Universitat de Girona

EFFICIENT 3D SCENE MODELING AND MOSAICING

Tudor Mircea NICOSEVICI

Universitat de Girona

PhD Thesis

Efficient 3D Scene Modeling and Mosaicing

Tudor Nicosevici

2009

Doctoral Programme in Technology

Supervised by: Rafael Garcia

Work submitted to the University of Girona in fulfilment of the
requirements for the degree of Doctor of Philosophy.

Abstract

Scene modeling has a key role in applications ranging from visual mapping to augmented reality. This thesis presents an end-to-end solution for creating accurate 3D textured models using monocular video sequences, with contributions at different levels.

First, we discuss a method developed within the framework of sequential Structure from Motion, where a 3D model of the environment is maintained and updated as new visual information becomes available. The camera pose is recovered by directly associating the 3D scene model with local image observations, using a dual registration approach. Compared to the standard Structure from Motion techniques, this approach decreases the error accumulation while increasing the robustness to scene occlusions and feature association failures, allowing 3D reconstructions for any type of scene.

We also develop an online approach for measuring similarities between images. In this way, images corresponding to the same scene region can be associated, allowing the reduction of drift and position uncertainties for mapping and navigation. Inspired from content-based image retrieval, the proposed approach uses visual vocabularies to represent images as occurrences of visual words. The technique is entirely sequential and automatic, making it suitable for online applications, such as robot navigation and mapping: (i) the vocabularies are built and updated online, during image acquisition, in order to efficiently represent the visual information present in the scene, and (ii) the vocabulary building and image indexing processes do not require any user intervention.

Lastly,motivated by the need to map large areas, we propose an online 3D model simplification algorithm. The simplification process uses plane-parallax to estimate the geometry of the scene, eliminating the necessity of explicit scene shape information. Such an approach offers two main advantages: (i) it is suited for online applications, where it can run parallel with the 3D reconstruction process, and (ii) as it does not require having the full model prior to the simplification, the algorithm allows mapping of larger, more complex scenes.

We discuss the efficiency of the proposals and compare them with other state of the art approaches, using a series of challenging datasets both in underwater and outdoor scenarios.

Resum

El modelat d'escenes juga un paper clau en un gran ventall d'aplicacions que van des de la generació mapes fins a la realitat augmentada. Aquesta tesis presenta una solució completa per a la creació de models 3D amb textura utilitzant seqüències de vídeo monoculars, proposant contribucions a diferents nivells.

En primer lloc es presenta un mètode desenvolupat en el context de *Structure from Motion* seqüencial, a on el model 3D de l'entorn és manté i s'actualitza a mesura que es va disposant de nova informació visual. La posició de la càmera es calcula directament registrant cadascuna de les imatges de la seqüència amb el model 3D de l'escena. En comparació amb les tècniques estàndard per obtenir l'estructura de l'escena, aquesta proposta redueix l'error acumulat essent més robusta en presència d'oclusions, i tolera millor els errors en l'associació de característiques, permetent la reconstrucció 3D de qualsevol tipus d'escena.

També s'ha desenvolupat un mètode *online* per a la mesura de similitud entre imatges. D'aquesta manera, les imatges corresponents a la mateixa regió de l'escena es poden associar, permetent la reducció de la deriva i la incertesa en la posició de la càmera, millorant la navegació i la construcció de mapes. Inspirat en la recuperació d'imatges basades en el contingut, el mètode presentat utilitza vocabularis visuals per a presentar les imatges com a aparicions de paraules visuals. La tècnica és completament seqüencial i automàtica, convertint-se en una eina adequada per a aplicacions *online*. Així mateix, els processos de generació del diccionari i indexat de les imatges no requereixen cap intervenció per part de l'usuari.

Motivat per la necessitat de construir mapes en àrees extenses, es proposa en aquesta tesis un algorisme de simplificació de models 3D. El procés de simplificació utilitza la informació que aporta el *parallax* per a l'estimació de la geometria de l'escena, eliminant la necessitat d'informació explícita sobre la morfologia de l'escena. Aquesta proposta ofereix dos avantatges: (i) és factible per a les aplicacions *online*, a on pot córrer de manera paral·lela amb la reconstrucció 3D, i (ii) donat que no requereix disposar del model complet abans d'iniciar la simplificació, l'algorisme permet generar mapes de grans dimensions d'escenes complexes.

L'eficiència de les propostes s'ha comparat amb altres mètodes de l'estat de l'art, utilitzant diversos conjunts de dades submarines i terrestres.

...to my parents.

Acknowledgements

First of all, I would like to thank Rafa, who always knew how to be not only a great supervisor, but also a good friend. Me writing this at the conclusion of the thesis, I owe much to his neverending enthusiasm, understanding and positive attitude.

I would also like to thank Professor S. Negahdaripour and Professor A. Zisserman for their guidance during my research stays at the University of Miami and the University of Oxford.

I am very grateful to my colleagues in the Underwater Vision Lab, for always being there, willing to lend a hand and for creating such an amazing working environment: Oli, without whose profiling and code optimization I would probably still be running my experiments; El Turco Peligroso, who always knew how to respond to any question with about four pages of equations; Nuno, who showed me that one can work at 3AM and still laugh about it; Pio, the guru of everything that can be represented in 0's and 1's; Josep, Ricard and all the others. Thanks, guys.

I also want to express my gratitude also to the "seniors", for being more than coworkers: Xcuf, Quim, Miki, Pere, Jorfif, the two Joans, Robert, Marc, Lladow, Figui and all the others who made me feel like home since the first day I arrived here.

Finally, my special thanks goes to my parents and family, who, though far away, were always close, always there, when I needed them the most.

Contents

List of Figures

vi

List of Tables

List of Acronyms

ANN Approximated Nearest Neighbor

AUV Autonomous Underwater Vehicle

BA Bundle Adjustment

BCM Brightness Constancy Model

BoW Bag of Words

CLAHE Contrast Limited Adaptive Histogram Equalization

CUDA Compute Unified Device Architecture

DLT Direct Linear Transformation

DoF Degrees of Freedom

DoG Difference of Gaussians

DPR-SfM Direct Pose Registration Structure from Motion

FFT Fast Fourier Transform

GDIM Generalized Dynamic Image Model

GPU Graphics Processing Unit

HD High Definition

HDR High Dynamic Range

IFREMER Institut Français de Recherche pour l'Exploitation de la Mer

ICP Iteratively Closest Point

JHU Johns Hopkins University

LDA Linear Discriminant Analysis

LMedS Least Median of Squares

LS Least Squares

MPEG Moving Picture Experts Group

MSER Maximally Stable Extremal Regions

NCC Normalized Cross Correlation

NN Nearest Neighbor

OMS Online Model Simplification

OVV Online Visual Vocabulary

PM Progressive Meshes

RANSAC Random Sample Consensus

ROV Remotely Operated Vehicle

RoI Region of Interest

SAD Sum of Absolute Differences

SfM Structure from Motion

SIFT Scale Invariant Feature Transform

SVD Singular Value Decomposition

SLAM Simultaneous Localization and Mapping

SNN Second Nearest Neighbor

SSD Sum of Squared Differences

SURF Speeded Up Robust Features

UoM University of Miami

UUV Unmanned Underwater Vehicle

UV Underwater Vehicle

Chapter 1

Introduction

1.1 Objectives

The aim of this thesis is to provide a complete framework for efficient 3D modeling. More specifically, given an image sequence of a scene, the objective is to provide a high-precision textured 3D reconstruction of the scene with virtually no human intervention.

The main focus of this thesis is on efficient modeling of 3D underwater scenes for scientific studies. Nevertheless, as shown in this work, we have successfully applied the technique in other areas of interest such as the reconstruction of small scale objects, outdoor natural scenes, urban environments, etc.

Although some successful 3D reconstruction algorithms have been reported in literature, they are limited to specific applications. Most techniques assume controlled or structured environments, where illumination, camera motion and scene geometry priors can be used. More importantly, these techniques can be applied to very limited scenes only, due to the complexity of the 3D reconstruction problem.

In contrast, we aim to develop an online generic framework for 3D scene re-

construction that can cope with wide areas of complex and highly unstructured environments. In order to achieve this, we focused on the following aspects:

Online process. The entire framework has been designed to process the data sequentially, enabling its use on online applications such as robot navigation and mapping.

Flexibility of acquisition. The 3D reconstruction algorithm uses image sequences that can be acquired by using any type of video/still cameras, with no constraints on the acquisition process. Moreover, the framework can readily cope with camera occlusions and temporary failures.

Stand-alone framework. While additional information can be integrated into the 3D reconstruction process, the framework does not require any additional sensor information. This increases the flexibility of the reconstruction process while decreasing the acquisition costs. In this way, underwater sequences can, for example, be acquired by using cameras mounted on inexpensive Remotely Operated Vehicles (ROVs) or even by divers using hand-held cameras. On the other hand, without absolute positioning sensors, vision systems are prone to drifting. We address this shortcoming by proposing a novel online cross-over detection system, that allows the detection of loops in camera trajectory along with camera pose[1] correction.

Efficient 3D modeling. The framework employs a novel online 3D model simplification algorithm, that allows mapping of larger, more complex scenes.

---

[1]Camera position and attitude (orientation).

1.2   Motivation

Throughout the history of the Earth, the most determinant element, that shaped it as we know it today, are the oceans. They are the origin of life on Earth and home of the widest biodiversity. Moreover, the oceans are the major factor in our climate, literally affecting almost every aspect of our daily lives.

Apparently paradoxical, the oceans represent the least studied region of the Earth's surface. The main reason behind this is the inaccessibility and hostility of this environment. This, however, is changing at a rapid pace. Our urge to find alternative food and energy sources, to understand climate changes and geological phenomena have determined the scientists to multiply their efforts into understanding this complex environment. Moreover, the latest technological advances provide the scientists with the basis for more efficient means to explore the underwater environment.

In this context, this thesis proposes a valuable tool for remote underwater studies. Images acquired by scuba divers using hand-held cameras can be used to obtain high detail textured 3D models of the seafloor. Using cameras mounted on Unmanned Underwater Vehicle (UUV) we can obtain 3D maps of high depth underwater regions that otherwise would be inaccessible to humans. Just to name a few, this proposal has applicability in (see Figure 1.1 for some examples of scientific oriented underwater imagery):

- **Biology.** Visual 3D maps of marine habitats provide important clues in studying the marine species and their interaction with the environment.

- **Ecology.** The impact of human activities on our environment has become a matter of great concern nowadays. Climate changes, intensive fishing and the destruction of habitats greatly affect the underwater biodiversity. In this

(a)
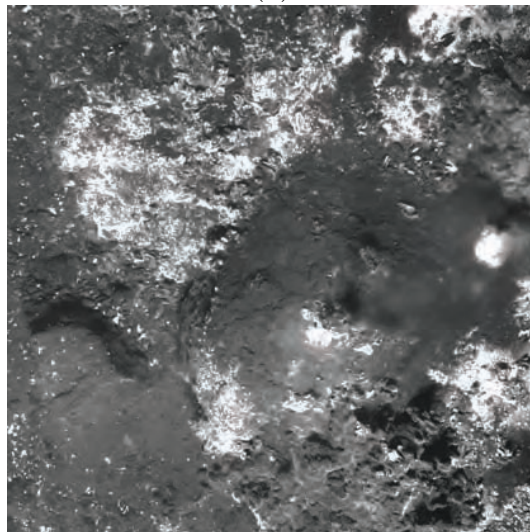
(b)

(c)

Figure 1.1: **Motivation – Scientific underwater imagery.** (a) a coral reef head near Bahamas; (b) underwater lava formations captured during MoMARETO'06 cruise – courtesy of IFREMER and (c) amphoras near Pianosa Island in the Mediterranean Sea – courtesy of Venus Project.

context, 3D models can be used to observe and monitor the changes that take place in the underwater habitats, such as coral reefs.

- **Geology.** Shape and texture of the regions with increased geological activity greatly aid geologists in order to understand the complex geological phenomena that take place underwater.

- **Archeology.** In our pursuit to understand history, we continuously search the depths of the oceans for new clues about our past. Unfortunately, in most cases the artifacts are too fragile or too inaccessible to be recovered for studying. In this case, 3D models can provide a viable solution for remote archeological studies.

Nowadays, an increasing number of underwater studies employ ROVs as an alternative to scuba divers. This eliminates the risks the divers are exposed to, especially in deep waters, while allowing more efficient studies. However, the use of ROVs poses a series of drawbacks: their operation requires specialized personnel and their range and depth is limited by the length of their umbilical cable, which connects the ROV with the ship. Various research groups have focused their efforts on developing underwater vehicles that would carry out missions autonomously. This requires that the vehicles be able to model the environment in order to navigate through it. The images acquired by cameras mounted on these vehicles can be processed and 3D maps of the environment can be obtained. Furthermore, the obtained maps can be used for navigation in subsequent missions where, for example, successive surveys of the same area are needed.

With the wide accessibility of high computational power, the use of wide area 3D modeling has become an area of interest in fields much closer to the end-user:

- **Urban 3D modeling.** Applications such as *Google Earth* [47] (see Figure

Figure 1.2: **Motivation – Urban architecture.** Google Earth view of downtown Miami depicting 3D models of the most iconic buildings.

1.2) or certain navigation applications such as *iGo* [63] offer 3D models of urban landmarks. The process of constructing the 3D models could be highly simplified by using automated 3D modeling techniques. Additionally, one could imagine applications where tourists would able to obtain 3D models along with information about the landmarks they are visiting.

- **Architecture.** Indoor / outdoor 3D models of buildings can be obtained for virtual marketing purposes. Also, using augmented reality, one could visualize beforehand the results of the restoration of a historical building for example, etc.

- **Virtual reality.** Computer games and virtual community applications such as *Second Life* [130] could be enriched with 3D models of real-life objects and buildings.

1.3   Challenges

The human brain interprets visual information provided by the eyes by generating 3D images of our surroundings. We use this information in order to orient ourselves and move through the environment. The flexibility and power of abstraction of the brain allows it to easily cope with constant challenges in our environment such as moving objects, lighting artifacts and so on. When it comes to computer vision however, things become ever more difficult. For this, in order to achieve flexibility and robustness, we need to address a series of challenges:

- Unstructured, natural 3D scenes consist of large amounts of objects with diverse shapes and textures. As the camera moves through the scene, objects constantly occlude each other (see Figure 1.3).

- Light changes (*e.g.* motion of the light source), moving shadows, altering of light reflections in specular surfaces due to point of view changes drastically modify the photometric properties of the scene. This effect is particularly emphasized in underwater scenes, where sun flicker (changes in light pattern due to sunlight being refracted on moving sea surface) dramatically changes the illumination field (refer to Figure 1.4 for details).

- Moving objects such as cars and pedestrians in urban environments or fishes and algaes in underwater environments (see Figure 1.5) violate the rigid scene assumption, inducing errors in scene geometry estimation.

All these are common challenges faced by computer vision systems. However, the underwater environment poses specific challenges that make underwater imagery a particularly difficult task (Figure 1.6):

- In water, light suffers a much higher rate of attenuation than in atmospheric
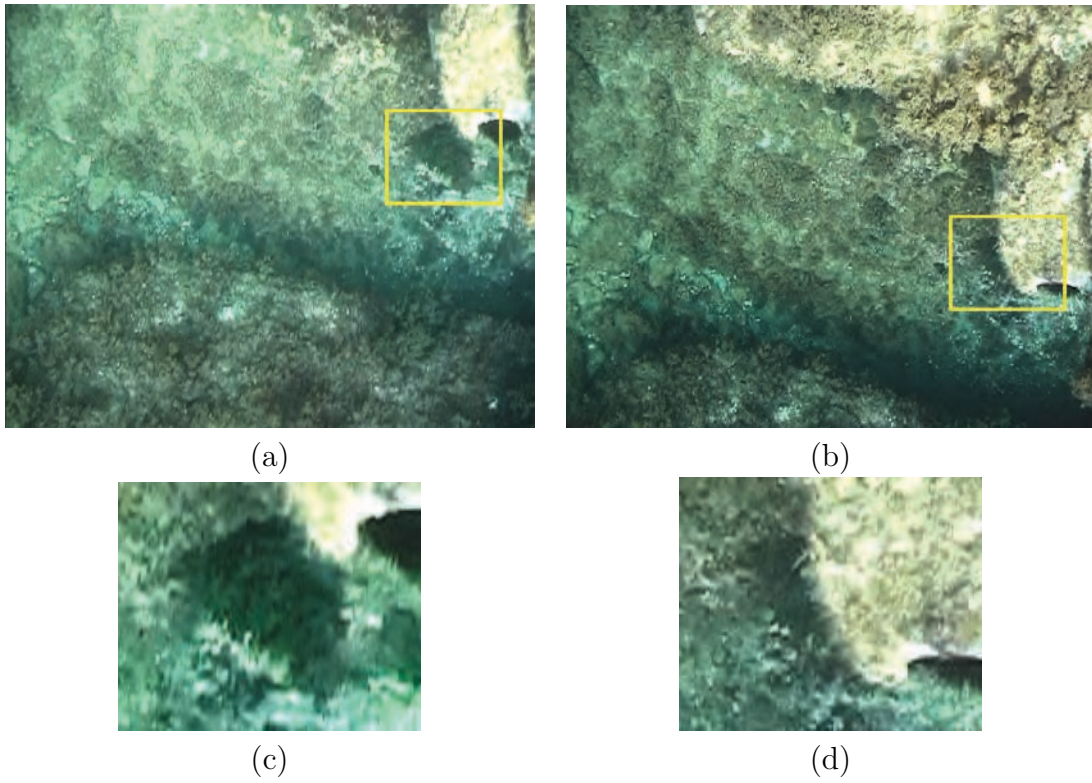
Figure 1.3: **Challenges – Scene occlusions.** (a) and (b) show the same underwater scene from two different camera view points; The region marked by the yellow rectangles is shown in detail in (c) and (d). Note the rock is clearly visible in (a) and (c) but almost occluded in (b) and (d).

conditions. This limits the maximum distance between the camera and the scene, resulting in a narrow coverage of the camera. In order to cover wide scene areas, large amounts of images have to be merged. Furthermore, due to light attenuation, at great depths, additional illumination sources have to be used. Generally, underwater vision systems employ focus lights as the latter can illuminate the scene at greater ranges. The drawback of the focus lights, however, is that they induce highly non-uniform lighting fields [42] (refer to Figure 1.6a for details).

- The contrast of underwater images is reduced due to light absorbtion, decreas-
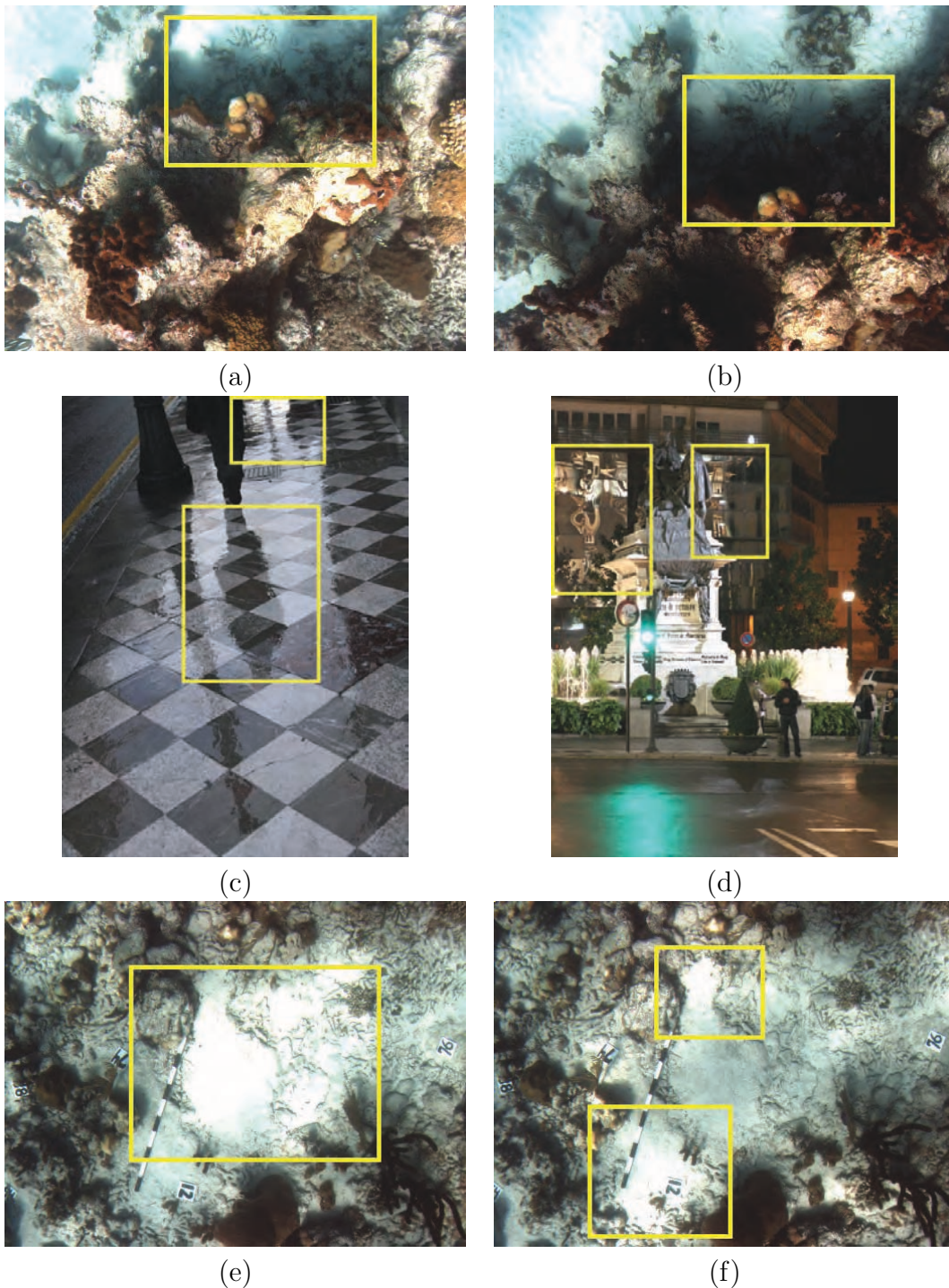
Figure 1.4: **Challenges – Light artifacts.** (a) and (b) illustrate an underwater scene a few frames apart. The details of the outlined region are visible in (a) while not distinguishable in (b) due to shadowing. Specular surfaces such as wet pavement (c) and windows (d) are highly reflective, inducing lighting artifacts. (e) and (f) show two frames of an underwater scene taken only $150ms$ apart. The light pattern changes drastically due to sun flickering.
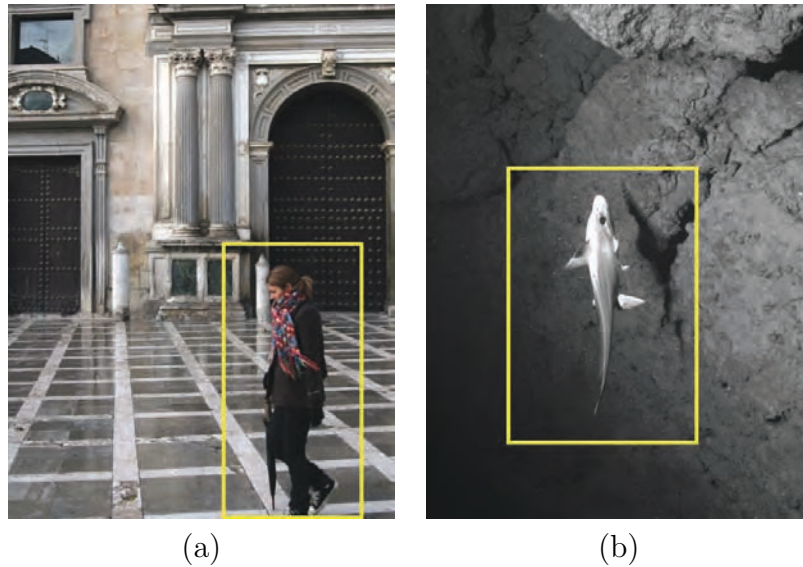
Figure 1.5: **Challenges – Moving objects.** Moving objects such as pedestrians in urban scenes (a) and fishes in underwater scenes (b) violate the rigid scene assumption.

ing the signal-to-noise ratio (see Figure 1.6b).

- Small suspended particles present in the sea water such as plankton and sediments generate the so called *scattering effect*. Practically, the scattering effect takes place due to the light changing direction when it enters in contact with the particles. The forward scattering bends the light beams traveling from the scene towards the camera, resulting in a blurring effect that reduces the level of detail of the images (Figure 1.6c). On the other hand, the backward scattering refracts the light from the light source towards the camera decreasing the contrast and inducing noise in the images (Figure 1.6d).

In addition to environment challenges, vision systems pose specific fundamental problems:

- Our aim is to develop a 3D modeling system where the camera can move freely through the scene. In order to maintain the generic character of the proposal, we assume that no external sensor information is used. In this case, scene
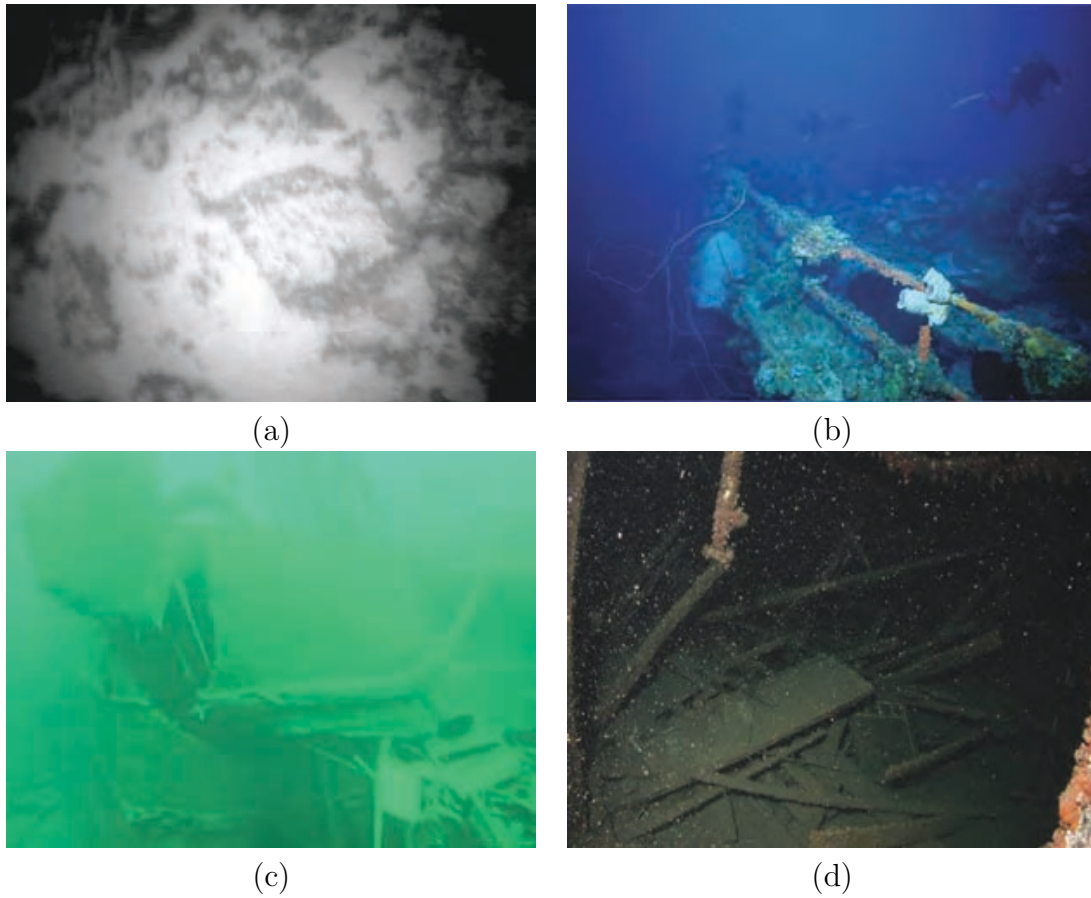
Figure 1.6: **Challenges – Underwater environment.** (a) the use of additional illumination sources induce non-uniform illumination fields (a); (b) light attenuation limits the range of the camera: the two divers in the background are hardly visible; (c) forward scattering blurs and decreases the contrast of underwater images and (d) backward scattering induces image noise.

geometry and camera motion are computed incrementally. Over wide scenes, small errors in the camera pose and scene geometry estimation build up over time. This error build-up can lead to estimations that drift away from the reality.

- Scene models are composed of 3D vertices. These vertices can be seen as discrete samples of the surfaces that are present in the scene. Over wide scene areas, millions of such vertices can be generated. Such large number of scene

samples can prove too large to be effectively managed by vision systems.

1.4   Contributions

This thesis has contributions at different levels, briefly described hereafter. A more extensive account of the contributions is presented in Chapter 6.

1.4.1   Structure from Motion

We develop a novel Structure from Motion (SfM) algorithm, where the scene model is generated using a two-step approach: (i) camera pose is directly obtained from the scene model and (ii) using the camera pose, the scene model is updated and extended. This approach reduces the accumulation of error and results in more accurate scene models.

Also, we propose a novel dual camera pose recovery method, which allows SfM algorithm to successfully cope with both planar and non-planar scenes.

1.4.2   Ortho-mosaic and Rendering

We propose a novel approach to generate synthetic 2D visual maps – *ortho-mosaics*. By exploiting the geometry of the scene, the approach takes into account surface normals and camera poses in order to assure maximum resolution and minimum distortions during the ortho-mosaic rendering. This method results in accurate and visually pleasant scene maps.

1.4.3   Loop Closure Detection

Loop closures are situations where the camera revisits an already surveyed region. These regions allow us to impose additional constraints in the geometry of the scene,

hence reducing the 3D estimation errors. This thesis proposes an online loop closing detection algorithm that uses Bag of Words (BoW) to measure visual similarities among camera frames. There are three main novelties that we propose here: (i) the visual vocabularies are built incrementally, enabling the use of the algorithm for online applications; (ii) the algorithm requires no training stage and no user intervention, and (iii) the feature clustering process uses a global data distribution criteria, resulting in more efficient visual vocabularies.

1.4.4 Vertex Selection

Scene models are formed from thousands to millions of 3D vertices. Most of these vertices are geometrically redundant (4 or more vertices laying on the same plane, 3 or more vertices laying on the same straight edge, etc.). We propose an online approach which analyzes the geometry of the scene and selects only those vertices that are geometrically representative for the scene. The method uses plane-parallax techniques that allow us to approximate the shape of the scene without explicitly recovering its geometry. In this way, feature selection can be carried out sequentially, as the scene model is being built.

The result is a 3D scene model with drastically reduced complexity that, at the same time, maintains the accuracy of the original model.

1.5 Thesis Outline

In Chapter 2 we review previous work on image registration techniques, visual feature extraction and matching, mosaicing and 3D reconstruction techniques. The review details those aspects of literature that are relevant to this thesis. Modern visual feature extractors and descriptors are described thoroughly as they constitute

the basis for the proposed 3D reconstruction framework. Also, other 3D reconstruction algorithms are discussed along with their limitations, illustrating the motivation behind this work.

Chapter 3 presents the proposed 3D reconstruction algorithm. The first part of the chapter provides a detailed description of each step of the algorithm. The algorithm is validated through a series of experiments presented in the last part of the chapter. In here, we discuss various experiments that we have carried out in underwater, natural and structured environments.

In Chapter 4, we present the cross-over detection algorithm. The algorithm is built on top of the 3D reconstruction process and allows online detection of loops in the camera trajectory. In the first part of the chapter, we provide a review of the literature regrading the cross-over detection problem. Next, a detailed description of the proposed algorithm is provided. Finally, we present a series of experiments along with a comparison with a state of the art loop closure detection algorithm.

Chapter 5 discusses the online model simplification algorithm, that works in parallel with the 3D reconstruction algorithm. First, we discuss the existing work related to 3D model simplification, followed by a detailed description of our simplification algorithm. The chapter concludes with a series of experimental results and comparison with a widely used model simplification algorithm.

Finally, Chapter 6 summarizes the contributions of the thesis and discusses ongoing and future work. This chapter also presents the publications of the author, that are most significant to the development of this thesis.

Chapter 2

Literature Review

Scene modeling has represented one of the most fundamental problems of computer vision since its birth, four decades ago. Despite this, until not long ago, scene modeling was more of an exploratory field with very limited applications. Recent advances in both hardware and algorithms, however, have increased the popularity of scene modeling within the scientific community. Applications of this field have become part of our everyday lives. *Google Earth*, for example, allows anyone with a computer and an Internet connection to take an instant virtual trip to any place on Earth.

This chapter presents a brief review of most the representative techniques in the general context of scene modeling.

2.1   Image Registration

Determining the transformations that take place between images as camera viewpoint changes is an essential problem in computer vision. This is widely known as the *image registration problem*. It constitutes the basis for camera motion estimation and scene modeling.

Image registration has been largely discussed in the literature, where a series of authors have proposed methods to tackle this problem. Largely, these methods can be classified into: frequency domain based methods (Fourier transform), dense methods (optical flow) and sparse methods (feature based), discussed hereafter.

2.1.1 Frequency Domain

Originally, frequency-based methods used phase-correlation in order to estimate the shifts between two images. This was later extended to account for rotation and scale transformations [118] and even affine transformations [148] using log-polar coordinates. A few authors have proposed the use of frequency domain methods for underwater image registration [123, 124].

However, frequency domain methods are computationally expensive, as they require Fast Fourier Transform (FFT) to be computed over the entire images.

2.1.2 Optical Flow

Optical flow methods estimate the disparity (apparent motion) of pixels between pairs of images. Generally, optical flow estimates the image flow field using the Brightness Constancy Model (BCM), in which it is assumed that the photometric properties (intensity and color) remain constant.

There are two main approaches in estimating the optical flow: global methods such as Horn-Schunck [61] which yield dense flow fields, and local methods such as Lucas-Kanade [80, 81] that produce non-dense regularized grid flow fields but are more robust to noise.

