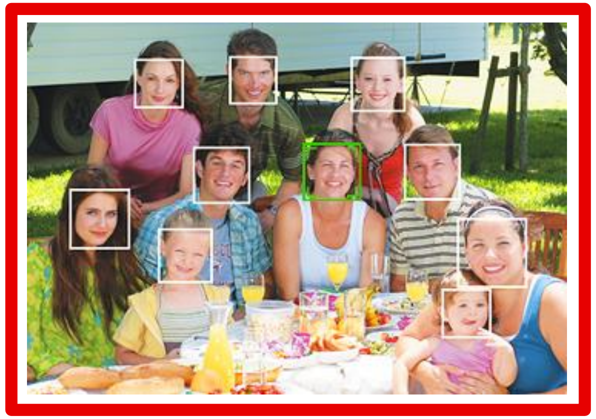
Development of intelligent systems (RInS)

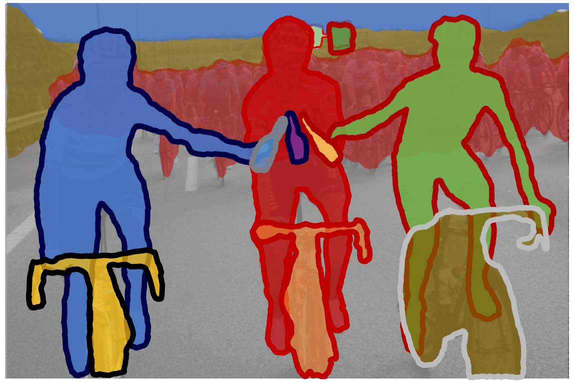
Object detection

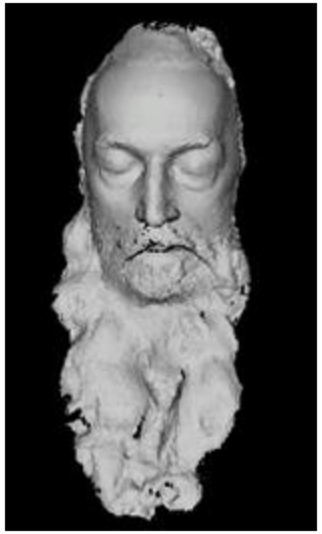
Matej Kristan, Danijel Skočaj
University of Ljubljana
Faculty of Computer and Information Science

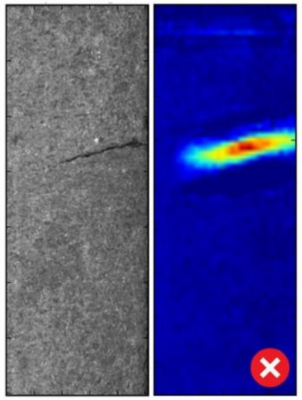
Academic year: 2021/22

Computer vision



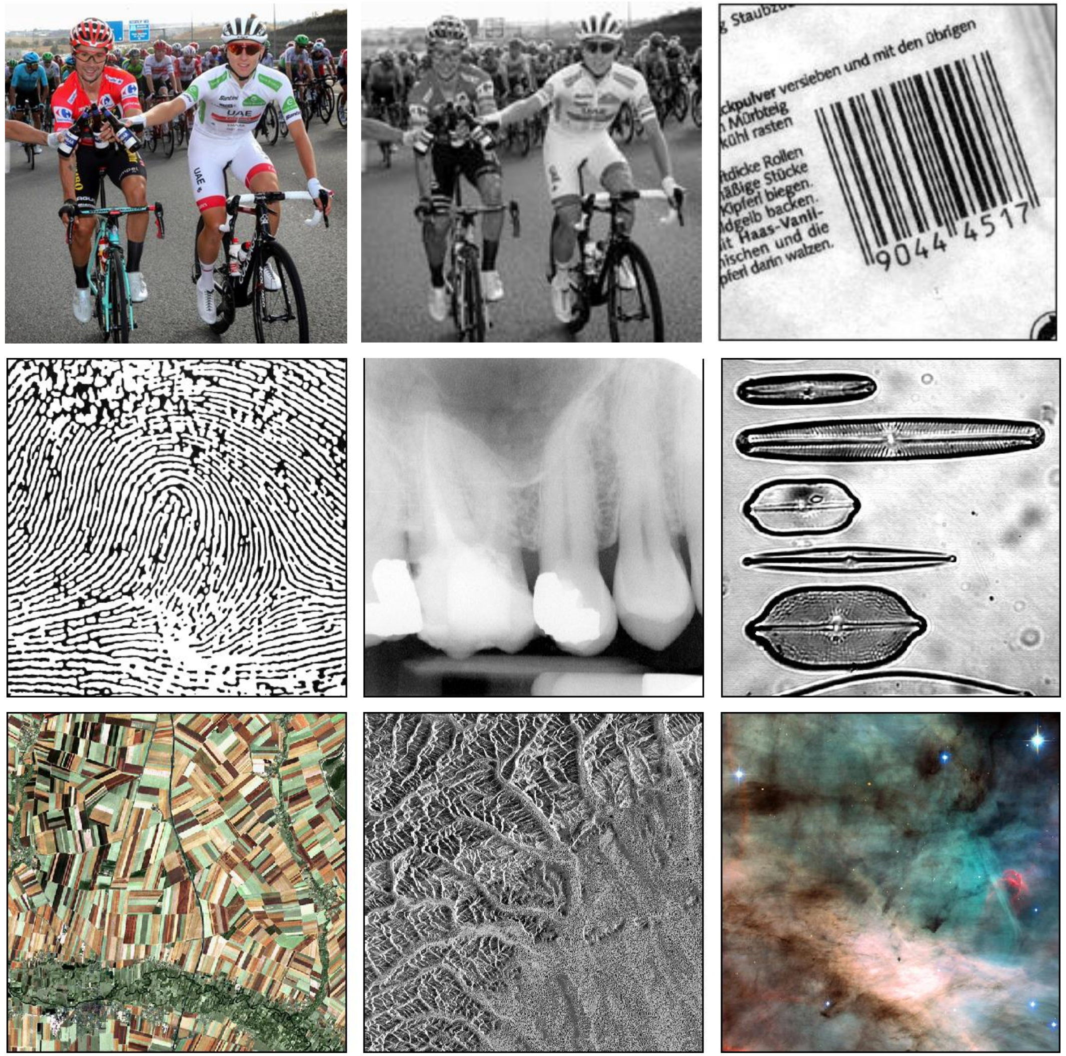






Visual information
Computer vision tasks
Face detection!

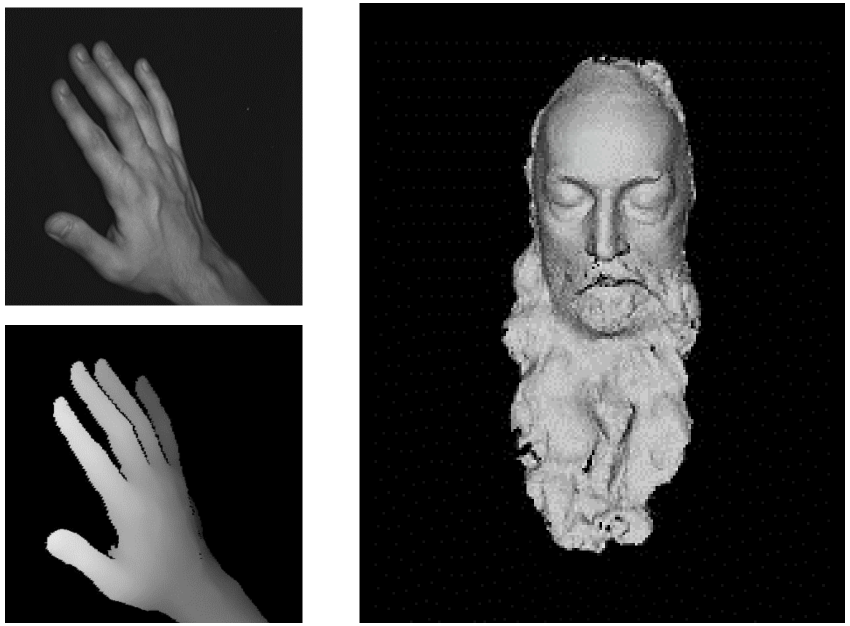
Visual information



Images

Video

3D



Classification

What is depicted in the image?

Categorisation

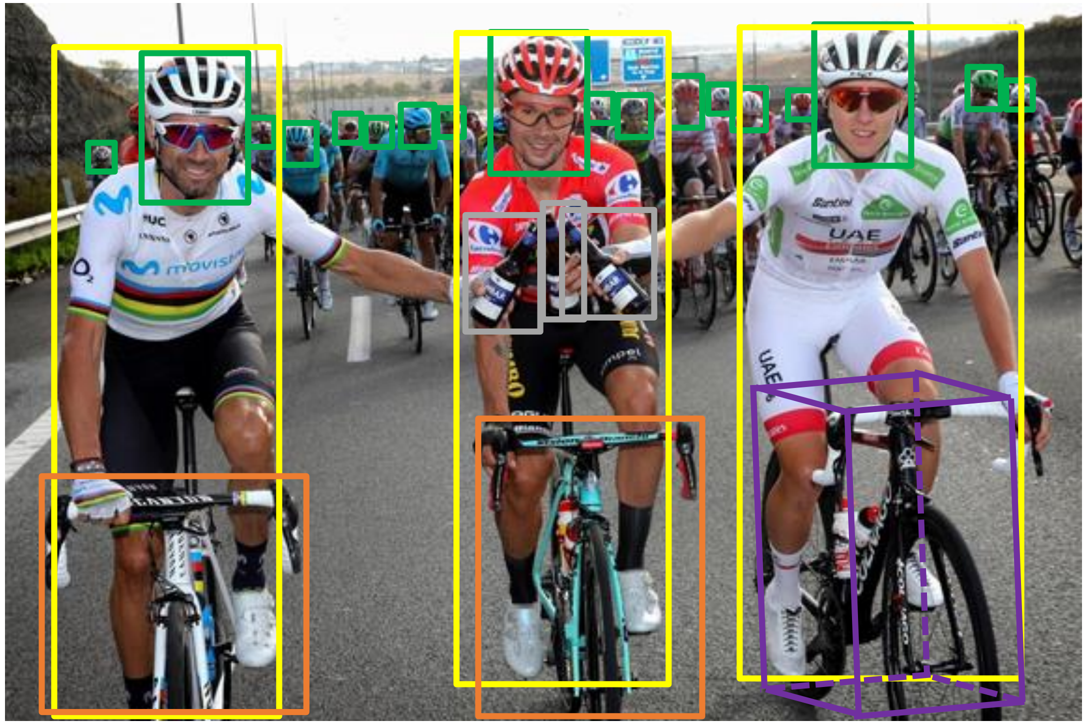
Recognition/identification of instances

Localisation

Detection

Where in the image?

Detection



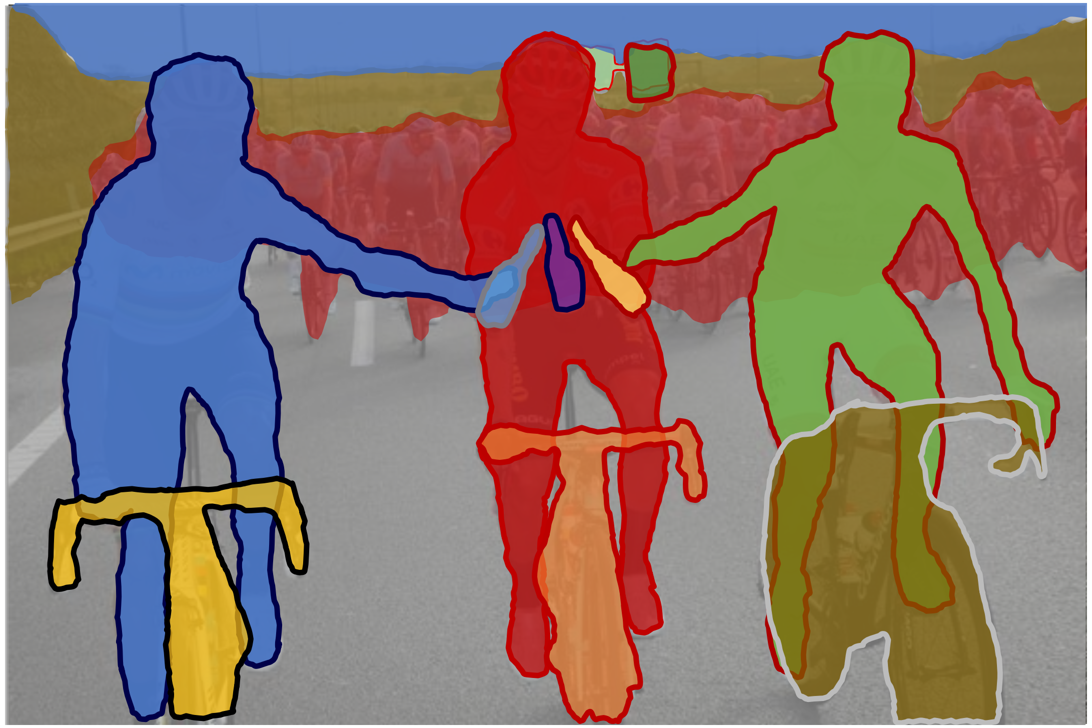
Instance segmentation

Segmentation

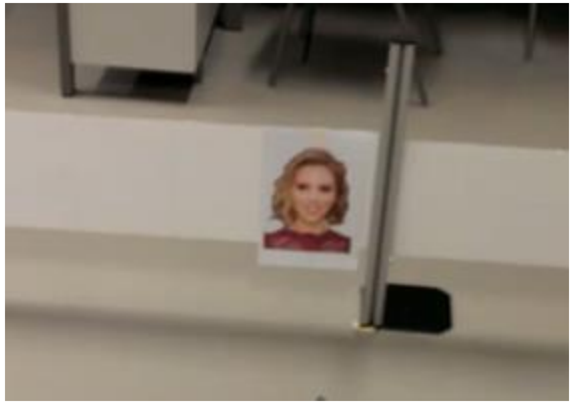
What does every pixel represent?

Semantic segmentation

Panoptic segmentation



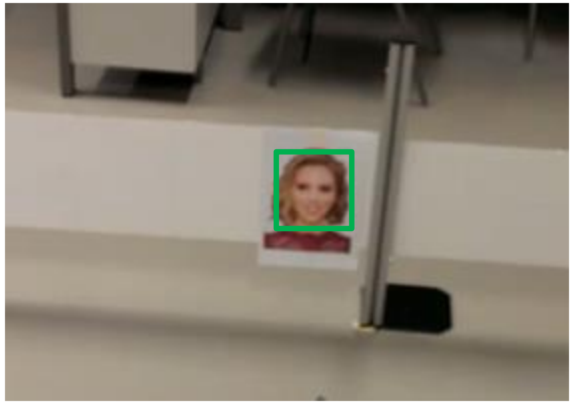
Two stage object detection and recognition

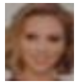


very fast
efficient

Face detection

HOG+SVM
AdaBoost
SSD





Face recognition

CNN
PCA/LDA

could be slower
computationally more complex

„Scarlet“

Observation model

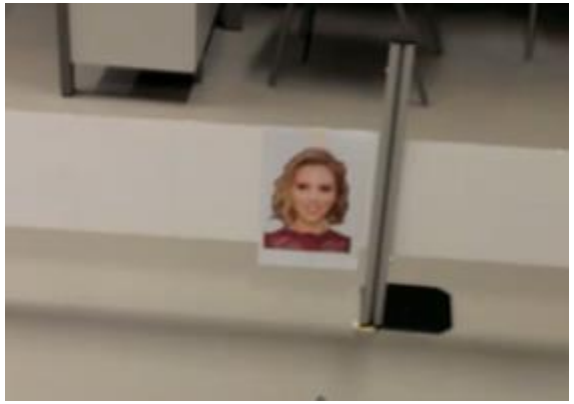
- Three face detectors given
  - HOG+SVM
  - AdaBoost
  - SSD
  - Any other?
- Not perfect
- Which one is better?
  - More true positives
  - Less false positives
- Test set
  - Images, videos
  - Different angles, illumination
  - Motion blur, etc.
- Observation model
  - Performance
    - at different distances and angles
    - at different illuminations

Robustification of detection

- Use and robustify the better detector
- Take into account temporal dimension
  - Repetitive detections more robust
  - Filter out false positives
- Take into account spatial dimension
  - Non-maximum suppression
  - Observation model

- Map the image from 2D image to 3D world
- Anchor the image into the map
- Non-maximum suppression in the map
- Redetection of faces from different directions

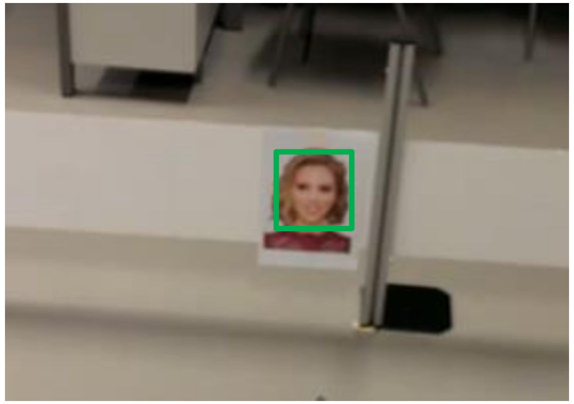
Two stage object detection and recognition

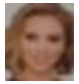


very fast
efficient

Face detection

HOG+SVM
AdaBoost
SSD





Face recognition

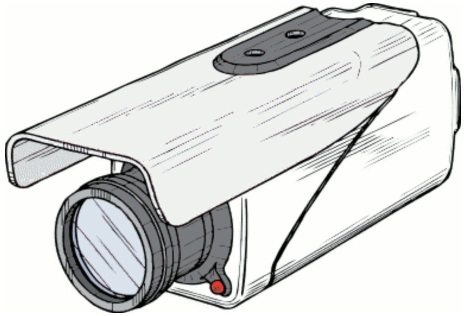
CNN
PCA/LDA

could be slower
computationally more complex

„Scarlet"

Univerza *v Ljubljani*

Machine Perception
Category detection

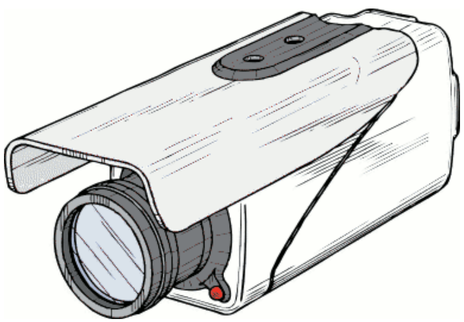


Matej Kristan

Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

Univerza v Ljubljani

ViCoS



Machine Perception
Category detection

Matej Kristan

Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

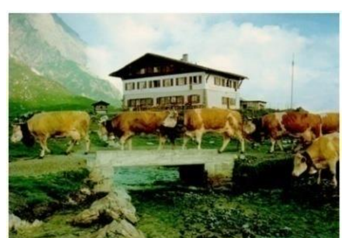
Object categorization

- How to detect/recognize any car?





