ÉCOLE CENTRALE LYON

Capteurs et Traitement d'Images

Rapport du BE 2

Localisation Automatique des Régions de Texte aux Images Numériques

Auteurs :
M. Julio CABALLERO
M. Matheus MACHADO

Encadrants :
M. Liming CHEN
M. Jean-Yves AULOGE

Version du
8 mars 2022

Table des matières

1 Introduction

L'objectif de ce BE est d'appliquer une technique permettant de reconnaître les zones d'une image où apparaît du texte. Ce texte peut être naturel (c'est-à-dire qu'il apparaît implicitement dans un élément de l'image) ou artificiel (qui a été inclus après coup, comme dans le cas des sous-titres). Normalement, les textes qui apparaissent dans les images ont les caractéristiques suivantes :

— Ils se trouvent généralement à l'avant de l'image et sont différenciés de l'arrière-plan afin d'être facilement lisibles.
— Ils sont généralement monochromes (sans changement de couleur).
— Leur taille est généralement limitée, de sorte qu'ils ne sont jamais assez grands pour remplir tout l'écran, mais restent détectables par l'œil humain (nombre minimum de pixels).

Il convient de noter qu'il est préférable de détecter du texte là où il n'y en a pas, plutôt que le contraire. C'est pourquoi un algorithme très sensible sera utilisé, qui détectera toutes les régions de texte, et occasionnellement une région qui ne contient pas de lettres.

La procédure à suivre se déroulera en 4 étapes, dont seront dites tout de suite, et expliquées dans la section suivante :

— Transformation de l'image numérique (la convertir en noir et blanc).
— Amélioration des patrons des régions avec des textes (multirésolution et binarisation).
— Localisation de régions de texte potentielles.
— Sélection de régions de texte efficaces.

Même si on a plusieurs images à analyser, on expliquera la procédure à suivre en apliquant cette technique à une des images, l'image numéro 5 (voir Figure 3).

Les codes développés sont disponibles sur github via le lien suivant :
Hyperlink Codes Matlab
Le document read.me est disponible pour donner des instructions à l'utilisateur.

2 Transformation de l'image numérique

Tout d'abord, l'image doit être chargée avec la fonction MatLab imread, et affichée avec la fonction imshow. Le premier sous-étape consiste en la conversion de l'image qui est généralement lue dans une matrice de 3 couleurs (rouge, vert et bleu) dans une image en niveaux de gris avec rgb2gray, et au cas où vous voudriez obtenir la transposition en cas de textes verticaux, la fonction transpose est celle indiquée. Cela peut également se faire de deux façons : transposer l'image, puis rechercher les textes avec les opérateurs morphologiques ou utiliser les opérateurs morphologiques déjà transposés dans l'image. Nous avons choisi la première méthode.

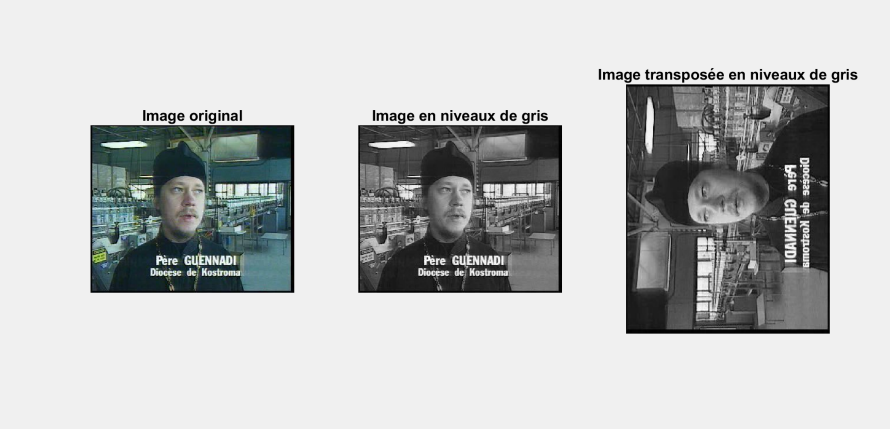

FIGURE 1 – Obtention de l'image en niveaux de gris, et de l'image transposée.

Si vous souhaitez déterminer manuellement la position des textes dans l'image, vous pouvez utiliser la fonction imtool.

```
15  %% 3.1. Digital image tranformation
16  image = evalin('base','image')
17  I = imread(image);
18  G = rgb2gray(I);
19  G_t = G';
20
21  figure()
22  subplot(1,3,1), imshow(I);
23  title("Image original")
24  subplot(1,3,2), imshow(G);
25  title("Image aux niveaux de gris")
26  subplot(1,3,3), imshow(G_t);
27  title("Image transposée en niveaux de gris")
28
```

FIGURE 2 – Code MatLab de la partie "Transformation de l'image numérique".