Lucas-Kanade is one of the most widely used methods based on the local Taylor

series approximation using partial spatial and temporal derivatives. It considers:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \xi,$$

where $I(x, y, t)$ is pixel intensity at coordinates (x, y) at time t and ξ is a remainder (small enough to be ignored). Making use of the BCM assumption along frames, we have

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0$$

or

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0$$

therefore,

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y = -\frac{\partial I}{\partial t}$$

Using I_x, I_y and I_t as the spatial and temporal derivatives we obtain $-I_t = I_xV_x + I_yV_y$ or simply $-I_t = \nabla I \cdot \vec{V}$ which is an equation that imposes a single constraint with two unknowns, thus not solvable as is. However, assuming constant flow within small windows, for instance, over 3×3 pixels, we can obtain a set of 9 equations:

$$I_{x_{11}} \cdot V_x + I_{y_{11}} \cdot V_y = -I_{t_{11}}$$
$$I_{x_{12}} \cdot V_x + I_{y_{12}} \cdot V_y = -I_{t_{12}}$$
$$I_{x_{13}} \cdot V_x + I_{y_{13}} \cdot V_y = -I_{t_{13}}$$
$$\vdots$$
$$I_{x_{33}} \cdot V_x + I_{y_{33}} \cdot V_y = -I_{t_{33}}$$

Therefore, we can construct an over-determined system of $3 \times 3 = 9$ equations:

$$\begin{pmatrix} I_{x_{11}} & I_{y_{11}} \\ I_{x_{12}} & I_{y_{12}} \\ \vdots & \vdots \\ I_{x_{33}} & I_{y_{33}} \end{pmatrix} \cdot \begin{pmatrix} V_x \\ V_y \end{pmatrix} = \begin{pmatrix} -I_{t_{11}} \\ -I_{t_{12}} \\ \vdots \\ -I_{t_{33}} \end{pmatrix}$$

This over-determined system $B \cdot \vec{v} = -b$ can be solved in a Least-Squares sense giving $B^{\mathsf{T}} \cdot B \cdot \vec{v} = B^{\mathsf{T}}(-b)$ and, therefore, $\vec{v} = (B^{\mathsf{T}} B)^{-1} B^{\mathsf{T}} (-b)$. Hence,

$$\begin{pmatrix} V_x \\ V_y \end{pmatrix} = \begin{pmatrix} \sum I_{x_{ij}}^2 & \sum I_{x_{ij}} \cdot I_{x_{ij}} \\ \sum I_{x_{ij}} \cdot I_{x_{ij}} & \sum I_{y_{ij}}^2 \end{pmatrix}^{-1} \cdot \begin{pmatrix} -\sum I_{x_{ij}} \cdot I_{t_{ij}} \\ -\sum I_{y_{ij}} \cdot I_{t_{ij}} \end{pmatrix}$$

Local optical flow methods yield a vector direction for each considered patch in the image.

In recent years, some authors have proposed better alternatives to BCM that assume linear changes in illumination – Generalized Dynamic Image Model (GDIM) [100, 106], and color [84, 101].

However, due to the formulation of the problem, the optical flow methods are not suitable for disparities that exceed 1 pixel. The solution for this is to use multi-resolution approaches [105]. Here, the images are gradually decimated and the optical flow is computed from coarse levels towards fine levels. This approach has its own drawback: it is slow (optical flow has to be computed at each level) and the maximum pixel disparity has to be known *a priori* in order to set the number of decimation levels. Also, multi-resolution approaches are very sensitive to noise, since errors in the estimation of optical flow at coarse levels will propagate to fine levels.

2.1.3 Feature Based

Feature based image registration methods focus on certain regions in the images, rather than images as a whole. Tracking the changes that these regions (features) suffer between images allows accurate image registration. For this, image features must be: *(i) repetitive* – features can be correctly tracked among images even in the presence of point of view and illumination changes and *(ii) discriminative* – in the sense that they can be uniquely matched in images.

Image registration based on image features involves three steps:

Feature detection. Extraction of the regions of interest in the images corresponding to image features such as: point features, line (edge) features, blob features, etc.

Feature description. Characterization of the image features in order to correctly identify and associate them.

Feature matching. Association of image features corresponding to the same region in the scene.

Hereafter, we discuss some of the most popular image feature detection, description and matching techniques, providing a detailed description of those used as basis for this work.

Feature detection

During this step, the actual locations of image pixels corresponding to the visual features are extracted (see Figure 2.1 for a comparison between different types of feature detectors). The outcome of the step depends on the image content and the type of feature extractor. However, regardless of these factors, the features have to be highly distinguishable from their neighborhood.
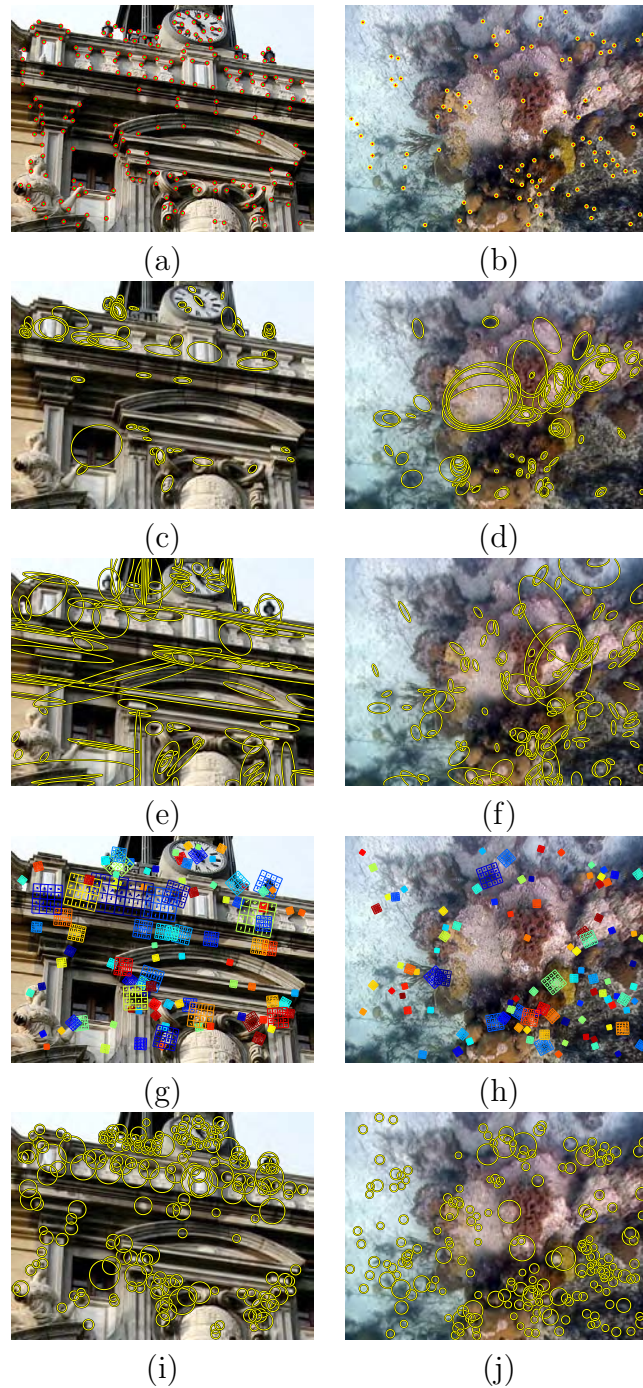
Figure 2.1: **Examples of feature fetectors.** Examples of features extracted from two different scenes – urban (left column) and underwater (right column): Harris (a) and (b), Hessian (c) and (d), MSER (e) and (f), SIFT (g) and (h) SURF (i) and (j). Number of extracted features was highly reduced for illustration clarity.

Harris Corner Detector is historically the most widely used point feature detector. Originally developed by Harris and Stephens in 1988 [55] to extract corner regions in structured environments (hence its name), it was successfully applied to all sorts of scenes. Harris Corner Detector extracts image points with high gradient in both X and Y directions. These points are locally discriminative in the sense that image patches centered in the points are highly dissimilar to any neighboring patches.

Harris Corner Detector uses the second moment matrix (also called the autocorrelation matrix) for feature extraction:

$$C(x, y) = \begin{bmatrix} I_x^2(x, y) & I_x I_y(x, y) \\ I_x I_y(x, y) & I_y^2(x, y) \end{bmatrix}$$

where:

$$I_x^2 = \sum_W (I_x(x_i, y, i))^2$$

$$I_y^2 = \sum_W (I_y(x_i, y, i))^2$$

$$I_x I_y = \sum_W I_x(x_i, y, i) I_y(x_i, y, i)$$

Matrix $C(x, y)$ captures the intensity structure of the local neighborhood. Its eigenvalues represent two principal intensity changes in the neighborhood of a point. Harris features points correspond to local maxima of K_{Harris}. Such points are invariant to rotation and arbitrary lightning changes.

$$K_{Harris} = \frac{I_x^2 I_y^2 - (I_x I_y)^2}{I_x^2 + I_y^2 + \varepsilon}$$

where ε is a small scalar added to avoid division by 0.

Hessian blob detector was one of the first image feature detectors. Proposed by Beaudet in 1978 [10], it represents the basis for many recent corner detectors.

The Beaudet operator is a rotationally invariant measurement of cornerness given by the determinant of the Hessian matrix M which represents a second-order partial derivative of an image I:

$$M(x,y) = \begin{bmatrix} I_{xx}(x,y) & I_{xy}(x,y) \\ I_{xy}(x,y) & I_{yy}(x,y) \end{bmatrix}$$

The second derivatives used in the Hessian matrix correspond to blobs and ridges[1], being represented by the local maxima of $K_{Hessian}$:

$$K_{Hessian}(x,y) = I_{xx}(x,y)I_{yy}(x,y) - I_{xy}^2(x,y)$$

Harris affine and hessian affine detectors are robust to image noise and invariant to rotation and lighting changes. However none of them is invariant to scale and affine transformations [92]. This makes them ineffective in wide base-line image registration where changes in camera viewpoint can induce significant geometric transformations.

Mikolajczyk et al. [94] have proposed adaptations of both Harris and Hessian feature extractors that are invariant to scale changes and affine transformations. In order to cope with scale changes, they propose the use of an scale selection method based on Laplacian. The idea is to select a scale that is characteristic to the local structure. For this, the Harris autocorrelation matrix is modified to include scale information:

---

[1]Blobs and ridges are compact image regions which differ from the background in terms of intensity, color or texture characteristics.

$$C_{Affine}(\mathbf{x}, \sigma_I, \sigma_D) = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x}, \sigma_D) & I_x I_y(\mathbf{x}, \sigma_D) \\ I_x I_y(\mathbf{x}, \sigma_D) & I_y^2(\mathbf{x}, \sigma_D) \end{bmatrix}$$

The local image derivatives are computed using Gaussian kernels of scale σ_D and averaged by smoothing with a Gaussian window of scale σ_I.

In the case of the Hessian feature extractor, the second order matrix becomes:

$$M_{Affine}(\mathbf{x}, \sigma_D) = \begin{bmatrix} I_{xx}(\mathbf{x}, \sigma_D) & I_{xy}(\mathbf{x}, \sigma_D) \\ I_{xy}(\mathbf{x}, \sigma_D) & Iyy(\mathbf{x}, \sigma_D) \end{bmatrix}$$

The affine shape of the neighborhood around the feature points in both Harris and Hessian cases is estimated using an iterative method using the eigenvalues of the second moment matrix.

SIFT detector uses Laplacian of Gaussian to extract image features that correspond to high gradient regions. However, in order to decrease the computational load, the Laplacian of Gaussian operator is approximated by Difference of Gaussians (DoG). The use of DoG was proposed by Lowe in [79] for both feature extraction and scale selection. For this, an image I is convolved with Gaussian filters at different scales, and the differences between successive Gaussian-blurred images are taken (see Figure 2.2):

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma)$$

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$$

where $G(x, y, k\sigma)$ is the Gaussian kernel at scale $k\sigma$.

Specifically, image I is convolved with different kernels by successively increasing k. The convolved images are grouped by octaves, which correspond to doubling
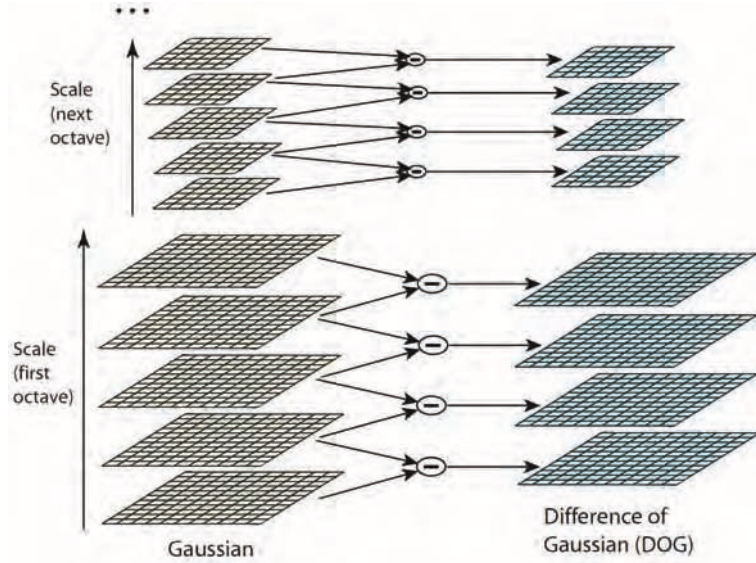
Figure 2.2: **SIFT feature detection.** Laplacian of Gaussian is approximated by DoG. The image I is convolved with Gaussians at different levels. Adjacent Gaussian images are subtracted to obtain DoG.

the value of k. The DoG is obtained by simply subtracting two adjacent convolved images. By stacking all the DoG's, we obtain a 3D space where the first two dimensions are given by x and y and the third dimension is the scale (this space is referred to as the scale-space). The keypoint features are obtained by extracting local extrema in the scale-space. In this way, the method extracts image features that are invariant to scale changes.

However, the precision of the extracted image features is limited by the resolution of the image I. In order to increase the precision of the feature extractor, Lowe proposes the use of a quadratic Taylor expansion of the scale-space, thus obtaining sub-pixel accuracy.

SURF detector uses the determinant of Hessian matrix for selecting both, the location of the keypoint and its scale [8]. The Hessian matrix $M(x, y, \sigma)$ of image I at point (x, y) is given by:
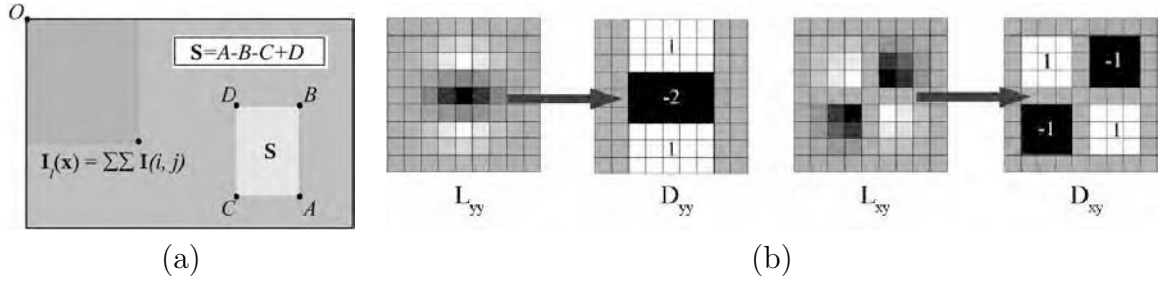
Figure 2.3: **SURF feature detection.** (a) Integral image calculation scheme. (b) Approximation of the second-order derivatives of Gaussian L_{yy} and L_{xy} with box filters D_{yy} and D_{xy} (mean / average filter)

$$M(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix}$$

L_{xx}, L_{xy} and L_{yy} are the convolutions of the Gaussian second-order derivatives with the image I at point (x, y).

Motivating that, in practice, the Gaussians need to be discretized and cropped, thus producing aliasing, Bay et al. approximate second-order derivatives of Gaussian with box filters (Figure 2.3b) which are applied on the integral version of I. The entry of an integral image I_Σ at a location (x, y) represents the sum of all pixels in the input image I of a rectangular region formed by the point (x, y) and the origin (Figure 2.3a).

The scale-space is built by gradually increasing the block filter size and stacking the responses (M). The feature extraction in the scale-space is carried out in a similar fashion with the one proposed by Lowe [79].

As reported in the literature and determined by our experiments, SURF has very similar performance to SIFT, but has a significant cut in computational load.

MSER detector was proposed by Matas et al. for detecting blob regions in wide baseline image registration [88]. As defined by the authors, a maximal region

is a connected component of an appropriately thresholded image. In other words, MSER extracts compact regions whose pixels have either higher (bright blobs) or lower (dark blobs) intensity values than all the pixels in their surroundings.

In order to extract MSER regions, the image is binarized using gradually increasing intensity thresholds. The binarization is used to extract compact dark and bright regions. MSER selects only those regions whose area changes insignificantly over a large range of intensity thresholds. These areas prove to be highly stable in both illumination (linear and non-linear) and affine transformations.

Feature description

For accurate image registration, each image feature has to be characterized so that image features corresponding to the same scene region – also referred to as pre-image region [19] – can be correctly matched. Blob features are generally characterized by extracting some statistics on the intensity or color values of the pixels forming the blob. However, in the case of point and edge features the characterization is not done on the features themselves. As images are subject to noise, illumination and geometric changes, measuring only one pixel (point features) or a small amount of pixels (edge features) would be highly unstable. In these cases, the solution is to characterize these types of features using their surrounding pixels.

Much of the robustness of feature tracking to lighting changes, noise and image transformations is obtained by choosing appropriate feature characterization (feature description). An extensive survey and comparison of state of the art feature characterization methods is discussed in [8,93]. Hereafter, we detail two widely used feature descriptors employed in this thesis.

SIFT descriptor. The detector used by SIFT provides interest points that are translation and scale invariant. Rotation, illumination and affine invariance is managed by the descriptor. The SIFT descriptor calculates an orientation corresponding to the dominant gradient direction of the neighborhood of a feature. All the following calculations are done relative to this orientation, so that all the features corresponding to the same pre-image point are aligned in the same direction (rotation invariance). In order to calculate the orientation, the gradient magnitude m and orientation Θ are calculated for each pixel in the neighborhood of the feature:

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}$$

$$\Theta(x, y) = \tan^{-1} \frac{L(x, y + 1) - L(x, y - 1))}{L(x + 1, y) - L(x - 1, y)}$$

L is obtained by smoothing I with a Gaussian at the scale where the feature was extracted. The gradient orientations Θ are used to form an orientation histogram with 36 bins. The highest peak in the histogram is detected and any other local peaks within 0.8 of the highest peak are used to create keypoints with those orientations. Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy. As a result, each detected SIFT keypoint is characterized by a vector (x, y, s, m, Θ), where x, y represent the coordinates of the keypoint, s the scale, m the magnitude and Θ the orientation. In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation.

A Gaussian weighting function with σ equal to $0.5\times$ the width of the descriptor window is used to assign a weight to the magnitude of each sample point. Convolution with Gaussian avoids sudden changes in the descriptor with small changes

in the position of the window, and gives less emphasis to gradients that are further away from the feature, as these are most prone to misregistration errors.

The keypoint descriptor allows for significant shift in gradient positions by creating an orientation histogram for each of the 4×4 sample regions. The orientation histogram consists of 8 bins covering the 360 degree range of orientations. Each bin is multiplied by a weight of $1 - d$ for each dimension, where d is the distance from the sample to the central value of the bin, measured in units of the histogram bin spacing.

The descriptor is formed from a vector containing the values of all the orientation histogram entries. SIFT uses 4×4 array of histograms with 8 orientation bins in each, computed from a 16×16 sample array. Therefore, the SIFT feature descriptors have $4 \times 4 \times 8 = 128$ elements.

Finally, the feature vector is normalized, to reduce the effects of illumination changes.

SURF descriptor   assigns a reproducible orientation to each detected keypoint, in order to get invariance to rotation. For this, the Haar-wavelet responses in x and y directions are calculated (shown in Figure 2.4) in a circular neighborhood of radius $6s$ around the keypoint, where s is its scale. Accordingly, the size of wavelet filters is adjusted to the scale. Using integral images, only six operations are needed to compute the response in x or y direction at any scale. The side length of the wavelets is $4s$. Once the wavelet responses are calculated and weighted with a Gaussian ($\sigma = 2.5s$) centered at the keypoint, the responses are represented as vectors in a space with the horizontal response strength along the abscissa and the vertical response strength along the ordinate (corresponding to x and y axis in Figure 2.4). The dominant orientation is estimated by calculating the sum of
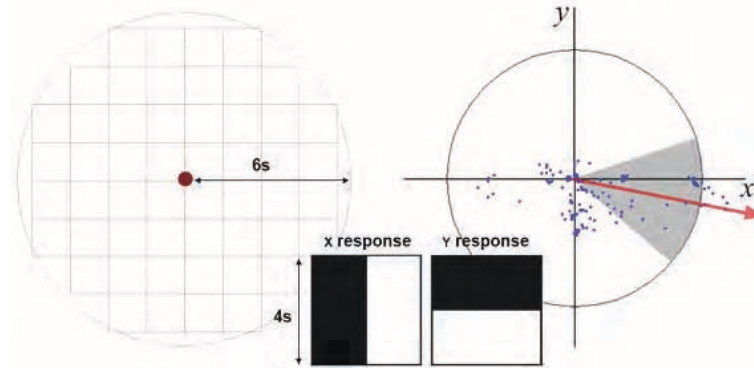
Figure 2.4: **SURF orientation assignment.** Left: Circular neighborhood of radius $6s$ around the keypoint, where wavelet responses are computed (s is a scale of the keypoint). Middle: Haar wavelet 2D filters used for SURF. Right: Representation space of the wavelet responses as vectors with coordinates x (horizontal response) and y (vertical response). The dominant orientation is estimated by calculating the sum of all vectors within a sliding orientation window covering an angle of $\pi/3$.

all responses within a sliding orientation window covering an angle of $\pi/3$. The horizontal and vertical responses within the window are summed, yielding a new vector. The longest vector lends its orientation to the interest point.

SURF descriptor is extracted by constructing a square region around each keypoint with the size $20s$, oriented along the keypoint orientation. The regions is then split into $16(4 \times 4)$ smaller square sub-regions and the Haar wavelet responses in horizontal direction (d_x) and vertical (d_y) are calculated (horizontal and vertical directions are relative to keypoint orientation as shown in Figure 2.5b). The wavelet responses d_x and d_y are weighted with a Gaussian ($\sigma = 3.3s$) centered at the keypoint, that increases the robustness to geometric deformations and localization errors. Then, d_x and d_y are summed up independently over each subregion to form a first set of entries in the feature vector. Furthermore, the sum of the absolute values of the responses $|d_x|$ and $|d_y|$ are extracted, providing information about the polarity of the intensity changes. Hence, each sub-region is represented by a four-dimensional descriptor vector $f = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$, underlying

(a)   (b)
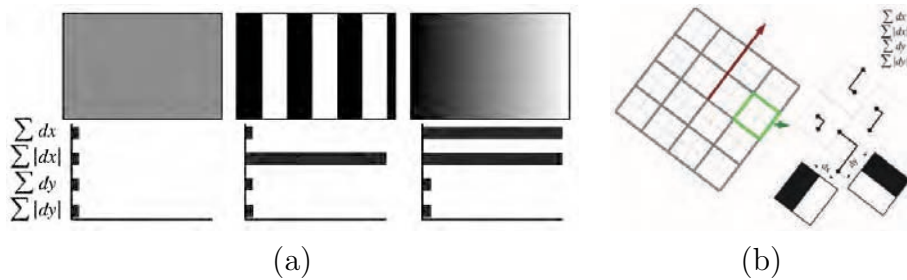
Figure 2.5: **SURF feature descriptor.** (a) The descriptor is constructed from absolute sums of wavelet responses along x and y over the $20s$ keypoint neighborhood split into 16 square sub-regions. The descriptor entries of a sub-region represent the nature of the underlying intensity pattern; (b) A square region centered in the keypoint is aligned with the keypoint orientation. "Horizontal" (x) and "vertical" (y) wavelet responses are also defined with respect to this orientation.

its intensity structure. By stacking up the descriptor vector for all the subregions, the result is a 64 dimension vector describing each key feature.

The wavelet responses are invariant to a bias in illumination (offset). Invariance to contrast (a scale factor) is achieved by normalizing the descriptor vector.

Feature Matching

Image features originating from the same pre-image region have similar photometric properties, reflected in a resemblance between feature descriptors. Early feature matching techniques used simple similarity measurements between features' neighborhood (correlation). Currently, correlation based methods are sparsely used in feature matching due to their lack of invariance to rotation and affine transformations. However, due its high precision in feature localization (subpixel correlation-based feature matching) [149], correlation is still used in some applications where rotation and affine transformations are absent or can be accounted for.

Modern feature matching techniques use similarity measurements between feature descriptors. As the descriptors themselves are invariant to illumination and camera viewpoint changes, these matching techniques are highly robust, increasing

the precision of image registration.

Correlation matching was initially used for area based feature matching [7, 39]. Here, the image features are extracted in image I_1 and the matches are obtained by applying correlation on a neighborhood of each pixel within a fixed-size search window in I_2 around the coordinates of each feature. The computational cost of this approach is very high since all possible candidates within each search window of each feature have to be analyzed. Later, this approach has been optimized by extracting image features (*e.g.* using Harris Corner extractor) and correlating windows centered on the extracted features.

Different feature matching methods have been developed based on the correlation measurement. They can be either a measurement of similarity – Normalized Cross Correlation (NCC) or a measurement of dissimilarity – Sum of Absolute Differences (SAD) and Sum of Squared Differences (SSD). Given a keypoint p_1 in image I_1 with coordinated (x_1, y_1) and a candidate match p_2 in image I_2 with coordinated (x_2, y_2) the NCC is defined on a rectangular neighborhood of these keypoints with size $2r + 1 \times 2r + 1$ as:

$$NCC(p_1, p_2) = \frac{\sum_{i=-r}^{i=r} \sum_{j=-r}^{j=r} (I_1(x_1 + i, y_1 + j) - \overline{I}_1)(I_2(x_2 + i, y_2 + j) - \overline{I}_2)}{r^2 \sqrt{\sigma^2(I_1)\sigma^2(I_2)}}$$

Here, $\overline{I}$ is the average and $\sigma^2(I)$ is the variance of image I:

$$\overline{I} = \frac{\sum_{i=-r}^{i=r} \sum_{j=-r}^{j=r} I(x + i, y + j)}{r^2}$$

$$\sigma^2(I) = \frac{\sum\limits_{i=-r}^{i=r} \sum\limits_{j=-r}^{j=r} (I(x+i, y+j) - \overline{I})^2}{r^2}$$

Similarly, SAD and SSD are defined as:

$$SAD(p_1, p_2) = \sum\limits_{i=-r}^{i=r} \sum\limits_{j=-r}^{j=r} |I_1(x_1+i, y_1+j) - I_2(x_2+i, y_2+j)|$$

$$SSD(p_1, p_2) = \sum\limits_{i=-r}^{i=r} \sum\limits_{j=-r}^{j=r} (I_1(x_1+i, y_1+j) - I_2(x_2+i, y_2+j))^2$$

Descriptor similarity.   In order to associate corresponding features in different images, a descriptor similarity measurement is needed. The most commonly used one is the Euclidean distance:

$$s(p_1, p_2) = \|f_1 - f_2\| \tag{2.1}$$

In order to find the corresponding features, the Euclidean distance is calculated between all the features in one image and all the features in the other image. The feature pairs corresponding to the minimum distances are most likely to correspond to the same pre-image region – Nearest Neighbor (NN). However, this is not always valid (not all features in one image have correspondence in the other) and this assumption can introduce outliers (erroneous feature pairs). One way to minimize this risk is to use a threshold on the maximum acceptable distance between feature descriptors, though optimal thresholds are dependent on the image data.

In [79], Lowe proposes a more general and robust approach. Here, he imposes that distance to the NN must be significantly lower (usually $\simeq 1.5\times$) than the distance to its Second Nearest Neighbor (SNN). In general, the SNN method performs well as correct matches need to be significantly closer than the closest incorrect

matches in order to achieve reliable matching. For false matches, there will likely be a number of other false matches within similar distances. The second nearest match provides an estimate of the density of false matches within this area of the feature space and at the same time identifies specific instances of feature ambiguity.

Figure 2.6 illustrates the matching process in the case of two distinct scenes: urban (Figure 2.6a and Figure 2.6c) and underwater (Figure 2.6b and Figure 2.6d).

2.1.4 Match Propagation

Match propagation represents a hybrid approach that combines feature-based with optical flow [74,147]. In match propagation, a sparse set of points are extracted from feature correspondences. Using these points as seeds, image registration is expanded using either optical flow or local matching. This approach combines the advantages of sparse and dense matching up to a certain degree.

2.2 Photo-mosaicing

Photo mosaicing (simply called "mosaicing" hereafter) is primarily a technique that allows widening the coverage of the scene by aligning (stitching) images taken by a panning or moving camera. Mosaicing has its origins in aerial photography, where images taken from planes or air balloons were manually aligned in order to obtain maps for military purposes. With the introduction of automated mosaicing techniques by means of image registration, photo mosaicing has extended his range of applications.

Mosaicing is widely used nowadays for underwater sea floor mapping to compensate for the narrow coverage of cameras due to limited visibility [38, 40, 41, 50,
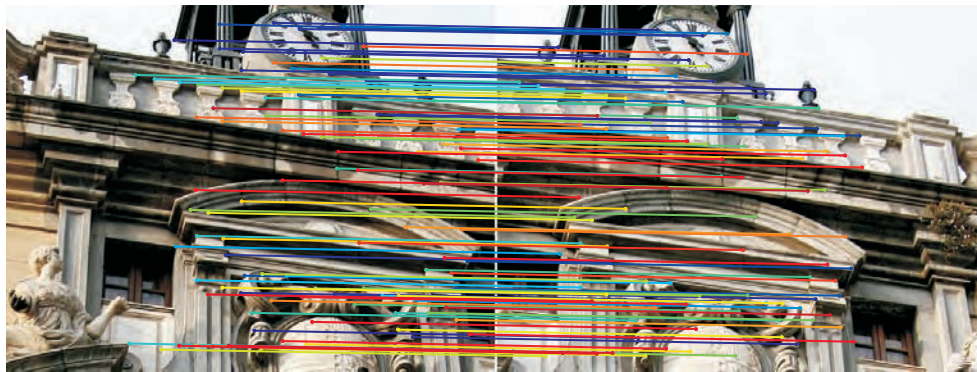
(a)
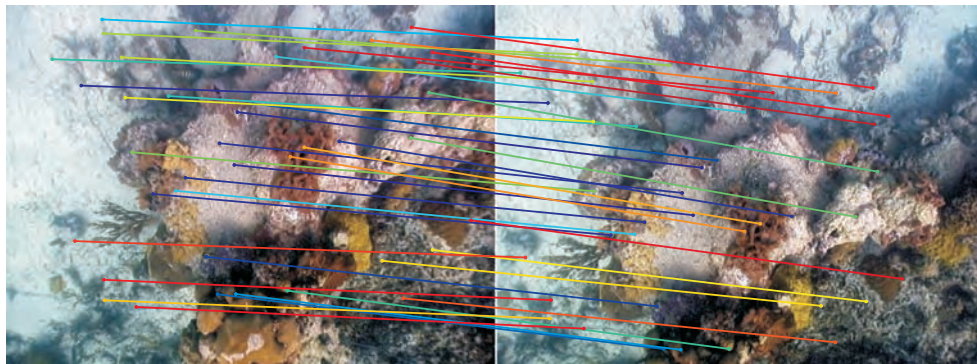
(b)
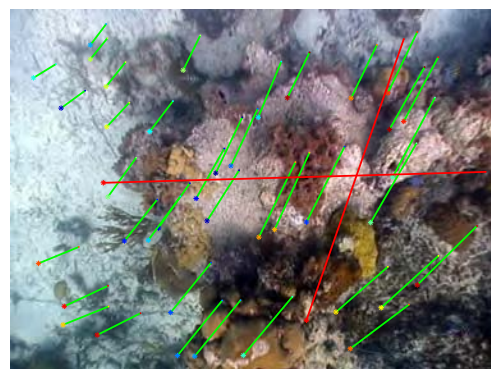
(c)   (d)

Figure 2.6: **Examples of feature matching.** (a) and (b) illustrate matching between two images for an urban and an underwater scene. (c) and (d) show the disparity of the features. Red lines denote mismatched features (outliers).

Figure 2.7: **Example of mosaicing.** Mosaic of a planar scene. Colored rectangles outline of the contributing images. Green corresponds to the first (reference) frame, yellow and blue to the second and third images respectively. The mosaic was generated using the projective homography model.

$65, 115, 132, 133]$. Furthermore, mosaicing techniques are successfully employed in applications such as navigation of underwater vehicles $[29, 37, 43, 49, 51]$, document analysis $[99]$, augmented reality $[138]$, scene stitching $[15, 139]$, etc.

Mosaics can be accurately employed in situations where the scene does not induce parallax $[56]$ – planar scenes or when the camera is rotated around its optical axis (see Figure 2.7 for an example of mosaic of a planar scene). In these cases the transformation induced on the images by the camera motion can be modeled as a planar transformation, called homography (H).

A homography is a planar projective transformation, represented by a 3×3 homogeneous matrix, relating the coordinate systems of two images I_1 and I_2 so that $p_1 = H \cdot p_2$:

$$\begin{pmatrix} \alpha \cdot x_1 \\ \alpha \cdot y_1 \\ \alpha \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

where $p_1 = (x_1 \ y_1 \ 1)^T$ and $x_2 = (x_2 \ y_2 \ 1)^T$ are 2D points in homogeneous coordinates; α is an arbitrary scale factor.

In the general case, homographies have 8 Degrees of Freedom (DoF). Depending on the application, the number of DoF can be reduced in order to limit the estimation errors. Table 2.1 provides a description of the common types of homographies.
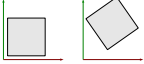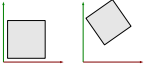
Transform	Figure	DoF	H
Translation		2	$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$
Euclidean		3	$\begin{pmatrix} cos(\phi) & -sin(\phi) & t_x \\ sin(\phi) & cos(\phi) & t_y \\ 0 & 0 & 1 \end{pmatrix}$
Similarity		4	$\begin{pmatrix} s \cdot cos(\phi) & -s \cdot sin(\phi) & t_x \\ s \cdot sin(\phi) & s \cdot cos(\phi) & t_y \\ 0 & 0 & 1 \end{pmatrix}$
Affine		6	$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix}$
Projective		8	$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}$

Table 2.1: **Homography motion models.** A 2-DoF homography allows only for translation between images. Euclidean transformations account for the translation and a rotation angle ϕ between images. The similarity model adds the scaling factor s. Affine motion model extends the similarity model by including the anisotropic scaling. Finally, the projective motion model describes any possible planar transformation between images induced by a 6-DoF camera motion.

The transformation H between images is obtained using image registration tech-

niques, most commonly using feature correspondences [104] (see Section 2.1.3). Depending on the homography model, a minimum of 1 to 4 correspondences are needed to compute H. In practice, tens to hundreds of correspondences are used in order to increase the precision of the homography in the presence of feature localization noise.

