****

***Short Project***

**Design and implementation of a face detection and location technique to crop faces from human images DB.**

Authors:

Michal Szabó / [michal.szabo@estudiant.upc.edu](mailto:michal.szabo@estudiant.upc.edu)

Aleix Granero Molinedo / [aleix.granero@estudiant.upc.edu](mailto:aleix.granero@estudiant.upc.edu)

Professor:

Joan Aranda Lopez / [joan.aranda@upc.edu](mailto:joan.aranda@upc.edu)

12. december 2021

**ABSTRACT**

**INDEX**

[1. Introduction 4](#_Toc90147713)

[2. Viola and Jones algorithm 4](#_Toc90147714)

[3.1 Haar-like features 5](#_Toc90147715)

[3.1.1 Specification 5](#_Toc90147716)

[3.1.2 Detection 5](#_Toc90147717)

[3.2 Integral image 8](#_Toc90147718)

[3.3 Applying Haar features 9](#_Toc90147719)

# 1. Introduction

# 2. Viola and Jones algorithm

# 3.1 Haar-like features

Haar-like features are [digital image](https://en.wikipedia.org/wiki/Digital_image) [features](https://en.wikipedia.org/wiki/Feature_(computer_vision)) used in [object recognition](https://en.wikipedia.org/wiki/Object_recognition). In the present project, Haar-features will be used to detect faces using the Viola and Jones algorithm.

### 3.1.1 Specification

With a human face it is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore, a common Haar feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. Additionally, a second feature has been selected relying on the property that the eyes are darker than the bridge of the nose.

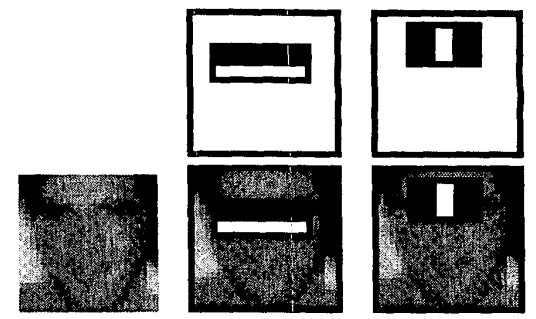


Figure 1 Most common Haar-features in face recognition

### 3.1.2 Detection

Once the Haar-features are defined, we must continue with the computation part, in which we will try to detect them in the images. Logically, it will be practically impossible to find exactly the specified Haar-features, since they are binary and our images are in grayscale, so we will limit ourselves to finding the regions of the images that are most similar.

The detection process consists on applying the Haar-feature in each region of the image and calculate the degree of similarity. This degree of similarity is determined as the difference between the sum of the intensities of the white pixels and the black pixels.

For example, if we want to find the Haar-feature specified in *figure 2* and it is applied in the region of the image specified in *figure 3*, the degree of similarity ∆ can be calculated using the following expression:

Equation 1 Similarity degree

Knowing that in the ideal case this degree will be 255, we can determine that the region that is closest to this number will be the one that has the closest resemblance to the specified Haar-feature.

|  |  |  |  |
| --- | --- | --- | --- |
| 230 | 200 | 50 | 0 |
| 255 | 230 | 0 | 100 |
| 200 | 230 | 100 | 100 |
| 200 | 230 | 50 | 50 |

|  |  |  |  |
| --- | --- | --- | --- |
| 255 | 255 | 0 | 0 |
| 255 | 255 | 0 | 0 |
| 255 | 255 | 0 | 0 |
| 255 | 255 | 0 | 0 |

Figure 3 Image region

Figure 2 Haar-feature specified

In practice we are not going to have only one Haar-feature, so we will search for the region of the image that has the closest resemblance to all the specified Haar-features. This value can be calculated as the product of the ∆ of each of the Haar-features:

Equation 2 Similarity degree for more than one feature

However, if we just specify a Haar-feature with a fixed dimension and look for regions that look like it, our face recognition method is going to get really bad results since the size of faces can change from one image to another. Therefore, it is necessary to change the Haar-feature size and check all the positions in the image for each size. For example, for the previously specified Haar-feature we would follow the following procedure:

Compute all the in each region of 1x4

Resize

Compute all the in each region of 1x2

Initial size

|  |  |  |  |
| --- | --- | --- | --- |
| 255 | 255 | 0 | 0 |

|  |  |
| --- | --- |
| 255 | 0 |

|  |  |
| --- | --- |
| 255 | 0 |
| 255 | 0 |
| 255 | 0 |

Compute all the in each region of 2x2

|  |  |
| --- | --- |
| 255 | 0 |
| 255 | 0 |

Resize

…

Compute all the in each region of 1x6

Resize

Resize

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 255 | 255 | 255 | 0 | 0 | 0 |

Figure 4 Procedure to compute all Δ

From a theoretical point of view this would be the best method to find the region that most closely resembles the specified Haar-feature, but nevertheless in practice this process is totally unfeasible since computing the ∆ of all possible sizes would require a high computational time. For example, if we are working with a 200x200 image and we want to find the previously specified Haar feature, the number of ∆ that would be calculated would be 201 million.

