# Modélisation et développement d'une plateforme intelligente pour la capture d'images panoramiques cylindriques

Frantz Pelissier

N°d'ordre : D.U : 2486

EDSPIC : 665


**UNIVERSITE BLAISE PASCAL - CLERMONT II**

ECOLE DOCTORALE
SCIENCES POUR L'INGENIEUR DE CLERMONT-FERRAND


# Thèse

Présentée par


## Frantz PÉLISSIER


Pour obtenir le grade de

## DOCTEUR D'UNIVERSITÉ

**SPÉCIALITÉ : Vision pour la robotique**

**Titre de la thèse :**

## Modélisation et développement d'une plateforme intelligente pour la capture d'images panoramiques cylindriques

soutenue publiquement le **11 septembre 2014** devant le jury :

| | |
|---|---|
| M. François Berry | Directeur de thèse |
| M. Omar Ait Aider | Co Directeur de thèse |
| M. Dominique Ginhac | Rapporteur |
| M. El Mustapha Mouaddib | Rapporteur |
| M. Richard Kleihorst | Examinateur |
| M. Youcef Mezouar | Examinateur |
| M. Yang Ni | Examinateur |

# Résumé

Dans la plupart des applications de robotique, un système de vision apporte une amélioration significative de la perception de l'environnement. La vision panoramique est particulièrement intéressante car elle rend possible une perception omnidirectionnelle. Elle est cependant rarement utilisée en pratique à cause des limitations technologiques découlant des méthodes la permettant. La grande majorité de ces méthodes associent des caméras, des miroirs, des grands angles et des systèmes rotatifs ensembles pour créer des champs de vision élargis. Les principaux défauts de ces méthodes sont les importantes distorsions des images et l'hétérogénéité de la résolution. Certaines autres méthodes permettant des résolutions homogènes, prodiguent un flot de données très important qui est difficile à traiter en temps réel et sont soit trop lents soit manquent de précision. Pour résoudre ces problèmes, nous proposons la réalisation d'une caméra panoramique intelligente qui présente plusieurs améliorations technologiques par rapport aux autres caméras linéaires rotatives. Cette caméra capture des panoramas cylindriques homogènes avec une résolution de $6600 \times 2048$ pixels. La synchronisation de la capture avec la position angulaire est possible grâce à une plateforme rotative de précision. Nous proposons aussi une solution au problème que pose le gros flot de données avec l'implémentation d'un extracteur de primitives qui sélectionne uniquement les primitives invariantes des images pour donner un système panoramique de vision qui ne transmet que les données pertinentes. Le système a été modélisé et une méthode de calibrage spécifiquement conçue pour les systèmes cylindriques rotatifs est présentée. Enfin, une application de localisation et de reconstruction 3D est décrite pour montrer une utilisation pratique dans une application de type Simultaneous Localization And Mapping (SLAM).

# Abstract

In most robotic applications, vision systems can significantly improve the perception of the environment. The panoramic view has particular attractions because it allows omnidirectional perception. However, it is rarely used because the methods that provide panoramic views also have significant drawbacks. Most of these omnidirectional vision systems involve the combination of a matrix camera and a mirror, rotating matrix cameras or a wide angle lens. The major drawbacks of this type of sensors are in the great distortions of the images and the heterogeneity of the resolution. Some other methods, while providing homogeneous resolutions, also provide a huge data flow that is difficult to process in real time and are either too slow or lacking in precision. To address these problems, we propose a smart panoramic vision system that presents technological improvements over rotating linear sensor methods. It allows homogeneous 360 degree cylindrical imaging with a resolution of $6600 \times 2048$ pixels and a precision turntable to synchronize position with acquisition. We also propose a solution to the bandwidth problem with the implementation of a feature etractor that selects only the invariant feaures of the image in such a way that the camera produces a panoramic view at high speed while delivering only relevant information. A general geometric model has been developped has been developped to describe the image formation process and a caligration method specially designed for this kind of sensor is presented. Finally, localisation and structure from motion experiments are described to show a practical use of the system in SLAM applications.

# Remerciements

En premier lieu, je tiens à remercier mon directeur de thèse, monsieur *François Berry*, pour m'avoir permis d'intégrer le LASMEA en tant que stagiaire, pour m'avoir ensuite permis de faire une thèse et pour la grande confiance qu'il m'a accordée tout au long de mes travaux. Et d'une manière générale, pour toutes les compétences qu'il m'a permis d'acquérir.

Je souhaiterais aussi exprimer ma gratitude à monsieur *Omar Ait Aider*, mon co-directeur de thèse, pour son soutien, sa disponibilité et toute l'aide qu'il m'a apportée dans l'assimilation des connaissances de vision.

Je remercie messieurs *Dominique Ginhac* et *El Mustapha Mouaddib* pour avoir accepté d'être mes rapporteurs.

J'adresse aussi mes remerciements à *Fabrice Dumas* et au département de mécanique pour leurs prestations, pour leur disponibilité et leur travail de qualité lors de la fabrication de la mécanique du prototype.

Merci à *Richard Vandaele* et *Thierry Tixier* pour leur expertise, leurs conseils et leur guidance par rapport aux conceptions hardwares que j'ai pu, grâce à eux, mener à bien. J'aimerais également leur dire à quel point j'ai apprécié leur grande disponibilité.

Un merci tout particulier à *Stephane Witzmann* qui m'a aidé et assisté lors des manipulations et pour le développement des softwares.

Merci aussi à *Nicolas Roudel*, *Ahmed Benzerouk*, *Fabio Dias* et *Pierre Avrazini*.

Je tiens à remercier les cinq stagiaires que j'ai eu la chance d'encadrer et qui m'ont beaucoup assisté dans mes travaux : *Nicolas Lopes*, *Adrien Thiers*, *Nabil Aidoudi*, *Deborah Cassiaux* et *Cong Li*.

Enfin, je tiens à remercier l'intégralité du personnel du laboratoire, pour son accueil et son soutien général.

# Acronymes

1D 1 Dimension

2D 2 Dimensions

3D 3 Dimensions

AAA Adéquation Algorithme Architecture

AIT Austrian Institute of Technology

ASIC Application Specific integrated circuit

CCD Charge Coupled Device

CMOS Complementary Metal Oxide Semiconductor

DMP Dual Mirror-Pyramid

DoG Difference of Gaussian

DREAM Research on Embedded Architecture and Multi-Sensor

DSP Digital signal processing

FAST Features from Accelerated Segment Test

FIFO First In First Out

FOV Field Of View

FPGA Field Programmable Gate Array

FPS Frame Per Second

FPN Fixed Pattern Noise

GPS Global Position System

HDL Hardware Description Language

HDR High Dynamic Range

IMU Inertial Measurement Unit

IP Institut Pascal

JTAG Joint Test Action Group

kO kilo Octets

LASMEA Laboratoire des Sciences et Matériaux pour l'Électronique, et d'Automatique

LE Logic Element

LM Levenberg Marquardt

LSE Least Squares Estimate

LVDS Low Voltage Differential Signaling

MFC Multi Functional Camera

MHz Méga Hertzs

MO Méga Octets

Mp Méga pixels

PID Proportionnel Intégral Dérivé

PLL Phase Locked Loop

RAM Random Access Memory

RANSAC RANdom SAmple Consensus

PAVIN Plateforme d'Auvergne pour les Véhicules INtelligents

RGB Red Green Blue

ROI Region Of Interest

RPM Rotations Par Minute

RPS Rotations Par Seconde

SEAMOVES Sensor Enabling Autonomous Motion by Optimized Visual Environment Sensing

SFM Structure From Motion

SIFT Scale Invariant Feature Transform

SIMD Single Instruction Multiple Data

SLAM Simultaneous Localization And Mapping

SLAN Simultaneous Localization And Navigation

SOC System On Chip

SOPC System On Programmable Chip

SRAM Static Random Access Memory

SSD Sum of Squared Differences

SURF Speeded Up Robust Features

SUSAN Smallest Univalue Segment Assimilating Nucleus

SVP Single View Point

TRT Thales Research and Technology

TOSA Thales Optronics SAS

UAV Unmanned Aerial Vehicle

USB Universal Serial Bus

VHDL Very High Speed Integrated Circuit (VHSIC) Hardware Description Language

VHSIC Very High Speed Integrated Circuit

VLSI Very-large-scale integration

VR Virtual Reality

WAAC Wide Angle Airborne Camera

WAOSS Wide Angle Optoelectronic Stereo Scanner

ZNCC Zero-Mean Normalized Cross-Correlation

# Table des matières

# CHAPITRE 1

# Introduction Générale

## 1.1 PRINCIPE DE LA VISION PANORAMIQUE

Depuis la naissance de la photographie au début du XIX ème siècle, beaucoup d'efforts ont été portés sur la reproduction des capacités de l'œil humain. L'invention du modèle *sténopé* de formation des images n'est autre qu'une simplification des phénomènes optiques ayant lieu dans un œil humain. Puis les technologies mises au point pour la capture des couleurs présentent une grande analogie avec le fonctionnement des capteurs de couleurs primaires de l'œil humain. Enfin, la stéréovision vise à reproduire la perception des profondeurs, ou le relief, en s'inspirant encore une fois de la stéréovision humaine.
Cela dit, l'œil humain n'est capable de capturer qu'un petit champ de l'espace qui l'entoure, et il en va de même pour les caméras et appareils photos classiques. Une perception omnidirectionnelle est une forme de vision comportant un champ visuel capable de couvrir toutes les directions. Créer des systèmes de vision avec un champ de vision plus large que celui de l'homme s'avère très attrayant. Cette piste de recherche a été intensivement creusée depuis l'invention de la photographie. Ainsi, dès les années 1840 naissent les premières photographies panoramiques à peine vingt ans après la naissance de la photographie elle même. Les premières vues panoramiques ne sont qu'un simple assemblage de photographies, mais rapidement, de premiers systèmes capturant des clichés à grand champ de vision voient le jour. Aujourd'hui encore, de nombreuses méthodes existent et aucune n'a réellement conquis la globalité du marché car chacune d'entre elles apporte un compromis différent qui ne présente pas toutes les qualités simultanément. C'est pour cette raison que cette partie de la vision est si peu employée dans des applications de robotique mobile où les informations visuelles sont exploitées pour localiser et diriger un robot. Cela dit, le fait de percevoir l'environnement dans toutes les directions est un avantage certain dans le cadre de la problématique de localisation. L'enjeu est donc important et les difficultés techniques sont aussi grandes.
Comme nous le voyons plus loin, il existe principalement quatre grandes catégories de méthodes :

- L'utilisation de grands angles,
- L'utilisation de miroirs,
- L'utilisation de plusieurs caméras,
- L'utilisation de mécaniques rotatives.

Enfin, une multitude de systèmes *hybrides* combinant ces différentes méthodes existe également. En général, il est difficile d'obtenir un système qui est capable de prodiguer une

vision à 360° à grande résolution sans distorsions et avec un débit suffisamment grand pour faire de la robotique mobile et de contrôler des robots allant à des allures de 10 mètres par seconde par exemple.

## 1.2 CONTEXTE DE LA THÈSE

Cette thèse s'inscrit dans le cadre du projet EURIPIDES Sensor Enabling Autonomous Motion by Optimized Visual Environment Sensing (SEAMOVES) qui débuta en octobre 2010 et s'acheva en novembre 2013. Ce projet d'un budget de 12 million d'euros rassemblait les entreprises suivantes :

- Thales Research and Technology (TRT),
- Institut Pascal (IP),
- Austrian Institute of Technology (AIT) ,
- Thales Optronics SAS (TOSA),
- ECA Robotics,
- Aldebaran.

Il avait pour but le développement d'un système de vision panoramique destiné à la navigation de véhicules en environnement urbain. L'architecture de ce système était basée sur un capteur d'images développé par AIT et qui proposait une approche novatrice de vision. Dans le cadre du projet, l'Institut Pascal avait plusieurs responsabilités ; le développement d'une architecture électronique de contrôle et de traitement pour le prototype de caméra, et la mise au point d'algorithmes de calibrage et de navigation destinés à être utilisés avec le capteur développé. Enfin, le développement d'un système complet de vision panoramique était aussi nécessaire afin de permettre l'aboutissement des deux premières tâches.
Cette thèse a aussi pris place au sein du laboratoire Laboratoire des Sciences et Matériaux pour l'Électronique, et d'Automatique (LASMEA) de l'Institut Pascal, et plus précisément dans l'équipe de recherche Research on Embedded Architecture and Multi-Sensor (DREAM).

## 1.3 BUT DES RECHERCHES ET PROBLÉMATIQUE

Les buts des recherches effectuées lors de cette thèse étaient au nombre de cinq :

### ÉTUDE DES TECHNIQUES PANORAMIQUES

Étude des différentes méthodes qui permettent de capturer des images panoramiques dans le domaine du numérique. Comparaison et classification de ces techniques en vue du développement d'une nouvelle caméra rotative.

### RÉALISATION D'UN SYSTÈME PANORAMIQUE ROTATIF

Réalisation d'un prototype de caméra panoramique rotative capable de capturer des images à 360 degrés. La fabrication de ce prototype s'est faite après l'étude des systèmes

existants afin de corriger quelques unes de leurs limitations technologiques et pour améliorer leurs performances.

IMPLÉMENTATION D'UN ALGORITHME DE TRAITEMENT

Afin de réduire le débit d'information provenant des capteurs panoramiques, une approche d'extraction de points d'intérêt a été mise en œuvre. Après une étude des différentes primitives (points d'intérêt dans les images) nous avons sélectionné l'une d'entre elle et l'avons implémentée directement dans l'architecture de traitement de notre système panoramique. L'extraction des primitives permet à la fois d'isoler les informations importantes et nécessaires au fonctionnement des algorithmes de localisation et de reconstruction, mais aussi de grandement diminuer la quantité de données à transmettre.

ÉTUDE ET MODÉLISATION GÉOMÉTRIQUE

Étude géométrique des systèmes panoramiques rotatifs afin de comprendre et d'analyser la formation des images en leur sein. Modélisation de ce processus de création d'images et comparaison des différences de propriétés des résultats d'un système à l'autre.

DÉVELOPPEMENT D'UNE MÉTHODE DE CALIBRAGE ET DE RECONSTRUCTION 3D

Après avoir étudié les méthodes de calibrage existantes, nous proposons une nouvelle méthode adaptée aux systèmes cylindriques. Nous avons également développé un algorithme de localisation et de reconstruction 3 Dimensions (3D) Structure From Motion (SFM) qui a été testé et validé sur des trajectoires de test dont les déplacements étaient connus.

Les travaux effectués lors de cette thèse peuvent donc être regroupés selon deux axes distincts. Un axe architecture regroupant la réalisation d'une nouvelle caméra panoramique avec architecture de traitement embarquée et implémentation d'algorithmes de traitement. Un axe vision regroupant la modélisation géométrique des caméras rotatives, et le développement de méthodes de calibration, de calcul de pose et de reconstruction 3D. Ces deux axes représentent les deux grandes contributions de cette thèse.

## 1.4 STRUCTURE DU MANUSCRIT

### 1.4.1 *Chapitre 2*

Le chapitre 2 présente l'étude de l'état de l'art des systèmes capturant des images panoramiques de nos jours. Chaque catégorie de méthodes y est étudiée et classée selon son genre, la question de la modélisation est aussi abordée afin de décrire la formation des images dans chaque cas. Les différentes techniques sont comparées les unes aux autres selon leurs avantages et leurs inconvénients en vue d'une application de type SLAM.

### 1.4.2 *Chapitre 3*

Après avoir brièvement introduit nos choix architecturaux et la structure de notre système, nos présentons la problématique d'implémentation de traitements sur le flot avec l'extraction de primitives dans le chapitre 3. Nous présentons l'étude relative aux extracteurs de primitives et nous montrons comment la primitive de *Harris & Stephens* a été choisie. Son principe et son implémentation dans notre système sont détaillés, une adaptation a éga-

lement été nécessaire pour l'implémentation sur l'architecture ciblée. Les résultats sont ensuite analysés.

### 1.4.3 *Chapitre 4*

Le chapitre 4 présente une analyse des systèmes à projections cylindriques. Les modélisations existantes sont analysées et notre approche est ensuite présentée. Nous montrons notre contribution qui réside à la fois dans une modélisation stéréoscopique plus complète et plus simple et nous débouchons vers la géométrie épipolaire et les choix architecturaux retenus pour notre système.

### 1.4.4 *Chapitre 5*

Le chapitre 5 présente l'étude des techniques de calibration existantes pour les systèmes panoramiques rotatifs et notre approche y est décrite et développée. Un algorithme de calcul de pose et de reconstruction 3D est ensuite proposé, puis les résultats de son utilisation et de sa validation dans notre environnement de test sont présentés.

### 1.4.5 *Conclusion*

Enfin nous faisons un bilan sur les différentes contributions, puis sur les perspectives de développement et de recherches futures qui ont été ouvertes dans la conclusion.

# CHAPITRE 2

# Imagerie panoramique

## 2.1 INTRODUCTION

La vision panoramique est un domaine de recherche qui présente une grande diversité technologique. En effet, on peut énumérer plus d'une dizaine de façons de capturer des images panoramiques à plus ou moins grand champ de vision (Field Of View (FOV)). Parmi ces méthodes, quatre sous-ensembles émergent :

- Les méthodes employant des optiques à large FOV,
- Les méthodes employant des miroirs,
- Les méthodes employant plusieurs caméras,
- Les méthodes employant une mécanique rotative.

Dans ce premier chapitre, notre but est d'étudier les différentes architectures de caméra pour les classifier et les comparer. Avant cela, il est nécessaire de rappeler l'architecture d'une caméra classique et d'expliquer le modèle de base de formation des images. Ainsi, nous posons les définitions et les notations qui sont utilisées pendant tout le reste de ce manuscrit. Par la suite nous étudions chaque type de système panoramique en expliquant les améliorations apportées au modèle standard. Enfin nous nous attardons sur la catégorie qui nous intéresse dans le cadre de cette thèse, à savoir les systèmes panoramiques à caméra linéaire rotative. La conclusion donne lieu à une comparaison des avantages et inconvénients de chaque méthode.

## 2.2 MODÈLE STANDARD

### 2.2.1 *Architecture*

Une caméra numérique classique, ou aussi appelée dans cette thèse une caméra conventionnelle, est un système de vision ayant pour but de capturer des images à champ de vision restreint (approximativement autour de 60 degrés horizontalement et 40 degrés verticalement).

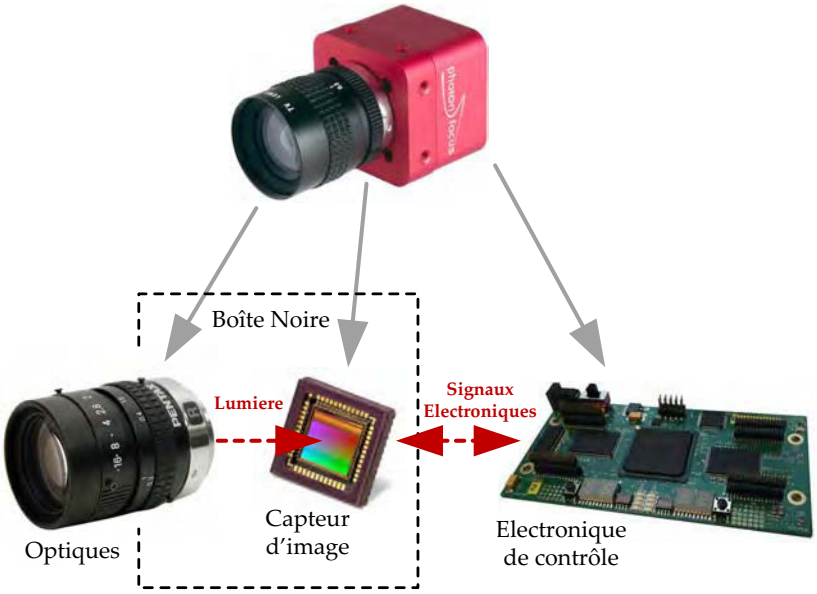


FIGURE 2.1: Architecture d'une caméra classique : composée d'une optique, d'une rétine et d'une électronique de traitement.

Une caméra numérique est généralement composée de trois principales parties comme l'illustre la figure 2.1 :

- Un objectif composé d'un ensemble de lentilles optiques,
- Un capteur d'image matriciel. Il est composé d'une matrice de photo-récepteurs. Il en existe de deux technologies différentes : Complementary Metal Oxide Semiconductor (CMOS) ou Charge Coupled Device (CCD),
- Une électronique de contrôle et éventuellement de traitement.

Les rayons lumineux qui traversent l'optique sont projetés sur le capteur qui se trouve dans une chambre noire. Le capteur convertit l'information lumineuse en information électronique qui est ensuite traitée et conditionnée par l'électronique de traitement. Les images sous forme de signaux électroniques sont ensuite transmises à un périphérique d'affichage ou à une autre unité de traitement. Cette architecture classique est commune à toutes les caméras et est également la base des caméras plus complexes ou à celles servant à la vision panoramique.

### 2.2.2 *Modélisation*

Avec les caméras classiques, la formation des images est souvent décrite via le modèle sténopé. Ce modèle, aussi appelé modèle *pinhole* est une simplification de la réalité. En effet la caméra est assimilée à une boîte noire percée d'un trou d'épingle ou la lumière s'engouffre. Une image se forme à l'intérieur sur la paroi opposée à celle où se trouve le trou. Basiquement, un objectif de caméra peut être assimilé au trou d'épingle du modèle sténopé.
La figure 2.2 décrit le processus de formation des images avec le modèle sténopé :

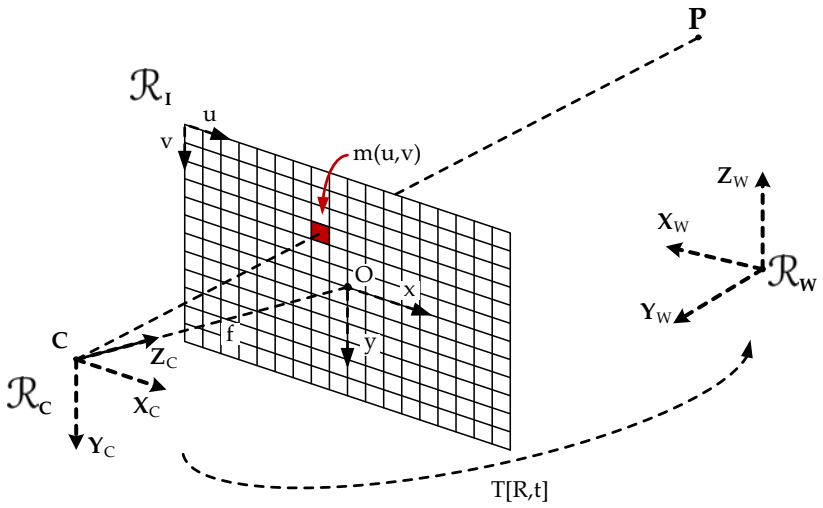


FIGURE 2.2: Modèle sténopé : Modèle géométrique de la formation des images dans une caméra classique

Afin de comprendre le processus de formation de l'image sur la rétine, il est nécessaire de le décomposer en étapes successives. Pour ce faire, il est d'usage de définir des repères géométriques pour les différentes étapes. Dans notre cas, on peut distinguer trois repères géométriques distincts :

- Le repère monde $\mathcal{R}_W$ qui représente le système de coordonnées dans le référentiel monde extérieur à la caméra.
- Le repère de la caméra $\mathcal{R}_C$ représente le système de coordonnées de la caméra. Le centre de ce repère est le centre optique de la caméra, l'axe $Z_c$ est l'axe optique et le plan formé par les axes $X_C$ et $Y_C$ est parallèle au plan image.
- Le repère $\mathcal{R}_I$ est le repère image une fois la projection faite. Ce repère possède seulement deux dimensions et comporte une échelle en pixels.

Pour passer du repère monde $\mathcal{R}_W$ au repère caméra $\mathcal{R}_C$, il faut appliquer la transformation $T[R,t]$ qui est composée d'une translation $t$ et d'une rotation $R$. En d'autres termes, la transformation $T$ décrit le mouvement du repère $\mathcal{R}_C$ au repère $\mathcal{R}_W$. Ensuite, la projection sur la rétine transforme les coordonnées 3D des points de l'espace en coordonnées 2 Dimensions (2D) sur la rétine et donc dans l'image. Le point P se retrouve donc projeté en

$m(u,v)$ et obtient des coordonnées en pixels. Le point O est appelé *Point Principal* et est le points d'intersection entre l'axe $Z_C$ du repère caméra et le plan image, il est la projection du centre optique sur l'image. La distance focale dépend uniquement de l'optique utilisée et est la distance entre le centre optique et le point principal. Plus la distance focale est grande, plus le champ de vision de la caméra sera étroit.

### 2.2.3 *Mise en équations*

Le point P a pour coordonnées homogènes $(X_W, Y_W, Z_W, 1)^T$ dans le repère monde $\mathcal{R}_W$. La transformation T appliquée à ces coordonnées donne les coordonnées $(X_C, Y_C, Z_C, 1)^T$ du point P en fonction du repère $\mathcal{R}_C$ selon l'équation :

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{2.1}$$

A partir des coordonnées $(X_C, Y_C, Z_C, 1)^T$, la projection sténopée est appliquée de la manière suivante :

$$s. \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \cdot Q \cdot \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{2.2}$$

C'est à dire :

$$s. \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & s_{uv} & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \tag{2.3}$$

Avec :

$$A = \begin{bmatrix} k_u & s_{uv} & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad Q = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.4}$$

$$A \cdot Q = \begin{bmatrix} k_u & s_{uv} & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f \cdot k_u & f \cdot s_{uv} & u_0 \\ 0 & f \cdot k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & f \cdot s_{uv} & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = K \tag{2.5}$$

Le paramètre $s$ est le facteur d'échelle et K définit les paramètres intrinsèques qui décrivent l'optique et le capteur d'image en une seule formule.

- $s_{uv}$ traduit la non-orthogonalité potentielle des lignes et des colonnes de cellules électroniques photosensibles qui composent le capteur de la caméra. Tout au long de ces travaux, ce paramètre est négligé.
- $u_0$ et $v_0$ sont les coordonnées du point principal dans l'image (la projection du centre optique).
- $f$ est la distance focale.
- $(x_u, x_v)$ sont les dimensions métriques d'un pixel.
- $(k_u, k_v)$ sont les facteurs d'échelle (respectivement horizontal et vertical) en pixels par millimètres. On a $k_u = \frac{1}{x_u}$ et $k_v = \frac{1}{x_v}$.
- $\alpha_u$ et $\alpha_v$ représentent le changement d'échelle entre les coordonnées métriques et les coordonnées en pixels. On a $\alpha_u = f.k_u$ et $\alpha_v = f.k_v$.

En bilan, la projection complète est exprimée comme suit :

$$s. \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{2.6}$$

Ce qui donne après développement et en retirant le facteur d'échelle $s$ :

$$\begin{cases} u = \alpha_u \left( \frac{R_{11}.X_W + R_{12}.Y_W + R_{13}.Z_W + t_x}{R_{31}.X_W + R_{32}.Y_W + R_{33}.Z_W + t_z} \right) + u_0 \\ v = \alpha_v \left( \frac{R_{21}.X_W + R_{22}.Y_W + R_{23}.Z_W + t_y}{R_{31}.X_W + R_{32}.Y_W + R_{33}.Z_W + t_z} \right) + v_0 \end{cases} \tag{2.7}$$

### 2.2.4 *Méthodologie de comparaison*

Des modifications de l'architecture classique permettent d'obtenir des champs de visions plus importants. Pour pouvoir comparer les différentes méthodes panoramiques, les trois axes de comparaison suivants sont utilisés :

- Architecture matérielle,
- Géométrie et propriétés de la projection des images,
- Nécessité de traitements du flot d'images.

HARDWARE :

L'architecture classique en trois éléments est la configuration minimale. En ajoutant de nouveaux éléments ou en multipliant le nombre de caméras, la complexité augmente. Ainsi, les architectures comportant un élément supplémentaire sont plus difficiles à mettre en œuvre. Les systèmes comportant des parties mécaniques sont considérés comme les plus complexes.
Pour la comparaison des caractéristiques comme la résolution et la cadence d'acquisition, nous partons du principe que le même capteur est utilisé dans les différentes architectures. La comparaison est faite en fonction des caractéristiques de ce capteur de *référence*.
Enfin, la capture de l'image peut être globale ou progressive, c'est à dire de type *Global Shutter* ou *Rolling Shutter* respectivement, le capteur de référence étant de type *Global Shutter*.

GÉOMÉTRIE :

De l'architecture employée découlent plusieurs propriétés géométriques de formation des images. Ainsi l'image panoramique peut avoir :

- Des projections géométriques différentes,
- Un ou plusieurs points de vues (centres de projections),
- Une densité de résolution homogène ou non,
- Un champ de vision plus ou moins grand mesuré en degrés.

TRAITEMENT :

Enfin, en fonction du type de projection et de l'éventuel besoin d'assembler des images ensembles, il peut être requis d'effectuer des traitements sur les images panoramiques produites :

- Un traitement d'assemblage d'images en une image panoramique finale,
- Un traitement de reprojection pour changer la géométrie de l'image.

BILAN DE LA MÉTHODE :

En bilan, le tableau 2.1 est utilisé pour évaluer chaque méthode panoramique.

| **Caractéristiques** | **Méthode analysée** |
|---|---|
| Architecture | Simple à Complexe |
| Résolution | Comparée à la référence |
| Cadence | Comparée à la référence |
| Capture | *Global* ou *Rolling Shutter* |
| Type d'images | Planaires, Cylindriques, Hémisphériques... |
| Centre de projection | Unique ou multiples |
| Champ de vision | En degrés |
| Densité de résolution | Homogène ou non |
| Reprojection | Nécessaire ou non |
| Stitching | Nécessaire ou non |

TABLE 2.1: Les différentes caractéristiques qui sont analysées lors de la comparaison des différentes méthodes de captures panoramiques.

## 2.3 SYSTÈMES *fisheyes*

### 2.3.1 *Architecture*

Une caméra *Fisheye*, est un système de vision ayant un champ de vision beaucoup plus large que celui d'une caméra classique. Ce champ peut atteindre 180 degrés dans deux directions et ainsi capturer une hémisphère complète. La Figure 2.3 montre une photo prise avec un de ces systèmes.

FIGURE 2.3: Exemple d'image hémisphérique capturée par une caméra *Fisheye*.

Une caméra *Fisheye* possède la même architecture que celle décrite sur la figure 2.1. La seule différence réside dans les propriétés de l'optique employée, qui permettent d'obtenir

le large champ de vision. La distance focale d'un objectif étant l'inverse proportionnelle du champ de vision, une grande focale donne un champ de vision étroit et inversement, une focale courte donne un grand champ de vision. En diminuant cette distance, on peut donc obtenir un champ de vision de plus en plus grand. Cela dit, dans un système à projection centrale (comme le modèle sténopé), cela ne peut être fait que jusqu'à une certaine mesure. En effet, avec un champ de vision de 180 degrés, un rayon lumineux d'une incidence de 90 degrés serait projeté sur le plan image à l'infini par rapport au point principal. Pour avoir une projection hémisphérique complète sur un format image défini, le modèle sténopé ne suffit donc pas. Un autre modèle de projection est alors nécessaire [121]. Les optiques *Fisheyes* sont d'ailleurs des optiques spéciales qui ne respectent pas le modèle sténopé.
La projection *Fisheye* est basée sur le principe que la distance entre le point image et le point principal sur la rétine est linéairement dépendant de l'angle d'incidence du rayon lumineux correspondant à ce point image. La figure 2.4 compare cette projection avec la projection sténopé vue précédemment.

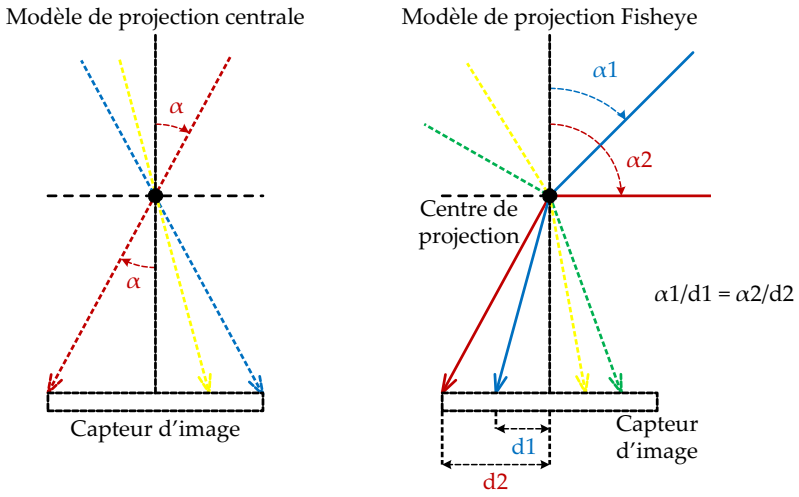


FIGURE 2.4: Comparaison de la projection centrale du modèle sténopé à gauche, avec la projection *Fisheye* à droite.

Les rayons lumineux sont donc réfractés dans la direction de l'axe optique.

### 2.3.2 *Modélisation*

Comme pour les autres systèmes, il n'existe pas de modélisation unique [138, 29], ici nous présentons l'une d'entre elle afin de se rendre compte de la différence avec le modèle sténopé standard.

Le modèle *Fisheye* est présenté en figure 2.5.

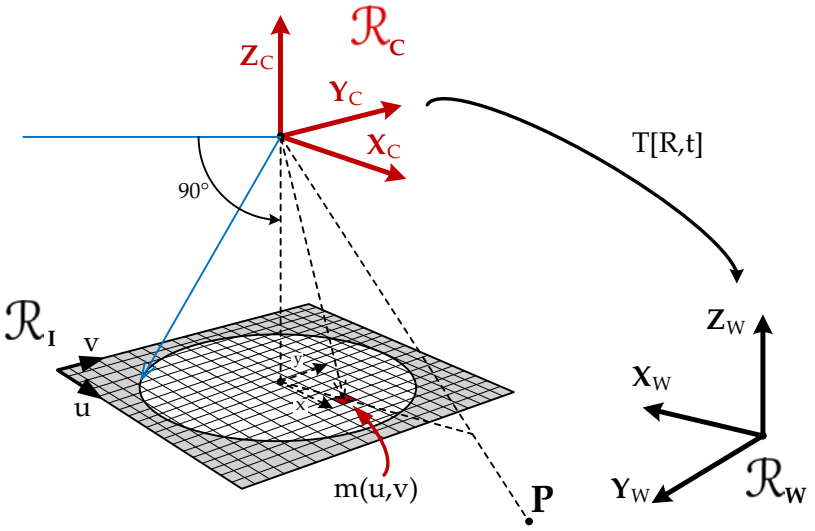


FIGURE 2.5: Modèle de projection *Fisheye*. Les rayons lumineux sont réfractés vers l'axe optique. Pour une optique *Fisheye* de 180 degrés, un rayon lumineux arrivant avec une incidence de 90 degrés (en bleu) est projeté sur le bord de l'image circulaire.

Comme pour le modèle sténopé, nous avons les trois mêmes repères, à savoir le repère monde $\mathcal{R}_W$, le repère caméra $\mathcal{R}_C$ et le repère image $\mathcal{R}_I$. La transformation $T[R,t]$ décrit le mouvement du repère caméra $\mathcal{R}_C$ au repère monde $\mathcal{R}_W$. On a donc toujours :

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{2.8}$$

Ce qui diffère, c'est la projection des objets selon le repère caméra sur l'image. En effet, nous ne sommes pas dans un cas qui respecte le modèle sténopé. C'est l'angle d'incidence des rayons lumineux qui définit la distance du point image au point principal. Une incidence de 90 degré a pour projection un point se trouvant an périphérie sur le cercle de l'image, comme illustré sur la figure 2.5. Ainsi on a le rapport suivant :

$$\frac{\alpha}{r} = \frac{90}{R} \tag{2.9}$$

Avec $\alpha$ l'angle d'incidence, $r$ la distance entre le point image et l'axe optique et $R$ le rayon du cercle image. Dans les systèmes *Fisheyes*, la distance focale est remplacée par le rayon $R$ en tant que facteur d'échelle. Sachant qu'on a le rapport suivant :

$$\frac{u}{v} = \frac{X_C}{Y_C} \tag{2.10}$$

Le modèle général est donné dans les travaux de Ellen Schwalbe [121], et l'expression des coordonnées image $m(u,v)$ d'un point $P(X_C, Y_C, Z_C)$ dans le repère $\mathcal{R}_C$ est donnée dans le système d'équation suivant :

$$u = \frac{\frac{2.R}{\pi}.\arctan[\frac{\sqrt{X_C^2+Y_C^2}}{Z_C}]}{\sqrt{\frac{Y_C^2}{X_C^2}+1}} + du + u_0 \qquad v = \frac{\frac{2.R}{\pi}.\arctan[\frac{\sqrt{X_C^2+Y_C^2}}{Z_C}]}{\sqrt{\frac{X_C^2}{Y_C^2}+1}} + dv + v_0 \tag{2.11}$$

Avec $u_0$ et $v_0$ les coordonnées du point principal dans l'image et $du$ et $dv$ les paramètres de modélisation de la distorsion des lentilles de l'objectif.

### 2.3.3 *Exemples*

Cette technique de vision panoramique est largement répandue et populaire grâce à sa simplicité de mise en œuvre. En effet, seul l'optique change par rapport à une caméra classique. Cela dit, la densité de résolution n'est pas constante sur les images produites, comme vu en Figure 2.3. La périphérie de l'image comporte une résolution inférieure à celle du centre de l'image pour un champ de capture similaire. Des solutions à ce problème existent [135] avec par exemple l'utilisation d'optiques appelées *Panomorph* ou la résolution en périphérie est augmentée grâce à une modification du jeux de lentilles. Pour obtenir une image planaire (comme celles produites par les modèles sténopés), il faut utiliser des algorithmes de rectification [64, 63] et cela est couteux en temps et en ressources de calcul lorsque le traitement est fait en temps réel.

### 2.3.4 *Conclusion*

En bilan, les caractéristiques de la méthode *Fisheye* sont résumées dans le tableau 2.2

Ces systèmes sont très similaires aux caméras classiques dans leurs propriétés. L'image est projetée sur un capteur classique, ce qui fait que la cadence et le mode d'acquisition ne sont pas changés par l'architecture. La résolution peut être légèrement inférieure due à la présence de zones de pixels non utilisées (zones en dehors du cercle projeté de l'image). Cela est parfois évité quand le capteur a une taille inférieure à celle du champ de projection. D'une manière générale, l'utilisation d'un seul capteur d'image pour capturer un champs de vision hémisphérique donne des résultats non satisfaisant en terme de résolution. En effet on obtient un très faible rapport résolution/champ de vision, et donc peu d'informations sur la scène observée.
La projection est centrale donc les perspectives sont respectées, cependant nous obtenons une image hémisphérique avec des distorsions et une résolution non homogène. Des traitements sont donc souvent nécessaires pour pouvoir en exploiter les images.

| **Caractéristiques** | ***Fisheye*** |
|---|---|
| Architecture | Simple |
| Résolution | $\leqslant$ Référence |
| Cadence | = Référence |
| Capture | *Global Shutter* |
| Type d'images | Hémisphériques |
| Centre de projection | Unique |
| Champ de vision | $\leqslant 360° \times 180°$ |
| Densité de résolution | Fovéale + Distorsions |
| Reprojection | Nécessaire |
| Stitching | Non nécessaire |

TABLE 2.2: Résumé des caractéristiques de la méthode *Fisheye*

## 2.4 SYSTÈMES CATADIOPTRIQUES

### 2.4.1 *Architecture*

Les caméras catadioptriques sont caractérisées par l'association d'une optique avec un miroir. Typiquement il s'agit d'une caméra classique filmant un miroir qui peut être de plusieurs natures. La réflexion de l'environnement sur le miroir est projetée sur le capteur de la caméra. Ainsi, en fonction de la forme du miroir, on obtient un champ de vision plus ou moins large. La Figure 2.6 présente un exemple de caméra catadioptrique employant un miroir hyperbolique.

Plusieurs type de miroirs sont possibles [103], d'une manière générale, les miroirs utilisables se divisent en deux catégories ; ceux ayant un centre de projection unique (Single View Point (SVP)) [9], et ceux en ayant plusieurs. Dans la catégorie des systèmes SVP, Baker et Nayar ont montrés en [8] que les type de miroirs qui vérifient cette condition sont les suivants :

- Parabolique,
- Hyperbolique,
- Ellipsoïdal,
- Planaire.

Les miroirs coniques, sphériques et les autres ne font pas partie de cette catégorie. La propriété SVP [7] est recherchée car elle donne la possibilité d'avoir une reconstruction des

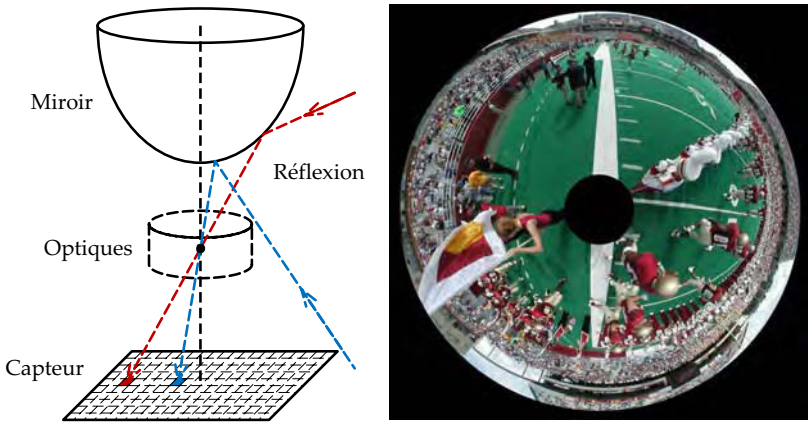


FIGURE 2.6: (a)Architecture d'une caméra catadioptrique. Elles sont basées sur l'association d'une optique avec un miroir. Dans cet exemple il s'agit d'un miroir hyperbolique. (b) Exemple d'image capturée avec une de ces caméras.

images panoramiques sans distorsion et d'obtenir des projections plus familières comme la projection planaire ou la projection cylindrique. Nous ne parlerons que de la catégorie SVP ici. Les capteurs catadioptriques qui ne possèdent pas de focale unique (non-SVP) sont moins utilisés pour cette raison.
Une autre caractéristique importante est celle des systèmes dit *Paracatadioptriques* [48] qui sont des caméras avec une optique *Orthographique* et un miroir paraboloïde.

### 2.4.2 *Modélisation*

Due à la diversité des types de miroirs employés, il existe aussi plusieurs modèles possibles. Certains ne s'appliquent qu'à un seul type de systèmes catadioptriques comme par exemple ceux comportant un miroir conique [23, 72]. D'autres sont des modèles généralisés qui peuvent s'appliquer à plusieurs types de miroirs. Concernant les systèmes SVP, plusieurs modélisations peuvent aussi être utilisées [40][132].

Alors que certaines d'entre elles se basent sur le fait que la surface réflective est connue et que la caméra soit paracatadioptrique (constituée d'un miroir paraboloïde et d'une lentille *orthographique* [48]). D'autres sont basées sur le principe d'un unique point SVP et utilisent un modèle équivalent de projection sur une sphère [37, 36]. C'est le cas de celui que nous allons présenter ici [30]. Il a été publié pour la première fois par Geyer et Daniilidis en 2000 [38]. La figure 2.7 présente ce modèle.

Le miroir utilisé, qu'il soit parabolique, hyperbolique, ellipsoïde ou planaire, est modélisé comme une sphère virtuelle ayant pour centre le centre de projection du miroir réel. Quatre repères sont nécessaires pour décrire les différentes transformations :

- Le repère monde $\mathcal{R}_W$ qui représente le système de coordonnées dans le référentiel monde extérieur à la caméra.
- Le repère de la sphère $\mathcal{R}_S$ représente le système de coordonnées en fonction de la sphère. Son centre est le centre de la sphère et se situe sur l'axe optique. L'axe $Z_c$ est

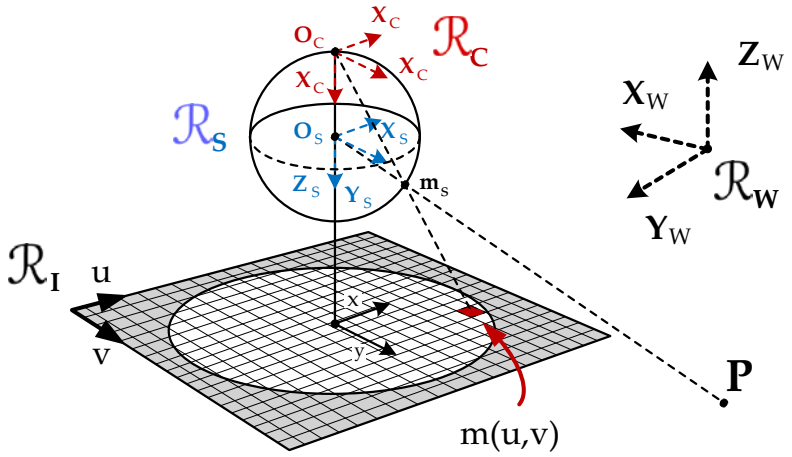


FIGURE 2.7: Modélisation générale de Geyer et Daniilidis [38]

confondu avec ce dernier et le plan formé par les axes $X_S$ et $Y_S$ est parallèle au plan image.

- Le repère de la caméra $\mathcal{R}_C$ représente le système de coordonnées en fonction de la caméra. Le centre du repère est le centre optique de la caméra, l'axe $Z_c$ est l'axe optique et le plan formé par les axes $X_C$ et $Y_C$ est parallèle au plan image.
- Le repère $\mathcal{R}_I$ qui est le repère image une fois la projection faite. Ce repère possède seulement deux dimensions et est à l'échelle des pixels.

La sphère a pour rayon $R = 2 \times h$ sachant que $h$ est la distance entre le centre de projection $O_S$ et le sommet du miroir paraboloïde. Lorsque nous avons un miroir planaire, cette distance est nulle tout comme le rayon de la sphère virtuelle. Le point 3D $P(X_W, Y_W, Z_W)$ est projeté sur la sphère en un point $m_S$ situé à l'intersection de la droite $(O_S P)$ et de la surface de la sphère. On obtient les coordonnées suivantes :

$$\begin{cases} X_S = \Lambda \times X_W \\ Y_S = \Lambda \times Y_W \\ Z_S = \Lambda \times Z_W \end{cases} \tag{2.12}$$

De plus, le point $m_S$ se situant sur la sphère, on a aussi :

$$X_S^2 + Y_S^2 + Z_S^2 = R^2 \tag{2.13}$$

Ensuite, le point $m_S$ est projeté dans l'image par un modèle sténopé virtuel ayant pour centre le point $O_C$ Le plan image est perpendiculaire à la droite $(O_C O_S)$ et est appelé image planaire catadioptrique. Cette projection est donnée par les équations :

$$\frac{X_S}{\xi - Z_S} = \frac{x}{\xi + \phi} \qquad \frac{Y_S}{\xi - Z_S} = \frac{y}{\xi + \phi} \tag{2.14}$$

Avec $\xi$ étant le paramètre qui détermine le type de miroir utilisé, il est égal à la distance $(O_C O_S)$. On a les conditions suivantes :

- Si $\xi = 1$, le miroir est une paraboloïde,
- Si $0 < \xi < 1$, le miroir est une hyperboloïde,
- Si $\xi = 0$, Le miroir est un plan.

Quant à $\psi$, il est un paramètre qui est fonction d'$\xi$ et de l'échelle. Enfin, les coordonnées à l'échelle des pixels sont données par les équations suivantes [86] :

$$u = \frac{\alpha_u.(\xi + \phi).X_W}{\xi.\sqrt{X_S^2 + Y_S^2 + Z_S^2} - Z_W} + u_0 \qquad v = \frac{\alpha_v.(\xi + \phi).Y_W}{\xi.\sqrt{X_S^2 + Y_S^2 + Z_S^2} - Z_W} + v_0 \tag{2.15}$$

### 2.4.3 *Exemples*

Les systèmes catadioptriques sont largement employés. En effet ils possèdent un grand avantage de simplicité de mise en œuvre et leur champ de vision est plus adapté à la robotique mobile. Il sont populaires dans les applications de type SLAM et on les retrouve sur divers robots mobiles, sur véhicules routier en [120], des robots mobiles en [75], sur des drones (Unmanned Aerial Vehicle (UAV)) [122], et même sur des voiliers [108]. La stéréovision est aussi exploitée [110][85].

D'autres travaux proposent un désanamorphose des images afin d'obtenir des images à projection planaire, cylindriques ou même cubique [65][130]. Cette étape appelée *Unwrapping* nécessite une calibration [139] afin de reprojeter les images.

### 2.4.4 *Conclusion*

Les caractéristiques sont résumées dans le tableau 2.3.

On a ici des propriétés très similaires à celles des caméras *Fisheyes*. L'emploie d'une caméra classique a pour conséquence de donner des résolutions, des fréquences d'acquisition et un mode de capture similaire aux caméras classiques. Comme pour les *Fisheyes*, c'est l'ajout d'un élément extérieur à la caméra qui lui donne la possibilité d'élargir son champ de vision.
En ce qui concerne la géométrie, la projection est toujours centrale mais la résolution n'est toujours pas homogène (on obtient une résolution de type fovéale inversée, c'est à dire une densité de résolution plus importante en périphérie du capteur). De plus, les distorsions radiales et tangentielles sont importantes et une reprojection de l'image est souvent nécessaire pour obtenir des images cylindriques ou d'un autre format plus exploitable.
En bilan, nous avons les mêmes problèmes que ceux posés par les systèmes *Fisheyes*, une résolution non homogène et d'une manière générale insuffisante pour nos besoins car elle est limitée par l'utilisation d'un unique capteur d'images.

| Caractéristiques | Catadioptrique SVP |
|---|---|
| Architecture | Intermédiaire (mirroir) |
| Résolution | < Référence |
| Cadence | = Référence |
| Capture | *Global Shutter* |
| Type d'images | Sphériques |
| Centre de projection | Unique |
| Champ de vision | $360° \times 90°$ |
| Densité de résolution | Fovéale inversée |
| Reprojection | Nécessaire |
| Stitching | Non nécessaire |

TABLE 2.3: Résumé des caractéristiques des méthodes catadioptriques SVP.

## 2.5 SYSTÈMES POLYDIOPTRIQUES

### 2.5.1 *Architecture*

Le fait d'utiliser un seul capteur pour générer des vues panoramiques, est un facteur très limitant, en particulier pour la résolution. L'emploi de plusieurs caméras peut multiplier rapidement la résolution totale des panoramas. Cette méthode est appelée polydioptrique et est illustrée sur la figure 2.8.

Sur la figure, cinq caméras sont positionnées de manière à couvrir un champ de vision horizontal de 360 degrés. On capture cinq images différentes avec des zones de chevauchements. Ces multiples images planaires peuvent être assemblées en un unique panorama par des méthodes dites de *Mosaicing* ou de *Stitching*.

### 2.5.2 *Stiching Vs Mosaicing*

DÉFINITIONS :

Il existe deux mots pour qualifier le processus d'assemblage d'images en une seule plus grande ; *Mosaicing* et *Stitching*. Une distinction entre les deux termes est apportée dans le livre *Panoramic Imaging* publié en 2008 [52]. En effet, la méthode *Mosaicing* correspondrait au cas ou les images ont été capturées lors d'un mouvements non contrôlé ou à des positions non connues. Quant à la méthode *Stitching*, elle concernerait les mouvements contrôlés et les positions connues, comme par exemple sur la figure 2.8 où les caméras sont solidaire, ou encore lors de la rotation contrôlée d'une caméra. Cela dit, dans les autres travaux, il n'y a pas vraiment de distinction entre les deux termes et ils semblent être utilisés de manière interchangeable comme des synonymes. Nous utiliserons préférablement l'appellation *Stitching* car c'est celle qui présente le plus de similitude avec la

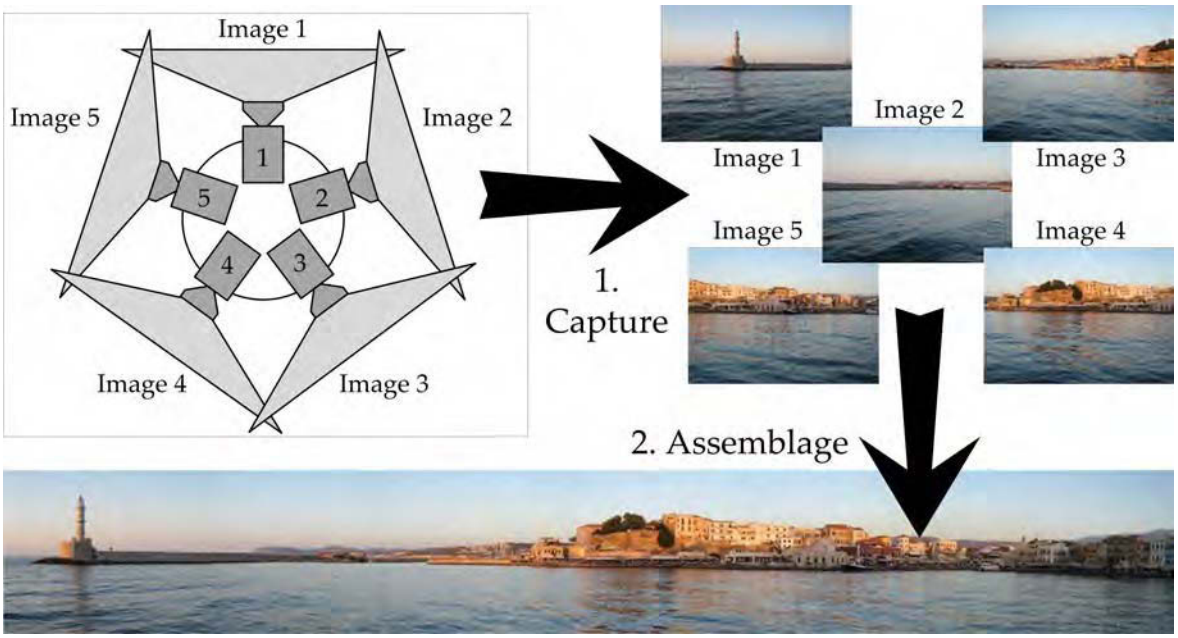


FIGURE 2.8: Exemple d'assemblage d'image panoramique à partir d'un système polydioptrique à 5 caméras.

constitution de panoramas à partir d'image provenant d'un système polydioptrique. Cela dit, dans certains travaux que nous aborderons, l'autre terme sera employé pour rester en accord avec le vocabulaire des auteurs.