The mosaics are then rendered by establishing a global frame (coordinate system) and aligning the images with respect to this global frame using homographies. Generally, the mosaic coordinate system is chosen to coincide with the coordinates of the first image; in this case the transformation between an image j and the mosaic (absolute homography) is obtained by chaining the local homographies (relative homographies) of the previous images: $^1H_j = ^1 H_2 \cdot^2 H_3 \cdot ... \cdot^{j-2} H_{j-1} \cdot^{j-1} H_j$. From this, it becomes evident that small errors in the relative homographies build up to generate important inaccuracies in the estimation of the absolute homographies. This problem is common to vision systems, where the camera position is computed incrementally (see Section 1.3).

Generally, homographies are prone to estimation inaccuracies due to:

Feature localization errors – are induced by image noise, aliasing, changes in lighting and camera viewpoint.

Outliers – are caused by feature matching errors, usually due to repetitive patterns in the scene.

Moving objects – violate the rigid scene assumption.

Non-planar scenes – violate the planarity assumption.

The effect of feature localization errors, outliers and moving objects can be reduced by using modern feature detectors and robust estimation methods such as

Random Sample Consensus (RANSAC) [33] or Least Median of Squares (LMedS) [122]. However, the violation of planarity represents a limitation of 2D mosaicing techniques.

Obviously, most outdoor scenes (landscapes, urban, underwater, etc.) are hardly planar. If the camera is not sufficiently far from the scene[2], the parallax effect produces significant errors in the homography estimations (see Figure 2.8). In this case, 3D reconstruction techniques represent a far more accurate alternative to 2D mosaicing.

2.3 3D Reconstruction

3D reconstruction techniques are concerned with the recovery of the shape of scenes and their representation as 3D models. Using such techniques, a $3D$ model of the scene is obtained, represented as a collection of 3D elements such as points (vertices), lines, planes, surfaces, etc.

In order to recover the geometry of the scene, 3D objects are related to their projection on the image plane. Assuming the pinhole model[3], this relation is given by the projection matrix Π so that:

$$^i\begin{pmatrix} \alpha \cdot p_x \\ \alpha \cdot p_y \\ \alpha \end{pmatrix} = {}^i_W\Pi_{(3\times4)} \cdot {}^W\begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

---

[2]As a thumb rule, if the camera-to-scene distance is more than $\sim 10\times$ the scene depth variations, the parallax-induced errors can be neglected.

[3]Pinhole camera model is a simplified representation of the cameras, where some of the transformations that light suffers inside the camera optics are ignored. It is the most widely used projective representation due to its simplicity.

(a)

(b)

(c)

(d)

Figure 2.8: **Mosaicing under parallax.** (a) and (b) show two images representing a light pole with a building in background. The planarity assumption does not stand here due to the big depth differences between scene elements with respect to the camera. Trying to register the images using mosaicing techniques generates important misalignments. In (c) the building facade is correctly aligned, being the predominant plane, however the light pole and the trees create a ghosting effect (shown in color). Image (d) illustrates the full mosaic (20 images) of the facade. Again, we can observe that the parallax effect induces misalignments (highlighted in yellow).
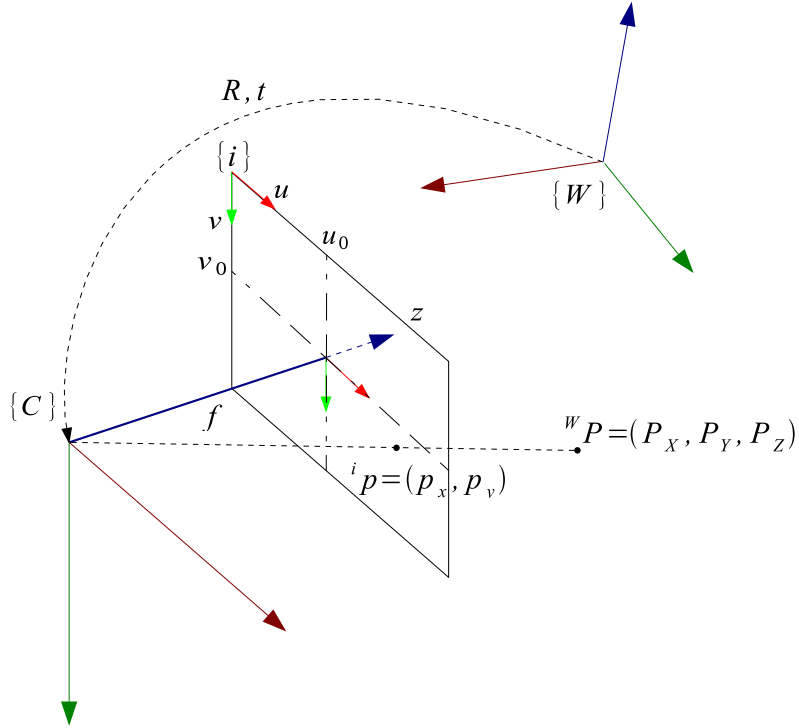
Figure 2.9: **Camera projection.** The 3D point ^{W}P is projected in the image plane onto point ^{i}p.

where $\mathbf{p} = (p_x, p_y)^T$ is the image projection of a 3D point $P = (P_x, P_y, P_z)^T$ (see Figure 2.9), α is an arbitrary scaling factor. The projection matrix Π is a function of the rotation $^{W}_{i}R$ and translation $^{W}_{i}t$ between the scene (world) and camera coordinate systems, and the intrinsic parameters (A) of the camera:

$$
^{i}_{W}\Pi_{(3\times4)} = A \cdot \begin{pmatrix} ^{W}_{i}R^{T}_{(3\times3)} & ^{W}_{i}t_{(3\times1)} \\ 0_{(1\times3)} & 1 \end{pmatrix} \tag{2.2}
$$

$$
A = \begin{pmatrix} k_u & k_c & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}
$$

(a) Pincushion Distortion

(b) Barrel Distortion
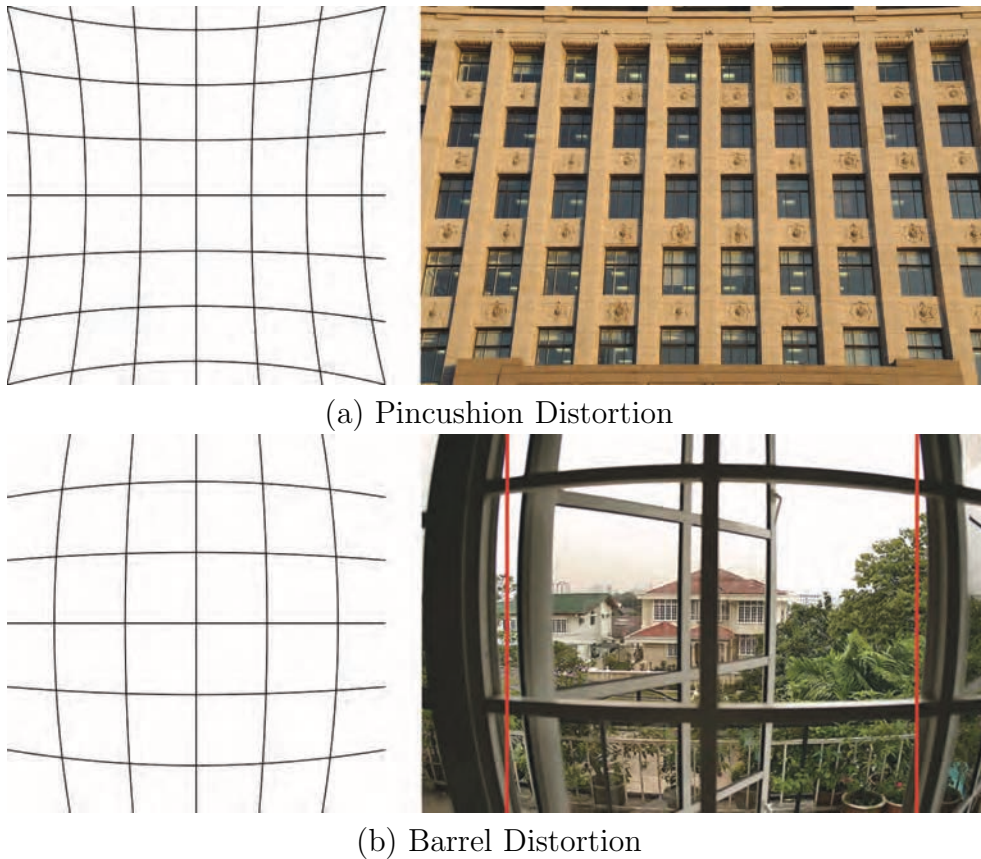
Figure 2.10: **Camera distortions.** Image (a) illustrates the pincushion distortion, typical for long focal lengths (tele-lenses), (b) shows the barrel distortion mostly found in wide-angle lenses (short focal distances).

where f is the focal length in millimeters and k_u, k_v are the relationships between pixels and world metric units (in $^{pix}/_{mm}$) along x and y axes of the image plane, respectively. The point where the camera focal axis intersects the image plane (principal point) is defined by (u_0, v_0). Finally k_c is the skew between image frame vectors (u, v) which, in the ideal case, is 0 (they are perpendicular), therefore, $k_c = \tan(\phi) \cdot k_v$, where ϕ is the skew angle between the image frame vectors.

In order to be able to accurately apply the pinhole camera model on real cameras, we have to account for the radial and tangential [14] distortions induced by the optical systems of these cameras (illustrated in Figure 2.10).

Considering an unitary focal length ($f = 1$), the projection of a 3D point $P = (P_x, P_y, P_z)$ is given by:

$$p_d = \begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} \frac{P_x}{P_z} \\ \frac{P_y}{P_z} \end{pmatrix}$$

by denoting $r^2 = x_d^2 + y_d^2$, considering the distortion model, we obtain the undistorted point p_u:

$$p_u = \begin{pmatrix} x_u \\ y_u \end{pmatrix} = (1 + k_{c_1} \cdot r^2 + k_{c_2} \cdot r^4 + k_{c_5} \cdot r^6) \cdot p_d + d_t$$

where d_t is the tangential distortion vector defined as follows:

$$d_t = \begin{pmatrix} 2 \cdot k_{c_3} \cdot x_d \cdot y_d + k_{c_4} \cdot (r^2 + 2 \cdot x_d^2) \\ k_{c_3} \cdot (r^2 + 2 \cdot y_d^2) + 2 \cdot k_{c_4} \cdot x_d \cdot y_d \end{pmatrix}$$

Parameters k_{c_1}, k_{c_2}, ..., k_{c_5} represent non-linear distortion coefficients. After undistorting, the point $(x_u, y_u, 1)^\mathsf{T}$ is projected into the image plane using the matrix of intrinsic camera parameters A:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = A \cdot \begin{pmatrix} x_u \\ y_u \\ 1 \end{pmatrix}$$

Camera intrinsic parameters $(\alpha_u, \alpha_v, \alpha_c, u_0, v_0)$ and the non-linear distortion co-efficients (k_{c_1}, k_{c_2}, ..., k_{c_5}) are obtained by camera calibration methods [32, 125].

So far, we have discussed the problem of estimating the projection of a 3D point in the camera plane given its 3D position. Nonetheless, we are interested in the reverse problem: given a projection of a 3D point (or another scene element) in the camera plane, how to recover the 3D position of the former. This problem cannot
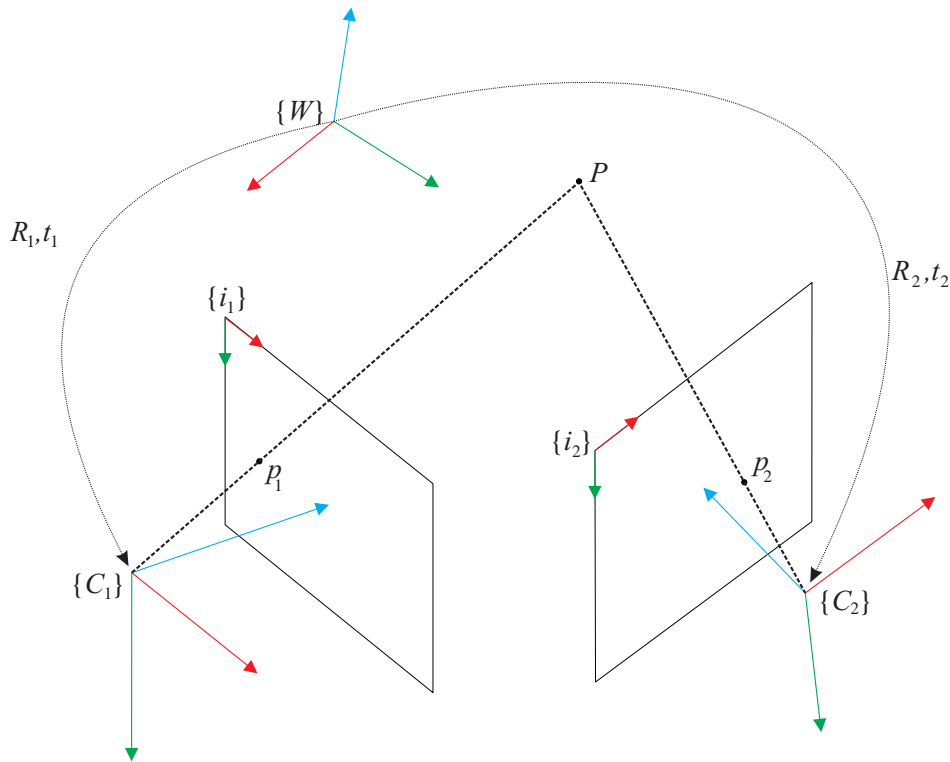
Figure 2.11: **Stereo triangulation.** Having the position of the projections of the 3D point P in two cameras $(p_1, p_2)$and the position of the cameras(R_1, t_1, R_2, t_2), the 3D position of P is defined by the intersection of the projection lines (C_1, p_1) and (C_2, p_2).

be solved from a single camera view. By analyzing Figure 2.9, we can see that any 3D point P' laying on the line (C, P) would yield the same projection p on the image plane. Consequently, having only the position of the 2D point p, there is an ambiguity in the position of P. This problem can be resolved given two or more camera views of P, as illustrated in Figure 2.11.

Hereafter, we briefly discuss some of the most widely used techniques for recovering the camera pose and the 3D geometry of a scene.

Stereo. Stereo vision algorithms use two or more cameras in a rigid setup. The relative pose of the cameras is obtained by calibration [57, 154]. In this case, the geometry of the scene can be obtained directly using epipolar geometry [56, 83].

At application level, online stereo vision techniques such as those presented in [30, 103, 111, 151] include robot navigation and visual servoing. Other proposals, more related to our work, focus on scene and object modeling [88, 144].

Stereo systems generally provide accurate results but require a more complex calibration and image acquisition process. The use of multiple cameras and the necessity of camera synchronization hardware greatly increases the acquisition costs. Moreover, stereo setups cannot be easily handled by humans and are difficult to mount on small size robots.

Structure from Motion. The SfM problem refers to recovering the structure of the scene using a single moving camera. The main advantage of Structure from Motion is actually the use of a single camera, which leads to a highly flexible and accessible image acquisition process.

From the algorithmic point of view, SfM is equivalent to stereo techniques except that the 3D camera motion for each time interval[4] has to be determined. As the camera motion and the scene structure are computed at the same time, the result of SfM is an up-to-scale representation of the scene[5] (see Figure 2.12). However, this scale ambiguity can be resolved if the size of any object in the scene is known.

Initial approaches of Structure from Motion used motion computation based on fundamental matrix (F) [9, 75] and trifocal tensor [35]. These approaches have a common drawback: position estimation based on motion integration leads to important drifts over relatively short distances. With the introduction of Bundle Adjustment (BA) techniques [76, 109, 136, 142], the effect of drifting can be partially reduced by globally minimizing the reprojection errors within the image sequence. In this context, some authors [16, 128, 145] have proposed batch SfM methods that

---

[4]Time elapsed between two consecutive frames captured by the camera.
[5]Assuming that the camera intrinsic parameters are known.
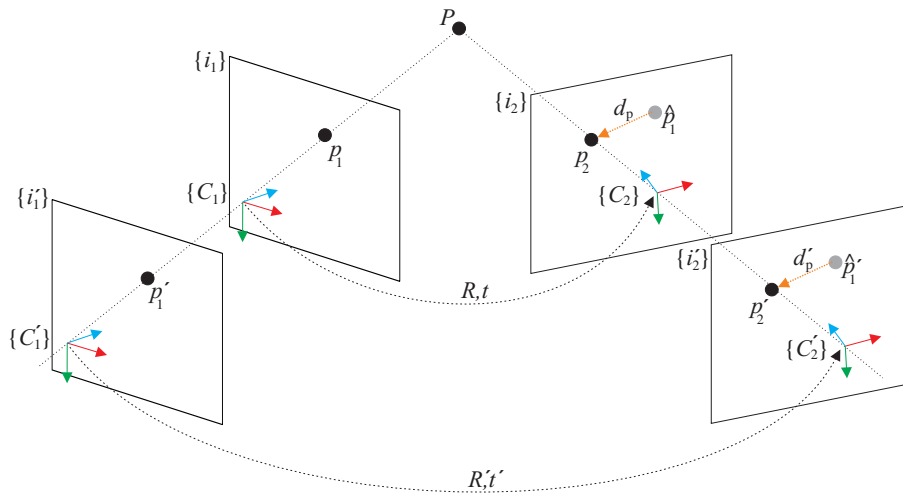
Figure 2.12: **SfM scale ambiguity.** The 3D position of point P from an arbitrary moving camera can be determined up to a scale factor. The disparity d_p of the point P determined by the camera motion R and t would be the same as d'_p when the motion R' and t' is greater with P further away.

use camera motion estimation followed by BA. However, in online applications, where accurate scene structure and camera poses have to be constantly available, repeatedly applying BA to correct for drifts is not feasible due to the high computational costs related to Bundle Adjustment. Additionally, motion-integration SfM methods suffer from another major drawback: instability at small camera motions. In this case, F is ill-conditioned [56], resulting in a poor estimation of the camera motion.

A more accurate alternative to motion integration is the direct recovery of camera pose. This can be achieved by associating 3D features with image features and estimating the camera pose using methods such as Direct Linear Transformation (DLT) [56]. In [135], the authors propose a batch SfM method based on DLT for reconstructing well-known world sites from Internet photo collections.

Oriented towards underwater imaging, Pizarro *et al.* [113, 114] propose a SfM framework that deals with large sequences by independently processing local

submaps. Within the submaps, the camera pose is recovered directly by using resection methods and the submaps are registered using global alignment techniques. While accurate, this approach has somewhat limited applications as it uses navigation priors for submap generation.

Factorization. Factorization methods use a special formulation to deal with scene structure and camera motion, decomposing image measurements (*i.e.* image disparities) into a product of two separate factors:

$$image\_disparities \Longleftrightarrow motion \times shape$$

The first solution to the factorization problem, introduced by Tomasi and Kanade, used rank constraints under orthographic camera projection [140]. This work was later extended to deal with more general camera models [53, 58, 137]. Factorization methods are mainly aimed at object and small scale reconstructions. Even though latest developments in factorization allow partially dealing with missing data [17, 87], these methods cannot deal with high degree of missing data, rendering them impractical for scene modeling.

Dense reconstruction. These methods use pixel disparities, rather than feature correspondences to recover the scene geometry. The result is a 2.5D model[6] of the scene, where the optical flow is used to estimate the depth of the scene points corresponding to each pixel in the image [67, 97]. Generally, dense reconstruction approaches use iterative methods based on Longuet-Higgins differential image motion model, being highly expensive in terms of computation.

---

[6] 2.5D models do not represent the full geometry of the scene. Alternatively, the scene is represented by a regular grid of points defined by their depth.

2.4 Conclusions

A wide range of applications, from remote scientific studies to augmented reality and virtual tourism benefit from automated visual mapping. Classical approaches involving 2D mosaicing are limited to quasi planar scenes. In reality, most environments are far from being planar and the necessity to map such environments led to an increased interest in 3D scene modeling. Despite this, most 3D techniques are application specific and inherently offline. Moreover, these approaches are rather limited, being able to handle only small scale / reduced complexity scenes.

Chapter 3

Direct Structure from Motion

This chapter is concerned with robust 3D scene modeling using a novel Structure from Motion algorithm – Direct Pose Registration Structure from Motion (DPR-SfM). The aim is to obtain a high precision texture model of a generic scene acquired using any off the shelf camera undergoing an arbitrary trajectory. The reconstruction algorithm does not require any camera position / attitude information, endowing DPR-SfM with flexibility to be readily used for any type of 3D scene modeling application, both underwater and terrestrial.

For this, we have designed the DPR-SfM algorithm to cope with the most common challenges (see Section 1.3):

- Object occlusions and perspective distortions.

- Invalid image frames due to camera obstructions, motion blur, etc.

- Moving objects.

- Image noise, low contrast and illumination changes (especially in the underwater environment).

DPR-SfM computes directly the pose of the camera without the necessity to recover the inter-frame motion. The structure of the scene is formed by sets of 3D vertices characterized by affine invariant local image descriptors. In this way, by associating image patches extracted from camera views with the 3D vertices, we can recover the camera pose with respect to the scene model. In DPR-SfM, the camera pose is obtained using a novel dual approach, allowing accurate camera pose estimations even in the presence of planar scenes, where most 3D reconstruction algorithms would fail.

Subsequently, the obtained camera poses are used to update the scene model as new features are tracked. Both camera pose estimation and scene model update steps use robust methods thus reducing the impact of poor camera pose/vertex estimations.

DPR-SfM algorithm works in two stages, as shown in Figure 3.1. First, it uses motion estimation techniques in order to obtain an initial model corresponding to a small subregion of the scene. In the second stage, using the initial model as a "seed", the subsequent camera poses are computed by registering 2D features with 3D vertices in the scene model. For each newly acquired image, once the camera pose is recovered, the scene model is updated by adding vertices corresponding to newly tracked features. In this way, as the camera moves, the model is extended to represent new regions of the scene.

As the data is being processed sequentially, camera pose and scene model estimations are constantly available, enabling the use of DPR-SfM for online applications such as robot navigation and mapping, *in situ* scientific studies, etc.

The remainder of this chapter details the flow of the DPR-SfM algorithm, followed by a discussion on various results that we have obtained by applying the proposed algorithm on outdoor and underwater image sequences. For the ease of
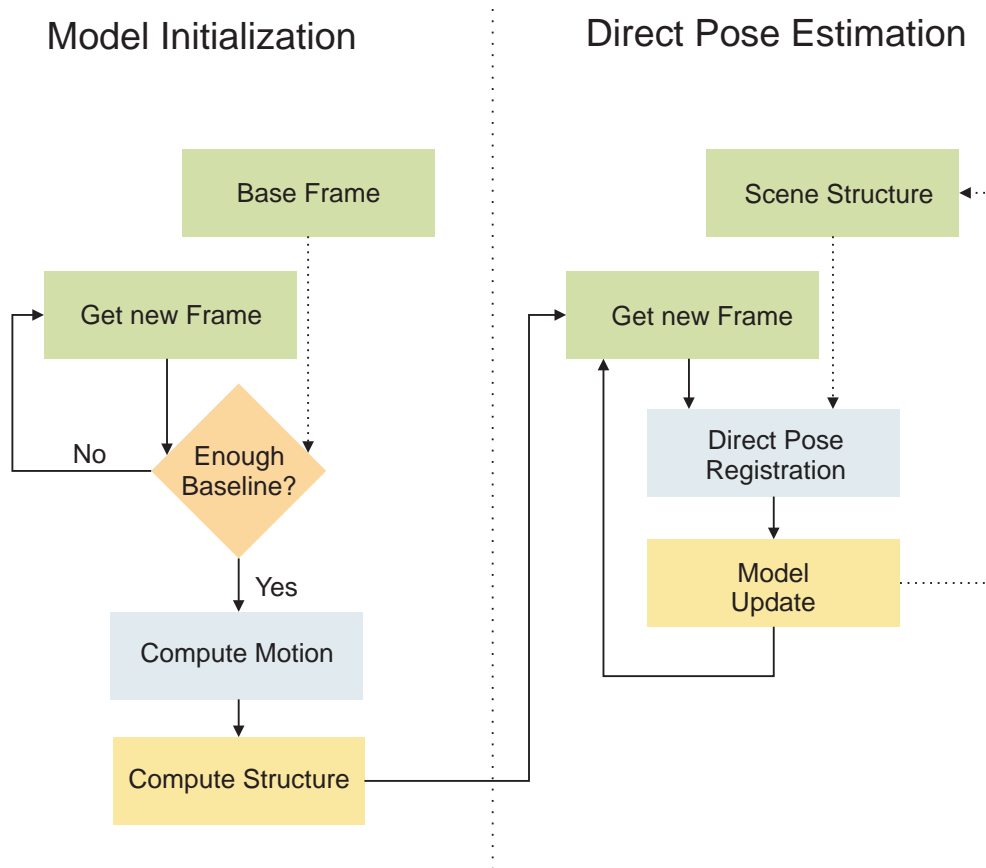
Figure 3.1: **Flowchart of the DPR-SfM algorithm.** The *model initialization stage* estimates the baseline between the base frame and a newly acquired frame. If the baseline is wide enough, the motion between the base frame and the acquired frame is recovered. Using the motion, the scene structure is estimated and the algorithm passes to the direct pose registration stage, otherwise the process is restarted using the next acquired frame. In the *direct pose registration stage*, the camera poses are obtained by extracting correspondences between the acquired images and the model. After each new camera pose estimation, the algorithm updates the model with new vertices corresponding to features tracked in the current image. In this way, the scene model grows as the camera surveys new regions of the scene.

the explanation, we illustrate the description of the DPR-SfM algorithm using a simple dataset[1] provided by the Visual Geometry Group of University of Oxford. Figure 3.2 depicts the input set of images of a house model.

---

[1]http://www.robots.ox.ac.uk/~vgg/data/dunster/images.tar.gz.
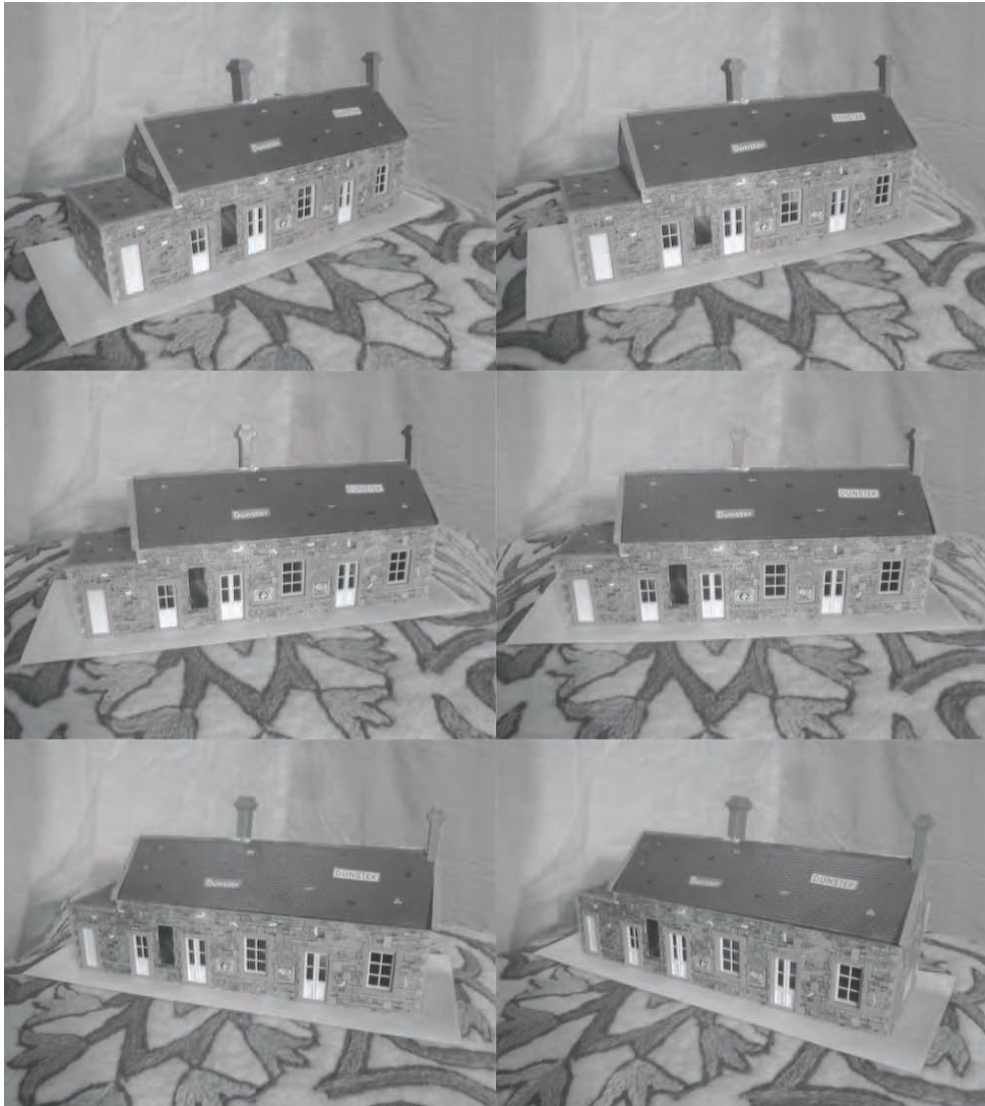
Figure 3.2: **DPR-SfM – House dataset.** The input sequence of 6 images captured by a camera undergoing a rotation around a house model.

3.1   Image Features

Feature tracking is the building block of any sparse 3D reconstruction algorithm. Tracking image features corresponding to a scene region (*i.e.* points, lines, patches, etc.), allows the 3D position of the scene features to be estimated.

Robust feature tracking is crucial to the accurate estimation of both the camera

poses and the structure of the scene. Maximizing the number of frames where a given scene feature is tracked improves the precision of its 3D position estimation and increases the number of inter-frame constraints, allowing a higher precision in camera pose estimation.

In order to ensure robust feature tracking in presence of geometric distortions and illumination changes, we have tested various state of the art point and blob feature extractors (see Section 2.1.3): Harris Affine, Hessian Affine, SIFT, SURF and MSER. As expected, point feature extractors generate more dense sets of features than blob feature extractors, providing a better coverage of the scene but having less discriminative power, increasing the chances of mismatching. In contrast, blob extractors produce more sparse but more stable sets of features with higher discriminative power.

In terms of feature descriptors, Harris, Hessian and MSER can be described using both SIFT and SURF, while SIFT and SURF use their own descriptors only.

3.2   Model Initialization

This stage generates a subregion ("seed") of the 3D model corresponding to the first few frames of the image sequence. This initial subregion is required by the second stage that subsequently extends it to the full 3D scene model.

The model is initialized by first fixing the first frame of the sequence as the base frame I_b. The camera pose corresponding to I_b will serve as the global reference frame (world frame) for the entire model. During model initialization, the camera motion between the reference and some image I_i is computed. I_i is chosen so that the baseline between I_b and I_i is sufficient to ensure a robust motion estimation. The baseline between images is approximated by translation induced by the homography

$^b{-}0.8mmH_i$ on the image centers, where ^bH_i is a projective homography obtained from feature correspondences between images I_b and I_i (see Section 2.2).

Generally, SfM algorithms use fundamental matrix for camera motion estimation. However, when the scene is planar or the parallax effect is small (*i.e.* small scene depth variations with respect to scene-to-camera distance), the fundamental matrix can be ill-conditioned [56]. In this case, a more robust solution is to use homography-based motion computation. On the other hand, when scene geometry induces significant parallax, homographies cannot correctly model the camera motion. In order to cover both cases, we use a dual approach for motion computation:

Fundamental matrix motion computation. Using the feature correspondences between images I_b and I_i (see Figure 3.3), we estimate the fundamental matrix F_{bi} using RANSAC-based Least Squares (LS) methods[2] [3], with the cost function given by the Sampson distance [126] (see Figure 3.3c,d):

$$E_{sampson}^k = \frac{[(p_b^k)^T F_{bi} p_i^k]^2}{(F_{bi} p_i^k)_1^2 + (F_{bi} p_i^k)_2^2 + (F_{bi}^T x_l^k)_1^2 + (F_{bi}^T x_l^k)_2^2} \qquad (3.1)$$

where $(Fp)_j^2$ represents the square of the j-th entry of vector Fp.

The camera rotation R_{bi}^F and translation t_{bi}^F are obtained by Singular Value Decomposition (SVD) of F_{bi} using [62, 77]:

$$F_{bi} = (A^{-1})^T \, \widehat{T}_{bi}^F \, R_{bi}^F \, A^{-1} \qquad (3.2)$$

where A is the known camera intrinsic matrix, R is the rotation matrix of the camera and $\widehat{T}$ is the translation skew-symmetric matrix ($\widehat{T}_{[x]} = t \times x$ for any vector x with t representing the camera translation). The approach yields

---

[2]After testing various fundamental matrix estimation methods, RANSAC-based LS method has been adopted as it proved to provide the most robust results in the case of small base lines.

4 possible solutions (2 translations and 2 rotations). The correct solution is obtained by applying cheirality constraints (*i.e.* reconstructed points must be in front of the camera) [120].

Homography motion computation. From the correspondences of I_b and I_i we compute the homography ${}^bH_i$ using RANSAC with the cost function given by:

$$E^H = p_b^k - {}^bH_i \, p_i^k$$

where p_b^k and p_i^k represent the kth feature correspondence in images I_b and I_i respectively.

By normalizing the homography between I_b and I_i:

$$\widehat{{}^bH_i} = -A^{-1} \, {}^bH_i \, A$$

we obtain the camera camera rotation R_{bi}^H and translation t_{bi}^H using SVD [31]:

$$\widehat{{}^bH_i} = R_{bi}^H - t_{bi}^H \, \eta^T$$

where η is the normal of the scene plane. This type of decomposition raises two solutions. The correct one corresponds to the plane normal pointing towards the camera.

