



Extraction hybride et description structurelle de caractères pour une reconnaissance efficace de texte dans les documents hétérogènes scannés : Méthodes et Algorithmes parallèles

Mahmoud Soua

► To cite this version:

Mahmoud Soua. Extraction hybride et description structurelle de caractères pour une reconnaissance efficace de texte dans les documents hétérogènes scannés : Méthodes et Algorithmes parallèles. Informatique et langage [cs.CL]. Université Paris-Est, 2016. Français. NNT : 2016PESC1069 . tel-01548457

HAL Id: tel-01548457

<https://pastel.hal.science/tel-01548457>

Submitted on 27 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Université Paris-Est

**Ecole Doctorale Mathématiques et Sciences et
Technologies de l'Information et de la Communication
(MSTIC)**

THÈSE

Présentée pour obtenir le grade de docteur en Informatique par

Mahmoud SOUA

**Extraction hybride et description structurelle de caractères pour
une reconnaissance efficace de texte dans les documents
hétérogènes scannés : Méthodes et Algorithmes parallèles**

Soutenue le 8 Novembre 2016 devant le jury composé de :

M. Antoine MANZANERA	Professeur, ENSTA ParisTech	Rapporteur
M. Thierry GERAUD	Professeur, EPITA	Rapporteur
M. Mohamed Hédi BEDOUI	Professeur, LabTIM	Examineur
M. Nicolas GAC	Maître de conférences, Université Paris-Sud	Examineur
M. Mohamed AKIL	Professeur, Professeur, ESIEE Paris	Directeur
M. Rostom KACHOURI	Professeur associé, ESIEE Paris	Encadrant

*Cette thèse est dédiée à mes parents,
sans lesquels je n'aurais jamais vu le jour.*

*Les machines sont déjà interactives
avec le cerveau : l'ordinateur et le
genre humain travaillent ensemble. Il
se pourrait d'ailleurs qu'un jour les
historiens se rendent compte que
l'événement le plus important du XXe
siècle, ce n'était pas la guerre, ni le
krach financier, mais le soir où
Kasparov, le joueur d'échecs, a perdu
sa partie contre une petite boîte en
métal. Et noté : La machine n'a pas
calculé, elle a pensé.*

Entretien Télérâma n° 3230 décembre

2011

George Steiner

Remerciements

Je tiens à exprimer toute ma reconnaissance à ceux qui ont contribué à ce travail, en particulier :

Mon directeur de thèse Monsieur Mohamed Akil et mon co-encadreur Monsieur Rostom Kachouri, de m'avoir bien dirigé et encadré tout au long de mes travaux de recherche, pour leurs disponibilités, leurs efforts pour la réalisation de ce travail.

Monsieur Antoine Manzanera enseignant-chercheur à l'ENSTA-ParisTech pour avoir accepté d'être rapporteur de cette thèse, pour ses conseils instructifs qu'il m'a donnés et pour ces remarques contribuant ainsi à l'amélioration de ce manuscrit. Monsieur Thierry Géraud professeur à l'EPITA pour avoir accepté d'être rapporteur de cette thèse et pour ses précieuses remarques. Monsieur Mohamed Hedi Bedoui professeur à la Faculté de Médecine de Monastir et Monsieur Nicolas Gac Maître de conférences à l'université Paris-Sud pour avoir accepté de faire partie de ce jury.

Je tiens à remercier mes collègues et amis du département IT et de l'équipe A3SI à l'ESIEE Paris. Une fois de plus je remercie mon père Habib, ma mère Rafia, mes soeurs Yosra et Ines pour leurs soutiens et leurs encouragements durant ces années d'études. Plus personnellement, je remercie ma fiancée Arij, pour son aide, son écoute et surtout son amour qui m'a été essentiel durant ces années. Je remercie tous mes plus proches amis, Oussama, Nejmeddine, Imen, Syrine, Alae, Yosra, Takwa, Ibtissem, Yaareb, Ramzi, Geoffray, Varun...

Résumé

La Reconnaissance Optique de Caractères (OCR) est un processus qui convertit les images textuelles en documents textes éditables. De nos jours, ces systèmes sont largement utilisés dans les applications de dématérialisation tels que le tri de courriers, la gestion de factures, etc. Dans ce cadre, l'objectif de cette thèse est de proposer un système OCR qui assure un meilleur compromis entre le taux de reconnaissance et la vitesse de traitement ce qui permet de faire une dématérialisation de documents fiable et temps réel. Pour assurer sa reconnaissance, le texte est d'abord extrait à partir de l'arrière-plan. Ensuite, il est segmenté en caractères disjoints qui seront décrits ultérieurement en se basant sur leurs caractéristiques structurales. Finalement, les caractères sont reconnus suite à la mise en correspondance de leurs descripteurs avec ceux d'une base prédéfinie.

L'extraction du texte, reste difficile dans les documents hétérogènes scannés avec un arrière-plan complexe et bruité où le texte risque d'être confondu avec un fond texturé/varié en couleurs ou distordu à cause du bruit de la numérisation. D'autre part, la description des caractères, extraits et segmentés, se montre souvent complexe (calcul de transformations géométriques, utilisation d'un grand nombre de caractéristiques) ou peu discriminante si les caractéristiques des caractères choisies sont sensibles à la variation de l'échelle, de la fonte, de style, etc. Pour ceci, nous adaptons la binarisation au type de documents hétérogènes scannés. Nous assurons également une description hautement discriminante entre les caractères se basant sur l'étude de la structure des caractères selon leurs projections horizontale et verticale dans l'espace. Pour assurer un traitement temps réel, nous parallélisons les algorithmes développés sur la plateforme du processeur graphique (GPU). Nos principales contributions dans notre système OCR proposé sont comme suit :

- **Une nouvelle méthode d'extraction de texte à partir des documents hétérogènes scannés incluant des régions de texte avec un fond complexe ou homogène.** Dans cette méthode, un processus d'analyse d'image est employé suivi d'une classification des régions du document en régions d'images (texte avec un fond complexe) et de textes (texte avec un fond homogène). Pour les régions de texte on extrait l'information textuelle en utilisant une méthode de classification hybride basée sur l'algorithme Kmeans (CHK) que nous avons développée. Cette méthode combine des approches locales et globales. Elle améliore la qualité de séparation texte/fond, tout en minimisant la quantité de distorsion pour l'extraction de texte à partir des documents scannés et bruités à cause du processus de la numérisation. Les régions d'images sont améliorées avec une Correction Gamma (CG) avant d'appliquer CHK. Les résultats obtenus d'expérimentations, montrent que notre méthode d'extraction de texte permet d'atteindre un taux de reconnaissance de caractères de 98,5% sur des documents hétérogènes scannés.
- **Un Descripteur de Caractère Unifié basé sur l'étude de la structure des caractères.** Il emploie un nombre suffisant de caractéristiques issues de l'unification des descripteurs de la projection horizontale et verticale des caractères réalisant une discrimination plus efficace. L'avantage de ce descripteur est à la fois sa haute performance et sa simplicité en termes de calcul. Il supporte la reconnaissance des

caractères multi-échelle et multi-fonte. Le descripteur proposé permet d'avoir une reconnaissance de caractère de 100% pour une fonte et une taille données.

- **Une parallélisation du système de reconnaissance de caractères.** Le processeur graphique GPU a été employé comme une plateforme de parallélisation. Flexible et puissante, cette architecture offre une solution efficace pour l'accélération des algorithmes de traitement intensif d'images. Notre mise en œuvre, combine les stratégies de parallélisation à fins et gros grains (fine/coarse-grained) pour accélérer les étapes de la chaîne OCR. En outre, les coûts de communication CPU-GPU sont évités et une bonne gestion mémoire est assurée. L'efficacité de notre mise en œuvre est validée par une expérimentation approfondie.

Mots-clefs

OCR, Système Temps Réel, Binarisation, Documents Hétérogènes, Description, Reconnaissance, Parallélisation, GPU.

Abstract

The Optical Character Recognition (OCR) is a process that converts text images into editable text documents. Today, these systems are widely used in the dematerialization applications such as mail sorting, bill management, etc. In this context, the aim of this thesis is to propose an OCR system that provides a better compromise between recognition rate and processing speed which allows to give a reliable and a real time documents dematerialization. To ensure its recognition, the text is firstly extracted from the background. Then, it is segmented into disjoint characters that are described based on their structural characteristics. Finally, the characters are recognized when comparing their descriptors with a predefined ones.

The text extraction, based on binarization methods remains difficult in heterogeneous and scanned documents with a complex and noisy background where the text may be confused with a textured background or because of the noise. On the other hand, the description of characters, and the extraction of segments, are often complex using calculation of geometrical transformations, polygon, including a large number of characteristics or gives low discrimination if the characteristics of the selected type are sensitive to variation of scale, style, etc. For this, we adapt our algorithms to the type of heterogeneous and scanned documents. We also provide a high discrimination between characters that description is based on the study of the structure of the characters according to their horizontal and vertical projections. To ensure real-time processing, we parallelise algorithms developed on the graphics processor (GPU). Our main contributions in our proposed OCR system are as follows:

- **A new binarisation method for heterogeneous and scanned documents including text regions with complex or homogeneous background.** In this method, an image analysis process is used followed by a classification of the document areas into images (text with a complex background) and text (text with a homogeneous background). For text regions is performed text extraction using a hybrid method based on classification algorithm Kmeans (CHK) that we have developed for this aim. This method combines local and global approaches. It improves the quality of separation text/background, while minimizing the amount of distortion for text extraction from the scanned document and noisy because of the process of digitization. The image areas are improved with Gamma Correction (CG) before applying HBK. According to our experiment, our text extraction method gives 98% of character recognition rate on heterogeneous scanned documents.
- **A Unified Character Descriptor based on the study of the character structure.** It employs a sufficient number of characteristics resulting from the unification of the descriptors of the horizontal and vertical projection of the characters for efficient discrimination. The advantage of this descriptor is both on its high performance and its simple computation. It supports the recognition of alphanumeric and multi-scale characters. The proposed descriptor provides a character recognition 100% for a given Face-type and Font-size.

-
- **Parallelization of the proposed character recognition system.** The GPU graphics processor has been used as a platform of parallelization. Flexible and powerful, this architecture provides an effective solution for accelerating intensive image processing algorithms. Our implementation, combines coarse/fine-grained parallelization strategies to speed up the steps of the OCR chain. In addition, the CPU-GPU communication overheads are avoided and a good memory management is assured. The effectiveness of our implementation is validated through extensive experiments.

Keywords

OCR, Real Time System, Binarization, Heterogeneous Documents, Description, Recognition, Parallelization, GPU.

Table des matières

Liste des figures	13
Liste des tableaux	15
Notations	17
Introduction Générale	21
1 Cadre et motivations	21
2 Organisation de la thèse et contributions	26
1 Reconnaissance de caractères dans les documents hétérogènes	29
1 Introduction	30
2 Extraction de texte dans les documents hétérogènes	32
2.1 Méthodes basées sur l'analyse de textures	33
2.1.1 Les méthodes structurelles	34
2.1.2 Les méthodes spatio-fréquentielles	34
2.1.3 Les méthodes statistiques	34
2.2 Méthodes basées sur l'analyse des Régions	35
2.3 Méthodes basées sur l'analyse des contours	36
3 Description des caractères	36
3.1 Approche stochastique	37
3.2 Approche statistique	37
3.2.1 Description du contour : descripteurs de Fourier	37
3.2.2 Description de la forme : moments de Zernike	38
3.3 Approche topologique	38
3.4 Approche structurelle	39
4 Reconnaissance des caractères	39
5 Architectures parallèles	40
6 Conclusion	41

2	Extraction hybride de texte dans les documents hétérogènes	43
1	Introduction	44
2	Extraction de texte	44
2.1	Binarisation	45
2.1.1	Les méthodes de binarisation Globales	45
2.1.2	Les méthodes de binarisation locales	46
2.2	Analyse de la structure du document	49
2.2.1	Les méthodes ascendantes	49
2.2.2	Les méthodes descendantes	50
2.3	Localisation des caractères	51
2.3.1	Localisation holistique	51
2.3.2	Localisation basée sur la reconnaissance des caractères . . .	51
2.3.3	Localisation basée sur la dissection	51
3	Nouvelle méthode hybride pour l'extraction de texte dans les documents hétérogènes : EHT	51
3.1	Séparation des régions texte/images dans les documents hétérogènes .	53
3.1.1	Classification des pixels	53
3.1.2	Recherche des colonnes basée sur les tabulations	54
3.1.3	Recherche des régions non-rectangulaires	56
3.2	Binarisation hybride de texte	57
3.2.1	Correction gamma des régions d'images pour une séparation fond/forme optimale	58
3.2.2	Optimisation de l'algorithme de correction gamma	67
3.2.3	Classification hybride basée sur le noyau Kmeans	71
3.2.4	Correction du rendu couleur basée sur l'étude de la densité de pixels	75
3.3	Localisation des caractères par dissection	77
4	Expérimentations	77
4.1	Binarisation hybride de texte	78
4.2	Optimisation algorithmique de la correction gamma	79
4.3	Extraction hybride de texte dans les documents hétérogènes	80
5	Conclusion	83
3	Reconnaissance de texte basée sur un nouveau descripteur structural de caractères	87
1	Introduction	88
2	Caractéristiques structurales conventionnelles des caractères	88
3	Proposition d'un système de reconnaissance de caractères CR-UCD	89
3.1	Nouvelle description structurale de caractères : UCD	90

3.1.1	Calcul des segments de caractères	90
3.1.2	Nombre de Segments	91
3.1.3	Position des Segments	92
3.1.4	Coordonnées des Barycentres	93
3.1.5	Proportion des dimensions	94
3.2	Sélection du modèle	96
3.2.1	Reconnaissance de la taille	96
3.2.2	Reconnaissance de la Fonte	97
3.3	Mise en correspondance des caractéristiques	98
4	Expérimentation	100
4.1	Évaluation du descripteur UCD	101
4.2	Évaluation du Système CR-UCD	103
5	Conclusion	105
4	Parallélisation sur GPU de l'extraction hybride du texte dans les documents hétérogènes	107
1	Introduction	109
2	Architectures Multi-cœur	110
2.1	Les CPU multi-cœur	111
2.2	Les processeurs graphiques GPU	112
3	Parallélisation de la méthode de l'extraction hybride de texte	113
3.1	Analyse temporelle de la méthode EHT	113
3.2	Stratégie de parallélisation de la méthode EHT	114
3.3	Accélération de la correction gamma (CG)	115
3.3.1	Calcul parallèle du seuil Otsu	116
3.3.2	Génération parallèle des matrices de co-occurrence	119
3.3.3	Calcul parallèle des caractéristiques de Haralick	122
3.3.4	Implantation sur un nœud de GPU	125
3.4	Accélération de la classification hybride basée sur le noyau K-means .	127
3.4.1	Accélération de l'algorithme Kmeans sur GPU : État de l'art	128
3.4.2	Stratégie de parallélisation de la méthode de classification hybride CHK	131
3.5	Accélération de l'analyse de structure du document (Séparation et localisation du texte)	136
4	Évaluation de la stratégie de parallélisation de la méthode EHT	137
4.1	Outils de développement et environnement de travail	138
4.2	Évaluation de la parallélisation de la correction gamma	138
4.3	Évaluation de la parallélisation de la méthode CHK	138
4.3.1	Variation de la taille des blocs	140

4.3.2	Variation de la quantité du bruit	141
4.3.3	Variation du contenu des documents	142
4.4	Évaluation de la parallélisation de la méthode d'analyse de structure de documents	143
4.5	Évaluation de l'accélération proposée de la méthode EHT	143
5	Conclusion	147
Conclusion Générale		149
1	Synthèse des contributions	150
2	Perspectives	152
Annexe		155
Liste de publications		203
Bibliographie		205

Liste des figures

1	Extraction de texte à partir d'un arrière-plan complexe. (a) : Image comprenant un texte avec un arrière-plan couleur complexe; (b) : Texte extrait par la méthode Sauvola [68].	22
2	Extraction de texte avec une variation de la luminance. (a) Image comprenant un arrière-plan avec une variation de la luminance (b) Texte extrait par seuillage global (c) Texte extrait par seuillage local.	23
3	Exemple d'extraction de texte erronée dans un document hétérogène. (a) Document hétérogène couleur composé de régions de textes et d'images; (b) Texte extrait par la méthode Sauvola MS_{kx} [69].	24
4	Description par matrices de pixels de la même lettre 'A'. (a) : Fonte Arial; (b) : Fonte Times; (c) : Fonte Times avec du bruit.	25
1.1	Principe de la dématérialisation de documents	30
1.2	Architecture classique d'un Système OCR.	30
1.3	Exemples de sources d'informations textuelles : Magazines, Journaux.	31
1.4	Exemples de sources d'informations textuelles. (a) : Chèques; Documents administratifs; (b) : Photographies, (c) : Légendes d'images.	31
1.5	Exemple de document hétérogène.	32
1.6	Extraction de texte dans un document scanné. (a) : Image Originale. (b) : Texte extrait.	33
1.7	Différentes techniques de reconnaissance. (a) : Mise en correspondance des pixels; (b) : Mise en correspondance des caractéristiques; (c) : Réseaux de neurones artificiels (RNA); (d) : Machines à vecteurs de supports (SVM).	40
1.8	(a) Ordinateur à mémoire partagée; (b) Ordinateur à mémoire distribuée.	41
1.9	(a) Ordinateur à mémoire distribuée avec un processeur de communication; (b) Ordinateur à mémoire distribuée dont les nœuds sont des architectures à mémoire partagée.	42
2.1	Dégradations possibles sur les documents scannés engendrés par la numérisation. (a) : éclairage non uniforme; (b) : bruit.	44
2.2	Architecture classique d'une chaîne d'extraction de texte.	45
2.3	La méthode <i>SauvolaMS$_{kx}$</i>	47
2.4	Extraction de texte difficile à partir d'un arrière-plan texturé.	49
2.5	Organigramme de la méthode d'extraction hybride de texte proposée : EHT.	52
2.6	Séparation du document en régions de texte et d'images : Algorithme PLA [56] de Tesseract [43] (Approche hybride).	53
2.7	Exemple de tabulations dans un document hétérogène.	54

2.8	Étapes de pré-traitement de l'algorithme PLA [56]. (a) : Image originale; (b) : Traits verticaux détectées; (c) : Régions d'images détectées; (d) : composantes connexes de texte filtrées.	55
2.9	Détection des traits horizontaux et verticaux : (a) : Image originale binaire, (b) : Fermeture (1), c : Ouverture (2), (d) : Fermeture - Ouverture (3); (e) : Ouverture horizontale (4); (f) : Ouverture verticale (5).	55
2.10	Recherche des colonnes. (a) : Tabulations candidates; (b) : Segments de Tabulations détectées; (c) : Tabulations connectés; (d) : Tabulations nettoyés.	56
2.11	Représentation des partitions de colonnes (CP) et des ensembles de partition de colonnes (CPset) dans un document hétérogène.	57
2.12	Recherche des régions et détermination de leurs ordre. (a) : Partitions de colonnes (CPs) candidates; (b) : colonnes détectées; (c) : chaînes de partitions typées; (d) : régions de textes et d'images.	58
2.13	Exemple de correction gamma. (a) : Image originale; (b) : Image avec correction gamma en utilisant la méthode GCM [71]	59
2.14	Correction gamma pour l'amélioration du contraste de l'image.	60
2.15	Exemple de génération des quatre matrices de co-occurrences. (a) : Déplacement et angle; (b) : Position des pixels voisins; (c) : Incrémentation des valeurs correspondantes dans la matrice de co-occurrence.	60
2.16	Classifications des images et les courbes de caractéristiques correspondantes : (a) : Images originales, (b) Courbes de caractéristiques de GCM; (c) : Courbes de caractéristiques échantillonnées. Courbe de contraste (rouge), Énergie (bleu), seuil d'Otsu (le vert) et la valeur gamma optimale (ligne verticale magenta).	69
2.17	Prédiction de la sous-rangée de valeurs gamma. Avec C = contraste, E = énergie, S_{Otsu} = Seuil Otsu, Ins : intervalle contenant gamma optimale.	70
2.18	La méthode de classification Kmeans.	72
2.19	La méthode CHK : Les phases globale et locale.	73
2.20	Cumul de la distorsion calculée à partir des résultats de binarisation des méthodes KG et CHK.	76
2.21	Localisation des caractères par Tesseract [43].	77
2.22	Comparaison visuelle de la qualité de binarisation. (a) : CHK; (b) : Lore et (c) : Tmms	79
2.23	Comparaison des qualité de binarisation d'un crop de document à partir de la base LRDE-DBD (a), entre : (b) CHK et (c) Sauvola Ms_{kx}	80
2.24	Binarisation avec les méthodes GCM [71] et CGCHK sur les images de texte de scènes naturelles issues de la base ICDAR. (a) : Image originale; (b) : GCM; (c) : CGCHK.	81
2.25	Comparaison de la qualité de CGCHK et de la méthode AGCM associée avec CHK. (a) : Images originales; (b) : Binarisation avec la correction gamma CG; (c) : Binarisation avec AGCM et CHK.	82
2.26	Comparaison des temps d'exécution de la méthode CG (bleu) et de la méthode AGCM (rouge) pour 113 images.	83
2.27	Évaluation visuelle de l'extraction de texte. (a). Simple et Multiple régions de documents scannées; (b) : Résultats CHK; (c) : Résultats GCM.	84
2.28	Évaluation visuelle de la méthode EHT : a. Document original, b. CHK et c. CGCHK, d. EHT	85

3.1	Reconnaissance de la lettre A avec CR-UCD.	89
3.2	Les deux vues horizontale et verticale d'un caractère. Les flèches indiquent le sens de parcours des pixels.	90
3.3	Dépendance entre l'échelle et la discontinuité des segments. (a) : Images en entrée; (b) : Contour Horizontal Extraît dans une première échelle; (c) : Contour Horizontal Extraît dans une échelle plus grande, (d) et (e) : Zoom correspondant aux segments extraits.	92
3.4	Position des segments dans le rectangle englobant le caractère. $x_0^{seg_i}$, $y_0^{seg_i}$ représentent les coordonnées des pixels de début x_0 , y_0 d'un segment seg_i $\forall i \in [0, Nb_{seg}]$. h et l sont respectivement la hauteur et la largeur du caractère.	92
3.5	Calcul de Barycentres (B) pour des lettres avec un nombre de segments différent. (a) : Pixels de début de segments; (b) : Calcul du barycentre du polygone construit, (c) : Barycentre résultant.	94
3.6	Invariance du Ratio par rapport à l'échelle des lettres A et J	95
3.7	Le descripteur unifié (<i>Unified Character Descriptor</i> (UCD)). (a) : Nombre de segments (NBS), (b). Position des segments (TMB, LMR); (c) : Barycentre ($\overline{Bx}$, $\overline{By}$); (d) : Ratio.	96
3.8	Typographie de l'alphabet latine.	97
3.9	La lettre A avec : (a) : <i>Sans Serif</i> et (b) : <i>Serif</i>	97
3.10	Reconnaissance de la fonte en se basant sur la séparation de caractères en blocs. Chaque numéro n dans un bloc désigne le nombre de segments non chevauchants avec la grille de séparation. Avec n est le nombre de segments dans un bloc, ep_{x_i} est l'épaisseur mesurée pour un segment en ordonnée x_i	98
3.11	Écart type (σ) des épaisseur des segments dans des caractères Multi-échelle et Multi-fonte. (a) : <i>Sans Serif</i> avec une variation modérée entre les segments fins et épais; (b) : <i>Serif</i> avec grande variation entre les segments fins et épais.	99
3.12	Principe de la mise en correspondance des caractéristiques.	100
3.13	Schéma fonctionnel de notre Système OCR proposé (CR-UCD).	100
3.14	Exemples de documents de la base LRDE-DBD.	103
3.15	Les templates multi-échelle et multi-fonte utilisées.	103
3.16	Cumul des taux de reconnaissance des systèmes Tesseract et CR-UCD sur 125 documents de la base LRDE-DBD	104
4.1	Contrainte temps réel de l'application de dématérialisation. T_{num} : Temps de numérisation du scanner, T_{ocr1} : Temps de reconnaissance de caractères de la méthode CR-UCD avant accélération de l'application.	109
4.2	Architecture interne d'un CPU.	111
4.3	Architecture interne d'un GPU.	112
4.4	Stratégie de parallélisation de la méthode EHT.	115
4.5	Stratégie de parallélisation de la méthode de correction gamma (CG).	116
4.6	Problème de la collision mémoire liée à la parallélisation du calcul de l'histogramme.	118
4.7	Stratégie de parallélisation du calcul de l'histogramme sur GPU.	119
4.8	Stratégie de parallélisation de la génération des matrices de co-occurrences sur GPU.	121

4.9	Indépendance du calcul des énergies et des contrastes à partir des quatre matrices de co-occurrences élémentaires.	122
4.10	Indépendance du calcul des énergies et des contrastes pour chaque matrice de co-occurrence élémentaire	123
4.11	Génération des caractéristiques de Haralick sur le GPU.	124
4.12	Exemple de la réduction parallèle, utilisée dans le calcul des caractéristiques de Haralick.	125
4.13	Implantation de la méthode de Correction Gamma (GC) sur un nœud de GPUs.	128
4.14	UV-Kmeans [97] : Première implantation Kmeans sur GPU. Avec A1 : valeur des pixels, A2 : nombre des pixels, Ci : centroids.	130
4.15	Stratégie de l'implantation parallèle de la méthode CHK sur GPU.	132
4.16	Parallélisme à Gros et fin grains dans l'implantation de CHK sur GPU.	133
4.17	Stratégie d'implantation de l'algorithme Kmeans sur GPU.	134
4.18	Optimisation du calcul des accumulateurs et des compteurs (resp $Accu_L$ et $Count_L$).	135
4.19	Kernel de la phase horizontale de dilatation.	136
4.20	Exemple de divergence de branches.	142
4.21	Évaluation du temps d'exécution de la méthode CHK sur CPU et GPU : (a) : Documents Non-scannés; (b) : Documents scannés.	143
4.22	Évaluation du temps d'exécution de la méthode CHK sur GPU et CPU en variant le contenu. (a) : Document original scanné; (b) : temps d'exécution sur GPU et sur CPU; (c) : Accélération du GPU par rapport au CPU.	144
4.23	Comparaison de la binarisation de la méthode EHT sur un crop de document scanné : a. GPU (F-measure = 98.5%); b. CPU (F-measure = 98.5%); Image de différence.	147
4.24	Traitement temps réel de l'OCR après accélération. T_{num} : Temps de numérisation du scanner, T_{ocr1} et T_{ocr2} : Temps de reconnaissance de caractères respectivement avant et après accélération.	148

Liste des tableaux

2.1	Caractéristiques de textures et de seuils Otsu calculés à partir de 100 valeurs gamma d'une image réunies dans un tableau d'instances Tab_{Ins}	64
2.2	Taux de reconnaissance des caractères extraits par CHK et par plusieurs méthodes de binarisation	79
2.3	Comparaison des méthodes de GCM et CGCHK basé sur OCR Précision pour les images de scènes de ICDAR	80
2.4	Comparaison des méthodes CHK et EHT en terme de précision de binarisation (Fmeasure) sur 125 images de la base LRDE-DBD (document de texte simple)	82
3.1	Précision de reconnaissance de caractère du descripteur UCD et celui de l'algorithme «i» en utilisant des modèles multi-fonte et multi-échelle	101
3.2	Précision OCR des descripteurs UCD et celui de l'algorithme «i» en utilisant un seul modèle de référence	102
3.3	Les seuils x-height utilisés	104
4.1	Analyse de la méthode CR-UCD sur CPU	110
4.2	Analyse de la méthode EHT sur CPU	114
4.3	Étude temporelle des différentes algorithmes de la méthode gamma sur CPU	117
4.4	Analyse du temps d'exécution des étapes de la méthode CHK sur CPU	129
4.5	Comparaison de Kmeans_Li avec différentes implantations Kmeans sur GPU dans la littérature	131
4.6	Les GPU employés	138
4.7	Évaluation du temps d'exécution de la méthode CG sur la base LRDE-DBD	139
4.8	Évaluation du temps d'exécution de Kmeans_GPU et Kmeans_Li sur 125 documents de la base LRDE-DBD	140
4.9	Évaluation du temps d'exécution de la méthode CHK sur CPU et GPU sur différentes tailles de blocs	141
4.10	Évaluation du temps d'exécution de la méthode CHK sur CPU et GPU sur des images bruitées	143
4.11	Temps de traitement moyen sur CPU et GPU de la méthode d'analyse de structure de documents pour une base de 125 documents hétérogènes de la base LRDE-DBD	145
4.12	Évaluation du temps réel de la méthode EHT proposée	145
4.13	Évaluation du temps réel sur GPU de la méthode EHT et trois méthodes de binarisation associées avec l'analyse structurale de document sur la base LRDE-DBD	146

4.14 Évaluation de l'accélération de la méthode CR-UCD	148
--	-----

Notations

x_c, y_c	Coordonnées du centre de gravité du contour
$x(n), y(n)$	Coordonnées d'un pixel n appartenant au contour
$D(n)$	Distance entre le centre de gravité avec chaque pixel n du contour
DFT	Transformé de Fourier Discrète (F_n)
F_D	Vecteur d'attributs de coefficients de Fourier
I	Image
$I(x, y)$	Pixel de coordonnées x, y dans l'image I
m	La couleur moyenne de l'image
IGT	Méthode de seuillage global itératif
S_{IGT}	Seuil de binarisation de la méthode IGT
$S_{Niblack}$	Seuil de binarisation de la méthode $Niblack$
$S_{Sauvola}$	Seuil de binarisation de la méthode $Sauvola$
S_{Otsu}	Seuil Otsu
k	Paramètre de contrôle pour les méthodes $Niblack, Sauvola$
w	Taille des fenêtres pour les méthodes de binarisation locales
σ	Ecart type des pixels dans une fenêtre
R	Plage dynamique de l'écart type σ
$S_{SauvolaMS_{kx}}$	Seuil de binarisation de la méthode $SauvolaMS_{kx}$
s	Echelle de sous échantillonnage allant de 1 à 4
I_s	Image I dans l'échelle s
T_s	Image qui contient les seuils de chaque pixel dans l'échelle s
B_s	Image binaire dans l'échelle s
S_s	Image contenant les composantes connexes sélectionnée dans l'échelle s
E_1	Image contenant composantes connexes sélectionnée dans tous les échelles
$E_1 - iZ$	Image générée après l'application de l'algorithme de l'influence de zones
PLA	Page Layout Analysis
CPs	Partitions de colonnes (Algorithme PLA)
$CPset$	Ensemble de partition de colonnes (Algorithme PLA)
EHT	Extraction Hybride de Texte
a	Angle associé à la matrice de co-occurences
d	Translation associée à la matrice de co-occurences
N_g	Nombre de niveaux de gris
Nb_{pixels}	Nombre de pixels dans l'image
p_1, p_2	Couple de niveaux de gris
$CMat_a$	Matrice de co-occurences d'angle a
γ	Gamma
γ_{opt}	Gamma optimale

moy	Moyenne de la matrice de co-occurences
Nb_{Images}	Nombre d'images correspondant aux 100 transformation gamma de l'image originale
Nb'_{Images}	Nombre d'images après optimisation de algorithmique de la correction gamma
E	Énergie de l'image
C	Contraste de l'image
E_a	Énergie élémentaire correspond à la matrice de co-occurences d'angle a
C_a	Contraste élémentaire correspond à la matrice de co-occurences d'angle a
σ_w^2	Variance intra-classe
$q_1(S_{Otsu})$	Nombre de pixels relatif a la 1 ^e classe
$q_2(S_{Otsu})$	Nombre de pixels relatif a la 2 ^e classe
$P(i)$	Proportion de pixels du niveau de gris i
$\mu_1^2(S_{Otsu})$	Moyennes de la 1 ^e classe
$\mu_2^2(S_{Otsu})$	Moyennes de la 2 ^e classe
Pe	Pic dans la courbe de l'énergie
Pc	Pic dans la courbe du contraste
Ps	Pic dans la courbe des seuil Otsu
Ins	Instance caractéristique d'une image
Tab_{Ins}	Tableau des Instances caractéristiques de 100 images
φ_{G0}	Centroid global d'indice 0
φ_{G1}	Centroid global d'indice 1
$\varphi_G(ite)$	Centroid global associé à l'itération ite
b_l	Indice des blocs dans l'image
φ centroid	
$Accu_G$	Accumulateur global
$Count_G$	Compteur global
$Accu_{Lbl}$	Accumulateur local
$Count_{Lbl}$	Compteur local
$Accu_{Lbl}^R$	Composante de la couleur Rouge de l'accumulateur local
$Accu_{Lbl}^G$	Composante de la couleur verte de l'accumulateur local
$Accu_{Lbl}^B$	Composante de la couleur Bleu de l'accumulateur local
φ_G^R	Composante de la couleur Rouge du centroid global
φ_G^G	Composante de la couleur verte du centroid global
φ_G^B	Composante de la couleur Bleu du centroid global
$distorsion$	Mesure de qualité de classification
$\varphi(I(x, y))$	Centroid associé au pixel $I(x, y)$
CHK	Classification hybride basée sur Kmeans
Nb_{itCHK}	Nombre d'itérations globales dans la méthode CHK
Nb_{blocs}	Nombre de blocs dans l'image
$Nb_{centroids}$	Nombre de centroids
$Nb_{itKmeans}$	Nombre d'itération de l'algorithme de classification Kmeans
Nb_{N_p}	Nombre de pixels noirs
Nb_{B_p}	Nombre de pixels blancs
HE	Contour droit dans la vue horizontale du caractère
VE	Contour droit dans la vue verticale du caractère
NBS	Nombre de segments
LMR	Position du segment (gauche, milieu, droit)

TMB	Position du segment (Haut, milieu, Bas)
l	Largeur du caractère
h	Hauteur du caractère
l_h	Largeur d'un bloc de du caractère dans la vue horizontale
h_h	Hauteur d'un bloc de du caractère dans la vue horizontale
l_v	Largeur d'un bloc de du caractère dans la vue verticale
h_v	Hauteur d'un bloc de du caractère dans la vue verticale
L	Largeur de l'image I
H	Hauteur de l'image I
S_y	Ordonnée du début de segment S
Bx, By	Coordonnées du barycentre
Nb_{Seg}	Nombre de segments d'un caractère
$Ratio$	Proportion du rectangle englobant du caractère
$Baseline$	Ligne de base
$Ligne\ x - height$	Ligne de hauteur
N_e	Nombre des épaisseurs mesurées
SDA_{V_1, V_2}	Somme des différences Absolues entre deux vecteurs
V	ensemble de vecteurs (descripteurs) dans un modèle
$Hist$	Histogramme niveau de gris
sub_{Hist}	Sous Histogramme niveau de gris relatif à un bloc de l'image
L_{Block}	Largeur d'un block dans un GPU
H_{Block}	Hauteur d'un block dans un GPU
$SIMD$	Single Instruction Multiple Data
$MIMD$	Multiple Instruction Multiple Data
$MIMD$	Multiple Instruction Multiple Data
$Nb_{ThreadBlock}$	Nombre de thread dans un block du GPU
Nb_{Blocks}	Nombre de blocs de threads
Nb_{Thread}	Nombre de Threads total
$threadBlock$	Indice du thread relatif à un block
$threadBlock_x$	l'abscisse du thread relatif à un block
$threadBlock_y$	l'ordonnée du thread relatif à un block
ind_{global}	Indice du thread relatif à un block
$thread_x$	l'abscisse du thread relatif à la grille du GPU
$thread_y$	l'ordonnée du thread relatif à la grille du GPU
$blockId$	Indice du block dans le GPU
$thread$	Indice absolu du thread dans le GPU
Occ	Tableau des Occurrences des valeurs de pixels
Tid	Tableau des indices des thread
IG	Index global
POS	Résultat de hachage de l'index global IG
$flag$	variable permettant la communication CPU et GPU
SP	Sommes partielles
$data$	valeur d'un élément de la matrice de co-occurrence
$Mem_{globale}$	Mémoire globale
$Mem_{partagée}$	Mémoire partagée
seg_i	Segment de caractère

Nb_{seg}	Nombre de segments dans un caractères
x_i, y_i	Coordonnées de pixels appartenant à un segment du caractère
$x_0^{seg_i}, y_0^{seg_i}$	Coordonnées de pixels de début de segments
h_0	Hauteur d'un caractères à une échelle 0
h_1	Hauteur d'un caractères à une échelle 1
l_0	Largeur d'un caractères à une échelle 0
l_1	Largeur d'un caractères à une échelle 1
<i>Baseline</i>	Ligne de base dans un mot
$x - height$	Mesure de hauteur d'un caractère
ep_{x_i}	Épaisseur à un abscisse x_i
Ne	Nombre de segments horizontales parcourues dans une image brute
Dim	Dimension d'un vecteur
Nb_{desc}	Nombre de descripteurs dans un modèle
SDA_{A-A}	Somme de différence des valeurs absolus entre deux descripteurs
GCM	Méthode de correction gamma
$CGCHK$	Correction gamma suivie de la méthode CHK

Introduction Générale

1 Cadre et motivations

La dématérialisation de documents est un processus clé et largement utilisé dans le monde de l'entreprise. Elle se définit comme une modalité d'échange ou de conservation de l'information sans support physique tel qu'un courrier papier, un chèque, un document papier issu d'un livre, d'un journal, ou d'un magazine, etc.

La technologie de la reconnaissance optique de caractères constitue un pilier fondamental pour la dématérialisation de documents. Une fois le document numérisé par un scanner, le texte est extrait de l'arrière-plan, puis, segmenté en caractères disjoints. Par la suite, ces caractères sont décrits et enfin reconnus. Dans les documents hétérogènes, les caractères peuvent être inclus dans des régions de texte avec un arrière-plan homogène, ou dans des graphiques, des photographies, des figures, des tables, etc.

Une approche unique pour extraire le texte quelque soit la zone du document, homogène ou complexe, rencontre des problèmes. Ainsi, chaque type de zone possède des caractéristiques différentes qui s'apparentent à des algorithmes de traitement d'images bien spécifiques. Ces algorithmes rencontrent eux même des difficultés pour isoler convenablement le texte à cause de sa complexité (textures, aspect dégradé de couleurs) ou de la mauvaise qualité du document (bruit, variation de la luminosité). Une fois le texte extrait, il est segmenté en caractères disjoints. Ensuite, les similarités de chaque lettre avec un ensemble de lettres d'une base de données sont calculées. La reconnaissance est effectuée en sélectionnant la plus petite des similitudes. Cette similarité nécessite une description efficace des caractères non influencée par des déformations possibles telles que la variation d'échelles, de fontes, etc.

Plusieurs difficultés subsistent concernant l'extraction du texte dans les documents hétérogènes. Nous nous concentrons sur l'isolation du texte dans le cas d'un arrière-plan texturé (Figure 1), ayant un aspect dégradé de couleurs et dans un document de mauvaise qualité (Figure 2). D'autre part, nous nous intéressons à l'automatisation de l'extraction du texte dans les documents hétérogènes (Figure 3).

La figure 1 présente un cas où le texte se situe sur un arrière-plan complexe. La variété de couleurs dans l'arrière-plan, ainsi que son mélange avec la couleur des caractères rend l'opération de l'extraction difficile même pour l'être humain. Dans ce cas, un seuillage global ou local ne permettra pas de binariser convenablement les caractères.



(a)



(b)

FIGURE 1: Extraction de texte à partir d'un arrière-plan complexe. (a) : Image comprenant un texte avec un arrière-plan couleur complexe ; (b) : Texte extrait par la méthode Sauvola [68].

L'analyse de textures est l'approche la plus efficace [168, 169, 170] permettant d'isoler le texte sur un arrière-plan complexe. Cependant, le faible contraste entre les caractères et l'arrière-plan affecte la qualité de l'extraction. Pour résoudre ce problème, nous proposons une approche d'analyse de texture tout en améliorant le contraste de l'image.

La complexité de la texture ne représente pas le seul problème. Aussi, la présence d'un arrière-plan de couleur dégradée, ou d'un document bruité avec une variation de la luminosité ajoute une complexité au traitement. Dans les méthodes globales [20, 21], un seul critère est appliqué pour classer tous les pixels du document en avant ou en arrière-plan. Ces méthodes supposent que le document traité est de bonne qualité et que les caractères sont bien isolés de l'arrière-plan [29].

En pratique, les documents scannés présentent souvent des problèmes tels que le faible contraste, la variation de la luminosité et le bruit dû à l'opération de la numérisation. Ceci altère la qualité de l'extraction textuelle. Ce qui nécessite l'utilisation d'une approche locale. Celle-ci applique plusieurs critères de classification, pour chaque groupe de pixels relativement proche dans l'espace euclidien. Cependant, les méthodes locales [93, 68] sont affectées

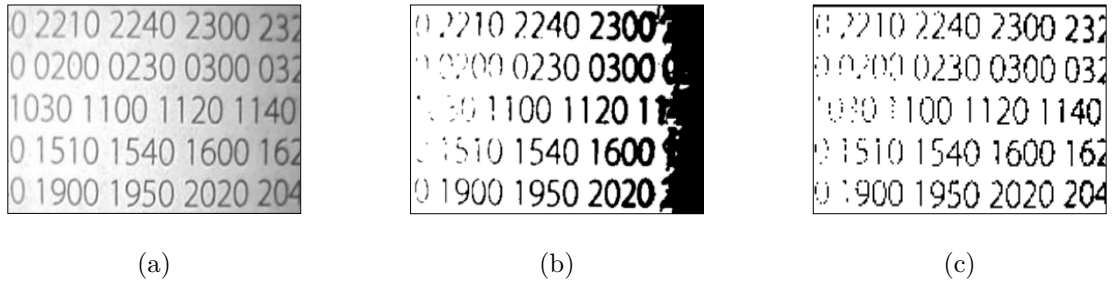


FIGURE 2: Extraction de texte avec une variation de la luminance. (a) Image comprenant un arrière-plan avec une variation de la luminance (b) Texte extrait par seuillage global (c) Texte extrait par seuillage local.

par le faible contraste entre le texte et l'arrière-plan.

Les méthodes globales [20, 21] sont rapides et efficaces quand elles sont appliquées sur un document avec une luminance uniforme. Sur les documents avec une luminance non uniforme, les méthodes locales [93, 68] sont plus performantes. Cependant, ces méthodes peuvent générer des artefacts dus aux traitements locaux. Pour résoudre le problème d'extraction de texte sur un arrière-plan dégradé, et dans un document bruité, nous adoptons une approche hybride, qui combine une approche locale basée sur la classification des pixels en avant-plan et arrière-plan avec une approche globale qui diminue la quantité de distorsion dans l'ensemble du document

En se basant sur nos approches d'extraction de texte dans les régions de texte et d'images, nous visons une approche générique pour séparer le texte de l'arrière-plan dans les documents hétérogènes. Les méthodes globales ou locales sont performantes sur des documents ayant en majorité un contenu textuel, mais elles ne sont pas adaptées à des images contenant des photos ou des dessins comme dans les documentations techniques ou dans les magazines.

La plupart de ces méthodes d'extraction de textes sont destinées à un seul type de document texte ou image. Par exemple, les méthodes globales sont efficaces pour des documents ne comprenant que du texte. Tandis que les méthodes locales améliorent les résultats pour les images contenant une variation de couleurs. Dans le cas d'un document hétérogène, il devient difficile de bien isoler à la fois les caractères dans les régions de texte et dans les images. Ainsi, les régions de texte et d'images peuvent être marquées manuellement par l'utilisateur avant l'extraction du texte. Ce qui représente une tâche fastidieuse lorsqu'on manipule une grande quantité de documents.

Dans l'optique d'une automatisation totale de l'extraction du texte dans les documents hétérogènes, nous proposons une approche adaptative qui sélectionne le traitement à appliquer selon la nature des régions homogènes ou complexes.

D'autre part nous améliorons l'extraction de texte dans chaque type de document. Une approche hybride est proposée pour extraire convenablement le texte dans les documents scannés affichant une variation de la luminosité et/ou un dégradé. Par ailleurs, nous adop-

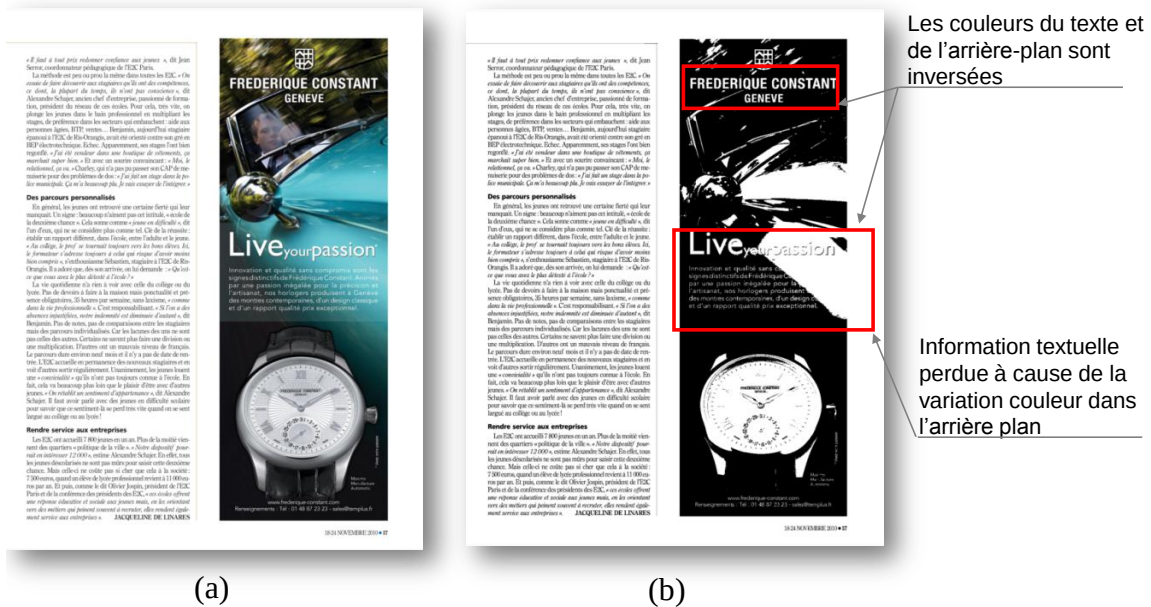


FIGURE 3: Exemple d'extraction de texte erronée dans un document hétérogène. (a) Document hétérogène couleur composé de régions de textes et d'images ; (b) Texte extrait par la méthode Sauvola Ms_{kx} [69].

tons une approche basée sur l'analyse de texture couplée avec une approche hybride pour l'extraction de texte dans les images et les graphiques. Le texte extrait est par la suite segmenté en caractères isolés. Puis, les caractères sont décrits sous forme d'un descripteur pour assurer leur identification.

La description d'un caractère consiste à le définir avec des attributs qui lui sont spécifiques et discriminants des autres caractères.

Dans ce contexte, deux approches existent [167] : Description par des pixels des matrices de caractères et description par des caractéristiques tels que la surface, la largeur, la hauteur, le nombre de segments, etc.

Dans la première approche, pour reconnaître le caractère, il suffit de comparer tous ses pixels à ceux de chaque caractère dans une base de données. Ce traitement est appelé mise en correspondance des matrices. Cette approche permet une reconnaissance de bas niveau des caractères en exploitant toute l'information utile dans l'image. La figure 4 illustre la lettre 'A' dans une grille de pixels selon une variation de fonte et de bruit. L'approche de mise en correspondance des matrices de pixels est fortement sensible à la variation de la fonte (Figure 4.a-b) et au bruit (Figure 4.b-c).

Contrairement à la première approche, l'extraction des caractéristiques emploie des informations de plus haut niveau. Ainsi, un caractère peut être défini par ses lignes : creux, courbures, segments, etc. De nombreux algorithmes ont été utilisés [122, 160] pour détecter

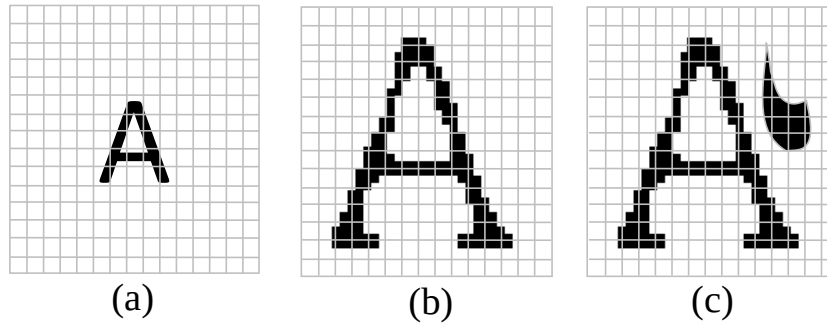


FIGURE 4: Description par matrices de pixels de la même lettre 'A'. (a) : Fonte Arial; (b) : Fonte Times; (c) : Fonte Times avec du bruit.

ces caractéristiques tel que : la détection de contours, la transformation en squelette, les transformations de Fourier ou d'ondelettes, la variation des moments ou des propriétés topologiques de la forme comme le nombre de segments, le nombre de trous et la connexité, etc.

Hormis la complexité de ces algorithmes, la difficulté réside dans le choix des caractéristiques à employer pour définir les caractères. Des caractéristiques de nature redondante ou bien sensible à la déformation de la taille et de la fonte augmentent la complexité du calcul sans pour autant augmenter le degré de la discrimination entre les caractères et ainsi le taux de reconnaissance. Il est important alors, de bien choisir les caractéristiques de façon qu'elles soient les plus invariantes au changement de la fonte et de l'échelle.

Dans une application de dématérialisation, la reconnaissance de caractères dans un document numérisé doit être exécutée durant la numérisation du document qui suit. Ce qui implique un traitement en flot de données.

De plus en plus de constructeurs fournissent des logiciels de reconnaissance de caractères en plus du matériel de la numérisation (scanner). Les logiciels, exécutés dans les scanners ne sont pas en mesure d'assurer un traitement en temps réel vu la complexité des algorithmes mis en œuvre.

Ce contexte a conduit à l'exploitation des architectures de plus en plus évoluées tels que les processeurs CPU multi-cœur. Ces architectures permettent d'accélérer considérablement les algorithmes grâce aux différents niveaux de parallélismes qu'elles offrent (parallélisme d'instructions, parallélisme de données et parallélisme des threads). Ceci a permis de concevoir des algorithmes de reconnaissance de caractères plus sophistiqués et complexes.

De nos jours, les processeurs graphiques « Graphics Processing Unit » (GPU), circuits initialement conçus pour l'affichage graphique ont connu une évolution importante. Ces circuits ont progressé vers des circuits dédiés au calcul hautement parallèle et surtout adapté aux algorithmes de traitement d'images. Si la rapidité de calcul théorique des GPU surpasse 15x celle des CPU, les CPU restent plus efficaces sur les parties séquentielles du code. Les algorithmes que nous concevons sont écrits dès le départ d'une façon adéquate pour les GPU. Le CPU exécutera les portions du code de nature séquentielle et commandera les traitements

sur le GPU.

2 Organisation de la thèse et contributions

Dans cette thèse nous nous intéressons aux étapes de l'extraction et de la description de caractères dans les documents hétérogènes. Dans ce contexte, nous proposons deux méthodes originales d'extraction et de description de caractères. Nous nous intéressons également à l'optimisation des temps de calcul du système OCR associé. Une étude et optimisation des algorithmes proposés sont effectuées et une parallélisation sur GPU des traitements les plus coûteux est assurée.

Ce manuscrit est organisé en quatre chapitres :

Dans le premier chapitre, nous présentons les systèmes de reconnaissance de caractères pour le traitement des documents hétérogènes. De plus, les trois approches d'extraction, de description et de reconnaissance des caractères sont étudiées. Nous relevons ainsi les approches les mieux adaptées pour traiter les documents hétérogènes.

Le second chapitre de ce manuscrit est dédié à la présentation de notre proposition d'extraction hybride de texte dans les documents hétérogènes. Dans cette méthode, un processus d'analyse d'image est employé suivi d'une classification des régions du document en régions d'image et de texte. Un pré-traitement est effectué sur la texture des régions d'images en utilisant la correction Gamma adéquate. Ensuite, l'information textuelle est extraite en utilisant une méthode de classification hybride. Cette méthode améliore la qualité de séparation texte/fond, tout en minimisant la quantité de distorsion dans les documents scannés. D'après nos expérimentations notre méthode d'extraction de texte permet d'atteindre un taux de reconnaissance de caractères de 98,5% sur des documents hétérogènes scannés issus de la base LRDE-DBD [69].

Le troisième chapitre représente un nouveau descripteur de caractères basé sur l'étude de leur structure. Il emploie des caractéristiques issues de l'unification de deux descripteurs dérivés de la projection horizontale et verticale des caractères permettant une discrimination plus efficace. Le descripteur, baptisé : « Unified Character Descriptor » (UCD) supporte la reconnaissance des caractères alphanumériques et multi-échelles. Il est validé à travers une nouvelle méthode de reconnaissance de caractères que nous appelons : « Character Recognition based on UCD » (CR-UCD). Ainsi, le descripteur proposé, permet d'avoir une reconnaissance de caractères de 100% pour une fonte et une taille données.

Nous proposons, dans le quatrième chapitre, une stratégie de parallélisation de l'extraction hybride de texte dans les documents hétérogènes. Le processeur graphique GPU a été utilisé comme cible d'implantation. Notre mise en œuvre, combine les stratégies de parallélisation à fin et à gros grains. Les algorithmes de classification Kmeans [66], de binarisation Otsu [65], et de calcul des caractéristiques de Haralick [63] ont été accélérés. En outre, les coûts de communication CPU-GPU sont réduits et une bonne gestion mémoire est assurée. L'extraction hybride de texte a été accélérée 6.74x sur GPU par rapport à son exécution sur

CPU permettant d'accélérer le système de reconnaissance de caractères CR-UCD conçu.

Chapitre 1

Reconnaissance de caractères dans les documents hétérogènes

Dans ce chapitre, nous présentons les systèmes de reconnaissance de caractères pour le traitement des documents hétérogènes. Nous relevons ainsi les approches les mieux adaptées pour extraire, décrire et reconnaître le texte dans les documents hétérogènes.

Sommaire

1	Introduction	30
2	Extraction de texte dans les documents hétérogènes	32
2.1	Méthodes basées sur l'analyse de textures	33
2.2	Méthodes basées sur l'analyse des Régions	35
2.3	Méthodes basées sur l'analyse des contours	36
3	Description des caractères	36
3.1	Approche stochastique	37
3.2	Approche statistique	37
3.3	Approche topologique	38
3.4	Approche structurelle	39
4	Reconnaissance des caractères	39
5	Architectures parallèles	40
6	Conclusion	41

1 Introduction

La dématérialisation a pour objectif de gérer de façon totalement électronique les documents papier qui transitent au sein d'un réseau d'entreprise, établissement universitaire ou autres [90] [91]. Pour ce faire, les documents papier sont numérisés et transformés en images numériques. Sur ces images, la reconnaissance optique de caractères (OCR) est appliquée pour identifier l'information textuelle. La figure 1.1 illustre un exemple d'application de dématérialisation.

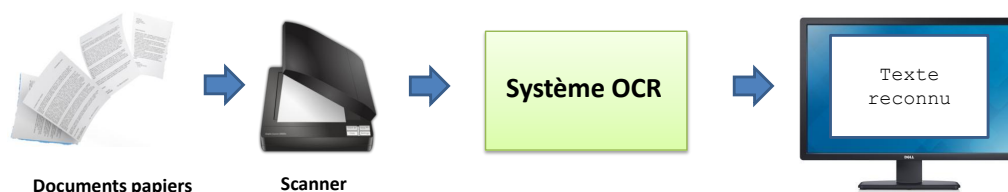


FIGURE 1.1: Principe de la dématérialisation de documents

Les systèmes OCR sont constitués principalement par trois étapes fondamentales : L'extraction de texte, la description et la reconnaissance des caractères [88]. La Figure 1.2 illustre un exemple classique de système OCR. L'extraction de texte consiste à séparer et à localiser les caractères à partir de l'arrière-plan. Ensuite, ces caractères sont décrits selon leurs formes, topologies, etc. Enfin, la reconnaissance consiste à identifier ces caractères. La reconnaissance pourra se servir d'une base de descripteurs prédéfinis et/ou d'un dictionnaire pour améliorer le taux de reconnaissance [92].

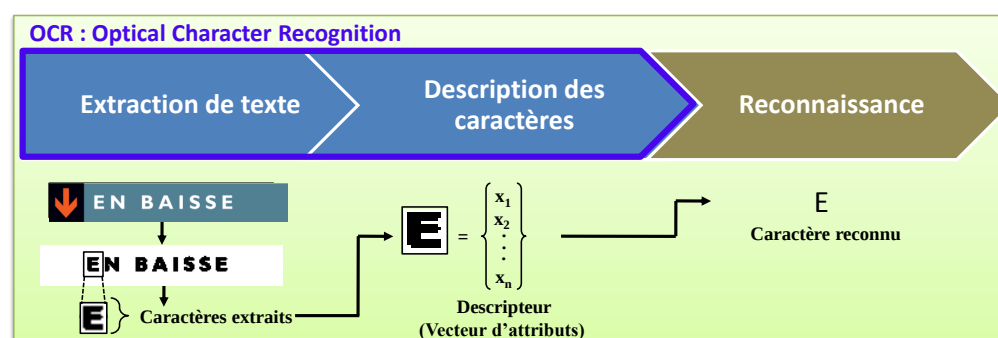


FIGURE 1.2: Architecture classique d'un Système OCR.

L'information textuelle intègre différents types de documents dématérialisés tels que les chèques, les documents administratifs, les photographies, etc. Cette diversité de documents peut se présenter dans un document unique hétérogène tel que les magazines, les journaux, etc (Figure 1.3).



FIGURE 1.3: Exemples de sources d'informations textuelles : Magazines, Journaux.

Les documents de texte numérisés sont classés principalement selon trois types : les documents avec du texte simple, les documents de scènes naturelles et les documents avec du texte de légendes [78]. Un document de texte simple (Figure 1.4a) possède un arrière-plan homogène, dans lequel les caractères sont distingués facilement du fond. Un document de scène naturelle (Figure 1.4b) est une image qui contient généralement un texte déformé sur un arrière-plan complexe. Cette complexité résulte d'une variété de textures ou de couleurs derrière le texte ce qui le rend moins lisible. Le texte de légende 1.4c accompagne généralement les images de scènes naturelles, il est ajouté manuellement pour commenter l'image. Dans ce type de documents les caractères sont lisibles car ils se situent sur un arrière-plan homogène.

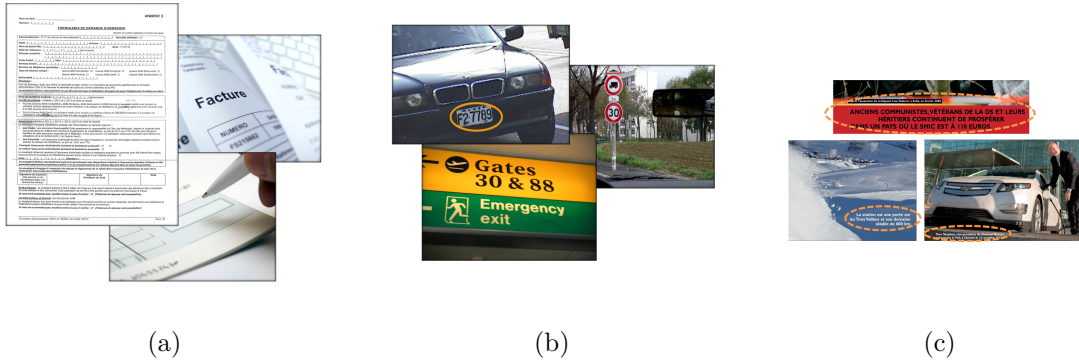


FIGURE 1.4: Exemples de sources d'informations textuelles. (a) : Chèques; Documents administratifs; (b) : Photographies, (c) : Légendes d'images.

En outre, des documents avec des images de contenu hétérogène peuvent être considérés comme un quatrième type de documents. Ils sont généralement nommés documents hétérogènes et comprennent au moins deux types de documents mentionnés ci-dessus que nous appellerons régions de textes et d'images dans le reste des chapitres. La figure 1.5 montre un exemple de document hétérogène. La présence des régions de texte et d'images dans les documents hétérogènes, implique des approches différentes et adaptées pour l'extraction, la

description et l'identification efficaces des caractères. En effet, la complexité des documents, rend la reconnaissance de caractères difficile surtout pour l'extraction du texte et la description des caractères en multi-fonte et multi-échelle. Plusieurs méthodes ont été proposées dans la littérature pour assurer une extraction de texte [163, 164, 71] et une description de caractères [158, 155, 145] robustes qui ciblent des types de documents spécifiques tel que les documents purement textuels, et les images de scènes naturelles.



FIGURE 1.5: Exemple de document hétérogène.

Dans ce chapitre, nous présentons les différentes étapes de traitement des systèmes de reconnaissance optique de caractères dans les documents hétérogènes. Les approches de l'étape d'extraction de texte sont exposées dans la section 2. Dans la section 3, les approches employées pour la description de caractères sont décrites. Les différentes méthodologies de l'étape de reconnaissance de caractères sont détaillées dans la section 4. Enfin, nous présentons succinctement quelques types d'architectures parallèles dans la section 5.

2 Extraction de texte dans les documents hétérogènes

Dans les documents hétérogènes, les caractères sont inclus dans des régions de texte avec un arrière-plan homogène, ou bien à l'intérieur des graphiques, des figures et dans les images de scènes naturelles qui possèdent généralement une texture complexe. De ce fait, l'extraction

de texte représente une tâche difficile à accomplir étant données les variations du texte dues aux différences de leurs tailles, de leurs styles, de leurs orientations et de leurs alignements, ainsi qu'à la complexité de l'arrière-plan de l'image.

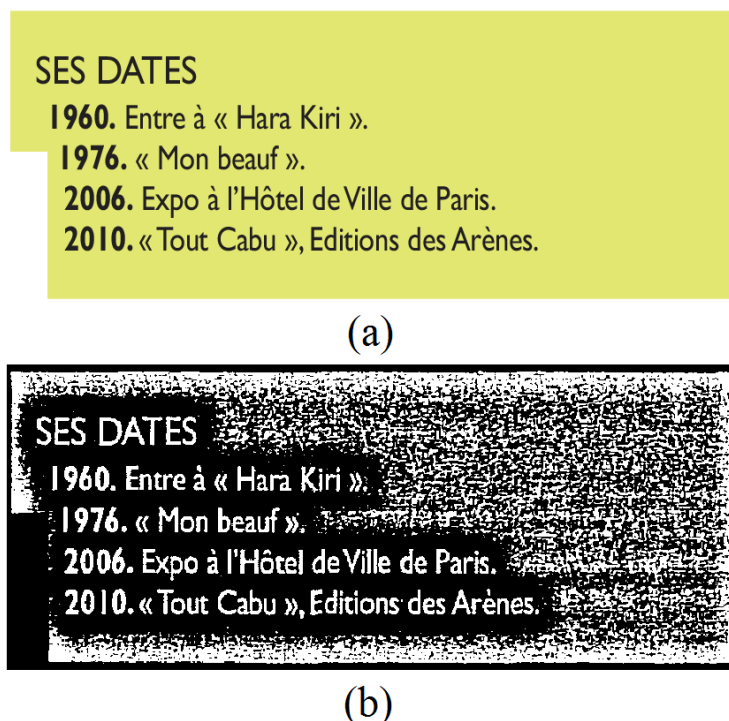


FIGURE 1.6: Extraction de texte dans un document scanné. (a) : Image Originale. (b) : Texte extrait.

On trouve dans la littérature les méthodes d'extraction de texte qui se basent sur l'analyse de textures, de régions et de contours. En effet, chaque approche est adéquate pour traiter un type de document spécifique. Ainsi, les méthodes d'extraction basées sur l'analyse de textures sont utilisées dans le cas d'un texte avec un arrière-plan complexe. Les méthodes d'extraction basée sur l'analyse de régions et de contours sont adaptées à un texte avec un arrière-plan homogène.

2.1 Méthodes basées sur l'analyse de textures

Une texture est définie par la répétition spatiale d'un même motif dans différentes directions de l'espace. La notion de texture est utilisée pour traduire un aspect homogène de la surface d'un objet sur une image. La texture se manifeste donc par une information visuelle qui permet de la décrire généralement à l'aide des adjectifs comme régulière, irrégulière, etc.

En pratique, on distingue deux grandes classes de textures, qui correspondent à deux niveaux de perception : Les macrotextures qui présentent un aspect régulier, sous formes de motifs répétitifs spatialement placés selon une règle précise. Les microtextures présentant des primitives «microscopiques» distribuées de manière aléatoire.

Dans les documents hétérogènes, la texture du texte est différente de celle de l'arrière-plan. Les critères visuels qui ont été retenus pour la texture sont : le contraste, la granularité, l'orientation, la forme, la finesse, la régularité et la rugosité. Dans ce contexte une multitude de méthodes structurales, spatio-fréquentielles et statistiques ont été proposées dans la littérature.

2.1.1 Les méthodes structurales

Les méthodes structurales [168, 169] tiennent compte de l'information structurale et contextuelle d'une forme et sont particulièrement bien adaptées aux textures macroscopiques. Une description structurale d'une texture implique la recherche des motifs élémentaires, leur description, puis, la détermination des règles conditionnant leur position. Les deux structures les plus importantes sont les structures de graphe et les structures syntaxiques. L'application des méthodes structurales est limitée à des textures simples et régulières [165].

2.1.2 Les méthodes spatio-fréquentielles

Le cerveau humain est capable d'analyser les images selon leur contenu fréquentiel [170]. Les méthodes faisant appel à un filtrage fréquentiel de l'image ont ainsi fait l'objet de bon nombre d'études en analyse texturale. La plupart de ces méthodes extraits des attributs à partir de l'image transformée par filtrage. Ces attributs seront exploités ensuite dans des algorithmes de classification ou de segmentation. Les deux approches par filtrage les plus couramment employées sont : les filtres de Gabor et la transformée en ondelettes.

L'avantage des représentations spatio-fréquentielles c'est qu'elles permettent de caractériser la texture à différentes échelles. Cependant l'inconvénient de ces méthodes est le réglage des paramètres des filtres. De plus, pour caractériser certaines textures, il est nécessaire d'avoir une fenêtre d'analyse assez grande, ce qui peut impliquer un temps de calcul élevé.

2.1.3 Les méthodes statistiques

Dans les méthodes statistiques, des paramètres statistiques sont estimés pour chaque pixel de l'image. Suivant la modalité des images à étudier, la signature la plus discriminante de la texture est à rechercher soit dans des méthodes qui exploitent directement les propriétés statistiques de la texture (matrices de co-occurrences, matrice de longueurs de plages, matrice de voisinage, fonction d'autocorrelation, modèle de Markov, modèle autorégressif, modèles issues de la morphologie mathématique), soit dans des méthodes qui exploitent les propriétés statistiques à partir d'un plan transformé dans lequel on réécrit l'image de texture (densité spectrale, méthode des extremas locaux, méthodes de transformation de Fourier) [62].

L'approche de l'extraction de texte basée sur l'analyse de textures est adaptée aux images présentant un texte sur un fond complexe. D'autre part, l'approche d'analyse des régions est mieux adaptée pour les régions de texte avec un arrière-plan homogène.

2.2 Méthodes basées sur l'analyse des Régions

Les méthodes basées sur l'analyse des régions s'appuient sur la technique d'analyse de composantes connexes. Ce type d'analyse est basé sur l'étude géométrique des contours qui composent les caractères [16]. Par exemple, on détecte dans [3] les contours verticaux et horizontaux de l'image suivie de quelques opérations de dilatation. Cette méthode ne réussit pas à détecter un texte incliné. La méthode de Zhong [18] utilise l'information des contours pour détecter les lignes textuelles. C'est une méthode efficace pour les images couleur complexe mais elle est restreinte à une taille de caractères donnée. C'est une tâche simple à implanter qui détecte rapidement le texte. Par contre, elle n'est pas robuste lorsque l'arrière-plan est complexe. L'approche par analyse des composantes connexes est adaptée à l'extraction d'un texte dans un arrière-plan homogène. Cette technique nécessite la binarisation préalable du texte en utilisant le seuillage ou la classification des pixels en fond/forme.

Le seuillage consiste à convertir une image couleur ou niveau de gris en une image binaire fond/forme. Les méthodes basées sur le seuillage génèrent l'image binaire où chaque pixel est comparé à un seuil donné. Si la valeur du pixel est inférieure à la valeur du seuil alors le pixel est mis à '0' (noir) sinon il est mis à '1' (blanc). Le problème est que le niveau de gris dans l'image n'est pas uniforme ainsi il est difficile de comparer les niveaux de gris des pixels à un seuil unique pour distinguer convenablement les pixels noirs appartenant à la forme des pixels noirs appartenant au fond. Une façon classique pour résoudre ce problème consiste à calculer de manière dynamique les seuils de séparation Fond/Forme. Les méthodes de binarisation de documents se répartissent en deux grandes familles : les méthodes globales et locales. Les méthodes globales appliquent un traitement identique à tous les pixels du document et les méthodes locales s'adaptent au contexte de chaque pixel en lui calculant un seuil approprié. Ces méthodes sont détaillées dans le chapitre 2.

La classification est autre technique de seuillage. Pour extraire le texte, les méthodes de classification divisent les pixels en deux classes : classe des pixels appartenant au texte et classe des pixels appartenant à l'arrière-plan. L'étape de classification des pixels peut être menée de manière supervisée ou non supervisée. Les méthodes supervisées sont des méthodes dans lesquelles les classes sont connues *a priori* avant d'identifier les pixels de l'image. Elles demandent une première phase d'apprentissage sur l'échantillon représentatif dans le but d'apprendre les caractéristiques de chaque classe et une deuxième phase pour décider de l'appartenance des pixels aux classes correspondantes. Les données segmentées de l'ensemble d'apprentissage proviennent d'un étiquetage manuel des images ou des régions d'intérêt en C classes par un ou plusieurs experts. Chaque classe se voit donc affecter un ensemble d'apprentissage, et les données de l'ensemble de test sont segmentées en fonction de l'ensemble d'apprentissage. Parmi ces méthodes on peut citer : la segmentation Bayésienne, la segmentation par les champs de Markov [145], les réseaux de neurones artificiels, etc.

L'intérêt des méthodes de classification non supervisées est qu'elles ne nécessitent aucune base d'apprentissage et par la suite aucune tâche préalable d'étiquetage manuel n'est requise. Les algorithmes non supervisés les plus répandus tendent à minimiser une fonction de coût. Chaque pixel est assigné à la classe la plus proche dépendant de sa distance aux noyaux (centroïdes) des classes. Le centroïde d'une classe étant un point connu dont l'appartenance à cette classe est garantie, il représente souvent la moyenne de ses pixels.

Nous distinguons plusieurs méthodes de classification [65] : Par exemple, Otsu [65] est

une technique très populaire [24], qui applique un seuil automatique qui réduit la variance inter-classes entre le texte et l'arrière-plan. Dans cette thèse on s'intéressera aux méthodes non-supervisées.

2.3 Méthodes basées sur l'analyse des contours

Les contours sont des caractéristiques robustes pour l'extraction du texte. L'épaisseur du contour, sa densité, le contraste entre le texte et l'arrière-plan sont les éléments qui distinguent le texte dans un document. En effet, les contours du texte sont identifiés et fusionnés, filtrés heuristiquement pour supprimer les éléments non textuels. En général, l'opérateur Canny [159] est utilisé pour détecter les contours, et un opérateur de morphologie mathématique est utilisé pour fusionner les contours textuels. Dans ce contexte plusieurs méthodes ont été proposées dans la littérature [160, 161, 162]. Cependant l'extraction de texte basée sur les contours est sensible au bruit et au faible contraste entre le texte et l'arrière-plan.

Une fois le texte extrait, ces caractères sont décrits afin de permettre leurs identifications dans l'étape de reconnaissance finale. Dans la suite nous abordons les différentes approches de description de caractères.

3 Description des caractères

Pour décrire les caractères extraits, on utilise un vecteur de primitives pertinentes qui discrimine les caractères les uns des autres. En effet, un mauvais choix des primitives influence négativement et considérablement les résultats même si on utilise une méthode de reconnaissance performante.

La description des caractères est basée généralement sur les matrices de pixels (*Template Description* : TD) ou sur les caractéristiques des caractères (*Feature Description* : FD). Les méthodes FD sont plus performantes que les méthodes TD dans la description des caractères [50].

Dans les méthodes TD [85, 47], on décrit les caractères par l'ensemble de leurs pixels. Malgré la simplicité de ces méthodes, elles ne sont pas robustes pour un texte bruité, le bruit pouvant être assimilé à une partie du caractère. De plus les méthodes TD sont capables de reconnaître uniquement les caractères de mêmes tailles et positions dans des rectangles englobants. Ces méthodes sont complexes en termes de temps de calculs. Moins complexes, les méthodes FD [48, 49, 50] décrivent les caractères en se basant sur quelques caractéristiques spécifiques qui désignent des particularités détectables dans les caractères comme les concavités, la longueur des contours, le nombre de segments, etc.

On référence diverses approches de description de caractères utilisées pour la reconnaissance de texte : stochastiques, statistiques, topologiques et structurelles.

3.1 Approche stochastique

L'approche stochastique est essentiellement utilisée pour la reconnaissance de l'écriture manuscrite. Elle permet de modéliser l'écriture en tenant compte des états rencontrés par le stylet tel que lever de crayon, l'origine et la fin de la lettre, l'intensité de l'ancre, etc. L'information temporelle et la forme du tracé permettent de coder l'écriture. Le caractère est modélisé sous forme de graphe d'états, où chaque état représente une observation et par les relations entre ces différents états. La technique utilisée est essentiellement basée sur l'utilisation de chaînes de Markov [145] où chaque caractère est décrit à l'aide d'une séquence de primitives et une séquence d'états. Les chaînes markoviennes cachées sont employées pour l'extraction des primitives de la forme. Une étape de classification est nécessaire. La plus généralement utilisée pour l'approche stochastique est l'algorithme de Viterbi et ses variantes [6]. La reconnaissance stochastique consiste à rechercher dans un graphe d'états le chemin le plus court décrivant les éléments observés. La technique utilisée est essentiellement basée sur l'utilisation de chaînes de Markov où chaque caractère est décrit à l'aide d'une séquence de primitives et une séquence d'états. Cependant, dans le cas de l'écriture imprimée, cette approche n'apporte que peu d'intérêt car il y a absence de l'aspect temporelle [158].

3.2 Approche statistique

L'approche statistique ou approche globale utilise des mesures faites sur le caractère afin de le coder sous forme de vecteur. Ce vecteur définit le caractère dans un nouvel espace de représentation. Cette approche consiste à construire un classifieur permettant de distinguer les différentes classes à l'intérieur de cet espace de représentation.

L'objet des méthodes de description statistique est de pouvoir décrire des comportements spécifiques à partir de modèles simples à utiliser. Parmi les différents descripteurs de forme, il y a les descripteurs de Fourier et de Zernike qui décrivent respectivement le contour et la forme.

Ce sont des descripteurs représentant un vecteur d'attributs invariants selon les principales transformations (homothétie, rotation). La première méthode basée sur la transformation de Fourier permet de décrire le contour externe du caractère alors que la méthode basée sur les moments de Zernike permet de décrire la forme du caractère en tant que région.

3.2.1 Description du contour : descripteurs de Fourier

Le principe des descripteurs de Fourier est d'échantillonner le contour sur un nombre de points préfixé afin de calculer la Transformée de Fourier Discrète (DFT). Les coefficients fréquentiels normalisés ainsi obtenus forment les descripteurs. Il est alors possible de calculer des descripteurs selon la courbure ou selon la distance au centre de gravité. D.Zhang et G.Lu ont montré que cette seconde solution est plus performante [155]. Le contour étant normalisé à N points, la distance D au centre de gravité de coordonnées (x_c, y_c) est établie pour chaque point n du contour de coordonnées $x(n), y(n)$, avec $n \in [0, N - 1]$ (équation 1.1).

$$D(n) = \sqrt{(x(n) - x_c)^2 + (y(n) - y_c)^2}, n = 0, 1, \dots, N - 1 \quad (1.1)$$

La Transformation de Fourier Discrète (DFT) est donnée par l'équation 1.2. A partir de la DFT, seul le module $|F_n|$ est conservé vu son invariance à la rotation.

$$F_n = \frac{1}{N} \sum_{i=0}^{N-1} D(n) \exp\left(\frac{-j2\pi in}{N}\right) \quad (1.2)$$

Le vecteur d'attributs F_D inclut les coefficients $|F_n|$ normalisés par $|F_0|$ pour s'affranchir des changements d'échelle (équation 1.3).

$$F_D = \left\{ \frac{|F_1|}{|F_0|}, \frac{|F_2|}{|F_0|}, \dots, \frac{|F_{\frac{N}{2}}|}{|F_0|} \right\} \quad (1.3)$$

3.2.2 Description de la forme : moments de Zernike

Ces moments de Zernike A_{nm} correspondent à la projection d'une fonction $f(x, y)$ décrivant une image sur une base ZP de fonctions orthogonales $V_{nm}(x, y)$ (eq.1.4)

$$A_{nm} = \frac{n+1}{\pi} \sum_{(x,y)} V_{nm}(x, y)^* f(x, y) \quad (1.4)$$

où $*$ désigne le conjugué du complexe. La base ZP est définie sur le cercle unité par :

$$ZP = \{V_{nm}(x, y) | (x^2 + y^2) < 1\} \quad (1.5)$$

Le polynôme complexe V_{nm} d'ordre de moment n et de répétition m tel que $n - |m| \geq 0$ (eq.1.6).

$$V_{nm}(x, y) = R_{nm}(r) e^{jm\theta} \quad (1.6)$$

avec $R_{nm}(r)$ est le polynôme orthogonal radiale ; définie par :

$$R_{nm}(r) = \sum_{s=0}^{(n-|m|)/2} (-1)^s \frac{(n-s)! r^{n-2s}}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n+|m|}{2} - s\right)!} \quad (1.7)$$

Les moments sont invariants par rotation, translation et changement d'échelle (après normalisation de la taille de la forme). De plus, grâce à l'exploitation d'une base de fonctions orthogonales, ces moments sont peu corrélés [156, 157].

