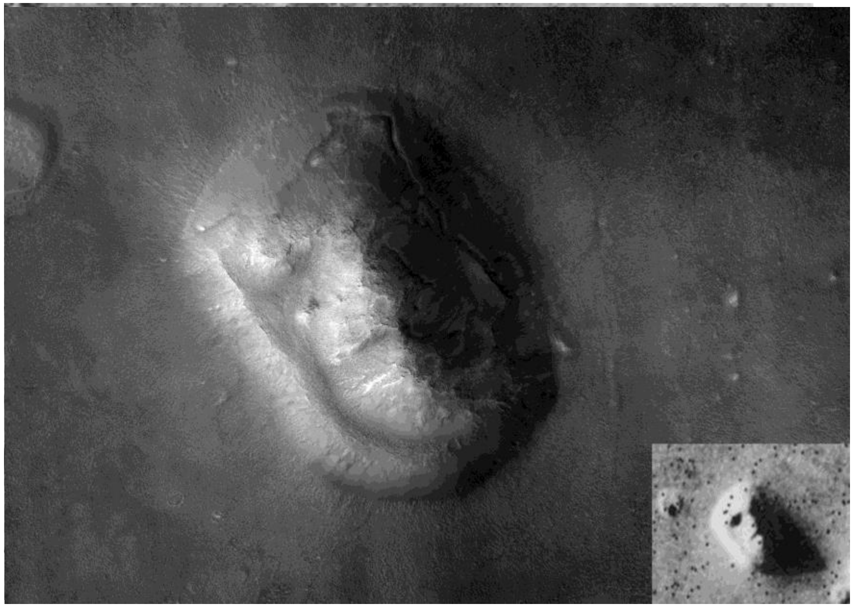
INF 5300 Advanced Topic: Video Content Analysis

Object detection

Asbjørn Berge

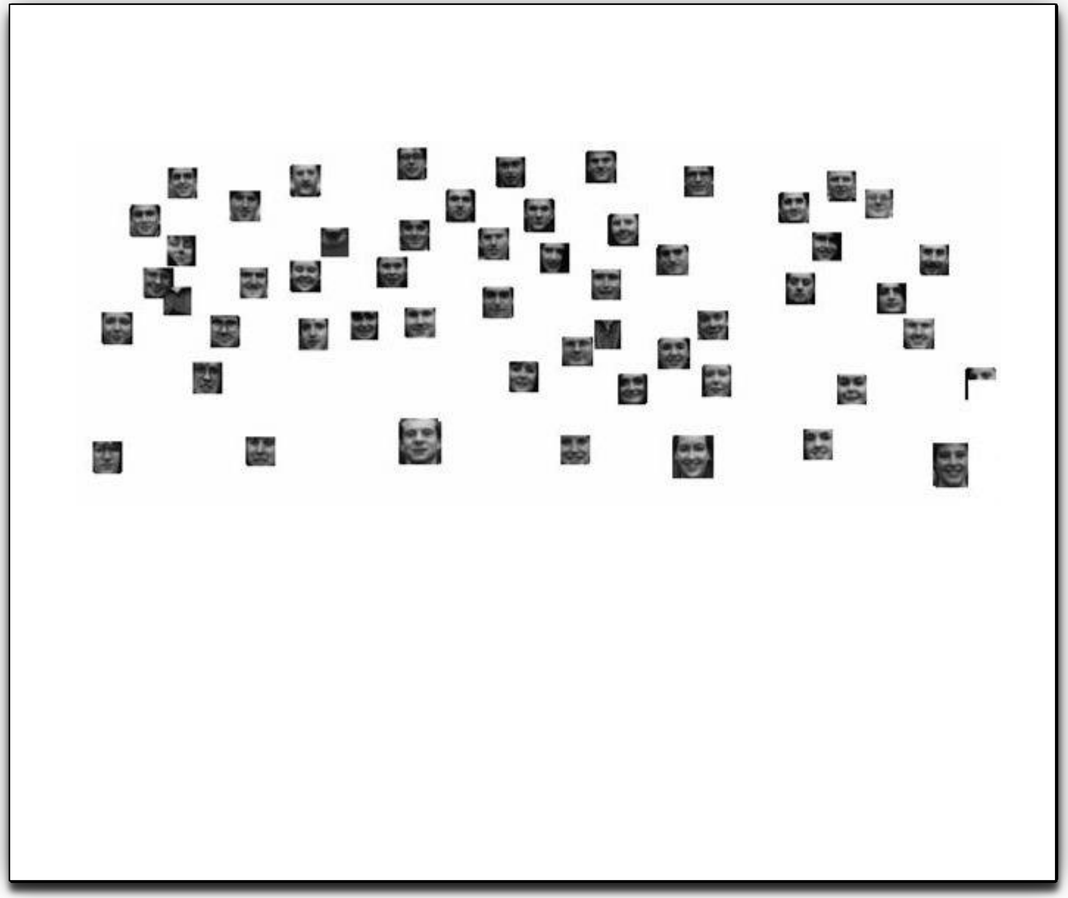
Salvador Dali (1974)
Gala Contemplating the Mediterranean Sea Which at Twenty Meters Becomes the Portrait of Abraham Lincoln – Homage to Rothko (first version)



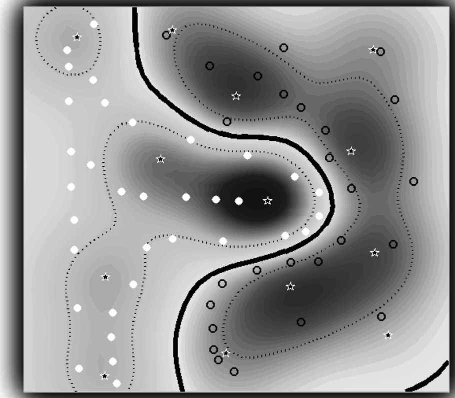
Small part of the Cydonia region, taken by the Viking 1orbiter and released by NASA/JPL on July 25, 1976

Mars Reconnaissance Orbiter image by its HiRISE camera of the "Face on Mars" Viking Orbiter image inset in bottom right corner.

Object Detection by classification: Motivation



Pictures from Romdhani et al. ICCV01



Detection via classification: Main idea

Basic component: a binary classifier

Car/non-car
Classifier

No, not a car.
Yes, a car.

Detection via classification: Main idea

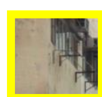
If object may be in a cluttered scene, slide a window around looking for it.

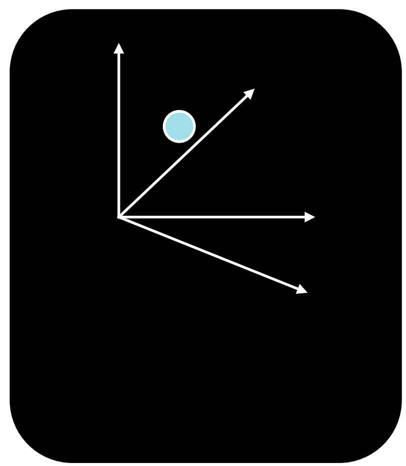
Car/non-car
Classifier

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



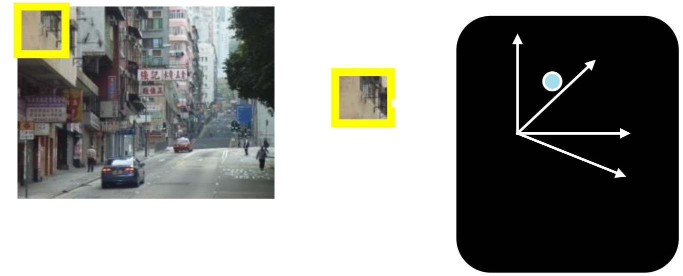


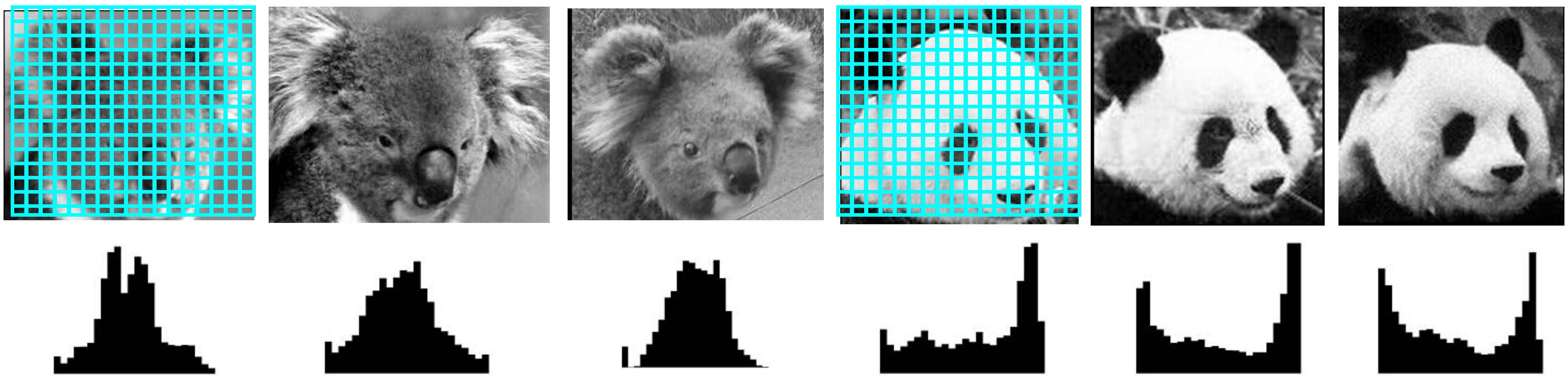
Car/non-car
Classifier

Detection via classification: Main idea

- Consider all subwindows in an image
  - Sample at multiple scales and positions
- Make a decision per window:
  - “Does this contain object category X or not?”
- In this section, we’ll focus specifically on methods using a global representation (i.e., not part-based, not local features).

Feature extraction: global appearance





Simple holistic descriptions of image content

- grayscale / color histogram
- vector of pixel intensities

Eigenfaces: global appearance description

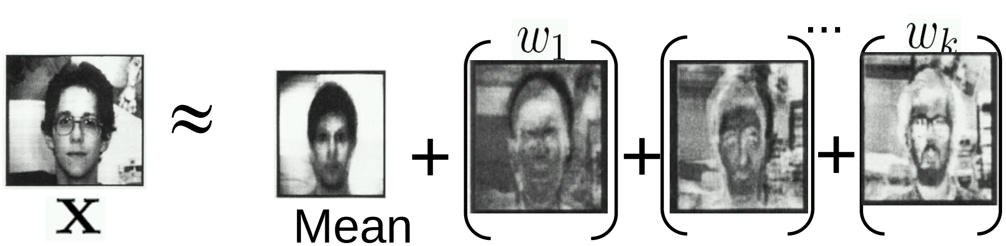
An early appearance-based approach to face recognition

Training images

Mean

Eigenvectors computed from covariance matrix

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



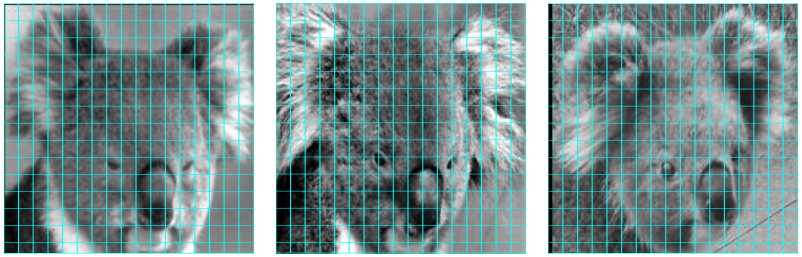
Project new images to “face space”.

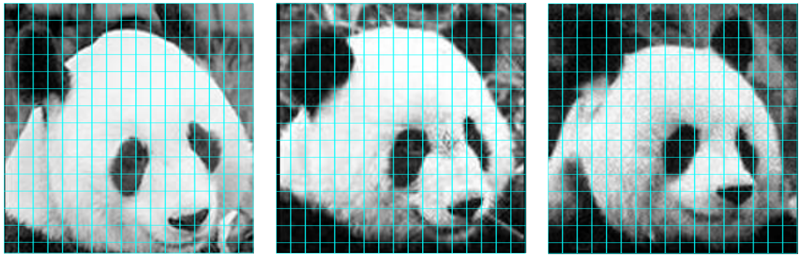
Recognition via nearest neighbors in face space

Turk & Pentland, 1991

Feature extraction: global appearance

- Pixel-based representations sensitive to small shifts



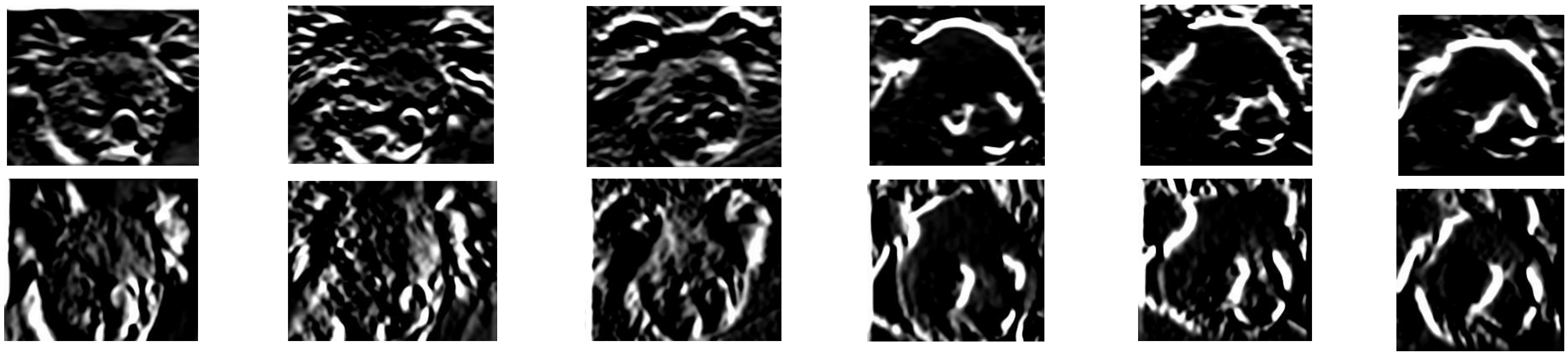


- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation

Cartoon example:
an albino koala

Gradient-based representations

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



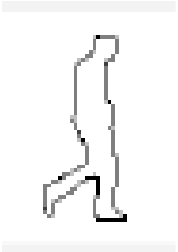
Gradient-based representations: Matching edge templates

- Example: Chamfer matching

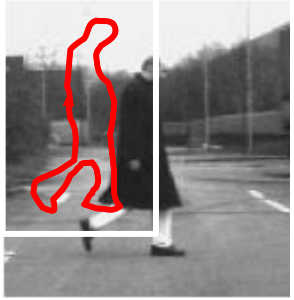
Input image

Edges detected

Distance transform



Template shape



Best match

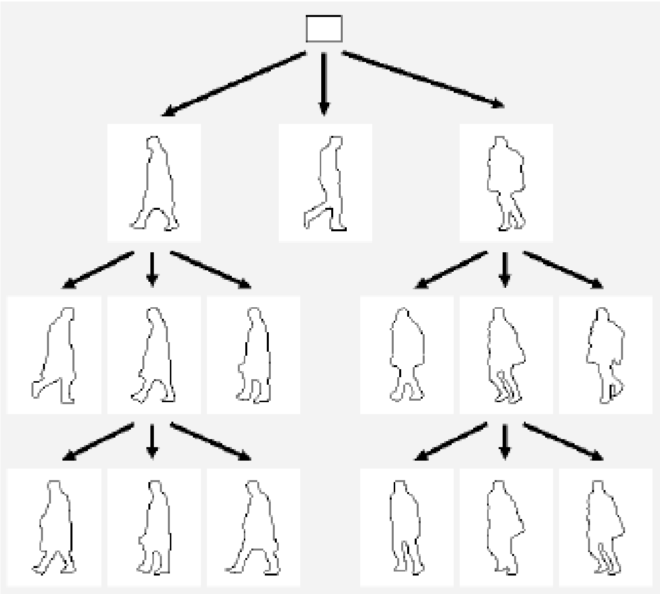
At each window position, compute average min distance between points on template (T) and input (I).

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

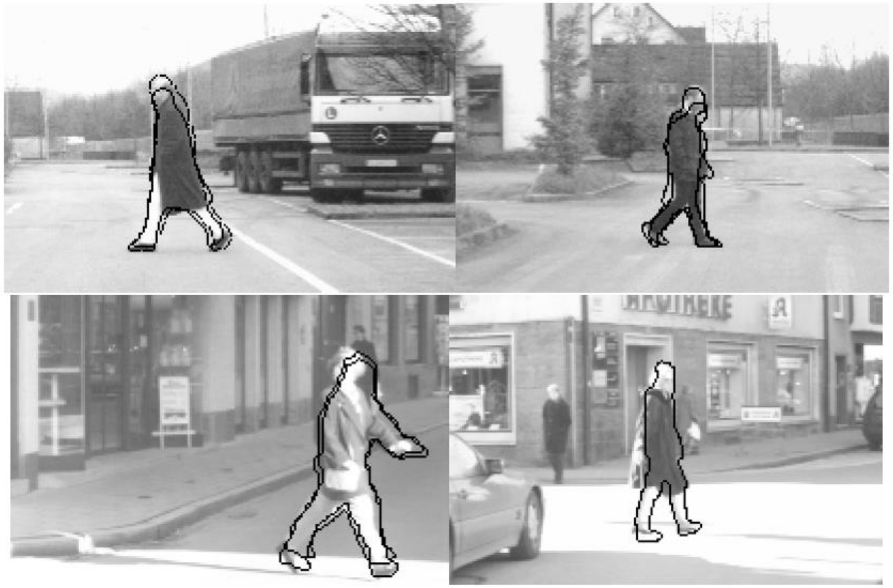
Gavrila & Philomin ICCV 1999

Gradient-based representations: Matching edge templates

- Chamfer matching



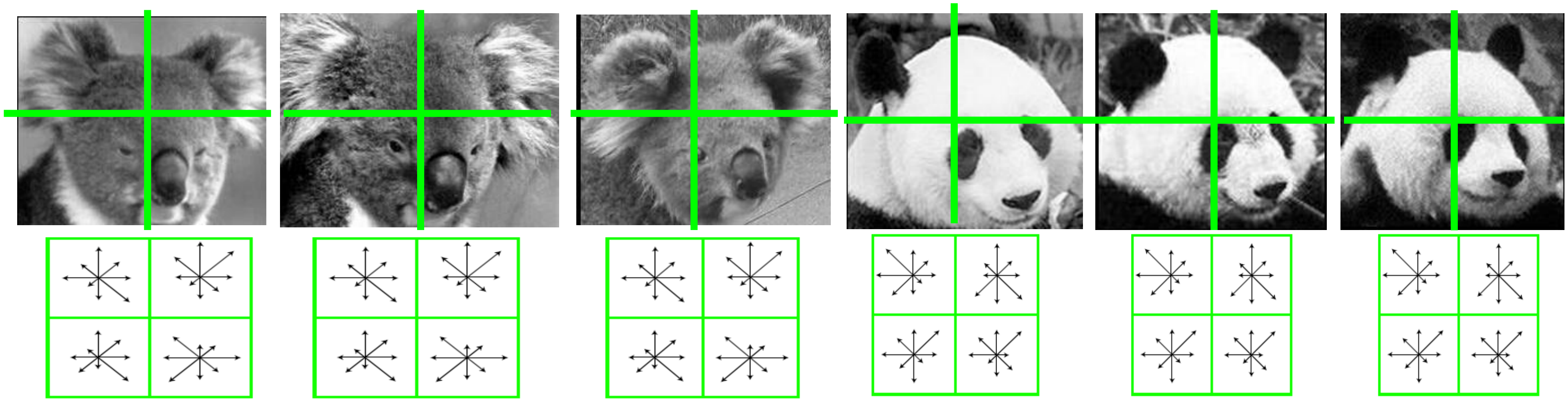
Hierarchy of templates



Gavrila & Philomin ICCV 1999

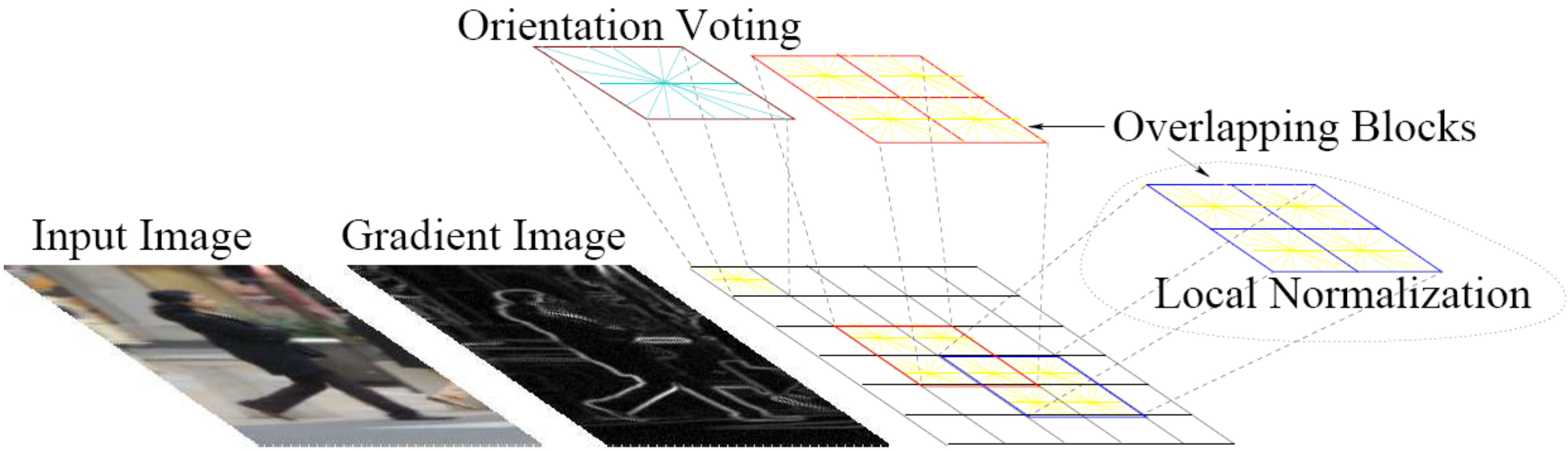
Gradient-based representations

- 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)

