# Un nouveau descripteur de texte pour la détection des lignes de texte multi-orientées dans les scènes réelles

Mehdi Felhi, Salvatore Tabbone, Nicolas Bonnier

# Un nouveau descripteur de texte pour la détection des lignes de texte multi-orientées dans les scènes réelles

**Mehdi Felhi*,** — Salvatore Tabbone* — Nicolas Bonnier ****

* *Laboratoire Loria, Université Lorraine*

*{mehdi.felhi,antoine.tabbone}@loria.fr*

** *Océ Print Logic Technologies*

*nicolas.bonnier@oce.com*

RÉSUMÉ. *Dans cet article, nous proposons un nouveau descripteur de texte basé sur la squelettisation et la fonction distance appliquées sur les composantes candidates de texte. Nous proposons aussi une approche complète d'extraction des lignes de texte multi-orientées et curvilignes. Notre nouvelle approche commence par extraire les composantes connexes. Ces régions seront regroupées pour former un graphe qui relie par l'intermédiaire de ses arcs les composantes similaires et voisines. Notre descripteur de texte sera appliqué dans un deuxième temps au niveau de chaque nœud du graphe. Pour cela, nous extrayons pour chaque composante son squelette et nous effectuons un ébarbulage qui consiste à garder uniquement les traits/régions d'épaisseurs quasi-uniformes. La proportion de ces régions par rapport à la taille de la composante est considérée comme un indicateur d'appartenance du nœud (correspondant à la composante analysée) à la classe « texte ». Ensuite, nous raffinons les connexions du graphe ainsi formées en utilisant un algorithme modifié de parcours en profondeur (DFS) appliqué sur les arcs. Ce traitement vise à éliminer les connexions non susceptibles de former des lignes de texte. Finalement, nous appliquons l'algorithme de Graph Cuts sur le graphe construit afin de détecter les régions/lignes de texte.*

ABSTRACT. *In this paper, we propose a new text descriptor based on skeleton and distance transform. Our approach begins with extracting connected components and selecting text character candidates. The text regions are represented as an undirected graph by connecting similar components. The text descriptor performs on the node level. Firstly, we calculate the skeleton of each text character candidate and then multiply it by the distance transform (that separate the foreground pixels from the background) in order to prune the skeleton by preserving only branches having low variation in terms of distance transform. We assume that the text characters have high percentage of "low variation" branches. Finally, our method predicts the label for all the nodes of the graph using the text descriptor and the Graph Cuts algorithm.*

MOTS-CLÉS : *détection de texte, lignes de texte multi-orientées, squelettisation, ébarbulage, algorithme de parcours en profondeur, Graph Cuts.*

KEYWORDS: *text detection, multi-oriented text lines, skeletonization, pruning, first depth search, Graph Cuts.*

## 1. Introduction

Le texte est souvent considéré comme l'une des structures les plus importantes à extraire dans les images de documents ainsi que dans les scènes réelles. En effet, le texte contient une source d'information riche d'un point de vue sémantique. Ces informations étant exploitables dans différents types d'applications, telles que l'indexation automatique, la compression des données, et l'amélioration de la qualité de l'impression. La détection et l'extraction du texte est une tâche complexe qui consiste à localiser et à séparer le texte des arrière-plans (homogènes ou complexes) et/ou des graphiques, principalement dans le but de reconnaître les caractères à l'aide d'un OCR. Plusieurs obstacles peuvent rendre la tâche de détection du texte de plus en plus difficile; comme le faible contraste entre le texte et le fond, l'existence de quelques/plusieurs régions non-texte texturées, le fond non-uniforme, et l'orientation du texte qui pourrait ne pas être horizontale. La catégorisation des méthodes de détection de texte est une tâche difficile puisque les classes de méthodes se chevauchent. D'un point de vue de la représentation des images, on peut classifier les méthodes de détection de texte en deux catégories principales; les méthodes basées sur la représentation spatiale et celles basées sur la représentation dans le domaine fréquentiel. L'idée principale derrière les approches liées au domaine fréquentiel est basée sur la conversion (ou la compression) des images. Cette conversion est garantie par l'intermédiaire d'opérateurs souvent linéaires qui assurent la transformation de l'image du domaine spatial à un domaine plus représentatif. Parmi les transformées les plus utilisées pour extraire le texte nous pouvons citer la transformée en cosinus discrète (TCD) [1,2], et la transformée en ondelettes discrète (TOD). L'avantage de cette dernière transformée par rapport à la TCD est le fait qu'elle représente à la fois l'information spatiale et l'information fréquentielle. La TOD est utilisée souvent comme une étape de pré-traitement permettant de localiser les régions ayant des fréquences élevées et ainsi les régions candidates de texte. D'autre part, les méthodes basées sur le domaine spatial exploitent les informations intrinsèques des pixels. Ces informations incluent par exemple l'orientation du gradient. Garcia *et al.* [6] exploitent la particularité suivante des caractères du texte afin de détecter les régions de texte: un caractère de texte contient des contours ayant plusieurs orientations différentes. D'autres travaux se sont basés sur la segmentation par régions de l'image. En effet, le texte peut se caractériser par sa couleur et/ou par sa forme géométrique. M. León *et al*. [7] représentent l'image sous forme d'un arbre Min-Max. Puis, ils sélectionnent les régions candidates de textes en se basant sur des descripteurs de texte, pour finir par extraire les régions uniformes formant ainsi les régions de texte. En outre, certaines méthodes de détection de texte utilisent des techniques d'apprentissage automatique. Par exemple R. Lienhart et al [8] proposent de générer des vecteurs caractéristiques des blocs de pixels et de les classer en texte et non-texte par le biais de réseaux de neurones artificiels. Néanmoins, la classe de ces méthodes qui utilisent l'intelligence artificielle a montré quelques limites. En fait, le système de détection de texte associé peut produire de fausses alarmes et des taux de rejet insatisfaisants à cause