APPLICATIONS :

L'assemblage d'images est utilisé dans différentes applications. L'assemblage de panoramas dans le cas de caméras polydioptriques n'en est qu'une parmi d'autres. On retrouve aussi le *Stitching* avec d'autres systèmes panoramiques que nous étudions plus loin, à savoir les systèmes pyramidaux et les systèmes rotatifs. D'une manière générale, le *Stitching* est très utilisé pour l'assemblage d'images provenant d'une unique caméra en mouvement, les téléphones mobiles dotés de caméra proposent aussi de plus en plus cette possibilité à l'utilisateur par exemple.

DESCRIPTION DE LA MÉTHODE :

Il n'existe pas une méthode unique pour l'assemblage d'images [25], beaucoup de travaux proposent des approches différentes et de bon résultats sont obtenus avec plusieurs d'entre elles. L'application dont il s'agit ici, se situe avec les caméras polydioptriques, c'est à dire typiquement un banc de caméras fixes les unes par rapport aux autres[82]. Dans l'ensemble des travaux sur le sujet, on peut voir certaines similitudes d'une approche à l'autre. Pour deux images à assembler, sachant qu'elles possèdent une zone de chevauchement, le *Stitching* se résume souvent à effectuer les opérations suivantes dans l'ordre :

- La détection de primitives invariantes dans les images, c'est à dire d'objets ou de zones facilement reconnaissable dans une autre image,
- L'appariement des primitives d'une image à l'autre,

- Le trie des paires et la suppression des paires erronées,
- L'estimation de la matrice d'homographie,
- L'assemblage des deux images selon les résultats obtenus d'appariement et de la matrice d'homographie.

L'appariement des primitives entre les deux images permet de reconnaitre la zone de chevauchement et de savoir à quel niveau les images s'assemblent. Comme l'étape d'appariement n'est pas infaillible, il est souvent nécessaire de chercher les faux appariement afin de les exclure. Ensuite, il est nécessaire de trouver la relation géométrique entre les deux images afin de reprojeter l'une d'entre elles et de pouvoir finalement les assembler. La matrice d'homographie [17], est une matrice $3 \times 3$ qui définit la transformation d'une image à l'autre. Elle est souvent exprimée selon l'équation :

$$\mu \times p_j = H \times p_i \qquad \mu . \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{2.16}$$

Où $H$ est la matrice homographique non singulière et inversible et $\mu$ le facteur d'échelle.

A partir de cette méthodologie, les algorithmes peuvent plus ou moins varier. Au niveau des primitives d'image (qui sont souvent des points particuliers dans les images), plusieurs méthodes de détection différentes peuvent être utilisées. Dans [140], une détection SIFT est employée alors que d'autres méthodes emploieront des primitives de type *Harris & Stephens* [76] ou Speeded Up Robust Features (SURF) [27][60][1]. Ensuite, beaucoup de travaux proposent d'utiliser un algorithme RANdom SAmple Consensus (RANSAC) pour trier les paires et d'éliminer les paries erronées. La matrice homographique est ensuite estimée avec des algorithmes d'optimisation comme celui de Levenberg Marquardt (LM) [60] ou alors l'algorithme RANSAC. Enfin, lorsque les images sont reprojetées et assemblées, une série de traitements est encore nécessaire pour supprimer les bords visibles des images En effet, les différences de gain et d'éclairage font que les images n'ont pas forcément la même dynamique les unes par rapport aux autres. Une compensation de gain peut être effectuée [24] et finalement un *Multi Band Blending*. Enfin, certains travaux proposent même une démarche d'évaluation des résultats comme en [61].

### 2.5.3 *Modélisation*

L'utilisation de caméras classiques pour la fabrication des systèmes multi-caméras offre la possibilité d'utiliser le modèle sténopé décrit plus haut. Chaque caméra est calibrée séparément et la transformation d'une caméra à l'autre est connue à la fin des calibrages (car elles sont fixes les unes par rapport aux autres). Cela dit la matrice homographique comporte les informations nécessaires pour décrire ce mouvement.

1 Les primitives et les détecteurs mentionnés ci-dessus sont présentés dans le chapitre 3

### 2.5.4 *Exemples*

Plusieurs systèmes existent dans le commerce et sont relativement moins populaires que les caméras catadioptriques. On peut trouver les caméras de l'entreprise *Point Grey* et la série *Ladybug* [3]. Le dernier modèle comportant 6 caméras et délivrant un flot panoramique de 30 Méga pixels (Mp) à 10 Frame Per Second (FPS).
La société *Immersive Media* propose un modèle de caméra capturant des panoramas de $360° \times 360°$ degrés avec 11 caméras [1]. Le débit est de 30 images par seconde à $2400 \times 1200$ pixels.
Enfin, du coté de la société *Arecont Vision* [5], la caméra *Suround Video* capture des panoramas de 20 Mp à 3 FPS.
Dans la recherche, des travaux ont aussi proposés des bancs de caméras avec traitement embarqué. Par exemple, dans [43], un modèle complet est réalisé avec 10 caméras (5 couples stéréoscopiques). Les images sont rectifiées sur le flot, alignées puis sont reprojetées en format cylindrique. L'assemblage est ensuite effectué en temps réel selon la méthodologie décrite plus haut. Ces traitements étant lourds à faire en temps réel, il existe peu de systèmes les proposant et le nombre d'applications est limité. On peut néanmoins trouver des applications routières [91] parmi quelques autres. Enfin, le champ de vision peut aussi être amélioré en combinant des optiques *Fisheyes* avec cette méthodologie [10].

### 2.5.5 *Conclusion*

En conclusion, le tableau 2.4 résume les caractéristiques de ces systèmes.

| **Caractéristiques** | **Polydioptrique** |
|---|---|
| Architecture | Intermédiaire (plusieurs caméras) |
| Résolution | $\leqslant$ Référence $\times$ nb caméras |
| Cadence | = Référence |
| Capture | *Global Shutter* |
| Type d'images | Multiples projections planaires |
| Centre de projection | Multiples |
| Champ de vision | jusqu'à $360° \times 360°$ |
| Densité de résolution | Homogène |
| Reprojection | Nécessaire |
| Stitching | Nécessaire |

TABLE 2.4: Résumé des caractéristiques des méthodes polydioptriques.

Bien que nous ayons des résolutions plus importantes et homogènes, d'autres limitations viennent rendre la méthode peu attrayante. En effet, afin d'avoir un panorama unique, beaucoup de traitements sont nécessaires et sont très couteux en temps de calcul

et en ressources de traitement. De plus, le fait d'avoir plusieurs centres de projections (un par caméra) rend la géométrie des images plus complexe. Le haut débit possible devient également un facteur limitant et est partiellement responsable des faibles débits rencontrés (souvent de l'ordre de 10 FPS). Cette méthode a besoin d'évolutions technologiques pour être plus viable dans des applications en temps réel.

## 2.6 SYSTÈMES PYRAMIDAUX

### 2.6.1 *Architecture*

Nous avons créé une catégorie de méthode que nous avons appelé *pyramidal* pour rassembler tous les systèmes qui sont à la fois polydioptriques et qui comportent des miroirs planaires. Ce nom fait référence à l'emploi fréquent de solides en forme de pyramides dont les cotés sont des miroirs. En d'autres termes, il s'agit donc d'une association de plusieurs caméras catadioptriques à miroirs planaires. Les miroirs sont souvent agencés en forme pyramidale avec un nombre de cotés variable. La figure 2.9 présente un exemple employant deux caméras et un tétraèdre de miroirs.

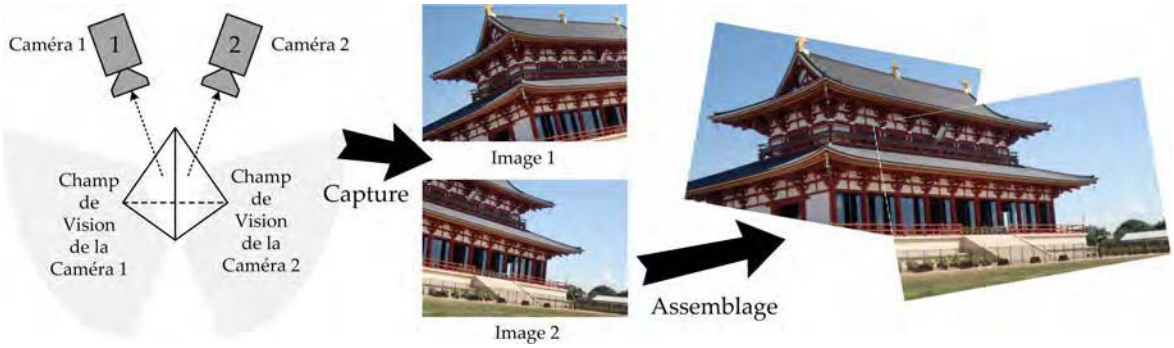


FIGURE 2.9: Exemple d'assemblage d'images à partir d'un système pyramidal à 2 caméras et un tétraèdre réfléchissant.

Le but de cette approche est de disposer les caméras sur des faces distinctes de manière à avoir deux caméras virtuelles ayant un centre de projection commun. En effet, on peut considérer l'ensemble miroir caméra comme une caméra virtuelle. De cette manière on obtient un système avec un seul point de vue pour conserver les perspectives. Les images sont ensuite assemblées avec des méthodes de *Stitching* présentant les mêmes caractéristiques que celles que nous avons présenté précédemment.

### 2.6.2 *Modélisation*

La modélisation de ces caméras peut se faire de plusieurs manières. La plus commune est d'utiliser le modèle sténopé et de l'appliquer à la caméra virtuelle. Cette caméra virtuelle est créée par le processus de réflexion et on la situe de l'autre coté du miroir en effectuant une symétrie planaire sur le centre de projection de la caméra. Une autre méthode moins utilisée consiste à reprendre le modèle catadioptrique présenté précédemment et d'utiliser un paramètre $\xi$ nul. Le calibrage se fait ensuite comme pour le modèle sténopé.

### 2.6.3 *Exemples*

Quelques systèmes commerciaux employant cette méthode sont disponibles. Mais aucun d'entre eux ne vise une application temps réel. Parmi d'autres on peut notamment citer ceux de l'entreprise *Fullview*. Deux systèmes sont proposés, le premier comporte cinq caméras et une pyramide à cinq cotés [2]. La vidéo panoramique générée a une résolution de 5 Mp et un débit de 30 FPS et est destinée à des applications de visio-conférence. Le

deuxième produit présente dix caméras et une pyramide à dix cotés avec un débit de 30 FPS à 4 Mp, l'application visée est la vidéo-surveillance.
Du coté des travaux publiés, on peut trouver plusieurs type de miroirs employés. Plusieurs systèmes pyramidaux [137, 81], certains avec des pyramides tronquées et d'autre avec des prismes hexagonaux [34]. Dans tous les cas, la propriété SVP est recherchée. En [49] et en [133], l'emploi d'une double pyramide tronquée est présenté (Dual Mirror-Pyramid (DMP)). Le placement des caméras est primordial, c'est lui qui va définir le champ de vision final obtenu. Certains travaux cherchent à modéliser le FOV en fonction des placements afin de l'optimiser et de rendre l'utilisation des capteurs la plus efficace possible [49, 133].

### 2.6.4 *Conclusion*

Tous ces systèmes sont difficiles à mettre en œuvre et une étape de *Stitching* est nécessaire pour générer une vue unique. Parfois, une reprojection cylindrique est employée. Le tableau 2.5 résume les caractéristiques de ces méthodes.

| **Caractéristiques** | **Pyramidales** |
|---|---|
| Architecture | Complexe (plusieurs caméras et miroirs) |
| Résolution | $\leqslant$ Référence $\times$ nb caméras |
| Cadence | = Référence |
| Capture | *Global Shutter* |
| Type d'images | Multiples projections planaires |
| Centre de projection | Unique possible |
| Champ de vision | Jusqu'à 360° $\times$ 360° |
| Densité de résolution | Homogène |
| Reprojection | Nécessaire |
| Stitching | Nécessaire |

TABLE 2.5: Résumé des caractéristiques des méthodes pyramidales.

De manière similaire aux systèmes polydioptriques simples (sans miroir) on obtient des propriétés comparables. La principale différence est le fait qu'on peut avoir un centre de projection unique. Les résolutions possibles ne sont limitées que par le nombre de caméras, il est possible de les optimiser en calculant les meilleurs placements des caméras et des miroirs. On peut aussi obtenir des champs de vision omnidirectionnels. Cette méthode présente aussi les mêmes limitations que la précédente et est pour cette raison très peu employée dans les application de robotique.

## 2.7 SYSTÈMES ROTATIFS MATRICIELS

### 2.7.1 *Architecture*

Les méthodes rotatives, contrairement aux autres présentées jusqu'à présent, emploient des mécaniques et se basent sur le mouvement des caméras. Une caméra classique est fixée sur une plateforme rotative qui fait varier son orientation. En fonction de l'application, la caméra est amenée à prendre des captures un certain nombre de fois par révolution et les images sont assemblées comme illustré en Figure 2.10.

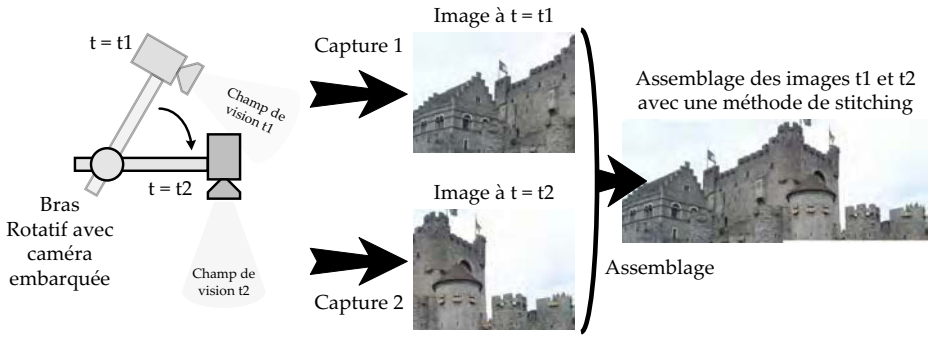


FIGURE 2.10: Schéma explicatif de la méthode de capture panoramique par caméra rotative : Une caméra rotative capture des vues durant sa rotation autour d'un axe. Les images sont ensuite assemblées en un panorama.

Cette méthode présente une limitation majeure dû au fait que les captures ne sont pas simultanées. La cadence est donc inférieure à celle de la caméra et un effet de *Rolling Shutter* est inévitable. La cohérence de l'image peut en être altéré (apparition d'artéfacts) dans le cas d'un mouvement du système ou de l'environnement. De plus, la capture doit être synchronisée avec les positions successives de la caméra pour avoir un système calibrable.

### 2.7.2 *Modélisation*

La modélisation de ces systèmes peut être faite en utilisant le modèle général sténopé. On peut aussi considérer le système comme un ensemble polydioptrique virtuel. La sous section précédente sur le modèle sténopé et celle sur les systèmes polydioptriques présentent déjà ce type de modélisation.

### 2.7.3 *Exemples*

On peut dénombrer quatre groupes de travaux principaux sur le sujet, chacun présentant une méthode d'acquisition différente.

TAIWAN INSTITUTE OF INFORMATION SCIENCES :

Le premier groupe de travaux utilise un ensemble de deux caméras montées sur un axe rotatif qui tourne manuellement [58]. Une prise stéréoscopique est effectuée tous les 15 degrés, ce qui fait un total de 24 couples d'images stéréoscopiques. Le résultat est un panorama stéréoscopique formant ainsi une image panoramique hexagonale. Les images sont assemblées par *Stitching* et le tout est reprojeté selon une vue cylindrique. L'application

visée est la réalité virtuelle (Virtual Reality (VR)) [59]. Le système étant manuel, il n'est pas utilisable de manière automatique et n'est pas capable de servir dans des applications en temps réel. Les résultats présentés sont ensuite améliorés en ajoutant un deuxième degré de liberté aux caméras afin de capturer des panoramas sphériques pour avoir une vision omnidirectionnelle [26].

MICROSOFT VISION TECHNOLOGY GROUP :

Dans ces travaux, un autre système rotatif est présenté [124]. Ici une seule caméra est en rotation pour la capture d'un panorama. Pendant la rotation, la caméra effectue 1350 captures d'images à différentes positions. Pour chaque image capturée, la colonne d'indice 60 est enregistrée pour la génération d'un premier panorama, et la colonne d'indice 260 est enregistrée pour la génération d'un deuxième panorama. Après les 1350 captures, les deux panoramas sont obtenus, le premier correspond à l'assemblage des 1350 colonnes d'indices 60 et le deuxième à l'assemblage des 1350 colonnes d'indices 260. Ces deux panoramas stéréoscopiques sont donc à projections cylindriques. Dans ces travaux, ces images sont appelées *Omnivergentes*. Une reconstruction stéréoscopique est ensuite proposée en [125, 126] où une carte de disparité est générée. Cette technique est très similaire à la rotation de caméras linéaires qui sera présentée en section suivante. Le fait est qu'avec une caméra classique, la vitesse d'acquisition est très lente et prends plusieurs minutes vu que la caméra doit capturer une image entière 1350 fois pour pouvoir assembler un panorama stéréoscopique[68]. L'utilisation de seulement deux colonnes de chaque image n'est pas optimal, une architecture dédiée serait plus avantageuse.

FACULTY OF COMPUTER AND INFORMATION SCIENCE, SLOVÉNIE :

Des travaux similaires sont présentés en [92]. Une caméra est également employée sur un bras rotatif et deux colonnes de l'image sont utilisées afin de capturer des panoramas stéréoscopiques. Une carte de disparité est aussi générée [93]. Le système est trop lent pour une utilisation en temps réel, c'est pour cela qu'une proposition de système comportant de multiples caméras est faite en [94]. Ce système n'étant pas réalisable, il est simulé avec 120 puis 1800 caméras.

INSTITUTE OF COMPUTER SCIENCE, THE HEBREW UNIVERSITY OF JERUSALEM :

Enfin, dans cette dernière série de travaux, une première étude sur le mouvement de caméra est faite en [98] où une caméra est tenue à la main pendant la capture. Les images résultantes sont assemblées avec une méthode de *Mosaicing* qui comporte une étape d'alignement, puis une d'assemblage. Ensuite, la caméra est placée sur un bras rotatif et la même approche est appliquée en [95]. Cette fois, deux colonnes séparées sont utilisées pour l'assemblage de panoramas stéréoscopiques et la génération d'image en anaglyphes[2] pour la visualisation 3D. Une méthode de génération de panoramas omni-stéréos avec un large entre-axe (distance entre les deux caméras) afin de capturer des scènes éloignées [101] est décrite. Dans d'autres publications, deux propositions de systèmes panoramiques sont faites. Le premier est un système d'optique permettant de capturer des panoramas sans élément mécanique ni mouvement [99]. Le deuxième est un miroir en spirale accompagné

2 Les anaglyphes sont des images faites pour la visualisation en relief grâce à l'emploi de lunettes spéciales

d'une lentille en spirale. Les deux systèmes sont décris sans être fabriqués [96, 102, 97]. Enfin, un algorithme de génération de cartes de disparité est proposé en [131] à partir des panoramas générés par la caméra rotative présentée précédemment. Le problème de cadence d'acquisition empêche toujours l'utilisation en temps réel.

LES AUTRES TRAVAUX :

D'autres travaux explorent les possibilités de l'approche [28], mais aucune méthode n'est utilisable en temps réel. Enfin, le problème de la dynamique est posé en [6] où une solution est proposée avec l'utilisation de filtres devant l'objectif. Ainsi, lors de la capture d'une image par la caméra, les colonnes sont plus ou moins atténuées en luminosité. Pendant la rotation on peut donc assembler plusieurs panoramas (en accumulant des colonnes de chaque atténuation en des panoramas distincts) et un algorithme High Dynamic Range (HDR) fusionne ces derniers en un nouveau panorama avec une dynamique améliorée.

### 2.7.4 *Conclusion*

Le tableau 2.6 résume les caractéristiques que nous venons de parcourir.

| **Caractéristiques** | **Rotation de caméras classiques** |
|---|---|
| Architecture | Complexe (Rotation) |
| Résolution | $\leqslant$ Référence $\times$ nb positions |
| Cadence | = Référence $\div$ nb positions |
| Capture | *Rolling shutter* |
| Type d'images | Prismatiques |
| Centre de projection | Unique possible |
| Champ de vision | $90° \times 360°$ |
| Densité de résolution | Homogène |
| Reprojection | Nécessaire |
| Stitching | Nécessaire |

TABLE 2.6: Résumé des caractéristiques des méthodes rotatives classiques.

Cette méthode, peut permettre la génération de panoramas cylindrique qui sont souvent convoités pour leur projection idéale en vision panoramique. Ainsi il n'y a pas besoin d'algorithmes de *Stitching* ce qui libère du temps de traitement. De plus, la résolution dépend d'un part de la résolution verticale de la caméra, et d'autre part du nombre d'images capturées (et donc du nombre choisi de positions). Ainsi, il n'y a pas de facteur limitant la résolution horizontale hormis la précision de la mécanique de rotation. Cela dit, le défaut de la lenteur est inévitable avec l'usage d'une caméra classique, ces systèmes ne sont donc

pas utilisables dans une problématique temps réel de type SLAM. Une amélioration consiste à utiliser une caméra linéaire, comme nous l'abordons dans la sous section suivante.

## 2.8 SYSTÈMES ROTATIFS LINÉAIRES

### 2.8.1 *Architecture*

Dans cette section, nous abordons les systèmes rotatifs qui emploient des caméras dites linéaires. Ce genre de caméra comporte un capteur d'image qui ne possède qu'une seule colonne de pixels. Les images capturées sont donc en 1 Dimension (1D). La caméra capture donc une colonne à chaque position de la plateforme rotative et les colonnes obtenues sont concaténées de la même manière que dans les travaux vu précédemment. L'image complète obtenue a une projection cylindrique avec plusieurs centres de projection si le centre de la caméra n'est pas situé sur l'axe de rotation. Cette méthode donne de meilleurs résultats en terme de fréquence d'acquisition car la caméra est optimisée pour la capture d'une seule colonne, sa vitesse est donc plus rapide que celle d'une caméra classique qui va mettre un certain temps à restituer une image 2D entière. De plus le facteur de remplissage peut être optimisé pour atteindre 100% plus facilement et ainsi permettre des zones photoréceptrices plus grandes pour réduire les temps d'intégration.[3]

### 2.8.2 *Exemples*

SYSTÈMES COMMERCIAUX :

Ces caméras comportent le plus grand potentiel de résolution de tous les systèmes présentés. En effet, on peut trouver des systèmes commerciaux capturant des images panoramiques de 66 à 138 Mp [4]. La société *Roundshot* [4] propose une série de caméras à très grandes résolutions mais aussi à cadences de capture lentes (avec un minimum de 6 secondes pour la capture d'un seul panorama). Les sociétés *SpheronVR* et *Panoscan* proposent aussi des caméras avec des résolutions atteignant les 700 Mp par panorama mais avec un temps de capture s'élevant à quelques minutes pour chaque image. Ces systèmes, bien que très lents, possèdent des mécaniques très précises et les acquisitions sont synchronisées avec les positions successives de la caméra. Ainsi, il est possible de faire de la photogrammétrie, car on peut connaitre la position de la caméra lors de la capture des images. En d'autres termes cela revient à dire que ces systèmes peuvent être modélisés et calibrés. A cause de ce défaut de lenteur, ces caméras sont peu employées [21], et ne sont pas utilisables dans une problématique temps réel de type SLAM.

LES TRAVAUX DE BENOSMAN, SMADJA ET ALI. :

Plusieurs travaux ont été effectués sur la réalisation de caméras similaires. En [15], Benosman et ali. présentent la conception d'un système employant une caméra à trois colonnes de pixels sur une plateforme rotative contrôlée par un moteur pas à pas [16]. Le tout produit des images de $1024 \times 3600$ pixels mais nécessite un temps d'acquisition de 4 minutes 30. Le but est de déterminer les formes géométriques des scènes intérieures. Une amélioration est apportée en [127] où Smadja et ali. présentent un nouveau système stéréoscopique ayant une vitesse d'acquisition plus rapide (36 secondes par panorama) et capturant des images d'un résolution de $2048 \times 3600$ pixels. Enfin, la construction d'une carte de disparité dense est décrite en [129].

3 La modélisation de ces systèmes est présentée en détail dans le chapitre 4.

DLR & DRESDEN UNIVERSITY OF TECHNOLOGY :

La plus grande contribution en la matière a été apportée par les organismes *DLR* et par l'université de Dresden en Allemagne. Les premiers travaux se sont intéressés à la caméra linéaire Wide Angle Optoelectronic Stereo Scanner (WAOSS) qui avait été conçue à la base pour des applications spatiales. Le design a été modifié en une nouvelle caméra linéaire appelée Wide Angle Airborne Camera (WAAC) et comportant un capteur CCD de trois colonnes. La caméra a été montée sur une plateforme rotative pour la capture panoramique à 180 degrés, la stéréo était rendue possible en déplaçant le prototype pour faire une nouvelle acquisition [104]. Cette première expérience a ensuite été approfondie avec la réalisation d'une nouvelle caméra appelée *Eyescan M1* [112]. Avec ce modèle, la vision à 360 degrés devient possible. Une caméra linéaire est montée sur une plateforme rotative. Le capteur employé comporte trois colonnes de 10200 pixels chacune, la résolution verticale des panoramas produits est donc également de 10200 pixels. La capture est cependant toujours lente et nécessite à peu près quatre minutes. Cette plateforme est ensuite utilisée dans la plupart des travaux qui suivent [51, 50].
Dans [42], une caméra linéaire est embarquée sur un véhicule en mouvement. Pendant le déplacement, la caméra est orientée de manière orthogonale à la direction du véhicule, elle capture et assemble une image pendant le mouvement, cette technique est appelée *Push-Broom*. Les imperfections de la route et les vibrations rendent l'image très perturbée et font apparaitre des distorsions. Une Inertial Measurement Unit (IMU) mesure les mouvements du véhicule afin de filtrer ces vibrations et ces perturbations. L'image est ensuite rectifiée. Enfin, une autre caméra est réalisée, le système Multi Functional Camera (MFC) utilisant une centrale inertielle (IMU) et un Global Position System (GPS) [20].
L'institut de photogrammétrie de Stuttgart a travaillé en collaboration avec DLR et a utilisé la version M2 de la caméra *Eyescan* [106, 44]. Un scanner laser est associé au système pour obtenir des informations télémétriques sur le panorama capturé. La fusion des données est étudiée et des applications de navigation sont proposées en [105].
Enfin, l'université de Dresden a aussi beaucoup contribué à cette série de travaux en développant un modèle mathématique pour ce type de système afin de déboucher sur des applications de photogrammétrie[4] urbaine [114]. Ensuite, un autre système panoramique de 94 Mp qui a un temps de capture de 40 secondes à 3 minutes est proposé [116].
L'ensemble de ces travaux a donné suite à la publication du livre *Panoramic Imaging* en 2008 [52].

LES AMÉLIORATIONS DE LA CADENCE :

Les caméras citées jusqu'à présent présentent toutes des vitesses d'acquisition très lentes. En 2006, Hiroita et ali. [46] proposent un modèle de caméra plus rapide avec une vitesse d'acquisition de 11,5 images par secondes pour une résolution de $7500 \times 15000$ pixels. La capture est synchronisée avec la position de la plateforme et une méthode de calibrage est aussi proposée. Puis en 2008, Nielsen et ali. [83] proposent une plateforme plus rapide qui peut atteindre la cadence de 30 FPS. Cela dit, la mécanique n'est pas asservie et la capture n'est pas synchronisée avec le mouvement de la caméra. Ce qui veut dire que la vitesse n'est pas stable et la taille des panoramas varie également de manière incontrôlable. Avec la capture de 6700 lignes par secondes, à 30 Rotations Par Seconde (RPS) on obtient des panoramas de 223 pixels horizontalement, la résolution verticale étant de 400 pixels.

4 La photogrammétrie est la discipline visant à mesurer la géométrie d'une scène à partir de photographies

Enfin, une amélioration significative est celle apportée par AIT en 2010 et en 2012 [14, 13]. Une caméra rotative appelée la *BiCa360* produit des panoramas à une vitesse de 10 FPS et une résolution de 256 pixels verticalement et avec une résolution horizontale non synchronisée et qui varie autour de 1000 colonnes. Les panoramas comportent la particularité d'être des formes de gradients analogiques, donc il n'y a pas d'images à proprement parler, ce qui permet de ne pas avoir de temps d'intégration.

### 2.8.3 *Conclusion*

Le tableau 2.7 fait un résumé des propriétés de ce type de système.

| **Caractéristiques** | **Rotation de caméras linéaires** |
|---|---|
| Architecture | Complexe (Rotation) |
| Résolution | = Référence × nb positions |
| Cadence | = Référence ÷ nb positions |
| Capture | *Rolling shutter* |
| Type d'images | Cylindriques |
| Centre de projection | Unique possible |
| Champ de vision | 90° × 360° |
| Densité de résolution | Homogène |
| Reprojection | Non nécessaire |
| Stitching | Non nécessaire |

TABLE 2.7: Résumé des caractéristiques des méthodes rotatives linéaires.

Tous les systèmes présentés ont le problème commun de cadence d'acquisition ralentie. Certains présentent des vitesses intéressantes, mais cela grâce au sacrifice de la résolution et de la synchronisation de la capture comme en [83].

## 2.9 CONCLUSION

Dans ce chapitre, nous avons étudié les différentes méthodes existantes pour la capture de vues panoramiques. Nous avons établi six catégories distinctes et avons répertorié leur caractéristiques. Le tableau 2.8 rassemble toutes les informations déjà expliquées plus haut et compare les six méthodes.

| **Caractéristiques** | **Meilleur Méthode** | **Moins bonne méthode** |
|---|---|---|
| Simplicité de l'architecture | Fisheyes | Pyram. Rotatif |
| Centre de projection unique | Tous sauf Poly. | Poly. |
| Images Cylindriques | Rotatif 1D | toutes les autres |
| Cadence | Tous sauf Rotatifs | Rotatif 1D |
| Champ de vision | Rotatif, Poly. & Pyram | Cata |
| Résolution | Rotatif 1D | Cata. Fish. |
| Densité de résolution homogène | Rotatif 1D | Cata. Fish. |
| Global Shutter | Tout sauf rotatif | Rotatif 1D et 2D |
| Pas de Reprojection | Rotatif 1D | toutes les autres |
| Pas de Stitching | Rotatif 1D, fish. et Cata. | Poly. Pyram. Rotatif 2D |

TABLE 2.8: Résumé des caractéristiques des différentes méthodes étudiées pour la production d'image panoramiques. En bleu apparaissent les avantages de la méthode linéaire rotative et en rouge ses défauts.

Mis à part ces catégories fondamentales, il existe d'autres méthodes pour la capture d'images panoramiques, et nous avons classifié les principales. Les autres façons peuvent être obtenues en combinant les approches présentées. Par exemple, en combinant l'emploi d'un objectif *Fisheye* avec celui d'un miroir, on obtient une caméra catadioptrique améliorée qui peut permettre la stéréovision [67]. Dans [87], une caméra catadioptrique est positionnée sur une plateforme rotative. Enfin dans [134], une série de prismes rotatifs, un miroir hyperboloïde et une caméra produisent des panoramas cylindriques segmentés.
En bilan, les méthodes qui présentent le plus grand intérêts sont celles qui prodiguent de grandes résolutions et qui fournissent des panoramas à projections cylindriques. Les traitements de reprojection et de *Stitching* sont coûteux en temps et en ressources, obtenir un panorama directement cylindrique nous évite de devoir implémenter ces traitements.

Les méthodes rotatives 1D et 2D produisent des résultats géométriques similaires et une qualité d'image équivalente. D'un autre côté, elles présentent trois défauts limitant :

- La lenteur d'acquisition,
- La capture de type *Rolling Shutter* qui peut induire l'apparition d'artéfacts et d'incohérences lors de capture de scènes mobiles,

- La complexité de mise en oeuvre.

En comparant les méthodes rotatives 1D et 2D, on s'aperçoit que l'architecture linéaire est optimisée et plus efficace :

- Meilleure fréquence de capture,
- Meilleur facteur de remplissage, souvent à 100%,
- Plus basse consommation car une seul colonne CMOS est employée.

La méthode rotative 1D est donc optimisée pour notre type d'application, cela dit, cela n'a pas été réellement exploité par les systèmes rotatifs existants. Pour ces raisons, une nouvelle plateforme rotative à capteurs linéaires a été développée dans le cadre de cette thèse. Elle est présentée et décrite dans le chapitre 3.

# CHAPITRE 3

## Architecture matérielle et logicielle

### 3.1 INTRODUCTION

Dans ce chapitre, quatre points sont abordés. Dans un premier temps, nous présentons l'architecture électronique et mécanique du système panoramique développé. Les ressources électroniques, les différents périphériques et les choix effectués sont expliqués afin de bien définir l'environnement d'implémentation dont nous disposons. Dans un second temps, nous décrivons l'implémentation de base permettant le fonctionnement de l'ensemble du système pour la capture d'images panoramiques. Nous abordons ensuite le sujet de l'extraction de primitives et expliquons l'avantage qu'il apporte aux systèmes de vision haut débit. Après avoir fait une comparaison des différents détecteurs existants, nous présentons l'implémentation d'un algorithme utilisant le détecteur de *Harris & Stephens*. Enfin, ce chapitre est conclu après la présentation des résultats obtenus sur notre système.

## 3.2 ARCHITECTURE MATÉRIELLE

### 3.2.1 *Architecture générale*

Un plateau rotatif est entrainé par une motorisation asservie en vitesse. Le plateau a été conçu pour pouvoir accueillir quatre têtes stéréoscopiques afin de multiplier la vitesse d'acquisition. Cela dit, les travaux présentés dans cette thèse ont été réalisés avec une tête stéréoscopique.
L'architecture du système ainsi réalisé est présentée en figure 3.1.

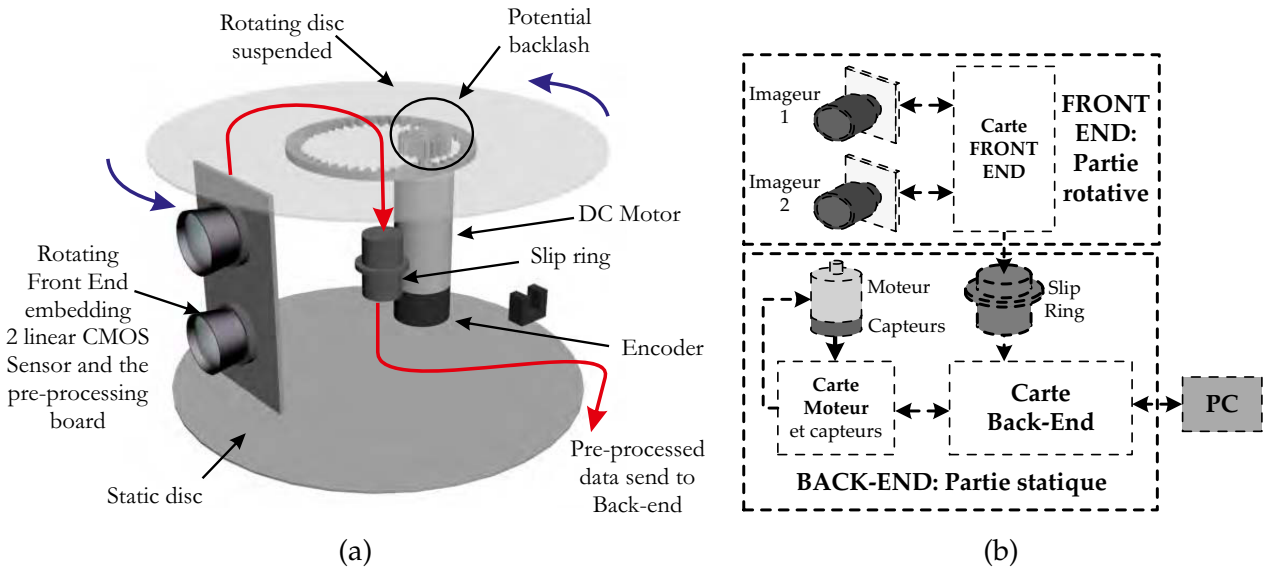


FIGURE 3.1: (a) La configuration géométrique retenue ainsi que l'architecture mécanique du système avec un seul couple stéréoscopique. (b) Schéma synoptique du système dans son intégralité.

Comme il est illustré en (b), le système est composé de deux parties :

- Le *Front-End* est la partie qui est en rotation et est représenté par le plateau circulaire supérieur sur la figure 3.1 (a). Le couple stéréoscopique visible sur le schéma est suspendu à ce plateau rotatif et l'ensemble constitue le *Front-End*. Le *Front-End* embarque l'électronique de contrôle des capteurs et d'acquisition des images panoramiques.
- Le *Back-End*, quant à lui, constitue le reste du système, c'est à dire toute la partie statique. Il comporte la mécanique nécessaire pour la mise en mouvement du *Front-End*, un ensemble de capteurs de position et deux cartes électroniques. La première est une carte de puissance appelée *Carte moteur* sur la figure 3.1 (b). La deuxième est une carte numérique de contrôle qui assure les fonctions suivantes :
  - Asservissement de la plateforme rotative en vitesse,
  - Calcul de la position de la plateforme, et déclenchement des intégrations de manière synchronisé,
  - Communication avec un ordinateur et transmission des images.

La rotation perpétuelle du plateau rend impossible la connexion filaire directe entre le

*Front-End* et le *Back-End*. Une technologie appelée *Collecteur Tournant* ou *Slip-Ring* fait l'intermédiaire et permet de faire passer les informations et l'alimentation entre le *Front-End* et le *Back-End*.
Deux capteurs permettent de mesurer la position du plateau à tout instant :

- Un codeur incrémental sur l'arbre du moteur pour connaitre sa position relative, et donc la position relative de la plateforme rotative,
- Une fourche optique sous le plateau pour connaitre sa position absolue.

Le codeur incrémental a une résolution de 500 points, ce qui donne une résolution quadratique possible de 2000 points. Le rapport de réduction appliqué à l'arbre du moteur étant de 6.6, la résolution de la plateforme rotative est donc de 13200 positions par tour. La fourche optique sert uniquement à connaitre la position initiale de la plateforme, c'est à dire la position ou la première colonne d'image peut être capturée. La fusion des données de ces deux capteurs indique la position relative et absolue du plateau tournant avec une précision de 13200 positions par tour. L'acquisition des caméras est synchronisée avec la position du plateau afin d'avoir une réelle corrélation entre position et capture. Chaque position détectée sert à déclencher une intégration avec les capteurs, cette intégration donne lieu à la capture d'une nouvelle colonne image. Le nombre de positions possible détermine donc directement la résolution horizontale des images. Nous avons choisi de capturer une image toutes les deux positions afin d'avoir une résolution horizontale de 6600 pixels. Cette résolution est choisie pour limiter la taille des images panoramiques et d'accélérer la vitesse d'acquisition. En conclusion, la plateforme est asservie en vitesse, mais l'acquisition est synchronisée avec la position. Cela permet une grande souplesse d'utilisation et l'immunité aux vibrations extérieures qui peuvent altérer la stabilité de la vitesse de rotation.

### 3.2.2 *Capteur employé*

La capture d'image panoramiques cylindriques se fait avec des capteurs linéaires, c'est à dire des capteurs ne disposant que d'une seule colonne de pixel.

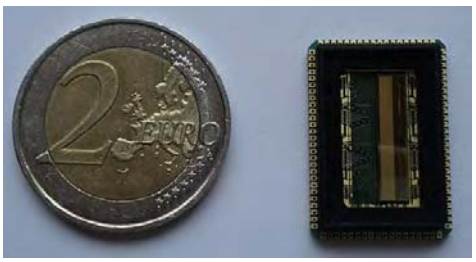

FIGURE 3.2: Photo du capteur linéaire utilisé : AWAIBA DRAGSTER DR-2080-7LCC

Dans le cas de notre système, nous avons choisi les capteurs CMOS de la société *Dragster AWAIBA*, comme présenté sur la photo 3.2. Ces capteurs sont ceux qui sont les plus rapides du marché avec une fréquence d'acquisition de 80000 colonnes par seconde. En prenant compte de la résolution horizontale de 6600 colonnes par image, le débit théorique maximal est de 12 panoramas par seconde. Cependant, ce débit ne peut être atteint qu'avec l'utilisation de dispositif d'éclairages puissants. Comme le but est d'utiliser le système en environnement urbain, l'éclairage ambiant est la seul source de lumière, en conséquence

le temps d'intégration doit être allongé et le débit maximal théorique ne pourra pas être atteint. La taille des pixels étant de sept micro mètres sur sept et le facteur de remplissage de 100%, la surface d'exposition est assez grande pour permettre un temps d'intégration plus court que pour les capteurs matriciels traditionnels. La résolution de ces capteurs noir et blanc est de 2048 pixels. C'est cette valeur qui contraint la résolution verticale de nos images panoramiques. Nous produisons donc des images de $6600 \times 2048$ pixels, soit 13,5 Mp par image.

### 3.2.3 *Front-End et Back-End*

Afin de remplir leurs fonctions, le *Front-End* et le *Back-End* ont chacun besoin d'une électronique de contrôle. Nous avons décidé de réaliser une carte polyvalente qui est capable de remplir les fonctions du *Front-End* et du *Back-End*. Cela permet de simplifier la réalisation du système en gagnant du temps de conception et en économisant des coûts de fabrication. Des cartes jumelles ont donc été conçues et comportent l'architecture électronique présentée en figure 3.3.

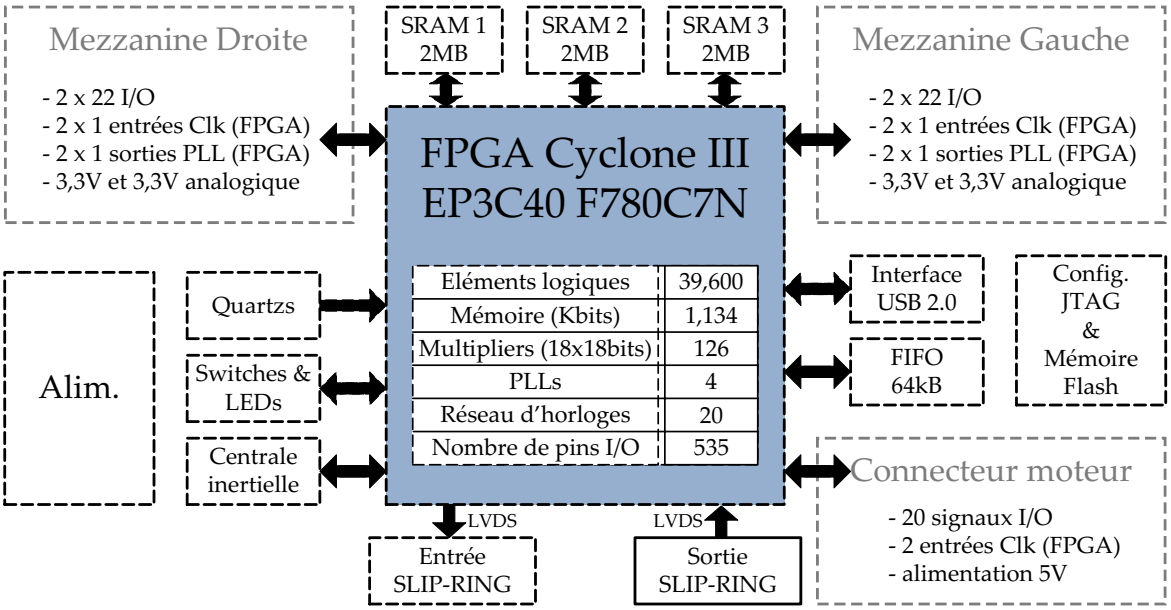


FIGURE 3.3: Le diagramme des ressources de la carte générique réalisée.

Un aperçu de l'une des cartes est illustré par la figure 3.4.

Les cartes comportent 8 couches et sont organisées autour d'un Field Programmable Gate Array (FPGA) de la société *ALTERA* (les ressources de ce FPGA sont listées sur la figure 3.3). Le FPGA est configuré en fonction de l'application désirée et peut même permettre l'implémentation de traitements des images panoramiques comme nous le montrons plus loin. Une certain nombre de périphériques ont été connectés au FPGA :

- Un micro-contrôleur Universal Serial Bus (USB) pour la communication avec le PC hôte (seulement utilisé dans le rôle du *Back-End*)
- Deux connecteurs d'extension pour la connection de deux mezzanines capteurs (seulement utilisées dans le rôle du *Front-End*)


FIGURE 3.4: Une photo de l'une des cartes génériques.

- Trois Static Random Access Memory (SRAM) de 2 Méga Octets (MO) chacune afin de prodiguer jusqu'à 6 MO de mémoire externe au FPGA pour les éventuels traitements vidéos qui seront implémentés. En prenant en compte une résolution de 8 bits par pixel, une moitié d'image panoramique peut être stockée.
- Une First In First Out (FIFO) externe de 64 kilo Octets (kO), cela peut être également utile pour les traitements de type vidéo.[1]
- Une centrale inertielle pour permettre la possibilité d'avoir les informations d'accélération instantanée. Ces informations peuvent s'avérer utile pour la correction des distorsions relatives au mouvement du système, dans les images panoramiques [42].
- Deux connecteurs Low Voltage Differential Signaling (LVDS), un dans chaque direction pour permettre de connecter deux cartes entre elles. Cela est utilisé pour connecter le *Front-End* avec le *Back-End* à travers le *Slip-Ring*.

Ces dernières propriétés ont aussi été choisies pour avoir la possibilité de chaîner les cartes les unes à la suite des autres. De cette manière, il est possible de connecter quatre cartes *Front-End* ensembles puis de les relier au *Back-End* à travers le Slip-Ring. Nous obtenons ainsi quatre couples stéréoscopiques chainés en guise de *Front-End* comme l'illustre la figure 3.5.

L'utilisation d'un bus LVDS entre le *Front-End* et le *Back-End* permet une transmission plus stable et plus efficace des données grâce à une plus grande immunité aux perturbations extérieures et une fréquence maximale de transmission plus élevée. L'architecture électronique des bus LVDS que nous avons choisi est mono-directionnelle, nous avons donc doté les cartes d'un bus d'entrée et d'un autre bus de sortie, ce qui rend le chaînage possible.

1 Les mémoires externes (SRAM et FIFO) ne sont pas utilisées dans l'application spécifique qui est présentée dans ces travaux, elles ont été implantées dans le design pour répondre aux besoins d'autres applications.

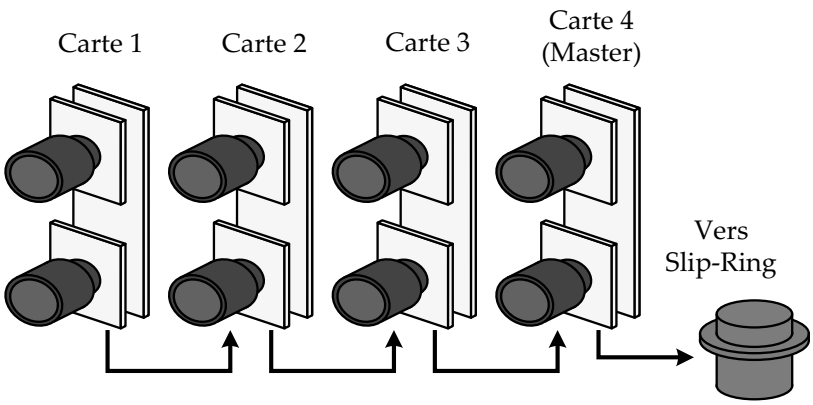


FIGURE 3.5: Illustration du principe de chaînage proposé.

## 3.3 IMPLÉMENTATION DES FONCTIONNALITÉS DE BASE DU SYSTÈME

Les deux cartes électroniques sont basées sur des architectures centralisées autour d'un FPGA. Ce FPGA doit être configuré afin d'assurer les fonctions relatives au *Front-End* ou au *Back-End*. Dans cette section, nous allons décrire les implémentations de base nécessaires au fonctionnement du système. Nous avons aussi défini notre bus selon un standard qui est utilisé tout au long des différentes étapes de traitement du flot vidéo. Ce bus est composé de trois signaux :

- Un bus pixel de 8 bits (les pixels étant codés sur 8 bits),
- Un signal de validation qui sert à indiquer quand un pixel est valide sur le bus de 8 bits,
- Un signal d'horloge à 160 Méga Hertzs (MHz).

Ce bus permet le transfert d'un pixel codé sur 8 bits par cycle d'horloge, c'est à dire qu'il a un débit maximum possible de 160 Mp par seconde (ce qui représente plus de onze images panoramiques par seconde).

### 3.3.1 *Framework Front-End*

L'architecture interne au FPGA de la carte *Front-End* est présentée en Figure 3.6.

Tout d'abord deux blocs assurent le contrôle des deux capteurs linéaires connectés à la carte *Front-End*. Le fait d'avoir deux blocs distincts est dû à la différence entre les deux capteurs. Deux configurations différentes sont également nécessaires pour obtenir des images d'une dynamique similaire. A la sortie des capteurs, nous avons un flot de quatre pixels en parallèle, et ces pixels sont codés sur 13 bits (conditions imposées par l'architecture des capteurs). Un bloc de formatage détaillé en figure 3.7 effectue une série de traitements bas niveau afin de mettre en place les signaux du bus vidéo décrit précédemment.

Ainsi, nous effectuons une soustraction d'offset numérique, un redimensionnement des bus, plusieurs contrôles de saturation et une sérialisation de manière à avoir un bus vidéo par capteur avec une cadence de 160 Mp par seconde.

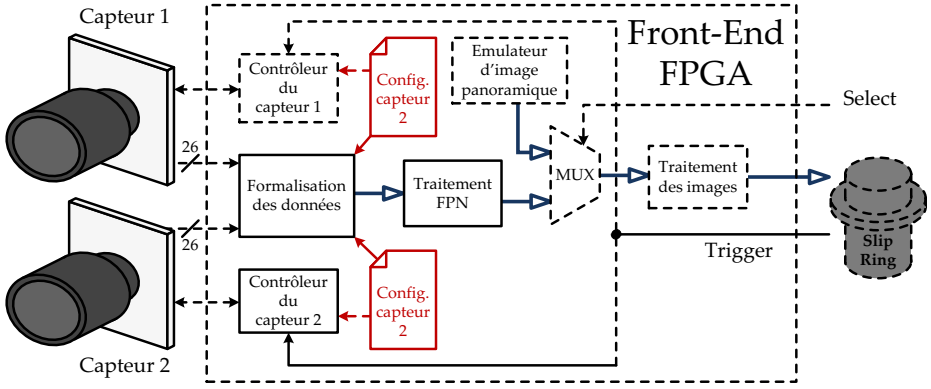


FIGURE 3.6: Structure de développement implémentée dans le FPGA de la carte *Front-End* et ses interactions avec les différents systèmes qui y sont connectés. Le bus vidéo est représenté par les flèches bleues foncées.

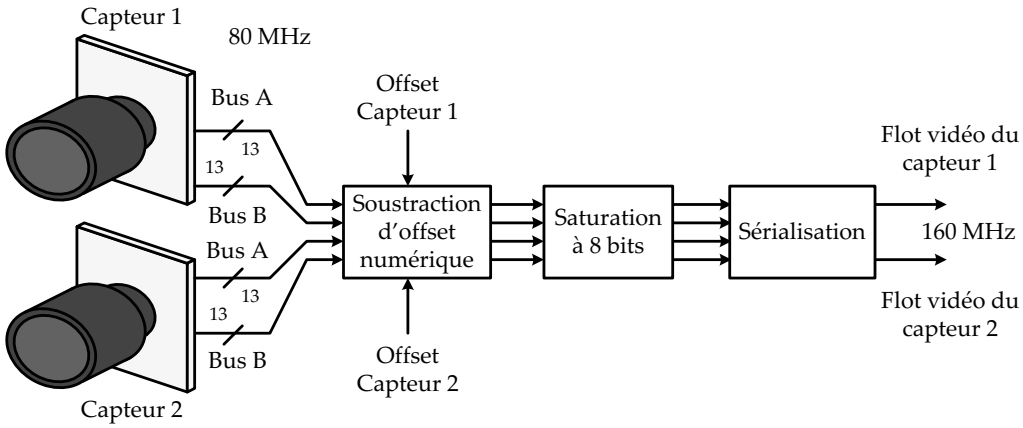


FIGURE 3.7: Détails du bloc de formatage des données.

Après cette opération de formatage des données, une étape de traitement du bruit spatial fixe (Fixed Pattern Noise (FPN)) est nécessaire. En effet, les capteurs présentent un bruit constant compte tenu de la différence physique des pixels et des amplificateurs qui leur sont associés. Cela se manifeste sous la forme d'artéfacts sur l'image qui apparaissent comme des lignes horizontales en certains endroits de l'image. Ce phénomène est illustré sur la Figure 3.8.

Le FPN a été mesuré expérimentalement et modélisé sous *Matlab*. Il s'avère être linéaire et la fonction de correction a été implémentée sur le flot vidéo dans le bloc *Traitement FPN* visible sur la Figure 3.6. Il a été mesuré expérimentalement avec des captures de niveaux de gris différents et avec une interpolation il a été possible de calculer les coefficients à appliquer pour le soustraire à l'image.

