République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE
Département d'automatique

MEMOIRE

de fin d'études

en vue de l'obtention du diplôme d'Ingénieur d'Etat

en Automatique

THEME

Tracking 3-D d'objets rigides à partir d'une séquence d'images.
Application à l'estimation de la pose d'une caméra en temps réel.

Dirigé par :

Pr. S. Ameur

Présenté par :

M. AIT TALEB Mohamed Nabil

Promotion : 2008-2009

Remerciements

Une pensée pieuse à Dieu qui a éclairé notre chemin et mené vers la concrétisation de ce modeste travail.

Nous tenons à remercier notre promoteur Monsieur le Professeur AMEUR Soltane pour sa disponibilité, son aide précieuse et de nous avoir fait profiter de sa rigueur scientifique, de son expérience et de nous avoir encouragé tout au long de ce travail.

Nous exprimons notre sincère gratitude aux chefs des départements d'Automatique M. BENSIDHOUM et d'Informatique M. DEMRI qui nous ont permis de collaborer.

Nos remerciements s'adressent aussi aux membres de Jury qui nous ont fait l'honneur de juger ce modeste travail.

Enfin, nos remerciements les plus sincères à nos familles et à tous ceux qui ont contribué de près ou de loin à la concrétisation de ce mémoire.

Dédicaces

Je dédie ce modeste travail à :

- *La mémoire de mon très cher grand-père que Dieu l'accueille dans Son Vaste Paradis.*
- *Mes très chers parents*
- *Mon frère Hakim et ma sœur Hanane*
- *Mes grands-parents*
- *Mes tantes et oncles*
- *Ma grande famille paternelle et maternelle*
- *A tous mes amis*

Nabil

Introduction

Chapitre 1 : Notions de base sur les systèmes d'acquisitions

Chapitre II : Description générale du problème de tracking 3-D.

Chapitre III : Détection et mise en correspondance des points d'intérêt.

Chapitre IV : Estimation de la pose de la camera.

Chapitre V : Application.

Introduction

Introduction

Le tracking 3-D a pour but de reproduire une caméra virtuelle recréant les moindres mouvements effectués par l'objet portant la caméra. C'est une branche commune entre les domaines très liés de la photogrammétrie et de la vision par ordinateur. Le tracking 3-D permet notamment d'obtenir des informations tridimensionnelles, à partir d'image bidimensionnelles. Le plus souvent, le but recherché est la localisation d'une caméra ou du robot sur lequel elle est embarquée dans un environnement connu ou inconnu. La vision 3-D propose toute une gamme de solutions pour la résolution ces problèmes de tracking. Nous consacrons notre étude à la considération des parties concernant l'estimation d'entités géométriques en s'appuyant sur l'extraction des primitives d'images (essentiellement les points d'intérêts).

Les principaux objets de recherche du domaine sont le développement d'algorithmes numériques et la preuve théorique des conditions minimales pour l'existence de solutions. Ces recherches sont d'un côté motivées par des problèmes pratiques concrets, de l'autre côté souvent par la volonté de résoudre des problèmes en utilisant de moins en moins d'informations.

Le domaine du tracking 3-D est particulièrement vaste du fait qu'il existe un grand nombre d'approches et de domaines d'applications, et aussi par ce qu'il existe beaucoup d'approches possibles pour résoudre le même problème.

Dans notre projet, nous nous focaliserons sur les modèles on-line en utilisant une seule camera. Les systèmes à cameras multiples sont plus complexes et ne feront pas l'objet d'étude dans notre mémoire. On considérera uniquement les objets et les scènes rigides, contrairement aux déformables et articulés (comme le corps humain par exemple) qui est un domaine plus complexe.

Beaucoup d'autres technologies ont été essayées dans le tracking 3-D mais elles ont toutes leurs faiblesses : les capteurs mécaniques ne sont pas assez précis, ils contraignent l'utilisateur à travailler dans un espace limité. Les capteurs magnétiques sont vulnérables aux distorsions causées par le métal présent dans l'environnement de travail, ce qui est une situation courante. Les capteurs à ultrasons souffrent du bruit et tendent à être imprécis lorsqu'ils sont employés pour de longues distances à cause de la variation de la température ambiante qui affecte considérablement la vitesse de propagation du son.

La vision artificielle a l'avantage de produire des solutions non encombrantes, précises et moins coûteuses. Néanmoins, celles-ci requièrent un investissement en effort considérable dans l'élaboration d'algorithmes robustes. Dans certains cas il est acceptable d'ajouter des repères ou des marqueurs spécifiques telles que des LEDs pour faciliter la détection des primitives (points, contours…). Cette technique n'est pas souhaitable dans certains cas de la réalité augmentée où les utilisateurs trouvent des difficultés pour placer les marqueurs. Dans ces cas là, il est préférable de travailler avec les caractéristiques naturelles comme les contours, les coins et les textures. Evidemment, ceci rend le tracking beaucoup plus difficile car certains objets possèdent peu de caractéristiques exploitables.

Dans le chapitre 1, nous rappelons dans un premier temps le fonctionnement de base des appareils d'acquisition, ce qui permettra de comprendre le processus d'acquisition puisque notre travail consistera à modéliser le processus inverse. Dans un second temps, nous présenterons brièvement une méthode pour la correction de la distorsion optique causée par le dispositif optique sur les images.

Le second chapitre sera consacré à l'exposition de quelques problèmes liés à la vision par ordinateur et s'intéressera à la modélisation mathématique du problème.

Le troisième chapitre sera dédié aux techniques de détection de points d'intérêts dans les images.

Dans le quatrième chapitre, nous verrons des techniques d'estimations des paramètres de calibrage de la camera et des technique récentes et avancées qui cherchent à améliorer la robustesse de cette estimation.

Le cinquième chapitre sera consacré au développement d'une application mettant en œuvre une approche pour le tracking 3-D.

Chapitre I : Notions de base sur les systèmes d'acquisitions

Préambule

Les cameras sont conçues pour enregistrer des séquences d'images. Au cœur de ce processus se trouvent des lentilles servant à focaliser la lumière sur un support d'acquisition (film pour analogique et CCDs pour numérique).
Les séquences d'images étant capturées dans un support d'enregistrement, celui-ci reconstituera la scène lorsqu'il sera déroulé à la même vitesse d'enregistrement. La majorité des caméras analogiques enregistrent à une vitesse de 24 images par secondes. Ces vitesses varient selon le standard considéré (30 pour le format NTSC et 25 pour le format PAL). Le choix du standard est lié aux différentes utilisations de la séquence d'images. Dans ce chapitre nous présenterons les éléments essentiels de ce type de capteurs.

I.1 Les lentilles

Les lentilles utilisées dans la réalisation de caméras sont conçues avec différents degrés de précisions. 'L'objectif' de la caméra comprend une multitude de lentilles appelées éléments (figure 1.1). On distingue les lentilles principales qui ont pour but de focaliser la lumière avec précision sur le plan du film et les lentilles auxiliaires qui permettent de corriger les imperfections telles que la distorsion et les aberrations chromatiques. Dans la plupart des cas, ces éléments sont enduits avec une fine couche d'un film qui réfracte ou filtre la lumière afin que la qualité de l'image soit meilleure. Outre ces éléments, l'objectif contient aussi un iris ou un diaphragme qui contrôle la quantité de lumière projetée sur le film.

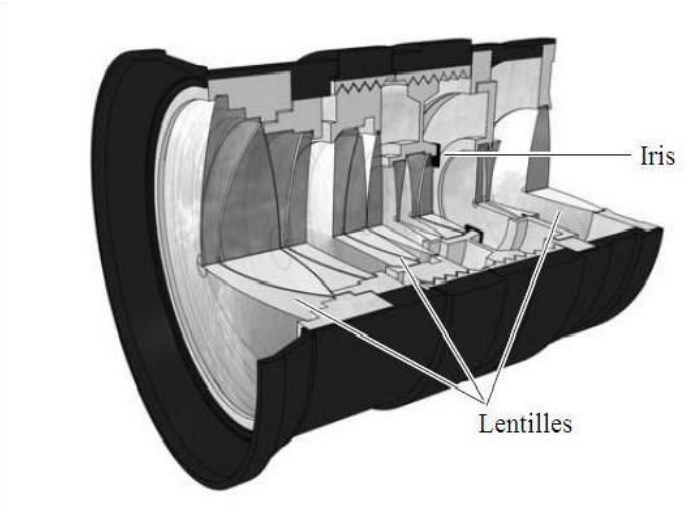


Figure 1.1 : Les composantes d'un objectif.

I.2 La distance focale

La caractéristique de la lentille qui a le plus d'incidence sur l'image finale est sans doute la distance focale. Elle est définie comme étant la distance entre le centre optique (centre de la lentille principale) et le plan de projection. Une courte distance focale produit un effet de perspective plus prononcé, un large sens de profondeur et un nombre élevé d'objets visibles de la scène (voir figure 1.2.a).

Une distance focale plus longue produit des images montrant un aspect plus plat de la scène avec un sens moins évident de perspective et aussi moins d'objets visibles de la scène (voir figure 1.2.b).

(a) (b)

Figure 1.2 : différence entre courte et longue distance focale.

Certaines lentilles sont appelées lentilles fixes, car leur distance focale reste inchangée. On distingue aussi les lentilles variables (ou lentilles de zoom), appelées ainsi en raison du caractère variable de leur distance focale. Le zoom d'une lentille variable est possible grâce à un ensemble d'éléments sis à l'intérieur de l'objectif qui se meuvent le long du cylindre, ceci a pour conséquence l'augmentation ou la diminution de la distance focale. Lorsque la lentille est en zoom (on part d'une distance focale donnée à une autre plus longue), les objets de la scène paraissent de plus en plus proches et remplissent davantage le cadre de l'image. Cela s'apparente à un déplacement d'une camera avec une lentille à distance focale fixe le long de l'axe principal en direction du sujet, mais il subsiste une différence subtile entre l'action de zoomer et une translation en profondeur. La figure 1.3 illustre bien cette différence, en effet on constate clairement que les relations des arrières et avant plans des éléments changent radicalement entre les deux.

Zoom **Translation en profondeur**

Figure 1.3 : Différence entre un zoom et un déplacement en profondeur**.**

Lorsqu'on analyse une image pour déterminer si la camera est en zoom ou en translation en profondeur, il suffit de vérifier la relation entre l'avant et l'arrière plan. S'ils ne se déplacent pas indépendamment, alors il y a de fortes chances que ce soit un zoom.

I.3 Champs et angles de vue (CDV et ADV)

Comprendre comment le plan de projection et la distance focale sont reliés requiert un peu d'exercice mental. Si on étend des lignes imaginaires à partir des quatre coins du film vers les bords de la lentille, on obtient une forme pyramidale qui s'élargit de plus en plus (connue sous le nom technique de Frustum) définissant un champ de vue pour une configuration particulière des lentilles et du plan de projection (Figure 1.4).
Le CDV est la mesure de ce que l'on peut actuellement voir dans la scène. Si on change l'une ou l'autre composantes en question (plan de projection et/ou distance focale), l'angle de vue (ADV) sera automatiquement affecté.
Il est aussi important de savoir que le CDV et l'ADV sont souvent utilisés de façon interchangeable. Cependant, les deux notions sont différentes **(*Figure 1.5*).** ADV mesure la portion du cercle visible par la caméra.

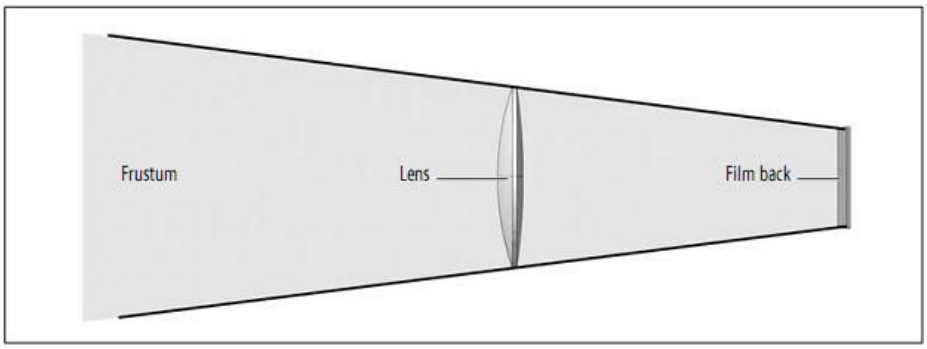


Figure 1.4 : Frustum.

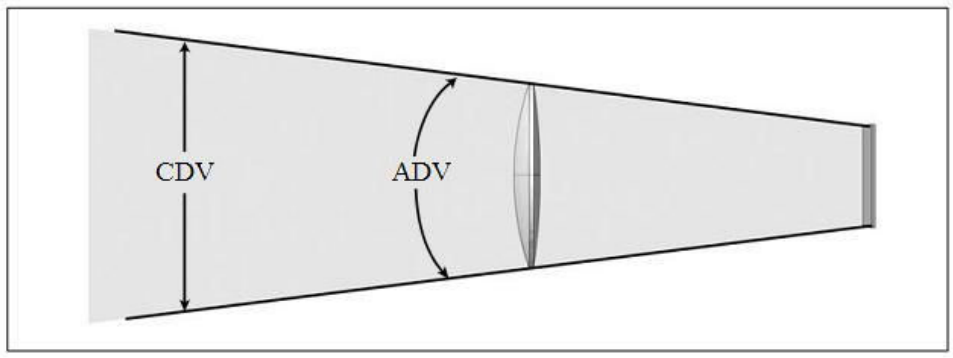


Figure 1.5 : Angle et champ de vue.

ADV est usuellement mesuré en degrés, par exemple, si la lentille a un ADV de 90°, elle peut couvrir approximativement un tiers de la scène de gauche a droite. Il y a une relation inverse entre la distance focale de la lentille et son ADV. Plus la distance focale est grande, moins est l'ADV et vice versa. Et c'est pourquoi les petits objectifs sont généralement connus pour leur large ADV vu la petitesse de leur distance focale. Par exemple, l'utilisation d'un objectif long signifie que le centre optique est loin du plan de projection, ce qui produit un frustum étroit et un ADV plus petit. Utiliser un plan de projection plus grand en laissant la distance focale comme telle aura le même effet.

I.4 Distorsion optique

La distorsion est une altération de la prise de vue d'un dispositif d'acquisition d'images tel qu'un appareil photo ou une camera. La distorsion est due à la position du diaphragme dans le système optique : plus le diaphragme est loin du centre optique, plus la distorsion est forte. Il existe deux formes de distorsions : radiale et tangentielle. La première a tendance à arrondir les bords de l'image. Cette distorsion est souvent plus flagrante sur des appareils à courte focale (grands angles).La seconde applique une sorte de rotation autour du centre de l'image. L'amplitude de cette rotation est variable suivant la position du point traité sur l'image. Cette distorsion est toutefois négligeable et nous ne la traiterons pas dans ce sujet.

Le diaphragme (tel un iris dans une lentille) limite l'accès de la lumière à l'intérieur de l'objectif. Si ce diaphragme est assez proche de la lentille principale, alors l'image capturée à travers cette lentille est relativement intacte (sans distorsion). Néanmoins, si le diaphragme est à une distance significative ou derrière la lentille, les choses seront différentes. Considérant la lumière qui traverse la lentille, par définition le rayon qui passe par le centre du diaphragme doit toujours parvenir au plan de l'image au centre du plan de projection (figure 1.6). Mais que dire de la lumière qui passe a travers les bords du diaphragme? Si le diaphragme est devant la lentille alors ces rayons de lumière peuvent atteindre la lentille avec un angle oblique causant des courbures vers le centre de l'image et dont il résulte distorsion en barillet. Vice versa si le diaphragme est derrière la lentille, dans ce cas les seuls rayons autorisés à passer a travers le diaphragme sont ceux autour des bords, ces rayons tendent à se courber vers l'extérieur et sont déformés en croissant.

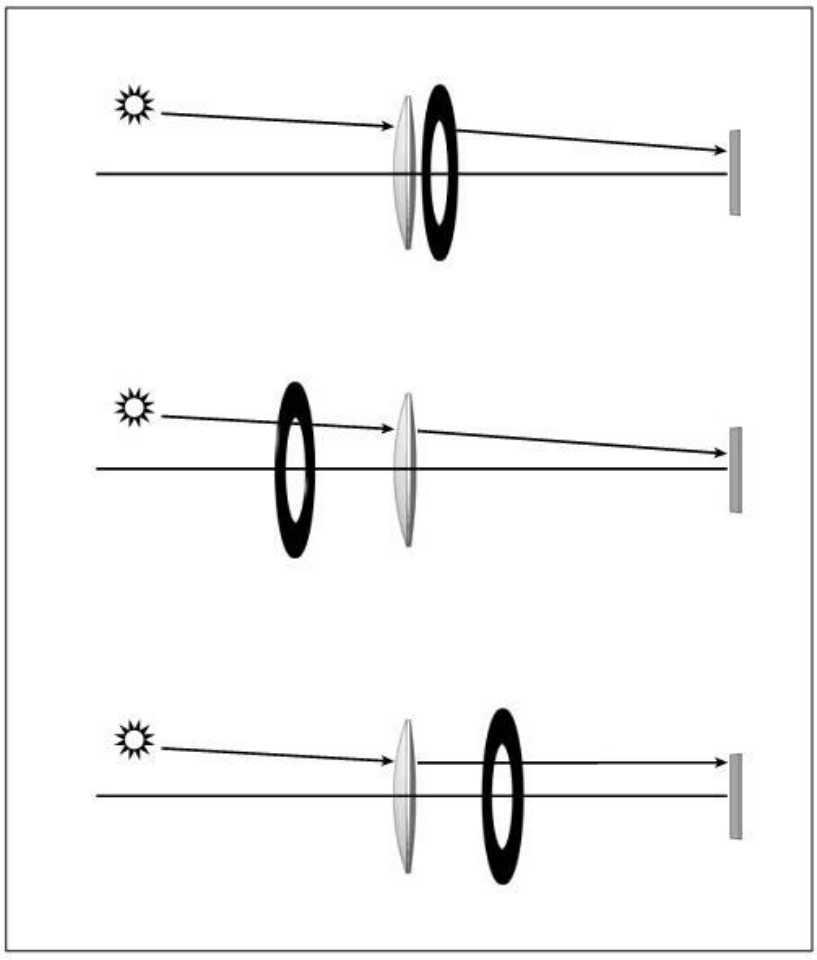


Figure 1.6: Les différentes positions du diaphragme

La distorsion est plus visible avec des lentilles à grands angles et\ou de zoom, et peut être très difficile et compliquée de la corriger dans de larges traitements séquentiels. On doit souvent corriger les distorsions des images trop flagrantes avant leurs utilisations dans le tracking 3-D.

Heureusement, la distorsion optique peut être modélisée par une transformation 2-D de l'image, donc pouvant être corrigée de manière algorithmique. Elle se présente en deux formes, radiale et tangentielle. Toutefois, cette dernière est négligeable vu qu'elle n'a pas d'effet néfaste sur l'image originale.

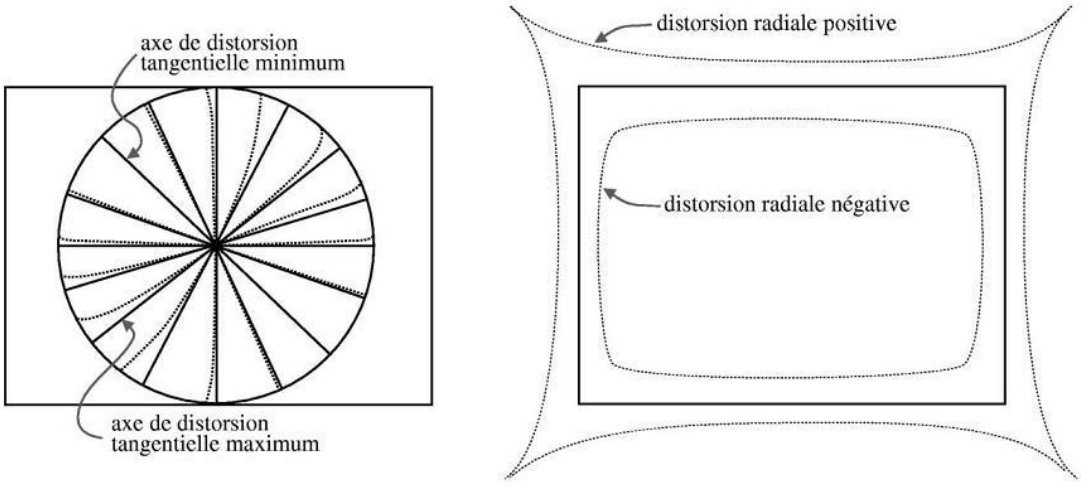


Figure 1.7 : Distorsion tangentielle et radiale**.**

I.4.1 Modélisation de la distorsion radiale [5]

Comme son nom l'indique, la distorsion radiale affecte différemment les points selon leur distance par rapport au centre de distorsion. La distorsion radiale révèle un caractère non linéaire ce qui complique légèrement sa modélisation. Elle peut-être modélisée par un polynôme vérifiant certaines contraintes.

Soit (x, y) le centre de distorsion, il s'agit en général du point principal qui correspondra à l'intersection de l'axe optique avec limage. Il arrive que le centre de distorsion soit décentré par rapport à l'image dans le cas d'assemblages optiques de mauvaise qualité.

Ce centre de distorsion peut toutefois être assimilé au centre de l'image sans trop d'effets néfastes sur le modèle. Soit (x_i, y_i) un ensemble de points sur l'image distordue et $(\hat{x}_i, \hat{y}_i)$ les points correspondants sur l'image rectifiée **(Figure 1.8).**

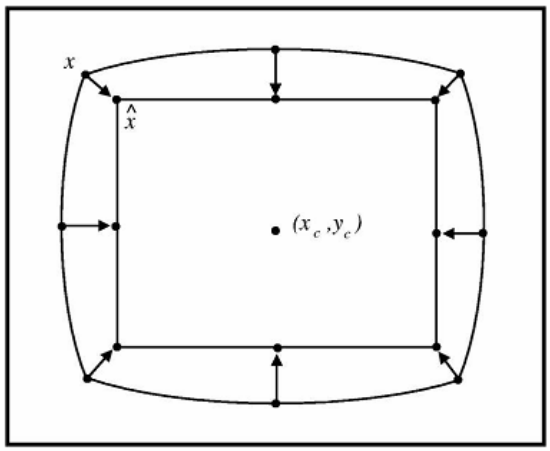


Figure 1.8 : Modélisation de la distorsion radiale.

La relation entre ces deux ensembles de points peut s'écrire de la façon suivante :

$$\begin{cases} \hat{x}_i = x_i + \Delta x \\ \hat{y}_i = y_i + \Delta y \end{cases} \quad 1.1$$

Avec

$$\begin{cases} \Delta x = (x_i - x_c)(k_1 r^2 + k_2 r^4) \\ \Delta y = (y_i - y_c)(k_1 r^2 + k_2 r^4) \end{cases} \quad 1.2$$

Et

$$r^2 = (x_i - x_c)^2 + (y_i - y_c)^2 \quad 1.3$$

Nous utilisons ici un polynôme de degré 4 en ne gardant que les puissances paires, ce qui est réputé pour être suffisant, il est toutefois possible d'utiliser un polynôme de degré 6, toujours en ne gardant que les puissances paires. La correction de la distorsion radiale est donc intimement liée a la recherche des coefficients k_1, k_2 , ... kn .

I.4.2 Correction de la distorsion optique [5]

Avant de s'attaquer au problème de distorsion, il faut préalablement préparer les données à l'aide de méthodes de traitement d'images et de géométrie.

- **Algorithme de correction**

La méthode proposée se décompose en 6 étapes. Certaines d'entres-elles sont indépendantes mais il est conseillé de les traiter dans l'ordre. Pour pouvoir effectuer la correction automatiquement, nous proposons l'emploi d'une mire (feuille en papier) composée de disques noirs alignés selon un quadrillage défini, sur fond blanc. Connaissant la position des disques sur la mire leur position altérée sur l'image, il est alors possible de paramétrer le modèle de distorsion et finalement de corriger cette distorsion.

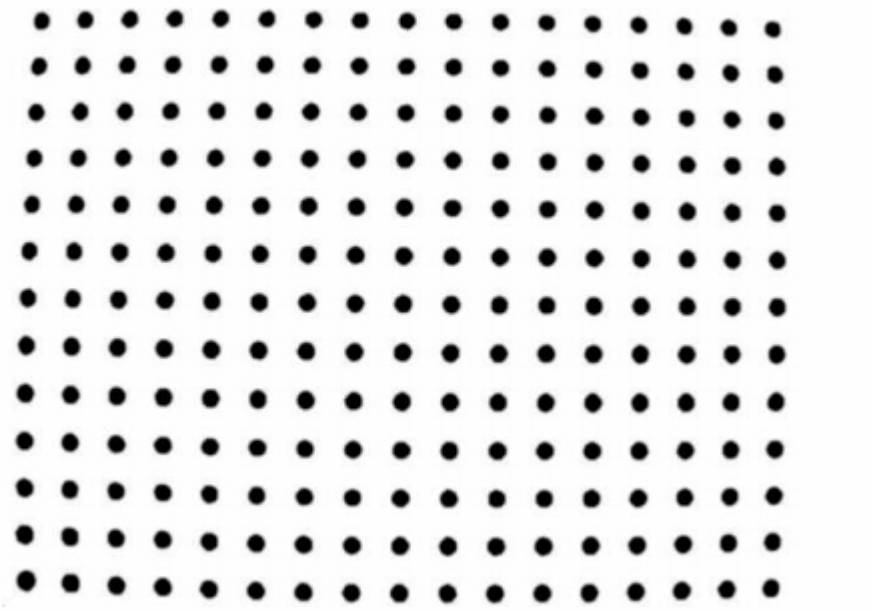


Figure 1.9 : Exemple d'une mire.

