Applied Computer Vision

David Vernon
Carnegie Mellon University Africa

vernon@cmu.edu
www.vernon.eu

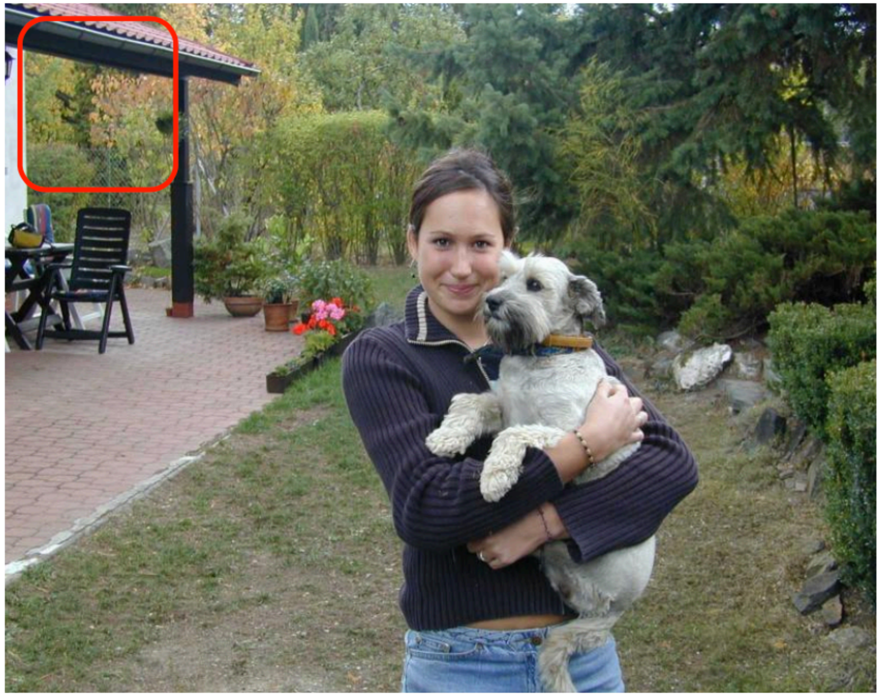
Lecture 19

Object Recognition

Haar features and object detection
Face detection

Haar Features & Object Detection

Basic idea: slide a window across image and evaluate a face model at every location



Credit: Svetlana Lazebnik

Haar Features & Object Detection

Sliding window detector must evaluate tens of thousands of location/scale combinations

- Faces are rare: 0–10 per image
- For computational efficiency, we should try to spend as little time as possible on the non-face windows
- A megapixel image has ~10^6 pixels and a comparable number of candidate face locations
- To avoid having a false positive in every image, the false positive rate has to be less than 10^{-6}

Credit: Svetlana Lazebnik

Haar Features & Object Detection

Viola and Jones Face Detection

- A seminal approach to real-time object detection
- Training is slow, but detection is very fast

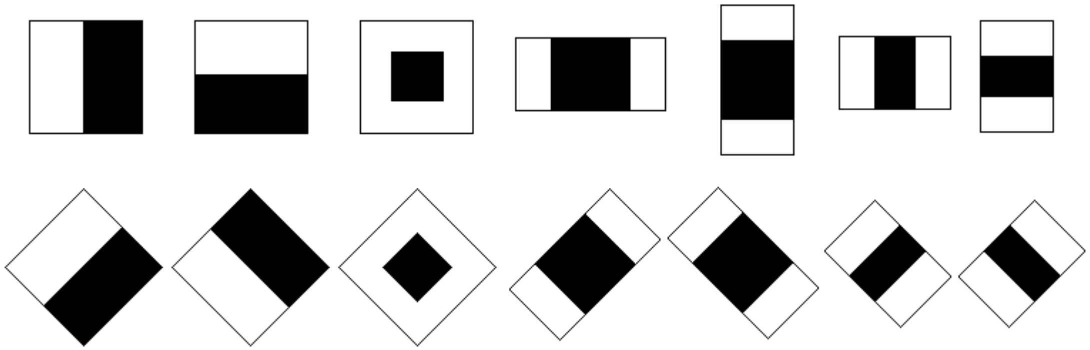
- Key ideas:
  - Haar-like image features
  - Integral images for fast feature evaluation
  - Boosting for feature selection
  - Attentional cascade for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004

Haar Features & Object Detection

Haar features: +1 and -1 filter coefficients



Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Haar Features & Object Detection

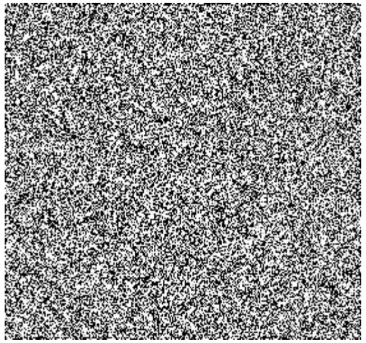
Feature value is sum of the pixels in the white region minus the sum of the pixels in the black region

Value = ∑ (pixels in white area) – ∑ (pixels in black area)

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

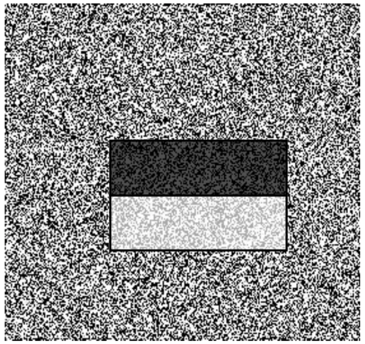
Haar Features & Object Detection

Feature value is sum of the pixels in the white region minus the sum of the pixels in the black region



Source images

Low filter response



High filter response

Credit: Svetlana Lazebnik

Haar Features & Object Detection

- The system is trained using at a particular scale in a standard size sub-image
- However, the classifiers are easily resized allowing this object detection technique to be applied at different scales

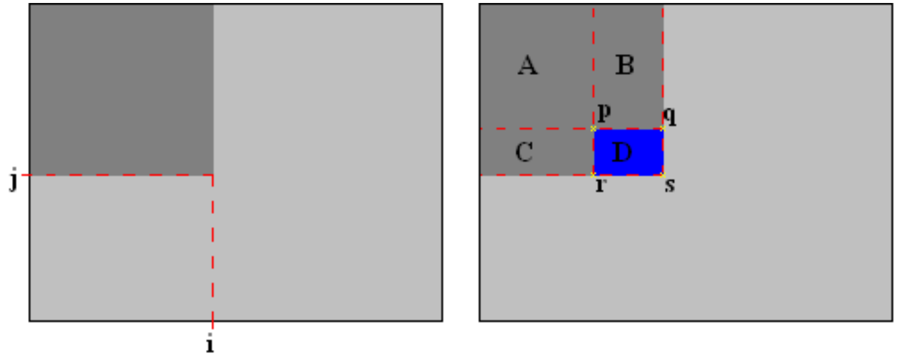
Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Haar Features & Object Detection

Efficient computation using an **integral image** [Viola and Jones 2001]

- The integral image $ii(i, j)$ of some image $ni(i', j')$

- Every point in the image $ii(i, j)$ is the sum of all pixels value in image $ni(i', j')$ where $i' \le i$ and $j' \le j$



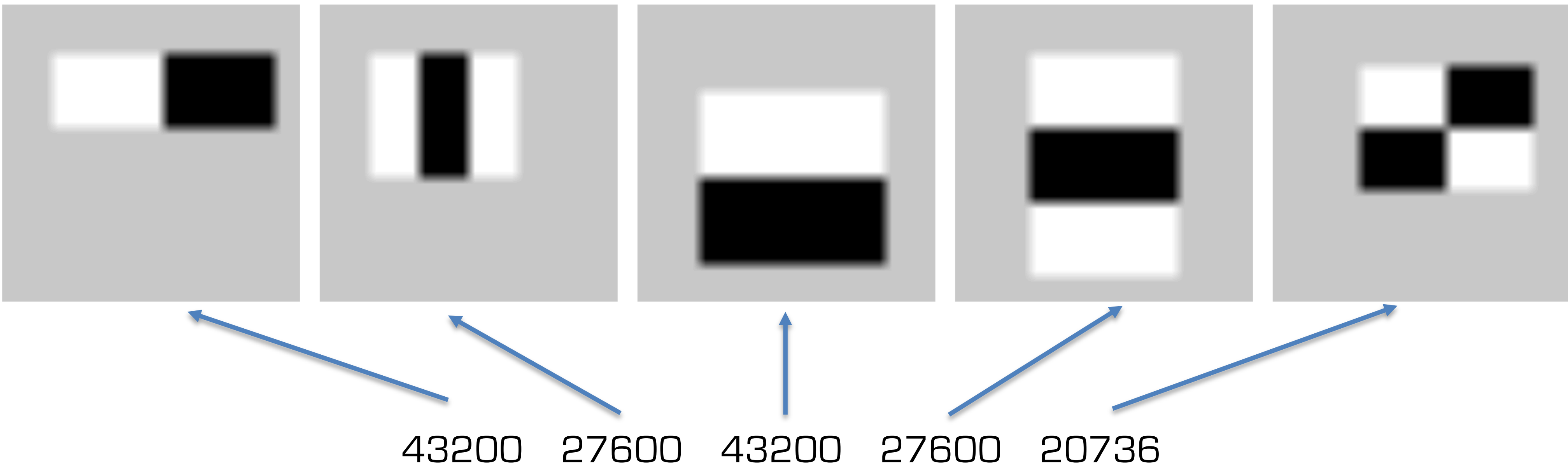
$$ii(i, j) = \Sigma_{i'=0..i} \Sigma_{j'=0..j} ni(i', j')$$

$$\begin{aligned} \text{sum(D)} &= ii(p) + ii(s) - ii(q) - ii(r) \\ &= \text{sum(A)} + \text{sum(A+B+C+D)} - \text{sum(A+B)} - \text{sum(A+C)} \end{aligned}$$