Afin de tester la transmission entre le *Front-End* et le *Back-End*, un module de génération d'images de synthèse a été implémenté. Un multiplexeur permet de choisir entre le flot d'images synthétiques et le flot d'images provenant des capteurs. Il est possible de tester le taux d'erreur de transmission en implémentant le même module de génération d'image dans le *Back-End* puis en en comparant les résultats.

Enfin, un dernier bloc de traitement est présent sur le schéma 3.6 et contient l'implémentation d'un extracteur de primitives visuelles que nous décrirons plus loin dans ce chapitre.

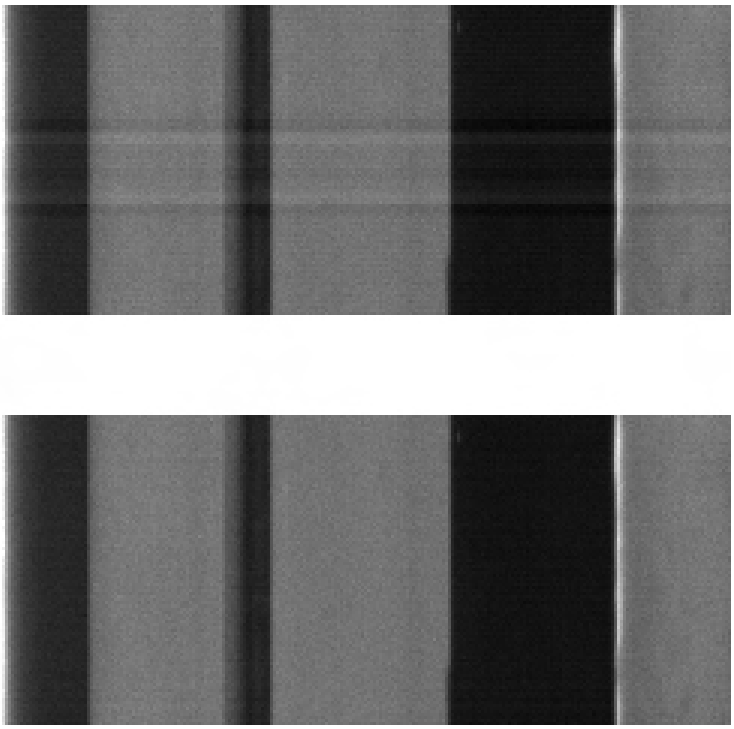

FIGURE 3.8: Comparaison d'une petite partie de l'image avant et après traitement du FPN.

### 3.3.2 *Framework Back-End*

L'architecture interne au FPGA de la carte *Back-End* est présentée en Figure 3.9.

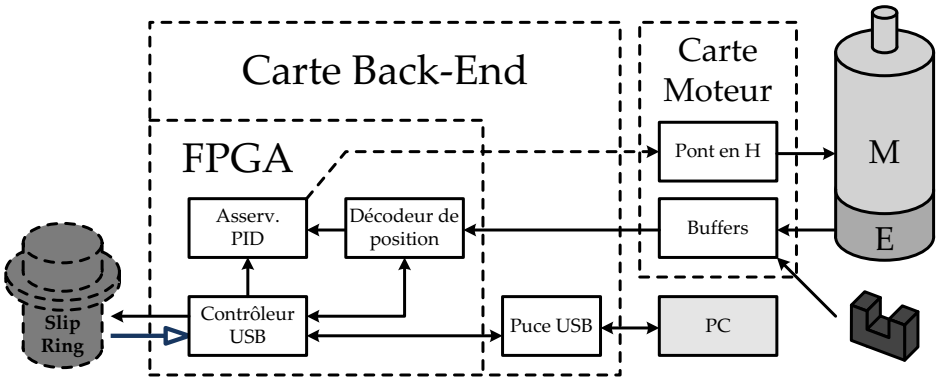


FIGURE 3.9: Schémas du *framework* implémenté dans le FPGA de la carte *Back-End* et ses interactions avec les différents systèmes qui y sont connectés

Le *Back-End* assure basiquement trois fonctions différentes. La première d'entre elles est l'asservissement du moteur et de la plateforme rotative. Nous avons choisi un asservissement en vitesse de type Proportionnel Intégral Dérivé (PID) afin d'avoir une vitesse la plus stable possible. Néanmoins, la capture des colonnes est synchronisée avec la position du plateau et non avec le temps. Cela rend la capture indépendante de la vitesse et une variation de celle-ci ne génère pas de distorsions dans l'image. La deuxième fonction du *Back-End* est de calculer la position du plateau et d'envoyer des signaux pour déclencher l'intégration des colonnes d'image au *Front-End*. Enfin, la dernière fonction est de

communiquer avec un ordinateur hôte et de transmettre les images capturées ou les données traitées. Un software a été programmé du coté PC afin de contrôler le système et de récupérer les images.

### 3.3.3 *Images et propriétés du système*

L'architecture présentée permet d'acquérir des images panoramiques cylindriques comme celle qui est montrée sur la Figure 3.10.

FIGURE 3.10: Un exemple de panorama capturé à 3 RPS.

Le capteur a été réglé selon les conditions d'éclairement (le temps d'intégration contraignant la vitesse de rotation), nous pouvons ici atteindre un débit de 3 images par seconde. Le FPN est soustrait sur le flot et la mise au point est réglée sur l'infinie. La figure 3.11 présente un agrandissement d'une partie de l'image afin de montrer la qualité de l'image.

FIGURE 3.11: Zoom présentant la qualité de l'image panoramique.

## 3.4 EXTRACTION DE PRIMITIVES

### 3.4.1 *Intérêt de l'extraction de primitives*

En traitement des images, une primitive est une petite partie ou une zone de l'image qui présente des caractéristiques remarquables. Dans certains cas, il peut s'agir d'une valeur scalaire décrivant localement une zone de l'image. En général, les primitives sont des coins ou des bords dans l'image, parfois même des blobs ou des zones saillantes. Trouver ces primitives dans une image est une tâche utile parce qu'elles admettent certaines caractéristiques qui suivant le cas peut être l'invariance à la translation, à la rotation, au changement d'échelle etc... Cette invariance peut être exploitée pour le suivi d'objet, de l'estimation de mouvement ou encore de la reconstruction 3D, pour ne citer que les applications majeures relatives au SLAM.

Le principe de l'extraction de primitives comporte deux étapes ; la détection et la description. La Figure 3.12 explique ces deux étapes et ce qu'elles apportent.

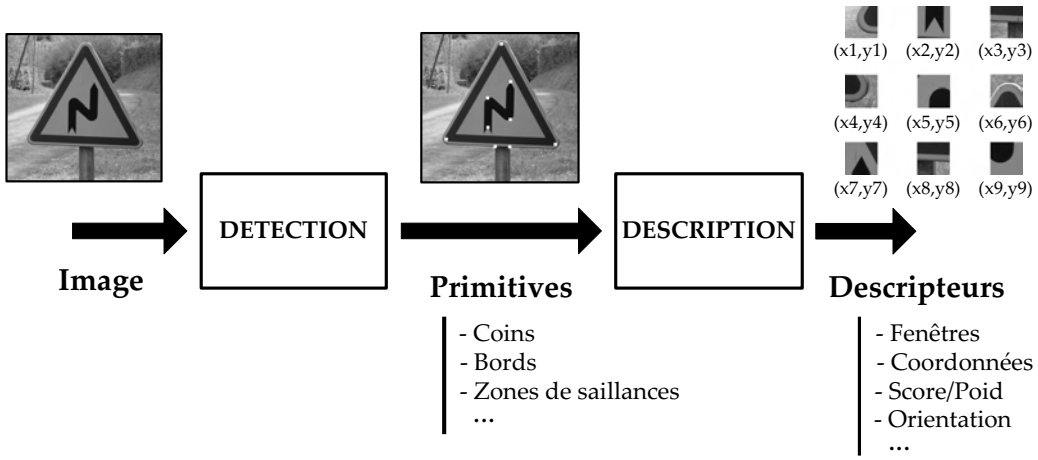


FIGURE 3.12: Extraction de primitives.

L'étape de détection consiste uniquement à trouver les primitives de l'image. Selon l'application, on peut privilégier un type de primitive. La Figure 3.12 présente l'extraction de primitives de type *coins*. La seconde étape consiste à construire un descripteur, c'est à dire un vecteur de paramètres qui va caractériser ces primitives. Ces informations peuvent être de plusieurs formes, telles que des fenêtres de pixels autour de la primitive, des coordonnées, un score résultant d'un calcul, ou encore une orientation... Un vecteur de paramètres est associé à chaque primitive détectée au préalable. Ces informations peuvent servir à diverses tâches par la suite comme le suivi des primitives d'une image à l'autre. Avec ce principe, il n'y a pas besoin de garder l'image entière pour la suite des opérations. Seuls les descripteurs sont utiles pour la suite du traitement, ce qui réduit considérablement le nombre de données à traiter. Après la description, on a souvent une troisième étape d'appariement qui recherche les points d'une image à l'autre en comparant les différents descripteurs. L'appariement permet de trouver des coordonnées 3D des points relatifs au système de vision, ce qui débouche sur les applications de *tracking*, de SLAM, ou encore de Simultaneous Localization And Navigation (SLAN).

Dans cette section, nous parcourons et comparons différents types de détecteurs afin d'en sélectionner un pour notre application. Cette sélection se fait en adéquation avec l'architecture de notre caméra rotative et aussi de nos besoins en détection.

### 3.4.2 *Types de détecteurs*

CHOIX DES DÉTECTEURS :

Nous avons dit qu'il existait plusieurs primitives différentes. Un type de primitive peut être détecté en utilisant un détecteur dédié. Par exemple il existe des détecteurs de coins, d'autres de lignes ou encore de cercles. Dans la détection de points, ce sont souvent des coins qui sont recherchés. Il existe une multitude de détecteurs de coins actuellement [35], et chaque année, de nouveaux sont inventés. Cette multitude de détecteurs a été largement étudiée et classifiée [107, 100]. Dans la plus grande partie des travaux s'attelant à cette tâche, les détecteurs sont comparés en termes de performances sur architectures de type processeur comme en [84] ou l'on compare les temps d'exécution. Le nombre de points détectables est aussi souvent un critère d'évaluation. D'autres sont comparés en fonction de leur descripteur lorsqu'il est incorporé dans le détecteur comme en [78]. Enfin, des comparaisons de performances en vue d'utilisation dans des applications spécifiques sont souvent présentées. Par exemple, en SLAM, des détections précises sont recherchées avec des méthodes d'appariement efficaces [62, 39].
Afin de gagner du temps, un parcours de ces publications nous a permis de constater que ce sont souvent les mêmes détecteurs qui sont utilisés. En vue d'une application de type SLAM ou SLAN, les détecteurs les plus utilisés sont listés dans [35]. On peut trouver quatre détecteurs très régulièrement employés :

- Le détecteur de *Harris et Stephens* [45],
- Le détecteur de *Shi-Tomasi* [136],
- Le DoG [73, 74],
- Le détecteur FAST [109].

Enfin, nous pouvons ajouter le *Fast Hessian* employé dans la méthode SURF [12, 11] pour sa popularité et ses résultats en terme de performances très comparables au DoG qui est utilisé dans la méthode SIFT [73, 74]. En effet, le détecteur SURF a été développé en tant qu'amélioration du détecteur SIFT en terme de vitesse d'exécution. Il est donc intéressant de l'ajouter à cette étude. Le détecteur de Moravec sera aussi étudié en raison de sa parenté avec celui de Harris et Stephens et de sa grande simplicité de mise en œuvre. Il peut donc être un bon candidat potentiel pour une implémentation dans notre plateforme.
Parmi ces six détecteurs, nous cherchons celui qui est le plus adapté à notre système et à son architecture électronique ; l'algorithme à choisir doit présenter la meilleure adéquation algorithme architecture. De plus, le détecteur doit tenir dans l'architecture du *Front-End* afin de bénéficier des avantages de la réduction de bande passante dans le *Slip-Ring*.
Concernant les évaluations des détecteurs qui existent, aucune ne propose une évaluation complète en vue d'une implémentation dans une architecture de type FPGA. En effet, les critères comme le temps d'exécution ne sont pas les mêmes pour une exécution sur processeur que sur une architecture dédiée.

Avec une architecture de traitement dédiée, on peut optimiser le temps d'exécution en parallélisant au maximum le traitement. L'algorithme peut, en général, être organisé sous la forme d'un *Pipe-Line* où il est décomposé en maillons de traitements distincts parallélisés.

DÉMARCHE D'ÉVALUATION :

Afin de comparer les détecteurs sur une base équivalente, nous avons choisi une démarche spécifique à l'implémentation sur architecture de type FPGA. Les différents détecteurs sont donc étudiés et un schéma synoptique est construit pour chacun d'entre eux. La structure des schémas synoptiques est optimisée en terme de parallélisme et favorise l'organisation *Pipe-Line* autant que possible. Au final, les algorithmes se décomposent principalement suivant deux types d'éléments :

- Des blocs mémoires,
- Des blocs de traitement.

Sur les synoptiques, nous cherchons à dénombrer les multiplications pour savoir le nombre de multiplicateurs requis lors d'une éventuelle implémentation. Nous cherchons aussi à dénombrer les convolutions, les opérations simples qui sont des traitements ne nécessitant que peu d'éléments logiques ne sont pas dénombrées. Nous estimons aussi les latences relatives aux temps d'accumulation d'images dans les buffers mémoires. Quant à elles, les latences de traitement sont dépendantes de la fréquence de fonctionnement du système et sont négligeables par rapport aux latences de bufferisation mémoire. Par exemple, une simple convolution nécessite d'accumuler plusieurs lignes de pixels, ce qui représente plusieurs milliers de cycles d'horloges alors que l'exécution parallèle de la convolution elle même ne nécessite que quelques cycles d'horloges (autour de 2 ou 3 cycles). Il est donc hasardeux et non nécessaire de les estimer dans cette évaluation comparative.

PARAMÈTRES DE COMPARAISON :

De cette démarche d'évaluation vont découler les informations suivantes pour chaque détecteur étudié :

- Le type d'algorithme (récursif, itératif, *Pipe-Line*...),
- L'accumulation de colonnes d'image brute nécessaires pour le traitement,
- La quantité de mémoire de travail nécessaire au traitement,
- La complexité calculatoire,
- La latence générale du traitement (qui est une approximation).

Le type de l'algorithme est la première étape, bien qu'il puisse y avoir une combinaison de plusieurs types, il est nécessaire de les départager sur ce critère. Un algorithme récursif ou itératif pose plus de problèmes sur une architecture FPGA alors qu'un *Pipe-Line* y est

facilement implémenté. Pour les ressources mémoires nécessaires, deux catégories sont à prendre en compte ; la quantité de colonnes d'image nécessaires en mémoire pour commencer le traitement, et la quantité de mémoire de travail nécessaire pour le fonctionnement du système. Cette dernière est fréquemment utilisée pour stocker des morceaux d'image pré-traités, donc les deux seront exprimées en nombre de colonnes image. La complexité calculatoire sera estimée en fonction du nombre d'opérations à faire et de leurs types. Enfin, la latence du traitement (la bufferisation mémoire principalement) est aussi estimée car elle est importante en temps réel. Elle est exprimée en temps relatif à la vitesse d'acquisition des images, c'est à dire le temps nécessaire pour capturer un certain nombre de colonnes d'image. Un traitement ayant une latence de x-colonnes nous donne une idée du temps passé en fonction de la vitesse d'acquisition de la plateforme.

### 3.4.3 *Détecteur de Moravec*

DESCRIPTION GÉNÉRALE :

Un des premiers détecteurs proposés fut celui de *Moravec* [79] en 1980. Il se base sur le calcul de la Sum of Squared Differences (SSD) entre une fenêtre autour d'un pixel candidat et une autre fenêtre décalée d'une faible distance dans un certain nombre de directions.

Un *score* est ensuite associé au pixel traité et représente le minimum des SSD calculées afin d'assurer que les coins détectés soient ceux aux locations qui présentent un changement maximum. Chaque pixel se voit associé un score, un seuillage permet de sélectionner les meilleurs scores à la fin de l'algorithme. L'équation 3.1 présente le calcul à effectuer pour obtenir un *score* :

$$score = \min\{E\} \qquad E(u,v) = \sum_{x,y} w(x,y).[I(x+u,y+v) - I(x,y)]^2 \tag{3.1}$$

Avec le couple $(u,v)$ qui prend successivement les quatre valeurs différentes $(1,0)$, $(1,1)$, $(0,1)$ et $(-1,1)$. Ce qui donne un total de quatre fenêtres gradients à calculer.

La fonction $w(x,y)$ retourne 1 si les coordonnées $(x,y)$ sont dans la fenêtre traitée et 0 sinon. On a donc pour chaque pixel, une valeur C qui est calculée en fonction d'une zone d'image qui dépend de la taille de la fenêtre d'intérêt. Si la taille de la fenêtre est de $3 \times 3$, et que les décalages $(u,v)$ considérés sont de un pixel à chaque fois, on a donc besoin d'avoir accès à une fenêtre de $5 \times 4$ pour pouvoir calculer le *score* du pixel central. Cela équivaut à accumuler quatre lignes d'image et quatre pixels à tout moment puisque la restitution d'images se fait ligne par ligne et pixel par pixel.

SYNOPTIQUE :

Le schéma synoptique de l'algorithme de *Moravec* est présenté en Figure 3.13.

BILAN :

Le tableau 3.1 résume les caractéristiques de l'implémentation de ce détecteur.

Ce détecteur présente aussi les faiblesses suivantes :

- Une réponse bruitée due à la fenêtre binaire $w(x,y)$,
- Seulement un ensemble de décalage à 45 degrés est considéré,

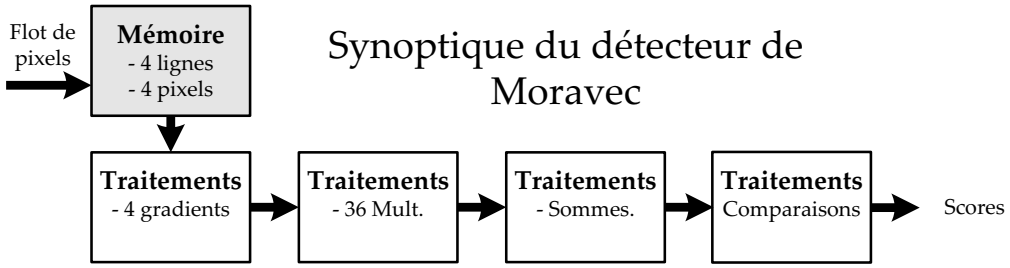


FIGURE 3.13: Synoptique de l'organisation en *Pipe-Line* du détecteur de *Moravec*

| Détecteur | Type | Mémoire image | Mémoire travail | Complexité | Latence |
|---|---|---|---|---|---|
| Moravec | *Pipe-Line* | 4 Lignes<br>4 Pixels | - | 4 Gradients,<br>27 Multiplications. | 4 Lignes<br>4 Pixels |

TABLE 3.1: Tableau récapitulatif des caractéristiques d'une implémentation du détecteur de *Moravec*

- Seulement le minimum des valeurs $E(u,v)$ est pris en compte.

### 3.4.4 *Harris & Stephens*

DESCRIPTION GÉNÉRALE :

Le détecteur développé par *Harris & Stephens* [45] en 1988 reprend le détecteur de *Moravec* et apporte une solution à ses problèmes. Il prend en compte tous les petits décalages avec le développement de Taylor suivant :

$$E(u,v) = \sum_{x,y} w(x,y).[I(x+u,y+u) - I(x,y)]^2 \qquad = \sum_{x,y} w(x,y).[I_x.u + I_y.v + O(u^2,v^2)]^2 \tag{3.2}$$

Le paramètre $O(u^2,v^2)$ est négligé, ce qui donne :

$$E(u,v) = u^2.\sum_{x,y} w(x,y).I_x^2(x,y) + v^2.\sum_{x,y} w(x,y).I_y^2(x,y) + 2.u.v.\sum_{x,y} w(x,y).I_x(x,y).I_y(x,y) \tag{3.3}$$

Une autre amélioration est le remplacement de la fonction binaire $w(x,y)$ par une fonction gaussienne $w(x,y) = exp(\frac{-(x^2+y^2)}{2.\sigma^2})$.

De plus, les petits décalages $[u,v]$ ont une approximation bilinéaire :

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix}.M.\begin{bmatrix} u \\ v \end{bmatrix} \tag{3.4}$$

Ou M est une matrice de $2 \times 2$ calculée via les dérivées de l'image (gradients verticaux et horizontaux) :

$$M = \begin{bmatrix} \sum_{x,y} w(x,y).I_x^2 & \sum_{x,y} w(x,y).I_x.I_y \\ \sum_{x,y} w(x,y).I_x.I_y & \sum_{x,y} w(x,y).I_y^2 \end{bmatrix} \tag{3.5}$$

A partir de là, nous pouvons trouver le *score* de *Harris & Stephens* avec la formule :

$$score = Det(M) - k.Trace(M)^2 \tag{3.6}$$

Avec $k$ une constante. Pour résumer l'algorithme, on peut exprimer l'ensemble selon l'équation suivante :

$$score = AB - C^2 - k.(A+B)^2 \tag{3.7}$$

Avec :

$$A = \frac{\delta I^2}{\delta x} \bigotimes w \qquad B = \frac{\delta I^2}{\delta y} \bigotimes w \qquad C = \left(\frac{\delta I}{\delta x}\frac{\delta I}{\delta y}\right) \bigotimes w \tag{3.8}$$

Avec $\bigotimes$ représentant l'opération de convolution. Chaque pixel de l'image obtient un *score*, et un seuillage permet ensuite de sélectionner les points les plus pertinents. Ce seuillage est effectué selon la formule suivante :

$$score(x,y) < \theta.max_{x,y}(score(x,y)) \tag{3.9}$$

$\theta$ étant une constante. Ce seuillage nécessite d'attendre la fin du traitement d'une image complète. Un seuil fixe est d'ailleurs souvent utilisé pour éviter d'avoir une image de latence.

SYNOPTIQUE :

Ces améliorations augmentent la complexité du calcul qui doit être fait pour chaque pixel de l'image. Ceci dit le nombre de lignes d'image à garder en mémoire est le même, c'est à dire 4 lignes et 4 pixels. La Figure 3.14 présente le synoptique de l'organisation en *Pipe-Line* de l'algorithme.

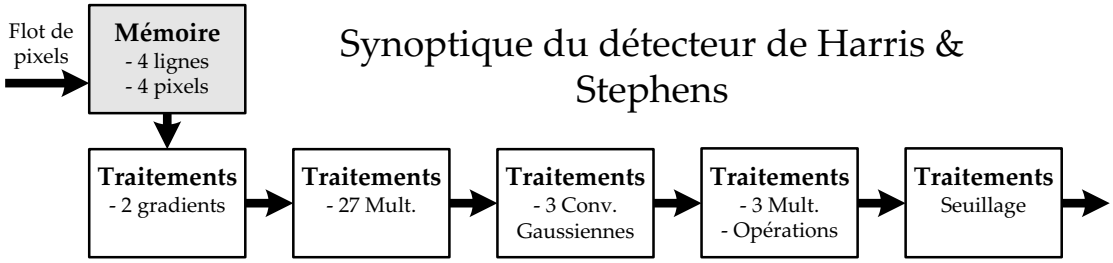


FIGURE 3.14: Synoptique de l'organisation en *Pipe-Line* du détecteur de *Harris & Stephens*

BILAN :

Les caractéristiques de cette implémentation sont résumées dans le tableau 3.2

Ce détecteur largement employé dans la communauté a aussi été modifié dans divers travaux [113].

| Détecteur | Type | Mémoire image | Mémoire travail | Complexité | Latence |
|---|---|---|---|---|---|
| Harris & Stephens | *Pipe-Line* | 4 Lignes 4 Pixels | - | 2 Gradients, 30 Multiplications, 3 Convolutions Gaussiennes. | 4 Lignes 4 Pixels |

TABLE 3.2: Tableau récapitulatif des caractéristiques d'une implémentation du détecteur de *Harris & Stephens*

### 3.4.5 *L'amélioration de Shi-Tomasi [136]*

DESCRIPTION GÉNÉRALE :

Une amélioration intéressante de l'algorithme de *Harris & Stephens* est celle présentée dans [136]. Le calcul des *scores* reste le même que dans la méthode de *Harris & Stephens*. Cependant, l'opération de seuillage est différente et une comparaison donne une valeur binaire qui définit si un point donné est pertinent à suivre.

$$\mathtt{goodToTrack}(x,y) = \min(\lambda_1,\lambda_2) > \theta.\max_{x,y}(\mathtt{score}(x,y)) \tag{3.10}$$

Cette amélioration est effective uniquement si l'opération de calcul de seuil est implémentée. Dans la grande majorité des cas, le seuil est fixe, ce qui revient à utiliser directement *Harris & Stephens*. Les paramètres $\lambda_1$ et $\lambda_2$ étant les valeurs propres de la matrice M vue dans l'équation 3.5. Comparé à l'algorithme de *Harris & Stephens*, cela requiert une opération supplémentaire. Le synoptique est donc le même que pour la version initiale de *Harris & Stephens*.

BILAN :

Les caractéristiques de cette implémentation sont résumées dans le tableau 3.3

| Détecteur | Type | Mémoire image | Mémoire travail | Complexité | Latence |
|---|---|---|---|---|---|
| Amélioration de *Shi-Tomasi* | *Pipe-Line* | 4 Lignes 4 Pixels | - | 2 Gradients, 31 Multiplications, 3 Convolutions Gaussiennes. | 4 Lignes 4 Pixels |

TABLE 3.3: Tableau récapitulatif des caractéristiques d'une implémentation du détecteur de *Shi-Tomasi*

Les détections de type *Harris & Stephens* restent sensibles au changement d'échelle et au changement d'orientation. C'est pour cela que d'autres travaux ont été effectués et des primitives plus robustes ont vu le jour. L'utilisation de DoG sur plusieurs échelles, a permis d'obtenir une amélioration significative des performances.

### 3.4.6 *Le DoG de la méthode SIFT*

DESCRIPTION GÉNÉRALE :

La méthode SIFT publiée en [73] est une méthode d'extraction complète qui comprend notamment un détecteur basé sur un DoG avec un espace des échelles, et un descripteur qui associe plusieurs paramètres aux points détectés. Nous allons seulement parler de la détection dans cette section.
La détection se divise en deux étapes [74], la création d'un espace de travail appelé *Scale Space* ou l'espace des échelles [71], puis la recherche d'extremum locaux dans cet espace. L'espace des échelles est un ensemble d'images dérivées d'une même image originelle, qu'il faut reconstruire complètement à chaque nouvelle image à traiter. L'ensemble est obtenu à partir de plusieurs séries d'images gaussiennes appelées *Octaves*. Chaque *Octave* est composée d'un nombre constant d'images ayant la même résolution, mais étant de plus en plus dégradées avec des flous gaussiens. Le passage d'une *Octave* à l'autre se fait en sous échantillonnant une des images de l'*Octave* précédente. Ainsi, on a des résolutions qui diminuent de moitié en passant d'une *Octave* à l'autre. Les images Gaussiennes consécutives dans chaque *Octave* sont sous-traitées deux à deux pour donner des images dites de DoG. L'espace tri-dimensionnel créé par la pile des images DoG ainsi obtenu, est ce qu'on appelle l'espace des échelles. En fonction du nombre d'images par *Octave* et du nombre d'*Octave*, les ressources nécessaires varient.
Le facteur d'échelle $\sigma$ est la variable qui permet de passer d'une image à l'autre dans la convolution gaussienne :

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{3.11}$$

Ensuite, les soustractions des images gaussiennes sont effectuées selon la formule :

$$DoG_{k,\sigma}(x, y) = L(x, y, k.\sigma) - L(x, y, \sigma) \tag{3.12}$$

Avec $k = 2^{\frac{1}{s}}$ et $s$ étant un entier déterminant le nombre d'images $n$ par *Octave* selon l'expression $n = s + 3$. Lorsque l'espace est construit, on le parcourt tridimensionnelle selon $x$, $y$ et $\sigma$ afin de comparer chaque pixel avec les 26 voisins qui l'entourent dans un cube de $3 \times 3 \times 3$ pixels. Lorsqu'un pixel est l'extrémum local, il est retenu comme point détecté.

SYNOPTIQUE :

La recherche en elle même est un processus relativement simple qui peut être fait combinatoirement. En effet il suffit de 26 comparaisons pour chaque triplet d'images DoG consécutives dans l'espace des échelles et pour chaque pixel de l'image de base. Cette opération doit cependant être effectuée dans les autres *Octaves*. En ce qui concerne la création de l'espace des échelles, l'algorithme est itératif et comporte un nombre d'itérations

constant en fonction du nombre d'images choisies par *Octave*. Chaque noyau gaussien est appliqué à l'image précédente dans l'*Octave*. Les itérations peuvent être réorganisées en architecture de type *Pipe-Line* vu que leur nombre est connu et constant. Ensuite un sous-échantillonnage est fait sur l'image centrale de la première *Octave* pour générer l'image de base de l'*Octave* inférieure, et les itérations gaussiennes reprennent selon le même procédé. Cette étape est répétée en fonction du nombre voulu d'*Octaves*. Cela engendre une latence plus importante que pour les algorithmes précédemment présentés. Cela dit, dans le cas ou l'on dispose de peu d'octaves, il n'est pas nécessaire de posséder l'image entière pour commencer le traitement des premiers pixels, on peut alors qualifier cette méthode de semi-globale. Mais lorsque les octaves sont nombreuses, il est nécessaire d'avoir des proportions importantes de l'image de base pour pouvoir générer la plus grande échelle, et on se rapproche des méthodes dites globales qui nécessitent des images complètes.
La Figure 3.15 propose une solution d'implémentation de la première *Octave*. Dans cet exemple, nous choisissons des paramètres minimaux avec trois *Octaves* et cinq échelles.

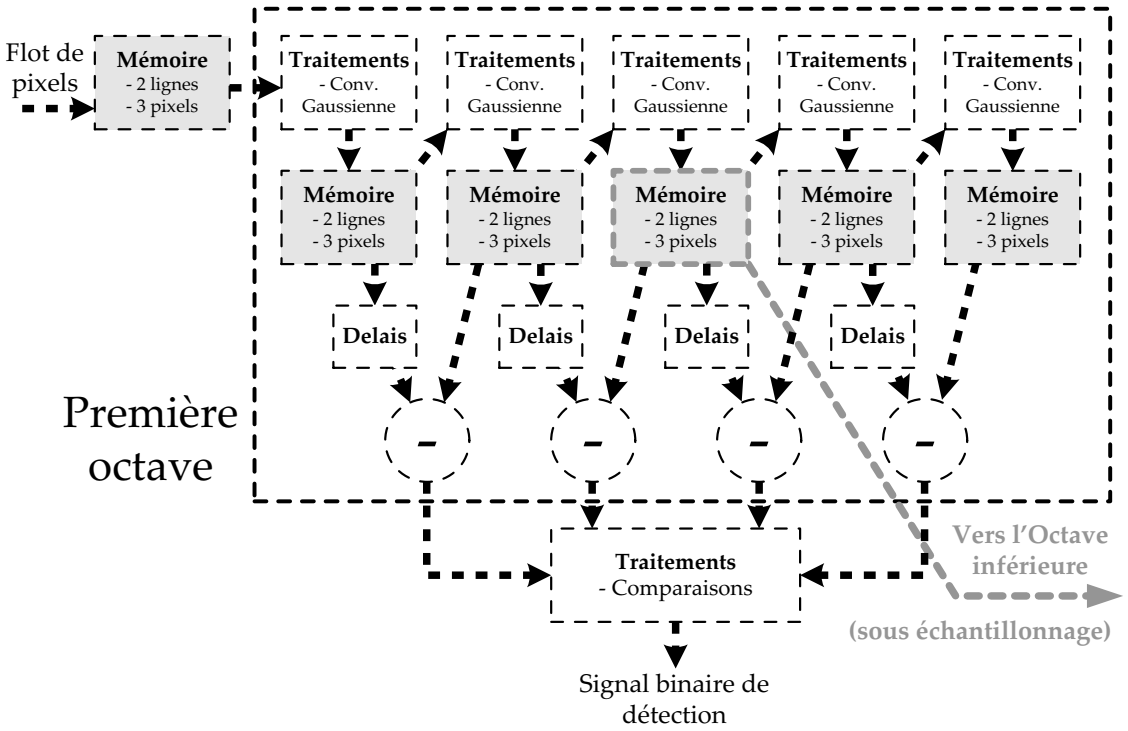


FIGURE 3.15: Organisation en *Pipe-Line* du détecteur utilisé dans SIFT. Le schéma présente une implémentation de la première octave avec cinq échelles. Le cadre en pointillé se répète pour chaque *Octave* inférieure.

On obtient un chaînage de blocs mémoires et de convolutions Gaussiennes. Chaque bloc mémoire accumule 2 lignes et trois pixels. Cependant, dans les *Octaves* inférieures, le sous échantillonnage réduit par deux la taille des lignes en mémoire. Il faut aussi tenir compte du sous échantillonnage pour calculer la latence totale (car on sous échantillonne aussi les colonnes, donc une latence de deux colonnes est nécessaire pour accumuler une seule colonne d'image sous-échantillonnée).

BILAN :

En bilan de cette implémentation, on constate que les ressources mémoire nécessaires sont bien supérieures aux autres implémentations parcourues jusqu'à présent. Les caractéristiques sont résumées dans le tableau 3.4.

| Détecteur | Type | Mémoire image | Mémoire travail | Complexité | Latence |
|---|---|---|---|---|---|
| DoG | *Pipe-Line*<br>Itératif | 2 Lignes<br>3 Pixels | 17,5 Lignes<br>45 Pixels | 15 Convolutions,<br>Soustractions,<br>Comparaisons. | 44 Lignes<br>69 Pixels |

TABLE 3.4: Tableau récapitulatif des caractéristiques d'une implémentation du détecteur de la méthode SIFT avec trois *Octaves* et cinq échelles.

Ces caractéristiques ont été calculées en fonction d'une implémentation comportant trois *Octaves* et cinq échelles comme dans le schéma 3.15. Bien que la mémoire de travail soit plus conséquente qu'avec les implémentations de type *Harris & Stephens*, on constate aussi que la latence est encore plus importante avec 44 lignes minimum de latence (sans compter la latence des calculs). Cet algorithme consomme trop de ressources mémoire et calculatoire (15 convolutions requises) pour pouvoir être intégré dans notre architecture. En effet, dans le cas de notre plateforme, une colonne image comporte 2048 pixels de 8 bits, cela donne une taille mémoire nécessaire de $2048 \times 44 = 90112$ Octets, ce qui représente 64% de ce dont nous disposons dans le FPGA. Sachant qu'une étape de filtrage et une de description (qui consomment aussi des ressources mémoire) sont nécessaires par la suite, ce choix de détecteur n'est pas adapté. Les mémoires externes peuvent cependant être utilisées, mais la latence augmente encore plus car il faut stocker plus de deux lignes après chacune des 15 convolutions Gaussiennes. Enfin, l'exemple proposé comporte seulement trois *Octaves*, l'algorithme est donc minimaliste dans notre cas et l'ajout d'*Octaves* supplémentaires fait grandir la latence et les besoins en mémoire de façon exponentielle.

### 3.4.7 *Fast Hessian*

DESCRIPTION GÉNÉRALE :

Le détecteur SIFT est basé sur un algorithme itératif, cela a pour conséquence de créer une grande latence. Pour améliorer le temps d'exécution, une autre méthode utilisant un espace d'échelle a été présenté par Herbert Bay et al. [12] en 2006. Cette méthode appelée SURF est très similaire à la précédente mais présente une architecture de traitement qui peut être complètement parallélisée, ce qui représente un net avantage pour notre problématique [11]. Cette fois, l'espace des échelles est construit à partir de la matrice hessienne de l'image, qui à l'échelle $\sigma$ est définie par l'équation :

$$H(x,y,\sigma) = \begin{bmatrix} \frac{\partial^2}{\partial x^2} G(\sigma) \otimes I(x,y) & \frac{\partial}{\partial x}\frac{\partial}{\partial y} G(\sigma) \otimes I(x,y) \\ \frac{\partial}{\partial x}\frac{\partial}{\partial y} G(\sigma) \otimes I(x,y) & \frac{\partial^2}{\partial y^2} G(\sigma) \otimes I(x,y) \end{bmatrix} \tag{3.13}$$

Le *score* du détecteur correspond au déterminant de cette matrice. Cela dit, la dérivée seconde d'un gaussien est une opération complexe et Bay et al. [12] proposent une approximation qui va accélérer les calculs. trois masques sont donc utilisés pour représenter les opérations $\frac{\partial^2}{\partial x^2}G(\sigma)$, $\frac{\partial^2}{\partial y^2}G(\sigma)$ et $\frac{\partial}{\partial x}\frac{\partial}{\partial y}G(\sigma)$ respectivement. Les masques sont simplement convolués à l'image pour obtenir la matrice Hessienne. Le *score* est ensuite calculé pour chaque pixel selon sa matrice Hessienne avec la formule :

$$score(x,y,\sigma) = D_{xx}(\sigma) \times D_{yy}(\sigma) - (0.9 \times D_{xy}(\sigma))^2 \tag{3.14}$$

Où $D_{xx}$, $D_{xy}$ et $D_{yy}$ sont respectivement les résultats des convolutions des trois masques avec l'image. L'espace des échelles est généré selon cette méthode. Pour chaque *Octave*, les différentes images sont générées à partir de l'image originelle et en appliquant les masques avec des tailles de plus en plus grandes ($9 \times 9$, $15 \times 15$, puis $21 \times 21$ etc...) De cette manière, la première *Octave* est créée. Ensuite l'image d'origine est sous échantillonnée pour chaque *Octave* inférieure et l'application des masques est effectuée de manière similaire. Les masques croissent cependant de 12 en 12 au lieu de croitre de 6 en 6.. A chaque *Octave* inférieure, les masques croissent à chaque fois d'une croissance qui est le double de la croissance de l'*Octave* précédente. Enfin, quand l'espace des échelles est généré, la recherche des extrémums se fait tri-dimensionnement de la même manière qu'avec la méthode DoG, sauf qu'en plus le *score* sert de sélection des candidats avec une étape ultime de seuillage.

SYNOPTIQUE :

Le synoptique de l'implémentation est présenté sur la Figure 3.16.

Afin de comparer avec le DoG, nous avons choisi le même nombre d'*Octaves* et le même nombre d'images par *Octave*. Tous les traitements sont effectués à partir de l'image de base, il n'y a plus de chaînage ici. La latence de traitement disparait donc, mais la latence d'accumulation subsiste et est même beaucoup plus importante. En effet, sur architecture de type processeur, l'image est complète lors du début du traitement, ce qui explique que SURF soit plus rapide. Cela dit, sur une architecture hardware, le traitement commence alors que l'image est en cours de restitution, le DoG a donc l'avantage.

BILAN :

En bilan, les caractéristiques et les ressources nécessaires à cette implémentation sont résumées dans le tableau 3.5.

| Détecteur | Type | Mémoire image | Mémoire travail | Complexité | Latence |
|---|---|---|---|---|---|
| SURF | parallèle<br>*Pipe-Line* | 26 Lignes<br>27 Pixels | 124 Lignes<br>126 Pixels | 1788 Convolutions,<br>36 Multiplications,<br>Comparaisons. | 323 Lignes<br>324 Pixels |

TABLE 3.5: Tableau récapitulatif des caractéristiques d'une implémentation du détecteur employé dans la méthode SURF

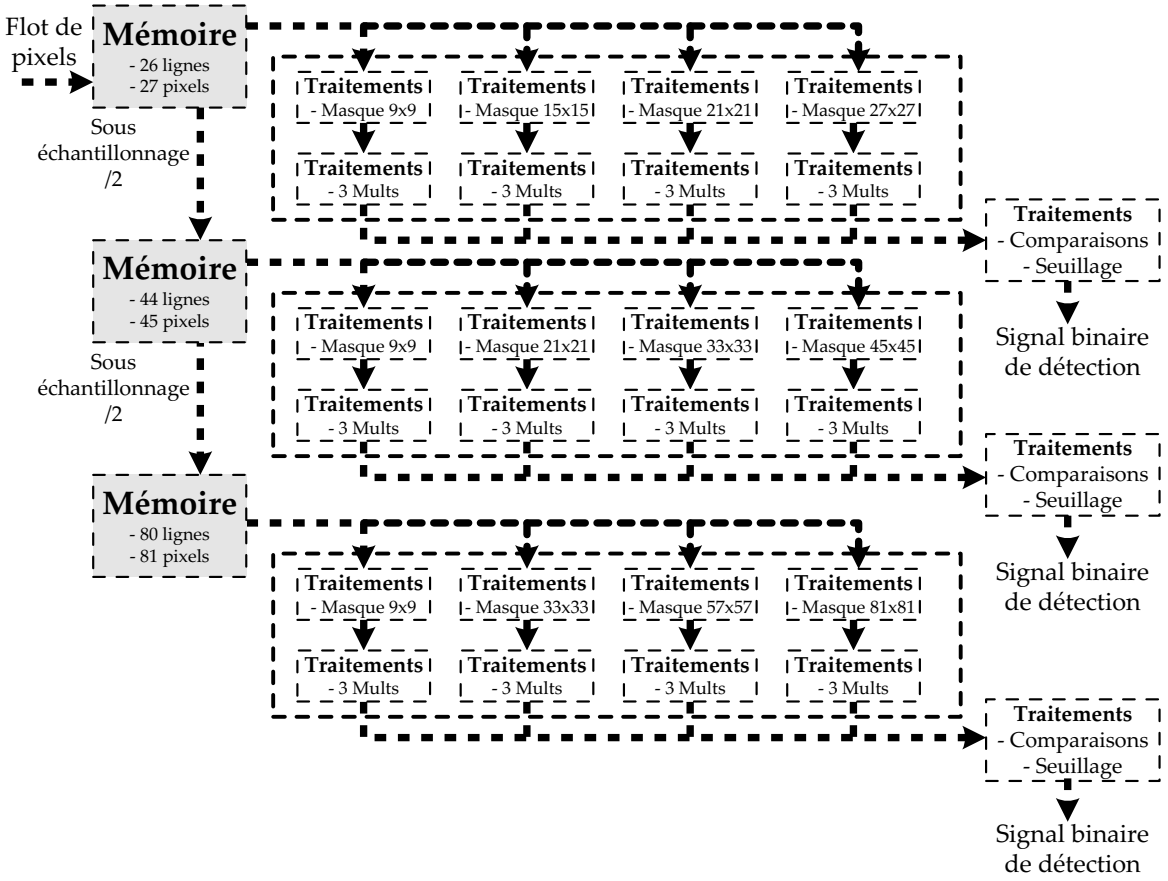


FIGURE 3.16: Synoptique de l'organisation parallèle du détecteur SURF

Les masques ont été exprimés en terme de convolution pour rester homogènes avec les autres algorithmes étudiés. Ainsi, l'application d'un masque de $3 \times 3$ équivaut à une convolution (car elles ont des tailles de $3 \times 3$ depuis le début de cette étude). Nous avons calculé le nombre de convolutions $3 \times 3$ nécessaires pour avoir l'équivalent de tous les masques décrits dans la figure 3.16. Le détecteur SURF consomme beaucoup trop de ressources de mémoire et de calcul, il est pas envisageable de pouvoir l'implémenter dans notre architecture.

### 3.4.8 *FAST*

DESCRIPTION GÉNÉRALE :

Enfin, un dernier détecteur intéressant est le détecteur FAST [109]. Ce détecteur nécessite d'accumuler six lignes d'image et cinq pixels pour pouvoir traiter le premier pixel. Le test opère en considérant un cercle de 16 pixels autour du coin candidat $p$. Le coin $p$ est considéré comme un coin si un ensemble de $n$ pixels consécutifs du cercle sont tous plus clairs que l'intensité du pixel central $I_p$ plus un seuil $t$, ou plus sombres que le pixel central $I_p - t$. Il est possible d'utiliser plusieurs valeurs pour $n$, les plus employées étant comprises entre 12 et 9.

Sur la figure 3.17 (a), nous avons 11 pixels consécutifs qui vérifient cette condition, dans le cas d'un FAST-9 nous avons donc un coin de détecté, mais pas dans le cas d'un FAST-12 par exemple. De plus, sur la figure 3.17 (c) nous avons un coin qui correspond à

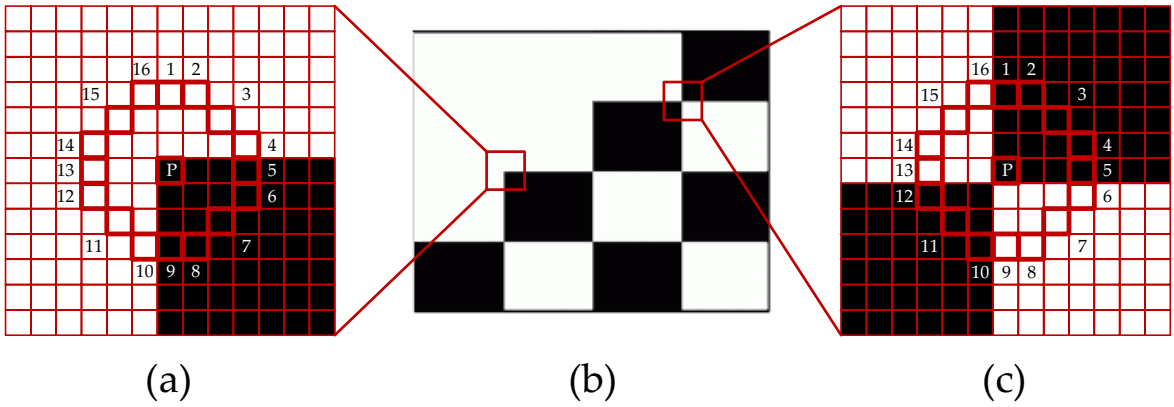


FIGURE 3.17: Exemple de detection avec l'algorithme FAST : (b) image traitée. (a) Primitive détectée. (c) Primitive non prise en charge par l'algorithme.

un croisement du damier de l'image, et il ne peut pas être détecté selon l'algorithme FAST. Cette méthode présente donc un inconvénient qui est de ne détecter qu'un petit ensemble de coins contrairement aux autres algorithmes étudiés plus haut qui les détectent tous. Cela dit, FAST est très facile à implémenter et très rapide grâce à sa simplicité calculatoire qui ne comporte que des comparaisons et des tests logiques binaires. Enfin, il n'y a pas de *score* associé aux primitives FAST, il n'est donc pas possible de filtrer les *scores* non maximums.

SYNOPTIQUE :

Le synoptique d'implémentation est donné en figure 3.18.

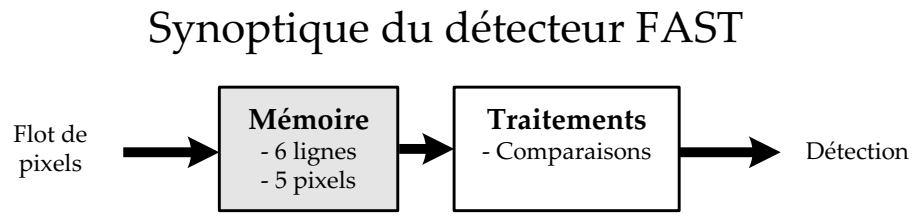


FIGURE 3.18: Synoptique de l'organisation en *Pipe-Line* du détecteur FAST

La simplicité d'implémentation de FAST est son principal avantage, seules une étape d'accumulation et une étape de traitement combinatoire sont nécessaires.

BILAN :

Les ressources et les caractéristiques du détecteur FAST sont résumées dans le tableau 3.6.

On constate ici que l'algorithme FAST est celui qui consomme le moins de ressources. Sa consommation mémoire et sa latence sont cependant légèrement plus importantes que pour les algorithmes de type *Harris & Stephens*. De plus, nous avons montré que certains cas de coins ne sont pas détectés. Ces raisons font que le détecteur FAST n'est pas un bon candidat en comparaison au détecteur de *Harris & Stephens* par exemple.

| Détecteur | Type | Mémoire image | Mémoire travail | Complexité | Latence |
|---|---|---|---|---|---|
| FAST | *Pipe-Line* | 6 Lignes<br>5 Pixels | - | Comparaisons | 6 Lignes<br>5 Pixels |

TABLE 3.6: Tableau récapitulatif des caractéristiques d'une implémentation du détecteur FAST

### 3.4.9 *Bilan comparatif*

Le tableau 3.7 résume les informations que nous avons calculées pour tous les détecteurs précédents.

| Détecteur | Type | Mémoire image | Mémoire travail | Complexité | Latence |
|---|---|---|---|---|---|
| Moravec | *Pipe-Line* | 4 Lignes<br>4 Pixels | - | 4 Gradients,<br>27 Multiplications. | 4 Lignes<br>4 Pixels |
| Harris & Stephens | *Pipe-Line* | 4 Lignes<br>4 Pixels | - | 2 Gradients,<br>30 Multiplications,<br>3 Convolutions Gaussiennes. | 4 Lignes<br>4 Pixels |
| Détecteur de *Shi-Tomasi* | *Pipe-Line* | 4 Lignes<br>4 Pixels | - | 2 Gradients,<br>31 Multiplications,<br>3 Convolutions Gaussiennes. | 4 Lignes<br>4 Pixels |
| DoG | *Pipe-Line*<br>Itératif | 2 Lignes<br>3 Pixels | 17,5 Lignes<br>45 Pixels | 15 Convolutions,<br>Soustractions,<br>Comparaisons. | 44 Lignes<br>69 Pixels |
| SURF | parallèle<br>*Pipe-Line* | 26 Lignes<br>27 Pixels | 124 Lignes<br>126 Pixels | 1788 Convolutions,<br>36 Multiplications,<br>Comparaisons. | 323 Lignes<br>324 Pixels |
| FAST | *Pipe-Line* | 6 Lignes<br>5 Pixels | - | Comparaisons | 6 Lignes<br>5 Pixels |

TABLE 3.7: Tableau comparatifs des six algorithmes étudiés.

Nous avons vu que les détecteurs invariants à l'échelle sont ceux qui génèrent un espace des échelles (SIFT et SURF). Ces deux détecteurs sont donc les plus performants mais consomment aussi beaucoup trop de ressources et leurs latences sont aussi les plus grandes. Ils ne sont pas retenus pour notre implémentation. Du côté du détecteur FAST, il est celui qui consomme le moins de ressources et sa latence n'est que de 6 lignes et quelques. Cependant, il n'est capable de détecter qu'une partie des coins de l'image, comme on l'a vu plus haut sur la figure 3.17, il n'est donc pas retenu non plus. Le détecteur de *Moravec*, quant à lui présente trois défauts qui ont été corrigés par l'algorithme de *Harris & Stephens*, et leur consommation est similaire. Le détecteur de *Harris & Stephens* l'emporte donc sur celui de *Moravec* et des quatre autres cités. Cela dit, nous n'utilisons pas l'étape de seuillage finale telle qu'elle est présentée par *Harris & Stephens* ou *Shi-Tomasi* afin d'éviter le traitement global de l'image, et c'est la seule chose qui diffère de l'un à l'autre. Ainsi, nous choisissons l'algorithme de *Harris & Stephens* sans sa méthode de seuillage car il est celui le plus en adéquation avec notre architecture d'accueil. Nous avons modifié l'étape de seuillage et nous l'expliquons en détail dans la section suivante.

## 3.5 IMPLÉMENTATION DE L'ALGORITHME DE HARRIS ET STEPHENS

### 3.5.1 *Problématique d'implémentation*

L'algorithme a intégralement été implémenté dans l'architecture FPGA du *Front-End* malgré ses ressources très limitées. Les composants de type FPGA sont idéaux pour les implémentations de type traitement des images [69]. Leurs avantages sont :

- Une grande polyvalence/flexibilité,
- La possibilité de paralléliser les traitements,
- Une adéquation optimale entre algorithme et architecture.

Beaucoup de travaux présentent déjà des implémentations de détecteurs de primitives [41]. Plus précisément, quelques travaux [22] ont déjà abordé la problématique de l'implémentation du détecteur de *Harris & Stephens*. Les travaux de Lidholm et al. [70] décrivent une implémentation du détecteur stéréoscopique. L'implémentation comporte un détecteur et un descripteur, puis utilise une nouvelle approche d'appariement pour calculer la pose de la caméra. Un des problèmes liés au détecteur de *Harris & Stephens*, est qu'il est fréquent qu'il détecte plusieurs pixels adjacents comme points d'intérêts. Pour sélectionner le point le plus représentatif (typiquement, celui qui a le meilleur score), il faut utiliser un module de traitement supplémentaire qui va filtrer les scores et laisser le maximum uniquement. Dans les travaux de Benedikt Dietrich [32] une implémentation qui propose une solution à ce problème est présentée. Après la détection, un filtre supprime les non maximas sur des fenêtres de trois par trois. Ce qui veut dire qu'il n'y a pas plus d'une détection pour chaque configuration de trois pixels carré. Cette étape dite de filtrage est essentielle et n'est pas présente dans toutes les implémentations proposées.
Ces implémentations présentées ont toutes été faites sur des architectures classiques comportant des caméras conventionnelles. Notre système quant à lui, possède plusieurs particularités que ne vont pas permettre l'implémentation classique de l'algorithme. Premièrement, il faut prendre en compte le fait que le sens de restitution de l'image se fait colonne par colonne et non ligne par ligne. De plus, il n'y a pas de début ni de fin aux images cylindriques par définition. Le seul début se présente au démarrage du système lorsque la première colonne est capturée. Ensuite, les captures sont continues pendant la rotation et toutes les colonnes acquises se placent bout à bout comme pour former une image continue perpétuelle. Il est donc possible de traiter les données comme appartenant à une unique image et ainsi de ne jamais avoir à rencontrer de bords.