des bases de données d'apprentissage généralement spécifiques pour certains types de textes et d'arrière-plan spécifiques.

Notre travail peut être considéré comme une approche par composantes connexes qui vise à généraliser certaines méthodes existantes [15, 16] où on considère que les épaisseurs des caractères de texte présentent des largeurs uniformes. Plusieurs papiers ont exploité cette propriété. Par exemple, dans [15] les auteurs ont introduit la transformée en épaisseur des traits (SWT), cette transformée permet de déterminer la largeur de chaque trait appartenant à chaque composante candidate de texte analysée, puis de sélectionner celles qui ont de faibles variations (en dessous d'un certain seuil) et les considérer comme des candidates de caractères de texte. La SWT est calculée en déterminant la distance séparant les deux gradients opposés le long des contours de la composante connexe considérée. Cette méthode réalise des taux de détection élevés quand le texte est bien contrasté. Alternativement, le gradient le long du contour de la composante n'est pas bien déterminé et le calcul de la largeur du trait peut échouer. H. Chen *et al.* [16] ont proposé une nouvelle version pour le calcul des épaisseurs de traits en se basant sur la fonction de distance qui surmonte le problème susmentionné. Formellement, les auteurs de [16] calculent la distance qui sépare chaque pixel de l'élément considéré avec l'arrière-plan qui l'entoure à l'aide de la fonction de distance, et ensuite propagent les valeurs maximales le long de ces traits. Enfin, ils considèrent que l'épaisseur réelle des traits formant le candidat de texte est le double de sa valeur calculée par l'intermédiaire de la fonction distance. Cependant, ces méthodes risquent d'échouer lorsque le caractère de texte présente des variations « globales » au niveau des traits qui constituent un caractère donné (comme le cas de la lettre "w" dans la police Times New Roman qui présentent deux régions « d'égale épaisseur »). En outre, les méthodes [15, 16] ne traitent pas les lignes de texte orienté.

Motivés par les travaux [15, 16], nous introduisons dans ce papier un nouveau descripteur de texte qui mesure l'uniformité « locale » des épaisseurs des traits composant les caractères de texte et permettant de sélectionner les candidats de texte qui sont composés de régions « d'égales épaisseurs » pour ensuite calculer une probabilité pour qu'un composant soit un caractère de texte en fonction du pourcentage des régions « d'égales épaisseurs » déterminées. D'autre part, et contrairement aux travaux connexes, notre méthode vise à détecter les lignes de texte multi-orientées et curvilignes.

## 2. L'approche proposée

### 2.1. Aperçu général

Notre approche se constitue essentiellement de quatre étapes principales :

1- Etape de pré-traitement : Cette étape consiste à rehausser le contraste entre l'avant-plan (potentiellement le texte) et le fond de l'image en appliquant un filtre non-linéaire. L'objectif de cette étape est de lisser l'image afin d'éliminer les détails indésirables ainsi que le bruit, tout en maintenant le contraste entre les différents objets de l'image. La phase de filtrage est suivie de l'extraction des composantes connexes à l'aide de l'algorithme « Maximally Stable Extremal Regions » (MSER).

2- Construction du graphe: Nous éliminons dans cette étape les composantes connexes qui constituent les régions non-textes et connectons les composantes similaires par des arêtes. Puis, nous affinons ces connexions en utilisant une approche de parcours de graphe en profondeur (DFS) et en imposant des critères de voisinage entre les arêtes qui forment le graphe en question afin de conserver uniquement les candidats de ligne de texte. Le choix des attributs du graphe et de l'algorithme DFS sera décrit dans les paragraphes suivants.

3- Etape de traitement: dans cette étape, nous appliquons notre descripteur de texte sur chaque nœud du graphe.

4- Coupure de graphe (Graph Cuts): À la fin de cette étape, chaque nœud du graphe (composante connexe) est étiqueté comme étant texte ou non-texte.

