République algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche
Scientifique

Inf.IA.03/17
جامعة جيجل
كلية العلوم الدقيقة والإعلام الآلي
المكتبة
رقم الجرد: 00992

Université Mohamed Seddik Ben Yahia- Jijel-
Faculté des Sciences Exactes et Informatique
Département de l'Informatique

Mémoire
De Fin d'Etude pour l'obtention du Diplôme de Master en Informatique

Spécialité: Intelligence Artificielle

Thème

Implémentation Et Emulation De Détecteur Harris Pour Smartphone

Réalisé Par :

- **BOULESBIAAT SABRINA**
- **BOUKABOU MANEL**

Encadré Par :

Dr. TAFFAR MOKHTAR

جامعة محمد الصديق بن يحي - جيجل
مكتبة الكلية
02
كلية العلوم الدقيقة والإعلام الآلي

Année universitaire

2016-2017

Remerciement

Je commence par remercie ***dieu****, de m'avoir donné le courage, volonté et patience pour mener a bon terme ce travail.*

Je tiens de remerciés ***Mr. Taffar Mokhtar*** *pour avoir bien accepté de diriger ce travail, pour son soutient, son sérieux, sa disponibilité, ses précieux conseils et son aide tout au long de l'élaboration de ce travail.*

Nous remercions les membres de jury qui ont bien voulu examiner et évaluer ce mémoire.

Nous nous acquittons, enfin, avec plaisir à tous nos enseignants pour la qualité de l'enseignement afin de nous fournir une formation riche.

Dédicace

A mes chers parents,

Que nul dédicace ne puisse exprimer ce que nous leurs doit, pour leur bienveillance, leur affection et leur soutien, en témoignage de nos profonds amours et nos grandes reconnaissances « Que Dieu vous garde et vous protège ».

A ma chère sœur et frère,

A tous mes amies,

Pour leur encouragement et leur soutien moral durant l'élaboration de ce projet de fin d'études.

MANEL

Résumé

Dans le cadre de notre préparation du diplôme de Master Informatique, option intelligence artificielle nous avons été amenés à effectuer notre projet de fin d'étude qui s'agissait d'étudier et de développer une application mobile sous Android. Aujourd'hui, Pour les entreprises, les marques, les chercheurs (ex., en botanique) et mêmes les agences touristiques, les applis mobiles sont devenues un moyen de création de services pour les consommateurs et les mobinautes. Les applications et sites mobiles permettent de consulter du contenu, de fournir un service adapté, de renseigner sur l'historique d'un immeuble pris en image directement sur les mobiles de type Smartphones et tablettes. En vision par ordinateur, les Smartphones sont un moyen de reconnaitre le type d'une plante sur une image prise en randonnée, la structure finie d'un projet en réalisation (par réalité augmentée), etc.

Par ce projet, nous commencerons par comprendre et programmer le détecteur Harris. Ce dernier nous permet d'extraire des points caractéristiques locaux d'une image afin pouvoir faire un appariement d'autres points sur d'autres images et pourvoir ainsi détecter les objets dans une base de donnée. Nous essayerons ensuite dans une autre partie, d'adapter ce programme en une application mobile sur une plateforme Androïd.

Ainsi, les objectifs fixés consistent à programmer le détecteur Harris sous forme d'une application destinée à la plateforme Androïd ; ensuite, émuler notre application sur le PC. Enfin, tester et valider l'application puis assurer sa portabilité effective sur un Smartphone.

Abstract

As part of our preparation of the Master's Degree in Computer Science, option: Artificial Intelligence we were led to carry out our end-of-study project which was to study and develop a mobile application on Android. Today, for businesses, brands, researchers (eg in botany) and even tourism agencies, mobile apps have become a means of creating services for consumers and mobile users. Mobile applications and sites allow users to view content, provide an adapted service, and provide information about the history of an immovable property directly on mobile phones such as smartphones and tablets. In computer vision, smartphones are a way of recognizing the type of a plant on an image taken in hiking, the finished structure of a project in realization (by augmented reality), and so on.

Through this project, we will begin by understanding and programming the Harris detector. The latter allows us to extract local points of an image in order to be able to make a pairing of other points on other images and thus be able to detect the objects in a database. We will then try in another part, to adapt this program to a mobile application on an Android platform.

Thus, the objectives set are to program the Harris detector in the form of an application intended for the Android platform; then emulate our application on the PC. Finally, test and validate the application and then ensure its effective portability on a Smartphone.

Liste des figures

Chapitre 01 : les détecteurs des points d'intérêt

Chapitre 02 : Détecteur de Harris

Chapitre 03 : Environnement de développement Android

Chapitre 04 : Implémentation et résultats

Table des matières

جامعة محمد الصديق بن يحي - جيجل
مكتبة الكلية
02
كلية العلوم الدقيقة و الإعلام الآلي

Chapitre 04 : Implémentations et résultats

1 Introduction générale

Le téléphone portable est devenu un véritable phénomène de société. Véritable bijou technologique, il connait un succès inimaginable. Cet objet a bouleversé la manière de communiquer. L'utilisation du téléphone devient de plus en plus fréquente, et l'obtention d'un Smartphone est devenue incontournable.

L'intelligence artificielle est définit comme « la construction de programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies par des êtres humains car elles demandent des processus mentaux de haut niveau tels que: l'apprentissage perceptuel, l'organisation de la mémoire et le raisonnement critique » [1,2]. On y trouve donc le côté « artificiel » atteint par l'usage des ordinateurs ou de processus électroniques élaborés et le côté « intelligence » associé à son but d'imiter l'intelligence humaine. Cette imitation peut se faire dans le raisonnement, par exemple, dans les jeux ou la pratique des mathématiques, dans la compréhension des langues naturelles, dans la perception : visuelle (interprétation des images et des scènes), auditive (compréhension du langage parlé) ou par d'autres capteurs, dans la commande d'un robot dans un milieu inconnu ou hostile.

La vision par ordinateur est une branche de l'intelligence artificielle dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs images prises par un système d'acquisition (par exemple : caméras, etc.)[3].

Une approche consiste à tenter d'imiter la vision humaine, il ne s'agit pas de fournir une explication de comment marche la vision biologique mais de créer un modèle qui vu de l'extérieur possède des propriétés semblables. Ce modèle artificiel peut-il être d'une utilité importante dans la détection et la reconnaissance d'objet [4].Cette manière de procéder peut être perçue comme un traitement des données visuelles par le biais de modèles fondés sur la géométrie, la physique, la biologie, les statistiques et la théorie d'apprentissage.

La vision par ordinateur cherche à appliquer ses théories et ses modèles à différents systèmes. Quelques exemples de systèmes d'application de la vision par ordinateur: Procédés de contrôle; Navigation; reconnaissance d'objets, l'estimation de mouvement, etc.

La reconnaissance de formes ou objet est un ensemble de techniques et méthodes visant à identifier des modèles informatiques à partir de données brutes afin de prendre une décision dépendant de la catégorie attribuée à ce modèle [5]. On considère que c'est une branche de l'intelligence artificielle qui fait largement appel aux techniques d'apprentissage automatique et aux statistiques.

La détection des zones d'intérêts est une des méthodes de la reconnaissance des formes qui consiste à mettre en évidence des zones d'une image jugées « intéressantes » pour l'analyse, c'est-à-dire présentant des propriétés locales remarquables. Selon la méthode de détection utilisée, les zones peuvent être de différentes natures : Contours, Points d'intérêt, Régions d'intérêt, constituant le résultat de la détection, et qui peut être utilisée comme point de départ de nombreux algorithmes de traitement d'images.

La détection des zones d'intérêt commencent par identifier des points d'intérêt qui vont se révéler être des sortes de barycentres des régions recherchées, telles que les méthodes multi-échelles fondées sur l'étude des détecteurs de points d'intérêt tel que Harris, DoG (Différence de Gaussienne), à différentes échelles de l'image. Ceci permet d'obtenir des régions soit circulaires soit elliptiques, selon le niveau

de raffinement voulu. Ces méthodes sont souvent intégrées à des algorithmes plus généraux tels que SIFT ou SURF, qui incluent un descripteur de région d'intérêt en plus d'un détecteur.

L'extraction du descripteur à une image consiste à calculer sur chaque zone détectée ce que l'on appelle un vecteur caractéristique, qui résume le contenu de la zone en question.

L'implémentation de ces algorithmes est possible et efficace soit sur ordinateur ou sur Smartphone sous Androïd qui dans sa genèse, part d'une idée de base simple, et très vite son succès fut tel qu'il a su devenir un incontournable du monde mobile.

2 Motivation

Nous allons dans ce travail s'intéresser à la détection des zones d'intérêt par points d'intérêt car ils sont une sources d'informations plus fiable que les contours, pas de contraintes sur la fonction d'intensité, robuste aux occultations, transformation photométrique et Changement d'échelle, pas d'opérations de chainage par rapport à l'approche contour et présents dans une grande majorité d'images en comparaison avec l'approche contour [6,7].
Ils fournissent une information non dense mais sûre, calculer des transformations inter-images, calculer le point de vue. Ce que permet de mettre facilement en correspondance deux images [8], ils sont aussi les plus rentables en termes de temps d'exécution [7].
La mise en correspondance est très utile pour la reconstruction de troisième dimension ainsi que la construction des panoramas et la reconnaissance des objets.
Nous avons choisi le détecteur Harris de points d'intérêt, il possède une bonne répétabilité et une invariance à la rotation, cependant il souffre de quelques limites: Sensibilité au bruit à cause de la dérivation, coût élevé en temps de calcul à cause de la convolution avec une gaussienne, sensibilité au changement d'échelle, sensibilité aux transformations affines.
Par ailleurs, la valeur de k utilisée par Harris-Stephens est empirique, elle n'a pas été prouvée analytiquement [7].
Et avec l'explosion des ventes de Smartphones ces dernières années les aplis Smartphone ont pris une place importante dans la vie quotidienne de millions de personnes, au point qu'il s'agit du système d'exploitation mobile avec le plus d'applications en circulation.
La reconnaissance visuelle d'images sur Smartphone est un domaine en pleine effervescence. Depuis Google Goggles qui a été la première application grand public populaire à offrir ce service, on trouve aujourd'hui une pléthore d'applications permettant de reconnaître des tableaux dans des musées, des couvertures de livres, des pochettes de CD ou de DVD, des publicités, etc. De nombreux magasins comme Kiabi, But, IKEA, Leroy Merlin2 enrichissent déjà leur catalogues avec du contenu destiné aux Smartphones accessible par reconnaissance d'images [9].
C'est pour ça nous allons développer le détecteur de points d'intérêt Harris comme étant une application mobile sous androïd qui est dès le quatrième trimestre 2010 devient le système d'exploitation mobile le plus utilisé au monde, on le retrouve non seulement dans les tablettes et Smartphones, mais aussi dans les téléviseurs, les consoles de jeux, les appareils photos, etc.

3 Organisation du mémoire

La suite de notre mémoire sera structurée comme suit :

Au premier chapitre

Intitulé « Détecteurs des points d'intérêts » nous présentons les détecteurs les plus connus, leur principe de base ainsi que leurs caractéristiques illustrées par des images.

Au deuxième chapitre

Nous parlerons dans ce chapitre intitulé « Le détecteur Harris » plus en détaille du détecteur Harris, les efforts fournis pour balayer ses limites par différents alternatifs ainsi ces caractéristiques.

Au troisième chapitre

Intitulé « Android Studio », nous présentons le système Androïd et l'Androïd studio qui est un environnement matériel et logiciel de développement des applications pour Androïd, les étapes d'installation et de configuration de l'environnement et les différents outils proposés. Nous se limitons aux notions utilisés dans notre application.

Au quatrième chapitre

Intitulé « Implémentation et résultats » nous présentons notre application et les résultats obtenus ainsi que notre expérimentation concernant la détection des points d'intérêt Harris sur des objets de la base de données VOC 2006.

Enfin, nous clôturons ce mémoire par une conclusion dans laquelle nous résumons notre travail.

1.1 Introduction

Un point d'intérêt est un point caractéristique de l'image, qui porte avec son voisinage une information riche, exploitable par les applications de plus haut niveau.

Il est souvent confondu avec un coin. Un point d'intérêt peut aussi être un point de jonction, un blob, un T voire un point de contour. Les premières questions qui se posent, et qui sont légitimes : Qu'est-ce qu'un point d'intérêt ? Quelle est son utilité ?

Dans ce qui suit, nous présentons un état de l'art détaillé sur les détecteurs les plus répandus.

1.2 Les points d'intérêt

1.2.1 Définitions

D'une façon générale, un point d'intérêt est défini comme un point ou le signal d'image change fortement dans plusieurs directions. Alors, un point d'intérêt peut être un pic, un coin ou une jonction. En vision par ordinateur, le terme point d'intérêt est souvent référencé au terme coin.

Avec la projection d'une scène 3D dans le monde réel en plan 2D (image), un coin dans l'image ne correspond pas souvent à un coin physique et inverse. Des faux coins devrait être enlevés parce qu'ils provoquent de fausses correspondances des images, donc rendent moins fiable la reconnaissance.

Moravec définit un point d'intérêt comme étant un point où la valeur moyenne de l'intensité du voisinage varie considérablement dans les quatre directions horizontales, verticales et diagonales. Pour le fait qu'un point d'intérêt est un point qui se distingue de son voisinage et qui est riche en termes d'information. [8] stipule qu'un point d'intérêt est associé à une discontinuité des niveaux de gris, de la texture, de la géométrie, etc. Cette définition est plus générale en termes de type de points d'intérêt détectables dans une image, et plus précise en considérant les éléments intrinsèques de l'image.

1.2.2 Différentes approches

De nombreuses méthodes ont été proposées pour détecter des points d'intérêts. Elles peuvent être classées grossièrement suivant trois catégories :

- Approches contours : l'idée est de détecter les contours dans une image dans un premier temps. Les points d'intérêts sont ensuite extraits le long des contours en considérants les points de courbures maximales ainsi que les intersections de contours.

- Approches intensité : l'idée est cette fois-ci de regarder directement la fonction d'intensité dans les images pour en extraire directement les points de discontinuités.
- Approches à base de modèles : les points d'intérêts sont identifiés dans l'image par mise en correspondance de la fonction d'intensité avec un modèle théorique de cette fonction du point d'intérêts considérés.

1.2.3 Utilité de la détection d'un point d'intérêt

En vision par ordinateur, traiter l'image en travaillant directement sur les pixels est déconseillé pour plusieurs raisons. D'abord en terme de qualité de résultats, car ils seront vulnérables à n'importe quel bruit si petit qu'il soit, puis en terme de temps de traitement, chose qui rend les résultats inexploitables. Par conséquent, le fait de résumer l'information contenue dans l'image devient une nécessité. Plusieurs outils caractéristiques sont proposés: points d'intérêt, contours, régions, etc.

Ces informations seront efficaces pour plusieurs types d'applications comme le recalage d'images, la mise en correspondance, le suivi d'objet, l'indexation, reconnaissance d'objet ou robotique mobile. Ceci étant, l'approche se basant sur les points d'intérêt est plus fiable car elle est plus robuste aux bruits et invariante aux transformations usuelles (rotation, changement de vue,...). Pour cette raison, plusieurs chercheurs ont consacré à la recherche des algorithmes pour détecter des points de façons plus fiables, stable et invariant sous quelques transformations.

1.2.4 Critères de bonne détection

Devant la difficulté de tirer objectivement des conclusions sur les résultats de détection des points d'intérêt, plusieurs critères d'évaluation ont été proposés. Les critères les plus utilisés sont ceux de la répétabilité introduit par Schmid et de la localisation.

La mesure de la répétabilité indique la probabilité de reconnaitre le même point d'intérêt dans des situations où l'image subit une des transformations ou déformations usuelles comme :

- **Rotation**

Figure 1.1 : Rotation de l'image.

➢ **Changement de vue** lors du changement de la vue, il se peut que le voisinage du point en question change et par conséquent le comportement du détecteur n'est plus le même (voir figure 1.2). Dans ce cas il faut faire attention au phénomène d'occlusion.

Figure 1.2 : Changement du point de vue d'une image.

➢ **Transformation photométrique**

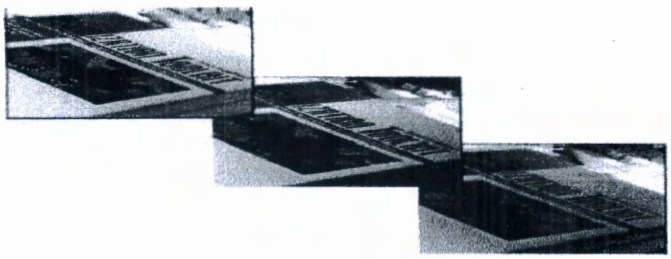


Figure 1.3 : Changement de caractéristiques photométriques de l'image.

➢ **Changement d'échelle** à ne pas confondre avec le zoom où la taille de l'image reste la même. Le changement d'échelle se produit lors du déplacement du capteur ou du changement de ses caractéristiques intrinsèques.

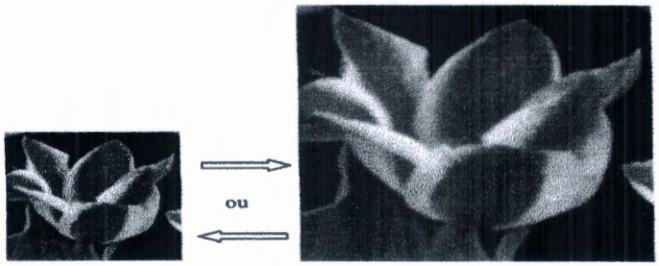


Figure 1.4 : Changement d'échelles dans les deux directions.