Du coté des ressources, nous disposons des quantités suivantes sur la carte du *Front-End* :

- 1134 kilo bits de mémoire interne au FPGA (équivalent à 70 colonnes d'image pour notre système),
- 126 Multiplicateurs embarqués dans le FPGA,
- 39600 Éléments logiques pour l'implémentation,
- 3 composants SRAM de 2 MO chacun,

- Une FIFO de 64 kO.

Ces ressources sont suffisantes pour une implémentation complète de l'algorithme, et nous n'auront pas besoin d'utiliser les SRAM et la FIFO externes au FPGA.

### 3.5.2 *Architecture générale de l'implémentation*

L'implémentation que nous proposons reprend nos travaux antérieurs sur des caméras classiques [19] et propose plusieurs améliorations, ainsi qu'une nouvelle approche adaptée aux systèmes cylindriques. Elle se base sur une implémentation du détecteur de *Harris & Stephens* et d'un descripteur. Cette implémentation se divise en quatre blocs comme le présente la Figure 3.19.

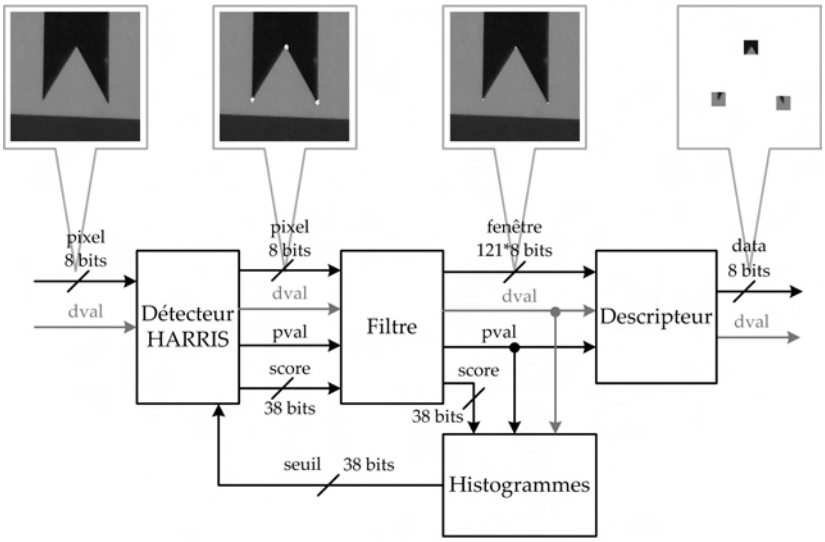


FIGURE 3.19: Architecture en quatre blocs de l'implémentation proposée de l'algorithme de Harris et Stephens. Une illustration de l'état de l'image à chaque étape du traitement est précisée sur la partie supérieure du schéma.

La détection est effectuée par le premier bloc et respecte la formule $R = AB - C^2 - k.(A + B)^2$ comme nous l'avons décrite dans la section précédente. Le seuillage de la détection a pour conséquences de faire apparaître plusieurs détections autour des primitives comme le montre la figure, une étape de filtrage est donc proposée afin de sélectionner le meilleur score du lot. Ensuite, l'étape de description associe deux types de données aux points détectés, une fenêtre de $11 \times 11$ pixels et l'adresse dans l'image du point en question. Ces données permettent d'avoir les informations nécessaires pour reconnaître le point dans une autre image. Elles sont concaténées dans une trame et sont sérialisées en sortie. Enfin, notre méthode de seuillage dynamique automatique est implémenté dans le dernier bloc *Histogrammes* et modifie le seuil appliqué au bloc *Détecteur* afin d'asservir le nombre de points détectés par image.

### 3.5.3 *Le bloc Détecteur*

L'étape de détection peut se découper en multiples sous étapes successives, et il n'y a pas de boucles dans le traitement. Pour ces raisons, il est aisé de l'implémenter sous la forme d'un grand *Pipe-Line*. Un *Pipe-Line* est une chaîne de traitement ou chaque maillon traite une partie des données avant de la transmettre au maillon suivant, cela facilite le traitement et optimise les temps de calculs. Les architectures de type FPGA sont propices à ce genre d'implémentation, nous avons donc profité des avantages de ces architectures et avons implanté un *Pipe-Line* en six étapes. La figure 3.20 présente cette implémentation en six étages.

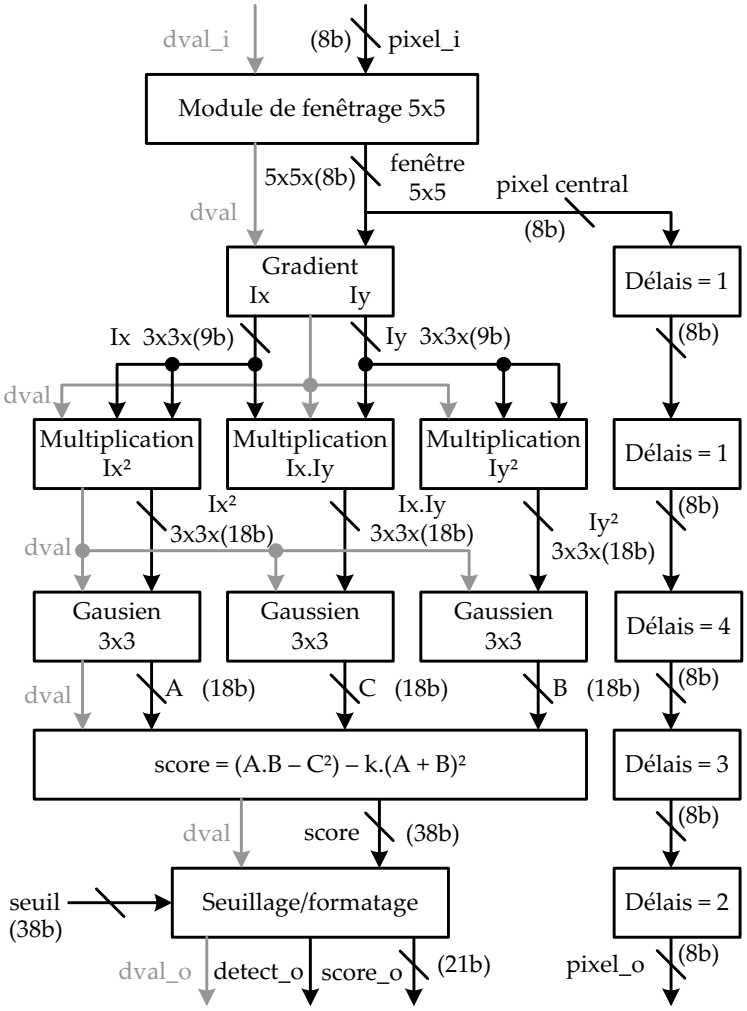


FIGURE 3.20: Architecture détaillée du bloc *Détecteur* de notre implémentation de l'algorithme de Harris

Le bus utilisé est toujours le bus vidéo que nous avons décri plus haut. Il comporte un signal d'horloge, un signal de validation et un bus de données. La taille de ce dernier sera cependant amenée à augmenter au fur et à mesure des calculs. Le bloc de détection ajoute deux nouveaux signaux à ce bus ; le signal *score_o* qui est la valeur du score de Harris ainsi calculée, et le signal *pval_o* qui est un nouveau signal de validation. Ce signal indique si le

score du point courant est au dessus du seuil dynamique, et donc s'il est considéré comme un point d'intérêt.

### ÉTAGE 1 : MODULE DE FENÊTRAGE

La première étape est une étape d'accumulation de données. En effet, l'algorithme de *Harris & Stephens* utilise des noyaux de convolution et nécessite donc de pouvoir accéder à des fenêtres de pixels de l'image. Or les pixels arrivent un par un à une cadence de 160 MHz dans notre système. Il est donc nécessaire d'accumuler un certain nombre de pixels dans une mémoire qui pourra être lue d'un coup au moment voulu. Deux opérations vont contraindre la quantité de lignes d'image que nous devons garder en mémoire à tout instant : le calcul des gradients et la convolution Gaussienne. Pour la convolution Gaussienne, nous avons besoin de posséder une fenêtre carrée d'image gradient de 3 pixels de large. Or pour calculer la valeur de chaque pixel de cette fenêtre, il faut une fenêtre carrée de 3 pixels de large aussi. Donc il faut pouvoir accéder à des fenêtres de $5 \times 5$ pixels à tout moment. Cela nous oblige à avoir en mémoire l'équivalent de quatre lignes image et cinq pixels. La figure 3.21 décrit l'architecture employée pour satisfaire cette nécessité.

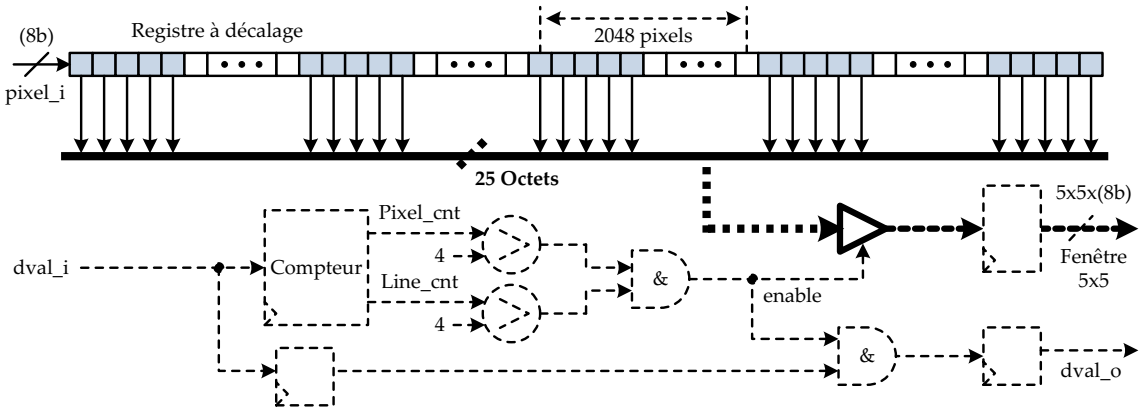


FIGURE 3.21: Description du module de fenêtrage

En sortie de ce module de fenêtrage, nous avons une fenêtre de $5 \times 5$ pixels de 8 bits à chaque coup d'horloge, les ressources occupées en mémoire sont optimisées avec cette méthode. Il faut cependant considérer une latence correspondant au temps nécessaire à la capture de 4 lignes et de 5 pixels. Or l'intégration des colonnes d'image est synchronisée avec la position du plateau dans notre système. Donc la latence de cette étape dépend de la vitesse de rotation du système.

### ÉTAGE 2 : GRADIENTS

Le calcul des gradients peut ensuite s'effectuer. Deux gradients sont implémentés : un gradient vertical $I_Y$ et un gradient horizontal $I_X$. Cette étape est exécutée en un cycle d'horloge et donne en sortie des fenêtres de gradients comportant $3 \times 3$ pixels.

### ÉTAGE 3 : MULTIPLICATIONS

L'étape suivante est la multiplication des fenêtres gradients entre elles. Trois nouvelles fenêtres de $3 \times 3$ pixels doivent être calculées à partir des deux fenêtres gradients.

$$\frac{\delta I^2}{\delta x} \qquad \frac{\delta I^2}{\delta y} \qquad \frac{\delta I}{\delta x}\frac{\delta I}{\delta y} \tag{3.15}$$

Lors de ces multiplications, il faut multiplier les valeurs des pixels ayant la même position dans les deux fenêtres opérandes. Comme il y a neuf pixels par fenêtre et trois couples fenêtres à multiplier, cela donne donc 27 multiplications à effectuer en parallèle. Il est possible de les calculer en un cycle d'horloge en utilisant les multiplicateurs du FPGA. Il en faut 27 et la latence du calcul ne prendre qu'un cycle d'horloge.

### ÉTAGE 4 : GAUSSIENS

Une convolution Gaussienne vient ensuite filtrer les données et permet de calculer les trois valeurs A, B et C. La figure 3.22 résume notre implémentation de cette étape.

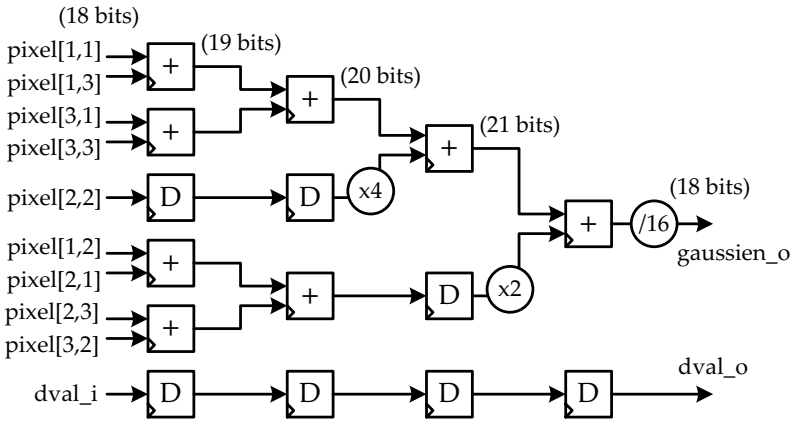


FIGURE 3.22: Architecture de la convolution Gaussienne implémentée, cette étape du calcul est réalisée en quatre cycles d'horloge.

La latence est de quatre cycles d'horloge.

### ÉTAGE 5 : CALCUL DES SCORES

Ainsi nous avons trois nombres scalaires A, B et C, il faut maintenant les combiner selon la formule $R = AB - C^2 - k.(A + B)^2$

Ce calcul comporte cette fois-ci des multiplications et des additions dont les résultats s'enchainent. Une pyramide de calculs combinatoires à trois étages est possible pour résoudre ce problème, la figure 3.23 en présente l'implémentation.

Le calcul nécessite trois cycles d'horloge. Au premier cycle, une addition et deux multiplications nous donnent les valeurs $(A + B)/4$, $AB$ et $C^2$. Ensuite, le deuxième cycle permet de calculer $(A + B)^2/8$ et $AB - C^2$ avec une multiplication et une soustraction. Enfin la formule totale est calculée au dernier cycle. Cette implémentation nécessite donc trois multiplications, ce qui nous fait un total de 30 blocs multiplicateurs en tout. Le score final calculé est codé en binaire signé sur 38 bits, comme détaillé sur la figure 3.23.

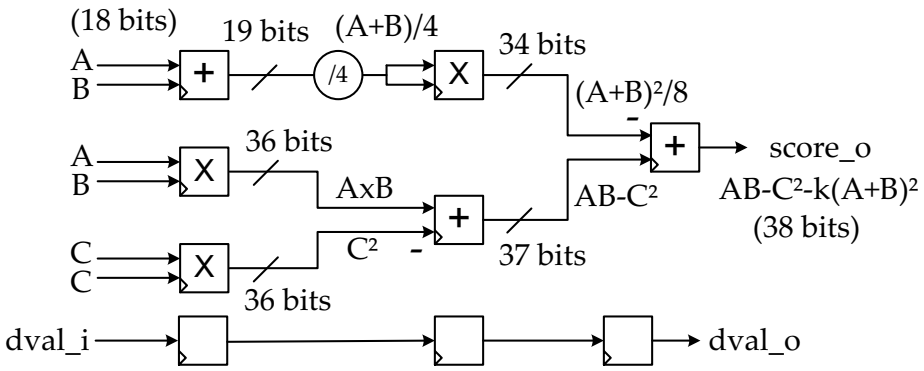


FIGURE 3.23: Illustration de l'étape finale du calcul des scores de Harris et Stephens. Trois cycles d'horloge sont nécessaires à la réalisation de ce calcul.

### ÉTAGE 6 : SEUILLAGE ET MISE EN FORME

Le score de *Harris & Stephens* est maintenant calculé et comporte 38 bits. Dans un premier temps, il faut le comparer au seuil afin de déterminer si le pixel en cours de traitement est un point suffisamment important, c'est à dire s'il représente un coin dans l'image ou non. Le score est alors comparé au seuil dynamique qui est envoyé à l'entrée de ce bloc. Nous verrons plus tard comment ce seuil dynamique est calculé à chaque panorama.

Le signal *pval_o* indique si le point courant est au dessus du seuil et le signal *score_o* garde le score de Harris. En effet, nous verrons aussi plus loin que nous avons besoin de garder en mémoire ce score pendant quelques lignes, il est donc intéressant de diminuer au maximum son occupation des mémoires limitées du FPGA. Premièrement, le formatage du signal doit être converti en "non-signé", on gagne ainsi un bit en moins puisque les scores finaux sont forcément positifs. Les neuf bits de poids fort et les 8 bits de poids faible contiennent une information négligeable. Ainsi, les neuf bits de poids fort sont supprimés et une saturation est effectuée dans le cas ou l'un de ces 9 bits était à l'état haut. Puis le score est divisé par 256 en supprimant les 8 derniers bits. Ce qui nous donne un score sur 21 bits, soit un gain de 19 bits sans altérer l'information contenue. Enfin, un dernier traitement élimine les points détectés trop près des bord supérieurs et inférieurs de l'image panoramique car il est impossible d'avoir le voisinage de $11 \times 11$ pixels autour d'un de ces pixels. Ces traitements prennent deux cycles d'horloge.

### RESSOURCES CONSOMMÉES PAR LE DÉTECTEUR COMPLET

L'ensemble des six traitements que nous venons de décrire en détail représente notre implémentation de la partie Détecteur de l'algorithme de *Harris & Stephens*, comme il était présenté sur le figure 3.20. Sur la figure, on constate aussi que le signal du pixel central de la fenêtre de $5 \times 5$ sortant du bloc FIFO est retardé de 13 cycles d'horloges. Le nombre 13 correspond au nombre de cycles nécessaires pour traverser les cinq derniers blocs du *Pipe-Line*. Ainsi, le niveau de gris du pixel central est retardé pour rester synchronisé avec le score de *Harris & Stephens* qui lui correspond. Il est nécessaire d'avoir ces signaux syn-

chronisés pour la suite du traitement. L'occupation totale du détecteur dans le FPGA est présentée dans le tableau 3.8 :

| **Type de Ressource** | **Occupation** | |
|---|---|---|
| Total Logic Elements | 1,634/39,600 | (4%) |
| Total Memory Bits | 65,360/1,161,216 | (6%) |
| Embedded Multipliers | 33/252 | (13%) |

TABLE 3.8: Le taux d'occupation du FPGA par l'implémentation du bloc de détection

### 3.5.4 *Le bloc Filtre*

Après la détection, que nous venons de décrire, nous avons le flot pixel qui se trouve enrichi de deux données supplémentaires :

- Un signal binaire *pval_o* qui indique si le pixel en cours est un point d'intérêt,
- Un signal *score_o* codé sur 21 bits qui contient la valeur du score de *Harris & Stephens* uniquement lorsque le signal *pval_o* est à l'état haut.

Lors de l'étape de détection, les points obtiennent tous un score qui va déterminer s'ils sont des points d'intérêts. Au voisinage d'un coin sur l'image, il est fréquent d'obtenir plusieurs détections comme illustré en figure 3.24 (a), il en résulte l'apparition d'un blob de détection.

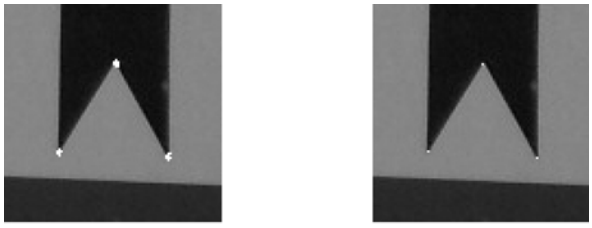


FIGURE 3.24: Illustration de l'allure d'une détection en sortie du bloc *Détecteur*. Les points blancs sont les points ayant passé l'étape de seuillage. (a) de multiples détections sont apparues au voisinage du coin. (b) Après un filtrage, seul le point le plus pertinent est sélectionné, les autres détections sont ignorées.

L'intérêt du filtre est de supprimer toutes les détections secondaires du même coin et de ne laisser que la détection avec le meilleur score (Figure 3.24 (b)). Pour ce faire, nous avons choisi de filtrer sur des fenêtres de $11 \times 11$ pixels. Cette taille de fenêtre correspond à la taille des patchs que nous assemblerons dans la partie description suivante. La figure 3.25 représente l'implémentation de ce filtre.

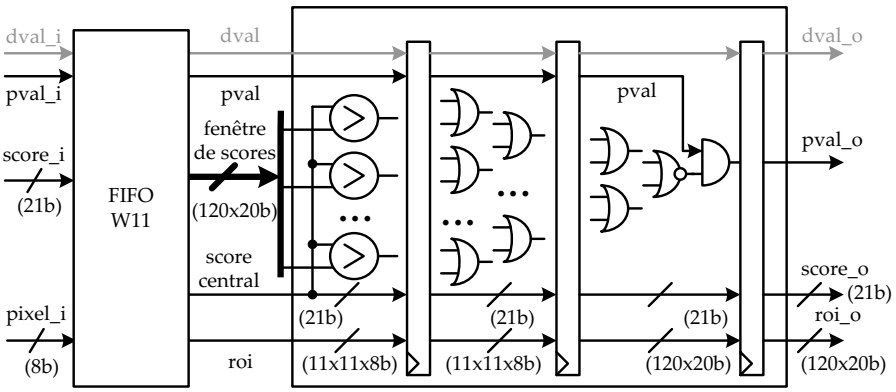


FIGURE 3.25: Architecture détaillée du bloc "Filter" de notre implémentation de l'algorithme de Harris

Comme le filtrage s'effectue sur des fenêtre de $11 \times 11$ pixels, nous devons dans un premier temps accumuler 10 lignes d'image, c'est à dire $10 \times 2048 + 11$ pixels ainsi que leurs scores de détection. Cela représente 20491 pixels soit $20491 \times (8 + 1 + 21)$, ce qui donne 614730 bits. Cette quantité de données doit être gardée en permanence en mémoire, il paraît totalement judicieux d'avoir diminué la taille des scores au maximum. En effet, avec la diminution de la taille des scores de *Harris & Stephens* (de 38 bits à 21 bits) on préserve 350 kilo bits de mémoire, ce qui représente près d'un tiers de la mémoire interne au FPGA employé. Une FIFO de 10 lignes sur le même schéma d'implémentation que la précédente a été implémentée. A chaque cycle d'horloge, une fenêtre de $11 \times 11$ pixels est émise en sa sortie. Tous les scores des pixels de cette fenêtre sont comparés au score du pixel central avec une pyramide de portes "ou". Le nombre de portes étant trop important pour effectuer ce traitement en combinatoire en seulement un cycle d'horloge selon l'analyse *timing*.

Le système devant fonctionner à une cadence de 160 MHz, il a donc fallu insérer deux registres pour segmenter la chaine combinatoire. Ainsi cette chaine s'effectue en trois cycles d'horloge. En sortie, le signal *pval_o* est mis à '0' si un pixel de la fenêtre comporte un score plus élevé que le pixel central. Sinon le signal *pval_o* reste à '1'. Avec ce système il n'y a jamais plus d'un point de *Harris & Stephens* dans toutes les fenêtres de $11 \times 11$ possibles dans l'image. Les ressources du FPGA occupées par ce bloc de filtrage sont listées dans le tableau 3.9

| **Type de Ressource** | **Occupation** | |
|---|---|---|
| Total Logic Elements | 11,313/39,600 | (29%) |
| Total Memory Bits | 600,296/1,161,216 | (52%) |
| Embedded Multipliers | 0/252 | (0%) |

TABLE 3.9: Le taux d'occupation du FPGA par l'implémentation du bloc de filtrage

On peut constater qu'a lui seul ce bloc occupe plus de la moitié des ressources mémoires et près d'un tiers des éléments logiques du FPGA.

### 3.5.5 *Le bloc Descripteur*

Le descripteur est une étape relativement simple qui consiste à mettre en forme les données pour l'envoi à la couche de communication. Le but est de fournir une image de $11 \times 11$ pixels autour des points détectés, c'est à dire ceux qui ont passé le seuil du détecteur et qui n'ont pas été filtrés. Cela représente donc 121 pixels de 8 bits qui sont empaquetés. Des informations supplémentaires sont ajoutées au paquet, à savoir les coordonnées X et Y du point dans l'image, et le score de *Harris & Stephens* sur 21 bits. La trame ainsi générée a l'allure de celle présentée en figure 3.26.

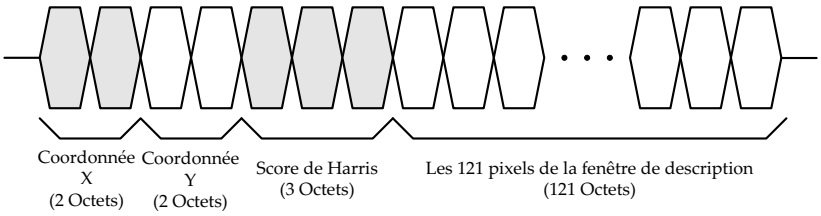


FIGURE 3.26: Organisation de la trame contenant le descripteur d'un point d'intérêt

Le score de *Harris & Stephens* est codé sur 21 bits pour sauvegarder la mémoire interne au FPGA lors de la bufferisation mémoire, cependant il est ensuite codé sur 3 octets dans la trame afin d'avoir un nombre entier d'octets. La figure 3.27 présente l'architecture du descripteur générant ces trames.

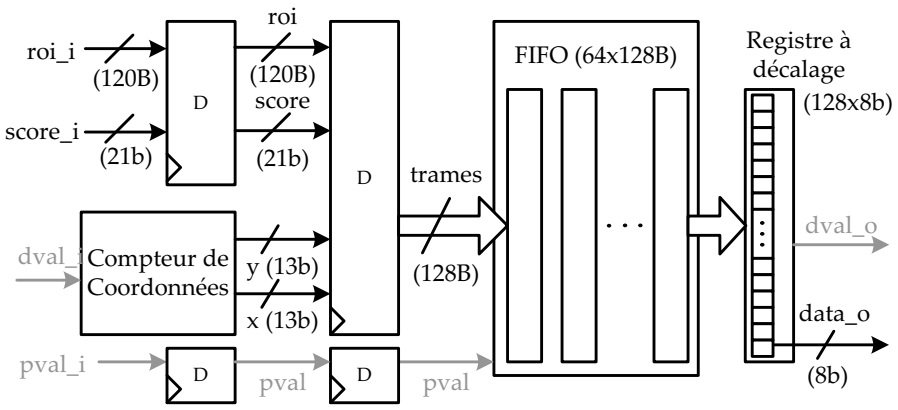


FIGURE 3.27: Architecture détaillée du bloc *Descripteur* de notre implémentation de l'algorithme de Harris

Une fois un paquet prêt à l'envoi, il est mis en attente dans une FIFO qui peut en garder plusieurs avant de les envoyer. Cela donne une certaine souplesse à la transmission car un goulot d'étranglement se situe au niveau du Slip-Ring. Bien que celui-ci ait un débit élevé, il ne peut transmettre qu'un octet à la fois. C'est pour cela qu'un registre à décalage transmet les octets du paquet en cours d'émission un à un.
Concernant l'entrée du bloc de description, pour avoir les 121 pixels en un cycle d'horloge, il faut avoir accumulé au moins 10 lignes d'images auparavant. Cela consomme énormément de ressources mémoires de le faire. Mais ici nous récupérons le flot accumulé dans l'étape de filtrage précédente. En effet, un filtrage sur des fenêtres de $11 \times 11$ pixels est

effectué, il suffit de retarder ces signaux pour les récupérer à l'entrée du bloc descripteur. On évite ainsi de gaspiller des ressources mémoires essentielles.

Le taux d'occupation de cette étape en est donc grandement réduit et est détaillé dans le tableau 3.10 :

| **Type de Ressource** | **Occupation** | |
|---|---|---|
| Total Logic Elements | 3,385/39,600 | (9%) |
| Total Memory Bits | 65,536/1,161,216 | (6%) |
| Embedded Multipliers | 0/252 | (0%) |

TABLE 3.10: Le taux d'occupation du FPGA par l'implémentation du bloc *Descripteur*

### 3.5.6 *Le bloc Histogrammes*

Avec un seuil de détection fixe, le nombre de détections varie à chaque nouvelle image. Dans certains cas, il est possible de voir le nombre de détections augmenter d'un facteur 10 et plus. L'étape de tri est nécessaire pour avoir un nombre fixe mais représente aussi une tache trop difficile à implémenter dans un FPGA. En effet, le tri nécessite un parcourt de liste qui est trop long à faire avec plus de 1000 points d'intérêt. Nous avons proposé des approches intermédiaires de tri dans d'autres travaux [18]. Dans cette thèse, nous proposons une nouvelle solution à mi-chemin vers un tri complet et qui est adaptée à l'architecture utilisée.
Nous proposons de changer le seuil dynamiquement (une fois à chaque nouvelle image) afin de stabiliser le nombre de détections. Pour chaque image, nous construisons l'histogramme des détections comme présenté en Figure 3.28.

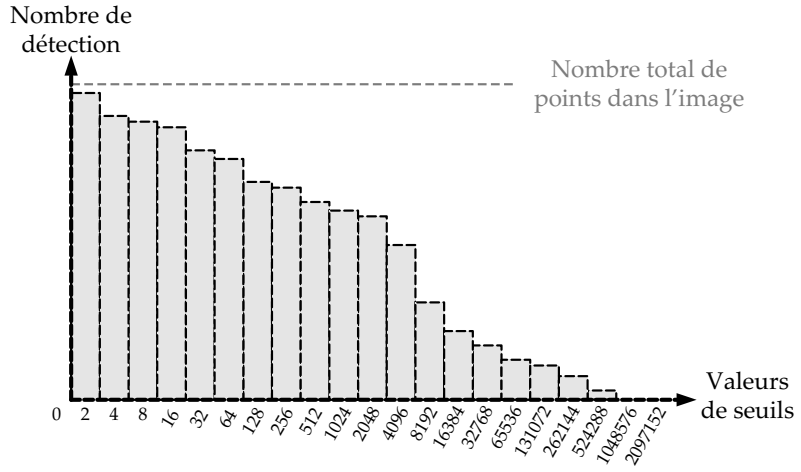


FIGURE 3.28: Exemple d'histogrammes. Lorsque le seuil est à zéro, tous les pixels sont détectés en tant que points d'intérêts, mais à mesure que le seuil augmente, le nombre de détection diminue pour atteindre zéro quand le seuil est trop haut.

L'échelle des abscisses est en puissance de deux et comporte les différentes valeurs de seuils de détection possibles. L'axe des ordonnées représente le nombre de détections correspondant à chaque seuil de l'axe des abscisses. Cet histogramme est complété au fur et à mesure que l'image est capturée et que les scores sont calculés. A la fin d'une image, un signal lance l'accumulation de l'histogramme. Nous parcourons l'histogramme depuis la gauche vers la droite et nous arrêtons quand le nombre des détections dépasse le nombre maximum choisi. Le seuil à appliquer sera celui qui précède, et l'histogramme est ensuite réinitialisé. Cela est fait en temps réel à chaque image selon l'architecture implémentée sur la figure 3.29

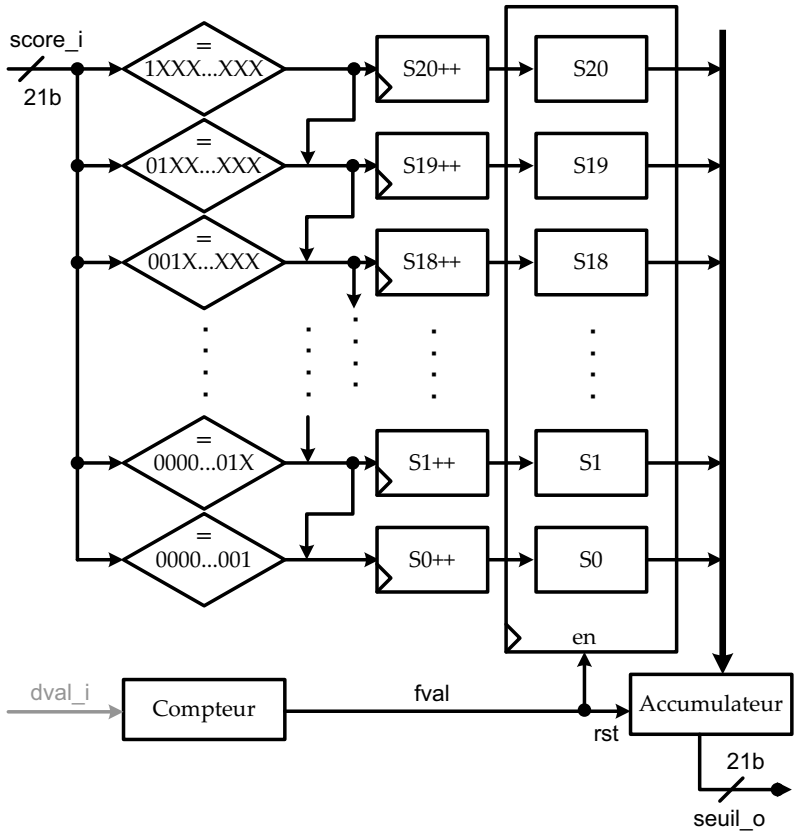


FIGURE 3.29: Architecture détaillée du bloc *Histogrammes* de notre implémentation de l'algorithme de Harris

L'algorithme de recherche qui parcourt l'histogramme prends 21 cycles d'horloge au maximum vu qu'il n'y a que 21 barres dans la figure. L'équilibrage présente plusieurs limites. Au démarrage, un seuil par défaut est appliqué, il faut ensuite une première image entière pour calculer le premier seuil et ainsi stabiliser le nombre de points. Le système est donc opérationnel dès la deuxième image. Le nombre de point peut ensuite légèrement osciller d'une image à l'autre entre les deux nombres de détection correspondant au seuil appliqué et le précédent.
L'implémentation décrite occupe les ressources du FPGA qui sont décrites dans le tableau 3.11.

La latence est variable et est de 21 cycles d'horloge maximum à partir du moment ou une image complète a été capturée.

| Type de Ressource | Occupation | |
|---|---|---|
| Total Logic Elements | 732/39,600 | (18%) |
| Total Memory Bits | 0/1,161,216 | (0%) |
| Embedded Multipliers | 0/252 | (0%) |

TABLE 3.11: Le taux d'occupation du FPGA par l'implémentation du bloc *Histogrammes*

## 3.6 RÉSULTATS ET CONCLUSION

L'intégralité de l'implémentation décrite occupe près de 44% des éléments logiques du FPGA et 63% de ses ressources mémoire. Ces valeurs sont exposées dans le tableau 3.12, sachant que ces nombres incluent le "framework" de base qui avait été décrit dans la section *Hardware*.

| Type de Ressource | Occupation | |
|---|---|---|
| Total des éléments logiques | 17,486/39,600 | (44%) |
| Total des pinnes | 117 on 536 | (22%) |
| Total de la mémoire | 731,192/1,161,216 | (63%) |
| Multiplicateur embarqués | 33/252 | (13%) |
| Total des Phase Locked Loop (PLL) | 1/4 | (25%) |

TABLE 3.12: Le taux d'occupation du FPGA pour l'implémentation complète en prenant en compte le "framework" de fonctionnement

La figure 3.30 (a) montre un panorama brut capturé en plein jour en milieu urbain.

La figure 3.30 (b) présente la détection de points d'intérêts correspondants à cette position de la caméra (les deux captures ont été effectuées à la suite et la détection a été faite par l'architecture implémentée sur le flot).

Cette détection est la première effectuée au démarrage du système, c'est à dire avec le seuil par défaut qui était trop faible. On constate la détection d'un très grand nombre de points. Dans cet exemple, le seuil maximum de détection était fixé à 2000. La figure 3.30 (c) présente la deuxième capture une fois le premier histogramme calculé, on constate que le nombre de points a baissé et est inférieur à 2000 comme attendu.

Ces images sont des représentations des points détectés dans un cadre ayant le même système de coordonnées que les images originelles. Cela dit, seuls les descripteurs comportant les fenêtres de $11 \times 11$ pixels, leurs adresses et leur score sont envoyées. Le reste de l'image est complété avec du noir par le logiciel d'affichage sur le PC.

La figure 3.31 présente un zoom de la figure 3.30 (c) en comparant l'image brute avec l'image de détection.

On peut voir que les coins sont bien détectés et que les parties de l'image ne présentant pas d'intérêt sont bien ignorées.

Grâce au bloc *Histogrammes*, il est possible de décider du nombre de points détectables par panorama. Dans nos applications nous travaillons avec des valeurs différentes, prenons un cas exemple ou nous désirons à peu près 2000 points par image de 360 degrés. Cela fait 2000 paquets de 128 octets par image, et donc 256 kilo octets par image au lieu de 13516800 octets. Le gain en bande passante dans ce cas est de 98%, c'est à dire que l'on utilise à peu près que 2% de la bande passante en flot d'image normal. Avec huit caméras qui fonctionnent simultanément à une cadence qui permet de recevoir 10 couples d'images Harris par secondes on a un débit de presque 8 Méga octets par seconde, ce qui est inférieur à la limite du Slip ring (160 Méga octets par secondes maximum) et de l'USB (45 Méga octets par secondes maximum). Il reste cependant de la place de traitement dans le *Back-End* pour aller plus loin en appariant les points de Harris par exemple, cela aurait pour effet de diminuer un peu plus le débit de données qui va vers l'ordinateur tout en allégeant son travail de traitement.

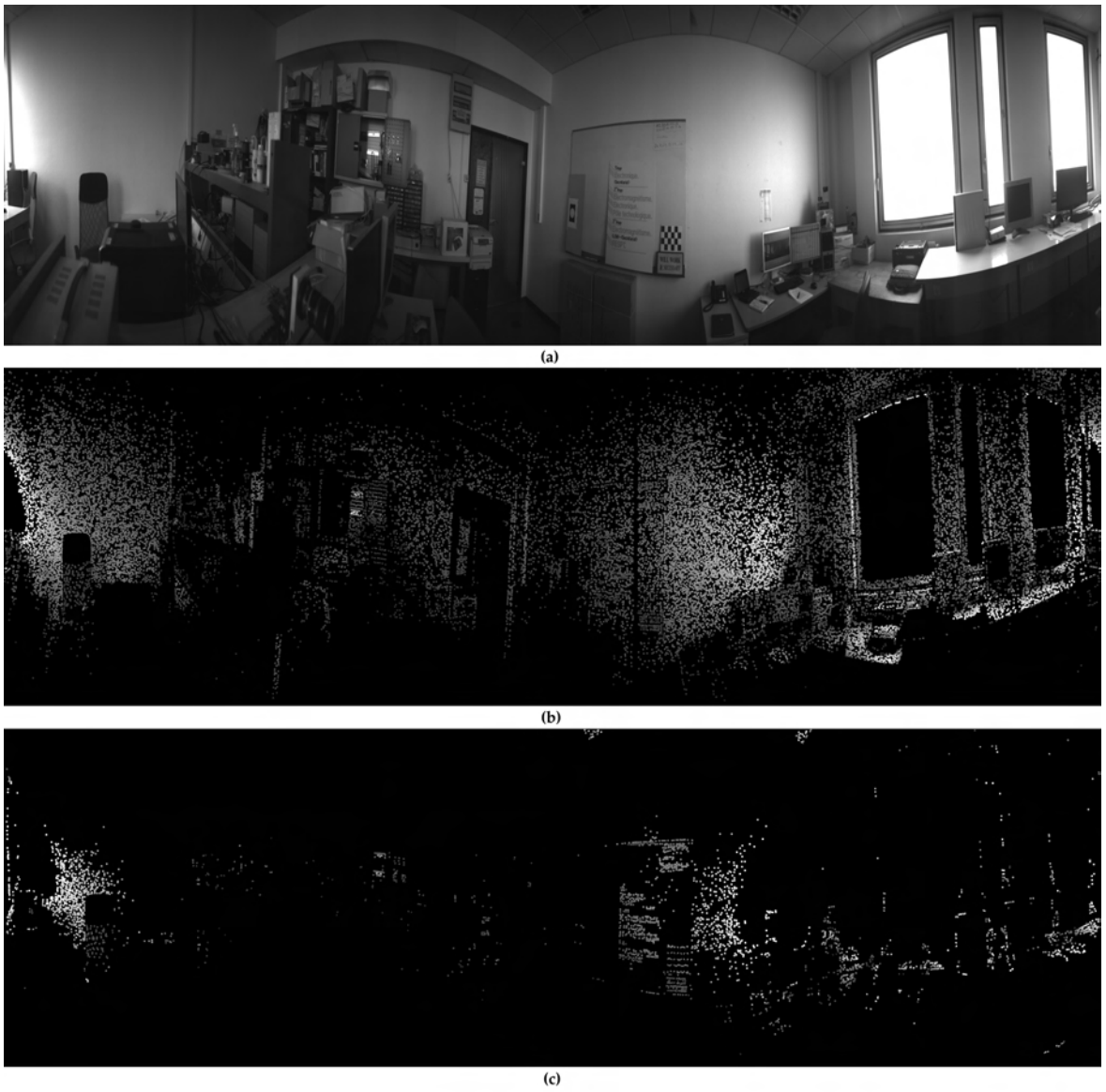


FIGURE 3.30: Résultat de l'implémentation sur un panorama complet. (a) Image brute sans traitement. (b) Image de points d'intérêts détectés au démarrage avant l'équilibrage du nombre de détection. (c) Image de points d'intérêts détectés après stabilisation du seuil à 2000 points maximum.

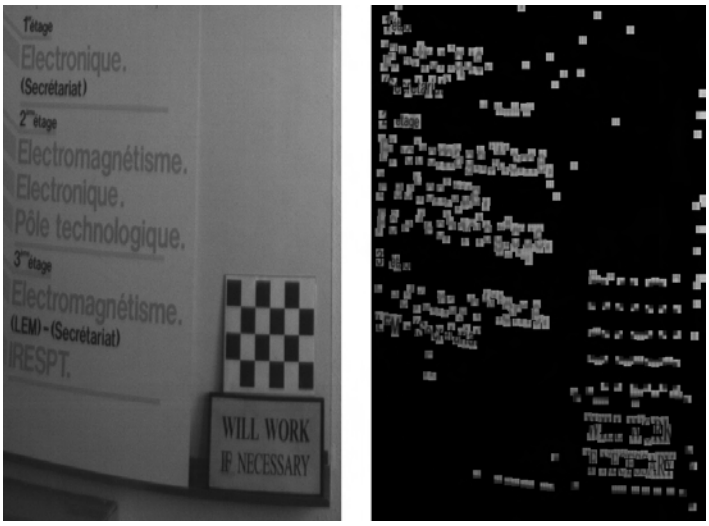


FIGURE 3.31: Comparaison entre l'image avant et après traitement. En haut : une zone de l'image de départ agrandie. En bas : même zone de l'image après traitement

# CHAPITRE 4

# Modèle géométrique des systèmes cylindriques

## 4.1 INTRODUCTION

En traitement d'image il est intéressant de connaître le processus de formation des images au sein des caméras. Ce processus de formation est donc modélisé sous forme de fonction mathématique comportant un certain nombre de paramètres inconnus qu'on identifie par la suite avec une étape dite de calibrage ou d'étalonnage. En d'autres termes, la modélisation d'une caméra donne naissance à un modèle mathématique qui permet de trouver les coordonnées image de la projection d'un point réel à partir de ses coordonnées 3D. On obtient donc la relation mathématique entre le monde réel et le monde image. Beaucoup de modèles existent déjà pour décrire les caméras conventionnelles. Le plus utilisé est le modèle sténopé. Cependant, pour les caméras rotatives, le modèle sténopé ne s'applique pas, ou plutôt seulement de manière partielle comme évoqué dans le chapitre 2.
Dans ce chapitre nous abordons la question de la modélisation des caméras rotatives à capteur linéaire.

Dans un premier temps, nous étudions quatre modélisations existantes qui ont été développées pour ce type de caméras. Après avoir comparé et analysé leurs avantages puis leurs inconvénients, nous proposons une nouvelle modélisation générale. Nous commençons avec un modèle de système comportant une seule caméra linéaire rotative. A partir de cette modélisation nous abordons le sujet de la géométrie épipolaire classique pour ensuite l'appliquer à ce type de systèmes, ce qui aboutit à une modélisation à deux caméras linéaires rotatives pour faire de la stéréovision. Enfin nous finissons par présenter une modélisation comportant jusqu'à huit caméras linéaires.

## 4.2 ÉTAT DE L'ART DES MODÉLISATIONS

### 4.2.1 *Introduction*

Dans cette section, nous allons présenter quatre séries de travaux effectués sur la problématique de la modélisation des systèmes linéaires rotatifs. Afin de comparer ces travaux, ils seront présentés de la même manière et avec une notation homogénéisée et harmonisée.

### 4.2.2 *Modélisation de Schneider et al.*

La première modélisation que nous présentons est celle de Schneider et al [117, 119] publiée pour la première fois en 2003. Ces travaux décrivent une modélisation d'un cas particulier de géométrie cylindrique qui est idéale. Ensuite, le modèle est enrichi de paramètres supplémentaires pour corriger les écarts de calibrage par rapport à l'idéal. La figure 4.1 présente un aperçu du modèle :

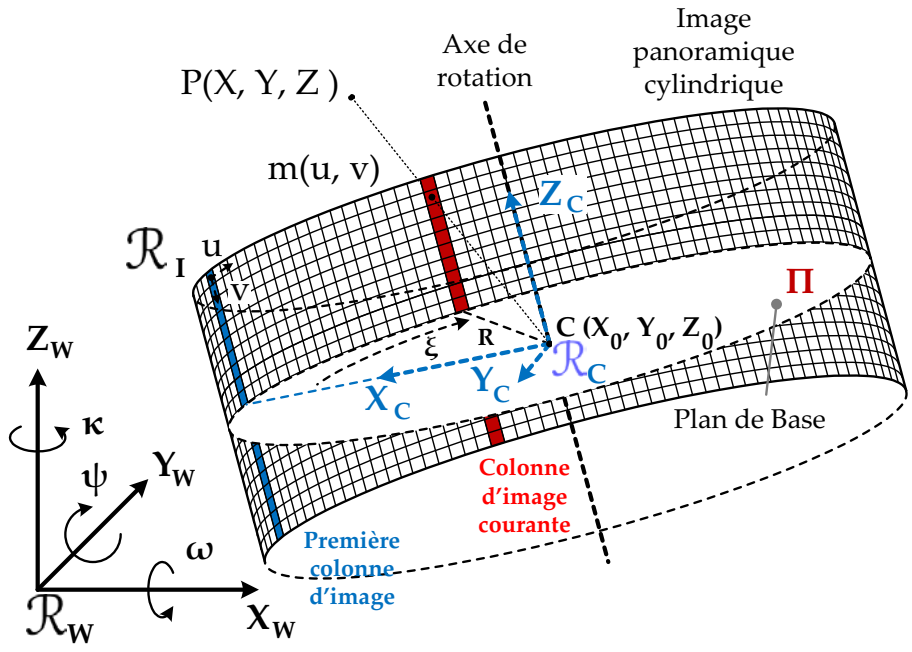


FIGURE 4.1: Modèle géométrique proposé dans les travaux de Danilo Schneider et al. en 2003 *Geometric modelling and calibration of a high resolution panoramic*

Pour modéliser les caméras cylindriques, une approche à repères multiples est généralement utilisée. Le passage d'un repère à l'autre modélise une partie du système et des transformations successives qui mènent au repère image. Sur le schémas on peut distinguer trois repères différents [115] :

- $\mathcal{R}_W$ est le repère Monde,
- $\mathcal{R}_C$ est le repère Caméra. Le centre C de ce repère est à la fois le centre de rotation et le centre de projection de la caméra. L'axe $Z_C$ est l'axe de rotation du système et l'axe

$X_C$ définit la direction de la première colonne de l'image. Le repère $\mathcal{R}_C$ est immobile dans ce modèle.

- $\mathcal{R}_I$ le repère de l'image avec pour colonnes $u$ et pour lignes $v$.

Après développement, les transformations qui permettent de passer d'un repère à l'autre donnent les équations suivantes. Soit un point $P(X,Y,Z)^T$ dans le repère monde $\mathcal{R}_W$. La projection du point $P$ sur l'image cylindrique donne lieu au point $m$ de coordonnées $m(u,v)$ :

$$u = \frac{1}{A_h}.\arctan(\frac{-y}{x}) + \frac{\xi_0}{A_h} + d_u \qquad et \qquad v = \frac{N}{2} + \frac{c.z}{A_v.\sqrt{x^2+y^2}} - \frac{\eta_0}{A_v} + d_v \tag{4.1}$$

Avec :

$$\begin{aligned} x &= r_{11}.(X-X_0) + r_{21}.(Y-Y_0) + r_{31}.(Z-Z_0) \\ y &= r_{12}.(X-X_0) + r_{22}.(Y-Y_0) + r_{32}.(Z-Z_0) \\ z &= r_{13}.(X-X_0) + r_{23}.(Y-Y_0) + r_{33}.(Z-Z_0) \end{aligned} \tag{4.2}$$

Sachant que les paramètres $r_{ij}$ sont les paramètres de la matrice de rotation $R$ qui définit la rotation entre les repères $\mathcal{R}_C$ et $\mathcal{R}_W$. Les paramètres sont au nombre de 12 et sont présentés dans le tableau 4.1.

On a donc un modèle qui confond l'axe vertical du capteur avec l'axe de rotation, ce qui a pour conséquences de négliger à la fois :

- Le décentrage du capteur avec l'axe de rotation, (le centre optique du capteur ne peut pas être situé sur l'axe de rotation),
- La déviation de l'orientation du capteur par rapport à l'axe de rotation (usinage de la rétine et placement imparfait de cette dernière sur le système),
- Les imperfections du mouvement de rotation (orientation de l'axe de rotation).

Deux paramètres supplémentaires $d_u$ et $d_v$ sont à prendre en compte. Ils permettent de corriger la modélisation en ajoutant des paramètres additionnels [118]. Dans la modélisation de base $d_u$ et $d_v$ sont égaux à zéro. Une amélioration du modèle est cependant présentée avec l'ajout de 19 paramètres supplémentaires. Ce qui fait un total de 31 paramètres à calibrer quand le modèle est dans sa forme la plus complète. Cela dit, l'expression de la modélisation complète n'est donnée nulle part dans les publications de ces travaux.
Avec ces paramètres supplémentaires les trois approximations sont partiellement corrigées :

- Les erreurs du mouvement de rotation sont partiellement compensées avec un paramètre de déviation angulaire,
- L'excentricité du capteur avec l'axe de rotation est prise en compte avec 3 paramètres,

| Type de Paramètre | Nom | Fonction |
|---|---|---|
| Intrinsèques | N | Nombre de lignes dans l'image |
| | $\eta_0$ | Composante vertical du point principal (intersection de l'axe $X_C$ et de l'image) |
| | $\xi_0$ | Composante horizontale du point principal |
| | $A_v$ | Résolution Verticale |
| | $A_h$ | Résolution Horizontale (angle de rotation par colonne) |
| | c | Distance principale (distance entre le centre de projection et le point principal) |
| Extrinsèques | $X_0$,$Y_0$ et $Z_0$ | Coordonnées du centre du repère $\mathcal{R}_C$ par rapport au repère $\mathcal{R}_W$. |
| | $\omega$, $\psi$ et $\kappa$ | Angles de la matrice de rotation R qui définit le mouvement du repère $\mathcal{R}_W$ au repère $\mathcal{R}_C$ |

TABLE 4.1: Les 12 paramètres du modèle présenté dans les travaux de Schneider et al.

- L'erreur de déviation de l'orientation du capteur est partiellement compensée avec 2 paramètres.

Ce modèle est donc incomplet et néglige plusieurs paramètres essentiels afin de garder une certaine simplicité. Dans un second temps, une compensation de cette négligence est proposée mais apporte plus de complication que nécessaire et elle ne permet toujours pas la modélisation complète des paramètres essentiels tels que l'orientation du capteur avec trois anges et les paramètres de déviation du mouvement de rotation.

### 4.2.3 *Modélisation de Smadja et al.*

La deuxième modélisation que nous présentons est celle de Smadja et al [128]. Le modèle présenté est un autre modèle à centre de projection unique, il s'agit donc également d'un cas particulier idéal. La figure 4.2 présente un aperçu du modèle :

Sur ce schéma on peut distinguer trois repères différents :

- $\mathcal{R}_W$ est le repère Monde,
- $\mathcal{R}_C$ est le repère Caméra. Le centre de ce repère est à la fois le centre de projection de la caméra et le centre de rotation du système. L'axe $Z_C$ est l'axe de rotation et l'axe

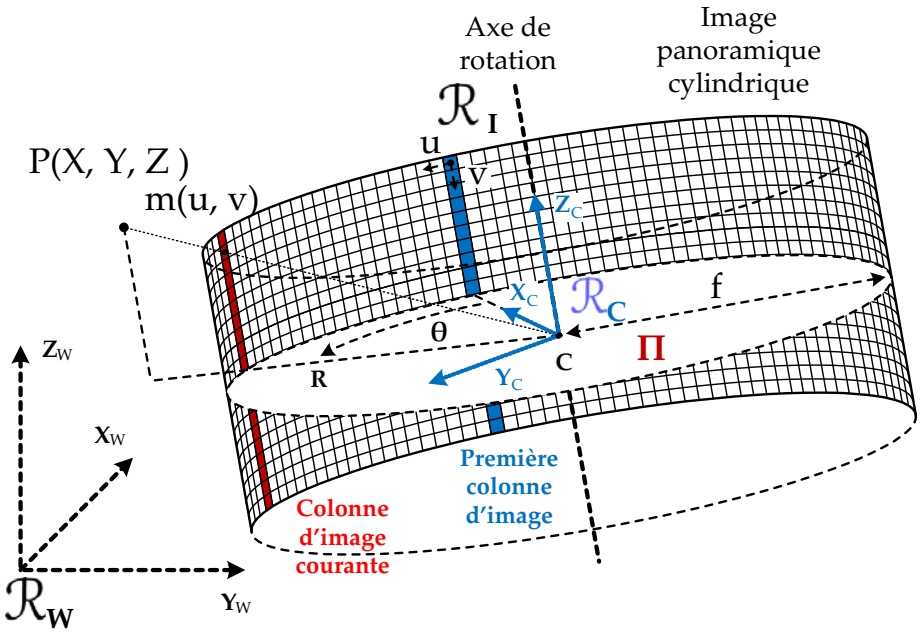


FIGURE 4.2: Modèle géométrique proposé dans les travaux de L. Smadja et al. en 2004 "Cylindrical Sensor Calibration using Lines"

$X_C$ définit la direction de la première colonne de l'image, il est donc confondu avec l'axe optique de la caméra.

