Object Recognition

Goal: Introduce central issues of object recognition, basic techniques, and emerging directions.

Outline:

1. What is object recognition and why is it challenging?
2. Historical perspective
3. Basic view-based classifiers and common problems
4. Boosting
5. Feature-based models
6. Multiple classes, context, and parsing

Matlab Tutorials and Demo Code:

- SIFT tutorial

Acknowledgements: Slides on Bag-of-Words models adapted from CVPR 2007 tutorial on recognition by Torralba, Fei-Fei and Fergus.

Background

Types of visual recognition problems:

- validation
- detection
- instance recognition (identification)
- category recognition
- scene/context recognition
- activity recognition

Challenges:

- variation in view point and lighting
- variation in shape, pose and appearance
- clutter and occlusion
- function versus morphology

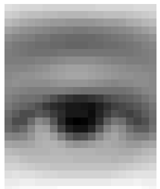
Historical Perspective:

- Blocks world
- 3D shape and part decomposition
- Perceptual organization
- Appearance-based models
- Context (3D and 2D) and Parsing

Let's begin by considering the linear subspace model (aka eigen-model) for appearance variations.

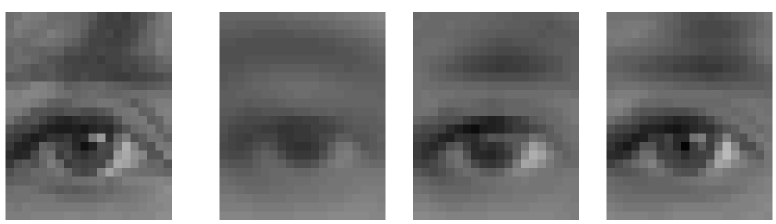
Subspace Models for Detection

Mean Eye:

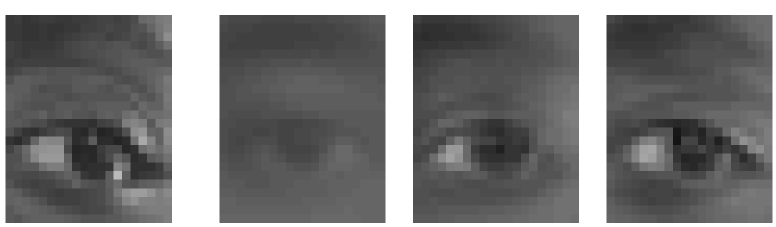


Basis Images $(1\!-\!6, \;\; \text{and} \;\; 10\!:\!5\!:\!35)$**:**

Reconstructions (for $K = 5, 20, 50$)**:**



Eye Image | Reconstruction (K = 5) | Reconstruction (K = 20) | Reconstruction (K = 50)



Eye Image | Reconstruction (K = 5) | Reconstruction (K = 20) | Reconstruction (K = 50)

Subspace Models for Detection

Generative model, $\mathcal{M}$, for random eye images:

$$\vec{\mathbf{I}} \; = \; \vec{\mathbf{m}} + \left(\sum_{k=1}^{K} a_k \vec{\mathbf{b}}_k \right) \; + \; \vec{\mathbf{e}}$$

where $\vec{\mathbf{m}}$ is the mean eye image, $a_k \sim \mathcal{N}(0, \sigma_k^2)$, σ_k^2 is the sample variance associated with the k^{th} principal direction in the training data, and $\vec{\mathbf{e}} \sim \mathcal{N}(0, \sigma_e^2 \, \mathbf{I}_{N^2})$ where $\sigma_e^2 = \frac{1}{N^2} \sum_{k=K+1}^{N^2} \sigma_k^2$ is the per pixel out-of-subspace variance. (The coefficients and errors are assumed to be independent.)

Random Eye Images:

Random draws from generative model (with K = 5, 10, 20, 50, 100, 200)

So the likelihood of an image under this model of eyes is

$$p(\vec{\mathbf{I}} \,|\, \mathcal{M}) \; = \; \left(\prod_{k=1}^{K} p(a_k | \mathcal{M}) \right) \; p(\vec{\mathbf{e}} \,|\, \mathcal{M})$$

where

$$p(a_k | \mathcal{M}) \; = \; \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{a_k^2}{2\sigma_k^2}} \quad , \qquad p(\vec{\mathbf{e}} \,|\, \mathcal{M}) = \prod_{j=1}^{N^2} \frac{1}{\sqrt{2\pi}\sigma_e} e^{-\frac{e_j^2}{2\sigma_e^2}} \quad .$$

Eye Detection

The log likelihood of the model is given by

$$
\begin{aligned}
L(\mathcal{M}) \;\equiv\; \log p(\vec{\mathbf{I}}\,|\,\mathcal{M}) \;&=\; \left(\sum_{k=1}^{K} \log p(a_k|\mathcal{M})\right) \;+\; \log p(\vec{\mathbf{e}}\,|\,\mathcal{M}) \\
&=\; \left(\sum_{k=1}^{K} \frac{-a_k^2}{2\sigma_k^2}\right) \;+\; \left(\sum_{j=1}^{N^2} \frac{-e_j^2}{2\sigma_e^2}\right) \;+\; const \\
&\equiv\; S_{in}(\vec{\mathbf{a}}) \;+\; S_{out}(\vec{\mathbf{e}}) \;+\; const
\end{aligned}
$$

Detector:

1. Given an image $\vec{\mathbf{I}}$
2. Compute the subspace coefficients $\vec{\mathbf{a}} = \mathbf{B}^T(\vec{\mathbf{I}} - \vec{\mathbf{m}})$
3. Compute residual $\vec{\mathbf{e}} = \vec{\mathbf{I}} - \vec{\mathbf{m}} - \mathbf{B}\vec{\mathbf{a}}$
4. For $S(\vec{\mathbf{a}}, \vec{\mathbf{e}}) = S_{in}(\vec{\mathbf{a}}) + S_{out}(\vec{\mathbf{e}})$, and a given threshold τ, the image patch is classified as an eye when