Les 6 étapes sont:

a. La détection des disques sur la mire par seuillage.

b. L'identification des 4 disques situés sur les coins du quadrillage.

c. Le calcul d'une homographie représentant la projection idéale des points de la mire sur une image sans distorsion.

d. L'identification des autres disques et leur mise en correspondance avec les points projetés (de façon idéale) à l'aide de l'homographie calculée dans l'étape 3.

e. Le calcul des paramètres de notre modèle de distorsion radiale.

f. La correction de distorsion sur l'image distordue.

- **Définition d'une homographie :**

Une homographie est une transformation linéaire permettant de décrire la projection d'un plan sur un autre plan. Une telle transformation peut s'écrire sous forme matricielle de la façon suivante :

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad 1.4$$

$X = (x, y, 1)^T$, $X' = (x', y', w')^T = \left(\frac{x'}{w'}, \frac{y'}{w'}, 1\right)^T$ sont les coordonnées homogènes des pixels de l'image.

Image distordue

Image corrigée

Discussion

Ce chapitre a été consacré à la présentation de notions de base nécessaires à la compréhension du fonctionnement interne des capteurs visuels. Les différentes définitions de la distance focale, du champ et angle de vue nous permettront de modéliser dans le prochain chapitre leurs comportements. En effet, notre travail consiste à reproduire le processus inverse de l'acquisition, autrement dit l'extraction des informations géométriques de la camera et ainsi celle de la scène.

Parfois, il se trouve que les images acquises présentent certaines altérations liées aux lentilles de la camera plus particulièrement celles à courte distance focale. Ces imperfections présentent des inconvénients qui peuvent fausser les calculs. Pour y remédier, nous avons présenté les étapes d'un algorithme permettant d'identifier les paramètres de correction.

Vu que l'appareil utilisé dans le cadre de ce projet ne présente pas d'effet de distorsion gênant, raison pour laquelle nous n'avons pas détaillé les étapes de l'algorithme de correction.

Chapitre II : Description générale du problème de tracking 3-D

Position du problème

Nous considérons un ensemble d'images d'une scène statique. Notre but principal est la détermination d'informations tridimensionnelles (mouvement, positions), à partir des images, qui, elles, sont bidimensionnelles. Les images dépendent de la structure de la scène, mais aussi des caractéristiques des cameras. Une définition possible du problème général de la vision 3-D est donc la modélisation de la scène et des cameras, c'est-à-dire l'estimation des paramètres définissant leur géométrie.

La principale source d'informations pour résoudre ce problème est l'ensemble des images ; plus particulièrement, la mise en correspondance des images, c'est-à-dire l'identification des mêmes objets ou primitives géométriques dans différentes images, qui fournissent des indices sur la structure de la scène et le positionnement des cameras. Les algorithmes que nous allons présenter dans la suite, utilisent les points d'intérêt mis en correspondance. Les points sont les primitives les plus utilisées, grâce à leur manipulation algébrique aisée et au fait que la plupart des images réelles permettent d'extraire beaucoup de points d'intérêt. D'autres primitives, telles des droites ou des courbes, sont également utilisées par beaucoup d'approches, mais leur traitement dépasserait le cadre de ce projet.

La structure de la scène est donc modélisée ici par un ensemble de points 3-D. Quant à la géométrie d'une camera, nous distinguons sa géométrie externe – sa position et orientation – de sa géométrie interne – l'ensemble de ses paramètres internes, décrivant ses propriétés optiques et autres, telles que la distance focale ou la taille des pixels. Une caméra dont la géométrie interne est connue, sera appelée calibrée.

II.1 Aperçu de quelques problèmes de modélisation 3-D

Nous décrivons quelques-uns des problèmes de base. Tous ces problèmes sont résolus, mais leur mise en oeuvre demande parfois une certaine expertise. Pour la plupart des problèmes, nous esquissons quelques conditions minimales pour leur solution.

II.1.1 Calibrage

Le calibrage permet d'estimer la géométrie interne et externe de la caméra, quoique souvent, seule la géométrie interne est recherchée (pour une caméra qui se déplace au cours d'une application ultérieure). Ici, la géométrie externe définit la position et l'orientation de la camera par rapport à l'objet de calibrage.
Un calibrage complet à partir d'une seule image requiert que l'objet de calibrage soit tridimensionnel.

II.1.2 Calcul de pose (position et orientation)

Il s'agit d'un sous problème du calibrage, et concerne une caméra dont la géométrie interne est connue (par exemple par calibrage). Le but est d'estimer le positionnement relatif entre la caméra et un objet connu, à partir de l'image de cet objet. La résolution de ce problème est surtout intéressante dans le cas d'objets plans : bien qu'une seule image ne suffit pas pour effectuer un calibrage complet, la pose peut être déterminée. Le calcul de pose, peut donc être très utile dans toute application nécessitant la localisation de la caméra par rapport à la scène,ce qui est le cas en réalité augmentée et en robotique. Dans notre étude, nous accorderons un intérêt particulier à la résolution de ce problème.

II.1.3 Triangulation

La triangulation consiste à estimer la structure de la scène à partir de plusieurs images de celle-ci prises par des caméras dont la géométrie complète (interne et externe) est connue. La figure 2.1 illustre la triangulation de deux points d'une scène, à partir de 2 images.

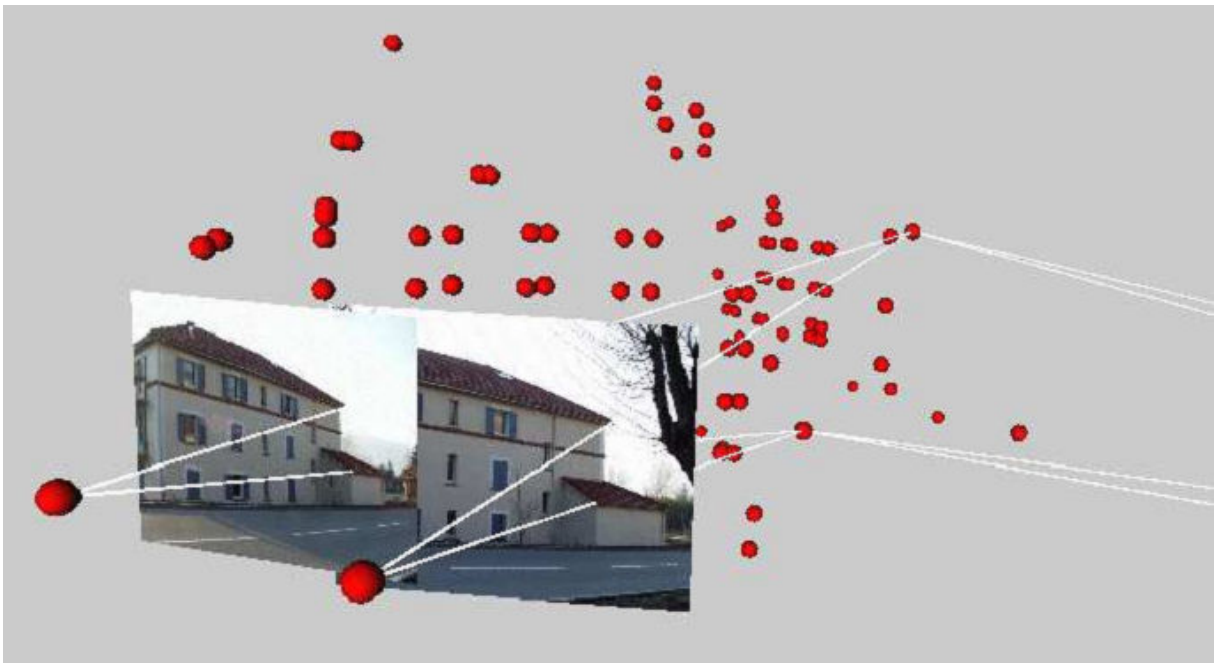


Figure 2.1: Triangulation de deux points d'une scène, à partir de 2 images.

Pour réaliser une triangulation il faut que les centres optiques soient distincts et que le point à trianguler ne se trouve pas sur la droite entre les deux centres optiques.

II.1.4 Estimation du mouvement

Un autre problème classique concerne une caméra en mouvement, dont la géométrie interne est connue (par exemple par un calibrage préalable). Dans ce cas, le mouvement de la camera peut être déterminé à partir d'images même si la scène observée est inconnue. Ensuite, nous nous trouvons dans le cas de figure du paragraphe précèdent, donc l'estimation de la structure de la scène est possible.

Le mouvement ne peut être estimé qu'à une échelle globale, c'est-à-dire que seules des distances relatives, et non absolues peuvent être calculées. L'estimation du mouvement peut être formulée comme un problème géométrique. Les figures 2.2, 2.3 et 2.4 illustrent l'estimation du mouvement entre deux images, prises au cours d'un déplacement d'une caméra. Nous supposons que la géométrie interne de la caméra est connue. Ceci permet en effet de calculer des lignes de vue pour chaque image (voir la figure 2.3). L'estimation du mouvement revient alors à positionner les deux images,

l'une par rapport à l'autre, tel que les lignes de vue correspondant se coupent dans l'espace (voir la figure 2.4).

Figure 2.2: Estimation du mouvement : deux images prises au cours d'un déplacement.

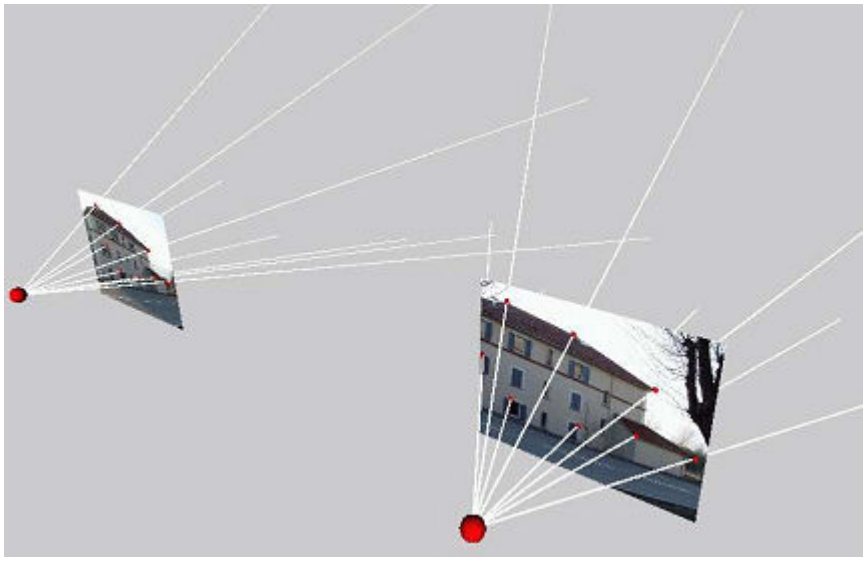


Figure 2.3: Estimation du mouvement : lignes de vue.

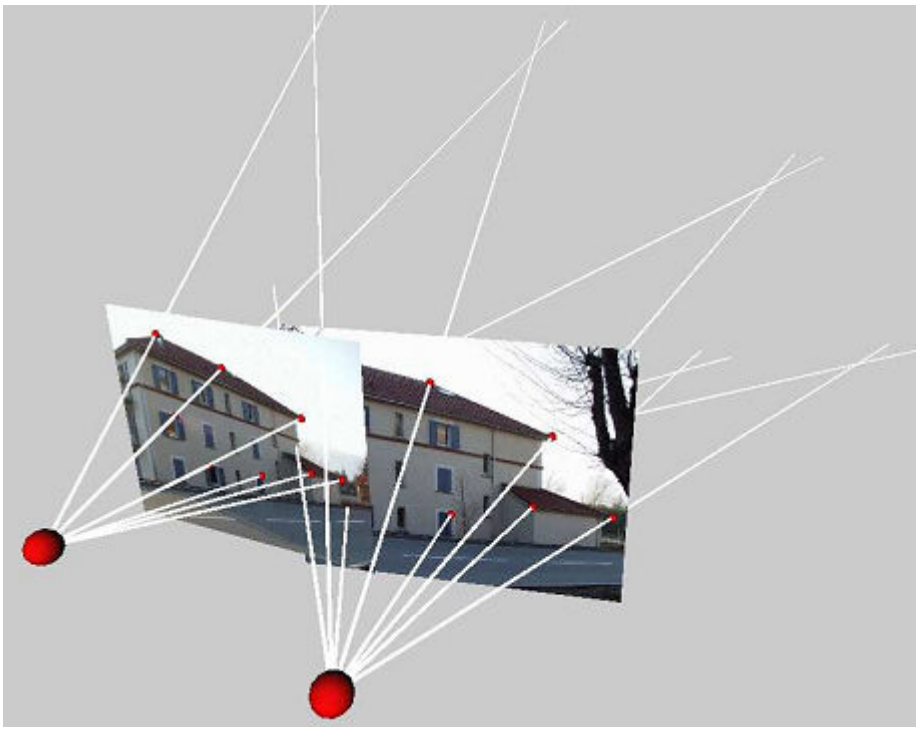


Figure 2.4: Estimation du mouvement : les lignes de vues se correspondants se coupent**.**

II.2 Différentes applications du tracking 3-D

Le tracking 3-D est utilisé dans beaucoup de domaines différents, et nous verrons quelques uns brièvement :

II.2.1 La réalité augmentée

Beaucoup de potentiel a été exploité dans ce domaine, au moins 6 domaines d'applications potentielles de la réalité augmentée ont déjà été identifiés. Le premier domaine est la recherche médicale. Un exemple appartenant à cette catégorie est un projet d'assistance à la biopsie développé à l'UNC (Université de la Caroline du Nord). Ce système de réalité augmentée permet au chirurgien d'observer la position de la tumeur et de son aiguille sous la peau. Un second domaine est la réparation, la maintenance et la fabrication de machinerie complexe. La compagnie Boeing a développé un système d'entraînement de ses mécaniciens, qui ajoute à l'image des étiquettes virtuelles identifiant le nom et la position de certaines pièces dans les

moteurs. Un troisième domaine concerne l'industrie touristique. Un système de R.A. peut effectuer l'ajout de notes et d'information textuelle à l'image d'un site observé. Un exemple est le système *LifeClipper* utilisé à Basel en Suisse pour des visites touristiques. L'utilisateur porte un système doté d'un HMD (Head-Mounted Devices) dont la position et l'orientation sont détectées à partir du système GPS. Ces données sont utilisées pour présenter de l'information relative aux sites touristiques observés par l'utilisateur. Un quatrième domaine est la planification de mouvement d'un robot. L'opération d'un robot manipulateur pouvant être complexe, un système de R.A. permet de manipuler une version virtuelle du robot dans l'environnement réel afin de définir une trajectoire correcte pour son mouvement. Une fois la trajectoire virtuelle bien définie, l'utilisateur peut lancer l'exécution de ce mouvement. Le cinquième domaine est l'industrie du divertissement. Une équipe de l'université de South Australia (UniSA) a développé une version augmentée du jeu vidéo Quake. Les utilisateurs portent un HMD et peuvent ainsi jouer le jeu dans l'environnement réel de leur choix (préalablement modélisé). Le dernier domaine d'application identifié est l'aviation militaire. Depuis plusieurs années, les pilotes peuvent porter des systèmes de visualisation qui superposent à l'image réelle des informations utiles à la navigation ou au combat aérien. Tous ces domaines font de l'incrustation d'objets ou d'images virtuels dans des scènes réelles, et ceci doit se faire en temps réel. Le tracking 3-D en temps réel est la composante critique des applications de la réalité augmentée les plus connues. Les objets virtuels doivent être très bien intégrés dans le monde réel, et la réponse du système doit être courte, laissant ainsi l'illusion nous faire croire que les deux mondes coexistent ensemble.

II.2.2 Asservissement visuel (Visual Servoing)

L'asservissement visuel implique l'utilisation d'une ou plusieurs cameras. Un système de vision par ordinateur contrôle la position d'un engin (robot), tel qu'un bras robotisé. Ceci requiert la détection, le tracking, l'asservissement et la saisie (le grasping). Il touche une grande partie de domaines de vision par ordinateur : la robotique, la cinématique, la dynamique, la commande, et les système à temps réel, et il est utilisé dans une riche variété d'applications telles que le suivi de trajectoire pour les voitures, la navigation pour les plateformes mobiles et la manipulation d'objets

génériques. Des informations extraites du tracking sont requises pour la mesure d'erreur entre la position courante du robot et sa position ciblée ou désirée à partir de sa camera embarquée qui lui sert de capteur visuel.
Par conséquent l'algorithme de tracking doit être précis, rapide et général.

II.2.3 Interface homme machine

Le tracking 3D peut être intégré dans une interface homme machine. Par exemple, il peut être utilisé pour mettre a jour la position d'un objet tenu par une main qui devra servir d'un pointeur 3D. L'utilisateur peut ainsi manipuler des objets virtuels dans un environnement qui lui est familier. La conception se trouvera ainsi simplifiée du fait que l'utilisateur peut exprimer ses idées en gardant les mêmes habitudes de l'environnement réel.

II.3 Modélisation mathématique du tracking 3-D

Le tracking 3-D a pour but de recouvrir d'une manière continue les six degrés de liberté de la caméra relativement à la scène, à savoir : les trois coordonnées cartésiennes définissant sa position X, Y et Z et les trois angles de rotation θ_x, θ_Y et θ_z qui définissent son orientation. Ou bien d'une manière analogue, le déplacement 3-D d'un objet relativement à la caméra.

Dans cette section nous introduirons les outils mathématiques nécessaires à la description du problème. Nous allons donc formuler la représentation de la caméra et le paramétrage de la pose. En particulier, l'orientation de la caméra peut être formulée comme étant une rotation 3-D dans l'espace et nous discuterons ses différentes représentations possibles.

II.3.1 Représentation d'une caméra [16]

Dans cette présente étude nous nous concentrons sur le modèle sténopé d'une caméra standard et ses variantes potentielles qui est approprié pour la plupart des applications du tracking 3-D. Néanmoins, il existe de récentes générations de caméras

appelées caméras omnidirectionnelles, conçues avec des miroirs paraboliques ou hyperboliques permettant ainsi l'obtention d'un champ et angle de vue très large. Cependant, les outils mathématiques requis pour leurs traitements restent très analogues et il est aisé de faire la transposition, c'est pour cela que nous restreignons notre description au modèle présenté en premier.

Dans ce modèle, on considère le centre de projection (le sténopé) comme étant l'origine d'un référentiel appelé *référentiel caméra.* Le plan image est positionné dans ce référentiel à une distance $z = -f$ (distance focale). On notera que l'image ainsi produite est inversée.

Sous ce modèle, un point 3-D du monde réel est projeté sur le plan image en un point qui correspond à l'intersection de ce plan et d'une droite passant par le centre de projection et le point 3-D (le rayon projecteur). Avec les deux triangles semblables ainsi formés (Figure 2.6), on constate que la position (u_i, v_i) dans l'image d'un point 3-D (X,Y,Z) est donnée par $(-f\frac{X}{Z}, -f\frac{Y}{Z})$.

On peut représenter cette relation sous la forme matricielle :

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-fX}{Z} \\ \frac{-fY}{Z} \\ \frac{Z}{Z} \end{bmatrix} = \begin{bmatrix} -fX \\ -fY \\ Z \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad 2.1$$

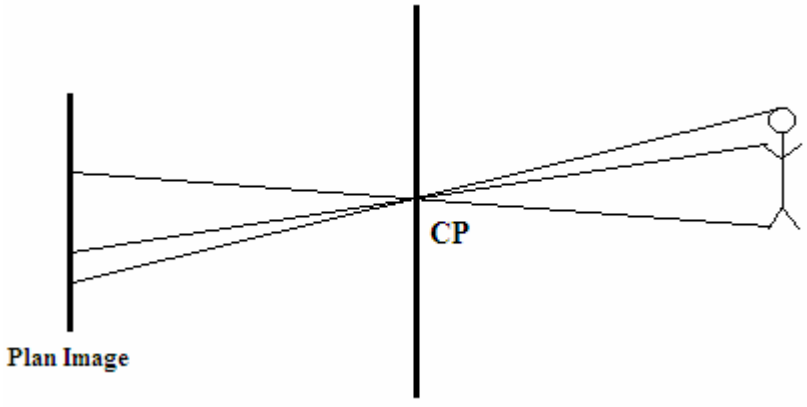


Figure 2.5 : Chaque point de l'image est le résultat d'un seul rayon de lumière passant à travers le sténopé (centre de projection : CP).

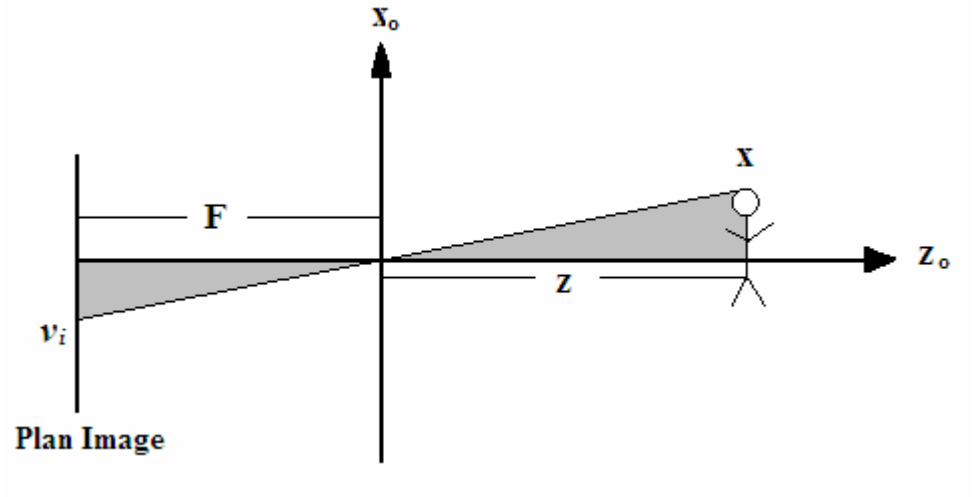


Figure. 2.6 : Coupe dans le plan X-Z d'une projection. Les triangles semblables permettent d'établir la relation Point-Image.

Cependant, le point image est ainsi exprimé en unités métriques dans un référentiel dont l'origine est située à l'intersection du plan image et de l'axe optique, c'est-à-dire au point principal. Pour fins de traitements dans l'image, il est plus pratique d'exprimer les coordonnées images en pixels dans un référentiel qui est situé dans un des coins du plan image. Il faut donc diviser ces coordonnées par la taille d'un pixel selon les axes horizontal *(k)* et vertical (*l*), puis ajouter le décalage (u_0, v_0) entre le point principal et le coin du plan image. On considère parfois un paramètre supplémentaire θ représentant l'angle entre les axes *u* et *v* du plan image. L'expression sous forme matricielle devient alors :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & -\alpha\cos\theta & u_0 \\ 0 & \dfrac{\beta}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad 2.2$$

On appelle cette matrice : *matrice de calibrage de la caméra K* où $\alpha = -\frac{f}{k}$ et $\beta = -\frac{f}{l}$.

Les paramètres *k, l, f,* u_0 et v_0 sont appelés *paramètres intrinsèques* de la caméra. L'angle θ étant toujours très près de 90° et son expression dans la matrice *K* étant sous la

forme de fonctions trigonométriques (e.g. $\sin\theta \sim 1$, $\cos\theta \sim 0$), par conséquent l'expression de la matrice K se retrouve ainsi simplifiée.

Ces paramètres décrivent la structure physique de la caméra et peuvent être fournis par le fabricant : ils sont alors appelés *paramètres nominaux.* En général, le calcul précis de ceux-ci est réalisé par un processus appelé *étalonnage géométrique de caméra.*

- **Modèle de projection perspective [2]**

Les photogrammètres ont recherché à calculer la pose de la caméra et ses paramètres internes à partir d'images bien avant les scientifiques de la vision par ordinateur. Dans ce projet nous essaierons de combiner les techniques d'estimation de la pose avec les extractions des caractéristiques des images plutôt que de s'intéresser à l'estimation uniquement.

