École Centrale Lyon

UE ECL C-4

Bureau d'étude

Rapport

Histograms of Oriented Gradients

Students :

Taïga Gonçalves

Allan Belhabchi

Teachers :

Liming Chen

Jean-Yves Auloge

March 27, 2022

Contents

1 Introduction

Facial recognition, retinal scans and other techniques coming out spies or science fiction movies are quite impressive. But, can it work in reality?

These techniques already exist, not only in high secured facilities in the US but also in more common technologies. Indeed, some social media such as Facebook are able to say who are the people on a picture and where there are.

This Bureau d'Étude aims to study a technique able to compare two different pictures and say how similar they are. To do so, the technique is based on Histograms of Oriented Gradients (HOG), which divides a given picture in a grid and gives the HOGs of each box. Then, comparing the HOG of these boxes allows to conclude on the similarity between the images.

2 Gradient computation

Before computing the histogram, the gradient of the image needs to be calculated. To do so, the following convolution operation will be performed to obtain the x and y derivatives of the image:

$$I_x = I * D_x \quad \text{and} \quad I_y = I * D_y$$

Given $D_x = [-1, 0, 1]$ and $D_y = [-1, 0, 1]^T$. The figure 1 represents the picture which will be used as a test. The figure 2 is the result of the aforementioned operation. This computation have been done in MatLab using the following code:

```
function [G,alpha_] = compute_gradient(I,plot,signed)
%COMPUTE_GRADIENT Computes the magnitude of the gradient and the
%orientation of the image I
if nargin<3
    signed = 'signed'; % Sets the default value of the 'plot' parameter
end

if nargin<2
    plot = 'no plot'; % Sets the default value of the 'plot' parameter
end

Dx = [-1 0 1];
Dy = [-1 0 1]';
Ix = conv2(I,Dx,'same');
Iy = conv2(I,Dy,'same');

G = (Ix.^2+Iy.^2).^0.5;
theta = atan2(Iy,Ix);
```

```

alpha = theta*180/pi;
alpha_ = zeros(size(alpha));
[nrows,ncols] = size(alpha_);
for i=1:nrows*ncols
    if alpha(i) >= 0
        alpha_(i) = alpha(i);
    else
        if strcmp(signed,'signed')
            alpha_(i) = alpha(i)+360;
        else
            alpha_(i) = alpha(i)+180;
        end
    end
end

%% Plot
if strcmp(plot,'plot')
    figure;
    sgtitle("Computation of the gradient of the image I");
    nrows = 2;
    ncols = 2;
    subplot(nrows,ncols,1); imshow(I); title("1. I = Original image");
    subplot(nrows,ncols,2); imshow(G,[]); title("1. G = Magnitude of the gradient");
    subplot(nrows,ncols,3); imshow(Ix,[]); title("1. Ix = X-derivative");
    subplot(nrows,ncols,4); imshow(Iy,[]); title("1. Iy = Y-derivative");
end
end
```

Figure 1: Original image

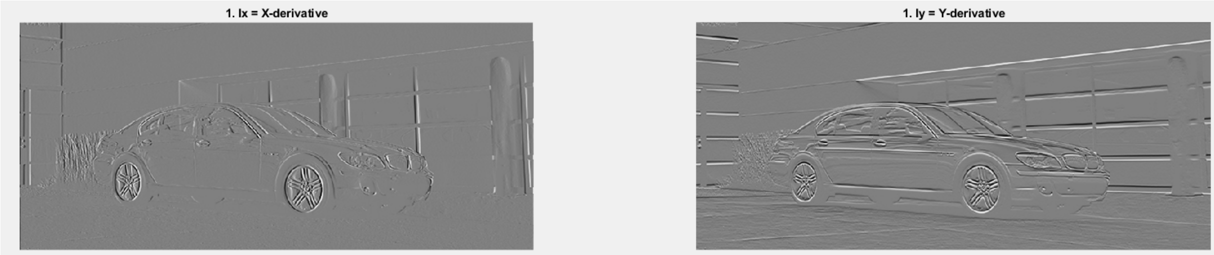

Figure 2: X and Y derivative of the original image

The magnitude of the gradient and its orientation can then be calculated using the following formula:

$$|G| = \sqrt{I_X^2 + I_Y^2} \quad \text{and} \quad \theta = arctan(\frac{I_Y}{I_X})$$