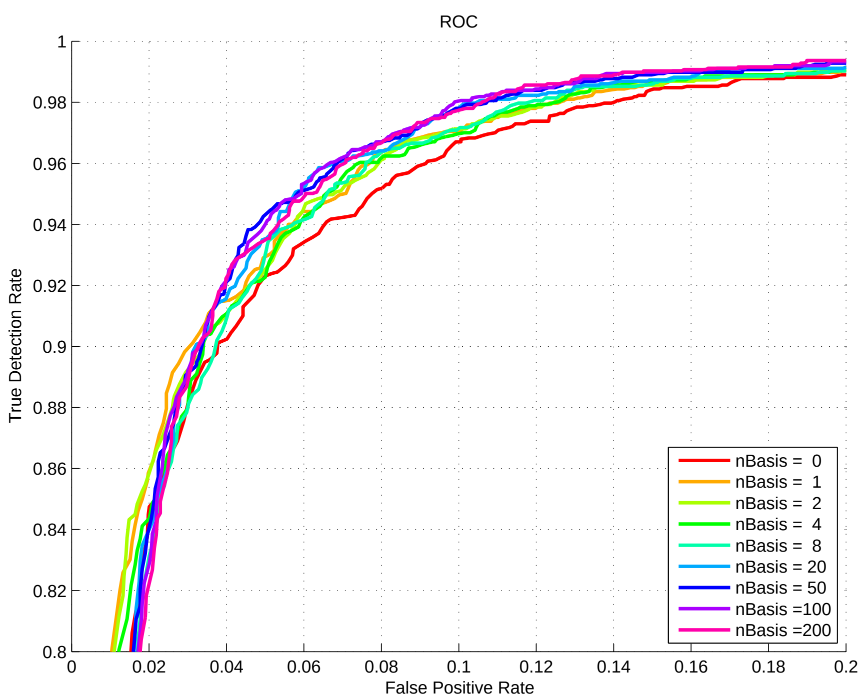
$$S(\vec{\mathbf{a}}, \vec{\mathbf{e}}) \;>\; \tau\;.$$

Performance Measures

	classified positives	classified negatives	
true examples	T_{pos} true positives	F_{neg} false negatives	$N_{pos} = T_{pos} + F_{neg}$
false examples	F_{pos} false positives	T_{neg} true negatives	$N_{neg} = F_{pos} + T_{neg}$
	C_{pos}	C_{neg}	N

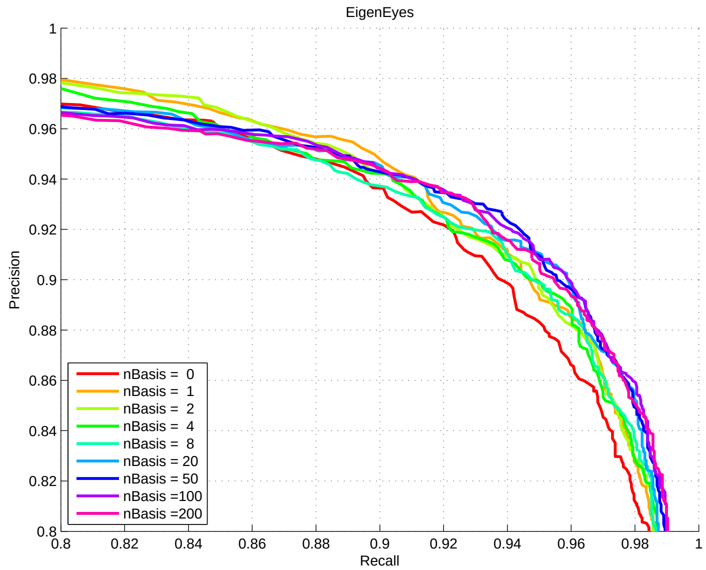
- true positive (detection) rate: $\rho_{tp} = T_{pos}/N_{pos}$
- true negative (reject) rate: $\rho_{tn} = T_{neg}/N_{neg}$
- false positive (false alarm / type I error) rate: $\rho_{fp} = 1 - \rho_{tn}$
- false negative (miss / type II error) rate: $\rho_{fn} = 1 - \rho_{tp}$

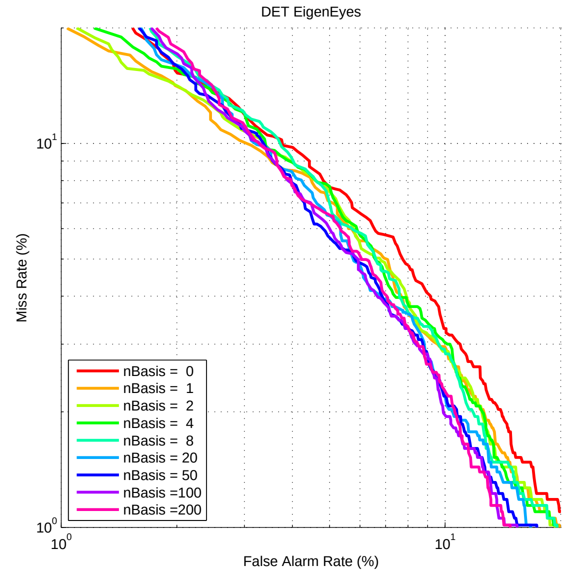
Receiver Operating Characteristic (ROC) Curves: trade-off between sensitivity (detection rate) and specificity (false positive rate), as a function of the decision threshold.



Disjoint test and training sets; non-eyes drawn at random from images. Notice the over-fitting at low FP rates.

Performance Measures (cont)





Detection is often done by sliding boxes of multiple sizes over an image, and testing each box for the presence of the target object. Here we should expect a large ratio of negative examples to positive examples, $r = N_{neg}/N_{pos}$. (E.g., sliding a single box over a 640×480 image, subsampling by 2 pixels, gives about 10^4 boxes. Since all but a handful are negatives, $r \approx 10^4$.)

Precision-Recall Curves (LEFT)

- Precision: T_{pos}/C_{pos} . What fraction of positive responses are correct hits?
- Recall: $\rho_{tp} = T_{pos}/N_{pos}$. What fraction of the true eyes do we actually find?
- Note: Beware of test sets with too few negatives, thereby biasing precision upwards (see below). The above-left plot used $r \approx 1.5$.