3.3 Approche topologique

L'approche topologique de description de caractères se base sur l'étude des propriétés de l'espace, de point de vue qualitatif. Concrètement, différentes caractéristiques topologiques sont intéressantes tel que le nombre de trous, l'évaluation des concavités, les mesures de pentes, de courbures, de la longueur et de l'épaisseur des traits, etc. Dans leur étude M. Maged et M. Fahmy présentent l'extraction de caractéristiques géométriques d'un caractère

arabe [66] en appliquant la technique des histogrammes directionnels sur le contour de l'image du caractère.

3.4 Approche structurelle

Les primitives structurelles ressemblent aux primitives topologiques [12, 13]. La différence est qu'elles sont généralement extraites non pas de l'image brute, mais à partir du squelette ou du contour du caractère. Ainsi, on ne parle plus de trous, mais de boucles ou de cycles dans une représentation filiforme du caractère. D'un point de vue général, la classification structurelle utilise une décomposition du caractère en objets primitifs et permet de décrire une partie de l'organisation de ces objets les uns par rapport aux autres. Plusieurs types de représentations permettent de stocker ces structures, soit au travers de cartes topologiques, soit par l'intermédiaire de graphes d'adjacences.

Khorsheed [153] présente une technique d'extraction de caractéristiques structurelles de l'écriture arabe cursive. Dans cette étude, le squelette de l'image binaire du mot est décomposé en un nombre de segments selon un certain ordre. Chaque segment est transformé en un vecteur de caractéristiques. Les caractéristiques recherchées sont les courbatures des segments, leurs longueurs relatives, et la position du segment par rapport au centre de gravité du squelette. Les résultats de cette préparation sont appliqués pour l'entraînement des Modèles Cachés de Markov (HMM) pour opérer la reconnaissance [145].

Les méthodes structurelles sont performantes pour la description des caractères et offrent une représentation moins sensible aux distorsions, variation d'échelles, etc. Dans la suite nous abordons les différentes approches d'identification de caractères dans les documents hétérogènes.

4 Reconnaissance des caractères

Dans un système OCR, l'étape de la reconnaissance consiste à identifier le caractère préalablement décrit. L'idée principale de la classification est d'attribuer un caractère non connu à une classe prédéfinie à partir de la description du caractère. Plusieurs méthodes de reconnaissance existent, parmi lesquels on trouve les machines à vecteurs de supports (SVM), les Modèles de Markov cachés [145], la mise en correspondance des pixels ou de caractéristiques, les réseaux de neurones artificiels (RNA), etc (Figure 1.7).

Dans les méthodes basées sur les réseaux de neurones artificiels [5], la reconnaissance est généralement composée de deux phases : l'apprentissage des caractères et leurs identification [166]. L'apprentissage permet au système d'élaborer sa bibliothèque de caractéristiques. La méthode d'apprentissage la plus triviale consiste à mémoriser différents vecteurs représentant chaque caractère. Dans le cas de reconnaissance multi-fonte, chaque fonte doit être préalablement apprise par le système. L'identification consiste à déterminer, à partir des caractéristiques apprises, la classe d'appartenance du caractère présenté.

La reconnaissance des caractères en utilisant la mise en correspondance des pixels est plus simple [85, 47]. Pour identifier les caractères, on compare l'image brute du caractère avec un

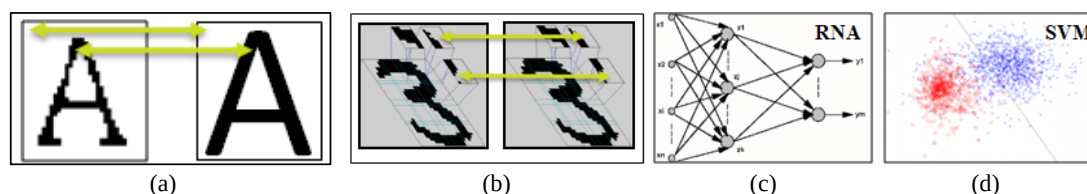


FIGURE 1.7: Différentes techniques de reconnaissance. (a) : Mise en correspondance des pixels ; (b) : Mise en correspondance des caractéristiques ; (c) : Réseaux de neurones artificiels (RNA) ; (d) : Machines à vecteurs de supports (SVM).

modèle constitué de différents types de caractères. Enfin, on affecte à l'image la classe du modèle qui lui correspond le mieux. L'avantage de cette approche est que la description de caractères ne nécessite pas une analyse de forme. Seule une image dont on dispose *à priori* va permettre la reconnaissance. En revanche, les désavantages de cette technique sont liés aux durées de traitement et à la sensibilité aux bruits et à la variation de fontes.

Moins complexe que les approches neuronales et contrairement à la méthode de la mise en correspondance des pixels qui effectue une reconnaissance à partir des pixels l'image, les techniques de classification de caractéristiques utilisent une représentation simplifiée du caractère [48, 49]. Cette représentation est réalisée à travers des primitives extraites de l'image brute du caractère. Ces caractéristiques sont comparées à celles d'autres caractères prédéfinis. Cette approche est plus robuste face au bruit et à la variation de fonte.

Les calculs scientifiques, notamment dans le domaine du traitement d'images ont toujours été consommateurs de ressources informatiques. L'accélération des algorithmes de traitement d'images est alors une nécessité. Dans ce contexte, le calcul parallèle demeure la solution adéquate.

5 Architectures parallèles

Les ordinateurs parallèles sont des machines qui comportent une architecture parallèle, constituée de plusieurs processeurs identiques, ou non, qui concourent au traitement d'une application. FLYNN [154] classe les architectures parallèles en architectures (*Single Instruction Multiple Data* : SIMD) et (*Multiple Instruction Multiple Data* : MIMD). Dans les architectures SIMD les processeurs exécutent la même opération simultanément sur des données différentes. Dans les MIMD les nœuds de calcul (un nœud de calcul est défini par un ensemble de processeurs et de mémoires) exécutent le même traitement mais pas forcément la même opération au même instant. Les architectures MIMD sont plus complexes mais d'usage plus général que les architectures SIMD. De nos jours, la plupart des ordinateurs possèdent des architectures MIMD comme les CPU multi-cœurs.

Le deuxième point qui différencie les architectures parallèles est l'utilisation de la mémoire. Les ordinateurs à mémoire partagée disposent d'une mémoire accessible directement et uniformément à chacun des processeurs (Figure 1.8.a). Cependant, les ordinateurs à mémoire

distribuée présente une mémoire propre pour chaque processeur (Figure 1.8.b).

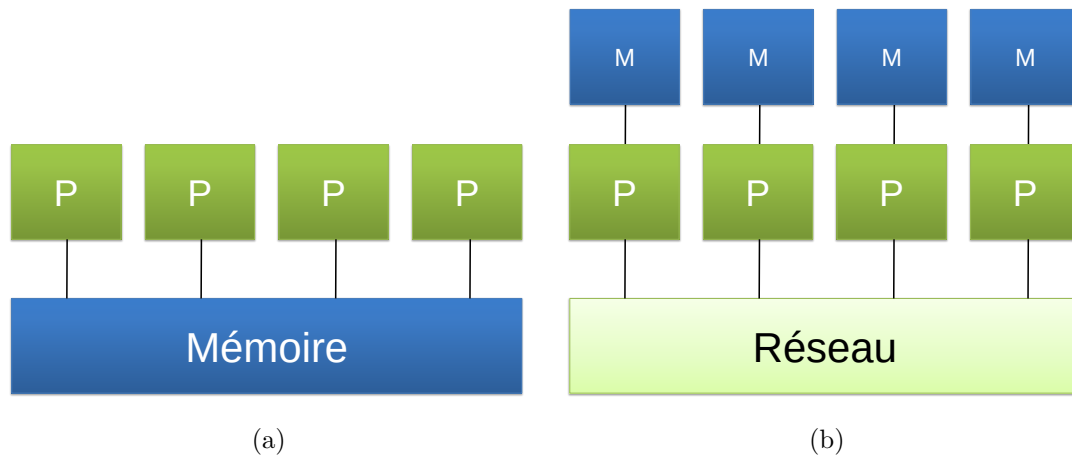


FIGURE 1.8: (a) Ordinateur à mémoire partagée; (b) Ordinateur à mémoire distribuée.

On inclut dans les architectures à mémoire distribuée aussi des processeurs qui ont un espace d'adressage unique et qui possèdent un mécanisme d'accès-mémoire distant. Ce mécanisme permet à un processeur d'accéder directement mais plus lentement à la mémoire d'un autre processeur à travers le réseau. L'existence d'une mémoire commune entre les processeurs simplifie la parallélisation des algorithmes.

Les architectures à mémoire partagée exploitent efficacement le parallélisme à grain fin. Celui-ci est plus facile à extraire et à contrôler que le parallélisme à grain plus gros qui est adéquat aux architectures à mémoire distribuée. Cependant, dans les ordinateurs à mémoire partagée, le nombre de processeurs est limité. Le goulot d'étranglement que constitue l'accès à une mémoire commune limite le nombre de processeurs à une dizaine.

Actuellement, les architectures à mémoire distribuée sont les seuls à utiliser simultanément plusieurs dizaines à quelques centaines de processeurs. On parle alors de parallélisme massif. Les architectures à mémoire distribuée sont plus extensibles que celles à mémoire partagée. C'est-à-dire qu'il est possible d'augmenter le nombre de nœuds de calcul. Il existe aussi aujourd'hui des ordinateurs à mémoire distribuée dont les nœuds de calcul sont des ordinateurs parallèles à mémoire partagée. Ce type d'architecture permet d'exploiter plusieurs niveaux de parallélisme simultanément. La figure 1.9.b montre un exemple de cette architecture.

6 Conclusion

Dans un document hétérogène, la complexité des régions de texte et d'images impliquent des approches différentes et adaptées pour l'extraction, la description et l'identification des caractères.

Pour l'extraction de texte, l'approche d'analyse de texture est adaptée aux images contenant un texte avec un fond complexe. Les approches d'analyse de composantes connexes (CC)

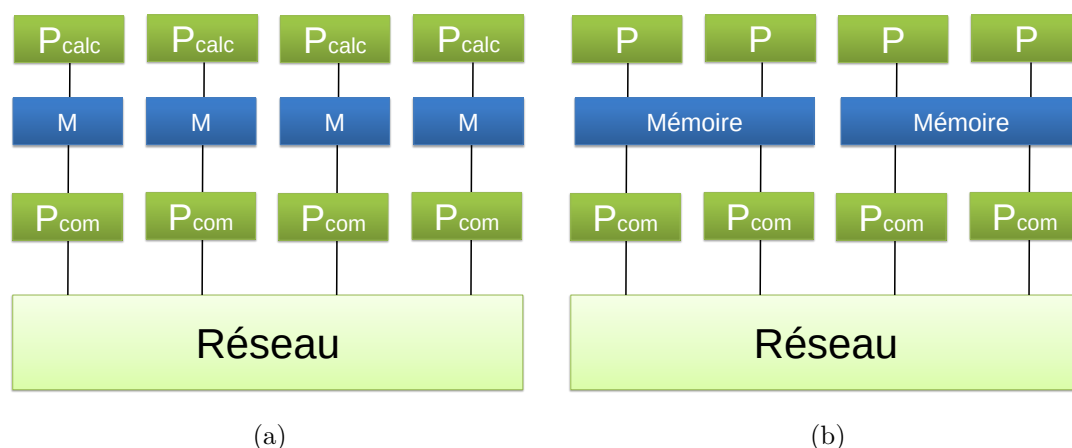


FIGURE 1.9: (a) Ordinateur à mémoire distribuée avec un processeur de communication ; (b) Ordinateur à mémoire distribuée dont les nœuds sont des architectures à mémoire partagée.

de classification et de seuillage sont mieux adaptées pour les régions de texte avec un fond homogène.

Pour la description du texte, les méthodes structurelles sont performantes et offrent une représentation invariante aux distorsions dues aux variations d'échelles, etc. L'identification des caractères en utilisant la mise en correspondance des caractéristiques est moins complexe.

Dans le chapitre qui suit, nous détaillons notre contribution dans le cadre de l'extraction du texte dans les documents hétérogènes où les caractères peuvent se retrouver soit sur des régions homogènes ou non homogènes (complexes).

Chapitre 2

Extraction hybride de texte dans les documents hétérogènes

Dans ce chapitre nous proposons une nouvelle méthode d'extraction hybride de texte dans les documents hétérogènes. Cette méthode assure une extraction fiable de texte et s'adapte à des arrière-plans complexes ou dégradé. Elle permet d'atteindre une F-mesure de 98.84% sur des documents hétérogènes scannés issus de la base LRDE-DBD.

Sommaire

1	Introduction	44
2	Extraction de texte	44
2.1	Binarisation	45
2.2	Analyse de la structure du document	49
2.3	Localisation des caractères	51
3	Nouvelle méthode hybride pour l'extraction de texte dans les documents hétérogènes : EHT	51
3.1	Séparation des régions texte/images dans les documents hétérogènes	53
3.2	Binarisation hybride de texte	57
3.3	Localisation des caractères par dissection	77
4	Expérimentations	77
4.1	Binarisation hybride de texte	78
4.2	Optimisation algorithmique de la correction gamma	79
4.3	Extraction hybride de texte dans les documents hétérogènes . . .	80
5	Conclusion	83

1 Introduction

Dans les documents hétérogènes scannés, il est difficile d'extraire du texte avec un arrière-plan complexe car les caractères se trouvent mélangés avec la texture de l'arrière-plan. Et même si le fond est homogène, la dégradation engendrée par le processus de la numérisation, tel que le bruit et l'éclairage non uniforme, diminue la précision de l'extraction.

Un autre problème se pose c'est qu'il est difficile d'isoler avec une haute précision les caractères dans un document hétérogène comprenant des régions de texte et des images en utilisant une méthode unique, car chaque méthode est adaptée à un seul type de document : texte ou image.

Dans ce chapitre nous proposons une nouvelle méthode efficace et automatique pour l'extraction de texte à partir des documents hétérogènes scannés quelque soit la nature des régions et la dégradation qu'elle contient.

Nous détaillons dans la section 2, les approches locale et globale de l'extraction de texte. Ensuite nous présentons notre méthode d'extraction de texte dans la section 3. Puis, nous évaluons dans la section 4 notre proposition en la comparant avec des méthodes robustes de l'état de l'art.

2 Extraction de texte

Généralement, la numérisation des documents papier altère la qualité des documents en y introduisant plusieurs dégradations telles que l'éclairage non-uniforme (Figure 2.1.a) ou le bruit (Figure 2.1.b).

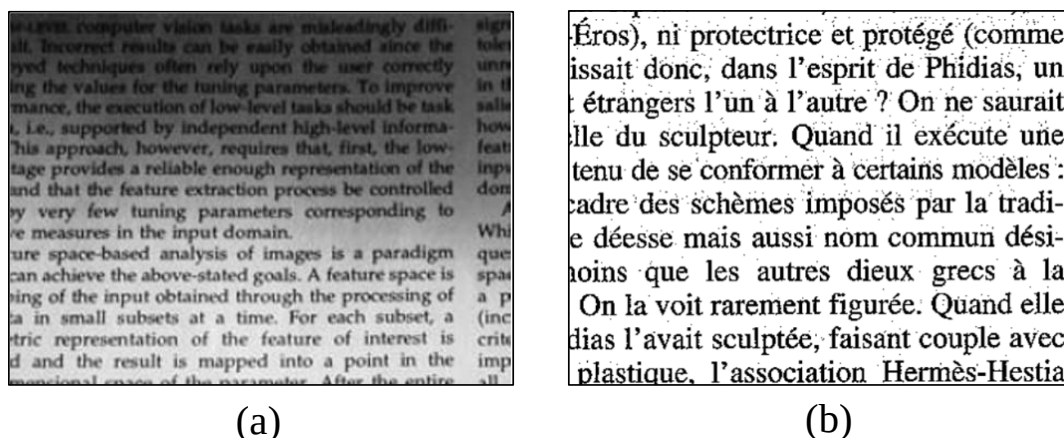


FIGURE 2.1: Dégradations possibles sur les documents scannés engendrés par la numérisation. (a) : éclairage non uniforme; (b) : bruit.

Une extraction efficace de texte dans ces documents est une des principales étapes dans un

système de reconnaissance optique de caractères. La figure 2.2 illustre les principales étapes d’une chaîne d’extraction de texte, notamment celles de la binarisation, de l’analyse de la structure du document et de la localisation des caractères.

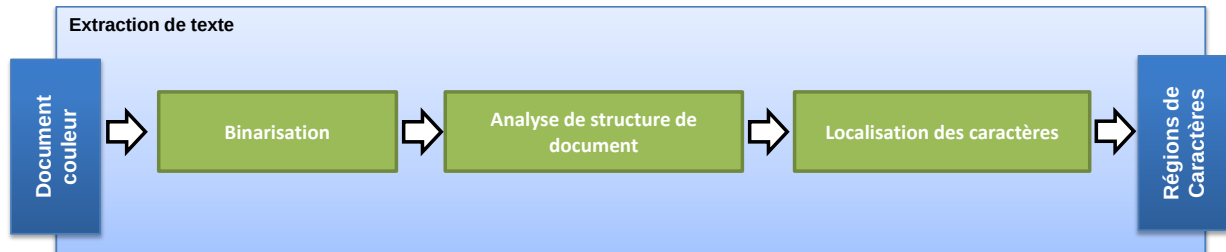


FIGURE 2.2: Architecture classique d’une chaîne d’extraction de texte.

2.1 Binarisation

L’extraction de texte se base sur une séparation texte/arrière-plan. Celle-ci est traduite par un traitement de binarisation qui convertit automatiquement les images de documents sous formes binaires de telle sorte que les pixels de l’avant-plan sont représentés par des pixels noirs et ceux de l’arrière-plan sont représentés par des pixels blancs [64]. En effet, l’utilisation de deux niveaux (binaire) diminue la quantité de calcul et permet d’utiliser des méthodes d’analyse d’image plus simples par rapport à celles conçues pour les images couleurs ou niveaux de gris [68]. En général, les méthodes de binarisation sont classées en deux catégories : globale et locale [64].

2.1.1 Les méthodes de binarisation Globales

Dans les méthodes de binarisation globales, les pixels de l’image sont classés en texte et arrière-plan selon un seuil global. Ces méthodes sont principalement basées sur les approches d’analyse d’histogramme et de classification. Dans la catégorie des méthodes basées sur l’analyse de l’histogramme, une des méthodes classique est celle de Rosenfeld et Kak [20]. Un seuil est sélectionné à partir de l’histogramme 2D du document. On suppose dans cette méthode que les pixels de chaque objet sont situés autour de chaque pic de l’histogramme.

Plus récente, la méthode de seuillage globale Itérative (*Iteratif Global Thresholding* : IGT) [21] comme son nom l’indique, calcule itérativement un seuil global S_{IGT} permettant de séparer au mieux le texte et l’arrière-plan. Pour une image donnée I , la couleur moyenne m de cette image est calculée et une nouvelle image I_1 est générée en se basant sur un étirement de l’histogramme de I . L’étirement de l’histogramme est effectué selon la formule suivante :

$$I_1(x, y) = 1 - \frac{1 - (I(x, y) - m + 1)}{1 - \min(I)} \quad (2.1)$$

Avec $\min(I)$ représente la valeur minimale des pixels de l'image I . Cette opération est réalisée itérativement. Pour chaque itération i une image I_i est calculée en se basant sur l'étirement de l'histogramme de l'image I_{i-1} . La convergence de l'algorithme est atteinte selon le critère suivant :

$$|m_i - m_{i-1}| < 0.05 \quad (2.2)$$

Une fois la condition d'arrêt atteinte, la binarisation de l'image I est effectuée selon le seuil global $S_{IGT} = m_k$, avec k représente l'indice de l'itération finale ($i \in \{1, \dots, k\}$). L'inconvénient majeur de cette méthode, basée sur l'étude d'histogramme, est la détection erronée de pics causés par un bruit [22]. En effet, les méthodes de binarisation globales sont efficaces, rapides et robustes lorsqu'elles sont appliquées aux documents avec un éclairage uniforme. Toutefois, les documents dégradés : bruités ou avec un contraste non uniforme, ne sont pas bien traités. Pour surmonter ce problème, les méthodes locales peuvent être utilisées.

2.1.2 Les méthodes de binarisation locales

Dans les méthodes de binarisation locales, les pixels de l'image sont classés en texte et arrière-plan selon un seuil local déterminé par le voisinage de chaque pixel. Certains travaux de binarisation sur des images de scènes naturelles ont été proposés [30, 31]. Dans ces méthodes, la binarisation est appliquée sur les régions de mots [30] ou de caractères [31] en utilisant l'algorithme de classification Kmeans. Deux d'entre eux effectuent une binarisation robuste grâce à une classification locale de Kmeans. Cependant, ces méthodes supposent que le texte est correctement localisé dans les étapes précédentes ce qui introduit des calculs supplémentaires.

Niblack [93] calcule un seuil basé sur la moyenne et l'écart type du voisinage de chaque pixel. Le nombre de pixels voisins est fixé selon une fenêtre de taille prédéfinie. Le seuil local $S_{Niblack}$ pour chaque pixel $I(x, y)$ est calculé à l'aide de la formule suivante :

$$S_{Niblack} = m + k \times \sigma \quad (2.3)$$

avec m et σ représentent respectivement la moyenne et l'écart type des pixels dans la région de la fenêtre centré sur le pixel $I(x, y)$. La valeur k est un paramètre qui sert à contrôler la quantité de texte dans la fenêtre locale. Pour conserver et gérer les détails locaux on aura besoin de fenêtres de petites tailles mais celles ci ne couvriront plus les objets textuels. En général, les fenêtres de tailles 15×15 et $k = -0.2$ sont recommandées [82]. En fait, Niblack identifie les régions de texte correctement en tant que texte, cependant, il a tendance à produire une grande quantité de bruit dans les régions non-textuelles.

Sauvola [94] améliore Niblack en utilisant la plage dynamique de l'écart type. Dans la méthode Sauvola, le seuil de binarisation est calculé selon la formule suivante :

$$S_{Sauvola} = m \times [1 + k \times (\frac{\sigma}{R} - 1)] \quad (2.4)$$

avec m et σ représentent la moyenne et l'écart type des pixels dans la région de la fenêtre centrée sur le pixel $I(x, y)$. R est la plage dynamique de l'écart type. Les paramètres recommandés pour cette méthode sont $R = 128$ et $k = 0.5$. En cas de faible séparation entre le

texte et l'arrière-plan, le résultat est dégradé de manière significative.

SauvolaMs_{kx} [69] a été proposé comme une amélioration de la méthode Sauvola en l'étendant vers la binarisation multi-échelles. La figure 2.3 illustre les étapes de cette méthode.

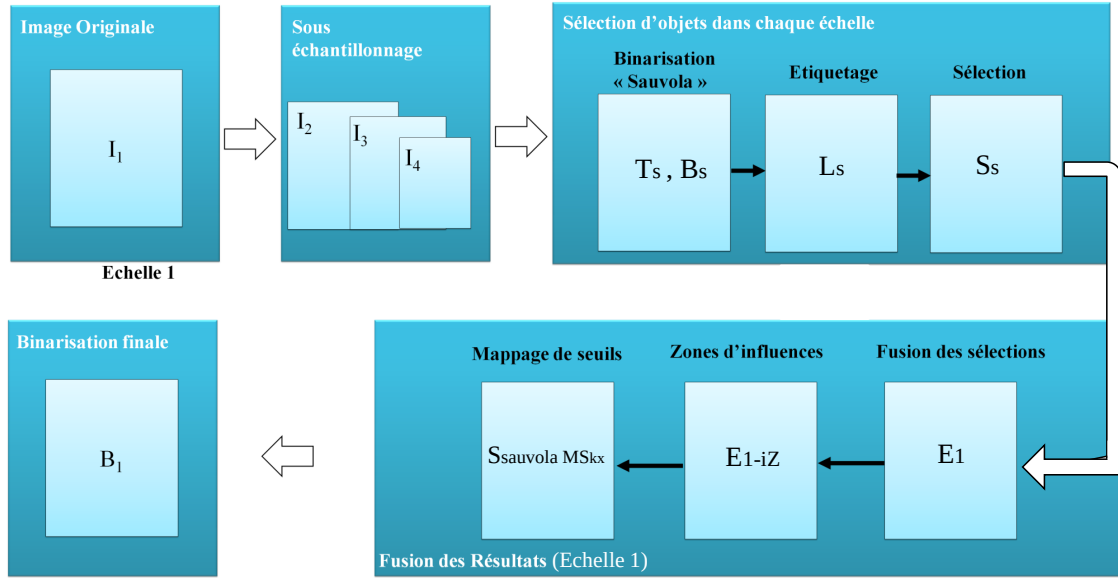


FIGURE 2.3: La méthode *SauvolaMs_{kx}*.

D'abord un sous-échantillonnage de l'image originale I_1 (Échelle 1) est réalisé sur quatre échelles s et produit trois images I_2 , I_3 et I_4 . Ensuite, *SauvolaMs_{kx}* binarise indépendamment les images I_i où $s \in [1, 4]$ en utilisant la méthode Sauvola, décrite précédemment. Pour chaque échelle s , deux images sont générées. T_s contient les seuils de binarisation de chaque pixel et B_s contient l'image binarisée.

Les objets de l'avant-plan sont étiquetés en se basant sur la classification des composantes connexes en texte simple, texte de titre de taille moyenne et texte de titre de taille large. À partir de l'image B_s les composantes connexes de très petites ou de très grandes tailles sont éliminées. En effet, les équations 2.5, 2.6 et 2.7 définissent respectivement les tailles spécifiques des objets à sélectionner dans les différentes échelles.

$$\begin{cases} \min \text{ région}(1) = 0 \\ \max \text{ région}(1) = w \times 2 \times 0.7 \end{cases} \quad (2.5)$$

$$\begin{cases} \min \text{ région}(s) = 0.9 \times \frac{\max \text{ région}(s-1)}{q^2} \\ \max \text{ région}(s) = \max \text{ région}(s-1) \times q^2 \end{cases} \quad (2.6)$$

$$\begin{cases} \min \text{ région}(s_{max}) = 0.9 \times \frac{\max \text{ région}(s_{max}-1)}{q^2} \\ \max \text{ région}(s_{max}) = +\infty \end{cases} \quad (2.7)$$

q et w représentent respectivement le facteur de réduction et la taille de la fenêtre. Pour chaque échelle s une image S_s est générée contenant les composantes connexes sélectionnées. Les objets sélectionnés dans les images S_s sont fusionnés et sur-échantillonnés en une seule image E_1 de même échelle que l'image I_1 . Un algorithme d'influence de zone [171] est appliqué à l'image E_1 pour propager l'information d'échelle et lisser les résultats, une image $E_1 - iZ$ est générée. Une interpolation des seuils de binarisation contenus dans les images T_s , avec $s \in [1, 4]$ en fonction de l'information d'échelle présente dans $E_1 - iZ$ permet de calculer l'image $S_{SauvolaMS_{kx}}$. Enfin une binarisation pixel par pixel de l'image I_1 est réalisée en utilisant les seuils calculés dans $S_{SauvolaMS_{kx}}$.

Cette méthode améliore la binarisation de *Sauvola* pour des petits et larges objets à l'intérieur d'un document automatiquement sans ajustement empirique de la taille des fenêtre au contenu.

Il faut reconnaître que les méthodes locales fournissent une solution adaptative selon les propriétés de chaque zone de document. Cependant, elles peuvent introduire du bruit tel que des artefacts dus aux traitements locaux. Pour améliorer la qualité de binarisation, certains travaux récents combinent les approches locales et globales [64]. Ils effectuent une binarisation locale efficace tout en réduisant le bruit global [64]. Par exemple, Gabara [36] utilise une technique de seuillage locale basée sur la détection globale des contours. Hybrid IGT [37] applique la binarisation globale IGT [21] sur l'image entière puis, la réapplique seulement sur les régions bruitées. Dans la plupart des cas, les méthodes hybrides améliorent la qualité de binarisation des documents dégradés.

Beaucoup de méthodes de binarisation basées sur la classification sont rapportées dans la littérature [64]. L'algorithme des c-moyennes floues (Fuzzy C-means : FCM) [25] est l'un des algorithmes les plus efficaces [64]. Il définit le degré d'appartenance des pixels de l'image au texte ou à l'arrière-plan. Le degré d'appartenance de chaque pixel est calculé selon une manière relativement complexe. Moins complexe [26], Kmeans est une méthode de classification simple et flexible qui peut être utilisée dans la binarisation d'images [64]. Cet algorithme [66] classe les pixels selon un calcul de distance. Il sera détaillé dans la section 3.2.3.

La qualité de l'extraction peut être altérée par la nature de l'arrière-plan complexe du texte. Cette complexité se traduit par une variation de couleurs ou par la présence de texture. La figure 2.4 illustre un cas où le texte se situe sur un arrière-plan de couleurs variées et de texture complexe. Dans ce cas, l'analyse de textures est efficace pour isoler le texte sur un arrière-plan texturé. Cependant, le faible contraste entre les caractères et l'arrière-plan peut engendrer un mélange de couleur entre le texte et le fond.

La plupart des méthodes d'extraction de texte sont destinées à un seul type de document texte/image. Par exemple, les méthodes globales sont efficaces pour des documents ne contenant que du texte. Tandis que les méthodes locales améliorent les résultats pour les



FIGURE 2.4: Extraction de texte difficile à partir d'un arrière-plan texturé.

images contenant une variation de couleurs. Dans le cas d'un document hétérogène, il devient difficile de bien isoler à la fois les caractères dans les régions de texte et dans les images.

2.2 Analyse de la structure du document

plusieurs approches de séparation des régions de texte et d'images, basées sur des interfaces logicielles homme-machine, existent dans la littérature [172]. Cependant, cette séparation représente une tâche fastidieuse lorsqu'on manipule une grande quantité de documents. Les approches ascendantes et descendantes [173, 175] assurent la détection automatique des régions texte/images dans les documents hétérogènes.

2.2.1 Les méthodes ascendantes

Les méthodes ascendantes [173, 174] analysent les composantes connexes pour les classer en texte, images, graphiques, etc. Ensuite, les composantes textuelles sont regroupées en utilisant la morphologie mathématique ou les graphes, etc., pour former des lignes textuelles, avec des types spécifiques tel que les titres et les sous-titres.

Y. Zhan a proposé une méthode ascendante pour la séparation texte/image [180] qui inclut une étape de prétraitement afin de lisser les blocs de texte et supprimer le bruit en utilisant un filtre gaussien. Ces blocs de texte sont divisés en composantes connexes textuelles et non-textuelles. Les composantes non textuelles sont éliminées par un filtrage. Le reste des composantes est fusionné selon plusieurs couches de texte en utilisant l'algorithme de classification K-means [66]. Un ensemble de contraintes est appliqué pour trouver les vraies couches de texte. La couche de texte s'affine à travers une étape de post-traitement. Cet algorithme est efficace pour un texte en couleurs, de plus il est insensible à la taille de la fonte. Cependant il rencontre des difficultés pour détecter l'écriture manuscrite et reconnaître les

régions de texte sur les arrière-plans complexes.

Un autre algorithme de séparation texte/image basé sur les caractéristiques des composantes connexes a été proposé pour un document manuscrit complexe [179]. Cet algorithme utilise la technique Adaptive Local Connectivity Map (ALCM) [133] pour regrouper les composantes connexes en lignes de texte. Ces lignes textuelles sont extraites par l'association des emplacements de leurs masques avec l'image binaire. Cette méthode possède l'avantage de traiter les lignes de texte fluctuantes ou obliques, et elle est utilisée pour les documents imprimés. Par contre, elle ne détecte pas les régions de texte non-rectangulaires.

Le principal avantage de l'approche ascendante est sa capacité à détecter des régions non-rectangulaires avec simplicité. Par contre, le principal inconvénient est son inefficacité à prendre en compte les structures de niveau supérieur dans l'image, telles que les colonnes. Cela conduit souvent à des régions trop fragmentées.

2.2.2 Les méthodes descendantes

Les méthodes descendantes [175] supposent que la page du document possède un modèle fixe, c'est-à-dire un nombre prédéfini de colonnes.

Dans ces méthodes, d'abord l'image est divisée selon des parties en utilisant des coupures verticales/horizontales récursives le long des espaces blancs dans la page. Ceux-ci représentent les limites des colonnes ou des paragraphes.

Par exemple, la méthode [177] détecte les caractères à partir d'un document hétérogène. L'image d'entrée est d'abord convertie en niveau de gris. Ensuite, une détection de contours est réalisée sur l'image par convolution avec des masques de Sobel [176]. Après l'élimination des non-maximas, l'image résultante est divisée en petits blocs de $m \times m$ pixels, où m est calculé en fonction de la résolution d'image [177]. Finalement, on classe les blocs à l'aide d'un seuil prédéfini qui permet de différencier le texte de l'image. Bien que cette méthode résout le problème du chevauchement entre le texte et le graphique, elle souffre de l'incapacité à générer des régions texte et images non-rectangulaires.

La méthode [178] traite des images de documents imprimés, pouvant contenir du texte dans les images de scènes naturelles. Elle est basée sur l'identification des contours multi-échelles pour la détection de texte, et sur les opérateurs morphologiques de dilatation pour la localisation de texte. Malgré que cette méthode traite et manipule les dégradations dans le document tel que le flou et le texte distordu, elle reste incapable de bien repérer les régions non-rectangulaires.

Généralement, les méthodes descendantes produisent des régions sémantiquement correctes et peu fragmentées. Cependant, ces méthodes rencontrent des problèmes pour produire des régions non-rectangulaires. De plus, seuls les documents qui s'apparentent avec le modèle de la page de l'hypothèse peuvent être traités.

L'idée est de combiner les deux approches pour tirer profit des avantages et de combler les inconvénients des deux approches. La méthode d'analyse de structure de document (*Page Layout Analysis* : PLA) [56] de Tesseract [43] associe les deux approches descendantes et ascendantes pour séparer efficacement les régions textuelles des régions d'images.

2.3 Localisation des caractères

La localisation des caractères est l'une des étapes les plus importantes dans un système OCR. L'erreur dans cette étape engendre une dégradation considérable dans le taux de reconnaissance finale du texte. Plusieurs approches de localisation de caractères existent dans la littérature. On peut regrouper ces méthodes principalement sous trois approches. Les méthodes holistiques, de dissection, et les méthodes basées sur la reconnaissance des caractères.

2.3.1 Localisation holistique

Cette approche repose sur la reconnaissance de la totalité des mots sans besoin de les segmenter en caractères [150, 151, 152]. Le problème majeur de cette catégorie est l'obligation d'utiliser un lexique bien défini car elle utilise directement les mots et non pas les lettres.

2.3.2 Localisation basée sur la reconnaissance des caractères

Dans cette catégorie de méthodes, on s'appuie sur la segmentation des caractères en images comparables à des alphabets. Généralement on utilise le critère de confiance pour décider si une segmentation de caractères est satisfaisante. Ce critère traduit le degré de certitude concernant la similitude du caractère localisé à un caractère de l'alphabet. On trouve les méthodes qui cherchent les caractères en se basant sur leurs matrices de pixels [143, 144] et les méthodes qui se basent sur leurs caractéristiques [145, 146, 147, 148, 149].

2.3.3 Localisation basée sur la dissection

Les méthodes de dissection [135] consistent à découper l'image en symboles quelque soit leurs contenus (caractères ou pas). On utilise souvent des critères tel que la hauteur, la largeur, la marge entre les composantes connexes voisins, etc. Plusieurs catégories dérivent de cette approches. Par exemple il y a les méthodes basées sur l'espacement (marge) entre les caractères [136, 137], les méthodes qui se basent sur l'analyse de la projection des pixels [138, 139] et les méthodes qui s'appuient sur l'analyse des composantes connexes [140, 141, 142].

D'une part, les méthodes de localisation basées sur la reconnaissance des caractères et les méthodes holistiques sont performantes pour un texte où l'espacement entre les caractères est aléatoire. D'autre part pour un texte qui comporte une marge fixe entre les caractères, une simple technique basée sur la dissection est suffisante. Dans ce contexte, la méthode Tesseract [43] combine les deux approches dissectrice et holistique pour séparer convenablement les caractères.

3 Nouvelle méthode hybride pour l'extraction de texte dans les documents hétérogènes : EHT

Dans l'optique d'une automatisation totale de l'extraction de texte dans les documents hétérogènes, nous proposons une approche adaptative qui sélectionne le traitement selon la nature des régions de texte sur fond simple ou complexe. D'après la littérature, pour résoudre le problème de l'extraction de texte à partir d'une texture complexe, l'amélioration du contraste est utilisée [71]. Nous proposons donc une approche d'analyse de textures qui permet d'améliorer le contraste de l'image avant la binarisation. Par ailleurs, les approches de binarisation globales [21, 64] et locales [68, 34] ne sont pas suffisantes pour résoudre le problème de l'extraction de texte sur un arrière-plan dégradé, ou sur un document bruité. Comme solution nous adoptons une approche hybride, qui combine une approche locale basée sur la classification des pixels en avant et arrière-plan avec une approche globale qui diminue la quantité de distorsion dans l'ensemble du document.

Comme le montre la figure 2.5, deux approches composent la méthode d'Extraction Hybride de Texte proposée, que nous appelons (EHT) : La binarisation du document et l'analyse de sa structure.

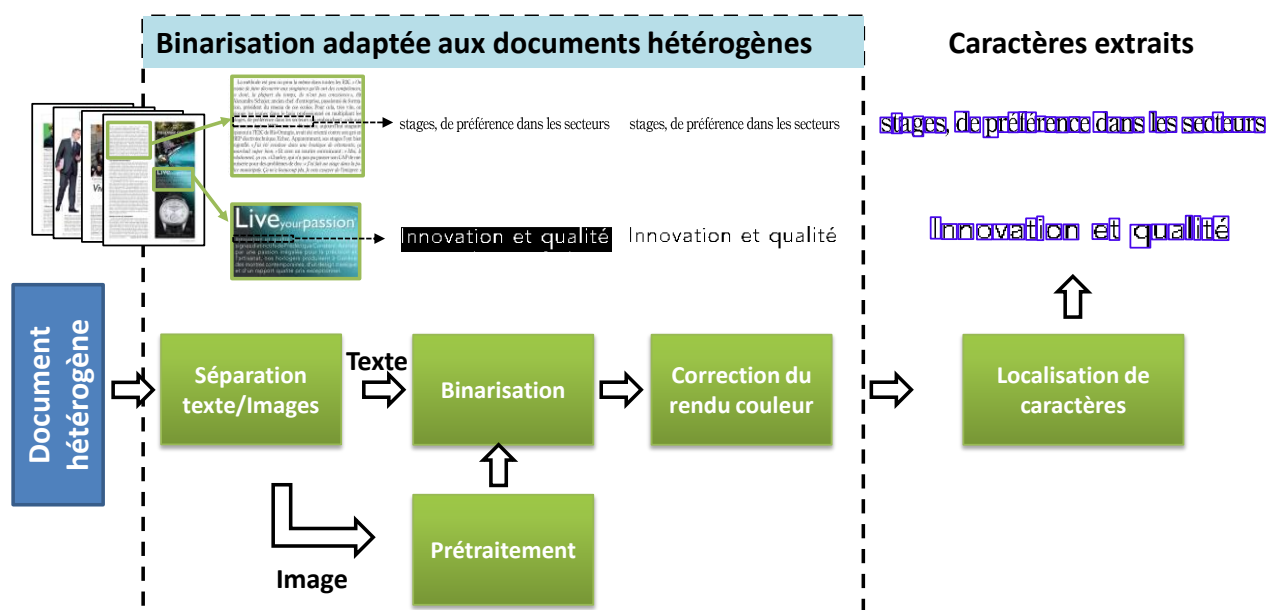


FIGURE 2.5: Organigramme de la méthode d'extraction hybride de texte proposée : EHT.

Dans l'approche de binarisation, on applique sur les régions d'images une correction gamma [71] pour améliorer leurs contrastes. On sépare le texte en se basant sur l'algorithme de classification Kmeans [66] à partir des régions de texte et d'images améliorées. Ensuite, on produit le bon rendu couleur à partir des régions d'images et de textes. L'approche d'analyse

de la structure du document possède deux niveaux. D'abord séparer le document en régions de texte et d'images avant leur binarisation. Ensuite, segmenter les paragraphes de texte binaires en caractères disjoints. Ces étapes sont détaillées dans la suite.

3.1 Séparation des régions texte/images dans les documents hétérogènes

Dans notre méthode d'extraction hybride de texte (EHT) nous devons reconnaître avec précision si une région du document représente un texte sur un fond homogène ou bien une image pour réaliser le traitement adéquat en vue de l'extraction convenable des caractères. Au départ, nous séparons le document en régions de texte et d'images en analysant la structure de la page du document. Comme nous l'avons montré dans la section précédente les approches hybrides sont les plus efficaces pour ce genre de traitement. Dans ce contexte, nous proposons d'employer l'algorithme (*Page Layout Analysis* : PLA) [56] de Tesseract [43] qui est l'un des algorithmes hybrides les plus connus (Figure 2.6). En effet, il permet de reconnaître avec précision les régions non-rectangulaires dans le document.

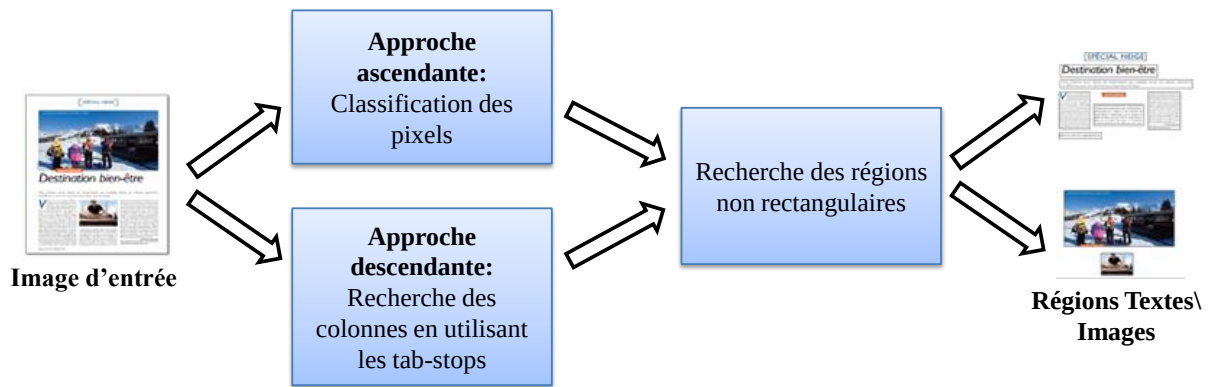


FIGURE 2.6: Séparation du document en régions de texte et d'images : Algorithme PLA [56] de Tesseract [43] (Approche hybride).

Cet algorithme combine deux approches ascendante et descendante. D'une part, l'approche ascendante est basée sur la morphologie mathématique (dilatation, érosion) et l'analyse des composantes connexes. D'autre part, l'approche descendante est basée sur la recherche des délimiteurs de colonnes, appelés tabulations, pour la détection des colonnes.

Les tabulations déterminent aussi les éléments non-rectangulaires placés à l'extérieur ou à l'intérieur des bords des colonnes. Nous distinguons des tabulations d'images, de colonnes et de citations dans la figure 2.7.

Dans la suite nous expliquons les étapes de la méthode PLA, à savoir : l'approche ascendante de *classification des pixels*, l'approche descendante de *recherche des colonnes* et la *recherche des régions non-rectangulaires*.



FIGURE 2.7: Exemple de tabulations dans un document hétérogène.

3.1.1 Classification des pixels

La première étape de l'algorithme PLA [56] consiste à classer les pixels du document en texte et images. Dans cette étape on commence par identifier les traits, les régions d'images, et les composantes connexes. Un traitement de morphologie mathématique est appliqué sur une binarisation Otsu du document considéré afin de détecter les traits horizontaux et verticaux (Figure 2.8.b) et le masque des images (Figure 2.8.c)

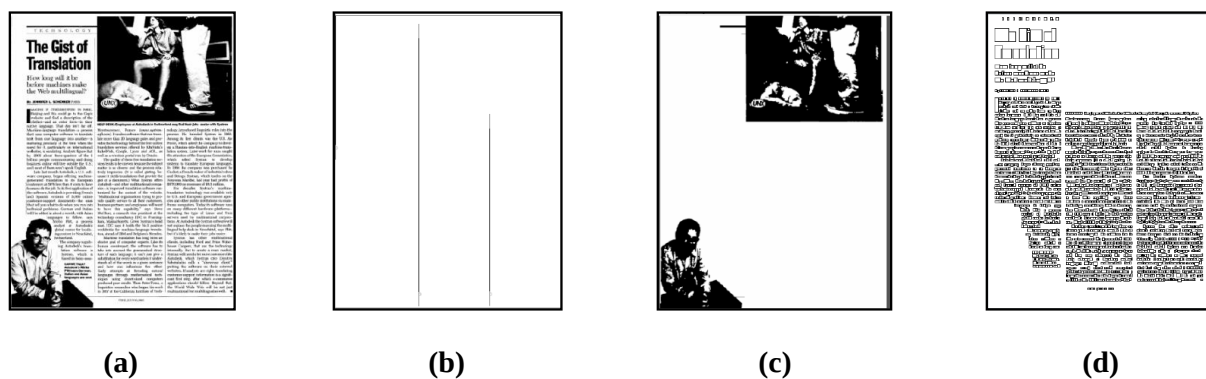


FIGURE 2.8: Étapes de pré-traitement de l'algorithme PLA [56]. (a) : Image originale ; (b) : Traits verticaux détectées ; (c) : Régions d'images détectées ; (d) : composantes connexes de texte filtrées.

La détection des lignes verticales et horizontales emploie la succession des opérations morphologiques comme suit :

3. NOUVELLE MÉTHODE HYBRIDE POUR L'EXTRACTION DE TEXTE DANS LES DOCUMENTS HÉTÉROGÈNES : EHT

- (1) Fermeture morphologique : Dilatation suivie d'une érosion : Cette opération permet de combler les petits trous ce qui permet de lier les pixels assez proches dans l'espace euclidien, ainsi, le texte devient épaisses (Figure 2.9b).
- (2) Ouverture morphologique : Érosion suivie d'une dilatation : Dans ce traitement on effectue une ouverture morphologique avec une grande boîte pour détecter les régions épaisses, pour qu'elles soient soustraites dans l'étape (3) (Figure 2.9c).
- (3) Soustraction (1) - (2) (Figure 2.9d).
- (4) Ouverture morphologique horizontale pour trouver les traits horizontaux (Figure 2.9e).
- (5) Ouverture morphologique verticale pour trouver les traits verticaux (Figure 2.9f).

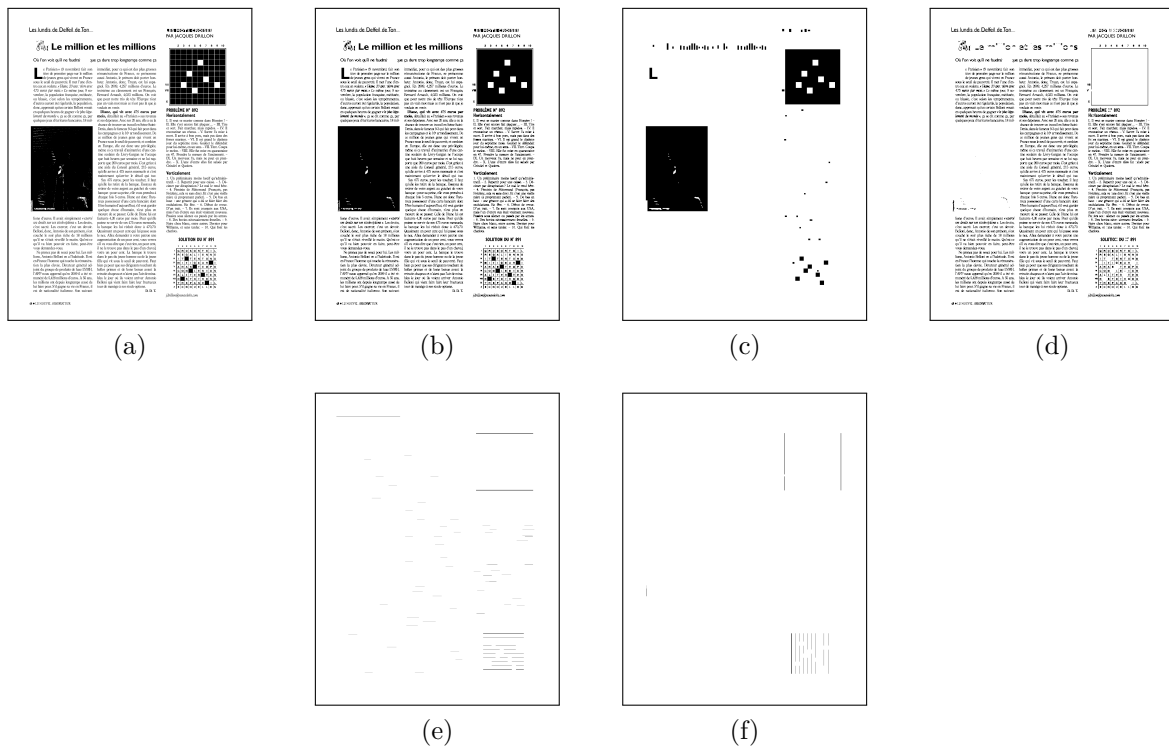


FIGURE 2.9: Détection des traits horizontaux et verticaux : (a) : Image originale binaire, (b) : Fermeture (1), c : Ouverture (2), (d) : Fermeture - Ouverture (3); (e) : Ouverture horizontale (4); (f) : Ouverture verticale (5).

Les régions de texte et d'images doivent être sémantiquement homogènes c'est-à-dire qu'elles appartiennent aux mêmes paragraphes ou régions d'images et non pas la combinaison de plusieurs régions. Ceci favorise le fait d'avoir la même couleur de texte et d'arrière-plan et ainsi facilite l'extraction du texte. Aussi, on veut déterminer l'ordre de lecture ce qui va nous servir dans les traitements ultérieurs pour la reconnaissance des caractères. La recherche des colonnes permet de trouver l'ordre de la lecture du texte.

3.1.2 Recherche des colonnes basée sur les tabulations

Parmi les composantes connexes textuelles, on identifie celles qui représentent des tabulations candidates. Celles-ci sont localisées en utilisant une recherche radiale [56] à partir de toutes les composantes connexes détectées dans l'approche ascendante de classification de pixels (section 3.1.1). Ainsi, chaque composante connexe est marquée comme tabulation droite, gauche ou comme non-tabulation (Figure 2.10.a). En effet, les composantes connexes (CCs) candidates sont groupées selon des lignes. Si la taille d'un groupe de composantes est suffisante, les CCs sont gardées sinon elles sont supprimées. L'algorithme «Least Median of Squares» [174] est utilisé pour tracer un segment de bout en bout entre les bords (droit ou gauche) pour chaque composante connexe.

Une fois tous les segments des tabulations localisés, toutes les lignes sont redessinées selon les directions moyennes de façon à ce que toutes les CCs membres d'une même tabulation se situent sur un coté du segment (Figure 2.10.b).

Pour connecter les tabulations on effectue le suivi (tracking) des lignes de texte. Les tabulations qui possèdent des lignes de textes en commun sont associées et sont connectées de manière qu'elles commencent à partir des mêmes ordonnées. A ce niveau les tabulations sont construites (Figure 2.10.d).

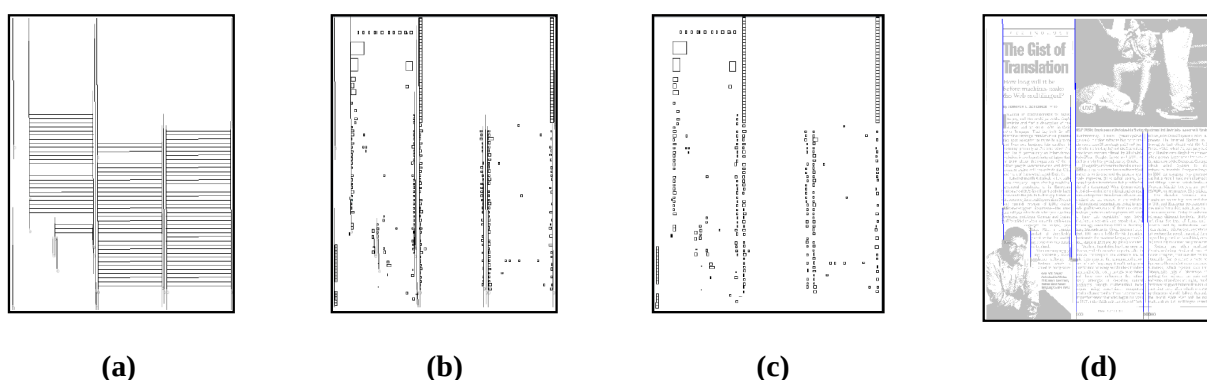


FIGURE 2.10: Recherche des colonnes. (a) : Tabulations candidates; (b) : Segments de Tabulations détectées; (c) : Tabulations connectés; (d) : Tabulations nettoyés.

Pour chercher les colonnes, on parcourt les CCs de gauche vers la droite et du haut vers le bas. Elles sont groupées en partitions de colonnes (*Column Partitions* : CPs), de sorte qu'elles ne dépassent pas les segments des tabulations construites. Un ensemble de CPs d'un seul parcours horizontal s'appelle un ensemble de partition de colonnes (*Column Partition Set* : CPset) (Figure 2.11). Chaque CPset est potentiellement une division de la page en colonnes.

L'objectif est de trouver les colonnes de sorte que les CPset optimales soient gardées. Ainsi, une liste de colonnes candidates est déterminée à partir des ensemble CPsets. Les colonnes

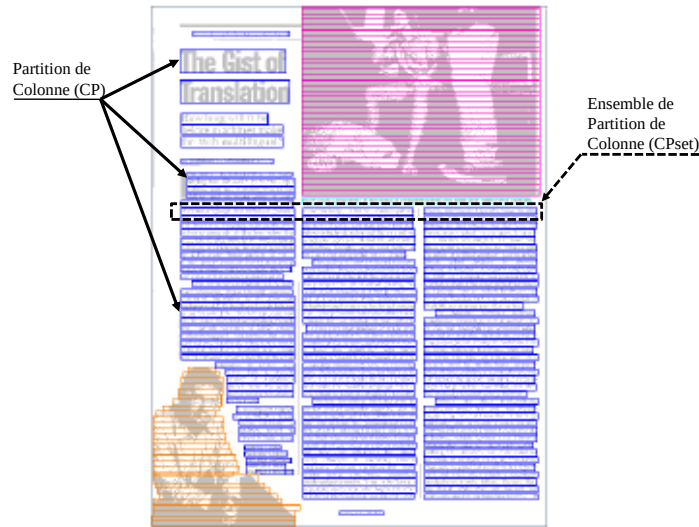


FIGURE 2.11: Représentation des partitions de colonnes (CP) et des ensembles de partition de colonnes (CPset) dans un document hétérogène.

dupliquées sont éliminées par la loi (A explique B) [56] (Figure 2.12.b).

3.1.3 Recherche des régions non-rectangulaires

Après avoir trouvé les colonnes on donne des types aux partitions CPs selon le nombre de colonnes qu'elles couvrent. Une partition qui ne dépasse pas une seule colonne est appelée «flowing». Les partitions qui couvrent plusieurs colonnes et qui ne dépassent pas les bords des colonnes sont des «pull-out». Enfin, les partitions qui dépassent complètement une ou plus qu'une colonne sont appelées «heading».

La figure 2.12.c montre les partitions CPs finales où les types «flowing» et «heading» sont en bleu, et «pull-out» est en orange. Les partitions de colonnes CPs de texte sont divisées en groupes avec une interligne uniforme qui génère des blocs de texte. Ensuite, les polygones des régions sont calculés de façon que les polygones soient isothétiques c'est-à-dire que les contours alternent entre les directions verticales et horizontales. La construction des polygones nécessite que les CPs soient contenues à l'intérieur des polygones et que les CPs d'autres régions ne chevauchent pas (Figure 2.12.d).

La séparation du document hétérogène en régions de texte et d'images est hautement complexe à cause du traitement intensif des opérateurs morphologiques de dilatation et d'érosion.

Après avoir séparé les régions du document en texte et images, les régions correspondantes

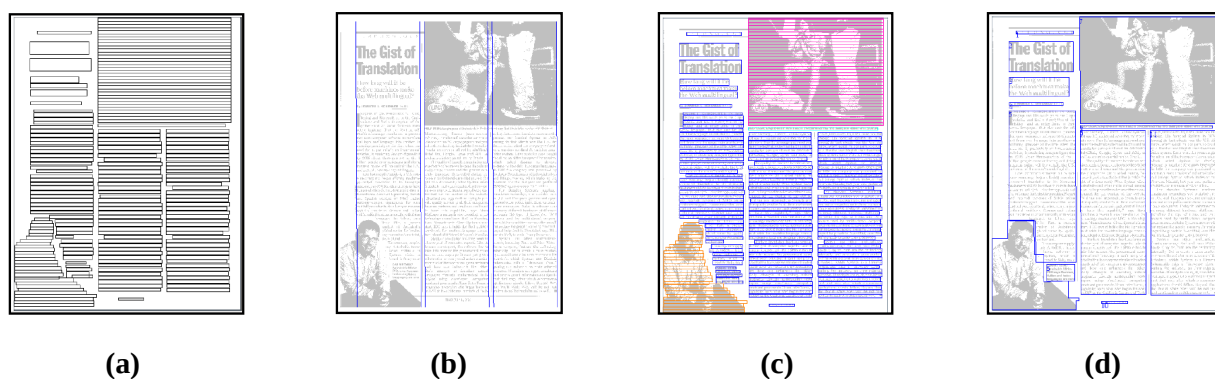


FIGURE 2.12: Recherche des régions et détermination de leurs ordre. (a) : Partitions de colonnes (CPs) candidates; (b) : colonnes détectées; (c) : chaînes de partitions typées; (d) : régions de textes et d'images.

dans le document initial sont binarisées adaptativement pour séparer le texte de l'arrière-plan.

3.2 Binarisation hybride de texte

Dans notre méthode EHT, les régions de texte sur un fond simple sont binarisées en utilisant une méthode de classification hybride basée sur l'algorithme Kmeans [66]. Contrairement aux régions du texte où les caractères sont facilement lisibles sur l'arrière-plan homogène, les régions d'images contiennent un texte qui se mélange avec la texture ce qui rend son extraction difficile. Pour ceci, nous appliquons une correction gamma basée sur l'analyse de textures pour améliorer leurs contrastes avant la binarisation. Dans la suite, nous expliquons les algorithmes de correction gamma, de classification hybride et de correction du rendu couleurs sur les régions du document. Pour chaque méthode nous donnons la complexité algorithmique relative à une image de région (I).

3.2.1 Correction gamma des régions d'images pour une séparation fond/forme optimale

Pour surmonter le problème de l'extraction du texte dans les images avec un fond complexe, quelques méthodes améliorent l'image avant de détecter le texte. Dans ce contexte, nous considérons les approches qui cherchent une valeur gamma pour aider à corriger la séparation entre l'avant et l'arrière-plan. La transformation gamma, exprimée par l'équation 2.8, est un opérateur qui dépend du paramètre γ . Avec une image claire ($\gamma < 1$) et une image sombre ($\gamma > 1$). L'image originale est décrite par une valeur γ égale à 1.

$$q(i, j) = p(i, j)^{\frac{1}{\gamma}} \quad (2.8)$$

3. NOUVELLE MÉTHODE HYBRIDE POUR L'EXTRACTION DE TEXTE DANS LES DOCUMENTS HÉTÉROGÈNES : EHT

La Méthode de Correction de Gamma (GCM) [71] est l'une des méthodes les plus robustes dans ce contexte. En utilisant une analyse statistique de textures, elle vise à supprimer les détails non-textuels dans les images en appliquant la valeur gamma appropriée. Comme

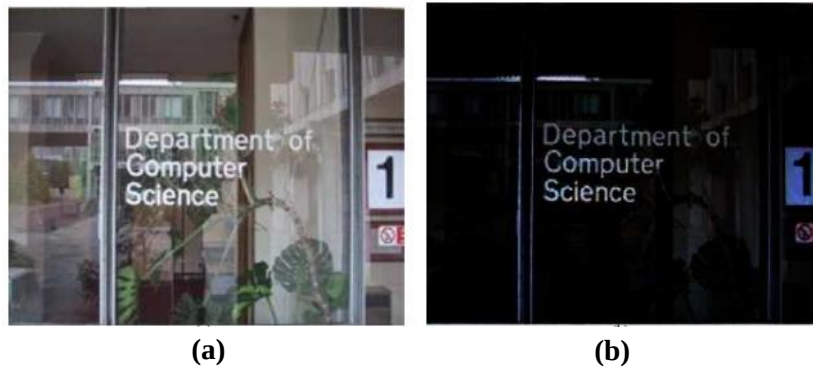


FIGURE 2.13: Exemple de correction gamma. (a) : Image originale ; (b) : Image avec correction gamma en utilisant la méthode GCM [71]

le montre la figure 2.13, cette méthode permet de supprimer partiellement ou complètement l'arrière-plan hétérogène tout en préservant les caractéristiques du texte.

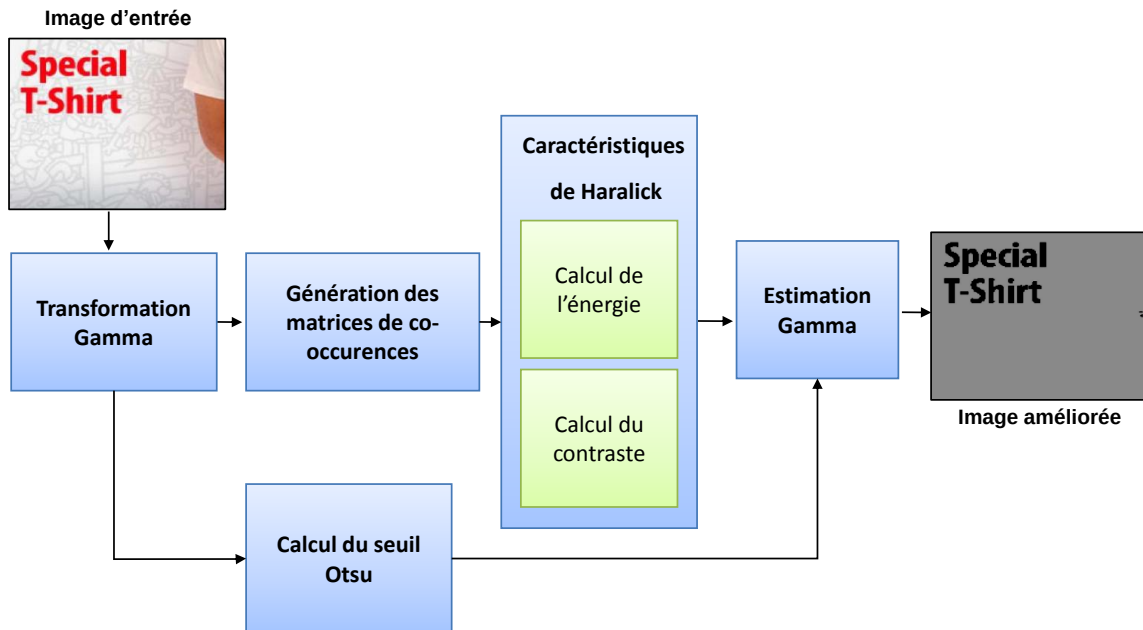


FIGURE 2.14: Correction gamma pour l'amélioration du contraste de l'image.

La figure 2.14 montre les étapes de traitement de la méthode de correction Gamma. Ainsi, GCM [71] transforme une image niveau de gris en Nb_{Images} images ($Nb_{Images} = 100$) en utilisant une rangée de valeurs gamma allant de 0.1 à 10.0 avec un pas de 0.1. Une matrice de co-occurrences est calculée pour chaque image. Les mesures de caractéristiques de texture

sont extraites à partir des quatre matrices de co-occurrences et la valeur-seuil est calculée pour chaque image en utilisant l'algorithme Otsu [65]. L'énergie, le contraste et le seuil sont examinés pour déterminer la valeur appropriée de gamma et ainsi sélectionner parmi les 100 images, celle qui offre le meilleur contraste entre le texte et l'arrière-plan.

La méthode GCM [71] repose donc sur quatre étapes fondamentales que nous détaillons dans la suite de cette section. Ces étapes sont : la génération des matrices de co-occurrences, l'extraction des caractéristiques de Haralick, le calcul du seuil Otsu et l'estimation de la valeur gamma optimale qui permet l'amélioration du contraste de l'image.

Génération des matrices de co-occurrences La matrice de co-occurrences [63], aussi appelée matrice de dépendances spatiales de niveaux de gris (SGLDM pour Spatial Grey Level Dependence Matrix), est une des approches les plus connues et les plus utilisées pour extraire les caractéristiques de textures tels que l'énergie et le contraste. En effet, elle permet d'analyser statistiquement la texture par l'étude des relations spatiales des couples de pixels. Elle caractérise ainsi la périodicité et la directivité des textures.

La matrice de co-occurrences s'intéresse aux relations qui existent entre les niveaux de gris des pixels séparés spatialement par un déplacement (translation) (d) et un angle (a). L'amplitude de la translation d n'excède pas, en général, 2 à 5 pixels, afin de ne prendre en compte qu'une information très locale de voisinage.

Le calcul des matrices de co-occurrences utilisé dans la méthode GCM est réalisé comme suit. Pour chaque image en niveaux de gris, on calcule quatre matrices de co-occurrence qui correspondent à une translation d égale à 1 et aux quatre angles a : 0° , 45° , 90° et 135° (Figure 2.15.a). La figure 2.15.b montre un exemple de positions de couples de pixels identiques (en bleu) dans la matrice de pixels de l'image. A partir des valeurs de ces pixels (Figure 2.15.b), on incrémente la case correspondante de coordonnée (p_1 : valeur du 1^e pixel, p_2 : valeur du 2^e pixel) dans la matrice de co-occurrence (figure 2.15.c). Le résultat est une matrice carrée de taille $N_g \times N_g$, où N_g représente le nombre de niveaux de gris (dans le cas de la figure 2.15, l'image contient quatre niveaux de gris).

L'algorithme 1 montre le principe de la génération des matrices de co-occurrences. Pour plus de clarté, nous illustrons le cas où l'angle a est égal à 0 . On considère une image I de largeur L et de hauteur H . Pour chaque couple de pixels voisins $I(x, y)$ et $I(x, y + 1)$ correspond un couple de niveaux de gris p_1 et p_2 . (Algorithme 1 : lignes 3-4). En parcourant l'image on incrémente la case correspondante de coordonnées p_1, p_2 dans la matrice de co-occurrences $CMat_a$ (Algorithme 1 : ligne 5).

L'algorithme de génération de matrices de co-occurrences pour Nb_{Images} images, présente une complexité de l'ordre de $\mathcal{O}(Nb_{pixels} \times Nb_{Images})$.

Calcul des caractéristiques de Haralick A partir des matrices de co-occurrences générées, il est possible de calculer plusieurs caractéristiques pour la description de la texture

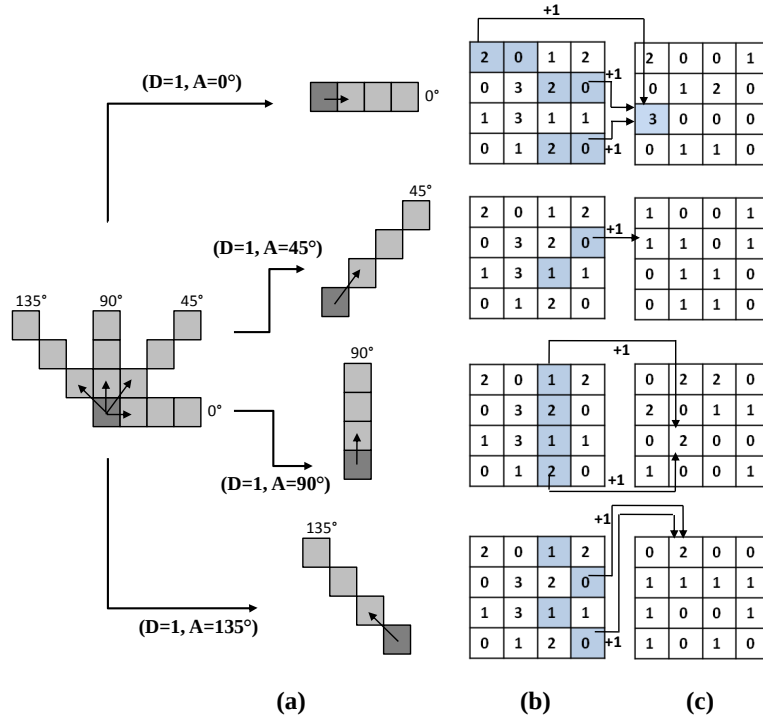


FIGURE 2.15: Exemple de génération des quatre matrices de co-occurrences. (a) : Déplacement et angle; (b) : Position des pixels voisins; (c) : Incrémentation des valeurs correspondantes dans la matrice de co-occurrence.

appelées caractéristiques de Haralick [63]. On trouve par exemple le contraste, l'énergie, la moyenne et la variance, etc. La méthode GCM [71] s'intéresse uniquement aux caractéristiques de contraste et de l'énergie [63]. Pour une matrice de co-occurrences $CMat_a$, ses caractéristiques sont calculées respectivement selon les équations 2.9 et 2.10.

$$E_a = \sum_{i,j=0}^{255} \frac{CMat_a^2(i,j)}{moy^2} \quad (2.9)$$

$$C_a = \sum_{i,j=0}^{255} \frac{(i-j)^2 \times CMat_a(i,j)}{moy} \quad (2.10)$$

moy représente la moyenne des valeurs de la matrice de co-occurrence $CMat_a$ et $i, j \in [0, 255]$ représentent les coordonnées dans la matrice. a est l'angle traité par la matrice de co-occurrences.

L'algorithme 2 détaille l'extraction des caractéristiques d'énergie et de contraste à partir des quatre matrices de co-occurrences. Pour chaque matrice $CMat_a$ (Algorithme 2 : ligne 1), on calcule l'énergie élémentaire E_a et le contraste élémentaire C_a (Algorithme 2 : lignes 2-12) avec $a \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$ l'angle traité. L'énergie E et le contraste C de l'image correspondent à la moyenne de toutes les caractéristiques élémentaires calculées E_a et C_a (Algorithme 2 : lignes 14-15).

Algorithme 1 Génération d'une matrice de co-occurrences d'angle : $a=0^\circ$ et de distance : $d=1$

Entrée : I : Image en niveau de gris de largeur L et de hauteur H

Sorties : $CMat_a$: Matrice de co-occurrence, l'angle $a = 0$

```

1: Pour  $x \in [0..L-1]$  Faire
2:   Pour  $y \in [0..H-1]$  Faire
3:      $p_1 := I(x, y)$ 
4:      $p_2 := I(x, y+1)$  //  $a=0$ 
5:      $CMat_a(p_1, p_2) ++$ 
6:   Fin Pour
7: Fin Pour

```

Algorithme 2 Calcul des caractéristiques de Haralick

Entrées : $CMat_a$: Matrices de co-occurrence, a : angles

Sorties : E, C, moy : Moyenne de la matrice de co-occurrence d'angle a

```

1: Pour  $a \in \{0, 45, 90, 135\}$  Faire
2:   Pour  $i \in [1..N_g]$  Faire
3:     Pour  $j \in [1..N_g]$  Faire
4:        $E_a ++ = \frac{CMat_a^2(i,j)}{moy^2}$ 
5:        $n := (i-j) \times (i-j)$ 
6:        $C_a ++ = \frac{n \times CMat_a(i,j)}{moy}$ 
7:     Fin Pour
8:   Fin Pour
9: Fin Pour
10:  $E := \frac{E_0 + E_{45} + E_{90} + E_{135}}{4}$ 
11:  $C := \frac{C_0 + C_{45} + C_{90} + C_{135}}{4}$ 

```

L'énergie E et le contraste C de l'image sont calculés comme étant la moyenne des énergies et des contrastes élémentaires générées par les quatre matrices de co-occurrences (equations 2.11 et 2.12).

$$E := \frac{\sum E_a}{4}, a \in \{0, 45, 90, 135\} \quad (2.11)$$

$$C := \frac{\sum C_a}{4}, a \in \{0, 45, 90, 135\} \quad (2.12)$$

La complexité du calcul des caractéristiques est de l'ordre de $\mathcal{O}(N_g^2)$ avec N_g est le nombre de niveaux de gris de l'image. Le calcul de l'énergie et du contraste est effectué pour Nb_{Images} , d'où la complexité totale de l'algorithme est de l'ordre de $\mathcal{O}(Nb_{Images} \times N_g^2)$.

Calcul du seuil Otsu L'algorithme Otsu [65] permet de binariser les images à travers un seuillage automatique. Cette méthode nécessite au préalable le calcul de l'histogramme

de l'image, puis, sa séparation en deux classes selon un seuil S_{Otsu} qui minimise la variance intra-classe. Ceci revient à séparer les pixels de l'image en deux classes, la première ayant un niveau maximal, typiquement 255, et la seconde un niveau minimal 0. Dans la suite nous donnons une explication théorique de la méthode de calcul du seuil Otsu suivie de l'algorithme correspondant.

Le principe de la méthode Otsu [65] consiste à trouver automatiquement le seuil qui minimise la variance intra-classe pondérée σ_w^2 (« w » désigne intra-classe) des pixels. Cette variance est donnée par l'équation 2.13.

$$\sigma_w^2 = q_1(S_{Otsu})\sigma_1^2(S_{Otsu}) + q_2(S_{Otsu})\sigma_2^2(S_{Otsu}) \quad (2.13)$$

Avec $S_{Otsu} \in [0, 255]$ est le seuil de séparation des deux classes. D'une part, les quantités $q_1(S_{Otsu})$ et $q_2(S_{Otsu})$ définissent les proportions de pixels relatifs aux deux classes. Elles sont données respectivement par les équations 2.14 et 2.15.

$$q_1(S_{Otsu}) = \sum_{i=0}^{S_{Otsu}} \frac{Hist[i]}{L \times H} \quad (2.14)$$

$$q_2(S_{Otsu}) = \sum_{i=S_{Otsu}+1}^{255} \frac{Hist[i]}{L \times H} \quad (2.15)$$

$Hist(i)$ désigne le nombre de pixels du niveau de gris i et L, H désignent respectivement la largeur et la hauteur de l'image I . D'autre part, on calcule les variances des classes $\sigma_1^2(S_{Otsu})$, $\sigma_2^2(S_{Otsu})$ selon les équations 2.16 et 2.17. Ces variances sont constituées par les pixels de niveaux de gris respectivement dans les intervalles $[0, S_{Otsu}]$ et $[S_{Otsu}+1, 255]$.

$$\sigma_1^2(S_{Otsu}) = \frac{1}{q_1(S_{Otsu})} \sum_{i=0}^{S_{Otsu}} (i - \mu_1(S_{Otsu}))^2 \frac{Hist[i]}{L \times H} \quad (2.16)$$

$$\sigma_2^2(S_{Otsu}) = \frac{1}{q_2(S_{Otsu})} \sum_{i=S_{Otsu}+1}^{255} (i - \mu_2(S_{Otsu}))^2 \frac{Hist[i]}{L \times H} \quad (2.17)$$

$\mu_1^2(S_{Otsu})$, $\mu_2^2(S_{Otsu})$ donnent les moyennes des classes constituées par les pixels de niveaux de gris respectivement dans les intervalles $[0, S_{Otsu}]$ et $[S_{Otsu}+1, 255]$ (équations 2.18 et 2.19).

$$\mu_1(S_{Otsu}) = \frac{1}{q_1(S_{Otsu})} \sum_{i=0}^{S_{Otsu}} i \frac{Hist[i]}{L \times H} \quad (2.18)$$

$$\mu_2(S_{Otsu}) = \frac{1}{q_2(S_{Otsu})} \sum_{i=S_{Otsu}+1}^{255} i \frac{Hist[i]}{L \times H} \quad (2.19)$$

Le seuil Otsu se calcule uniquement à partir de l'histogramme normalisé de l'image pour que le procédé soit indépendant de la taille de l'image. La relation entre les variances intra-classe et inter-classes peut être utilisée pour faciliter le calcul. La moyenne totale de l'image

μ s'écrit quelque soit S_{Otsu} dans $[0,255]$ selon l'équation 2.20.

$$\mu = q_1(S_{Otsu})\mu_1(S_{Otsu}) + q_2(S_{Otsu})\mu_2(S_{Otsu}) \quad (2.20)$$

La variance totale de l'image σ^2 s'écrit par l'équation .

$$\sigma^2 = \sum_{i=0}^{255} i^2 \frac{Hist[i]}{L \times H} - \mu^2 \quad (2.21)$$

En se basant sur la variance intra-classes σ_w^2 (Equation 2.13) et sur la variance inter-classes donnée par l'équation suivante :

$$\sigma_b^2 = q_1(1 - q_1)(\mu_1 - \mu_2)^2 \quad (2.22)$$

on peut approximer la variance totale σ^2 par la relation suivante :

$$\sigma^2 = \sigma_w^2 + \sigma_b^2 \quad (2.23)$$

Ainsi minimiser la variance intra-classe revient à maximiser la variance inter-classe. Cette variance inter-classe est calculée pour chaque seuil S_{Otsu} possible entre 0 et 255. Le seuil Otsu correspond à la variance inter-classe maximale.

Les étapes de calcul du seuil Otsu dans le cadre de notre proposition d'extraction hybride de texte sont représentées dans l'algorithme 3. Dans ce contexte, on applique une transformation gamma à l'image d'entrée selon la formule de la ligne 4. D'abord on calcule l'histogramme normalisé de l'image transformée $Hist$ (Algorithme 3 : lignes 1-6). A partir de cet histogramme, on calcule les proportions des pixels (q) et leurs moyennes (μ) dans chaque classe (Algorithme 3 : lignes 14-15). A partir de ces paramètres on calcule pour chaque classe la variance inter-classe σ_b^2 pour chercher la valeur maximale (Algorithme 3 : lignes 16-23). Cette valeur correspond au seuil Otsu cherché S_{Otsu} (Algorithme 3 : ligne 24). La complexité totale de cet algorithme pour Nb_{Images} images est de l'ordre de $\mathcal{O}(Nb_{Images} \times (Nb_{pixels} + N_g))$.