Mathématiquement, la formation d'une image peut être définie comme une projection de l'espace 3-D dans le plan de l'image comme l'illustre la (figure 2.7). Les coordonnées euclidiennes d'un point 3-D $\mathbf{M} = [X, Y, Z]^T$ exprimées dans le référentiel du monde (scène) et son point 2-D correspondant $\mathbf{m} = [u, v]^T$ dans l'image sont reliés par l'équation suivante :

$$s\tilde{m} = P\tilde{M} \qquad 2.3$$

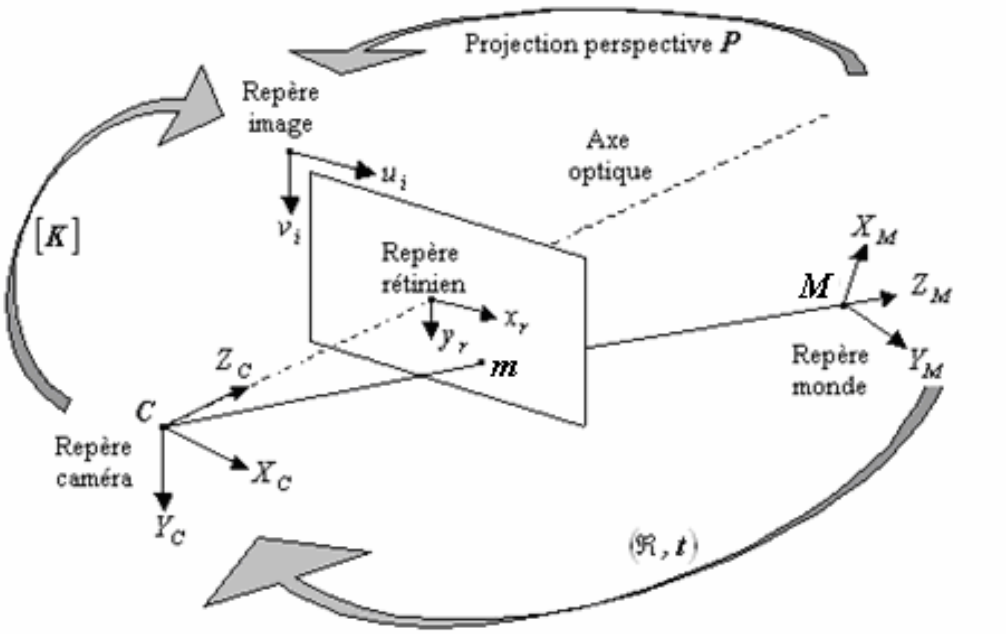


Figure 2.7 : Illustration du modèle de la projection perspective.

Où ***s*** est un facteur d'échelle, $\tilde{m} = [u, v, 1]^T$ et $\tilde{M} = [X, Y, Z, 1]^T$ sont les coordonnés homogènes des points **m**, **M** et **P** (3x4) est une matrice de projection. **P** est définie à un facteur d'échelle, donc ne dépend que de 11 paramètres. Elle est souvent prise comme étant une matrice de projection en perspective car elle décrit le comportement simplifié d'une caméra d'assez bonne qualité.

Une telle matrice de projection peut être décomposée ainsi :

$$\mathbf{P = K[R \mid t]} \qquad 2.4$$

Où :

- **K** est la matrice de calibrage de la camera (3x3) qui dépend des paramètres internes de la caméra comme la distance focale ;
- **[R | t]** est la matrice des paramètres externes de la caméra (3x4) qui correspond à la transformation euclidienne du référentiel de la scène au référentiel de la caméra : **R** représente la matrice de rotation (3x3) et **t** un vecteur de translation. Nous décrirons ces dernières composantes avec plus de détails dans les paragraphes qui vont suivre.

- **Matrice de calibrage d'une caméra**

Comme on l'a déjà vu auparavant, la matrice de calibrage K d'une caméra contient les paramètres intrinsèques de la caméra, appelés aussi paramètres internes.

$$\mathrm{K} = \begin{bmatrix} \alpha_u & \tau & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad 2.5$$

Où : τ désigne une inclinaison, c'est un paramètre non nul uniquement si les axes u et v ne sont pas perpendiculaires ce qui est rare dans les caméra modernes.
Si on suppose que le point principal **c** se situe dans le centre de l'image, qui est souvent une approximation très raisonnable. De la même manière, si on suppose que les pixels sont de forme carrée alors α_u et α_v deviendront égaux.

- **Matrice des paramètres externes**

La matrice **[R|t]** (3x4) des paramètres externes définit l'orientation et la position de la caméra par rapport à la scène. Elle est formée de la matrice de rotation **R** augmentée du vecteur de translation **t** et on se réfère souvent à elle comme étant la pose de la caméra.
Dans les applications de tracking, souvent la matrice de calibrage K est connue. On se focalise à estimer **R** et **t** ou bien d'une manière réciproque à estimer la position et l'orientation des objets dans la scène par rapport à la caméra.
Un point 3-D représenté par un vecteur $\mathbf{M}_w$ dans le repère du monde réel sera représenté par un autre vecteur $\mathbf{M}_c = \mathbf{R}\mathbf{M}_w + \mathbf{t}$ par rapport au repère de la caméra. A l'aide de cette relation on peut facilement récupérer la position du centre de la caméra (centre optique) **C** dans le repère du monde réel.
On a :

$$\mathbf{0} = \mathbf{RC} + \mathbf{t} \rightarrow \mathbf{C} = -\mathbf{R}^{-1}\,\mathbf{t}. \qquad 2.6$$

- **Estimation de la matrice de calibrage**

Dans la plupart des méthodes de tracking 3-D, les paramètres internes sont supposés fixes et connus. Ce qui veut dire que la caméra ne pourra pas zoomer à cause de la difficulté à distinguer entre un changement de la distance focale et une translation de la caméra selon l'axe Z. Ces paramètres peuvent être estimés dans une étape antérieure à celle du tracking lui-même à l'aide d'images test.

On peut trouver différentes techniques pour calibrer une caméra et la plus classique d'entres elles est de placer une grille spéciale dans le champ de vue de la caméra. Souvent on utilise une grille de calibrage 3-D formée de disques noirs sur un fond blanc comme celle qu'on voit sur la figure suivante.

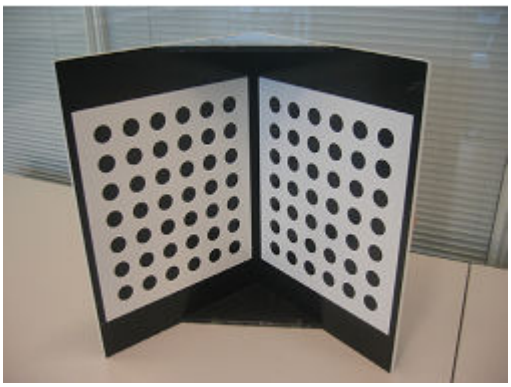


Figure 2.8 : grille de calibrage 3-D utilisée pour estimer la matrice de calibrage K.

Dans cet exemple les coordonnées 3-D des coins des carrés blancs par rapport au coin supérieur gauche de la grille sont connues. Il est relativement facile de détecter ces coins dans l'image et il suffit de faire une correspondance entre les points 3-D et les points 2-D dans l'image pour calculer les paramètres de la matrice de projection.

II.3.2 Paramétrage de la pose d'une caméra [16]

Pour des buts d'optimisation numérique des estimations, la pose de la camera doit être paramétrée d'une façon appropriée. Pour la représentation des translations, aucun problème ne se pose. Cependant, représenter une rotation dans R^3 est beaucoup plus difficile à cerner.

Il est bien connu qu'une rotation dans R^3 contient 3 degrés de liberté (un pour chaque axe) mais pour la représenter, six contraintes non linéaires additionnelles sont nécessaires pour maintenir la matrice de rotation orthonormée: Trois pour maintenir les trois colonnes unitaires et trois autres pour les maintenir orthogonales. Cependant, il est gênant d'utiliser directement les 9 éléments de cette matrice pour des raisons de complexité d'algorithmes d'estimation.

Nous verrons brièvement les différents paramétrages qui se sont avérées efficaces pour le tracking 3-D : Les angles d'Euler, les quaternions et les maps exponentielles (*Exponential Maps*). Toutes les trois ont leurs faiblesses mais cette dernière s'est avérée la plus efficace dans notre but. Cependant, si on se restreint à de petites rotations, elles sont toutes équivalentes du fait qu'elles produisent toutes le même premier ordre d'approximation.

II.3.2.1 Les angles d'Euler

Une matrice de rotation R peut toujours être représentée comme le produit des trois matrices correspondant aux rotations autour des trois axes X, Y et Z. Il existe plusieurs conventions sur la façon dont ces matrices sont combinées. Si on prend l'exemple d'une rotation autour des axes X, Y et Z avec des angles de α, β et γ respectivement produit la matrice de rotation R suivante :

$$R = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\wp & -\sin\wp \\ 0 & \sin\wp & \cos\wp \end{bmatrix} \qquad 2.7$$

Ainsi, pour une matrice de rotation donnée, on peut facilement extraire les angles de rotation avec l'opération inverse en identifiant ses coefficients et leur expression analytique.

Les angles d'Euler peuvent effectuer un travail satisfaisant pour un large domaine de rotation de camera. Cependant, ils ont un bien connu défaut : lorsque deux des trois axes s'alignent, l'un n'a plus d'effet. Ce problème est connu sous le nom de Gimbal lock. La représentation que nous verrons par la suite est adoptée pour palier à ce problème.

II.3.2.2 Les Quaternions

Une rotation dans l'espace 3-D peut aussi être représentée par un quaternion unitaire, les quaternions sont des nombres hyper complexes qui peuvent être écrits comme une combinaison linéaire a+bi+cj+dk, avec $i^2 = j^2 = k^2 = ijk = -1$. Ils peuvent aussi être interprétés comme un scalaire et un vecteur 3-D $(a, \vec{v})$. Une rotation autour du vecteur unitaire $\vec{w}$ par l'angle θ est représentée par le quaternion unitaire suivant :

$$q = (\cos(\frac{1}{2}\theta), \vec{w}\sin(\frac{1}{2}\theta)) \qquad 2.8$$

Ainsi, pour faire une rotation au point 3-D M, on l'écrit sous forme de quaternion:
$p = (0, M)$ et le point résultant après la rotation sera :

$$p' = q \cdot p \cdot \overline{q} \qquad 2.9$$

Où **.** est la multiplication de quaternions et

$$\overline{q} = (\cos(\frac{1}{2}\theta), -\vec{w}\sin(\frac{1}{2}\theta)) \qquad 2.10$$

est le conjugué de q.

Cette représentation évite le problème du Gimbal lock, mais il reste toujours trois contraintes additionnelles qu'il faut ajouter aux algorithmes d'estimation pour maintenir la norme de q unitaire. Si q est estimé par des techniques d'optimisation

numérique, alors ceci peut être obtenu en rajoutant le terme quadratique k (1 - ||q||²) à la fonction « objectif », où k est un poids suffisamment grand. Ceci tend à augmenter la complexité de l'algorithme, ce qui n'est pas souhaité en général. La prochaine représentation qu'on verra a été apportée pour éviter à la fois le problème du gimbal lock et aussi l'ajout des contraintes additionnelles.

II.3.2.3 La map exponentielle (*Exponential map*)

Contrairement aux quaternions, la map exponentielle requiert seulement 3 paramètres pour décrire une rotation. Elle ne souffre pas du problème du Gimbal lock et ne requiert pas de contraintes additionnelles. Ses singularités apparaissent dans des régions d'espace de paramètres spécifiques, qui peuvent facilement être évitées.

Soit $\vec{W} = [W_X, W_Y, W_Z]^T$ un vecteur 3-D et θ = || W || sa norme. Une rotation angulaire de θ autour d'un axe de direction $\vec{W}$ peut être représentée par la série infinie suivante :

$$\exp(\Omega) = I + \Omega + \frac{1}{2!}\Omega^2 + \frac{1}{3!}\Omega^3 + \ldots \qquad 2.11$$

Où Ω est la matrice anti-symétrique suivante :

$$\Omega = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} \qquad 2.12$$

Ceci est la représentation en map exponentielle qui doit son nom au fait que l'équation précédente est de la même forme que le développement en série de l'exponentielle. Elle peut être approximée en utilisant la formule de Rodrigues suivante :

$$R(\Omega) = \exp(\Omega) = I + \sin\theta\hat{\Omega} + (1-\cos\theta)\hat{\Omega}^2 \qquad 2.13$$

Où Ω est la matrice anti-symétrique correspondante au vecteur unitaire $\vec{w}/\|\vec{w}\|$.

Du premier coup d'œil, il est facile de remarquer que la formule précédente présente des limitations dans le cas où $\theta = \|\vec{w}\|$ tend vers zéro. Mais ceci peut facilement être évité puisque la formule peut être réécrite comme :

$$R(\Omega) = \exp(\Omega) = I + \frac{\sin\theta}{\theta}\Omega + \frac{(1-\cos\theta)}{\theta^2}\Omega^2 \qquad 2.14$$

Et ceci en remplaçant les deux termes $\frac{\sin\theta}{\theta}$ et $\frac{(1-\cos\theta)}{\theta^2}$ par les deux premiers termes de leurs développements de Taylor.

Cette représentation présente des limitations lorsque $\|\vec{w}\| = 2n\pi$ avec n ≥ 1. Dans ce cas, il n'y a effectivement aucune rotation quelque soit la direction de $\vec{w}$. En pratique, on peut facilement éviter ceci lorsque l'angle de rotation devient plus large que π, en remplaçant $\vec{w}$ par le terme $(1 - \frac{2\pi}{\|\vec{w}\|})\vec{w}$, qui représente la même rotation lorsque la norme est inférieure à π. En d'autres termes, une rotation de θ radians par $\vec{V}$ est équivalente à une rotation de $(2\pi-\theta)$ radians par $-\vec{V}$.

Pour résumer, la représentation en map exponentielle présente une rotation par un vecteur 3-D qui donne au même temps son axe et l'amplitude de son angle. Elle évite le problème du gimbal lock des angles d'Euler, et ne requiert pas de contraintes additionnelles pour préserver la norme du quaternion. Ainsi cette représentation est la plus appropriée pour notre but.

II.3.2.4 Linéarisation des petites rotations

Dans le tracking 3-D, le mouvement de la camera entre deux images successives peut souvent être petit. Dans ce cas, il est approprié d'utiliser le premier ordre d'approximation de la rotation qui linéarise le problème d'estimation de la pose et simplifie les calculs.

Soit M' la position 3-D d'un point M après une rotation **R** de l'origine par un petit angle. Toutes les représentations qu'on a vues précédemment produisent toutes le même premier ordre d'approximation :

$$M' = RM \approx (I + \Omega)M = M + \Omega M \qquad 2.15$$

Où Ω est la matrice anti-symétrique qu'on a vue précédemment.

Discussion

Ce chapitre a donné un aperçu des contraintes imposées par les caméras et leur géométrie. Le problème est donc posé : Peut-on reconstituer la trajectoire d'une caméra embarquée sur un Robot (ou un bras robotique) en mouvement en exploitant uniquement les images d'une scène rigide ?
L'auto calibrage d'une caméra consiste à estimer ses paramètres intrinsèques et extrinsèques à partir de la séquence d'images d'une scène prise par cette même caméra.

Dans la première partie de ce chapitre nous avons vu les différents problèmes possibles à résoudre dans le tracking 3-D, chose qui nous aidera à bien cerner notre problème qui consiste à calculer la pose à partir d'une séquence d'images prise par une caméra. Nous considérerons une scène dont la géométrie est connue en plaçant une mire (comme un damier par exemple).

Dans la deuxième partie de ce chapitre nous avons étudié le modèle mathématique d'une caméra et la meilleure projection la représentant (perspective). Vu qu'on aura à reproduire une caméra virtuelle dans notre application (chapitre 5) pour visualiser les résultats obtenus, cette partie est donc d'un grand intérêt pour bien paramétrer notre caméra.

Chapitre III : détection de points d'intérêt et leurs mise en correspondance.

Préambule

La détection de points d'intérêts est au même titre que la détection de contours, une étape préliminaire à de nombreux processus de vision par ordinateur. Par exemple, si deux images successives prises d'une séquence vidéo peuvent être reliées alors il est possible d'extraire des informations sur les profondeurs des objets dans l'environnement ainsi que le mouvement de la caméra.

Il est fastidieux de comparer deux images pixel par pixel car cela engendrerait un grand temps de calcul Intuitivement, on peut considérer uniquement les régions qui présentent une certaine similitude. De telles primitives sont appelées points d'intérêts (ou coins) et sont détectées par des algorithmes adéquats .L'utilité de détecter ces points est une étape clé dans plusieurs applications de traitement d'images et de vision artificielle. Parmi ces applications on peut citer :

- La robotique mobile.
- Le suivi de mouvement (objets ou camera).
- L'imagerie médicale (image registration).
- La reconstitution d'images panoramiques.
- La détection et reconnaissance de formes et d'objets 2-D et 3-D.

III.1 Avantages des points d'intérêts

Les principaux avantages des points d'intérêt par rapport aux contours sont :

- La fiabilité des sources d'informations
- La robustesse aux occultations (soit occulté complètement, soit visible),
- Pas d'opérations de chaînage contrairement aux contours,
- Présents dans une grande majorité d'images.

Plusieurs détecteurs de points ont été mis en œuvre avec différentes approches permettant de déceler les points les plus significatifs qui peuvent correspondre à l'un des cas suivants : Jonction en L, Y, T, X, ou en flèche comme illustré dans la figure 3.1.

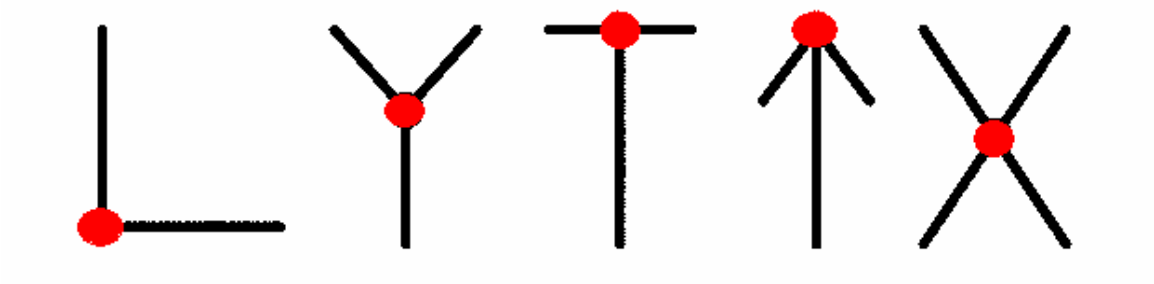


Figure 3.1: Jonction en L-jonction en Y-jonction en T-jonction en flèche-jonction en X

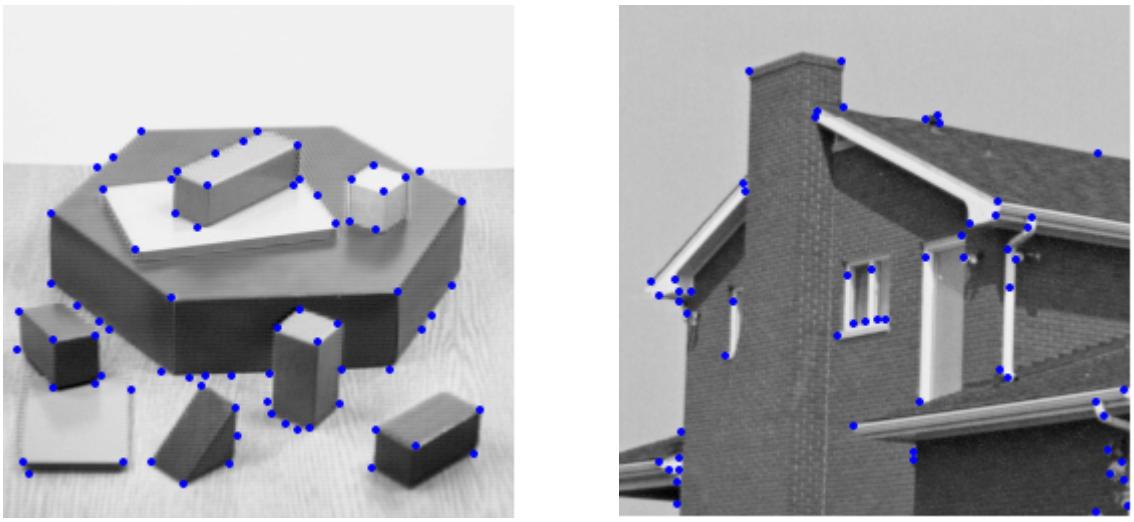


Figure 3.2: exemple de détection de points et de différentes jonctions.

III.2 Approches utilisées

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

2. Approche intensité: l'idée est cette fois-ci de considérer la fonction d'intensité dans les images pour en extraire directement les points de discontinuités.

3. Approches à base de modèles : les points d'intérêts sont identifiés dans l'image parmi les correspondances de la fonction d'intensité avec un modèle théorique de cette fonction.

Les approches de la deuxième catégorie sont celles utilisées généralement. Les raisons sont : indépendance vis à vis de la détection de contours (stabilité), indépendance vis à vis du type de points d'intérêts (méthodes plus générales).

III.3 Algorithmes de détection de points

De nombreux algorithmes ont été conçus pour répondre à ce besoin et les plus utilisés sont présentés dans l'ordre chronologique suivant :

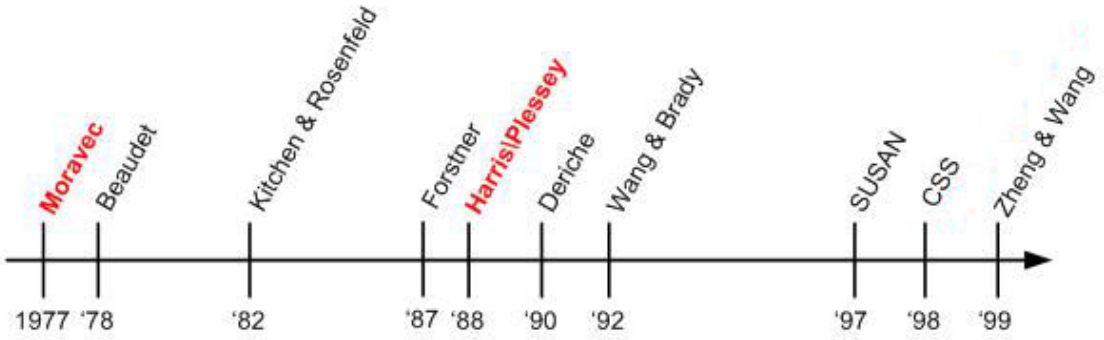


Figure 3.3: présentation des détecteurs selon la chronologie.

Dans notre projet on étudiera le détecteur de Moravec et celui de Harris\Stephen en raison de leur simplicité et de leur temps de calcul raisonnable.

III.3.1 Détecteur de Moravec [17]

Développé en 1977 par Hans P. Moravec en vu de ses recherches sur le Stanford Cart. Il a introduit le concept des points d'intérêts comme étant des régions distinctes des images et a conclu qu'ils pouvaient être utilisés comme un moyen pour faire correspondre ces mêmes régions dans une séquence d'images.

L'idée du détecteur de Moravec 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. Plus précisément on considère la fonction :

$$E(x,y)=\sum_{u,v}\omega(u,v)\left|I(x+u,y+u)-I(u,v)\right|^2 \qquad 3.1$$

Où :

- ω spécifie la fenêtre/voisinage considérée (valeur 1 à l'intérieur de la fenêtre et 0 à l'extérieur);
- I (u, v) est l'intensité au pixel (u, v);
- E (x, y) représente la moyenne du changement d'intensité lorsque la fenêtre est déplacée de (x, y).

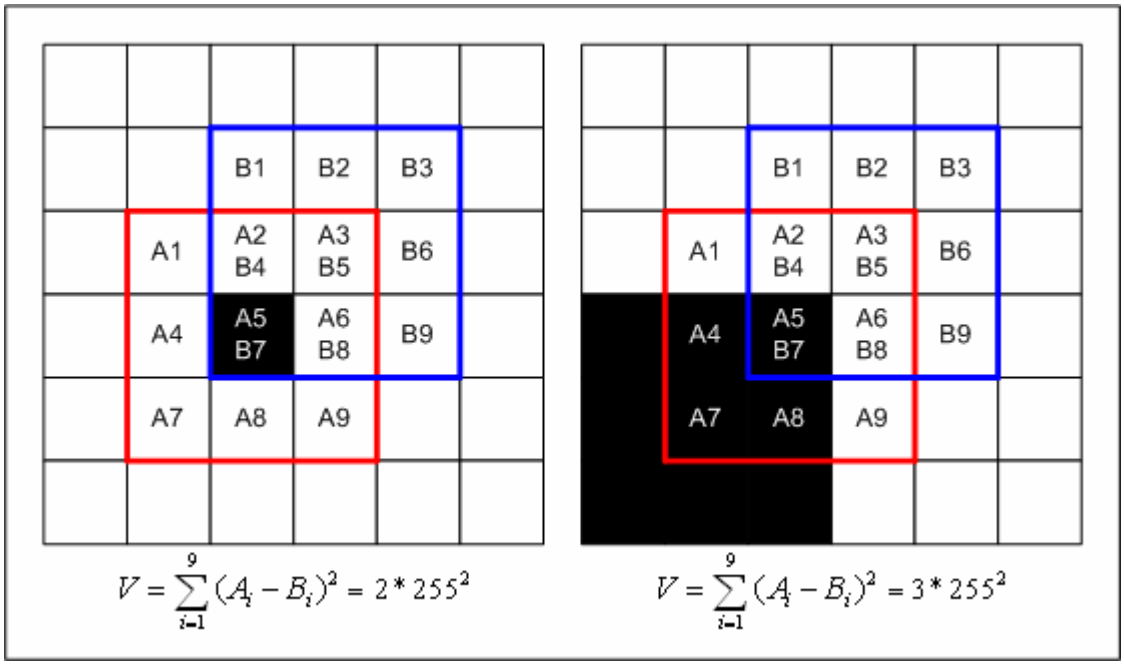


Figure 3.4: exemple de calcul de variation d'intensité V.

