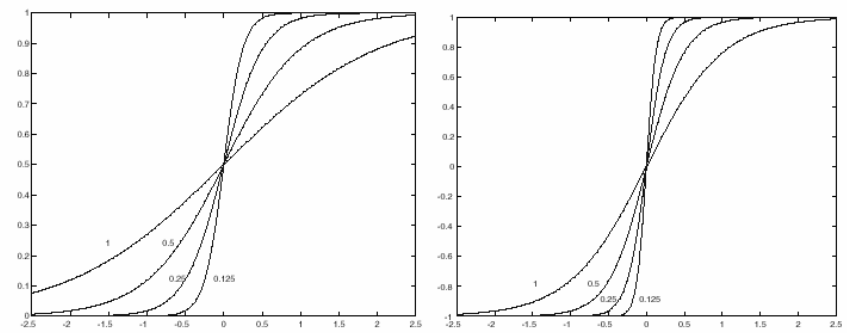
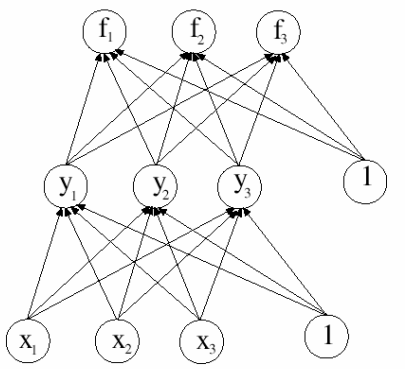
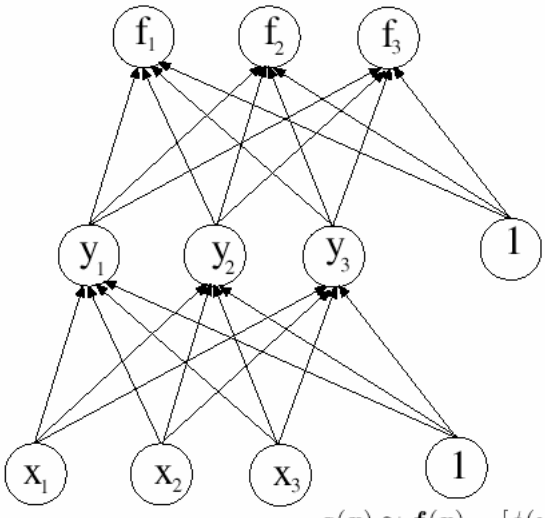


Figure 22.14. On the **left**, a series of squashing functions obtained using $\phi(x;\nu) = \frac{e^{x/\nu}}{1+e^{x/\nu}}$, for different values of ν indicated on the figure. On the **right**, a series of squashing functions obtained using $\phi(x;\nu,A) = A\tanh(x/\nu)$ for different values of ν indicated on the figure. Generally, for x close to the center of the range, the squashing function is linear; for x small or large, it is strongly non-linear.

- Multi-layered perceptron
- Approximate complex decision boundaries by combining simple linear ones
- Can be used to approximate any nonlinear mapping function from the input to the output.





$$\boldsymbol{g}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{x}) = [\phi(\boldsymbol{w}_{21} \cdot \boldsymbol{y}), \phi(\boldsymbol{w}_{22} \cdot \boldsymbol{y}), \ldots \phi(\boldsymbol{w}_{2n} \cdot \boldsymbol{y})]$$

$$\boldsymbol{y}(\boldsymbol{z}) = [\phi(\boldsymbol{w}_{11} \cdot \boldsymbol{z}), \phi(\boldsymbol{w}_{12} \cdot \boldsymbol{z}), \ldots \phi(\boldsymbol{w}_{1m} \cdot \boldsymbol{z}), 1]$$