A partir des caractéristiques de l'énergie, du contraste et du seuil Otsu un tableau d'instances Tab_{Ins} est construit. Ce tableau présente 100 instances Ins . Chaque instance donne une valeur gamma et les valeurs de caractéristiques correspondantes (tableau 2.1). A partir de ce tableau on estime la valeur gamma optimale pour laquelle le contraste de l'image sera amélioré [71].

Estimation de la valeur gamma optimale Les valeurs de l'énergie et du contraste de l'image originale qui possède la valeur gamma égale à 1 sont examinées pour estimer la valeur gamma optimale. Les pics dans les courbes de l'énergie, du contraste et du seuil Otsu sont notés respectivement par $Pe = 0.05$, $Pc = 1000$ et $Ps = 0.5$.

Trois règles sont employées pour estimer la valeur gamma optimale [196]. La 1^e règle est appliquée si l'énergie est supérieure à Pe . Sinon si l'énergie est strictement inférieure à Pe ,

Algorithme 3 Calcul du Seuil Otsu

Entrée : I : Image en niveau de gris de largeur L et de hauteur H
Sortie : S_{Otsu} : Seuil Otsu

```

1:  $Hist[i] := 0$  pour tout  $i \in [0..N_g - 1]$ 
2: Pour  $x \in [0..L - 1]$  Faire
3:   Pour  $y \in [0..H - 1]$  Faire
4:      $val := I(x, y)^\gamma$ 
5:      $Hist[val] ++$ 
6:   Fin Pour
7: Fin Pour
8:  $q_0 := Hist[0]/(L \times H)$ 
9:  $\mu_0 := 0$ 
10: Pour  $i \in [0..N_g - 1]$  Faire
11:    $q_i := q_{i-1} + Hist[i]/(L \times H)$ 
12:    $\mu_i := \mu_{i-1} + i \times Hist[i]/(L \times H)$ 
13: Fin Pour
14:  $S_{Otsu} := 0$ 
15:  $\sigma_{b,max} := 0$ 
16: Pour  $i \in [0..N_g - 1]$  Faire
17:   Si  $(q_i \neq 0) \wedge (q_i \neq 1)$  Alors
18:      $\sigma_{b,i} := (\mu_{255} \times q_i - \mu_i)^2 / q_i \times (1 - q_i)$ 
19:   Sinon
20:      $\sigma_{b,i} := 0$ 
21:   Fin Si
22:   Si  $(\sigma_{b,i} > \sigma_{b,max})$  Alors
23:      $\sigma_{b,max} := \sigma_{b,i}$ 
24:      $S_{Otsu} := i$ 
25:   Fin Si
26: Fin Pour

```

la 2^e règle est appliquée au cas où le contraste est supérieur à P_c , et la 3^e règle est appliquée au cas où le contraste est strictement inférieur à P_c .

L'algorithme 4 explique la 1^e règle. Si la valeur de l'énergie $E \geq P_e$ (Algorithme 4 : ligne 1), on cherche une instance à partir du tableau 2.1 où la valeur du seuil Otsu maximal est égale à (P_s) (Algorithme 4 : ligne 2).

Si plusieurs instances sont trouvées, on sélectionne celle qui possède une valeur maximale de contraste égale à (P_c) et d'énergie $E \geq P_e$ (Algorithme 4 : lignes 3-5). Si aucune instance n'est trouvée, on cherche la valeur du seuil S_{Otsu} la plus proche de P_s . La valeur de gamma qui correspond à cette instance sélectionnée représente la valeur optimale de gamma (Algorithme 4 : lignes 8-10).

La complexité de l'algorithme 4 est de l'ordre de $\mathcal{O}(Nb_{Images})$.

L'algorithme 5 détaille la 2^e règle d'estimation de la valeur gamma optimale. Si la valeur

TABLE 2.1: Caractéristiques de textures et de seuils Otsu calculés à partir de 100 valeurs gamma d'une image réunies dans un tableau d'instances Tab_{Ins}

Inst	Gamma	Contraste	Énergie	Seuil Otsu
1	0.1	20.45	0.005404	0.90588
2	0.2	67.96	0.001928	0.82353
...	...	...	...	...
53	5.3	472.4	0.10082	0.25098
54	5.4	474.4	0.10321	0.25098
55	5.5	475.8	0.105	0.24706
56	5.6	466	0.1099	0.24706
...	...	...	...	...
99	9.9	352.7	0.4772	0.32941
100	10	350.9	0.48	0.33333

de l'énergie $E < Pe$ et la valeur de contraste $C \geq Pc$ (Algorithme 5 : ligne 1), on cherche une instance qui possède la valeur d'énergie $E \geq 0,1$, la valeur de contraste $C \geq Pc$ et le seuil valeur de Ps à partir de la rangée gamma de 1 à 10 (Algorithme 5 : ligne 2). Si plus d'une instance est trouvée, sélectionner une instance dont la valeur de l'énergie maximale est égale à (P_e) et la valeur de Contraste est supérieure à Pc (Algorithme 5 : lignes 3-5). Si un tel cas n'est pas trouvé, on cherche un candidat entre les valeurs gamma de 0,1 et 0.9. La valeur du seuil doit être la plus proche de Ps . La valeur de gamma correspondant à cette instance est la valeur optimale (Algorithme 5 : lignes 8-10).

Dans le cas de l'algorithme 5 on traite seulement les $Nb_{Images} - 10$ images ayant des valeurs gamma entre 1 et 10 ($\gamma \in [1, 10]$) (Algorithme 5 : lignes 2-3). Ainsi la complexité totale de l'algorithme est de l'ordre de $\mathcal{O}(Nb_{Images} - 10)$.

La 3^e règle est représentée à travers l'algorithme 6. En effet, Si la valeur de l'énergie $E < Pe$ et la valeur de Contraste $C < Pc$ (Algorithme 6 : ligne 1), on cherche une instance I qui possède une valeur d'énergie $E \geq 0.1$ (Algorithme 6 : ligne 2), la valeur maximale de contraste doit être supérieure à 100 pour les valeurs gamma entre 1 et 10 (Algorithme 6 : lignes 3-5). Si aucune instance n'est trouvée, on cherche une instance entre les valeurs gamma de l'intervalle $[0.1, 1]$ de telle sorte que la valeur du seuil devrait être plus proche de Ps . La valeur gamma de cette instance sélectionnée est la valeur optimale (Algorithme 6 : lignes 8-10).

La complexité de l'algorithme 6 pour Nb_{Images} images est de l'ordre de $\mathcal{O}(Nb_{Images})$. Il est clair que le facteur principal de la haute complexité des algorithmes de la méthode de correction gamma est le nombre d'images Nb_{Images} qui est toujours égal à 100.

Dans la suite, on propose de réduire la complexité de la méthode de correction gamma en réalisant une prédiction d'un sous-ensemble plus petit que l'ensemble de 100 images. Ainsi, dans cette optimisation, le calcul de la valeur gamma optimale est réalisé sur un nombre d'images relativement réduit.

Algorithme 4 Règle 1 pour l'estimation de la valeur Gamma : Cas où l'énergie E est supérieure au pic de l'énergie Pe

Entrées : Tab_{Ins} : tableau d'instances, P_s , P_c , P_e

Sortie : γ_{opt} : Gamma optimale

```

1: Si ( $E \geq Pe$ ) Alors
2:   Trouver les instances  $Ins$  : ( $S_{Otsu} = Ps$ )
3:   Si (plusieurs instances  $Ins$ ) Alors
4:     selectionner une instance  $max(C)$  et ( $E \geq Pe$ )
5:     Retourner la valeur gamma optimale
6:   Sinon Si (Une seule instance) Alors
7:     Retourner la valeur gamma optimale
8:   Sinon Si (Pas d'instances) Alors
9:     Trouver  $S_{Otsu}$  le plus proche possible de  $Ps$ 
10:    Retourner la valeur gamma optimale
11:  Fin Si
12: Fin Si

```

Algorithme 5 Règle 2 pour l'estimation de la valeur Gamma : Cas où l'énergie E est inférieure au pic de l'énergie Pe et le contraste C est supérieur au pic du contraste Pc

Entrées : Tab_{Ins} : tableau d'instances, P_s , P_c , P_e

Sortie : γ_{opt} : Gamma optimale

```

1: Si ( $E < Pe$ ) et ( $C \geq Pc$ ) Alors
2:   Trouver les instances  $Ins$  : ( $E \geq 0.1$ ) et ( $C \geq Pc$ ) et ( $\gamma \in [1, 10]$ ) et ( $S_{Otsu} = Ps$ )
3:   Si (plusieurs instances  $Ins$ ) Alors
4:     selectionner une instance  $max(E)$  et ( $C \geq Pc$ )
5:     Retourner la valeur gamma optimale
6:   Sinon Si (Une seule instance) Alors
7:     Retourner la valeur gamma optimale
8:   Sinon Si (Pas d'instances) Alors
9:     Selectionner une instance dans l'intervalle  $\gamma \in [0.1, 0.9]$  avec un seuil  $S_{Otsu}$  le plus proche possible de  $Ps$ 
10:    Retourner la valeur gamma optimale
11:  Fin Si
12: Fin Si

```

3.2.2 Optimisation de l'algorithme de correction gamma

La quantité de calcul requise pour la correction gamma est considérable d'après sa complexité algorithmique calculée dans la section précédente. Nous proposons d'accélérer ce traitement en réduisant la quantité de calcul et en conservant la qualité des résultats.

Pour ceci, nous proposons une optimisation algorithmique qui prédit une sous-rangée de valeurs gamma ou la valeur optimale est supposée être trouvée par la méthode originale.

Algorithme 6 Règle 3 pour l'estimation de la valeur Gamma : Cas où l'énergie E est inférieure au pic de l'énergie Pe et le contraste C est inférieur au pic du contraste Pc

Entrées : Tab_{Ins} : tableau d'instances, P_s , P_c , P_e

Sortie : γ_{opt} : Gamma optimale

```

1: Si ( $E < Pe$ ) et ( $C < Pc$ ) Alors
2:   Trouver les instances  $Ins$  : ( $E \geq 0.1$ )
3:   Si (plusieurs instances  $Ins$ ) Alors
4:     sélectionner une instance  $max(C) > 100$  pour tout  $\gamma \in [1, 10]$ 
5:     Retourner la valeur gamma optimale
6:   Sinon Si (Une seule instance) Alors
7:     Retourner la valeur gamma optimale
8:   Sinon Si (Pas d'instances) Alors
9:     Sélectionner une instance dans l'intervalle  $\gamma \in [0.1, 1]$  avec un seuil  $S_{Otsu}$  le
       plus proche possible de  $P_s$ 
10:    Retourner la valeur gamma optimale
11:   Fin Si
12: Fin Si

```

Dans la suite nous nommons notre Accélération de la Méthode de Correction Gamma par (AGCM).

L'AMCG utilise un ensemble de règles issues d'une analyse comportementale des paramètres. Elle est expliquée dans la suite.

Étude comportementale des paramètres utilisés dans GCM Nous étudions le comportement des courbes d'énergie, de contraste, de seuil Otsu et des valeurs gamma optimales. Cette étude permet de construire des relations entre les comportements de ces paramètres et des sous-rangées des valeurs gamma susceptibles de contenir la valeur gamma optimale.

La figure 2.16 illustre les courbes de contrastes, de l'énergie et de seuil Otsu calculées par la méthode GCM. Ces courbes sont construites à partir des 100 valeurs gamma calculées par GCM. En fonction des pics et des changements brusques dans la courbe de contraste, chaque image est classée comme ayant un contraste strictement croissant, décroissant ou constant. Les changements brusques sont appelés discontinuités. Ils représentent un changement radical dans le rendu de l'image résultante. Ces courbes montrent de façon claire le comportement des caractéristiques et nous permettent d'extraire les relations entre la courbe et la valeur gamma optimale, non pas par des critères numériques, mais plutôt par la forme de la courbe.

Une courbe de contraste strictement croissante (voir figure 2.16a) possède une valeur gamma optimale égale à 10 et signifie généralement que l'image possède des caractéristiques de seuil Otsu, d'énergie et de contraste qui varient légèrement. Dans une courbe strictement décroissante, en général, la valeur gamma optimale se situe à proximité de son point de pic. Ce

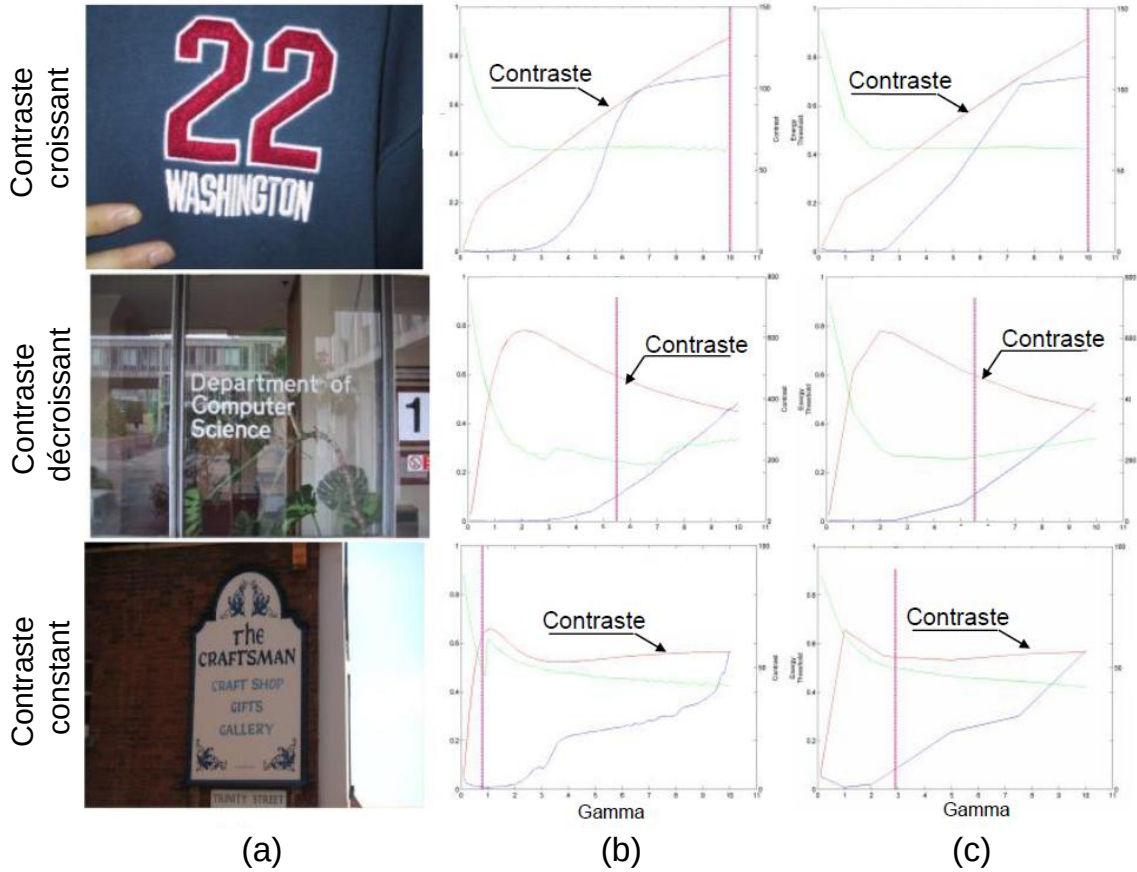


FIGURE 2.16: Classifications des images et les courbes de caractéristiques correspondantes : (a) : Images originales, (b) Courbes de caractéristiques de GCM ; (c) : Courbes de caractéristiques échantillonnées. Courbe de contraste (rouge), Énergie (bleu), seuil d'Otsu (le vert) et la valeur gamma optimale (ligne verticale magenta).

pic est toujours situé dans l'intervalle $\gamma = [0.9, 3.5]$ [196]. (Voir figure 2.16b). Une courbe de contraste constante (voir figure 2.16c) désigne une image qui est plus influencée par le comportement de la courbe de l'énergie que par le comportement de la courbe de contraste. Dans ce cas, si la courbe d'énergie est strictement croissante cela indique la présence d'une faible valeur gamma.

En outre, une courbe échantillonnée peut produire une version simplifiée et fiable des courbes tout en évitant le calcul des valeurs qui ne sont pas des candidats pour la valeur gamma optimale.

Échantillonnage de la courbe Nous proposons de calculer les paramètres de contraste, de l'énergie et de seuil d'Otsu sur des points particuliers. Ces points correspondent à quelque valeurs gamma parmi les 100 valeurs possibles. Ils sont donnés par l'ensemble γ suivant :

$$\gamma = [0.1, 1.0, 2.0, 2.5, 5.0, 7.5, 10.0]$$

Ces valeurs ont été choisies parmi les valeurs gamma qui définissent le comportement des courbes de paramètres de sorte que la courbe entière peut être classée de façon fiable comme dans la méthode de correction gamma originale.

La figure 2.16 montre trois échantillons d'images représentatives des trois classes de la base d'image et leurs courbes continues de caractéristiques. Les courbes échantillonnées peuvent être comparées aux courbes continues où nous pouvons voir que les allures des courbes extraites sont entièrement conservées.

Prédiction de la sous-rangée Gamma Les relations fournies par l'étude comportementale peuvent être appliquées afin de prédire la sous-rangée où se trouve la valeur gamma optimale. Ces relations sont formalisées dans l'organigramme de la figure 2.17.

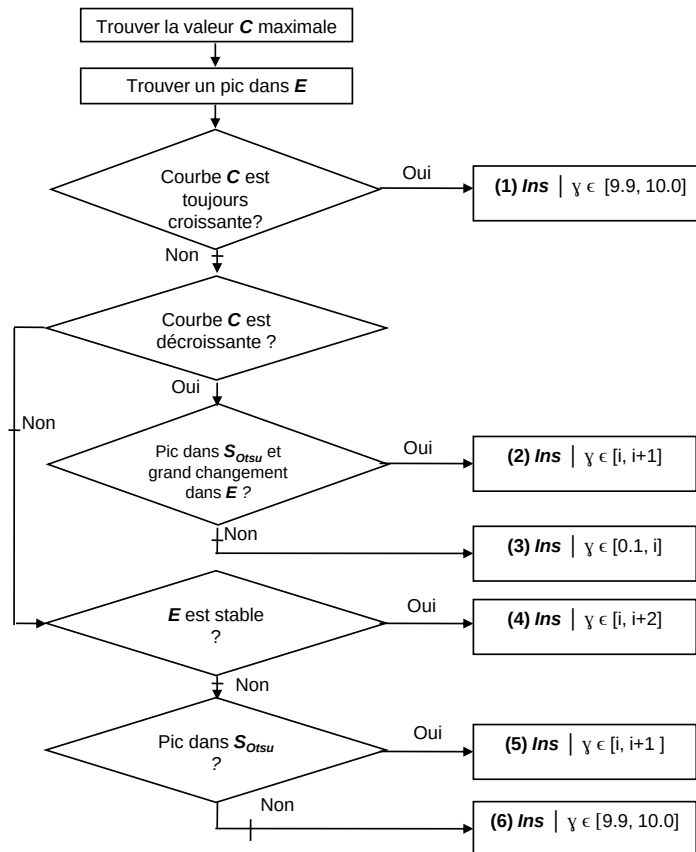


FIGURE 2.17: Prédiction de la sous-rangée de valeurs gamma. Avec C = contraste, E = énergie, S_{Otsu} = Seuil Otsu, Ins : intervalle contenant gamma optimale.

L'organigramme comporte 6 branches possibles numérotées de (1) à (6). Ces branches peuvent être regroupées en 3 principales : Dans la branche (1), la courbe de contraste est strictement croissante. Dans les branches (2) et (3), la courbe de contraste est décroissante.

Enfin pour les branches (4), (5) et (6) la courbe de contraste est constante.

En examinant les résultats de GCM, nous pouvons remarquer que les images qui présentent un fond lisse ou une image qui est sombre avec un peu de lumière produira une valeur gamma optimale égale à $\gamma_{opt} = 10.0$, alors que les images avec du texte clair et un fond hétérogène ont généralement une valeur gamma optimale qui appartient au milieu de toutes les valeurs considérées, mais elle ne se trouve jamais dans les limites supérieure ou inférieure de la plage des valeurs.

Une fois la valeur gamma optimale estimée, l'image en niveau de gris qui correspond à cette valeur gamma sera binarisée pour séparer le texte de l'arrière-plan. Pour assurer cette séparation, nous proposons de remplacer le seuillage Otsu de la méthode GCM [71] par une nouvelle classification hybride qui combine des approches locale et globale. Cette méthode sera appliquée également sur les régions de texte et non seulement sur les régions d'images. Nous appelons la correction gamma associée à la classification hybride basée sur Kmeans (CGCHK).

La sous-rangée prédite contiendra un nombre d'images largement inférieur à 100 images ($Nb'_{Images} \ll Nb_{Images}$) ce qui diminuera la complexité totale de la méthode.

3.2.3 Classification hybride basée sur le noyau Kmeans

Une fois les régions d'images sont améliorées par la transformation gamma optimale. Ces régions ainsi que celles de texte sont binarisées. Que se soit pour les régions d'images ou de textes, nous proposons pour cela une nouvelle méthode de binarisation basée sur l'algorithme de classification Kmeans [66]. Celui-ci est l'un des algorithmes de classification non supervisée le plus populaire. Il est principalement utilisé pour l'extraction de données [38] et le traitement d'images [39, 195]. En ce qui concerne le contexte du traitement d'images, Kmeans a pour but de diviser tous les pixels de l'image en k groupes distincts. Chaque groupe est représenté par la moyenne des valeurs de tous ses pixels. On appelle cette moyenne (Centroid).

L'algorithme de la méthode Kmeans est représenté sur la figure 2.18. Dans la première étape d'initialisation, le nombre et les valeurs des centroids sont fixés en fonction du besoin de l'application. Par exemple pour la binarisation on utilise deux centroids initialisés en noir et blanc. Nous notons par $\varphi = \{\varphi_0, \varphi_1, \dots, \varphi_i, \dots, \varphi_{k-1}\}$ l'ensemble de k centroids initiaux. La classification des pixels repose sur le calcul de leur similarité avec les centroids φ . Cette similarité est traduite par un calcul de distance. Pour une meilleure qualité de classification, la distance euclidienne est en général employée [40]. Etant donné un centroid φ_i et un pixel $I(x, y)$ la distance euclidienne est définie par l'équation 2.24.

$$d(\varphi_i, I(x, y)) = \sqrt{(\varphi_i^R - I^R(x, y))^2 + (\varphi_i^G - I^G(x, y))^2 + (\varphi_i^B - I^B(x, y))^2} \quad (2.24)$$

Avec φ_i et $I(x, y)$ sont représentés dans l'espace couleur RGB par : $\varphi_i = (\varphi_i^R, \varphi_i^G, \varphi_i^B)^T$ et $I(x, y) = (I^R(x, y), I^G(x, y), I^B(x, y))^T$, où $i \in [0, k - 1]$.

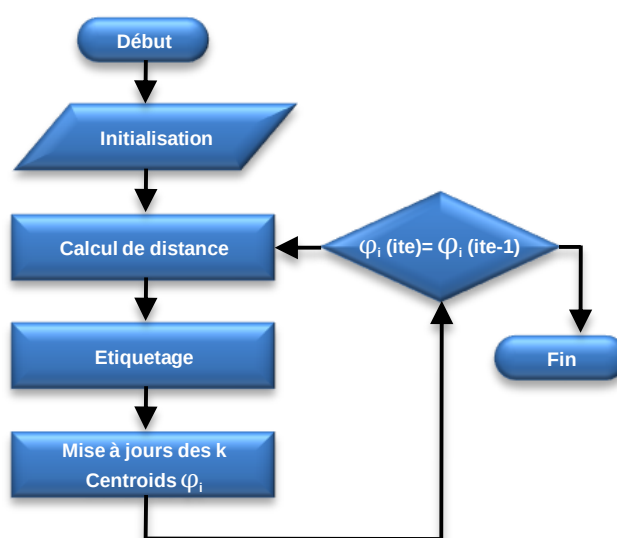


FIGURE 2.18: La méthode de classification Kmeans.

L'étape de l'étiquetage des pixels définit leurs appartenances aux centroids les plus proches en fonction des distances minimales calculées. En outre, les valeurs de pixels, et leurs nombres dans chaque classe i sont ajoutés respectivement à l'accumulateur et au compteur correspondants $Accu = \{Accu_0, Accu_1, \dots, Accu_i, \dots, Accu_{k-1}\}$, et $Count = \{Count_0, Count_1, \dots, Count_i, \dots, Count_{k-1}\}$ avec $Accu_i = (Accu_i^R, Accu_i^G, Accu_i^B)^T$.

Finalement, dans l'étape de mise à jours des k centroids, chaque centroid φ_i , $i \in [0, k - 1]$ est re-estimé en calculant la moyenne des pixels affectés à sa classe i . Ce processus se base sur la division des accumulateurs par les compteurs calculés précédemment. Les étapes de calcul de la distance, l'étiquetage et la mise à jour des centroids sont répétées jusqu'à ce que tous les pixels soient définitivement attribués à de nouvelles classes et les centroids ne changent plus de valeurs. Avant chaque itération, les accumulateurs et les compteurs sont remis à zéro.

En se basant sur la méthode Kmeans, nous proposons une nouvelle méthode d'extraction hybride de texte. Dans le reste du document, nous appelons cette méthode CHK (Classification Hybride basée sur Kmeans). Les phases globale et locale de la méthode proposée sont présentées dans la figure 2.19.

Tout d'abord, dans la phase locale, le document numérisé est divisée en Nb blocs. En effet, le nombre total de blocs est déterminé après la configuration de la taille de bloc, qui peut être réglé à petite, moyenne ou grande taille en fonction de la taille de caractères utilisés dans le document. Ensuite, l'algorithme Kmeans est appliqué indépendamment sur chaque bloc. La phase globale rassemble les résultats obtenus par le Kmeans local de manière itérative et reboucle jusqu'à convergence. Plus de détails de la méthode CHK sont décrits dans l'algorithme 7.

Deux centroids initiaux sont utilisés pour représenter le texte et l'arrière-plan. Nous les

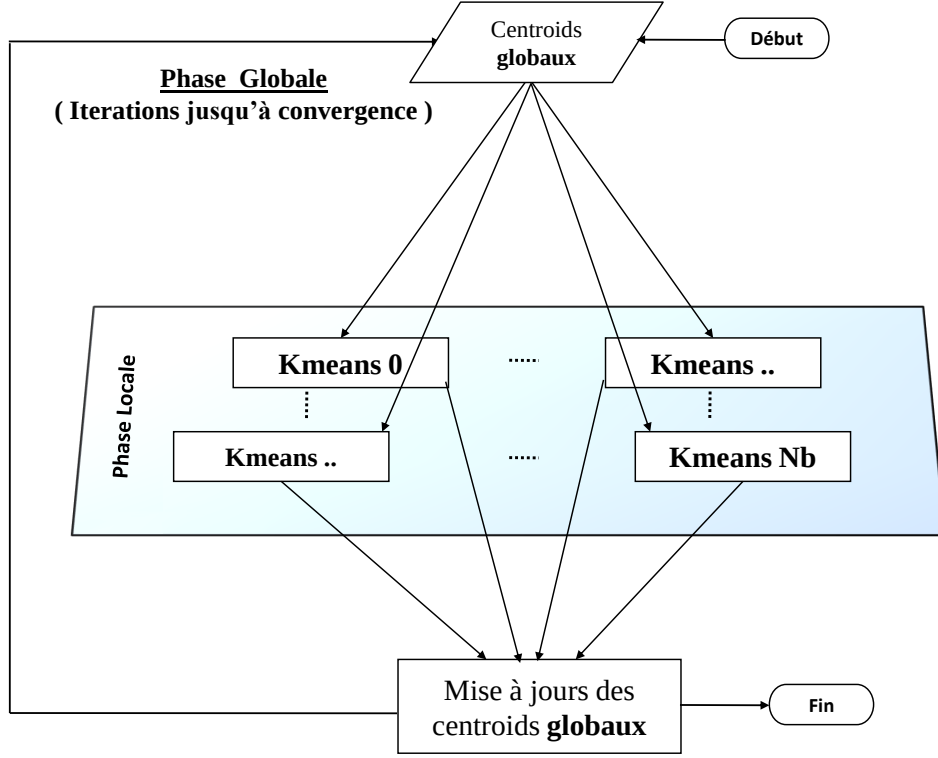


FIGURE 2.19: La méthode CHK : Les phases globale et locale.

notons par $\varphi_G = \{\varphi_{G0}, \varphi_{G1}\}$. On initialise φ_{G0} et φ_{G1} avec des valeurs en noir et blanc. Ainsi, $\varphi_{G0} = (0, 0, 0)^T$ et $\varphi_{G1} = (255, 255, 255)^T$ désignent respectivement les classes de l'arrière-plan et du texte (Algorithme 7 : Ligne 1). Étant donné que le document est divisé en Nb blocs de tailles égales, chaque bloc comprend deux centroids locaux donnés par l'ensemble :

$$\varphi_L = \{(\varphi_{L0,0}, \varphi_{L0,1}), (\varphi_{L1,0}, \varphi_{L1,1}), \dots, \varphi_{Lbl,i}, \dots, (\varphi_{L Nb-1,0}, \varphi_{L Nb-1,1})\}$$

avec $\varphi_{Lbl,i} = (\varphi_{Lbl,i}^R, \varphi_{Lbl,i}^G, \varphi_{Lbl,i}^B)^T$, $i \in [0, 1]$, et $bl \in [0, Nb - 1]$. Au début, chaque centroid local est initialisé avec le centroid global correspondant : $\varphi_{Lbl,0} = \varphi_{G0}$ et $\varphi_{Lbl,1} = \varphi_{G1}$, $bl \in [0, Nb - 1]$. En plus, chaque bloc possède deux accumulateurs et deux compteurs locaux donnés respectivement par les ensembles $Accu_L$ et $Count_L$ données respectivement comme suit :

$$Accu_L = \{(Accu_{L0,0}, Accu_{L0,1}), (Accu_{L1,0}, Accu_{L1,1}), \dots, \\ Accu_{Lbl,i}, \dots, (Accu_{L Nb-1,0}, Accu_{L Nb-1,1})\}$$

$$Count_L = \{(Count_{L0,0}, Count_{L0,1}), (Count_{L1,0}, \\ Count_{L1,1}), \dots, Count_{Lbl,i}, \dots, (Count_{L Nb-1,0}, \\ Count_{L Nb-1,1})\}$$

avec $Accu_{Lbl,i} = (Accu_{Lbl,i}^R, Accu_{Lbl,i}^G, Accu_{Lbl,i}^B)^T$ où $i \in [0, 1]$ and $bl \in [0, Nb - 1]$.

Algorithme 7 CHK : Classification Hybride basée sur Kmeans

Entrée : I : Image
Sortie : I_b : Image binaire

```

1:  $\varphi_{G0} := (0, 0, 0)^T, \varphi_{G1} := (255, 255, 255)^T$ 
2: Tant que  $\varphi_{Gi}(ite++) \neq \varphi_{Gi}(ite)$  Faire
3:   Pour  $b_l \in [0..Nb - 1]$  Faire
4:      $I_b = Kmeans[I(b_l, \varphi_G)]$ 
5:      $Accu_G := 0$ 
6:      $Count_G := 0$ 
7:     Pour  $i \in [0, 1]$  Faire
8:        $Accu_{Gi} := Accu_{Gi} + Accu_{Lbl,i}$  Eq.2.25
9:        $Count_{Gi} := Count_{Gi} + Count_{Lbl,i}$ 
10:    Fin Pour
11:  Fin Pour
12:   $\varphi_{Gi} = \frac{Accu_{Gi}}{Count_{Gi}}; \quad i \in [0, 1]$  Eq.2.26
13: Fin Tant que

```

Comme indiqué dans la ligne 4 de l'algorithme 7, l'algorithme Kmeans est appliqué à travers tous les blocs du document numérisé. Lorsque les Kmeans convergent dans chaque bloc, tous les accumulateurs locaux sont additionnés dans un accumulateur global : $Accu_G = \{Accu_{G0}, Accu_{G1}\}$ avec, $Accu_{Gi} = (Accu_{Gi}^R, Accu_{Gi}^G, Accu_{Gi}^B)^T, i \in [0, 1]$. La ligne 7 de l'algorithme 7 illustre le calcul des accumulateurs globaux. Ce processus est effectué par l'addition de chaque composante de couleur donnée par l'équation 2.25 :

$$Accu_{Gi} = \begin{pmatrix} Accu_{Gi}^R + Accu_{Lbl,i}^R \\ Accu_{Gi}^G + Accu_{Lbl,i}^G \\ Accu_{Gi}^B + Accu_{Lbl,i}^B \end{pmatrix}, \quad i \in [0, 1], \quad (2.25)$$

$bl \in [0, Nb - 1]$

De même, les compteurs locaux de chaque bloc sont accumulés dans les compteurs globaux donnés par $Count_G = \{Count_{G0}, Count_{G1}\}$. Finalement, les accumulateurs globaux $Accu_G$ sont divisés par les compteurs correspondants $Count_G$ pour calculer les nouveaux centroids φ_{Gi} comme indiqué dans l'équation 2.26 :

$$\varphi_{Gi} = \begin{pmatrix} \varphi_{Gi}^R \\ \varphi_{Gi}^G \\ \varphi_{Gi}^B \end{pmatrix} = \begin{pmatrix} \frac{Accu_{Gi}^R}{Count_{Gi}} \\ \frac{Accu_{Gi}^G}{Count_{Gi}} \\ \frac{Accu_{Gi}^B}{Count_{Gi}} \end{pmatrix}, \quad i \in [0, 1] \quad (2.26)$$

Les centroids de chaque bloc sont réinitialisés avec les centroids globaux φ_{G0} de et φ_{G1} . Si la convergence globale est atteinte, l'algorithme CHK s'arrête.

Suite à la binarisation des effets de bords peuvent apparaitre. On appelle ceci distorsion. Dans la suite, nous démontrons l'efficacité de la méthode CHK qui réduit le critère de distorsion.

En général, pour évaluer l'efficacité de la classification Kmeans, la mesure de l'erreur quadratique moyenne est employée comme un indicateur de performance [41, 42]. Ce critère est définie par l'équation 2.27 :

$$distorsion = \frac{1}{Np} \sum_{i=0}^{Np-1} |I(x, y) - \varphi(I(x, y))|^2 \quad (2.27)$$

Np désigne le nombre de pixels de l'image I de hauteur H et de largeur L . On note avec $\varphi(I(x, y))$ les centroids associés aux pixels $I(x, y)$, avec $x \in [0, L - 1]$ et $y \in [0, H - 1]$.

Pour comparer les résultats, nous concevons une méthode de classification basée sur un Kmeans Global (KG). Dans cette méthode, Kmeans est exécuté sur tout le document et non pas localement en blocs. Les centroids initiaux des méthodes KG et CHK sont initialisés aux valeurs RGB $\varphi_{G0} = (0, 0, 0)^T$ et $\varphi_{G1} = (255, 255, 255)^T$. Pour évaluer les performances de classification de ces deux méthodes, nous mesurons le critère de distorsion de KG et CHK sur la base de documents LRDE-DBD [69].

La figure 2.20 illustre le cumul des distorsions des deux méthodes de classification KG et CHK sur 125 documents scannés. Le résultat obtenu montre que CHK améliore à peu près $3\times$ la mesure de distorsion par rapport à KG. En effet, le traitement de Kmeans sur de petites régions (traitement local), réduit les distorsions localement ce qui minimise la distorsion globale. Nous pouvons conclure donc que notre approche améliore la qualité de classification tout en réduisant la distorsion.

La complexité de l'algorithme CHK est de l'ordre de $\mathcal{O}(Nb_{itCHK} \times Nb_{blocs} \times Nb_{pixels} \times Nb_{centroids} \times N_{itKmeans})$ avec Nb_{itCHK} : le nombre d'itérations globales, Nb_{blocs} : nombre de blocs dans l'image, Nb_{pixels} est le nombre de pixels par bloc, $Nb_{centroids}$ est le nombre de centroids par bloc, et $N_{itKmeans}$ représente nombre d'itérations Kmeans par bloc. L'algorithme CHK proposé est complexe, notamment à cause de son noyau Kmeans [66].

A ce niveau, les régions binaires produites peuvent être incorrectes (texte blanc sur arrière-plan noir). Ainsi, un traitement supplémentaire est nécessaire pour produire le rendu couleur correcte.

3.2.4 Correction du rendu couleur basée sur l'étude de la densité de pixels

Plusieurs méthodes de binarisation [93, 68, 70] supposent un texte sombre sur un fond clair. Sinon, ils ne parviennent pas à produire les couleurs binaires (noir et blanc) correctes. Pour surmonter ce problème, certains procédés contrôlent la couleur du texte et du fond. Par exemple, Clark [80] utilise une méthode simple basée sur l'hypothèse que le nombre de

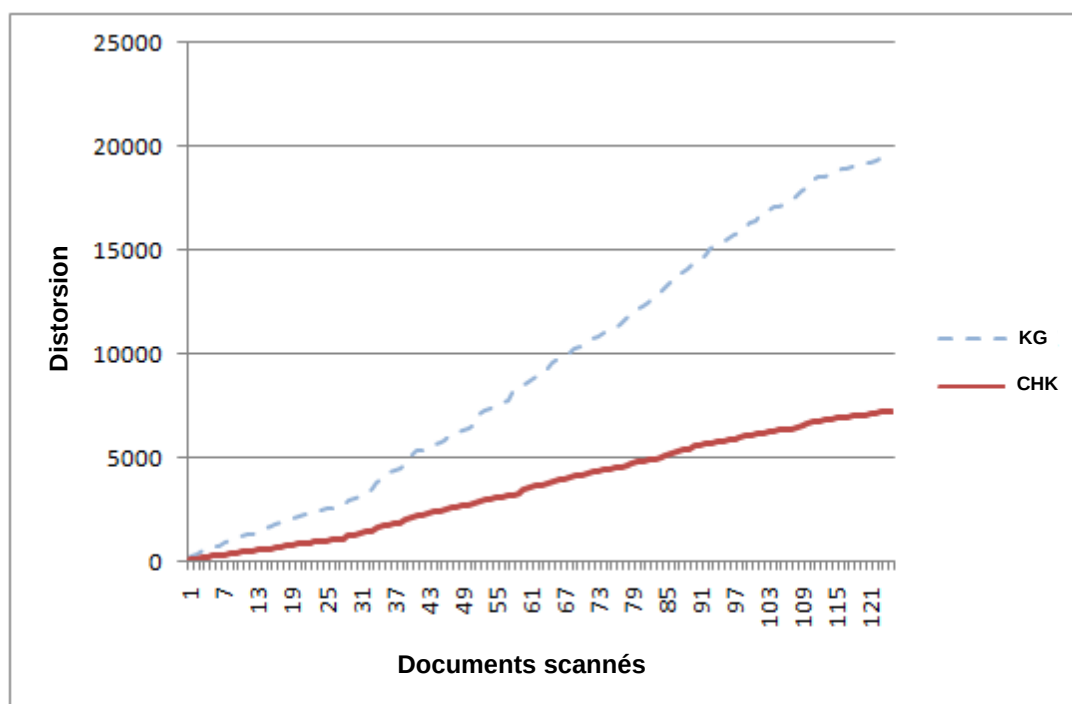


FIGURE 2.20: Cumul de la distorsion calculée à partir des résultats de binarisation des méthodes KG et CHK.

pixels de fond est supérieur au moins à un pixel. Cette méthode échoue si des caractères épais occupant une surface importante dans la fenêtre utilisée. Ainsi, nous appliquons le même concept de technique basée sur la densité de pixels sur les régions extraites. Nous contrôlons la polarité des couleurs de l'avant et de l'arrière-plan selon l'équation 2.28.

$$\begin{cases} I(x, y) & \text{Si } N_{Bp} > N_{Wp} \\ \overline{I(x, y)} & \text{Sinon} \end{cases} \quad (2.28)$$

avec $I(x, y)$ désigne un pixel, $i \in [0, N_{Bp} + N_{Wp}]$, et N_{Np} , N_{Bp} représentent respectivement le nombre de pixels noirs et blancs dans la région étudiée.

La complexité de cet algorithme dépend du nombre de pixels de l'image Nb_{pixels} d'où la complexité de l'algorithme est de l'ordre de $\mathcal{O}(Nb_{pixels})$.

On peut visualiser le résultat de la binarisation par la reconstruction du document, en copiant les régions d'images et de texte respectivement binarisées par les méthodes CGCHK et CHK dans un seul document. La copie est réalisée en les parcourant pixel par pixel à partir des leurs coordonnées d'origine. Suite à cette étape nous obtenons un document binaire.

Ce traitement sert à visualiser et à évaluer le résultat de binarisation de la méthode EHT proposée.

3.3 Localisation des caractères par dissection

Dans notre extraction de texte, nous utilisons la méthode de dissection de Tesseract [43] pour localiser les caractères qui fait partie de la méthode d'analyse structurelle de document PLA [56] .

Une fois les régions de texte séparées de l'arrière-plan dans les régions de texte simple et d'images, les lignes de texte sont identifiées par l'algorithme de recherche de lignes (*Line Finding*) [132] employé dans l'algorithme d'analyse de structure du document PLA [56].

Les lignes de base (baseline) sont modélisées selon l'inclinaison du texte de façon à gérer le texte incliné. Cette modélisation est effectuée en groupant les composantes connexes qui sont sur les parties les plus rectilignes de la ligne de base d'origine. Des partitions de composantes connexes organisées en lignes sont ainsi générées.

Les lignes de texte construites sont segmentées en mots de manière différente selon l'espacement entre les caractères. Les lignes de texte qui contiennent des caractères équidistants (ceux qui possèdent une marge fixe) sont séparées directement en caractères en respectant la marge fixe. La figure 2.21 illustre un exemple de mot avec des caractères équidistants.



FIGURE 2.21: Localisation des caractères par Tesseract [43].

Une ligne de texte qui comporte des marges non fixes et aléatoires est divisée en mots en utilisant des marges aléatoires. Une phase de résolution des marges non fixes est réalisée au moment de la reconnaissance des caractères.

4 Expérimentations

La méthode EHT proposée a été évaluée sur les bases d'images LRDE-DBD¹ [69] et IC-DAR [81]. La base LRDE-DBD est composée de 125 documents de texte français extraits du magazine "Le Nouvel Observateur"². L'ensemble des données fournies par la base est

1. Copyright(c) 2012. EPITA and Development Laboratory (LRDE) with permission from Le Nouvel Observateur. LRDE-DBD est disponible sur le site web : [http : //www.lrde.epita.fr/cgi - bin/twiki/view/Olena/DatasetDBD](http://www.lrde.epita.fr/cgi-bin/twiki/view/Olena/DatasetDBD)

2. Le Nouvel Observateur. Issue 2402, November 18-24, 2010 et disponible sur le site web : [http : //tempsreel.nouvelobs.com](http://tempsreel.nouvelobs.com)

composé d'images au format A4 dont la résolution est de 300 dpi. Notre évaluation a été effectuée sur les 125 documents numérisés. Ces documents sont légèrement dégradés à cause du bruit généré lors du processus de la numérisation. Nous utilisons également les 125 images binarisées et sorties texte (OCR) de la base LRDE comme vérité terrain.

Dans les expérimentations qui suivent, on fixe la taille de bloc à 32×32 pixels pour la méthode CHK car cette taille donne une qualité de binarisation acceptable sur la base LRDE-DBD [103]. Nous utilisons la métrique F-measure pour évaluer l'exactitude de notre binarisation. La F-measure d'un système est mesurée en fonction du Rappel (R) et de la Précision (P) du système étudié selon l'équation 2.29. Le rappel (R) est défini par le nombre de pixels pertinents correctement classifiés au regard du nombre de pixels pertinents contenus dans l'image étudié. La précision (P) représente le nombre de pixels pertinents correctement classifiés rapporté au nombre de pixels total dans l'image (2.31).

$$Fmeasure = \frac{2PR}{P + R} \quad (2.29)$$

avec

$$R = \frac{Nb_{pixels} \text{ pertinents correctement classifié}}{Nb_{pixels} \text{ pertinents contenus dans l'image}} \quad (2.30)$$

et

$$P = \frac{Nb_{pixels} \text{ pertinents correctement classifiés}}{Nb_{pixels} \text{ total dans l'image}} \quad (2.31)$$

En outre, Tesseract 3.02 a été utilisée pour effectuer l'étude de la reconnaissance de caractères. Enfin, nous évaluons notre approche d'extraction hybride de texte quelque soit le type du document.

4.1 Binarisation hybride de texte

La qualité de binarisation de texte influe directement sur le résultat de la reconnaissance de caractères. Dans ce contexte, Tesseract 3.02 [43] est utilisé pour lire le texte extrait à l'aide de notre méthode CHK. Actuellement, nous comparons le taux de reconnaissance de caractère basé sur les documents binaires de texte générés par notre méthode CHK avec plusieurs méthodes de binarisation [69, 68, 34, 65, 93, 44, 45] sur les mêmes documents LRDE-DBD. Le tableau 2.2 illustre le taux de reconnaissance des différentes méthodes évaluées.

La méthode CHK donne le meilleur taux de reconnaissance avec 91% de précision OCR grâce à sa binarisation robuste (Figure 2.22). D'autre part, la méthode Sauvola Ms_{kx} donne un taux acceptable de reconnaissance de 89%. Elle assure ainsi une extraction efficace de texte, mais dans certaines régions du document, les artefacts peuvent apparaître conduisant à une mauvaise reconnaissance comme on le voit dans la figure 2.23.

Dans la suite, nous comparons le taux de reconnaissance de la méthode GCM [71] et la méthode CGCHK proposée afin de montrer son efficacité lors de la séparation de texte dans les images de scènes naturelles. Nous rappelons que cette méthode améliore l'image en utilisant la méthode de correction gamma GC associée avec la méthode de binarisation CHK.

TABLE 2.2: Taux de reconnaissance des caractères extraits par CHK et par plusieurs méthodes de binarisation

Méthodes	Taux de reconnaissance OCR (%)
CHK [70]	91
Sauvola Ms_{kx} [69]	89
Wolf [34]	88
Sauvola [68]	87
Lelore [44]	85
Otsu [65]	84
Niblack [93]	80
TMMS [45]	73

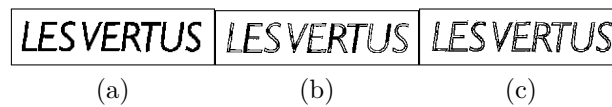


FIGURE 2.22: Comparaison visuelle de la qualité de binarisation. (a) : CHK ; (b) : Lelore et (c) : Tmms

Le tableau 2.3 compare GCM [71] et CGCHK sur des images de scène naturelles issues de la base ICDAR Robust Reading [81].

Nous notons que CGCHK donne un meilleur taux de reconnaissance de caractère soit 85,2%. Nous expliquons ce résultat par le fait que la méthode CHK améliore Otsu dans l'étape de binarisation de texte. En effet, la stratégie de traitement hybride employée prend en compte à la fois l'information locale et globale de pixel pour fournir un résultat plus efficace. Nous montrons un résultat visuel dans la figure 2.24. Comme nous pouvons le voir, le texte est mieux séparé de l'arrière-plan en utilisant la méthode CGCHK.

4.2 Optimisation algorithmique de la correction gamma

La base d'images ICDAR [81] a été utilisée comme entrée pour évaluer la méthode AGCM proposée pour l'accélération algorithmique de méthode de correction gamma. Cette base contient 113 images couleurs avec du texte.

Les méthodes GC et AGCM ont été exécutées sur un CPU Intel i3 à 3,07 GHz. D'une

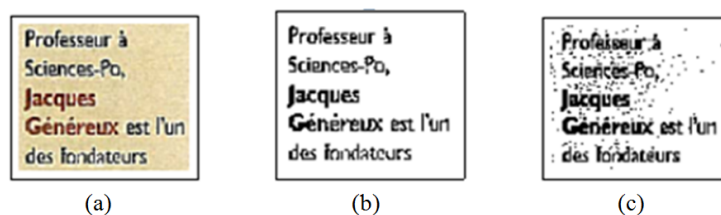


FIGURE 2.23: Comparaison des qualité de binarisation d'un crop de document à partir de la base LRDE-DBD (a), entre : (b) CHK et (c) Sauvola Ms_{kx} .

TABLE 2.3: Comparaison des méthodes de GCM et CGCHK basé sur OCR Précision pour les images de scènes de ICDAR

Méthodes	Taux de reconnaissance OCR (%)
GCM : GC + Otsu	84.4
CGCHK : GC + CHK	85.2

part, la prédiction de sous-rangées gamma de l'AGCM améliore considérablement le temps d'exécution de la méthode GC tout en préservant la qualité des résultats. D'autre part, le résultat de la méthode AGCM contient la même qualité des résultats que la méthode GCM, ce qui signifie que l'optimisation proposée est robuste et fiable (Figure 2.25, 1^e et 2^e images). Les règles de comportement de l'algorithme de prédiction de sous-rangées de gamma peut parfois conduire à une meilleure extraction de texte que la correction gamma CG. Un exemple de ce cas est présenté dans la figure 2.25.c. Ceci est principalement dû à l'analyse des pics dans les courbes d'énergie et de seuil. La figure 2.26 présente les temps d'exécution superposés des méthodes CG et AGCM sur l'ensemble des images de la base ICDAR [81].

Le temps d'exécution de l'AGCM ne peut être généralisé en une fraction de temps d'exécution de la méthode GC parce que la sous-rangée gamma prédite est de taille variable de 0,1 (Branches 1 et 6) à 2,0 (Branches 4). Cela fait que le nombre d'images avec un gamma modifié qui doivent être calculés et analysés est variable, tandis que dans la méthode GC on analyse toujours 100 images. En dépit de cette variabilité, le temps d'exécution de AGCM présente une moyenne d'accélération égale à 4,456x et dans le meilleur des cas, une accélération de 11.43x (Figure 2.26). Ces résultats sont significatifs et encourageant pour l'intégration du procédé dans un système temps réel.

4.3 Extraction hybride de texte dans les documents hétérogènes

Dans cette section nous évaluons notre méthode hybride de binarisation CHK visuellement puis en termes de taux de reconnaissance de caractères OCR.

D'abord nous validons la complémentarité entre les deux méthodes CHK et CGCHK pour la

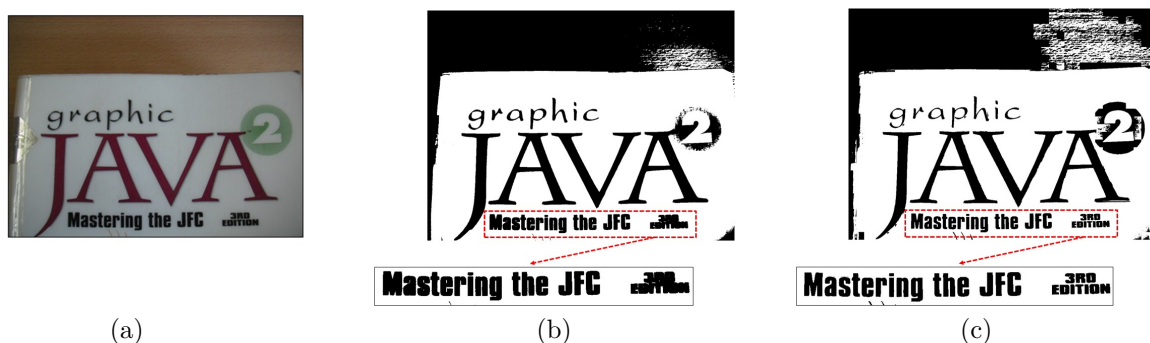


FIGURE 2.24: Binarisation avec les méthodes GCM [71] et CGCHK sur les images de texte de scènes naturelles issues de la base ICDAR. (a) : Image originale ; (b) : GCM ; (c) : CGCHK.

binarisation des documents hétérogènes. On appelle les images de texte sur fond homogènes par texte simple. Cette évaluation visuelle est illustrée dans la figure 2.27 .

Selon la figure 2.27 - (e, b - *), nous pouvons voir que CHK fournit une haute qualité d'extraction de texte sur des images de texte simples, où les caractères sont bien distingués de l'arrière-plan homogène. Cependant, cette extraction n'est pas précise sur des images de Scènes naturelles avec un arrière-plan complexe (voir la figure 2.27-(g, b)).

En réalité, CGCHK surpasse CHK sur des images de texte de Scène naturelles avec un fond complexe et produit une extraction textuelle plus efficace d'après la figure 2.27(g, c). Cependant, selon la figure 2.27 (e, c-*), CGCHK est moins précise que CHK sur des images de texte Simples. Ainsi, les caractères binaires se touchent légèrement en raison de la transformation gamma.

D'autre part, nous notons que les méthodes CGCHK et CHK inversent les couleurs du texte et de l'arrière-plan sur l'image de texte Simple (la figure 2.27-(e, c - **), la figure 2 - (e, b - **)) et sur des images de texte de Scène (la figure 2.27 - (f, c), la figure 2.27 - (f, b)).

Pour conclure, les méthodes CHK et GCM possèdent des avantages complémentaires en traitant le texte simple ou les images de scènes naturelles. En revanche la méthode GCM ne produit pas une bonne extraction sur les images de régions multiples.

Nous illustrons dans la figure 2.28 les résultats de binarisation de texte des méthodes CHK, CGCHK et EHT lors du traitement d'un document hétérogène de la base LRDE-DBD. Nous notons que EHT donne le meilleur résultat. En effet CHK produit visuellement une haute qualité de binarisation sur les régions de texte. Cependant, elle ne parvient pas à produire les couleurs appropriées du texte et du fond car il suppose que la couleur du texte est foncée et que le fond est en couleur clair. CGCHK réussit dans la binarisation des images. Cependant, elle engendre une perte d'information dans les régions de texte avec un fond homogène. Nous évaluons les méthodes CHK et EHT sur des images de texte simple et sur des documents hétérogènes. Nous démontrons à travers le tableau 2.4 la précision en terme de Fmeasure de la méthode EHT sur les documents LRDE-DBD qui ne comprennent que des informations de texte.

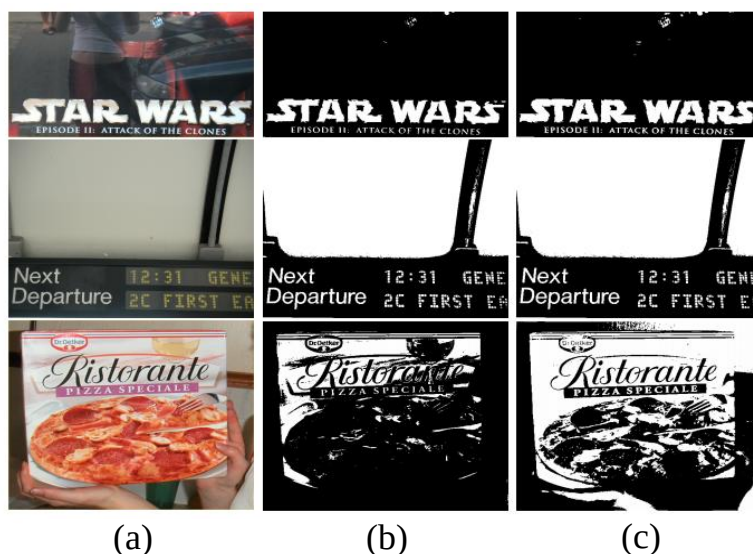


FIGURE 2.25: Comparaison de la qualité de CGCHK et de la méthode AGCM associée avec CHK. (a) : Images originales ; (b) : Binarisation avec la correction gamma CG ; (c) : Binarisation avec AGCM et CHK.

TABLE 2.4: Comparaison des méthodes CHK et EHT en terme de précision de binarisation (Fmeasure) sur 125 images de la base LRDE-DBD (document de texte simple)

Méthodes	FMeasure (%)
EHT [1]	98,84
CHK [70]	98,21

Comme attendu, EHT et CHK affichent des précisions Fmeasures proches sur des images de texte simple.

Tout de même, il est important de noter que la haute précision EHT est due à la technique d'étude de densité de pixels qui produit correctement les couleurs de texte et du fond. En outre, certaines régions de texte comprennent un fond non homogène, sur lequel, CGCHK donne une meilleure qualité de binarisation. Dans la suite, nous démontrons l'efficacité de notre proposition en termes de taux de reconnaissance de caractères sur les documents hétérogènes. Pour assurer cette évaluation, nous choisissons un ensemble de documents LRDE-DBD comprenant des images de texte et de scènes naturelles. Comme il n'existe pas de vérité terrain OCR pour ces documents, nous les construisons manuellement. La méthode EHT affiche un taux de reconnaissance de 98.5%. Ce taux élevé est dû à la bonne répartition du traitement entre les méthodes CHK et CGCHK. En outre, CGCHK permet de bien extraire

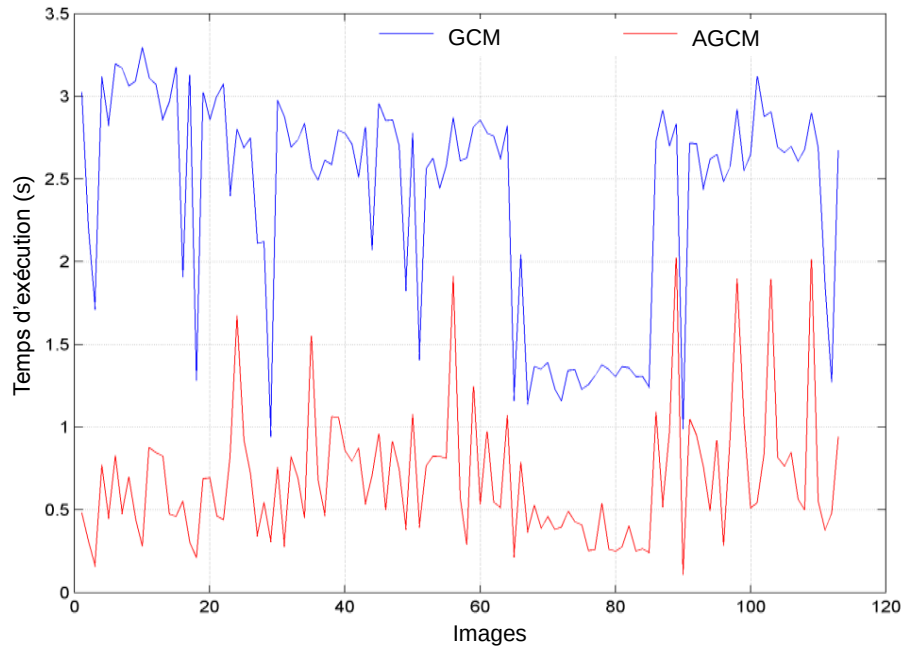


FIGURE 2.26: Comparaison des temps d'exécution de la méthode CG (bleu) et de la méthode AGCM (rouge) pour 113 images.

le texte des images de Scènes naturelles contrairement à CHK qui mélange les couleurs du fond et de la forme.

5 Conclusion

L'extraction de texte est une étape primordiale dans tout processus de reconnaissance de caractères. Cependant, il est difficile d'extraire de texte dans les documents hétérogènes. Pour ceci, nous avons proposé une nouvelle méthode d'extraction hybride de texte basée sur l'algorithme de classification Kmeans et de correction gamma (EHT). On utilise la correction gamma pour le contraste dans les arrières-plans complexes. De plus, nous avons proposé une méthode robuste pour l'extraction de texte. C'est une méthode de classification hybride qui combine des approches locales et globales et qui est basée sur l'algorithme Kmeans.

Suite à nos expérimentations, nous notons que EHT effectue une extraction de caractères robuste quelque soit les régions de texte avec des fonds homogènes ou complexes. Nous notons également que la méthode EHT est très adaptée pour le traitement parallèle grâce au non dépendance des données que présentent ces algorithmes. Ainsi, nous exploitons une architecture parallèle afin d'accélérer la méthode dans le chapitre 4.

Une fois extrait, le texte sera segmenté en caractères, puis, chaque caractère est décrit sous forme d'un vecteur caractéristique afin de l'identifier ultérieurement. Cette description doit être invariante aux différentes formes que peuvent avoir les caractères. Dans le chapitre 3 nous présentons notre proposition de description structurelle de caractères.



FIGURE 2.27: Évaluation visuelle de l'extraction de texte. (a). Simple et Multiple régions de documents scannées; (b) : Résultats CHK; (c) : Résultats GCM.

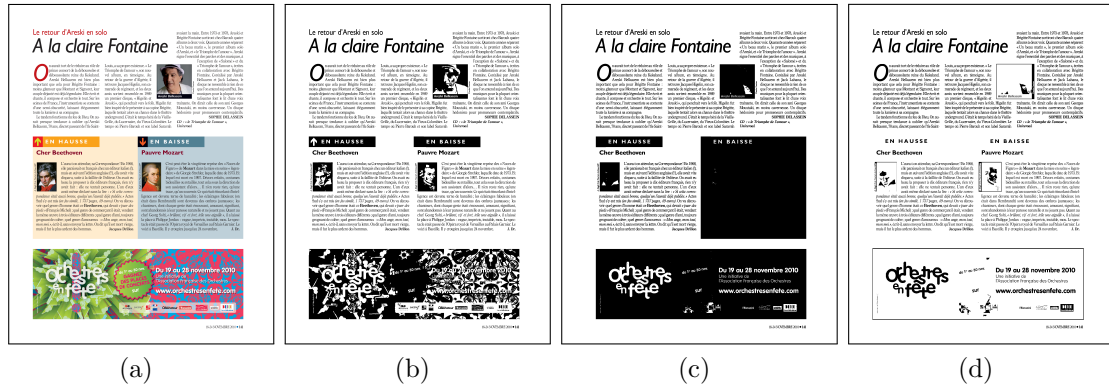


FIGURE 2.28: Évaluation visuelle de la méthode EHT : a. Document original, b. CHK et c. CGCHK, d. EHT

Chapitre 3

Reconnaissance de texte basée sur un nouveau descripteur structurel de caractères

Dans ce chapitre nous proposons un nouveau descripteur de caractères permettant une reconnaissance multi-fonte et multi-échelle. Une chaîne de reconnaissance de texte basée sur notre extraction hybride de texte et notre description structurelle de caractères est également proposée.

Sommaire

1	Introduction	88
2	Caractéristiques structurelles conventionnelles des caractères	88
3	Proposition d'un système de reconnaissance de caractères CR-UCD	89
3.1	Nouvelle description structurelle de caractères : UCD	90
3.2	Sélection du modèle	96
3.3	Mise en correspondance des caractéristiques	98
4	Expérimentation	100
4.1	Évaluation du descripteur UCD	101
4.2	Évaluation du Système CR-UCD	103
5	Conclusion	105

1 Introduction

La description d'un caractère consiste à le définir en lui attribuant une représentation unique. Cette représentation permet de le distinguer parmi un ensemble de caractères au moment de sa classification. De façon générale, la description de caractères est sensible à plusieurs facteurs tel que la variation de l'échelle et de la fonte pour des caractères d'une même classe, ou la présence de bruit dans les documents.

Dans notre travail nous proposons une description structurelle de caractères en se basant sur la description de leurs caractéristiques extraites. En plus, nous proposons la sélection du modèle adéquat de caractères (fonte et taille) et la classification finale des caractères par la mise en correspondance des descripteurs avec ceux du modèle sélectionné.

Dans la section 2, nous exposons les problèmes liés à la description des caractères. Ensuite, dans la section 3 nous présentons le descripteur proposé, la méthode de sélection du modèle de caractères et l'étape de reconnaissance. Les résultats obtenus sont présentés et discutés dans la section 4. Ainsi, nous évaluons notre descripteur UCD en termes de taux de reconnaissance de caractères.

2 Caractéristiques structurelles conventionnelles des caractères

Le choix des caractéristiques des caractères et de leurs méthodes d'extraction constitue une étape critique dans la conception d'un système de reconnaissance optique de caractères.

Les caractéristiques structurelles sont généralement extraites non pas de l'image brute du caractère mais couramment à partir du squelette ou du contour de la forme des lettres. Ces caractéristiques se divisent en deux catégories : locales et globales [121]. Les caractéristiques locales décrivent les régions locales du caractère. Les méthodes employant des caractéristiques locales décomposent les composantes connexes en arcs, jonctions et vecteurs. Les caractéristiques globales décrivent la structure globale des segments du caractère. Par exemple on peut définir le niveau de gris moyen des pixels dans le rectangle englobant le caractère. Les caractéristiques globales sont plus robustes pour la classification des caractères [122]. Ces caractéristiques devraient satisfaire plusieurs conditions [122] telles que l'invariance aux transformations (rotation, redimensionnement, échelle, fonte, etc.) et l'insensibilité au bruit.

Dans notre proposition nous sélectionnons des caractéristiques structurelles globales pour assurer l'invariance à l'échelle et à la fonte.

3 Proposition d'un système de reconnaissance de caractères CR-UCD

Nous présentons dans cette section notre proposition de description structurale multi-échelle et multi-fonte des caractères. Un exemple de lettre (*A*) reconnu avec la description proposée est donné dans la figure 3.1. Nous supposons que la lettre *A* est extraite du mot (*Académies*). Cette figure illustre une solution de reconnaissance qui inclut l'étape de description de caractères proposée. La même figure inclut aussi les étapes de sélection du modèle et de classification.

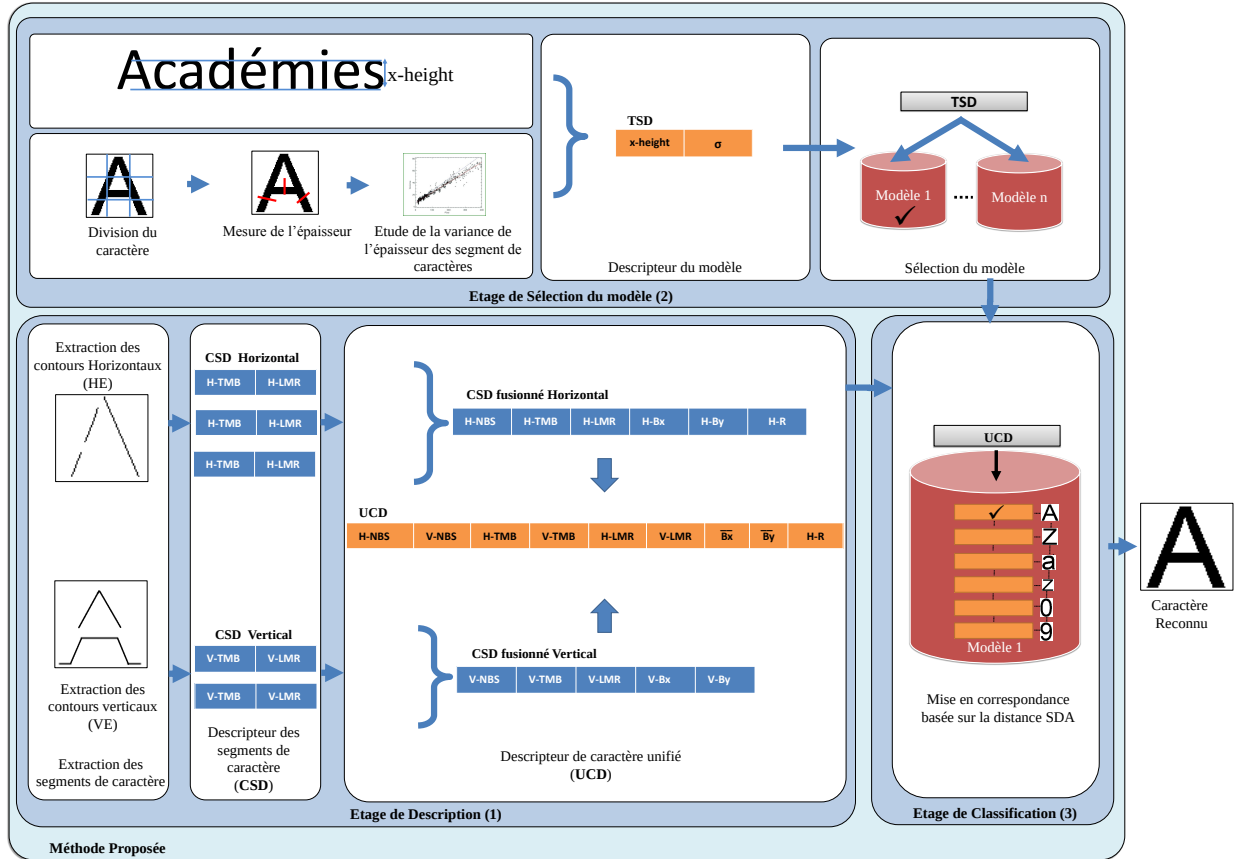


FIGURE 3.1: Reconnaissance de la lettre *A* avec CR-UCD.

Dans l'étage de description (1), nous construisons le descripteur de caractère unifié (*Unified Character Descriptor* : UCD) de chaque caractère à identifier. Ce descripteur contient les caractéristiques structurales du caractère. Ensuite, dans l'étage de sélection de modèle (2), un descripteur (*Template Selection Descriptor* : TSD) contenant la fonte et la taille du caractère à identifier est généré. C'est un descripteur qui permet de sélectionner le modèle approprié pour la classification. Dans l'étage de classification (3), au descripteur (*Unified Character Descriptor* : UCD) est associé avec le descripteur UCD le plus proche dans la base de donnée représenté par le modèle sélectionné.

Les étapes de description, de sélection de modèle et de classification sont introduites dans les trois sections suivantes.

3.1 Nouvelle description structurelle de caractères : UCD

Dans cette section, nous présentons notre stratégie d'extraction de caractéristiques structurales. Le but de cette stratégie est d'utiliser un nombre suffisant de caractéristiques qui contribuent efficacement à la discrimination des caractères.

Généralement, avant toute extraction de caractéristiques structurales, une étape de prétraitement est primordiale dans laquelle plusieurs traitements peuvent être employés tel que le seuillage, le filtrage, la détection de contour, etc [51]. Ceci permettra par exemple de nettoyer l'image du bruit ou bien de détecter les segments de caractères à partir des contours [51].

3.1.1 Calcul des segments de caractères

Nous calculons les segments de caractères qui représentent les contours droits des caractères. A travers une étape de prétraitement sur les images de caractères binaires, nous calculons les segments de caractères à travers une détection des contours droits depuis deux vues du caractère : horizontale et verticale (Figure 3.2).

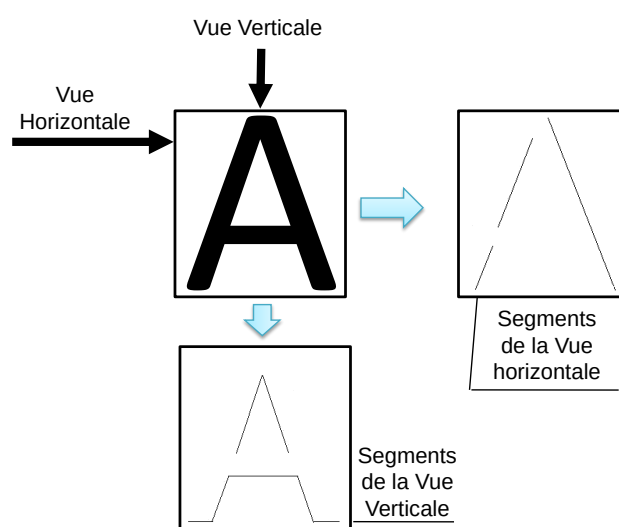


FIGURE 3.2: Les deux vues horizontale et verticale d'un caractère. Les flèches indiquent le sens de parcours des pixels.

Ainsi, dans l'extraction des contours horizontaux (*Horizontal Edge* : HE), pour chaque pixel $I(x, y)$ on calcule le contour droit selon l'équation 3.1 où $x \in [1..h]$ et $y \in [1..l]$, avec h et l sont respectivement la hauteur et la largeur du caractère. Nous notons par 0 et 1 respectivement les pixels noirs et blancs. Un pixel est comparé avec son voisin de droite. Si ce dernier est blanc alors $I(x, y)$ est marquée comme un contour HE sinon si son voisin est noir ou que $I(x, y)$ est blanc alors ce pixel n'appartient pas au contour et il est marqué comme $\overline{HE}$.

$$I(x, y) \in \begin{cases} \text{HE} & \text{Si } (I(x, y) = 0) \text{ et } (I(x, y + 1) = 1) \\ \overline{\text{HE}} & \text{Sinon} \end{cases} \quad (3.1)$$

De même, l'extraction des contours verticaux (*Vertical Edge* : VE) pour chaque pixel $I(x, y)$ est calculée comme indiqué dans l'équation 3.2. Cette fois ci, chaque pixel noir $I(x, y)$ est comparé à son voisin du bas, Si ce dernier est blanc alors $I(x, y)$ est marqué comme un contours VE sinon si son voisin est noir ou que $I(x, y)$ est blanc alors ce pixel n'appartient pas au contour et il est marqué comme $\overline{\text{VE}}$.

$$I(x, y) \in \begin{cases} \text{VE} & \text{Si } (I(x, y) = 0) \text{ et } (I(x + 1, y) = 1) \\ \overline{\text{VE}} & \text{Sinon} \end{cases} \quad (3.2)$$

L'ensemble des contours horizontaux et verticaux extraits forment des composantes connexes que nous appelons en ce qui suit segments.

Comme le montre la figure 3.3, les segments de caractères extraits sont d'abord décrits par la position des segments dans le rectangle englobant du caractère.

Les caractéristiques que nous proposons sont le nombre de segments (NBS), la position des segments (TMB, LMR), les coordonnées du barycentre (Bx, By), et la proportionnalité entre la hauteur et la largeur du caractère (Ratio). Ces caractéristiques sont expliquées en détails dans les paragraphes suivants.

3.1.2 Nombre de Segments

La description des caractères par leurs nombres de segments a été largement utilisée dans la littérature [181, 182].

Cette caractéristique augmente le degré de discrimination entre les caractères qui ne possèdent pas le même nombre de segments. Pour la même fonte de caractères le nombre de segments ne varie que rarement.

Cependant, lors de la mise à l'échelle des caractères, certaines distorsions apparaissent sur les segments à cause de l'effet de l'Aliasing [56]. Comme le montre la figure 3.3, en raison de ce phénomène, le même caractère peut avoir un nombre de segments différent sur différentes échelles. En effet, les segments sont fragmentés sur les plus grandes échelles. Pour ceci, nous proposons de vérifier la continuité des segments extraits et de les fusionner s'ils sont fragmentés.

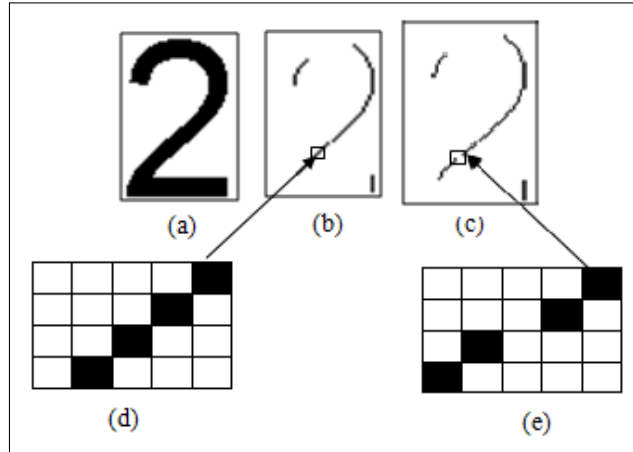


FIGURE 3.3: Dépendance entre l'échelle et la discontinuité des segments. (a) : Images en entrée ; (b) : Contour Horizontal Extraît dans une première échelle ; (c) : Contour Horizontal Extraît dans une échelle plus grande, (d) et (e) : Zoom correspondant aux segments extraits.

3.1.3 Position des Segments

Le descripteur de segments (*Character Segment Descriptor* : CSD) est composé de deux caractéristiques de position de segments dans le rectangle englobant le caractère (*Top, Middle, Bottom* : TMB) et (*Left, Middle, Right* : LMR) comme illustré dans la figure 3.1.

La première caractéristique, TMB, désigne la position d'un segment $seg_i \forall i \in [1, Nb_{seg}]$ se situant soit en haut, au milieu, ou en bas du rectangle englobant du caractère. On calcule TMB selon l'équation 3.3.

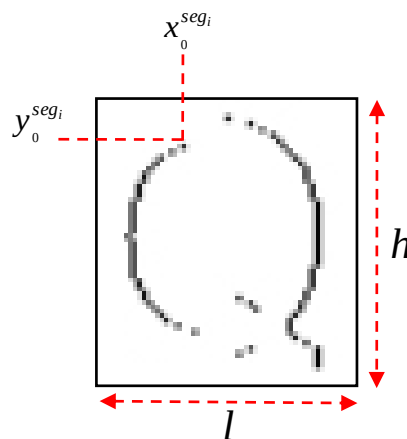


FIGURE 3.4: Position des segments dans le rectangle englobant le caractère. $x_0^{seg_i}$, $y_0^{seg_i}$ représentent les coordonnées des pixels de début x_0 , y_0 d'un segment $seg_i \forall i \in [0, Nb_{seg}]$. h et l sont respectivement la hauteur et la largeur du caractère.

$$TMB = \begin{cases} 1 & \text{Si } (y_0^{seg_i} > \frac{h}{2}) \text{ (Haut)} \\ 2 & \text{Si } (y_0^{seg_i} = \frac{h}{2}) \text{ (Milieu) , } \forall i \in [1, Nb_{seg}] \\ 3 & \text{Si } (y_0^{seg_i} < \frac{h}{2}) \text{ (Bas)} \end{cases} \quad (3.3)$$

$y_0^{seg_i}$ désigne l'ordonnée du début y_0 du segment seg_i et h est la hauteur du caractère (Figure 3.4). La deuxième caractéristique LMR désigne la position du segment qui peut se situer soit à gauche, à droite ou au milieu du rectangle englobant du caractère. LMR est calculée selon l'équation 3.4.

$$LMR = \begin{cases} 1 & \text{Si } (x_0^{seg_i} < \frac{l}{2}) \text{ (Gauche)} \\ 2 & \text{Si } (x_0^{seg_i} = \frac{l}{2}) \text{ (milieu) , } \forall i \in [1, Nb_{seg}] \\ 3 & \text{Si } (x_0^{seg_i} > \frac{l}{2}) \text{ (Droite)} \end{cases} \quad (3.4)$$

avec $x_0^{seg_i}$ représente l'abscisse du début x_0 du segment seg_i et l est la largeur du caractère (Figure 3.4).