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Haar Features & Object Detection

162336 features in a 24x24 pixel image



Credit: Wang 2013

Haar Features & Object Detection

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate strong *ensemble* classifier
  - A weak learner need only do better than chance

- Training consists of multiple *boosting rounds*
  - During each boosting round, we select a weak learner that does well on examples that were hard for the previous weak learners
  - "Hardness" is captured by weights attached to training examples

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

Credit: Svetlana Lazebnik

Haar Features & Object Detection

Training

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest *weighted* training error
  - Raise the weights of training examples misclassified by current weak learner

Credit: Svetlana Lazebnik

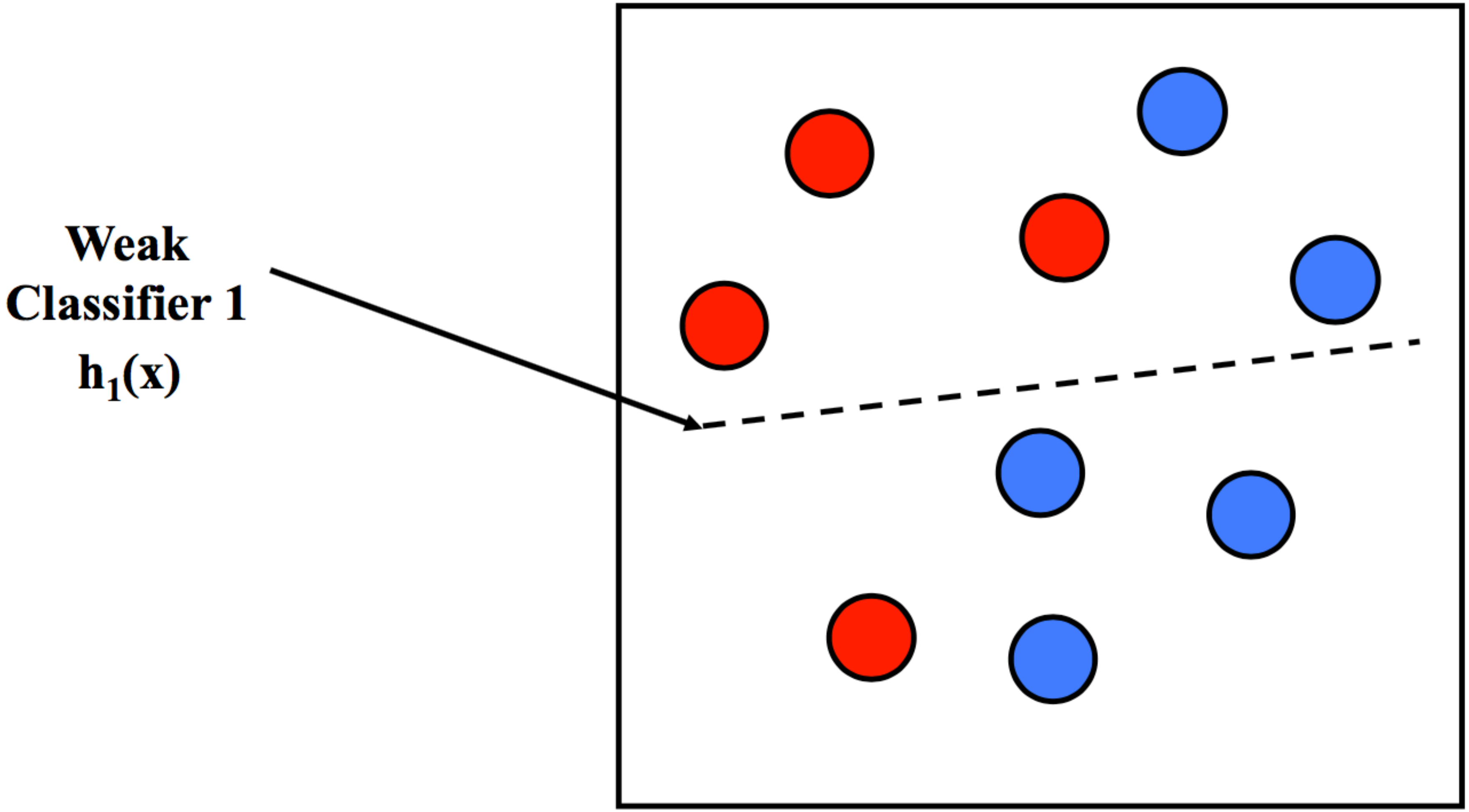
Haar Features & Object Detection

Training

- Compute final classifier as linear combination of all weak learners
  - weight of each learner is directly proportional to its accuracy
- Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme
  - e.g., AdaBoost