1.2.5 Exemples d'utilisation des points d'intérêt

- **Construction de panorama**

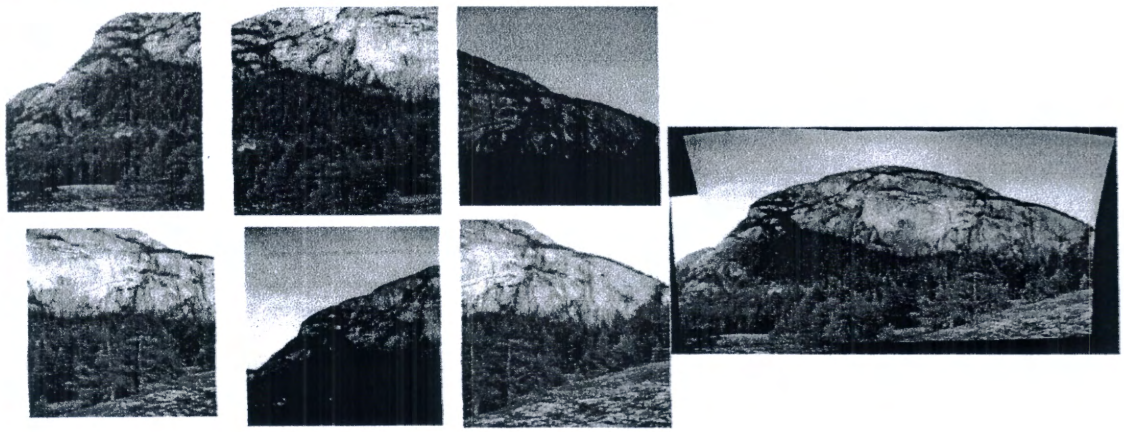


Figure 1.5 : Construction de panoramas .

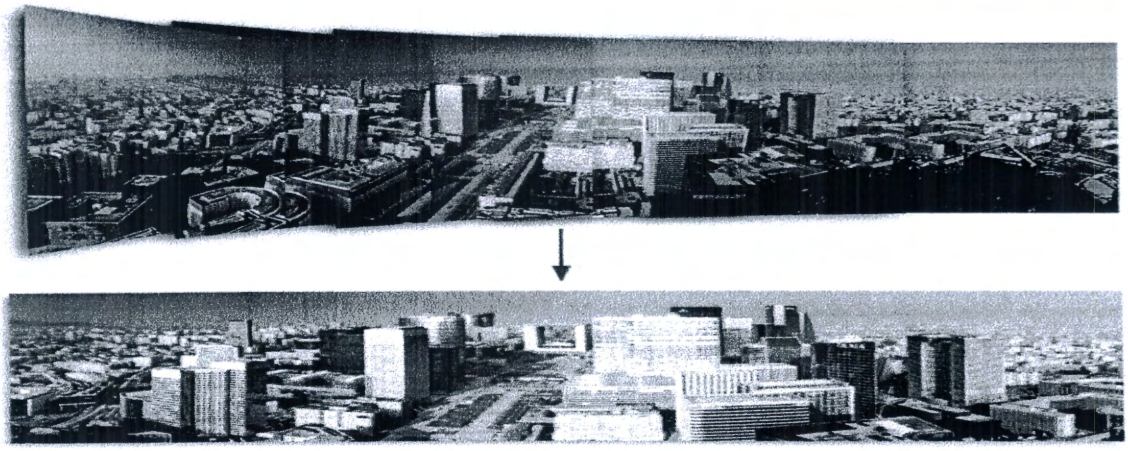


Figure 1.6 : Construction de panoramas.

- **Fabriquer un panoramique**

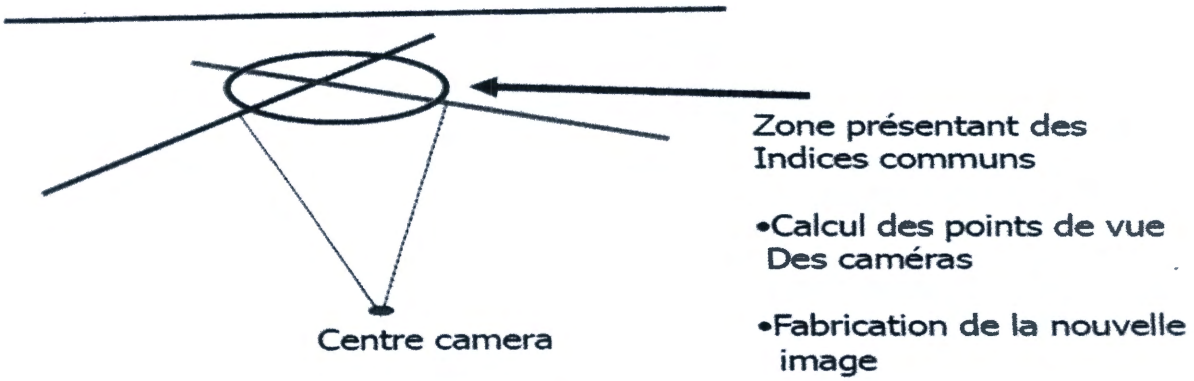


Figure 1.7 : Fabriquer un panoramique.

- **Extraction des points d'intérêt**

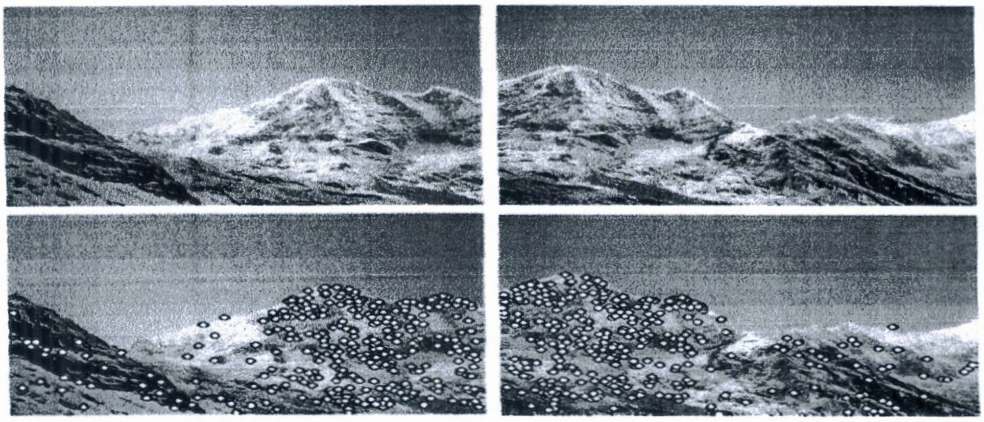


Figure 1.8 : Extraction des points d'intérêt.

- **Détection des points en correspondance (avec l'hypothèse d'homographie)**

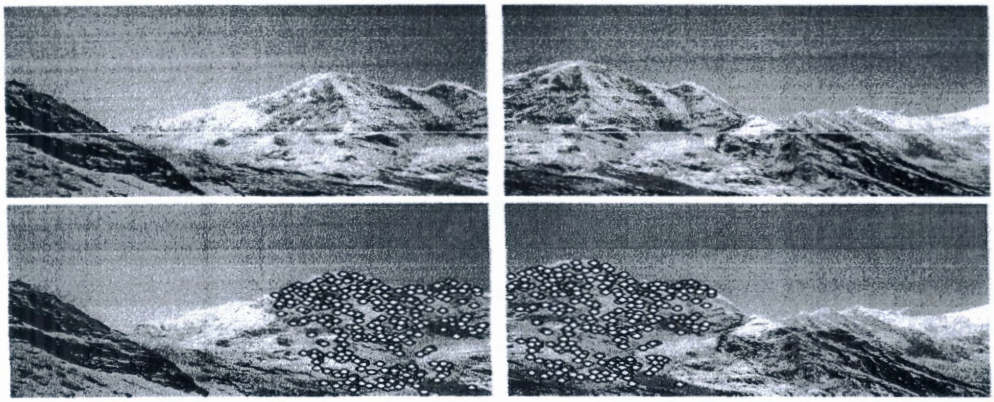


Figure 1.9 : Détection des points en correspondance.

- **Recollement des images**

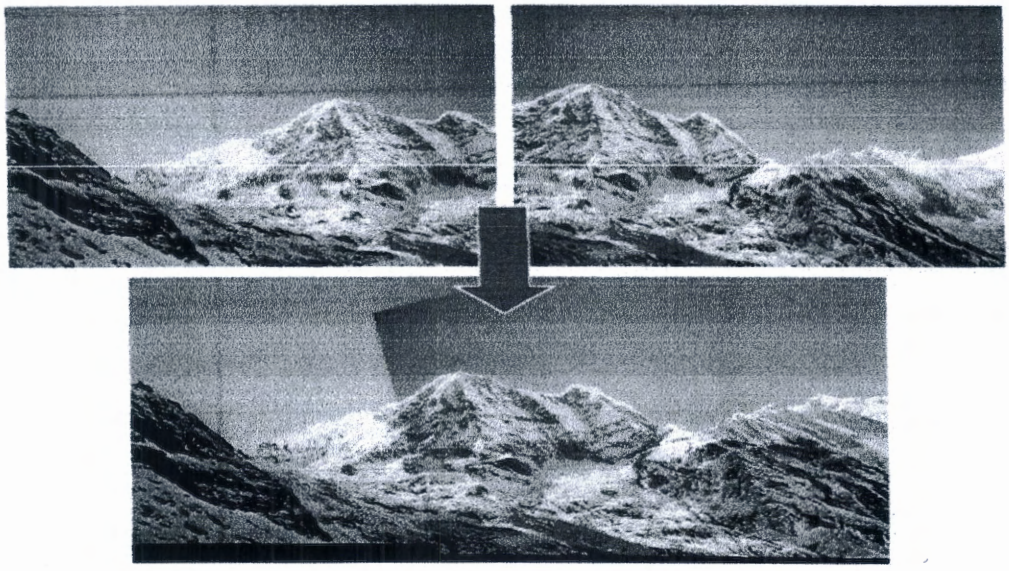


Figure 1.10 : Recollement des images.

1.3 Quelques détecteurs de points d'intérêt

Nous pouvons scinder la majorité des détecteurs de points d'intérêt en deux grandes familles. Le premier est celle des détecteurs qui se basent sur le changement d'apparence, quant à la deuxième est celle des détecteurs se basant sur des fondements mathématiques. L'idée de base de la première famille est plutôt intuitive puis traduite par des équations mathématiques, alors que les détecteurs de la deuxième famille ont été basés sur des fondements mathématiques, notamment sur la notion de dérivation .

1.3.1 Détecteur de Moravec

Nous ne pouvons pas parler des détecteurs de points d'intérêt sans citer le détecteur de Moravec (1981), souvent considéré comme à l'origine de cette discipline. Même s'il n'est pas de la popularité du détecteur de Harris, SUSAN ou autres, le détecteur de Moravec reste la référence et le point de départ de l'approche de détection de points d'intérêt.

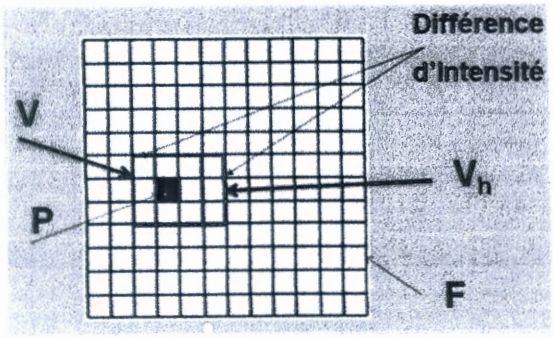
Dans sa thèse, Moravec présente une idée inouïe concernant les caractéristiques locales de l'image à savoir les points d'intérêt. Il considère qu'un point d'intérêt de l'image est un point où la valeur moyenne de l'intensité de son voisinage varie considérablement dans les quatre directions : horizontale, verticale, et diagonales (figure 1.12). Idée de base observer les changements d'intensité en déplaçant localement une fenêtre.

Figure1.11 : différents cas de changement d'intensité.

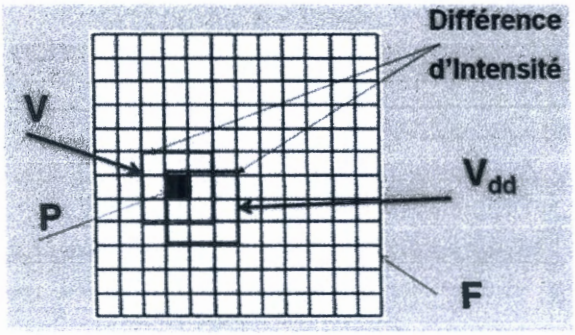
La méthode de Moravec :

- **Etape 1 :** Déterminer une fenêtre F de taille 12x12 pour parcourir l'image I. Toutes les étapes de détection suivantes s'effectuent dans cette fenêtre glissante.

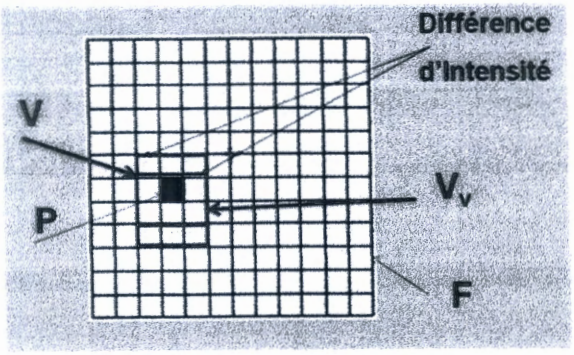
- **Etape 2 :** Pour chaque point P de la fenêtre F, déterminer un voisinage V centré en P de taille 3x3 et quatre autres voisinages V_h, V_v, V_d et V_g,de même taille, qui représentent respectivement le décalage de V dans quatre directions : Horizontale (0°), verticale (90°) et diagonales (45° et 135°).



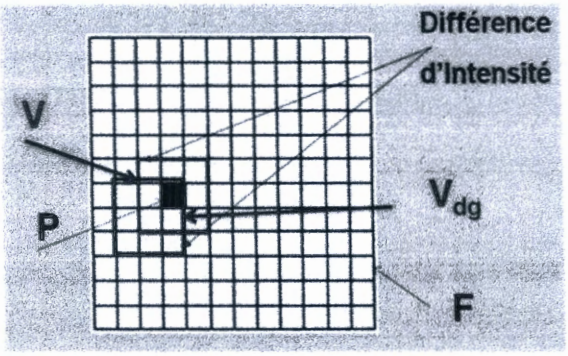
a. Décalage horizontal de V (0°)



b. Décalage diagonal de V (45°)



c. Décalage vertical de V (90°)



d. Décalage diagonal de V (135°)

Figure 1.12. Calcul dans la fenêtre F de la différence d'intensité entre le voisinage V du point P et les voisinages décalés respectivement dans les quatre directions.

- **Etape 3 :** Calculer via la fonction E (équation 1.1) le changement moyen d'intensité respectif entre V et les quatre voisinages cités ci-dessus (Figure1.12).

$$E(x,y) = \sum_{u,v} w(u,v)\,|I(x+u, y+v) - I(u,v)|^2 \quad (1.1)$$

Telles que :

- (u,v) représente les coordonnées absolues des pixels dans l'image.
- w est la fonction qui représente le masque pour limiter les calculs sur le voisinage V.

Elle vaut 1 si (u,v) ϵ V et 0 sinon.

- (x,y) représente la direction de la translation. Ses valeurs possibles sont :
- (1,0) : translation horizontale selon l'axe X. (0°)
- (1,1) : translation diagonale selon les axes X et Y. (45°)
- (0,1) : translation verticale selon l'axe Y. (90°)
- (-1,1) : translation diagonale selon les axes X (dans le sens inverse) et Y (135°)

- I est la fonction qui représente l'intensité des pixels de l'image.

Après le calcul de E, nous sommes devant l'une des trois situations suivantes (Figure1.13):

1. La fonction E prend de faibles valeurs dans toutes les directions (x,y): c'est le cas d'une zone homogène.
2. La fonction E varie considérablement selon l'un des axes X,Y sans l'autre. C'est le cas d'un contour horizontal (cas 2 dans la figure1.13) ou vertical.
3. La fonction E varie fortement dans les quatre directions considérées. Dans ce cas, Moravec déclare le pixel en question comme point d'intérêt candidat et garde la plus petite valeur de E.

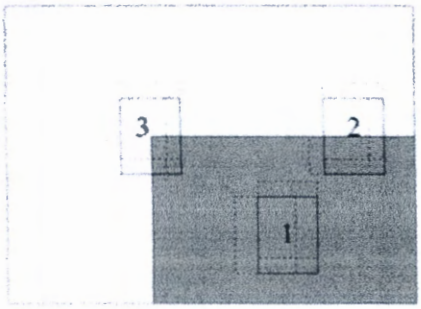


Figure 1.13 : Situations possibles lors du calcul de la fonction E dans les quatre directions.

- **Etape 4 :** Après le parcours de la fenêtre F (voir étape 1), plusieurs points peuvent être déclarés points d'intérêt candidat. Dans cette étape Moravec prend la valeur minimale de E notée dans l'étape 3 de chaque candidat, et garde la plus grande valeur minimale. Le point en question est déclaré définitivement point d'intérêt.