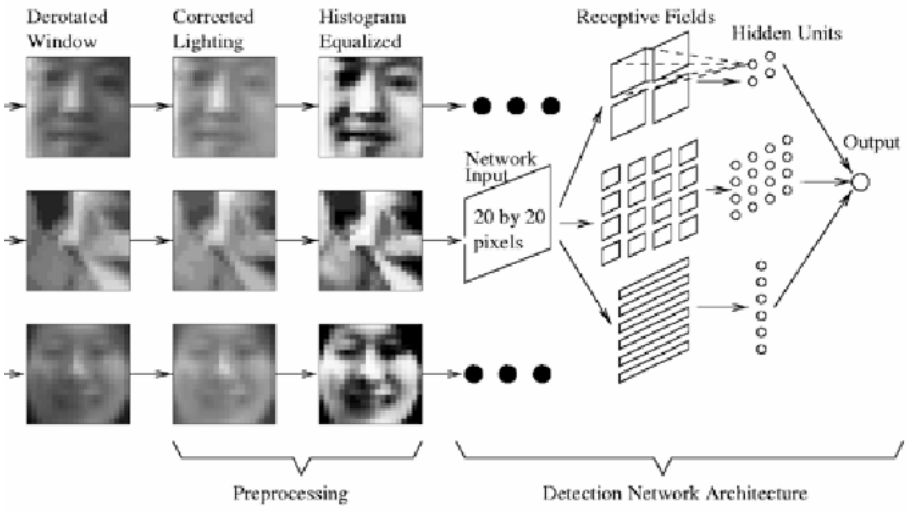
$$\boldsymbol{z}(\boldsymbol{x}) = [x_1, x_2, \ldots, x_p, 1]$$

Training

- Given input, output pairs *(x,o):*
- Choose parameters to minimize error on training set

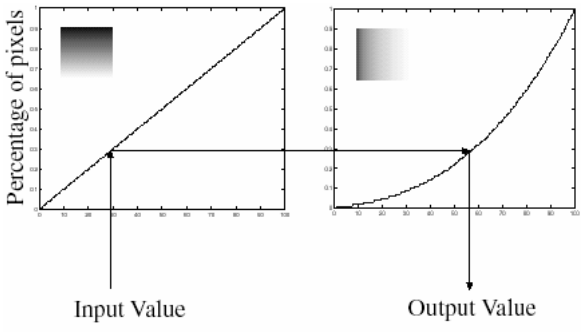
$$Error(p) = \left(\frac{1}{2}\right) \sum_e \left(n\left(x^e; p\right) - o^e\right)$$

- Where *p* is the set of weights
- Stochastic gradient descent, computing gradient using trick (backpropagation, aka the chain rule)
- Stop when error is low, and hasn't changed much

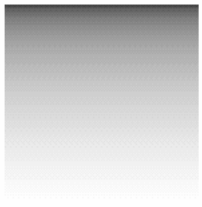


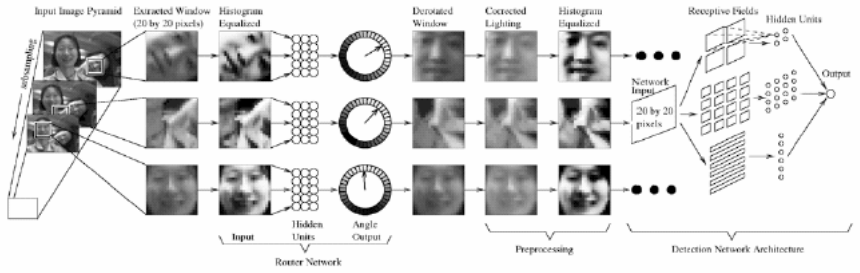
The vertical face-finding part of Rowley, Baluja and Kanade's system

Figure from "Rotation invariant neural-network based face detection," H.A. Rowley, S. Baluja and T. Kanade, Proc. Computer Vision and Pattern Recognition, 1998, copyright 1998, IEEE



Histogram equalisation gives an approximate fix for illumination induced variability





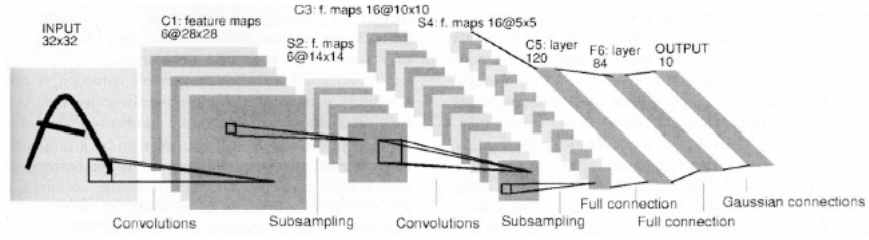
Architecture of the complete system: they use another neural net to estimate orientation of the face, then rectify it. They search over scales to find bigger/smaller faces.