The figure 3 and 4 show the results of these operations. While the 3 have been obtained using the previous code, the 4 used another function (written below) that represents all the pixels having the gradient orientation in the interval [0; 180] with yellow and all gradients in the interval [180; 360] with blue.

```
function [] = plot_orientation(alpha_signed)
% Represents all the pixels having the gradient orientation in the interval
% [0 180] with yellow and all gradients in the interval [180 360] with blue

plot = ones(size(alpha_signed));

[nrows,ncols] = size(alpha_signed);
for i=1:nrows*ncols
    if (0 <= alpha_signed(i) && alpha_signed(i) <= 180)
        plot(i) = 2;
    end
end

figure;
colormap = [0 0 1 ; 1 1 0];
imshow(plot,colormap);

end
```

Figure 3: Magnitude of the gradient

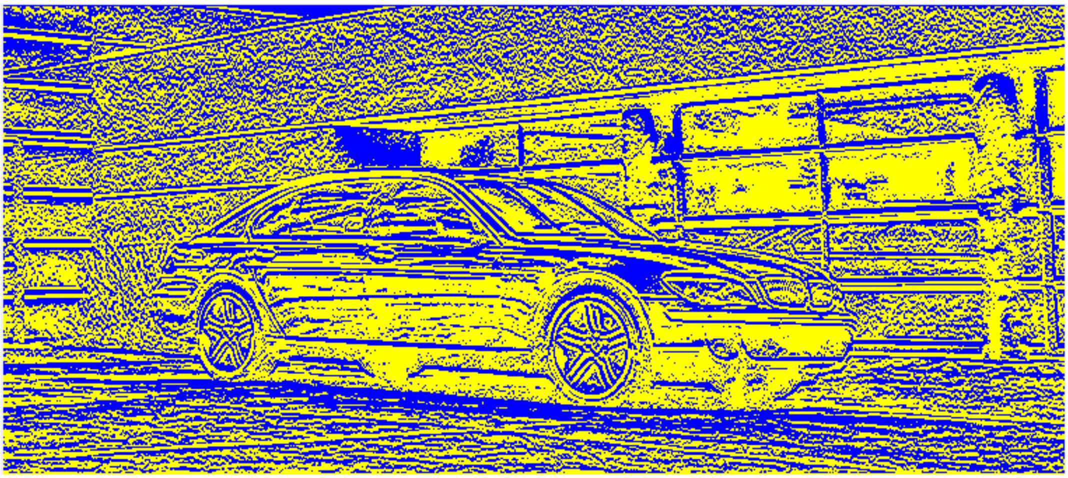

Figure 4: Orientation of the gradient

3 HOG computation

Now that it is possible to compute the gradient of the image. This part will focus on the creation of the HOG.

3.1 Division of the image into cells

To obtain numerous histograms, the original image first needs to be divided into smaller cells. To do so, the following function has been created:

```
function I_divided = divide_cells(I,input,plot)
% Divide the image I into multiple cells
if nargin<3
    plot = 'no plot'; % Sets the default value of the 'plot'
        parameter
end

if nargin<2
```

```
    input = 'dialog'; % Sets the default value of the 'input' parameter
    % This parameter is used for HOG_features.m which already calls a
    % dialog box
end

if strcmp(plot,'plot')
    figure;
    imshow(I);
end

%% Dialog box
if strcmp(input,'dialog')
    prompt = {'Enter cell width:','Enter cell height:'};
    dlgtitle = 'Input';
    dims = [1 35];
    definput = {'20','20'};
    input = inputdlg(prompt,dlgtitle,dims,definput);
    width = str2num(input{1});
    height = str2num(input{2});
else
    width = input{1};
    height = input{2};
end




%% Creation of a cell object containing the divided image
[max_height,max_width] = size(I);
n_rows = ceil(max_height/height);
n_cols = ceil(max_width/width);
I_divided = cell(n_rows,n_cols);
for i=1:n_rows
    for j=1:n_cols
        if i==n_rows && j==n_cols
            I_divided{i,j} = I((i-1)*height+1:max_height , (j-1)*width+1:max_width);
        elseif i==n_rows
            I_divided{i,j} = I((i-1)*height+1:max_height , (j-1)*width+1:j*width);
        elseif j==n_cols
            I_divided{i,j} = I((i-1)*height+1:i*height , (j-1)*width+1:max_width);

        else
```

```
                I_divided{i,j} = I((i-1)*height+1:i*height , (j-1)*
                    width+1:j*width);
            end

        end
    end

%% Plots the result if asked to do so
if strcmp(plot,'plot')
    I_plot = cat(3,I,I,I);
    for i=1:floor(max_height/height)
        I_plot(i*height,:,1)=255;
        I_plot(i*height,:,2)=0;
        I_plot(i*height,:,3)=0;
    end
    for i=1:floor(max_width/width)
        I_plot(:,i*width,1)=255;
        I_plot(:,i*width,2)=0;
        I_plot(:,i*width,3)=0;
    end
    title('Divide I into cells');
    imshow(I_plot);
end
end
```

This code allows the user to choose the height and the width of the cell using a dialog box. The figure 5 has been obtained with a width and height of 20.