### 2.2. Pré-traitement

Avant de détecter les composantes connexes de l'image, une étape de filtrage est indispensable pour éviter les fausses alarmes ainsi que les faux rejets. En fait, les régions texte et non-texte pourraient présenter du bruit qui empêche éventuellement l'extraction des composantes connexes. Pour cela, nous avons choisi la technique de diffusion anisotropique [17] pour deux raisons principales: 1) sa capacité à préserver le contraste entre le texte et le fond et 2) sa capacité d'éliminer le bruit et de lisser des régions homogènes. Une fois l'image est filtrée, nous appliquons l'algorithme MSER [18] pour extraire les composantes connexes. Principalement, cette technique est utilisée comme une méthode de détection de « blobs ». Une région MSER est une région qui est soit plus claire ou plus sombre que le fond qui l'entoure. Cette méthode a prouvé sa supériorité comparée à plusieurs descripteurs de textures existants [19] en termes de robustesse et de performances. Par ailleurs, le MSER a été utilisé comme une étape de prétraitement pour la méthode de détection de texte

[16] et les auteurs ont montré que cette méthode est capable de détecter facilement les caractères du texte comme étant des composantes homogènes.

Dans le présent travail, nous avons utilisé la librairie [20] qui inclue une implémentation du MSER. La fonction MSER extrait les zones covariantes de l'image. Ces régions seront ensuite entourées par des ellipses qui les englobent et qui les caractérisent, cette étape permet de décrire chaque objet par quatre paramètres. En fait, la définition de chaque ellipse est assurée par le calcul du centre de chaque région (moyenne) en première étape, et par l'extraction des éléments indépendants de la variance de l'ensemble des pixels composant l'élément considéré en seconde étape. Les expressions de ces paramètres pour chaque région $R_c$ découlent des expressions de la moyenne $\mu^c$ et de la variance $\Sigma^c$ ainsi définies:

$$\mu^c = \frac{1}{|R_c|}\sum_{x\in R_c} x, \quad \Sigma^c = \frac{1}{|R_c|}\sum_{x\in R_c}\left(x-\mu^c\right)^T\left(x-\mu^c\right) \tag{1}$$

Où $|R_c|$ exprime la surface de la région $R_c$.

A noter que, $\mu^c = \left(\mu_1^c, \mu_2^c\right)$ est un vecteur à deux dimensions et que le nombre total des éléments indépendants de la variance est égale à trois. Ces cinq scalaires définissent ainsi l'équation d'une ellipse :

$$\left(x-\mu^c\right)^T S_c^{-1}\left(x-\mu^c\right) = 1 \tag{2}$$

Où

$$S_c = \begin{pmatrix} s_1^c & s_{1,2}^c \\ s_{1,2}^c & s_2^c \end{pmatrix}. \tag{3}$$

Les paramètres ainsi définis caractérisent pour chaque ellipse la forme géométrique et l'étalement de la composante connexe correspondante. Pour finaliser cette tâche, nous projetons les matrices $\left(S_c^{-1}\right)_{c\in I}$ dans l'espace propre de la manière suivante:

$$S_c^{-1} = P_c^T \begin{pmatrix} \lambda_1^c & 0 \\ 0 & \lambda_2^c \end{pmatrix} P_c \tag{4}$$

Ainsi, le grand axe $a_c$ et le petit axe $b_c$ de l'ellipse sont calculés comme suit:

$$a_c = \max\left(\lambda_1^c, \lambda_2^c\right) \quad b_c = \max\left(\lambda_1^c, \lambda_2^c\right) \tag{5}$$

Dans le paragraphe suivant, nous allons utiliser les paramètres $a_c$ et $b_c$ afin d'éliminer les régions non-textes pour construire le graphe contenant les candidats de texte.

### 2.3. Construction du graphe

Le but de cette étape est de construire un graphe qui renferme les candidats de texte. Pour ce faire, nous commençons par l'application de certaines règles afin d'éliminer les régions non-textes. Par exemple, une composante connexe de très petite taille est considérée comme non-texte, de même, une région qui a un ratio hauteur/largeur élevé par rapport à l'unité est négligée. D'autre part, les composantes connexes qui présentent des variations faibles par rapport à leurs tailles respectives sont ignorées. Pour mettre en œuvre ces contraintes, nous proposons d'utiliser les paramètres de l'ellipse $a_c$ et $b_c$ correspondants aux composantes extraites. En fait, le ratio $a_c / b_c$ approxime le rapport hauteur / largeur d'un caractère de texte quelle que soit son orientation. A noter que cette approximation est beaucoup plus fiable que la technique des boîtes englobantes d'un point de vue précision lorsque les lignes textes ne sont pas horizontales. Par ailleurs, nous avons remarqué que la compacité de chaque composante peut être estimée en comparant l'aire de la région avec l'aire de l'ellipse correspondante. En fait, si une région présente de faibles variations le long des deux axes (grand axe $a_c$ et petit axe $b_c$ ), c'est qu'elle définit une région compacte. Or, un caractère de texte présente souvent une faible solidité. Ainsi, nous proposons d'éliminer les régions ne respectant pas les propriétés suivantes :

$$\begin{cases} \dfrac{a_c}{b_c} < t_1 \\ \dfrac{\pi . a_c . b_c}{|R_c|} < t_2 \end{cases} \tag{6}$$