Detection Error Trade-off (DET) Curves (RIGHT)

- Miss rate (i.e., false negative rate, ρ_{fn}) versus false alarm rate (i.e., false positive rate, ρ_{fp}).
- Log-log axes highlight the important regime of small false negative and positive rates.
- For a particular application with an estimated ratio of $r = N_{neg}/N_{pos}$, the precision is

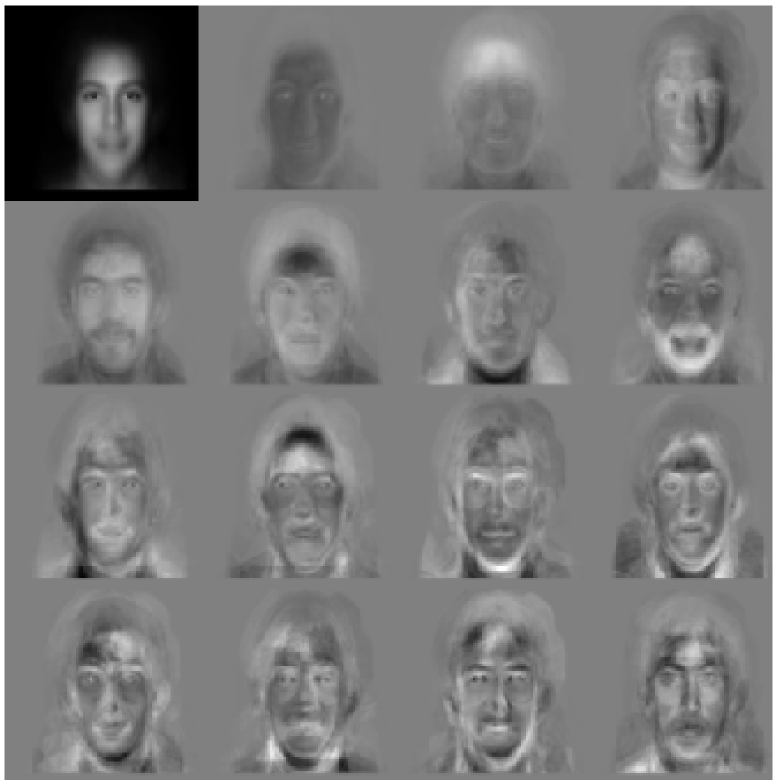
$$P \equiv T_{pos}/C_{pos} = \frac{(1-\rho_{fn})N_{pos}}{(1-\rho_{fn})N_{pos} + \rho_{fp}N_{neg}} = \frac{(1-\rho_{fn})}{(1-\rho_{fn}) + r\rho_{fp}}.$$

  Therefore the precision decreases as the ratio, r, increases.
- With ρ_{fn} and ρ_{fp} about 5% (see ROC or DET plots above) and $r \approx 10^4$, we find the precision $P \approx 0.002$ (i.e., of every 1000 hits, roughly 2 are expected to be eyes – this is exceptionally noisy). This motivates the reduction of ρ_{fp} by several orders of magnitude.

Face Detection

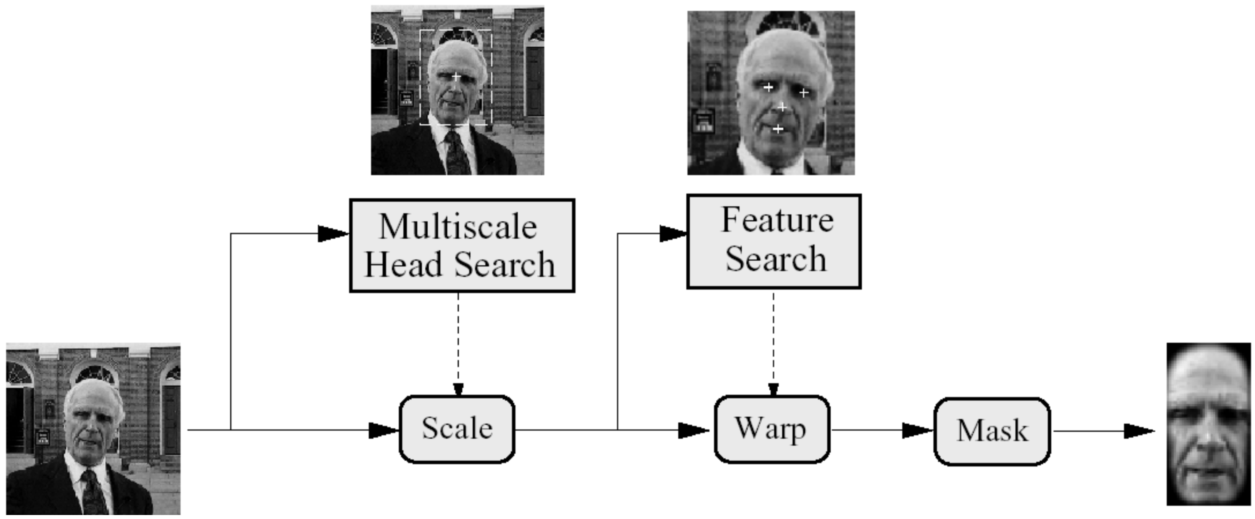
The wide-spread use of PCA for object recognition began with the work Turk and Pentland (1991) for face detection and recognition.

Shown below is the model learned from a collection of frontal faces, normalized for contrast, scale, and orientation, with the backgrounds removed prior to PCA.



Here are the mean image (upper-left) and the first 15 eigen-images. The first three show strong variations caused by illumination. The next few appear to correspond to the occurrence of certain features (hair, hairline, beard, clothing, etc).

Face Detection/Recognition



Moghaddam, Jebara and Pentland (2000): Subspace methods are used for head detection and then feature detection to normalize (warp) the facial region of the image.

Recognition: Are these two images (test and target) the same?

Approach 1: *Single Image Subspace Recognition:*

Project test and target faces onto the face subspace, and look at distance within the subspace.

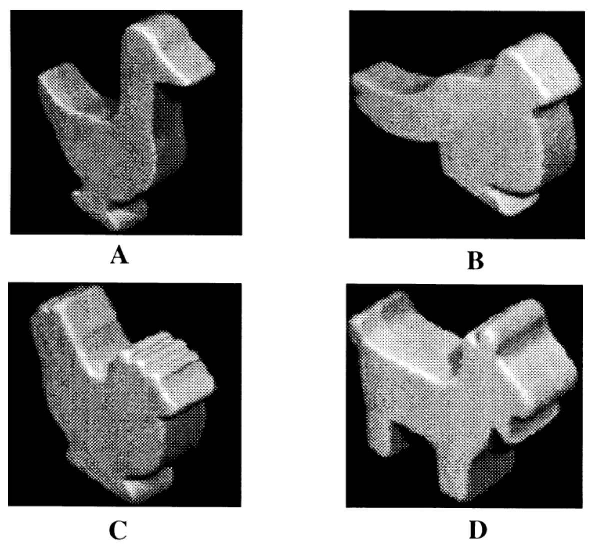
Approach 2: *Intra/Extra-Personal Subspace Recognition:*

