Université de Versailles Saint Quentin en Yvelines - France

University of Balamand Liban

École Doctorale Sciences et Technologies de Versailles
and Faculty of Engineering-Balamand

THÉSE DE DOCTORAT

présentée par :
Rabih Al Nachar

Spécialité : Traitement des images ; Vision par ordinateur

TOWARDS AN EFFICIENT FEATURE DETECTOR: EDGE CORNERS AND ITS APPLICATIONS

Soutenue le 06/11/2014, devant le jury constitué de :

Président:	Patrick Garda	Professeur, Université Pierre et Marie Curie.
Directeurs:	Patrick Bonnin	Professeur, Université de Versailles
	Elie Inaty	PhD, University of Balamand
Rapporteurs:	Sylvie Lelandais	Professeur, Université d'Evry Val d'Essonne
	Edwidge Pissaloux	Professeur, Université Pierre et Marie Curie.
Examinateur:	Yasser Alayli	Professeur, Université de Versailles

Table of Contents

List of Figures

List of Tables

Abstract

In this thesis, a new feature detector is proposed. The new features are edge corners located on the contours of a studied image. These points are edge points where a deviation in the edge direction occurs. In addition, they are repeatable versus similarity, affine transformations and also robust to noise at the boundaries of the object's image. Due to their repeatability, these corners are used in a shape recognition application. Also, a smaller set of corners called "Dominant Corners" or "DCs" is extracted form the original set of corners using a new proposed polygonal approximation algorithm. These DCs form the vertices of a polygon that best approximate their contour. Two applications using the edge corners are also developed. The first one is an image registration application that forms invariant primitives using the DCs. The second application is a word recognition application where the edge corners located on the characters contours are used in a simultaneous segmentation/recognition process to recognize the characters in a deformed word image.

Keywords: Edge corners, DCs, polygonal approximation, image registration, word recognition, affine transformation, invariant primitives, simultaneous segmentation/recognition.

Résumé

Nous proposons dans cette thèse un nouveau détecteur de « Coins » de contour dans une image. Ces coins sont les sommets de la ligne polygonale approximant le contour. Ils peuvent appartenir ou non au contour. Ils correspondent à une déviation importante de la direction de ce contour. Aussi, ils sont répétables en présence de transformations affines ou similitudes et sont robustes au bruit présent aux frontières d'une image. Grâce à cette répétabilité, les coins sont utilisés dans une application de reconnaissance de la forme.

Les coins peuvent être classés selon leur force. Ainsi sous ensemble de ces coins, appelé "Coins Dominants", peuvent être extraits formant les sommets du polygone « minimal » qui représente le contour, pour un nombre de segments donné.

Deux applications, basées sur les Coins/Coins Dominants du contour ont été réalisées :

- La première est une application de recalage d'images où de nouvelles primitives invariantes constituées de quatre "Coins Dominants" du contour ont été proposées.
- La seconde application est la reconnaissance des caractères dans une image déformée où les coins du contour des caractères ont été utilisés dans un processus de segmentation / reconnaissance simultané.

Mots-clés: Point d'Intérêts, Coins et Coins Dominants du contour, approximation polygonale, recalage d' images, reconnaissance des caractères, transformation affine, primitives invariantes, segmentation et reconnaissance simultané.

Acknowledgments

First of all, I would like to express my deep sense of respect and gratitude towards my PhD directors: Prof. Patrick Bonnin and Dr. Elie Inaty who have guiding me throughout this work. I am especially grateful to their remarks and supports. I have learned from them a lot: how to evaluate a work, how to express my ideas, how to drive them among others ideas … I would also to express my deep gratitude towards my PhD reviewers: Prof. Sylvie Lelandais and Prof. Edwige Pissaloux who have made a deep review of this thesis and due to their valuable comments I was able to improve the quality of the work.

Next, I would like to thank my family: parents and wife for their support, love and sacrifice. Without them, I could not achieve this work.

1

Résumé Détaillé de la Thèse

Nous proposons dans cette thèse un nouveau détecteur de « Coins » de contour dans une image. Ces « Coins » correspondent à une déviation importante de la direction de ce contour. Ils sont obtenus à partir d'un détecteur d'une autre primitive basée contour : les « Segments » de contour. Ils constituent les sommets de la ligne polygonale approximant le contour. Les Segments, et par conséquent les Coins de Contours sont des primitives images très importantes, car elles sont nombreuses dans les images de scènes d'environnements faits par l'homme : d'intérieur ou d'extérieur urbain [13-16].

Bien qu'un « Coin » puisse très bien être considéré comme un « Point d'Intérêt » en tant que point particulier du contour, il faut différencier ces deux primitives. En effet, les points d'intérêt sont obtenus localement, sans l'extraction des contours au préalable. Ils ne font pas nécessairement partie d'un contour, ou n'en sont pas nécessairement à proximité. Les Coins de Contour appartiennent aux contours, où en sont très proches (dans le cas de la fusion de deux Demi-Coins).

Les coins peuvent être classés selon leur force. Ainsi un sous ensemble de ces coins, appelé "Coins Dominants", peut être extrait formant les sommets du polygone « minimal » qui représente le contour, pour un nombre minimal de segments donné.

Les Coins, comme les Coins Dominants sont répétables en présence de transformations affines ou similitudes et sont robustes au bruit présent sur les frontières des objets de l'image. Grâce à cette répétabilité, les Coins sont utilisés dans une application de reconnaissance de la forme.

Deux applications, basées sur les Coins Dominants du contour ont été réalisées :

La première est une application de recalage d'images où une nouvelle primitive invariante constituée de quatre "Coins Dominants" successifs du contour a été proposée.

La seconde application est la reconnaissance de caractères dans une image déformée où les coins du contour des caractères ont été utilisés dans un processus de segmentation/reconnaissance simultané.

Le Recalage d'Image détermine la modélisation de la transformation géométrique qui permet d'aligner les mêmes points de deux images d'une même scène prise à

différents points de vue, sous différents angles, éventuellement à différents instants et avec des caméras de différentes caractéristiques.

Définition: La « Répétabilité » d'une grandeur physique, i.e. les Coins Dominants extraits d'une image, est la mesure de leur stabilité lorsque l'image subit des transformations.

En fait, la répétabilité des Coins Dominants a été étudiée en présence de déformations affines de l'image et les résultats obtenus sont très bons. Ainsi, les Coins Dominants peuvent être utilisés pour une application de Recalage d'Images où le temps entre les deux prises de vue est relativement court, telle que la période entre les deux images d'une séquence vidéo. Sous cette contrainte, la déformation réelle entre deux images est petite, et peut correctement être modélisée par une transformation affine [44]. Ainsi, nous suggérons d'utiliser cette technique dans une application de robotique mobile et autonome : la surveillance de routes ou d'autoroutes à l'aide de drones aériens. En effet, le recalage permet de Compenser le mouvement global du fond lié au déplacement de la caméra montée sur le drone, ce qui permet la détection du mouvement différentiel de petites cibles, telles que les automobiles sur les routes.

Les quatre principales étapes du Recalage d'Images sont :

- la détection de Points de Contrôle (PC) : Ces points doivent être stables ou répétables malgré les transformations de l'image,
- la Mise en Correspondance de Points de Contrôle : les PC des différentes images sont associés en utilisant des grandeurs invariantes,
- l'Estimation de la Transformation : à partir de l'association précédente des points de contrôle,

Ré échantillonnage de l'Image : en appliquant la transformation de l'étape précédente, la première image est synthétisée avec le point de vue de la seconde..

Dans ce travail, les Coins Dominants sont utilisés comme Point de Contrôle, car ils sont stables par transformation, et un nouvel algorithme, très efficace, de mise en correspondance de ces points a été proposé.

La seconde application concerne la Reconnaissance de Caractères. Nous avons cherché à reconnaître des caractères déformés et connectés. Aussi, nous avons proposé une méthode permettant simultanément la segmentation et la reconnaissance de chaque caractère dans un mot d'une image. Cette méthode a été testée sur des mots dont les lettres ont été déformées, comme par le système CAPTCHA, pour la sécurité sur internet.

Notre rôle n'est pas de jouer un « hacker » essayant de pirater l'accès d'une application internet. Au contraire, l'utilisation du détail de notre application permettra d'améliorer la sécurité des applications existantes. Notre principal objectif est la conception et la réalisation d'une application de robotique, notamment d'interaction homme – robot. Le robot doit reconnaître les commandes ou les informations écrites à la main produites par un opérateur humain : les caractères manuscrits étant déformés et potentiellement connectés.

1.1. Détecteur de Coins basé Contours

Les principales étapes sont les suivantes :

(i) Détection des Contours: calcul du Gradient par la méthode de Kirsch [17, 18], seuillage sur la norme, affinage, prolongation et chaînage. Le chaînage a été modifié pour détecter « proprement » les coins du contour. La sortie du détecteur de contour est une image des étiquettes contours, associée à une base de données. Les contours forment les bords des objets de la scène, où se situent les coins que nous recherchons.

(ii) Détection des Segments de Contours [28, 198]: les segments de contour sont des parties du contour en forme de segments de droite. Ils sont également des Primitives Image. Le but d'utiliser un détecteur de segments de contour est de diviser le contour en une séquence de segments de droites de différentes longueurs. Les segments de droite sont obtenus à partir des codes de Freeman des points de contour. La robustesse du détecteur de segment de droite est son aptitude à détecter proprement un segment même si le contour est corrompu par des pixels bruités. Par définition, un pixel bruité est un point de contour dont la direction est différente de la direction principale du segment. Ce détecteur doit détecter de manière adéquate ces points de bruit et les éliminer,

afin que le détecteur de « Coins » de l'étape suivante ne les confonde pas avec de vrais Coins. En effet, Points Bruités et Vrais Coins correspondent à des déviations du contour.

(iii) Détection de Coins [28, 198]: un coin est définit comme le point d'intersection de deux lignes droites non co-linéaires de longueur appropriée (lors des expérimentations du chapitre 5, paragraphe 5.1, le seuillage sur la longueur minimale des segments est pris égal à 10).

(iv) Détection de Coins Dominants [53]: parmi l'ensemble des coins de contour, un sous ensemble appelés Coins Dominants CDs est sélectionné en éliminant itérativement les Coins « moins marqués ». Ces CDs ont une grande répétabilité sous différentes transformations d'images.

1.1.1. Détecteur de Contour avec Adaptation

Le Détecteur de Contour est constitué de deux regroupements, fonctionnant séquentiellement, des principales étapes d'une Détection classique de Contours. Ce nombre minimal de deux regroupements des étapes, mais également des calculs locaux en chaque pixel de l'image garantit la rapidité de la détection [17].

Le Détecteur de Contour programmé pour le projet CLEOPATRE [17] a pour objectif d'extraire les longues lignes droites de l'image. Or nous cherchons à l'utiliser pour détecter des Coins de Contour, problématique différente, d'où la nécessité de l'adapter.

1.1.1.1. Premier Regroupement

Ce premier regroupement commence par le calcul du vecteur gradient, en norme et en argument, en chaque pixel de l'image par l'opérateur de Kirsch [18]. Les étapes de Seuillage sur la Norme du gradient et d'Affinage des Contours regroupées permettent d'obtenir une image binaire des (points de) contours affinés, c'est-à-dire d'un pixel d'épaisseur.

Un problème survient au niveau de la phase du gradient. C'est le problème d'Arrondi des Angles, présenté Figure 1.1 sur un angle droit, et Figure 1.2 sur un angle aigu. Les deux figures présentent l'image originale et de la norme du gradient. Comme le « Coin de Contour » est l'intersection de deux « Segments de Contour » non colinéaires, d'une longueur minimale, il risque de ne pas être correctement détecté à cause de ce problème.

La solution que nous mettons en œuvre après la Détection de Segments de Contour est d'introduire les « Demi – Coins » (DCs). Un Demi Coin est une intersection de deux Segments de Contour non colinéaires, dont l'un d'entre eux n'a pas une longueur suffisante, comme les deux points C1 et C2 de la Figure 1.3 (a). Ainsi, si deux « Demi Coins » sont proches (distance inférieure à 3 pixels), le « Coin » réel est obtenu par intersection des deux segments de contour de longueur appropriée, si celle-ci est suffisamment proche des deux demi coins. La Figure 1.3 (b) présente les deux demi coins C1 et C2, ainsi que le coin réel C correspondant. Le Coin C n'appartient pas au contour dans ce cas, mais en est proche.

Fig.1.1. Norme du Gradient en un angle droit avec l'opérateur de Kirsh.

Fig.1.2. Norme du Gradient en un angle aigu avec l'opérateur de Kirsh .

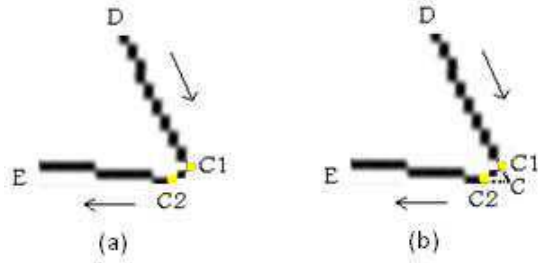


Fig.1.3. (a) C1 et C2 sont deux DCs. (b) C est un Coin "réel".

1.1.1.2. Second Regroupement

Le second regroupement est constitué des étapes de prolongation et de chaînage des contours, mais séquencées de manière inhabituelle [17]. En effet, la prolongation intervient comme procédure de l'étape de chaînage si le pixel courant est une extrémité de contour. Prolongation et Chaînage fonctionnent en parallèle : un pixel est prolongé puis chaîné, ceci pixel par pixel.

Les étapes de Prolongation / Chaînage ont été conçues pour extraire les longues lignes droites [17]. Ainsi à une intersection le chaînage du pixel en ligne « droite » est privilégié, et l'intersection n'est pas « relevée ». Notre problématique est tout autre : la détection de Coin de contours. Elle nécessite donc une adaptation.

En effet, notre problème est le suivant. Considérons l'image de la tasse Figure 1.4 (a), et l'image des Points de Contour correspondant Figure 1.4 (b), où les Points A et B sont représentatifs du problème. La procédure initiale suit la « ligne droite » (cf Figure n° 1.4 (c)). L'adaptation permet de considérer les différentes possibilités de chaînage (cf Figure n° 1.4 (d)), qui sont maintenant considérés comme des points doubles. De cette manière, les points A et B pourront être détectés comme des Coins de Contour (cf Figure 1.5).

Fig.1.4. (a) Image d'une Tasse. (b) Image des Contours. (c) Ancienne procédure aux points A et B. (d) Nouvelle Procédure : Introduction d'un double point.

Figure 1.5. Nouvelle Procédure lors d'une Fourche.

1.1.2. Détecteur de Segments de Droite

On doit être capable de distinguer un Segment de Contour Idéal (SCI) d'un Segment de Contour Réel (SCR). Un Segment de Contour Idéal comporte des points de contour dont les codes de Freeman (permettant de passer du point courant au point suivant) comportent une ou deux directions au maximum. Par exemple la Figure 1.6 présente 5 segments de contours idéaux SCI1 à SCI5. SCI1 et SCI5 sont constitués de points de contour comportant une seule direction (ou code de Freeman) : respectivement 0 et 1.Les trois autres segments de contours idéaux : SCI2 à SCI4 sont constitués de points de contours de deux directions principales ; ils diffèrent par leur fréquence d'occurrence de ces deux directions. Pour SCI2, la fréquence d'occurrence de code « 0 » est double de celle de code « 1 ». Ainsi le code « 0 » est la direction principale « pdir » et le code « 1 » la direction secondaire « sdir ».

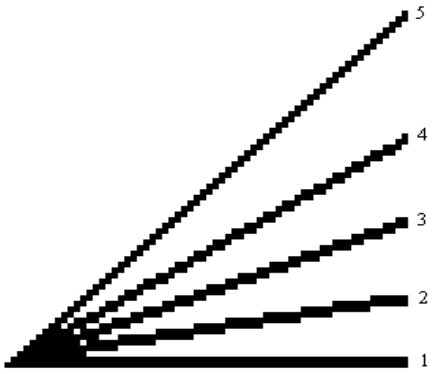


Fig.1.6. 5 Segments de Contour Idéaux SCI1 à SCI5.

Dans des situations réelles, les Segments de Contour Idéaux n'existent que très peu, à cause du bruit sur les contours. Ainsi, des pixels de bruit figurent dans les contours, comme le montrent les points cerclés de la Figure 1.7 (a).Ainsi, un algorithme intelligent doit être capable de détecter ces pixels de bruit, et de les éliminer de manière à trouver la bonne direction du segment de contour.

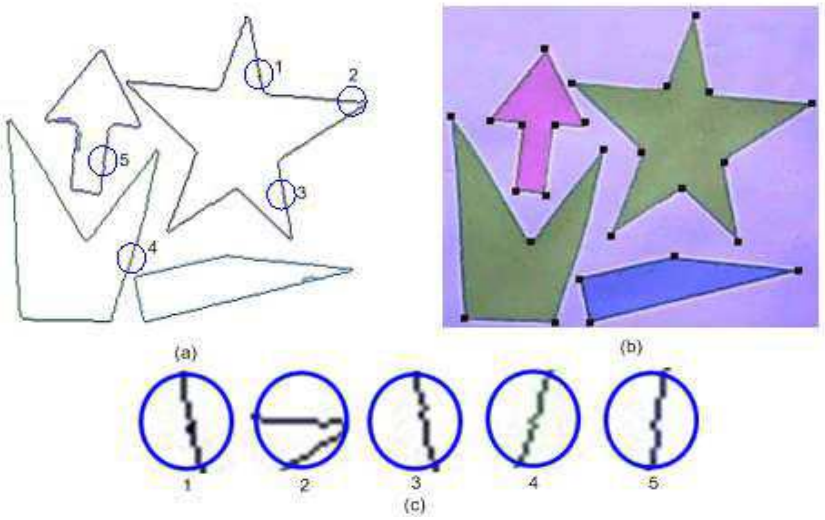


Fig.1.7. (a) Image Réelle de Contour. (b) Image Réelle et Coins Détectés. (c) Cinq cas où des pixels de bruits apparaissent en (a).

1.1.3. Détecteur de Coins de Contour

Les Coins de Contour sont définis comme étant les intersections de deux segments de contours consécutifs non co-linéaires, avec une longueur minimale. Dans nos expérimentations, la taille minimale des segments de contours a été fixée à 10. La Figure 1.7 (b) présente les coins détectés sur une image réelle.

Un Coin de Contour est caractérisé par deux paramètres. Le premier est son angle entre les deux segments de contours. Le second est le rapport des longueurs de ces deux segments. Au chapitre 5, nous avons étudié la répétabilité de ces deux paramètres face à la variation d'échelle. Il est apparu qu'ils sont quasiment invariants. Ainsi, nous avons suggéré une application robotique : une application de Reconnaissance de Forme 2D utilisant les Coins de Contour. L'idée est d'embarquer notre algorithme sur un robot mobile et autonome. Dans la phase d'apprentissage, un opérateur humain présente au robot la forme 2D à rechercher. L'algorithme extrait les Coins de Contour qui sont caractérisés par l'angle et le rapport des longueurs des Segments de Contours le constituant. Dans la phase opérationnelle, le robot recherche la forme 2D apprise. L'algorithme détecte les Coins de Contour de chaque contour, et les met en correspondance en utilisant leurs angles et rapports de longueur des segments.

1.1.4. Détails de l'algorithme de détection des Segments et des Coins de Contour

Les Détections des Segments puis des Coins de Contour sont réalisées sur les contours extraits. 6 paramètres caractérisent chaque Segment de Contour :

- La direction primaire *pdir* et sa fréquence *pcount*.
- La direction secondaire *sdir* et sa fréquence *scount*.
- Autres directions *odir* et leur frequence *ocount*.

Pour illustrer l'algorithme, prenons un contour composé de deux segments de contour non colinéaires, respectivement en noir et en vert sur la Figure 1.8.

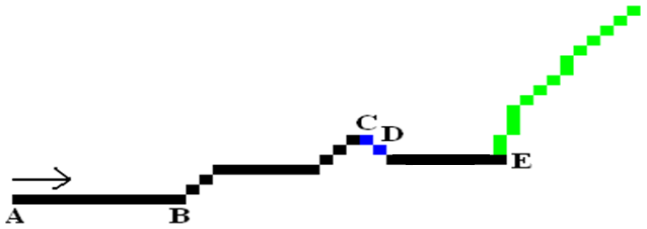


Fig.1.8. Détection de Segments de Contour.

L'algorithme démarre du point A. La direction courante (ou code de Freeman) est « 0 » : il est enregistré comme *pdir* et sa fréquence *pcount* est incrémenté de A à B. En B, le contour est dévié et la direction courante vaut « 1 », enregistré en tant que *sdir*. Sa fréquence *scount* et aussi *pcount* sont incrémentés jusqu'en C. En C et D, la direction courante est « 7 » est enregistré en *odir* et sa fréquence *ocount* est égal à deux au point D. Entre D et E la direction courante est « 0 », soit *pdir* ainsi *pcount* est incrémenté et *ocount* est « effacé ». Cependant, à partir de E sur la portion en vert la direction courante n'est pas *pdir*, ainsi *ocount* ne sera pas effacé et va dépasser le seuil prédéfini, égal à deux dans nos expérimentations. L'algorithme prend le point E comme fin du premier segment de contour et début du second. Ensuite, E sera détecté comme Coin de Contour.

Les variables utilisées dans l'algorithme sont:

- *cdir*: direction du point de contour courant.
- *pdir*: direction du point de contour précédent.
- adir: direction d'avance.

- *fdir*: direction finale du segment de contour.

- *mdir*: direction principale du segment de contour.

- *sdir*: direction secondaire du segment de contour.

- *Acc*1: accumulateur de la direction principale.

- *Acc*2: accumulateur de la direction secondaire.

- *Accn*: accumulateur d'autre direction.

- *L*: longueur courante du segment de contour.

- *hn*: seuil maximal sur le nombre des points bruits permis.

- *hl*: seuil minimal sur la longueur du segment de contour.

- *EndofStraightEdge*: variable logique pour déclarer la fin du segment de droite.

L'algorithme de détection du segment de contour et de sa direction finale est présenté dans Figure 1.9.

1.1.1.5. Suppression itérative de Coins de Contour

La suppression peut être initialisée à partir d'un nombre quelconque de coins sur un contour donné. Le but est d'éliminer itérativement les coins. Le coin éliminé à chaque itération correspond à l'erreur minimale d'approximation. L'élimination itérative s'arrête lorsqu'un critère d'arrêt est atteint. Le critère d'arrêt est le taux de compression CR.

$$CR = {}^{n}/_{nc} \tag{1.1}$$

où n est le nombre de points de contour, et nc le nombre de Coins de Contour à trouver.

L'objectif est de minimiser la fonction « Global Integral Square Error » ou (GISE). L'ensemble de Coins de Contour restants formera l'ensemble de Coins Dominants (Dominant Corner DC).

L'algorithme utilise deux grandeurs : la " Local Integral Square Error" (*LISE*) d'un segment et la "Local Integral Square Error Variation" (*LISEV*) due à la suppression d'un Coin de Contour. Pour illustrer ces notions, considérons le contour en noir, approximé par la ligne polygonale Cor1, Cor2, Cor3 et Cor4 de la Figure 1.10.

- *Considère le point courant comme la tête du contour étudié.*
- *Initialisation: cdir =pdir = -1.*
- *Tant que le point courant n'est pas la queue du contour*
  - *Si (cdir!=pdir)*
    - *Itère deux fois*
      - *Mettre à zéro L, Acc1, Acc2 and Accn.*
      - *Commencer par le point courant.*
      - *Si on est dans la première itération alors adir=cdir: continuer selon le premier parcours du point courant.*
      - *Si on est dans la seconde alors adir=(pdir+4)%8:Continuer selon le deuxième parcours.*
      - *Point d'avance est le point suivant du point courant selon adir.*
      - *Initialisation: mdir=adir and L=1*
      - *adir = direction courant du point d'avance.*
      - *EndofStraightEdge = faux.*
      - *Tant que (!EndofStraightEdge)*
        - *Si (adir == mdir)*
          - *Incrémenter Acc1 et mettre à zéro Accn*
        - *Si non si adir!=mdir pour la première fois*
          - *sdir = adir*
        - *Si non si (adir == sdir)*
          - *Incrémenter Acc2*
        - *Si non*
          - *Incrémenter Accn*
        - *Si(Accn > hn)*
          - *Si(L > hl)*
            - *Le contour courant est considéré comme "Segment Contour".*
            - *fdir = (mdir*Acc1+sdir*Acc2)/(Acc1+Acc2)*
            - *EndofStraightEdge = Vrai.*
        - *Incrémenter L*

Fig.1.9. Algorithme de détection des segments de contour.

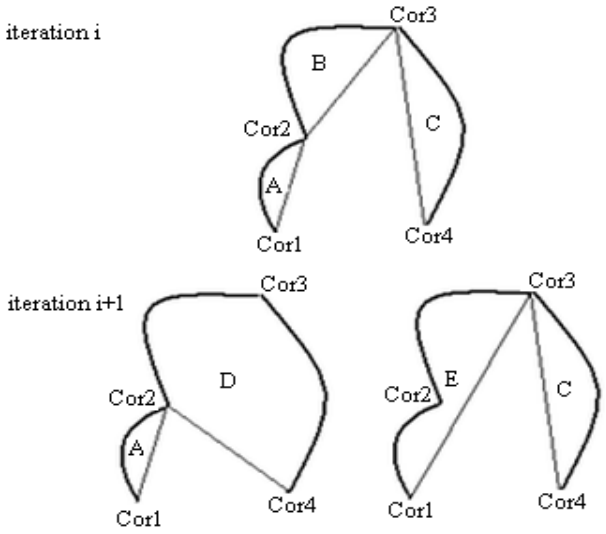


Fig.1.10. Elimination d'un Coin de Contour.

A l'itération i, la portion de contour comprise entre Cor1et Cor2 est approximée par le segment [Cor1Cor2]. L'erreur locale correspondante *LISE* est la somme des carrés des distances entre le contour approximé, et le segment de contour l'approximant, ce qui est égal à l'Aire A. L'erreur globale du contour complet compris entre Cor1 et Cor4 *GISE* à cette itération est la somme de toutes les LISE, soit : A+B+C.

A l'itération i+1, l'algorithme a le choix de supprimer les Coins Cor2 ou Cor3. Si le Coin Cor3 est supprimé la variation d'aire associée $LISEV_3$ est égale à la nouvelle D moins l'ancienne (B+C) (cf équation Eq (1.2)). La nouvelle aire globale *GISE* soit ($GISE_{i+1}$) est égale à l'ancienne ($GISE_i$) plus la nouvelle variation ($LISEV_3$). Il faut effectuer un calcul similaire pour supprimer le Coin Cor2 (cf Eq (1.3)). Finalement, l'algorithme élimine le Coin (Cor2 ou Cor3) qui correspond à l'aire minimale *GISE*.

$$LISEV_3 = D - (B + C) \tag{1.2}$$

$$LISEV_2 = E - (A + B) \tag{1.3}$$

1.1.6. Contribution

La principale faiblesse des détecteurs existants de Points d'Intérêt ou de Coins est la qualité de leur détection avec des images réelles où le bruit notamment sur les frontières des objets est relativement important. De plus les détecteurs de points d'intérêt sont basés sur l'intensité des pixels de l'image. Par conséquent leur détection est dépendante d'un ou de plusieurs seuils sur l'intensité, généralement constant, ce qui la rend très sensible aux bruits.

Nous avons présentés deux nouveaux détecteurs de primitive image : les Coins de contour, et les Coins Dominants de contour. Les Coins de contour sont « très répétables » ce qui les rend applicables dans de nombreuses applications comme l'approximation polygonale, le recalage d'images, la reconnaissance de caractères etc... De plus, comme les coins sont regroupés grâce à leur appartenance à un contour, nous n'avons pas besoin d'utiliser une méthode de mise en correspondance coûteuse en temps de calculs comme la méthode RANSAC pour mettre en correspondance les coins provenant de deux images, ou d'un modèle et d'une image.

Comme les contours sont l'une des primitives les plus importantes car répétable lors de différentes transformations d'images [3], nous les avons utilisé pour détecter nos coins qui leur appartiennent. Ainsi, notre but est de détecter des primitives images de type « Point » qui sont à la fois correctement localisées et répétables malgré des déformations de l'image. Par conséquent, ils peuvent être utilisés comme Points d'Intérêt sur le contour d'un objet, où le descripteur est calculé localement à chaque point, en fonction du point et de son environnement.

Les « Coins Dominants » sont des primitives image de type Point « très répétables » sous des transformations affines, et sont utilisés comme sommets de la ligne polygonale qui approxime au mieux le contour. Les différences entre les travaux existants et nos travaux sont la nature et la stabilité des points sélectionnés.

Les points sélectionnés par Masood [127] sont les points qui correspondent à une déviation dans la direction du contour. Les points que nous sélectionnons sont des coins de contour qui sont des intersections de deux segments de droite approximant le contour. Ainsi nous ne détectons pas un coin à chaque changement de direction du contour correspondant à un bruit sur le contour. En plus, Masood élimine itérativement les coins en utilisant une mesure de l'erreur appelée « Associated Error Value » ou (AEV). L'AEV à un point dominant est le carré de la distance de ce point à la ligne joignant les points dominants précédent et suivant.

L'erreur que nous utilisons la « Global Integral Square Error » ou (GISE) est proportionnelle à la surface entre la portion de contour et le segment l'approximant. Elle est similaire au critère utilisé par Wall and Danielson [118], alors que l'AEV est une distance maximale, comme utilisé par Pavlidis [117].

1.2. Première Application : le Recalage d'Images

Cette application est basée sur les Coins Dominants extraits par l'algorithme de Suppression Itérative de Coins présenté paragraphe 1.1.

Considérons deux images appelées images courante et de référence prise d'une même scène, mais à deux différents instants, avec un intervalle temporel relativement faible. La première étape consiste en la sélection des Coins Dominants pour chaque contour dans chaque image. Puis, une primitive est formée pour chaque quadruplet de

Coins Dominants consécutifs du même contour. Ainsi, une primitive est un ensemble de points d'intérêt ou de Coins Dominants formant une quantité invariante par transformation d'images. Ensuite, les primitives sont associées entre images, en utilisant cet invariant. Finalement, chaque association de primitives vote pour un modèle de transformation affine. Le modèle ayant obtenu le plus grand nombre de votes est celui retenu pour le recalage.

Notre méthode de Recalage d'Image utilise un modèle de transformation affine, simple et adéquat lorsque l'intervalle de temps entre les prises de vue est relativement faible. Nous avons montré la grande répétabilité des Coins Dominants en présence d'une Transformation Affine au Chapitre 6, paragraphe 6.5, c'est la raison pour laquelle nous les avons utilisés. Une application de notre méthode pourrait être la surveillance de trafic routier à l'aide d'un drone aérien, où la difficulté est d'extraire le mouvement de petites cibles : les véhicules, à partir d'une caméra elle-même en mouvement. Le recalage permet la compensation du mouvement « global » de la caméra.

1.2.1. Sélection automatique des Coins Dominants

Le taux de compression *CR* n'est pas un critère d'arrêt optimal pour cette application, car un nombre différent de Coins Dominants est obtenu pour les images courante et de référence. En revanche, le ratio *r* entre les aires *GISE* initiale et finale l'est.

$$r = \frac{GISE}{GISE_0} \tag{1.4}$$

La GISE est une aire globale, somme des aires locales LISE. Le rapport entre les aires est un paramètre invariant par transformation affine, donc reste un invariant par transformation affine. De plus, la variation d'aire due à la suppression d'un Coin LISEV est également préservée par transformation affine. Ainsi l'élimination itérative des Coins, basée sur la variation d'aire permettant d'obtenir la plus petite valeur permet de garder des Coins Dominants qui se correspondent dans les deux images.

1.2.2. Construction des Primitives

Chaque quadruplet de Coins Dominants successifs constitue une primitive comme présenté sur l'image de la feuille de la Figure 1.11 (a). Le rapport R des deux triangles présenté Figure 1.11 (b) forme la première mesure invariante de la primitive.

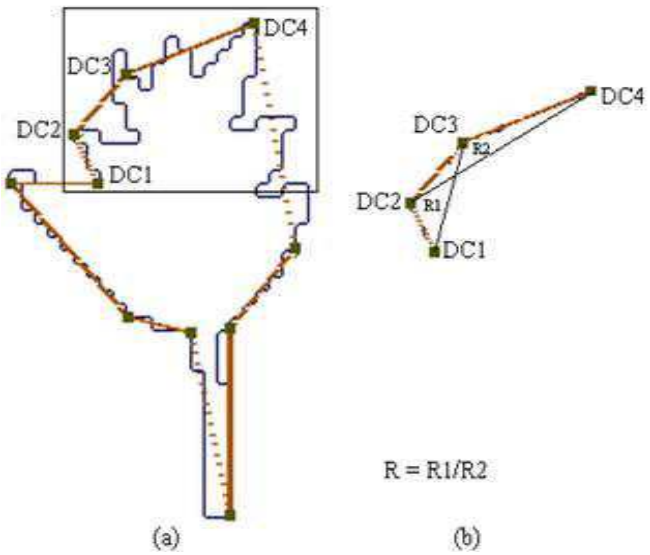


Fig.1.11. (a) Construction de Primitives sur la feuille. (b) Les deux aires triangulaires construites à partir de la Primitive, et la première mesure invariante R.

Les angles entre les Coins Dominants constituent la seconde mesure invariante issue du quadruplet de Coins Dominants de la primitive. Par l'expérience, nous avons apprécié la répétabilité des angles sous différentes transformations affines.

1.2.3. Mise en Correspondance de Primitives

Une primitive de l'image courante est associée à une primitive de l'image de référence, si et seulement si :

- Les deux primitives ont le même rapport d'Aires : *R*.
- Les Coins Dominants se correspondant ont les mêmes angles.

1.2.4. Estimation du Modèle de Transformation par Transformée de Hough

La relation entre les Points correspondante à une transformation Affine est:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (1.5)$$

où (x',y',1) et (x,y,1) sont les coordonnées homogènes des deux Coins Dominants relies par la transformation. a_{ij} sont les 4 paramètres de la transformation linéaire associée, tx,ty sont les deux paramètres de translation.

Comme la transformation comporte 6 paramètres, l'espace de Hough utilisé est de dimension 6. Les paramètres de la transformation linéaire a_{ij} s'étendent sur l'intervalle [-2;2] par pas de 0.01, ce qui représente 400 divisions par paramètre. Les paramètres de translation s'étendent sur l'intervalle [-200;200] par pas de 5, ce qui représente 80 divisions par paramètre. Deux modèles de transformation affine sont identiques si les 6 paramètres appartiennent à la même division.

Supposons $DC_1(x_1,y_1)$, $DC_2(x_2,y_2)$, $DC_3(x_3,y_3)$ et $DC_4(x_4,y_4)$ les 4 Coins Dominants successifs constituant la primitive de l'image de référence, et $DC'_1(x'_1,y'_1)$, $DC'_2(x'_2,y'_2)$, $DC'_3(x'_3,y'_3)$ et $DC'_4(x'_4,y'_4)$ de l'image courante. Le modèle de transformation affine de l'équation Eq. (1.5) peut être ré-écrit de la manière suivante: Eq 1.6 et Eq 1.7.

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ tx \end{pmatrix} or\ X' = \mathbf{M}.h \tag{1.6}$$

$$\begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \end{pmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{pmatrix} a_{21} \\ a_{22} \\ ty \end{pmatrix} or\ Y' = \mathbf{M}.h' \tag{1.7}$$

où les vecteurs h and h' contiennent les paramètres de la transformation affine et peuvent être estimés par les équations Eq 1.8 et Eq 1.9 :

$$h = \mathbf{M}^{-1}.X' \tag{1.8}$$

$$h' = \mathbf{M}^{-1}.Y' \tag{1.9}$$

Finalement, le modèle de la transformation, contenu dans h et h', est celui qui obtiendra le plus grand nombre de vote dans l'espace de Hough précédemment décrit.

1.2.5. Notre Contribution

Nos deux principales contributions dans cette application sont:

- la construction d'une nouvelle primitive à partir d'un quadruplet de Coins Dominants consécutifs. L'importante répétabilité des coins dominants

implique une importante répétabilité de cette nouvelle primitive. Ceci permet d'augmenter la robustesse par rapport à d'autres primitives.

- la Mise en Correspondance de primitives, basée sur des paramètres invariants : rapport d'aire et angles entre Coins Dominants.

1.3. Seconde Application: Reconnaissance de Caractères

Notre algorithme a été conçu pour « attaquer »le système CAPTCHA dans le but de segmenter et de reconnaître les caractères d'une image. Nous ne sommes pas des « hackers » cherchant à pirater des accès d'applications internet utilisant le système CAPTCHA comme outil de sécurité. Au contraire, notre but est de trouver des faiblesses du système existant de manière à le sécuriser et à le rendre plus robuste. Notre algorithme pourra ensuite être employé dans une autre application de robotique interactive. L'opérateur présentera l'information à donner au robot (commande ou autre) sous forme d'écriture manuscrite.

L'algorithme est basé sur les Coins de Contour (CCs). Chaque Coin de Contour est caractérisé par son angle "*Ang*" et les longueurs « *L1, L2* »des deux segments adjacents.Selon ces caractéristiques, nous avons défini trois types de CCs :

- CCsForts: *Ang* > 90^o et *L1,L2*>10 pixels. Par exemple, les points A et D sont des Forts CCs sur le« Z » de la Figure 1.12
- CCs Faibles: *Ang*> 90^o et soit *L1* < 10 soit *L2* < 10. Par exemple, les points B et E sont des Faibles CCs sur la Figure 1.12
- Coins de Liaison qui apparaissent entre deux caractères connectés.

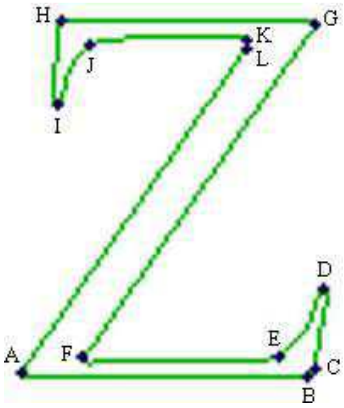


Fig.1.12. CCs sur l'image du caractère Z.

1.3.1. Schéma Proposé : Segmentation et Reconnaissance Simultanées des Caractères

Les Figures 1.13 et 1.14 présentent la reconnaissance d'une image déformée composée de deux caractères : G et 3.

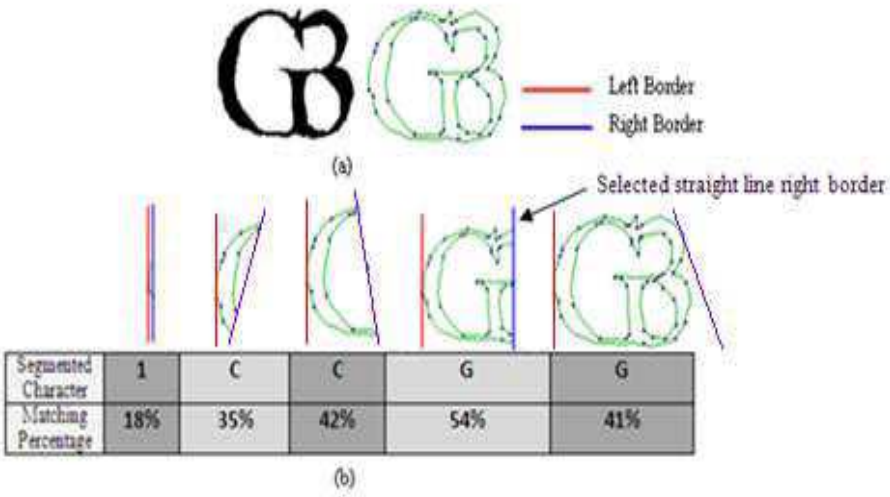


Fig.1.13. Recherche de la meilleure Ligne constituent la Frontière Droite, segmentant 2 caractères déformés et connectés. (a) image déformée et ses CCs. (b) différentes segmentations et les reconnaissances correspondantes avec leur taux.

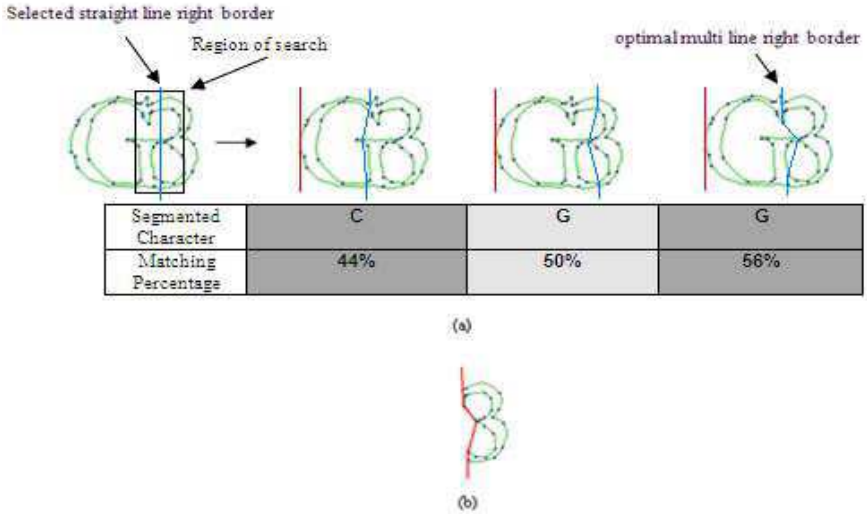


Fig.1.14. Segmentation Optimale utilisant une ligne brisée comme frontière droite.
(a) Trois différentes tentatives de segmentation utilisant une frontière sous forme de ligne brisée, et leur taux de reconnaissance. (b) La partie restante à reconnaître par la procédure.

La reconnaissance est compose de quatre phases:

- La frontière gauche (en rouge) est initialisée sous forme de ligne brisée. La frontière droite (en bleu), également sous forme de ligne brisée passant par les CCs, est recherchée par un déplacement sur la droite. Chaque essai est appelé "Tentative de Segmentation".

- Pour chaque Tentative de Segmentation, la partie de l'image comprise entre les frontières gauche et droite est associée à tous les caractères ou symboles de test. La sortie de cette tentative est un taux de reconnaissance. Cinq tentatives de segmentation sont présentés Figure 1.13 (b). L'algorithme retient la Tentative de Segmentation ayant le taux de reconnaissance le plus élevé, et la frontière droite correspondante, sous forme de ligne brisée passant par les CCs.

- Les CCs de la frontière droite retenue, ainsi que tous les CCs voisins sont présentés Figure 1.14 (a). La frontière est la séquence de segments passant par les CCs. Trois différentes frontières possibles sont présentées Figure 1.14 (a), avec leur taux de reconnaissance. La frontière optimale, avec un taux de reconnaissance de 56% est associée au caractère « G ».

- La frontière droite optimale du caractère courant devient frontière gauche du caractère suivant, cf Figure 1.14 (b).

L'algorithme détaillé est présenté Figure 1.15.

- *Les bords LB et RB sont des lignes verticales passant par le premier coin à gauche de l'image à examiner.*
- *Boucle:*
- *Tant que (RB ne passe pas par le dernier coin à droite)*
  - *Segmenter la partie de l'image comprise entre LB et RB.*
  - *Reconnaitre cette partie par la comparaison avec les images des caractères originaux.*
  - *Sortir le pourcentage de la reconnaissance.*
- *La partie segmente qui possède le pourcentage de reconnaissance le plus haut représente un caractère original reconnu.*
- *Si le RB correspondant ne passe pas par le dernier contour coin à droite.*
  - *RB sera LB.*
  - *Aller à Boucle.*
- *Sinon*
  - *Fin de l'algorithme: Tous les caractères sont segmentés et reconnus.*

Fig.1.15. L'algorithme proposé.

1.3.1.1. Segmentation

Le but de la segmentation est de trouver les frontières gauche (FG) et droite (FD) de chaque caractère. La première FG est la ligne verticale passant par la plupart CCs à gauche. La frontière de droite est une ligne brisée qui sélectionne à chaque Coin de Contour, un segment parmi un ensemble de M-Segments comportant un segment vertical et *M-1*segments dont les variations d'angles sont comprises entre -45<Δθ<+45. La Figure 1.16 représente l'ensemble de M segments à partir du coin CC de coordonnées (32,35) avec M = 5.

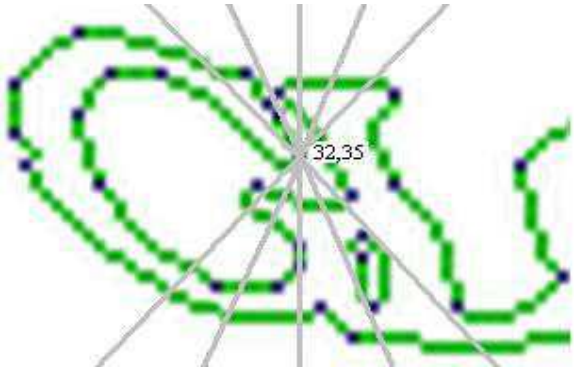


Fig.1.16. Ensemble de M segment au Coin CC(32,35) pour M=5.

Le meilleur segment de la frontière de droite, extrait de l'ensemble de M-Segments, permettant le taux de reconnaissance le plus important, est de sorte que la ligne brisée comporte un maximum de Coins de Contour. Pour cela les coins voisins dont la différence d'abscisse est inférieure à 10 sont pris en compte. La Figure 1.17 illustre ce principe. Le meilleur segment de la frontière droite est présenté en gris et le Coin de Contour correspondant CC(29,30).

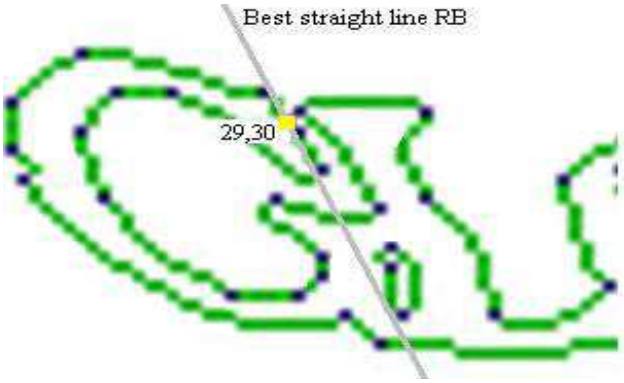


Fig.1.17. Meilleure Frontière de Droite pour le Caractère G.

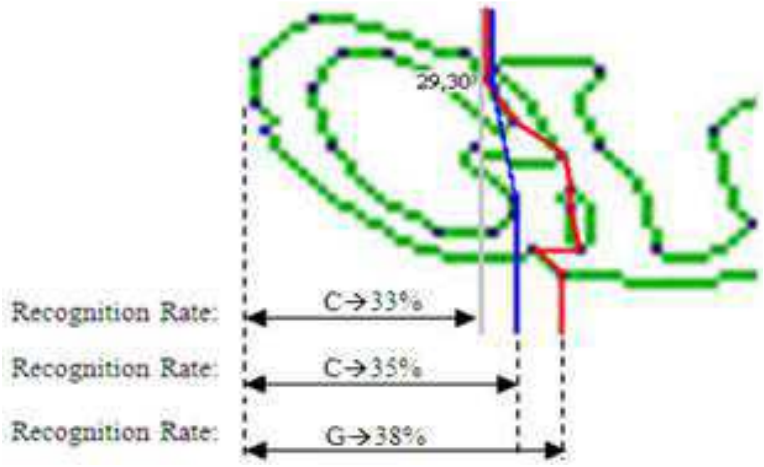


Fig.1.18. Trois Frontières Droites à partir du CC de coordonnées (29,30).

La Figure 1.18 présente trois frontières droites potentielles sous forme de lignes brisées, ainsi que leur taux de reconnaissance. Pour la frontière grise, le caractère reconnu est un « C » et le taux de reconnaissance est de 33%. Pour la frontière bleue, le caractère reconnu est également « C » avec un taux de 35%. Pour la frontière rouge, le caractère reconnu est « G » avec un taux de reconnaissance de 38%. Ainsi, la frontière droite optimale est la frontière rouge. Elle devient la nouvelle frontière droite pour le prochain caractère à segmenter.

1.3.1.2. Reconnaissance

Après chaque tentative de segmentation, une image segmentée est obtenue cf Figure 1.19 (a). Elle sera associée à chaque caractère de la base de test. La Figure 1.19 (b) présente l'image test du caractère (caractère G) correspondant à l'image segmentée de la Figure 1.19 (a).

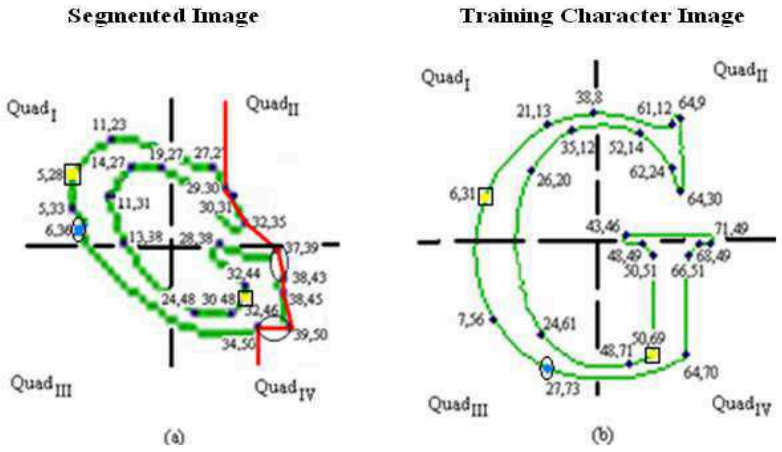


Fig.1.19. (a) Image Segmentée. (b) Image du Caractère Correspondant de la base de test.

Trois paramètres sont utilisés pour mettre en correspondances les CCs :

- Leurs angles.
- Le nombre de contour traversés de chaque côté de l'angle d'un CC. Pour illustrer la méthode, considérons le CC (11, 31) de la Figure 1.20. On dessine 3 lignes droites avec une déviation de pente de 45^{o} et on enregistre le nombre de contours traversés par chaque ligne droite de chaque côté de l'angle. Par exemple, pour la ligne droite « 2 » le nombre de contours traversés sont 1 de côté gauche et 2 de côté.Ce paramètre est utilisé pour éliminer un certain nombre d'associations erronées. Sans lui, les CCs (5,28) et (11,31) ne peuvent pas être différenciés en utilisant seulement leurs angles.
- L'index binaire ou l'index de quadrant. Le caractère segmenté comme le caractère test sont divisés en quatre quadrants, cf Figure 1.19. Chaque CC appartient à l'un d'entre eux.

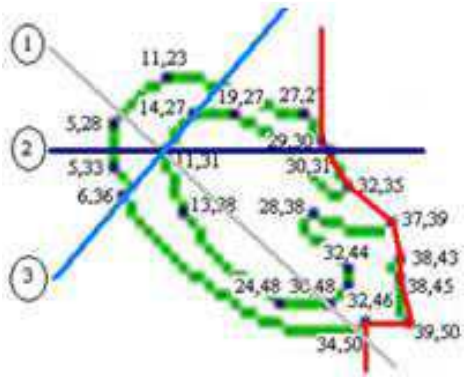


Fig.1.20. Trois lignes droites dessinées à partir de CC(11, 31) pour trouver le nombre de contours traversés.

Deux CCs, l'un de l'image de caractère de la base de test et l'autre de l'image segmentée, sont -soit Coins Trés Probablement Equivalents (CTPE), -soit Coins Equivalents ou bien Coins non Equivalents.

1.3.1.2.1. CTPEs

Les conditions sont:

- Les deux CCs et leurs CCs directement chainés ont le même angle.
- Ils ont le même nombre de contours traversés.

- Ils appartiennent au même quadrant.

CCs (5,28)-(6,31) et (32,46)-(50,69) sont marqués en jaune respectivement dans les deux images dans la Figure 1.19 sont CTPEs.

1.3.1.2.2. CC Mis en Correspondance

Les conditions sont:

- Ils appartiennent au même quadrant avec une certaine déviation t du quadrant correspondant.
  - A cause de la déformation globale, certains CCs peuvent être déplacés d'un quadrant à un quadrant voisin. Dans la Figure 1.19, CCs (6,36) et (27,73) sont Equivalents. Mais, CC (6,36) est déplacé du quadrant 3 au quadrant 1.
- Ils doivent avoir les mêmes nombres de contours traversés sauf pour ceux qui traversent les frontières gauche ou droite.
  - A cause de la fusion de deux caractères successifs, quelques segments de contour, localisés aux frontières du caractère, peuvent avoir disparus. Dans la Figure 1.19 (a), les segments disparus sont entourés. C'est pour cela on ne peut pas considérer le nombre de contours traversés par une ligne qui à son tour traverse les frontières.

Un CC de l'image de base peut être Equivalent à plusieurs CCs de l'image segmentée.

1.3.1.2.3. Système basé sur la Logique Floue

Un Système basé sur la Logique Floue est proposé pour évaluer le taux de correspondance entre le caractère à reconnaitre et les caractères de la base de test. La Logique Floue est utilisée compte tenu de la déformation aléatoire des caractères à reconnaître. Le schéma bloc de l'algorithme est proposé dans la Figure 1.21.

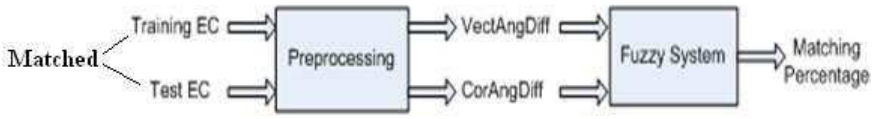
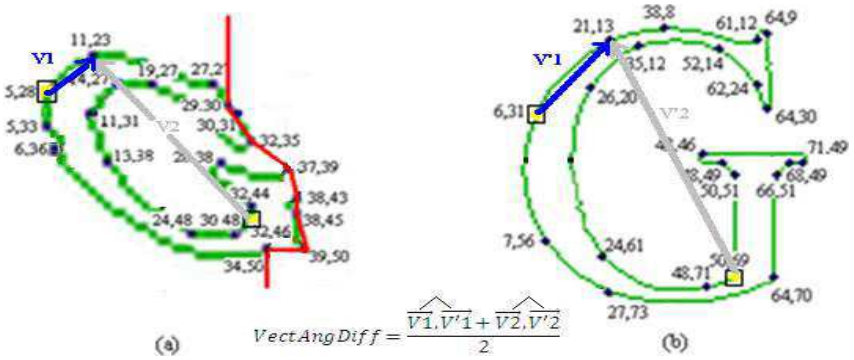


Fig.1.21. Schéma Bloc du Système à Base de Logique Floue.

Le caractère à reconnaitre et le caractère de la base de test sont fournis sous forme de Coins associés. La sortie de la première étape est constituée de deux paramètres : « Vector Angle Difference » "*VectAngDiff*" et« Corner Angle Difference » "*CorAngDiff*". Ces paramètres sont les entrées du "Fuzzy System" qui calcule le taux de correspondance.

VectAngDiff est illustré Figure 1.22. Considérons deux Coins de Contour associés (11,23) et (21,13). Dans chaque image, dessinons le vecteur reliant le CoinauxCTPEs. Deux vecteurs, un dans chaque image, sont dits correspondants si leurCTPEs sont correspondants. Les CTPEs correspondants sont colorés Figure 1.22 et sont nommés (V1,V'1) et (V2,V'2). *VectAngDiff* est la moyenne des "vectors angles differences".

CorAngDiff est illustré dans la Figure 1.23, en utilisant la même paire de Coins. La direction moyenne "*ASE*" de deux segments de contour successifs est enregistrée. On a deux "*ASE*" correspondant aux deux coté de l'angle. La différence entre ces deux *ASE* est enregistrée comme *CorAngDiff*.



$$VectAngDiff = \frac{\widehat{V1,V'1} + \widehat{V2,V'2}}{2}$$

Fig.1.22. VectAngDiff entre les vecteurs correspondants
de deux CCs Equivalents (11,23) et (21,13).

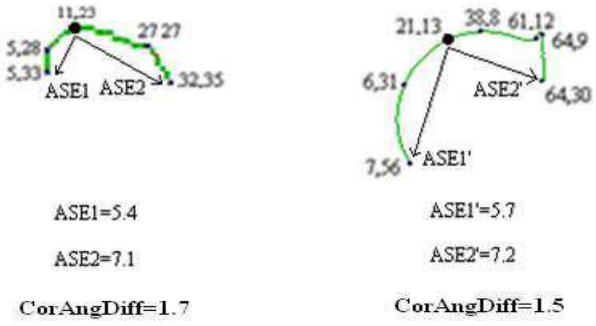
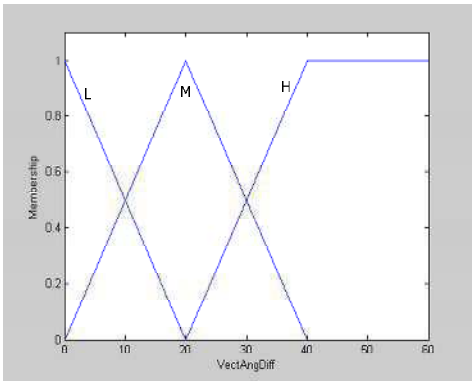


Fig.1.23. "Corner Angle Difference" en un CC.

Le système, présente Figure 1.21, a deux entrées: "*VectAngDiff*" et "*CorAngDiff*" et une sortie "*Matching Percentage*". Alors, la "Fuzzification" commence par la sélection d'une fonction de membre pour chacun comme présenté Figure 1.24.

Nous avons choisi ces formes particulières parce qu'elles mènent expérimentalement aux taux d'association les plus élevés.



(a)

L M H

(b)

Alike Like

(c)

Fig.1.24. Fonctions Membres.

Les lois de flou sont résumées Table 1.1. Les utilisant, le taux d'association en sortie sera "Like" ou "Alike" avec un pourcentage entre 0 et 1.

Table 1.1. Lois de Flou.

		VectAngDiff		
		L	M	H
CorAngDiff	L	Like	Like	Alike
	M	Alike	Alike	Alike
	H	Alike	Alike	Alike

Pour ce taux d'association, on trouve les pourcentages associés à "Like" et "Alike" en utilisant les inverses des fonctions gaussiennes cf Figure 1.25.