- $\mathcal{R}_I$ le repère de l'image avec pour colonnes $u$ et pour lignes $v$.

En considérant le repère $\mathcal{R}_C$ centré en C, les coordonnées de la projection $p(x,y,z)^T$ du point $P(X,Y,Z)^T$ sur l'image cylindrique sont données par :

$$x = f.\frac{X_C}{Z_C} \qquad y = f.\frac{Y_C}{Z_C} \qquad z = f.\frac{Z_C}{R} \tag{4.3}$$

Avec $R = \sqrt{X_C^2 + Z_C^2}$.
Les coordonnées image du point $m$ sont ensuite exprimées via les coordonnées cylindriques $(R, \theta, Z_C)$ par la transformation :

$$\begin{cases} u = K_u.\theta = K_u.\arctan(\frac{Y_C}{X_C}) \\ v = v_0 - K_v.f.z = v_0 - K_v.f.\frac{Z_c}{R} \end{cases} \tag{4.4}$$

Ou $K_u$ et $K_v$ sont les pas de discrétisation de l'image et $v_0$ la coordonnée verticale de la projection du centre C du système sur la colonne de pixels. Les six paramètres $(t_x, t_y, t_z, \epsilon, \theta, \phi)$ de la transformation $(R,t)$ qui mène le repère $\mathcal{R}_W$ au repère $\mathcal{R}_C$ n'ont pas été injectés dans l'équation afin de préserver sa forme linéaire.
Les dix paramètres du modèle sont résumés dans le tableau 4.2.

Nous avons ici un modèle particulier qui décrit les systèmes dont le centre de rotation est confondu avec tous les centres de projection de la caméra lors de son mouvement. Il s'agit donc d'un système à centre de projection unique.

| Type de Paramètre | Nom | Fonction |
|---|---|---|
| Intrinsèques | $K_u$ | correspond à l'inverse de la taille verticale d'un pixel. |
| | $K_v$ | Le pas angulaire (équivaut à $2.\pi$ divisé par le nombre de colonnes acquises pendant une rotation complète. |
| | $f$ | La distance focale de la caméra |
| | $v_0$ | La coordonnée de la projection du centre de la caméra sur la colonne de pixels |
| Extrinsèques | $t_x$,$t_y$ et $t_z$ | Coordonnées du centre du repère $\mathcal{R}_C$ par rapport au repère $\mathcal{R}_W$. |
| | $\epsilon$, $\theta$ et $\phi$ | Angles de la matrice de rotation R qui définit le mouvement entre les repères $\mathcal{R}_W$ et $\mathcal{R}_C$ |

TABLE 4.2: Les 10 paramètres du modèle présenté dans les travaux de Smadja et al.

Les conséquences sont les mêmes que pour la version de base du modèle précédent, à savoir la négligence des réalités suivantes :

- Les imperfections du mouvement de rotation,
- Le décentrage du capteur avec l'axe de rotation,
- La déviation de l'orientation du capteur par rapport à l'axe de rotation.

Dans ces travaux, il n'y a pas de paramètres additionnels pour corriger ces approximations. La modèle est donc à la fois un cas particulier de géométrie et un cas idéal.

### 4.2.4 *Modélisation de Parian et al.*

La troisième modélisation est publiée par les travaux de Parian et al [88, 89]. Le modèle présenté est un autre modèle à centre de projection unique. Il s'agit d'un modèle idéal qui est ensuite enrichi par l'ajout de paramètres supplémentaires.

Sur le schéma on peut distinguer quatre repères différents :

- $\mathcal{R}_W$ est le repère Monde.
- $\mathcal{R}_B$ est le repère Auxiliaire ou de la Base. Le centre de ce repère est à la fois le centre de projection de la caméra et le centre de rotation du système. L'axe $Z_B$ est l'axe de

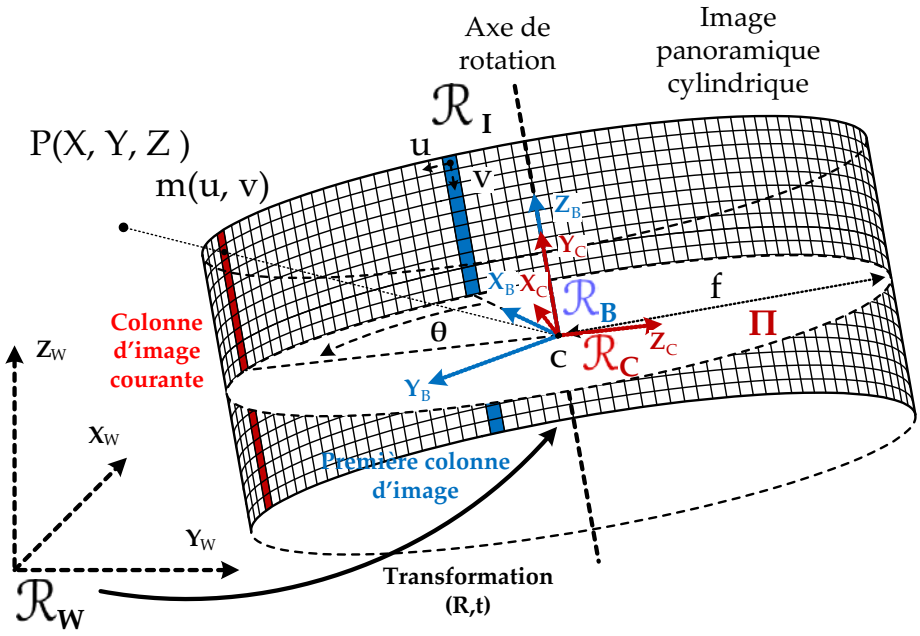


FIGURE 4.3: Modèle géométrique proposé dans les travaux de Parian et al. en 2004 "A Refined Sensor Model For Panoramic Cameras".

rotation et l'axe $X_B$ définit la direction de la première colonne de l'image, il est donc l'axe optique de la caméra.

- $\mathcal{R}_C$ est le repère Caméra.
- $\mathcal{R}_I$ le repère de l'image avec pour colonnes $u$ et pour lignes $v$.

Le repère de la base modélise la rotation du capteur et les erreurs mécaniques La modélisation idéale donne lieu à l'équation suivante :

$$\begin{bmatrix} 0 \\ y \\ -c \end{bmatrix} = \lambda . P^{-1} . R_z^t . (j . A_h) . M_{w,\phi,k} . \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix} \tag{4.5}$$

Avec :

$$P = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad et \qquad y = (i - \frac{N}{2}) . A_v \tag{4.6}$$

Sachant que $(0, y, -c)^T$ sont les coordonnées d'un point image dans le repère de la caméra $\mathcal{R}_C$, que P est la matrice de transfert du repère $\mathcal{R}_C$ au repère $\mathcal{R}_B$ et que $R_z$ est la matrice de rotation autour de l'axe $Z_B$ qui mène à la position du repère caméra courant. Enfin, la variable $\lambda$ est le facteur d'échelle. Les autres variables, qui sont au nombre de 10, sont les paramètres du modèle et sont présentées dans le tableau 4.3.

Le modèle néglige les paramètres suivants :

| Type de Paramètre | Nom | Fonction |
|---|---|---|
| Intrinsèques | $A_h$ | La résolution de la rotation |
| | $A_v$ | La taille verticale d'un pixel du capteur |
| | c | La constante de la caméra |
| | N | Nombre de pixel sur le capteur linéaire |
| Extrinsèques | X0,Y0 et Z0 | Coordonnées du centre du repère $\mathcal{R}_B$ par rapport au repère $\mathcal{R}_W$. |
| | $M_{w,\phi,k}$ | La matrice de rotation 3D qui décris le mouvement entre les repères $\mathcal{R}_W$ et $\mathcal{R}_B$ |

TABLE 4.3: Les 10 paramètres du modèle présenté dans les travaux de Parian et al.

- Les imperfections du mouvement de rotation,
- Le décentrage du capteur avec l'axe de rotation,
- La déviation de l'orientation du capteur par rapport à l'axe de rotation.

Ce modèle idéal est ensuite amélioré avec l'ajout de nouveaux paramètres qui visent à prendre en compte les réalités listées ci-dessous :

- Les paramètres de "thumbling" décrivant les erreurs mécaniques (avec 3 à 6 paramètres),
- Les distorsions de la lentille,
- Le décalage du point principal,
- La constante de la caméra,
- La résolution de la rotation,
- Erreurs d'orientation du capteur linéaire (avec 2 paramètres),
- L'excentricité du centre de projection par rapport au centre du repère de la base $\mathcal{R}_B$ (avec 2 paramètres).

Ces paramètres additionnels sont aux nombre de 12, et permettent la prise en compte de toutes les erreurs vues précédemment au coût d'une complication du modèle. Cela dit, l'approche de compensation des erreurs post-calibrage est une complication inutile. Certains paramètres comme les angles d'orientation du capteur ou l'excentricité ne sont pas des paramètres à négliger comme le montre les résultats de ces travaux. En effet, dans

l'expérience de calibrage proposée, le fait de négliger excentricité du centre de projection provoque un décalage de deux pixels dans l'image par exemple. Il est donc clair que ces paramètres doivent être inclus dès le début dans le modèle de base.

### 4.2.5 *Modélisation de Huang et al.*

Enfin, le dernier modèle étudié est celui de Huang et al [55] qui a été publié en 2003 pour la première fois. Ensuite, d'autre travaux abordent aussi le problème de la stéréovision [57, 53]. Puis, le modèle a été sujet à plusieurs modifications et la version finale a été présentée en 2010 [111]. Ce modèle est le seul à centres de projections multiples comme présenté sue la figure 4.4.

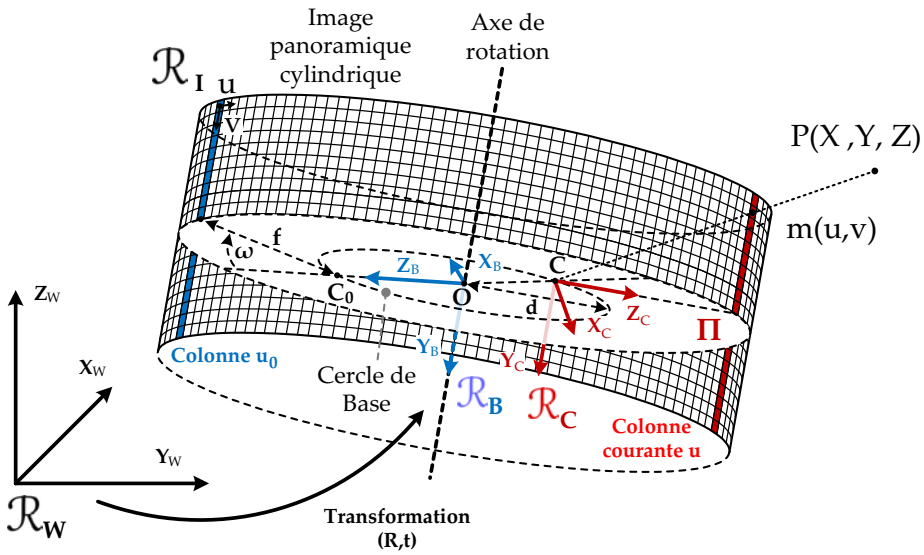


FIGURE 4.4: Modèle géométrique proposé dans les travaux de Fay Huang et ali. en 2010 : "Pose Estimation of Rotating Sensors in the Context of Accurate 3D Scene Modeling"

Sur le schémas 4.4 on peut distinguer quatre repères différents [56] :

- $\mathcal{R}_W$ est le repère monde dans lequel les points 3D sont exprimés.
- $\mathcal{R}_B$ est le repère de la base du système. Le centre du repère est assimilé au centre de rotation et l'axe $Y_B$ est l'axe de rotation. L'intersection de l'axe $Z_B$ avec le cercle de base définit le point $C_0$ qui est le centre de projection lorsque la caméra est en position de capturer la première colonne de l'image.
- $\mathcal{R}_C$ est le repère Caméra dans sa position courante. Son centre C est le centre de projection de la caméra et l'axe $Z_C$ définit la direction de la colonne de l'image courante.
- $\mathcal{R}_I$ le repère de l'image avec pour colonnes u et pour lignes v.

La matrice de transformation $T(R,t)$ décrit le mouvement entre le repère monde $\mathcal{R}_W$ et

le repère de la base$\mathcal{R}_B$ avec la translation $t$ ayant pour paramètres $(x_0, y_0, z_0)$. La rotation $R_{\phi(u)}$ décrit ;la position de la plateforme lorsque la caméra capture la colonne d'index $u$. Pendant la rotation, le centre de projection de la caméra décrit un cercle de rayon $d$ autour de l'axe de rotation. Ce cercle est appelé le cercle de base et le plan qui le contient est appelé le plan de base $\mathcal{P}$ du système. On a aussi l'expression de la coordonnée verticale de la projection d'un point :

$$v(u) = \begin{pmatrix} 0 \\ u.\tau - v_0 \\ f \end{pmatrix} \tag{4.7}$$

Où $\tau$ est la taille verticale d'un pixel et $v_0$ la coordonnée verticale du point principal. Ainsi on peut noter les coordonnées d'un point $P_W$ dans le repère monde $\mathcal{R}_W$ selon l'équation suivante :

$$P_w = t + R.R_{\phi(u)} \left[ \lambda.R_\omega . \begin{pmatrix} 0 \\ u.\tau - v_0 \\ f \end{pmatrix} + R. \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right] \tag{4.8}$$

L'orientation du capteur est définie selon deux rotations $R_\omega$ et $R_\xi$ qui définissent respectivement le lacet $\omega$ et le roulis $\xi$. Enfin, la matrice de rotation $R_u(\alpha, \beta, \delta)$ définit le décalage d'orientation du capteur par rapport à l'axe optique, et le vecteur $(\Delta_x, \Delta_y, \Delta_z)$ définit son décalage de position.
Le modèle complet est exprimé selon l'équation :

$$P_w = t + R.R_{\phi(u)} \left[ \lambda.R_\xi.R_\omega . \left[ R_u . \begin{pmatrix} \Delta_x - u_0 \\ u.\tau + \Delta_y - v_0 \\ \Delta_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ f \end{pmatrix} \right] + \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} \right] \tag{4.9}$$

Les paramètres sont au nombre de 19 et sont présentés dans le tableau 4.4.

Dans cette version du modèle nous avons une modélisation précise avec 19 paramètres contre 4 seulement dans la version précédemment présentée. Certains paramètres présentent une redondance, comme par exemple les cinq angles qui définissent l'orientation du capteur. Une simplification aurait été possible.

### 4.2.6 *Conclusion*

Nous allons à présent comparer les quatre modèles étudiés selon les critères suivants :

- Le nombre de paramètres intrinsèques et extrinsèques pour rendre compte de la complexité des modèles et du nombre d'inconnues à calculer,
- La modélisation de l'axe de rotation, si elle est complète ou non,
- La prise en compte de l'excentricité du centre de projection par rapport à l'axe de rotation,

| Type de Paramètre | Nom | Fonction |
|---|---|---|
| Intrinsèques | d | Le rayon du cercle principal, c'est à dire la distance entre le centre de rotation O et le centre de projection C de la caméra. |
| | f | La distance focale de la caméra |
| | $\omega$ | L'angle entre l'axe optique de la caméra et la droite (OC) |
| | $\xi$ | L'angle entre l'axe optique de la caméra et le plan principal |
| | $W$ | Le nombre de colonnes dans l'image |
| | $(\Delta_x, \Delta_y, \Delta_z)$ | le décalage du centre de projection du capteur |
| | $R_u(\alpha, \beta, \delta)$ | le décalage de l'orientation du capteur |
| | $(u_0, v_0)$ | les coordonnées du point principal dans l'image |
| Extrinsèques | t | le vecteur de translation 3D qui décrit le mouvement entre les repères $\mathcal{R}_W$ et $\mathcal{R}_B$. |
| | R | La matrice de rotation 3D qui décrit le mouvement entre les repères $\mathcal{R}_W$ et $\mathcal{R}_B$ |

TABLE 4.4: Les 19 paramètres du modèle présenté dans les travaux de Fay Huang et al.

- La prise en compte des trois angles décrivant la déviation de l'orientation du capteur.

La comparaison des quatre modèles est présentée dans le tableau 4.5.

Dans les quatre travaux étudiés, trois d'entre eux présentent des modèles à centre de projection unique. Avoir un centre de projection unique c'est dire que la caméra est placée d'une façon parfaite sur la plateforme tournant de manière à ce que son centre optique ne bouge pas, c'est à dire qu'il est supposé se trouver sur l'axe de rotation. Cela est difficilement réalisable en pratique et aboutira forcément à des erreurs de calibration plus ou moins importantes. C'est pour cela que deux de ces trois travaux proposent ensuite un affinage de ces écarts de calibrage avec l'ajout de paramètres supplémentaires. Donc par une représentation géométrique de ces écarts moins une approximation. Un autre problème important que présentent trois des quatre travaux étudiés est la non modélisation de l'orientation du capteur par rapport à la plateforme. Elle est toujours considérée comme idéale avec des valeurs d'angle approximées à 0. Deux des travaux proposent des corrections à ce sujet mais aucun d'entre eux ne modélise l'orientation complètement avec trois angles(deux tout au plus), seul la dernière modélisation de Huang et ali. le fait réellement. Enfin les erreurs du mouvement mécanique ne sont pas non plus toujours modélisées.

| Modèles | Nb. de paramètres | | Axe de rotation | centres de projection | Inclinaison capteur |
|---|---|---|---|---|---|
| | Intrins. | Extrins. | | | |
| Schneider et al. | 6 | 6 | idéal | unique | idéale |
| Schneider et al.(complète) | 25 | 6 | 1 paramètre | multiples (2 param) | 2 paramètres |
| Smadja et al. | 6 | 4 | idéal | unique | idéale |
| Parian et al. | 4 | 6 | idéal | unique | idéale |
| Parian et al.(complète) | 16 | 6 | 6 paramètres | multiples | complète |
| Huang et al. | 13 | 6 | complète | multiples | 5 paramètres |

TABLE 4.5: Bilan comparatif des modèles étudiés

Toutes ces erreurs et ces approximations rendent les modèles idéaux et pas réalisables. Dans les versions enrichies de paramètres de correction, on constate une grande complexification des systèmes qui vont jusqu'à comporter une trentaine de paramètres. Dans les deux travaux qui adoptent cette démarche, l'étape de calibration est effectuée dans des pièces spécialement préparées avec des centaines de mires de calibrage qui doivent être placées tout autour du capteur avec des précisions de l'ordre du dixième de millimètre. Cela représente une grosse contrainte pour l'utilisateur, qui doit donc posséder de gros moyens et des connaissances en la matière pour pouvoir calibrer le système.

En conclusion, la faiblesse des trois premières approchse est de ne pas complètement prendre en compte les réalités suivantes :

- Modélisation du mouvement de rotation réel (c'est à dire en prenant en compte les erreurs mécaniques qui font que l'axe de rotation n'est pas celui prévu théoriquement),
- Modélisation de l'orientation du capteur avec trois angles,
- Décentrage du centre de projection avec l'axe de rotation.

Concernant les travaux qui proposent les modélisations les plus complètes, deux problèmes surviennent :

- Une multiplication du nombre de paramètres et une complexification des équations qui aboutit à des problèmes non linéaires comportant des singularités dans certains cas. Ces singularités sont des cas de géométries ou le système ne peut pas converger et le calibrage n'est donc pas possible.
- Complexification des méthodes de calibrage qui nécessitent des moyens de plus en plus importants et contraignants.

## 4.3 MODÉLISATION GÉNÉRALE À UNE CAMÉRA

Comme cela a été expliqué dans la section précédente, trois réalités physiques sont à prendre en compte :

- Axe de rotation libre par rapport au repère monde. L'axe de rotation n'a pas une position idéale par rapport au repère monde, et possède 6 degrés de liberté (trois translations et trois angles d'orientation).
- Centre optique de la caméra libre par rapport à l'axe de rotation, c'est à dire la possibilité d'avoir un système à plusieurs centres de projection.
- Orientation de l'axe optique de la caméra libre par rapport à l'axe de rotation du mécanisme. C'est à dire qu'il faut modéliser les trois degrés de liberté qui définissent les trois angles d'orientation du capteur.

Notre approche dans cette thèse est de modéliser au mieux tout ces paramètres dans un modèle qui se veut le plus général possible. De la sorte, le modèle doit être adaptable à tous les systèmes linéaires rotatifs. De plus nous allons minimiser le nombre de paramètres en incluant les erreurs mécaniques et autres directement dans les paramètres de base du système. C'est à dire qu'au lieu de modéliser un système idéal et de l'enrichir ensuite de paramètres supplémentaires, nous allons directement le modéliser avec toutes ses imperfections. Cela permettra d'obtenir un meilleur rapport fidélité/simplicité et nous présenterons une méthode simple de calibrage associée dans le chapitre 5.

### 4.3.1 *Modèle mono-caméra*

Soit $\mathcal{S}$ un système rotatif quelconque. Le système $\mathcal{S}$ comporte une seule caméra linéaire rotative. Ce système quelconque est décrit par la figure 4.5 :

Soit $\mathcal{D}$, l'axe de rotation réel du système $\mathcal{S}$, qui est différent de l'axe de rotation idéal définit pas la conception mécanique. Cette différence est due aux erreurs et aux imprécisions mécaniques du système rotatif. L'axe $\mathcal{D}$ n'est donc pas forcément perpendiculaire au plan formé par l'horizon du repère monde. Soit C le centre de projection de la caméra, et $C_0$ le centre de projection de la caméra lorsque le système rotatif est dans sa position initiale. C'est à dire lorsque la caméra est positionnée telle qu'elle capture la première colonne de l'image panoramique. Le point C ne se situe pas forcément sur l'axe de rotation. Soit la droite $\mathcal{E}$ qui passe par le point C et qui coupe l'axe de rotation perpendiculairement au point O Le plan $\Pi$ est le plan normal à l'axe de rotation $\mathcal{D}$ et passant par le point C. Le plan $\Pi$ contient toutes les positions successives du centre de projection de la caméra lors de sa rotation autour de l'axe $\mathcal{D}$, ainsi que la droite $\mathcal{E}$. Ainsi, le centre de projection de la caméra décrit un cercle de rayon d inscrit dans le plan $\mathcal{P}$. Le point formé par l'intersection du plan $\Pi$ et de l'axe de rotation est le centre de rotation O du système. Ce point va servir de point de repère en tant que centre de rotation de positon $(0 \quad 0 \quad 0)$.

Pour décrire le mouvement et la projection de ce système, plusieurs repères orthonormés sont nécessaires :

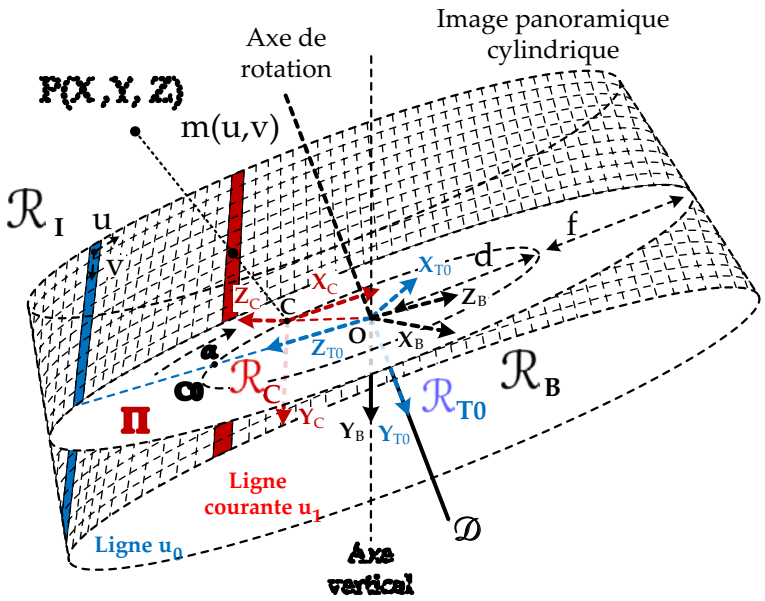


FIGURE 4.5: Modèle général d'un système à caméra linéaire rotatif. On constatera que l'image n'est pas tout à fait un cylindre, cela est due à la modélisation des trois angles $(\phi, \theta, \psi)$ de l'erreur d'orientation du capteur. Cela pose le problème de la résolution non-homogène.

- $\mathcal{R}_B$ est le repère de la base du système (qui est notre repère *Monde*),
- $\mathcal{R}_{T0}$ est le repère de la plateforme rotative à la position zéro,
- $\mathcal{R}_T$ est le repère de la plateforme rotative à la position courante,
- $\mathcal{R}_C$ est le repère de la caméra linéaire,
- $\mathcal{R}_I$ est le repère de l'image.

Les trois repères $\mathcal{R}_B$, $\mathcal{R}_{T0}$ et $\mathcal{R}_T$ ont pour centre commun, le point O qui est le centre de rotation. Le plan $X_B O Z_B$ formé par les axes $X_B$ et $Z_B$ du repère base $\mathcal{R}_B$ est le plan horizontal. Le passage du repère $\mathcal{R}_B$ au repère $\mathcal{R}_{T0}$ décrit l'inclinaison de l'axe de rotation et du plan de la plateforme tournante. Le repère $\mathcal{R}_{T0}$ définit également l'orientation initiale de la plateforme(à l'exposition de la première colonne image). L'axe $Y_{T0}$ du repère $\mathcal{R}_{T0}$ est confondu avec l'axe de rotation du système, il en est de même pour l'axe $Y_T$ du repère $\mathcal{R}_T$. Le passage du repère $\mathcal{R}_{T0}$ au repère $\mathcal{R}_T$ est une rotation autour de l'axe de rotation du système et d'angle $\alpha$ dans le sens direct, comme illustré sur la figure 4.5. Le repère $\mathcal{R}_T$ décrit donc la position courante du plateau rotatif selon l'angle $\alpha$. Enfin, le repère $\mathcal{R}_C$ est celui de la caméra et la projection qui crée l'image panoramique se fait selon lui. Il est issu d'un mouvement de translation à partir du repère $\mathcal{R}_T$ selon l'axe $Z_T$ et correspond à la modélisation du décentrage. L'orientation du repère $\mathcal{R}_C$ par rapport au repère $\mathcal{R}_T$ est définie selon trois angles afin de représenter tous les cas d'orientation possibles de la rétine et de l'optique.

Les quatre transformations qui décrivent les mouvements d'un repère à l'autre sont les suivantes :

- $R^c$ est la rotation 2D qui décrit le mouvement du repère $\mathcal{R}_B$ vers le repère $\mathcal{R}_{T0}$, c'est à dire l'inclinaison du plan de la plateforme tournante. $R^c$ est composée de deux rotations intrinsèques élémentaires :
  - $\omega$ autour de l'axe $X_B$ dans le sens direct (de $Y_B$ vers $Z_B$),
  - $\beta$ autour de l'axe $Y'_B$ dans le sens direct (de $Z'_B$ vers $X'_B$).

  On a donc $R^c = R_y(\omega).R_x(\beta)$.
- $R^b$ est la rotation 1D qui décrit le mouvement du repère $\mathcal{R}_{T0}$ vers le repère $\mathcal{R}_T$. $R^b$ est composée d'une seule rotation élémentaire :
  - $\alpha$ autour de l'axe $Y_{T0}$ dans le sens direct (de $Z_{T0}$ vers $X_{T0}$).

  On a donc $R^b = R_y(\alpha)$.
- $R^a$ est la rotation 3D qui décrit le mouvement du repère $\mathcal{R}_T$ vers le repère $\mathcal{R}_C$. $R^a$ est composée de trois rotations intrinsèques élémentaires :
  - $\psi$ autour de l'axe $X_T$ dans le sens direct (de $Y_T$ vers $Z_T$),
  - $\theta$ autour de l'axe $Y'_T$ dans le sens direct (de $Z'_T$ vers $X'_T$),
  - $\phi$ autour de l'axe $Z''_T$ dans le sens direct (de $X''_T$ vers $Y''_T$).

  On a donc $R^a = R_z(\phi).R_y(\theta).R_x(\psi)$.
- $t^a$ est la translation qui définit le décentrage entre le centre de rotation et le centre de projection de la caméra. Il représente le mouvement de translation qui a lieu du repère $\mathcal{R}_T$ vers le repère $\mathcal{R}_C$. $t^a$ est une translation élémentaire :
  - de valeur $+d$ selon l'axe $Z_T$.

Soit $T^a$ la matrice de transformation homogène qui regroupe la rotation $R^a$ et la translation $t^a$, et les matrices de transformations $T^b$ et $T^c$ qui sont les versions homogènes respectives des rotations $R^b$ et $R^c$.

Nous avons donc :

$$T^a = \begin{bmatrix} \cos(\theta).\cos(\phi) & \sin(\psi).\sin(\theta).\cos(\phi) - \cos(\psi).\sin(\phi) & \cos(\psi).\sin(\theta).\cos(\phi) + \sin(\psi).\sin(\phi) & 0 \\ \cos(\theta).\sin(\phi) & \sin(\psi).\sin(\theta).\sin(\phi) - \cos(\psi).\cos(\phi) & \cos(\psi).\sin(\theta).\sin(\phi) + \sin(\psi).\cos(\phi) & 0 \\ -\sin(\theta) & \sin(\psi).\cos(\theta) & \cos(\psi).\cos(\theta) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

$$T^b = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

$$T^c = \begin{bmatrix} \cos(\beta) & \sin(\beta).\sin(\omega) & \sin(\beta).\cos(\omega) & 0 \\ 0 & \cos(\omega) & -\sin(\omega) & 0 \\ -\sin(\beta) & \cos(\beta).\sin(\omega) & \cos(\beta).\cos(\omega) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

### 4.3.2 *Mise en équation*

Établissement des transformations de passage entre le repère camera et le repère base : Soit le point P définit par les coordonnées suivantes :

- $P_B = [X_B, Y_B, Z_B]^T$ dans le repère Base $\mathcal{R}_B$,
- $P_C = [X_C, Y_C, Z_C]^T$ dans le repère Caméra $\mathcal{R}_C$,
- $m = [u, v]^T$ est la projection du point P sur l'image panoramique (avec ses coordonnées en pixels).

Pour passer des coordonnées $P_B = [X_B, Y_B, Z_B]^T$ aux coordonnées $P_C = [X_C, Y_C, Z_C]^T$ il faut effectuer les trois transformations présentées ci-dessus :

$$\begin{bmatrix} X_B \\ Y_B \\ Z_B \\ 1 \end{bmatrix} = T^c \times T^b \times T^a \times \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \quad \text{et} \quad \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = (T^a)^{-1} \times (T^b)^{-1} \times (T^c)^{-1} \times \begin{bmatrix} X_B \\ Y_B \\ Z_B \\ 1 \end{bmatrix} \tag{4.13}$$

La projection perspective du point $P_B$ respecte le modèle sténopé décrit dans le chapitre 2. On a donc l'expression suivante :

$$s. \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} X_B \\ Y_B \\ Z_B \\ 1 \end{bmatrix} \tag{4.14}$$

Ce qui donne après développement et en retirant le facteur d'échelle s :

$$\begin{cases} u = \alpha_u \left( \frac{R_{11}.X_B + R_{12}.Y_B + R_{13}.Z_B + t_x}{R_{31}.X_B + R_{32}.Y_B + R_{33}.Z_B + t_z} \right) + u_0 \\ v = \alpha_v \left( \frac{R_{21}.X_B + R_{22}.Y_B + R_{23}.Z_B + t_y}{R_{31}.X_B + R_{32}.Y_B + R_{33}.Z_B + t_z} \right) + v_0 \end{cases} \tag{4.15}$$

Dans l'équation précédente, seule la coordonnée $v$ est valide car la projection Sténopé ne s'applique que sur une ligne. Les paramètres $\alpha_u$ et $u_0$ disparaissent car ici, la projection

sténopé ne s'applique que verticalement. Ce qui va définir la coordonnée $u$ c'est la position angulaire de la caméra linéaire. Il est donc nécessaire de calculer la colonne $u$ séparément avec une autre méthode. Pour ce faire, il faut passer par les coordonnées du point $P$ dans le repère $R_C$ sans faire de projection :

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = (T^a)^{-1} \times (T^b)^{-1} \times (T^c)^{-1} \times \begin{bmatrix} X_B \\ Y_B \\ Z_B \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} X_B \\ Y_B \\ Z_B \\ 1 \end{bmatrix} \tag{4.16}$$

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11}.X_B + R_{12}.Y_B + R_{13}.Z_B + t_x \\ R_{21}.X_B + R_{22}.Y_B + R_{23}.Z_B + t_y \\ R_{31}.X_B + R_{32}.Y_B + R_{33}.Z_B + t_z \\ 1 \end{bmatrix} \tag{4.17}$$

La coordonnée $X_C$ doit être nulle pour que le point $P$ soit dans le plan $Y_B O Z_B$ qui est dans le champ de vision de la caméra linéaire :

$$X_C = R_{11}.X_B + R_{12}.Y_B + R_{13}.Z_B + t_x = 0 \tag{4.18}$$

Sachant qu'en développant la matrice de transformation $T$ on obtient :

$$\begin{aligned} R_{11} =& \cos(\phi).\cos(\theta)(\cos(\alpha).\cos(\omega) - \sin(\alpha).\sin(\omega).\cos(\beta)) \\ & - \sin(\theta).(\sin(\alpha).\cos(\omega) + \cos(\alpha).\sin(\omega).\cos(\beta)) \\ & + \sin(\phi).\cos(\theta).\sin(\omega).\sin(\beta) \end{aligned} \tag{4.19}$$

$$\begin{aligned} R_{12} =& (\cos(\phi).\sin(\theta).\sin(\psi) - \sin(\phi).\cos(\psi)).(\cos(\alpha).\cos(\omega) - \sin(\alpha).\sin(\omega).\cos(\beta)) \\ & + \sin(\omega).\sin(\beta).(\sin(\phi).\sin(\theta).\sin(\psi) + \cos(\theta).\cos(\psi)) \\ & + \cos(\theta).\sin(\psi).(\sin(\alpha).\cos(\omega) + \cos(\alpha).\sin(\omega).\cos(\beta)) \end{aligned} \tag{4.20}$$

$$\begin{aligned} R_{13} =& (\cos(\phi).\sin(\theta).\cos(\psi) + \sin(\phi).\sin(\psi)).(\cos(\alpha).\cos(\omega) - \sin(\alpha).\sin(\omega).\cos(\beta)) \\ & + \sin(\omega).\sin(\beta).(\sin(\phi).\sin(\theta).\cos(\psi) - \cos(\theta).\sin(\psi)) \\ & + \cos(\theta).\cos(\psi).(\sin(\alpha).\cos(\omega) + \cos(\alpha).\sin(\omega).\cos(\beta)) \end{aligned} \tag{4.21}$$

$$t_x = d.\sin(\theta) \tag{4.22}$$

En factorisant l'équation $X_C = 0$ on obtient une équation de type $A.\cos(\alpha) + B.\sin(\alpha) = C$ avec $A$, $B$, et $C$ des constantes qui dépendent des angles $\beta$, $\omega$, $\psi$, $\phi$, $\theta$ et des cordonnées du point $P_B$. La solution de cette équation peut être trouvée en utilisant l'identité remarquable suivante :

$$A.\cos(\alpha) + B.\sin(\alpha) = R.\cos(\alpha - x) \tag{4.23}$$

$$Avec : \qquad R = \sqrt{A^2 + B^2} \qquad et \qquad x = \arctan(\frac{B}{A}) \tag{4.24}$$

Cela nous permet d'exprimer notre problème de la manière suivante :

$$R.\cos(\alpha - x) = C \tag{4.25}$$

En isolant l'angle $\alpha$ on obtient l'équation :

$$\alpha = \arccos\left(\frac{C}{\sqrt{A^2 + B^2}}\right) + \arctan\left(\frac{B}{A}\right) \tag{4.26}$$

La solution de l'équation donne la valeur de l'angle $\alpha$ tel que la plateforme rotative soit dans la position ou le point P de l'espace est inclut dans le plan image. Il existe deux solutions possibles à cette équation :

- la première est celle ou le point P se trouve devant la caméra (la coordonné $Z_C$ du point P dans le repère caméra est positive),
- la deuxième est celle ou le point P se trouve derrière la caméra (la coordonné $Z_C$ du point P dans le repère caméra est négative).

La bonne solution est trouvée en injectant les deux solutions dans l'équation 4.17 et en gardant le cas ou la coordonnée $Z_C$ est positive.

Une fois la valeur $\alpha$ calculée, la relation qui la définit selon la colonne de l'image est $\alpha = \frac{2.\pi.u}{W}$. Avec $W$ qui est le nombre de colonnes par image panoramique soustrait de 1.

On a donc un couple de coordonnées $(u, v)$ avec pour expression :

$$\begin{cases} u = \frac{W}{2.\pi}.\left(\arccos\left(\frac{C}{\sqrt{A^2+B^2}}\right) + \arctan\left(\frac{B}{A}\right)\right) & tel \quad que \quad Z_C > 0 \\ v = \alpha_v \left(\frac{R_{21}.X_B + R_{22}.Y_B + R_{23}.Z_B + t_y}{R_{31}.X_B + R_{32}.Y_B + R_{33}.Z_B + t_z}\right) + v_0 \end{cases} \tag{4.27}$$

### 4.3.3 *Prise en compte des erreurs*

Le tableau 4.6 résume les paramètres du système qui a été présenté.

Ici, le paramètre $s_{uv}$ est négligé. Il n'y a pas réellement de paramètres extrinsèques car on exprime les coordonnées des objets directement par rapport au repère base $\mathcal{R}_B$ de la caméra. De plus, un paramètre "extrinsèque" au sens classique du terme est un paramètre non lié à l'optique ni à la rétine. Cependant, nous nous trouvons dans un cas qui n'est pas classique et le capteur a une partie mécanique dont les paramètres de mouvement sont internes au système. Cela dit on peut aussi considérer les deux paramètres $\omega$ et $\beta$ comme des paramètres extrinsèques et le repère base $\mathcal{R}_B$ comme le repère monde, c'est pour cela que ces deux paramètres figurent dans la ligne des paramètres extrinsèques dans le tableau.

Un autre aspect important est celui des angles $\psi$, $\theta$ et $\phi$ décrivant l'orientation du capteur. Nous avons vu qu'il est essentiel de modéliser ces angles et de ne pas les négliger pour

| Type de Paramètre | Nom | Fonction |
|---|---|---|
| Intrinsèques | $\alpha_v$ | le changement d'échelle entre les coordonnées métriques et les coordonnées en pixels. Il est égal au rapport de la distance focale sur la taille verticale d'un pixel. |
| | $v_0$ | la valeur de la ligne ou l'axe optique (L'axe $Z_C$ du repère caméra $R_C$) est sécant avec le plan image. |
| | $W$ | Le nombre de colonnes dans l'image panoramique |
| | d | le rayon du cercle de base |
| | $\psi$, $\theta$ et $\phi$ | les trois angles décrivant l'orientation libre de l'axe optique du capteur par rapport au repère $\mathcal{R}_T$ |
| Extrinsèques | $\omega$, $\beta$ | Les deux angles qui décrivent l'inclinaison de l'axe de rotation par rapport au plan horizontal du repère monde $\mathcal{R}_B$. Ces deux paramètres peuvent aussi être considérés comme étant extrinsèques. |

TABLE 4.6: Les neuf paramètres de notre modèle mono caméra

avoir une modélisation réaliste. Cependant, une valeur non nulle à l'un de ces angles provoque des distorsions dans l'image et elle ne sera plus de type cylindrique. La figure 4.6 présente l'effet de ces angles sur la géométrie de l'image.

Lorsque les trois angles $\psi$, $\theta$ et $\phi$ sont nuls, le repère caméra $R_C$ possède la même orientation que le repère $R_T$. Dans cette configuration précise, l'image panoramique est un cylindre, et c'est ce que nous cherchons à obtenir dans l'idéal. Des valeurs non nulles pour les angles $\psi$ et $\phi$ altèrent la forme cylindrique de l'image. Pour garder une projection cylindrique, il faut donc faire en sorte de fabriquer des systèmes avec ces deux angles le plus proche possible de zéro (car il est impossible en pratique d'avoir zéro exactement) afin d'avoir des distorsions négligeables.

Quant à l'angle $\theta$, il décrit un mouvement de rotation autour de l'axe $Y_c$ parallèle à l'axe de rotation. Ce mouvement se caractérise par un effet de "Lacet" sur le capteur, comme l'illustre la figure 4.6. L'effet de lacet ne crée pas de distorsions et l'image reste cylindrique. Cet angle a même une valeur volontairement différente de zéro dans les travaux de Huang et al (entre 0 et $\frac{\pi}{2}$) afin de pouvoir faire de la stéréovision. De plus, le lacet ne modifie pas l'homogénéité de la résolution de l'image ni n'altère sa cylindricité, il est donc tout à fait tolérable. En bilan, lors du design d'un système rotatif, il faut viser des angles $\psi = 0$ et $\phi = 0$, avec la possibilité d'une plus grande souplesse pour l'angle $\theta$.

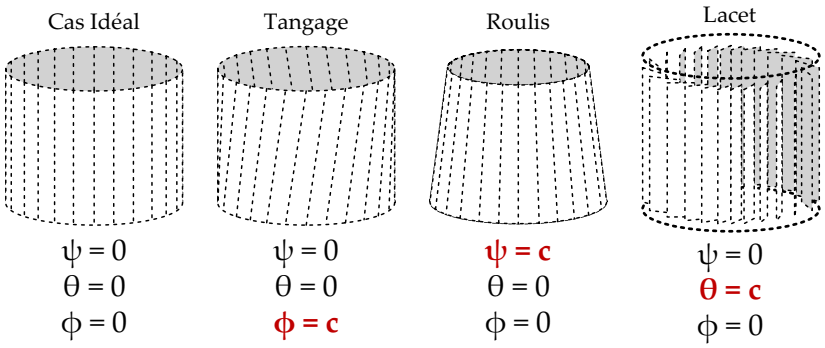


FIGURE 4.6: Pour chaque angle $\psi$, $\theta$ et $\phi$, une valeur différente de zéro produit une distorsion spécifique sur l'image cylindrique. La distorsion provoquée par l'angle $\theta$ n'altère pas l'image. Les distorsions provoquées par les angles $\psi$ et $\phi$ altèrent la géométrie cylindrique de l'image et sont donc à minimiser. En pratique elles provoquent des distorsions négligeables, mais elles sont tout de même prisent en compte dans le modèle pour obtenir une plus grande précision du système.

Avec ce modèle, la prise en compte des erreurs pointées au cours de l'étude des modèles existants est faite dès le début et nous avons bien un modèle général qui peut être adapté à tout système linéaire rotatif. Le tableau 4.7 montre la comparaison de tous les modèles étudiés avec celui qui vient d'être présenté.

| Modèles | Nb. de paramètres | | Erreur de l' axe de rotation | Erreur de décentrage | Erreur d' orientation |
|---|---|---|---|---|---|
| | Intrins. | Extrins. | | | |
| Schneider et al. | 6 | 6 | idéal | unique | idéale |
| Schneider et al.(complète) | 25 | 6 | 1 paramètre | multiples (2 param) | 2 paramètres |
| Smadja et al. | 4 | 6 | idéal | unique | idéale |
| Parian et al. | 4 | 6 | idéal | unique | idéale |
| Parian et al.(complète) | 16 | 6 | 6 paramètres | multiples | complète |
| Huang et al. | 4 | 0 | idéal | multiples | 1 paramètre |
| Cette thèse | 9 | 0 | complète | multiples | complète |

TABLE 4.7: Bilan comparatif des modèles étudiés avec le modèle développé au cours de cette thèse.

Le nombre de paramètres reste raisonnable en comparaison avec les travaux qui présentent une précision comparable (22 à 31 paramètres pour les modèles de Parian et al. et de Schneider et al. contre 9 pour notre modèle). Les singularités sont aussi évitées avec l'inclinaison de la plateforme par rapport au repère de la base. La complexité du système est

donc minimale par rapport à la généralité de la modélisation et une méthode de calibration simple est présentée dans le chapitre 5.

## 4.4 GÉOMÉTRIE ÉPIPOLAIRE

Pour pouvoir faire de la reconstruction 3D, il faut définir une géométrie épipolaire spécifique à notre système. Dans cette section nous allons faire un bref rappel de géométrie épipolaire classique (utilisant le modèle sténopé). L'équation d'une droite épipolaire sera calculée avec les méthodes classiques au sein d'un exemple type. Ensuite nous utiliserons la démarche et l'adapterons au cas de systèmes cylindriques. Nous commencerons par un cas général et analyserons ensuite des cas particuliers afin de déterminer la configuration stéréoscopique la plus adaptée à nos besoins.

### 4.4.1 *Géométrie épipolaire avec le modèle sténopé*

Lorsque un même point est capturé par deux caméras différentes, positionnées à des endroits différents, on obtient deux images de ce point. La géométrie épipolaire permet d'établir la correspondance des points entre deux images puis de retrouver les coordonnées 3D des points par triangulation. La figure 4.7 illustre ce principe.

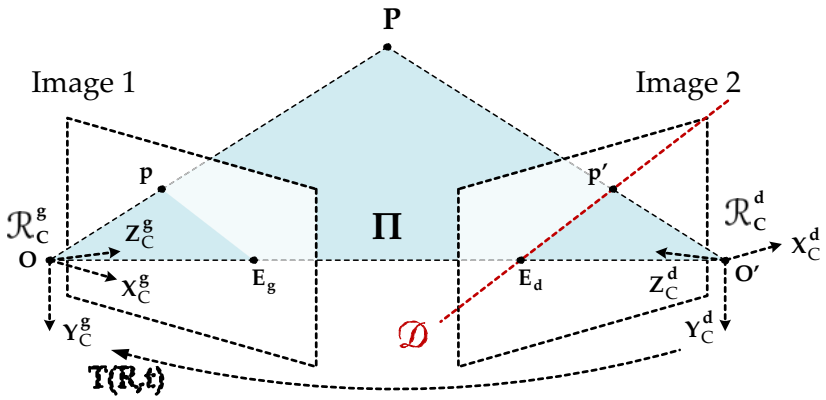


FIGURE 4.7: Géométrie épipolaire classique. Un point P est projeté sur deux images et donne naissance aux points $m(u,v)$ et $m'(u',v')$.

Soit deux caméras disposées comme sur le schéma et donnant lieux aux images 1 et 2 qui sont respectivement les images gauche et droite. Chaque caméra dispose de son repère propre, $\mathcal{R}_C^g$ pour la caméra gauche et $\mathcal{R}_C^d$ pour la caméra droite. Les centres de projection des caméras sont respectivement les points O et O′. Soit un point P visible sur les deux images, la projection du point P sur la caméra de gauche est le point $p(u,v)^T$ et $p'(u',v')^T$ sur la caméra de droite. Enfin la droite qui relie les deux centres optiques coupe les deux plans images en $E_g$ et en $E_d$ qui sont les deux épipôles.
Soit Π, le plan qui contient le point P et les deux centres de projections des caméras. La conséquence directe est que le plan Π contient aussi les deux projections du point P et les épipôles.
En traitement de la vision, pour une projection p d'un point P sur l'image de gauche, on va chercher sa deuxième projection p′ sur l'image de droite afin de pouvoir trianguler la position du point réel P. La première étape consiste donc à trouver la deuxième projection p′ dans l'image de droite. Et selon la géométrie du système, le point p′ se trouve sur la droite $\mathcal{D}$ définie par l'intersection du plan Π avec du plan image de la caméra de droite. Cette droite $\mathcal{D}$ est la droite épipolaire qui correspond au point p, et nous allons présenter

le calcul de son équation :

Soit T la transformation qui décrit le mouvement du repère $\mathcal{R}_C^d$ au repère $\mathcal{R}_C^g$. Cette transformation est composée d'une rotation R et d'une translation t.

On a l'égalité suivante :

$$(\overrightarrow{Op} \wedge \overrightarrow{OO'}).\overrightarrow{O'p'} = 0 \tag{4.28}$$

Or on peut voir sur le schéma que :

$$\overrightarrow{OO'} = t \qquad \overrightarrow{O'p'} = R.p' \qquad \overrightarrow{Op} = p \tag{4.29}$$

L'égalité devient donc $(p \wedge t).R.p' = 0$. Le produit scalaire $p \wedge t$ peut être exprimé différemment selon la formule $p \wedge t = p^T.[t]_X$ ou $[t]_X$ est la matrice anti-symétrique de t. On obtient :

$$p^T.([t]_X.R).p' = 0 \qquad avec \qquad E = [t]_X.R \tag{4.30}$$

E est la matrice essentielle du système et décrit la relation entre les points p et p′ en fonction du mouvement$T(R,t)$.

En projetant les points p et p′ sur les images gauche et droite respectivement, on obtient les points m et m′ de coordonnées $(u,v)$ et $(u',v')$ respectivement.

$$(K^{-1}.m).E.(K^{-1}.m') = 0 \qquad m^T.(K^{-1})^T.E.K^{-1}.m' = 0 \tag{4.31}$$

Ici, $F = (K^{-1})^T.E.K^{-1}$ est la matrice fondamentale du système qui contient toute l'information nécessaire sur la géométrie épipolaire du système. Elle vérifie l'équation $m^T.F.m' = 0$.

A partir d'ici, on peut exprimer les coordonnées $(u,v)$ et $(u',v')$ de cette manière :

$$\left[\begin{pmatrix} u & v & 1 \end{pmatrix}.F\right].\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \tag{4.32}$$

Ce qui permet d'exprimer les coefficients a, b et c tels que :

$$\begin{pmatrix} a & b & c \end{pmatrix}.\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \tag{4.33}$$

Le résultat est l'équation de la droite épipolaire qui est l'ensemble des points pouvant être la projection du point P dans l'image de droite. Ceci correspond à l'équation de la droite $\mathcal{D}$ dans l'image de droite, et peut être écrit de la forme $a.u' + b.v' + c = 0$ avec :

$$\begin{aligned} a &= (t_z.r_2 - t_y.r_3).p \\ b &= (t_z.r_1 - t_x.r3).p \\ c &= (t_x.r2 - t_y.r1).p \end{aligned} \tag{4.34}$$

En d'autres termes, il est possible d'exprimer la relation comme suit :

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \cdot \begin{bmatrix} U \\ V \\ 1 \end{bmatrix} \tag{4.35}$$

Avec $E = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ et $U = \frac{u-u_0}{\alpha_u}$ et $V = \frac{v-v_0}{\alpha_v}$.

Ainsi il est possible de calculer l'équation de la droite $\mathcal{D}$ qui correspond à l'ensemble des points dans l'image de droite où il est possible de retrouver la projection du point P connaissant les coordonnées de sa projection dans l'image de droite, la transformation qui mène d'une caméra à l'autre et les paramètres intrinsèques des deux caméras. En stéréovision, il est utile de connaitre l'équation de cette droite pour éviter de perdre du temps et des ressources en cherchant la projection du point P dans toute l'image.
Il est aussi possible de simplifier cette équation en choisissant des cas particuliers de transformation. Par exemple, si la transformation entre les deux repères est une translation pure en X, cela revient à dire que les deux plans images sont inclus dans un même plan :

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} U \\ V \\ 1 \end{bmatrix} \tag{4.36}$$

On obtient :

$$\begin{aligned} a &= 0 \\ b &= -t_x \\ c &= t_x.V \end{aligned} \tag{4.37}$$

En ré-injectant ces résultats dans l'équation de la droite $\mathcal{D}$, on trouve $V = V'$, ce qui veut dire que la projection du point P a la même coordonnée verticale, il suffit donc d'aller chercher sur la ligne de l'image de gauche ayant cette coordonnée verticale. Cela simplifie et accélère grandement le processus de recherche et c'est pour cela que les fabricants de caméras stéréoscopiques essaient souvent de se rapprocher le plus possible de ce cas particulier idéal. Sinon, on effectue une rectification stéréoscopique pour simuler ce cas. La connaissance de la réalité géométrique épipolaire est donc essentielle dans le design d'un capteur stéréoscopique.

### 4.4.2 *Géométrie éipolaire cylindrique*

Nous avons vu que l'établissement de l'équation des droites épipolaires est essentiel dans le processus de design d'un système stéréoscopique. Nous avons également vu comment établir l'équation de ces droites dans le cas sténopé classique. Dans cette partie, nous allons appliquer la même méthodologie pour le cas de nos modèles cylindriques. En effet, ils ne

peuvent pas être considérés complètement comme des modèles sténopés. Bien que cela soit possible verticalement, ça ne l'est pas horizontalement. De plus le fait d'avoir un centre de projection pour chaque colonne d'image rends la tâche encore plus complexe.