L'image suivante présente le résultat du détecteur de Moravec

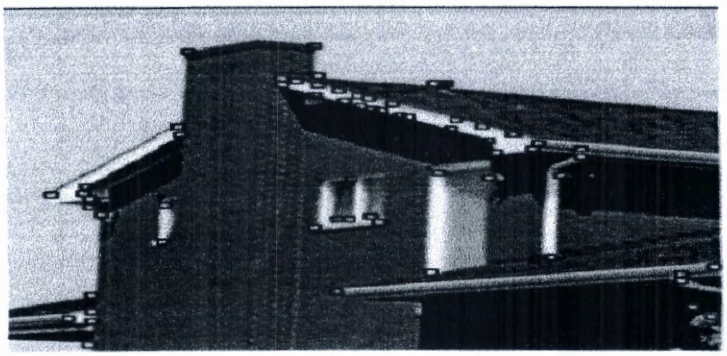


Figure 1.14 : Résultat détecteur Moravec.

Algorithme du détecteur de Moravec [14]

1. Pour chaque pixel (x; y) d'une image, calculer 8 valeurs de E différentes correspondant à 8 déplacements en x et y :

   $E1(x; y) = E_{1,0}(x, y)$

   $E2(x; y) = E_{1,1}(x,y)$

   $E3(x; y) = E_{0,1}(x,y)$

   $E4(x; y) = E_{1,-1}(x,y)$

   $E5(x; y) = E_{-1,1}(x,y)$

   $E6(x; y) = E_{0,-1}(x,y)$

   $E7(x; y) = E_{-1,0}(x,y)$

   $E8(x; y) = E_{-1,-1}(x,y)$

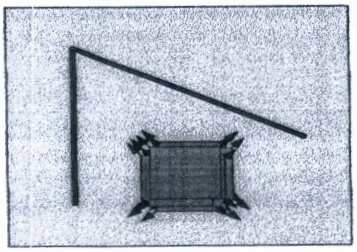
2. Pour chaque pixel on calcule la valeur minimum des Ei.
3. On garde comme point d'intérêt les points pour lesquels ce minimum est supérieur à un seuil. Selon le seuil on aura plus ou moins de points d'intérêts

1.3.2 Détecteur de Harris

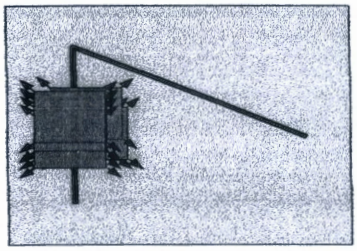
Avec Stephens, Harris a proposé en 1988 un algorithme de détection qui est certainement aujourd'hui le plus connu et qui est toujours très utilisé car il donne de très bons résultats et est assez simple a mettre en œuvre. Les points d'intérêts recherchés sont des points au voisinage lesquels l'image varie significativement dans plusieurs directions. Ce peut être des coins, des jonctions en T, des jonctions en Y, etc. Comme souvent pour les algorithmes de détections de primitive, il faut éviter de retenir des points situés sur un contour d'un objet pour deux principales raisons

1. Lorsque l'objet est observé d'un point de vue différent, le bord de l'objet sur l'image ne correspond plus à la même zone physique dans la scène.

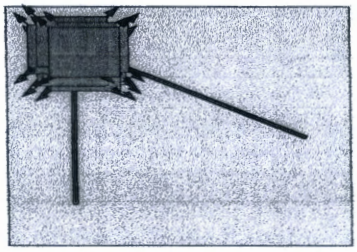
2. Un point situé sur un contour n'est généralement pas stable car les points du même bord situes dans son voisinage ont souvent une apparence similaire.



Zone "homogène" : aucun changement dans aucune direction



"contour" : changement dans une seule direction



"coin" : changements dans plusieurs directions

Figure.1.15 : Détecter les coins sans retenir les contours.

Les améliorations proposées par Harris et Stephens :

- **Réponse anisotropique**: c'est-à-dire que le détecteur n'a pas le même comportement dans toutes les directions. Ceci est dû au fait que Moravec décale les voisinages dans des directions multiples de 45°.
- **Solution :**

Développement de Taylor de la fonction d'intensité au voisinage du pixel (u,v) :

$$I(x+u,y+v) = I(u,v) + x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta x} + o(x^2+y^2) \qquad (1.2)$$

Et par suite la nouvelle formule de la fonction E est :

$$E(x,y) = \sum_{u,v} w(u,v)\,[x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta x} + o(x^2+y^2)]^2 \qquad (1.3)$$

En négligent le terme $o(x^2,y^2)$ l'expression de $\bar{E}$ devient

$$E(x,y) = Ax^2 + 2Cxy + By^2 \qquad (1.4)$$

Par conséquent le couple (x,y) peut prendre plusieurs valeurs, ce qui rend le détecteur de Harris moins anisotropique que Moravec.

- **Réponse bruitée :** Du fait que la fenêtre utilisée par Moravec soit carrée et binaire, l'estimation de la variation d'intensité est bruitée. Harris et Stephens voulaient la remplacer par une fenêtre circulaire qui ne soit pas binaire pour donner plus de poids informatif aux pixels proches du centre.
- **Solution :**

Utilisation d'un filtre gaussien qui répond aux deux remarques faites :

$$w(u,v) = \exp-(u^2+v^2)/2\sigma^2 \qquad (1.5)$$

- **Réponse trop forte aux contours :** Ceci est dû au fait que seul le minimum de E (équations 1.1) est pris en considération .

Harris – Stephens proposent la forme matricielle suivante qui décrit la surface de l'image à un point donné :

$$E(x,y) = (x,y) * M(x,y)^t \qquad (1.6)$$

Telle que :

$$M = \begin{bmatrix} A & \cdots & C \\ \vdots & \ddots & \vdots \\ C & \cdots & B \end{bmatrix} \qquad (1.7)$$

Les valeurs propres de M sont proportionnelles à la courbure de la surface :

- Si les deux valeurs propres ont de faibles valeurs, alors la surface en question représente une zone homogène.
- Si l'une des valeurs propres sans l'autre possède une grande valeur, alors la surface contient un contour.
- Si les deux valeurs propres possèdent de grandes valeurs, alors la surface contient un coin.

Le calcul des valeurs propres est coûteux, cependant un calcul équivalent est effectué sur la mesure R proposée par Harris-Stephens :

$$R = Det(M) - k\,(Trace(M))^2 \qquad (1.8)$$

Telles que : $Det(M) = AB - C^2$ et $Trace(M) = A+B$

Sachant que si λ_1 et λ_2 sont les valeurs propres de M et puisque

$$Det(M) = \lambda_1.\lambda_2 \text{ et } Trace(M) = \lambda_1 + \lambda_2 \qquad (1.9)$$

Alors, le raisonnement ci-haut est équivaut à :

- Si R possède une petite valeur, alors la surface en question représente une zone homogène.
- Si $R < 0$ alors la surface contient un contour.
- Si $R > 0$ alors la surface contient un coin.

Nous abordons plus en détail ce détecteur dans le chapitre03 ainsi que l'étape de la mise en correspondance utilisée pour la comparaison entre deux images après la détection des points d'intérêts.

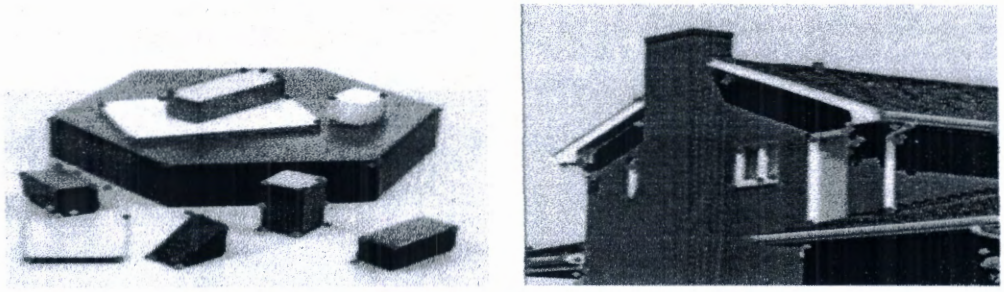


Figure.1.16 : Points d'intérêt extraits par le détecteur de Harris.

1.3.3 Détecteur SUSAN

Etabli par Smith-Brady, SUSAN (Smallest Univalue Segment Assimilating Nucleus) (1997) [7] n'est pas qu'un détecteur de coins, il est aussi un détecteur de contour [11]. Comme son nom l'indique SUSAN cherche de petites zones USAN, zones qui contiennent des éléments semblables au pixel en cours de traitement. L'algorithme appliqué peut être réparti en quatre étapes.

- **Etape 1 : Zone USAN**

La première étape de détection est la détermination de la zone USAN (Univalue Segment Assimilating Nucleus) zone qui se situe à l'intérieure du masque circulaire et dans laquelle nous cherchons les pixels qui ont un niveau de gris proche de celui du noyau du masque (pixel en cours de traitement) (Figure 1.17.a) Plusieurs situations se présentent (Figure 1.17.b).

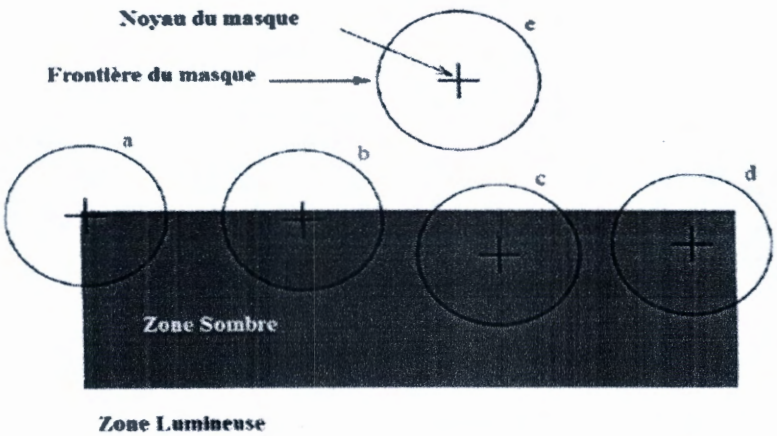


Figure 1.17. a. Quatre masques circulaires à différents endroits sur une image simple

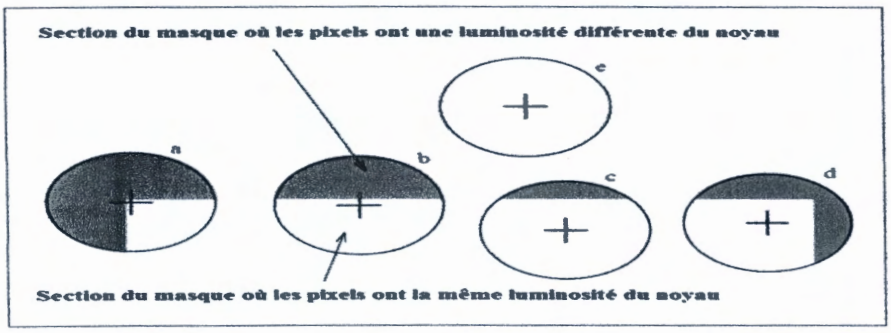


Figure 1.17. b. Quatre masques circulaires avec coloration de la zone de similarité. Les zones USAN sont représentées par les parties blanches des masques.

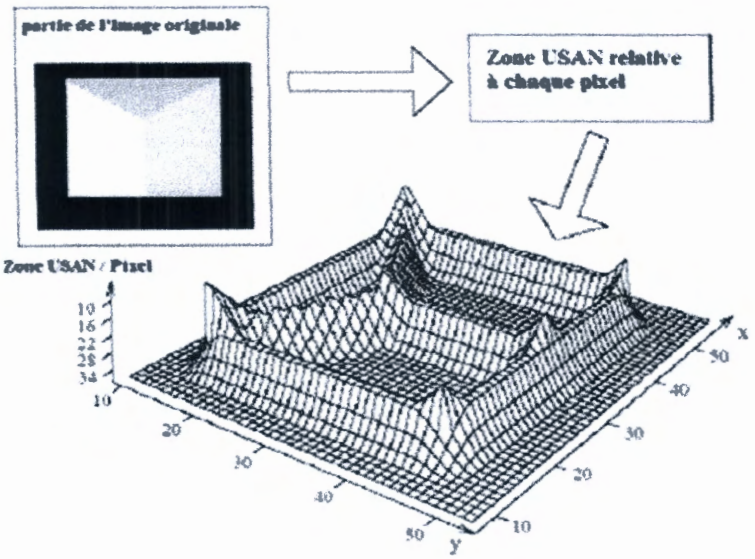


Figure 1.17.c. Représentation tridimensionnelle de la zone USAN.

Figure 1.17. Différentes étapes du détecteur SUSAN.

La zone USAN de chaque pixel est exprimée par le nombre de pixels qui lui sont semblables à une erreur t près (équation 1.10). Comme nous pouvons voir dans la (Figure1.17.b) la zone USAN contient un coin lorsqu'elle a relativement une petite valeur par rapport à la taille du masque, elle est maximale dans une région homogène de l'image et contient un point de contour si elle est supérieure à la moitié de la taille du masque. (Figure1.17.c) montre que les minima locaux atteints par la zone USAN représentent des coins dans l'image. L'équation (1.10) montre le calcul de la zone USAN.

$$n(\vec{r_0}) = \sum_{\vec{r}} c(\vec{r}, \vec{r_0}) \qquad (1.10)$$

Telle que :

$$c(\overrightarrow{r},\overrightarrow{r_0}) = \begin{cases} 1 \; si \; |I(\overrightarrow{r}) - I(\overrightarrow{r_0})| \leq t \\ 0 \; si \; |I(\overrightarrow{r}) - I(\overrightarrow{r_0})| > t \end{cases} \qquad (1.11)$$

Avec :

- I est la fonction d'intensité
- t est la marge de similarité entre r et r_0
- $r_0 = (x_0, y_0)$ position du noyau et r = (x,y) position du pixel comparé au noyau à l'intérieur du masque.

Pour améliorer les résultats, une version lissée de l'équation x.10 est utilisée soit :

$$c(\overrightarrow{r},\overrightarrow{r_0}) = e^{\frac{-I(\overrightarrow{r})-I(\overrightarrow{r_0})^6}{t}} \qquad (1.12)$$

Dans cette équation la valeur de la puissance de l'exponentielle est empirique.

- **Etape 2 : Réponse R**

Par construction, la réponse R du détecteur SUSAN est inversement proportionnelle à la taille de la zone USAN.

$$R(\overrightarrow{r_0}) = \begin{cases} g - (\overrightarrow{r_0}) sin(\overrightarrow{r_0}) < g \\ sinon \\ 0 \end{cases} \qquad (1.13)$$

Où g est le seuil géométrique, il vaut $n_{max}/2$ sachant que n_{max} est la valeur maximale que peut prendre n (équations 1.10).
Pour une valeur plus grande de g, le détecteur devient un détecteur de contours.

- **Etape 3 : Suppression des faux coins**

Ce sont des points où la valeur de R est positive, mais qui ne sont pas réellement de vrais coins. Pour les éliminer, les auteurs calculent le barycentre de la zone USAN et ne retiennent que les points où le barycentre est éloigné du noyau.

- **Etape 4 : Suppression du non-maxima local**

Seuls les points qui représentent des maxima locaux sont retenus définitivement autant que coins. Du fait qu'il ne fait pas recours à la dérivation, le détecteur SUSAN présente le grand avantage d'être robuste aux bruits.

1.3.4 Le détecteur SIFT

Le détecteur SIFT (Scale Invariant FeatureTransform) [13], développé par [Lowe, 2004], est très probablement le plus utilisé. Il présente l'avantage majeur d'être à la fois invariant aux rotations et aux changements d'échelle. De plus, la densité de points détectés est élevée. La particularité de la méthode réside dans le calcul combiné des points d'intérêt et de leurs descripteurs associés. Un descripteur est un vecteur caractérisant le voisinage local en un point d'intérêt. Il caractérise le point d'intérêt de par son unicité.

D'autres méthodes ont été développées pour faire évoluer SIFT. [Morel et Al, 2009] s'intéressent aux déformations de l'image, modélisables localement par une fonction affine en considérant la scène localement plane (Figure 1.19). [Ke et Sukthankar, 2004] présentent l'évolution PCA-SIFT (Principal Components Analysis SIFT) où la rapidité d'exécution a été optimisée.