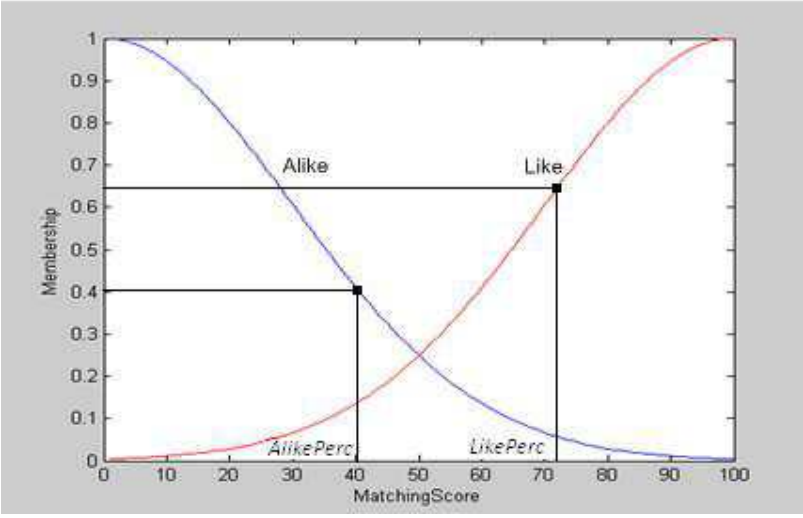


Fig. 1.25. Calcul des Pourcentages "Like" et "Alike" à partir des fonctions inverses.

Finalement, le pourcentage d'association peut être calculé:

$$MatchingPercentage = \frac{Like * LikePerc + ALike * ALikePerc}{Like + Alike} \qquad (1.10)$$

Les Coins associés, des caractères à reconnaître et de la base de tests, ayant le plus haut score d'association sont marqués. Après chaque association de tous les Coins, l'étape de Vérification est une étape très importante pour la reconnaissance. Nous vérifions l'ordre des liaisons entre les Coins marqués des caractères à reconnaître et de la base de test. S'ils sont reliés correctement, le calcul du pourcentage d'association global continue. Sinon, les caractères ne se correspondent pas.

Pour le pourcentage d'association global, on accumule pour chaque CC de base marqué le pourcentage du nombre des points de contour équivalents. Cela est fait par la multiplication du "*MatchingPercentage*" par les longueurs des segments de contour adjacents.

$$nbptsM \mathrel{+}= MatchingPercentage * adjacent\ SEs\ lengths \qquad (1.11)$$

On calcule également le nombre total des points de contour "*nptsT*" du caractère de base. Ainsi, La pourcentage d'équivalence totale "*OveralMP*" est égale à :

$$OveralMP = nbptsM*100/nbptsT \quad (1.12)$$

Finalement, l'image segmentée est reconnue comme l'image du caractère de base correspondant à la valeur maximale de *OveralMP*.

1.4. Conclusion

Dans cette thèse nous avons introduit un nouveau détecteur de Coins, basé sur les contours de l'image. Il obtient de meilleures performances que les détecteurs de points d'intérêt de la littérature, notamment en termes de répétabilité. La première étape est la détection des contours. Puis un détecteur de segments de droite examine tous les contours pour détecter les parties pouvant être approximées par des segments de droite. L'intersection de deux segments de contours successifs d'une longueur minimale, non colinéaires forme un Coin.

Les Coins obtenus sont robustes vis-à-vis des déformations entre images, notamment en cas de transformation affine, ainsi qu'en cas de variation d'échelle, sans incorporer dans leur détection un invariant vis-à-vis de la variation d'échelle. Les résultats expérimentaux prouvent que notre « Edge Based Corner Detector » (EBCD) est très intéressant, comparé aux détecteurs existants. Nous avons proposé une application de Reconnaissance de Forme 2D l'utilisant. La reconnaissance est correcte, même en présence de plusieurs formes.

Basée sur les Coins de Contour, une nouvelle technique d'Approximation Polygonale est également proposée. En fixant comme critère d'arrêt le taux de Compression *CR* ou la valeur de l'erreur sur la surface, l'algorithme supprime un par un les coins de contour de manière à garantir une erreur minimale en surface. Les coins restants lors de l'arrêt, appelés Coins Dominants sont les sommets de l'Approximation polygonale, approximant de manière optimale le contour. Les résultats expérimentaux sont bons, comparés aux méthodes de la littérature.

Compte tenu de la répétabilité des Coins Dominant lors des Transformations Affines, observée lors de nos expérimentations, nous les avons utilisés lors d'une première application au Recalage d'Images. Ainsi nous avons proposé une nouvelle primitive formée d'un quadruplet de Coins Dominants successifs. Le rapport entre les aires des deux triangles formés ainsi que les angles entre Coins Dominants sont des

invariants par Transformation Affine, utilisé pour la mise en correspondances de primitives entre l'image courante et l'image de référence.

Finalement, un algorithme de Segmentation – Reconnaissance simultanée de caractères a été proposé pour reconnaître les mots du système CAPTCHA de Yahoo. L'objectif en reconnaissant ces mots n'est pas de devenir un « hacker », mais de mettre en évidence les faiblesses du système. L'algorithme proposé classifie les Coins de Contour en Forts, Faibles et Coins de Connexion entre caractères. Pour segmenter les caractères, la notion de frontières gauche et droites est introduite. Ces frontières passent au minimum par un Coin de Connexion.

Ces frontières passent nécessairement par au moins un point de connexion et la partie d'image entre les frontières est la partie à segmenter. Cette partie est à l'entrée de la phase de Reconnaissance. Si elle est reconnue comme un caractère de la base de tests, l'algorithme continue avec la reconnaissance du caractère suivant de l'image. Pour pallier le problème de la déformation aléatoire de chaque caractère, nous avons utilisé un système de Logique Floue permettant de reconnaitre les caractères à partir de leurs Coins de Contour. Les résultats de nos expérimentations ont montré la flexibilité de notre système face à la déformation et à la connexion des caractères. Le détail de nos algorithmes sera utile pour améliorer les manières de déformer et de connecter les caractères des systèmes comme MSN ou Google CAPTCHA, de manière à être plus robuste contre les « hackers ».

Plusieurs applications robotiques peuvent être développées. La première est la reconnaissance de formes basée sur les Coins de Contour que nous avons développée pour des applications plus complexes telles que la reconnaissance d'objets où plus d'un contour existent. Dans cette application, la mise en correspondance des coins de manière individuelle n'est pas suffisante, car elle génère un nombre important de fausses associations. Par conséquent, une nouvelle stratégie de mise en correspondance a été proposée, basée sur la mise en correspondance d'ensemble de coins de contour. Chaque contour de l'image test est associé à un contour de l'image courante en utilisant une distribution de contours associés où des coins de contours faible peuvent apparaître ou disparaître.

La seconde est la surveillance de route à partir de drones. La compensation du mouvement global important de la caméra permet l'estimation du mouvement local

des cibles de petite taille en mouvement. Elle peut être réalisée grâce à notre procédure de recalage d'images.

La troisième est la communication entre l'être humain et le robot, par écriture manuscrite. L'homme écrit l'information à transmettre au robot, et le robot la reconnait.

2

Introduction

Computer vision is a field responsible for analyzing and processing the acquired image to extract numerical and symbolic information from images [1, 2] in order to output the appropriate result. This field has many applications ranging from tasks such as machine vision systems to research into artificial intelligence and computers or robots. Machine vision usually refers to a process of combining computer vision methods and technologies to provide automated inspection and robot guidance in industrial applications. For example, a mobile robot typically uses computer vision for navigation, for producing a map of its environment (SLAM) and for detecting and avoiding obstacles. In addition, simple low level algorithms have been introduced so far for a better visualization of the image.

As stated before, the starting point in all computer vision techniques is to extract information from images that must be robust to different variations in real world applications concerning illumination, translation, rotation and scaling and also to noise like cluttered scene due to background. Here, the notion of "interest local features" becomes relevant. Local features have been shown to be well suited to matching and recognition as well as too many other applications as they are robust to occlusion, background clutter and other content changes. They are well defined information extracted from an image and that are spatially localized at specific locations in that image in the form of isolated points, continuous curves or connected regions.

Feature detection is usually performed as the first operation on an image; it examines every pixel to see if there is a feature present at that pixel. For real time applications, the features neighboring regions can be the only image parts used to extract the needed information. In this way, the required processing time and the storage amount can be minimized.

There are a lot of feature detectors that resemble on searching on informative features in an image but vary widely in the kinds of features detected, the computational complexity and the repeatability. Here, repeatability is an important measure to classify a feature detector, it is the stability of the feature in the same location in an image even after applying to it different transformations especially rotation and scaling and after adding to it some noise. The feature matching comes after the feature detection problem and forms a central process to any intelligent

activity. The success of the matching relies on the quality of the features and their description but also on the quality of the matching procedure. First, the extracted features have to be stable enough against temporal changes due either to motion, viewpoint, lighting variations or blurring. Therefore, the features have to be salient and accurately characterized in order to avoid any matching ambiguities. Second, the matching procedure has to rely on a similarity measure that correctly distinguishes the features in order to avoid false pairing.

2.1. From feature detectors to applications

Computer vision is a very important field of research with a huge set of applications. According to the nature of the application, a feature detector tries to find and explore some image parts [9,10,11]. Any detector developed to extract this information in real scenes should have several properties. In application like object recognition, it should be able to form a specific representation of the existing objects like object's descriptors [12] or object's model that could identify an object from others. On the other hand it should be robust to noise and to various image transformations and changes like viewpoint, rotation, illumination, translation and scaling. In addition, it should be fast enough especially when used for real time applications.

A very large number of feature detectors have been developed so far. They differ in the nature, the number and the repeatability of features detected and in the computational complexity.

Some detectors use a smoothing step before start searching for the image features. The smoothing concept is introduced in the scale space theory [3,4,5] and the smoothing level introduced to an image is represented as a scale level in this theory. Thus, the smoothing degree can reveal the size of objects within an image. While increasing the smoothing degree, objects vanish from the image. Small objects vanish first and larger objects later. The smoothing level at which an object vanishes basically reflects the size of this object. For this reason the smoothing parameter is also called “scale”. It is used for two main reasons either to smooth the transition between image regions of different contrast or to study the repeatability of image features over scale.

There is a variety of feature detectors that differ by the technique used to extract these features. Some detectors can be applied directly on the image intensities to extract their features. Whereas other detectors are applied indirectly on image intensities since they are based on the features detected by other detectors. For example, a corner detector can detect corner points that are edge points. So an edge detector should be applied prior the corner detection. In general, the feature detectors can be divided into four groups: edges, corners, blobs and ridges.

In practice, edges are one of the most important features. They are usually defined as sets of points in the image which have a strong gradient norm. This means that an edge is the boundary between two regions of different contrast. In addition, image corners are also very used features. Earlier, corners were points on the edge where a rapid change in direction occurs. Corners detectors were applied on edges and the detected points are called edge corners. Later on, the search of corners starts including all image parts not only image edges. Therefore, corners become image points corresponding to a high level of curvature in the image gradient. These points are frequently known as interest points. In the other hand, blobs are groups of pixels in the image that preserve some common property across it like intensity values, as opposed to corners that are more point-like. Therefore, blobs are image parts in the image space constructed by pixels of similar contrast. They can be used to form models or descriptors for objects that can be recognized by their color distribution. Finally, a ridge has been proposed as a useful feature for image analysis [6,7] and has been successfully applied to image segmentation [8]. Nevertheless, ridge descriptors are frequently used for road extraction in aerial images and for extracting blood vessels in medical images. Its goal is to capture the interior of elongated objects in the image domain.

Interest point is an important image feature, presented in a lot of works in image processing domain, detected directly using the image intensities in its local neighborhood. Many applications are based on interest points in general like stereo matching, object modeling and recognition [13,14], image registration [15], video tracking, pose estimation and SLAM [16]. Using stereovision technique in a mobile robot, a 3D model of a 3D polyhedral object can be formed from the distribution of its 3D interest points. Therefore, for an autonomous moving robot, these interest points

are used to localize the robot and build a map of its surrounding environment (SLAM).

2.2. Suggested feature detector and its applications

We have proposed a feature detector, called Edge Based Corner Detector (EBCD), whose aim is to provide features points that can be more robust to noise and less time consuming for some image processing applications.

The purpose is to segment the image first into feature segments, called "Straight Edges", than into feature points, called "Edge Corners" or simply "ECs", than select among the ECs a smaller set called "Dominant Corners" or simply "DCs". The Straight Edges are edge parts that are linked in the form of nearly straight lines. Therefore, a contour can be seen as a sequence of Straight Edges. Their detection is very robust to the normally introduced noise on the image edges. Than the ECs are edge points that correspond to the intersections of every non collinear Straight Edges. Finally, DCs are only the ECs having a strong repeatability under the affine transformation. Therefore, a classification method is developed to eliminate week ECs to obtain those strong ones (DCs). This classification is known as polygonal approximation.

The obtained DCs are characterized by their repeatability under affine transformation. Therefore, we have suggested some robotic applications to benefit from this important characteristic. The first one is an image registration application where the goal is to determine the geometric model or transformation that aligns the image points in the two studied images, called source and sensed images. The deformation relating these two images should be very small in order to be well estimated by an affine model. Practically, this can be obtained when these images are acquired sequentially for the same scene with a small time interval. This registration method can be used to model the global motion of a drone's camera, monitoring a certain road, in the purpose to estimate the real motion of small moving targets, e.g. cars, on the road. The second application is a character recognition application designed for autonomous robots in a human-robot interaction. A human presents to the robot some written commands as words containing one or more characters that can be connected or not. The robot acquires the commands as images and should

recognize the characters and identifies the order. This application is simulated and tested in this thesis as a CAPTCHA breaker. The CAPTCHA is a computer program that generates randomly deformed words containing one or more characters. The random deformation introduced to each character simulates the real deformation in handwriting. In addition, the ECs have shown a strong repeatability under these deformations and so they are used as landmarks in this application. However, we are not seeking to play the role of a hacker and steal an access to log on to certain internet application. Due to our study and success to break these CAPTCHA schemes, one can improve these schemes to be more robust.

Our proposed detector is based on ECs rather than interest points. The search of ECs is restricted only on edges rather than all the image space. It is defined as an edge point where a change in the edge direction occurs. It is more robust than an interest point since it is characterized by its angle and the length ratio of its two adjacent segments. Therefore, the choice is on ECs since the edges are very fast to extract and are less sensitive to noise.

2.3. Thesis outline

Chapter 3 presents the Interest point/Corner features background, edge detectors background and an image registration background.

Chapter 4 presents our segmentation work starting by edge segments, to edge corners (ECs), than dominant corners (DCs) with their invariant parameters.

Chapter 5 shows the experimental results on ECs and DCs. The ECs results are compared to various existing interest point detectors. The DCs extraction which is the result of a polygonal approximation method is compared to other polygonal approximation techniques in the literature.

Chapter 6 presents our first application on image registration by showing the repeatability of the dominant corners versus affine transformations, the primitive construction, the primitive invariant parameters and the proposed voting scheme.

Chapter 7 presents our second application on character recognition by presenting first overview on existing techniques, than explaining the suggested system used to

recognize the characters in a deformed word image, and finally presenting the experimental results.

Chapter 8 presents the conclusion and future works.

3

Background:

Interest Point Features

Interest points have been used in many computer vision tasks, such as image registration [52], image matching [91], object recognition [92, 93] and motion analysis [94]. Broadly speaking, there are two different interest point detection strategies adopted in literature. The first [61, 62, 99-104] is based on the analysis of pre-segmented contours and classified as edge corners detectors, while the second [28, 107, 108] is based on the differential analysis of the raw gray-scale image and classified as interest point detectors. Therefore, an interest point is a well defined point in the image space and it is easy to detect and represents certain variation in their local neighborhood.

Interest points in gray-scale images are characterized by using the first and second derivatives of the image luminance function. Although this method does not require pre-segmented image contours, it is sensitive to the noise amplification effects of the second-derivative operators. An interest point detector can be classified as a Template-based detector [97]. Template-based detector detects the similarity between a given oriented template image and each image sub-window. The points that correspond to maximum similarity are classified as interest points. Because multiple orientation templates are used, the technique is computationally expensive.

So our features classification can be as follows:

- Intensity based interest point features or simply interest points.
- Edge based corners and Edge Dominant Corners (DCs).

Our contribution is on the second class of features. The major strength of our new proposed features is in their robustness to noise and scale variation. They are based on edges that are less sensitive to noise and don't require a large computation time. However, existing interest point detectors are intensity based or region based. This will make them more sensitive to the noise existing normally in real images.

Intensity based interest point features are compared to the newly suggested edge based corner features so Section 3.1 provides a survey on various interest point features overview. Section 3.2 presents first various edge corner detectors including a survey on various edge detectors that form the basis of the suggested corner detector. This section also presents various edge segmentation techniques based on polygonal approximation that are compared to our polygonal approximation technique using the edge dominant corners. Section 3.3 presents a survey on image registration techniques

that are compared to our image registration application using the edge dominant corners. Section 3.4 presents our contribution.

3.1. Interest point features overview

3.1.1. Interest points are local image features

In general, a feature is considered as a "piece of information" extracted from an image. The level of interest differ form one application to another. Extracted features form an image form the starting point for high level algorithms such as recognition, tracking … An important property of a good feature is its repeatability: it reflects the ability of a feature extracted from an image to be extracted again in the same image but under deformation like adding noise, affine transformations, illumination variations and occlusion. A very large number of feature detectors have been developed so far. These vary widely in the kinds of interest points detected, the computational complexity and the repeatability.

Some algorithms search for global features, named "global features detectors", that try to represent the whole image such shape contexts [63, 64] while others seek for local features that represent interest features located in specific regions in the image and they are named "local features detectors" or "interest point detectors". Many techniques are classified under this category and each searches for different features in the image. All of these interest point detectors are based on three different bases: some are distribution based [65-73], others are differential based [74] and others are spatial frequency like the ones that use gabor filters and wavelets [75-77]. The most famous distribution based techniques are appearance-based or model-based local feature detectors. As an example, face recognition algorithms [203, 204, 205] rely on special features in the human face that distinguishes it from other objects. So, they try to extract eyes, mouth [78, 79] and nose [80] from an image or rely on skin color [81].

Features are classified into three classes: Line features, region features and point features. Edges [39, 135], contours [41] and level lines [33] form the line features used to represent object contours, roads, coastal line, etc. Straight lines are very important image features used in remote sensing applications to register city or roads networks images. Regions [42, 201, 202] form the region features used to segment an

image into high contrast closed boundary regions that may represent particular objects: Forests [129], lakes [130] and buildings [131] are formed. Interest points [43] form the point features. These point features are image corners [28], line intersections, centroids of regions, curvature extremes, and others.

Next, we will present various interest point detectors.

3.1.2. Moravec detector

Moravec corner detector [99] is one of the earliest corner detectors. To check the presence of a corner at a given pixel C, a 3 x 3 window is placed at C and its four direct neighbors: horizontal, vertical and two diagonals as shown in Figure 3.1.

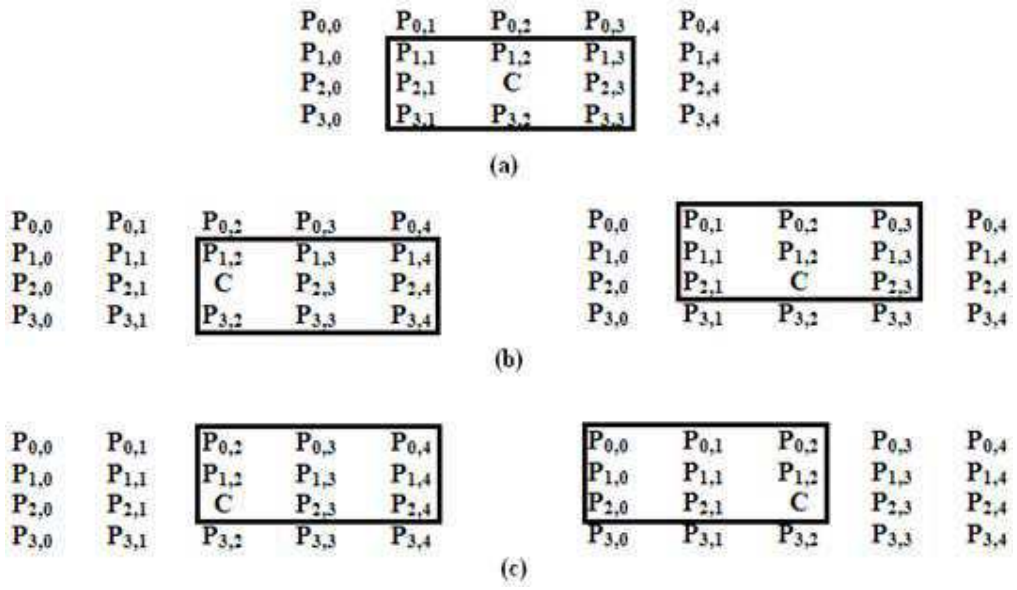


Fig.3.1. Original and shifted windows around the tested pixel C. (a) Original window. (b) Windows shifted horizontally and vertically. (c) Windows shifted diagonally [99].

The differences of the pixel intensities in the corresponding positions between the original window and the four neighboring ones are calculated. For each neighbor, the difference function is given by:

$$E(u,v) = \sum_{i,j} W(i,j).(f(i+u;j+v) - f(i,j))^2 \tag{3.1}$$

Where *W(i,j)* is one of four windows, *f(x,y)* is the image function and *(u,v)* are the coordinates of the corresponding shift. Their values are:

- (1;0): for the horizontal shift.
- (0;1): for the vertical shift.
- (1;1): for the diagonal shift up-right.
- (-1;1): for the diagonal shift up-left.

A corner is detected at pixel C if the minimum of the four values of *E(u,v)* is above a threshold value.

The major problem in Moravec detector is in the false detection of edge points as interest points. Sometimes when the edge changes its direction, it will be chosen as interest point. This weakness makes the Moravec detector not suitable for our robotic applications where the stability of the detected corners is very essential.

3.1.3. Harris detector

Moravec corner detector constructs the starting point for Harris corner detector [100]. Harris has entered several improvements to the Moravec detector to achieve better performance in detection and better robustness to noise.

The first drawback in Moravec detector is its anisotropic response since a discrete set of shifts at every 45^{o} is considered (only four discrete shifts are considered at every pixel). To overcome this fact, the shifts are considered in all directions as small shifts in continuous form. The shifts u and v in the x and y directions are approximated by a Taylor expansion up to $O(x^2;y^2)$ as follows:

$$f(i+u;j+v) \approx f(i;j) + f_x(i;j).x + f_y(i;j).y \tag{3.2}$$

Where f_x and f_y are the partial derivatives of the image function f.

The difference function E becomes:

$$E(x,y) \approx \sum_{i,j} W(i,j).(f_x(i;j).x + f_y(i;j).y)^2 \tag{3.3}$$

In matrix form:

$$E(x,y) \approx (x\ y)A\begin{pmatrix} x \\ y \end{pmatrix} \tag{3.4}$$

Where:

$$\boldsymbol{A} = \sum_{i,j} W(i,j).\begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix} \tag{3.5}$$

A is called the Harris matrix.

The second drawback in Moravec detector is its noisy response due to the usage of a hard shaped rectangular window. The Euclidean distances from the center pixel of the window to the edge pixels vary for different directions. This is easily resolved by using a circular window. Being binary puts equal emphasis on all intensity variation measures regardless of their distance from the center of the window. Intuitively, more weight should be put on measurements made closer to the center of the window like a Gaussian window:

$$W(i,j) = \frac{1}{2\pi t} e^{-(i^2+j^2)/2t} \tag{3.6}$$

The third drawback of Moravec detector is its response to edges so an incorrect detection of corners. In fact, any imperfections in an edge due to noise, pixilation, or intensity quantization may lead to a local minimum intensity variation over all shift directions that may lead to incorrect corner detection. Harris detector presents a way to distinguish between edge response and corner response. The eigenvalues of the 2 x 2 matrix **A** in Eq. (3.5) reflect the presence of a corner or an edge point because they represent the curvature at the tested point C.

- If the two eigenvalues are small, point C lies in a homogeneous region in intensity.
- If one of them is small and the other is large, point C lies on an edge. Moving along an edge has small intensity variation (small eigenvalue) while moving across it the variation will be considerable (large eigenvalue).
- If both eigenvalues are large, the point C represents a corner point. High curvature is present at this point.

Harris and Stephens have found a very big computation load while calculating the eigenvalues of the matrix **A**. So, they replace this calculation by the function *R*:

$$R = Det(A) - kTrace(A)^2 \tag{3.7}$$

Where k is a tunable parameter. *R* is positive on corner points, negative on edge points and very small in flat region.

The Harris detector, discussed so far, is partially invariant to affine intensity change and invariant to image rotation. However, it is not invariant to scale change. Figure 3.2 shows the decreasing in the repeatability of the Harris corner points due to the scale variation.

Therefore, the multi-scale Harris detector is introduced. Two Gaussian filters are used. The first one of scale *t*, named local scale, is used as a first step of the detection process to blur the image *f* and reduce the noise effect as shown in Eq. (3.8). The other of scale *s*, named integration scale, is used to smooth the image at different values of *s* in order to extract the corners at multi-scale space. *L(x,y,t)* is also called the smoothed image of *f(x,y)*.

$$L(x, y, t) = f(x, y) * G(x, y, t) \tag{3.8}$$

Where *G(x,y,t)* is 2D gaussian filter given by,

$$G(x, y, t) = \frac{1}{2\pi t} e^{\frac{-(x^2+y^2)}{2t}} \tag{3.9}$$

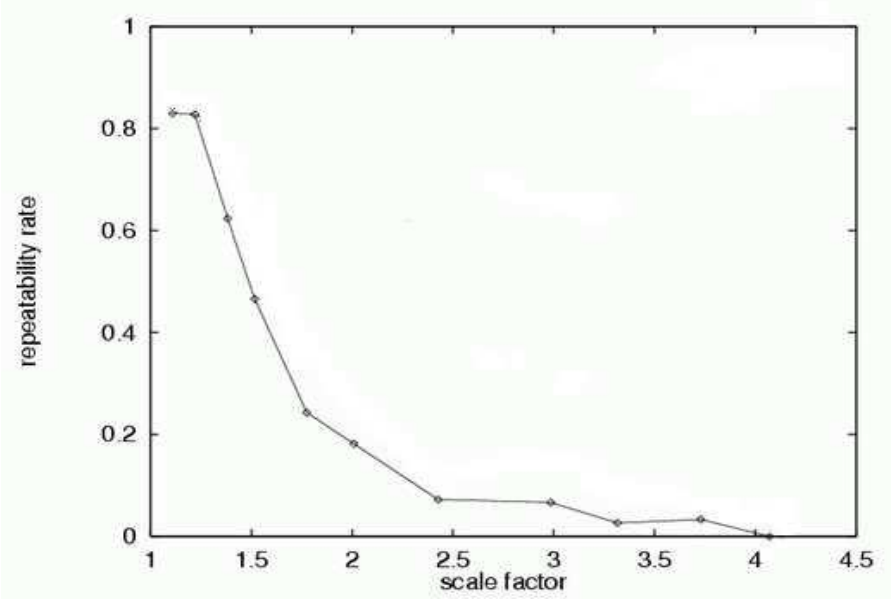


Fig.3.2. Performance degradation of Harris detector due to scale change [100].

The matrix **A** after integrated smoothing becomes the multi-scaled second moment matrix μ:

$$\mu(x,y,t,s) = \begin{bmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{bmatrix} * G(x,y,s) \tag{3.10}$$

$$= \int_{a=-\infty}^{+\infty} \int_{b=-\infty}^{+\infty} \begin{bmatrix} L_x^2(x-a;y-b;t) & L_x(x-a;y-b;t)L_y(x-a;y-b;t) \\ L_x(x-a;y-b;t)L_y(x-a;y-b;t) & L_y^2(x-a;y-b;t) \end{bmatrix} . G(a,b,s) da db$$

Where L_x and L_y are the first order partial derivatives of the smoothed image $L(x,y,t)$.

The same Harris function R, named now as multi scale Harris corner measure M_c, can be used here in multi scale to judge the presence of a corner:

$$M_c(x,y,t,s) = Det\big(\mu(x,y,t,s)\big) - k.trace\big(\mu(x,y,t,s)\big)^2 \tag{3.11}$$

The Harris detector presents an important improvement for Moravec detector but it is not very robust to noise considering real images that are used in our robotic applications. Comparative results are presented in Chapter 5.

3.1.4. Shi-Tomasi corner detector

The concept of this method [101, 102] is similar to Harris detector. Harris method relies on the matrix **A** that represents a local autocorrelation function while Shi-Tomasi or Tomasi-Kanade method relies on the inverse of matrix **A**.

$$\boldsymbol{A}^{-1} = \frac{1}{\langle f_x^2 \rangle . \langle f_y^2 \rangle - \langle f_x f_y \rangle^2} \begin{bmatrix} \langle f_y^2 \rangle & -\langle f_x f_y \rangle \\ -\langle f_x f_y \rangle & \langle f_x^2 \rangle \end{bmatrix} \tag{3.12}$$

At the tested point, if the minimum of the two eigenvalues of $\mathbf{A}^{-1}$ is above a threshold value than a corner is detected. Figure 3.3 shows the detected corners by Shi-Tomasi detector on a given image.

Fig.3.3. Corner points detected by Shi-Tomasi detector [101].

This detector is based on the Harris detector. However, it presents one slight variation in the selection criteria. It works quite well where even the Harris corner detector fails. However, it still suffers when it is applied to real images.

3.1.5. Level curve curvature approach

The Level curve curvature [61,62] is an approach to detect corner points where the curvature of level curves on the edge is very high. The rescaled level curve curvature $K(L,t)$ is a product between the gradient magnitude operator raised to the power three and the level curvature operator,

$$K(L,t) = L_x{}^2 L_{yy} + L_y{}^2 L_{xx} - 2L_x L_y L_{xy} \tag{3.13}$$

Where L_{xy}, L_{xx} and L_{yy} are the second order partial derivatives of the smoothed image $L(x,y,t)$.

In scale space, while moving from finer to coarse level which means from low to large smooth levels, the maxima of any function detected in the image space will decrease but remains maxima and also for the minima they will increase but remaining minima [4, 5]. Due to this fact, the corner detection depends on the selected scale and also is very sensitive to noise. A good solution is to normalize the function K using γ-normalization,

$$\mathrm{K_{norm}(L,t) = t^{2}(L_x{}^2 L_{yy} + L_y{}^2 L_{xx} - 2L_x L_y L_{xy})} \tag{3.14}$$

$$(\tilde{x};\tilde{y};\tilde{t}) = argmaxlocal_{(x,y,t)}(K_{norm}(x,y,t)) \tag{3.15}$$

Where γ = 7/8.

So extrema $(\tilde{x};\tilde{y};\tilde{t})$ of the obtained operator, shown in Eq. (3.15), are points having high gradient norms and also high curvature of level curves. These points $(\tilde{x};\tilde{y})$ are corner points that can be used as descriptors locations.

This detector introduces a third dimension "scale space" for selection the interest points. Therefore, the detected points will be nearly invariant over scale but their detection is very time consuming especially in robotic applications where a fast decision is required.

3.1.6. SUSAN detector

SUSAN stands for "Smallest Univalue Segment Assimilating Nucleus" [103]. The essential use of this detector is as a corner detector but it also can be used as an edge detector. The idea is to use the pixel brightness at a tested point and compare it to neighboring pixels to detect the interest point. To achieve that, a circular mask is placed at the tested point called "nucleus" and all the points within the mask are the neighboring points. The area of the neighboring points that have brightness similar to the one of the nucleus is the area of importance. This area is known as USAN. In addition, an important achievement in this approach is that it doesn't require any derivative calculation on the image function which reduces the noise effect and also it does not require prior noise reduction.

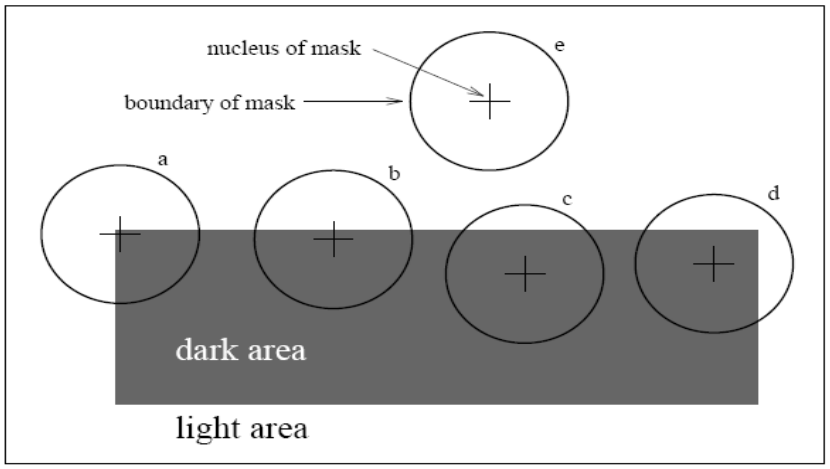


Fig.3.4. Circular masks applied in different nucleus positions in an image of a dark rectangle lying in a white background [103].

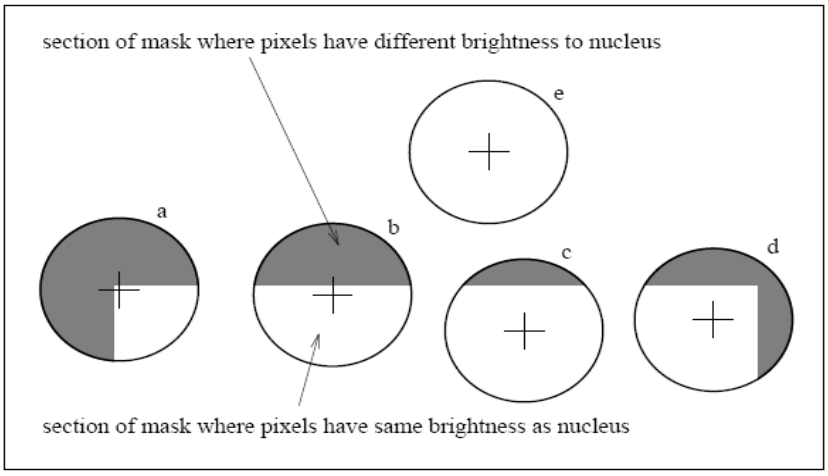


Fig.3.5. Corresponding USANs shown as white parts within the masks [103].

Figures 3.4 and 3.5 give a general explanation of this method. In the first figure, five different masks are taken in different regions of the figure. In the second one, the USAN area is represented by the white part inside the mask. It is clear that the USAN area is at a maximum if the nucleus of the mask lies on a homogenous region in brightness, while it falls to the half on an edge nucleus and also it falls more on a corner nucleus. As a result, when the nucleus of the mask moves from a flat region in brightness to an edge region, USAN attains a minimum value while it falls more when it attains a corner point. Thus, the notation smallest USAN (SUSAN) arises.

The used circular mask has a radius of 37 pixels. This mask is placed at every pixel (nucleus) in an input image and the brightness in all pixels in the mask is compared to the one in the nucleus as stated before. To give a similarity decision, a comparison function *C* is used,

$$C(\vec{r},\vec{r_0}) = \begin{cases} 1, & if\ ||f(\vec{r}) - f(\vec{r_0})\}|| \le t \\ 0, & if\ ||f(\vec{r}) - f(\vec{r_0})|| > t \end{cases} \tag{3.16}$$

Where $\vec{r}$ represents the position of any pixel in the mask $and\ \vec{r_0}$ represents the position of the nucleus and *t* is a threshold on the difference of brightness. The performance of this detector is independent on any fine-tuning on the value of the parameter *t*.

The comparison values taken at all neighboring pixels are summed to form a decision represented by function *n*,

$$n(\vec{r_0}) = \sum_{\vec{r}} C(\vec{r},\vec{r_0}) \tag{3.17}$$

Actually *n* represents the number of neighboring pixels that have brightness similar to the nucleus's brightness.

The SUSAN can be used as an edge finder and the decision is taken by comparing *n* to another threshold *g*, named geometric threshold, set to be equal 3/4 the number of pixels in the mask. In addition, an edge response function *R* is formed,

$$R(\vec{r_0}) = \begin{cases} g - n(\vec{r_0}), & if\ n(\vec{r_0}) < g \\ 0, & otherwise \end{cases} \tag{3.18}$$

Note that R attains a maximum if n has a minimum value which leads to conclude that the nucleus corresponding to $\vec{r_0}$ is an edge point.

Instead of using the comparison function C shown in Eq. (3.16) that leads to sharp results (0 or 1), a more stable function can be used,

$$C(\vec{r}, \vec{r_0}) = e^{-\frac{(f(\vec{r})-f(\vec{r_0}))^6}{t}} \tag{3.19}$$

The SUSAN can be used also as a corner finder which is very similar to the SUSAN edge finder. The tuning of the parameters t and g is more important. The choice of a value of g affects the quality of the corners detected and also affects their number. For example, a low value of g will lead to sharper detected corners therefore a small number of them. However, this threshold value can be fixed in the algorithm and won't need any further tuning. In other hand, the value of the threshold t won't affect the quality of the outputs rather than their number since t reflects the allowed amount in change of brightness in the mask. If a value of t used is increased, the USAN areas will increase in the masks so the number of detected corners will decrease. Therefore, the value of this parameter can be used to control the quantity of the outputs rather than their quality.

The corner finder algorithm can be summarized as follows:

1. Place a circular mask at all the points in an input image.
2. Calculate the comparison value at every pixel in the mask using Eq. (3.19).
3. Calculate the number of those pixels that are very similar in brightness to the nucleus using Eq. (3.17).
4. Use Eq. (3.18) with smaller value of g than the value used in edge finder case to detect maxima that correspond to corners.
5. Eliminate the false positive detected corners by finding the center of gravity of the USAN area in the mask and measuring the distance between it and the nucleus. When the distance is very small, the detected corner is false reported and should be rejected. Next, this statement is clarified.

The center of gravity of the USAN region in the mask is shown as follows:

$$\vec{\bar{r}} = \frac{\sum_{\vec{r}} \vec{r}.C(\vec{r},\vec{r_0})}{\sum_{\vec{r}} C(\vec{r},\vec{r_0})} \tag{3.20}$$

Consider next Figure 3.6 that shows a simple portion of an image and three tested points (a), (b) and (c) in it. The first two points are on the edge and lie on one side or another of the edge, the third point (c) is also on the edge but lie in a region of brightness half a way between the 2 main regions that form the edge. USANs and their center of gravity of the three cases are shown to the right using a small 3 x 3 mask. Notice that for the case (c) the center of gravity and the nucleus are confounded. This case can report a corner presence since the number of pixels similar in brightness to it is small. So, this false detection can be corrected by taking the distance between the center of gravity and the nucleus. If it is large enough, the detected corner is a true corner else the detected corner is false and should be rejected.

Results of the SUSAN corner finder applied to a real image are shown in Figure 3.7.

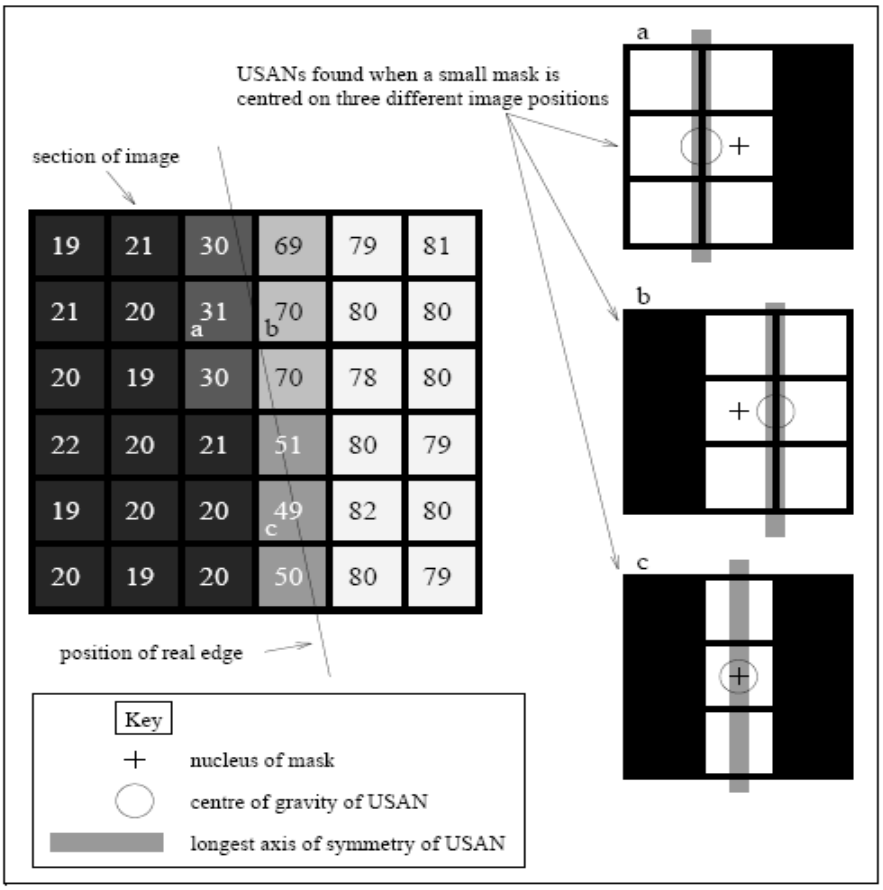


Fig.3.6. Center of gravities of USANs at different nucleuses in a portion of an image [103].

Fig.3.7. SUSAN corner finder applied to a video captured image with t = 25 [103].

This detector is very fast and can be well used in real time robotic applications. However, it is related to a threshold value used to classify an image pixel as a corner or an edge point. Therefore, its efficiency will decrease when it is applied to real images as it is shown in chapter 5.

3.1.7. Harris-Laplace Detector

This method [104, 105] represents first the image in scale space by convolving it with a Gaussian filter with variable scale *t*. It uses different functions based on the first and second derivatives of the smoothed image like laplacian or gradient or difference of Gaussian … and search for local extrema over image space and scale space. These extrema represent the interest points locations in space and scale. The characteristic scale at an interest point is the scale for which the extrema is detected over scale. This is a very important parameter in matching two images because it reveals the scale factor between two images. Consider two images one is a scaled version of the other by a scale *t*, if the characteristic scale of an interest point in the original image is *a* and the one of the corresponding interest point in the scaled image is b then $t \approx \frac{a}{b}$. Figure 3.8 explains this fact using two images; one is a scaled version of the other by a scale factor of 2.5. It is shown that the corresponding

characteristic scales 10.1 and 3.89 using a function F used to build the scale space have a ratio very close to 2.5.

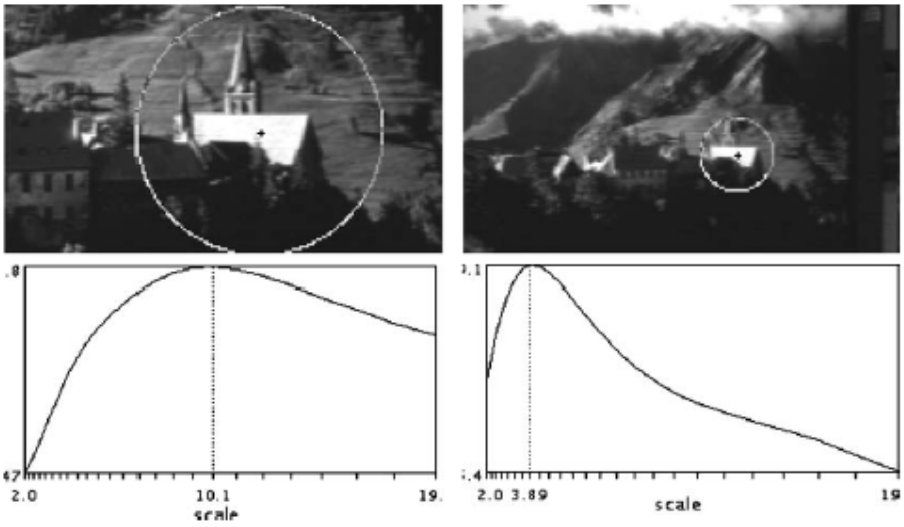


Fig.3.8. Characteristic scale in scale space [105].

Existing feature detecting discussed so far rely on one operator applied to the image f(x,y) in order to detect interest points. These points correspond to the extrema over image spatial and scale domains. This operator could be the Laplacian of Gaussian, Difference of Gaussian, gradient, Harris, etc. Harris-Laplacian technique uses the combination of two operators to detect these points: Harris and Laplacian of Gaussian operators.

Harris operator imposes itself one of the most powerful corner detectors.

Table 3.1: Experimental results of various interest points detectors [105].

	Laplacian	DOG	gradient	Harris
detected	46%	38%	30%	16%
correct/ detected	29%	28%	22%	23%
correct	13.3%	10.6%	6.6%	3.4%

Experimental results in Table 3.1 show the success of the Laplacian operator over other operators in detecting interest points taking as parameter the correctness of the detected in the same image over scales. Row 2 shows the percentage of detected points over scales with a characteristic scale. Row 3 shows the correct detected points; a point is detected correctly if the ratio between its characteristic scale at the current

scale s_i and its one at previous scale s_j is equal to ratio of these scales s_i and s_j as discussed previously. Row4 shows the total correct percentage. Therefore, the idea arises to use first Harris corner detector to find interest points over image domain that correspond to the maxima of the multi-scale Harris corner measure $M_c(x, y, t, s)$,

$$(\hat{x}, \hat{y}, t) = argmaxlocal_{(x,y)} M_c(x, y, t, s) \tag{3.21}$$

Then the scale normalized Laplacian operator $\nabla^2{}_{norm}$ is used to detect among these interest points those who correspond to maxima in scale domain. The characteristic scale is given by:

$$\hat{t} = argmaxmin_t \nabla^2{}_{norm}(\hat{x}, \hat{y}, t) \tag{3.22}$$

The descriptors are Gaussian derivatives which are computed at the characteristic scale using up to 4th order derivatives. Invariance to rotation is obtained by "steering" the derivatives in the direction of the gradient. To obtain a stable estimation of the gradient direction, the peak in a histogram of local gradient orientations is used. Invariance to the affine intensity changes is obtained by dividing the derivatives by the steered first derivative.

Matching interest points in two images is done by comparing their descriptors using Mahalanobis distance [106]. Figure 3.9 shows the corresponding interest points in two images one of them is a scaled image of the other by a scale of 1.92. Thus, the correspondence is found at corresponding smoothing scale *s*.

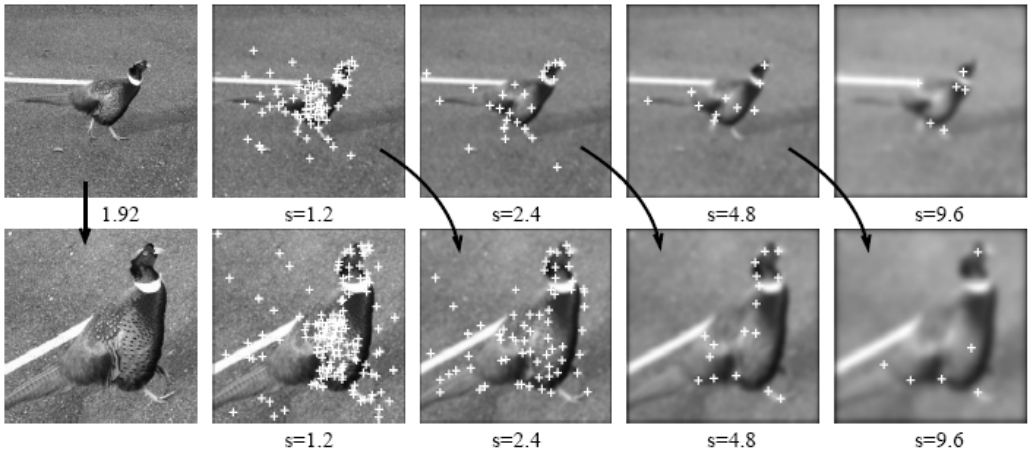


Fig.3.9. Points detected on different smoothing levels [105].

Figure 3.10 shows the repeatability of the Harris-Laplacian detector compared to other detectors.

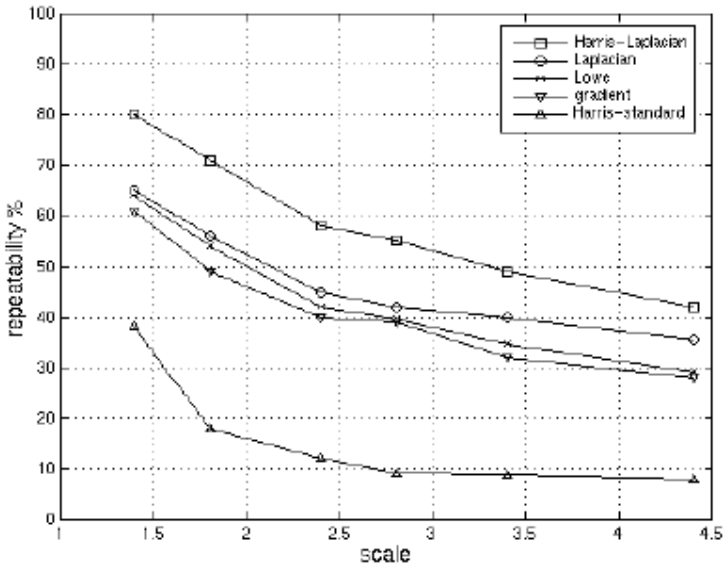


Fig.3.10. Repeatability over scale as performance evaluation.

Results provided here show the quality of this technique. The repeatability is very high not only over scale change but also with respect to change in viewpoint as shown in Figures 3.11 and 3.12. In Figure 3.11, there are 180 and 176 detected points detected in the left and right images. The number of initial matches is 23 and there are 14 inliers. In Figure 3.12, there are 34 inliers using scale and affine regions.

Fig.3.11 Points detected in two images with different viewpoint and scale change of 2.7 [105].

Fig.3.12. Points detected with their characteristic scales in two images with viewpoint change of 30° and scale change of 1.8 [106].

This detector introduces the scale space representation of an image which makes him very slow for robotic applications as shown in chapter 5.

3.1.8. Scale Invariant Feature Detector (SIFT)

The SIFT feature points, introduced by D. Lowe [82], and their descriptors are shown to be repeatable versus image scaling, rotation and partially invariant to change in illumination [82] and 3D camera viewpoint [83]. In addition, they are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion [63], clutter, or noise. Also, these features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

The output of the SIFT algorithm is taken by a cascade filtering approach. This cascade is constructed form four essential steps [63]: scale space ectrema detection, keypoint localization, orientation assignment and keypoint description.

SIFT searches for features that are in variant especially to scale. In order to achieve that, the image function is taken into various scales and the search is restricted on features that are stable. In scale space, the two dimensional image function is convolved with Gaussian smoothing filter with different smoothing level. Extrema of the image, that can be used as interest points, can be detected using the laplacian of Gaussian ($\nabla^2 G(x, y, t)$) operator [57, 58].

$$\nabla^2 G(x,y,t) = G_{xx}(x,y,t) + G_{yy}(x,y,t) = \left(\frac{r^2 - t}{t^2}\right) e^{\frac{-r^2}{t}} \; where \; r^2 = x^2 + y^2. \tag{3.23}$$

Where $G_{xx}(x,y,t)$ and $G_{yy}(x,y,t)$ are the second order partial derivative of the Gaussian filter $G(x,y,t)$.

However, Lowe has introduced the difference of Gaussian (DoG) operator as an approximation of the LoG operator as derived in Eqs. (3.24) and (3.25).

$$D(x,y,t) = L(x,y,kt) - L(x,y,t) \tag{3.24}$$

$$D(x,y,t) = f(x,y) * G(x,y,kt) - f(x,y) * G(x,y,t) \tag{3.25}$$

$$= f(x,y) * (G(x,y,kt) - G(x,y,t))$$

Where $L(x,y,t)$ is the smoothed image with level $\sqrt{t}$ and $D(x,y,t)$ is the difference of two smoothed images: one of them with level $\sqrt{kt}$ and the other with level $\sqrt{t}$.

The derivation of the DoG from LoG is extracted from the diffusion equation,

$$t\nabla^2 G = \frac{\delta G}{\delta t} \approx \frac{G(x,y,kt) - G(x,y,t)}{kt - t} \tag{3.26}$$

$$G(x,y,kt) - G(x,y,t) \approx t^2(k-1)\nabla^2 G \tag{3.27}$$

Since t^2 is the scale normalization required for the scale invariant laplacian and $(k-1)$ is a constant factor over all scales, the DoG is approximated to LoG. The practical calculation of DoG is shown in Figure 3.13. The scales are grouped per octave (each octave is a semi-open interval of scales from the starting scale to its double). Then, the difference between two successive smoothed images is taken.

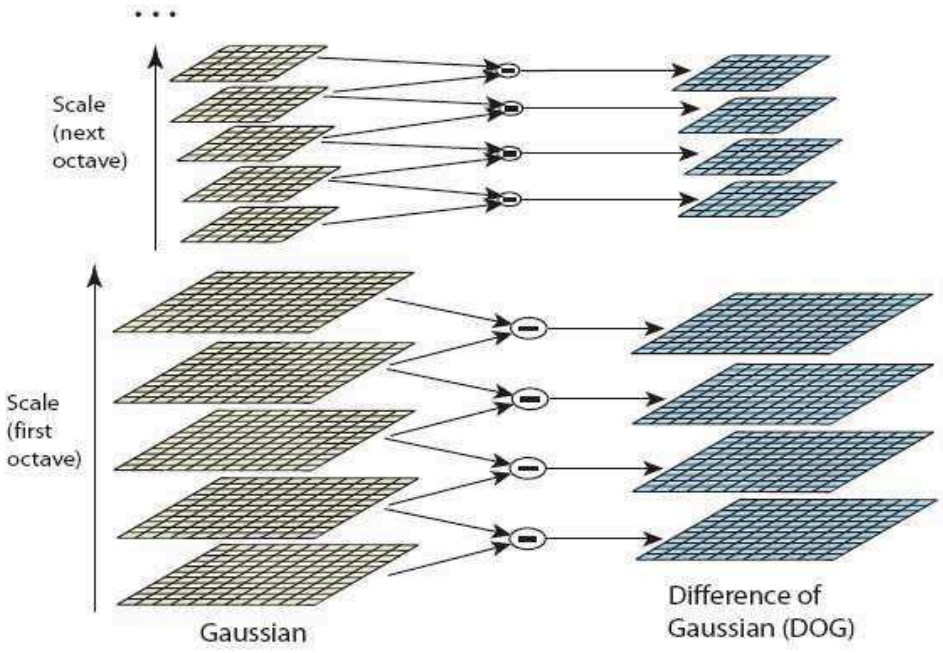


Fig.3.13. DoG calculation [63].

The goal of any feature detector is to select special features over scale. The SIFT features are the points in the image space that correspond to an extrema of the DOG over scales. Lowe detects an extremum by comparing current pixel intensity to its 8 neighbors in the same scale and then comparing it to the nine corresponding neighbors in the previous and next DOG image as illustrated in Figure 3.14.

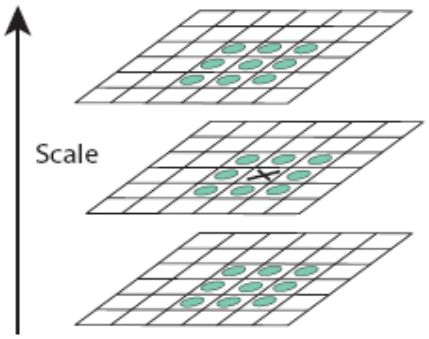


Fig.3.14. Detecting an extremum by comparing it to its 26 neighbors [63].

Two major problems appear for extrema detection. The first one is the sampling frequency in scale domain which mean the optimal number of scales selected per octave. It was shown experimentally [63] that three samples per octave is the optimal solution. The second one is the sampling frequency in the image domain. Here, the

first scaled image in the octave is pre-smoothed by a scale t=1.6. It is shown [63] that this smoothing level offers experimentally maximum repeatability of the points in the image after some transformations. Then the pre-smoothed image can be expanded in size using linear interpolation to make full use of the input image and to discard high frequency components relative usually to noise.

After detecting the DOG extrema, accurate localization is initiated. This is done by using taylor expansion up to quadtratic terms on the scale space DOG function D(x, y, t). In addition, this step tests the stability of the detected extrema. A point is judged as stable if it has a strong contrast in its region that means it will be more robust to noise than other points. Thus, many extrema that are unstable are removed and won't be considered as interest points. Also, edge extrema are discarded since they are also vulnerable to noise.

At every detected interest point in the smoothed image *L(x,y,t)*, the gradient vector is calculated at the scale level as follows:

$$|\nabla L(x,y)| = \sqrt{(L(x-1,y) - L(x+1,y))^2 + (L(x,y-1) - L(x,y+1))^2} \quad (3.28)$$

$$\varphi_{\nabla L}(x,y) = tan^{-1}\left(\frac{L(x,y-1) - L(x,y+1)}{L(x-1,y) - L(x+1,y)}\right) \quad (3.29)$$

A descriptor vector is formed at every interest point. It is highly distinctive and partially invariant to the variations such as illumination, 3D viewpoint, etc. In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. To form this vector, first a set of orientation histograms are created on 4x4 pixel neighborhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16 x 16 region around the keypoint such that each histogram contains samples from a 4 x 4 subregion of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. Since there are 4 x 4 = 16 histograms each with 8 bins, the vector has 128 elements. This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination. To reduce the effects of non-linear illumination, a threshold of 0.2 is applied and the vector is again normalized. Figure 3.15 gives a simple example on the calculation of the descriptor vector. Instead of 16 x 16 region, an 8 x 8

region is used. So, the descriptor will be 2 x 2 and its size will be 2x2x8=32 as shown in the right figure.

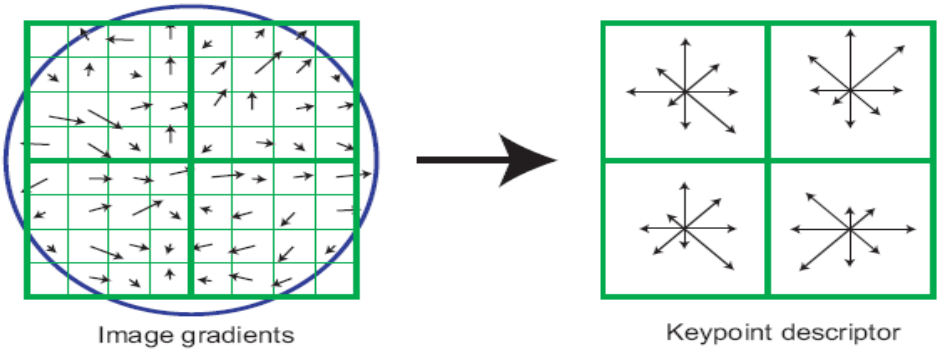


Fig.3.15. A simple 2 x 2 SIFT descriptor [63].