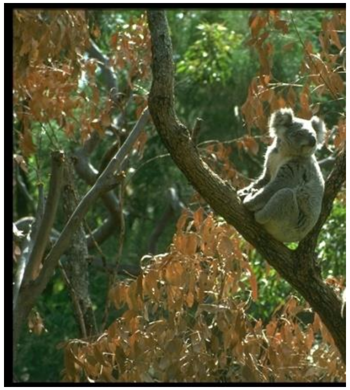
- How to detect/recognize any cow?

Challenge: Robustness


Illumination


Object pose


Clutter


Occlusion


Within-class variability


Aspect

Detection by classification: Standard approach

- Apply machine learning to learn "features" that differ between object in interest and background.
- Basic component: a binary classifier

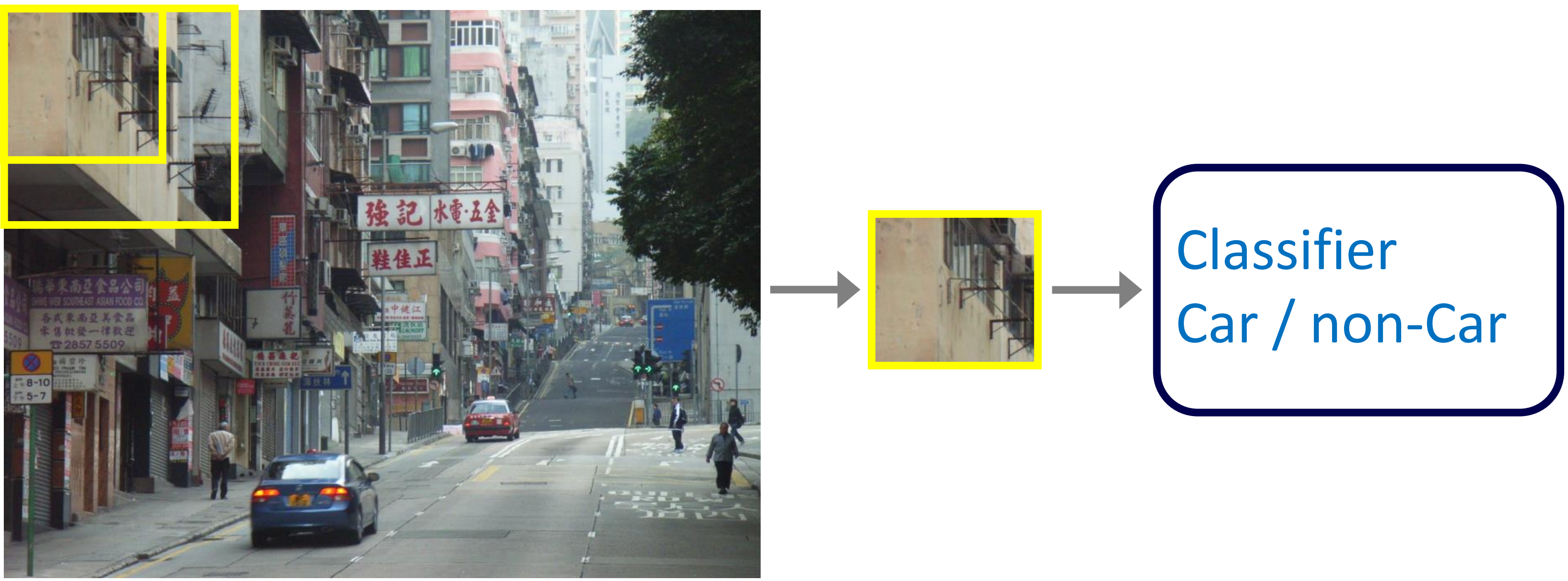
Classifier:
car / non-car

No, not a car.

Yes, a car.

Detection by classification: Standard approach

- Apply a sliding window to cope with clutter and localize the object.



- This is essentially a greedy search using a (very) large number of local decisions.

Detection by classification: Standard approach

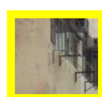
A bit more detailed description:

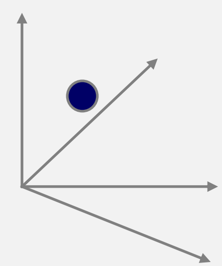
1. Get training data
2. Determine features (semi or fully automatic)
3. Train a classifier

Examples of cars

Examples of non-cars

Training examples





Feature space

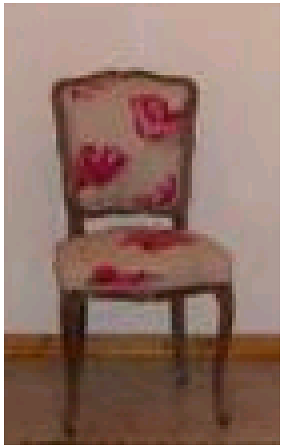
Classifier
Car / non-car

Not a car

Images taken from: Kristen Grauman

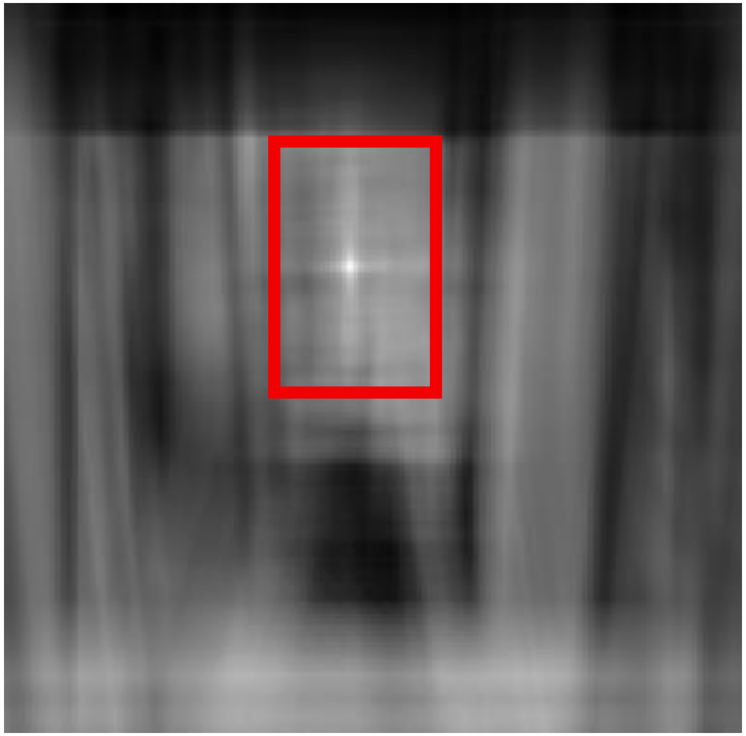
Is recognition really that difficult?

A chair



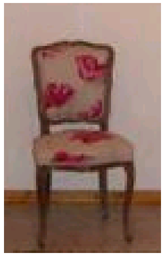
Find this chair in the image

Output of a normalized cross correlation.

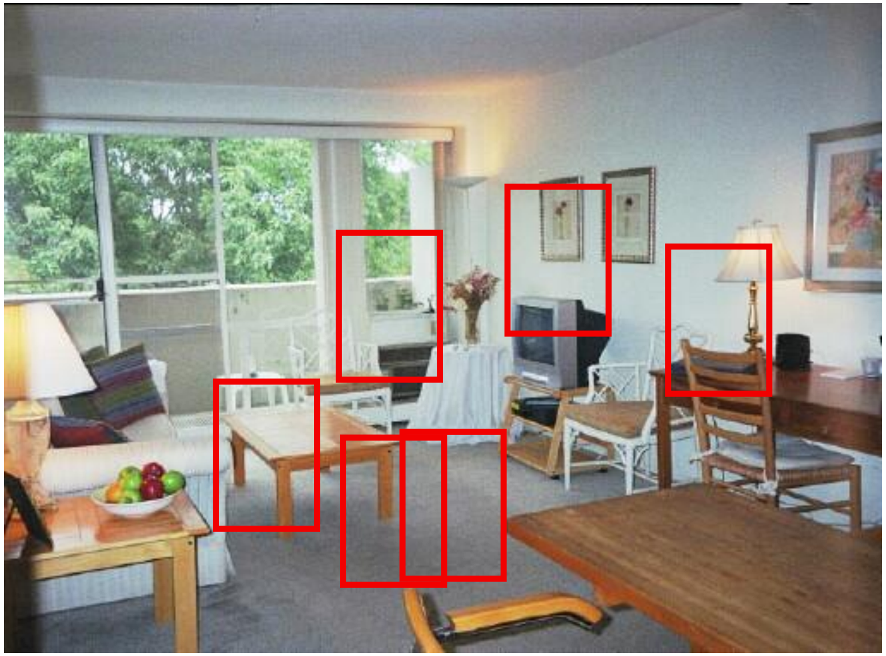


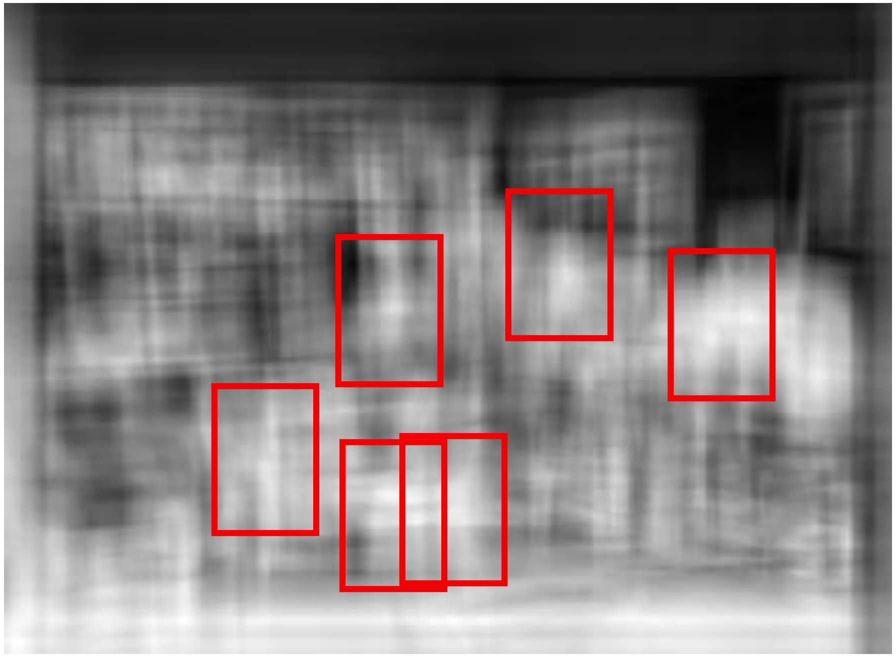
Not really?!

Is recognition really that difficult?



Analyze this!

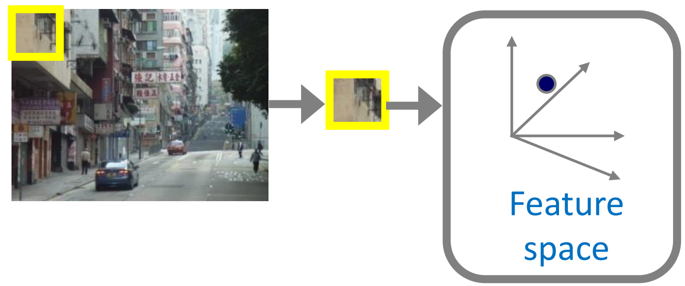


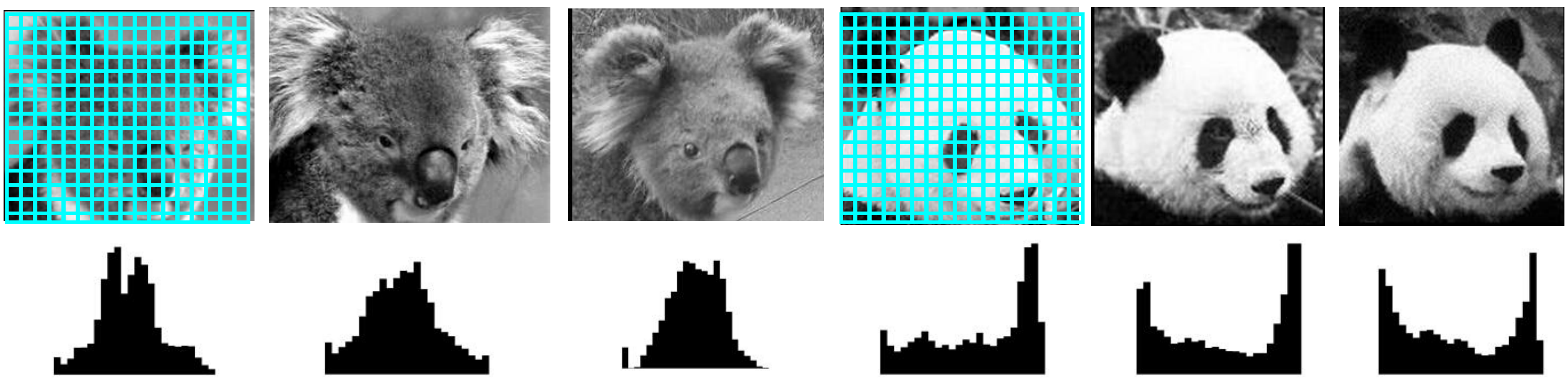


Completely useless – does not work at all.

Main issue: Feature space!

Straight-forward global features





A simple holistic description of image content, e.g.,

- Intensity/color histograms
- Intensity vector
- …

Learned global features (e.g., PCA)

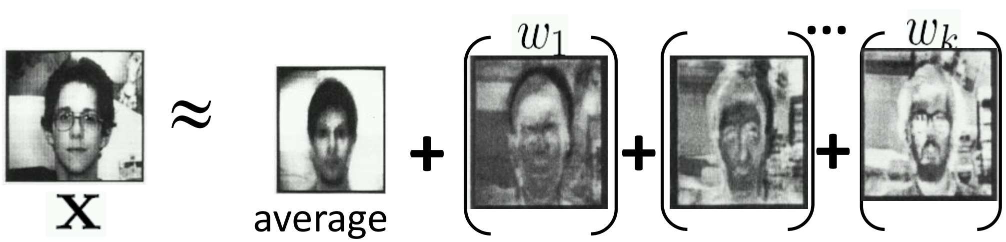
The PCA learns features that maximize dataset reconstruction

Train images

Average image

Eigenimages calculated from the covariance matrix

Calculate a low-dimensional representation by a linear subspace.



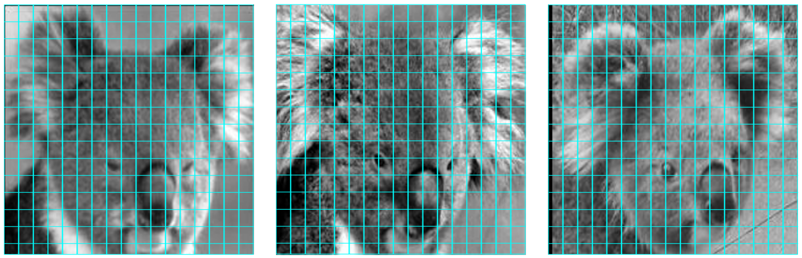
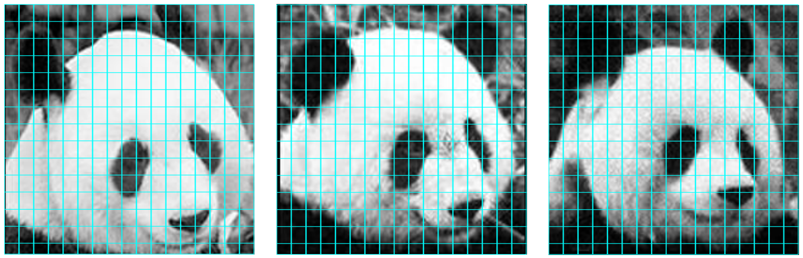
Project new image into the subspace.

Recognize by using a nearest-neighbor search in the subspace.

[Turk & Pentland, 1991]

Hand-crafting global features

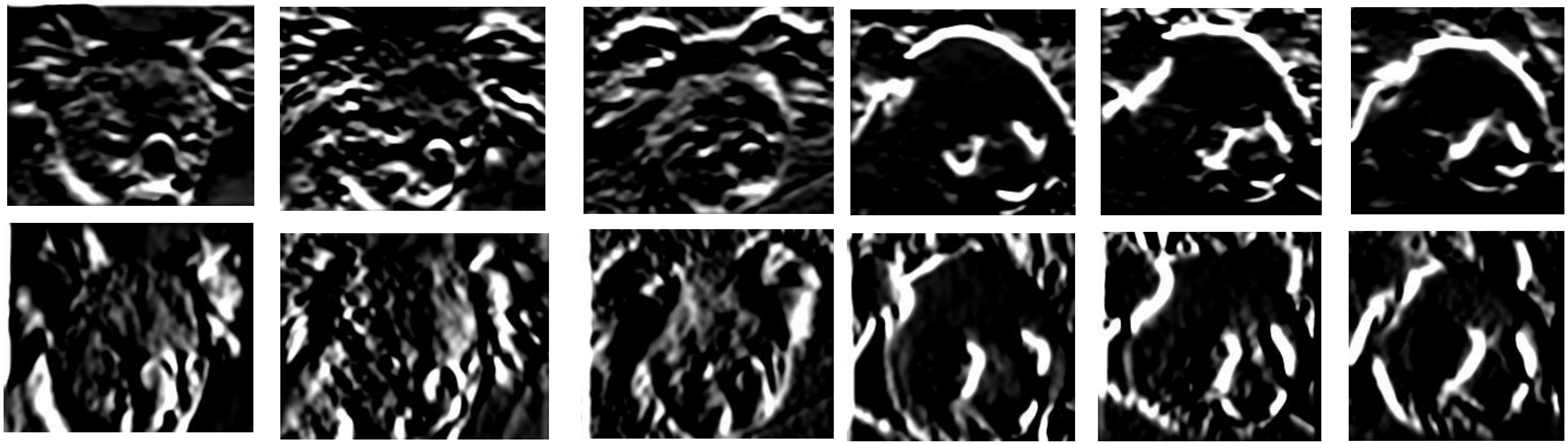
- Problem 1: Pixel-based representations, are sensitive to small shifts:

- Problem2: Color or gray-level representation is sensitive to illumination changes or within-class color variations.