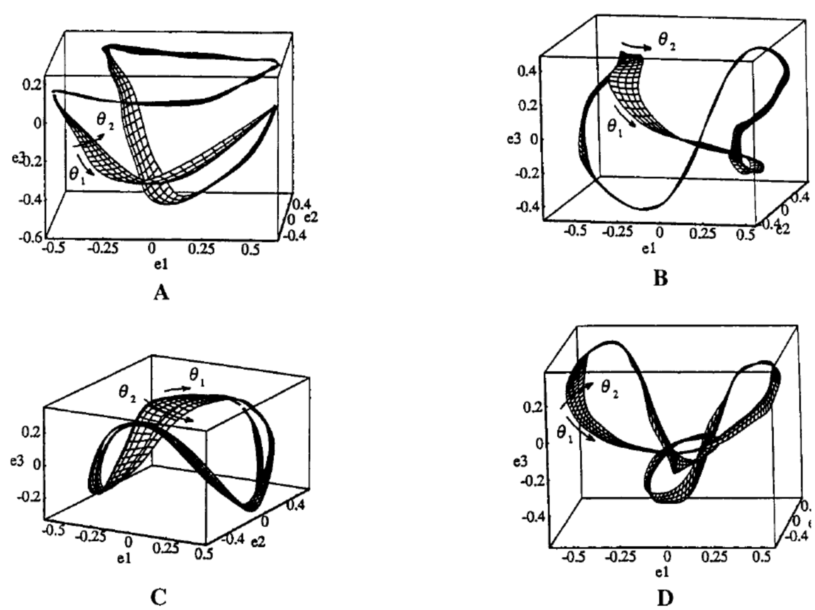
- An intra-personal subspace is learned from difference images of the same persion under variation in lighting and expression.
- The extra-personal subspace learned from difference between images of different people under similar conditions.

Object Recognition

Murase and Nayar (1995)

- images of multiple objects, taken from different positions on the viewsphere
- each object occupies a manifold in the subspace (as a function of position on the viewsphere)
- recognition: nearest neighbour assuming dense sampling of object pose variations in the training set.





Gaussian Class-Conditional Models

Consider Gaussian models for multiple classes (e.g., eyes and non-eyes). For model M_k, we assume a Gaussian observation density $p(\vec{\mathbf{d}} \,|\, M_k)$, e.g., over subspace coefficients $\vec{\mathbf{d}}$.

For two classes, M_1 and M_2, let the prior probabilities be $p(M_1)$ and $p(M_2) = 1 - p(M_1)$. The observation densities are Gaussian with means $\vec{\mu}_k$ and covariances C_k (for $k = 1, 2$). Then, the posterior probability for model M_k, given the data $\vec{\mathbf{d}}$, is

$$p(M_k \,|\, \vec{\mathbf{d}}) \;=\; \frac{p(M_k)\, G(\vec{\mathbf{d}};\, \vec{\mu}_k,\, C_k)}{p(\vec{\mathbf{d}})}.$$

The log odds $a(\vec{\mathbf{d}})$ for model M_1 over M_2 is defined to be

$$\begin{aligned} a(\vec{\mathbf{d}}) \;&\equiv\; \log\left[\frac{p(M_1 \,|\, \vec{\mathbf{d}}\,)}{p(M_2 \,|\, \vec{\mathbf{d}}\,)}\right] \\ &=\; \log\left[\frac{p(M_1)\; |C_2|^{1/2}}{p(M_2)\; |C_1|^{1/2}}\right] \;+ \\ &\quad \frac{1}{2}\left[(\vec{\mathbf{d}} - \vec{\mu}_2)^T C_2^{-1} (\vec{\mathbf{d}} - \vec{\mu}_2) - (\vec{\mathbf{d}} - \vec{\mu}_1)^T C_1^{-1} (\vec{\mathbf{d}} - \vec{\mu}_1)\right] \quad (1) \end{aligned}$$

Thresholding the log odds at zero yields the decision boundary.

The decision boundary is a quadratic surface in $\vec{\mathbf{d}}$ space (a quadratic discriminant). When both classes have the same covariance, i.e., $C_1 = C_2$, the quadratic terms in (1) cancel and the decision boundary becomes a hyperplane.

Logistic Regression

Let's return to the posterior class probability:

$$p(M_1 \,|\, \vec{\mathbf{d}}) = \frac{p(\vec{\mathbf{d}} \,|\, M_1)\, p(M_1)}{p(\vec{\mathbf{d}} \,|\, M_1)\, p(M_1) + p(\vec{\mathbf{d}}|M_2)\, p(M_2)} \; . \tag{2}$$

Dividing the numerator and denominator by $p(\vec{\mathbf{d}} \,|\, M_1)p(M_1)$ gives:

$$p(M_1 \,|\, \vec{\mathbf{d}}) \;=\; \frac{1}{1 + e^{-a(\vec{\mathbf{d}})}} \;\; , \quad a(\vec{\mathbf{d}}) = \ln \frac{p(\vec{\mathbf{d}} \,|\, M_1)\, p(M_1)}{p(\vec{\mathbf{d}} \,|\, M_2)\, p(M_2)} \; . \tag{3}$$

The posterior probability of M_1 grows as a grows, and when $a = 0$, the posterior is $P(M_1 \,|\, \vec{\mathbf{d}}) = \frac{1}{2}$. That is, $a(\vec{\mathbf{d}}) = 0$ is the decision boundary.

Let's assume a linear decision boundary (independent of any specific parametric form for the observation densities); i.e., let

$$a(\vec{\mathbf{d}}) \;=\; \vec{\mathbf{w}}^T \vec{\mathbf{d}} + b \tag{4}$$

To learn a classifier, given IID training exemplars, $\{\vec{\mathbf{d}}_j, y_j\}$, where $y_j = \{1, 2\}$, we minimize the negative log likliehood:

$$\begin{aligned} \log p(\{\vec{\mathbf{d}}_j, y_j\} \,|\, \mathbf{w}, b) \;&\propto\; p(\{y_j\} \,|\, \{\vec{\mathbf{d}}_j\}, \mathbf{w}, b) \\ &= \sum_{j:y_j=1} p(M_1 \,|\, \vec{\mathbf{d}}_j) \sum_{j:y_j=2} (1 - p(M_1 \,|\, \vec{\mathbf{d}}_j)) \end{aligned} \tag{5}$$

Although this objective function cannot be optimized in closed-form, it is convex; it has a single minimum. So we can optimize it with some form of gradient descent, and the initial guess is not critical.

Issues with Class-Conditional and LR Models

Class-Conditional Models:

- The single Gaussian model is often rather crude. PCA coefficients often exhibit significantly more structure (cf. Murase & Nayar).
- A Gaussian model will also be a poor model of non-eye images.
- As a result of this unmodelled structure, detectors based on single Gaussian models are often poor.