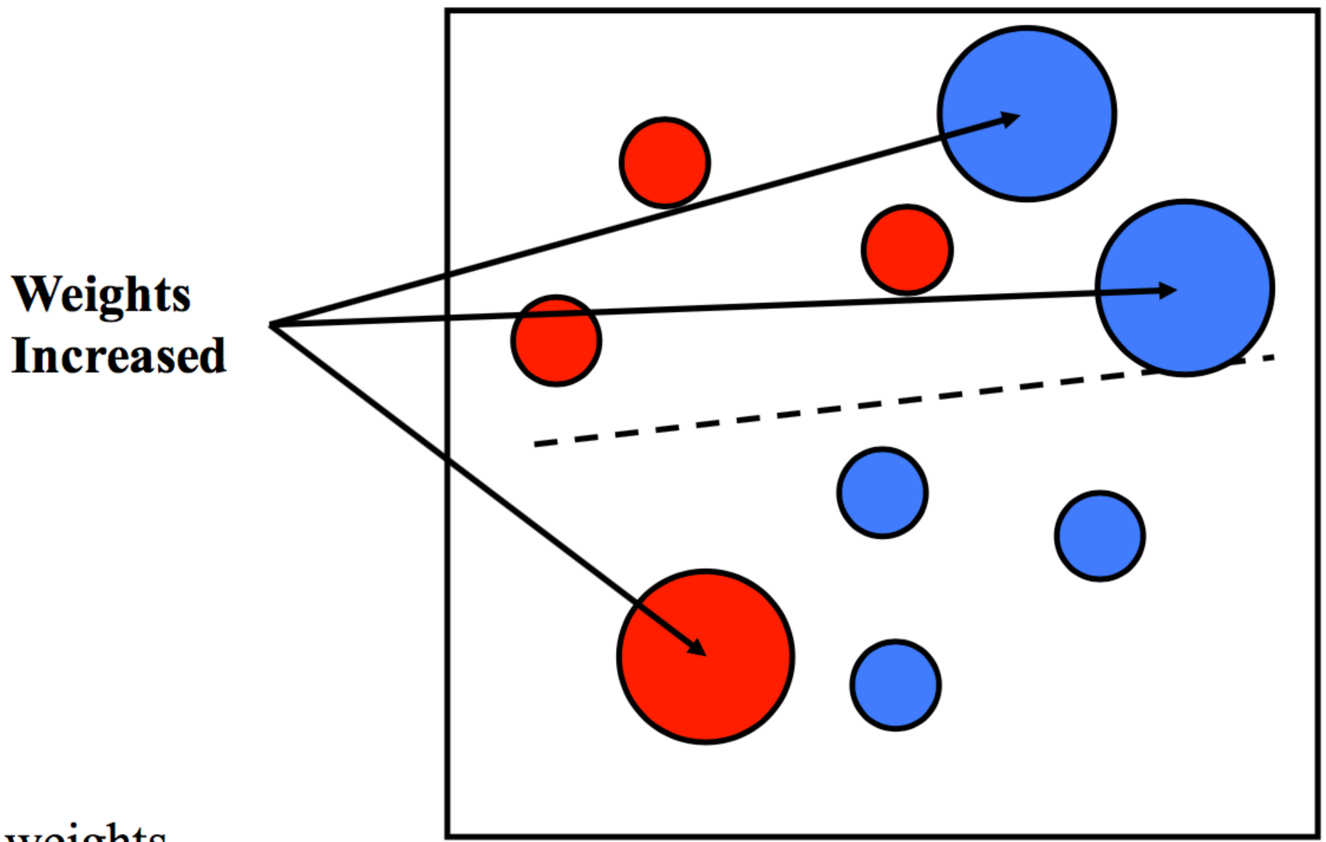
Credit: Svetlana Lazebnik

Haar Features & Object Detection



Credit: Svetlana Lazebnik

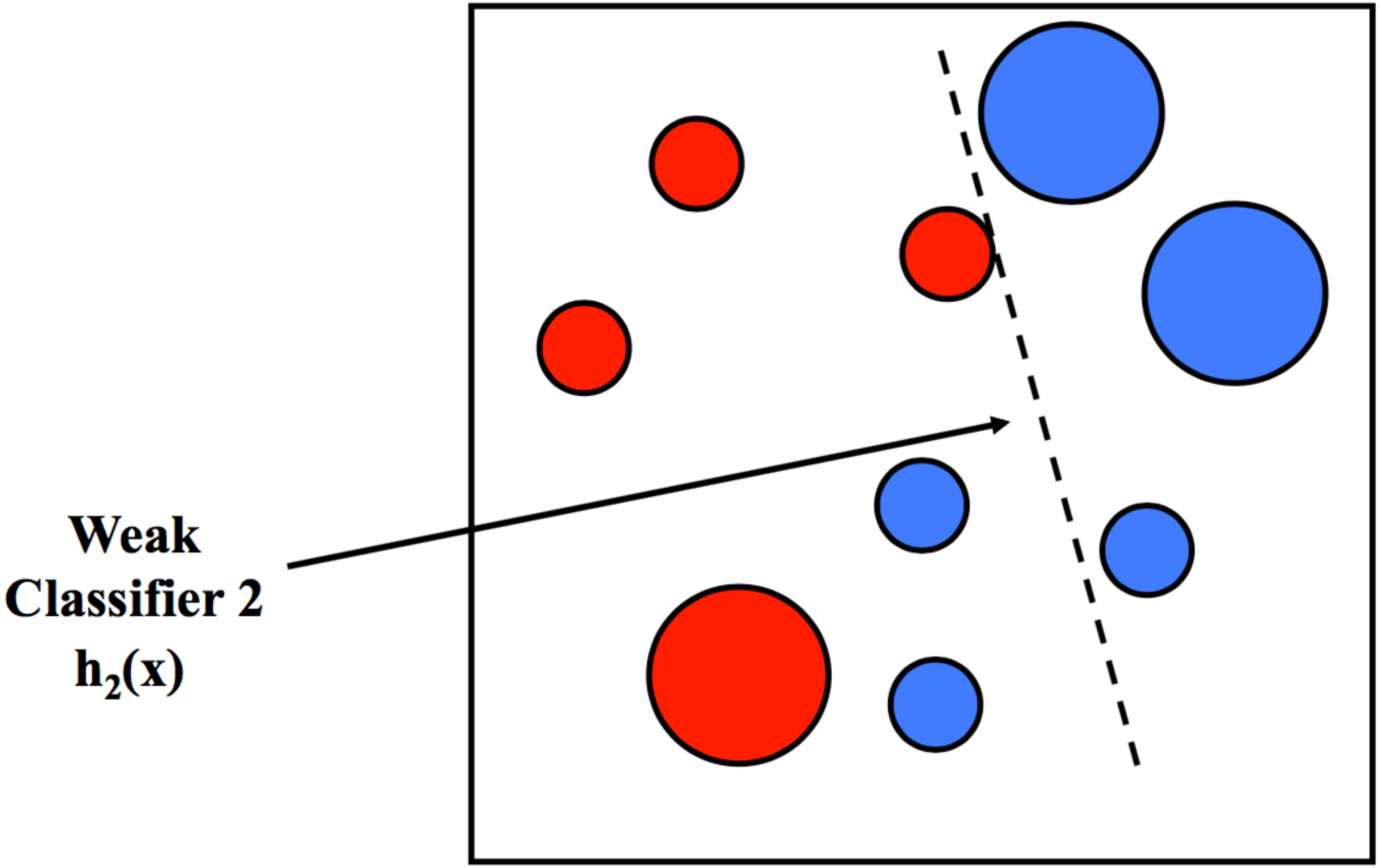
Haar Features & Object Detection



Increasing the weights forces subsequent classifiers to focus on residual errors

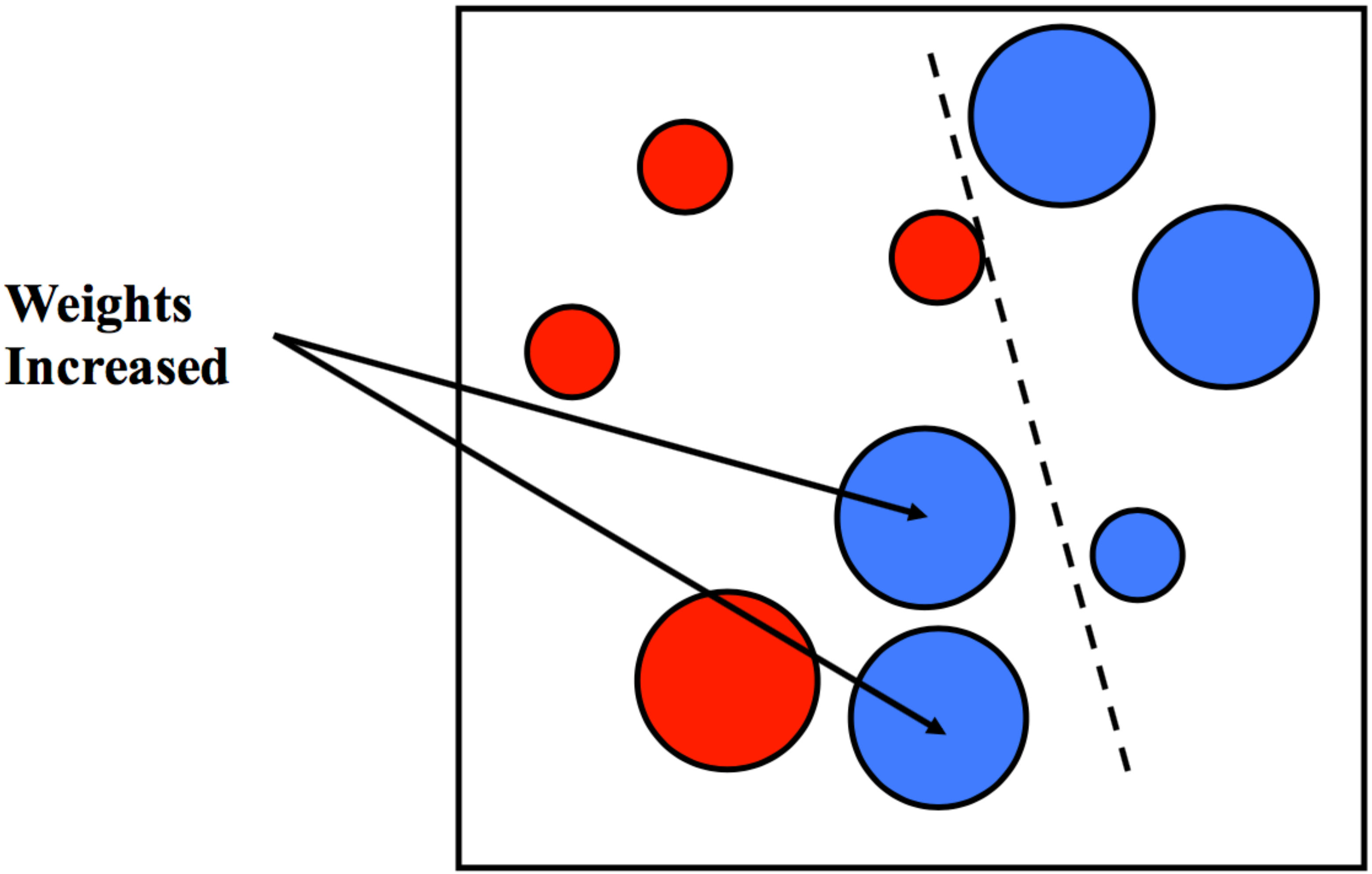
Credit: Svetlana Lazebnik

Haar Features & Object Detection



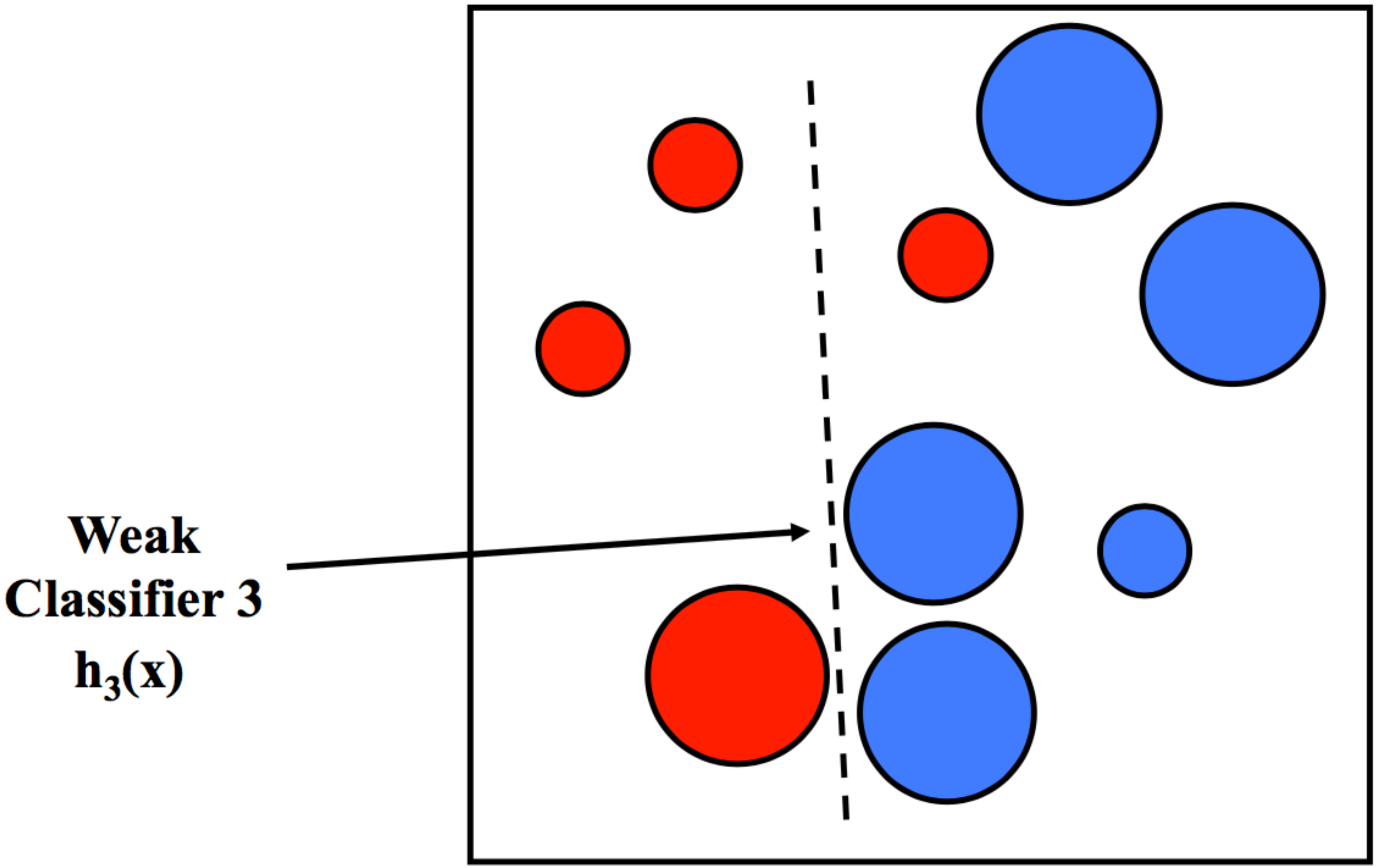
Credit: Svetlana Lazebnik

Haar Features & Object Detection



Credit: Svetlana Lazebnik

Haar Features & Object Detection

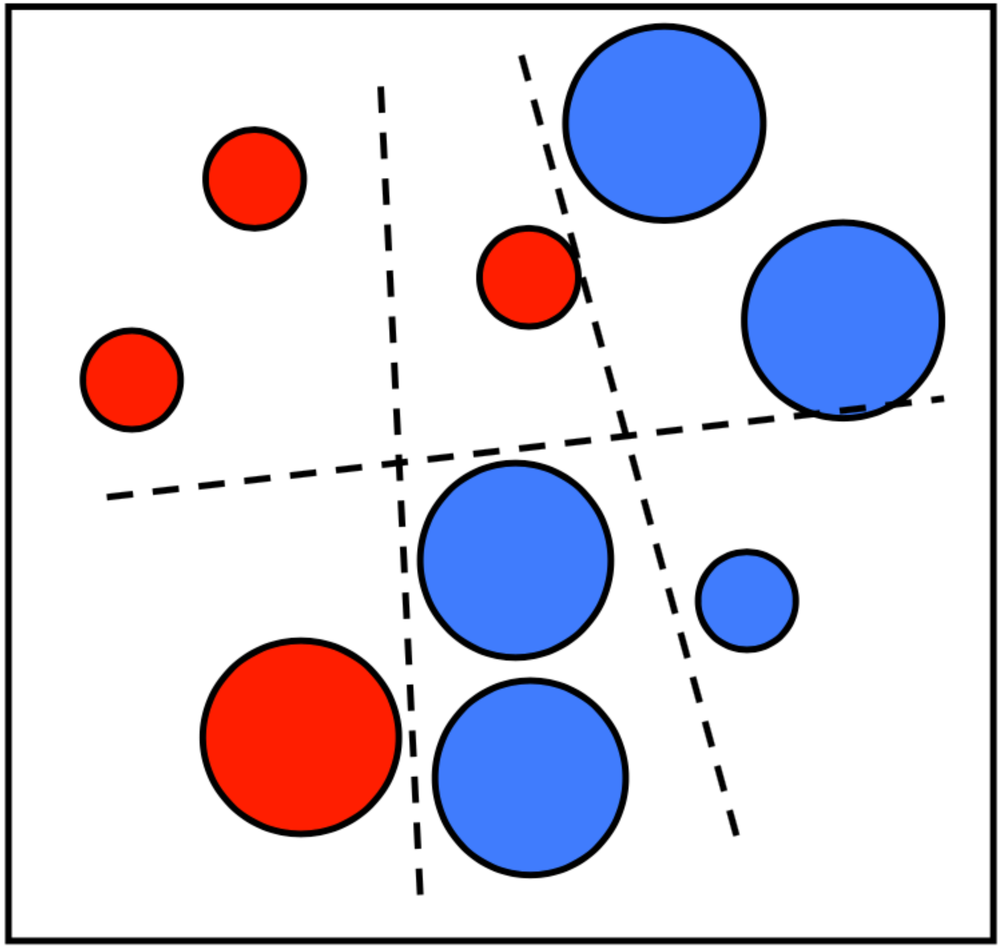


Credit: Svetlana Lazebnik

Haar Features & Object Detection

Final classifier is a weighted combination of the weak classifiers

$$h(\boldsymbol{x}) = \text{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\boldsymbol{x})\right]$$



Credit: Svetlana Lazebnik

Haar Features & Object Detection

- Weak learners here are defined based on thresholded Haar features

value of Haar feature

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

window

parity +1/-1

threshold

- Note: the parity value just serves to change the direction of the threshold to be either less that or greater than, as appropriate

Credit: Svetlana Lazebnik

Haar Features & Object Detection

- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter & threshold combination as the weak learner