Hand-crafting global features

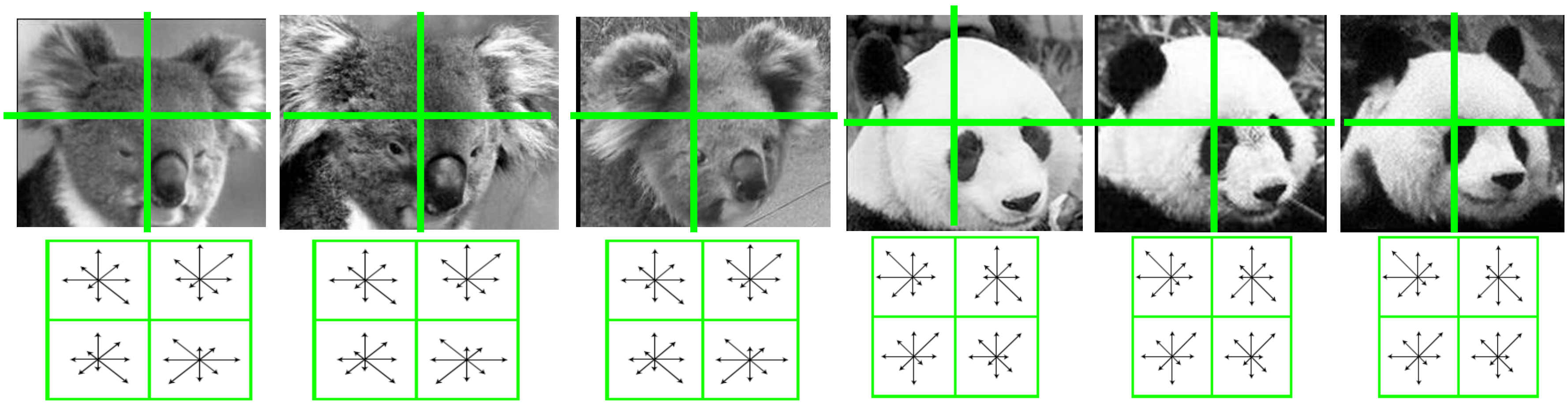
- Solution: Edges, contours and oriented intensity gradients



Change intensity features into gradient-based features…

Gradient-based representation

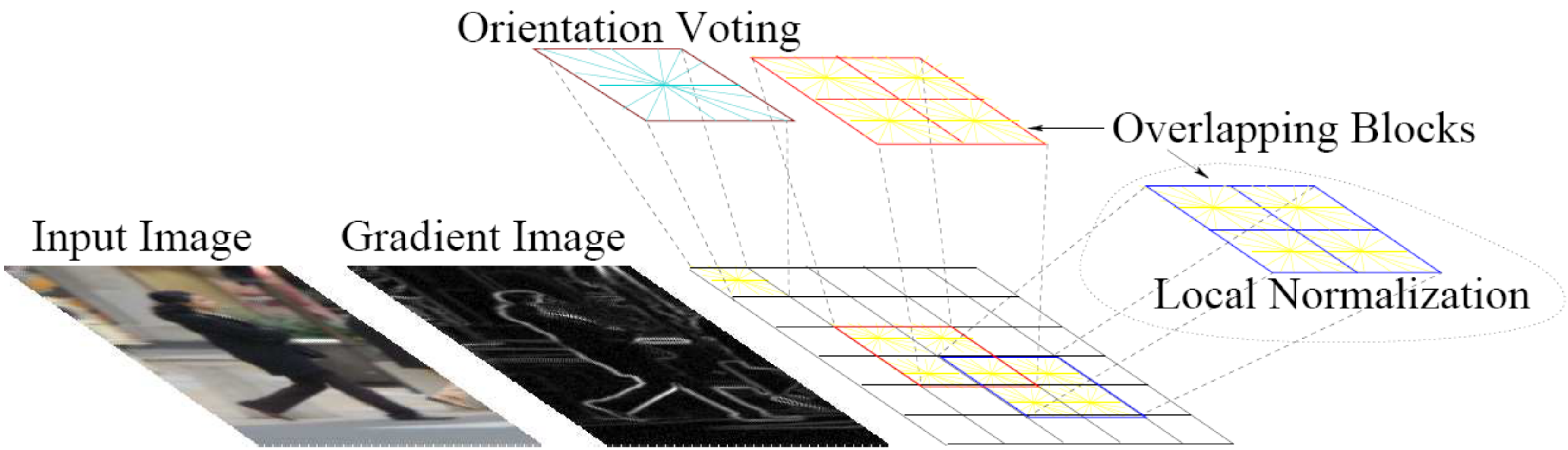
- Edges, contours and oriented intensity gradients



- Encode local gradient distributions using histograms
  - Locally unordered: invariant to small shifts and rotations
  - Contrast normalization:
    addresses non-uniform illumination and varying intensity.

Gradient-based representation: HOG

Histogram of Oriented Gradients: HOG



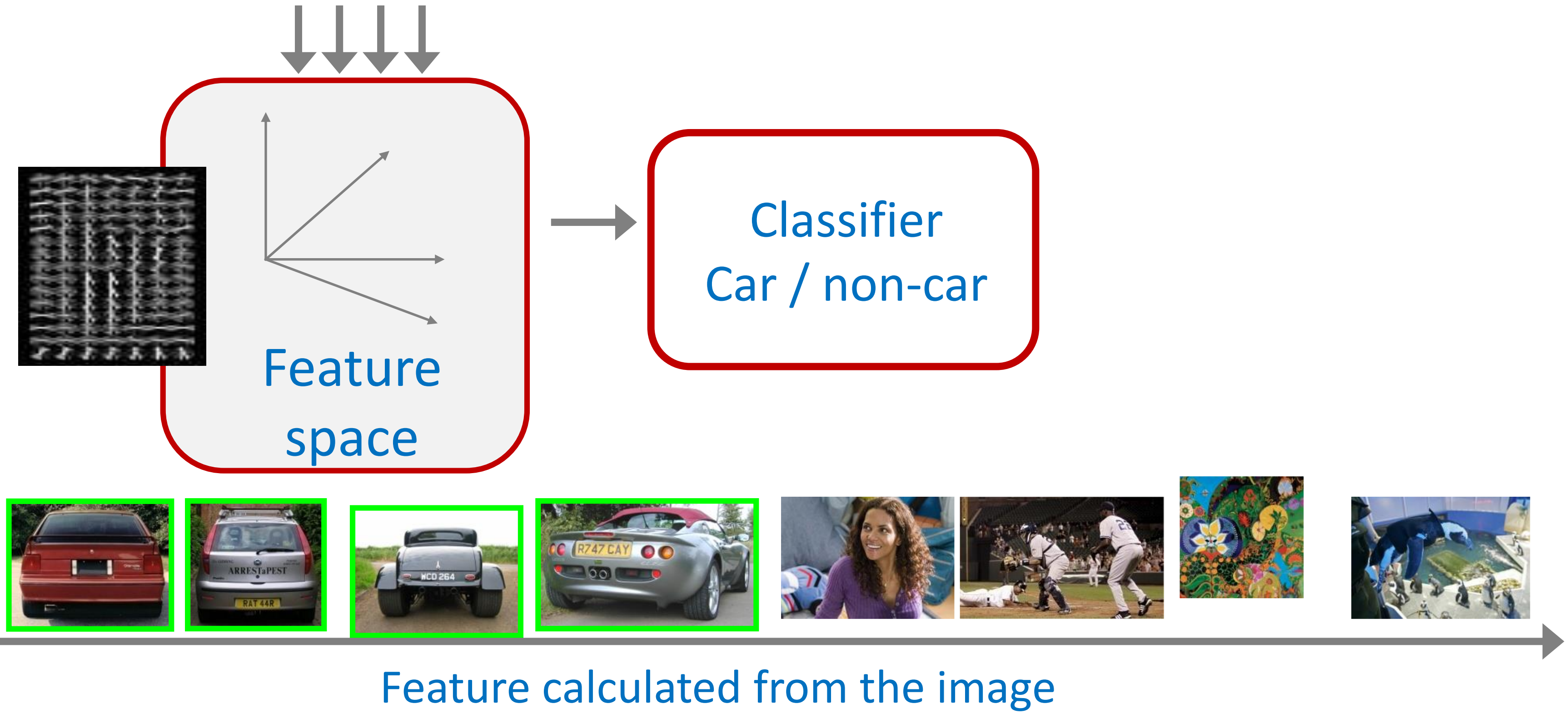
For each cell visualize the strongest gradient orientations.

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

[Dalal & Triggs, CVPR 2005]

Let’s build a classifier

- We hand-crafted a feature descriptor that is invariant to illumination changes and small deformation.
- How do we calculate a decision in each sub-window?



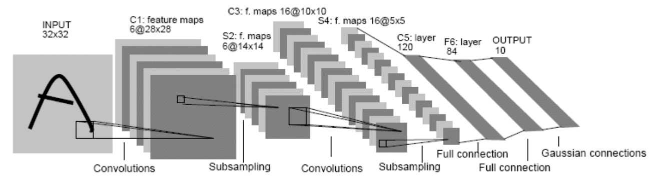
Lots of choices for a classifier

Nearest neighbor

10^6 examples

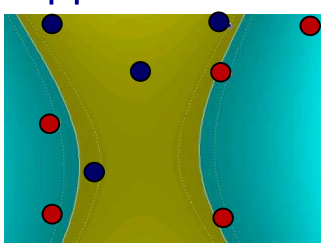
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



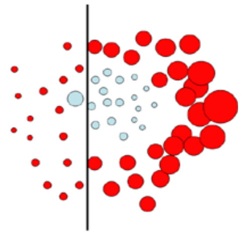
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



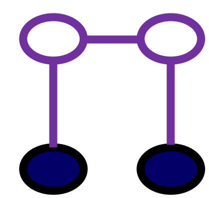
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



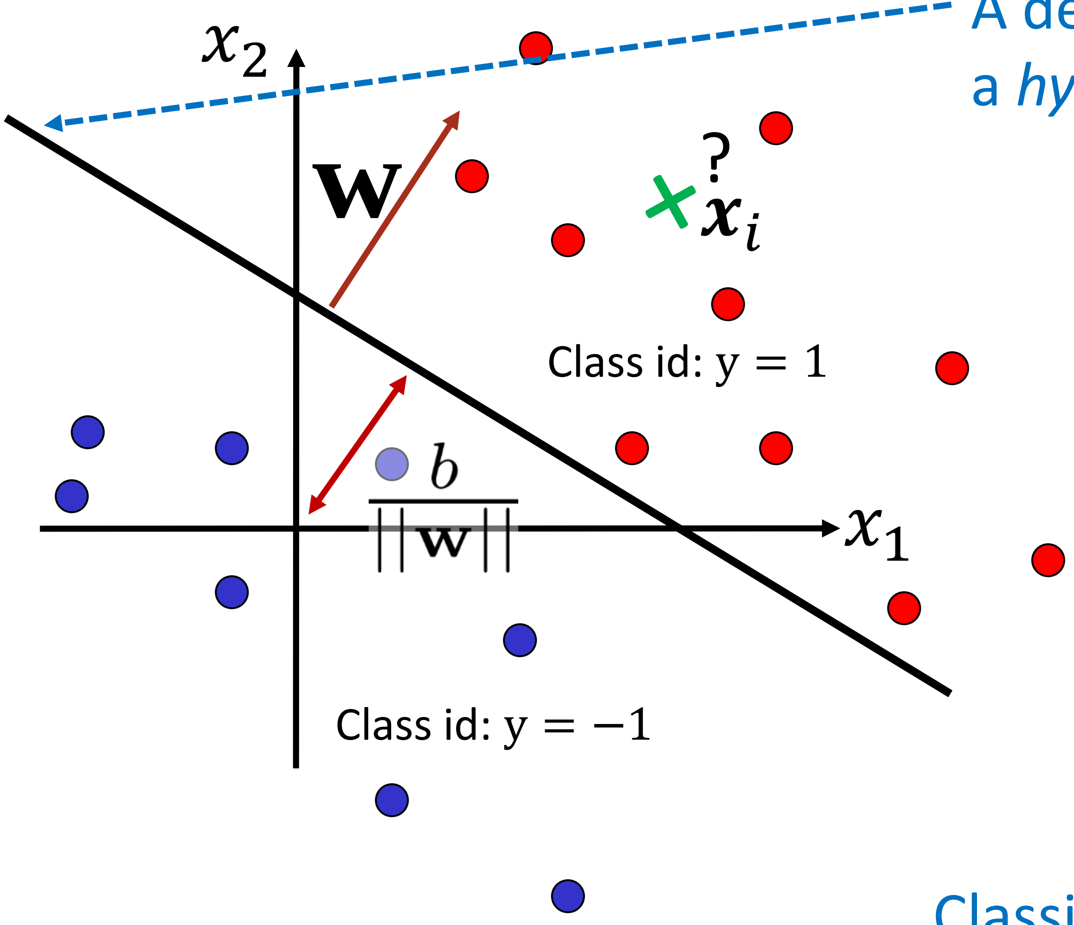
Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,…

Conditional Random Fields



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

Consider a linear classifier



A decision boundary, in general, a *hyper-plane*:

$$ax_1 + cx_2 + b = 0$$

Define:

$$\mathbf{w} = \begin{bmatrix} a \\ c \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

A general hyper-plane eq:

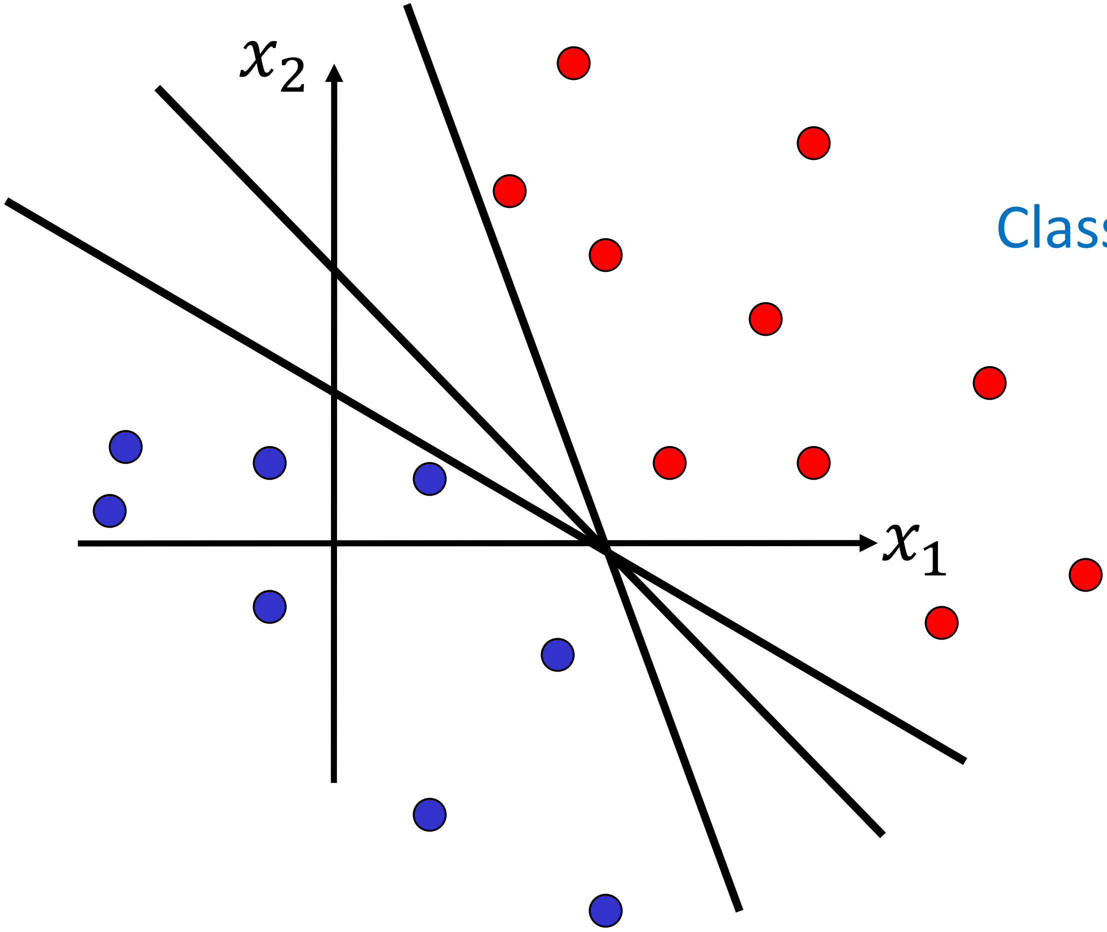
$$\mathbf{w}^T \mathbf{x} + b = 0$$

Classification of x = sign checking:

Learning = Choosing $\boldsymbol{w}$ and b!

$$f(\mathbf{x}) = \text{sign}\,(\mathbf{w}^T \mathbf{x}_i + \text{b})$$

Best separation hyper-plane?