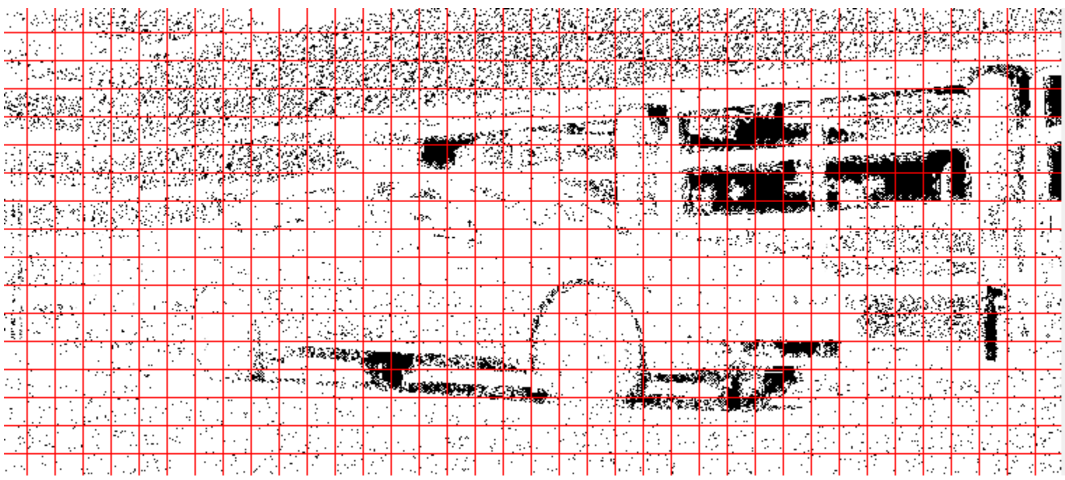

Figure 5: Divison of the image into cells

3.2 Computation of the HOG

The histogram of oriented gradients can now be computed for each cells of the image. The user can choose the number of bins of the histogram in the same way that he chose the height and the width of the cells. The histogram will be computed using the sign gradient with a weighted vote. This vote weight is defined as the magnitude of the gradient. The figure 6 represents the HOG features of the figure 1, obtained by using the following code, using width = height = 30, n_bins=9 as parameters.

```
function [histograms] = HOG_features(I,plot, width, height, n_bins)
% Computes the HOG features for a given image I
if nargin<3
    dialogbox = "dialogbox";
else
    dialogbox = "none";
end

if nargin<2
    plot = 'no plot'; % Sets the default value of the 'plot' parameter
end

%% Dialog box
if strcmp(dialogbox,'dialogbox')
    prompt = {'Enter cell width:','Enter cell height:','Enter the number of bins'};
    dlgtitle = 'Input';
    dims = [1 35];
    definput = {'30','30','9'};

    input = inputdlg(prompt,dlgtitle,dims,definput);
    width = str2num(input{1});
    height = str2num(input{2});
    n_bins = str2num(input{3});
end

%% Divide the image into cells using divide_cells.m
[G, alpha_signed] = compute_gradient(I,'no plot','signed');
G_divided = divide_cells(G,{width,height},'no plot');
alpha_divided = divide_cells(alpha_signed,{width,height},'no plot');
[n_rows, n_cols] = size(G_divided);

%% Histogram computation
edges = 0: 360/n_bins : 360;
histograms = cell(n_rows,n_cols);
for i=1:n_rows
    for j=1:n_cols
        alpha = alpha_divided{i,j};
        hist = zeros(1,n_bins);
        for n=1:n_bins
            if n==n_bins
```

```
                count = (edges(n)<= alpha)&(alpha <= edges(n+1)).*G_divided{i,j}.^0.5;
            else
                count = (edges(n)<= alpha)&(alpha < edges(n+1)).*G_divided{i,j}.^0.5;
            end
            hist(n) = sum(count,'all');
        end
        histograms{i,j} = hist;
    end
end

%% Plot the histograms
if strcmp(plot,'plot')
    [max_height, max_width] = size(I);
    height = max_height/n_rows;
    width = max_width/n_cols;
    f = figure();
    pos = f.Position;
    f.Position = [pos(1) pos(2) max_width, max_height];
    for i=1:n_rows
        for j=1:n_cols
            pos = [width*(j-1)/max_width 1-height*(i)/(max_height) width/(max_width), height/(max_height)];
            subplot('Position',pos);
            histogram('BinEdges',edges,'BinCounts',histograms{i,j});
            ax = gca;
            disableDefaultInteractivity(ax);
            set(gca,'visible','off');
        end
    end
end

end
```