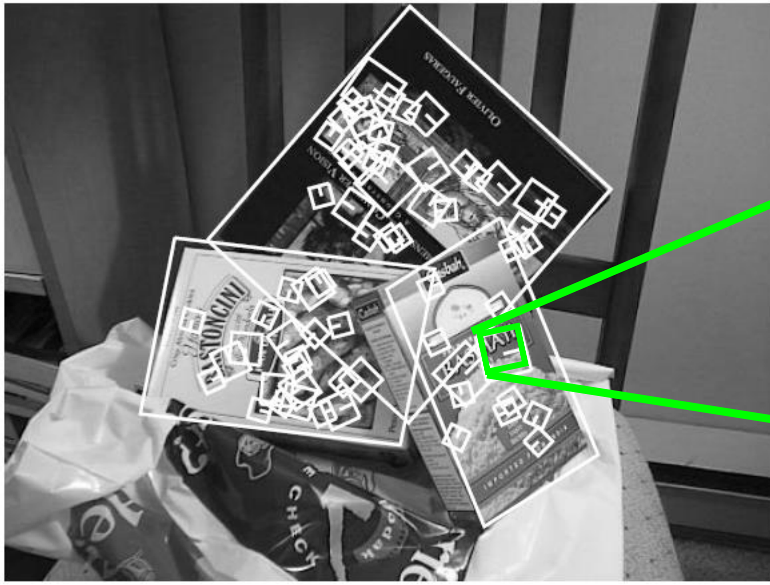


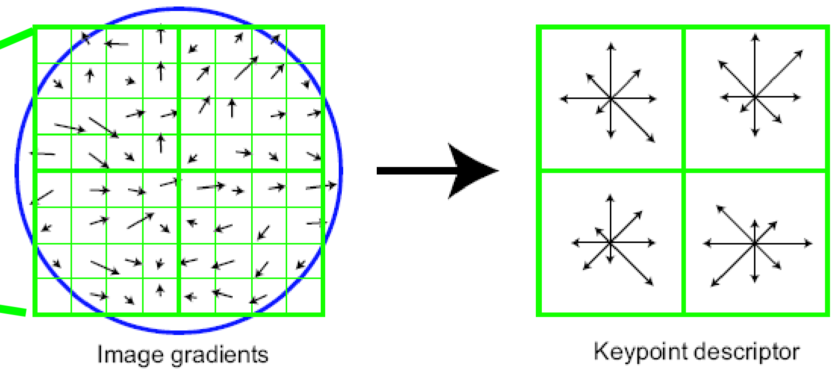
Map each grid cell in the input window to a histogram counting the gradients per orientation.

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

Dalal & Triggs, CVPR 2005

Gradient-based representations: SIFT descriptor





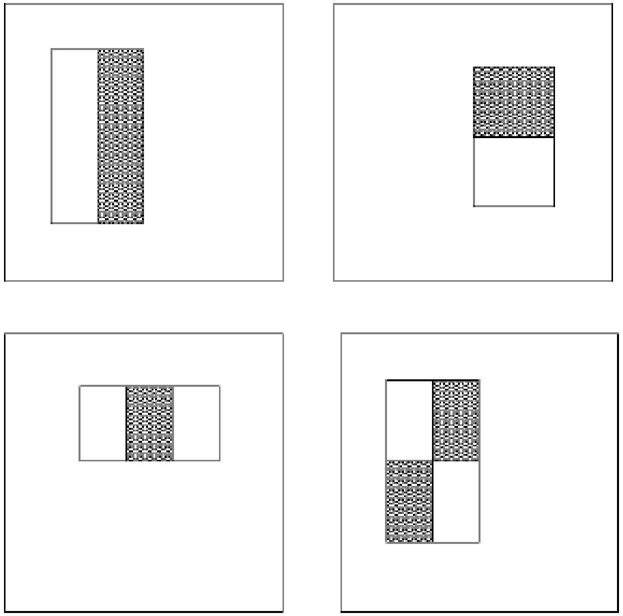
Local patch descriptor (more on this later)

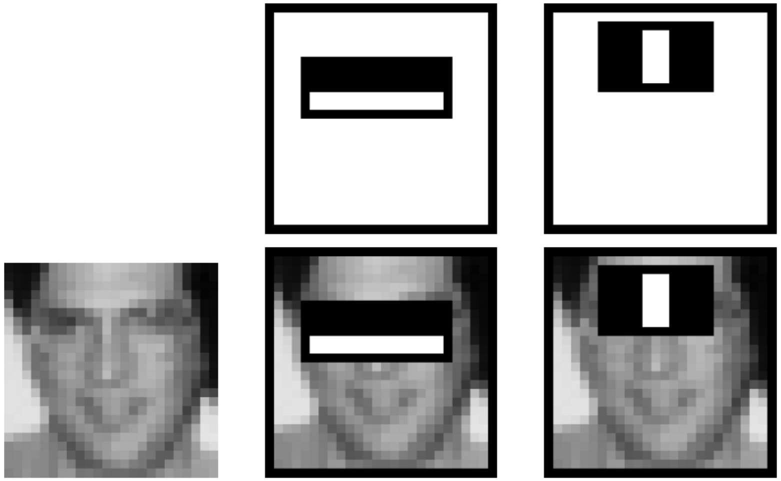
Code: http://vision.ucla.edu/~vedaldi/code/sift/sift.html

Binary: http://www.cs.ubc.ca/~lowe/keypoints/

Lowe, ICCV 1999, Berg, Berg and Malik,2005

Gradient-based representations: Rectangular features





Compute differences between sums of pixels in rectangles

Captures contrast in adjacent spatial regions

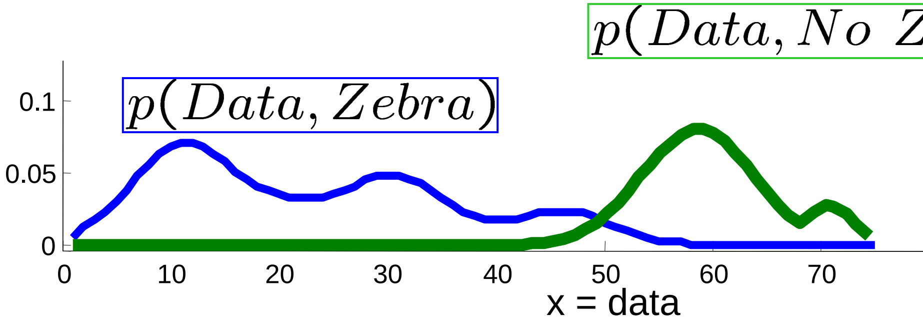
Similar to Haar wavelets, efficient to compute

Viola & Jones, CVPR 2001

Discriminative vs. generative

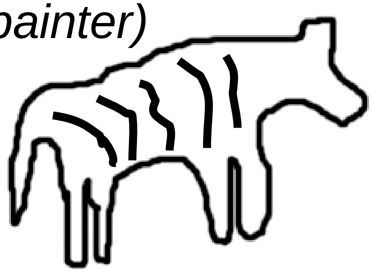
- Generative model
  - separately model class-conditional and prior densities

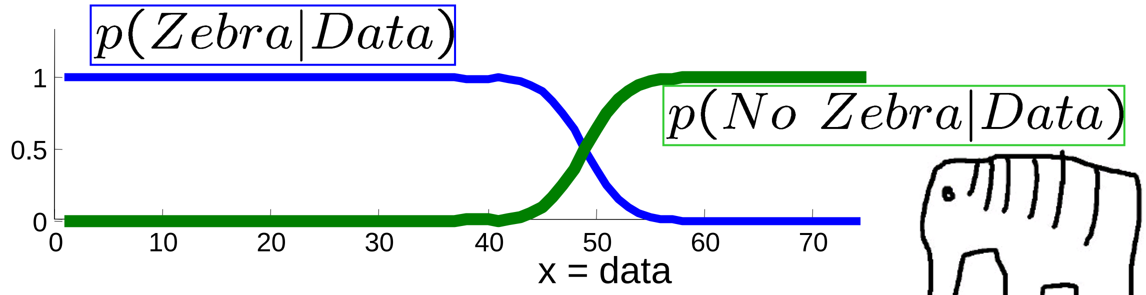
(*The artist*)



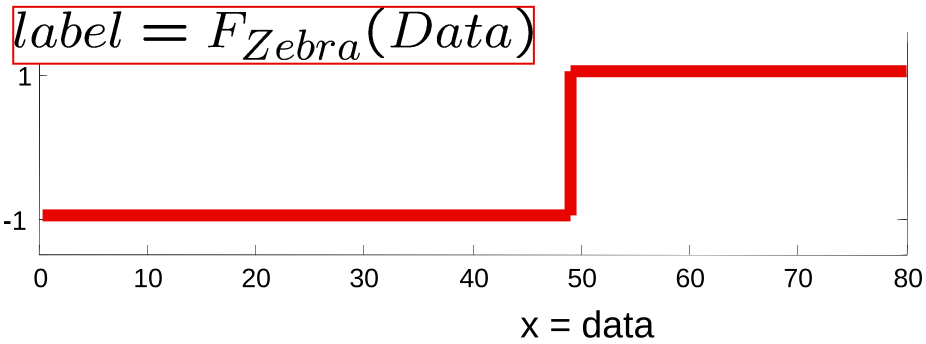
- Discriminative model
  - directly model posterior

(The lousy painter)





- Classification function



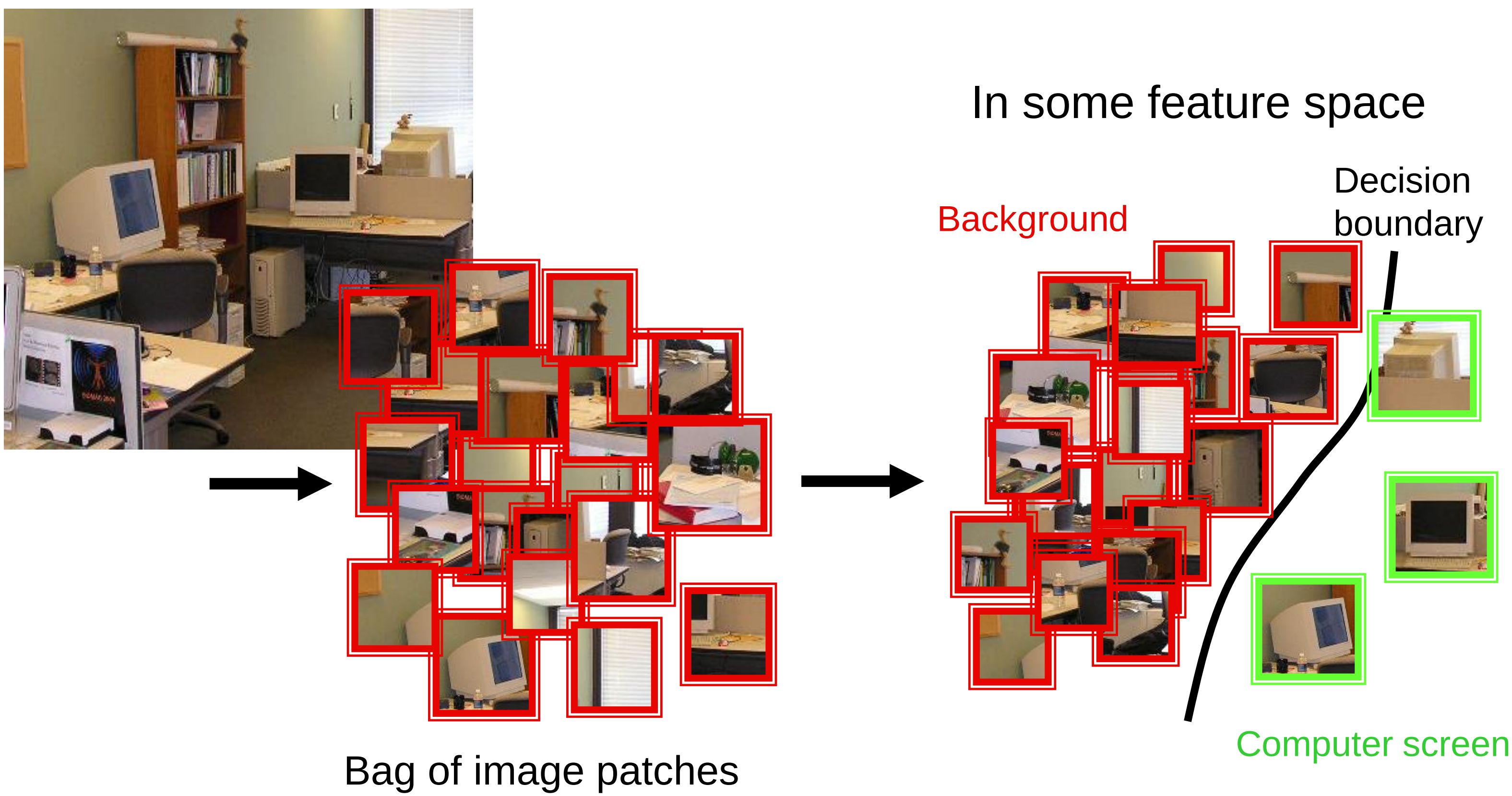
Discriminative vs. generative models

- Generative:
  - + possibly interpretable
  - + can draw samples
  - - models variability unimportant to classification task
  - - often hard to build good model with few parameters

- Discriminative:
  - + appealing when infeasible to model data itself
  - + excel in practice
  - - often can't provide uncertainty in predictions
  - - non-interpretable

Discriminative methods

Object detection and recognition is formulated as a classification problem.
The image is partitioned into a set of overlapping windows
… and a decision is taken at each window about if it contains a target object or not.



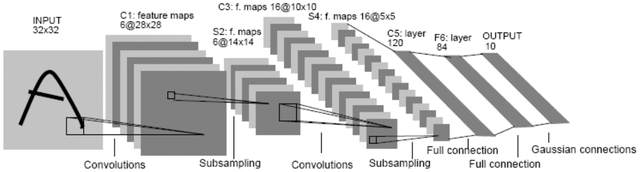
Discriminative methods

Nearest neighbor

10^6 examples

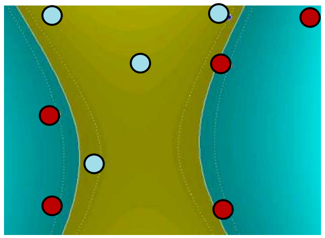
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



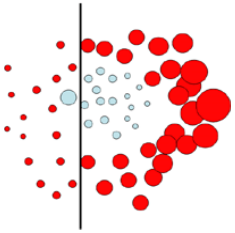
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Support Vector Machines



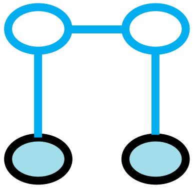
Guyon, Vapnik
Heisele, Serre, Poggio, 2001,…

Boosting



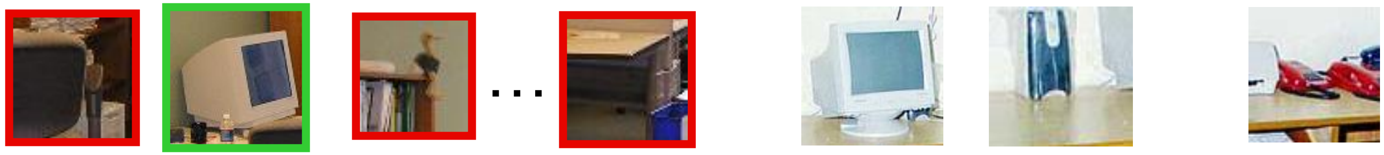
Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,…

Conditional Random Fields



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

Binary classification



Features x = x_1 x_2 x_3 … x_N x_{N+1} x_{N+2} … x_{N+M}

Labels y = -1 +1 -1 -1 ? ? ?

Training data: each image patch is labeled as containing the object or background

Test data

- Classification function

$$\hat{y} = F(x)$$ Where $F(x)$ belongs to some family of functions

- Minimize misclassification error

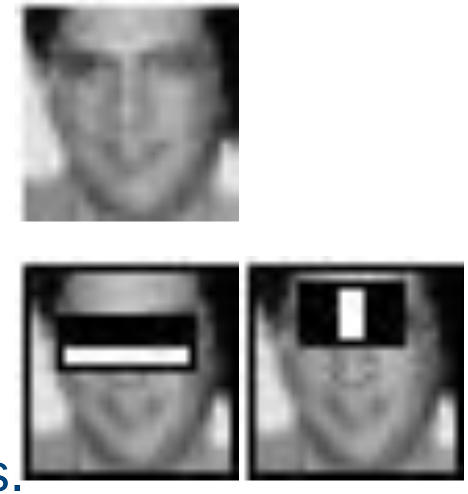
(Not that simple: we need some guarantees that there will be generalization)

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

Features

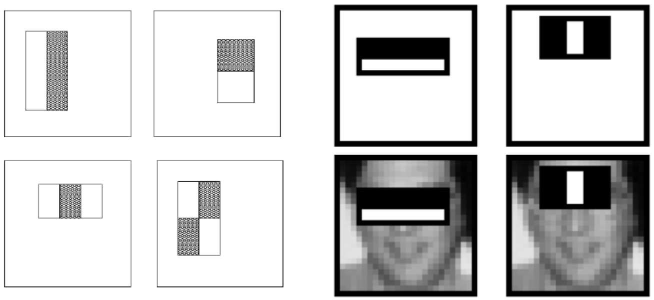


- Can a simple feature (i.e. a value) indicate the existence of a face?
- All faces share some similar properties
  – The eyes region is darker than the upper-cheeks.
  – The nose bridge region is brighter than the eyes.
  – **That is useful domain knowledge**

- Need for encoding of Domain Knowledge:
  – ***Location - Size:*** eyes & nose bridge region
  – ***Value:*** darker / brighter

Feature extraction

"Rectangular" filters

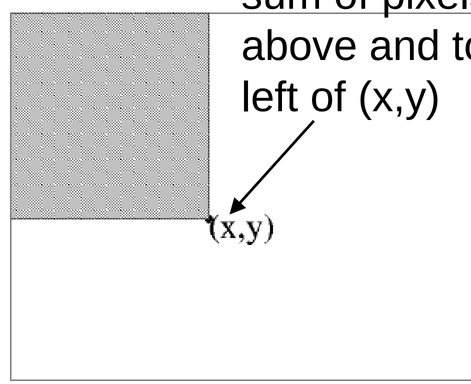


Feature output is difference between adjacent regions

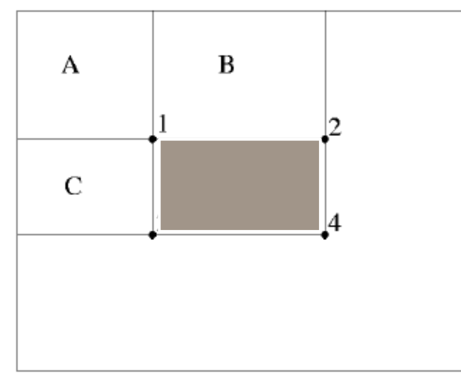
Efficiently computable with integral image: any sum can be computed in constant time

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)



Integral image



$$D = 1 + 4 - (2 + 3)$$
$$= A + (A + B + C + D) - (A + C + A + B)$$
$$= D$$

Viola & Jones, CVPR 2001

Rapid computation of rectangular features

IMAGE

0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

INTEGRAL IMAGE

0	1	2	3
1	4	7	11
2	7	11	16
3	11	16	21

Three goals for a face detector

1. *Feature Computation*: features must be computed as quickly as possible
2. *Feature Selection*: select the most discriminating features
3. *Real-timeliness*: must focus on potentially positive image areas (that contain faces)