Comment peut-on mesurer la variation d'intensité d'une région centrée en un pixel d'une image ? On propose de calculer cette variation entre la fenêtre (en rouge) centrée en A5 typiquement de taille (3x3, 5x5, 7x7 pixels) avec chacune des fenêtres décalées (en bleu) d'un pixel dans les huit directions principales (horizontales, verticales et les quatre diagonales). Pour un décalage donné, cette variation n'est autre que la somme des carrés des différences d'intensités entres pixels correspondants.

Pour comprendre comment l'opérateur de Moravec détecte les coins, considérons la figure suivante qui nous illustre les différents cas de figure.

Dans la position A, la fenêtre est située à l'intérieur d'un objet (ou arrière-plan) où l'intensité est relativement constante, donc un décalage de la fenêtre dans n'importe quelle direction produira une faible variation d'intensité. Pour une fenêtre contenant un contour, comme dans la position B, un décalage le long du contour donnera une faible

variation d'intensité alors qu'un décalage perpendiculaire au contour produira une forte variation. Enfin, les position C et D qui correspondent respectivement à un coin et un pixel isolé donneront une forte variation d'intensité dans toutes les directions de décalage.

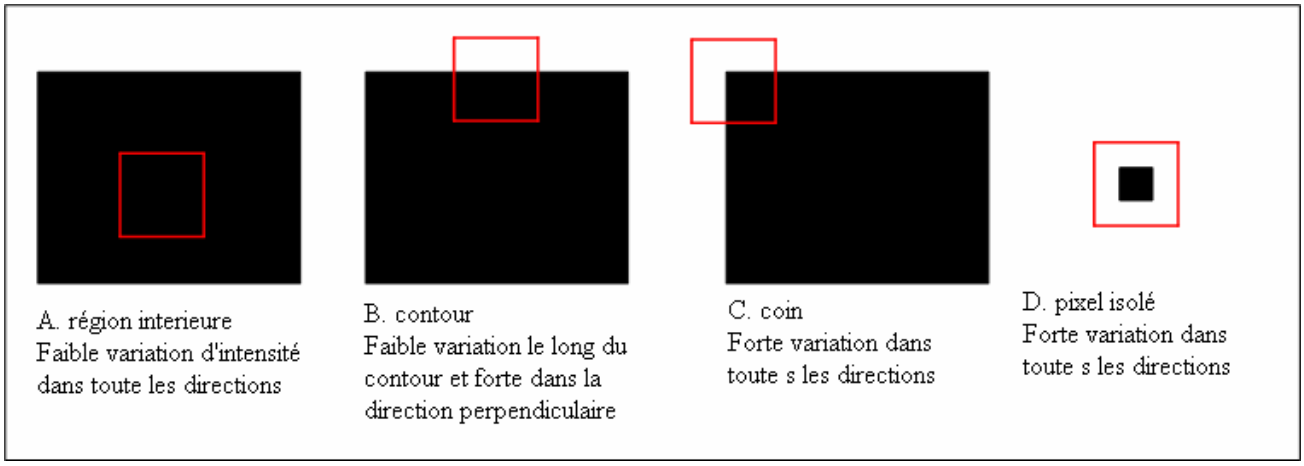


Figure 3.5 : différents cas pour l'opérateur de Moravec.

III.3.2 Détecteur de Harris et Stephen (1988) [18]

Le détecteur de Moravec fonctionne dans un contexte limité. Il souffre en effet de nombreuses limitations. Harris et Stephen ont identifié certaines limitations et en les corrigeant, en ont déduit un détecteur de coins très populaire : le détecteur de Harris. Les limitations du détecteur de Moravec prises en compte sont :

III.3.2.1 La réponse est anisotropique

Le détecteur de Moravec a une réponse anisotropique en raison de son caractère discret. Pour palier à ce problème une fonction est nécessaire pour mesurer la variation d'intensité dans toutes les directions. Harris et Stephen ont formulé une expression analytique de l'opérateur Moravec telle une dérivée d'une fonction. Ici, une approche beaucoup plus intuitive mais moins rigoureuse du point de vu mathématique est considérée pour arriver à cette fonction qui n'est autre que le gradient au voisinage

du pixel considéré. Ceci pouvant être calculé en appliquant une convolution spécifique à l'aide d'un masque ad hoc.

- **Le produit de convolution**

Un filtre linéaire est caractérisé par un produit de convolution entre le signal d'entrée et de sortie du filtre. Ce produit *P* est fonction d'un masque de convolution composé d'un ensemble de coefficients C_{ij} avec $i=i_0, i_1 \ldots i_k$ et $j=j_0, j_1 \ldots j_k$ tel que :

$$P(I(x_0, y_0)) = \sum_{i,j} C_{ij}.I(x_0 + i, y_0 + j) \qquad 3.2$$

Dans le cas d'un masque 3*3 appliqué sur un bloc image, les opérations effectuées sont les suivantes (ces masques sont appliqués à toute l'image par balayage):

Masque 3*3 : $\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$; Voisinage 3*3 d'un pixel : $\begin{pmatrix} g_1 & g_2 & g_3 \\ g_4 & g_5 & g_1 \\ g_7 & g_8 & g_9 \end{pmatrix}$

Convolution obtenue :

$g_s = a.g_1 + b.g_2 + c.g_3 + d.g_4 + e.g_5 + f.g_6 + g.g_7 + h.g_8 + i.g_9$ 3.3

Remarque

Les convolutions peuvent être effectuées avec des masques d'ordres différents, 3*3, 1*3, 3*1, …… n*m. Mais il est possible de definir tous les masques de dimensions inferieures ou egales à n*m avec un masque n*m, par exemple :

Masque 1*3 : $\begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$

Masque 3*3 correspondant : $\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$

Masque 5*5 correspondant : $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Notre gradient est approximé par convolution des masques illustrés dans la figure suivante :

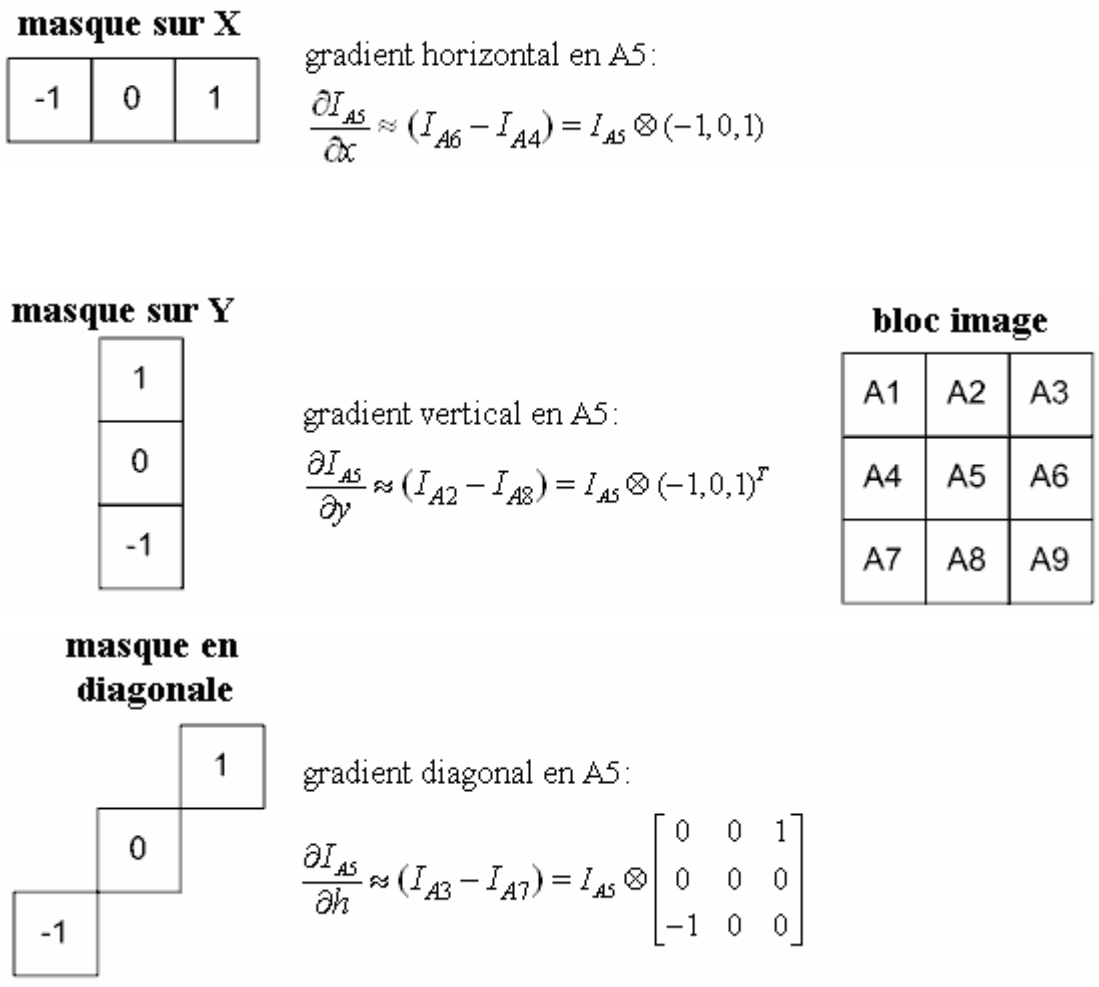


Figure 3.6: Approximation du gradient par convolution avec des masques.

- **Application du gradient pour le calcul de la variation d'intensité**

La variation d'intensité peut être écrite sous forme du gradient de l'image. Pour un décalage donné (u, v), la variation peut s'écrire comme suit :

$$V_{u,v}(x,y) = \sum_{\forall idans\_la\_fenetre(x,y)} \left(u\frac{\partial I_i}{\partial x} + v\frac{\partial I_i}{\partial y} \right)^2 \qquad 3.4$$

Pour un décalage donné (u, v)= (0, 1) ou (u, v)= (1, 0) nous avons l'approximation de la variation d'intensité selon la figure qui suit. Les résultats obtenus en effectuant l'expansion analytique de l'opérateur Moravec sont identiques à l'équation 3.1 excepté les termes représentant les propriétés d'ordre supérieur que Harris et Stephen ont ignoré.

direction X

A1	A2 B1	A3 B2	B3
A4	A5 B4	A6 B5	B6
A7	A8 B7	A9 B8	B9

Variation d'intensité sur X (Moravec):

$$V_x = \sum_{i=1}^{9} (A_i - B_i)^2 = \sum_{i=1}^{9} (B_i - A_i)^2 \approx \sum_{i=1}^{9} \left(\frac{\partial I_i}{\partial x} \right)^2$$

Où: $\frac{\partial I_i}{\partial x} \equiv I_i \otimes (-1,0,1) \approx B_i - A_i$

direction Y

B1	B2	B3
A1 B4	A2 B5	A3 B6
A4 B7	A5 B8	A6 B9
A7	A8	A9

Variation d'intensité sur Y (Moravec):

$$V_y = \sum_{i=1}^{9} (A_i - B_i)^2 = \sum_{i=1}^{9} (B_i - A_i)^2 \approx \sum_{i=1}^{9} \left(\frac{\partial I_i}{\partial y} \right)^2$$

Où: $\frac{\partial I_i}{\partial y} \equiv I_i \otimes (-1,0,1)^T \approx B_i - A_i$

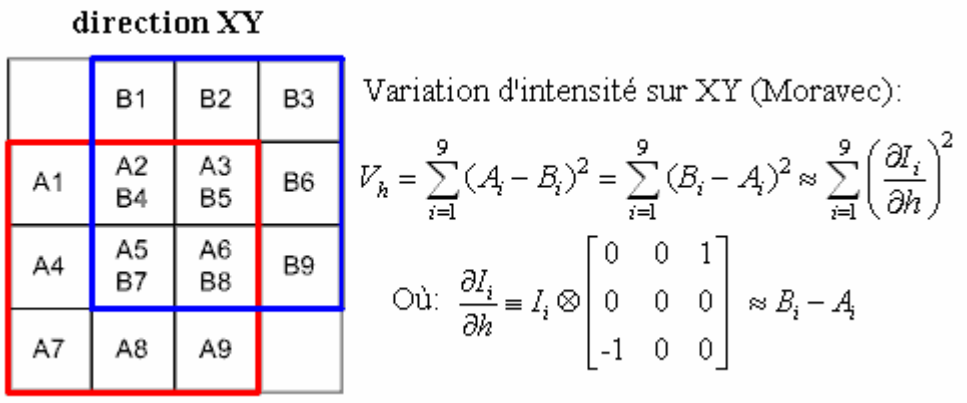


Figure 3.7 : calcul de la variation de l'intensité.

Notez bien que l'équation 3.1 n'est pas vraiment égale à la variation de l'intensité calculée par l'opérateur Moravec. L'avantage de cette nouvelle méthode est qu'elle permet de mesurer la variation d'intensité dans n'importe quelle direction en choisissant u et v de manière appropriée. Néanmoins, cette méthode reste toujours anisotropique car cette estimation est basée sur les gradients horizontal et vertical mais à un degré moindre.

III.3.2.2 La réponse est bruitée

Vu que la fenêtre considérée par l'opérateur Moravec est carrée et binaire, l'estimation de la variation d'intensité est alors bruitée. Avoir une fenêtre carrée produira une distance euclidienne différente selon des directions choisies. Comme le montre la figure suivante.

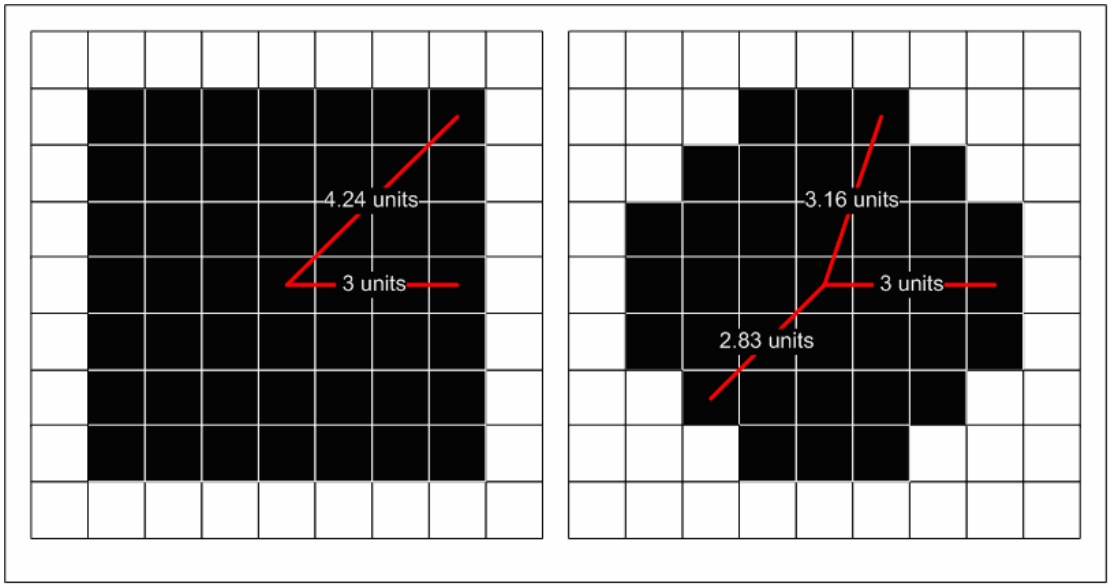


Figure 3.8 : Différence de variation de la distance euclidienne entre une fenêtre carrée et une fenêtre circulaire (1,24 pour la première et 0,17 pour la seconde).

Ayant un comportement binaire la méthode affectera les même poids à l'ensemble des pixels sis à l'intérieur de la fenêtre considérée sans prendre en compte leurs distances par rapport au centre. Ces deux problèmes sont résolus en utilisant une fenêtre gaussienne qui attribut un poids à chaque pixel en fonction de la distance qui le sépare du centre.

La mesure de la variation d'intensité peut être illustrée comme suit pour un décalage horizontal et une fenêtre gaussienne 3x3 :

Fenetre gaussienne

w1 .04	w2 .12	w3 .04
w4 .12	w5 .36	w6 .12
w7 .04	w8 .12	w9 .04

$$V_x = \sum_{i=1}^{9} w_i (A_i - B_i)^2 = \sum_{i=1}^{9} w_i (B_i - A_i)^2 \approx \sum_{i=1}^{9} w_i \left(\frac{\partial I_i}{\partial x} \right)^2$$

$$\text{Où} \quad : \frac{\partial I_i}{\partial x} \equiv I_i \otimes (-1, 0, 1) \approx B_i - A_i$$

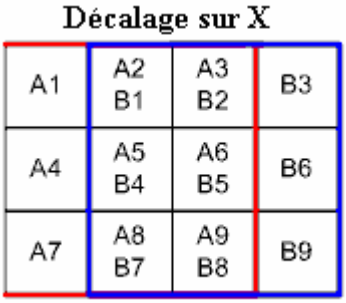


Figure 3.9 : illustration d'une fenêtre gaussienne et un décalage sur X.

Cette variation peut être calculée de la façon suivante :

$$V_{u,v}(x,y) = \sum_{\forall idans\_la\_fenetre\_centré\_en(x,y)} w_i \left(u\frac{\partial I_i}{\partial x} + v\frac{\partial I_i}{\partial y} \right)^2 \qquad 3.5$$

III.3.2.3 Large réponse aux contours

L'inconvénient avec l'opérateur de Moravec est qu'il répond de manière trop forte aux contours, ceci étant dû aux bruits et à la pixellisation. Harris et Stephen ont résolu ce problème en reformulant la manière de considérer si un pixel peut être un coin ou non. Ils considèrent donc la variation d'intensité dans toutes les directions.

L'équation (3.4) peut être réécrite de la manière suivante :

$$V_{u,v}(x,y) = \sum_{\forall idans\_la\_fenetre\_centré\_en(x,y)} w_i \left(u\frac{\partial I_i}{\partial x} + v\frac{\partial I_i}{\partial y} \right)^2 \qquad 3.6$$

$$= \sum_{\forall idans\_la\_fenetre\_centré\_en(x,y)} w_i \left(u^2\frac{\partial I_i{}^2}{\partial x} + 2uv\frac{\partial I_i}{\partial x}\frac{\partial I_i}{\partial y} + v^2\frac{\partial I_i{}^2}{\partial y} \right) \qquad 3.7$$

$$= Au^2 + 2Cuv + Bv^2 \qquad 3.8$$

Où : $A = \left(\frac{\partial I}{\partial x}\right)^2 \otimes w$, $B = \left(\frac{\partial I}{\partial y}\right)^2 \otimes w$, $C = \left(\frac{\partial I}{\partial x}\frac{\partial I}{\partial y}\right) \otimes w$

Et $\otimes$ est le produit de convolution.

Puis on obtient la forme matricielle suivante :

$$V_{u,v}(x,y) = Au^2 + 2Cuv + Bv^2 \quad 3.9$$

$$= \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \quad 3.10$$

Où $M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$

Cette forme matricielle est intéressante du fait qu'elle contient tous les opérateurs différentiels décrivant une surface d'image centrée en un pixel (x, y). Les valeurs propres de M sont proportionnelles aux principales courbures de la surface.

Considérant les quatre cas de la position de la fenêtre vus dans l'opérateur Moravec (figure 3.5).

La position A illustre le cas où la fenêtre est à l'intérieur d'un objet ou d'un arrière plan, la variation d'intensité est supposée faible en raison de l'absence de courbures et ainsi les valeurs propres prendront des valeurs faibles. Pour une fenêtre contenant un contour comme dans la position B, l'une des valeurs propres prendra une forte valeur alors que l'autre sera de faible valeur. Enfin, dans les positions C et D correspondant respectivement à un coin et à un pixel isolé, les valeurs propres prendront ainsi de fortes valeurs.

Soient λ_1 et λ_2 les valeurs propres de la matrice M. L'analyse précédente indique que le plan formé par λ_1 et λ_2 se divise en trois régions distinctes comme le montre la figure suivante :

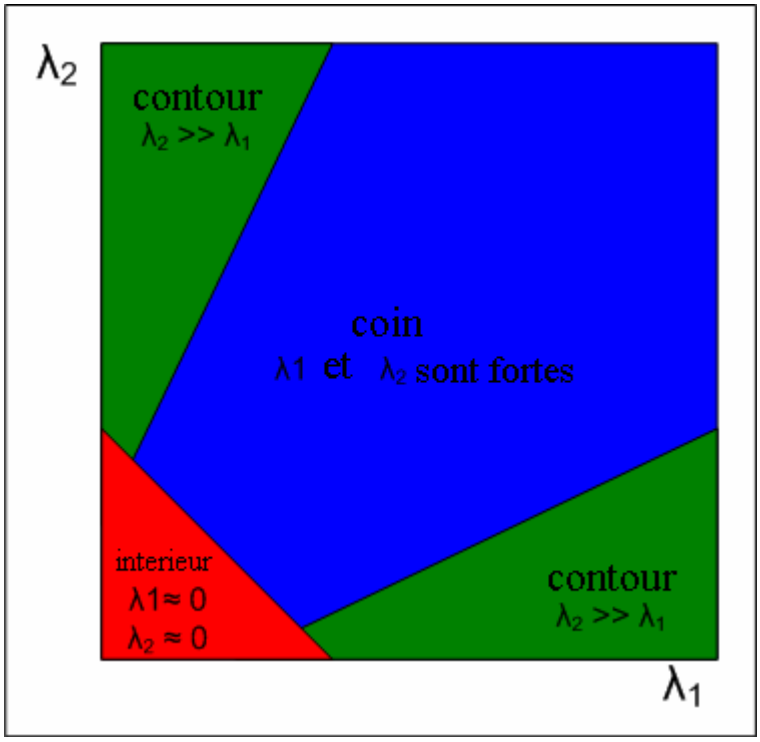


Figure 3.10 : les différentes régions formées par les valeurs propres.

Figure 3.11 : exemple de détection de points d'intérêt en utilisant Harris (k=0.04) dans une image réelle.

III.4 Mise en correspondance par corrélation [19]

Le problème essentiel à résoudre est le problème dit de mise en correspondance, qui consiste à trouver les points qui correspondent entre les images des deux poses. La résolution de ce problème a été l'objet de nombreuses recherches dans le monde entier, et les différentes approches se distinguent essentiellement par les contraintes imposées sur les objets présents dans la scène (segments, courbes, objets plans ou lisses, contours d'occultations …) et par les applications proposées (robotique mobile, photogrammétrie, réalité virtuelle …).

Les méthodes de mise en correspondance peuvent pour la plupart être séparées en deux classes : celles à base de primitives (*Feature Based*) et les méthodes surfaciques.

III.4.1 Méthodes à base de primitives

Beaucoup de domaines de la vision par ordinateur, qu'il s'agisse de la stéréoscopie, de la reconnaissance d'objets, ou du suivi d'objet (tracking), proposent des approches à base de primitives. La définition d'une primitive est arbitraire et la seule généralisation qu'on puisse en faire est que c'est en quelque sorte une représentation fidèle et utile d'une image. Les primitives ont généralement les propriétés suivantes : unicité, répétitivité et signification physique. Dans le contexte de la stéréoscopie, le but des approches par primitives est d'obtenir des correspondances fiables, même en présence d'une certaine quantité de bruit dans les images.

La plupart des approches par primitives se sont intéressées aux contours et aux coins. Les contours sont obtenus par un détecteur de contours de type CANNY, puis chaînés pour obtenir des chaînes de contours, et parfois des primitives de plus haut niveau sont extraites de ces chaînes de contours (segments, polygones, courbes splines). Certaines méthodes utilisent d'autres types de primitives, comme des régions ou des structures topologiques. L'avantage de ces méthodes est qu'à la fois elles permettent plus de flexibilité dans le choix des algorithmes (relaxation, programmation dynamique, prédiction-verification, …), elles sont beaucoup plus rapides grâce à la réduction de l'information contenue dans les images, et elles permettent d'obtenir une précision égale à celle des primitives des images. Cependant, elles sont fortement limitées par le

nombre de ces primitives présentes dans les images, et leurs résultats sont des données 3-D éparses. En conséquence, on assiste depuis quelques années à un retour vers les méthodes surfaciques, qui donnent des résultats denses, beaucoup plus intéressants dans de nombreuses applications.

III.4.2 Méthodes surfaciques

Les méthodes surfaciques sont nées de la constatation que deux images sont localement semblables autour des points se correspondant, puisque localement la projection de la surface 3-D associée aux voisinages de ces points est similaires, d'où la dénomination surfacique. Ces méthodes consistent à fixer d'abord un point d'intérêt dans une image, puis, à l'aide d'une mesure de corrélation de type distance euclidienne ou produit scalaire, à chercher un point dans l'autre image dont le voisinage soit semblable à celui du point de référence.

La mise en correspondance surfacique a l'avantage sur les méthodes à base de primitives de fournir une représentation dense de la scène observée : en effet, on peut chercher un correspondant pour tous les points de l'image de référence, même si on ne les trouve pas nécessairement. Par contre, pour que les voisinages des points se correspondant aient des fonctions d'intensité similaire, il est théoriquement nécessaire que les surfaces observées satisfassent les contraintes photométriques et géométrique suivantes :

- **Contrainte lambertienne :** les surface doivent en chaque point être lambertiennes, c'est-à-dire que l'intensité de la projection dans chaque image d'un point 3D doit être indépendante du point de vue (en particulier, il ne doit pas y avoir de réflexions spéculaires sur les surfaces observées).
- **Contrainte fronto-parallèle :** les surfaces doivent être fronto-parralèlles, c'est-à-dire parallèles aux plans rétiniens des deux poses de la camera.
- **Contrainte de continuité :** la plupart des méthodes font l'hypothèse que les surfaces sont au moins localement continues. Aucune méthode de mise en correspondance surfacique à ce jour n'est capable d'apparier de manière satisfaisante et de reconstruire des objets comportant une texture 3-D complexe comme des buissons ou des arbres.