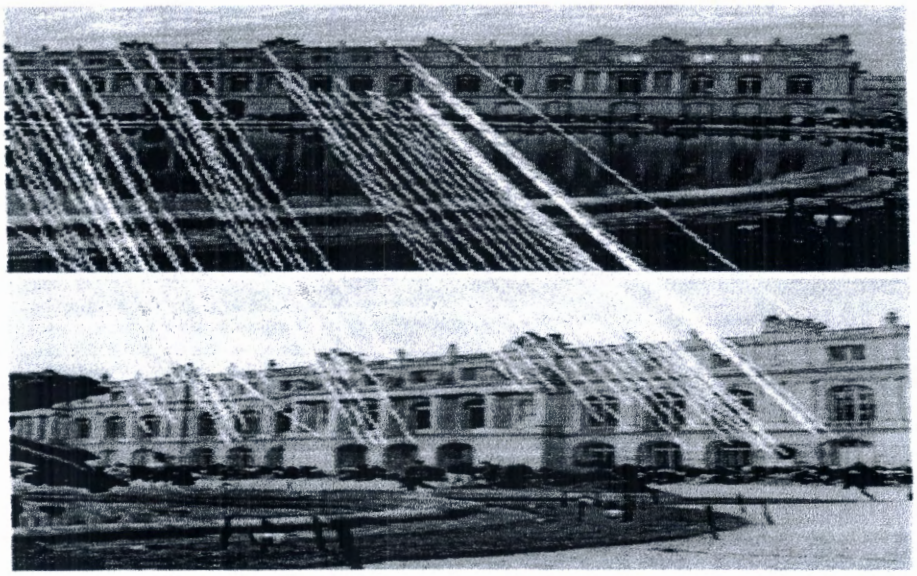


Figure 1.18 : Application du détecteur A-Sift sur deux images prises de deux points de vue très différents.

a. Description de la méthode SIFT (Scale Invariant FeatureMethod)

La méthode SIFT a pour but la détection de points d'intérêts et des descripteurs associés, mais aussi l'appariement de ces points pour former des points homologues (Figure 1.19).

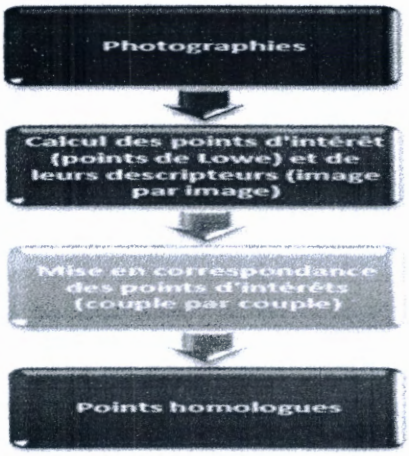


Figure 1.19 : Description de la méthode SIFT.

- **Etape1 : Détection des extrémas dans l'espace des échelles**

Il s'agit de générer un espace des échelles suivi de **DoG**(Difference of Gaussians) pour détecter les points de SIFT. L'image de départ est tout d'abord convoluée à différentes gaussiennes dont l'écart-type varie d'un facteur σ à 2σ, ce qui fournit une série d'images appelée *octave* (Figure 1.20). Ce procédé est réitéré avec l'image originale de l'octave précédente mais sous-échantillonnée de moitié et ayant un écart type de 2σ. Dans un second temps, pour chaque octave, les différences entre les images gaussiennes (DoG) sont calculées. Un point SIFT est un extrema du DoG mais aussi des DoG des niveaux au-dessus et au-dessous (Figure 1.21). Tous les points SIFT détectés subissent ensuite quelques filtrages visant à éliminer les points les moins contrastés ainsi que les points situés sur une arrête.

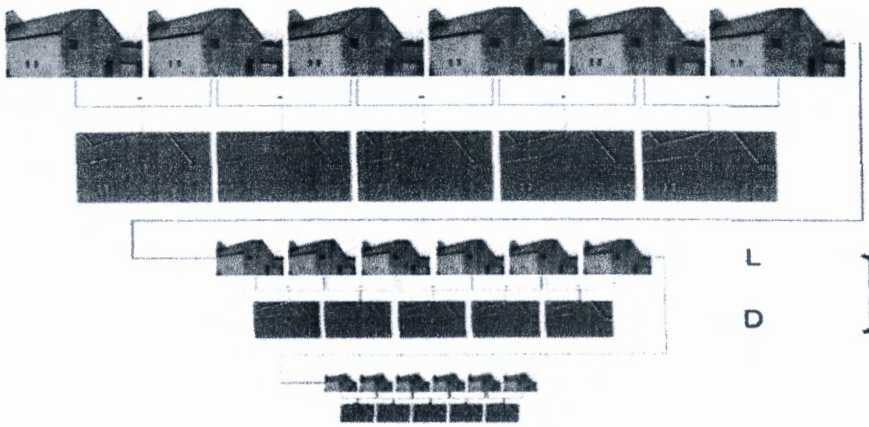


Figure 1.20 :Construction de l'espace des échelles.

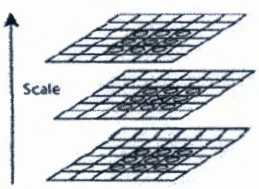


Figure 1.21 : Détection des extrema modifié.

- **Etape2: Génération des descripteurs SIFT**

Le descripteur peut être défini comme la caractérisation du comportement de l'image au voisinage des points d'intérêt. Dans un premier temps, les orientations des gradients des pixels voisins du point SIFT détecté sont calculées. Ces orientations sont rangées dans un histogramme comme l'illustre la (Figure 1.22).

L'orientation du point détecté correspond au plus haut pic de l'histogramme.

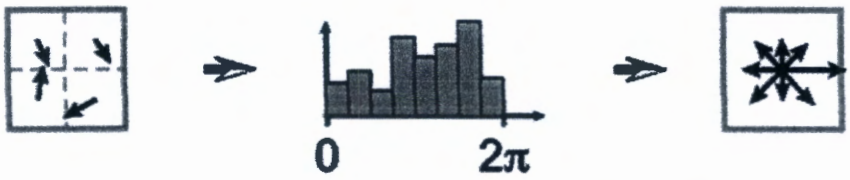


Figure 1.22 : Création de l'histogramme des gradients .

Dans un second temps, sont calculés les histogrammes des orientations des gradients des pixels voisins des points d'intérêt au sein de fenêtres situées autour du point d'intérêt détecté. Les valeurs des histogrammes sont représentées sous la forme d'un vecteur pour chaque fenêtre. Le descripteur SIFT est en fait constitué de la concaténation de tous ces vecteurs en un seul vecteur. Ce descripteur est ensuite normalisé.

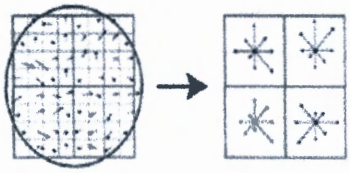


Figure 1.23 : A gauche : les gradients de l'image, à droite : le descripteur SIFT résultant .

- **Etape3 : L'appariement des points SIFT**

L'appariement des points SIFT repose sur le calcul des distances euclidiennes dans l'espace des descripteurs (espace de dimension 128). La méthode se base sur la construction d'un *k-d tree* et sur l'emploi l'algorithme *Best Bin First*.

La méthode SIFT permet donc de détecter et d'apparier des points qui sont peu sensibles aux rotations, aux variations d'échelle, au bruit et aux transformations affines dans une certaines limite. En revanche, la méthode demeure sensible au diachronisme (dates de prises de vue différentes). Le nombre de points détecté grâce à cette méthode est très élevé et couvre l'image de façon dense. L'une des faiblesses de la méthode est la rapidité des calculs.

b. Applications des descripteurs SIFT

La reconnaissance d'objet, les avantageuses propriétés des descripteurs SIFT caractère discriminant, invariance à la translation, à la rotation et au changement d'échelle et robustesse aux transformations affines en général (distorsions), aux changements de points de vue 3D ainsi qu'aux changements de luminosité) en font un excellent choix pour d'autres applications dont quelques-unes sont énumérées ici.

- Localisation de robot en environnement inconnu
- L'assemblage de panorama
- Modélisation, reconnaissance et suivi d'objets 3D
- Descripteurs SIFT 3D pour la reconnaissance de mouvements
- Analyse du cerveau humain au moyen d'images RMN 3D

c. Variantes et extensions

- **RIFT :** est une variante de SIFT adaptée aux images texturées sur lesquelles la notion d'orientation principale n'a pas vraiment de sens, tout en étant invariant aux rotations. Le descripteur est construit en utilisant des patches circulaires normalisés divisés en anneaux également circulaires et d'égale largeur ; un histogramme d'orientation de gradient est calculé à partir de chacun de ces anneaux. Pour garantir l'invariance à la rotation, l'orientation est mesurée pour chaque point du centre vers l'extérieur.

- **G-RIF** (ou *Generalizedrobust invariant feature*) est un descripteur de contexte regroupant les orientations des contours, les densités des contours et les informations de teintes dans une forme unifiée, combinant information perceptive et codage spatial. Une nouvelle méthode de classification est utilisée, basée sur un modèle probabiliste utilisant les votes des descripteurs locaux dans un voisinage.

1.3.5 Les descripteurs SURF [15]

La construction du descripteur SURF (SpeededUp RobustFeatures) est initialement basée sur le détecteur Fast-Hessien pour détecter les points d'intérêt. Ensuite, on calcule les réponses d'ondelettes de Haar dans les directions horizontales dx et verticale dy dans un cercle de rayon R centrée au point d'intérêt. Le problème d'invariance en échelle est géré par le choix de la valeur de R. La distribution des réponses d'ondelettes de Haar sous la forme (Pdx,Pdy,P|dx| ,P|dy|) est estimée dans les 16 quadrants 4x4 autour du point considéré comme le SIFT. La signature locale est de dimension 4x4x4= 64 ce qui permet de réduire le temps de

calcul des caractéristiques et de recherche des correspondances. Le descripteur SURF est un détecteur de points d'intérêts invariant aux changements d'échelle et aux rotations.

Afin d'obtenir une invariance à la rotation et à l'échelle, l'algorithme reprend les mêmes techniques que SIFT. Après la détermination de la valeur d'échelle et de l'orientation principale du PI, une région d'intérêt est découpée en bloc de 4x4. Dans chaque bloc des descripteurs simples sont calculés formant un vecteur v défini par :

1.3.6 Détecteur FAST: Features from Accelerated Segment Test (2006)

Là encore, les auteurs se basent sur la luminance au voisinage des pixels. Le voisinage de chaque pixel est le cercle de Bresenham. Les auteurs choisissent un voisinage de 16 pixels soit un cercle de rayon r = 3 px (figure 1.24). Si sur le cercle existe n pixels contigus tous supérieurs à Ic + t (ou inférieurs à Ic- t) alors le noyau C est déclaré comme coin, tel que t représente le seuil. La valeur considérée de n est 12 (figure 1.25).

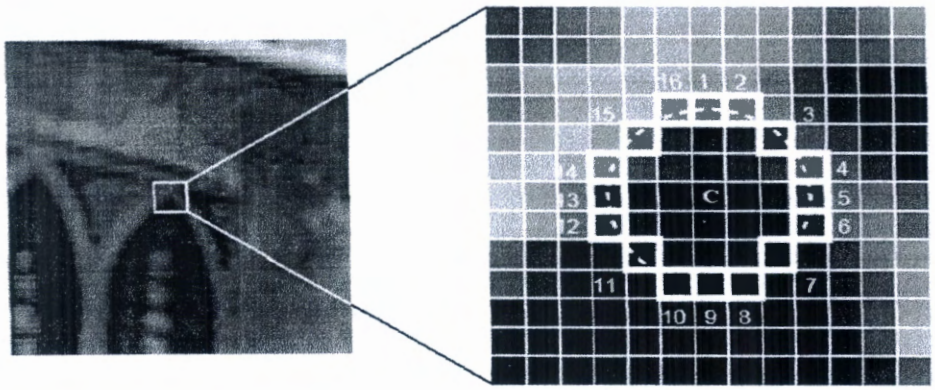


Figure 1.24 : Considération du voisinage de Bresenham du point C de 16 pixels (r = 3 px).

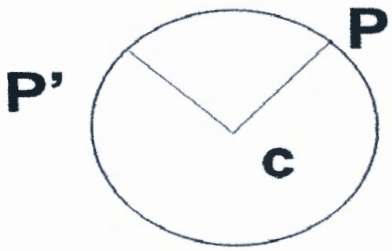


Figure 1.25 : Comparaison de n pixels contigus au centre (ici n=12).

Le détecteur proposé est rapide et possède une répétabilité meilleure que SUSAN, Harris autres. Les auteurs ont prouvé empiriquement que la meilleure valeur de n est 9 et pas 12.

1.4 Détection de régions d'intérêt

1.4.1 La méthode MSER : (Maximally Stable Extremal Region)2002.

Les MSER sont des régions de l'image qui sont des composantes connexes « stables » de l'image seuillée. Plus exactement, ces régions sont déterminées en appliquant un seuillage à l'image (c'est-`a-dire que tous les pixels ayant une intensité inférieure à un seuil donné deviennent noirs et les autres deviennent blancs) avec une valeur pour le seuil qui part de zéro et augmente continument. Durant le processus, on étudie l'évolution de la taille de chaque région « noire ». Lorsque le grossissement de cette taille passe par un minimum local (en fonction de l'augmentation du seuil), la composante connexe est dite « au plus stable ». On a alors obtenu une MSER (le mot « extremal » évoque la propriété selon laquelle les pixels de la MSER sont tous plus foncés — ou tous plus clairs — que les pixels situés aux frontiéres de la région).

Figure 1.26 : détection des régions d'intérêt.

➢ **Principe**

MSER est basé sur l'idée de prendre des régions qui restent à peu près les mêmes dans un large éventail de seuils.

- Les pixels en dessous d'un seuil donné sont blancs et tous ceux ci-dessus ou égaux sont noirs.

- Si on nous montre une séquence d'images de seuil. Avec un cadre correspondant au seuil t, on verra d'abord une image noire, puis des volumes blancs correspondant aux minimums d'intensité locaux apparaîtront alors plus grands.

- Ces blancs se fusionneront finalement, jusqu'à ce que l'image entière soit blanche.

- L'établissement de tous les composants connectés dans la séquence est l'ensemble de toutes les régions extrêmes.

Le mot extremal se réfère à la propriété que tous les pixels à l'intérieur du MSER ont soit une intensité supérieure (régions extrêmes brillantes) soit une intensité inférieure (régions extrêmes sombres) que tous les pixels sur sa limite extérieure.

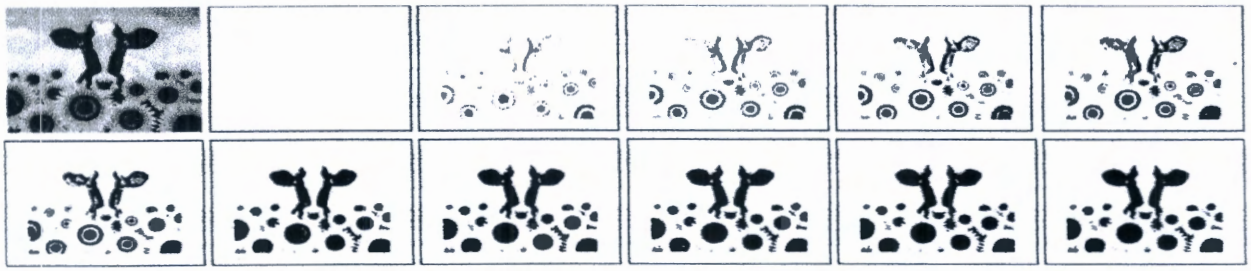


Figure1.27 : Appliquer une série de seuils.

Les régions extrêmes ont les propriétés importantes, que l'ensemble est fermé sous une transformation continue (et donc projective) des coordonnées de l'image, c'est-à-dire qu'il est invariant affine et peu importe si l'image est déformée ou incliné.

Étant donné que les régions sont définies exclusivement par la fonction d'intensité dans la région et la bordure extérieure, cela conduit à de nombreuses caractéristiques clés des régions qui les rendent utiles. Sur une large gamme de seuils, la binarisation locale est stable dans certaines régions et possède les propriétés suivantes:

- Invariance à la transformation affine des intensités d'image
- Positionnement: seules les régions dont le support est presque identique sur une gamme de seuils sont sélectionnées.
- La détection de grande échelle, bien que la structure fine et grande est détectée.
- L'approche est plutôt sensible aux effets d'éclairage naturel en tant que changement de lumière de jour ou d'ombres en mouvement.

➢ **Propriétés de MSER**

1. Fonctionne bien sur des images contenant des régions homogènes avec des limites distinctives ainsi pour les petites régions.
2. Ne fonctionne pas bien avec les images avec un flou de mouvement
3. Multipropriété: la plus grande gamme de produits chimiques offre une meilleure stabilisation des prix et des images animées et une amélioration de la performance de la capture en cours d'utilisation et des images animées.
4. Bonne répétabilité
5. Affinement invariant.

6. • Une implémentation intelligente en fait l'un des détecteurs de région les plus rapides.

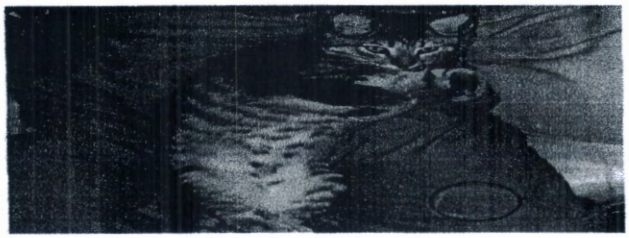


Figure 1.28 : Résultat de la détection.