3.1.4 Coordonnées des Barycentres

A la différence des barycentres classiques calculés à partir des moments géométriques, nous proposons une technique simple pour le calcul du barycentre des débuts de segments. En effet, nous considérons le polygone composé uniquement par les pixels des débuts de segments.

Les caractères Latins possèdent des emplacements différents de leurs barycentres, comme le montre la figure 3.5.

Nous calculons les coordonnées Bx et By du barycentre comme étant la moyenne des coordonnées $(x_0^{seg_i}, y_0^{seg_i})$ de tous les pixels des débuts de segments (Figure 3.5). Ce calcul est illustré par les équations 3.5 et 3.6.

$$Bx = \frac{\sum_{i=1}^{Nb_{seg}} (x_0^{seg_i})}{Nb_{seg}} \quad (3.5)$$

L'abscisse du barycentre Bx est égal à la moyenne des abscisses des pixels de début de segments seg_i , par rapport au nombre total de segment Nb_{seg} .

$$By = \frac{\sum_{i=1}^{Nb_{seg}} (y_0^{seg_i})}{Nb_{seg}} \quad (3.6)$$

De même, l'ordonnée du barycentre By est égal à la moyenne des ordonnées des pixels de début de segments $y_0^{seg_i}$, par rapport au nombre total de segment Nb_{seg} .

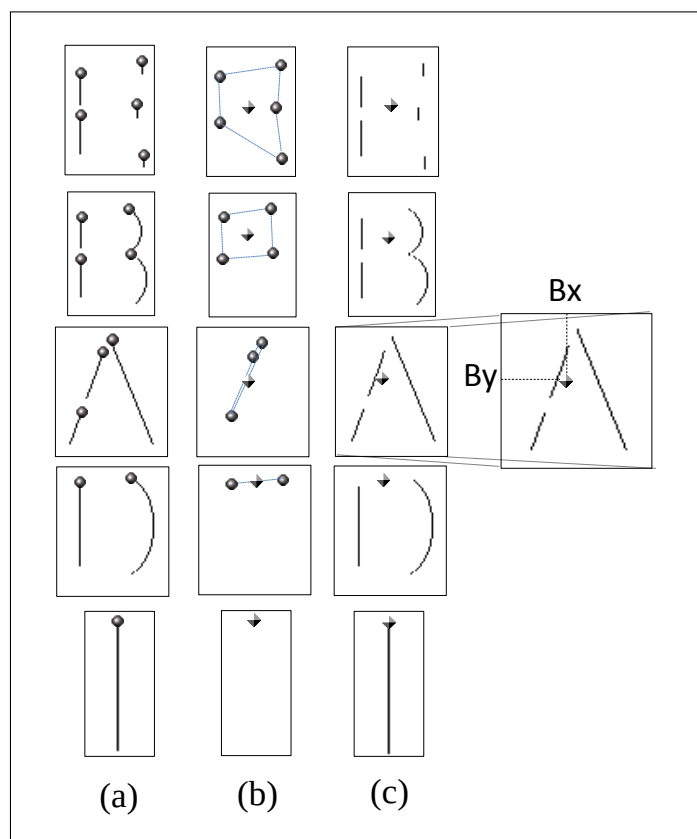


FIGURE 3.5: Calcul de Barycentres (B) pour des lettres avec un nombre de segments différent. (a) : Pixels de début de segments; (b) : Calcul du barycentre du polygone construit, (c) : Barycentre résultant.

Le calcul des coordonnées Bx et By est effectué pour les deux descripteurs de segments CSD : horizontal et vertical.

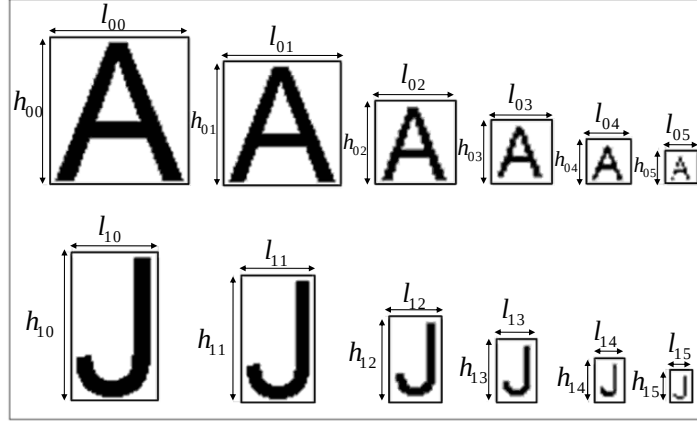
3.1.5 Proportion des dimensions

Pour améliorer la discrimination entre les caractères, nous proposons de calculer le rapport entre la hauteur et la largeur du rectangle englobant du caractère qu'on appelle (*Ratio*). Cette caractéristique est calculée selon l'équation 3.7. *Ratio* correspond à la division de la largeur l par la hauteur h .

$$Ratio = \frac{l}{h} \quad (3.7)$$

La figure 3.6 montre une représentation multi-échelle des caractères A et J .

Comme nous pouvons le voir, le *Ratio* reste constant pour le même caractère et ce pour différentes tailles. L'équation 3.8 confirme cette observation. Ainsi, le rapport entre la largeur et la hauteur reste constant pour les caractères A ($h_0 \times l_0$) et J ($h_1 \times l_1$) pour différentes


 FIGURE 3.6: Invariance du Ratio par rapport à l'échelle des lettres A et J .

échelles $[0, 1, 2, 3, 4, 5]$.

$$\begin{cases} \text{Ratio}(A) = \frac{l_{00}}{h_{00}} = \frac{l_{01}}{h_{01}} = \frac{l_{02}}{h_{02}} = \frac{l_{03}}{h_{03}} = \frac{l_{04}}{h_{04}} = \frac{l_{05}}{h_{05}} \\ \text{Ratio}(J) = \frac{l_{10}}{h_{10}} = \frac{l_{11}}{h_{11}} = \frac{l_{12}}{h_{12}} = \frac{l_{13}}{h_{13}} = \frac{l_{14}}{h_{14}} = \frac{l_{15}}{h_{15}} \end{cases} \quad (3.8)$$

Cependant, dans la même échelle, le *Ratio* permet de discriminer facilement les deux caractères A et J , comme le montre l'équation 3.9. Le rapport entre la largeur l et la hauteur h est différent entre les caractères A et J pour la même échelle.

$$\begin{cases} \text{Echelle}_0 : [\text{Ratio}(A) = \frac{l_{00}}{h_{00}}] \neq [\text{Ratio}(J) = \frac{l_{10}}{h_{10}}] \\ \text{Echelle}_1 : [\text{Ratio}(A) = \frac{l_{01}}{h_{01}}] \neq [\text{Ratio}(J) = \frac{l_{11}}{h_{11}}] \\ \vdots \\ \text{Echelle}_5 : [\text{Ratio}(A) = \frac{l_{05}}{h_{05}}] \neq [\text{Ratio}(J) = \frac{l_{15}}{h_{15}}] \end{cases} \quad (3.9)$$

En effet, le *Ratio* assure une invariance à l'échelle et augmente la discrimination entre les caractères dans la même échelle.

Basé sur ces caractéristiques de segments extraits à partir des vues horizontale et verticale, le descripteur de caractères (UCD) est calculé.

La figure 3.7 représente le descripteur UCD. C'est un vecteur unique, d'où son nom « unifié », qui regroupe les caractéristiques calculées.

Nombre de Segments		Position des Segments: Haut, Milieu, Bas		Position des Segments: Gauche, Milieu, Droit		Barycentre		Proportion des dimensions
H-NBS	V-NBS	H-TMB	V-TMB	H-LMR	V-LMR	$\overline{Bx}$	$\overline{By}$	Ratio
(a)		(b)		(c)		(d)		

FIGURE 3.7: Le descripteur unifié (*Unified Character Descriptor* (UCD)). (a) : Nombre de segments (NBS), (b). Position des segments (TMB, LMR); (c) : Barycentre ($\overline{Bx}$, $\overline{By}$); (d) : Ratio.

Dans ce descripteur on trouve le nombre de segments (H-NBS, V-NBS) et leurs positions (H-TMB, V-TMB, H-LMR, V-LMR) respectivement dans les vue horizontale et verticale du caractère. Les coordonnées du barycentre Bx et By sont calculées comme étant la moyenne des coordonnées Bx et By dans les vues horizontales et verticales (à partir des vecteurs CSD). Enfin, nous proposons le *Ratio* comme dernier élément de description.

3.2 Sélection du modèle

Même si le descripteur UCD est invariant à l'échelle, il reste plus performant sur les petits caractères. Pour une taille plus large, nous proposons de sélectionner un modèle de caractères pour améliorer l'invariance multi-échelle de notre méthode.

Un modèle définit une fonte et une taille de caractères. Les descripteurs UCD des caractères appartenant aux mêmes modèles sont réunis dans des bases de données prédéfinies. L'étape de sélection du modèle consiste à identifier la fonte et la taille du caractère à reconnaître et ainsi déterminer son modèle et la base de données qui lui est associée. Pour une fonte de caractère identifiée, il est possible de permettre une reconnaissance avec moins de calcul car on parcourt des bases de données spécifiques associées au modèle sélectionné.

Dans la suite nous expliquons le calcul de la taille et la fonte du caractère pour choisir le modèle approprié pour la classification.

3.2.1 Reconnaissance de la taille

Les lignes de texte sont composées de trois zones typographiques : la zone de l'ascendeur, la zone de la hauteur (x-height) et la zone du descendeur [60].

Comme le montre la figure 3.8, ces zones sont délimitées par quatre lignes horizontales virtuelles : la ligne ascendante, la ligne descendante, la ligne de hauteur x-height, et la ligne de base (baseline).

La zone x-height représente la hauteur des lettres minuscules. Pour l'alphabet latin la reconnaissance de la fonte est relativement facile et se base sur la hauteur x-height [62, 59].

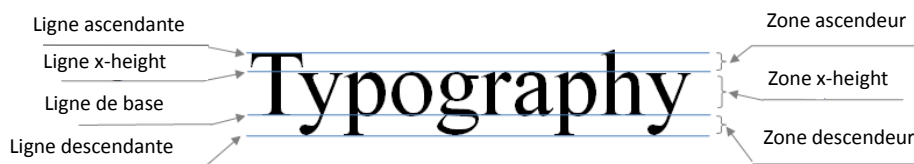


FIGURE 3.8: Typographie de l'alphabet latine.

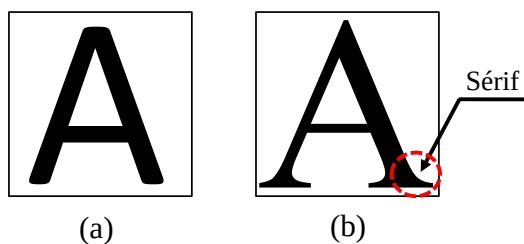
Dans notre proposition, nous identifions la taille des caractères à l'aide de la hauteur x-height calculée par l'équation 3.10 [62].

$$\text{x-height} = \text{Baseline} - \text{ligne x-height} \quad (3.10)$$

x-height correspond ainsi à la différence entre les ordonnées de la ligne de base baseline et celle de la ligne x-height.

3.2.2 Reconnaissance de la Fonte

La reconnaissance de la fonte donne des détails sur la conception typographique des caractères. Les fontes peuvent être divisées en deux catégories principales : *Serif* et *Sans Serif*. Nous entendons par *Serif* les empattements des lettres (les petits segments fins à l'extrémité des lettres (Figure 3.9)). En absence de *Serif*, les types de fonte des caractères sont connus comme *Sans Serif*. Comme exemple, la lettre *A* représentée par le type *Serif* et par le type *Sans Serif* est illustrée dans la figure 3.9.

FIGURE 3.9: La lettre *A* avec : (a) : *Sans Serif* et (b) : *Serif*.

Généralement, on suppose que les fontes en *Serif* contiennent des caractères avec une grande variation de l'épaisseur entre les différents segments [61]. Les segments dans ce cas représentent les traits bruts qui composent le caractères et non pas les contours comme nous les avons vu dans la section 3.1.

Cependant les caractères *Sans Serif* sont caractérisés par une faible variation entre les épaisseurs de leurs segments. Nous proposons d'étudier cette variation en utilisant l'écart type (σ) à partir des différents segments du caractère (Équation 3.11 et 3.12).

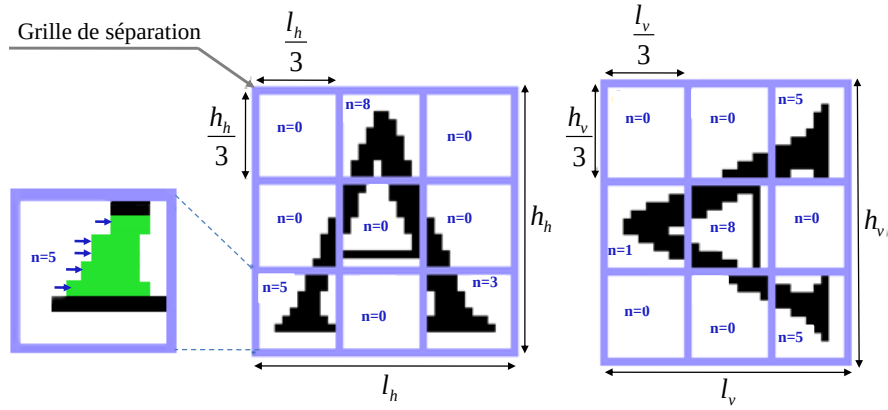


FIGURE 3.10: Reconnaissance de la fonte en se basant sur la séparation de caractères en blocs. Chaque numéro n dans un bloc désigne le nombre de segments non chevauchants avec la grille de séparation. Avec n est le nombre de segments dans un bloc, ep_{x_i} est l'épaisseur mesurée pour un segment en ordonnée x_i .

$$\sigma = \sqrt{\frac{1}{Ne-1} \sum_{i=1}^{Ne} (ep_{x_i} - \mu)^2} \quad (3.11)$$

ep_{x_i} représente l'épaisseur mesurée dans un seul bloc de caractère et μ est la moyenne de Ne épaisseurs mesurées : $i \in [1, Ne]$.

$$\mu = \frac{1}{Ne} \sum_{i=1}^{Ne} ep_{x_i} \quad (3.12)$$

Pour ce faire, nous divisons le rectangle englobant du caractère en 3×3 blocs. Comme le montre la figure 3.10, nous déterminons dans chaque bloc la médiane des épaisseurs des lignes non chevauchantes avec les bords des blocs.

Les lignes très longues ou courtes qui ne donnent pas une indication importante quant à l'épaisseur sont éliminées si elles touchent les bords des blocs.

Comme illustré dans la figure 3.11, l'écart type de l'épaisseur (σ) est à peu près constant entre les différentes tailles de la fonte mais il varie considérablement entre les différentes typographies.

3.3 Mise en correspondance des caractéristiques

La dernière étape de la solution de reconnaissance de caractère, représentée dans la figure 3.1, consiste à mettre en correspondance le descripteur UCD du caractère à identifier avec les descripteurs UCD des caractères du modèle sélectionné.

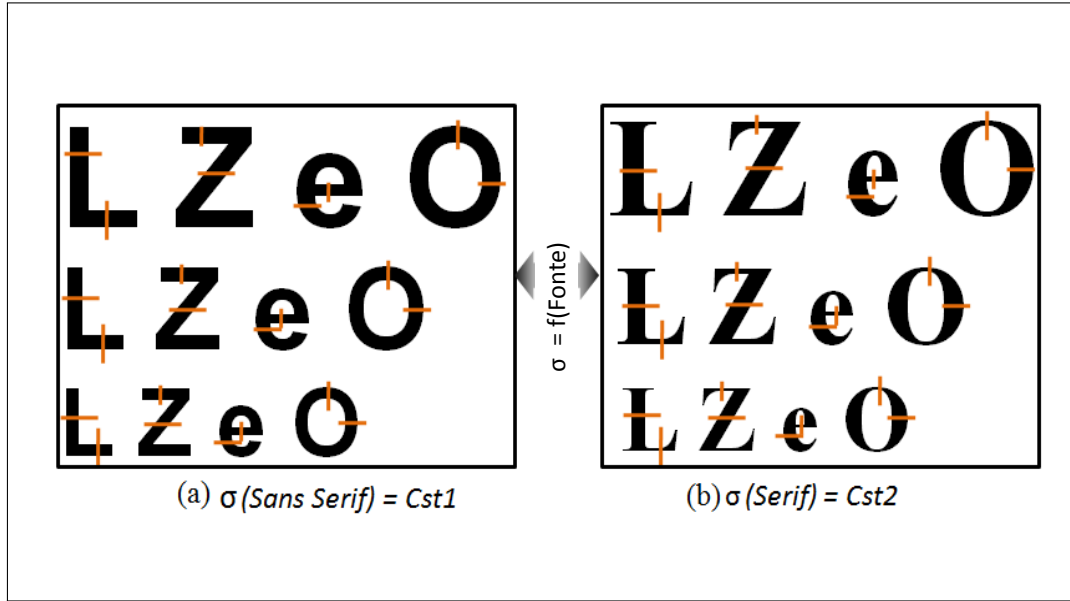


FIGURE 3.11: Écart type (σ) des épaisseurs des segments dans des caractères Multi-échelle et Multi-fonte. (a) : *Sans Serif* avec une variation modérée entre les segments fins et épais ; (b) : *Serif* avec grande variation entre les segments fins et épais.

La classification se base sur une mesure de similarité. Le degré de similarité de deux vecteurs peut être calculé en utilisant une métrique de mesure de similarité.

Dans notre proposition, on utilise la métrique de la Somme des Différences Absolues (SDA) [57]. La distance SDA est extrêmement rapide [183], d'où son intérêt dans une application temps réel. Étant donné deux descripteurs V_1 et V_2 de dimensions Dim et $i \in [1, Dim]$, la distance SDA est donnée par l'équation 3.13.

$$SDA_{V_1-V_2} = \sum_{i=1}^{Dim} |V_1(i) - V_2(i)| \quad (3.13)$$

Cette distance correspond à la somme des différences absolues des N caractéristiques dans chacun des descripteurs V_1 et V_2 .

La figure 3.12 illustre le principe de la mise en correspondance proposée.

Nous mesurons la distance SDA entre V_1 et un ensemble $V = \{V_0..V_j..V_{Nb_{desc}}\}$ du modèle sélectionné élément par élément, avec $j \in [0, Nb_{desc}]$ où Nb_{desc} représente le nombre total des descripteurs du modèle. La distance minimale SDA_{A-A} indique que le vecteur V_j est le plus similaire au vecteur V_1 .

Le descripteur UCD construit inclut les caractéristiques structurelles de nombre de segments, de leurs positions, les coordonnées du barycentre et la proportionnalité du caractère. Le descripteur du modèle TSD contient la fonte et la taille des caractères. La classification emploie la métrique de similarité rapide SDA. Dans la suite on évalue notre descripteur unifié

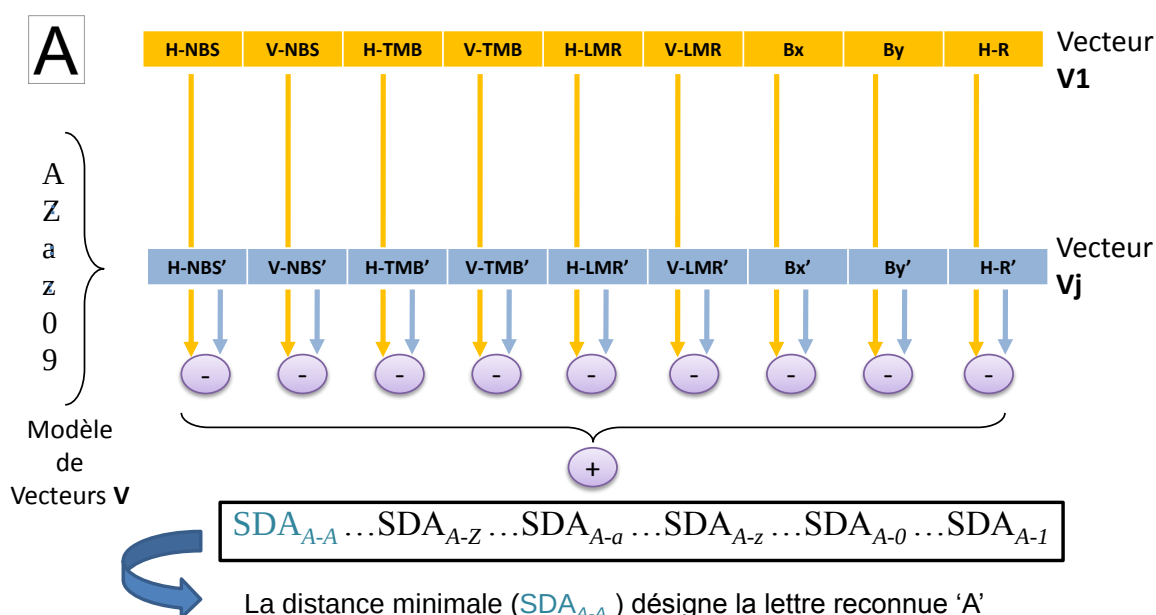


FIGURE 3.12: Principe de la mise en correspondance des caractéristiques.

de caractères UCD en termes de reconnaissance de caractères, en utilisant le descripteur du modèle TSD proposé et la classification.

4 Expérimentation

Pour évaluer la description de caractères UCD, nous proposons une mise en œuvre d'un système OCR complet qui intègre les étapes d'extraction de texte, de segmentation, de sélection de modèle et de reconnaissance (3.13).



FIGURE 3.13: Schéma fonctionnel de notre Système OCR proposé (CR-UCD).

Nous utilisons la méthode d'extraction hybride de texte EHT proposée dans le chapitre 2 pour séparer le texte de l'arrière-plan. Ensuite, le texte extrait est segmenté en caractères disjoints avec l'algorithme de l'Analyse de disposition de Page (PLA) [56]. Enfin, nous appliquons la solution de description, de sélection de modèle et de classification proposée dans

ce chapitre (en rouge).

Dans cette section, nous évaluons tout d’abord le taux de reconnaissance de caractères de notre descripteur UCD par rapport à celui de l’algorithme «i» [55] en traitant un texte avec les contraintes multi-fonte et multi-échelle.

Ensuite, nous évaluons la précision de notre système CR-UCD avec celui de Tesseract 3.02 [43] en se basant sur le taux de reconnaissance de caractère.

4.1 Évaluation du descripteur UCD

Nous étudions séparément le taux de reconnaissance du descripteur UCD en multi-fonte et multi-échelle.

Pour évaluer la précision OCR de notre descripteur UCD, nous avons généré des images comprenant un texte multi-échelle et multi-fonte alphanumérique (nombres, majuscules, minuscules, et quelque caractères spéciaux («,», «.», «;», «!», «?»)). Nous comparons notre proposition à celle du récent Algorithme «i» [55].

TABLE 3.1: Précision de reconnaissance de caractère du descripteur UCD et celui de l’algorithme «i» en utilisant des modèles multi-fonte et multi-échelle

Méthode	Taille Fonte	18	24	36	72	90
Descripteur «i»	<i>Serif</i>	90	78	90	95	98
	<i>Sans Serif</i>	80	77	82	93	95
Descripteur UCD	<i>Serif</i>	100	100	100	100	100
	<i>Sans Serif</i>	98	98	98	98	98

Le Tableau 3.1 montre une comparaison entre le descripteur UCD et celui de l’algorithme «i» en utilisant la Précision de reconnaissance de caractère (%). Pour la multi-échelle on utilise des tailles de caractères variant de 18 pt à 90 pt. La multi-fonte comprend les fontes *Serif* et *Sans Serif*. On désigne ainsi six modèles chaque modèle possède une fonte et une taille donnée (ie : modèle 1 fonte : *Serif* et taille : 18).

Dans cette évaluation, chaque caractère d’un modèle est comparé avec les caractères du même modèle. Ainsi on évalue la robustesse de notre description seulement pour la discrimination des caractères d’une même fonte et une même taille.

Les résultats obtenus démontrent que notre méthode surpasse l’algorithme «i» atteignant une moyenne de 99% de précision de reconnaissance sur tous les modèles évalués.

Le problème rencontré par notre description est la non-reconnaissance de caractères similaires comme « I » et « l » dans les différentes échelles de la fonte *Sans Serif*.

En effet ces caractères possèdent exactement la même morphologie à cause de l'absence des empattements (*Serif*) dans leurs terminaisons. Donc il est impossible de discriminer les lettres « I » et « l » par la description. Ceci diminue le taux de reconnaissance dans un système OCR.

Comme perspectives, les méthodes lexicales peuvent être utilisées dans ce cas, pour améliorer la performance de la description [185]. En effet, la reconnaissance des caractères ambigus (« I », « l ») peut être réalisé en identifiant la globalité du mot dans lequel ils sont inclus. Ainsi on compare un descripteur global du mot à un ensemble de descripteurs de mots prédéfinis dans une base de données qui s'appelle dictionnaire. Une fois le mot identifié, les caractères ambigus sont reconnus.

Dans le tableau 3.2 nous illustrons la performance du descripteur UCD et celui de l'algorithme « i » en utilisant un seul modèle de référence défini par la fonte *Sans Serif* et la taille 90 pt. La classification des caractères est effectuée en comparant les descripteurs des caractères à identifier uniquement avec les descripteurs du modèle de référence. Ceci permettra d'évaluer la robustesse face à la multi-fonte et à la multi-échelle.

TABLE 3.2: Précision OCR des descripteurs UCD et celui de l'algorithme « i » en utilisant un seul modèle de référence

Méthode	Taille		18	24	36	72	9
	Fonte						
Descripteur «i»	Serif		12	20	25	59	98
	Sans Serif		15	14	20	50	95
Descripteur UCD	Serif		31	50	52	91	100
	Serif		25	30	44	70	98

Nous pouvons voir que dans ce cas le descripteur UCD surpasse encore le descripteur de l'algorithme « i » dans les fontes *Serif* et *Sans Serif*.

Malgré que le descripteur UCD donne une haute précision sur une seule fonte et une seule taille (première évaluation), la précision de reconnaissance de caractère diminue lorsqu'on traite des caractères multi-fonte et multi-échelle.

On remarque que l'erreur de reconnaissance est importante entre les fontes et entre les échelles. De plus, cette erreur est plus grande lorsque la différence de l'échelle des modèles

est plus large. Par conséquent, la sélection du modèle dans le système CR-UCD demeure nécessaire pour comparer les caractères avec les modèles les plus proches.

4.2 Évaluation du Système CR-UCD

Nous évaluons notre proposition CR-UCD sur la base LRDE-DBD [69] composée de 125 documents de magazine.

Comme indiqué dans la figure 3.14, les documents de la base LRDE-DBD incluent différentes tailles de texte, et de fontes *Serif* et *Sans Serif*.



FIGURE 3.14: Exemples de documents de la base LRDE-DBD.

Nous illustrons dans la figure 3.15 les modèles utilisés dans la mise en correspondance des caractéristiques.

Nous générons six modèles répartis en deux classes : *Serif* et *Sans Serif*.

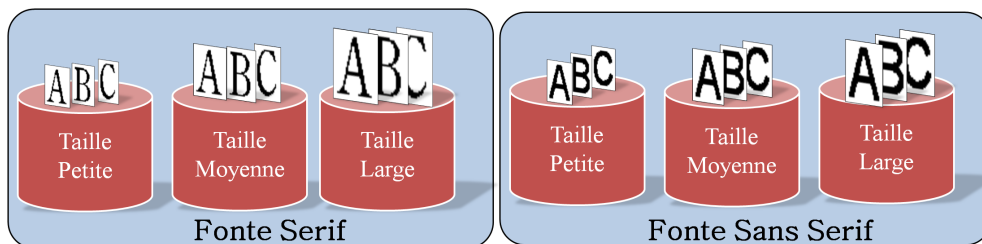


FIGURE 3.15: Les templates multi-échelle et multi-fonte utilisées.

Chaque répartition, inclut trois sous-modèles qui consistent en trois plages de tailles de caractères : petites, moyennes, et larges. Ce choix de répartition regroupe les plus proches en tailles et en fontes. Ainsi on réduit l'écart entre les échelles et les fontes dans chaque modèle.

Les petits caractères désignent le texte dans les paragraphes, les caractères moyens désignent les sous titres et les caractères larges désignent les titres.

TABLE 3.3: Les seuils x-height utilisés

Tailles de caractères	Seuils x-height (pixels)
Petite	0-30
Moyenne	30-55
Large	55-100

Pour distinguer ces tailles, les seuils, donnés dans le Tableau 3.3 sont employés.

Dans la suite, nous comparons notre système CR-UCD à celui de Tesseract [43].

Pour faire une comparaison équitable, nous avons désactivé le dictionnaire Tesseract que nous ne traitons pas encore dans notre travail.

La figure 3.16 montre le cumul des taux de reconnaissance pour les systèmes Tesseract et CR-UCD sur les 125 documents de la base LRDE. Le système de reconnaissance de caractères CR-UCD affiche un taux de reconnaissance de 60%, 1.5x meilleur que celui de Tesseract sur 125 documents de la base LRDE-DBD.

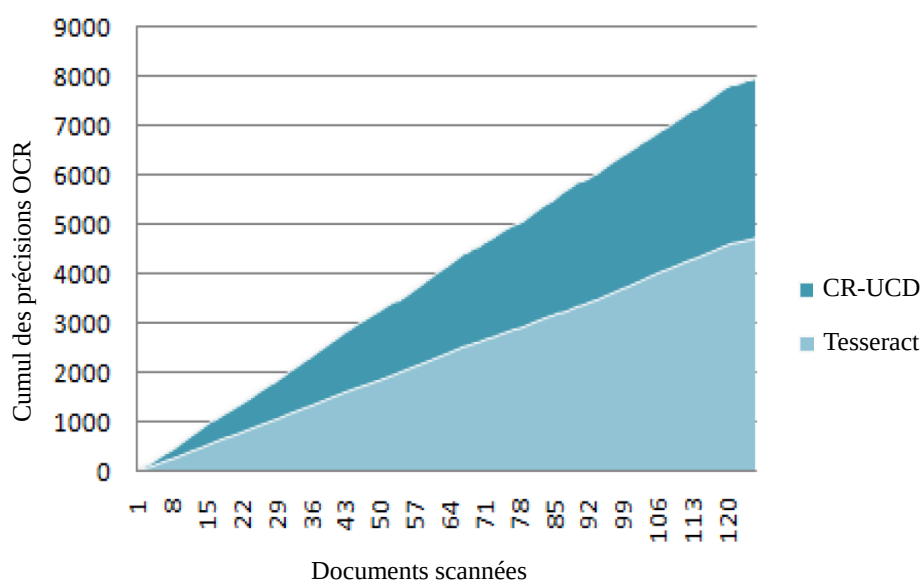


FIGURE 3.16: Cumul des taux de reconnaissance des systèmes Tesseract et CR-UCD sur 125 documents de la base LRDE-DBD .

5 Conclusion

Nous avons proposé dans ce chapitre un nouveau descripteur de caractères unifié (UCD). Les caractéristiques structurelles globales des caractères ont été combinées comme le nombre de segments, la position des segments, les barycentres, et la proportion de la largeur et de la hauteur. Le descripteur a été évalué au sein d'une chaîne de reconnaissance de caractères que nous avons proposée et développée (CR-UCD). Cette chaîne inclut les étapes d'extraction hybride de texte, de segmentation du texte en caractères disjoints, de sélection du modèle approprié et de classification rapide. Grâce à cette chaîne, nous avons démontré que notre descripteur est robuste et invariant aux changements d'échelle et de fonte. Cette chaîne affiche une précision de reconnaissance de caractère 1.5x meilleure que Tesseract en désactivant le dictionnaire de ce dernier pour fournir une comparaison équitable.

Chapitre 4

Parallélisation sur GPU de l'extraction hybride du texte dans les documents hétérogènes

Dans le cadre d'une application temps réel de dématérialisation, nous proposons dans ce chapitre des stratégies de parallélisation pour l'accélération de l'extraction hybride de texte (EHT), proposée dans le chapitre 2, sur les plateformes multi-cœur (CPU-GPU). Les algorithmes parallèles développés permettent une accélération de l'ordre de 6.74x par rapport aux algorithmes séquentiels.

Sommaire

1	Introduction	109
2	Architectures Multi-cœur	110
2.1	Les CPU multi-cœur	111
2.2	Les processeurs graphiques GPU	112
3	Parallélisation de la méthode de l'extraction hybride de texte	113
3.1	Analyse temporelle de la méthode EHT	113
3.2	Stratégie de parallélisation de la méthode EHT	114
3.3	Accélération de la correction gamma (CG)	115
3.4	Accélération de la classification hybride basée sur le noyau K-means	127
3.5	Accélération de l'analyse de structure du document (Séparation et localisation du texte)	136
4	Évaluation de la stratégie de parallélisation de la méthode EHT	137
4.1	Outils de développement et environnement de travail	138
4.2	Évaluation de la parallélisation de la correction gamma	138
4.3	Évaluation de la parallélisation de la méthode CHK	138
4.4	Évaluation de la parallélisation de la méthode d'analyse de structure de documents	143

4.5	Évaluation de l'accélération proposée de la méthode EHT	143
5	Conclusion	147

1 Introduction

Nous considérons l'application de dématérialisation qui transforme un ensemble de documents papiers en documents numériques éditables en utilisant un scanner multifonction de type Sharp MX-C250. Pour un document donné, la reconnaissance de caractères est réalisée sur un poste de travail distant du scanner au moment de la numérisation du document successeur. Ainsi, la reconnaissance doit s'exécuter durant le processus de numérisation (T_{num}) qui est égal à 3.15 secondes pour un document A4 et de résolution 300 dpi (figure 4.1) qui représente la contrainte temps réel de notre application de dématérialisation.

Dans ce contexte, la chaîne de reconnaissance de caractères développée (CR-UCD) nécessite 8.75 secondes comme temps de traitement d'un document de format A4 et de résolution 300 dpi ce qui ne satisfait pas la contrainte temps réel comme le montre la figure 4.1). Nous proposons d'accélérer les algorithmes les plus coûteux en temps de calculs de la chaîne CR-UCD.

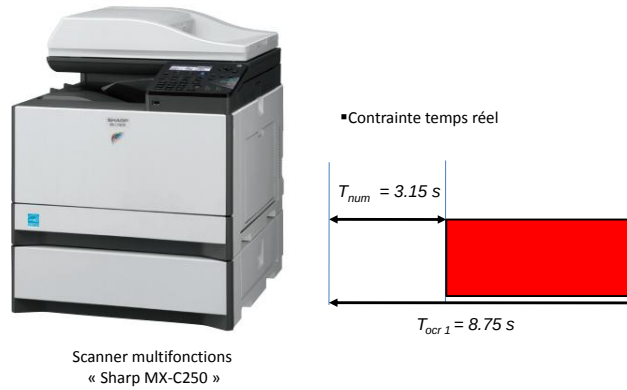


FIGURE 4.1: Contrainte temps réel de l'application de dématérialisation. T_{num} : Temps de numérisation du scanner, T_{ocr1} : Temps de reconnaissance de caractères de la méthode CR-UCD avant accélération de l'application.

Nous comparons dans le tableau 4.1 les différentes étapes de la chaîne de reconnaissance de caractères CR-UCD en termes de temps d'exécution. La mesure du temps d'exécution est effectuée sur un CPU Intel i3 d'architecture *Nehalem* de fréquence 2.5 GHz et de mémoire RAM de 4 Go.

La description et la reconnaissance de caractères possèdent des temps d'exécutions relativement faibles par rapport au temps total de la chaîne de reconnaissance de caractères CR-UCD. Par contre, la méthode d'extraction hybride de caractères (EHT) possède le temps d'exécution le plus important (8.16 secondes) qui représente 93.25% du temps total de la chaîne.

Nous accélérons la méthode d'extraction hybride de texte (EHT) car elle est la plus complexe et elle représente 93.25% du temps total de la chaîne de reconnaissance de caractères CR-UCD. De plus cette méthode est fortement parallélisable, comme nous l'avons montré

TABLE 4.1: Analyse de la méthode CR-UCD sur CPU

Étapes de la méthode CR-UCD	Temps (s)	Temps (%)
Extraction hybride de caractères	8.16	93.25
Description de caractères	0.36	4.11
Reconnaissance de caractères	0.23	2.62
Total	8.75	100

dans le chapitre 2.

Dans notre travail, nous utilisons les architectures parallèles tel que les CPU multi-cœur et les GPU. Les CPU multi-cœur sont efficaces pour les traitements qui ne sont pas exécutés systématiquement, c'est-à-dire dont l'exécution est conditionnée par le succès d'un test. Spécialement, les GPU se montrent très adaptées aux algorithmes de traitement d'images vus leur potentiel à paralléliser les tâches de calcul intensives sur les pixels des images.

Les CPU et les GPU peuvent s'utiliser conjointement pour augmenter la performance d'un algorithme lorsque celui-ci est composé à la fois d'étapes de calcul intensif et de traitement conditionnel. La stratégie de parallélisation convenable dépend de plusieurs éléments tel que les dépendances de données, les contraintes de ressources matérielles et la complexité des algorithmes.

Dans ce chapitre nous présentons notre stratégie d'accélération de la chaîne de reconnaissance de caractères. Nous introduisons dans la section 2 les architectures multi-cœur des CPU et GPU. Ensuite, nous exposons dans la section 3 la stratégie de parallélisation de la méthode d'extraction hybride de texte sur GPU-CPU. Enfin, nous évaluons les résultats obtenus en étudiant les expérimentations réalisées concernant les différentes parties de la méthode d'extraction de texte dans la section 4.

2 Architectures Multi-cœur

Les processeurs multi-cœur sont très variés et difficiles à comparer. Les CPU-multi-cœur et les GPU s'avèrent des architectures performantes [123]. Ils sont efficaces surtout pour différents types d'algorithmes de traitement d'images [124, 125]. Dans la suite nous présentons les architectures génériques des CPU et des GPU.

2.1 Les CPU multi-cœur

L'architecture des CPU multi-cœur est composée d'unités d'exécution complexes pour l'exécution de programmes génériques. Ce type de programmes emploie un grand nombre d'instructions conditionnelles lui permettant d'exécuter des flux de traitements non homogènes et de les répartir entre ses unités de calcul.

Le CPU est adapté à la mise en œuvre efficace de fils d'exécution (thread) complexes et non homogènes selon un paradigme de parallélisme orienté tâches [126].

Les architectures CPU homogènes se composent de quelques cœur de traitement identiques, une mémoire partagée et une hiérarchie de caches.

Paradigme de développement logiciel sur CPU : Il existe plusieurs bibliothèques qui permettent la programmation des threads dans un CPU. On trouve principalement, la librairie PThread de bas niveau adopté par la plupart des systèmes d'exploitation. Son désavantage c'est la nécessité d'un nombre considérable de lignes de code. La librairie OpenMP est une alternative plus simple pour le programmeur, qui permet plus de productivité car elle est de plus haut niveau [187].

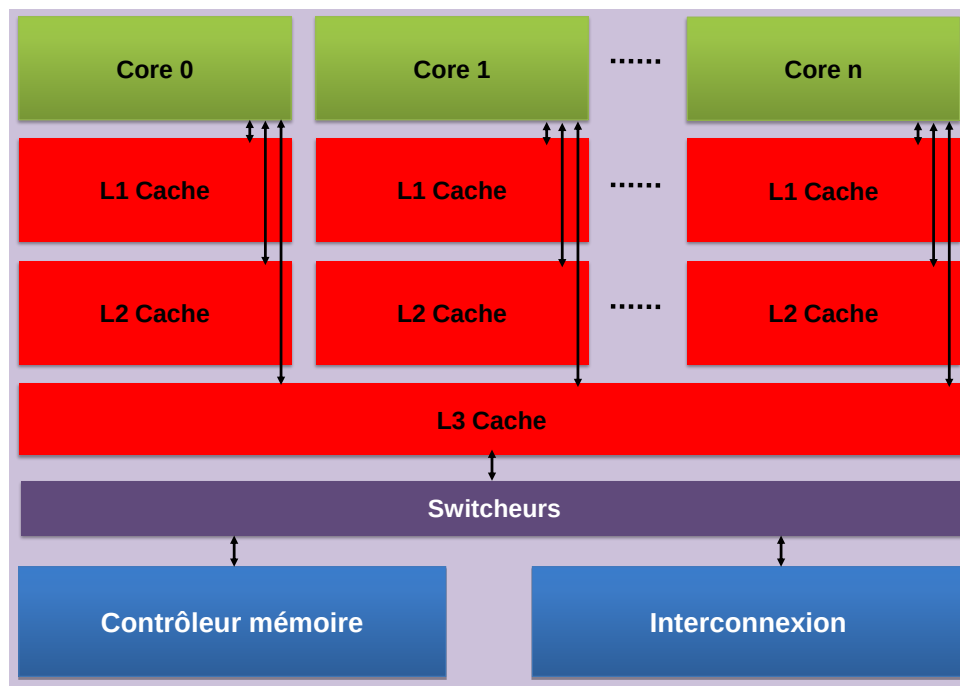


FIGURE 4.2: Architecture interne d'un CPU.

Les GPU ont évolué au point où de nombreuses applications du monde réel sont facilement mises en œuvre et fonctionnent significativement plus rapidement que sur les CPU multi-cœur. D'après Nvidia, « Les architectures informatiques futures seront des systèmes hybrides avec les GPU parallèles travaillant en collaboration avec les CPU multi-cœur ». [129]

2.2 Les processeurs graphiques GPU

L'architecture massivement parallèle des GPU est composée de multiples unités de calcul et d'un ordonnanceur adapté pour effectuer des traitements de type SIMD sur un grand nombre de données. Cette structure est très bien adaptée au parallélisme orienté données, c'est-à-dire à un parallélisme très homogène, dans lequel des traitements simples sont appliqués à une taille importante de données. Ce type de parallélisme est prédominant en traitement d'images, où les données sont de grande taille et où, pour chaque pixel, les traitements à effectuer sont similaires [188]. Notons que pour pouvoir alimenter cette multitude de processeurs en données, les GPU possèdent une bande passante vers leurs mémoires bien plus importante que les CPU. Actuellement un GPU d'architecture Pascal atteint une bande passante de l'ordre de 1 TB/s vers sa mémoire [189] alors qu'un CPU de type Intel KNL possède 500 Gb/s et un CPU de type Intel Haswell ne dispose que de 100 Gb/s

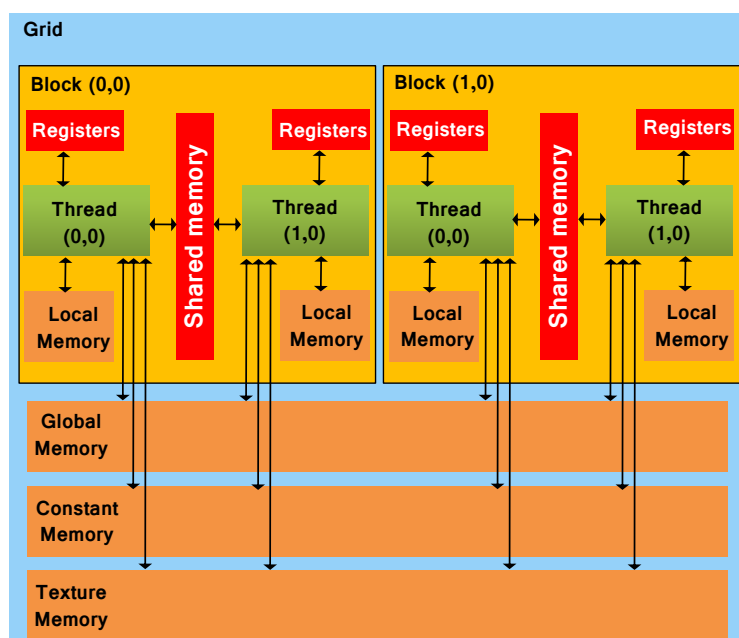


FIGURE 4.3: Architecture interne d'un GPU.

Organisation des unités de calcul : Les GPU comprennent des multiprocesseurs massivement parallèles composés de plusieurs processeurs ou unités de calcul. Ces unités sont capables d'effectuer simultanément la même opération sur des données multiples (SIMD). Le GPU contient aussi, des ordonnanceurs de threads, et des unités de contrôle pour répartir les tâches sur les unités de calcul.

Organisation de la mémoire : Les GPU incluent des mémoires globales, partagées (Shared), de texture, de constantes, et des registres [130]. La mémoire principale pour les GPU est la mémoire globale. Elle est accessible par tous les processeurs et elle peut contenir 1

GB ou plus. Par exemple pour les GPUs Nvidia GTX 660 et Quadro K620 la taille de la mémoire globale est de 2 GB. L'accès à cette mémoire est relativement lent, sa latence étant de l'ordre de 600 ns. Cependant, cette mémoire contient souvent deux niveaux de caches L1 et L2, ce qui permet l'accélération de son accès. En effet, on enregistre temporairement dans les mémoires caches les données en provenance de la mémoire RAM et, puisque les mémoires caches sont plus rapides et plus proches des unités de calcul, on diminue ainsi le temps d'accès à ces données [194]. La mémoire de constantes est rapidement accessible par tous les processeurs du GPU, sa latence n'est pas mentionnée dans la littérature car elle est négligeable par rapport à la latence de la mémoire globale. Le désavantage des mémoires de constantes et de textures est qu'elles sont accessibles seulement en lecture. Les mémoires locales sont des mémoires accessibles seulement par les threads d'un même multiprocesseur. Elles sont utilisées par le compilateur dans le cas où les registres ne suffisent plus pour contenir plus de données.

Paradigme de développement logiciel sur GPU : Pour exploiter les plateformes GPU, il est possible d'utiliser plusieurs technologies. Les plus connues sont Open Computing Language (OpenCL) [16] et Compute Unified Device Architecture (CUDA) [129]. La technologie OpenCL est un standard libre, portable sur différents types d'architectures (CPU, GPU, FPGA). D'autre part, la technologie CUDA permet une programmation en C/C++ haut niveau. Comme le montre la figure 4.3, dans CUDA les threads sont regroupés selon des blocks. Le conteneur de tous les blocks est appelé Grid.

Notre stratégie de parallélisation exploite les processeurs graphiques GPU tout en adoptant conjointement les paradigmes de développement CUDA et OpenCL. Elle est détaillée dans la suite.

3 Parallélisation de la méthode de l'extraction hybride de texte

Dans cette section, nous analysons le temps d'exécution de la méthode EHT. Ensuite, nous présentons la stratégie de parallélisation de ces étapes.

3.1 Analyse temporelle de la méthode EHT

Nous rappelons les différentes étapes qui constituent notre méthode d'extraction hybride de texte dans les documents hétérogènes dans le tableau 4.2.

Nous avons mesuré les temps d'exécution de ces traitements sur un processeur CPU Intel i3 de fréquence 2.5 GHz et de mémoire RAM de 4 Go. Ces mesures ont été réalisées pour un ensemble de documents hétérogènes issus de la base LRDE-DBD contenant à la fois des images et du texte [103]. Les proportions des temps d'exécution des différentes étapes sont représentées également dans le tableau 4.2.

Nous remarquons que le temps d'exécution le plus important concerne (b) CGCHK (appliqué sur les régions d'images) affichant 4.27 secondes soit 52.32 % du temps d'exécution total de la méthode EHT. La complexité de cet algorithme est la plus importante essentiellement

TABLE 4.2: Analyse de la méthode EHT sur CPU

Étapes de la méthode EHT	Temps (s)	Temps (%)	Complexité
(a) Analyse de structure du document (Séparation et localisation du texte)	2.5	30.63	+++
(b) CGCHK	4.27	52.32	++++
(c) CHK	1.372	16.81	++
(d) Correction du rendu couleur	0.028	0.34	+
Total	8.16	100	

à cause de la complexité des algorithmes de génération des matrices de co-occurrences et de calcul des caractéristiques de Haralick. De même, les étapes (c) CHK (appliqué sur les régions de texte) et (a) l'analyse de la structure du document sont coûteux en temps de calcul affichant respectivement 16.81% et 30.63% du temps d'exécution de la chaîne. Leurs complexités sont considérables vu (a) l'emploi intensif des opérations morphologiques et (c) le calcul itératif. (d) L'étape de correction du rendu couleur est la plus rapide en affichant un temps d'exécution de 0.028 secondes représentant 0.34% par rapport au temps total de la méthode EHT.

L'algorithme CGCHK réalise dans un premier temps la correction gamma (CG), puis exécute l'algorithme CHK pour binariser l'image améliorée. Dans la suite nous parallélisons sur GPU (dans l'ordre) les algorithmes (b) CG, (c) CHK et de (a) l'analyse de structure de document. Nous gardons l'exécution de l'étape (d) sur le CPU car son temps d'exécution est faible.

3.2 Stratégie de parallélisation de la méthode EHT

Nous avons proposé une stratégie de parallélisation hybride CPU-GPU illustrée par la figure 4.4. Notre stratégie traite chaque étape de la méthode d'extraction hybride de texte (EHT) sur CPU, sur GPU ou bien en hybride sur les deux architectures. Le choix de la cible d'implantation dépend de la nature des algorithmes et de l'architecture matérielle. Les méthodes parallélisées sur CPU-GPU sont celles qui possèdent des parties dans l'algorithme fortement parallélisables et d'autre parties présentant des dépendances de données qui empêchent leurs parallélisation. De plus, l'intensité de calcul sur le GPU est faible dans ces méthodes de sorte qu'elle ne masque pas la quantité de transfert mémoire ce qui ralentit l'ensemble du traitement. Parmi ces méthodes on trouve l'analyse de structure du document qui se base sur une parallélisation des opérateurs morphologiques employés dans les méthodes de séparation texte/image et dans la localisation de caractères. De plus, le calcul du seuil Otsu est parallélisé de façon hybride CPU-GPU.

Les méthode parallélisées entièrement sur GPU sont les méthodes dont les algorithmes sont

caractérisés par des indépendances de traitement qui les rend totalement parallélisables. Aussi, l'intensité de calcul de ce type d'algorithmes est haute ce qui lui permet de masquer le temps de transfert mémoire. C'est le cas pour les algorithmes de calcul des caractéristiques de Haralick et de la génération des matrices de co-occurrences.

Finalement les méthodes traitées sur le CPU sont les méthodes qui présentent beaucoup de branchements de traitement tel que l'algorithme de l'estimation de la valeur gamma optimale ou bien les algorithmes qui sont extrêmement rapide sur CPU et qui ne nécessitent pas de parallélisation comme la correction du rendu couleur. Nous avons parallélisé également le traitement du calcul du seuil Otsu et le calcul des caractéristiques de Haralick et des matrices de co-occurrences sur deux GPU. Ceux-ci sont commandés par deux threads du CPU.

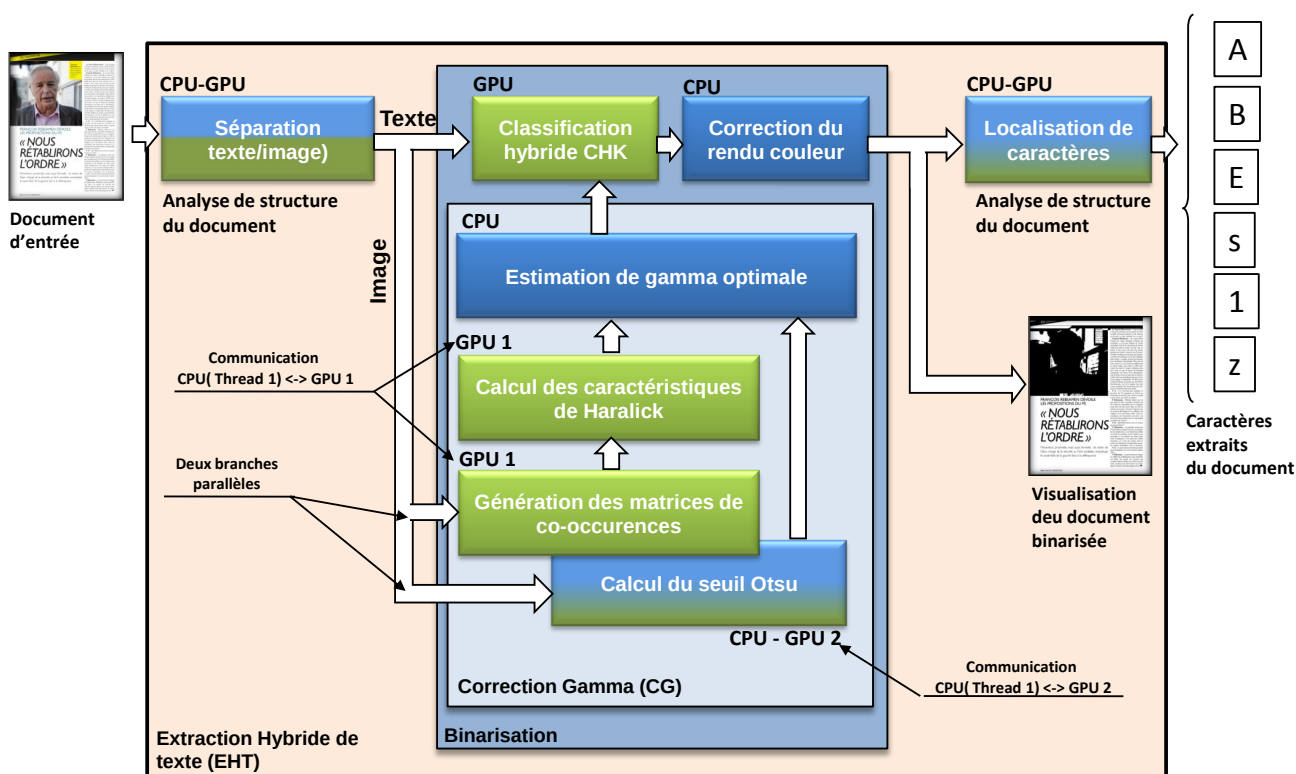


FIGURE 4.4: Stratégie de parallélisation de la méthode EHT.

3.3 Accélération de la correction gamma (CG)

Avant l'extraction du texte, les régions d'images sont améliorées en utilisant la correction gamma. Comme l'illustre la figure 4.5, ce traitement comprend différents algorithmes permettant de calculer le seuil Otsu, les matrices de co-occurrences, l'énergie, le contraste et

d'estimer les valeurs gamma optimales.

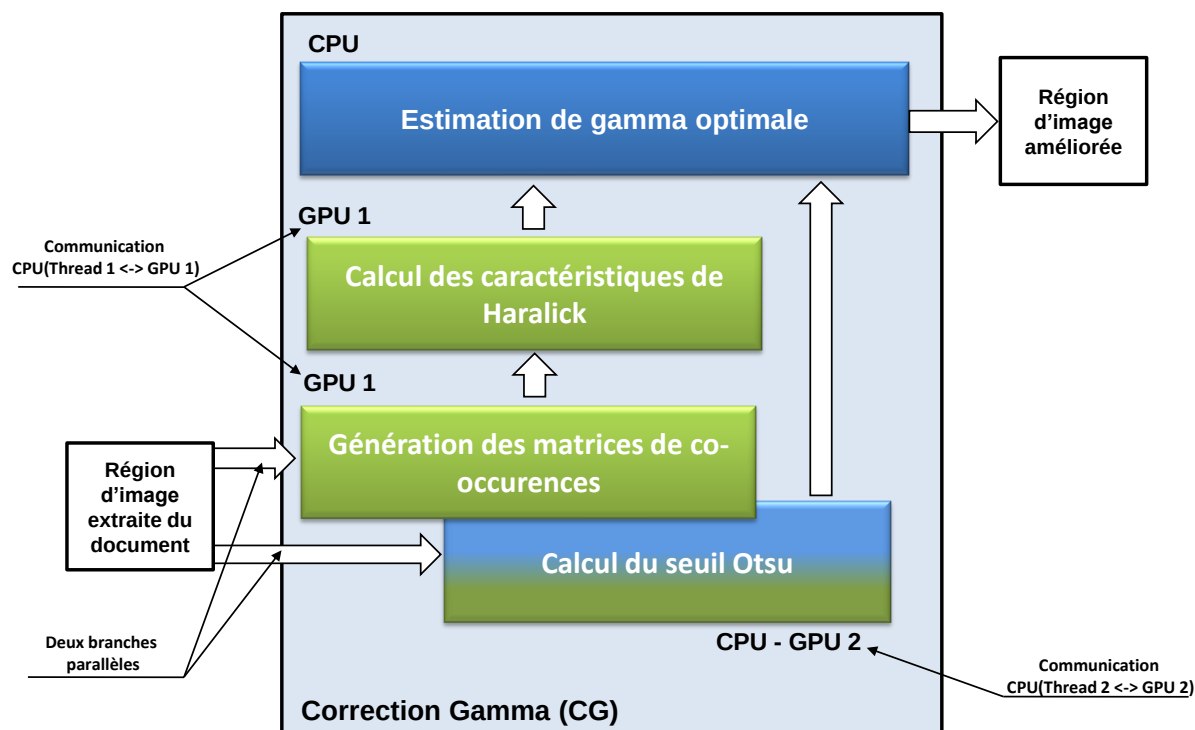


FIGURE 4.5: Stratégie de parallélisation de la méthode de correction gamma (CG).

Nous analysons le temps d'exécution de ces algorithmes sur CPU Intel i3 en se basant sur le tableau 4.3.

On remarque que les algorithmes les plus coûteuses en temps d'exécution sont : le calcul du seuil Otsu, l'extraction des matrices de co-occurrence, et la détermination de l'énergie et du contraste. On se propose dans la suite de paralléliser ces algorithmes sur le GPU. Quant à l'estimation de la valeur gamma, elle sera exécutée sur le CPU vu qu'elle inclut principalement des traitements conditionnels adéquats pour le CPU (voir chapitre 2). Nous expliquons dans la suite la parallélisation effectuée sur le GPU.

3.3.1 Calcul parallèle du seuil Otsu

Le calcul du seuil Otsu s'effectue en deux phases : La génération de l'histogramme de l'image, puis, le calcul du seuil à partir des variances entre les classes d'où le nom d'inter-classes. La génération de l'histogramme de l'image est l'élément le plus complexe dans l'algorithme. En effet, pour une image I la complexité de la génération de l'histogramme, qui est de l'ordre de $\mathcal{O}(Nb'_{Images} \times Nb_{pixels})$, est largement supérieure à celle du calcul du seuil qui est de

TABLE 4.3: Étude temporelle des différentes algorithmes de la méthode gamma sur CPU

Méthode	Temps (s)	Temps (%)
Seuil Otsu	1.5	35.12
Matrices de co-occurences	1.37	32.08
Énergie et contraste	1.2	28.10
Estimation de Gamma	0.2	4.68
Total	4.27	100

l'ordre de $\mathcal{O}(Nb'_{Images} \times N_g)$ avec Nb_{pixels} et N_g sont respectivement le nombre de pixels et le nombre de niveaux de gris dans l'image (voir chapitre 2) et Nb'_{Images} est le nombre d'image après l'optimisation algorithmique de la correction gamma. La haute complexité de la génération de l'histogramme est adéquate pour le traitement sur GPU car elle engendre une très haute intensité de calcul par rapport à la quantité de transfert mémoire CPU-GPU. Cependant la faible complexité du calcul du seuil entraîne une faible intensité de calcul par rapport à la quantité de transfert mémoire. C'est pour cette raison que nous proposons une stratégie de parallélisation hybride (CPU-GPU) pour le calcul du seuil Otsu. Nous parallélisons la génération de l'histogramme qui n'inclut pas de branchements sur le GPU. La partie la moins complexe de recherche du seuil est réalisée sur le CPU.

Le calcul de l'histogramme de niveau de gris d'une image consiste à calculer la proportion de pixels pour chaque niveau. Pour les pixels de mêmes valeurs, on incrémente un compteur dans la même case mémoire. La parallélisation des histogrammes engendre souvent un problème de collision mémoire [190]. Une collision surgit lorsque nous avons plusieurs écritures dans le même emplacement mémoire. La figure 4.6 illustre un exemple de calcul d'histogramme où on observe une collision lors de l'écriture, par l'incrémement «+1», dans les emplacements mémoires d'index «23» et «123». Il est donc nécessaire de bien gérer l'accumulation des valeurs dans la hiérarchie mémoire.

Il est possible d'utiliser des opérations dites atomiques sur le GPU. Ces opérations garantissent un accès exclusif à chaque thread sans toutefois qu'un ordre soit défini entre les threads pour l'écriture à l'adresse en question avant que l'opération en cours ne soit finie.

Actuellement, les histogrammes sont beaucoup plus faciles à manipuler sur des architectures GPU grâce à l'amélioration de la performance des opérations atomiques dans l'architecture Kepler [112] et le support natif des opérations atomiques dans la mémoire partagée de l'architecture Maxwell [191].

La figure 4.7 illustre notre configuration du GPU et la stratégie générale de l'implantation

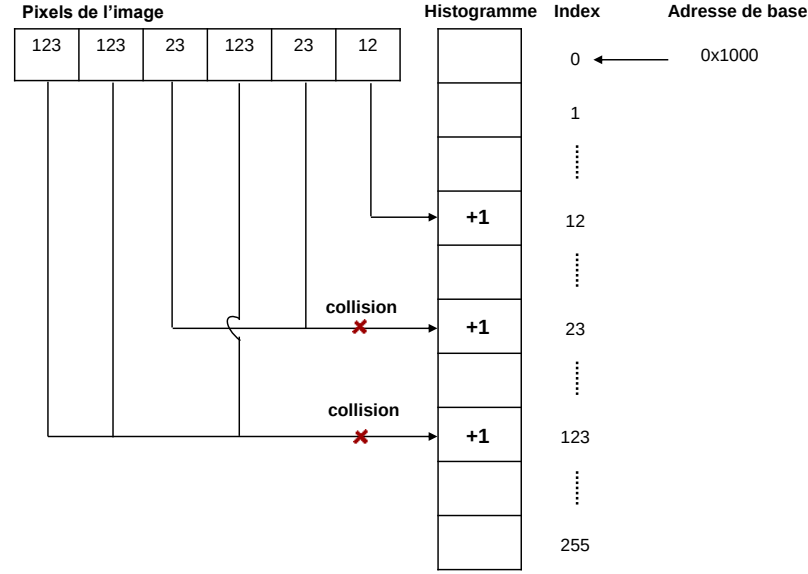


FIGURE 4.6: Problème de la collision mémoire liée à la parallélisation du calcul de l'histogramme.

employée. Pour l'implantation de l'histogramme, nous configurons le GPU en blocks de tailles $Nb_{ThreadBlock}$ égales à 16×16 threads représentant un pixel par thread. Le nombre de blocks total dans l'image (Nb_{Blocks}) est calculé selon la formule de l'équation 4.1.

$$Nb_{Blocks} = \frac{((L \times H) + Nb_{ThreadBlock} - 1)}{Nb_{ThreadBlock}} \quad (4.1)$$

L et H représentent respectivement la largeur et la hauteur de l'image. L'idée est que chaque groupe de threads dans un block du GPU calcule la proportion de pixels dans chaque niveau de gris inclus dans le block produisant un sous-histogramme ($SubHist$). Ces derniers sont accumulés par la suite dans un histogramme final unique ($Hist$). Cet histogramme est stocké dans la mémoire globale du GPU sous la forme d'un tableau de taille 256 représentant les niveaux de gris de l'image.

Comme le montre l'algorithme 8, $Hist$ est initialisé à 0 en parallèle par les 256 threads du block d'indice 0 (Algorithme 8 : lignes 1-3). Chaque thread applique la transformation gamma sur son pixel correspondant (Algorithme 8 : Ligne 5).

Ensuite, tous les threads incrémentent en parallèle les niveaux de gris correspondant dans $Hist$ (Algorithme 8 : ligne 9). La génération des paramètres Moyenne et Variance est effectuée sur le CPU suivie du calcul de la variance inter-classes. Enfin, le seuil est déterminé comme étant la valeur maximale des variances inter-classe calculées. La complexité après

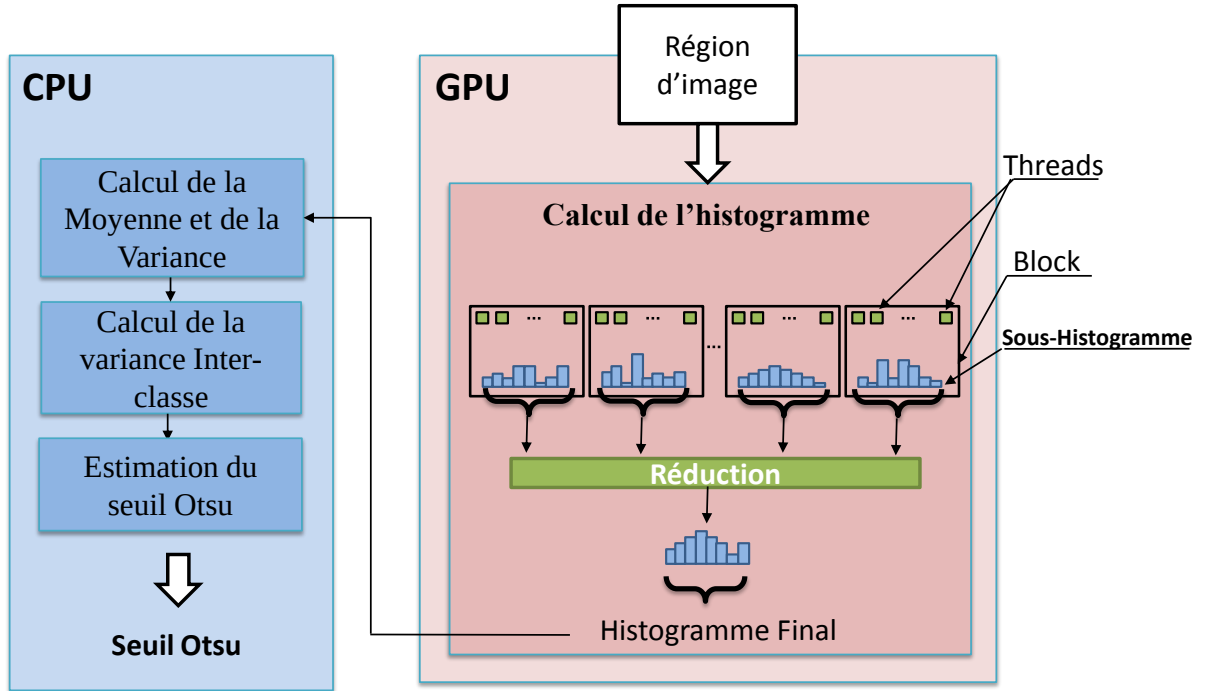


FIGURE 4.7: Stratégie de parallélisation du calcul de l'histogramme sur GPU.

parallélisation revient à $\mathcal{O}((Nb'_{Images} \times Nb_{pixels})/Nb_{Thread}) + \mathcal{O}(Nb'_{Images} \times N_g)$ avec Nb'_{Images} représente le nombre d'image après l'optimisation algorithmique de la correction gamma. Comme $Nb'_{Images} \times Nb_{pixels} \gg Nb'_{Images} \times N_g$ alors la complexité totale de calcul du seuil Otsu est de l'ordre de $\mathcal{O}(Nb'_{Images} \times Nb_{pixels})$.

3.3.2 Génération parallèle des matrices de co-occurrence

Le calcul des matrices de co-occurrence est mieux adapté aux processeurs scalaires qui ne traitent qu'une seule donnée à la fois [192]. Néanmoins, quelques travaux ont parallélisé le traitement sur GPU [192, 193]. La complexité de l'algorithme séquentiel pour la génération d'une matrice de co-occurrence, est de l'ordre de $\mathcal{O}(Nb'_{Images} \times Nb_{pixels})$. Nb_{pixels} représente le nombre de pixels total dans l'image et Nb'_{Images} est le nombre d'image après l'optimisation algorithmique de la correction gamma.

Pour notre parallélisation, nous configurons le GPU en blocks de tailles 16×16 car d'après nos expérimentations cette taille donne un meilleur temps d'exécution. Ensuite, nous calculons les éléments de la matrice de co-occurrence localement et on les range dans les mémoires partagées. Enfin, on somme les résultats de tous les blocs dans la matrice finale qui se trouve dans la mémoire globale. Nous exposons notre stratégie de parallélisation dans l'algorithme

Algorithme 8 Génération de l'histogramme : Algorithme parallèle**Entrées :** I : Image en niveau de gris**Sorties :** $Hist$: Histogramme final avec 256 niveaux de gris

```

1: Pour tout  $thread \in block_0$  Faire en parallèle
2:    $Hist[threadblock_x] := 0$  // Initialisation de Hist (situé dans la mémoire globale) à 0
3: Fin Pour
4: Pour tout  $thread \in [1..nb_{threads}]$  Faire en parallèle
5:    $val := I[thread_x]^\gamma$  // Calcul de la transformation gamma
6:    $SubHist[val]++$  // Chaque block calcule un sous histogramme  $SubHist$  (situé dans
   la mémoire partagée)
7:    $Hist[threadblock_x] += SubHist[threadblock_x]$ 
8: Fin Pour

```

9.

Algorithme 9 Génération parallèle d'une matrice de co-occurrences d'angle $a : 0^\circ$ et de translation $d : 1$: Algorithme parallèle**Entrées :** I : Image en niveau de gris**Sorties :** $CMat_a$: Matrice de co-occurrence, l'angle $a = 0$

```

1: Pour tout  $thread \in [1..Nb_{threads}]$  Faire en parallèle
2:    $ind_{global} = thread_x \times L + thread_y$  // Calcul de l'index global de chaque thread
3:    $threadBlock = threadBlock_x \times L_{Block} + threadBlock_y$  // Calcul de l'index local de
   chaque thread dans son block
4:    $p_1 := I(thread_x, thread_y)$ 
5:    $p_2 := I(thread_x, thread_y + 1)$  // a=0
6:    $ind_{global} := p_1 \times N_g + p_2$ 
7:    $POS := ind_{global} \bmod 2N_g$ 
8:    $Tid[POS] := threadBlock$ 
9:    $Occ[POS]++$ 
10: Fin Pour
11:  $synctreads()$  // synchronize tous les threads
12: Pour tout  $thread \in [1..block]$  Faire en parallèle
13:   Si ( $threadBlock = Tid[POS]$ ) Alors // Désignation du thread qui va accumuler
   l'occurrence correspondante  $Occ[POS]$ 
14:      $CMat_a[ind_{global}] += Occ[POS]$ 
15:   Fin Si
16: Fin Pour

```

D'abord, on distribue les pixels de la mémoire globale vers les registres. Ainsi, chaque thread possède deux valeurs : le niveau de gris du pixel courant p_1 et celui de son voisin p_2 selon la direction traitée (Algorithme 9 : lignes 4 et 5). A partir de ces valeurs, un index global IG pointant l'emplacement de coordonnées (p_1, p_2) dans la matrice de co-occurrence est calculé selon l'équation 4.2.

$$IG = p_1 \times N_g + p_2 \quad (4.2)$$

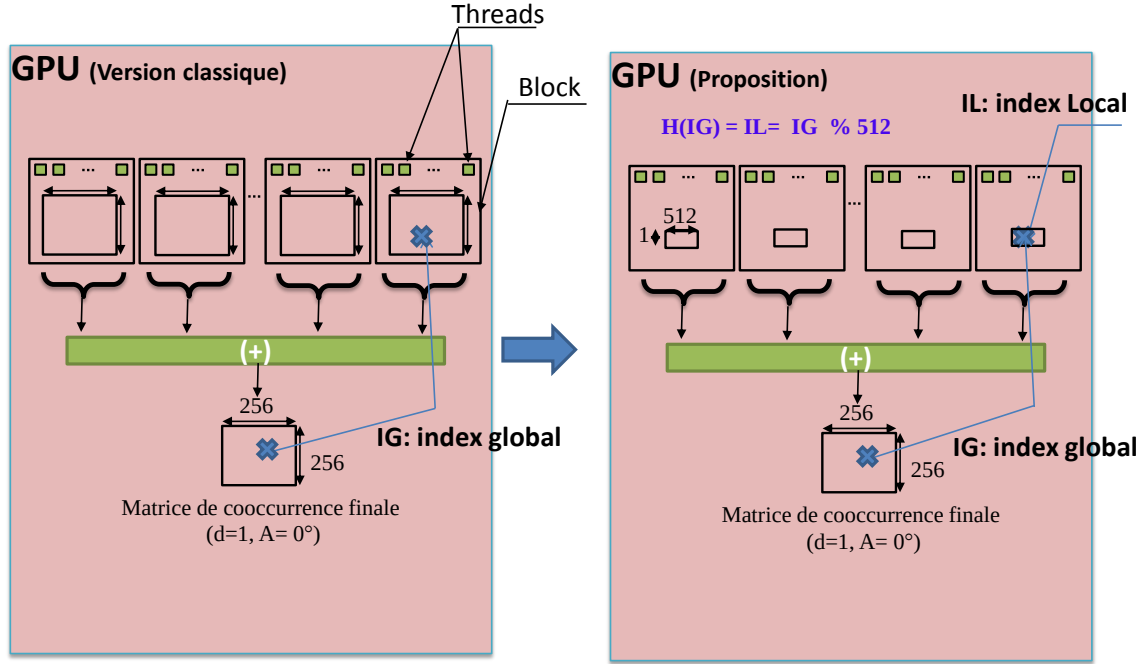


FIGURE 4.8: Stratégie de parallélisation de la génération des matrices de co-occurrences sur GPU.

N_g représente le nombre de niveaux de gris. La valeur IG calculée appartient à l'intervalle $[0, N_g \times N_g]$, comme on traite des images en N_g niveaux de gris. Nous proposons que dans chaque block, au lieu de calculer une matrice de co-occurrence locale de taille $N_g \times N_g$ et la stocker dans la mémoire partagée, on suppose que dans chaque bloc de l'image nous avons au maximum $2 \times N_g$ niveaux de gris possibles.

Pour résoudre ce problème nous proposons de ramener la valeur de l'index globale appartenant à l'intervalle $[0, N_g^2]$ vers un intervalle plus petit $[0, 2 \times N_g]$ à travers une opération de hachage (H). On calcule la nouvelle position $H(IG) = POS$ d'un thread de valeurs de pixels p_1 et p_2 selon la fonction de hachage de l'équation 4.3. IG est l'index global calculé dans l'équation 4.2. E représente le résultat de la division entière de IG par le nombre de niveaux de gris N_g .

$$H(IG) = IG - (N_g) \times E\left(\frac{IG}{N_g}\right) \quad (4.3)$$

Une addition atomique est effectuée pour incrémenter les cases de la matrice de co-occurrence dans la mémoire globale avec le contenu des mémoires partagées. Pour chaque block de threads, on utilise deux buffers *Occ* et *Tid* dans la mémoire partagée. *Occ* contient l'occurrence des valeurs des pixels p_1 et p_2 dans un block de threads. *Tid* contient l'index du dernier Thread qui a mis à jour *Occ*. La complexité de l'algorithme parallèle de la génération des

matrices de co-occurrences est de $\mathcal{O}(Nb'_{Images} \times Nb_{pixels})/Nb_{Thread}$. Nb_{Thread} représente le nombre de threads et il est égale à $Nb_{pixels} \times Nb'_{Images}$. Nb'_{Images} donne le nombre d'image après l'optimisation algorithmique de la correction gamma.

Nous avons calculé les quatre matrices de co-occurrences $CMat_0$, $CMat_{45}$, $CMat_{90}$ et $CMat_{135}$ qui correspondent respectivement aux quatre directions ; 0° , 45° , 90° et 135° .

L'étape suivante consiste à calculer, à partir des quatre matrices, $CMat_0$, $CMat_{45}$, $CMat_{90}$ et $CMat_{135}$, deux des caractéristiques de Haralick à savoir l'énergie E et le contraste C .

3.3.3 Calcul parallèle des caractéristiques de Haralick

L'énergie (E) et le contraste (C) d'une image donnée représentent respectivement les moyennes des énergies élémentaires (E_0 , E_{90} , E_{45} , E_{135}) et des contrastes élémentaires (C_0 , C_{90} , C_{45} , C_{135}) des quatre matrices de co-occurrences.

Le calcul de ces caractéristiques présente plusieurs indépendances. D'une part, le calcul de l'énergie (E) est indépendant du calcul du contraste (C) comme le montre la figure 4.9.

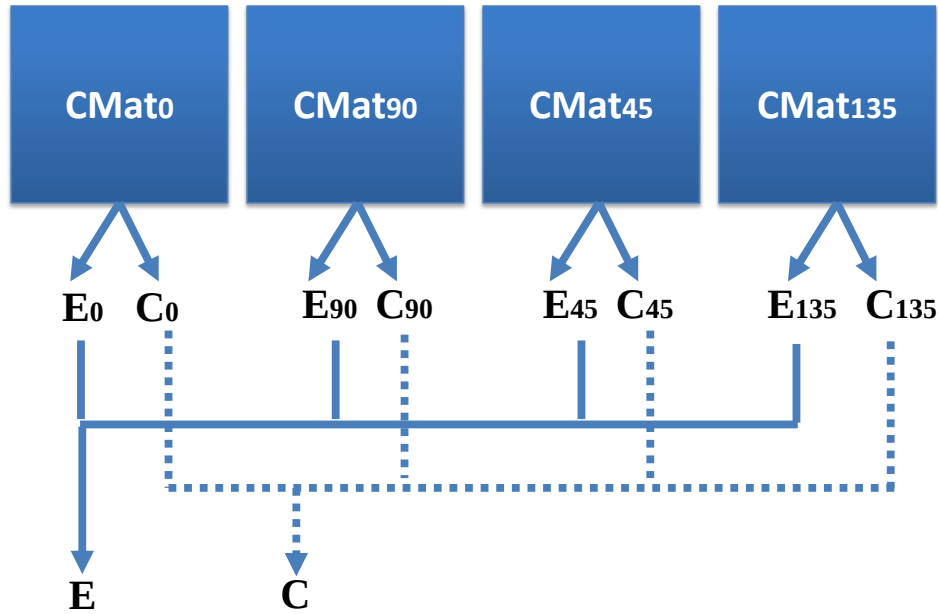


FIGURE 4.9: Indépendance du calcul des énergies et des contrastes à partir des quatre matrices de co-occurrences élémentaires.