Dans la pratique, ces contraintes sont plutôt surmontables. En effet, la contrainte lambertienne peut devenir localement lambertienne si on normalise localement les intensités, comme on le verra plus bas, ou si plutôt que d'appliquer la méthode aux images brutes on l'applique à des images filtrées. De cette manière, l'algorithme cherchera à mettre en correspondance les variations locales de l'intensité, et non plus l'intensité elle-même.

De même, la contrainte fronto-parallèle n'a pas besoin d'être strictement vérifiée, et même des surfaces légèrement inclinées par rapport au plan rétinien de la camera pourront être mises en correspondance.

Les méthodes surfaciques ont peu évolué depuis l'état des lieux dressé par KOSCHAN en 1993 [20]. Globalement, les méthodes se sont stabilisées et, à quelques exceptions près, les nouveautés ont été plutôt techniques (parallélismes, réalisations et implantations matérielles, temps réel) qu'algorithmiques.

III.4.3 Méthodes classiques

Le problème de la mise en correspondance est simplifié en supposant que les surfaces sont localement lambertiennes et quasi fronto-parallèles. Le principe du calcul de la carte de disparité est alors, pour chaque point de la première image :

1. calculer la corrélation entre une fenêtre l x h de l'image d'intensité centrée en ce point et une fenêtre de même taille centrée en chaque point de l'autre image susceptible de lui correspondre.
2. choisir comme correspondant de ce point celui qui maximise la mesure de corrélation, et en déduire la disparité.

On en déduit alors une carte dense de disparité pour toute l'image, qui peut s'écrire sous forme d'une fonction :

$$d : [0, L] \text{ x } [0, H] \Rightarrow \mathrm{R}$$
$$(u_1, v_1) \Rightarrow d(u_1, v_1)$$

Cette fonction est définie partout sauf en quelques régions éparses où aucune correspondance n'a été trouvée, parce qu'il y a une occultation, une discontinuité, ou que la surface ne satisfait pas aux contraintes liées à la méthode. Avant de décrire

l'algorithme proprement dit, il est nécessaire de savoir comment on choisit l'intervalle de disparité admissible (c'est-à-dire quels points de la deuxième image sont susceptibles de correspondre à un point de la première image), et quelle mesure de critère de corrélation utiliser pour mesurer la ressemblance de deux fenêtres des images.

III.4.3.1 Choix de l'intervalle de disparité

Il est important de savoir restreindre l'espace de recherche des correspondances dans la deuxième image, et ce pour deux raisons. La première est qu'évidemment, plus le nombre de candidats à la correspondance est grand, plus le temps de calcul sera long. La seconde est que plus il y a de candidats, plus on a de chances de faire un faux appariement, c'est-à-dire d'apparier deux points des images qui ne correspondent pas à un point 3-D physique.

On réduit donc cet espace de recherche en émettant l'hypothèse tout à fait justifiée que la caméra ne se déplace pas trop vite entre deux images consécutives. De ce fait, pour retrouver le correspondant d'un point d'une image donnée, on se suffira de le rechercher dans un rectangle assez large, englobant la fenêtre centrée à ce point d'intérêt, dans la deuxième image.

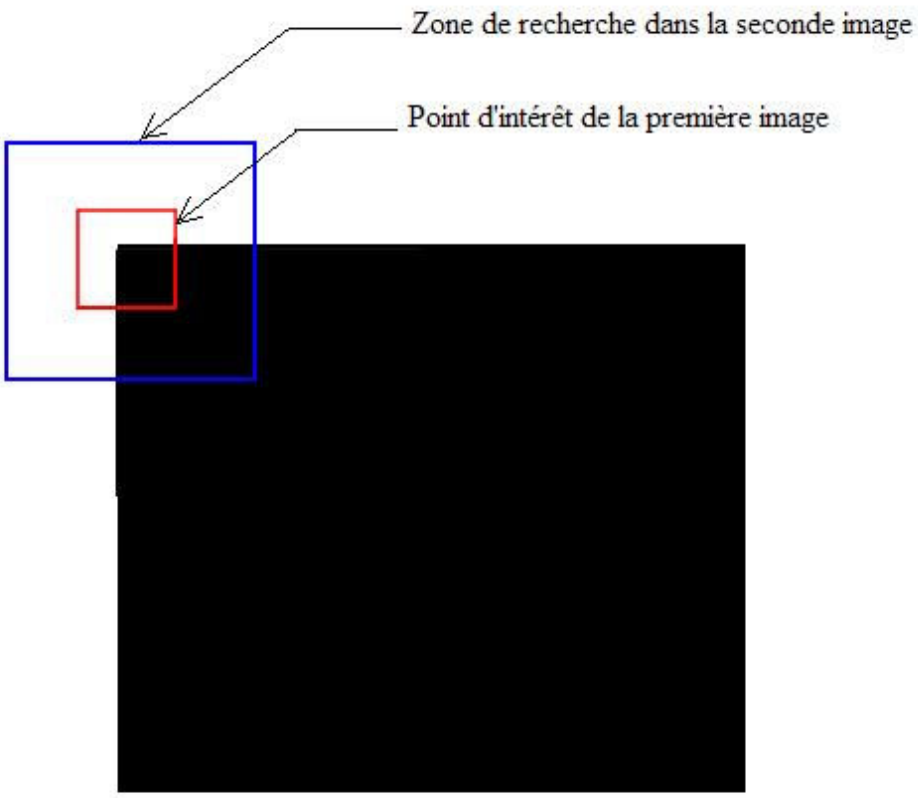


Figure 3.12: illustration de la zone de recherche.

III.4.3.2 Critère de corrélation

Pour obtenir de bons résultats il est nécessaire de choisir un critère de corrélation, c'est-à-dire une mesure de la corrélation entre deux fenêtres des images, qui permette d'obtenir les bonnes correspondances quelle que soit la situation. Pour notre application, les qualités requises pour le critère de corrélation sont :

- permettre de discriminer nettement un bon appariement parmi les points candidats ;
- être robuste aux différentes formes de bruit présentes dans les images (photométriques et géométriques) ;
- permettre la mise en correspondance de surfaces qui ne satisfont que vaguement les contraintes lambertiennes et fronto-parallèle, c'est-à-dire des surfaces légèrement inclinée ou quasi lambertiennes.

La plupart des critères de corrélation dérivent des deux mesures de base : la distance euclidienne entre deux vecteurs formés par les intensités des images (notée

SSD, comme Sum of Squared Differences), et le produit scalaire de ces deux vecteurs (noté CC, comme Cross-Correlation ou corrélation croisée). Il en ressort d'une publication comparant une vingtaine de critères de corrélation [22] que les trois critères de corrélation donnant les meilleurs résultats quelles que soient les conditions de prise de vue sont les critères ZSSD, ZNSSD et ZNCC que nous détaillerons ci-après, et que le critère classique de distance euclidienne reste très performant sous certaines contraintes de prises de vue. Pour chacun de ces critères **c**,

$$c : [0, L] \times [0, H] \times [d_{\min}, d_{\max}] \to \Re$$
$$(x, y, d) \to c_{x,y}(d)$$

La disparité attribuée au point (x, y) est :

$$d(x, y) = \max_{d \in]d\min, d\max[} c_{x,y}(d) \qquad 3.11$$

Si le maximum est atteint en une des bornes (ce qui signifie que l'intervalle de disparité a sans doute été mal choisi), aucune valeur n'est affectée à la disparité.

III.4.3.2.1 Le critère SSD

Le critère SSD (Sum of Squared Differences) correspond à minimiser la somme des différences des intensités sur l'ensemble de la fenêtre de corrélation :

$$c_{x,y}(d) = \sum_{-l \le i \le l ; -h \le j \le h} \left(I_1(x+i, y+j) - I_2(x+d+i, y+j) \right)^2 \qquad 3.12$$

Où I_1 et I_2 sont les fonctions d'intensité dans chacune des deux images. Ce critère est très sensible aux différences d'illumination entre les deux images.

III.4.3.2.2 Le critère ZSSD

Le critère ZSSD (Zero-mean Sum of Squared Differences) consiste à minimiser la somme des différences de l'écart des intensités à leur moyenne calculée sur l'ensemble de la fenêtre de corrélation :

$$c_{x,y}(d) = \sum_{i,j}\left((I_1(x+i,y+j)-\bar{I}_1(x,y))-(I_2(x+d+i,y+j)-\bar{I}_2(x,y))\right)^2 \quad 3.13$$

Où I_1 et I_2 sont les fonctions d'intensité dans chacune des deux images, et où $\bar{I}_1$ et $\bar{I}_2$ sont les moyennes calculées de ces intensités sur chaque fenêtre de corrélation centrée en (x, y). Ce critère permet donc de mettre en correspondance des surfaces dont l'illumination est légèrement différente.

Dans la pratique, on considère que les fonctions $\bar{I}_1$ et $\bar{I}_2$ varient peu sur la fenêtre de corrélation, et on utilise le critère suivant :

$$c_{x,y}(d) = \sum_{i,j}\left((I_1(x+i,y+j)-\bar{I}_1(x+i,y+j)\right)$$

$$-\left(I_2(x+d+i,y+j)-\bar{I}_2(x+d+i,y+j)\right)\right)^2 \quad 3.14$$

$$= \sum_{i,j}\left(I'_1(x+i,y+j)-I'_2(x+d+i,y+j)\right)^2 \quad 3.15$$

On voit que c'est équivalent à filtrer d'abord les images pour obtenir les images I'_1 et I'_2 puis à appliquer la méthode SSD.

III.4.3.2.3 Le critère ZNSSD

Le critère ZNSSD (Zero-mean Normalized Sum of Squared Differences) consiste à minimiser la somme des différences des intensités filtrées par la moyenne sur l'ensemble de la fenêtre de corrélation, normalisée par la variance locale des intensités.

$$c_{x,y}(d) = \frac{-\sum_{i,j}\left(\left(I_1(x+i,y+j)-\bar{I}_1(x,y)\right)-\left(I_2(x+d+i,y+j)-\bar{I}_2(x,y)\right)\right)^2}{\sqrt{\sum_{i,j}\left(I_1(x+i,y+j)-\bar{I}_1(x,y)\right)^2}\cdot\sqrt{\sum_{i,j}\left(I_2(x+d+i,y+j)-\bar{I}_2(x,y)\right)^2}} \quad 3.16$$

On effectue la même simplification que précédemment, et on remarque qu'au plus l'un des deux termes du dénominateur est constant lorsque **d** varie, pour obtenir le critère simplifié :

$$c_{x,y}(d) = \frac{-\sum_{i,j}\left(I_1'(x+i,y+j)-I_2'(x+d+i,y+j)\right)^2}{\sqrt{\sum_{i,j}\bar{I}_2(x+d+i,y+j)^2}} \quad 3.17$$

III.4.3.2.4 Le critère CC

Le critère CC (cross corrélation) est au fait le produit scalaire des deux vecteurs d'intensité :

$$c_{x,y}(d) = \sum_{i,j} I_1(x+i,y+j)I_2(x+d+i,y+j) \quad 3.18$$

Dans la pratique ce critère est inutilisable puisqu'il favorise les zones d'intensité élevée, pas forcement similaire a la fenêtre de corrélation de l'image de référence. On lui préfère largement le critère normalisé ZNCC.

III.4.3.2.5 Le critère ZNCC

Le critère ZNCC (Zero-mean Normalised Cross-Correlation) est en fait le cosinus des vecteurs d'intensité. Plus il est proche de 1, plus les vecteurs sont semblables :

$$c_{x,y}(d) = \frac{\sum_{i,j}\left(I_1(x+i,y+j)-\bar{I}_1(x,y)\right)\left(I_2(x+d+i,y+j)-\bar{I}_2(x,y)\right)}{\sqrt{\sum_{i,j}\left(I_1(x+i,y+j)-\bar{I}_1(x,y)\right)^2}\cdot\sqrt{\sum_{i,j}\left(I_2(x+d+i,y+j)-\bar{I}_2(x+y)\right)^2}} \qquad 3.19$$

Soit en forme simplifiée :

$$c_{x,y}(d) = \frac{\sum_{i,j} I_1'(x+i,y+j)I_2'(x+d+i,y+j)}{\sqrt{\sum_{i,j}\bar{I}_2(x+d+i,y+j)^2}} \qquad 3.20$$

III.4.3.2.6 Comparaison

Dans la pratique on a pu noter que les critères qui donnent les meilleurs résultats sont ZSSD, ZNSSD et ZNCC. ZSSD et ZNSSD donnent des résultats comparables. Mais ZNSSD requiert en plus le calcul des variances d'une des images, et on pourra donc lui préférer si la rapidité d'exécution est un critère déterminant. ZNCC donne des résultats de qualité similaire, mais on a pu noter qu'il met plus facilement en correspondance des zones d'intensité très différentes (du blanc avec du noir), et génère donc un peu plus de faux-appariements.

Discussion

La première partie de ce chapitre a été consacrée à la détection des points d'intérêt et à la présentation des différents outils mathématiques nous permettant de comprendre les algorithmes utilisés. Nous avons en particulier mis en évidence les raisons qui nous ont poussé à choisir les points d'intérêt parmi les autres primitives comme les contours.

Nous avons vu qu'il existe beaucoup de détecteurs chacun ses avantages et ses inconvénients. Notre choix s'est finalement porté sur celui de Moravec et de Harris en raison de leur simplicité d'implémentation et du temps de calcul relativement faible.

Dans la seconde partie, on s'est appuyé sur la corrélation qui est un moyen indispensable pour mesurer le degré d'appariement des points détectés sur des paires d'images au fil de la séquence.

Nous verrons dans le prochain chapitre que la mise en correspondance est fondamentale pour le calcul des paramètres de la caméra (externes et internes) que ça soit dans un environnement connu ou pas et aussi quelque soit les méthodes utilisées.

Chapitre IV : Estimation de la pose de la camera

Préambule

L'estimation de la matrice des paramètres externes à partir d'un flux vidéo est considérablement facilitée par une estimation à priori de la position de la caméra. Nous présenterons quelques approches d'estimation de ces paramètres sans aucune connaissance à priori de la position de caméra, mais en donnant quelques correspondances entre les points 3-D dans le système de coordonnées du monde est leurs projections sur l'image plan. C'est à la fois une bonne introduction pour d'autres approches d'estimations et aussi très utiles dans la pratique pour l'initialisation des algorithmes de tracking 3-D. Ensuite nous nous intéresserons à la géométrie épipolaire, plus particulièrement à la matrice fondamentale qui caractérise la géométrie épipolaire calibrée.

Quelques méthodes d'estimation de la matrice fondamentale sont présentées par la suite. La détermination de la matrice fondamentale est réalisée à partir d'un ensemble de points appariés sur une paire d'images issues d'une séquence. L'extraction et la mise en correspondance de points sont des notions traitées au Chapitre 3. La matrice fondamentale nous permet d'estimer les paramètres de la caméra.

Deux approches différentes peuvent être considérées pour estimer ces paramètres externes. La première suppose que la scène observée est connue en donnant par exemple des correspondances entre les points 3-D M_i et leurs projections dans l'image m_i. La deuxième approche s'appuie essentiellement sur la géométrie épipolaire, qui grâce à l'estimation de la matrice fondamentale nous permet de reconstituer le mouvement de la caméra.

IV.1 Estimation de la matrice des paramètres externes de la camera dans une scène connue

Dans les sections qui suivront, on supposera qu'un ensemble de n correspondances entre les points 3-D M_i et leurs projections m_i est donné. Nous nous intéresserons à la matrice de projections P qui projette les M_i en m_i. En d'autres termes, nous essaierons de résoudre pour tout i $PM_i \equiv m_i$, ou $\equiv$ dénote l'égalité à un facteur d'échelle.

IV.1.1 Combien de correspondances sont nécessaires ?

Lorsque les paramètres internes sont connus, n=3 correspondances Mi ↔ mi connues produit 4 solutions possibles. Pour n = 4 ou 5 correspondances, il existe au moins 2 solutions en configuration générale, mais lorsque ces points sont coplanaires, et qu'il n'existe aucun triplet de points colinéaires, la solution est unique pour $n \geq 4$. Pour $n \geq 6$ correspondances la solution est unique.

IV.1.2 La méthode de transformation linéaire directe (DLT)

La DLT a été développée d'abord par les photogrammètres et ensuite a été introduite dans le domaine de la vision par ordinateur. Cette méthode peut estimer la matrice de projection perspective P en résolvant un système linéaire même lorsque les paramètres internes sont inconnus. Chaque correspondance Mi ↔ mi produit les deux équations linéairement indépendantes suivantes :

$$\frac{P_{11}X_i + P_{12}Y_i + P_{13}Z_i + P_{14}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} = u_i \qquad 4.1$$

$$\frac{P_{21}X_i + P_{22}Y_i + P_{23}Z_i + P_{24}}{P_{31}X_i + P_{32}Y_i + P_{33}Z_i + P_{34}} = v_i \qquad 4.2$$

Ces équations peuvent être réécrites sous la forme Ap = 0, où p est le vecteur formé par les coefficients P_{ij}. La solution p = 0 est bien sûr sans intérêt et une solution correcte peut être obtenue par décomposition en valeurs singulières (SVD) de A, qui correspond au vecteur propre associé à la valeur propre minimale de A.

Ainsi les paramètres internes et externes peuvent être extraits de P, mais il a été prouvé qu'il serait fastidieux d'appliquer cette approche directement dans le tracking 3-D. Dans la pratique, estimer simultanément les paramètres internes et externes dépend fortement de la géométrie et du nombre de correspondances. Dans des configurations favorables, 15 à 20 correspondances peuvent suffire. Mais dans des configurations

désavantageuses, mêmes des centaines peuvent ne pas suffire parfois. Dans de telles situations, il est souvent préférable d'estimer les paramètres internes séparément. Plus précisément, pour le tracking 3-D l'utilisation d'une caméra calibrée et estimer seulement ses paramètres externes $[R \mid t]$ (sa position et son orientation) produit des résultats beaucoup plus satisfaisants.

Une fois la matrice de calibrage K connue, la matrice des paramètres externes peut être extraite de P ainsi : $[R \mid t] \sim K^{-1}P$. La matrice 3x3 formée par les 3 colonnes de la matrice résultante n'est pas forcement la matrice de rotation. Cependant il existe des techniques pour corriger cela.

- **Décomposition en valeurs singulières**

Soit A une matrice réelle $m \times n$, cette matrice est décomposable de façon unique de la manière suivante :

$$A = UDV^T$$

Où :

U est une matrice orthogonale $m \times n$ dont les colonnes sont formées par les vecteurs propres de AA^T.

V est une matrice orthogonale $n \times n$ dont les colonnes sont les vecteurs propres de $A^T A$.

D est une matrice diagonale $n \times n$ définie comme suit :

D = diag $(\sigma_1, \sigma_2, ..., \sigma_n)$ telle que les σ_i soient les valeurs singulières de A et ordonnées de la manières suivante : $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$.

On rappelle que les valeurs singulières de A sont définies comme étant les racines carrées des valeurs propres de $A^T A$.

Où $\sigma_i = \sqrt{\lambda_i}$ et λ_i est une valeur propre de $A^T A$.

Dans des conditions réelles, les localisations des pixels m_i sont parfois bruitées. Et comme on sait que l'algorithme de DLT minimise seulement des erreurs algébriques contrairement aux erreurs quantifiant un sens physique (erreur géométrique). Les paramètres ainsi estimés par cette méthode doivent être affinés par l'optimisation

itérative de l'algorithme ''erreur de reprojection non linéaire'' que nous verrons dans le prochain paragraphe.

- **Erreur de reprojection non linéaire**

L'avantage des précédentes méthodes est qu'elles n'exigent pas une estimation initiale et aussi parce qu'elles sont rapides. Cependant, elles sont très sensibles aux bruits, donc présentent un manque de précision. En particulier, l'algorithme **DLT** produit une solution qui minimise une erreur algébrique alors qu'il serait préférable de minimiser une erreur géométrique.

Si les mesures m_i sont bruitées, le calcul de la pose de la caméra doit être affiné en considérant le minimum de la somme des erreurs de reprojection, qui n'est autre que le carré des distances entre la projection des points 3-D et leurs coordonnées 2-D mesurées. On peut ainsi écrire :

$$[\mathrm{R} \mid \mathrm{t}] = \arg\min_{[\mathrm{R} \mid \mathrm{t}]} \sum_i dist^2\left(P\tilde{M}_i, m_i\right)$$

4.3

La solution est optimale lorsque les erreurs de mesures sont indépendantes et gaussiennes. Une telle minimisation est obtenue par des méthodes itératives. Ces méthodes requièrent généralement une estimation de la pose initiale qui peut être fournie par les méthodes vues précédemment, ou bien lorsqu'il s'agit du tracking 3-D, par un modèle de mouvement appliqué aux estimations précédentes.

IV.1.3 Estimation de la pose à partir d'un plan 3-D

La pose de la camera peut aussi être estimée à partir d'une structure planaire lorsque les paramètres internes sont connus. Cette méthode est souvent utilisée dans le tracking 3-D du fait que les projections des structures planaires sont relativement faciles à détecter dans les images.

La relation entre un plan 3-D et sa projection sur l'image peut être représentée par une matrice homogène 3x3 appelée matrice d'homographie. Considérons le plan dans l'espace Z = 0. L'expression de la matrice d'homographie qui met en correspondance un point M = (X, Y, 0) sur ce plan avec son point 2-D *m* sur l'image sous la projection perspective P = K[R | t] peut être calculée comme suit :

$$\tilde{m} = P\tilde{M} \tag{4.4}$$

$$= K\begin{pmatrix} R^1 R^2 R^3 t \end{pmatrix}\begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} \tag{4.5}$$

$$= K\begin{pmatrix} R^1 R^2 t \end{pmatrix}\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{4.6}$$

$$= H\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{4.7}$$

Où R^1, R^2 et R^3 sont respectivement la première, la seconde et la troisième colonne de la matrice de rotation R.

Une fois les matrices H et K connues, la pose de la camera peut facilement être retrouvée. La matrice H peut être estimée a partir de 4 correspondances $M_i \leftrightarrow m_i$ en utilisant un algorithme de DLT similaire a celui qu'on a vu précédemment.

Comme on a: $H_w^t = K(R^1 R^2 t)$, le vecteur de translation et les deux premières colonnes de la matrice de rotation peuvent être calculés a partir du produit : $K^{-1} H_w^t$. Et comme les colonnes de la matrice de rotation sont orthogonales, la troisième colonne R^3 peut facilement être retrouvée par le produit vectoriel R^1 x R^2. Et comme précédemment, la pose peut être affinée par une minimisation non linéaire.

IV.2 Géométrie épipolaire [3]

La géométrie épipolaire est indépendante de la structure de la scène observée. Elle dépend des paramètres intrinsèques de la caméra et des positions successives de la caméra lors des différentes prises de vue. Toutes les positions de la caméra sont relatives à la première.

IV.2.1 Définitions

La géométrie épipolaire caractérise les relations spatiales entre deux images successives qui contiennent les projections d'un point 3-D. La figure 4.1 illustre la géométrie épipolaire d'un système de deux caméras. Dans le cas d'une seule caméra en mouvement la géométrie épipolaire reste inchangée.

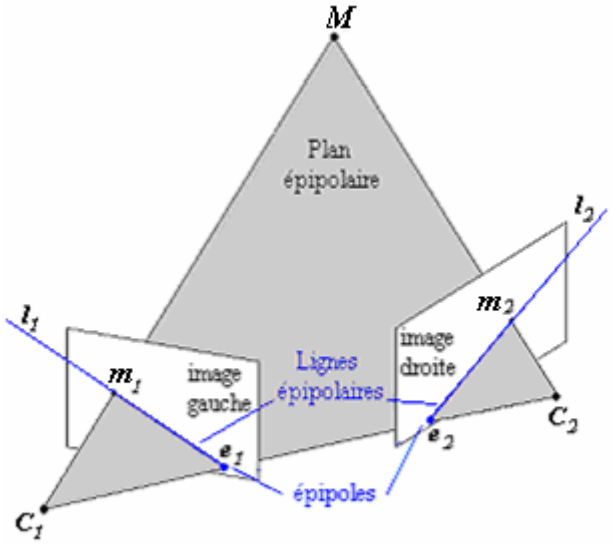


Figure 4.1 : La géométrie d'un système de deux caméras

Soit ***M*** un point 3-D défini dans le repère monde. Ce point forme avec les deux centres de caméras C_1 et C_2 le plan (C_1MC_2) nommé plan épipolaire. Les deux centres de caméras forment une droite (C_1C_2) appelée base. Il existe un faisceau de plans épipolaires associé à cette base. Par définition, un plan épipolaire coupe les deux plans image en deux droites l_1 et l_2 appelées droites épipolaires.

Dans le cas d'un faisceau de plans épipolaires, il y aura un faisceau de droites épipolaires pour chaque image. La figure 4.2 illustre le fait que pour une même base il y

a un faisceau de droites et de plans épipolaires. Une droite épipolaire dans une image coupe la base en un point ***e*** appelé épipôle qui est l'image du centre de projection de l'autre caméra.