3 Amélioration des patrons des régions avec des textes

Dans ce cas, nous utiliserons deux opérations : d'abord nous utiliserons la multi-résolution et ensuite nous réduirons la taille et la résolution de l'image, en appliquant un facteur "m". Pour ce faire, la méthode de l'interpolation "nearest" sera utilisée, afin que chaque valeur corresponde à celle du voisin le plus proche.

Plus tard, nous utiliserons la deuxième opération : la binarisation. Pour ce faire, nous devons sélectionner un seuil par rapport à 255 (le nombre indiquant les pixels d'intensité maximale ou blancs), tel que 60% de cette valeur. Tout ce qui est supérieur à cette valeur devient 1 (blanc), et tout ce qui est inférieur devient o (noir).

```
29  %% 3.2. Enhancement of text region patterns
30
31  % m = 0.13;
32  % t = 0.7;
33  %%%%%%%%%%%%%%%%%%%%%%%%
34  H = imresize(G,m,'nearest');
35  J = im2bw(H,t);
36
37
38  figure()
39  subplot(1,3,1), imshow(H);
40  title("Image binaire")
41  subplot(1,3,2), imshow(J);
42  title('Image multi-résolution')
43
```

FIGURE 3 – Code MatLab de la partie "Amélioration des patrons des régions avec des textes".

4 Localisation des régions des textes potentielles

Dans cette phase, le but est de faire en sorte que toutes les régions blanches deviennent des rectangles, afin qu'elles soient plus faciles à indiquer (juste avec les coordonnées du point supérieur gauche et du point inférieur droit). Pour ce faire, nous allons appliquer des masques morphologiques :

— **Masque 1 :** Consiste à résoudre ce qui suit pour chaque ligne : le premier et le dernier pixel avec des valeurs unitaires sont choisis, et tous ceux entre eux sont convertis en 1.

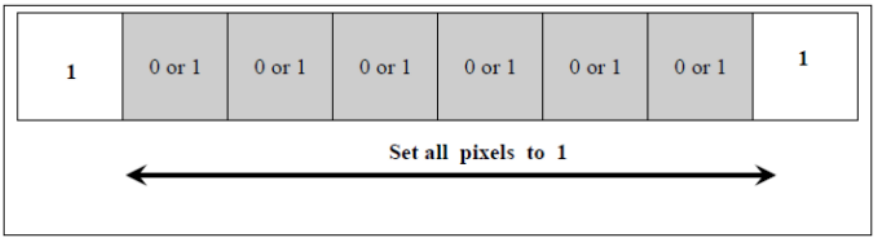

FIGURE 4 – Représentation logique du masque 1.

```
1
2   % masque 1 de la partie 3.3
3
4   function I = masque1_4(I)
5       f = size(I,1);
6       for i = 1:f
7           A = find(I(i,:)); %on utilise la fonction find pour trouver les nonzero
8           if length(A) >= 2 %si on a plus d'un pixel différent de 0
9               gauche = min(A); %border pixel de gauche
10              droite = max(A); %border pixel de droite
11              if droite-gauche > 0.75 * f  %M4 -> negative form elimination
12                  I(i,gauche:droite) = 0;
13              else
14                  I(i,gauche:droite) = 1; %on construit la ligne de 1s
15              end
16          end
17      end
18  end
```

FIGURE 5 – Code MatLab du masque 1.

— **Masque 2 :** Pour chaque pixel u, regardez le pixel situé deux colonnes plus loin et une ligne plus loin, v. Si u et v valent tous les deux 1, alors les 6 pixels qui se trouvent dans ce rectangle 2x3 vaudront également 1.
— **Masque 3 :** Homologue au masque 2, mais avec des rectangles 2x2 au lieu de 2x3.