D'autre part, le calcul des énergies et des contrastes élémentaires est indépendant entre les quatre matrices de co-occurrences (Figure 4.9). De plus, pour chaque matrice de co-occurrences le calcul d'une énergie et d'un contraste élémentaire (ie : E_0 et C_0) est indépendant entre les éléments des matrices (Figure 4.10).

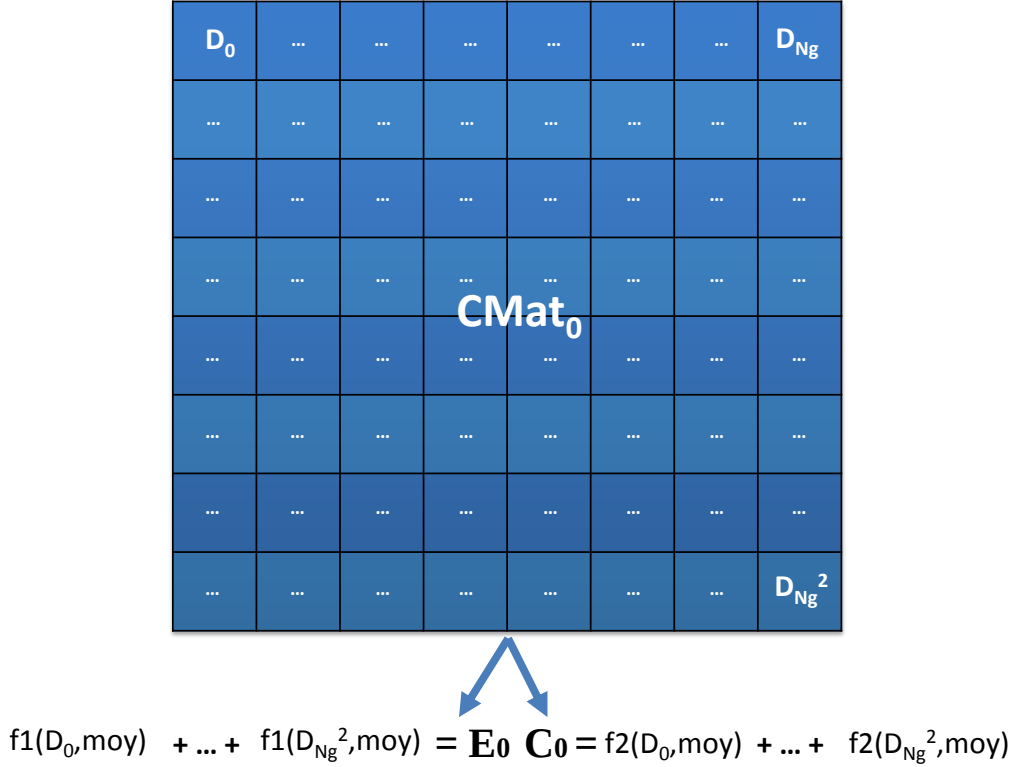


FIGURE 4.10: Indépendance du calcul des énergies et des contrastes pour chaque matrice de co-occurrence élémentaire

Le calcul des caractéristiques élémentaires E_a et C_a est donné par les équations 4.4 et 4.5.

$$E_a = \sum f_1, \quad a \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\} \quad (4.4)$$

$$C_a = \sum f_2, \quad a \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\} \quad (4.5)$$

f_1 et f_2 représentent les opérandes des sommes des caractéristiques élémentaires. Ils sont données par les équations 4.6 et 4.7.

$$f_1 = \frac{CMat^2(i, j)}{moy^2}, \quad i, j \in [0, 255] \quad (4.6)$$

$$f_2 = \frac{(i - j)^2 CMat(i, j)}{moy}, \quad i, j \in [0, 255] \quad (4.7)$$

Nous exploitons ces trois indépendances. Ainsi, nous parallélisons le calcul de l'énergie (E) et du contraste (C) (parallélisme à plus gros-grains). Aussi, nous parallélisons le calcul des énergies et des contrastes élémentaires ($E_0..E_{135}$, $C_0..C_{135}$) entre les quatre matrices de co-occurrences (parallélisme à gros-grains). Nous parallélisons également le calcul des fonctions f_1 et f_2 entre les éléments des matrices (parallélisme à fin-grains).

Comme illustré dans la figure 4.11-a, pour calculer parallèlement l'énergie (E) et le contraste (C), nous configurons le GPU en deux plans de threads d'indices $z \in [0, 1]$. Un plan permet de calculer l'énergie (E) et le deuxième permet de calculer le contraste (C) (parallélisme à plus gros-grains). De plus on répartit les cœurs du GPU en blocks de taille $N_g \times N_g$. Selon le plan z , chaque block calcule l'énergie ou le contraste d'une matrice de co-occurrence (parallélisme à gros-grains). De plus, chaque thread calcule une fonction f_1 ou f_2 (parallélisme à grains fins).

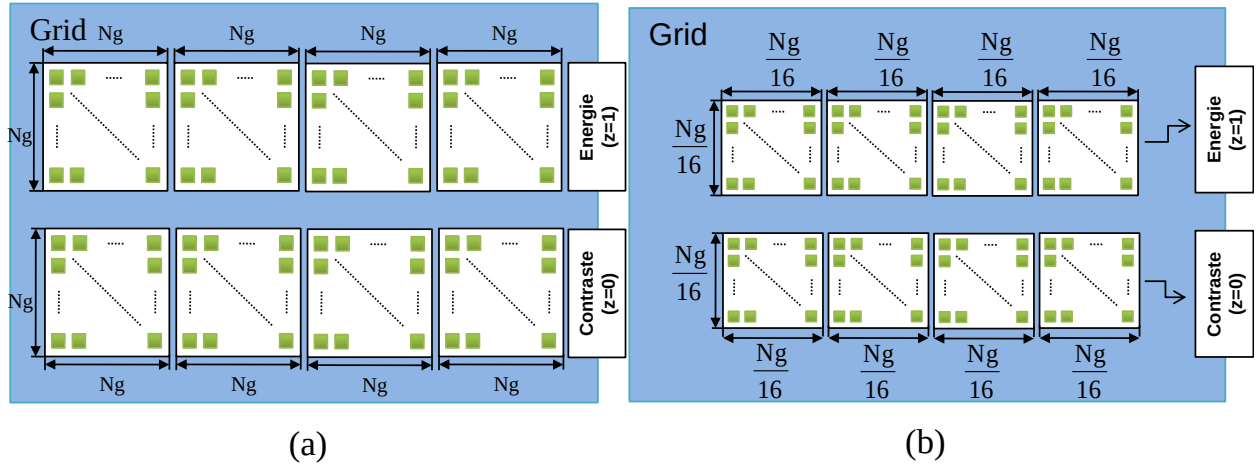


FIGURE 4.11: Génération des caractéristiques de Haralick sur le GPU.

La somme partielle des fonctions f_1 et f_2 est calculée en parallèle. Pour ceci, on utilise une approche dichotomique. Un ensemble de paires de termes est sommé en parallèle. Ensuite, on itère le même traitement sur les paires de résultats. Ce traitement s'appelle réduction parallèle. La figure 4.12 donne un exemple où on somme les termes décimaux d'un tableau. Trois cycles sont suffisants pour calculer la somme du tableau en parallèle.

Notre implantation de la parallélisation se compose donc de deux phases de réduction. D'abord, les threads calculent les sommes partielles (SP) des caractéristiques puis les rangent dans les mémoires partagées du GPU. Ensuite, nous réduisons ces SP pour calculer les caractéristiques finales.

Dans la suite, l'algorithme 10 de notre stratégie de parallélisation est expliqué.

Dans la première phase de réduction, les éléments des quatre matrices de co-occurrences sont distribués à partir de la mémoire globale vers les registres (Algorithme 10 : lignes 4-14). Chaque thread d'un block calcule un terme de l'équation 2.9 et l'additionne aux une sommes partielles (SP). Ce résultat partiel est rangé dans la mémoire partagée du block.

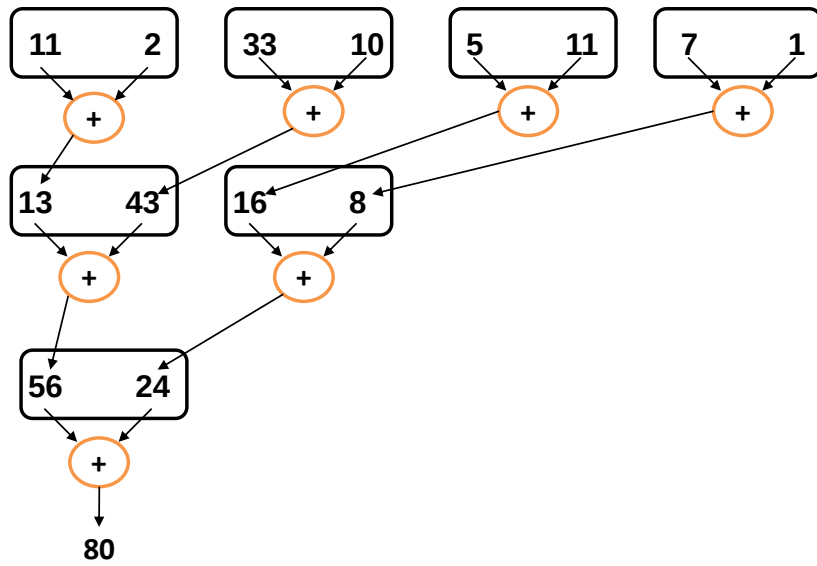


FIGURE 4.12: Exemple de la réduction parallèle, utilisée dans le calcul des caractéristiques de Haralick.

Dans la deuxième phase de réduction, représentée dans la figure 4.11-b, chaque block réduit les sommes partielles d'une matrice de co-occurrences.

Le plan d'indice ($z = 0$) se charge de calculer le contraste tandis que le plan d'indice ($z = 1$) se charge de calculer l'énergie. L'algorithme 11 montre cette réduction.

Ces valeurs sont réduites en une seule valeur comme le montre la figure 4.11.b. Ainsi, on reconfigure le GPU en 8 blocks de tailles 16x16 threads. Chaque block calcule l'énergie E et le contraste C pour chaque matrice. Après réduction, on obtient 4 valeurs de l'énergie (E_a) et 4 valeurs de contraste (C_a) simultanément avec $a \in 0^\circ, 45^\circ, 90^\circ, 135^\circ$.

La complexité de l'algorithme séquentielle est égale à $\mathcal{O}(Nb'_{Images} \times N_g^2)$, tandis que la complexité de l'algorithme parallèle est de $\mathcal{O}(\frac{Nb'_{Images} \times N_g^2}{Nb_{Thread}})$ avec Nb_{Thread} représente le nombre de threads utilisés dans la première phase de la réduction.

La correction gamma présente des étapes de traitement indépendantes, comme le calcul du seuil Otsu et le calcul de l'énergie et du contraste. Dans la suite, nous exploitons cette indépendance en utilisant un nœud de GPU pour accélérer l'ensemble de la méthode.

3.3.4 Implantation sur un nœud de GPU

Généralement, on utilise plusieurs GPUs à cause de l'espace mémoire insuffisant pour l'application sur un seul GPU ou simplement pour ajouter de la puissance de calcul afin d'augmenter le débit de calcul [131].

Algorithme 10 Calcul des caractéristiques de Haralick pour 4 matrices de co-occurrence : Calcul et réduction partielles des caractéristiques dans les mémoires partagées : Algorithme parallèle

Entrées : $CMat_a$: 4 Matrices de co-occurrence, $a \in \{0, 45, 90, 135\}$

Sorties : SP : Sommes partielles (mémoire globale)

```

1: Pour tout  $thread \in [1..Nb_{threads}]$  Faire en parallèle
2:    $ind_{global} = thread_x \times L + thread_y$  // Calcul de l'index global de chaque thread
3:    $threadBlock = threadBlock_x \times L_{Block} + threadBlock_y$  // Calcul de l'index local de
   chaque thread dans son block
4:   Si  $(-1 < thread_y)$  et  $(thread_y < N_g)$  Alors
5:      $data := CMat_0[ind_{global}]$ 
6:      $moy := moy_1$ 
7:   Sinon Si  $(N_g - 1 < thread_y)$  et  $(thread_y < 2 \times N_g)$  Alors
8:      $data := CMat_{45}[ind_{global}]$ 
9:      $moy := moy_2$ 
10:  Sinon Si  $(2 \times N_g - 1 < thread_y)$  et  $(thread_y < 3 \times N_g)$  Alors
11:     $data := CMat_{90}[ind_{global}]$ 
12:     $moy := moy_3$ 
13:  Sinon Si  $(3 \times N_g - 1 < thread_y)$  et  $(thread_y < 4 \times N_g)$  Alors
14:     $data := CMat_{135}[ind_{global}]$ 
15:     $moy := moy_4$ 
16:  Fin Si
17:  Si  $(z = 0)$  Alors
18:     $E_a := \frac{data \times data}{moy \times moy}$ 
19:  Sinon Si  $(z = 1)$  Alors
20:     $C_a := \frac{n \times data}{moy}$ 
21:  Fin Si
22:  Réduction1
23:   $Mem_{globale}[ind_{global}] := Mem_{partagée}[threadBlock]$ 
24: Fin Pour

```

Dans notre contexte, on se propose d'utiliser un ensemble de GPUs (nœud) pour accélérer d'avantage le traitement de nos algorithmes sans avoir une limitation en mémoire. Le nœud employé contient deux GPUs Nvidia. Le premier est un GPU GeForce GTX 660 issue la famille Kepler. Le deuxième est un GPU Quadro 620K issue de la famille Maxwell.

Nous parallélisons les deux étapes indépendantes de la méthode de correction gamma à savoir le calcul du seuil Otsu et le calcul des caractéristiques de Haralick.

Nous privilégions la précision des nombres ainsi nous utilisons des données en double précision pour le calcul des caractéristiques de Haralick [127]. Comme le GPU Kepler est plus adapté pour la manipulation des nombre en double précision, on calcule des caractéristiques de Haralick sur l'architecture Kepler (GPU1) et on cherche le seuil Otsu sur l'architecture

Algorithme 11 Calcul des caractéristiques de Haralick pour quatre matrices de co-occurrence : Réduction finale des caractéristiques dans les mémoires partagées : Algorithme parallèle

Entrées : SP : Sommes Partielles (Dans la mémoire globale)

Sorties : E_a et C_a , $a \in \{0, 45, 90, 135\}$

```

1: Pour tout  $thread \in [1..Nb_{threads}]$  Faire en parallèle
2:    $ind_{global} = thread_x \times L + thread_y$  // Calcul de l'index global de chaque thread
3:    $threadBlock = threadBlock_x \times L_{Block} + threadBlock_y$  // Calcul de l'index local de
   chaque thread dans son block
4:   Si ( $z = 0$ ) Alors
5:      $Mem_{partagée}[threadBlock] := Mem_{globale}[ind_{global}]$ 
6:   Sinon Si ( $z = 1$ ) Alors
7:      $Mem_{partagée}[threadBlock] := Mem_{globale}[ind_{global}]$ 
8:   Fin Si
9:   Réduction2
10: Fin Pour
11: Si ( $z = 0$ ) Alors
12:    $E_a := Mem_{partagée}[0]$ 
13: Sinon Si ( $z = 1$ ) Alors
14:    $C_a := Mem_{partagée}[0]$ 
15: Fin Si

```

Maxwell (GPU2).

Nous décrivons dans la figure 4.13 notre stratégie de parallélisation.

Le CPU assure la communication entre les deux GPUs (GPU1 et GPU2) à travers un balancement automatique des tâches. Pour ceci, on utilise deux threads (Thread 1 et Thread 2) sur le CPU contrôlant respectivement GPU1 et GPU2.

Dans le cas représenté sur la figure 4.13, le GPU2 finit son calcul du seuil Otsu en premier et vérifie l'état du GPU2. Cette vérification est assurée à travers une variable partagée entre les deux Threads 1 et 2. Le GPU2 prend en charge une partie du calcul des caractéristiques de Haralick.

D'après notre étude dans la section 3, la classification hybride basée sur l'algorithme Kmeans est coûteuse en temps de calcul. Dans la suite, nous proposons une parallélisation pour accélérer son traitement.

3.4 Accélération de la classification hybride basée sur le noyau K-means

Pour extraire le texte à partir d'un arrière-plan homogène ou complexe, après l'amélioration de l'image avec la correction gamma (CG), nous avons élaboré la méthode de classification hybride basée sur l'algorithme Kmeans (CHK) [128]. Cette méthode inclut principale-

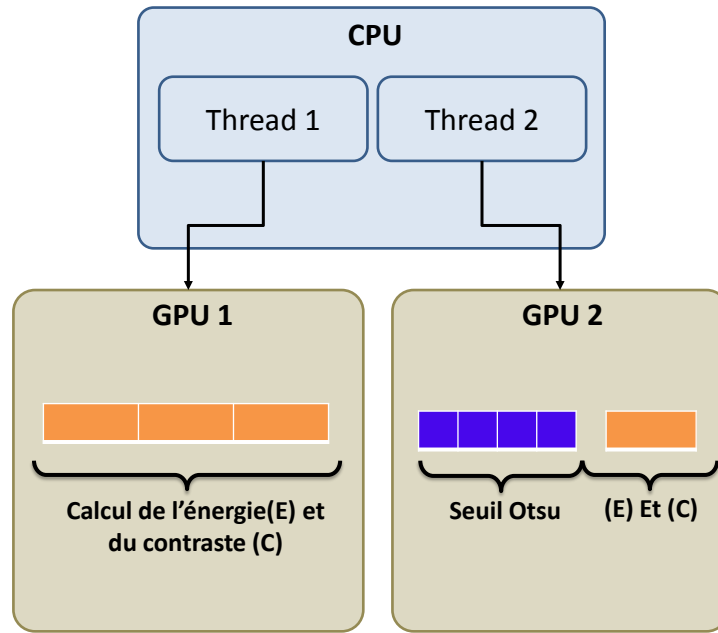


FIGURE 4.13: Implantation de la méthode de Correction Gamma (GC) sur un nœud de GPUs.

ment deux traitements parallélisables. Le premier traitement correspond à l'application du Kmeans indépendamment entre les blocs de l'image. Le deuxième traitement correspond au calcul des distances indépendamment entre les pixels et les moyennes des pixels appartenant aux mêmes classes (centroids) au sein du Kmeans. Nous exploitons ces indépendances pour paralléliser la méthode sur GPU.

Dans la suite, le temps d'exécution de la méthode d'extraction hybride basée sur Kmeans (CHK) est étudié pour distinguer les traitements les plus coûteux en temps de calcul. Le tableau 4.4 montre que le temps d'exécution des différentes étapes de la méthode CHK : initialisation, phase locale et phase globale. L'initialisation comprend les allocations mémoires et le chargement des données. Le processus de la phase locale consiste à exécuter l'algorithme Kmeans sur tous les blocs du document. Le processus de la phase globale comprend la vérification de la convergence globale et la mise à jour des centroids.

Le temps de calcul de l'algorithme Kmeans affiche 99.98% du temps d'exécution total de la méthode CHK. Ainsi, la réduction du temps de traitement du noyau Kmeans diminuera le temps global de la méthode. Dans la suite, nous proposons de paralléliser l'algorithme Kmeans. Nous commençons par introduire un état de l'art sur les implantations GPU du Kmeans existantes dans la littérature.

3.4.1 Accélération de l'algorithme Kmeans sur GPU : État de l'art

L'algorithme Kmeans comporte cinq étapes : (1) L'initialisation des centroids, (2) le calcul des distances, (3) l'accumulation d'une part des valeurs des pixels et d'autre part du

TABLE 4.4: Analyse du temps d'exécution des étapes de la méthode CHK sur CPU

Étapes	Temps (s)	Temps (%)
Initialisation	0.00001	0.001
Phase locale Kmeans	1.369	99.98
Phase globale	0.0001	0.013
Total	1.37	100

nombre des pixels de chaque classe, (4) la mise à jours des centroids et (5) la vérification de la convergence consiste à comparer la valeur du centroid courante avec celle de l'itération précédente. En cas d'égalité, Kmeans converge c'est-à-dire que les centroids ne changent plus de valeurs (Figure 2.18).

Le calcul des distances est communément parallélisé sur GPU car il s'effectue indépendamment entre les points des données [97]. De ce fait, on adopte souvent une configuration GPU ou chaque point de données est traité par un thread exploitant ainsi le parallélisme entre les données.

La différence entre les implantations réside d'une part dans la façon d'utiliser la hiérarchie mémoire. Ainsi, plus les mémoires partagées et les registres sont utilisés, plus le traitement sera rapide à cause de la latence faible de ces mémoires. D'autre part, moins la quantité de traitement exécuté sur le CPU, moins sera la quantité de transfert mémoire entre le CPU et le GPU.

UV-Kmeans [97] est la première mise en œuvre rapide et très populaire du Kmeans sur GPU (Figure 4.14). Dans ce travail, les centroids et les données sont stockées respectivement dans les mémoires de constantes et de texture. En outre, les mémoires les plus rapides tels que les registres et la mémoire partagée ne sont pas utilisés. D'autre part, les opérations d'accumulations et de calcul des nouveaux centroids sont effectuées sur le CPU. Toutefois, ceci augmente la latence mémoire lors du transfert de données entre CPU et GPU.

W. Fang [98] accélère 5x UV-Kmeans [97] en minimisant le transfert mémoire entre CPU et GPU. En effet, il utilise une grille de bits pour accumuler, à l'intérieur du GPU, le nombre de points de données appartenant à chaque classe. Cette technique minimise l'utilisation des opérations atomiques coûteuses en temps de calcul. Cependant, la grille de bits occupe beaucoup d'espace mémoire lorsque le nombre de centroids est élevé. La mise à jour est effectuée sur le CPU. De plus, la gestion de la mémoire n'est pas optimale car les registres et la mémoire partagée ne sont pas utilisés.

Récemment, des travaux [99, 96] effectuent plus d'étapes de l'algorithme Kmeans sur le

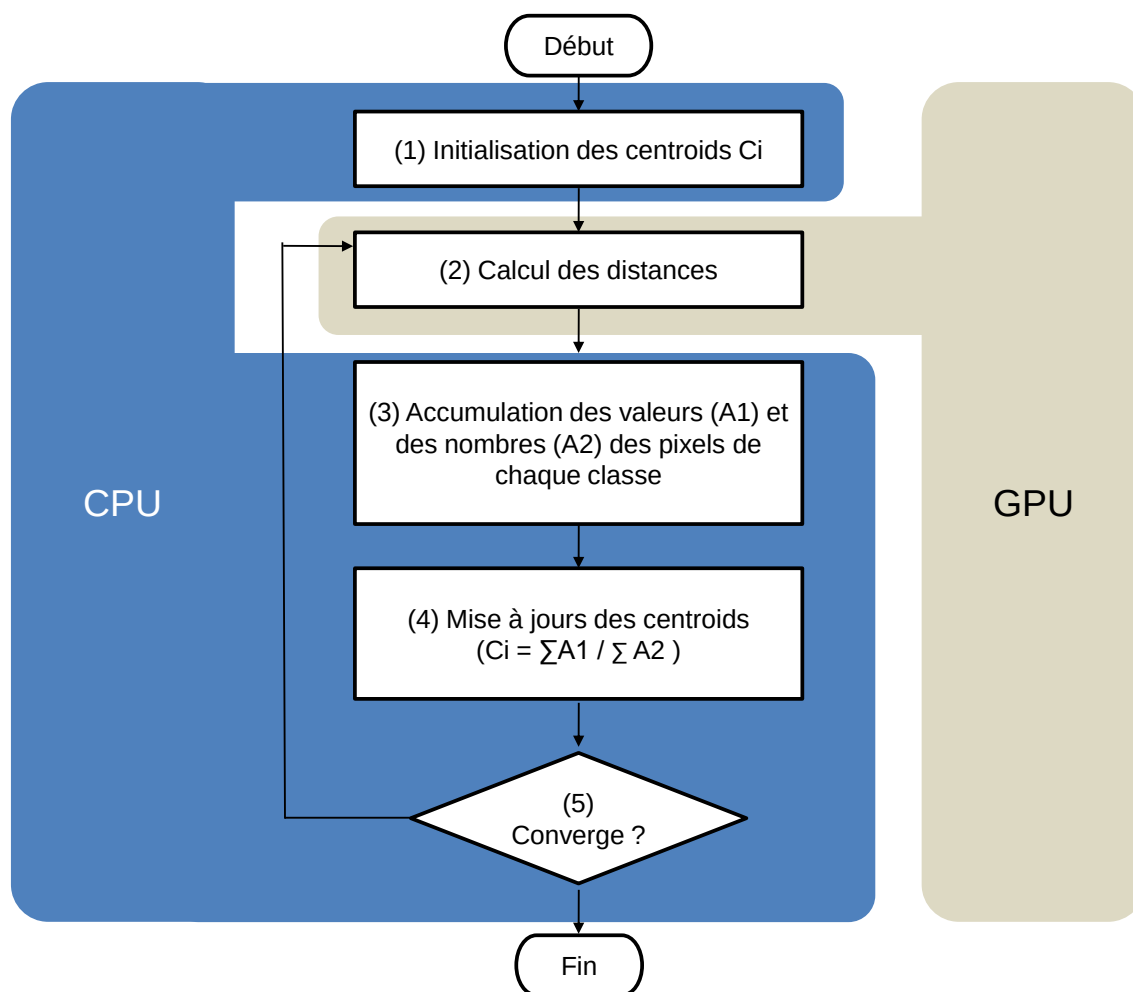


FIGURE 4.14: UV-Kmeans [97] : Première implantation Kmeans sur GPU. Avec A1 : valeur des pixels, A2 : nombre des pixels, Ci : centroids.

GPU. Par exemple, dans [99], les auteurs utilisent le CPU uniquement pour l'initialisation des centroids et contrôler le GPU. En outre, dans [96], plusieurs étapes d'accumulation par réduction sont effectuées sur GPU avant de mettre à jour des centroids sur le CPU. Cette méthode, améliore les implantations existantes [97, 98] en optimisant la gestion mémoire. En effet, les données d'entrée sont stockées dans des registres qui permettent un accès mémoire très rapide.

Dans la méthode HP_Kmeans [117], l'étape de mise à jour des centroids est effectuée sur le CPU, mais une certaine accélération est atteinte en utilisant les transferts asynchrones et les flux cuda (Cuda Streams).

Dans [100], l'algorithme Kmeans a été mis en œuvre dans le contexte de la segmentation d'images. La phase la plus complexe de l'algorithme Kmeans : «l'étiquetage» a été parallélisée. La mise à jour des centroids est effectuée sur le CPU. En effet, ils emploient un petit nombre de centroids par rapport au nombre de pixels [100]. Ainsi, le temps de transfert des données entre le CPU et le GPU est masqué par la tâche de calcul de distance.

Le Tableau 4.5 illustre une étude comparative entre plusieurs implantations GPU du Kmeans comme Kmeans_Li [96], HP_Kmeans [117], UV_Kmeans [97] et GPU_Miner [98]. Les temps d'exécution minimal et maximal de chaque méthode sont rapportés dans le tableau suivant [96].

TABLE 4.5: Comparaison de Kmeans_Li avec différentes implantations Kmeans sur GPU dans la littérature

Méthodes	Temps (s)
Kmeans_Li [96]	0,34 - 1,15
HP_Kmeans [117]	1,45 - 9,03
UV_Kmeans [97]	1,86 - 8,67
GPU_Miner [98]	4,26 - 40,6

Nous pouvons voir à travers cette étude que Kmeans_Li [96] surpasse toutes les méthodes évaluées. Le point commun entre ces méthodes est la réalisation de la mise à jour des centroids sur le CPU, car elles utilisent des architectures GPU incapables de réaliser des opérations atomiques sur des nombres à virgules flottantes. Dans notre travail, nous effectuons une implantation parallèle efficace de CHK sur GPU basée sur un algorithme Kmeans accéléré totalement sur GPU.

3.4.2 Stratégie de parallélisation de la méthode de classification hybride CHK

Deux étapes sont élaborées dans notre proposition de parallélisation de la méthode CHK. Tout d'abord, les caractéristiques locales sont regroupées en utilisant des additions atomiques dans la mémoire partagée à l'intérieur des blocks de threads. Deuxièmement, toutes les caractéristiques locales sont ajoutées à celles globales en utilisant les additions atomiques. Cette technique diminue la surcharge de l'opération atomique. En effet les opérations atomiques sur la mémoire globale peuvent être très coûteuses, car ils ont besoin de sérialiser un nombre potentiellement élevé de threads. Pour réduire cette surcharge, souvent on additionne les variables déclarées dans la mémoire partagée de chaque block de threads [113]. Chaque traitement sur un block s'arrête lorsque les centroids de l'itération courante sont égales à ceux de l'itération précédente. Cela signifie que le critère de convergence de Kmeans est atteint.

Par la suite, le CPU lance le kernel global qui calcule les centroids φ_G (Algorithme 12 : ligne 9). Pour ce faire, on n'a pas besoin de transférer les accumulateurs $Accu_G$ et les compteurs $Count_G$ vers le CPU, en revanche nous les gardons dans la mémoire globale pour minimiser les transferts mémoire CPU-GPU coûteux en temps d'exécution.

Si les centroids de l'itération globale courante sont égaux à ceux de la précédente, la convergence globale est atteinte. Dans ce cas, un drapeau (*flag*) est mis à 1, sinon il est mis à 0. Par la suite, le CPU vérifie si *flag* est égal à 1 pour décider de terminer ou pas le programme comme indiqué dans la figure 4.15.

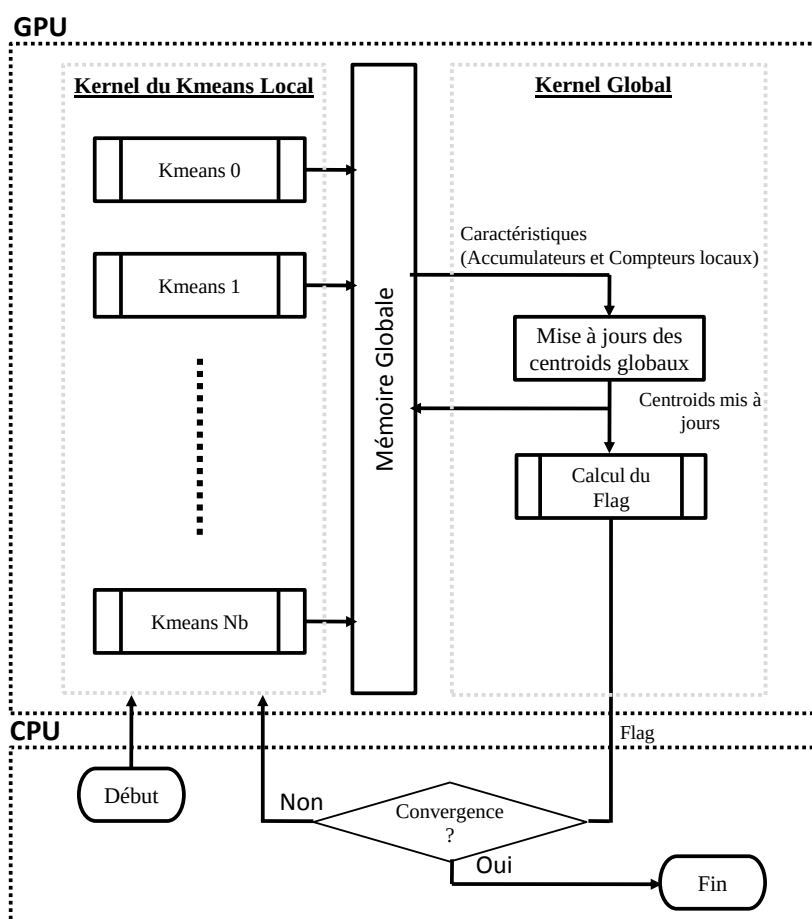


FIGURE 4.15: Stratégie de l'implantation parallèle de la méthode CHK sur GPU.

Pour permettre au CPU de vérifier la variable *flag*, cette variable est transférée du GPU vers le CPU. Vu sa petite taille (1 Byte), ce transfert possède une faible latence mémoire ce qui ne dégrade pas les performances de notre implantation CHK.

Il est important de savoir exploiter le parallélisme disponible dans les applications à différentes granularité [197]. Cette granularité est contrainte par les caractéristiques des algorithmes et de matériel sur lequel elle tourne. C'est important DE sélectionner la granularité adéquate pour exposer d'avantage des degrés de parallélismes.

Le parallélisme à grain fin est utilisé en cas de faible intensité d'arithmétique. Par exemple elle peut se traduire par le parallélisme au niveau instruction. Le parallélisme à gros grains nécessite une haute intensité arithmétique. Un exemple de parallélisme à gros-grain est de

distribuer l'ensemble des itérations de boucles à travers les processeurs. La granularité la plus efficace dépend de l'algorithme et l'environnement matériel dans lequel il est exécuté. Dans la plupart des cas, si la durée de la communication et de la synchronisation est élevée par rapport à la durée de la tâche de calcul, il sera généralement avantageux de travailler en gros-grain. Nous réalisons *un parallélisme à fins et à gros grains* (Figure 4.16).

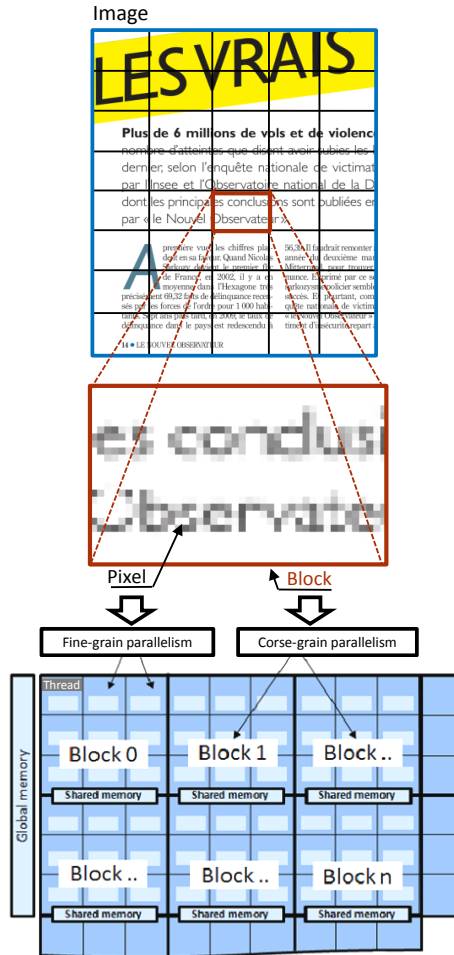


FIGURE 4.16: Parallélisme à Gros et fin grains dans l'implantation de CHK sur GPU.

Dans le niveau gros-grains, nous parallélisons le traitement entre les blocs du document. Ainsi, nous parallélisons la méthode Kmeans entre les blocks de GPU. Dans le niveau fin-grains, nous parallélisons le traitement des pixels à l'intérieur des blocks [108]. C'est pourquoi nous parallélisons le calcul des distances et les étapes d'étiquetage entre les threads (Figure 4.17).

Nous concevons deux fonctions CUDA (appelées aussi kernels). Elles gèrent respectivement les phases locale et globale de la méthode CHK (Figure 2.19) d'où leurs noms respectifs : fonction locale et fonction globale.

Ainsi, dans la fonction locale, l'algorithme Kmeans est lancé en parallèle sur tous les blocs de l'image. D'autre part, la mise à jour des centroids est réalisée dans la fonction globale.

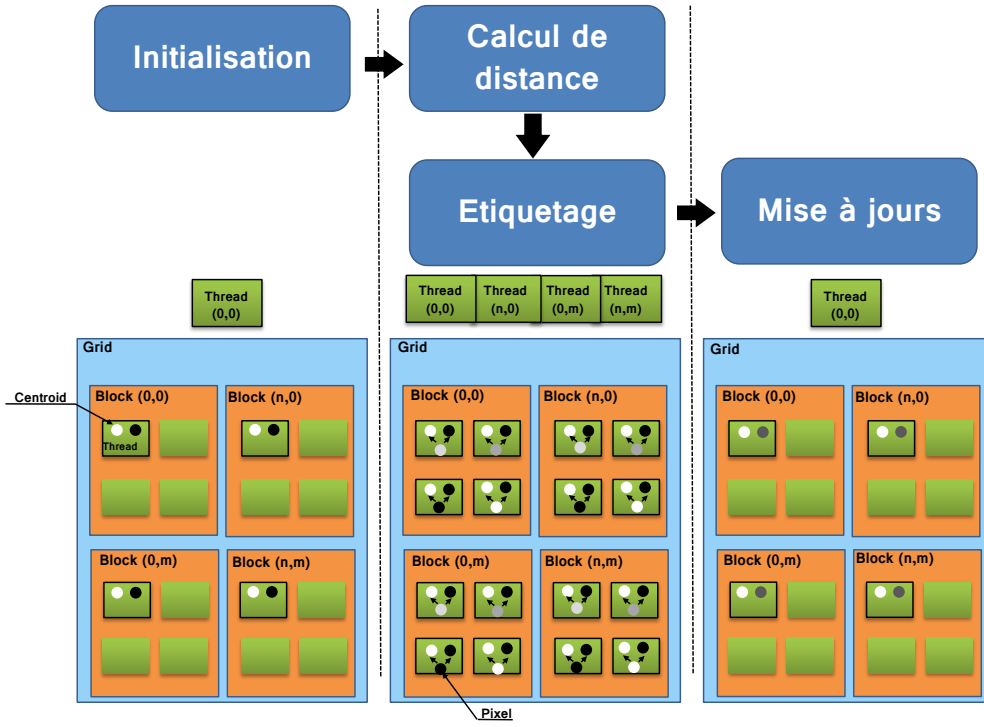


FIGURE 4.17: Stratégie d'implantation de l'algorithme Kmeans sur GPU.

Algorithme 12 Classification hybride basée sur le noyau Kmeans (CHK) : Algorithme parallèle

Entrées : b_l : Bloc d'image

Sortie : I_b : Image binaire

Sorties : C_{G0}, C_{G1} : Les deux centroids globaux

- 1: $\varphi_{G0} := (0, 0, 0)^T, \varphi_{G1} := (255, 255, 255)^T$
 - 2: **Pour Tout** $b_l \in [0..Nb - 1]$ **Faire en parallèle**
 - 3: $Kmeans(b_l, \varphi_G, I_b)$
 - 4: **Pour** $i \in [0, 1]$ **Faire**
 - 5: $Accu_{Gi} := Accu_{Gi} + Accu_{Lb,i}$
 - 6: $Count_{Gi} := Count_{Gi} + Count_{Lb,i}$
 - 7: **Fin Pour**
 - 8: **Fin Pour**
 - 9: $\varphi_{Gi} := \frac{Accu_{Gi}}{Count_{Gi}}; i \in [0, 1]$
-

Une gestion efficace de la mémoire est assurée en employant des mémoires rapides. Pour cela, nous limitons l'utilisation de la mémoire globale en envoyant les données vers la mémoire partagée et les registres. En effet, les centroids locaux φ_L , les accumulateurs $Accu_L$ et les compteurs $Count_L$ de chaque bloc sont alloués dans la mémoire partagée pour un accès rapide aux blocks de threads. En outre, les pixels sont transférés de la mémoire globale vers

les registres ce qui permet un accès mémoire très rapide au cours de l'exécution du programme. Les centroides φ_G , accumulateurs $Accu_G$ et compteurs $Count_G$ sont alloués dans la mémoire globale car ils sont partagés par tous les thread du GPU.

Le calcul de la distance est lancé indépendamment entre les pixels. Par conséquent, nous traitons en parallèle Nb_{Thread} threads. La complexité est de l'ordre de $\mathcal{O}(((L \times H) \times k) / Nb_{Thread})$ avec $L \times H$ est le nombre total de pixels dans le document numérisé et k est le nombre de centroides dans chaque bloc. Le nombre de threads doit correspondre au nombre total de pixels $L \times H$. De même, l'étape de l'étiquetage de Kmeans, où chaque pixel est affecté à sa classe la plus proche, est parallélisé sur Nb_{Thread} threads.

Dans notre implantation CHK, nous *diminuons le temps de communication CPU-GPU* en exécutant entièrement Kmeans sur le GPU y compris la mise à jour des centroides. Dans chaque block, un thread maître effectue la mise à jour des centroides. Nous soulignons que le document numérisé contient un nombre élevé de blocs car la taille de ces derniers est relativement petite par rapport à l'image. De cette façon, le nombre total de threads maîtres dans le GPU reste élevé ce qui maintient une haute occupation du GPU. En outre, la mise à jour des centroides sur le GPU est possible grâce à la disponibilité de la division entière et de l'addition atomique sur des nombres à virgule flottante pour des architectures GPU récentes comme Fermi et Kepler [112].

Durant l'étape de l'étiquetage dans l'algorithme Kmeans, une réduction est réalisée sur chaque block. Ainsi, les variables locales $Accu_L$ et $Count_L$ sont calculées avant d'être additionnées dans les variables globales : $Accu_G$ et $Count_G$, comme illustré dans la figure 4.18.

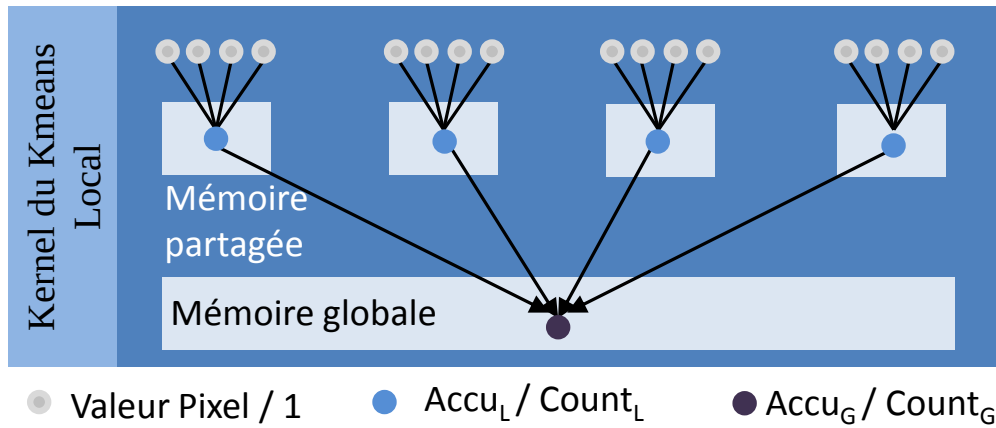


FIGURE 4.18: Optimisation du calcul des accumulateurs et des compteurs (resp $Accu_L$ et $Count_L$).

Dans la section qui suit, nous évaluons notre implantation de la méthode de classification hybride basée sur Kmeans (CHK) qui extrait le texte des régions de texte homogènes, ainsi

que des régions d'images après amélioration avec la correction gamma.

3.5 Accélération de l'analyse de structure du document (Séparation et localisation du texte)

La séparation texte/image [56] a été accélérée sur GPU en utilisant le paradigme de développement logiciel OpenCL [16]. Plus précisément la partie accélérée concerne la détection des traits dans le document. C'est une étape complexe comme nous l'avons expliqué dans le chapitre 2. On rappelle que la détection des traits verticaux et horizontaux nécessite les opérations suivantes :

- (a) Fermeture morphologique : dilatation suivie d'une érosion,
- (b) Ouverture morphologique : érosion suivie d'une dilatation,
- (c) Soustraction (a) - (b),
- (d) Ouverture morphologique horizontale,
- (e) Ouverture morphologique verticale.

Les opérations de fermeture et d'ouverture s'appuient sur les opérateurs morphologiques de la dilatation et de l'érosion. Dans cette section nous expliquons comment ces opérateurs ont été accélérés sur GPU.

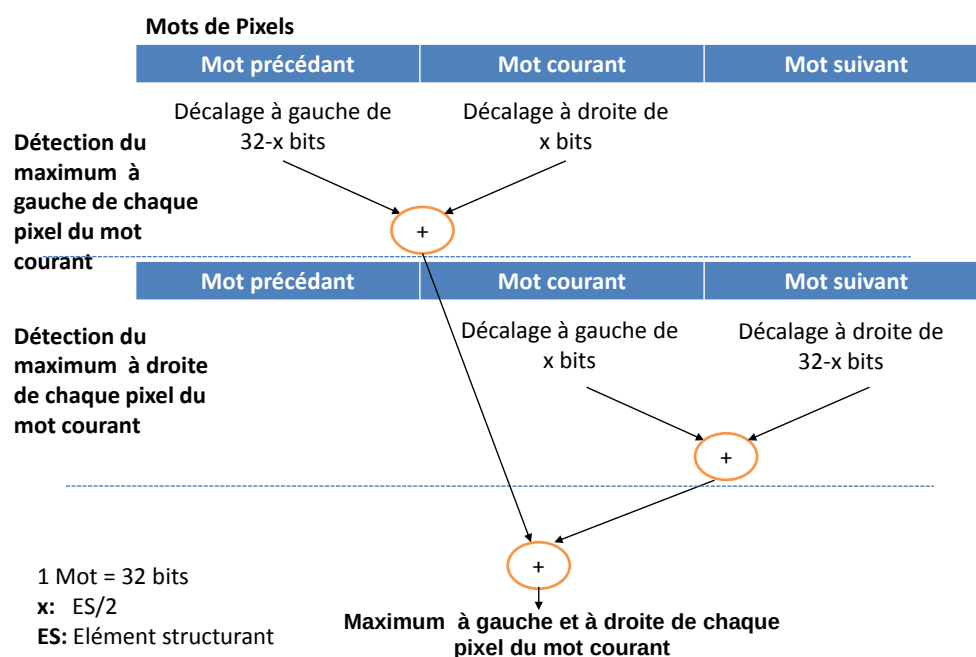


FIGURE 4.19: Kernel de la phase horizontale de dilatation.

On considère une image binaire où chaque pixel est codé sur un bit en mémoire. La dilatation et l'érosion sont effectuées selon deux passes successives : horizontales puis verticales.

Dans les phases horizontales de dilatation et d'érosion, on configure le GPU en une seule dimension de taille $L \times H$ avec L et H représentant respectivement la largeur et la hauteur de l'image. Chaque block du GPU contient 256 threads et chaque thread traite un mot de 32 bits c'est à dire 32 pixels en parallèle.

Pour la phase horizontale de dilatation, on utilise le kernel illustré dans la figure 4.19 pour vérifier si l'élément structurant ES possède un Hit. On dit qu'un élément structurant possède un Hit si l'un de ces éléments touche un pixel blanc de l'image (égal à 1). En cas de Hit la valeur du pixel courant, qui correspond au centre de l'élément structurant, est mise à 1 dans l'image.

On considère les trois mots «*précédant*», «*courant*» et «*suivant*» qui désignent la succession des pixels dans la mémoire. En se basant sur les décalages droit et gauche, on détecte d'abord la valeur maximale à gauche de chaque pixel *courant* (pixels du mot courant). Ensuite on détecte la valeur maximale à droite de chaque pixel. On additionnant les deux résultat par un 'ou' logique les pixels courant qui possède un voisin égal à 1 (cas d'un Hit) sont mis à 1.

La phase horizontale de l'érosion possède le même traitement que celui de la dilatation, sauf qu'on utilise le 'et' logique à la place du 'ou' logique pour déterminer le minimum à droite et à gauche de chaque pixel.

Dans les phases verticales de dilatation et d'érosion on configure le GPU en deux dimensions (H et L). Pour la dilatation verticale on utilise un 'ou' logique pour additionner les 32 bits qui représentent les pixels *courant* avec ceux en haut et ceux en bas pour remettre à 1 les pixels qui possèdent un voisin vertical égal à 1.

Pour l'opération verticale de l'érosion on utilise un 'et' logique qui additionne les 32 bits représentant les pixels *courant* avec ceux en haut et ceux en bas pour remettre à 0 les pixels qui possèdent un voisin vertical égal à 0.

4 Évaluation de la stratégie de parallélisation de la méthode EHT

Dans cette section, nous évaluons le temps d'exécution de la méthode d'Extraction Hybride de Texte (EHT). Nous présentons d'abord la base d'images et les architectures matérielles employées. Ensuite, nous évaluons l'ensemble de la méthode CGCHK pour extraire le texte depuis les régions avec un fond complexe.

Nous évaluons la méthode CHK pour extraire le texte depuis les régions avec un fond homogène. Pour ceci nous discutons d'abord l'efficacité de notre implantation du noyau Kmeans sur le GPU. Nous comparons ensuite le temps d'exécution de CHK entre les plateformes GPU et CPU pour montrer l'accélération atteinte. Plusieurs expérimentations ont été réalisées en variant la taille de blocs, la quantité de bruit (Documents scannés et non scannés) et le contenu du document (zones de texte, fond coloré). En ce qui suit, nous présentons la base de documents hétérogènes ainsi que les architectures CPU et GPU employées pour notre évaluation.

4.1 Outils de développement et environnement de travail

La même base d'images LRDE-DBD [103], utilisée dans les chapitres précédents, est employée dans notre évaluation. Celle-ci est composée de documents hétérogènes de type magazine pour évaluer notre méthode d'extraction hybride de texte. Cette base a été sélectionnée pour l'hétérogénéité de ces documents en terme de régions d'images et de texte en plus des documents scannées et non-scannées qu'elle offre. D'autre part, nous utilisons un PC avec un processeur CPU Intel i3 de 3.07 GHz. Nous exploitons également deux cartes graphiques NVidia respectivement de types GeForce GTX 660 et Quadro K620 dont les principales caractéristiques sont représentées dans le tableau 4.6.

TABLE 4.6: Les GPU employés

GPU	GeForce GTX 660 Kepler (GK106)	Quadro K620 Maxwell (GM107)
Nombre de cœurs	960	1059
Fréquence (MHz)	980	190
Mémoire globale (MB)	2048	2048

De plus, la version CUDA 5.0 est utilisée. L'accélération est extraite en se basant sur les temps d'exécution calculés après le transfert mémoire entre CPU et GPU.

4.2 Évaluation de la parallélisation de la correction gamma

Dans cette section nous évaluons le temps d'exécution de la méthode de correction gamma (CG). Dans ce contexte, Nous analysons ces trois étapes de calcul du seuil Otsu, de génération des matrices de co-occurrences et le de calcul de l'énergie et du contraste.

D'après le tableau 4.7, nous avons accéléré les méthodes de calcul de seuil, de génération des matrices de co-occurrences et de calcul des caractéristiques de Haralick. Aussi, notre implantation Multi-GPU a permis de chevaucher le temps d'exécution du calcul du seuil Otsu avec le reste des méthodes et ainsi accélérer 26x la méthode de correction gamma.

4.3 Évaluation de la parallélisation de la méthode CHK

Dans la première partie de cette section, nous évaluons notre implantation du Kmeans, noyau de la méthode CHK, en termes de temps de calcul avec les méthodes déjà existantes dans la littérature. Dans la deuxième partie de cette section, nous évaluons notre implantation CHK sur GPU pour montrer son accélération par rapport à sa version sur CPU. En outre, la performance est évaluée selon différents paramètres du document (quantité de bruit et complexité de contenu).

TABLE 4.7: Évaluation du temps d'exécution de la méthode CG sur la base LRDE-DBD

Méthode	CPU (ms)	GPU (ms)	Multi-GPU (ms)
Calcul du seuil Otsu	1800	14,66	-
Génération des matrices de co-occurrences	1670	95,19	-
Calcul de l'énergie et du contraste	1460	37,64	-
Total	3470	147.49	132.83

Nous comparons notre parallélisation du Kmeans avec la meilleure mise en œuvre à notre connaissance celle de Kmeans_Li [96]. Nous appelons notre implantation de l'algorithme Kmeans «Kmeans_GPU».

Kmeans_Li[96] fournit une des implantations les plus efficaces de Kmeans sur GPU. Dans son travail, les données sont divisées en blocs de 256 points de données. En outre, les centroids sont stockés dans la mémoire constante. Les données sont transférées une seule fois au départ à partir de la mémoire globale vers les registres. Chaque thread calcule la distance entre un point de donnée et tous les centroids et détermine le centre le plus proche. Pour le calcul des nouveaux centroids, une stratégie de *diviser pour régner* est employée. En effet, les données sont divisées en n' groupes, où n' est initialisée à $\frac{n}{M}$. Avec n est la taille de données appartenant au centroid correspondant, et M est un multiple du nombre de multiprocesseur dans le GPU. Chaque groupe est réduit pour obtenir un centroid temporaire. N' est ensuite mis à jour, en divisant les centroids temporaires par n' pour les groupes et les réduire de manière itérative sur le GPU jusqu'à ce que n' soit inférieur à M , indiquant que le calcul sur GPU n'a plus d'avantage par rapport au calcul sur CPU. Enfin, les centroids finaux sont calculés sur le CPU.

Contrairement à la méthode de Y.Li [96], nous utilisons des registres pour stocker les points de données. Les registres fournissent un accès rapide à la mémoire pendant l'exécution du programme ce qui permet d'accélérer d'avantage Kmeans. D'autre part, nous utilisons les mémoires partagées pour stocker les valeurs des centroids. Au début, chaque thread détermine son centroid le plus proche. Ensuite, les centroids sont calculés directement à partir de la première distribution de pixels sur les blocks du GPU et ils sont stockés dans la mémoire partagée. Enfin, nous réduisons ces centroids sur le GPU jusqu'à la convergence.

Y.Li [96] a évalué sa mise en œuvre en utilisant un ensemble de données de 4 millions de points sur une architecture de GPU GTX 280 et un processeur i5. Nous avons repris l'implantation Kmeans_Li disponible sur notre architecture matérielle. En outre, nous avons utilisé la base de données LRDE-DBD composé de documents de 8 millions de pixels ce qui revient à un ensemble de données plus grand que celui employé par Y.Li. Le tableau 4.8

montre la comparaison entre les temps d'exécution de Kmeans_GPU et Kmeans_Li sur l'ensemble de données décrit.

TABLE 4.8: Évaluation du temps d'exécution de Kmeans_GPU et Kmeans_Li sur 125 documents de la base LRDE-DBD

Méthodes	Temps (ms)
Kmeans_GPU [58]	425
Kmeans_Li [96]	600

Kmeans_GPU est 1,45x plus rapide que Kmeans_Li. En effet, dans Kmeans_GPU nous ne redistribuons pas les pixels sur le GPU, après calcul de la distance. Par conséquent, nous gagnons en temps de traitement et nous assurons une lecture coalescente de la mémoire. Toutefois, dans la mise en œuvre de Y.Li, après le calcul de distance, les données sont redistribuées sur le GPU pour réduire les centroids temporaires ce qui prend un temps d'exécution considérable. En effet, le GPU est configuré avec un nouvel indice de block et des nouveaux indices de pixels sont calculés. Ainsi, les indices de pixels deviennent désordonnés et provoquent une non coalescence dans la lecture mémoire. En outre, dans Kmeans_GPU tous les centroids sont calculés parallèlement à l'intérieur des blocks à la différence de Kmeans_Li qui ne les réarrange pas.

4.3.1 Variation de la taille des blocs

Dans la suite, nous évaluons les performances de notre implantation de la méthode CHK sur GPU. On varie la taille des blocs pour voir son effet sur l'accélération GPU. Dans ce contexte, le tableau 4.9 montre la moyenne du temps d'exécution CHK sur 125 documents numérisés.

L'accélération de CHK diminue lorsqu'on augmente la taille des blocs. En fait, l'exécution sur le GPU est plus rapide lorsque le nombre de pixels dans les blocs d'image est proche du nombre physique des threads dans les blocks du GPU. Dans ce cas, chaque pixel est traité directement par un thread. Dans le cas contraire, le pixel suit une file d'attente jusqu'à ce qu'une ressource (thread) se libère. De plus, le taux de divergence de branches est inférieur lorsqu'on utilise peu de pixels. En effet, tous les threads dans un même block (Warp) exécutent la même instruction à n'importe quel instant t (exécution en SIMD). La divergence de branches se produit lorsque des threads dans un Warp se dirigent vers un chemin d'exécution différent des autres. Dans le cas de divergence de branches présenté dans la figure 4.20 on peut voir que les branches d'exécution sont sérialisés. Les threads exécutent l'instruction Inst2 selon le chemin d'exécution 2 puis, le reste des threads exécutent l'instruction Inst3 selon le chemin d'exécution 1. Dans ce cas on perd 50% de la performance.

TABLE 4.9: Évaluation du temps d'exécution de la méthode CHK sur CPU et GPU sur différentes tailles de blocs

Méthode	Blocs (pix)	Plateforme	Temps (ms)	Accélération
CHK	8x8	CPU	2716	-
		GPU	388	7x
	16x16	CPU	2546	-
		GPU	425	6x
	32x32	CPU	2411	-
		GPU	1050	2,29x

En affichant 388 ms et 425 ms sur GPU, le CHK est capable d'être intégré dans l'application de dématérialisation de document qui nécessite de binariser un document en 460 ms. Pour ces raisons, nous considérons que la taille de bloc 16x16 pixels est la plus appropriée pour notre application, car elle assure un traitement à la fois rapide et de qualité de binarisation de texte acceptable [84].

4.3.2 Variation de la quantité du bruit

Dans la suite, nous discutons la performance de notre implantation en variant la quantité de bruit. La figure 4.21 illustre le temps de traitement CHK sur le CPU et le GPU, représentés respectivement par les couleurs bleu et un rouge en pointillée. Cette évaluation est effectuée à travers 125 documents scannés et non scannés de la base LRDE-DBD. Le bruit est plus important sur les documents scannés (Figure 4.21.b) que sur ceux non scannés (Figure 4.21.a) en raison de la dégradation engendrée par la numérisation du document. Le temps d'exécution varie considérablement entre les deux types de documents sur le CPU. Toutefois, il reste à peu près constant sur le GPU. En effet, la complexité de bruit induite dans les documents numérisés nécessite plus de tâches de calcul qui sont efficacement parallélisées sur le GPU. Pour faire une comparaison pertinente, nous considérons la moyenne de temps d'exécution CHK sur les 125 documents de la base LRDE-DBD comme indiqué dans le tableau 4.10.

La version GPU de CHK est 6x plus rapide que celle du CPU. D'autre part, le traitement de CHK sur GPU est plus rapide sur les documents non scannés, affichant 347 ms par rapport à 425 ms sur les documents scannés. En effet, le temps d'exécution pour les documents scannés est supérieur à celui sur les documents non scannés à cause du nombre d'itérations supplémentaire que nécessite l'algorithme Kmeans pour réduire la distorsion dans les documents scannés ce qui affecte le temps global de la méthode. Contrairement à la mise en œuvre de CHK sur CPU, notre parallélisation CHK sur GPU est peu sensible au bruit introduit dans les documents par la numérisation.

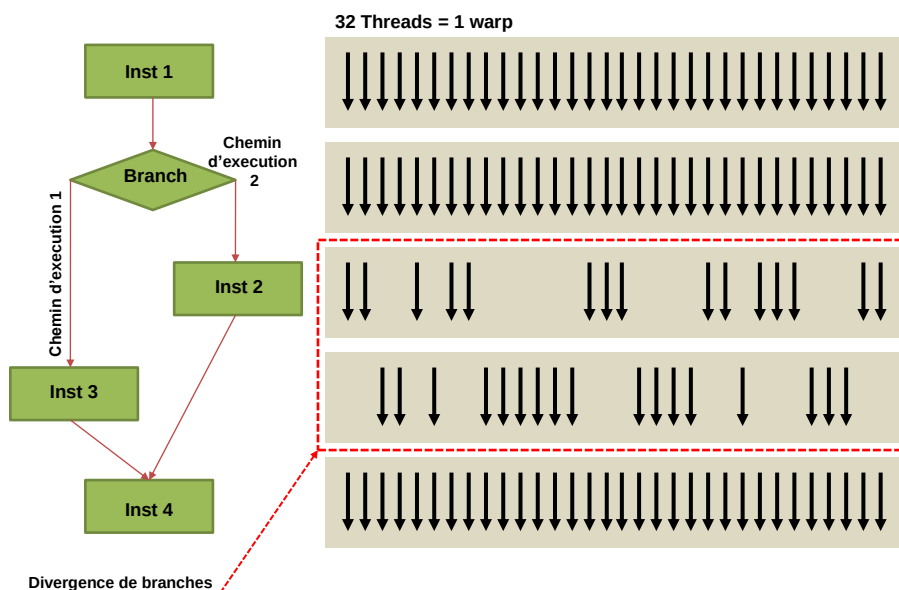


FIGURE 4.20: Exemple de divergence de branches.

En plus du bruit qui peut apparaître dans les documents scannés, le contenu de ces documents est très variable. Il peut inclure du texte, des images, du fonds coloré, etc. Dans la suite on se propose d'étudier le comportement de notre implantation GPU de la méthode CHK en variant le contenu des documents scannés.

4.3.3 Variation du contenu des documents

En ce qui suit, nous évaluons le temps de traitement CHK en faisant varier le contenu du document (nombre de zones de texte, fond coloré). Dans la figure 4.22 le document1 contient quelques zones de texte. Le document2 comprend plusieurs zones de texte et le document3 contient de nombreuses zones de texte et un fond couleur. Le temps CPU augmente à travers les trois types de documents. En effet, plus un document inclut du texte plus le temps de traitement est grand. Ainsi, il atteint 4000 ms sur le document3 par rapport à 1100 ms sur le document1. Le temps d'exécution du GPU est à peu près constant (425 ms) sur les trois types de documents. En outre, l'accélération GPU augmente avec l'augmentation de la complexité du document. En effet, plus les documents comprennent des informations textuelles, plus on a besoin de ressources de calculs. Ainsi, le GPU atteint une accélération de 7,5x pour la binarisation du document3 par rapport à 3,5x pour la binarisation document1.

En effet, la mise en œuvre de l'implantation de CHK sur GPU fournit un temps de binarisation de 460 ms quel que soit le bruit ou le contenu dans les documents. Dans la suite, nous comparons le temps d'exécution de la méthode EHT pour extraire le texte à partir des documents hétérogènes.

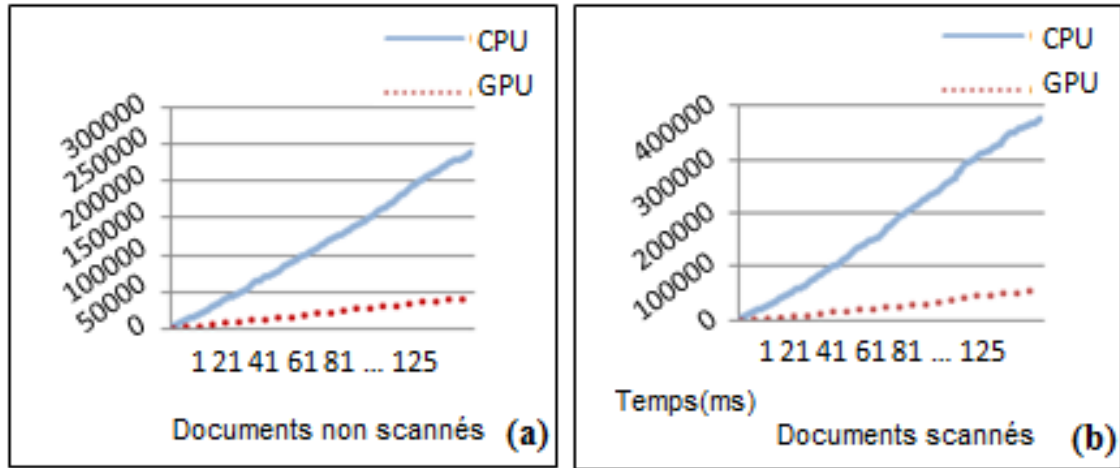


FIGURE 4.21: Évaluation du temps d'exécution de la méthode CHK sur CPU et GPU : (a) : Documents Non-scannés ; (b) : Documents scannés.

TABLE 4.10: Évaluation du temps d'exécution de la méthode CHK sur CPU et GPU sur des images bruitées

Méthode	Images	Plateformes	Temps (ms)	Accélération
CHK	Scannée	CPU	2546	-
		GPU	425	6x
	Non scannée	CPU	1930	-
		GPU	347	5.5x

4.4 Évaluation de la parallélisation de la méthode d'analyse de structure de documents

Nous mesurons le temps d'exécution de l'algorithme de l'analyse de structure de documents sur le CPU et le GPU. Le tableau 4.11 résume cette évaluation. Ainsi, le temps moyen d'exécution sur le CPU est égal à 2.5 seconde, tandis que celui sur le GPU est égal à 0.668 secondes. En effet, la parallélisation des opérateurs de morphologie mathématique intensivement employées a permis d'accélérer presque 4x le temps de traitement.

4.5 Évaluation de l'accélération proposée de la méthode EHT

Dans cette partie, nous évaluons la parallélisation de l'ensemble de la méthode EHT. Nous résumons dans le tableau 4.12 les temps d'exécution des différentes étapes de la méthode EHT et son temps d'exécution total sur GPU qui correspond à 1.26 secondes (en gras).

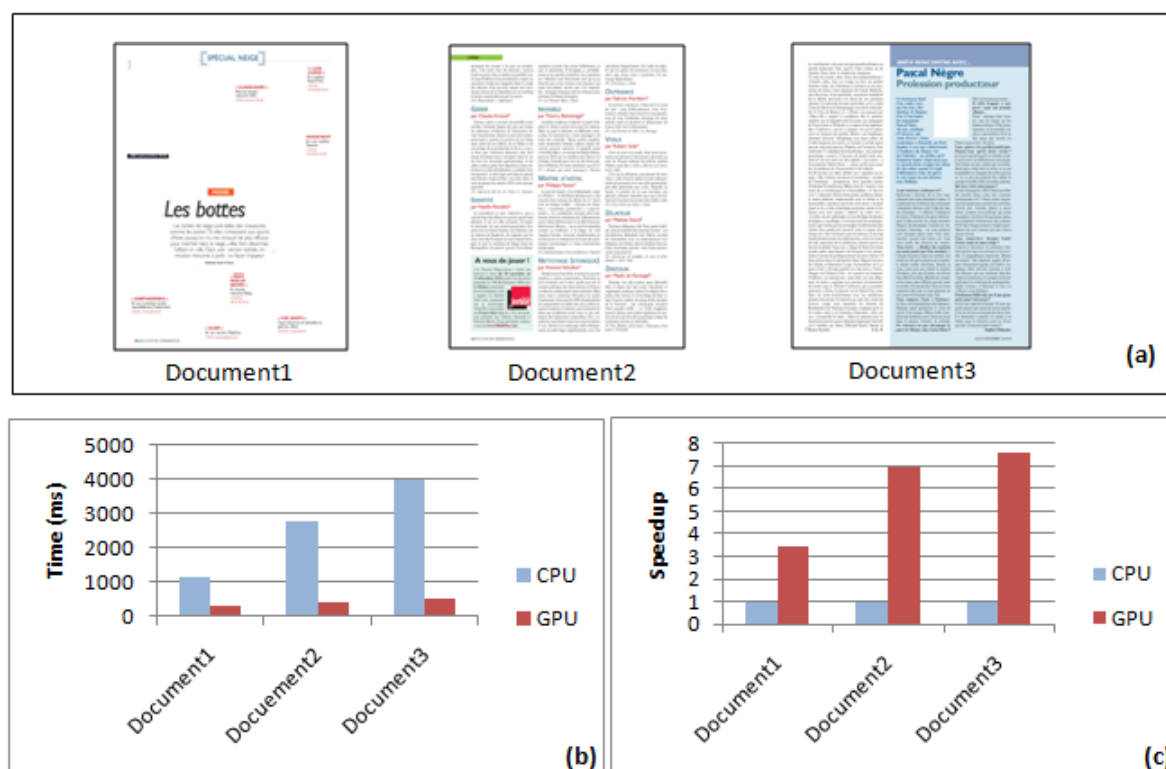


FIGURE 4.22: Évaluation du temps d'exécution de la méthode CHK sur GPU et CPU en variant le contenu. (a) : Document original scanné; (b) : temps d'exécution sur GPU et sur CPU; (c) : Accélération du GPU par rapport au CPU.

Dans la suite, nous comparons la méthode EHT avec des méthodes de binarisation de l'état de l'art associées avec la méthode d'analyse de structures de documents PLA (version GPU) [56]. Nous rappelons brièvement les méthodes de binarisation (Niblack [93], Sauvola [94] et *Sauvola* $M_{s_{kx}}$ [68]). Nous donnons également les détails d'implantations parallèles de ces méthodes sur GPU.

Niblack [93] est un algorithme de seuillage local. Il calcule un seuil pour chaque pixel en glissant une fenêtre rectangulaire sur l'image en niveaux de gris. La valeur de chaque seuil est calculée en se basant sur l'écart-type et la moyenne des pixels locaux à une fenêtre centrée autour du pixel à binariser. Ainsi, un seuil local $S_{Niblack}$ pour chaque pixel donné est calculé par la formule de l'équation 2.3.

Niblack a été parallélisé sur le GPU GeForce 9500 GT par B.Singh [115]. Il atteint une vitesse moyenne 20.84x plus rapide que sa vitesse sur CPU Intel Core 2 Duo de 2,66 Ghz. L'avantage de Niblack est qu'il identifie correctement les zones de texte. Cependant, il tend à produire une grande quantité de bruit dans les régions non-textuelles [114].

Sauvola [94] a proposé un algorithme similaire à Niblack [93]. Il propose une hypothèse supplémentaire sur la distribution des valeurs de pixels en niveau de gris associé au texte et au non-texte. Il suppose que les pixels associés au texte possèdent des valeurs proches du 0. Tandis que les pixels du non-texte possèdent des valeurs proches de 255. Le seuil Sauvola

TABLE 4.11: Temps de traitement moyen sur CPU et GPU de la méthode d'analyse de structure de documents pour une base de 125 documents hétérogènes de la base LRDE-DBD

Méthode	Plateforme	Temps (s)	Accélération
Analyse de structure de documents	CPU	2.5	-
	GPU	0.668	3.74x

TABLE 4.12: Évaluation du temps réel de la méthode EHT proposée

Méthodes parallélisées sur GPU	Temps (s)
Analyse de structure de documents	0.668
CG	0.132
CHK	0.425
Correction du rendu couleur	0.028
EHT	1.26

est calculé selon l'équation 2.4 (voir chapitre 2).

B.Singh a parallélisé Sauvola sur GPU [116]. Son implantation sur un GPU GeForce 9500 GT affiche une accélération moyenne de 20.8x par rapport à la même méthode sur CPU Intel Core 2 Duo de 2,66 Ghz.

Les deux versions de Niblack et Sauvola sur GPU emploient la même stratégie de parallélisation [116]. En effet, l'image d'entrée est stockée dans la mémoire texture. Ensuite, la taille des blocks et de la grid sont calculés en fonction des dimensions de l'image en entrée. Chaque thread calcule le seuil pour chaque pixel dans l'image de sortie.

La méthode *Sauvola_{MSkx}* [68] a été proposée pour améliorer celle de Sauvola dans le but d'extraire un texte multi-échelle. Elle est composée de quatre étapes. Tout d'abord, l'image est échantillonnée à différentes échelles. Ensuite, chaque image est binarisée selon le procédé de la méthode. Un seuil est fixé pour chaque objet selon l'appartenance de son échelle. Une image finale comprenant les seuils est produite et par la suite binarisée. Comme une mise en œuvre de la méthode *Sauvola_{MSkx}* n'est pas disponible sur GPU, nous avons parallélisé le calcul du seuil sur chaque échelle et nous avons gardé les autres étapes sur le CPU.

Généralement, le réglage des paramètres k et w pour ces trois méthodes nécessite une connaissance préalable de l'ensemble des documents. Pour notre évaluation, nous employons les paramètres recommandés par [68]. Nous ajustons $w = 51$ et $k = -0,2$ pour la méthode Niblack. Pour Sauvola et $Sauvola_{MSkx}$ nous utilisons $w = 51$ et $k = 0,34$. Le tableau 4.13 montre la comparaison des temps d'exécution entre les méthodes EHT et du reste des méthodes binarisations, à noter : Niblack, Sauvola, et $Sauvola_{MSkx}$, aux quelles nous avons ajouté le traitement d'analyse de structure de documents.

TABLE 4.13: Évaluation du temps réel sur GPU de la méthode EHT et trois méthodes de binarisation associées avec l'analyse structurelle de document sur la base LRDE-DBD

Méthodes parallélisées sur GPU	Temps de binarisation (s)	Temps de l'extraction du texte (Binarisation et Analyse de structure du document) (s)
EHT	0.425	1.26
Niblack et Analyse structurelle	0.594	1.262
Sauvola et Analyse structurelle	0.595	1.263
$Sauvola_{MSkx}$ et Analyse structurelle	1.660	2.328

Cette évaluation montre que la binarisation de EHT donne le meilleur temps d'exécution sur le GPU GTX 660. Notre méthode est 3.9x plus rapide que $Sauvola_{MSkx}$ et 1.4x plus rapide que Niblack et Sauvola. En effet, notre implantation est capable de binariser rapidement les pixels car toutes les variables nécessaires aux threads sont stockées dans les registres rapides en accès et dans les mémoires partagées. En revanche, Niblack et Sauvola ont besoin de calculer un seuil pour chaque pixel avant binarisation. Pour cela, ils ont besoin d'extraire un grand nombre de valeurs de pixels voisins à partir de la mémoire de texture afin de calculer la moyenne et l'écart type pour chaque pixel. Sinon, EHT emploie des mémoires plus rapides par rapport à $Sauvola_{MSkx}$ qui utilise encore la mémoire texture qui est lente en accès pour calculer les seuils de pixels sur plusieurs échelles. Par conséquent, EHT surpasse $Sauvola_{MSkx}$ sur le GPU.

Niblack et Sauvola affichent des temps d'exécution proches, cela est dû au flux de traitement similaire. Cependant, Sauvola est plus lent en raison de sa formule de calcul de seuil plus complexe. Finalement, en plus de la meilleure qualité qu'offre EHT par rapport au méthodes Niblack, Sauvola et $Sauvola_{MSkx}$, démontrée dans le chapitre 2, notre implantation de EHT sur GPU est la plus rapide. $Sauvola_{MSkx}$ donne le temps d'exécution le plus lent. En effet, le traitement multi-échelle implique un surcoût en temps de calcul. Dans la suite, nous évaluons la qualité de notre implantation EHT sur GPU par rapport à son implantation sur CPU pour montrer la fiabilité de notre mise en œuvre. En associant les binarisations de l'état de l'art avec le traitement d'analyse de structure de document, la méthode EHT affiche le

temps d'exécution le plus rapide 1.26 secondes (Temps affiché en gras dans le tableau 4.13)

Généralement l'architecture GPU a des contraintes au niveau de la mémoire et des règles d'exécutions pour assurer un temps d'exécution minimal. La figure 4.23, montre le résultat de binarisation de texte entre de la méthode EHT sur les deux plateformes CPU et GPU.

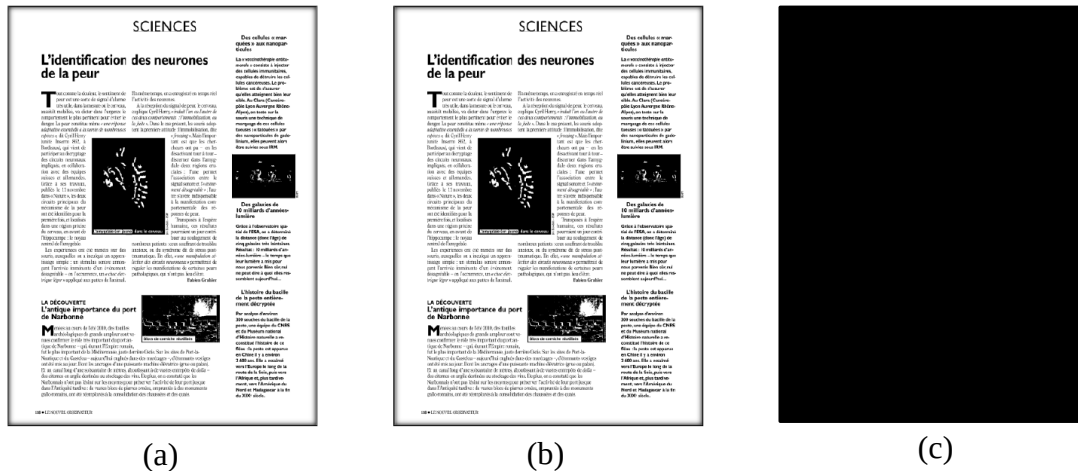


FIGURE 4.23: Comparaison de la binarisation de la méthode EHT sur un crop de document scanné : a. GPU (F-mesure = 98.5%) ; b. CPU (F-mesure = 98.5%) ; Image de différence.

Selon la figure 4.23, la version EHT sur GPU ne montre aucune perte de données de point de vue qualité de binarisation de texte par rapport à son résultat sur CPU. Pour prouver ce résultat visuel, nous calculons la précision F-mesure de EHT sur CPU et GPU. Les résultats obtenus montrent que les deux plateformes permettent d'assurer la même valeur F-mesure. De plus, la figure 4.23 (c) illustre l'image de différence des images (a) et (b). En effet, notre programme de GPU ne rencontre pas de limitation ou de conflit mémoire qui produit des résultats différents.

5 Conclusion

L'extraction de texte est un processus clé dans le système de reconnaissance de texte. Dans ce contexte, nous avons proposé une nouvelle méthode pour l'extraction hybride de texte (EHT). Elle engendre une haute précision de reconnaissance de caractères affichant 98.5% de taux de reconnaissance.