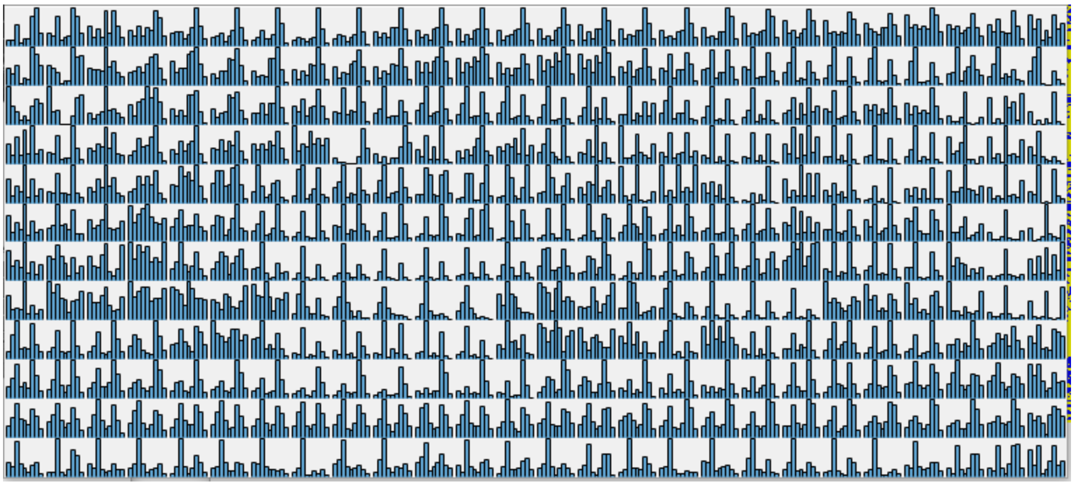

Figure 6: HOG features of the initial image

4 Practical case

Now that a function has been created to compute the HOG features of an image, it will be used to compare two different, but similar images. To do so, the cosine similarity measure will be used:

$$similarity = cos(\theta) = \frac{A \times B}{||A|| \times ||B||} \tag{1}$$

```
function [similarity] = cosine_similarity(object1,object2)
% Computes the degree of similarity between two objects using the cosine
% similarity
object1 = object1(:); % Transforms the matrix into a single column vector
object2 = object2(:); % Same concept

similarity = dot(object1,object2) / (norm(object1)*norm(object2));
end
```

In order to evaluate the efficiency of the kind of technique, the HOG features will be first used to compare two similar images represented in the figure 7, to check if it detects the similarities. Then, it will be used to compare two completely different images represented in figure 8, here, it should not see any similarities.

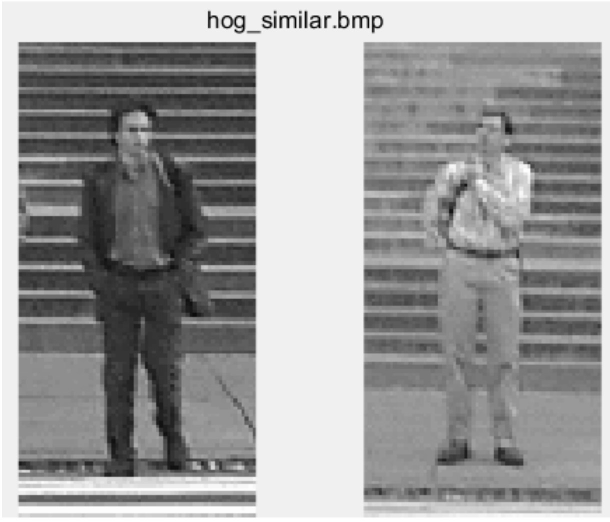

Figure 7: Comparison of two similar images

Figure 8: Comparison of two different images

The similarity has first been calculated without dividing the images into cells and by allocating 12 bins for the histograms. The similar images obtain a similarity of 0.9971 while the different images have only 0.8753.

The same images have then been analyzed after having divided them into square cells of 16, and by allocating 9 bins for the histograms. The similar images obtain a similarity equals to 0.9301 while the different ones have only 0.7150.

These results show that the HOG features can effectively distinguish similar images and different the ones.

5 Conclusion

Comparing two images with HOG seems give satisfying results. Indeed, there is a significant difference between the results given when the pictures are similar and the ones when they are not. Moreover, the algorithms behind the technique can be adapted in order not to be time consuming and thus become useless.

The only remaining point is to define a threshold. Indeed, here when the difference is higher than 0.1, one can say that the pictures are not similar, but how small does the threshold have to be to let them claim that the men on the picture is actually the same men? With a more consequent database, finding the scale of accuracy of the technique should be achievable and it would be possible to use the technique in real situations in real softwares.