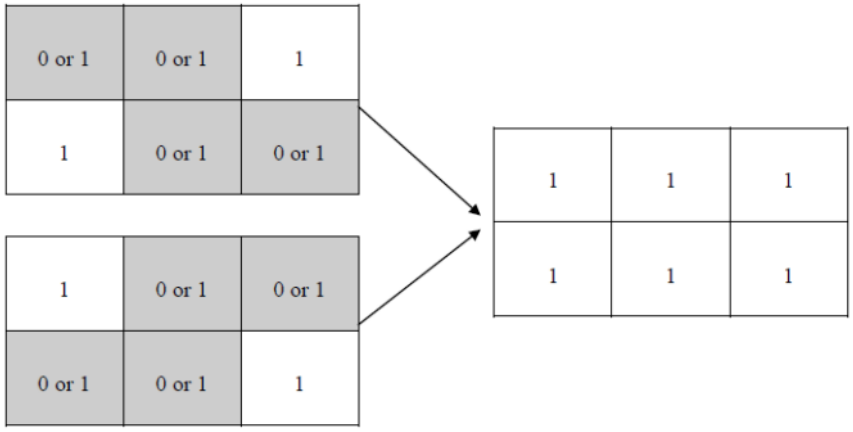

FIGURE 6 – Représentation logique du masque 2.

```
% mask M2 de la partie 3.3

function I = masque2(I)
    [A,H] = size(I);
    for i=1:A-1
        for j = 1:H-2   % on va analiser les matrices de 2x3 en commençant par l'élément en haute et à gauche
          if (I(i,j)==1 && I(i+1,j+2)==1) % si les coins en haute et gauche et en bas et droite sont 1s
              I(i,j:j+2) = 1; % on remplie la première rangée de 1s
              I(i+1,j:j+2) = 1; % on remplie la deuxième rangée de 1s
          elseif (I(i,j+2) == 1 && I(i+1,j) == 1) % si les coins en haute et droite et en bas et gauche sont 1s
               I(i,j:j+2) = 1; % on remplie la première rangée de 1s
               I(i+1,j:j+2) = 1; % on remplie la deuxième rangée de 1s
          end
        end
    end
end
```

FIGURE 7 – Code MatLab du masque 2.

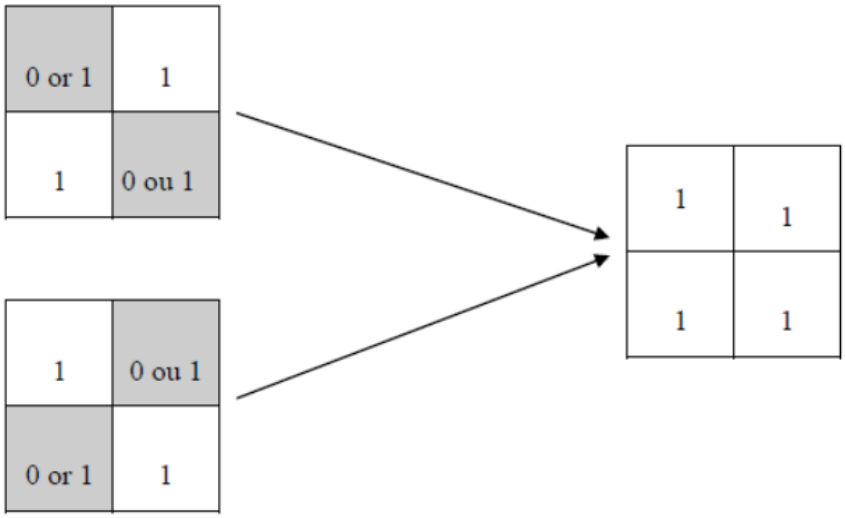

FIGURE 8 – Représentation logique du masque 3.

```
% masque M3 de la partie 3.3

function I = mask3(I)
    [L,H] = size(I);
    for i=1:L-1
        for j = 1:H-1
          if (I(i,j)==1 && I(i+1,j+1)==1 && (I(i,j+1) == 0 || I(i+1,j) ==0) )
              I(i,j+1) = 1;
              I(i+1,j) = 1;
          elseif (I(i,j+1) == 1 && I(i+1,j) == 1 && (I(i,j)==0 || I(i+1,j+1)==0))
                  I(i,j)=1;
                  I(i+1,j+1)=1;
          end
        end
    end
end
```

FIGURE 9 – Code MatLab du masque 3 .

— **Masque 4 :** Similaire au masque 1, mais si la distance entre le premier 1 et le dernier 1 de cette rangée est supérieure à 75% de la taille de la rangée (ou un autre seuil préalablement choisi), alors ils sont remplis de 0 au lieu de 1. Le code de ce masque est inclus au code du masque 1.

— **Masque 5 :** Cela permet de supprimer les 1 isolés, c'est-à-dire ceux qui n'ont pas de 1 voisin. Pour ce faire, nous regardons simplement la valeur des 8 pixels entourant chaque pixel u, et s'ils sont tous à 0 et que u est à 1, nous mettons u à 0 également.

