Appearance-based recognition

Kristen Grauman

UT-Austin

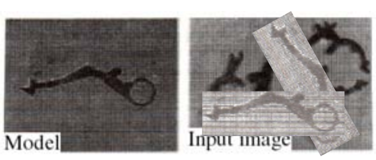
Thursday, Nov 6

Today

- Review: alignment-based recognition
- Appearance-based recognition
  - Classification
    - Skin color detection example
  - Sliding window detection
    - Face detection example

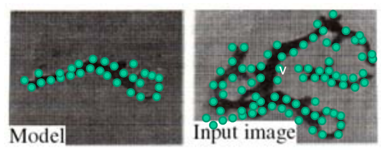
Hypothesize and test: main idea

- Given model of object
- New image: hypothesize object identity and pose
- Render object in camera
- Compare rendering to actual image: if close, good hypothesis.



How to form a hypothesis?

All possible assignments of model features to image features?

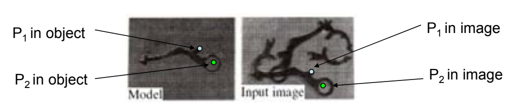


Pose consistency / alignment

- Key idea:
  - If we find good correspondences for a small set of features, it is easy to obtain correspondences for a much larger set.
- Strategy:
  - Generate hypothesis transformation using small numbers of correspondences
  - Backproject: Transform *all* model features to image features
  - Verify: see if for this alignment the model and image agree

Example: 2d affine mappings

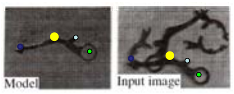
- Say camera is looking down perpendicularly on planar surface



- We have two coordinate systems (object and image), and they are related by some affine mapping (rotation, scale, translation, shear).

Alignment: backprojection

- Having solved for this transformation from some number of detected matches (3+ here), can compute (hypothesized) location of any *other* model point in the image space.



$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- Verify, e.g., based on edge agreement

Issue with hypothesis & test alignment approach

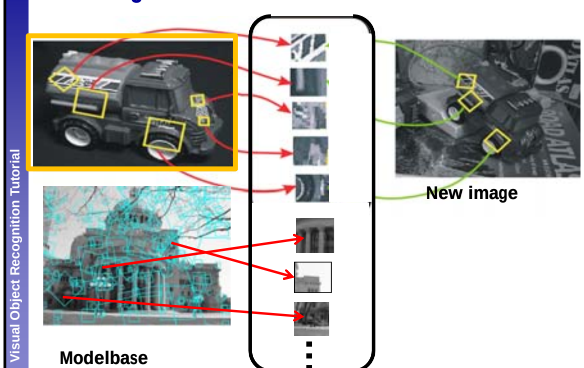
- May have false matches
  - We want *reliable* features to form the matches
    - **Local invariant features** useful to find matches, and to verify hypothesis
- May be too many hypotheses to consider
  - We want to look at the *most likely* hypotheses first
    - **Pose clustering (i.e., voting):** Narrow down number of hypotheses to verify by letting features *vote* on model parameters.

Pose clustering and verification with SIFT [Lowe]

To detect **instances** of objects from a model base:

1) Index descriptors (distinctive features narrow possible matches)

Indexing local features



Visual Object Recognition Tutorial

Pose clustering and verification with SIFT [Lowe]

To detect **instances** of objects from a model base:

1) Index descriptors (distinctive features narrow possible matches)
2) Generalized Hough transform to vote for poses (keypoints have record of parameters relative to model coordinate system)
3) Affine fit to check for agreement between model and image features (fit and verify using features from Hough bins with 3+ votes)

Planar objects

Model images and their SIFT keypoints

Input image

Model keypoints that were used to recognize, get least squares solution.

Recognition result

[Lowe]

3d objects

Background subtract for model boundaries

Objects recognized,

Recognition in spite of occlusion

[Lowe]

Recall: difficulties of voting

- Noise/clutter can lead to as many votes as true target
- Bin size for the accumulator array must be chosen carefully
- (Recall Hough Transform)

- In practice, good idea to make broad bins and spread votes to nearby bins, since verification stage can prune bad vote peaks.