Figure from "Rotation invariant neural-network based face detection," H.A. Rowley, S. Baluja and T. Kanade, Proc. Computer Vision and Pattern Recognition, 1998, copyright 1998, IEEE

Figure from "Rotation invariant neural-network based face detection," H.A. Rowley, S. Baluja and T. Kanade, Proc. Computer Vision and Pattern Recognition, 1998, copyright 1998, IEEE

Convolutional neural networks

- Also known as gradient-based learning
- Template matching using NN classifiers seems to work
- Natural features are filter outputs
  - probably, spots and bars, as in texture
  - but why not learn the filter kernels, too?
- Recall: a perceptron approximate convolution.
- Network architecture: Two types of layers
  - Convolution layers: convolving the image with filter kernels to obtain filter maps
  - Subsampling layers: reduce the resolution of the filter maps
  - The number of filter maps increases as the resolution decreases



A convolutional neural network, LeNet; the layers filter, subsample, filter, subsample, and finally classify based on outputs of this process.

Figure from "Gradient-Based Learning Applied to Document Recognition", Y. Lecun et al Proc. IEEE, 1998 copyright 1998, IEEE

Fig. 4. Size-normalized examples from the MNIST database.

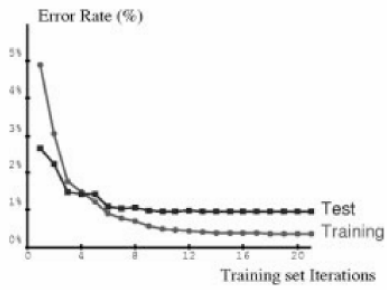


Fig. 5. Training and test error of LeNet-5 as a function of the number of passes through the 60 000 pattern training set (without distortions). The average training error is measured on-the-fly as training proceeds. This explains why the training error appears to be larger than the test error initially. Convergence is attained after 10–12 passes through the training set.

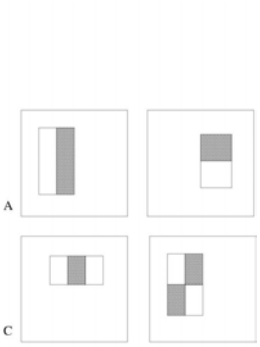
LeNet is used to classify handwritten digits. Notice that the test error rate is not the same as the training error rate, because the test set consists of items not in the training set. Not all classification schemes necessarily have small test error when they have small training error. Error rate 0.95% on MNIST database

Figure from "Gradient-Based Learning Applied to Document Recognition", Y. Lecun et al Proc. IEEE, 1998 copyright 1998, IEEE

Viola & Jones Face Detector

- "Robust Real-time Object Detection" Paul Viola and Michael Jones in ICCV 2001 Workshop on Statistical and Computation Theories of Vision
- State of the art Face detector
- Rapid evaluation of simple features for object detection
- Method for classification and feature selection, a variant of AdaBoost
- Speed-up through the Attentional Cascade

Definition of simple features for object detection



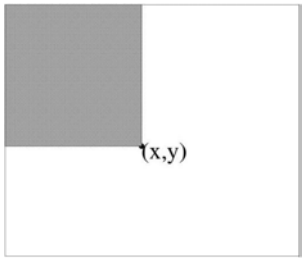
3 rectangular features types:

- *two-rectangle feature* type (horizontal/vertical)
- *three-rectangle feature* type
- *four-rectangle feature* type

Using a 24x24 pixel base detection window, with all the possible combination of horizontal and vertical location and scale of these feature types the full set of features has 49,396 features.

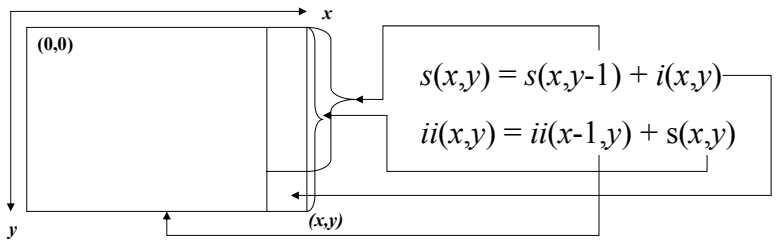
The motivation behind using rectangular features, as opposed to more expressive steerable filters is due to their extreme computational efficiency.

