الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

présenté par

BOUAZZA Bilal

&

MELBOUS Youcef

pour l'obtention du diplôme Master en Électronique option Systèmes de Vision Robotique

Thème

Reconnaissance de chiffres imprimés/manuscrits en temps réel via WebCam. Application à la reconnaissance de numéro de mobile

Proposé par : Mr .Abderrahmane Namane

Année Universitaire 2011-2012

Remerciements

Listes des acronymes et abréviations

DAO : Dessin Assisté par Ordinateur

RVB : Rouge, Vert, Bleu

CIE : Commission Internationale de l'Eclairage

CMJ : Cyan, Magenta, Jaune

dpi : dots per inch

ppp : points par pouce

DF : les descripteurs de Fourier

DFT : transformée de Fourier discrète

Open : CV Open Computer Vision

RAD : Développement rapide d'applications

GUI : Interface Utilisateur Graphique

IPP : Integrated Performance Primitives

k-ppv : k-plus proche voisine

Bibliographie

[1] Khouloud Meskaldji, Extraction et traitement de l'information : Un prototype d'un Système de recherche d'images couleurs par le contenu magistère, Université Mentouri de Constantine, 2009.

[2] Hadjila Feth Allah et Bouabdallah Réda, Reconnaissance des visages en utilisant les réseaux de neurones. Mémoire d'ingénieur. Université de Tlemcen. 2003.

[3] DOUMANDJI Samah, BOULFANI Yasmine, « Implémentation sur DSP TMS320C5000 de filtres optimaux appliqués aux images et introduction de réseaux neuronaux », Projet de fin d'études, Electronique, ENP, Alger, 2004.

[4] V.Wu and R. Manmatha, «Document image clean-up and binarization», Proceedings of IS&T/SPIE Symposium on Electronic Imaging, 3305:263–273, 1998, (Cite pages 16 ET 23).

[5] J. Bernsen, «Dynamic thresholding of grey-level images», In Proc. Eighth Int 'l Conf. on Pattern Recognition, pages 1251–1255, 1986. (Cité pages 17 et 23).

[6] G. Rajput, M. Mali, « Marathi Handwritten Numeral Recognition using Fourier Descriptors and Normalized Chain Code», These, Department of Computer Science, Gulbarga University, Gulbarga 585106 Karnataka, Solapur University, Solapur Maharastra, India.

[7] R. Horaud, O. Monga. Vision par ordinateur – outils fondamentaux. Editions Hermès, 1993.

[8] H.D. Cheng, Yen-Hung Chen. « Fuzzy partition of two -dimensional histogram and its application to thresholding ». Pattern Recognition, vol.32, pp.825-843, 1999.

[9] H.D. Cheng, C. H. Chen, H.H. Chiu and Huijuan Xu. « Fuzzy Homogeneity Approach to Multilevel Thresholding ». IEEE Transactions on Image Processing, vol. 7, n. 7, July 1998.

[10] Sue Wu, Adnan Amin. « Automatic thresholding of grey-level using multi-stage approach », 7th International Conference on Document Analysis and Recognition, vol.1, August 2003.

[11] N.Otsu. « A threshold selection method from grey-level histograms ». IEEE Trans. Syst. Man. Cybern., vol.SMC-8, 1978.

[12] J.N. Kapur, P.K. Sahoo, A.K.C. Wong. « A New method for gray-level picture threshold using the entropy of the histogram ».Graphical Models and Image processing, 29, 1985.

[13] T.Pun. « A New method for gray-level picture threshold using the entropy of the histogram ». Signal processing, vol.2, n°3, 1980.

[14] H.D. Cheng, Jim-Rong Chen and Jiguang Li. « Threshold selection based on fuzzy c-partition entropy approach ». Pattern Recognition, vol. 31, No 7, pp. 857-870, 1998.

[15] Carlos A.B. Mello and Rafael D. Lins. « Image segmentation of historical documents ». http://www.upe.poli.br/dsc/recpad/site_hist/visual2000.pdf.

[16] http://fr.wikipedia.org/wiki/M%C3%A9thode_d%27Otsu.

[17] A. Chottera and M. Shridhar, "Feature extraction of manufactured parts in the presence of spurious surface reflections, «Canadian Electron. J., vol. 7, no. 4, pp. 29-33, 1982.

[18] M. Sonka, V. Hlavac, R. Boyle. Image Processing, Analysis and Machine Vision. PWS Publishing, seconde edition, 1999.

[19] Freeman, H. (1961). On the encoding of arbitrary geometric configurations. IEEE Trans on Electr . Comput., 10 :260–268.

[20] Freeman, H. (1977). Computer processing of line drawing images. ACM Computer Survey, 6:57–97.

[21] Robert Laganière. OpenCV 2 Computer Vision Application Programming Cookbook. First published: May 2011.

ملخص:

التعرف التلقائي على الأرقام هي تكنولوجيا تستخدم في وقتنا الحالي في عدة مجالات، هنالك عدة طرق تستعمل للتعرف على الأرقام. والهدف من هذا المشروع هو تقديم وشرح خوارزمية التعرف التلقائي على الأرقام المكتوبة بخط اليد في الوقت الحقيقي عن طرق كاميرا الفيديو الرقمية، وتقديم نتائج مستخلصة هذه الخوارزمية.

كلمات المفاتيح: الأرقام مكتوبة بخط اليد، التعرف التلقائي على الأرقام مكتوب بخط اليد.

Résumé :

La reconnaissance automatique des chiffres est une technologie qui utilise à l'heure actuelle dans plusieurs domaines, il existe plusieurs méthodes utilisées pour identifier les numéros. le but de ce projet est d'introduire et explique l'algorithme de reconnaissance automatique des chiffres manuscrites en temps réel via webcam, et présenter les résultats issus de cet algorithme.

Mots clés : Chiffres manuscrites, reconnaissance automatique des chiffres manuscrits.

Abstract :

Automatic recognition of numbers is a technology that uses the current time in a Sev areas, exist Sev method used to identify the numbers. The aim of this project is to introduce and explain the algorithm for automatic recognition of handwritten digits in real time via webcam, and present the results of this algorithm.

Keywords: Handwritten numerals**,** Automatic recognition of Handwritten numerals**.**

SOMMAIRE

Liste des figures

Liste des tableaux

Introduction générale

La reconnaissance de l'écriture manuscrite remonte à plus d'une trentaine d'années. Une grande classe d'application est aujourd'hui à l'étude : les applications bancaires ou postales.

Ces applications montrent clairement les spécificités du domaine de la reconnaissance de l'écriture manuscrite par rapport à celui de la reconnaissance optique des caractères (OCR : Optical Character Recognition) qui concerne les caractères imprimés ou dactylographiés. Il est nécessaire de distinguer également la reconnaissance en ligne (on-line) de l'écriture manuscrite, qui relève plutôt de l'interfaçage entre l'homme et l'ordinateur (un stylo spécial est connecté à la machine et ne fonctionne que sur une tablette sensible), de la reconnaissance hors ligne (off-line) ou l'entrée est une image numérique de chiffre. Seule la reconnaissance hors ligne sera considérée dans ce travail.

Dans se travail, nous présentons une méthode de reconnaissance automatique des chiffres manuscrits. Cette méthode consiste à l'application de suivi de contour pour représenter les frontières de chiffres. Ensuite, on fait une description de ces frontières par l'application de descripteur de Fourier avec codage de freeman. Un vecteur de forme calculé sera utilisé par le module de classification qui consiste à la dernière étape de reconnaissance des chiffres imprimés et manuscrits.

L'objet de ce mémoire est la reconnaissance des chiffres imprimés et manuscrits. Il est organisé en trois chapitres. Le premier chapitre présente une généralité sur le traitement d'image.

Le deuxième chapitre présente les étapes de la chaine de reconnaissance des chiffres imprimés et manuscrits.

Nous y détaillons, notamment, les formules mathématique et les algorithmes qui sont utilisés. Le troisième chapitre sera consacré à la partie application et résultats.

CHAPITRE 1 Généralités sur le Traitement d'Image

1.1 Introduction

Le traitement d'images désigne une discipline des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information.

Il s'agit donc d'un sous-ensemble du traitement du signal dédié aux images et aux données dérivées comme la vidéo (par opposition aux parties du traitement du signal consacrées à d'autres types de données : son et autres signaux monodimensionnels notamment), tout en opérant dans le domaine numérique (par opposition aux techniques analogiques de traitement du signal, comme la photographie ou la télévision traditionnelles).

Dans ce chapitre nous présentons quelques notions de base du domaine de traitement d'image numérique tels que : la définition d'mage, image numérique, les types d'image, formats d'images, caractéristiques d'image, système de traitement d'image, filtrage, le seuillage, binarisation par seuillage, segmentation et en fin quelques exemples concrets de traitement d'images.

Quelques exemples types d'informations qu'il est possible d'obtenir d'une image numérique:

- La luminance moyenne
- Le contraste moyen
- La couleur prédominante
- Le taux d'acuité moyen (précis ou flou)
- Le taux d'uniformité des couleurs
- La présence ou l'absence de certains objets

1.2 Définition de l'image

L'image est une représentation d'une cène ou d'un objet par la peinture, la sculpture, le dessin, la photographie, le film, etc.

C'est aussi un ensemble structuré d'informations qui, après affichage sur l'écran, ont une signification pour l'œil humain.

Elle peut être décrite sous la forme d'une fonction $\mathbf{I}(x,y)$ de brillance analogique continue, définie dans un domaine borné, Les x et y sont les coordonnées spatiales d'un point de l'image et $\mathbf{I}$ est une fonction d'intensité lumineuse et de couleur. Sous cet aspect, l'image est inexploitable par la machine, ce qui nécessite sa numérisation

1.3 Image numérique

Contrairement aux images obtenues à l'aide d'un appareil photo, ou dessinées sur du papier, les images manipulées par un ordinateur sont numériques (représentées par une série de bits).

L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs prélevé à l'emplacement correspondant dans l'image réelle, ou calculé à partir d'une description interne de la scène à représenter

La numérisation d'une image est la conversion de celle-ci de son état analogique en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f(x,y) où :

X, y : coordonnées cartésiennes d'un point de l'image.

f(x, y) : niveau de gris en ce point

Pour des raisons de commodité de représentation pour l'affichage et l'adressage, les données images sont généralement rangées sous formes de tableau $\mathbf{I}$ de n lignes et p colonnes. Chaque élément $\mathbf{I}(x, y)$ représente un pixel de l'image et à sa valeur est

associé un niveau de gris codé sur m bits (2^m niveaux de gris ; 0 = noir ; 2^m-1 = blanc). La valeur en chaque point exprime la mesure d'intensité lumineuse perçue par le capteur.

1.3.1 Les classer des images numériques

On peut classifier les images numériques selon deux critères:

1.3.1.1 Selon la représentation numérique

Ils existent deux types:

a Images matricielles (ou images bitmap)

Elle est composée comme son nom l'indique d'une matrice (tableau) de points à plusieurs dimensions, chaque dimension représentant une dimension spatiale (hauteur, largeur, profondeur), temporelle (durée) ou autre (par exemple, un niveau de résolution).

b Images vectorielles

Le principe est de représenter les données de l'image par des formules géométriques qui vont pouvoir être décrites d'un point de vue mathématique. Cela signifie qu'au lieu de mémoriser une mosaïque de points élémentaires, on stocke la succession d'opérations conduisant au tracé. Par exemple, un dessin peut être mémorisé par l'ordinateur comme « une droite tracée entre les points (x1, y1) et (x2, y2) », puis « un cercle tracé de centre (x3, y3) et de rayon 30 de couleur rouge ».

L'avantage de ce type d'image est la possibilité de l'agrandir indéfiniment sans perdre la qualité initiale, ainsi qu'un faible encombrement. L'usage de prédilection de ce type d'images concerne les schémas qu'il est possible de générer avec certains logiciels de DAO (Dessin Assisté par Ordinateur). Ce type d'images est aussi utilisé pour les animations Flash, utilisées sur Internet pour la création de bannières publicitaires, l'introduction de sites web, voire des sites web complet.

1.3.1.2 Selon la représentation des couleurs

Il existe plusieurs modes de codage informatique des couleurs, le plus utilisé pour le maniement des images est l'espace colorimétrique Rouge, Vert, Bleu (RVB ou RGB – Red green Blue).

a Les modèles de couleur

La couleur est l'une des composantes principales pour la description et l'analyse des images couleurs dans le domaine de la recherche d'images par le contenu. Comme nous avons, les couleurs perceptibles par l'œil humain représente une petite gamme de l'ensemble du spectre électromagnétique qui représente tous les rayons, depuis les rayons cosmiques et les rayons X jusqu'aux ondes électriques (Figure 1.1).

Comme la démonstration de la (Figure1.1) du spectre électromagnétique, les couleurs visibles par l'œil humain s'étendent entre les longueurs d'ondes allant de 4000 à 7000 angströms, représentant respectivement les couleurs : violet et rouge et toutes les couleurs entre les deux.

Toutes les autres ondes s'étendant des rayons cosmiques des étoiles jusqu'aux ondes FM de nos radios ne peuvent pas être perçues par l'œil humain. Cette petite gamme du spectre est l'espace de couleur perceptible par l'œil humain [1].

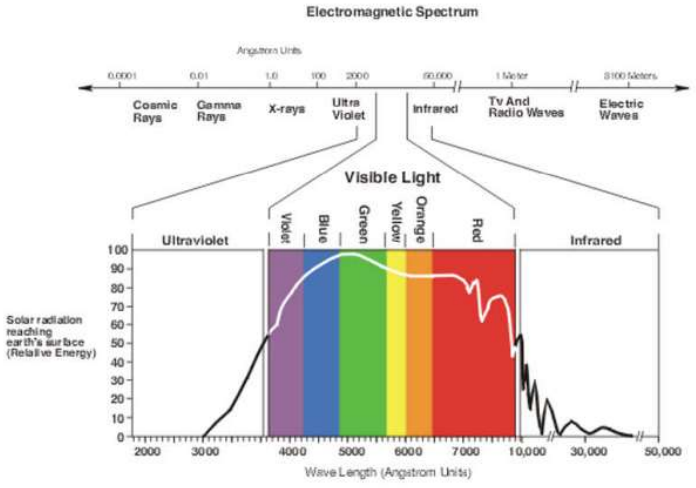


Figure 1.1 : Le spectre électromagnétique.

b Définition du modèle de couleur

Comme toutes les représentations mathématiques des phénomènes physiques, les couleurs peuvent être exprimées selon différentes manières, chacune ayant ses avantages et ses inconvénients.

Un modèle de couleur (modèle chromatique) est un modèle mathématique abstrait décrivant la façon dont les couleurs peuvent être représentées en tant que tuples de nombres, généralement de trois ou quatre valeur ou composantes de couleur. Le but d'un modèle de couleur est de faciliter la spécification des couleurs d'une manière [1].

Les modèles sont des abstractions qui ne peuvent pas décrire une couleur spécifique sans que la référence ne soit définie au préalable.

La première distinction majeure entre les modèles de couleur est la dépendance du diapositif utilisé. Les coordonnées d'une couleur dans un modèle indépendant sont toujours les mêmes sur tous les dispositifs d'affichage. Un exemple de cette catégorie est le modèle XYZ de la CIE (Commission Internationale de l'Eclairage). D'un autre coté, un modèle qui dépend du dispositif de sortie aura différents coordonnées pour la même couleur pour des dispositifs différents. RVB (Rouge, Vert, Bleu) et CMJ (Cyan, Magenta, Jaune) sont des exemples de cette catégorie.

La deuxième distinction est l'uniformité perceptuelle des couleurs. Ceci veut dire qu'une même variation de la valeur des composantes est toujours perçue comme la même variation de couleur. En d'autres termes, la mesure de la variation perçue par un humain est égale à la mesure de la distance mathématique.

Dans ce qui suit nous allons présenter un modèle de couleur de puis plusieurs modèles de couleur :

- **Le modèle RVB(en anglais RGB)**

Le modèle RVB est le modèle de couleur le plus utilisé pour la représentation de la couleur. Il est composé des trois couleurs primaires : rouge, vert et bleu. Ce modèle est aussi le plus utilisé pour reproduction de la couleur sur les dispositifs d'affichage tels

que la télévision et les écrans des ordinateurs. Ces trois couleurs sont appelées les couleurs primaires additives (Figure 1.2).

