École Centrale Lyon

UE ECL C-4

Bureau d'étude

Rapport

Automatic Text Regions Locating in Digital Images

Students :
Taïga Gonçalves
Allan Belhabchi

Teachers :
Liming Chen
Jean-Yves Auloge

March 7, 2022

Contents

1 Introduction

With new technologies and internet boom, there are increasingly more pictures published every days. However, some pictures must be quickly analysed for different reasons. For example, YouTube's policy is to banish violent and hateful speeches characterized by some key words. It is yet impossible for a human being to analyse all these pictures in order to check if they do or do not contain these words. This is why it is important to find a method which allows to automatically analyse the published pictures.

The goal of this work is to study the reproductibility of a patent belonging to *l'École Centrale de Lyon* published by the Doctors Walid Mahdi, Mohsen Ardabilian and the Professor Liming Chen. This patent describes a text locating method in a picture.

Indeed, the functioning of a numerical device can be summed up by a sequence of zeros and ones. However, these sequences can print on a screen a picture which actually means something, even showing a text that the eye can analyse. Thenceforth, it is not obvious that a machine can understand the meaning of what it prints.

2 Load specifications

Before starting the implementation of the former method, it is right to raise several questions about this work. Indeed, the first question which springs to mind is : how is it possible to evaluate the accuracy of this method ? Shall it detect every text in every pictures, even a text that a human's eye cannot ?

Then, even when an answer to this question is given, one can still wonder about the use and the drawbacks of this method. Of course a light example has been described during the introduction but do not seems to be concrete enough.

2.1 Efficiency criteria of the proposed technique

To answer the first question asked higher, one can propose some efficiency criteria, based on different characteristics of the texts encountered in pictures.

2.1.1 Text size

The method should be able to detect, at least, the same elements as a human eye. But this criteria is not useful here because some pictures can be zoomed, so the machine has in its memory an actual text.

Fluently, numerical notice boards are using displays printing letters with dots from rectangles containing 7x9 dots.

By taking this fact in consideration, it seems possible to encode and print a letter by using a rectangle sizing 7x9 pixels. A good method should be able to detect, at least, a text written with letters as small.

2.1.2 Text location

The method must also be able to detect text in every existing location, that is to say :

- Whatever if the text is written vertically, horizontally or in an other direction
- Whatever the background and the color of the text (as long as the text is visible for a human eye, obviously it will be quite hard to detect a white text on a white background)

2.1.3 Character font

Furthermore, it is obvious that all texts are not written with the same character font. An efficient text location method must be independent of the character font.

2.1.4 The specifications in fact

These specifications give only a list of criteria to take into account but do not explain how this work can take them into account.

Based on the criteria developed higher, a data base composed by 62 pictures has been created, gathering pictures containing texts with different sizes, backgrounds, character font...

The idea is to use the method on every pictures of the data base and see if it works. In a perfect scenario, the method shall work on every pictures. But, considering that it needs to enter a few parameters depending of the picture to work, it is possible to say that the goal will be to find the parameter which maximize the number of pictures where the text location is exact.

2.2 Potential drawbacks of the proposed technique

Now one knows how to evaluate the accuracy of a such technique, they have the right to wonder about its use and the potential drawbacks it raises.

Like explained during the introduction, a such technique could allow the machines to sort out the pictures : those which presents a text and those which do not. By locating the text, it is possible to create a second algorithm which can extract the text from the area located by the proposed technique. Then it is possible to analyse the meaning of a text in a picture, but also to digitize several books only by scanning it.

Sorting out and locating the pictures before using the complex algorithms looking for a text in a picture allows to spare time and to act quicker than if they had to analyse every pictures.

3 Method application

This method will now be implemented in Matlab. It is possible to cut the proposed technique in five principal parts:

1. Digital image transformation
2. Enhancement of text region patterns
3. Potential text regions detection
4. Effective text regions selection

5. Delimitation of text region boundaries

3.1 Digital image transformation

This first part aims at creating a standard picture, which can be easily used whatever the original quality of the picture.

To do so, one have to take a picture, called *picture* here (in format .jpg) and read it with Matlab. The read picture will be called *I*.

Then *I* may be colored so the first step is to convert in gray levels called *G*. After that, depending if the text is vertical or horizontal, one have to take the transposition of G, so the method can be implemented only for horizontal texts and this command will transform vertical text in horizontal text. Finally, a third step can be to narrow the space of work of the algorithm by preselecting a region where the text should be (considering the goal of this algorithm and in order to test the accuracy of the method, this step is not a part of this example).