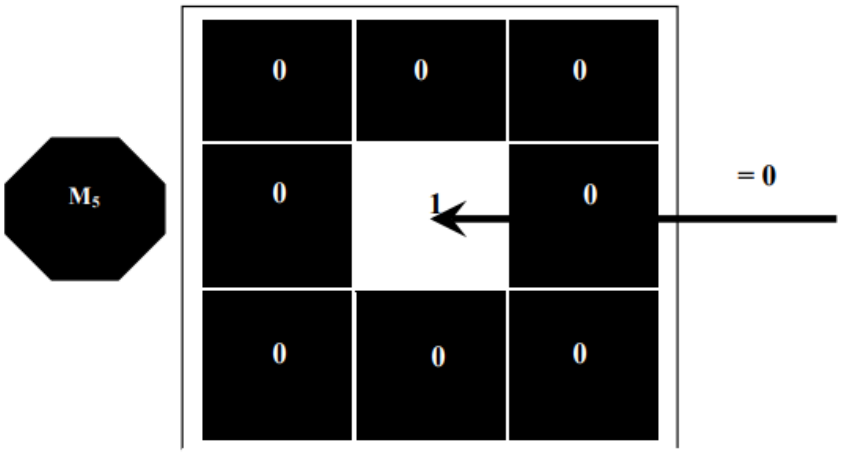

FIGURE 10 – Représentation logique du masque 5.

```
%masque 5

function I = masque5(I)
     [L,H] = size(I);
     for i=1:L-2
         for j=1:H-2
             if (sum(I(i:i+2,j:j+2),'all')==1 && I(i+1,j+1)==1)
                 I(i+1,j+1) = 0;
             end
         end
     end

end
```

FIGURE 11 – Code MatLab du masque 5.

Tout d'abord, nous appliquerons le masque 5, qui sert à éliminer les formes négatives, puis nous réaliserons un algorithme avec une boucle while où nous appliquerons les masques 1 (donc 4), 2 et 3 de manière répétée. Nous sortirons de la boucle while lorsque l'algorithme n'appliquera plus de modifications, de sorte que nous aurons obtenu des rectangles blancs dans toutes les régions présélectionnées.

```
%% 3.3. Potential text regions localization

% figure(3)
% count = 0;
% count_2 = sum(I,'all');
fprintf('On y va \n')
J2 = J;
flag = 1;
J2 = masque5(J2);
while (min(min(J==J2)) == 0 || flag == 1)
    flag = 0;
    J = J2;
    fprintf('On est passé pour la boucle \n')
    M1 = masque1_4(J); %masque 4 inclue dans la masque 1
    M1 = masque2(M1);
    M1 = masque3(M1);
    J2 = M1;
end
J = J2;


subplot(1,3,3), imshow(J);
title('Image après appliquer les masques 1, 2, 3')

```

FIGURE 12 – Code MatLab de la partie "Localisation des régions des textes potentielles".

5 Sélection des régions de textes effectives

Jusqu'à présent, nous avons trouvé des régions potentielles où les textes peuvent être trouvés. C'est maintenant au tour de choisir parmi toutes ces régions candidates, celles qui ont un vrai texte. Pour ce faire, nous allons utiliser une procédure à l'aide des histogrammes de chaque région, mais tout cela après avoir converti l'image originale à travers un BPS ou "Background Pixel Separation". Dans la suite de cet article, nous allons voir les deux procédures en détail.

FIGURE 13 – Obtention des régions candidates à contenir des textes.

5.1 BPS (Background Pixel Separation)

Cette technique consiste à choisir un seuil, qui est un pourcentage du nombre total de pixels. Disons que nous avons choisi un seuil de 20%, et que la taille de l'image est de 40x40, soit 1600 pixels. Avec cette technique, les 320 pixels ayant l'intensité la plus élevée (c'est-à-dire que nous commençons à compter à partir de 255) vaudront 1, et le reste vaudra la même valeur ou la valeur du pixel numéro 321 (en suivant l'ordre décroissant de l'intensité). Bien que, dans l'énoncé, ces deux possibilités nous aient été proposées pour l'application de cette technique, nous avons opté pour la première, de sorte que les pixels situés sous le seuil conservent la même valeur qu'auparavant.

FIGURE 14 – Obtention de l'image en utilisant la technique BPS.

5.2 Filtrage efficace des régions de texte

Au début du rapport, une propriété très curieuse et utile du texte dans les images a été mentionnée : il contraste souvent avec l'arrière-plan pour en faciliter la lecture. Nous allons utiliser l'histogramme pour détecter ce contraste. Plus précisément, nous rechercherons les valeurs les plus fréquentes (les plus répétées) et nous verrons la distance qui les sépare. Si la distance est supérieure à un certain seuil, nous considérerons que le contraste est suffisant pour que la région contienne du texte. Sinon, si la distance est très faible, ce candidat est écarté. S'il existe plusieurs valeurs ayant le même nombre de pixels (et situées à l'un des deux maxima), la plus petite distance possible sera choisie.