Another important aspect to highlight is that the Haar-features have different minimum sizes, and grow differently, always under the condition that in each region (whether white or black) there must be the same number of pixels. For example, in the following figure we can see how two different Haar-features would grow.

|  |  |
| --- | --- |
| 255 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 255 | 255 | 255 | 0 | 0 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| 255 | 255 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 255 | 255 | 255 | 0 | 0 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 255 | 255 | 0 | 0 |

|  |  |  |
| --- | --- | --- |
| 0 | 255 | 0 |

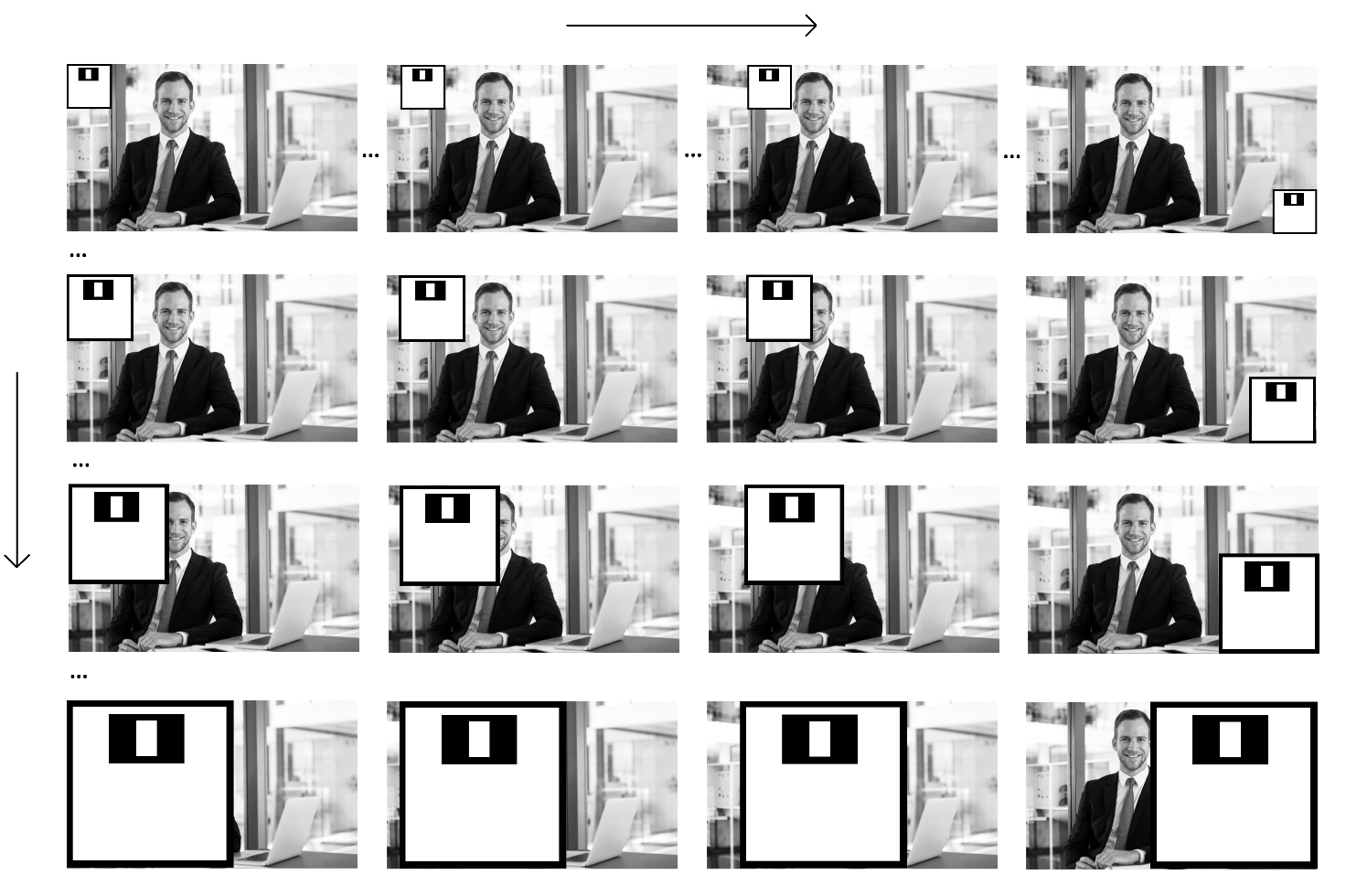
Figure 5 Increase of two different Haar-features

As can be seen, the second Haar-feature grows faster and therefore the number of ∆ to be calculated would be lower. In the case where we want to find a region of the image where these two Haar-features predominate, this event could become a problem, since we would be working with different sizes.