Figure 3.16 shows the extraction of SIFT points on four different object images. The two left columns images contains two different viewpoint of the object while the right column images show the extracted feature points.

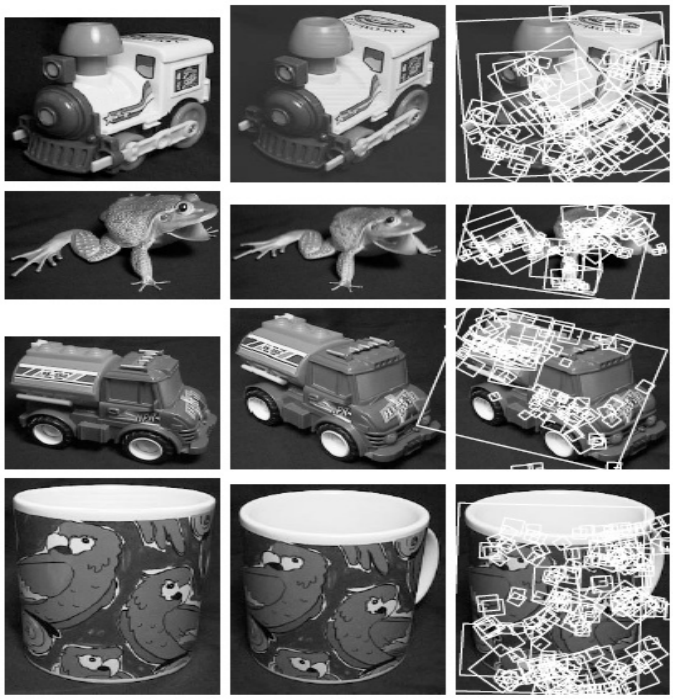


Fig.3.16. Kepoints detection [63].

A practical example for object recognition using SIFT operator is shown in Figure 3.17. The top image is the input image where the desired objects to detect are the four objects represented in Figure 3.16.

Fig.3.17. Object recognition using SIFT keypoints [63].

This SIFT points as presented are invariant over scale. However, they are not very robust to noise existing in real images as shown in chapter 5.

3.1.9. PCA-SIFT

This technique [84] is inspired by the SIFT operator and its main goal is to minimize the dimensionality of the SIFT keypoint features vector. It runs the first three steps of the SIFT on an input image (scale space extrema detection, keypoint localization and orientation assignment). Then it places a 41 x 41 patch centered at the keypoint to form the feature descriptor vector. The group of the high dimensional features vectors extracted from the patches of the input image form the input to the PCA (Principle Component Analysis [85]) algorithm. PCA outputs low dimensional

vectors approximately orthogonal that construct a basis to represent the high dimensional feature vectors. These feature vectors are projected onto the basis. Then their coordinated are extracted and used as descriptors for matching. PCA extracts this low dimensional basis by calculating the eigenvectors and eigenvalues of the covariance matrix constructed by the high dimensional vectors. The eigenvectors that represent 90%, for example, of the total energy are only selected to form the basis. Hence, the reduction in dimensionality is achieved sine the number of these eigenvectors is less than the number of initial features vectors. Thus, the time reduction of PCA-SIFT with respect to SIFT is due to the reduction in the size of the feature descriptor vector after projection onto the PCA basis. So, matching process time is reduced due to this reduction in dimensionality. Another important achievement in PCA-SIFT is that the eigenvectors are quasi orthogonal and this will force the features vectors (used by SIFT) to be more distinctive after projection onto the basis of eigenvectors (used by PCA-SIFT) and this fact will lead to a better performance in matching these vectors.

This detector presents a way to reduce the computation time of SIFT descriptors and also makes them more distinctive. However, the problem of the false detection in real images due to noise is still the same.

3.1.10. Speeded Up Robust Feature Detector (SURF)

The SURF operator [62, 87] is based on the Hessian matrix. It is partly inspired by the SIFT descriptor but it is several times faster and also, as claimed by the authors, more robust against different image transformations than SIFT.

Hessian matrix is a square matrix of the second order partial derivative of a smoothed image function *L(x,y,t)* [59],

$$H(x,y,t) = \begin{bmatrix} L_{xx}(x,y,t) & L_{xy}(x,y,t) \\ L_{yx}(x,y,t) & L_{yy}(x,y,t) \end{bmatrix} \tag{3.30}$$

It is used as a measure of curvature of L where its responses at specific points reflect the presence of local extrema or saddle points. In addition, the Determinant of Hessian operator "DoH" can be used as an affine covariant blob detector that responds to saddles at its maxima and minima [3]. The scale normalized determinant of hessian, referred also as Monge-Ampere operator [60], is given by:

$$detHL = t^2(L_{xx}L_{yy} - L^2{}_{xy}) \tag{3.31}$$

$$(\tilde{x}; \tilde{y}; \tilde{t}) = argmaxlocal_{(x,y,t)}(detHL(x, y, t)) \tag{3.32}$$

The points $(\tilde{x}; \tilde{y})$ are the feature points detected by the Determinant of Hessian (DoH). These points correspond to a high curvature of the image intensities and are nearly invariant to affine transformations [3]. In terms of scale selection, blobs defined from scale-space extrema of the determinant of the Hessian (DoH) also have slightly better scale selection properties under non-Euclidean affine transformations than the more commonly used Laplacian operator [3].

SURF relies on the determinant of the Hessian matrix but uses an approximation of its Laplacian partial derivatives since their Gaussian filters are not ideal in any case. Thus, second derivative Gaussian filters are approximated by box filters and are evaluated very fast using integral images. Also, the performance using these box filters is comparable to the methods that use cropped Gaussian filters.

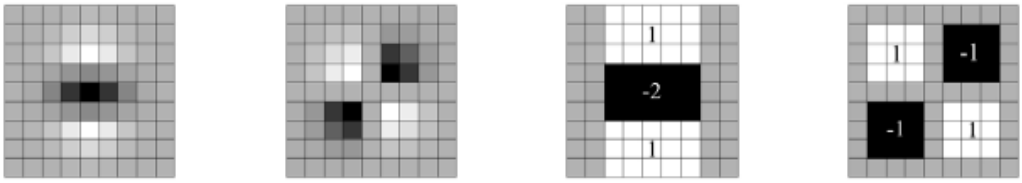


Fig.3.18. Partial second order derivatives of Gaussian filters and Box filters. First 2 images represent the Gaussian filters in the y and xy direction. The last 2 represent their approximated box filters [87].

Figure 3.18 represents the second derivative Gaussian filters L_{yy} and L_{xy} and their approximates 9 x 9 box filters D_{yy} and D_{xy}. As a result the determinant of Hessian matrix can be approximated as follows:

$$\det(H) = L_{xx}L_{yy} - L_{xy}{}^2 \approx D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{3.33}$$

The maxima of the determinant of Hessian reveal the locations of keypoints that are invariant in scale space. This process starts by smoothing the image using Gaussian kernel at different scale and when moving from a scale level to another the image is sub-sampled. In SURF, instead of sub-sampling the image, the box filters mask is up-scaled in size in the new scale level and the image size is unchanged. The following filter masks are used: 9 x 9, 15 x 15, 21 x 21 and 27 x 27. At larger scales, the change in the mask size should be doubled when going from an octave to another

in scale. So, the change in size rate will increase from 6 to 12 to 24 … Then the maxima locations are interpolated using the same method discussed in SIFT [89]. The need for this interpolation is due to the fact that the difference in scale between the first layers in every octave is relatively large.

To form the SURF descriptors, Haar-wavelets [90] responses are calculated in x and y direction in a circular neighborhood of radius 6*t* around the interest point where *t* the scale at which the interest point was detected. In fact, the size of the wavelet depends on the scale *t* and its length is 4*t*. Therefore for large scales the size of the wavelets filters is big. Only six operations are needed to compute the response in x or y direction at any scale. After calculating the wavelet response, the region centered at the interest point is weighted with a Gaussian (circle with a radius 2.5*t*) to reduce the effect of abrupt changes in pixel values. In this region, the responses at various pixels are represented by 2D vectors with horizontal and vertical components and these vectors are summed within a sliding orientation window covering an angle of 60^{o}. Dominant vector, which corresponds to the greatest magnitude, are the only one considered and the others are rejected. The size of the sliding window is a parameter derived experimentally. Figure 3.19 shows the SURF interest points on an image, Haar wavelets and the descriptors windows.

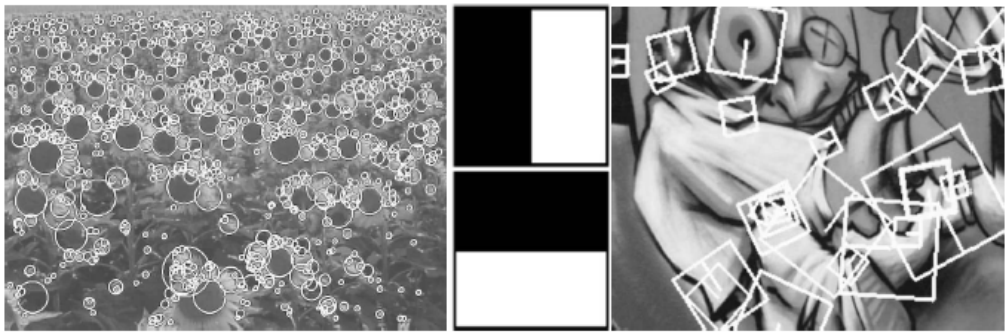


Fig.3.19. SURF Interest points. (a) represents the detected interest points-center of the surrounding circles with radius equal to the corresponding scale. (b) Haar wavelets. (c) Descriptor windows centered at the interest points and rotated according to the dominant vector in it [87].

The first step in the descriptor's construction is constructing a square region centered around the interest point, oriented along the dominant vector and having a size of 20*t*. This square window is divided into square sub-regions of 4 x 4 in size with 5 x 5 regularly spaced sample points inside. Then, the Haar wavelet responses

are calculated horizontally d_x and vertically d_y at each pixel taken with respect to the dominant vector orientation in the square window. The responses are summed in algebraic and absolute values $\sum d_x, \sum d_y, \sum |d_x|, \sum |d_x|$ in the corresponding locations in the 4 x 4 sub-regions to form a feature vector of 4x4x4=64 dimensions. The absolute values are taken into consideration to extract the information about the polarity of the intensity changes. In addition, to make the descriptor invariant to contrast, the vector is normalized to a unit length vector.

Compared to SIFT, SURF is much faster (approximately by 3 times) but it is not robust to illumination variation and viewpoint change as well as SIFT. Therefore, it is also not very robust to noise in real images.

3.1.11. Gradient Location Orientation Histogram (GLOH)

Gradient location-orientation histogram (GLOH) [86] is an extension of the SIFT descriptor. It extracts more detailed information about the SIFT descriptor to increase its robustness and distinctiveness. A log-polar histogram centered at the descriptor keypoint is formed and corresponding gradient vectors vote for their bins. It is composed from three bins in radial direction (the radius is set to 6, 11, and 15) and eight in angular direction. The gradient orientations are quantized in 16 bins. This gives a 272 bin histogram. The size of this descriptor is reduced with PCA. The covariance matrix for PCA is estimated and the 128 largest eigenvectors are used for description similar to PCA-SIFT.

3.2. Edge based corner detectors

In contrast to intensity based interest points, edge corners are edge points that correspond to a deviation in the edge direction. Firstly, the image is pre-segmented into contours: contours in the image are extracted and chain coded. Then, algorithms are developed to detect corners along these contours. According to the contour-based approach, corners are defined as the intersection points or junction points between straight edge segments. The chain codes can be used in corner detection [95, 96]. However, the main difficulty in contour based corners is the ability to extract reliable image segmentation in the edge detector.

Gradient-based detector [61, 98] is an example of edge corner detector. It relies on measuring the curvature of an edge that passes through a neighborhood. The

strength of the corner response depends on both the edge strength and the rate of change of edge direction.

3.2.1. The basis of edge corner detectors: edge detectors

Most edge detections are performed in four steps shown in Figure 3.20:

- Gradient computation, in norm (magnitude of the edge) and direction (at π / 2 of the edge direction) using classical operators [20-22,25,110-113].
- Threshold on gradient norm to extract initial edge points. Every image pixel having a gradient vector norm greater than a given threshold is classified as an edge pixel.
- Thinning the edge is the process of making the edges of thickness equal to one pixel width [17, 182, 183]: a pixel is classified as an edge point if its gradient norm is superior to the gradient norm of its two neighboring pixels in the direction of the gradient [11].
- Linking the edge points to form the image contours [17, 184]. After detecting all edge pixels, every edge pixel is linked to its neighboring one if it exists. If it does not exist, prolongation/closing starts and tries to link the edge pixel to its nearest one.

A lot of effort was done in [11] to gather the edge detection steps to reduce the number of image scans which are very time consuming. It is possible to gather the gradient vector computation step with the thresholding step because this last step requires only the knowledge of the current pixel. With slight modification to the thinning procedure, the thinning step can be also gathered with them. The new principle is the following: a pixel is kept as an edge point if its gradient norm is superior to the gradient norm of its past neighboring pixel in the direction of the gradient. The second gathering gathers the linking and closing steps. The principle is as follows: at an edge pixel, if an unlinked neighboring edge pixel exists than the linking procedure is initialized otherwise the closing step is initialized.

3.2.1.1. Gradient computation

Most edge detectors are based in some way on measuring the intensity gradient at a point in the image. The gradient operator [56] is characterized by its norm and its direction. At a tested pixel, the gradient norm gives the amount of intensity difference

between the current pixel and its direct neighboring ones. Therefore, this difference reflects the strength of the edge (greater difference implies stronger edge). In other hand, the gradient direction gives the direction of the greatest change which is normal to the edge direction as shown in Figure 3.21.

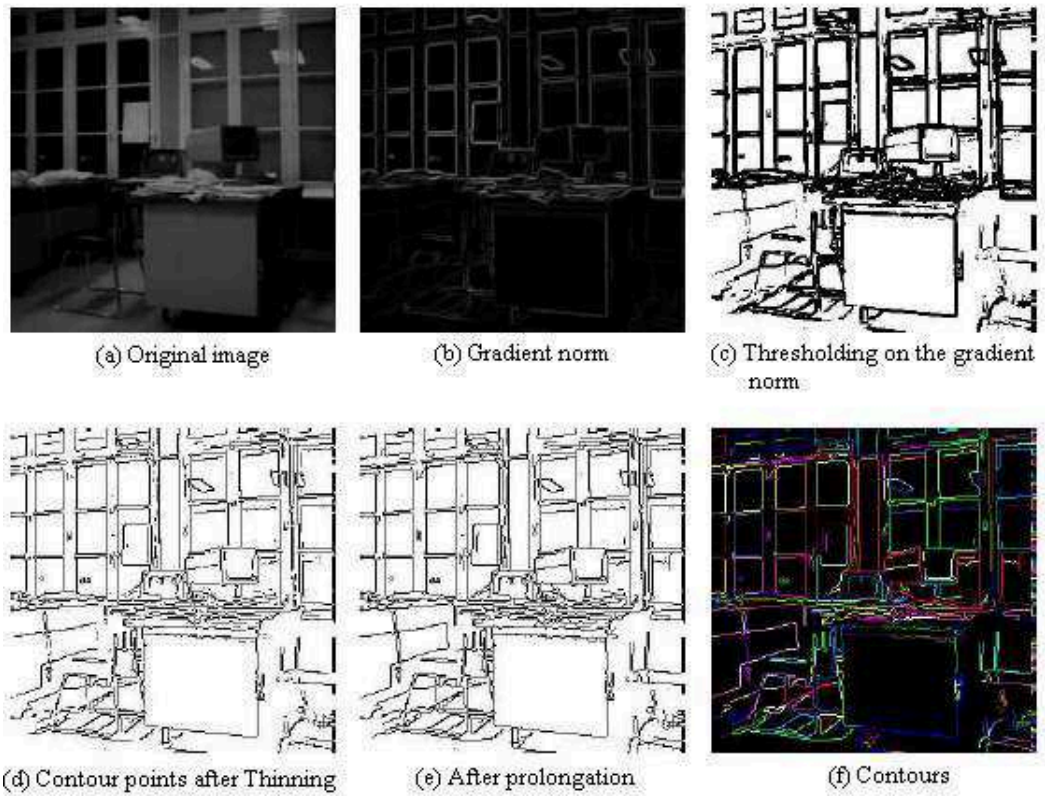


Fig.3.20. Edge detection steps [11].

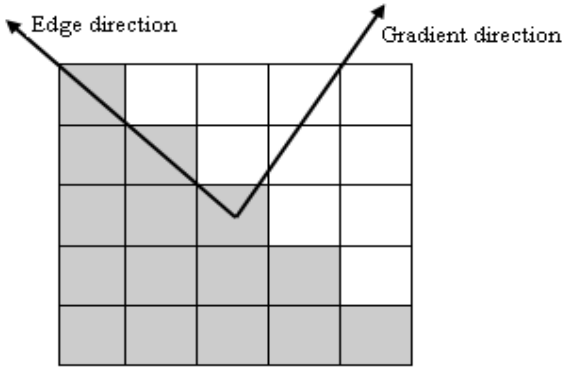


Fig.3.21. Gradient direction is normal to edge direction.

As an example on the gradient magnitude, Figure 3.22 shows the horizontal gradient, as the first derivative, peaks that correspond to the edge between two regions of intensities in the top row image. Figure 3.23 shows three edge images,

corresponding to three different values of *th*, of a face image. Figure 3.23 (b) corresponds to a low threshold, Figure 3.23 (c) corresponds to a medium threshold and Figure 3.23 (d) corresponds to a high threshold. Here, we should note that the choice of the threshold is very important. A very low threshold can introduce a lot of unwanted edges whereas a big threshold will keep only very strong edges so it can neglect some informative edges.

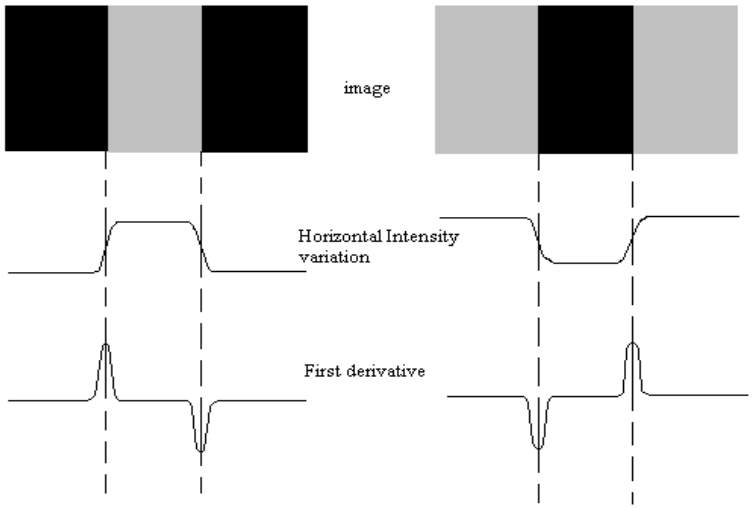


Fig.3.22. Peaks of the gradient norm corresponds to the edge.

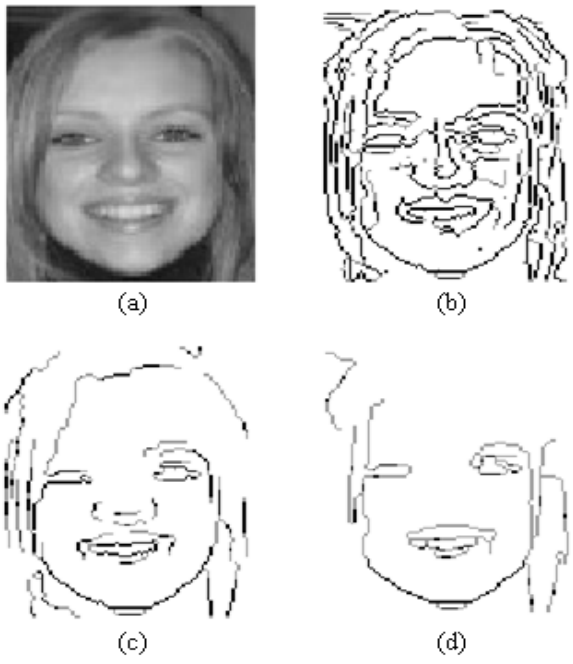


Fig.3.23. (a) Original image. (b), (c) and (d) Corresponding edge images.

Mathematically, the image can be seen as a function *f(x,y)* of two variables *x* and *y*, and the local gradient ∇ at every point in the image is given by,

$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.34}$$

The magnitude and the argument of this vector represent respectively the amplitude and orientation of the contour in the image,

$$\left|\overrightarrow{Grad}\big(f(x,y)\big)\right| = \sqrt{\frac{\partial f(x,y)}{\partial x}^2 + \frac{\partial f(x,y)}{\partial y}^2} \tag{3.35}$$

$$Arg\big(\overrightarrow{Grad}\big(f(x,y)\big)\big) = tan^{-1}\left(\frac{\frac{\partial f(x,y)}{\partial y}}{\frac{\partial f(x,y)}{\partial x}}\right)$$

In image processing, due to large computation time required for calculating the gradient magnitude and argument, the continuous gradient function is approximated by discrete masks. The search for edge points starts by placing this mask at every pixel and calculates the intensity difference in more than one direction. Every pixel having a gradient norm greater than a given threshold *th* is classified as an edge pixel.

Many operators have been suggested so far for gradient vector computation. Robert operator [110] was introduced first in 1965. It uses a 2x2 mask. The results are very sensitive to noise and not suitable for our case. Prewitt [112] (1970), Kirsch [18] (1971) and Sobel [20,111] (1978) operators have used 3x3 masks (filters with finite impulse responses). The results are correct and fast enough for robotic applications. Therefore, these operators are our target operators. Then we have Canny [113] (1986), Deriche [21, 195] (1987) and Shen-Castan [22] (1992) operators using recursive filters with infinite impulse responses. The results are much better than the previous ones but they require huge computation time relative to our suggested applications.

3.2.1.2. Comparative results

Each of the presented edge operators has its own characteristics. The selection of an operator depends on the desired application. Some applications like medical imaging requires excellent edge detection which is time consuming whereas other applications like mobile robot requires real time vision algorithms and don't require

perfect edge detection. Therefore, we can classify the presented edge detectors into two classes:

- Classical operators: Roberts, Kirsch, Prewitt and Sobel operators.
- Gaussian operators: Canny, Deriche and Shen-Castan.

For the classical operators desired in our application, the edge detection is simple, very fast and the edges are detected with their orientations. However, these operators are very sensitive to noise. For the Gaussian operators, they are more complex, more time consuming. However, they are more robust to noise and provide better, accurate, and well localized edges.

Figure 3.24 shows Gradient norm images of the Robert, Sobel and Prewitt operators.

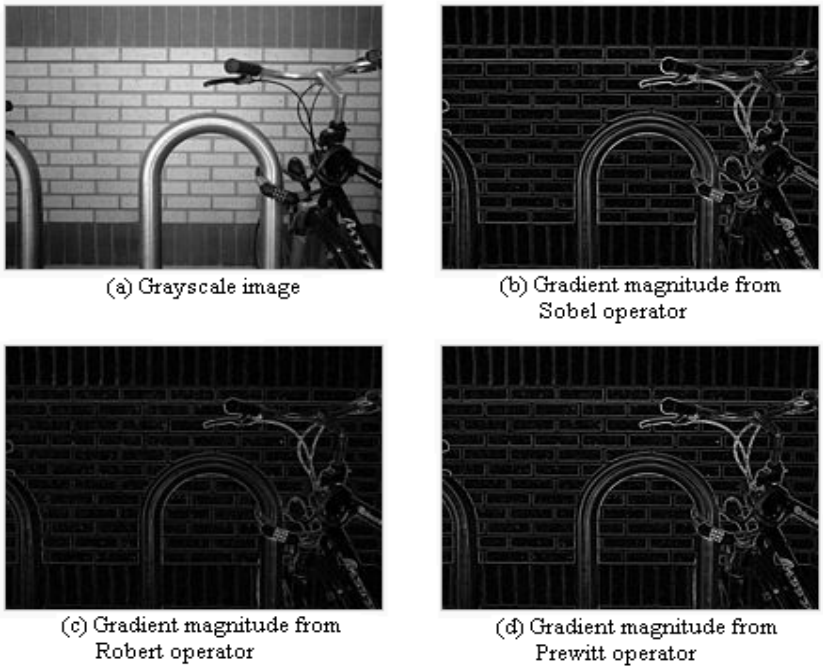


Fig.3.24. Gradient magnitude images of various operators.

Figure 3.25 shows the difference in accuracy in the presence of noise between the outputs of two classical operators (Robert and Sobel in Figures 3.25 (b) and (c) respectively) and the output of a Gaussian operator (Canny in Figure 3.25 (d)).

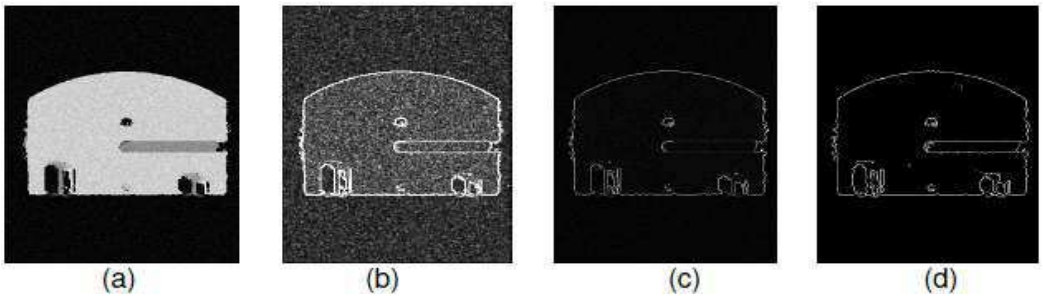


Fig.3.25. Edge detection on a noisy image. (a) original image with noise. (b) output of Sobel operator. (c) output of Robert operator. (d) output of Canny operator [175].

3.2.2. Edge segmentation: Polygonal Approximation

After edge detection, edge segmentation is a crucial step to detect edge corners. An edge corner can be defined as an intersection of two non collinear edge segments. In addition on a given contour, some edge corners that have greatest interest are selected to form the vertices of a polygon approximating the contour. This is known as polygonal approximation. Therefore, a survey of various polygonal approximation methods using corners (a survey can be found in [116]) is provided in this section.

The polygonal approximation problem is defined by Kolesnikov [24, 121-123] as follows: an open *N*-vertex polygonal curve P in 2-dimensional space is represented as the ordered set P of vertices. The output coarser curve Q consists of *M* vertices that from a subset of P and $M<N$. The end points of Q are the end points of P.

In Figure 3.26, Assume that we try to approximate a polygon's part starting at point $P_i(x_i,y_i)$ and ending at point $P_j(x_j,y_j)$ by a straight segment $[P_iP_j]$.

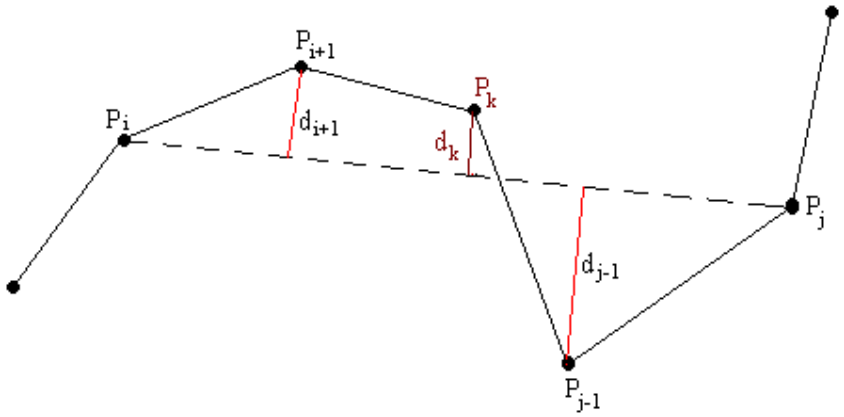


Fig.3.26. Approximating a polygon.

The distance d_k between a vertex P_k and the approximation segment can be expressed as follows:

$$d_k = \frac{\left|y_k - a_{ij}x_k - b_{ij}\right|}{\sqrt{1 + {a_{ij}}^2}} \tag{3.36}$$

Where the coefficient a_{ij} and b_{ij} are the coefficients of the straight line (P_iP_j):

$$a_{ij} = \frac{y_j - y_i}{x_j - x_i} \tag{3.37}$$

$$b_{ij} = y_i - ax_i$$

According to this distance, two approximation errors are introduced. The first one is the sum of all distances between the polygon vertices and the approximating line (P_iP_j):

$$e_1 = \sum_{k=i+1}^{j-1} d_k \tag{3.38}$$

The second one is the maximal distance between the vertices and the approximating line:

$$e_2 = \max(d_k)\,, \qquad i < k < j \tag{3.39}$$

After extracting the image corners based on straight edges of a shape, we have developed the "polygonal approximation" technique that consists of selecting a number of well chosen corners to construct the polygon that best approximates the contour of that shape. Polygonal approximation is widely used in object recognition [29] because it smoothes the contour of the objects in the images without loss of critical information.

Pavlidis in [117] presents an algorithm, known as split and merge technique. It divides an edge into a set of segments that has each one a maximal distance to the edge less than a given threshold d as shown in Figure 3.27.

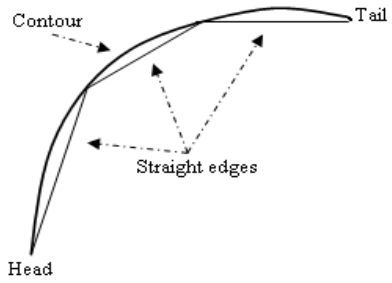


Fig.3.27. Polygonal Approximation.

The parameter used in this technique is the maximal distance d_{max} between the segment and the edge as described in Figure 3.28. If d_{max} is greater than the threshold d, the segment [AB] is divided into two segments; [AM] and [MB] where M is the edge point corresponding to d_{max}.

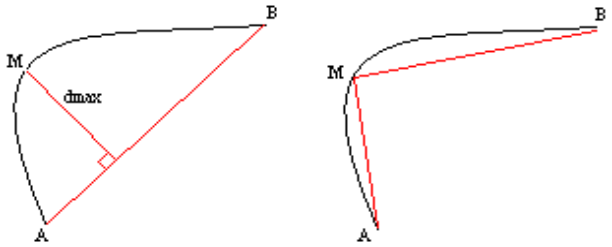


Fig.3.28. Segment division according to maximal distance.

Wall and Danielsson in [118] presented another algorithm that tries to find the segment [AB] shown in Figure 3.28 by moving its endpoint B starting from the edge point A along the edge until a certain criterion is no longer verified. The criterion used here is the maximal area per unit of length of the deviation between the edge and the corresponding approximated segment that should be less than a given threshold.

Pavilidis and Wall-Danielsson techniques use two different mathematical approximation errors to select the endpoints of the segments approximating the edge. However, our corner detector tests directly the edge to select the edge corners. The major improvement is in the good selection of these corners and especially in the detection of Complete Corners located sometimes outside the edge as shown in Chapter 4.

Another technique for obtaining a polygonal approximation of an object's contour based on an updated Hough Transform is presented in [119]. On the other hand, the

method in [120] is dedicated for the polyline representation of a scanned text or graphic objects.

Pinheiro in [124] searched for edge points, called curvature extremes, that correspond to a change in the direction of the curve (edge) using a curvature function which form the polygon vertices. These extremes are selected at different scale level of the smoothed image since when the scale level increases, the edge details become smoother and the number of extremes will be reduced.

Parverz and Mahmoud in [125] searched for cut-points on the contour of the studied shape. These points correspond to a deviation in the contour direction. Then the algorithm tries to minimize the number of detected cut-points until a terminating condition is satisfied. The final approximated polygon will have these cut points as extremes.

In [126], the authors presented a technique based on detecting dominant points on the contour and then iteratively suppressing the redundant ones in order to obtain the best approximated polygon with the minimal number of segments.

The algorithm of Masood in [127] also detects first dominant points called break points. Given the contour presented by linked edge points, every edge point presenting a deviation in the edge direction is classified as a break point as shown in Figure 3.29.

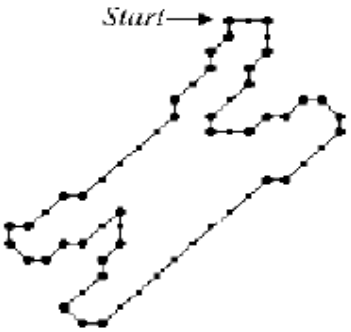


Fig.3.29. Masood break points.

The edge directions are coded using Freeman codes [96]. For every sequence of three break points $P_{k-1}(x_{k-1}, y_{k-1})$, $P_k(x_k, y_k)$ and $P_{k+1}(x_{k+1}, y_{k+1})$, draw the perpendicular line from P_k to the line $(P_{k-1}P_{k+1})$. This perpendicular squared distance shown in Eq

(3.40) and drawn in Figure 3.30, called Associated Error Value (*AEV*), is the measure used to reflect the strength of a break point.

$$\frac{((x_k - x_{k-1})(y_{k+1} - y_{k-1}) - (y_k - y_{k-1})(x_{k+1} - x_{k-1}))^2}{(x_{k-1} - x_{k+1})^2 + (y_{k-1} - y_{k+1})^2} \quad (3.40)$$

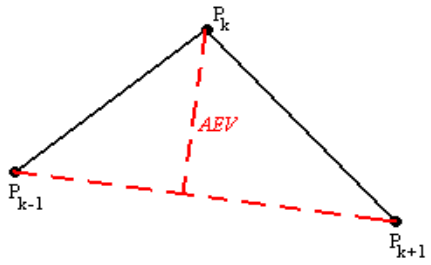


Fig. 3.30. *AEV* calculation at a vertex P_k.

The algorithm compares the *AEV* sequence and identifies the least *AEV*'s breakpoint to be deleted. This is an iteration process that eliminates some break points until a compression ratio or a maximal error is reached. The resulting Integral Square Error (*ISE*) is the sum of the remaining *AEV*s. In addition, after eliminating a dominant corner, a stabilization algorithm is initiated. It compares the current *ISE* to its predecessor and successor values and relocates the break point to obtain smaller *ISE*.

Finally, Marji and Siy in [128] presented a very similar technique to Masood's technique but it differs by the measure of error of each dominant point. Here, a dominant point is characterized especially by its strength that means its non collinearity with respect to its direct neighbors.

Masood and Marji-Siy methods are the two methods from the state of the art that are very close to our polygonal approximation technique based on DCs. The comparative results are shown in Chapter 5. Two essential points make our technique very competitive compared to existing ones. The first one is the reselection of already suppressed points (DCs). The second is the detection of Complete Corners outside the edges as explained in Chapter 4.

3.2.3. Existing edge corner detector: Corner detection using difference chain code as curvature

This corner detector [107] detects corners located on edge corners. By definition, an edge corner is an edge point where the curvature is high.

The corner detection algorithm is composed from several steps:

- Edge thinning: the authors have developed their own thinning algorithm applied on the image edges to obtain compact edges of thickness equal to one pixel. They have used mathematical morphology for this purpose.
- Coding the edge slope at every edge point: the chain code of an edge can have a value from 0, 1, 2, 3, 4, 5, 6, 7 in anti-clockwise direction (Freeman codes [96]), where 0 means moving one unit in x direction making angle 0^o with the x axis, code 1 represents 45^o from the x axis and so on. Therefore, the chain code thus codes the slope of the curve.
- Boundary smoothing: this step aligns the stray edge pixels along the dominant slope of the line and removes all spurious codes on that line. Figure 3.31 shows the result of the smoothing step.

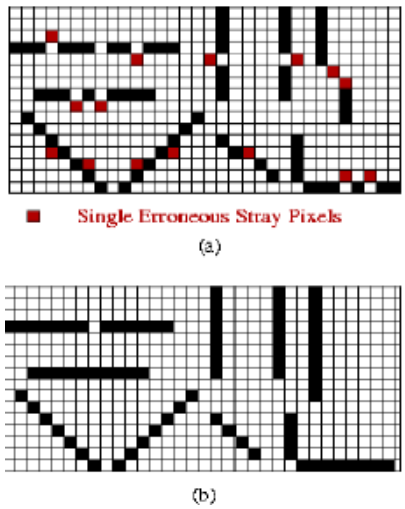


Fig.3.31. (a) Various erroneous stray pixels cases. (b) Results after smoothing [107].

- Avoiding false corners and detecting true corners: a corner is an edge point where a change in the edge direction occurs. Not all changes in the edge direction correspond to true corners. Thus, a true corner is an intersection of two lines of an appropriate length. If the length of one of these two lines is below a threshold, a corner is classified as a false corner as shown in Figure 3.32.

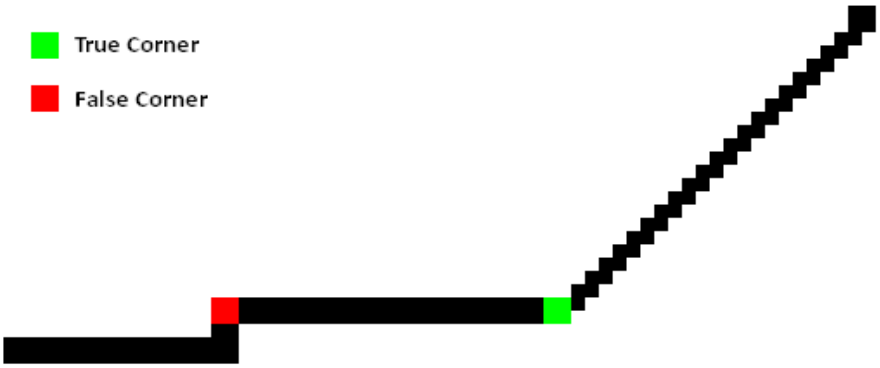


Fig. 3.32. True and False corners detection on an edge.

Figure 3.33 shows the comparative results with Harris detector [100] and He-Yang detector [109].

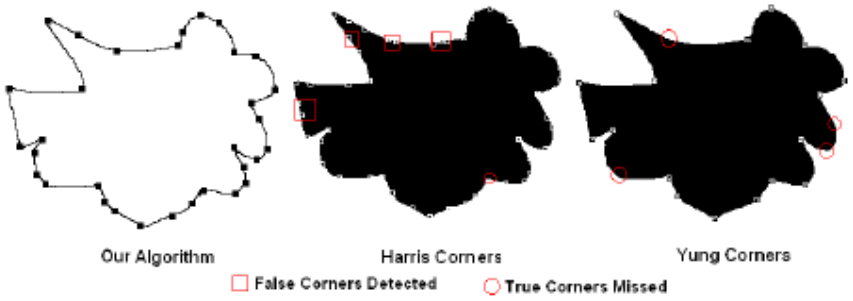


Fig.3.33. Test image with regular curvature change [107].

The detector is also tested on the same image in the presence of noise like Gaussian, Poisson, Speckle and Salt-Pepper. It is also compared to the same previous detectors as shown in Figure 3.34. It has a good performance in the presence of noise.

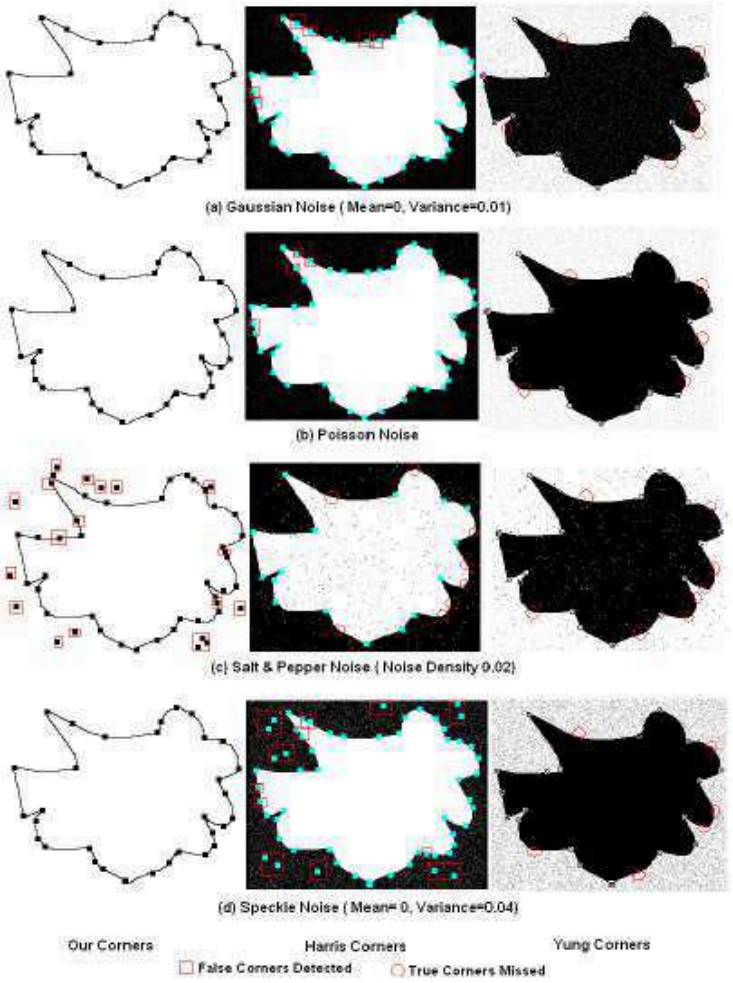


Fig.3.34. Corners extracted on noisy images [107].

It also has good immunity versus affine transformation. Figure 3.35 shows the detected corners on the original image and the transformed versions using affine transformations.

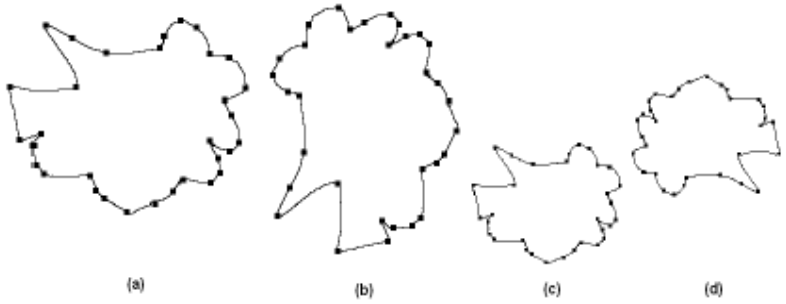


Fig.3.35. Transformation invariance of the corner detector. (a) Original (b) rotated 270^{o} (c) scaled 50% (d) scaled 50% and rotated 90^{o}.

This detector is edge based as our detector. It has good results when applied to synthetic images. However in robotic applications where we have real images, the proposed corner detector may fail especially when we have noisy edge pixels. We have proposed in our detector a solution to eliminate these noisy pixels as explained in Chapter 4.

3.3. First application: Image Registration

3.3.1. Introduction

Image registration [38] is one of the fundamental tasks in image processing. In recent years, many image registration approaches have been developed, leading to a great evolution in this domain [38]. Image registration is frequently used in remote sensing [30] for a wide variety of tasks such as change detection, image fusion, and image overlay. It is also used in image matching [33, 133], stereovision [34], image mosaicking and animation [36, 35], motion analysis [15], motion compensation to compensate the global motion of a camera in order to track the local motion of small targets in the acquired images sequence. Finally one of the most developed image registration applications are on medical imaging [37]. Most of the image registration techniques aim is to detect specific points, called control points (CPs), in the source and sensed images. These CPs are then used to estimate the transformation model that aligns the two images. Some of traditional techniques required the manual selection of the CPs at significant landmarks of the images. The primary drawback to this approach is that a trained expert is needed to manually select each individual CP in the remotely sensed images. This is very time consuming, especially when dealing with the large volumes of remote sensing data available today. Therefore, an automatic method of aligning such images is highly desired. Thus, the development of an intelligent algorithm that can automatically explore the CPs and then match them is very important.

Image registration is the process of transforming different images spaces into one coordinate system. These images may be taken by different cameras or by the same cameras but at different times or from different viewpoints. Usually the inputs of a registration process are two images, one called source image and the other called sensed image, related by a real deformation model. In fact, the whole process starts by detecting image features in both images. These features should have some invariant

measures under the estimated deformation. Then, features in the source image are matched with those of the sensed image using their invariance measures. Thus, the real deformation model is estimated by a mathematical transformation model determined by the corresponding features in both images. By applying the obtained transformation model to the sensed image (image resampling), the two images can be compared, aligned or analyzed.

Image registration techniques can be classified as follows:

- Spatial domain techniques. They are applied directly on image intensities or on image features like edges [15,39], contours [40, 41], regions [42], interest points [43] and lines [33].
- Frequency domain techniques. A common frequency-domain technique is phase correlation [31], which is based on the Fourier Shift Theorem. The Fourier coefficients of first image are divided by the Fourier coefficients of the second image, and the inverse of the result is an image with a single peak. This peak indicates the translation between the two images. This technique has also been extended to account for rotation and scaling [32].

In real experiments, the deformations produced in images of a real scene in motion or acquired by a moving camera are better estimated using a projective (elastic) or affine transformations rather than similarity or Euclidian transformations [44]. On the other hand, when the time interval between successively acquired images is large, the projective transformation describes better the deformation occurring in the image than the affinity. However, if the time interval is relatively small, affine model is a very good approximation of the real deformation [44]. In fact, the main difference between affine and projective transformations is that the deformation produced by projectivity for a given line does not depend only on its orientation, like in the case of affinity, but also on its position relative to the camera. Thus, if the time interval between two successive scenes is relatively small, the position of the elements in a scene will remain nearly the same and thus the transformation can be modeled by an affinity.

The majority of these registration techniques can be decomposed into four steps.

- Feature detection: Salient and distinctive objects (closed-boundary regions, edges, contours, corners, etc) in both source and sensed images are detected. An overview on feature detectors has been presented in Sections 3.1 and 3.2. These features can be represented by their point representatives called control points (CPs). However Section 3.3.2 presents various primitive construction schemes, based on grouping on or more CPs, in the literature.
- Feature matching and classification: The correspondence between the features in both images is established. Various matching approaches are proposed. Some feature matching methods use feature descriptors to match features in the studied source and sensed images. Other matching methods use the spatial information of the features as keys for matching. The cross correlation method could be the famous method used to match features or their descriptors. The matching methods are classified in section 3.3.3.
- Model estimation: the image transformation parameters, that align the two images, are estimated. Theses parameters are computed using different techniques like Hough transform, RANSAC, least square estimation technique, etc as explained in section 3.3.4.
- Image resampling and transformation: The sensed image is transformed by means of the estimated model. An overview of various geometric transformation models is provided in section 3.3.5

The new image registration technique presented in this thesis is targeting mainly for motion analysis where the deformation between the source image and the target one can be well modeled by an affine transformation. The reason is that the detected DCs have shown very good repeatability under affine transformations as shown in Chapter 6. Thus, the suggested robotic application is surveillance of a road by a drone's camera. The drone's camera is in motion and the acquired images have a small time interval. The goal is to estimate the global motion of the camera in order to determine the local motion of small targets, e.g. cars, moving on the road. This application is to be build as a future work.

3.3.2. Primitive construction: From features to primitives

Primitives can be formed by one feature, for example an interest point like Harris corner [136, 137], or by a group of features like grouping intersecting level lines to

form a particular shape [135]. The way of grouping features into primitives depends on the estimated transformation relating the two images to register. For example, a similarity transformation preserves angles so three non collinear corner points can be grouped into one primitive in a triangular form. The three obtained angles are three primitive invariant measures in this case. An invariant measure is a quantity that remains constant for all viewpoints of the scene. According to the registration application constraints, the real deformation between two acquired images can be modeled by a similarity, affinity or projective models. For example, when acquiring images by a moving camera with small acquisition time interval, the deformation can be well modeled by an affine model [44]. Each kind of these transformations has its own invariant measures. For example, the ratio of areas is an invariant measure under an affine transformation but it is variable under a projective transformation.

Our primitive is formed by four consecutive DCs located on the same contour. The DCs are more stable than Harris corners versus affine deformations as shown in Chapter 6.

3.3.3. Classification and feature matching

The classification of registration methods based on Maintz et al [176] survey is designed for medical image registrations. They have introduced nine basic criteria each of which is again subdivided into one or two levels. Wyawahare et al [178] and Chapnich et al [177] have cited various approaches to image registration. In addition, Zitova et al [38] have presented a survey on image registration methods where they have classified the image matching techniques into two categories: Area based and Feature based.

This classification is suitable also for our robotic image registration application.

3.3.3.1. Classification of registration methods

Registration methods are classified in several classes. The targeted application, its constraints, the quality of registration and the time interval between the registered images play the essential role in the selection of a class. In each class, the application constraints are specified.

3.3.3.1.1. Dimensionality

We should distinguish between "spatial registration methods" where all the dimensions are spatial dimensions and "registration of time series methods" where the time dimension is added.

For spatial registration methods, they are categorized by the number of dimensions used. It could be 2D-2D, 2D-3D or 3D-3D. Note that the method computation time is related to the number of dimensions used. They can be addressed in many registration applications that can be achieved by off line registration. In these applications, speed issue is not very important rather than the quality of the registration obtained.

For registration of time series methods, more than two images with spatial dimensions taken at short or long time intervals are studied for several reasons, such as monitoring the bone growth in children (long time interval) or monitoring of healing (short time interval), or evaluation of drug effects (various time interval), etc.

3.3.3.1.2. Source of features

The source of features is divided into three classes: Extrinsic, Intrinsic and Non-image based.

Extrinsic methods rely on artificial objects attached to the patient for medical applications, objects which are designed to be well visible and accurately detectable. As such, the registration of the acquired images is comparatively easy, fast, can usually be automated, and, since the registration parameters can often be computed explicitly, has no need for complex optimization algorithms.

Intrinsic methods rely only on patient generated image content so they don't rely on artificial objects attached to the body. In these methods, salient points or landmarks are identified on the patient body and used in the registration process.

Non-image based methods are used to register multimodal images if the imaging coordinate systems of the two scanners involved are somehow calibrated to each other and also the patient remains motionless during images acquisition. Therefore, in these methods there is no need for external attached objects to body neither for landmarks to be detected on the body since the two studied images are on the same coordinate system.

3.3.3.1.3. Nature of transformation

Various mathematical transformations [44] are introduced to model the real deformation between the studied images. A transformation is called rigid, when only translations and rotations are allowed. If scaling factor is also allowed, it becomes affine transformation and maps parallel lines onto parallel lines. If it does not conserve collinearity but still maps lines onto lines, it is called projective. Finally, if it maps lines onto curves, it is called curved or elastic. Section 3.3.5 provides a detailed explanation of some of these transformations.

3.3.3.1.4. Domain of transformation

A transformation is called "Global" if it applies to the entire image space or in other terms all image parts can be modeled by the same transformation. It is called "Local" if is applied to subsections of the image since each has its own transformation model.

3.3.3.1.5. Interaction

Many users prefer fully automatic methods (no human interaction). However, the argument is that many current methods have a trade off between minimal interaction and speed, accuracy or robustness.

The interaction level of registration methods can de divided into three levels referring to the control exerted by a human operator over the registration algorithm. It is called "Interactive" when the user does the registration himself, assisted by software supplying a visual or numerical impression of the current transformation, and possibly an initial transformation guess. Semiautomatic when the user interacts only to initialize the algorithm. For example, segmentation based methods are semi automatic intrinsic methods that need user initialization. Automatic when the user only supplies the algorithm with the image data and possibly information on the image acquisition. Extrinsic methods are usually easily automated since the marker objects are designed to be well visible and detectable in the studied images.

3.3.3.1.6. Method of parameter determination

The transformation's parameters can be calculated directly for the available data or calculated by searching for the optimal solution (set of parameters) of some functions defined on the parameter space. This class is detailed in section 3.3.4.

3.3.3.1.7. Modalities involved

Registration methods are grouped into four classes based on the involved modalities: "Monomdal", "Multimodal", "Modality to model" and "Patient to modality".

In Monomodal applications, the images to be registered belong to the same modality. For example, two images of the same modality are taken of the patient under two different positions for diagnostic purposes.

As opposite to Monomodal class, in Multimodal registration tasks the images to be registered are taken from two different modalities.

In modality-to-model, only one image is involved and the other is a model. The task here is to register an image referred to a mathematical model. For example, the registration of an MR brain image to a mathematically defined model of gross brain structures.

In patient-to-modality, only one image is also involved but the other is the patient himself. For example, in some treatments, the patient can be positioned with the aid of an X ray simulator to a pre treatment image. Thus, the registration task is performed using only the acquired patient images so it is classified as patient to modality.

3.3.3.1.8. Object

Registration methods are also classified based on the particular region of anatomy to be registered: head, thorax, abdomen, etc.

3.3.3.1.9. Subject of registration

There classes of subjects are introduced: Intrasubject, Intersubject and Atlas.

When all the studied images are acquired for a single patient, the registration is called Intrasubject. If the two studied images do not belong to the same patient but to different patients, e.g. patient and model, the registration is called Intersubject. If one of the images belongs to a single patient and the other is constructed from an image information database obtained using imaging, the registration is called Atlas.

3.3.3.2. Feature matching methods

Feature matching [38] is divided into two classes: Area based and Feature based. Our matching strategy can be classified as Feature based method. The template matching in section 3.3.3.2.2.8) and Matching using level lines primitives in section

3.3.3.2.2.7 are the two matching schemes that can be compared to our method as shown in Chapter 6.

3.3.3.2.1. Area based methods

There are two different approaches. The first one is the correlation like methods. The normalized cross correlation [185] function is one of the most important methods to measure the similarity between windows pairs from the source and sensed images. It is given by,

$$CC_W = \frac{\sum_W (I_{(x,y)} - \bar{I})(J_{(x,y)} - \bar{J})}{\sqrt{\sum_W (I_{(x,y)} - \bar{I})^2} \sqrt{\sum_W (J_{(x,y)} - \bar{J})^2}} \tag{3.41}$$

Where W is the window placed in the two images, I(x,y) and J(x,y) are the intensities of the pixel (x,y) in the source and sensed images and $\bar{I}, \bar{J}$ are their respective means in W. The windows pairs for which the maximum cross correlation is achieved are set as corresponding pairs. The cross correlation can be used to align two translated images only or when slight rotation or scaling are also present. Another method [186] similar to cross correlation and uses simpler distance measure is the sequential similarity detection algorithm. It accumulates the sum of squared differences of the intensities of the pixels lying into the windows pairs. If the obtained value is below a preset threshold than the two windows pairs are set as corresponding.

The second approach in area based method is the Frequency methods. They are preferred rather than the correlation in some cases especially for computational speed. The studied images are first represented in frequency domain. Then the phase correlation method, based on Fourier Shift Theorem [187], is applied to register translated with slight scaling images. Peak in the phase correlation function reveals the transformation relating the two images.

3.3.3.2.2. Feature based methods

Various methods have been introduced so far trying to give a similarity measure between image features in the two studied images using their spatial relations or using some feature descriptors.

3.3.3.2.2.1. Graph matching

Graph matching [188] is one of the feature based matching algorithms that gathers global information on the whole scene. The similarity measure used is the

number of feature points in the sensed image that, after a particular transformation, are mapped within a given range next to feature points in the source image. The transformation parameters corresponding to the highest similarity measure are selected to form the transformation relating the two images.

3.3.3.2.2.2. Clustering

Clustering technique [189] is another feature based method. For every pair of feature points from both images, the parameters of the transformation that maps the one onto the other is calculated and a point is added in the space of the transformation parameters. In this space, the points of parameters that closely map the highest number of feature points tend to form a cluster while mismatches fill the parameter space randomly. For the cluster of the highest number of points, its centroid represents the set of parameters of the desired transformation.

3.3.3.2.2.3. Chamfer matching

Chamfer matching [190, 191] is a technique for finding the best fit of edge points (or any other image feature) from the two studied images, by minimizing a generalized distance between them. The chamfer distance between two shapes can be efficiently computed using a distance transform (DT). This transformation takes a binary feature image as input, and assigns to each pixel in the image the distance to its nearest feature. The distance between a template and an edge map can then be computed as the mean of the DT values at the template point coordinates. The matching can be made more robust by using the mean of the thresholded distance,

$$d_{chamfer,th} = \frac{1}{n}\sum_{u_i \in I} \max\left(\min_{v_j \in J}\left(|u_i - v_j|\right), th\right) \tag{3.42}$$

Where u_i is a feature point in the source image I, v_j is its closest feature point in the sensed image J and *th* is a preset threshold to reduce the effect of outliers.

3.3.3.2.2.4. Cross-Ratio projective invariant measure

Suk and Flusser [132] have developed invariant features under projective transformation which is the cross-ratio of five points. Their features are point based for recognition of projectively deformed polygon. They have used this cross ratio as a description of five feature points to match candidates in the two studied images.

In general, when the distance between the camera and the object is not much greater than the size of the object, the introduced distortion is well approximated by a projective model [132]. Therefore, projective invariant features are needed to be developed.