En imposant ces contraintes, nous obtenons les régions candidates de texte. Nous les considérons comme étant les nœuds du graphe. Ensuite, nous mesurons la relation entre les nœuds afin de les regrouper en calculant les similarités entre eux. Les arêtes du graphe doivent connecter des nœuds semblables et voisins. Pour cette raison, nous proposons quatre caractéristiques binaires qui permettront de quantifier la similitude et la proximité entre deux régions candidates texte $c_i$ et $c_j$ (voir tableau 1). Les heuristiques qu'on propose sont indépendantes de l'échelle et l'orientation du texte.

- La distance relative (DR): Nous avons défini ce paramètre puisque les composantes du texte appartenant à la même ligne de texte sont proches et que les composantes voisines sont susceptibles d'avoir la même étiquette (texte ou non-texte).

- Rapport d'échelle (RE): Nous proposons de comparer uniquement les longueurs respectives des grands axes des composantes $c_i$ et $c_j$. En effet, les caractères d'une même ligne de texte possèdent des longueurs proches du point de vue longueur des grands axes contrairement au rapport entre les petits axes: par exemple, le caractère « i » présente un petit axe $b_i$ très faible comparé à celui du caractère « m » $b_m$. Toutefois, les deux caractères ont des longueurs de grands axes proches (ie. $a_m \cong a_i$).
- Position relative (PR): Cette fonction calcule l'angle que fait l'arête reliant les centres respectifs de chaque couple de nœuds dans le graphe avec l'horizontal. Elle sera l'entrée de notre approche DFS (voir paragraphe suivant) qui vise à affiner les connexions du graphe afin de conserver uniquement les connexions qui forment des lignes de texte (quelles que soit leurs orientations ou leurs courbures).
- Différence entre les niveaux de gris (DNG): Cette fonction est utilisée à cause du fait que les caractères de texte voisins ont souvent des valeurs de couleur proches.

| **Heuristique** | **Définition** |
|---|---|
| *Distance Relative* | $DR(c_i, c_j) = \frac{\left\| \mu^{c_i} - \mu^{c_j} \right\|_2}{b_{c_i} + b_{c_j}}$ |
| *Rapport d'Echelle* | $RE(c_i, c_j) = \frac{a_{c_i}}{a_{c_j}}$ |
| *Position Relative (angle)* | $RP(c_i, c_j) = \tan^{-1}\left( \frac{\left(\mu_2^{c_j} - \mu_2^{c_i}\right)}{\left(\mu_1^{c_j} - \mu_1^{c_i}\right)} \right)$ |
| *Différence de Niveaux de Gris* | $DNG(c_i, c_j) = \left\| I(c_i) - I(c_j) \right\|$ |

Tableau 1: Heuristiques binaire pour le couple de noeud ($c_i$,$c_j$).

Comme indiqué précédemment, le graphe doit comporter les candidats de texte comme étant des nœuds. Les arêtes relient les composantes similaires. Cette configuration vise à regrouper les caractères appartenant à une même ligne de texte. À cette fin, nous avons utilisé le SVM comme classifieur sur deux bases de données d'apprentissage. Ces deux bases de données sont constituées par un ensemble de couple de nœuds différents. La première se compose de caractères de texte voisins appartenant à une même ligne de texte tandis que la deuxième est composée d'un ensemble aléatoire de composants non-textes voisins. Nous avons utilisé les heuristiques *DR*, *RE*, et *DNG* comme une entrée au SVM.

## 2.4. Le descripteur de texte

Deux raisons principales nous ont motivés à proposer une nouvelle version qui vise à généraliser la SWT [15]. La première raison réside dans le fait que l'épaisseur des traits qui composent chaque caractère de texte est la même pour tous

les pixels appartenant à la direction orthogonale au vecteur gradient du contour de ce caractère. La valeur de cette épaisseur peut être approchée par deux fois la valeur de la transformée de distance appliquée au pixel ayant la valeur maximale dans cette direction. Cette idée a été proposée dans [16]. Toutefois, cette valeur est unique, donc il est inutile de la calculer pour chaque pixel appartenant aux candidats de texte. La deuxième raison consiste à éliminer les faux rejets. Généralement, les caractères de texte ne sont pas composés de traits d'épaisseur «globale» uniforme. Néanmoins, ils sont composés de plus d'un trait d'épaisseur « locale » uniforme. Dans [16], les auteurs n'ont pas pris en compte cette propriété et éliminent donc les composantes ayant une variance d'épaisseur « globale » élevée.

