BE1 - Automatic Text Regions Locating in Digital Images

Capteurs et traitement d'images

Emeline GOT - Alexandre JOYEUX

Table des matières

1 Introduction

Dans ce bureau d'étude, nous cherchons à réaliser une méthode de détection de texte dans des images sélectionnées à partir de différents programmes vidéos (publicités, journaux télévisés, films etc.). La détection de texte permet de détecter les différentes zones de texte dans les images, peu importe la taille et le style de police, et même en cas d'image avec un fond complexe. Ces techniques de détection de textes sont très utilisées, notamment dans l'extraction d'information à partir d'images ou bien dans la reconnaissance de logos dans les programmes TV.

La méthode mise en oeuvre dans ce BE utilise des moyens basiques pour détecter les zones de texte dans les images. Elle permet de prendre en compte des caractéristiques particulières des images comme un fond complexe, la présence de bruit ou bien une faible résolution.

La méthode se décompose en 4 étapes :

1. Transformation digitale de l'image
2. Amélioration des zones de texte
3. Localisation des zones de texte potentielles
4. Sélection des zones de texte réelles

Nous allons donc étudier par la suite l'implémentation de ces 4 étapes.

2 Transformation digitale de l'image

2.1 Conversion au format Bitmap

Nous souhaitons travailler avec des images de format bitmap. Nous allons donc convertir les images du format jpeg au format bmp. Pour cela, nous utilisons les formules *imread* et *imwrite* de Matlab.

```
Ijpg = imread('5.jpg','jpg');
imwrite(Ijpg,'5.bmp','bmp');
```

2.2 Conversion en niveau de gris

La deuxième étape consiste en la conversion de l'image I en une image en niveau de gris. Pour cela, on utilise la fonction *rg2gray* de Matlab.

```
I=imread('5.bmp');
G=rgb2gray(I);
```

2.3 Transposée de l'image

Dans le cas où les textes sont verticaux, il est alors intéressant de transposer la matrice correspondant à l'image pour pouvoir appliquer la méthode que l'on va implémenter et qui s'applique pour des textes horizontaux.

Pour obtenir la transposée avec Matlab, il suffit d'écrire *A.'* avec *A* la matrice que l'on souhaite transposer.

2.4 Définition d'une sous-zone

Si l'on sait où un texte a le plus de chance de se trouver (c'est le cas pour des sous-titres par exemple), il est possible de limiter la zone de recherche pour les zones de texte. Pour cela, il suffit de limiter l'image que l'on a de départ à l'intervalle que l'on souhaite étudier.

```
J=I(a:b,c:d);
```

Avec *a, b, c, d* qui délimitent la nouvelle zone.

3 Amélioration des zones de texte

Afin de rendre la localisation de texte plus facile dans une image, il est possible d'améliorer les zones de texte probables. Pour cela, nous allons voir deux méthodes : la conversion en binaire et l'approche de multi-résolution.

3.1 Multi-résolution

La méthode de multi-résolution est basée sur le fait que généralement, les textes apparaissent comme une ligne droite sur les images de basse résolution. Cette méthode consiste donc à réduire la résolution d'une image afin d'augmenter les chances de localisation des lignes de texte.

```
J=imresize(G,0.125,'nearest');
```

3.2 Conversion en binaire

La conversion d'une image en niveau de gris correspond à un seuillage. C'est-à-dire que l'on va définir un seuil pour lequel tous les pixels qui valent moins que le seuil pour l'image en niveau de gris vont être associés à la valeur 0 (qui correspond à la couleur noire) et les autres à la valeur 1 (couleur blanche).

```
M=0.7;
BW=im2bw(J,M);
```

Ici, le seuil est la valeur de M.

4 Localisation des zones de texte potentielles

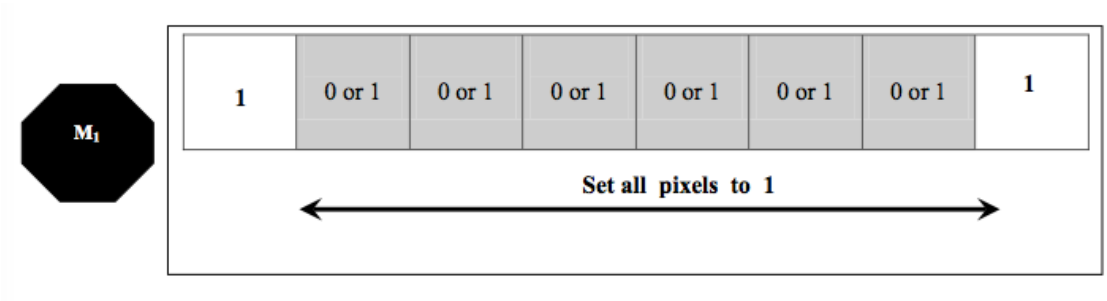
Une fois obtenue une image binaire avec des zones de texte améliorée, cette étape permet de localiser les régions qui pourraient contenir du texte dans une image.