Par la variation de leur combinaison, d'autres couleurs peuvent être obtenues. Ce modèle peut être représenté sous forme d'un cube unitaire (Figure 1.3) avec : noir (0,0,0), blanc(1,1,1), rouge (1,0,0), vert (0,1,0), et le bleu (0,0,1). Et les couleurs secondaires (Figure 1.4): cyan (0, 1,1), magenta (1, 0,1) et le jaune (1, 1,0).

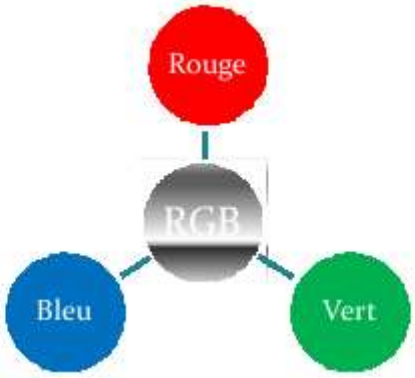


Figure 1.2 : Les trois couleurs primaires additives de lumière : Rouge, Vert, et Bleu

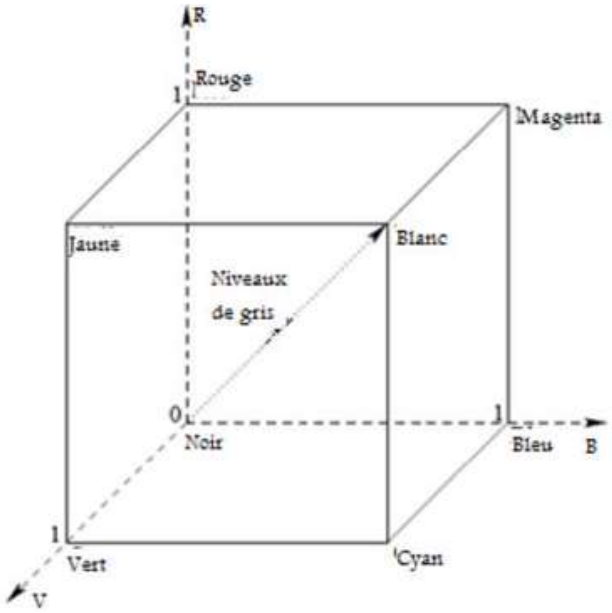


Figure 1.3 : Le modèle de couleur RVB

1.4 Les type d'images

On distingue généralement les différents types d'images suivants :

1.4.1 Images en teintes (ou niveaux de gris)

Pour les images en niveaux de gris, le travail se fait avec une palette de 256 couleurs. Il s'agit cependant ici d'une palette bien déterminée. Le rouge, le vert et le bleu ont des parts égales et cela pour les 256 valeurs définies, ce qui donne une tonalité entre le noir et le blanc. Quand R, V et B ont pour valeur 0, la valeur ainsi définie est un noir profond. Quand les trois parts de couleurs ont pour valeur 255, c'est un blanc pur qui est défini. Une valeur intermédiaire, par exemple 192,192,192, définit une couleur grise d'une certaine luminosité. La palette d'une image en niveaux de gris est constituée par les 256 couleurs comprises entre 0, 0,0 et 255,255,255 qui y sont sauvegardées dans 256 nuances de gris.

Les illustrations suivantes montrent à gauche le tableau typique des nuances de gris, constitué par 256 nuances de gris du noir au blanc, et à droite l'application d'un tel tableau de nuances de gris sur une photo.

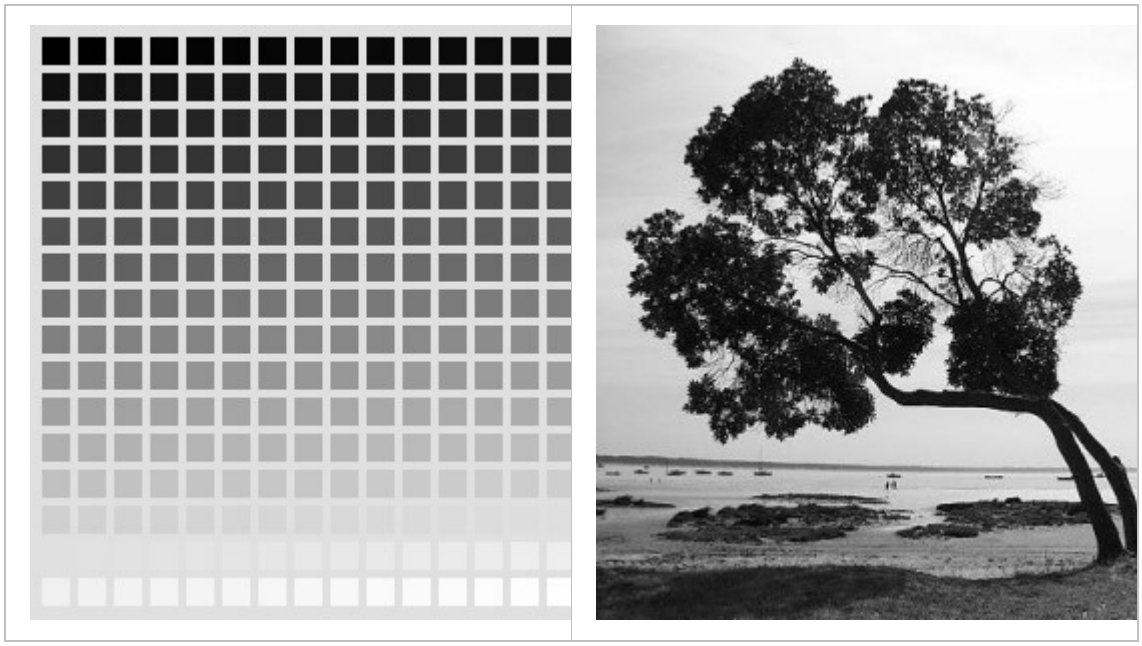
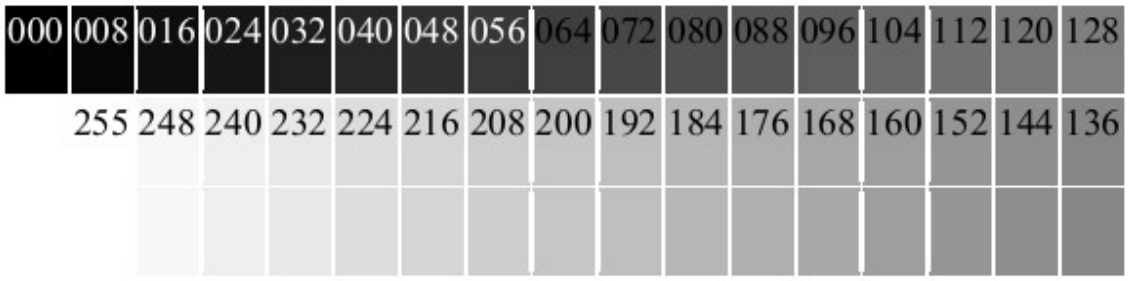


Figure 1.4 : Le modèle à niveaux de gris

Dans l'illustration de l'exemple, il est vrai, la photo après avoir été ramenée aux nuances de gris, a été fixée à nouveau sur une pleine intensité des couleurs et a été en suite sauvegardée dans le format JPEG. Le graphique devient ainsi beaucoup plus petit. Car même en ramenant une photo avec pleine intensité des couleurs à une photo en nuances de gris, il est travaillé énormément avec le tramage à diffusion d'erreur Dithering (Errer-Diffusion), ce qui rend vite les tailles de fichiers de photo excessives.



Tableaux 1.1 *:* Valeurs des niveaux de gris et teintes correspondantes.

1.4.2 Images à palettes

- **images en 256 couleurs (8 bits)**

Pour réduire la place occupée par l'information de couleur, on utilise une palette de couleurs « attachée » à l'image. On parle alors de couleurs indexées : la valeur associée à un pixel ne véhicule plus la couleur effective du pixel, mais renvoie à l'entrée correspondant à cette valeur dans une table (ou palette) de couleurs appelée look-up table ou LUT en anglais, dans la quelle on dispose de la représentation complète de la couleur considérée.

Selon le nombre de couleurs présentes dans l'image, on peut ainsi gagner une place non négligeable : on considère en pratique que 256 couleurs parmi les 16 millions de couleurs 24 bits sont suffisantes. Pour les coder, on aura donc une palette occupant 24 bits x 256 entrées, soit 3 x 256 octets, et les pixels de l'image seront associés à des index codés sur un octet.

L'occupation d'une telle image est donc de 1 octet par pixel plus la LUT, ce qui représente un peu plus du tiers de la place occupée par une image en couleurs 24 bits

(plus l'image contient de pixels, plus le gain de place est important, la limite étant le tiers de la place occupée par l'image en couleurs vraies).

- **Images 24 bits (ou « couleurs vraies »)**

Il s'agit d'une appellation trompeuse car le monde numérique (fini, limité) ne peut pas rendre compte intégralement de la réalité (infinie). Le codage de la couleur est réalisé sur trois octets, chaque octet représentant la valeur d'une composante couleur par un entier de 0 à 255.

Ces trois valeurs codent généralement la couleur dans l'espace RVB. Le nombre de couleurs différentes pouvant être ainsi représenté est de 256 x 256 x 256 possibilités, soit près de 16 millions de couleurs. On considère commodément que ce système permet une restitution exacte des couleurs, c'est pourquoi on parle de « couleurs vraies ».

R	V	B	Couleur
0	0	0	noir
0	0	1	nuance de noir
255	0	0	rouge
0	255	0	vert
0	0	255	bleu
128	128	128	gris
255	255	255	blanc

Tableaux 1.2 : Représentation en variant des couleurs (24 bits).

Les images bitmap basées sur cette représentation peuvent rapidement occuper un espace de stockage considérable, chaque pixel nécessitant trois octets pour coder sa couleur.

1.5 Format d'image

Un format d'image est une représentation informatique de l'image, associée à des informations sur la façon dont l'image est codée et fournissant éventuellement des indications sur la manière de la décoder et de la manipuler.

La plupart des formats sont composés d'un en-tête contenant des attributs (dimensions de l'image, type de codage, LUT, etc.), suivi des données (l'image proprement dite). La structuration des attributs et des données diffère pour chaque format d'image.

- la date, l'heure et le lieu de la prise de vue.
- les caractéristiques physiques de la photographie (sensibilité ISO, vitesse d'obturation, usage du flash…)

a Tableau comparatif de différents formats d'images

	Type (matriciel/ vectoriel)	Compression des données	Nombre de couleurs supportées	Affichage progressif	Animation	Transparence
JPEG	matriciel	Oui, réglable (avec perte)	16 millions	Oui	Non	Non
JPEG2000	matriciel	Oui, avec ou sans perte	32 millions	Oui	Oui	Oui
GIF	matriciel	Oui, Sans perte	256 maxi (palette)	Oui	Oui	Oui
PNG	matriciel	Oui, sans perte	Palettisé (256 couleurs ou moins) ou 16 millions	Oui	Non	Oui (couche Alpha)
TIFF	matriciel	Compression ou pas avec ou sans pertes	de monochrome à 16 millions	Non	Non	Oui (couche Alpha)
SVG	vectoriel	compression possible	16 millions	* ne s'applique pas *	Oui	Oui (par nature)

Tableaux 1.3 : Tableau comparatif de différents formats d'images

1.6 Caractéristiques d'une image numérique

L'image est un ensemble structuré d'informations caractérisé par les paramètres suivants:

1.6.1 Pixel

Contraction de l'expression anglaise " Picture Eléments ": éléments d'image, le pixel est le plus petit point de l'image, c'est une entité calculable qui peut recevoir une structure et une quantification. Si le bit est la plus petite unité d'information que peut traiter un ordinateur, le pixel est le plus petit élément que peuvent manipuler les matériels et logiciels d'affichage ou d'impression. La lettre A, par exemple, peut être affichée comme un groupe de pixels dans la figure ci-dessous :

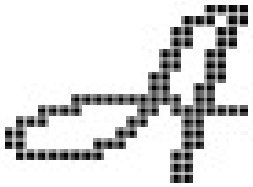


Figure 1.5 : La lettre A écrites avec un groupement de pixel**.**

La quantité d'information que véhicule chaque pixel donne des nuances entre images monochromes et images couleurs. Dans le cas d'une image monochrome, chaque pixel est codé sur un octet, et la taille mémoire nécessaire pour afficher une telle image est directement liée à la taille de l'image.

Dans une image couleur (R.V.B.), un pixel peut être représenté sur trois octets : un octet pour chacune des couleurs : rouge (R), vert (V) et bleu (B).

1.6.2 La texture

Une texture est une région dans une image numérique qui a des caractéristiques homogènes. Ces caractéristiques sont par exemple un motif basique qui se répète, ou des caractéristiques fréquentielles. Une texture est composée de Texel, l'équivalent des pixels.

1.6.3 Dimension

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image.

1.6.4 Résolution

La résolution est définie par un nombre de pixels par unité de longueur de l'image à numériser en dpi (dots per inch) ou ppp (points par pouce)]. On parle de définition pour un écran et de résolution pour une image.

Plus le nombre de pixels est élevé par unité de longueur de l'image à numériser, plus la quantité d'information qui décrit l'image est importante et plus la résolution est élevée (et plus le poids de l'image est élevé).

La résolution d'une image correspond au niveau de détail qui va être représenté sur cette image. Pour la numérisation il faut considérer les 2 équations suivantes :

(X*résolution) = **x** pixels

(Y*résolution) = **y** pixels

Où X et Y représentent la taille (pouce ou cm, un pouce=2,54 centimètres) de la structure à numériser, où résolution représente la résolution de numérisation, et où **x** et **y** représentent la taille (en pixels) de l'image.

1.6.5 Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur.

1.6.6 Histogramme

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (couleur) dans l'image. Pour diminuer l'erreur de quantification, pour comparer deux images obtenues sous des éclairages différents, ou encore pour mesurer certaines propriétés sur une image, on modifie souvent l'histogramme correspondant.

Il permet de donner un grand nombre d'information sur la distribution des niveaux de gris (couleur) et de voir entre quelles bornes est repartie la majorité des niveaux de gris (couleur) dans les cas d'une image trop claire ou d'une image trop foncée.

Il peut être utilisé pour améliorer la qualité d'une image (Rehaussement d'image) en introduisant quelques modifications, pour pouvoir extraire les informations utiles de celle-ci.

1.6.7 Contours

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative.

1.6.8 Luminance

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet.

1.6.9 Contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images.

Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste **C** est défini par le rapport :

$$C = \frac{L1-L2}{L1+L2} \quad \text{..... (1.1)}$$

1.6.10 La taille d'une image

Pour connaître la taille d'une image, il est nécessaire de compter le nombre de pixels que contient l'image, cela revient à calculer le nombre des cases du tableau, soit la hauteur de celui-ci que multiplie sa largeur. La taille de l'image est alors le nombre des pixels que multiplie la taille (en octet) de chacun de ces éléments.

Exemple : pour une image de 240 X 420 en TrueColor :

Nombre de pixels :

240 X 420 = 100800

Taille de chaque pixel : 24 bits / 8 = 3 octets

Le poids de l'image est ainsi égal à :

100800 X 3 = 302.400 égal 302.400/1024 = 295 Ko

1.7 Système de traitement d'image

Dans le contexte de la vision artificielle, le traitement d'images se place après les étapes d'acquisition et de numérisation, assurant les transformations d'images et la partie de calcul permettant d'aller vers une interprétation des images traitées. Cette phase d'interprétation est d'ailleurs de plus en plus intégrée dans le traitement d'images, en faisant appel notamment à l'intelligence artificielle pour manipuler des connaissances, principalement sur les informations dont on dispose à propos de ce que représentent les images traitées (connaissance du domaine).

a Acquisition et numérisation

C'est le mécanisme qui permet l'obtention d'une image numérique (représentée par une matrice) à deux dimensions à partir d'une scène à trois dimensions, en passant par un système optique, l'image continue f(x,y) est approximée par des échantillons qui sont obtenus par discrétisation des coordonnés (x,y) (ce qu'on appelle l'échantillonnage) et la discrétisation des amplitudes de ces points là (c'est la quantification).