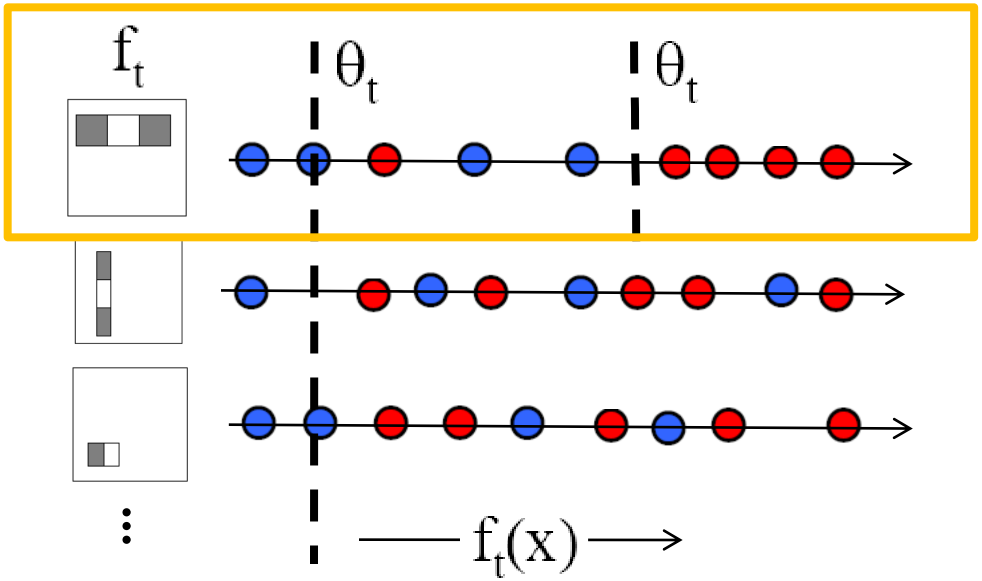
Boosting

- 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

Viola-Jones detector: AdaBoost

- **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:

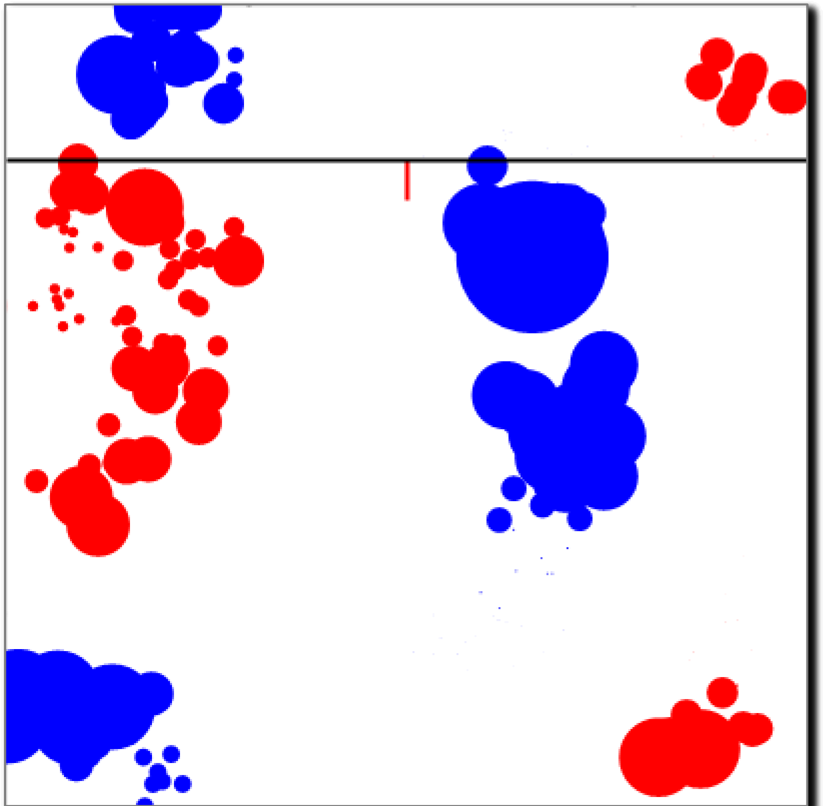
$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

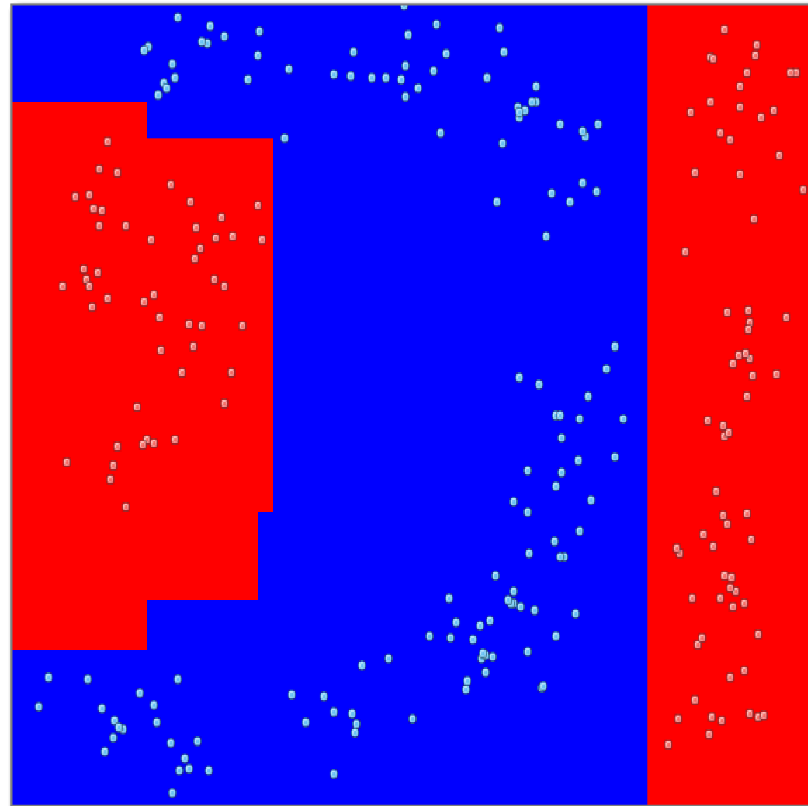
For next round, reweight the examples according to errors, choose another filter/threshold combo.

Boosting

Iteratively reweighting training samples.

Higher weights to previously misclassified samples.





50 rounds

SINTEF

Technology for a better society

AdaBoost

- Stands for "**Ada**ptive" **boost**
- Constructs a "strong" classifier as a linear combination of weighted simple "weak" classifiers

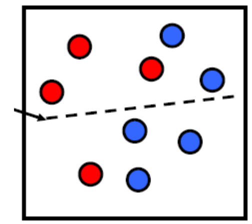
$$\underbrace{F}_{\text{Strong classifier}}(\underbrace{x}_{\text{Image}}) = \underbrace{\alpha_1}_{\text{Weight}} \underbrace{f_1}_{\text{Weak classifier}}(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

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



$\{x_1, \ldots x_n\}$

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.

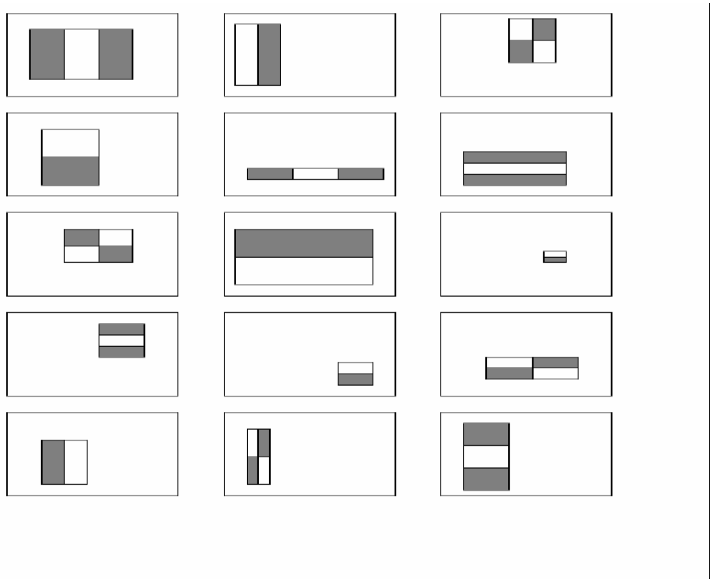
Freund & Schapire 1995

AdaBoost - *Characteristics*

- Features as weak classifiers
  - Each single rectangle feature may be regarded as a simple weak classifier
- An iterative algorithm
  - AdaBoost performs a series of trials, each time selecting a new weak classifier
- Weights are being applied over the set of the example images
  - During each iteration, each example/image receives a weight determining its importance

Feature selection

- Problem: Too many features
  - In a sub-window (24x24) there are ~160,000 features (all possible combinations of orientation, location and scale of these feature types)
  - impractical to compute all of them (computationally expensive)
- We have to select a subset of relevant features – which are informative - to model a face
  - **Hypothesis**: "A very small subset of features can be combined to form an effective classifier"
  - How?
    - AdaBoost algorithm



Relevant feature Irrelevant feature

AdaBoost – *Feature Selection*

Problem

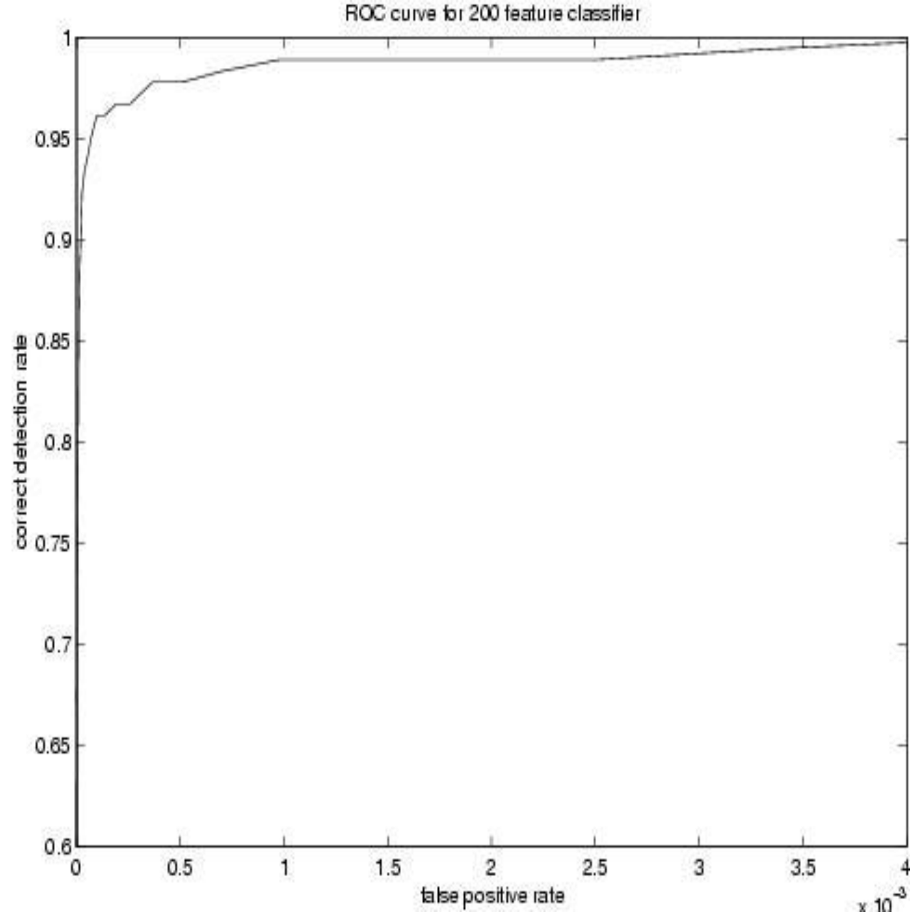
- On each round, large set of possible weak classifiers (each simple classifier consists of a single **feature**) – Which one to choose?
  - choose the most efficient (the one that best separates the examples – the lowest error)
  - choice of a classifier corresponds to choice of a feature
- At the end, the 'strong' classifier consists of T features

Conclusion

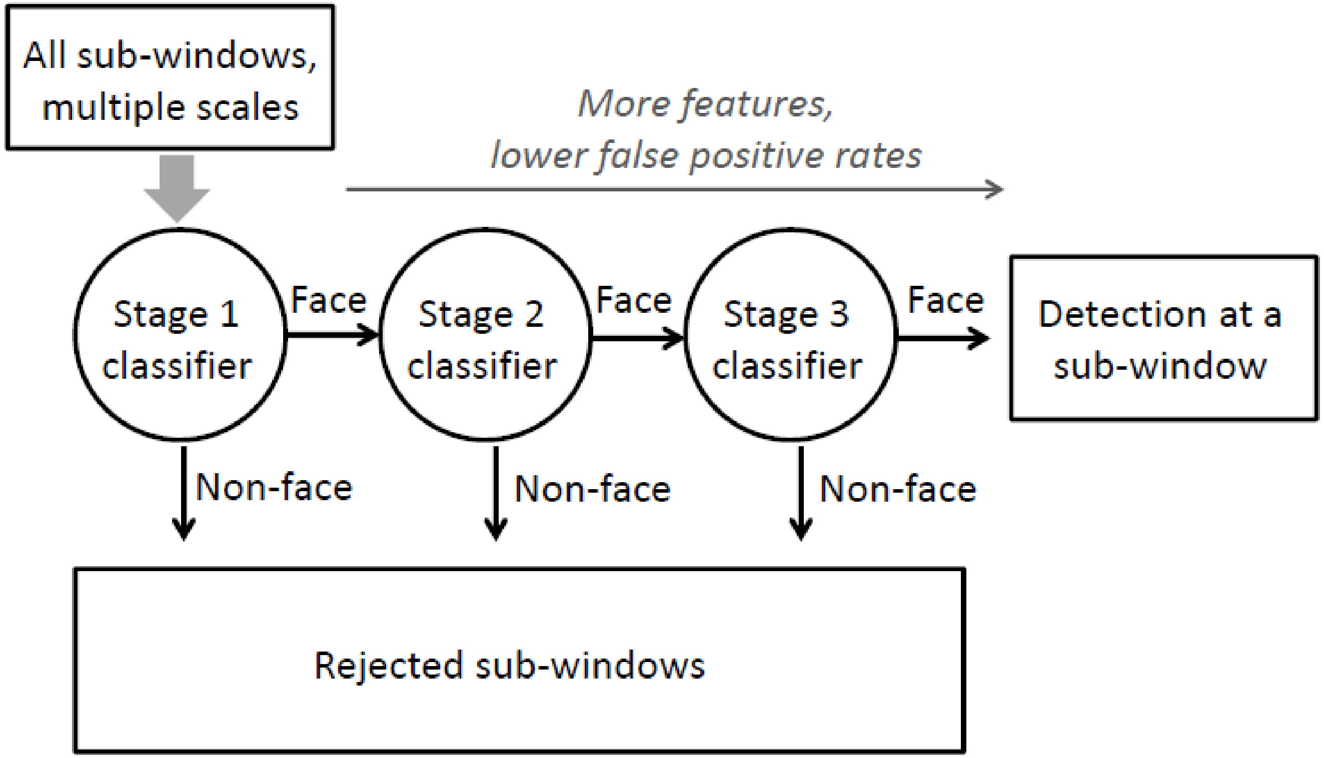
- AdaBoost searches for a small number of good classifiers – features (feature selection)
- adaptively constructs a final strong classifier taking into account the failures of each one of the chosen weak classifiers (weight appliance)
- AdaBoost is used to **both** select a small set of features and train a strong classifier

Now we have a good face detector

- We can build a 200-feature classifier!
- Experiments in original paper showed that a 200-feature classifier achieves:
  - 95% detection rate
  - 0.14×10^{-3} FP rate (1 in 14084)
  - Scanned all sub-windows of a 384x288 pixel image in 0.7 seconds (on Intel PIII 700MHz)
- The more the better (?)
  - Gain in classifier performance
  - Lose in CPU time
- Verdict: good & fast, but not enough
  - Competitors achieve close to **1 in a 1.000.000 FP rate!**
  - 0.7 sec / frame **IS NOT** real-time.



Cascading classifiers for detection



- Form a *cascade* with low false negative rates early on
- Apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative

Training a cascade of classifiers

Keep in mind:

- Competitors achieved 95% TP rate, 10^{-6} FP rate
- These are the goals. Final cascade must do better!

Given the goals, to design a cascade we must choose:

- Number of layers in cascade (strong classifiers)
- Number of features of each strong classifier (the 'T' in definition)
- Threshold of each strong classifier (the $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$ in definition)

Optimiza… pr…

- Con…

TREMENDOUSLY
DIFFICULT
PROBLEM

Strong classifier definition:

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

where $\alpha_t = \log(\frac{1}{\beta_t})$, $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$

A simple framework for cascade training

Do not despair. Viola & Jones suggested a heuristic algorithm for the cascade training:

- does not guarantee optimality
- but produces a "effective" cascade that meets previous goals

Manual Tweaking:

- overall training outcome is highly depended on user's choices
- select f_i (Maximum Acceptable False Positive rate / layer)
- select d_i (Minimum Acceptable True Positive rate / layer)
- select F_{target} (Target Overall FP rate)
- possible repeat trial & error process for a given training set

Until F_{target} is met:

- Add new layer:
  - Until f_i , d_i rates are met for this layer
    - **Increase** feature number & train new strong classifier with **AdaBoost**
    - Determine rates of layer on validation set

A simple framework for cascade training

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

While $F_i > F_{target}$

  i++

  $n_i = 0$; $F_i = F_{i-1}$

  while $F_i > f \times F_{i-1}$

    ○ n_i ++

    ○ Use P and N to train a classifier with n_i features using AdaBoost

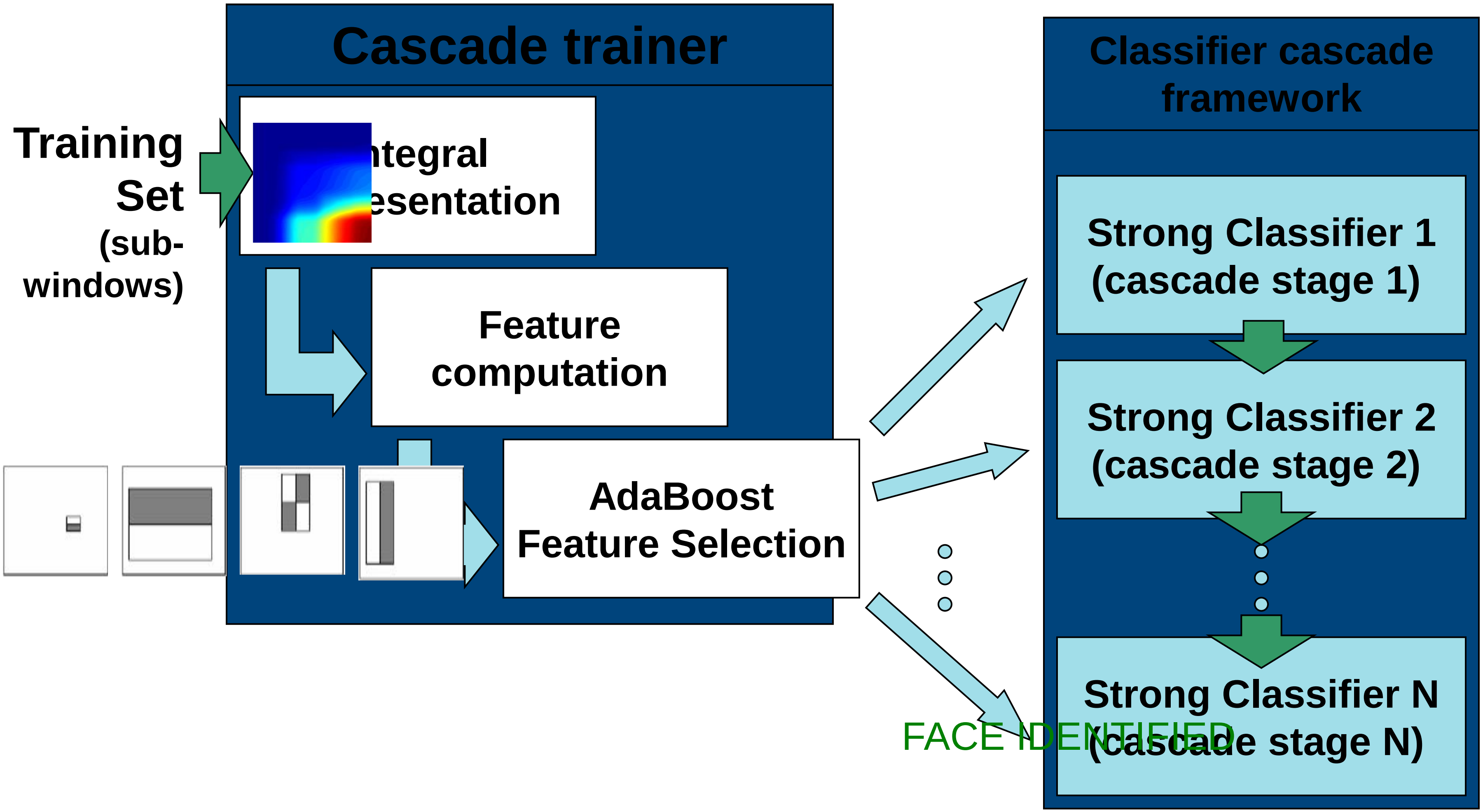
    ○ Evaluate current cascaded classifier on validation set to determine F_i and D_i

    ○ Decrease threshold for the ith classifier until the current cascaded classifier has a detection rate of at least $d \times D_{i-1}$ (this also affects F_i)

  $N = \varnothing$

  If $F_i > F_{target}$ then evaluate the current cascaded detector on the set of non-face images and put any false detections into the set N.