A general hyper-plane eq:

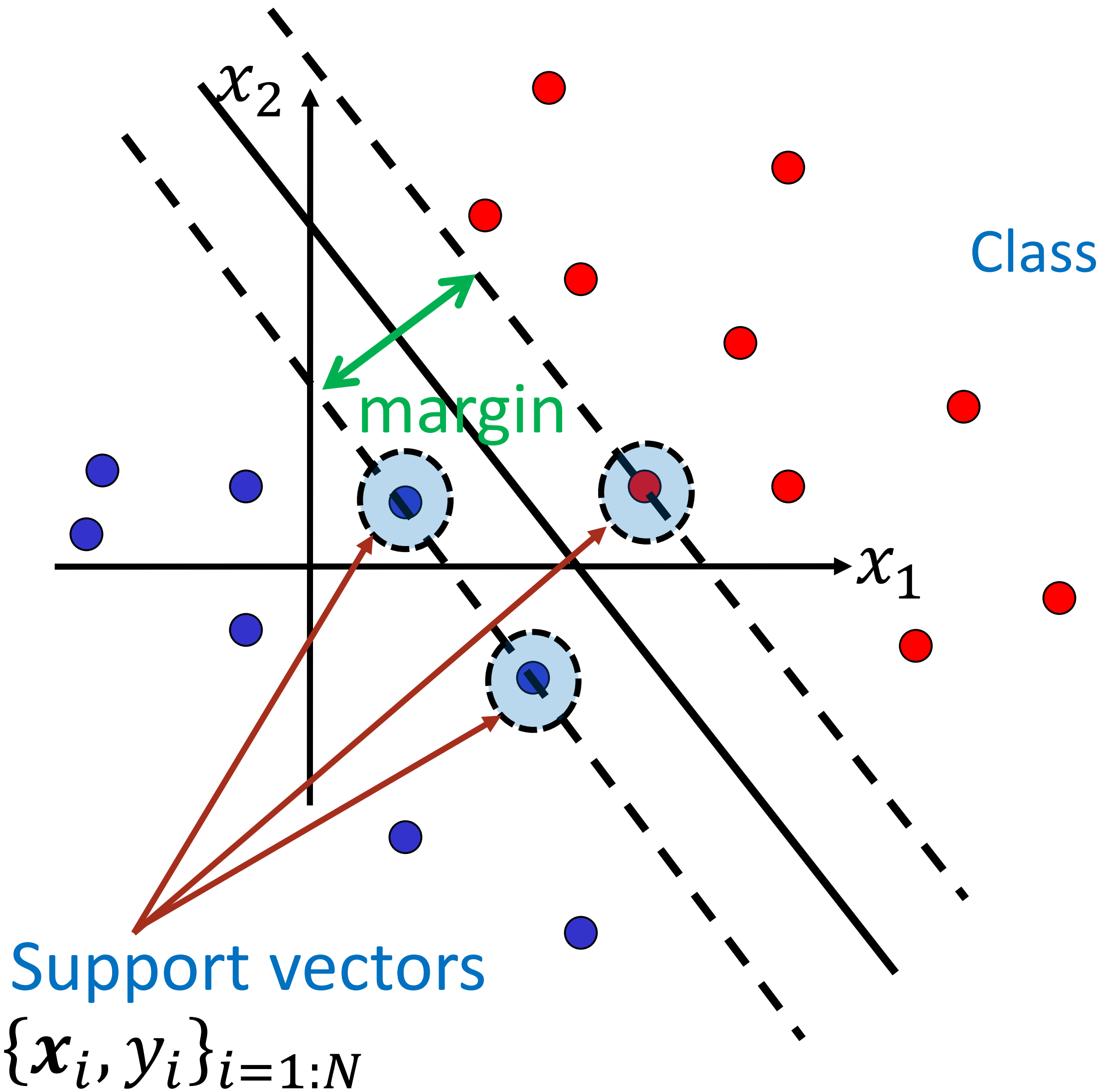
$$\mathbf{w}^T\mathbf{x} + b = 0$$

Classification of x = sign checking:

$$f(\mathbf{x}) = \text{sign}\,(\mathbf{w}^T\mathbf{x} + \text{b})$$

Choosing $\boldsymbol{w}$ and b?

Best separation hyper-plane?



A general hyper-plane eq:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

Classification of x = sign checking:

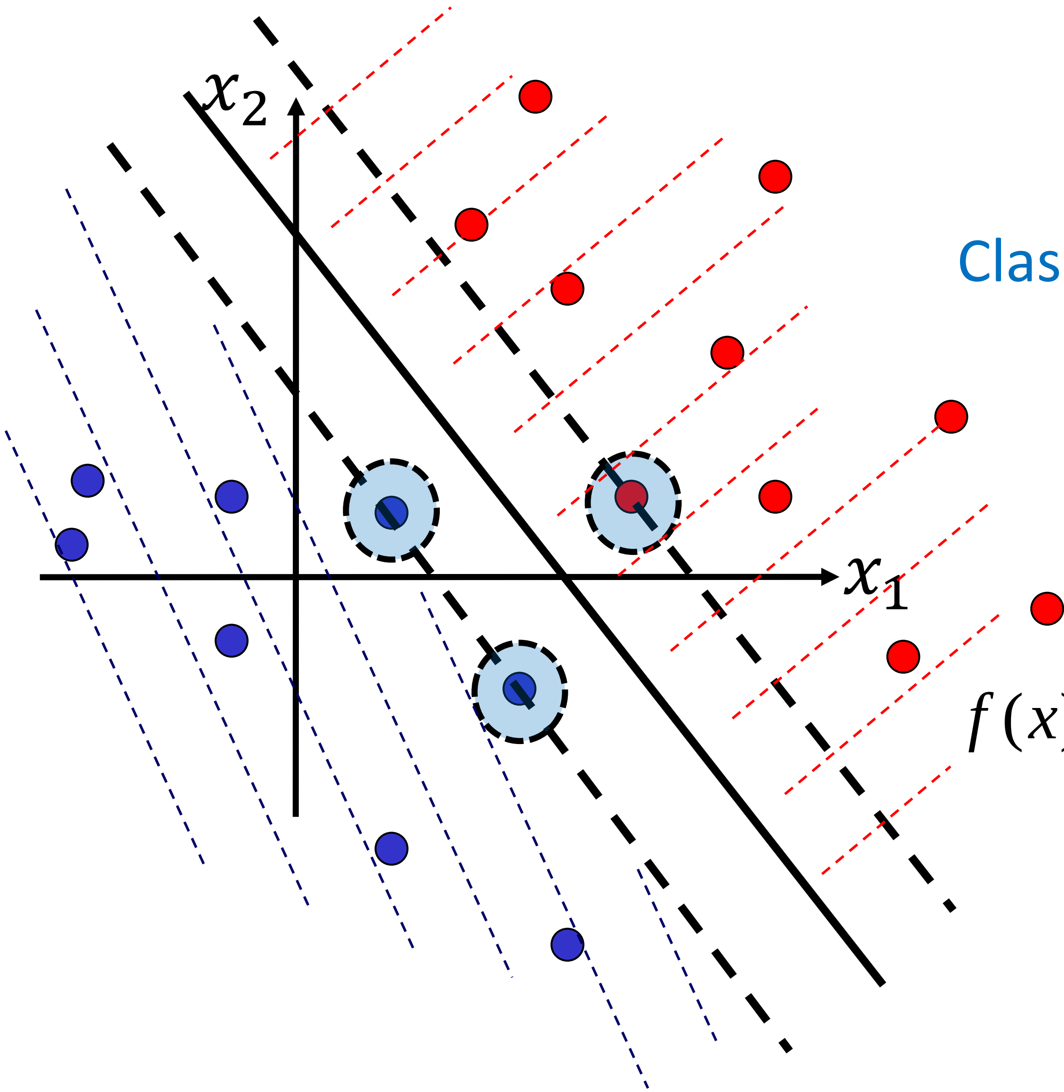
$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + \text{b})$$

- The hyper-plane that maximizes the margin between positive $(y_i = 1)$ and negative $(y_i = -1)$ training examples.

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

Have to select SVs and learn α_is!

Classification == similarity to SVs



A general hyper-plane eq:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

Classification of x = sign checking:

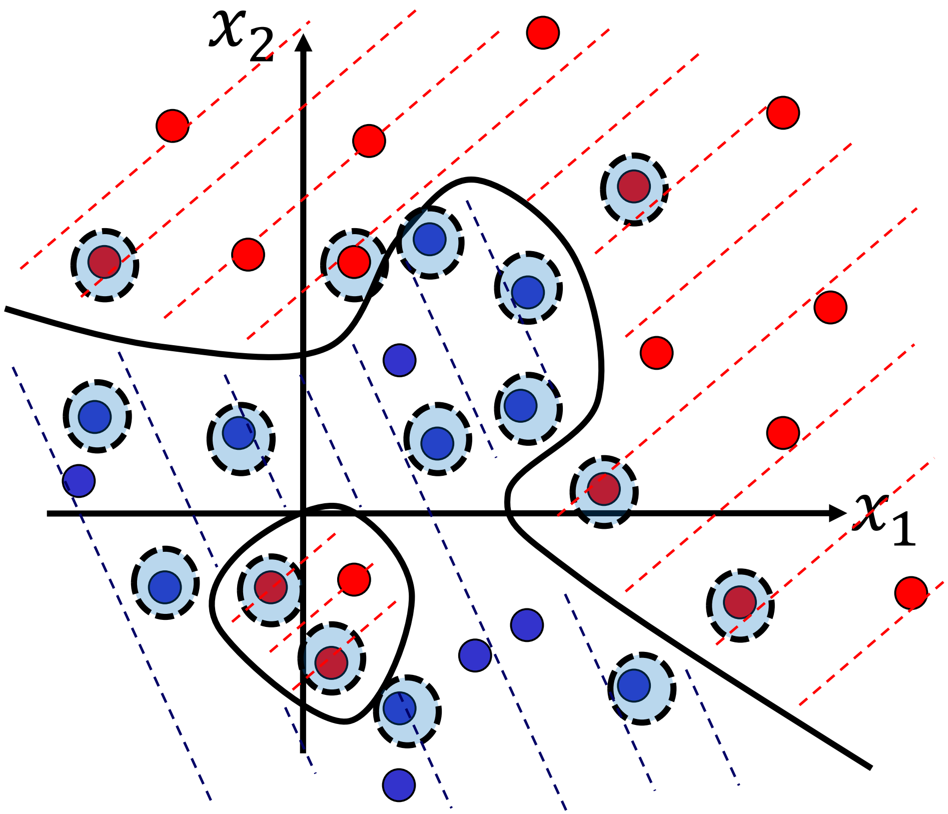
$$f(\mathbf{x}) = \text{sign}\,(\mathbf{w}^T\mathbf{x} + \text{b})$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

$$\begin{aligned} f(x) &= \text{sign}\,(\mathbf{w}^T\mathbf{x} + \text{b}) \\ &= \text{sign}\left(\sum_{\text{i}} \alpha_{\text{i}} y_i \mathbf{x}_i{}^T\mathbf{x} + b\right) \\ &= \text{sign}\left(\sum_{\text{i}} \alpha_{\text{i}} y_i \boxed{\phi(\mathbf{x}_i, \mathbf{x})} + b\right) \end{aligned}$$

A "kernel"

Non-linear kernels



A non-linear kernel, e.g.,:

$$\phi(\mathbf{x}_i, \mathbf{x}) = e^{(-\frac{1}{2}(\mathrm{x}_i - \mathrm{x})^2 / \sigma^2)}$$

Classification funct. unchanged:

$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{\mathrm{i}} \alpha_{\mathrm{i}} y_i \phi(\mathbf{x}_i, \mathbf{x}) + b\right)$$

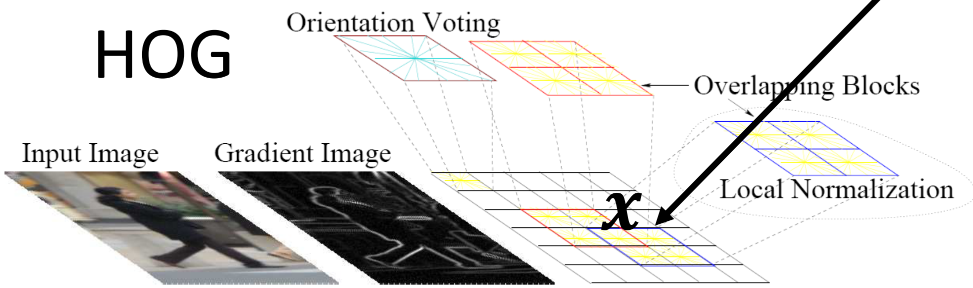
Non-linear kernels lift the dimensionality of the data such and apply a linear hyper-plane in this high-dimensional space.
The hyper-plane becomes nonlinear in the original space.

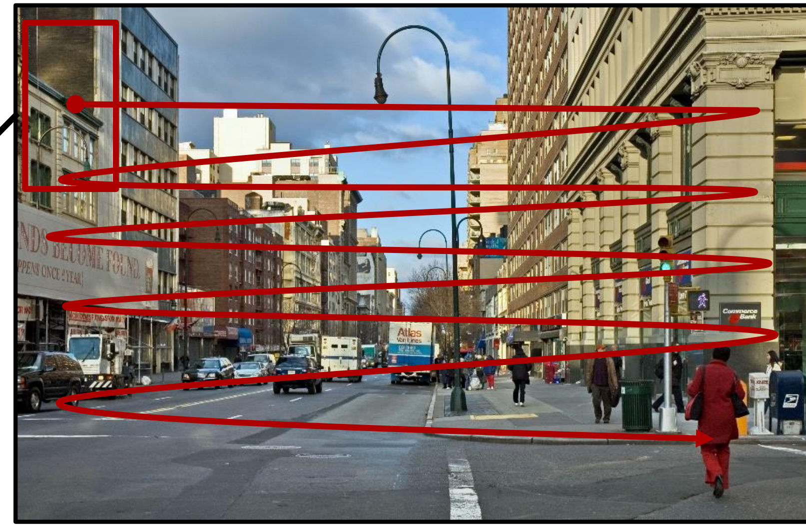
Application: Pedestrian detection

- Sliding window:

1. extract HOG at each displacement
2. classify by a linear SVM

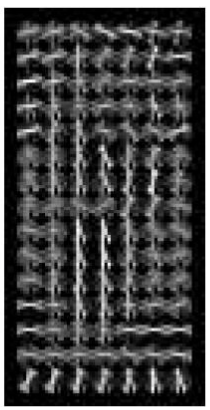
HOG



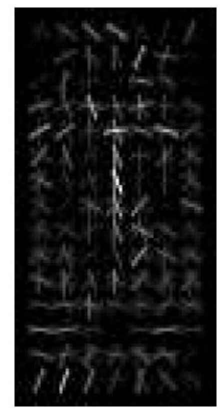


$$f(x) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$


input


HOG


HOG cells weighted by the positive support vectors


HOG cells weighted by the negative support vectors

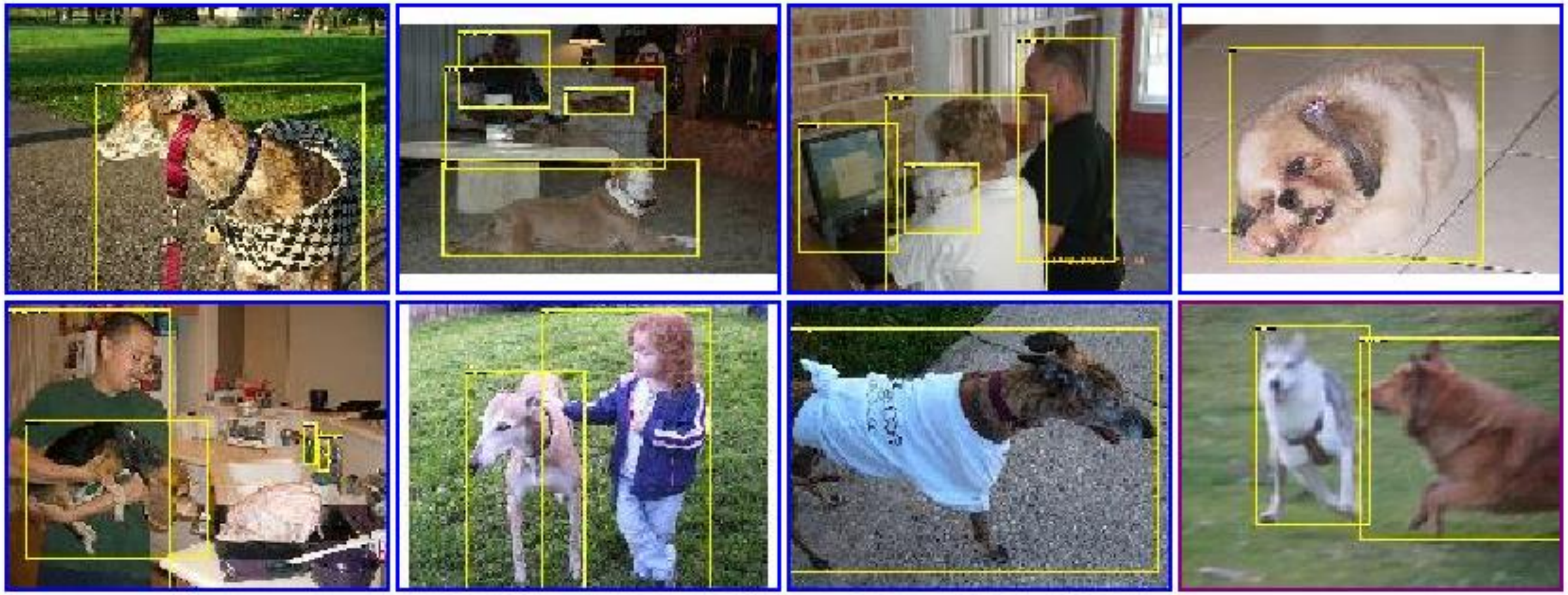
Dalal and Triggs, Histograms of oriented gradients for human detection, CVPR2005

Pedestrian detection HoG+SVM

Navneet Dalal, Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Objects (non)rigidly deform

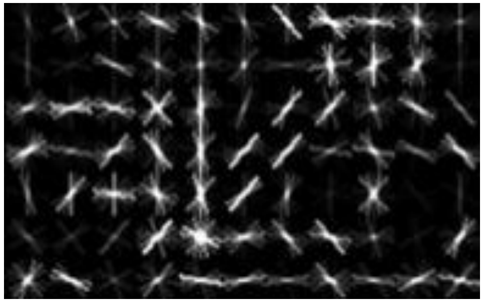
- Nonrigid/deformable objects poorly detected using a fixed structure.



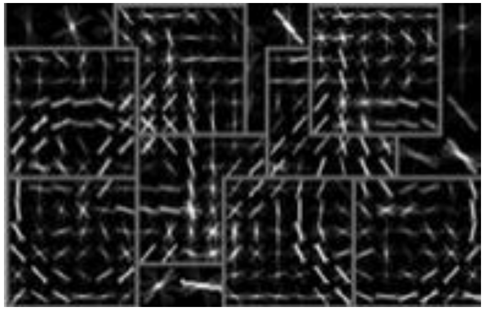
Deformable parts models (DPM)

- Each part is a HOG descriptor.
- Learn a classifier for HOGs and geometric constraints simultaneously by structured SVM.