Logistic Regression:

- Discriminative model does not require a model of the observations, and often has *fewer* parameters as a result.
- LR with its linear decision boundary is only expressive enough for simple problems.

Alternatives:

- An alternative approach is to consider warped and aligned view based models (see Cootes, Edwards, & Taylor, 1998).
- Richer density models of the subspace coefficients are possible (e.g., nearest neighbour as in Murase & Nayar, or mixture models).

Breakthrough:

- More sophisticated discriminative models with simple (fast) feature extraction (see Viola & Jones, 2004).

AdaBoost: Binary Classification Problem

Given training data $\{\vec{x}_j, y_j\}_{j=1}^N$, where

- $\vec{x}_j \in \mathbb{R}^d$ is the feature vector for the j^{th} data item,
- $y_j \in \{-1, 1\}$ denotes the class membership of the j^{th} item $\vec{x}_j$,

we seek a classifier $F(\vec{x})$ such that $y(x) \equiv \mathrm{sign}(F(\vec{x}))$ approximates (in some sense) the training data; i.e., the given class indicator y_j should agree with the model $\mathrm{sign}(F(\vec{x}_j))$ as much as possible.

AdaBoost is an algorithm for greedily training classifiers $F(\vec{x}\ )$ which take the form of *additive linear models*:

$$\begin{aligned} F_m(\vec{x}) &= \sum_{k=1}^{m} \alpha_k f_k(\vec{x}; \vec{\theta}_k) \qquad (6) \\ &= F_{m-1}(\vec{x}) + \alpha_m f_m(\vec{x}; \vec{\theta}_m). \end{aligned}$$

Here $m \geq 1$ and

- $F_m(\vec{x})$ is a weighted (i.e. α_k) sum of simpler functions $f_k(\vec{x}; \vec{\theta}_k)$.
- Note the simpler functions depend on parameters $\vec{\theta}_k$, which we need to fit along with the weights α_k.
- Here we take the simpler functions $f_k(\vec{x}; \vec{\theta}_k)$ to be weak classifiers, providing values in $\{-1, 1\}$ (e.g., decision stumps).
- We use $F_0(\vec{x}) \equiv 0$ in the recursive definition above.

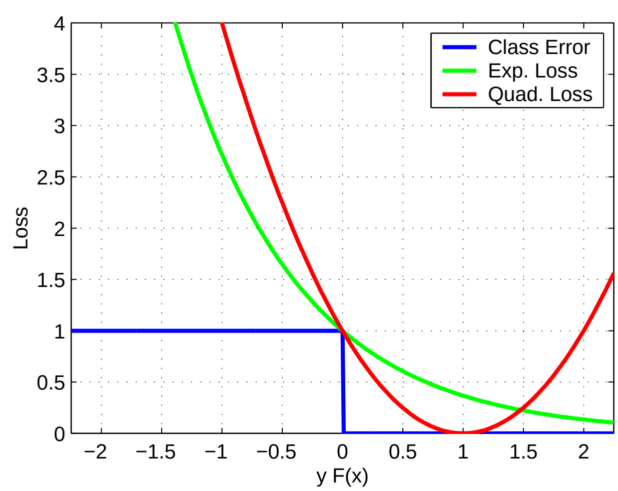
Exponential Loss

We seek a model $F_m(\vec{x})$ such that $\text{sign}(F_m(\vec{x}_k))$ agrees with the class indicator $y_j \in \{-1, 1\}$, as much as possible, in the training data.

How should we measure agreement/disagreement? Since y_k should have the same sign as $F_m(\vec{x}_k)$, it is convenient to consider $y_k F_m(\vec{x}_k)$, which should be positive.

Loss (cost) functions of $z \equiv yF(\vec{x})$:

- **0-1 Loss (classification error):** $C(z) = 1$ if $z \leq 0$, else 0. This loss function is hard to optimize because it is discontinuous.
- **Quadratic Loss:** $C(z) = (z-1)^2$. Easy to optimize but penalizes $F(\vec{x})$ when it's large with the correct sign (confident and correct).
- **Exponential Loss:** $C(z) = exp(-z)$. Smooth and monotonic in z. Large cost for $F(\vec{x})$ with wrong sign and large magnitude (i.e. confident and wrong). Still a crude approximation to 0-1 loss.



Greedy Fitting and AdaBoost

Suppose we have trained a classifier $F_{m-1}(\vec{x})$ with $m-1$ additive components, and we wish to add one more component, i.e.,

$$F_m(\vec{x}) \; = \; F_{m-1}(\vec{x}) + \alpha_m \, f_m(\vec{x}; \vec{\theta}_m) \, .$$

Suppose we choose α_m and $\vec{\theta}_m$ to minimize the exponential loss

$$\begin{aligned}
\sum_{j=1}^{N} C(y_j F_m(\vec{x}_j)) \; &\equiv \; \sum_{j=1}^{N} e^{-y_j F_m(\vec{x}_j)} \\
&= \; \sum_{j=1}^{N} e^{-y_j F_{m-1}(\vec{x}_j)} e^{-y_j \alpha_m f_m(\vec{x}_j, \vec{\theta}_m)} \\
&= \; \sum_{j=1}^{N} w_j^{(m-1)} e^{-y_j \alpha_m f_m(\vec{x}_j, \vec{\theta}_m)}
\end{aligned}$$

Here the weight $w_j^{(m-1)} = e^{-y_j F_{m-1}(\vec{x}_j)}$ is just the exponential loss for the previous function $F_{m-1}(\vec{x})$ on the j^{th} training item.

- The weights are largest for data points which the previous function $F_{m-1}(\vec{x})$ confidently classifies incorrectly, i.e., $y_j F_{m-1}(\vec{x}_j) \ll 0$.
- The weights are smallest for points confidently classified correctly, i.e., for $y_j F_{m-1}(\vec{x}_j) \gg 0$.

This greedy fitting of the weak classifiers in an additive model leads to the AdaBoost learning algorithm (see Friedman et al, 2000).