Le descripteur proposé est basé sur la squelettisation et sur la transformée de distance. Le squelette qu'on va construire doit décrire fidèlement la structure de la composante correspondante (candidate de texte). Ainsi, nous avons besoin d'ébarbuler les branches squelettiques qui ne sont pas utiles pour l'analyse de ces composantes ou qui ont de faibles contributions à la description de sa structure. L'idée derrière l'ébarbulage des branches squelettiques est basée sur le fait que les branches pointant vers la frontière sont des branches « inutiles » [25]. Par conséquent, nous proposons le critère suivant: les branches ayant une variance en transformée de distance élevée sont supprimées. Mathématiquement, les branches qui présentent un rapport écart-type/moyenne supérieur à un seuil $t_3$ donné sont éliminées. Pour ce faire, nous commençons par détecter les points d'intersections des branches dans le squelette considéré afin de les analyser séparément de manière à éliminer les branches qui ont une variance d'épaisseur élevée. Ainsi, nous obtienons des régions «d'égales épaisseurs ». Finalement, nous calculons le pourcentage de ces régions pour chaque composante candidate de texte. Intuitivement, les régions présentant des pourcentages élevés des régions « d'égales épaisseurs » sont plus susceptibles d'être des régions de texte. Ce pourcentage est calculé comme suit:

$$P_c = \frac{\left| |R_c| - 2 \times \sum_{x \in Sk_c} dist(x) \right|}{|R_c| + 2 \times \sum_{x \in Sk_c} dist(x)} \tag{7}$$

Où, $S_{k_C}$ représente les régions «d'égales épaisseurs » déduites après l'étape d'ébarbulage. *dist* est la distance séparant les pixels appartenant au squelette de l'arrière-plan. A noter que la valeur de $P_c$ varie dans l'intervalle [0, 1]. Une faible valeur de $P_C$ signifie un pourcentage élevé des régions « d'égales épaisseurs » et une probabilité élevée pour que le composant $C$ représente un caractère de texte, donc nous proposons d'utiliser une fonction décroissante du paramètre $P_c$ comme une densité de probabilité pour qu'un composant soit du texte. En effectuant une analyse statistique sur une base de données d'apprentissage contenant des caractères de texte avec leurs valeurs correspondantes $P_c$, nous avons remarqué que l'histogramme ainsi

construit peut être approché par une demie-gaussienne [21]. Par conséquent, nous estimons la variance de cette distribution par l'intermédiaire de l'estimateur classique sans biais. En conclusion, la densité de probabilité d'un composant *C* étant du texte peut être définie comme suit:

$$f(C) = \frac{2}{\sqrt{2\pi\hat{\sigma}^2}}\left(\exp\left(-\frac{P_c^{\,2}}{2\hat{\sigma}^2}\right)\right), (x \geq 0) \tag{8}$$

Où, $\hat{\sigma}$ représente la valeur de la variance associée à la distribution gaussienne estimée (voir Figure 1).

### 2.5. Raffinement du graphe

Avant d'appliquer le Graph Cuts et de segmenter ainsi l'image, nous éliminons les composantes ayant de faibles probabilités de représenter du texte. Concrètement, si à une composante *C* correspond une valeur de *f (C)* inférieure à *f (0) / T*, elle sera automatiquement éliminée du graphe. Deuxièmement, nous raffinons toutes les connexions (arêtes) entre chaque couple de nœuds en fonction de la distribution spatiale des caractères dans une même ligne de texte: dans cette étape, nous proposons une approche modifiée de parcours de graphe en profondeur (DFS) appliquée aux arêtes du graphe (et non pas aux nœuds contrairement au DFS classique). Nous utilisons pour cela, le paramètre *PR* définie dans le paragraphe 2.3. Cette étape vise à éliminer les connexions entre les composantes qui ne pourraient pas appartenir à une même ligne de texte. En fait, nous supposons que la différence entre les angles que font deux arêtes successives avec l'horizontale $(PR(i,j), PR(j,k))$ et qui relient trois nœuds $(c_i, c_j, c_k)$ du graphe devrait être aussi petite que possible (plus petite que $\theta_0$). Cette dernière hypothèse reste vraie même si la ligne de texte présente une courbure. Par ailleurs, en effectuant un DFS sur les arêtes du graphe, nous forçons un nœud à appartenir au groupe ayant le plus grand nombre possible de nœuds connectés et obéissant au critère mentionné ; dans le reste de l'article *(Cr)* désignera ce critère. L'algorithme 1 montre un pseudo-code de l'approche DFS proposée. A noter que la fonction "valid_neighbors" de l'algorithme 1 impose le critère *(Cr)*.