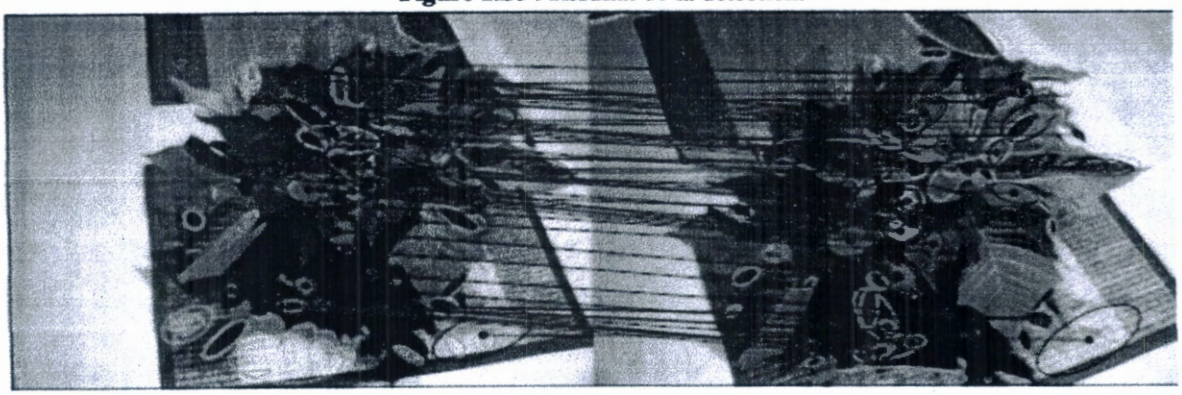


Figure 1.29 : Mise en correspondance entre régions d'intérêt.

➢ **MSER contre d'autres détecteurs de région**

- Changement de point de vue: MSER effectue le meilleur.
- Changement d'échelle: Hessian-Affine, MSER, Harris-Affine sont les meilleurs
- Le bruit : tous les détecteurs sont invariants au flou de l'image. MSER n'est pas invariable avec un flou de mouvement
- Changement JPEG: MSER est le meilleur.

1.5 Conclusion

Nous avons présenté quelques approches permettant d'extraire les caractéristiques d'une image. Ces méthodes sont basées sur la détection des points d'intérêt. Certaines méthodes sont simples à mettre en œuvre et donnent des résultats assez corrects (Harris et SIFT). D'autres méthodes comme Moaravec, SURF sont moins performant. Il existe néanmoins plusieurs autres méthodes non abordées ici qui offrent des perspectives très intéressantes.

2.1 Introduction

Les points d'intérêt ont été produits dans le but de réduire le maximum d'information d'une image en quelques points. Quelques applications existent, le plus souvent dans un objectif de reconnaissance de formes ou d'appariement d'images.

2.2 Idée du détecteur Harris

Harris et Stephen ont identifié certaines limitations du détecteur de Moravec qui fonctionne dans un contexte limité et souffre en effet de nombreuses limitations, en les corrigeant, et déduisez un détecteur de coins très populaire.

2.3 Présentation

Un algorithme utilisé pour la détection est le Détecteur Combiné de Coins et de Contours de Harris et Stephens (1988) [17]. Cet algorithme développé pour le suivi (tracking) d'objets en 3D est utilisé pour résoudre le problème de détection de coins. L'idée du détecteur est de considérer le voisinage d'un pixel (une fenêtre) et de déterminer les changements moyens de l'intensité dans le voisinage considéré lorsque la fenêtre se déplace dans diverses directions.

On considère 3 cas :

- Si la région considérée a une intensité approximativement constante, tout déplacement produira un petit changement.
- Si la région considérée passe un contour, un déplacement le long du contour produira un petit changement mais un déplacement perpendiculairement au contour produira un grand changement.
- Si la région considérée passe un coin, un déplacement dans n'importe quel sens produira un fort changement.

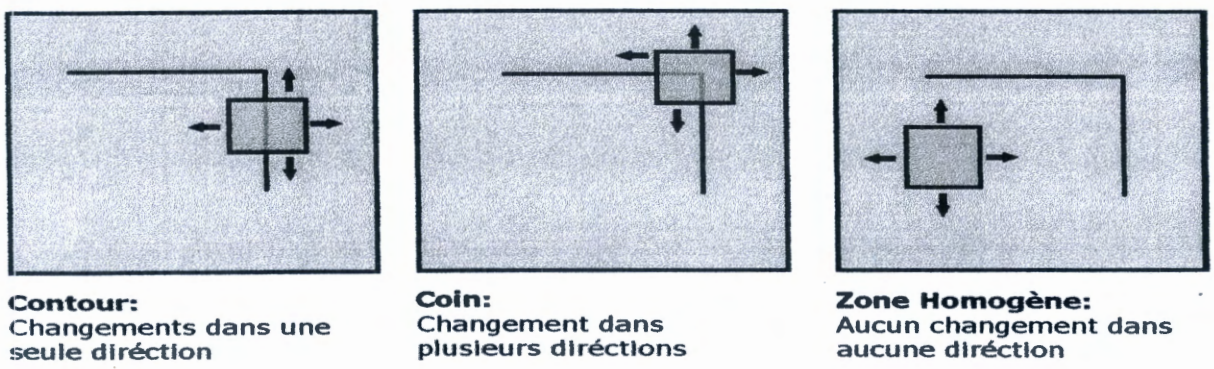


Figure 2.1 : Les trois cas de changements d'intensité considérés.

Définissons une image par sa fonction intensité I (x,y) à la valeurs dans R . On considère ainsi que des images en niveaux de gris.

Si on traduit ça par une expression mathématique on peut définir à partir de l'intensité I des images la fonction de changement produit par un déplacement E :

$$E(x,y) = \sum_{u,v} w(u,v)\,[I(x+u,y+v) - I(u,v)]^2 \qquad (2.1)$$

On note Ix et Iy les dérivées respectivement en x et y.

Les nombreuses variantes utilisent chacune différents calculs pour arriver a une fonction dont les points a extraire correspondent a des maxima locaux .

On considère le développement de Taylor de la fonction d'intensité I au voisinage du pixel (u, v) :

$$I(x+u,y+v) = I(u,v) + x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta y} + o(x^2+y^2) \qquad (2.2)$$

D'où :

$$E(x,y) = \sum_{u,v} w(u,v)\,[x\frac{\delta I}{\delta x} + y\frac{\delta I}{\delta y} + o(x^2+y^2)]^2 \qquad (2.3)$$

Pour les petits déplacements on peut négliger le terme o(x2, y2) d'où :

$$E(x,y) = Ax2 + 2Cxy + By2 \qquad (2.4)$$

Avec :

- $A = \frac{\delta I^2}{x} \otimes w$
- $B = \frac{\delta I^2}{y} \otimes w$
- $C = \frac{\delta I}{\delta x}\frac{\delta I}{\delta y} \otimes w$

Afin de réduire le bruit dans la fenêtre utilisée Harris propose l'utilisation d'une fenêtre gaussienne :

$$w(u,v) = exp - (u^2+v^2)/\sigma^2 \qquad (2.5)$$

Enfin, pour prendre en compte le comportement général de la fonction E localement, on écrit:

$$E(x,y) = (x,y) * M(x,y)^t \qquad (2.6)$$

Avec

$$M = \begin{bmatrix} A & \cdots & C \\ \vdots & \ddots & \vdots \\ C & \cdots & B \end{bmatrix} \qquad (2.7)$$

M : symétrique, définie positive ⇒ décomposition en valeurs propres.

La matrice M caractérise le comportement local de la fonction E, les valeurs propres (λ_1 et λ_2) de cette matrice correspondent en effet aux courbures principales associées à E. En revenant aux 3 cas :

- Si R possède une petite valeur, alors la surface en question représente une zone homogène.
- Si R < 0 alors la surface contient un contour.
- Si R > 0 alors la surface contient un coin.

Harris propose donc une métrique pour détecter au même temps les coins et les contours :

$$R = Det(M) - k.tr(M)^2 = (\lambda_1 . \lambda_2) - (\lambda_1 + \lambda_2)^2 \qquad (2.8)$$

Où k est une constante qu'on peut régler. Normalement k = 0.04, empiriquement
Les valeurs de R sont positives au voisinage d'un coin, négatives au voisinage d'un contour et faibles dans une région d'intensité constante. Les points qui vont nous intéresser sont ceux pour lesquels R > 0. Lors de la détection par grille, on garde uniquement le point qui maximise cette réponse dans chaque case.

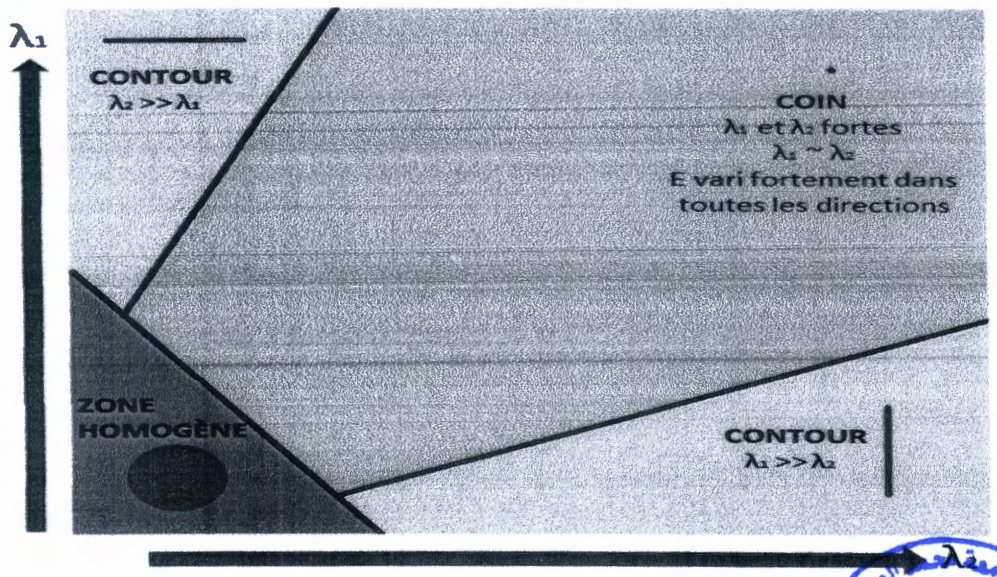


Figure 2.2 : Classification des valeurs propres.

جامعة ... مكتبة الكلية 02

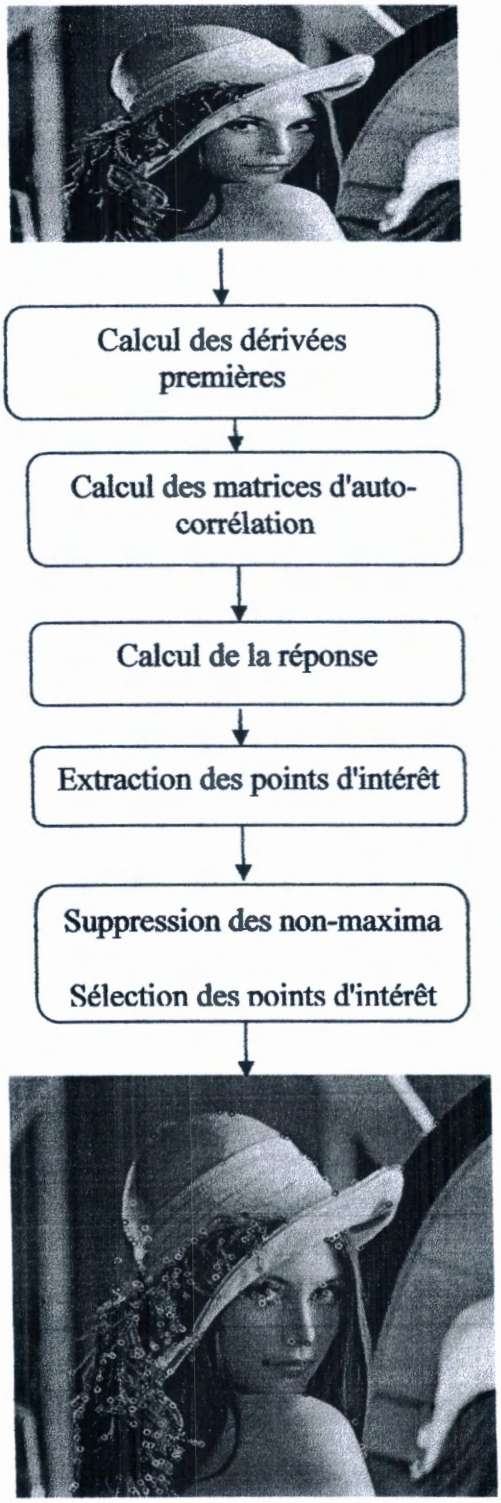


Figure 2.3 : Détecteur de Harris - Afin de retrouver les points d'intérêts, le détecteur de Harris calcule pour chaque pixel, la matrice d'auto-corrélation à partir des deux composantes des vecteurs gradients de l'image. Ensuite, la matrice de réponse du détecteur est obtenue à partir de ces matrices. Enfin, les points d'intérêts, ici marqués d'une croix, sont localisés à partir de cette réponse.

2.3.1 Propriété [18]

- **Invariant à la rotation** Même après rotation la forme reste la même, et les valeurs propre de la matrice aussi

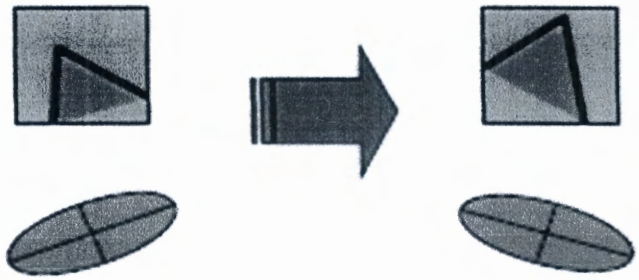


Figure 2.4 : Invariant à la rotation.

- **Détecteur de Harris n'est pas invariant au changement d'échelle**

Figure 2.5 : Détecteur Harris n'est pas invariant au changement d'échelle.

- **Détecteur de Harris donne des résultats précis des coins**

- Les points extraits avec cette méthode correspondent bien à la définition de point saillant.
- Les points extraits sont bien situés sur les coins et pas sur les lignes [17]. Lors de la corrélation on veut éviter les points situés sur les lignes par la nature de l'algorithme de corrélation.
- L'image est localement très similaire sur le long d'une ligne. La Figure 2.6 montre les points extraits sur une région d'une image aérienne d'Amiens.
- Les points sont bien situés sur les coins mais par contre, on trouve beaucoup de points sur les bâtiments (ce qui était attendu). Ces points présentent des déformations par effet stéréo surtout dans le cas des images aériennes.

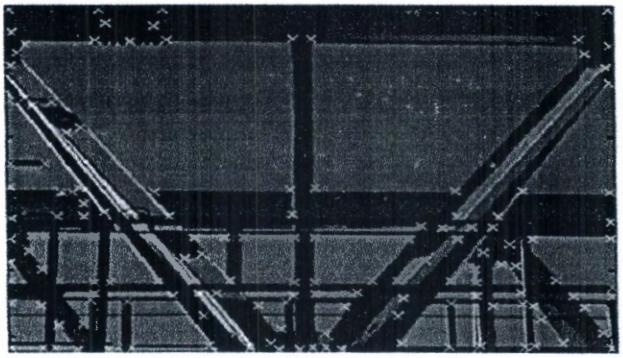


Figure 2.6 : Détection des coins par le détecteur de Harris.

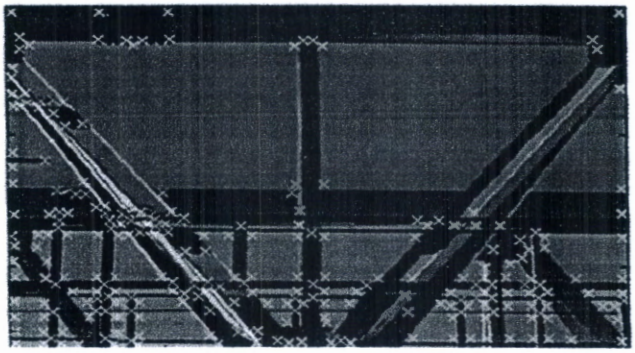


Figure 2.7 :Détection des coins par le détecteur Moravec .

2.4 Limites du détecteur

Le détecteur de Harris-Stephens possède une bonne répétabilité et une invariance à la rotation. Cependant il souffre de quelques limites:

- Sensibilité au bruit à cause de la dérivation.
- Coût élevé en temps de calcul à cause de la convolution avec une gaussienne.
- Sensibilité au changement d'échelle.
- Sensibilité aux transformations affines

Par ailleurs, la valeur de k utilisée par Harris-Stephens est empirique, elle n'a pas été prouvée analytiquement.

Exemple : Harris-Laplacien

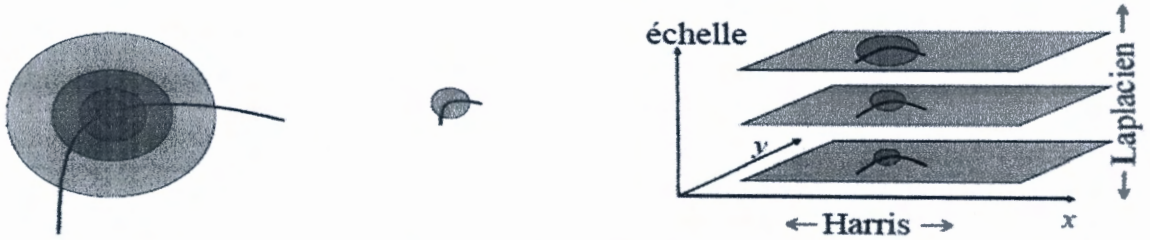


Figure 2.8 : Harris-Laplacien.

2.5 Mise en correspondance

Les méthodes de corrélation sont utilisées depuis longtemps pour mettre en correspondance des pixels sur la base d'informations d'intensités. L'idée est de définir une mesure de similarité entre les pixels de deux images.

Les pixels sont les primitives les mieux adaptés pour la mise en correspondance [17].

