UNIVERSITAT POLITÈCNICA DE CATALUNYA
(UPC)

The design of a robust 3D Reconstruction system for video sequences in non controlled environments

by

Nicolás Herrero Molina

Advisors: José Luís Landabaso Díaz
Ferran Marqués Acosta

in the

MERIT- European Master of Research on Information and Communication Technologies
Teoria del Senyal i les Comunicacions

February 2010

"Making a computer see was something that leading experts in the field of Artifial Intelligence thought to be at the level of difficulty of a summer student's project back in the 60's. Forty years later the task is still unsolved and seems formidable"

Olivier Faugeras

Abstract

Along this thesis, a novel and robust approach for obtaining 3D models from video sequences captured with hand-held cameras is adressed. This work defines a fully automatic pipeline that is able to deal with different types of sequences and acquiring devices. The designed and implemented system follows a *divide and conquer approach*. An smart frame decimation process reduces the temporal redundancy of the input video sequence an selects the best conditioned frames for the reconstruction step. Next, the video is split into overlapped clips with a fixed and small number of Key-frames. This allows to parallelize the Structure and Motion process which translates into a dramatic reduction in the computational complexity. The short length of the clips allows an intensive search for the best solution at each step of the reconstruction, which improves the overall system performance. The process of feature tracking is embedded within the reconstruction loop for each clip as a difference with other approaches. The last contribution of this thesis is a final registration step that merges all the processed clips to the same coordinate frame. This last step consists on a set of linear algorithms that combine information of the structure (3D points) and motion (cameras) shared by partial reconstructions of the same static scene to more accurately estimate their registration to the same coordinate system. The performance for the presented algorithm as well as for the global system is demonstrated in experiments with real data.

Acknowledgements

Saltándome completa y voluntariamente el protocolo de escribir esta tesis íntegramente en inglés, me gustaría empezar agradeciendo a Ferrán Marqués su apoyo y atención desde que empezamos a colaborar hace ya algún tiempo. Sobre todo quisiera agradecerle el haberme ofrecido la posibilidad de embarcarme en esta experiencia doctoral ya que, gracias a ella, he podido conocer a mis colegas de la iniciativa VideoSurfing en Telefónica I+D. No tengo más que palabras de agradecimiento tanto para José Luís Landabaso como para José Carlos Pujol. He aprendido más de vosotros en este tiempo que entre carrera y máster. En especial, un millón de gracias para José Luís por haberse echado al hombro tanto la dirección de esta tesis como tantas cosas para ayudarme en el día a día.

Por supuesto, agradecer a mi familia todo su apoyo y cercanía así como su comprensión con mis largos ratos de autismo autoimpuesto debido a deadlines, redacciones de tesis, trabajos, etc.

Y por último y por ello mejor, todas las gracias del mundo a ti, Rebe.

Contents

Abstract iv

Acknowledgements v

List of Figures ix

List of Tables xi

1 Introduction 1
- 1.1 State of the art . . . 2
- 1.2 Motivation and contributions . . . 4

2 Theoretical Background 7
- 2.1 Projective geometry . . . 7
  - 2.1.1 Homogeneous coordinates . . . 8
  - 2.1.2 Hierarchy of geometries . . . 9
  - 2.1.3 The 2D projective plane . . . 10
  - 2.1.4 The 3D projective space . . . 13
- 2.2 The pinhole camera . . . 16
  - 2.2.1 Camera rotation and translation . . . 17
  - 2.2.2 Perspective transformation . . . 17
  - 2.2.3 Intrinsic parameters . . . 18
- 2.3 Epipolar geometry . . . 19
  - 2.3.1 The fundamental matrix . . . 19
  - 2.3.2 The Essential Matrix . . . 21
- 2.4 Structure And Motion . . . 23
  - 2.4.1 Feature point detectors . . . 24
  - 2.4.2 Two view relationship . . . 26
  - 2.4.3 Structure computation . . . 27
  - 2.4.4 Camera resectioning methods . . . 28

3 Structure and Motion for Video Sequences. System overview 31
- 3.1 System Architecture . . . 31

3.2 Selection of relevant frames 32
3.2.1 Feature detection and matching 33
3.2.2 Frame Decimation 34
3.3 Robust Multi-View Structure and Motion 36
3.3.1 Feature tracking 36
3.3.2 Initial-pair estimation 37
3.3.3 Updating Structure and Motion 38
3.3.4 Registration of partial reconstructions 40

4 Registration of independent 3D reconstructions 43
4.1 Overview of the algorithm 44
4.2 Problem Statement 46
4.3 Linear Algorithm due to Structure 47
4.4 Linear Algorithm due to Motion 49
4.5 Refinement of results 51
4.5.1 Minimizing Algebraic error (LIN) 52
4.5.2 Minimizing geometric error (STR) 52

5 Experimental setup 55
5.1 Types of sequences 55
5.2 Partial Reconstructions 57
5.2.1 Frame decimation analysis 57
5.2.2 Comparison with other schemes 59
5.3 Registration Performance 61
5.4 Qualitative Results 67

6 Conclusions and future work 69

Bibliography 71

List of Figures

2.1 Ray Space model for the Projective plane $\mathbb{P}^2$. Points and lines are described as rays and planes respectively intersecting the Euclidean plane $\mathbb{R}^2$.(Reprint from [1]) 12

2.2 Structure of a pinhole camera. It consists on a camera center $\mathbf{C}$ and an image plane in front of it. The principal point is denoted as $\mathbf{p}$.(Reprint from [1]) 16

2.3 The projection matrix P describing the intrinsic and extrinsic properties of a pinhole camera is composed of three steps: a rigid transformation from world coordinates to camera coordinates, a perspective transformation and a 2D homography relating points in the image plane to pixels $\mathbf{x}$.(Reprint from [1]) 17

2.4 Epipolar geometry. Camera centres $\mathbf{C}$ and $\mathbf{C}'$, the 3D point $\mathbf{X}$ and its images $\mathbf{x}$ and x' lie in the same plane(Reprint from [1]) 20

2.5 Illustration of the stratified approach for 3D reconstruction. (a-b) Input images for the 3D reconstruction system. (c-d)Projective reconstruction. (e-f) Affine reconstruction. (g-h) Metric reconstruction. (i-j) Photo-realistic reconstruction by means of planar texture mapping. (Reprint from [1]) 22

2.6 The four possible solution for the initial pose estimation. Only the solution that triangulates all the correspondences in front of both cameras is accepted.(Reprint from [1]) 26

2.7 Point triangulation. There not exist an exact solution on the position of $\mathbf{X}$ since its detected images are corrupted by noise. Therefore its value is estimated $\hat{\mathbf{X}}$ and the quality of the estimation is given by the reprojection error. (Reprint from [1]) 29

3.1 Example of workflow for our Structure and motion system. a) Input data is a video sequence recording a rigid scene. b) The scene is reconstructed as cloud of points. c) The cloud of points may be furtherly improved through a process of densification. 32

3.2 Block diagram of the full SaM system. As seen a *divide an conquer approach* is followed by means of a parallelization of the reconstruction loop 33

3.3 Graphical explanation of the registration of partial reconstructions step. The triangles in red refer to overlapping cameras 40

5.1 Snapshots of the sequences considered along the results chapter. (a-d) Cat Sequence. (e-h) Dragon Sequence. (i-l) Snake Sequence. (m-p) Sagrada Familia Sequence. 56
5.2 Graphical analysis of the frame decimation step for the sequences . 57
5.3 3D model for Cat Sequence 61
5.4 3D model for Dragon Sequence 62
5.5 3D model for Snake Sequence 63
5.6 3D model for Sagrada Familia Sequence 64
5.7 Dense 3D models for Cat and Dragon sequences 66
5.8 Dense 3D models for Snake and Sagrada Familia Sequences 67

List of Tables

2.1 Summary of invariants and permitted transformations for each type of geometry 9
2.2 Summary of point and line relationships and properties in $\mathbb{P}^2$ 11

4.1 Configurations for RANSAC's Minimum Sample Set. 45

5.1 Numerical evaluation of the reconstruction of the four sequences for two different approaches. The quality measurements considered are the number of triangulated points (Points) and correctly resected cameras (Cams), the number of KFs with respect the total numbe of frames, errors before (E_{preBA}) and after (E_{postNA}) Bundle Adjustment and total reconstructed time (T_rec). The column Clips refers to the number of subsequences created from the original input video. 60
5.2 Numerical evaluation of four different registration schemes on the given datasets. 65

To my family...

Chapter 1

Introduction

In the recent years, three-dimensional reconstruction of non-controlled environments from user's photos and videos has attracted major interest from the Computer Vision community. During decades, research efforts in this field and, more precisely, in Structure and Motion techniques (SaM), were mainly focused in providing solutions to particular problems related to 3D reconstruction: feature detection, matching and tracking, structure and camera pose computation, densification methods, *etc.* Nowadays, the objective is to develop robust and automatic pipelines for three-dimensional reconstruction, by means of combining this set of already mature algorithms [2–4].

Moreover, reconstruction of large scenarios from unordered photo collections, has become one of the major topics of research in SaM [2, 3, 5]. Its tremendous potential for touristic and social applications has attracted the attention of several technological companies. As a consequence, the first commercial products have been launched allowing users to create their own three-dimensional content and interact with it. Microsoft Photosynth [6], based on the previous work of Washington University [2], is the most relevant contribution in this field. From sets of images uploaded from users, the system is capable of reconstructing the 3D scene and representing it as a cloud of points. The position and orientation from where the photos were taken are also provided. In addition, it allows the user to interact with the created 3D world.

Nevertheless, although the mentioned great efforts in 3D reconstruction from photographs, there is still an active research in SaM from video sequences. Some examples of it are the recent development of real-time systems for several applications such as dense city modeling [7] or visual odometry applications [4]. Some specific issues associated to SaM from video are still been tackled in recent papers

in literature: drift propagation through frames as the sequence length grows [8–10], temporal redundancy [11] and the difficulty of providing the reconstruction loop with an accurate set of tracked features through frames.
Along this work, a fully automatic and robust pipeline for 3D reconstruction from video sequences is presented. Special interest was devoted to design a system able to reconstruct different types of static sequences: objects, cars, buildings, *etc.* Moreover, the reconstruction pipeline is intended to deal with low quality acquiring devices, such as cell phones and cheap digital cameras.

1.1 State of the art

Although the achieved level of maturity in SaM, there are some issues that still limit the capabilities of this kind of systems. Among them, reducing the large computational complexity that these type of algorithms demand has shown to be an attractive topic for the research community. Bad-scalability of Newton-like optimization [12] represents the most restrictive bottleneck in terms of computational complexity for this discipline. Therefore, several approaches are found in literature to face this problem [5, 13].
In the case of photo collections, an accepted practice consists of pre-analyzing the set of snapshots and clustering them upon an affinity criterium. Clusters are independently processed [5, 13, 14] and *locally* optimized. That is the well-known *out-of-core optimization*. The equivalent when dealing with video sequences consists on defining atomic structures (*i.e.* triplets [15, 16]) that, once reconstructed and combined [17], represent the whole scene. This type of practice allows a parallelization of both the reconstruction process and the optimization step, which dramatically reduces the amount of computational resources needed.
As seen, the basic idea of this type of approaches consists on decoupling the problem into small pieces that are independently processed and, therefore, optimized demanding less computational resources. This independent processing requires of an additional merging stage to combine and homogeneize the obtained partial results. In fact, although partial reconstructions represent the same static scene, they are usually referred to distinct coordinate systems (due to the nature of the reconstruction algorithms used). Hence, points, cameras and all the geometric entities representing a 3D scene need to be geometrically registered to the same global coordinate frame. In the following, references to SaM systems following this

divide-and-conquer strategy, as well as geometric data registration algorithm are presented.

Geometric data registration has been a widely studied topic for several applications: SaM, geo-referentiation of SaM point clouds, fusion of active and passive sensors, visual odometry, *etc.* Depending on the nature of retrieved 3D data, registration algorithms may be classified as purely based in geometry or in data correspondences.

Geometry-based algorithms do not process data information directly. Instead, they use intermediate geometric primitives, which are matched by iteratively minimizing a geometric distance function. Iterative Closest Point (ICP) [18] between 3D point clouds is the most representative technique. It is commonly used for refinement and requires of a good initialization because of its tendency to be trapped in local minima. Some examples using this type of registration technique are [19, 20].

On the other hand, correspondence-based algorithms for registration make use of *a priori* known shared data. Correspondence can be achieved by exploiting texture (*i.e.* SIFT-like descriptor matching [21]) or temporal redundancy (*i.e.* overlapped cameras in video sequences), among others. This allows to compute the registration directly from the data, without recourse to other primitives.

In SaM from video sequences, the problem of registering reconstructions can be tackled as an iterative approach to avoid drift accumulation over frames. A hierarchical method to align consecutive camera triplets into a global projective frame is presented in [15]. Shared 3D points, retrieved from 2D correspondences, and one or two overlapped cameras are exploited to obtain a projective transformation between camera triplets and their associated structures. The process is followed by a bundle adjustment (BA) step and iterated until the whole sequence has been correctly registered. In [16], a generalization of [15] is proposed by selecting new triplets to be merged in an adaptive manner according to sequence motion, frame rate and amount of parallax.

Even for realtime reconstruction systems from video sequences, a sequential register of partial reconstructions is used. In this case, the objective is to avoid the drift propagation [4, 7, 22]. In [7], after an initial pose estimation, new cameras views and 3D points are iteratively added into a local metric coordinate system. The process is periodically refreshed (defined as a *firewall* introduction) and, therefore, a similarity transformation needs to be estimated: relative rotation and translation are obtained from one camera overlap, while the relative scale is obtained from 3D point correspondences.

The aforementioned algorithms use 3D correspondences and overlapping projection matrices as geometric primitives for aligment. In [17], Viewpoint-Invariant Patches (VIP) are defined as new primitives for obtaining a similarity transformation to register two independent metric reconstructions. VIP codifies the coordinates and normal direction of a 3D point (which encodes the relative translation and rotation), a SIFT descriptor and a patch scale. Hence, obtaining a similarity transformation is possible with just a single VIP correspondence.

1.2 Motivation and contributions

This thesis has been carried out within the development of a video-to-3D product (Video Surfing[1]). The objective of this project consists of implementing a system that allows users to generate their own 3D scenarios from recorded static scenes. Its workflow is as follows: the user records a touristic place, object, face, *etc.* and uploads this video to the project server. Next, the uploaded sequence is processed by means of a robust Structure and Motion pipeline that generates a dense 3D model of the recorded scene as well as recovers the path followed by the camera. Finally, the retrieved 3D information is used to feed an space-time player that allows the user to interact with the video and the obtained 3D model.
The main contribution of this thesis consists on the definition of a *divide and conquer* strategy for the reconstruction pipeline. As many other approaches in literature, the scalability problem is faced by decoupling the full reconstruction into smaller pieces that are independently processed and further combined. In this particular case, the input video sequence is split into shorter subsequences. In order to make this approach possible, a set of novel linear algorithms for registering the partial reconstructions have been co-developed, implemented and analyzed.
Moreover, several strategies have been introduced in order to provide a robust reconstruction for each one of the subsequences. A frame decimation step has been defined with the aim of providing the reconstruction looop with a set of well conditioned input Key-Frames. In addition, the process of pose estimation for the first pair of views and upgrade of the SaM for each one of the clips have been revisited in order to make them as robust an reliable as possible. Finally, a novel approach regarding feature tracking is the last contribution of this thesis. This process has been embedded within the reconstruction combining 2D an 3D information.

[1] http://surfing.tidprojects.com/MWVC

The remainder of the document is as follows. In Chapter 2, basic concepts of Projective geometry and Computer Vision are presented in order to serve as a quick reference for the rest of the chapters. Chapter 3 provides a global overview of the full system, as well as the presentation of the mentioned contributions of this thesis to the reconstruction loop: frame decimation and feature tracking. Chapter 4 is devoted to the mathematical formulation of both the linear algorithms for registering partial reconstructions and their optimization strategies. In Chapter 5, results with real sequences are provided and, finally, Chapter 6 presents the achieved conclussions and introduces some ideas for future research lines.

Chapter 2

Theoretical Background

The objective of this chapter is to present the associated mathematical concepts and computer vision techniques used for obtaining 3D reconstructions from image projections. The outline of the chapter is as follows. In section 2.1, the mathematical tools described by Projective Geometry are presented. This set of tools is on the basis of description of the image formation process. Section 2.2 is devoted to the description of the pinhole camera, which refers to the simplest mathematical definition of an image acquiring device. Section 2.3 introduces the Projective geometry between pairs of views and section 2.4 presents a set of techniques to extend them to a larger number of views.

2.1 Projective geometry

We are all familiar with Euclidean geometry since it is the way we were taught for describing our three-dimensional world. It fits very well in the processes of measuring angles, longitudes or even describing shapes since they are all invariants to this type of geometry. However, Euclidean geometry is not enough for describing the process of image formation in Computer Vision. Lengths and angles are not preserved, parallel lines meet at vanishing points, shapes look to be deformed (*i.e*, circles may seem ellipses), *etc*. Therefore, a more complete geometry able to describe this set of facts is used: Projective geometry.

In order to introduce the topic, let us focus in the property that, in the image formation model, parallel lines may seem to intersect. In 2D Euclidean geometry, two lines almost always meet in a plane point and if do not so, they are named *parallel* or they are said to meet at *infinity*. Unfortunately, this last assertion

crashes with the concept that *infinity* does not exist and it is only a convenient mathematical tool.
Nevertheless, we may upgrade the Euclidean world by means of adding these points at infinity (*ideal points*) where parallel lines meet. Thus, the Euclidean space is converted to a new geometric object, the Projective space. We need to think the Euclidean geometry as just a subset of a bigger one, the Projective geometry. The latter presents additional properties and allows different suitable transformations for describing the image formation model.
The conclusion that arises from the previous ideas is that a convenient way of representing the real 3D world is by extending it to the four-dimensional Projective space. In the following, we present the insights and mathematical formulation of this type of geometry and its applications in the field of Computer Vision.

2.1.1 Homogeneous coordinates

In order to formalize the aforementioned upgrade of the Euclidean space to a Projective one, we need to introduce some mathematical notation to allow us identifying or describing points at infinity. This is achieved by using homogeneous coordinates.
Let us represent a point in the Euclidean plane as the pair of numbers $\mathbf{x}_e = [x\ y]^T$. The conversion to homogeneous coordinates is performed just by adding an extra dimension $\mathbf{x}_p = [x\ y\ 1]^T$, that will define whether a point is located at infinity or not. In order to generalize the conversion to homogeneous coordinates we will define that $[x\ y\ 1]^T$, $[2x\ 2y\ 2]^T, \dots [kx\ ky\ k]^T$, represent the same point for any non-zero value of k. Let us consider that the general expression for a point in the Projective plane is given $[x\ y\ z]^T$.
It is straight forward to see that the inverse conversion of coordinates is achieved by simply dividing per z the two first coordinates $\left[\frac{x}{z}\ \frac{y}{z}\right]^T$ and it is here where the concept of infinity arises. If the third coordinate of an homogeneous point equals zero, it is equivalent to say that it is located at infinity. An intuitive demonstration of this concept is to apply the inverse conversion to non-homogeneous coordinates of:

$$[x\ y\ 0]^T \mapsto [\frac{x}{0}\ \frac{y}{0}]^T \sim [\infty, \infty]^T$$

We have presented how to extend the 2 -dimensional Euclidean space $\mathbb{R}^2$ to the Projective space $\mathbb{P}^2$ but it is direct to see that it can be extended to any conversion from $\mathbb{R}^n$ to $\mathbb{P}^n$. Nevertheless, it is important to keep track of what dimensions

	Euclidean	**Similarity**	**Affine**	**Projective**
Transformations				
Rotation	√	√	√	√
Translation	√	√	√	√
Isotropic scaling		√	√	√
Non-Isotropic scaling			√	√
Deformation			√	√
Homographies				√
Invariants				
Length	√			
Angle	√	√		
Length ratio	√	√		
Parallelism	√	√	√	
Incidence	√	√	√	√
Cross ratio	√	√	√	√

TABLE 2.1: Summary of invariants and permitted transformations for each type of geometry

represent in our problem. In $\mathbb{P}^2$, ideal points form the *line at infinity* and for the 4-dimensional $\mathbb{P}^3$ they form the *plane at infinity*, which is a useful tool for autocalibration.

2.1.2 Hierarchy of geometries

We have introduced the concept of Projective geometry as a superset of the well known Euclidean geometry, but they are not the only ones. In fact, there exist two other intermediate geometries between Euclidean and Projective: *Similarity* and *Affine*. These four geometries may be compared and studied from the point of view of the available invariants relying from the possible transformations in each space.

In table 2.1 invariants and transformations for each type of geometry are presented. Some benefits of projective transformations are that they preserve type (points remain points and lines remain lines), incidence and *cross ratio*. In addition, the number of permitted transformations (bijections between two projective basis) is higher than for Euclidean. This extra transformations are on the basis of the good description of image formation model that Projective geometry provides.