Nous avons parallélisé la méthode d'extraction hybride de texte (EHT) sur GPU. Notre implantation est basée sur la combinaison des stratégies parallèles à fins et à gros grains et

une gestion efficace de la mémoire. Aussi, nous avons minimisé les coûts de communication entre le CPU et le GPU grâce au traitement de l'algorithme Kmeans en totalité sur GPU. Finalement la contrainte temps réel de l'application de dématérialisation est satisfaite comme on peut le voir sur la figure 4.24. Le tableau 4.14 illustre les temps d'exécution obtenus

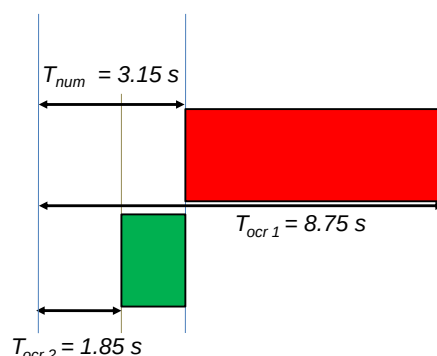


FIGURE 4.24: Traitement temps réel de l'OCR après accélération. T_{num} : Temps de numérisation du scanner, T_{ocr1} et T_{ocr2} : Temps de reconnaissance de caractères respectivement avant et après accélération.

concernant le système de reconnaissance CR-UCD, proposé dans le chapitre 3, avant et après l'accélération de la méthode EHT. Ainsi, nous avons accéléré la méthode EHT 6.74x par rapport à son traitement sur CPU. Ceci a permis d'accélérer 4.72x le temps de traitement du système CR-UCD.

TABLE 4.14: Évaluation de l'accélération de la méthode CR-UCD

Étapes de la méthode CR-UCD	Temps Initial (s)	Temps Après parallélisation sur GPU (s)	Accélération
Extraction hybride de caractères	8.16	1.26	6.74x
Description de caractères	0.36	-	-
Reconnaissance de caractères	0.23	-	-
Total	8.75	1.85	4.72x

Conclusion Générale

Le cadre de cette thèse est la dématérialisation temps réel de documents textuels, laquelle consiste à transformer les documents papiers en document numériques éditables en utilisant un système de reconnaissance optique de caractères OCR.

Généralement, la dématérialisation de documents textuels est constituée de deux étapes. D'abord les documents papier sont numérisés par un scanner. Ensuite, les documents numériques sont transférés en flots de données vers un ordinateur distant pour effectuer la reconnaissance optique de caractères.

A travers cette thèse, nous avons proposé un nouvel système de reconnaissance de caractères basée sur une description de caractères robuste qui assure une qualité de reconnaissance de 100% et ce pour des caractères multi-fonte et multi-échelle. Par ailleurs nous avons accéléré le traitement du système OCR proposé 4.72x en parallélisant l'étape de l'extraction hybride sur GPU.

La première partie de cette thèse a été consacrée à la présentation des systèmes de reconnaissance de caractères, notamment, ses étapes d'extraction de texte, de description et de reconnaissance de caractères. Nous nous sommes concentrés essentiellement sur les étapes d'extraction de texte et de description des caractères.

Nous avons étudié les approches d'extraction de texte dans différents types de documents qui incluent l'information textuelle. D'une part nous avons analysé l'extraction de texte dans les images de scènes naturelles où l'arrière plan du texte est souvent texturé et d'autre part l'extraction de texte dans les images de texte sur un arrière-plan homogène (régions de texte simples).

L'extraction de texte dans une image inclut les étapes d'amélioration pour augmenter le contraste entre le texte et l'arrière-plan, de binarisation pour séparer le texte de son arrière-plan, et finalement de localisation des caractères.

Sur les arrière-plans texturés, l'extraction de texte est une étape difficile car les pixels du texte et de l'arrière plan peuvent se mélanger. De plus, le faible contraste entre le texte et l'arrière-plan diminue la lisibilité des caractères. Dans ce contexte, nous avons relevé que les approches d'analyse statistique de textures sont mieux adaptées pour améliorer le contraste entre le texte et son arrière-plan complexe avant l'étape de binarisation. Le contraste est amélioré en utilisant des méthodes d'analyse statistiques de textures telle que la méthode de correction gamma (GCM) qui se base sur l'analyse des caractéristiques de l'énergie et du

contraste de l'image en utilisant les matrices de co-occurrences.

Généralement, on n'a pas besoin d'améliorer le contraste de l'image sur les régions de texte avec un fond homogène. Une simple méthode de binarisation globale suffit pour binariser l'image et faciliter la localisation des caractères. Ces images sont exposées à des dégradations externes liées à la numérisation, comme le bruit ou la non uniformité de la lumière affectant la qualité de l'extraction de texte, ce qui affecte la qualité de l'étape de binarisation. Dans ce cas les méthodes globales de binarisation sont sensibles à la non-uniformité de la lumière dans l'image. Pour remédier à ce problème, on emploie des approches de binarisation locales qui sont robustes face à cette non-uniformité de lumière. Cependant, celles-ci génèrent des artefacts qui dégradent ultérieurement la reconnaissance de caractères. Une approche hybride qui combine des méthodes globale et locale permet de gérer la non-uniformité et le bruit sans pour autant générer des artefacts.

L'existence de régions d'image et de texte simple dans un même document rend la tâche de l'extraction plus compliquée car on se retrouve généralement soit avec des méthodes destinées à extraire le texte des images, soit avec des méthodes destinées à extraire le texte des régions d'image simples. Une approche qui s'adapte aux types des régions dans le document hétérogène est primordiale. Une fois le document hétérogène binarisé. La localisation des caractères est réalisée en se basant sur l'analyse des composantes connexes.

Après l'extraction des caractères, ceux-ci sont décrits avec des attributs uniques pour assurer leur identification dans la dernière étape de la reconnaissance. La description des caractères basée sur les caractéristiques permet d'extraire des informations pertinentes à partir de l'image brute des caractères. Ainsi, un caractère peut être défini par : ses lignes, creux, courbures, segments, etc.

Il existe plusieurs approches pour décrire les caractères tels que les approches statistiques, topologiques, spatio-temporelles, structurelles, etc. Nous avons relevé que les méthodes de description structurelles sont parmi les méthodes les plus performantes face à la variation de la forme des caractères dans le document. Ce type de caractéristiques est extrait généralement après transformation de l'image par : la détection de contour, la transformation en squelette, etc.

Les caractéristiques structurelles sont choisies de façon à éviter la redondance. Dans le cas contraire, on augmente la complexité du calcul sans pour autant augmenter le degré de la discrimination entre les caractères. D'autre part les caractéristiques sont choisies de façon qu'ils soient le moins sensible que possible à la déformation de la taille et de la fonte.

1 Synthèse des contributions

Nous avons proposé une nouvelle méthode d'extraction hybride de texte dans des documents hétérogènes. C'est une méthode qui s'adapte aux types de régions dans l'image. Elle prend en considération les particularités locales de chaque région, et ce de manière complètement automatisée.

Dans l'extraction de texte proposée, une analyse de la structure du document est employée pour le séparer en régions d'images et de textes. Ensuite, une correction gamma basée sur l'analyse statistique de la texture est effectuée sur les régions d'images. L'information textuelle est extraite en utilisant une méthode de classification hybride basée sur l'algorithme Kmeans à partir des régions d'images et de textes simples. Cette méthode améliore la qualité de séparation texte/fond, tout en minimisant la quantité de distorsion dans les documents scannés. Nous avons extrait les caractères en une approche de dissection de caractères qui consiste à découper les lignes de texte en utilisant une marge d'espacement fixe. Malgré l'hétérogénéité et la complexité des documents notre méthode d'extraction de texte permet d'atteindre un taux de reconnaissance de caractères de 98,5% sur des documents hétérogènes scannés issues de base LRDE-DBD [103].

Ensuite, nous avons mis en place un nouveau descripteur de caractères qui emploie des caractéristiques structurales issues de l'unification de deux descripteurs dérivés de la projection horizontale et verticale des caractères, permettant une discrimination plus efficace. Nous avons effectué une détection des frontières situés sur les segments à droite et en bas des caractères. A partir des nouveaux segments détectés, nous avons calculé les caractéristiques : le nombre de segments, la position des segments, l'emplacement du barycentre du caractère, dans le rectangle englobant le caractère, et la proportion entre largeur et hauteur des caractères.

Le descripteur proposé, supporte la reconnaissance des caractères alphanumériques en multi-échelle et multi-fonte. Il permet d'atteindre un taux de reconnaissance de 100% pour une fonte et une taille données. A partir du descripteur proposé nous avons réalisé un système de reconnaissance de caractères (CR-UCD). L'évaluation que nous avons menée, donne un taux de reconnaissance 1.5x meilleure que celle du système Tesseract [43] sans utilisation de correction linguistique.

Dans le cadre d'une application de dématérialisation, nous avons accéléré l'étape de l'extraction hybride de texte proposée selon des stratégies de parallélisation en utilisant les architectures de (CPU) Multi-cœurs et d'accélérateurs graphiques (GPU). Pour paralléliser nos algorithmes, nous avons choisi une stratégie de parallélisation hybride entre ces deux architectures car les GPU sont efficaces pour le calcul massivement parallèles et les CPU sont rapides sur les parties séquentielles des algorithmes.

Nous avons proposé une stratégie de parallélisation de l'extraction hybride de texte, étape la plus coûteuse en temps de calcul dans le système CR-UCD proposé. Notre mise en œuvre, combine les stratégies de parallélisation à fin et à gros grains. Plusieurs algorithmes ont été accélérés tel que le Kmeans [66] pour la classification, Otsu [65] pour le calcul du seuil, et Haralick [63] pour le calcul des caractéristiques. En outre, les coûts de communication CPU-GPU sont réduits et une bonne gestion mémoire est assurée. Finalement, l'extraction hybride de texte a été accélérée 6.64x sur GPU par rapport à son exécution sur CPU ce qui permet d'accélérer 4.72x le système de reconnaissance de caractères CR-UCD conçu.

2 Perspectives

Il existe plusieurs pistes de recherche qui améliorent les méthodes présentées dans cette thèse et auxquelles nous nous sommes intéressées.

Pour la méthode EHT proposée, les caractères localisés par la méthode de dissection, basée sur des marges fixes, peut produire deux problèmes. Les caractères localisés peuvent être collés ou fragmentés. Pour résoudre ce problème on peut relocaliser les caractères au cas où le taux de reconnaissance de caractères n'est pas de qualité. Dans ce cas, une dissection selon des marges variées ou bien une fusion des caractères sera effectuée itérativement et conjointement avec les étapes de description et de reconnaissance jusqu'à avoir un taux de reconnaissance élevé.

Pour améliorer un résultat de reconnaissance, jugé non satisfaisant, d'un caractère donné, nous ajouterons un post-traitement pour vérifier et corriger la reconnaissance du caractère. Ainsi, on utilisera une correction linguistique qui se base sur la vérification du mot englobant le caractère en question en utilisant un dictionnaire de mots préalablement construit. Ce traitement résoudra le problème de la ressemblance de la forme des caractères tel que « I » et « l » dans la fonte « Arial ».

La chaîne de reconnaissance de caractères développée permet de reconnaître efficacement du texte imprimé présent dans un document hétérogène. Aussi, cette chaîne pourra être utilisée sur des documents homogènes contenant uniquement du texte. Dans ce cas, la correction gamma n'est pas effectuée et le document est directement binarisé par la méthode de classification hybride basée sur l'algorithme Kmeans. Pour appliquer la chaîne uniquement sur des images tel que les scènes naturelles, la correction gamma sera réalisée avant l'étape de la binarisation. Cependant à cause de la grande distorsion des caractères présents dans ce type de documents (perspective, inclinaison, éclairage, etc.) et la variation de la fonte (fonte calligraphiques), il faudra corriger préalablement les distorsions du texte avant la phase de description. Ensuite il faudra adapter la description pour de caractères très variée en fontes, en utilisant une approche de type Deep Learning.

Il est possible de paralléliser davantage la chaîne CR-UCD grâce à la grande indépendance de données qu'offre l'algorithme de correction gamma. En effet, le calcul des caractéristiques de Haralick et du seuil Otsu sont réalisés sur Nb_{images} exemplaires de l'image originale transformés avec une valeur gamma allant de 0.1 à 10.0 avec un pas de 0.1. Ainsi, le calcul parallèle des caractéristiques de Haralick et du seuil Otsu que nous avons réalisé pourra être réalisé parallèlement entre les Nb_{images} images.

L'utilisation de nouvelles architectures GPU plus récentes tel que l'architecture « Pascal » permettra d'élaborer de nouvelles stratégies de parallélisation plus performantes, grâce à sa technologie évoluée. Par exemple, la technologie NVLink permettra d'interconnecter les GPUs du nœud de calcul, sur lequel nous avons parallélisé la correction gamma, directement sans passer par le CPU. Ainsi, on offre un accès mémoire plus rapide, une bande passante plus large.

L'accélération de la méthode EHT a permis l'intégration de la chaîne CR-UCD dans l'application de dématérialisation qui concerne les scanners de vitesse de numérisation jusqu'à 32 ppm (pages par minutes). Pour un scanner plus rapide, il faudra accélérer davantage le traitement et plus précisément l'étape d'analyse de structure de document employé dans la méthode EHT.

Annexe

Annexe A

GPU Parallel implementation of the new Hybrid Binarization based-Kmeans method (HBK)

M. Soua, R. Kachouri, M. Akil.
Journal of Real-Time Image Processing, pp 1-15, 2014

Mahmoud Soua · Rostom Kachouri · Mohamed Akil

GPU Parallel implementation of the new Hybrid Binarization based-Kmeans method (HBK)

Received: date / Revised: date

Abstract The Optical Character Recognition (OCR) is a process that converts characters within images into text documents. In paperless applications, OCR systems have to ensure a better accuracy as well as a high speed. One of the most important steps in OCR is binarization. In this context, we proposed recently the Hybrid Binarization based-Kmeans method (HBK) [1]. HBK offers a satisfying recognition rate while scoring 91% accuracy. In the other hand, running on an Intel Core i3 CPU processor, the HBK requires at least 1.9 seconds to process one A4 300 dpi document. However, binarization step should not exceed 460 ms in our real time OCR system. For this, we propose in this paper a parallel implementation of the HBK method on the NVIDIA GTX 660 Graphic Processing Unit (GPU). Our implementation, combines fine-grained and coarse-grained parallelism strategies for the best GPU use. In addition, the costly CPU-GPU communication overhead is avoided and an efficient memory management is ensured. The effectiveness of our implementation is validated through extensive experiments, which demonstrate that the proposed HBK parallelization accelerates the studied process. Indeed, we ensure the binarization of one document in just 425 ms. Consequently, the implemented design is able to meet the targeted real time OCR system in paperless application.

Keywords OCR · Scanned documents · Binarization · HBK · Kmeans · CUDA · GPU

1 Introduction

Optical Character Recognition (OCR) [2] is the process that recognizes numerical characters on printed pages

M. Soua
ESIEE Paris, IGM, A3SI, 2 Bd Blaise Pascal, BP 99, 93162
Noisy-Le-Grand, France.
E-mail: mahmoud.soua@esiee.fr

R. Kachouri E-mail: rostom.kachouri@esiee.fr ·
M. Akil E-mail: mohamed.akil@esiee.fr

and converts them to a machine readable text file. The OCR quality is measured according to the character recognition rate [3]. Generally, the acceptance rate varies according to the business operation or recognition task. For a common A4 document, scanned at 300 dpi, error rates are often in the range of 1% to 10% of all characters in page [4]. Actually, both accuracy and high speed of OCR are required in the real time applications [5]. For example, in the banking check reading [5], the acceptable error rate is close to zero and the OCR should be very fast. It may have a very high throughput of up to 150.000 documents per hour [5]. Also, in the mail sorting [6], the acceptable rate is close to zero. The recognition rate, is about 30.000 letters per hour [5]. In our work, we focus on the paperless [7,8] real time application, in which, a high speed scanner Epson Perfection V100 Photo scans over one A4 300 dpi document per 23 second. This application is depicted in Figure 1. Both binarization and recognition are required in the OCR processing [9]. Binarization plays a very important role in the OCR toolchain. In this operation, documents are converted into bi-level representation distinguishing the foreground from the background. We consider that foreground objects represent printed text, a legend, or a drawing while the complementary objects correspond to the background. Indeed, a high accuracy in extracting untouchable and unfragmented characters is very important to perform efficient recognition.

Actually, we are interested in two points: enhancing the binarization quality to improve the recognition rate and speedup the execution time to increase the number of documents processed per second.

On the first point, we proposed recently an adaptive new method called Hybrid Binarization based on Kmeans (HBK) [1], basically designed for scanned documents. Firstly, the HBK method performs the Kmeans clustering algorithm on local areas seeking more binarization robustness. Then, it gathers local features in a global manner to improve the local binarization quality. Basically, the HBK method outperforms state of the art binarization methods such as SauvolaMS_{kx} [10] and

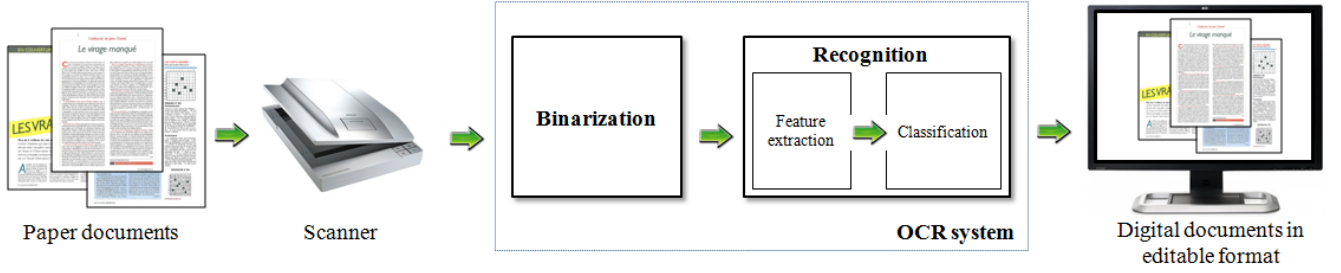


Fig. 1 Paperless application toolchain based on OCR system

Niblack [11]. It scores a character recognition rate of 91% when performing on 125 scanned magazine documents [1].

On the second point, we consider our paperless application time constraint. Given that the binarization step should take, generally, 2% from the total OCR execution time [13], the processing time required to binarize one document is 460 ms in our real time system. According to our experiments, the HBK method requires at least 1.9 seconds to process one A4 300 dpi document. Thus, it does not meet the real time constraint of our paperless application. Actually, the running time of Kmeans in the HBK method is time consuming [14]. Therefore, parallelizing it is a promising approach to overcome the challenge of the huge computational requirement [14] and accelerate HBK. In this context, different implementations of Kmeans on the Graphic Processor Unit (GPU) have been proposed [15,16,17,14,18]. Those papers mainly address Kmeans as general purpose clustering algorithm. According to our knowledge, the first popular Kmeans GPU implementation is the UV_Kmeans [15]. It achieves a speedup of 10x to 40x as compared to four-threaded Minebench [20] running on a dual-core, hyper-threaded CPU. The GPU_Miner [16] improves 5x UV_Kmeans [15] by decreasing the transfer memory latency. HP_Kmeans [19] claims another speedup of 2x to 4x as compared with UV_Kmeans [15] and 20x to 70x speedup as compared with GPU_Miner [16]. Y.Li implementation [14] is 4x to 8x faster than UV_Kmeans [15] when performing on an NVIDIA GTX 280 GPU.

In this paper, we propose an efficient parallel implementation of the HBK binarization method. Our approach is established through two levels of parallelism. The first level is the coarse-grained parallelism when parallelizing the Kmeans algorithm between thread blocks. The second level is the fine-grained parallelism, in which, we parallelize the distance computation and the labeling Kmeans stages between threads. Our HBK implementation ensures an efficient memory management and limits the CPU-GPU memory transfer overheads.

The rest of the paper is organized as follow. In Sect.2 we explain the HBK binarization method and the related quality and real time evaluations. Sect. 3 provides a brief presentation of the GPU architecture. Next, in Sect. 4,

we review the related works that parallelized Kmeans on GPU and we describe our proposed HBK GPU implementation. Then, experimental results are discussed in Sect. 5. Finally, conclusion is drawn in Sect.6.

2 Hybrid Binarization based on Kmeans (HBK)

The HBK method combines local and global approaches to ensure robust binarization. Indeed, it is well adapted for noisy documents thanks to the local approach. Moreover, the global process improves the binarization quality when merging the local binarization features. Figure 2 shows the HBK method and more details are described in Algorithm 1.

To binarize documents, two initial centroids are employed. We note by $C_G = \{C_{G0}, C_{G1}\}$ the global centroid set. They are initialized with black and white values. Thus, $C_{G0} = (0, 0, 0)^T$ and $C_{G1} = (255, 255, 255)^T$ design respectively the background and the foreground clusters. The input document is divided into Nb blocks. The size of the blocks can be small, medium or large depending on the size of characters [1]. Moreover, each block includes two local centroids given by the C_L set:

$$C_L = \{(C_{L0,0}, C_{L0,1}), (C_{L1,0}, C_{L1,1}), \dots, (C_{Lbl,i}, \dots, (C_{L(Nb-1),0}, C_{L(Nb-1),1})\}$$

with $C_{Lbl,i} = (C_{Lbl,i}^R, C_{Lbl,i}^G, C_{Lbl,i}^B)^T$, $i \in [0, 1]$, $bl \in [0, Nb - 1]$. Furthermore, each block includes two local accumulators and counters given respectively by the sets $Accu_L$ and $Count_L$:

$$Accu_L = \{(Accu_{L0,0}, Accu_{L0,1}), (Accu_{L1,0}, Accu_{L1,1}), \dots, (Accu_{Lbl,i}, \dots, (Accu_{L(Nb-1),0}, Accu_{L(Nb-1),1})\},$$

$$Count_L = \{(Count_{L0,0}, Count_{L0,1}), (Count_{L1,0}, Count_{L1,1}), \dots, (Count_{Lbl,i}, \dots, (Count_{L(Nb-1),0}, Count_{L(Nb-1),1})\}$$

with $Accu_{Lbl,i} = (Accu_{Lbl,i}^R, Accu_{Lbl,i}^G, Accu_{Lbl,i}^B)^T$, where, $i \in [0, 1]$ and $bl \in [0, Nb - 1]$. In the beginning, all local centroids are initialized with C_{G0} and C_{G1} . Then,

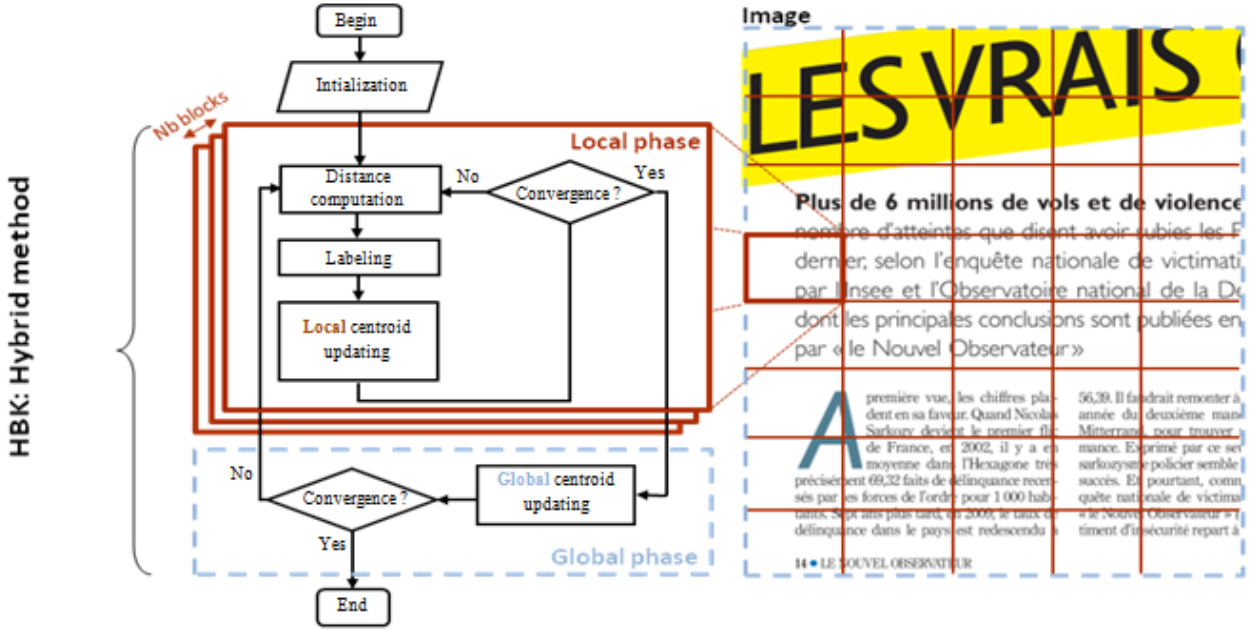


Fig. 2 The HBK binarization method

Algorithm 1 HBK(Img)

C_G : Global centroids
 ite : Current iteration index
 bl : Block index
 Nb : Total number of blocks
 $Accu_{Gi}$: Global accumulator
 $Count_{Gi}$: Global Counter
 $Accu_{Lbl,i}$: Local accumulator
 $Count_{Lbl,i}$: Local Counter

```

// Global centroid initialization
1:  $C_{G0} \leftarrow (0, 0, 0)^T$ ,  $C_{G1} \leftarrow (255, 255, 255)^T$ 
// Repeat until convergence
2: while  $C_G(ite) \neq C_G(ite - 1)$  do
// Local Kmeans clustering on each block
3:   for  $bl \in [0, Nb - 1]$  do
4:      $Kmeans(bl, C_G)$ 
5:     for  $i \in [0, 1]$  do
6:        $Accu_{Gi} \leftarrow Accu_{Gi} + Accu_{Lbl,i}$ 
7:        $Count_{Gi} \leftarrow Count_{Gi} + Count_{Lbl,i}$ 
8:     end for
9:   end for
// New centroid computation
10:   $C_{Gi} \leftarrow \frac{Accu_{Gi}}{Count_{Gi}}$ ,  $i \in [0, 1]$ 
11: end while
  
```

the Kmeans clustering algorithm [21] is applied across all blocks as stated in lines 3 and 4. When it converges, the local accumulators $Accu_L$ are gathered into global ones defined by $Accu_G = \{Accu_{G0}, Accu_{G1}\}$, $Accu_{Gi} = (Accu_{Gi}^R, Accu_{Gi}^G, Accu_{Gi}^B)^T$ with $i \in [0, 1]$ (line 6). This operation is based on one addition per each color component. Similarly, local counters are accumulated into the global ones given by $Count_G = \{Count_{G0}, Count_{G1}\}$ (line 7). Finally, the global centroids C_G are computed

according to $Accu_G$ and $Count_G$ division (line 10). The algorithm stops when the global convergence is reached, else it reiterates while local centroids C_L are reset with C_{G0} and C_{G1} . Following, we show the robust binarization quality of the HBK method through an OCR evaluation.

2.1 HBK OCR-based Evaluation

The Optical Character Recognition (OCR) is a process by which text characters contained in an image can be recognized. The binarization quality has a direct influence on the OCR result. Commonly, the computer uses an OCR Engine to recognize characters in an image. In our evaluation, we employ the well-known Tesseract 3.02 engine [22]. The OCR is performed directly on binarized scanned documents. For this we use the LRDE-DBD¹ dataset [23]. It is composed of 125 scanned and non-scanned documents extracted from "Le Nouvel Observateur"² magazine. These documents have A4 format and 300-dpi resolution. According to HBK, the block size of 32x32 guarantees a robust binarization quality [1]. Table 1 shows the OCR recognition rate including the HBK and several binarization methods.

The OCR results are close: The recent Sauvola Ms_{kx} [10] gives an acceptable OCR rate by scoring 89% of

¹ Copyright(c) 2012. EPITA and Development Laboratory (LRDE) with permission from Le Nouvel Observateur. LRDE-DBD is available online on the web site: <http://www.lrde.epita.fr/cgi-bin/twiki/view/01ena/DatasetDBD>

² Le Nouvel Observateur. Issue 2402, November 18-24, 2010 and available on the website: <http://tempsreel.nouvelobs.com>

Table 1 OCR accuracy evaluation of HBK and well-known binarization methods.

Methods	Accuracy (%)
HBK [1]	91
SauvolaMS _{kx} [10]	89
Lelore [24]	85
Niblack [11]	80
TMMS [25]	73

OCR accuracy. It ensures a robust text binarization however in some document areas, artifacts may appear leading to character miss-recognition. The HBK method gives the best OCR rate scoring 91% of accuracy thanks to its robust binarization results. Indeed, the local approach gives robust binarization quality and the global approach clears the artifacts generated by the local one. The main issue encountered by outperformed methods is related to large objects and the fact that parameters do not fit to the contents. The HBK approach succeeds in improving the SauvolaMS_{kx} algorithm because text with colored background is correctly retrieved. Indeed, according to our experiments, it ensures an F-Measure [26] of 98% while SauvolaMS_{kx} reaches only 95% [10], with F-Measure is a metric that evaluates the pixel-based accuracy. The TMMS [25] method gives degraded results despite the efficient binarization of the text edges. This is due to the uneven illumination introduced by the scanned documents.

In this work, we aim to perform a binarization that can be integrated in a real time OCR application. In the next subsection, we check the HBK ability to satisfy the real time constraint of our paperless application.

2.2 HBK Real Time Evaluation

The experiments were conducted on an Intel Core i3 CPU performing at 3.07 GHz with a memory of 4 GB. Evaluations were performed using an average of 5 subsequent executions. In our target paperless application, the Epson Perfection V100 Photo scanner digitizes one A4 300 dpi document per 23 seconds. Actually, when the scanner digitizes a document, the OCR is performed simultaneously on the previous scanned document. Hence, the OCR time constraint is equal to the scan time (23 seconds). The HBK method binarizes one document per 1.9 seconds. Unfortunately, this performance does not meet to real time constraint, because it exceeds 2% [13] from the total OCR execution time required. Following, we profile the HBK method to overview which processed parts are the most time consuming. Three parts were profiled: Initialization, Local phase and Global phase. The initialization includes memory allocations and data initialization. The local phase process includes the running time of Kmeans in all blocks of the document. The global phase process includes the check of the global

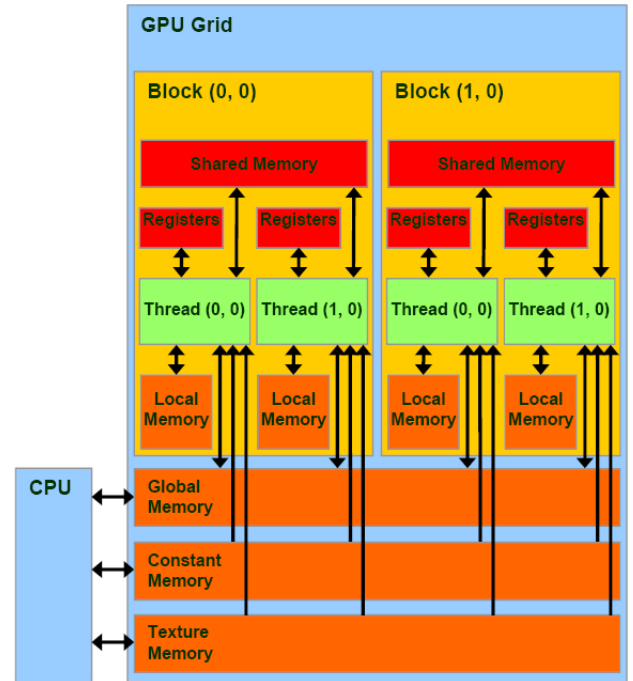
Table 2 Profiling of the HBK stages

Processes	Time (%)
Initialization	0.001
Local phase (Kmeans)	96.593
Global phase	0.013

convergence and the centroid updating. According to table 2, the overall HBK time execution is bounded with the Kmeans one. Indeed, reducing the Kmeans processing time will naturally decrease the HBK one. According to [18], the Graphic Processing Unit (GPU) develops continuously and provides a promising platform for parallelizing K-means [14]. Therefore, the GPU is an adequate alternative to accelerate Kmeans, and thus HBK. In the next section, we present the GPU architecture.

3 GPU Architecture

GPU is a plenty powerful many-core processor that support parallel data processing and high precision computation. It includes a set of Single Instruction Multiple Data (SIMD) Streaming Multiprocessors (SM). The SMs perform simultaneously the same operation on multiple data. Each SM is composed of several processors (cores). GPUs include a texture, constant, global, local, shared and register memories as shown in Figure 3 [27].

**Fig. 3** GPU Architecture in CUDA environment

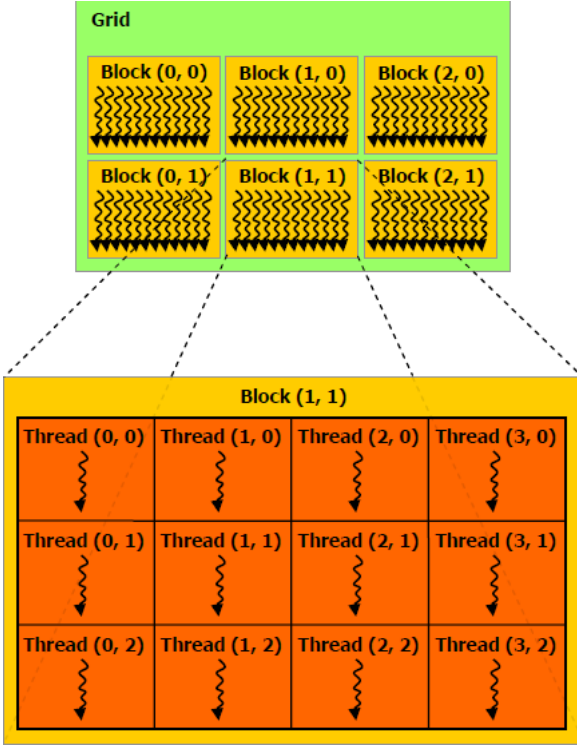


Fig. 4 GPU Execution model in CUDA environment

GPU grid, blocks and threads are explained a bit later. GPUs have dedicated texture hardware optimizing the 2D spatial locality. In addition, GPUs include a constant memory. It is a fast memory readable by all GPU cores. We note that, constant and texture memories are read-only which constitutes a major drawback if we need to write data. The main memory in GPUs is the global one. It can be accessed from all device cores. Depending on the GPU model, it can accommodate one Giga Bytes (GB) or more which let process large amount of pixels. The global memory is optimized to simultaneously handle a lot of memory requests providing a considerable large bandwidth. The access to this memory is cached which makes it very fast [28]. Besides, local memory is an abstraction of global one. It is accessed only by thread blocks. Moreover, it resides off chip, which makes it as expensive to access. The compiler makes use of local memory when it determines that there is not enough register space to hold variables. In the other side, the GPU includes shared memory which is 100x faster than global one. It can be used as a user-managed cache L1 to reduce the number of slow global memory accesses. This kind of memory is common to a group of cores and can be concurrently accessed by several ones. In such case, cores may collaborate in a given task, or simply store temporally data and reduce then register pressure. Eventually, the fastest memory access in GPU is provided by registers. Unfortunately, it has small memory storage. In the case of Fermi architecture, there are about two

Mega Bytes (MB), while current CPUs only have few Kilo Bytes (KB).

To exploit the GPU platform, it is possible to use multiple technologies. The most popular ones are the Open Computing Language (OpenCL) [29] and the Compute Unified Device Architecture (CUDA) [30]. In one side, OpenCL is an open standard offering cross-platform capabilities. The most important drawback of OpenCL is the high complexity of the Application Programming Interface (API) and the execution model [31]. In the other side, CUDA is an API that allows high C and C++ programming level on NVIDIA GPUs. In addition, the CUDA environment is more user friendly and able to port the programming code much more consistently. As shown in Figure 3, CUDA notes the SMs and the cores respectively by blocks and threads. The container of all CUDA blocks is called grid. According to the CUDA environment, a virtual architecture is considered defining an execution model as described in Figure 4 [28].

The execution model is then well suitable for parallelizing treatments in image processing because blocks group thousands of independent threads ready to process a huge number of pixels in parallel. In the next section, we explain our HBK GPU parallelization.

4 HBK Parallelization on GPU

The previous analysis, in section 2.2 shows that the HBK execution time is bounded with the Kmeans clustering one. For this, we aim to accelerate Kmeans. Following, we introduce the related Kmeans GPU acceleration methods. Then, we detail our proposed HBK implementation strategy based on our Kmeans GPU version.

4.1 Related Works

Following, we give an overview about existing Kmeans implementations on the GPU. The first fast and very popular Kmeans GPU implementation is the UV_Kmeans [15]. In this work, centroids and input data are stored respectively in the constant and texture memories. Each thread processes a single data point and the new cluster centroids are computed in the CPU device. In the UV-Kmeans implementation, the accumulation operations, are done on the CPU. However it increases the memory latency when transferring data between CPU and GPU. In addition, faster memories such as registers and shared memory are not used. W. Fang [16] accelerates 5x UV-Kmeans [15] by minimizing memory latency transfer between CPU and GPU. Indeed, he uses a bitmap grid to determine, inside GPU, the number of data points belonging to each cluster. This technique avoids large amount of atomic additions. However, the grid bitmap uses a lot of memory when the centroid number is high. Moreover, the memory management is not optimal. Indeed, registers and shared memory, are not employed.

Recently, works [17,14] perform more Kmeans steps on the GPU. For example in [17], authors use the CPU only for centroid initialization and the control flow. Also, in [14], several reduction steps are done on GPU before performing the centroid update stage on the CPU. This method, improves the works [15] and [16] by optimizing the memory management. Actually, the input data is stored in registers that allows very fast memory access. Y.Li [14] designs an algorithm that exploits GPU on-chip fast memories to significantly decrease the data access latency.

Some works perform the centroid updating on the CPU [17,18]. This usage is due to the employment of GPU architecture that are unable to achieve atomic floating point operations. For example, in the HP_Kmeans [19] method, the centroid updating step is performed on the CPU, but some accelerations are achieved like asynchronous transfer and cuda streams. In [18], researchers have implemented the Kmeans algorithm in the context of image segmentation. They parallelize only the labeling phase. However, the updating of centroids is performed on the CPU. Indeed, they employ a small centroid number compared to pixel number [18]. Hence the data transfer time between CPU and GPU is masked by the huge distance computation task.

In our work, we perform an efficient HBK GPU implementation based on a powerful Kmeans algorithm. Actually, HBK processes documents according to small blocks. Then, we apply Kmeans independently on each block. Following section explains our HBK GPU implementation.

4.2 HBK GPU Implementation

In our implementation, we perform *the coarse and the fine grained levels of parallelism*: In the coarse-grained level, we parallelize the document block treatment between blocks. In the fine-grained level, we parallelize the pixel processing inside blocks [28]. Figure 5 shows this concept.

Actually, in our HBK implementation, we parallelize Kmeans between the GPU blocks. In the other hand, we parallelize the distance computation and the labeling stages between threads (Figure 6). For this aim, we design two CUDA kernels: The Local and the Global Kernels. They handle respectively the local and the global HBK phases (Figure 2). All of the centroids C_G , accumulators $Accu_G$ and counters $Count_G$ are allocated in the global memory. After the CPU fetches image into the global memory, it launches the Local Kernel on the GPU device. Thus, the Kmeans algorithm is performed on each block, in parallel.

An *efficient memory management* is ensured by employing fast GPU memories. We limit the global memory utilization by dispatching the data to shared memory and registers. Indeed, the local centroids C_L , accumulators $Accu_L$ and counters $Count_L$ of each block

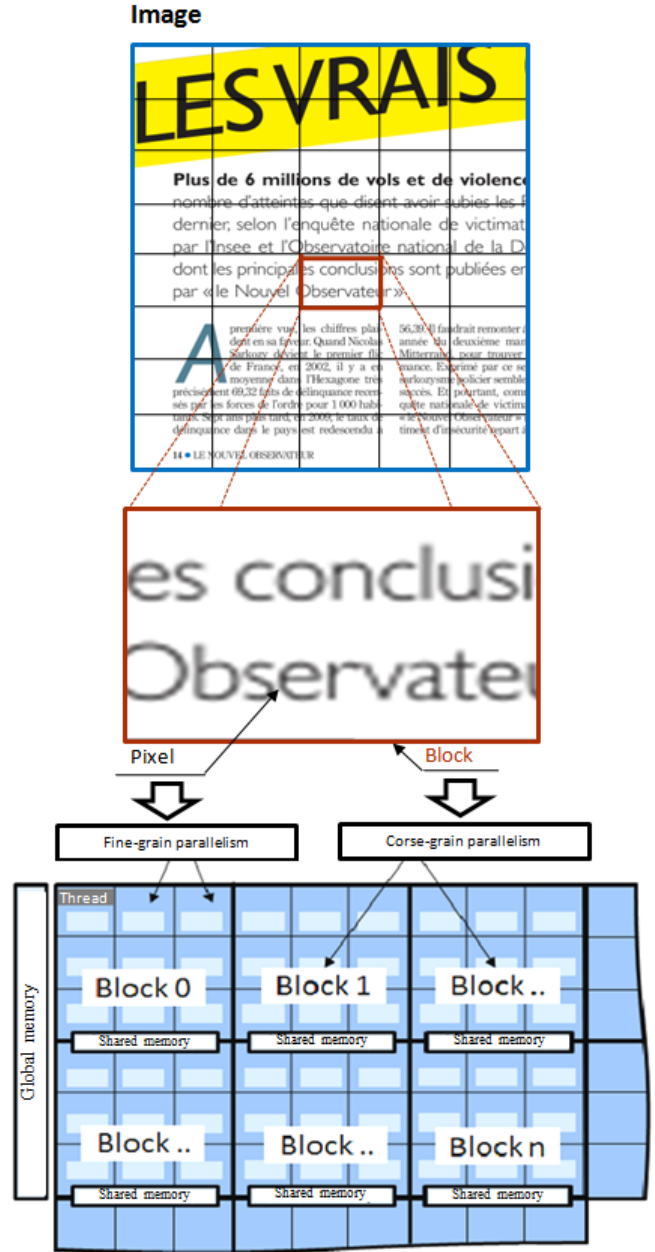


Fig. 5 Fine-grain and coarse grain parallelism levels in the HBK GPU implementation

are allocated in the shared memory for fast access to block threads. In addition, pixels are transferred from the global memory to registers which allows a very fast memory access during the program execution.

In the Kmeans initialization step, each two local centroids C_L are initialized with the RGB values $(0,0,0)^T$ and $(255,255,255)^T$. We note that, the distance computation between centroids and pixels is performed independently between pixels. Therefore, we process it in parallel between w threads. Then, the complexity decreases from $O(Npxk)$ to $O(Npxk/w)$ with Np is the

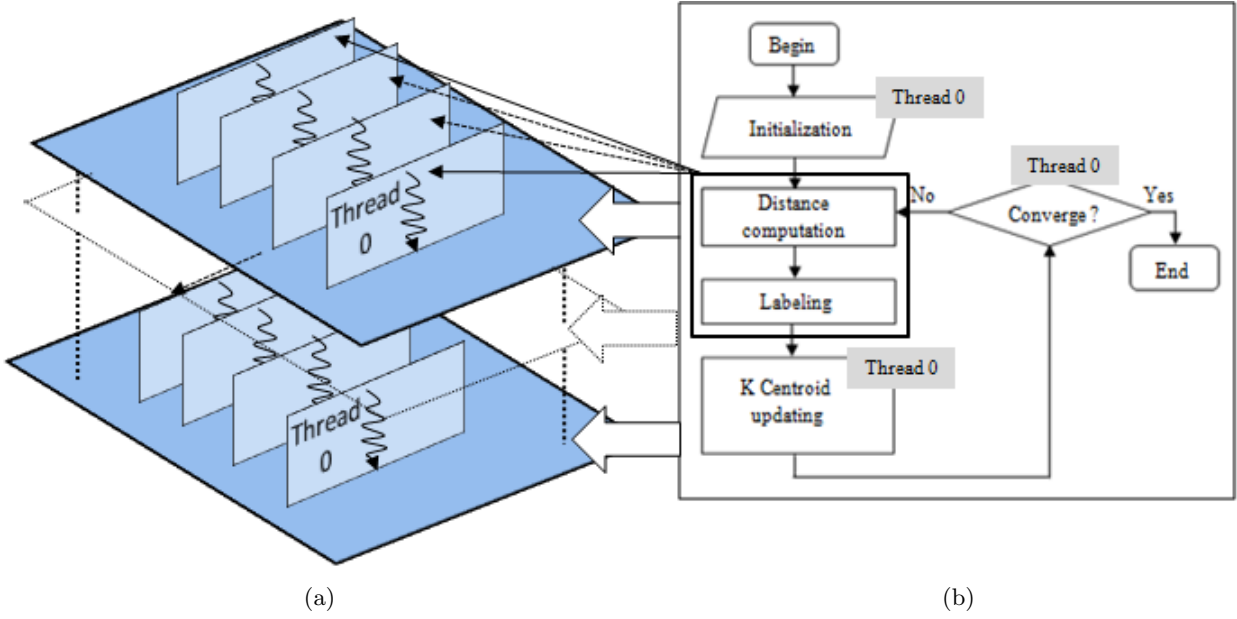


Fig. 6 Kmeans implementation strategies on GPU. a. The GPU thread blocks (multiprocessors) b. The Kmeans clustering algorithm

total number of pixels in the scanned document and k is the centroid number in each block. We should carefully map threads and pixels depends on their positions in the image. Moreover the thread number should match the total number of pixels to ensure a reliable binarization. Similarly, the labeling step of Kmeans, where each pixel is assigned to its cluster, is parallelized on w threads.

In our HBK implementation, we decrease the CPU-GPU communication overheads by performing a fully Kmeans in the GPU. Indeed, the centroid updating is performed on the GPU device. One master thread in each block perform this operation. Actually, the scanned document contains high number of blocks because their sizes are relatively small compared to the image one. This way, the total number of master threads in the whole GPU remains high. Then, GPU resources are efficiently exploited. In addition, updating centroids on the GPU is possible thanks to the availability of integer and floating-point atomic addition on recent GPU architectures like Fermi and Kepler [32].

During the labeling stage of the Kmeans algorithm, a reduction is performed on each block. Thus, local features $Accu_L$ and $Count_L$ are computed before being gathered into the global ones: $Accu_G$ and $Count_G$, as seen in Figure 7. Indeed, two steps are made. Firstly, local features are gathered using the shared atomic addition within the threads blocks. Secondly, all local features are added to the global ones using the global atomic additions. This technique decreases the global atomic operation overhead. Indeed atomic operations on global memory can be very costly, as they need to serialize a potentially large number of threads in the kernel. To reduce

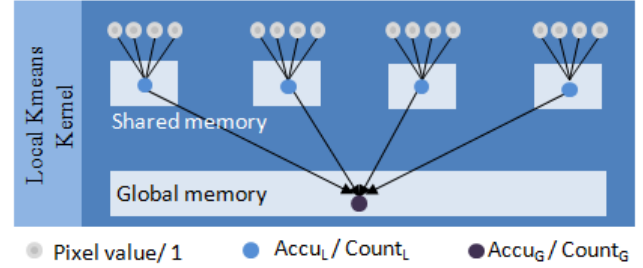


Fig. 7 Optimization of accumulator and counter variables computation

this overhead, we usually addition operations first to variables declared in the shared memory of each thread block [33]. Every block process stops when centroids of the current iteration are equal to the previous iteration ones. This means that the convergence criterion of Kmeans is reached. At that time, the CPU launches the Global Kernel. We do not need to transfer the accumulators $Accu_G$ and the counters $Count_G$ to the CPU because we keep them on the global memory. Based on these features, the Global Kernel computes centroids C_G . Similarly to the local convergence, global centroids are submitted to the convergence test. If the centroids of the current global iteration are equal to those of the previous one, the global convergence is reached. In this case, a flag is set to 1, otherwise it is set to 0. Subsequently, the CPU checks whether the computed flag is equal to 1 value to decide ending the program as shown in Figure 8. The transfer of the flag variable maintains a low CPU-GPU overhead which enhances the HBK performances.

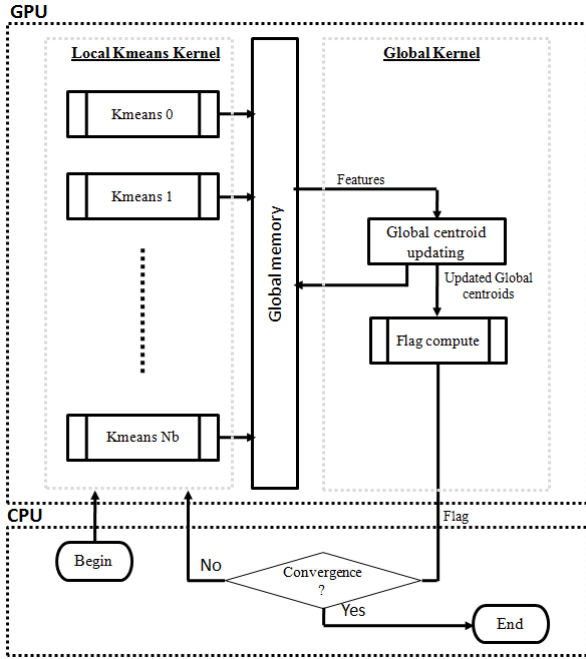


Fig. 8 The HBK implementation workflow

In the next section, we evaluate our HBK GPU implementation.

5 Experimental Results

In previous work [1], we have demonstrated that the HBK method is effective in binarizing scanned documents. In this section, we evaluate the execution time of HBK. After presenting the employed materials, we discuss the efficiency of our GPU-based Kmeans implementation. Then, we compare the HBK processing time between GPU and CPU to show the GPU acceleration. For this, several experiments were performed by varying parameters such as; block sizes, amount of noise (Scanned, NScanned documents) and document content (text areas, colored background). Finally, we evaluate the HBK binarization quality on CPU and GPU to show our implementation reliability.

5.1 Materials

In this evaluation, both scanned and non-scanned LRDE-DBD dataset [23] are employed. In the rest of the paper, we note the non-scanned documents by "NScanned". Our experiments were conducted on PC with an Intel Core i3 CPU performing at 3.07 GHz. In addition, we employ an NVIDIA GeForce GTX 660 GPU. This one has 5 SIMD multiprocessors. Each one contains 192 processors and performs at 980 MHz. In addition, every multiprocessor has 64 KB of configurable RAM including 16

KB of shared memory and 48 KB of L1 cache. The global memory has 2 GB with a peak bandwidth of 144.7 GB/s. Moreover, CUDA version 5.0 is used. In order to visualize clearly the speedup effect, the HBK execution time is computed after input and output data transfer between CPU and GPU. Before we evaluate our HBK GPU implementation, we study our fully GPU-based Kmeans. We name this implementation Kmeans_SD because it handles small datasets. Then, we adapt our Kmeans parallelization strategy to handle large datasets.

5.2 GPU-based Kmeans Time Evaluation

The Kmeans acceleration has been widely discussed in the literature [14]. Table 3 shows a comparative study between several GPU based Kmeans implementations such as; Kmeans_Li [14], HP_Kmeans [19], UV_Kmeans [15] and GPU_Miner [16]. The minimum and maximum execution time of every method are represented according to [14].

Table 3 Kmeans_Li comparison with several state of the art Kmeans GPU implementations

Methods	Time (s)	
	Small Data	Large Data
Kmeans_Li [14]	0,22 - 2,26	0,34 - 1,15
HP_Kmeans [19]	1,45 - 9,03	-
UV_Kmeans [15]	2,84 - 34,54	1,86 - 8,67
GPU_Miner [16]	61,39 - 474,83	4,26 - 40,6

We can see from this study that the Kmeans_Li [14] outperforms all of the evaluated methods in both small and large data sizes. We propose in the following section to compare our Kmeans GPU based implementation to the Kmeans_Li [14] method which is the best implementation. Evaluation was conducted within different data sizes. We call our Kmeans implementation, respectively, Kmeans_SD and Kmeans_LD according to the use of small or large datasets.

5.2.1 Kmeans Time Evaluation on Small Dataset (Kmeans_SD)

HBK relies on parallelizing Kmeans on small datasets. For this data size, Kmeans_SD fully processes on the GPU. However, Kmeans_Li processes the centroid updating on the CPU. In this subsection, we note Kmeans_SD and Kmeans_Li respectively by Kmeans-based GPU update and Kmeans-based CPU update. We compared the two Kmeans implementations on 8x8, 16x16 and 32x32 cropped documents. According to Figure 9, our Kmeans-based GPU update provides the lowest execution time in all cropped documents.

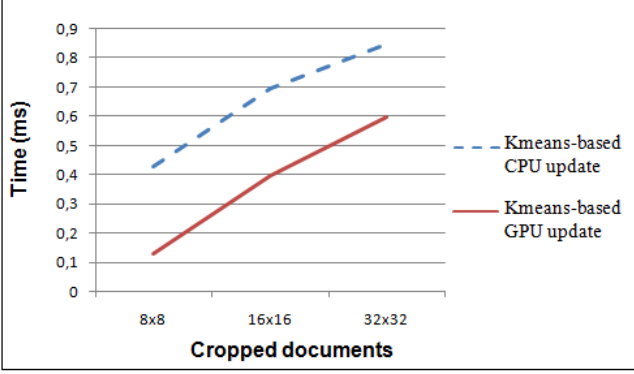


Fig. 9 Time evaluation of GPU Kmeans implementation with CPU and GPU based centroid update when processing different cropped documents

Actually, the CPU-GPU memory transfer cost is high on small data compared to the computation gained benefits. Thus, it is better to keep the centroid update stage of Kmeans on GPU. Indeed, our Kmeans_SD succeeds in avoiding a valuable CPU-GPU data transfer time. In addition, we observe that the smaller processed data size is, the lower Kmeans execution time is. Thus, 32x32 data size provides the highest execution time, however, 8x8 data size provides the lowest one. In the following, we adapt our Kmeans parallelization strategy to handle large dataset and show its efficiency.

5.2.2 Kmeans Time Evaluation on Large Dataset (Kmeans_LD)

Commonly, the Kmeans algorithm is processed on large datasets like A4 documents. In this context, Kmeans_Li [14] provides one of the most efficient Kmeans implementations. In this work, the data is divided into blocks of 256 data points. Moreover, centroids are stored in the constant memory. In the beginning of the algorithm, the data is dispatched once from the global memory to registers. Each thread computes the distance between each data point and all the centroids. Thus, it determines the closest center to the corresponding point. For the new centroid computation, a divide and conquer strategy is employed. Indeed, the data is divided into n' groups, where n' is initialized with $\frac{n}{M}$. With n is the data size belonging to the corresponding centroid, and M is a multiple of stream multiprocessor number in the GPU. Each group is reduced to get a temporary centroid. n' is then updated, by dividing the temporary centroids into n' groups and reduce them iteratively on the GPU until n' is smaller than M , indicating the GPU has no advantage than the CPU for further computing. In the end, the final centroids are computed on the CPU.

We adapt our Kmeans parallelization strategy to handle large datasets. Then, we compare this Kmeans_LD implementation to the Kmeans_Li one. Actually, we use

registers to store data points. Registers provide fast memory access during the process execution which will accelerate Kmeans. In another hand, we use shared memory to store the centroid values. In the beginning, each thread determines its closest centroid. Then, centers are computed directly from the first pixel distribution on the GPU blocks and store them in the shared memory. Finally, we reduce these centroids on the GPU until we get the final ones.

Y.Li evaluated his implementation using a dataset of 4 million points on a GTX 280 GPU architecture and an i5 CPU. We have implemented Kmeans_Li on our available hardware given in Section 5.1. In addition, we used the LRDE-DBD dataset composed with images of 8 million pixels which is larger than the dataset employed by Y.Li. Table 4 shows the time comparison between Kmeans_LD and Kmeans_Li on the LRDE-DBD dataset.

Table 4 Time evaluation of Kmeans_LD and Kmeans_Li on 125 (2516x3272) LRDE-DBD images

Methods	Time (ms)
Kmeans_LD	425
Kmeans_Li [14]	600

Kmeans_LD is 1.45x faster than the Kmeans_Li. Actually, in Kmeans_LD we do not redistribute the pixels on the GPU after distance computation. Therefore, we save processing time and ensure a coalescent memory read. However in the Y.Li Kmeans implementation, after distance computation the data is redistributed on the GPU to reduce the temporary centroids which is time consuming. Indeed, the GPU is setup with a new block number and a new pixel indexes are computed and dispatched to threads. Hence, pixel indexes become disorderly and cause a non coalescent memory read. In addition, in Kmeans_LD all centroids are computed in parallel inside blocks unlike Kmeans_Li that do rearrangement subsequently for each centroid.

Table 5 shows a comparison between our Kmeans GPU-based implementation and several state of the art methods in which we describe their strengths and areas of improvement. Actually, our method outperforms Kmeans_Si [18] because we parallelize efficiently the centroid updating stage on the GPU. The same strategy is done by UV_Kmeans [15] but the main limitation encountered, as in GPU_Miner [16] and Kmeans-Ta [17], is the lack of fast memory use. However, our method employs fast memories like registers and shared memory. Moreover, we ensure a Kmeans implementation that is more efficient than Kmeans_Li [14] according to two sides. Firstly, we reduce the transfer time between CPU and GPU when processing all Kmeans steps on the GPU for small datasets. Secondly, on the large datasets, we ensure more coalescent global memory access than

Table 5 Strengths and areas of improvement comparison between different GPU-based kmeans implementations.

Methods	Strengths	Areas for improvement
Our Kmeans implementation	Avoids costly CPU-GPU transfer time and ensures a coalescent memory access	Processing very large data
Kmeans.Li [14]	Decreases memory latency by using fast registers	Costly data transfer between CPU and GPU
GPU_Miner [16]	Avoids large atomic operations	Poor use of fast GPU memories
UV_Kmeans [15]	Uses the cache mechanism for high speed memory access	Fast memories such as registers and shared are not used
Kmeans.Ta [17]	Parallelizes all complex Kmeans steps on the GPU including the centroid updating stage	The use of global memory to permanently store centroids and data increases the memory latency
Kmeans.Si [18]	Offloads the Kmeans labeling stage to the GPU	Do not parallelize the Kmeans centroid updating stage

Kmeans.Li [14] because we do not rearrange data pixels. Thus, pixel indexes remain ordered in each block. In addition, all centroids are processed in parallel unlike Kmeans.Li [14] that processes and reduces one centroid at a time. In fact, our Kmeans implementation handles efficiently small and large datasets. Indeed, it decreases the memory transfer latency by using a carefully designed workload between CPU and GPU. Moreover, it ensures a coalescent and fast memory access thanks to the effective memory management.

Following, we evaluate the HBK GPU implementation that includes our Kmeans proposal. Then, we show its efficiency.

5.3 GPU-based HBK Time Evaluation

In the first part of this section, we evaluate the HBK GPU implantation to show the speedup effect when compared to HBK CPU-based one. For this, we study our HBK GPU implementation to determine the suitable block size that satisfies our paperless real time constraint. Moreover, the time performance of our GPU implementation is evaluated on different document parameters (noise amount and content). In the second part of this section, we compare the HBK GPU real-time score with that of different binarization GPU-based methods in the literature.

5.3.1 GPU-based HBK Speedup

Following, we evaluate our proposed HBK GPU implementation performances. In the beginning, we vary the block size parameters to see its effect on the GPU speedup. In this context, Table 6 shows the HBK execution time average across 125 Scanned documents.

We observe that the larger block size is, the less is the HBK processing time on CPU and the greater it

Table 6 Time evaluation of HBK method on CPU and GPU with different block sizes

Method	Blocs (pix)	Device	Time (ms)	Speedup
HBK	8x8	CPU	2716	-
		GPU	388	7x
	16x16	CPU	2546	-
		GPU	425	6x
	32x32	CPU	2411	-
		GPU	1050	2,29x

is on GPU. Moreover, the speedup of the HBK GPU implementation decreases when increasing the size of the blocks. Actually, the execution time on the GPU is faster when the number of pixels in the image blocks is close to the physical number of threads in the GPU blocks. In this case, each pixel is processed directly by a thread. Otherwise, the pixel follows a queue until a resource (thread) gets free. Moreover, the number of divergence branches is lower using smaller data. In another hand, by scoring 425 ms and 388 ms, the HBK is able to be integrated in the OCR paperless application that require to binarize one document per 460 ms. For these reasons, we consider that the block size of 16x16 pixels is the most suitable for our paperless application because it ensures both fast and acceptable binarization quality [1].

Next, we discuss the HBK time performance on GPU when varying the amount of noise. Figure 10 visualizes the HBK processing time on the CPU and the GPU architectures, represented respectively by purple color and dotted red one. This evaluation is performed across 125 Scanned and NScanned LRDE-DBD documents. We note that, the noise is greater on the Scanned documents (Figure 10.b) than on the NScanned ones (Figure 10.a) because of the degradation in the scan process. The execution time varies greatly between the two document

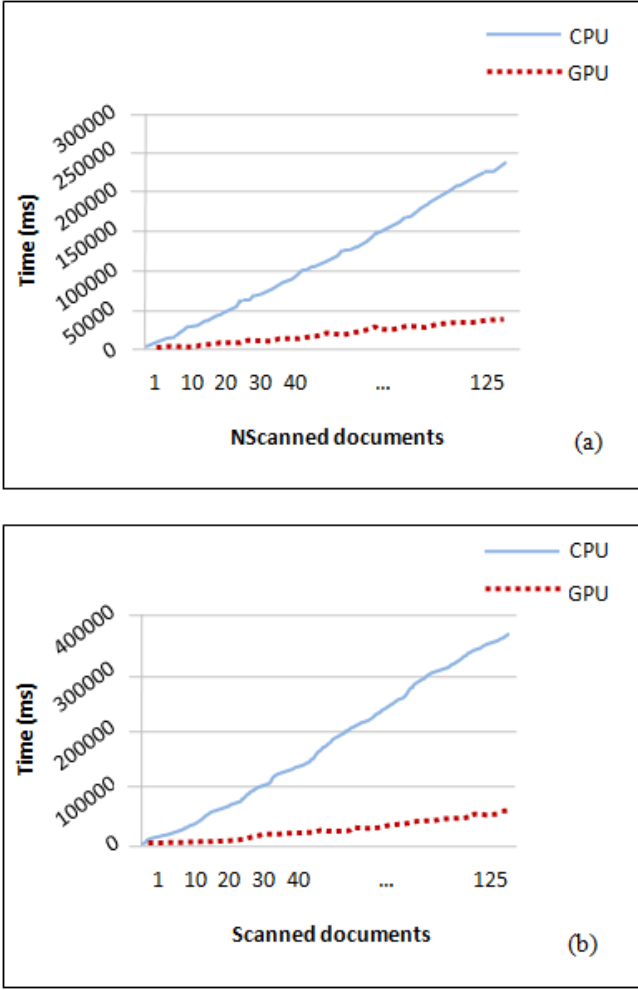


Fig. 10 Time evaluation of the HBK binarization on GPU and CPU. a NScanned documents. b Scanned documents

types on CPU. However, the process remains nearly constant on GPU. Indeed, the noise complexity induced in the Scanned documents requires more computation tasks which are efficiently parallelized on the GPU. To make a relevant comparison, we consider the HBK average time on the LRDE-DBD 125 documents as shown in Table 7.

Table 7 Time evaluation of the HBK method on CPU and GPU with different noise complexity images

Method	Images	Device	Time (ms)	Speedup
HBK	Scanned	CPU	2546	-
		GPU	425	6x
	NScanned	CPU	1930	-
		GPU	347	5.5x

The GPU version is 6x faster than the CPU one. In the other hand, the HBK GPU implementation is faster on NScanned documents, reaching 347 ms compared to

425 ms on Scanned ones. Indeed, the difference between execution time on Scanned and NScanned documents is due to the process complexity that affects the overall time. In fact, unlike the CPU implementation, our HBK parallelization on GPU is slightly sensitive to the noise introduced in documents after the scanning process.

Following, we evaluate the HBK processing time while varying the document content (number of text areas, colored background). In, Figure 11, the document1 contains a few text areas. The document2 includes more text areas and document3 contains many text areas and a colored background. The CPU time increases across the three document types. The more a document includes text and background the more processing time is. It reaches 4000 ms on document3 compared to 1100 ms in document1. The GPU execution time still nearly constant around 425 ms over the three document types. Moreover, the GPU speedup increases with the increase of the document complexity. The GPU reaches a speedup of 7.5x when binarizing the document3 compared to 3.5x when binarizing the document1. Indeed, the more information documents include, the more devices requires computation tasks. The GPU scores higher speedup because it is efficient for parallelizing high amount of computation. Indeed, the GPU implementation of HBK is able to establish a binarization time lower than our paperless application real time constraint of 460 ms, whatever are the noise or contents in the documents. Following, to establish the conceptual merit of our HBK implementation, we compare its real time with a set of binarization methods on GPU.

5.3.2 Real-time evaluation of HBK and different binarization methods

Several binarization methods were studied in our previous work [1]. A few of them have been implemented on GPU in the literature. In this section, we compare our GPU-based HBK to the GPU implementations of Niblack [11], Sauvola [12] and *Sauvola_{Mask}* [10] algorithms. We give a brief description of each algorithm and its GPU implementation.

Niblack [11] is a local thresholding algorithm. It computes a pixel-wise threshold by sliding a rectangular window over the gray level image. The threshold value for each pixel is decided by local mean and local standard deviation over a specific window size around each pixel. Thus, the local threshold $T(x, y)$ for pixel (x, y) is calculated by formula:

$$T(x, y) = m(x, y) + k \cdot s(x, y) \quad (1)$$

Where $m(x, y)$ and $s(x, y)$ are the local mean and the local standard deviation of the pixels within the local window region. The value of k controls the amount of

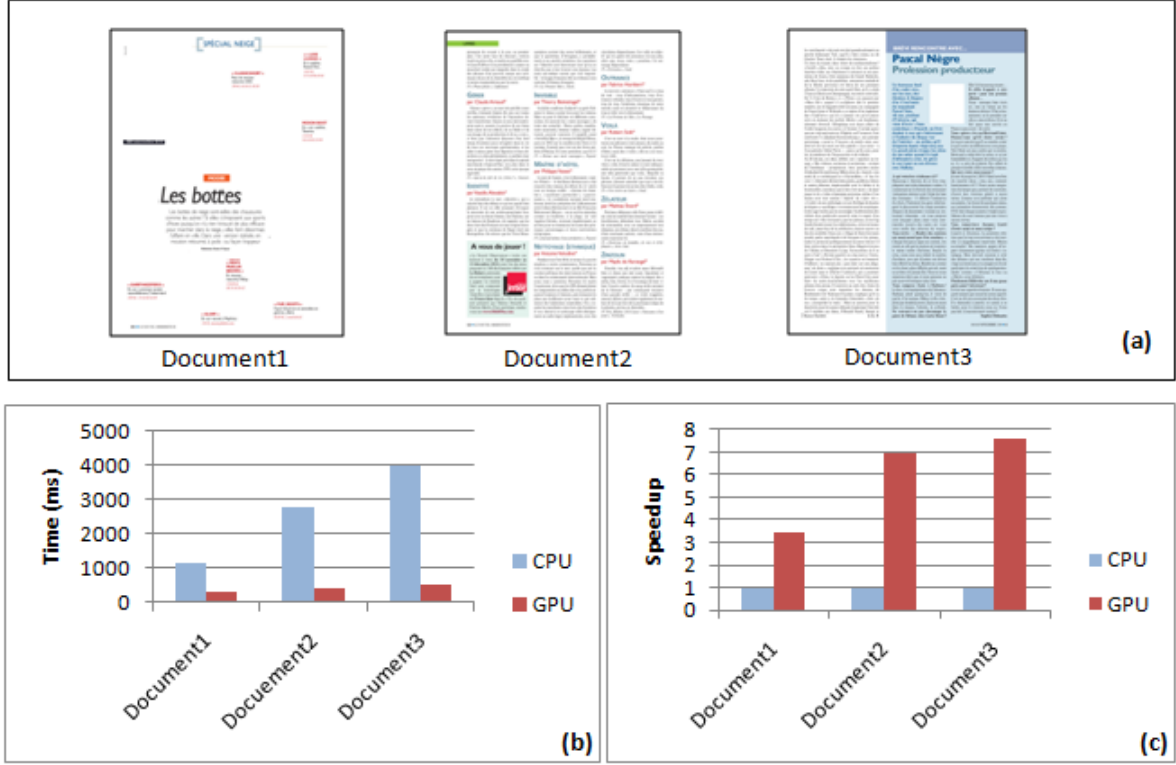


Fig. 11 Time evaluation of the HBK method on GPU and CPU when varying content. a Original scanned documents. b Processing time on GPU and CPU. c The speedup of GPU against CPU

text region inside the local window. Niblack was parallelized on the GPU by B.Singh [35]. It achieved an average speed up of 20.84x over the serial implementation when running on GeForce 9500 GT GPU and an Intel core 2 Duo CPU of 2.66 Ghz. The advantage of Niblack is that it always identifies the text regions correctly as foreground but on the other hand tends to produce a large amount of binarization noise in non-text regions [34].

Sauvola[12] proposed an algorithm similar to Niblack [11]. It made some assumptions based on the distribution of grey values associated with foreground and background pixels. The Sauvola threshold is computed as:

$$T(x, y) = m(x, y) + [1 + k \cdot (\frac{s(x, y)}{R} - 1)] \quad (2)$$

Where $m(x, y)$, $s(x, y)$ and k are the same parameters given in Niblack. R is the dynamic range of standard deviation. B.Singh parallelized Sauvola on GPU [36]. This GPU implementation achieved an average speed-up of 20.8x compared to the serial program.

Both Niblack and Sauvola GPU versions employ the same parallelization strategy. Indeed, the input image is stored as texture in the device memory. After that, block and grid sizes were computed according to the dimensions of the input images. A single thread calculates the threshold for a single pixel in the output image. In

our work, we have implemented Niblack and Sauvola on our GTX 660 GPU device.

Sauvola_{MSkx} [10] was proposed to improve Sauvola to handle the multi-scale text. It is composed of four steps. First, the image is sampled on different scales. Then, each image is binarized according to the conventional Sauvola method. Next, a threshold is set for each object according to its scale belonging. A final image including thresholds is produced and then binarized. According to our knowledge, *Sauvola_{MSkx}* was not implemented on the GPU yet. Thus, we have implemented our own version. Indeed, we parallelized the threshold computation on each scale and we kept the other steps performing on the CPU.

Generally, the adjustment of parameter k and w of these three methods requires prior knowledge about the set of documents. For our evaluation, we employ the parameters recommended by [10]. Indeed, we adjust $w = 51$ and $k = -0.2$ for the Niblack method. For Sauvola and *Sauvola_{MSkx}* we use $w = 51$ and $k = 0.34$. Table 8 shows the real time comparison between HBK, Niblack, Sauvola, and *Sauvola_{MSkx}* methods.

The evaluation result shows that HBK gives the best execution time. Indeed, it is 3.9x faster than *Sauvola_{MSkx}* and 1.4x faster than Niblack and Sauvola. Actually, our HBK GPU-based implementation is able to binarize pixels rapidly because all thread needed vari-

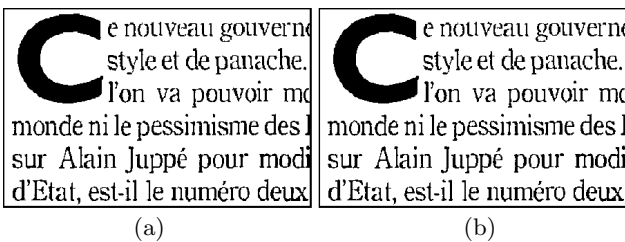
Table 8 GPU Real time evaluation of HBK and three binarization methods on the LRDE-DBD Dataset

Methods	Time (ms)
HBK	425
Niblack	594
Sauvola	595
<i>Sauvola_{MSkx}</i>	1660

ables are stored in the fast on-chip registers and shared memories. In the other hand, both Niblack and Sauvola need to compute a threshold for each pixel before binarizing the data. For this, they require fetching a great number of neighbor values from the texture memory to compute the mean and the standard deviation for each pixel. Otherwise, HBK employs faster memories compared to *Sauvola_{MSkx}* which uses further huge texture accesses to compute pixel thresholds on several scales. Therefore, HBK outperforms *Sauvola_{MSkx}* on the GPU. Niblack and Sauvola times are close, this is because the workflow is similar. However, Sauvola is slower because of its more complex threshold formula. *Sauvola_{MSkx}* gives the slowest execution time. Indeed, the multi-scale processing implies a cost on computation time. Following, we evaluate the binarization quality of our HBK GPU-based and CPU-based implementations to show the reliability of our implementation.

5.4 GPU-based HBK Quality Evaluation

Generally, the GPU architecture has some memory constraints and model execution rules to make faster implementations. According to Figure 12, the HBK GPU version does not show any loss in the binarization quality compared to the CPU one.

**Fig. 12** GPU (a) and CPU (b) based binarization quality comparison of the HBK method on cropped Scanned document

To prove this visual result, we compute the pixel based accuracy of the HBK on both CPU and GPU. The result gives the same F-Measure, for both devices. Indeed, our GPU program does not encounter any memory limitation or conflict that produces different binarization results.

6 Conclusion

In the paperless applications, the OCR systems are used to recognize text in the digital documents. In our work, we focused on the paperless real time application, in which, a scanner processes one A4 300 dpi paper per 23 seconds. The binarization is a very important component in the OCR tool chain. In this context, recently, we have proposed the HBK binarization method. It offers a high OCR accuracy compared to well known binarization methods. However, HBK does not respect our paperless real time constraint. It processes one document per 1.9 seconds, exceeding the binarization real time constraint of 460 ms. For this, we have proposed a parallel implementation of the HBK method on the Graphic Processing Unit (GPU). Our proposed CUDA implementation is based on combining fine-grained and coarse-grained parallel strategies and uses efficient memory management. According to our experiments, the GPU implementation of HBK performs one document per 425 ms which satisfy the paperless real time application constraint while offering a high OCR accuracy using 16x16 block size. Indeed, we have effectively avoided the communication overhead between the CPU and the GPU thanks to the full processing of Kmeans algorithm on GPU. In addition, our HBK GPU implementation is slightly sensitive to noise and document contents. The performed GPU-based comparisons of HBK to recent and well known binarization methods show that our proposed strategy implementation has a better performance and runs on GPU much faster than the compared methods.

References

1. Soua, M., Kachouri, R., Akil, M.: A new hibrid binarization method based on Kmeans. In: International Symposium on Communications, Control, and Signal Processing (2014)
2. Xiu, P., Baird, H.S.: Whole-Book Recognition. In: Pattern Analysis and Machine Intelligence, IEEE Trans. vol. 34, no. 12, pp. 2467-2480 (2012)
3. Kae, A., Huang, G.B., Doersch, C., and Learned-Miller, E.G.: Improving state-of-the-art OCR through high-precision document-specific modeling. In: CVPR, pp. 1935-1942 (2010)
4. Collins-Thompson, K., Schweizer, C., Dumais, S.: Improved String Matching Under Noisy Channel Conditions. In: Proceedings of the tenth international conference on Information and knowledge management, pp. 357-364 (2011)
5. Eikvil, L.: OCR-Optical Character Recognition (1993)
6. Gaceb, D., Eglin, V., Lebourgeois, F.: A new mixed binarization method used in real time application of automatic business document and postal mail sorting. In: Int. Arab J. Inf. Technol. 10(2), 179-188 (2013)
7. Ashari, E., Hornsey, R.: FPGA Implementation of real-time Adaptive Image Thresholding. In: Proc. Photonic Applications in Astronomy, Biomedicine, Imaging, Materials Processing, and Education, (2004)
8. Fong, W.: Document Imaging: A step toward a paperless office, <http://web.simmons.edu/~chen/nit/NIT%2792/133-fon.htm>

9. Kumar, D. and al.: MAPS: Midline analysis and propagation of segmentation. In: Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing. Article No. 15 (2012)
10. Lazzara, G., Graud, T.: Efficient Multiscale Sauvola's Binarization. In: International Journal on document analysis and recognition, (2013)
11. Niblack, W.: An Introduction to Digital Image Processing. In: Standberg Publishing Company, (1985)
12. J.Sauvola, T.Seppanen, S.Haapakoski, M.Pietikainen, Adaptive Document Binarization, 4th Int. Conf. On Document Analysis and Recognition, Ulm, Germany, pp.147-152 (1997).
13. Srinivasan, S. and al.: Performance characterization and acceleration of Optical Character Recognition on handheld. In: Workload Characterization (IISWC), IEEE International Symposium on, pp. 1-10 (2010)
14. Li, Y., Zhao, K., Chu, X. and Liu, J.: Speeding up k-Means Algorithm by GPUs. In: Proc. IEEE 10th Intl Conf. Computer and Information Technology (CIT), pp. 115-122 (2010)
15. Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., and Skadron, K.: A Performance Study of General-Purpose Applications on Graphics Processors Using CUDA. In: J. Parallel Distributed Computing, vol. 68, no. 10, pp. 1370-1380, Oct 2008
16. Fang, W. and al.: Parallel Data Mining on Graphics Processors. In: technical report, Hong Kong Univ. of Science and Technology, (2008)
17. Hong-tao, B., Li-li H., Dan-tong O., Zhan-shan L., and He, L.: KMeans on Commodity GPUs with CUDA. In: Proc. WRI World Co
18. Sirotkovic, J., Dujmic H., and Papic, V.: K-Means Image Segmentation on Massively Parallel GPU Architecture. In: Proc. 35th Intl Convention MIPRO, pp. 489-494 (2012)
19. Wu, R., Zhang, B., and Hsu, M.: Clustering billions of data points using GPUs, In: Proceeding of the combined workshops on Unconventional high performance computing workshop plus memory access workshop, Ischia, Italy, pp. 1-6, (2009)
20. Pisharath J., Liu, Y. Liao, W.-k., Choudhary A., Memik G., and Parhi, J.: 'Nu-MineBench 2.0'. In: CUCIS Technical Report Center for Ultra-Scale Computing and Information Security, Northwestern University, (2005)
21. LLOYD, S. P.: Least square quantization in PCM. In: IEEE Transactions on Information Theory vol.28, no.2, 129-137 (1982)
22. Smith, R.: An Overview of the Tesseract OCR Engine. In: Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR), 629-633 (2007)
23. EPITA and Development Laboratory (LRDE)
<http://www.lrde.epita.fr/cgi-bin/twiki/view/Olena/DatasetDBD>
24. Lelore, T., Bouchara, F.: Super-resolved binarization of text based on the FAIR algorithm. In: Proceedings of international conference on Document Analysis and Recognition, 13: 303-314 (2010)
25. Fabrizio, J., Marcotegui, B.: Cord, M.: Text segmentation in natural scenes using toggle-mapping. In: Image Processing (ICIP), 16th IEEE International Conference, 2373-2376 (2009)
26. Chen, T.Y. and al. On the statistical properties of the F-measure, Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on, pages 146-153, (2004)
27. NVIDIA.: CUDA C best practices guide (version 4.0), Santa Clara, California, USA. (2011)
<http://www.khronos.org/opencv/>
28. NVidia CUDA C Programming Guide,
<http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>
29. Khronos. OpenCL: The open standard for parallel programming of heterogeneous systems,
<http://www.khronos.org/opencv/>
30. Parallel programming and computing platform, CUDA, NVIDIA.
http://www.nvidia.com/object/cuda_hom_new.html.
31. Tompson, J. and Schlachter, K.: An introduction to the programming Model. In: Distributed Computing CSCI-GA.2631-001 (2012)
32. NVIDIA's Next Generation CUDA TM Compute Architecture: Kepler TM GK110,
<http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf>
33. Yi, Y., Lai, C., Petrov, S.: Efficient parallel CKY parsing using GPUs. In: J. Logic Computation, pp 175-185 (2011)
34. Khurshid, K. and al. Comparison of Niblack inspired Binarization methods for ancient documents, In proceeding of: 16th Document Recognition and Retrieval Conference, part of the IS and T-SPIE Electronic Imaging Symposium, San Jose, CA, USA, (2009)
35. Singh, B.M. and al.: Parallel Implementation of Niblack's Binarization Approach on CUDA. International Journal of Computer Applications 32(2):22-27, (2011).
36. Singh, B.M. and al.: Parallel Implementation of Souvola's Binarization Approach on GPU. In: International Journal of Computer Applications 32(2):28-33, (2011)

Annexe B

New Hybrid Binarization Method Based On Kmeans

M. Soua, R. Kachouri, M. Akil.

6th International Symposium on Communications, Control and Signal Processing (ISCCSP),
Athens, 21-23 May 2014

A NEW HYBRID BINARIZATION METHOD BASED ON KMEANS

Mahmoud Soua

Rostom Kachouri

Mohamed Akil

soua.mahmoud@esiee.fr

rostom.kachouri@esiee.fr

mohamed.akil@esiee.fr

ESIEE Paris, LIGM, A3SI, 2 Bd Blaise Pascal, BP 99, 93162 Noisy-Le-Grand, France.