Between the two solutions (R_{bi}^F, t_{bi}^F) and (R_{bi}^H, t_{bi}^H), we choose the most accurate one. This is done by estimating the 3D position of the image features with respect to each solution using LS Intersection. Then, the accuracy of the camera motion is given by the back-projection error:
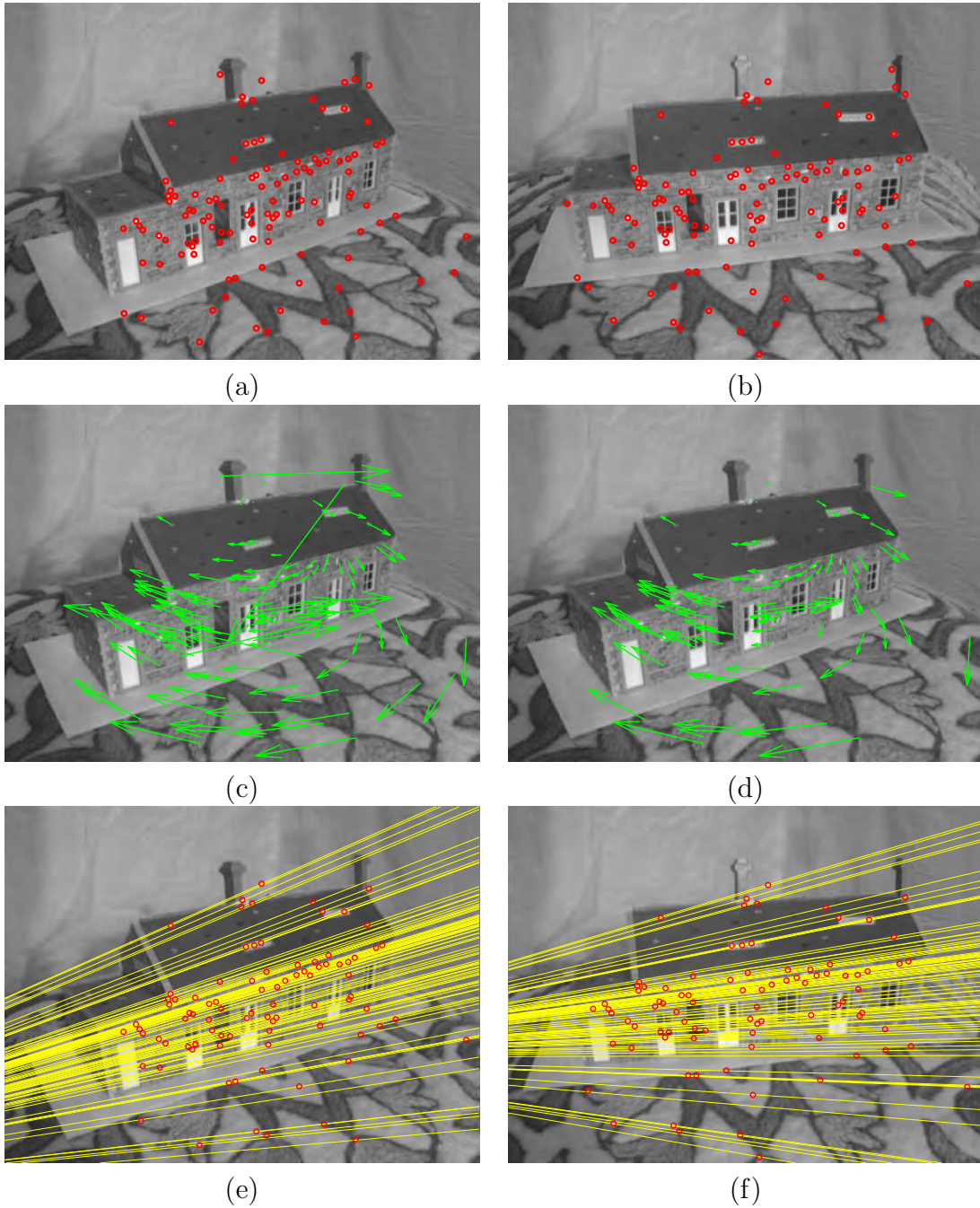
Figure 3.3: **DPR-SfM – Camera motion.** When there is enough camera motion between the base frame (left column) and the current frame I_i (right column), the pose is computed. (a) and (b) show the extracted image features. (c) show the initial feature disparity after matching, (d) shows the feature disparity after outlier rejection, in this case using F. (e) and (f) illustrate the epipolar lines for I_b and I_i, respectively.

$$E_{bi} = \sum_{k=1}^{N} (\|p_b^k - \Pi_b P^k\| + \|p_i^k - \Pi_i P^k\|) \tag{3.3}$$

where, p_b^k and p_i^k are the corresponding image features in images I_b and I_i respectively; P^k is the estimated 3D position of kth feature.

The solution corresponding to the smallest retrojection error E_{bi} is chosen and the corresponding set of 3D points is used to initialize the scene model.

In order to complete the set of camera poses, we recover the pose of the cameras corresponding to the intermediate frames between I_b and I_i by directly registering the camera views with the 3D model (Section 3.4). Figure 3.4 illustrates the initial model for the House dataset, corresponding to the first three frames.

3.3   Scene Model

The scene model was designed to contain geometric along with photometric information. The geometry of the scene is described in terms of 3D vertices, defined by their position $[X \ Y \ Z]^T$ with respect to a common world frame. Photometrically, the vertices are characterized by descriptors obtained from their corresponding image feature descriptors.

The image descriptor vectors can be seen as noisy measurements of the image gradient within a feature patch. As the features are tracked, multiple measurements of the same patch are obtained. Hence, we improve feature tracking by modifying the similarity measurement in eq. (2.1) to include multiple observations:

$$s(\mathbf{f}^k, f_i^k) = \|\frac{\sum f^k}{n} - f_i^k\| \tag{3.4}$$
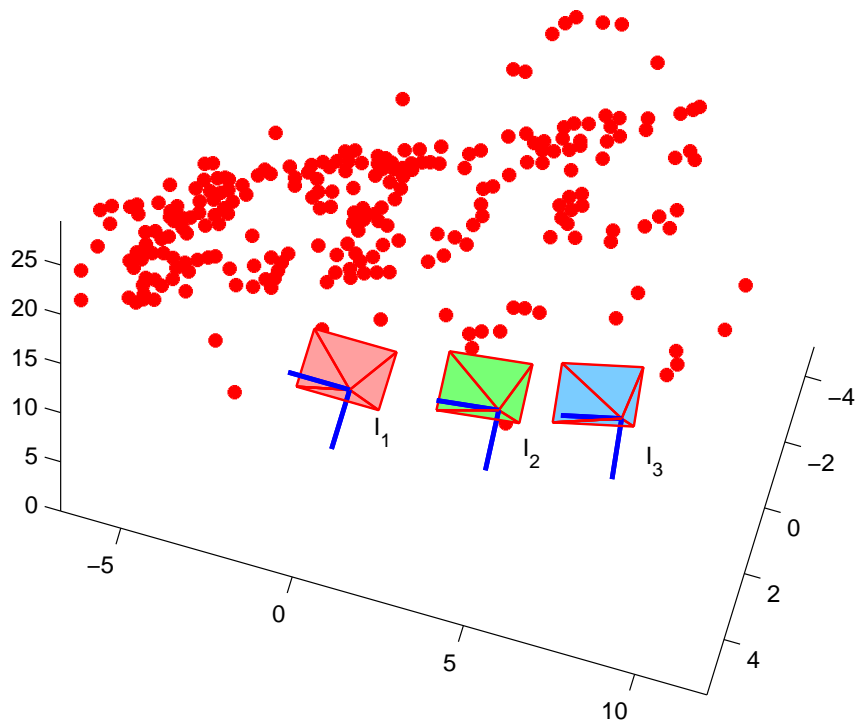
Figure 3.4: **DPR-SfM – Initial model.** Initial 3D scene model (red dots) and camera poses. The model initialization was done using frame 1 (red) and 3 (blue). Camera pose for frame 2 (green) was obtained by direct registration.

where V^k represents the descriptor vector of vertex[3] k, and n represents the total number of images where the vertex was tracked. Using such a descriptor representation allows for more stable vertex tracking in presence of image noise, illumination changes and projective distortions.

When associating vertices with image features using eq. (3.4), we impose distance

---

[3]Here, we use the term *vertex* to express a set of image features corresponding to the same scene point. The actual 3D position of the vertex does not need to be calculated at this point.

thresholds for $s(V^k, v_i^k)$ to reduce the number of outliers. The threshold values were established empirically. As all the feature descriptors are normalized, the established thresholds proved to provide optimum results (for both SIFT and SURF descriptors) in all the test sequences.

In practice, using a direct approach for feature association in eq. (3.4) involves a high computational load. Depending on the resolution and the feature extractor type, an image can yield thousands of features that have to be associated with tens of thousands of features from each feature group[4] in the scene model. We highly reduce this computational load by using a k-dimensional tree (kd-tree) approach. Using kd-trees, we hierarchically decompose the scene model feature space into a relatively small number of subregions so that no region contains too many features [4] (see Figure 3.5). This provides a fast way to access any scene model feature. In order to associate an image feature, we traverse down the hierarchy until we find the subregion containing the match and then scan through the few features within the subregion to identify the correct match. In the implementation that we used [96], we obtained a decrease in the computational time with respect to classical NN of about 5 times.

3.4   Direct Camera Registration

This section deals with the direct recovery of the camera pose with respect to the scene model, without the need of any *a priori* information on camera motion or pose. This way, the robustness of the DPR-SfM algorithm is increased, allowing it to naturally deal with camera occlusions, loop closures and position estimation errors.

---

[4]DPR-SfM supports simultaneous use of different feature types. In the scene model, the features are grouped by extractor/descriptor.
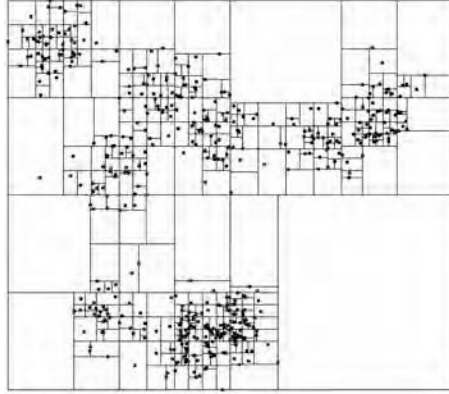
Figure 3.5: *Kd*-**tree partitioning.** The *k*-dimensional feature space is hierarchically partitioned in subregions containing a small amount of features.

In Section 3.3 we explain how to associate image and scene model features. From this, we obtain 3D-to-image correspondences with the aim of recovering camera pose (R_i, t_i) with respect to the world frame (see Figure 3.6). The camera pose is obtained using RANSAC with the cost function:

$$E_i = \sum_{k=1}^{N} \|p_i^k - \Pi_i P^k\| \tag{3.5}$$

In order to robustly cope with different types of scenes, we propose a novel dual approach for camera pose recovery (similar to the one described in the Section 3.2): *(i)* if the scene region seen in the current image has enough parallax, we use projective matrix to recover the camera pose; *(ii)* if the scene region is planar or close to being planar, the projection matrix is ill-conditioned [56], in which case we use a homography approach. In order to determine the planarity of the scene, for each RANSAC sample, we fit a plane L to the 3D vertices using a LS method. If the distance between the plane L and all the other 3D vertices (from the 3D-to-image correspondences) is small enough, we consider the scene region as being planar.
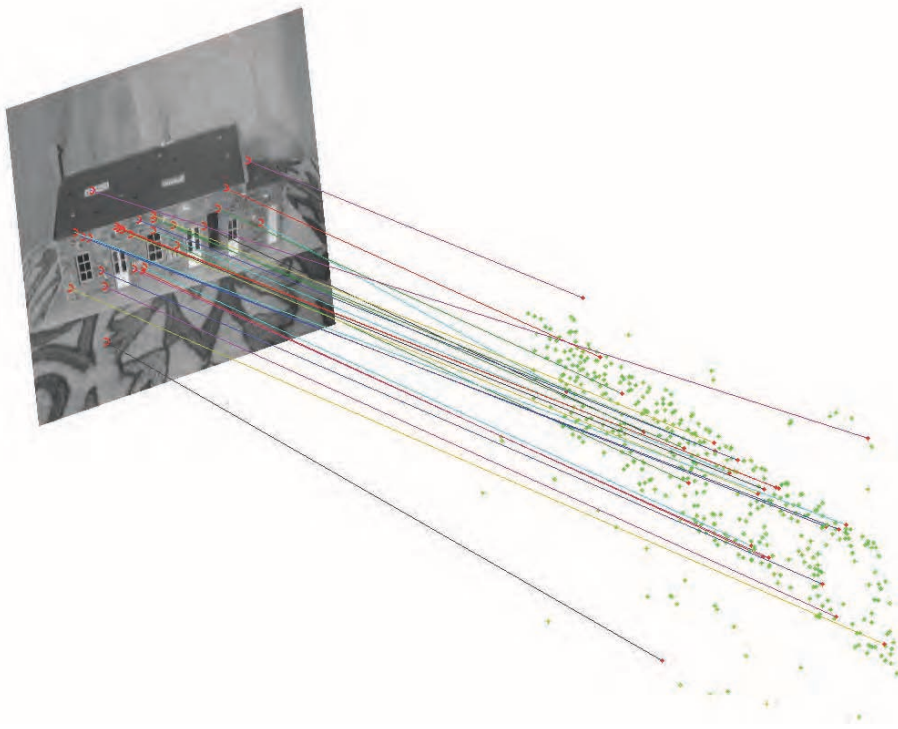
Figure 3.6: **DPR-SfM – Direct pose registration.** Example of pose registration of frame 4: the image features are associated with the scene model. The camera pose is estimated using the projection matrix.

The method is summarized in Table 3.1. The camera pose estimation methods are detailed hereafter:

Projection matrix-based. Provided the set of 3D-to-image correspondences, we obtain the projection matrix Π_k using DLT. From equation (2.2), we obtain the camera pose (R_k, t_k).

Homography-based. We compute the planar transformation ^iH_L, so that:

$$p_i^k = {}^iH_L \cdot p_L^k$$

where p_L^k is the projection of P_k onto plane L. Applying SVD on ^iH_L, we obtain the relative transformation $(^iR_L, {}^kt_L)$ between the plane L and the

Table 3.1: **Camera pose recovery process.**

1. While not enough RANSAC samples.
2. Choose randomly a set of 3D-to-image correspondences.
3. Fit a plane L to the 3D vertices from the set.
4. Check if the other vertices (corresponding to I_k) lay close to plane L.
5. If yes, compute R and t based on the homography using the set of correspondences.
6. If no, compute R and t based on the projection matrix using the set of correspondences.
7. Go to 1.

camera. Thus, the pose of the camera is obtained from:

$$t_i = t_L \cdot {}^iR_L + {}^it_L$$

$$R_i = {}^iR_L \cdot R_L$$

with t_L and R_L representing the pose of plane L in the world coordinate system.

Once a (R_k, t_k) have been obtained using the RANSAC dual method, the camera pose is further adjusted using a LS method that minimizes the back-projection error shown in eq. (3.5).

3.5 Model Update

As the camera moves, the DPR-SfM algorithm updates the scene model as new features are extracted and tracked, generating new 3D vertices. This section discusses the model updating process along with the outlier management.

As new images are fed to the DPR-SfM algorithm and the image features are associated with scene model features (see Section 3.3), three scenarios arise:

Image features matched with model features with known 3D position.

These feature associations are used to recover the camera pose, as explained in Subsection 3.4. The outliers are detected by reprojecting the 3D vertices into the image (eq. (3.5)). Vertices with a reprojection error higher than a pre-established threshold are eliminated. Inliers are added to the model to create new constraints. Every time an additional image feature is associated with a particular 3D vertex, the position of the vertex is refined, taking advantage of this new constraint. The refinement is done by minimizing the sum of the reprojection errors E_k in all the images where the vertex was tracked:

$$E_k = \sum_{i=1}^{M} \|p_i^k - \Pi_i P^k\| \tag{3.6}$$

Image features matched with model features with no 3D position.

Adding new image features to already existing model features provides additional information that ultimately leads to the recovery of 3D vertex position. In this case, the back-projection approach cannot be used for outlier rejection as the 3D position of the vertex is unknown at the time. Alternatively, we use a fundamental matrix based approach. For each image feature p_i^k we choose a feature p_i^l from its associated feature track so that their corresponding camera poses $(R_k,\ t_k)$ and $(R_l,\ t_l)$ have the widest possible

baseline (the wider the baseline the more discriminative the process). From the relative transformation between the two cameras (R_{kl}, t_{kl}) we compute the fundamental matrix F, as shown in equation (3.2). This allows us to use the Sampson distance shown in eq. (3.1).

If the image feature p_i^k yields a distance $E_{sampson}$ larger than a pre-established threshold, it is regarded as an outlier and the feature association is eliminated, otherwise it is added to the model. When enough views of a feature are available, the position of the corresponding vertex is calculated using a multi-view factorization approach [83]. The vertex position is then refined using a LS method (see eq. (3.6)).

Unmatched image features. If the image features could not be consistently associated to any model features, they are used to generate new feature entries in the model.

Since not all model features are tracked reliably enough to produce accurate 3D vertices, the model is constantly checked and features that do not provide a consistent tracking are eliminated in order to minimize the unnecessary clutter of the model.

Figure 3.7 illustrates the final 3D model of the House sequence along with the recovered camera poses.

3.6   Ortho-mosaicing and 3D Representation

A great deal of underwater studies require the assessment of 2D visual maps (see Section 1.2). When the regions of interest contain significant 3D relief, classical mosaicing techniques prove inaccurate due to the parallax effect. We propose a
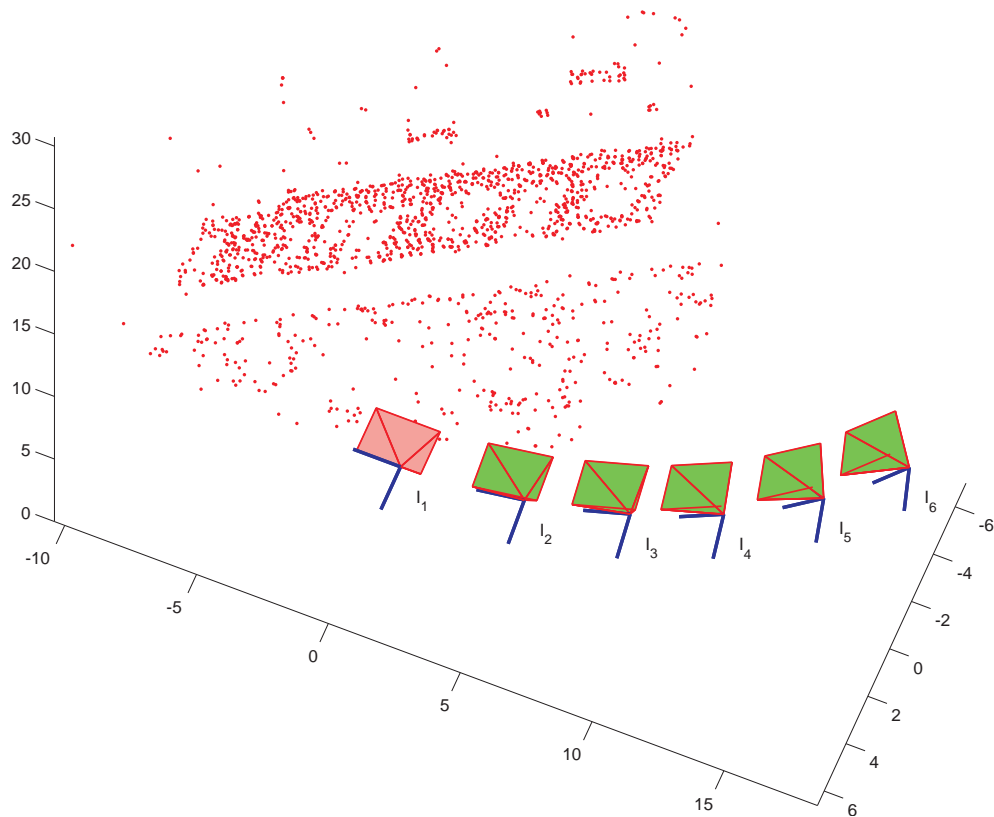
Figure 3.7: **DPR-SfM – Final model.** 3D model of the House sequence containing $\simeq 2,000$ vertices (red dots) along with the camera poses. The first camera pose (shown in red) defines the global coordinate system of the model.

solution to this shortcoming, where the 3D scene model is ortho-projected into a plane. The result is a virtual "high-altitude" view of the scene called *ortho-mosaic*. In other words, an ortho-mosaic is the equivalent to a 2D mosaic acquired from a camera located far from the scene.

The ortho-mosaic is obtained by first creating a continuous model of the scene. The continuous model is defined by triangular patches with the corners defined by the 3D vertices [6]. Within the patches, we can obtain the 3D position of any point

using linear or cubic interpolation[5].

An ortho-projection plane O is then chosen to have the same tilt as the average tilt of continuous model. This maximizes the projection area, providing the highest level of mosaic detail. Then, all the patches are mapped onto the destination plane along projection rays perpendicular to plane O (see Figure 3.8).

The plane O is digitized based on a predefined resolution; each point p_O^k on the grid corresponds to a pixel in the ortho-mosaic. In order to render the mosaic, we define the following transformation relating each point p_O^k to a corresponding point p_i^k from the original images:

$$p_i^k = \Pi_i T_n p_O^k \tag{3.7}$$

where T_n is the ortho-projection transformation of the patch $[P_1 \ P_2 \ P_3]$ and Π_i is the camera projection matrix corresponding to frame I_i, as shown in Figure 3.8a.

Figures 3.9a and 3.10 illustrate the results of the ortho-mosaicing process for the the House sequence and an underwater scene respectively.

For the cases where 3D information is required, the ortho-mosaic is used as texture for rendering the 3D surface. The result is a complete model that includes both geometrical and photometrical information of the scene. In Figure 3.11 we show two views of the 3D model of the underwater scene. Here, the surface was obtained by using cubic interpolation. In the case of the House scene, illustrated in Figure 3.9b, linear interpolation is more suitable.

---

[5]For natural and unstructured scenes, where the shapes are usually smooth, cubic interpolation provides the best results.

(a)

(b)
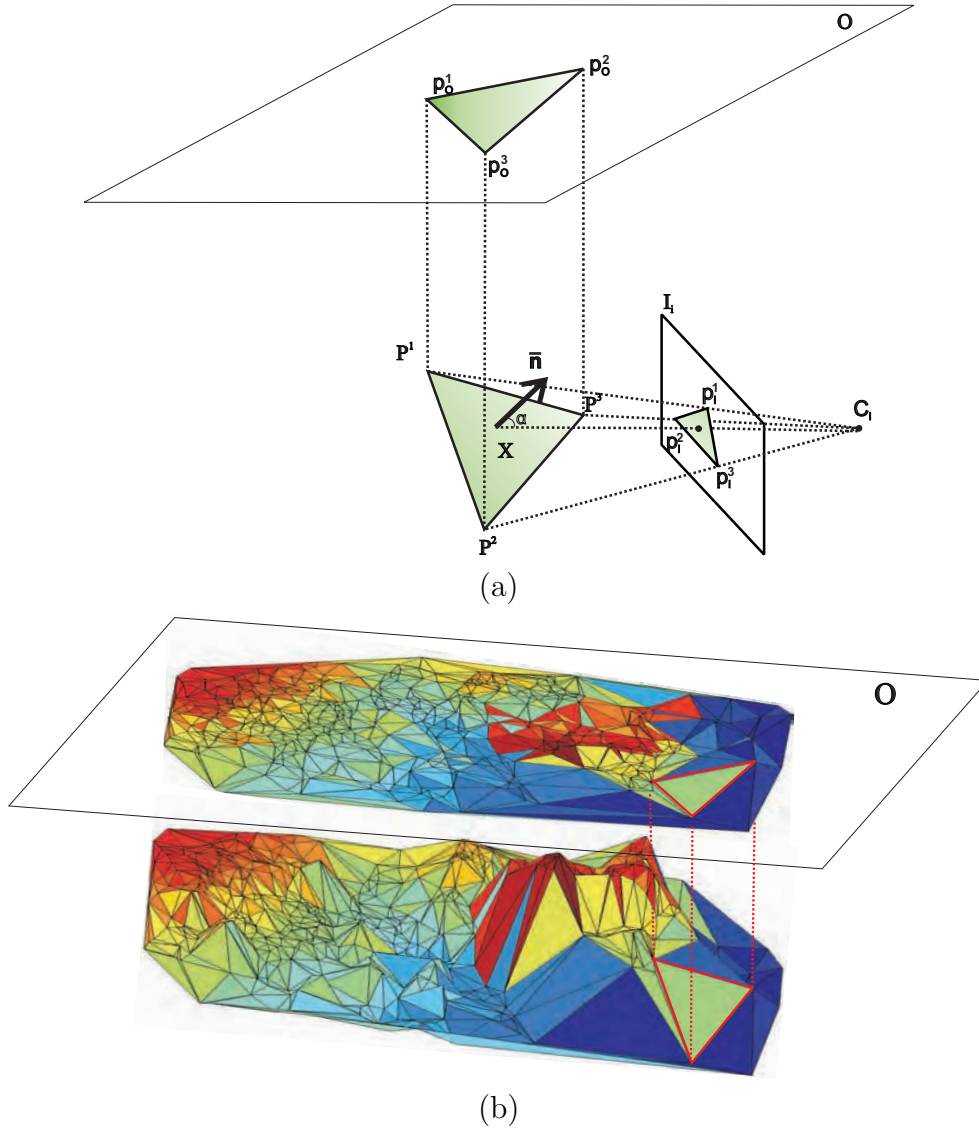
Figure 3.8: **Principles of ortho-mosaicing.** In figure (a) The model patch $[P_1\ P_2\ P_3]$ is ortho-projected onto the plane O. The corresponding ortho-mosaic patch $[p_O^1\ p_O^2\ p_O^3]$ is rendered using eq. (3.7) from image I_i, chosen so that the angle α between the patch normal and the camera principal axis is minimum. In (b), for clarity purposes, we show the ortho-projection of a seafloor model containing a coral-reef formation (Bahamas dataset). This model will be discussed in detail in Section 3.7.

3.7 Experimental Results

In this section, we discuss the performance of the DPR-SfM algorithm. The evaluation focused on two main aspects: *(i)* the accuracy of both scene model and camera

(a)

(b)

Figure 3.9: **Model of the house scene.** (a) shows the ortho-mosaic of the house. In this case, there is no gain in using the ortho-mosaic since all the camera views cover the entire scene. (b) is a view of the textured model; the 3D surface was generated using linear interpolation, which is more suited for structured scenes, containing planes and straight edges.
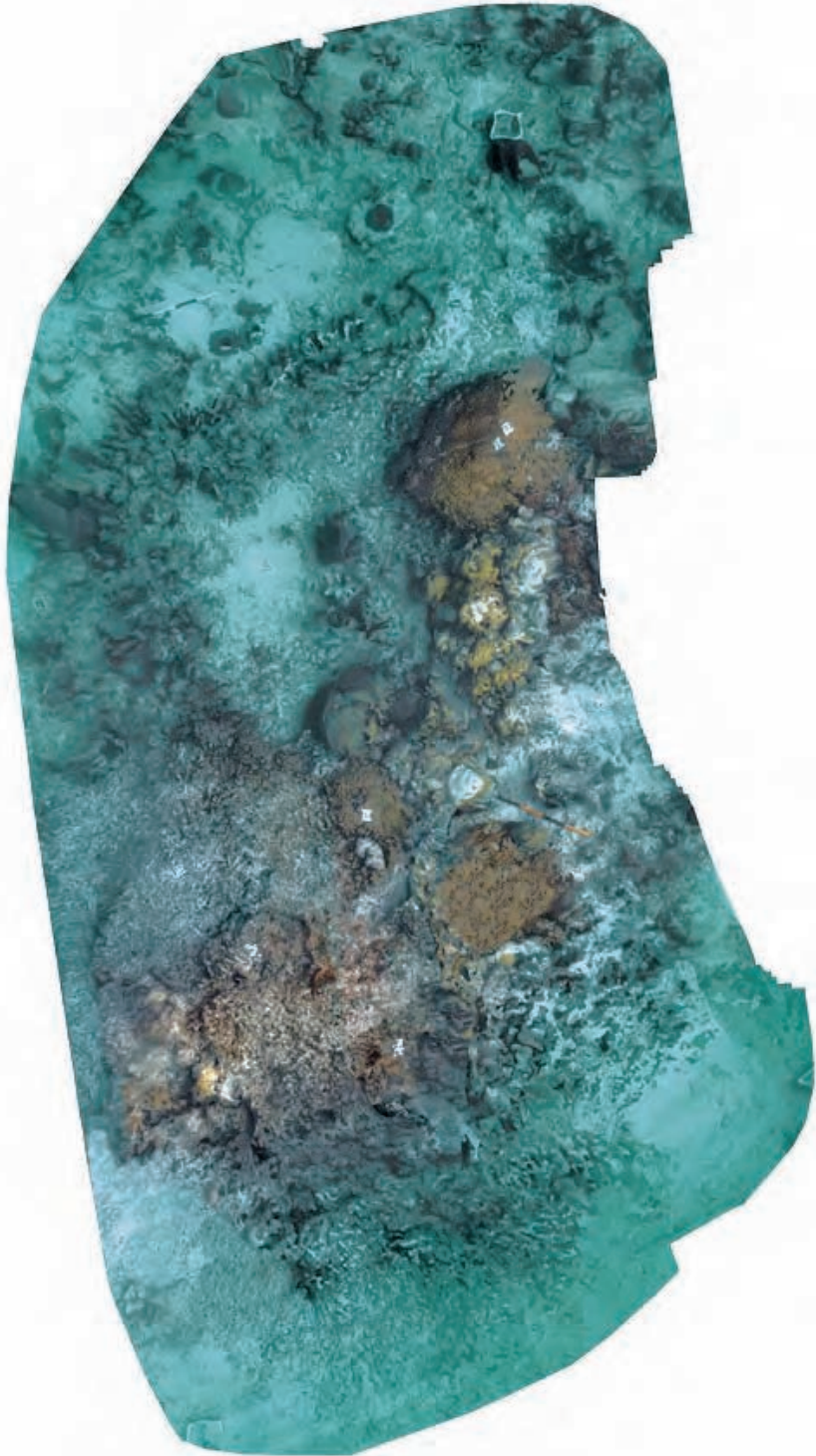
Figure 3.10: **Ortho-mosaic of an underwater scene.** The rendered mosaic simulates a high-altitude view of the scene, depicting coral-reef formations.
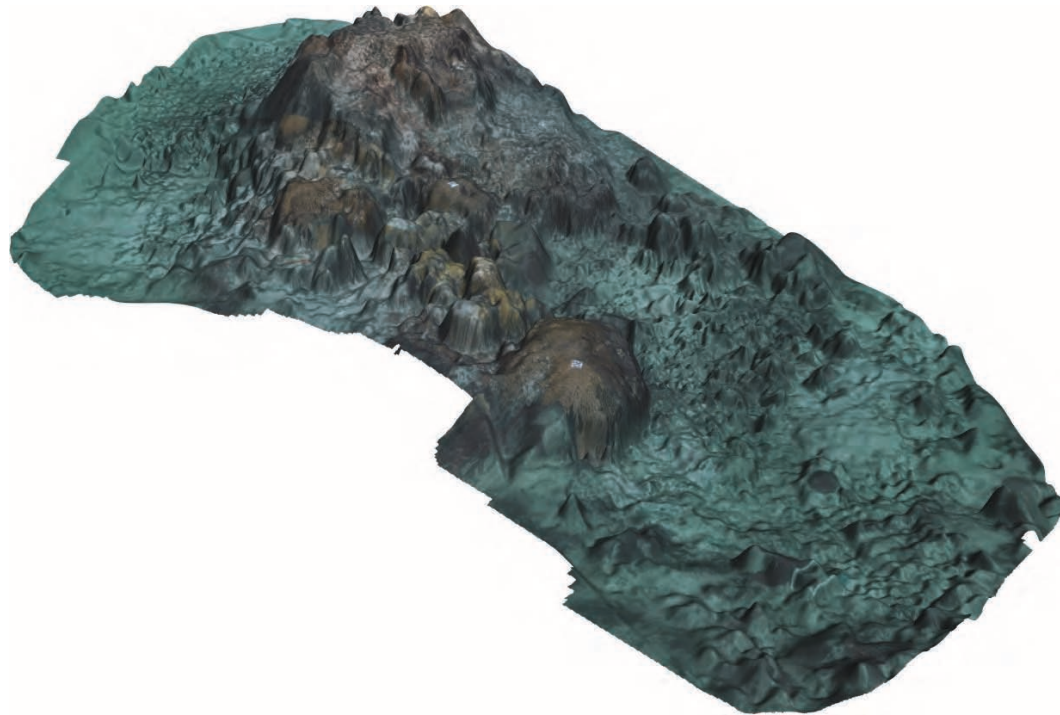
(a)
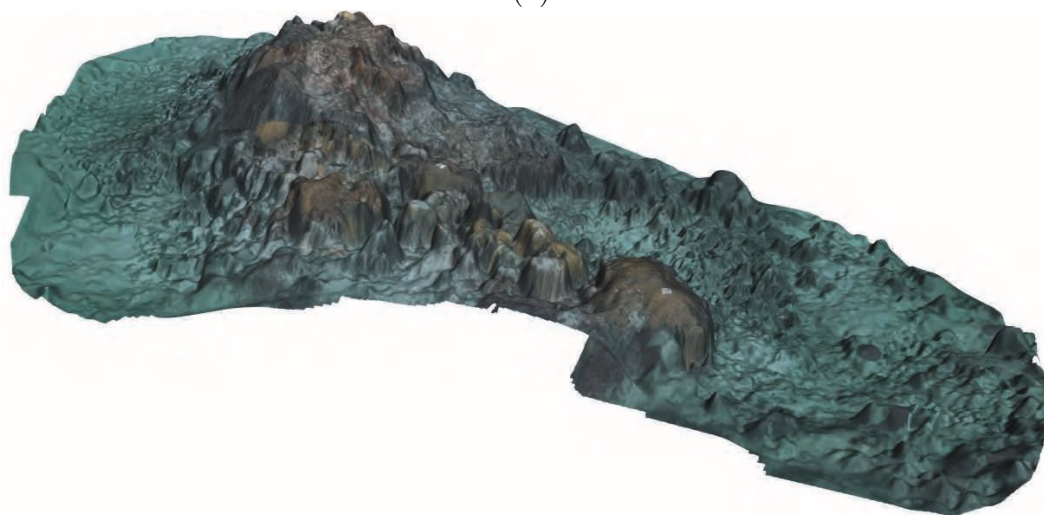
(b)

Figure 3.11: **3D model of an underwater scene.** Two views of the underwater scene model obtained by texture rendering the ortho-mosaic on the 3D surface. Here the surface was obtained using cubic interpolation.

pose estimations and *(ii)* the robustness of the algorithm when faced to common challenges such as: illumination changes, shadows, scattering, low contrast images, moving objects, specular surfaces, obstructions, objects with complex geometry, etc.