2.1.3 The 2D projective plane

In this section we introduce the homogeneous representation of points and lines in the Projective plane $\mathbb{P}^2$ and their intrinsic relationship. Recall that, given a point $\mathbf{x}_e = [x\ y]^T$ in $\mathbb{R}^2$, its corresponding representation in $\mathbb{P}^2$ is formed just by adding a third coordinate $\mathbf{x} = [x\ y\ 1]^T$. Also recall that the scaling factor is not relevant since $[x\ y\ 1]^T = [kx\ ky\ k]^T$ for any k different to zero. Point $[0\ 0\ 0]^T$ is not allowed.

A line in the plane is represented by the equation $ax + by + c = 0$ or equivalently with the vector $[a\ b\ c]^T$. Neither for lines the scale factor is relevant and it is evident that $[a\ b\ c]^T = k[a, b\ c]^T$. In addition, if $\mathbf{x} = [x\ y\ 1]^T$ and $\mathbf{l} = [a\ b\ c]^T$ the previous equation of the line can be obtained as the inner product:

$$\mathbf{x}^T\mathbf{l} = ax + by + c = 0 \tag{2.1}$$

Equation (2.1) yields an important result: the point $\mathbf{x}$ lies on the line $\mathbf{l}$ if and only if $\mathbf{x}^T\mathbf{l} = 0$. This is the so-called *incidence relation*. We may introduce further relationships between points and lines in order to introduce an important geometric consequence in the Projective plane: *The Duality Theorem*.

Line intersection and line joining two points: Given two lines $\{\mathbf{l}_1, \mathbf{l}_2\}$ and two points $\{\mathbf{x}_1, \mathbf{x}_2\}$ in the Projective plane, the point $\mathbf{x}$ where $\{\mathbf{l}_1, \mathbf{l}_2\}$ intersect and the line $\mathbf{l}$ joining $\{\mathbf{x}_1, \mathbf{x}_2\}$ are respectively given by the cross products:

$$\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2 \tag{2.2}$$

$$\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2 \tag{2.3}$$

Ideal points: Let us consider two parallel lines $\mathbf{l} = [a\ b\ c]^T$ and $\mathbf{l}' = [a\ b\ c']^T$. Applying (2.2), it yields that the point $\mathbf{x}_\infty$ is :

$$\mathbf{x}_\infty = (c' - c)[b\ \ -a\ 0]^T = [x\ y\ 0]^T, \tag{2.4}$$

and it represents the point at infinity where two parallel lines meet. This makes a difference with Euclidean Geometry where no possible intersection exists for the case of parallel lines.

Points		**Lines**	
Coordinates	$\mathbf{x} = [x\ y\ z]^T$	Coordinates	$\mathbf{l} = [a\ b\ c]^T$
Incidence	$\mathbf{x}^T\mathbf{l} = 0$	Incidence	$\mathbf{x}^T\mathbf{l} = 0$
Alignment	$det[\mathbf{x_1 x_2 x_3}] = 0$	Concurrence	$det[\mathbf{l_1 l_2 l_3}] = 0$
Line joining 2 points	$\mathbf{x} = \mathbf{l_1} \times \mathbf{l_2}$	2 lines intersection	$\mathbf{l} = \mathbf{x_1} \times \mathbf{x_2}$
Infinity point	$[x\ y\ 0]^T$	Infinity line	$[0\ 0\ c]^T$
Transformation rule	$\mathbf{x}' = \mathtt{H}\mathbf{x}$	Transformation rule	$\mathbf{l}' = \mathtt{H}^{-T}\mathbf{l}$

TABLE 2.2: Summary of point and line relationships and properties in $\mathbb{P}^2$

Line at infinity: Let us consider two ideal points $\mathbf{x} = [x\ y\ 0]^T$ and $\mathbf{x}' = [x'\ y'\ 0]^T$. Applying (2.3),the line $\mathbf{l}_\infty$:

$$\mathbf{l}_\infty = [0\ 0\ xy' - x'y]^T = [0\ 0\ c]^T \tag{2.5}$$

Arrived to this point the reader should have noticed an implicit relationship between points and lines. This relationship is on the basis of the Duality Theorem:

To any theorem of 2-dimensional projective geometry there corresponds a dual theorem, which may be derived by interchanging the roles of points and lines in the original theorem.

In table 2.2 a summary of the already explained and some extra relationships between points and lines are listed. For further details refer to [1].

A model of the Projective plane: It has been demonstrated that, when upgrading the Euclidean to the Projective plane, a point in $\mathbb{R}^2$ transforms in to a set of points in $\mathbb{R}^3$, or more precisely $\mathbb{P}^2$, related by a non-null scale factor. Therefore, a point $\mathbf{x} = [x_1\ x_2\ x_3]^T$ could be interpreted as a line in a three-dimensional space passing through the origin and intersecting in the $\mathbb{R}^2$ plane. On the other hand, a line $\mathbf{l} = [a\ b\ c]^T$ may be considered as plane intersecting the $\mathbb{R}^2$ plane. Moreover, the line at infinity $\mathbf{l}_\infty$ consists on the vertical plane $x_3 = 0$ since it contains all the points with third coordinate equal to zero (ideal points). In order to clear concepts up, a graphical explanation is presented in figure 2.1.

Hierarchy of transformations Let us now introduce the formulation of the already presented concept of Projective transformation. A Projective transformation is a linear mapping between two homogeneous 3-vectors represented by a

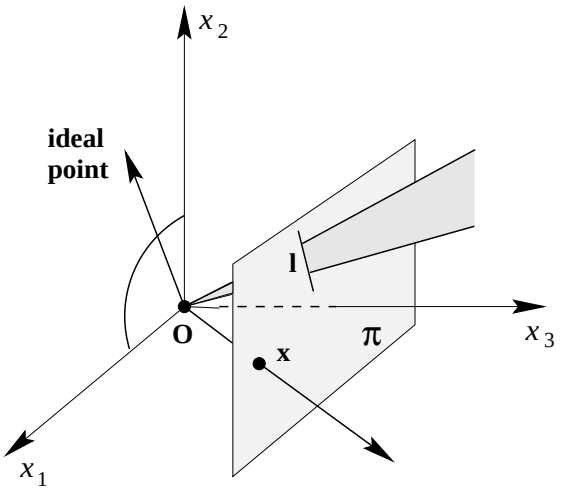


FIGURE 2.1: Ray Space model for the Projective plane $\mathbb{P}^2$. Points and lines are described as rays and planes respectively intersecting the Euclidean plane $\mathbb{R}^2$.(Reprint from [1])

non-singular 3×3 matrix $\mathtt{H}$:

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \tag{2.6}$$

or more briefly $\mathbf{x}' = \mathtt{H}\mathbf{x}$. It is important to note that an scale factor does not affect $\mathtt{H}$ and therefore it is said to be an homogeneous matrix.

Moreover, it is interesting to describe the four important specializations for a Projective transformation an their geometric properties. In Fig. 2.5 the 4 possible transformations are illustrated from the reconstruction point of view. Invariants for the four types of geometries from Table 2.1 are a consequence of the nature of the following transformations:

1. *Euclidean Transformation.* An Euclidean transformation is defined as applying a rotation plus a translation to a given geometric primitive.

$$\mathbf{x}' = \begin{bmatrix} cos\theta & -sin\theta & t_x \\ sin\theta & cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} = \mathtt{H}_e\mathbf{x} \tag{2.7}$$

   As seen it preserves lengths and distances since no scaling is applied. Because of that, this type of transformation is usually referred as Isometry.

2. *Similarity Transformation.* An Similarity transformation is defined as applying an isotropic scaling to an Euclidean transformation

$$\mathbf{x}' = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} = \mathtt{H}_s \mathbf{x} \tag{2.8}$$

When speaking about reconstruction, the term *metric* is usually employed when no extra information to images is provided. Since no data about the scale of the reconstruction, nor the orientation or the global position is provided, the metric reconstruction is said to be valid *up to a similarity transformation.* A similarity transformation preserves ratios of distances, shape, angles, *etc.* but not real distances.

3. *Affine transformation.* An affine transformation is formed by a singular linear transformation followed by a translation:

$$\mathbf{x}' = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} = \mathtt{H}_a \mathbf{x} \tag{2.9}$$

The non-isotropic scaling leads to not preserve shape, distance or angles. Nevertheless parallel lines remain parallel after an affine transformation and ratios of lengths or areas keep themselves unchanged.

4. *Projective transformation.* A projective transformation is a general non-singular linear transformation of homogeneous coordinates as expressed in (2.6). A projective transformation is the generalization of an affine transformation for non-homogeneous coordinates. The most relevant invariant for this type of transformation is the *cross-ratio* [1].

2.1.4 The 3D projective space

Along this subsection we extend the already presented concepts of the $\mathbb{P}^2$ geometry (points lines and transformations) to adapt them to the study of the projective space $\mathbb{P}^3$. More precisely, the concepts of planes, 3D points and the intrinsic relationships among them are introduced. As for the 2D Projective plane, the most relevant consequence is the *Point/Plane Duality Theorem.*

A point in $\mathbb{P}^3$ is represented by the homogeneous 4-vector $\mathbf{X} = [X_1\ X_2\ X_3\ X_4]^T$ and it is related to its representation $\mathbf{X}_e$ in $\mathbb{R}^2$ as the de-homogenization:

$$\mathbf{X}_e = [X\ Y\ Z]^T = \left[\frac{X_1}{X_4}\ \frac{X_2}{X_4}\ \frac{X_3}{X_4}\right]^T \tag{2.10}$$

On the other hand, the general equation of a plane follows $\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$. From 2.10, it provides the incidence relationship: $\pi_1 X_1 + \pi_2 X_2 + \pi_3 X_3 + \pi_4 X_4 = 0$ or more briefly:

$$\pi^T \mathbf{X} = \mathbf{X}^T \pi = 0 \tag{2.11}$$

Recall that, in the previous section, it was stated that points and lines were somehow equivalent. In $\mathbb{P}^3$ this equivalency is given between points and planes. That is the *Duality Theorem for Points and Planes*[1]. In the following, some of its consequences are presented.

Three points define a plane: given 3 points in general position $\mathbf{X}_1, \mathbf{X}_2$ and $\mathbf{X}_3$, the plane π which contains both three is the solution of the homogeneous system:

$$\begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \mathbf{X}_3^T \end{bmatrix} \pi = \mathtt{M}_x \pi = 0, \tag{2.12}$$

or, in other words, the null-space of the rank-3 matrix $\mathtt{M}_x$.

Three planes join in a point: this relation is dual to the previous one. Given 3 planes in general position π_1, π_2 and π_3, the point $\mathbf{X}$ where they uniquely meet is obtained as the solution of:

$$\begin{bmatrix} \pi_1^T \\ \pi_2^T \\ \pi_3^T \end{bmatrix} \mathbf{X} = \mathtt{M}_\pi \mathbf{X} = 0 \tag{2.13}$$

Projective Transformation: given a 4×4 $\mathbb{P}^3$ transformation $\mathtt{H}$, the rule for transforming points and planes is:

$$\mathbf{X}' = \mathtt{H}\mathbf{X} \tag{2.14}$$

$$\pi' = \mathtt{H}^{-T} \pi \tag{2.15}$$

It is important to keep in mind this last result, since it will be the basis of the registration of partial reconstructions algorithm in Chapter 4.

Line representation: representing lines in 3-space is something awkward to do since it is a primitive with four degrees of freedom. Its natural representation should be a 5-vector, which is not compatible with the formulation of points and planes. Hence, several approaches have been proposed in literature. Plucker Matrices/Vectors is the most accepted one.
Given two points $\mathbf{X}$ and $\mathbf{Y}$, the line joining them is represented by the 4×4 skew-symmetric Plucker matrix:

$$\mathtt{L} = \mathbf{X}\mathbf{Y}^T - \mathbf{Y}\mathbf{X}^T, \tag{2.16}$$

whose elements are $l_{ij} = X_i Y_j - Y_i X_j$. Since $\mathtt{L}$ is skew-symmetric, it is usually represented by the Plucker vector:

$$\mathbf{l} = \{l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}\} \tag{2.17}$$

Moreover, a dual Plucker representation $\mathtt{L}^*$ is obtained for a line formed by the intersection of two planes π and δ:

$$\mathtt{L}^* = \pi\delta^T - \delta\pi^T, \tag{2.18}$$

$\mathtt{L}$ and $\mathtt{L}^*$ are related through their elements by the rewrite rule:

$$l_{12} : l_{13} : l_{14} : l_{23} : l_{24} : l_{34} := l^*_{34} : l^*_{42} : l^*_{23} : l^*_{14} : l^*_{13} : l^*_{12}$$

Finally, given a projective transformation $\mathtt{H}$, the rule for transforming lines is:

$$\mathtt{L}' = \mathtt{H}\mathtt{L}\mathtt{H}^T \tag{2.19}$$

Hierarchy of transformations. The concept of transformation for the 3D Projective space is analogous to that of the 2D Projective plane. A Projective transformation $\mathtt{H}$ is now a 4×4 linear mapping between 4-vector primitives. The expressions for the Euclidean, Similarity, Affine and Projective specializations are exactly the same than for the 2D case. However, we need to take into account that, for this cases, $\mathtt{R}$, $\mathtt{A}$ and $\mathbf{t}$ refer to three-dimensional rotation, deformation and translation respectively.

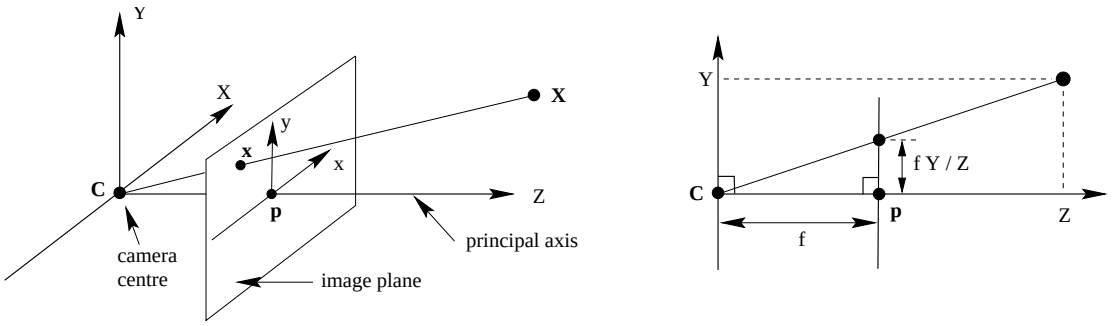


FIGURE 2.2: Structure of a pinhole camera. It consists on a camera center **C** and an image plane in front of it. The principal point is denoted as **p**.(Reprint from [1])

This section has presented the basic mathematical formulation of points, planes and lines in the Projective space. Moreover, the different transformations that can be applied to this type of primitives have been introduced. In chapter 4, the linear algorithms presented will rely on several concepts presented here. Now, let this mathematical aspects serve to introduce the pinhole camera model.

2.2 The pinhole camera

The pinhole camera model is the simplest mathematical representation of perspective camera. It represents a good approximation to the behaviour for most of real devices. A simple pinhole camera consists on a projection centre **C** and a principal plane. Nevertheless, the model can be furtherly improved by taking non-linear effects into account as, for instance, radial distorsion. In figure **??** the basic structure for this type of camera is detailed.

From the mathematical point of view, a pinhole camera is modeled by its projection matrix $\mathtt{P}$. The general projection model transfers a 3D point $\mathbf{X}$ to an image point $\mathbf{x}$ following the relation $\mathbf{x} = \lambda \mathtt{P}\mathbf{X}$. The parameter λ stands for an *up to scale umbiguity*.

The projection matrix $\mathtt{P}$ is factorized as follows:

$$\mathtt{P} = \mathtt{K}\,[\mathtt{R}|\mathbf{t}]\,, \tag{2.20}$$

where $\mathtt{K}$ and $[\mathtt{R}|\mathbf{t}]$ are respectively the intrinsics and extrinsic parameters of the camera. Nevertheless, in order to clear concepts up, (2.20) will be derived from the 3 components relating a 3D point with its corresponding 2D point. In Fig. 2.3 a graphical summarization of the pinhole camera projection model is presented.

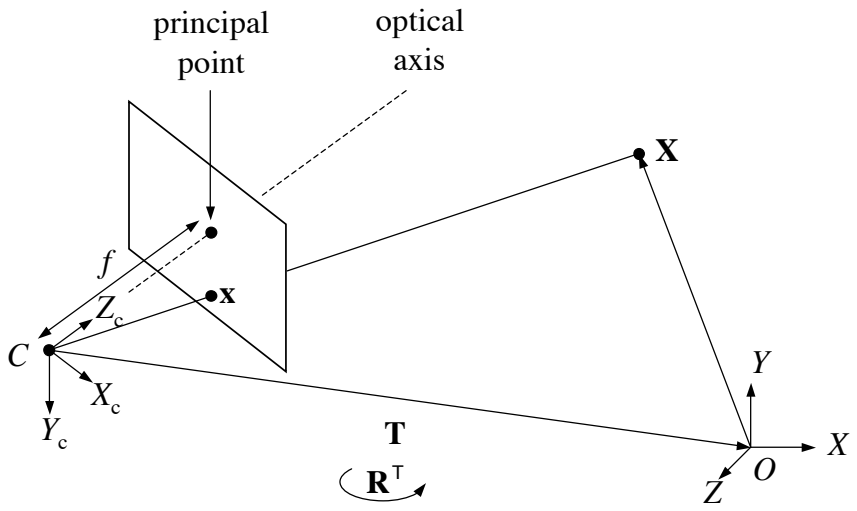


FIGURE 2.3: The projection matrix P describing the intrinsic and extrinsic properties of a pinhole camera is composed of three steps: a rigid transformation from world coordinates to camera coordinates, a perspective transformation and a 2D homography relating points in the image plane to pixels $\mathbf{x}$.(Reprint from [1])

2.2.1 Camera rotation and translation

Rotation and translation are usually referred as the *rigid-body transformation*. It consists on a change of linear projective basis from world coordinates to camera centre coordinates. That is transforming point $\mathbf{X} = [X\ Y\ Z\ 1]^T$, in the reference world frame, to $\mathbf{X}_c = [X_c\ Y_c\ Z_c\ 1]^T$. This relation can be compacted with the following expression:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.21}$$

where $\mathtt{R}$ is a 3×3 rotation matrix describing camera orientation and $\mathbf{t}$ refers to the translation of the camera centre with respect to the world origin $\mathbf{O}$. $\mathtt{R}$ and $\mathbf{t}$ are denoted the extrinsic parameters of the camera.

2.2.2 Perspective transformation

The second component is related to the perspective transformation from 3D points $\mathbf{X}_c = [X_c\ Y_c\ Z_c\ 1]^T$ to 2D points $\mathbf{u} = \lambda[u\ v\ 1]^T$ on the camera image plane. From

the schematic representation of the camera in Fig. 2.3 and by using similar triangles, we arrive to the expression:

$$u = f\frac{X_c}{Z_c} \; v = f\frac{Y_c}{Z_c} \tag{2.22}$$

The parameter f is the focal length. The consequence of varying its value is just an scaling over the image plane. Therefore if we set $f = 1$ we may obtain the expression for the perspective function defining the projectivity:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{2.23}$$

The point $\mathbf{u}$ is defined only up to scale since it is independent of the magnitude of $\mathbf{X}$ as a consequence of the Ray Space model (see Fig. 2.1).

2.2.3 Intrinsic parameters

Finally, the matrix of intrinsic parameters $\mathtt{K}$ relates a point $\mathbf{u}$ in the image plane to a pixel $\mathbf{x} = [x \; y \; 1]^T$ according to the focal length and other parameters such as skew or principal point:

$$\mathbf{x} = \lambda \mathtt{K}\mathbf{u} \tag{2.24}$$

The intrinsics matrix $\mathtt{K}$ is an upper triangular camera that contains the parameters related to camera calibration. $\mathtt{K}$ is defined as:

$$\mathtt{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

In (2.25), f_x and f_y refer to the focal length for both the dimensions x_c and y_c, s is the skew factor and $\mathbf{p} = [p_x \; p_y \; 1]^T$ represents the principal point of the pinhole camera. In the general case, pixels are assumed to be *squared* in which case $f_x = f_y = f$, $s = 0$ and the principal point is located in the center of the image.
As a conclusion, when combining equations (2.21), (2.23) and (2.25) the expression for the projection matrix in (2.20) is obtained:

$$\mathbf{x} = \lambda \mathtt{K}[\mathtt{R}|\mathbf{t}]\mathbf{X} = \lambda \mathtt{P}\mathbf{X} \tag{2.26}$$

From the point of view of Projective geometry, we may summarize the derivation of camera projection matrix as the concatenation of three factors: a 3D space homography (extrinsics), a 3D to 2D perspective transformation and, finally, a 2D space homography (intrinsics). Now, let us study the geometrical relationships beween a pair of cameras, which is on the basis of any Structure and Motion system.