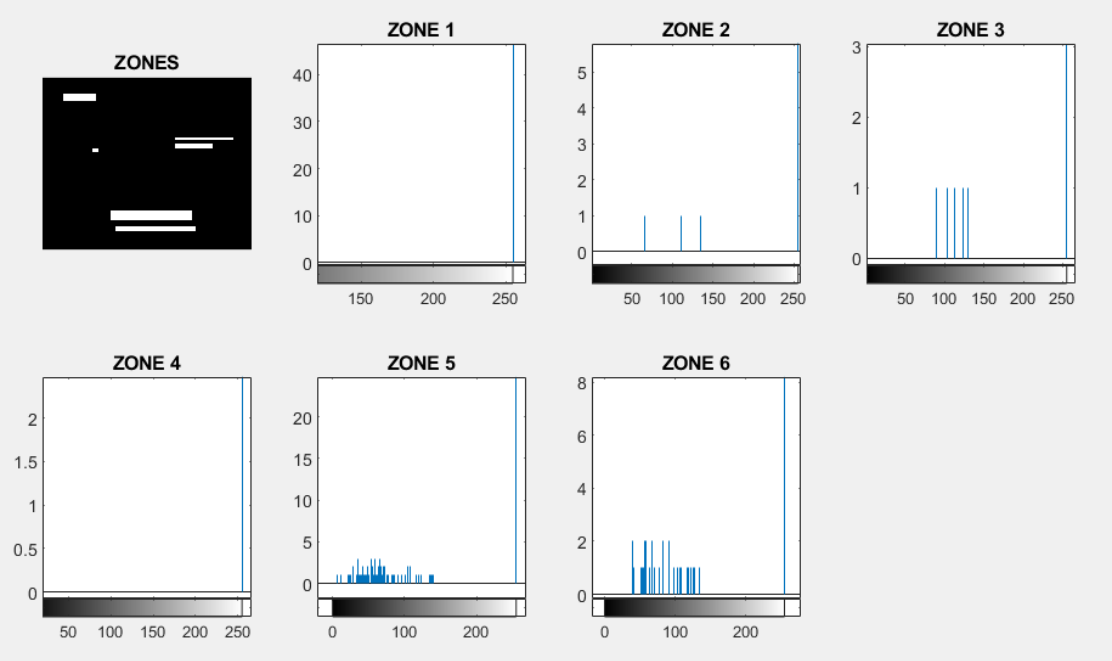

FIGURE 15 – Obtention des histogrammes des régiones des textes potentielles.

Par exemple, comme le montre la figure 15 et en sachant que les zones sont énumérées de haut en bas et de gauche à droite, les zones 1 à 4 ne sont pas prises en compte car les distances entre leurs deux maxima locaux sont faibles, ce qui ne laisse que les zones 5 et 6. On classifie les régions correspondantes comme texte.

En ce qui concerne le code MatLab, il est donc nécessaire d'obtenir d'abord la position des rectangles 1s. Comme expliqué précédemment, nous avons besoin de 4 coefficients, qui sont les positions dans les deux directions des points supérieur gauche et inférieur droit. Il est important de noter que les coordones des pixels aprés la multi-résolution ont une position differente de l'image originale. De cette façon, le processus de conversion est réalisé.

```
%% 3.4. Selection of effective text regions

% Background pixels separation

% lim = 15; % POURCENTAGE limite du nombre total des pixels pour obtenir u
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Gs = BPS(imresize(G,m,'nearest'),lim);

figure()
subplot(1,3,1), imshow(I), title('Image originale')
subplot(1,3,2), imshow(G), title('I en niveaux de gris')
subplot(1,3,3), imshow(Gs), title('BPS')

% Effective text region filtering

[f,c] = size(J);
bloc = ones(f,c); %matrice de la même taille que J pour la comparer
coord = [];     %coord sera une matrice nx4 avec les coordonnées d'une
                %région blanche dans chaque file
for i = 1:f
    for j = 1:c
        if(J(i,j)==1 && bloc(i,j) == 1) %je cherche les 1s dans J
            i1 = i ;
            j1 = j ; %chaque fois que je trouve un 1 je stocke les coord de la région
            i2 = i ; %coord de la région : coord du coin à gauche haut(i1, j1) + coord du coin à droite bas (i2, j2)
            j2 = j ;
            while (i2+1 <= f && J(i2+1,j1) == 1)
                i2 = i2 + 1 ;
            end

            while (j2+1<= c && J(i1,j2+1) == 1)
                j2 = j2 + 1 ;
            end
            coord = [coord ; [i1 j1 i2 j2]] ;
            % et on va effacer la ligne qu'on a déjà traité de notre
            % matrice bloc pour pas repeter la démarche
            bloc(i1:i2,j1:j2) = 0 ;

        elseif (J(i,j) == 0) && (bloc(i,j) == 1)
            bloc(i,j) = 0 ; %si on trouve un 0 dans J on met un 0 dans bloc
        end
    end
end