Integral image

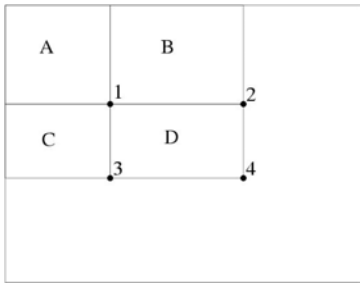


Def: The *integral image* at location (x,y), is the sum of the pixel values above and to the left of (x,y), inclusive.

Using the following two recurrences, where $i(x,y)$ is the pixel value of original image at the given location and $s(x,y)$ is the cumulative column sum, we can calculate the integral image representation of the image in a single pass.



Rapid evaluation of rectangular features



Using the integral image representation one can compute the value of any rectangular sum in constant time.

For example the integral sum inside rectangle D we can compute as:

$$ii(4) + ii(1) - ii(2) - ii(3)$$

As a result two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively.

Challenges for learning a classification function

- Given a feature set and labeled training set of images one can apply number of machine learning techniques.
- Recall however, that there is 45,396 features associated with each image sub-window, hence the computation of all features is computationally prohibitive.
- Hypothesis: A combination of only a small number of these features can yield an effective classifier.
- Challenge: Find these discriminant features.

Boosting Classifiers

- Objective: Combine week classifiers (weak learner: classify the training date correctly 51% of the time) to obtain a stronger one.
- Simple example: Majority votes
- Most successful and Popular: **AdaBoost**
- **AdaBoost** *Freund and Schapire*:
  - a greedy feature selection process
  - Select a small set of good classifiers and combine them
  - Associate large weight with good classifier and smaller weights with poor ones
  - Final strong classifier takes the form of a perception, a weighted combination of the weak classifiers followed by a threshold.
  - Strong Classifier**:** $h(x) = \Sigma_t \alpha_t h_t(x)$

A variant of AdaBoost for aggressive feature selection

- Given example images (x_1,y_1) , … , (x_n,y_n) where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = 1/(2m), 1/(2l)$ for training example i, where m and l are the number of negatives and positives respectively.

For $t = 1 \ldots T$

1) Normalize weights so that w_t is a distribution
2) For each feature j train a classifier h_j and evaluate its error ε_j with respect to w_t.
3) Chose the classifier h_j with lowest error.
4) Update weights according to:

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\varepsilon_i}$$

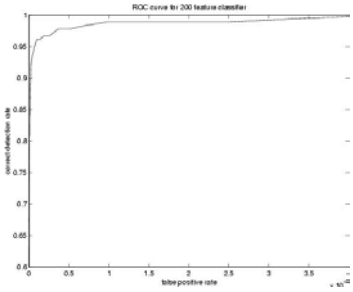
where $e_i = 0$ is x_i is classified correctly, 1 otherwise, and

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

- The final strong classifier is:

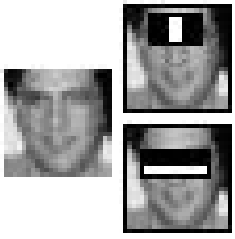
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}, \qquad \text{where} \qquad \alpha_t = \log(\frac{1}{\beta_t})$$

Performance of 200 feature face detector



The ROC curve of the constructed classifies indicates that a reasonable detection rate of 0.95 can be achieved while maintaining an extremely low false positive rate of approximately 10^{-4}.

- First features selected by AdaBoost are meaningful and have high discriminative power
- By varying the threshold of the final classifier one can construct a two-feature classifier which has a detection rate of 1 and a false positive rate of 0.4.



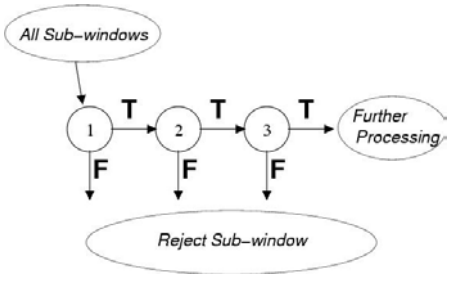
Speed-up through the Attentional Cascade

- Simple, boosted classifiers can reject many of negative sub-windows while detecting all positive instances.
- Series of such simple classifiers can achieve good detection performance while eliminating the need for further processing of negative sub-windows.
- Overall false positive rate

$$F = \prod_{i=1}^{K} f_i$$

- Overall Detection rate

$$D = \prod_{i=1}^{K} d_i$$



Processing in / training of the Attentional Cascade

Processing: is essentially identical to the processing performed by a degenerate decision tree, namely only a positive result from a previous classifier triggers the evaluation of the subsequent classifier.