2.3 Epipolar geometry

Epipolar Geometry is defined as the intrisic Projective geometry between two views. It only depends on cameras internal parameters and its relative pose. Its origins are from 1913 when Kruppa [23] demonstrated that, given 5 common 3D points to two views, it is possible to obtain the relative rotation and translation between the two cameras.
Let us assume a point $\mathbf{X}$ in the 3D space is imaged to two views $\mathtt{P}$ and $\mathtt{P}'$ at the points $\mathbf{x}$ and $\mathbf{x}'$ as depicted in ??. The epipolar geometry between two cameras is coded within the following three primitives:

- **Epipoles**: are the points $\mathbf{e}$ and $\mathbf{e}'$ where the baseline (line joining $\mathbf{C}$ and $\mathbf{C}'$) intersect.
- **Epipolar plane**: is the plane π formed by the baseline an the 3D point $\mathbf{X}$.
- **Epipolar lines**: are the lines $\mathbf{l}$ and $\mathbf{l}'$ where the epipolar plane and the image planes intersect. In addition they are the lines connecting the points in the image plane $\mathbf{x}$ and $\mathbf{x}'$ with their corresponding epipoles.

The most relevant consequence of epipolar geometry is that a given point $\mathbf{x}$ has its correspondence in image $\mathtt{P}'$ along the epipolar line $\mathbf{l}'$. This is an important result for applications such as stereo correspondence for depth estimation, outlier discarding after feature matching, *etc.*

2.3.1 The fundamental matrix

The fundamental matrix $\mathtt{F}$ is defined as the algebraic representation of epipolar geometry. There are several geometrical and algebraic derivations for $\mathtt{F}$ and the reader is referred to [1] for a formal definition of them. Nevertheless, we may derive an expression for the fundamental matrix from the correspondence condition.

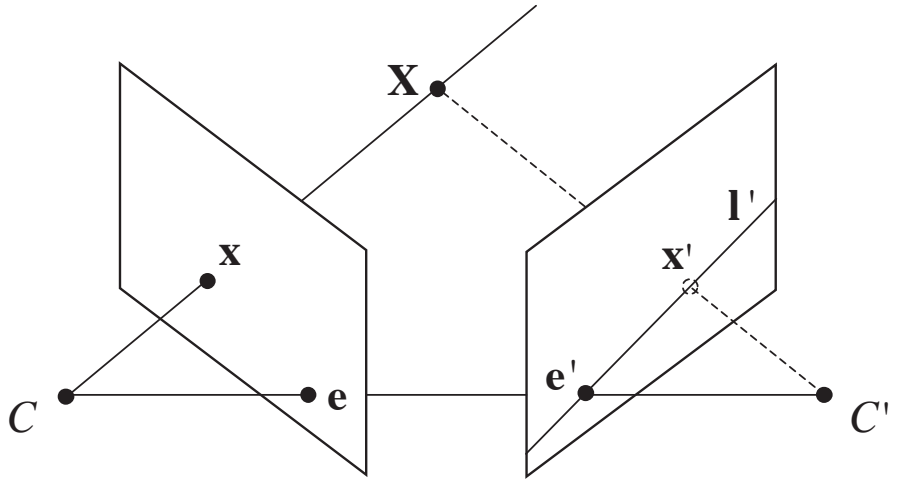


FIGURE 2.4: Epipolar geometry. Camera centres $\mathbf{C}$ and $\mathbf{C}'$, the 3D point $\mathbf{X}$ and its images $\mathbf{x}$ and x' lie in the same plane(Reprint from [1])

As mentioned, given the epipolar geometry of a pair of cameras $\mathtt{P}$ and $\mathtt{P}'$, a point in the first image has its correspondence along the epipolar line in the second image. Therefore, we could understand the fundamental matrix to be the mapping $\mathbf{l}' = \mathtt{F}\mathbf{x}$. From the incidence relation of 2.1 the point $\mathbf{x}'$ belongs to the line $\mathbf{l}'$ if and only if $\mathbf{x}'^T\mathbf{l}' = 0$. Hence, we have derived the main condition for the Fundamental matrix:

$$\mathbf{x}'^T\mathtt{F}\mathbf{x} = 0, \tag{2.27}$$

which is a 3×3 rank-2 matrix with 7 degrees of freedom.
Given a sufficient number (at least 8) of matching points $\{\mathbf{x}_i, \mathbf{x}'_i\}$ in two images, it is possible to derive a linear algorithm for determining the exact expression of $\mathtt{F}$. For a single correspondence equation 2.27 can be expanded to:

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \tag{2.28}$$

If we define the 9-vector $\mathbf{f}$ as the vector obtained from the entries of $\mathtt{F}$ in row-major order, we can express 2.28 as vector inner product:

$$(x'x, x'y, x', y'x, y'y, yx, y, 1)\mathbf{f} = 0 \tag{2.29}$$

Finally for a set of n correspondences the fundamental matrix can be obtained by solving the following problem:

$$\mathtt{A}\mathbf{f} = \begin{pmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y_1x_1 & y_1 & 1 \\ x'_2x_2 & x'_2y_2 & x'_2 & y'_2x_2 & y'_2y_2 & y_2x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y_nx_n & y_n & 1 \end{pmatrix} \mathbf{f} = 0 \tag{2.30}$$

The least squares solution for $\mathbf{f}$ of the overdetermined system is the right null vector of the measurement matrix $\mathtt{A}$ if 8 or more correspondences are available. This is the well known *8-point algorithm*. However, this is not enough for the habitual case. Correspondences are noisy an generally affected by outliers and therefore more robust strategies are required.

One solution consists on pre-conditioning the matrix $\mathtt{A}$ before solving the problem. The idea is to apply a non-isotropic scaling to the set of corresponding points such that the error is uniformly distributed among the three dimensions. That practice is known as the *Normalized 8-point algorithm*. Furthermore the presented linear algorithm may act as the kernel of a RANSAC [24] process in order to get rid of outliers.

Finally, the problem may be formulated in a optimization framework. The Maximum Likelihood Solution for $\mathtt{A}\mathbf{f} = 0$ minimizes the distance from transerred points to ther epipolar line. The sum of this set of distances form the *Point to Epipolar line* cost function. This kind of solution is commonly known as the *Gold Standard Algorithm* [1].

2.3.2 The Essential Matrix

The essential matrix $\mathtt{E}$ is a particularization of the fundamental matrix $\mathtt{F}$ for the case of normalized coordinates. A normalized projection matrix $\hat{\mathtt{P}}$ is obtained by the product $\hat{\mathtt{P}} = \mathtt{K}^{-1}[\mathtt{R}|\mathbf{t}] = [\mathtt{R}|\mathbf{t}]$. That implies cancelling the effect of the known intrinsic camera calibration. Assuming the projection model from 2.26 the normalized coordinates for an image point are $\hat{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x}$.

The equation defining the essential matrix for a set of normalized correspondences $\{\hat{\mathbf{x}}, \hat{\mathbf{x}}'\}$ is:

$$\hat{\mathbf{x}}'^T\mathtt{E}\mathbf{x} = 0 \tag{2.31}$$

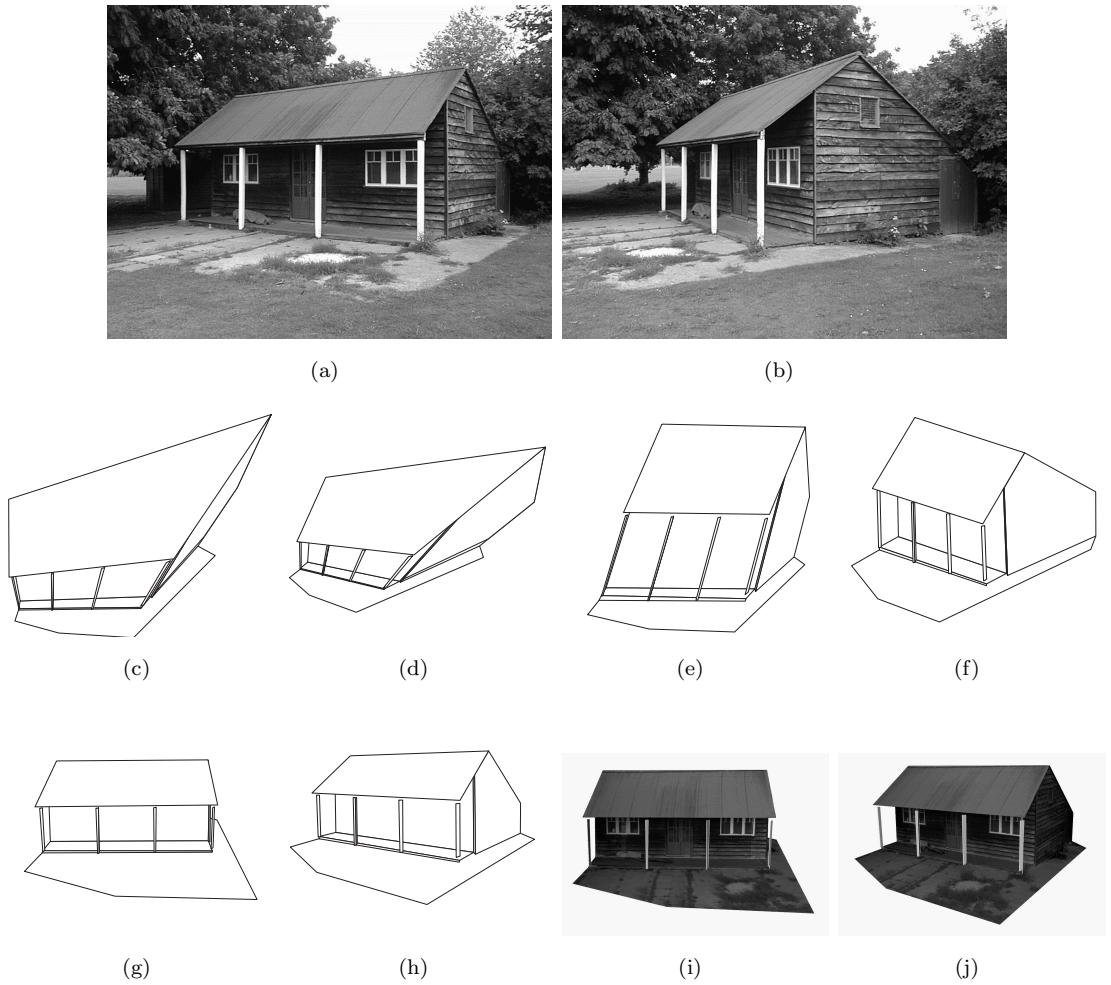


FIGURE 2.5: Illustration of the stratified approach for 3D reconstruction. (a-b) Input images for the 3D reconstruction system. (c-d)Projective reconstruction. (e-f) Affine reconstruction. (g-h) Metric reconstruction. (i-j) Photo-realistic reconstruction by means of planar texture mapping. (Reprint from [1])

Moreover, it is straightforward to derive its intrinsic relation with $\mathtt{F}$:

$$\mathtt{E} = \mathtt{K}'^{T}\mathtt{F}\mathtt{K} \tag{2.32}$$

As will be shown in the following point, the factorization of the Essential matrix plays a fundamental role when defining the relative pose estimation between two calibrated cameras.

2.4 Structure And Motion

Structure And Motion refers to the set of techniques that allows obtaining a 3D model of a recorded rigid scene, as well as the position of the camera(s) used for recording it. Let us focus in the simplest situation where we are provided with two snapshots of a single moving camera recording an static scenario. Let us also assume that no constraints about the acquiring device or the scene are known.
If enough correspondences between images $\{\mathbf{x}_i, \mathbf{x}'_i\}$ are provided, the reconstruction task consists on obtaining both the camera matrices $\{\mathtt{P}, \mathtt{P}'\}$ and the 3D points $\{\mathbf{X}^i, \mathbf{X}'^i\}$ that project to the 2D correspondences. Moreover, if the number of corresponding points is sufficient to compute a fundamental matrix $\mathtt{F}$, the scene may be reconstructed up to a projective ambiguity. The pipeline is the following:

1. **Fundamental Matrix**: Compute $\mathtt{F}$ from point correspondences.
2. **First pair estimation**: Compute $\mathtt{P}$ and $\mathtt{P}'$ from $\mathtt{F}$.
3. **Triangulation**: for each correspondence, obtain the associated 3D point $\mathtt{X}^i$.
4. **Resection**: if more cameras are available, their projection matrix may be retrieved from 2D/3D correspondences.
5. **Bundle Adjustment**: After each resection and its corresponding triangulation, the retrieved position of 3D points and cameras is refined via a non-linear global optimization [12].

However, if the retrieved SaM needs to feed, for instance, a visualization system, this projective ambiguity needs to be removed. That is, the scene has to be expressed in a metric or in an Euclidean coordinate system. If no constraints are assumed, the classical technique for obtaining the structure and the pose of cameras is the well known *stratified approach*. The scene is reconstructed in a projective frame and is progressively upgraded to be affine and, finally, metric. An schematic description is depicted in Fig. 2.5.
With this kind of approach, the ambiguity of the reconstructed scene is reduced as long as the scene is progressively upgraded. If we denote the desired Euclidean Reconstruction as $(\{\mathtt{P}_e, \mathtt{P}'_e\}, \mathbf{X}^i_e)$, the three steps of the stratified approach are the following:

- *Projective reconstruction*: the scene is defined up to a projective transformation $\mathtt{H}$ with 15 dof:

$$\mathtt{P}_e = \mathtt{P}\mathtt{H}^{-1}, \mathtt{P}'_e = \mathtt{P}'\mathtt{H}^{-1} \text{ and } \mathbf{X}^i_e = \mathtt{H}\mathbf{X}^i$$

- *Affine reconstruction*: the scene is defined up to an affine transformation $\mathtt{H}_a$ with 11 dof:

$$\mathtt{P}_e = \mathtt{P}_a\mathtt{H}_a^{-1}, \mathtt{P}'_e = \mathtt{P}'_a\mathtt{H}_a^{-1} \text{ and } \mathbf{X}^i_e = \mathtt{H}_a\mathbf{X}^i_a$$

- *Metric reconstruction*: the scene is defined up to a similarity transformation $\mathtt{H}_s$ with 7 dof:

$$\mathtt{P}_e = \mathtt{P}_m\mathtt{H}_s^{-T}, \mathtt{P}'_e = \mathtt{P}'_m\mathtt{H}_s^{-T} \text{ and } \mathbf{X}^i_e = \mathtt{H}_s\mathbf{X}^i_m$$

The information required for upgrading the scene at each step of the stratified approach, is gathered from constraints associated to both the cameras or the scene or, in other words, the self calibration process. The step from projective to affine reconstruction consists on determining the plane at infinity π and, next, mapping it to its assumed position in an affine frame $\pi_\infty = (0, 0, 0, 1)^T$. Furthermore, the step for upgrading to a metric consists on identifying the absolute conic Ω_∞ which is a planar conic located at the plane of infinity π_∞. Further details may be found in [1].

The internal parameters of the cameras are one of the aforementioned constraints to be exploited in the stratified approach. If they are known the scene can be upgraded to be metric since a metric reference is available $\mathtt{K}$. From the mathematical point of view, the knowledge of the internal parameters of cameras are closely related to ω, the image of the absolute conic (IAC) :

$$\omega = \mathtt{K}^{-T}\mathtt{K}^{-1} \tag{2.33}$$

In the following, the mathematical formulation for all the steps mentioned is presented. Note that we will be assuming that cameras are calibrated beforehand and, therefore, the scene may be directly reconstructed in a metric frame.

2.4.1 Feature point detectors

As presented along this chapter, the algorithms for obtaining the 3D reconstruction of a rigid scene are based on a set of sparse 2D feature points tracked through

images. The extracted feature points are desired to have some special characteristics in order to make them a reliable input data for the reconstruction system: repeatability, invariance to rotation and scaling and independence to illumination changes.

There are many approaches in literature based in different criteria to determine which characteristics are exploited to obtain repeatable points from images. Among them, Harris corners [25] and Scale-Space extrema detectors [21, 26, 27] are the most popular techniques.

Harris detector, searches for corners in the image based on the matrix:

$$\mathtt{A} = \sum_{x,y} w(x,y) \begin{pmatrix} L_x^2 & L_{xy} \\ L_{xy} & L_y^2 \end{pmatrix} \tag{2.34}$$

$w(x,y)$ refers to a circular Gaussian Kernel that weights the image samples by giving more relevance to the central ones. L_{ij} refer to the derivatives of the image along the axes. matrix $\mathtt{A}$ is evaluated in order to define whether a corner was found or not. With that objective the expression $R = det(\mathtt{A}) - \alpha tr^2(\mathtt{A})$, gives an idea of how likely the processed point will be a corner or not.

On the other we have the Scale-Space extrema detectors [21, 26, 27] that have become very popular in recent years due to its application to other fields like image retrieval or augmented reality. The stages for this type of feature extractors are the following:

1. **Scale-Space Extrema Detection**: Image is analyzed at different scales by means of a convolution with gaussian kernels of different standard deviation. For each scale, the points considered as local maxima both in space and scale are selected as candidates.

2. **Key Point Localization**: an stability measure studying the local curvature of the point candidates is applied in order to discard those points that are unstable.

3. **Orientation Assignment**: orientation is assigned to each interest points based on a local study of its gradient.

4. **Key Point descriptor**: a N-position vector is assigned to each Key Point. The vector is constructed with the values obtained during orientation assignment and gradient study. Therefore, this forces the feature points to be identified with invariance to rotation, translation, scale and illumination.

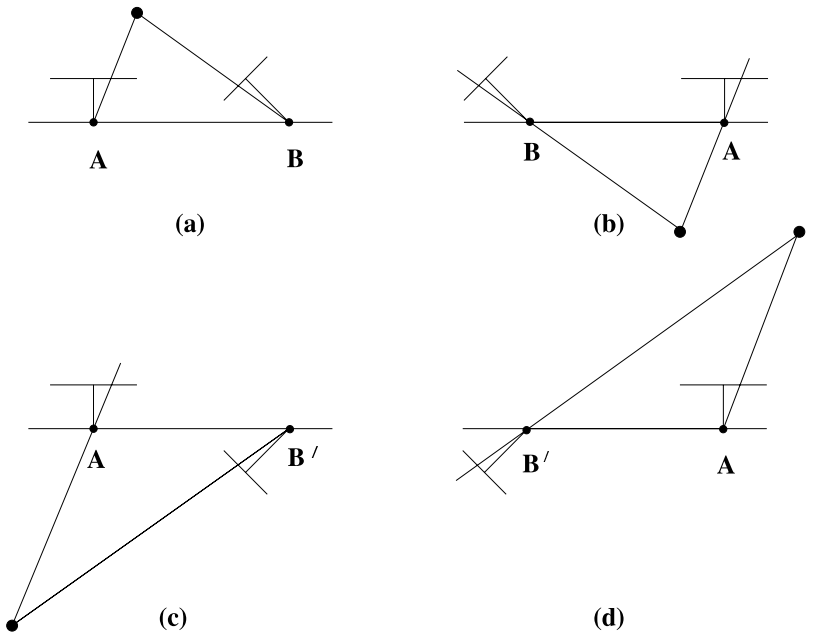


FIGURE 2.6: The four possible solution for the initial pose estimation. Only the solution that triangulates all the correspondences in front of both cameras is accepted.(Reprint from [1])

2.4.2 Two view relationship

Two view relationship is the starting point for a wide set of SaM pipelines [2]. When relative orientation and translation between two cameras is retrieved, an initial 3D point triangulation 2.4.3 can be performed. This serves as the anchor point for adding new cameras to the initial pair 2.4.4 and subsequently adding more points to the reconstruction. Therefore, the reader may notice that the quality and reliability of this initial estimation will determine the final overall system performance.

Given a set of correspondences $\{\mathbf{x}, \mathbf{x}'\}$ for two calibrated views, it is possible to determine their relative orientation up to an scale factor relative to the translation vector between views and a four-fold ambiguity as depicted in Fig. 2.6. This goal is achieved by means of factorizing the essential matrix $\mathtt{E}$ computed from the set of correspondences.