Consider three non collinear points (x_i,y_i), (x_j,y_j) and (x_k,y_k) and their transformed points (x_i',y_i'), (x_j',y_j') and (x_k',y_k'). Their projective coordinates are $(x_i,y_i,1)$, $(x_j,y_j,1)$, $(x_k,y_k,1)$ and $(x'_i,y'_i,1)$, $(x'_j,y'_j,1)$ and $(x'_k,y'_k,1)$ related by a projective transformation given by the matrix $\boldsymbol{H}$:

$$H = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & 1 \end{bmatrix} \tag{3.43}$$

The relation between the coordinates of a point (x,y) and those of its transform (x',y') can be described by,

$$x' = \frac{a_0 x + a_1 y + a_2}{c_0 x + c_1 y + 1} \tag{3.44}$$

$$y' = \frac{b_0 x + b_1 y + b_2}{c_0 x + c_1 y + 1} \tag{3.45}$$

The areas A and A' [132] of the triangles whose vertices are (x_i,y_i), (x_j,y_j), (x_k,y_k) and (x_i',y_i'), (x_j',y_j'), (x_k',y_k') respectively are given by,

$$A = \frac{1}{2}\begin{vmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{vmatrix}, A' = \frac{1}{2}\begin{vmatrix} x_i' & y_i' & 1 \\ x_j' & y_j' & 1 \\ x_k' & y_k' & 1 \end{vmatrix} \tag{3.46}$$

Combining Eqs (3.4), (3.5) and (3.6) give the relation between A and A',

$$A' = A\sqrt[3]{J(x_i,y_i)J(x_j,y_j)J(x_k,y_k)} \tag{3.47}$$

Where J(x,y) is the Jacobian of the projective transform at point (x,y),

$$J = \frac{\begin{vmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & 1 \end{vmatrix}}{(1 + c_0 x + c_1 y)^3} \tag{3.48}$$

The relation given in Eq. (3.49) is so called relative invariant projective transform and it is utilized to derive the absolute projective invariant which is the cross-ratio of five points,

$$\rho = \frac{(1,2,3)(1,4,5)}{(1,2,4)(1,3,5)} \tag{3.49}$$

3.3.3.2.2.5 Matching using a contour based method

Li, Manjunath and Mitra [40] have proposed an image registration technique based on region boundaries and strong edges as primitives. They have detected open and closed contours and have used different matching criterion for each class.

The contour extraction forms the first step in their operator. For this purpose, they have used a Laplacian-of-Gaussian (LoG) operator and the edges are located at zero crossing points. At every edge point (x,y), a strength measure is introduced. It is proportional to the slopes of the LoG along the x and y directions. Then a contour is retained if:

- The edge strength at each point along it is greater than T_1.
- At least one point on the contour has strength greater than T_2.

Where T_1 and T_2 are two preset thresholds ($T_1 < T_2$).

Each contour is coded using the chain codes of its edge points using Freeman codes [96]. The primitive matching or primitive/primitive correspondence is based on a correlation measure between every two contours A and B: A from the source image and B from target image. This correlation function is based on the chain codes of the contours, reflects the similarity between them and the shift that must be introduced to one of them in order to best fit the other. The contour matching process begins with the matching of closed contours. For every closed contour, five shape attributes are computed: the perimeter, the longest and shortest distances from boundary to the centroid, and the first and second invariant moments. These moments were defined originally for 2D images. For the case of 2D contours, the first and second moments can be defined as:

$$h_1 = \frac{1}{n^2}\sum_{i=1}^{n}[(x_i - x_c)^2 + (y_i - y_c)^2] \tag{3.50}$$

$$h_2 = \frac{1}{n^4}\sum_{i=1}^{n}[(x_i - x_c)^2 - (y_i - y_c)^2]^2 + \frac{4}{n^4}\sum_{i=1}^{n}[(x_i - x_c)(y_i - y_c)]^2 \tag{3.51}$$

Where x_i and y_i are the coordinates of each point along the contour. x_c and y_c are the coordinates of the centroids of the contour. n is the length of the contour.

Two closed contours, one from each image, are accepted as candidate matches if the differences between their five attributes are below some preset thresholds. Then two candidate contours are matched if they have the same chain codes taking into consideration the possible shift in one of the.

For open contours, corner points that correspond to a deviation in the contour chain code are detected first. The contour segments surrounding the corner points are then used as 1D template in finding the corresponding matches in the two images.

The matched contours enter in the estimation of the transformation model. The transformation is assumed to be an affine transformation. For two matched edge points, (x,y) and (x',y'), on two matched contours in the two images, the 2D affine relationship can be expressed as follows:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = s \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix} \tag{3.52}$$

Where s represents the scaling level, θ represents the rotation angle and tx, ty represent the translation along the two orthogonal directions respectively. These four transformation unknowns are found using least squares sense based on all matched points.

3.3.3.2.2.6. Affine invariants in convex hulls

Yang and Cohen [134] have proposed a registration method for scene recognition under affine distortion. Their affine invariants are the areas of triangles whose vertices are three vertices among four consecutive ones of a convex hull.

For a set of feature points in the plane, the convex hull is the smallest convex object containing all the points. These feature points can be corner points, inflexion points, fiducial or marking points etc. the authors have used the algorithm developed by Bykat [181] to find the convex hull of a set of points. The convex hull bounds the set of these points from the outside, as illustrated in Figure 3.36. Thus, the convex hull is suitable in a shape representation or shape matching applications.

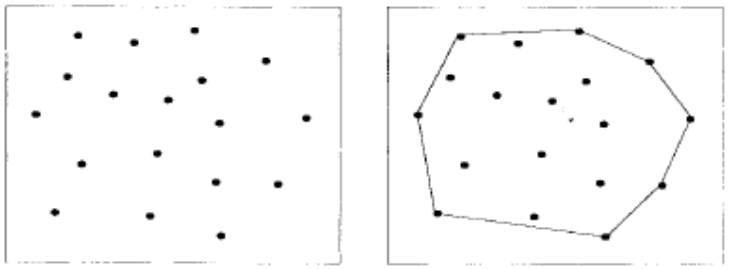


Fig.3.36. Convex hull of a scatter of feature data [134].

Barbar et al. [192] have explained the Bykat convex hull algorithm, called "QuickHull". It can be broken down to the following steps:

1. Find the two points with minimum and maximum abscissas; those are bound to be part of the convex hull.
2. Use the line formed by these two points to divide the set in two subsets of points, which will be processed recursively.
3. Determine the point, on one side of the line, with the maximum distance from the line. The two points found before along with this one form a triangle.
4. The points lying inside of that triangle cannot be part of the convex hull and can therefore be ignored in the next steps.
5. Repeat the previous two steps on the two lines formed by the triangle (not the initial line).
6. Keep on doing so on until no more points are left, the recursion has come to an end and the points selected constitute the convex hull.

In a test scene image, which has undergone an affine distortion or occlusion, a part of the convex hull may change or the number of its vertices may increase, decrease or remain unchanged. In Figure 3.37, three images are taken for the same scene where some objects are added or disappeared. The corresponding convex hulls are drawn in Figure 3.38. It is clear that the vertices 7 and 8 in Figure 3.38 (b) have no corresponding in Figure 3.38 (a). Also, vertex 8 in Figure 3.38 (c) corresponds to vertex 9 in Figure 3.38 (b). For this reason, the authors have suggested first an algorithm to detect corresponding vertices in two convex hulls related by an affine transformation.

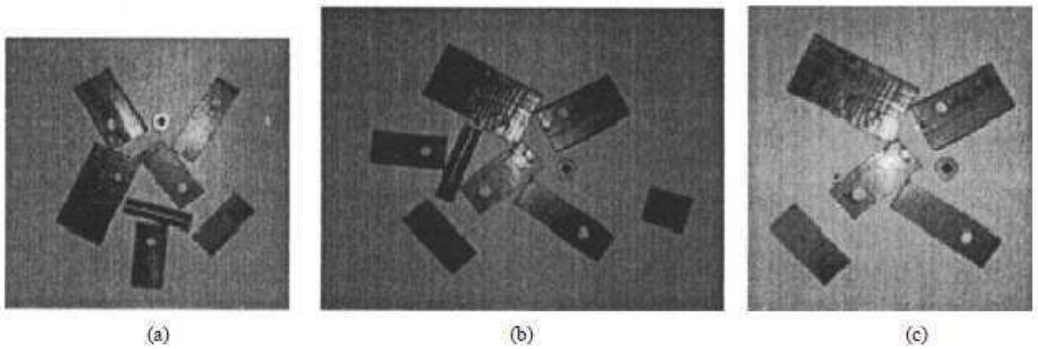


Fig.3.37. Image scenes with objects added or disappearing [134].

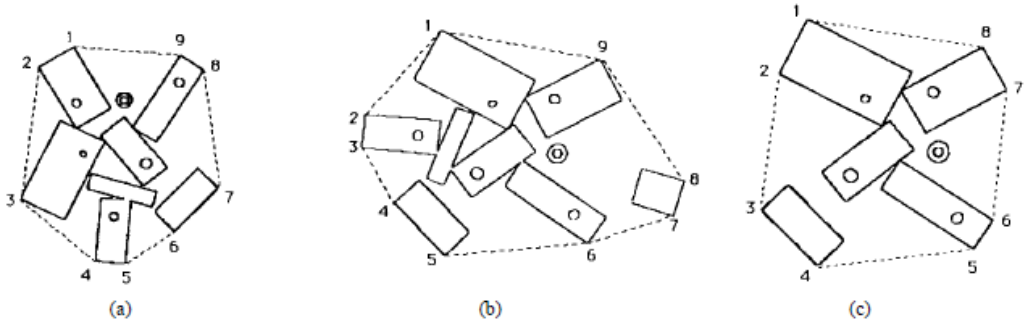


Fig.3.38. Corresponding convex hulls of the images in Figure 3.37 [134].

After detecting the corresponding vertices on two convex hulls related by an affine transformation, groups of four consecutive vertices are formed on each convex hull. The ratio of areas of the two triangles per group is the invariant affine measure used to detect matched groups as shown in Figure 3.39.

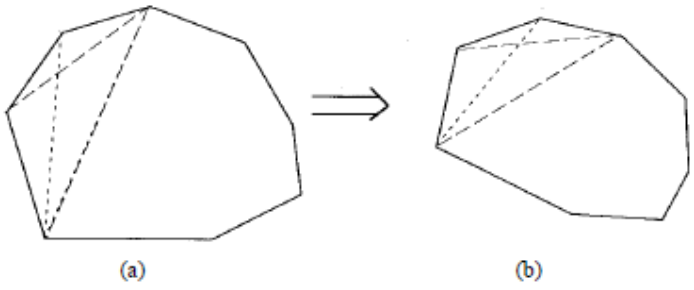


Fig.3.39. Affine invariants in Convex hulls [134].

The four points in every group with the corresponding points in the matched group enter in the calculation of the six unknowns of the affine model described in section 3.3.5.

3.3.3.2.2.7 Matching using level lines primitives

Almehio, Bouchafa and Zavidovique [135] have recently presented their work on image registration. It is based on level lines that form the boundaries of the image level sets. A level set is a set of image pixels with intensities greater than or equal to a given threshold. One can extract all the level sets in an image by using a series of thresholds. The obtained level sets are included one in another. Therefore, the level lines could be locally juxtaposed but never cross.

The estimated transformation model that relates the two images to register is either an affine or projective. For affine transformation, the authors have formed, using intersecting level lines, two primitives in the form of "Y" and "Z" shapes shown in Figure 3.40. So three intersecting level lines (four non collinear points) are needed to complete these two forms. For the Y-shape, the four non collinear points P_0, P_1, P_2 and P_3 define barycentric coordinates in considering P_0 with respect to the other points. These barycentric coordinates $\{a_1, a_2, a_3\}$ are affine invariants and are formulated as follows:

$$\begin{Bmatrix} a_1P_1 + a_2P_2 + a_3P_3 = P_0 \\ a_1 + a_2 + a_3 = 0 \end{Bmatrix} \tag{3.53}$$

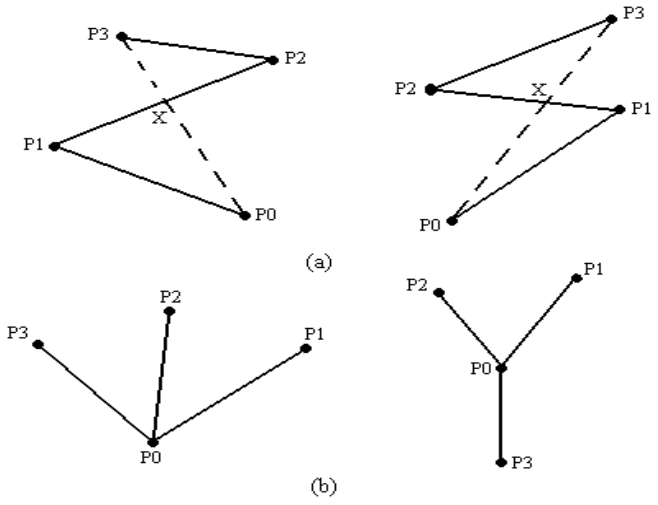


Fig.3.40. Primitive shapes for an affine transformation. (a) Z-shape. (b) Y-shape.

For the Z-shape, the ratio of the lengths of two collinear segments is set as the invariant measure. Thus, the two ratios r_1 and r_2 derived in Eq. (3.54) are used as invariant measures.

$$r_1 = \frac{||P_3X||}{||P_0X||} \; and \; r_2 = \frac{||P_2X||}{||P_1X||} \tag{3.54}$$

Where X is the intersecting point as shown in Figure 3.41 (a).

For two matched primitives in the two images, the four points $\{P_0(x_0,y_0), P_1(x_1,y_1), P_2(x_2,y_2)$ and $P_3(x_3,y_3)\}$ of the first primitive and the corresponding ones $\{Q_0(x'_0,y'_0), Q_1(x'_1,y'_1), Q_2(x'_2,y'_2)$ and $Q_3(x'_3,y'_3)\}$enter in the calculation of the affine model. To do so, the affine model in Eq. (3.53) can be expended into,

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \end{pmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{bmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ t_x \end{pmatrix}, \begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{pmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{bmatrix} \begin{pmatrix} a_{21} \\ a_{22} \\ t_y \end{pmatrix} \tag{3.55}$$

A least square estimation technique is used to solve this system.

For projective transformation, the W-shape is formed as shown in Figure 3.41.

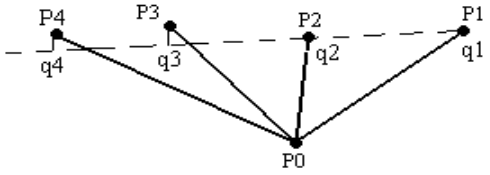


Fig.3.41. Primitive shape for a projective transformation [135].

In this case, four intersecting level lines are needed which is equivalent to five non collinear points. The cross ratio is an invariant measure under a projective transformation [44]. Therefore, the authors have used this property to derive the W-shape invariant measure,

$$Cross(q_1, q_2, q_3, q_4) = \frac{|q_1q_2||q_3q_4|}{|q_1q_3||q_2q_4|} \tag{3.56}$$

Where q_1, q_2, q_3 and q_4 are the four collinear projections the four points P_1, P_2, P_3 and P_4.

In this method, the repeatability of level lines is not considered in the primitive construction phase. Thus, many constructed primitives in the source image will not have corresponding primitives in the sensed image. Some of them will be matched incorrectly and this can hamper the solution. This is illustrated in Chapter 6 section 6.5.3.

3.3.3.2.2.8. Template matching

Bentoutou et al. [136] have proposed a registration technique for multitemporal or multisensor images in the area of remote sensing and Lou et al. [137] have presented their approach for an automatic registration of NOAA AVHRR satellite images. The two techniques are very similar. They have used the canny edge detector to detect first the required edges [25]. The edge points are selected by setting a threshold on the gradient magnitude. The threshold value is assigned as the average gradient magnitude [147].

They have used the normalized cross correlation as a similarity measure between the two regions lying inside two windows centered at the tested CPs in both images. If this correlation corresponds to a local maxima, than the tested CPs are set to be corresponding. They have claimed that some of these corresponding CPs are wrong. However, they have proposed a solution that can eliminate some of these false matches.

In the reference image, Harris corner detector [100] is used to detect the corners that form the CPs of the suggested registration applications. A correspondence mechanism between image regions around these points must be established to match the reference and the sensed images. This correspondence mechanism is called template matching. It finds the regions in the sensed image that are similar to those surrounding the CPs in the reference image. From every matched region, a CP position is extracted. The most important aspect of template matching is the similarity measure that is used to determine the degree of resemblance of windows in two images. They have defined first the central moment m_{pq} of a window W inside an image *f(i,j)*. It is given by,

$$m_{pq} = \sum_{(i,j)\in W} (i - x_c)^p (j - y_c)^q f(i,j) \tag{3.57}$$

Where p+q is the order of the moment and (x_c,y_c) are the coordinates of the centre of gravity of the window W. Their similarity measure is the normalized central moment nm_{pq}, given in Eq.(3.57), that is invariant under translation, rotation and scaling [136].

$$nm_{pq} = {m_{pq}}\Big/{m_{00}^{(p+q+2)/2}} \tag{3.58}$$

Every three matched CPs pairs from the reference and sensed images enter in the calculation of transformation model relating the two images. The same model, derived in Eq. (3.55), is obtained and solved also using least square estimation technique.

This method tries to match two primitives (Harris corners), one from each image, using the similarity (cross correlation) of their surrounding regions. The fact that the primitive is formed by one interest point rather than a group of them and also the introduced similarity measure lead to a large number of false matches as shown in Chapter 6 section 6.5.4.

3.3.3.2.2.9. Elastic matching

Elastic matching [193] has been employed in many image pattern matching problems such as face recognition, motion analysis, medical image analysis and computer vision. It is defined as an optimization problem with respect to a linear or nonlinear pixel to pixel mapping of two images I and J. Consider I and J as two sets of feature points $u_{i,j}$ and $v_{x,y}$ respectively where (i,j) and (x,y) are their coordinates in I and J respectively. Let F denote a 2D-2D mapping from I to J. Thus, the elastic matching is the minimization problem of the following objective function T with respect to F:

$$T_{I,J}(F) = D(I, F(J)) \tag{3.59}$$

Where D(.,.) is an Euclidian or absolute distance between the two image patterns and F(J) is the image obtained by fitting J to I using F. Therefore, F is the transformation that minimizes the objective function T. The distance D obtained is deformation invariant distance due to the minimization problem.

3.3.3.2.2.10. Relaxation

The relaxation [194] is a recursive parallel algorithm. Initially, a set of possible matching criteria is selected when matching every primitive from the first image with every primitive in the second image. For example, the angles of corners and the segment lengths are two matching criteria when the primitive is an image corner. This set is organized as a collection of nodes corresponding each to a primitive in the first image. For each node (primitive in the first image), a vector representing the primitive's geometric information and the set of labels representing the possible matching criteria is formed. A special label called "Null Character" signifies that the current primitive has no corresponding primitive in the second image. For each label

l, we associate the number $p_i(l)$ that is interpreted as the probability that l is effectively the matching criterion corresponding to the current primitive in the first image. Thus, we have $p_i(l)\epsilon[0;1]$ and $\sum_l p_i(l) = 1$. These probabilities will be updated by considering the coherence of the neighborhood. If relatively a lot of primitives in the neighborhood are compatible with the matching criterion l, than $p_i(l)$ will increase otherwise it will decrease. The essential task in this technique is the rule used to update the probabilities.

3.3.4. Model transformation estimation

A lot of techniques have been proposed so far to estimate the transformation that maps the set of features of the sensed image into the corresponding set in the source image.

We have used the Hough transform (section 3.3.4.1) to estimate our affine model since it has lead experimentally to correct estimation.

3.3.4.1. Classical Least square technique

This estimation technique is used by some previously explained feature matching techniques (Li et al. in section 3.3.3.2.2.5, Almehio et al. in section 3.3.3.2.2.7 and Bentoutou et al. in section 3.3.3.2.2.8). To illustrate, let us take a simple example: Given a set of N two dimentional points $\{(x_1,y_1), \ldots, (x_N,y_N)\}$, we try to find the straight line $y = ax+b$ that best fine the set of points. The associated error can be:

$$E(a,b) = \sum_{i=1}^{N} (y_i - (ax_i + b))^2 \tag{3.60}$$

The Least square technique consists of finding the optimal values of a and b that minimizes the error E by solving these two equations:

$$\frac{\delta E}{\delta a} = 0 \; and \; \frac{\delta E}{\delta b} = 0 \tag{3.61}$$

By finding the derivative of $E(a,b)$ with respect to a and b, we will obtain:

$$\frac{\delta E}{\delta a} = \sum_{i=1}^{N} 2(y_i - (ax_i + b)(-x_i) = 0 \tag{3.62}$$

$$\frac{\delta E}{\delta b} = \sum_{i=1}^{N} 2(y_i - (ax_i + b)) = 0$$

Eq. (3.62) can be rewritten as

$$\left(\sum_{i=1}^{N} {x_i}^2\right) a + \left(\sum_{i=1}^{N} x_i\right) b = \sum_{i=1}^{N} x_i y_i \tag{3.63}$$
$$\left(\sum_{i=1}^{N} x_i\right) a + \left(\sum_{i=1}^{N} 1\right) b = \sum_{i=1}^{N} y_i$$

In matrix form,

$$\begin{pmatrix} \sum_{i=1}^{N} {x_i}^2 & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} x_i y_i \\ \sum_{i=1}^{N} y_i \end{pmatrix} \tag{3.64}$$

Or

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{N} {x_i}^2 & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{i=1}^{N} x_i y_i \\ \sum_{i=1}^{N} y_i \end{pmatrix} \tag{3.65}$$

In image registration application and for a given transformation relating two images (source and sensed images), the least squares problem becomes finding the optimal transformation parameters that maps correctly the matched points in the two studied images. For example, Almehio et al [135] and Bentoutou et al [136] have used the least squares method to estimate their affine model. In Eq. (3.65) denote by **X'**, **Y'** the vectors of coordinates of the points in the sensed image, **h** and **k** the vectors of affine parameters and by **M** the 3x3 matrix. So, it can be rewritten as,

$$X' = M.h, Y' = M.K \tag{3.66}$$

The least squares task is to find the matrices ***h*** and ***k*** that minimizes the norms ||**X'-M.h**|| and ||**Y'-M.K**||. So by following the same solution method from Eq. (3.61) to Eq. (3.66), we can find solutions as follows,

$$X' = (M^T M)^{-1} M^T X \tag{3.67}$$

$$Y' = (M^T M)^{-1} M^T Y$$

Where **X**, **Y** the vectors of coordinates of the points in the source image.

The classical least square method can lead often to incorrect solutions due to the existence of more than one solution. However, the iterative least square estimation is a suggested solution to eliminate iteratively incorrect solutions and keep finally the correct one. The other two presented methods "RANSAC" and "Hough Transform" lead to correct solutions with noticeable difference with incorrect ones.

3.3.4.2. RANSAC

The RANdom SAmple Consensus (RANSAC) algorithm proposed by Fischler and Bolles [23] is a general parameter estimation approach designed to cope with a large proportion of outliers in the input data. Unlike many of the common robust estimation techniques such as M-estimators and least-median squares that have been adopted by the computer vision community from the statistics literature, RANSAC was developed from within the computer vision community. RANSAC is a resampling technique that generates candidate solutions by using the minimum number observations (data points) required to estimate the underlying model parameters. As pointed out by Fischler and Bolles [23], unlike conventional sampling techniques that use as much of the data as possible to obtain an initial solution and then proceed to prune outliers, RANSAC uses the smallest set possible and proceeds to enlarge this set with consistent data points.

The algorithm for image registration can be summarized as follows:

1. Select randomly the minimum number of points from the source and sensed images required to determine the model parameters.
2. Solve the system of parameters of the target transformation model relating the two images.
3. Determine how many points from the both images fit with a predefined tolerance.
4. If the fraction of the number of inliers over the total number of points exceeds a predefined threshold, re-estimate the model parameters using all the identified inliers and terminate.
5. Otherwise, repeat steps 1 through 4.

3.3.4.3. Hough transform

The concept of the Hough transform is to accumulate, in the interior of a space of representative parameters, the information confirming the presence of some particular forms or some particular transformations [119, 149].

Each image transformation has its own number of parameters called degrees of freedom (section 3.3.5). For example, an affine transformation has six parameters. To estimate these parameters, we need three points from the source image and their three matched points from the sensed image to solve the system. For this reason in Almehio et al. registration method [135], an affine primitive is a set of three non collinear level lines or four points. Note that the coordinates of the fourth point can be used only for verification. In this case, the Hough space has six dimensions (one dimension per parameter). Every matched couple of primitives gives its vote to a point of six coordinates in this space. The voting scheme accumulates the votes for the same point in the space. Finally, the coordinates of the point having the highest accumulated vote form the parameters of the target affine model.

One of the most important Hough transform feature is the selection of the parameter space. A high dimensional parameter space is not only slow but it can easily overrun the available memory. Finding lines in polar coordinates (ρ,θ) that are two dimensional requires a two dimensional Hough space corresponding to ρ and θ. Finding planes (y = ux + vy + w) requires a three dimensional Hough space corresponding to u, v and w. To illustrate the importance of the discretization of the parameter space to minimize the required memory space, consider for example the task of finding circles in a 400x200 image. A circle is defined by its center C(x,y) and its radius R. Therefore, the required Hough space is three dimensional space corresponding to x, y and R. Allowing the center C to be anywhere in the image, adds the constraint $0<x<400$, $0<y<200$ and $0<R<200$. The used three dimensional array accumulator has a size of 400x200x200 = 16 million values. Therefore, one must add other restricted constraints on the three dimensions to lower the accumulator size. This is called discretization of the space: It may be assumed that the radius of the searched circle does not exceed a certain value much less than 200. It may also be assumed that the centers of neighboring circles are at a minimal distance d from each other. Thus, the discretization of x and y (and even R) dimensions can be decreased to a lower number of values leading to a great minimization in the accumulator size.

3.3.5. Geometric transformation model: Affine transformation and its invariants [44]

Transformations are classified starting by the most particular one, the isometric transformation, than similarity transformation until reaching the more generalized affine transformation used in our application. Each transformation has its own number of degrees of freedom increasing form the isometric to the affine transformation. Therefore, the choice of the transformation model that could best estimate the image deformation is very important since additional degrees of freedom may imply a wider transformation space. Thus, it is better to select first which kind of transformations is more suitable for a studied application then construct the primitives and their invariant measures based on the selected transformation. We have restricted our study on affine transformation since it is the transformation used in our application.

An affine transformation is a non singular linear transformation followed by a translation. The matrix form of the affine transformation relating two points M(x,y) and N(x',y') in the two images is given by:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.68}$$

The parameters a_{11}, a_{12}, a_{21} and a_{22} form the elements of the 2x2 matrix **A**, called affine matrix, given in Eq. (3.68). (t_x, t_y) form the translation vector coordinates. The affine transformation is a decomposition of two fundamental transformations, namely rotations and non isotropic scalings. **A** can be written in terms of the parameters Ω, φ, λ_1 *and* λ_2 as shown in Eq. (3.69),

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{3.69}$$

$$A = R(\Omega).R(-\varphi).D.R(\varphi) \tag{3.70}$$

where Ω is the rotation angle, φ is the scaling angle both of them with respect to the horizontal axis, λ_1 is the scaling factor across direction of scaling (set by φ), λ_2 is the scaling factor across the normal direction to the direction of scaling. **R(Ω)** and **R(φ)** are rotation matrices and **D** is a diagonal matrix given by,

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \mathbf{R}(\varphi) = \begin{bmatrix} cos\varphi & sin\varphi \\ -sin\varphi & \cos\varphi \end{bmatrix} \tag{3.71}$$

Using Equations (3.69), (3.70) and (3.71), the affine parameters a_{ij} can be rewritten as follows,

$$\begin{pmatrix} a_{11} = \lambda_1 cos\varphi \cos(\varphi - \Omega) + \lambda_2 sin\varphi \sin(\varphi - \Omega) \\ a_{12} = \lambda_1 sin\varphi \cos(\varphi + \Omega) - \lambda_2 cos\varphi \sin(\varphi - \Omega) \\ a_{21} = \lambda_1 cos\varphi \cos(\varphi + \Omega) + \lambda_2 sin\varphi \cos(\varphi - \Omega) \\ a_{22} = -\lambda_1 sin\varphi \sin(\varphi - \Omega) + \lambda_2 cos\varphi \cos(\varphi - \Omega) \end{pmatrix} \tag{3.72}$$

To illustrate the role of matrix A, consider the image point M shown in Figure 3.42. The scaling axes x' and y' are also drawn with respect to the original image axes x and y. The scaling factor of M, derived in Eq. (3.73), is relative to its angle α with respect to scaling axis x'.

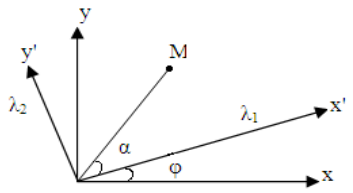


Fig.3.42. Scaling directions in an affine transformation.

$$s = \sqrt{\lambda_1{}^2 cos^2(\alpha) + \lambda_2{}^2 sin^2(\alpha)} \tag{3.73}$$

The affine invariants are the following as stated in [44]:

- Parallel lines: Consider two parallel lines. These intersect at a point at infinity. Under an affine transformation this point is mapped to another point at infinity. Consequently, the parallel lines are mapped to lines which still intersect at infinity, and so are parallel after the transformation.
- Ratio of lengths of parallel line segments: The length scaling of a line segment depends only on the angle between the line direction and scaling directions. Suppose the line is at angle α to the x axis of the orthogonal scaling direction, then the scaling magnitude is $\sqrt{\lambda_1{}^2 cos^2\alpha + \lambda_2{}^2 sin^2\alpha}$. This scaling is common to all lines with the same direction, and so cancels out in a ratio of parallel segment lengths.
- Ratio of areas: This invariance can be deduced directly from the decomposition Eq. (3.73). Rotations and translations do not affect area, so

only the scalings by λ_1 and λ_2 matter here. The effect is that area is scaled by λ_1 λ_2 which is equal to det(**A**). Thus the area of any shape is scaled by det(**A**), and so the scaling cancels out for a ratio of areas. It will be seen that this does not hold for a projective transformation.

3.4. Our contribution

The major weakness of existing interest points and edge corner detectors is in the correct detection in real images where the noise at the included objects boundaries is relatively high. In addition, interest point detectors are based on image intensities. Therefore, the detection of their interest points is highly dependent on threshold on intensities which is not automatic and very sensitive to noise.

Our contribution is to detect our first features, "repeatable edge corner points" on an object contour that are very accurate, automatic and less sensitive to noise. They can be used in many computer vision applications like image registration [52], polygonal approximation [53], object recognition, etc. In fact, we have used the corners angles and lengths ratios of its two adjacent segments as matching keys. Thus, two objects are matched if their contours have matched corners in angle and length ratio. In addition, since corners are grouped together according to their contour, we don't need to use an existing grouping method like RANSAC [23].

We have focused on corners located on edges since edges are one of the most important image features that are repeatable versus various image transformations [3]. Our goal is to detect feature points that are also well localized and repeatable against many image deformations. Therefore, they can be used as interest points on an object's contour where a descriptor can be formed at every interest point using its local information from its surrounding pixels.

Using the detected edge corners, we have proposed an approach for detecting our second features, "Dominant Corners", inspired by that presented by Masood [127]. We have searched for dominant corners that best approximate a given shape by a polygon having these corners as vertices. The differences between our work and the existing works are in the nature and stability of the selected points and in the method used to select them among others. The selected points of Masood [127] are points that

correspond to a deviation in the edge direction. Our selected points are edge corners that are intersections of two straight edges. For us, not every deviation in the edge direction corresponds to an edge corner. It may correspond to a noisy edge direction. In addition, Masood iteratively eliminates unwanted corners using a measure called associated error value (AEV). The AEV at a dominant point is the perpendicular squared distance of this point to the straight line joining its previous and next dominant points. However, the error measure associated to our DC is the Integral square error which is proportional to the area bounded by the DC straight edges and their approximating polygon's segment. So, it is similar to the criterion used by Wall and Danielson [118] that relies on the area of the region included between the edge part and its approximated segment rather than only relying on the maximal distance like in the method of Pavlidis [117].

The most important matter in image registration is the repeatability of the CPs for a side and the correct CPs matching. Since unrepeatable and false matched CPs hamper the solution in a lot of existing techniques like Lou et al. [137], Bentoutou et al. [136] and Almehio et al. [135] techniques. In our technique, we have built and matched image primitives composed by a group four consecutive edge corners called "Dominant Corners". This grouping gives to the primitive more efficiency in terms of uniqueness and dissimilarity. In addition, we have suggested an efficient matching algorithm to reduce the number of false matches.

The DCs repeatability can be compared to that of the Harris corners introduced by Bentoutou et al. [136] and Lou et al. [137]. In this work, the introduced iterative suppression technique leads to high repeatable DCs (CPs). High repeatable primitives are formed by a group of four DCs. In addition, two primitives are matched if their four DCs are corresponding. This will lead to a smaller set of repeatable voters (primitives) and also more accurate transformation estimation compared to the voters introduced by Harris corners in [136, 137].

Using the detected dominant corners on an image contour, our primitive is formed by grouping four consecutive DCs. So our technique can be classified as feature based technique since it is based on DCs that are feature points in the image domain. Its invariant measure is the ratio of the areas of two triangles whose vertices are three

DCs among the four. This invariant measure is similar to that used by Yang and Cohen [134]. However, our DCs are more repeatable than their convex hull vertices.

The matched primitives from both images enter as voters to estimate the affine model of six parameters. Therefore, we have relied on the Hough technique based on a Hough space of six dimensions and apply a voting scheme to accumulate the votes of every set of six parameters. The set that gets the highest votes constructs our target affine model.

Two main goals are achieved in this research. The first one is the detection of new CPs (DCs) that are very repeatable versus affine deformations. The second one is a new matching scheme proposed for proper transformation estimation.

4

A robust Edge Based Corner Detector (EBCD): Straight Edges, Edge Corners and Dominant Corners

An edge corner is an edge point that corresponds to a deviation in the edge direction. Or in other terms, it is the intersection point between two non collinear straight edges where a straight edge is an edge segment linked nearly in the form of a straight line.

4.1. EBCD block diagram

The main steps of this detector can be summarized as follows:

(v) Edge detection with updates in the linking phase: we have used the Kirsch edge detector with updates, described in section 4.2, for proper detection of edge corners. The output of an edge detector is a binary image of edges. These edges form the contour of the objects in the image that are our candidates to test the presence of corners.

(vi) Straight edges detector: the straight edges are parts of a contour linked in the form of straight lines. They can be called "Edge Segments" and described in section 4.3. The goal of applying a straight edge detector is to divide a contour into a sequence of straight edge segments of different lengths using the edge point chain codes. The robustness of the straight edge detector is its ability to properly detect a straight edge even if it is corrupted with some noisy pixels. By definition, a noisy pixel is an edge pixel whose direction is different from the main straight edge direction. This detector should detect adequately these noisy pixels and eliminate them in order for the corner detector to extract the true corners and not mix them with noisy pixels since both noisy and corner points correspond by definition to an edge deviation. We published this detector in [28, 198].

(vii) Corner Detector: a corner is defined as the intersection point of two non collinear straight edges of appropriate lengths (In Chapter 5 experiments in section 5.1: the length threshold is set to be equal 10 pixels). This is described in section 4.4. In addition, a pattern recognition application using corners is developed in section 4.5. We published this detector in [28, 198].

(viii) Dominant corners detector: among the set of edge corners, a smaller set called "Dominant Corners" or "DCs" is selected iteratively. These DCs have a great repeatability under various image transformations as described in section 4.6. We published this detector in [53].

The algorithm block diagram is shown in Figure 4.1.

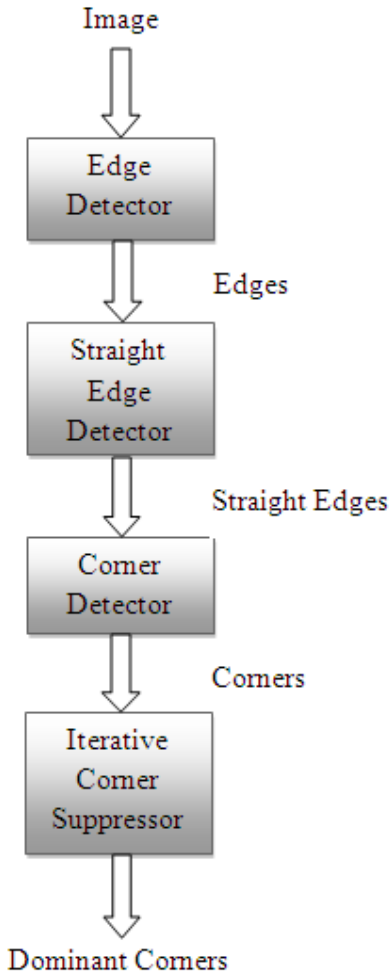


Fig.4.1. The corner detection functions.

4.2. Edge detector with suggested updates

We have used the existing edge detector based on Kirsch algorithm composed of the five classical steps of edge detection (see chapter 3). These steps are grouped into two functions for a temporal optimization. This grouping is a part of the open source CLEOPATRE project [17] but used with updates to meet our requirements. The first grouping performs gradient vector calculation, thresholding on the gradient norm, and edge thinning. The second one operates on edge linking and closing.

4.2.1. First grouping: gradient vector calculation, thresholding and edge thinning

4.2.1.1. Gradient vector computation

We have used Kirsch operator that offers a very good compromise for our mobile and autonomous robotic applications between the quality of results and the low processing time.

At every image pixel, the gradient vector is calculated by placing the Kirsch four masks (chapter 3). The norm and the direction of this vector are calculated using Eqs. (3.47) and (3.48). Therefore, the gradient norm reflects the strength of the edge while the direction is normal to the edge direction. Figure 4.2 (a) provides an example where the image intensity (8 bit character) is shown at every pixel. The gradient norm, based on Kirsch, is calculated at every pixel in Figure 4.2 (b).

0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255
0	0	0	0	100	255	255	255	255	255

(a)

0	0	0	0	0	0	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	100	255	155	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(b)

Fig.4.2. Gradient norm on an edge using Kish operator.

4.2.1.2. Thresholding

Thresholding is the process that enables to extract the edge points. This is achieved by presetting a threshold on the gradient norm called "*thH*". If an image pixel has a gradient norm greater than *thH*, it will be classified as an edge pixel. For example in Figure 4.2 (b), the corresponding edge points are grey colored and form the edge in this figure.

4.2.1.3. Thinning

Usually, the thresholding phase can produce more than one pixel in the normal direction to the edge. For example in Figure 4.2, if the threshold is equal to 80, the

edge width, shown in Figure 4.2 (b), will have a width of 3 pixels in the normal direction to the edge. In addition, this problem of thick edges is well illustrated in Figure 4.3. It is shown that an edge can be of thickness more than one pixel in the normal direction to the edge.

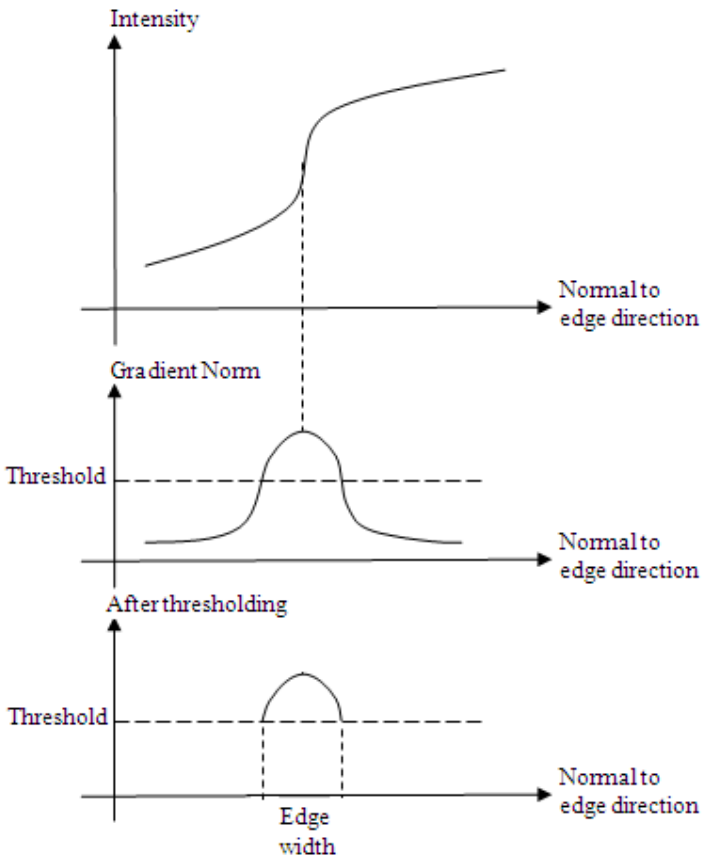


Fig.4.3. More than one pixel can exist in the normal direction to the edge.

For this reason, the thinning step is introduced. Its goal is to obtain an edge of thickness equal to one pixel at maximum. To explain the algorithm, one must know these definitions:

- Current pixel: the pixel currently tested.
- Past pixels: pixels in the 3x3 neighborhood of the Current pixel and that are already tested as edge pixels.

The thinning algorithm is shown in Figure 4.4.

- *If the current pixel is an edge pixel.*
  - *Test its Past pixels in the normal direction to the edge.*
  - *If one of them is also an edge pixel.*
    - *If its gradient norm is greater than that of the Current pixel*
      - *The current pixel is removed from the edge points list.*
    - *Else*
      - *The Past edge pixel is the one removed.*

Fig.4.4. Thinning algorithm.

4.2.1.4. First problem: rounding angle problem

The three already discussed steps are grouped together into one function. Thus, for a single image processing (video mode) and at every pixel, the gradient vector is calculated then thresholding on the gradient norm is applied then thinning step is initiated. This grouping output a binary image of the edge points called "edge image".

However, a problem arises since the gradient computations will round the angle. Figure 4.5 and Figure 4.6 show the result on right and acute angles respectively. Note that the detected angles are rounded. To solve this problem, we will introduce the notion of “half corner” in the corner detection phase explained in section 4.4.

Fig.4.5. Gradient norm on a right angle using Kish operator.

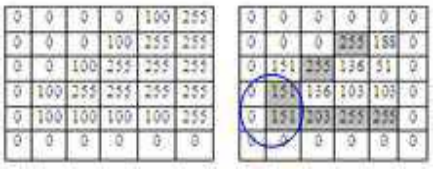


Fig.4.6. Gradient norm on an acute angle using Kish operator.

4.2.2. Second grouping: edge linking and closing

4.2.2.1. Automatic Edge linking

The edge linking processes all the pixels in the generated binary edge image. The idea is to link the neighboring edge pixels together in the attempt to form the contours in a given image. A contour is a sequence of edge pixels linked together. It is characterized by a Head pixel, a Tail pixel and by the transition codes (directions of the edge coded using Freeman codes) at every belonging edge pixel from the Head to the Tail.

At any unlinked Current edge pixel, the automatic linking algorithm is shown in Figure 4.7.

- *For all the 8 neighbors in the 3x3 neighborhood.*
  - *Search for the unlinked optimal edge neighbor having the greatest gradient norm.*
- *If this optimal edge neighbor exists.*
  - *Link it to the Current pixel by setting its edge direction towards this neighbor.*
  - *This optimal edge neighbor becomes the Current pixel and the algorithm restarts.*
- *Otherwise the linking ends.*

Fig.4.7. Linking algorithm at an unlinked edge pixel.

4.2.2.2. Automatic Closing

The closing phase is initiated when trying to link an edge pixel and no unlinked neighboring edge pixel exists as shown in Figure 4.8 (a) or in other terms when linking ends. The closing first introduces a second threshold less than that used in the thresholding phase called "*thL*". It starts by examining the unlinked pixels in three directions that form a cone of 45^{o} at vertex as shown in Figure 4.8 (b). The closing algorithm at an unlinked edge pixel is shown in Figure 4.9. The variables used in the algorithm are:

- *AllNeighLinked*: a flag signaling that the three tested neighbors are already linked.

- *AllNeighWeek*: a flag signaling that all the three test neighbors have a gradient norm below *thL*.

- *EdgeFound*: a flag signaling that one of the three tested neighbors is an unlinked edge pixel.

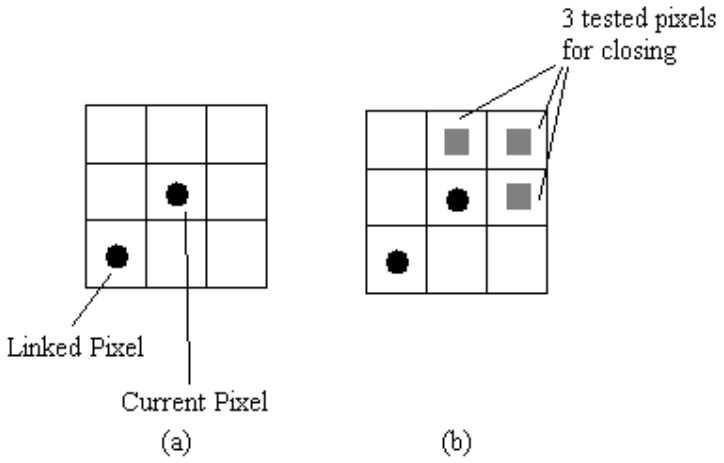


Fig.4.8. The 3 selected pixels for closing.

- *For the unlinked neighbor in one of the three defined directions having the greatest gradient norm among the three.*
  - *If it exists*
    - *If it is an edge pixel*
      - *Closing ends and returns EdgeFound.*
    - *Else if its gradient norm is greater than thL.*
      - *It will be selected and linked as an edge point.*
      - *It becomes the Current pixel.*
      - *Restart the algorithm.*
    - *Otherwise the closing ends and returns AllNeighWeak.*
  - *Otherwise the closing ends and returns AllNeighLinked.*

Fig.4.9. Closing algorithm at an unlinked edge pixel.

4.2.2.3. Grouping

The linking and closing procedures can be linked easily in one automatic algorithm that requires one examination of the pixels in the edge image in video mode. The algorithm is explained in Figure 4.10 where the only variable used is:

- *dmax*: the maximal allowed prolongation distance.

- *For each pixel in the edge image (video processing).*
  - *If the Current pixel is an unlinked edge pixel.*
    - *Set it as the Head of the contour.*
    - *Repeat until setting the Tail*
      - *Launch the linking algorithm shown in Figure 4.7.*
      - *Launch and continue the prolongation algorithm shown in Figure 4.9 whenever the prolongation distance is less than dmax.*
      - *If the prolongation has ended with AllNeighLinked flag than set the Tail as the last linked edge pixel.*
      - *Else if the prolongation has ended with AllNeighWeak flag than cancel all the prolongated pixels so far and set the Tail as the last linked edge pixel.*
      - *Else if the prolongation has ended with EdgeFound flag than continue.*

Fig.4.10. Automatic Linking/Closing algorithm.

4.2.2.4. Second problem: Linking in straight direction

The explained linking step was designed to follow "strong edges in straight direction" in an image, in order to extract structures of the 3D scene. When a fork between edges is encountered, it follows the strongest edge with the greatest magnitude. Our problem with this strategy is the ignorance of the other possible edges due to their smaller gradient magnitudes. This can lead to an elimination of real corners when we have an intersection of two edges in the form of T. This problem is an inappropriate edge linking because the existing edge detector focuses on detecting straight edges. Figure 4.11 (a) presents a colored image of two rectangles with different colors and Figure 4.11 (b) shows the corresponding edge image. Two edges are formed and each one is represented by a different color. Consider the points A and B where real corners exist. Each of the edge pixels at A and B has two neighboring edge pixels where one of them is in a straight direction and the other in the normal direction. Using the existing linking strategy, the algorithm chooses and links only the one who has the greatest gradient norm which is, in this case, the straight neighbor

that does not correspond to a change in the edge direction. So, no corners will be reported at A and B at the end of our corner detector.

4.2.2.5. Our Suggested Update to solve the problem

Therefore, the linking phase at these points should be updated as follows:

- If an edge point has more than one neighboring edge points, mark these points as double points, like points A and B in Figure 4.11. (b).
- At the double points:
- The straight edge detector, detailed in §4.3, is launched and two straight edge segments, starting from the double point, should be detected.
- If the detected edge segments are not collinear with appropriate lengths (greater than a preset threshold), than the double point corresponds to an edge corner.

Usually, our algorithm is initiated to check a corner presence at only edge pixels having deviations in the edge direction. Therefore, we mark the double points in the linking phase to force the algorithm to test them for corner presence besides those who correspond to an edge deviation. Since the double points do not correspond usually to an edge deviation.

In Figure 4.11 (b), we have shown two edge pixels A and B among many that correspond to an intersection of two edges in the "T" form. The existing edge linking, Figure 4.11 (d), from top to bottom at A has followed the straight direction which is direction 6 in Freeman code without considering the second intersecting green edge. Also the edge linking from left to right at B has followed the straight direction which is direction 1 in Freeman code without considering the second intersecting blue edge. Thus, in both situations, the existing linking algorithm will not take any action according the intersection between the two edges. So, when searching for corners, a corner will not be detected at edge points A and B since they do not correspond to an edge deviation. To overcome this problem, we use our updated linking phase, shown in Figure 4.11 (c), which marks the points A and B as double points. In the straight edge detector (next stage), at these double points we will obtain two non collinear straight edges: at A, the navy and green edges. At B, the navy and blue edges. In the corner detector, these two points will be then reported as corners.

Fig.4.11. The problem in the linking phase of the existing edge detector: (a) original image. (b) edge image. (c) updated linking phase with double points. (d) old linking phase.

4.3. Straight edges

Our objective is to detect edge segments that can be considered straight using only their chain codes. However, we must distinguish first between two kinds of straight edges: Perfect and real straight edges.

4.3.1. Perfect straight edges

We notice that a corner can be defined as an intersection of two non collinear straight edges. So our idea is to classify a given edge as a sequence of non collinear straight edges. A perfect straight edge is an edge whose chain code is composed of one code or two codes at maximum. There are eight different straight edges corresponding to a unique code among the eight Freeman codes. For example, a perfect horizontal straight edge has a chain code of only 0 or 4 and a perfect straight edge along the first diagonal has a chain code of 1 or 5 as shown in Figure 4.12.

In addition to these eight cases, a perfect straight edge has a chain code composed of two codes: one primary and the other secondary with a difference equal to one between them. The classification of these two codes is done according to their

frequency of occurrence in the edge's chain code. This is illustrated in Figure 4.13. In Figure 4.13, five edges are considered starting from one origin O. Edges1 and edge5 are those that have a unique code 0 and 1 respectively, and their slope are 0^o and 45^o. Edge3 is the one that have double codes 0 and 1 of the same frequency of occurrence so its slope is 22.5^o and it is the bisector of the angle formed by edge1 and edge5. Edge2 is near to edge1 and has also double codes 0 and 1 but with different frequency of occurrence. Code 0 is primary and code 1 is secondary. Same result can be seen in edge4 that has code 1 as a primary and code 0 as secondary.

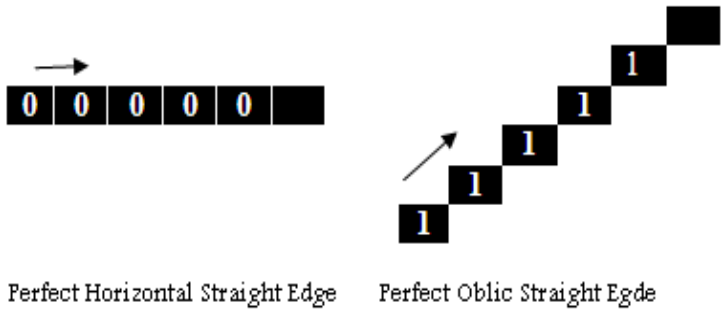


Fig.4.12. Straight Edges with unique code.

As a conclusion, we can say that the perfect straight edges that have double codes are of two kinds. The first kind of edges has double codes of same frequency (edge3). It is equidistant between two straight edges of unique code (edge1 and edge5). The second kind of edges also has double codes but of different frequency (edges2 and4) and it is also between two straight edges of unique code corresponding to the primary or secondary code. Here, the nearest one's code forms the primary code. In Figure 4.13, Edge1 is the nearest to Edge2, so the primary code for Edge2 is 0.

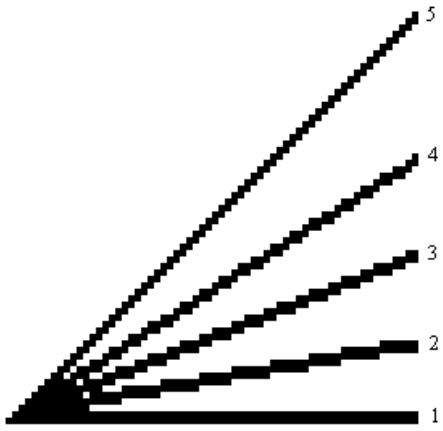


Fig.4.13. Straight edges with double Freeman codes.

4.3.2. Algorithm explanation and real straight edges

In real images, the chain code of a straight edge that should be composed of one or two successive codes has usually more than two codes. Our goal is to identify which of them are the primary and secondary codes and reject the remaining codes. This problem is due to natural noise at the object borders.

Real straight edges are shown in Figure 4.14 (a) where some perturbations, circled in the image of edges Figure 4.14 (c), are encountered along these edges. The challenge is to build an intelligent algorithm that can identify these perturbations and detect real straight edges that meet at corner points as shown with their angles in Figure 4.14 (b).

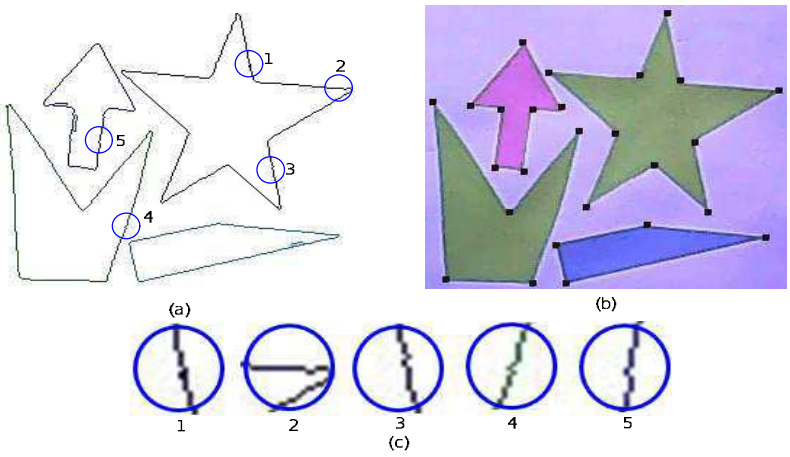


Fig.4.14. (a) image of edges, (b) detected corners, (c) noisy pixels.

While moving across the edges in a given image, our straight edge algorithm can be initiated at the current edge point in one of two cases:

- If it is a double point.
- If it corresponds to a deviation in the edge direction.

The initiation of the algorithm is shown in Figure 4.15 where the variables used are:

- *ppdir*: pixel previous edge direction as shown in Figure 4.16 at a corner point A located on an edge.

- *cdir*: pixel current edge direction.

- *diff*: direction difference.

- *FlagInit*: flag to signal the initialization.

- *FlagInit = false.*
- *If the current edge pixel is a double point*
  - *FlagInit = true.*
- *Else*
  - *Calculate diff = cdir – ppdir.*
  - *If (diff != 0)*
    - *FlagInit = true.*
- *If (FlagInit)*
  - *Initiate the Straight edge algorithm twice at the current pixel: in the current direction cdir and in the opposite direction to ppdir.*

Fig.4.15. Condition to initiate the straight edge detector.

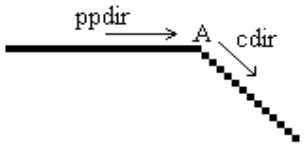


Fig.4.16. The current and previous directions at an edge pixel A.

The purpose is to test the existence of a real straight edge with successive infected pixels less than *m* and of a length greater than a threshold *d* (In the experiments in Chapter 5 section 5.1 on detecting the straight edges and corners, *d* is et to be 10 pixels and *m* to be 2 pixels). The idea behind the algorithm is to start testing an image edge starting from its head to its tail. Consider the first encountered edge direction as primary direction and the next encountered edge direction as secondary edge direction. While moving the current pixel to the tail, increment the frequency of primary or secondary directions when the current pixel's direction corresponds to primary or secondary directions, respectively. Or increment the frequency of infected pixels, if the current pixel's direction differs from them. If the number of infected pixels exceeds a given threshold *m*, the edge test stops at the current pixel. If the edge length is greater than another threshold *d*, the edge traversed so far is considered as a straight edge.

Let us define the variables used in the algorithm:

- *pdir*: edge primary direction.

- *pdirfreq*: frequency of *pdir*.

- *sdir*: edge secondary direction.

- *sdirfreq*: frequency of *sdir*.

- *InfectedCount*: number of infected pixels encountered so far.

- *el*: edge length.

- *SecondaryFound*: a flag that is set when a secondary direction is found.

- *diff*: absolute difference between *cdir* and *pdir*

The algorithm details are shown in Figure 4.17.

An infected pixel is an edge pixel, located on a real straight edge, whose edge direction is different than those of the corresponding perfect edge. Therefore, the variable *diff* is introduced. The search is for the edge direction, if exists, that differs by one from the primary direction *pdir* knowing that a perfect straight edge can have a chain code composed at maximum of two codes with difference equal to one. The variable *InfectedCount* is incremented each time an infected pixel in encountered and it is cleared only if the current edge pixel direction *cdir* is equal to *pdir*. On the other hand, if *cdir* is equal to *sdir*, the variable *InfectedCount* should not be cleared. In Figure 4.18 (a), consider the example of a straight edge whose pixels are in black. It is linked with *pdir* = 1 and *sdir* = 2. At point A, the *cdir* is 3 so *InfectedCount* is incremented. Next at point B, the *cdir* is 2 and it is clear that the remaining edge in gray forms a different straight edge. If we reset *InfectedCount* at B when *cdir* = *sdir* = 2, the algorithm will not stop and will consider both edges, in black and in gray, as one straight edge which is incorrect. A final point should be discussed which is the need to update *pdir* and *sdir*. The logical reason for this case is that a straight edge primary direction is not necessary the first encountered *cdir* as shown on the straight edge in Figure 4.18 (b). Initially *pdir* = 4 and *sdir* = 5. But after traversing the remaining edge pixels, it is clear that *pdir* should be 5 and *sdir* should be 4.

- *Record the first edge direction at E (Freeman code) as pdir and initialize pdirfreq = 1. Initialize InfectedCount = 0, el=1 and SecondaryFound = false and proceed to the next linked pixel.*
- *Iterate until the current pixel C reaches the tail of the edge.*
  - *Calculate diff.*
  - *If (Not SecondaryFound):*
    - *if (diff==1), record cdir as sdir, initialize sdirfreq = 1 and put SecondaryFound = true.*
    - *else if (cdir==pdir) than increment pdirfreq.*
    - *else increment InfectedCount.*
  - *Else*
    - *If(diff>=2)or((diff==1)and(cdir!=sdir))current pixel is an infected pixel than increment InfectedCount.*
      - *if (InfectedCount>m): meaning that we are reached the maximum allowable infected pixels: stopping criterion=>Ends of straight edge test:*
        - *if(el>=d) than the current edge is a real straight edge.*
    - *else if (cdir==pdir) than increment pdirfreq and set InfectedCount=0.*
    - *else if (cdir ==sdir) than increment sdirfreq.*
    - *update pdir and sdir: pdir is the direction that has the maximum frequency between (pdir, pdirfreq) and (sdir, sdirfreq).*
  - *el++.*

Fig.4.17. Straight edge detector.