DPR-SfM has been successfully tested under various conditions, briefly discussed hereafter:

- We applied the algorithm on image sequences captured using both still and video cameras. The algorithm successfully coped with both high overlap images in video sequences and low overlap images in sequences acquired by still cameras. The DPR-SfM provides accurate estimations even in the case of temporarily static cameras, where most SfM algorithms would fail. The minimum overlap between images is given by the minimum number of views where a feature needs to be tracked before its 3D position is estimated, which can be set by the user. We generally use a minimum of 3 views per each tracked feature for redundancy.

- We tested the algorithm in the presence of occlusions and pose estimation failures (*e.g.* excessive motion blur). The pose of the camera was correctly estimated immediately after the situation disappeared. From our experiments, we have concluded that the camera pose can be correctly estimated, if there is at least $\sim 20\%$ overlap between the 3D model and the images.

- The conducted experiments included sequence acquisitions under extreme lighting conditions, obtaining accurate results: sun-flickering in shallow waters, low lighting and increased turbidity/scattering, strobe/focus lighting in deep waters.

In the discussion that follows, we generally assess the accuracy of the DPR-SfM algorithm on absolute basis as, to the best of our knowledge, there are no freely

available SfM algorithms for comparison that can cope with such large scale reconstructions.

All the data-sets presented here were acquired using various off the shelf cameras, undergoing a random trajectory with no constraints. For all the sequences, we assume that the internal parameters of the cameras are known and do not change throughout the image acquisition (*i.e.* no zooming), and the radial distortion is corrected. The estimation of the camera internal parameters and radial distortion parameters were obtained using a checkerboard pattern and Bouguet's camera calibration toolbox [12].

3.7.1   Car Scene

In this sequence we used synthetically generated images, allowing the usage of ground truth in order to quantify the accuracy of the DPR-SfM on both camera pose and scene geometry estimations.

The scene, comprised by a parked car in front of a building, was chosen to incorporate common challenges in urban environments: occlusions, object transparency, light reflections, shadows, uniform textures, etc. The rendering of the scene was carried out using ray-tracing as it is capable of producing very high degree of photorealism [116]. Ray-tracing generates images by tracing the path of light through pixels in an image plane [131], accurately modeling light alterations (reflections, shadows, transparency).

The sequence consists of 20 frames with $1,024 \times 1,024$ pixels, captured from a camera undergoing a translation motion along the building facade with a slight panning (see Figure 3.12 for some examples). The length of the translation is 10m with a mean distance between the camera and scene (the facade of the building) of $\simeq$ 9m. In order to accurately compare the results with the ground truth, we fix

Figure 3.12: **Car Scene – Input images.** Synthetic images generated using ray-tracing rendering. Here, we illustrate 4 of the 20 frames showing some of the challenges: specular objects (car body and building windows) induce inter-reflections, irregular illumination due to shadows (garage door, doors and pavement), transparency (car windows), etc.

the scale of the model by fixing the first two camera poses in the initialization step. The following camera poses are estimated by direct registration with the model (see Figure 3.13).

For comparison purposes, we used 4 types of feature extractors: Harris, Hessian, SIFT and SURF. The processing time for the sequence was $\simeq 14mins$[6]. A detailed

[6]The DPR-SfM algorithm was implemented in *Matlab*, partially using C++ routines. All the

(a)

(b)
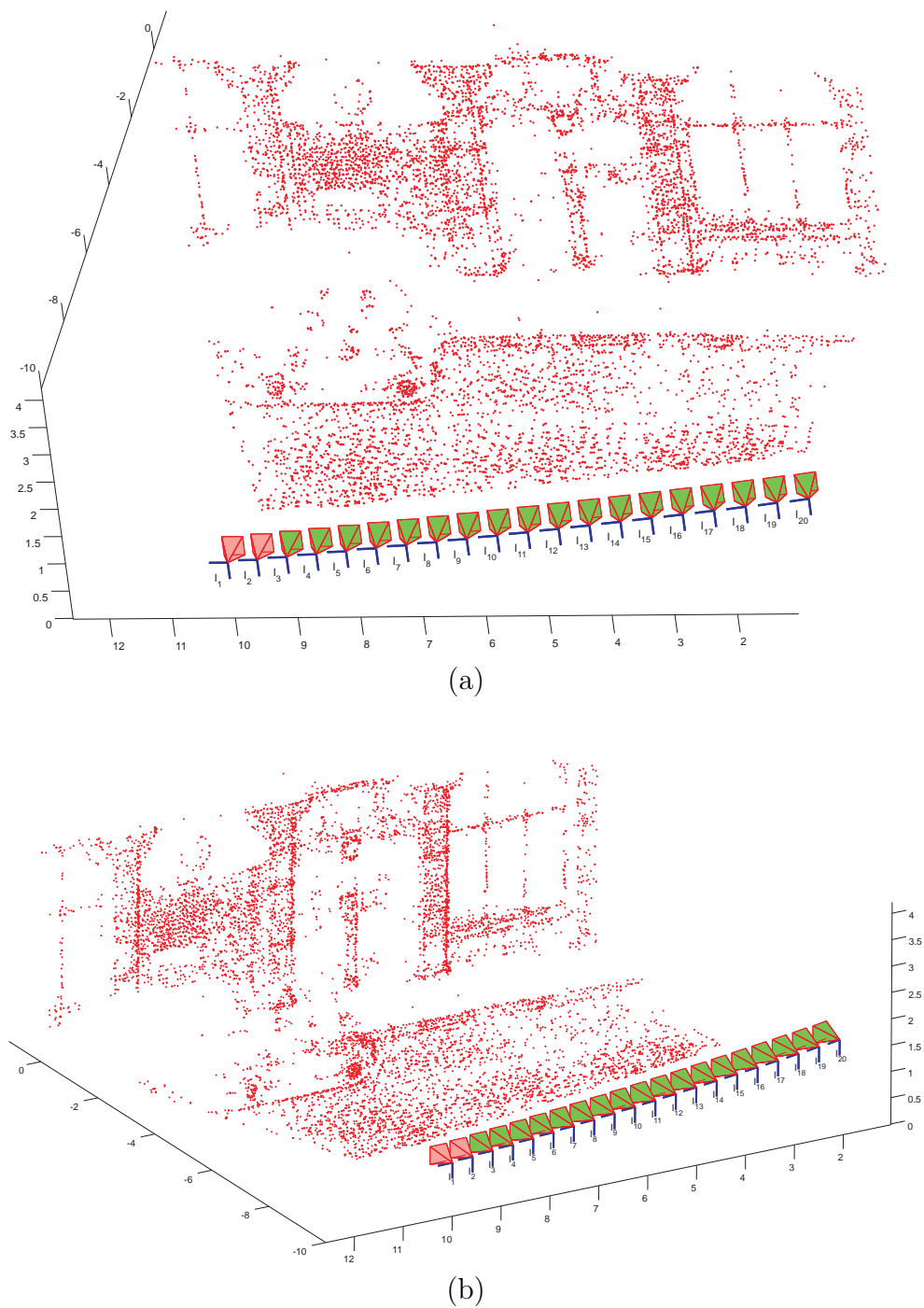
Figure 3.13: **Car Scene – 3D model.** Two views of the 3D model containing 9,800 vertices – 2,900 Harris, 2,600 Hessian, 2,400 SURF and 1,800 SIFT. The first two camera poses (shown in red) were fixed in order to recover the scale. The remaining camera poses (green) were estimated by direct registration along with scene model (red dots).

Table 3.2: **Car scene – Processing time.** Average processing time for each step (*seconds/frame*)

Feat. Extraction	Feat. Matching (ANN)	Camera Pose	Vertex Position
40.1	2.1	0.2	0.6
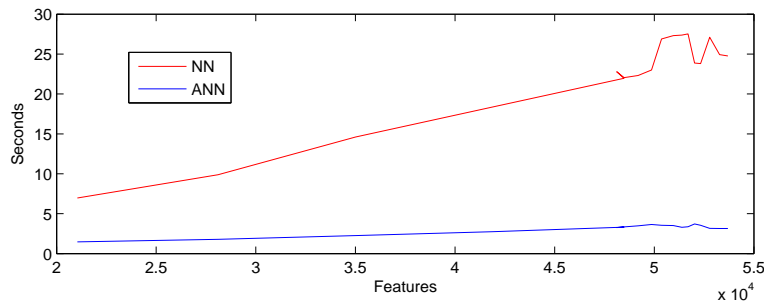
Figure 3.14: **Car scene – Scene feature matching time.** Comparison of average times for matching image and scene features vs. number of features in scene. The number of features in the image is constant (12,000). Using ANN decreases drastically the computation times.

description of execution times is presented in Table 3.2. We processed this sequence using both NN and Approximated Nearest Neighbor (ANN). The use of ANN provides a significant gain in computation time (see Figure 3.14): NN times are quadratic in the number of features while ANN times are linear.

The resulting scene model is illustrated in Figure 3.13:

- Invalid vertices formed by reflective surfaces are removed.

- Facade regions partially occluded by the car are correctly modeled (*e.g.* left of the building entrance).

The ray-tracing software was modified to generate the ground truth 3D position of the points in the scene corresponding to each pixel in the rendered images. Knowing the position of the extracted visual features, the accuracy of the model is quantified by comparing the vertex position estimations with the ground truth.

experiments presented in this work were executed on an Intel Core Duo 2.13 GHz 64-bit platform.

Figure 3.15a illustrates the average residuals (XYZ) for the vertices generated by each feature extractor. While very similar, SIFT and SURF have slightly greater residuals than Harris and Hessian, due to the nature of the extractors (see Section 2.1.3). The evolution of error in camera pose estimation is shown in Figure 3.15b.

In ideal conditions (absence of noise, distortions, blurring, etc.), both scene geometry and camera pose estimations are accurate and the error accumulation (drifting) is very small. Additionally, we want to test the robustness and accuracy of DPR-SfM for realistic scenarios. For this, we use a Monte Carlo test by adding noise to image features, aiming to:

- Assess the accuracy of the model and camera pose estimations in presence of noise.

- Robust camera pose estimation and vertex position estimation use a pre-established threshold ρ for outlier rejection (see Sections 3.4, 3.5). We test how this threshold affects the DPR-SfM accuracy.

As we consider feature localization errors to follow a normal distribution, we use a zero-mean gaussian noise with a known standard deviation σ. For each test, we fix the value of ρ and we generate the model with increasing values of σ until a valid model cannot be generated. We use two values for ρ: 1.5 and 2.5 (values typically used in DPR-SfM). The errors in scene model and camera pose are given by ε_v, ε_p and ε_a, where ε_v is the average error in vertex position estimation and ε_p is the average error in camera position estimation (both error measurements are given by the average Euclidian distance). The error in camera attitude estimations ε_a is given by the average of absolute differences over all the rotations:

$$\varepsilon_a = \frac{\sum_{i=1}^{N} |\phi_i - \overline{\phi}_i| + |\theta_i - \overline{\theta}_i| + |\psi_i - \overline{\psi}_i|}{3N}$$
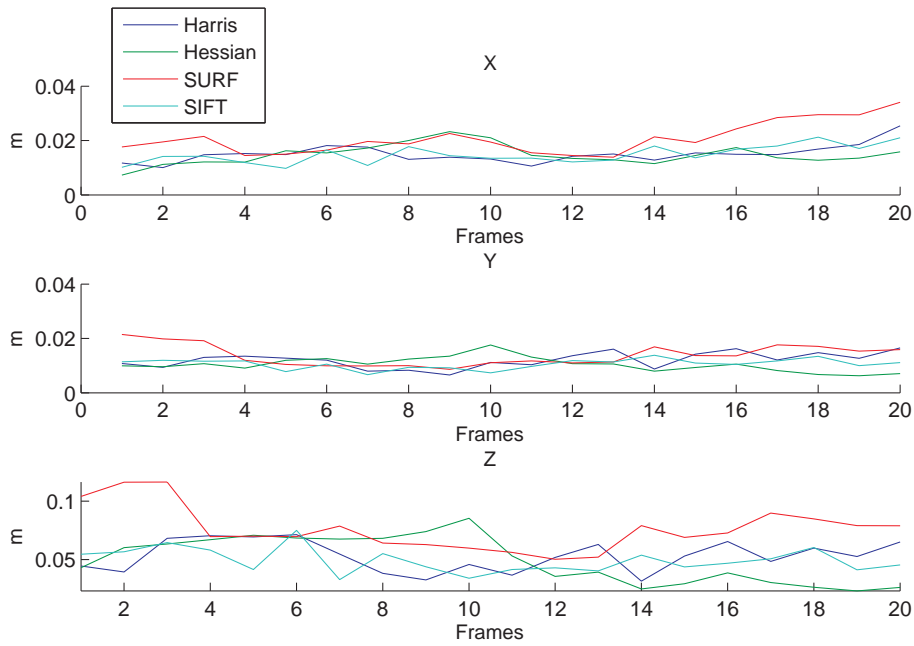
(a)
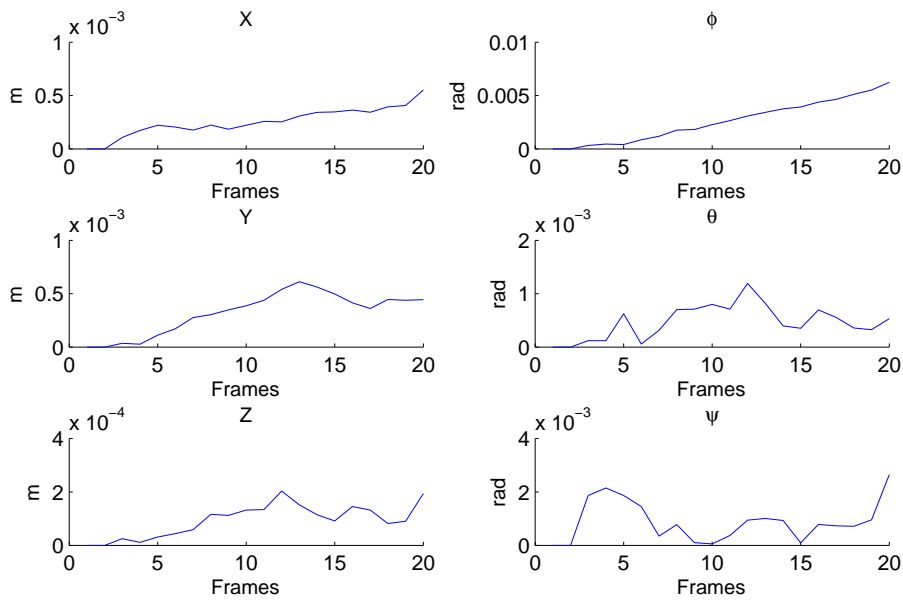
(b)

Figure 3.15: **Car scene – Reconstruction errors.** Figure (a) shows the vertex position residuals by frames, for each feature extractor. The extractors yield comparable results, with small error accumulation. In (b) we represent the error in camera pose. The residuals in both position and attitude are very small with a slow error accumulation.

Table 3.3: **Car scene – Monte Carlo test results.** The results for two values of ρ. The values for ε_v and ε_p are expressed in $m^{\cdot}10{-}3$ and ε_a is expressed in $rad \cdot 10^{-3}$. *Vert./fr.* represents the average number of vertices registered in each frame.

σ	$\rho = 1.5$				$\rho = 2.5$			
	ε_v	ε_p	ε_a	vert./fr.	ε_v	ε_p	ε_a	vert./fr.
0	47.9	1.3	0.03	2226	61.2	3.0	0.05	2449
0.2	48.9	1.4	0.10	2211	61.9	3.2	0.14	2447
0.4	48.8	1.8	0.16	2148	63.6	3.3	0.18	2446
0.6	50.5	3.3	0.21	1939	65.5	4.0	0.23	2435
0.8	51.8	4.2	0.25	1598	66.0	4.4	0.27	2416
1.0	57.9	4.1	0.31	1285	66.4	3.7	0.29	2291
1.2	63.3	7.5	0.7	970	73.8	6.2	0.32	2309
1.4	69.1	14.5	0.81	848	74.0	5.1	0.34	2180
1.6	65.7	19.6	0.85	329	81.8	4.7	0.43	2035
1.8	–	–	–	–	85.6	6.0	0.45	1857
2.0	–	–	–	–	90.7	7.0	0.50	1671
2.2	–	–	–	–	106.6	7.8	0.61	1471
2.4	–	–	–	–	124.6	7.9	0.72	1237

where $(\overline{\phi}_i \, \overline{\theta}_i \, \overline{\psi}_i)$ is the estimated orientation and $(\phi_i \, \theta_i \, \psi_i)$ is the ground truth orientation for camera pose i; N is the total number of frames.

Table 3.3 details the results of the Monte Carlo tests. The noise in image features has little impact on both model and camera pose estimations, especially when a low threshold is used. However, as the noise level increases, the use of a very restrictive threshold highly reduces the number of vertices (see Figure 3.16). This affects the camera registration precision, ultimately leading to the impossibility to generate a valid model.

Figure 3.17 illustrates the distribution of the noise in the image features for each threshold. The DPR-SfM can generate a valid scene model even in the presence of an overwhelming number of outliers (more than 60%).
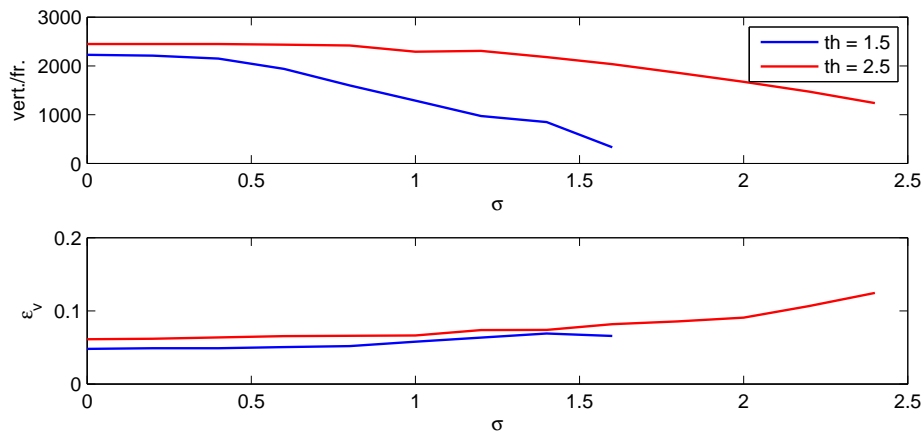
Figure 3.16: **Car scene – Feature evolution in Monte Carlo test.** The average number of vertices drops as the noise level increases (Top Figure). Using a more relaxed threshold keeps a larger number of vertices but slightly decreases the accuracy of the vertices (Bottom Figure).
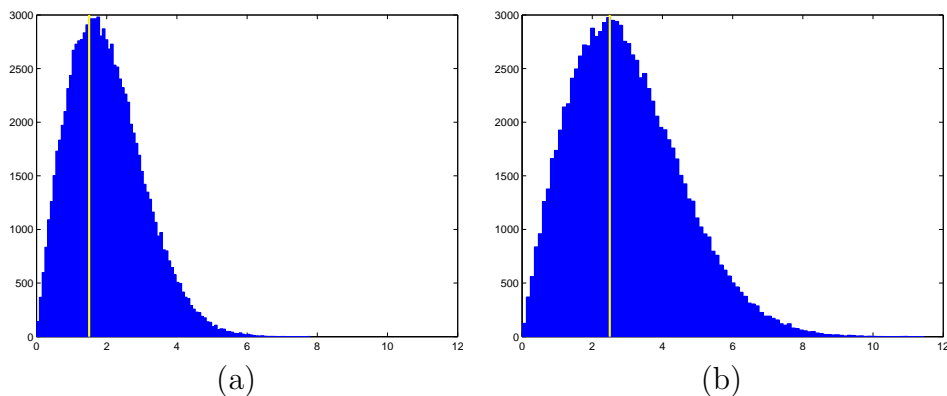
Figure 3.17: **Car scene – Image feature noise distribution.** The two histograms correspond to the maximum noise level where DPR-SfM could generate a valid model for $\rho = 1.5$ and $\rho = 2.5$ respectively. In (a) 35.4% of the features fall within the threshold (yellow line) while in (b) 39.2%.

3.7.2 Water-tank Sequence

This sequence is part of a series of experiments, used for testing the performance of the DPR-SfM algorithm under realistic conditions. The dataset was acquired by a camera mounted on the Johns Hopkins University (JHU) ROV at the JHU
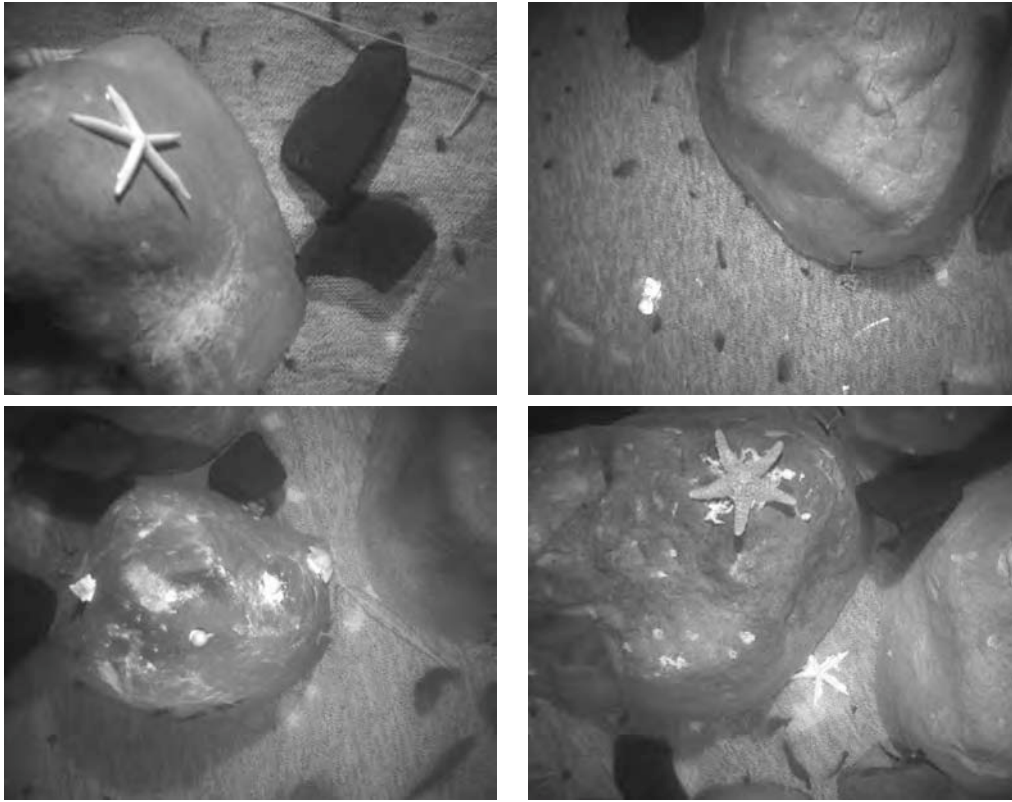
Figure 3.18: **Water-tank sequence – Input images.** Sample images from the dataset depicting some of the objects present in the scene.

test tank. The bottom of the tank was populated with rocks and shells, simulating the appearance and geometry of a typical seafloor scene. The size of the scene is $\simeq 5 \times 5$m. The sequence, comprised of 3,500 images (see Figure 3.18), was acquired at a constant distance of 1.2m above the bottom of the tank. After the visual survey, the tank was drained and scanned with a Leica Geosystems laser scanner, obtaining 3.8 millon points with an estimated accuracy of 1.2mm.

The objective of this experiment was to assess the 3D reconstruction accuracy of the DPR-SfM using the ground truth, under a realistic scenario. For this, we applied the DPR-SfM on the full sequence of 3,500 images. The resulting model, consisting of 610,000 vertices, is illustrated in Fig. 3.19 along with the estimated
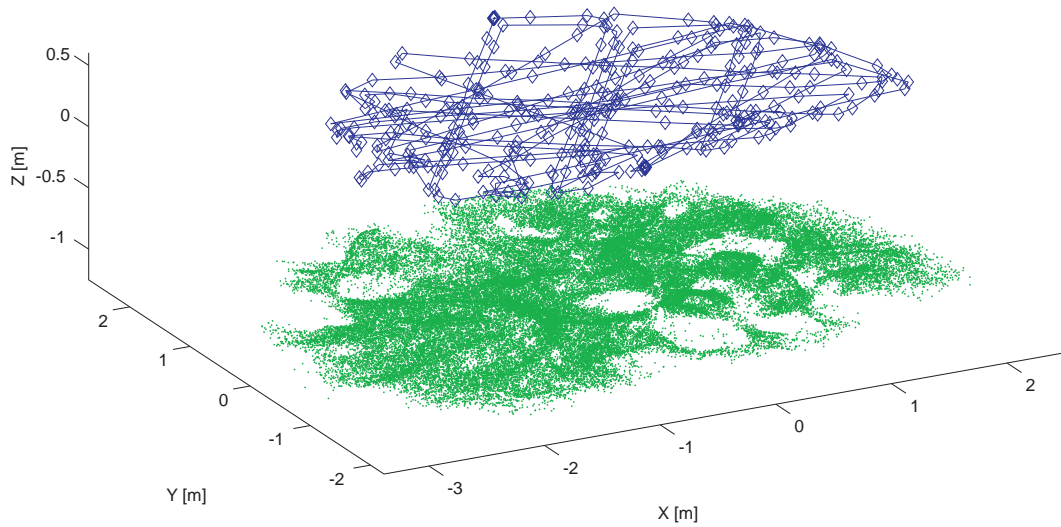
Figure 3.19: **Water-tank sequence – Scene model and camera trajectory.** The model consists of $610,000$ vertices, shown in green. The camera trajectory is marked in blue. Both the model and the camera trajectory were subsampled for illustrative purposes.

camera trajectory.

The accuracy of the reconstruction was quantified by comparing the obtained model with the laser scan. For this, we first manually aligned the two models using 3D point correspondences. The alignment was further refined using Iteratively Closest Point (ICP) [153]. In order to assess the accuracy of DPR-SfM, we triangulate the surfaces, obtaining a continuous representation of the two models. By sub-sampling the continuous surfaces, we quantify the reconstruction error using the Hausdorff distance [98] between the two models. Figure 3.20 illustrates the error distribution within the reconstruction. The average error was 0.011m (0.91% of the scene depth) and the maximum error was 0.092m (7.6% of the scene depth). Nevertheless, the wide regions of the tank bottom with higher error correspond to changes in the carpet shape as the tank was drained for the laser scanning. For details on the acquisition process and comparative 3D reconstruction results, refer to [113].
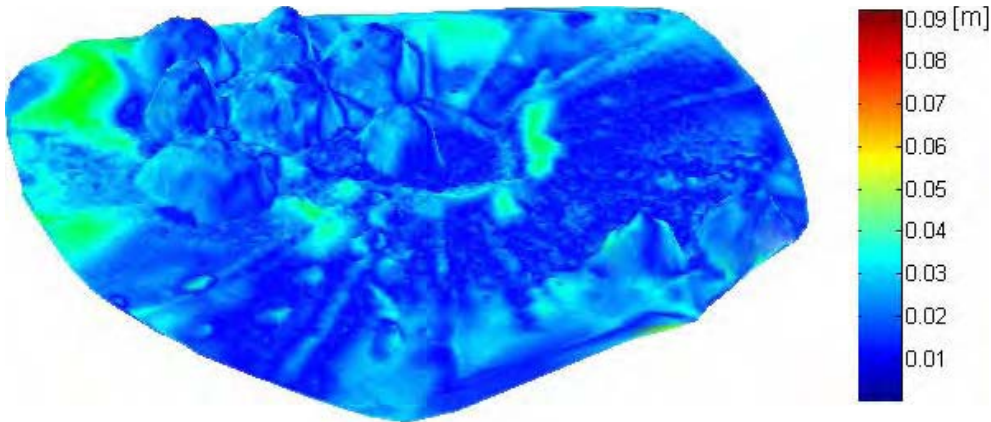
Figure 3.20: **Water-tank sequence – Error distribution.** The color encoded by error magnitude, lighter areas correspond to higher errors. The errors on the flat areas do not correspond to errors in the estimated 3D model, but to the carpet being buoyant.

3.7.3 Rocks Loop

In this experiment, we discuss the capability of the DPR-SfM algorithm to model outdoor, unstructured scenes.

The scene, illustrated in Figure 3.21, is formed by a random arrangement of rocks. The image sequence was acquired using a monochrome camera with a resolution of 696×520 pixels. A sample of the images is shown in Figure 3.22. During the acquisition, the camera was looking downwards, towards the scene, and rotated so that its y axis is tangent to direction of movement, simulating a down-looking camera mounted on an UUV.

The sequence of 740 frames was processed using HarrisAffine-SURF and SURF-SURF, yielding 170,000 vertices – 86,000 Harris and 84,000 SURF (see Figure 3.23a). We obtain an average back-projection error of 1.72 pixels, with 1.67 pixels for Harris and 1.75 pixels for SURF. The average track length for Harris is 12.1 frames while for SURF is 14.3 frames. This shows that, in case of unstructured environments, Harris provides better precision in feature localization, while SURF is more robust to image transformations.

Figure 3.21: **Rocks loop – Overview.** The scene is comprised by a round area with a diameter of $\simeq 8$m. The area is covered by rocks with varying sizes and textures, ideal for simulating an underwater relief.
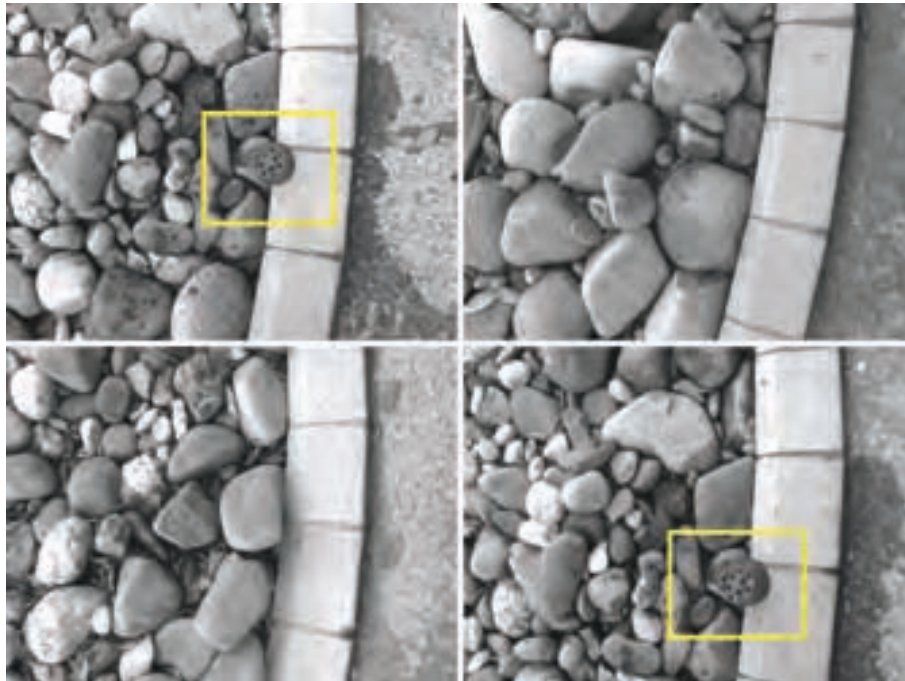
Figure 3.22: **Rocks loop – Input images.** A sample set of the input images. We used a plastic object (highlighted in yellow) to mark the beginning and ending of the loop.

(a)

(b)
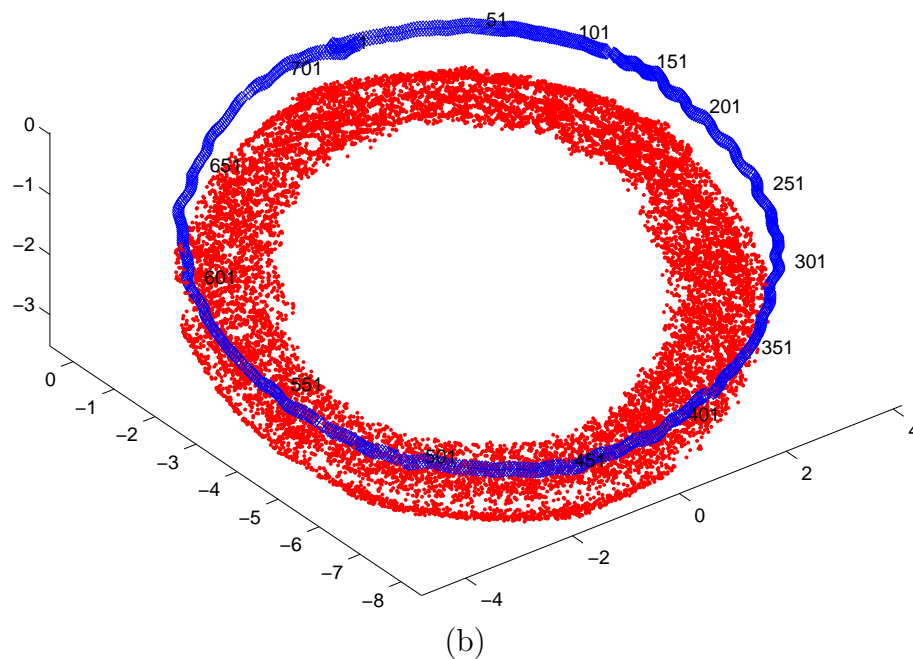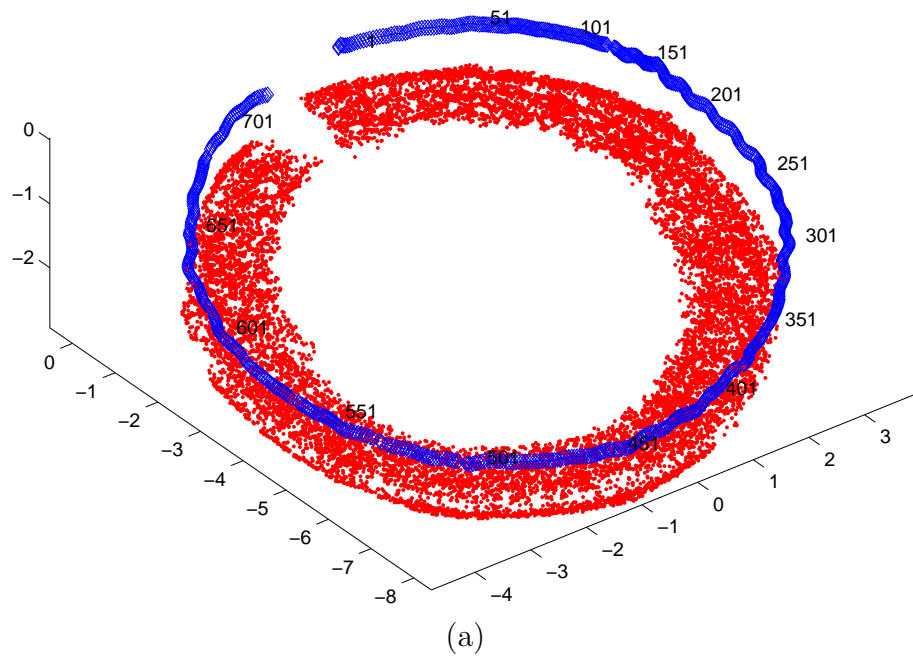
Figure 3.23: **Rocks loop – 3D model and camera trajectory.** Figure (a) illustrates the resulting model along with the estimated camera trajectory. The drifting generates a gap in the model where the loop should be completed. The model is corrected after loop closure detection and BA (b).