First of all, the normalized projection matrix for the first camera is fixed to be $\hat{\mathtt{P}} = [\mathtt{I}|\mathbf{0}]$ and $\hat{\mathtt{P}}' = [\mathtt{R}|\mathbf{t}]$ for the second one. From the known calibration $\mathtt{K}$ of the cameras, it is possible to compute essential matrix $\mathtt{E}$ from (2.31). Furthermore, Essential matrix may be factorized as $\mathtt{E} = [\mathbf{t}]_x\mathtt{R} = \mathtt{SR}$ where $\mathtt{S}$ is a skew-symmetric matrix. In [28] a method is presented for obtaining $\mathtt{R}$ and $\mathbf{t}$ from the SVD of $\mathtt{E}$

using the following matrices:

$$\mathtt{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \mathtt{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

Let us assume that the SVD of $\mathtt{E}$ is $\mathtt{U}\text{diag}(1,1,0)\mathtt{V}^T$. The aforementioned matrices $\mathtt{S}$ and $\mathtt{R}$ are then $\mathtt{S} = \mathtt{U}\mathtt{Z}\mathtt{U}^T$ and $\mathtt{R} = \mathtt{U}\mathtt{W}\mathtt{V}^T$. Finally the four possible solutions for matrix $\hat{\mathtt{P}}' = [\mathtt{R}|\mathbf{t}]$ are:

$$\hat{\mathtt{P}}' = [\mathtt{U}\mathtt{W}\mathtt{V}^T|\mathbf{u}_3],\ \hat{\mathtt{P}}' = [\mathtt{U}\mathtt{W}\mathtt{V}^T|-\mathbf{u}_3],\ \hat{\mathtt{P}}' = [\mathtt{U}\mathtt{W}^T\mathtt{V}^T|\mathbf{u}_3],\ \hat{\mathtt{P}}' = [\mathtt{U}\mathtt{W}^T\mathtt{V}^T|-\mathbf{u}_3], \quad (2.35)$$

where $\mathbf{u}_3$ is the last column of $\mathtt{U}$ from SVD($\mathtt{E}$). The last step consists on selecting which of the four solutions is the correct one. The selection criterium is to choose the solution such that its triangulated points are all in front of both cameras. In Fig. 2.6 is clear that only one of them will satisfy that condition. This is commonly known as the *cheirality constraint*.

2.4.3 Structure computation

Structure computation deals with the process of obtaining the position of a 3D point $\mathbf{X}$ given a its images $\{\mathbf{x}\}^j$ along a set of views $\{\mathtt{P}\}^j$ that is also provided. Let us assume that the point $\mathbf{X}$ projects to a pair of images following the projection model from (2.26):

$$\mathbf{x} = \lambda\mathtt{P}\mathbf{X};\ \mathbf{x}' = \lambda\mathtt{P}'\mathbf{X};$$

The point $\mathbf{X}$ is common for both cameras $\mathtt{P}$ and $\mathtt{P}'$. Nevertheless, detected points in images are contamined with noise and therefore it does not exist an exact solution for $\mathbf{X}$. Its most likely position needs to be estimated $\hat{\mathbf{X}}$. In Fig. 2.7 this effect is ilustrated: the estimated 3D point $\hat{\mathbf{X}}$ backprojects to the pair of views with some error. That is the well know *reprojection error*.

$$\epsilon = d(\mathbf{x}, \hat{\mathbf{x}}') + d(\mathbf{x}, \hat{\mathbf{x}}') = d(\mathbf{x}', \mathtt{P}\hat{\mathbf{X}}) + d(\mathbf{x}', \mathtt{P}'\hat{\mathbf{X}}), \quad (2.36)$$

where $d(\mathbf{a}, \mathbf{b})$ stands for the 2D Euclidean distance between two vectors. This measure asses the quality of the triangulation and will be used in other algorithms in the SaM pipeline.

As it was done for the Fundamental matrix 2.3.1, the objective is to express the projection equations from (2.36) as a linear homogeneous system $\mathbf{A}\mathbf{X} = \mathbf{0}$. By

removing the homogeneous scale factor, the projection of the first camera may be expressed by means of the cross product $\mathbf{x} \times \mathtt{P}\mathbf{X}$. With this type of expression, each 2D point and its correspondent camera matrix provide 3 linear equations in the entries of $\mathbf{X}$:

$$\begin{aligned} x(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) = 0, \\ y(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) = 0, \\ x(\mathbf{p}^{2T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) = 0, \end{aligned}$$

where $\mathbf{p}^{iT}$ refers to the $i{-}th$ row of projection matrix $\mathtt{P}$. This set of linear equations may be expressed on the form $\mathtt{A}\mathbf{X} = 0$, where the measurement matrix $\mathtt{A}$ follows:

$$\mathtt{A} = \begin{bmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{bmatrix} \tag{2.37}$$

As seen, each camera contributes with two linear independent equations. The benefit is that the method for computing matrix $\mathtt{A}$ is valid for an unlimited number of projections, since it just consists on stacking the two equations that each camera provides. Again, as for the Fundamental matrix case, the system (2.37) may be solved by means of Least Squares. Moreover the problem may be casted to the optimization framework. As any optimization problem, it consists on a minimization of some cost function (*i.e* reprojection error from (2.36)) subject to some constraint: the Fundamental matrix condition $\mathbf{x}'^{T}\mathtt{F}\mathbf{x} = 0$.
In section 4.5 in chapter 4, a deeper discussion about the different possible cost functions to be minimized can be found.

2.4.4 Camera resectioning methods

Now the objective is, given a set of 3D points $\mathbf{X}_i$ and their 2D projections $\mathbf{x}_i$ in one view, obtain the projection matrix $\mathtt{P}$ of that camera. Once again, the objective is to develop a linear algorithm such that the problem may be formulated as $\mathtt{A}\mathbf{p} = \mathbf{0}$ similar to the triangulation or fundamental matrix cases. $\mathtt{A}$ corresponds to a measurement matrix computed from the provided 3D/2D correspondences and $\mathbf{p}$ to a 12-vector containing the entries of the camera matrix $\mathtt{P}$.

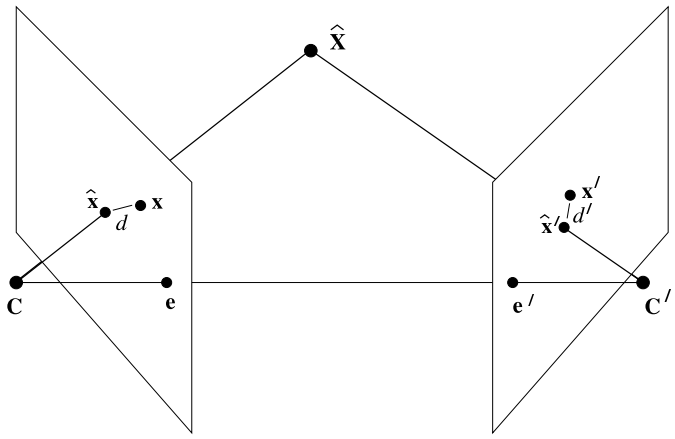


FIGURE 2.7: Point triangulation. There not exist an exact solution on the position of $\mathbf{X}$ since its detected images are corrupted by noise. Therefore its value is estimated $\hat{\mathbf{X}}$ and the quality of the estimation is given by the reprojection error. (Reprint from [1])

Let us assume that $\mathbf{x}_i = [x_i\ y_i\ z_i]^T$ and that the projection equation may be expressed as the cross product $\mathbf{x_i} \times \mathtt{P}\mathbf{X}_i = \mathbf{0}$ as it was done for the triangulation case. Therefore, for each 3D/2D correspondence, we may derive the following relationship:

$$\begin{bmatrix} \mathbf{0}^T & -z_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ z_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \mathbf{0}, \tag{2.38}$$

that provides 3 *linearly dependent* equations. Therefore each correspondence provides two independent equations:

$$\begin{bmatrix} \mathbf{0}^T & -z_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ z_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \mathtt{A}_i\mathbf{p} = \mathbf{0}, \tag{2.39}$$

where $\mathtt{A}_i$ represents the measurement matrix associated to a single correspondece. $\mathbf{p}_j$ is the column vector containing the entries of the $j-th$ row from projection matrix $\mathtt{P}$. In order to derive a linear algorithm for $\mathtt{P}$, one may stack the measurement matrices $\mathtt{A}_i$ in order to obtain a $2n \times 12$ matrix $\mathtt{A}$. Therefore the projection matrix $\mathtt{P}$ is computed by solving the set of equations $\mathtt{A}\mathbf{p} = \mathbf{0}$.

Again, the measurement matrix may be pre-conditioned by scaling the set of both 2D and 3D points in order to better conditionate the problem and the proposed algorithm act as the kernel of a RANSAC process. Finally once RANSAC has classified 2D/3D correspondences as inlier/outliers, the solution may be refined

from an optimization framework.

Since both 3D and 2D points are noisy, the solution for $\mathtt{P}$ will not be exact. Therefore the backprojection of $\mathbf{X}$ through the estimated camera $\hat{\mathtt{P}}$ will contain some error as for the triangulation case. The accumulation of this error for all 2D/3D correspondences determines the cost function to be minimized during the refinement step.

$$C(\mathtt{P}) = \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i) = \sum_i d(\mathbf{x}_i, \hat{\mathtt{P}}\mathbf{X}_i) \tag{2.40}$$

If the measurement errors are assumed to be Gaussian, the MLE for $\mathtt{P}$ will be:

$$\mathtt{P}_{ML} = \underset{\mathtt{P}}{\operatorname{argmin}} \; C(\mathtt{P}) \tag{2.41}$$

This non-linear minimization requires the use of iterative techniques, such as Levenberg-Marquadt [12].

Up to this point, the basic background for both Projective geometry and Structure And Motion has been presented. In the next chapters we will refer to several of the concepts introduced here. More precisely, chapter 3 presents an overview of the full reconstruction pipeline. Therefore concepts from section 2.4 such as first pair estimation, triangulation, resection, *etc.* will be constantly referred. On the other hand, the algorithms presented in chapter **??**, take profit of the Point/Plane Duality Theorem and, hence, they rely in several concepts of Projective geometry.

Chapter 3

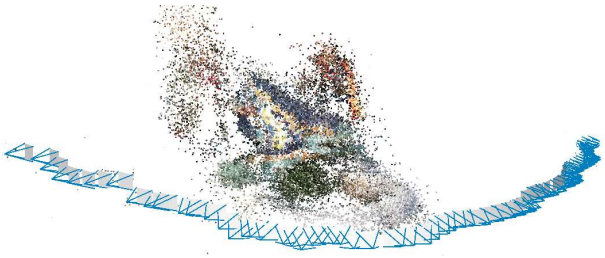
Structure and Motion for Video Sequences. System overview

The objective of this chapter is to present the adopted strategies for obtaining 3D models from recorded videos of static scenes as depicted in 3.1. We aim to develop a system such that users can record a rigid scene and upload the video to a server where, after some processing, a 3D reconstruction of the recorded scene is returned.
In order to develop a robust system, a large set of algorithms needed to be developed in order to face the associated issues to this kind of systems as they were mentioned in chapter 1. The result is a robust pipeline based in a *divide an conquer* approach that is able to deal with different types of sequences or qualities of the acquiring device, while efficiently making use of computational resources.
In the following, the key-steps of the reconstruction pipeline are presented.

3.1 System Architecture

A global description of the proposed SaM pipeline is depicted in Fig. 3.2. The first step consist on detecting SIFT-like features for all the input frames of the video sequence. This process represents one of the main bottlenecks in all SaM systems although the use of modern GPU's may speed it up.
The next step consists on removing the temporal redundancy from the video stream through an smart frame decimation process. Apart from discarding redundant frames, the decimation scheme provides the SaM block with a set of well-conditioned Key-Frames (KF's) up to some criteria of relative motion and number of correspondences between images.

(a) Snapshots



(b) SaM

(c) Densification

FIGURE 3.1: Example of workflow for our Structure and motion system. a) Input data is a video sequence recording a rigid scene. b) The scene is reconstructed as cloud of points. c) The cloud of points may be furtherly improved through a process of densification.

After that, the input video sequence is splitted into subsequences in order to allow a parallel processing that dramatically reduces the computational complexity of the system. Each one of the video clips and their associated KF's are the input to a SaM and Feature tracking block. As it will be explained, the process of feature tracking is embeddded in the reconstruction loop as a difference with other *state of the art* approaches [2, 5].

Finally, a merge step registers all the partial 3D reconstructions obtained from SaM loops into a common coordinate frame. This stage is mandatory since the retrieved partial reconstructions are referenced to an arbitrary coordinate system although they represent the same static scene. An additional stage may use the retrieved pose of the cameras to provide a dense representation of the rigid scene instead of a point cloud.

3.2 Selection of relevant frames

In this section we analyze the first stage of the SaM pipeline. Initially, the system is fed with an input video sequence where a rigid scene was recorded. The objective of this step is to select and isolate the relevant information from the video stream for both the spatial and temporal dimensions. The large amount of pixels from each image is reduced to a smaller set of detected SIFT-like points and a frame

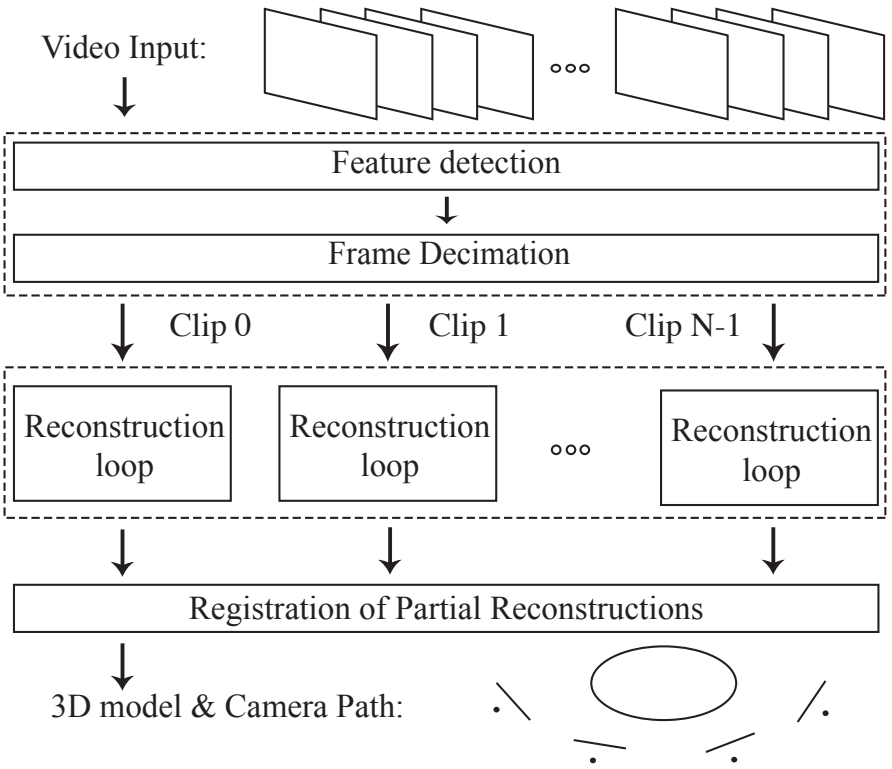


FIGURE 3.2: Block diagram of the full SaM system. As seen a *divide an conquer approach* is followed by means of a parallelization of the reconstruction loop

decimation step selects those frames that may be relevant for the reconstruction loop.

3.2.1 Feature detection and matching

The system may work with three different feature point extractors: SIFT [21], SURF [26] and DART [27]. All of them are classified as *space-scale extrema* detectors and therefore they share some similar characteristics as, for instance, that they all are rotation- and scale-invariant. In fact, the processes of detection and matching may be isolated, since the matching implementation is common for the three types of detectors although not compatible between them. This is possible due to the fact that they all assign a N-position descriptor array to each detected feature whose typical values for N are 64 or 128.
From the implementation point of view, our system defines a data type for the detected features, with the following information:

- $I(x, y)$: 2D coordinates in the image plane of the detected feature with subpixel precision.

- $C(r, g, b, a)$: color coordinates of the feature.
- *Track ID*: label associated to the track that the feature belongs to.
- *Frame ID*: frame index of the image where the feature was detected.
- *Descriptor*: 64 or 128 positions vector describing the spatial neighborhood of the feature.

A data structure of this type is mandatory since feature points neeed to identified along the global scene for both SaM and Visualization processes.
The feature matching process follows the *kd-tree*-based approach from [21] for retrieving the nearest neighbor of a SIFT-like descriptor. When comparing two images the feature descriptors of one of them are hierarchly stored into a kd-tree. The feature descriptors from the remaining image act as queries. The leaf whose descriptor presents the lowest euclidean distance d_0 is selected as the corresponding feature if and only if, this distance is $d_0 < 0.6d_1$. d_1 represents the Euclidean distance with respect to the second NN. This condition is crucial since there will always be a NN. Furthermore, it is important to note that the use of kd-trees reduces the complexity of the matching from $O(n^2)$ to $O(nlogn)$.
During the feature matching process some of the feature correspondences may be uncorrectly matched. Therefore, raw feature correspondences obtained in an initial feature matching are used to feed a RANSAC algorithm that computes a fundamental matrix `F` between frames. This matrix is used to classify each putative match as outlier/inlier [1]. The threshold is set to 1 pixel of *point to epipolar line* cost.

3.2.2 Frame Decimation

When dealing with video sequences for SaM systems, the usefulness of a pre analysis or a frame decimation step raises due to two main reasons. First, it reduces the temporal redundancy natural from video sequences by providing a reduced frame set that should speed up the reconstruction process. Second, it feeds the SaM system with a well conditioned set of Key Frames (KF's) in order to make a reliable reconstruction possible.
An straightforward frame decimation scheme could consist on simply reducing the frame rate that the camera provides. However, this blind downsampling is not adequate according to the desired features for the KF's . Some of them are listed in [11]

- KF's need to be sharp, that is not affected by motion blurring or auto-focus artifacts.

- Baseline and parallax between frames has to be large enough to allow a good conditioning of the two-frame epipolar geometry.

- A sufficient number of point correspondences $\{\mathbf{x}_i, \mathbf{x}'_i\}$ between pairs of frames has to be available.

Therefore, a blind and uniform downsampling does not take into account all these desired features. If a high frame rate is selected, we are introducing redundancy in the system and, therefore, compromising a large amount of computational resources. Moreover, the baseline between frames is not ensured to be appropiate and, hence, it could cause and ill-conditioning of the mathematical problem. On the other hand, if the frame rate is selected to be low, the connectivity between frames can not be guaranteed and thus lead to the introduction of *cuts* along the sequence. Finally, the algorithm is desired to be *idempotent*.
We propose a new frame decimation algorithm inspired by [11] in order to fulfill all the required and listed conditions for KF's. For our case, the algorithm is *feature-based* and therefore different conditions had to be defined in order to derive some criteria for determining the redundancy of each single frame.
The algorithm pipeline is as follows: frames I_i from the input sequence I_n are sorted into a list F in order of increasing number of detected feature points. The set of feature points associated to frame I_i is denoted as F_i. Since the type of detected features are scale-space extrema 2.4.1, we may assume that *focused* or *sharp* frames are the ones with larger number of the features. It must be noted that this assertion only makes sense when the number of feature points for a frame is compared with respect to its neighbors in a small time interval.
Next, frames I_i from F are sequentially processed in order to evaluate if they are redundant or not and, if so, they are removed from the input sequence I_n. The condition to determine if I_i is redundant, consists on evaluating the image affinity between its two neighbors I_{i-1} and I_{i+1}. Recall from the KF's conditions that enough baseline between frames is mandatory but also a sufficiently large number of common features. This condition translates to a relative affinity between images bounded from above (frames need to be relevantly different) subject to some constraints in the number of features and in the apparent 2D motion.

Let us define the image affinity measure as the Jaccard index, which represents the similarity between sample sets:

$$a_i = \frac{|F_{i-1} \cap F_{i+1}|}{|F_{i-1} \cup F_{i+1}|}, \tag{3.1}$$

where operator $|\bullet|$ denotes number of elements in a set. The aparent motion m_i is obtained as the median of 2D motion for each feature correspondence:

$$m_i = \underset{j}{\text{median}} ||\mathbf{x}_j^{i-1} - \mathtt{H}\mathbf{x}_j^{i+1}||, \tag{3.2}$$

where $\mathtt{H}$ refers to an homography relating both frames. In our experiments, I_i is considered to be redundant if its neighbors I_{i-1} and I_{i+1} fullfil the following conditions:

$$I_i \begin{cases} a_i > 0.25 \\ m_i < 10\% \text{ of Im. Diag.} \\ |F_{i-1} \cap F_{i+1}| \geq 100 \end{cases} \tag{3.3}$$

Frames are processed in the same order than they are stored in F and multiple passes could be required. The algorithm stops when no frame is discarded after a pass. With this scheme, non-sharp frames are the first to be removed since they are the first to be evaluated. The frame decimation algorithm adapts to the motion of the camera with respect to the scene and outputs a set of frames whose relative motion is more isotropic (see results in chapter 5).

3.3 Robust Multi-View Structure and Motion

3.3.1 Feature tracking

A 3D point $\mathbf{X}^j$ may be visible to a certain set of cameras $\mathtt{P}_i$ and consequently produce a set projections $\mathbf{x}_j^i$. A usual practice consists on grouping this set of projections into tracks or, in other words, assigning the same tag to each one of them.

