# REAL-TIME FACE DETECTION AND TRACKING

Presentation by Ramsri Goutham Golla, ASU

# Topics to be covered

1.Viola Jones face detection algorithm

- Haar features
- Integral image
- Adaboost
- Cascading

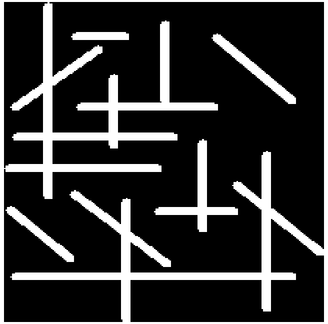
2. Face Tracking algorithm

- Mean shift
- Histogram backprojection
- Continuous Adaptive Mean shift (camshift )

# Viola Jones face detection algorithm

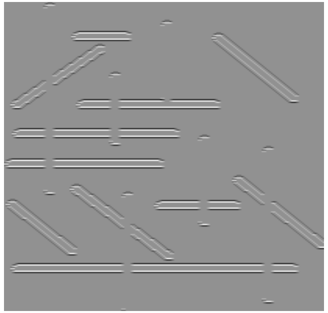
## Haar features | Integral Image | Adaboost | Cascading

### Basic introduction to edge detection



| | | |
|---|---|---|
| −1 | −1 | −1 |
| 2 | 2 | 2 |
| −1 | −1 | −1 |

Convolution Kernel

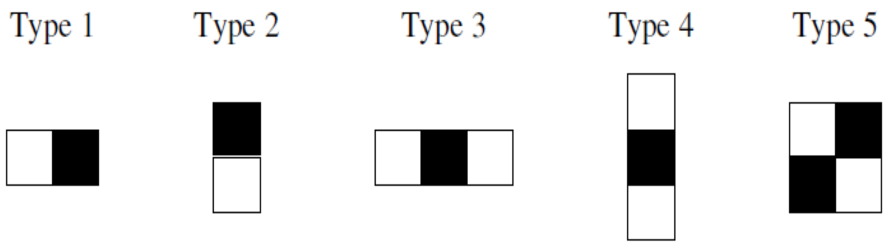


Output image(right) has high intensity at pixels where the convolution kernel pixel pattern matches perfectly with the input image.

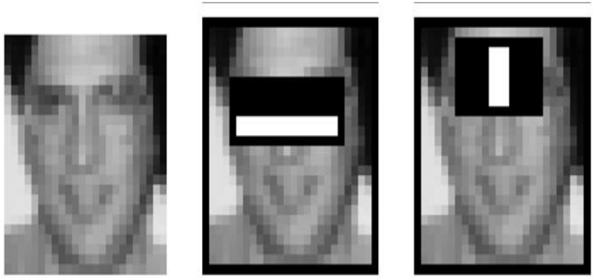
# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- Haar features are similar to these convolution kernels which are used to detect the presence of that feature in the given image.
- Each feature results in a single value which is calculated by subtracting the sum of pixels under white rectangle from the sum of pixels under black rectangle.




Haar features used in viola Jones

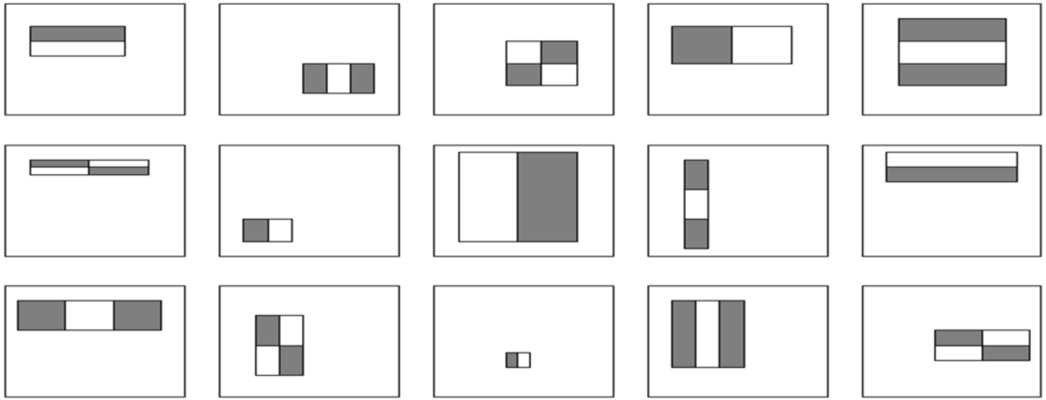


Applying on a given image

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- Viola jones algorithm uses  a 24x24 window as the base window size to start evaluating these features in any given image.
- If we consider all possible parameters of the haar features like position, scale and type we end up calculating about 160,000+ features in this window.



# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- Since it is clear that huge number of these rectangular haar features have to be evaluated each time Viola Jones have come up with a neat technique to reduce the computation rather than summing up all pixel values under the black and white rectangles every time.