| Algorithme 1: L'approche DFS pour le raffinement du graphe |
|---|
| **Entrée:**<br>*PR = (PR(ci,cj));*<br>*Edge_i*; // pour chaque arête i, nous calculons sa profondeur correspondante (longueur de la ligne de texte potentielle), puis nous sélectionnons toutes les arêtes qui appartiennent à cette éventuelle ligne de texte ;<br>*PTS;* // définit l'ensemble d'arêtes valides qui relient les nœuds similaires |

et voisins;
**Sortie:**
*D_i* ; // profondeur correspondante à l'arête i;
*C_i* ; // ensemble d'arêtes correspondants au plus long chemin (profondeur) en commençant par l'arête i;
**Pseudo-algorithme:**

```
[D_i   C_i] = DFS(RP, Edge_i, PTS_i)
if (isempty(valid_neighbors(Edge_i)))
    D_i = 1;
    C_i = Edge_i;
    else
        v = valid_neighbors(Edge_i);
        PTS = {PTS}\{Edge_i}; // on élimine Edge_i
        [D_v_m C_v_m] = argmax(DFS(RP, Edge_vk, PTS), vk);
        D_i = D_i + D_v_m;
        C_i = [C_i   C_v_m];
end
```

En accomplissant cette étape, nous définissons l'ensemble des arêtes valides du graphe ainsi que les éventuelles lignes de texte.

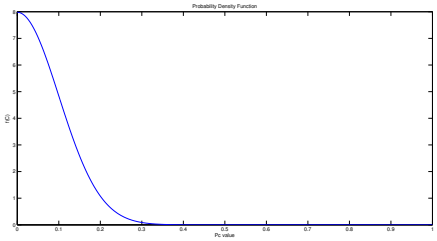



Figure 1: La fonction densité de probabilité qui définie la probabilité d'un composant *C* d'appartenir à la région texte.

.

### 2.5. Segmentation : Graph Cuts

L'objectif de cette étape est de définir et de minimiser une fonction de coût afin de séparer les zones « textes » de celles « non-textes ». L'entrée de cette étape de segmentation est le graphe *G* décrit dans les paragraphes précédents. Dans ce paragraphe, nous décrivons brièvement l'expression de la fonction à optimiser afin de segmenter le graphe $G$ (et ainsi l'image) et de labelliser les composantes candidates de textes détectées précédemment.

$$E(G) = E_{data}(G) + E_{smooth}(G) = \sum_{C \in G} D_c(l_c) + S. \sum_{(C_1,C_2) \in G} V_{C_1,C_2}(l_{C_1}, l_{C_2}) \quad [8]$$

où $$S = \alpha . f(0) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$


(a) Image originale


(b) Génération du graphe après le regroupement des composantes similaires et voisines

(c) Les valeurs $P_c$ calculées pour chaque composante candidate de texte: les valeurs des pixels de fond sont mis à la valeur -0,5 pour plus de visibilité. Nous remarquons que les régions qui se composent de traits "l'égales d'épaisseurs" ont des valeurs très faibles de $P_c$ par rapport aux autres régions

(d) Résultat final: après calcul du descripteur de texte pour chaque nœud et après l'étape de segmentation

Figure 2: Exemple de résultat de détection réalisée par la méthode proposée

L'équation 8 montre la forme générale de la fonction de coût appliquée sur le graphe $G$. Cette équation contient principalement trois termes importants :

- $D_C(l_C)$ : C'est la fonction qui calcule le coût d'appartenance du nœud $C$ à la classe $l_C \in \{"texte","non\ texte"\}$. Dans cet article nous utilisons l'expression de la densité de probabilité $f(C)$ pour la définir. Explicitement, nous considérons que le coût d'appartenance du nœud $C$ à la classe « texte » est égal à $1 - f(C)$. Par ailleurs, nous considérons que le coût d'appartenance du nœud $C$ à la classe « non texte » est égal à $f(C)$.

- $V_{C_1,C_2}(l_{C_1}, l_{C_2})$ : C'est la fonction qui quantifie le voisinage entre les nœuds $C_1$ et $C_2$. Dans cet article elle est égale à $DR(C_1, C_2)$.

- $S$ : Cette matrice est déterminée empiriquement. Elle sert à pénaliser le fait que deux nœuds voisins appartiennent à deux classes différentes.

Le problème de segmentation revient donc à résoudre un problème de minimisation d'énergie (équation 8). Pour cela, nous proposons d'utiliser la méthode des Graph Cuts [26,27,28]. La Figure 2 montre un exemple de détection du texte par l'approche proposée.