One of the major issues in SaM is to feed the reconstruction loop with a good set of features tracked along time. Traditionally, feature matching and tracking has been carried out prior to the reconstruction loop and therefore using no available 3D information [2]. In our work, the process of asigning feature points to existing or new tracks is interlaced within the SaM loop. This practice robustifies the feature tracking process since it combines 2D and 3D information.

Algorithm 1 Estimate the best conditioned pair for the initial pose estimation

$\rho_H^{opt} = 0$; $\mathtt{P}_0 =$ NULL; $\mathtt{P}_1 =$ NULL;
for $i = 0$ to Number of KFs **do**
  for $j = i$ to Number of KFs **do**
    $\rho_H(i,j)$=featureMatching(I_i, I_j);
    if $\rho_H(i,j) \geq 0.5$ **then**
      $(\mathtt{P}_i, \mathtt{P}_j)$=estimateInitialPair$(F_i, F_j)$;
      T=triangulate3Dpoints$(\mathtt{P}_i, \mathtt{P}_j)$;
      D=discardNonConsistent3Dpoints$(\mathtt{P}_i, \mathtt{P}_j, \mathbf{X})$;
      if $\rho_H(i,j) \geq \rho_H^{opt}$ and $T - D > 100$ **then**
        $\rho_H^{opt} = \rho_H(i,j)$;
        $\mathtt{P}_0 = \mathtt{P}_i$;
        $\mathtt{P}_1 = \mathtt{P}_j$;
      else
        continue
      end if
    else
      continue
    end if
  end for
end for

3.3.2 Initial-pair estimation

Selecting a good initial pair for starting the process of reconstruction is a crucial choice that will determine the final overall performance. Thus this step needs to be as robust as possible. Since the input for the SaM block is a small set of KF's selected from a short clip (20KFs), an exhaustive search may be carried out as explained in Algorithm 1.
Nevertheless, there are some requirements for the initial pair that need to be fulfilled:

- The epipolar geometry needs to be satisfied, that is well modelled by a fundamental matrix $\mathtt{F}$.

- The selected pair of frames can not configurate a degenerate case [1].

- There must be enough correspondences (at least 8 according to [1] but we fixed it to be over 100).

With the listed requirements, a quality measure may be defined both for selecting the best pair for the initial pose estimation and for pre-discarding bad conditioned pairs, which speeds up the full search.

For each tested pair (let us assume I_i and I_k), a feature matching is performed and correspondences are used for robustly computing both $\mathtt{F}$ and $\mathtt{H}$ through a RANSAC process. In our case, we measure the good conditioning of the initial pair as the ratio of outliers ρ_H when trying to model the pair of images with an homography $\mathtt{H}$:

$$\rho_H(i,k) = 1 - \frac{|F_i \cap F_k|_H}{|F_i \cap F_k|}, \tag{3.4}$$

where $|F_i \cap F_k|_H$ refers for the total matches considered as inliers for the homography $\mathtt{H}$ case. The threshold for classifying a correspondence as inlier/outlier to an homography is of 1 pixel of 2D distance $m_{ik}^j = ||\mathbf{x}_j^i - \mathtt{H}\mathbf{x}_j^k||$. Pairs with $\rho_h < 0.5$ are considered to be ill-conditioned and therefore not considered in further steps. For the rest of the cases, the projection matrices of the pair are computed.

The algorithm for estimating the relative position between a pair of cameras is the one that was explained in Section 2.4.2. It consists on fixing one of the cameras to be placed at the origin $\mathtt{P}_i = \mathtt{K}[\mathtt{I}|\mathbf{0}]$ and estimating the relative rotation and translation of the other one $\mathtt{P}_k = \mathtt{K}[\mathtt{R}|\mathbf{t}]$. This is achieved by means of factorizing the Essential matrix $\mathtt{E} = \mathtt{K}^T\mathtt{F}\mathtt{K}$ as in [1].

Once the projection matrices are retrieved, the process is followed by the linear triangulation step in 2.4.3. Obtained structure from triangulation is further refined by means of discarding those points with large uncertainity (angle less than 1° degree) and/or reprojection error above 1 pixel). The pair of cameras $\{\mathtt{P}_i, \mathtt{P}_k\}$ with the biggest score $\rho_{\mathtt{H}}(i,j)$ subject to a number of valid 3D points after triangulation greater than 100, is selected as the initial pair.

After this purge step, the process of feature tracking starts. In this case feature tracking is straightforward. Each pair of matched features are assigned into a common track since 3D points have been triangulated for the very first time.

3.3.3 Updating Structure and Motion

After initial pair estimation, the system enters the loop of progressively adding new cameras and triangulating more points. Candidate projection matrices for cameras to be incorporated are estimated from 2D/3D correspondences inside a RANSAC process and refined via a Gold Standard Resection algorithm [1]. For the structure computation, the linear triangulation algorithm from 2.4.3 is used.

Camera resection. Once again, since the number of KF's selected from each video clip is small, we may exhaustively search which the best camera is to be

added at each iteration in the update loop. In our system, resection for each new camera is performed with respect to other already resected camera. That implies estimating the projection matrix just by means of the common 2D/3D points shared with another already resected view. According to our experience this relative resection, robustifies the process of SaM, since the linear process of determining the projection matrix is best conditioned.
Candidate cameras $\mathtt{P}_i$ to be resected are stored to a list LP in order of decreasing shared 2D/3D points with respect to already resected cameras. Not all combinations are stored since, at least, 3 2D/3D points may be visible for the candidate view. Nevertheless we make the threshold more restrictive and candidate cameras are not considered if they not share at least 20 2D/3D points with respect to already resected views. A new candidate is accepted if its reprojection error after resection is below 5 pixel. Otherwise, the next element inside LP list is processed.
Structure computation As before, linear triangulation and recursive feature tracking to already existing cameras follows resectioning. As mentioned in 3.3.2, some constraints relating to reprojection error and 3D position uncertainity are introduced to evaluate the consistency of triangulated 3D points. Those considered to be non consistent are discarded. The whole pipeline ends with an optional Bundle Adjustment (BA) to improve the consistency of the SaM updating. The implementation selected for this task was the one from [12]. It is a very efficient implementation since it takes profit from the sparseness of the measurement matrices involved in the optimization processes for SaM. That is the well-known technique SBA (Sparse Bundle Adjustment).

At this step of the SaM pipeline, several possible situations need to be checked for each pair of matched features in order to ensure a reliable feature tracking. Given two matched features $\{\mathbf{x}_i^j, \mathbf{x}_k^r\}$ their superindices j and r refer to the track they are assigned to. Several situations are ro be considered:

- If their value is not assigned yet, a new track is created and then $j = r$.
- If j was assigned but not r, $\mathbf{x}_k^r$ is assigned to track j and the other way round.
- Both features are matched but they belong to different tracks, $j \neq r$. In this case, the possibility of merging tracks is studied and, if it is not possible, both of them are deleted for the sake of safety.

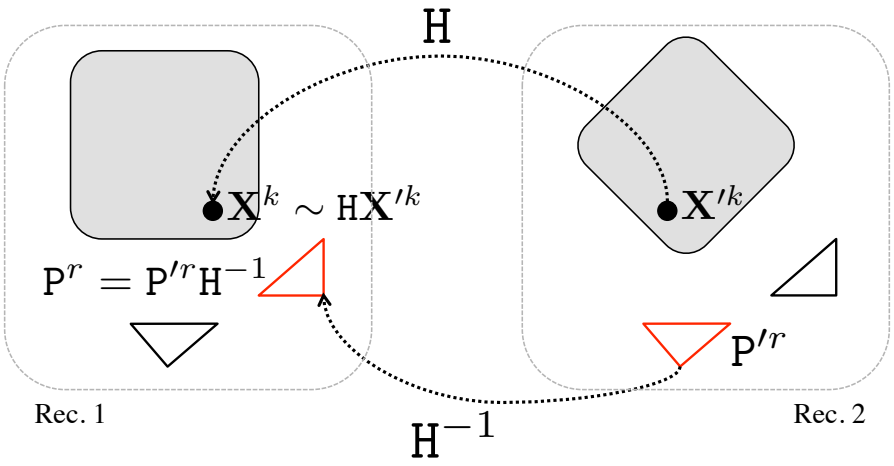


FIGURE 3.3: Graphical explanation of the registration of partial reconstructions step. The triangles in red refer to overlapping cameras

The condition for merging both tracks is evaluated by checking if there exist a resected view which contains a pair of features with the tracks j and r. If not both tracks are considered to be same and, consequently, they are merged.

3.3.4 Registration of partial reconstructions

As presented in 2.4.2 the process of relative pose estimation for the first pair is not constrained to be performed in a concrete coordinate system. Therefore, each one of the metric reconstructions of clips will differ in rotation, translation and an scale factor although they represent the same static scene. Hence, a step registering all of them into the same metric coordinate system is necessary. An intuitive explanation is depicted in Fig. **??**.

Although the problem will be properly formulated in the Chapter 4, let us give some ideas about it, in order to introduce the tecnical issues associated to the algorithm. The key idea of the registration technique consists on, given to independent reconstructions, exploit their 3D correspondences to derive a similarity transform $\mathtt{H}_\mathtt{s}$ that uniquely relates both of them. The correspondences are assumed to be common 3D points and overlapped cameras.

In our implementation, the clips are selected in such a way that they share at least 10 Key Frames. Therefore, the overlapped cameras are known beforehand and no process of *camera matching* is needed. For the case of points, the matching is a little more *tricky*. As a difference with SIFT-like features, triangulated points are not provided with a descriptor that uniquely identifies them. Hence, the matching must be performed in the 2D space.

An exhaustive matching is carried out between clips and all the cameras are pair by pair matched in order to select those common 2D points that have a 3D point associated. That is, they are grouped in some *track*. Once the point and camera correspondences are selected, they feed the algorithm from chapter 4 whose output is the desired similarity transformation $\mathtt{H}_S$.

Chapter 4

Registration of independent 3D reconstructions

As described in the previous chapter, the presented 3D reconstruction system follows a *divide and conquer* or sequential approach, where the input sequence is split into shorter sub-sequences that are independently reconstructed. This makes a difference with reference to purely differential approaches, where the whole video is processed during a single thread of work.

SaM from video sequences was initially studied as a purely differential technique [29]. After obtaining an initial pose estimation for two or three views (*i.e*, the factorization of Essential matrix as in 2.4.2), new camera views are progressively computed from feature correspondences, allowing new points to be triangulated and included in the 3D scene structure. Although this approach is suitable for short video sequences, some issues arise as it is applied to longer sequences, where it is difficult to keep a stable tracking of correspondences: one can notice a drift of the position of the features as the sequence progresses.

An efficient way to face drift propagation while keeping a stable tracking of features consists of decoupling the problem by dividing the sequence into subsequences or atomic subparts that are independently reconstructed and registered [15, 16]. An accepted practice divides a sequence into overlapped triplets of images (for trifocal tensor estimation) whose associated reconstructions are progressively merged into a global coordinate system.

Another reason to support the division of a large sequence into smaller pieces that are registered is to overcome the issues of increasing computational cost and bad scalability of nonlinear Newton-like optimization algorithms used in SaM as the complexity of the scene (number of cameras and 3D points) increases.

During this chapter, novel registration algorithms for merging partial reconstructions from video sequences into a common coordinate frame are presented. Our technique combines texture correspondence (shared 3D points between independent reconstructions) and temporal redundancy (planes defining overlapped cameras) into simple, linear and robust algorithms for the registration of 3D reconstructions.
In addition, since the number of common 3D points and cameras that the the proposed approach can handle is not bounded from above, longer subsequences than triplets of images may be used. This allows the registration (a space homography) to be estimated from the information of a larger region of the 3D space (*i.e* the fields of view of common cameras and location of common 3D points), which leads to improved registration accuracy.

4.1 Overview of the algorithm

Along this chapter, our approach to the problem of registering partial 3D reconstructions to a common cordinate frame is addresed. The objective is to find an optimal transformation $\mathtt{H_s}$ that uniquely relates two independent metric reconstructions of the same rigid scene. The optimality of this transformation is ensured by forcing it to minimize some error function suited to the problem. This transformation is used to transfer geometric entities such as space points $\mathbf{X}$, lines $\mathbf{l}$ and planes π between coordinate frames of two partial reconstructions.
Two metric reconstructions are related by a similarity transform $\mathtt{H}_s$ with 7 degrees of freedom, while two projective reconstructions by a general projective homography $\mathtt{H}$ with 15 dof. In the former case, the two metric reconstructions differ in orientation, translation and scale, although they represent the same rigid scene. Nevertheless, instead of independently estimating the intrinsic parameters of $\mathtt{H_s}$, ($\mathtt{R}$, $\mathbf{t}$ and σ), a stratified approach is followed. A projective transformation between reconstructions $\mathtt{H}$ is computed and furtherly is upgraded to $\mathtt{H_s}$ by projecting it to the space of similarity transforms (for further details refer to[1]). This kind of approach is interesting for two reasons: it allows our registration algorithm to work in projective, affine or metric spaces and simplifies the process of independently estimating the seven parameters of a similarity transform.
The basic idea of the proposed approach is the following: given a set of corresponding 3D points or planes, it is possible to determine a unique transformation $\mathtt{H}$ such that, the line joining a 3D point (or plane) in a coordinate system and the

♯ cameras (N_c)	♯ 3D points (N_p)	♯ Equations (N_e)
0	5	15
1	4	21
2	0	18

TABLE 4.1: Configurations for RANSAC's Minimum Sample Set.

transferred 3D point (or plane) from another coordinate system is not defined. As seen, we indistinctly refer to 3D points and planes, which is a direct consequence of the point / plane duality theorem from Projective Geometry presented in Chapter 2.

If a pinhole camera model is assumed, the rows of the projection matrix P may be interpreted as the homogeneous expression of three planes π_i(4.1). The linear algorithm handles camera overlapping as the correspondence between identical planes in different coordinate systems.

$$\mathtt{P} = \mathtt{K} \begin{pmatrix} \mathtt{R} & \mathtt{t} \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} \pi_1^T \\ \pi_2^T \\ \pi_3^T \end{pmatrix} \tag{4.1}$$

With the aim of robustifying the system and due to the noisy and prone to outliers nature of reconstruction systems, two strategies will be followed. Regarding to noise, registration algorithms are able to deal with $N \geq 5$ correspondences as it will detailed in section 4.3. Equality case represents the minimum sample to obtain an estimation of H, since each correspondence provides 3 independent linear equations. The ability of the system to process bigger $N's$ may be interpreted as an *averaging* of the noise present in the positions of the retrieved correspondences. However, since it is not possible to discard outliers just with a linear solution, a RANSAC [24] algorithm is used in order to carry out this selection of good correspondences. In our approach, one or a combination of the two algorithms that are proposed acts as the *kernel* of the RANSAC procedure. Moreover, since the nature of our technique combines both points and planes as input data, RANSAC will accept, as the Minimum Sample Set (MSS), all those different combinations providing $N_e \geq 15$ independent linear equations. Possible configurations for the MSS and number of equations provided are listed in table 4.1.

Finally, the obtained transformation providing a reliable consensus during RANSAC, will be used as the input to the refinement step. As will be described in 4.5, several strategies are proposed to optimize the computed transformation.

In the following sections, the problem of registering partial reconstructions will be properly formulated and the algorithms for points and planes detailed. The chapter ends by presenting different approaches for refining a primary solution obtained whose performance will be discussed along the results chapter 5.

4.2 Problem Statement

Assume two 3D metric reconstruction of the same static scene, $\{\mathtt{P}^{\mathtt{i}}, \mathbf{X}_j\}$ and $\{\mathtt{P}^{\mathtt{m}}, \mathbf{X'}_n\}$, are given in distinct coordinate systems. Both reconstructions were obtained from a set of observed points in each one of them, $\{\mathbf{x}_j^i\}$ and $\{\mathbf{x'}_m^n\}$. Also assume that correspondences between both reconstructions, 3D points $\{\mathbf{X_k}, \mathbf{X'_k}\}$ and overlapped cameras $\{\mathtt{P}^{\mathtt{r}}, \mathtt{P}'^{\mathtt{r}}\}$ are available and they provide $N \geq 15$ linearly independent equations.

The goal is to find the 3D homography $\mathtt{H_s}$ (similarity transformation) that uniquely relates both coordinate systems. Following the rule for points $\mathbf{X}_k = \mathtt{H}_s\mathbf{X'}_k$ and $\mathtt{P}^r = \mathtt{P}'^r\mathtt{H}_s^{-1}$ for cameras we ensure that 2D projections are preserved. Therefore, once the homography $\mathtt{H}_s$ is found, both partial reconstructions could be expressed in the same coordinate system: $\{\mathtt{P}^r \cup \mathtt{P}'^r\mathtt{H}_s^{-1}, \mathbf{X}_k \cup \mathtt{H}_s\mathbf{X'}_k\}$.

In practice, reconstructions are noisy and contamined from outliers. Therefore the first step will consist on determining the set of correspondences that may be considered as inliers. As mentioned, a RANSAC process is used for this aim. It just remains to define a metric that determines wether a plane or a point correspondence is cosidered to be a wrong match. The metric used is the Symmetric Transfer Error 4.4 but it has to be specified for both types of primitive. For the case of 3D point correspondences the error is expressed as:

$$\epsilon_k^2 = \sum_i d^2(\mathbf{x}_k^i, \mathtt{P}^i\mathtt{H}_s\mathbf{X'}_k) + \sum_m d^2(\mathbf{x}_k^m, \mathtt{P}'^m\mathtt{H}_s^{-1}\mathbf{X}_k) \tag{4.2}$$

For a camera correspondes the expression is:

$$\zeta_k^2 = \sum_j d^2(\mathbf{x}_j^r, P^r H_s\mathbf{X'}_j) + \sum_n d^2(\mathbf{x}_n^r, P'^m H_s^{-1}\mathbf{X}_n) \tag{4.3}$$

The threshold is fixed to 1 pixel of reprojection error for deciding points and cameras to be considered in further steps.

Next, with the already retrieved inliers, the problem is reformulated into an optimization framework. Now the objective is to obtain a similarity transformation $\mathtt{H}_s$

that minimizes some cost function. Two cases will be considered for the optimization: minimization of the Algebraic error and the minimization of Geometric error. For the latter case, the cost function is a composition of the Symetric Transfer error for each point or camera correspondence:

$$STR = \sum_k \epsilon_k^2 + \sum_r \zeta_r^2 \tag{4.4}$$

Newton-like optimization algorithms may be used to minimize (4.4) or other cost functions from an initial estimate of $\mathtt{H}_s$ (see 4.5). In the following we present a set of linear algorithms for estimating $\mathtt{H}_s$ both from motion and structure clues that can be easily combined in order to provide a more robust algorithm for merging partial reconstructions.

4.3 Linear Algorithm due to Structure

Given $N_p > 5$ 3D point correspondences $\{\mathbf{X}_k, \mathbf{X'}_k\}_{k=0,1,\ldots,N_p-1}$ is possible to find a linear transformation $\mathtt{H}$ such that satisfies the following relation $\mathbf{X}_k \sim \mathtt{H}\mathbf{X'}_k$. The equality case is trivial since there always exists a valid solution for $\mathtt{H}$. In fact, for 5 correspondences, the problem can be tackled as a classic linear change of base between two *4-dimensional* vectorial spaces. However, as it was previously stated, 3D reconstructions are noisy and an algorithm able to handle $N_p \geq 5$ correspondences is mandatory.