Pour cela, nous allons appliquer des masques morphologiques pour délimiter les blocs susceptibles de contenir du texte en remplissant les zones entre les mots et les lettres.

Dans un premier temps, nous avons réalisés trois masques différents.

4.1 Masque 1

Le masque 1 consiste à trouver sur chaque ligne de l'image, les pixels qui valent 1. On repère ensuite le premier 1 et le dernier 1 de chaque ligne. Et on passe tous les pixels entre ces deux limites à la valeur 1.



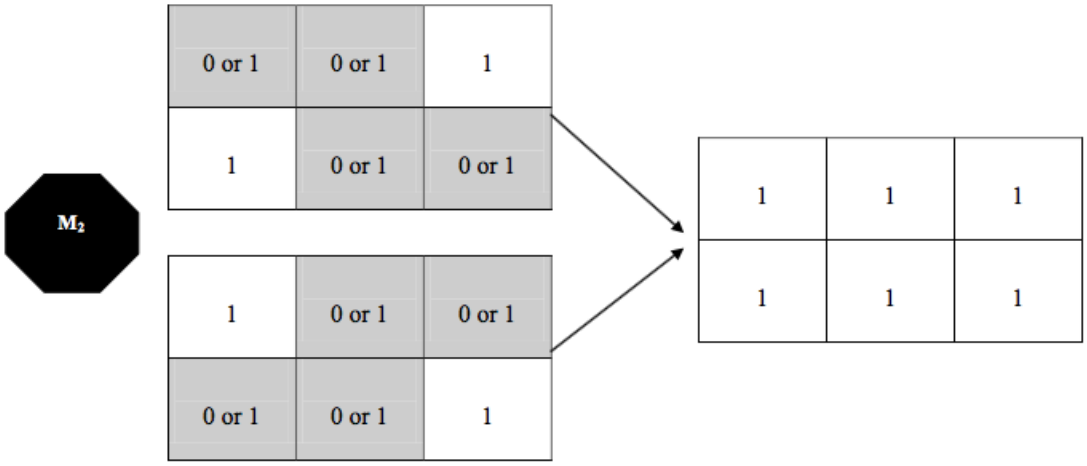
Le code correspondant est le suivant :

```
function [ sortie ] = mask1( BW )

    sortie=BW;
    [x,y]=find(BW);
    l=size(x,1);

    for i=1:l
        last=find(x==x(i),1,'last');
        %Recherche du dernier élément qui vaut 1
        sortie(x(i),y(i):y(last))=1;
        %Recouvrement en blanc de la zone entre
        %un élément et le dernier qui vaut 1 sur cette ligne
```

```
    end
14 end
```

4.2 Masque 2

Le deuxième masque permet de fermer des rectangles en considérant deux 1 qui seraient placés en diagonale l'un par rapport à l'autre. Il y a deux cas possibles qui correspondent au cas favorable du masque 2 : le premier avec un 1 en bas à gauche du rectangle et un 1 en haut à droite ; le deuxième avec un 1 en haut à gauche et un 1 en bas à droite.