Training: is also much like the training of a decision tree, namely subsequent classifiers are trained only on examples which pass through all the previous classifiers. Hence the task faced by classifiers further down the cascade is more difficult.

To achieve efficient cascade for a given false positive rate F and detection rate D we would like to minimize the expected number of features evaluated N:

$$N = n_0 + \sum_{i=1}^{K}\left(n_i \prod_{j<i} p_j \right)$$

Since this optimization is extremely difficult the usual framework is to choose a minimal acceptable false positive and detection rate per layer.

Algorithm for training a cascade of classifiers

- User selects values for f, the maximum acceptable false positive rate per layer and d, the minimum acceptable detection rate per layer.
- User selects target overall false positive rate F_{target}.
- P = set of positive examples
- N = set of negative examples
- $F_0 = 1.0$; $D_0 = 1.0$; $i = 0$

While $F_i > F_{target}$
- i++
- $n_i = 0$; $F_i = F_{i-1}$
- while $F_i > f$ x F_{i-1}
  - n_i ++
  - Use P and N to train a classifier with n_i features using AdaBoost
  - Evaluate current cascaded classifier on validation set to determine F_i and D_i
  - Decrease threshold for the ith classifier until the current cascaded classifier has a detection rate of at least d x D_{i-1} (this also affects F_i)
- $N = \varnothing$
- If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set N.

Experiments (dataset for training)

- 4916 positive training example were hand picked aligned, normalized, and scaled to a base resolution of 24x24
- 10,000 negative examples were selected by randomly picking sub-windows from 9500 images which did not contain faces

Experiments cont.
(structure of the detector cascade)

- The final detector had 32 layers and 4297 features total

Layer number	1	2	3 to 5	6 and 7	8 to 12	13 to 32
Number of feautures	2	5	20	50	100	200
Detection rate	100%	100%	-	-	-	-
Rejection rate	60%	80%	-	-	-	-

- Speed of the detector ~ total number of features evaluated
- On the MIT-CMU test set the average number of features evaluated is 8 (out of 4297).
- The processing time of a 384 by 288 pixel image on a conventional personal computer about .067 seconds.
- Processing time should linearly scale with image size, hence processing of a 3.1 mega pixel images taken from a digital camera should approximately take 2 seconds.

Operation of the face detector

- Since training examples were normalized, image sub-windows needed to be normalized also. This **normalization** of images can be efficiently done using two integral images (regular / squared).
- **Detection at multiple scales** is achieved by scaling the detector itself.
- The amount of **shift** between subsequent sub-windows is determined by some constant number of pixels and the current scale.
- **Multiple detections** of a face, due to the insensitivity to small changes in the image of the final detector were, were combined based on overlapping bounding region.

Results

Testing of the final face detector was performed using the MIT+CMU frontal face test which consists of:

• 130 images

• 505 labeled frontal faces

Results in the table compare the performance of the detector to best face detectors known.

False detections	10	31	50	65	78	95	110	167	422
Viola-Jones	78.3%	85.2%	88.8%	89.8%	90.1%	90.8%	91.1%	91.8%	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	-	90.1%	89.9%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-	-	-
Roth-Yang-Ajuha	-	-	-	-	94.8%	-	-	-	-

Rowley at al.: use a combination of 1wo neural networks (simple network for prescreening larger regions, complex network for detection of faces).

Schneiderman at al.: use a set of models to capture the variation in facial appearance; each model describes the statistical behavior of a group of wavelet coefficients.

Results cont.

Summary of contributions

- The paper presents general object detection method which is illustrated on the face detection task.
- Using the integral image representation and simple rectangular features eliminate the need of expensive calculation of multi-scale image pyramid.
- Simple modification to AdaBoost gives a general technique for efficient feature selection.
- A general technique for constructing a cascade of homogeneous classifiers is presented, which can reject most of the negative examples at early stages of processing thereby significantly reducing computation time.
- A face detector using these techniques is presented which is comparable in classification performance to, and orders of magnitude faster than the best detectors know today.