AdaBoost Algorithm

for all training exemplars: $j = 1...N,\ w_j^{(1)} = 1$

for $m = 1$ to M **do**

Fit weak classifier m to minimize the objective function:

$$\epsilon_m = \frac{\sum_j w_j^{(m)} I(f_m(\vec{x}_j, \vec{\theta}_m) \neq y_j)}{\sum_j w_j^{(m)}}$$

where $I(b) = 1$ if boolean b is true, and 0 otherwise

$\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$

for all i **do**

$$w_j^{(m+1)} = w_j^{(m)} e^{\alpha_m I(f_m(\vec{x}_j) \neq y_j)}$$

end for

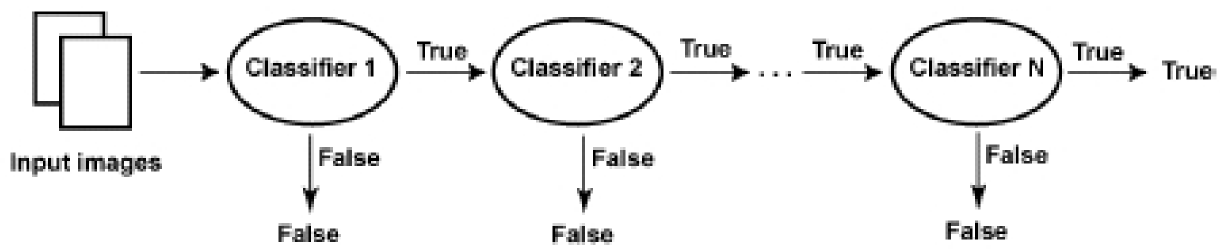
end for

After learning, the final classifier is

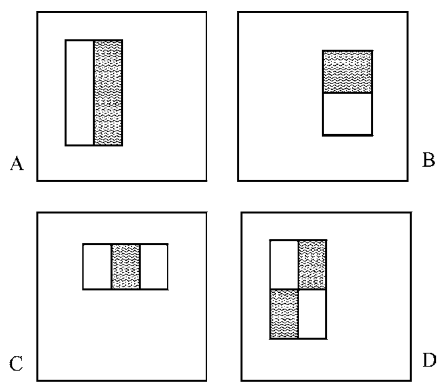
$$g(\vec{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m f_m(\vec{x}, \vec{\theta}_m)\right) \tag{7}$$

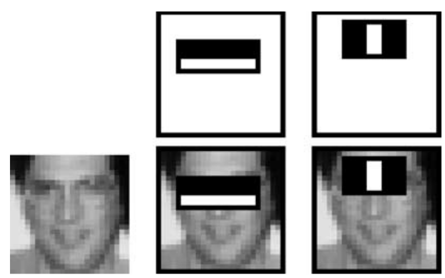
Viola and Jones Face Detector

Rejection cascade architecture (sequence of classifiers with thresholds chosen to keep the false negative rate low):

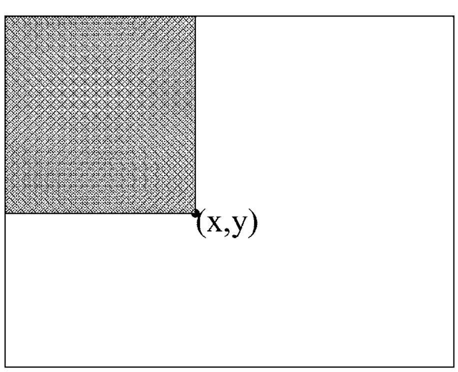


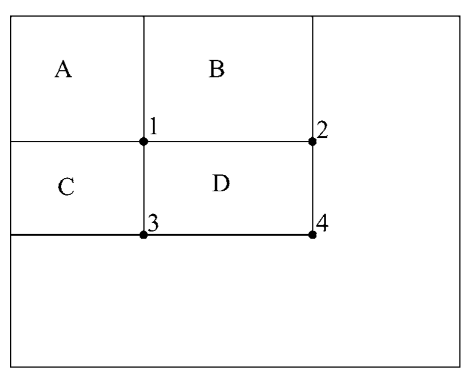
Features are formed from Haar filters...





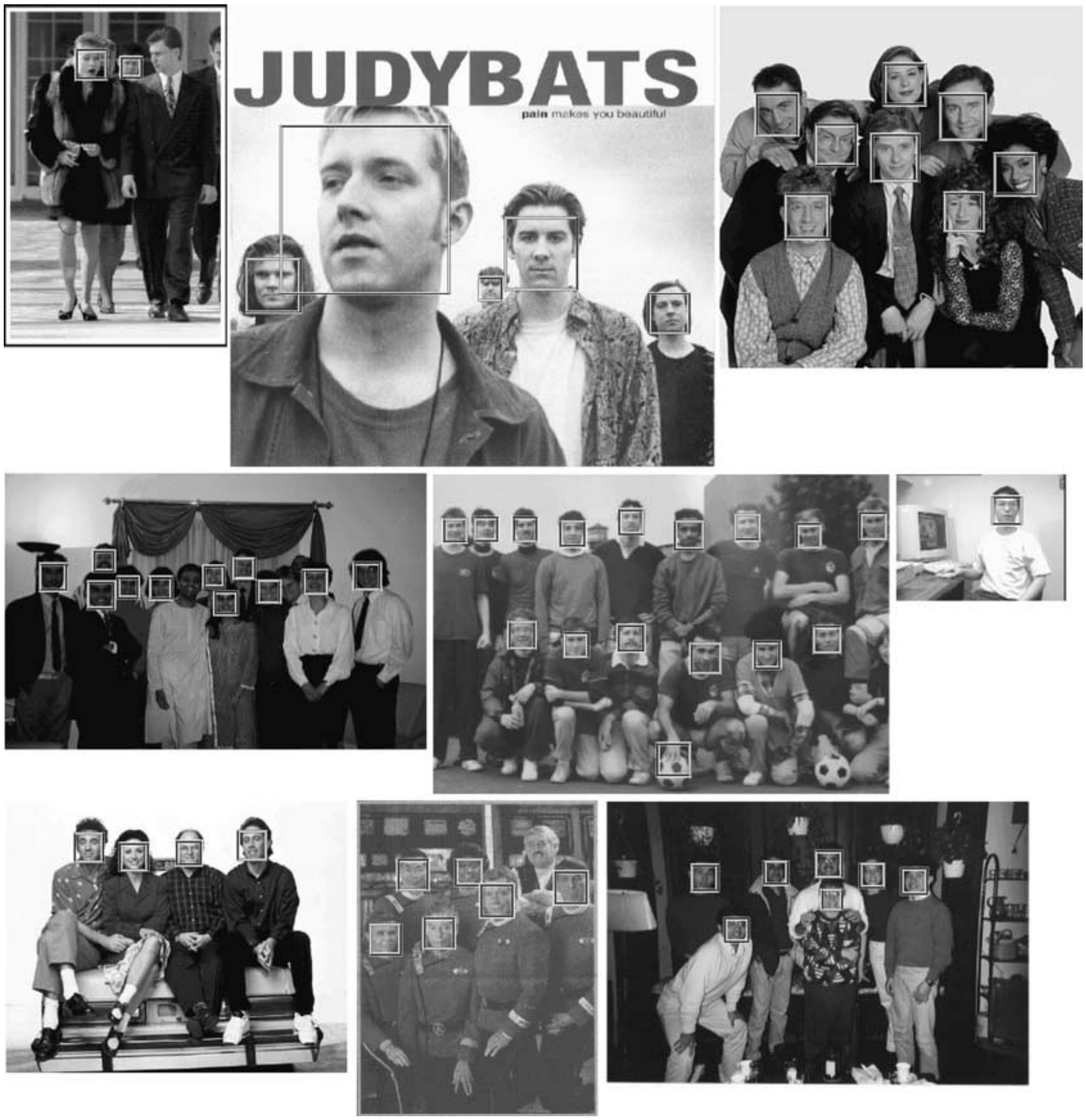
These features can be computed rapidly using integral images.