```

FIGURE 16 – Code MatLab de la partie "Sélection des régions de textes effectives (1)".

```
[f_2, c_2] = size(coord)

if f_2 == 0
    fprintf("Il n'y a pas de texte dans cette image")
else
%     histo_t = 50; % seuil / threshold pour établir les piques de l'histogramme
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% le plus petit le seuil, le plus de choses qu'il détecte comme text
    figure(4)
    nb_reg = f_2;
    subplot(2,round(nb_reg/2)+1,1), imshow(J)
    subplot(2,round(nb_reg/2)+1,2), imhist(Gs(coord(1,1):coord(1,3), coord(1,2):coord(1,4)))
    ylim auto
    if nb_reg > 1
        for i = 1:nb_reg-1
            subplot(2,round(nb_reg/2)+1,i+2), imhist(Gs(coord(i+1,1):coord(i+1,3), coord(i+1,2):coord(i+1,4)))
            ylim auto
        end
    end

    coord_check = [];
    for i = 1:f_2
        i1 = coord(i,1);
        j1 = coord(i,3);
        i2 = coord(i,2);
        j2 = coord(i,4);
        reg = Gs(i1:j1,i2:j2);
        [qty, loc] = imhist(reg);

        ord = sort(qty); %mettre en ordre tous les éléments de quantités
        P1 = find(qty==ord(length(qty)));    % prendre la quantité la plus grande
        P2 = find(qty==ord(length(qty)-1)); %et la deuxième plus grande
        dist = 260; % plus de 255 c'est bon
        for i = 1:length(P1)
            for j = 1:length(P2)
                dist_prov = abs(P1(i)-P2(j));
                if dist_prov < dist && dist_prov ~= 0
                    dist = dist_prov;
                end
            end
        end
        dist
        % on a déjà pris la distance la plus pétite, au cas où elle est répétée
```

FIGURE 17 – Code MatLab de la partie "Sélection des régions de textes effectives (2)".

```
155             if dist > histo_t/100*255
156                 coord_check = [coord_check; i1 i2 j1 j2];
157             end
158         end
159         aux = size(coord_check)
160
161         if aux(1) == 0
162             fprintf("Il n'y a pas de texte dans cette image (distance trop courte)")
163         else
164             image_finale = I;   %on va dessiner les carrés blancs pour chaque région définitive
165
166
167             for i = 1:aux(1)
168                 % on trouve les coordonnées pour la bonne taille de l'image originale
169                 i1 = round(coord_check(i,1));
170                 j1 = round(coord_check(i,2));
171                 i2 = round(coord_check(i,3));
172                 j2 = round(coord_check(i,4));
173                 [xi,xf,yi,yf] = coZone(J,i1,i2,j1,j2,G);
174                 % on dessine les lignes blanches du carré
175                 image_finale(xi,yi:yf,1:3) = 255;
176                 image_finale(xf,yi:yf,1:3) = 255;
177                 image_finale(xi:xf,yi,1:3) = 255;
178                 image_finale(xi:xf,yf,1:3) = 255;
179             end
180
181             if only_finalfigure == 1
182                 for i = 1:10
183                     close(figure(i))
184                 end
185             end
186             figure()
187             imshow(image_finale)
188         end
189     end
190     %
```

FIGURE 18 – Code MatLab de la partie "Sélection des régions de textes effectives (3)".

Après modifier les différentes paramètres, on a réussi à obtenir les bons résultats avec ceux-ci :

— t = 0.815
— m = 0.173
— lim = 20.2%
— $histo_t$ = 19.8%

Rappelons que t est le seuil limite que nous appliquons lors de la transformation de l'image en niveaux de gris en une image binaire, m est le facteur de multirésolution, lim est le seuil (en pourcentage) qui est appliqué pour la technique BPS, et enfin $histo_t$ est la distance limite (en pourcentage par rapport aux 256 niveaux de gris) entre les deux maxima de l'histogramme qui indique si la région présélectionnée contient du texte ou non.

6 Amélioration de la méthode

La méthode sans amélioration peut donner lieu à une zone de texte incomplète, probablement due à la méthode d'interpolation choisie. Dans cette section, nous présentons un processus pour combattre ce problème.

Pour effectuer une délimitation horizontale, nous choisissons la ligne de la région avec le plus de pixels appartenant aux lettres, généralement la ligne avec le nombre maximum de pixels avec la valeur L, puisque l'étape BPS a déjà été effectuée. Nous comparons ensuite cette ligne avec la ligne précédente et la suivante pour décider s'il faut tout fusionner en une seule zone. Le processus se répète jusqu'à ce que les limites supérieure et inférieure de la région de texte se stabilisent.