- They have introduced the concept of integral image to find the sum of all pixels under a rectangle with just 4 corner values of the integral image.

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- In an integral image the value at pixel (x,y) is the sum of pixels above and to the left of (x,y)

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Input image

Sum above and to left

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 2 | 4 | 6 |
| 3 | 6 | 9 |

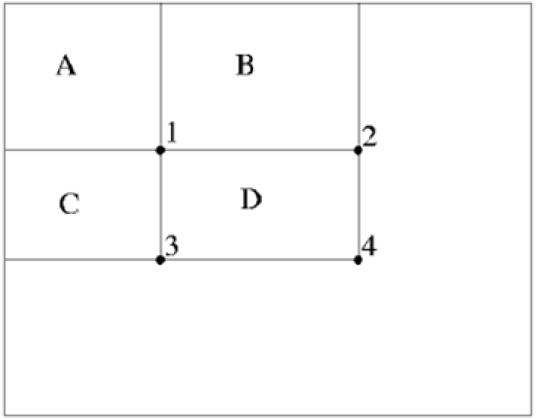
Integral image

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- Integral image allows for the calculation of  sum of all pixels inside any given rectangle using only four values at the corners of the rectangle.

Integral image




Sum of all pixels in

D = 1+4-(2+3)

= A+(A+B+C+D)-(A+C+A+B)

= D

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- As stated previously there can be approximately 160,000 + feature values within a detector at 24x24 base resolution which need to be calculated. But it is to be understood that only few set of features will be useful among all these features to identify a face.

All Features

Relevant feature

Irrelevant feature

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- Adaboost is a machine learning algorithm which helps in finding only the best features among all these 160,000+ features. After these features are found a weighted combination of all these features in used in evaluating and deciding any given window has a face or not. Each of the selected features are considered okay to be included if they can atleast perform better than random guessing (detects more than half the cases).
- These features are also called as weak classifiers. Adaboost constructs a strong classifier as a linear combination of these weak classifiers.

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

Strong classifier

Weak classifier

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

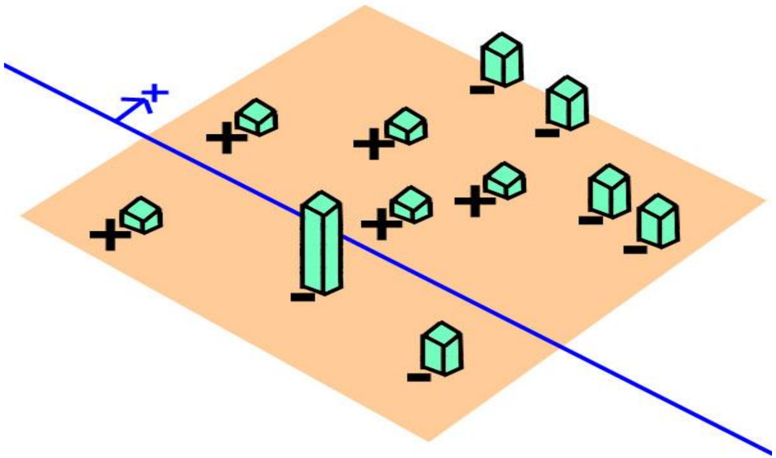
- AdaBoost starts with a uniform distribution of “weights” over training examples.

- Select the classifier with the lowest weighted error (i.e. a “weak” classifier)

- Increase the weights on the training examples that were misclassified.

- (Repeat)

- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.



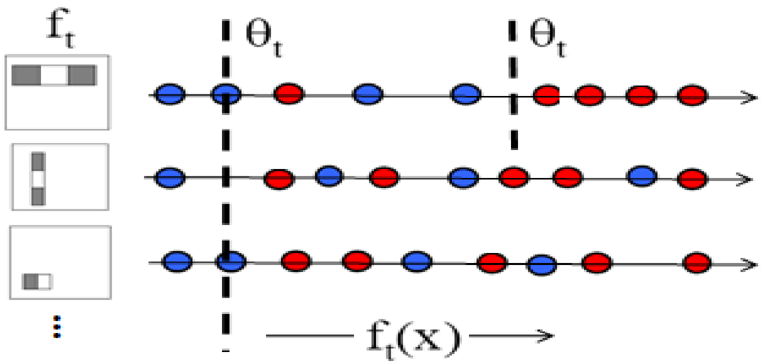
$$h_{\text{strong}}(\mathbf{x}) = \begin{cases} 1 & \alpha_1 h_1(\mathbf{x}) + \dots + \alpha_n h_n(\mathbf{x}) \geq \frac{1}{2}(\alpha_1 + \dots + \alpha_n) \\ 0 & \text{otherwise} \end{cases}$$