Soit deux caméras linéaires $\mathcal{S}_g$ et $\mathcal{S}_d$ qui capturent respectivement les images $I_1$ et $I_2$. Les deux caméras disposent de leurs paramètres propres et ont été calibrées au préalable. Les paramètres sont donc considérés comme des constantes connues dans les calculs qui vont suivre. Ces paramètres sont $\alpha_{v_g}$, $v_{0_g}$, $W_g$, $d_g$, $\psi_g$, $\theta_g$ et $\phi_g$ pour la caméra $\mathcal{S}_g$ et $\alpha_{v_d}$, $v_{0_d}$, $W_d$, $d_d$, $\psi_d$, $\theta_d$ et $\phi_d$ pour la caméra $\mathcal{S}_d$. Soit $T_{dg}$ la transformation qui décrit le mouvement de la caméra $\mathcal{S}_d$ vers la caméra $\mathcal{S}_g$, cette transformation comporte une rotation $R^{dg}(\delta_x, \delta_y, \delta_z)$ et une translation $t^{dg}(t_x, t_y, t_z)$. En jouant sur les contraintes appliquées à ce mouvement, on peut créer des cas particuliers plus ou moins réalisables.
Il existe différentes configurations possibles qui permettent la stéréovision cylindrique à partir de deux caméras linéaires, qu'elles soient sur la même plateforme rotative ou avec leur plateforme rotative propre. Les travaux présentés par Huang et al. [54, 57] présentent une classification de ces configurations. La classification comporte six cas de configurations distincts :

- *Multi-view Panoramas* : C'est le cas le plus général, il n'y a aucune contrainte sur le mouvement $T_{21}$. On a les six degrés de libertés : $R^{dg}(\delta_x, \delta_y, \delta_z)$ et $t^{dg}(t_x, t_y, t_z)$.
- *Parallel-axis Panoramas* : Les axes de rotation des deux systèmes sont parallèles. On a quatre degrés de liberté : $R^{dg}(0, \delta_y, 0)$ et $t^{dg}(t_x, t_y, t_z)$.
- *Levelled Panoramas* : Les plans de base des deux caméras sont confondus. On a trois degrés de liberté : $R^{dg}(0, \delta_y, 0)$ et $t^{dg}(t_x, 0, t_z)$.
- *Co-axis Panoramas* : Les deux axes de rotation sont confondus, il est possible à partir de cette configuration d'utiliser le même plateau tournant pour les deux caméras. On a deux degrés de liberté : $R^{dg}(0, \delta_y, 0)$ et $t^{dg}(0, t_y, 0)$.
- *Concentric Panoramas* : Les plans de base des deux caméras sont confondus, ce qui revient à dire que les axes de rotations sont confondus et que les centres de rotations sont confondus également. On a un seul degré de liberté : $R^{dg}(0, \delta_y, 0)$ et $t^{dg}(0, 0, 0)$.
- *Symmetric Panoramas* : Les plans de base sont confondus avec un axe de rotation commun aux deux caméras. Les deux caméras possèdent des paramètres identiques et sont disposées symétriquement sur le plateau tournant. On n'a aucun degré de liberté : $R^{dg}(0, 0, 0)$ et $t^{dg}(0, 0, 0)$.

Cette classification permet de partir du cas le plus général pour descendre dans les cas particuliers afin d'arriver au cas *Symmetric Panoramas* qui est finalement utilisé dans les travaux cités. Ce dernier cas de géométrie est cependant à la fois un cas particulier mais aussi un cas idéal qui n'est pas réalisable en pratique. En effet lorsque l'on dispose deux caméras linéaires sur un plateau tournant, les deux centres optiques décrivent deux cercles distincts qui ne sont jamais inclus dans un plan commun. Cela est dû aux imperfections du mouvement de rotation et aux imperfections de positionnement des caméras sur le plateau. La figure 4.8 présente une hiérarchisation de la classification.

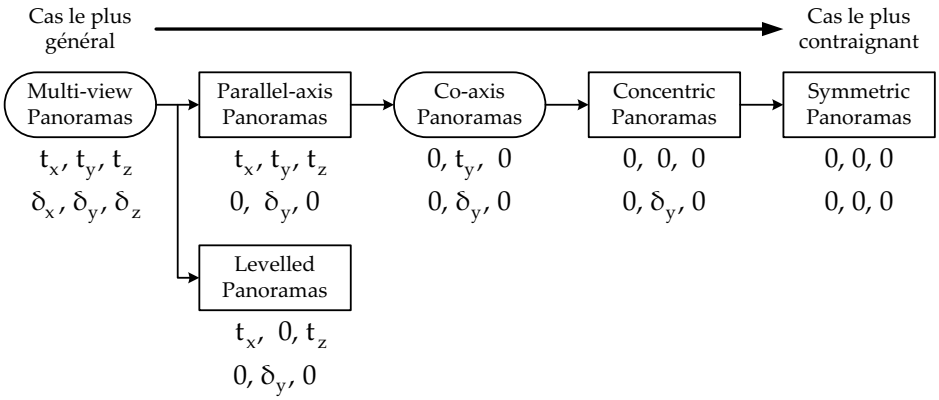


FIGURE 4.8: Classification hiérarchisé des différentes configurations possibles pour deux caméras linéaires

Il n'y a que deux cas réellement réalisables ; le cas général sans contrainte de mouvement, et le cas co-axis. Toutes les configurations qui placent les cercles de base des deux caméras dans un plan commun (*levelled*, *concentric* et *symmetric*) sont irréalistes. De plus, il n'est pas possible non plus d'avoir les axes de rotation de deux systèmes parallèles. D'un autre coté, si les deux caméras sont placées sur le même plateau tournant, elles auront un axe de rotation commun. Bien que cet axe est en pratique différent de celui qui est théoriquement prévu par le design (à cause des imperfections mécaniques), il reste commun aux deux caméras, ce qui rend le cas "co-axis" réalisable. Les deux caméras ainsi disposées n'auront par contre pas de centre de rotation commun.
Pour les cas non réalisables, il est possible d'avoir une précision géométrique suffisante et de pouvoir considérer qu'on est dans un cas idéal. Dans notre cas, nous allons chercher à éviter le plus possible les approximations et d'éviter les coûts de fabrication élevés relatifs aux plateformes mécaniques précises.
Dans cette partie, nous allons étudier la configuration la plus générale (multi-view panoramas) et la configuration à axe de rotation unique (co-axis panoramas). Les quatre autres configurations n'ont pas été retenues à cause de leur manque de réalisme pratique. Elles sont néanmoins développées en annexe A.

### 4.4.3 *Le cas "Multi-view Panoramas"*

Le cas le plus général de géométrie, est celui où les deux systèmes $\mathcal{S}_g$ et $\mathcal{S}_d$ n'ont aucun paramètre en commun et n'ont pas de contrainte de positionnement l'un par rapport à l'autre. Leurs paramètres intrinsèques sont donc distincts et sont considérés comme connus. La figure 4.9 illustre ce cas général appelé *Multi-view* :

Soit deux caméras linéaires $\mathcal{S}_g$ et $\mathcal{S}_d$ sur leur plateforme rotative respective et capturant les images $I_g$ et $I_d$. Soit P, un point de l'espace visible sur les deux images. Le point P est projeté en $m_g$ de coordonnées connues $(u_g, v_g)$ sur l'image $I_g$. A partir de cette projection connue, des paramètres des deux caméras et du mouvement entre elles, il est possible de calculer la courbe épipolaire qui est l'ensemble des projections $m_d$ possibles du point P dans l'image $I_d$.

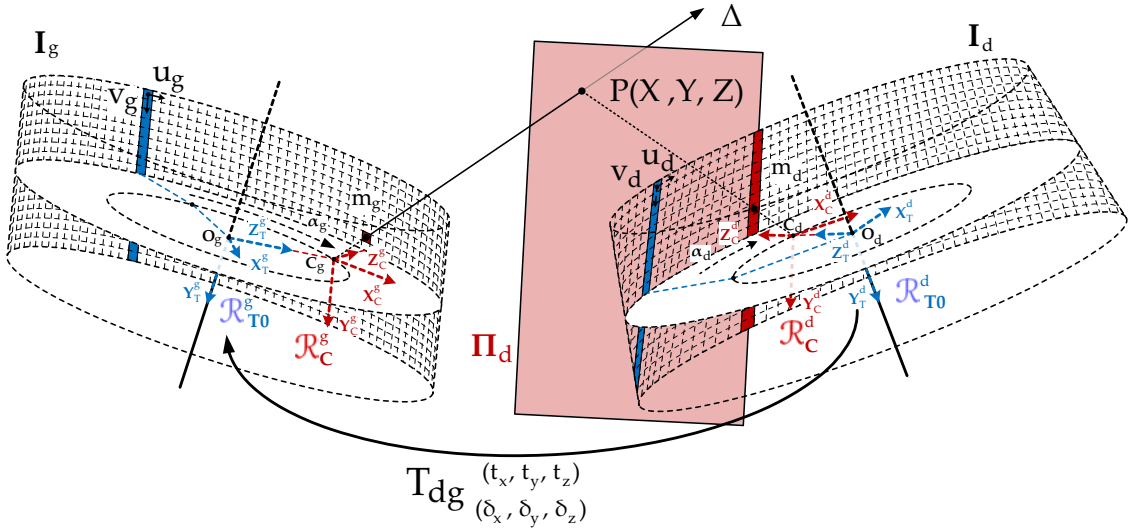


FIGURE 4.9: Le cas *Multi-view* est caractérisé par un mouvement $T_{dg}$ 3D avec 6 degrés de liberté.

Soit $\Delta^g_C$ la droite qui passe par le point P et par le centre de projection $C_g$ de la caméra $S_g$. La droite $\Delta^g_C$ intersect l'image en $m_g(u_g, v_g)^T$ et a donc pour équation dans le repère $\mathcal{R}^g_C$ :

$$\Delta^g_C = \lambda. \begin{bmatrix} 0 \\ \sin(\beta) \\ \cos(\beta) \end{bmatrix} \quad \text{avec} \quad \beta = \arctan(V_g) \quad \text{et} \quad V_g = \frac{v_g - v_{0g}}{\alpha_{vg}} \tag{4.38}$$

Selon le repère caméra $\mathcal{R}^d_C$ de la caméra $S_d$, la droite $\Delta^g_C$ devient $\Delta^d_C$ :

$$\Delta^d_C = (T^a_d)^{-1}.(T^b_d)^{-1}.T_{dg}.T^b_g.T^a_g.\Delta^g_C \tag{4.39}$$

Avec les transformations $T^a$ et $T^b$ décrivant les mouvement des repères $\mathcal{R}_T$ vers $\mathcal{R}_C$ et $\mathcal{R}_T$ vers $\mathcal{R}_{T0}$ respectivement. La transformation $T_{dg}$ quant à elle décrit le mouvement qui a lieu entre les deux caméras, c'est à dire le mouvement qui mène le repère $\mathcal{R}^d_{T0}$ de la caméra $S_d$ au repère $\mathcal{R}^g_{T0}$ de la caméra $S_g$. Ce dernier est un mouvement sans contrainte comportant une rotation 3D d'angles $(\delta_x, \delta_y, \delta_z)$ et une translation 3D de vecteur $(t_x, t_y, t_z)$. Les coordonnées de la projection du point P dans l'image $I_d$ sont inconnues. L'angle $\alpha_d$, qui est inconnu, indique la position de la plateforme rotative associée à $S_d$ lorsque le point P est dans son champ de vision. La valeur $\alpha_d$ indique donc la ligne de l'image où se trouve la projection du point P, cela est illustré en rouge sur la figure 4.9. En développant l'expression de $\Delta^d_C$, on trouve une expression de la forme :

$$\Delta^d_C = \begin{bmatrix} \lambda.(Q_{11}.\cos(\alpha_d) + Q_{12}.\sin(\alpha_d) + Q_{13}) + Q_{14}.\cos(\alpha_d) + Q_{15}.\sin(\alpha_d) + Q_{16} \\ \lambda.(Q_{21}.\cos(\alpha_d) + Q_{22}.\sin(\alpha_d) + Q_{23}) + Q_{24}.\cos(\alpha_d) + Q_{25}.\sin(\alpha_d) + Q_{26} \\ \lambda.(Q_{31}.\cos(\alpha_d) + Q_{32}.\sin(\alpha_d) + Q_{33}) + Q_{34}.\cos(\alpha_d) + Q_{35}.\sin(\alpha_d) + Q_{36} \end{bmatrix} \tag{4.40}$$

Avec les valeurs $Q_{11}$ à $Q_{36}$ qui sont des variables dépendantes des paramètres des différentes transformations. Nous avons donc l'équation de la droite $\Delta^d_C$ dans le repère $\mathcal{R}^d_C$.

Dans ce même repère, la caméra $\mathcal{S}_d$ a son champ de vision décrit par le plan $\Pi$ comme illustré sur la figure 4.9. Dans le repère $\mathcal{R}_C^d$, le plan $\Pi$ a pour équation $x = 0$. En injectant cela dans l'équation de la droite $\Delta_C^d$ on obtient :

$$(Q_{11}.\cos(\alpha_d) + Q_{12}.\sin(\alpha_d) + Q_{13}).\lambda + Q_{14}.\cos(\alpha_d) + Q_{15}.\sin(\alpha_d) + Q_{16} = 0 \quad (4.41)$$

Et donc :

$$\lambda = -\frac{Q_{14}.\cos(\alpha_d) + Q_{15}.\sin(\alpha_d) + Q_{16}}{Q_{11}.\cos(\alpha_d) + Q_{12}.\sin(\alpha_d) + Q_{13}} \quad (4.42)$$

Puis en ré injectant $\lambda$ dans l'équation de la droite $\Delta_C^d$ on obtient les coordonnées du point $P_d$ dans le repère $\mathcal{R}_C^d$ en fonction de l'angle $\alpha_d$ :

$$\Delta_C^d = \begin{bmatrix} X_C^d \\ Y_C^d \\ Z_C^d \end{bmatrix} =$$

$$\begin{bmatrix} 0 \\ -\frac{(Q_{14}.\cos(\alpha_d)+Q_{15}.\sin(\alpha_d)+Q_{16}).(Q_{21}.\cos(\alpha_d)+Q_{22}.\sin(\alpha_d)+Q_{23})}{Q_{11}.\cos(\alpha_d)+Q_{12}.\sin(\alpha_d)+Q_{13}} + Q_{24}.\cos(\alpha_d) + Q_{25}.\sin(\alpha_d) + Q_{26} \\ -\frac{(Q_{14}.\cos(\alpha_d)+Q_{15}.\sin(\alpha_d)+Q_{16}).(Q_{31}.\cos(\alpha_d)+Q_{32}.\sin(\alpha_d)+Q_{33})}{Q_{11}.\cos(\alpha_d)+Q_{12}.\sin(\alpha_d)+Q_{13}} + Q_{34}.\cos(\alpha_d) + Q_{35}.\sin(\alpha_d) + Q_{36} \end{bmatrix} \quad (4.43)$$

En simplifiant l'expression on obtient :

$$Y_C^d = \frac{\begin{array}{l}(Q_{11}Q_{24} - Q_{14}Q_{21}).\cos^2(\alpha_d) + (Q_{12}Q_{25} - Q_{15}Q_{22}).\sin^2(\alpha_d) + \\ (Q_{11}Q_{25} + Q_{12}Q_{24} - Q_{14}Q_{22} - Q_{15}Q_{21}).\cos(\alpha_d).\sin(\alpha_d) + \\ (Q_{11}Q_{26} + Q_{13}Q_{24} - Q_{14}Q_{23} - Q_{16}Q_{21}).\cos(\alpha_d) + \\ (Q_{12}Q_{26} + Q_{13}Q_{25} - Q_{15}Q_{23} - Q_{16}Q_{22}).\sin(\alpha_d) + (Q_{13}Q_{26} - Q_{16}Q_{23})\end{array}}{Q_{11}.\cos(\alpha_d) + Q_{12}.\sin(\alpha_d) + Q_{13}} \quad (4.44)$$

$$Z_C^d = \frac{\begin{array}{l}(Q_{11}Q_{24} - Q_{14}Q_{31}).\cos^2(\alpha_d) + (Q_{12}Q_{25} - Q_{15}Q_{32}).\sin^2(\alpha_d) + \\ (Q_{12}Q_{24} + Q_{13}Q_{25} - Q_{14}Q_{32} - Q_{15}Q_{31}).\cos(\alpha_d).\sin(\alpha_d) + \\ (Q_{11}Q_{26} + Q_{13}Q_{24} - Q_{14}Q_{33} - Q_{16}Q_{31}).\cos(\alpha_d) + \\ (Q_{12}Q_{26} + Q_{13}Q_{25} - Q_{15}Q_{33} - Q_{16}Q_{32}).\sin(\alpha_d) + (Q_{13}Q_{26} - Q_{16}Q_{33})\end{array}}{Q_{11}.\cos(\alpha_d) + Q_{12}.\sin(\alpha_d) + Q_{13}} \quad (4.45)$$

Le quotient de $Y_C^d$ par $Z_C^d$ élimine les deux dénominateurs. Il est aussi possible de simplifier les expressions de $Y_C^d$ et de $Z_C^d$ en utilisant les formules de linéarisation suivantes :

$$\cos^2(\alpha) = \frac{1 + \cos(2.\alpha)}{2} \qquad \sin^2(\alpha) = \frac{1 - \cos(2.\alpha)}{2} \qquad \cos(\alpha).\sin(\alpha) = \frac{\sin(2.\alpha)}{2} \quad (4.46)$$

La projection du point $P_d$ sur l'image $I_d$ présente donc une équation de la forme :

$$V_d = \frac{Y_C^d}{Z_C^d} = \frac{A_{11}.\cos(2\alpha_d) + A_{12}.\sin(2\alpha_d) + A_{13}.\cos(\alpha_d) + A_{14}.\sin(\alpha_d) + A_{15}}{A_{21}.\cos(2\alpha_d) + A_{22}.\sin(2\alpha_d) + A_{23}.\cos(\alpha_d) + A_{24}.\sin(\alpha_d) + A_{25}} \quad (4.47)$$

Avec :

$$\begin{aligned}
A_{11} &= (Q_{11}Q_{24} - Q_{14}Q_{21})/2 - (Q_{12}Q_{25} - Q_{15}Q_{22})/2 \\
A_{12} &= (Q_{11}Q_{25} + Q_{12}Q_{24} - Q_{14}Q_{22} - Q_{15}Q_{21})/2 \\
A_{13} &= Q_{11}Q_{26} + Q_{13}Q_{24} - Q_{14}Q_{23} - Q_{16}Q_{21} \\
A_{14} &= Q_{12}Q_{26} + Q_{13}Q_{25} - Q_{15}Q_{23} - Q_{16}Q_{22} \\
A_{15} &= (Q_{11}Q_{24} - Q_{14}Q_{21})/2 + (Q_{12}Q_{25} - Q_{15}Q_{22})/2 + Q_{13}Q_{26} - Q_{16}Q_{23} \\
A_{21} &= (Q_{11}Q_{24} - Q_{14}Q_{31})/2 - (Q_{12}Q_{25} - Q_{15}Q_{32})/2 \\
A_{22} &= (Q_{12}Q_{24} + Q_{13}Q_{25} - Q_{14}Q_{32} - Q_{15}Q_{31})/2 \\
A_{23} &= Q_{11}Q_{26} + Q_{13}Q_{24} - Q_{14}Q_{33} - Q_{16}Q_{31} \\
A_{24} &= Q_{12}Q_{26} + Q_{13}Q_{25} - Q_{15}Q_{33} - Q_{16}Q_{32} \\
A_{25} &= (Q_{11}Q_{24} - Q_{14}Q_{31})/2 + (Q_{12}Q_{25} - Q_{15}Q_{32})/2 + Q_{13}Q_{26} - Q_{16}Q_{33}
\end{aligned} \quad (4.48)$$

$A_{11}$, à $A_{25}$ sont des constantes qui sont fonctions des paramètres de calibrage des deux caméras et du mouvement $T_{dg}$. Sachant qu'on a $V_d = \frac{v_d - v_{0d}}{\alpha_{vd}}$ et $\alpha_d = \frac{2.\pi.u_d}{W_d}$. Cette équation décrit la courbe épipolaire dans l'image $I_d$ correspondant au point P de l'image $I_g$.
Une limite est à connaître ; comme l'équation est obtenue en calculant l'intersection d'une droite avec un plan, le cas où les deux plans images sont confondus (et que la droite est donc inscrite dans le plan) ne comporte pas d'intersection à proprement parler. dans ce cas, l'équation calculée ne peux pas être utilisée, et nous avons une droite épipolaire verticale avec l'égalité $V_d = V_g$. Ce cas idéal sera rencontré dans certaines des configurations suivantes.
La connaissance de l'équation de la courbe épipolaire est primordiale pour optimiser le temps de recherche de correspondance, mais sa complexité reste grande et il sera difficile de l'exploiter lors d'une recherche de correspondance en temps réel. Un autre problème se présente avec la configuration *multi-view* ; l'acquisition des images se fait en fonction du temps avec les systèmes rotatifs. De cela découle le fait que le moment où la première caméra capture le point P ne sera pas forcément le même pour l'autre caméra. Ce qui veut dire qu'il y a un facteur $\Delta t$ entre les deux acquisitions et, si l'objet, la scène ou une des caméras bouge, le calcul de la position du point P sera compromis et comportera une composante *mouvement*.
Afin d'analyser plus loin les équations épipolaires, il faut se pencher sur les relations qui existent entre les angles $\psi$, $\theta$ et $\phi$ et les équations épipolaires. Pour analyser méthodiquement l'effet des valeurs de ces angles, trois cas peuvent être décris :

- Cas A : La cas le plus général, où on considère les trois angles $\psi$, $\theta$ et $\phi$ comme quelconques,
- Cas B : Le cas cylindrique où les deux angles $\psi$ et $\phi$ sont nuls,
- Cas C : Le cas parfait où les trois angles $\psi$, $\theta$ et $\phi$ sont nuls.

L'intérêt de cette classification est de voir l'effet de la simplification de ces angles sur les courbes épipolaires et ainsi de savoir s'il est intéressant ou non de les choisir proches de zéro lors de la conception. Les cas A, B et C donnent lieu à une équation de la même forme dans la configuration *Multi-view*, il n'y a donc pas de simplification intéressante de l'équation de la courbe épipolaire.
La configuration A est la plus générale et est réalisable car il n'y a aucune contrainte de mouvement entre les deux systèmes. Cela dit, il n'est pas intéressant de faire de la stéréovision comme cela à cause de la complexité calculatoire des courbes épipolaires et du facteur $\Delta t$.

### 4.4.4 *Le cas "Co-axis Panoramas"*

Le cas suivant appelé *Co-axis Panoramas* présente la contrainte d'avoir un unique axe de rotation pour les deux systèmes. Ce qui est impossible à réaliser pour deux systèmes distincts mais réaliste si les deux caméras sont placées sur le même plateau tournant. La figure 4.10 présente la configuration de ce cas particulier de géométrie.

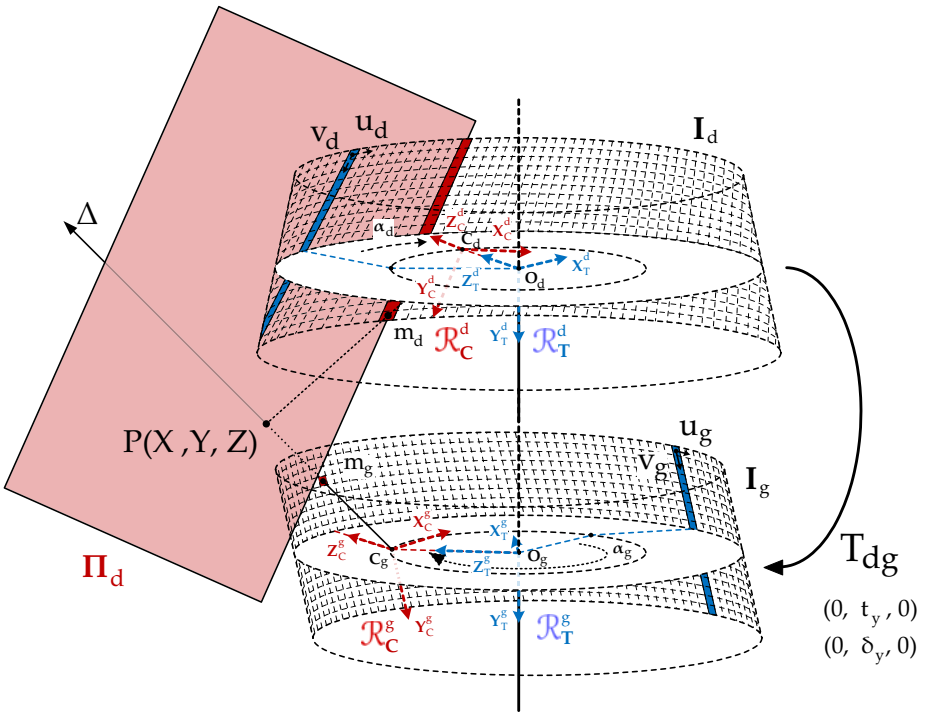


FIGURE 4.10: Le cas "Co-axis Panoramas" est caractérisé par un mouvement $T_{dg}$ 3D avec 2 degrés de liberté.

Dans cette configuration, seule une translation $t_y$ et une rotation $\delta_y$ sont permises, les autres composantes doivent être nulles. Le mouvement $T_{dg}$ comportera alors une rotation $R^{21}(0, \delta_y, 0)$ et une translation $t^{21}(0, t_y, 0)$.

En injectant ces paramètres dans l'équation 4.47, on obtient la nouvelle équation :

$$V_d = \frac{Y_C^d}{Z_C^d} = \frac{A'_{11}.\cos(2\alpha_d) + A'_{12}.\sin(2\alpha_d) + A'_{13}.\cos(\alpha_d) + A'_{14}.\sin(\alpha_d) + A'_{15}}{A'_{21}.\cos(2\alpha_d) + A'_{22}.\sin(2\alpha_d) + A'_{23}.\cos(\alpha_d) + A'_{24}.\sin(\alpha_d) + A'_{25}} \quad (4.49)$$

Avec $V_d = \frac{v_d - v_{0d}}{\alpha_{vd}}$ et $\alpha_d = \frac{2.\pi.u_d}{W_d}$. Et $A'_{11}$ à $A'_{25}$ des constantes qui sont fonction des paramètres des deux caméras et du mouvement $T_{dg}$.

Dans le cas B, l'équation devient la suivante :

$$V_d = \frac{\sin(\beta).[d_g.\sin(\theta_d + \alpha_d - \alpha_g - \delta_y) - d_d.\sin(\theta_d)] + \cos(\beta).[t_y.\sin(\theta_g + \alpha_g - \theta_d - \alpha_d + \delta_y]}{\cos(\beta).[d_g.\sin(\theta_g) - d_d.\sin(\theta_g + \alpha_g + \delta_y - \alpha_d)]} \tag{4.50}$$

Sachant que cette équation n'est valable que si $\theta_g$ est différent de $\theta_d$. Si $\theta_g = \theta_d$, les plans images sont confondus et on a $u_d = u_g$.
Dans le cas C, nous avons les deux plans images qui sont confondus. On a donc $u_d = u_g$ et la courbe épipolaire est une droite verticale.

Ici on obtient une simplification intéressante lorsque les plans images sont confondus. Pour l'obtenir, il faut que l'orientation des deux capteurs soit idéale avec les trois angles nuls. D'une manière générale, la configuration "Co-axis" est réalisable, car si on place deux caméras sur un même plateau, elles auront obligatoirement le même axe de rotation. Ce qui n'est pas réalisable c'est d'avoir les deux orientations idéales, mais il est possible de s'en rapprocher. Plus la mécanique d'un tel système sera précise, plus on se rapprochera du cas C qui présente l'égalité $u_d = u_g$. Ce cas est d'autant plus idéal qu'il permet aussi de n'avoir aucun $\Delta t$ entre la capture d'un point par les deux caméras puisqu'elles ont leurs plans de vision confondus. Si le placement des deux caméras sur le plateau tournant est suffisamment précis, on peut assimiler la courbe épipolaire à une droite d'équation $u_d = u_g$ et le $\Delta t$ peut être négligé. Sinon, avec une rectification numérique de l'image, il est possible de le corriger.

### 4.4.5 *Sélection de la meilleur configuration stéréoscopique*

En conclusion, deux configurations sont réalisables ; la configuration générale qui nécessite deux systèmes rotatifs, et la configuration *co-axis* qui n'en nécessite qu'un seul. La configuration *co-axis* est en plus idéale pour faire de la stéréovision car il est possible de minimiser le décalage temporel entre la capture du même point par les deux caméras. De plus, si le placement des caméras sur le plateau est suffisamment précis, on obtient des courbes épipolaires rectilignes ce qui facilite le travail d'éventuels algorithmes d'appariement.
Cependant, nous avons vu qu'il n'est pas possible de placer les caméras avec une précision qui permet d'avoir leurs plans optiques confondus. Les angles d'orientation des deux caméras ne sont jamais nuls, c'est pour cela que notre modélisation les prend en compte. Cela dit, il est intéressant de réaliser un système qui tente d'atteindre cette géométrie pour tendre vers ce cas idéal. En effet, cette configuration est la seule qui permette de se rapprocher le plus possible des caractéristiques idéales qui nous intéressent, à savoir :

- Générer des images cylindriques,
- Avoir des courbes épipolaires rectilignes ($u_d = u_g$),
- un facteur $\Delta t$ minimisé entre les deux acquisitions du même point.

Le mode *co-axis* est à la fois réalisable et présente une généralité complète. De plus, seuls deux paramètres sont nécessaires pour définir le mouvement entre les deux caméras,

un angle $\delta$, et un vecteur unitaire de translation $t$. Cette simplicité n'empêche pas qu'il soit possible, en jouant avec ces deux paramètres, d'obtenir tous les cas de stéréovision ; verticale et horizontale. C'est donc cette configuration que nous avons retenue.

## 4.5 MODÈLE STÉRÉOSCOPIQUE

### 4.5.1 *Modélisation stéréoscopique*

A présent que nous avons étudié les différentes géométries de stéréoscopie possibles, la configuration *co-axis* va être exploitée dans notre modélisation stéréoscopique. Cette configuration est également celle utilisée dans le système PanoraMOS présenté dans le chapitre 3.
Le modèle est présenté en figure figure 4.11.

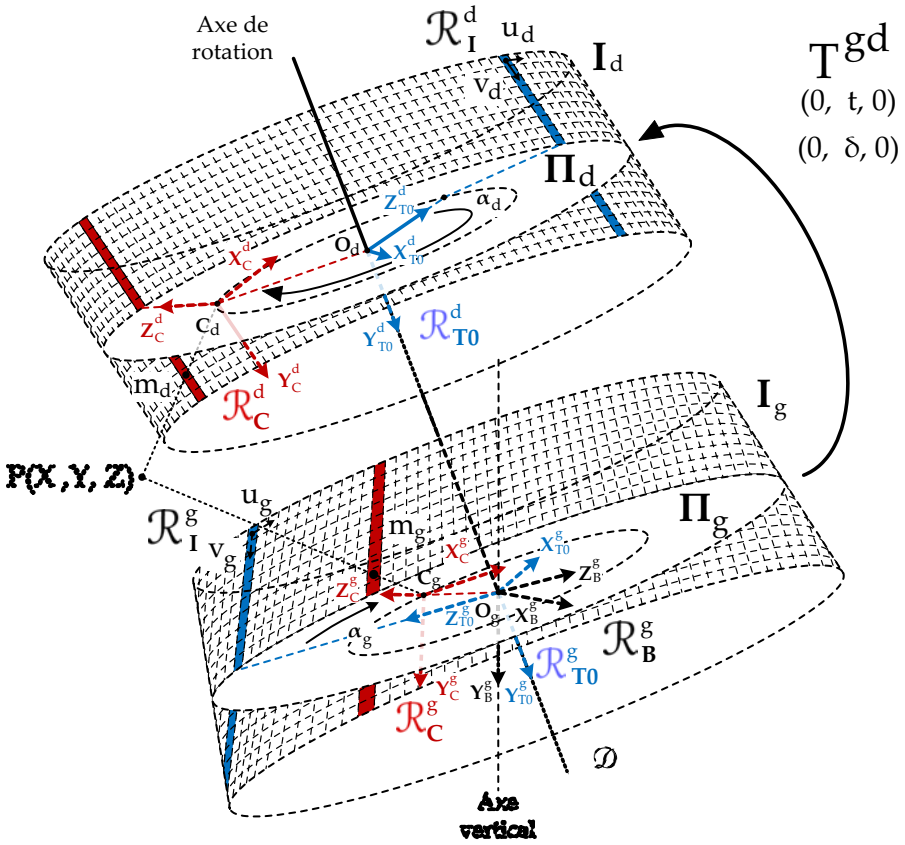


FIGURE 4.11: Modèle général d'un stéréoscopique à caméras linéaires rotatives. On constatera que les deux images ne sont pas tout à fait des cylindres, cela est dû à la modélisation des angles $(\phi_d, \theta_d, \psi_d)$ et $(\phi_g, \theta_g, \psi_g)$ de l'erreur d'orientation des capteurs.

Nous avons deux caméras, une principale et une secondaire. Sur le schéma, la caméra principale est celle du bas. Elle est nommée "caméra principale" car c'est elle qui va définir l'orientation de l'axe de rotation et l'orientation du plateau lorsque toutes les caméras captureront la première colonne de leur image respective.

Les deux caméras sont fixées rigidement sur la plateforme rotative. Leurs centres optiques vont donc effectuer des cercles autour de l'axe de rotation pendant le mouvement et ces deux cercles définissent des plans parallèles. Nous avons donc les deux plans de base parallèles. Il suffit donc de deux paramètres pour décrire l'orientation et la position de la seconde caméra par rapport à la première.

Soit la transformation $T^{gd}$ qui définit le mouvement du repère $R^g_{T0}$ vers le repère $R^d_{T0}$. Elle est composée d'une rotation d'angle $\delta$ autour de l'axe $Y^g_{T0}$ et d'une translation $t$ se-

lon l'axe $Y^g_{T0}$. Cette transformation définie la position de la caméra secondaire lorsque la caméra principale capture sa colonne initiale. Ainsi, la position initiale de la caméra secondaire est définie selon la position initiale de la caméra principale. Il faut aussi noter que ces deux positions ne sont pas forcément alignées, et dépendent de la valeur de l'angle $\delta$. Ainsi, pour tout ajout de caméra, il nous faut seulement deux paramètres supplémentaires plus les paramètres propres à la nouvelle caméra.

### 4.5.2 *Mise en équation*

Le modèle de la caméra principale $\mathcal{S}_g$ est le même que celui présenté dans la partie sur la modélisation simple. On a donc toujours la même matrice de transformation :

$$\begin{bmatrix} X^g_C \\ Y^g_C \\ Z^g_C \\ 1 \end{bmatrix} = (T^a_g)^{-1} \times (T^b_g)^{-1} \times (T^c)^{-1} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R^g_{11}.X + R^g_{12}.Y + R^g_{13}.Z + t^g_x \\ R^g_{21}.X + R^g_{22}.Y + R^g_{23}.Z + t^g_y \\ R^g_{31}.X + R^g_{32}.Y + R^g_{33}.Z + t^g_z \\ 1 \end{bmatrix} \tag{4.51}$$

Avec $(X, Y, Z)^T$ les coordonnées du point P d'après le repère $\mathcal{R}_B$ de la base du système, et $(X^g_C, Y^g_C, Z^g_C)^T$ les coordonnées du point P dans le repère $\mathcal{R}^g_C$ de la caméra principale. Et les équations des coordonnées dans l'image du point $P_B$ sont :

$$\begin{cases} u_g = \frac{W_g}{2.\pi}.\left(\arccos\left(\frac{C_g}{\sqrt{A_g^2+B_g^2}}\right) + \arctan\left(\frac{B_g}{A_g}\right)\right) \\ v_g = \alpha^g_v \left(\frac{R^g_{21}.X+R^g_{22}.Y+R^g_{23}.Z+t^g_y}{R^g_{31}.X+R^g_{32}.Y+R^g_{33}.Z+t^g_z}\right) + v^g_0 \end{cases} \tag{4.52}$$

Pour la caméra secondaire, il faut rajouter une transformation supplémentaire nommée $T^{gd}$ et qui contient une rotation élémentaire $\delta$ autour de l'axe $Y^g_{T0}$ et une translation élémentaire t en $Y^g_{T0}$ :

$$T^{gd} = \begin{bmatrix} \cos(\delta) & 0 & \sin(\delta) & 0 \\ 0 & 1 & 0 & t \\ -\sin(\delta) & 0 & \cos(\delta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.53}$$

Cette transformation décrit le mouvement du repère $R^g_{T0}$ au repère $R^d_{T0}$ qui est le repère de la plateforme tournante au niveau de la caméra secondaire lorsqu'elle intègre la colonne initiale de pixels.

La transformation qui permet de passer des coordonnées $(X, Y, Z)^T$ relatives au repère de la base $R_B$ aux coordonnées $(X^d_C, Y^d_C, Z^d_C)^T$ relatives au repère caméra secondaire $R^d_C$ est donc définie par l'équation suivante :

$$\begin{bmatrix} X^d_C \\ Y^d_C \\ Z^d_C \\ 1 \end{bmatrix} = (T^a_d)^{-1} \times (T^b_d)^{-1} \times (T^{gd})^{-1} \times (T^c)^{-1} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R^d_{11}.X + R^d_{12}.Y + R^d_{13}.Z + t^d_x \\ R^d_{21}.X + R^d_{22}.Y + R^d_{23}.Z + t^d_y \\ R^d_{31}.X + R^d_{32}.Y + R^d_{33}.Z + t^d_z \\ 1 \end{bmatrix} \tag{4.54}$$

On constatera que la transformations $T^c$ est identique pour les deux caméras, l'ajout d'une caméra ne nécessite que deux nouvelles transformations $T_d^a$, $T_d^b$ et $T^{gd}$.

Ce qui donne au final, un système d'équations de la même forme pour la caméra secondaire :

$$\begin{cases} u_d = \frac{W_d}{2.\pi}.\left(\arccos\left(\frac{C_d}{\sqrt{A_d^2+B_d^2}}\right)+\arctan\left(\frac{B_d}{A_d}\right)\right) \\ v_d = \alpha_v^d\left(\frac{R_{21}^d.X+R_{22}^d.Y+R_{23}^d.Z+t_y^d}{R_{31}^d.X+R_{32}^d.Y+R_{33}^d.Z+t_z^d}\right)+v_0^d \end{cases} \tag{4.55}$$

### 4.5.3 *Prise en compte des erreurs*

Les tableaux 4.8 et 4.9 résume les paramètres du système stéréoscopique qui a été présenté.

| Type de Paramètre | Nom | Fonction |
|---|---|---|
| Intrinsèques | $\alpha_v^g$ | le changement d'échelle entre les coordonnées métriques et les coordonnées en pixels. Il est égal au rapport de la distance focale sur la taille verticale d'un pixel. |
| | $v_0^g$ | la valeur de la ligne ou l'axe optique (L'axe $Z_C^g$ du repère caméra $R_C^g$) est sécant avec le plan image. |
| | $W_g$ | Le nombre de colonnes dans l'image panoramique |
| | $d_g$ | le rayon du cercle de base |
| | $\psi_g$, $\theta_g$ et $\phi_g$ | les trois angles décrivant l'orientation libre de l'axe optique du capteur par rapport au repère $\mathcal{R}_T^g$ |
| Extrinsèques | $\omega$, $\beta$ | Les deux angles qui décrivent l'inclinaison de l'axe de rotation par rapport au plan horizontal du repère monde $\mathcal{R}_B$. Ces deux paramètres peuvent aussi être considérés comme étant extrinsèques. |

TABLE 4.8: Les 9 paramètres du modèle simple (pour la caméra principale) se trouvent inchangés dans le cas de la modélisation stéréoscopique

Nous avons toujours une modélisation des erreurs décrites dans les parties précédentes et le modèle est toujours aussi général. En effet, en jouant sur les valeurs des paramètres, on peut avoir une configuration verticale idéale avec les plans optiques confondus, ou on peut aussi obtenir une configuration horizontale idéale avec les plans optiques parallèles

| Type de Paramètre | Nom | Fonction |
|---|---|---|
| Intrinsèques | $\alpha_v^d$ | le changement d'échelle entre les coordonnées métriques et les coordonnées en pixels. Il est égal au rapport de la distance focale sur la taille verticale d'un pixel. |
| | $v_0^d$ | la valeur de la ligne ou l'axe optique (L'axe $Z_C^d$ du repère caméra $R_C^d$) est sécant avec le plan image. |
| | $W_d$ | Le nombre de colonnes dans l'image panoramique |
| | $d_d$ | le rayon du cercle de base |
| | $\psi_d$, $\theta_d$ et $\phi_d$ | les trois angles décrivant l'orientation libre de l'axe optique du capteur par rapport au repère $\mathcal{R}_T^d$ |
| Extrinsèques | $\delta$ | la valeur de l'angle qui décrit la différence d'orientation des repères $R_t^g$ et $R_t^d$ l'un par rapport à l'autre. |
| | $t$ | la distance verticale (selon l'axe$Y_C^g$) entre les repères $R_t^g$ et $R_t^d$. |

TABLE 4.9: Les 9 paramètres supplémentaires dues à l'ajout de la seconde caméra au modèle simple.

et les plans de base confondus.
Le passage d'un système mono-caméra à un système stéréoscopique ajoute neuf paramètres supplémentaires. Le nombre de paramètre double mais la complexité n'augmente pas car les caméras peuvent être calibrées séparément.

### 4.5.4 *Triangulation*

A présent que le modèle stéréoscopique a été présenté, nous allons passer à son utilisation. Nous considérons ici que nous avons un système stéréoscopique calibré, et que les 18 paramètres sont connus et que la mise en correspondance 2D-3D a été effectuée. Dans une problématique de triangulation, nous voulons calculer les coordonnées réelles du point $P(X,Y,Z)^T$ en fonction du modèle et des coordonnées connues de ses projections $m_g$ et $m_d$ dans les deux images. En d'autres termes, les inconnues sont X, Y et Z. Il nous faut donc trois équations pour les calculer. Comme nous connaissons les coordonnées $u_g$ et $v_g$, nous connaissons aussi $\alpha_g = \frac{2.\pi.v_g}{W_g}$. Et donc, la première équation est inutile. Nous repartirons donc de l'équation de $X_C^g = 0$ qui est la relation qui nous a permis de connaître $u_g$ dans

la partie sur la modélisation simple. La coordonnée $X_C^g$ doit être nulle pour que le point P soit dans le plan $Y_B O Z_B$ qui est dans le champ de vision de la caméra linéaire :

$$X_C^g = R_{11}^g.X + R_{12}^g.Y + R_{13}^g.Z + t_x^g = 0 \tag{4.56}$$

Ici, tous les paramètres de la matrice de transformation sont connus, il n'est plus nécessaire d'isoler $\alpha_g$. De plus en développant et en factorisant l'expression de $v_g$ on obtient :

$$X.(R_{21}^g - R_{31}^g.V_g) + Y.(R_{22}^g - R_{32}^g.V_g) + Z.(R_{23}^g - R_{33}^g.V_g) = 0 \tag{4.57}$$

Avec $V_g = \frac{v_g - v_0^g}{\alpha_v^g}$ On a donc un couple d'équations pour la caméra principale :

$$\begin{cases} R_{11}^g.X + R_{12}^g.Y + R_{13}^g.Z + t_x^g = 0 \\ X.(R_{21}^g - R_{31}^g.V_g) + Y.(R_{22}^g - R_{32}^g.V_g) + Z.(R_{23}^g - R_{33}^g.V_g) = 0 \end{cases} \tag{4.58}$$

Pour la seconde caméra, c'est le même principe, nous obtenons donc deux équations supplémentaires :

$$\begin{cases} R_{11}^d.X + R_{12}^d.Y + R_{13}^d.Z + t_x^d = 0 \\ X.(R_{21}^d - R_{31}^d.V_d) + Y.(R_{22}^d - R_{32}^d.V_d) + Z.(R_{23}^d - R_{33}^d.V_d) = 0 \end{cases} \tag{4.59}$$

En bilan on a un système de quatre équations :

$$\begin{cases} R_{11}^g.X + R_{12}^g.Y + R_{13}^g.Z + t_x^g = 0 \\ X.(R_{21}^g - R_{31}^g.V_g) + Y.(R_{22}^g - R_{32}^g.V_g) + Z.(R_{23}^g - R_{33}^g.V_g) = 0 \\ R_{11}^d.X + R_{12}^d.Y + R_{13}^d.Z + t_x^d = 0 \\ X.(R_{21}^d - R_{31}^d.V_d) + Y.(R_{22}^d - R_{32}^d.V_d) + Z.(R_{23}^d - R_{33}^d.V_d) = 0 \end{cases} \tag{4.60}$$

Il suffit de trois équations pour résoudre le système, mais nous en avons quatre. Il y a donc deux moyens de procéder[1] :

- En utilisant les deux équations issues de $v_g$ et $v_d$ et une seule équation issue de $u_g$ ou $u_d$,
- En utilisant les deux équations issues de $u_g$ et $u_d$ et une seule équation issue de $v_g$ ou $v_d$.

Ce choix se fait en fonction de la géométrie du système, et tout particulièrement en fonction du choix des paramètres $\delta$ et t. En effet, si la géométrie du système est telle que les plans optiques des deux caméras sont parallèles, c'est à dire que le point P est projeté sur la même colonne d'image dans les deux images. Alors, les deux équations issues de $u_g$ ou $u_d$ sont en fait la même équation, et ne permettent pas de résoudre le système si elles sont

1 Il est également possible de calculer une solution optimale avec toutes les équations.

utilisées ensembles. Dans ce cas il faut utiliser la première possibilité.
D'un autre coté, si la géométrie du système est telle que les deux plans de base sont confondus et que les orientations des caméras sont telles que le point P soit projeté sur la même ligne dans les deux images, alors les deux équations en $v_g$ et $v_d$ sont similaires. Dans ce cas il faut utiliser la deuxième possibilité.
Ce choix doit se faire uniquement dans les cas de géométrie idéaux. Dans le cas général où on n'a jamais une géométrie parfaitement alignée, on peut utiliser les deux méthodes sans distinction. Dans notre cas, nous avons dit que notre objectif est de diminuer le facteur de temps entre la capture du même point par les deux caméras. Le système va donc tendre à avoir ses deux plans optiques confondus. Il sera donc plus judicieux d'utiliser la première méthode. Ainsi, voici le système de trois équation à résoudre :

$$\begin{cases} R^g_{11}.X + R^g_{12}.Y + R^g_{13}.Z + t^g_x = 0 \\ X.(R^g_{21} - R^g_{31}.V_g) + Y.(R^g_{22} - R^g_{32}.V_g) + Z.(R^g_{23} - R^g_{33}.V_g) = 0 \\ X.(R^d_{21} - R^d_{31}.V_d) + Y.(R^d_{22} - R^d_{32}.V_d) + Z.(R^d_{23} - R^d_{33}.V_d) = 0 \end{cases} \tag{4.61}$$

Dans un premier temps, nous posons :

$$A_1 = R^g_{11} \qquad A_2 = R^g_{12} \qquad A_3 = R^g_{13} \qquad A_4 = t^g_x \tag{4.62}$$

$$B_1 = (R^g_{21} - R^g_{31}.V_g) \qquad B_2 = (R^g_{22} - R^g_{32}.V_g) \qquad B_3 = (R^g_{23} - R^g_{33}.V_g) \tag{4.63}$$

$$C_1 = (R^d_{21} - R^d_{31}.V_d) \qquad C_2 = (R^d_{22} - R^d_{32}.V_d) \qquad C_3 = (R^d_{23} - R^d_{33}.V_d) \tag{4.64}$$

Le système d'équations peut donc être exprimé comme ceci :

$$\begin{cases} A_1.X + A_2.Y + A_3.Z + A_4 = 0 \\ B_1.X + B_2.Y + B_3.Z = 0 \\ C_1.X + C_2.Y + C_3.Z = 0 \end{cases} \tag{4.65}$$

Après isolation et résolution, on trouve :

$$X = \frac{A_4.(B_2.C_3 - C_2.B_3)}{B_1.C_2.A_3 - B_1.A_2.C_3 - C_2.A_1.B_3 - C_1.B_2.A_3 + A_2.C_1.B_3 + A_1.B_2.C_3} \tag{4.66}$$

$$Y = \frac{A_4.(B_1.C_3 - C_1.B_3)}{B_1.C_2.A_3 - B_1.A_2.C_3 - C_2.A_1.B_3 - C_1.B_2.A_3 + A_2.C_1.B_3 + A_1.B_2.C_3} \tag{4.67}$$

$$Z = \frac{A_4.(B_1.C_2 - C_1.B_2)}{B_1.C_2.A_3 - B_1.A_2.C_3 - C_2.A_1.B_3 - C_1.B_2.A_3 + A_2.C_1.B_3 + A_1.B_2.C_3} \tag{4.68}$$

En injectant les valeurs des variables $A_1$ à $C_3$, puis les valeurs $R_{11}$ à $R_{33}$ dans les équations ci-dessus[2], on obtient des équations fonction de $V_g$, $V_d$, $\cos(\alpha_g)$ et de $\sin(\alpha_g)$. Les autres paramètres sont des constantes issues des valeurs des paramètres des deux caméras et du système calibré.

Avec un système calibré on a juste à injecter les valeurs $V_g$, $V_d$, $\cos(\alpha_g)$ et de $\sin(\alpha_g)$ d'un couple de points appariés pour calculer les coordonnées 3D $(X, Y, Z)$ de ce point selon le repère de la base du système $\mathcal{R}_B$. Ainsi il est possible de faire de la triangulation avec un tel système cylindrique.

### 4.5.5 *Vers une modélisation multiple*

On a montré que l'ajout d'une nouvelle caméra peut se faire sans remettre en question les paramètres de calibrage et la modélisation de la première. L'ajout d'une nouvelle caméra fait apparaître 9 nouveaux paramètres supplémentaires, et une nouvelle étape dans la calibration. En effet, il est plus simple de calibrer les caméras séparément afin d'identifier leurs paramètres séparément afin de limiter la complexité de l'opération. Les caméras peuvent aussi être calibrées les unes par rapport aux autres. En conclusion, l'ajout d'une nouvelle caméra induit l'apparition de neuf nouveaux paramètres, une calibration de cette caméra seule, et une autre calibration pour chaque caméra déjà présente sur la plateforme selon les besoins de l'application envisagée.

Il est donc intéressant de multiplier les couples de caméras afin d'améliorer la vitesse d'acquisition panoramique.

2 L'expression entière des solutions est trop volumineuse pour être affichée sur une seule page.

## 4.6 CONCLUSION SUR L'ARCHITECTURE RETENUE

Dans ce chapitre, nous avons étudié quatre modèles géométriques différents pour les caméras linéaires rotatives. Leur étude et leur comparaison nous ont permis de réaliser un nouveau modèle général qui prend en compte plus de réalités physiques sans pour autant multiplier inutilement le nombre de paramètres. Ensuite, nous avons étudié la géométrie épipolaire adaptée à ces types de systèmes afin de déterminer la façon la plus efficace de faire de la stéréovision. Cela nous a permis de développer un modèle général complet et qui peut prendre en charge plusieurs caméras. En bilan, il a aussi été montré que :

- La stéréovision est plus efficace quand elle est verticale, cela minimise le $\Delta t$ entre les captures du même point par les deux caméras,
- Il faut que les plans optiques des caméras linéaires d'un couple stéréoscopique soient confondus pour que les courbes épipolaires soient des droites,
- Il faut avoir des angles $\psi$, $\theta$, et $\phi$ nuls pour simplifier la géométrie épipolaire et avoir des images cylindriques sans distorsions.

Le système panoramique réalisé présenté dans le chapitre 3 possède une architecture qui tends à valider ces trois points comme l'illustre la figure 4.12.

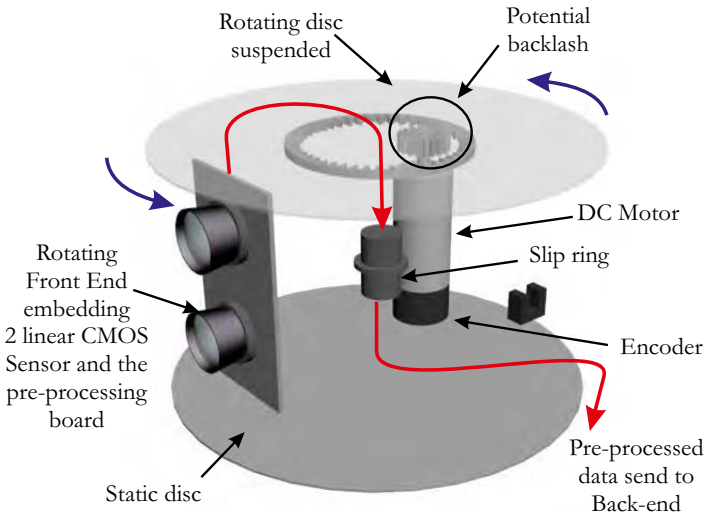


FIGURE 4.12: La configuration géométrique et mécanique du système avec un seul couple stéréoscopique.

Les propriétés du système réalisé tendent donc vers la description du cas idéal que nous avons décrit, et les écarts avec cette description sont pris en compte dans le modèle stéréoscopique que nous avons présenté dans ce chapitre.

# CHAPITRE 5

# Calibrage et Reconstruction 3D

## 5.1 INTRODUCTION

Dans le chapitre 4 nous avons étudié quatre modélisations afin de développer notre approche, et de proposer une modélisation stéréoscopique générale. Dans ce chapitre, nous abordons la suite logique de la modélisation en nous intéressant au calibrage.
Dans un premier temps, nous rappelons l'intérêt du calibrage et faisons une classification générale des différentes méthodes. Ensuite, nous présentons les méthodes de calibrage destinées aux systèmes panoramiques cylindriques proposées par les quatre groupes de travaux vus dans le chapitre 4 et les étudions pour aboutir à une comparaison. Nous analysons les avantages, les inconvénients et les problèmes liés à chaque méthode et débouchons sur notre proposition de méthode de calibrage. Enfin, nous terminons sur le sujet de la localisation et de la reconstruction 3D en présentant une expérimentation faite avec notre prototype.

## 5.2 MÉTHODES DE CALIBRATION CYLINDRIQUES

Le calibrage est une procédure utilisée en vision pour identifier les paramètres d'un modèle de caméra. Sans ces paramètres, il est impossible de faire de la photogrammétrie, ou encore de la reconstruction 3D par exemple. En général, il y a deux principaux groupes de méthodes de calibrage :

- La correspondance 2D/3D plus communément connue sous le nom *Spacial Resection*,
- L'ajustement de faisceaux ou *Bundle Adjustment*.