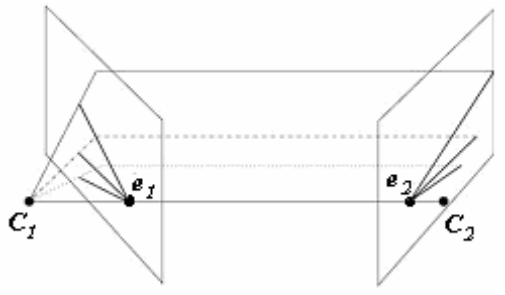


Figure 4.2 : Faisceaux de droites et de plan épipolaires

Soient deux points image m_1 et m_2 les deux projections du point 3-D ***M***. Les deux points image se trouvent sur les deux droites épipolaires associées au plan épipolaire de ***M***. Pour chaque position possible de ***M*** à l'intérieur du plan épipolaire considéré, les points image changent de position mais restent le long de la droite épipolaire associée. On peut donc dire que toutes les paires de points m_1, m_2 qui se trouvent sur deux droites épipolaires correspondantes, sont des projections possibles du point ***M***. Nous venons de décrire la contrainte épipolaire pour l'appariement de points dans deux images. De ce fait, le point m_2 est contraint de se trouver sur la ligne épipolaire l_2 passant par l'épipôle e_2 dans la seconde image. De façon réciproque, on peut dire que le point m_1 se trouve sur la ligne épipolaire l_1 passant par l'épipôle e_1 dans la première image.

La contrainte épipolaire peut s'exprimer algébriquement par l'équation fondamentale :

$$m_2^T F m_1 = 0 \qquad 4.8$$

Où ***F*** est une matrice 3× 3 appelée matrice fondamentale.

L'équation fondamentale (4.8) exprime le fait que le point m_2 se trouve le long de la ligne épipolaire l_2, correspondant au point m_1 :

$$l_2 = F m_1 \qquad 4.9$$

Cette équation induit que l'épipôle $\boldsymbol{e_2}$ est un point de la ligne $\boldsymbol{l_2}$. Ainsi $e_2^T l_2 = e_2(Fm_1) = (e_2^T F)m_1 = 0$ pour tout point appartenant à l'image 1. On en déduit que $e^T{}_2 F = 0$ et de façon similaire $Fe_1 = 0$.

IV.2.2 La matrice fondamentale

La matrice fondamentale est une représentation algébrique de la géométrie épipolaire non calibrée. Elle peut être définie de façon géométrique en considérant la géométrie épipolaire, ou algébrique grâce aux matrices de projection. D'un point de vue géométrique, on peut dire que la matrice fondamentale est la matrice de transformation qui à des points d'une image fait correspondre des droites de l'autre image, ces points étant exprimés en pixels dans les repères images. Nous posons que la matrice fondamentale $\boldsymbol{F_{12}}$ décrit la transformation épipolaire gauche-droite ou la transformation passant de l'image 1 à l'image 2. L'image de référence peut être inversée. Dans ce cas, la matrice fondamentale notée $\boldsymbol{F_{21}}$ équivaut à la transposée de $\boldsymbol{F_{12}}$. Dans la suite, la matrice fondamentale $\boldsymbol{F_{12}}$ sera notée $\boldsymbol{F}$. Dans ce cas, la contrainte épipolaire pour deux points image $\boldsymbol{m_1}$ et $\boldsymbol{m_2}$ s'exprime par l'équation (4.10).
Si la deuxième image devient l'image de référence, la matrice fondamentale est : $\boldsymbol{F_{21}}$.
En tenant compte de nos notations, on peut écrire que $\boldsymbol{F_{21}} = \boldsymbol{F}^T$. L'équation fondamentale dans ce cas devient :

$$m_1^T F^T m_2 = 0 \qquad 4.10$$

La matrice fondamentale peut être définie algébriquement à partir des matrices de projection. Si le système de coordonnées de la première position de la caméra coïncide avec le système de coordonnées monde, on obtient alors les expressions :

$$m_1 = K\begin{bmatrix} I_{3x3} & 0 \end{bmatrix} M \qquad 4.11$$

$$m_2 = K\begin{bmatrix} R & t \end{bmatrix} M \qquad 4.12$$

Où

- ***K*** représente la matrice des paramètres intrinsèques de la caméra,
- ***R*** et ***t*** représentent la matrice de rotation et le vecteur de translation qui décrivent la transformation permettant d'exprimer les points extraits dans la seconde image dans le repère de la première image. On en déduit les expressions des deux matrices de projection P_1 et P_2 :

$$P_1 = K\begin{bmatrix} I_{3x3} & 0 \end{bmatrix} \quad 4.13$$

$$P_2 = K\begin{bmatrix} R & t \end{bmatrix} \quad 4.14$$

A partir des équations (4.11) et (4.12), nous déduisons une nouvelle expression du point 3-D :

$$P_1^+ m_1 = M \quad 4.15$$

Où P_1^+ est la pseudo-inverse de la matrice de projection P_1.

La pseudo-inverse de P_1 est la matrice :

$$P_1^+ = P_1^T \left(P_1 P_1^T\right)^{-1} \quad 4.16$$

Avec $P_1 P_1^T = I$.

A partir des expressions des deux matrices de projection, on peut redéfinir l'expression d'une ligne épipolaire (4.9). Nous savons d'après la définition de la projection perspective que la projection d'un point 3-D ***M*** forme une droite joignant ce point 3-D au centre de la caméra C_1 en coupant le plan image en un point 2-D m_1 . Ces deux points sont représentés dans la seconde image en introduisant la seconde matrice de projection P_2. Le point 3-D est représenté par l'expression suivante : $P_2 P_1^T m_1$ et le centre de la première caméra dans la seconde image est exprimé par : $P_2 C_1$. Le point $P_2 C_1$ est donc l'épipôle dans la seconde image. La ligne épipolaire l_2 est la ligne joignant ces deux points projetés :

$$l_2 = \left[e_2\right]_x \left(P_2 P_1^+\right) m_1 \quad 4.17$$

Avec $[e_2]_x$ est la matrice antisymétrique associée au vecteur e_2.

En comparant cette équation avec l'équation (4.9), on en déduit l'expression de la matrice fondamentale :

$$F = [e_2]_X P_2 P_1^+ = [P_2 C_1]_X P_2 P_1^+ \qquad 4.18$$

En remplaçant P_1^+, P_2 et C_1 par leur expression réciproque, on peut réexprimer la matrice F :

$$F = [Kt]_X KRK^{-1} \qquad 4.19$$

La définition des matrices antisymétriques indique que pour un vecteur t et une matrice non singulière N on a :

$$[t]_x N = N^{-T}[N^{-1}t]_x \qquad 4.20$$

En posant $N^T = K^{-1}$ et à partir de l'équation (4.19), on en déduit l'expression de la matrice fondamentale F :

$$F = K^{-T}[t]_X RK^{-1} \qquad 4.21$$

Avec $[t]_x$: matrice antisymétrique associée au vecteur :

$$t = [t_x, t_y, t_z]^T : [t]_X = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

Une matrice fondamentale a deux caractéristiques propres. C'est une matrice singulière, définie à un facteur d'échelle près :

• Une matrice définie à un facteur d'échelle près

La matrice fondamentale *F* est définie à un facteur d'échelle près. Cependant, la matrice fondamentale *F* a seulement sept degrés de liberté. Ceci se traduit par l'existence de sept éléments indépendants parmi les neuf que comporte la matrice *F*.
Lors de l'estimation de la matrice fondamentale ***F***, une contrainte sera ajoutée pour prendre en considération le fait que la matrice ***F*** est définie à un facteur d'échelle près. La matrice fondamentale est alors exprimée par huit paramètres indépendants.

IV.2.3 Estimation de la matrice fondamentale

D'après l'équation de la contrainte épipolaire (4.8), nous pouvons dire que chaque paire de points associés $\boldsymbol{m}_1 = (u_1, v_1, 1)^T$ et $\boldsymbol{m}_2 = (u_2, v_2, 1)^T$, donne lieu à une équation linéaire avec les coefficients inconnus de ***F*** :

$$u_1u_2f_{11} + u_1v_2f_{21} + u_1f_{31} + v_1u_2f_{12} + v_1v_2f_{22} + v_1f_{32} + u_2f_{13} + v_2f_{23} + f_{33} = 0 \qquad 4.22$$

Cette équation linéaire peut être mise sous la forme suivante :

$$U^T f = 0 \qquad 4.23$$

Où : $U = [u_1u_2, u_1v_2, u_1, v_1u_2, v_1v_2, v_1, u_2, v_2, 1]$

$$f = [f_{11}, f_{21}, f_{31}, f_{12}, f_{22}, f_{32}, f_{13}, f_{23}, f_{33}]^T$$

Si on dispose de *n* correspondances gauche-droite, on obtient l'équation matricielle suivante :

$$Af = \begin{bmatrix} u_{11}u_{21} & u_{11}v_{21} & u_{11} & v_{11}u_{21} & v_{11}v_{21} & v_{11} & u_{21} & v_{21} & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ u_{1n}u_{2n} & u_{1n}v_{2n} & u_{1n} & v_{1n}u_{2n} & v_{1n}v_{2n} & v_{1n} & u_{2n} & v_{2n} & 1 \end{bmatrix} \qquad 4.24$$

La matrice fondamentale ***F*** est estimée, à un facteur d'échelle près, comme l'unique solution du système homogène suivant, formé à partir des équations (4.16), *n* étant le nombre de correspondances de points entre les deux vues :

$$m_{2i}^{T} F m_{1i} = 0, \forall i \in [1, n] \qquad 4.25$$

La matrice fondamentale contient toute l'information géométrique disponible et permet d'obtenir la géométrie épipolaire à partir de deux vues non-calibrées.

IV.2.4 Estimation des paramètres extrinsèques

Les paramètres extrinsèques expriment le mouvement tridimensionnel faisant passer du repère monde à celui de la caméra. Dans le cas d'une caméra en mouvement, nous considérons que l'estimation des paramètres extrinsèques revient à estimer un mouvement tridimensionnel reliant les positions successives de la caméra entre deux prises de vues. Le mouvement ainsi estimé est un mouvement relatif par rapport à la première position de la caméra. Le déplacement prend en compte le changement de position et d'orientation de la caméra.

IV.2.4.1 La matrice essentielle

Les méthodes d'estimation des paramètres extrinsèques d'une caméra s'appuient sur la factorisation de la matrice essentielle ***E***. La matrice essentielle est équivalente à la matrice fondamentale pour des coordonnées image normalisées. Un point image exprimé en coordonnées normalisées est un point dont les coordonnées tiennent compte des paramètres intrinsèques et extrinsèques de la caméra. Les coordonnées des points images sont donc exprimées en unités métriques. Si nous connaissons la matrice des paramètres intrinsèques de la caméra ***K***, nous pouvons estimer les coordonnées normalisées d'un point image $\boldsymbol{m}_1$:

$$\tilde{m}_1 = K^{-1} m_1 \qquad 4.26$$

De la même façon, un point image ***m***$_2$ s'exprimera par : $\tilde{m}_2 = K^{-1}m_2$ sous l'hypothèse d'utiliser une caméra unique en mouvement.

En tenant compte des équations générales d'une matrice de caméra $\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R} \mid \boldsymbol{t}]$ et d'un point dans une image $\boldsymbol{m} = \boldsymbol{PM}$ et en appliquant l'expression générale associée à (4.26), nous pouvons exprimer les coordonnées normalisées d'un point image : $\tilde{m} = [R| t]\, M$.

La matrice essentielle en termes de coordonnées images normalisées pour l'appariement de points $m_1 \leftrightarrow m_2$ est définie par :

$$\tilde{m}_2^T E \tilde{m}_1 = 0 \qquad 4.27$$

En remplaçant $\hat{q}_1$ et $\hat{q}_2$ par leur expression, l'équation (4.27) devient :

$$m_2^T K^{-T} E K^{-1} m_1 = 0 \qquad 4.28$$

Si nous considérons des points images exprimés en coordonnées normalisées et selon l'expression de la matrice fondamentale (4.8), nous pouvons réécrire la contrainte épipolaire

$(m_2{}^T F m_1)$ à l'aide de l'équation (4.21):

$$m_2^T K^{-T} [t]_X R K^{-1} m_1 = 0 \qquad 4.29$$

Avec : - $[\boldsymbol{t}]_x$: la matrice antisymétrique associée au vecteur de translation ***t,***

- ***R*** : la matrice de rotation fonction des trois angles d'Euler.

On en déduit l'expression de la matrice essentielle :

$$E = [t]_X R \qquad 4.30$$

La matrice ***E*** est donc composée de termes représentant la translation et la rotation de la caméra entre les deux prises de vues considérées.

La matrice essentielle $\boldsymbol{E}$ peut être estimée théoriquement à partir de cinq correspondances de points bien que les deux matrices $\boldsymbol{R}$ et $[\boldsymbol{t}]_x$ aient chacune trois degrés de liberté. On montre que la matrice essentielle est une matrice de rang 2 dont une valeur singulière est nulle et les deux autres identiques.

IV.2.4.2 Méthodes d'estimation du mouvement de la caméra

L'expression de la matrice essentielle peut être obtenue en comparant l'équation (4.28) et l'équation de la contrainte épipolaire (4.8). Nous en déduisons que :

$$E = K^T F K \qquad 4.31$$

La connaissance de la matrice fondamentale et des paramètres intrinsèques de la caméra nous permet d'obtenir directement la matrice essentielle.

Les paramètres du mouvement de la caméra peuvent être estimés en décomposant la matrice essentielle. La factorisation n'est pas unique. Le vecteur de translation $\boldsymbol{t}$ ne peut être déterminé qu'à un facteur d'échelle près. L'orientation de la caméra est connue avec un angle $\theta \pm \pi$. Ces deux indéterminations donnent quatre mouvements possibles de la caméra entre les deux prises de vue par rapport à la scène observée. La figure 4.3 illustre les quatre cas.

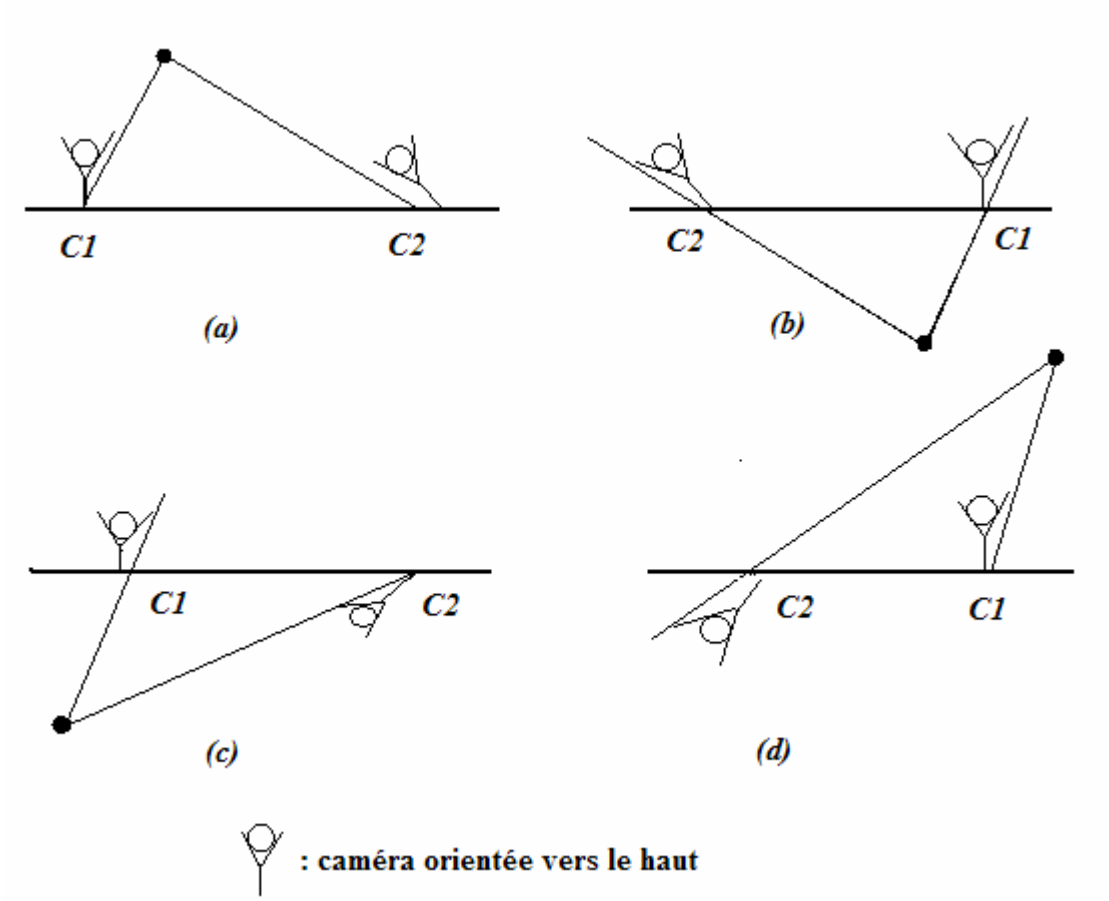


Figure 4.3 : différents cas d'orientations possibles de la caméra.

L'ambiguïté sur le signe de $\boldsymbol{t}$ (c'est-à-dire sur la direction de la translation), et celle sur l'angle de $\boldsymbol{R}$ peuvent être levées grâce à une correspondance de points. A partir d'une première estimation du vecteur $\boldsymbol{t}$ et de la matrice $\boldsymbol{R}$, on peut déterminer la position 3-D du point image considéré. Le point reconstruit doit se trouver devant la caméra. De nombreuses les méthodes font appel à la reconstruction 3-D du point image considéré pour valider chaque couple $(\boldsymbol{R}|\,\boldsymbol{t})$.

Plusieurs méthodes permettent d'estimer le mouvement de la caméra. Nous présenterons uniquement la méthode basée sur la décomposition en valeurs singulières du fait de sa simplicité.

- **Méthode basée sur la décomposition en valeurs singulières**

Cette méthode a été développée par Tsai et Huang en 1984 [15]. Elle utilise la décomposition en valeurs singulières de la matrice essentielle, $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$. Compte tenu du fait que les matrices $\boldsymbol{U}$ et $\boldsymbol{V}$ sont des matrices orthogonales, la matrice $\boldsymbol{E}$ s'écrit de la

façon suivante en faisant apparaître une matrice anti-symétrique et une matrice orthogonale :

$$E = \underbrace{UMU^T}_{[t]_X}\underbrace{UNV^T}_{R} \qquad 4.32$$

Avec

$$M = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{et} \quad N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

En remplaçant ***M*** et ***N*** par leur transposées, nous obtenons la seconde factorisation possible.

IV.3 Minimisation de l'erreur à l'aide des moindres carrés

La plupart des algorithmes de tracking 3-D estiment la pose de la caméra comme étant la solution d'un problème de minimisation des moindres carrés. Autrement dit, le problème revient à trouver les paramètres de la pose qui minimisent les erreurs résiduelles r_i comme ceci :

$$P = \arg\min_{p} \sum_{i} r_i^2 \qquad 4.33$$

L'expression peut être aussi réécrite sous forme vectorielle :

$$P = \arg\min_{p} \left\| f(p) - b \right\|^2 \qquad 4.34$$

Où **p** est un vecteur définissant les paramètres de la pose de la caméra, **b** est le vecteur qui contient les mesures et ***f*** est une fonction qui relie la pose de la camera aux mesures. Parfois, la fonction 'objectif ' ***f*** peut se présenter sous forme non linéaire en raison des carrés qu'elle contient, et souvent peu de paramètres sont à calculer, dans ce cas là l'algorithme de Newton est le plus adapté.

Dans d'autres cas, la fonction ***f*** est linéaire. Les paramètres de la pose peuvent ainsi être des inconnus d'un ensemble d'équations pouvant être écrites sous la forme matricielle suivante : **Ap = b**.

Souvent, le nombre d'éléments contenus dans **b** dépasse le nombre de paramètres dans **p**.

De la dernière équation on peut calculer la solution **p** d'après l'expression $\mathbf{p} = \mathbf{A}^{+}\mathbf{b}$

Avec $\mathbf{A}^{+} = (\mathbf{A}^{T}\mathbf{A})^{-1}\mathbf{A}^{T}$ est la pseudo inverse de **A**.

Dans notre cas, les mesures ont toutes la même influence, mais il serait plus convenable d'attribuer à chacune d'elles un poids qui diffère selon la qualité de l'information qu'elles apportent ; alors le problème se réécrit de la façon suivante :

$$P = \left(A^{T}WA\right)^{-1}A^{T}Wb \qquad 4.35$$

Où **W** est une matrice de poids, généralement diagonale.

Avec cette méthode l'estimation est calculée d'une manière itérative et les coefficients de la matrice **W** sont réestimés à chaque itération selon les valeurs des résidus. Comme nous allons le voir dans la prochaine section ceci permet une estimation robuste même en présences de fausses mesures.

IV.4 Estimation robuste [14]

Une estimation robuste est généralement indispensable dans le tracking 3-D pour contrer l'influence des données aberrantes. Par exemple, si la pose de la camera est estimée à partir d'un ensemble de correspondances Mi ↔ mi, si ces dernières contiennent une seule erreur, la pose calculée peut être très loin de la vraie solution. RANSAC et M-estimateurs sont deux méthodes populaires pour éviter cela.

Ces deux méthodes sont complémentaires. La méthode M-estimateurs produit des solutions très précises mais elle exige une estimation initiale pour bien converger. Par contre RANSAC ne requiert pas une telle estimation, mais elle produit une solution qui ne tient pas compte de toutes les données disponibles, et pour cela elle manque de précision. Ainsi lorsque une estimation initiale est disponible, la méthode la plus

appropriée est M-estimateurs. Et comme RANSAC nous fournit une telle estimation, elle peut être affinée par M-estimateurs.

IV.4.1 M-estimateurs

Il est bien connu qu'une estimation de la méthode des moindres carrés peut aussi être considérée comme une maximisation de la probabilité logarithmique en supposant que les observations sont indépendantes et ont une distribution gaussienne. La méthode des moindres carrés est très sensible aux erreurs flagrantes. Des modèles d'erreur plus réalistes ont été introduits par les statisticiens, ce qui les a mené à reformuler la fonction objectif, Au lieu de minimiser :

$$\sum_i r_i^2 \qquad 4.36$$

Où ri est l'erreur résiduelle comme vue dans l'équation (4.33), On minimise sa version robuste :

$$\sum_i \rho(r_i) \qquad 4.37$$

Où p est un M-estimateur qui réduit l'influence des aberrances. De nombreuses fonctions comme celle-ci ont été proposées. Deux des plus connues sont les suivantes :

- **Estimateur de Huber**

$$\rho Hub(x) = \begin{cases} x^2/2 & |x| \le c \\ c(|x| - c/2) & ailleurs \end{cases} \qquad 4.38$$

- **Estimateur de Tukey**

$$\rho Tuk(x) = \begin{cases} \frac{c^2}{6}\left[1-\left(1-\left(\frac{x}{c}\right)^2\right)^3\right] & |x| \le c \\ \frac{c^2}{6} & ailleurs \end{cases} \qquad 4.39$$

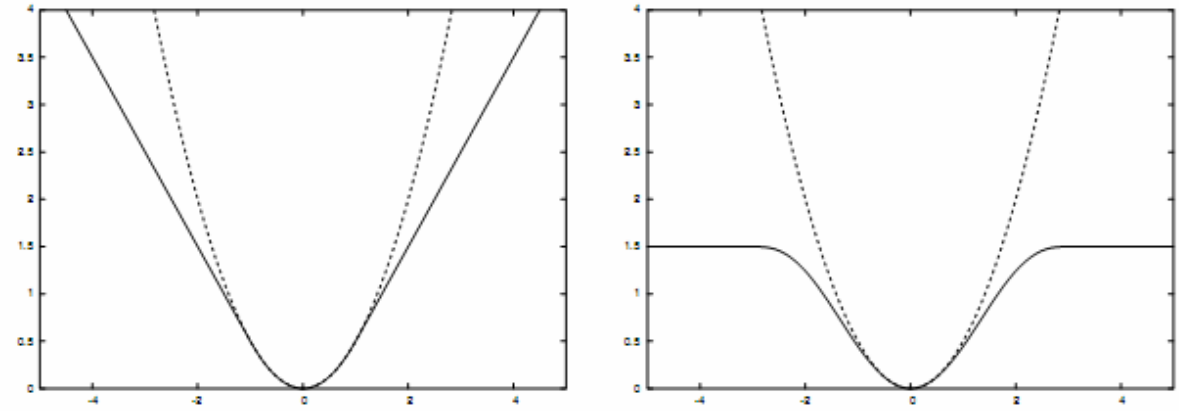


Figure 4.4: A gauche, le graphe de l'estimateur de Huber pour c = 1 ; A droite le graphe de l'estimateur de Tukey pour c = 3.
Dans les deux graphes, la courbe en pointillés représente le graphe des moindres carrées $x^2 / 2$.

Les deux équations précédentes se comportent comme un estimateur des moindres carrés ordinaire $x^2/2$ lorsque x est petit. Une fois que ||x|| s'écarte de plus en plus de c, l'estimateur de Huber devient linéaire pour réduire l'influence des grandes erreurs résiduelles, alors que l'estimateur de Tukey devient plat, ainsi les grandes erreurs résiduelles n'ont plus aucune influence. Cependant, l'estimateur de Huber est convexe, il fournit une convergence vers le minimum global plus efficace. Autre part, l'estimateur de Tukey est plus approprié pour supprimer l'influence des aberrances lorsque l'algorithme de minimisation peut être initialisé avec une estimation assez proche du minimum réel.
Le seuil c est généralement choisi pour être proportionnel à l'écart type mesuré des erreurs résiduelles des données exploitables.