Today

- Review: alignment-based recognition

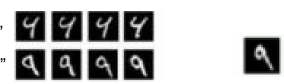
- Appearance-based recognition
  - Classification
    - Skin color detection example
  - Sliding window detection
    - Face detection example

Supervised classification

- Given a collection of *labeled* examples, come up with a function that will predict the labels of new examples.

“four”

“nine”



?

Training examples

Novel input

- How good is some function we come up with to do the classification?
- Depends on
  - Mistakes made
  - Cost associated with the mistakes

Supervised classification

- Given a collection of *labeled* examples, come up with a function that will predict the labels of new examples.

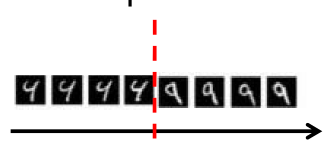
- Consider the two-class (binary) decision problem
  - $L(4 \rightarrow 9)$: Loss of classifying a 4 as a 9
  - $L(9 \rightarrow 4)$: Loss of classifying a 9 as a 4

- **Risk** of a classifier *s* is expected loss:

$$R(s) = \Pr(4 \rightarrow 9 \mid \text{using } s)L(4 \rightarrow 9) + \Pr(9 \rightarrow 4 \mid \text{using } s)L(9 \rightarrow 4)$$

- We want to choose a classifier so as to minimize this total risk

Supervised classification



Feature value **x**

Optimal classifier will minimize total risk.

At decision boundary, either choice of label yields same expected loss.

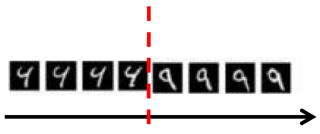
If we choose class “four” at boundary, expected loss is:

$$= P(\text{class is } 9 \mid \mathbf{x})\, L(9 \rightarrow 4) + P(\text{class is } 4 \mid \mathbf{x}) L(4 \rightarrow 4)$$

If we choose class “nine” at boundary, expected loss is:

$$= P(\text{class is } 4 \mid \mathbf{x})\, L(4 \rightarrow 9)$$

Supervised classification



Feature value **x**

Optimal classifier will minimize total risk.

At decision boundary, either choice of label yields same expected loss.

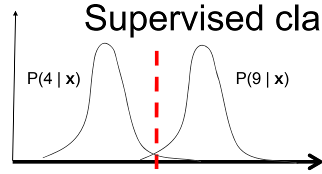
So, best decision boundary is at point **x** where

$$P(\text{class is } 9 \mid \mathbf{x})\, L(9 \to 4) = \text{P}(\text{class is } 4 \mid \mathbf{x}) L(4 \to 9)$$

To classify a new point, choose class with lowest expected loss; i.e., choose "four" if

$$P(4 \mid \mathbf{x}) L(4 \to 9) > P(9 \mid \mathbf{x}) L(9 \to 4)$$

Supervised classification



Feature value **x**

Optimal classifier will minimize total risk.

At decision boundary, either choice of label yields same expected loss.

So, best decision boundary is at point **x** where

$$P(\text{class is } 9 \mid \mathbf{x})\, L(9 \to 4) = \text{P}(\text{class is } 4 \mid \mathbf{x}) L(4 \to 9)$$

To classify a new point, choose class with lowest expected loss; i.e., choose "four" if

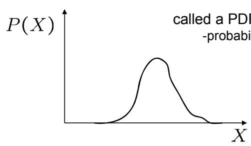
$$\boxed{P(4 \mid \mathbf{x})} L(4 \to 9) > \boxed{P(9 \mid \mathbf{x})} L(9 \to 4)$$

How to evaluate these probabilities?