La représentation obtenue ne peut être parfaite à cause du bruit introduit dans l'image lors de son acquisition [2].

b Visualisation

Un dispositif de visualisation permet l'affichage de l'image. Les restituteurs permettent de transformer le signal numérique qui est la matrice image en un signal analogique visible à l'œil humain pour cela on dispose d'une multitude de supports qui sont employés (moniteur vidéo, impression sur papier…) [2].

1.8 Filtrage

On peut scinder les filtres en trois grandes catégories :

1.8.1 Filtres linéaires

Les premières et les plus simples méthodes de filtrage sont basées sur le filtrage linéaire, chacun de ses opérateurs est caractérisé par sa réponse impulsionnelle h(x, y), l'expression qui relie les entrées avec les sorties est donnée par la relation suivante :

$$\mathbf{S}(i,j) = \Sigma\,\Sigma_{u,v}\,(E(i,j)\,.\,h(i-u,j-v)) = h(i,j)*E(i,j) \quad \text{.....(1.2)}$$

Où : u, v varient de moins l'infini à plus l'infini.

Le filtrage linéaire est un produit de convolution c à d une combinaison linéaire du voisinage du pixel concerné.

a Filtre moyenneur (lissage)

L'intensité du pixel considéré est remplacée par la moyenne des pixels de son voisinage, la taille de la zone (fenêtre) entourant le pixel est un paramètre important, plus cette dimension est grande, plus Sa sensibilité au bruit diminue, et le lissage devient important (le flou s'accentue).

Le filtre moyenneur est un filtre passe-bas c à d qu'il laisse passer les basses fréquences (les faibles changements d'intensité de l'image) et atténue les hautes fréquences (variations rapides) [2].

$$1/9 * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Masque du filtre.

Exemple :

Image originale.

Image filtrée.

Figure 1.6: Application du filtre moyenneur (lissage)

b Filtre gaussien

L'expression gaussienne en deux dimensions est donnée par :

$$G_0(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2-y^2}{2\sigma^2}\right) \quad \text{(1.3)}$$

L'intérêt de ce filtre est que l'on contrôle facilement le degré de filtrage à travers le paramètre **σ**. Le filtre gaussien donne plus d'importance aux pixels proches du pixel central, et diminue cette importance au fur et à mesure que l'on s'éloigne de celui-ci, il a les mêmes inconvénients que le filtre moyenneur c à d il dégrade les contours [2].

La discrétisation de ce filtre pour un **σ** égale à 0.6 donne le masque suivant :

1/16 *

1	2	1
2	4	2
1	2	1

Masque du filtre.

c Filtre rehaussement de contour

C'est un filtre passe haut c à d il met en évidence les changements rapides de l'intensité de l'image (les hautes fréquences) et laisse les zones uniformes inchangées (basses fréquences) [2].

-1	-1	-1
-1	9	-1
-1	-1	-1

Masque du filtre

Exemple :

Figure 1.7: Rehaussement de contour

1.8.2 Filtres non linéaires

Ils sont conçus pour régler les problèmes des filtres linéaires, sur tout pour ce qui concerne la mauvaise conservation des contours. Leur principe est le même que celui des filtres linéaires, il s'agit toujours de remplacer la valeur de chaque pixel par la valeur d'une fonction calculée dans son voisinage, la seule différence c'est que cette fonction n'est plus linéaire mais une fonction quelconque (elle peut inclure des opérateurs de comparaisons).

a Filtre médian

Sur un voisinage à huit, le nouveau niveau de gris du pixel centre est choisi comme étant la valeur médiane de tous les pixels de la fenêtre d'analyse centrée sur ce dernier. Son avantage est qu'il garde la netteté des éléments qui constituent l'image sans étaler les transitions [2].

19	30	12
15	18	11
26	16	14

→

19	30	12
15	16	11
26	16	14

30	26	19	18	16	15	14	12	11

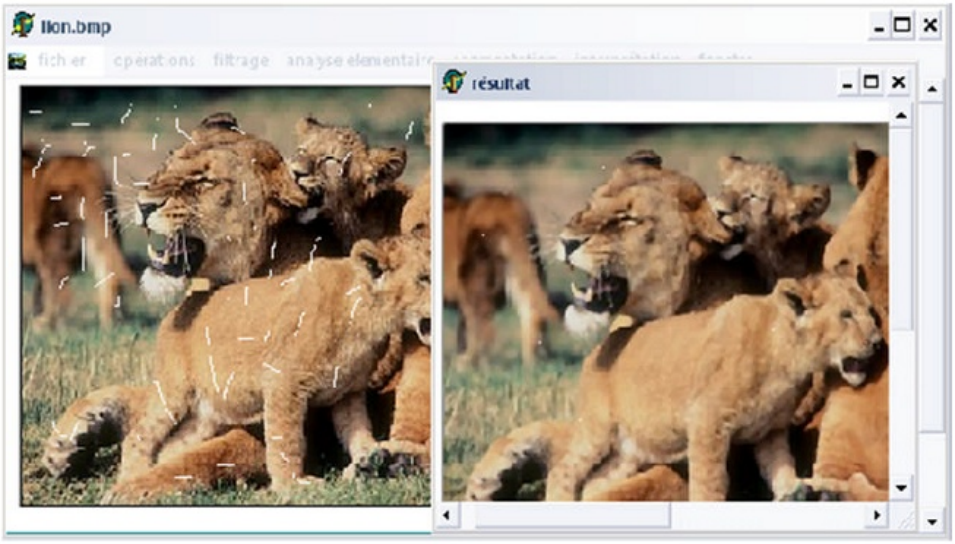


Figure 1.8 : Principe du filtre médian

b Filtre maximum

Même principe mais la valeur choisie est la valeur maximale.

19	30	12
15	18	11
26	16	14

→

19	30	12
15	30	11
26	16	14

30	26	19	18	16	15	14	12	11

Figure 1.9 : Principe du filtre maximum

c Filtre minimum

Même chose, sauf que la valeur choisie est la valeur minimale.

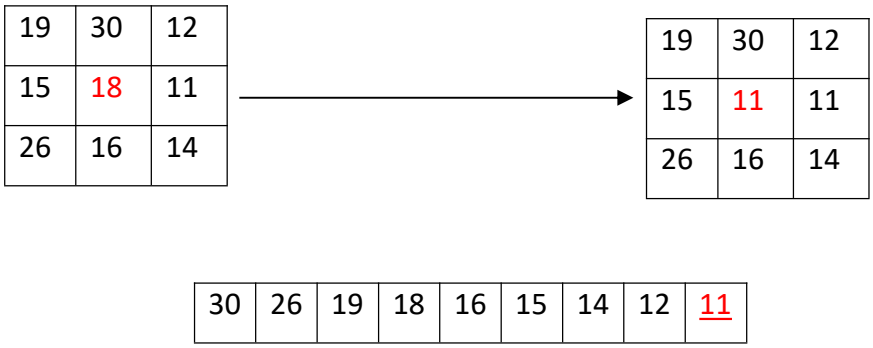


Figure 1.10 : Principe du filtre minimum

1.8.3 Les filtres morphologiques

Les filtres morphologiques sont souvent utilisés pour éliminer des pixels isolés considérés comme un bruit dans une image binarisée. Ces méthodes utilisent un élément structurant.

Parmi ces opérations morphologiques il y a la dilatation, l'érosion, l'ouverture et la fermeture mathématiques.

a La dilatation

Elle permet d'éliminer les pixels blancs isolés mais ajoute des pixels noirs au contour des objets présents dans l'image. Le résultat de cette opération est l'augmentation de la taille de ces objets.

b L'érosion

Elle permet d'éliminer les pixels noirs isolés au milieu des parties blanches de l'image. Le résultat de cette opération est la diminution de la taille des objets présents dans l'image.

c L'ouverture

L'ouverture est constituée par une opération d'érosion suivie d'une dilatation. Elle permet de retrouver les taches noires dans l'image.

d La fermeture

La fermeture est l'opération inverse de l'ouverture, qui consiste à faire subir à l'image une dilatation suivie d'une érosion. Elle permet d'éliminer les blancs qui se trouvent dans l'objet [3].

1.9 Le seuillage

Le seuillage ou binarisation consiste à transformer l'image codée sur 6, 8 ou 16 bits, en une image binaire code sur 1 bit.

Le seuillage permet de sélectionner les parties de l'image qui intéressent l'opérateur, par exemple 2 types de grains (blancs et gris) dans un mélange. On peut donc, par exemple, attribuer à tous les pixels de l'image numérique qui ont un niveau de gris compris entre deux valeurs $\mathbf{i_1}$ et $\mathbf{i_2}$, choisies par l'opérateur, la valeur 1; à tous les autres pixels est attribuée la valeur 0.

$$I_b(i,j) = \begin{cases} 1 & si\ \ I_b(i,j) \geq s \\ 0 & si\ non \end{cases} \ldots\ldots\ldots\ldots\ldots (1.4)$$

Après seuillage, les parties des images sélectionnées seront traduites en noir et blanc.

1.9.1 Seuillage global

La première solution pour fixer le seuil est de procéder par tâtonnement à partir de l'image en noir et blanc résultante censée mettre en évidence les contours. Le seuil correspond à une valeur réelle dans l'intervalle [0,1]. Au préalable, il est nécessaire de normaliser l'image du module du gradient afin que tous les pixels se trouvent également dans l'intervalle [0, 1]. Cette méthode est très simple mais peu efficace [4].

1.9.2 Seuillage local

Dans cette technique de seuillage contrairement aux précédentes, le traitement n'est pas identique en tout point de l'image. On s'intéresse ici aux pixels avoisinant les contours les plus significatifs de l'image. L'idée est de garder les contours les plus forts de l'image mais en essayant d'assurer leur continuité. Deux seuils sont nécessaires pour implanter la technique : un seuil haut **Sh** et un seuil bas **Sb**. Le seuil haut va servir à sélectionner les contours les plus significatifs dans l'image du module du gradient.

Ces contours sont contenus dans l'image résultante en noir et blanc. Le seuil bas permet de mettre en évidence des contours moins forts de l'image [5].Ces contours sont conservés dans l'image résultante seulement s'ils sont situés dans le voisinage des contours les plus significatifs mis en évidence par le seuillage avec **Sh**. Généralement le voisinage est défini par les huit voisins.

1.10 Segmentation

La segmentation d'image est une opération de traitement d'images qui a pour but de rassembler des pixels entre eux suivant des critères prédéfinis. Les pixels sont ainsi regroupés en régions, qui constituent un pavage ou une partition de l'image. Il peut s'agir par exemple de séparer les objets du fond. Si le nombre de classes est égal à deux, elle est appelée aussi binarisation.

Si l'homme sait naturellement séparer des objets dans une image c'est grâce à des connaissances de haut niveau (compréhension des objets et de la scène). Mettre au point des algorithmes de segmentation de haut niveau (chaque région est un objet sémantique) est encore un des thèmes de recherche les plus courants en traitement d'images.

La segmentation est une étape primordiale en traitement d'image. À ce jour, il existe de nombreuses méthodes de segmentation, que l'on peut regrouper en quatre principales classes :

1. Segmentation fondée sur les régions (en anglais : région-based segmentation). On y trouve par exemple : la croissance de région (en anglais : région-growing), décomposition/fusion (en anglais : split and merge).

2. Segmentation fondée sur les contours (en anglais : edge-based segmentation).

3. Segmentation fondée sur classification ou le seuillage des pixels en fonction de leu intensité (en anglais : classification ou thresholding).

4. Segmentation fondée sur la coopération entre les trois premières segmentations.

1.11 Quelques applications concrètes de traitement d'images

• Contrôle de présence/absence : Sur des chaînes de production, on vérifie en bout de chaîne avec une caméra vidéo la présence d'une pièce dans un ensemble plus complexe. Pour cela, bien souvent il suffit de faire un simple seuillage dans une région spécifique.

• Contrôle du niveau de maturation des fruits sur une chaîne de conditionnement. Il s'agit de reconnaître à la couleur et à la texture du fruit son degré de maturité et donc la catégorie sous laquelle il sera emballé puis vendu.

• Construction et correction de cartes géographiques d'après des images satellites ou des images aériennes. On recale d'après des informations topographiques les images reçues, puis on les met sur la carte en correspondance avec les informations trouvées dans l'image : voies de communication, voies et plans d'eau, parcelles agricoles...

• Surveillance et évaluation de la production agricole. Il est possible de déterminer le degré de maturation des cultures, la quantité d'eau nécessaire pour l'irrigation, le rendement moyen... On peut ainsi établir des prévisions à large échelle de la récolte à venir.

• Reconnaissance de l'écriture: La reconnaissance de l'écriture manuscrite progresse de jour en jour. Elle est suffisamment opérationnelle pour que la majorité des adresses, même manuscrites, soient reconnues automatiquement sur le courrier postal.

• Recherche d'image par le contenu: L'objectif de cette technique est de rechercher, parmi une base de données d'images, les images similaires à une image exemple, ou ayant certaines caractéristiques, par exemple rechercher toutes les images comportant un vélo.

1.12 Conclusion

On vu dans ce chapitre quelque notion sur la vision par ordinateur ainsi que quelque technique de traitement d'image. Nous avons détailles dans le chapitre suivant une des techniques de la reconnaissance des chiffres.

CHAPITRE 2 : Descripteur de forme et classification

2.1 Introduction

Parmi les objectifs de la vision assistée par l'ordinateur est la reconnaissance des formes. Elle consiste en une étape importante pour la mise en œuvre de plusieurs applications actuelles.

Par conséquent, il existe un intérêt croissant sur ce domaine de recherche ces dernier temps. Dans ce chapitre on détaille la chaine de reconnaissance des chiffres imprimés et manuscrits suivant l'organigramme présenté par Figure 2.1 [6].

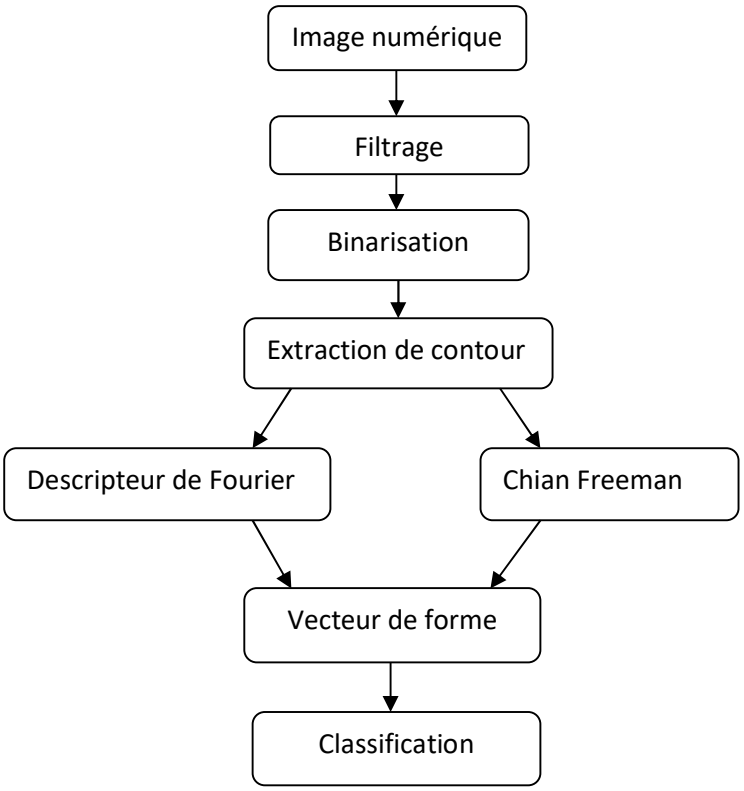


Figure 2.1: Schéma synoptique de méthode de traitement

2.2 L'image numérique

Le traitement d'images est basé essentiellement sur l'image numérique qui est l'ensemble des pixels placés sous forme matricielle.