Les régions sont en effet mal adaptées à la mise en correspondance (la taille d'une région est différente d'une image à une autre), tout comme les contours.

Considérons deux images 1 et 2 d'une scène, il s'agit ici de déterminer, pour un élément de l'image 1, l'élément qui lui correspond dans l'image 2 et éventuellement dans d'autres images. La mise en correspondance de primitives est un problème fondamental de la vision par ordinateur. C'est un processus intermédiaire entre les processus dit de hauts niveaux : reconstruction, reconnaissance, etc., et ceux de bas niveaux : extraction d'indices.

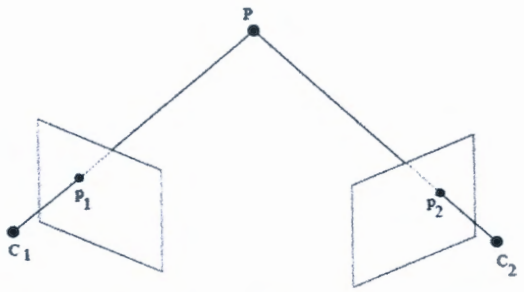


Figure 2.9 : Deux pixels p1 et p2 correspondants.

Il existe 3 types de primitives comme on a montré dans le chapitre précédent : points, segments ou contours, régions et repose sur l'introduction de mesure de similarité (ressemblance) entre primitives dans plusieurs images. La mise en correspondance est, en général, ambiguë. Des contraintes géométriques sur la position d'un correspondant dans l'image 2 sont utilisées pour réduire le nombre de correspondants potentiels.

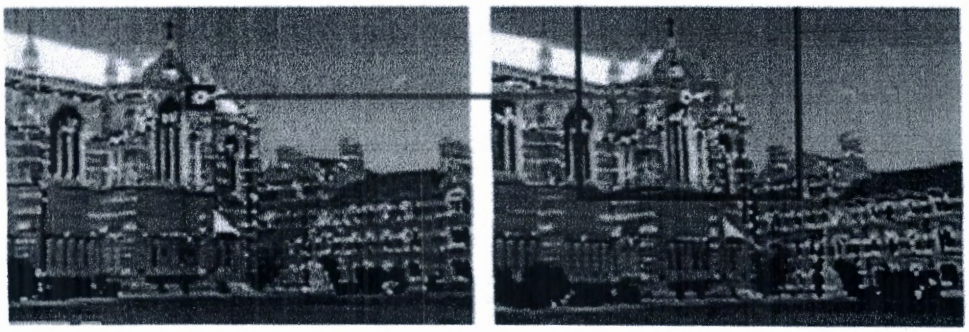


Figure 2.10 : mis en correspondance entre images.

Le principe est de considérer, pour un pixel p1 de l'image 1, une fenêtre rectangulaire centrée en p1 et de calculer sa corrélation/distance avec une fenêtre dans la deuxième image. La fonction de corrélation est alors maximum en p2 correspondant de p1 dans la deuxième image (distance minimum).

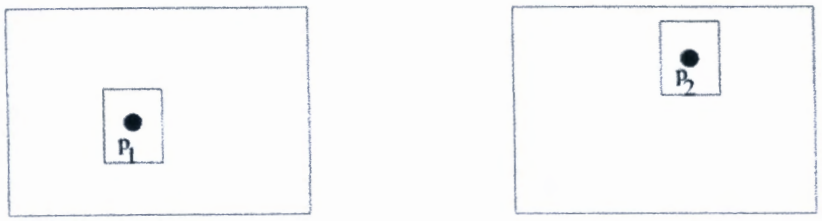


Figure 2.11 : Mise en correspondance.

a. Mise en correspondance - fonctions de dissimilarité

Pour une fenêtre de taille 2N + 1 × 2P + 1

- **Sum of absolute differences (SAD)**

$$SAD\ (p1(u1, v1), p2(u2, v2)) = \sum_{i=-N}^{i=N} \sum_{j=-P}^{j=p} |I1(u1 + i, v1 + j) - I2(u2 + i, v2 + j)| \quad (2.9)$$

- **Sum of squared differences (SSD)**

$$SSD\ (p1(u1, v1), p2(u2, v2)) = \sum_{i=-N}^{i=N} \sum_{j=-P}^{j=p} (I1(u1 + i, v1 + j) - I2(u2 + i, v2 + j))^2 \quad (2.10)$$

Pour une fenêtre de taille 2N + 1 × 2P + 1

- **Zero Mean Sum of squared differences (ZSSD)**

$$ZSSD\ (p1(u1, v1), p2(u2, v2)) =$$

$$\sum_{i=-N}^{i=N} \sum_{j=-P}^{j=p} [(I1(u1 + i, v1 + j) - I1(u1, v1)) - \overline{(I2(u2 + i, v2 + j) - I1(u2, v2))}]^2 \quad (2.11)$$

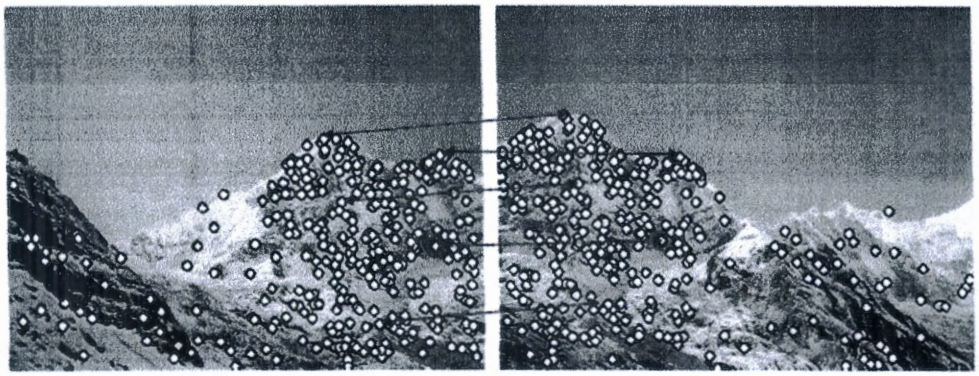


Figure 2.12. Mise en correspondance.

2.6 Conclusion

Le détecteur de Harris s'est avéré être une bonne solution pour la détection automatique des points d'intérêt. Il permet de filtrer les points sur les lignes et de détecter des points situés sur des coins pour faciliter la mise en correspondance et pour robustifier la transformation globale de l'image. C'est ce détecteur que nous allons implémenter sur un Smartphone.

3.1 Introduction

À l'origine, « Android » était le nom d'une PME américaine, Android Incorporated, créée en 2003 puis rachetée par Google en 2005, qui avait la ferme intention de s'introduire sur le marché des produits mobiles. L'objectif d'Android était de développer un système d'exploitation mobile plus intelligent, qui ne se contenterait pas uniquement de permettre d'envoyer des SMS et transmettre des appels. Ce système entravait la possibilité de développer facilement des applications qui s'adapteraient à tous les téléphones, surtout entre constructeurs.

En janvier 2007, Apple dévoilait l'iPhone, un téléphone tout simplement révolutionnaire pour l'époque, capable d'aller sur internet, de lire des vidéos, etc. Le problème pour les autres constructeurs étant que pour atteindre le niveau d'iOS (iPhone OS, le système d'exploitation pour iPhone), il aurait fallu des années de recherche et développement à chaque constructeur.

C'est pourquoi est créée en novembre de l'année 2007 l'Open Handset Allianceune organisation qui comptait à sa création 35 entreprises évoluant dans l'univers du mobile, dont Google. L'OHA compte à l'heure actuelle 80 membres, elle cherche à développer des standards open source pour les appareils mobiles.

Depuis sa création, la popularité d'Android a toujours été croissante. C'est au quatrième trimestre 2010 qu'Android devient le système d'exploitation mobile le plus utilisé au monde, on le retrouve non seulement dans les tablettes et Smartphones, mais aussi dans les téléviseurs, les consoles de jeux, les appareils photos, etc.

3.2 Outils de développement d'une application Android

Android utilise le langage Java pour permettre à des développeurs de créer des applications. Avant de commencer le développement, il faut installer tous les programmes et les composants nécessaires. A savoir un IDE (Eclipse (java), Netbeans, Intellij, Android Studio), le JDK (Java Developpement Kit), le SDK(Software developpement Kit) et et un émulateur AVD(Android Virtual device (non obligatoire). nous allons détailler brièvement l'ensemble de ces outils dans les sections suivantes .

3.3 L'environnement Android Studio

Google a créé un environnement de développement complet pour la création d'application mobile Android nommé Android Studio. Il est open source et disponible gratuitement, permettant de réaliser des projets sur différents types de support, tablette ou Smartphone. Il permet principalement d'éditer les fichiers Java et les fichiers de configuration d'une application Android.

✓ **Le Java Dévelopment Kit**

Il existe deux plateformes en Java :

- Le JRE (Java Runtime Environment), qui contient la JVM (Java Virtual Machine, les bibliothèques de base du langage ainsi que tous les composants nécessaires au lancement d'applications. En gros, c'est l'ensemble d'outils qui permet d'exécuter l'application.

- Le JDK (Java Development Kit), qui contient le JRE (afin d'exécuter les applications Java), mais aussi un ensemble d'outils pour compiler et déboguer le code, transformé en byte code et envoyé à la machine virtuelle Java.

✓ **SDK (Software Development Kit)**

Conçu par Google (et l'Open Handset Alliance), le Kit de développement logiciel Android où SDK Android est composé de plusieurs éléments pour aider les développeurs à créer, compiler, déployer et maintenir des applications mobiles :

— des API (interfaces de programmation) ;
— des exemples de code ;
— de la documentation ;
— des outils – parmi lesquels un émulateur – permettant de couvrir quasiment toutes les étapes du cycle de développement d'une application. [16]

✓ **AVD (Android Virtual Device)**

L'émulateur ou AVD est un outil de test et de débogage des applications. C'est une implémentation de la machine virtuelle Dalvik et ART, faisant de celle-ci une plate-forme exécutant les applications Android comme le ferait n'importe quel téléphone Android.

Lors de la création de l'AVD, on définit la configuration de l'appareil virtuel. Cela comprend, par exemple, la résolution, la version Android de l'API et la densité de l'écran. On

peut définir plusieurs AVD avec des configurations différentes et les lancer en parallèle. Cela nous permet de tester plusieurs configurations des périphériques en une fois. [16]

3.4 Structure d'un projet Android

Tout comme n'importe quel projet Java, la structure d'un projet Android est organisée sous forme d'une arborescence de répertoire. Cette structure doit être bien respectée. L'arborescence du projet montrée ci-dessous est automatiquement générée lors de sa création.

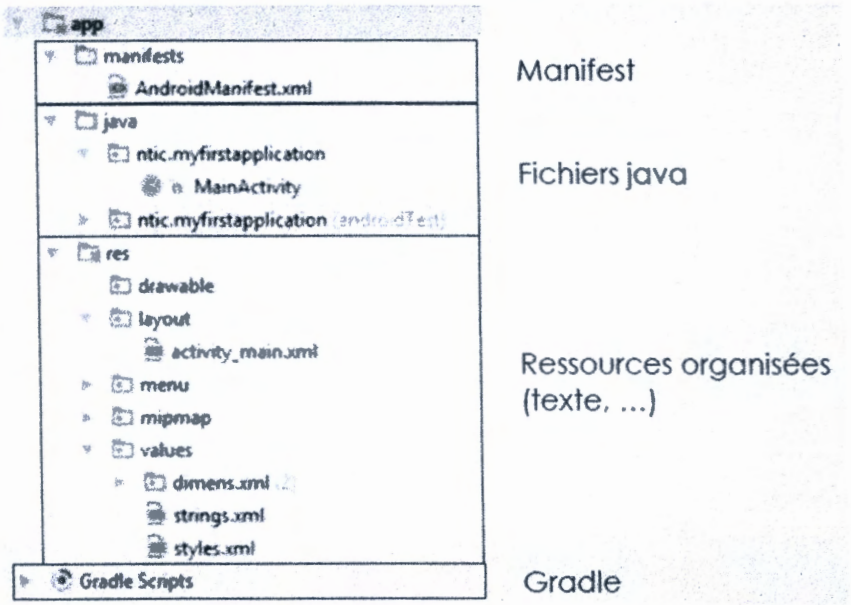


Figure 3.1 : Arborescence d'un projet Android.

Explication des différents répertoires

- **AndroidManifest.xml :** décrit les propriétés de l'application.
- **res/:**
  res/**drawable-Xdpi**: ressources images (height,low,...)
  res/**layout**: description des IHMs en XML des activités.
  res/**menu**: contient la description des menus en XML.
  res/**values**: chaines de caractères et dimensions.

- **src/:**les sources java du projet
- **bin/**:contient les fichiers compilés

  bin/**classes**: les classes compilées en **.class**

  bin/**classes.dex**: exécutable généré par la JVM Dalvikou ART

  bin/**myapp.zip**: les ressources de l'application.

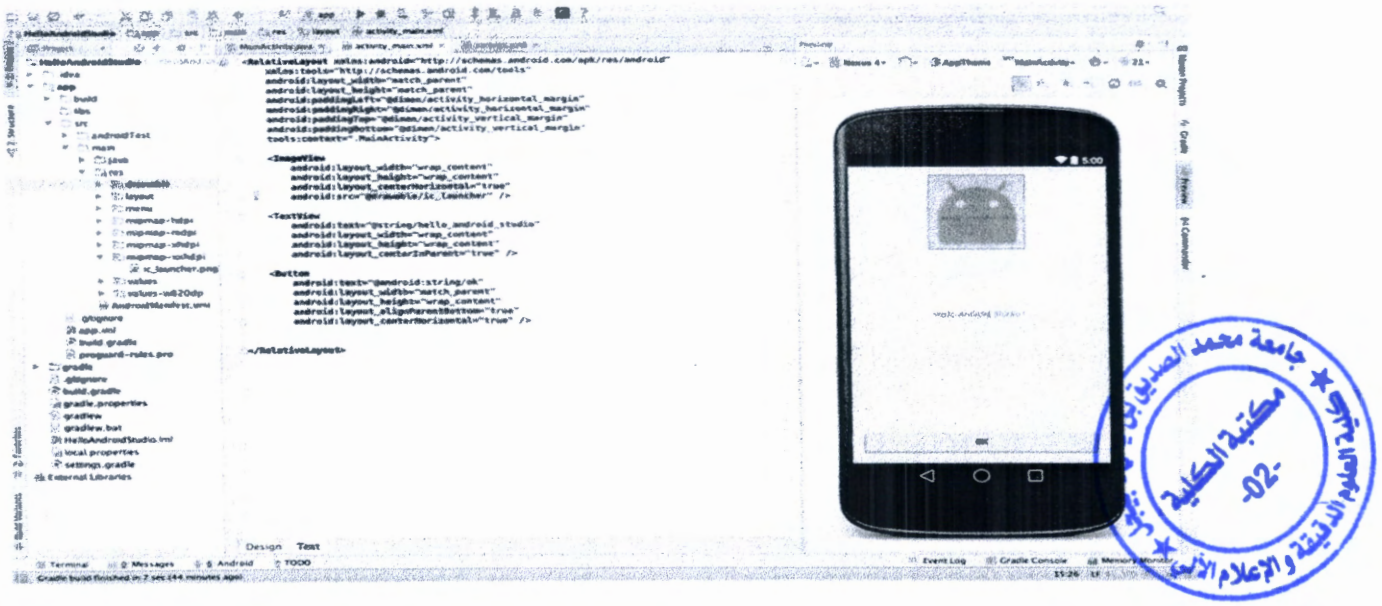
  bin/**myapp.apk**: La véritable application Android qui sera installée sur l'appareil.

3.5 Composantes d'une application Android[23]

Une application Android est composée de plusieurs éléments qu'il faut assembler pour obtenir un tout cohérent. Plus une application est complexe, plus le nombre de pièces utilisées sera grand. Une application Android peut être composée des éléments suivants :

- **Interface graphique :**

Sous Android, une interface peut être décrite soit avec une description déclarative XML ou directement dans le code d'une activité en utilisant le langage Java. La façon la plus simple de réaliser une interface est d'utiliser la méthode déclarative XML via la création d'un fichier XML que nous placerons dans un sous dossier du dossier /res de l'application. [16]



جامعة محمد الصديق بن يحيى
مكتبة الكلية
02
كلية العلوم الدقيقة و الإعلام الآلي

Figure 3.2 : interface graphique de l'environnement d'Android

- **AndroidManifest.xml**

- Décrit les permissions requises par l'application
- Accès a l'Internet, lecture-écriture dans la liste de contacts, Géo-localisation..etc.
- Décrit les besoins matériels et logiciels de l'application appareil photo, Bluetooth, ecranmulti-touch,

- **Les activités :**

Une activité peut être assimilée à un écran qu'une application propose à son utilisateur. Pour chaque écran de notre application, nous devrons donc créer une activité. La transition entre deux écrans correspond au lancement d'une activité ou au retour sur une activité placée en arrière-plan.

Une activité est composée de deux volets :

- la logique de l'activité et la gestion du cycle de vie de l'activité qui sont implémentés en Java dans une classe héritant de la classe *Android.app.Activity* ;
- l'interface utilisateur, qui pourra être définie soit dans le code de l'activité soit de façon plus générale dans un fichier XML placé dans les ressources de l'application. [16]

Le changement d'état d'une activité provoque le déclenchement de la méthode callback correspondante :VoidonCreate(...),VoidonStart(),VoidonRestart(),VoidonResume(),VoidonPause(),VoidonStop() ,VoidonDestroy().