The theory explained higher can be implemented in Matlab with the following code :

```
% Read the image

I = imread(picture.jpg);

% Stage 1: Convert the colored image I into an image in grey
    level G
G =  rgb2gray(I);
[Gh,Gw] = size(G);

% Stage 2: Transpose G to make possible the detection of the
    regions of
% vertical texts
% G = G'

% Stage 3: If possible, define a sub-zone of the image in which
     text
% regions are going to be looked for
```

3.2 Enhancement of text region patterns

The next step is a part of preprocessing by finding the most probable text location. To do so, the goal is to convert the grey image *G* in a binary image called *K*.

First, the image will be converted in an image of a lower resolution. In fact, this will allow to highlight the text since it would appear in the shape of a full line. This process should be done by choosing the nearest neighbors method with a coefficient of M set as 0.131 (which means that the original image will become M times smaller).

Now that text lines are more visible, the image will be transformed into a binary image *K*. The binary image should differentiate the background in black and potential text lines in white. Based on this fact, it is possible to define a threshold (here equal to 0.7) and compare each pixel with its nearest neighbors to see if there is a huge gap between them (higher than 0.7). If there is a gap, it is possible that this pixel is part of a text line, so the binarized image *K* will let this pixel completely white, else the pixel will become black.

The Matlab encoding used to realize this step is detailed below :

```
K = G;

% Multi-resolution method
M = 0.131; % Default value: 0.125 0.145
K = imresize(K, M, 'nearest');
K_resize = K; % Stores the image to plot it later

% Conversion of G into a binary image
threshold = 0.7; % Default value: 0.7 0.6
K = im2bw(K,threshold);
K_BW = K; % Stores the image to plot it later
```

3.3 Potential text regions localization

This step consists in creating a block around the text found. Indeed, the former step only gave a few pixels which might correspond to a text but did not give an area. The goal of this step is to merge pixels representing the same text line using a mask. While the application of the masks on the picture generate big changes, the idea is to continue applying the mask.

```
[Kh,Kw]=size(K);
Kbis = im2bw(ones(Kh,Kw));
while norm(double(K-Kbis))> 0.01 % While the morphological mask does effecive changes to the image
    Kbis = K;
    K=M1(K);
    K=M3(K);
    K=M2(K);
end
K_M1_M2_M3 = K; % Stores the image to plot it later
```

This code presents three functions *M1*, *M2* and *M3*, which are in another file representing three different masks :

- The first mask, set all pixels to 1 when the border pixels on the left and on the right are valued by 1
- The second mask leads to a diagonal closure when two border pixels on the diagonal are valued by 1
- The third mask aims as well at a diagonal closure but with two pixels that touch.

```
function [J] = M1(I)
% First morphological mask
% We set all pixels to 1 when the border pixels on the left and on the
% right are valued by 1
[H,W]=size(I);
for x=1:H % The image is scanned line by line
    for y_left=1:W
        for y_right=y_left:W
            if (I(x,y_left)==1 && I(x,y_right)==1) % Checks if the border pixels are valued by 1
                for y=y_left:y_right % If that so, set all pixels between y_left and y_right to 1
                    I(x,y)=1;
                end
            end
        end
    end
end
J=I;
end
```

```
function [J] = M2(I)
% Second morphological mask
% Diagonal closure when two border piwels on the diagonal are valued by 1
[H,W]=size(I);
for x = 1:H-1 % The image is scanned line by line
    for y_top=1:W
        for y_bot=1:W
            if (I(x,y_top)==1 && I(x+1,y_bot)==1) % Checks if the two borders pixels on the diagonal are valued by 1
                y_left = min(y_top,y_bot);
                y_right = max(y_top,y_bot);
                for y=y_left:y_right % If that so, set all pixels between them to 1
                    I(x,y)=1;
                    I(x+1,y)=1;
                end
            end
        end
    end
end
J=I;

end
```

```
function [J] = M3(I)
% Third morphological mask
% Diagonal closure when two border pixels on the diagonal are
    valued by 1

J = bwmorph(I,'diag'); % The code already exists using bwmorph

end
```

3.4 Effective text regions selection

The image K obtained in the previous section highlighted some regions which represents text lines, but also regions without. This section thus aims at removing the latter. To do so, the grey image G will be transformed in a different grey image called H. Since text pixels should have a different color from the background, it should be possible to enhance this gap. To do so, considering an original image with a grey level $a \in [0, L]$, the new image will have a grey level v such that:

$$v = \begin{cases} a, & a \leq u \\ L, & otherwise \end{cases}$$

The value of u is chosen so that 2% of the pixels in the image will have a grey level of L. This percentage can be modified using the parameter *u_threshold* in the following code:

```
L = 255; % Represents the white colour
u = 255;
nb = sum(G(:)==u); % Number of pixels having the value 255
nb_tot = Gh*Gw;
u_threshold = 0.02;
while(nb/nb_tot < u_threshold && u>0)
    u = u - 1;
    nb = nb + sum(G(:)==u);
end
H = highlight_char(G,u,L,'a');
```

```
function [J] = highlight_char(I,u,L,fmin)
% Highlight character pixels from the background
% It consist of mapping each gray level a=0:L into a gray level
    v=u:L according to the transformation v=f(a). This function
    transforms the grey level of every pixels having a>u by L,
    otherwise the grey level value becomes equal to fmin (fmin =
    a or u).
[H,W]=size(I);
for x=1:H % The image is scanned line by line
    for y=1:W
        if I(x,y) > u
            I(x,y) = L;
            else
                if fmin=="u"
                    I(x,y) = u;
                end
            end
        end
end
J = I;
end
```

Once the image H has been obtained, each potential text regions detected in K can be analysed. To do so, K will be first transformed in a list *region_coor* which contains the coordinate of each potential text regions. Then an histogram will be created for each of these regions in order to locate the positions P1 and P2 of the two most important peaks (local maximum) as shown in figure 1. It will be then considered that effective text regions have a distance $D(P1, P2) = abs(P2 - P1)$ greater than a specific threshold. This threshold is equal to 35% of the total bin number in the gray scale levels. The value of 0.35 can be modified using the parameter *D_threshold* in the code.

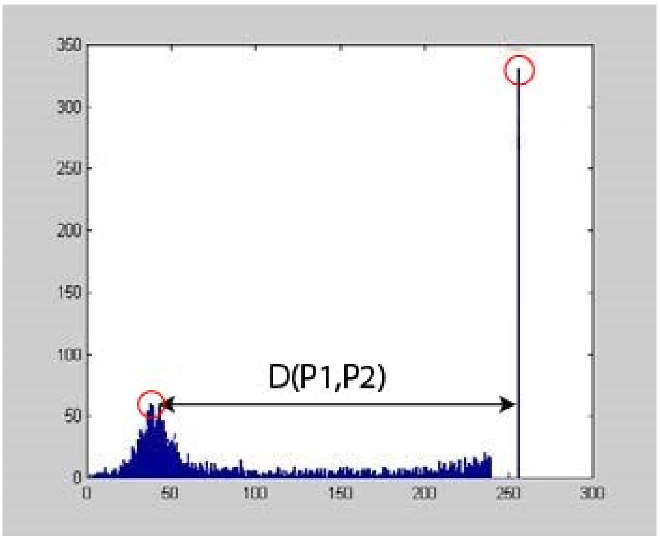

Figure 1: Histogram of one potential text region

The following code represents the implementation of the previously described method:

```
regions_labels = bwlabel(imresize(K,size(G)),8); % Creates
   labels to distinguish each regions
n_regions = max(regions_labels(:)); % Total number of regions
regions_coor = zeros(n_regions,4); % Coordinates of the regions
    (the regions are supposed to be rectangular)
for i=1:n_regions
    [rows,cols] = find(regions_labels==i);
    rtop = min(rows);
    rbot = max(rows);
    cleft = min(cols);
    cright = max(cols);
    new_region = [rtop rbot cleft cright];
    regions_coor(i,:) = new_region;
end

regions = [];
regions_labels_2 = regions_labels;
for i=1:size(regions_coor,1)
    rect = regions_coor(i,:);
    extract_H = H(rect(1):rect(2),rect(3):rect(4));
    count = imhist(extract_H);
    [max1,P1] = max(count); % Highest value in the histogram of the region
    count = count([1:P1-1 P1+1:end]); % Remove the maximum value
    [max2,P2] = max(count); % Second highest value in the histogram of the region

    D_P1_P2 = abs(P2-P1);
    tot_bin_nb = sum(imhist(H)>0); % Total bin number in the gray scale levels
    D_threshold = 0.35; % Default value: 0.15 0.6
    if (D_P1_P2 > D_threshold*tot_bin_nb)
        regions = [regions ; rect]; % The region has an effective text
    else
        regions_labels_2(regions_labels_2==i)=0; % Deletes the region because it doesn't have any texts
    end
end
```

3.5 Delimitation of text region boundaries

Now that the effective text regions have been located and stored in the list *regions*, the text can be delimited by white rectangles in the original image *I*, using the following code:

```
imresult = I;
for i=1:size(regions,1)
    Rh = representative_line(H,regions(i,:),L);
    [x_top,x_bot] = h_delim(H,Rh,L);
    [y_left,y_right] = v_delim(H,Rh,L);
    for x=x_top:x_bot
        imresult(x,y_left,:)=[255 255 255];
        imresult(x,y_right,:)=[255 255 255];
    end
    for y=y_left:y_right
```

```
            imresult(x_top,y,:)=[255 255 255];
            imresult(x_bot,y,:)=[255 255 255];
        end
end
```

3.6 Improving the code

To improve the algorithm, two others morphological masks have been implemented. The purpose of these masks is to remove potential text regions which do not contain any text for sure. Their functioning are described in the figures 2 and 3.