Training phase
Testing phase



pros …

Extremely fast feature computation

Efficient feature selection

Scale and location invariant detector

- Instead of scaling the image itself (e.g. pyramid-filters), we scale the features.

Such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, hands)

… and cons

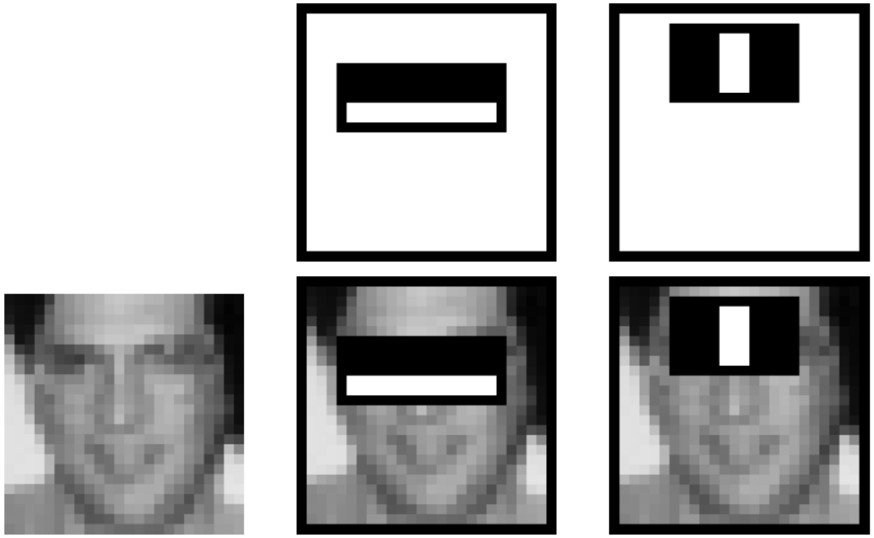
Detector is most effective only on frontal images of faces

- can hardly cope with 45º face rotation

Sensitive to lighting conditions

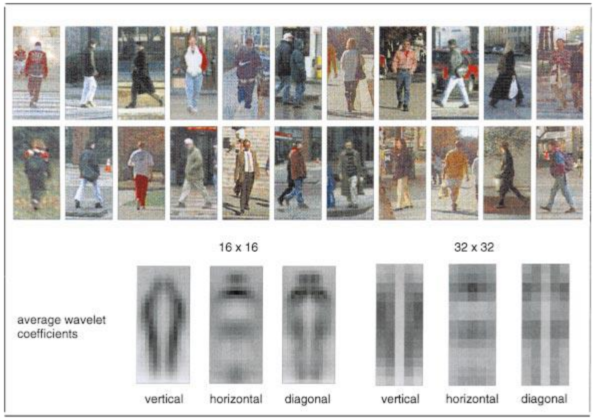
We might get multiple detections of the same face, due to overlapping sub-windows.

Viola-Jones Face Detector Live demo



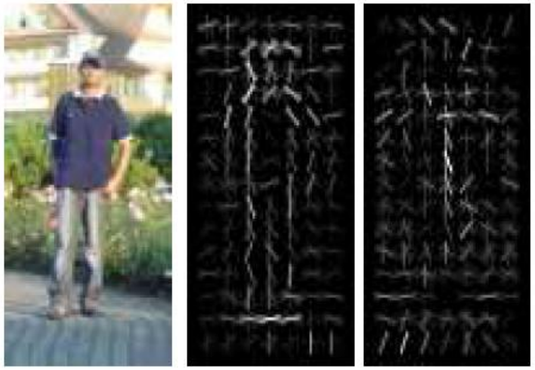
Pedestrian detection

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



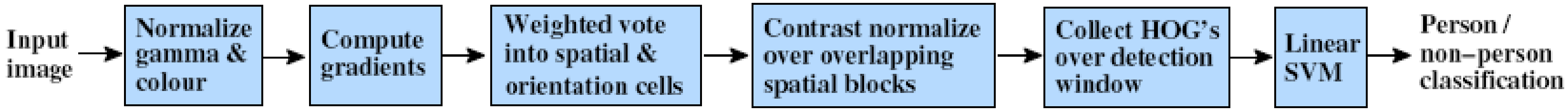
SVM with Haar wavelets [Papageorgiou & Poggio, IJCV 2000]

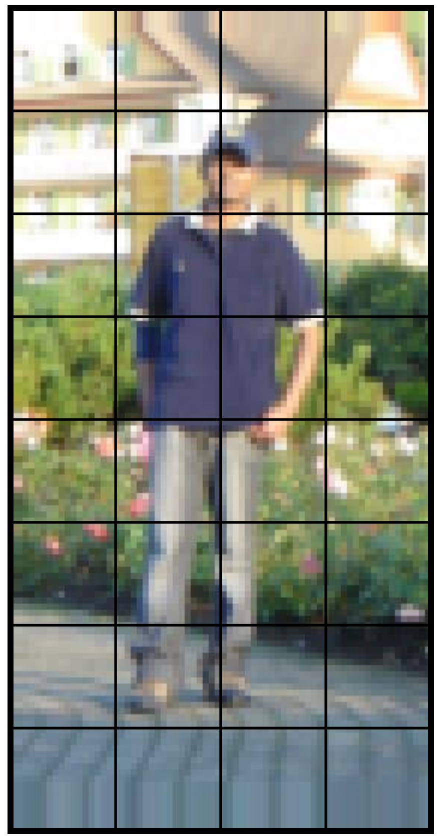
Space-time rectangle features [Viola, Jones & Snow, ICCV 2003]



SVM with HoGs [Dalal & Triggs, CVPR 2005]

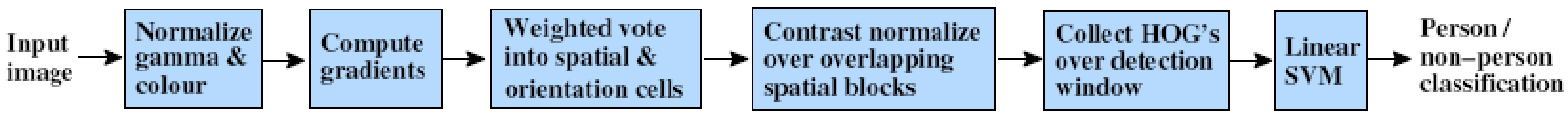
A short detour: HOG features for pedestrian detection





Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

A short detour: HOG features for pedestrian detection



-1	0	1

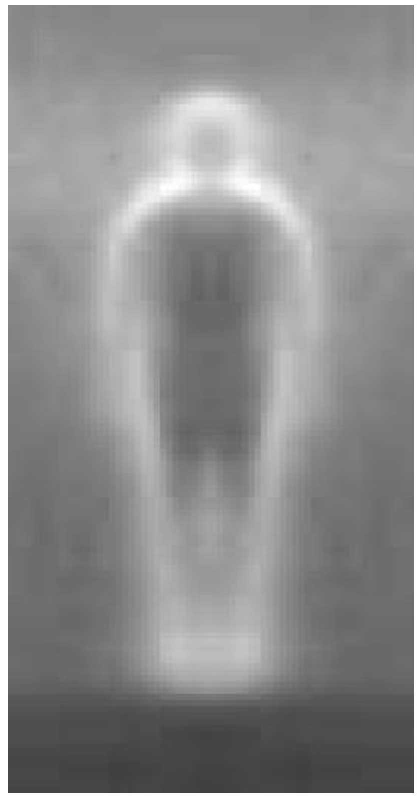
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



0	1
-1	0

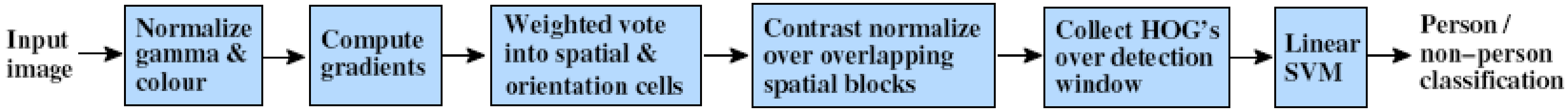
diagonal

-1	0	1
-2	0	2
-1	0	1

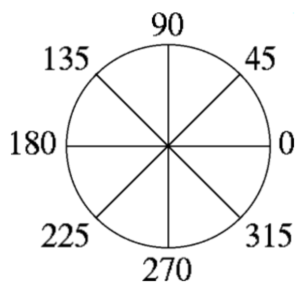
Sobel

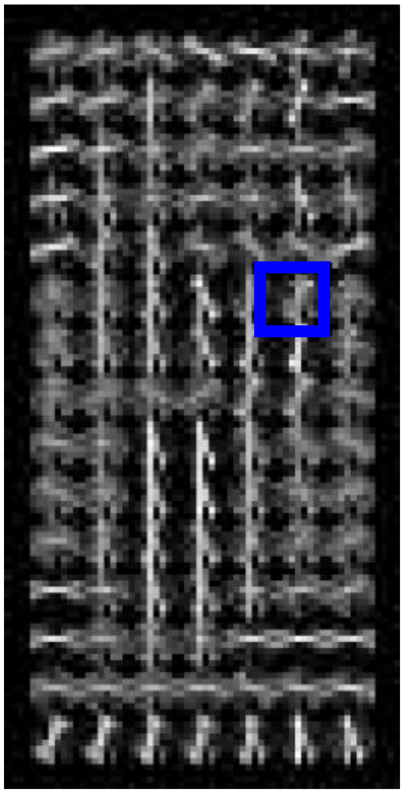
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

A short detour: HOG features for pedestrian detection



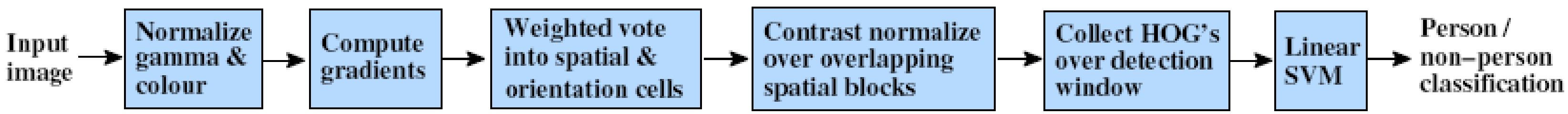
- Histogram of gradient orientations
  - -Orientation





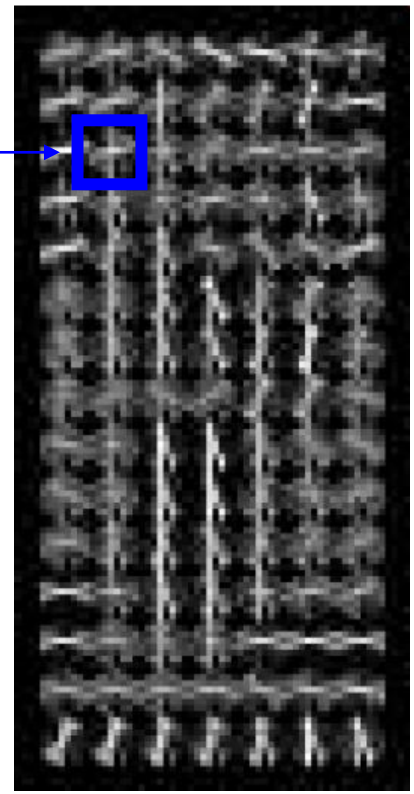
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

A short detour: HOG features for pedestrian detection



8 orientations

X=

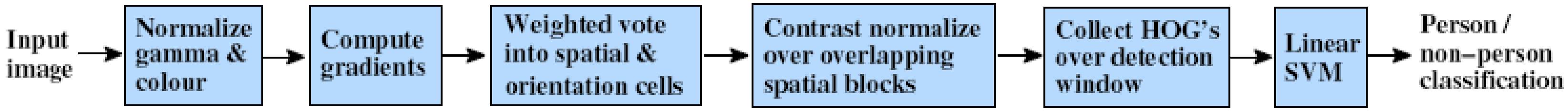


$\in R^{840}$

15x7 cells

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

A short detour: HOG features for pedestrian detection



$$0.16 = w^T x - b$$

$$sign(0.16) = 1$$

$=>$ pedestrian

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

HOG Detector

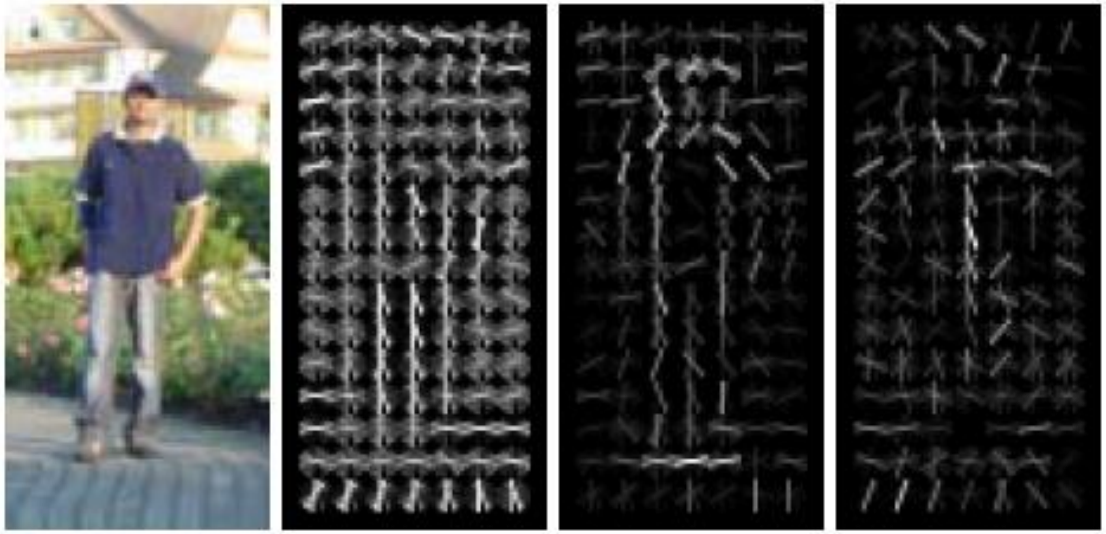


Image | HOG descriptor | HOG descriptor weighted by pos. SVM weights | neg. SVM weights

Window-based detection: strengths

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

Kristen Grauman

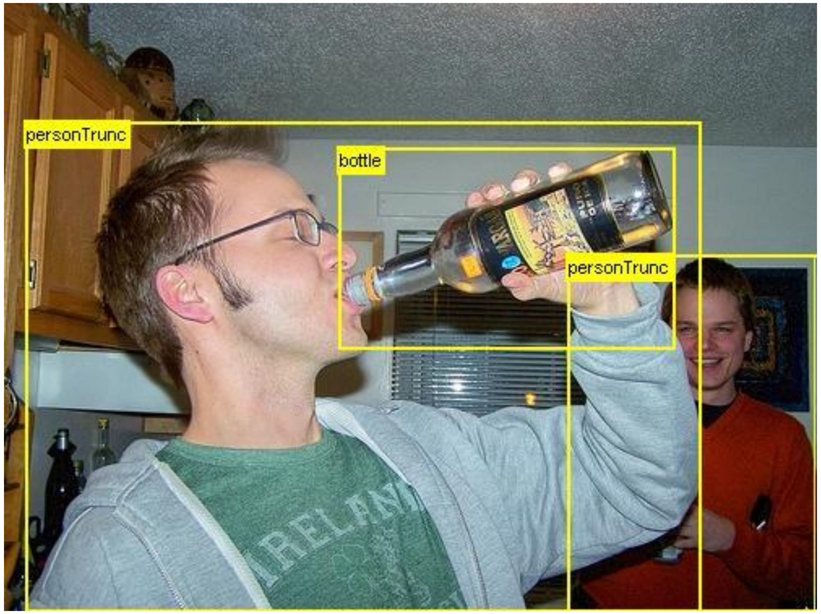
Window-based detection: Limitations

- 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

Kristen Grauman

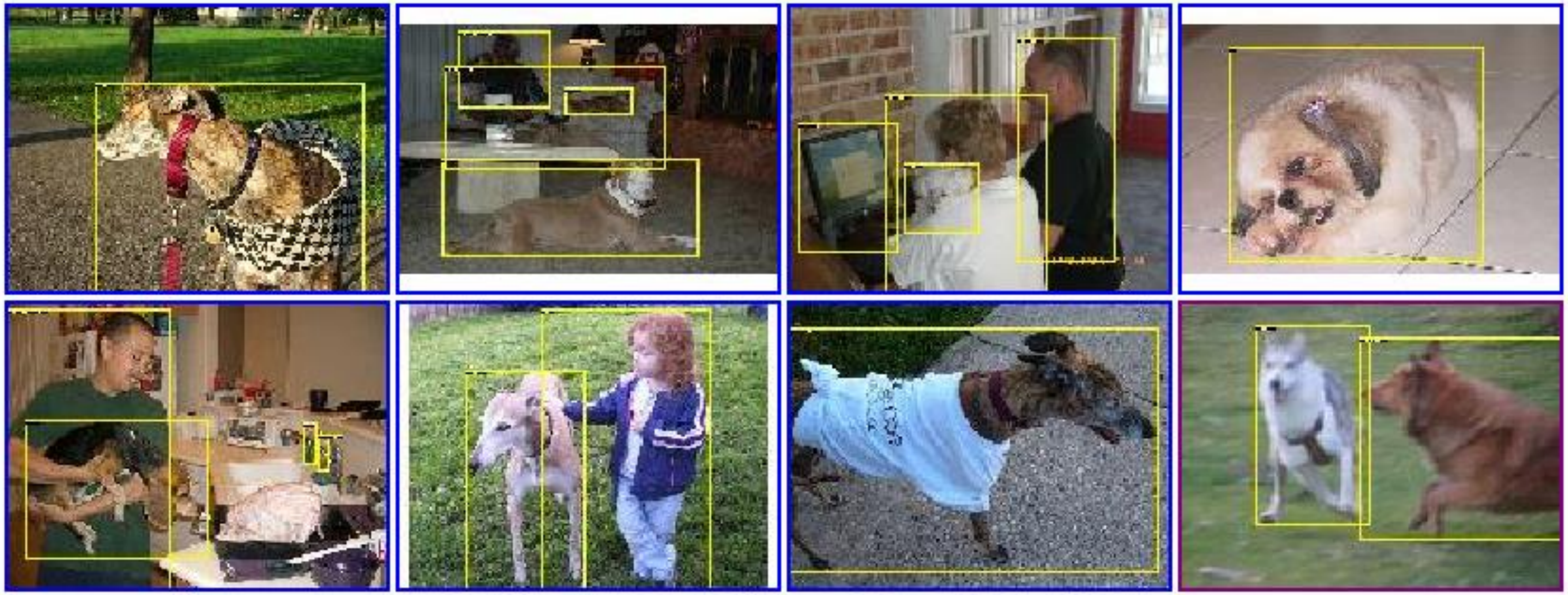
Limitations (continued)

- Not all objects are “box” shaped



Limitations (continued)

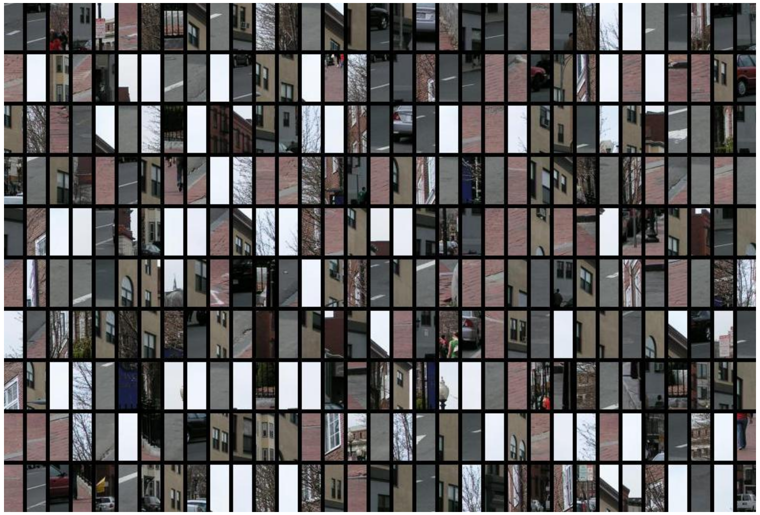
- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions



Limitations (continued)