***Slide taken from a presentation by Qing Chen, Discover Lab, University of Ottawa***

# Viola Jones face detection algorithm

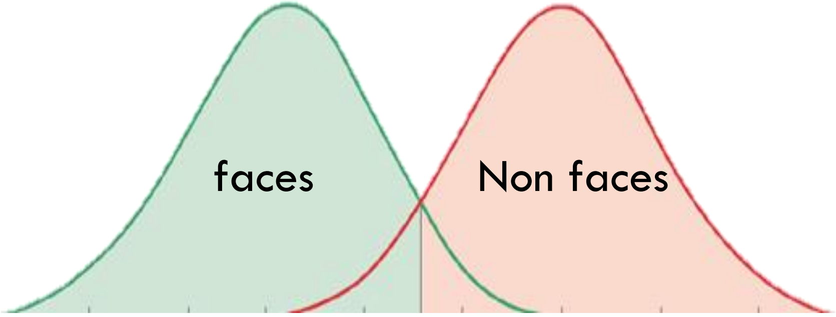
## Haar features | Integral Image | Adaboost | Cascading

- Adaboost finds the single rectangular feature and threshold that best separates the positive (faces) and negative (non faces) training examples, in terms of weighted error.




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

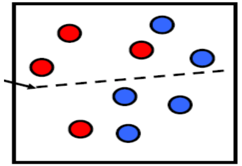
A Gaussian weak classifier used



# 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 $\epsilon_t$.
  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



$\{x_1, \ldots x_n\}$

For T rounds

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

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

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

Slide from K.Grauman

# Viola Jones face detection algorithm

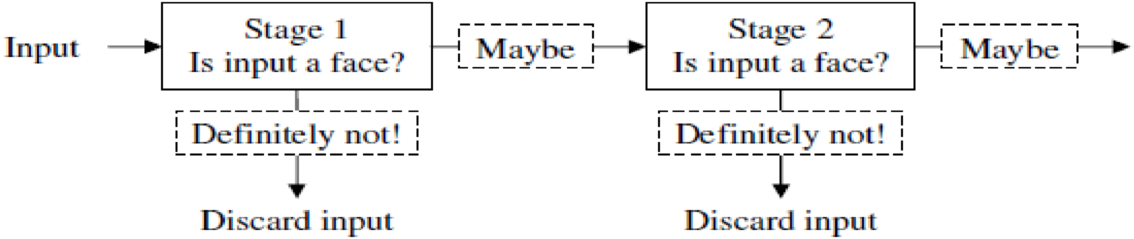
## Haar features | Integral Image | Adaboost | Cascading

- The basic principle of the Viola-Jones face detection algorithm is to scan the detector many times through the same image – each time with a new size.
- Even if an image should contain one or more faces it is obvious that an excessive large amount of the evaluated sub-windows would still be negatives (non-faces).
- So the algorithm should concentrate on discarding non-faces quickly and spend more on time on probable face regions.
- Hence a single strong classifier formed out of linear combination of all best features is not a good to evaluate on each window because of computation cost.

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- Therefore a cascade classifier is used which is composed of stages each containing a strong classifier. So all the features are grouped into several stages where each stage has certain number of features.
- The job of each stage is used to determine whether a given sub window is definitely not a face or may be a face. A given sub window is immediately discarded as not a face if it fails in any of the stage.



# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

**Training a cascade**

- To design a cascade we must choose:
  - Number of stages in cascade (strong classifiers).
  - Number of features of each strong classifier.
  - Threshold of each strong classifier (the $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$ in definition)
- Optimization problem:
  - Can we find optimum combination?

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

**Slide courtesy: Kostantina Palla, University of Edinburgh**

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- Since finding optimum combination is extremely difficult. Viola & Jones suggested a heuristic algorithm for the cascade training.
- Manual Tweaking:
  - select $f_i$ (Maximum Acceptable False Positive rate / stage)
  - select $d_i$ (Minimum Acceptable True Positive rate / stage)
  - select $F_{target}$ (Target Overall False Positive rate)
- Until $F_{target}$ is met:
  - Add new stage:
    - Until $f_i$ , $d_i$ rates are met for this stage
      - Keep adding features & train new strong classifier with AdaBoost.

**Slide courtesy: Kostantina Palla, University of Edinburgh**

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

- 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 $F_i$)