  - Reweight examples

- Computational complexity of learning:

  $O(MNK)$

  M rounds, N examples, K features

Credit: Svetlana Lazebnik

Haar Features & Object Detection

Given n example images $x_1..x_n$ together with classifications $y_1..y_n$
where $y_i = 0, 1$ for negative and positive examples, respectively

Initialise weights $w_{1,i} = 1 / (2*(m*(1- y_i) + l*y_i))$
where m and l are the number of negative and positive examples respectively

For t=1,…,T

1. Normalize the weights (i.e. for all i): $w_{t,i} = w_{t,i} / (\Sigma_{j=1..n} w_{t,j})$

2. For each feature, j, train a weak classifier $h_j(x)$ and evaluate the error taking into account the weights: $\varepsilon_j = \Sigma_i \; w_{t,i} \mid h_j(x_i) - y_i \mid$

3. Select the classifier, $h_j(x)$, with the lowest ε_j, save as $c_t(x)$ with error ε_t

4. Update the weights (i.e. for all i): $w_{t+1,i} = w_{t,i} \, \beta_t^{(1- e_i)}$
where $e_i = \mid c_t(x_i) - y_i \mid$ and $\beta_t = \varepsilon_t / (1-\varepsilon_t)$

The final strong classifier is: $h(x)$= 1 if $\Sigma_{t=1..T} \alpha_t c_t(x) \geq ½ \, \Sigma_{t=1..T} \alpha_t$
0 *otherwise*
where $\alpha_t = log \; 1/\beta_t$

Haar Features & Object Detection

Given n example images $x_1..x_n$ together with classifications $y_1..y_n$
where $y_i = 0, 1$ for negative and positive examples, respectively