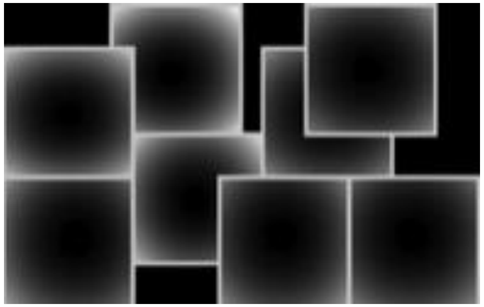
Root (global) part

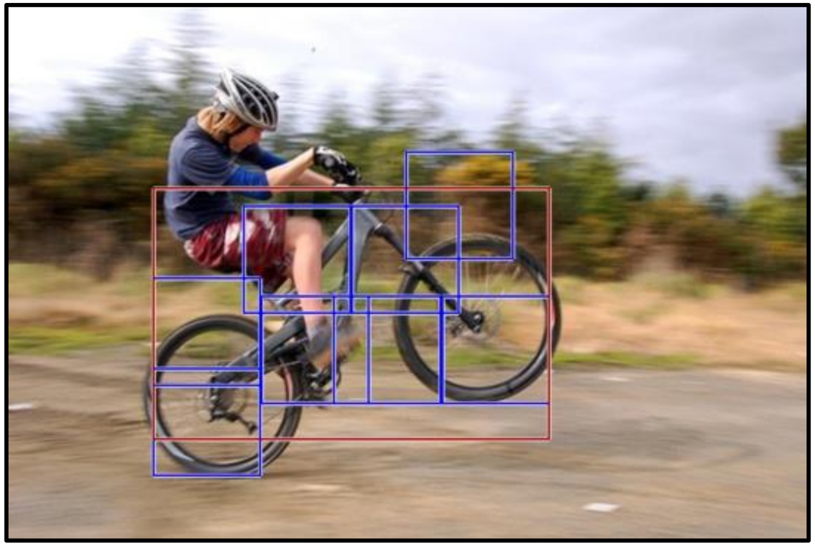


Overlaid parts



Geometric constraints





P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, **Object Detection with Discriminatively Trained Part Based Models** IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, Sep. 2010

A Great tutorial on DPMs at ICCV2013: http://www.cs.berkeley.edu/~rbg/ICCV2013/

Time/computation criticality

- A lot of applications are time- and resources-critical
- Require efficient feature construction
- Require efficient classification
- A case study:
    - Face detection

Face detection

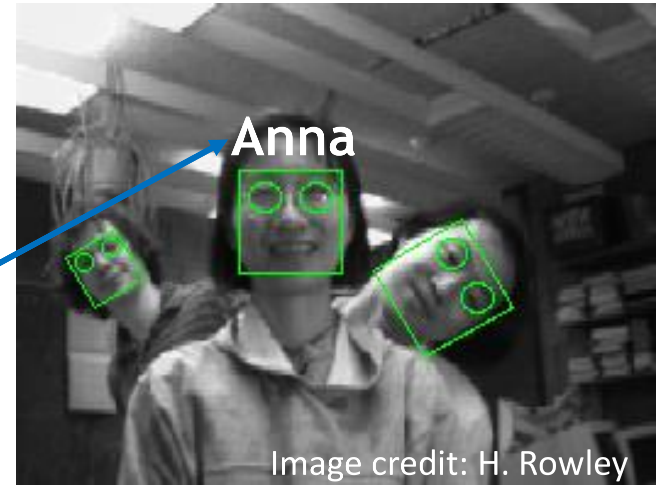
Application specifics:

- Frontal faces are a good example, where the global appearance model + sliding window works well:
  - Regular 2D structure
  - Central part of the face is well approximated by rectangle.

Images from: Kristen Grauman

Faces: Terminology

- Detection:
  Given an image, where are the faces?



- Recognition:
  Whose face is it?

- Classification:
  Gender, Age?

Fast face detection

- To apply in real-time applications
    1. Feature extraction should be **fast**
    2. Classifier application should be **fast**
- These points addressed next

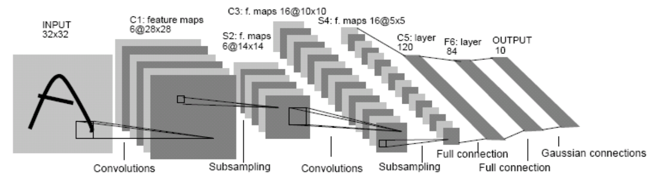
Choice of classifiers

Nearest neighbor

10^6 examples

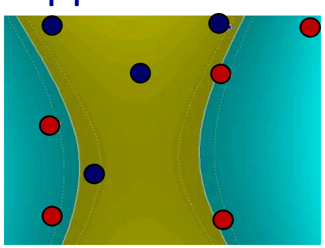
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



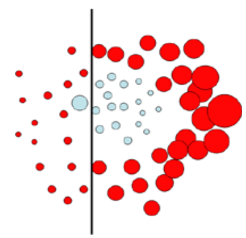
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



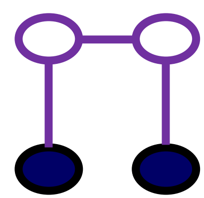
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



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

Conditional Random Fields



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

Boosting

- Build a strong classifier from a combination of many "weak classifiers" – weak learners (each at least better than random)
- Flexible choice of weak learners
  - This includes fast but inaccurate classifiers!

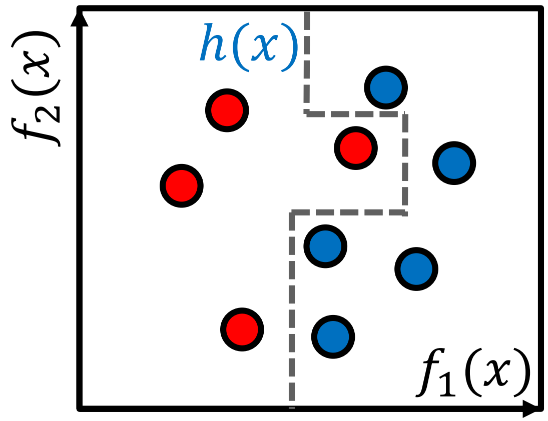
- We'll have a look at the AdaBoost (Freund & Schapire)
  - Simple to implement.
  - Basis for the popular Viola-Jones face detector.

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, 1999.

Adaboost: Intuition

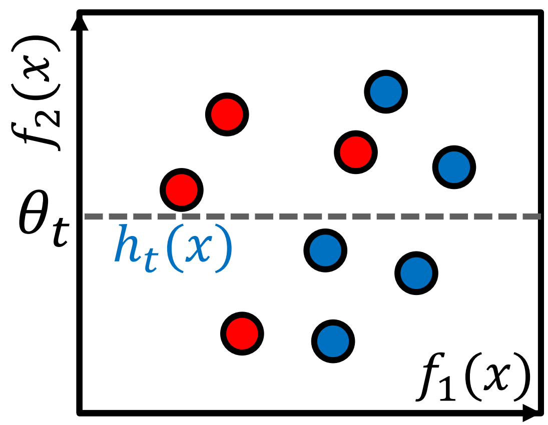
- Task: Build a classifier which is a weighted sum of many classifiers

$$h(x) = \begin{cases} 1 & \underbrace{\sum_{t=1}^{T} \alpha_t}_{\text{weight}} \underbrace{h_t(x)}_{\text{weak classifier}} \geq \frac{1}{2}\sum_{t=1}^{T}\alpha_t \\ 0 & \text{otherwise} \end{cases}$$

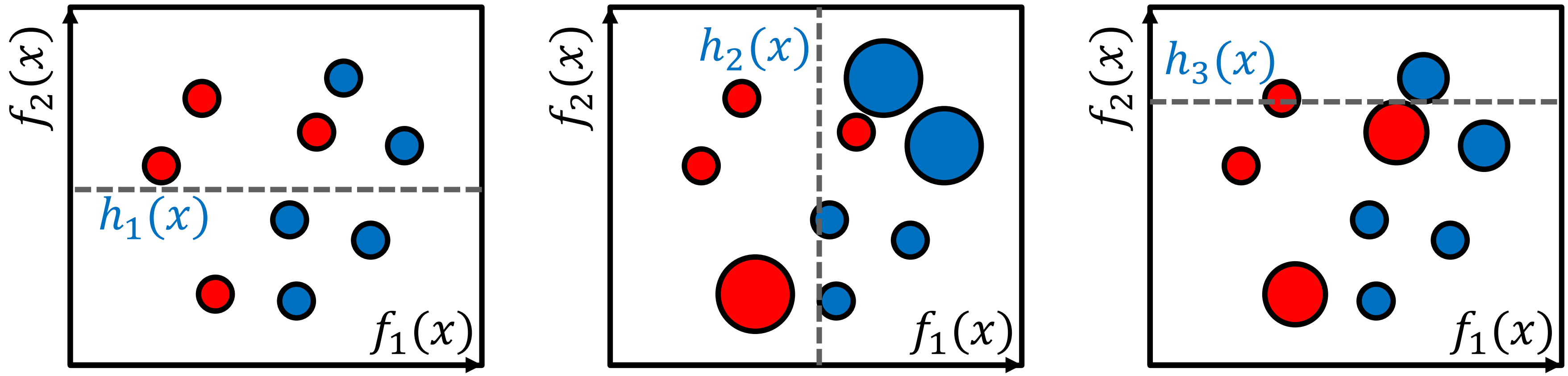


Example of a weak classifier:

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



AdaBoost: Intuition



- Train a sequence of weak classifiers.
- Each weak classifier splits train examples with at least 50% accuracy.
- Those examples that are incorrectly classified by the weak classifier, get more weight in training the next weak classifier.

Final classifier is a combination of many weak classifiers!

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

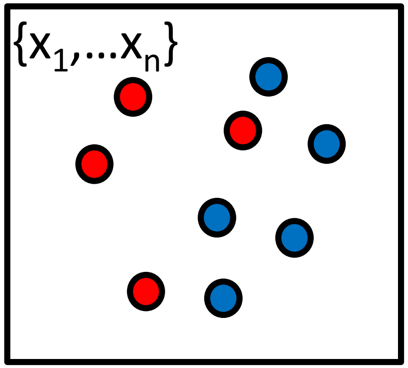
AdaBoost algorithm

- 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 of training samples.



Repeat T-times:
(add a weak classifier)

Select a feature that minimizes weighted classification error and build a weak classifier with that feature.

Reweight the examples:
Incorrectly classified $\Rightarrow$ higher weights
Correctly classified $\Rightarrow$ lower weights

Final classifier is a combination of weak classifiers, which are weighted according to their error.

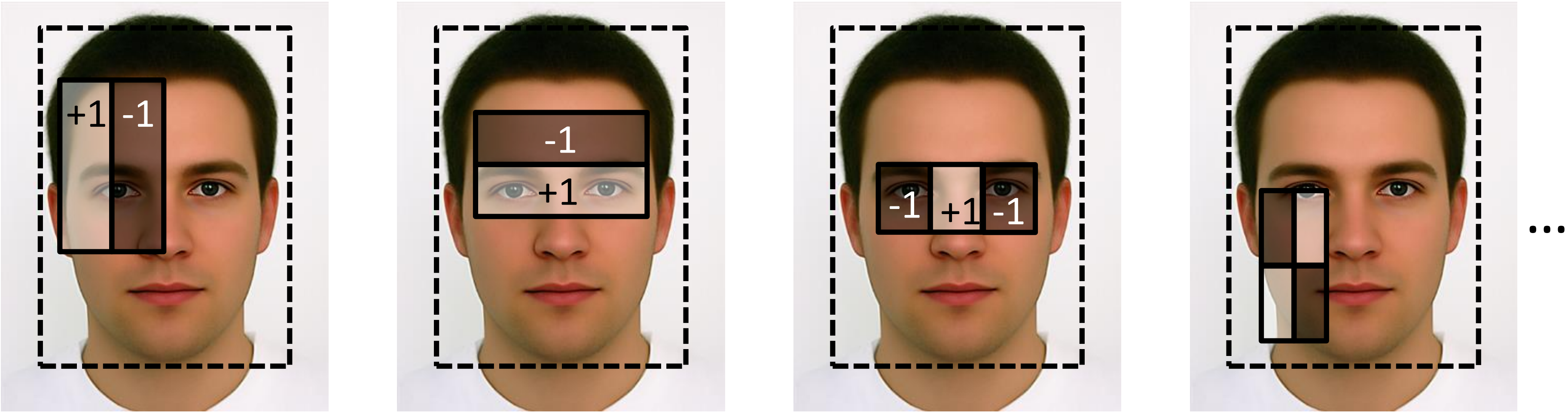
[Freund & Schapire 1995]

Face detection

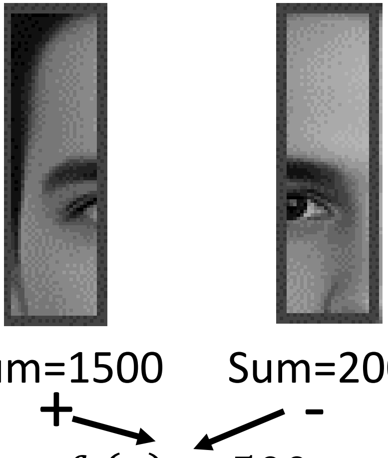
- To apply in real-time applications
    1. Feature extraction should be **fast**
    2. Classifier application should be **fast (weak classifiers = fast evaluation)**

Computing features

Simple rectangular filters as feature extractors



$f_1(x)$ $f_2(x)$ $f_3(x)$ $f_4(x)$

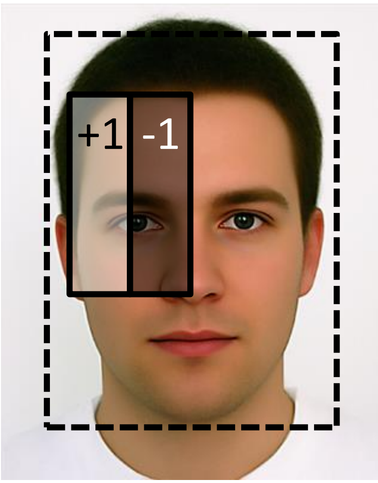


$f_1(x) = 500$

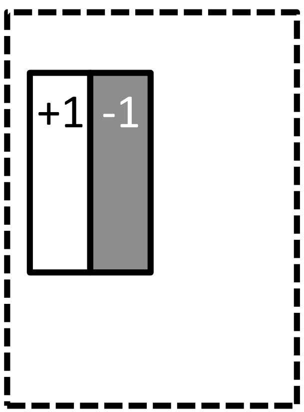
The output of each feature is the difference between the intensity in „black“ and „white“ regions.
Black is weighted as -1, white as +1.

Computing features

Simple rectangular filters as feature extractors



$f_1(x)$

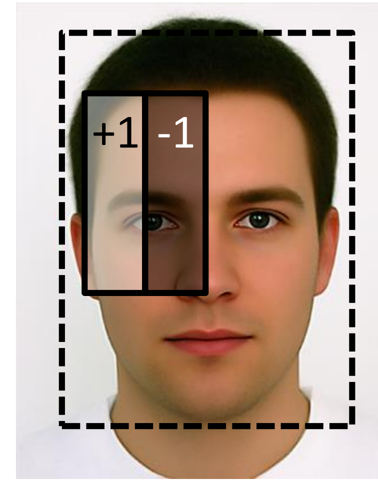


Require evaluation at many displacements and multiple scales!

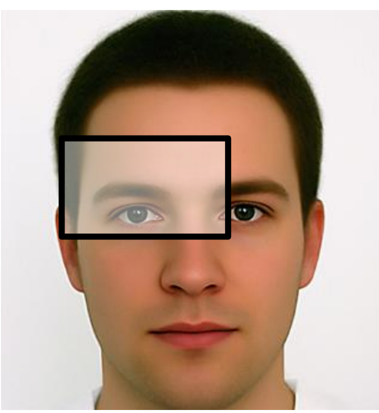
Possible to evaluate such a simple filter efficiently!

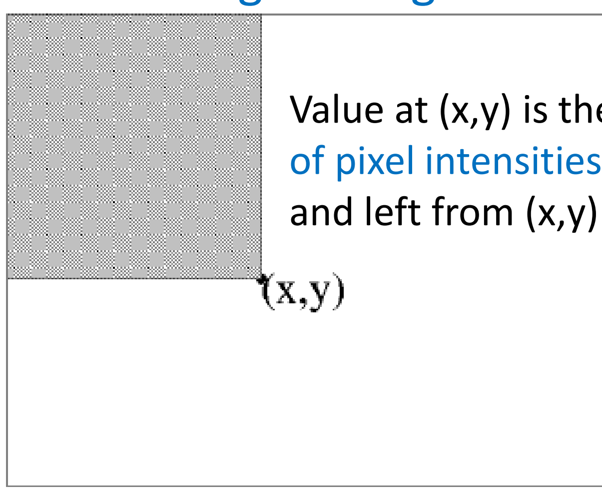
Efficient computation – Integral images

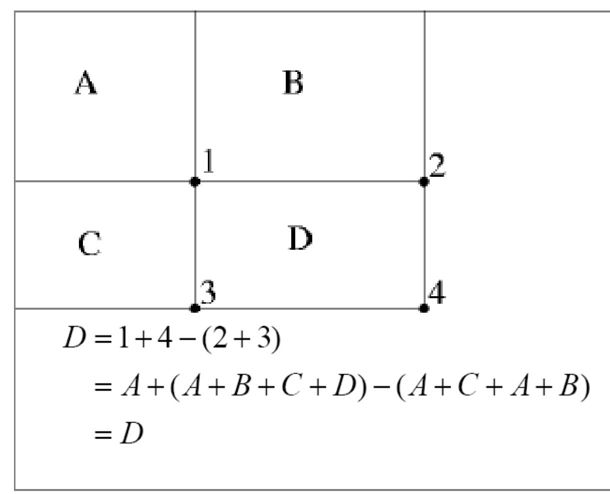
- Our filters are based on sums of intensities within rectangular regions.
- This can be done in constant time for arbitrary large region!
- Require precomputing integral image.