A partir d'un environnement dont la géométrie est connue (par exemple avec l'utilisation de mires géométriques), de la projection de cet environnement sur l'image et du modèle de la caméra, les méthodes de correspondance 2D/3D vont itérativement affiner l'estimation des paramètres du système. Il n'est pas nécessaire dans ce cas de prendre plusieurs images ni de faire des mouvements. Cela dit, il est difficile de connaitre la géométrie d'une scène de manière suffisamment précise.
Le deuxième groupe de méthodes évite ce problème. Pour un ensemble donné de points 3D obtenus à partir de différents points de vues, l'ajustement de faisceaux peut être défini comme le problème de raffinement simultané des coordonnées 3D décrivant la géométrie de la scène, des paramètres relatifs au déplacement entre les points de vues et des paramètres de la caméra selon son modèle. Ici, c'est le mouvement qui doit être connu afin de minimiser le nombre de paramètres à identifier. De plus, les appariements doivent être suffisamment fiables pour ne pas inclure des paires erronées.
Ces démarches sont aussi utilisables avec les systèmes cylindriques mais les méthodes classiques basées sur les modèles sténopés ne le sont pas. La différence majeure est que la projection des points n'est pas linéaire dans le cas des caméras cylindriques contrairement au cas sténopé. Tout comme les modélisations, les méthodes doivent donc être changées.

C'est pour cela que les travaux dans le domaine proposent de nouvelles méthodes plus ou moins différentes selon les modèles employés qui sont également non standardisés.
Dans cette section, nous étudions et présentons quatre méthodes publiées destinées aux systèmes cylindriques avant de déboucher sur la présentation de notre approche. Nous évaluons les différentes méthodes selon les modèles, la linéarité des algorithmes, la possibilité d'apparition de singularités et la complexité de la mise en œuvre.

### 5.2.1 *Calibration de Schneider et al.*

Dans leurs travaux, Schneider et al. proposent plusieurs méthodes de calibrage et exposent les résultats. Pour rappel du chapitre 4, voici le modèle tel qu'il est présenté en [118] :

$$u = u_0 - c.\arctan(\frac{-y}{x}) + \Delta_u \qquad v = v_0 - \frac{c.z}{\sqrt{x^2+y^2}} + \Delta_v \tag{5.1}$$

Avec :

$$\begin{aligned} x &= r_{11}.(X-X_0) + r_{21}.(Y-Y_0) + r_{31}.(Z-Z_0) \\ y &= r_{12}.(X-X_0) + r_{22}.(Y-Y_0) + r_{32}.(Z-Z_0) \\ z &= r_{13}.(X-X_0) + r_{23}.(Y-Y_0) + r_{33}.(Z-Z_0) \end{aligned} \tag{5.2}$$

Les paramètres du modèle sont alors $(u_0, v_0, c)$ pour les paramètres intrinsèques, $(X_0, Y_0, Z_0)$ et $(\omega, \psi, \kappa)$ pour les extrinsèques. Enfin, $\Delta_u$ et $\Delta_u$ sont utilisés pour modéliser les erreurs [114].
Deux méthodes de calibration sont proposées mais sans être décrites. La première est une méthode de correspondance 2D/3D où une salle de calibration est nécessaire. Des cibles réfléchissantes sont installées à 360 degrés autour du prototype et leurs coordonnées sont calculées avec une autre caméra. A partir de cette vérité terrain, du modèle et d'une capture de la salle par la caméra panoramique, un logiciel qui n'est pas décrit, calibre le système et estime les valeurs des paramètres. Une série de calibrages est effectuée. Lors du premier calibre, les paramètres $\Delta_u$ et $\Delta_u$ sont fixés à zéro, les résultats de calibration présentent une erreur de 25.20 pixels. Ensuite, différents paramètres sont ajoutés au système et la calibration est réinitialisée. On constate qu'on atteint une erreur inférieure à 1 pixel lorsque les quatre paramètres suivants sont inclus dans le modèle :

- L'orientation extérieure ,
- L'orientation intérieure (3 paramètres),
- L'excentricité du centre de projection (1 paramètre),
- Le non parallélisme du capteur (2 paramètres).

Cela montre à quel point ces paramètres sont nécessaires pour la précision des systèmes cylindriques et qu'ils doivent être pris en compte dans la modélisation.
La deuxième méthode présentée est un ajustement de faisceaux dans un environnement balisé de mires réfléchissantes [115]. Cette fois les coordonnées ne sont pas connues et la caméra est déplacée pour faire des captures à différents points de vues. La façon dont les mires sont détectées n'est pas expliquée, et l'algorithme de calibrage n'est pas présenté. Le problème des singularités est alors souligné lors de cette expérience. En effet, lors d'une translation horizontale pure entre deux points de vue, il y a une singularité et le calibrage échoue. Un déplacement vertical est proposé pour l'éviter. Cependant, il y a aussi des risques d'apparition de singularités avec moins de trois points de vues.
En bilan de ces méthodes, le tableau 5.1 résume leurs caractéristiques. Nous avons choisi la version du modèle enrichie des composants essentiels décrits plus haut.

Cette calibration n'étant pas décrite, il est difficile de prendre une position. Ce qu'il est possible de dire est que l'utilisation de mire 3D à 360 degrés est trop compliqué à mettre en œuvre. En ce qui concerne l'ajustement de faisceaux proposé, le problème des singularités est abordé. Le modèle ne prend pas en compte le cas avec plusieurs centres de projection (le cas le plus réaliste) et il manque un angle pour décrire l'orientation du capteur par rapport au mouvement.

### 5.2.2 *Calibration de Smadja et al.*

Le système présenté par Smadja et ali. [127, 129] produit des images couleurs de $2048 \times 3600$ pixels. On a vu la modélisation associée dans le chapitre 4 et il s'agit d'une projection cylindrique à un seul point de vue. Les paramètres sont au nombre de 9 : 6 extrinsèques $(t_x, t_y, t_z, \epsilon, \theta, \phi)$ et 3 intrinsèques $(K_v, f, v_0)$.
Les 6 paramètres extrinsèques décrivent la transformation qui permet de passer du repère

| **Méthode** | **Schneider et ali.** | **Schneider et ali.** |
|---|---|---|
| **Type de calibrage** | *Correspondance 2D/3D* | *Ajustement de faisceaux* |
| paramètres extrinsèques | 6 | 6 |
| paramètres intrinsèques | 6 | 6 |
| Orientation capteur | 2 | 2 |
| Projection | Centrale | Centrale |
| Utilisation de mire | complexe : mire à 360 degrés | Balisage du terrain |
| Primitives | points réfléchissants | points réfléchissants |
| Linéaire | non | non |
| Singularités | non | oui |

TABLE 5.1: Récapitulation des caractéristiques des deux méthodes proposées par Schneider et ali.

monde $\mathcal{R}_W$ au repère caméra $\mathcal{R}_C$ et ne sont pas inclus dans le modèle afin de préserver sa linéarité dans une premier temps. On a ainsi le système suivant à calibrer :

$$\begin{cases} u = K_u.\theta = K_u.\arctan(\frac{Y_C}{X_C}) \\ v = v_0 - K_v.f.z = v_0 - K_v.f.\frac{Z_c}{R} \end{cases} \tag{5.3}$$

Le paramètre $K_u$ correspondant à l'inverse de la taille vertical d'un pixel, il est utilisé en tant que paramètre connu. On se retrouve donc avec deux paramètres intrinsèques : $K_v.f$ et $v_0$.

L'utilisation de lignes pour la calibration a deux avantages. Le premier est qu'elle permet une linéarisation du problème car la projection des points n'est pas linéaire. Le deuxième est que les singularités sont évitées. Considérons un segment L de l'espace, qui est projeté sur le cylindre. L'équation de sa projection dans l'image n'est autre que l'intersection du plan N contenant le segment L et le centre de projection du système, avec le cylindre de l'image.

Soit $N_1.X_C + N_2.Y_C + N_3.Z_C = 0$ l'équation du plan $N_C$. Cette équation peut être exprimée comme suit :

$$\frac{Z_C}{R} = -\frac{N_1}{N_3}.\frac{X_C}{R} - \frac{N_2}{N_3}.\frac{Y_C}{R} \tag{5.4}$$

Et en combinant les deux dernières équations on obtient :

$$\begin{cases} u = K_u.\theta \\ v = v_0 + K_v.f.\frac{N_1}{N_3}.\cos(\frac{u}{K_u}) + K_v.f.\frac{N_2}{N_3}.\sin(\frac{u}{K_u}) \end{cases} \tag{5.5}$$

Cette équation décrit la projection du segment L dans l'image à partir du repère caméra $\mathcal{R}_C$, c'est à dire sans tenir compte des six paramètres extrinsèques qui décrivent le passage de $\mathcal{R}_W$ à $\mathcal{R}_C$.

Afin de compléter la modélisation et d'inclure ces six paramètres extrinsèques, la formalisation de *Plucker* est utilisée [123].

Cette formalisation permet une représentation simplifiée de lignes 3D en les décrivant par deux vecteurs. Le vecteur $m$ qui définit la direction de la ligne dans $\mathbb{R}^3$, et le vecteur $n$ qui définit sa position par rapport à l'origine du repère $\mathbb{R}^3$. Ce couple de vecteurs définit chaque ligne de manière unique : $L = [m, n]^T$ tel que $n.m = 0$. on a donc une matrice $3 \times 2$ comportant six valeurs pour chaque ligne de l'espace $\mathbb{R}^3$. Cette expression permet de noter les transformations de manière simplifiée. Dans notre cas, l'expression du plan $N_C$ peut être exprimée de cette manière :

$$N_C = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = [-RT_X|R] \begin{bmatrix} m \\ n \end{bmatrix} \tag{5.6}$$

Ou $T_X$ est la matrice antisymétrique qui correspond au produit de la matrice de translation $t$, et $\begin{bmatrix} m \\ n \end{bmatrix}$ la formalisation de *Plucker* pour le segment $L$ dans le repère monde $\mathcal{R}_W$. Et sachant qu'une ligne projetée sur le cylindre a pour équation dans l'image :

$$v = v_0 + A.\cos(\frac{u}{K_u}) + B.\sin(\frac{u}{K_u}) \tag{5.7}$$

Avec $A = K_v.f.\frac{N_1}{N_3}$ et $B = K_v.f.\frac{N_2}{N_3}$. On a donc l'expression :

$$\begin{bmatrix} A \\ B \\ 1 \end{bmatrix} = K_v.f.[-RT_X|R] \begin{bmatrix} m \\ n \end{bmatrix} \tag{5.8}$$

Cette dernière relation permet, d'obtenir pour chaque ligne détectée dans l'image les relations suivantes :

$$\begin{cases} A = K_v.f.\frac{N_1}{N_3} \\ B = K_v.f.\frac{N_2}{N_3} \end{cases} \tag{5.9}$$

Avec $N = [N_1, N_2, N_3] = -[RT_x|R]L$. En d'autres termes, chaque ligne détectée qui a ses valeurs $L = [m, n]^T_W$ connues dans $\mathcal{R}_W$, donne deux équations.

Il faut donc détecter quatre lignes pour avoir suffisamment d'équations pour la calibration. En effet, avec cette méthode, seuls les 6 paramètres extrinsèques et les deux paramètres intrinsèques $K_v.f$ et $v_0$ peuvent être identifiés.
La figure 5.1 résume les différentes étapes de la procédure de calibrage proposée.

Le calibrage est fait avec une mire comportant deux plans perpendiculaires sur lesquelles ont été tracées des lignes horizontales. La géométrie de la mire est connue et les lignes sont exprimées en fonction du repère monde. Après la capture d'une image panoramique de la mire, une image de contours est générée afin d'isoler les lignes du panorama. La double transformée de *Hough* [47] est ensuite calculée pour chaque point $(u, v)$ de

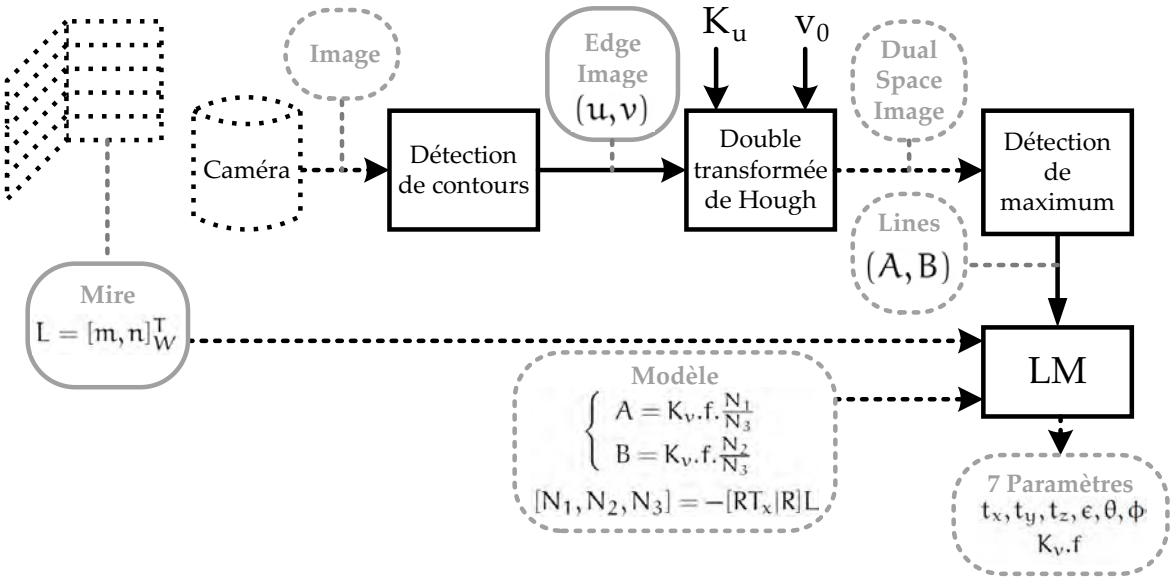


FIGURE 5.1: Synoptique de l'algorithme de calibration proposé par Smadja et ali.

l'image de contours, et une image *Dual Space* est générée. Dans l'image *Dual Space*, chaque maximum correspond à une sinusoïde respectant l'équation 5.7 de l'image de contours. La détection de maximums donne donc deux paramètres $(A, B)$ correspondants à une ligne dans l'image originelle. Enfin, l'algorithme de LM calcule les sept paramètres du système à partir des points $(A, B)$ respectant l'équation 5.8 et du modèle de projection des lignes de l'équation 5.5.
Le paramètre $v_0$ est calculé séparément avec deux méthodes possibles utilisant l'image *Dual Space*. Ces traitements sont itératifs et linéaires également. Nous avons donc un algorithme complètement linéaire. Le tableau 5.2 fait un résumé des caractéristiques de cette méthode.

| **Méthode** | **Smadja et ali.** |
|---|---|
| paramètres extrinsèques | 6 |
| paramètres intrinsèques | 2 |
| Orientation capteur | 0 |
| Projection | Centrale |
| Type de calibrage | *Spacial Resection* |
| Utilisation de mire | simple |
| Primitives | lignes |
| Linéaire | oui |
| Singularités | non |

TABLE 5.2: Récapitulation des caractéristiques de la méthode proposée par Smadja et ali.

En bilan nous avons ici une méthode complètement linéarisée grâce au concours de deux facteurs :

- L'utilisation d'un modèle simplifié (centre de projection unique et pas de modélisation de l'orientation du capteur),
- L'utilisation de primitives de type lignes.

De plus, l'emploi de primitives de type lignes évite les problèmes liés aux singularités de la projection des points.

### 5.2.3 *Calibration de Parian et al.*

La méthode de calibration de Parian et ali. [88, 89] utilise le modèle précédemment présenté dans le chapitre 4. L'équation est la suivante :

$$\begin{bmatrix} 0 \\ y \\ -c \end{bmatrix} = \lambda . P^{-1} . R_z^t . (j . A_h) . M_{w,\phi,k} . \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix} \tag{5.10}$$

Avec :

$$P = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \text{et} \qquad y = (i - \frac{N}{2}) . A_v \tag{5.11}$$

Nous avons six paramètres extrinsèques pour définir l'orientation extérieure : $(X0, Y0, Z0)$ et $(w, \phi, k)$. Le modèle de base ne comporte qu'un seul paramètre intrinsèque (l'orientation intérieure) mais est enrichi graduellement par des paramètres additionnels qui sont les suivants :

- La constante de la caméra, le décalage du point principal, et la distorsion radiale de l'optique (4 paramètres),
- L'excentricité du centre de projection (1 paramètre),
- L'orientation du capteur linéaire (3 paramètres),
- Les imprécisions du mouvement mécanique (6 paramètres).

Ce qui fait un total de 15 paramètres intrinsèques à calculer. La procédure de calibrage utilise un environnement balisé de 96 cibles circulaires. Les coordonnées cartésiennes des balises sont connues grâce à l'emploi d'un théodolite. Un ajustement de faisceaux est effectué avec quatre positions successives de la caméra. Les coordonnées des balises sont utilisées pour vérifier l'exactitude des résultats. En ajoutant progressivement les groupes de paramètres au modèle, on obtient une précision de plus en plus importante pour atteindre 0.65 pixels avec l'ensemble des paramètres.

Cela dit, l'équation avec tous les paramètres n'est pas linéaire et est très instable. C'est pourquoi la méthode de calibrage est améliorée en [90] avec l'utilisation de lignes 3D pour augmenter la stabilité. En effet, comme dans les travaux de Smadja et ali, il est montré que l'utilisation de lignes en tant que primitives de calibrage augmente la stabilité du système et évite les singularités. Les images sont alors traitées par un détecteur de bords de Canny. Les cotés détectés sont filtrés pour ne laisser que les détections de lignes droites et les points résultants sont utilisés dans la procédure d'ajustement de faisceaux précédente. La procédure de calibrage et les algorithmes employés ne sont pas précisés dans ces travaux. Le tableau 5.3 résume les caractéristiques de cette méthode de calibrage :

| **Méthode** | **Parian et ali. [88, 89]** | **Parian et ali. [90]** |
|---|---|---|
| paramètres extrinsèques | 6 | 6 |
| paramètres intrinsèques | 15 | 15 |
| Orientation capteur | 3 | 3 |
| Projection | Centrale | Centrale |
| Type de calibrage | Ajustement de faisceaux | Ajustement de faisceaux |
| Utilisation de mire | balisage de terrain | non |
| Primitives | cibles circulaires | lignes 3D |
| Linéaire | non | non |
| Singularités | oui | non |

TABLE 5.3: Récapitulation des caractéristiques de la méthode proposée par Parian et ali.

En conclusion sur cette méthode, on peut souligner le même problème de stabilité et de singularité qu'avec Schneider et ali. Dans l'utilisation d'ajustement de faisceaux pour le calibrage, il y a un problème de singularité qui apparaît avec certains mouvements de la caméra. Une solution possible est d'utiliser des primitives de type ligne 3D.

### 5.2.4 *Calibration de Huang et al.*

Le modèle [111] a été présenté dans le chapitre 4. L'équation générale est la suivante :

$$P_w = t + R.R_{\phi(u)} \left[ \lambda.R_\xi.R_\omega. \left[ R_u. \begin{pmatrix} \Delta_x - u_0 \\ u.\tau + \Delta_y - v_0 \\ \Delta_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ f \end{pmatrix} \right] + \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} \right] \tag{5.12}$$

On a six paramètres extrinsèques et 13 intrinsèques, comme résumé dans le tableau 4.4 du chapitre 4. L'approche utilisée est une correspondance 2D/3D dans un espace connu. Des balises représentant un motif automatiquement détectable sont placées tout autour de la caméra à des positions connues Ces positions $(X, Y, Z)$ correspondent au centre du motif détectable de chaque balise et ont été mesurées avec un théodolite. Plusieurs captures

de la scène sont utilisées. Le problème est ensuite linéarisé pour simplifier sa résolution et une méthode de Least Squares Estimate (LSE) est utilisée pour calculer les paramètres. Cette linéarisation rend le système très stable au calibrage et il est donc possible d'estimer les paramètres de manière itérative. Le tableau 5.4 rassemble les informations sur cette méthode.

| **Méthode** | **Huang et ali.** |
|---|---|
| paramètres extrinsèques | 6 |
| paramètres intrinsèques | 13 |
| Orientation capteur | 5 |
| Projection | Multi-centrale |
| Type de calibrage | Correspondance 2D/3D |
| Utilisation de mire | Complexe : environnement balisé |
| Primitives | points SIFT |
| Linéaire | oui |
| Singularités | non |

TABLE 5.4: Récapitulation des caractéristiques de la méthode proposée par Huang et ali.

L'utilisation d'un espace balisé dont la géométrie est mesurée rend possible l'expression du problème de façon linéaire avec une approche de correspondances 2D/3D. La modélisation est ici la plus complète de celles présentées et prend en compte l'orientation du capteur et les centres de projection multiples. Le problème de cette approche réside dans l'utilisation d'un espace mesuré avec des balises, il peut être considéré comme une mire de calibrage géante. L'emploi de ce genre de mire monumentale est très difficile avec les caméras panoramiques car il faut une mire qui soit omnidirectionnelle pour avoir un calibrage stable.

### 5.2.5 *Conclusion de l'analyse*

Nous avons étudié quatre méthodes de calibrage destinées aux systèmes linéaires rotatifs. Tous les résultats de cette étude sont résumés dans le tableau 5.5.

Les deux groupes de méthodes existantes pour la calibration (ajustement de faisceaux et correspondances 2D/3D) sont utilisables dans le cas des systèmes panoramiques. Les principaux problèmes qui apparaissent avec les systèmes rotatifs sont les singularités avec les méthodes d'ajustement de faisceaux, et la non-linéarité des équations à optimiser. Les travaux parcourus proposent diverses solutions pour régler ces difficultés. Par exemple, l'utilisation de primitives plus stables comme des lignes 3D ou le fait d'inclure une composante verticale dans les mouvements du système.
Du coté des correspondances 2D/3D, l'utilisation de mires monumentales rend la pratique difficile et complexe à mettre en œuvre, mais aussi présente l'avantage d'une plus grande stabilité et la possibilité de linéariser le problème.

| **Méthodes** | **Schneider et ali.** | **Smadja et ali.** | **Parian et ali.** | **Huang et ali.** |
|---|---|---|---|---|
| paramètres extrinsèques | 6 | 6 | 6 | 6 |
| paramètres intrinsèques | 6 | 2 | 15 | 13 |
| Orientation capteur | 2 | 0 | 3 | 5 |
| Projection | Centrale | Centrale | Centrale | Multi-Centrale |
| Type de calibrage | *Corresp./Faisceaux* | Corresp. 2D/3D | *Faisceaux* | Corresp. 2D/3D |
| Utilisation de mire | Complexe/non | simple | Complexe/non | Complexe |
| Primitives | points | lignes | points/lignes | points SIFT |
| Linéaire | non | oui | non | oui |
| Singularités | non/oui | non | oui/non | non |

TABLE 5.5: Comparaison des différentes méthodes de calibrage étudiées.

Dans cette thèse, nous recherchons une approche simple à mettre en œuvre et donc explorons les options de type ajustement de faisceaux. De plus, notre modélisation étant plus optimisée, elle comporte moins de paramètres à estimer et donc est plus simple à utiliser. Cette section commence par présenter l'approche proposée, ensuite nous rentrons dans les détails de l'algorithme.

## 5.3 APPROCHE DE CALIBRAGE PROPOSÉE

### 5.3.1 *Description générale*

Notre méthode est basée uniquement sur une approche de type ajustement de faisceaux, nous utilisons des correspondances 2D/2D entre des images capturées à des positions distinctes. La caméra est déplacée d'une position à l'autre en effectuant des translations pures (sans changement d'orientation) et les images sont capturées pendant que la caméra reste immobile à chaque position. Si quelques simples précautions sont prises, comme imposer à la caméra un mouvement rectiligne à l'aide d'un rail, cette méthode assure le calcul des valeurs numériques de tous les paramètres sans singularités. Cela est aussi rendu possible en ajoutant une inclinaison au plateau rotatif par rapport au plan horizontal. En effet, nous avons vu dans la section précédente qu'il est nécessaire d'avoir une composante verticale dans les mouvements pour éviter les singularités. Cinq positions successives forment la trajectoire en "L" présentée en figure 5.2 de manière à avoir deux fois trois positions alignées avec un angle droit entre les deux alignements.

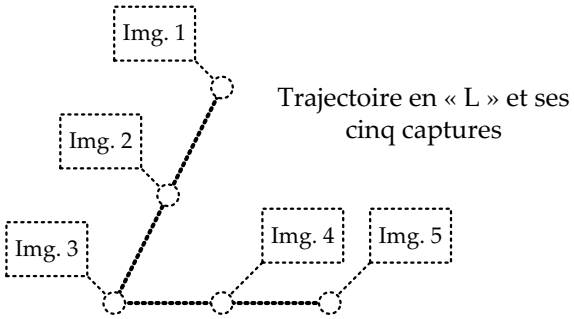


FIGURE 5.2: Schéma explicatif de la trajectoire utilisée dans notre méthode de calibrage.

Aucune mire de calibrage n'est nécessaire, et il n'y a pas besoin de connaître la géométrie du lieu utilisé lors de la procédure. Le minimum requis est de capturer trois images panoramiques complètes à trois positions différentes. Cependant nous utilisons cinq positions pour plus de fiabilité et le déplacement entre les positions est mesuré.
L'algorithme général utilisé est détaillé en figure 5.3.

Les cinq images capturées sont traitées par l'algorithme de détection de *Harris & Stephens* [45] puis un descripteur est associé à chaque détection.[1] Le descripteur consiste en une fenêtre de pixels décrivant le voisinage de chaque point et est utilisé dans l'étape d'appariement qui est une corrélation de type Zero-Mean Normalized Cross-Correlation (ZNCC) [31].
Les fausses paires peuvent éventuellement être supprimées manuellement à la fin de cette étape pour garantir une estimation des paramètres la plus fiable possible et aussi pour faciliter la convergence vers le bon minima de la fonction de coûts. A partir de là, le modèle présenté dans le chapitre 4 est utilisé avec les coordonnées images des paires de points pour mettre en place un système d'équations qui est injecté dans le bloc des moindres carrés. Ensuite, l'algorithme de *Levenberg-Marquardt* [77, 66] optimise de manière itérative le résultat en estimant les paramètres.

1 Le chapitre 3 aborde la question du choix de l'algorithme de détection.

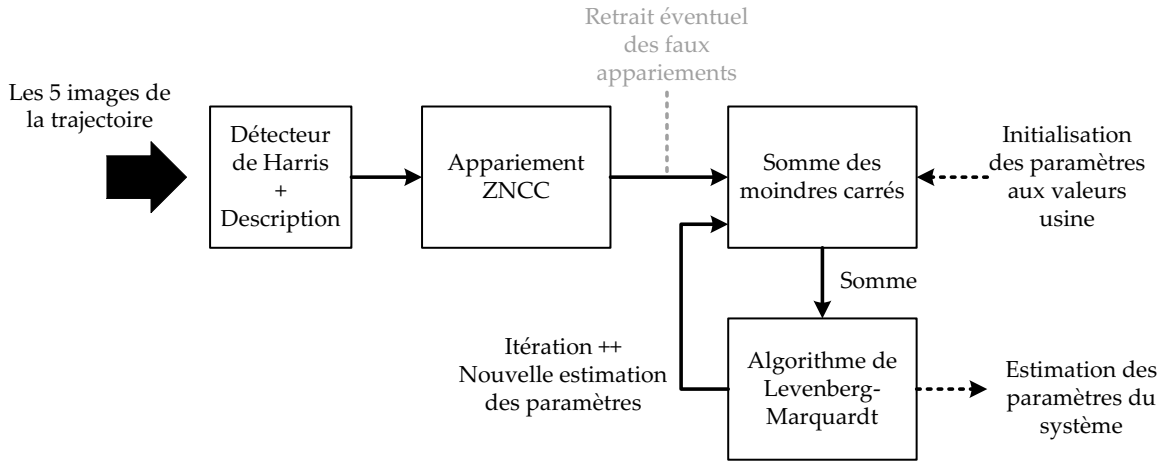


FIGURE 5.3: Synoptique de notre algorithme de calibrage.

### 5.3.2 *Calcul des paramètres*

Considérons un ensemble de cinq images panoramiques complètes capturées en cinq positions différentes comme indiqué plus haut. Les déplacements entre les cinq positions sont des translations pures, donc sans aucune composante de rotation. On note $m_i^g$ et $m_j^d$ les projections d'un point 3D $P_i$ sur deux des images panoramiques de la séquence ; $I_g$ et $I_d$ respectivement.

La translation de la caméra entre ces deux images $I_g$ et $I_d$ est notée $\delta_d^g$. Selon le modèle géométrique présenté dans le chapitre 4, le mouvement du repère de la caméra entre le moment où elle a capturé $m_i^g$ et celui où elle a capturé $m_j^d$ est :

$$R = \left[R^c.R_i^b.R^a\right]^T.R^c.R_j^b.R^a = \left[R_i^b.R^a\right]^T.R_j^b.R^a \tag{5.13}$$

$$t = -\left[R^c.R_i^b\right]^T.t^a + \left[R^c.R_i^b.R^a\right]^T.(\delta_d^g + R^c.R_j^b.R^a.t^a) \tag{5.14}$$

Avec $[*]^T$ qui est la transposée de la matrice $*$.

$R_i^b$ et $R_j^b$ sont les rotations de la plateforme rotative autour de l'axe Y pour que le plan formé par le capteur linéaire et l'axe optique de son objectif contienne le point $P_i$. En d'autres termes, il s'agit de l'orientation de la plateforme rotative pour laquelle le point $P_i$ est dans le champ de vision du capteur linéaire. Ainsi il est possible de calculer ces rotations en connaissant les composantes des points $m_i^g$ and $m_j^d$ avec la matrice :

$$R_i^b = \begin{bmatrix} \cos(u_i^g * \Delta\theta) & 0 & \sin(u_i^g * \Delta\theta) \\ 0 & 1 & 0 \\ -\sin(u_i^g * \Delta\theta) & 0 & \cos(u_i^g * \Delta\theta) \end{bmatrix} \tag{5.15}$$

Avec $\Delta\theta$ qui est le pas angulaire qui dépend du nombre de colonnes $W$ d'une image panoramique (6600 dans notre cas). Sa valeur est donnée par l'équation :

$$\Delta\theta = \frac{2\pi}{W} \tag{5.16}$$

Notons $E_{ij}^{gd}$ la matrice essentielle des deux colonnes qui contiennent $m_i^g$ et $m_j^d$, nous pouvons ensuite écrire :

$$E_{ij}^{gd} = [t]_X.R \tag{5.17}$$

Avec $[t]_X$ étant la matrice antisymétrique du vecteur t. Donc, la contrainte épipolaire de la caméra est écrite :

$${}^T[K^{-1}.m_i^g].E_{ij}^{gd}.K^{-1}.m_j^d = 0 \tag{5.18}$$

Chaque paire de points de correspondances donne une de ces équations où les inconnues sont les paramètres intrinsèques et extrinsèques de la caméra $\alpha_v$, $v_0$, $R^c$, $R^a$ et , $t^a$. Il faut un minimum de neuf équations pour procéder, mais en pratique bien plus sont utilisées. Le système d'équations ainsi obtenu permet de définir une fonction de coûts non linéaires. Les équations sont considérées comme une famille de fonctions indexées par les variables muettes $m_i^g$ et $m_j^d$, et indexées par l'ensemble des 9 paramètres que nous cherchons :

$$f(x_{ij};\theta) = {}^T[K^{-1}.m_i^g].E_{ij}^{gd}.K^{-1}.m_j^d = y_{ij} \tag{5.19}$$

Avec $x_{ij}$ comportant les coordonnées des points $m_i^g$ et $m_j^d$ et $\theta$ représentant les neuf paramètres à identifier.

$$S(\theta) = \sum (y_{ij} - f(x_{ij};\theta))^2 \tag{5.20}$$

Finalement, en utilisant les moindres carrés, le problème est réduit à la minimisation de l'équation 5.20 en utilisant l'algorithme de *Levenberg-Marquardt* [66, 77].

### 5.3.3 *Estimation des paramètres Usine*

L'algorithme de calibrage a besoin de valeurs initiales pour estimer les paramètres finaux. Plus les paramètres initiaux sont proches de la réalité, plus l'algorithme aura de chances de converger vers la bonne solution. Il est donc primordial de ne pas mettre de coté cette étape. Nous appelons *paramètres usine* les paramètres que le système tend à avoir selon les fichiers de conception. Par exemple, les dessins techniques du plateau rotatif sont faits de manière à avoir les angles de l'orientation du capteur tels que $R^a(\psi,\theta,\phi) = (0,0,0)$. Bien qu'en pratique il y a souvent une légère déviation due à l'imperfection de la fabrication,

ces paramètres servent pour l'initialisation.
Les paramètres $W$ et $v_0$ dépendent de la résolution de l'image. L'image étant de $6600 \times 2048$ pixels, on a $W = 6600$ et $v_0 = 1024$. Le paramètre $\alpha_v$ correspond au rapport entre la distance focale et la taille verticale d'un pixel. L'optique employée comporte une focale de $12,5mm$ et la taille verticale d'un pixel est de 7 micro mètres. On a donc une estimation de $\alpha_v$ qui est la suivante : $\alpha_v = 1785.7$.
L'estimation des angles $\omega$ et $\beta$ est expliquée sur la figure 5.4.

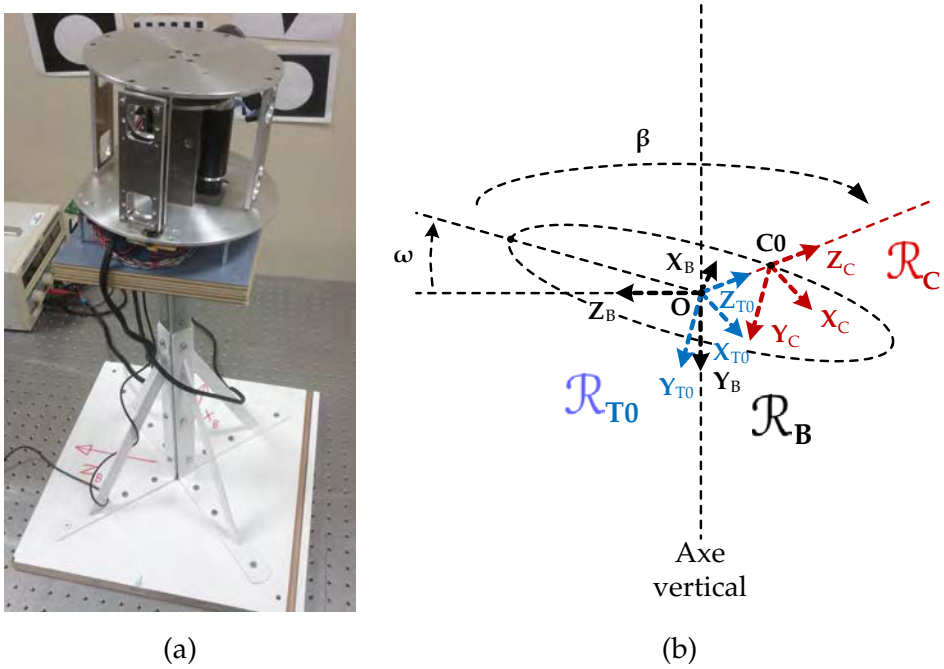


FIGURE 5.4: (a) Une photographie du système sur le socle. (b) Schéma de l'estimation des angles $\omega$ et $\beta$.

La caméra a été fixée sur un socle qui possède la même orientation que le repère $\mathcal{R}_B$. La valeur de $\omega$ dépend de l'inclinaison choisie par rapport à ce repère et a été mesurée à 7° sur la plateforme. La valeur de $\beta$ dépend de la position de la fourche optique qui définit l'orientation de la caméra lors de la capture de la première colonne de l'image. Cette valeur a été mesurée à 100°. Enfin, la distance d a été mesurée à $10cm$.
Le tableau 5.6 rassemble les neuf paramètres.

| Paramètres | $\alpha_v$ | $v_0$ | $R^a(\psi)$ | $R^a(\theta)$ | $R^a(\phi)$ | $W$ | $R^c(\omega)$ | $R^c(\beta)$ | $t^a(d)$ |
|---|---|---|---|---|---|---|---|---|---|
| Usine | 1785.7 | 1024 | 0 | 0 | 0 | 6600 | 100° | 7° | 0.1 m |

TABLE 5.6: Estimation des paramètres usine, qui sont utilisés pour l'initialisation de l'algorithme de calibrage.

### 5.3.4 *Conclusion sur le calibrage*

Cette méthode de calibrage n'a pas été complètement validée car la minimisation du système s'avère parfois instable. Cela est du en partie à l'estimation des paramètres initiaux

et à l'utilisation de primitives qui comportent parfois de faux appariements. L'idéal serai d'ajouter une linéarisation de l'estimation des paramètres initiaux ou d'utiliser des primitives plus spécifiques comme les points à l'infinie. A l'heure actuelle, notre méthode fonctionne de manière fiable sur des images de plus faible résolution, mais ne converge pas toujours pour la résolution de la caméra PanoraMOS. Bien que cette méthode ne soit pas encore validée complètement, on peut la comparer aux méthodes précédentes. Le tableau 5.7 présente les résultats de comparaisons.

| **Méthodes** | **Schneider et ali.** | **Smadja et ali.** | **Parian et ali.** | **Huang et ali.** | Notre approche |
|---|---|---|---|---|---|
| paramètres extr. | 6 | 6 | 6 | 6 | 2 |
| paramètres intr. | 6 | 2 | 15 | 13 | 7 |
| Orientation capteur | 2 | 0 | 3 | 5 | 3 |
| Projection | Centrale | Centrale | Centrale | Multiple | Multiple |
| Type de calibrage | *Corresp./Ajust.* | *Corresp.* | *Ajust.* | *Corresp.* | *Ajust.* |
| Utilisation de mire | Complexe/non | simple | Complexe/non | Complexe | non |
| Primitives | points | lignes | points/lignes | points SIFT | points *Harris* |
| Linéaire | non | oui | non | oui | non |
| Singularités | non/oui | non | oui/non | non | non |

TABLE 5.7: La comparaison finale de notre approche aux autres méthodes de calibrage étudiées. Les abréviations *Ajust.* et *Corresp.* correspondent respectivement *Ajustement de faisceaux* à et à *Correspondance* 2D/3D.

Notre méthode comporte une modélisation à la fois optimisée et générale, ce qui lui donne un nombre réduit de paramètres. Cependant, elle reste non linéaire et est parfois instable bien que les singularités aient été évitées. La méthode a été testée sur des données synthétiques et réelles mais les résultats ne sont pas présentés dans cette thèse car il reste beaucoup d'améliorations à lui apporter pour la rendre plus stable et robuste. Dans la suite de ce chapitre, nous utilisons donc les paramètres Usine.

## 5.4 LOCALISATION ET RECONSTRUCTION 3D

Une fois calibré, le modèle peut servir à localiser la caméra dans son environnement. Cette localisation peut se faire à partir des images capturées par la caméra à diverses positions. On peut ainsi estimer le mouvement entre deux de ces positions. Ensuite, à partir de cette estimation, il est possible de faire une reconstruction 3D de l'environnement et de construire une carte du milieu dans lequel évolue le système. Une technique de type SFM est alors utilisée. Cette approche générale que nous venons de décrire est appelée SLAM. Bien que l'étape de calibration ne soit pas encore complètement validée, il est possible d'utiliser une estimation des paramètres. Cette estimation est suffisante pour que l'algorithme de localisation converge sans difficultés, avec cependant un cout en précision.
Dans cette section, nous présentons l'algorithme de localisation, puis nous introduisons ensuite la SFM et exposerons notre approche. Enfin, les méthodes sont testées avec le prototype et les résultats sont présentés. La précision du système avec les paramètres usine est également mesurée afin de rendre compte des performances hors calibration.

### 5.4.1 *Estimation du mouvement entre deux images*

L'estimation du mouvement entre deux images se fait ici avec l'utilisation de l'algorithme de *Levenberg-Marquardt* [80] en association avec l'algorithme RANSAC [33]. La figure 5.5 décrit les différentes étapes de l'algorithme général.

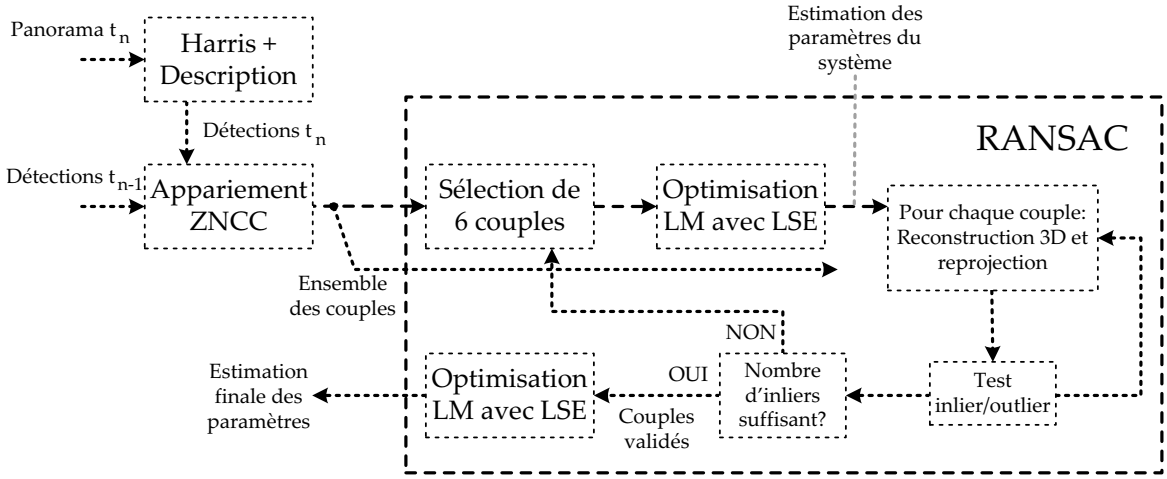


FIGURE 5.5: Synoptique de notre algorithme de localisation.

L'objectif est de calculer le vecteur de paramètres suivant :

$$P = [\phi, \theta, \psi, t_x, t_y, t_z] \tag{5.21}$$

Avec $R^m(\phi, \theta, \psi)$ et $t^m(t_x, t_y, t_z)$ représentant respectivement le mouvement de rotation et la translation ayant eu lieu entre les positions à $t_{n-1}$ et à $t_n$. Soient deux images $I_n$ et $I_{n-1}$ respectivement capturées par le système panoramique en ces deux positions. Le détecteur de *Harris & Stephens* commence par repérer les points d'intérêts et un descripteur leur est associé. La convolution ZNCC apparie les points et trouve un ensemble de couples de points. Ces couples de points peuvent comporter de faux appariements, l'utilisation de

RANSAC permet d'affiner la liste des couples en retirant les couples erronés.
Six des couples sont donc choisis aléatoirement et une optimisation employant l'algorithme de LM avec un LSE estime les paramètres du mouvement. Les autres couples sont ensuite testés avec ces paramètres. Chaque couple voit ses coordonnées 3D calculées puis est reprojeté dans l'image panoramique. Les coordonnées image obtenues sont comparées avec les coordonnées effectives et une erreur est calculée. Les couples obtenant une erreur inférieure à un seuil défini sont sélectionnés, les autres sont considérés comme des *outliers* ou données aberrantes. L'algorithme s'arrête lorsqu'un nombre suffisant d'*inliers* est trouvé, sinon, un nouvel échantillon de six points est sélectionné aléatoirement et le tout recommence.
Lorsqu'un nombre de couples suffisant respecte le modèle, un affinage des paramètres est effectué avec l'algorithme d'optimisation LM/LSE.
La matrice essentielle $E_{ij}^{gd}$ encapsule à la fois la rotation et la translation qui ont lieu entre deux captures d'images. La différence avec la géométrie épipolaire classique réside dans le fait que la matrice $E_{ij}^{gd}$ change pour chaque paire de points alors qu'elle reste constante dans le cas sténopé. Il faut donc exprimer $E_{ij}^{gd}$ en fonction des seuls paramètres restant constants, c'est à dire les internes et le mouvement.
Le mouvement entre les deux repères caméras aux positions nécessaires pour capturer les deux points d'un couple est défini par les relations suivantes :

$$R = [R^c.R_i^b.R^a]^T.R^m.R^c.R_j^b.R^a \tag{5.22}$$

$$t = -[R^c.R_i^b]^T.t^a + [R^c.R_i^b.R^a]^T.(t^m + R^m.R^c.R_j^b.R^a.t^a) \tag{5.23}$$

Pour chaque paire de points, nous obtenons une équation de la forme :

$$[K^{-1}.m_i^g]^T.E_{ij}^{gd}.K^{-1}.m_j^d = 0 \tag{5.24}$$

Cette fois les inconnues sont les paramètres des matrices $R^m$ et $t^m$, ce qui donne six inconnues. Donc il nous faut un minimum de six correspondances entre les deux images pour pouvoir retrouver le mouvement qui a eu lieu entre elles.

### 5.4.2 *Reconstruction 3D avec SFM*

La reconstruction 3D est un procédé qui vise à reconstruire la structure 3D de l'environnement à partir des mouvements de la caméra. La figure 5.6 résume ce principe.

Une fois que le mouvement de la caméra entre deux prises de vue a été calculé, chaque paire de correspondances $(m_i^g, m_j^d)$ est utilisée pour reconstruire les points $P_i$ 3D avec une triangulation des deux rayons optiques partant des deux centres optiques de la caméra aux deux positions ou elle a pu observer chaque point en question. Les points sont reconstruits dans le repère de la caméra, ensuite on peut écrire pour les points de la première image :

$$s.\begin{bmatrix} 0 \\ v_i^g \\ 1 \end{bmatrix} = K.\begin{bmatrix} I_{3\times3} & 0_{3\times3} \end{bmatrix}.\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = M.\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.25}$$

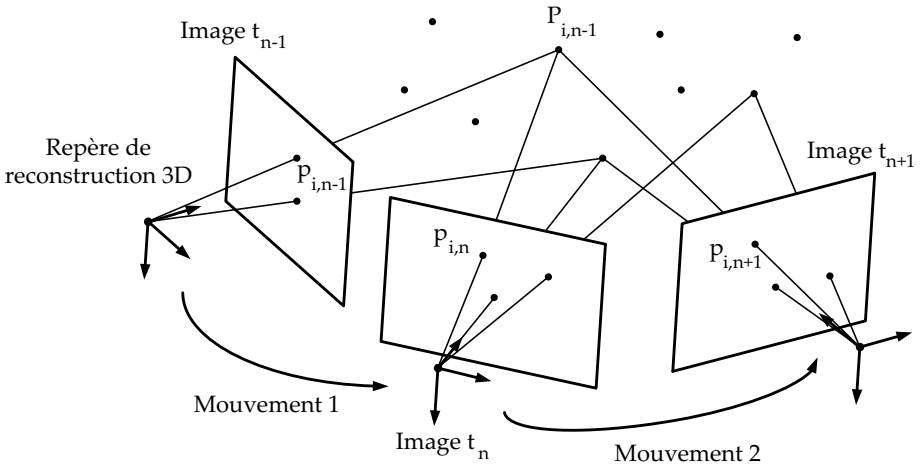


FIGURE 5.6: Illustration du principe de la reconstruction 3D avec la SFM

Avec $I_{3\times3}$ qui est la matrice identité $3 \times 3$, $0_{3\times3}$ une matrice nulle et $M^g$ la matrice $3 \times 4$ de projection. Nous pouvons le ré-écrire comme il suit :

$$\begin{cases} 0 = \frac{M_{11}.X+M_{12}.Y+M_{13}.Z+M_{14}}{M_{31}.X+M_{32}.Y+M_{33}.Z+M_{34}} \\ v_i^g = \frac{M_{21}.X+M_{22}.Y+M_{23}.Z+M_{24}}{M_{31}.X+M_{32}.Y+M_{33}.Z+M_{34}} \end{cases} \tag{5.26}$$

Et pour le second point on peut écrire :

$$s.\begin{bmatrix} 0 \\ v_i^d \\ 1 \end{bmatrix} = K.\begin{bmatrix} R & t \end{bmatrix}.\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = M'.\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5.27}$$

Enfin, cela donne :

$$\begin{cases} 0 = \frac{M'_{11}.X+M'_{12}.Y+M'_{13}.Z+M'_{14}}{M'_{31}.X+M'_{32}.Y+M'_{33}.Z+M'_{34}} \\ v_i^d = \frac{M'_{21}.X+M'_{22}.Y+M'_{23}.Z+M'_{24}}{M'_{31}.X+M'_{32}.Y+M'_{33}.Z+M'_{34}} \end{cases} \tag{5.28}$$

Un ensemble de quatre équations qui permettent de calculer P est obtenu sous la forme suivante :

$$A.P = b \tag{5.29}$$

La solution de ce système est donnée par la pseudo-inverse de $A$ :

$$P = ([K]^T.K)^{-1}.[K]^T.b \tag{5.30}$$

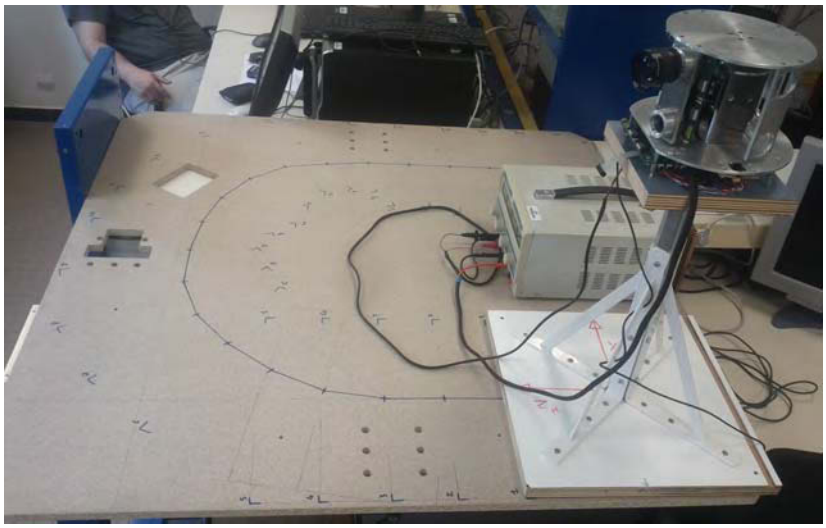

FIGURE 5.7: Photographie de l'environnement de l'expérimentation.

### 5.4.3 *Expérimentations et résultats*

Afin de tester les algorithmes de localisation et de reconstruction 3D présentés, un environnement de test a été mis au point comme celui présenté sur la figure 5.7.

La caméra *PanoraMOS*, qui est décrite dans le chapitre 3, est placée à diverses positions successives sur une trajectoire mesurée. La séquence comporte 21 positions équidistantes les unes par rapport aux autres de 10 cm. La caméra *PanoraMOS* n'était pas calibrée lors de cette expérimentation, les paramètres usine présentés dans la section précédente ont dont été utilisés directement. Cela dit, l'inclinaison de la plateforme rotative a été retirée car elle n'est utile que pour les séquences de calibrage, donc le paramètre $\beta$ a pour valeur 0.
L'algorithme a été exécuté en mode *Hors-Ligne* sur la séquence d'images avec *Matlab*. Le détecteur SIFT a été choisi pour plus de robustesse et un exemple d'appariement est montré sur la figure 5.8.

L'algorithme converge et donne les résultats présentés en figure 5.9.

La trajectoire calculée comporte une forme similaire à celle effectuée par la caméra et la structure de la scène est cohérente. Afin de mesurer la précision de la localisation, les données calculées sont comparées aux données réelles sur la figure 5.10.

On constate un écart entre les mesures et les positions calculées qui grandit au fur et à mesure de la trajectoire pour atteindre un décalage de 3,19 cm à la dernière position. Ce qui correspond à un écart de 4% d'erreur. Cette erreur est due à l'utilisation des paramètres usine à la place des paramètres réels qui doivent être obtenus avec une étape de calibrage. Cependant il est possible de parler de convergence car l'algorithme converge bel et bien vers le minimum permis par les paramètres estimés.

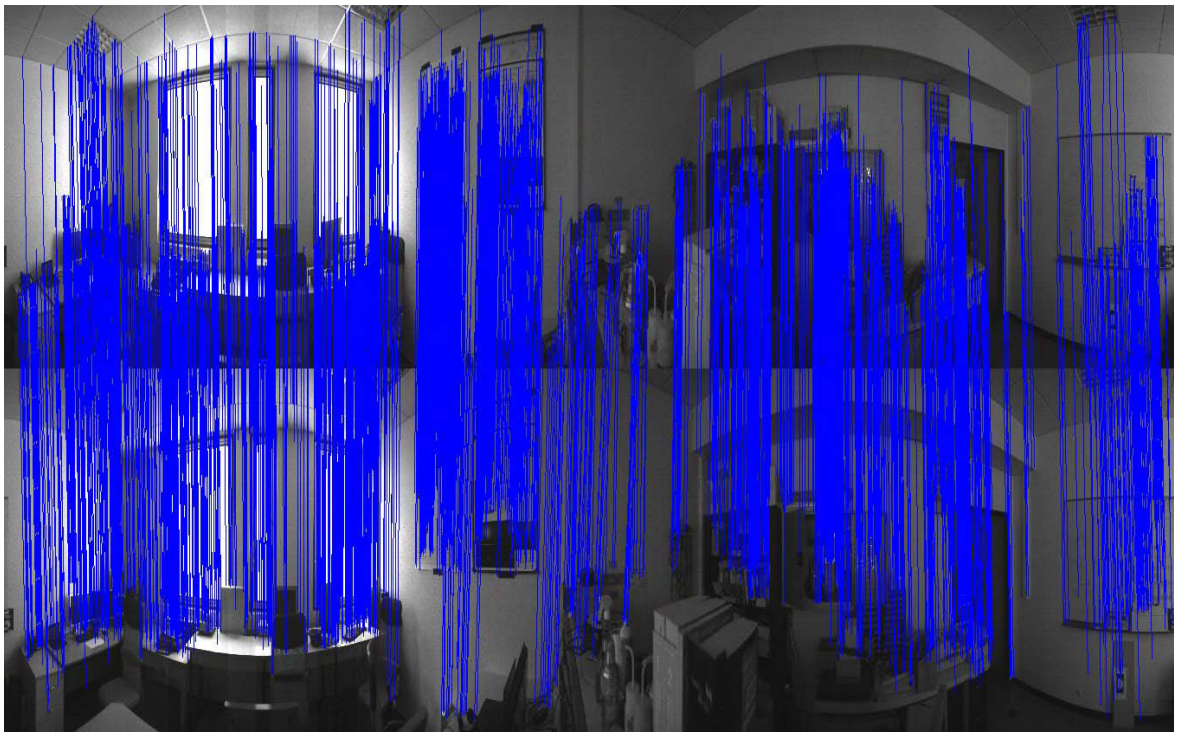

FIGURE 5.8: Couple d'image avec appariement des points détectés entre deux des images capturées pendant l'expérimentation.