Les algorithmes de Gauss Newton et Levenberg-Marquardt peuvent aussi être appliqués pour minimiser la somme des erreurs résiduelles après l'introduction de M-estimateurs, sachant que ce dernier peut comprendre des fonctions plus complexes (autres que Huber et Tukey).

IV.4.2 RANSAC

RANSAC, acronyme pour "RANdom SAmple Consensus", est un algorithme publié pour la première fois en 1981 par Fischler et Bolles. C'est une méthode itérative permettant d'estimer les paramètres d'un modèle mathématique à partir d'un ensemble de données observées qui contiennent des données aberrantes. RANSAC est un algorithme non déterministe, car il produit un résultat correct avec une certaine probabilité, cette probabilité augmentant avec le nombre d'itérations.

C'est une méthode générale pour vérifier la robustesse d'une estimation. Cette méthode a été développée dans le contexte de l'estimation de la pose d'une caméra, elle est très facile à implémenter et son avantage est qu'elle n'a pas besoin d'initialisation des paramètres à estimer. D'un autre côté, elle est souvent plus lente que la minimisation utilisant M-Estimateur. Par elle-même, elle est aussi moins précise mais peut être assistée en recherchant les minimisations pour affiner la solution.

Pour avoir des données utilisables, RANSAC extrait hasardement les plus petits sous-ensembles possibles pour générer les hypothèses du modèle de paramètres, ça maximise les chances pour qu'au moins un sous-ensemble ne contienne pas d'erreurs aberrantes et donc va produire une hypothèse valide. Par exemple, le premier article de RANSAC utilise l'algorithme des trois points (P3P) pour estimer la pose de la caméra en sélectionnant aléatoirement des triplets de correspondances. Pour chaque position, les points 3-D qui sont reprojetés assez près de leurs correspondant 2-D sont traités comme des données exploitables**.** RANSAC garde la pose qui engendre le plus grand nombre de données exploitables**.**

Plus formellement, on veut estimer les paramètres $\mathbf{P_i}$ d'un modèle à partir d'un ensemble de **S** mesures en sachant que quelques unes de ces mesures peuvent être erronées. A partir de là, on peut dire que ce modèle de paramètres requiert un minimum de *n* mesures pour qu'il puisse être estimé de manière acceptable. N échantillons de *n* points seront aléatoirement sélectionnés et chacun de ces échantillons sera utilisé pour

estimer un modèle de paramètres **P***i* et aussi de trouver le sous-ensemble **S***i* $\subset$ **S** des points qui sont cohérents avec cette estimation, seul le sous-ensemble qui engendre le plus grand **S***i* sera retenu et affiné par la méthode des moindres carrés en utilisant tous les points qui sont dans **S***i*. Une autre alternative plus efficace, est d'utiliser une méthode robuste des moindres carrés utilisant tous les éléments de **S** initialisée par la méthode RANSAC : ça permettra d'utiliser les données exploitables qui ont été initialement considérées comme des données aberrantes**.**

La tolérance d'erreur utilisée pour décider de la cohérence des estimations (observations) avec le paramètre estimé peut être considérée comme (souvent) les écarts types des erreurs de mesures. Pour garantir cette cohérence avec une probabilité **P**, le nombre **N** doit être supérieur à log (1-P)/log (1-w^n) où *w* est la proportion des données exploitables. Cette valeur augmente avec la taille des sous-ensembles et le rapport des données aberrantes**,** mais il est assez facile de voir qu'il demeure étonnement petit pour des valeurs de **P,** *w* et *n* raisonnable.

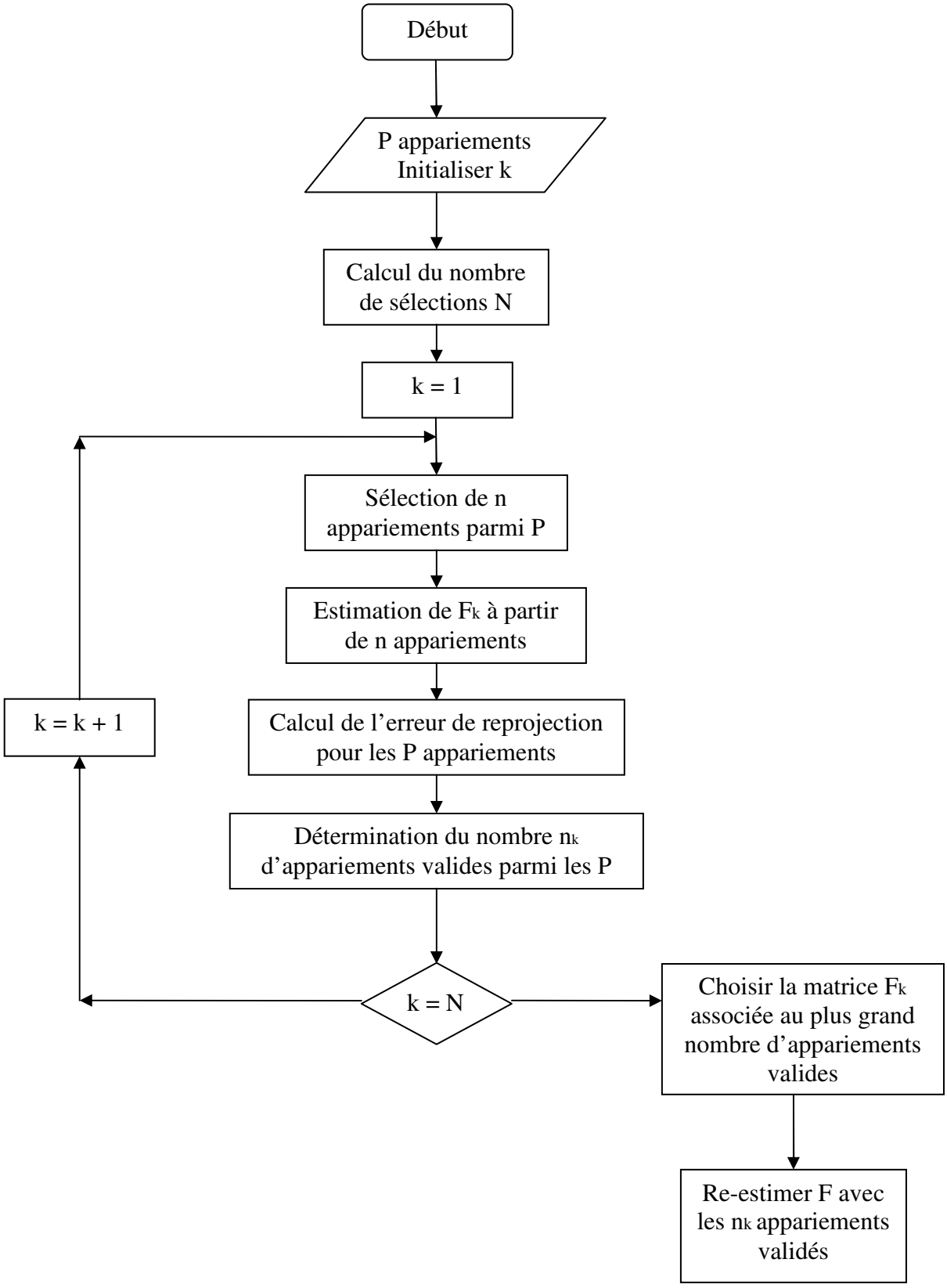


Figure 4.5 : Organigramme de RANSAC appliqué à l'estimation de la matrice fondamentale.

Discussion

Dans ce chapitre nous avons présenté deux approches essentielles pour estimer la position et l'orientation de la caméra. La première est utilisée lorsque certaines données métriques sur la scène sont disponibles. Cette méthode convient lorsque la caméra où le robot portant la camera évolue dans un environnement connu comme les milieux industriels où il est aisé de faire certaines correspondances entre les objets de la scène et leurs projections dans l'image. Cependant, il est parfois difficile voire impossible de recueillir ces données dans certaines applications comme par exemple en robotique mobile (exploration spatiale, exploration sous-marine …) où aucune information sur la scène n'est disponible. Dans ce cas, les méthodes basées sur la géométrie épipolaire s'imposent comme le moyen parfait pour estimer la position et l'orientation de la caméra et par la même occasion du robot sur lequel elle est embarquée.

Dans la majorité des cas, les images sont bruitées et le risque de faire une fausse correspondance est élevé entraînant ainsi une fausse estimation des paramètres. L'algorithme de Ransac qu'on a présenté a été élaboré à cet effet pour filtrer les données aberrantes des données fiables pour permettre une bonne estimation des paramètres de la caméra. Une fois les paramètres estimés, les résultats peuvent être affinés à l'aide d'algorithmes de minimisation de l'erreur géométrique comme par exemple l'algorithme des moindres carrés.

Chapitre V : Application.

Préambule

Ce chapitre sera consacré à la réalisation d'un programme mettant en pratique quelques principes étudiés dans le cadre de ce mémoire. Il consistera à retrouver en temps réel les six degrés de liberté d'une caméra dont la géométrie interne sera estimée par un calibrage off-line (en temps différé). Cette caméra évoluera dans un environnement dont la géométrie de certains objets est connue (un damier par exemple), voir figure 5.2.

Afin de réaliser cette application, nous avons utilisé le langage C++ dans l'environnement de développement Microsoft Visual Studio 2008 ainsi qu'une bibliothèque d'Intel très reconnue dans le domaine de la vision par ordinateur qui n'est autre que OpenCV (Open Source Computer Vision).

V.1 But de l'application

Notre application est destinée pour des domaines variés tel que la robotique et la réalité augmentée. Dans certaines applications robotiques, la caméra est soit embarquée sur un bras ou bien sur le corps même du robot, ainsi la position et l'orientation du robot ou de l'organe terminal seraient les mêmes que celles de la caméra.

Dans la réalité augmentée, le but est d'intégrer des objets virtuels dans une scène réelle en temps réel. La difficulté étant de reconstruire les changements d'apparence des objets virtuels lors des mouvements de caméra. Dans la figure 5.1 l'objet virtuel (la voiture) est parfaitement intégré dans la scène puisque après mouvement de la caméra le rendu de l'objet reste cohérent avec la scène.

Figure 5.1 : exemple d'intégration d'un objet virtuel dans une scène réelle (en temps réel).

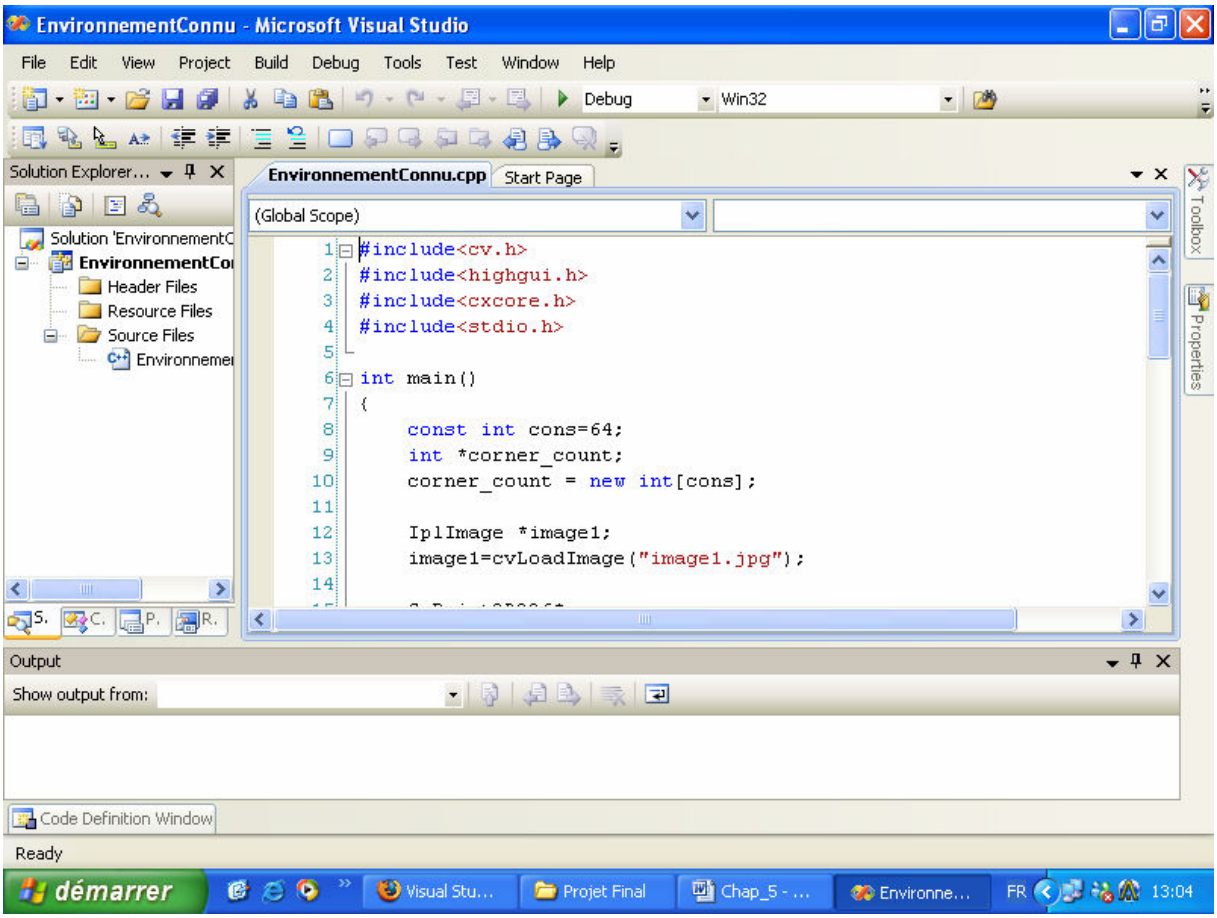
V.2 Outils utilisés

V.2.1 Visual Studio 2008 [1]

Microsoft Visual Studio est une suite de logiciels de développement pour Windows conçu par Microsoft. Pour notre application, on a utilisé la version 2008.

Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications Web ASP.NET, des Services Web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++ (celui que l'on utilisera), Visual C# et Visual J# utilisent tous le même environnement de développement intégré (IDE, Integrated Development Environment), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Par ailleurs, ces langages permettent de mieux tirer parti des fonctionnalités du Framework .NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications.

Voici une capture d'écran de l'interface graphique du logiciel Visual Studio 2008 :



Voici quelques touches raccourci pouvant servir pendant le développement d'applications dans l'environnement Visual Studio 2008 :
Ctrl+Shift+N permet de créer un nouveau projet.
Ctrl+Shift+O permet d'ouvrir un projet déjà existant.
La touche F7 permet de compiler un programme.
La combinaison de touches Ctrl+F5 exécute un programme compilé avec succès.

V.2.2 OpenCV (Open Source Computer Vision) [21]

OpenCV est une bibliothèque de fonctions graphiques destinée au développement d'applications de traitement d'images et de vision par ordinateur. Elle a été conçue à la base par les ingénieurs de la compagnie Intel. Comme son nom l'indique elle est open source, donc gratuite. Les exemples de réalisations sont très vastes, de la reconnaissance faciale à la segmentation d'images en passant par le tracking, cette bibliothèque offre vraiment toutes les fonctions utiles au développement d'applications de traitement d'images.
En plus de ces fonctions de base, OpenCV fournit aussi toute une gamme de fonctions algébriques nécessaires au développement des applications ciblées, aussi, elle dispose de fonctions systèmes permettant de capturer des flux vidéo à partir de caméras, de créer des fenêtres Windows et d'ouvrir des fichiers image et vidéo.

Il est très important de lier cette bibliothèque dans l'environnement de développement pour pouvoir utiliser ses fonctions. Les principales composantes à lier sont :

- **cv.h** : contient les fonctions d'analyse et de traitement d'images.
- **cvxcore** : contient tous les types et structures spécifiques au traitement d'images comme :
  IplImage : structure pouvant stocker les paramètres d'une image.
  CvMat : utilisée pour manipuler les matrices.
  CvPoint2D32f : utilisée pour manipuler les coordonnées des pixels.
  …etc
- **highgui** : (High Graphical User Interface) contient des fonctions pour la création d'interfaces graphiques tel que les fenêtres Windows.
- **cvcam** : pour la capture de vidéo en temps réel (à partir d'une caméra).

Pour chaque nouveau projet créé, il est nécessaire de lui lier les bibliothèques utilisées.

V.3 Approche proposée pour l'application

Pour notre application, nous avons imprimé un damier sur une feuille A4 qui nous servira d'objet de calibrage externe. Les dimensions et le nombre de carrés constituant le damier sont bien connus. A ce damier nous allons associer le repère monde dont l'origine sera située à son coin supérieur gauche (les rotations et les translations de la caméra seront relatives à ce repère). Les directions des axes X et Y seront parallèles au plan du damier quant à l'axe Z, il sera perpendiculaire.

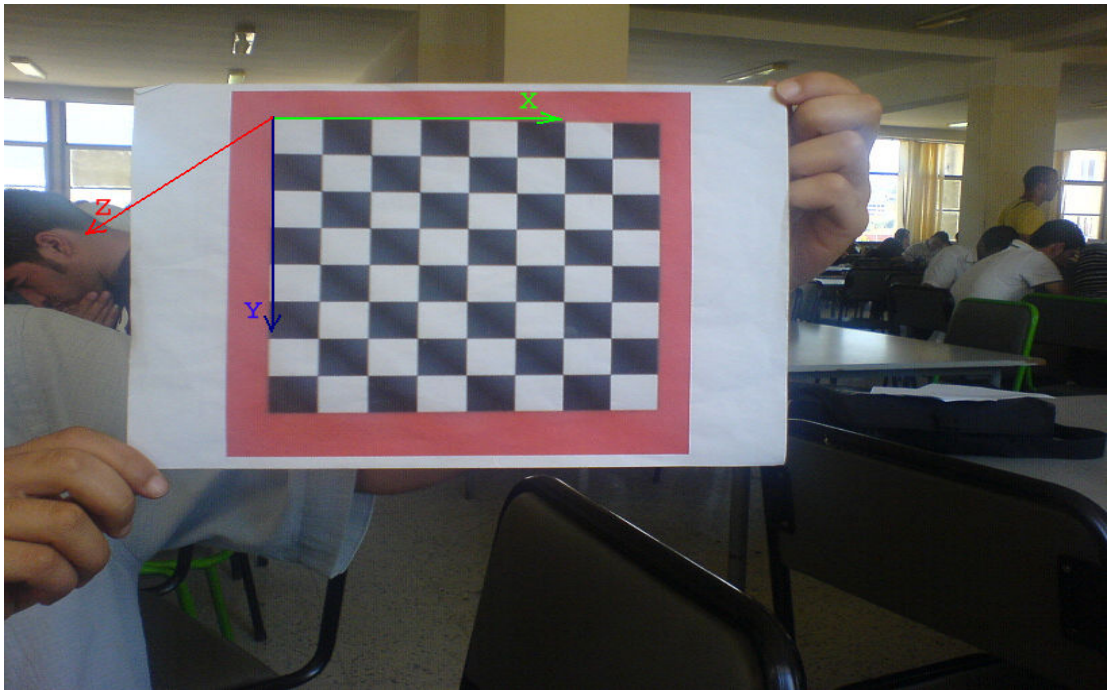


Figure 5.2 : position du repère monde sur l'objet de calibrage.

Avant d'entamer le calcul de la pose de la caméra, il est nécessaire de connaître la résolution des images ainsi que la focale (données fournies par le constructeur) pour remplir la matrice de calibrage interne K qui sera fixe tout au long du tracking.

Notre application se divise en deux parties. La première s'occupe de la détection de tous les points 2-D du damier en les plaçant dans un vecteur selon un ordre bien spécifique (ligne par ligne). Cette partie est basée essentiellement sur le détecteur de Harris. La seconde

partie se charge du suivi des points 2-D détectés préalablement, on gardera ainsi les mêmes correspondances de ces points. De ce fait, le premier point 3-D du damier correspondra toujours au premier élément du vecteur de points 2-D, la même cohérence est gardée pour tous les autres points.

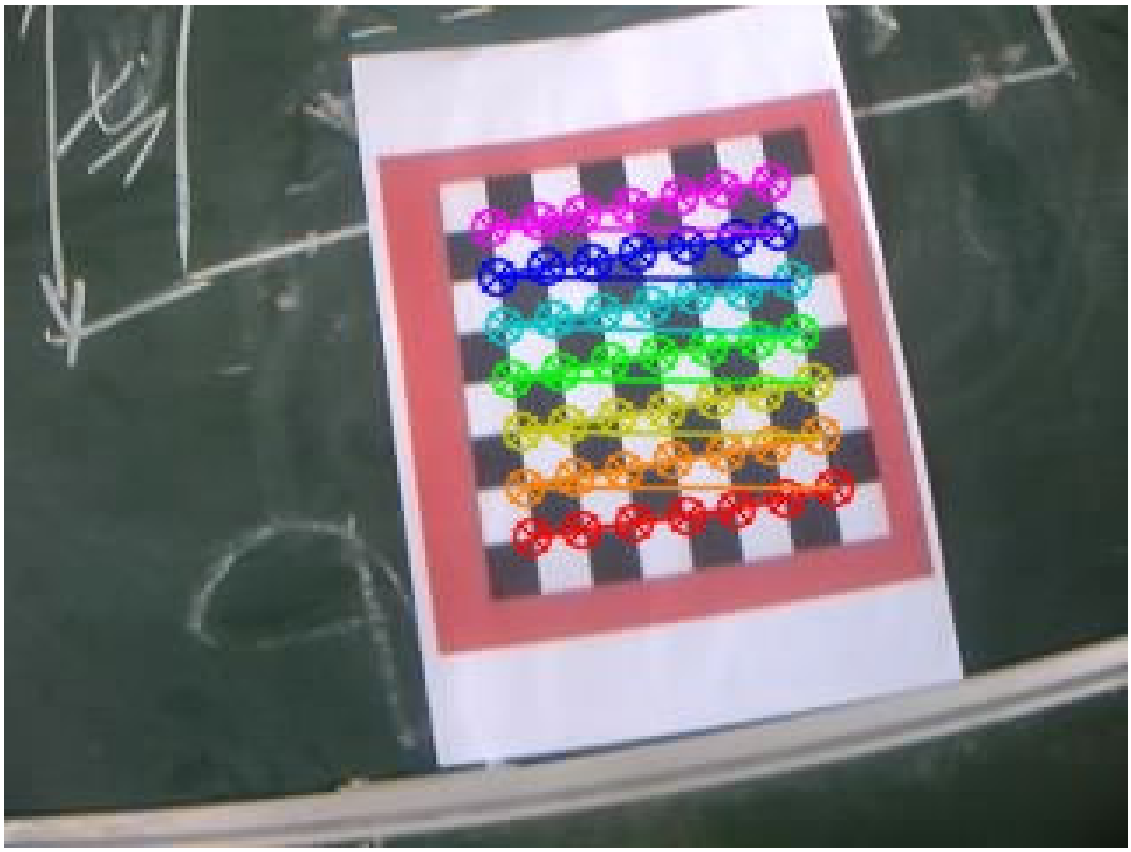


Figure 5.3 : illustration de la détection de points sur le damier.

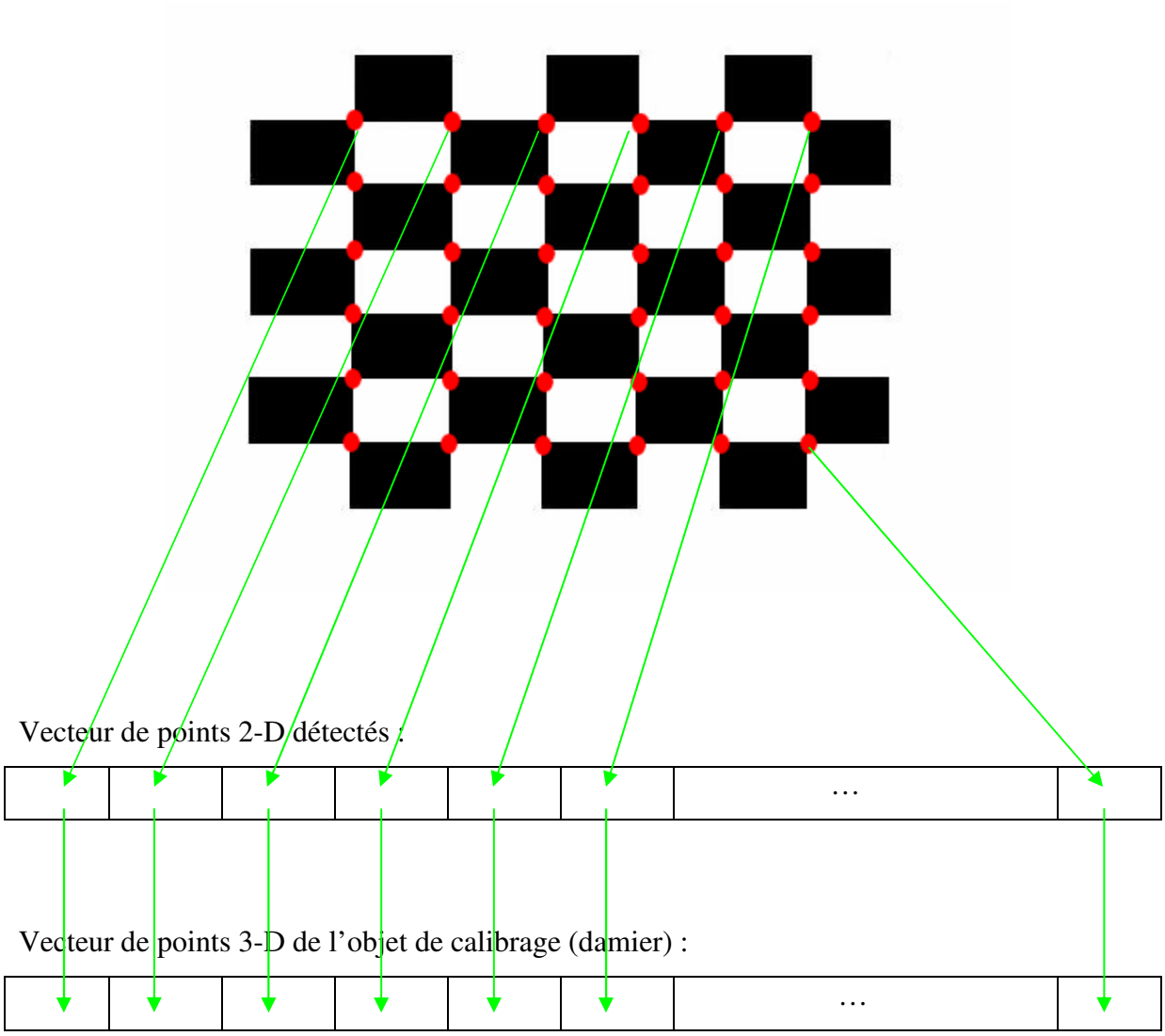


Figure 5.4 : correspondance entre les points 3-D de l'objet et les points 2-D détectés.