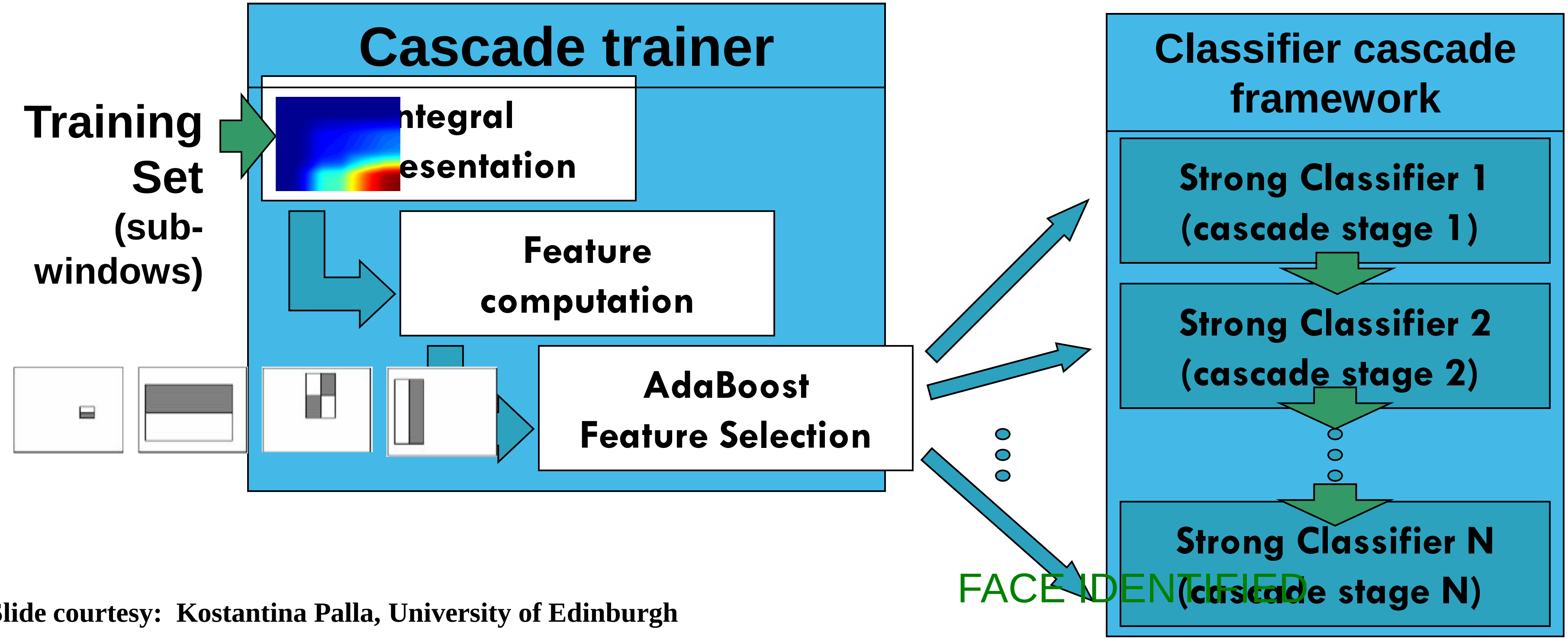
  $N = \varnothing$

  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 $N$.