2.3 Filtrage

2.3.1 Filtre moyenneur

Nous avons utilisé dans ce projet un filtre moyenneur qui a été représenté en détail au chapitre 1.

2.4 Binarisation

2.4.1 Définition

La binarisation ou seuillage est la technique de classification la plus simple où les pixels de l'image sont partagés par un seul seuil ***s*** en deux classes : ceux qui appartiennent au fond et ceux qui appartiennent à la scène (l'objet). L'image est alors séparée en deux classes de façon à ce que l'information comprise entre 0 et ***s*** est retenue et l'autre non, ou vice-versa.

Soit l'image I (M x N), supposons que f(x, y) représente le niveau de gris du pixel aux coordonnées (x, y), $\mathbf{0 \leq} x \leq M$, $\mathbf{0 \leq} y \leq N$ et ***s*** est le seuil choisi, les pixels de l'objet sont ceux ayant le niveau de gris inférieur à ***s*** et les autres ayant le niveau de gris supérieur à **s** sont des pixel du fond. Alors, l'image binarisée G de la figure 2.2 est déterminée par les pixels (x, y) dont la valeur est :

$$G(x,y) = \begin{cases} 1 & si \quad f(x,y) > s \\ 0 & si \quad f(x,y) \leq s \end{cases} \quad \text{..... (2.1)}$$

Image originale Image binarisée

Figure 2.2 : Image binarisée

Il existe trois grandes techniques de sélection du seuil ***s*** [7], global, local et dynamique. Comme il y a des différentes façons de déterminer le seuil s, il peut être considéré comme une fonction sous forme de $s = t\big((x,y), p(x,y), f(x,y)\big)$ où $p(x,y)$ représente des propriétés locales du point (x,y). Si ***s*** ne dépend de que la valeur $f(x,y)$ du point, le seuil est global, s'il dépend en plus de $p(x,y)$, s est un seuil local. Et si s dépend à la fois de(x,y), de $p(x,y)$ et de$f(x,y)$, on dit le seuil dynamique ou bien adaptatif.

Pour la binarisation locale, la classification d'un pixel dépend non seulement du pixel soi-même mais aussi de ses informations locales. Dans Cheng [8], c'est la moyenne des pixels du voisinage qui est prise en compte lorsqu'on construit l'histogramme de deux dimensions. Dans Cheng [9], les informations locales sont inclues dans le homogramme qui indique le degré d'homogénéité correspondant à chaque niveau de gris dans l'image. La détermination du seuil se base sur ce homogramme. Sachant l'importance des informations du voisinage pour la classification, Sue Wu et Adnan Amin [10] proposent une méthode de seuillage en deux étapes pour l'image de documents. Après l'étape de seuillage global sur l'image entière, le seuillage sur des sous-images qui contiennent des composants connectées est effectué. La méthode donne de bons résultats sur l'ensemble des images d'enveloppe postale.

Dans la méthode de binarisation globale un seuil unique est calculé à partir d'une mesure globale sur toute l'image. Il nous permet de décider l'appartenance d'un pixel à l'objet ou au fond. Les méthodes d'Otsu [11], de Kapur [12], de Pun [13], ou de Cheng et Chen [14] peuvent être tenues comme des représentants de cette approche. Chacun a de différentes stratégies pour atteindre leur but. Par exemple, la méthode décrite dans d'Otsu essaie de maximiser la variance entre deux classes, tandis que d'autres méthodes dans Kapur, Pun, Cheng et Chen, Mello [15] se basent sur la théorie de maximum d'entropie ou d'entropie floue.

2.4.2 Méthode d'Otsu

Dans la méthode d'Otsu, le seuil qui minimise la variance inter-classe est recherché à partir de tous les seuillages possibles :

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \text{..... (2.2)}$$

Les poids ω_i représentent la probabilité d'être dans la n^{ieme} classe, chacune étant séparée par un seuil ***t.*** Finalement, les σ_i^2 sont les variances de ces classes.

Otsu montre que minimiser la variance inter-classe revient à maximiser la variance intra-classe:

$$\sigma_b^2(t) = \sigma^2 - \sigma_\omega^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \text{..... (2.3)}$$

Qui est exprimée en termes des probabilités de classe ω_i et des moyennes de classes μ_i qui à leur tour peuvent être mises à jour itérativement. Cette idée conduit à un algorithme efficace.

2.4.3 Les étapes de la méthode

Cette méthode de binarisation nécessite au préalable le calcul de l'histogramme. Puis la séparation en deux classes est effectuée.

a Calcul de l'histogramme

Le calcul de l'histogramme est très simple.

On initialise un tableau hist avec des 0 Généralement, ce tableau est constitué de 256 cases correspondant aux 255 niveaux de gris d'une image.

En suite, si **p (i, j)** fréquence représente la valeur du pixel au point **(i, j)**, on balaye toute l'image et on compte le nombre de fois ou un niveau de gris apparait.

b Séparation en deux classes

La séparation se fait à partir des moments des deux premiers ordres : la moyenne et l'écart type. Pour que le procédé soit indépendant du nombre de points dans l'image N, on normaliser l'histogramme : $\mathbf{p_i = n_i/N}$ ou $\mathbf{n_i}$ représente le nombre de pixels de niveau **i**.

On peut calculer alors les deux moments utilisés :

$$\mu(k) = \sum_{i=1}^{i=k} i * P_i \qquad et \qquad \omega(k) = \sum_{i=1}^{i=k} P_i \text{..... (2.4)}$$

On note $\mu_T = \mu(256\text{-}1)$, où 256 est le nombre totale de niveaux de gris.

Si on appelle ω_0 la probabilité de la classe C_0 et ω_1 la probabilité de la classe C_1 alors:

$\omega_0 = \omega\ (k^*)$ où **k*** représente le niveau de seuil.

Et

$$\omega_1 = 1 - \omega\ (k^*)$$

Si on note de même μ_1 et μ_0 avec :

$$\mu_0 = \mu\ (k^*)/\ \omega\ (k^*) \text{ et } \mu_1 = (\mu_T - \mu\ (k^*))/\ (1 - \omega(k^*)) \text{..... (2.5)}$$

Or l'image totale conserve certaines propriétés, d'où on peut tirer les relations :

$$\omega_0 * \mu_0 + \omega_1 * \mu_1 = \mu_T \quad et \quad \omega_0 + \omega_1 = 1$$

En introduisant un paramètre pour évaluer la qualité du niveau de seuillage, on obtient :

$$S^2 = \omega_0 * \omega_1 (\mu_1 - \mu_0)^2 \text{..... (2.6)}$$

La valeur précédente est fonction de **k**.

On calcule donc cette valeur pour les 256 niveaux de gris de l'image. En fait on peut déjà enlever les valeurs 0 et 255 qui correspondent à affecter tous les pixels à la même classe.

A partir de ω(k) et μ(k) on calcule donc :

$$S^2\ (k) = (\mu_T * \omega(k) - \mu\ (k))^2\ /\omega\ (k) * (1 - \omega\ (k)) * \text{..... (2.7)}$$

La valeur du seuil **k*** est obtenue pour le maximum de S^2.

Il ne reste plus qu'à comparer la valeur de tous les pixels de l'image au seuil ainsi trouvé. Les résultats de l'application de cette méthode sont représentés par la (Figure 2.3).

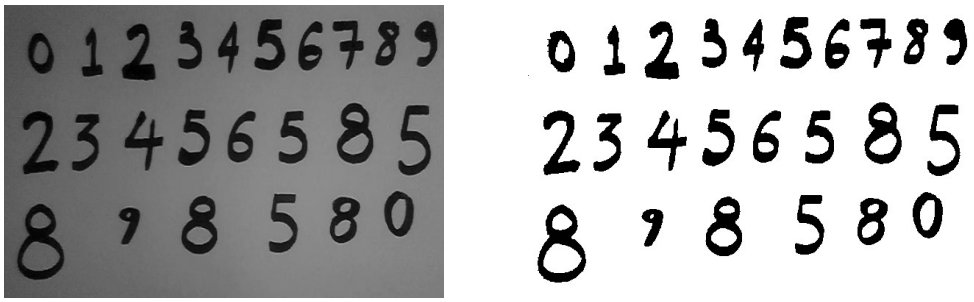


Image initiale. Image binaire.

Figure 2.3 : Image binarisée par Otsu.

On remarque la présence d'un bruit important dans le fond de l'image. Cependant on arrive à distinguer convenablement les caractères [16].

2.5 Algorithme du suivi de contour

Cet algorithme est très rapide car il ne nécessite que peu de calculs à chaque étape. Il présente l'avantage de détecter les pixels de frontières suivant toutes les directions. Tous les pixels seront traités une seule fois [17].

L'algorithme de suivi de contour permet de détecter les contours d'objet d'images, Il pouvant être représentés de la façon suivant :

- Les objets sont constitués de pixels « noirs », leur valeur « 0 » sera attribués, tandis que l'extérieur (fond) est formé de pixels « blancs » de valeur « 1 ».

Ces contours sont classés selon deux catégories :

- Ceux qui englobent un objet. Ils sont dits extérieurs, c'est la silhouette de l'objet.
- Ceux qui sont noyés dans un objet, ce sont les contours intérieurs.

Apres la détection du premier pixel contour, la nature de la transition détermine sont type :

- Si cette transition est de nature fond forme (1 – 0), c'est un contour de type externe.
- Sinon (0 – 1), le contour est de type interne voir (Figure 2.4)

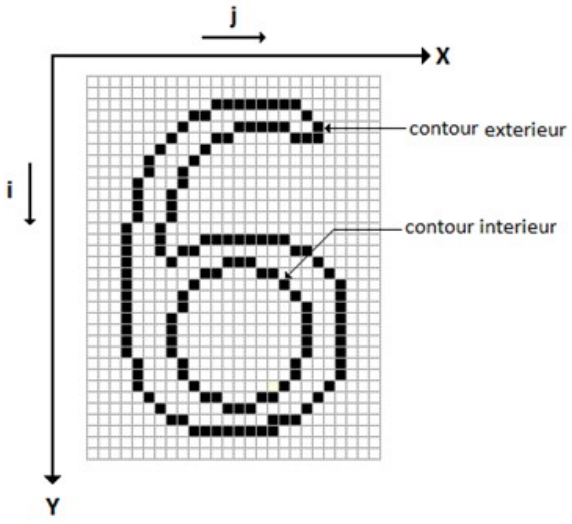


Figure 2.4 : Image de référence

Cette méthode se faire à balayer l'image jusqu'à trouver une transition. C'est le premier pixel de contour.

Etiqueter les huit voisinages de ce pixel. Tester si ces pixels appartiennent au contour. À chaque fois qu'un pixel objet est rencontré, on lui étiquette ses huit voisinages et continuer jusqu'à la rencontre du premier pixel trouvé.

2.5.1 Implémentation de la méthode

Les principales étapes de l'algorithme sont les suivantes :

1er étape :

Balayage de l'image jusqu'à trouvé le premier pixel objet.

Premier pixel contour à (i_1,j_1) Le pixel qui le précède est noté par (i_D,j_D).

2eme étape :

A partir du pixel (i_D,j_D) en tournant autour du pixel (i_1,j_1) de façon à balayer ses voisins dans le sens horaire, on numérote les sept autres voisins du pixel (i_1,j_1) comme 2 ,……,8.

3eme étape :

Evaluer les coordonnées **Lx(k)**, **Ly(k)** du **k^{ieme}** voisin de **(i_1,j_1)** en utilisant la table de la (Figure 2.5)

	Lx(k)	Ly(k)
1	i_d	j_d
2	Lx(1)+k1	Ly(1)-k2
3	Lx(2)-k2	Ly(2)-k1
4	Lx(3)-k2	Ly(3)-k1
5	Lx(4)-k1	Ly(4)+k2
6	Lx(5)-k1	Ly(5)+k2
7	Lx(6)+k2	Ly(6)+k1
8	Lx(7)+k2	Ly(7)+k1

Figure 2.5 : coordonnés des 8 voisins

Coordonnées du pixel contour **(i_1, j1)**

Coordonnées du 1er pixel voisin **(i_d, jd)**

K1 = $j_d - j_1$

K2 = $i_d - i_1$

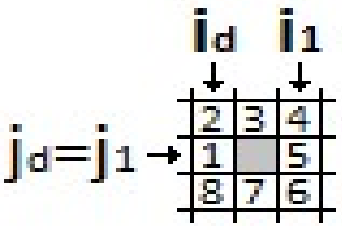


Figure 2.6 : Définition du point contour

4eme étape :

Si le **k^{ieme}** voisin est un point noir, il sera le point suivant du contour et on définit **(i_1,j_1)** en ce point. **(i_d,j_d)** sera le **(k – 1) ^{ieme}** point de la table aller à l'étape 2.

5eme étape :

Si le **k^{ieme}** voisin est un pixel blanc, prendre k=k+1 et aller à l'étape 3.

6eme étape :

Continuer le processus jusqu'à la rencontre du premier point contour détecté.

7eme étape :

Affecter à tous les points contours découverts une valeur différente de 0 et de 1 comme intensité du contour et numéro du contour. Cela permet d'éviter la détection du même contour une notre fois.

8eme étape :

Poursuivre le balayage de l'image pour détecter de nouveaux contours jusqu'à la fin de l'image.