Initialise weights $w_{1,i} = 1 / (2*(m*(1- y_\mathrm{i}) + l*y_\mathrm{i}))$
where m and l are the number of negative and positive examples respectively

For t=1,…,T

1. Norm

2. For … ing into acc

3. Selec

4. Upda

The fin

The weights are changed at each "round of boosting": $t = 1, \ldots, T$

Considering $w_{a,\, b}$

a is the number of the current round of boosting

b is the training image number

Haar Features & Object Detection

Given n example images $x_1..x_n$ together with classifications $y_1..y_n$

where $y_i = 0, 1$ for negative and positive examples, respectively

Initialise weights $w_{1,i} = 1 / (2*(m*(1- y_i) + l*y_i))$

where m and l are the number of negative and positive examples respectively

For t=1,…,T

1. Normalize the weights (i.e. for all i): $w_{t,i} = w_{t,i} / (\Sigma_{j=1..n} w_{t,j})$

2. For each feature, j, train a weak classifier $h_j(x)$ and evaluate the error taking into account the weights: $\varepsilon_j = \Sigma_i \; w_{t,i} \mid h_j(x_i) - y_i \mid$

3. Selec

4. Upda

The fina

Consequently, the sum of all weights will be 1

This means that we can view $w_{t,i}$ as a probability distribution

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Haar Features & Object Detection

Given n example images $x_1..x_n$ together with classifications $y_1..y_n$
where $y_i = 0, 1$ for negative and positive examples, respectively

Initialise weights $w_{1,i} = 1 / (2*(m*(1- y_i) + l*y_i))$
where m and l are the number of negative and positive examples respectively

For $t=1,\ldots,T$

1. Normalize the weights (i.e. for all i): $w_{t,i} = w_{t,i} / (\Sigma_{j=1..n} w_{t,j})$
2. For each feature, j, train a weak classifier $h_j(x)$ and evaluate the error taking into account the weights: $\varepsilon_j = \Sigma_i \; w_{t,i} \mid h_j(x_i) - y_i \mid$
3. Select the classifier, $h_j(x)$, with the lowest ε_j, save as $c_t(x)$ with error ε_t
4. Update the weights (i.e. for all i): $w_{t+1,i} = w_{t,i} \, \beta_t^{(1- e_i)}$

The final

Training a classifier means taking the single feature and determining the threshold which minimizes the misclassifications

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Haar Features & Object Detection

Given n example images $x_1..x_n$ together with classifications $y_1..y_n$
where $y_i = 0, 1$ for negative and positive examples, respectively

Initialise weights $w_{1,i} = 1 / (2*(m*(1- y_i) + l*y_i))$
where m and l are the number of negative and positive examples respectively

For $t=1,\ldots,T$

1. Normalize the weights (i.e. for all i): $w_{t,i} = w_{t,i} / (\Sigma_{j=1..n}\, w_{t,j})$
2. For each feature, j, train a weak classifier $h_j(x)$ and evaluate the error taking into account the weights: $\varepsilon_j = \Sigma_i\ w_{t,i} \mid h_j(x_i) - y_i \mid$
3. Select the classifier, $h_j(x)$, with the lowest ε_j, save as $c_t(x)$ with error ε_t
4. Update the weights (i.e. for all i): $w_{t+1,i} = w_{t,i}\, \beta_t^{(1-e_i)}$

The final

Pick the best weak classifier ... i.e. the one with the lowest error

where $\alpha_t = log\ 1/\beta_t \alpha_t = log\ 1/\beta_t$

Haar Features & Object Detection

Given n example images $x_1..x_n$ together with classifications $y_1..y_n$
where $y_i = 0, 1$ for negative and positive examples, respectively

Initialise weights $w_{1,i} = 1 / (2*(m*(1- y_i) + l*y_i))$
where m and l are the number of negative and positive examples respectively

For t=1,…,T

1. Normalize the weights (i.e. for all i): $w_{t,i} = w_{t,i} / (\Sigma_{j=1..n}\, w_{t,j})$
2. For each feature, j, train a weak classifier $h_j(x)$ and evaluate the error taking into account the weights: $\varepsilon_j = \Sigma_i\, w_{t,i}\, |\, h_j(x_i) - y_i\, |$
3. Select the classifier, $h_j(x)$, with the lowest ε_j, save as $c_t(x)$ with error ε_t
4. Update the weights (i.e. for all i): $w_{t+1,i} = w_{t,i}\, \beta_t^{(1- e_i)}$
where $e_i = |\, c_t(x_i) - y_i\, |$ and $\beta_t = \varepsilon_t / (1-\varepsilon_t)$

The final

Update the weights on the images leaving the weights on misclassified images the same and reducing the weights on correctly classified images by β_t (slightly different formulation to previous explanation)

Haar Features & Object Detection

Given n example images $x_1..x_n$ together with classifications $y_1..y_n$
where $y_i = 0, 1$ for negative and positive examples, respectively

Initialise weights $w_{1,i} = 1 / (2*(m*(1- y_i) + l*y_i))$
where m and l are the number of negative and positive examples respectively

For $t=1,\ldots,T$

1. Normalize the weights (i.e. for all i):

2. For each f … aking into account th…

3. Select the c…

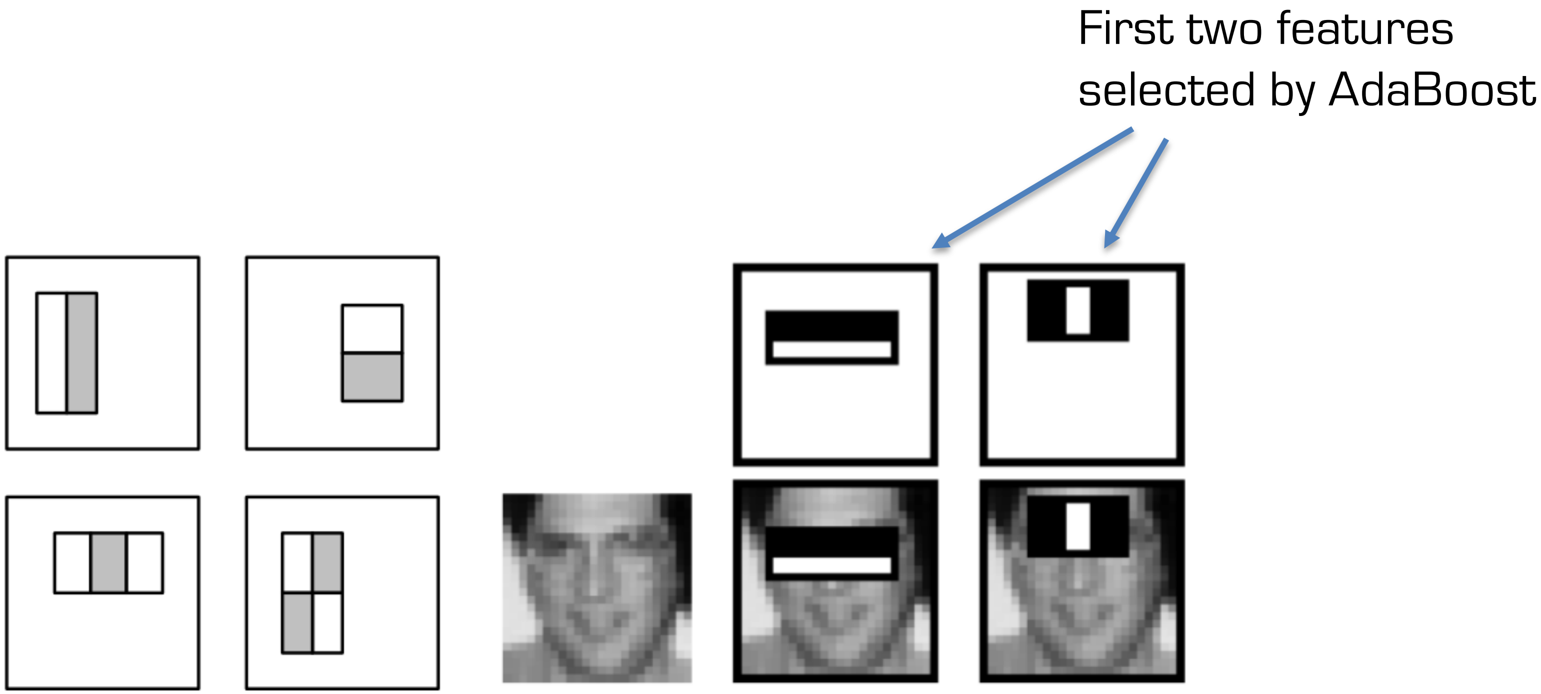
4. Update the…
where $e_i = | c_t(x_i) - y_i |$ and $\beta_t = \varepsilon_t / (1-\varepsilon_t)$