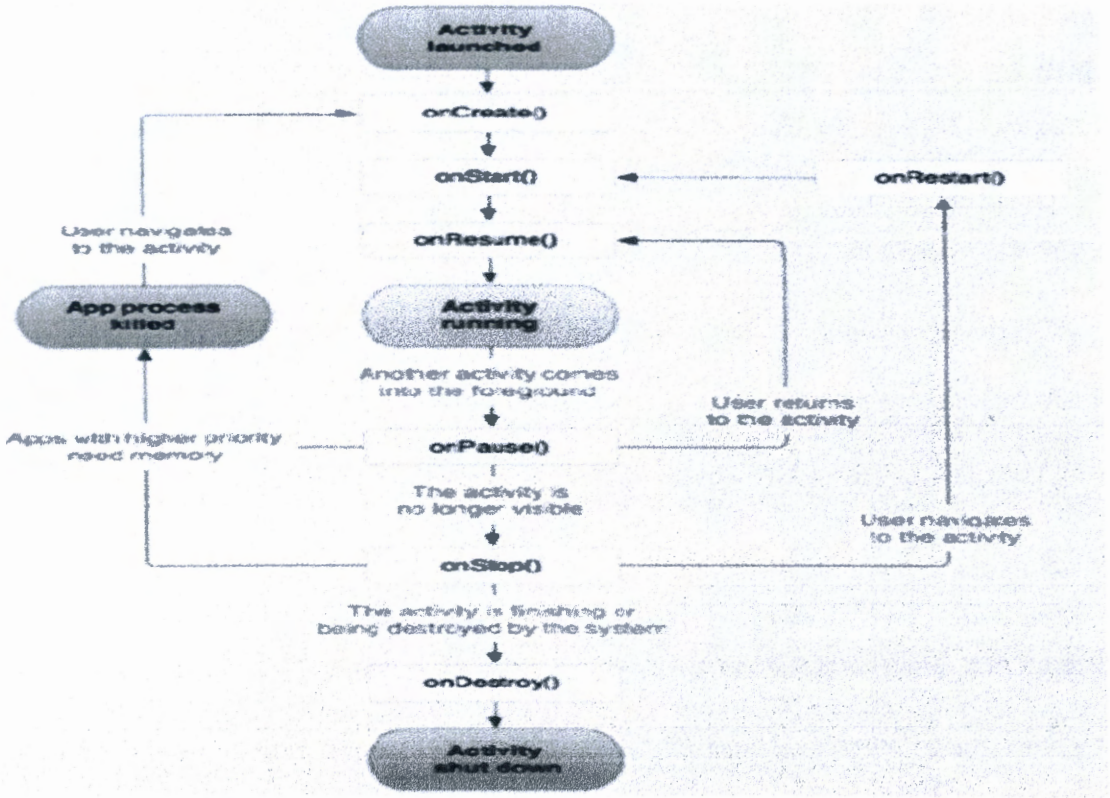


Figure 3.3: Présente cycle de vie d'une application Android

- **Les services :**

Le service est un composant qui fonctionne en tâche de fond, de manière invisible. Ses principales utilisations sont la mise à jour de sources de données ainsi que d'activités visibles et le déclenchement de notifications. Le meilleur exemple est le joueur de musique qui peut fonctionner pendant qu'on passe d'un écran à un autre. Un service est une classe Java qui doit hériter de la classe Android.app.Service. [16]

- **Les récepteurs d'Intents :**

Il permet à une application d'être à l'écoute des autres afin de répondre à certaines tâches, qui lui sont destinées et qui sont envoyées par d'autres composants applicatifs.

Un récepteur d'intent est une classe Java qui doit hériter de la classe Android.content.BroadcastReceiver, et doit redéfinir la méthode "Onreceive()".[16]

- **Les vues :**

Les vues sont les briques de construction de l'interface graphique d'une activité Android. Les objets des vues représentent des composants à l'écran de l'appareil qui permet d'interagir avec l'utilisateur. [16]

- **Les layouts :**

Un gabarit, ou layout dans la documentation officielle, ou encore mise en page. Il s'agit en fait d'un conteneur qui aide à positionner les objets, qu'il s'agisse de vues ou d'autres gabarits au sein d'une interface. On peut imbriquer des gabarits les uns dans les autres, ce qui permettra de créer des mises en forme évoluées. [16]

Example: Button, EditText, TextView, CheckBox

```
<?xml version="1.0" encoding="utf8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="hello world !!!" />
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="coucou" />
</LinearLayout>
```

Figure 3.4 Représentation de layout

- **Les éléments d'interaction : intents et notifications**

Les Intents permettent de gérer l'envoi et la réception de messages afin de faire coopérer les applications. Le but des Intents est de déléguer une action à un autre composant, une autre application ou une autre activité de l'application courante. [16]

✓ **Intention**

La communication interne du système Android est basée sur l'envoi et la réception de messages exprimant l'intention d'une action. Représentant une description abstraite d'une

opération à effectuer, chacun de ces messages peut être émis à destination d'un autre composant de la même application (une activité, un service, etc.) ou celui d'une toute autre application. Issu de la classe Android.content.intent, ce message permet de véhiculer toutes les informations nécessaires à la réalisation de l'action. Les objets Intent ont essentiellement trois utilisations: ils permettent de démarrer une activité au sein de l'application courante ou de solliciter d'autres applications et d'envoyer des informations. [16]

✓ **Notification**

Les notifications représentent un moyen plus standard et efficace de prévenir l'utilisateur d'un événement. Les notifications permettent d'alerter l'utilisateur et de garder un historique des dernières notifications dont l'utilisateur n'a pas encore pris connaissance. Pour être plus efficace, une notification sera souvent combinée à un effet sonore : émission d'un son, utilisation du vibreur du téléphone,... etc. [16]

Besoins matériels et logiciels :

- 2 Go de mémoire RAM, mais on ne va pas se cacher qu'en dessous de 8 Go vous risquez d'être limité.
- Plus de 1,5 Go d'espace disque pour tout installer.
- Niveau processeur, l'émulation ne peut se faire que sur 1 cœur de votre processeur, donc augmenter le nombre de cœurs ne vous servira pas à grand-chose. C'est vraiment la puissance pure qui compte. Il n'y a donc pas de minima mais le plus rapide sera le mieux.
- Nous avons utilisée le système Windows 7. Il est préférable d'avoir un processeur Intel® qui supporte Intel® VT-x, Intel® EM64T (Intel® 64), et l'ExecuteDisable (XD) Bit, de manière à pouvoir accélérer fortement l'émulation.

3.6 Le moteur d'exécution d'Android

C'est cette couche qui fait qu'Android n'est pas qu'une simple « implémentation de Linux pour portables ». Elle contient certaines bibliothèques de base du Java accompagnées de bibliothèques spécifiques à Android et la machine virtuelle « Dalvik ».

Un moteur d'exécution (« *runtime system* » en anglais) est un programme qui permet l'exécution d'autres programmes. Pour utiliser des applications développées en Java sur un ordinateur on a besoin du JRE (« Java Runtime Environment »). Il s'agit du moteur d'exécution nécessaire pour lancer des applications écrites en Java.

La figure 3.5 est un schéma qui indique les étapes nécessaires à la compilation et à l'exécution d'un programme Java standard.

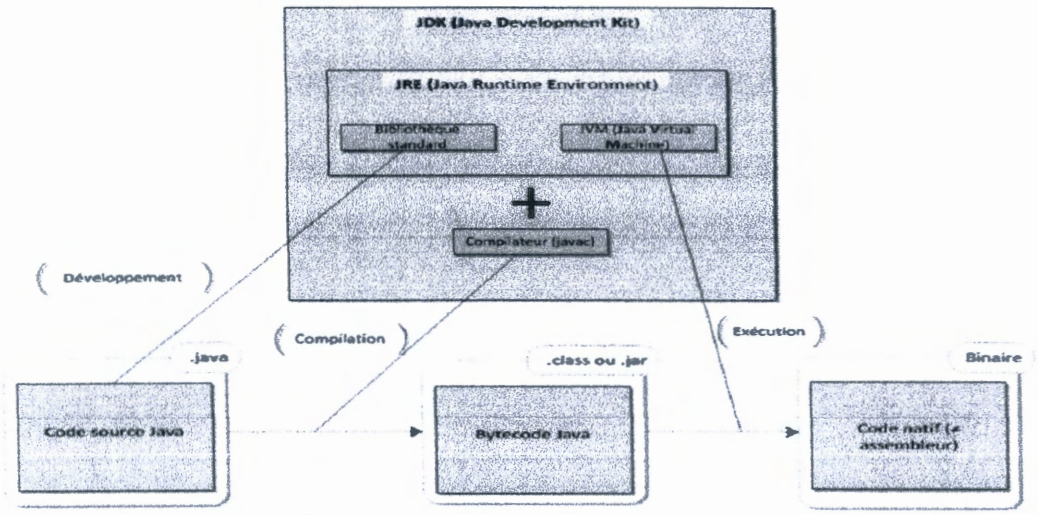


Figure 3.5 : présente les étapes nécessaires à la compilation et l'exécution.

« Dalvik » cette machine virtuelle est optimisée pour mieux gérer les ressources physiques du système. Elle permet par exemple de laisser moins d'empreinte mémoire (la quantité de mémoire allouée à une application pendant son exécution) ou d'utiliser moins de batterie qu'une machine virtuelle Java.

La plus grosse caractéristique de Dalvik est qu'elle permet d'instancier un nombre très important d'occurrences de lui-même : chaque programme a sa propre occurrence de Dalvik et elles peuvent s'exécuter sans se perturber les unes les autres. La Figure 3.6 est un schéma qui indique les étapes nécessaires à la compilation et à l'exécution d'un programme Android standard.

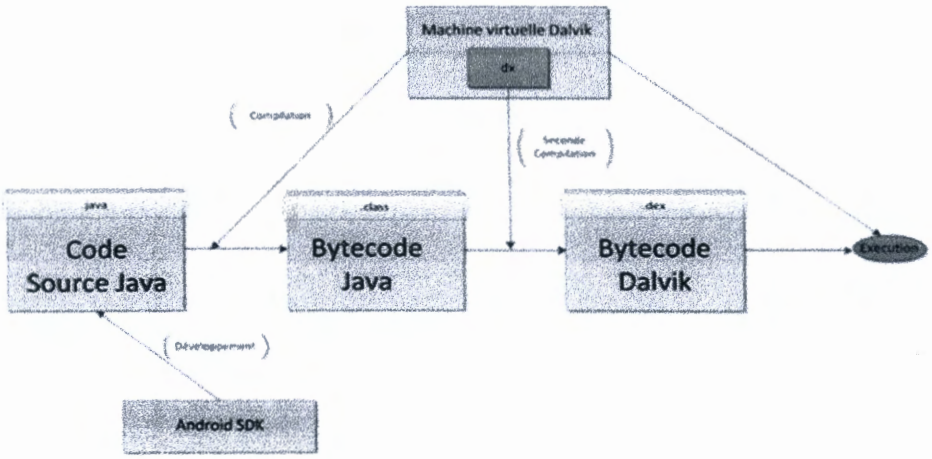


Figure 3.6 : La machine virtuelle Dalvik.

Le code Java est converti en bytecode Java .Le bytecode Java ne pouvait être lu que par une machine virtuelle Java, mais que Dalvik n'était pas une machine virtuelle Java. Il faut donc procéder à une autre conversion à l'aide d'un programme qui s'appelle « dx » qui s'occupe de traduire les applications de bytecode Java en bytecodeDalvik, qui, lui, est compréhensible par la machine virtuelle.

3.6.1 Configuration de l'appareil mobile réel

Pour configurer un appareil mobile avec une machine il est nécessaire d'installer le driver de l'appareil mobile (voir figure 3.7) et d'activer le débogage (voir figure 3.8)

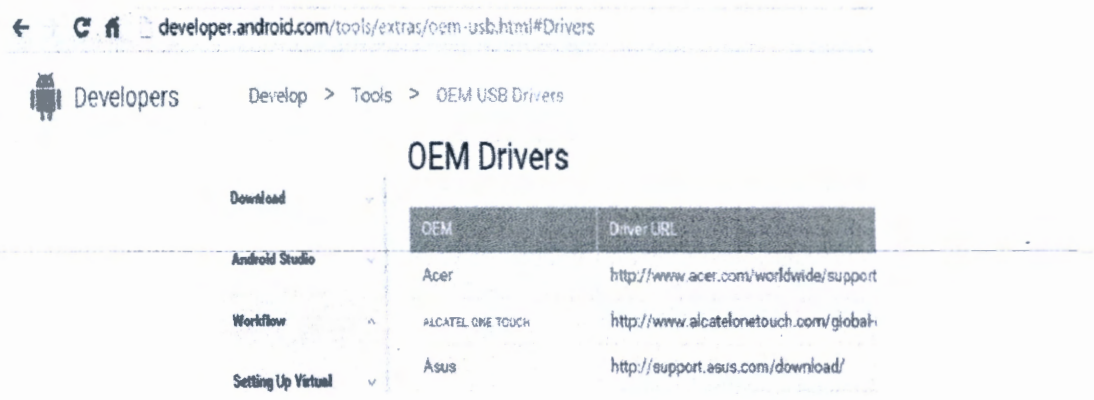


Figure 3.7 : installation du driver de l'appareil mobile.

Figure3.8 : les étapes de configuration d'un appareil mobile.

Les limites d'un appareil Smartphone dédié au développement

- Une puissance processeur plus faible
- Une RAM limitée
- Des capacités de stockage permanent limitées
- De petits écrans avec de faibles résolutions
- Des coûts élevés de transfert de données
- Des taux de transfert plus lents avec une latence élevée
- Des connexions réseau moins fiables
- Des batteries à autonomie limitée

3.7 Avantages d'Androïd

✓ **Open source** Le contrat de licence pour Android respecte les principes de l'open source, alors on peut à tout moment télécharger les sources. Android utilise des bibliothèques open source puissantes, comme par exemple SQLite pour les bases de données et OpenGL pour la gestion d'images 2D et 3D (pour faire des jeux).

✓ **Gratuit :** Android est gratuit, pour produire notre propre téléphone sous Android c'est gratuit. En revanche, pour poster des applications sur le Play Store, ça coûtera la modique somme de 25$.

✓ **Facile à développer** Une API, ou « interface de programmation » en français, est un ensemble de règles à suivre pour pouvoir dialoguer avec d'autres applications. Par exemple, je peux demander à Google Maps de m'afficher la carte de Paris c-à-d, quelles méthodes sont à utiliser avec quels paramètres et ce qu'elles retournent.

✓ **Facile à vendre :** Le Play Store (anciennement Android Market) est une plateforme immense et très visitée, c'est donc une mine d'opportunités pour quiconque veut diffuser une application dessus.

✓ **Flexible** Le système est extrêmement portable, il s'adapte à beaucoup de structures différentes. Les Smartphones, les tablettes.

✓ **Complémentaire** L'architecture d'Android est inspirée par les applications composites, et encourage par ailleurs leur développement. Ces applications se trouvent essentiellement sur internet et leur principe est que vous pouvez combiner plusieurs composants totalement différents pour obtenir un résultat surpuissant. Par exemple, si on combine l'appareil photo avec le GPS, on peut poster les coordonnées GPS des photos prises.

3.8 Conclusion

Dans ce chapitre, nous avons présenté l'environnement Android Studio. C'est un environnement très simple à utiliser. Il offre des avantages considérables pour le développement des applications smartPhone. Il intègre le SDK d'Android qui permet une programmation de très haut niveau.

4.1 Introduction

Dans ce chapitre nous allons discuter de la conception et du développement de notre système de détection et la mise en correspondance des images .Puis, nous présenterons les résultats expérimentaux obtenus par notre système.

Notre système a été implémenté pour les Smartphones sous le système Androïd, il permet de détecter les points d'intérêt Harris, ces points sont utilisés pour la reconnaissance des objets dans les images.