Après avoir rempli les deux vecteurs précédents, on calcule la pose de la caméra en utilisant les correspondances entre ces deux vecteurs. Les paramètres externes sont retrouvés par identification.

$$m_i = K\,[R\,|t]\,M_i$$

Avec

$m_i = (u_i, v_i)$ sont les points 2-D détectés dans l'image.

$M_i = (X_i, Y_i, Z_i)$ avec $Z_i = 0$ (situés sur le même plan) sont les points 3-D du damier.

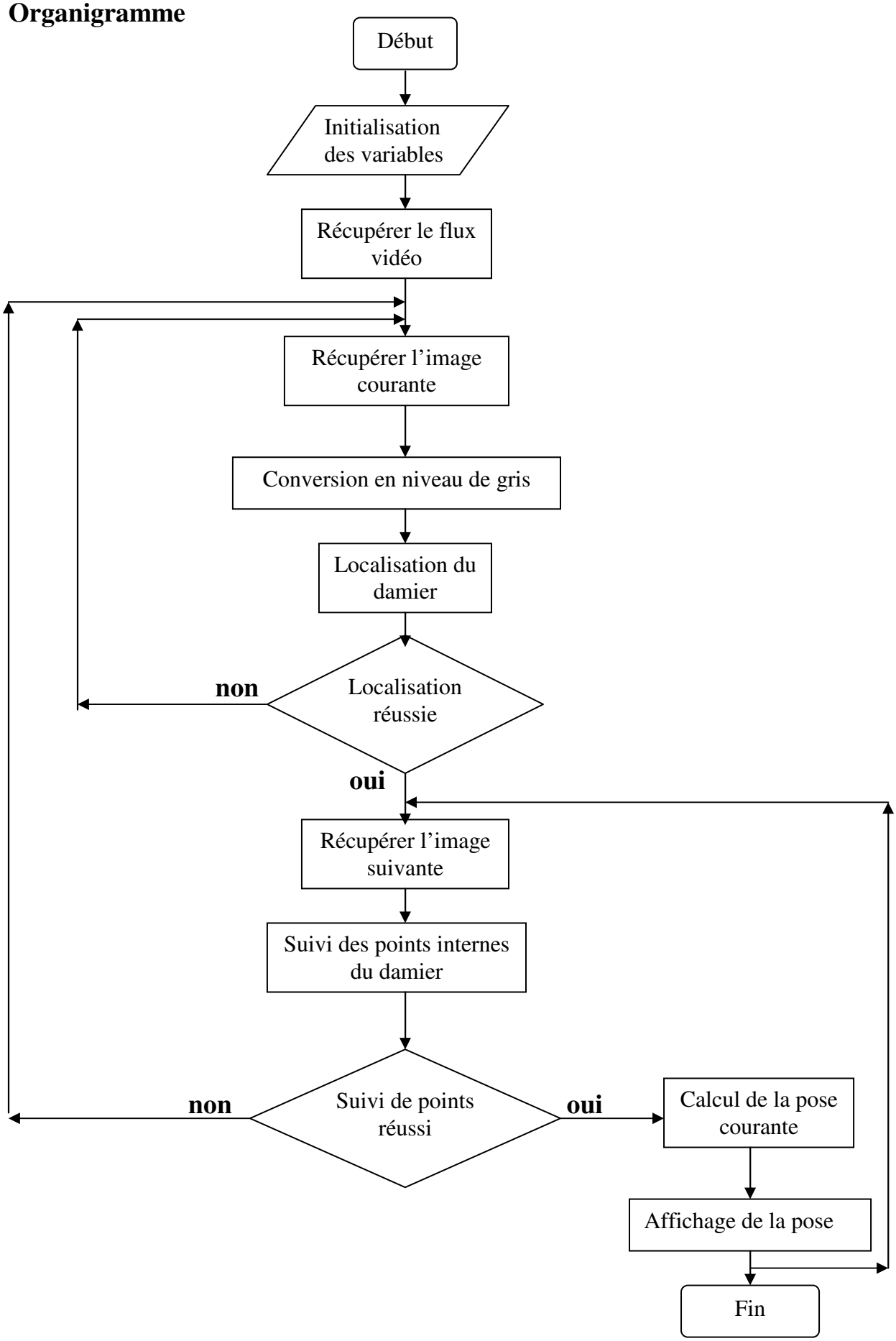
Les résultats seront affichés sous forme de vecteur t= (t_x , t_y , t_z) pour la translation et de même pour la rotation R= (R_x , R_y , R_z) qui sera interprété de la même manière que les maps exponentielles, à savoir :
L'axe de rotation est le vecteur R, et l'amplitude (angle) de la rotation est déterminée en calculant la norme du vecteur R.

V.4 Algorithme :

Pour implémenter notre application, nous allons créer une classe appelée ProcessImage qui contiendra les différents membres et fonctions suivante :

- **InitDonnées** : crée la matrice ObjetPoints qui contient les positions 3-D du damier. Elle crée aussi la matrice R de rotation et le vecteur t de translation initialement nuls.
- **CapturerVideo** : se charge de récupérer le flux vidéo image par image à partir d'une caméra.
- **ConvertirImage** : prend en paramètre une image en couleur et renvoi l'image en niveau de gris.
- **TrouverDamier** : basée sur une fonction OpenCv qui permet de détecter les points d'intérêt internes d'un damier. En effet, cette fonction utilise l'algorithme de détection de points d'intérêt (Harris) vu dans le chapitre 3. ces points seront stoker dans le vecteur ImagePoints1.
- **TrackerPoints** : prend en paramètre deux images (courante et suivante) ainsi que le vecteur ImagePoints1 renvoyé par la fonction précédente et nous renvoie un vecteur ImagePoints2 des nouvelles positions de ces points. Aussi, cette fonction permet de remplir un vecteur de même taille que ImagePoints1 appelé ErreurTrack dont on va se servir pour s'assurer que tous les points ont été suivis correctement (il contiendra des 1 pour les points retrouvés et des 0 ailleurs).
- **CalculPose** : prend en paramètres la matrice de calibrage K (connue par calibrage offline), les positions 3-D des points du damier (ObjetPoints) et leurs correspondants 2-D dans l'image (ImagePoints2) pour ainsi estimer la matrice de rotation R et le vecteur de translation t. Cette fonction est basée sur l'algorithme DLT vu dans le chapitre 4.

- **Organigramme**

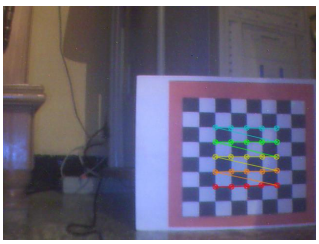


V.5 Résultats

On verra dans cette section quelques exemples de déplacements de la caméra et les résultats fournis par notre application. On précisera que les tests ont été effectués avec des séquences d'images réelles.

- Translation selon l'axe X

Séquence d'images test 1



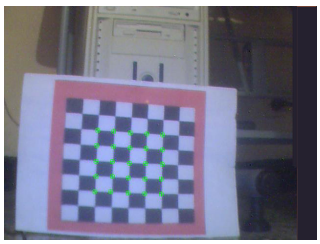



	Image 1	**Image 2**	**Image 3**	**Image 4**
t_x	3.8056	4.4835	5.2500	6.0928
t_y	2.6014	2.6217	2.6115	2.6087
t_z	19.5794	19.6895	19.9306	20.1127
R_x	-0.0521	-0.0269	-0.0162	-0.0178
R_y	0.4555	0.4233	0.4869	0.4338
R_z	3.0466	3.0582	3.0676	3.0779

Table 5.1 : résultats d'une translation selon l'axe X.

- Translation selon l'axe Y

Séquence d'images test 2

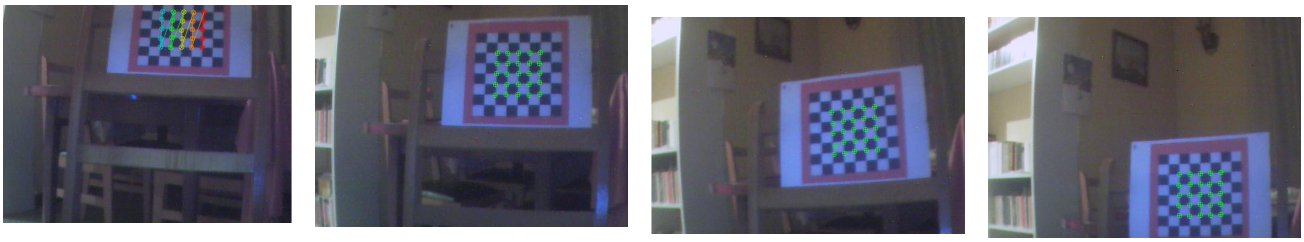


	Image 1	**Image 2**	**Image 3**	**Image 4**
$\mathbf{t_x}$	3.2623	3.4381	3.6893	3.7372
$\mathbf{t_y}$	-2.1475	-1.2674	-0.2615	0.4311
$\mathbf{t_z}$	23.8656	23.9794	23.7832	23.5097
$\mathbf{R_x}$	0.2464	0.2423	0.2319	0.2268
$\mathbf{R_y}$	-0.2578	-0.2257	-0.2120	-0.2124
$\mathbf{R_z}$	1.4291	1.4405	1.4506	1.4515

Table 5.2 : résultats d'une translation selon l'axe Y.

- Translation selon l'axe Z

Séquence d'images test 3

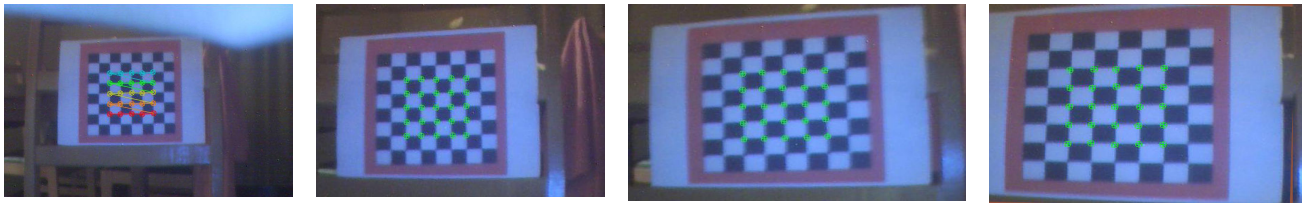


	Image 1	**Image 2**	**Image 3**	**Image 4**
t_x	2.2475	2.3471	2.2985	2.2719
t_y	-3.0468	-3.0422	-3.0473	-3.1516
t_z	30.3752	31.4325	32.2993	33.1952
R_x	0.2793	0.2761	0.2756	0.2726
R_y	-0.3156	-0.3092	-0.3099	-0.3121
R_z	1.4140	1.4258	1.4200	1.4079

Table 5.3 : résultats d'une translation selon l'axe Z.

- Rotation selon l'axe Y

Séquence d'images test 4

	Image 1	Image 2	Image 3	Image 4
t_x	14.0028	13.2544	11.0032	9.5191
t_y	1.0590	1.0467	1.0288	1.0177
t_z	36.1161	36.8034	36.0124	36.7243
R_x	0.2605	0.2568	0.2265	0.1761
R_y	0.2824	0.3381	0.4243	0.4663
R_z	3.1290	3.1269	3.1205	3.1165

Table 5.4 : résultats d'une rotation selon l'axe Y.

- Rotation selon l'axe Z

Séquence d'images test 5

	Image 1	Image 2	Image 3	Image 4
t_x	1.6790	1.6261	1.6826	1.6513
t_y	-5.1519	-5.1678	-5.0562	-5.2457
t_z	22.5759	22.6946	22.7108	22.4939
R_x	0.1722	0.1949	0.2015	0.1863
R_y	-0.1961	-0.1801	-0.1975	-0.1875
R_z	0.8493	0.7101	0.5907	0.4893

Table 5.5 : résultats d'une rotation selon l'axe Z.

- Déplacement aléatoire

	Image 1	Image 2	Image 3	Image 4
t_x	2.6341	3.2524	4.0684	5.5680
t_y	-0.8699	-0.6740	-0.7859	-3.2841
t_z	27.7158	28.1570	28.5142	25.1966
R_x	-0.7681	-0.7147	-0.5676	-0.0255
R_y	-0.4837	-0.4732	-0.4892	-0.7753
R_z	1.4335	1.4126	1.3663	1.5183

Table 5.6 : résultats d'un déplacement aléatoire.

V.6 Interprétation des résultats

On remarque que les translations sont très bien interprétées par le programme. La table 5.1 illustre bien un changement significatif selon l'axe X (seul tx varie) et une légère variation des autres composantes dues à l'instabilité de la caméra qu'on peut très bien négliger.
Idem pour les tables 5.2 et 5.3 qui interprètent de la même manière les résultats suivant les axes considérés (resp. Y et Z).
Pour les rotations, la table 5.4 montre un changement selon l'axe Y qui interprète une rotation de la caméra autour de ce même axe. Néanmoins, une translation considérable est observée selon l'axe X. Ceci est dû au fait qu'une rotation selon l'axe Y induit un changement d'apparence assimilé à une translation selon l'axe X. La même interprétation est faite pour la rotation selon l'axe X qui implique une translation selon l'axe Y.
D'après la table 5.5, la rotation selon l'axe Z ne présente aucun problème puisque seule la composante Rz change. Cela s'explique par le fait qu'aucune translation ne peut être apparentée à cette rotation.

Les valeurs exprimées dans les tables précédentes sont données en unités métriques arbitraires qui représentent la longueur d'un carreau du damier pour les translations et en radians pour les rotations.

Discussion

Ce chapitre donne un aperçu des applications du calibrage extrinsèque d'une caméra. Notre travail a donné lieu au développement d'un programme pouvant déterminer la position 3-D et l'orientation d'une camera à l'aide d'un objet de calibrage dont les dimensions sont bien connues et ce en temps réel. Les résultats obtenus sur les translations pures sont très satisfaisants. Cependant, on remarque que parfois des translations s'ajoutent aux rotations pures. Pour visualiser les résultats d'une manière plus claire, nous pouvons relier les resultats fournis par le programme à un moteur 3-D qui représente une caméra virtuelle pouvant se déplacer dans une plateforme 3-D.

Conclusion

Conclusion

Le sujet dont a fait l'objet ce mémoire a pour but de présenter un problème classique dans le domaine de la vision 3-D par ordinateur, à savoir le tracking 3-D qui consiste à retrouver la pose d'une caméra (position et orientation dans l'espace) par rapport à un repère arbitraire dans l'espace. Les applications du tracking 3-D se trouvent être très nombreuses et particulièrement lorsque la résolution se fait en temps réel, cela permet par exemple d'asservir visuellement un bras robotique, d'aider un robot mobile à se retrouver et à s'orienter dans l'espace ou bien d'incruster des objets virtuels dans des scènes réelles dans le cas de la réalité augmentée.

Nous avons pu constater dans notre étude que le choix d'une approche pour résoudre un problème de tracking 3-D reste très lié à l'application pour laquelle est destiné le travail.

Cependant, même après quelque vingt années de recherche les systèmes de vision basés sur le tracking 3-D restent toujours dépendants des marqueurs placés dans la scène comme les LEDs ou des objets dont le modèle 3-D est connu. Cette méthode est la seule capable à nos jours de produire des résultats suffisamment fiables exécutée en un temps de calcul raisonnable. Donc s'il est possible d'identifier des éléments dont la géométrie est connue sans pour autant gêner l'utilisateur, cette solution se révèle comme étant le meilleur moyen pour garantir des résultats acceptables.

Les approches concernant l'utilisation du tracking 3-D dans des environnements inconnus (aucune information géométrique de la scène) sont d'autant plus difficiles car la résolution demande plusieurs itérations pour éliminer les données aberrantes et aussi pour vérifier plusieurs contraintes liées au modèle du mouvement de la caméra.

Toutefois, cette situation est appelée à changer vu la puissance que gagne de plus en plus les ordinateurs permettant ainsi un traitement en temps réel des primitives présentes naturellement dans les images, donc indépendamment des marqueurs artificiels.

Lors de l'implémentation de l'application nous avons remarqué que la performance des algorithmes est souvent liée à la qualité de la mise en œuvre. Un calibrage grossier des paramètres internes d'une caméra peut suffire à donner de bons résultats alors qu'un calibrage plus précis peut apporter des améliorations surprenantes.

Même si les algorithmes de tracking 3-D sont sur le point de devenir pratiques sans utilisations de marqueurs, nous devons être conscients que le caractère récursif de la plupart de ces algorithmes les rend fragiles. Ils doivent être initialisés manuellement, et ils ne peuvent pas être re-initialisables si le processus échoue pour une quelconque raison. Dans la pratique, même les méthodes les plus robustes souffrent souvent de ces échecs pour des raisons de mouvements brusques, ou tout simplement du fait que l'objet cible disparaît momentanément du champ de vue. Parfois ces problèmes peuvent être redressés en combinant les données des images avec les données dynamiques fournies par des capteurs classiques comme par exemple les capteurs à ultrasons. Ces capteurs permettront de prédire la position de la camera ou un mouvement relatif qui peut être raffiné par une technique de vision similaire à celle qu'on a traitée dans notre projet. Une telle combinaison est possible pour des applications comme celles de la robotique et de la réalité augmentée.

Par conséquent, l'approche la plus désirable est de développer des méthodes purement basées sur les images qui peuvent détecter un objet cible et calculer sa position 3-D à partir d'une seule image. Si de telles méthodes seront suffisamment rapides, elles pourront être utilisées pour initialiser et re-initialiser le système aussi souvent qu'il en aura besoin.

Nous espérons que ce mémoire suscitera un engouement et un intérêt scientifique particulier chez les promotions futures afin qu'elles puissent améliorer le modeste travail que nous avons entamé, et ce en développant l'étude et la mise en œuvre des algorithmes basés sur des primitives autre que les points d'intérêt comme les contours et les textures. Toutefois, cette étude nous a permis d'entrevoir avec plus de motivations d'éventuels travaux de recherche dans le vaste domaine qu'est la vision par ordinateur et a suscité chez nous l'envie d'explorer les multiples facettes de cette science.

Glossaire

Termes et abréviations utilisés

Photogrammétrie : Application de la stéréophotographie aux relevés des formes, des reliefs.

Film : Support d'acquisition pour les caméras analogiques (pellicule).

CCDs : Charged Coupled Device, Support d'acquisition pour les caméras numériques.

NTSC : Format de capture de 30 images par seconde.

PAL : Format de capture de 25 images par seconde.

CDV : Champs de vue.

ADV : Angle de vue.

Frustum : Pyramide créée par le CDV et le ADV.

Mire : Grille de calibrage.

Map exponentielle : Vecteur rotatif.

Pose : Paramètres internes et externes de la caméra.

CP : Centre de projection (Sténopé).

RA : Réalité augmentée.

SSD : Sum of Squared Differences, somme des differences des carrés.

CC : Cross-Correlation , corrélation croisée.

ZSSD : Zero-mean Sum of Squared Differences.

ZNSSD : Zero-mean Normalized Sum of Squared Differences.

ZNCC : Zero-mean Normalised Cross-Correlation.

DLT : Méthode de transformation linéaire directe.

SVD : Singular Values Decomposition, Décomposition en valeurs singulières.

OpenCv : Open Source Computer Vision. Bibliothèque pour le traitement d'images et la vision par ordinateur.

C++ : Langage de programmation utilisé pour le développement de notre application.

Symboles mathématiques

(X, Y, Z) : Position réelle (3-D) du point.

(u_i, v_i) : Position 2-D du point 3-D.

θ_x : Angle de rotation selon l'axe X.

θ_Y : Angle de rotation selon l'axe Y.

θ_z : Angle de rotation selon l'axe Z.

R^3 : Repère a 3 dimensions (monde réel).

exp (…) : Exponentiel.

X^T : Transposée d'une matrice X.

$X^{-T} = (X^T)^{-1}$: Transposée inverse d'une matrice X

M : Point 3-D de la scène, ses coordonnées homogènes sont : $\hat{Q}$ =(X,Y,Z,1) ^{T} .

m : Point 2-D d'une image, ses coordonnées homogènes sont : $\hat{q}$ =(u,v) ^{T}.

K : Matrice des paramètres internes.

f : Distance focale.

(u_0, v_0) : Coordonnées du point principal.

t : Vecteur de translation.

$[t]_X$: Matrice antisymétrique associée au vecteur t.

R : Matrice de rotation.

F : Matrice fondamentale.

E : Matrice essentielle.

H : Matrice d'homographie.

C : Centre de projection ou centre optique.

P : Matrice de projection associée au modèle pinhole de la caméra.

P^+ : Pseudo-inverse de la matrice P.

l : Ligne épipolaire.

e : Epipôle.

I (u, v) : Intensité au pixel (u, v).

$\otimes$: Produit de convolution.

λ_1 , λ_2 : Valeurs propres.

σ_1, σ_2 : Valeurs singulières.

Bibliographie

[1]: I. Horton
Ivor Horton's begenning Visual C ++ 2008, Wiley, 2008.

[2]: M. Armstrong
Self calibration from image sequences, Thèse doctorat, Université d'Oxford, 1996.

[3]: R. Hartley, A. Zisserman
Multiple View Geometry in Computer Vision, Cambridge University Pr, 2003.

[4]: T. Dobbert
Matchmoving: The Invisible Art of Camera Tracking, Sybex, 2005.

[5]: Institut Charle Cros
http://www.igm.univ_mlv.fr/~vnozick/

[6]: P-A. Fortin
Gestion des occultations en réalité augmentée, Maître ès Sciences, Faculté des sciences et de génie, Université Laval, Quebec, 2006.

[7]: S. Perreault
Algorithme modélisation de l'apparence, Maître ès Sciences, Faculté des sciences et de génie, Université Laval, Quebec, 2007.

[8]: K. G-Derpanis
Overview of the RANSAC algorithm, 2005.
http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac

[9]: A. Topol
VRLM: étude, mise en oeuvre et applications, thèse ingéniorat, Conservatoire national des arts et des métiers, 2001.

[10]: F. Landragin
Interaction multimodale dans un environnement virtuel, thèse ingéniorat, Institut d'informatique et d'entreprise, 1998.

[11]: N. Sinha, J. Frahm, M. Pollefeys, Y. Genc
GPU based video feature tracking and matching, department of computer science, Université de Caroline du nord.

[12]: E. Arnaud, E. boyer
Analyse d'images et extraction de caractéristiques.
http://perception.inrialpes.fr/~Arnaud/vision/

[13]: D. Bereziat
Traitement des images médicales, Université Pierre et Marie Curie, 2008.

[14]: O. Moslah
Reconstitution 3-D densifiée d'un environnement urbain par analyse d'une séquence vidéo monoculaire, rapport de stage, université Cergy-Pontoise, 2004.

[15] : R.Y. Tsai, T.S. Huang
Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces, In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 6, 1984.

[16] : J.Leiterman
Vector Games Math Processors, Wordware Publishing, Inc, 2002.

[17]: H. Moravec
Robot Rover Visual Navigation. Ann Arbor, Michigan: UMI, Research Press, 1981.

[18]: C.Harris, M.Stephens, A combined corner and edge detector, in Fourth
Alvey Vision Conference, Manchester, 1988.

[19]: Z. Zhang, R. Deriche, O.Faugeras, Q.Luong, A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry, Artificial Intelligence,1995.

[20]: A. Koschan
What is new I computational sterio since 1989, A survey on current sterio paper, rapport technique, Technical University of Berlin, Department of Computer Science, 1993.

[21]: Intel® Company
Open Source Computer Vision Library
http://www.intel.com/software/products/ipp/index.htm

[22] : P. Aschwanden, W. Guggenbühl, Experimental results from a comparative study on correlation-type registration algorithms, ISPRS Workshop, Bonn, Germany, 1992.