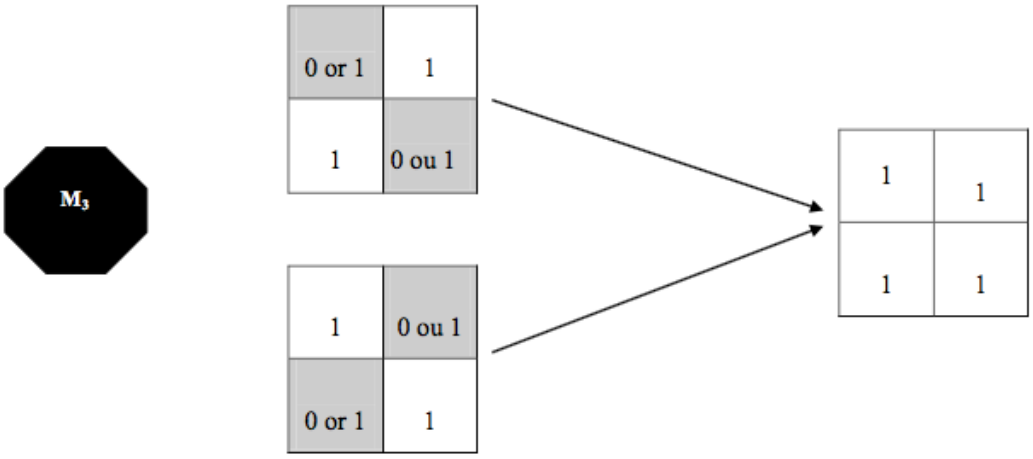
Le code correspondant est le suivant :

```
function [ sortie ] = mask2( BW )
2

    [h,l]=size(BW);
4   sortie=BW;
    [x,y]=find(BW);
6   n=size(x,1);

8   for i=1:n
      if y(i)<(l-1)
10

        if x(i)==1 %Si on est à une extrémité
12      %Pixel en haut à gauche du rectangle vaut 1
          if BW(x(i)+1,y(i)+2)==1
14        %Pixel en bas à droite du rectangle vaut 1
            sortie(x(i):x(i)+1,y(i):y(i)+2)=1;
16          %On colorie le rectangle en blanc
          end
18

        elseif x(i)==h %Si on est à l'autre extrémité
20      %Pixel en bas à gauche du rectangle
          if BW(x(i)-1,y(i)+2)==1
22        %Pixel en haut à droite du rectangle vaut 1
```

```
            sortie(x(i):x(i)-1,y(i):y(i)+2)=1;
24          %On colorie le rectangle en blanc
          end

26
        else
28        if BW(x(i)-1,y(i)+2)==1 %Rectangle normal cas 1
            sortie(x(i)-1:x(i),y(i):y(i)+2)=1;
30        end

32        if BW(x(i)+1,y(i)+2)==1 %Rectangle normal cas 2
            sortie(x(i):x(i)+1,y(i):y(i)+2)=1;
34        end

36      end
      end
38  end
end
```

4.3 Masque 3

Le masque 3 permet de fermer des carrés en étudiant les cas où les 1 sont placés en diagonale. Il correspond à une modification du masque 2.




```
1 function [ sortie ] = mask3( BW )

3   [h,l]=size(BW);
    sortie=BW;
5

    [x,y]=find(BW);
7   n=size(x,1);

9   for i=1:n
      if y(i)<l
```

```
11            if x(i)==1
                  if BW(x(i)+1,y(i)+1)==1
13                    sortie(x(i):x(i)+1,y(i):y(i)+1)=1;
                  end
15
               elseif x(i)==h
17                  if BW(x(i)-1,y(i)+1)==1
                        sortie(x(i):x(i)-1,y(i):y(i)+1)=1;
19                end

21             else

23                  if BW(x(i)-1,y(i)+1)==1
                          sortie(x(i)-1:x(i),y(i):y(i)+1)=1;
25                  end

27                  if BW(x(i)+1,y(i)+1)==1
                        sortie(x(i):x(i)+1,y(i):y(i)+1)=1;
29                  end

31             end
          end
33    end
      end
```

4.4 Application des masques morphologiques

Afin d'effectuer l'application de ces masques, on les applique à l'image tant que chaque application est source de changements non négligeables. On s'arrête donc lorsque les masques n'ont plus d'effet.

```
K=mask1(BW);
2 N=mask2(K);
while not(isequal(N,mask3(mask2(N))))
4   N=mask3(mask2(N));
end
```

5 Sélection des zones de texte réelles

Une fois que nous avons localisé des zones de textes potentielles à l'aide des masques morphologiques, il faut s'intéresser aux zones qui sont réellement du textes et donc faire un tri au niveau des zones obtenues. Pour cela, on utilise deux méthodes : la séparation des pixels de fond et le filtrage des zones de texte.