In order to solve this problem, in the current project a window with initial dimensions 24x24 has been created, which contains inside a Haar-feature with initial dimensions 14x8. This initial window is applied to all the positions of the image to compute the ∆ values, and then it is resized by multiplying its dimensions by 1.25. Then all ∆ values for all positions are recalculated, and so on until the window size exceeds the image dimension.

In the following figure these process can be seen.

Figure 6 Initial size of the window



Resize (increments of 0.25)

Changing position (pixel by pixel)

Figure 7 Procedure to compute all possible Δ given a specific Haar-feature

# 3.2 Integral image

Despite the fact that the computation time to compute the ∆ of the Haar-features is significantly reduced by not calculating all the possible sizes (increments of 0.5 are used), performing the calculation of these values ​​still assumes a very high computational time since hundreds and thousands of operations must be performed for each of them. The time complexity of these operations is O (N2).

For example, assuming that the Haar-feature specified in *figure 8* is being used, a total of 16 sums would be performed.

|  |  |  |  |
| --- | --- | --- | --- |
| 255 | 255 | 0 | 0 |
| 255 | 255 | 0 | 0 |
| 255 | 255 | 0 | 0 |
| 255 | 255 | 0 | 0 |

Figure 8 Haar-feature

Logically, calculating this number of operations for each of the ∆ would be totally unfeasible, since as the Haar-feature increases the number of operations also increases, and that is why it is necessary to use a more optimal method: the integral image.

In the integral image, a given pixel is the sum of all the pixels to the left and above (including itself). Given the image in *figure 9*, its integral image has been computed (*figure 10*).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 230 | 200 | 200 | 50 | 0 | 0 |
| 255 | 230 | 230 | 0 | 100 | 100 |
| 200 | 230 | 230 | 100 | 50 | 100 |
| 200 | 230 | 230 | 50 | 50 | 100 |
| 200 | 200 | 230 | 50 | 100 | 50 |
| 200 | 200 | 230 | 50 | 50 | 50 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 230 | 430 | 630 | 680 | 680 | 680 |
| 485 | 915 | 1345 | 1395 | 1495 | 1595 |
| 685 | 1345 | 2005 | 2155 | 2305 | 2505 |
| 885 | 1775 | 2665 | 2865 | 3065 | 3365 |
| 1085 | 2175 | 3295 | 3545 | 3845 | 4195 |
| 1285 | 2575 | 3925 | 4225 | 4575 | 4975 |

Figure 10 Integral image

Figure 9 Image

With the use of the integral image, we significantly reduce the number of operations, since now only 4 will be necessary, regardless of the size of the Haar-feature, and the running time will be O(1). For example, supposing that we want to apply the Haar feature of *figure 8* in *image 9*, the ∆ will be computed as:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 230 | 430 | 630 | 680 | 680 | 680 |
| 485 | 915 | 1345 | 1395 | 1495 | 1595 |
| 685 | 1345 | 2005 | 2155 | 2305 | 2505 |
| 885 | 1775 | 2665 | 2865 | 3065 | 3365 |
| 1085 | 2175 | 3295 | 3545 | 3845 | 4195 |
| 1285 | 2575 | 3925 | 4225 | 4575 | 4975 |

3845 – 680 + 230 – 1085

Figure 11 Δ computation using integral image

# 3.3 Applying Haar features

images = 6;

haars = 2;

count = 0;

window = 24;

itr = 0;

I\_features = {};

for img = 1:1 %images

[row,col] = size(Ig);

for haarSize = 7:0.25:floor(min([row/24,col/24]))-0.25

mask = 24;

itr = itr+1;

window(itr) = floor(haarSize\*mask);

for haar = 1:1 %haars

fprintf(strcat('Info: Calculating feature values for Haar:',int2str(row-window(itr)),'\n'))

fval = zeros(row-window(itr),col-window(itr));

for i = 1:row-window(itr)

for j = 1:col-window(itr)

count = count+1;

%Calculating Feature value for face images

fval(i,j) = calcHaarVal2(II, haar, haarSize, i, j);

end

end

I\_features{img,itr} = fval;

end

end

end