**Slide courtesy: Kostantina Palla, University of Edinburgh**

# Testing phase

# Training phase

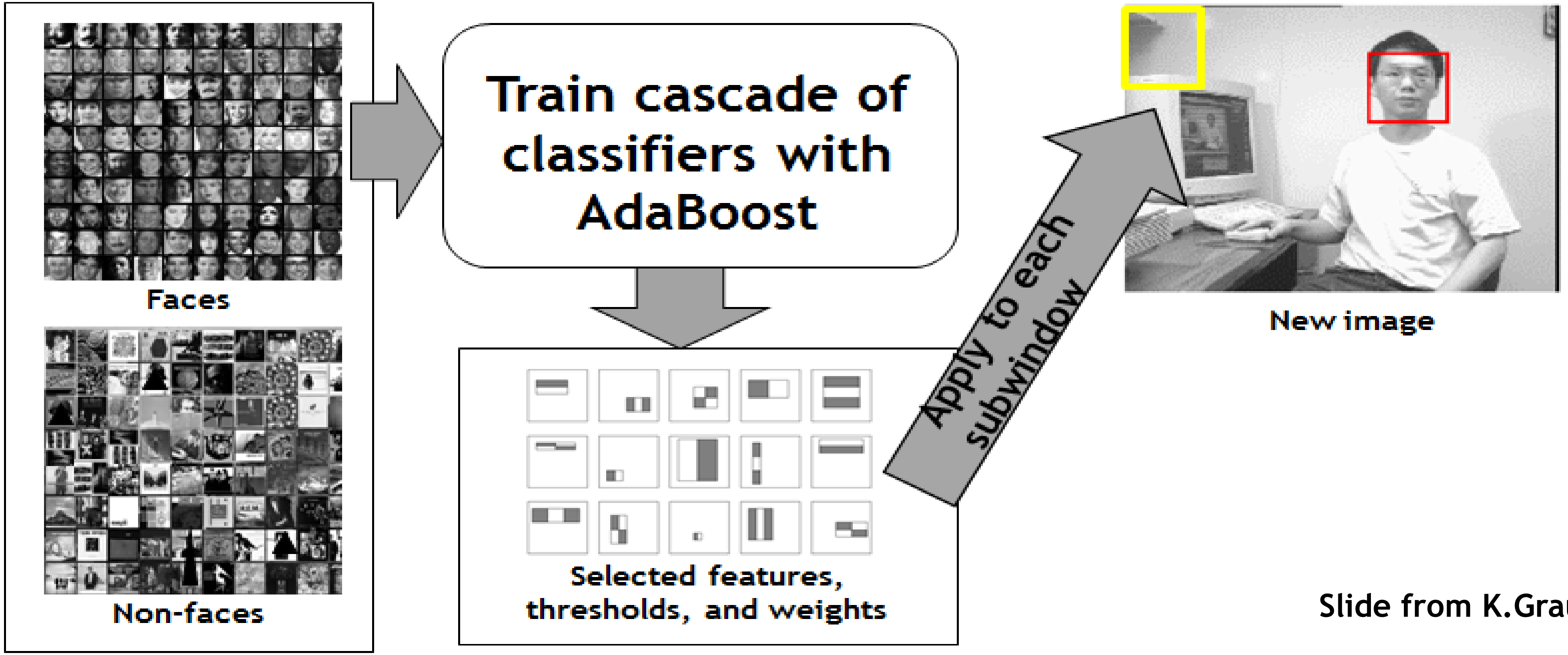



Slide courtesy: Kostantina Palla, University of Edinburgh

# Viola Jones face detection algorithm

## Haar features | Integral Image | Adaboost | Cascading