Probability

Basic probability

- X is a random variable
- P(X) is the probability that X achieves a certain value



called a PDF
-probability distribution/density function

- $0 \le P(X) \le 1$
- $\int_{-\infty}^{\infty} P(X)dX = 1$ or $\sum P(X) = 1$

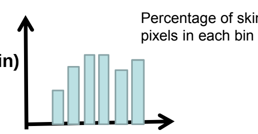
continuous X   discrete X

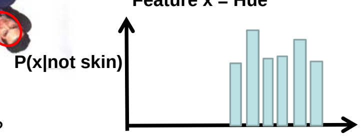
- Conditional probability: P(X | Y)
  - probability of X given that we already know Y

Source: Steve Seitz

Example: learning skin colors

- We can represent a class-conditional density using a histogram (a "non-parametric" distribution)

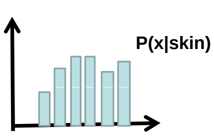


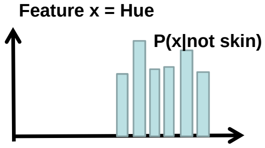


What if feature dimension is high?

Example: learning skin colors

- We can represent a class-conditional density using a histogram (a "non-parametric" distribution)





Now we get a new image, and want to label each pixel as skin or non-skin.

What's the probability we care about to do skin detection?

Bayes rule

$$\underbrace{P(skin \mid x)}_{\text{posterior}} = \frac{\underbrace{P(x \mid skin)}_{\text{likelihood}}\,\underbrace{P(skin)}_{\text{prior}}}{P(x)}$$

$$P(skin \mid x) \;\alpha\; P(x \mid skin) P(skin)$$

Where does the prior come from?

Example: classifying skin pixels

Now for every pixel in a new image, we can estimate probability that it is generated by skin.

Brighter pixels → higher probability of being skin

Figure from Gary Bradski

Classify pixels based on these probabilities

- 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

Example: classifying skin pixels

- Black=pixels classified as skin

Jones and Rehg, CVPR 1999.

Example: classifying skin pixels

Figure 6: A video image and its flesh probability image

Figure 7: Orientation of the flesh probability distribution marked on the source video image

Gary Bradski, 1998

Example: classifying skin pixels

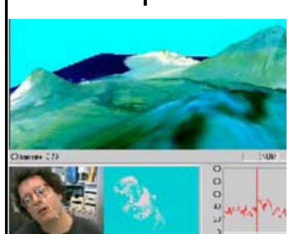


Figure 13: CAMSHIFT-based face tracker used to over a 3D graphic's model of Hawaii

Figure 12: CAMSHIFT-based face tracker used to play Quake 2 hands free by inserting control variables into the mouse queue

Using skin color-based face detection and pose estimation as a video-based interface

Gary Bradski, 1998

Today

- Review: alignment-based recognition
- Appearance-based recognition
  - Classification
    - Skin color detection example
  - Sliding window detection
    - Face detection example

Detection via classification: Main idea

Visual Object Recognition Tutorial

Basic component: a binary classifier

 → **Car/non-car Classifier**

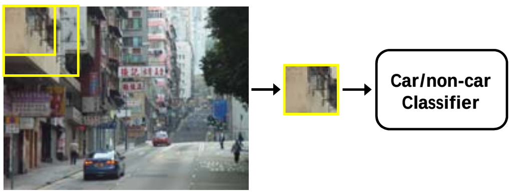
↓

No, not a car. / **Yes, car.**

K. Grauman, B. Leibe

Detection via classification: Main idea

If object may be in a cluttered scene, slide a window around looking for it.



(Essentially, our skin detector was doing this, with a window that was one pixel big.)