Integral image

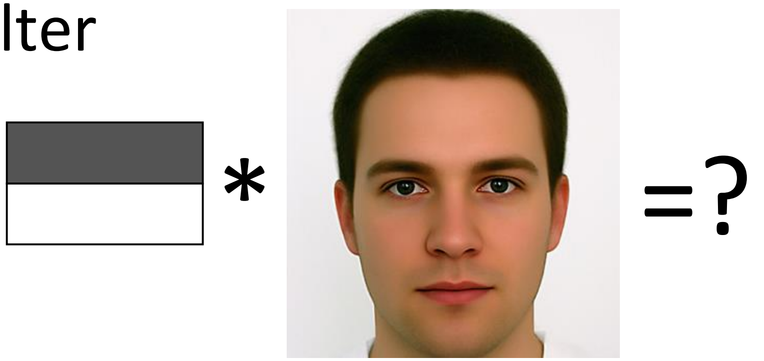


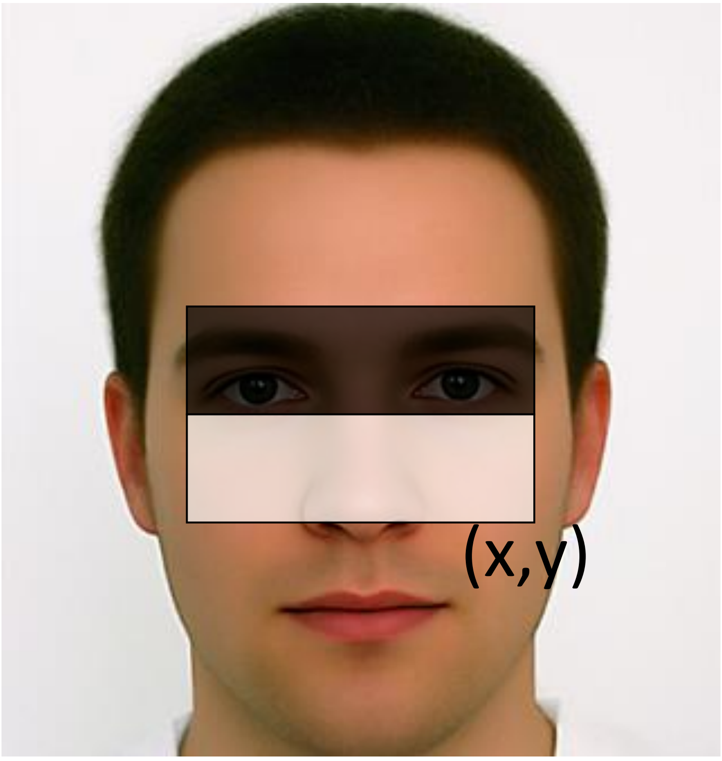


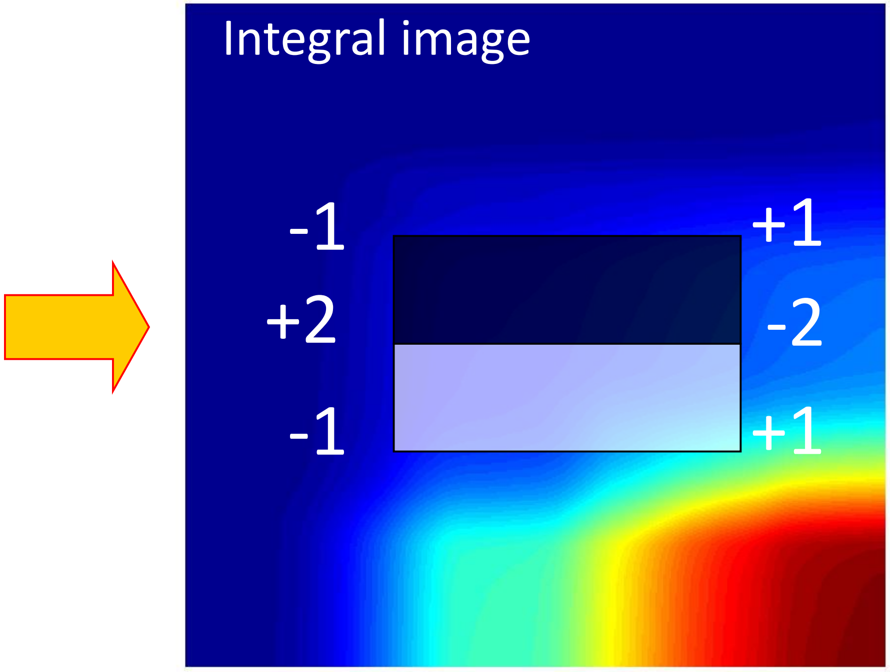


Efficient computation – Integral images

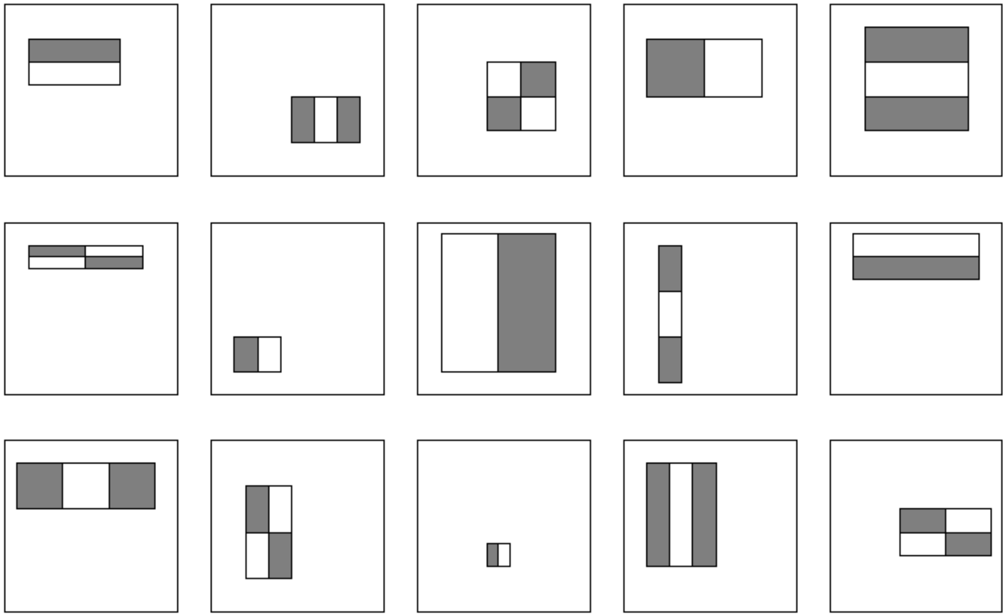
- Consider a more complex filter







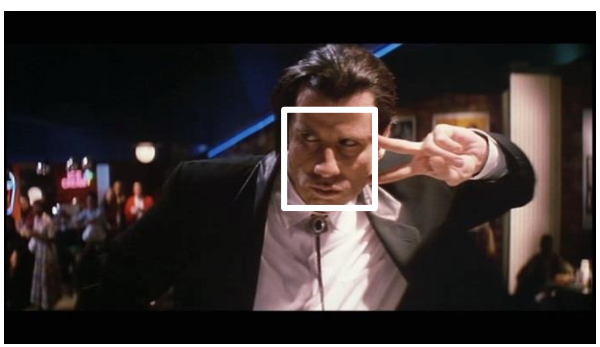
Large collection of filters



etc...

Account for all possible parameters:
position, scale, type

More than 180,000 different features in a 24x24 window.

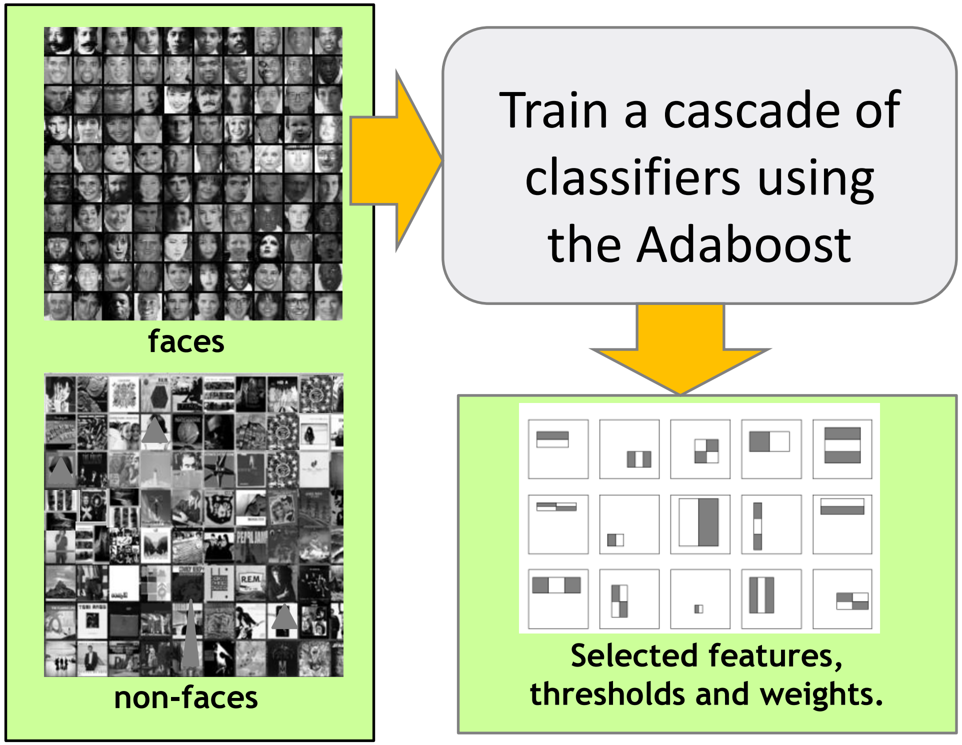


Apply Adaboost for
(i) selecting most informative features and
(ii) composing a classifier.

[Viola & Jones, CVPR 2001]

Slide credit: Kristen Grauman

Training Adaboost for face detection



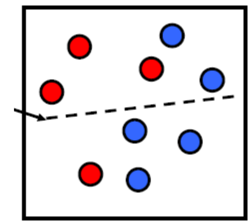
AdaBoost algorithm

- 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 $\alpha_t = \log \frac{1}{\beta_t}$

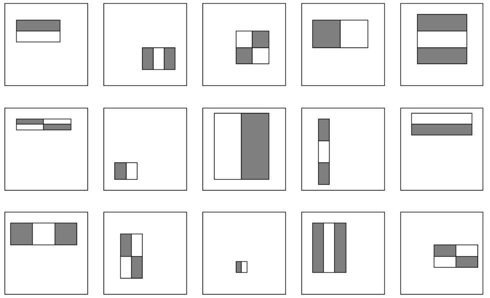
Start with uniform weights of training samples.



$\{x_1, \ldots x_n\}$

Repeat T-times

Build a test classifier along each feature. Calculate weighted classification error for each feature and choose a feature with smallest error (and its classifier).

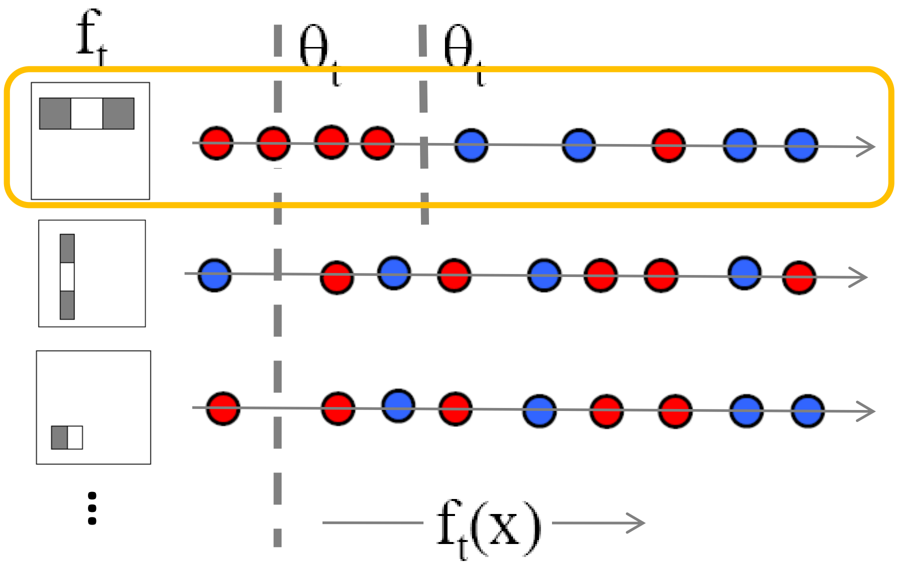


...

[Freund & Schapire 1995]

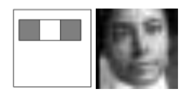
Feature selection and classification: Adaboost

- In each round select a single feature and threshold that best separate positive (faces) and negative (non-faces) examples given the weighted error.



Outputs of rectangular filters (features) for faces and non-faces.

Obtained weak classifier:



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

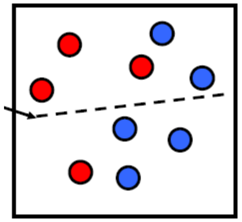
For next round of training reweight the training examples using the errors and select the next feature-threshold pair.

[Viola & Jones, CVPR 2001]

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 of training samples.



$\{x_1, \ldots x_n\}$

Repeat T-times

Build a test classifier along each feature. Calculate weighted classification error for each feature and choose a feature with smallest error (and its classifier).

Reweight the examples:
Incorrectly classified $\Rightarrow$ higher weights
Correctly classified $\Rightarrow$ lower weights

Final classifier is a combination of weak classifiers, which are weighted according to their error.

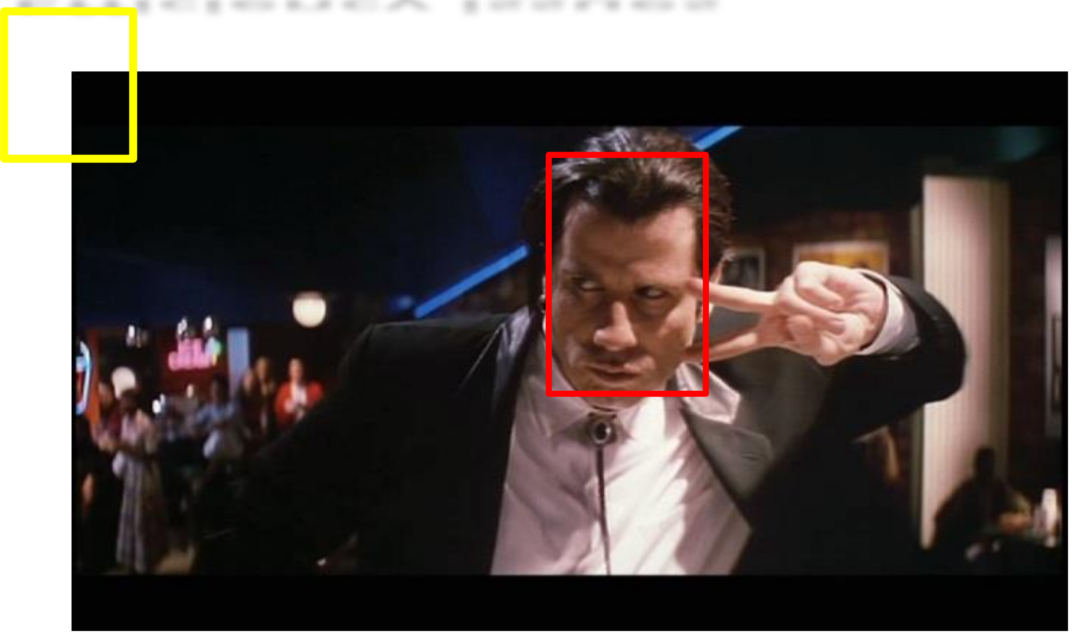
[Freund & Schapire 1995]

Adaboost and feature selection (summary)

- Image features = weak classifiers
- In each round of the Adaboost:
  1. Evaluate each rectangular filter on each training example
  2. Sort examples w.r.t. filter responses
  3. Select the threshold for each filter (with minimum error)
     - Determine the optimal threshold in sorted list
  4. Select the best combination of filter and threshold
  5. The weight of the features is the classification error
  6. Reweight examples

P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004.
(first version appeared at CVPR 2001)

Efficiency issues

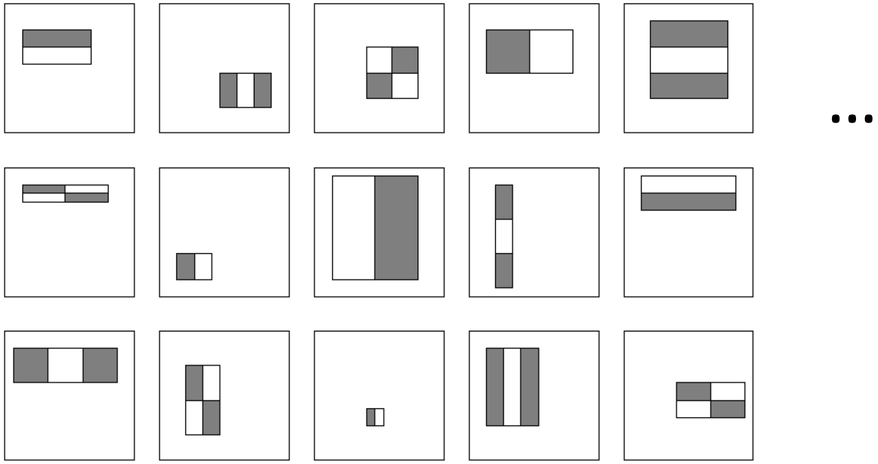


Extract features at each bounding box and apply Adaboost classifier.

- Filter responses can be evaluated fast.
- But each image contains a lot of windows, that we need to classify – potentially great amount of computation!
- How to make detection efficient?

Cascade of classifiers

- Efficient: Apply less accurate but fast classifiers first , to reject the windows that obviously do not contain the particular category!