The robustness of the straight edge detector against noise due to shadowing effect is due to the usage of the parameter *Accn*. Normally, the shadowing effect will introduce some noisy edge pixels having their directions different from the two main straight edge directions.

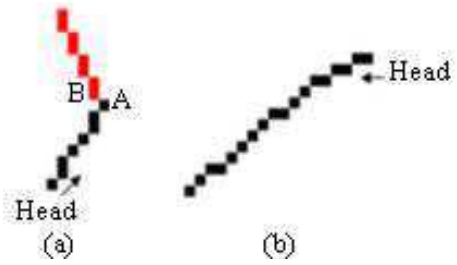


Fig.4.18. Two straight edges.

4.4. Corner detection

By definition, a corner is the intersection of two non collinear straight edges with appropriate lengths which means greater than a preset threshold. We should distinguish between two situations: the first one is when the two non collinear straight edges intersect at an edge point and the second one is when they intersect outside the edge.

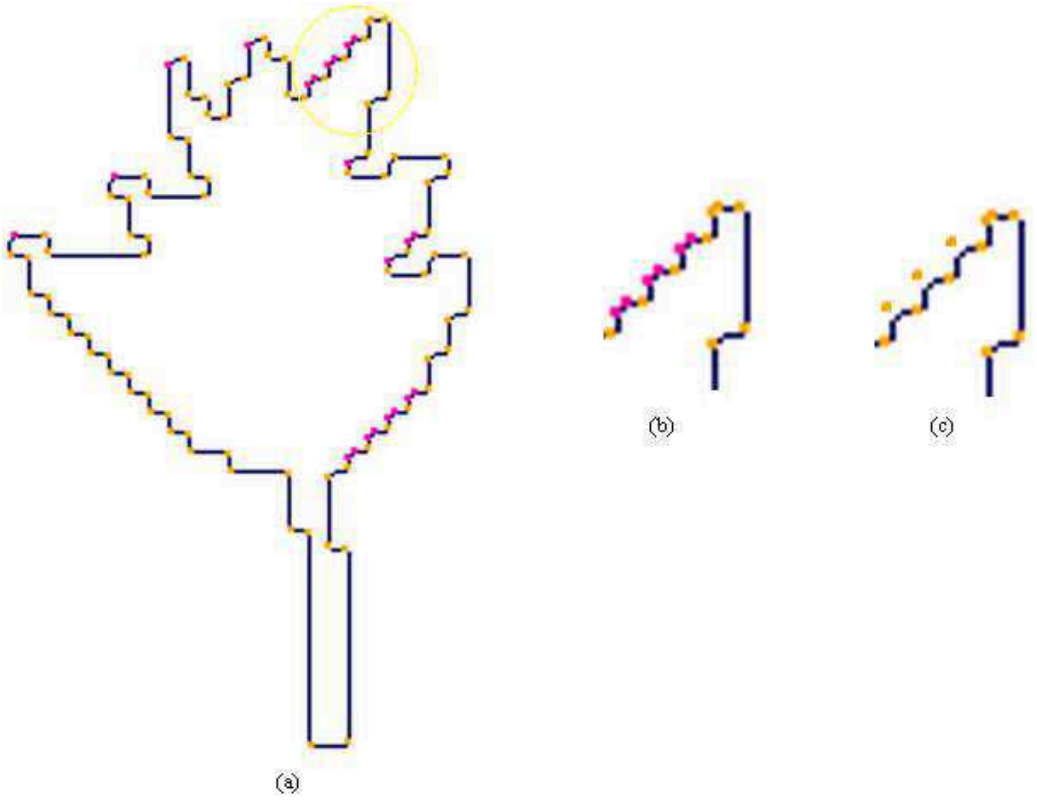


Fig.4.19. (a) edge image of a leaf shape with CCs in orange and HCs in pink. (b) The CCs and HCs of the top right circled part. (c) HCs combination results.

In the first situation, the corner point is called "Complete Corner" or "CC". They are shown as orange points on the contour of a leaf shape of Figure 4.19. In the second situation, sometimes due to the fact that the edge detector rounds the angle described in section 4.2.1.4, the intersection point of two non collinear straight edges is located outside the edge. To overcome this error, we mark the endpoints of the two straight edges as "Half Corners" (HCs). These HCs are shown as white points in Figure 4.19. By definition, a HC is an endpoint of a straight edge. It corresponds to an intersection point between a straight edge and a very small edge part (not classified as a straight edge due to its small length) like the points C1 and C2 shown in Figure

4.20. If the distance between two HCs is very small (less than three pixels), we extend their straight edges. If they intersect at a very small distance from the two HCs than a true corner is reported.

In Figure 4.20, C1 is a HC since a change in direction occurs from 7 to 5 and one of its intersecting edges DC1 is a straight edge with considerable length while the other edge C1C2 is not due to its small length. C2 is also a HC for the same reason. Thus, these two HCs can form a true corner by extending the corresponding straight edges along the directions of the corresponding segments. If they intersect at a point very near to the original HCs with a considerable gradient norm, this intersection is considered as a true corner point. In Figure 4.20, the straight edge DC1 is a straight edge passing through C1 and having a direction (slope) 7. EC2 is also a straight edge passing through C2 and having a direction 0. They will intersect at point C. The summary of the algorithm for HCs combination is shown in Figure 4.21.

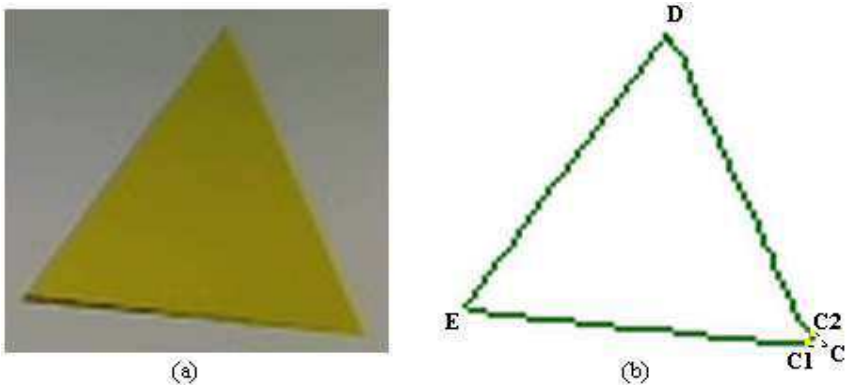


Fig.4.20. (a) Original image. (b) Edge image and HCs.

Figure 4.19 (a) shows the edge CCs in orange and HCs in white located on the contour of a leaf shape. After the application of the HCs combination algorithm, every two neighboring HCs satisfying the combination conditions will be combined into one CC located outside the contour. The three pairs of HCs in the image part shown in Figure 4.19 (b) are combined into three CCs in Figure 4.19 (c).

- *For each pair of HCs:*
  - *Calculate the distance d between them.*
  - *If d<3 pixels:*
    - *Prolongate the straight edges starting from the HCs.*
    - *If they intersect at a point I of maximal distance of 3 pixels from both HCs and I is not a neighbor of a complete corner CC*
      - *I is reported as CC.*

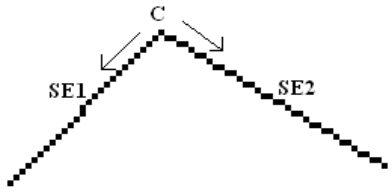
Fig.4.21. HCs combination algorithm.

Each CC is characterized by its angle, which is the difference between the directions of the two intersecting straight edges, and its adjacent segments lengths ratio. These corners are compared in chapter 5 with other interest point detectors in the literature (SUSAN, SIFT, FAST, Harris and Harris-Laplace) already explained in the bibliography chapter 3 (§3.1).

4.5. Image matching using corners: a 2D shape recognition application

In this section, we have developed a 2D shape recognition application using the edge corners. The goal is to recognize a target shape even when it is rotated, translated and essentially scaled. Mathematically, this deformation is well modeled by a similarity transformation (see chapter 3). Theoretically, this transformation conserves angle and lengths ratio. Therefore, two characteristics are selected for our edge corners (as shown in Figure 4.22):

- Angle: it is the difference between the two adjacent straight edges directions.
- Length Ratio (LR): it is the ratio of the lengths of the two adjacent straight edges.



SE1	SE2	C
dir=5.1	dir=7.6	Angle = 7.6-5.1 = 2.5
Length=30	Length=40	LR = 40/30 = 1.34

Fig.4.22. Edge corner characteristics: Angle and LR.

Practically, we have studied the stability of these two characteristics under scale variation. For this purpose, we have selected five corners (A, B, C, D and E) on the contours of an image at scale S shown in Figure 4.23 (a). Also, we have taken three other figures shown in Figure 4.23 (b), (c) and (d) at different scales S/2, S/3 and S/4 respectively. Table 4.1 show the repeatability of corners characterized by their angles and length ratios (LR). It is clear that the angle and the LR remain approximately constant against scale variation. Thus, these two parameters can be chosen as invariant corner's parameters under image similarity transformation.

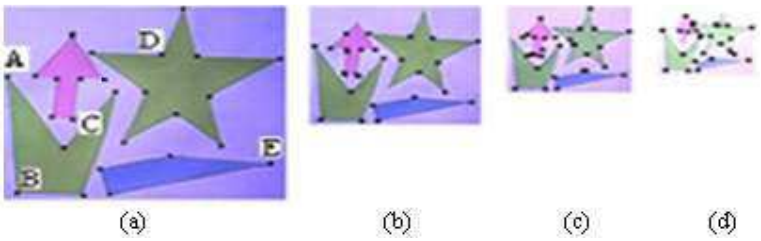


Fig.4.23. Repeatability of corners under scale variation.

Table 4.1. Length ratio and angle of some corners at different scale.

S	S1		S1/2		S1/3		S1/4	
	L.R	**Angle**	**L.R**	**Angle**	**L.R**	**Angle**	**L.R**	**Angle**
A	1.64	<6,7>	1.62	<6,7>	1.65	<6,7>	1.68	<6,7>
B	2.37	<0,2>	2.33	<0,2>	2.26	<0,2>	2.28	<0,2>
C	2.51	<2,4>	2.66	<2,4>	2.6	<1,4>	3.12	<2,4>
D	2.69	<2,4>	2.61	<2,4>	2.57	<2,4>	1.42	<2,4>
E	1.55	<4,4>	1.01	<4,4>	1.1	<4,4>	1.17	<4,4>

We have developed an algorithm using the corner's angle and LR as invariant parameters to detect a 2D shape in a given image. The idea is to embed the suggested algorithm in an autonomous robot whose target is to detect a desired shape and tries to bring it.

The contour is one of the most important image features that are repeatable versus a lot of image transformations except for occlusion. Therefore, we can rely on this feature to detect our corners that are also repeatable as shown before. These corners are grouped according to their contour. This is also an efficient property that relates the corners on the same contour. Without this property, we must use another technique to group the corners like RANSAC [23] or other. Since the target 2D

object/shape can be composed from several contours, the matching between two object's images is based on matching of the corresponding contours.

The algorithm training phase consists of presenting to the robot a real image that contains clearly the target shape/object to detect as shown for the star shape in Figure 4.24. The straight edges are extracted and then the corners are reported. Thus, a descriptor called "Training Descriptor" is formed. It is a vector of the ordered contour's corners defined by their angles and LRs.

Fig.4.24. Training Image.

After the construction of the Training Descriptor, the robot starts searching for the object. So, it enters the test phase. In each acquired test image, it will detect the corners of every contour presented in the image. A test descriptor is formed per contour. Then, the matching process starts. It will match each test descriptor with the training one. Two descriptors are matched if their ordered corners are matched in Angle and LR. Figure 4.25 (a) shows a test image, Figure 25 (b) shows the different contours in the test image and their edge corners and Figure 4.25 (c) shows the corners of the matched contour only. So, the matched shape is the star shape even with the big scale change between the star shapes in the training and test images. The overall matching algorithm is shown in Figure 4.26.

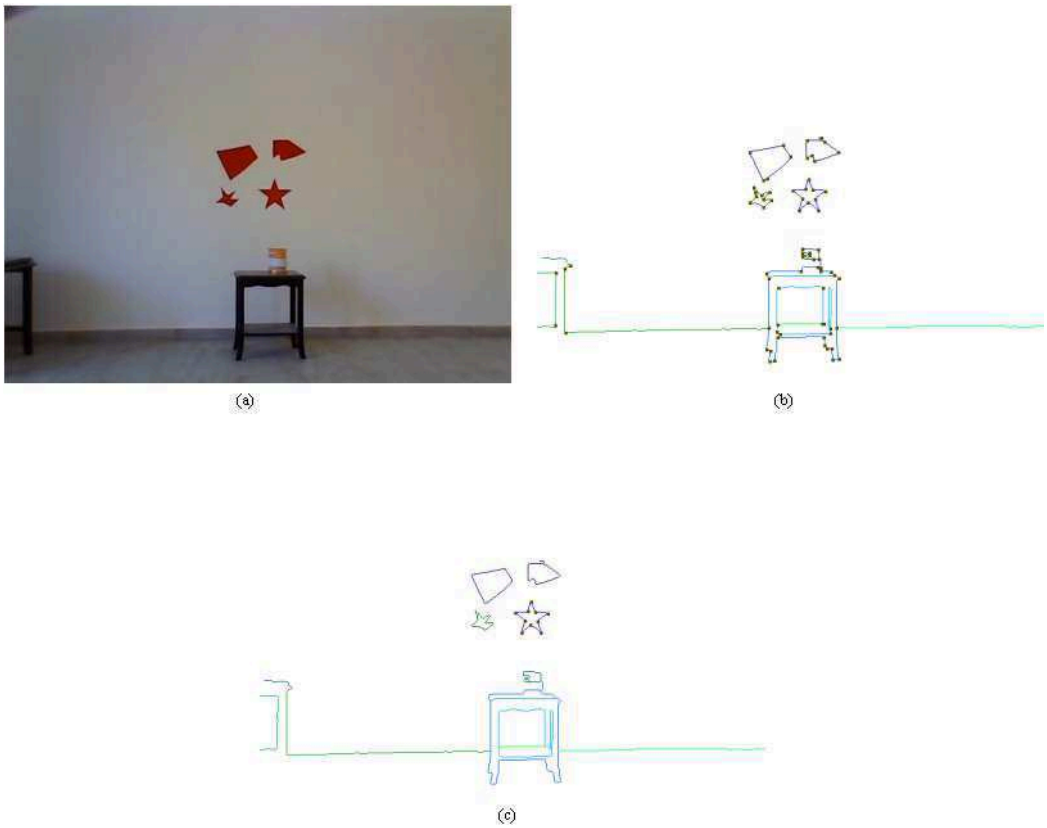


Fig.4.25: (a) Test image. (b) Corresponding image of contours with corners shown on the matched one.

- *For an input training and test image:*
  - *Detect all edge point.*
  - *Link them into contours.*
  - *For each contour //The training image corresponds to one contour.*
    - *Extract the straight edges.*
    - *Detect the edge corners characterized by the length ratio of their adjacent segments and by their angles.*

Fig.4.26. The corners descriptors.

- *For each test contour //belongs to the test image.*
  - *For each corner*
    - *For each corner from the training contour*
      - *Select the couple of corners having the closest length ratios and angles. If the ratio of their length ratios is between 0.9 and 1.1 and the absolute difference of their angles is less than 0.3*
        - *Mark this couple as most likely matched corners.*
  - *Order the corners on the training and test contours according to their linking order with respect to the marked corners respectively.*
  - *For each corner (other than the marked one) from the ordered list of corners on the test contour*
    - *For each corner (other than the marked one) from the ordered list of corners on training contour*
      - *If the ratio of their length ratios is between 0.9 and 1.1 and the absolute difference of their angles is less than 0.3.*
        - *The two corners are matched.*
- *The test contour with the largest matches is the matched contour.*

Fig.4.27. Matching a test contour to a training contour using their corners descriptors.

In the algorithm shown in Figure 4.26, the extraction of corners descriptors (length ratio of adjacent segments and angle) is explained. The algorithm shown in Figure 4.27 explains how we can match two contour, training and test, by matching their corners following their linking order. The ratios of the lengths of two corners are considered equal if their ratio is between 0.9 and 1.1. In addition, their angles are considered equal if their absolute difference is less than 0.3.

4.6. Detecting dominant corners from edge corners: Polygonal Approximation

The corners, reported by the suggested corner detector, are characterized by the lengths of its two sides (straight edges) and by its angle between its two sides. The corners for the chromosome shape in Figure 4.28 (a) are shown in the linked edge image in Figure 4.28 (b) with their angle directions.

The main goal of this function is to introduce an operator that is able to extract a polygon that best approximate an object's contour. The vertices of this polygon are selected among the set of edge corners detected so far. The best approximation is obtained by selecting only strong corners called "Dominant Corners". For this

purpose, we have introduced first a corner strength measure and then developed a technique to remove iteratively the corners with weakest strength measures. This is described by a function called "Iterative Corner Suppression".

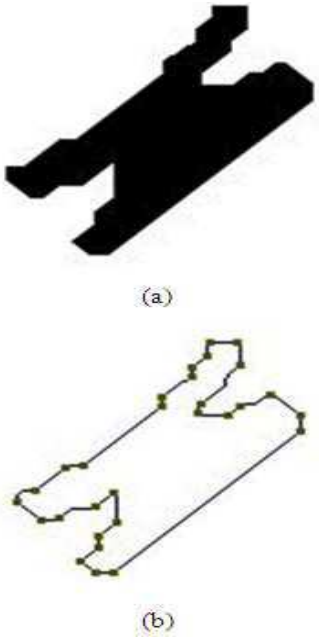


Fig.4.28. Detected corners for a chromosome shape: (a) Chromosome shape, (b) Linked edge image.

4.6.1. The corner strength measure

Some parameters should be defined to illustrate the corner strength measure:

- *LISE*: is the Local Integral Square Error between a polygon's segment and its approximated edge part. The *LISE* relative to a polygon's segment is given by,

$$LISE = \sum_{k=1}^{n} {d_k}^2 \tag{4.1}$$

Where *n* is the total number of edge points on the approximated edge part and *d* is the distance between an edge point and the polygon's segment. Figure 4.29 shows an edge part in black approximated by a polygonal segment in grey. The *LISE* is the sum of squared distances d_k of edge point P_k (moving from P_1 to P_n) relative to the polygon's segment.

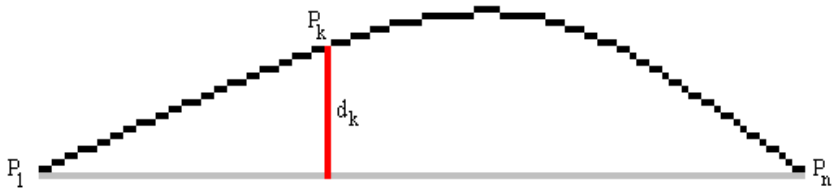


Fig.4.29. Illustration for *LISE* measure.

This measure is proportional to the area of the region bounded by the edge part and the approximating segment.

- *GISE*: is the Global *ISE* and it is equal to the sum of *LISEs* of polygon's segments. Since the *LISE* of a segment is proportional to the local area limited by the corresponding segment and edge part, the *GISE* is also proportional to the global area which is the sum of all local areas.
- *LISEV*: Local *ISE* Variation due to the removal of a corner from the list of polygon's vertices. This measure reflects the corner strength. Consider the contour shown in Figure 4.30 and its three corners P, Q and R. A, B and C are the areas of the regions bounded by edge parts and corresponding polygon's segments. The *LISEV* due to the removal of corner Q is proportional to the difference between the new area C (after removal) and the sum of the two areas A and B (before removal). The strength of a corner is proportional to this measure.

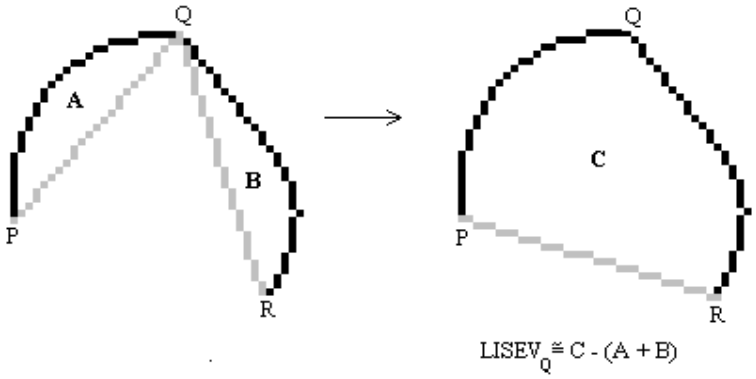


Fig.4.30. *LISEV* calculation.

4.6.2. Iterative corner suppression

Starting from the original set of detected edge corners on a contour, the goal of the iterative corner suppression technique is to obtain a smaller set of corners, called

"Dominant Corners" or "DCs" that form the vertices of a polygon that best approximates the contour.

The algorithm efficiency is compared in chapter 5 to other existing polygonal approximation algorithms detailed in chapter 3. They have eliminated their corners until a given compression ratio CR is met:

$$CR = {}^{n}/_{nc} \tag{4.2}$$

where *n* is the number of shape's edge points and *nc* is the number of selected corners that form the polygon vertices. In order to compare, we have followed in this chapter the same stopping criterion.

At a given *CR*, the objective function is to minimize the global integral square error (*GISE*). For a particular shape as in Figure 4.31, since *n* is constant *CR* becomes a function of *nc* only. Therefore, the problem is limited to obtain the minimal possible ISE for a given number of dominant corners *nc*. In general, the entry of the algorithm is the parameter *CR*, *nc* will be calculated automatically for every contour.

Figure 4.31 shows the approximated polygon of the chromosome shape used in Figure 4.28 at various *nc*. The global ISE decreases while *nc* increases.

The ICS algorithm is shown in Figure 4.32.

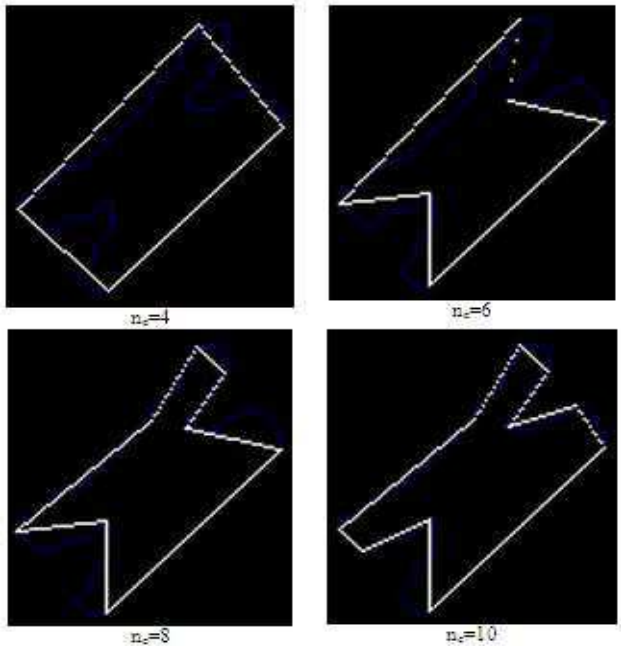


Fig.4.31. Polygonal approximation at various *nc*.

- *For a given contour of n edge points and m corners, select all the detected corners as polygon's vertices following the linking order.*
- *The number of dominant corners nc = m.*
- *Iterate until the current CR = n/nc becomes greater than or equal to a specified threshold.*
  - *For each current corner "Corc" from the list of selected corners:*
    - *Find the previous and next selected corners "Corp" and "Corn".*
    - *Calculate the LISEV due to its removal: $LISEV_i$ = new segment LISE – old segments LISEs.*
    - *If any of the previously removed corner located between "Corp" and "Corn" has a greater LISE variation ($LISEV_i$) than reselect this corner, remove the current corner "Corc" from the list of vertices and set the final LISEV as $LISEV_i$.*
  - *Remove the corner that corresponds to the minimal LISEV from the list of selected corner.*
  - *Decrement nc by 1.*
- *Calculate the global ISE for the entire polygon which is the sum of LISEs of introduced segments.*

Fig.4.32. Iterative corner suppression algorithm.

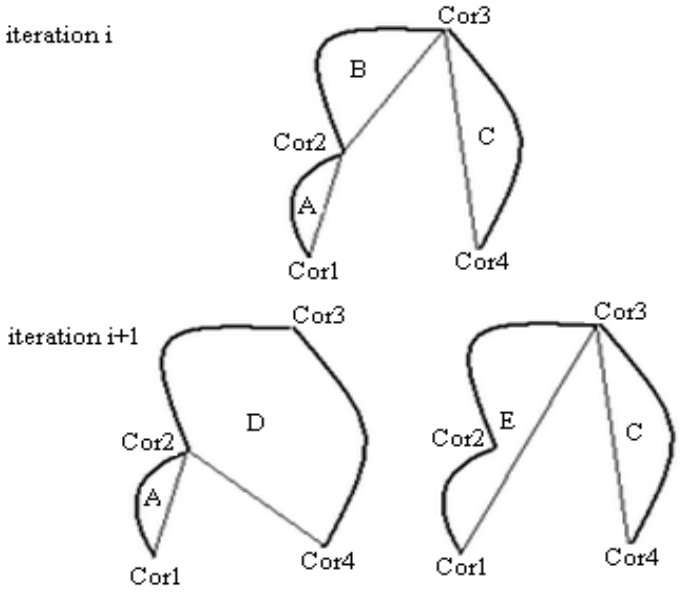


Fig.4.33. Corner suppression.

Starting by considering all the corners as polygon's vertices, the idea behind the ICS algorithm is to decrease the number of vertices iteratively of the approximated polygon. Here, the suppressed corner (vertex) is the one that its suppression will cause the minimal increase to the current global ISE and this will ensure the selection of the optimal polygon at every iteration (at every value of *nc*). Consider the case presented in Figure 4.33. We have four selected corners at iteration *i* so there are three polygon's segments with their corresponding LISE represented by the areas A, B and C. The idea is to calculate the LISEV caused by the removal of one of the two corners Cor2 or Cor3, for example, at iteration *i*+1. D is the area that represents the LISE of Cor3 and E is the one that represents the ISE of Cor2. The removal of Cor3 will add a new polygon's segment [Cor2Cor4] and eliminate two others [Cor2Cor3] and [Cor3Cor4]. Therefore, we can write

$$\mathrm{LISEV}_3 = \mathrm{D} - (\mathrm{B} + \mathrm{C}) \tag{4.3}$$

Same procedure will apply for calculating the LISEV caused by the removal of Cor2 with:

$$\mathrm{LISEV}_2 = \mathrm{E} - (\mathrm{A} + \mathrm{B}) \tag{4.4}$$

If LISEV_3 is smaller than LISEV_2 than Cor2 will be removed otherwise Cor3 will be removed.

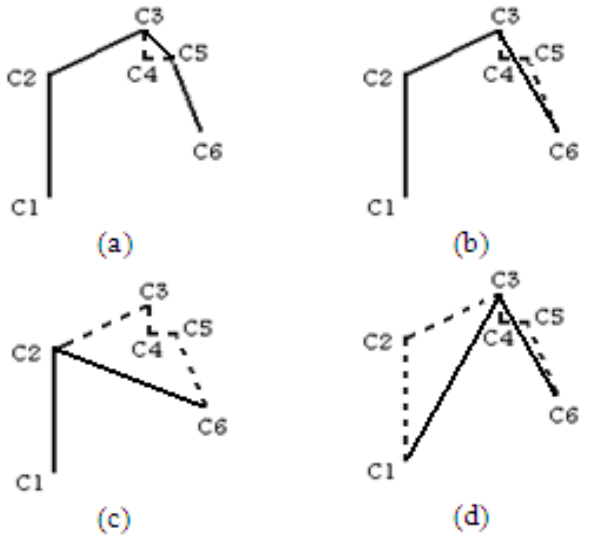


Fig.4.34. Corner reselection.

While the corners are suppressed one after the other, the LISEV of a corner that depends on the current corner "*Corc*" and the directly selected neighboring corners (previous one "*Corp*" and next one "*Corn*"), may change since "*Corp*" or "*Corn*" may also change. So we should always update and then check the LISEV of even the suppressed corners to ensure the optimality of the suppression at each iteration. This can be illustrated using the list of corners shown in Figure 4.34 where all the five corners are selected initially. This is a tested case by the algorithm where C4 (Figure 4.34 (a)), C5 (Figure 4.34 (b)) then C3 (Figure 4.34 (c)) are suppressed iteratively first and the remaining polygon has only C1, C2 and C6 as vertices. Now if we calculate the LISEV of removing C2 and that of removing C3, C4 and C5 taking C1 as "*Corp*" and C6 as "*Corn*", we find that the LISEV of removing C3 is the greatest one. So we should suppress C2 and reselect C3 (Figure 4.34 (d)). Then, the LISEV resulting from adding [C1C6] is that of removing C3 and it will be compared with the LISEV of removing the remaining selected corners existing on the whole contour to deselect the corner with the minimal one.

4.6.3. Towards an automatic stopping criterion

Using the *CR* as a stopping criterion works well to compare the efficiency of our proposed polygonal approximation algorithm with others algorithms on the same shape. However, when we intend to use this stopping criterion to generate the DCs on the same shape but at different resolutions, this will generate different number of DCs for each resolution. Therefore, to generate the same number at any resolution, one should adjust the *CR* each time which is not feasible. Therefore, an automatic stopping criterion that can generate nearly the same number of DCs at any resolution is needed.

In chapter 6, we present our work on an image registration application using the DCs. However as discussed before, using the *CR* as a stopping criterion will not give the same number of DCs on corresponding contours in the two images to register. So, we have suggested a new automatic stopping criterion to select the same number of DCs. In addition, we present a shape recognition application to recognize shapes at different resolutions using the DCs. In this application, we see the benefit of the automatic selection of DCs as a stopping criterion compared with the *CR*.

Another possibility to benefit from DCs is a shape recognition application to recognize shapes at different resolutions (at different scales) by using the number of DCs as a stopping criterion. If we have two images of the same shape but at different resolutions (at different scales), the number *Nd* of DCs on the low resolution shape's contour (smaller shape) is smaller than that on the high resolution shape's contour. Due to the repeatability of the DCs over scale, if we choose *Nd* as a stopping criterion for DCs selection on the high resolution shape's contour, the obtained *Nd* DCs will be corresponding to the *Nd* DCs on the low resolution shape.

5

Experimental Results on Corners and Dominant Corners

This chapter shows the experimental results on the two new proposed image features: edge corners and dominant corners. Section 5.1 shows the corners experimental results and Section 5.2 shows those of the dominant corners.

5.1. Experimental results on edge corners

Experimental results are presented in three parts using three sets of images. The first experiment uses a synthetic image, the second one uses a newly introduced simple real images database used in our robotic application and the third one uses a set of existing real images. In addition, the results of 2D object recognition are also shown.

5.1.1. First experiment: synthetic images

The first experiment using the newly proposed detector is on the test synthetic image used by SUSAN [151] and it is shown in Figure 5.1.

Figure 5.2 (a) shows the results of our newly proposed EBCD detector, which indicate the corners and their angles on the edges of the test image. They are compared to the results of the five existing operators: SUSAN, Harris, Harris-Laplace, FAST and SIFT as revealed in Figure 5.2 (b), (c), (d), (e) and (f), respectively. These operators are simulated using their MATLAB codes provided in the website [152]. The results show the ability of the proposed EBCD to identify corners even with small acute or large obtuse angles which are normally very difficult to detect.

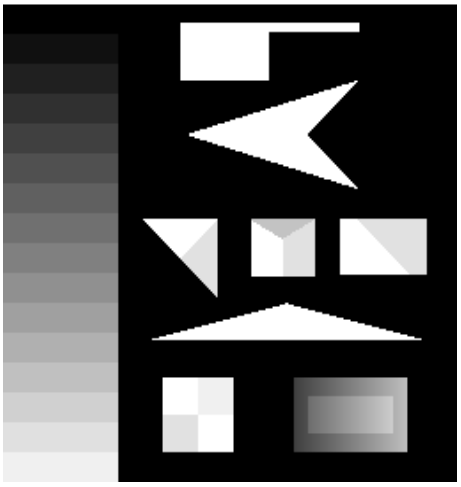


Fig.5.1. SUSAN's test image [151].

By definition, a False Positive corner is a real corner not detected and a False Negative corner is an ordinary point detected as a corner. It is shown clearly that the numbers of False Positive and False Negative corners reported by the EBCD are the smallest among all as revealed in Table 5.1.

The comparative results shown in Table 5.1 and also in Tables 5.2, 5.3, 5.4 are derived manually. From the original image, one can count the number of true corners. From the output image with detected corners, one can count the number of false and true detected corners. Note that "NA" stands for Not Applicable. This term is used for the False Negative results of the SIFT operator. Since SIFT is not introduced to detect edge corners, SIFT points that are not edge corners and that are classified as False Negative Corners will be detected in all image space. So to be fair with this detector, we have put "NA" instead of its number of False Negative Corners.

Table 5.1. Quantitative results on SUSAN's test image.

Corner Detector	#Real Corners	#False Positive Corners	#False Negative Corners	#True Detected Corners
EBCD	61	0	1	61
SUSAN	61	19	1	42
Harris	61	37	3	24
Harris-Laplace	61	45	0	16
FAST	61	58	0	3
SIFT	61	20	NA	41

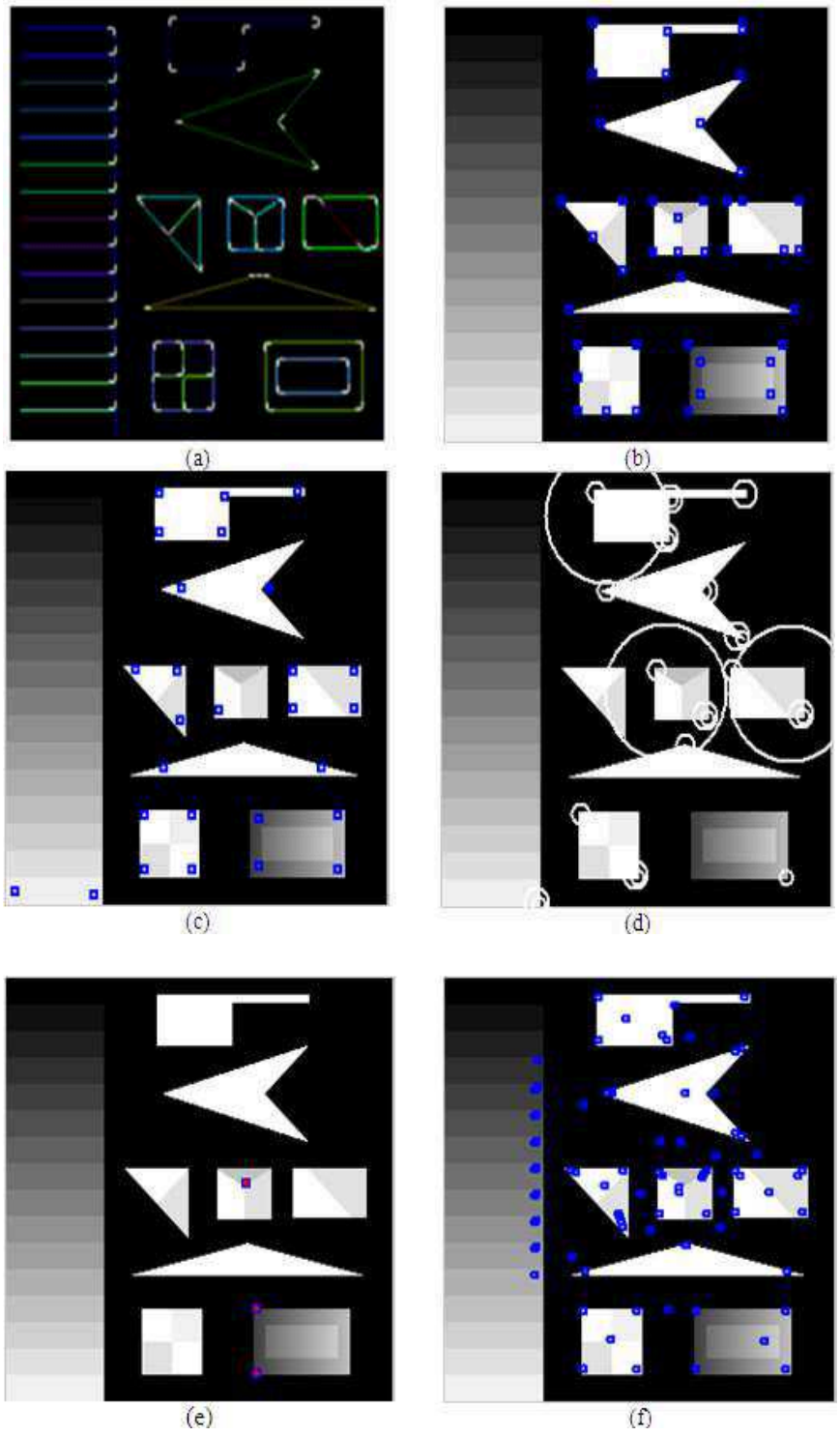


Fig.5.2. Output of tested detectors: (a) EBCD, (b) SUSAN, (c) Harris, (d) Harris-Laplace, (e) FAST, and (f) SIFT.

5.1.2. Second experiment: newly introduced simple real images database

The second experiment shows the robustness of our detector against scale, rotation variation and viewpoint change in comparison to the other operators. The test image is a real image shown in Figure 5.3 contains four different shapes with different corner angles. This image is taken manually by a webcam. The first set contains also four images that correspond to a rotation of the camera by 30^{o} for each image with respect to the camera central axis normal to the image plane as shown in the first row of Figure 5.4. The second set contains four images that correspond to four different scale levels formed by changing the distance between the camera and the image plane as presented in the first row of Figure 5.5. For each image, the distance is doubled adequately to obtain a scale reduction by two. The first row of Figure 5.6 shows the third experimental set which contains also four images of the main database taken at different viewpoint position.

Fig.5.3. Main image database of simple real images.

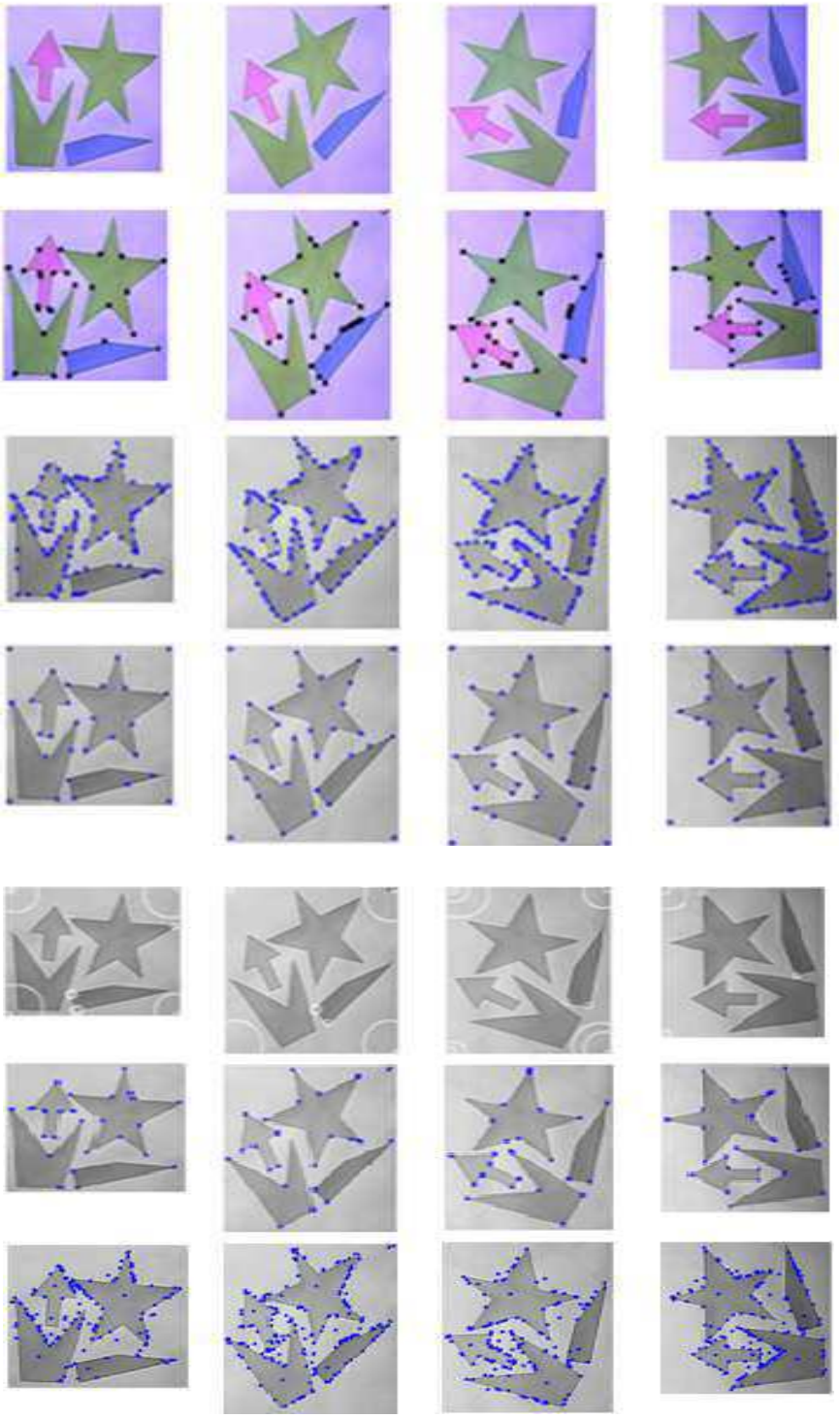


Fig.5.4. Rotation results. Row 1: original images. Row 2: Proposed detector outputs. Row 3: SUSAN outputs. Row 4: Harris outputs. Row 5: Harris-Laplace outputs. Row 6: FAST outputs. Row 7: SIFT outputs.

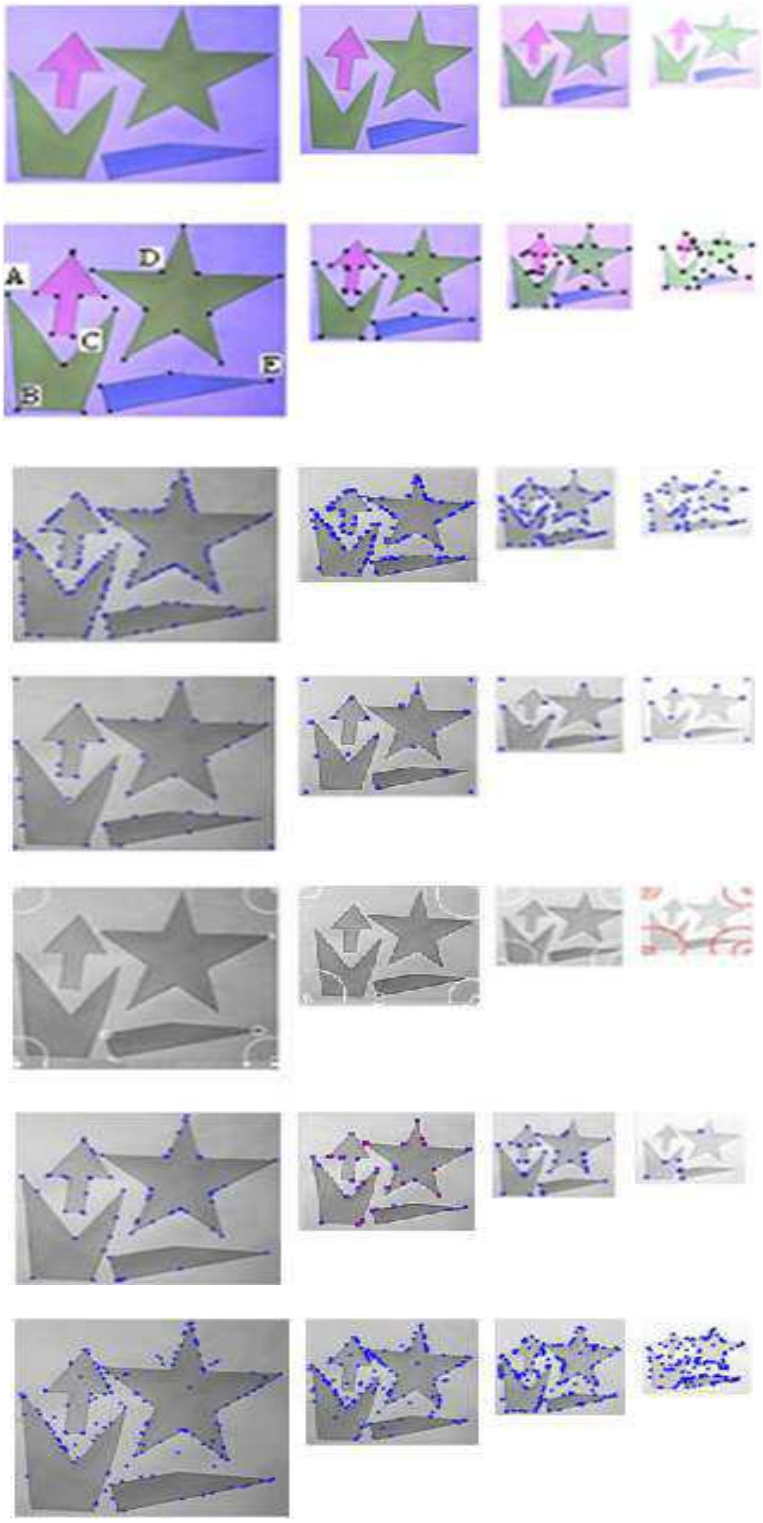


Fig.5.5. Scale variation results. Row 1: original images. Row 2: Proposed detector outputs. Row 3: SUSAN outputs. Row 4: Harris outputs. Row 5: Harris-Laplace outputs. Row 6: FAST outputs. Row 7: SIFT outputs.

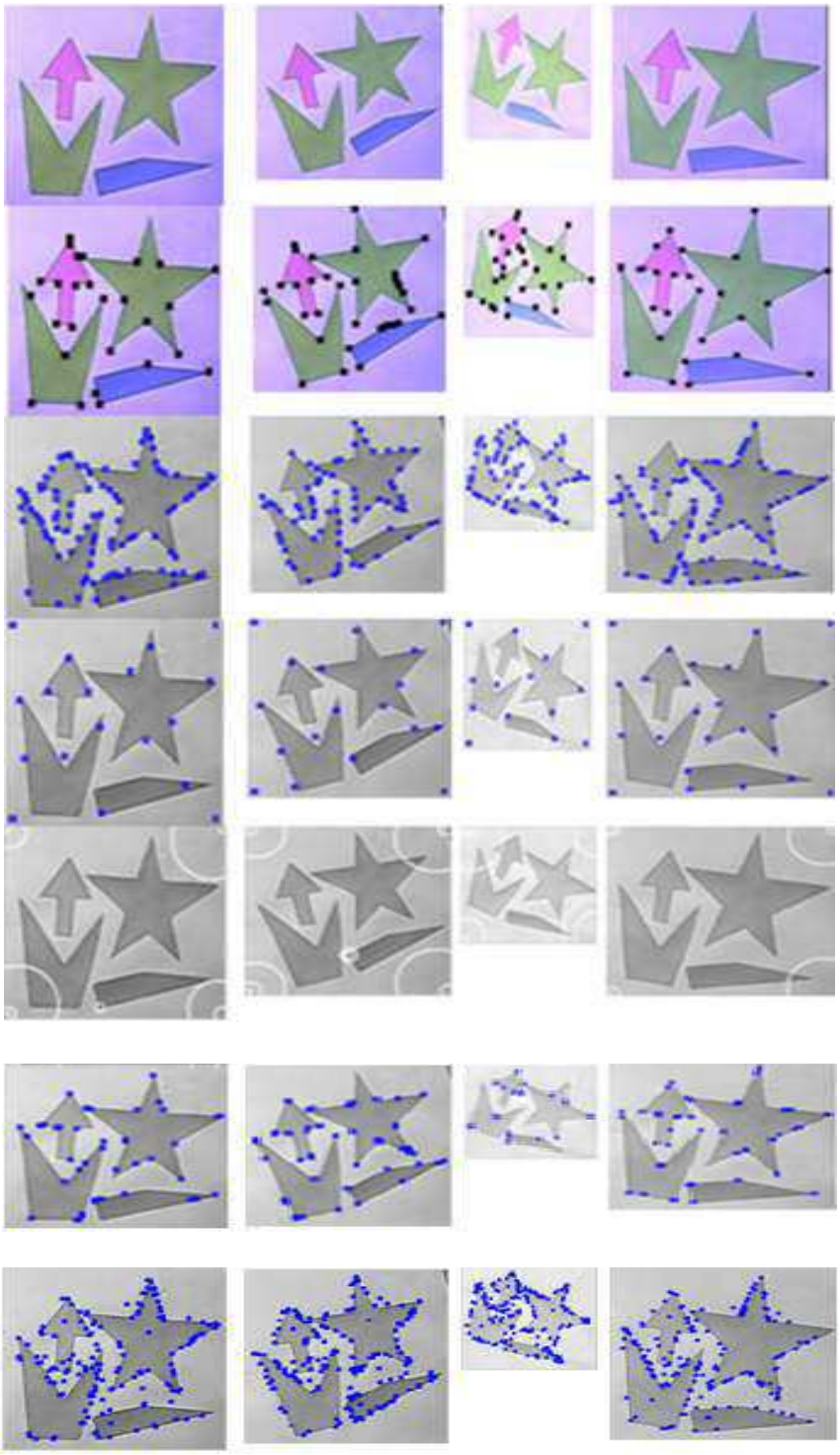


Fig.5.6. Viewpoint change results. Row 1: original images. Row 2: Proposed detector outputs. Row 3: SUSAN outputs. Row 4: Harris outputs. Row 5: Harris-Laplace outputs. Row 6: FAST outputs. Row 7: SIFT outputs.

Figure 5.4, Figure 5.5 and Figure 5.6 show the results of applying our proposed EBCD and the existing operators on the three sets. Tables 5.2, 5.3 and 5.4 contain the

quantitative results of the EBCD and the existing operators on the three sets. In Table 5.2 we show results on the image rotated by 60°. On the other hand, Table 5.3 presents results on the image scaled by half from the original one. While in Table 5.4, we discuss results on the image taken on the second viewpoint position. We can see the competitive results of our EBCD compared to others. The repeatability of the detected corners by EBCD, shown in Tables 5.2, 5.3 and 5.4, versus various image transformations makes them more practical to use in many image processing applications, especially those relying on repeatable points, than interest points.

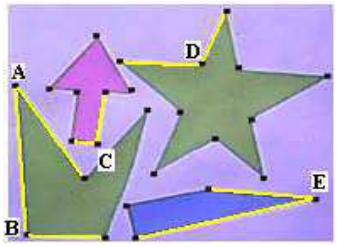


Fig.5.7. The five corners studied versus scale variation.

Table 5.2. Quantitative results on the image rotated by 60°: third column of Fig.5.4.

Corner Detector	#Real Corners	#False Positive Corners	#False Negative Corners	#True Detected Corners
EBCD	26	2	6	24
SUSAN	26	2	132	24
Harris	26	3	7	23
Harris-Laplace	26	24	6	2
FAST	26	4	0	22
SIFT	26	7	NA	19

Table 5.3. Quantitative results on the image scaled by half: second column of Fig.5.5.

Corner Detector	#Real Corners	# False Positive Corners	# False Negative Corners	# True Detected Corners
EBCD	26	3	4	23
SUSAN	26	6	77	20
Harris	26	11	7	15
Harris-Laplace	26	25	4	1
FAST	26	4	20	22
SIFT	26	15	NA	11

Table 5.4. Quantitative results on the image taken at second viewpoint: second column of Fig.5.6.

Corner Detector	#Real Corners	#False Positive Corners	#False Negative Corners	#True Detected Corners
EBCD	26	2	8	24
SUSAN	26	5	154	21
Harris	26	13	9	13
Harris-Laplace	26	22	4	4
FAST	26	1	20	25
SIFT	26	20	NA	16

Finally, the computation times of all the algorithms are investigated. The EBCD is implemented and executed using C++ compiler and other detectors are executed using MATLAB. The machine used has 1.7 Ghz processor with 512 MB RAM. In addition, we have coded one program written in C++ and MATLAB and we have found that the execution time ratio between MATLAB and C++ is nearly 70.4. The main image, shown in Figure 5.3, of size 320x240 pixels is used to explore the computation time of all detectors. Table 5.5 shows the normalized computation time results of the tested operators using MATLAB. It is shown that the EDBC requires more computation time than most other detectors.

Table 5.5. Computation time of various detectors.

Operator	EBCD	SUSAN	Harris	Harris Laplace	FAST	SIFT
Time in seconds	7.1315	0.047	3.297	47.391	1.016	1.344

5.1.3. Third experiment: real images

Our third test is conducted on three real images taken from the literature [153] as shown in Figure 5.8. In Figure 5.9 we present the corners detected by our proposed EBCD. Figures 5.10, 5.11, 5.12, 5.13 and 5.14 show the results of other detectors. The numbers of true detected corners, False Positive corners and False Negative corners for the rectangles and house images (Figure 5.8) are shown in Table 5.6 and Table 5.7 respectively.

Fig.5.8. Original real images [153].

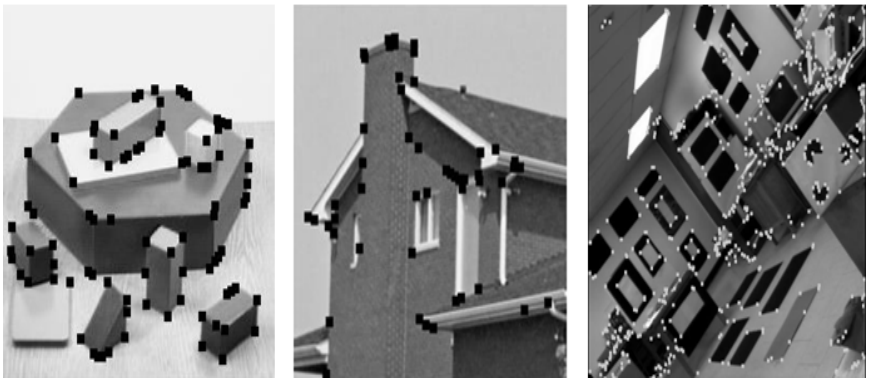


Fig.5.9. Detected corners by our proposed EBCD.

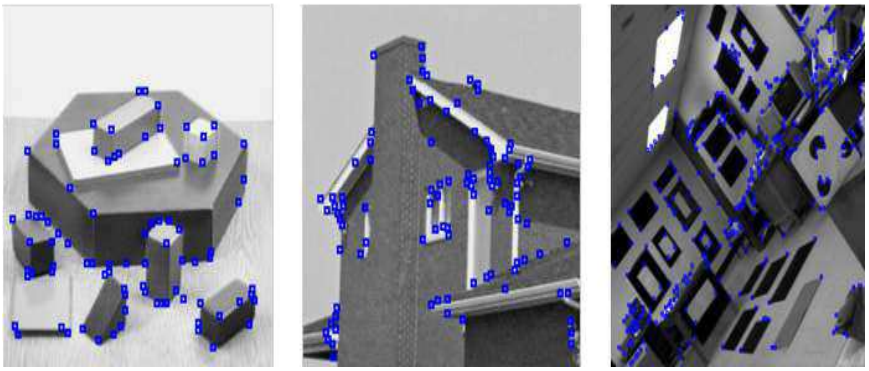


Fig.5.10. Detected corners by SUSAN detector.

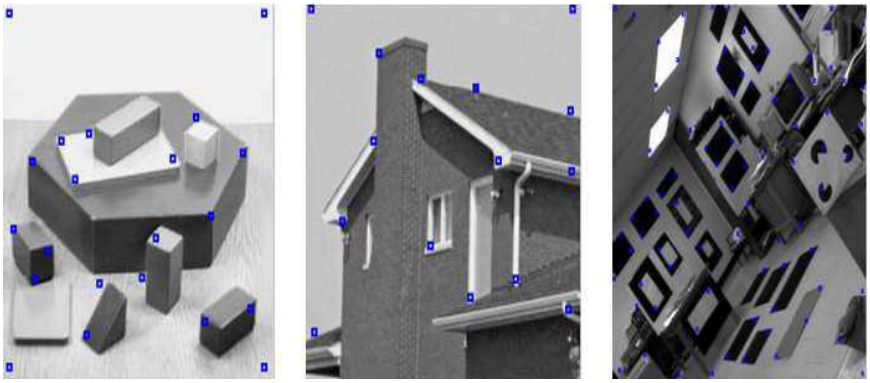


Fig.5.11. Detected corners by Harris detector.

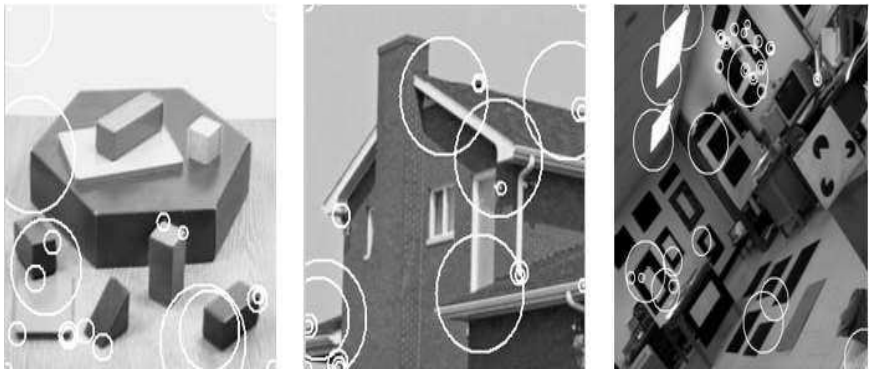


Fig.5.12. Detected corners by Harris-Laplace detector.