Matching by relations

- In previous approached, we assume we can find one template to match the object:
  - Problem: objects with complex configuration spaces: Appearance is highly variable.
    - internal degrees of freedom:
      - articulated objects (e.g. human body),
      - deformable objects (e.g., a fish, snake,…)
    - Class variability: how to things like recognize cars, motorbikes,
    - (possibly) shading
  - Solution: find multiple simple templates, then say object is present if these templates agree.

Matching by relations

- Advantages:
  - Simple templates are easy to learn:e.g. it is easy to learn an eye template than a whole face template. (much less appearance variability for eyes)
  - We might use simple probability models: Some independence properties can be exploited.
  - Simple templates can be shared: we can match many object with relatively small number of templates, e.g., animal faces have eyes, nose, mouth with slightly different spatial layout.
- Simple individual templates can be used to construct complex objects

Simplest

- Define a set of local feature templates (image patches)
  - could find these with filters, etc.
  - corner detector+filters
- Think of objects as patterns of these features
- Each template votes for all patterns that contain it
- Pattern with the most votes wins

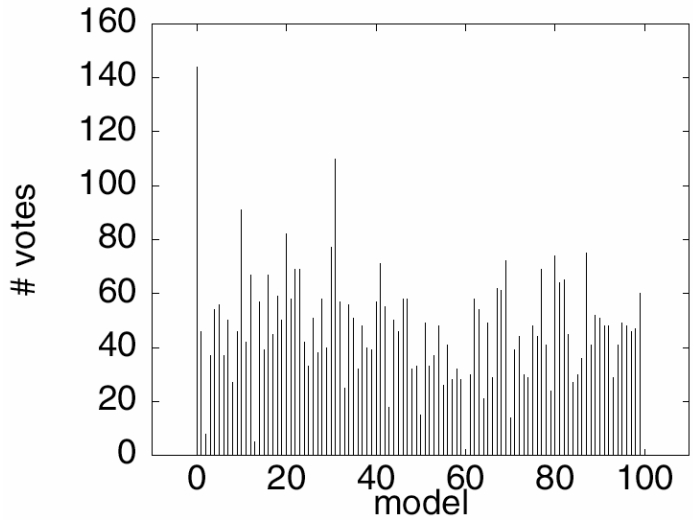


Figure from "Local grayvalue invariants for image retrieval," by C. Schmid and R. Mohr, IEEE Trans. Pattern Analysis and Machine Intelligence, 1997 copyright 1997, IEEE

Probabilistic interpretation

- Write

$$P\{\text{patch of type } i \text{ appears in image}|j\text{'th pattern is present}\} = p_{ij}$$

- Assume

$$P\{\text{patch of type } i|\text{no pattern is present}\} = p_{ix}$$

- Likelihood of image given pattern

$$p_{ij} = \mu \text{ if the pattern can produce this patch and 0 otherwise}$$
$$p_{ix} = \lambda < \mu \text{ for all } i.$$

that n_p patches came from that pattern and $n_i - n_p$ patches come from noise, is

$$P(\text{interpretation}|\text{pattern}) = \lambda^{n_p}\mu^{(n_i - n_p)}$$

Employ spatial relations

A feature matches to an object only if there are nearby features which also match to the object and are in the proper configuration.

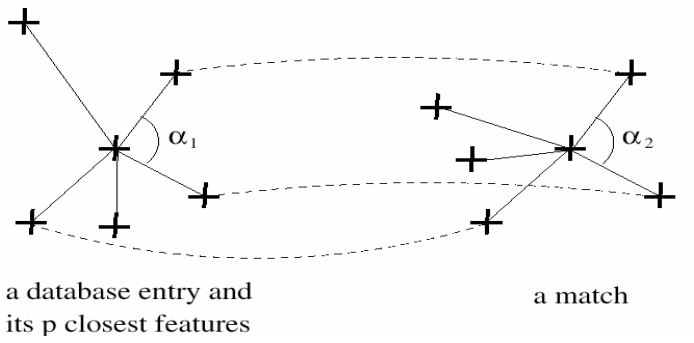


Figure from "Local grayvalue invariants for image retrieval," by C. Schmid and R. Mohr, IEEE Trans. Pattern Analysis and Machine Intelligence, 1997 copyright 1997, IEEE