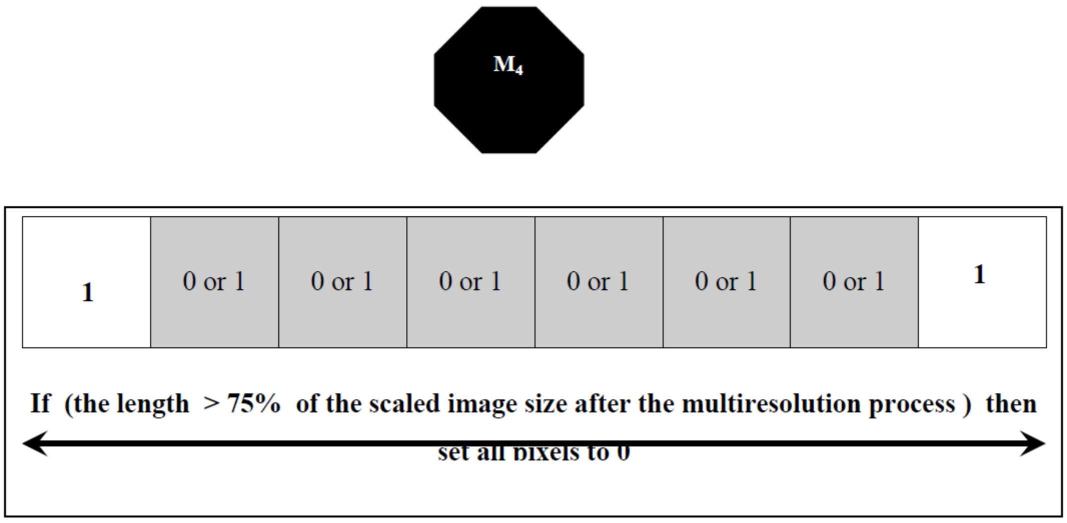
M_4
1
0 or 1
0 or 1
0 or 1
0 or 1
0 or 1
0 or 1
1
If (the length > 75% of the scaled image size after the multiresolution process) then set all pixels to 0