5.1 Séparation des pixels de fond

Le but de cette méthode est de mettre en évidence les lettres par rapport au fond des images. Pour cela nous appliquons une méthode de séparation d'intensité à l'image en niveau de gris

obtenue précédemment. En suivant la méthode décrite dans l'énoncé, nous obtenons le code suivant :

```
function [ u,J ] = separation( I )

  L=255;
  H=imhist(I);
  [h,l]=size(I);
  seuil=round(0.02*h*l);
  i=256;
  Nb=0;

  while Nb<seuil
      i=i-1;
       Nb=Nb+H(i);
  end

  u=i;

  J=I;
  for i=1:h
      for j=1:l
          if I(i,j)>=u-1
              J(i,j)=L;
          end
      end
  end
end
```

5.2 Filtrage des zones de texte

On s'intéresse maintenant séparément à chacune des zones de texte obtenues avec l'ensemble des méthodes précédentes. Pour chacune de ces zones, on vérifie son potentiel à être une zone de texte. Pour cela, on regarde l'histogramme de cette zone et plus particulièrement les pics de cet histogramme. Si ces pics sont trop rapprochés, alors la zone est disqualifiée pour être considérée comme véritable zone de texte et inversement. Ainsi on effectue un tri des zones de textes trouvées en fonction d'un seuil fixé.

Le code correspondant est le suivant :

```
[u,O]=separation(G);

[h,l]=size(N);
for i=1:h
    for j=1:l
        if N(i,j)==1
            x=i;
            y=j;

            while x<h && N(x+1,j)==1
                x=x+1;
```

```
                end

                while y<l && N(x,y+1)==1
                    y=y+1;
                end

                H=imhist(O(i/0.125:x/0.125,j/0.125:y/0.125));

                max=0;
                k1=0;
                k2=0;

                for n=1:256
                    if H(n)>max
                        k1=n;
                        max=H(n);
                    end
                end

                H(k1)=0;
                max=0;

                for n=1:256
                    if H(n)>max
                        k2=n;
                        max=H(n);
                    end
                end

                threshold=0.15*256;
                if abs(k1-k2) < threshold
                    N(i:x,j:y)=0;


                end
            end
        end
    end
```

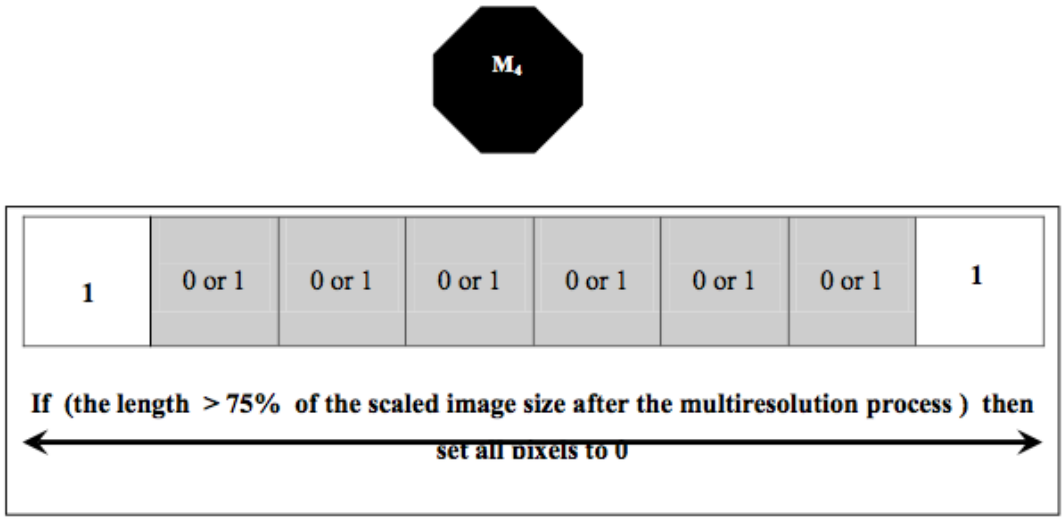
6 Améliorations

Pour améliorer les techniques vues précédemment pour la détection des zones de texte, on implémente deux autres masques.

6.1 Masque 4

Le masque 4 est une amélioration du masque 1. Il considère aussi des lignes de pixels mais si les deux 1 sont trop éloignés (de plus de 75% de l'image après la méthode de multirésolution),

tous les pixels sont passés à 0 plutôt qu'à 1.