ABSTRACT

The document binarization is a fundamental processing step toward Optical Character Recognition (OCR). It aims to separate the foreground text from the document background. In this article, we propose a novel binarization technique combining local and global approaches using the clustering algorithm Kmeans. The proposed Hybrid Binarization, based on Kmeans (HBK), performs a robust binarization on scanned documents. According to several experiments, we demonstrate that the HBK method improves the binarization quality while minimizing the amount of distortion. Moreover, it outperforms several well-known state of the art methods in the OCR evaluation.

Index Terms— Scanned documents, binarization, Kmeans, OCR

1. INTRODUCTION

Document binarization is an important pre-processing for image analysis especially Optical Character Recognition (OCR) [1]. It tries to separate the foreground text from the background. In general, approaches that deal with binarization are categorized in two main classes: global and local [2]. Principally, global methods are based on histogram, classification and clustering approaches. In the histogram category, Rosenfeld and Kak [3] select one threshold from the histogram of 2D document. They assume that gray values of each object are located around each histogram peak. Another well-known method is the Iterative Global Thresholding (IGT) [4]. It is composed of two phases executed iteratively. Firstly, the average color of the image is computed and subtracted from the image. Then, a histogram stretching is performed, in such way that the remaining pixels take up all of the gray-scale tones. The main drawback of histogram methods is the inaccurate detection of peaks caused by noise [5]. In the classification approaches, Otsu [6] is a very popular technique [7]. It finds an automatic threshold that reduces the inter-class variance between foreground and background. Many clustering methods are reported in the binarization literature. The Fuzzy C-means (FCM) [8] is one of the most efficient algorithms [2]. It sets the image pixels into the correspondent

foreground or background cluster. The membership degree of each pixel is computed according to a relatively complex fashion. Less complex [9], the Kmeans algorithm [10] clusters pixels according to a distance computation. According to the literature, Kmeans is a simple and flexible clustering method [11] that can be employed in binarization [2]. In general, global binarization methods are efficient when they are applied to documents with uniform illumination. However, degraded documents, including stains, and uneven contrast, are not well processed by global methods [12]. To overcome this problem, local methods are used to binarize degraded documents. For example, Niblack [13] computes a threshold based on the mean and the standard deviation of small neighborhood around each pixel. Actually, the Niblack method identifies the text regions correctly as foreground, however, it tends to produce a large amount of noise in non-text regions. Sauvola [14] improves Niblack by using the dynamic range of image gray-value standard deviation. In case of low separation between foreground and background, the result is degraded significantly. Wolf [15] outperforms Sauvola by maximizing the local contrast. However, performance degradation arises if there is a sharp change in background gray values across the image. Recently, Sauvola $M_{S_{kx}}$ [16] was proposed as an improvement to Sauvola. It begins with data sub-sampling. Then, binarization is performed at different scales. It should recognize that, local methods provide an adaptive solution since the binarization decision varies according to the properties of each document area. Local methods may introduce some noise like artefacts due to the local areas treatments. To improve the binarization quality, some recent works combine local and global approaches [2]. They perform an efficient local binarization while reducing the noise as the global one do [17]. For example, Gabara [18] uses local thresholding technique based on global edge detection. Hybrid IGT (HIGT) [19] applies the IGT global binarization [4] on the whole image then, reapplies it only on noisy areas. In most cases, the hybrid methods improve binarization quality on degraded documents.

In this context, we propose in this paper an adaptive method which we term Hybrid binarization based on Kmeans (HBK) to binarize scanned documents. Our method performs the Kmeans algorithm on local areas to get an improved re-

sult. Then, it employs a global technique to enhance local binarization. Performed experiments show that HBK is a robust binarization method. The obtained binarization allow to achieve higher OCR quality. The organization of the paper is shown as follow. Firstly, the Kmeans algorithm is presented in Section 2. We describe our proposed HBK method in Section 3. Next, in Section 4, experimental results are discussed. Finally, conclusion is drawn in Section 5.

2. KMEANS CLUSTERING

Kmeans [10] is one of the most popular unsupervised clustering algorithms, mainly used for data mining [20] and image processing [21]. Regarding the image processing context, Kmeans aims to partition all pixels in the image into k distinct clusters. Each cluster is represented by a single centroid.

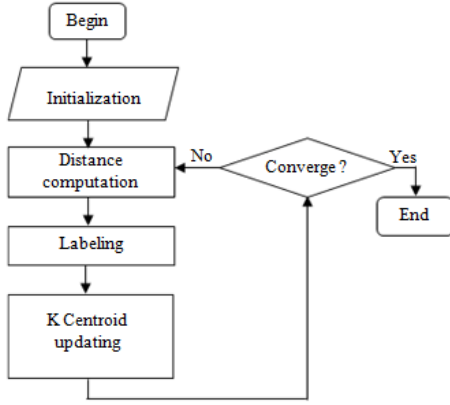


Fig. 1. The process flow of Kmeans

The Kmeans process flow is shown in Figure 1 and described as follow. In the first stage, the centroids number and values are initialized depending on the application. We note by $C = \{C_0, C_1, \dots, C_i, \dots, C_{k-1}\}$ the set of k input centroids. Afterward, the distances between centroids and pixels are computed to measure their similarity. For better clustering quality, the Euclidean distance is generally employed [22]. Given a centroid C_i and a pixel P_j the related Euclidean distance is defined as:

$$d(C_i, P_j) = \sqrt{(C_i^R - P_j^R)^2 + (C_i^G - P_j^G)^2 + (C_i^B - P_j^B)^2} \quad (1)$$

Where C_i and P_j are represented in the RGB color space by: $C_i = (C_i^R, C_i^G, C_i^B)^T$ and $P_j = (P_j^R, P_j^G, P_j^B)^T$, knowing that $i \in [0, k-1]$, $j \in [0, Np-1]$ and Np is the total number of pixels in the image. Next, the labeling stage sets the membership of pixels to the closest centroids according to the minimum computed distances. In addition, the pixel values and number in each cluster i are

added respectively to the corresponding accumulator $Accu = \{Accu_0, Accu_1, \dots, Accu_i, \dots, Accu_{k-1}\}$, and counter $Count = \{Count_0, Count_1, \dots, Count_i, \dots, Count_{k-1}\}$ variable sets, with $Accu_i = (Accu_i^R, Accu_i^G, Accu_i^B)^T$. Finally, in the k centroid updating stage, each centroid C_i , $i \in [0, k-1]$ is re-estimated by computing the average assigned pixels to the corresponding cluster i . This process is done based-on the previous computed accumulators and counters. The distance computation, the labeling and the centroid updating stages are repeated until overall pixels are firmly assigned and centroid are no longer being changed. Before each iteration, accumulators and counters are reset to zero. In next section, we describe our proposal which is based fundamentally on the Kmeans algorithm.

3. PROPOSED METHOD : HYBRID BINARIZATION BASED ON KMEANS (HBK)

As a hybrid approach, our proposed binarization method performs higher local binarization robustness and reduces the possible appearing artifacts by using global correctness. In the rest of the paper we call this method HBK. Global and local phases of the HBK method are shown in Figure 2.

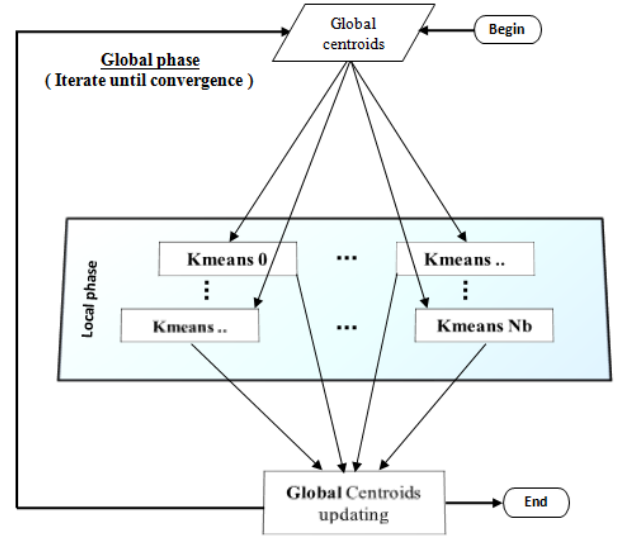


Fig. 2. The HBK method: global and local phases

Firstly, in the local phase, the input image (scanned document) is divided into Nb blocks. Then, the Kmeans algorithm is applied in each block. Afterward, the global phase gathers the obtained Kmeans results from each block iteratively and performs a correctness loop until convergence. More details of the HBK method are described in Algorithm 1.

Considering the binarization context, two initial centroids are employed. We note by $C_G = \{C_{G0}, C_{G1}\}$ the global centroid set. We initialize C_{G0} and C_{G1} with black and white

Algorithm 1 HBK(Img)

```

1:  $C_{G0} \leftarrow (0, 0, 0)^T, C_{G1} \leftarrow (255, 255, 255)^T$ 
2: while  $C_{Gi}(ite) \neq C_{Gi}(ite - 1), i \in [0, 1]$  do
3:   for  $bl \in [0, Nb - 1]$  do
4:      $Kmeans(bl, C_G)$ 
5:      $Reset(Accu_G, Count_G)$ 
6:     for  $i \in [0, 1]$  do
7:        $Accu_{Gi} \leftarrow Accu_{Gi} + Accu_{Lbl,i}$  Eq.2
8:        $Count_{Gi} \leftarrow Count_{Gi} + Count_{Lbl,i}$ 
9:     end for
10:   end for
11:    $C_{Gi} \leftarrow \frac{Accu_{Gi}}{Count_{Gi}}, i \in [0, 1]$  Eq.3
12: end while

```

values. Thus, $C_{G0} = (0, 0, 0)^T$ and $C_{G1} = (255, 255, 255)^T$ design respectively the background and the foreground clusters. The document is divided into blocks of equal sizes. Each block includes two local centroids given the set $C_L = \{(C_{L0,0}, C_{L0,1}), (C_{L1,0}, C_{L1,1}), \dots, C_{Lbl,i}, \dots, (C_{L Nb-1,0}, C_{L Nb-1,1})\}$, with $C_{Lbl,i} = (C_{Lbl,i}^R, C_{Lbl,i}^G, C_{Lbl,i}^B)^T$ given that $i \in [0, 1]$, $bl \in [0, Nb - 1]$. In the beginning, each local centroid is initialized with the corresponding global one: $C_{Lbl,0} = C_{G0}$ and $C_{Lbl,1} = C_{G1}$, $bl \in [0, Nb - 1]$. In addition, each block has two local accumulators and counters given respectively by the sets $Accu_L$ and $Count_L$:

$$Accu_L = \{(Accu_{L0,0}, Accu_{L0,1}), (Accu_{L1,0}, Accu_{L1,1}), \dots, Accu_{Lbl,i}, \dots, (Accu_{L Nb-1,0}, Accu_{L Nb-1,1})\} \text{ and}$$

$$Count_L = \{(Count_{L0,0}, Count_{L0,1}), (Count_{L1,0}, Count_{L1,1}), \dots, Count_{Lbl,i}, \dots, (Count_{L Nb-1,0}, Count_{L Nb-1,1})\}$$

with $Accu_{Lbl,i} = (Accu_{Lbl,i}^R, Accu_{Lbl,i}^G, Accu_{Lbl,i}^B)^T$ where $i \in [0, 1]$ and $bl \in [0, Nb - 1]$.

As stated in line 4, the Kmeans algorithm is applied across all blocks in the scanned document. When the Kmeans process on each block converges, all the local accumulators are gathered into a global one: $Accu_G = \{Accu_{G0}, Accu_{G1}\}$ with, $Accu_{Gi} = (Accu_{Gi}^R, Accu_{Gi}^G, Accu_{Gi}^B)^T, i \in [0, 1]$. Line 7 states for the global accumulators compute. This process is performed by one addition per each color component as stated in Eq.2:

$$Accu_{Gi} = \begin{pmatrix} Accu_{Gi}^R + Accu_{Lbl,i}^R \\ Accu_{Gi}^G + Accu_{Lbl,i}^G \\ Accu_{Gi}^B + Accu_{Lbl,i}^B \end{pmatrix}, i \in [0, 1], \quad (2)$$

$bl \in [0, Nb - 1]$

Similarly, local counters of each block are accumulated into global ones given by $Count_G = \{Count_{G0}, Count_{G1}\}$. Finally, global accumulators $Accu_G$ are divided by the correspondent counters $Count_G$ to compute the C_{Gi} new centroids as shown in Eq.3:

$$C_{Gi} = \begin{pmatrix} C_{Gi}^R \\ C_{Gi}^G \\ C_{Gi}^B \end{pmatrix} = \begin{pmatrix} \frac{Accu_{Gi}^R}{Count_{Gi}} \\ \frac{Accu_{Gi}^G}{Count_{Gi}} \\ \frac{Accu_{Gi}^B}{Count_{Gi}} \end{pmatrix}, i \in [0, 1] \quad (3)$$

If the global convergence is reached, the HBK algorithm stops, else it reiterates and local centroids of each block are reset with the computed global ones C_{G0} and C_{G1} . In the next section, we demonstrate that our method gives satisfying binarization quality thanks to the local approach.

4. EXPERIMENTAL RESULTS

In the following subsections, we present the employed dataset. Then, we demonstrate the efficiency of our proposed HBK method while reducing the distortion criterion. Finally, we compare our proposal to several state of the art methods in term of OCR evaluation.

4.1. The LRDE Document Binarization Dataset (LRDE-DBD)

In our experiments, we evaluate our HBK method on the LRDE-DBD¹ database [16]. It is composed of French text documents extracted from "Le Nouvel Observateur"² magazine. The provided dataset is composed of images with A4 format and 300-dpi resolution. Evaluation tests were performed on 125 scanned documents.

4.2. Distortion criterion-based evaluation

In general, to evaluate the efficiency of Kmeans clustering, the sum of cluster distortions is usually employed as a performance indicator [23, 24]. Given an image with Np pixels, we note by $\mathcal{C}(p_i)$ the Kmeans associated centroid of one pixel p_i , with $i \in [0, Np - 1]$. The distortion criterion measure is defined by Eq.4:

$$distortion = \frac{1}{Np} \sum_{i=0} |p_i - \mathcal{C}(p_i)|^2 \quad (4)$$

¹Copyright(c) 2012. EPITA and Development Laboratory (LRDE) with permission from Le Nouvel Observateur. LRDE-DBD is available online on the web site: <http://www.lrde.epita.fr/cgi-bin/twiki/view/Olena/DatasetDBD>

²Le Nouvel Observateur. Issue 2402, November 18-24, 2010 and available on the website: <http://tempsreel.nouvelobs.com>

For comparison reason, we implement a simple Kmeans binarization method that we term Global Binarization Kmeans (GBK). In this method, a single Kmeans is performed over the whole image. Keeping the goal of the binarization context, initial centroids of both GBK and HBK are set to the RGB values $C_{G0} = (0, 0, 0)^T$ and $C_{G1} = (255, 255, 255)^T$. To evaluate the clustering performances of these two methods, we measure the distortion criterion of GBK and HBK binarization on the LRDE-DBD database.

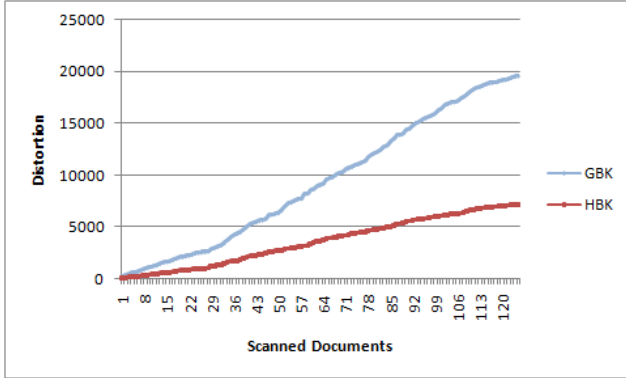


Fig. 3. Accumulated distortion computed from the obtained binarization results of GBK and HBK methods

Figure 3 shows the accumulated distortions of the two compared binarization methods on 125 scanned documents. The obtained result demonstrates that the HBK improves nearly 3x the distortion measure compared to GBK. Indeed, the treatment of Kmeans on small areas, gives local optimal distortions that minimizes the global one. We can come to the conclusion that our approach improves the binarization quality while reducing distortion in employed Kmeans clustering. In the next section, we perform an OCR evaluation to measure the HBK binarization efficiency.

4.3. OCR-based evaluation

Optical Character Recognition (OCR) is a process by which text characters contained in an image can be recognized. The binarization quality has a direct influence on the OCR result. In this context, the Tesseract 3.02 OCR [25] is used. Actually, we compare the character recognition rate of our HBK approach against seven well-ranked binarization methods [16, 14, 15, 6, 13, 26, 27] on the LRDE-DBD documents. Table 1 shows the OCR recognition rate of the evaluated methods. The first observation to be noted from this table, is that the OCR results are very close: The recent Sauvola Ms_{kx} gives an acceptable OCR rate by scoring 89% of accuracy. It ensures well text binarization but in some document areas, artifacts may appear leading to character miss-recognition. In the other side, the proposed HBK method gives the best OCR rate scoring 91% of accuracy thanks to its proper binarization re-

Table 1. OCR Accuracy evaluation of HBK and seven well-ranked binarization methods.

Methods	Images	OCR Accuracy (%)
HBK	Scanned	91
Sauvola Ms_{kx} [16]	Scanned	89
Wolf [15]	Scanned	88
Sauvola [14]	Scanned	87
Lelore [26]	Scanned	85
Otsu [6]	Scanned	84
Niblack [13]	Scanned	80
TMMS [27]	Scanned	73

sults. Indeed, in the HBK method, the local binarization gives robust binarization quality and the global approach eliminates the artifacts generated by the local approach.

5. CONCLUSION

Binarization is an important pre-processing step for optical character recognition (OCR). In this paper, we proposed a new hybrid binarization method based on the Kmeans clustering algorithm. We show that HBK performs robust binarization while minimizing the amount of distortion. Several experiments are performed on real magazine documents. The proposed HBK method outperforms several state of the art methods in the OCR-based evaluation while scoring 91% of accuracy. In future work, we intend to adapt the block sizes with kind of text in order to ensure multiscale binarization. In the other side, we note that the HBK method is highly adapted to parallel processing thanks to the no-dependency between image blocks. For this, we plan to use a parallel architecture in order to accelerate the HBK execution time.

6. REFERENCES

- [1] P. Xiu, H.S. Baird, *Whole-Book Recognition*, Pattern Analysis and Machine Intelligence, IEEE Trans. vol. 34, no. 12, 2012.
- [2] P. Stathis, E. Kavallieratou, N.Papamarkos, "An evaluation technique for binarization algorithms", Journal of Universal Computer Science, vol. 14, no. 18,3011-3030, 2008.
- [3] A. Rosenfeld, and A. C. Kak, *Digital Picture Processing*, 2nd ed. New York: Academic, 1982.
- [4] E. Kavallieratou, "A binarization algorithm specialized on document images and photos", 8th Int. Conf. on document Analysis and Recognition, pp. 463-467, 2005.

- [5] C. A. Glasbey, "An Analysis of histogram-based thresholding algorithm, *Graph*", *Models Image Processing* 55, 532-537, 1993.
- [6] N. Otsu, "A threshold selection method from gray-level histograms", *IEEE Transactions on Systems, Man and Cybernetics*, 9(1): 62-66, 1979.
- [7] F. Martn, "Analysis tools for gray level histograms", *Proc. Of SPPRA*, pp. 11-16, 2003.
- [8] R. Duda, and P. Hart, "Pattern Classification and Scene Analysis", Wiley, New York, 1986.
- [9] S. Ghosh, S.K. Dubey, "Comparative Analysis of K-Means and Fuzzy C-Means Algorithms", (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, 2013.
- [10] S. P. LLOYD, "Least square quantization in PCM", *IEEE Transactions on Information Theory* vol.28, no.2, 129-137, 1982.
- [11] V. Pritesh , O. Bhavesh , "A Survey on K-mean Clustering and Particle Swarm Optimization", *International Journal of Science and Modern Engineering (IJISME)*, 24-26, 2013.
- [12] M. Valizadeh, E. Kabir, "An adaptive water flow model for binarization of degraded document images", *International Journal on Document Analysis and Recognition (IJDAR)* June 2013, vol. 16, no. 2, pp 165-176, 2013.
- [13] W. Niblack, *An Introduction to Digital Image Processing*, Standberg Publishing Company, 1985.
- [14] J. Sauvola, M. Pietikainen, "Adaptive document image binarization", *Pattern Recognition*, vol. 33, 225-236, 2000.
- [15] C. Wolf, J.-M. Jolion, "Extraction and recognition of artificial text in multimedia documents", *Pattern Analysis and Applications*, vol. 6, 309-326, 2004.
- [16] G. Lazzara, T. Graud, "Efficient Multiscale Sauvolas Binarization", *International Journal on document analysis and recognition*, 2013.
- [17] M. P. Greenleaf, J. Shen, D. Yoon, "Local-Global Image Binarization reconstructing the cellular structure of polymer foam materials", *Computer-Aided Design and Applications*, 10(6), pp 919-928, 2013.
- [18] E. Gabarra, A. Tabbone, "Combining global and local threshold to binarize document of images", *Pattern Recognition and Image Analysis*, vol: 3523, 173-186, 2005.
- [19] V. Sokratis, E. Kavallieratou, R. Paredes, K. Sotiropoulos, "A hybrid binarization technique for document images", *Learning Structure and schemas from documents studies in computational intelligence*. Vol. 375, 165-179, 2011.
- [20] K. Singh, D. Malik, N. Sharma, "Evolving limitations in Kmeans algorithm in data mining and their removal", *IJCEM International Journal of Computational Engineering and Management*, vol. 12, 105-109, 2011.
- [21] N.A.M. Isa, S.A. Salamah, "Adaptive fuzzy moving K-means clustering algorithm for image segmentation", *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 4, 2009.
- [22] S. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions", *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4), 300-307, 2007.
- [23] M. B. Al-Daoud, N. B. Venkateswarlu and S. A. Roberts, "New methods for the initialization of clusters", *Pattern Recognition Lett.*, vol. 17, no. 15, 451-455, 1996.
- [24] J. M. Pena, J. A. Lazano, P. and Larranaga, "An empirical comparison of four initialization methods for the K-means algorithm", *Pattern Recognition Lett.*, 20, 1027-1040, 1999.
- [25] R. Smith, "An Overview of the Tesseract OCR Engine", *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, IEEE Computer Society, 629-633, 2007.
- [26] T. Lelore, F. Bouchara, "Super-resolved binarization of text based on the FAIR algorithm", *Proceedings of international conference on Document Analysis and Recognition*, 839-843, 2011.
- [27] J. Fabrizio, B. Marcotegui, M. Cord, : "Text segmentation in natural scenes using toggle-mapping". *Image Processing (ICIP)*, 16th IEEE International Conference, 2349-2352, 2009.

Annexe C

Improved Hybrid Binarization based on Kmeans for Heterogeneous document processing

M. Soua, R. Kachouri, M. Akil.

9th International Symposium on Image and Signal Processing and Analysis (ISPA), Zagreb,
7-9 Sept 2015

Improved Hybrid Binarization based on Kmeans for Heterogeneous document processing

Mahmoud Soua
soua.mahmoud@esiee.fr

Rostom Kachouri
rostom.kachouri@esiee.fr

Mohamed Akil
mohamed.akil@esiee.fr

Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI, ESIEE Paris, France

Abstract—Nowadays, more and more scanned documents are converted into editable electronic representation. This proceeding relies on the Optical Character Recognition (OCR) tool-chain. Generally, an OCR system is based on the important binarization step that separates character strokes from the background document. In this context, one of more robust binarization methods is the recently proposed Hybrid Binarization based on Kmeans (HBK). It handles effectively scanned documents which includes text on simple background. Nevertheless, in Heterogeneous documents, HBK ends up with some issues when extracting foreground text from complex background images. Moreover, HBK assumes to have a dark foreground against a clear background. Otherwise, it fails to render correct binarization colors. In this paper, we propose to improve the HBK method for handling efficiently Heterogeneous documents. Indeed, our proposal employs a layout analysis process that classify document regions into text and image. Image regions are enhanced with Gamma Correction (GC) before HBK binarization. Text regions are treated directly with HBK, keeping its effectiveness on text with homogeneous background. To ensure a robust and independent color rendering in the binarized documents, we control the labeling polarity of text and background through a pixel density-based technique. According to our experiments on LRDE and ICDAR datasets, we demonstrate that I-HBK outperforms HBK when dealing with Heterogeneous documents in both F-measure and OCR accuracy.

Keywords—OCR, Binarization, Heterogeneous Documents, HBK, Gamma Correction

I. INTRODUCTION

Optical Character Recognition (OCR) is a great interest in pattern analysis field. Its goal is to recognize the characters in a document to form a digital text file which can be edited and processed. Binarization, is a common and important first step in OCR systems. It converts the pixel values of document images into two-level representations for text and non-text regions.

Several binarization methods were proposed in the literature [5]. According to our knowledge, the Hybrid Binarization based on Kmeans (HBK) [7][1] outperforms state of the art methods such as SauvolaMS_{kx} [6] and Niblack [4]. HBK scores a character recognition rate of 91% when performing on magazine documents [7]. Actually, its hybrid approach ensures a robust binarization on scanned documents including text on homogeneous background. Indeed, it produces a local binarization for characters and decreases artifacts thanks to its global correctness. However, regarding heterogeneous documents including text, drawing and natural scene images, HBK fails to extract correctly text from complex background.

Moreover, similar to conventional binarization methods, HBK makes the assumption to have dark foreground against clear background and leads sometimes to generate inappropriate binary document colors. To overcome these limits, we propose in this paper an Improved HBK binarization that we call I-HBK.

To deal with text extraction from complex background, documents can be enhanced to increase the image visibility and details which helps to distinguish characters. In this context, some algorithms perform image enhancement before the binarization step [11]. For example, we can find the Histogram Equalization [9], Low Pass Filter [12], High Pass Filter [12], Homomorphic Filter [13] and image Gamma Correction (GC) algorithms [10]. Compared to the above methods, Gamma correction overcomes effectively the problem of uneven illumination [16]. In the other hand, the more efficient algorithm employed in the literature to handle the text and the background polarity problem, is the pixel density-based technique [17].

In the following, we study the binarization process of heterogeneous documents in section 2. Then, we describe our proposed I-HBK method in section 3. Next, in section 4, obtained results are shown and discussed. Finally conclusion is drawn in section 5.

II. HETEROGENEOUS DOCUMENT BINARIZATION

Scanned text documents are categorized into three main kinds: documents with Simple text images, documents with Caption text images and documents with Scene text images [15]. Furthermore, documents with heterogeneous content images can be considered as a fourth kind of scanned documents. It is generally named Heterogeneous Document and includes at least two of the above mentioned document types. Figure 1 illustrates an example of Heterogeneous Document that contains both text regions (Figure 1-a) and image regions (Figure 1-b: Caption text image, Scene text image).

Generally extracting text from Scene text images is more challenging than in Caption text ones. For this, we consider in the following only Simple text images and Scene text images to study the performance of robust binarization methods and discuss their limits on Heterogeneous documents. In the literature, our recently proposed Hybrid Binarization based on Kmeans (HBK) [7][1] outperforms well known binarization methods such as Niblack [4] and Sauvola [5]. It ensures high local binarization robustness and reduces the possible appearing artifacts using global correctness. In this binarization method, the

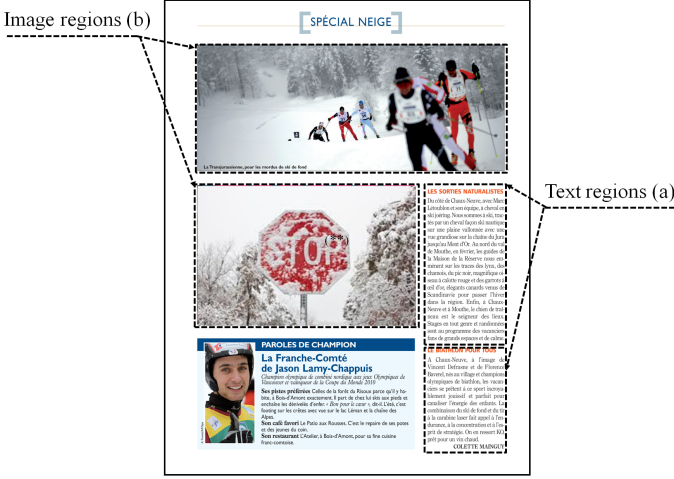


Figure 1: Heterogeneous Document: a. Text regions, b. Image regions.

input image is divided into equal sized blocks that can be set to small, medium or large according to the used character sizes in the document. Then, the Kmeans clustering algorithm [3] is applied on each block independently. The global phase gathers the obtained pixel values and number belonging to the related clusters in each block. The algorithm performs a correctness loop until convergence. According to Figure 2-(e,b-*), we can see that HBK provides high binarization quality on Simple text images, where characters are well separated. However, it does not do well on Scene text images with complex background, see Figure 2-(g,b).

To overcome the problem of extracting text from complex background images, some methods enhance the image before the binarization process. In this context, we consider approaches that look for a gamma value to help correct separation between foreground and background. The Gamma Correction Method (GCM) [8] is one of the most robust methods in this field. It aims to suppress non-text background details from scene text images by applying appropriate gamma value. For this, the GCM [8] transforms the input image to gamma-transformed images using a range of gamma values from 0.1 to 10.0 with increments of 0.1, resulting in 100 different images. A graylevel co-occurrence matrix for each image is computed to extract contrast and energy. Extracted textural feature measures of co-occurrence matrices with four orientations are averaged and threshold value is computed for each image using the Otsu algorithm [2]. Finally, Gamma, Contrast, Energy and Threshold values are examined to determine the appropriate value of gamma. Actually, GCM outperforms HBK on Scene text images with complex background and produces an efficient binarization quality Figure 2(g,c). However, according to Figure 2(e,c-*), GCM provides less accurate binarization quality than HBK on Simple text images. Thus, binarized characters are slightly touched due to the gamma transformation process. Table I illustrates the OCR Accuracy of HBK and GCM methods for Simple and Scene text images. The obtained results support the previous discussed visual observations.

We can see that, HBK outperforms GCM in processing

Table I: OCR Accuracy (%) of GCM and HBK methods on Heterogeneous documents: Simple and Scene text images

Methods	Simple text image (LRDE [6])	Scene text Images (ICDAR [18])
HBK [7]	94.11	50
GCM [8]	93.88	98.38

Simple text images by scoring 94.11% of OCR Accuracy. However, GCM is better than HBK in extracting text from scene text images when reaching 98,38% of OCR rate. Hence, we conclude that GCM and HBK methods can be complementary in Heterogeneous document binarization process.

In an another hand, we note that GCM and HBK methods yield to inverted foreground and background colors on Simple text image (Figure 2-(e,c-*), Figure 2-(e,b-*)) and on Scene text images (Figure 2-(f,c), Figure 2-(f,b)). We illustrate in Table II the effectiveness of the two studied methods. We note by D_f and C_f respectively dark and clear foreground. Likewise, C_b and D_b mean respectively dark and clear background.

Table II: Evaluation summary of HBK and GCM effectiveness on Simple and Scene text images

Methods	Simple text images				Scene text Image	
	D_f & C_b		C_f & D_b		-	-
	Single	Multiple	Single	Multiple	Single	Multiple
HBK [7]	Yes	Yes	No	No	No	No
GCM [8]	No	No	No	No	Yes	No

We observe from the Table II that on single regions HBK and GCM have a cross effectiveness when processing simple or Scene text regions. However, GCM always fails to provide high quality binarization when processing multiple regions. Otherwise, we remind that both HBK and GCM fail in some cases to extract the appropriate foreground and background colors. Considering the previous study, we describe our proposed I-HBK method in the following section.

III. IMPROVED HYBRID BINARIZATION BASED ON KMEANS (I-HBK)

Our proposed I-HBK method aims to improve HBK by providing a better binarization quality and OCR Accuracy on Heterogeneous documents. In this type of documents, the textual information can be included in text or image regions. For this reason, our idea is to benefit from the cross advantages of HBK and GCM on both region types. Indeed processing HBK on text regions and GCM on image regions improves the final binarization quality. As shown in Figure 3, our proposed method includes four major stages. Firstly, a Layout analysis is performed to identify text and image regions. Secondly, a binarization algorithm based on Gamma Correction (GC) is tailored to handle image regions. We name it Gamma Correction-based HBK (GCHBK). Similarly, HBK is performed on text regions. The third stage consists on a Pixel Density-based Technique (PDT), where a foreground and a background color study is performed. Finally, binarized regions are gathered in one single document.

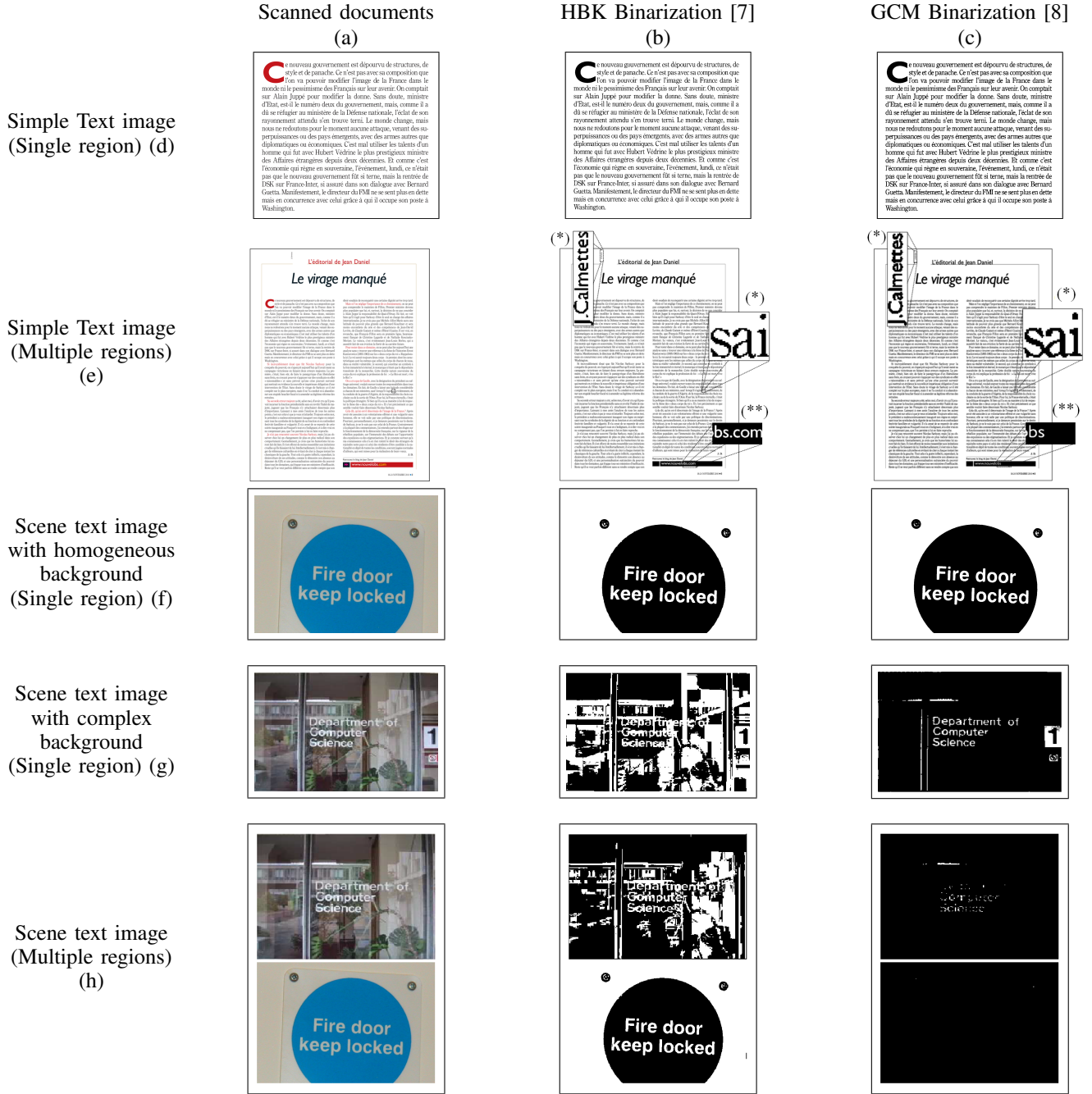


Figure 2: Visual binarization: a. Single and Multiple regions of scanned documents, b. HBK results, c. GCM results

A. Layout Analysis

It is important to recognize accurately whether a document region is text or image to perform the adequate treatment. For this, we perform a layout analysis on the input documents. We use the Page Layout Analysis (PLA) algorithm [19], part of the well known Tesseract engine 3.02 [19]. Actually, the PLA uses bottom-up methods, including binary morphology and Connected Component (CC) analysis, to estimate the type of the CCs. In addition, the method uses some measures to detect the tab-stops of column boundaries, indents, table columns. The column layout of the pages is determined from the detected

tab-stops. The column layout constrains the construction of partitions of the page that are then gathered into text regions. A similar process is applied to detect image regions.

B. Binarization stage (HBK and GCHBK)

After extracting regions from the heterogeneous document, we employ the appropriate binarization algorithm to process respectively text and image regions. Indeed, we improve the HBK method processing on image documents by including a pre-processing image enhancement. For this, we use a Gamma Correction approach (GC) that helps to separate foreground

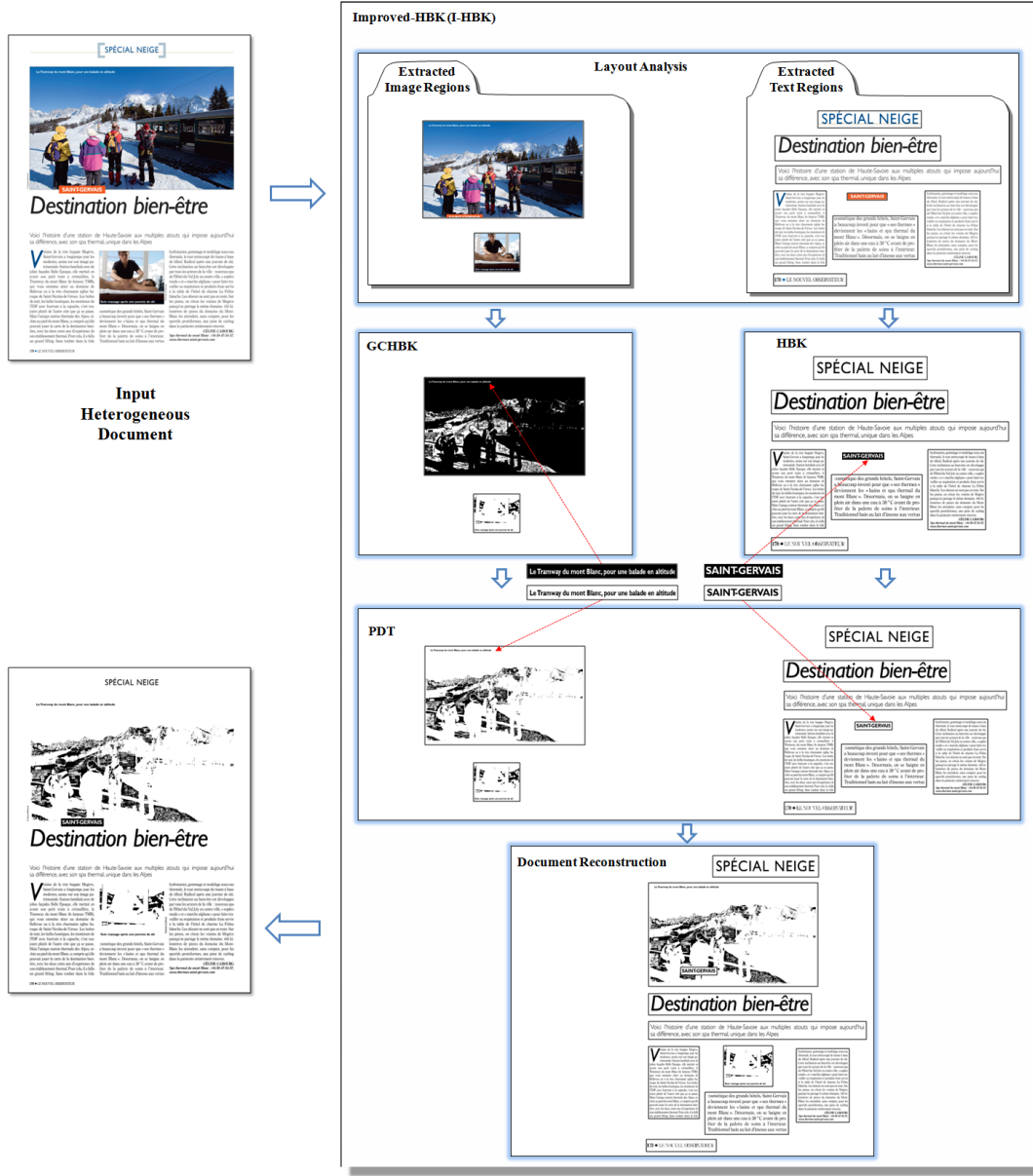


Figure 3: Flow diagram of the proposed I-HBK method

from complex background in text scene images. Actually, we employ GCM [8] Gamma Correction stage and we replace the Otsu binarization method with the HBK one. According to the literature [7], binarizing with HBK instead of Otsu is more effective scoring 91% of OCR Accuracy against 84%. We name this approach Gamma Correction-based HBK (GCHBK). Our evaluation of the proposed GCHBK is shown in the experimental results section.

C. Pixel Density-based Technique (PDT)

According to our Knowledge several binarization methods [4][5][7] assume to have a dark foreground against a clear background. Otherwise, they fail to render correct binarization colors. To overcome this problem some methods control the labelling polarity of text and background. For example, Clark [17] uses a simple decision logic based on the assumption

that the background pixel number is greater than the text pixel one. This method fails in case characters are thick and occupy a significant area of the used window technique. So instead of using a window approach, we apply the same concept of pixel density-based technique on the extracted regions. This way, we control the foreground and background color polarity according to Equ 1.

$$P_i = \begin{cases} 0 & \text{if } N_{Bp} > N_{Wp} \\ P_i & \text{else} \end{cases} \quad (1)$$

with P_i designs a pixel, i in $[0, N_{Bp} + N_{Wp}]$, and N_{Bp} , N_{Wp} are respectively the number of black and white pixels in the studied region. The final reconstitution stage gathered the binarization of the two methods into one single document according to the recorded region coordinates.

IV. EXPERIMENTAL RESULTS

In the following, we evaluate our proposed I-HBK method. Firstly, we introduce the employed materials. Then, we compare the OCR Accuracy of the GCM and our proposed GCHBK technique to show its effectiveness when binarizing scene text images. Following, we evaluate our proposed I-HBK when comparing it with HBK according to Fmeasure and Visual quality. Finally, we study I-HBK using the OCR Accuracy.

A. Materials

The proposed I-HBK method was evaluated on LRDE-DBD¹ [6] and ICDAR² [18] datasets. In the following experiments, we fix the HBK block size to 32x32 pixels. According to the literature, this block size gives an acceptable binarization quality [7]. We use the Fmeasure metric to evaluate the binarization accuracy. In addition, Tesseract 3.02 was employed to perform the character recognition study. Following, we evaluate our GCHBK binarization approach on natural scene images.

B. GCM and GCHBK Evaluation

We evaluate our proposed enhancement of HBK binarization on image documents using the Gamma Correction approach (GC). Indeed, we perform the image enhancement part of GCM then we use HBK as a final binarization process in place of Otsu (see section III-B). Table III compares GCM and our GCHBK technique on ICDAR scene text images. We note that GCHBK provides better character recognition

Table III: Comparison of GCM and GCHBK methods based on OCR Accuracy for ICDAR scene text images

Methods	OCR Accuracy (%)
GCM: GC + Otsu	84.4
GCHBK: GC + HBK	85.2

accuracy by scoring 85.2%. We explain this result with the fact that HBK method outperforms Otsu in the binarization step. Indeed, the employed hybrid processing strategy takes into account both local and global pixel information to provide efficient binarization result. We provide a visual result in Figure 4. As we can see, characters are better separated using GCHBK.

C. I-HBK evaluation based on Visual quality

Following, we show in Figure 5 the binarization results of HBK, GCHBK and our I-HBK methods when processing a sample Heterogeneous document from the LRDE-DBD dataset. We note that I-HBK gives clearly the best binarization result. HBK produces high binarization quality on text regions. However, it fails in rendering the appropriate colors of text and background because it assumes that text is in dark color

¹Copyright(c) 2012. EPITA and Development Laboratory (LRDE) with permission from Le Nouvel Observateur. LRDE-DBD is available on-line on the web site: <http://www.lrde.epita.fr/cgi-bin/twiki/view/Olena/DatasetDBD>

²ICDAR 2013 dataset is available on the web site <http://dag.cvc.uab.es/icdar2013competition>

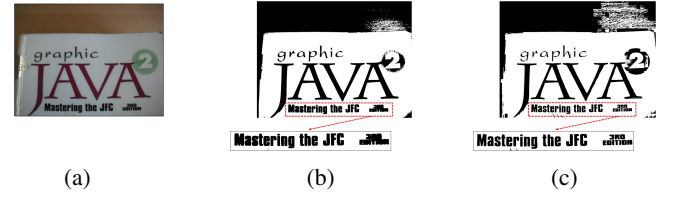


Figure 4: Binarization quality of GCM and GCHBK on text scene image from ICDAR dataset. a. Original image, b. GCM binarization, c. GCHBK binarization

and background is in clear one. GCHBK succeed in binarizing image regions. However it shows information loss in text regions with homogeneous background.

Following we evaluate HBK and I-HBK methods on both Simple text images and Heterogeneous documents.

D. Comparison of HBK and I-HBK methods based on analytic analysis

The HBK method is robust binarization method on Simple text images. Following, we demonstrate in Table IV the reliability of I-HBK on LRDE-DBD images that include only text information.

Table IV: Comparison of HBK and I-HBK methods based on Binarization-Accuracy on 125 Simple Text LRDE-DBD images

Methods	FMeasure (%)
HBK [7]	98.21
I-HBK	98.84

As expected, I-HBK and HBK have a close Fmeasure result on simple text images. Indeed, the processed documents contain only text regions that include, in most cases, dark text on light foreground. We observe a small improvement as I-HBK handles properly regions with light text on dark background. Thus, text and background colors are rendered correctly thanks to the used pixel density-based technique. In addition, some text regions include non-homogeneous background, on which I-HBK gives better text binarization quality. Next, we demonstrate the effectiveness of our proposal using the OCR Accuracy on Heterogeneous documents.

To process Heterogeneous documents, we choose a set of LRDE-DBD documents including Scene text images. For this purpose, we construct their OCR ground-truth as they are not available in the LRDE-DBD Dataset. Figure 6 shows the OCR accuracy of our proposed I-HBK and HBK methods on these documents. We observe that our I-HBK proposal improves considerably the OCR accuracy of HBK. This is due to the good workload distribution between HBK and GCHBK methods. Furthermore, GCHBK allows to extract well the text in Scene text images contrary to HBK that mixes foreground and background colors.

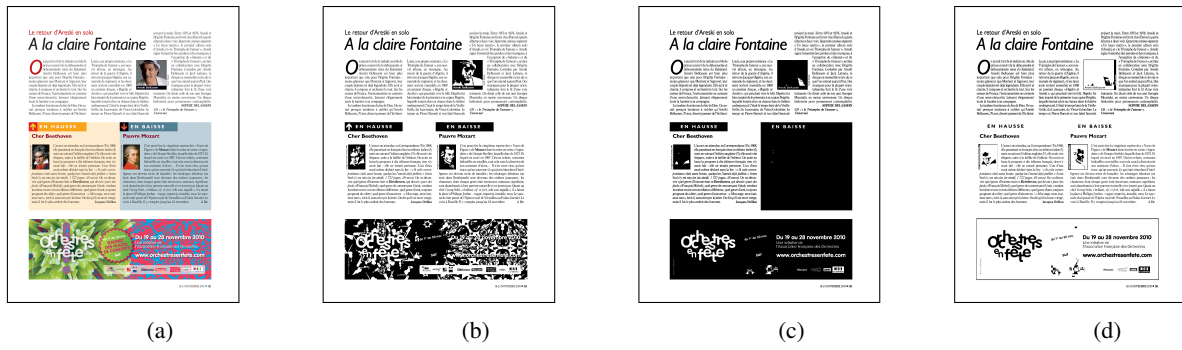


Figure 5: Visual evaluation of the I-HBK method: a. Original document, b. HBK binarization and c. GCHBK binarization, d. IHBK binarization

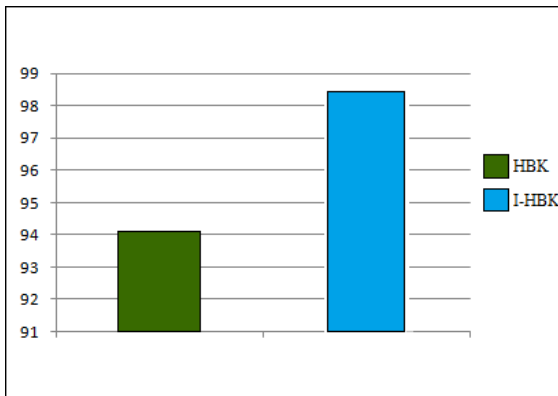


Figure 6: Comparison of HBK and I-HBK OCR accuracy on Heterogeneous LRDE-DBD documents

V. CONCLUSION

Binarization is an important step in any task of image processing and analysis. Recently we proposed the Hybrid Binarization based on Kmeans method (HBK). To process heterogeneous documents, we proposed in this paper a new binarization method named Improved-HBK (I-HBK). In addition of HBK text binarization, our proposal performs layout analysis scheme, then employs a modified Gamma Correction approach GCHBK to extract text from complex background. In addition, it studies the image regions to control the final color rendering of text and background. According to experiments, we demonstrated that I-HBK outperforms HBK in both FMeasure, and OCR accuracy when processing LRDE and ICDAR documents.

REFERENCES

- [1] M.Soua, R.Kachouri, M.Akil GPU parallel implementation of the new hybrid binarization based on Kmeans method (HBK) Journal of Real-Time Image Processing, Springer: 1-15, 2014.
- [2] N. Otsu. A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man and Cybernetics, 9(1): 62-66, 1979.
- [3] S. P. LLOYD, Least square quantization in PCM, IEEE Transactions on Information Theory 28(2): 129-137, 1982.
- [4] W. Niblack, An Introduction to Digital Image Processing, Standberg Publishing Company, 1985.
- [5] J. Sauvola, M. Pietikainen, Adaptive document image binarization, Pattern Recognition, 33: 225-236, 2000.
- [6] G. Lazzara, T. Gaud, Efficient Multiscale Sauvolas Binarization, International Journal on document analysis and recognition, 2013.
- [7] Soua, M., Kachouri, R., Akil, M.: A new hybrid binarization method based on Kmeans. In: IEEE International Symposium on Communications, Control, and Signal Processing: 118-123, 2014.
- [8] C. P. Sumathi and G. Gayathri Devi, Automatic Text Extraction From Complex Colored Images Using Gamma Correction Method, Journal of Computer Science, 10(4): 705-715, 2014
- [9] Pizer, S. M., E.P. Ambum and J.D. Austin, Adaptive histogram equalization and its variation, Computer Vision, Graphics, and Image Processing 39(3): 355-368, 1987.
- [10] Asadi Amiri, S., Hassanpour .H. and Pouyan, A., Texture Based Image Enhancement Using Gamma Correction, Middle-East Journal of Scientific Research. 6: 569-574, 2010.
- [11] Gilboa, G., Sochen, N. and Zeevi, Y. Y., Image enhancement and denoising by complex diffusion processes, IEEE Transactions on Pattern Analysis and Machine Intelligence. 26(8): 1020-1036, 2004.
- [12] Guillon, S., Baylou, P., Najim, M. and Keskes, N., Adaptive nonlinear filters for 2D and 3D image enhancement, Signal Processing, 67(3): 237-254, 1998.
- [13] Fries, R. and Modestino, J., Image enhancement by stochastic homomorphic filtering, IEEE Transactions on Acoustics, Speech, and Signal Processing, 27(6), 625-637, 1979.
- [14] Farid, H., "Blind inverse gamma correction", IEEE Transactions on Image Processing, 10: 1428-1433, 2001.
- [15] C.p Sumathi. A survey on various approaches of text extraction in images. International Journal of computer sciences engineering 3(4): 15-19, 2012.
- [16] K.Kaur, N.Kanwal, J.S.Bhullar. A Technique for Enhancement of Gray Image using Local Gamma Correction. International Journal of Computer Applications. 105(5):36-39 ,2014
- [17] P. Clark and M. Mirmehdi. Rectifying perspective views of text in 3D scenes using vanishing points. Pattern Recognition, 36(11): 2573-2686, 2003.
- [18] Karatzas, D. and F.Shafait, S.Uchida, M.Iwamura. ICDAR Robust Reading Competition. Document Analysis and Recognition (ICDAR): 1484-1493, 2013.
- [19] R. Smith, An Overview of the Tesseract OCR Engine. International Conference on Document Analysis and Recognition, 2007.

Annexe D

Efficient Multiscale and Multifont Optical Character Recognition System based on robust Feature Description

M. Soua, R. Kachouri, M. Akil.

International Conference on Image Processing Theory, Tools and Applications (IPTA), Orleans, 10-13 Nov 2015

Efficient Multiscale and Multifont Optical Character Recognition System based on robust Feature Description

Mahmoud Soua, Rostom kachouri and Mohamed Akil

Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI, ESIEE Paris, France
e-mail: mahmoud.soua@esiee.fr, rostom.kachouri@esiee.fr, mohamed.akil@esiee.fr

Abstract—Optical Character Recognition (OCR) is the process of translating images of text into a comprehensible machine format. Generally, an OCR system is composed of binarization, segmentation and recognition stages. Given an extracted binary character, the recognition stage ensures its description and decides its corresponding ASCII code. In this paper, we propose a new OCR system that aims to high speed, Multiscale and Multifont character recognition. Our proposal is based essentially on robust description using a new Unified Character Descriptor (UCD). In addition, a character type-face and font-size recognition is performed to choose the adequate template for faster matching process. Obtained OCR Accuracy of our proposed System is 1.5x higher then that reached by Tesseract on the LRDE dataset.

Keywords—OCR System, Multiscale, Multifont, Feature Extraction, Feature Matching, SAD technique.

I. INTRODUCTION

Optical Character Recognition (OCR) deals with the problem of recognizing optically processed characters. OCR systems involve three major stages to completely recognize characters; Binarization, Segmentation, and Recognition stages [1]. Firstly, Binarization separates the text characters from the background [6]. Then, Segmentation stage aims to locate text regions in the processed documents [7]. Finally, the Recognition stage consists on a very sensitive character description and decision steps.

Generally, state-of-the-art methods in the description step are based either on Template Description (TD) or on Feature Description (FD) [1]. In TD Methods [1][2], characters are described based on their pixel information. Despite their simple use, these methods are not robust on noisy characters. In addition, calculation are burden with unnecessary pixel description. Less complex, the FD methods [3][4][5] perform the description of characters based on some specific Features.

In the other hand, the decision stage is made according to a matrix or extracted feature matching. As mentioned above, pixel comparison in matrix matching is extremely sensitive to noisy characters [1][2]. This stage can be performed using classification methods such as RNA, SVM [3][4][5]. They give interesting results, however they still complex comparing to matching process. Indeed, simple feature matching ensures a good trade off between accuracy and computation complexity [5].

In this paper, we propose a novel Multiscale and Multifont OCR System based on a robust feature description. Firstly, we ensure the right template selection. Then, we compute

a Unified Character Descriptor (UCD) and a fast Matching process is performed.

In the following, we present our proposed OCR System in section 2 where the new template selection and Character Recognition based on UCD (CR-UCD) method is explained. The obtained results are shown and discussed in section 3. Finally conclusion and future work are drawn in section 4.

II. PROPOSED OCR SYSTEM BASED ON CR-UCD RECOGNITION

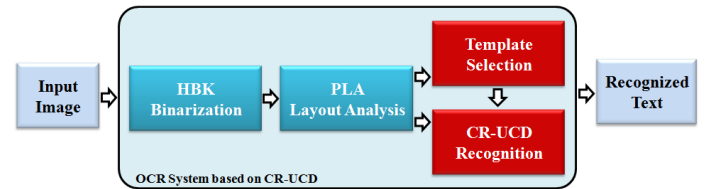


Fig. 1: Block diagram of our proposed OCR System

We design a complete OCR System that performs character recognition on input image containing text information. As seen in Figure 1, our OCR System includes the stages of binarization, segmentation, template selection and recognition. In the binarization stage we use our recently proposed Hybrid Binarization Based on Kmeans (HBK) [6][13] method to separate correctly the text and the background even on noisy images. Then, binarized image is segmented with the Page Layout Analysis (PLA) [10] part of the well known Tesseract 3.02 engine [20]. In the template selection stage, character font-size and type-face are recognized and subsequently the adequate template is chosen. Finally, we perform our CR-UCD recognition in which each character is represented firstly in the Description stage using one UCD feature. Then, characters are recognized based on Sum of Absolute Difference (SAD) feature matching with the selected template. An example of letter 'A' recognition with our proposed method is given in Figure 2, we assume that the considered letter 'A' is extracted from the word 'Académies'. In this figure, the different proposed stages of our new method, namely: the Template Selection, Description and Decision stage are illustrated. In the first stage a Template Selection Descriptor (TSD) is generated to allow the appropriate template choice. In an other hand, the description stage extracts the Unified Character Descriptor (UCD) from each character and feeds it to the decision stage. In this stage the extracted UCD vector is matched with the

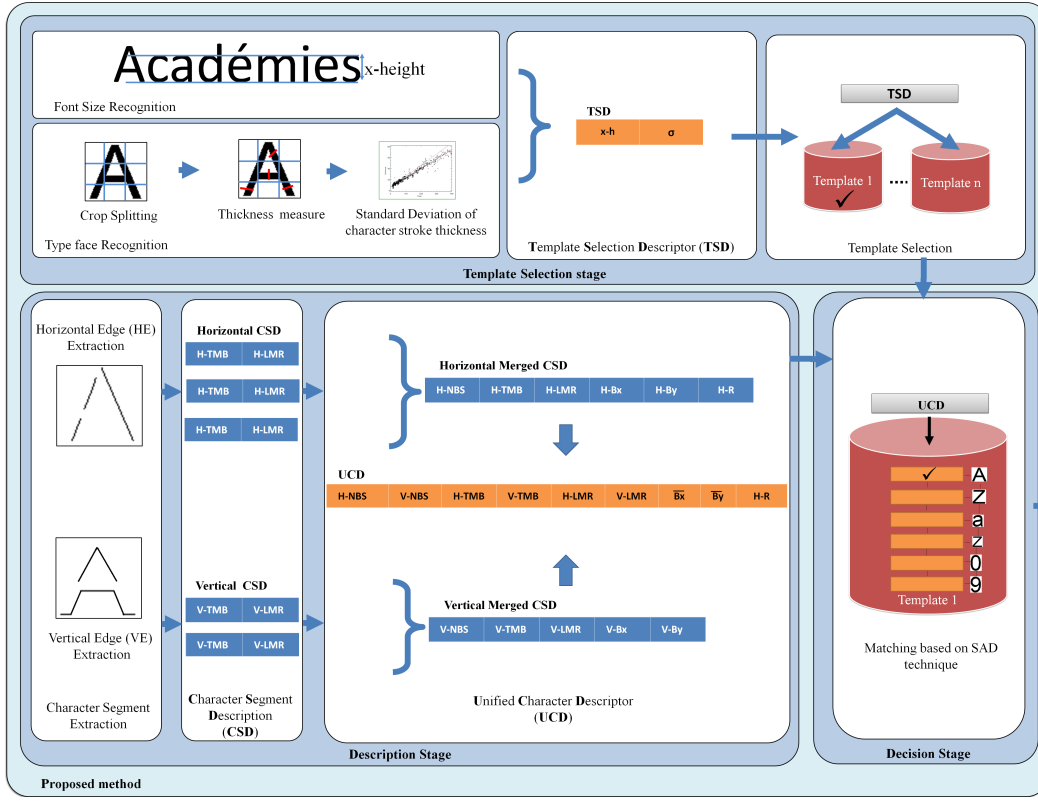


Fig. 2: Letter 'A' recognition with the proposed method

closest character in the selected template. These stages are given with more details in subsequent sections. Firstly, we explain the Description stage. Then, we discuss the template selection one and we present the decision stage.

A. The Description Stage

In this section, we present our Feature Extraction Strategy. The aim of this stage is to employ a sufficient number of characteristics that helps to discriminate characters efficiently. For this, we perform in our work a Character Segment Extraction by using a simple Edge Detection. We extract then horizontal and vertical character segments. In the Horizontal Edge (HE) extraction, for each pixel P_{ij} we compute the right edge according to Equation (1). Where $i \in [1..h]$ and $j \in [1..w]$, given that h and w are respectively the height and width of the character bounding box. We note by 0 a black pixel and 1 a white one.

$$P_{i,j} \in \begin{cases} \text{HE} & \text{If } (P_{i,j} = 0) \text{ And } (P_{i,j+1} = 1) \\ \overline{\text{HE}} & \text{If } ((P_{i,j} = 0 \text{ And } (P_{i,j+1} = 0) \text{ Or } (P_{i,j} = 1)) \end{cases} \quad (1)$$

In the other side, the Vertical Edge (VE) extraction, for each pixel P_{ij} is computed as shown in Equation (2).

$$P_{i,j} \in \begin{cases} \text{VE} & \text{If } (P_{i,j} = 0) \text{ And } (P_{i+1,j} = 1) \\ \overline{\text{VE}} & \text{If } ((P_{i,j} = 0 \text{ And } (P_{i+1,j} = 0) \text{ Or } (P_{i,j} = 1)) \end{cases} \quad (2)$$

Generally, when re-scaling characters or when we handle the same character font in many sizes, we change the character morphology and some distortions can appear in the character description. This issue is named the aliasing behavior [10]. As shown in Figure 3, due to this phenomena, one character can have a different number of segments on multiple Font-Sizes.

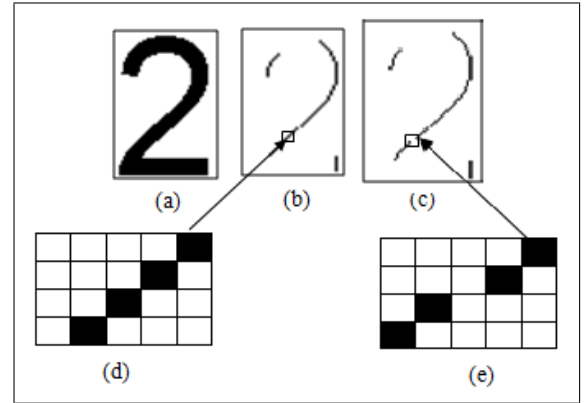


Fig. 3: Dependency of scale and character segment discontinuity: a. Input Images, b. Extracted Horizontal Edges in one scale, c. Extracted Horizontal Edges in another scale, d and e. A corresponding zoom views of the extracted segments

For this, we propose to check the extracted segments to merge some edges based on their neighborhood. To ensure an efficient and scale invariant method, we use each time a

variable neighbour pixel number for the check. This parameter is adjusted according to h and w values of the processed character (its size). This proposed technique allows to improve the description of multi-scale character.

As shown in Figure 2, the Character Segment Description (CSD) of each Horizontal and Vertical extracted segments is firstly ensured by the Character Segment Position (TMB and LMR). Then one Horizontal Merged CSD is constructed while adding respectively H-TMB and H-LMR of each Horizontal segment. To which we concatenate other features like the total Horizontal Character Segment Number (H-NBS), the Horizontal Character Barycentre Coordinates (H-Bx and H-By), and the Horizontal Character Ratio (H-R). We do the same thing for the Vertical Merged CSD computation except the Character Ratio (R) feature which is not used in this case. The employed features in this work are presented and explained with more details in the following subsections:

1) *Character segment Number Feature (NBS)*: We consider that this feature increases the discrimination between characters that do not have the same number of segments.

2) *Character Segment Position Feature (TMB, LMR)*: As shown in Figure 2, the CSD vector is composed of two features: The first one, TMB, refers to the Top, Middle and Bottom segment positions in the character bounding box. We compute TMB according to Equation (3).

$$TMB = \begin{cases} 1 & \text{If } (S_y > \frac{h}{2}) \text{ (Top)} \\ 2 & \text{If } (S_y = \frac{h}{2}) \text{ (Middle)} \\ 3 & \text{If } (S_y < \frac{h}{2}) \text{ (Bottom)} \end{cases} \quad (3)$$

with S_y is the starting y coordinates of the profiled segment and h is the height of the character bounding box. The second feature is LMR. It refers to the Left, Middle and Right segment positions in the character bounding box. It is computed as shown in Equation (4).

$$LMR = \begin{cases} 1 & \text{If } (S_x < \frac{w}{2}) \text{ (Left)} \\ 2 & \text{If } (S_x = \frac{w}{2}) \text{ (Middle)} \\ 3 & \text{If } (S_x > \frac{w}{2}) \text{ (Right)} \end{cases} \quad (4)$$

with S_x is the starting x coordinates of the profiled segment and w is the width of the character bounding box.

3) *Character Barycentre Coordinate Feature (Bx, By)*: Starting from the motivation that different character shapes have different barycentre positions, as shown in Figure 4, we propose a novel and simple barycentre computation technique. Indeed, we consider the polygon composed only with the starting segment pixels. Then, we compute the x and y coordinates of the corresponding barycentre for both Horizontal and vertical Merged CSD Vectors.

4) *Character Ratio Feature (R)*: To improve the discrimination between characters that have similar number of segments and Barycentre position, we propose to compute

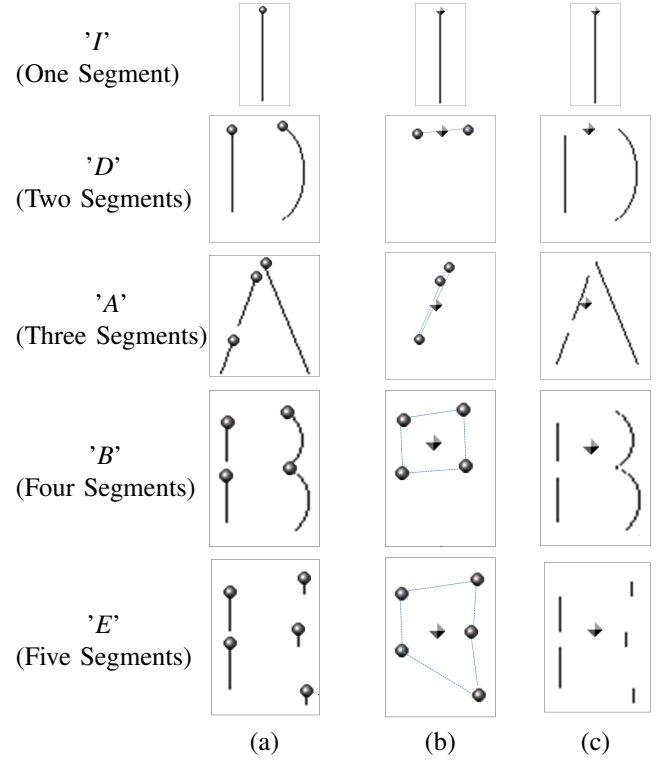


Fig. 4: Barycentre (B) computation process on different segment number letters: a. Starting pixels, b. Barycentre of polygon computation, c. Barycentre result

the Ratio between the height and width of the processed character bounding box. This feature is computed according to the Equation (5).

$$R = \frac{w}{h} \quad (5)$$

Figure 5 shows a Multiscale representation of characters 'A' and 'J'. As we can see, the Ratio feature (R) does not changes for the same character with different sizes (Equation (6)):

$$\begin{cases} R('A') = \frac{w_{00}}{h_{00}} = \frac{w_{01}}{h_{01}} = \frac{w_{02}}{h_{02}} = \frac{w_{03}}{h_{03}} = \frac{w_{04}}{h_{04}} = \frac{w_{05}}{h_{05}} \\ R('J') = \frac{w_{10}}{h_{10}} = \frac{w_{11}}{h_{11}} = \frac{w_{12}}{h_{12}} = \frac{w_{13}}{h_{13}} = \frac{w_{14}}{h_{14}} = \frac{w_{15}}{h_{15}} \end{cases} \quad (6)$$

However, in the same scale, the Ratio feature (R) allows to discriminate easily the two different characters 'A' and 'J' (Equation (7)):

$$\begin{cases} Scale_0 : [R('A') = \frac{w_{00}}{h_{00}}] \neq [R('J') = \frac{w_{10}}{h_{10}}] \\ Scale_1 : [R('A') = \frac{w_{01}}{h_{01}}] \neq [R('J') = \frac{w_{11}}{h_{11}}] \\ \vdots \\ Scale_5 : [R('A') = \frac{w_{05}}{h_{05}}] \neq [R('J') = \frac{w_{15}}{h_{15}}] \end{cases} \quad (7)$$

In fact, the Ratio feature (R) ensures a Multiscale invariance and increases the discrimination between characters in the

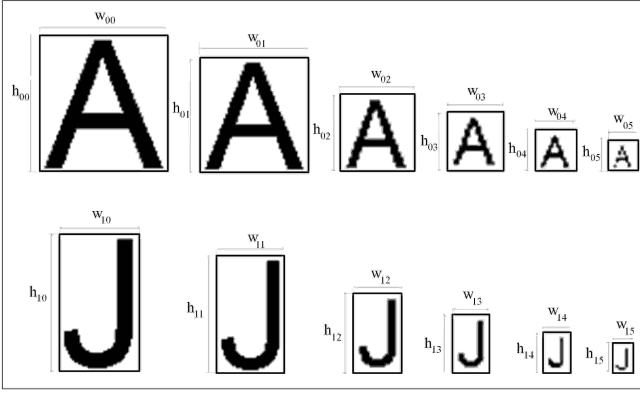


Fig. 5: Scale invariance of the character ratio feature of 'A' and 'J' letters with six different scales

same scale. Based on these extracted features from Horizontal and Vertical segments, the Unified Character Descriptor (UCD) is computed. Indeed, Horizontal and Vertical features are organized in a subsequent way in one single UCD vector as illustrated in Figure 6. Except the Barycentre x and y coordinates are computed according to the average of Horizontal and Vertical ones from the CSD vectors.

Number of Segments		Segment Position: Top, Middle, Bottom		Segment Position: Left, Middle, Right		Barycentre		Ratio
H-NBS	V-NBS	H-TMB	V-TMB	H-LMR	V-LMR	$\overline{Bx}$	$\overline{By}$	H-R
(a)	(b)	(c)	(d)					

Fig. 6: The Unified Character Descriptor (UCD): a. Character Segment Number Feature (NBS), b. Character Segment Position Feature (TMB, LMR), c. Character Barycentre Feature ($\overline{Bx}$, $\overline{By}$), d. Character Ratio Feature (R)

B. The Template Selection Stage

In our proposed method, the template selection stage is performed through type-face and font-size recognition. Indeed, even if the above UCD is scale invariant, it stills more powerful on small size ranges. To improve the Multiscale characteristic of our method, we propose to compute effectively the character font-size to choose the appropriate template to use. In addition, for an identified type-face character it is possible to make the OCR system handle a document with less effort. Following we detail our template selection proposal.

1) *Font-Size Recognition (x-h)*: Text line documents are composed of three typographical zones: the ascender, the x-height and the descender zones [16]. As shown in Figure 7, these zones are delimited by four virtual horizontal lines, Ascender, x-height, Base and Descender lines. The x-height

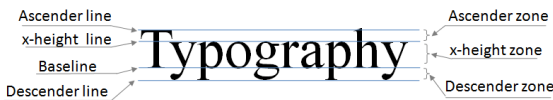


Fig. 7: Typography of Latin Alphabets

zone is the height of the body of lowercase letters referring to the distance between the baseline and the x-height one of lower-case letters in a typeface. For Latin script the Font-Size recognition is relatively an easy task [18][14]. In our proposal, we identify the character size with the help of the computation of the connected components x-h. Hence, we apply the following formula shown in Equation (8) [18].

$$x-h = \text{Baseline} - \text{x-height line} \quad (8)$$

2) *Type-Face Recognition (σ)*: Type-face recognition can give details on the structural and the typographical design of characters. Type-faces can be divided into two main categories: *serif* and *sans serif*. We mean by serifs the small features at the end of strokes within letters. Printed type-faces without serifs are known as *sans serif*. 'A' letter with *serif* and *sans serif* type-faces is shown in Figure 8.

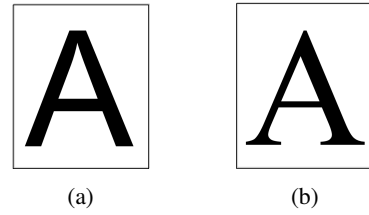


Fig. 8: 'A' letter with: a. *Sans Serif* and b. *Serif* type-faces

Actually, it is assumed that serif type-faces contain characters with moderate or dramatic difference between thick and thin strokes [17]. However sans serif characters are characterized with a low difference between stroke thicknesses. For this, we propose to study the Standard Deviation (σ) across the different character strokes (Equation (9)).

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu)^2}; \quad \mu = \frac{1}{N} \sum_{i=1}^N (x_i) \quad (9)$$

where x is the measured thickness in one character block, N is the number of character strokes and μ is the average of the N measured thicknesses; $i \in [1, N]$. To do this, we divide the character image generated by the layout analysis stage into 3x3 blocs.

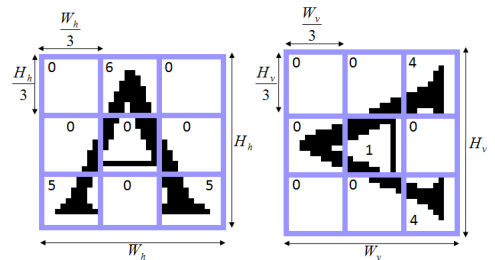


Fig. 9: Font recognition based on the splitting of characters in equal size blocks. Each block number refers to the median thickness of the considered non overlapping strokes

As shown in Figure 9, we determine in each block the thickness as the median of non overlapping lines with the block edges. During this process, vertical (stem, hairline, etc) and horizontal strokes (cross stroke, cross bar, etc) are checked. Very long or short lines (such as brackets, apex) that do not give a significant indication regarding the thickness are eliminated thanks to the 3x3 bloc division. As illustrated in Figure 10, the measured stroke thickness Standard Deviation (σ) is approximately constant between different Font-Sizes however it varies considerably between different type-faces.

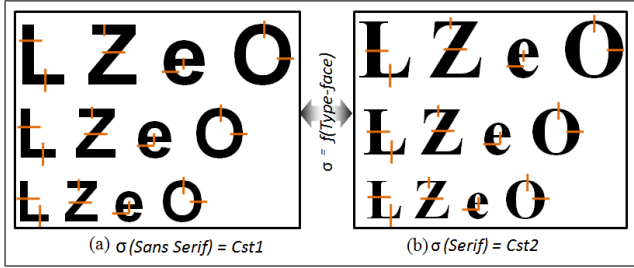


Fig. 10: Standard Deviation (σ) of stroke thickness in Multi-scale and Multifont characters. a. *Sans Serif* with no/moderate transition in the strokes, b. *Serif* with dramatic thick/thin transition in the strokes

C. The Decision Stage

The final stage of our work consists on matching the UCD vector of the input character to the different UCD vectors of the selected template (see Figure 2). The matching is performed by using the Sum of Absolute Differences (SAD) technique [12] distance process.