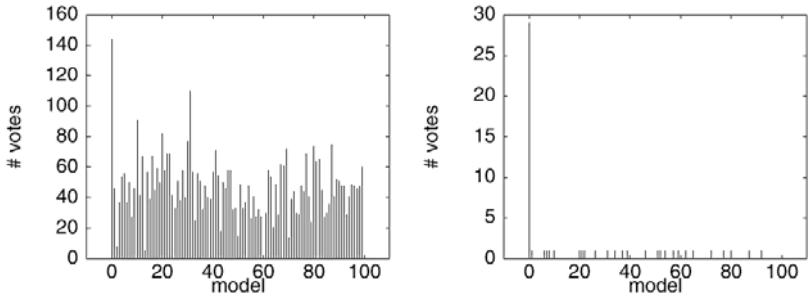


Figure from "Local grayvalue invariants for image retrieval," by C. Schmid and R. Mohr, IEEE Trans. Pattern Analysis and Machine Intelligence, 1997 copyright 1997, IEEE

Use probability representation for the spatial constraints

- work from R. Fergus etc *"Object class recognition by Unsupervised Scale-Invariant Learning"* CVPR 2003
- Task: Learn and recognize object class model from unlabeled and un-segmented cluttered scenes.
- Contributions:
  - Entropy based feature detector is used to select regions and their scale within the image.
  - Objects are modeled as flexible constellations of parts (simple templates). Use probability representation for all aspects of the object:
    - Appearance: A
    - Shape: X
    - Relative Scale: S
    - Occlusion: h
  - In learning stage, the model are estimated using expectation-maximization in a maximum-likelihood setting. In recognition stage, this model is used in a Bayesian manner to classify images.

- How to represent category to capture the essence, which is common to the objects that belong to it and flexible enough to accommodate object variance:
  - Object categories are represented as a collection of features, or parts. Each part has a distinctive appearance and spatial position.
  - In this work, a probability approach is used to model objects as random constellations of parts.
  - This model explicitly accounts for appearance variations, shape variations and for the randomness in the presence/absence of features due to occlusion and detector errors.
- Feature selection and representation:
  - Entropy based feature detector to select regions and their scale within the image.
    - For each point of the image a histogram is made of the intensities in a circular region of radius s.
    - The entropy H(s) of this histogram is calculated and the saliency of the region is measured by
    - The N regions with highest saliency over the image provide the features for learning and recognition.
  - Once the regions are identified, they are cropped from the image and rescaled to a 11X11 pixel patch, which makes the feature vector in 121 dimensional space. To reduce the dimension, PCA is applied to these features.

- Probability model for object class:
  - An object model consists of a number of parts, each part has an appearance, relative scale and can be occluded or not. Shape is represented by the mutual position of the parts.
  - The model is generative and probabilistic, so appearance, shape, scale and occlusion are all modeled by probability density functions, which are Gaussian.
- Once we learn the model, we can build the two-class classifier by using the Bayesian decision R: (Posterior=likelihood * Prior)

$$\begin{aligned} R &= \frac{p(\text{Object}|\mathbf{X},\mathbf{S},\mathbf{A})}{p(\text{No object}|\mathbf{X},\mathbf{S},\mathbf{A})} \\ &= \frac{p(\mathbf{X},\mathbf{S},\mathbf{A}|\text{Object})\,p(\text{Object})}{p(\mathbf{X},\mathbf{S},\mathbf{A}|\text{No object})\,p(\text{No object})} \\ &\approx \frac{p(\mathbf{X},\mathbf{S},\mathbf{A}|\,\theta)\,p(\text{Object})}{p(\mathbf{X},\mathbf{S},\mathbf{A}|\theta_{bg})\,p(\text{No object})} \end{aligned}$$

- X: location. S: relative scales A: appearance.