The major drawback of these types of environments is the impossibility of an exact quantification of the reconstruction accuracy due to the lack of ground truth. We overcome this by designing the camera trajectory to have a loop form, so that its beginning overlaps its ending (see Figure 3.22). This way, we establish constraints between the two ends of the loops (see Appendix B). After detecting the loop closure and applying BA, we correct the estimation errors up to a high degree of precision (see Figure 3.23b). We use this corrected model as the ground truth and compare it with the original result, quantifying the accuracy of the DPR-SfM. Figures 3.24 and 3.25 illustrate the error evolution in vertex position and camera pose respectively.

3.7.4   Pool Trials

We present one of the experiments we have conducted in the Underwater Robotics Center of the University of Girona. Shown in Figure 3.26a, the center is endowed with a pool used for performing tests of small class underwater vehicles. The Underwater Vehicles (UVs) are controlled and monitored from a submerged control room, allowing the researchers to have live panoramic view of the experiments.

The tests were performed using *Ictineu*, an open frame, small class Autonomous Underwater Vehicle (AUV) (see Figure 3.26). The modular design of Ictineu allows us to set up different types of sensors, depending on the mission environment and purpose. For our experiments, we have used an off the shelf, low end, 384×288 pixels monochrome camera. The camera was mounted on Ictineu on a down-looking configuration.

The AUV was set to follow predetermined trajectories, while the camera was acquiring images of a poster mounted on the bottom of the pool, simulating a seafloor scene.

The aim of the experiments is to observe the behavior of the DPR-SfM algorithm
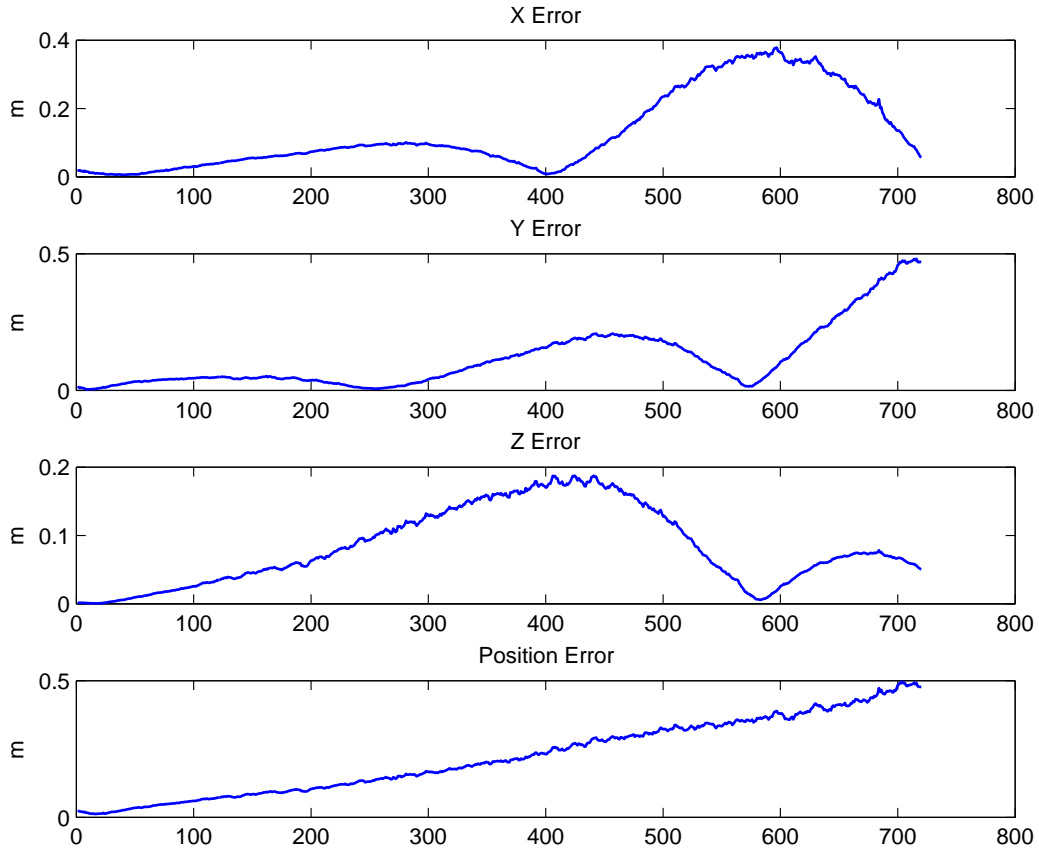
Figure 3.24: **Rocks loop – model estimation errors.** The evolution of the vertex position estimation errors by frame for each degree of freedom. Bottom plot illustrates the total vertex estimation error.

in the presence of flat scenes. In these cases (*e.g.* sandy seafloor regions, building facades, etc.), SfM algorithms fail due to the lack of parallax. Our dual approach, on the other hand, allows us to handle these situations (see Sections 3.2 and 3.4).

In the presented experiment, we have acquired a sequence of 150 frames while Ictineu was following a straight trajectory, maintaining a constant distance to the poster of $\simeq 1.5m$ (refer to Figure 3.27 for examples of images from the dataset). During the experiment, there was a brief communication error between Ictineu and
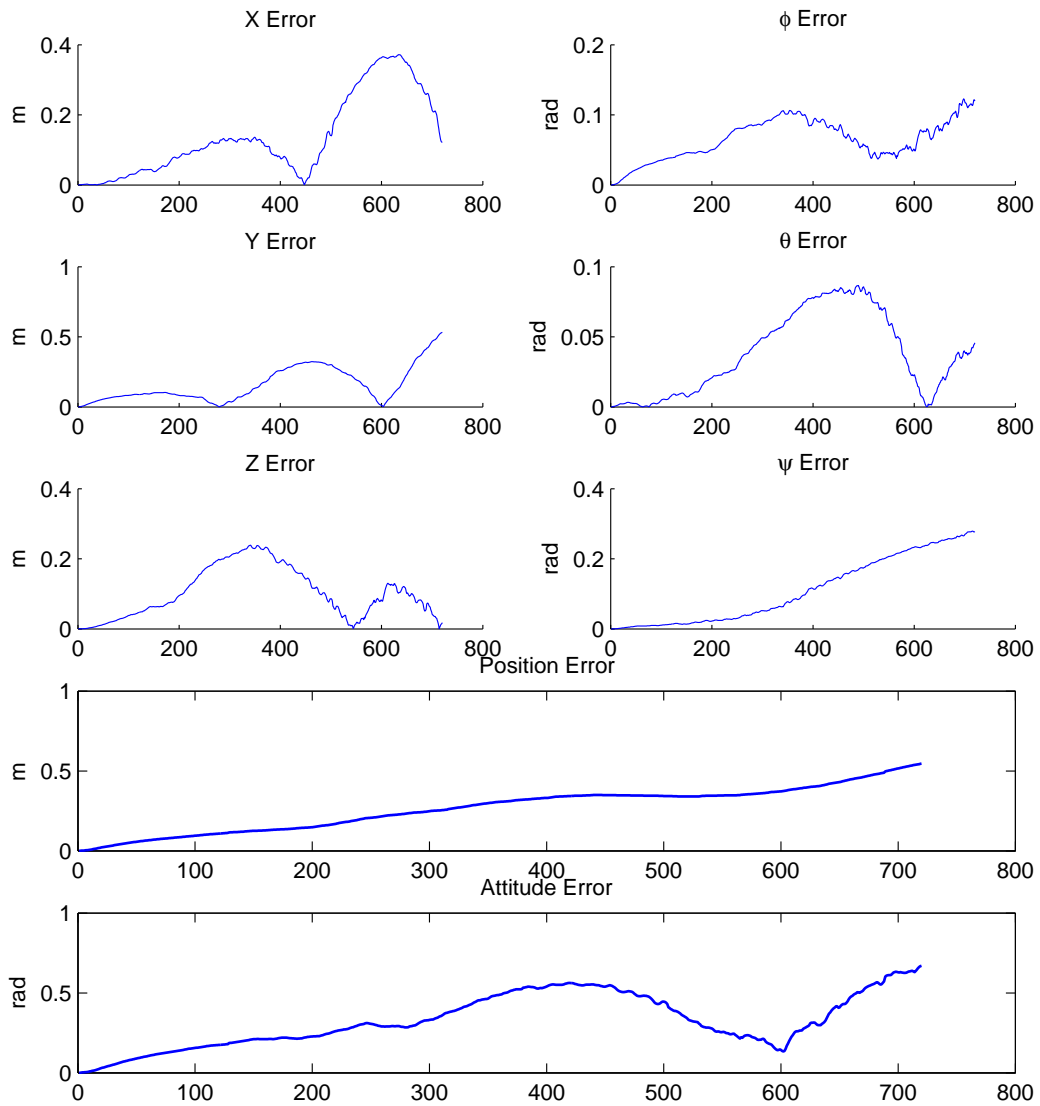
Figure 3.25: **Rocks loop – Camera pose errors.** Camera pose estimation error evolution by frame, for each degree of freedom. Bottom two plots illustrate total estimation errors for position and attitude respectively.

(a)
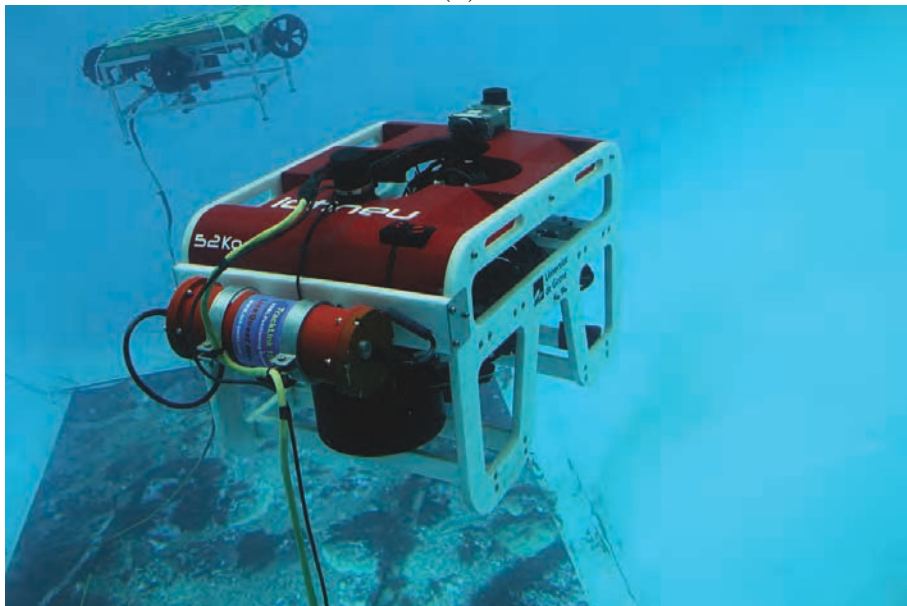
(b)

Figure 3.26: **Pool trials – Experimental setup.** (a) Underwater Robotics Laboratory of the University of Girona and (b) Ictineu AUV (foreground) with the seafloor poster during the experiments, photographed from the submerged control room.
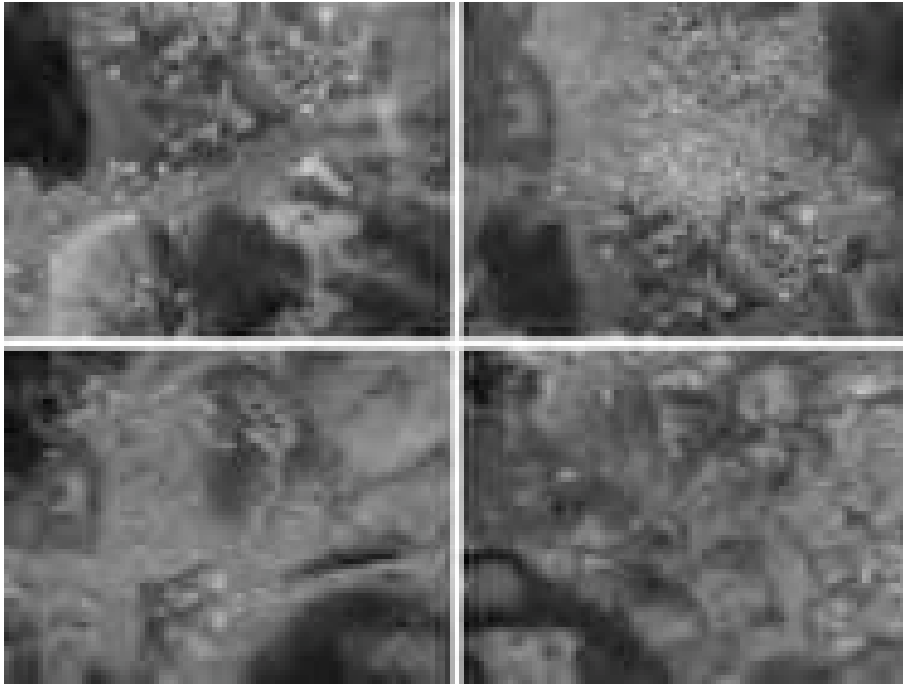
Figure 3.27: **Pool trials – Input images.** Images from the sequence of the poster simulating an underwater scene.

the control room generating some invalid frames to be captured. This offered an ideal situation to test the robustness of the DPR-SfM algorithm when faced to camera obstructions / errors.

After processing the sequence, we obtained 10,000 HarrisAffine and 7,000 SURF vertices. In both cases, we used SURF for description. Figure 3.28 illustrates the result of the reconstruction. The gap in the camera trajectory corresponds to the communication error. DPR-SfM was able to recover from this situation, correctly registering the following frames.

In order to account for the precision of the reconstruction we first first determine the average scene plane using Least-Squares fitting to the 3D vertices. As the scene is planar, we define the reconstruction error as the Euclidean distance between the plane and the 3D vertices. The distribution of the reconstruction error is illustrated
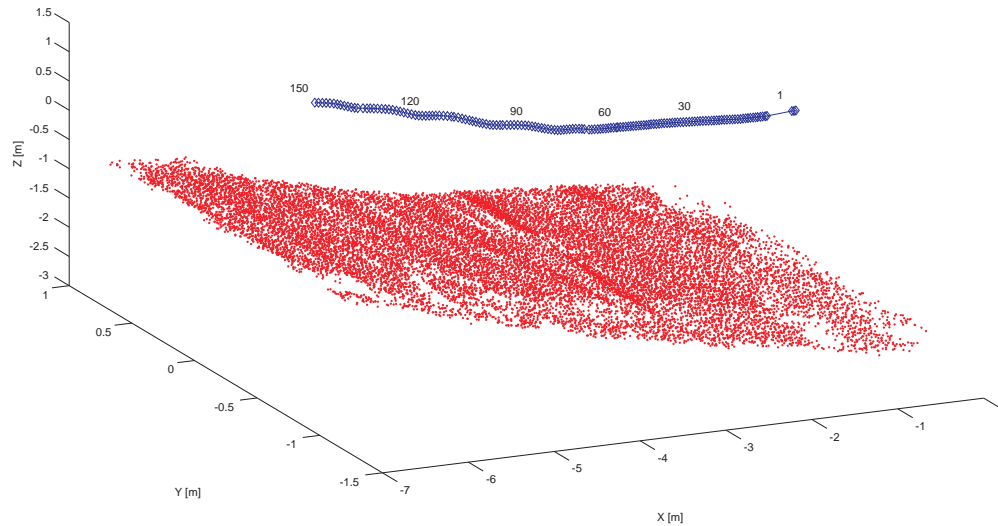
Figure 3.28: **Pool trials – 3D model and camera trajectory.** 3D model of the poster and camera trajectory. There is a gap in the camera trajectory due to a communication error between the UV and the control room.
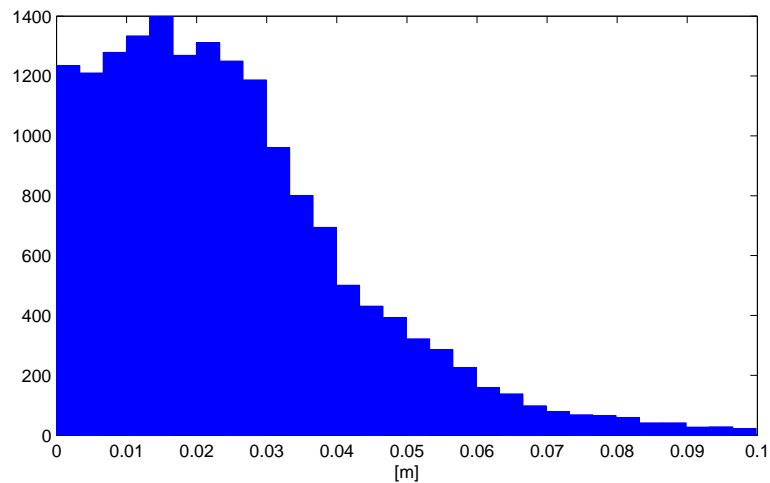
Figure 3.29: **Pool trials – Reconstruction error histogram.** We calculate the reconstruction error as the Euclidean distance between the scene plane and the vertices.

in Figure 3.29.

3.7.5   Coral Reef Sequence

Here we discuss the results obtained from sequence depicting a coral reef area. This dataset is part of a larger survey of a benthic habitat undertaken in shallow waters in The Bahamas. The images were acquired by the University of Miami (UoM) using a hand-held HD camera. The sequence consists of 1,100 images of 962×540 pixels (the resolution of the images was reduced from 1920×1080 due to interlacing). The area was surveyed with the camera following a "lawnmower" trajectory, with partial overlap between adjacent columns. This provides a complete coverage of the area while offering additional constraints in the model.

The sequence covers $\simeq 150m^2$ and was chosen to include different types of topologies and textures often found in underwater scenes. Figure 3.30 depicts typical entities found in the dataset. We recover the scene model using HessianAffine-SURF and SURF-SURF features with an outlier rejection threshold $\rho = 1.5$, obtaining 270,000 vertices (130,000 HessianAffine and 140,000 SURF). Figure 3.31 illustrates the scene model and camera trajectory – the number of vertices in the model has been reduced 10 times in order to avoid cluttering in the figure.

The aim of this experiment is to asses the accuracy of the model with respect to the texture types present in the scene. For this, we consider the average back-projection error for each reconstructed vertex. Figure 3.32 shows that the precision of the vertex reconstruction is highly related to the saliency of the corresponding image features[7]. Moreover, it can be observed that there is a strong correlation between the vertex precision and the type of its neighboring scene type (*e.g.* vertices in rocky and coral reef areas are more accurate than ones in sandy areas).

Using the constraints between adjacent columns in the camera trajectory (see

---

[7]The saliency represents a quality measurement of the features. It is related to the image gradient in the neighborhood of the feature, so that higher saliency corresponds to more accurate and discriminant features.

Figure 3.30: **Coral reef sequence – Input images.** Sample images from the input sequence showing different types of regions: coral reef formations, rocks, algaes, sand, etc.

Appendix B), we apply BA on the sequence. We use the result as reference to quantify the errors in the reconstruction. The error evolution in camera pose estimation is illustrated in Figure 3.33. As the camera is registered directly with the model, the errors do not increase significantly along the columns in the camera trajectory, reducing drastically the error accumulation.

3.7.6 Mequinenza Sequence

In this experiment, we aim to test the behavior of the DPR-SfM under difficult image conditions. The sequence was captured in the Ebro river, Mequinenza, Catalunya by the Ictineu AUV using a down-looking monochrome camera. Due to the high turbidity in the water, we used additional lighting, which increased the visibility but induced shadows and non-uniform illumination patterns. Moreover, due to back-scattering, the images have low contrast (see Figure 3.34).

The sequence, comprised by 2,900 frames of 384×288 pixels resolution, was first pre-processed using Contrast Limited Adaptive Histogram Equalization (CLAHE)

(a)

(b)
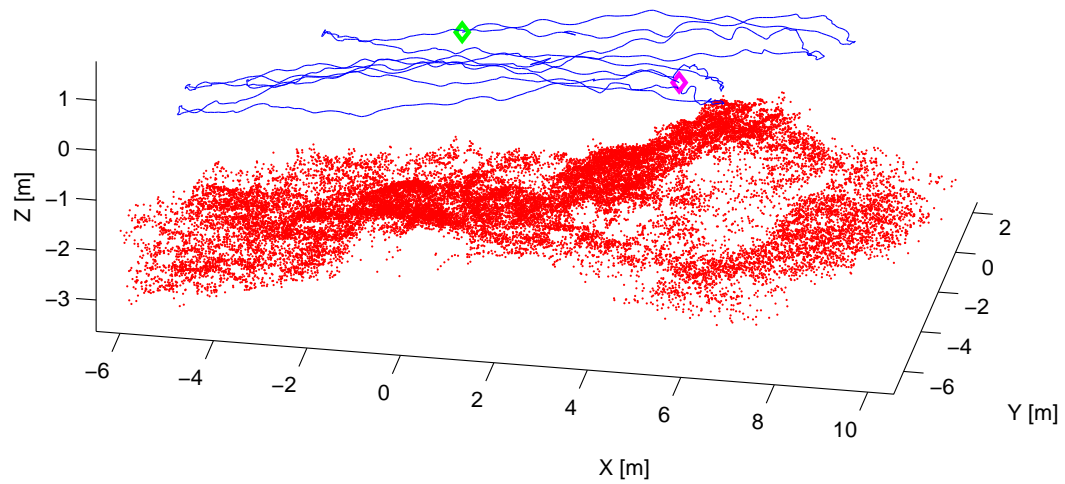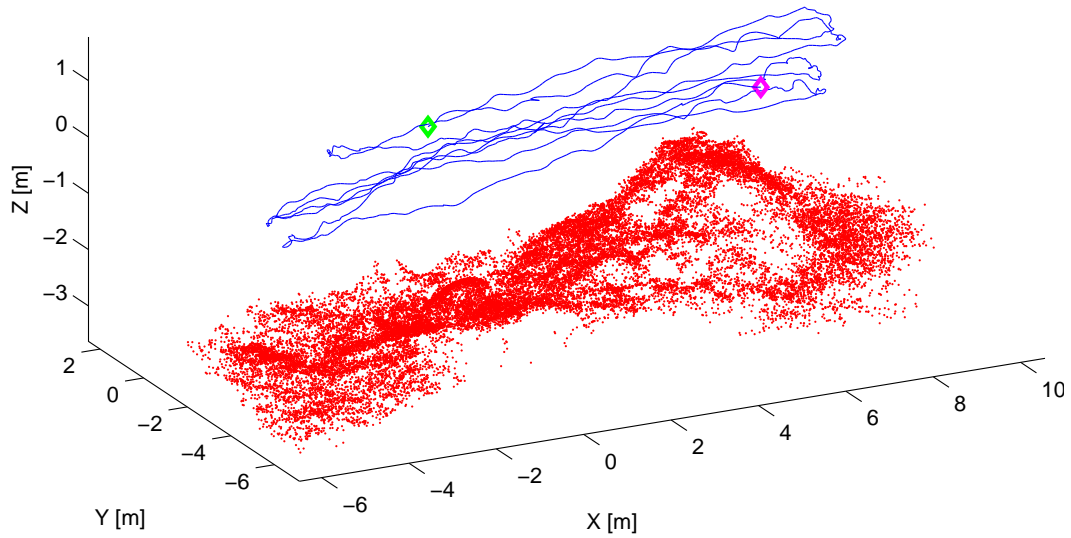
Figure 3.31: **Coral reef sequence – 3D model and camera trajectory.** (a) simplified scene model and camera trajectory: green and magenta markers show the beginning and end of trajectory respectively; (b) another view of the scene model.

(a)

(b)

(c)
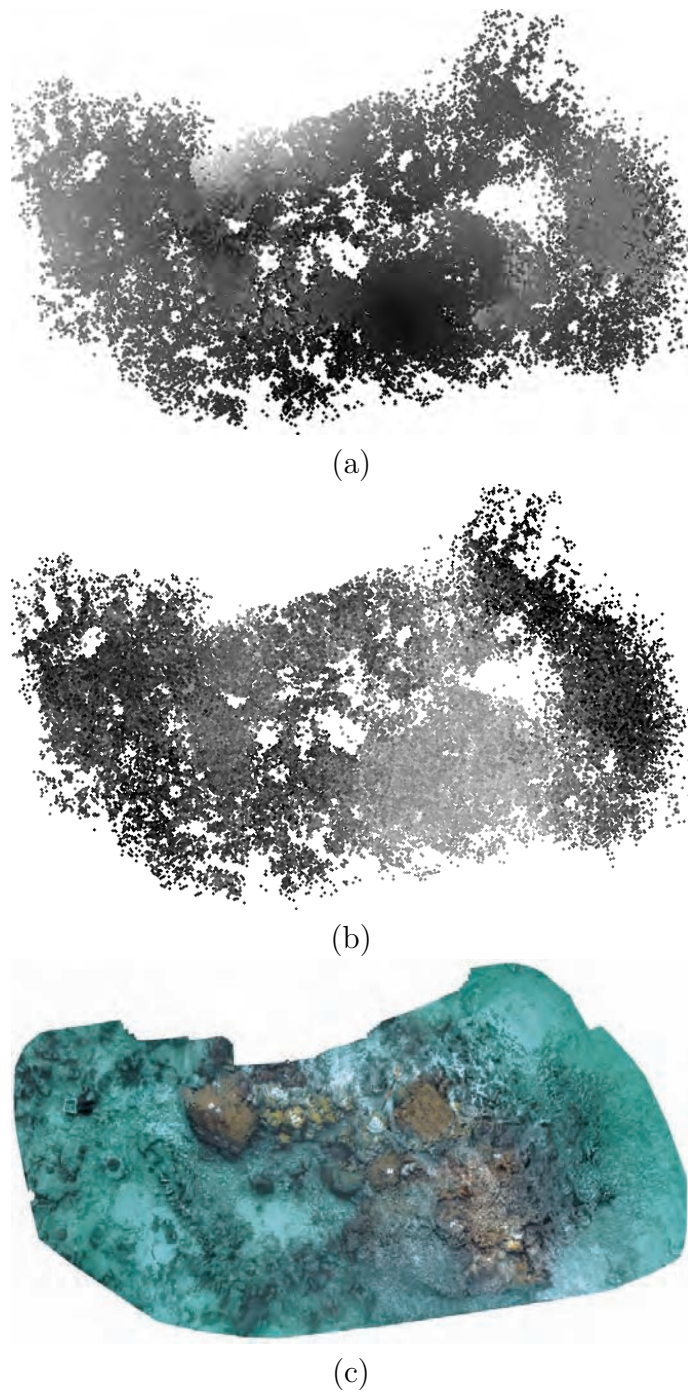
Figure 3.32: **Coral reef sequence – Vertex error.** Figure (a) shows the back-projection error distribution. Darker values correspond to higher accuracy. The distribution of image feature saliency is shown in (b); lighter values correspond to higher saliency. The ortho-mosaic of the scene is provided for reference in (c), showing the relation between region types, feature saliency and vertex accuracy.
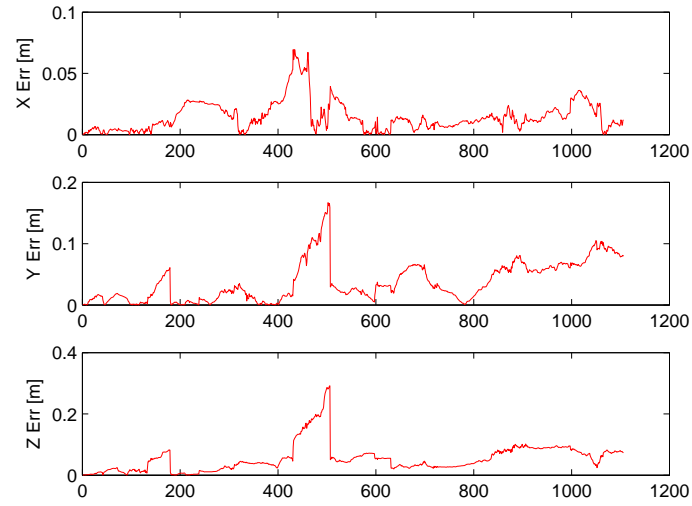
(a)
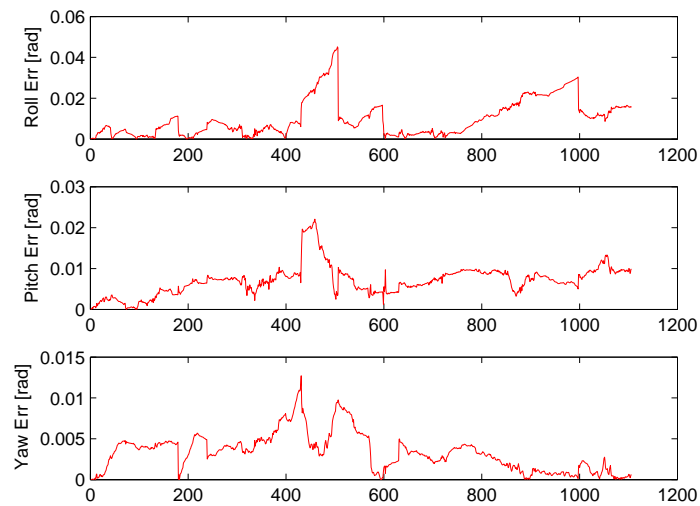
(b)

Figure 3.33: **Coral reef sequence – Camera pose errors by frames.** (a) camera pose errors and (b) camera attitude errors.

[155] in order to enhance the quality of the images. Using SURF-SURF features, we obtained 220,000 vertices. Figure 3.35 illustrates the resulting camera trajectory and scene model (the number of features has been reduced for illustration clarity). The model shows an environment with complex geometry, also, the trajectory of

Figure 3.34: **Mequinenza sequence – Input images.** Image samples depicting some of the challenges of sequence: scattering, light absorbtion, shadows, complex scene geometry, etc.

the camera depicts a motion of Ictineu with sudden changes in heading and motion direction due to the water currents.

Using an outlier rejection threshold ρ of 1.5, we obtain an average back-projection error for the whole model of 0.9 pixels.

3.7.7 Urban Experiment

This experiment was aimed at testing the DPR-SfM algorithm for large-scale urban modeling applications. For this, we acquired a sequence of Unirii Square in Timisoara, Romania. The square, illustrated in Figure 3.36, has a rectangular shape, measuring $\simeq 155 \times 120$m and is surrounded by historical buildings of various shapes and textures. We used a low-end Pentax Optio A30 digital camera for video acquisition, while walking through the square following a loop trajectory. The resulting

(a)

(b)
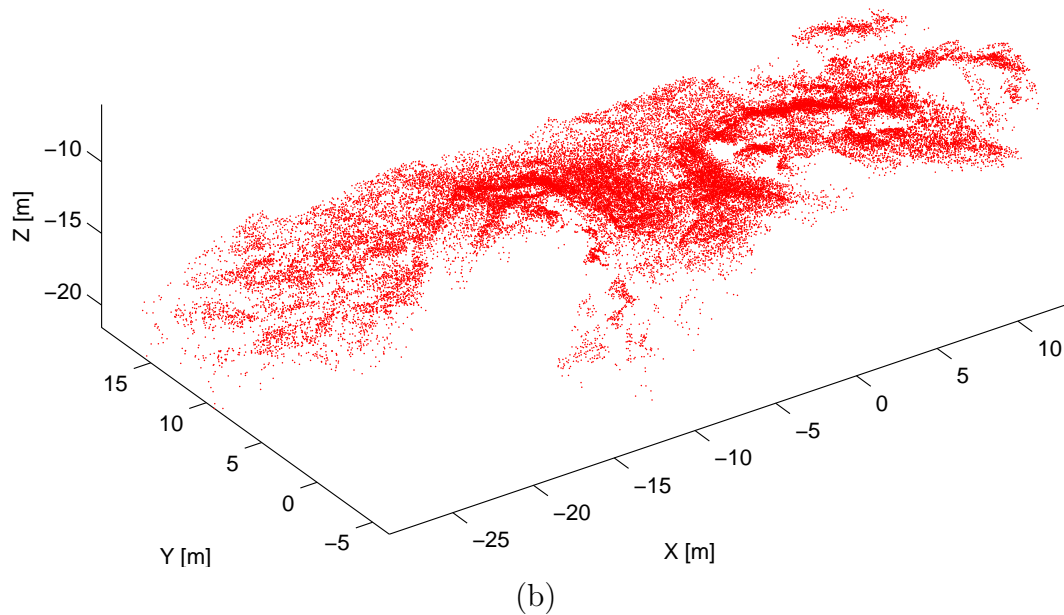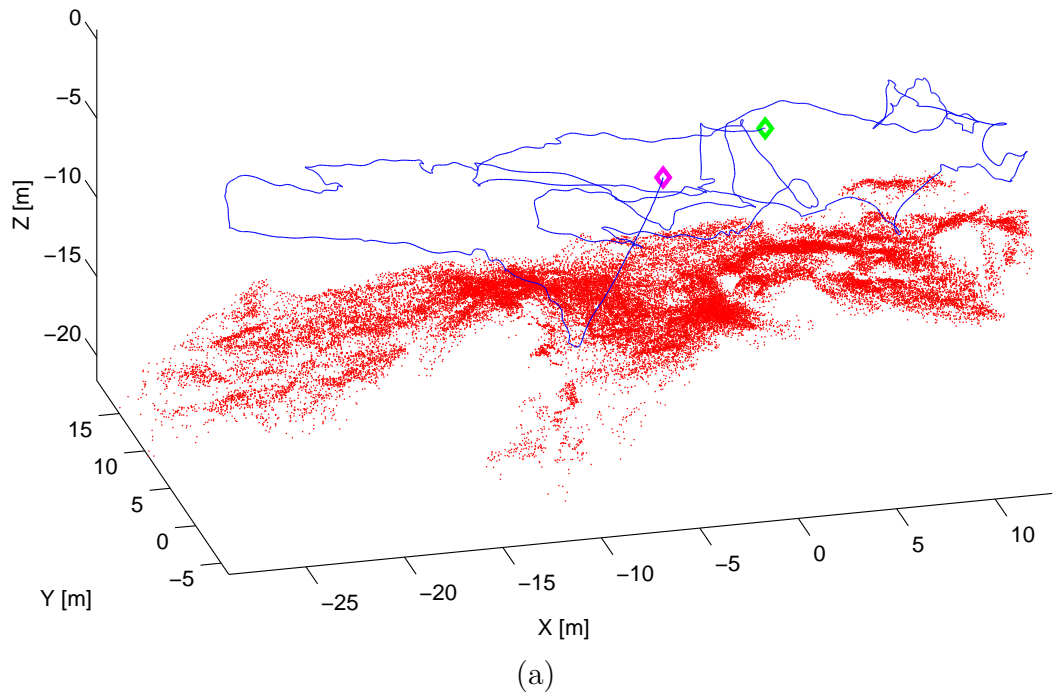
Figure 3.35: **Mequinenza sequence – 3D model and camera trajectory.** (a) scene model along with camera trajectory: green and magenta markers show the beginning and end of trajectory respectively; (b) another view of the scene model.