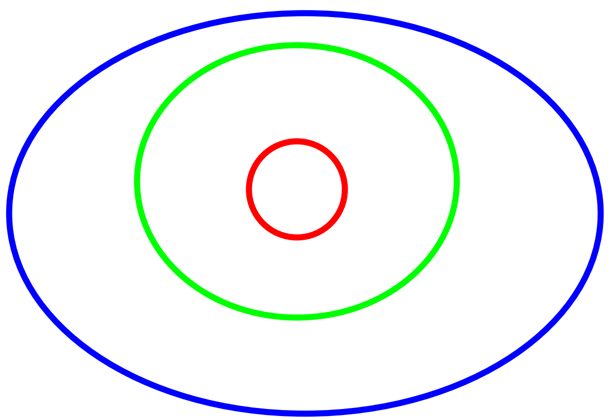


$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

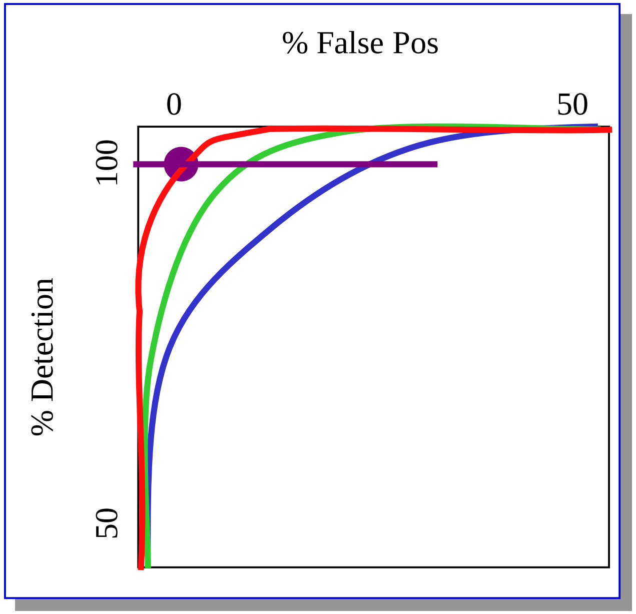
Cascade of classifiers

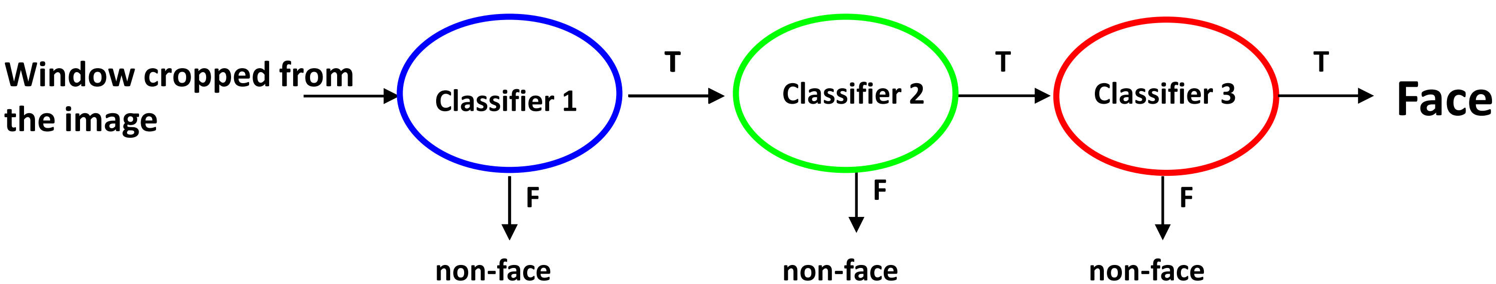
- Chain classifiers from least complex with low true-positive rejection rate to most complex ones:

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

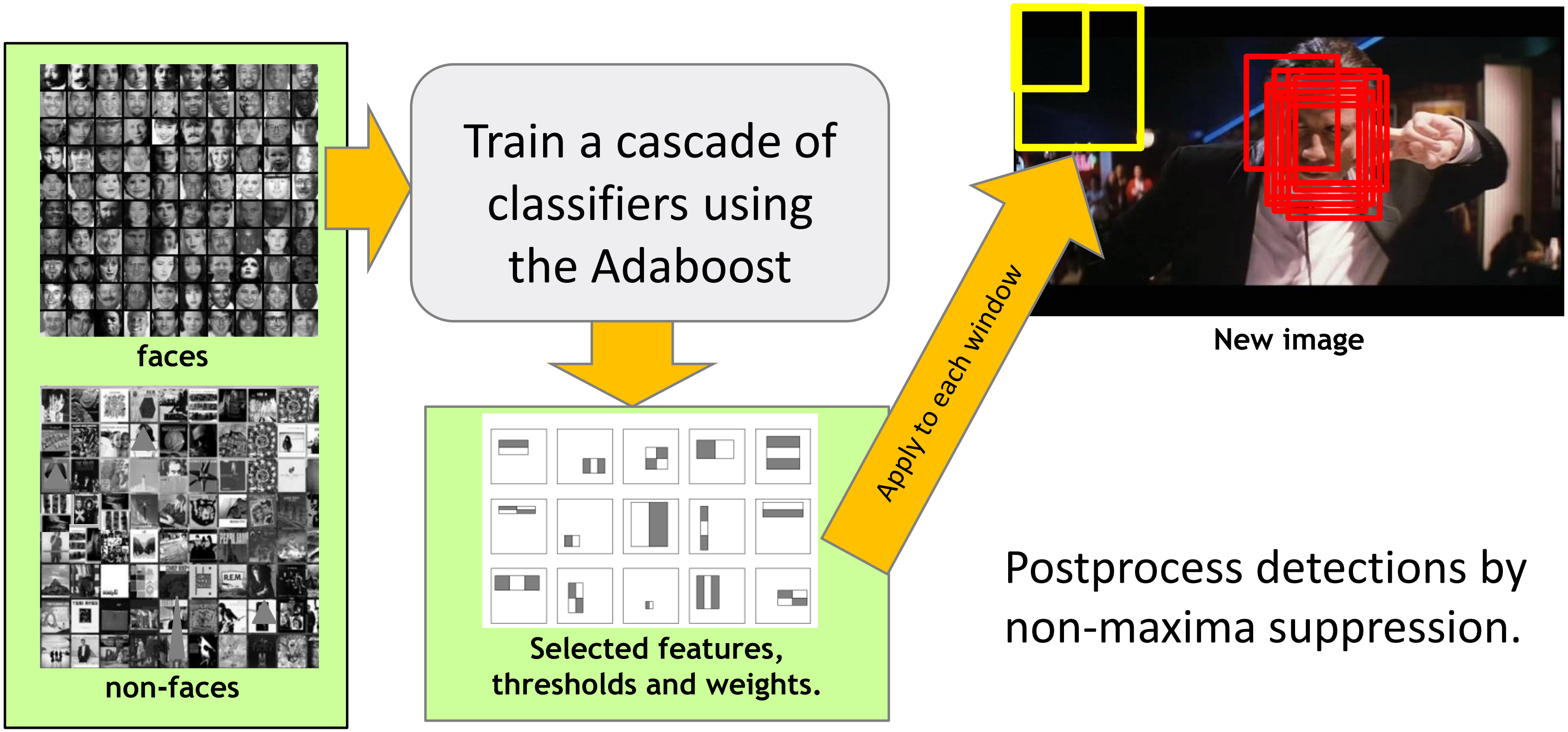


The ROC curve





Viola-Jones face detector

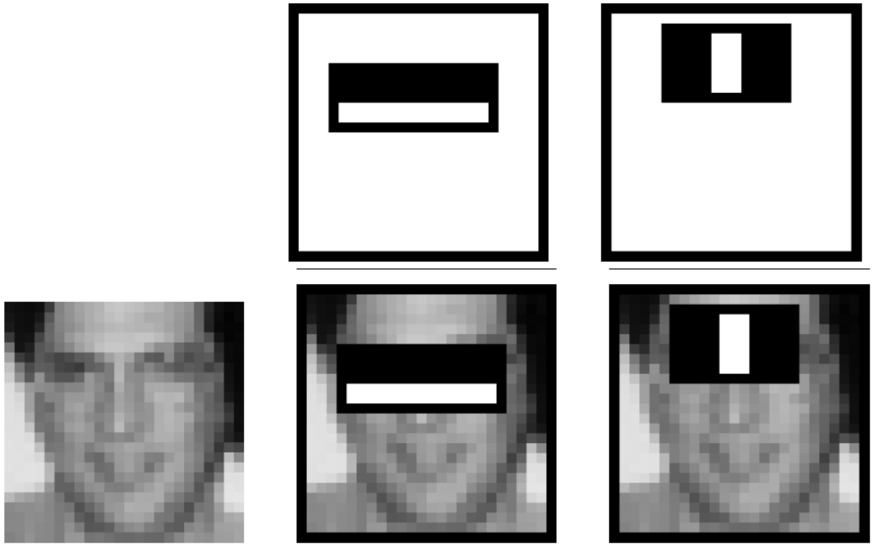


Postprocess detections by non-maxima suppression.

- Train using 5k positives and 350M negatives
- Real-time detector using 38 layers in cascade
- 6061 features in the final layer (classifier)
- [OpenCV implementation: http://sourceforge.net/projects/opencvlibrary/]

Viola-Jones: results

Guess what these correspond to!



Interesting:
First two selected features.

Performance

384x288 images, detection 15 fps on 700 MHz Intel Pentium III desktop (2001).

Training time = weeks!

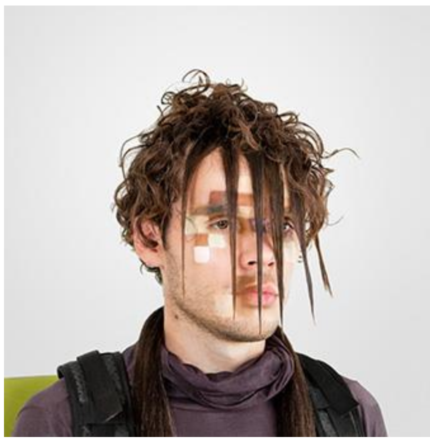
Detection in progress

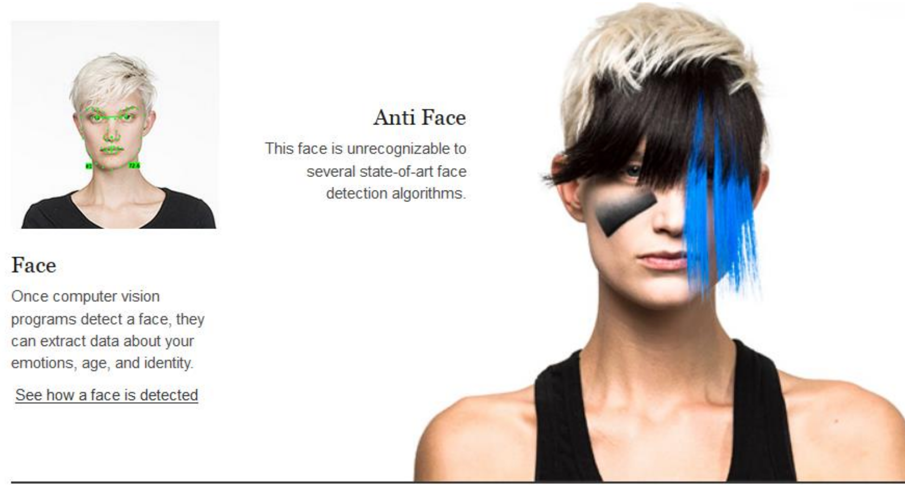
- The video visualizes all the “features”, i.e., filter responses checked in a cascade.
- Observe the increase of cascade once close to face.

http://cvdazzle.com/

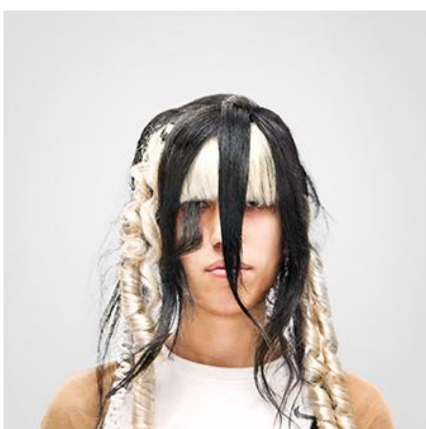
Make your face invisible

- Know how it works?
  Brake it!



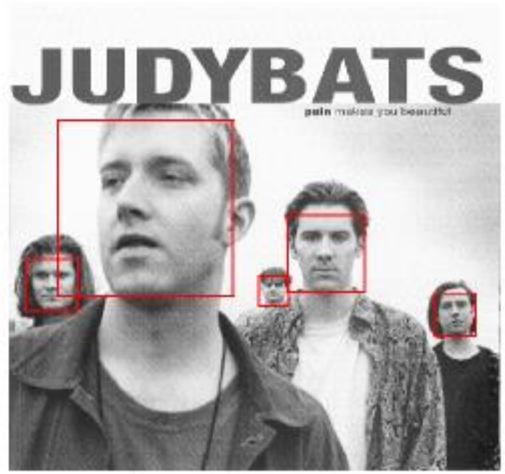


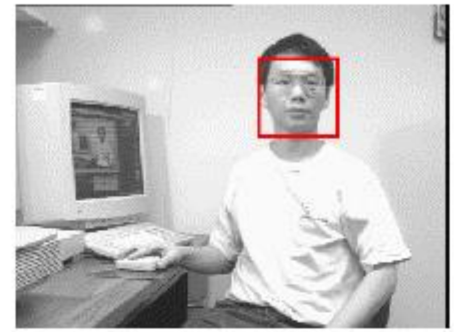
Camouflage from face detection.