The final strong classifier is: $h(x)= 1$ if $\Sigma_{t=1..T}\alpha_t c_t(x) \geq ½ \, \Sigma_{t=1..T}\alpha_t$
0 *otherwise*
where $\alpha_t = log\ 1/\beta_t$

The final strong classifier is a weighted combination of the weak classifiers where the weights are related to the training errors from each of the weak classifiers

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Haar Features & Object Detection



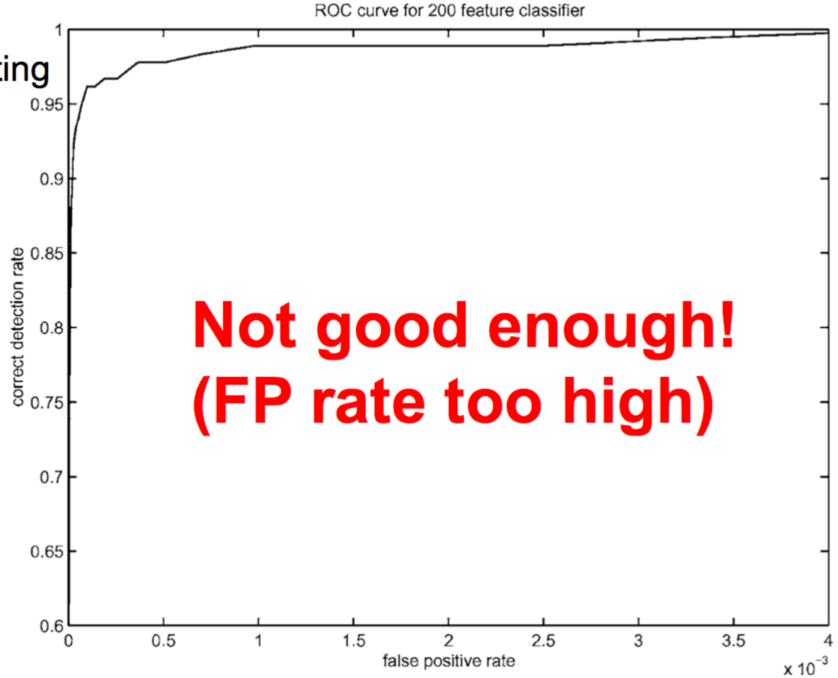
This feature combination can yield 100% detection rate and 50% false positive rate

Credit: Markus Vincze, Technische Universität Wien

Haar Features & Object Detection

- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084

Receiver operating characteristic (ROC) curve



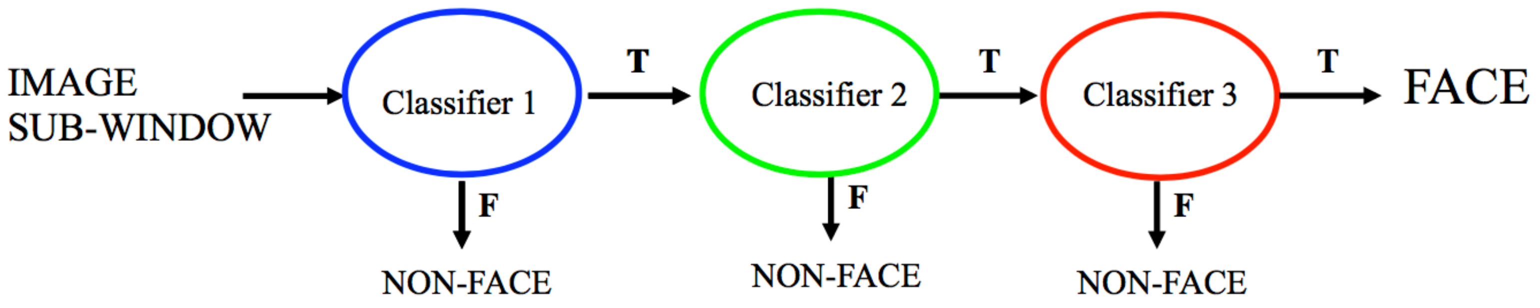
- Recall that to avoid having a false positive in every image, our false positive rate has to be less than 10^{-6}

Credit: Svetlana Lazebnik

Haar Features & Object Detection

Classifier cascade

- Start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



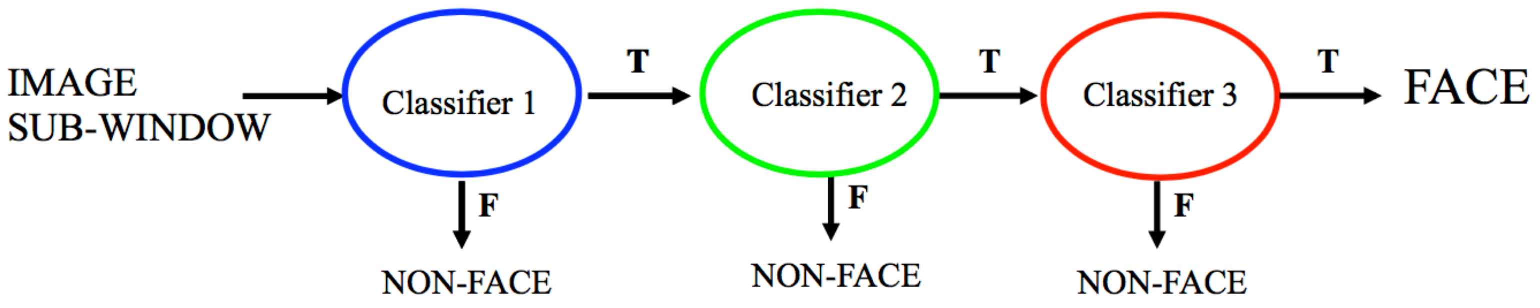
Credit: Svetlana Lazebnik

Haar Features & Object Detection

Classifier cascade

Solves several problems:

- Improves speed by early rejection of non-face regions by simple classifiers
- Reduces false positive rates

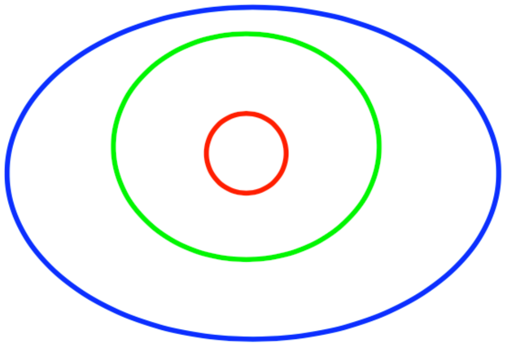


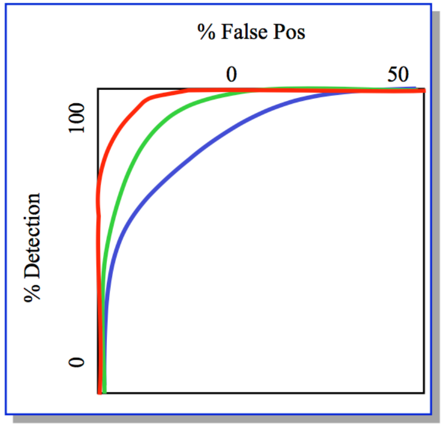
Credit: Svetlana Lazebnik

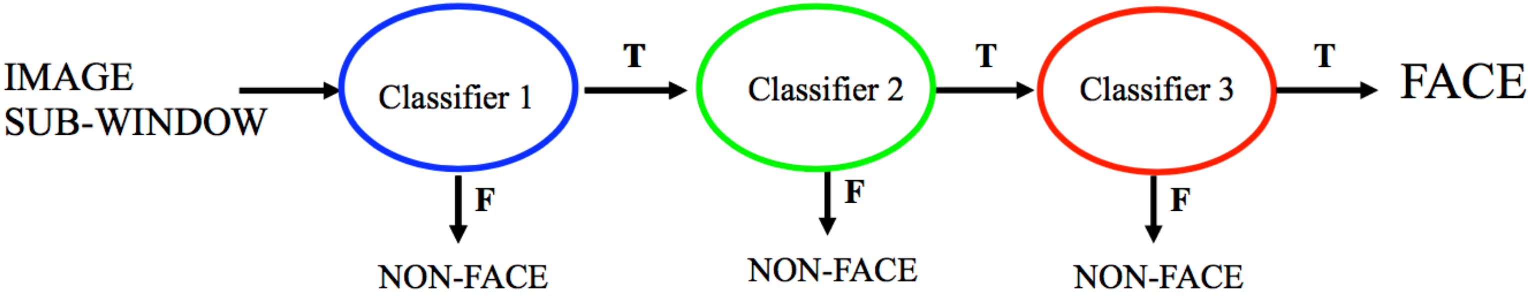
Haar Features & Object Detection

Classifier cascade

Chain classifiers that are progressively more complex and have lower false positive rates:







Credit: Svetlana Lazebnik

Haar Features & Object Detection

Classifier cascade

- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages

- A detection rate of 0.9 and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 ($0.3^{10} \approx 6 \times 10^{-6}$)

Credit: Svetlana Lazebnik

Haar Features & Object Detection

Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a *validation set*
- If the overall false positive rate is not low enough, then add another stage

Credit: Svetlana Lazebnik

Haar Features & Object Detection

Training the cascade

- The classifiers in the cascade are trained using AdaBoost on the remaining set of example images
- Thus, if the first stage classifier rejects a number of images then these images are not included when training the second stage classifer
- But use false positives from current stage as the negative training examples for the next stage

Haar Features & Object Detection

Training the cascade

- In this way the sub-images which do not contain the object are gradually removed from consideration leaving just the objects which are sought

- Most negative windows are rejected by the first couple of stages in the cascade

  - hence the computation for these windows is low (relative to those which have to go through more stages in the cascades)

  - It is this which reduces the computation time

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

Haar Features & Object Detection

Training the cascade

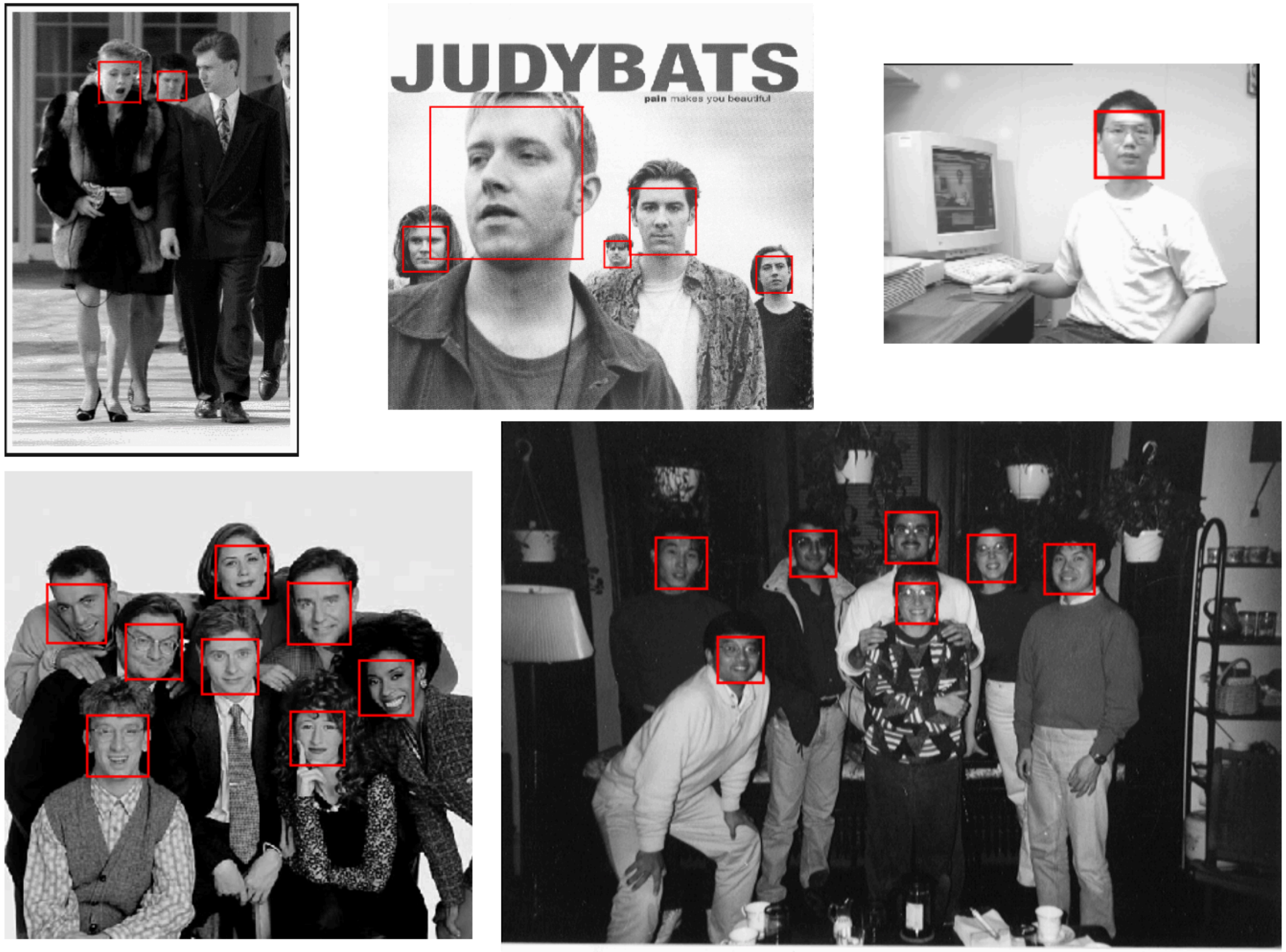
38 stages used in Voila and Jones's final face recognition system

Haar Features & Object Detection

Face identification

- 38 stages
- 6000+ features
- 4916 positive samples
- 9544 negative samples
- Scale independence

Haar Features & Object Detection



[Viola, Jones 2004]