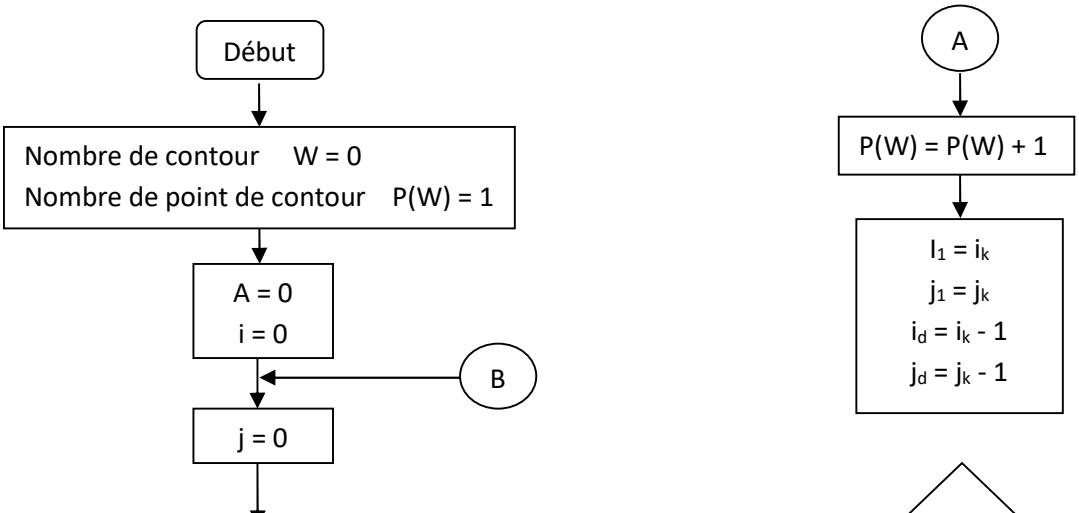
2.5.2 Tableau comparatif

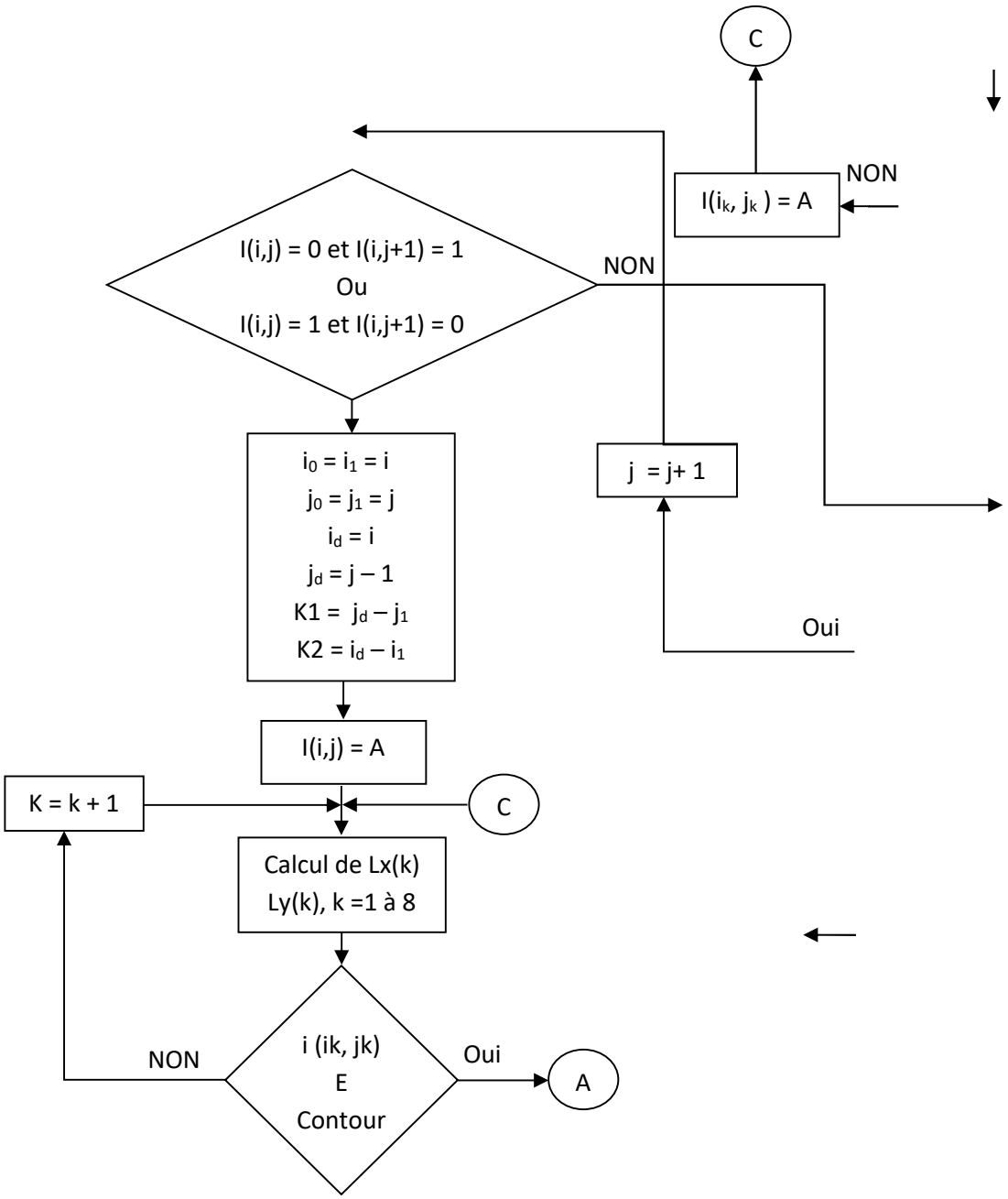
L'extraction du contour d'un objet est une méthode très important dans le traitement d'image, il y à plusieurs méthodes de l'extraction du contour et pour cela en fait une comparaison entre trois méthodes sur la vitesse d'exécution, sensibilité au bruit et difficulté d'implémentation. Le tableau suivant définie cette comparaison.

Méthode	Vitesse d'exécution	Sensibilité au bruit	Difficulté d'implémentation
Gradient Laplacien Suivi du contour	Rapide Moin rapide Très rapide	Très sensible Sensible Moin sensible	Moin facile Difficile Facile

Tableaux 2.1 : Tableau comparatif de différente méthode

Organigramme de la détection de contour





2.6 Descripteur de forme

Les descripteurs (ou paramètres) de forme sont des nombres qui représentent chaque forme et permettent de les classer. La forme est généralement une description très riche d'un objet. L'extraction d'attribut géométrique a été le fer de lance de la recherche d'image devant ces dernières années. De nombreuses solutions ont été proposées pour représenter une forme, nous distinguons deux catégories de

descripteurs de formes : les descripteurs basés sur les régions et les descripteurs basés sur les frontières.

Les premiers font classiquement référence aux moments invariants et sont utilisés pour caractériser l'intégralité de la forme d'une région. Ces attributs sont invariants aux transformations géométriques comme la translation, la rotation et le changement d'échelle. La seconde approche fait classiquement référence aux descripteurs de Fourier et porte sur une caractérisation des contours de la forme.

2.6.1 Les moments géométriques

Les moments géométriques [18] permettent de décrire une forme à l'aide de propriétés statistiques. Ils représentent les propriétés spatiales de la distribution des pixels dans l'image. Ils sont facilement calculés et implémentés. Par contre, cette approche est très sensible au bruit et aux déformations et le temps de calcul de ces moments est très long.

2.6.2 Descripteur de Fourier

L'une des techniques les plus promettant de description de forme est basée sur les descripteurs de Fourier.

Supposons que nous avons **N** points de la frontière d'une région. Nous pouvons considérer la région comme placée dans le plan complexe, avec l'axe des ordonnées comme axe imaginaire et l'axe des abscisses comme l'axe réel. Alors les coordonnées **x, y** de chaque point du contour à analyser peuvent être représentées comme des nombres complexes (x+jy).

Le contour peut être écrit sous la forme de séquence de nombres complexes.

$$Z_i = x_i + jy_i \qquad i = 0,1,2, \ldots, N-1$$

La séquence est périodique pour chaque complet du contour. Les descripteurs de Fourier sont définis comme suit :

$$A(u) = 1/N \sum_{i=0}^{N-1} Z_i \exp[2\pi ui/N] \qquad u = 0,1,2, \ldots, N-1 \ldots\ldots\ldots (2.8)$$

La transformation étant réversible

$$Z_i = \sum_{u=0}^{N-1} A(u) \exp[j2\pi ui/N] \qquad i = 0,1,2, \ldots, N-1 \ldots\ldots\ldots (2.9)$$

Avant d'utiliser les descripteurs de Fourier il faut éliminer leur dépendance de la position, la dimension, l'orientation et le point de départ, comme suit :

1 - Le changement de la position du contour altère A(0) seulement.

2 - En multipliant la dimension du contour par une constante, les descripteurs de Fourier sont multipliés par une constante également.

3 - La rotation du contour dans le domaine spatial requiert la multiplication de chaque coordonnées par $\mathbf{exp}\,(\mathbf{j\theta})$ où $\boldsymbol{\theta}$ est l'angle de rotation. La linéarité de la transformée de Fourier fait que la multiplication des coefficients du domaine fréquentiel produit le même effet.

4 - Le changement du point de départ du contour dans le domaine spatial (temporel) correspond à multiplier le k^{ieme} coefficient du domaine fréquentiel par $\mathbf{exp}\,(\mathbf{jkT})$ où $\mathbf{T}$ est une fraction de période par laquelle le point de départ a été décalé ($\mathbf{T}$ varie de 0 à 2π , le point de départ traverse le contour une fois).

On normalise les DF en prenant **A(0)** égal à 0 (pour rendre les DF indépendantes de la position) et on divise chaque coefficient par l'amplitude de la premier DF A(1) (pour normaliser leur taille).

2.6.3 Algorithme utilisant les descripteurs de Fourier

Soit (x(m), y(m)) les coordonnées d'un pixel appartenant au contour ; avec $0 \leq m \leq L\text{-}1$

En prenant la transformée de Fourier discrète (DFT) de la donnée $\big(x(m), y(m)\big)\ 0 \leq m \leq L-1$, on obtient les coefficients de Fourier pour $0 \leq k \leq L-1$

$$A(k) = 1 \Big/ L \sum_{m=0}^{L-1} Z(m) \exp\,(-j2\pi km/L\,) \ldots\ldots (2.10)$$

$$\exp\,(\mathrm{j}\theta_0) = \cos\theta_0 + \mathrm{j}\sin\theta_0 \ldots\ldots (2.11)$$

$$\mathrm{Z(m)} = \mathrm{x(m)} + \mathrm{jy(m)} \ldots\ldots (2.12)$$

D'après l'équation (2.11) et (2.12) :

$$A(k) = 1/L \sum_{m=0}^{L-1} [x(m)+jy(m)][\cos(2\pi km/L) - j\sin(2\pi km/L)] \ldots (2.13)$$

$$a(k) = Re(A(k)) = 1/L \sum_{m=0}^{L-1} [x(m)*\cos(2\pi km/L) + y(m) * \sin(2\pi km/L)] \ldots (2.14)$$

$$b(k) = Im(A(k)) = 1/L \sum_{m=0}^{L-1} [-x(m)*\sin(2\pi km/L)] + y(m) * \cos(2\pi km/L)] \ldots (2.15)$$

On distingue les composants DC $a(0)$ et *b(0)* puisqu'il donne l'information seulement sur la position du centre de l'image, On laisse

$$|A(k)| = \sqrt{|a(k)|^2 + |b(k)|^2} \ldots\ldots\ldots\ldots\ldots (2.16)$$

$$S(k) = A(k)/A(1) \qquad k = 1, \ldots, L-1 \ldots\ldots\ldots\ldots (2.17)$$

Ou $|a(k)|, |b(k)|$ de note la valeur absolue des nombres complexes $a(k)$et $b(k)$. Alors, il devient facile de voir que $A(k)$ est invariant pour la rotation, la translation et plus loin $S(k)$ est invariant pour le changement d'échelle.

Les descripteurs invariants s(k) sont appelés descripteurs de Fourier.

Noter que pour L_1, $L_2 < e_1 < e_2 <$ (L-1) /2 S(1), ... S (e_1) sont les basses fréquences de l'image alors que S (e_2), ..., S [(L-1)/2] sont les hautes fréquences, où [(L-1)/2] représente la part entière de (L-1)/2. La part de basses fréquences de S (k) est déterminée par la forme d'un objet, alors que les hautes fréquences donnent les détails. Puisque les hautes fréquences sont dégradables par le bruit et ne contribuent pas beaucoup à la reconnaissance de l'objet composé aux basses fréquences, souvent on les ignore. Cette compression significative de la donnée est l'une des raisons de l'utilisation des descripteurs de Fourier.

Dans une étude d'une image qui contient plus de 500 points, seulement les 32 premiers composants (S [1], ..., S [32]) ont été utilisés comme vecteur de forme sur plus de 500 coefficients.

2.7 La Chaine Freeman

C'est la méthode la plus ancienne de description des contours dans les images et aussi la plus utilisée encore aujourd'hui [19][20]. Si elle est moins utilisée en reconnaissance des formes qu'elle le fut, elle est très fréquemment utilisée dans des applications récentes comme la transmission des images par zones (par exemple dans MPEG).

2.7.1 Définition

Le codage de Freeman [20] sont générées par localisation du pixel de contour, également il fixe le premier pixel de départ, ensuite il déplace au long de la frontière de contour soit dans le sens horaire ou antihoraire, cherche le pixel suivant du contour et assigne le nouveau pixel par un code en fonction de son emplacement par rapport au pixel précédent. La procédure cherche le pixel suivant est se termine lorsque le pixel de départ est rencontrée.

Les codes peuvent être 4-directionnel ou 8-directionnel selon 4-connectivité ou 8-connectivité d'un pixel à son pixel voisin de contour. Une chaîne 8-directionnelle d'une image codé est illustrée par la (Figure 2.7).

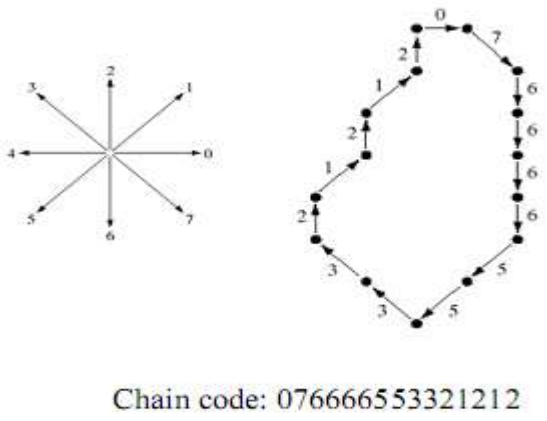


Figure 2.7 : chaîne code de 8-connectivité

On constate que la chaîne code pour différents chiffres donne différentes longueurs du code et chaque longueur de chaîne code dépend de la taille de la référence de l'objet.

Dans certain cas la taille de la référence de l'objet est très élève, et pour résoudre se problème il faut normaliser la chaîne code comme suit :
Supposer que cette chaîne code est généré pour un contour traversant en sens anti horaire (sens horaire).

V1= [1 2 2 2 1 2 1 2 1 1 2 0 1 0 1 0 0 0 0 0 0 7 7 0 0 0 1 0 0 0 0 0 0 0 0 7 0 0 7 0 7 6 6 6 7 6 6 7 6 6 6 6 6 7 6 6 6 6 5 6 6 6 5 6 6 5 5 6 5 4 4 5 4 5 4 5 4 4 4 5 4 4 4 4 4 3 4 4 4 3 4 4 3 4 4 3 4 4 3 2 3 3 2 3 2 3 2 3 2 2 2 2 2 2 2 2]

Calculer la fréquence des codes 0, 1, 2 ..... 7, du vecteur V1 pour obtenir le vecteur

fréquence V2 comme ci-dessous.

V2= [23 8 19 10 20 9 20 8]

La fréquence normalisée représenté par le vecteur V3, calculé par la formule suivant :

$$V3 = \frac{V2}{|V1|} \qquad |V1| = \sum V2 \ldots\ldots\ldots\ldots (2.18)$$

Pour l'exemple considéré ci-dessus, on a :

V3= [0.1966 0.0684 0.1624 0.0855 0.0015 0.0769 0.1709 0.0684]

Finalement, la concaténation de la vecteur V2 et V3 on obtient un vecteur caractéristique de taille 16 comme ci-dessous.

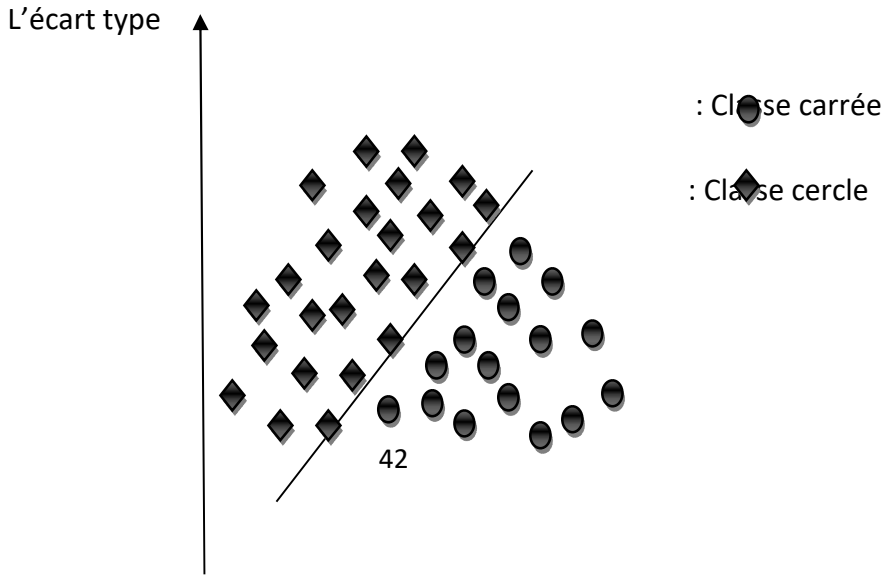
V4= [23 8 19 10 20 9 20 8 0.1966 0.0684 0.1624 0.0855 0.0015 0.0769 0.1709 0.0684]

2.8 Classification

On distingue principalement deux types de classification

2.8.1 Méthode statistique

L'extraction des statistiques produit des valeurs numériques qui son confrontées aux modèles statistique caractérisant chaque classe.



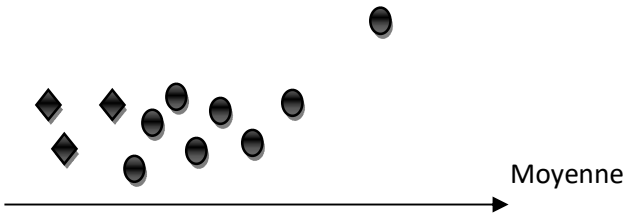


Figure 2.8 : Classification entre deux classes

2.8.2 Méthode syntaxique (structurelle)

L'extraction des primitives produit des valeurs symboliques des relations qui font l'objet d'une analyse structurelle ou syntaxique.