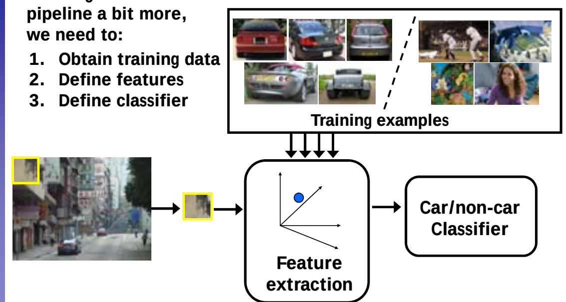
Visual Object Recognition Tutorial

K. Grauman, B. Leibe

Detection via classification: Main idea

Fleshing out this pipeline a bit more, we need to:

1. Obtain training data
2. Define features
3. Define classifier



Visual Object Recognition Tutorial

K. Grauman, B. Leibe

Detection via classification: Main idea

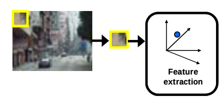
- Consider all subwindows in an image
  - Sample at multiple scales and positions (and orientations)
- Make a decision per window:
  - "Does this contain object category X or not?"

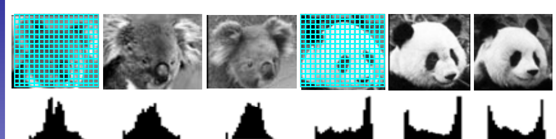
Visual Object Recognition Tutorial

K. Grauman, B. Leibe

33

Feature extraction: global appearance





Simple holistic descriptions of image content

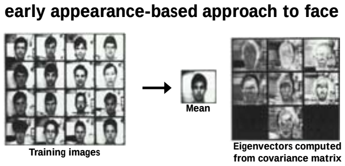
- grayscale / color histogram
- vector of pixel intensities

Visual Object Recognition Tutorial

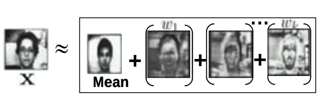
K. Grauman, B. Leibe

Eigenfaces: global appearance description

An early appearance-based approach to face recognition



Generate low-dimensional representation of appearance with a linear subspace.



Project new images to "face space".

Recognition via nearest neighbors in face space

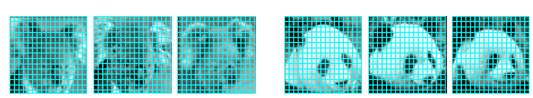
Turk & Pentland, 1991

Visual Object Recognition Tutorial

K. Grauman, B. Leibe

Feature extraction: global appearance

- Pixel-based representations sensitive to small shifts



- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation

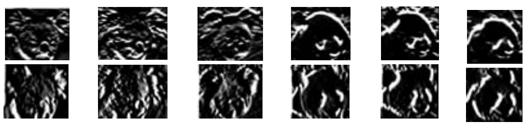
Cartoon example: an albino koala

Visual Object Recognition Tutorial

K. Grauman, B. Leibe

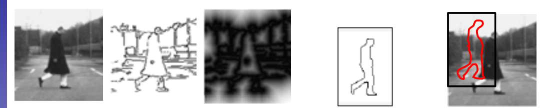
Gradient-based representations

- Consider edges, contours, and (oriented) intensity gradients



Gradient-based representations: Matching edge templates

- Example: Chamfer matching



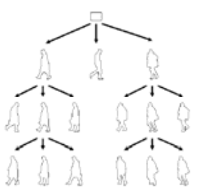
At each window position, compute average min distance between points on template (T) and input (I).

$$D_{chamfer}(T,I) \equiv \frac{1}{|T|}\sum_{t \in T} d_I(t)$$

Gavrila & Philomin ICCV 1999

Gradient-based representations: Matching edge templates

- Chamfer matching

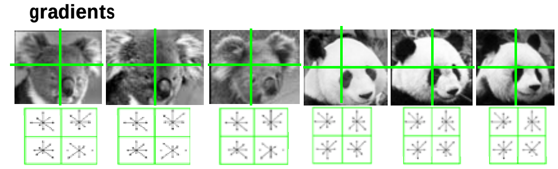


Hierarchy of templates

Gavrila & Philomin ICCV 1999

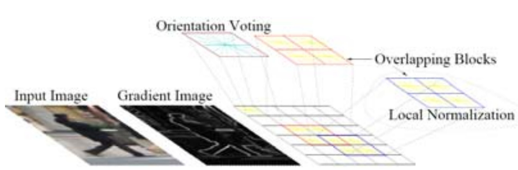
Gradient-based representations

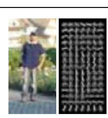
- Consider edges, contours, and (oriented) intensity gradients



- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination

Gradient-based representations: Histograms of oriented gradients (HoG)



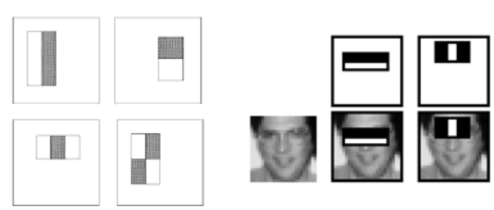


Map each grid cell in the input window to a histogram counting the gradients per orientation.

Code available: http://pascal.inrialpes.fr/soft/olt/

Dalal & Triggs, CVPR 2005

Gradient-based representations: Rectangular features



Compute differences between sums of pixels in rectangles

Captures contrast in adjacent spatial regions, efficient to compute

Each feature parameterized by scale, position, type.

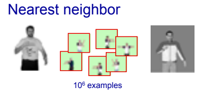
Viola & Jones, CVPR 2001

Classifier construction

- How to compute a decision for each subwindow?

Classifier construction: many choices…

Nearest neighbor

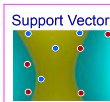


Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks

LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines

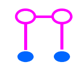


Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting

Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006,…

Conditional Random Fields



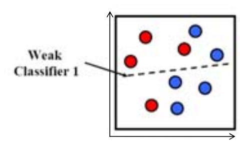
McCallum, Freitag, Pereira 2000; Kumar, Hebert 2003
…

Slide adapted from Antonio Torralba

Boosting

- Build a strong classifier by combining number of “weak classifiers”, which need only be better than chance
- Sequential learning process: at each iteration, add a weak classifier
- Flexible to choice of weak learner
  - including fast simple classifiers that alone may be inaccurate
- We’ll look at Freund & Schapire’s AdaBoost algorithm
  - Easy to implement
  - Base learning algorithm for Viola-Jones face detector

AdaBoost: Intuition



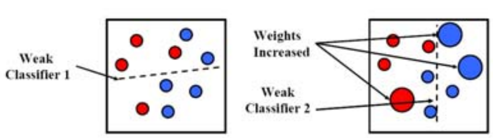
Consider a 2-d feature space with positive and negative examples.

Each weak classifier splits the training examples with at least 50% accuracy.

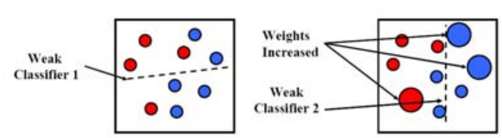
Examples misclassified by a previous weak learner are given more emphasis at future rounds.

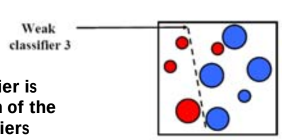
Figure adapted from Freund and Schapire

AdaBoost: Intuition



AdaBoost: Intuition



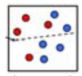


Final classifier is combination of the weak classifiers

AdaBoost Algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples



For T rounds

Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995

Visual Object Recognition Tutorial

Faces : terminology

- *Detection*: given an image, where is the face?
- *Recognition*: whose face is it?