Pour effectuer une délimitation verticale, nous ajoutons à la ligne de référence tous les pixels qui sont à droite ou à gauche de la ligne de référence et qui ont la même couleur que la ligne de référence en respectant le principe d'élimination négative.

```
230   function [xi,xf,yi,yf] =  textBoundaries(Ir,xi,xf,yi,yf)
231 -     I = pixel_separation2(Ir,0.02);
232 -     L =  max(max(I));
233 -     maxCount = 0;
234 -     start = xi;
235 -     for i = xi:xf
236 -         countL = sum(I(i,yi:yf) == L);
237 -         if countL > maxCount
238 -             start = i;
239 -             maxCount = countL;
240 -         end
241 -     end
```

FIGURE 19 – Partie de la fonction textBoundaries responsable pour l'amélioration

```
243       %Up boundary
244 -     if start ~= 1
245 -     refLine = (I(start,:) == L);
246 -     upBound = start - 1;
247 -     upLine =  (I(upBound,:) == L);
248 -     while sum(refLine(upLine == refLine)) > 0 && upBound > 1
249 -         upBound = upBound -  1;
250 -         upLine =  (I(upBound,:) == L);
251 -     end
252 -     upBound = upBound ;
253 -     else
254 -         upBound = start;
255 -     end
```

FIGURE 20 – Processus pour la limite supérieure, pareil pour les autres limites

7 Résultats

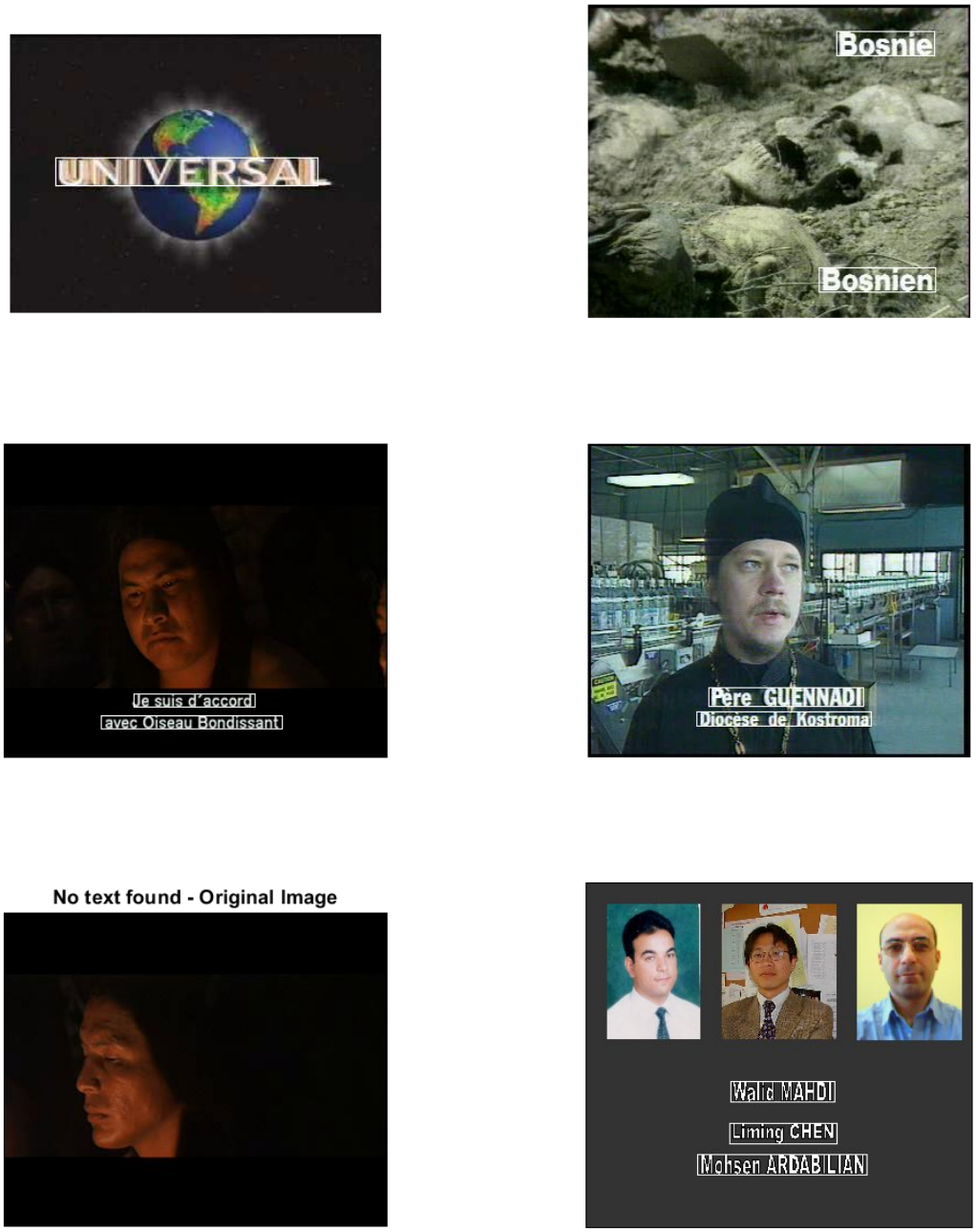

FIGURE 21 – Résultats obtenus avec les méthodes appliquées (tous les résultats sont disponibles dans le fichier sur le git)