As described in section 4.1, the idea behind this linear algorithm is to find a valid $\mathtt{H}$ that forces the line joining a 3D point $\mathbf{X}_k$ and its transferred correspondence $\mathtt{H}\mathbf{X'}_k$ to be not defined. Let us assume that $\mathbf{u}$ and $\mathbf{v}$ respectively refer to the *k-th* correspondence of 3D points $\{\mathbf{X}_k, \mathbf{X'}_k\}$ and $\mathbf{v'}$ corresponds with the transferred version of $\mathbf{X'}_k$, $\mathbf{v'} = \mathtt{H}\mathbf{X'}_k$. The line joining $\mathbf{u}$ and $\mathbf{v'}$ is defined by the *skew-symmetric* Plucker matrix $\mathtt{L}(\mathbf{u}, \mathbf{v'})$. Recall from 2.1.4 that a Plucker matrix defining a line in a 3D projective space is obtained from the expression:

$$L(\mathbf{u}, \mathbf{v'}) = \mathbf{u}\mathbf{v'}^T - \mathbf{v'}\mathbf{u}^T = \begin{pmatrix} 0 & l_{12} & l_{13} & l_{14} \\ -l_{12} & 0 & l_{23} & l_{24} \\ -l_{13} & l_{23} & 0 & l_{34} \\ -l_{14} & -l_{24} & -l_{34} & 0 \end{pmatrix} \tag{4.5}$$

Using a compact representation, (4.5) is equivalent to the *6-dimensional* Plucker vector $\mathbf{l}(\mathbf{u}, \mathbf{v}') = (l_{12}, l_{13}, l_{14}, l_{23}, l_{24}, l_{34})$, whose scalars l_{ij} are:

$$l_{ij} = u_i v'_j - v'_i u_j \tag{4.6}$$

If the desired transformation $\mathtt{H}$ is expressed as a 4-vector matrix $\mathtt{H} = \{\mathbf{h}_i^T\}_{i=1...4}$ and is combined with the aforementioned expression $\mathbf{v}' = \mathtt{H}\mathbf{v}$, we may derive an expression (4.6) depending only on the input data $\mathbf{u}$, $\mathbf{v}$ and the desired transformation H:

$$\begin{aligned} l_{ij} &= u_i v'_j - v'_i u_j \\ &= u_i \mathbf{h}_j^T \mathbf{v} - u_j \mathbf{h}_i^T \mathbf{v} \\ &= (u_i \mathbf{v}^T)\mathbf{h}_j - (u_j \mathbf{v}^T)\mathbf{h}_i \end{aligned} \tag{4.7}$$

Since we want the 3D line $\mathtt{L}(\mathbf{u}, \mathbf{v})$ to be not defined, we force its elements to be equal zero. Therefore it provides 6 different linear dependent equations in the entries of $\mathtt{H}$:

$$\begin{aligned} l_{12} &= (u_1 \mathbf{v}^T)\mathbf{h}_2 - (u_2 \mathbf{v}^T)\mathbf{h}_1 = 0 \\ l_{13} &= (u_1 \mathbf{v}^T)\mathbf{h}_3 - (u_3 \mathbf{v}^T)\mathbf{h}_1 = 0 \\ l_{14} &= (u_1 \mathbf{v}^T)\mathbf{h}_4 - (u_4 \mathbf{v}^T)\mathbf{h}_1 = 0 \\ l_{23} &= (u_2 \mathbf{v}^T)\mathbf{h}_3 - (u_3 \mathbf{v}^T)\mathbf{h}_2 = 0 \\ l_{24} &= (u_2 \mathbf{v}^T)\mathbf{h}_4 - (u_4 \mathbf{v}^T)\mathbf{h}_2 = 0 \\ l_{34} &= (u_3 \mathbf{v}^T)\mathbf{h}_3 - (u_4 \mathbf{v}^T)\mathbf{h}_3 = 0 \end{aligned}$$

With the provided set of equations, we may express the problem as $\mathtt{A}_k\mathbf{h} = 0$, where $\mathtt{A}_k$ is the 6×16 measurement matrix associated to a single correspondence, while $\mathbf{h} = (\mathbf{h}_1^T, \mathbf{h}_2^T, \mathbf{h}_3^T, \mathbf{h}_4^T)^T$ is the column vector resulting from stacking vertically the four planes $\mathbf{h}_i$ conforming the transformation $\mathtt{H}$. Let us define the 6×4 matrix:

$$W(\mathbf{u}) = \begin{pmatrix} -u_2 & u_1 & 0 & 0 \\ -u_3 & 0 & u_1 & 0 \\ -u_4 & 0 & 0 & u_1 \\ 0 & -u_3 & u_2 & 0 \\ 0 & -u_4 & 0 & u_2 \\ 0 & 0 & -u_4 & u_3 \end{pmatrix} \tag{4.8}$$

Therefore, the matrix $\mathtt{A}_k$ obtained from a single correspondence follows:

$$\mathtt{A}_k = W(\mathbf{u}) \otimes \mathbf{v}^T = W(\mathbf{X}_k) \otimes \mathbf{X}_k'^T, \tag{4.9}$$

where $\otimes$ stands for the Kronecker product. Hence, the problem for a single correspondence is expanded as:

$$\underbrace{\left(W(\mathbf{u}) \otimes \mathbf{v}^T\right)}_{\mathtt{A}_k} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{4.10}$$

Each correspondence provides 6 linear equations, but only 3 of them are linearly independent. Hence, at least 5 correspondences are necessary to estimate the 15 parameters of the projective transformation $\mathtt{H}$. In order to incorporate new correspondences to (4.10), let us define the $6N_p \times 16$ matrix $\mathtt{A}$, obtained by stacking N_p measurement matrices $\mathtt{A}_k$ for each one of the correspondences. In presence of noise, $\mathbf{h}$ may be found as the solution for the classical optimization problem:

$$\min ||\mathtt{A}\mathbf{h}|| \text{ subject to } ||\mathbf{h}|| = 1 \tag{4.11}$$

The solution is obtained by means of solving a classical SVD problem, where $\mathbf{h}$ is the right singular vector vector associated to $\sigma_{16} = \frac{||\mathtt{A}\mathbf{h}||}{||\mathbf{h}||}$, the smaller singular value of $\mathtt{A}$.

4.4 Linear Algorithm due to Motion

When aligning partial reconstructions from overlapped subsequences, it is possible to exploit camera correspondences in order to obtain a projective transformation between independent reconstructions. Let us assume that there exist N_c camera correspondences $\{\mathtt{P}^r, \mathtt{P}'^r\}$. In that case, the measurement matrix $\mathtt{A}$ from equation (4.11) may be extended to allow camera correspondences and derive the matrix $\mathtt{H}$ that satisfies $\mathtt{P}^r \sim \mathtt{P}'^r\mathtt{H}^{-1}$. Moreover, if $N_c > 2$ is it possible to obtain $\mathtt{H}$ without any 3D point correspondence.

The rows of a projection matrix $\mathtt{P}$ are the homogeneous coordinates of three planes $\{\pi_i\}$ intersecting in the camera camera centre (4.1). Recall from section 2.1.4 that, given two corresponding planes$\{\pi, \pi'\}$ in different coordinate systems, the transformation rule relating both primitives follows $\pi = \mathtt{H}^{-T}\pi'$. For simplicity purposes, let us express the *i-eth* plane of camera $\mathtt{P}^r$ as $\pi_i^r = \mathbf{u}$ and its correspondence

in camera $\mathtt{P}'^r$ as $\pi'^r_i = \mathbf{v}$. In addition, the transformed version of the latter is $\mathtt{H}^{-T}\pi'^r_i = \mathbf{v}'$.

Note, that the inversion applied to the transposed of the transformation $\mathtt{H}$, makes it more difficult to derive a close-form solution the one for the 3D points algorithm. With the aim of simplifying the mathematical derivation, we flip the structure of the problem and, instead of estimating $\mathtt{H}$ from the 3D line $\mathbf{l}^*(\mathbf{u}, \mathtt{H}^{-T}\mathbf{v})$, the Plucker vector used is $\mathbf{l}^*(\mathtt{H}^T\mathbf{u}, \mathbf{v})$. In [30], it is demonstrated that if $\mathbf{l}^*(\mathbf{u}, \mathbf{v}' = \mathtt{H}^{-T}\mathbf{v}) = \mathbf{0}$, that is equivalent to $\mathbf{l}^*(\mathbf{u}' = \mathtt{H}^T\mathbf{u}, \mathbf{v}) = \mathbf{0}$.

The last remaining step consists on defining the Plucker components of the 3D line $\mathbf{l}^*(\mathbf{u}', \mathbf{v})$. If we expressed $\mathtt{H}^T$ as the 4-vector matrix $\mathtt{H}^T = (\mathbf{h}_1^{*T}, \mathbf{h}_2^{*T}, \mathbf{h}_3^{*T}, \mathbf{h}_4^{*T})$, the components of the line follow:

$$l^*_{ij} = u'_i v_j - v_i u'_j = \mathbf{u}^T \mathbf{h}^*_i v_j - v_i \mathbf{u}^T \mathbf{h}^*_j \tag{4.12}$$

Proceeding in the same way that for the linear algorithm due to structure, the rest of the process is straightforward. First of all, the set of six equations for each plane correspondence are obtained by equaling to zero the components of the Plucker Vector:

$$\begin{aligned}
l_{12} &= (v_1\mathbf{u}^T)\mathbf{h}^*_2 - (v_2\mathbf{u}^T)\mathbf{h}^*_1 = 0 \\
l_{13} &= (v_1\mathbf{u}^T)\mathbf{h}^*_3 - (v_3\mathbf{u}^T)\mathbf{h}^*_1 = 0 \\
l_{14} &= (v_1\mathbf{u}^T)\mathbf{h}^*_4 - (v_4\mathbf{u}^T)\mathbf{h}^*_1 = 0 \\
l_{23} &= (v_2\mathbf{u}^T)\mathbf{h}^*_3 - (v_3\mathbf{u}^T)\mathbf{h}^*_2 = 0 \\
l_{24} &= (v_2\mathbf{u}^T)\mathbf{h}^*_4 - (v_4\mathbf{u}^T)\mathbf{h}^*_2 = 0 \\
l_{34} &= (v_3\mathbf{u}^T)\mathbf{h}^*_3 - (v_4\mathbf{u}^T)\mathbf{h}^*_3 = 0
\end{aligned}$$

This set of equations lead to the problem $\mathtt{B}_k\mathbf{h}^* = \mathbf{0}$, where $\mathbf{h}^* = (\mathbf{h}_1^{*T}, \mathbf{h}_2^{*T}, \mathbf{h}_3^{*T}, \mathbf{h}_4^{*T})^T$ and $\mathtt{B}_k$, using the matrix $W(\mathbf{v})$ from (4.8), follows:

$$\underbrace{\left(W(\mathbf{v}) \otimes \mathbf{u}^T\right)}_{\mathtt{B}_k} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{4.13}$$

From the previous expression, we may obtain the values of the 16-dimensional vector $\mathbf{h}^{*T}$ and therefore, the desired transformation matrix $\mathtt{H}$ is found. However if camera correspondences are desired to be used jointly with 3D points, an additional

step is required. In order to make compatible the set of equations provided by a point or a plane correspondence, we need to transform the matrix $\mathtt{B}_k$ to the matrix $\mathtt{A}_k$ from (4.9) so that it can be stacked with the rest of measurement matrices from point correspondences. That is done just by permuting the columns of $\mathtt{B}_k$:

$$\mathtt{A}_k = \mathrm{ColPerm}(\mathtt{B}_k) \tag{4.14}$$

Now, it is clear that if N_p common 3D points $\{\mathbf{X}_k, \mathbf{X}'_k\}$ and N_c cameras are known between two partial reconstructions, the linear algorithm in (4.11) can be used to estimate the transformation $\mathtt{H}$ from a combined measurement matrix $\mathtt{A}$. The dimensionality of this matrix is $6(N_p + 18Nc) \times 16$ and is formed by stacking measurement matrices $\mathtt{A}_k$ obtained both from point and plane correspondences.
To improve the numerical conditioning of (4.11), data normalization is essential. This includes a normalization in the image plane, which affects the observed image points, and a normalization in space, which affects the 3D points and projection matrices.

4.5 Refinement of results

Each one of the presented algorithms solves the problem $\mathtt{A}\mathbf{h} = \mathbf{0}$ where $\mathbf{h}$ contains the elements of the desired transformation $\mathtt{H}$ and $\mathtt{A}$ is the measurement matrix obtained from point and / or camera correspondences. In order to refine the solution obtained, and initial solution is computed just using the correspondences considered as inliers during the RANSAC step. Next, the obtained solution serves as the input to an optimization framework that will derive an optimal solution. Two approaches are proposed:

- *Minimization of algebraic error (LIN):* a linear cost function is used.
- *Minimization of algebraic error (STR):* the function to be minimized is the non-linear Symmetric Transfer Error (4.4).

In Chapter 5, the performance of these two approaches will be evaluated for two possible configurations: registration of partial reconstructions only from 3D point correspondences (P) and both from points and overlapped cameras. Therefore results will be provided for four possible schemes: (*i*) P-LIN, (*ii*) PC-LIN, (*iii*) P-STR and (*iv*) PC-STR.

4.5.1 Minimizing Algebraic error (LIN)

Algebraic error or distance is a linear cost function that makes no assumptions about the nature of the problem or the solution to be found. Given the problem of finding the optimal $\mathbf{h}$ for the over-determinated homogeneous system $\mathtt{A}\mathbf{h} = \mathbf{0}$, the algebraic error vector is defined as $\epsilon = \mathtt{A}\mathbf{h}$. Given two corresponding vectors $(\mathbf{u}_i, \mathbf{v}_i)$ related by an homography $\mathtt{H}$, their associated algebraic distance is:

$$d^i_{alg}(\mathbf{u}_i, \mathtt{H}\mathbf{v}_i) = ||\epsilon_i||^2 = ||\mathtt{A}_i\mathbf{h}||^2 = ||(W(\mathbf{u}_i) \otimes \mathbf{v}_i^T)\mathbf{h}||^2, \tag{4.15}$$

where $W(\mathbf{u}_i)$ and $\mathtt{A}_i$ are the matrices defined in (4.8) and (4.10) respectively. The global cost function used for finding the optimal $\mathbf{h}$ from the whole set of correspondences is obtained as the sum of the algebraic costs for each single correspondence:

$$||\epsilon||^2 = \sum_i ||\epsilon_i||^2 = \sum_i d^i_{alg}(\mathbf{u}_i, \mathtt{H}\mathbf{v}_i) \tag{4.16}$$

Finally the formulation of the problem reduces to the following minimization problem:

$$\mathbf{h} = \arg\min(||\epsilon||^2 = \sum_i (||\epsilon||^2) \text{ subject to } ||\mathbf{h}||^2 = 1 \tag{4.17}$$

Minimizing the algebraic error provides a linear (and therefore unique) and computationally cheap solution. Nevertheless, since no assumption is done about the nature of the problem, the quantity to be minimized is non geometrically or statistically meaningful. This leads to find finer solutions in the non-linear minimization framework.

4.5.2 Minimizing geometric error (STR)

Minimization of the geometric error is a criterium that approaches to the concept of statistical signal processing. We are provided with a set of measured variables $\mathbf{u}_i$ and a we try to fit a model $\mathtt{H}$ to obtain a set of estimated variables $\hat{\mathbf{u}}_i = \mathtt{H}\mathbf{v}_i$. The parameters of $\mathtt{H}$ are the hidden variables that define a cost function that will determine the goodness of the model. The cost function is designed according to the nature of the problem and its minimization provides the optimal value of $\mathtt{H}$. For this case, the selected cost function is the Symmetric TRansfer error as it was defined in (4.4). STR considers the forward $\mathtt{H}$ and the backward $\mathtt{H}^{-1}$ transformation of geometric primitives an their sum of reprojection errors in the 2D space.

The benefit of minimizing a non-linear cost function is that it provides a meaningful and accurate solution. On the other hand, non-linear functions do not usually present a single solution and therefore they require to be provided with a good initial guess. In addition the computational complexity grows exponentially with the number of variables involved in the minimization. Nevertheless, in chapters 5, it will be shown that the set of variables required to complete a reliable minimization process can be dramatically reduced.

Chapter 5

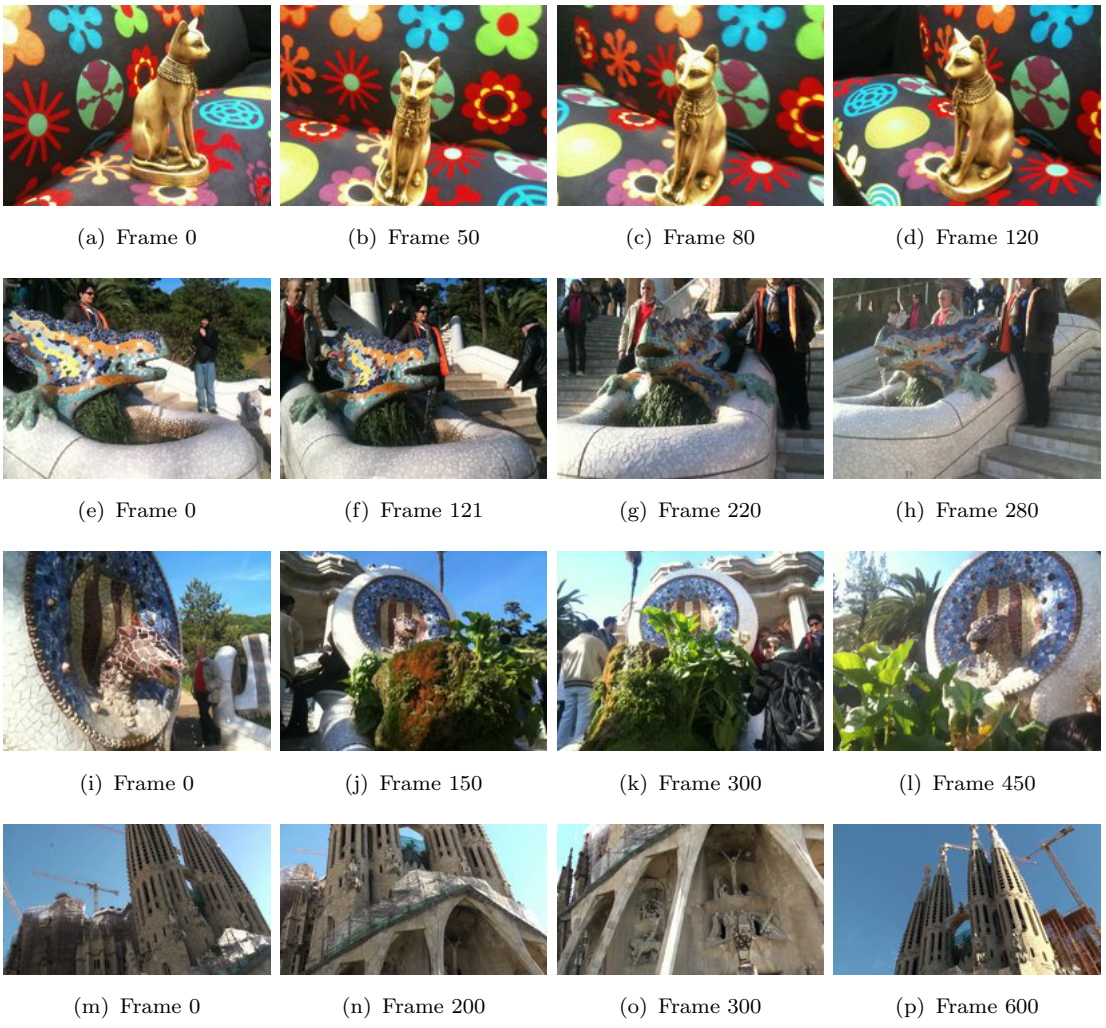
Experimental setup

Along this chapter, experimental results for the main contributions of this thesis are presented. The performance and reliability of the reconstruction system, has been analyzed in three stages: partial reconstructions, registration performance and visual results. In the first stage, the frame decimation step and the pipeline for obtaining independent reconstructions are analyzed. Frame decimation (FD) is evaluated upon a criterion of feature correspondence and smooth motion. In the second step, several configurations for registering obtained partial reconstructions are studied, both from the type of data used and the minimization strategy adopted. Finally, the obtained pose for the cameras for each one of the sequences is used to feed a well-know patch-based densification to provide attractive visual results and validate the retrieved position of cameras.

5.1 Types of sequences

One of the greatest challenges in SaM consists on dealing with the scale of the scene to be reconstructed. More precisely, the aim of Computer Vision researchers in that discipline is to derive strategies that are able handle different nature of scenes: little objects, middle-scaled scenes (*i.e* statues, motorbikes, *etc.*) or large-scale scenarios which is attracting a lot of interest in the recent times [2, 3, 5, 20]. Therefore, we want to prove the reliability and adaptability of our system by providing results for different types of sequences according to the type of scenario recorded.