Visual Object Recognition Tutorial

Example: Face detection

- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a "patch"/window

- Now we'll take AdaBoost and see how the Viola-Jones face detector works

K. Grauman, B. Leibe

51

Visual Object Recognition Tutorial

Feature extraction

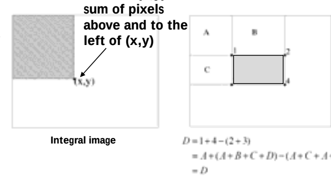
"Rectangular" filters

Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time

Avoid scaling images → scale features directly for same cost

Value at (x,y) is sum of pixels above and to the left of (x,y)



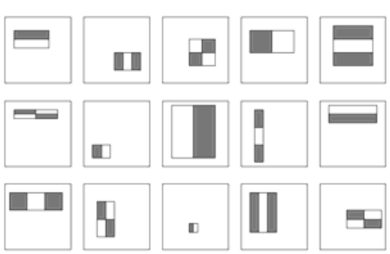
Viola & Jones, CVPR 2001

K. Grauman, B. Leibe

52

Visual Object Recognition Tutorial

Large library of filters



Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

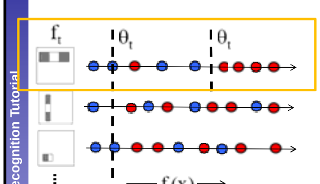
Use AdaBoost both to select the informative features and to form the classifier

Viola & Jones, CVPR 2001

Visual Object Recognition Tutorial

AdaBoost for feature+classifier selection

- Want to select the single rectangle feature and threshold that best separates **positive** (faces) and **negative** (non-faces) training examples, in terms of *weighted* error.



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

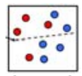
For next round, reweight the examples according to errors, choose another filter/threshold combo.

Viola & Jones, CVPR 2001

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \, |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
     $$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$
     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{\cdots}$.
- The final strong classifier is:
  $$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
  where $\alpha_t = \log \frac{1}{\beta_t}$

AdaBoost Algorithm

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995

AdaBoost for Efficient Feature Selection

- Image Features = Weak Classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples

Viola and Jones, Robust object detection using a boosted cascade of simple features, CVPR 2001

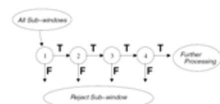
- Even if the filters are fast to compute, each new image has a lot of possible windows to search.
- How to make the detection more efficient?

Visual Object Recognition Tutorial

Cascading classifiers for detection

For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,

- **Filter for promising regions with an initial inexpensive classifier**
- **Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain**



Fleuret & Geman, IJCV 2001
Rowley et al., PAMI 1998
Viola & Jones, CVPR 2001

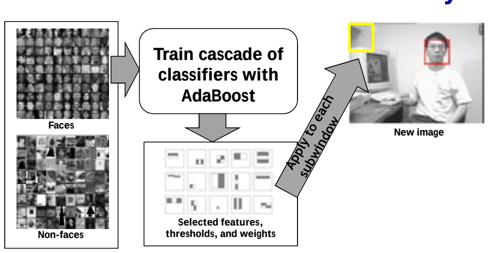
K. Grauman, B. Leibe

Figure from Viola & Jones CVPR 2001

58

Visual Object Recognition Tutorial

Viola-Jones Face Detector: Summary



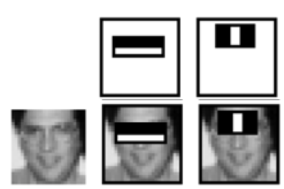
- **Train with 5K positives, 350M negatives**
- **Real-time detector using 38 layer cascade**
- **6061 features in final layer**
- **[Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]**