Figure 2: Fourth morphological mask

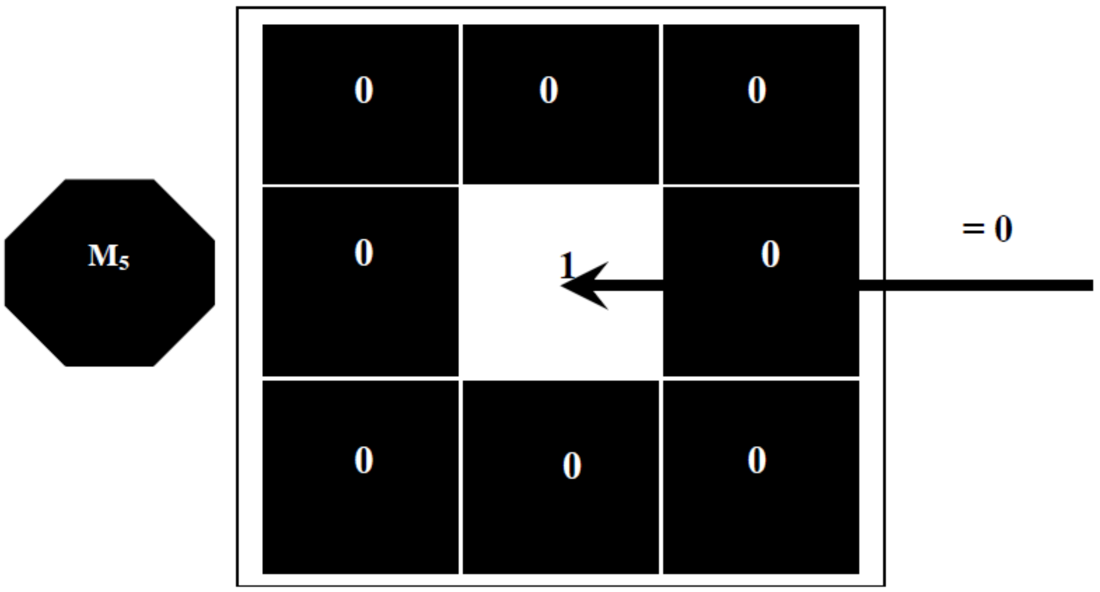

Figure 3: Fifth morphological mask

These two morphological masks are used in the image *K* before the other masks. The following code represents the algorithm of these two masks:

```
function [J] = M4(I)
% Fourth morphological mask
% We set all pixels to 1 when the border pixels on the left and
    on the
% right are valued by 1
[H,W]=size(I);
for x=1:H % The image is scanned line by line
    for y_left=1:ceil(W*0.25)
        for y_right=floor(W*0.75+y_left):W % Reduces the number
            of iteration because the length of the segment
           needs to be greater than .75*W
            if (I(x,y_left)==1 && I(x,y_right)==1) % Checks if
               the border pixels are valued by 1
                for y=y_left:y_right % If that so, set all
                   pixels between y_left and y_right to 1
                    I(x,y)=0;
                end
            end
        end
    end
end
J=I;
end
```

```
function [J] = M5(I)
% Fifth morphological mask
I = imcomplement(I);
I = bwmorph(I,'fill'); % The mask already exists in matlab
J = imcomplement(I);
end
```

4 Results

The patent shows some experimental results of the proposed method applied to video images with complex background. These results are shown on the figure 4.

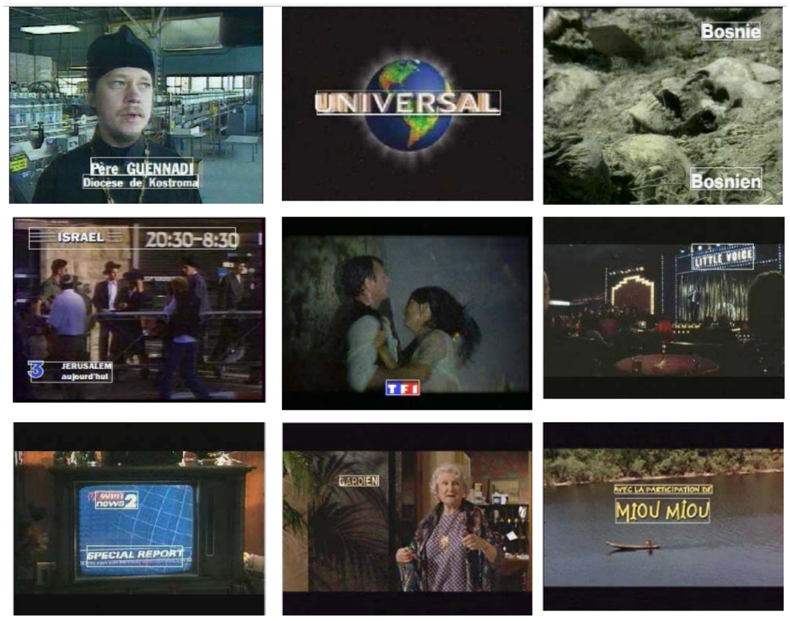

Figure 4: Experimental results of the proposed method applied to video images with complex background extracted from the patent

The text in all these pictures is framed without any mistake. But, the parameters used to frame them depends of the picture and are not universal. This is why it can be useful to use this technique on all of the pictures but with the same parameters in order to check the accuracy of the technique.

4.1 Pictures of the patent

After trying different parameters, it has been concluded that the optimum[1] parameters are, conserving the same names than in the code higher :

- M = 0.131
- threshold = 0.7
- u_threshold = 0.02
- D_threshold = 0.35

[1]Parameters which maximize the locating of the text, to the detriment of the creation of wrong frames