- If considering windows in isolation, context is lost

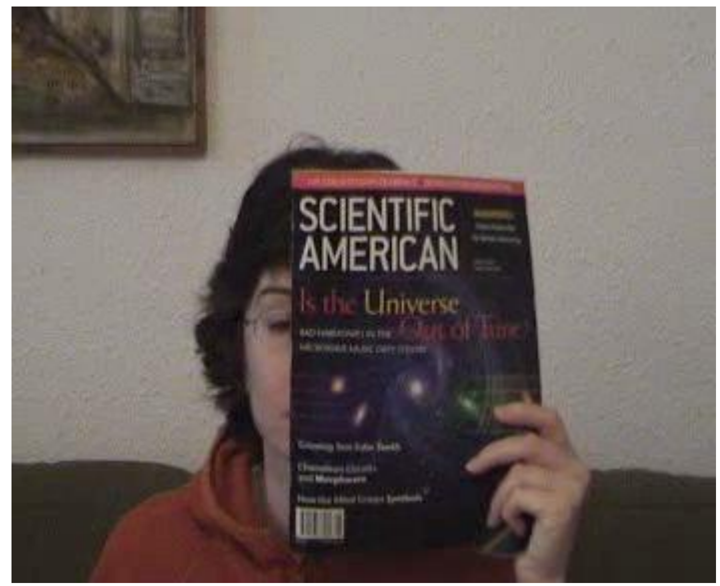
Sliding window



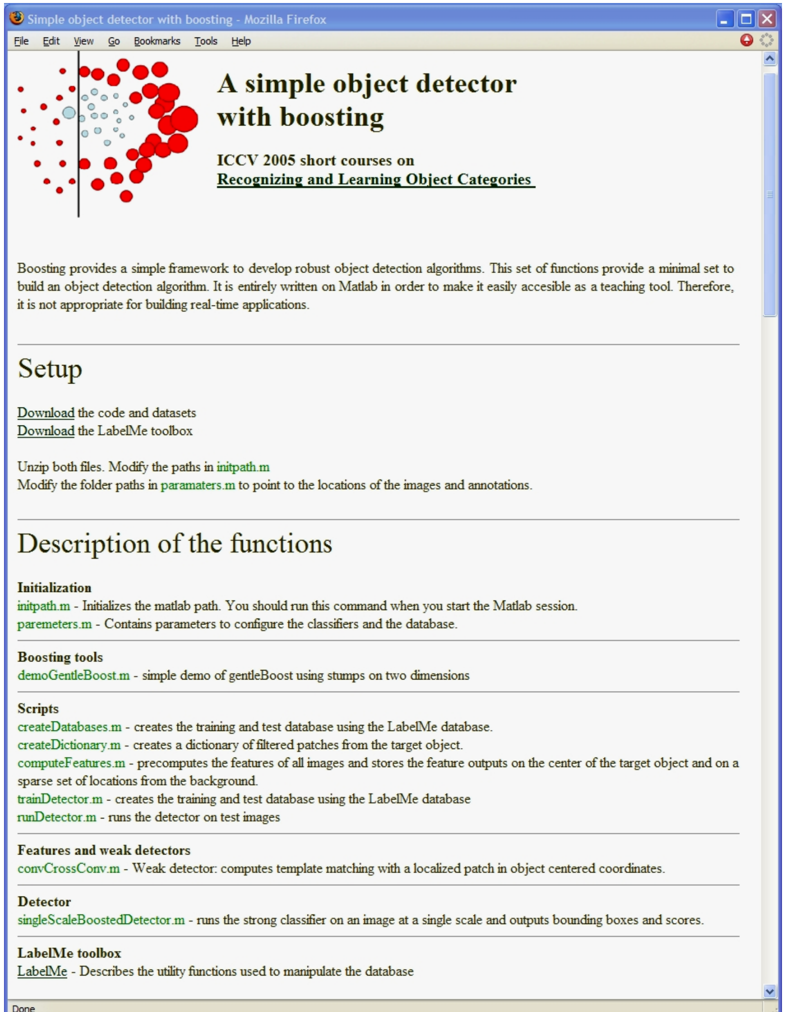
Detector's view

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



A simple object detector with Boosting



A simple object detector with boosting

ICCV 2005 short courses on
Recognizing and Learning Object Categories

Boosting provides a simple framework to develop robust object detection algorithms. This set of functions provide a minimal set to build an object detection algorithm. It is entirely written on Matlab in order to make it easily accesible as a teaching tool. Therefore, it is not appropriate for building real-time applications.

Setup

Download the code and datasets
Download the LabelMe toolbox

Unzip both files. Modify the paths in initpath.m
Modify the folder paths in paramaters.m to point to the locations of the images and annotations.

Description of the functions

Initialization
initpath.m - Initializes the matlab path. You should run this command when you start the Matlab session.
paremeters.m - Contains parameters to configure the classifiers and the database.

Boosting tools
demoGentleBoost.m - simple demo of gentleBoost using stumps on two dimensions

Scripts
createDatabases.m - creates the training and test database using the LabelMe database.
createDictionary.m - creates a dictionary of filtered patches from the target object.
computeFeatures.m - precomputes the features of all images and stores the feature outputs on the center of the target object and on a sparse set of locations from the background.
trainDetector.m - creates the training and test database using the LabelMe database
runDetector.m - runs the detector on test images

Features and weak detectors
convCrossConv.m - Weak detector: computes template matching with a localized patch in object centered coordinates.

Detector
singleScaleBoostedDetector.m - runs the strong classifier on an image at a single scale and outputs bounding boxes and scores.

LabelMe toolbox
LabelMe - Describes the utility functions used to manipulate the database

http://people.csail.mit.edu/torralba/iccv2005/

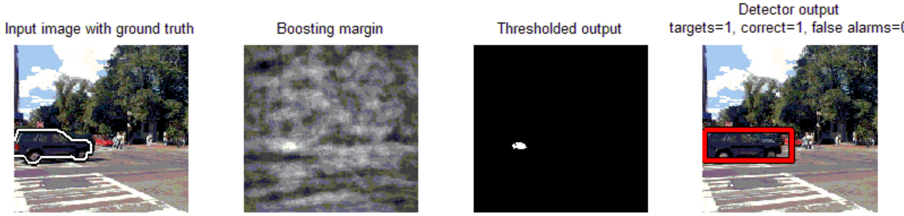
Download

- Toolbox for manipulating dataset
- Code and dataset

Matlab code

- Gentle boosting
- Object detector using a part based model

Dataset with cars and computer monitors



Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Strong classifier

Features vector

Weight

Weak classifier

Boosting

- Defines a classifier using an additive model:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \dots$$

Strong classifier

Features vector

Weight

Weak classifier

- We need to define a family of weak classifiers

$f_k(x)$ from a family of weak classifiers

Boosting

Boosting fits the additive model

$$F(x) = f_1(x) + f_2(x) + f_3(x) + \ldots$$

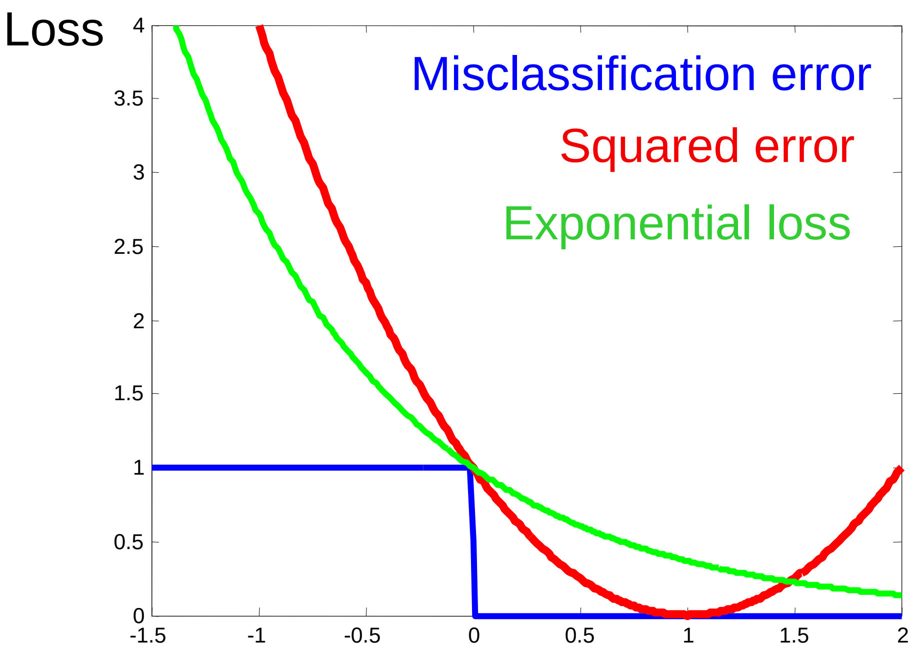
by minimizing the exponential loss

$$J(F) = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Training samples

The exponential loss is a differentiable upper bound to the misclassification error.

Exponential loss



yF(x) = margin

Squared error

$$J = \sum_{t=1}^{N} [y_t - F(x_t)]^2$$

Exponential loss

$$J = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Boosting

Sequential procedure. At each step we add

$$F(x) \leftarrow F(x) + f_m(x)$$

to minimize the residual loss

$$(\phi_m) = \arg\min_{\phi} \sum_{t=1}^{N} J\left(y_i, F(x_t) + f(x_t; \phi)\right)$$

Parameters weak classifier

Desired output

input

For more details: Friedman, Hastie, Tibshirani. "Additive Logistic Regression: a Statistical View of Boosting" (1998)

gentleBoosting

- At each iteration:

  We chose $f_m(x)$ that minimizes the cost:

$$J(F + f_m) = \sum_{t=1}^{N} e^{-y_t(F(x_t)+f_m(x_t))}$$

Instead of doing exact optimization, gentle Boosting minimizes a Taylor approximation of the error:

$$J(F) \propto \sum_{t=1}^{N} \boxed{e^{-y_t F(x_t)}} (y_t - f_m(x_t))^2$$

Weights at this iteration

At each iterations we just need to solve a weighted least squares problem

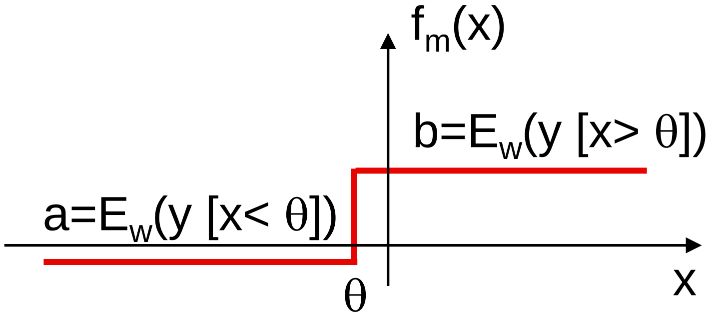
For more details: Friedman, Hastie, Tibshirani. “Additive Logistic Regression: a Statistical View of Boosting” (1998)

Weak classifiers

- The input is a set of weighted training samples (x,y,w)
- Regression stumps: simple but commonly used in object detection.

$$f_m(x) = a[x_k < \theta] + b[x_k \geq \theta]$$

Four parameters: $[a, b, \theta, k]$



fitRegressionStump.m

gentleBoosting.m

```
function classifier = gentleBoost(x, y, Nrounds)

…

for m = 1:Nrounds

    fm = selectBestWeakClassifier(x, y, w);

    w = w .* exp(- y .* fm);

    % store parameters of fm in classifier
    …
end
```

Initialize weights w = 1

Solve weighted least-squares

Re-weight training samples

From images to features: Weak detectors

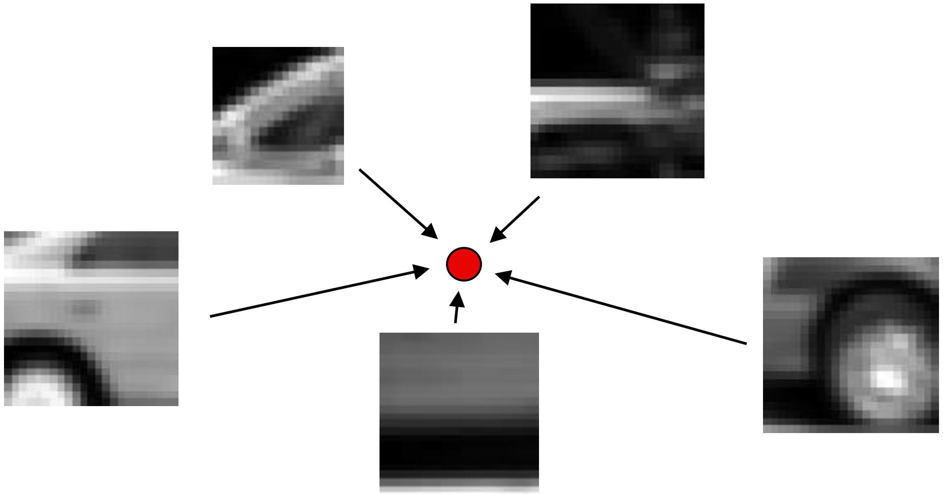
We will now define a family of visual features that can be used as weak classifiers ("weak detectors")

$$\longrightarrow h_i(I, x, y) \longrightarrow$$

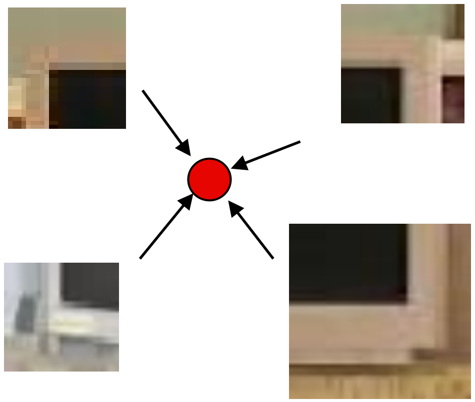
Takes image as input and the output is binary response.
The output is a weak detector.

Weak detectors

Part based: similar to part-based generative models. We create weak detectors by using parts and voting for the object center location



Car model



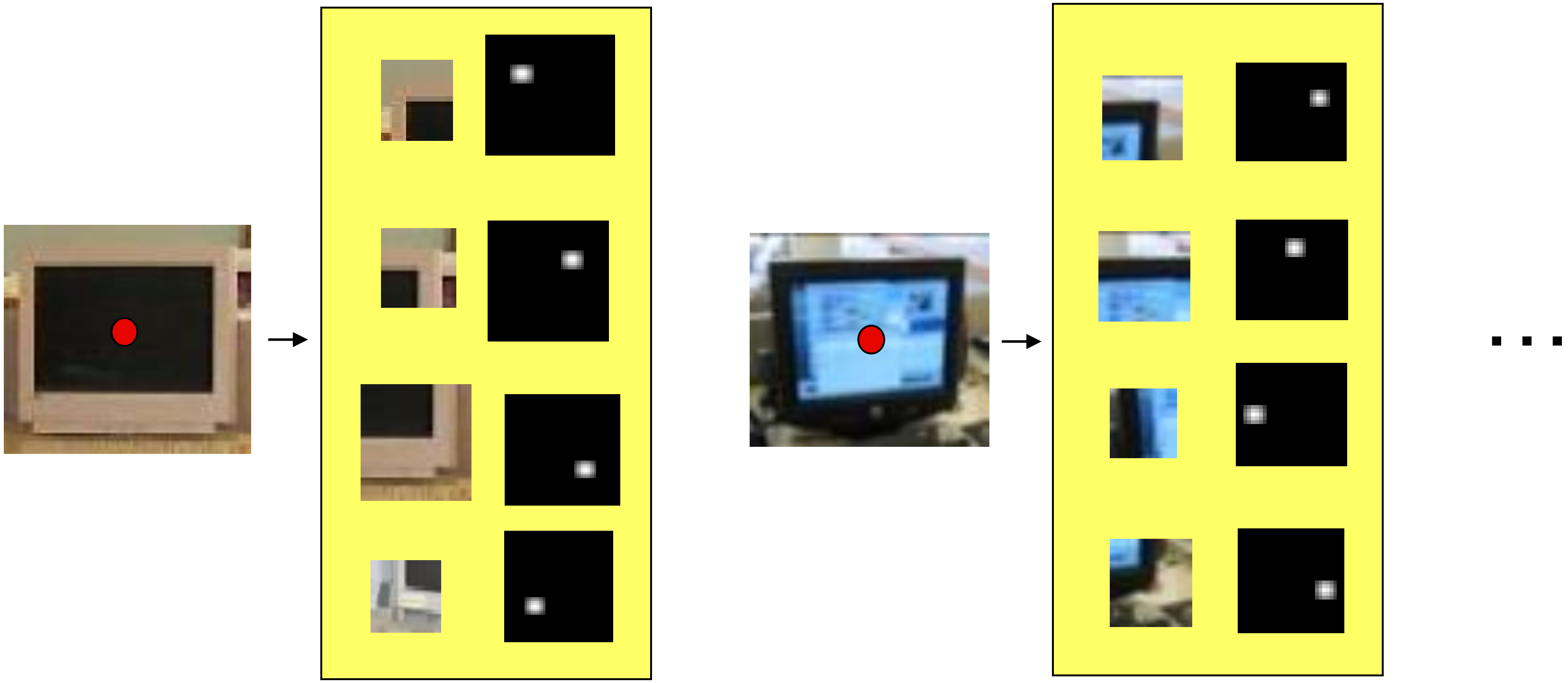
Screen model

These features are used for the detector in the gentleboost tutorial.

Weak detectors

First we collect a set of part templates from a set of training objects.

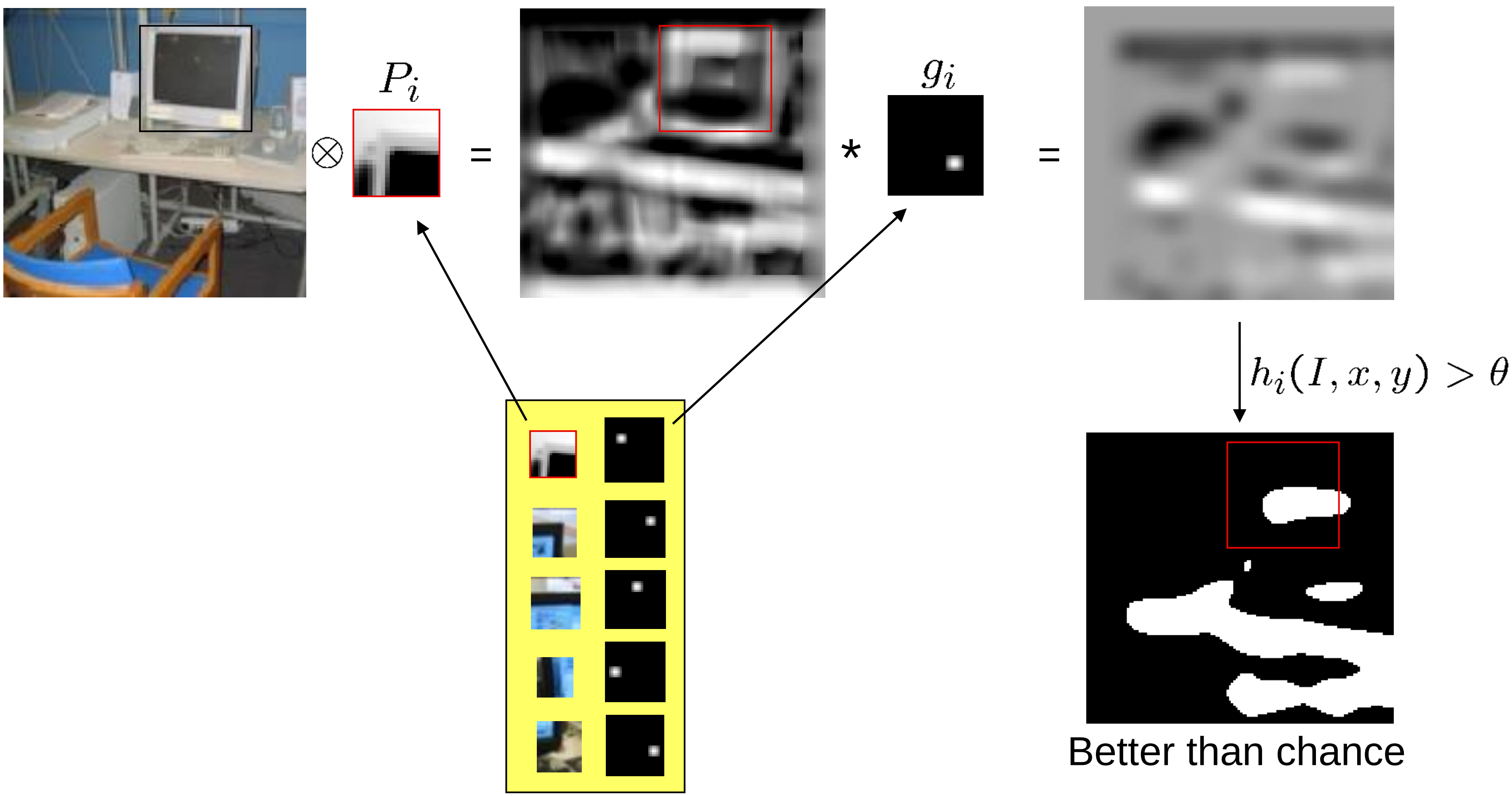
Vidal-Naquet, Ullman (2003)