K. Grauman, B. Leibe

59

Visual Object Recognition Tutorial

Viola-Jones Face Detector: Results

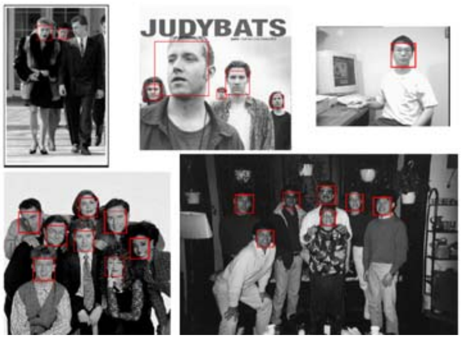


First two features selected

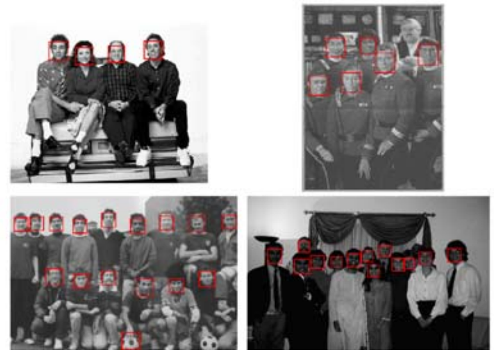
K. Grauman, B. Leibe

60

Viola-Jones Face Detector: Results



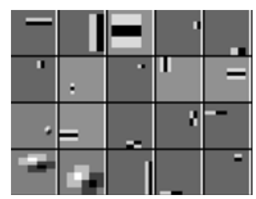
Viola-Jones Face Detector: Results



Viola-Jones Face Detector: Results

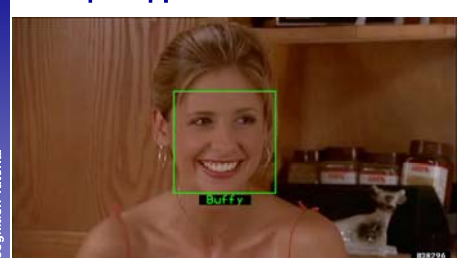
Detecting profile faces?

Detecting profile faces requires training separate detector with profile examples.



Viola-Jones Face Detector: Results

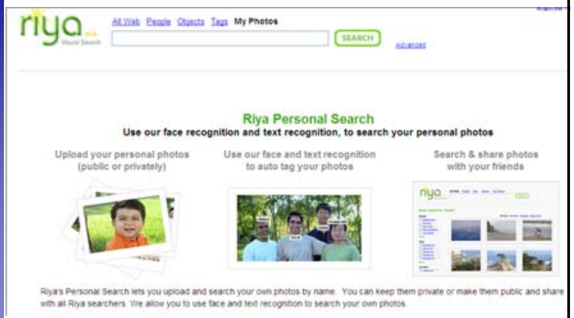
Example application



Frontal faces detected and then tracked, character names inferred with alignment of script and subtitles.

Everingham, M., Sivic, J. and Zisserman, A.
"Hello! My name is... Buffy" - Automatic naming of characters in TV video, BMVC 2006.
http://www.robots.ox.ac.uk/~vgg/research/nface/index.html

Example application: faces in photos



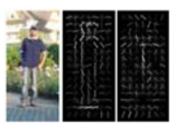
- Other classes that might work with global appearance in a window?

Pedestrian detection

- Detecting upright, walking humans also possible using sliding window's appearance/texture; e.g.,

SVM with Haar wavelets [Papageorgiou & Poggio, IJCV 2000]

Space-time rectangle features [Viola, Jones & Snow, ICCV 2003]



SVM with HoGs [Dalal & Triggs, CVPR 2005]

Highlights

- Sliding window detection and global appearance descriptors:
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes

Limitations

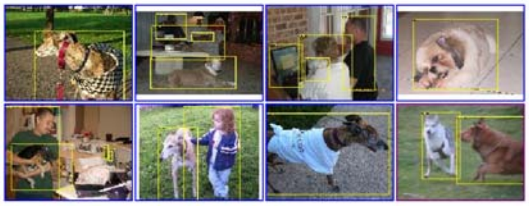
- High computational complexity
  - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - If training binary detectors independently, means cost increases linearly with number of classes
- With so many windows, false positive rate better be low

Limitations (continued)

- Not all objects are "box" shaped

Limitations (continued)

- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions



Limitations (continued)

- If considering windows in isolation, context is lost

Sliding window

Detector's view

Figure credit: Derek Hoiem

Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

Image credit: Adam, Rivlin, & Shimshoni

Models based on local features will alleviate some of these limitations…