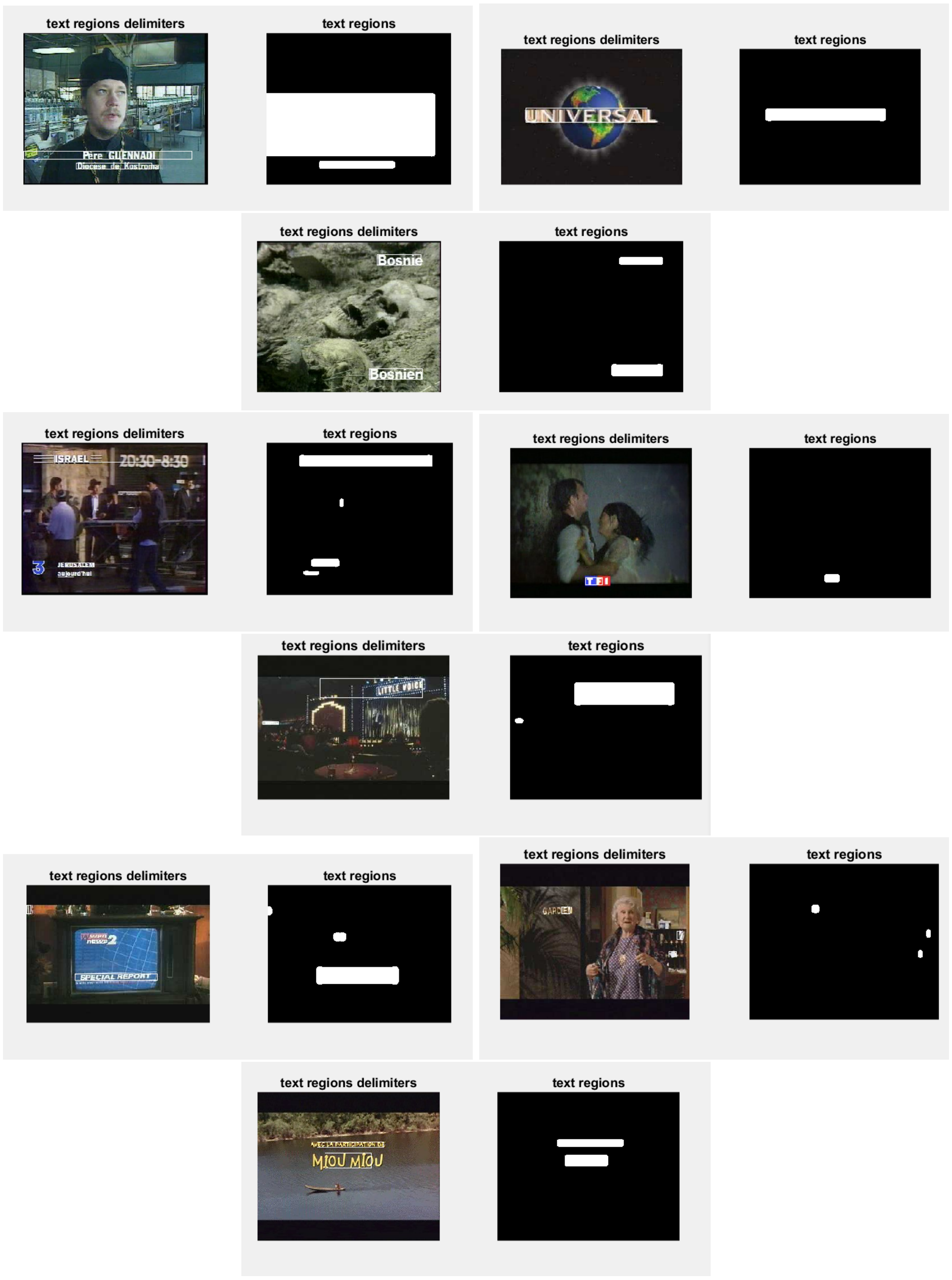

Figure 5: Experimental results of the proposed method applied to video images used in the patent

The figure 5 shows that the technique is not as accurate as the paper presents it (if the parameters cannot be changed for each picture). In the figure 4, the text is framed with a greater precision and without any mistakes, whereas in the figure 5, some parts of the pictures are framed while there aren't any text inside it.

The accuracy of the technique is not as perfect as one could want to analyse precisely a picture. But, it can downsize the search area for a more accurate algorithm.

4.2 Other pictures

In order to evaluate the accuracy of a proposed technique, it would not be correct to try it only on the pictures given in the paper. This is why in this part, it is proposed to analyse the efficiency of the technique on different kinds of pictures.

4.2.1 Pictures without any text

In every pictures presented before, the goal was to locate text regions in the picture, but it is also possible to use the technique backwards by identifying the pictures where there is no text.

This is why the technique have been used on 6 different pictures which do not present any text, represented on the figure 6