## 5.5 CONCLUSION

Nous avons donc présenté une nouvelle méthode pour calibrer, localiser et reconstruire une carte 3D en utilisant un système d'acquisition panoramique à capteur linéaire rotatif. Contrairement aux méthodes existantes, notre algorithme de calibrage utilise un modèle géométrique complet qui ne souffre pas de configurations singulières. De plus, seuls les points de correspondances des images sont utilisés. Un algorithme qui estime le mouvement du capteur et qui construit une carte 3D de l'environnement a aussi été présenté et a été testé avec des données réelles. Les résultats obtenus montrent la faisabilité de l'approche et prouvent la précision des algorithmes utilisés lorsque l'on utilise des images panoramiques capturées par notre caméra *PanoraMOS*. Des résultats similaires ont été obtenus avec deux autres caméras panoramiques, la *Bica360* et le *OWL* du projet SEAMOVES.

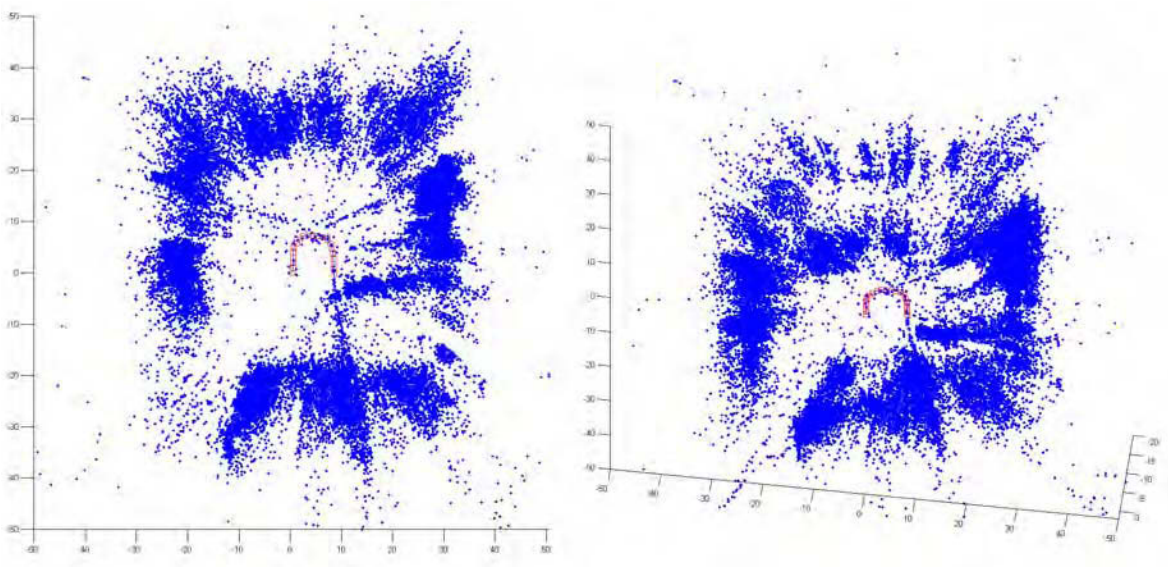

FIGURE 5.9: Résultats obtenus. En Rouge, la trajectoire de la caméra et en bleu, les points détectés reconstruits. Deux angles sont présentés pour rendre compte de la structure de la scène.

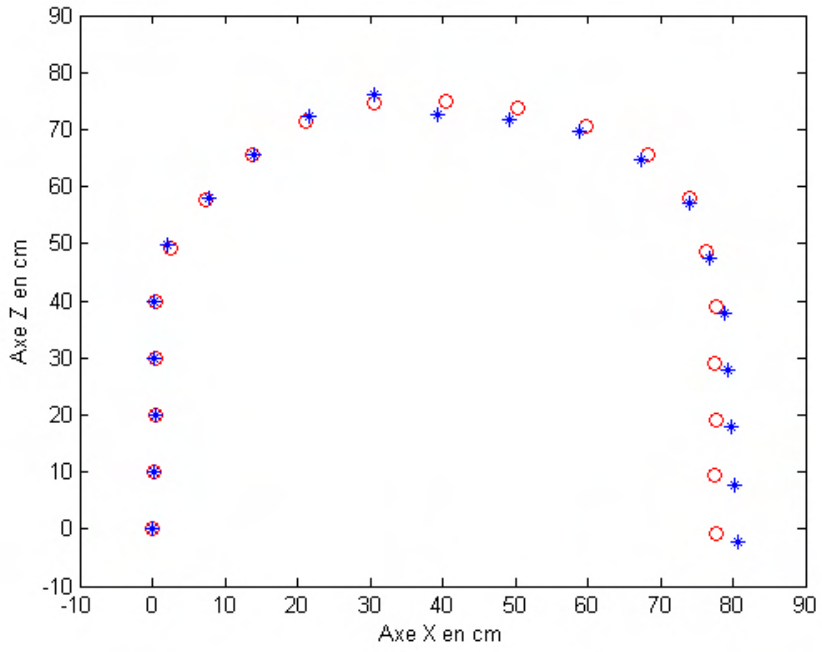


FIGURE 5.10: Comparaison entre la trajectoire calculée en bleu et la vérité terrain en rouge.

# CHAPITRE 6

# Conclusion

Au cours de ces travaux, nous nous sommes consacrés à la réalisation d'un système de stéréo-vision panoramique complet. Un algorithme d extraction de primitives a été adapté et directement implémenté dans l architecture de traitement sur le flot. Afin de satisfaire les objectifs d'utilisation de système proposé, une modélisation novatrice a été proposée, puis une méthode de calibrage et un algorithme de localisation ont été décrits. Ce chapitre résume ces différents topiques et conclut cette thèse. Les perspectives d'avenir et les voies de recherche débloquées sont ensuite présentées.

## 6.1 PRINCIPALES CONTRIBUTIONS

### 6.1.1 *Imagerie panoramique*

Dans le premier chapitre, l'état de l'art des méthodes panoramiques à été analysé. Les différentes technologies qui permettent de capturer des images panoramiques ont été étudiées, comparées et classifiées. Les avantages et les inconvénients de chaque méthode ont été mis en reliefs afin de montrer les intérêts que comporte l'utilisation de systèmes rotatifs. Il est possible d'en conclure les avantages suivants :

- Une plus grande résolution,
- La géométrie est de nature cylindrique,
- Aucun traitement d'assemblage ni de correction de la géométrie n'est nécessaire.

Cela dit, les inconvénients suivants sont aussi à prendre en compte :

- Une plus grande lenteur d'acquisition,
- L'apparition possible d'artéfacts à cause du *Rolling-Shutter* dû à l'acquisition progressive,
- La complexité de mise en œuvre.

Probablement à cause de ces défauts, les méthodes rotatives on relativement peu été étudiées. Il existe donc très peu de plateformes panoramiques rotatives, aucuns travaux sérieux sur la modélisation et le calibrage, et une absence totale de travaux sur l'utilisation de ces systèmes en temps réel. Pour ces raisons, cette thèse propose d'essayer de combler ces lacunes.
Dans ce contexte, nous avons montré qu'il est plus avantageux d'utiliser des caméras linéaires, en opposition aux caméras classiques, pour les raisons suivantes :

- Leur architecture est optimisée pour ce type d'application,

- La fréquence de capture est plus importante,
- Le facteur de remplissage peut atteindre les 100%,
- La consommation d'énergie est plus faible.

En conclusion de ce chapitre, les caméras linéaires rotatives sont un moyen encore trop peu exploré pour faire de la vision panoramique en temps réel, et il est intéressant de chercher à corriger leurs inconvénients afin de bénéficier de leurs avantages applicatifs.

### 6.1.2 *Architecture matérielle et logicielle*

Ce chapitre a présenté un système panoramique complet. Doté d'un couple de caméras linéaires, il est le premier capable de supporter des applications de type SFM en temps réel. Les propriétés et les innovations suivantes ont été décrites :

- Un système de capteurs pour connaitre la position relative et absolue de la caméra,
- Une architecture de capture optimisée avec des capteurs linéaires rapides,
- La synchronisation entre la mécanique et la capture, ce qui rends le calibrage possible,
- Une architecture électronique reconfigurable pour traiter le flot d'images directement dans la tête stéréoscopique,
- La possibilité de faire de la stéréo-vision,
- La possibilité de multiplier les têtes pour accélérer la capture.

La combinaison de ces technologies et de ces innovations font de PanoraMOS le premier système panoramique rotatif utilisable dans des applications de localisation en temps réel.

Un état de l'art des détecteurs de primitives est également proposé afin de choisir un détecteur adapté à nos besoins particuliers. Le détecteur de *Harris* étant sélectionné, son implémentation complète a été décrite et adaptée aux flot d'images cylindriques. L'implémentation dans la tête d'acquisition confère les avantages suivants :

- Une réduction significative de la quantité de traitement a effectuer dans le Back-End,
- Une réduction significative du flot de donnés a transmettre via le joint tournant (goulot d'étranglement du système).

Ainsi dans cette partie nous avons proposé un système intelligent complet d'acquisition qui débite un flot panoramique pré-traité et matériellement prêt pour son domaine applicatif.

### 6.1.3 *Modèle géométrique des systèmes cylindriques*

Une fois le système matériellement complet, il est nécessaire de décrire correctement sa géométrie pour pouvoir l utiliser avec precision et efficacité dans notre contexte applicatif. Ce chapitre a présenté les différents travaux qui existent sur le sujet, bien qu'ils ne soient pas nombreux, quatre sources différentes présentent leurs approches. Ces travaux comportent les défauts suivants :

- Des modélisations incomplètes :
  - Pas de prise en compte des défauts d'orientation du capteur,
  - Pas de prise en compte des imperfections du mouvement rotatif,
  - Pas de prise en compte de la modélisation de l'entre-axe.
- Une complexité non nécessairement importante,
- Pas de modélisation stéréoscopique.

Le modèle qui a été présenté dans ce chapitre corrige l'intégralité de ces problèmes sans pour autant introduire de complexité supplémentaire. Une étude de la géométrie épipolaire inhérente aux systèmes rotatifs a également été proposée.

### 6.1.4 *Calibrage et Reconstruction 3D*

Enfin, après la modélisation, les étapes de calibrage puis d'utilisation concrètes sont possibles. Dans ce chapitre, l'étude des travaux précédents est achevée avec la comparaison des méthodes de calibrages existantes. Ces méthodes, associées à des modèles géométriques incomplets, proposent des approches soit trop complexes, soit imprécises. En effet la moitié d'entre elles décrivent une calibration nécessitant une salle de calibrage balisée de mires optiques. Ces mires devant être positionnées de manière très précise, il est nécessaire de mesurer leur positions avec un autre dispositif de vision. L'autre partie des travaux étudiés ignorent le problème de singularité due au mouvement de translation décrit dans ce chapitre. Une nouvelle approche a donc été proposée avec une grande simplicité de mise en œvre et une méthode pour éviter l'apparition de la singularité.

Ensuite, une méthode et un algorithme de SFM spécialement adapté aux images cylindriques est présenté. Une expérimentation de localisation est enfin présentée et expérimentée. L'analyse des résultats a montré qu'il est possible de reconstruire la trajectoire mais aussi que la précision doit être améliorée. Les conclusion qu'il est possible de tirer est que ce type de système est tout à fait viable dans une application de type SFM en raison de quelques améliorations supplémentaires de l'algorithme de calibrage.

## 6.2 PERSPECTIVES

### 6.2.1 *Architecture matérielle*

Quelques travaux restent cependant inachevés et seront poursuivis après la rédaction de ce manuscrit. En effet, le mode de fonctionnement avec huit caméras (quatre couples stéréoscopiques) n'a pas encore été testé. C'est dans ce scénario que tout l'intérêt de l'extraction de primitives prendra son sens car avec quatre caméras ou plus, les flots possibles dans le *Slip-Ring* et avec la communication USB ne sont plus suffisants.

### 6.2.2 *Calibrage*

Beaucoup de travaux sont aussi à finir au niveau du calibrage. Bien que la méthode fonctionne sur données synthétiques et sur certaines séquences réelles, il y a certains problèmes de stabilité à résoudre et il est nécessaire d'augmenter la robustesse générale de l'approche.

### 6.2.3 *Localisation et reconstruction 3D*

Quant à l'application de localisation, elle est déjà relativement stable et a montré de bons débuts de résultats en environnement urbain également. Il manque cependant le recalage 2D/3D et le fait de posséder les estimations des paramètres du modèle. Enfin, une linéarisation de l'étape d'initialisation est aussi nécessaire pour rendre le système plus précis et plus efficace.

# ANNEXE A

## Géométrie Epipolaire Cylindrique

### A.1 LE CAS *parallel-axis panoramas*

La configuration *Parallel-axis Panoramas* est obtenue lorsque l'on ajoute à la configuration précédente la contrainte que les deux axes de rotation soient parallèles. Cela est représenté en figure A.1.

Le fait que les deux axes de rotation soient parallèles requiert d'avoir $\delta_x = 0$ et $\delta_z = 0$. Ce qui donne un mouvement $T_{dg}$ qui est composé d'une rotation $R^{21}(0, \delta_y, 0)$ et d'une translation $t^{21}(t_x, t_y, t_z)$.

En injectant ces paramètres dans l'équation 4.47, on obtient une nouvelle équation de la même forme :

$$V_d = \frac{Y_C^d}{Z_C^d} = \frac{A'_{11}.\cos(2\alpha_d) + A'_{12}.\sin(2\alpha_d) + A'_{13}.\cos(\alpha_d) + A'_{14}.\sin(\alpha_d) + A'_{15}}{A'_{21}.\cos(2\alpha_d) + A'_{22}.\sin(2\alpha_d) + A'_{23}.\cos(\alpha_d) + A'_{24}.\sin(\alpha_d) + A'_{25}} \quad (A.1)$$

Avec $V_d = \frac{v_d - v_{0d}}{\alpha_{vd}}$ et $\alpha_d = \frac{2.\pi.u_d}{W_d}$. Et $A'_{11}$ à $A'_{25}$ des constantes qui sont fonction des paramètres des deux caméras et du mouvement $T_{dg}$.

Dans le cas B, l'équation devient la suivante :

$$V_d = \frac{\begin{array}{c}\sin(\beta).[d_g.\sin(\theta_d + \alpha_d - \alpha_g - \delta_y) - d_d.\sin(\theta_d) - t_x.\cos(\theta_d + \alpha_d) + t_z.\sin(\theta_d + \alpha_d)] \\ + \cos(\beta).[t_y.\sin(\theta_g + \alpha_g - \theta_d - \alpha_d + \delta_y)]\end{array}}{\cos(\beta).[d_g.\sin(\theta_g) - d_d.\sin(\theta_g + \alpha_g + \delta_y - \alpha_d) - t_x.\cos(\theta_g + \alpha_g + \delta_y) + t_z.\sin(\theta_g + \alpha_g + \delta_y)]} \quad (A.2)$$

Dans le cas C, l'équation devient la suivante :

$$V_d = \frac{\sin(\beta).[d_g.\sin(\alpha_d - \alpha_g - \delta_y) - t_x.\cos(\alpha_d) + t_z.\sin(\alpha_d)] + \cos(\beta).[t_y.\sin(\alpha_g - \alpha_d + \delta_y]}{\cos(\beta).[-d_d.\sin(\alpha_g + \delta_y - \alpha_d) - t_x.\cos(\alpha_g + \delta_y) + t_z.\sin(\alpha_g + \delta_y)]} \quad (A.3)$$

La simplification devient intéressante, mais on reste avec une équation de la même forme. De plus, il n'est pas possible d'avoir les axes de rotation de deux systèmes réellement parallèles en pratique. Enfin, le problème du décalage de temps potentiel entre les captures des mêmes points par les deux caméras rend la stéréovision et l'appariement plus difficile.

### A.2 LE CAS *levelled panoramas*

Le cas *Levelled Panoramas* est un cas particulier de parallélisme d'axe de rotation dans lequel, les deux plans principaux sont confondus. Pour rappel, le plan principal d'un système est le plan qui contient toutes les position successives du centre optique de la caméra lors de sa rotation. La figure A.2 illustre cette contrainte.

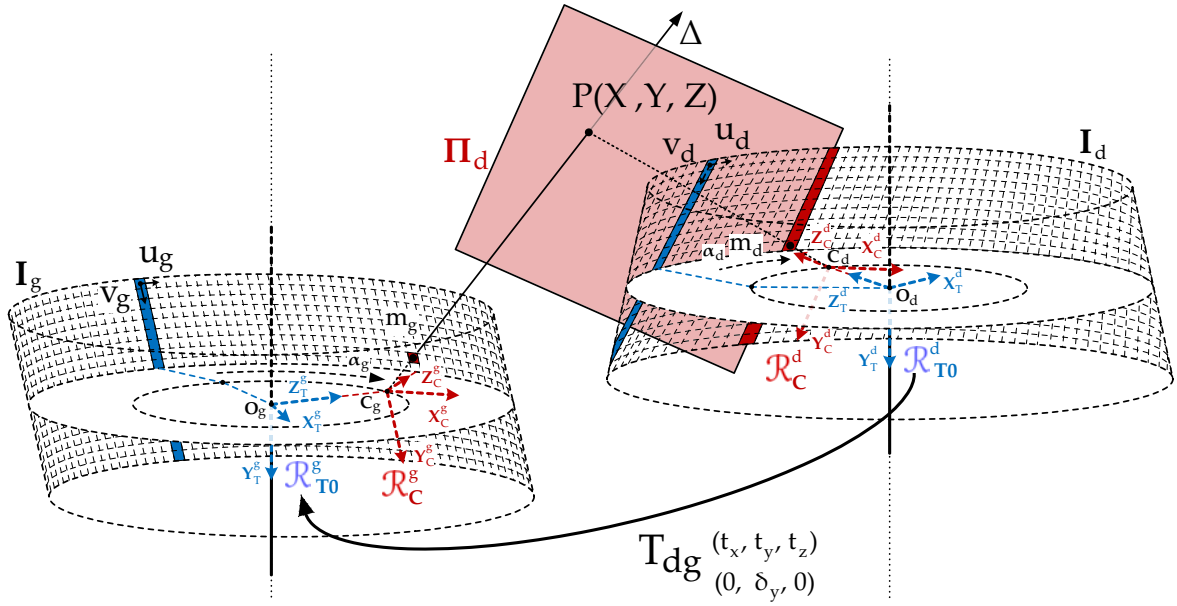


FIGURE A.1: Le cas *Parallel-axis Panoramas* est caractérisé par un mouvement $T_{dg}$ 3D avec 4 degrés de libertés.

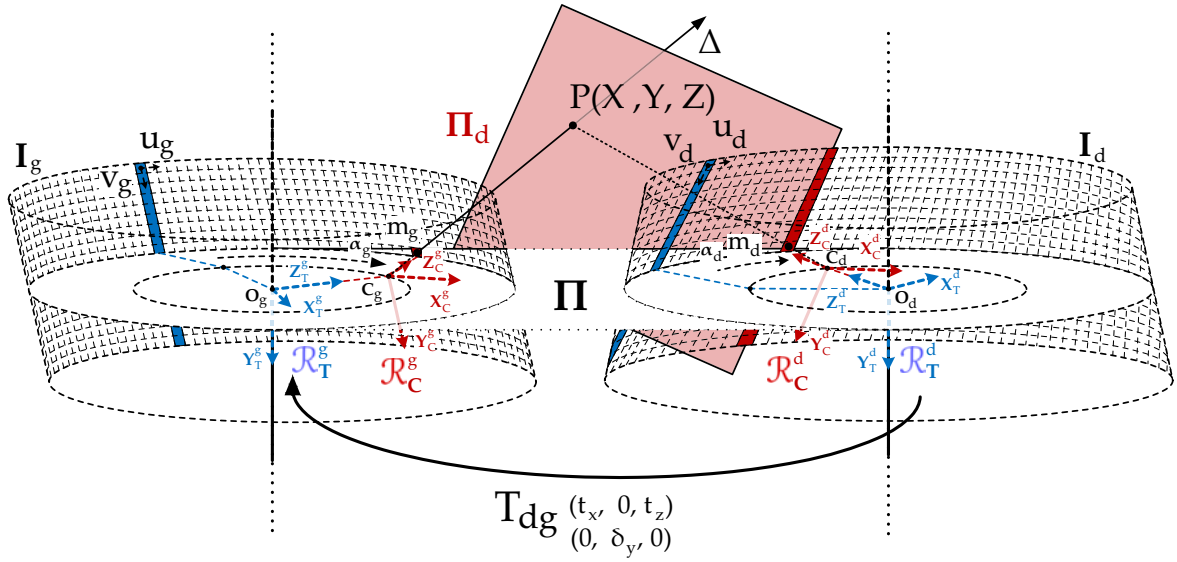


FIGURE A.2: Le cas *Levelled Panoramas* est caractérisé par un mouvement $T_{dg}$ 3D avec 3 degrés de libertés.

Pour que cela soit possible, il faut avoir une composante de translation $t_y = 0$ et toujours les deux angles $\delta_x = 0$ et $\delta_z = 0$. On a donc un mouvement $T_{dg}$ qui comporte une rotation $R^{21}(0, \delta_y, 0)$ et une translation $t^{21}(t_x, 0, t_z)$.

En injectant ces paramètres dans l'équation 4.47, on obtient la nouvelle équation :

$$V_d = \frac{Y_C^d}{Z_C^d} = \frac{A'_{11}.\cos(2\alpha_d) + A'_{12}.\sin(2\alpha_d) + A'_{13}.\cos(\alpha_d) + A'_{14}.\sin(\alpha_d) + A'_{15}}{A'_{21}.\cos(2\alpha_d) + A'_{22}.\sin(2\alpha_d) + A'_{23}.\cos(\alpha_d) + A'_{24}.\sin(\alpha_d) + A'_{25}} \quad (A.4)$$

Avec $V_d = \frac{v_d - v_{0d}}{\alpha_{vd}}$ et $\alpha_d = \frac{2.\pi.u_d}{W_d}$. Et $A'_{11}$ à $A'_{25}$ des constantes qui sont fonction des paramètres des deux caméras et du mouvement $T_{dg}$.

Dans le cas B, l'équation devient la suivante :

$$V_d = \frac{\sin(\beta).[d_g.\sin(\theta_d + \alpha_d - \alpha_g - \delta_y) - d_d.\sin(\theta_d) - t_x.\cos(\theta_d + \alpha_d) + t_z.\sin(\theta_d + \alpha_d)]}{\cos(\beta).[d_g.\sin(\theta_g) - d_d.\sin(\theta_g + \alpha_g + \delta_y - \alpha_d) - t_x.\cos(\theta_g + \alpha_g + \delta_y) + t_z.\sin(\theta_g + \alpha_g + \delta_y)]} \quad (A.5)$$

Dans le cas C, l'équation devient la suivante :

$$V_d = \frac{\sin(\beta).[d_g.\sin(\alpha_d - \alpha_g - \delta_y) - t_x.\cos(\alpha_d) + t_z.\sin(\alpha_d)]}{\cos(\beta).[-d_d.\sin(\alpha_g + \delta_y - \alpha_d) - t_x.\cos(\alpha_g + \delta_y) + t_z.\sin(\alpha_g + \delta_y)]} \quad (A.6)$$

Encore une fois nous avons un cas de positionnement impossible à réaliser en pratique pour les mêmes raisons. Le problème du décalage potentiel de temps $\Delta t$ est toujours présent.

## A.3 LE CAS *concentric panoramas*

La configuration concentrique est obtenue quand le centre de rotation des deux caméras est unique. En d'autres termes c'est quand les deux plans principaux sont confondus avec un axe de rotation unique. On a un seul degré de liberté : $R^{21}(0, \delta_y, 0)$ et $t^{21}(0, 0, 0)$.

En injectant ces paramètres dans l'équation 4.47, on obtient la nouvelle équation :

$$V_d = \frac{Y_C^d}{Z_C^d} = \frac{A'_{11}.\cos(2\alpha_d) + A'_{12}.\sin(2\alpha_d) + A'_{13}.\cos(\alpha_d) + A'_{14}.\sin(\alpha_d) + A'_{15}}{A'_{21}.\cos(2\alpha_d) + A'_{22}.\sin(2\alpha_d) + A'_{23}.\cos(\alpha_d) + A'_{24}.\sin(\alpha_d) + A'_{25}} \quad (A.7)$$

Avec $V_d = \frac{v_d - v_{0d}}{\alpha_{vd}}$ et $\alpha_d = \frac{2.\pi.u_d}{W_d}$. Et $A'_{11}$ à $A'_{25}$ des constantes qui sont fonction des paramètres des deux caméras et du mouvement $T_{dg}$. On constate qu'il n'y a toujours pas de simplification notable de la complexité de l'équation épipolaire malgré les contraintes sur le mouvement $T_{dg}$.

Dans le cas B, l'équation devient la suivante :

$$V_d = \frac{\sin(\beta).[d_g.\sin(\theta_d + \alpha_d - \alpha_g - \delta_y) - d_d.\sin(\theta_d)]}{\cos(\beta).[d_g.\sin(\theta_g) - d_d.\sin(\theta_g + \alpha_g + \delta_y - \alpha_d)]} \quad (A.8)$$

Sachant qu'il faut que $\theta_g$ soit différent de $\theta_d$.

Dans le cas C, nous avons les deux plans images qui sont confondus. On a donc $u_d = u_g$ et la courbe épipolaire est une droite verticale.

Pour le $\Delta t$, il dépendra des variables $\theta_g$, $\theta_g$, $\alpha_g$, $\delta_y$ et $\alpha_d$. Un autre cas remarquable est

celui ou on a $\theta_g = \frac{\pi}{2}$, $\theta_d = -\frac{\pi}{2}$, $\delta_y = \pi$ et $d_d = d_g = d$. A ce moment là nous avons une stéréovision de type horizontal contrairement à la stéréovision verticale du mode *Co-axis*. L'équation devient alors :

$$V_d = \frac{\sin(\beta).[d.\sin(\theta_d + \alpha_d - \alpha_g - \delta_y) - d]}{\cos(\beta).[-d - d.\sin(\theta_g + \alpha_g + \delta_y - \alpha_d)]} = V_g.\frac{1 - \sin(\theta_d + \alpha_d - \alpha_g - \delta_y)}{1 + \sin(\theta_g + \alpha_g + \delta_y - \alpha_d)} \quad \text{(A.9)}$$

Donc cela revient à avoir une courbe épipolaire droite et d'équation $V_d = V_g$. Cela dit, avec cette géométrie, un point de l'espace n'est pas capturé au même moment par les deux caméras, il y a donc le facteur $\Delta t$ qui entre en jeu. Cette configuration a donc deux défauts : elle découle d'un mouvement $T_{dg}$ non réalisable et le facteur $\Delta t$ entre les captures du même point est significatif.

## A.4 LE CAS *symmetric panoramas*

Enfin, la dernière configuration présente les mêmes contraintes que la précédente avec en plus la contrainte que les deux caméras aient les mêmes paramètres intrinsèques et qu'il n'y a aucune transformation entre les deux caméras. On a alors aucun degré de liberté pour le mouvement $T_{dg}$ $R^{21}(0,0,0)$ et $t^{21}(0,0,0)$. La stéréovision reste possible avec les angles $\theta_d$ et $\theta_g$ tels que $\theta_d = -\theta_g = \theta$.

En injectant ces paramètres dans l'équation 4.47, on obtient la nouvelle équation :

$$V_d = \frac{Y_C^d}{Z_C^d} = \frac{A'_{11}.\cos(2\alpha_d) + A'_{12}.\sin(2\alpha_d) + A'_{13}.\cos(\alpha_d) + A'_{14}.\sin(\alpha_d) + A'_{15}}{A'_{21}.\cos(2\alpha_d) + A'_{22}.\sin(2\alpha_d) + A'_{23}.\cos(\alpha_d) + A'_{24}.\sin(\alpha_d) + A'_{25}} \quad \text{(A.10)}$$

Avec $V_d = \frac{v_d - v_{0d}}{\alpha_{vd}}$ et $\alpha_d = \frac{2.\pi.u_d}{W_d}$. Et $A'_{11}$ à $A'_{25}$ des constantes qui sont fonction des paramètres des deux caméras et du mouvement $T_{dg}$.

Dans le cas B, l'équation devient la suivante :

$$V_d = \frac{\sin(\beta).[\sin(\theta + \alpha_d - \alpha_g) - \sin(\theta)]}{\cos(\beta).[\sin(\theta + \alpha_d - \alpha_g) - \sin(\theta)]} = V_g \quad \text{(A.11)}$$

Sachant qu'il faut que $\theta$ ne soit pas nul.
Dans le cas C $\theta$ est nul, les deux caméras sont alors exactement dans la même position, ce qui revient à avoir une seule caméra. Il n'y a donc aucune courbe épipolaire car la stéréovision n'est plus possible ici.

Cette dernière configuration n'est pas réaliste car deux caméras distinctes ne peuvent pas avoir le même centre optique à moins de faire deux captures en changeant l'orientation de la caméra. Dans ce cas, il y aura un facteur temps entre les deux captures ce qui rend le système inutilisable en temps réel.

# Bibliographie


[1] dodeca2360, 1994.

[2] the fc 1005 camera and the fc 110 camera, 1999.

[3] Ladybug, 2013.

[4] Roundshot, 2013.

[5] Surroundvideo, 2013.

[6] Manoj Aggarwal and Narendra Ahuja. High dynamic range panoramic imaging. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 2–9. IEEE, 2001.

[7] Tom as Svoboda and Tom as Pajdla. Central panoramic cameras : Geometry and design. 1997.

[8] Simon Baker and Shree K Nayar. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, 35(2) :175–196, 1999.

[9] Simon Baker and Shree K Nayar. Single viewpoint catadioptric cameras. *PVo1*, pages 39–71, 2001.

[10] Hynek Bakstein and Tomas Pajdla. Panoramic mosaicing with a 180 field of view lens. In *Omnidirectional Vision, 2002. Proceedings. Third Workshop on*, pages 60–67. IEEE, 2002.

[11] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3) :346–359, 2008.

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf : Speeded up robust features. In *Computer Vision–ECCV 2006*, pages 404–417. Springer, 2006.

[13] Ahmed Nabil Belbachir, Manfred Mayerhofer, Daniel Matolin, and J Colineau. Real-time 360Â° panoramic views using bica360, the fast rotating dynamic vision sensor to up to 10 rotations per sec. In *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, pages 727–730. IEEE, 2012.

[14] Ahmed Nabil Belbachir, Roman Pflugfelder, and Roman Gmeiner. A neuromorphic smart camera for real-time 360Â° distortion-free panoramas. In *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras*, pages 221–226. ACM, 2010.

[15] R Benosman and J Devars. Panoramic stereo vision sensor. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 767–769. IEEE, 1998.

[16] Ryad Benosman and Sing B Kang. *Panoramic vision : sensors, theory, and applications*. Springer, 2001.

[17] Udhav Bhosle, Sumantra Dutta Roy, and Subhasis Chaudhuri. Multispectral panoramic mosaicing. *Pattern recognition letters*, 26(4) :471–482, 2005.

[18] Merwan Birem and Francois Berry. Hardware architecture for visual feature extraction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, Algarve, Portugal, Oct 2012.

[19] Merwan Birem and Franois Berry. Fpga-based real time extraction of visual features. *2012 IEEE International Symposium on Circuits and Systems*, pages 3053–3056, May 2012.

[20] Anko Börner, Heiko Hirschmüller, Karsten Scheibe, Michael Suppa, and Jürgen Wohlfeil. Mfc-a modular line camera for 3d world modeling. *Lecture Notes in Computer Science, Berlin*, 2008.

[21] Paul Bourke. Synthetic stereoscopic panoramic images. In *Interactive Technologies and Sociotechnical Systems*, pages 147–155. Springer, 2006.

[22] Metz Brandon and Morfopoulos Arin. Rapid corner detection using fpgas. In *ational Aeronautics and Space Administration*, 2010.

[23] Eric Brassart, Laurent Delahoche, Cyril Cauchois, Cyril Drocourt, Claude Pegard, and El Mustapha Mouaddib. Experimental results got with the omnidirectional vision sensor : Syclop. In *Omnidirectional Vision, 2000. Proceedings. IEEE Workshop on*, pages 145–152. IEEE, 2000.

[24] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1) :59–73, 2007.

[25] Li-huan CAI, Ying-hao LIAO, and Dong-hui GUO. Study on image stitching methods and its key technologies [j]. *Computer Technology and Development*, 18(3) :1–4, 2008.

[26] Chien-Wei Chen, Li-Wei Chan, Yu-Pao Tsai, and Yi-Ping Hung. Augmented stereo panoramas. In *Computer Vision–ACCV 2006*, pages 41–49. Springer, 2006.

[27] Naoki Chiba, Hiroshi Kano, Michihiko Minoh, and Masashi Yasuda. Feature-based image mosaicing. *Systems and Computers in Japan*, 31(7) :1–9, 2000.

[28] Junguk Cho, Joon Hyuk Cha, Yong Min Tai, Young-Su Moon, and Shihwa Lee. Stereo panoramic image stitching with a single camera. In *Consumer Electronics (ICCE), 2013 IEEE International Conference on*, pages 256–257. IEEE, 2013.

[29] Jonathan Courbon, Youcef Mezouar, Laurent Eck, and Philippe Martinet. A generic fisheye camera model for robotic applications. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1683–1688. IEEE, 2007.

[30] Xiao-Ming Deng, Fu-Chao Wu, and Yi-Hong Wu. An easy calibration method for central catadioptric cameras. *Acta automatica sinica*, 33(8) :801–808, 2007.

[31] Luigi Di Stefano, Stefano Mattoccia, and Federico Tombari. Zncc-based template matching using bounded partial correlation. *Pattern recognition letters*, 26(14) :2129–2134, 2005.

[32] Benedikt Dietrich. Design and implementation of an fpga-based stereo vision system for the eyebot m6. *University of Western Australia*, 2009.

[33] Martin A Fischler and Robert C Bolles. Random sample consensus : a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6) :381–395, 1981.

[34] Chunyu Gao, Hong Hua, and Narendra Ahuja. A hemispherical imaging camera. *Computer Vision and Image Understanding*, 114(2) :168–178, 2010.

[35] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3) :335–360, 2011.

[36] Christopher Geyer and Kostas Daniilidis. Equivalence of catadioptric projections and mappings of the sphere. In *Omnidirectional Vision, 2000. Proceedings. IEEE Workshop on*, pages 91–96. IEEE, 2000.

[37] Christopher Geyer and Kostas Daniilidis. A unifying theory for central panoramic systems and practical implications. In *Computer Visio–ECCV 2000*, pages 445–461. Springer, 2000.

[38] Christopher Geyer and Kostas Daniilidis. Catadioptric projective geometry. *International Journal of Computer Vision*, 45(3) :223–243, 2001.

[39] Arturo Gil, Oscar Martinez Mozos, Monica Ballesta, and Oscar Reinoso. A comparative evaluation of interest point detectors and local descriptors for visual slam. *Machine Vision and Applications*, 21(6) :905–920, 2010.

[40] J-J Gonzalez-Barbosa and Simon Lacroix. Fast dense panoramic stereovision. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1210–1215. IEEE, 2005.

[41] Deborah Goshorn, Junguk Cho, Ryan Kastner, and Shahnam Mirzaei. Field programmable gate array implementation of parts-based object detection for real time video applications. In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pages 582–587. IEEE, 2010.

[42] D Griessbach, A Börner, M Scheele, K Scheibe, and S Sujew. Line scanner in combination with inertial measurement unit. In *Panoramic Photogrammetry Workshop*, volume 34, pages 19–22, 2004.

[43] Luis E Gurrieri and Eric Dubois. Stereoscopic cameras for the real-time acquisition of panoramic 3d images and videos. In *IS&T/SPIE Electronic Imaging*, pages 86481W–86481W. International Society for Optics and Photonics, 2013.

[44] Norbert Haala, Ralf Reulke, Michael Thies, and Tobias Aschoff. Combination of terrestrial laser scanning with high resolution panoramic images for investigations in forest applications and tree species recognition. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(5/W16), 2004.

[45] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[46] Tomoyuki Hirota, Hajime Nagahara, and Masahiko Yachida. Calibration of rotating line camera for spherical imaging. In *Computer Vision–ACCV 2006*, pages 389–398. Springer, 2006.

[47] Paul VC Hough. Method and means for recognizing complex patterns, December 1962.

[48] Guoqiang Hu, Darren Aiken, Sumit Gupta, and Warren E Dixon. Lyapunov-based range identification for paracatadioptric systems. *Automatic Control, IEEE Transactions on*, 53(7) :1775–1781, 2008.

[49] Hong Hua, Narendra Ahuja, and Chunyu Gao. Design analysis of a high-resolution panoramic camera using conventional imagers and a mirror pyramid. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2) :356–361, 2007.

[50] Fay Huang, Shou Kang, Georgy Gimel'farb, Ralf Reulke, Martin Scheele, and Karsten Scheibe. Cylindrical panoramic cameras-from basic design to applications. Technical report, CITR, The University of Auckland, New Zealand, 2002.

[51] Fay Huang, Reinhard Klette, Anko Börner, Ralf Reulke, Martin Scheele, and Karsten Scheibe. Hyper-resolution and polycentric panorama acquisition and experimental data collection. Technical report, CITR, The University of Auckland, New Zealand, 2001.

[52] Fay Huang, Reinhard Klette, and Karsten Scheibe. *Panoramic imaging : sensor-line cameras and laser range-finders*, volume 11. Wiley, 2008.

[53] Fay Huang, Reinhard Klette, and Yun-Hao Xie. *Pose Estimation for Sensors Which Capture Cylindric Panoramas*. Springer, 2008.

[54] Fay Huang, Reinhard Klette, and Yun-Hao Xie. Sensor pose estimation from multi-center cylindrical panoramas. In *Advances in Image and Video Technology*, pages 48–59. Springer, 2009.

[55] Fay Huang, Shou Kang Wei, and Reinhard Klette. Calibration of panoramic cameras using 3d scene information. In *Geometry, Morphology, and Computational Imaging*, pages 335–345. Springer, 2003.

[56] Fay Huang, Shou-Kang Wei, and Reinhard Klette. Comparative studies of line-based panoramic camera calibration. In *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*, volume 7, pages 71–71. IEEE, 2003.

[57] Fay Huang, Shou Kang Wei, and Reinhard Klette. Rotating line cameras : epipolar geometry and spatial sampling. *Institute for Mathematics and its Applications, University of Minnesota, IMA Preprint Series, report*, 2105 :46, 2006.

[58] Ho-Chao Huang and Yi-Ping Hung. Spisy : The stereo panoramic imaging system. In *RAMS 97, Third Workshop on Real-Time and Media Systems*, pages 71–78. Citeseer, 1997.

[59] Ho-Chao Huang and Yi-Ping Hung. Panoramic stereo imaging system with automatic disparity warping and seaming. *Graphical Models and Image Processing*, 60(3) :196–208, 1998.

[60] Luo Juan and Oubong Gwun. Surf applied in panorama image stitching. In *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*, pages 495–499. IEEE, 2010.

[61] Hwa-Sung Kim, Hyun-Chul Kim, Won-Kyu Lee, and Chang-Hun Kim. Stitching reliability for estimating camera focal length in panoramic image mosaicing. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 596–599. IEEE, 2000.

[62] Jonathan Klippenstein and Hong Zhang. Quantitative evaluation of feature extractors for visual slam. In *Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference on*, pages 157–164. IEEE, 2007.

[63] G Kweon. Panoramic image composed of multiple rectilinear images generated from a single fisheye image. *J. Opt. Soc. Korea*, 14 :109–120, 2010.

[64] Gyeong-Il Kweon and Young-Ho Choi. Image-processing based panoramic camera employing single fisheye lens. *J. Opt. Soc. Korea*, 14 :245–259, 2010.

[65] Jie Lei, Xin Du, Yun-fang Zhu, and Ji-lin Liu. Unwrapping and stereo rectification for omnidirectional images. *Journal of Zhejiang University SCIENCE A*, 10(8) :1125–1139, 2009.

[66] K Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quart. App1. Math.*, (2) :164–168, 1944.

[67] Weiming Li and YF Li. Single-camera panoramic stereo imaging system with a fisheye lens and a convex mirror. *Optics Express*, 19(7) :5855–5867, 2011.

[68] Yin Li, Heung-Yeung Shum, Chi-Keung Tang, and Richard Szeliski. Stereo reconstruction from multiperspective panoramas. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(1) :45–62, 2004.

[69] Yiran Li. Fpga implementation for image processing algorithms. *Digital Signal Processing*, 2006.

[70] Jörgen Lidholm, Fredrik Ekstrand, and Lars Asplund. Two camera system for robot applications ; navigation. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 345–352. IEEE, 2008.

[71] Tony Lindeberg. *Scale-space theory in computer vision*. Springer, 1993.

[72] Gonzalo López-Nicolás and Carlos Sagüés. Catadioptric camera model with conic mirror. In *BMVC*, pages 1–10, 2010.

[73] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[74] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110, 2004.

[75] Wen Lik Dennis Lui and Ray Jarvis. Eye-full tower : A gpu-based variable multi-baseline omnidirectional stereovision system with automatic baseline selection for outdoor mobile robot navigation. *Robotics and Autonomous Systems*, 58(6) :747–761, 2010.

[76] Satya Prakash Mallick. Feature based image mosaicing. *Department of Electrical and Computer Engineering, University of California, San Diego*, 2002.

[77] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2) :431–441, 1963.

[78] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10) :1615–1630, 2005.

[79] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, obotics Institute, Stanford University, 1980.

[80] Jorge J Moré. The levenberg-marquardt algorithm : implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[81] Vic Nalwa. A true omnidirectional viewer. Technical report, technical report, Bell Laboratories, 1996.

[82] Frank Nielsen. Surround video : a multihead camera approach. *The visual computer*, 21(1-2) :92–103, 2005.

[83] Frank Nielsen, Alexis André, and Shigeru Tajima. Real-time spherical videos from a fast rotating camera. In *Image Analysis and Recognition*, pages 326–335. Springer, 2008.

[84] Navid Nourani-Vatani, P Borges, and J Roberts. A study of feature extraction algorithms for optical flow tracking. In *Australasian Conference on Robotics and Automation*, 2012.

[85] Mark Ollis, Herman Herman, and Sanjiv Singh. *Analysis and design of panoramic stereo vision using equi-angular pixel cameras*. Citeseer, 1999.

[86] Radu Orghidan, El Mustapha Mouaddib, and Joaquim Salvi. Omnidirectional depth computation from a single image. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1222–1227. IEEE, 2005.

[87] Tomáš Pajdla. Stereo with oblique cameras. *International Journal of Computer Vision*, 47(1-3) :161–170, 2002.

[88] Jafar Amiri Parian and Armin Gruen. An advanced sensor model for panoramic cameras. *ISPRS Int. Arch. Photogrammetry Remote Sensing Spatial Information Sciences*, 35 :24–29, 2004.

[89] Jafar Amiri Parian and Armin Gruen. A refined sensor model for panoramic cameras. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34, 2004.

[90] Jafar Amiri Parian and Armin Gruen. Panoramic camera calibration using 3d straight lines. In *ISPRS Panoramic Photogrammetry Workshop, Berlin, Germany*, pages 24–25, 2005.

[91] Min Woo Park, Kyung Ho Jang, and Soon Ki Jung. Panoramic vision system for an intelligent vehicle using a laser sensor and cameras. In *The 17th Intelligent Transport Systems World Congress*, 2010.

[92] Peter Peer and Franc Solina. Mosaic-based panoramic depth imaging with a single standard camera. In *Stereo and Multi-Baseline Vision, 2001.(SMBV 2001). Proceedings. IEEE Workshop on*, pages 75–84. IEEE, 2001.

[93] Peter Peer and Franc Solina. Panoramic depth imaging : Single standard camera approach. *International Journal of Computer Vision*, 47(1-3) :149–160, 2002.

[94] Peter Peer and Franc Solina. Towards a real time panoramic depth sensor. In *Computer Analysis of Images and Patterns*, pages 107–115. Springer, 2003.

[95] Shmuel Peleg and Moshe Ben-Ezra. Stereo panorama with a single camera. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999.

[96] Shmuel Peleg, Moshe Ben-Ezra, and Yael Pritch. Omnistereo : Panoramic stereo imaging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3) :279–290, 2001.

[97] Shmuel Peleg, Moshe Ben-Ezra, and Yael Pritch. Panoramic imaging with horizontal stereo. *Panoramic Vision : sensors, theory, and applications New York : Springer-Verlag*, pages 143–I60, 2001.

[98] Shmuel Peleg and Joshua Herman. Panoramic mosaics by manifold projection. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 338–343. IEEE, 1997.

[99] Shmuel Peleg, Yael Pritch, and Moshe Ben-Ezra. Cameras for stereo panoramic imaging. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 208–214. IEEE, 2000.

[100] Nicolas Pinto, Youssef Barhomi, David D Cox, and James J DiCarlo. Comparing state-of-the-art visual features on invariant object recognition tasks. In *Applications of computer vision (WACV), 2011 IEEE workshop on*, pages 463–470. IEEE, 2011.

[101] Yael Pritch, Moshe Ben-Ezra, and Shmuel Peleg. Automatic disparity control in stereo panoramas (omnistereo). In *Omnidirectional Vision, 2000. Proceedings. IEEE Workshop on*, pages 54–61. IEEE, 2000.

[102] Yael Pritch, Moshe Ben-Ezra, and Shmuel Peleg. Optics for omnistereo imaging. In *Foundations of Image Understanding*, pages 447–467. Springer, 2001.

[103] Orghidan Radu et al. *Catadioptric stereo based on structured light projection*. Universitat de Girona, 2006.

[104] Ralf Reulke and Martin Scheele. Ccd-line digital imager for photogrammetry in architecture. *International Archives of Photogrammetry and Remote Sensing*, 32 :195–201, 1997.

[105] Ralf Reulke, Aloysius Wehr, and D Griesbach. *Mobile panoramic mapping using CCD-line camera and laser scanner with integrated position and orientation system*. Springer, 2006.

[106] Ralf Reulke, Aloysius Wehr, Martin Scheele, and Karsten Scheibe. Panoramic mapping using ccd-line camera and laser scanner with integrated position and orientation system. Technical report, CITR, The University of Auckland, New Zealand, 2003.

[107] V Rodehorst and A Koschan. Comparison and evaluation of feature point detectors. In *Proc. 5th International Symposium Turkish-German Joint Geodetic Days" Geodesy and Geoinformation in the Service of our Daily Life", Berlin, Germany*, 2006.

[108] Miguel Romero, Yan Guo, Sio-Hoi Ieng, Frederic Plumet, Ryad Benosman, and Bruno Gas. Omni-directional camera and fuzzy logic path planner for autonomous sailboat navigation. In *Iberoamerican Conference on Electronics Engineering and Computer Science*, page 50, 2011.

[109] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[110] C Salinas, Hector Montes, G Fernandez, P Gonzalez de Santos, and M Armada. Catadioptric panoramic stereovision for humanoid robots. *Robotica*, 30(5) :799–811, 2012.

[111] Karsten Scheibe, Fay Huang, and Reinhard Klette. Pose estimation of rotating sensors in the context of accurate 3d scene modeling. *J. UCS*, 16(10) :1269–1290, 2010.

[112] Karsten Scheibe, Hartmut Korsitzky, Ralf Reulke, Martin Scheele, and Michael Solbrig. Eyescan-a high resolution digital panoramic camera. In *Robot Vision*, pages 77–83. Springer, 2001.

[113] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(5) :530–535, 1997.

[114] D Schneider and HG Maas. Development and application of an extended geometric model for high resolution panoramic cameras. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(B5) :366–371, 2004.

[115] D Schneider and HG Maas. Combined bundle adjustment of panoramic and central perspective images. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5/W8) :4, 2005.

[116] D Schneider, M Pötzsch, and HG Maas. Accuracy and application potential of the 94 megapixel rgb macro-scanning camera pentacon scan 5000. In *CIPA Symposium*, pages 534–537, 2005.

[117] Danilo Schneider and Hans-Gerd Maas. Geometric modelling and calibration of a high resolution panoramic camera. *Optical 3-D Measurement Techniques VI*, pages 122–129, 2003.

[118] Danilo Schneider and Hans-Gerd Maas. A geometric model for linear-array-based terrestrial panoramic cameras. *The Photogrammetric Record*, 21(115) :198–210, 2006.

[119] Danilo Schneider and HG Maas. Geometrische modellierung und kalibrierung einer hochauflösenden digitalen rotationszeilenkamera. *Oldenburger 3D-Tage*, 27, 2003.

[120] Miriam Schönbein, Bernd Kitt, and Martin Lauer. Environmental perception for intelligent vehicles using catadioptric stereo vision systems. In *Proceedings of the 5th European Conference on Mobile Robots (ECMR), Örebro, Schweden*, 2011.

[121] Ellen Schwalbe. Geometric modelling and calibration of fisheye lens camera systems. In *Proc. 2nd Panoramic Photogrammetry Workshop, Int. Archives of Photogrammetry and Remote Sensing*, volume 36, page W8, 2005.

[122] Abd El Rahman Shabayek, Cédric Demonceaux, Olivier Morel, and David Fofi. Vision based uav attitude estimation : progress and insights. *Journal of Intelligent and Robotic Systems*, 65(1-4) :295–308, 2012.

[123] Ken Shoemake. Plücker coordinate tutorial. *Ray Tracing News*, 11(1) :4, 1998.

[124] Heung-Yeung Shum, Adam Kalai, and Steven M Seitz. Omnivergent stereo. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 22–29. IEEE, 1999.

[125] Heung-Yeung Shum and Richard Szeliski. Stereo reconstruction from multiperspective panoramas. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 14–21. IEEE, 1999.

[126] Heung-Yeung Shum and Richard S Szeliski. Stereo reconstruction from multiperspective panoramas, October 2003.

[127] L Smadja, R Benosman, and J Devars. Determining epipolar constraint on cylindrical images and using it for 3d reconstruction. In *Proc. ICAR*, 2001.

[128] Laurent Smadja, Ryad Benosman, and Jean Devars. Cylindrical sensor calibration using lines. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 3, pages 1851–1854. IEEE, 2004.

[129] Laurent SMADJA, Erwan BIGORGNE, Ryad BENOSMAN, and Jean DEVARS. Génération de cartes de disparité denses à partir d'images panoramiques cylindriques haute définition. In *19Â° Colloque sur le traitement du signal et des images, FRA, 2003*. GRETSI, Groupe d Etudes du Traitement du Signal et des Images, 2003.

[130] LC Su, CJ Luo, and F Zhu. Obtaining obstacle information by an omnidirectional stereo vision system. *International Journal of Robotics and Automation*, 24(3), 2009.

[131] Changming Sun and Shmuel Peleg. Fast panoramic stereo matching using cylindrical maximum surfaces. *Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions on*, 34(1) :760–765, 2004.

[132] Tomáš Svoboda and Tomáš Pajdla. Epipolar geometry for central catadioptric cameras. *International Journal of Computer Vision*, 49(1) :23–37, 2002.

[133] Kar-Han Tan, Hong Hua, and Narendra Ahuja. Multiview panoramic cameras using mirror pyramids. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(7) :941–946, 2004.

[134] Kenji Tanaka and Susumu Tachi. Tornado : omnistereo video imaging with rotating optics. *Visualization and Computer Graphics, IEEE Transactions on*, 11(6) :614–625, 2005.

[135] Simon Thibault, Jocelyn Parent, Hu Zhang, Martin Larivière-Bastien, Anne-Sophie Poulin-Girard, Aymen Arfaoui, and Pierre Désaulniers. Developments in modern panoramic lenses : lens design, controlled distortion, and characterization. In *International Conference on Optical Instruments and Technology (OIT2011)*, pages 81970I–81970I. International Society for Optics and Photonics, 2011.

[136] Carlo Tomasi and Jianbo Shi. Good features to track. *CVPR94*, 600 :593–593, 1994.

[137] Christian Weissig, Oliver Schreer, Peter Eisert, and Peter Kauff. The ultimate immersive experience : panoramic 3d video acquisition. In *Advances in Multimedia Modeling*, pages 671–681. Springer, 2012.

[138] Yalin Xiong and Kenneth Turkowski. Creating image-based vr using a self-calibrating fisheye lens. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 237–243. IEEE, 1997.

[139] Xianghua Ying and Zhanyi Hu. Catadioptric camera calibration using geometric invariants. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(10) :1260–1271, 2004.

[140] Lin Zeng, Dexiang Deng, Xi Chen, and Yunlu Zhang. A self-adaptive and real-time panoramic video mosaicing system. *Journal of Computers*, 7(1) :218–225, 2012.

# Table des figures

# Liste des tableaux