8 Conclusion et questions

Tout au long de ce BE, nous avons appris à manipuler les images afin de détecter les régions contenant du texte. Après de nombreuses heures de travail, nous nous sommes rendu compte que ce n'est pas une tâche simple, car les textes peuvent se présenter sous différentes formes. Même si nous avons essayé d'optimiser les procédures et les algorithmes, nous n'avons pas pu obtenir des paramètres fixes valables pour toutes les images. C'est pourquoi nous avons décidé de créer une interface où ces paramètres peuvent être facilement modifiés.

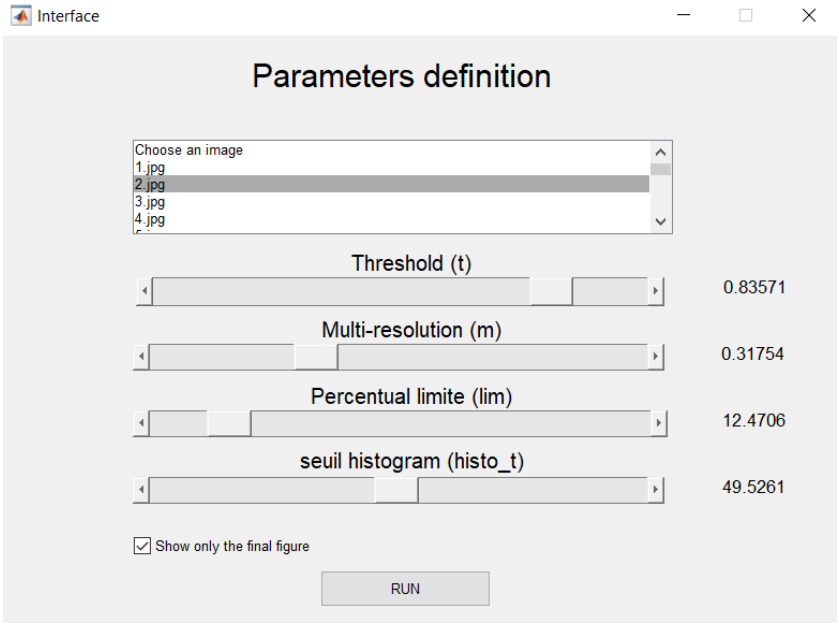

FIGURE 22 – Interface GUI pour la sélection de l'image et les paramètres

De cette façon, à partir d'une interface développée en utilisant le Guide User Interface (GUI), il est possible de choisir l'image à analyser dans la liste, d'effectuer des ajustements aux paramètres pour améliorer la résolution.

Ainsi, si les résultats obtenus pour une image particulière ne sont pas satisfaisants, des ajustements rapides peuvent être effectués.

8.1 Comment devons-nous évaluer la précision de la technique proposée ? doivent être utilisés ?

Pour évaluer la précision de la technique proposée, plusieurs facteurs doivent être pris en compte. Premièrement, la polyvalence, c'est-à-dire la possibilité d'obtenir des paramètres fixes valables pour des images très différentes. Dans ce cas, nous avons vu que cette technique n'est pas tout à fait exacte.

Par la suite, la précision pour une image spécifique doit également être analysée. C'est-à-dire, en changeant les paramètres, voir s'il est possible de détecter les régions avec du texte, et aussi voir si les boîtes sont limitées aux zones de texte, et ne sont pas plus grandes ou plus petites qu'elles ne devraient l'être. Sur ces deux derniers aspects, nous pouvons affirmer la précision de cette technique, puisque, à l'exception de quelques images spécifiques, de très bons résultats sont généralement obtenus en modifiant les paramètres.

8.2 Quels sont les inconvénients potentiels de la technique proposée. Veuillez justifier votre réponse.

Comme mentionné ci-dessus, cette technique a ses limites, car aucun paramètre n'a été trouvé qui soit valable pour toutes les images. Pour chaque type d'image et selon le contraste du texte avec l'àrriere-plan, nous devons changer les facteurs pour obtenir une bonne performance de l'algorithme, ce qui correspond à une inconvénient de la méthode proposé. Par conséquent, nous pensons que cette technique pourrait être utilisée pour accélérer un processus de détection de texte, mais seulement s'il y a une inspection ultérieure par un technicien. De cette façon, l'opérateur pourra corriger les paramètres lorsqu'il constatera que les résultats ne sont pas ceux escomptés.