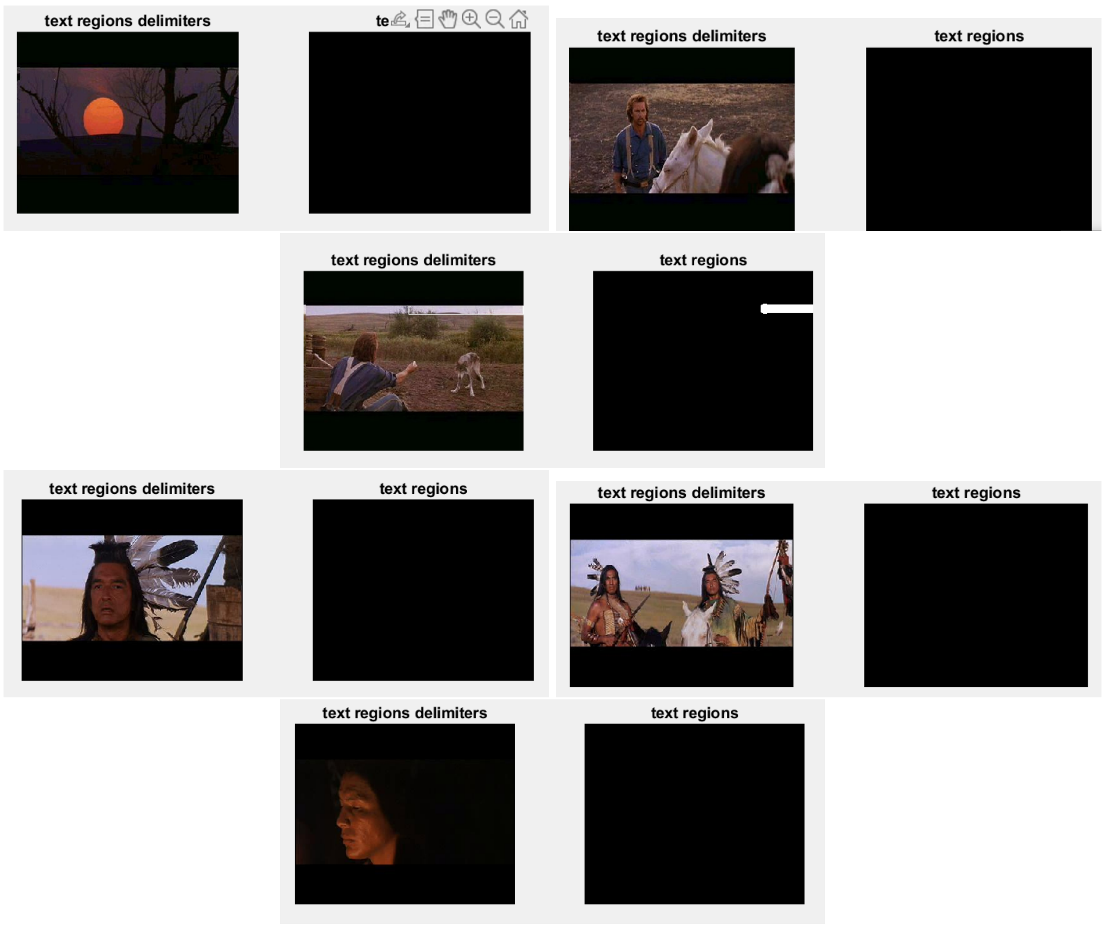

Figure 6: Pictures without text

One picture out of six has been considered like if it was presenting a text. Moreover, this picture is tricky because it presents a text like strip which can be confusing considering how the technique works.

After seeing such results, it seems possible to use this technique to sort out the pictures presenting a text. However, this can be possible only if the algorithm can detect all the pictures which contain at least one text region.