Weak detectors

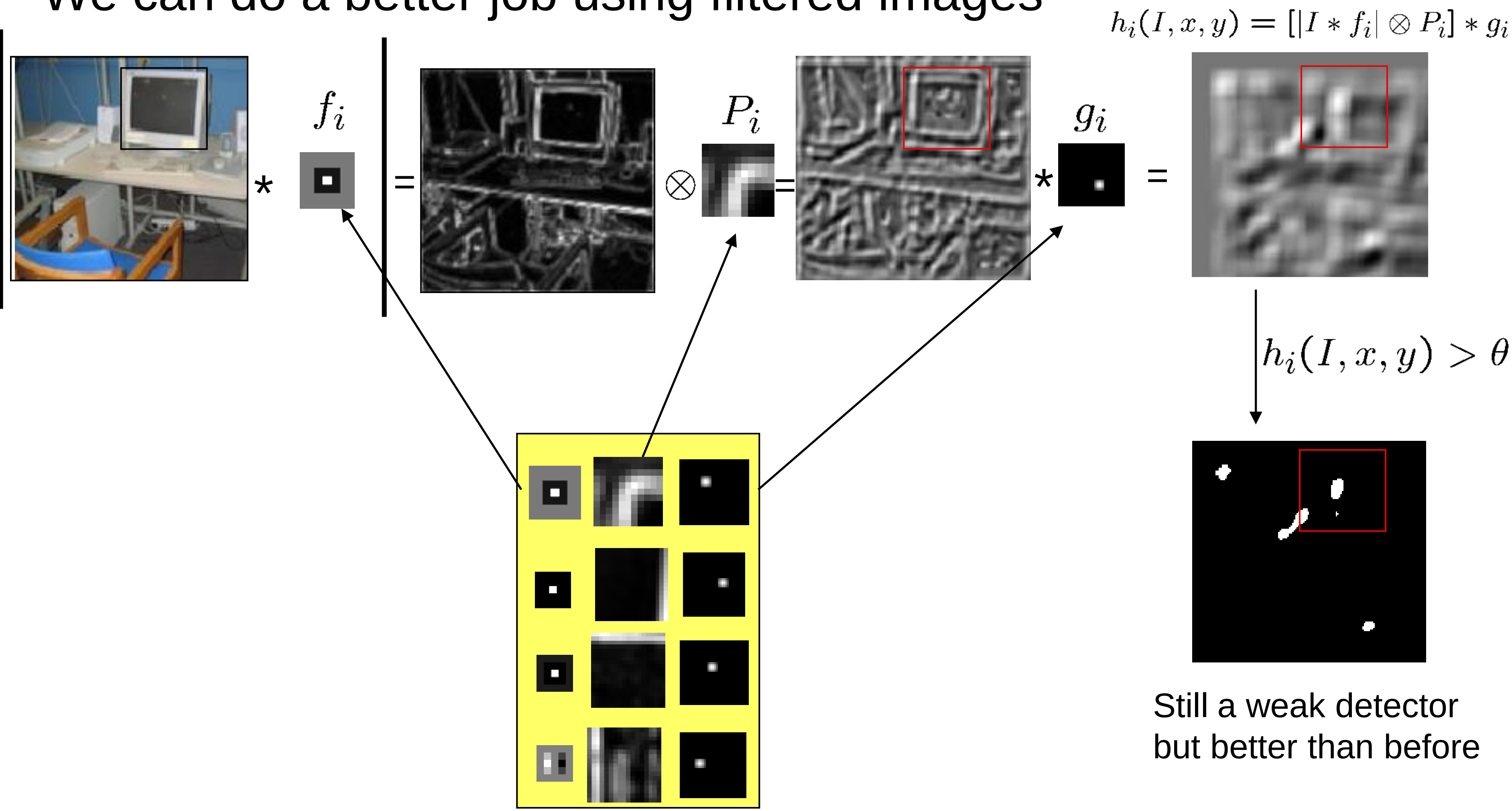
We now define a family of “weak detectors” as:

$$h_i(I, x, y) = [I \otimes P_i] * g_i$$



Weak detectors

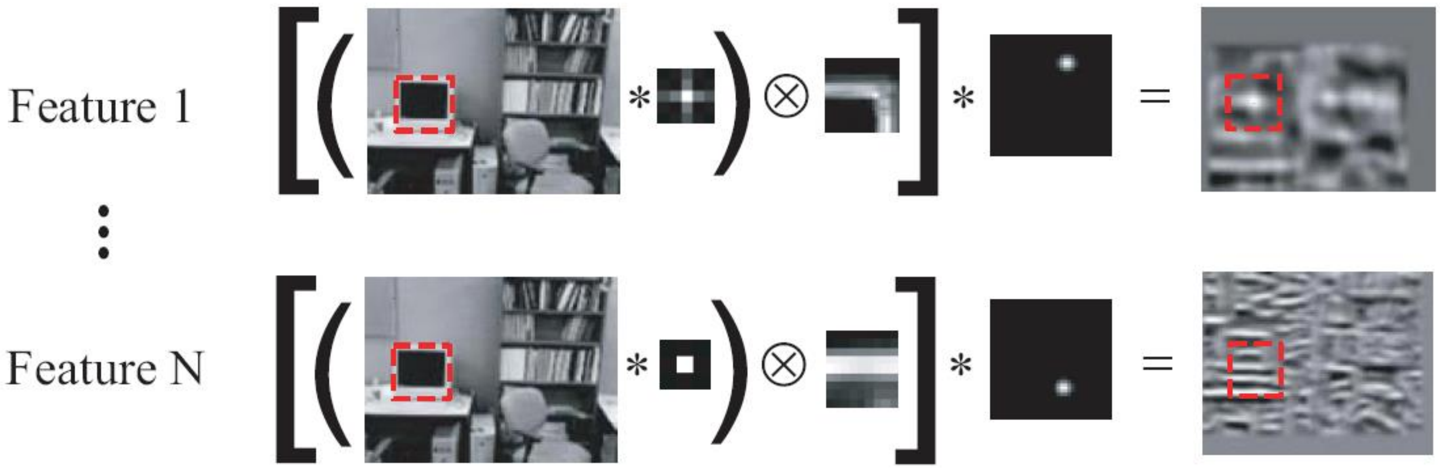
We can do a better job using filtered images



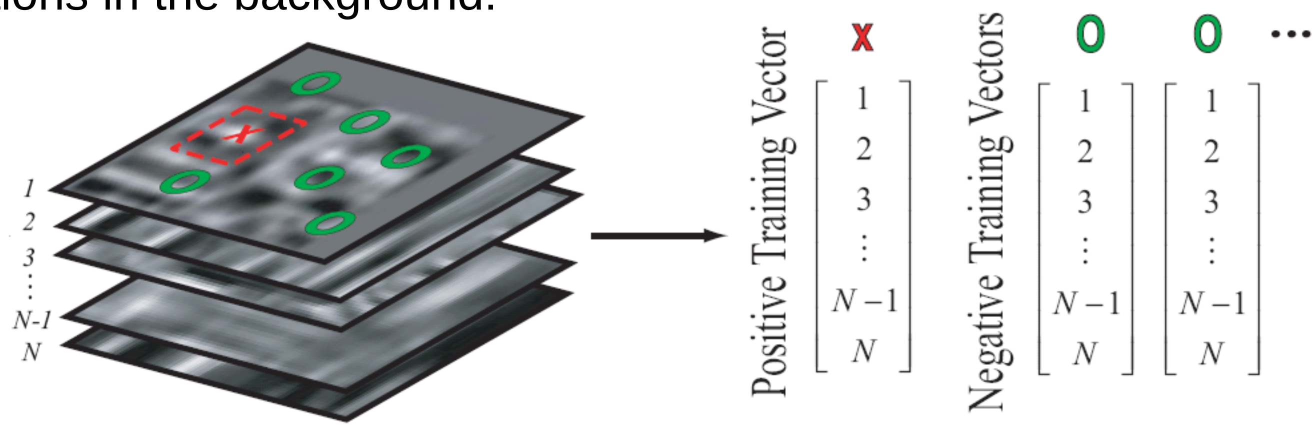
Still a weak detector
but better than before

Training

First we evaluate all the N features on all the training images.

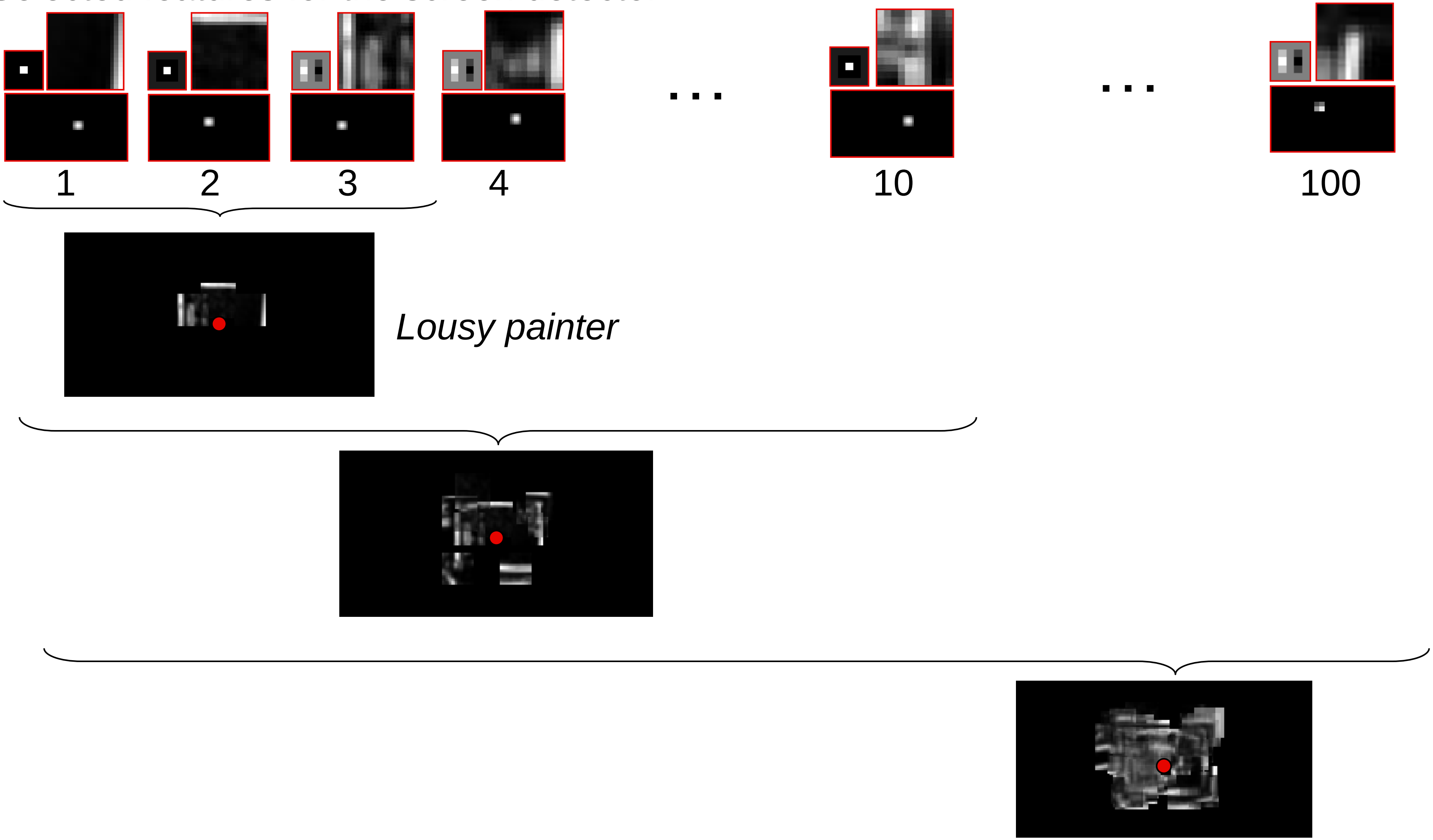


Then, we sample the feature outputs on the object center and at random locations in the background:



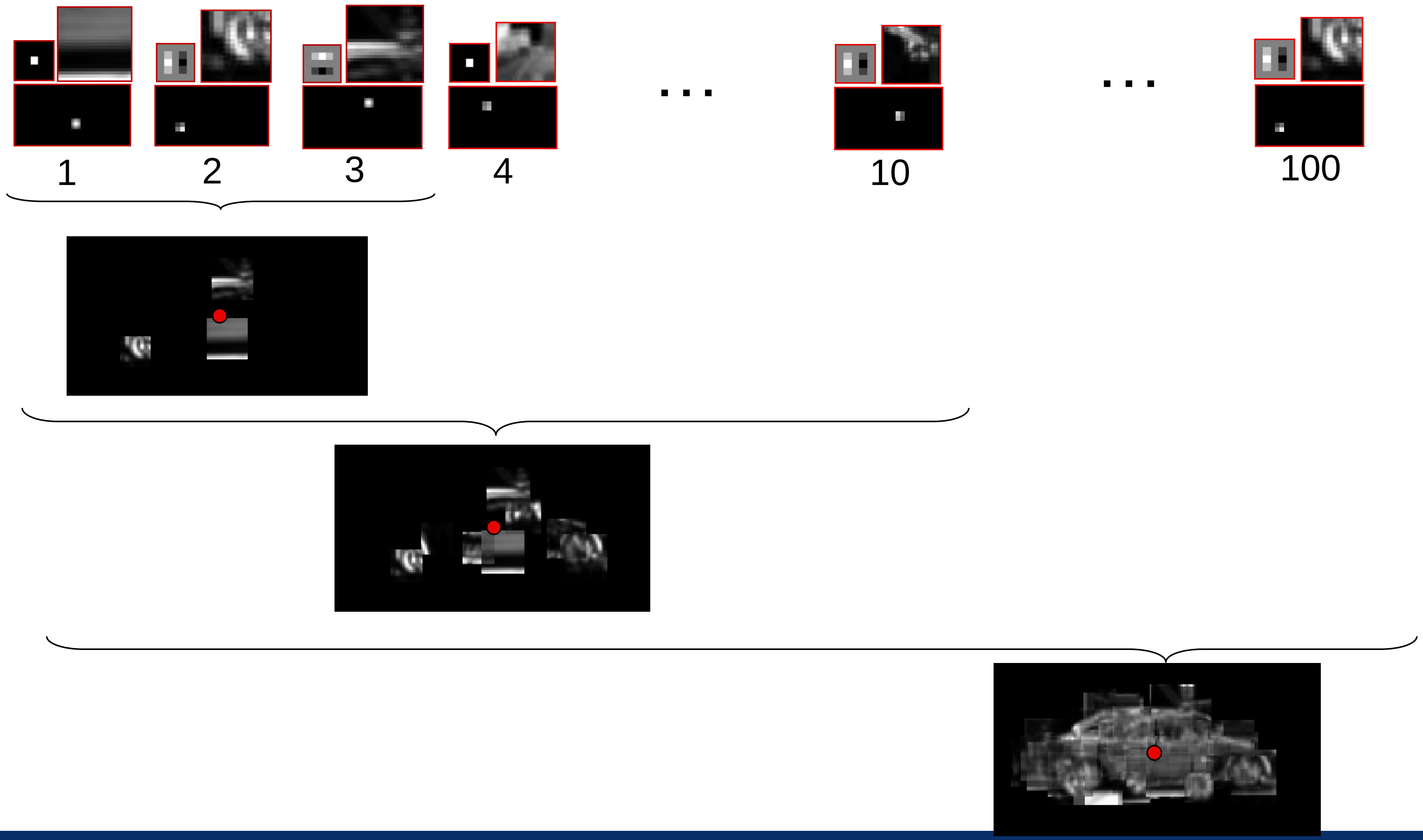
Representation and object model

Selected features for the screen detector

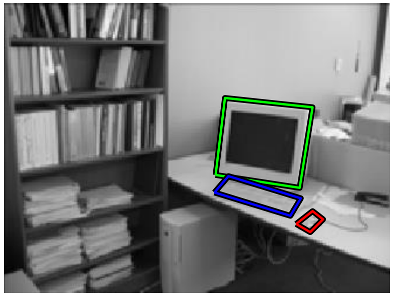


Representation and object model

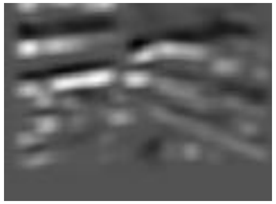
Selected features for the car detector



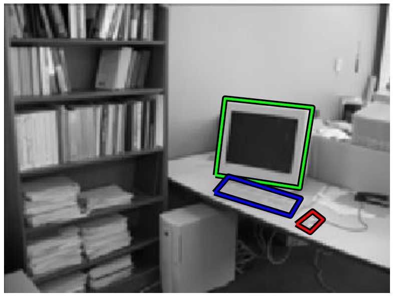
Example: screen detection



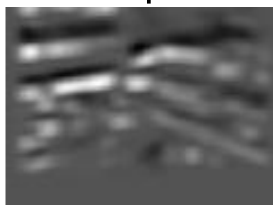
Feature
output



Example: screen detection



Feature
output

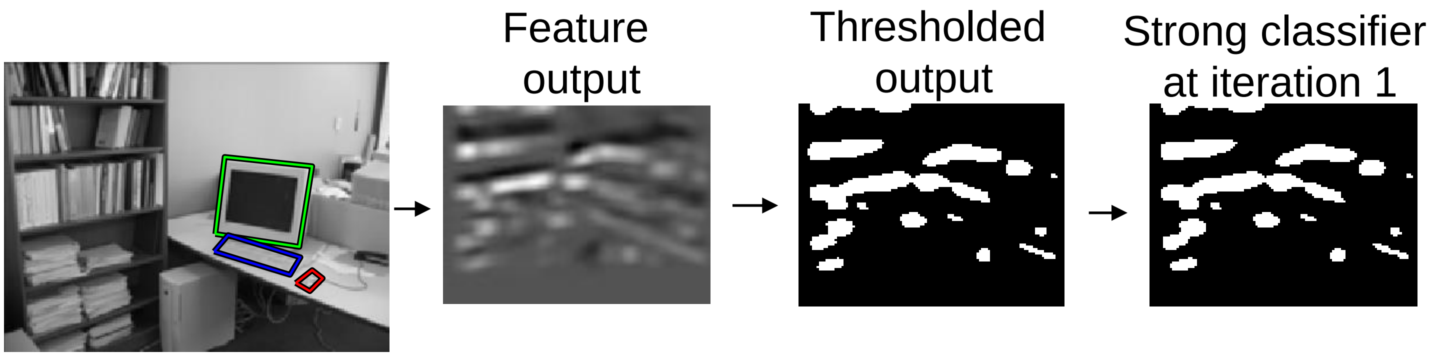


Thresholded
output

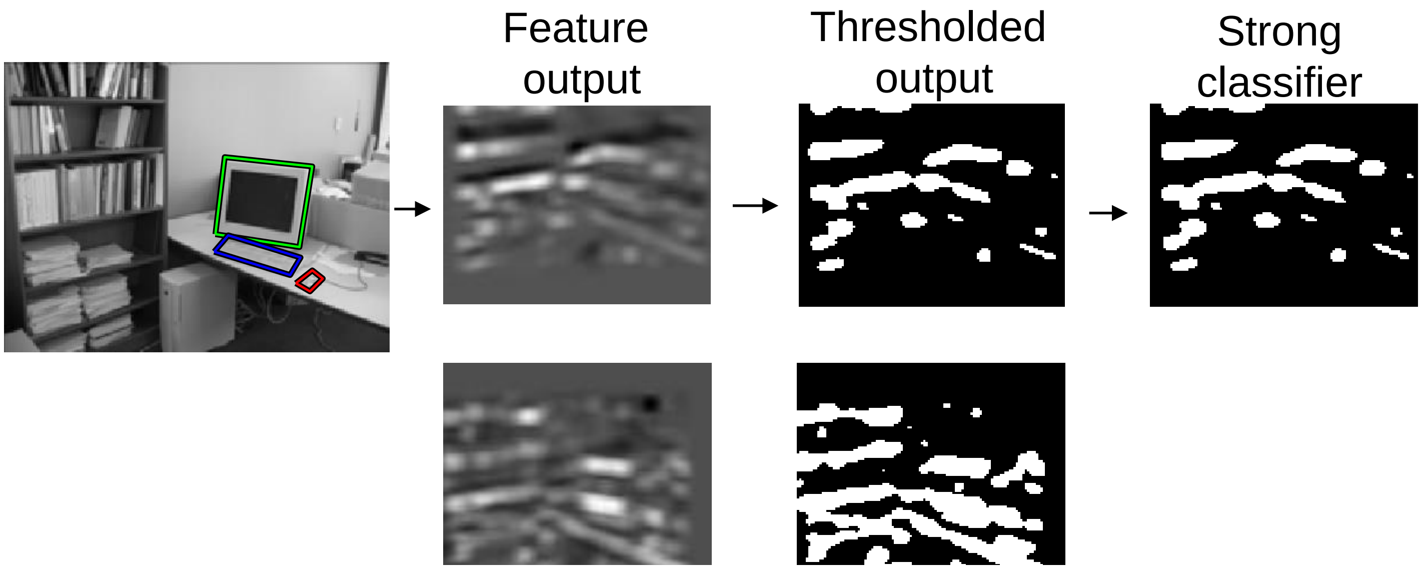
Weak ‘detector’

Produces many false alarms.