- Prior can be calculated by counting. So we need to learn the likelihood function, which can be factored out by density function of different aspects of the object:

$$p(\mathbf{X},\mathbf{S},\mathbf{A}|\,\theta) = \sum_{\mathbf{h}\in H} p(\mathbf{X},\mathbf{S},\mathbf{A},\mathbf{h}|\,\theta) =$$

$$\sum_{\mathbf{h}\in H} \underbrace{p(\mathbf{A}|\mathbf{X},\mathbf{S},\mathbf{h},\theta)}_{Appearance}\underbrace{p(\mathbf{X}|\mathbf{S},\mathbf{h},\theta)}_{Shape}\underbrace{p(\mathbf{S}|\mathbf{h},\theta)}_{Rel.\ Scale}\underbrace{p(\mathbf{h}|\theta)}_{Other}$$

- Build the density model for each aspect of the object:
  - Model each part p's appearance as a point in some appearance space following Gaussian distribution with mean and covariance parameters

$$\theta_p^{app} = \{\mathbf{c_p}, V_p\}$$

$$\frac{p(\mathbf{A}|\mathbf{X},\mathbf{S},\mathbf{h},\theta)}{p(\mathbf{A}|\mathbf{X},\mathbf{S},\mathbf{h},\theta_{bg})} = \prod_{p=1}^{P}\left(\frac{\mathrm{G}(\mathbf{A}(h_p)|\mathbf{c}_p,V_p)}{\mathrm{G}(\mathbf{A}(h_p)|\mathbf{c}_{bg},V_{bg})}\right)^{d_p}$$

  - Since shape is mutual location of all the points, model the shape as a joint Gaussian density of the locations of features within a hypothesis with parameter $\theta^{shape} = \{\boldsymbol{\mu}, \Sigma\}$.

$$\frac{p(\mathbf{X}|\mathbf{S},\mathbf{h},\theta)}{p(\mathbf{X}|\mathbf{S},\mathbf{h},\theta_{bg})} = \mathrm{G}(\mathbf{X}(\mathbf{h})|\boldsymbol{\mu},\Sigma)\,\alpha^f$$

  - The model for the background assumes features to be spread uniformly over the image which has area α
  - Model the scale of each part p relative to a reference frame as a Gaussian density which has parameters , $\theta^{scale} = \{t_p, U_p\}$.

$$\frac{p(\mathbf{S}|\mathbf{h},\theta)}{p(\mathbf{S}|\mathbf{h},\theta_{bg})} = \prod_{p=1}^{P}\mathrm{G}(\mathbf{S}(h_p)|t_p,U_p)^{d_p}\,r^f$$

  - The background model assumes a uniform distribution over scale (within a range r)

– And model for occlusion and statistics of the features finder:

$$\frac{p(\mathbf{h}|\theta)}{p(\mathbf{h}|\theta_{bg})} = \frac{p_{Poiss}(n|M)}{p_{Poiss}(N|M)} \frac{1}{{}^{n}C_{r}(N,f)} p(\mathbf{d}|\theta)$$

– The first term models the number of features detected using a Poisson distribution, which has a mean M. The second term is the book-keeping term for hypothesis variables.

- So from the training data, if we can learn the parameters for each model of different aspect of object,
- Then we can do the two–class classification.
- The task of learning, which is to estimate the parameters:
  –
  $$\theta = \{\boldsymbol{\mu}, \Sigma, \mathbf{c}, V, M, p(\bar{\mathbf{d}}|\theta), t, U\}$$
  – is carried out by expectation maximization (EM) algorithm.

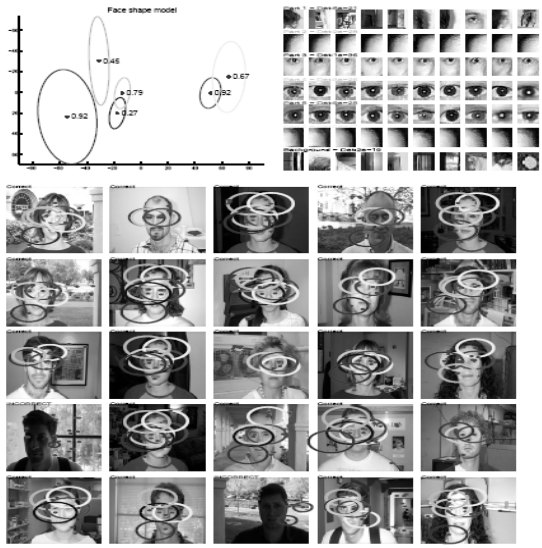


Figure from "Object Class Recognition by Unsupervised Scale-Invariant Learning," R. Fergus P. Perona and A. Zisserman, CVPR 2003, IEEE.