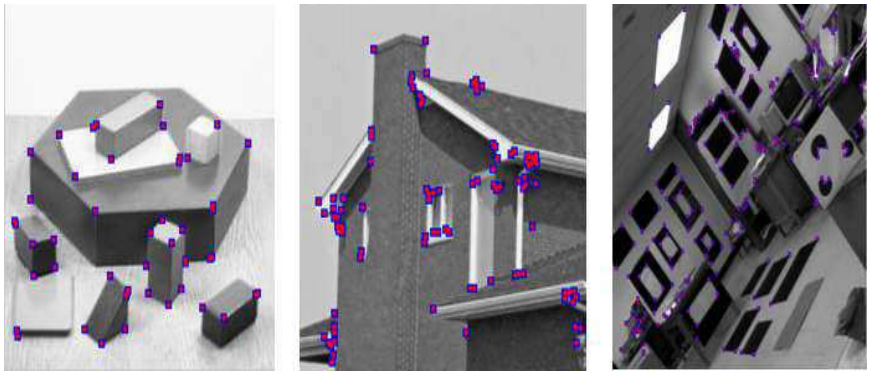


Fig.5.13. Detected corners by FAST detector.

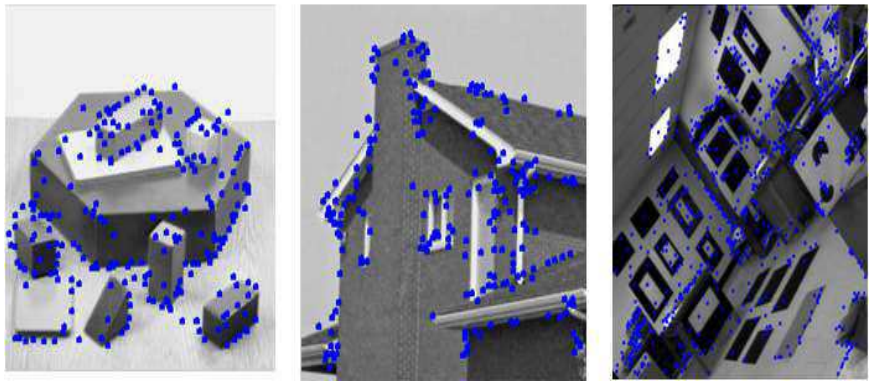


Fig.5.14. Detected corners by SIFT detector.

Table 5.6. Quantitative results on the rectangles image.

Corner Detector	#Real Corners	#False Positive Corners	#False Negative Corners	#True Detected Corners
EBCD	61	9	31	52
SUSAN	61	11	32	50
Harris	61	46	6	15
Harris-Laplace	61	52	9	9
FAST	61	25	5	36
SIFT	61	17	NA	44

Table 5.7. Quantitative results on the house image.

Corner Detector	#Real Corners	#False Positive Corners	#False Negative Corners	#True Detected Corners
EBCD	35	7	20	28
SUSAN	35	8	24	27
Harris	35	26	6	9
Harris-Laplace	35	28	8	7
FAST	35	9	69	24
SIFT	35	10	NA	25

From these quantitative results, we can deduce the following observations. First, the number of false corners reported by the EBCD is very small compared to that of the true ones. The numbers of true and false detected corners are derived visually from the figures. Second, the corner detection is nearly independent of the contrast of the regions surrounding the corner. Corners are detected on edges between white-black regions, black-grey regions, grey light-grey dark regions and even between regions of very low contrast. Third, corners of various angles from very acute to obtuse ones are detected. Fourth, the detected corners are of variety of angles starting from small acute angles to big obtuse ones and also they are of various sides lengths. Finally, the number of False Negative corners, especially with SIFT detector, does not reflect the true efficiency of the detector. SIFT detector is not built to detect specially edge corner points. It detects all scale invariant points. Therefore, this number is not applicable for all experiments.

5.2. Experimental results on dominant corners: Polygonal Approximation

Our proposed algorithm is tested on three different shapes shown in Figure 5.15: (a) Chromosome, (b) Leaf and (c) semicircles [154]. The results obtained are compared to those presented in various papers [125-128].

Other than the ISE, a new parameter is introduced for the comparison: the weighted sum of square error WE given by

$$WE = {}^{ISE}/_{nc} \tag{5.1}$$

Since our algorithm requires a colored or grey image as an input not only a digital image or edge image while the others are tested directly on a digital image, we need a unique platform for comparison. From here, we have selected manually on the edge image derived by our algorithm the vertices of the polygon reported by each method in [125-128], then calculate the corresponding ISE and show the results in Table 5.8.

Table 5.8: Comparative Results for the Chromosome, Leaf and semicircle shapes.

Contour	Method	n_c	*CR*	*ISE*	*WE*
Chromosome (n=296)	Marji and Siy [128]	10	29.6	546.2	54.6
	Carmona-Poyato et al. [126]	12	24.6	439.9	36.6
	Masood [127]	12	24.6	302.5	25.2
	Parvez and Mahmoud [125]	11	26.9	586.9	53.3
	Our Method	10	29.6	429.5	42.9
		11	26.9	370.9	33.7
		12	24.6	302.5	25.2
Leaf (n=840)	Marji and Siy	17	49.4	2124.4	124.9
	Carmona-Poyato et al.	21	40	1396.2	66.5
	Masood	23	36.5	1203.1	52.3
	Parvez and Mahmoud	23	36.5	1264.9	55.0
	Our Method	17	49.4	2089.9	122.9
		21	40	1250.2	59.5
		23	36.5	1192.3	51.8
Semicircles (n=461)	Marji and Siy	15	30.7	789.5	52.6
	Carmona-Poyato et al.	21	21.9	490.1	23.3
	Masood	26	17.7	334.1	12.8
	Parvez and Mahmoud	14	32.9	957.2	68.4
	Our Method	14	30.7	574.5	41.0
		21	21.9	347.3	16.5
		26	17.7	270.0	10.4

From Table 5.8, it can be seen clearly that our algorithm has better *ISE* and *WE* compared to others. This is due to three main reasons. The first one is the excellent location of the corners due to the efficient straight edge detector. So, the optimal polygon is the one whose vertices are selected among these corners. The second one

is the efficient technique to select the best corners as polygon vertices that will lead to the minimal error (*ISE*) at a specific *nc*. The third one is the update of the LISE, at each iteration, even for previously suppressed corners. This feedback will ensure that, when eliminating a corner or selecting a new segment, the maximal LISE is set for this segment. This fact will show the real LISEV caused by selecting this segment and as a result will ensure the selection of the segment with the real minimal LISEV.

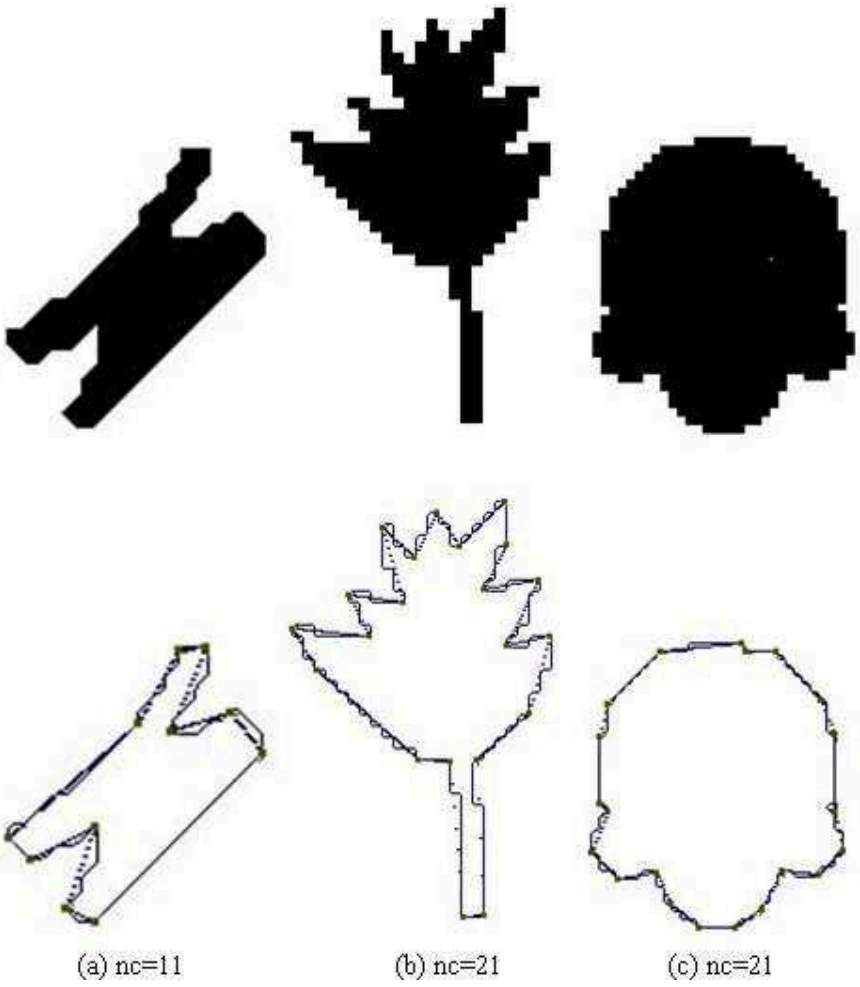


Fig.5.15: Tested shapes and their polygonal approximations at a particular *nc*.

For a real image that contains more than one shape or contour, *nc* cannot be fixed and considered as an input for the algorithm since existing contours maybe best approximated by polygons of different number of vertices. In other words, *nc* cannot be fixed for all contours. Here, *CR* or *WE* plays the big role and must be used both or at least one of them as inputs. By specifying a threshold for the *WE* parameter used as an input, the role of a polygonal approximation algorithm becomes to find the greatest

nc per contour that corresponds to the greatest *WE* less than the specified threshold. Here the *CR* is an output. On the other hand, by specifying a threshold for *CR* used as an input, the goal still to find the greatest *nc* per contour, that corresponds to the minimal *CR* greater than the threshold. In this case, *WE* is an output. So the choice of selecting which parameter or maybe both depends on the specific application.

Finally, Figure 5.15 shows the polygons approximating the tested shapes at a given *nc*. It can be seen clearly that the results are very precise.

6

First Application Using Corners: Image Registration

6.1. Introduction: Method Outlines

This chapter will discuss our first application using the edge corners and especially the DCs. This application is an image registration application where the aim is to model the transformation that related the source and sensed images to register. Therefore, the suggested registration technique is targeted mainly to register images where the deformation between them is well approximated by an affine model. To meet this constraint, the acquisition time interval between the two studied images should be small. A suggested application is to estimate the motion of a camera (e.g. drone's camera) acquiring a video sequence of images to monitor the traffic on a road. The goal is to estimate the motion of moving small targets, e.g. cars, on the road. We have published our image registration application in [52, 199].

Some precise definitions are required in this chapter:

1. Primitive: it is a set of image features like interest points, level lines, edge corners, etc that form a specific shape providing some invariant measures relative to the studied transformation model.
2. Invariant measure: An invariant measure is a quantity, resulting form the evaluation of an algebraic expression, derived form the primitive and keeping invariant under the studied transformation.
3. Transformation model: it is the affine transformation that maps the sensed image to the source image correctly.

Our goal in this chapter is to define new primitives that have invariant parameters under affine transformation. We need first very repeatable image interest points. Our DCs meet this constraint as shown in §6.5. In addition, this transformation has more than one invariant measure, as detailed in chapter 3, which should be followed to form a primitive. One of these measures is the ratio of areas (Figure 6.4). For this reason, our primitive is a set of four non collinear DCs located on the same contour. The primitive invariant measure is the ratio of two triangles areas drawn by these four DCs. Note that the illumination invariant condition is not required, since the image features used here are the edges that have normally a strong immunity against illumination variation.

After the construction of the primitives in both source and sensed images, each primitive in the source image searches for its best match in the sensed image. The detected couple will vote for the affine model that relates them. After the voting process, the model that gets the highest vote is selected as the target model approximating the motion between the two images.

Our main contribution is in the nature of the detected primitives from one side and the efficiency of the detected points (DCs) used in the primitive construction. Most registration techniques, based on primitives, form their primitives in both images focusing mainly on specific shapes that have some invariant properties without taking into consideration the repeatability of these primitives [33]. Therefore, a large number of unrepeatable primitives will be introduced in both images. This in turn will introduce a lot of wasted time in the primitive construction and voting phases. In addition, unrepeatable voters (primitives) can hamper the solution and can even lead to an incorrect one especially when they are incorrectly matched together in both images. Our image registration results are compared in §6.5 with those of existing techniques in the literature explained in chapter 3.

The main steps of the method can be summarized as follows:

(i) Automatic selection of DCs: On a given contour, the DCs are selected among the original set of edge corners. It is not efficient to use the *CR*, explained in chapter 5, as a stopping criterion due to two reasons. The first one is that it will lead to a lot of non corresponding DCs on two corresponding contours. The second one is the need for an automatic criterion to select DCs. Therefore, we have suggested and used our automatic selection described in section 6.2.

(ii) Primitive construction: A primitive is a group of four consecutive Dominant DCs located on the same contour. Since the ratio of areas is one of the invariant parameters against affine transformation, we have taken the ratio of areas of two triangles formed by two selected triplets of DCs in one primitive as an invariant measure. This is described in section 6.3.

(iii) Primitive matching and model estimation: When matching two primitives, one from the source image and the other from the sensed one, two parameters are taken into consideration. The first one is the ratio of areas while the second

one is the corner's angle directions difference. These two parameters are shown invariant under affine transformation. This is described in section 6.4.

One can add a fourth step which is image resampling to align the source image and the transformed sensed image. The stages of the algorithm are shown in Figure 6.1. The image registration experimental results are provided in section 6.5.

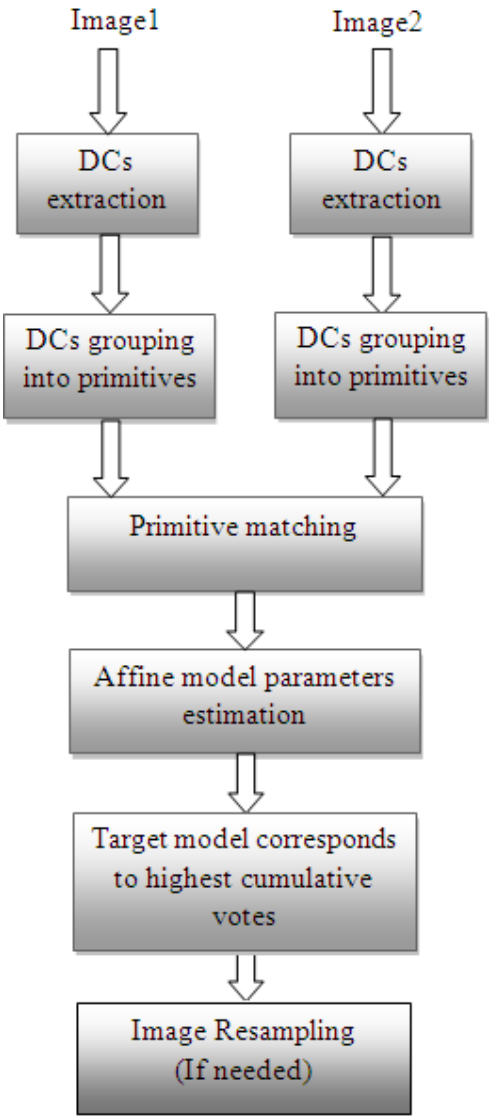


Fig.6.1. General Overview of the suggested algoritm.

6.2. Automatic selection of dominant corners

In Chapter 5, we have explained the function "Iterative Corner Suppression" that detects the DCs on a given contour given a certain compression ration *CR*. In real

images, experimental results have lead to bad selection of corresponding corners using the *CR* factor or in other terms different numbers of DCs are generated on corresponding contours in the source and sensed images to register (as shown in section 6.2.2). To illustrate this point, assume that two images of the same scene are taken but at different viewpoints. Non corresponding edges could appear. Even corresponding edges in the two images may have different number of corners. Therefore, we cannot rely on a compression factor *CR* as a stopping criterion because it will lead to a lot of non corresponding corners. On the other hand, even corresponding corners may have different *ISE* values if the scaling factor relating the two images is relatively high. Thus, we need another stopping criterion that solves this issue.

From chapter 5, the basic measure according to it a corner could be removed is the *LISEV* (reflecting the corner strength) that is added iteratively to the *GISE*. The affine transformation has the ratio of areas an invariant parameter [44]. So we can state that any two *LISEVs (LISEV1 > LISEV2)* corresponding to two corners on the same contour, will remain in the same order under an affine transformation. Thus, their order is invariant with respect to an affine transformation. We expect that only corners with corresponding *LISEVs* will show up in an image and its transformed one since the elimination of a corner is based on its *LISEV* value compared to others *LISEVs*. On the other hand, taking the ratio r of the current *GISE* with respect to the initial one $GISE_0$ is a good measure that can be used as a stopping criterion.

$$r = \frac{GISE}{GISE_0} \tag{6.1}$$

While eliminating the corners one after the other, *GISE* will increase. Thus, we can stop the elimination automatically when this ratio exceeds a fixed threshold. This way, even when the scaling parameter between the two images is considerable, we can still obtain only the corresponding DCs in both images.

6.2.1. The algorithm

The algorithm of iterative corner suppression already presented in Figure 4.32 is updated to the one presented in Figure 6.2.

- *Select all the detected corners as polygon's vertices following the linking order.*
- *The number of dominant corners nc = m.*
- *Calculate $GISE_0$.*
- *Repeat until the current r = $GISE/GISE_0$ becomes greater than or equal to a specified threshold.*
  - *For each current corner "Corc" from the list of selected corners:*
    - *Find the previous and next selected corners "Corp" and "Corn".*
    - *Calculate the LISEV due to its removal: $LISEV_c$ = new segment LISE – old segments LISEs.*
    - *If any of the previously removed corner located between "Corp" and "Corn" has a greater LISE variation ($LISEV_r$) than reselect this corner, remove the current corner "Corc" from the list of vertices and set the final LISEV as $LISEV_r$.*
  - *Remove the corner that corresponds to the minimal LISEV from the list of selected corner.*
  - *Decrement nc by 1.*
  - *Calculate the GISE for the entire polygon which is the sum of LISEs of introduced segments.*

Fig.6.2. Iterative corner suppression algorithm.

6.2.2. Shape recognition using the automatic selection of DCs

In this section, we study the number of DCs on the contours of two shapes related by an affine transformation using the two stopping criteria: *CR* and *r*.

Figure 6.3 shows two images of the chromosome shapes related by an affine transformation where a scale variation by a factor near to two occurs in the horizontal direction. In Figure 6.3 (a), the number of edge points is 487. We set a *CR* = 3% that leads to 14 DCs. The *GISEs* ratio *r* is in this case equal to 1.3. In Figure 6.3 (b), we use first the *CR* = 3% as a stopping criterion and we obtain 11 DCs shown on the edge image in the middle. Note that 10 DCs are corresponding to the original set of DCs. Then, we use *r* = 1.3 as a stopping criterion and we obtain 13 DCs shown on the edge image to the right. 12 DCs are corresponding to the original set of DCs. It is shown that using r as a stopping criteria leads practically to better results.

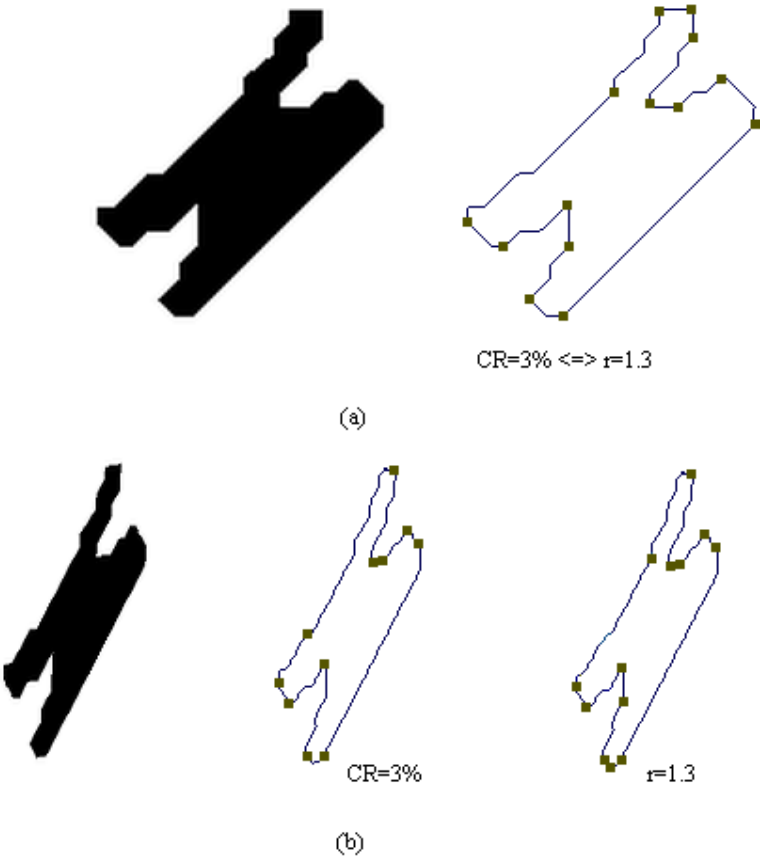


Fig.6.3. DCs on a chromosome shape using both stopping criteria.

Since our algorithm can select nearly corresponding DCs on both original and deformed shapes, we can suggest a shape recognition application based on DCs in the attempt to recognize shapes at different resolutions.

6.3. Primitive Construction

We present two options for primitive construction from n_c DCs located on the same contour. The first option is to group every four consecutive DCs. So the total number of primitives is n_c. The second option is to take four DCs out of n_c. Than the total number of primitives is the combination of four DCs out of n_c $C_{n_c}^4$. For a given contour of n_c DCs, assume that x is the number of unrepeatable DCs. The number $N1$ of repeatable primitives using first option is given by:

$$N1 = nc - x - 3, \text{where } x < n_c - 3 \tag{6.2}$$

Whereas the number $N2$ of repeatable primitives using second option is given by:

$$N2 = C_{nc-x}^{4}, \text{where } x < n_c - 3 \quad (6.3)$$

In Equations (6.2) and (6.3), if x is greater than or equal to n_c-3 than the number of repeatable primitives $N1$ and $N2$ will be negative which means that no corresponding primitives exist.

Table 6.1 provides a theoretical study of the repeatability of primitives using both options. A contour with eight DCs is introduced. The repeatability of the primitives is shown for various values of x. It is shown that the repeatability of the first option primitives is higher than that of the second option with smaller number of primitives. Therefore, if we construct the primitives based on the first option, we can ensure the best repeatability with less number of primitives in turn reduce the voting time. In the last row in Table 6.1, we assume that e is the number of unrepeatable DCs. By replacing x by e and n_c by eight in Eqs. (6.2) and (6.3), we can find the number of unrepeatable primitives in the two options.

Table 6.1. Repeatability performance using consecutive or non consecutive DCs.

	Consecutive DCs (option 1)			**Non Consecutive DCs (option 2)**		
# unrepeatable DCs: x	#Primitives *nc*	#Repeatable Primitives *N*1	Repeatability %	#Primitives C_8^4	# Repeatable Primitives *N*2	Repeatability %
1	8	4	50	70	35	50
2	8	3	37.5	70	15	21.42
3	8	2	25	70	5	7.1
e<5	8	5-e	$\frac{5-e}{8} \times 100$	70	C_{8-e}^4	$\frac{C_{8-e}^4}{8} \times 100$

The strength of a DC is proportional to its *LISEV* value. The *LISEV* is the variation introduced to the GISE due to the removal of a corner from the list of polygon's vertices. It is proportional to the area limited by the corresponding segment and edge part. Figure 6.4 (c) explains the $LISEV_3$ corresponding to DC_3. The average of the four corners *LISEVs* is set as the primitive *LISEV*. Primitives are classified according to their *LISEV*. The strongest are those who have the highest *LISEVs*. The vote of each primitive will be biased by its *LISEV* since strong primitives are formed by DCs of high *LISEVs*. This means that the corners are of high repeatability or high probability of occurrence in both images.

To illustrate the method, a polygon of ten DCs as vertices approximating the contour of a leaf image is shown in Figure 6.4 (a). The four windowed DCs are

grouped into one primitive as presented in Figure 6.4 (b). Two triangles, DC1$\widehat{DC2}$DC3 and DC2$\widehat{DC3}$DC4, are considered. The ratio of their areas *R* is the invariant parameter and will be used for matching with other primitives in the second transformed image.

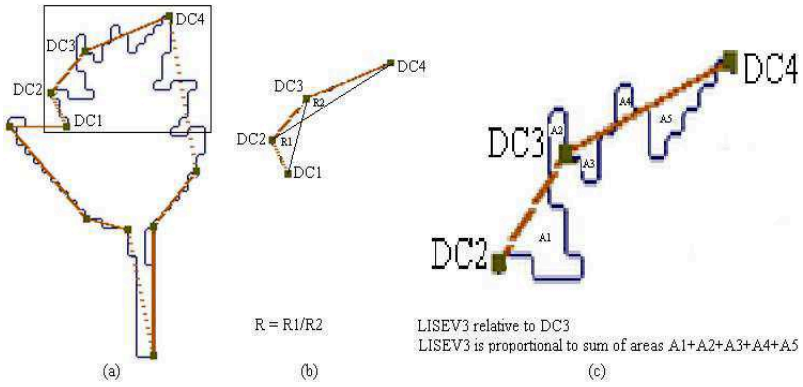


Fig. 6.4. Grouping four consecutive DCs into one primitive.

6.4. Primitive matching and model estimation

After the construction of the primitives in the source and sensed images, these primitives are compared together for matching and only the ones matched enter in the model estimation.

6.4.1. Two invariant parameters for primitive matching

Using only the ratio *R* to match two primitives will lead to a lot of false positive matches. This means that two non corresponding primitives that have similar ratio of areas are considered by the algorithm as corresponding. To minimize these false positives, we propose to add another parameter: the corners directions.

These directions are the directions of the two meeting straight edges coded in Freeman code (0,4 for horizontal right-left, 2,6 for vertical up-down, 1,5 for first diagonal and 3,7 for second diagonal) [96]. In section 6.5.1, we can show experimentally that the detected corners are very repeatable against affine transformation. More details on this experiment, how we can set an affine model and how we can setup the affine deformation can be found in section 6.5.1.

Definition1: The repeatability of a physical quantity derived from an image, e.g. corner angle, is the stability of this quantity (conserving its value) when deformation, e.g. affine deformation, is introduced to the image.

Figure 6.5 shows the matching algorithm of two primitives: one from the source image and the other from the sensed image.

- *Unmark all DCs in the sensed primitive.*
- *For each DC in the source primitive*
  - *Calculate ADD1: the Angle Directions Difference of the DC's adjacent SEs.*
  - *For each unmarked DCs in the sensed primitive*
    - *Calculate ADD2.*
    - *If(||ADD1-ADD2||<0.5)*
      - *Mark the DC in the sensed primitive.*
- *For each DC in the sensed primitive*
  - *If a DC is unmarked*
    - *Return with no match.*
- *Calculate the ratio of areas R1 in the source primitive.*
- *Calculate the ratio of areas R2 in the sensed primitive (DCs are ordered according to their matched ones in the source primitive).*
- *If(||R1-R2||<=0.1)*
  - *Return with match.*
- *Else*
  - *Return with no match.*

Fig.6.5. Matching algorithm.

6.4.2. Model estimation using Hough transform

Consider now two images related by an affinity. We adopt a group voting scheme based on Hough transform [149] for the six unknown parameters of the affine transformation. The idea behind the Hough transform (see chapter 3) is to accumulate, in a space of representative parameters, the information that insures the presence of a certain shape or model. In our case, the desired model to be found is an affine model that has six unknowns. So, the used Hough space has six parameters and each matched couple of primitives from the two images will increase by one the accumulator of the corresponding point in this space.

In fact, the division of the four axes corresponding to the affine parameters a_{ij}, shown in Eq. (3.69), is equal to 0.01 and the range is between [-2;2] (we have 400

division/parameter). Whereas the one used for the two translations axes is equal to five and the range is between [-200;200] (we have 80 division/parameter). Thus, two primitives are matched or said to be very close in the algorithm shown in Figure 6.6 if their six parameters are equal (belong to the same division). For two matched primitives, three corresponding DCs from each one are enough to calculate these parameters and give their vote for the set of the six obtained parameters. Finally, the set that gets the highest votes will be selected as the target affine model.

Let $DC_1(x_1,y_1)$, $DC_2(x_2,y_2)$, $DC_3(x_3,y_3)$ and $DC_4(x_4,y_4)$ be the DCs constructing a training primitive in the source image. In addition, let $DC'_1(x'_1,y'_1)$, $DC'_2(x'_2,y'_2)$, $DC'_3(x'_3,y'_3)$ and $DC'_4(x'_4,y'_4)$ to be the DCs constructing the corresponding test primitive in the sensed image. Since the affine matrix has six degrees of freedom, three DCs, $\{DC_1, DC_2, DC_3\}$ from the training primitive with their three matched ones, $\{DC'_1, DC'_2, DC'_3\}$, from the matched test primitive will be used to form the model. Thus, the affine transformation [44] in Eq. (3.68) can be rewritten as,

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ tx \end{pmatrix} or\ X' = \mathbf{M}.h \tag{6.4}$$

$$\begin{pmatrix} y'_1 \\ y'_2 \\ y'_3 \end{pmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{pmatrix} a_{21} \\ a_{22} \\ ty \end{pmatrix} or\ Y' = \mathbf{M}.h' \tag{6.5}$$

The affine parameters presented in vectors h and h' are calculated in Eq. (6.6) and Eq. (6.7), respectively:

$$h = \mathbf{M}^{-1}.X' \tag{6.6}$$

$$h' = \mathbf{M}^{-1}.Y' \tag{6.7}$$

The overall algorithm is fully presented in Figure 6.6. In this algorithm, we use the four points from each primitive in the parameters calculation. For each primitive, we form four sets of three DCs each by a combination three DCs out of four. Then, we let the four sets from the first primitive form four affine models with the corresponding sets from the second primitive. If two primitives are really corresponding, their DCs should have same angle directions difference. In addition, the four formed models should have nearly the same affine parameters. In this case, the two matched primitives give their vote to their average affine model. At the end of the voting process, if the difference between the number of votes of the elected model

and other models is not relatively high, we can form an additional set of primitives and thus an additional number of voters. In this set of primitives, a corner is grouped with its three nearest neighboring DCs to form a primitive. This way of grouping is also invariant under affine transformations.

- *Initialize FirstVoting to true*
- *Input the two images, image1 and image2, related by an affine transformation*
- *Extract the DCs for each image*
- *Voting Label:*
- *If FirstVoting*
  - *Group, in each image, every four consecutive DCs as primitive for first voting*
- *Else*
  - *Group, in each image, every corner with its three nearest ones as primitive for second voting*
- *For every primitive Pi extracted from image1*
  - *For every primitive Pj extracted from image2*
    - *If the four corners in Pi and their corresponding ones in Pj have the same directions difference and similar ratio R*
      - *Consider four triplets of DCs in Pi and four triplets in Pj*
      - *Let every pair of corresponding triplets in Pi and Pj form an affine model*
      - *If the four formed models are very close*
        - *A vote is added for the current model (Average model)*
- *If FirstVoting and the highest vote is much larger than other votes*
  - *The model corresponding to the highest vote is the searched affine model*
- *Else*
  - *If not FirstVoting*
    - *For each formed model*
      - *Add Second Votes to First Votes*
    - *The model corresponding to the highest vote is the searched affine model*
  - *Else //FirstVoting and the highest vote is not much larger than other votes*
    - *Set FirstVoting to false*
    - *Goto VotingLabel*

Fig.6.6. Image registration algorithm.

6.5. Experimental results

The results are shown using synthetic and real images.

6.5.1. First synthetic images set

The goal from this set is to show the repeatability of the detected DCs versus various affine transformations. This set is composed of two images: a source leaf

image and a target leaf image as shown in Figure 6.7 (a) and (b) respectively. The target image is generated from the source image using an affine model with the following values: a_{11} = 0.8800, a_{12} = 0.1907, a_{21} = -0.1008, a_{22} = 0.7728 (corresponding to $\Omega = 10^{o}$, $\varphi = 15^{o}$, $\lambda_1 = 1.1$ and $\lambda_2 = 1.3$), t_x = -150 and t_y = 15.

Due to the introduced affine deformation, some DCs in the source image will not have corresponding DCs in the target image. However, the number of non corresponding DCs is small relative to the number of corresponding ones. These two numbers in the source and target images are derived manually since the remaining number of these DCs is small. Therefore, their repeatability is calculated manually.

We set the threshold r (ratio of current *GISE* and initial *GISE* used as a stopping criterion) to be r = 5. The number of DCs extracted in the source image shown in Figure 6.7 (a) is 16 out of 98 corners while their number in the target image shown in Figure 6.7 (b) is 17 out of 122 among them 15 DCs are corresponding ones.

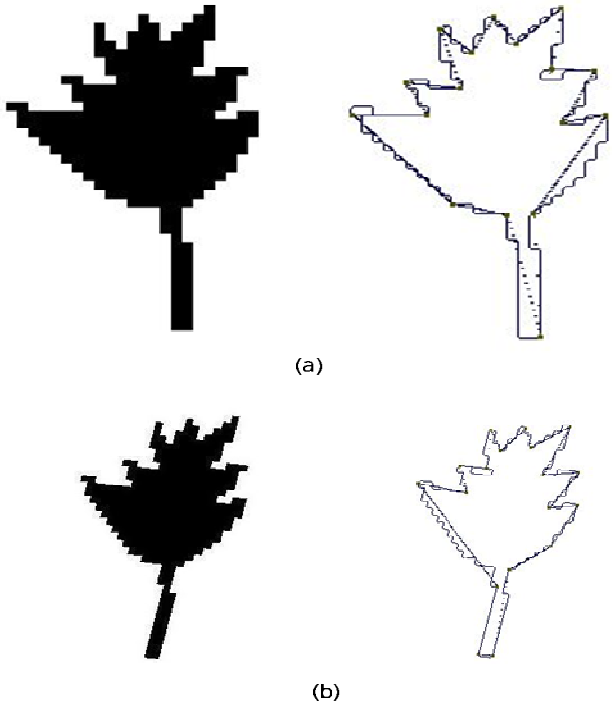


Fig.6.7. Polygonal approximation and DCs: (a) Source image, (b) Target image.

Next, we will show the repeatability of DCs under various deformations between the source and target images caused by affine transformation. The source image is the leaf image shown in Figure 6.7 (a). In each experiment we fix three out of the four

affine parameters (Ω, φ, λ_1 and λ_2) and vary the remaining one. So for each experiment we have an affine model. Than we generate the target image by applying the resulting affine model to the source image (as it is shown in Figure 6.7). The DCs are automatically detected on both contours (source and target) using the same value of stopping criterion *r*. Finally, we count manually the number of corresponding DCs on both contours. Thus, the repeatability is the number of corresponding DCs over the total number of DCs on the source contour. For example, in Figure 6.8, the repeatability of DCs and Harris corners used in [136, 137] versus the scaling factor λ_1/λ_2 is evaluated. The minimal value of the DCs repeatability is 70% at scale ratio of four which is considered high when dealing with the suggested application of video image sequence with small interval of time. Figure 6.9 shows the repeatability of DCs and Harris corners versus scaling angle φ. It is clear that the worst DCs repeatability is 75% which will lead to a high repeatability of the formed primitives. Figure 6.10 presents the repeatability versus the rotation angle Ω between the source and the target images. The worst DC repeatability is 85% which means that the rotation angle Ω has a small influence on the repeatability value. In these results, we have selected the stopping criterion as the remaining number *nc* of DCs which is 20 DCs.

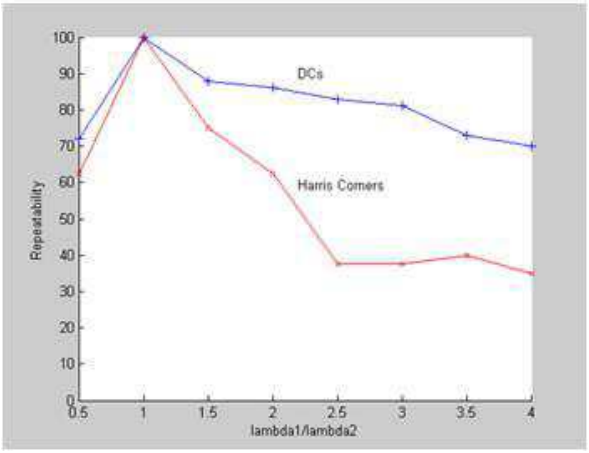


Fig.6.8. Repeatability of corners versus scaling factor λ_1/λ_2 ($\Omega = 10^o$, $\varphi = 15^o$ and $\lambda_2 = 1$).

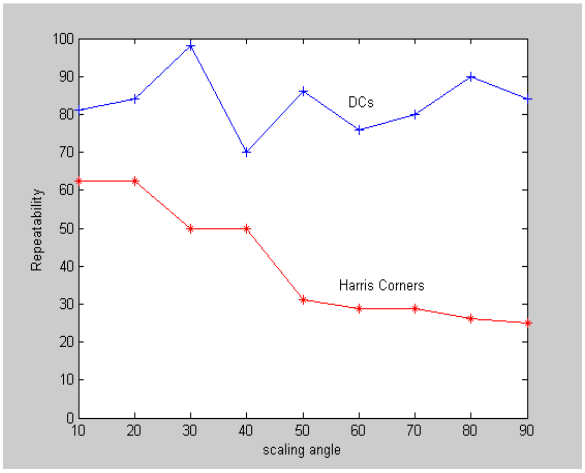


Fig.6.9. Repeatability of corners versus scaling angle φ (Ω = 10°, λ_1 =1.3 and λ_2 = 0.8).

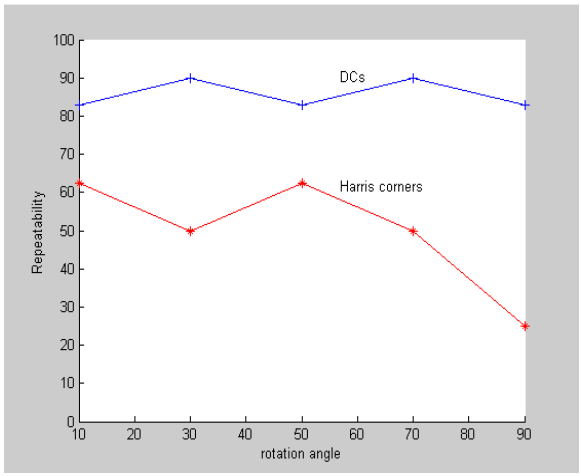


Fig.6.10. Repeatability of corners versus rotation angle Ω (φ = 10°, λ_1 =1.3 and λ_2 = 0.8).

We can even obtain higher DCs repeatability if we select for example 80% of these DCs that correspond to the highest ISEs. In Figure 6.10, it is shown that for the given values of the affine parameters (φ = 10°, λ_1 =1.3, λ_2 = 0.8 and Ω = 90°), the repeatability of DCs is 85% (17 corresponding DCs out of 20). If we select only 80% of these DCs, the repeatability becomes 93.75% (15 corresponding DCs out of 16).

It is shown clearly that the suggested DCs detector leads to more repeatable CPs (DCs) in comparison with the Harris detector used in [136, 137]. The high repeatability of the DCs leads to a highly repeatable primitives. This is an important primitive property that is necessary in order to use the primitives for voting for the affine model relating the target and source images.

6.5.2. Second synthetic images set

We used the synthetic image used by Almehio [135]. This image is shown in Figure 6.11 where we present both the source image (a) and the target image (b). The target image is generated from the source image using a known affine model. The goal is to estimate the model using our technique and than compare it with the real and Almehio models as shown in Table 6.2.

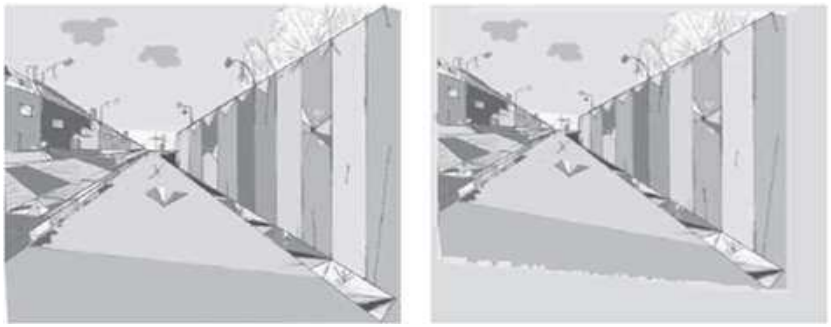


Fig.6.11. Synthetic images [135]. (a) Source image. (b) Target image.

According to the models shown in Table 6.2, we can say that the two methods give nearly similar affine parameters.

Table 6.2. Estimated affine models.

	a_{11}	a_{12}	a_{21}	a_{22}	t_x	t_y
Real Model	0.9	0	0.05	0.85	0	0
Our Model	0.91	-0.04	-0.04	0.81	2	4
Almehio Model [135]	0.88	0.008	0.05	0.85	-0.1	-7.7

6.5.3. Third synthetic images set

The goal from this experiment is to show the repeatability of the obtained primitives and the automatic reduction of their number compared to others primitives. In addition, we will discuss the time reduction and efficiency of the proposed voting scheme.

The construction of primitives using the straight edges without any constraints leads to a major problem. The problem is in the generation of a lot of non corresponding primitives in both source and target images. It can also lead to an incorrect solution. Non corresponding primitives in one image could hamper the

solution due to their possible incorrect matches (False Positive matches) to other primitives in the second image. To illustrate this, consider the two images show in Figure 6.12. Figure 6.12 (a) shows the original shape and Figure 6.12 (b) is the result of an affine transformation on the original image with the following values: a_{11} = 0.5187, a_{12} = -0.0114, a_{21} = -0.3876, a_{22} = 1.6150 (corresponding to $\Omega = 10^{o}$, $\varphi = 5^{o}$, $\lambda_1 = 2$ and $\lambda_2 = 0.6$), t_x = -150 and t_y = -70. Visually, the two images are very different. The second image is very deformed.

Figure 6.13 shows the image straight edges and their corners of the two images presented in Figure 6.13. In Figure 6.14 (a) and (b), the DCs are shown as solid points by setting the threshold r given in Eq. (6.1) to be equal to four for the source image and the transformed one, respectively. The corresponding number of DCs is equal to eight in both images.

Fig.6.12. (a) Source shape. (b) Transformed shape.

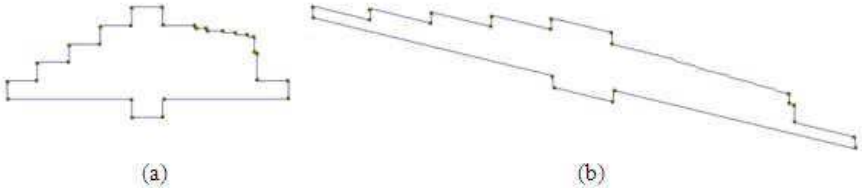


Fig.6.13. Level line segments endpoints.

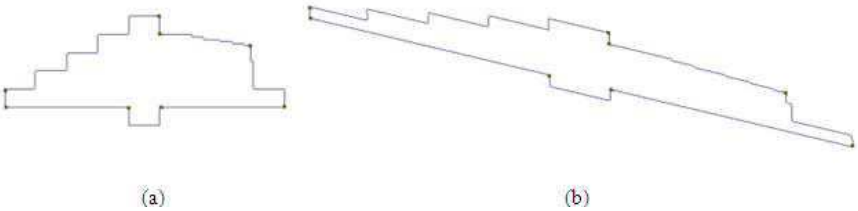


Fig.6.14. DCs on both images.

The primitives are formed using the eight DCs in both images shown in Figure 6.14 (a) and (b). Since all the eight DCs are repeatable, all formed primitives are also repeatable. In the primitive construction phase, primitives are constructed by only DCs. Therefore, week corners like the unrepeatable corners of Figure 6.13 are filtered out from the beginning and will not enter in the primitive construction phase. This way, since the CPs are equivalent to DCs only, their reduced number compared to the total number of corners leads to a reduced set of highly repeatable primitives. In addition, the voting scheme will be accurate due to the high repeatability of the primitives and also less time consuming since the number of introduced primitives is smaller and can be controlled through the input parameter *r*.

6.5.4. NOAA AVHRR real image

The goal from this experiment is to show the enhancement, in terms of CP construction and matching, that can be done to the work of Lou et al. [137] using the DCs primitives. Using the algorithm presented in Figure 6.6, two primitives, from the

source and sensed images are considered corresponding if their corresponding DCs have:

- same directions difference.
- similar ratio of areas *R*.
- similar affine models formed by the voting of every triplet from the target primitive with the corresponding triplet from the sensed primitive.

In the work of Lou [137], two CPs from the source and sensed images are considered corresponding if they have similar regions lying inside windows with proper size centered at these CPs. The similarity measure is a normalized cross correlation that reflects the degree of resemblance between the two regions.

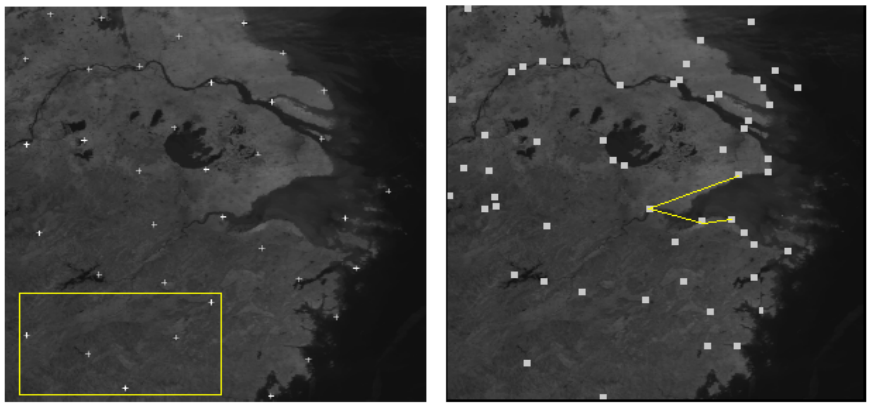


Fig.6.15. A NOAA AVHRR image. (a) CPs of Lou et al. [137]. (b) DCs as CPs.

The CPs detected by Lou are shown in Figure 6.15 (a). Consider the five CPs lying inside the yellow rectangle. If we place five windows with the same size centered at these CPs, we can notice that all the regions lying inside the windows are very similar. This is the main reason for the wrong correspondence reported by Lou et al. [137]. In Figure 6.15 (b), our proposed CPs (DCs) are presented located on the image edges. For every image contour, the DCs are selected from the set of edge corners by setting the threshold *r* to be equal to two. Consider the four DCs to the right joined by yellow segments. These DCs belong to the same edge and form one primitive. Using the three corresponding conditions for primitives matching makes this primitive very dissimilar to any other primitive. Experimentally, no other primitive from Figure 6.15 (b) have the same characteristic.

6.5.5. Real images set

For real applications, it is instructive to test the proposed image registration technique using real images. In the real interior and exterior urban scenes, there are a lot of straight edges and corners. These corners are nearly stable from one image to another. This observation makes the corners very important CPs that can be used for image registration.

Our algorithm relies on DCs extracted by approximating their contour by a polygon. Therefore, the repeatability of the DCs is based on linking the same contour in both studied images. In colored real images, some image parts containing two neighboring regions of a small contrast are encountered. The edge between these two regions is not always the same in the source and target images even if the deformation between them is relatively small. Non corresponding primitives will be generated in this case. However, still in colored real images, a lot of neighboring regions will have a high contrast enough to generate repeatable contours and thus repeatable primitives.

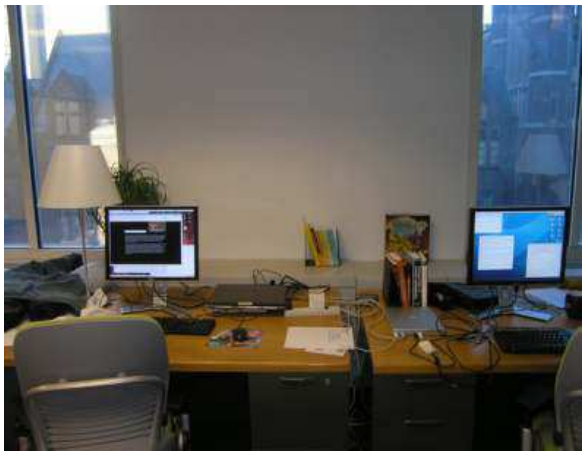
Fig.6.16. Two tested real images of a common scene.

Figure 6.16 shows two real images taken for a desk in a classical indoor scene with a camera in motion. Two matched primitives are shown in Figure 6.17 and the image of difference between the target and the resampled source image using the obtained affine model is revealed in Figure 6.18. Figure 6.18 (a) shows the second scene image (Figure 6.16 (b)), Figure 6.18 (b) shows the transformed image of the first scene image (Figure 6.16 (a)) and Figure 6.18 (c) shows the image difference between them. In Figure 6.18 (c), Each pixel's intensity is obtained by taking the absolute difference of corresponding pixels intensities in both images. Thus darker

pixels represent higher difference between the compared intensities. Also, the alignment between the two images is shown**.** In addition, it is shown that the transformed image is correctly registered and the difference between the two images (dark pixels) comes from the registration's interpolation.

For the voting process, the division of the four axes corresponding to the affine parameters a_{ij} is equal to 0.01. Whereas the one used for the two translations axes is equal to five. Let us concentrate on the primitives, composed of four yellow points, circled in both source and target images as revealed in Figure 6.19. Each of the two primitives in the source image is matched to the corresponding one in the target image. They vote for the same model with the following values: $a_{11} = 0.9682$, $a_{12} = -0.0158$, $a_{21} = 0.0106$, $a_{22} = 1.0053$, $t_x = 5$ and $t_y = 15$. These primitives are repeatable because they belong to two closed rectangular contours (computer screen contour and computer window contour) with strong contrast with their background. Therefore, the more we have shapes with strong contrast with its environment the more likely to have more repeatable primitives. In addition, many other primitives vote for the same model. They don't necessary belong to polygonal contours. The majority belongs to contours with high gradient vector norm. These contours are located between two highly contrast regions. Figure 6.18 (a) shown the target image of the scene and Figure 6.18 (b) shows the transformed image of the source image using the obtained model. In Figure 6.18 (c), the image of difference is presented. The level of brightness of a pixel is proportional to the absolute difference of the corresponding target and transformed pixels. Therefore, the darker the pixel means the greater is the difference. In real images, the number of corresponding DCs, thus the number of matched primitives, is less than those in synthetic images due to the complexity of these real images. Also, the numbers of false positive and true negative matches increase. However, using group primitive voting or the sum of all votes leads in most cases to the true model with a good difference with the nearest false one.

Fig.6.17. Two matched primitives circled in yellow in the two real images.



(a)

(b)

(c)

Fig.6.18. (a) Second scene image. (b) First scene transformed image by the calculated model. (c) Image of difference between them.

7

Second Application Using Corners: Character Recognition

In this chapter, we provide first in Section 7.1 a background on our suggested application based on edge corners: character recognition. Section 7.2 explains how to use ECs for character recognition. Section 7.3 presents the proposed scheme. Section 7.4 shows the experimental results.

7.1. Character recognition overview

This section starts by providing the problem definition in section 7.1.1. Than by presenting the Yahoo scheme, in section 7.1.2, that is based on connected characters strategy and that forms also the basis of our work. Section 7.1.3 is an overview on fuzzy logic that is introduced in the proposed scheme. Section 7.1.4 provides the contribution from this application.

7.1.1. Introduction: What is CAPTCHA?

Our goal from working on word recognition is not to be hackers to steal access to some internet application using deformed word images as entry points. However, it is rather a warning that the introduced schemes, even based on connected characters strategy, are not very robust and can be attacked. From the explanation of our technique, one can conclude with a number of improvements that can be made to the design of the CAPTCHA schemes. In addition, this application is a prior work for our main robotic application. This application is classified as a Human-robot interaction where the goal is to communicate with images containing written commands or information. This application known as Handwritten application is a future work application that should be developed.

Ahn et. al [138] have proposed several novel designs for CAPTCHA to obtain more complicated schemes and make them more difficult to recognize for computer programs. In [139], a survey on the CAPTCHA's earlier works can be found. Yan and El Ahmad [47] have reported a low cost segmentation attack on the MSN CAPTCHA's scheme with a success rate of about 60%. They have also developed another work [140] based on pattern recognition algorithms to break visual CAPTCHA schemes generated by a web service. Mori and Malik [141] have developed a shape context matching method that can identify the EZ-Gimpy image with a success rate of 92% and the Gimpy image with a success rate of 33%. Chellapilla and Simard [50] have studied various Human Interaction Proofs (HIPs)

and have developed a machine learning algorithm with a good success rate. Moy, et. al [142] have developed a correlation algorithm that estimates the distortion in a text image. They have achieved a success rate in identifying the word in EZ-Gimpy of 99% and in Gimpy-r of 78%. An attack on a simple CAPTCHA scheme using Neural Network with a success rate of 66% is reported in [143]. Gao et. al [144] have combined the segmentation and recognition to attack Yahoo HIPs and have achieved a success rate of 78%. Their pattern recognition technique is designed especially to divide and conquer the characters in a word image composed of connected characters. It starts by a preprocessing step to remove noisy pixels around the characters contours and then fix the broken characters. Then a guide line principle is introduced and according to it they can classify the characters into three groups that can help the next recognition step. Finally, the extraction of each character takes place. It starts by recognizing the top left character by using its feature of projection only. Then, the recognized character is removed from the image and the extraction restarts.

We have proposed a fuzzy logic scheme for this problem. There are two important reasons on the application of fuzzy techniques for character recognition. The reasons as stated in [146] are:

- Fuzzy techniques are powerful tools for knowledge representation and processing.
- Fuzzy techniques can manage the vagueness and ambiguity efficiently.

The vagueness in CAPTCHA is due to warping ambiguity, geometrical deformations and uncertain knowledge.

7.1.2. Yahoo Scheme

Our algorithm can be tested on any CAPTCHA's scheme that follows the same Yahoo's design characteristics. We will present the experimental results on a database generated according to the Yahoo characteristics [144] and also compare with other works using the same Yahoo scheme.

The new Yahoo scheme used for security purposes is based on the connected characters principles [144]. It consists of merging the characters horizontally after distorting them as shown in Figure 7.1. The obtained text image can still be easier for a human to guess but it will be harder for a computer attack.

In order to define the Yahoo scheme, we have used the study in [144]. The authors have collected from Yahoo's website 100 random samples. By analyzing these samples, they have reported the following characteristics:

- Six to eight characters are used in each sample.
- The background is white and the foreground is dark gray.
- Only 10 upper cases, 12 lower cases and 7 digits are used in the challenges. In order to guarantee the usability, the other 33 alphanumeric characters are not used. For example, it is not easy to identify I with 1 clearly for a human being.
- Warping is used for character distortion.
- Characters mostly connect with its neighbors.
- Scattered points are located around the challenge text.
- The whole text of some CAPTCHAs is cosine distorted. From left to right in horizontal direction, the previous character is upper than the next character in first quarter and last quarter parts. But in the middle part the previous character is lower than the next character. And other CAPTCHAs are flat.
- Characters are placed in accordance with their location at the guide lines.

(a) (b)

Fig.7.1. The current Yahoo CAPTCHA [144].

Starting from these characteristics, our algorithm is based on the following observations:

- Knowing the number of characters in advance (between six and eight) helps in the segmentation.
- The strong foreground and background contrast makes the detection of the characters contour possible, accurate and even repeatable for deformed characters.
- The linking phase in the edge detection eliminates easily the scattered points.
- Most of the characters connections correspond to ECs. Therefore, ECs can be used as keypoints for characters segmentation.

7.1.3. Fuzzy logic

In this section, we present first an introduction to fuzzy logic and then an overview on different fuzzy logic techniques applied in image processing.

7.1.3.1. Introduction: What is fuzzy logic?

Fuzzy Logic [145] is built around the concept of reasoning in degrees, rather than in boolean (yes/no) expressions like computers do. A fuzzy system is composed of input and output variables, called linguistic variables, defined in terms of fuzzy sets. Decision rules are specified by logically combining fuzzy sets. The combination of fuzzy sets defined for input and output variables, together with a set of fuzzy rules that relate one or more input fuzzy sets to an output fuzzy set, built a fuzzy system.

Different block diagrams have been introduced for a fuzzy logic system. However, the most commonly used [208] is composed from three main functions:

- Fuzzification: The input/output deterministic variables are transformed to linguistic variables defined as fuzzy sets. A membership function (Triangular, Trapezoidal or Gaussian) is selected to represent each fuzzy set.
- Fuzzy Laws: The fuzzy laws are applied on the input linguistic variables to calculate the output linguistic variable.
- Defuzzification: The output deterministic variable is estimated from the output linguistic variable using the inverse of the output membership function.

In image processing, fuzzy logic is widely used for clustering (known as fuzzy clustering detailed in section 7.1.3.2.1) or segmenting an image. It is also used in image feature detection (e.g. edge detection explained in section 7.1.3.2.2) and for shape/character recognition (detailed in sections 7.1.3.2.3 and 7.1.3.2.4).

7.1.3.2. Fuzzy logic overview

7.1.3.2.1. Fuzzy clustering

Fuzzy-C-Means (FCM) [155] is one of the most famous unsupervised fuzzy clustering techniques that are applied with success in image segmentation [156-166]. Although the original FCM algorithm yields good results for segmenting noise free images, it fails to segment images corrupted by noise or containing inaccuracy edges. This sensitivity is essentially due to the absence of utilization of the information on the spatial position of pixels to be classified.

The idea [179] is to partition a finite collection of n elements into a collection of C fuzzy clusters. FCM clustering [157] can be described as follows: Let X={x_1, x_2, ..., x_n} denoted a set of n objects to be partitioned into C clusters, where each x_j has d features. The FCM algorithm minimizes the objective function J defined as follows:

$$J = \sum_{i=1}^{C} \sum_{j=1}^{n} (u_{ij})^m D(x_j, v_i) \tag{7.1}$$

Where:

- v_i represents the ith cluster center.
- u_{ij} represents the membership degree of the jth object to the ith cluster.
- D represents a distance metric (e.g. Euclidean distance) that measures the similarity between an object and cluster center.
- $m \geq 1$ is the degree of fuzzification.