2.8.2.1 Mesure de distance entre deux points

Comment mesurer la distance entre deux points $d(x_1, x_2)$:

- **Distance Euclidienne** :

$$d^2(x_1, x_2) = \sum_i (x_1 - x_2)^2 \text{ ..... (2.19)}$$

- **Distance de Manhattan :**

$$d(x_1, x_2) = \sum_i |x_1 - x_2|^2 \text{ ..... (2.20)}$$

- **Distance de Sebstyen :**

$$d^2(x_1, x_2) = (x_1 - x_2).\omega.(x_1 - x_2)\grave{} \text{..... (2.21)}$$

ω : Matrice diagonal de Pondération

- **Distance de Mahalanobis :**

$$d^2(x_1, x_2) = (x_1, x_2)C^{-1}(x_1, x_2)\grave{} \text{..... (2.22)}$$

C : Matrice de covariance

2.8.2.2 Mesure de distance entre deux classes :

Comment mesurer la distance entre deux classes D (C_1, C2)

- **Distance de plus proche voisin (Nearest Neighbors : NN**)

$$\min(d(j,j), i \in C_1, j \in C_2) \text{..... (2.23)}$$

- **Distance maximum :**

$$\text{Max}(d(j,j), i \in C_1, j \in C_2) \text{..... (2.24)}$$

- **Distance moyenne :**

$$\frac{\sum d(i,j)}{n_1 n_2} \text{..... (2.25)}$$

- **Distance des centres de gravité :**

  $d\,(u_1, u_2)$ u_1 : Vecteur moyen de la classe 1

- **Distance de Ward** :

$$\sqrt{\frac{n_1 n_2}{n_1+n_2}} d\,(u_1, u_2) \text{ ..... (2.26)}$$

- **Règle du plus proche voisin**

On note par t_k^i = [t_{k1}^i , t_{k2}^i ,....t_{km}^i] le k^{ieme} vecteur caractéristique dans la classe i pour 1≤m≤16 on note aussi par le $\bar{t}_m^i$ et σ_m^i la valeur de la moyenne et l'écart type de la m^{ieme} composante du vecteur caractéristique dans la classe i respectivement

En plus on aura : $\bar{t}_m^i$= [: $\bar{t}_1^i$: $\bar{t}_2^i$..... :$\bar{t}_{16}^i$]

Qui représente le vecteur caractéristique moyen de la classe (Feature vector)

La distance entre une forme de test et la base de données se calcule comme suit :

X=[x_1,x_2.....x_{16}]T et t_k^iest donnée par :

$$d(X,t_k^i)=\sum_{m=0}^{16} \frac{\left|x(m)-\bar{t}_m^i\right|}{\sigma_m^i} \text{..... (2.27)}$$

K : nombre de motif par classe

σ_{tm}^i : L'écart type de m^{ieme} composent de la classe i

Avec la règle du plus proche voisin, la classe i est assignée ou affectée a X si le vecteur caractéristique dans la classe i a le minimum de distance a partir de X parmi toutes les formes d'apprentissage (training).

(Le nombre de classe X est le nombre de motif) c a d

X classe i si $d(X,t_k^i) \leq d(X,t_p^j)$

$1 \leq j \leq M$ et $1 \leq p \leq K$

Avec M le nombre de classes et K le nombre de motifs par classe.

Mais avec la règle de distance moyenne minimale, la classe i est affectée à X, si le vecteur caractéristique moyen de la classe i a la distance minimale à partir de X parmi tous les vecteurs moyens :

C a d :

X classe i, si $d(X,\bar{t}_k^i) \leq d(x,\bar{t}_p^j)$ tel que $0 \leq j \leq M$

$$d(X,\bar{t}_k^i) = \sum_{m=0}^{16} \frac{\left| x(m) - \bar{t}_m^i \right|}{\sigma_m^i} \quad \text{..... (2.28)}$$

2.9 Conclusion

Nous avons présenté dans ce chapitre les différentes étapes essentielles que constitue notre chaine de reconnaissance, notamment, le filtrage, la binarisation, le suivi de contour, les descripteurs de Fourier et finalement le module de classification.

CHAPITRE 3 : Implémentation et résultats

3.1 Introduction

Dans ce dernier chapitre nous commençons tout d'abord par la représentation des logiciels et matériels utilisés pour développer ce projet. Ensuite nous présentons les tests et résultats expérimentaux.

3.2 Mise en situation

3.2.1 Environnement matériel

Afin de mener à bien ce projet, il a été mis à notre disposition un ensemble de matériels dont les caractéristiques sont les suivantes : un ordinateur Acer avec les caractéristiques suivantes :

– Processeur : Intel® CPU T2130 @ 1.86Ghz

– RAM : 1.00 Go de RAM

– Carte graphique : 256 Mb

3.2.2 Environnement logiciel

- **Logiciel de développement :**

Microsoft Builder C++ 6

- **Bibliothèque graphique :**

Open CV (Open Computer Vision)

3.3 Builder C++

Le Builder C++ est un environnement de programmation visuel orienté objet pour le développement rapide d'applications (RAD). En utilisant Builder C++, nous pouvons créer des applications Windows 32 bits très efficaces, avec un minimum de codage manuel. Le Builder C++ fournit tous les outils qui sont nécessaires pour développer, tester, déboguer et déployer des applications, incluant une importante bibliothèque de composants réutilisables, un ensemble d'outils de conception, des modèles d'applications et de fiches, ainsi que des experts de programmation. Ces outils simplifient le prototypage et réduisent la durée du développement.

Pourquoi le Builder C++6 ?

- Développé des programmes Windows C++ avec plus de facilité et de rapidité qu'auparavant.
- Crée des applications console Win 32 ou des programmes Win 32 GUI (Interface Utilisateur Graphique).
- Analyse, exploration et visualisation des données.

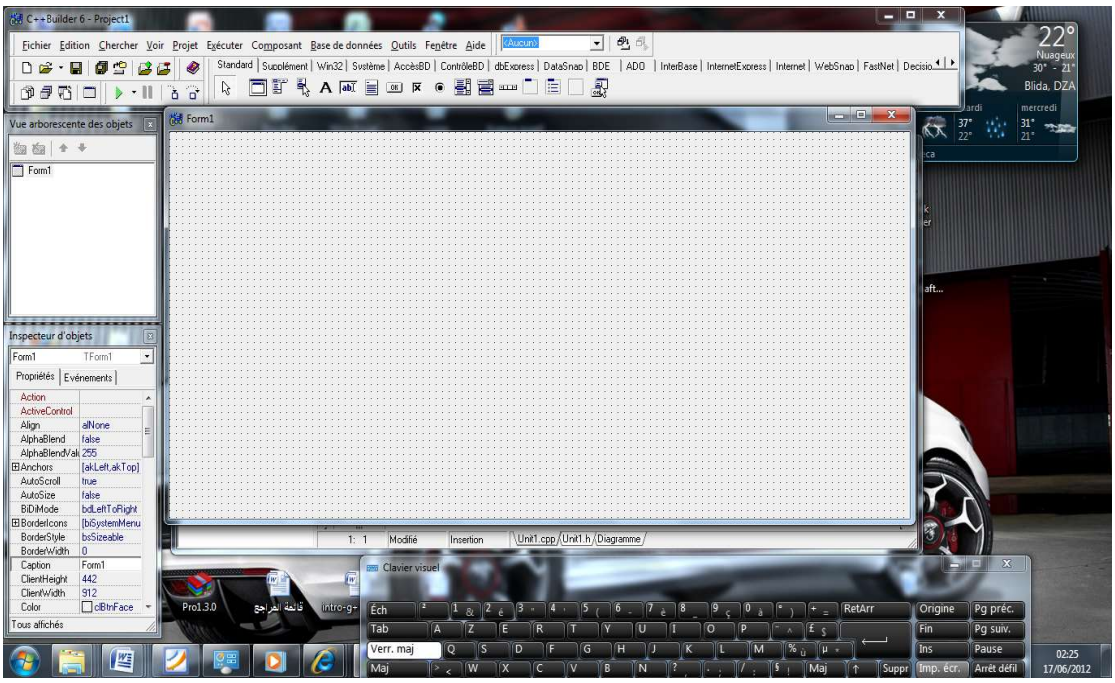


Figure 3.1 : Le langage Builder C++ 6

3.4 La librairie OpenCv

Un des objectifs buts d'OpenCV est d'aider les utilisateurs à construire rapidement des applications sophistiquées de vision à l'aide de simples opérations de vision par ordinateur.

OpenCV (Open Source Computer Vision) est une bibliothèque proposant un ensemble de plus de 2500 algorithmes de vision par ordinateur, elle a était écrite en C et C++, puis des interfaces ont été développées pour Python, Ruby, Matlab et autre langages. Elle est distribuée sous une licence BSD (libre) pour les plates-formes Windows, GNU/Linux, Android et MacOS. Cette bibliothèque est orientée vers des applications en temps réel [21].

Initialement écrite en C il y a 10 ans par des chercheurs de la société Intel, OpenCV est aujourd'hui développée, maintenue, documentée et utilisée par une communauté de plus de 40 000 membres actifs. C'est la bibliothèque de référence pour la vision par ordinateur, aussi bien dans le monde de la recherche que celui de l'industrie.

3.4.1 Présentation d'OpenCV

OpenCV (Open Computer Vision) est une bibliothèque libre de visualisation en temps réel pour le langage C/C++, elle idéalement les optimisations IPP (Integrated Performance Primitives) d'Intel mais fonctionne aussi très bien sans. Les domaines d'utilisation sont variés : IHM, robotique, détection et reconnaissance d'objets ou visage ainsi que suivi et étude de leurs mouvements.

OpenCV est optimisée pour les processeurs multi-cœurs, comme elle est multiplateformes c'est-à-dire disponible pour L'Unix, Windows, Mac OS X) et gratuite sous licence BSD.

3.4.2 Caractéristiques d'OpenCV :

- Image de manipulation de données (répartition, les sorties, la copie, la création, conversion).

• Image et vidéo I/O (fichier et l'appareil photo d'entrée en fonction, image/fichier de sortie vidéo).

• La manipulation de matrices et vecteurs et routines d'algèbre linéaire (valeurs propres produits, solveurs, SVD).

• Diverses structures de données dynamiques (listes, les files d'attente, ensembles, arbres, graphes).

• Traitement de l'image de base (filtrage, détection de contour, de détection d'angle, l'échantillonnage et l'interpolation, de conversion des couleurs, des opérations morphologiques, histogrammes, pyramides d'images).

• L'analyse structurale (composantes connexes, le traitement du contour, transformée de distance, des moments différents, correspondant modèle, transformée de Hough, approximation polygonale, montage en ligne, ellipse montage, triangulation de Delaunay).

• Calibration caméra et de repérage mires de calibrage, le calibrage, estimation de la matrice fondamentale, l'estimation homographie, la correspondance stéréo.

• Analyse du mouvement (flot optique, segmentation de mouvement, suivi).

• Reconnaissance d'objets (Eigen-méthodes, HMM).

• Basic GUI (l'image d'affichage / vidéo, le clavier et la manipulation de la souris, des barres de défilement).

• Image d'étiquetage (ligne, coniques, des polygones, dessin de texte).

❖ Organisation logicielle

Les programmes décrits ont été développés en C++ Builder 6. Ce langage s'est avéré puissant et offre un graphisme de qualité.

Vu également la rapidité de C++ grâce à la disponibilité d'une large collection d'APIs (Application Programming Interface), ce langage nous semble le plus adéquat pour le développement de cette application et la création de son interface graphique.

3.5 Interfaces de l'application

3.5.1 Interface principal

Cette interface représente le menu principal de l'application que nous avons réalisé dans ce projet.

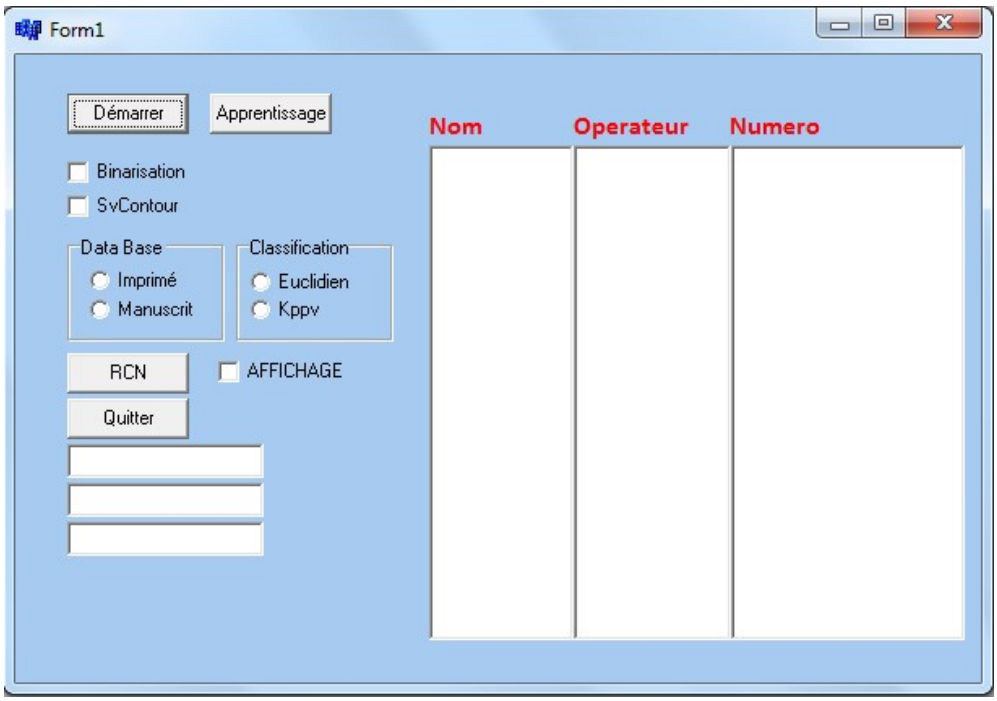


Figure 3.2 : Interface principale de programme

3.5.2 Menu de commande

Ce menu compose de quatre boutons simples, quatre boutons radio, trois check box et trois afficheurs (édit) voir la figure 3.3

1- Bouton pour démarrer la Cam.
2- Check box de binarisation voir figure 3.7

3- Check box pour l'extraction de contour voir figure 3.8

4- Bouton radio pour choisir la data base pour les chiffres imprimer.

5- Bouton radio pour choisir la data base pour les chiffres manuscrits.

6- Bouton de démarrer la reconnaissance des chiffres voir figure 3.9

7- Bouton pour quitter l'application.