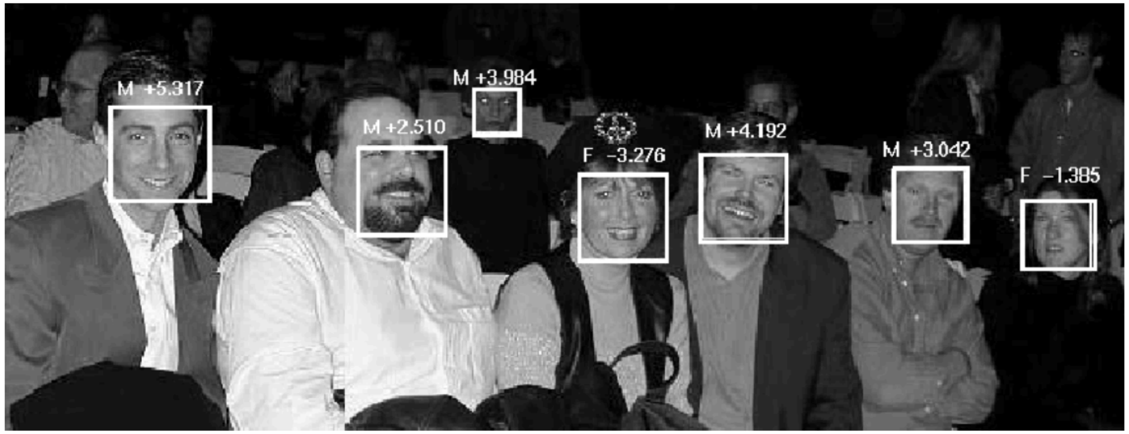
Credit: Svetlana Lazebnik

Haar Features & Object Detection

Facial Feature Localization

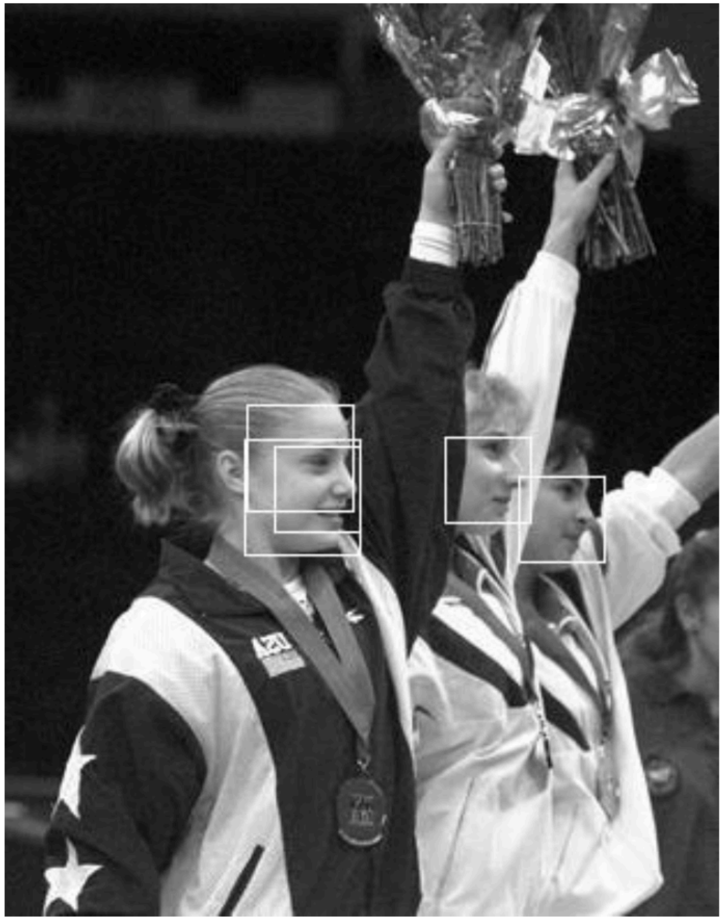
Profile Detection

Male vs.
female



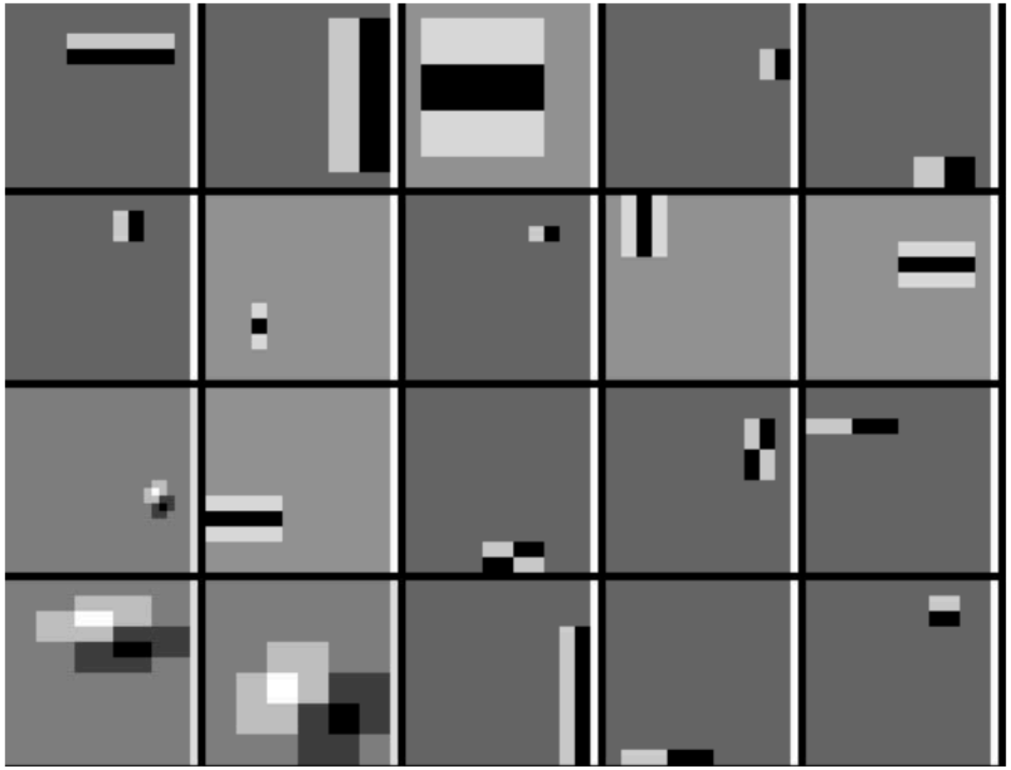
Credit: Svetlana Lazebnik

Haar Features & Object Detection



Credit: Svetlana Lazebnik

Haar Features & Object Detection



Credit: Svetlana Lazebnik

Demos

The following code is taken from the faceDetection project in the lectures directory of the ACV repository

See:

```
faceDetection.h
faceDetectionImplementation.cpp
faceDetectionApplication.cpp
```

```
/*
  Example use of openCV to perform face detection using Haar features and boosted classification
  ---------------------------------------------------------------------------------------------
  Application file

  David Vernon
  27 July 2017
*/

#include "faceDetection.h"

int main() {

   int end_of_file;
   bool debug = true;
   char filename[MAX_FILENAME_LENGTH];

   FILE *fp_in, *fp_out;

   if ((fp_in = fopen("../data/faceDetectionInput.txt","r")) == 0) {
     printf("Error can't open input faceDetectionInput.txt\n");
     prompt_and_exit(1);
   }

   if ((fp_out = fopen("../data/faceDetectionOutput.txt","w")) == 0) {
     printf("Error can't open output faceDetectionOutput.txt\n");
     prompt_and_exit(1);
   }

   printf("Example of how to use openCV to perform face detection using Haar features and boosed classification.\n\n");
```

```
/* ------------------------------------------------------------------------------------------
 * This code is provided as part of "A Practical Introduction to Computer Vision with OpenCV"
 * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
 */


// Load Haar Cascade(s)

char* file_location = "../data/Media/";

vector<CascadeClassifier> cascades;
char* cascade_files[] = {
   "haarcascades/haarcascade_frontalface_alt.xml" };
int number_of_cascades = sizeof(cascade_files)/sizeof(cascade_files[0]);
for (int cascade_file_no=0; (cascade_file_no < number_of_cascades); cascade_file_no++)
{
   CascadeClassifier cascade;
   string filename(file_location);
   filename.append(cascade_files[cascade_file_no]);
   if( !cascade.load( filename ) )
   {
      cout << "Cannot load cascade file: " << filename << endl;
      return -1;
   }
   else cascades.push_back(cascade);
}

/* ------------------------------------------------------------------------------------------ */
```

```
    do {

        end_of_file = fscanf(fp_in, "%s", filename);

        if (end_of_file != EOF) {
            //if (debug) printf ("%s\n",filename);

            printf("\n Performing face detection using Haar features and boosed classification on %s \n",filename);
            faceDetection(filename, cascades[HAAR_FACE_CASCADE_INDEX]);
        }
    } while (end_of_file != EOF);


    fclose(fp_in);
    fclose(fp_out);

    return 0;
}
```

```
/*
  Example use of openCV to perform face detection using Haar features and boosted classification
  ---------------------------------------------------------------------------------------------
  Implementation file

  David Vernon
  27 July 2017
*/

#include "faceDetection.h"

void faceDetection(char *filename, CascadeClassifier& cascade) {

   VideoCapture camera;
   char inputWindowName[MAX_STRING_LENGTH]         = "Input Image";
   char outputWindowName[MAX_STRING_LENGTH]        = "Cascade of Haar Classifiers";
   Mat inputImage;
   Mat outputImage;
   char c;
   vector<Rect> faces;
   Mat gray;

   namedWindow(outputWindowName,     CV_WINDOW_AUTOSIZE);
```

```
/* check to see if the image is the camera device     */
/* if so, grab images live from the camera            */
/* otherwise proceed to process the image in the file */

if (strcmp(filename,"camera") != 0) {

   inputImage = imread(filename, CV_LOAD_IMAGE_COLOR); // Read the file

   if (!inputImage.data) {                            // Check for invalid input
      printf("Error: failed to read image %s\n",filename);
      prompt_and_exit(-1);
   }

   printf("Press any key to continue ...\n");

   cvtColor(inputImage, gray, CV_BGR2GRAY );
   equalizeHist(gray, gray );
   cascade.detectMultiScale( gray, faces, 1.1, 2, CV_HAAR_SCALE_IMAGE, Size(30, 30) );

   for (int count = 0; count < (int)faces.size(); count++ )
      rectangle(inputImage, faces[count], cv::Scalar(255,0,0), 2);

   imshow(outputWindowName, inputImage);
}
```

```
else {
   /* ---------------------------------------------------------------------------------------
    * Adapted from code provided as part of "A Practical Introduction to Computer Vision with OpenCV"
    * by Kenneth Dawson-Howe © Wiley & Sons Inc. 2014.  All rights reserved.
    */

   // Cascade of Haar classifiers (most often shown for face detection).

   //camera.open(1); // David Vernon ... this is the original code and uses an external USB camera
   camera.open(0);   // David Vernon ... use this for the internal web camera

   camera.set(CV_CAP_PROP_FRAME_WIDTH, 320);  // David Vernon ... has no effect on my webcam so resizing below
   camera.set(CV_CAP_PROP_FRAME_HEIGHT, 240);

   if( camera.isOpened() ) {
      Mat current_frame;
      do {
         camera >> current_frame;
         if( current_frame.empty() )
            break;

         // vector<Rect> faces; // David Vernon ... moved to start of function
         // Mat gray;           // David Vernon ... moved to start of function

         //resize(current_frame,current_frame,Size(),0.5,0.5); // David Vernon

         cvtColor( current_frame, gray, CV_BGR2GRAY );
         equalizeHist( gray, gray ); // David Vernon: irrespective of the equalization, well illumiated images are required
         cascade.detectMultiScale( gray, faces, 1.1, 2, CV_HAAR_SCALE_IMAGE, Size(30, 30) );

         for( int count = 0; count < (int)faces.size(); count++ )
            rectangle(current_frame, faces[count], cv::Scalar(255,0,0), 2);

         imshow(outputWindowName, current_frame );
         c = waitKey(10);   // This makes the image appear on screen ... DV changed from original
      // } while (c == -1); // David Vernon
      } while (!_kbhit());
   }
}
/* ------------------------------------------------------------------------------------- */
```

```
    do{
        waitKey(30);
    } while (!_kbhit());

    getchar(); // flush the buffer from the keyboard hit

    destroyWindow(outputWindowName);
}
```

Reading

R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.

Section 14.1.1 Face Detection