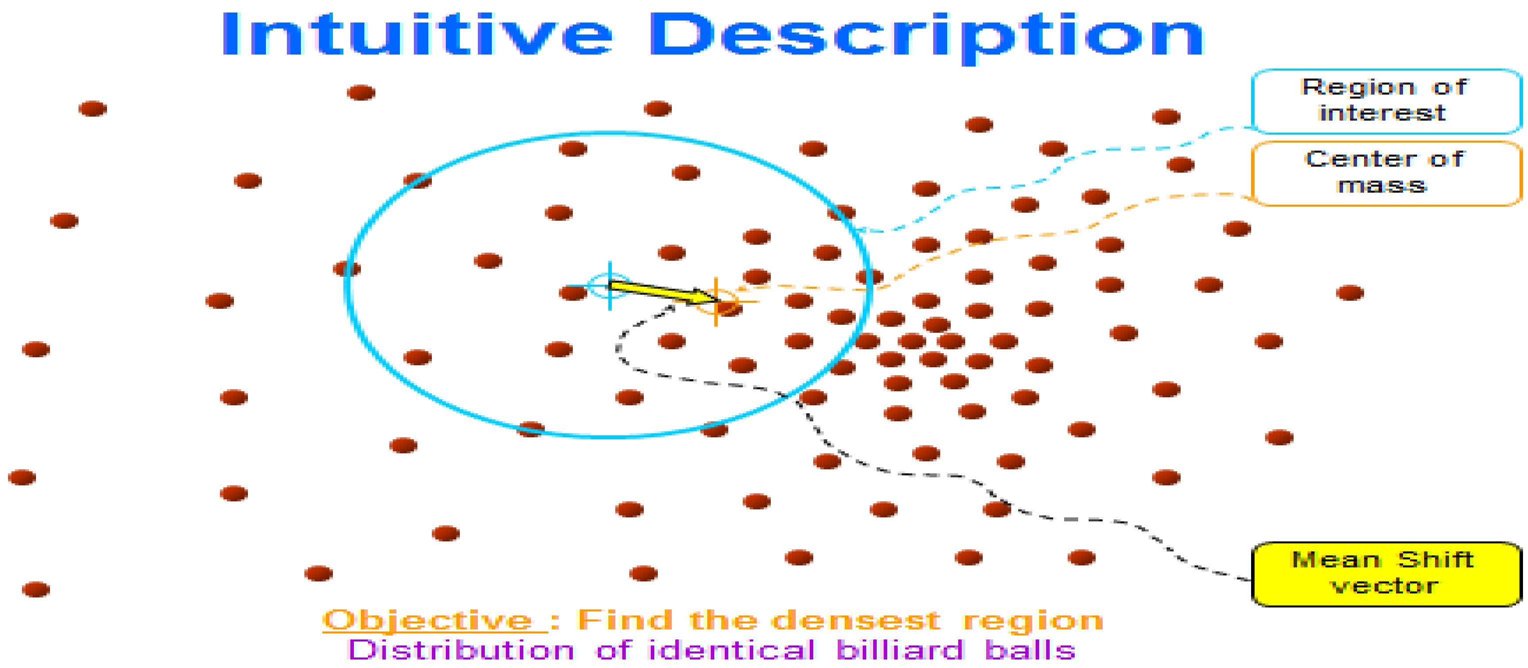
- Summary




Slide from K.Grauman

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift




Slide courtesy : Yaron Ukrainitz & Bernard Sarel

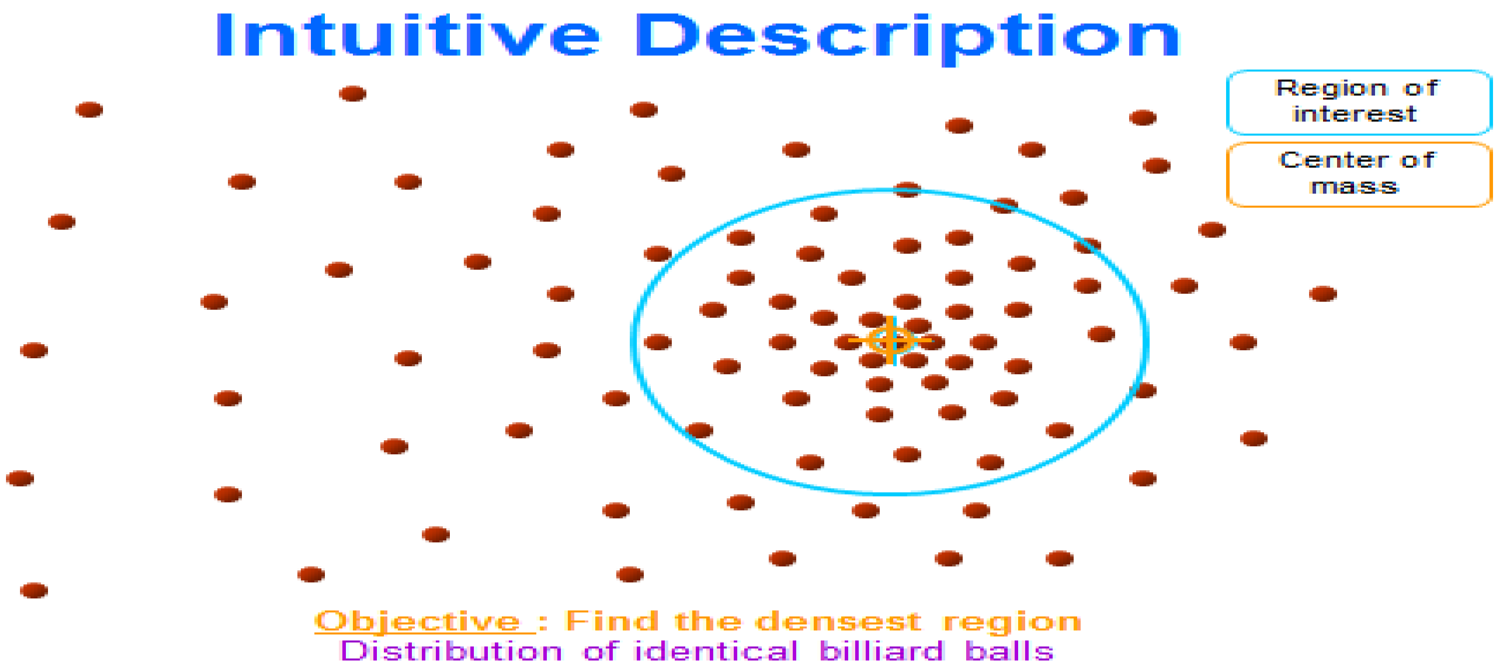
# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift


Slide courtesy : Yaron Ukrainitz & Bernard Sarel

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

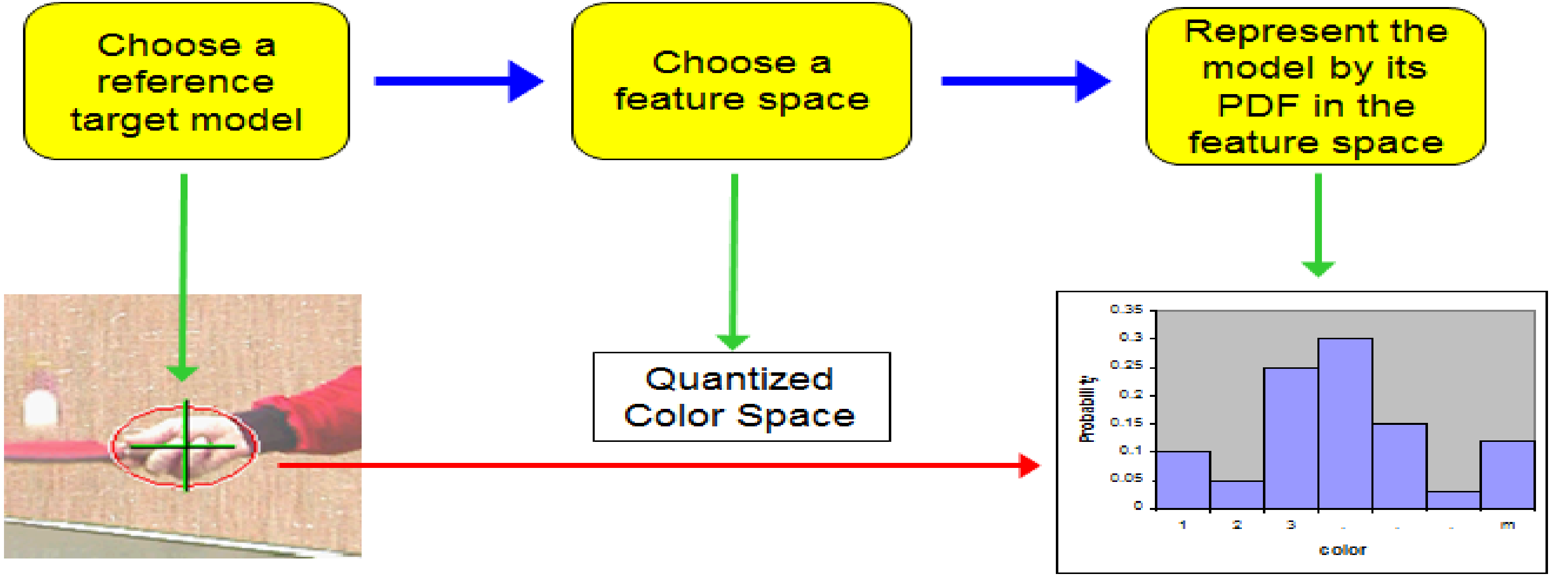



Slide courtesy : Yaron Ukrainitz & Bernard Sarel

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

- Mean shift Tracking

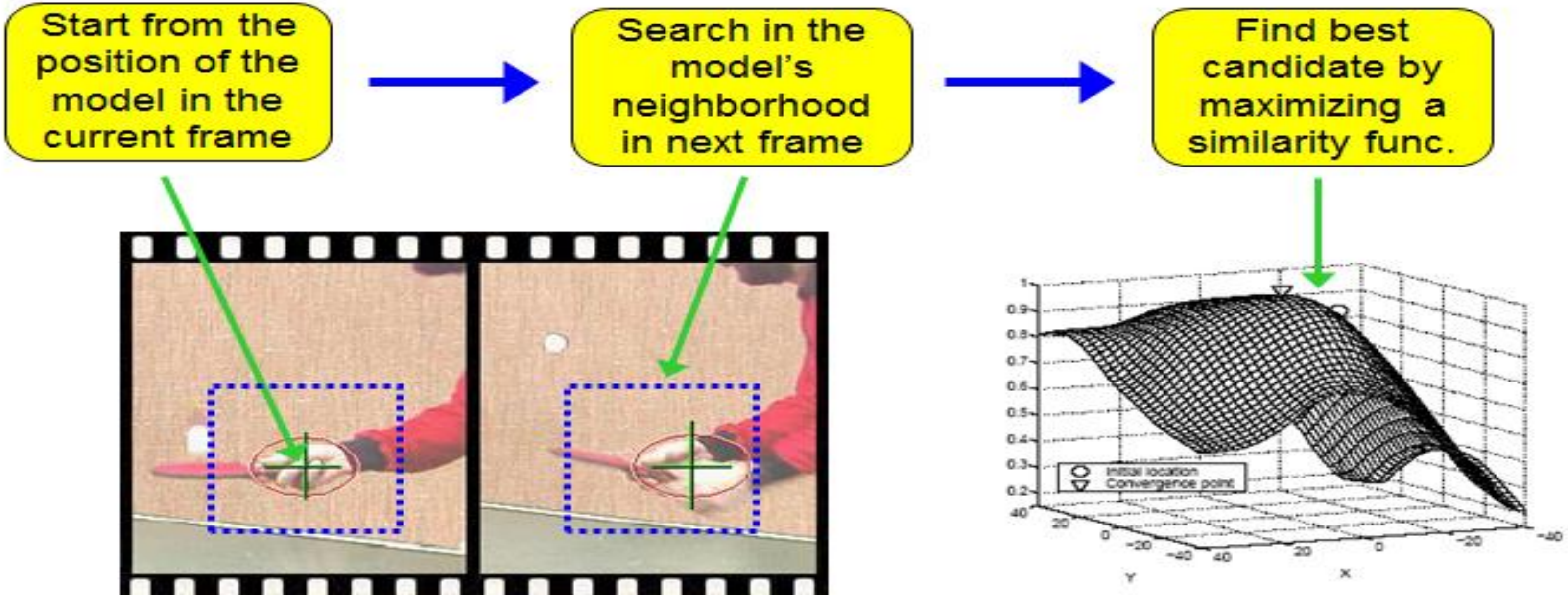



Slide courtesy : Yaron Ukrainitz & Bernard Sarel

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

- Mean shift Tracking

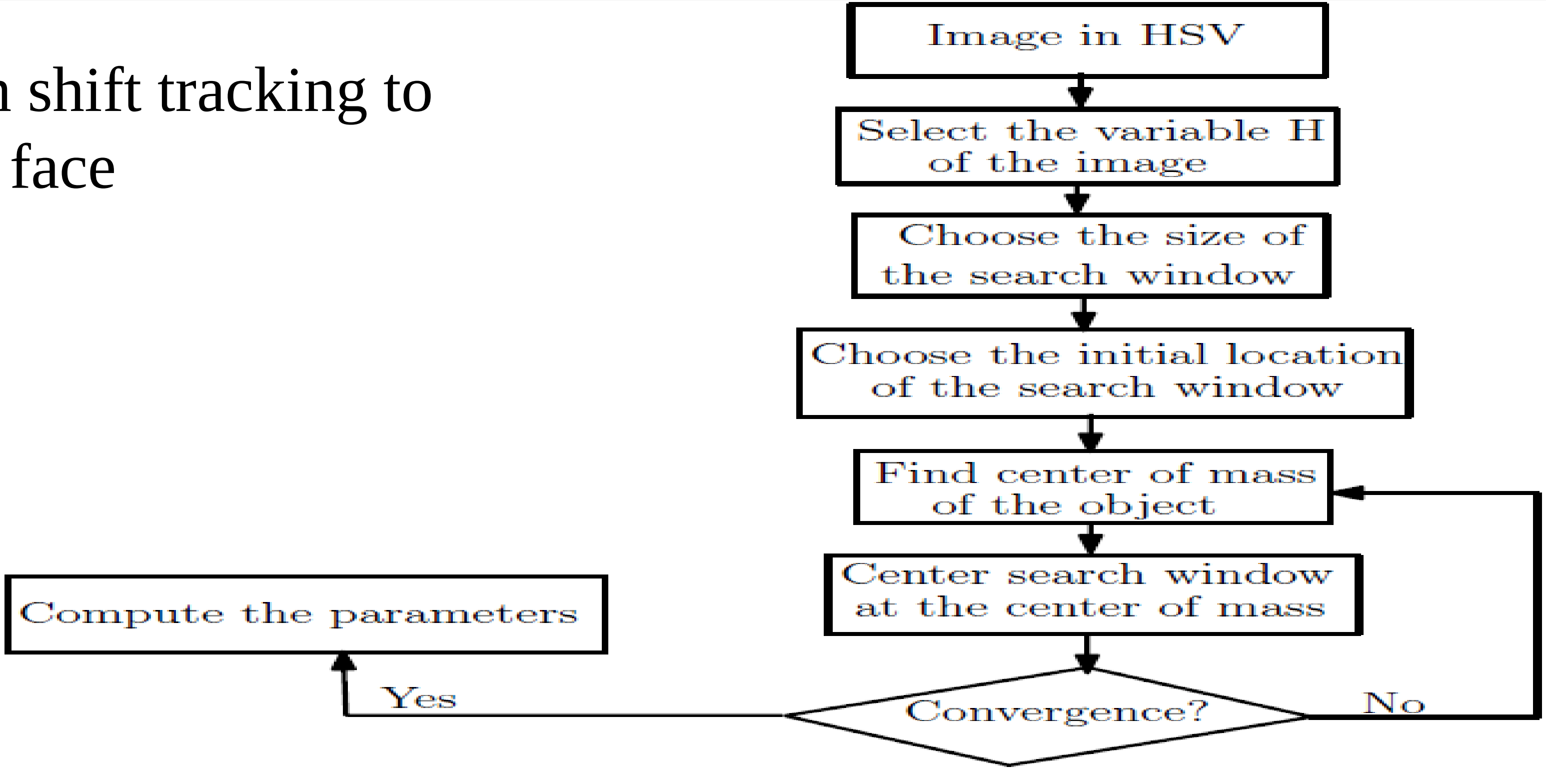



Slide courtesy : Yaron Ukrainitz & Bernard Sarel

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

Mean shift tracking to track face



# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

- In the beginning a window is chosen to track the face. The hue values under the window are extracted and a histogram is formed using those values.
- This histogram is used as a reference and for any new image the pixel value in it are replaced from the value of the reference histogram corresponding to that pixel value. This technique is called histogram back projection. Mean shift is applied on this back projected image.

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

- To apply mean shift the area and the center of mass are calculated under the window using following equations.
- Area (also called zeroth moment): $M00 = \sum_x \sum_y I(x,y)$
- First order moments : $M10 = \sum_x \sum_y x \times I(x,y); M01 = \sum_x \sum_y y \times I(x,y)$
- The center of mass (xc,yc) are then calculated using : $xc = \frac{M10}{M00}; yc = \frac{M01}{M00}$

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

- CAMshift is an algorithm to continuously adapt the window size according to the size of the object that is being tracked.
- In each new iteration the window size is adjusted based on the first moment(area) calculated inside the previous window.
- The size of window is adjusted as :

$$s = 2 * \sqrt{\frac{M_{00}}{256}}.$$

- The window width and length are changed as s and 1.2s.

# Face Tracking Algorithm

## Mean Shift | Histogram Back Projection| CAMshift

- Summary
- Select a target window around object you want to track in an image , choose color space (eg: HSV)and extract histogram of the target window.
- For any new image change the color space and calculate probability map using histogram back projection . Apply meanshift on the window and converge to the centre of mass which represents  moved object centroid.
- Adjust window size using zeroth moment(area) around the object that is tracked. Follow the same for every new image coming.

# Face Detection and Tracking

# Face Detection and Tracking