## 3. Résultats expérimentaux

Pour les expérimentations, nous avons évalué notre nouvelle approche en utilisant la base de données publique ICDAR [22] conçue pour les compétitions ICDAR 2003 et 2005 pour la localisation de texte. Cette base de données se compose de deux ensembles d'images, le premier ensemble d'images est consacré à l'apprentissage et contient 258 images tandis que le deuxième est consacré aux tests et contient 251 images. Toutes les images de la base de données ICDAR sont des images de scènes naturelles et sont capturées à l'aide d'une caméra numérique dans des conditions de luminosité différentes. Principalement, deux heuristiques sont utilisées pour l'évaluation ; le taux de précision et le taux rappel. Le taux de précision est défini comme le rapport entre le nombre des régions correctement détectées et le nombre total de régions détectées. Tandis que le taux de rappel est défini comme le ratio entre les régions correctement détectées et le nombre total de régions qui existent dans la vérité terrain. Les formules mathématiques de ces heuristiques sont explicitement définies dans [22].

Les paramètres de l'algorithme proposé ont été empiriquement déterminés:

$t_1 = 6,\ t_2 = 3,\ t_3 = 0.5,\ T = 10,\ \alpha = 5$

Le tableau 2 montre une comparaison entre les performances de notre méthode et les travaux connexes [15,16,23,24] sur la base de données ICDAR.

| Méthode | Approche proposée | Epshtein [15] | Chen [16] | Minetto [23] | Fabrizio [24] |
|---|---|---|---|---|---|
| Précision | 69% | 73% | 73% | 63% | 46% |
| Rappel | 61% | 60% | 60% | 61% | 39% |

Tableau 2: Résultats expérimentaux sur la base de données ICDAR [22]

Nous en concluons que notre méthode réalise des résultats prometteurs en la comparant aux performances des méthodes existantes. Cependant, notre méthode est conçue pour faire face aux problèmes de détection des lignes de texte multi-orientées et les lignes de texte présentant des courbures. Or, la base de données ICDAR ne comporte que des lignes de texte non courbées et souvent orientées horizontalement. Comme il n'existe pas de bases de données publiques conçues pour la détection des lignes de texte courbées, nous avons collecté 45 images qui contiennent des régions textes où les lignes de texte sont multi-orientées, courbées ou horizontales afin de construire notre propre base de données. Nous avons construit manuellement la vérité terrain associée à cette base de données et nous avons utilisé les mêmes heuristiques à savoir le taux de précision et le taux de rappel. Nous avons également implémenté la méthode décrite dans le papier [16] afin de faire une comparaison des performances. Nous avons choisi de faire une comparaison avec la méthode [16] puisqu'elle réalise l'un des meilleurs taux de précision sur la base de données ICDAR. Le tableau 3 montre une évaluation des performances de notre méthode avec celle de la méthode [16].

| Méthode | Approche proposée | Chen [16] |
|---|---|---|
| Précision | 75% | 52% |
| Rappel | 67% | 43% |

Tableau 3: Résultats expérimentaux de la détection sur notre propre base de données

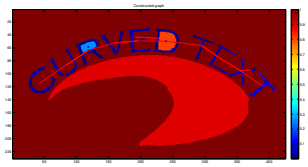
Les figures 3 et 4 montrent la capacité de notre méthode à détecter les lignes de texte incurvées tandis que la méthode [16] échoue sur ce type de lignes de texte (voir figure 4).


(a) Images originales

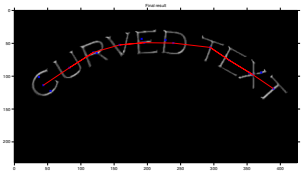

(b) Résultats de la détection

Figure 3: Détection des lignes de texte incurvées en utilisant notre méthode proposée.

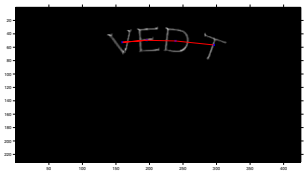

(a) Image originale




(b) Construction et raffinement du graphe




(c) Résultat final de détection en utilisant l'approche proposée




(d) Résultat de détection en utilisant la méthode [16]

Figure 4: Comparaison de la précision de détection des lignes de texte incurvé réalisé par la méthode proposée en la comparant avec celle proposée dans [16]

## 4. Conclusion

Dans ce papier, nous avons présenté une nouvelle approche de détection de texte dans les scènes réelles. La méthode commence par regrouper les nœuds similaires et les composantes candidates de texte voisines dans un graphe. Les connexions entre les nœuds sont raffinées par une approche de DFS modifiée afin de ne conserver que les liens entre les caractères qui appartiennent à une même ligne de texte. Ensuite, nous avons défini un nouveau descripteur de texte et nous l'avons considéré comme étant un attribut correspondant à chaque nœud du graphe. Ce descripteur est une fonction du pourcentage des régions « d'égales épaisseurs » incluses dans les composantes candidates de texte. Contrairement aux travaux existants, la méthode proposée est capable de détecter les lignes de textes multi-orientées et incurvées. Les résultats expérimentaux ont démontré que notre approche réalise des taux de précisions et de rappel prometteurs en la comparant aux méthodes existantes.