The result is a real-time face detector with good classification performance (Viola and Jones, 2004).

Viola and Jones, Results



Feature-Based Near Duplicate Detection

Sufficiently similar images will contain similar features, occurring in similar spatial configurations.

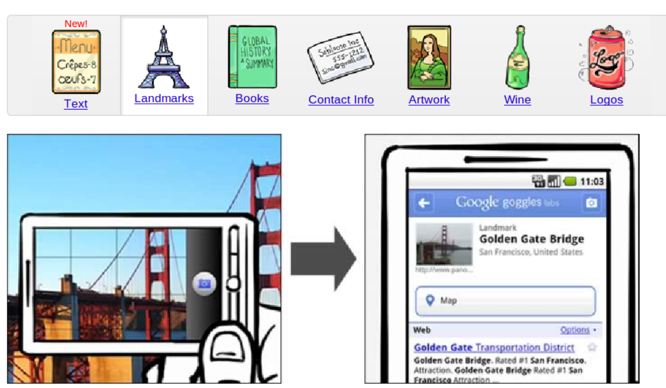
Training: Extract SIFT descriptors from training image(s).

Testing:

- For each SIFT feature in test image, find a match from training features (ANN search with good near neighbour distance ratio).
- Select training images with sufficiently many matching features
- Robustly fit a parametric warp (eg, affine), and rank selected images based on the number of inliers.

Applications:

- Detecting / tracking images of same scene/object with small variations in viewpoint, occlusion, and lighting (eg, see [Lowe 2004])
- Image retrieval – eg, Google Goggles searches 10^8 images, with 10^3 features/image and a distributed KD-tree for ANN search.



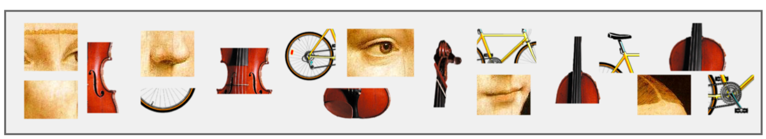
Works surprisingly well on specific types of images.

Bag of Words

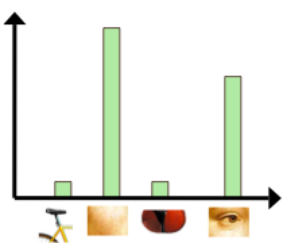
Feature-based approach to *category* recognition (and unsupervised discovery), modeling appearance by first-order feature statistics (the frequency of feature occurrence), thereby ignoring spatial layout and higher-order statistics.

Recognition:

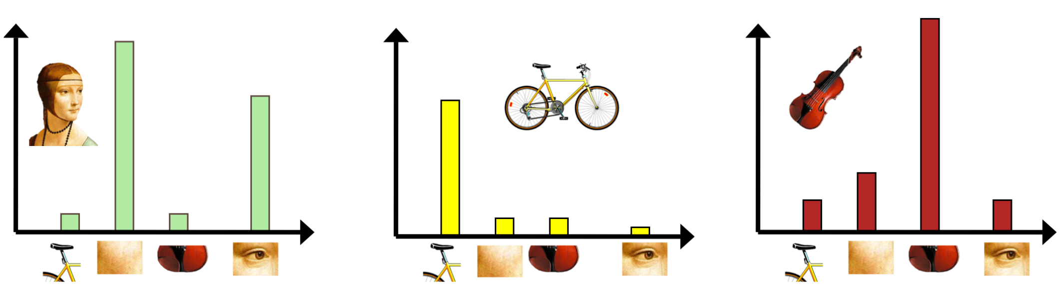
1. feature detection



2. compute distribution (histogram) of feature occurrence



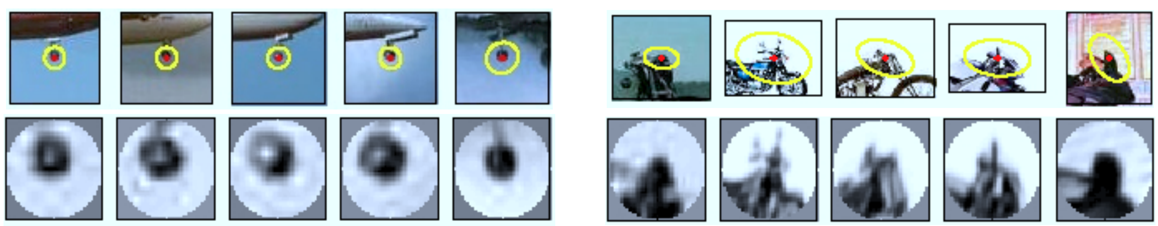
3. matching (generative or discriminative)



Bag of Words: Feature-Based Representation

Feature Detection: Harris/DOG maxima, regular grid or random patches, maximally stable extremal regions (MSER) [Matas et al 2002], etc.

Canonical Coordinates: for affine region detectors, warp elliptical regions to circular disks (for robustness to viewpoint, see [Mikolajczyk et al 2005; Sivic et al 2005])



Local Descriptor: patch descriptor, such as SIFT, w or w/o rotation

Codebook formation using vector quantization. This simplifies the feature space, and provides robustness intra-class variability.

- *K-Means:* Given N data vectors $\{\vec{\mathbf{y}}_i\}_{i=1}^N$, assign each vector to one of K disjoint clusters; let l_{ij} be 1 when $\vec{\mathbf{y}}_i$ belongs to cluster j and 0 otherwise. Find cluster centers $\vec{\mu}_j$ and assignments l_{ij} to minimize

$$E = \sum_{i,j} l_{ij} \, ||\vec{\mathbf{y}}_i - \vec{\mathbf{c}}_j||^2 \, , \quad \text{s.t.} \sum_{j=1}^{K} l_{ij} = 1 \, .$$

Generative Bag-of-Words Models

Naive Bayes: Class-conditional models of the frequency of visual words $\{w_j\}_{j=1}^{M}$, for classes $\{c_k\}_{k=1}^{K}$. (For unsupervised learning, cluster the empirical word distributions for a set of training images.)