Example: screen detection



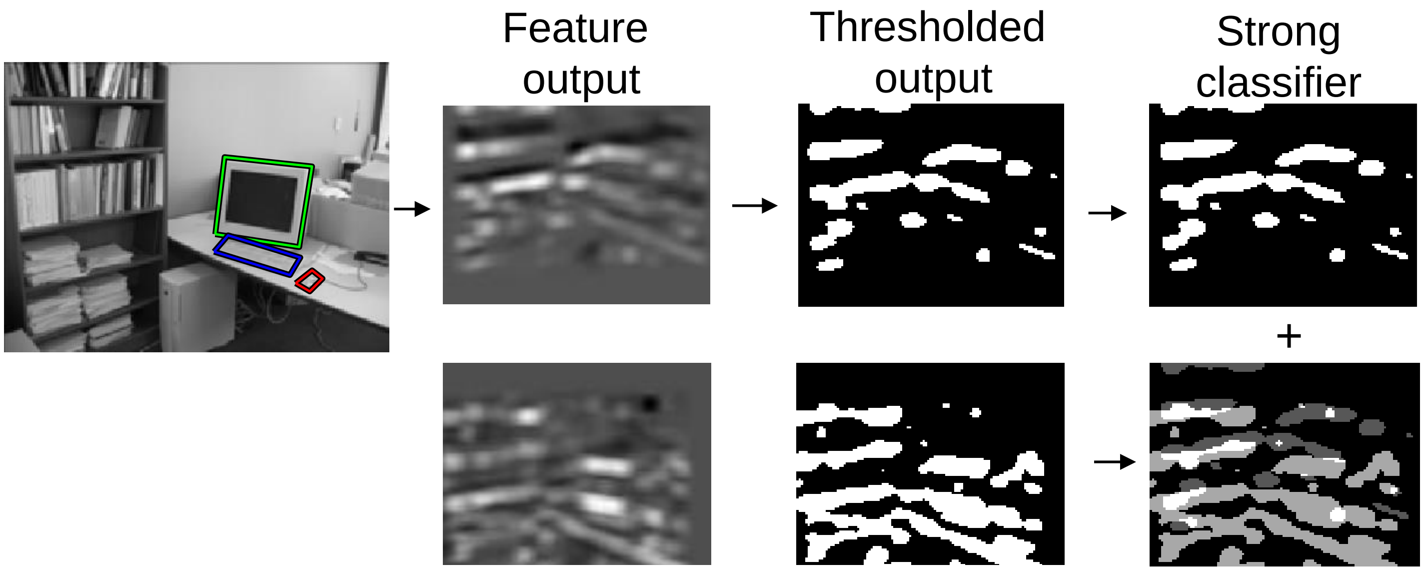
Example: screen detection



Second weak 'detector'

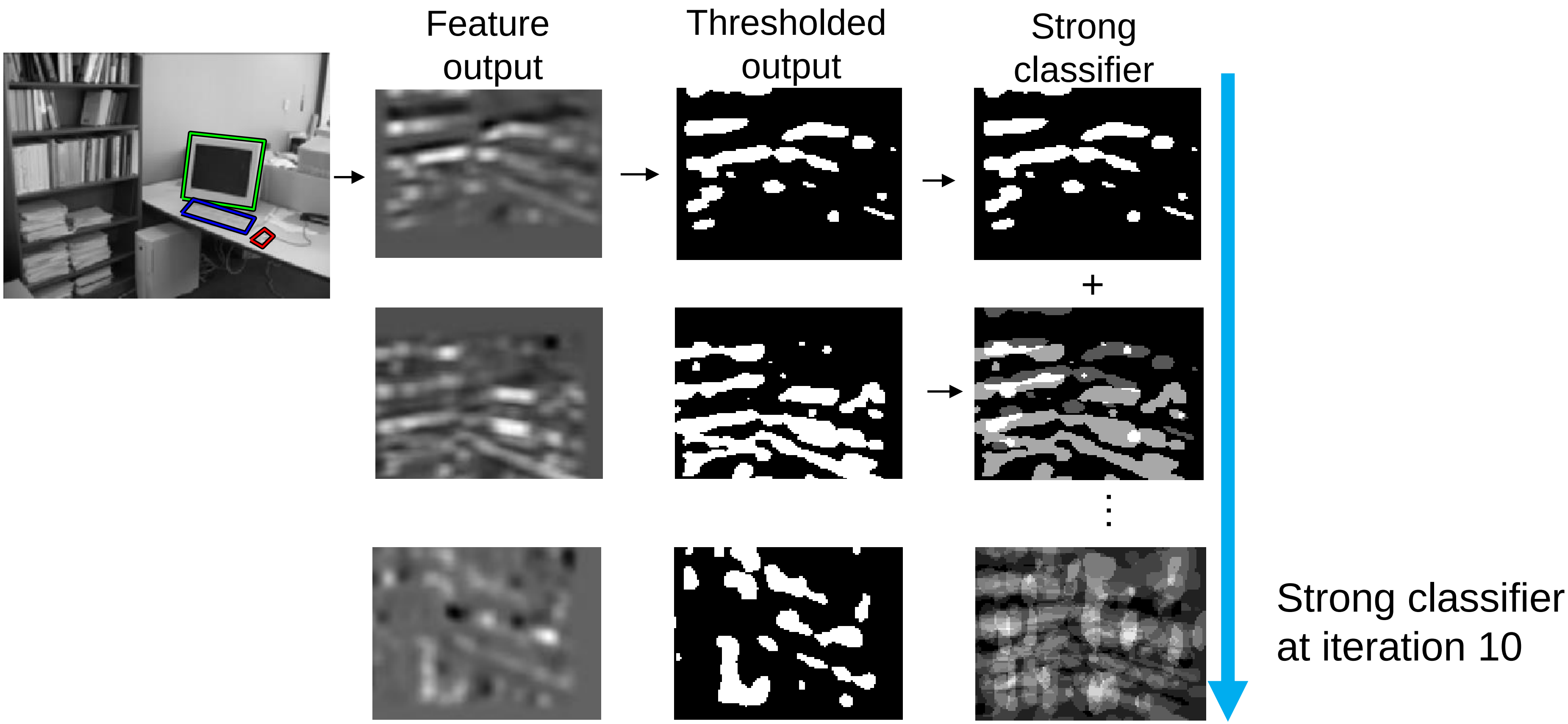
Produces a different set of false alarms.

Example: screen detection

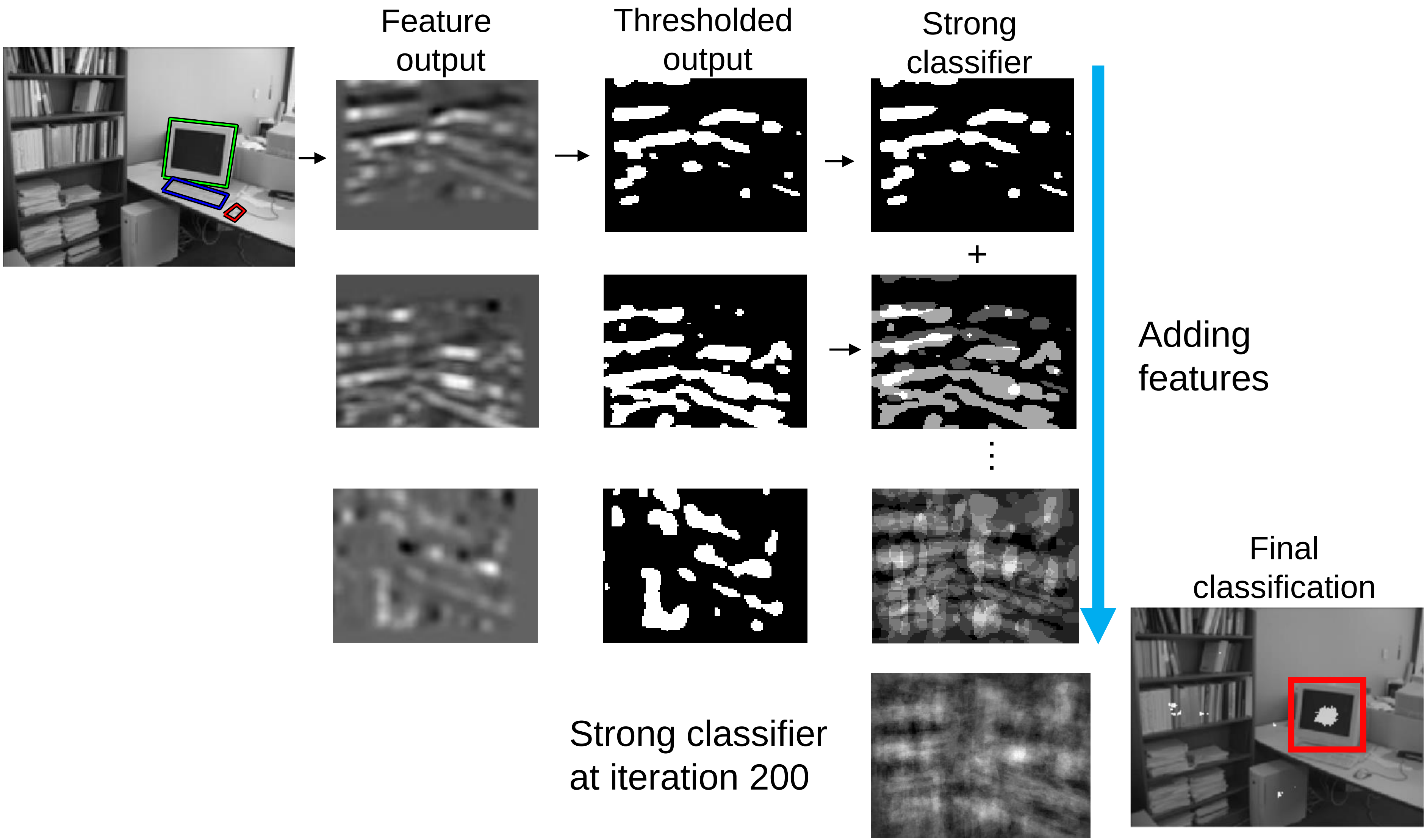


Strong classifier at iteration 2

Example: screen detection



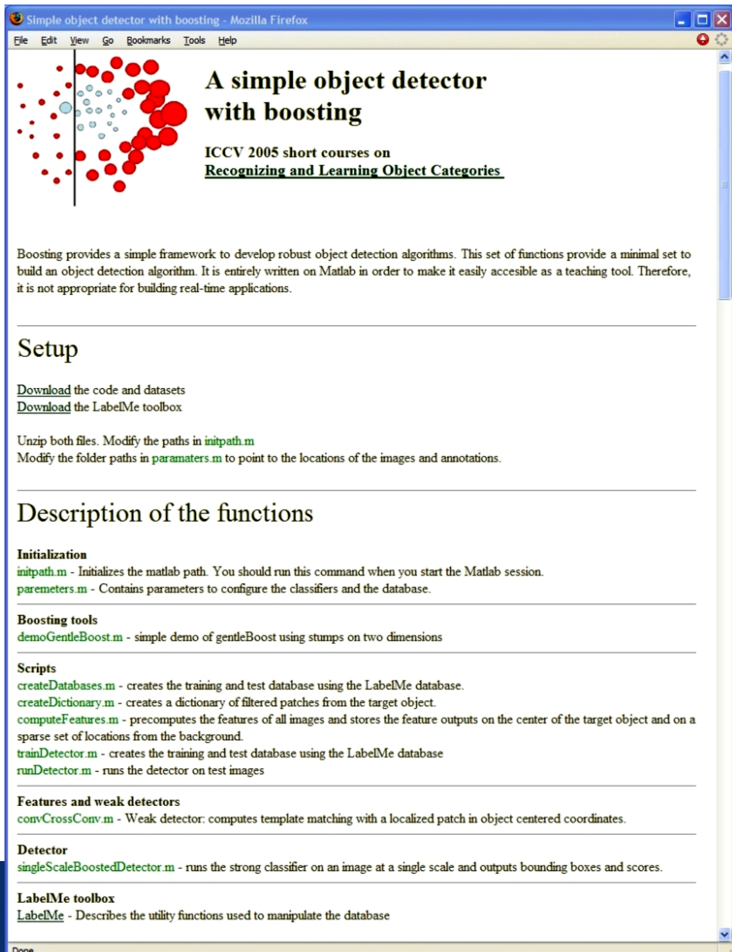
Example: screen detection



Demo of gentleBoost on the LabelMe dataset

Demo of screen and car detectors using parts, Gentle boost, and stumps:

> runDetector.m



Simple object detector with boosting - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

A simple object detector with boosting

ICCV 2005 short courses on
Recognizing and Learning Object Categories

Boosting provides a simple framework to develop robust object detection algorithms. This set of functions provide a minimal set to build an object detection algorithm. It is entirely written on Matlab in order to make it easily accesible as a teaching tool. Therefore, it is not appropriate for building real-time applications.

Setup

Download the code and datasets
Download the LabelMe toolbox

Unzip both files. Modify the paths in initpath.m
Modify the folder paths in paramaters.m to point to the locations of the images and annotations.

Description of the functions

Initialization
initpath.m - Initializes the matlab path. You should run this command when you start the Matlab session.
paremeters.m - Contains parameters to configure the classifiers and the database.

Boosting tools
demoGentleBoost.m - simple demo of gentleBoost using stumps on two dimensions

Scripts
createDatabases.m - creates the training and test database using the LabelMe database.
createDictionary.m - creates a dictionary of filtered patches from the target object.
computeFeatures.m - precomputes the features of all images and stores the feature outputs on the center of the target object and on a sparse set of locations from the background.
trainDetector.m - creates the training and test database using the LabelMe database
runDetector.m - runs the detector on test images

Features and weak detectors
convCrossConv.m - Weak detector: computes template matching with a localized patch in object centered coordinates.

Detector
singleScaleBoostedDetector.m - runs the strong classifier on an image at a single scale and outputs bounding boxes and scores.

LabelMe toolbox
LabelMe - Describes the utility functions used to manipulate the database

Done

Probabilistic interpretation

- Generative model

$$p(features,\ object\ class)$$

- Discriminative (Boosting) model.

Boosting is fitting an additive logistic regression model:

$$p(object\ class \mid features) = \frac{1}{1 + e^{-\sum h_i(I,x,y)}}$$

It can be a set of arbitrary functions of the image

This provides a great flexibility, difficult to beat by current generative models. But also there is the danger of not understanding what are they really doing.

Weak detectors

- Generative model

$$p(features,\ object\ class)$$

- Discriminative (Boosting) model.

Boosting is fitting an additive logistic regression model:

$$p(object\ class \mid features) = \frac{1}{1 + e^{-\sum h_i(I,x,y)}}$$

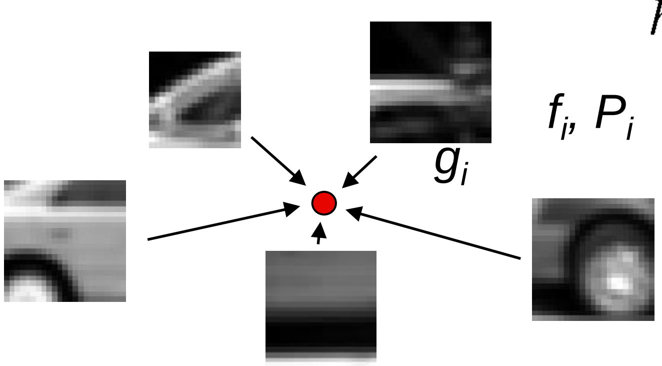
$$h_i(I,x,y) = [(I * f_i) \otimes P_i] * g_i$$

Image

Feature

Part template

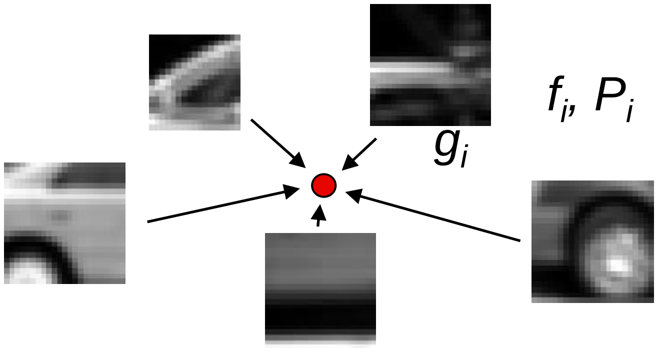
Relative position
wrt object center



Object models

- Invariance: search strategy

- Part based



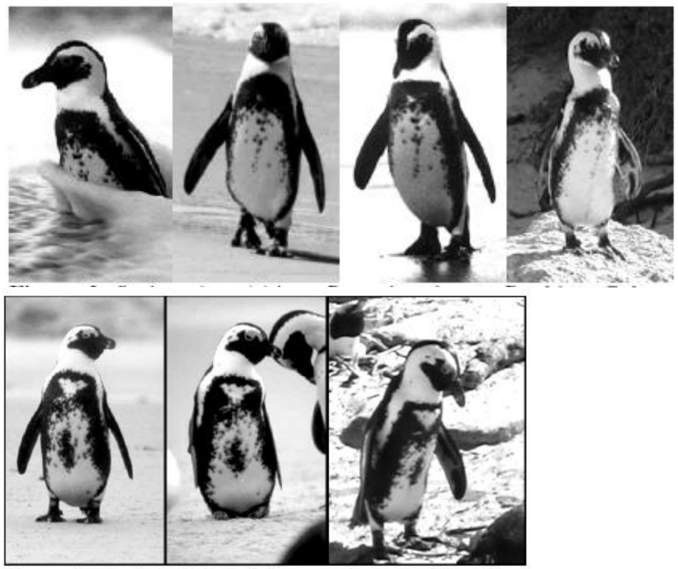
Here, invariance in translation and scale is achieved by the search strategy: the classifier is evaluated at all locations (by translating the image) and at all scales (by scaling the image in small steps).

The search cost can be reduced using a cascade.

Summary and reading materials

- Basic pipeline for window-based detection
  - Model/representation/classifier choice
  - Sliding window and classifier scoring
- Boosting classifiers: general idea
- Viola-Jones face detector
  - key ideas: rectangular features, Adaboost for feature selection, cascade
- Some reading suggestions:
  - Richard Szeliski: Computer Vision: Algorithms and Applications, Chap 14.1 http://szeliski.org/Book/
  - Friedman, Hastie, Tibshirani. “Additive Logistic Regression: a Statistical View of Boosting” (1998)
  - Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001. The paper is available online at http://www.ai.mit.edu/people/viola/
  - OpenCV documentation: http://opencv.itseez.com/modules/objdetect/doc/cascade_classification.html
  - Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection., In IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR’2005), pp. 886–893, San Diego, CA.

A cool (pun intended) example



Fifth International Penguin Conference, Ushuaia, Tierra del Fuego, Argentina, September 2004

Fifth International Penguin Conference
Ushuaia, Tierra del Fuego, Argentina

Automated Visual Recognition of Individual African Penguins

Tilo Burghardt,
Barry Thomas, Peter J Barham, Janko Ćalić

University of Bristol, Department of Computer Science,
MVB Woodland Road, Bristol BS8 1UB, United Kingdom,
September 2004

burghard@cs.bris.ac.uk

This project uses the Viola-Jones Adaboost face detection algorithm to detect penguin chests, and then matches the pattern of spots to identify a particular penguin.