It is important to note that the work for this thesis was developed within the context of a real video-to-3D project. Since users will likely provided videos recorded with cell-phones, the acquiring device will be an extra criterion to classify the



(a) Frame 0 (b) Frame 50 (c) Frame 80 (d) Frame 120

(e) Frame 0 (f) Frame 121 (g) Frame 220 (h) Frame 280

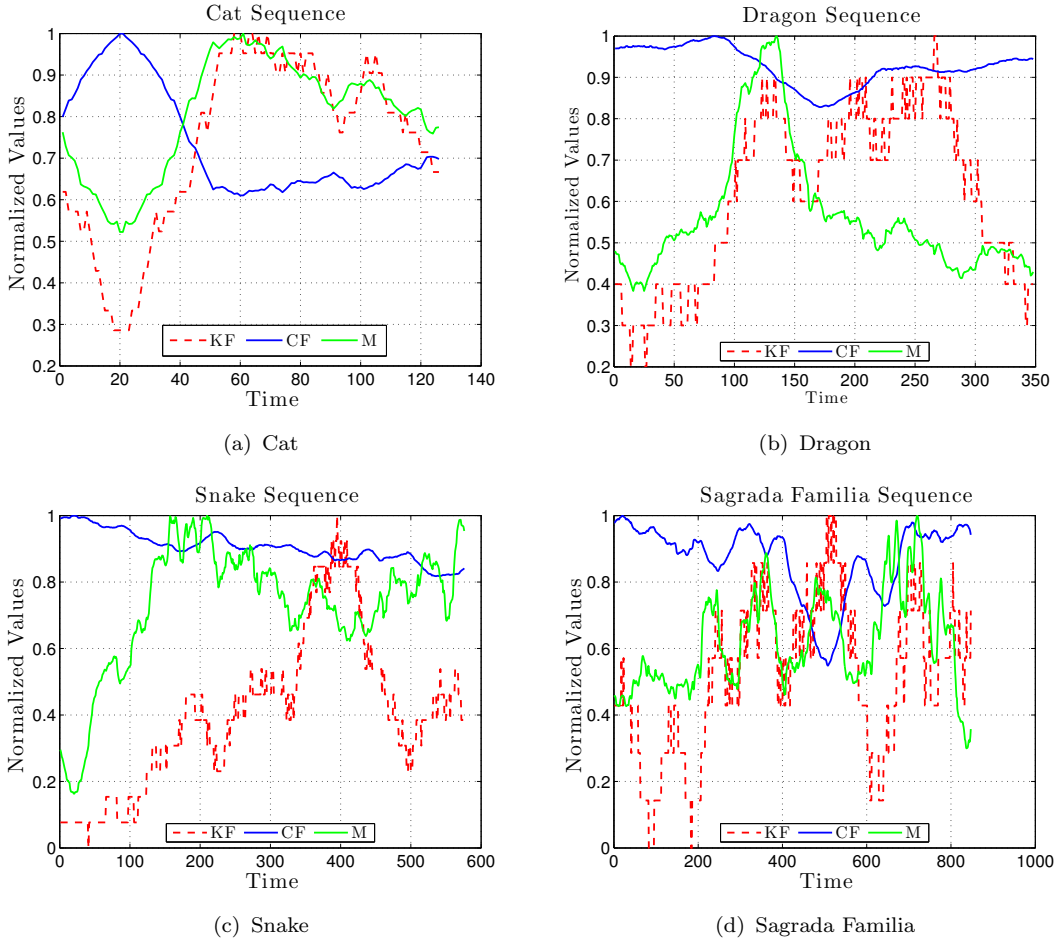
(i) Frame 0 (j) Frame 150 (k) Frame 300 (l) Frame 450

(m) Frame 0 (n) Frame 200 (o) Frame 300 (p) Frame 600

FIGURE 5.1: Snapshots of the sequences considered along the results chapter. (a-d) Cat Sequence. (e-h) Dragon Sequence. (i-l) Snake Sequence. (m-p) Sagrada Familia Sequence.

sequences studied. Results will be provided for a High Definition Sony HDSR11 Handy cam and a iPhone-3GS cellphone. Intrinsics for both devices were calibrated before hand by means of a chessboard pattern calibration with the technique described in [31].

The four sequences Fig. 5.1 are considered along this chapter of results. In the Cat sequence, a small figure of a cat over a pillow was recorded with an iPhone-3GS. Dragon and Snake sequences refer to similar mid-scale statues from Parc Guell in Barcelona. They were also recorded with an iPhone. The last one, Sagrada Familia, corresponds to Gaudi's temple recorded from a car with a HD camera.



(a) Cat (b) Dragon

(c) Snake (d) Sagrada Familia

FIGURE 5.2: Graphical analysis of the frame decimation step for the sequences

5.2 Partial Reconstructions

5.2.1 Frame decimation analysis

Let us start by analyzing the performance of the frame decimation step for the four sequences provided. A good frame decimator should adapt to the dynamics of the input video sequence in order to provide the reconstruction step with a set of keyframes sharing enough features and with an isotropic spatial distribution. Therefore, we may summarize these desired characteristics just by saying that the frame decimator should adapt to the dynamics of the input video sequence.
Let us define the following signals in order to graphically represent the behaviour of the frame decimation step:

1. **Common Features**: Number of shared features between adjacent frames. It will help to analyze the evolution of availability of correspondences along

the sequence:

$$CF[n] = |F_n \cap F_{n+1}|, \tag{5.1}$$

where F_i refers to the total number of detected features for the $i-th$ frame.

2. **Apparent motion**: intuitive idea of the displacement between views. It is computed as the median of the 2D euclidean distance between the position of the detected features in one image and their transferred correspondences by means of a computed homography:

$$M[n] = \underset{j}{\text{median}}||\mathbf{x}_j^n - \mathtt{H}\mathbf{x}_j^{n+1}||, \tag{5.2}$$

3. **Key Framing**: Binary signal $KF[n]$ that indicates whether the current frame is a KF or not.

The expected behaviour of the frame decimator is to introduce a larger concentration of KFs in the intervals where $M[n]$ raises and the other way round. On the other hand, if the evolution of $CF[n]$ leads to a decrease in the number of features, the frame decimator will also concentrate more KF's in that time interval in order to ensure the possibility of a reliable feature tracking.
Since the objective is to compare the concentration of KF's with respect to the evolution of $CF[n]$ and $M[n]$, let us simplify the expressions of the mentioned signals in order to provide a clearer representation of them. First, $CF[n], M[n]$ are smoothed by means of a moving average with a $L = 40$ sample rectangular window. The same smoothness is applied to the binary signal $KF[n]$ in order to obtain an intuitive description of the concentration of KF's in the neighborhood of each time interval:

$$CF'[n] = \frac{1}{L}\sum_{i=n-\frac{L}{2}}^{n+\frac{L}{2}} CF[i]$$

$$M'[n] = \frac{1}{L}\sum_{i=n-\frac{L}{2}}^{n+\frac{L}{2}} M[i]$$

$$KF'[n] = \frac{1}{L}\sum_{i=n-\frac{L}{2}}^{n+\frac{L}{2}} KF[i]$$

Finally, in order to represent them in the same figure, the three signals are normalized with respect to their maximum values. Therefore, the dynamic range for

them will be between 0 and 1.

$$\hat{CF}'[n] = \frac{CF'[n]}{max(CF'[n])}$$
$$\hat{M}'[n] = \frac{M'[n]}{max(M'[n])}$$
$$\hat{KF}'[n] = \frac{KF'[n]}{max(KF'[n])}$$

In Fig. 5.2, the smoothed and normalized versions of the apparent motion, evolution of common features between frames and concentration of KFs are presented for the four sequences considered in this chapter of Results. As seen, the presented frame decimation scheme adapts to the dynamics of the input video sequence as it was desired.

If we focus in Fig. 5.2-a, it is evident that the concentration of KF's directly follows the evolution of the apparent motion and inversely the dynamic of the common features. In the interval $(400, 600)$ of the Sagrada Familia sequence, we observe a sudden decrease of common features as well as a rushed acceleration in the moving speed of the camera. For that time interval, it is noticed an increase of the KF concentration in order to ensure the connectiviy KFs between during the reconstruction process. Although not being so evident, the same behaviour can be observed for the rest of sequences.

Hence, the conclusion that arises is that our frame decimation step smartly reduces the temporal redundancy of the video sequence. As a difference with other approaches, for instance reducing the frame rate of the input video, our scheme is fully automatic and it does not require from a special adjustment depending on the recorded scene. This process helps on dealing with sequences of different sizes, since no manual tuning is needed.

5.2.2 Comparison with other schemes

Now it is desired to analyze the joint performance of the two contributions presented in chapter 3, frame decimation and in-loop feature tracking, and compare it to other approaches. Two possible configurations have been studied: (a) OL-US, a naive approach where the input sequence is uniformly downsampled and the feature tracking is done before reconstruction, and (b) IL-FD, with embedded feature tracking within SaM step and the frame decimation scheme from 3.2.2, which is one of the main contributions of this thesis.

Scene	Method	Points	Cams	KFs/Fs	Clips	E_{preBA}/E_{postBA}	$T_{Rec.}/T_{Total}$
Cat	OL-US	7412	63	65/127	1	0.533/0.310	4.5s/5.0s
	IL-FD	6159	51	51/127	3	0.263/0.259	5.0s/5.0s
Dragon	OL-US	10367	34	70/349	4	4.840/0.314	6.8s/13.9s
	IL-FD	19850	54	56/349	5	0.264/0.227	13.8s/13.9s
Snake	OL-US	18239	116	116/577	8	1.581/0.323	23.1s/23.1s
	IL-FD	19711	77	77/577	8	0.264/0.260	23.1s/23.1s
S. Fam.	OL-US	18603	147	170/848	6	0.369/0.218	29.3s/33.9s
	IL-FD	16919	81	81/848	7	0.242/0.226	33.9s/33.9s

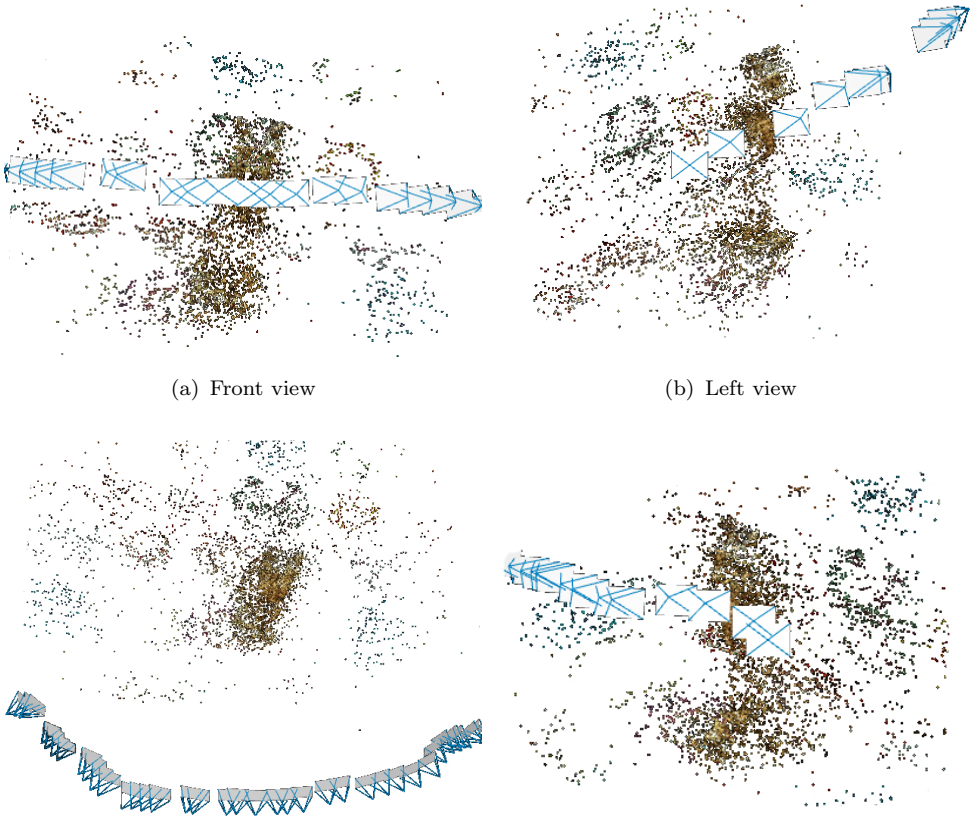
TABLE 5.1: Numerical evaluation of the reconstruction of the four sequences for two different approaches. The quality measurements considered are the number of triangulated points (Points) and correctly resected cameras (Cams), the number of KFs with respect the total numbe of frames, errors before (E_{preBA}) and after (E_{postNA}) Bundle Adjustment and total reconstructed time (T_rec). The column Clips refers to the number of subsequences created from the original input video.

In order to generate the results for OL-US setup, sequences were split into fixed-length clips of 150 frames with 75 frames of overlap. Next a deterministic frame decimation based on a uniform sampling was applied. The sampling rate selected was of 1/5 for Dragon, Snake and Sagrada Familia and of 1/2 for Cat sequence. Those were the configurations providing the best performance for sequences with the OL-US setup.

Since both schemes provide different number of KFs, one of the quality parameters listed in Table 5.1 is the reconstruction time T_{Rec}. This parameter expresses the total time length defined by the connected KFs that were correctly resected. As seen, IL-FD outperforms OL-US for almost all the sequences except for Snake scene where T_{Rec} is the same. The IL-FD configuration provides a more isotropic temporal distribution of KFs and a more stable feature tracking. Hence, isolated frames are less frequent than for OL-US and, therefore, T_{Rec} is larger.

The number of triangulated 3D points is similar for both configurations although total frames processed in IL-FD is lower than in OL-US. That indicates that bounding the affinity factor in equation (3.1) during FD, allows to introduce larger amounts of non-triangulated points at each iteration of the SaM update loop. In other words, it acts as a *renewal factor*.

Furthermore, the global reprojection error between both approaches has been studied before and after a global Bundle Adjustment (BA). Post-BA error is similar for any configuration or sequence. However the reprojection error before BA is significantly lower for the IL-FD step for the first three sequences (captured with



(a) Front view (b) Left view

(c) Zenital view (d) Right view

FIGURE 5.3: 3D model for Cat Sequence

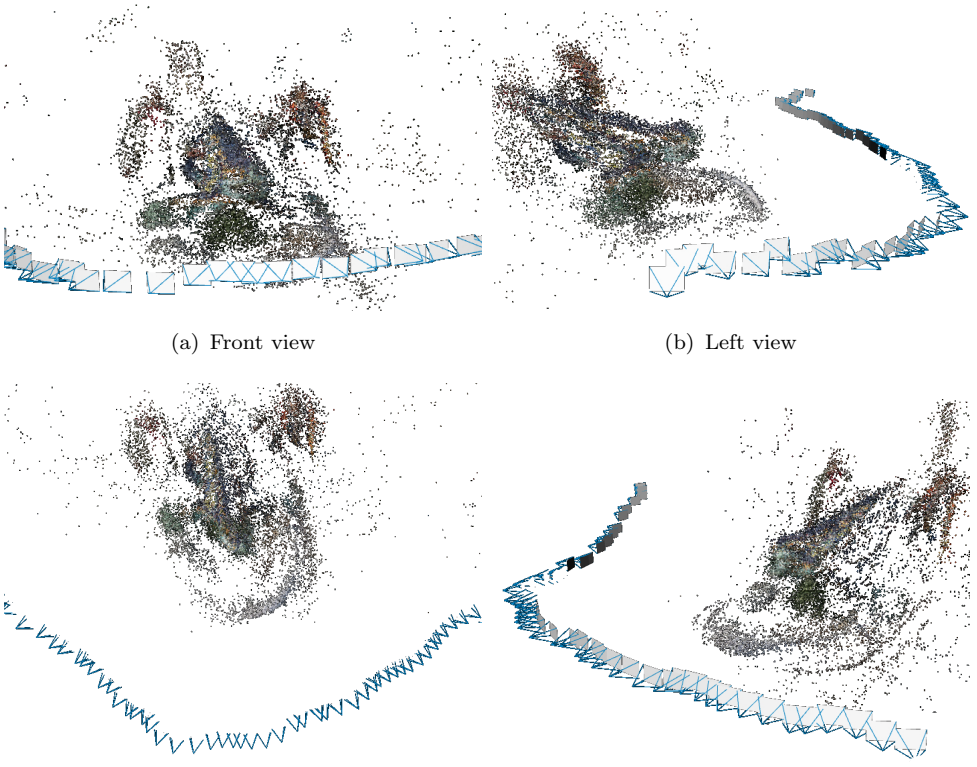
cell-phone). That leads us to think that the reliability of the triangulated points with the IL-FD scheme is enough to avoid the BA in some stages of the reconstruction. That would significantly reduce the reconstruction time.
Finally, the obtained 3D models for each one of the sequences, are depicted in figures 5.3, 5.4, 5.5 and 5.6.

5.3 Registration Performance

Along this section, the objective is to analyze the performance of the algorithms described in chapter 4. Depending on the type of input data, two configurations will be analyzed:

1. Only point correspondences (P)
2. Corresponding points and cameras (PC).

In addition, two minimization strategies are proposed:



(a) Front view (b) Left view

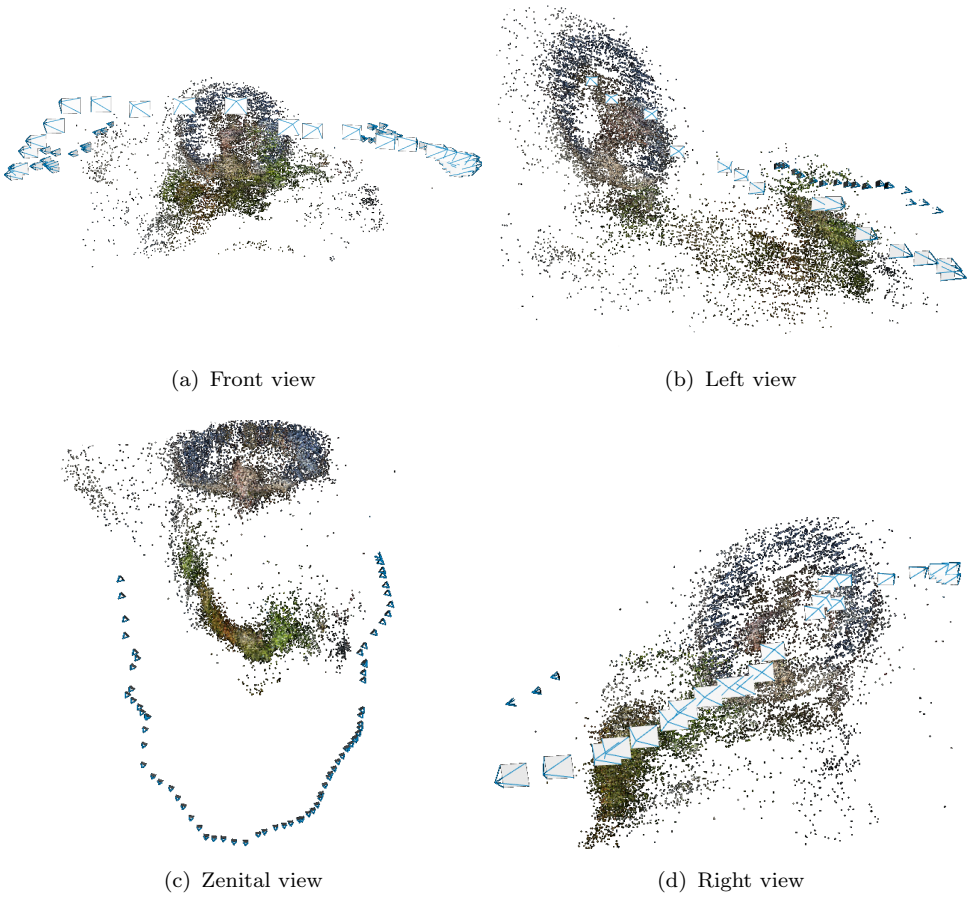
(c) Zenital view (d) Right view

FIGURE 5.4: 3D model for Dragon Sequence

1. Minimizing a linear Algebraic error (LIN)

2. Minimization of a non linear Geometric error (STR).

As we have seen so far, there are four setups whose robustness and reliability must be analyzed: P-LIN, PC-LIN, P-STR and PC-STR (see section 4.5 from chapter 4 for further details). The results to be analyzed are the total number of reconstructed points in the final reconstructions and the error obtained in the last merge step both before and after an optimal non-linear BA.
Recall from the global description of the system in chapter 3, that some *safety* constraints were introduced along the reconstruction process. Triangulated points with larger triangulation angle than 1 degree or with mean reprojection error over 1 pixel were discarded. The same constraints are introduced after each registration step. Therefore it is straightforward to see that the higher the quality of the registration step, the larger the number of reconstructed 3D points.
In table 5.2, a summarization of the values obtained for each sequence for each one of the four listed configurations is presented. The "Before registration" column refers to the values of mean number of 3D points and error obtained per



(a) Front view (b) Left view

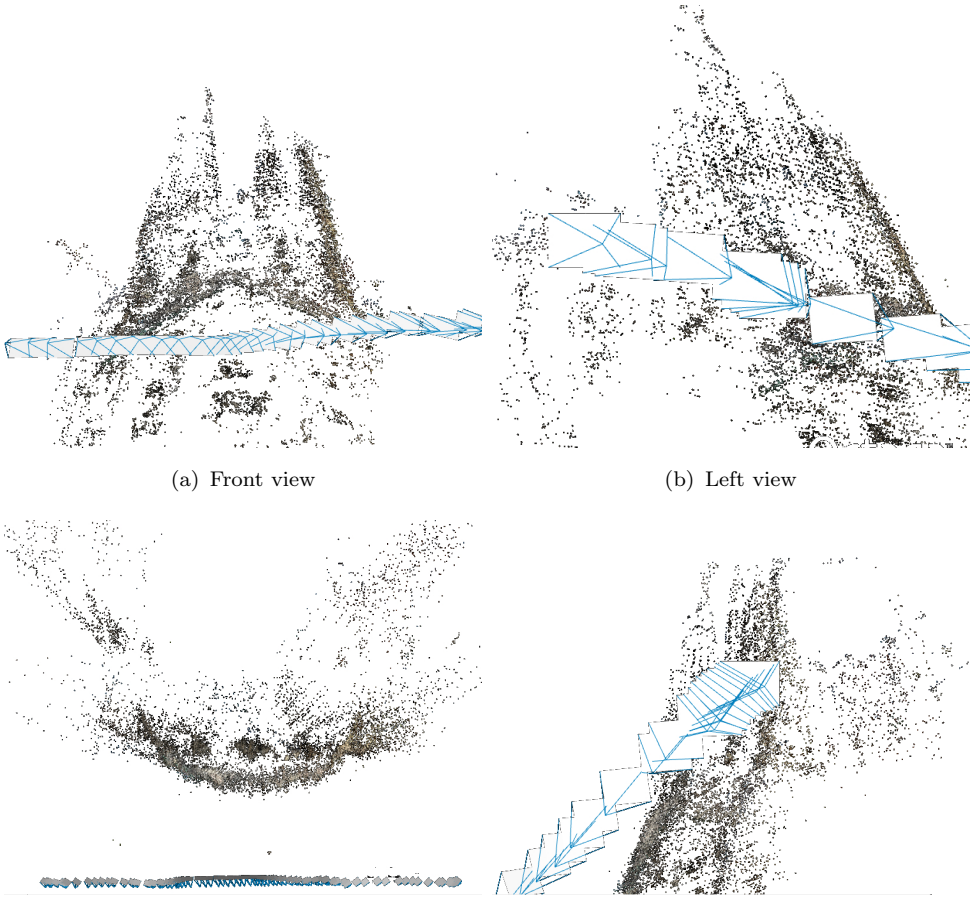
(c) Zenital view (d) Right view

FIGURE 5.5: 3D model for Snake Sequence

reconstructed clip. This values show how accurate the reconstruction process was. Therefore, possible errors after the merging step ("After registration" column) could be only associated to the registration stage.