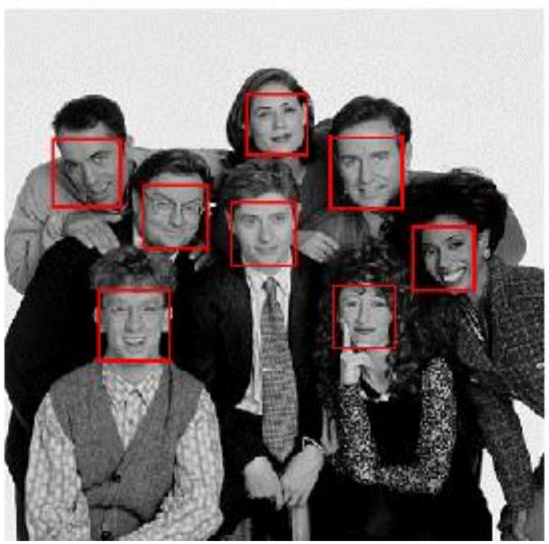


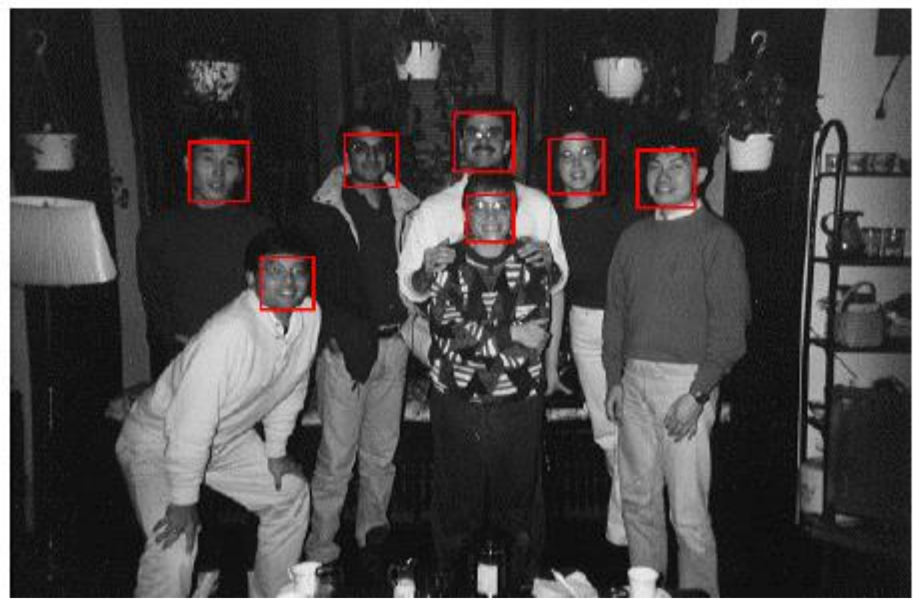
http://cvdazzle.com/

Viola-Jones: results



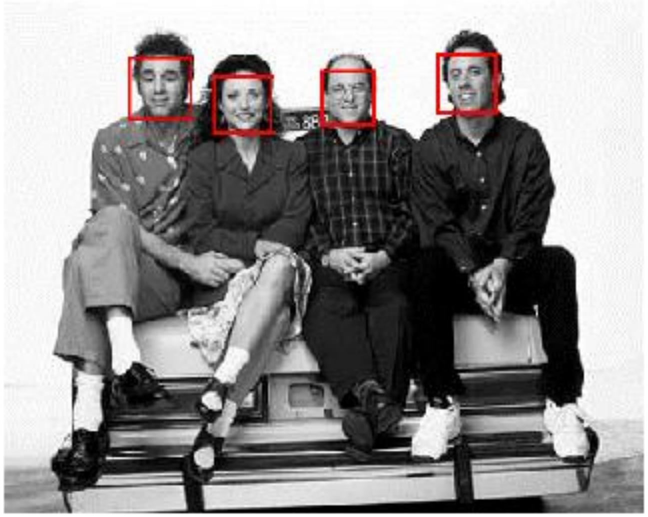


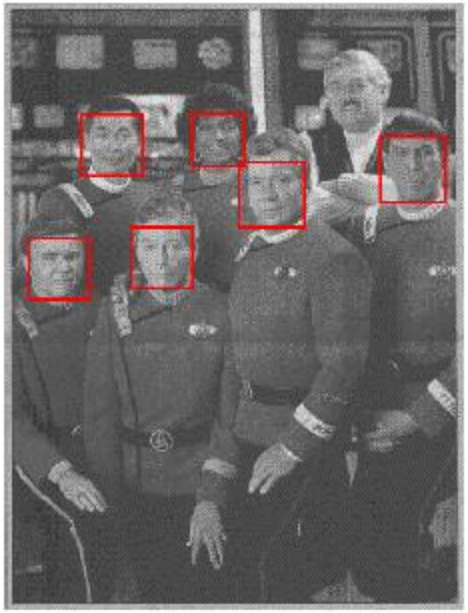


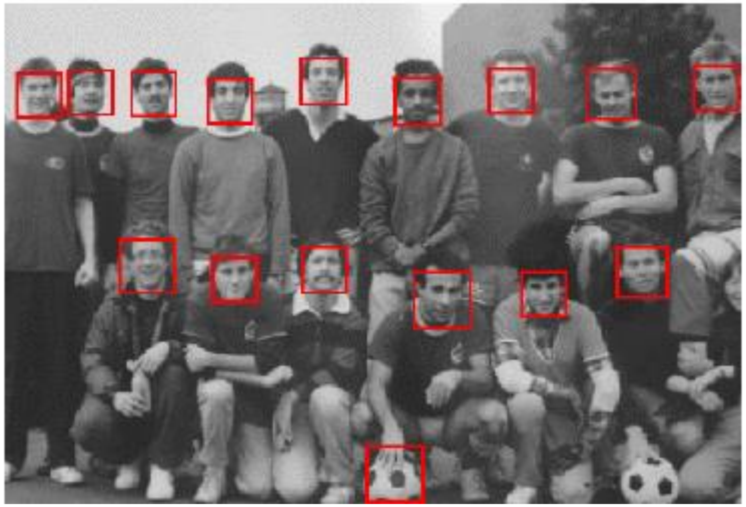


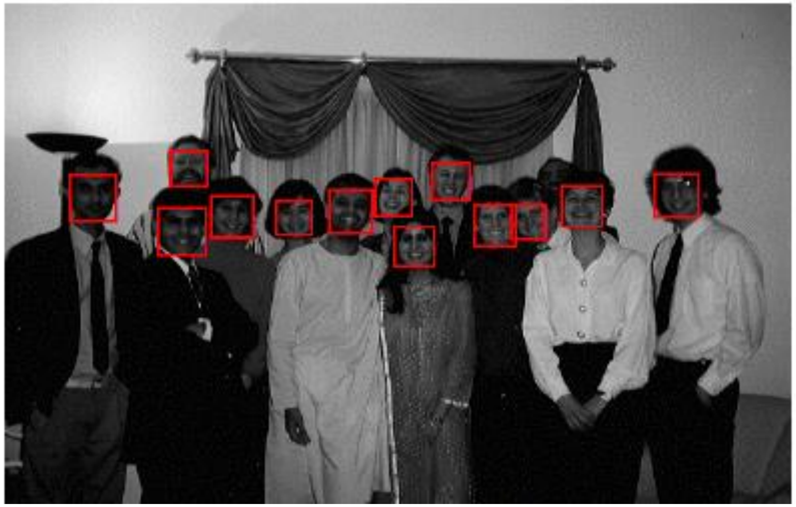
Slide credit: Kristen Grauman

Viola-Jones: results







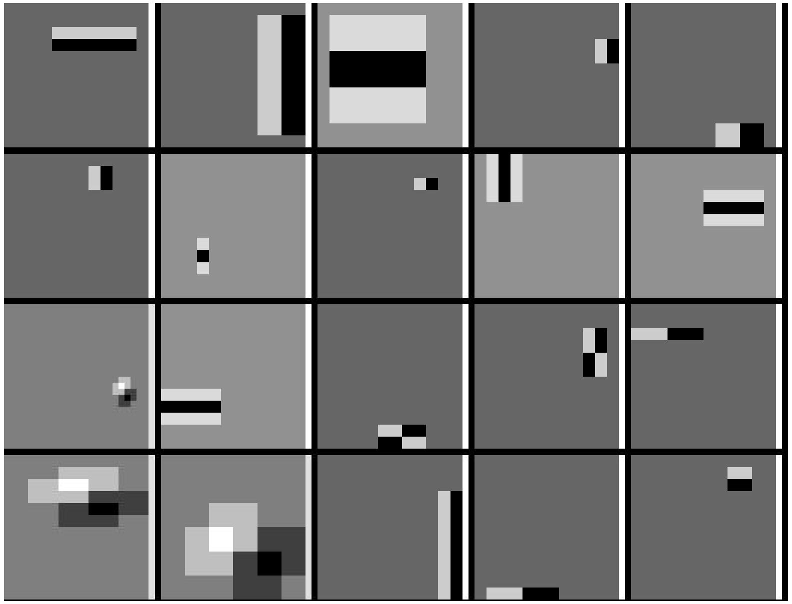


Viola-Jones: results

Note the missing profiles! Detector trained only on frontal faces.

Profile detection

Profile detection requires learning a separate detector using profile faces.

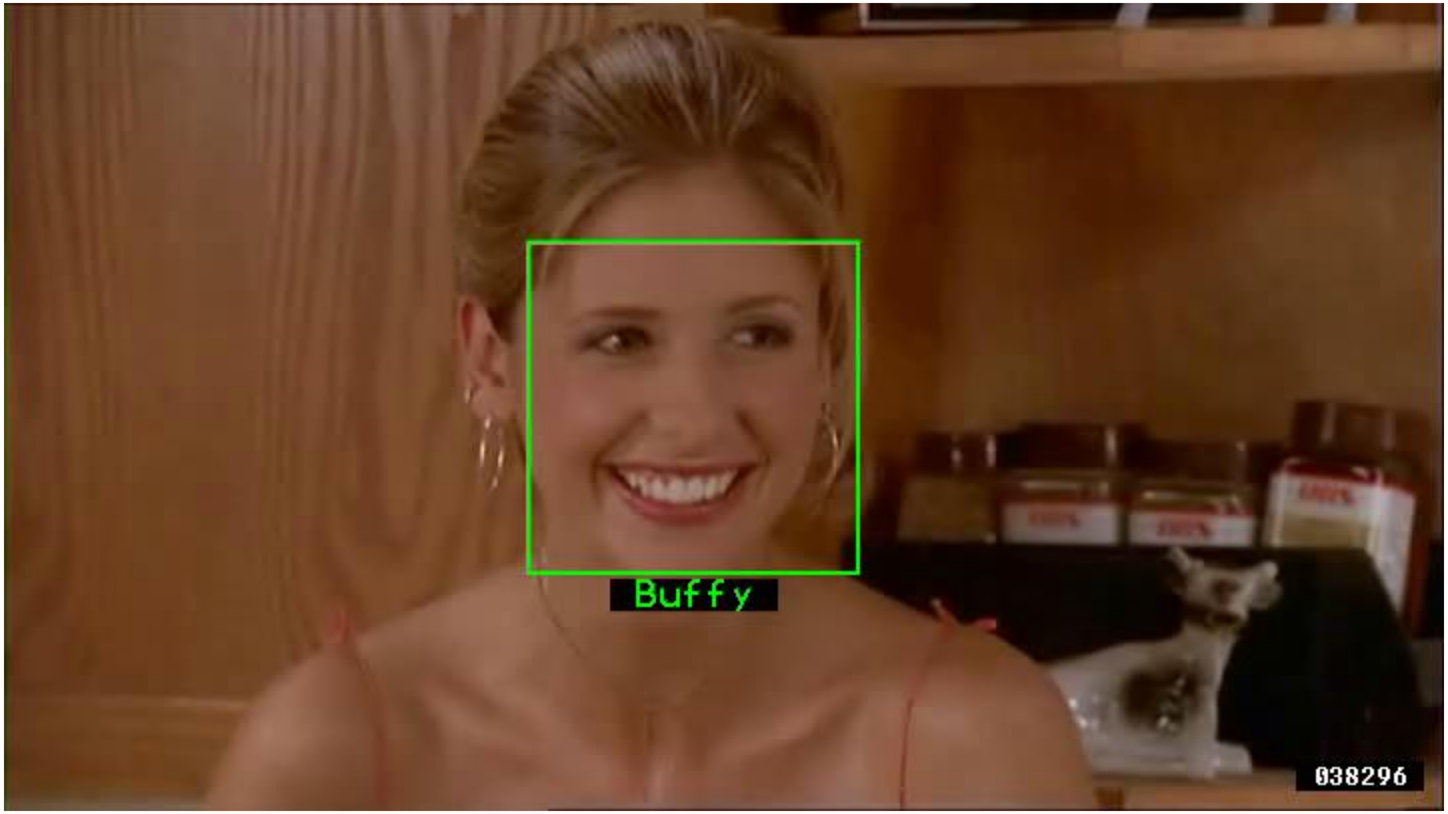


Profile detection

Try it at home!

- Viola & Jones detector was a great success
  - First (!) real-time face detector
  - Lots of improvements since initial publication
- C++ implementation OpenCV [Lienhart, 2002]
  - http://sourceforge.net/projects/opencvlibrary/
- Matlab wrappers for C++:
  - OpenCV version: Mex OpenCV
  - Without OpenCV: http://www.mathworks.com/matlabcentral/fileexchange/20976-fdlibmex-fast-and-simple-face-detection

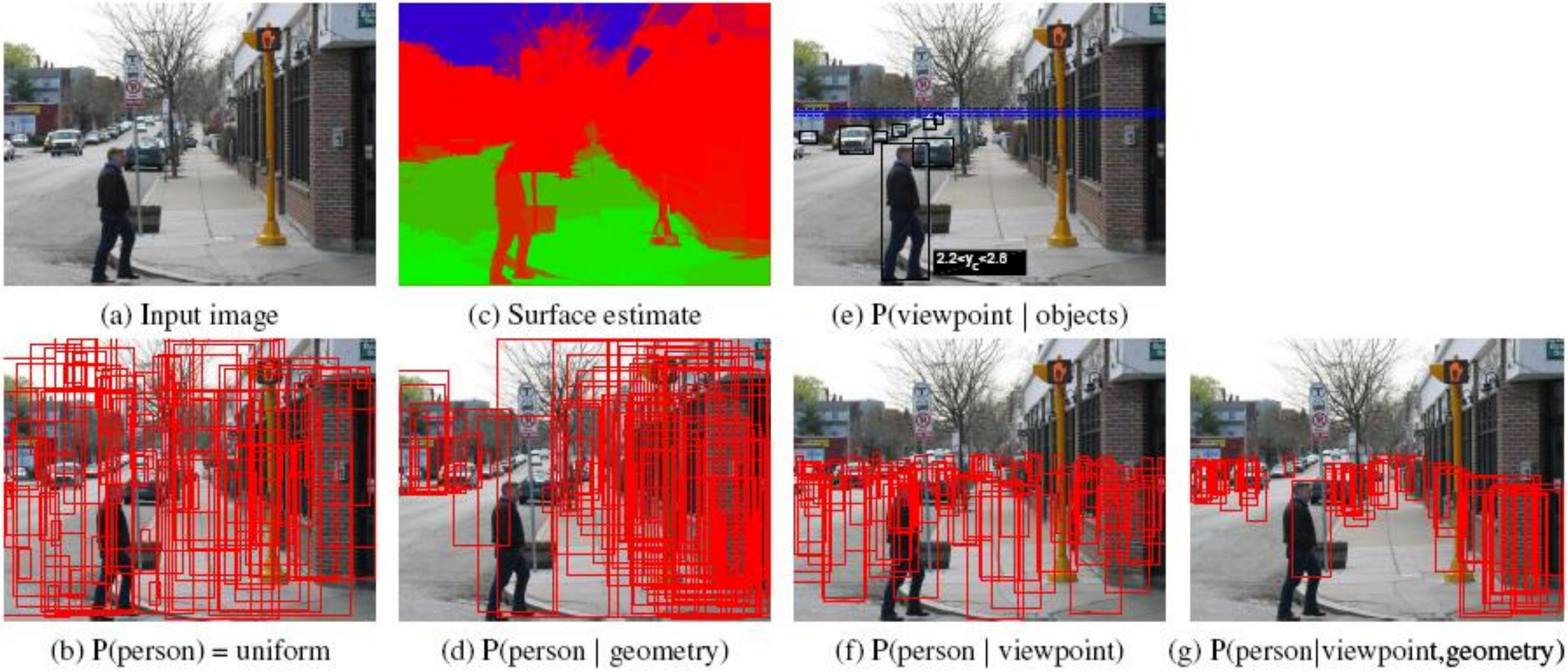
Application example



Frontal faces detected and tracked. Names inferred from subtitles and scripts.

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

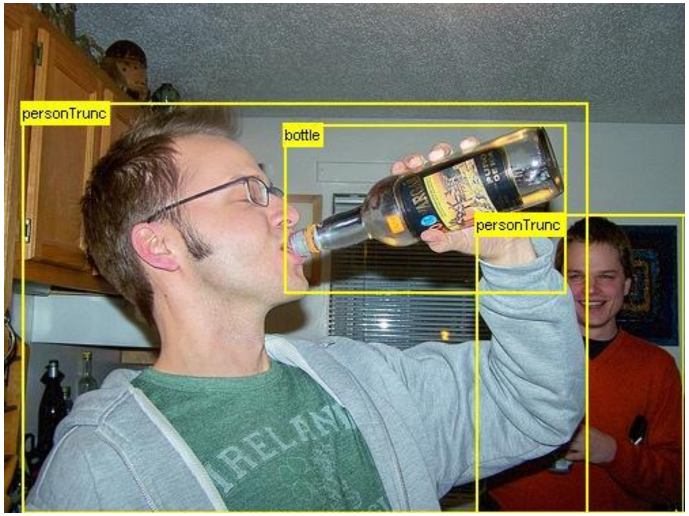
Boosting by context



D. Hoiem, A. Efros, and M. Herbert. [Putting Objects in Perspective.]() CVPR 2006.

Drawbacks remain…

- Some objects poorly described by a single box



- Occlusion not accounted for at all

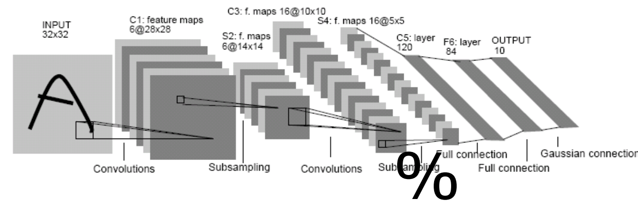
Choice of classifiers

Nearest neighbor

10^6 examples

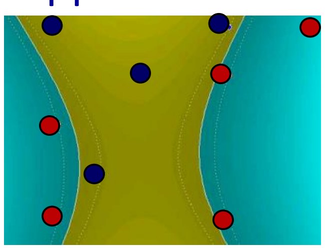
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



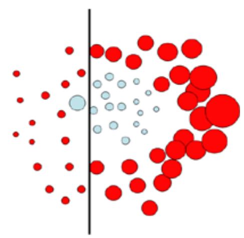
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



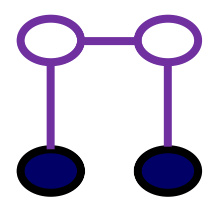
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



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

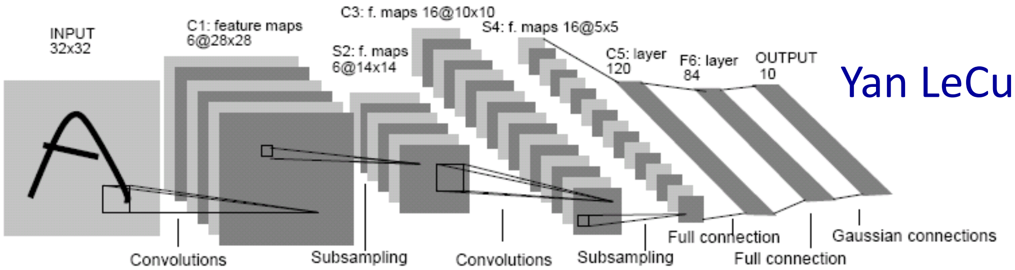
Conditional Random Fields



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

Recent advances in feature learning

- Learning features by convolutional neural networks



Yan LeCun, http://yann.lecun.com/

- First successful application to large-scale object detection by Krizhevsky et al. [1]
- Huge number of parameters to learn (millions).
- Impressive performance.
- Currently a hot research topic (Microsoft, Facebook, Google, etc.)

[1] Krizhevsky et al., ImageNet Classification with Deep Convolutional Neural Networks, NIPS2012

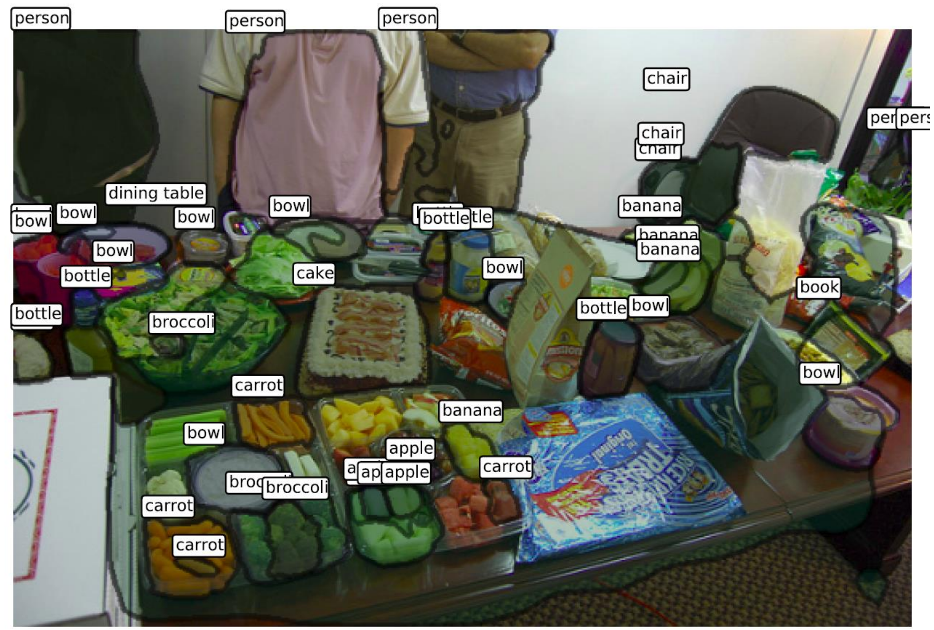
Recent advances in feature learning

- Some examples on [COCO challenge]
- CNN trained for region proposals + CNN trained for classification.

Zagoruyko et al., "FAIR" team at MS COCO & ILSVRC Object Detection and Segmentation Challenge, ICCV2015

Recent advances in feature learning

- Some examples on [COCO challenge]
- CNN trained for region proposals + CNN trained for classification.



Zagoruyko et al., "FAIR" team at [MS COCO & ILSVRC Object Detection and Segmentation Challenge], ICCV2015

References

- David A. Forsyth, Jean Ponce, Computer Vision: A Modern Approach (2nd Edition), (prva izdaja dostopna na spletu)
- R. Szeliski,Computer Vision: Algorithms and Applications, Springer, 2011
- Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004.
- Viola-Jones Face Detector
  - C++ implementation in OpenCV [Lienhart, 2002]
    - http://sourceforge.net/projects/opencvlibrary/
  - Matlab wrappers:
    - http://www.mathworks.com/matlabcentral/fileexchange/19912
- Convolutional neural networks
  - Yan LeCun, http://yann.lecun.com/
  - Caffe, Torch, Tensor flow, etc.