Inference:

$$c^* = \arg\max \, p(w_{1:M} \,|\, c) \, p(c) \, , \quad p(w_{1:M} \,|\, c) = \prod_{j=1}^{M} p(w_j \,|\, c)$$

Latent Topic Models: Low-dimensional latent models for category *discovery*. In Probabilistic Latent Semantic Analysis, for image d and visual word w we define K latent topic models $\{c_k\}_{k=1}^{K}$ for which

$$p(w \,|\, d) \; = \; \sum_{k=1}^{K} p(w \,|\, c_k) \, p(c_k \,|\, d) \tag{8}$$

Learning: Estimate $p(w \,|\, c_k)$ and $p(c_k \,|\, d)$ using EM to maximize the data likelihood, given images $\{d_i\}_{i=1}^{N}$, and visual words $\{w_j\}_{j=1}^{M}$:

$$L \; = \; \sum_{i=1}^{N} \sum_{j=1}^{M} p(w_j \,|\, d_i)^{n(w_j, d_i)} \tag{9}$$

where $n(w_j, d_i)$ is the number of times word w_j occurs in image d_i.

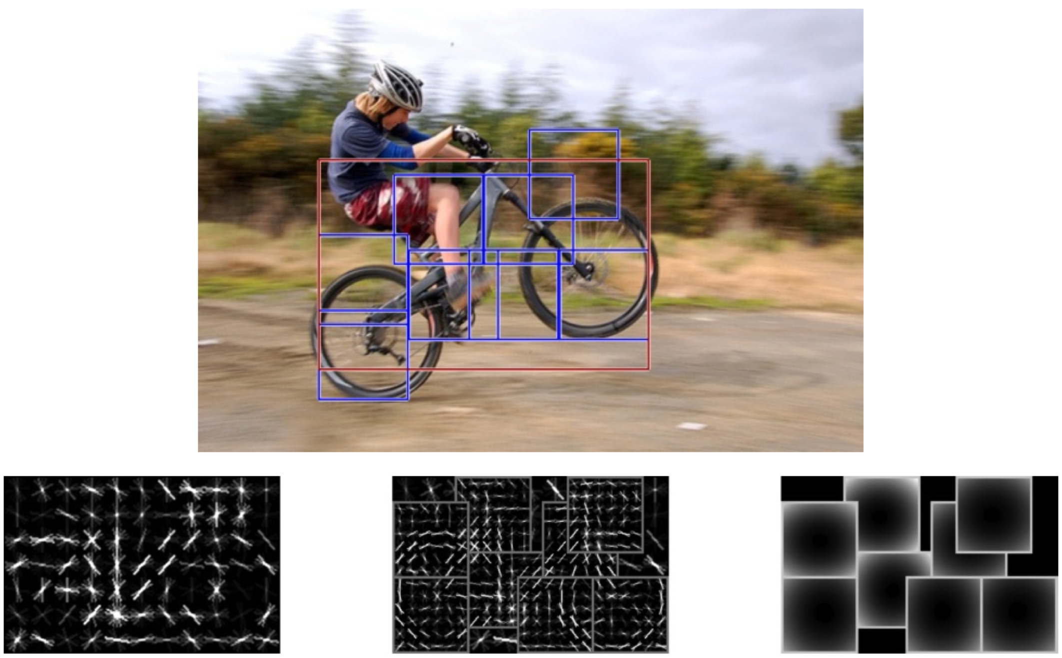
Inference: Given a test image d and word counts $n(w_j, d)$, consider the data likelihood (9) with $N = 1$. The terms $p(w_j \,|\, d)$ can be expanded using (8), with the learned values for $p(w_j \,|\, c_k)$. Then EM can be used to infer the topic distribution $p(c_k \,|\, d)$. [Sivic et al ’05; Hofmann ’01]

And of course there's more ...

Kernel methods for discriminative classification, e.g., with the pyramid match kernel [Grauman & Darrell, 2005].

Discriminative methods with spatial structure, e.g., using spatial pyramid kernels [Lazebnik et al, 2009]

Part-based deformable models, e.g. [Felzenswalb et al, 2010]



and more ...

Further Readings

P. Belhumeur et al (1997) Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE PAMI*, 19(7):711-720

T. Cootes, G. Edwards, and C.J. Taylor (1998) Active appearance models, *Proc. ECCV*.

S. Dickinson (1999) The evolution of object categorization and the challenge of image abstraction.

P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan. Object detection with discriminatively trained part based models. *IEEE PAMI*, 32

J. Friedman, T. Hastie, and R. Tibshirani, Additive logistic regression: a statistical view of boosting, *Ann. Statistics* 28, 2000, pp. 337-407.

G. Golub and C. van Loan (1984) *Matrix Computations*. Johns Hopkins Press, Baltimore.

K. Grauman and T. Darrell (2007) The Pyramid Match Kernel: Efficient learning with sets of features. *JMLR* 8: 725–760.

T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*, Springer, 2001.

T. Hofmann (2001) Unsupervised learning by probabilistic latent semantic analysis. *MLJ* 42:177–196

S. Lazebnik, C. Schmid, and J. Ponce (2009) Spatial pyramid matching. in *Object Categorization: Computer and Human Vision Perspectives*, S. Dickinson et al (eds), Cambridge University Press

J. Matas, O. Chum, M. Urba, and T. Pajdla (2002) Robust wide baseline stereo from maximally stable extremal regions." *Proc BMVC*, pp. 384-396.

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, T. Kadir and L. Van Gool (2005) A Comparison of Affine Region Detectors. *IJCV* 65(12):43–72.

B. Moghaddam, T. Jebara, T. and A. Pentland, A. (2000) Bayesian face recognition. *Pattern Recognition*, 33(11):1771-1782

H. Murase and S. Nayar, S. (1995) Visual learning and recognition of 3D objects from appearance. *IJCV* 14:5–24.

J. Sivic, B. Russel, A. Efros, A. Zeisserman, W. Freeman (2005) Discovering objects and their location in images. *Proc ICCV*

M. Turk and A. Pendland (1991) Face recognition using eigenfaces, *J. Cog. Neurosci.*, 3(1):71–86.

P. Viola, and M. Jones (2004) Robust real-time face detection, *IJCV* 58:137–154.