## 5. Bibliographie

[1] D. Chen, J.-M. Odobez, H. Bourlard: Text detection and recognition in images and video frames, Pattern Recognition, Volume 37, Issue 3, March 2004, Pages 595-608.
[2] Y. Zhong, H. J. Zhang, A. K. Jain: Automatic Caption Localization in Compressed Video. IEEE Trans. Pattern Anal. Mach. Intell. 22(4): 385-392 (2000).
[3] J. Li, R. M. Gray: Text and Picture Segmentation by the Distribution Analysis of Wavelet Coefficients. ICIP (3) 1998: 790-794.
[4] H. Li, D. S. Doermann, O. E. Kia: Automatic text detection and tracking in digital video. IEEE Transactions on Image Processing 9(1): 147-156 (2000).
[5] D. Q. Zhang, and S.-F. Chang: General and Domain-specific Techniques for Detecting and Recognizing Superimposed Text in Video, ICIP 2002.
[6] C. Garcia and X. Apostolidis: Text detection and segmentation in complex color images,. Proceedings of ICASSP 2000, (2000) Vol. IV, 2326–2330.
[7] M. Le´on, S. Mallo and A. Gasull: A Tree Structured-Based Caption Text Detection Approach, Proceedings of the fifth IASTED International Conference Visualization, Imaging, and Image Processing , pp 220-226, September 7-9, 2005, Spain.
[8] E. K.Wong, M. Chen: A new robust algorithm for video text extraction. Pattern Recognition 36 (2003) 1397-1406.
[9] R. Lienhart, A. Wernicke: Localizing and segmenting text in images and videos, IEEE Transactions Circuits Syst. Video Technol. 12 (4) (2002) 256-268.
[10] T. Sato, T Kanade, E.K. Hughes, and M.A. Smith: Video OCR for Digital News Archive. IEEE International Workshop on Content Based Access of Image and Video Databases CAIVD'98, pages 52-60, January 1998.
[11] V.Y. Mariano and R. Kasturi: Locating Uniform-Colored Text in Video Frames. In Proc. International Conference on Pattern Recognition, volume 4, pages 539-542, 2000.
[12] M. Li , M. Bai , C. Wang and B. Xiao: Conditional random field for text segmentation from images with complex background. Pattern Recognition Letters, vol. 31, no. 14, pp. 2295 - 2308 , 2010.
[13] E. Haneda, C. A. Bouman: Text Segmentation for MRC Document Compression. IEEE Transactions on Image Processing 20(6): 1611-1626 (2011).

[14] W. Pan, T.D. Bui, C.Y. Suen: Text detection from scene images using sparse representation," ICPR 2008. 19th International Conference on Pattern Recognition , vol., no., pp.1-5, 8-11 Dec.
[15] B. Epshtein, E. Ofek, and Y. Wexler: Detecting text in natural scenes with stroke width transform. in CVPR, 2010, pp. 2963 –2970.
[16] H. Chen, S. Tsai, G. Schroth, D. Chen, R. Grzeszczuk, B. Girod: Robust text detection in natural images with edge-enhanced maximally stable extremal regions, IEEE International Conference on Image Processing (ICIP), September 2011.
[17] P. Perona and J. Malik: Scale-Space and Edge Detection Using Anisotropic Diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(7):629-639, July 1990.
[18] J. Matas, O. Chum, M. Urban, T. Pajdla: Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. BMVC 2002.
[19] P.-E. Forssén, D. G. Lowe: Shape Descriptors for Maximally Stable Extremal Regions. ICCV 2007.
[20] A. Vedaldi, B. Fulkerson, "VLFeat: An Open and Portable Library of Computer Vision Algorithms" 2008.
[21] L.S. Nelson: The Folded Normal Distribution. J Qual Technol 12 (1980). (4): 236–238.
[22] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young: ICDAR 2003 robust reading competitions, ICDAR 2003: 682-687.
[23] R. Minetto, N. Thome, M. Cord, J. Fabrizio, and B. Marcotegui: Snoopertext: A multiresolution system for text detection in complex visual scenes, in ICIP, 2010, pp. 3861–3864.
[24] J. Fabrizio, M. Cord, and B. Marcotegui: Text extraction from street level images, in CMRT, 2009, pp. 199–204.
[25] J.-H. Jang, K.-S. Hong: Detection of curvilinear structures and reconstruction of their regions in gray-scale images. Pattern Recognition 35(4): 807-824 (2002).
[26] Y. Boykov, O. Veksler, R. Zabih: Efficient Approximate Energy Minimization via Graph Cuts IEEE transactions on PAMI, vol. 20, no. 12, p. 1222-1239, November 2001.
[27] V. Kolmogorov,R. Zabih: What Energy Functions can be Minimized via Graph Cuts? IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 26, no. 2, February 2004, pp. 147-159.
[28] Y. Boykov, V. Kolmogorov: An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 26, no. 9, September 2004, pp. 1124-1137.