4.2 Conception du système

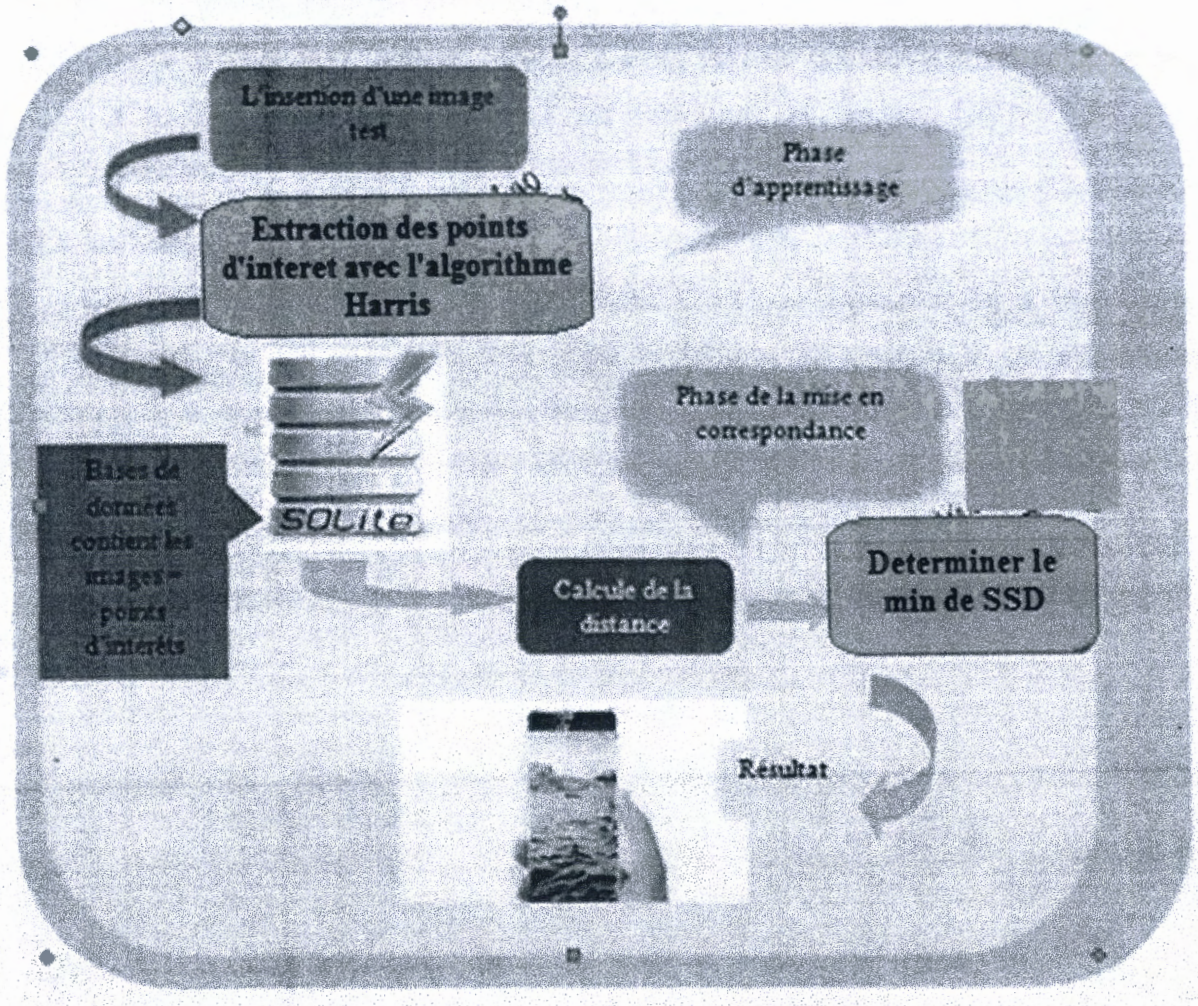


Figure 4.1: Architecture général du système.

4.3 Représentation du projet

Le but de notre projet est de détecter les coins d'une image par le détecteur de Harris qui extrait les points caractéristiques afin de les matcher, puis on a essayé dans l'autre partie,

d'adapter ce programme en une application mobile pour une plateforme Androïd, et émuler notre application sur le PC grâce à un émulateur d'application (Android sur Windows). Enfin, valider l'application et la translater sur un vrai Smartphone.

4.3.1 Environnement et outils du travail :

Pour réaliser notre travail, nous avons utilisé le système d'exploitation Windows 7 qui est compatible avec le langage choisie.

-l'Androïd studio (version 2.0).

- Un émulateur nox .

- Un portable Smartphone Samsung S3 Neo.

- La bibliothèque Open cv version 2.4.10 : Est une bibliothèque bibliothèque libre d'analyse d'images et de vision par ordinateur en langage C/C++. Elle a été développée à l'origine par Intel.

4.3.2 La base de données

On a utilisé la classe DatabaseOperations. C'est une extension de la bibliothèque SQLiteOpenHelper qui représente une base de données open source, et qui supporte les fonctionnalités standards des bases de données relationnelles comme la syntaxe SQL. Elle nécessite peu de mémoire lors de l'exécution (environ 250 kilo octet), ce qui en fait un bon candidat pour être intégré dans d'autres environnements d'exécution.

Pour l'apprentissage, nous avons choix un ensemble de 4 images de la base de données VOC 2006.

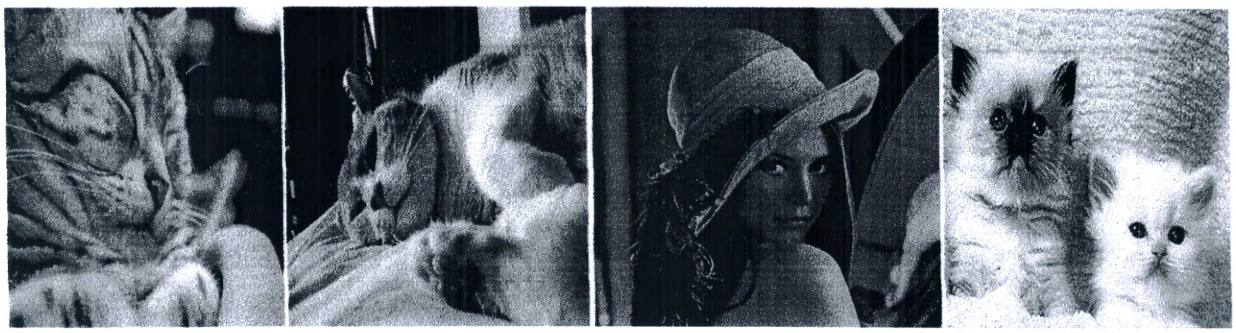


Figure4.2 : Quelques images mis dans la base de donnés.

La structure de la base de données qu'on a utilisée est la suivante :

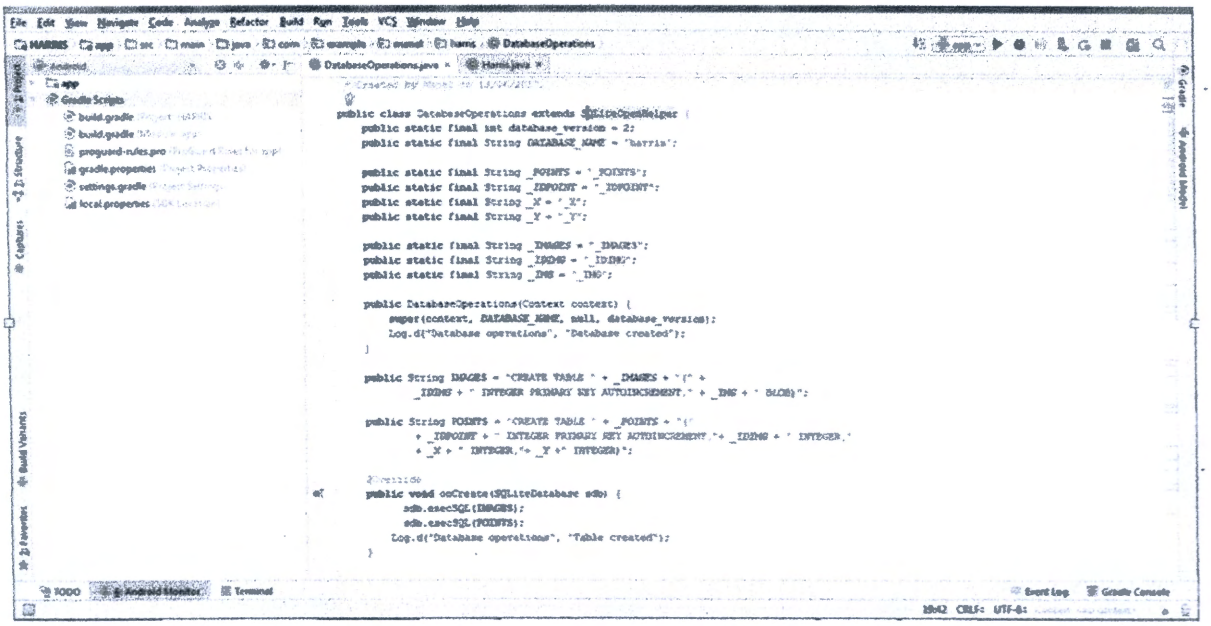


Figure4.3 : structure de la base de donnés utilisé.

4.3.3 Apprentissage

Dans cette étape, nous essayons d'identifier les points d'intérêt dans l'image à l'aide de la méthode de détection de coin Harris. Les étapes sont les suivantes, ou chaque point de l'image, considérez une fenêtre de pixels autour de ce point.

$$E(x,y) = \sum_{u,v} w(u,v)\,[I(x+u,y+v) - I(u,v)]^2$$

L'algorithme :

1. On transforme l'image en niveaux de gris.
2. Pour tout pixel p de l'image on calcule les dérivées partielles I en X et Y.
3. Pour tout pixel p de l'image on calcule les produits des dérivées partielles I^2_x et I^2_y.
4. On applique ensuite le filtre Gaussien sur l'image dérivée en a utilisée une fenêtre de 3x3 et la matrice de convolution $\{1, 2, 1\}$ $\{2, 4, 2\}$ $\{1, 2, 1\}$
5. Apres on a définir pour chaque pixel la matrice M.

$$M = \begin{bmatrix} G\,I^2x & \cdots & GIx\,Iy \\ \vdots & \ddots & \vdots \\ GIx\,Iy & \cdots & G\,I^2y \end{bmatrix}$$

6. Ensuite en calcul la réponse du détecteur R de Harris pour chaque pixel

$$R = Det(M) - k.\,tr(M)^2$$

D'où k: est un constante exprime expérimentalement il est montrée que les valeurs possible sont $0.04 \leq k \leq 0.06$ (on a fixée le paramètre k=0.05).

Det : determinant de la matrice $Det = G\,I^2x * G\,I^2y - (GIx\,Iy)^2$

Tr : trace de la matrice $Tr = G\,I^2x + G\,I^2y$

$$R = G\,I^2x * G\,I^2y - (GIx\,Iy)^2 - k(G\,I^2x + G\,I^2y)^2$$

7. les points d'intérêt sont des pixels qui ont une valeur de R supérieurs à un seuil.

Notre seuil =900000000.

Figure 4.4 : lenna avec les points d'intérêt.

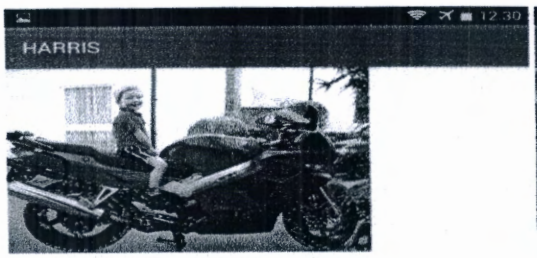
8. ensuite on a réduits les points d'intérêt avec l'algorithme ANMS (Suppression Adaptive Non-Suppression Maximale), afin qu'on puisse sauvegarder l'exécution de l'algorithme avec moins de points d'intérêt qui couvrent l'espace de l'image plus uniformément. Les points d'intérêt sont supprimés en fonction de la force d'angle et seuls ceux qui sont au maximum dans un quartier de rayon r pixels sont conservés. Sur le plan

conceptuel, nous initialisons le rayon de suppression, puis on les augmente jusqu'à ce que le nombre souhaité d'intersection des points d'intérêt soit obtenu.

Les points résultats sont les points qui ont une force d'intensité plus élevé et par la suite nous les stockons dans la base de données DatabaseOperations.

Figure 4.5 : lenna après la réduction des points d'intérêt.

Voila quelques exemples d'exécution :



جامعة محمد الصديق بن يحي
مكتبة الكلية
02

Figure 4.6 : Exemple de l'exécution de notre système.

4.3.3 mise en correspondance

Dans cette étape nous avons utilisé le classifieur de k-ppv pour calculer la similarité entre l'image requête et celles de la base d'apprentissage, un seuil de classification a été fixé expérimentalement. Les descripteurs similaires au descripteur modèle (requête) sont ceux qui ont une valeur distance SSD (Sum of Squared Differences) minimal pour le calcule de similarité.

Algorithme SSD (Sum of Squared Differences)

Déclarations :

Soient : image_requete et image_base : deux images de taille NxM
P1(x1,y1) et P2(x2,y2) :deux points Harris corresponds aux images.

```
Fonction SSD(image_requete, image_base , x1,y1,x2,y2)
  Début
    somme=0;
      pour  i = -1; i < 2; i++ ;
        pour  j = -1; j < 2; j++;

          somme=somme + (image_requete.getpixel(x1+i, y1+j)-
                         image_base.getpixel (x2+i, y2+j)) ² ;
        Finpour
      Finpour
      return somme;
  Fin
Fin Fonction
```

Conclusion générale

Dans ce projet de fin d'étude, nous somme intéressés à la conception et à la réalisation d'une application sous système Android.

Notre travail a été organisé en trois parties, la première était de développer un détecteur de Harris pour une extraction parfaite des points d'intérêt des images en utilisant l'environnement de développement Android Studio, par la suite dans la deuxième partie on a réalisé l'apprentissage du système a fin de calculer l'appariement qui est le calcule de la ressemblance entre images, à la fin on a adapté ce programme dans un émulateur, et ainsi assurer la portabilité effective de l'application sur Smartphone.

A l'issue de la réalisation de ce projet de fin d'étude, nous pouvons affirmer que notre projet nous a été d'une grande utilité dans la mesure où il nous a permit de nous familiariser avec plusieurs outils à savoir :

- Maitriser les connaissances liées aux détecteurs de points caractéristiques locaux : points Harris, Sift, Surf, ect, à partir d'objets visuels dans l'image
- Le développement avancé des applications avec JAVA
- Apprendre le fonctionnement des différentes plateformes mobile
- Apprendre des connaissances avancées sur le développement des applications mobile, en particulier sous la plate-forme Android
- Pouvoir développer et émuler une application pour vision par ordinateur, sur un environnement (tel que Andoid Studio) pour Smartphone Andoid

En perceptive, pour améliorer notre produit et faciliter les taches à l'utilisateur, nous pensons à : développer ce système afin de pouvoir faire l'indexation et la reconnaissance des images.

[1] Http://www.larousse.fr/encyclopedie/divers/intelligence_artificielle/187257

[2] Http://cache.media.enseignementsupercherche.gouv.fr/file/Actus/86/1/Conclusion Groupe Travail8France_IA738861.pdf

[3] Introduction à la vision par ordinateur pour débutants » (*version du 21 octobre 2014 sur l'Internet Archive*), Refrobot, doc pdf.

[4] Vision par ordinateur outils fondamentaux Radu HORAUD CNRS et Olivier MONGA INRIA Deuxième édition Ouvrage rédigé avec le concours du Ministère de l'enseignement supérieur et de la recherche DIST Editions Hermes.

[5] Richard O. Duda, Peter E. Hart, David G. Stork, *Pattern Classification*, Wiley-interscience, 2001 (ISBN 0-471-05669-3) [détail des éditions]

[6] Http://devernay.free.fr/cours/vision/pdf/c4.pdf

[7] THÈSE DE DOCTORAT Présentée par Ibrahim GUELZIM, Discipline : Sciences de l'ingénieur. Spécialité : Informatique et Télécommunications Sujet de la thèse : CONTRIBUTIONS AUX TRAITEMENTS D'IMAGES PERSPECTIVES ET OMNIDIRECTIONNELLES PAR DES OUTILS STATISTIQUES. Soutenue le 12 mai 2012.

[8] Parisot, P, Suivi d'objets dans des séquences d'images de scènes déformables : De l'importance des points d'intérêt et du maillage 2D. Thèse de doctorat. Institut National Polytechnique de Toulouse. 2009.

[9] Quelques idées sur le bas niveau en vision par ordinateur Marie-Odile Berger, INRIA Nancy Grand Est Equipe Magrit, doc pdf

[10] Projet de fin d'études Reconnaissance d'images sur smartphones Vincent Angladon IMA 2013

[11] Smith, S.M. and Brady, J.M, "SUSAN - a new approach to low level image processing". International Journal of Computer Vision 23. pp:45-78. (May 1997).

[12] Rosten, E. and Drummond, T., "Machine Learning for High-Speed Corner Detection". ECCV. pp: 430-443. 2006.

[13] Rapport de stage : « Caractérisation de la matrice des discontinuités et du massif rocheux », TROCHON Marie-Laure, 2003, TROCHON Marie-Laure.pdf.

[14] Elise Arnaud, « Traitement d'images Détection de contours », Université Joseph Fourrier, dept-info.labri.fr/~vialard/Traitement/cours/cours4.pdf

[15] MEMOIRE Présenté par : Mr. Yacine GAFOUR Pour l'obtention du diplôme de Magister en Informatique Thème DETECTION DE VIDEOS SIMILAIRES soutenu Le 09 / 04 / 2014 à USTO

[16] P. Forssen, « Détecteurs de région », cour image www.micc.unifi.it/delbimbo/wp-content/.../A34%20MSER.pdf

[17] Rapport de stage : Détection et identification de points homologues par corrélation d'images Joan ESPADALER RODÉS 2 juillet 2012 - 21 décembre 2012

[18] Cours : « Informatique visuelle - Vision par ordinateur Extractions de caractéristiques - les points d'intérêt » Elise Arnaud.pdf

[19] Lindeberg, T. Edge detection and ridge detection with automatic scale selection. Int. J.Comput Vision; vol 30(No 2):117-156. 1998.

[20] Mikolajczyk, K. and Schmid, C., "Scale and affine invariant interest point detectors"Int. J. Comput. Vision, Volume 60.Number 1.pp: 63-86. 2004.

[21] Openclasseroom.com

[22] Http://www.tutos-android.com/introduction-a-android-studio)

[23] Cours Android Studio Dr chaoueche Université de Constantine.