8- Afficheur pour la valeur de seuil pour la binariastion d'Otsu voir figure 3.3

9- Afficheur pour afficher le numéro qui déjà capter voir figure3.3

10- Afficheur pour afficher le numéro sous qui est dans le rectangle voir figure 3.3

11- Bouton pour l'apprentissage voir figure 3.5

12- Bouton radio pour choisir la classification Euclidienne

13- Bouton radio pour choisir la classification K-la plus proche voisin

14- Check box pour pointer le numéro au tableau d'affichage voir figure 3.4

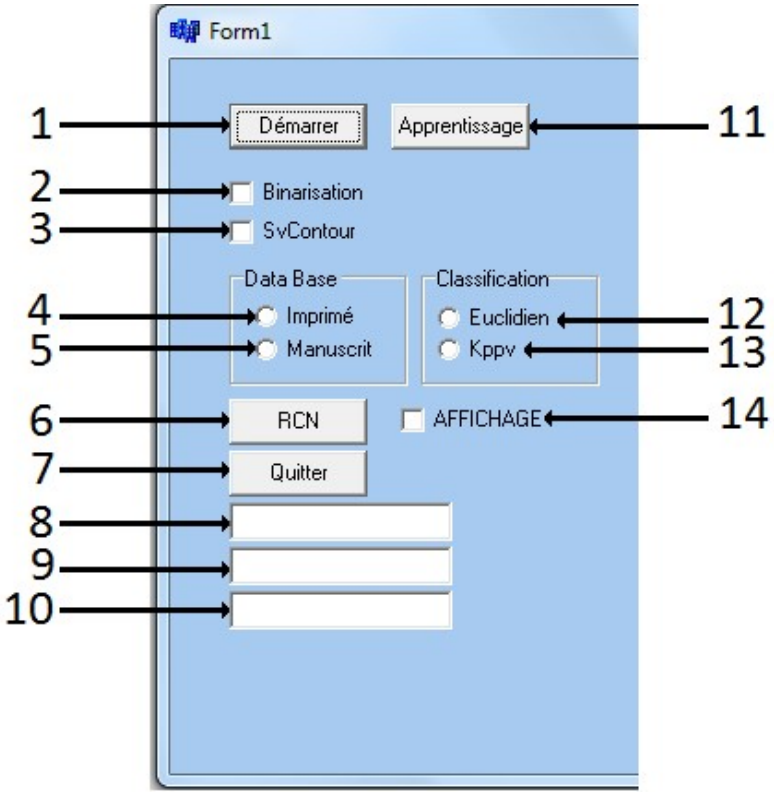


Figure 3.3 : Menu de commande

3.5.3 Tableau d'affichage

Le tableau (1) qui est dans cette interface est spécifié pour afficher les numéros du téléphone qui est reconnu par notre application.

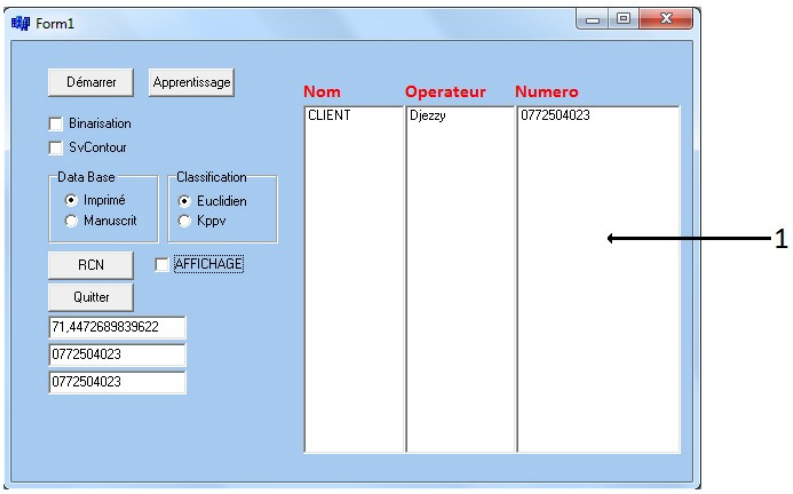


Figure 3.4 : Résultat de pointage un numéro via webcam

Le menu d'apprentissage ce fait pour charger la data base.

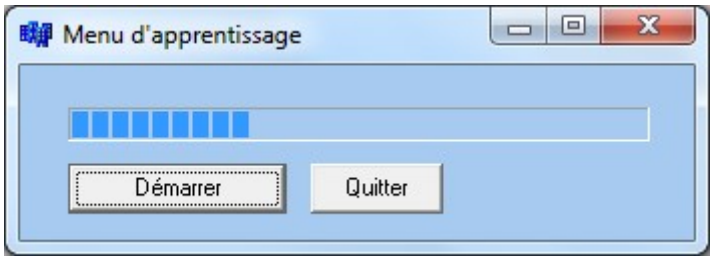


Figure 3.5 : Fenêtre de l'apprentissage

3.6 Reconnaissance des chiffres

Le travail que nous avons fait dans ce projet pour la reconnaissance des chiffres il est base sur les différentes étapes qui représentées par la figure 2.1 dans le chapitre 2.

Les résultats de ces étapes sont représentons comme suit :

3.6.1 Résultats de Filtrage :

Dans cette étape l'image de document acquise via Webcam est filtrée avec un filtre moyenneur, dans le but de réduire le bruit. Le résultat du filtrage de cette étape est représenté dans la figure suivante.

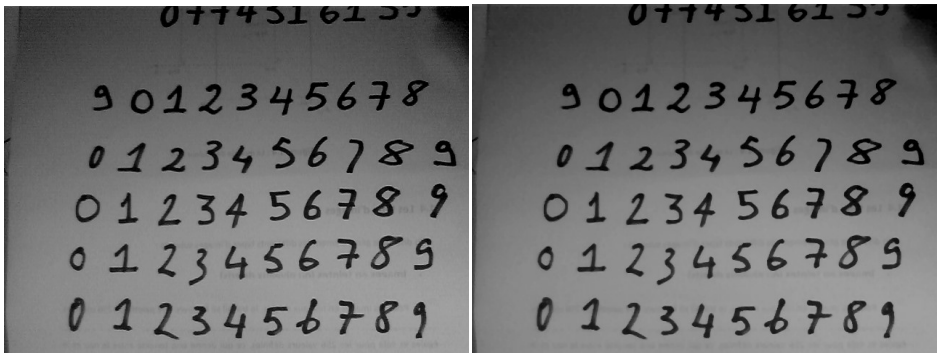


Image origine image filtré

Figure 3.6 : Imager filtré par un filtre moyenneur

3.6.2 Résultats de la binarisation :

Dans cette phase on a appliqué une binarisation globale basée sur la méthode d'Otsu. Le résultat de la binarisation est représenté dans la figure 3.7.

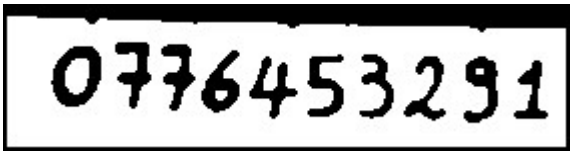


Figure 3.7: Résultat de la binarisation par méthode d'Otsu sur des chiffres manuscrits via webcam

3.6.3 Résultats de l'extraction de contour :

Dans cette partie on a appliqué l'algorithme de suivi de contour sur une image de chiffres manuscrits acquise par webcam. On ne garde que les contours externes du chiffre qui vont être utilisés ultérieurement comme l'indique la figure 3.8.

Figure 3.8 : Résultat de l'extraction de contour via webcam

3.6.4 Résultats de la reconnaissance des chiffres :

Dans cette étape qui représente la reconnaissance des chiffres ; toutes les étapes de la chaine de reconnaissance décrite précédemment ont été appliquées. Une base de données a été utilisée pour faire la classification. Les résultats de la classification sont représentés dans la figure 3.9.

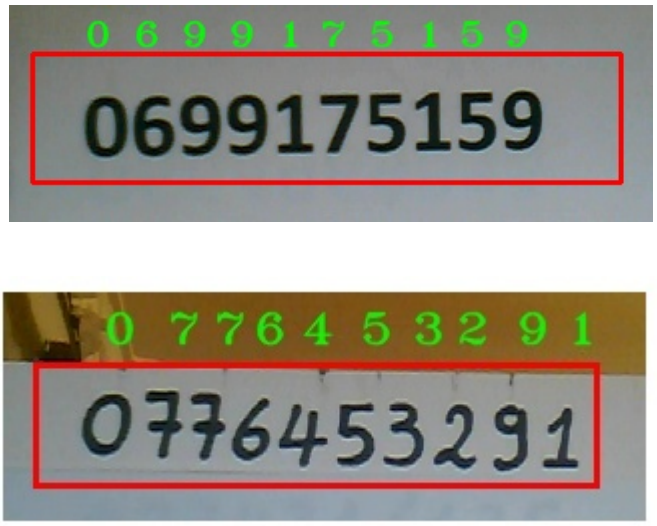


Figure 3.9 : Résultat de détection et reconnaissances des chiffres manuscrit et imprimé via webcam

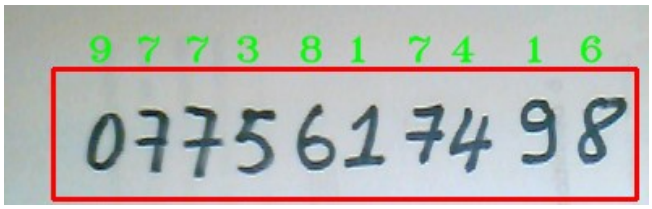


Figure 3.10 : Résultat de fausse détection et reconnaissances des chiffres manuscrit via webcam

3.7 Base d'apprentissage

La reconnaissance des chiffres fonctionne avec une base de données d'images enregistrées précédemment à partir d'une webcam. Cette partie le programme a réussi à créer une base de données, où les chiffres des sujets différents ont été identifiés, dans ce cas, en donnant le nom complet. On donne quelque échantillon des chiffres qui nous avons utilisé dans la data base Figure 3.11.

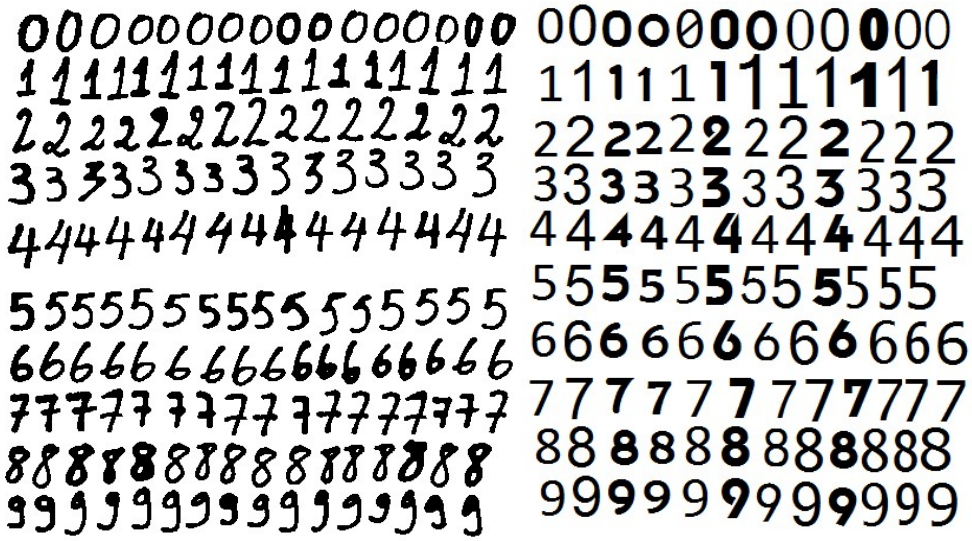


Figure 3.11 : Quelque échantillon des chiffres dans la data base

3.8 Segmentation des chiffres

La segmentation consiste à séparer les chiffres les uns des autres. La difficulté de cette étape réside dans le fait que les chiffres ne sont pas alignés et ne présentent pas la même taille, de plus, dans certains cas, les chiffres se chevauchent. Par conséquents, il faut utiliser des techniques de séparation permettant d'isoler les chiffres

La technique qu'on a utilisée est représentée comme suite à la figure 3.12

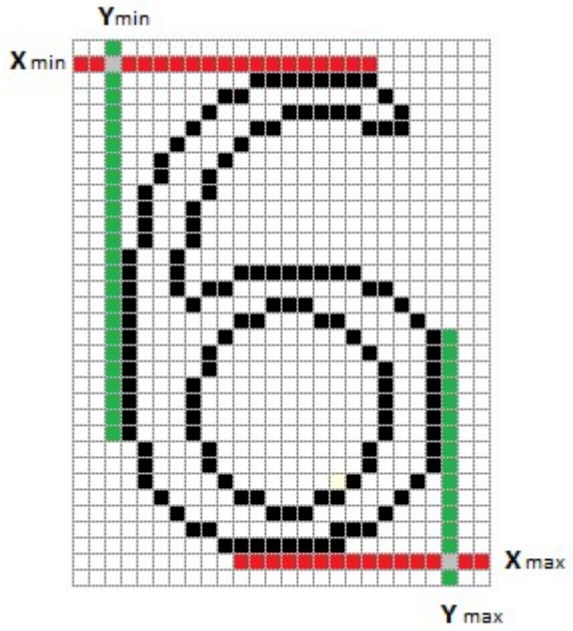


Figure 3.12 : Représentation des coordonnes min max

- On applique cette technique sur une image contour
- On cherche les coordonnés max et min du contour qui représente un seule chiffre

Figure 3.13 : Résultat de segmentation

3.9 Tests

L'application a été testée en temps réel sue des images provenant de la webcam. Il était important d'avoir les mêmes conditions d'éclairage et de distance par rapport au document écrit afin de mieux conclure les résultats obtenus.

3.9.1 Résultats des tests

3.9.1.1 Test avec des chiffres du data base

Dans cette partie nous avons fait un test avec des chiffres imprimés et manuscrits de la base de données. Nous avons obtenu un taux de reconnaissance presque égale à 100% en utilisant les deux classificateurs ; le k-ppv et la distance Euclidienne » comme le montre les figures 3.14 et figure 3.15.

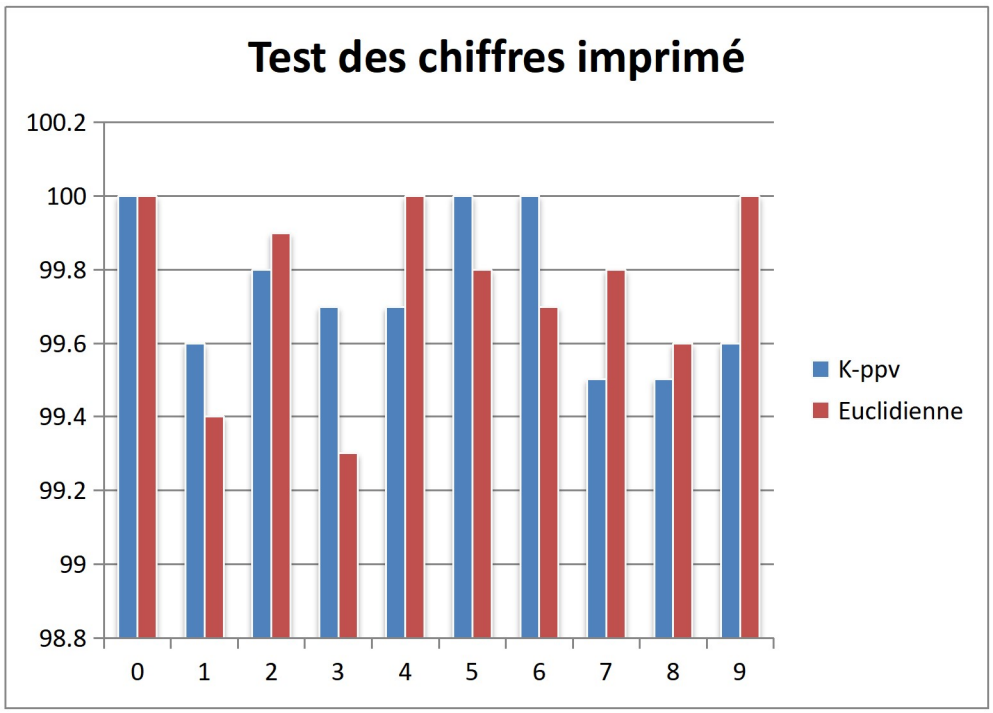


Figure 3.14 : Le taux de reconnaissance des chiffres imprimés du data base

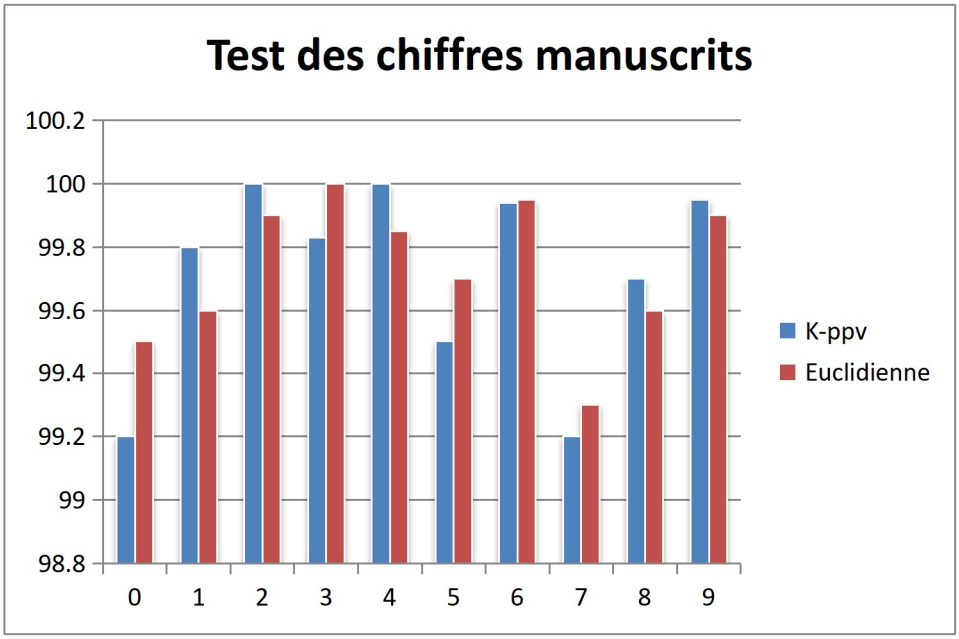


Figure 3.15 : Le taux de reconnaissance des chiffres manuscrits du data base

3.9.1.2 Test avec des chiffres d'un ensemble test

Les tests appliqués à des chiffres imprimés de l'ensemble test donnent des taux de reconnaissance presque égale à 100% avec les deux types de classificateurs (voir figure 3.16).