Adaboost for chest stripe detection

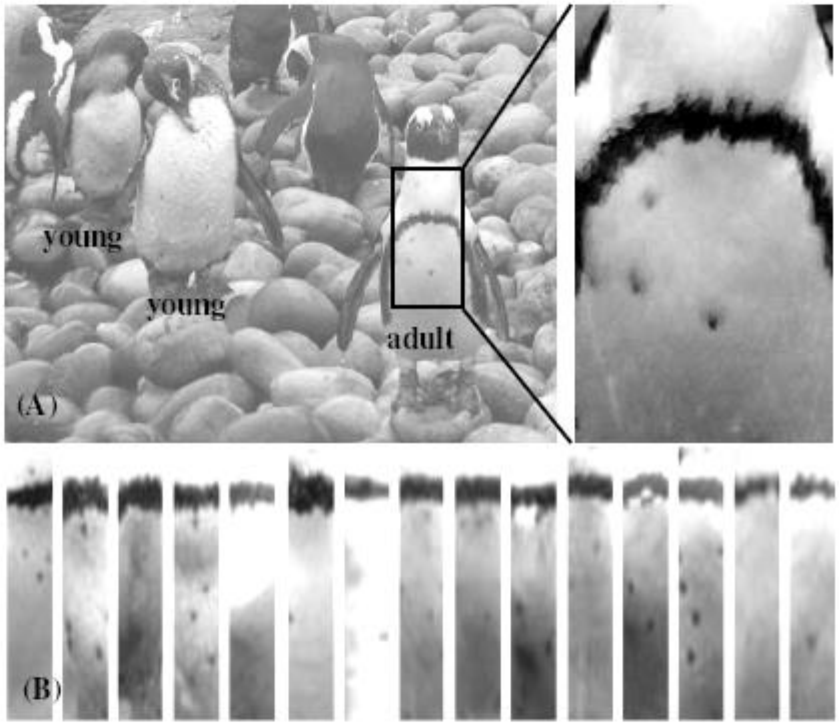
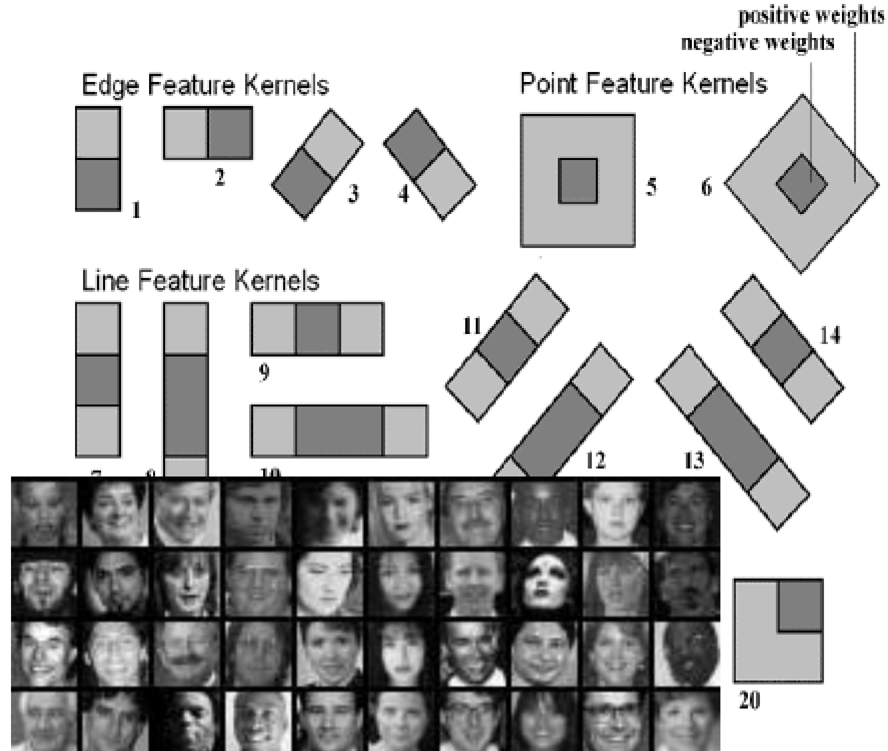


Figure 6. *Distinctive Chest Stripe of Adult African Penguins:* **(A)** Adult African penguins carry a distinctive and stable black stripe on their chest whilst young members of the species still change the colour of their chest feathers. **(B)** Various chests of adult African penguins under different lighting conditions. (figure source [19])



Use rectangular features, select good features to distinguish the chest from non-chests with Adaboost

Burghart, Thomas, Barham, and Calic. Automated Visual Recognition of Individual African Penguins , 2004.

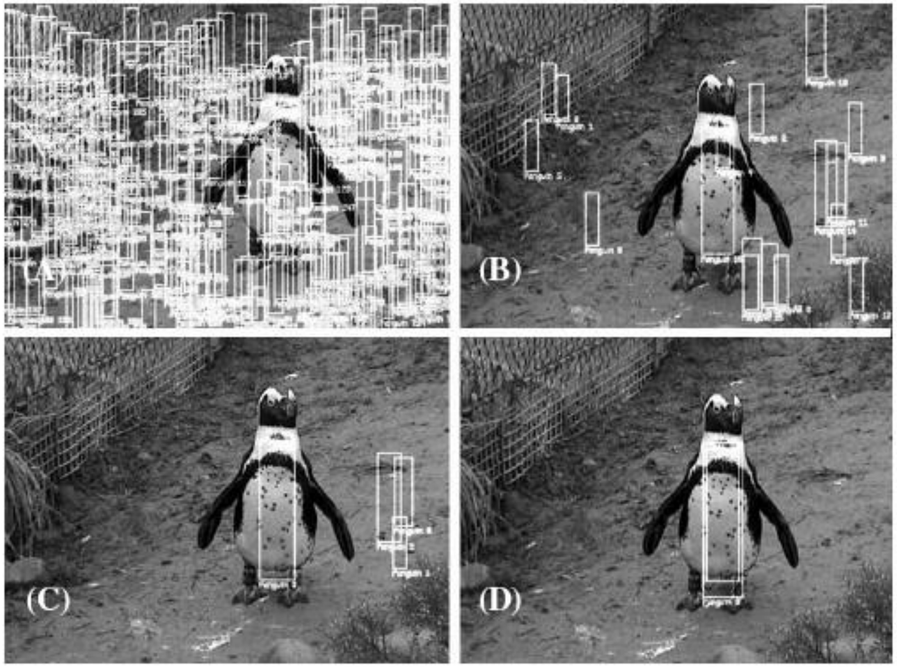


Figure 12. *Application of Attentional Cascades on Chests:* **(A)** Image areas that are accepted as likely to represent a chest after one stage are marked as white rectangles. **(B)** After three stages… **(C)** After five stages… **(D)** …and after seven stages with final result. (figure source [18], [19])

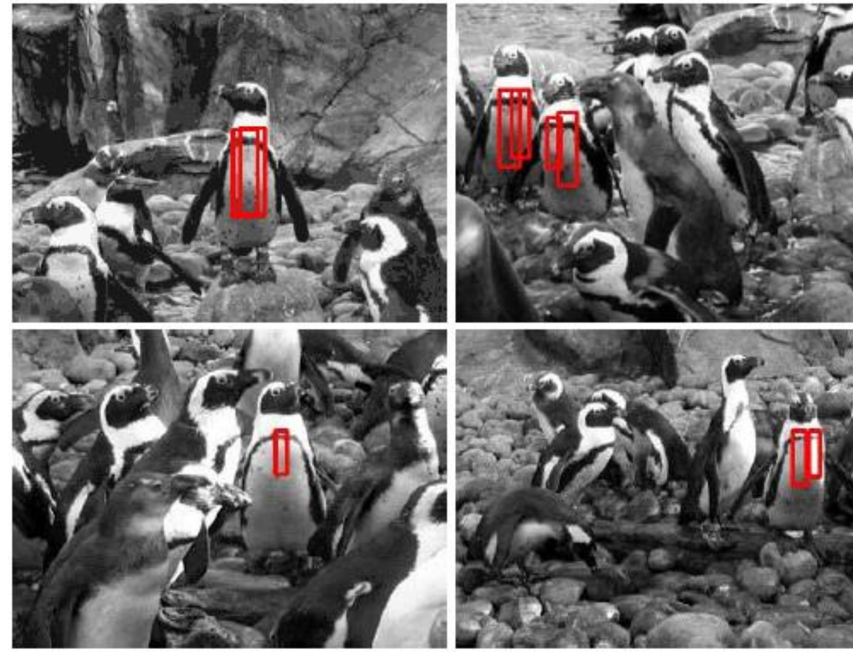


Figure 10. *AoI Detector Spotting Frontal Penguin Chests:* The detector was tested on a series of black and white still images and footage. Some result images are shown above. The detector might fire several times on one and the same chest instance. (figure

Attentional cascade

Penguin chest detections

Burghart, Thomas, Barham, and Calic. Automated Visual Recognition of Individual African Penguins , 2004.

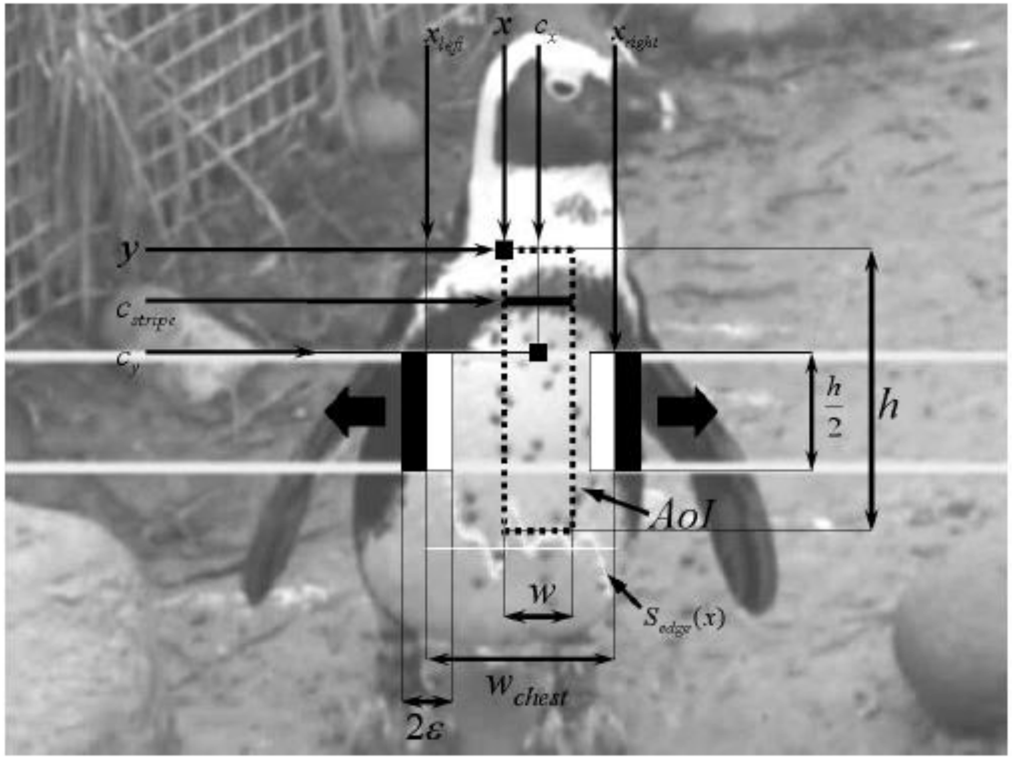
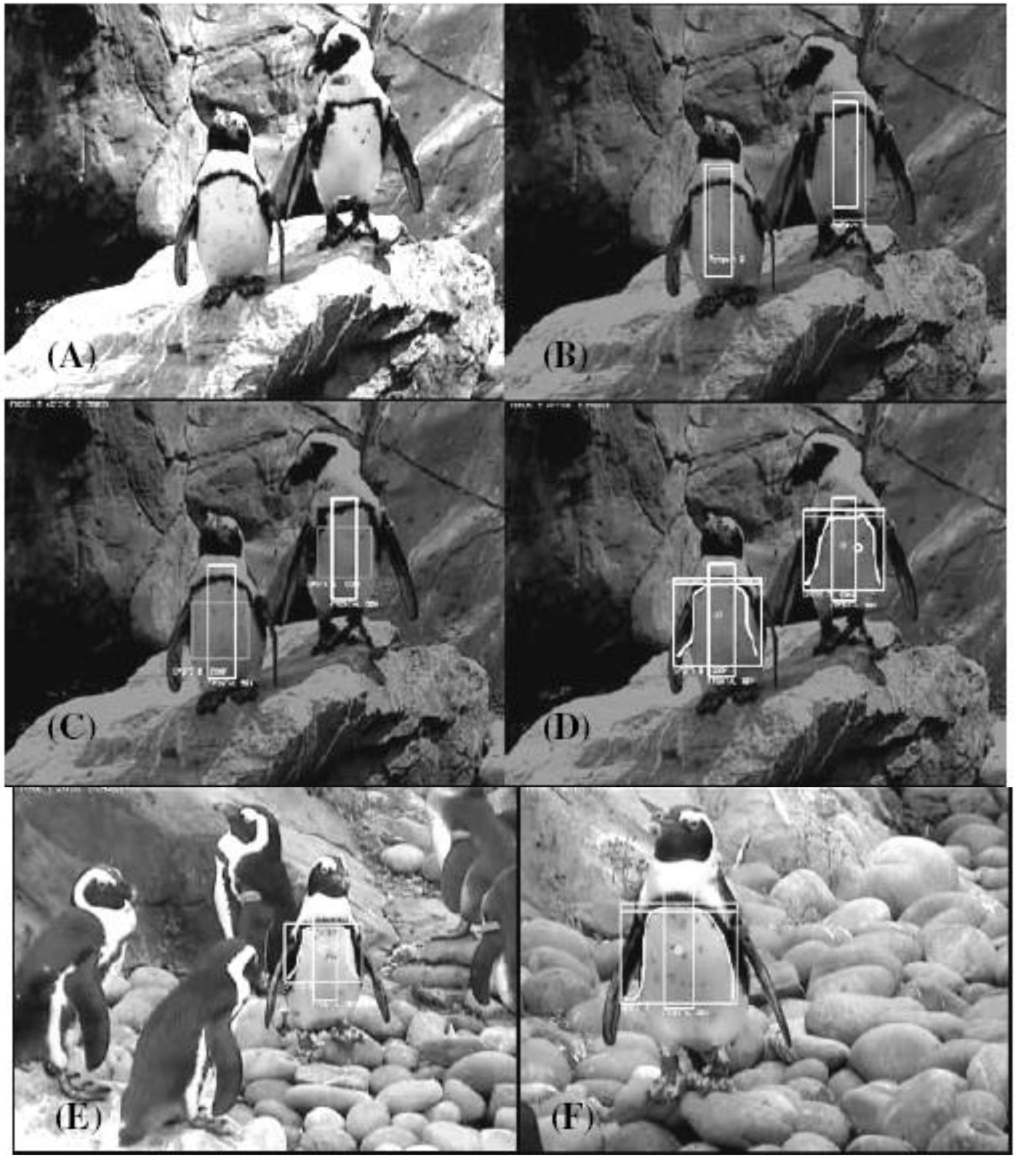


Figure 14. *Visual Description of the Chest Width Measurement:* Starting from an upper central point of the chest AoI two locally operating edge detectors moving apart search for the left and right boundary of the assumed chest. (figure source [17])

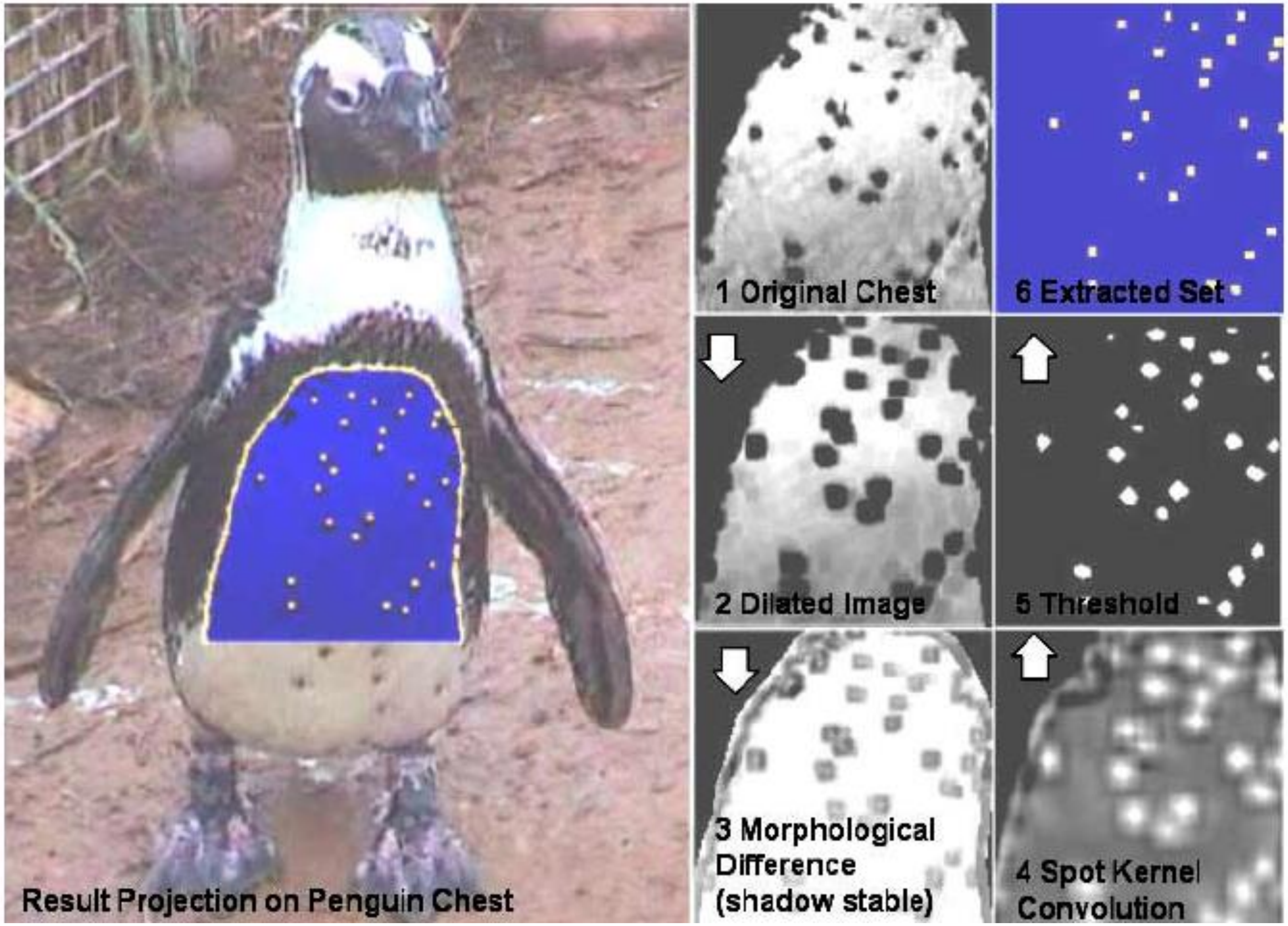
Given a detected chest, try to extract the whole chest for this particular penguin.

Burghart, Thomas, Barham, and Calic. Automated Visual Recognition of Individual African Penguins , 2004.



Example detections

Burghart, Thomas, Barham, and Calic. Automated Visual Recognition of Individual African Penguins , 2004.



Perform **identification** by matching the pattern of spots to a database of known penguins.

Burghart, Thomas, Barham, and Calic. Automated Visual Recognition of Individual African Penguins , 2004.

Penguin detection & identification

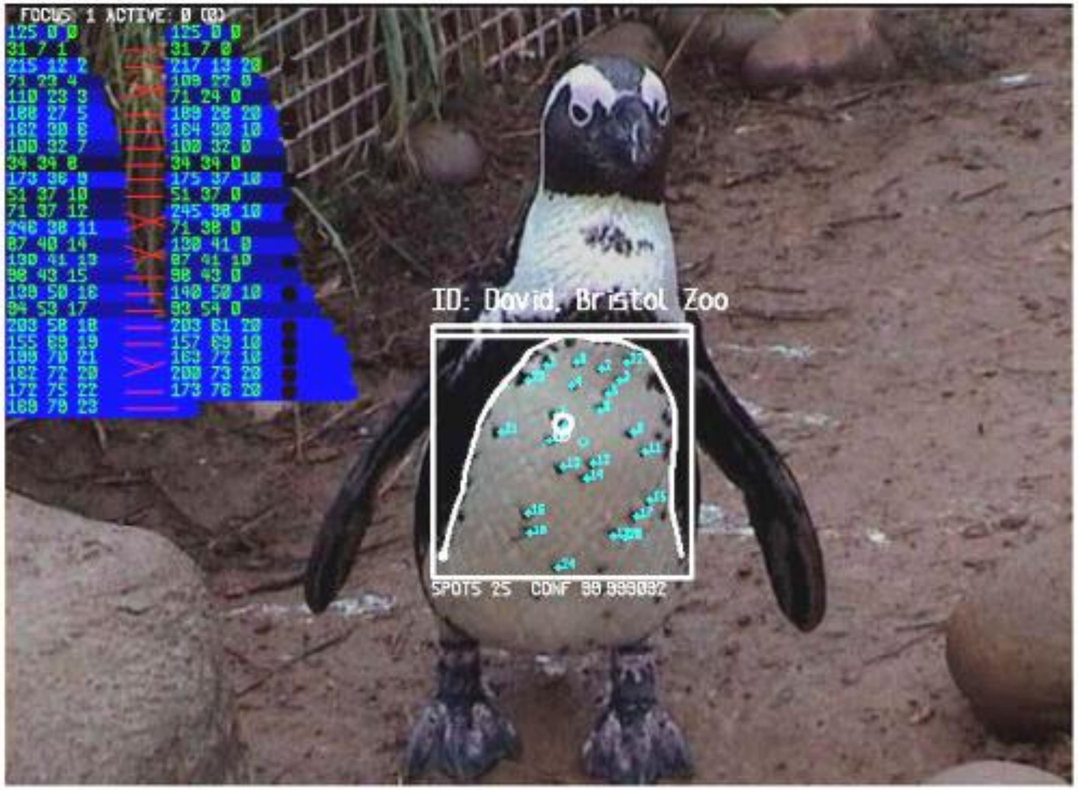


Figure 1. *Identification of an African Penguin by its Chest Pattern:* Screenshot of Software Prototype; African penguins carry a unique pattern of black spots on their chest. The detection of the chest location and the decomposition of the spot pattern allow checking a photographed individual (here penguin 'David' from Bristol Zoo) against a population database. (figure source [18], [19])

Burghart, Thomas, Barham, and Calic. Automated Visual Recognition of Individual African Penguins , 2004.