Following the pipeline from chapter 3, the four sequences used for results were pre-decimated and split into subsequences with a fixed number of KFs (20 in our experiments). Moreover, the clips were selected in such a way that they present some overlapping (10 KFs in our experiments) in order to favour the algorithm to be provided with camera correspondences. The reconstruction technique for each one of the clips is IL-FD from 5.2 since is our best-performance configuration.

In table 5.1, numerical results are summarized for the four real sequences considered in our experiments. Due to the robustness and accuracy of the reconstruction step, the four studied configurations yield outstanding registrations with respect to the reprojection error. Nevertheless, some considerations must be done in order to analyze the real performance of the registration algorithms.



(a) Front view (b) Left view

(c) Zenital view (d) Right view

FIGURE 5.6: 3D model for Sagrada Familia Sequence

When comparing P and PC for both minimization strategies LIN and STR, the robust algorithm using point and camera correspondences outperforms the one using only points in the number of recovered 3D points. This clearly indicates that, as it was expected, the combination of points and cameras into a single linear algorithm robustifies the registration step. Since the quality of the registration is higher, less 3D points are discarded according to error or angle constraints.
When comparing the two minimization approaches LIN and STR, the difference in the results obtained are much more relevant. Recall that LIN strategy solves in a first step a linear minimization by means of SVD decomposition. Therefore, it is not assuming anything about the nature of the problem and, hence, the solution is not expected to be as precise as desired. That is why it is usually followed by

Scene	Setup	Before registration (mean values per clip)		After registration (merged reconstruction)		
		3D points	Rep. error	3D points	E_{pre-BA}	$E_{post-BA}$
Cat	P-LIN	2482	0.269	5196	52.2	0.291
	PC-LIN			5334	16.6	0.257
	P-STR			6157	0.264	0.259
	PC-STR			6159	0.263	0.259
Dragon	P-LIN	8236	0.271	19322	10.1	0.259
	PC-LIN			19601	11.0	0.2607
	P-STR			19824	0.263	0.227
	PC-STR			19850	0.264	0.227
Snake	P-LIN	4707	0.268	4830	79.8	0.223
	PC-LIN			16321	59.7	0.230
	P-STR			19474	0.280	0.260
	PC-STR			19711	0.264	0.260
S. Fam	P-LIN	3962	0.258	14047	69.1	0.221
	PC-LIN			14057	38.2	0.221
	P-STR			16781	0.241	0.224
	PC-STR			16919	0.242	0.226

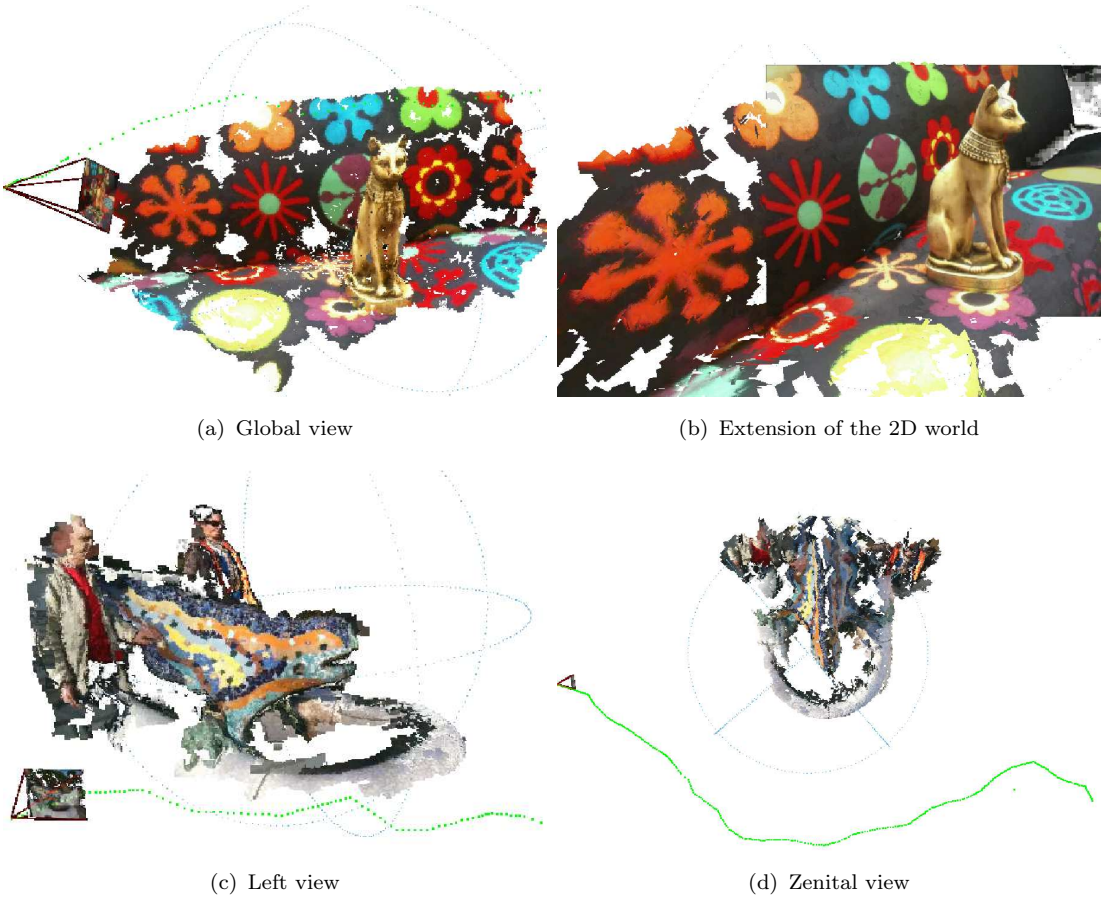
TABLE 5.2: Numerical evaluation of four different registration schemes on the given datasets.

a non-linear global minimization step (Bundle Adjustment):

$$E_{postBA} = \min_{\mathtt{P}_r, \mathbf{X}^i} \sum_{r=0}^{N_c-1} \sum_{i=0}^{N_X-1} d(\mathtt{P}_r \mathbf{X}^i, \mathbf{x}_i^r)^2 \tag{5.3}$$

On the other hand STR minimizes a geometric cost function, the Symmetric TRansfer error (4.4), which is adapted to the nature of the problem. However, the big difference is that, for STR, the variables involved in the minimization process are only those that are shared between pairs of reconstruction (connecting variables in [13]). On the contrary, in LIN non-linear refinement all the variables from the reconstructed scene are involved. Here is where it comes the big deal: *LIN refinement is global and STR refinement is local.*

When comparing results from table 5.2, it is straightforward to see that the non-linear minimization step for LIN is mandatory since the difference between pre-BA and post-BA is about of 2 or 3 orders of magnitude. As opposed to LIN, the difference between the reprojection errors E_{preBA} and E_{postBA} is barely perceptive. Therefore, the conclusion that arises is that, in our registration technique, it is equivalent to minimize a global cost function than a local one. Nevertheless the



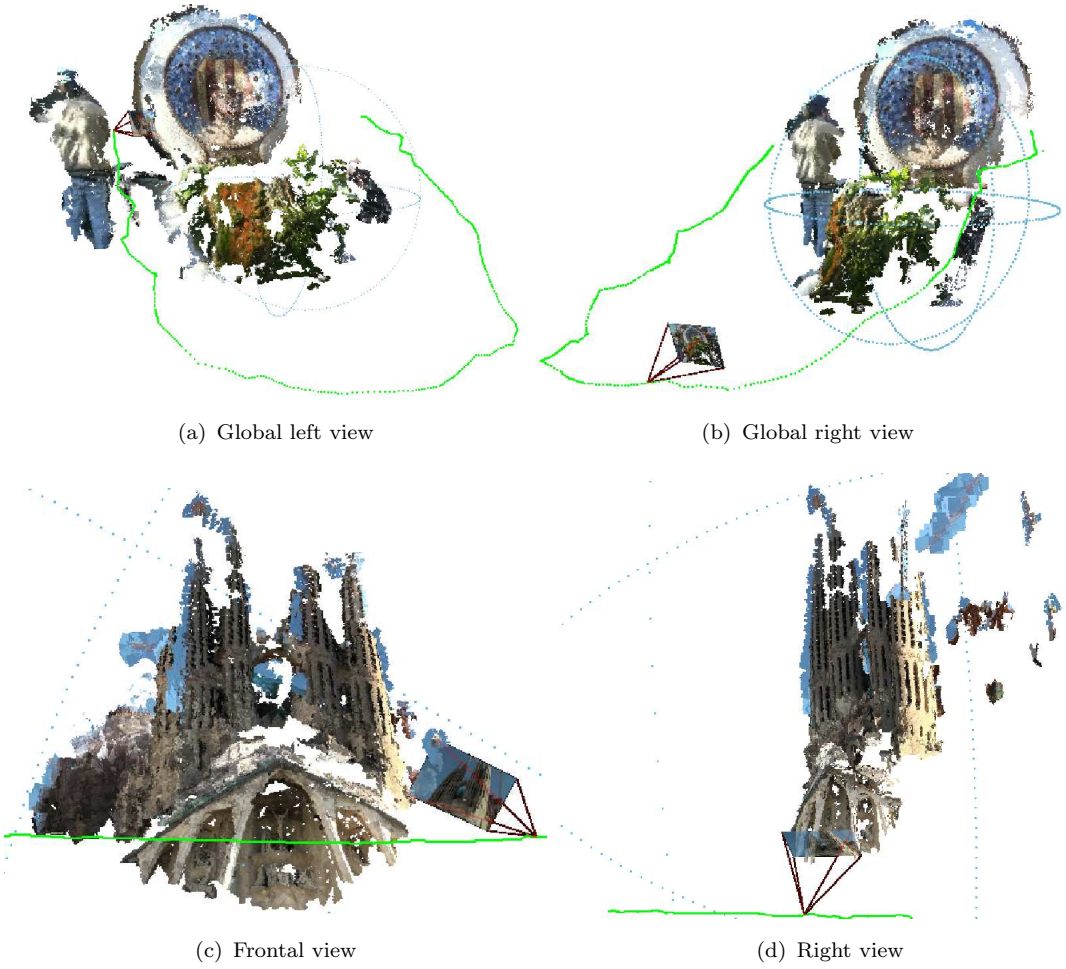
(a) Global view (b) Extension of the 2D world

(c) Left view (d) Zenital view

FIGURE 5.7: Dense 3D models for Cat and Dragon sequences

advantage is that the amount of computational resources needed is significantly less than for the global Bundle Adjustment.

Finally, let us note that it is not only a matter of a reduction on computational complexity. If we focus on the P-LIN result for Snake sequence, there is a great difference between the number of 3D points with respect to the rest of registering techniques. Bundle Adjustment requires from a good initialization to achieve its best performance and as seen for large error values in the LIN configuration is able to dramatically reduce them. However, if a good starting point is not provided, the solution may fall to a local minimum of the merging process. For this particular case, the aforementioned situation occurred in the fifth iteration. The consequence was that about 14.000 points were deleted due to a large reprojection error or a small triangulation angle.



(a) Global left view (b) Global right view

(c) Frontal view (d) Right view

FIGURE 5.8: Dense 3D models for Snake and Sagrada Familia Sequences

5.4 Qualitative Results

Up to this point, the reliability and accuracy of the proposed algorithms have been numerically proven. Now we want to validate the full reconstruction pipeline described along this work from a visual appearance point of view.

As seen in figures 5.3, 5.4, 5.5 and 5.6, the retrieved cloud of points barely allows to identify the sequence that we have reconstructed. Hence, one may use a densification algorithm to provide a better visual results by using the obtained information during the SaM stage. In our case, we use the retrieved pose of the cameras marked as KF, as the input for Furukawa's patch-based densification algorithm [32].

The basic idea behind this dense algorithm is the following: given a set of calibrated cameras, define a dense set of points obtained by applying a grid over an image and track them along the rest of views with an optical-flow approach.

Next robustly triangulate the features, and track them according to some photo-consistency measurement. The key idea is that the set of points from this approach is denser than for SIFT-like features since they are pre-defined, not detected. The final step consists on analyzing the neighborhood of each one of the triangulated points, and obtain and approximation for its orientation. This orientation serves to correctly place an small patch approximating the surface where the point was triangulated. Furthermore, photo-realistic effects may be achieved by projecting the texture in images to the defined 3D patch.

The presented algorithm requires from a very accurate set of cameras' pose. Therefore, this is also a way to validate the correctness of the obtained the cameras with our SaM pipeline. In Fig's. 5.7 and 5.8, visual results for Furukawa's dense algorithm using our camera calibrations are presented. As seen, we have obtain very attractive results for the four sequences studied. For instance, Fig. 5.7-b represents the 3D world captured from the camera center in a certain time index. As seen, the obtained dense 3D world, perfectly extends the 2D image plane. This is a consequence of the accuracy of the retrieved camera pose. Photo-realistic results were also obtained for the other considered sequences.

Chapter 6

Conclusions and future work

Along this Thesis, a full Structure And Motion pipeline has been designed and implemented. The presented algorithms were designed with the aim of tackling the two main problems for these types of systems: scalability and robustness.
A novel frame decimation scheme has been proposed in order to provide the reconstruction step with a small set of relevant and well conditioned KFs. Moreover, the frame decimation stage has allowed defining a criterion for spliting the input video sequence into smaller parts of a fixed number of KFs. That has allowed the parallel reconstruction strategy detailed in chapter 3. Also with regards to the reconstruction loop, the feature tracking process has been embedded within the structure computation and camera pose estimation stages as a difference with the majority of approaches.
Results have validated our reconstruction algorithms and have shown them to outperform other strategies. More precisely, the frame decimation step has proved to accurately adapt to the dynamics of the input video sequence in order to provide a set of meaningful selected frames. Its combination with the proposed feature tracking technique, has provided a reliable and robust reconstruction strategy capable to handle different types of sequences, captured with devices of different quality.
With respect to the *divide and conquer* approach proposed, a set novel linear algorithms, that is able to combine point and camera correspondences, has been defined to permit the registration of independent partial reconstructions of the same static scene. Furthermore, the optimization strategy when registering partial reconstructions has been designed and analyzed.
Provided results for real sequences have demonstrated that, given a set of optimized 2D reconstructions and a set of connecting variables (common points and

cameras between reconstruction), is it possible to perform a local instead of an expensive global optimization with comparable performance results. This strategy has reduced the total amount of computational resources needed for the reconstruction process.

Finally, results for the whole pipeline have been used to feed a densification algorithm in order to prove the overall performance of the system. Photorealistic effects were achieved using our retrieved 3D information, validating this way our reconstruction system.

In the future, there are some points that we would like to tackle. First of all, we have proposed a technique for reducing the computational complexity of the full reconstruction system. Therefore, it would be interesting to accurately study the amount of computational save that our approach is achieving with respect to other strategies.

Next, the possibility of extending the system to allow collaborative reconstructions between users is something that we are very interested in implementing. Our current registration system based on point correspondeces may allow the common registration of the same static scene acquired from different users. Nevertheless, in order to obtain a fully authomatic system, some efforts need to be devoted in the fields of scene recognition or image retrieval. The system is intended to be able to identify whether the sequences uploaded by different users refer to the same static scene and, hence, merge their reconstructions.

Finally, the visually appealing results obtained with Furukawa's dense algorithm, have motivated us to develop our own densification algorithm which will adapt to the specific requirements of our project. Several strategies will be studied including meshes, patches or dense stereo.

Bibliography

[1] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. ISBN 0521623049.

[2] N. Snavely, S.M. Seitz, and R.S. Szeliski. Modeling the world from internet photo collections. *IJCV*, 80(2), November 2008.

[3] R. Gherardi A. M. Faranzaena, A. Fusiello. Structure-and-motion pipeline on a hierarchical cluster tree. In *Proceedings of the IEEE International Workshop on 3-D Digital Imaging and Modeling*, October 2009.

[4] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image Vision Comput.*, 27(8):1178–1193, 2009.

[5] Xiaowei Li, Changchang Wu, Christopher Zach, Svetlana Lazebnik, and Jan-Michael Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, pages 427–440, Berlin, Heidelberg, 2008. Springer-Verlag.

[6] Microsoft Photosynth. URL http://www.phosynth.net.

[7] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *IJCV*, 78(2-3):143–167, 2008.

[8] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. *ECCV*, pages 311–326, 1998.

[9] Changchang Wu, Brian Clipp, Xiaowei Li, Jan-Michael Frahm, and Marc Pollefeys. 3d model matching with viewpoint-invariant patches (vip). *CVPR*, 0:1–8, 2008.

[10] David Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. *ECCV*, pages 649–663, 2000.

[11] David Nistér. Frame decimation for structure and motion. In *SMILE '00: Revised Papers from Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, pages 17–34, London, UK, 2001. Springer-Verlag. ISBN 3-540-41845-8.

[12] M.I.A. Lourakis and A.A. Argyros. *The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm.* Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004.

[13] Kai Ni, Drew Steedly, and Frank Dellaert. Out-of-core bundle adjustment for large-scale 3D reconstruction. *ICCV*, 2007. URL http://frank.dellaert.com/pubs/Ni07iccv.pdf.

[14] S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R.S. Szeliski. Building rome in a day. *ICCV*, 2009.

[15] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. *ECCV*, pages 311–326, 1998.

[16] David Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. *ECCV*, pages 649–663, 2000.

[17] Changchang Wu, Brian Clipp, Xiaowei Li, Jan-Michael Frahm, and Marc Pollefeys. 3d model matching with viewpoint-invariant patches (vip). *CVPR*, 0:1–8, 2008.

[18] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 13(2):119–152, 1994.

[19] W. Zhao, D. Nistér, and S. Hsu. Alignment of continuous video onto 3d point clouds. *CVPR*, 2:964–971, 2004.

[20] R.S. Kaminsky, N. Snavely, S.M. Seitz, and R. Szeliski. Alignment of 3d point clouds to overhead images. *CVPR Workshop*, 0:63–70, 2009.

[21] D.G. Lowe. Distinctive image features from scale-invariant keypoints. 60(2): 91–110, November 2004.

[22] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23, 2006.

[23] E. Kruppa. Zur ermittlung eines objektes aus zwei perspektiven mit innerer orientierung. *Sitz.-Ber. Akad. Wiss., Wien, Math. Naturw.*, 1913.

[24] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. ISSN 0001-0782.

[25] Chris Harris and Mike Stephens. A combined corner and edge detector. In *The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[26] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[27] T. Adamek D. Marimon, A. Bonnin and R. Gimeno. DART: Efficient scale-space extraction of DAISY keypoints. *Submitted to CVPR*, 2010.

[28] Richard I. Hartley. Estimation of relative camera positions for uncalibrated cameras. pages 579–587. Springer-Verlag, 1992.

[29] Marc Pollefeys, Luc J. Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[30] José I. Ronda, Antonio Valdés, and Guillermo Gallego. Line geometry and camera autocalibration. *JMIV*, 32(2):193–214, 2008.

[31] Zhengyou Zhang and Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, 1998.

[32] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, 2007.