The membership degree u_{ij} of x_j to the ith cluster is determined by calculating the gradient of the objective function J with respect to u_{ij}. It is given by:

$$u_{ij} = (\sum_{k=1}^{C} (D(x_j, v_i)/D(x_k, v_i))^{\frac{1}{m-1}})^{-1} \tag{7.2}$$

The cluster centers v_i ($1 \leq i \leq C$) are determined by calculating the gradient of J with respect to v_i:

$$v_j = (\sum_{j=1}^{n} (u_{ij})^m x_j)/(\sum_{j=1}^{n} (u_{ij})^m) \tag{7.3}$$

The FCM algorithm can be summarized in the following steps:

- Step 1: Fix the cluster number. Initialize the centers by random points from data set.
- Step 2: Update the membership degrees by using Eq. (7.2).
- Step 3: Update centers using Eq. (7.3).
- Step 4: Repeat steps 2 and 3 until convergence.

The convergence of this algorithm will be reached when the change in membership values is less than the threshold.

Applied in image processing, the FCM's object x_j is an image pixel and the FCM clusters are the regions of similar contrast. For traditional FCM algorithm, the only used pixel's feature is the grey level feature which leads to bad classification in the presence of noise. Kalti et al [157] have tried to minimize this drawback by introducing additional pixel features like the spatial information of a pixel in its neighborhood.

7.1.3.2.2. Edge detection in digital images using fuzzy logic

Alshennawi et al [206] have proposed an edge detection technique in digital images using fuzzy logic. The major improvement is in the detection without determining a prior threshold value or need training algorithm. Patel and More [209] have suggested nearly the same edge detection method with the same enhancement but for Cellular Learning Automata (CLA). Next, we will detail the edge detection technique proposed by Alshennawi.

The input and output images of the suggested fuzzy system are both 8 bit quantized. Thus, their grey levels are between 0 and 255. The first step is the fuzzification where membership functions are selected for the input and output variables (pixel's grey level) shown in Figure 7.2. The membership functions are triangular and experimentally it is found that the best result to be achieved at the range black from zero to 80 gray values and white from 80 to 255. Thus, we have two fuzzy sets for the input variable: Black and White and three for the output variable: Black, Edge and White.

The second step, after fuzzification, is the introduction of fuzzy laws to find the fuzzy output. The proposed approach begins by segmenting the image into regions using 3x3 matrix. The condition of each pixel (Black, White or Edge) is depending on the weights of the eight neighbors that are degree of Black and degree of white. Figure 7.3 shows all the cases where the checked pixel is classified as an edge pixel: If the levels of the eight neighbors represented in one line are black and the remaining ones are white (Figure 7.3 (a)) then the checked pixel is an edge pixel. If the levels of four sequential pixels are black and those of the remaining four neighbors are white (Figure 7.3 (b)) then the checked pixel is an edge pixel.

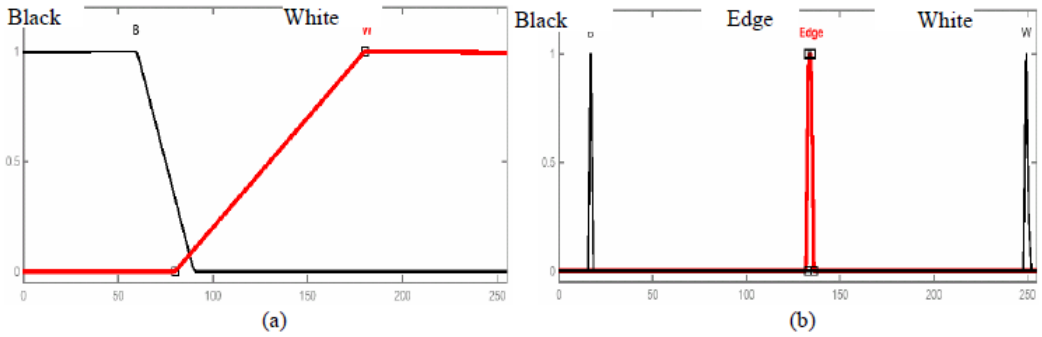


Fig.7.2. (a) Membership associated to the input variable. (b) Membership associated to the output variable [206].

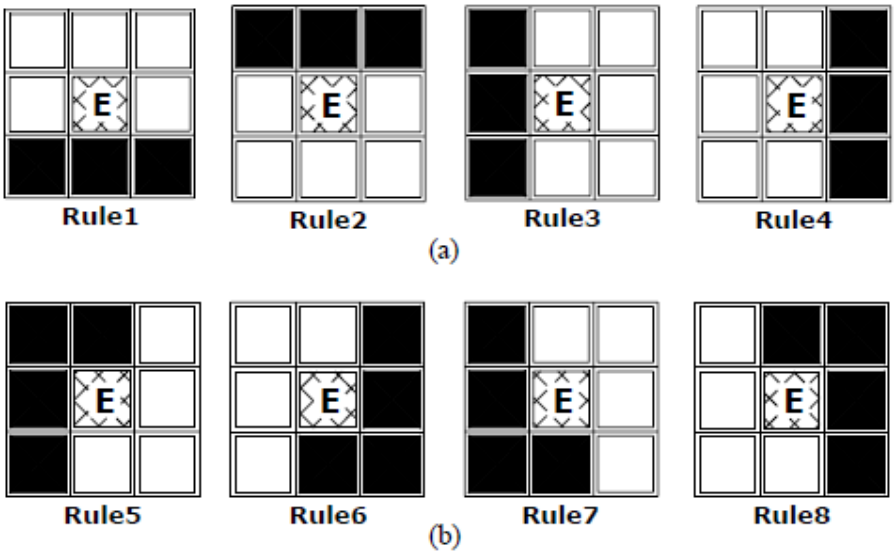


Fig.7.3. Fuzzy laws [206].

The third step is the defuzzification. The output image is a binary edge image where Black/White pixels are set to 0 grey level and Edge pixels are set to 255.

The major drawback of this method, as also claimed in the previous section 7.1.3.2.1, is in the only usage of the pixels grey levels without any usage of their spatial information. This fact makes the detection of edges fails in the presence of noise.

7.1.3.2.3. Recognizing hand-drawn geometric shapes using fuzzy logic

Fonseca et al [167] have proposed a fuzzy logic technique to recognize hand-drawn geometric shapes interactively. Their algorithm recognizes elementary geometric shapes: triangles, diamonds, rectangles, circles, ellipses, lines and arrows shown in Figure 7.4 (a) and five gestural commands: delete, undo, wavy-line, move

and copy shown in Figure 7.4 (b). Shapes are recognized independently of changes in rotation, size or number of individual strokes. Commands are shapes drawn using a single stroke.

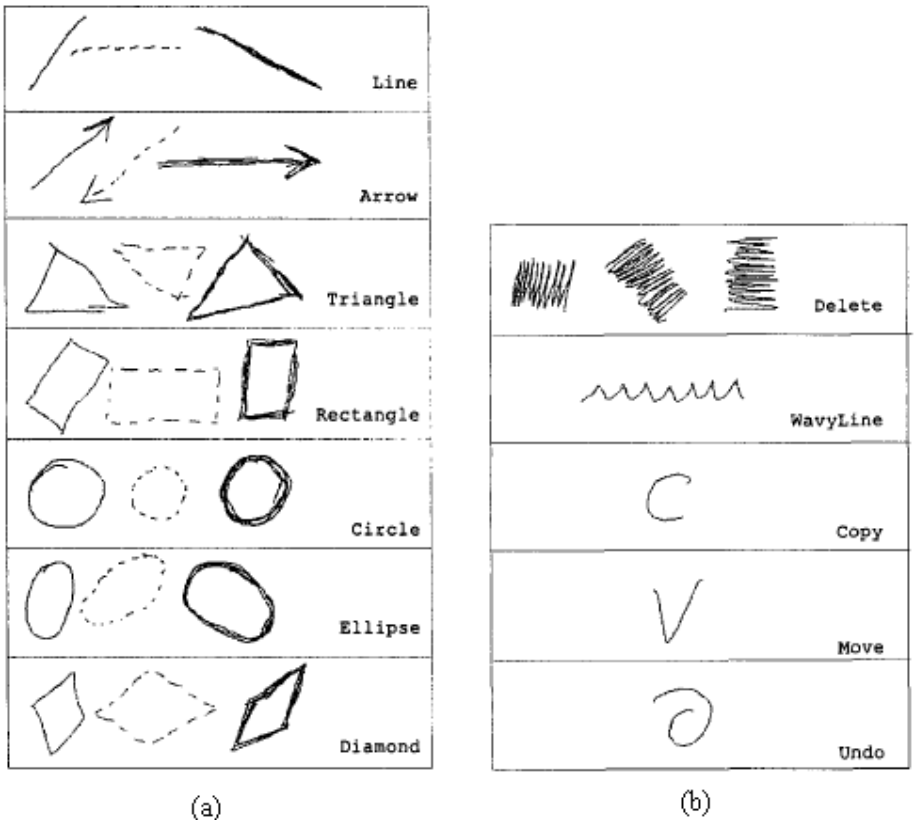


Fig.7.4. (a) Multi-stroke geometric shapes. (b) Uni-stroke shapes [167].

A training set is formed by drawing each shape thirty times: ten times using solid lines, ten times using dashed lines and ten times using bold lines. Next, the convex hull of these shapes is computed using Graham's algorithm [180]. Then using this convex hull three special polygons are computed and drawn. Using a simple three-point algorithm, the largest-area triangle that fits inside the convex hull is identified. The second polygon is the largest-area inscribed quadrilateral and the third is the smallest area enclosing rectangle (See Figure 7.5). Finally, the area and perimeter of each polygon are computed to estimate features and degrees of likelihood for each shape class.

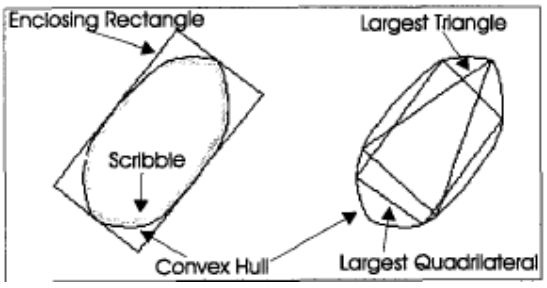


Fig.7.5. Polygons used to estimate features [167].

Each particular shape is characterized by a set containing one or more of these geometric features. For example, to distinguish Diamonds from other shapes the area of the largest triangle that fits inside the convex hull *(Alt)* is divided by the area of the largest quadrilateral (*Alq*). The obtained ratio has values between 0.5 and 0.6 for diamonds and bigger ones for other shapes as shown in Figure 7.6. Another example, in order to identify Rectangles two ratios are used. One measures the largest quadrilateral that fits inside the convex hull against the enclosing rectangle. For rectangular shapes the area of the convex hull will be very close to that of the enclosing rectangle (*Aer*) and this one will be very close to the largest quadrilateral. The *Ach/Aer* and *Alq/Aer* ratios will have values near unity for rectangles as shown in Figure 7.7.

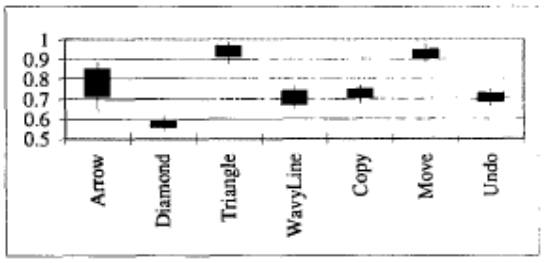


Fig.7.6. Percentiles for the ratio *Alt/Alq* [167].

The fuzzification starts by selecting a fuzzy set for each shape per geometric feature. For example, the fuzzy sets for the Diamond and Rectangles shapes versus the geometric feature *Alq/Aer* is shown in Figure 7.8. Each fuzzy set boundary is determined experimentally and can be concluded as well from the bottom row image in Figure 7.7. "Dom" in Figure 7.8 refers to degree of freedom.

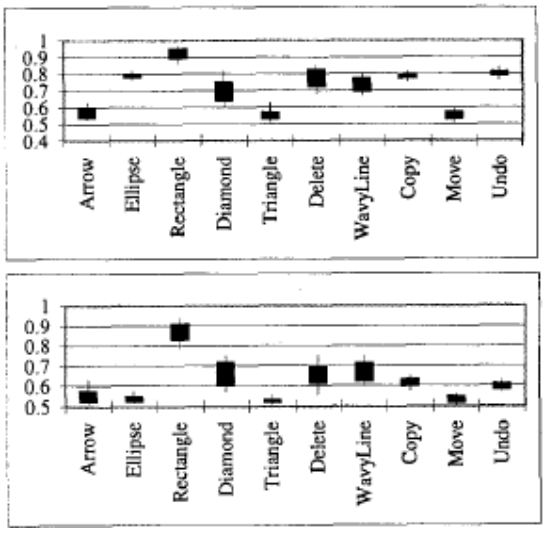


Fig.7.7. Percentiles for the ratios *Ach/Aer* and *Alq/Aer* [167].

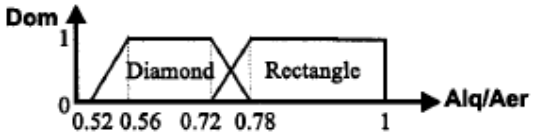


Fig.7.8. Fuzzy sets [167].

Next fuzzy rules are applied to take a decision and recognize the studied shape. The recognizer calculates the degree of membership for each shape class. This degree is the result of AND together degrees of membership for the relevant fuzzy sets. As an example, the fuzzy rule that recognizes a Diamond can be stated as follows:

If Alt/Alq is like Diamond AND

Alq/Ach is not like Ellipse AND

Alq/Aer is not like Rectangle

THEN

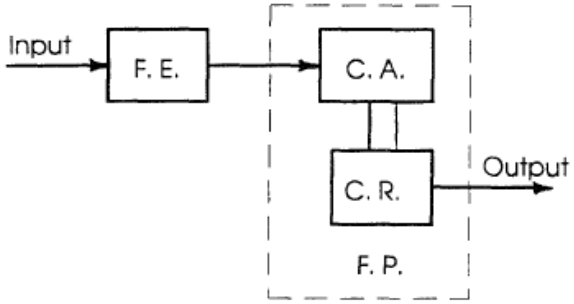
Shape is Diamond.

Where *Alt/Alq, Alq/Ach and Alq/Aer* are geometric features that characterize Diamond, Ellipse and Rectangle shapes respectively. "AND" denotes the conjunction of fuzzy predicates dom(f AND g) = min(dom(f),dom(g)) and "NOT" is defined by dom(NOT f) = 1 – dom(f).

This method is very important recognition method since it is based on geometric features of shapes. In addition, the fuzzification makes it very robust against natural random deformation introduced to each hand-drawn shape.

7.1.3.2.4. Recognizing handwritten characters using fuzzy logic

Fuzzy logic is also introduced for recognizing handwritten characters [168, 207]. The proposed system is shown in Figure 7.9.



F.E. - Feature Extractor
C.A. - Character Analyzer
C.R. - Character Recognition section
F.P. - Fuzzy Processor

Fig.7.9. Proposed character recognizer [168].

The input image is a handwritten character image from A to Z. It is partitioned into 4x4 matrix cells as shown in Figure 7.10.

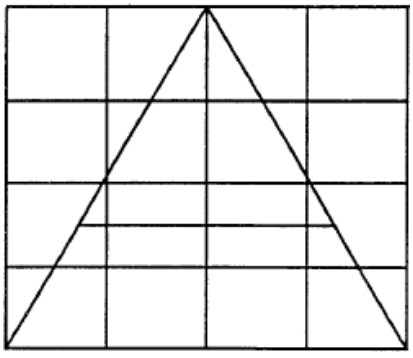


Fig.7.10. Character 'A' [168].

Each character cell is analyzed for characteristic features according to a specific pattern. Therefore, a pattern extractor is used for this purpose. Then, the Character Analyzer and Character Recognition System which constitute the fuzzy processor

consists of a number of IF ..... THEN .... ELSE rules (fuzzy rules). These rules may be applied to the output from the pattern extractor which will be in the form of a fuzzy matrix of all the cells constituting a particular character, each cell having the membership value, and the category name under which it has membership associated with them. The membership values may be classified into 'Low', 'Medium', and 'High' with membership values overlapping between them as shown in Figure 7.11. So, three fuzzy sets are introduced for this application.

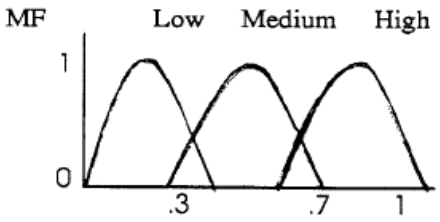


Fig.7.11. Membership functions for each pattern in each cell [168].

This method is a pattern recognition method that can lead to wrong recognition when the deformation introduced to the handwritten character is not small. In addition, it will not work in an attempt to recognize connected characters since some parts of a character can be merged with its successive character. In this case, a pattern recognizer would fail.

7.1.4. Our contribution

The proposed character recognition application is designed to attack and break CAPTCHA schemes to segment and recognize correctly the included characters. Most of existing methods have developed pattern recognition or machine learning algorithms to attack the CAPTCHA schemes but they haven't reported till now effective attacks on the new Yahoo CAPTHCA scheme based especially on the connected characters. In addition, they have tried to segment all the characters and after that try to recognize them.

Our proposed algorithm tries to solve the same problem of these studied ones but using new features called "Edge Corners (ECs)". These ECs are edge points corresponding to edge deviations that are repeatable over affine transformation. In addition, a given contour can be approximated using a polygon whose vertices are dominant ECs. These points are detected after the construction of the contours of an image using an edge detector. We have developed a simultaneous segmentation and

recognition technique per character similar to that suggested by Gao et.al [144] with a good success rate. The main difference is in the features used. They have developed pattern recognition scheme and we have used ECs to develop a fuzzy logic matching scheme.

In addition, we have developed a very efficient fuzzy logic algorithm for the character recognition based on the distribution of the ECs on the contour. The fuzzification idea comes from the introduced random deformation on the characters. It is shown to give good matching results even with this randomness. The overall suggested technique is able to solve many segmentation difficulties shown by other researchers. The idea proposed by Kalti [157] inspires us to develop spatial information rather than grey level information for robust matching of our image features (ECs). In addition, in our recognizer, we have followed nearly the same procedure of Fonseca [167] with also geometric features. However, our geometric features are very different since our application constraints are different.

We have published our work on ECs in [28, 198] and our work on character recognition using fuzzy logic in [200].

7.2. Edge corners "ECs" classification

In this application, edge corners "ECs" defined in chapter 4 are used. Each corner is characterized by:

- its angle which is the absolute difference of the two adjacent straight edges directions.
- the lengths of its two adjacent segments or straight edges.

Each EC is classified according to its two characteristics into "Strong" or "Week" corner (detailed in the next section). Therefore, we have made a small update to our EBCD to detect and classify the ECs according to these two types.

7.2.1. ECs Detection: Update to the EBCD

After detection of all edge pixels, the corner detection is initialized. At each edge pixel, three variables are used:

- *dirc*: the current direction of the edge at the current edge point.

- *dirp*: the previous direction of the edge at the previous edge point.

- *dirdiff*: their absolute difference given by,

$$dirdiff = |dirc - dirp| \tag{7.4}$$

$$if(dirdiff) > 4\ than\ dirdiff = 8 - dirdiff \tag{7.5}$$

The corner detection algorithm is shown in Figure 7.12.

- *Unmark all edge pixels.*
- *Iterate twice.*
  - *Foreach unmarked edge pixel.*
    - *Calculate dirdiff as shown in Eqs. (7.1) and (7.2).*
    - *If ((dirdiff>=2 and First Iteration)or(dirdiff==1 and Second Iteration))*
      - *Start the straight edge test along the direction dirc.*
      - *Start the straight edge test along the direction (dirp+4)%8.*
      - *If the two tests succeeded*
        - *Report the current pixel as a corner.*
        - *Mark the two straight edges.*

Fig.7.12. Corner detection algorithm.

Corners that are highly repeatable versus various deformations are those who have *dirdiff* above two. Therefore, the edge pixels whose *dirdiff* is greater or equal to two are tested first. In both directions of each of these pixels, we test the existence of two real straight edges *SE*1 and *SE*2 of length greater than three pixels for example. If the test succeeded, the current pixel is classified as a corner. All the pixels belonging to a detected straight edge are marked and will not be tested as corners. After testing the pixel with *dirdiff* larger than two, the edge pixels whose *dirdiff* is equal to one are tested next for a corner presence.

A corner is classified as strong corner if it has a *dirdiff* greater or equal to two and its two straight edges lengths are long enough (more than 3 percent of the total edge points on their contour). Otherwise, it is classified as a week corner. Figure 7.13 shows the edge image of the z character and its detected corners as solid points. The

total number of edge points is 330 pixels. From Table 7.1, points A, D, F and I are strong corners and have been tested first and reported as corners since their *dirdiff* is equal to two. As shown in Table1, each corner corresponds to two straight edges *SE*1 and *SE*2. Each of these SEs is characterized by its length: *Length* (number of pixels) and its direction: *dir* (mean of pixels directions).

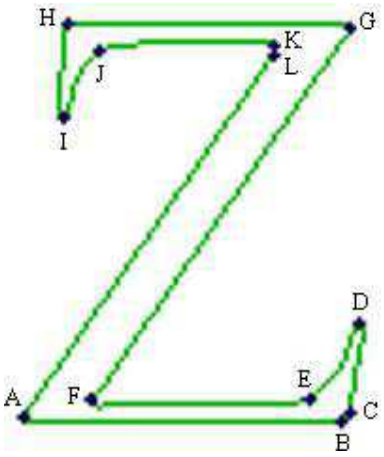


Fig.7.13. Edge image of the z character and its ECs.

Table 7.1. Corners Information of the z character.

Corners	***cdir,pdir,dirdiff***	**Straight Edges**			**Classification**
		SE1,SE2	Directions *dir1-dir2*	Lengths *Length1,length2*	
A	7,5,2	AB,AL	0-1.3	71,81	Strong
B	1,0,1	BC,BA	1.3-4	3,71	Week
C	2,1,1	CD,CB	1.9-4.6	20,3	Week
D	5,3,2	DE,DC	5.5-5.8	17,20	Strong
E	4,5,1	ED,EF	1.2-4	17,49	Week
F	1,3,2	FE,FG	0.1-1.3	49,83	Strong
G	3,2,1	GH,GF	4-5.3	63,83	Week
H	6,5,1	HG,HI	7.9-5.9	63,21	Week
I	1,7,2	IJ,IH	1.5-1.9	15,21	Strong
J	1,0,1	JK,JI	0-5.3	39,15	Week
K	6,7,1	KJ,KL	4.1-5.6	39,3	Week
L	5,6,1	LK,LA	2.3-5.3	3,81	Week

7.2.2. Why ECs?

We have focused in our research on detecting edge points that can have a strong immunity with respect to local and global warping introduced by Yahoo and other schemes. The Yahoo word image has a size of 290x80 pixels [150]. After applying

the corner detection on a set containing 100 random samples collected from Yahoo scheme, we have concluded with these observations:

***Observation* 1:** Some new week corners can appear (type I).

***Observation* 2:** Some week corners can disappear (type II).

***Observation* 3:** Rarely Strong corners disappear or new ones appear (type III).

***Observation* 4:** Above 95% of the connections between two characters contain at least one corner: Connection Corner (type IV).

***Observation* 5:** The border that splits two connect characters is not necessarily a straight line. It corresponds usually to a sequence of non collinear straight segments.

***Observation* 6:** When there is more than one Connection Corner (CC) on border between two characters, the maximal absolute difference between their abscissas is less than ten pixels.

The corners of types I and II are introduced due to the local warping that deforms the character and also due to the affine deformation applied to it. Type III corners can disappear only due to the merging of the two characters. It is shown in Figure 7.14 that the strong corners located to the left and to the right of the contour of the digit four are disappeared due to the connections to its neighboring characters.

Figure 7.14 (a) shows the edge image with the detected ECs, as solid points, of the Yahoo text shown in Figure 7.1 (b). Figure 7.14 (b) shows the edge images of the matched characters/digits. We can see also the corners of types I and II when we compare the corners on the contours of the deformed characters and the original characters shown in Figure 7.14 (a) and (b) respectively. In Figure 7.14 (b), the strong corners are shown on the characters/digits contours. On the other hand, Figure 7.14 (a) shows the connection corners (CCs) on the warped word image contour. Notice that from the first five circled CCs from the left located on the connection between the characters G and L of the deformed word in Figure 7.14 (a), we can draw the border that can split them correctly. It is clear that this border is formed by a set of consecutive straight segments each one of them is composed of two CCs.

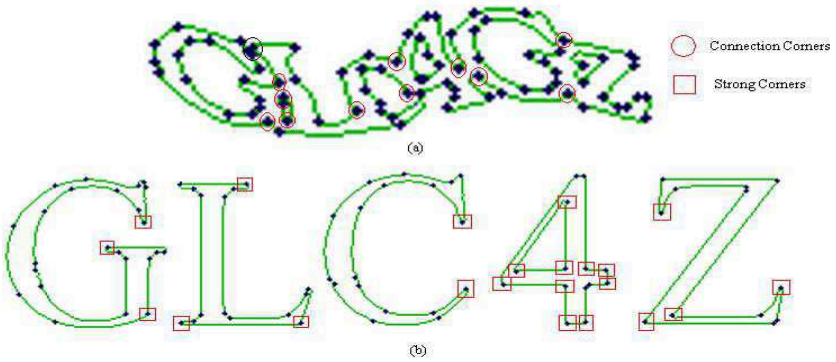


Fig.7.14. (a) Connection corners on the deformed characters contours. (b) Strong corners on the original characters contours.

Based on these observations, we have built an intelligent algorithm that can segment and recognize simultaneously each character of a Yahoo word image using its detected ECs.

7.3. Proposed scheme: Simultaneous Segmentation-Recognition

The proposed scheme consists of a simultaneous segmentation and recognition of the characters/digits in a deformed word image. Initially, a training set is formed. It contains the images of the characters/digits used by the Yahoo scheme to form the warped word image. The ECs of each training image are detected as shown for the characters G, L and C and for the digit 4 in Figure 7.14 (b). Then a test image, containing the deformed word image, is fed to the algorithm in an attempt to recognize correctly the characters/digits in it.

The segmentation consists of selecting a part from the input word image by setting a left and right border in an attempt to localize the target character. The recognition is used to match the segmented part with the characters/digits training set and outputs a matching percentage.

Figure 7.15 shows the proposed attack on a deformed word image composed of two characters: G and 3. The attack is characterized by five phases:

- Fix the left border (in red) and move the right border (in blue) across the ECs from left to right. These borders are straight lines passing through one EC. This is called "Segmentation trial".

- For each segmentation trial, the image part located between the left and right borders is matched to the training characters/digits. This operation outputs a matching percentage. Five segmentation trials are shown in Figure 7.15 (b).
- When all segmentation trials are considered or when the right border passes through the most right EC, we search for the trial having the highest matching percentage and select its right border for further testing to form the optimal multi-line (sequence of segments) right border. At any EC, the number of generated straight line right borders is much less than that of multi-line right borders. Therefore, to reduce the number of right borders to be tested, we consider first the straight line right borders formed at every EC in the image than we select only the one that achieve the highest matching percentage. It will be used as a foundation base to form the optimal multi line right border. In Figure 7.15 (b), the fourth trial is considered corresponding to highest matching percentage. In fact, this line is used to localize the region of connection between two consecutive characters.
- The EC of the selected straight line right border will localize all neighboring ECs with abscissas absolute difference less than or equal to ten pixels as revealed in Figure 7.16 (a) for all ECs located inside the black box. These located ECs will enter the test to form the optimal multi-line right border. This border is a sequence of segments passing by the ECs that are considered as CCs, according to Observation 5. Among all possible borders that pass through these ECs, the optimal one is the one that corresponds to the highest matching percentage. In this step, all training characters/digits are taken for matching. In Figure 7.16 (a), the optimal right border composed of four consecutive segments is shown and the recognition rate achieved is about 56% corresponding to the character "G".
- The optimal right border becomes the left border for the segmentation process of the next character/digit as shown in Figure 7.16 (b). The whole process will be repeated until each character in the word image will be segmented and recognized.

The idea of the proposed algorithm is shown in Figure 7.17 and it will be fully detailed throughout this chapter. The optimal segmentation is the one corresponding to the highest matching percentage obtained by the recognition stage. Initially, the

segmentation starts by taking the input deformed word image. Each segmented part, like for example the one located between borders LB-RB1 in Figure 7.18, enters the recognition phase to match the training characters. This phase will output a matching percentage. Then a new segmented part is generated, like the one located between borders LB-RB2 in Figure 7.18, and a new matching percentage is generated. The combined process segmentation/recognition will be repeated until the resulting segmented part becomes the whole input image. Then, the optimal multi line right border is formed to segment the optimal segmented part having the highest matching percentage. This part is then removed from the input word image and the whole segmentation-recognition process restarts again on the remaining image until all the characters/digits are recognized.

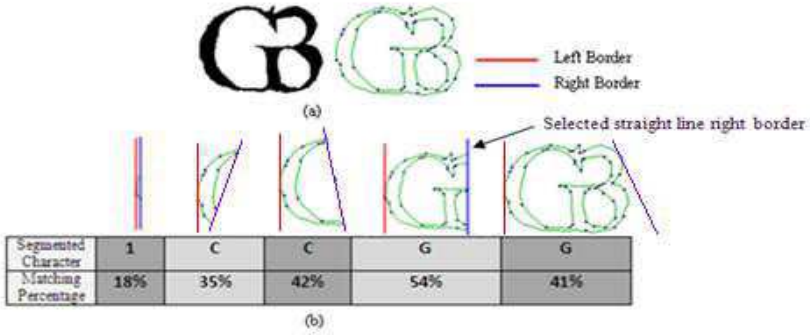


Fig.7.15. Search for the best straight line right border segmenting 2 deformed connected characters. (a) deformed image and its ECs. (b) various segmented parts and its corresponding recognized character with the matching percentage.

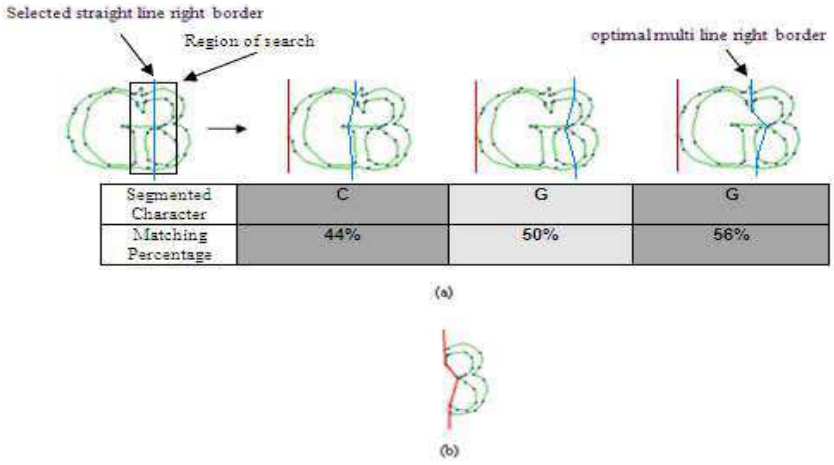


Fig.7.16. Optimal segmentation using multi line right borders. (a) Three different segmentation trials using three multi line right borders with the corresponding matching percentage. (b) The remaining part to recognize by the same procedure.

- *The LB and RB are vertical lines passing by the most left EC of the test image.*
- *Loop:*
- *While (RB does not pass by the most right EC)*
  - *Segment the part of the test image located between LB and RB.*
  - *Recognize it by matching with the training character images.*
  - *Output a matching percentage.*
  - *Form the straight lines RB at the next EC to the right.*
- *The straight line RB achieving the highest matching percentage enters into the search for the optimal multi line RB.*
- *If the optimal multi line RB does not pass by the most right EC.*
  - *LB is set as RB.*
  - *Goto Loop.*
- *Else*
  - *End of the algorithm: All characters are segmented and recognized.*

Fig.7.17. The overall algorithm.

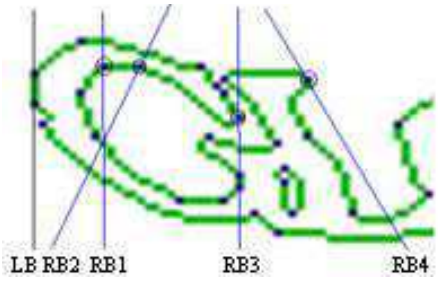


Fig.7.18. Segmenting and recognizing the first character of a part of the deformed GLC4GZ word image.

Figure 7.18 shows the segmentation and recognition process of the character G of the deformed word from Figure 7.1 (b). The left border *LB* is in black to the left. The other four borders are four straight line right borders *RBs* selected from hundreds of right borders. Each *LB-RB* couple corresponds to a segmentation trial and the segmented image's part located between them is matched to the characters/digits training set. *RB3* is the selected straight line right border that corresponds to the highest matching percentage among all straight line right borders as shown in Table 7.2. If the optimal right border is kept as a straight line passing by one EC as shown in Figure 7.19 (a), the segmented part will not correspond adequately to the G character.

Thus, the straight right border adjustment to a multi-line border is needed. It represents the fourth phase of the proposed attack. The goal is to increase the matching percentage, to give accurate segmentation of the desired character as it is shown in Figure 7.19 (b) and to draw the optimal left border of the next letter L. Table 7.2 shows the matching percentage and the matched training character/digit for each segmented part.

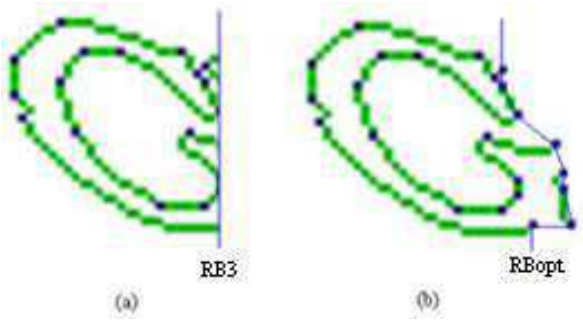


Fig.7.19. The segmentation of the "G" character of Figure 7.18. (a) selected straight line right border. (b) optimal multi-line right border.

Table 7.2. Matching percentage of various segmented image parts.

Segmented Part between	Matched Character	Matching Percentage
LB-RB1	D	22%
LB-RB2	C	26%
LB-RB3	C	33%
LB-RB4	G	23%
LB-RBopt	G	36%

Next we will explain each function of the algorithm alone and how they can corporate to attack the Yahoo scheme.

7.3.1. Segmentation

Based on Observation 4, above 95% of the connection edge points between two merged characters are corners called Connection Corners (CCs). Since the objective of any attacking scheme is to have a success rate above 0.01% [2, 3], we can speculate that the intermediate border, that splits correctly two connected characters, passes probably by a CC. In addition, using Observations 5 and 6, we can define a border as a segment or sequence of segments that passes through one or more CCs.

The segmentation's aim is to find the *LB* and *RB* that define adequately each character's borders in conjunction with the character recognition function. The first character segmentation starts by fixing the *LB* as a vertical straight line passing by the most left EC while moving *RB* through the ECs from the most left to the most right one. At every EC and for proper segmentation, the *RB* is set as *M* straight lines: One vertical line and *M-1* straight lines with slope deviations $-45<\Delta\theta<+45$. In Figure 7.20, the corresponding straight lines for the EC(32,35) are shown for $M=5$. This set is called "M Borders Set" (MBS). Then, the best straight line *RB* corresponding to the segmented part with the highest matching percentage is adjusted to a multi-line *RB* for more segmentation precision. For this purpose, a set called "Combination Borders Set" (CBS) is formed using only the EC of the best straight line *RB* and its neighboring ones. The optimal multi-line *RB* is adjusted to one of the CBS set. At this level, we should distinguish between:

- Definition1: M borders set (MBS) where each border is a straight line.
- Definition2: Combination borders set (CBS) where each border is a sequence of segments (multi-line).

7.3.1.1. Character's borders identification: M-Borders Set (MBS)

When starting the segmentation of the first character, it is intuitive to set the *LB* to be a vertical line passing through the first left corner. This is obvious since the ECs of the first character are all to the right of this *LB*. The construction of the *RB* is not that intuitive since it is located between two possibly connected characters where usually a border is a sequence of segments passing through one or more ECs. In addition, we don't know a priori which ECs are the CCs. Therefore, in order to set the adequate *RB*, the ECs set is ordered from left to right, then every EC is considered as a CC and the corresponding MBS is formed. The current EC used in this purpose is called *CCor*. For each *RB,* we try to recognize the character, using its ECs, located between *LB* and *RB*. The straight *RB*, from one of the MBSs, that corresponds to the best recognized character with the highest matching percentage is considered. Adjustments must be performed to obtain the optimal multi-line *RBopt*. Then for the recognition of the next character, we set the *LB* as the *RBopt* of the previously recognized character. This process will iterate until the *RBopt* passes through the last EC to the right of the image.

Due to deformations introduced to each character and especially rotation, some CCs are translated from their original positions. For example, the two most right strong corners of the G character shown in Figure 7.14 (b) have almost the same abscissa. However, their corresponding ones on the deformed word image of Figure 7.14 (a) have not. Therefore, we cannot set the RB only as a line passing by a CC. A good choice to overcome the variations in the ECs positions is to set RB also as an oblique straight line with a given slope. Some Yahoo CAPTCHAs are cosine distorted [18]. According to Observation 6, the maximal absolute difference between the abscissas of two CCs belonging to the same *RB* is less than ten pixels. Knowing that the height of the characters in a Yahoo scheme varies between 15 to 80 pixels [21], the maximum rotation introduced, corresponding to the ten pixels maximal difference between two CCs, is $\theta = tan^{-1}\left(\frac{10}{15}\right) = 33.7^{o}$. Therefore we take *M* lines with slope deviations $-45<\Delta\theta<+45$ as stated before. As a result and at every EC, a set of *M* straight line borders, called "MBS" (M Borders Set), with M different slopes (with slope deviation $\Delta\theta$) is formed. Figure 7.20 shows the MBS corresponding to the point (32, 35) set as *CCor* for M = 5 and $\Delta\theta = 22.5^{o}$.

However, the straight *RB* is not adequate because it passes only through one CC and does not represent a good choice of LB for the upcoming character. Therefore, a more accurate multiline RB segmentation procedure is used for which the CBS set is introduced.

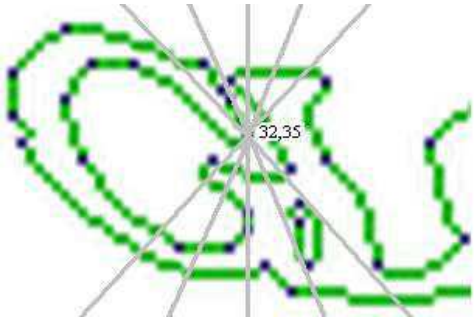


Fig. 7.20. The MBS (M=5) drawn in grey at EC (32,35).

7.3.1.2. RB adjustment: Combination Borders Set (CBS)

After detecting the best straight *RB* from the MBS set, we form a new set of borders passing through its corresponding current corner (*CCor)* and its neighboring ECs. We must define:

- Definition 3: Neighboring corner: an EC is classified as a neighboring corner to a CCor if it is located below it and the absolute difference between their abscissas does not exceed ten pixels according to Observation 6.

For the *CCor*, the borders formed by a combination of all its neighboring corners are called Combination Borders Set (CBS). Assume that n is the number of neighboring CCs, ordered by their ordinates, relative to a given *CCor*. All borders formed using *CCor* and its neighboring corners should have *CCor* as the first top corner. A border is composed of K corners ($1 \leq K \leq n+1$). The border passes through *CCor* and K-1 of its neighboring corners and it is composed of K+1 segments. The first segment is a vertical segment starting from top image border and passing through the first top *CCor*. The second segment connect *CCor* to its nearest neighboring corner. The remaining segments connect the successive neighboring corners from top to bottom. The last segment is a vertical segment passing through the last neighboring corner and ending on the bottom image border. Thus, the number of borders, composed of K corners, is a combination C_n^{K-1} since all the borders must pass through *CCor*.

The total number of possible borders nB, in this CBS set, formed by a *CCor* and its n neighboring corners is given by

$$nB = 1 + \sum_{i=1}^{n} C_n^i \tag{7.6}$$

The main idea behind introducing the MBS set first is to minimize the number of candidate borders to test. After detecting the best straight line *RB* from the MBS set, the multi-line RB_{opt} search starts by forming the CBS set corresponding only to the EC in the best vertical line *RB*. Then, the obtained *RB* is adjusted to obtain the RB_{opt} from the CBS corresponding to the highest Matching Percentage. This way, we can minimize the number of CBSs generated. The *RB* adjustment is also shown in Figure 7.22.

Figure 7.21 shows the corners coordinates located on the contour of the first two characters G-L from the Yahoo word image in Figure 7.1 (b).

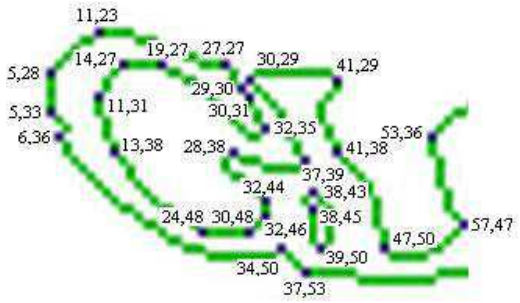


Fig. 7.21. Corner coordinates of G-L characters.

In Figure 7.22, we have drawn through the point CCor (29,30) three borders. The grey border is a straight border passing through only CCor while the red and blue are multi-lines borders passing through CCor and its neighboring corners. This CCor has n=13 neighboring corners. Therefore we obtain nB=8192 total number of possible borders. The drawn borders are only three possibilities of nB borders that are drawn starting by CCor. In addition, the matched character and the corresponding recognition percentage are also shown for each border.

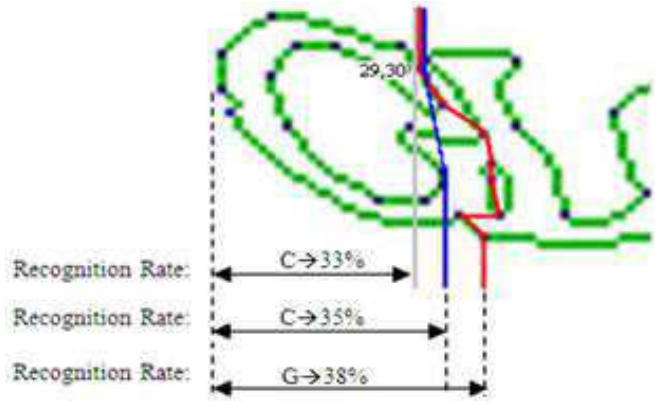


Fig. 7.22. Three borders, from the CBS, drawn to split the G-L characters.

7.3.1.3. *The algorithm: ECs based segmentation*

The segmentation algorithm is shown in Figure 7.23 using first the FBS. The variables used are defined as follows:

- *IsFirstCharRec*: Boolean variable used to check if the first character in the Yahoo word is recognized.

- *LB*: array of points (CCs) corresponding to the Left Border.

- *SetLB*: Boolean variable used to flag when *LB* should be set.

- *RB*: array of points (CCs) corresponding to the Right Border.

- *RBopt*: optimal *RB*.

- *CCor*: Current Corner.

- *SL*: Straight Line.

- *CShape*: array of ECs that form the Current Shape.

- *M*: input parameter equivalent to the number of straight lines per MBS.

- $\Delta\theta$: input parameter equivalent to the slope deviation.

- *b*: temporary border.

- *Mark all the ECs*
- *Loop:*
- *Set SetLB = true.*
- *For every CCor from the ECs set considered from left to right corner.*
  - *If(SetLB)// we want to set LB*
    - *If(!IsFirstCharRec)*
      - *Set LB as a vertical line passing through CCor. //CCor is the first left corner in this case.*
    - *Else*
      - *Set LB as optimal RB found: RBopt.*
    - *SetLB = false.*
  - *Else //we want to set RB*
    - *Form M straight lines passing through CCor and having different slopes with slope deviation $\Delta\theta$.*
    - *For every SL from the M straight lines set.*
      - *Set RB as SL.*
      - *Mark all ECs lying between LB and RB to form the ECs of CShape.*
      - *Call the function "Recognition":*
        - *Match the ECs of CShape with those of the characters-digits training shapes.*
        - *Report the matching percentage.*
- *Set RB and its CCor corresponding to the highest matching percentage.*

*//**** RB adjustment ****

- *Form the CBSs corresponding to CCor and its neighbors.*
- *For every border b from the CBSs.*
  - *Set RB as b.*
  - *Mark all ECs lying between LB and RB to form the ECs of CShape.*
  - *Call the function "Recognition":*
    - *Match the ECs of CShape with those of the characters/digits training shapes.*
    - *Report the matching percentage.*
- *Adjust RBopt to b corresponding to the highest matching percentage.*
- *IsFirstCharRec = true.*
- *If(RBopt does not contain the last EC to the right)*
  Goto Loop.

Fig.7.23. ECs based segmentation algorithm.

Our algorithm result on segmenting the characters in the word image in Figure 7.20 shows that the vertical border (border *RB3* in Figure 7.18) from the MBS formed at EC(32,35) is the best straight *RB* corresponding to highest recognition percentage. Then, the CBSs are formed corresponding to this EC. From these CBSs, the multi-line RB_{opt} will be selected to be the red border shown in Figure 7.22.

7.3.2. Recognition

After setting the *LB*, the search starts to select the multi-line *RBopt* that is always achieved by a recognition process. Therefore, the cooperation between segmentation and recognition is required and the optimal segmentation is achieved by the best recognition.

7.3.2.1. *EC properties used for recognition*

Three parameters are used for recognizing the ECs:

- The directions and the lengths of its two straight edges (SEs).
- The numbers of crossed edges from both sides of the EC's angle.
- The bin index or quadrant index.

7.3.2.1.1. Directions and lengths of the two SEs

Each EC is an intersection of two non collinear SEs as explained in the algorithm shown in Figure 7.12. The SE direction is the average of its edge pixels directions. In addition, the SE length is the number of these pixels. Therefore, an EC is characterized by the directions and lengths of the two adjacent SEs.

7.3.2.1.2. Number of crossed edges

Knowing the SEs directions at a given EC, we can calculate the EC's angle and form the bisector line. We can also select two additional lines passing through the EC and having a slope $\pm 45^{o}$ with respect to the bisector line. These line are labeled and drawn at EC(11,31) in Figure 7.24. The number of crossed edges by each line and from both sides of the angle is recorded. For each segmented part of a test image limited by a *LB* and *RB*, only the crossed edges located between the *LB* and the *RB* are taken into consideration as shown in Figure 7.24. Figure 7.24 shows the segmented part from the test image shown in Figure 7.21 where the *RB* is the red one. It is matched to the training image presented in Figure 7.25 (b). Line 2 is the angle bisector at EC(11,31). From the acute angle's facing side, the number of crossed edges

is equal to two for the three lines. From the opposite side, the number of crossed edges is equal to one. Note that the numbers of crossed edges from the two sides is not necessary equal at every EC. Using the number of crossed edges will eliminate a lot of false matches.

7.3.2.1.3. Bin index

In the segmented part image from the test image and the training image, two perpendicular lines, drawn in black in Figure 7.25, passing by the image center point are drawn. They partition each image into four bins or quadrants (Quad) indexed from one to four. Every EC records the index of the corresponding bin.

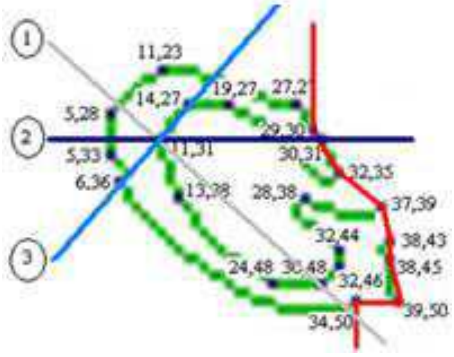


Fig. 7.24. Segmented part from the test image with the three lines drawn at EC(11,31).

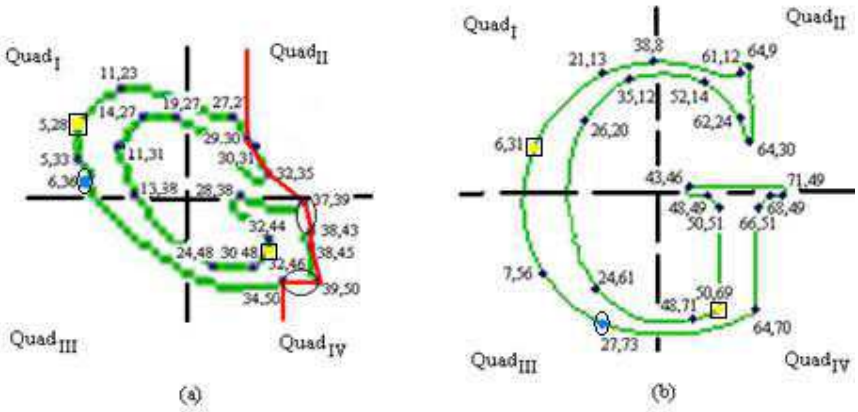


Fig.7.25. (a) Segmented part from the test image with quadrants (b) Training edge image relative to the G character with quadrants.

7.3.2.2. Most likely matched ECs "MLMC"

Two ECs, one from the training image and the other from the segmented part image, are said to be MLMC if:

- The two ECs and their two directly linked ones have the same straight edges directions.
- They have the same numbers of crossed edges.
- They belong to the same bin.

After segmenting an image part from a test image, it will be matched to all training images. The MLMCs are searched first. The SEs lengths are not considered due to the difference that can occur due to scaling as shown in Figure 7.25. Also, the appearance and disappearance of some ECs can modify the SE length of the directly linked EC. On the other hand, taking the direction average of pixels belonging to the same SE as its final direction is a good way to overcome the encountered noisy directions. For example in Figure 7.25 (b), the SE between the two ECs (26,20) and (24,61) has some noisy directions directly above EC(24,61). However, the final SE direction is around 1.9 which is the true direction starting form (24,61).

The ECs (5,28)-(6,31) and (32,46)-(50,69) marked in yellow in Figure 7.25 (a) and (b) are the MLMC that form the base for detecting other matched ECs. Note that the usage of the numbers of crossed edges has eliminated a lot of false matches. For example, EC(11,23) could be matched to EC(35,12) or EC(5,28) could be matched to EC(26,20) without taking into consideration the numbers of crossing edges.

7.3.2.3. Matched ECs

Due to the introduced deformations in each character's image, some ECs can have a variation in their bin index especially when they are located near the borders between two bins. To overcome these deformations, two MLMC matching conditions are followed but with some relaxations. These conditions are called relaxed matching conditions:

- They belong to the same bin with a certain deviation *r* from the corresponding bin is allowed. r represents a percentage of the bin width ($0<r<20\%$).
- They should have the same number of crossed edges except for the lines that cross the left or right borders.

Due to global warping, some ECs can transfer from a bin to another. Therefore, any bin of the segmented part image can be extended r% of its size in vertical or

horizontal direction. By this relaxation, matched ECs (6, 36) and (27, 73) marked in light blue in Figure 7.25 (a) and (c), respectively, belong to the same bin.

Due to the merging of successive characters/digits, some edge parts can disappear. In Figure 7.25 (b) for example, the circled edge parts between ECs (37,39) and (38,43) and between ECs (39,50) and (34,50) have been disappeared. This fact can lead to an incorrect number of crossed edges of some ECs. Therefore, we don't take into consideration the numbers of crossed edges for the lines that cross the left or right borders.

Every two ECs from both the training image and the segmented part image, that succeeded to pass the relaxed matching conditions, are subject for further testing. We have searched for parameters that can distinguish together an EC from another. Two parameters are introduced for every EC: the SEs directions and the position with respect to the MLMCs. Those two parameters are influenced by the image deformations which make the use of a sharp decision technique to match them is very difficult. Therefore, a fuzzy logic-based scheme is proposed for matching the ECs that have already pass the relaxed matching conditions.

7.3.2.4. Proposed Fuzzy system

Our proposed fuzzy system will give the matching percentage of the two studied ECs, one from the training image and the other from the segmented part image. Therefore, it is important to introduce some parameters that can be used in matching corresponding ECs and filter out false matches. These parameters cannot be fully invariant since the deformation made on the image's characters is random. However, the robustness of these parameters and the introduction of an efficient fuzzy system can compensate the randomness and make them powerful keys for correct matching.

By experiments, we have observed two ECs properties that are less influenced by the deformation. The first parameter is the angle difference between corresponding vectors in the training and segmented part image.

Definition 4: Corresponding Vectors: A vector formed by a training MLMC and a training EC and another vector formed by a test MLMC and a test EC are said to be corresponding vectors if their MLMCs and ECs are corresponding.

In Figure 7.26, three corresponding vectors out of many are shown: (5,28)-(11,23) corresponding to (6,31)-(21,13), (5,28)-(6,36) corresponding to (6,31)-(27,73) and (5,28)-(28,38) corresponding to (6,31)-(43,46). It can be seen clearly that these vectors have different magnitudes due to deformation and variation in scaling level but they have nearly the same angles. Therefore, the vector angle difference "*VecAngDiff*" between corresponding vectors can be selected as first parameter for matching ECs. This parameter reflects the amount of deviation of an EC from its original position due to the deformation introduced to the image.

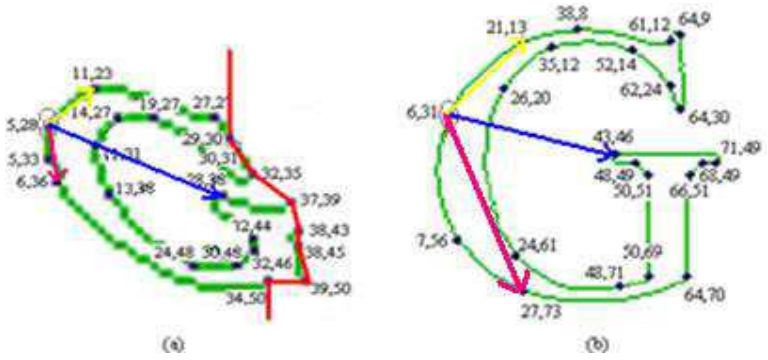


Fig.7.26. Vector angle difference of two matched ECs.

Using *VecAngDiff* parameter reduces the searching space for possible matches. However when used alone, we have observed that some false matches still appear. To further reduce these false matches, a second parameter is introduced. It is the difference between corresponding average adjacent SEs directions relative to an EC. It is called "*CorAngDiff*". For example in Figure 7.27, the corresponding adjacent SEs, from both corner side, of two matched ECs (11,23) and (21,13) are shown. It can be seen that the corresponding average SEs (ASE) directions are almost the same. Therefore, *"CorAngDiff*"is set as the average of the differences between corresponding ASEs. It reflects the resemblance between the directions of the edges adjacent to two matched ECs

These two parameters are used by the fuzzy system to output the matching percentage. The overall process involves two stages as shown in Figure 7.28. The first one is the preprocessing stage which output the two parameters, the vector angle difference *VecAngDiff* and the corner angle difference Cor*AngDiff* relative to the two ECs under consideration. The second stage represents the fuzzification process which

starts by setting membership functions for *VecAngDiff* and Cor*AngDiff* in order to obtain the matching percentage.

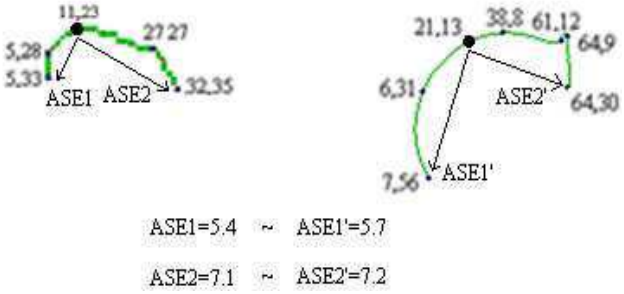


Fig. 7.27. Adjacent SEs directions of two matched ECs.

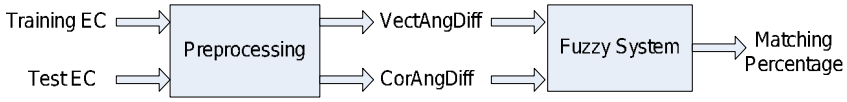


Fig. 7.28. Proposed fuzzy system.

7.3.2.4.1. Preprocessing: *VecAngDiff* and *CorAngDiff*

This section explains how to obtain the two parameters, *VecAngDiff* and *CorAngDiff*, which are used by the fuzzy system to estimate the matching percentage of the two ECs under investigation. These parameters can be classified as geometric features that characterize an EC. The geometric features are more robust to noise than grey level features (Kalti et al and Fonseca et al in section 7.3).

To illustrate the first parameter, consider two matched ECs one from the training image and the other from the segmented part of a test image. In each image, we form the corresponding vectors. For illustration, consider ECs (6,36) and (7,56) shown in Figure 7.26. Since we have two MLMCs, we obtain two couple of corresponding vectors: (5,28)-(6,36) and (6,31)-(7,56) which correspond to the first MLMC and (32,46)-(6,36) and (50,69)-(7,56) which correspond to the second MLMC. The angles difference in degree of every two corresponding vectors relative to the two matched ECs is taken. The average of these differences is set as *VecAngDiff*. This variable reflects the amount of shifting in position of an EC. However, alone it could not lead to the correct match especially when the MLMCs are also shifted due to the deformation. Therefore we have introduced the second parameter the *CorAngDiff*.

For the second parameter, some definitions must be known first:

Definition 5: Test Direction Average: From one side of a test EC, the Test Direction Average is the average direction of two successive SEs.

Definition 6: Training Direction Average: From one side of a training EC, the Training Direction Average is the average direction of two or more successive SEs leading to the smallest difference with respect to the corresponding Test Direction Average.

Based on the first three observations from Section 4.3, some new corners could appear and some corners could disappear. In addition, due to the stretching introduced to the training image, usually some corners disappear in the test image. Therefore, when calculating the "Training Direction Average", we take the average direction of an unspecified number of successive SEs that leads to the smallest difference with respect to the corresponding Test Direction Average.