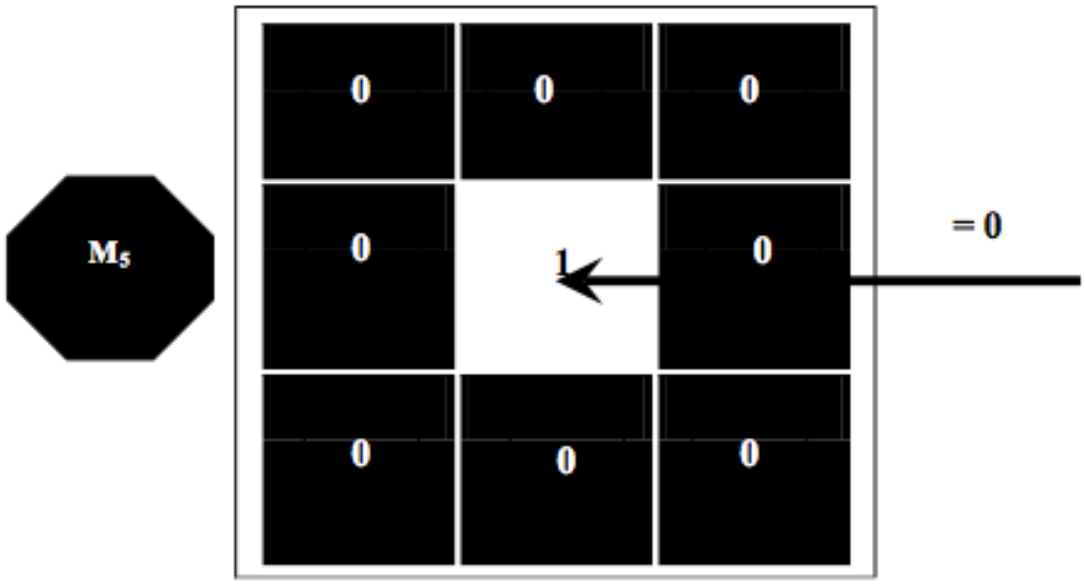


Voici le code pour le masque 4 :

```
function [ sortie ] = mask4( I )

  sortie=I;
  [x,y]=find(I);
  [a,b]=size(I);
  l=size(x,1);
  for i=1:l
    last=find(x==x(i),1,'last'); %Recherche du dernier élément qui vaut 1
    if (last-i)>(0.75*a*b) %Si on dépasse les 75% de l'image
      sortie(x(i),y(i):y(last))=0; %On met tous les pixels en noir
    else
      sortie(x(i),y(i):y(last))=1; %Sinon les pixels en blanc comme dans
          mask1
    end
  end
end
```

6.2 Masque 5

Le masque 5 permet d'éliminer les pixels isolés. Si un pixel blanc est entouré de pixel noir, alors le pixel est passé au noir. Cette étape doit être réalisée avant les autres masques afin que les pixels isolés dans les images n'interviennent pas dans la création de rectangle par la suite avec les autres masques.




```
function [ sortie ] = mask5( I )

    sortie=I;
    [x,y]=find(I);
    [a,b]=size(I);
    l=size(x,1);

    for i=1:l
      if x(i)>1 && x(i)<a && y(i)>1 && y(i)<b
        if I(x(i)-1,y(i)-1)==0 && I(x(i)-1,y(i))==0 && I(x(i)+1,y(i)+1)==0
           && I(x(i),y(i)-1)==0 && I(x(i),y(i)+1)==0 && I(x(i)+1,y(i)-1)==0
            && I(x(i)+1,y(i))==0 && I(x(i)+1,y(i)+1)==0 %On vérifie que
            tous les pixels autour sont noirs
          sortie(x(i),y(i))=0;
        end
      end
    end
end
```

7 Expériences

Les résultats ne sont pas très pertinents, il y a probablement des erreurs dans les fonctions utilisées.


UNISOL 2500


Walt Disney


15 MARS
ANNÉE TERRIENNE 2454

8 Questions

8.1 Comment devrions-nous évaluer la précision de la technique proposée ? Quel critère devrait être utilisé ?

Pour évaluer la technique proposée, nous pourrions nous baser sur la largeur des zones détectées par rapport au texte réel. Si les cadres sont proches des textes, cela veut dire que la méthode est très précise car elle ne considère pas de pixel en trop. Si les cadres sont trop éloignés, alors cela suggère que la méthode n'est pas assez précise et qu'elle ne détecte pas assez la zone réelle du texte.

8.2 Quels sont les désavantages potentiels à la technique proposée ? Justifiez votre réponse.

Cette technique permet de détecter des lignes de texte mais ne permet pas de déterminer s'il y a plusieurs mots par exemple. Nous savons où se situent les textes mais s'il y a un texte en haut à gauche de l'image et un deuxième texte en haut à droite de l'image aligné, nous ne savons pas qu'il y a deux textes distincts et cette méthode ne permettra de n'en détecter qu'un.