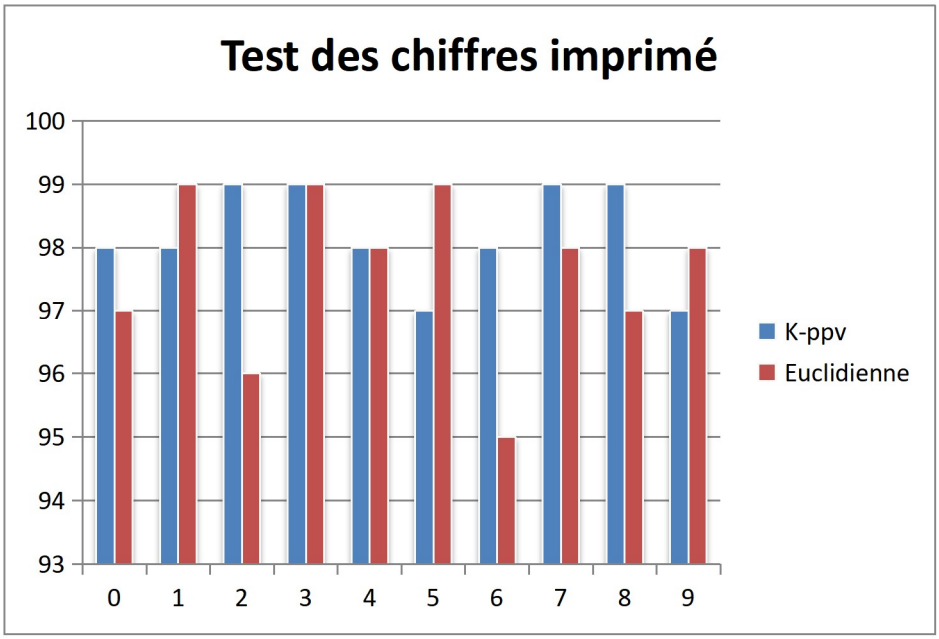


Figure 3. 16 : Le taux de reconnaissance des chiffres d'un ensemble test Imprimée

Pour les tests sur les chiffres manuscrits le même résultat a été obtenu pour les deux cas de classification, le pourcentage de reconnaissance varie par rapport à celui des chiffres imprimés. Dans le cas de k-ppv, le taux de reconnaissance des chiffre 0, 3 et 4 est de 90% et pour les chiffre 1,2,5,6,7 et 8 ça varie entre 40% et 60%. Le deuxième classificateur donne des taux de reconnaissances variant entre 60% et 80%, ce qui montre bien que le cas du classificateur euclidien est meilleur par rapport que le cas de k-ppv pour la reconnaissance des chiffres manuscrits.

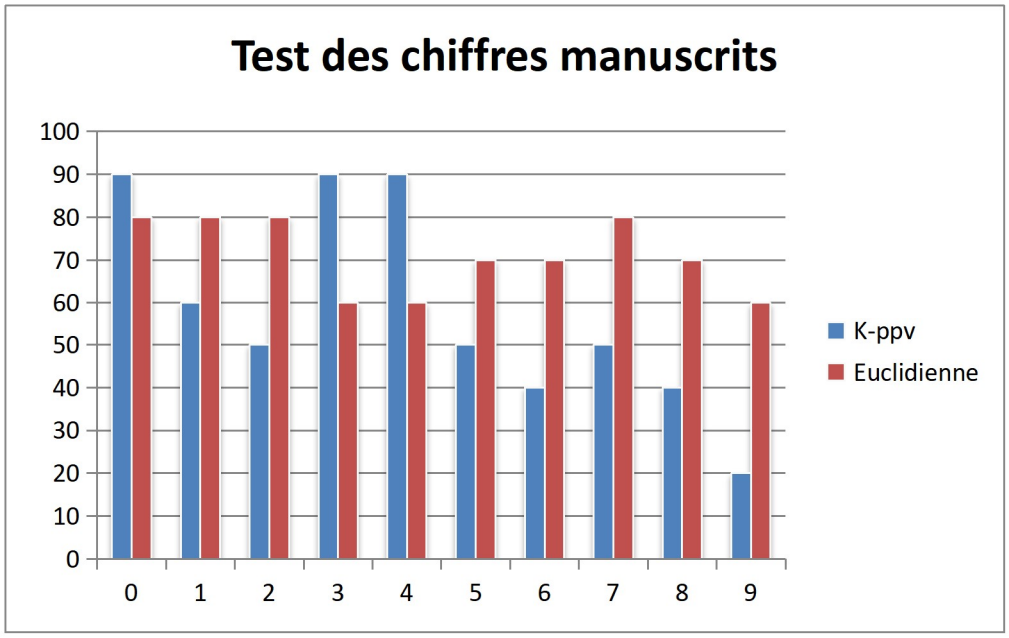


Figure 3.17 : Le taux de reconnaissance des chiffres d'un ensemble test manuscrit

On remarque que la méthode euclidienne il donne des résultats spécifie par rapport a la méthode k-ppv.

3.9.3 Taux de reconnaissance global par K_ppv et Euclidienne

Classificateur	Base données Imprimée	Base données manuscrit	Ensemble test Imprimée	Ensemble test manuscrit
k-ppv	99.94%	99.71%	98.2%	58%
Euclidien	99.75%	99.73%	97.6%	71%

Tableau 3.1 : Taux global de reconnaissance

3.10 Conclusion

Le passage de l'algorithme a l'application est la phase la plus importante et après une longue série de tests nous avons réussi a trouvé la combinaison de paramètres qui convient au système de reconnaissance développé. Les résultats présentés montre bien la l'applicabilité et la fiabilité du système en temps réel et la supériorité du classificateur euclidien par rapport à celui du k-ppv.

Conclusion générale

Au cour de notre travail, nous avons fait la réalisation d'une application pour la reconnaissance des chiffres imprimé et manuscrit via webcam au temps réelle.

En premier lieu, nous avons présenté quelque notion de base au traitement d'image.

En deuxième lieu, nous avons présenté les types de filtrage qu'il existe, ensuite les méthodes de binarisation. Ainsi les différentes méthodes utilisent pour l'extraction des paramètres de contenu de l'image. Ensuite nous avons présenté les types de classification. Après cela nous les avons évalué et choisis quelque unes pour notre projet pour l'implémenter.

En troisième lieu, on conçu des bases d'apprentissages, par prendre on considération les différents type d'écriture des chiffres pour chaque personne. A la fin, on charge la base sous le programme pour commencer le test avant fixé le nombre d'échantillons qui donnent de résultat acceptable.

A travers ce travail, nous avons pu acquérir beaucoup de la connaissance dans le domaine du traitement d'image.

Ce projet donne la chance pour utiliser et appliquer les outils appris au cours de notre formation, par toucher la cote d'application et découvrir le vaste domaine au cours de la programmation par langage C++ (builder C++ 6).

Annexes

- ## Ouvrir le flux d'une vidéo / caméra

Avant de pouvoir lire un quelconque **flux vidéo** il faut déjà l'ouvrir. Définissez premièrement la variable qui va stocker ce flux

```
CvCapture *capture;
```

a Ouvrir une vidéo

Ouvrir une vidéo est chose relativement aisée. Par contre il se peut que ve.us ne puissiez pas ouvrir tout type de vidéo. Ceci peut entre autre varier selon la version d'OpenCV mais aussi selon les différentes librairies installées sur votre système d'exploitation.

```
capture = cvCreateFileCapture("/patte/to/your/video/test.avi");
```

- ## Ouvrir le flux d'une Webcam / Caméra

Avec un flux vidéo récupéré en temps réel les capacités d'OpenCV prennent tout leur sens. Que ce soit une webcam ou une caméra, il est possible de récupérer ce qu'elles voient. De manière générale il est facile de lire le flux d'une webcam. Si vous souhaitez utiliser une caméra analogique (non numérique) il vous faudra probablement utiliser un convertisseur pour pouvoir la brancher en USB sur votre ordinateur. Notez par ailleurs que selon ce convertisseur (Dazzle, Easy Cap etc) vous aurez très probablement besoin du driver approprié selon votre système d'exploitation.

```
capture = cvCreateCameraCapture( CV CAP ANY );
```

- Le paramètre dans cvCreateCameraCapture() permet de définir quel flux vidéo récupérer. Généralement le CV_CAP_ANY permet de prendre le premier disponible (donc par exemple une webcam intégrée à votre ordinateur). Il se peut que cela ne marche pas tout le temps, donc si vous rencontrez un problème mettez 0 à la place ! Les valeurs diffèrent ensuite selon le port utilisé (usb, firewire). Pour plus d'information sur les valeurs disponibles, référez-vous au fichier highgui.h
- Toujours dans la même optique, vous pouvez y indiquer un autre chiffre pour ne pas récupérer le premier flux vidéo (par exemple /dev/video0 sur Linux) mais un autre. Ainsi si vous avez une Webcam intégrée et une caméra branchée en USB, et que vous souhaitez récupérer le flux de cette dernière, indiquez 1 à la méthode.

a Vérifier que le flux est bien ouvert

Il est possible qu'ouvrir le flux vidéo ne se passe pas comme prévu. Pour pallier à ce problème, il est fortement conseillé de réaliser une petite vérification sur la variable "capture".

```
if (!capture) {
    printf("Ouverture du flux vidéo
    impossible !\n");
    return 1;
    }
```

Pour améliorer la vérification vous pourriez même retenter l'ouverture du flux X fois jusqu'à décider d'arrêter (pour éviter une boucle infinie). En effet dans certains cas, la webcam/caméra n'est pas détectée immédiatement. Si l'on ne laisse pas un petit intervalle de temps à l'ouverture, alors il faudrait lancer le programme plusieurs fois de suite jusqu'à ce qu'il ne fasse plus l'erreur.

- ## Récupérer une image et l'afficher

Le principe du traitement vidéo est de procéder à une analyse du flux image par image**.** A chaque traitement d'image terminé vous affichez l'image résultante dans une fenêtre.

a Créer une fenêtre

Ce qui est bien avec OpenCV c'est que la création d'une fenêtre ne change pas en fonction du système d'exploitation. Par ailleurs vous pouvez **créer plusieurs fenêtres** pour afficher dedans plusieurs résultats (images ou vidéos). Une fenêtre est définie par un nom :

```
cvNamedWindow("GeckoGeek Window", CV WINDOW AUTOSIZE);
```

Le flag "CV_WINDOW AUTOSIZE" (qui est en fait un nombre : 0 ou 1) permet d'indiquer si la fenêtre doit s'adapter automatiquement ou non.

b Récupérer une image

Une image est stockée dans une classe spécifique à OpenCV : **IplImage.** Donc quelque part dans votre code avant de commencer vous devez la définir :

```
IplImage *image;
```

Il existe deux façons de récupérer une image du flux vidéo. La première (et la plus utilisée) est la suivante :

```
image = cvQueryFrame(capture);
```

Il faut indiquer en paramètre le flux vidéo à utiliser pour récupérer l'image. Cela devient vital dans le cas où vous utilisez plusieurs sources de vidéo !

L'autre possibilité pour lire l'image est d'utiliser deux méthodes séparées. La première permet de demander à OpenCV de récupérer une image du flux vidéo et de la stocker en mémoire. La méthode retourne 0 ou 1 en fonction du résultat.

```
cvGrabFrame(capture);
```

Ensuite il suffit de récupérer l'image dans la variable appropriée :

```
image = cvRetrieveFrame(capture);
```

Notez toutefois que c'est cette dernière méthode qui prend le plus de temps (5 à 10 fois plus que cvGrabFrame).

c Afficher une image

Une fois l'image stockée dans une variable il suffit de l'afficher dans une fenêtre préalablement créée.

```
cvShowImage("GeckoGeek Window", image);
```

Notez qu'en général, avant d'afficher l'image, on effectue des traitements sur cette dernière : zoom, contraste, reconnaissances variées (faciale par exemple), etc. Prenez en compte que la communauté d'OpenCV offre de nombreux codes en exemple qu'il est conseillé de parcourir pour vous faire une idée des possibilités.

• Boucler et temporiser

Pour récupérer les images petit à petit il vous faudra **créer une boucle** et faire une petite pause entre chaque image.

a Faire une pause

Faire une pause dans OpenCV permet en même temps de récupérer une interaction clavier. Cela permet ainsi d'analyser la touche qui a subie une interaction et de procéder à certaines actions en conséquence. Notez que ce système peut malheureusement entrer en conflit selon votre programme. Par exemple si vous utilisez en plus une librairie graphique (comme QT) avec des champs texte, il vous faudra trouver une solution pour partager ces événements clavier.

```
key = cvWaitKey(10);
```

La commande cvWaitKey() attente X ms et renvoie l'interaction clavier (s'il y en a eu une).

Le temps d'attente varie en fonction de framerate que vous souhaitez imposer. Par contre vous ne pourrez pas aller au-delà de ce que propose votre caméra. Notez par ailleurs que mettre un temps d'attente trop court peut causer un freeze sur certaines versions de Linux. Mettez donc en général au moins 5 ms.

b Faire une boucle

Et enfin il vous faut assembler tous les aspects dont on a parlé ci-dessus pour lire et afficher la vidéo ! Ici nous nous basons sur l'interaction clavier pour arrêter la boucle while. Il existe d'autres moyens d'arrêter la boucle (comme se baser sur le retour de cvGrabFràme).

```
// Boucle tant que l'utilisateur n'appuie pas sur la touche q (ou Q)
while(key !_ 'q' && key'Q') {
      // On récupère une image
      image = cvQueryFrame(capture);
     // On affiche l'image dans une fenêtre
     cvShowImage("GeckoGeek Window", image);
     //Onattend 10ms
     key = cvWaitKey(10);
}
```

• Libérer la mémoire utilisée

Etape très (très) importante, libérer toute la mémoire que vous avez utilisé en créant ces divers objets.

a Libérer la fenêtre

Une seule fonction suffit à fermer toutes les fenêtres :

cvDestroyAllWindows0 ;

Toutefois dans le cas où vous ne souhaiteriez que fermer une seule et unique fenêtre, vous pouvez utilisez cette dernière :

```
cvDestroyWindow("GeckoGeek Window");
```

b Libérer la variable d'image

Lorsque vous récupérez une image d'un flux vidéo vous n'avez pas besoin de la "libérer" dans la boucle while, ni même après. OpenCV gère ceci automatiquement et vous n'avez qu'à votre charge de dés allouer une image créée par vos soins. Si vous veniez à le faire, pensez donc à faire ceci :

```
cvReleaseImage(&uneAutreImage);
```

c Libérer la capture vidéo

Et enfin, il vous faut libérer la capture vidéo (entre autre pour que la webcam/caméra ne soit plus utilisée par le programme).

```
cvReleaseCapture(&capture);
```