At an EC of the segmented part image and from the two sides we calculate the Test Direction Averages. In addition, for the matched EC in the training image and also from the two sides we calculate the Training Direction Averages. Therefore, the *CorAngDiff* is the average of the differences between corresponding Training and Test Direction Averages. For example in Figure 7.26, the two ECs (11,23) and (21,13) are matched. We record for each side of the test EC (11,23) the test direction average of the two successive SEs. Thus, for the right side, we consider the SEs (11,23)-(27,27) and (27,27)-(32,35) and for the left side we consider the SEs (11,23)-(5,28) and (5,28)-(5,33). On the other hand for the training EC (21,13), to obtain the nearest directions to the test direction averages for EC (11,23), we should record:

- For the right side the training average direction of four successive SEs: (21,13)-(38,8), (38,8)-(61,12), (61,12)-(64,9) and (64,9)-(64,30)

- For the left side, the training average direction of two successive SEs (21,13)-(6,31), and (6,31)-(7,56).

The edge parts formed by the SEs to the left and to the right of EC (21,13) correspond actually to those formed by two SEs to the left and to the right of EC (11,23). Note that ECs (38,8) and (61,12) in the training image have disappeared in the segmented part image due to warping.

7.3.2.4.2. Fuzzification

The fuzzification process starts by setting membership functions for its two inputs and one output numerical variables (Alshennawi et al and Fonseca et al. in section 7.1.3). A membership function can have different shapes [196, 197]: triangular, trapezoidal, Gaussian, bell-shaped, etc. The triangular shape should be a start membership function to any fuzzy problem since it gives one of the best performances [196] and it can be upgraded to more advanced function when the results are not quite good. Therefore, three triangular membership functions, shown in Figure 7.29, are used for the numerical variables *VectAngDiff* (expressed in degree) and *CorAngDiff* (expressed in Freeman code [96]). The corresponding fuzzy sets are "H", "M" and "L" which stand for high, medium, and low, respectively. Moreover, we have used two Gaussian membership functions to represent the fuzzy sets "Like" and "Alike" introduced for the output variable *MatchingScore* as revealed in Figure 7.30 (c). The Gaussian function is characterized by its smoothness that is needed to output the matching score in our case. Note that many triangular membership functions for *VectAngDiff and CorAngDiff* have been tested corresponding to different fuzzy sets. However, the selected fuzzy sets and their member functions shown in Figure 7.30 have achieved one of the best matching performances. These particular fuzzy sets are selected since the experimental results based on them achieve high matching percentage.

7.3.2.4.3. Fuzzy Laws

After fuzzification, the input numerical variables *VectAngDiff* and *CorAngDiff* are transformed to linguistic variables. Fuzzy laws are logical operations that set the linguistic output variable *MatchingScore* according to the values of the input linguistic variables. *MatchingScore* has two memberships "Like" and "Alike". The fuzzy laws are as follows:

- if VectAngDiff is "L" and CorAngDiff is also "L" than MatchingScore will be "Like".
- if VectAngDiff is "M" and CorAngDiff is "L" than MatchingScore will be "Like".
- Otherwise MatchingScore will be "Alike".

Table 7.3 summarizes the fuzzy laws.

Table 7.3. Fuzzy laws.

		VectAngDiff		
		L	M	H
CorAngDiff	L	Like	Like	Alike
	M	Alike	Alike	Alike
	H	Alike	Alike	Alike

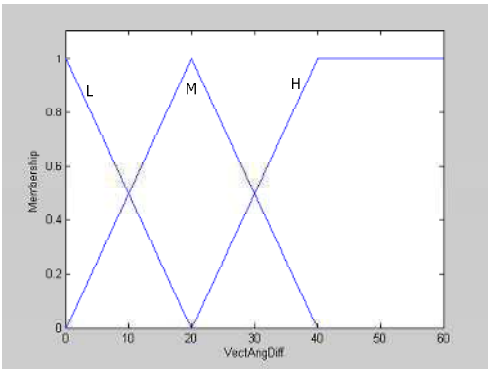
7.3.2.4.4. Defuzzification

Defuzzification is the process that evaluates numerically the linguistic output variable (Alshennawi et al. in section 7.1.3). After applying the fuzzy laws, the linguistic output variable *MatchingScore* has two partial memberships between zero and one to the sets "Like" and "Alike". For exam ple, a *MatchingScore* could have 0.65 for "Like" and 0.4 for "Alike" as shown in Figure 7.30. Then, the inverse of the two Gaussian member functions for "Like" and "Alike" respectively, are used to get the Like percentage "*LikePerc*" and the Alike percentage "*AlikePerc*" as numerical values.

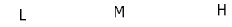
Therefore, the system's numerical output, called *MatchingPercentage*, can be derived as expressed in Eq (7.7),

$$MatchingPercentage = \frac{Like * LikePerc + ALike * ALikePerc}{Like + Alike} \qquad (7.7)$$

After Defuzzification, each EC from the segmented part image will have matching percentages with their matched ECs from the training image. The matched training EC that corresponds to the highest matching percentage is selected. Thus, couples of matched ECs are formed. Table 7.4 presents the matched couples and their matching percentage for the images shown in Figure 7.25. Note that the MLMC are given a 100% matching percentage.



(a)



(b)

(c)

Fig.7.29. The three membership functions: (a) for *VectAngDiff*, (b) for *CorAngDiff* and (c) for *MatchingScore*.

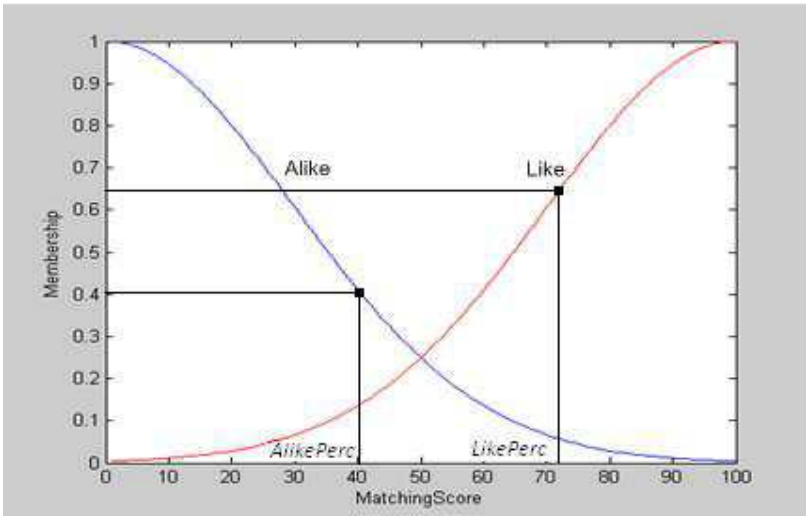


Fig. 7.30. Calculating the LikePerc and AlikePerc by inverse member functions.

7.3.2.5. Verification

Verification step is very important for a correct recognition of a segmented part image. After constructing the couples of matched ECs including the MLMC, the idea is to check the correctness of their edge linking order taking into consideration the probability of appearance or disappearance of some corners.

Starting by any couple from the MLMC set in each image, the two ECs from any matched couple should have the same linking order taking into consideration the order of matched ECs only. For example for Figure 7.25, one MLMC couple is (5,28)-(6,31). In Figure 7.25 (a), the EC (5,33) is the first linked EC with respect to EC(5,28) in the direction 6. Also in Figure 7.26 (b), the EC (7,56) is the first linked EC with respect to EC(6,31) in the direction 6. Therefore, the matched couple (5,33)-(7,56) is verified. One can easily verify all the remaining matched couples presented in Table 7.4. Note that the ECs (38,8) and (61,12) and others don't enter in the order calculation since these ECs have no match. These ECs are considered as Type I or Type II that are introduced due to the image deformation.

7.3.2.6. Overall matching percentage

The fuzzy system outputs the matching percentage of every matched ECs couple from the training and segmented part images. The issue is to calculate the overall matching percentage between the two images. Usually while recognizing a training character/digit image, the recognition of an EC having large SEs lengths gives more percentage than the recognition of an EC having small SEs lengths. Therefore in the training image, each matched EC is weighted by the sum of the lengths of its two adjacent SEs. The unmatched ECs have a zero matching percentage. Thus, the training character/digit contour can be seen as a set of weighted ECs where the sum of their SEs length corresponds to its total number of edge points. Thus, the overall matching percentage is the weighted average of the ECs matching percentages. This measure reflects the recognition percentage between a segmented part image and one of the training images. Finally, the training character image that corresponds to the highest matching percentage is the recognized character/digit for the segmented part from the test image.

Table 7.4. Matched couples for the images shown in Figure 7.25.

Test ECs	Training ECs	Matching Percentage	Matching Type
(5,28)	(6,31)	100%	MLMC
(5,33)	(7,56)	83%	Matched ECs
(6,36)	(27,73)	24%	Matched ECs
(11,23)	(21,13)	48%	Matched ECs
(13,38)	(24,61)	59%	Matched ECs
(14,27)	(35,12)	26%	Matched ECs
(19,27)	(62,24)	55%	Matched ECs
(27,27)	(64,9)	24%	Matched ECs
(28,38)	(43,46)	87%	Matched ECs
(30,48)	(48,71)	71%	Matched ECs
(32,46)	(50,69)	100%	MLMC
(32,44)	(50,51)	65%	Matched ECs

Figure 7.31 shows the recognition algorithm to generate the overall matching percentage. The variables used in this algorithm are:

- *nbptsM*: number of edge points matched in the training image, initially cleared.

- *nbptsT*: total number of edge points in the training image.

- *For every EC from the training character image.*
  - *For every EC from the segmented part from test image.*
    - *If (the difference between the two corresponding straight edges directions is below 0.5) and (they belong to the same bin)and(they have the same numbers of crossed edges from both sides of the angle)*
      - *The couple of ECs is a MLMC.*
- *For every EC from the training character image.*
  - *For every EC from the segmented part from test image.*
    - *If (they belong to the same bin with a certain deviation r from the corresponding bin (0<r<20%)) and (they have the same number of crossed edges except for the lines that cross the left or right borders).*
      - *For every couple from MLMC set.*
        - *Form the training vector composed by the current training EC and the training EC from the current MLMC.*
        - *Form the test vector composed by the current test EC and the test EC from the current MLMC.*
        - *Calculate the angle difference VectAngDiff between the two vectors*
        - *For the current test EC, record from each side the direction average, called "test direction average", of the two successive SEs.*
        - *For the current training EC, record from each side the direction average, called "training direction average", of the two or more successive SEs leading to the smallest difference with respect to the corresponding test direction average.*
        - *Calculate the differences between the training and test direction averages taken from the two corner sides.*
        - *Calculate the average CorAngDiff of the two differences.*
        - *Do the fuzzification of the two parameters VectAngDiff and CorAngDiff.*
        - *Calculate the corresponding MatchingScore.*
        - *Do the Defuzzification to obtain the MatchingPercentage using Eq. (7.4).*
  - *Set the couple training/test ECs having the maximal MatchingPercentage as matched ECs.*
  - *nbptsM+=(maximal MatchingPercentage)*(adjacent SEs length of the matched Training EC),*
- *Verify the edge linking of the matched couples.*
- *If(Verified)*
  - *Calculate the total number of edge points nbptsT in the training image,*
  - *Calculate the overall matching percentage = nbptsM*100/nbptsT.*
- *Else*
  - *Overall matching percentage = 0.*

Fig.7.31. Recognition algorithm.

7.4. Experimental results

7.4.1. First experiment: Recognition under warping

In this experiment, we have used one deformed character. The studied parameter is the level of deformation which is the level of warping. To generate a huge database, we have formed our own program that generates images of one deformed character. The warping, as mentioned in [49,169], is composed from three main functions. The first one is the generation of a random displacement at every image's pixel. The

second one is convolving the random displacement field with a Gaussian filter (low pass filter) with standard deviation *Sigma*. The third one is the generation of the deformed image by interpolation.

The warping level is controlled by two parameters: the standard deviation *Sigma* of the Gaussian filter and the magnitude *Alpha* of the random displacement field. Figures 7.32 and 7.33 show the effect of these two parameters on the G character image.

Fig.7.32. (a) Original image, (b) deformed image: Sigma=3 and Alpha=1, (c) deformed image: Sigma=3 and Alpha=20.

Fig.7.33. (a) Original image, (b) deformed image: Sigma=5 and Alpha=35, (c) deformed image: Sigma=7 and Alpha=75.

For low values of *Sigma* as in Figure 7.32, the warping is called local warping that produces small ripples, waves and elastic deformations along the pixels of the character [49]. For high values of *Sigma* as in Figure 7.33, the warping is global which produces elastic deformations at the character level. In addition, the deformation level is proportional to the parameter *Alpha* since it is the magnitude of the random displacement field generated at every pixel.

Our original database is composed of 19 characters. We have followed the observation in [144] that shows that Yahoo scheme uses 10 upper cases, and 12 lower cases and 7 digits with probability of appearance in 1000 random samples.

The recognition percentage shown in Figure 7.34, is obtained using our proposed fuzzy system based recognition algorithm on a database of 1000 random deformed characters generated for every pair of *Sigma* and *Alpha*.

When *Sigma* increases, the recognition percentage increases for the same values of *Alpha*. The reason for that is when *Sigma* increases the deformation transfers from local to global which smoothes the deformation on the characters contours. For example in Figure 7.35 (b) and (c), two deformed images of character '2' corresponding to two different values of *Sigma* are shown. It is clear that for *Sigma*=3 (local deformation) in Figure 7.35 (b), the contour is highly deformed and a lot of noisy ECs appear. Whereas for *Sigma*=7 (global deformation) shown in Figure 7.35 (c), the deformation is relatively small.

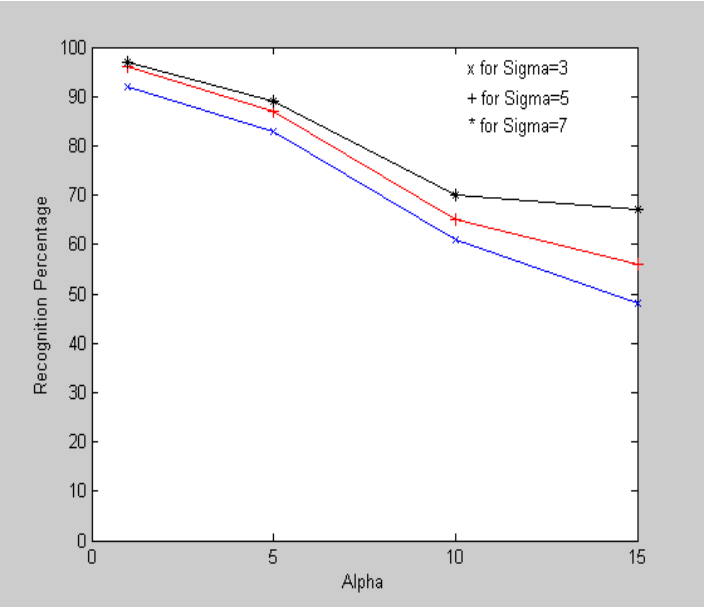


Fig.7.34. Recognition percentage versus Alpha and for different values of *Sigma*.

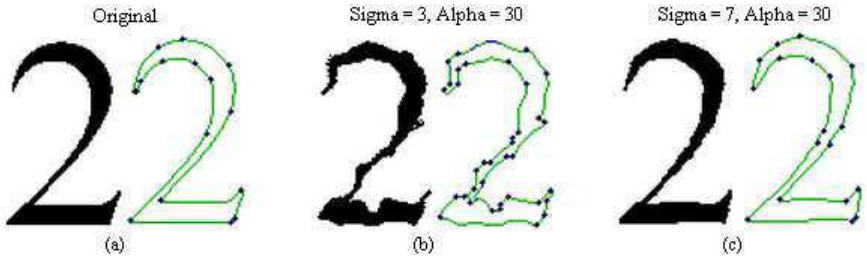


Fig.7.35. Deformation levels at different values of *Sigma*.

7.4.2. Second experiment: Segmentation/Recognition of connected characters

In this experiment, we try to show the efficiency of the suggested technique to segment and recognize two deformed and connected characters. Therefore, we have formed a database which is composed of 1000 samples, each of them contains two connected characters. For all the images in this database, we have fixed the two parameters *Sigma* and *Alpha*. On the other hand, we have introduced a new parameter *dist* that reflects the distance between the two connected characters. For negative values of *dist*, the two characters are connected. Thus, in this experiment, we have fixed the warping level for all images and made the connection level as the only variable. Figure 7.36 shows the recognition percentage of the algorithm at different connection levels for different values of *dist*. It is clear that as the characters become more connected, thus higher negative values of *dist*, the ability to correctly recognize the characters will be smaller as expected.

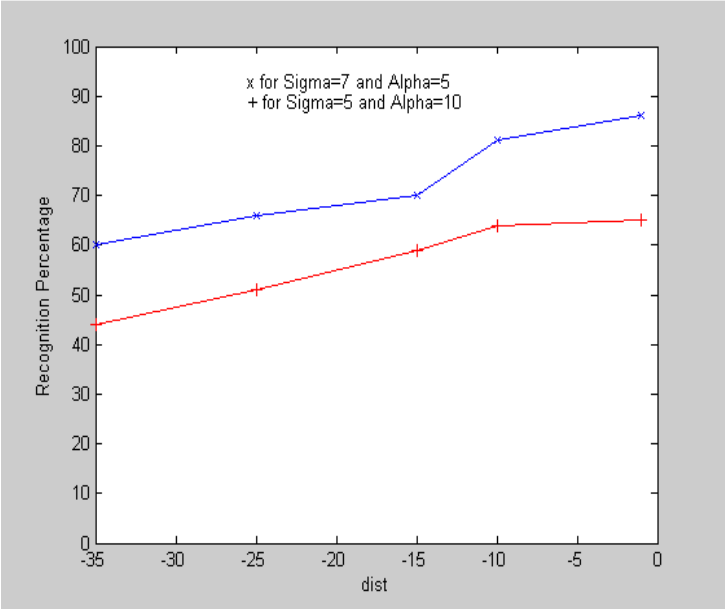


Fig.7.36. Recognition percentage at different values of *dist*.

In addition, Figure 7.37 shows the segmentation of the characters in three different sample images taken from the generated database. The border in blue between the characters passes at least by one connection corner.

Fig.7.37. (a) Two connected characters. (b) Segmentation using connection corners (CCs).

7.4.3. Third experiment: Breaking Yahoo CAPTCHA scheme

In this experiment, a database of 1000 samples is created where each sample is a composed of six deformed connected characters generated following the Yahoo scheme explained previously. The obtained recognition percentage is 57.3% which is higher than 54.7 % recognition percentage obtained in [144] using a Yahoo database of 1000 samples.

In addition, a step by step Segmentation/Recognition of a Yahoo CAPTCHA sample image is provided in Figure 7.38.

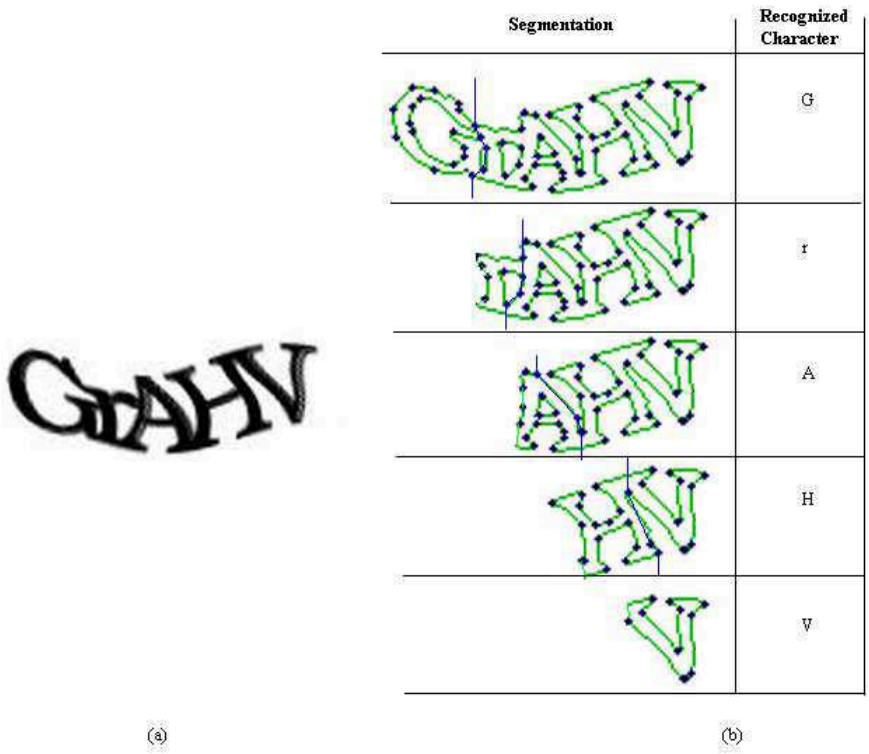


Fig.7.38. (a) A CAPTCHA sample word image. (b) Segmentation/Recognition steps.

8

Conclusion and Future Works

In this thesis, a novel corner detector is introduced based on image edges. It can outperform existing interest point detectors especially in repeatability. The edge detection constructs the starting phase. Then a suggested straight edge detector examines all the formed contours in the input image and detects the edge parts that can be considered as straight edges. These straight edges are straight lines that exist frequently at the borders of various objects in real scenes especially human made environments like buildings, cars, doors… The importance of detecting these straight edges remains in their role of detecting edge corner points. The intersection of two non collinear straight edges of appropriate length (greater than a threshold) is reported as a corner. The newly proposed detector has good results against some image transformation especially affine transformation. It detects almost scale invariant corners without using scale space representation. Thus, selecting only corners located at edges with high gradient norm and whose adjacent segments are long enough gives its more immunity against affine transformation especially scale variation. Its novelty is due to its simplicity using only image edges.

Experimental results show that our proposed EBCD is a very interesting corner detector compared to other existing algorithms. We have proposed a 2D object recognition application using these corners and the recognition was fluent even in the presence of different shapes. On the other hand, several applications can be built using this detector and the corner points. 3D reconstruction or 3D modeling of polyhedral objects can be based on exploring its corner points with its geometric distribution within the object. Localization of an autonomous robot can be specified by using 3D corners as reference points in a stereovision system and that can lead to a SLAM application. Polygonal approximation of an object's contour can be addressed using a number of these corners as vertices of the approximating polygon.

Based on the edge corners, a new polygonal approximation technique is proposed. By fixing an entry parameter as a stopping criterion, Compression Ratio *CR* or weighted sum error *WE*, the algorithm starts per contour by removing iteratively the corners that introduce the minimal possible ISEV to the global ISE measure. At the end, the remained corners, called DCs, form the vertices of the polygon that can best approximate the current contour. The experimental results have shown good results in comparison with other existing methods. In our opinion, this is due to the efficient straight edge detector that explores all the contour corners efficiently and then to the

iterative polygonal approximation algorithm that removes, at each iteration, the corner corresponding to the smallest LISEV and at the same time updates and reexamines the LISEV of already removed corners. By this way, we can ensure that the remained corners form the polygon that best fit their contour.

As a first application on DCs, a novel technique for image registration using dominant corners located on image edge is presented. First, it was shown experimentally that these DCs have very good repeatability versus affine transformation. Therefore, the suggested image registration is applicable to images where the introduced deformation can be modeled by an affine transformation. Then primitives are formed by grouping every four DCs: consecutive and nearest. The ratio of the areas of two triangles formed in every primitive construct the first invariant measure used to match a couple of primitives in a source and target images. In addition, the angle directions difference of the two adjacent straight edges relative to a DC form the second invariant measure of this DC. Therefore, two primitives, one from the source image and the other from the sensed image, are matched using the three tests:

- matching of the primitive area ratio R.
- matching the four DCs angle directions difference.
- matching of the affine models formed by the four triplets in the first primitive and the corresponding four triplets in the second primitive.

This scheme eliminates a lot of false matching and makes the difference high between the number of votes for the correct model and other false ones. The suggested algorithm can be used in image registration where the time interval between sequences of images is relatively small.

Finally, a simultaneous segmentation-recognition algorithm for an efficient attack on a Yahoo CAPTCHA is proposed. The goal from breaking these CAPTCHAs is not to be hackers but to explore the weaknesses in their design. The suggested algorithm is based on new proposed kinds of edge corners: week corners, strong corners and connection corners. For segmenting each character in the image, left and right borders are introduced. These borders pass necessary by at least one connection corner and the image part in between is the segmented image part. This part then enters into

recognition. If it is recognized as a character/digit from the training database, the overall algorithm restarts and tries to segment and recognize the following characters/digits in the image. To come up with the random deformation introduced to each character in the image, we have introduced a fuzzy system for recognizing the characters using their edge corners. The experimental results show the flexibility of this system to recognize connected and deformed characters at different levels. The obtained success rate on the generated Yahoo database is very good and exceeds the required 0.01% to classify a successful attack. In addition, the introduced algorithm can be very useful to do some improvement on schemes using the connection characters principle like MSN or Google CAPTCHAs in order to be more robust against attackers.

For future works, the first attempt is to study the repeatability of the detected DCs by the edge detector (EBCD) versus projective transformation. If the obtained repeatability is acceptable then we can generalize our image registration application to more general cases. As an example, our algorithm can be applicable in the case where the time interval between the two studied images is not small. In this case, the projective transformation is the best transformation that can model the real deformation that relates the two images.

More than one robotic application should be developed next:

The first one is the shape recognition application based on the edge corners that developed for a more complicated application like object recognition where more than one contour could exist. In this application, matching individual corners is not enough since it will generate a lot of false matches. Therefore, we will introduce a new matching strategy based on a group matching per contour. Each contour in a training image is matched to a contour in a test image using the distribution of the matched corners taking especially into consideration the possibility of appearance or disappearance of some week edge corners.

The second one is the road surveillance by a drone's camera to remove the global motion of the camera and estimate the local moving targets motion. This can be done by testing and analyzing the suggested image registration application on acquired images sequence.

The third one is the communication in a human-robot interaction using images containing commands or info. For example, a robot can help in fetching some items identified by a key number or a key word or maybe a key character. Thus, a person presents to the robot the handwritten key acquired as an image by the robot that in his turn should recognize the written characters and fetch the corresponding item.

Bibliography

[1] L.G. Shapiro and G.C. Stockman. Computer Vision. *Prentice Hall*, 2001.

[2] T. Morris. Computer Vision and Image Processing. *Palgrave Macmillan*. 2004.

[3] T. Lindeberg. Scale-space. *In: Encyclopedia of Computer Science and Engineering (Benjamin Wah, ed), John Wiley and Sons*, Volume IV, pages 2495-2504, 2008.

[4] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision,* 30(2): 77-116, 1998.

[5] T. Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *J. of Applied Statistics*, 21(2): 224-270, 1994.

[6] J. Koenderink and A. van Doorn. Local features of smooth shapes: Ridges and courses. *In SPIE Proc. Geometric Methods in Computer Vision II*, volume 2031, pages 2-13, 1993.

[7] J. Koenderink and A. van Doorn. Two-plus-one-dimensional differential geometry. *Pattern Recognition Letters*, 15(5): 439-444, 1994.

[8] L. Griffin, A. Colchester, and G. Robinson. Scale and segmentation of images using maximum gradient paths. *Image and Vision Computing*, 10(6):389-402, 1992.

[9] W. Bijsterveld, and J.A. Bolaño. Detection and Classification of Road Signs for Automatic Inventory Systems using Computer Vision. *Integrated Computer-Aided Engineering*, 19(3): 285-298, 2012.

[10] D.T. Lin and D.C. Pan. Integrating a Mixed-Feature Model and Multiclass Support Vector Machine for Facial Expression Recognition. *Integrated Computer-Aided Engineering*, 16(1): 61-74, 2009.

[11] A. De Cabrol, P. Bonnin, V. Hugel, K. Boucheffra and P. Blazevic. Temporally Optimized Edge Segmentation for Mobile Robotics Application. *SPIE Optics Photonics, Application of Digital Image Processing XXVIII*, San Diego California, USA, August 2005.

[12] J. Hou, Z. Chen, X. Qin, and D. Zhang. Automatic image search based on improved feature descriptors and decision tree. *Integrated Computer-Aided Engineering*, 18(2): 167-180, 2011.

[13] D. Lowe. Local Feature View Clustering for 3D Object Recognition. *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, 2001.

[14] F. Rothganger, S. Lazebnik, C. Schmid and J. Ponce. 3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. http://www.cvr.ai.uiuc.edu/ponce_grp/publication/paper/ijcv04d.pdf.

[15] S. Bouchafa and B. Zavidovique. Efficient cumulative matching for image registration. *Image and Vision computing*, volume 24, 2006.

[16] S. Se, D.Lowe and J. Little. Vision-based Mobile Robot Localization and Mapping using Scale-Invariant Features. http://www.cs.ubc.ca/~lowe/papers/icra01.pdf.

[17] P. Bonnin, A. de Cabrol. Projet RNTL Cléopatre COST Vision Robotique. 2006.

[18] R. Kirsch. Computer determination of the constituent structure of biological images. *Computers and Biomedical Research* 4, 1971.

[19] J.M.S Prewitt. Object enhacement and extraction. *Picture processing &psychopictorics, BS Lipkin&A.Rosenfelded, Academic Press*, New York USA, 1979.
[20] I. Sobel. Neighborhood coding of binary images for fast contour following and general binary array processing. *Computer Graphics & Image Processing* USA, volume 8, pages 127-135, 1978.
[21] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *Int. J. Computer Vision*, volume 1, pages 167–187, April 1987.
[22] J. Shen and S. Castan. An Optimal Linear Operator for Step Edge Detection. *CVGIP*, volume 54, pages 112–133, 1992.
[23] M.A. Fischler, R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM*, volume 24, pages 381-395, 1981.
[24] A. Kolesnikov. Efficient Algorithms for Vectorization and Polygonal approximation. PhD Thesis, 2003
[25] J. Canny. A computational approach to edge detection. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, volume PAMI-8, pages 679–698, 1986.
[26] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, volume 31, 1998.
[27] M. Herman, T. Kanade and S. Kuroe, Incremental Acquisition of a Three-Dimensional Scene Model from Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume Pami-6, No. 3, 1984.
[28] R.Nachar, E.Inaty, P.Bonnin and Y.alayli. "A robust edge based corner detector", *17th IEEE Mediterranean Electrotechnical Conference*, Beirut, Lebanon, pp: 242-246, April 13-16, 2014.
[29] N. Ayache and O. Faugeras. A new approach for the recognition and positioning of two dimensional objects. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, volume 8, 1986.
[30] M. Ali and D. Clausi. Automatic registration of SAR and visible band remote sensing images. I*n Proc. IEEE Int. Geosci. Remote Sens. Symp*, volume 3, pages 1331–1333, 2002.
[31] G. Wolberg and S. Zokai. Robust image registration using log-polar transform. *Proceedings of IEEE International Conference on Image Processing*, volume 1, 2000.
[32] B. Reddy and B. Chatterji. An FFT-based technique for translation, rotation and scale invariant image registration. *IEEE Trans. Image Process*, 5(8): 1266–1271, August 1996.
[33] Y. Almehio and S. Bouchafa. Matching image using invariant level-line primitives under projective transformation. *In Canadian Conference on Computer and Robot Vision (CRV)*, 2010.
[34] N. Gouiffes, N. Lertchuwongsa and B. Zavidovique. Mixed color/level lines and their stereo-matching with a modified hausdorff distance. *Integrated Computer-Aided Engieering*, 2011.
[35] T. Wong, P. Kovesi and A. Datta. Projective Transformations for Image Transition Animations. *14th International Conference on Image Analysis and Processing (ICIAP)*, 2007.
[36] W. Zhi-guo, W. Ming-Jia and W. Yu-qing. Image Mosaic Technique Based on the Information of Edge. *3rd International Conference on Digital Manufacturing and Automation (ICDMA)*, 2012.

[37] C.R. Maurer and J.M. Fitzpatrick. A review of medical image registration. *In Interactive Image-Guided Neurosurgery (R. J. Maciunas, ed.)*, Park Ridge, IL: American Association of Neurological Surgeons, pages 17–44, 1993.
[38] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and Vision Computing* 21, 2003.
[39] S. Wenchang, S. Jianshe, G. Xiaofei and Z. Lin. An improved InSAR image registration algorithm. *2nd International Conference on Industrial Mechatronics and Automation (ICIMA)*, volume 2, 2010.
[40] H. Li, S. Manjunath and S.K. Mitra. A Contour-Based Approach to Multisensor Image Registration. *IEEE Transactions on Image Processing*, volume 4, 1995.
[41] S. Kumar, K.V. Arya, V. Rishiwal and P.N. Joglekar. Robust Image Registration Technique for SAR Images. *First International Conference on Industrial and Information Systems*, 2006.
[42] O. Chum and J. Matas. Geometric Hashing with Local Affine Frames. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 2006.
[43] H. Lin, P. Du, W. Zhao, L. Zhang and H. Sun. Image registration based on corner detection and affine transformation. *3rd International Congress on Image and Signal Processing*, volume 5, 2010.
[44] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. *Cambridge university press, 2nd Edition*, 2003.
[45] Y. Almehio. A Cumulative Framework for Image Registration using level-line Primitives. PhD thesis, Universite Paris Sud XI, 2012.
[46] L. von Ahn, M. Blum and J. Langford. Telling Humans and Computer Apart Automatically, *Comm. Of the ACM*, 46, Aug. 2003.
[47] J. Yan and A. S. El Ahmad. A Low-cost Attack on a Microsoft CAPTCHA. *15th ACM Conference on Computer and Communications Security (CCS'08)*, Virginia, USA, Oct 27-31, 2008.
[48] K. Chellapilla, K. Larson, P. Simard and M. Czerwinski. Designing human friendly human interaction proofs, *ACM CHI'05*, 2005.
[49] K. Chellapilla, K. Larson, P. Simard and M. Czerwinski. Building Segmentation Based Human-friendly Human Interaction Proofs. *2nd Int'l Workshop on Human Interaction Proofs, Springer-Verlag*, LNCS 3517, 2005.
[50] K. Chellapilla and P. Simard. Using Machine Learning to Break Visual Human Interaction Proofs. *Neural Information Processing Systems (NIPS), MIT Press*, 2004.
[51] K. Chellapilla, K. Larson, P. Simard, M. Czerwinski. Computers beat humans at single character recognition in reading-based Human Interaction Proofs. *In 2nd Conference on Email and Anti-Spam (CEAS'05)*, 2005.
[52] R.Nachar, E.Inaty, P.Bonnin and Y.alayli. "Image registration based on edge dominant corners", 9th International Conference on Computer Vision, Theory and Applications, Lisbon, Portugal, pp: 433-440, January 5-8, 2014.
[53] R.Nachar, E.Inaty, P.Bonnin and Y.alayli. "Polygonal Approximation of an Object Contour by Detecting Edge Dominant Corners Using Iterative Corner Suppression", 9th International Conference on Computer Vision, Theory and Applications, Lisbon, Portugal, pp: 247-256, January 5-8, 2014.
[54] J.J. Koenderink and A.J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics* 55, pages 367-375, 1987.
[55] J.J. Koenderink and A.J. van Doorn. Generic neighborhood operators. *IEEE Trans. Pattern Anal. Machine Intell.* 14(6): 597-605, 1992.

[56] L.P. Kuptsov. Gradient. *In Hazewinkel, Michiel, Encyclopedia of Mathematics, Springer*, ISBN 978-1-55608-010-4, 2001.
[57] http://en.wikipedia.org/wiki/Laplace_operator.
[58] http://opencv.itseez.com/doc/tutorials/imgproc/imgtrans/laplace_operator/laplace_operator.html.
[59] http://en.wikipedia.org/wiki/Hessian_matrix.
[60] D. Gilbarg and N.S. Trudinger. Elliptic Partial Differential Equations of Second Order. *Springer-Verlag*, ISBN 3-540-41160-7 ISBN 978-3540411604, Berlin, 1983.
[61] L. Kitchen and A. Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters* 1(2): 95–102, 1982.
[62] J.J. Koenderink and W. Richards. Two-dimensional curvature operators. *Journal of the Optical Society of America*: Series A 5 (7): 1136–1141, 1988.
[63] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2): 91-110, 2004.
[64] G. Borgefors. Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(6), Nov. 1988.
[65] http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
[66] http://www.acemedia.org/aceMedia/files/document/wp7/2006/eccv06-inria.pdf
[67] F. Suard, A. Rakotomamonjy, A. Bensrhair, A. Broggi. Pedestrian Detection using Infrared images and Histograms of Oriented Gradients. *Intelligent Vehicles Symposium*, Tokyo, Japan, June 13-15, 2006.
[68] A.E. Johnson and M. Hebert. Recognizing Objects by Matching Oriented Points. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 1997.
[69] http://www.cv.tu-berlin.de/fileadmin/fg140/Head_Pose_Estimation.pdf.
[70] http://www.cs.cmu.edu/~rahuls/pub/cvpr2004-keypoint-rahuls.pdf.
[71] http://robotics.caltech.edu/readinggroup/vision/mikolajcICCV2001.pdf.
[72] http://www.vision.ee.ethz.ch/~surf/eccv06.pdf.
[73] R. Zabih and J. Wood_ll. Non-parametric Local Transforms for Computing Visual Correspondence. *In Proceedings of European Conference on Computer Vision*, pages 151-158, Stockholm, Sweden, May 1994.
[74] W.T. Freeman and E.H. Adelson. The Design and Use of Steerable Filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9), Sep. 1991.
[75] http://www.fit.vutbr.cz/research/view_pub.php?file=%2Fpub%2F9598%2Fprispevek.pdf&id=9598.
[76] http://disp.ee.ntu.edu.tw/~pujols/Gabor%20wavelet%20transform%20and%20its%20application.pdf.
[77] J. Lim, Y. Kim and J. Paik. Comparative Analysis of Wavelet-Based Scale-Invariant Feature Extraction Using Different Wavelet Bases. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 2(4): 29-38, December 2009.
[78] G. Chow and X. Li. Towards a system for automatic facial feature detection. *In Pattern recognition*, 26(12): 1739-1755, 1993.
[79] P. Seitz and M. Bichsel. The digital doorkeeper: automatic face recognition with the computer. *In IEEE Inter. Carnahan Conf. on security technology*, 1991.
[80] G. Yang and T.S. Huang. Human face detection in a complex background. *In Pattern recognition*, 27(1): 53-63, 1994.

http://www.alab.t.utokyo.ac.jp/~shino/scanned_paper/02/02/Pattern%20RecognitionVol.27NO.1.pdf.
[81] M. Yang and N. Ahuja. Detecting Human Faces In Color Images. *Proc. IEEE Int'l Conf. Image Processing*, pages 127-139, Oct 1998.
[82] D. Lowe. Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.
[83] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 13(5): 441-450, may 1991.
[84] Y. Ke, and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. *Computer Vision and Pattern Recognition*, 2004.
[85] T. Joliffe. Principal Component Analysis. *Springer-Verlag*, 1986.
[86] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(10), Oct. 2005.
[87] H. Bay, A. Ess, T. Tuytelaars and L.V. Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110(3): 346--359, 2008.
[88] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *In: CVPR* (1), pages 511–518, 2001
[89] http://ece631web.groups.et.byu.net/References/Invariant%20features%20from%20interest%20point%20groups.pdf.
[90] https://www.uni-hohenheim.de/~gzim/Publications/haar.pdf.
[91] M. F. Costabile, C. Guerra, and G.G. Pieroni. Matching shapes: a case study in time varying images. *Computer Vision, Graphics and Image Processing*, volume 29, pages 296–310, 1985.
[92] H. C. Liu and M. D. Srinath. Partial classification using contour matching in distance transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 12 pages 1072–1079, 1990.
[93] M. H. Han and D. Jang. The use of maximum curvature points for the recognition of partially occluded objects. *Pattern Recognition*, volume 23, pages 21–33, 1990.
[94] L. Dreschler and H. Nigel. Volumetric model and 3D trajectory of a moving car derived from monocular TV-frame sequence of a street scene. *Proceedings of IJCAI*, pages 692–697, 1992.
[95] H. L. Beus and S. S. H. Tiu. An improved corner detection algorithm based on chain-code plane-curves. *Pattern Recognition*, 20(20):291–296, 1987.
[96] H. Freeman and L. S. Davis. A corner finding algorithm for chain code curves. *IEEE Tran. Compt.*, volume 26, pages 297–303, 1977.
[97] K. Rangarajan, M. Shah, and D. V. Brackle. Optimal corner detection. *Computer Vision, Graphics and Image Processing*, volume 48, pages 230–245, 1989.
[98] A. Singh. Gray level corner detection - A generalization and a robust real-time implementation. *Computer Vision, Graphics and Image Processing*, 51(1): 54–69, 1990.
[99] H. Moravec. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, *Tech Report CMU-RI-TR-3, Carnegie-Mellon University, Robotics Institute*, September 1980.
[100] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.
[101] J. Shi and C. Tomasi. Good Features to Track. *9th IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[102] C. Tomasi and T. Kanade. Detection and Tracking of Point Features. *Pattern Recognition*, volume 37, pages 165–168, 2004.
[103] S. M. Smith and J. M. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision* 23(1): 45–78, May 1997.
[104] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision* 60(1): 63–86, 2004.
[105] K. Mikolajczyk and C. Schmid. An Affine Invariant Interest Point Detector. *In: ECCV*, pages 128–142, 2002.
[106] Mahalanobis, P. Chandra. On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India* 2(1): 49–55. 1936 Retrieved 2012.
[107] N. Nain, V. Laxmi, B. Bhadviya and N. Chand Singh. Corner Detection using Difference Chain Code as Curvature. *Proc. of the International MultiConference of Engineers and Computer Scientists (IMECS)*, volume 1, Hong Kong 2008.
[108] S. Pei and C. Lin. The Detection of Dominant Points on Digital Curves By Scale Space Filtering. *Pattern Recognition Society*, 25(11), 1992.
[109] X.C. He and N. H. C. Yung. In Curvature Scale Space Corner Detector with Adaptive Threshold and Dynamic Region of Support. *International Conference in Pattern Recognition, IEEE Computer Society*, volume 2, pages 791–794. 2004.
[110] L. G. Roberts. Machine perception of 3-D solids. *Optical and Electro Optical Information Processing*. MIT Press, 1965.
[111] V. Torre and T. A. Poggio. On edge detection. *IEEE Trans. Pattern Anal. Machine Intell.*, 8(2): 163-187, Mar 1986.
[112] Prewitt. Object enhancement and extraction. *Picture processing & psychopictorics, BS Lipkin & A.Rosenfeld ed*, Academic Press, New York USA, 1979.
[113] J.F. Canny. Finding Edges and Lines in Images. Technical report "*Artificial Intelligence Laboratory" of Massachussetts, Institute of Technology,* number 720, Juin 1983.
[114] P. Bonnin. Chaînes Algorithmiques Simples de Détection de Contours et de Régions Polycopié de Cours.
[115] http://www.cse.unr.edu/~bebis/CS791E/Notes/EdgeDetection.pdf.
[116] J. Dunham. Optimum Uniform Piecewise Linear Approximation of Planar Curves. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 8(1): 67-75, 1986.
[117] T. Pavlidis. Algorithms for Graphics and Image Processing. *Springer Verlag*, 1982.
[118] K. Wall and P. Danielsson. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision Graphics and Image Processing*, volume 28, pages 220-227, 1984.
[119] A. Gupta, S. Chaudhury and G. Parthasarathy. A Hough Transform Based Approach to Polyline Approximation of Object Boundaries. *Proceedings. 11th IAPR International Conference on Pattern Recognition*, volume.III. Conference C: Image, Speech and Signal Analysis, 1992.
[120] A. Mikheev, L. Vincent and V. Faber. High-Quality Polygonal Contour Approximation Based on Relaxation. *Proceedings Sixth International Conference on Document Analysis and Recognition*, 2001.
[121] A. Kolesnikov. Fast Algorithm for ISE-Bounded Polygonal Approximation. *15th IEEE International Conference on Image Processing*, 2008.

[122] A. Kolesnikov. Minimum Description Length Approximation of Digital Curves. *16th IEEE International Conference on Image Processing*, 2009.
[123] A. Kolesnikov. Nonparametric Polygonal and Multimodel Approximation of Digital Curves with Rate-Distortion Curve Modeling. *18th IEEE International Conference on Image Processing*, 2011.
[124] A. Pinheiro. Piecewise Approximation of Contours Through Scale-Space Selection of Dominant Points. *IEEE Transaction on Image Processing*, 19(6), 2010.
[125] M.T. Parvez and S.A. Mahmoud. Polygonal Approximation Of Planar Curves Using Triangular Suppression. *10th International Conference on Information Science, Signal Processing and their Applications*, 2010.
[126] A. Carmona-Poyato, F.I. Madrid-Cuevas, R. MedinaCarnicer and R. Munoz-Salinas. Polygonal approximation of digital planar curves through break point suppression. *Pattern Recognition*, 43(1): 14-25, 2010.
[127] A. Masood. Optimized polygonal approximation by dominant point deletion. *Pattern Recognition* 4(1): 227- 239, 2008.
[128] M. Marji and P. Siy. Polygonal representation of digital planar curves through dominant point detection – a nonparametric algorithm. *Pattern Recognition* 37, pages 2113- 2130, 2004.
[129] M. Sester, H. Hild and D. Fritsch. Definition of ground control features for image registration using GIS data. *Proceedings of the Symposium on Object Recognition and Scene Classification from Multispectral and Multisensor Pixels*, CD-ROM, Columbus, Ohio, 1998.
[130] M. Holm. Towards automatic rectification of satellite images using feature based matching. *Proceedings of the International Geoscience and Remote Sensing Symposium IGARSS'91*, Espoo, Finland, 1991.
[131] Y.C. Hsieh, D.M. McKeown, F.P. Perlant. Performance evaluation of scene registration and stereo matching for cartographic feature extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 1992.
[132] T. Suk and J. Flusser. The projective invariants for polygons. *Computer Analysis of Images and Patterns Lecture Notes in Computer Science*, volume 970, 1995.
[133] T. Tuytelaars and L.Van Gool. Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision* 59(1), 2004.
[134] Z. Yang and F. Cohen. Image registration and object recognition using affine invariants and convex hulls. *IEEE Transactions on Image Processing*, 8(7), July 1999.
[135] Y. Almehio, S. Bouchafa and B. Zavidovique. Level line primitives for image registration with figures of merit. *Integrated Computer-Aided Engineering*, 21(2): 101-118, 2014.
http://hal.archives-ouvertes.fr/hal-00841278.
[136] Y. Bentoutou, N. Taleb, K. Kpalma, and J. Ronsin. An automatic image registration for applications in remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, 43(9): 2127-2137, 2005.
[137] X. Lou, W. Huang, B. Fu, and J. Teng. A feature-based approach for automatic registration of noaa avhrr images. *In IEEE International Conference on Geoscience and Remote Sensing Symposium*, 2006.
[138] L. Ahn, M. Blum, N.J. Hopper and J. Langford. CAPTCHA: using hard AI problems for security, *22nd international conference on Theory and applications of cryptographic techniques*, Springer-Verlag, Berlin, 2003

[139] C. Pope and K Kaur. Is It Human or Computer? Defending E-Commerce with CAPTCHA. *IEEE IT Professional*, March 2005.
[140] J. Yan and A.S. El Ahmad. Breaking Visual CAPTCHAs with Naïve Pattern Recognition Algorithms. *In Proc. of the 23rd Annual Computer Security Applications Conference (ACSAC'07)*, IEEE computer society, USA, Dec 2007.
[141] G. Mori and J. Malik. Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, June 2003.
[142] G. Moy, N. Jones, C. Harkless and R. Potter. Distortion estimation techniques in solving visual CAPTCHAs. *IEEE CVPR*, 2004.
[143] Casey. Using AI to beat CAPTCHA and post comment spam, 2005. http://www.mperfect.net/aiCaptcha/
[144] H. Gao, W. Wang, Y. Fan. Divide and Conquer: An Efficient Attack on Yahoo! CAPTCHA. *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2012.
[145] H.Zjmmermann. Fuzzy set theorv and its applications. *Allied Publishers Ltd and Kluwer Academic Publishers*, 2nd Ed. New Delhi, 1996.
[146] http://be.et.uni-magdeburp.de/-hamidl.
[147] C. Schmid, R. Mohr, and C. Bauckhage, Evaluation of interest point detectors, *Int. J. Comput. Vis.*, 37(2): 151–172, Jun 2000.
[148] J. Denton. Two Dimensional Projective Point Matching. July 2002.
[149] R.Duda and P.Hart, Use of the Hough Transformation to Detect Lines and Curves in Pictures, Comm. ACM, Vol. 15, pp. 11–15, 1972.
[150] https://edit.yahoo.com/registration?.intl=us&new=1&.done=http%3A//mail.yahoo.com&.src=ym&.v=0&.u=ak37rod3tebb2&partner=&.partner=&pkg=&stepid=&.p=&promo=&.last=#
[151] http://users.fmrib.ox.ac.uk/~steve/susan/
[152] http://www.mathworks.com/matlabcentral/fileexchange/17894-keypoint-extraction
[153] http://www.ee.surrey.ac.uk/
[154] C.H. The and R.T. Chin. On the detection of dominant points on digital curves, *IEEE Transactions of Pattern Analysis and Machine Intelligence* 11, pp. 859-872, 1989.
[155] C. Bezdek, Pattern Recognition with Fuzzy Objective Functions Algorithms, *Plenum Press, New York*, 1981.
[156] K. Suresh, R. Mohana and A. RamaMohanReddy. Improved FCM algorithm for Clustering on Web Usage Mining", *IJCSI International Journal of Computer Science Issues*, 8(1): 42-46, 2011.
[157] K. Kalti and M. Mahjoub. Image Segmentation by Gaussian Mixture Models and Modified FCM Algorithm, *The International Arab Journal of Information Technology*, 11(1):11-19, 2014.
[158] N. Ahmed, M. Yamany, N. Mohamed, and N.Farag. A Modified Fuzzy C-Means Algorithm for Bias Field Estimation and Segmentation of MRI Data, *IEEE Transaction on Medical Imaging*, 21(3): 193-199, 2002.
[159] W. Cai, S. Chen, and D. Zhang. Fast and Robust Fuzzy Cmeans Clustering Algorithms Incorporating Local Information for Image Segmentation, *Pattern Recognition*, 40(3): 825-838, 2007.
[160] S. Chuang, L. Tzeng, S. Chen, J. Wu, and T. Chen. Fuzzy C-Means Clustering with Spatial Information for Image Segmentation, *Elsevier Science*, 30(1): 9-15, 2006.

[161] C. Chen and Q. Zhang. Robust Image Segmentation using FCM with Spatial Constraints Based in New Kernel-Induced Distance Measure, *IEEE Transaction. Systems Man Cybernetics*, 34(4): 1907-1916, 2004.
[162] J. Hemanth, D. Selvathi, and J.Anitha. Effective Fuzzy Clustering Algorithm for Abnormal MR Brain Image Segmentation, *in Proceedings of IEEE International Advance Computing Conference, Patiala*, pp. 609-614, 2009.
[163] J. Kawa and E. Pietka. Image Clustering with Median and Myriad Spatial Constraint Enhanced FCM, *in Proceedings of the 4th International Conference on Computer Recognition Systems, Berlin*, vol. 30, pp. 211-218, 2005.
[164] A. Tolias and M. Panas. On Applying Spatial Constraints in Fuzzy Image Clustering using a Fuzzy Rule Based System, *IEEE Signal Processing Letters*, 5(10): 245-247, 1998.
[165] J. Wang, L. Dou, N. Che, D. Liu, B. Zhang, and J. Kong. Local Based Fuzzy Clustering for Segmentation of MR Brain Images, *in Proceedings of the 8th IEEE International Conference on Bioinformatics and Bioengineering, Athens*, pp. 1-5, 2008.
[166] K. Jain, N. Murty, and J. Flynn. Data Clustering: A Review, *ACM Computing Surveys*, 31(3): 264-323, 1999.
[167] M.Fonseca and J. Jorge. Using Fuzzy Logic to Recognize Geometric Shapes Interactively, *The Ninth IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 291-296, 2000.
[168] S. Sasi and J.S. Bedi. Handwritten Character Recognition using Fuzzy Logic, *Proceedings of the 37th Midwest Symposium on Circuits and Systems*, vol. 2, pp. 1399-1402, 1994.
[169] R.Deriche, Fast Algorithms for Low-Level Vision, *IEEE Trans. on PAMI*, 12(1), pp. 78-87, January 1990.
[170] http://zone.ni.com/reference/enXX/help/372656B01/lvasptconcepts/wa_application/
[171] http://users.fmrib.ox.ac.uk/~steve/susan/susan/node14.html.
[172] http://www.cs.utah.edu/~manasi/coursework/cs7960/p1/project1.html.
[173] http://dsp.stackexchange.com/questions/1714/best-way-of-segmenting-veins-in-leaves.
[174] P. Bonnin, Méthode Systématique de Conception et d'Applications en Vision par Ordinateur, *PhD thesis*, January 1991.
[175] http://wwwmath.tau.ac.il/~turkel/notes/Maini.pdf.
[176] J.B.A. Maintz and M. A. Viergever. A Survey of medical Image Registration, *Medical Image Analysis*, vol 2. Oxford University Press, 1998.
[177] J.V.Chapnick, M.E.Noz, G.Q. Maguire, E.L.Kramer, J.J.Sanger, B.A.Birnbaum, A.J.Megibow, Techniques of Multimodality Image Registration, *Proceedings of the IEEE nineteenth Annual North East Bioengineeering conference*, 18-19 March 1993.
[178] M. Wyawahare, P. Patil and H. Abhyankar, Image Registration Techniques: An overview, *International Journal of Signal Processing, Image Processing and Pattern Recognition,* vol. 2, no.3, September 2009.
[179] U. Maulik and S. Bandyopadhyay, Genetic algorithm based clustering technique, *Pattern Recognition*, vol.33, pp. 1455–1465, 2000.
[180] J.Rourke, Computarionul geomerry in C. *Cambridge University Press*, 2nd edition, 1998.
[181] A. Bykat, Convex hull of a finite set of points in two dimensions, *Inform. Process. Lett.,* vol. 7, pp. 296–298, 1978.

[182] A. Jain, *Fundamentals of Digital Image Processing*, *Prentice-Hall*, Chap. 9, 1989.
[183] S. Pal. *Some Low Level Image Segmentation Methods, Algorithms and their Analysis*. *PhD thesis, Indian Institute of Technology*, 1991.'
[184] S. Annadurai, R. Shanmugalakshmi, Fundamentals of Digital Image Processing, *ebook by Pearson Education India*, 2006.
[185] W.K. Pratt, Digital Image Processing, 2nd ed., Wiley, New York, 1991.
[186] D.I. Barnea, H.F. Silverman, A class of algorithms for fast digital image registration, *IEEE Transactions on Computing*, vol. 21, 1972.
[187] R.N. Bracewell, The Fourier Transform and Its Applications, *McGraw-Hill*, New York, 1965.
[188] A. Goshtasby, G.C. Stockman, Point pattern matching using convex hull edges, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 15, 1985.
[189] G. Stockman, S. Kopstein, S. Benett, Matching images to models for registration and object detection via clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, 1982.
[190] H.G. Barrow, J.M. Tenenbaum, R.C. Bolles, H.C. Wolf., Parametric correspondence and chamfer matching: Two new techniques for image matching, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Massachusetts, 1977.
[191] G. Borgefors, Hierarchical chamfer matching: a parametric edge matching algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, 1988.
[192] Barber, C. Bradford, Dobkin, P.David, Huhdanpaa, Hannu, The quickhull algorithm for convex hulls, *ACM Transactions on Mathematical Software*, vol. 22, no. 4, 1996.
[193] Uchida, Seiichi. A Survey of Elastic Matching Techniques for Handwritten Character Recognition. *IEICE TRANS. INF. & SYST*. E88-D, vol. 8, 2005.
[194] Z. Zhang. Le problème de la mise en correspondance: L'état de l'art, *unité de recherche INRIA Sophia-Antipolis*, 1993.
[195] T. Ea, L. Lacassagne, P. Garda, Execution temps reel des detecteurs de contours de Deriche par des processeurs RISC, *Congrès Adéquation Algorithme Architecture*,France,1998,
(http://www.ief.upsud.fr/~lacas/Publications/AAA98.pdf)
[196] J. Zhao and B.K. Bose, Evaluation of membership functions for fuzzy logic controlled induction motor drive, *the 28th IEEE Annual Conference of the Industrial Electronics Society*, vol. 1, pp. 229-234, 2002.
[197] J. Garibaldi and R. John, Choosing Membership Functions of Linguistic Terms, *the 12th IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 578-583, 2003.
[198] R.Nachar, E.Inaty, P.Bonnin and Y.alayli. "A robust edge based corner detector (EBCD)", *accepted for publication in IJIG journal,* April 2014.
[199] R.Nachar, E.Inaty, P.Bonnin and Y.alayli. "Towards an automatic image co-registration technique using edge dominant corners primitives", *accepted for publication in ICAE journal,* May 2014.
[200] R.Nachar, E.Inaty, P.Bonnin and Y.alayli. "Edge Corner Based Segmentation Technique to Break down CAPTCHA Using Fuzzy Logic", *accepted for publication in JETWI journal,* July 2014.
[201] A. Borji. "Boosting bottom-up and top-down visual features for saliency estimation", in *Proc. IEEE CVPR*, pp. 438-445, June 2012.

[202] S.Lee, J.Kim, K.Choi, J.Sim and C.Kim. "Video saliency detection based on spatiotemporal feature learning", in *Proc. IEEE ICIC*, pp. 1120-1124, October 2014.
[203] S.Aly, A.Youssef and L.Abbott. "Adaptive feature selection and data pruning for 3D facial expression recognition using the kinect", in *Proc. IEEE ICIC*, pp.1361-1365, October 2014.
[204] X. Li, Q. Ruan, and Y. Ming. "3D facial expression recognition based on basic geometric features", in *Proc. IEEE ICSP*, pp.1366–1369, 2010.
[205] H. Soyel and H. Demirel. "Optimal feature selection for 3D facial expression recognition using coarse-to-fine classification", *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 18, no. 6, pp. 1031–1040, 2010.
[206] A. Alshennawi and A. Aly. "Edge Detection in Digital Images Using Fuzzy Logic Technique", *Word Academy of Science, Engineering and Technology*, vol. 3, pp. 166-174, 2009.
[207] K. Reddy, D. Rao and K. Rajesh. "Hand Written Character Detection by Using Fuzzy Logic Techniques", *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, pp. 256-261, March 2013.
[208] L. Zadeh. "Fuzzy sets", *Information and Control*, vol. 8, pp: 338–353, 1965.
[209] D. Patel and S. More. "Edge Detection Technique by Fuzzy Logic and Cellular Learning Automata using Fuzzy Image Processing", *International Conference on Computer Communication and Informatics (ICCCI)*, Jan 2013.