III. EXPERIMENTAL RESULTS

In this section, we evaluate firstly our proposed UCD descriptor regarding the robustness against the 'i' algorithm [9] descriptor when dealing with Multifont and Multisize constraints. Then, we evaluate our CR-UCD system against the well known Tesseract 3.02 OCR engine [20] on the same previously mentioned constraints.

A. The UCD Descriptor Evaluation based on OCR Accuracy

To evaluate the OCR-accuracy of our proposed UCD descriptor we use Multiscale and Multifont computer generated alphanumeric images containing number, upper-case, lower-case and special characters. For comparison reasons, we assess also the recently proposed 'i' Algorithm [9]. Table 1 shows a comparison between UCD and 'i' Algorithm descriptors using OCR Accuracy. In this evaluation, each character is matched with the same character font and size in the template. Obtained results demonstrate that our method outperforms the 'i' algorithm reaching an average of 99% of OCR accuracy on all assessed font-sizes and Type-faces. The main issues encountered by our description is the miss-recognition of

Table 1: OCR Accuracy of UCD and 'i' Algorithm descriptors using Multifont and Multisize template

Method	Type-face	Font-Size				
		18	24	36	72	90
'i' Algorithm	<i>Serif</i>	90	78	90	95	98
	<i>Sans Serif</i>	80	77	82	93	95
CR-UCD	<i>Serif</i>	100	100	100	100	100
	<i>Sans Serif</i>	98	98	98	98	98

similar characters like 'I' and 'l' in the different scales of the *sans serif* Type-face. In future work, lexical methods can be used in that case to enhance the performance of our description.

In Table 2 we illustrate the performance of UCD and the 'i' Algorithm descriptors using one single template defined by the *sans serif* type-face and the 90pt font-size for all compared character fonts and sizes. We can see that in this case the

Table 2: OCR Accuracy of UCD and 'i' Algorithm descriptors using one single font and size template

Method	Type-face	Font-Size				
		18	24	36	72	90
'i' Algorithm	<i>Serif</i>	12	20	25	59	98
	<i>Sans Serif</i>	15	14	20	50	95
CR-UCD	<i>Serif</i>	31	50	52	91	100
	<i>Sans Serif</i>	25	30	44	70	98

UCD still outperforming the 'i' algorithm descriptor in *serif* and *sans serif* fonts. Despite that our proposed descriptor gives a high accuracy on single type-face and font-size, the OCR Accuracy drops down when dealing with Multifont and Multiscale data. Hence, the usefulness of the template selection process in the CR-UCD System.

B. The CR-UCD System Evaluation on the LRDE Dataset

Following, we evaluate our CR-UCD proposal when processing on the LRDE-DBD dataset¹ [19] composed of 125 magazine documents. As shown in Figure 11, the LRDE-DBD includes different font-sizes, *serif* and *sans serif* type-faces.



Fig. 11: Sample of the LRDE-DBD documents

¹Copyright(c) 2012. EPITA and Development Laboratory (LRDE) with permission from Le Nouvel Observateur. LRDE-DBD is available on-line on the web site: [http : //www.lrde.epita.fr/cgi - bin/twiki/view/Olena/DatasetDBD](http://www.lrde.epita.fr/cgi-bin/twiki/view/Olena/DatasetDBD)

We show in Figure 12 the employed templates used in the Feature matching. We generate six templates categorized into two classes within *serif* and *sans serif* type-faces. Each category, includes three sub-templates consisting of three scale ranges: small, medium, and large character sizes.

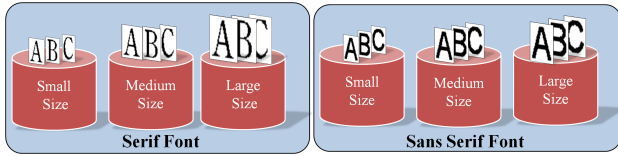


Fig. 12: The employed Multisize and Multifont templates

Small characters refer to the core paragraph text, medium characters refer to subtitles and large size characters are considered as the titles. Empirically fixed, the employed thresholds to distinguish font-sizes are given in Table 3.

Table 3: Employed x-height thresholds

Character sizes	x-height thresholds (pixels)
Small	0-30
Medium	30-55
Large	55-100

To show a relevant evaluation, we compare our proposed CR-UCD OCR system to the Tesseract [20] one. To make a fair comparison, we disabled the Tesseract dictionary option that we did not handle yet in our work. Figure 13 shows that the proposed CR-UCD method outperforms Tesseract one by reaching 1.5x higher OCR Accuracy on the 125 magazine documents of the LRDE Dataset.

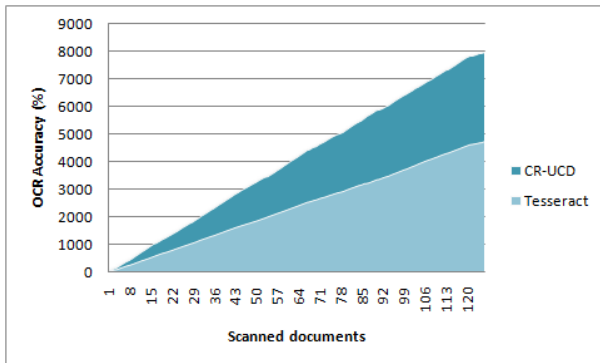


Fig. 13: OCR Accuracy evaluation of Tesseract and the proposed CR-UCD method

IV. CONCLUSION AND FUTURE WORK

In this paper, we proposed an efficient OCR system based on a new and robust description. Thanks to the used UCD descriptor, the template selection, and the employed fast feature matching our proposal offers a high OCR Accuracy compared to well known description and recognition methods. In future work, we intend to improve the proposed OCR efficiency

by using additional semantic rules to be able to differentiate similar characters like 'I' and 'l'. In addition, the template selection process could be enhanced to distinguish thick and italic characters. We plan also to use dictionary to enhance the final recognition results.

REFERENCES

- [1] J.R. Prasad, Template matching algorithm for Gujarati Character Recognition, IEEE 2nd International conference on Emerging Trends in Engineering and Technology, ICETET (2009)
- [2] Z.You-qing, A Recognition Method of Car License Plate Characters Based on Template Matching Using Modified Hausdorff Distance. IEEE International Conference on Computer, Mechatronics, control and electronic Engineering (CMDE) (2010)
- [3] F.Aghdasi, Automatic Licence Plate recognition system, IEEE AFRICON (2004)
- [4] J.Jiao, A configurable method for multi-style license plate recognition. Pattern Recognition, Vol: 42, No: 3, pages 358-369 (2009)
- [5] M.M Farhad, An efficient Optical Character Recognition Algorithm using Artificial Neural Network by Curvative Properties. IEEE International conference on informatics, electronics & vision (2014)
- [6] M.soua, R.Kachouri, A.Akil. A new hybrid binarization method based on Kmeans, ISSCP 2014.
- [7] S.Kopf, T.Haenselmann, W.Effelberg. Robust Character Recognition in Low-Resolution Images and videos.
- [8] A. Antanacopoulos, C. Clausner, C. Papadopoulos, and S.Platschacher. ICDAR2013 Competition on Historical Newspaper Layout Analysis., International Conference on Document Analysis and Recognition, 2013
- [9] S.Sharstry, Gumasheela, T.DUTT, D.S.Vinay. "i" A novel algorithm for optical character recognition (OCR), 2013.
- [10] R. Smith, "Hybrid Page Layout Analysis via Tab-Stop Detection", ICDAR '09 Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, Pages 241-245, 2009
- [11] R. Smith, "An Overview of the Tesseract OCR Engine", Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR), IEEE Computer Society, 629-633, 2007.
- [12] K.Lengwehasarit, "Probabilistic partial-distance fast matching algorithms for motion estimation", pp 139 - 152. Circuits and Systems for Video Technology, IEEE Transactions on (Volume:11, Issue: 2), 2002.
- [13] M.Soua, R.Kachouri, M.Akil. GPU parallel implementation of the new hybrid binarization based on Kmeans method (HBK) Journal of Real-Time Image Processing, Springer, 2014
- [14] Q.Akram, S.Hussain, Z.Habibt, Font-Size Independent OCR for Noori Nastaleeq.
- [15] http://www.infor.uva.es/~descuder/docencia/IG/letterform_anatomy.pdf
- [16] Z. Abdelwahab and R. Ingold, Optical Font Recognition Using Typographical Features, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp.877-882, 1998
- [17] <http://www.sitepoint.com/the-old-style-type-face/>
- [18] Souza, M. ; Naoi, S. ; Suen, C.Y. A. Automatic filter selection using image quality assessment, 2003
- [19] G. Lazzara, T. Géraud, "Efficient Multiscale Sauvolas Binarization", International Journal on document analysis and recognition, 2013
- [20] R. Smith, An Overview of the Tesseract OCR Engine. International Conference on Document Analysis and Recognition, 2007.

Annexe E

Unsupervised Image Segmentation Based on Localpixel Clustering and Low-Level Region Merging

R. Kachouri, M. Soua, , M. Akil.

2nd IEEE International Conference on Advanced Technologies for Signal and Image Processing, Monastir (Tunisie), Mars 2016

Unsupervised Image Segmentation Based on Local pixel Clustering and Low-Level Region Merging

Rostom Kachouri
rostom.kachouri@esiee.fr

Mahmoud Soua
soua.mahmoud@esiee.fr

Mohamed Akil
mohamed.akil@esiee.fr

Université Paris-Est, Laboratoire d'Informatique Gaspard-Monge, Equipe A3SI, ESIEE Paris, France

Abstract—Heterogeneous image segmentation is one of the most important tasks in image processing. It consists in partitioning the image into a set of disjoint regions. In this paper, we propose a new unsupervised image segmentation method that we call Unsupervised Image Segmentation (UIS). Our proposal performs an efficient image partition efficiently into primitive regions. This process is ensured by a local adaptive Kmeans and a novel centroids initialization. Then, similar sets are agglomerated to form homogeneous regions. For that, a low-level feature merging is employed according to a hierarchical linkage approach. Finally, in case of over-segmentation, appearing outlier regions are removed using a post process stage. Therefore, the UIS method allows to determine automatically the image region number. Indeed, it extends the Kmeans clustering to obtain meaningful regions. Several experiments were conducted using two heterogeneous image datasets. A comparison with well-known segmentation methods was also performed using the Liu's factor measure.

Keywords—Region based image segmentation, Kmeans clustering, Region agglomeration, Similarity measure, Color, Texture, Over-segmentation, Outlier region.

I. INTRODUCTION

Heterogeneous image segmentation is a fundamental processing for image description and recognition tasks. It aims to partition the image into homogeneous regions. Subsequently, pixels are assigned to semantic meaningful objects. Several survey papers [1], [2] cover the major image segmentation algorithms providing high quality results. These algorithms can be categorized into two main approaches [3]: Boundary-based and Region-based methods. The first category [4] operates on the image edges. Indeed, edges are detected and then linked into contours representing the object boundaries [5]. Edge-based methods are very robust handling properly images with significant contrast variation between regions. However, the presence of small gaps in edge boundaries allows merging of dissimilar regions. In addition, they are sensitive to noise which leads often to generate non closed contours [6], [5]. The second category, representing region-based methods, is more noise immune [5] and produces enclosed regions [7]. Methods of this category operate by partitioning groups of dissimilar pixels regarding one or more features such as brightness, color, texture, etc. Then, similar pixels which are neighbors, are grouped together in order to reaches a set of meaningful regions.

As a well known region-based method, Region growing [8] enlarges gradually areas around starting pixels (seeds) based on a predefined criteria. Nevertheless, in this method

user should choose initial seeds within the most homogeneous areas. Hence the inconvenience of this method is the choice of initial seeds. So that obtained results are not always optimal. Unlike Region-growing, the Split and Merge technique [9] generates automatically an initial partition of homogeneous regions. Several methods were proposed in this context [10], [29], [30]. For example, the Split and Merge based-quadtrees method [10] performs the split phase by assuming that the image is an entire block. Then regarding on homogeneity test. Heterogeneous blocks are divided into four sub-blocks. The treatment is stopped for each sub-block which reach homogeneity. In the merge phase, regions are gathered based on predefined similarity criterion and a Region Adjacency Graph (RAG). The main advantage of the Split and Merge methods is the fine subdivision of the image, however, they remain complex due to the tedious split treatments. Otherwise, methods based on pixel clustering, offer less complexity in region-based image segmentation [11], [12], [42]. More recent developments in this field [44] classify the input image pixels into multiple clusters based on their distance from each other [11], membership function [13], maximum likelihood criterion [14], etc. In this context, one of the most used methods for color image segmentation is the clustering algorithm Kmeans [16], [31]. However, the major drawback of methods using this classification algorithm is that user needs to set, in the beginning, the region number. Furthermore, in the agglomeration stage of these methods, different linkage techniques [17] can be employed. In the single linkage technique [17], the cluster similarity is defined by the shortest distance from any member of a cluster to any member of another one. This method shows a total insensitivity to shape and size of clusters. However it is sensitive to outlier pixels [18]. Another technique is the complete linkage [17]. It is based on the computation of the greatest distance from any member of a cluster to any member of another cluster. This one is not strongly affected by outlier pixels. However, the complete linkage method can break large clusters, and has trouble with convex shapes [18]. The level of complexity may be reduced by using centroid linkage technique [17]. Indeed, the distance between two clusters is represented by the distance between their centroids. Whenever two clusters are combined, a new centroid is computed. Nevertheless, it can fail in case of complicated cluster shapes. Indeed, centroids, in such case, do not represent efficiently the clusters [18]. It is obvious that each method has its own limitations and advantages. Recent works, try to combine more than one linkage technique to improve the agglomeration results [19], [20], [21].

In this paper, a novel region-based segmentation method is proposed. Firstly, the proposed method ensures an unsupervised split stage based on local Kmeans clustering. Obtained primitive regions are merged based on a combination of color and texture features. Different linkage techniques are employed to gather efficiently the similar regions. In case of over-segmentation, a post process is used to suppress the unwanted regions. In fact, our main contributions consist on a offering a full automatic segmentation method. A novel initialization method for the local Kmeans algorithm is proposed. Moreover, different linkage techniques are combined in the merge phase.

In the following, we describe our proposal in section 2. Next, in section 3, obtained results are shown and discussed. Finally conclusion is drawn in section 4.

II. PROPOSED METHOD: UNSUPERVISED IMAGE SEGMENTATION (UIS)

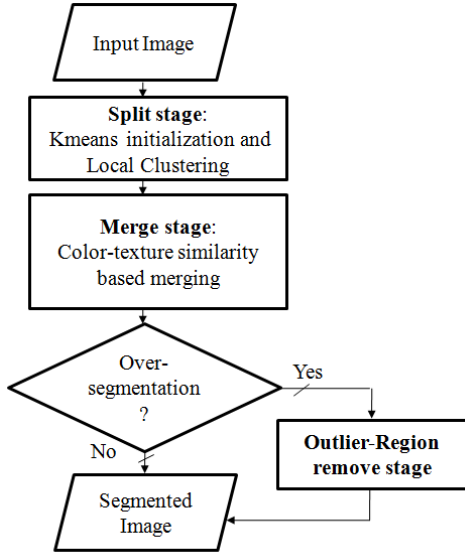


Figure 1: Flow diagram of the proposed Unsupervised Image Segmentation (UIS) method

In this section, we describe the different stages of our Unsupervised Image Segmentation (UIS) proposal. As shown in Figure 1, two main stages are employed. The first one is the split stage where a window based approach is used to perform a local robust Kmeans clustering [11]. For this end, we divide the image into equal size blocks. The second stage is the merge phase. It consists in agglomerating the regions with both color and texture similarity measure. Many color spaces can be employed. In this work we use the Lab [22] color space for better segmentation results [45]. To compute distance between pixels in the image, the Euclidian distance [23] is used. After the merge stage, outlier region remove could be employed in case of over-segmentation. Finally, regions in the segmented image are shown with the color of their centroids to show a more expressive semantic regions. In next subsections, we describe, with more details, our proposed method stages.

A. Split stage

The first stage of our proposal consists in clustering local image blocks using the Kmeans algorithm [11]. Actually, we employ small data blocks, where the block Width and Height represent respectively a ratio of 25% from the image Width and Height. According to the literature [36] this size ensures that each block have high rate to contain one single semantic meaningful objects. However, it can contain more objects. Actually, we believe that this block size is enough to form two meaningful regions. This region number is shown to be appropriate by our experiments. Therefore, we initialize the local Kmeans with a set of two centroids, noted by $C_{bl} = \{C_{bl,0}, C_{bl,1}\}$, with $bl \in [0, Nb - 1]$ and Nb is the total number of blocks in the image. We consider for that nine pixels in each block as a potential initial centroids. These pixels are located in the border and the middle of the blocks as shown in Figure 2.

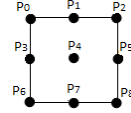


Figure 2: Position of the nine selected pixels in each block

Actually, among these nine selected pixels, we consider that, at least, one pixel belong to the second region if it exists. Then, based on the used color space, we initialize the local Kmeans in each block with the two most farthest pixels from the nine selected ones. After initialization, each block is clustered with Kmeans until convergence. We note by $R = \{R_0, R_1, \dots, R_{k-1}\}$ the set of the obtained regions within all blocks in the image. k is then the total number of regions, and it is equal to at most $2 \times Nb$ when each block contains two regions. Generally, $k \leq 2 \times Nb$. the UIS method gives a robust clustering thanks to its good initialization and to the efficient Lab color space use. In the next, we describe the Region Merging stage.

B. Merge stage

In this stage, the employed similarity criterion, are very important to determine which region is similar to which one. In general, the combination of color and texture features provides better segmentation result [37]. The advantages of both color and texture based segmentation are quite well preserved: we obtain sharp boundaries and homogeneous regions [32]. Actually, segmentation purely based on texture gives fuzzy boundaries but usually homogeneous regions [28]. Whereas, segmentation based on color is more sensitive to local variations in color but provides sharp boundaries [27]. Many researcher proposed various approaches for texture analysis [40], for color analysis [39] or both of them [37], [38]. In our work, we use both color and texture features in addition to the adjacency criterion.

1) *Color Features*: We consider that two regions can be merged only if they are adjacent, and have similar color. For that, as shown in Figure 3, we extract for each region three features: $R_{i,c}$, $R_{i,dmin}$, $R_{i,dmax}$ with $i \in [0, k - 1]$. Where, $R_{i,c}$ is the computed centroid of the region R_i . It defines the

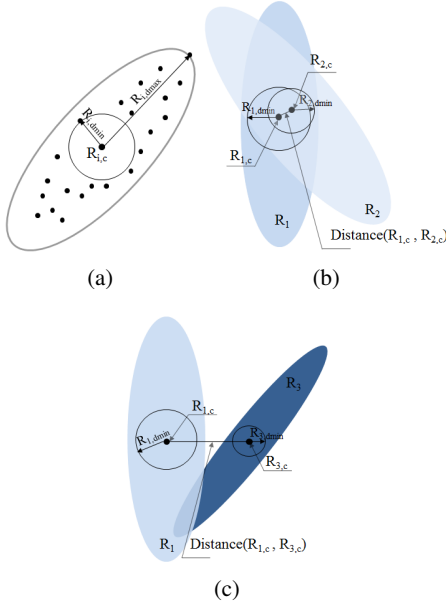


Figure 3: Region representation in the Lab color space: a. Basic model, b. Similar regions and c. Dissimilar regions

median of region colors. $R_{i,dmin}$ and $R_{i,dmax}$ are respectively the smallest and the largest color variations in the region R_i (see Figure 3(a)). To decide if two regions R_i and R_j , $i, j \in [0, k - 1]$ are similar (Figure 3(b)) or not (Figure 3(c)), we compare the Euclidean distance $Distance(R_{i,c}, R_{j,c})$ between their respective centroids $R_{i,c}$ and $R_{j,c}$ to a given threshold, as stated in Equation 1.

$$Distance(R_{i,c}, R_{j,c}) \leq Threshold \quad (1)$$

The idea is that the applied threshold is variable depending on the two considered regions, through the use of the $Max(R_{i,dmin}, R_{j,dmin})$ term. As illustrated in Figure 3, we can see that the two regions R_1 and R_2 (Figure 3(b)) are similar since $Distance(R_{1,c}, R_{2,c}) \leq Max(R_{1,dmin}, R_{2,dmin})$. However R_1 and R_3 (Figure 3(c)) are dissimilar because $Distance(R_{1,c}, R_{3,c}) > Max(R_{1,dmin}, R_{3,dmin})$.

2) *Texture Features*: We define a coefficient that we call TD (see Equation 2) to consider the Texture Degree of the processed image. To compute TD , we evaluate the ratio between Img_{dmax} and Img_{dmin} as shown in Equation 2.

$$TD = \frac{Img_{dmax}}{Img_{dmin}} \quad (2)$$

Where Img_{dmin} and Img_{dmax} are the average distance of, respectively, all the $R_{i,dmin}$ and all the $R_{i,dmax}$ in the image Img , $i \in [0, k - 1]$. The final employed threshold is then given by the Equation 3:

$$Threshold = Max(R_{i,dmin}, R_{j,dmin}) + TD \quad (3)$$

with $i, j \in [0, k - 1]$. So, the TD term allows to increase the final threshold value depending on the texture degree of the processed image. Indeed, we note that in low textured image, $\forall i \in [0, k - 1]$, $R_{i,dmax}$ tends to be equal to $R_{i,dmin}$ ($R_{i,dmax} \simeq R_{i,dmin} \Rightarrow TD \simeq 1$). Otherwise, high textured image, $R_{i,dmax}$ is highly greater than $R_{i,dmin}$ $\forall i \in [0, k - 1]$, ($R_{i,dmax} \gg R_{i,dmin} \Rightarrow TD \gg 1$).

C. Outlier-Region remove stage

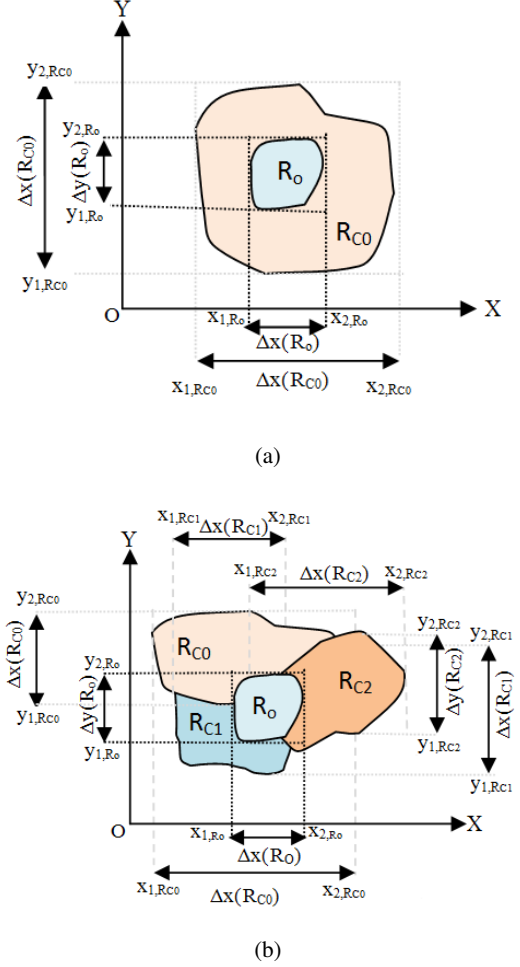


Figure 4: The two cases of region locations a. Region R_0 totally enclosed in a region R_{C0} . b. Region R_0 partially included in three regions R_{C0} , R_{C1} and R_{C2} .

It is possible that the merged image contains some outlier regions. Generally, in such case, we talk about over-segmentation. To overcome this problem, we eliminate these small areas by merging them with the appropriate regions. Generally, considering the context of the application, meaningful objects in an image exist within a marge of sizes. Then, it is possible to impose a minimum valid object area for segmented regions [33]. Hence, researchers set the maximum pixel number for a valid object region [33], [34], [35]. In our work, we extract objects in a general purpose context

(natural scene, single object). For this, we evaluate the sum of pixel number across the regions from the largest to smallest ones. When the pixel number reaches 60% from the total pixel number, the remaining areas will be considered as outliers. These regions are merged with the bounding ones. We note by (x_{1,R_i}, x_{2,R_i}) and (y_{1,R_i}, y_{2,R_i}) the horizontal and vertical limits of regions R_i . For each R_i , we note by Δ_X and Δ_Y the measures referring respectively to their Width and Height. They are defined as following: $\Delta_X(R_i) = x_{2,R_i} - x_{1,R_i}$, $\Delta_Y(R_i) = y_{2,R_i} - y_{1,R_i}$ with $i \in [0, Nb]$ and Nb is the total number of regions. The outlier suppression is shown in Figure 4 and Algorithm 1. Following, we note by O the origin, R_o an outlier region and R_{Ci} $i \in [0, Nb]$ the set of the candidate bounding ones. We study the measures Δ_X and Δ_Y to determine if a given region R_{Ci} bounds R_o . Two conditions rise up. Firstly, as shown in Figure 4(a), R_o can be totally bounded by an R_{Ci} (Line 4). In this case, these segments are merged together. Secondly, R_o can partially be bounded by several regions R_{Ci} (Line 7). Then, R_o is merged with the most color similar one. This case is illustrated in Figure 4(b).

Algorithm 1 Outlier-Region remove stage

```

1: for  $i := 0$  to  $k - 1$  do
  //  $R_o$  and  $R_{Ci}$  overlapping
2:   if  $(x_{1,R_o} > x_{1,R_{Ci}})$  or  $(x_{2,R_o} < x_{2,R_{Ci}})$  or
3:    $(y_{1,R_o} > y_{1,R_{Ci}})$  or  $(y_{2,R_o} < y_{2,R_{Ci}})$  then
  //  $R_o$  is totally enclosed in  $R_{Ci}$  (Figure 4.a)
4:     if  $(\Delta_X(R_o) \leq \Delta_X(R_{Ci}))$  and
5:      $(\Delta_Y(R_o) \leq \Delta_Y(R_{Ci}))$  then
6:       Merge  $R_o$  and  $R_{Ci}$ 
  //  $R_o$  is partially included in  $R_{Ci}$  (Figure 4.b)
7:   else
8:     Merge  $R_o$  with the most color similar region  $R_{Ci}$ 
9:   end if
10: end if
11: end for

```

As shown in Figure 5, outlier regions designed by arrows (Figure 5(a)) are correctly merged with the bounding and color similar regions in the image (Figure 5(b)).

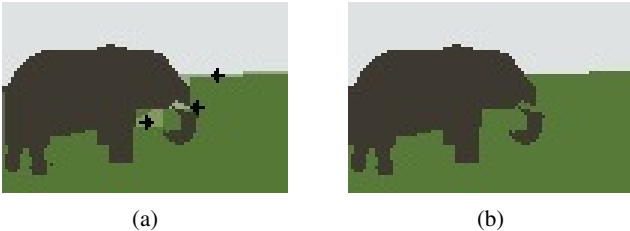


Figure 5: Outlier-Region remove: a. Merged Image, b. Final UIS Segmented Image

III. EXPERIMENTAL RESULTS

In the following, we evaluate our proposed UIS method on both Corel¹ [24] and Berkeley² [25] datasets. Evaluation is performed according to visual accuracy comparison and Liu's factor measuring.

A. The Liu's factor metric

We employ the Lius'factor metric [26] to evaluate the obtained results with our UIS method. The following equation shows the Lius'factor formula.

$$L(Img) = \sqrt{k''} \times \sum \frac{\xi_i^2}{\sqrt{\mathcal{S}(R_i)}}, i \in [0, k - 1] \quad (4)$$

where, Img is the segmented image, We note respectively k'' the final obtained number of regions and $\mathcal{S}(R_i)$ the size of the region R_i , ie its total number of pixels. ξ_i is the sum of the Euclidean distance of each pixel in the region R_i between the segmented image and the original one. It defines the region color error. The best image segmentation results are indicated with a small values of L . Generally, images with large color error regions or small regions are penalized.

B. UIS evaluation based on Visual Accuracy and Liu's factor measuring

The Figure 6 illustrates the obtained UIS segmentation results on five image samples from different Corel categories (Figure 6(a)).

As a first observation, it is clear that our UIS method gives efficient segmentation quality. Indeed, it provides the right region delimitation and the correct region number for all illustrated images. Actually, the local Kmeans employment produces a robust clustering by dividing the original image into k primitive regions. In addition, we observe that region contours are well preserved after the split stage (Figure 6(b)). According to Figure 6(c), the merge stage decreases efficiently the region number by connecting similar regions as explained in section 2.2. The obtained region number in this stage is noted by k' . At this level, the final segmentation result may be obtained as it is the case of the Dinosaur image. Otherwise, in case of over-segmentation, the outlier regions are suppressed efficiently (Figure 6(d)). k'' is the final number of regions in the obtained segmented image. Following, we show that we improve the segmentation quality when eliminating the outlier regions. According to the table below, the Liu's coefficient is enhanced in the final segmented image compared to the merged one. Indeed, the less are the image segments small, the lower is the Liu's coefficient and that is prove the efficiency of the UIS method.

For comparison reasons, we evaluate the proposed UIS with the state of the art methods on the well-known Berkeley dataset. Figure 7 illustrates segmentation results of Normalized cuts (NCUTS) [42], Dominant Set (DSet) [43], DSet-DBSCAN [44] and our proposed UIS. Results of the compared methods are taken from the literature [44]. We observe that

¹Corel dataset could be downloaded at: <http://savvash.blogspot.fr/2008/12/benchmark-databases-for-cbir.html>

²Berkeley dataset is available at the web site: <https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds>

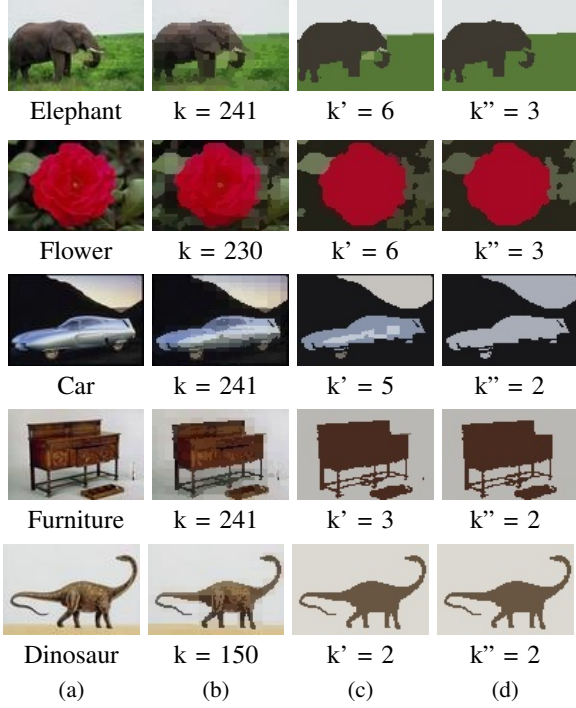


Figure 6: Segmentation results and obtained region number on the Corel dataset: a. Original Image Samples, b. Local Kmeans splitted Images, c. Merged Images and d. UIS Segmented Images

Table I: Liu's Factor on Corel images

Image / Stage	Merged Image	Segmented Image
Elephant	1.53×10^{-6}	7.80×10^{-7}
Flower	1.52×10^{-6}	9.06×10^{-7}
Car	1.33×10^{-6}	5.39×10^{-7}
Furniture	1.44×10^{-6}	6.51×10^{-7}
Dinosaur	4.88×10^{-7}	4.88×10^{-7}

NCUTS is not able to avoid the problem of partitioning coherent regions into multiple segments. The same problem is encountered with the DSet method. This because it generates only convex shapes. In the other side, DSet-DBSCAN and UIS can partition the images into meaningful segments. Actually, DSet-DBSCAN extends the clustered DSet regions with the DBSCAN algorithm. Therefore it can handle non-convex shapes but still generating over-segmentation with the appearance of few outlier regions on the final results (Figure 7(c)). Nevertheless, the proposed UIS method processes as well non-convex shapes thanks to the hierarchical merge. In addition, the powerful outlier remove stage decreases considerably the amount of overlapping regions.

Following, we compare our segmentation results with, the Split and Merge [41], Watershed [8] and Verma [25] methods, using the Liu's factor metric. As shown in Figure 8, our proposed method gives $3\times$ lower Liu's factor values than Watershed, $2\times$ lower values than Verma and $1.5\times$ lower values than the Split and Merge method.

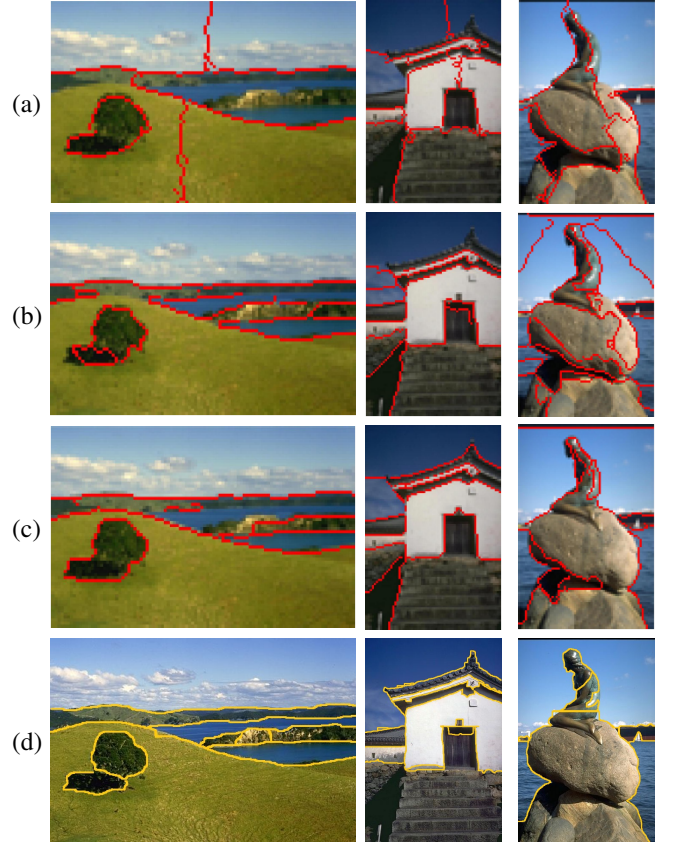


Figure 7: Segmentation result comparison on the Berkeley dataset: a. NCUTS, b. DSet, c. DSet-DBSCAN and d. Proposed UIS

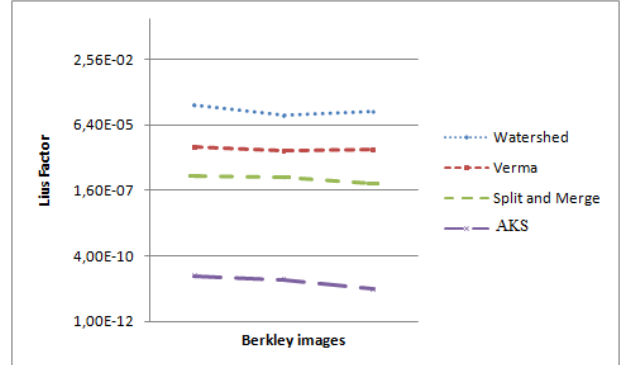


Figure 8: Liu's factor-based state of the art segmentation method evaluation on Berkeley dataset

IV. CONCLUSION

Image segmentation is a sensitive key point in image processing. A new automatic and full unsupervised image segmentation method was proposed in this paper. Several experiments are performed using Corel and Berkeley image datasets. We show that the UIS provides robust image segmentation results and determines correctly image region number. Conducted results, prove that UIS outperforms state of the

art methods on visual accuracy comparison and Liu's factor measuring.

REFERENCES

- [1] Luccheseysz, L., and S.K. Mitray. "Color image segmentation: A state-of-the-art survey", Image processing, Vision, and Pattern Recognition 67.2, 207-221, 2001.
- [2] N. R. Pal, S.K. Pal, A review on imagesegmentation techniques, Pattern Recognition, Vol. 26(9), pp. 12771294, 1993.
- [3] R. M. Haralick, L.G. Shapiro, Survey: image segmentation techniques, Comput. Vision, Graphics Image Process., Vol. 29, pp. 100132, 1985
- [4] Ho, S.Y., Lee, K.Z. Design and analysis of an efficient evolutionary image segmentation algorithm. J. VLSI Signal process. 35, 29-42, 2003.
- [5] Sameena Banu, The Comparative Study on Color Image Segmentation Algorithms, International Journal of Engineering Research and Applications, Vol. 2, Issue 4, pp.1277-1281, 2012
- [6] Huet, F. and Philipp-Foliguet, S. , Etude de contours haute-chelle pour la segmentation et la fermeture de contours en prsence de zones textures et ou brutes, Traitement du Signal, vol. 15, no. 8, 3-16 ,1998
- [7] Leila M. Garcia Fonseca, Fernando Mitsuo Ii, Satellite imagery segmentation: a region growing approach, in VIII Brazilian Symposium on Remote Sensing, 677-680, 1996
- [8] Jun Tang, Color Image Segmentation algorithm Based on Region Growing. International Conference on Computer Engineering and Technology. Vol 6, 634-637, 2010
- [9] S.L. Horowitz and T. Pavlidis, Picture Segmentation by a Directed Split and Merge Procedure, Proc. ICPR, Denmark, pp.424-433, 1974.
- [10] Kelkar, D., Gupta, S., Improved Quadtree Method for Split Merge Image Segmentation, International Conference on Emerging Trends in Engineering and Technology, 44-47, 2008
- [11] LLOYD, S. P.: *Least square quantization in PCM*. In: *IEEE Transactions on Information Theory* vol.28, no.2, 129-137, (1982)
- [12] Banerjee, B. , Surender, V.G. ; Buddhiraju, K.M., "Satellite image segmentation: A novel adaptive mean-shift clustering based approach" , PGeoscience and Remote Sensing Symposium (IGARSS), IEEE International, 4319-4322, 2012.
- [13] Yong Yang, Image Segmentation By Fuzzy C-Means Clustering Algorithm With A Novel Penalty Term, Computing and Informatics, Vol. 26, 1731, 2007
- [14] Dempster, A. P.Laird, N. M.Rubin, D. B.: Maximum-Likelihood from In complete Data Via the EM Algorithm. J. Roy. Statist. Soc., Vo l. B39, 1077, pp. 138
- [15] Ambrose, C.Govaert, G.: Convergence of an EM-Type Algorithm for Spatial Clustering. Pattern Recognition Letters, Vol. 19, 1998, pp. 919927
- [16] Ms.Chinki Chandhok ,Mrs.Soni Chaturvedi, Dr.A.A Khurshid, "An Approach to Image Segmentation using Kmeans Clustering Algorithm (IJIT), 1(1),2012.
- [17] J. Bien and R. Tibshirani, Hierarchical Clustering with Prototypes via Minimax Linkage, Journal of the American Statistical Association, 2011, vol. 106, no 495, pages 1075-1084 2011
- [18] J.A.S. Almeida and al., Improving hierarchical cluster analysis: A new method with outlier detection and automatic clustering, Chemometrics and Intelligent Laboratory Systems 87, 208-217, 2007.
- [19] G.M. Downs, J.M. Barnard, Clustering methods and their uses in computational chemistry, in: K.B. Lipkowitz, D.B. Boyd (Eds.), Reviews in Computational Chemistry, vol. 18, Wiley, United Kingdom, pp.1-40, 2002
- [20] S. Guha, R. Rsatogi, K. Shim, Cure: an efficient clustering algorithm for large databases, Inf. Syst. 26(1),35-58, 2001
- [21] Han, M. Kamber, Data Mining: Concepts and Techniques, 2nd edition, Kaufmann, Ch. 7, 2006
- [22] L. Shamir, Human Perception-based Color Segmentation Using Fuzzy Logic, <http://www.phy.mtu.edu/~lshamir/publications/flcolor.pdf>
- [23] S. Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions", International Journal of Mathematical Models and Methods in Applied Sciences, 1(4),300-307, 2007
- [24] Wang, James Ze, Jia Li, and Gio Wiederhold. "SIMPLcity: Semantics-sensitive integrated matching for picture libraries." Pattern Analysis and Machine Intelligence, IEEE Transactions on 23.9,: 947-963, 2001
- [25] Verma, O.P. and al., A Simple Single Seeded Region Growing Algorithm for Color Image Segmentation using Adaptive Thresholding, CSNT, 2011, 500-503, 2011
- [26] Jianqing Liu , Yee-Hong Yang, "Multiresolution Color Image Segmentation IEEE Transaction on pattern analysis and machine intelligence Vol no. 7, JULY 1994.
- [27] Mirmehdi M, Xie XH, Suri J. Handbook of Texture Analysis. Singapore: World Scientific, 2008.
- [28] Unser M. Texture classification and segmentation using wavelet frame. IEEE Trans Image Process 1995; 4: 15491560
- [29] Gui Yufeng and al., A split and merge algorithm based on grey theory, IEEE conference, 2011.
- [30] Karim, Z.,Pattern based object segmentation using split and merge, IEEE conference, 2009.
- [31] Neelambike S,Parashuram Baraki,Pattern based object segmentation using split and merge, International Journal of Advanced Research in Computer Science and Technology, 2014.
- [32] Piotr Gut and al., Edge-Based Robust Image Registration for Incomplete and Partly Erroneous Data. In: Computer Analysis of images and patterns, 2001
- [33] Cludio Rosito Junga, Jacob Scharcanskib., Robust watershed segmentation using wavelets. In: Image and Vision Computing, Volume 23, Issue 7, Pages 661669, 2005.
- [34] Xi-Wen Zhang and al., Extraction of karyocytes and their components from microscopic bone marrow images based on regional color features. In: Pattern Recognition, Volume 37, Issue 2, Pages 351361, 2004.
- [35] Mrudula Karande and Prof. D. B. Kshirsagar, Probabilistic Model Based Image Segmentation. In: The International Journal of Multimedia and Its Applications (IJMA) Vol.6, No.2, April 2014.
- [36] M. Celenk, "Hierarchical color clustering for segmentation of textured images," ssst, pp.483, 29th Southeastern Symposium on System Theory (SSST '97), 1997
- [37] Dengsheng Zhang, Improving Image Retrieval Performance by using both Color and Texture features, Proceedings of the Third International Conference on Image and Graphics, 2004, pp.172
- [38] H. Kavitha, M.V.Sudhamani, Content Based Image Retrieval Based on Global and Region Content of an Image, International Journal of Advanced Computing and Communication Systems, vol.1 Issue.1, 2014
- [39] Stachowicz, M.S., Lemke, D., Image segmentation and classification using color features, Video/Image Processing and Multimedia Communications on VIPromCom, pages 57-64, 2002
- [40] Lucas Bastos and al. ,Automatic Texture Segmentation Based on k-means Clustering and Co-occurrence Features, Systems, Signals and Image Processing,. Pages: 141-144, 2008
- [41] Dirk Balthasar, [http : //www.uni - koblenz.de/ lb/lb_downloads/](http://www.uni-koblenz.de/lb/lb_downloads/), 2002
- [42] J.Shi and J.Malik, "Normalized cuts and image segmentation" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22,no.8,pp. 167-172, 2000.
- [43] M.Pavan and M.Pelillo, "Dominant sets and pairwise clustering," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 29, no. 1,pp. 167-172, 2007.
- [44] Jian Hou, Qi Xia, "Merging dominant sets and DBSCAN for Robust clustering and image segmentation", IEEE International Conference on Image Processing, (ICIP), 2014.
- [45] Oliver, A. and al. , Improving Clustering Algorithms for Image Segmentation using Contour and Region Information,Automation, Quality and Testing, Robotics, IEEE International Conference on (Vol:2), 315-320, 2006

Liste de publications

Articles dans des revues internationales

- Mahmoud Soua, Rostom Kachouri, Mohamed Akil (2014). GPU parallel implementation of the new hybrid binarization based on Kmeans method (HBK). Journal of Real-Time Image Processing, Springer. Pages 1-15
- Mahmoud Soua, Rostom Kachouri, Mohamed Akil (2016). Efficient GPU implementation of the Improved Hybrid Binarization based on Kmeans classification and gamma correction methods (I-HBK). EURASIP Journal on Advances in Signal Processing (en soumission).

Communications internationales

- Mahmoud Soua, Rostom Kachouri, Mohamed Akil (2014). A new hybrid binarization method based on Kmeans. In : the 6th IEEE Int'l Symposium on Communications, Control, and Signal Processing, 21-23 Mai, Athene, Grèce, ISCCSP'2014.
- Mahmoud Soua, Rostom Kachouri, Mohamed Akil (2015). Improved Hybrid Binarization based on kmeans for Heterogeneous document processing. In : the 9th IEEE Int'l Symposium on Image and Signal Processing and Analysis, 7-9 Septembre, Zagreb, Croatia, ISPA'2015.
- Mahmoud Soua, Rostom Kachouri, Mohamed Akil (2015). Efficient Multiscale and Multifont Optical Character Recognition System based on robust Description. In the 5th IEEE Int'l Conference on Image Processing Theory, Tools and Applications, 10-13 Novembre, Orleans, France, IPTA'2015.
- Rostom Kachouri, Mahmoud Soua, Mohamed Akil (2016). Unsupervised Image Segmentation Based on Localpixel Clustering and Low-Level Region Merging. 2nd IEEE International Conference on Advanced Technologies for Signal and Image Processing, Monastir (Tunisie), Mars 2016.

Bibliographie

- [1] - M.Soua, R.Kachouri, M.Akil. Improved Hybrid Binarization based on kmeans for Heterogeneous document processing. In : the 9th IEEE Int'l Symposium on Image and Signal Processing and Analysis, 7-9 Septembre, Zagreb, Croatia, ISPA'2015.
- [2] J.Mantas. An Overview of character recognition methodologies. Pattern Recognition, Vol 19, n 6, pp 425-430 (1986)
- [3] V.Govindan and GT.Candela. Comparison of hand printed digit classifier. National Institution of Standard and technologie. Image recognition Group. (1993)
- [4] ISO 1831 :1980, Printing Specification for Optical Character Recognition
- [5] Amit Choudhary , Rahul Rishi, A Fused Feature Extraction Approach to OCR : MLP vs. RBF, ICT and Critical Infrastructure : Proceedings of the 48th Annual Convention of Computer Society of India- Vol I Volume 248 of the series Advances in Intelligent Systems and Computing pp 159-166
- [6] The Viterbi algorithm, GD Forney Jr - Proceedings of the IEEE, 1973.
J.F.Jodouin. Les réseaux neurométriques. Collection Hermes. (1994)
- [7] A.Leturg. La reconnaissance des caractères manuscrits sans apprentissage. Thèse de doctorat. Paris VI. (1990)
- [8] L.Heute. Reconnaissance de caractères manuscrits : Applications à la lecture automatique des chèques et des enveloppes postales, THèse de doctorat, Rouen, (1994).
- [9] J.C.Simon. La reconnaissance des formes à l'épreuve des faits. La recherche n 312, pp 975-985, 1992.
- [10] Agne S., Rogger M., Benchmarking of Document Page Segmentation, Part of the IS and T Conference on Document Recognition and retrieval (VII, San Jose) california, january 2000, pp 165-171.
- [11] A.Belaid, L.pierron, L.Nadjman, D.Reynen. La numérisation de documents : Principe et évaluation des performances.
- [12] Trier, I., Jain, A., and Taxt, T. Feature extraction methods for character recognition. A survey. Pattern Recognition 29, 4 (1995), 641-662.
- [13] Dargenton, P. Contribution à la segmentation et à la reconnaissance de l'écriture manuscrite. PhD thesis, INSA Lyon, 1994.
- [14] NVIDIA. : CUDA C Best Practice Guide (version 4.0), Santa Clara, California, USA (2011)
- [15] NVIDIA CUDA C Programming Guide, [http ://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html](http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html)

- [16] Khronos, OPENCL : The open Standard for parallel programming of heterogenous systems, <http://www.khronos.org>
- [17] Parallel programming and computing platform, CUDA, NVIDIA
- [18] P. Xiu, H.S. Baird, *Whole-Book Recognition*, Pattern Analysis and Machine Intelligence, IEEE Trans. vol. 34 , no. 12, 2012.
- [19] P. Stathis, E. Kavallieratou, N.Papamarkos, "*An evaluation technique for binarization algorithms*", Journal of Universal Computer Science, vol. 14, no. 18,3011-3030, 2008.
- [20] A. Rosenfeld, and A. C. Kak, *Digital Picture Processing*, 2nd ed. New York : Academic, 1982.
- [21] E. Kavallieratou, "*A binarization algorithm specialized on document images and photos*", 8th Int. Conf. on document Analysis and Recognition, pp. 463-467, 2005.
- [22] C. A. Glasbey, "*An Analysis of histogram-based thresholding algorithm, Graph*", Models Image Processing 55, 532-537, 1993.
- [23] N. Otsu, "*A threshold selection method from gray-level histograms*", IEEE Transactions on Systems, Man and Cybernetics, 9(1) : 62-66, 1979.
- [24] F. Martín, "*Analysis tools for gray level histograms*", Proc. Of SPPRA, pp. 11-16, 2003.
- [25] R. Duda, and P. Hart, "*Pattern Classification and Scene Analysis*", Wiley, New York, 1986.
- [26] S. Ghosh, S.K. Dubey, "*Comparative Analysis of K-Means and Fuzzy C-Means Algorithms*", (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 4, no. 4, 2013.
- [27] S. P. LLOYD, "*Least square quantization in PCM*", IEEE Transactions on Information Theory vol.28, no.2, 129-137, 1982.
- [28] V. Pritesh , O. Bhavesh , "A Survey on K-mean Clustering and Particle Swarm Optimization", International Journal of Science and Modern Engineering (IJISME), 24-26, 2013.
- [29] M. Valizadeh, E. Kabir, "*An adaptive water flow model for binarization of degraded document images*", International Journal on Document Analysis and Recognition (IJ-DAR) June 2013, vol. 16, no. 2, pp 165-176, 2013.
- [30] A.N Lai and G. Lee, "Binarization by Local K-means Clustering for Korean Text Extraction", IEEE International Symposium on Signal Processing and Information Technology, pp. 117-122, Dec. 2008.
- [31] K. Kita and T. Wakahara, "Binarization of Color Characters in Scene Images using K-means Clustering and Support Vector Machines", in ICPR, 2010, pp. 3183-3186.
- [32] W. Niblack, *An Introduction to Digital Image Processing*, Standberg Publishing Company, 1985.
- [33] J. Sauvola, M. Pietikainen, "*Adaptive document image binarization*", Pattern Recognition, vol. 33, 225-236, 2000.
- [34] C. Wolf, J.-M. Jolion, "*Extraction and recognition of artificial text in multimedia documents*", Pattern Analysis and Applications, vol. 6, 309-326, 2004.

- [35] M. P. Greenleaf, J. Shen, D. Yoon, "*Local-Global Image Binarization reconstructing the cellular structure of polymer foam materials*", Computer-Aided Design and Applications, 10(6), pp 919-928, 2013.
- [36] E. Gabarra, A. Tabbone, "*Combining global and local threshold to binarize document of images*", Pattern Recognition and Image Analysis, vol : 3523, 173-186, 2005.
- [37] V. Sokratis, E. Kavallieratou, R. Paredes, K. Sotiropoulos, "*A hybrid binarization technique for document images*", Learning Structure and schemas from documents studies in computational intelligence. Vol. 375, 165-179, 2011.
- [38] K. Singh, D. Malik, N. Sharma, "*Evolving limitations in Kmeans algorithm in data mining and their removal*", IJCEM International Journal of Computational Engineering and Management, vol. 12,105-109, 2011.
- [39] N.A.M. Isa, S.A. Salamah, "*Adaptive fuzzy moving K-means clustering algorithm for image segmentation*", Consumer Electronics, IEEE Transactions on, vol. 55, no. 4, 2009.
- [40] S. Cha, "*Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions*", International Journal of Mathematical Models and Methods in Applied Sciences, 1(4),300-307, 2007.
- [41] M. B. Al-Daoud, N. B. Venkateswarlu and S. A. Roberts, "*New methods for the initialization of clusters*", Pattern Recognition Lett., vol. 17, no. 15, 451-455, 1996.
- [42] J. M. Pena, J. A. Lazano, P. and Larranaga, "*An empirical comparison of four initialization methods for the K-means algorithm*", Pattern Recognition Lett., 20, 1027-1040, 1999.
- [43] R. Smith, "*An Overview of the Tesseract OCR Engine*", Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR), IEEE Computer Society, 629-633 , 2007.
- [44] T. Lelore, F. Bouchara, "*Super-resolved binarization of text based on the FAIR algorithm*", Proceedings of international conference on Document Analysis and Recognition, 839-843, 2011.
- [45] J. Fabrizio, B. Marcotegui, M. Cord, : "*Text segmentation in natural scenes using toggle-mapping*". Image Processing (ICIP), 16th IEEE International Conference, 2349-2352, 2009.
- [46] J.R. Prasad, Template matching algorithm for Gujrati Character Recognition, IEEE 2nd International conference on Emerging Trends in Engineering and Technology, ICE-TET (2009)
- [47] Z.You-qing, A Recognition Method of Car License Plate Characters Based on Template Matching Using Modified Hausdorff Distance. IEEE International Conference on Computer, Mechatronics, control and electronic Engineering (CMDE) (2010)
- [48] F.Aghdasi, Automatic Licence Plate recognition system, IEEE AFRICON (2004)
- [49] J.Jiao, A configurable method for multi-style license plate recognition. Pattern Recognitionn, Vol : 42, No : 3, pages 358-369 (2009)
- [50] M.M Farhad, An efficient Optical Character Recognition Algorithm using Artificial Neural Network by Curvative Properties.IEEE International conference on informatics, electronics é vision (2014)

- [51] S.K.Thilagavathy, Rrecognition Of Distorted Character Using Edge Detection Algorithm, International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 4, June 2013
- [52] M.soua, R.Kachouri, A.Akil. A new hybrid binarization method based on Kmeans, ISSCP 2014.
- [53] S.Kopf, T.Haenselmann, W.Effelberg. Robust Character Recognition in Low-Resolution Images and videos.
- [54] A. Antanacopoulos, C. Clausner, C. Papadopoulos, and S.Plutschacher. ICDAR2013 Competition on Historical Newspaper Layout Analysis., International Conference on Document Analysis and Recognition, 2013
- [55] S.Sharstry, Gumasheela, T.DUTT, D.S.Vinay. "i" A novel algorithm for optical character recognition (OCR), 2013.
- [56] R. Smith, "*Hybrid Page Layout Analysis via Tab-Stop Detection*", ICDAR '09 Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, Pages 241-245, 2009
- [57] K.Lengwehasarit, "*Probabilistic partial-distance fast matching algorithms for motion estimation* ", pp 139 - 152. Circuits and Systems for Video Technology, IEEE Transactions on (Volume :11 , Issue : 2), 2002.
- [58] M.Soua, R.Kachouri, M.Akil. GPU parallel implementation of the new hybrid binarization based on Kmeans method (HBK) Journal of Real-Time Image Processing, Springer, 2014
- [59] Q.Akram, S.Hussain, Z.Habibt, Font-Size Independent OCR for Noori Nastaleeq.
- [60] Z. Abdelwahab and R. Ingold, Optical Font Recognition Using Typographical Features,IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp.877-882, 1998
- [61] [http ://www.sitepoint.com/the-old-style-type-face/](http://www.sitepoint.com/the-old-style-type-face/)
- [62] Souza, M. ; Naoi, S. ; Suen, C.Y. A.Automatic filter selection using image quality assessment, 2003
- [63] R. M. Haralick, L.G. Shapiro, Survey : image segmentation techniques, Comput. Vision, Graphics Image Process., Vol. 29, pp. 100-132, 1985
- [64]
- [65] N. Otsu, "*A threshold selection method from gray-level histograms*", IEEE Transactions on Systems, Man and Cybernetics, 9(1) : 62-66, 1979.
- [66] S. P. LLOYD, "*Least square quantization in PCM*", IEEE Transactions on Information Theory vol.28, no.2, 129-137, 1982.
- [67] W. Niblack, *An Introduction to Digital Image Processing*, Standberg Publishing Company, 1985.
- [68] J. Sauvola, M. Pietikainen, "*Adaptive document image binarization*", Pattern Recognition, vol. 33, 225-236, 2000.
- [69] G. Lazzara, T. Géraud, "*Efficient Multiscale Sauvolas Binarization*", International Journal on document analysis and recognition, 2013.

- [70] Soua, M., Kachouri, R., Akil, M. : A new hybrid binarization method based on Kmeans. In : IEEE International Symposium on Communications, Control, and Signal Processing, pp. 118-123 (2014)
- [71] C. P. Sumathi and G. Gayathri Devi, Automatic Text Extraction From Complex Colored Images Using Gamma Correction Method, Journal of Computer, Volume 10, Issue 4 Science, Pages 705-715, 2014
- [72] Pizer, S. M., E.P. Ambum and J.D. Austin, Adaptive histogram equalization and its variation, Computer Vision, Graphics, and Image Processing Vol. 39, No.3, (1987), 355-368.
- [73] Asadi Amiri, S., Hassanpour ,H. and Pouyan, A.,Texture Based Image Enhancement Using Gamma Correction, Middle-East Journal of Scientific Research , Vol. 6, (2010), 569-574.
- [74] Gilboa, G., Sochen, N. and Zeevi, Y. Y., Image enhancement and denoising by complex diffusion processes, IEEE Transactions on Pattern Analysis and Machine Intelligence , Vol. 26, No. 8, (2004), 1020-1036.
- [75] Guillon, S., Baylou, P., Najim, M. and Keskes, N., Adaptive nonlinear filters for 2D and 3D image enhancement, Signal Processing, Vol. 67, No. 3 (1998), 237-254.
- [76] Fries, R. and Modestino, J., Image enhancement by stochastic homomorphic filtering, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 27, No. 6, (1979), 625-637.
- [77] Farid, H., "Blind inverse gamma correction", IEEE Transactions on Image Processing, Vol. 10, (2001),pp. 1428-1433.
- [78] C.p Sumathi. A survey on various approaches of text extraction in images. International Journal of computer sciences ingeneering Vol3. No 4 2012
- [79] K.Kaur, N.Kanwal, J.S.Bhullar. A Technique for Enhancement of Gray Image using Local Gamma Correction. International Journal of Computer Applications. Vol 105-No.5,2014
- [80] P. Clark and M. Mirmhedi. Rectifying perspective views of text in 3d scenes using vanishing points. Pattern Recognition, 36 :2573-2686, 2003.
- [81] Karatzas, D. and F.Shafait, S.Uchida, M.Iwamura. ICDAR 2013 Robust Reading Competition. Document Analysis and Recognition (ICDAR), pp 1484-1493, 2013
- [82] Trier, O.D. et Jain, A.K. 1995. Goal-directed evaluation of thresholding methods, IEEE Trans. Pattern Anal. Mach. Intell. 17 (12) 1191-1201
- [83] A novel algorithm for optical character recognition (OCR), 2013.
- [84] Soua, M., Kachouri, R., Akil, M. : A new hibrid binarization method based on Kmeans. In : International Symposium on Communications, Control, and Signal Processing (2014)
- [85] Xiu, P., Baird, H.S. : Whole-Book Recognition. In : Pattern Analysis and Machine Intelligence, IEEE Trans. vol. 34, no. 12, pp. 2467-2480 (2012)
- [86] Kae, A., Huang, G.B., Doersch, C., and Learned-Miller, E.G. : Improving state-of-the-art OCR through high-precision document-specific modeling. In : CVPR, pp. 1935-1942 (2010)

- [87] Collins-Thompson, K., Schweizer, C., Dumais, S. : Improved String Matching Under Noisy Channel Conditions. In : Proceedings of the tenth international conference on Information and knowledge management, pp. 357-364 (2011)
- [88] Eikvil, L. : OCR-Optical Character Recognition (1993)
- [89] Gaceb, D., Eglin, V., Lebourgeois, F. : A new mixed binarization method used in real time application of automatic business document and postal mail sorting. In : Int. Arab J. Inf. Technol. 10(2), 179-188 (2013)
- [90] Ashari, E., Hornsey, R. : FPGA Implementation of real-time Adaptive Image Thresholding. In : Proc. Photonic Applications in Astronomy, Biomedicine, Imaging, Materials Processing, and Education, (2004)
- [91] Fong, W. : Document Imaging : A step toward a paperless office,
<http://web.simmons.edu/~chen/nit/NIT%2792/133-fon.htm>
- [92] Kumar, D. and al. : MAPS : Midline analysis and propagation of segmentation. In : Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing. Article No. 15 (2012)
- [93] Niblack, W. : An Introduction to Digital Image Processing. In : Standberg Publishing Company, (1985)
- [94] J.Sauvola, T,Seppanen, S.Haapakoski, M.Pietikainen, Adaptive Document Binarization, 4th Int. Conf. On Document Analysis and Recognition, Ulm, Germany, pp.147-152 (1997).
- [95] Srinivasan, S. and al. : Performance characterization and acceleration of Optical Character Recognition on handheld. In : Workload Characterization (IISWC), IEEE International Symposium on, pp. 1-10 (2010)
- [96] Li,Y., Zhao, K., Chu, X. and Liu, J. : Speeding up k-Means Algorithm by GPUs. In : Proc. IEEE 10th Int'l Conf. Computer and Information Technology (CIT), pp. 115-122 (2010)
- [97] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J.W., and Skadron, K. : A Performance Study of General-Purpose Applications on Graphics Processors Using CUDA. In : J. Parallel Distributed Computing, vol. 68, no. 10, pp. 1370-1380, Oct 2008
- [98] Fang, W. and al. : Parallel Data Mining on Graphics Processors. In : technical report, Hong Kong Univ. of Science and Technology, (2008)
- [99] Hong-tao, B., Li-li H., Dan-tong O., Zhan-shan L., and He, L. : KMeans on Commodity GPUs with CUDA. In : Proc. WRI World Co
- [100] Sirotkovic, J., Dujmic H., and Papic, V. : K-Means Image Segmentation on Massively Parallel GPU Architecture. In : Proc. 35th Int'l Convention MIPRO, pp. 489-494 (2012)
- [101] Pisharath J., Liu, Y. Liao, W.-k., Choudhary A., Memik G., and Parhi, J. : 'Nu-MineBench 2.0'. In : CUCIS Technical Report Center for Ultra-Scale Computing and Information Security, Northwestern University, (2005)
- [102] LLOYD, S. P. : Least square quantization in PCM. In : IEEE Transactions on Information Theory vol.28, no.2, 129-137 (1982)
- [103] EPITA and Development Laboratory (LRDE)
<http://www.lrde.epita.fr/cgi-bin/twiki/view/Olena/DatasetDBD>

- [104] Lelore, T., Bouchara, F. : Super-resolved binarization of text based on the FAIR algorithm. In : Proceedings of international conference on Document Analysis and Recognition, 13 : 303-314 (2010)
- [105] Fabrizio, J., Marcotegui, B. : Cord, M. : Text segmentation in natural scenes using toggle-mapping. In : Image Processing (ICIP), 16th IEEE International Conference, 2373-2376 (2009)
- [106] Chen, T.Y. and al. On the statistical properties of the F-measure, Quality Software, 2004. QSIC 2004. Proceedings. Fourth International Conference on, pages 146-153, (2004)
- [107] NVIDIA. : CUDA C best practices guide (version 4.0), Santa Clara, California, USA. (2011)
<http://www.khronos.org/opencv/>
- [108] NVidia CUDA C Programming Guide,
<http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html>
- [109] Khronos. OpenCL : The open standard for parallel programming of heterogeneous systems,
<http://www.khronos.org/opencv/>
- [110] Parallel programming and computing platform, CUDA, NVIDIA.
http://www.nvidia.com/object/cuda_hom_new.html.
- [111] Tompson, J. and Schlachter, K. : An introduction to the programming Model. In : Distributed Computing CSCI-GA.2631-001 (2012)
- [112] NVIDIA's Next Generation CUDA TM Compute Architecture : Kepler TM GK110,
<http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf>
- [113] Yi, Y., Lai, C., Petrov, S. : Efficient parallel CKY parsing using GPUs. In : J. Logic Computation, pp 175-185 (2011)
- [114] Khurshid, K. and al. Comparison of Niblack inspired Binarization methods for ancient documents, In proceeding of : 16th Document Recognition and Retrieval Conference, part of the IS and T-SPIE Electronic Imaging Symposium, San Jose, CA, USA, (2009)
- [115] Singh, B.M. and al. : Parallel Implementation of Niblack's Binarization Approach on CUDA. International Journal of Computer Applications 32(2) :22-27, (2011).
- [116] Singh, B.M. and al. : Parallel Implementation of Souvola's Binarization Approach on GPU. In : International Journal of Computer Applications 32(2) :28-33, (2011)
- [117] Wu, R., Zhang, B., and Hsu, M. : Clustering billions of data points using GPUs,. In : Proceeding of the combined workshops on Unconventional high performance computing workshop plus memory access workshop, Ischia, Italy, pp. 1-6, (2009)
- [118] M. Flynn. Some computer organization and their effectiveness. IEEE Transactions on Computers, 21 : 948-960, September 1972
- [119] Zhi Gup and al. A quantitative Analysis of the speed Factors of FPGA over Processors. In. Int'Symp. on Field-Programmable gate Arrays (FPGA), Monterrey, CA, (2004)
- [120] Ben Cope and al. Performance comparison of Graphics Processors to Reconfigurable Logic : A case Study. In. IEEE Transaction on Computers Vol 59, No 4, (2010)

- [121] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998
- [122] Y. Rui, A. C. She, and T. S. Huang, Modified Fourier descriptors for shape representation a practical approach,
- [123] George Teodoro; et al.Coordinating the use of GPU and CPU for improving performance of compute intensive applications, 2009 IEEE International Conference on Cluster Computing and Workshops Page(s) :1 - 10 , (2009)
- [124] Hasan Baig et al.Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on. Performance evaluation of CPU-GPU and CPU-only algorithms for detecting defective tablets through morphological imaging techniques. Page(s) :1 - 6. (2012).
- [125] Abu Asaduzzaman et al. In; SoutheastCon 2015 A time-efficient image processing algorithm for multicore/manycore parallel computing. (2015)Page(s) : 1 - 5
- [126] Aurélien Plyer, Architectures massivement parallèles et vision artificielle bas-niveau, Thèse (2013)
- [127] Mohamed CherietCharacter Recognition Systems : A Guide for Students and Practitioners
- [128] M.soua, R.Kachouri, M.Akil Hybrid binarisation based on Kmeans, ICCSP, 2014
- [129] NVIDIA, "What is GPU Computing," 2012. [Online]. Available : http://www.nvidia.com/object/GPU_Computing.html. [Accessed 18 1 2012].
- [130] Ouajdi Brini, Real-Time CPU-GPU Demodulator for the LTE Physical Layer, Latin American Symposium on Circuits and Systems (LASCAS) 2016
- [131] http://www.gris.informatik.tu-darmstadt.de/cuda-workshop/tutorial/Advanced_CUDA_03.pdf
- [132] R. Smith, "A simple and efficient Skew Detection Algorithm via text Row accumulation", Proc of the third Int. Conf. on document Analysis and Recognition (Vol. 2), IEEE 1995, pp. 1145-1148
- [133] Zhixin Shi A Steerable Directional Local Profile Technique for Extraction of Hand-written Arabic Text Lines, 2009 10th International Conference on Document Analysis and Recognition (2009)
- [134] Zhixin Shi A Steerable Directional Local Profile Technique for Extraction of Hand-written Arabic Text Lines, 2009 10th International Conference on Document Analysis and Recognition (2009)
- [135] R.J. Evey, Use of a computer to design character recognition logic, Proc. Eastern Jt. Comp. Conf., pp. 205-211, 1959.
- [136] M.E. Stevens, Automatic Character Recognition - A State of the Art Report, MES Nat. Bureau of Standards Tech Note, No. 112, 1961.
- [137] R.L. Hoffman and J.W. McCullough, Segmentation methods for recognition of machine-printed characters, IBM Journ. of Res. and Dev., pp. 153-65, March 1971.
- [138] H.S. Baird, S. Kahan and T. Pavlidis, Components of an omnifont page reader, Proc. 8th Int. Conf. on Pattern Recognition, Paris, pp. 344-348, 1986.

- [139] Y. Lu, On the segmentation of touching characters, Int. Conf. on Document Analysis and Recognition, Tsukuba, Japan, p440-443, Oct. 1993.
- [140] M. Cesar and R. Shinghal, Algorithm for segmenting handwritten postal codes, Int. J. Man Mach Stud., vol. 33, no. 1, pp. 63-80, Jul. 1990.
- [141] G. Seni and E. Cohen, External word segmentation of off line handwritten text lines, Pattern Recognition, vol. 27, no. 1, pp.41-52, Jan. 1994.
- [142] R.A. Wilkinson, Comparison of massively parallel segmenters, National Inst. of Standards and Technology Technical Report, Gaithersburg, MD, Sept. 1992.
- [143] C.J.C. Burges, J.I. Be and C.R. Nohl, Recognition of Handwritten Cursive Postal Words using Neural Networks, Proc. USPS 5th Advanced Technology Conference, page A-117, Nov/Dec. 1992.
- [144] K. Fukushima and T. Imagawa, Recognition and segmentation of connected characters with selective attention, Neural Networks, vol. 6, no. 1, pp. 33-41, 1993.
- [145] M. Gilloux, Hidden Markov Models in Handwriting Recognition, Fundamentals in Handwriting Recognition, S. Impedovo (Ed.), NATO ASI Series F : Computer and Systems Sciences, vol. 124, Springer Verlag, 1994.
- [146] A.M. Gillies, Cursive Word Recognition using Hidden Markov Models, Proc. USPS 5th Advanced Technology Conference, Nov/Dec. 1992.
- [147] J. Rocha and T. Pavlidis, New method for word recognition without segmentation, Proc. SPIE Vol. 1906 Character Recognition Technologies, pp. 74-80, 1993.
- [148] H. Ouladj, G. Lorette, E. Petit, J. Lemoine and M. Gaudaire, From Primitives to Letters : A Structural Method to Automatic Cursive Hanwritting Recognition, The 6th Scandinavian Conference on Image Analysis, Finland, page 593, June 1989
- [149] J.C. Simon, Off-line Cursive Word Recognition, Proceedings of the IEEE, page 1150, July 1992
- [150] L.D. Earnest, Machine Recognition of Cursive Writing, C. Cherry editor, Information Processing, Butterworth, London, 1962, pp. 462-466.
- [151] J-V. Moreau, B. Plessis, O. Bourgeois and J-L. Plagnaud, A Postal Check Reading System, Int. Conf. on Document Analysis and Recognition, Saint Malo, France, page 758, Sept. 1991.
- [152] B. Plessis, A. Sicsu. L. Heute, E. Lecolinet, O. Debon and J-V. Moreau, A Multi-Classifer Strategy for the Recognition of Handwritten Cursive Words, Proc. Int. Conf. on Document Analysis and Recognition, Tsukuba City, Japan, pp. 642-645, Oct. 1993.
- [153] Khorshed, M. System and methods for arabic text recognition basedfd on effective arabic text feature extraction. United States patent Appliation 0811911, 2012
- [154] Flynn, M., Some Computer Organizations and Their Effectiveness, IEEE Trans. Comput., Vol. C-21, p. 948, 1972.
- [155] Zhang D., Lu G., A comparative study of curvature scale space and Fourier descriptors for shape-based image retrieval, JVCIR, vol. 14, no 1, 2002, pp. 39-57.
- [156] Khotanzad A., Hong Y., Invariant Image Recognition by Zernike Moments, PAMI, vol. 12, no 5, 1990, pp. 489-497.

- [157] Chong C., Raveendran P., Mukundan R., A comparative analysis of algorithms for fast computation of Zernike moments, PR, vol. 36, no 3, 2003, pp. 731-742.
- [158] Stephane. L. Reconnaissance de caractères industriels par application d'un système de réseaux de neurones à boucle de rétroaction . Thèse 1998
- [159] Canny, J., A Computational Approach To Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8 :679-714 (1986).
- [160] Shijian Lu Scene text extraction based on edges and support vector regression. International Journal on Document Analysis and Recognition (IJDAR) Vol 18 n 12, 2015
- [161] H.M. Suen, J.F. Wang, Text string extraction from images of colour-printed documents, IEEE Proceedings of Vision, Image and Signal Processing, 143 (1996) 210-216.
- [162] Q. Yuan, C.L. Tan, Text Extraction from Gray Scale Document Images Using Edge Information, Washington, Sept. 10-13 (2001) 302-306
- [163] Chitrakala Gopalan , Manjula(2008) ,Text Region Segmentation From Heterogeneous Images, International Journal Of Computer Science And Network Security, Vol.8 No.10, pp.108-113.
- [164] X. Liu And J.Samarabandu(2006), Multiscale Edge-Based Text Extraction From Complex Images, Proc. International Conference Of Multimedia And Expo, pp.1721-1724.
- [165] Isabelle. c, Analyse de textures en traitement d'images, cours, Télécom ParisTech.
- [166] Huei-Yung Lin, Optical character recognition with fast training neural network, 2016 IEEE International Conference on Industrial Technology (ICIT) 2016
- [167] Ali. F. OCR based feature extraction and template matching algorithms for Qatari number plate. 2016 International Conference on Industrial Informatics and Computer Systems (CIICS)
- [168] G. Nguyeb, Extraction de zones d'intérêts dans une image de textures ; Rapport de stage, laboratoire informatique, image et interaction université de la Rochelle 2009
- [169] Journet N., Analyse d'images de documents anciens : une approche texture, thèse de doctorat, université de la Rochelle, 2006
- [170] L. Journaux, et al. classification de textures par descripteurs généralisés de Fourier dans différents contextes de réduction de dimension, GRETSI, Groupe d'études du Traitement du Signal et des Images, 2009
- [171] J.-M. Chassery and A. Montanvert. Geometrical representation of shapes and objects for visual perception. In Geometric Reasoning for Perception and Action, volume 708 of LNCS, pages 163-182. Springer, 1993.
- [172] A. Antonacopoulos and H. Meng, Document Analysis Systems V Volume 2423 of the series Lecture Notes in Computer Science pp 236-244
- [173] R.W. Smith, The Extraction and Recognition of Text from Multimedia Document Images, PhD Thesis, University ofBristol, November 2009.
- [174] Rousseeuw, P. J. "Least Median of Squares Regression". Journal of the American Statistical Association. 79 : 871-880. 1984.

- [175] Nagy, G., Kanai, J., Krishnamoorthy, M., Thomas, M. et Viswanathan, M. (1988). Two complementary techniques for digitized document analysis. In DOCPROCS : Proceedings of the ACM conference on Document processing systems, pages 169-176, New York, NY, USA. ACM.
- [176] O. R. Vincent, A Descriptive Algorithm for Sobel Image Edge Detection, Proceedings of Informing Science and IT Education Conference (InSITE) 2009
- [177] Sachin. G. Text Extraction from Document Images Using Edge Information, Annual IEEE India Conference, pp 1-4 ; 2009
- [178] Xiaoqing Liu. Multis pp 1721-1724 2006 cale Edge-Based Text Extraction from Complex Images, 2006 IEEE International Conference on Multimedia and Expo,
- [179] K. Sobottka. Identification of text on colored book and journal covers, Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on pp 57-62, 1999
- [180] W., Zhang, H., He, X. (2007). "Region-based license plate detection". Journal of Network and Computer Applications.
- [181] P. Fah, Efficient coding of high resolution typographic characters, Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82. pp 440-443 (Volume :7), 1982
- [182] Anuj Sharm,a Online Handwritten Gurmukhi Character Recognition Using Elastic Matching, Image and Signal Processing, 2008. CISP '08. Congress on, pp 391-396 (Volume :2), 2008
- [183] Simon Korman, FasT-Match : Fast Affine Template Matching. Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on pp 2331-2338, 2013
- [184] M. D. Malkauthekar. Analysis of euclidean distance and Manhattan Distance measure in face recognitionn, Computational Intelligence and Information Technology, 2013. CIIT 2013. Third International Conference on, pp 503 - 507, 2013
- [185] Chu-yu Guo, A Japanese OCR post-processing approach based on dictionary matching, 2013 International Conference on Wavelet Analysis and Pattern Recognition. pp 22-26, 2013
- [186] Frédéric GAVA. Approches fonctionnelles de la programmation parallèle et des méta-ordinateurs. Sémantiques, implantations et certification,thèse, université paris 12, 2005.
- [187] André Buchau,, Parallelization of a Fast Multipole Boundary Element Method with Cluster OpenMP, IEEE transactions on Magnetics, Vol. 44, No. 6, 2008
- [188] Welch, Eric, "A Study of the use of SIMD instructions for two image processing algorithms" (2012). Thesis. Rochester Institute of Technology.
- [189] Jingbo Zhou, Qi Guo, H. V. Jagadish, Wenhao Luan, Anthony K. H. Tung, Yueji Yang, Yuxin Zheng, Generic Inverted Index on the GPU, 2016
- [190] Mo Xu, Ningyi Xu, Chunshui Zhao, Feng-Hsiung Hsu, Efficient Weighted Histogramming on GPUs with CUDA, China Microsoft Research Asia, Tech report, 2012
- [191] NVIDIA, Pascal architecture whitepaper, <https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf>

- [192] B. Chapman, Exploring the GPU for enhancing parallelism on color and texture analysis, Parallel computing : from multicores and GPU's to Petascale, 2010
- [193] Asadollah Shahbahrami Parallel implementation of Gray Level Co-occurrence Matrices and Haralick texture features on cell architecture, The Journal of Supercomputing March 2012, Volume 59, Issue 3, pp 1455-1477
- [194] Kyoshin Choo, Understanding and Optimizing GPU Cache Memory Performance for Compute Workloads, 2014 IEEE 13th International Symposium on Parallel and Distributed Computing, pp 189-16, 24-27 June 2014
- [195] Rostom Kachouri, Mahmoud Soua, Mohamed Akil (2016). Unsupervised Image Segmentation Based on Localpixel Clustering and Low-Level Region Merging. 2nd IEEE International Conference on Advanced Technologies for Signal and Image Processing, Monastir (Tunisie), Mars 2016.
- [196] Rostom Kachouri, Christian Armas, Mohamed Akil. Gamma correction acceleration for real-time text extraction from complex colored images. Image Processing (ICIP) 2015.
- [197] David J. Lilja (ed.), Architectural Alternatives for Exploiting Parallelism, IEEE Computer Society Press, Los Alamitos, CA, ISBN 0-8186-2642-9, 1992.