4.2.2 Exceptions

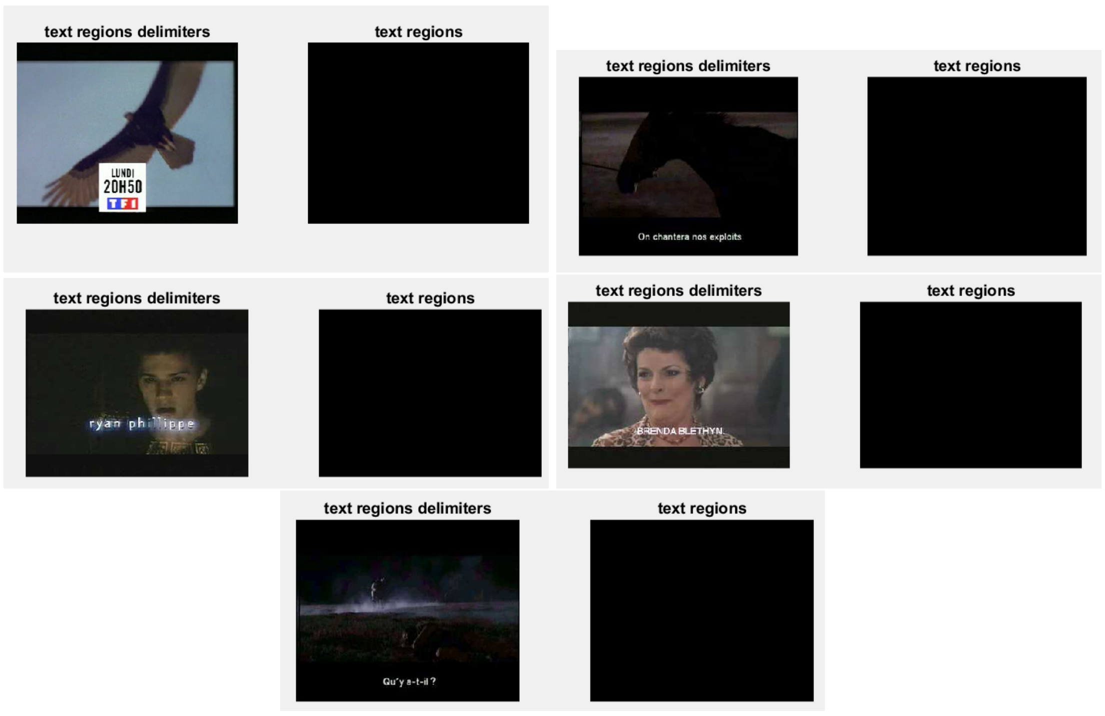

Figure 7: Exceptions

Unfortunately, figure 7 shows that some pictures might present a text but are still be considered as pictures without text.

Here, it seems important to notice that in all these pictures, the text is written in a special way : quite small, with an original character font, and a color like the background.

These exceptions shows that the technique do not satisfy the criteria stated during part 2. Some particular texts will not be identified with this technique.

4.2.3 Incomplete answers

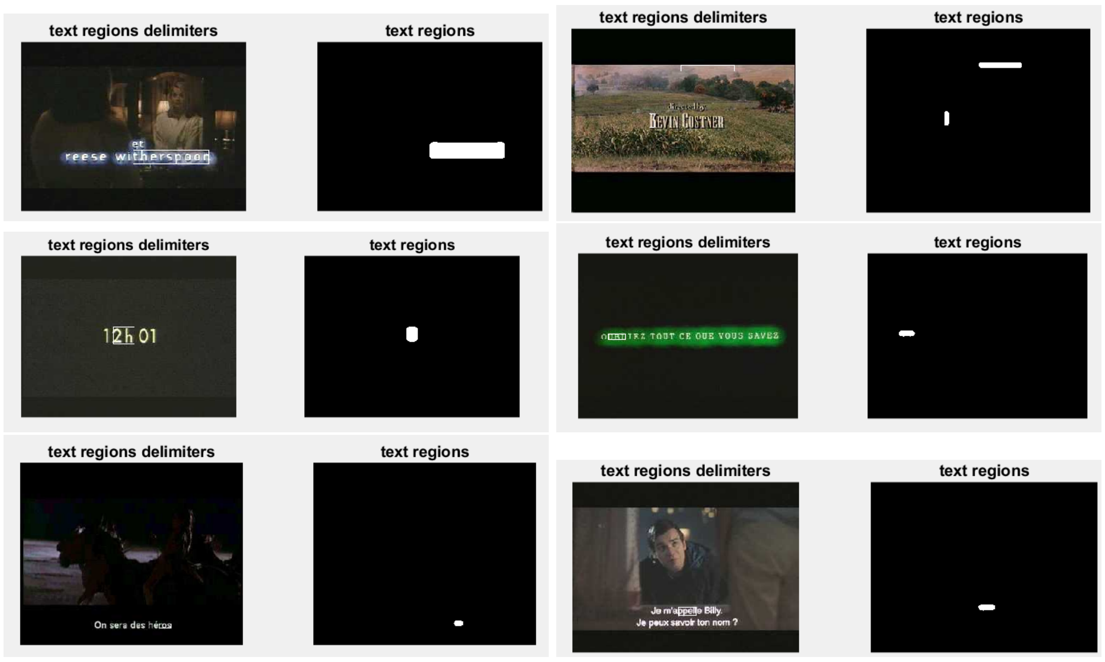

Figure 8: Incomplete answer

Finally, it can be noticed that, even when the technique locates a text in a picture, the framed area is not always accurate. Indeed, figure 8 shows that some parts of sentences are cut in half and not considered as text.

This shows that this technique cannot be self-sufficient and needs another technique to correct its mistakes.

4.2.4 A bright note

For now, only the mistakes committed by the technique have been pointed out, but it seems important to note that about a quarter of the pictures of the data base have been correctly analysed. Even thought this technique is not self sufficient, it can be a useful tool to quickly sort out a big number of pictures.

5 Conclusion

In this document, a specific technique has been implemented to locate text regions. After considering that a good technique shall be able to locate text whatever its character font, background or length as long as it is possible to read it, the technique has been applied on a data base.

The results are not as one can expect in a load specifications : many mistakes are made. However, it allows to scan many pictures in a few time. So even if it is not as accurate as one might want for their text locating technique, this one presents also some advantages.

Finally, it could be interesting to see how to couple this technique with another one, perhaps less quick, but more accurate, in order to optimize the ratio : accuracy–time of execution.