Figure 3.36: **Urban experiment – Overview of the Unirii Square.** Aerial view of the Unirii Square.

image sequence contains 961 frames of 640×460 pixels in resolution (see Figure 3.37).

After applying DPR-SfM on the sequence using SURF-SURF, the resulting model, shown in Figure 3.38, contains 240,000 vertices. The drift due to error integration is obvious at the loop closure, where the facades of the buildings are repeated (see Figure 3.38b). The main reason behind the high drift in this dataset is the decreased precision in feature localization due to the low quality of the images: the camera uses a high compression ratio MPEG2 codec, which results in loss of details in images.

After the loop closure detection (see Section 4.3.4), we corrected the model, as shown in Appendix B. The result is shown in Figure 3.39.

Considering the model after BA as the ground truth, we calculate the camera

Figure 3.37: **Urban experiment – Input images.** Sample images from the dataset, showing some of the typical challenges such as moving objects, occlusions, sun flickering, lack of texture, etc. Also, the partial overlap between the first and last image can be clearly observed.
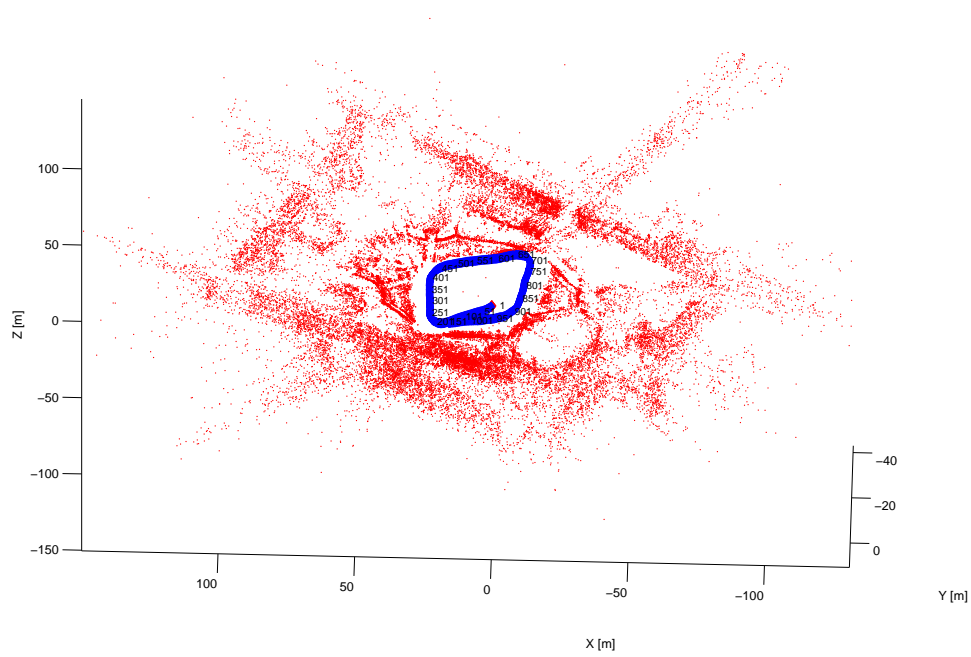
(a)
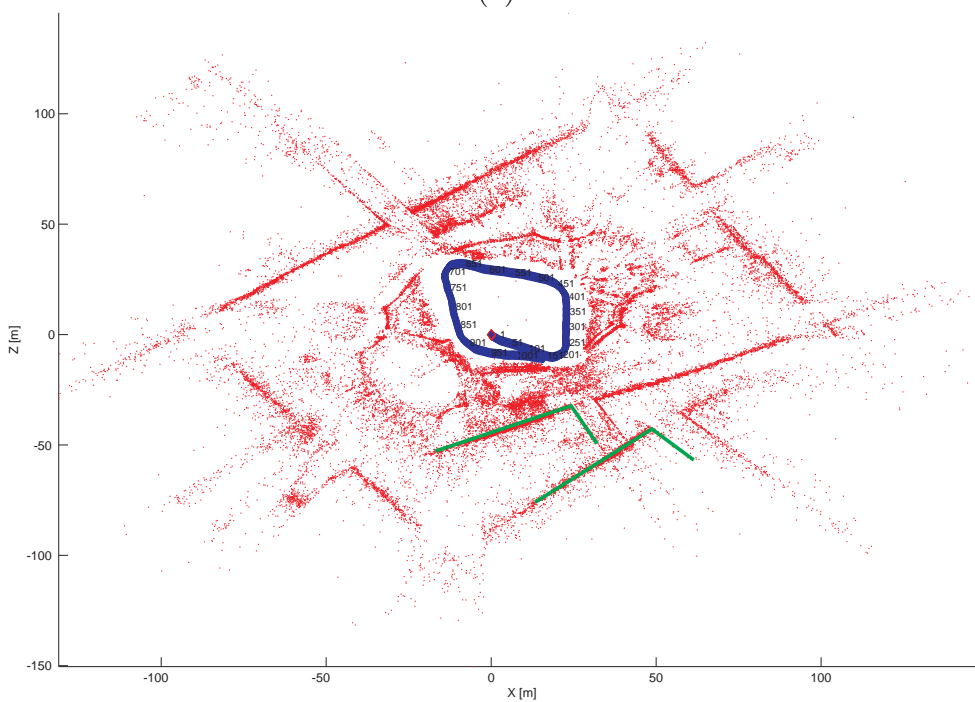
(b)

Figure 3.38: **Urban experiment – 3D model and camera trajectory.** (a) scene model (red) along with camera trajectory (blue) – the number of vertices in figure was reduced by 10 times to avoid cluttering; (b) top view of the scene model clearly depicting the drift at the loop closure (repeated edges at the bottom marked in green).
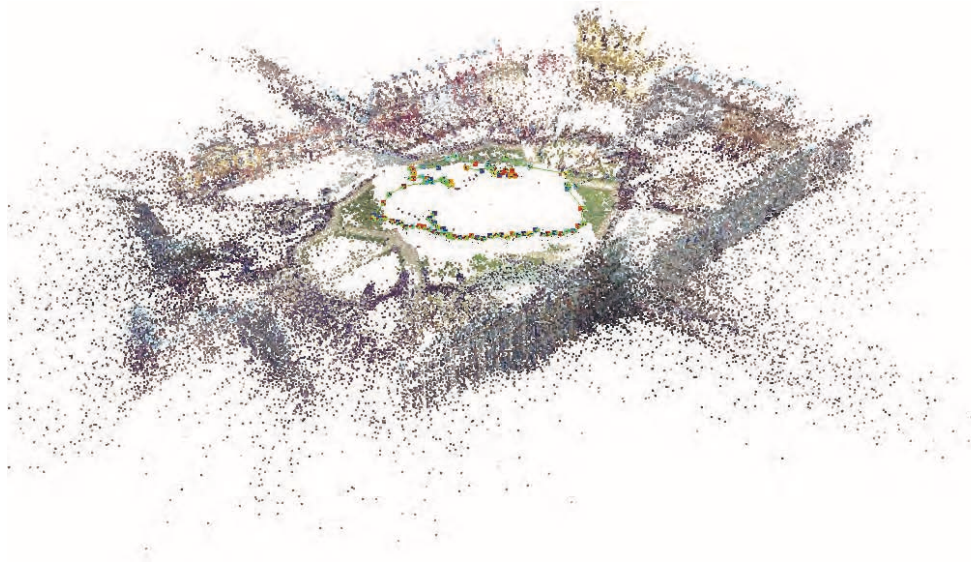
(a)

(b)

Figure 3.39: **Urban experiment – 3D model and camera trajectory after BA.**
(a) view of the 3D model using colored vertices, and the camera trajectory; (b) top view
of the 3D model aligned with an aerial view of Unirii Square from *Google Earth* – the
reconstruction fits the photo accurately.
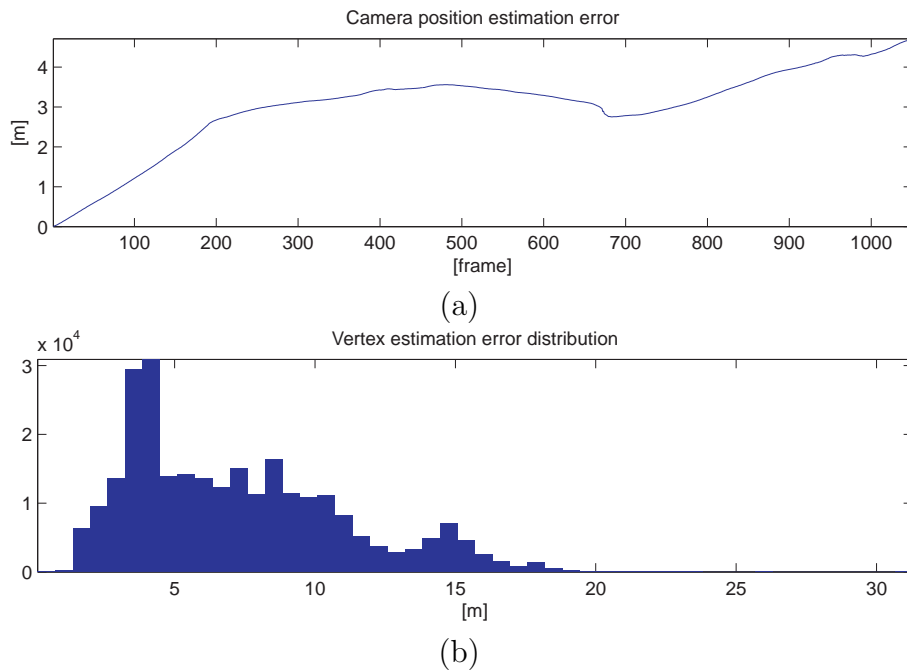
Figure 3.40: **Urban experiment – Estimation errors.** (a) total camera position drift: evolution by frames; (b) vertex estimation error distribution.

pose and vertex position estimation errors by comparing the models before and after the BA (in a similar fashion to the experiment described in Section 3.7.3). Figure 3.40 illustrates the error for both camera and vertex estimations.

3.8 Discussion

In this chapter we presented a novel SfM algorithm for large scale scene modeling. The algorithm generates the scene models sequentially, using a two stage approach. Initially, DPR-SfM creates a seed model corresponding to a small subregion of the scene, using camera motion estimation techniques. In the second stage, the scene model is extended to cover the entire surveyed area. During scene reconstruction, the camera pose is recovered by directly registering camera views with the scene model. This increases the accuracy and robustness of DPR-SfM, allowing it to

successfully cope with situations often found in visual surveys such as occlusions, camera temporary failures, etc. Also, using direct camera pose registration highly increases the flexibility of the DPR-SfM.

Generally, state of the art SfM algorithms require additional sensor information or impose constraints on the image acquisition (*e.g.* minimum camera movement between frames for correct motion estimation). DPR-SfM can be readily applied on image sequences acquired with any type of camera, both still and video, with no constraints on the camera acquisition process. Also, the presented SfM algorithm does not require navigation priors. However, sensor information such as camera pose can be used to decrease the computational cost of the algorithm.

The direct camera pose registration uses a novel dual RANSAC projective/ homography approach which allows the DPR-SfM algorithm to accurately model both planar and non-planar scenes. This is particularly important in underwater and urban scenes, where parts of the scene can have significant parallax while others can be perfectly planar.

Robust estimation methods are also used on vertex position recovery. Experiments show that using a dual layer (camera and model) RANSAC approach increases the stability and accuracy of the method, especially in challenging environments, such as underwater, where image blurring and low contrast decrease the efficiency of feature tracking.

We have also developed an efficient and flexible scene representation. It allows the 3D modeling of large and complex scenes while enabling the parallel use of multiple visual feature extractors/descriptors. In this context, we employed a *kd*-tree scheme for efficient feature matching and camera registration even for large scene models.

Results show that DPR-SfM can efficiently cope with large and complex recon-

structions[8] (*e.g.* Section 3.7.2).

There are several ongoing and future topics that may improve the work presented in this chapter. After camera pose registration, the image patches around features can be warped using camera-to-model transformations. This would reduce the limitations of feature extractor/descriptors of coping with extreme geometric distortions, increasing the efficiency of feature matching. Also, the accuracy of feature localization can be improved by using cross-correlation as a refinement step after feature tracking. Feature-to-model association computational costs can be highly decreased by using GPU-based parallel processing, *e.g.* using *NVIDIA CUDA*.

---

[8]We consider the complexity of the 3D modeling problem to be quantified by the amount of data involved in the reconstruction (*i.e.* number of camera poses and vertices), rather than the metric size of the scene, as the size of the reconstructed area depends only on the camera-to-scene distance and the properties of the camera lenses.

Chapter 4

Online Loop Detection

Vision-based navigation is essentially a *dead reckoning* process. During navigation and map building, vision systems estimate the camera pose relative to either previous poses or an environment map, while they build the map from observations relative to camera poses. All estimations are prone to aliasing, noise, image distortions and numerical errors (see Section 1.3), leading to inaccuracies in both pose and map inferences. While generally small, these inaccuracies build up in time, leading to significant errors over large camera trajectories.

These errors can be reduced by taking advantage of the additional information resulting from *cross-overs*. Cross-overs (or loop-closures) are situations when a camera revisits a region of the scene during a visual survey. If correctly detected, these situations can be exploited in order to establish new constraints, allowing both camera pose and map errors to be decreased (see Figure 4.1), either using offline approaches such as BA [19,85,91,127,143] or online approaches employing gaussian filters such as the popular Kalman Filter [18,36,44,119] or non-parametric methods such as those using particle filters [82,95], etc. In this context, the main open issue is the correct and efficient detection of loop closures.
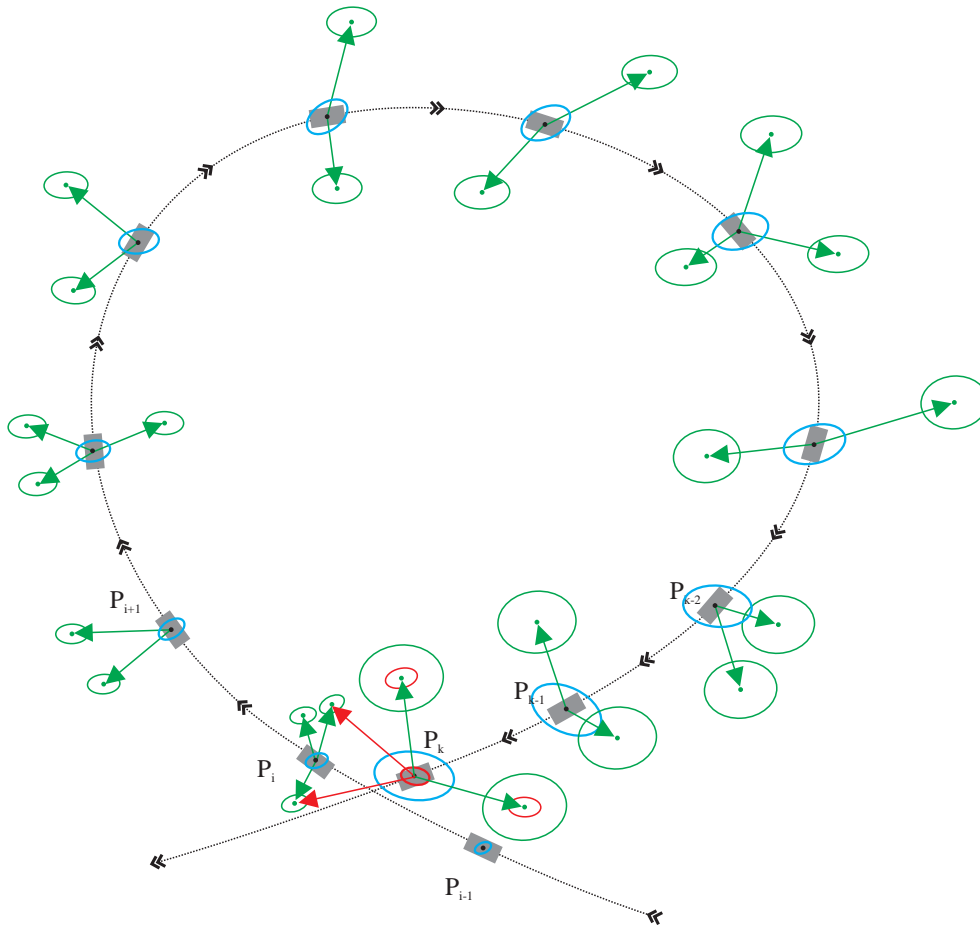
Figure 4.1: **Loop closure detection.** As the the camera moves, there is an increasing uncertainty related to both the camera pose and the environment map. At instant t_k, the camera revisits a region of the scene previously visited at instant t_i. If the visual observations between instants t_k and t_i can be associated, the resulting information can be used not only to reduce the pose and map uncertainties at instant t_k, but it also can be propagated, reducing the uncertainties at prior instants.

Loop closure detection is an inherently complex problem due to the amount of data that needs to be analyzed. As typical image feature extractors yield thousands of features per image, after just a few hundred frames, the resulting map contains tens to hundreds of thousands of features. A brute force loop closure detection, where the current visual observations are compared to the entire map, would be much too computationally expensive, especially for online applications.

As an alternative, the complexity of the loop closure problem can be reduced by narrowing the search to the vicinity of the current camera pose. This is a widely used approach, mainly in the Simultaneous Localization and Mapping (SLAM) community, where the vision system is modeled as a sensor with a known uncertainty. During navigation, an uncertainty is associated to each vehicle pose and the loop closures are detected by matching current observations with the region of the map corresponding to the current uncertainty space [27,28,64,112]. However, an accurate estimation of the vehicle uncertainty is a complex problem and is generally affected by linearization approximations. To counterbalance this shortcoming, assuring the detection of the cross-over, current observations may be compared with a region of the map corresponding to a higher covariance than the estimated one [66,89]. Doing so becomes computationally expensive, especially over large trajectory loops, where the covariance of the camera is high. Moreover, the noise model used for covariance estimation does not account for inaccuracies resulting from obstruction, temporary motion blur, sensor failures, etc. These situations lead to poor vehicle pose estimation, not reflected in the uncertainty estimation, in which case the loop closure may not be detected.

A more robust and computationally efficient alternative is to represent entire images as observations rather than individual image features. In this context, crossovers are detected on the basis of image similarity, drastically decreasing the amount of data that needs to be processed. The reduced computational cost related to such approaches enable brute force cross-over detection, even for large camera trajectories. This allows correct detection of trajectory loops, independent of camera pose and covariance estimation accuracy.

Initial proposals on image similarity cross-over detection use image representations based on a single global descriptor, embodying visual content such as color or
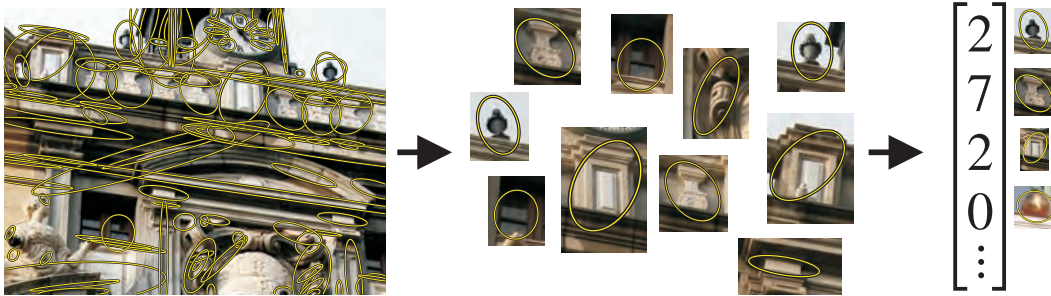
Figure 4.2: **BoW image representation.** Images are represented by histograms of generalized visual features.

texture [13, 70, 72, 117, 141]. Such global descriptors are sensitive to camera viewpoint and illumination changes, decreasing the robustness of the cross-over detection.

The emergence of modern feature extractors and descriptors (see Section 2.1.3) has led to the development of new appearance-based cross-over detection techniques that represent visual content in terms of local image descriptors [1, 2, 22, 23, 146]. Inspired from advances in the fields of object recognition and content-based image retrieval [110, 134, 152], recent examples of such approaches describe images using BoW (see Figure 4.2). BoW image representation employs two stages: (*i*) in the training stage, sets of visual features are grouped or clustered together to generate *visual vocabularies* - collections of generalized visual features or *visual words*; (ii) in the second stage, the images are represented as histograms of visual word occurrences. While discarding the geometric information in images, BoW proved to be very robust methods for detecting visual similarities between images, allowing efficient cross-over detection even in presence of illumination and camera perspective changes, partial occlusions, etc.

The disadvantage of current state of the art BoW methods for loop-closing is the presence of the training stage, which requires the visual features of the scene to be available before the actual survey takes place. Unfortunately, this is an inefficient approach since it requires laborious preparations and involves strong *a priori*

knowledge of the surveyed area.

We propose a novel framework for Online Visual Vocabulary (OVV) building. It requires no user intervention and no *a priori* information about the environment. OVV creates a reduced vocabulary as soon as visual information becomes available during the camera survey. As the camera moves, the vocabulary is constantly updated in order to correctly model the visual information present in the scene.

The vocabulary is built using a novel clustering method that takes into account the global distribution of visual data, increasing its efficiency. Also, we present a novel method for feature-cluster association and image indexing, suited for incremental vocabularies.

OVV is implemented on DPR-SfM (see Chapter 3) with the objective of quantifying visual similarities between frames, allowing online detection of loop-closures. Hereafter, we detail the proposed visual vocabulary building and image indexing techniques.

4.1   Visual Vocabulary

All the state of the art visual vocabulary-based loop-closure algorithms assume an initial training stage [1, 2, 22, 23]. This stage involves pre-acquiring visual features from the scene. These features are then used to build the visual vocabulary by means of some clustering method. Typical offline vocabulary building methods use *K-means*, *K-medians* or fixed-radius clustering algorithms, which require the user to set various parameters such as the number of clusters in the vocabulary. Finding the adequate parameters for an optimum vocabulary is a tedious task which generally involves a trial and error approach. For example, a vocabulary with too many words would not have enough abstraction power to detect similarities between
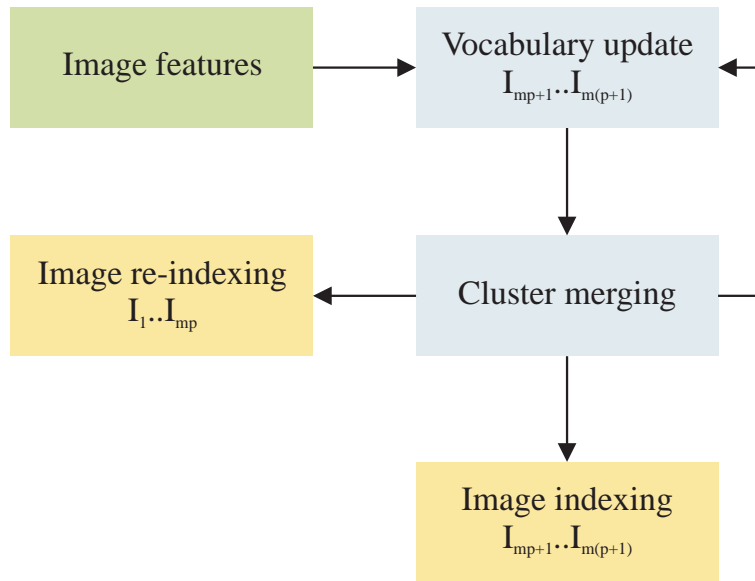
Figure 4.3: **Flowchart of OVV and image indexing.** Every m frames, the vocabulary is updated with new visual features extracted from the last m frames. The complete set of features in the vocabulary is then merged until convergence. The obtained vocabulary is used to index the last m images. Also, the previously indexed frames are re-indexed, to reflect the changes in the vocabulary.

images. In contrast, a vocabulary with too few words would be too confusing and too generalized to be discriminant.

OVV uses a novel incremental visual vocabulary building technique that is both scalable (thus suitable for online applications) and automatic (see Figure 4.3). For this, we use of a modified version of agglomerative clustering [11]. Agglomerative clustering algorithms begin with each element as a separate cluster (called hereafter *elementary clusters*) and merge them using some similarity measurement into successively larger clusters until some criterion is met (*e.g.* minimum number of clusters, maximum cluster radius, etc.).

4.1.1 Vocabulary Building

In our proposal, elementary clusters are generated from visual tracking of scene points (see Chapter 3), with each elementary cluster corresponding to one feature track. The visual vocabulary is built by incrementally merging these clusters. The building process can be summarized in two steps (see Figure 4.4):

- **Vocabulary initialization step.** The vocabulary is initialized with the elementary clusters corresponding to the first m images. Clusters are gradually merged until convergence (the merging criterion is discussed in detail in Section 4.1.3).

- **Vocabulary update step.** As the camera moves, more visual information of the scene becomes available, which needs to be contained in the vocabulary. Therefore, from every block of m images, new elementary clusters are extracted. These clusters are added to the vocabulary and the complete set of clusters is gradually merged until convergence. This step is repeated for each block of m new images.

4.1.2 Cluster Characterization

Each cluster in the vocabulary is defined by its position in the n-dimensional space and its size (radius). This provides complete information about both the cluster distribution and the interaction between clusters. As previously shown, all the input information (for both initialization and update) comes from elementary clusters, so that all the other clusters in the vocabulary are formed by merging the elementary clusters. As the elementary clusters are generated from feature tracking[1], we define them through:

---

[1]Feature tracking provides multiple (noisy) observations of a scene point.
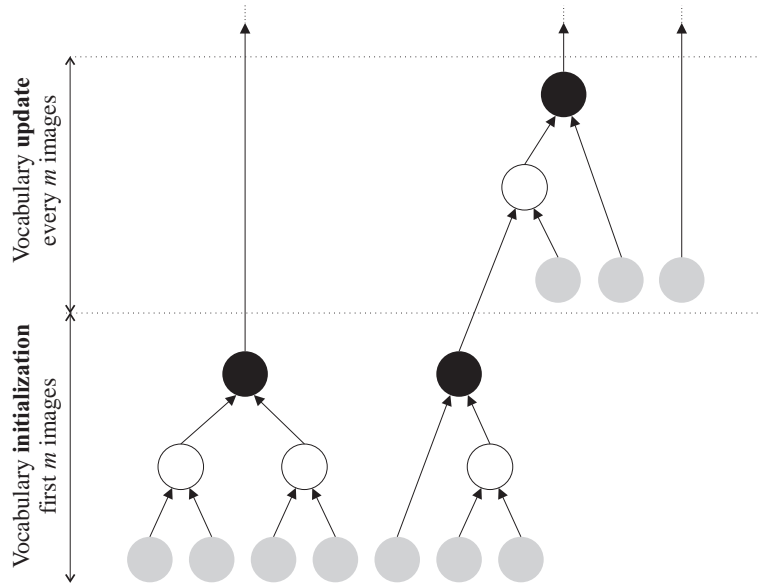
Figure 4.4: **Iterative visual vocabularies.** In the initialization step (bottom part) the vocabulary is populated with elementary clusters (marked in gray), extracted from the first m images. These clusters are merged until convergence. The final clusters of each step are marked in black. In the update step (top part), new elementary clusters obtained from blocks of m images are added to the vocabulary. The complete set of clusters are then merged until convergence.

$$C_k = \frac{\sum f_k^i}{n}$$

$$R_k = \frac{\sum (f_k^i - C_k)(f_k^i - C_k)^{\mathrm{T}}}{n}$$

where C_k is the cluster centroid given by the mean of feature vectors corresponding to scene point k and R_k is the covariance matrix of the observations of point k.

Each cluster merging involves the joining of two clusters (see Figure 4.4). The parameters (C, R) of the newly generated cluster are obtained directly from the merging clusters, without the need of recomputing them from the original data. This saves both computational time and memory, especially in the case of large clusters. The position and size of the new cluster are given by [68]:

$$C_{ab} = \frac{n_a C_a + n_b C_b}{n_a + n_b}$$

$$R_{ab} = \frac{n_a - 1}{n_a + n_b - 1} R_a + \frac{n_b - 1}{n_a + n_b - 1} R_b$$
$$+ \frac{n_b \cdot n_a}{(n_a + n_b)(n_a + n_b - 1)} [(C_a - C_b)(C_a - C_b)^{\mathrm{T}}]$$

where C_a and C_b are the centroids of the merging clusters, having n_a and n_b elements respectively.

4.1.3 Cluster Merging

Generally, clustering algorithms use some similarity measurement to decide which data should be grouped into clusters. Often, similarity measurements are represented by distances in the n-dimensional data space, such as: Euclidean distance, Manhattan distance [69], Chebyshev norm [52], Mahalanobis distance [86], vector angle, etc. These clustering criteria analyze the data only locally and can be suboptimal, especially in high-dimensional, cluttered spaces such as those used for visual feature representation.

We propose a novel clustering method that takes into account the global distribution of data, increasing both the distance between clusters and their compactness. This is crucial, as the efficiency of visual vocabularies is determined by two properties: (i) *repetitiveness* – similar image features should be associated with the same cluster and (ii) *discriminative power* – dissimilar image features have to be associated with different clusters.

The proposed method, based on Fisher's linear discriminant [34] [90], clusters

the data in order to maximize the following objective function:

$$Q = \frac{tr(S_B)}{tr(S_W)}$$

where $tr()$ is the trace operator, S_B represents the *between clusters scatter matrix* and S_W represents the *within clusters scatter matrix* given by:

$$S_B = \frac{1}{N} \sum n_k (C - C_k)(C - C_k)^{\mathrm{T}}$$

$$S_W = \frac{1}{N} \sum n_k R_k$$

where C is the global centroid of the data, N represents the total number of data elements and n_k is the number of data elements contained in cluster k.

Practically, the merging takes place in two steps:

1. For each cluster, we search for merging candidates in its neighborhood (in the Euclidean sense), using a k-dimensional tree (kd-tree) approach [4].

2. For each possible merging pair of clusters, we compute the objective function Q' that would be obtained if the two clusters were merged. If there is an increase in the value of the objective function, then two clusters are merged and Sb, Sw are updated accordingly[2].

Each merging step changes the distribution of data in the vocabulary, requiring the re-computation of both S_B and S_W. As a direct re-computation would be very costly, we propose an incremental update scheme:

---

[2]In practice, we first compute the gain in Q for each possible merging pair, creating a list from the highest to the lowest gain. The clusters are merged following the order on the list, making the merging step independent of the order in which the clusters are analyzed.

$$S'_B = S_B + \frac{n_a + n_b}{N}(C - C_{ab})(C - C_{ab})^\mathrm{T}$$
$$- \frac{n_a}{N}(C - C_a)(C - C_a)^\mathrm{T} - \frac{n_b}{N}(C - C_b)(C - C_b)^\mathrm{T}$$

$$S'_W = S_W + \frac{n_a + n_b}{N}(R_{ab}) - \frac{n_a}{N}(R_a) - \frac{n_b}{N}(R_b)$$

where S'_B and S'_W are the updates of S_B and S_W, respectively; C_{ab} and R_{ab} are the centroid and covariance matrix of the merged cluster.

4.1.4 Convergence Criterion

The two steps shown in Section 4.1.3 are repeated, gradually merging clusters, until no more merges are possible (that would increase the value of the objective function Q). In this way, the repetitiveness and discriminative power of the resulting vocabulary are maximized. Moreover, using a natural convergence criterion, the process eliminates the need of user-set parameters such as cluster radius or number of clusters, specific to other vocabulary building algorithms.

4.1.5 New Clusters

During the vocabulary update step, new elementary clusters are added, containing new visual features. For each newly added elementary cluster ζ_e, S_B and S_W have to be updated accordingly. Similar to the merging step, we avoid recalculating the scatter matrices by proposing a novel update method.

The update of S_W simply involves the covariance matrix R_e of ζ_e, weighted by its number of elements n_e[3]:

---

[3]The number of elements in an elementary cluster corresponds to feature track length.

$$S'_W = \frac{NS_W + R_e}{N + n_e}$$

Adding any new cluster in the vocabulary affects the global data centroid C. The new centroid C' is obtained from:

$$C' = \frac{CN + C_e n_e}{N + n_e}$$

Taking into account the changes in C, S_B is updated using:

$$S'_B = \frac{N}{N + n_e}(S_B + \delta_C^{\mathrm{T}}\delta_C - V^{\mathrm{T}}\delta_C - \delta_C^{\mathrm{T}}V)$$
$$- \frac{n_e}{N + n_e}(C_e - C')^{\mathrm{T}}(C_e - C')$$

where $\delta_C = C' - C$, V is the weighted sum of differences between each newly added cluster centroid and global data centroid. V is obtained incrementally by using:

$$V' = \frac{NV + N\delta_C + n_e(C_e - C')}{N + n_e}$$

4.1.6 Linear Discriminant Analysis

Using the cluster information contained in the visual vocabulary, we aim to find a data transformation that would maximize cluster separability and would allow us to reduce the dimensionality of the data, thus increasing the speed of both vocabulary building and image indexing. For this, we consider maximizing the following Linear Discriminant Analysis (LDA) objective function [34] [90] [25]:

$$J(w) = \frac{w^{\mathrm{T}} S_B w}{w^{\mathrm{T}} S_W w}$$

where w is a vector determining the maximum cluster separability direction. Formulating the maximization of $J(w)$ as a generalized eigenvalue problem, we obtain a data transformation G from the eigenvectors corresponding to w. By selecting m columns of G corresponding to the highest values of w, we reduce the dimensionality of the data to s dimensions.

4.1.7 Vocabulary Update Criterion

In Section 4.1.1, in order to simplify the explanation, we state that the vocabulary is updated each m images. In practice, the vocabulary is updated adaptively, rather than at fixed intervals, so that it constantly represents an accurate model of the visual content in images.

During image indexing, features are associated with clusters in the vocabulary (see Section 4.2.1). For each association of a feature f_l with a cluster ζ_k we check if the feature falls within the cluster, using:

$$|f_l - C_k| \le 3\sigma_k \tag{4.1}$$

where σ_k is the standard deviation of cluster ζ_k. In eq. (4.1), the absolute value $|\cdot|$ and the comparison are to be understood componentwise.

At each vocabulary update step, we index images until the percentage of features falling within the radius of their associated clusters drops below 90%. At this point, we consider that the vocabulary does not model correctly the visual content in the images, hence we update the vocabulary.

4.2 Image Indexing

Inspired from text document indexing [73], BoW techniques use visual vocabularies to represent the images by associating the features present in each of the images with the visual words in the vocabulary [21, 110, 152]. The result is a histogram representing the number of occurrences of each visual word in the image. The similarity between images is calculated by comparing these histograms.

When detecting cross-overs, it is paramount that image features be correctly associated with clusters, even in presence of illumination and perspective changes. We partially achieve this by maximizing the repetitiveness and discriminative power of the vocabulary (see Section 4.1.3). However, in the context of online vocabularies, we need to define a third property: *stability*. As the vocabulary is constantly updated, the aim is to ensure that similar features are associated with the same clusters at different stages of the vocabulary update. We achieve this property through a novel feature-cluster association technique.

4.2.1 Cluster Association

The association between features and visual words is performed by comparing each feature with all the clusters in the vocabulary. The feature is then associated with the most similar cluster. Most image indexing techniques calculate the similarity between features and clusters using distances in the feature space (see Section 4.1.3). This approach is suitable for image indexing in the case of static vocabularies[4] [134].

As we use an online approach for vocabulary building, such a feature association method would not be stable. In Figure 4.5a, feature f is associated with the closest cluster ζ_b. After the vocabulary is updated, clusters ζ_a and ζ_c are merged, yielding

---

[4]A static vocabulary represents a vocabulary that is calculated before the image indexing stage and does not change throughout it.

ζ_a

ζ_b

f

ζ_c

(a)

ζ_{ac}

ζ_b

f

(b)

ζ_{ac}

ζ_b

f

(c)
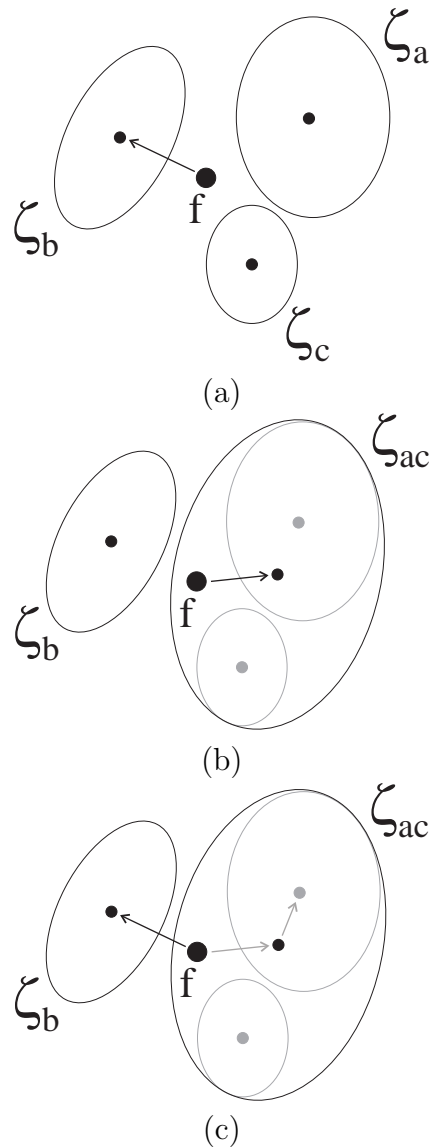
Figure 4.5: **Feature-cluster association.** In (a) the feature f is associated with cluster ζ_b, using feature-to-cluster centroid distance. After the vocabulary update, clusters ζ_a and ζ_c are merged. The centroid of the newly obtained cluster ζ_{ac} is now closer to f. Using a classical approach, feature f would be associated with ζ_{ac} (b). Using hierarchical trees, feature f is correctly associated with cluster ζ_b (c).

a new cluster ζ_{ac} (Figure 4.5b). As the feature f is now closer to the centroid of the new cluster ζ_{ac}, it would be associated to it. In this case, feature f would be associated with different clusters before and after the vocabulary update.

Alternatively, the proposed feature-cluster association technique uses a tree-based approach. The trees are formed during the vocabulary building process. The nodes of the trees represent the clusters while the branches define the cluster hierarchy. The roots of the trees correspond to the visual words while the leafs of the trees correspond to the elementary clusters (see Figure 4.4).

During the feature-cluster association, the trees are visited top-down, calculating the similarity (Euclidean distance) between each feature and the tree nodes (see Figure 4.5c). In order to speed up the association process, we visit only those trees corresponding to visual words in the vicinity of the feature. For this, we calculate the distance between the feature and the visual words, and select the trees where:

$$D(f, \zeta_k) \leq \tau D_m \qquad (4.2)$$

with $D(f, \zeta_k)$ being the distance between feature f and ζ_k; D_m is the minimum distance between the feature f and the visual words and τ is a user-defined constant ($\tau \geq 1$).

The selected trees are visited in parallel (see Figure 4.6). For efficiency purposes, a stopping criterion similar to eq. (4.2) is used, hence avoiding visiting branches that contain nodes that are not close to f. The feature is finally associated to the visual word corresponding to the most similar leaf.

4.2.2   Image Re-indexing

During the update process, the configuration of the vocabulary changes. Consequently, the similarity between images indexed at different update stages cannot be computed. Also, indexing the images after each vocabulary update is not a viable solution due to the high computational cost.
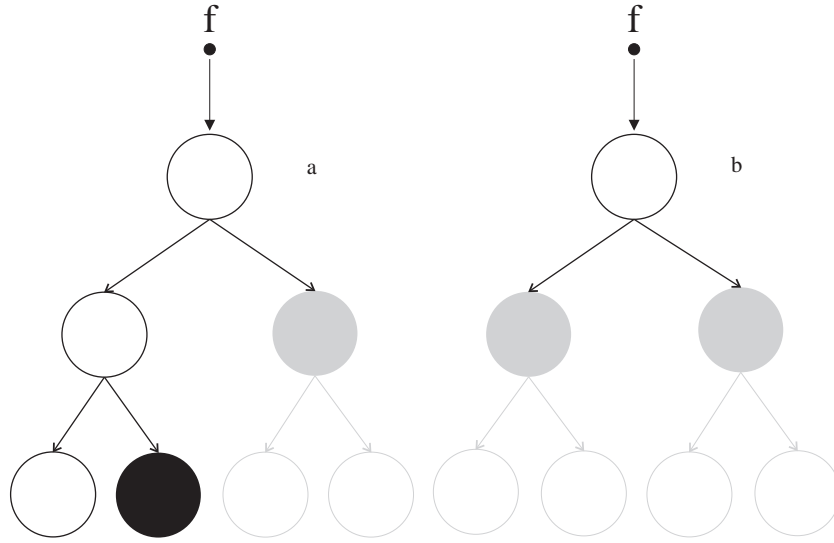
Figure 4.6: **Top-down feature-cluster association.** The trees are visited by comparing each node with the feature. If a node is too dissimilar to the feature (marked in light grey), the rest of the tree corresponding to the node is not visited. The feature is associated with ζ_a due to the highest similarity between f and the leaf marked in black.

We propose a novel solution to this shortcoming by defining a transformation $^p\Gamma_{p-1}$ that embodies the changes in the vocabulary during the update stage. This transformation allows a fast re-indexing of the images (hence eliminating the need of repeated image indexing):

$$\widetilde{W}_I^p = {}^p\Gamma_{p-1}W_I^{p-1}$$

where W_I^{p-1} is the indexing of image I at vocabulary update stage $p-1$ and $\widetilde{W}_I^p$ is an approximation of the image indexing I at vocabulary update stage p.

During update, the visual vocabulary undergoes the following changes:

1. Adding of elementary clusters. If these new clusters are not absorbed into already existing clusters, they contain new visual information. In this case, it is very unlikely that any feature from any image before the update would have been associated to them. Therefore, the bins $\widetilde{W}_I^k$ are initialized to 0.

2. Cluster merging. In the case that two (or more) clusters merge, any feature previously associated with any of these clusters would be associated to the newly formed cluster. In this case, the number of occurrences associated with the new cluster is the sum of occurrences of the merging clusters.

To reflect these changes, ${}^p\Gamma_{p-1}$ has to initialize the histogram elements corresponding to newly added clusters and sum the elements corresponding to merging clusters. For a better understanding, let us consider the following example: at stage $p-1$ the indexing of image I yields $[w_1\ w_2\ w_3]^{\mathrm{T}}$ corresponding to the visual vocabulary containing $(\zeta_1,\ \zeta_2,\ \zeta_3)$; during the vocabulary update, clusters $\zeta_1,\ \zeta_2$ merge into ζ_{12} and a new cluster ζ_4 is added. In this case, the transformation ${}^p\Gamma_{p-1}$ becomes:

$$
\begin{bmatrix} w_{12} \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}
$$

4.2.3 Image Similarity

The visual resemblance between images is quantified by measuring the similarity of their corresponding histograms[5]. As the histograms are represented by vectors containing the occurrences of the visual words, we calculate their similarity using the normalized scalar product (cosine of the angle between vectors) [134]:

$$
s_{rq} = \frac{W_r^{\mathrm{T}} W_q}{\|W_r\|_2 \cdot \|W_q\|_2} \tag{4.3}
$$

where s_{rq} is the similarity score between images I_r and I_q, W_r and W_q are the histograms of the images; $\|W\|_2 = \sqrt{W^{\mathrm{T}}W}$ is the L_2 norm of vector W.

---

[5]Here, the term "histogram" of image I refers to a vector embodying the number of occurrences of each visual word in I.

In eq. (4.3), the similarity score is highly influenced by histogram elements corresponding to visual words with high occurrence. Generally, these words represent visual features commonly found in the images, thus having low discriminative power. In order to counterbalance this shortcoming, the elements of the histograms are weighted using *term frequency-inverse document frequency* [5]:

$$\overline{w}_k = \frac{n_{ki}}{o_i} \log \frac{m_p}{O_k}$$

where n_{ki} is the number of occurrences of word k in image I_i, o_i is the total number of words in I_i, O_k is the total number of images containing word k and m_p is the total number of indexed images.

4.2.4 Cross-over detection

During online navigation and mapping, an increased value of s_{rq} between the current image and any previous one indicates a high probability that the two images correspond to the same region of the scene (*i.e.* loop closing). This information can be used for both introducing new constraints in the mapping model and reducing the navigation-related uncertainties.

Noise, low contrast and especially motion blur may sometimes decrease the efficiency of image indexing, leading to false positives when detecting cross-overs. Assuming a smooth camera motion, there must be a certain degree of overlap between neighboring frames in the image sequence. In other words, if an image I_q has a high degree of visual similarity with some other image I_r, the neighbors of I_r must also be (at least partially) visually similar to I_q.

Seeing the similarity between image I_q and all the images in the sequence as a time-dependent measurement, we employ individual compatibility test [20] in order

to reject false positives.

4.3 Experimental Results

This section discusses some experiments designed to evaluate the two novelties: (i) the incremental vocabulary building and (ii) the image indexing based on hierarchical trees. The efficiency and accuracy of the OVV algorithm is tested using a data association and a comparison with ground truth. The obtained results are compared with with a state-of-the-art offline visual vocabulary building technique using K-means.

In the first experiment, we provide a detailed analysis of the influence of LDA dimension reduction s and relative threshold τ on the accuracy and computational cost of OVV. The two parameters are user-set and provide an increase in the computational efficiency of OVV at the expense of a slight decrease in the accuracy of the results. Experiments show that these two parameters are not data sensitive, so that for the rest of the experiments we used a fixed setting that provides a good balance between speed and accuracy.

The remaining experiments focused on the detection of loop-closing in various environments, including underwater and outdoor scenes.

4.3.1 Laboratory Experiment

The first experiment was carried out in the laboratory, using a relatively flat scene that contains books, boxes and magazines. The scene composition was chosen to be visually complex, combining uniform (low texture) regions, natural scenes, geometric figures and abstract drawings.

The test sequence consists of 215 images of 640×480 pixels, acquired using a
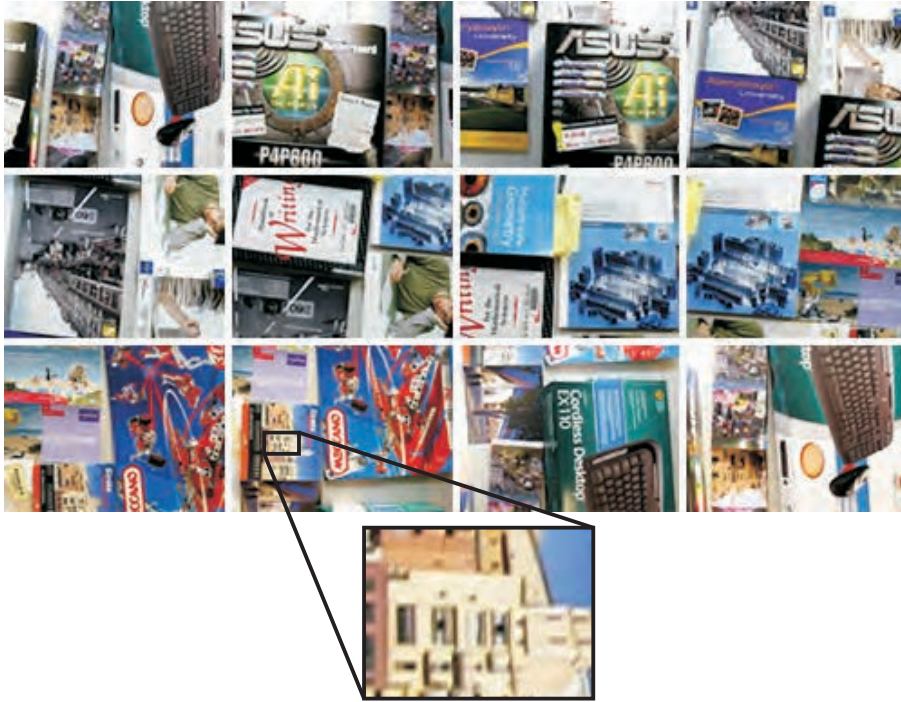
Figure 4.7: **Laboratory experiment – Input image sequence.** Sample images from the input sequence. The first and the last images have a partial overlap. The blow-up shows the motion blur and defocusing.

Canon G9 compact camera (see Figure 4.7 for some snapshots of the sequence). The images contain a certain amount of motion blur and defocusing, allowing us to test the robustness of the visual vocabulary.

The camera is moved while in a down-looking orientation, describing a loop trajectory with a partial overlap between the first and the last images. Figure 4.8 illustrates the resulting scene model and camera trajectory, after applying DPR-SfM on the image sequence. The detection and extraction of features was carried out using SURF, yielding $\sim$37,000 tracks corresponding to the 3D vertices. Each image feature is represented using a 64-element normalized vector as described in Section 2.1.3.

The vocabulary was initialized using the visual information extracted from the
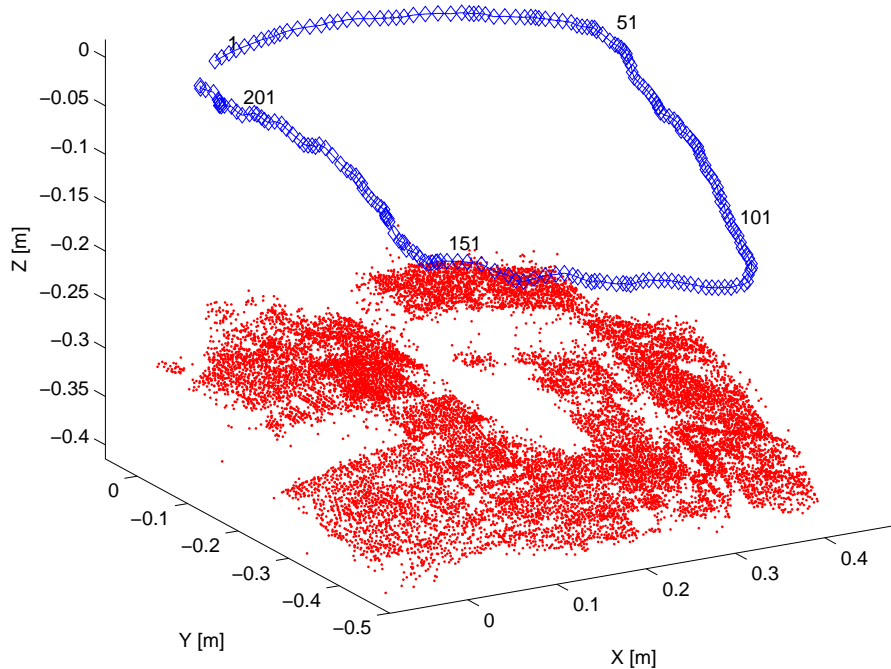
Figure 4.8: **Laboratory experiment – 3D model and camera trajectory.** The scene model contains ∼37,000 vertices (marked in green). The camera describes a loop trajectory (marked in blue) with an overlap between the first and last images.

first 20 images. During sequence analysis there are 10 vocabulary updates, resulting in a final vocabulary containing 3,485 visual words. Figure 4.9 illustrates the evolution of the vocabulary. Towards the end of the sequence, the growth rate of the vocabulary decreases, as there is little new visual information contained in the last images. The instants when the vocabulary was updated can be better observed in Figure 4.10, along with the computational times of vocabulary building and frame indexing.

OVV can be adjusted using two user-set parameters. Unlike other visual vocabulary algorithms, where various parameters need to be adjusted for each dataset in order to obtain accurate results, the user parameters in OVV are data independent. The first parameter s determines the number of LDA dimensions used for feature
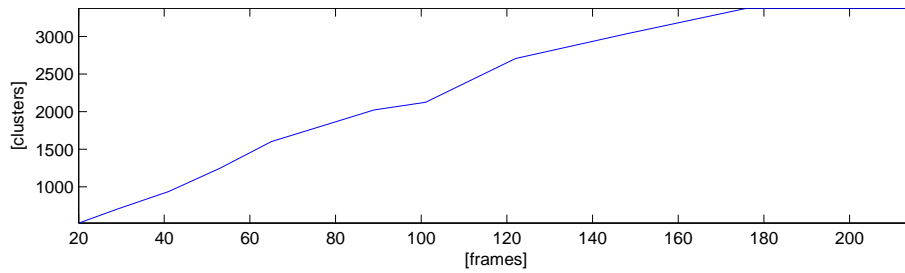
Figure 4.9: **Laboratory experiment – Vocabulary size evolution.** The vocabulary was initialized using the first 20 frames. After 10 updates, the final vocabulary contains $\simeq 3,400$ visual words.
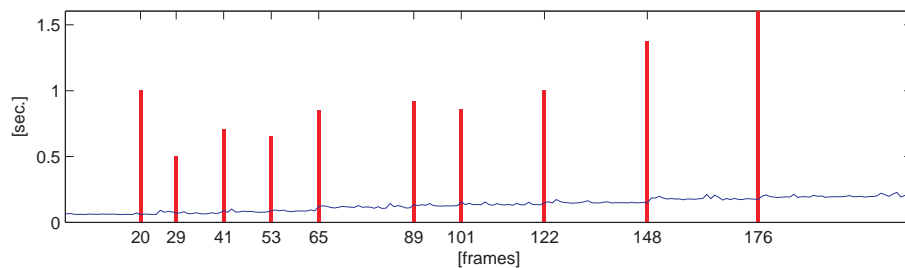
Figure 4.10: **Laboratory experiment – Computational times.** The vocabulary building time (red bars) and the frame indexing time (blue line) evolution vs. the number of frames. A total of 10 vocabulary updates took place with an average of 0.9 sec./update. The average indexing time was 0.13 sec./frame.

clustering and image indexing. A lower number of dimensions decreases both the clustering and frame indexing times, while slightly decreasing the accuracy of the results. The second parameter τ determines the amount of tree branches that are simultaneously visited during frame indexing. A lower value of this parameter decreases the computational time related to frame indexing, while slightly decreasing the accuracy of frame indexing.

We designed two tests that assess the efficiency of the OVV and influence of the parameters on the accuracy of the results. In the first test, we use a direct data association experiment. For each image feature, we associate an elementary cluster that corresponds to the smallest Euclidean distance in the feature space. The image features are then "sent down" the indexing trees. If the image features end

up at the leaf corresponding to the associated elementary cluster, it is considered a hit and a miss otherwise. A high ratio of hits denotes a stable vocabulary and feature labeling which is crucial for accurate results, especially in the case of dynamic vocabularies used in OVV, as we show in Section 4.2.1. The second test is aimed at evaluating the accuracy of the visual similarity in representing the actual overlap between images. For this, we compare the similarity matrix (see Figure 4.11) with the overlap ground truth matrix. The overlap matrix was obtained by exhaustively calculating the projective homography between each two images from the sequence. From the homographies, we obtained the overlap ratio between all images in the sequence. We represent the accuracy of the frame similarity matrix by the average of absolute differences between the similarity and the overlap matrices.

The two tests were repeated for different values of s and τ. Table 4.1 shows the accuracy and execution time versus LDA dimensionality reduction. The results clearly show the advantages of LDA. Reducing the dimensionality of data to 24 we obtain more accurate results and greatly increased computational efficiency with respect to full 64 dimensions when no LDA is used. However, decreasing the data dimensionality further diminishes the discriminative power of the vocabulary. This increases the similarity score between non-overlapping frames, reducing the overall accuracy of the result. Additional tests on other datasets show that $s = 24$ provides the ideal tradeoff between accuracy and computational efficiency.

Augmenting the value of τ (see Table 4.2), increases the number of tree branches that are simultaneously visited during image indexing. As expected, this results in increased accuracy at the expense of higher computational costs. Using $\tau = 1.4$ offers the ideal trade-off between indexing speed and accuracy, as using higher values increases the related computational cost with no real gain in accuracy. As in the previous case, $\tau = 1.4$ proved to be the ideal value for all the datasets we have

Table 4.1: **Laboratory experiment – OVV accuracy and execution times vs. LDA dim. reduction.** As the number of dimensions decreases, total vocabulary building time (2*nd* column) and average frame indexing time (3*rd* column) are reduced, also decreasing the hit percentage (4*th* column) and increasing the visual similarity average error wrt. image overlap (5*th* column). The first row shows the results without using LDA.

LDA Dim. s	Vocab. Time [sec.]	Index. Time [sec./fr.]	Hits [%]	Error
no LDA	11.9	0.24	99.1	0.0714
64	10.9	0.24	99.6	0.0668
48	9.9	0.17	99.5	0.0674
32	8.6	0.13	99.3	0.0682
24	8.3	0.11	99.2	0.0695
16	6.5	0.08	98.8	0.0793
8	5.7	0.05	98.0	0.1216

Table 4.2: **Laboratory experiment – OVV accuracy and execution times vs. τ.** Using a higher τ, the average frame indexing time (2*nd* column) increases as more tree branches are visited simultaneously, improving the hit percentage (3*rd* column) and decreasing the visual similarity average error with respect to image overlap (4*th* column).

τ	Index. Time [sec./fr.]	Hits [%]	Error
1.0	0.10	95.0	0.0738
1.1	0.11	97.0	0.0731
1.2	0.11	98.4	0.0715
1.3	0.12	98.9	0.0701
1.4	0.13	99.2	0.0695
1.5	0.15	99.2	0.0693

tested.

In order to provide the reader with an objective evaluation, we compare the results obtained using OVV with an off the shelf BoW algorithm based on K-means clustering. We have chosen this approach for comparison, due to its popularity in computer vision and visual SLAM community. We set the number of words in the vocabulary to be the same as the number of words in the OVV in its final form – 3,485 words. Due to the random nature of K-means clustering, we ran the clustering algorithm 20 times and chose the vocabulary corresponding to the maximum cluster compactness. The average computational time was 8.9 sec./run. The frames were
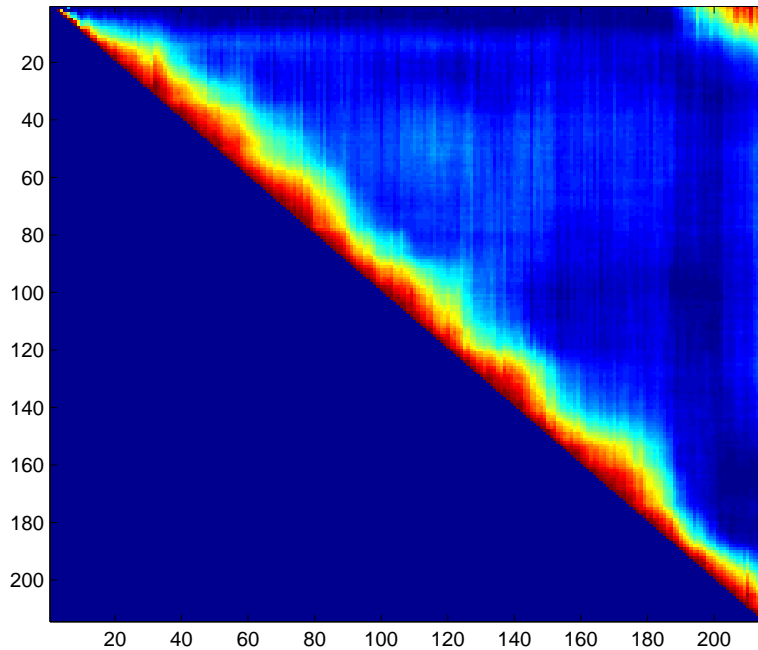
Figure 4.11: **Laboratory experiment – Image similarity matrix.** High values close to the main diagonal correspond to the similarity of the images with their close neighbors. The bright region in the upper-right corner of the matrix denotes an overlap between frames in the beginning and the end of the sequence.

indexed using minimum Euclidean distance feature-cluster association with an average computational time of 0.3 sec./frame, resulting in an average error between the similarity matrix and frame overlap of 0.0985. This shows that, while incremental, OVV provides better accuracy than offline K-means algorithm.

The last part of the laboratory experiment consisted in the detection of the loop closure. For this, we build the image similarity matrix, shown in Figure 4.11. The similarity matrix illustrates a high degree of visual resemblance between the first images and the last images of the sequence (upper-right corner).

Figure 4.12 illustrates the similarity score between I_{215} and all the images in the sequence. The peak at image I_1 indicates a high visual similarity between frames I_1 and I_{215}, corresponding to a cross-over (see Figure 4.13). The visual similarity score between the two images is 0.8, accurately representing the ground truth overlapping

Figure 4.12: **Laboratory experiment – Image similarity for query image I_{215}.** The plot shows the similarity between frame I_{215} and all the previous frames. The peak on the far right of the plot corresponds to time-adjacent frames. The peak corresponding to I_1 indicates an overlap.

Figure 4.13: **Laboratory experiment – Loop detection.** The first (left) and last (right) images of the sequence correspond to the same region of the scene, determining a loop closure.

ratio of 0.82.

4.3.2 Coral Reef Experiment

This experiment is aimed at testing the efficiency of the OVV method in describing natural, unstructured environments for underwater robot navigation and mapping. The image sequence, acquired using a ROV near the Bahamas by the UoM, is comprised by 235 frames of 720×530 pixels. The surveyed scene contains a coral formation and its surroundings, combining rich texture areas (vegetation and rock

Figure 4.14: **Reef experiment – 3D model and camera trajectory.** The scene model contains $\simeq 62,000$ vertices. The trajectory of the camera has several cross-overs.

formations) and uniform areas (sandy regions).

We applied DPR-SfM on the sequence using SURF features. Figure 4.14 illustrates the 3D reconstruction and the camera trajectory estimation. The resulting $\simeq 62,000$ SURF feature tracks were used to generate the vocabulary as the scene was being reconstructed. The vocabulary was initialized using the first 20 frames and updated 9 times, containing 4,343 in its final form. Analyzing the vocabulary evolution in Figure 4.15, it can be seen that the vocabulary grows fast at the beginning of the sequence. Towards the end, the vocabulary increase rate slows and the vocabulary update frequency lowers, as there is little unmodeled visual information left in the scene.

After vocabulary building and image indexing, the resulting similarity matrix in Figure 4.16 successfully points out the cross-overs in the camera trajectory. An exemplification of this is provided in Figure 4.17, where a query for frame I_{204} shows two peaks at frames I_{52} and I_{155}, with similarity scores of 0.73 and 0.75 respectively.
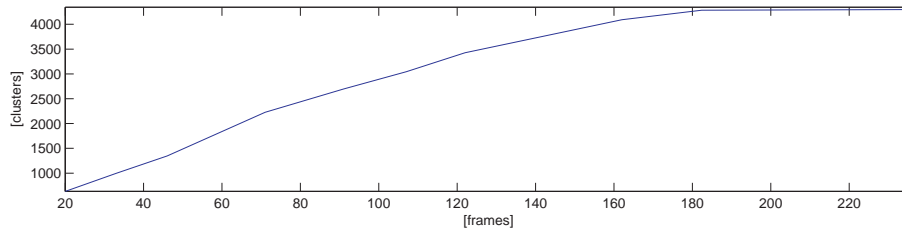
Figure 4.15: **Reef experiment – Vocabulary size evolution.** The vocabulary was initialized using the first 20 frames. After 9 updates, the final vocabulary contains $\simeq 3,400$ visual words.
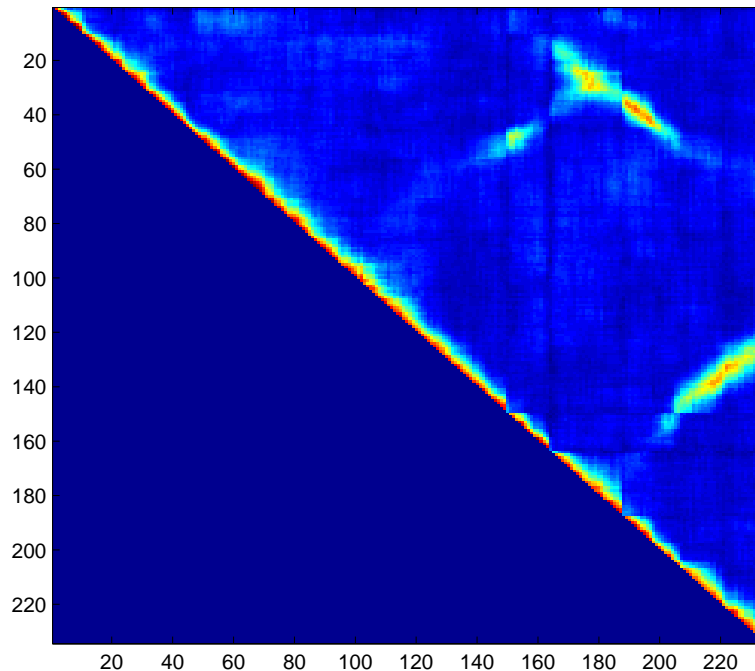
Figure 4.16: **Reef experiment – Image similarity matrix.** The bright regions off the main diagonal correspond to multiple cross-overs.

The estimated overlap ratio between I_{204} and frames I_{52} and I_{155} is 0.78 and 0.8 respectively, showing that the similarity scores closely represent the overlap between images. Figure 4.18 clearly illustrates that the three frames correspond to the same region of the scene.

To quantify the precision of the similarity matrix in approximating the image overlap, we compared it with the overlap ground truth using the average of absolute
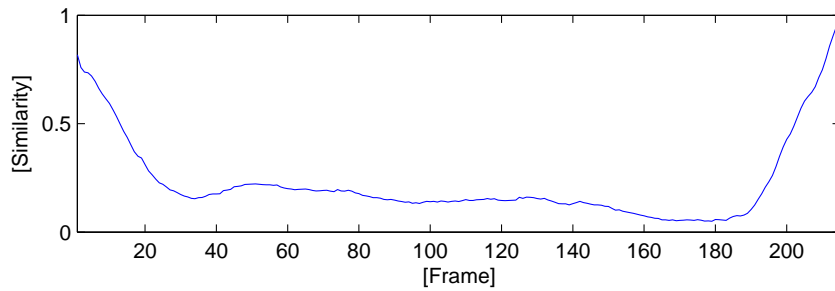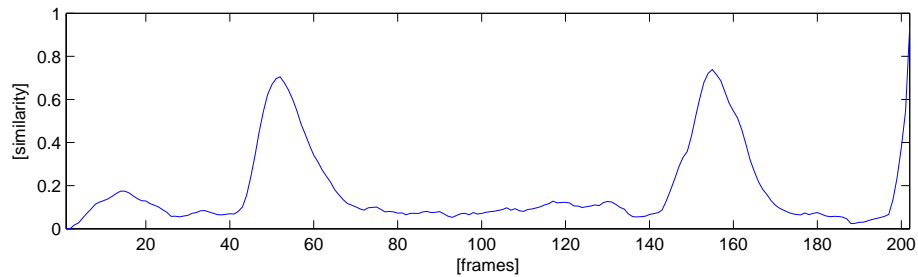
Figure 4.17: **Reef experiment – Image similarity for query image I_{204}.** The plot shows the similarity between frame I_{204} and all the previous frames. The two peaks corresponding to frames I_{52} and I_{155} indicate that all three frames correspond to the same region of the scene.

I_{204}
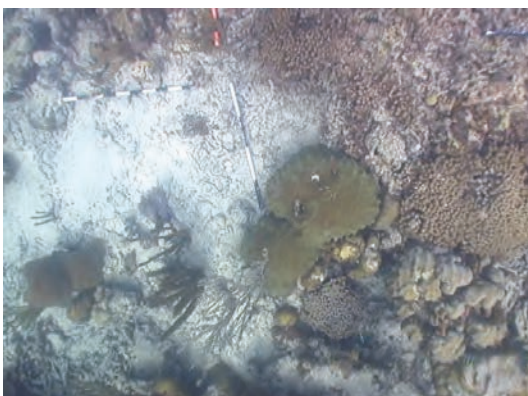
I_{52}
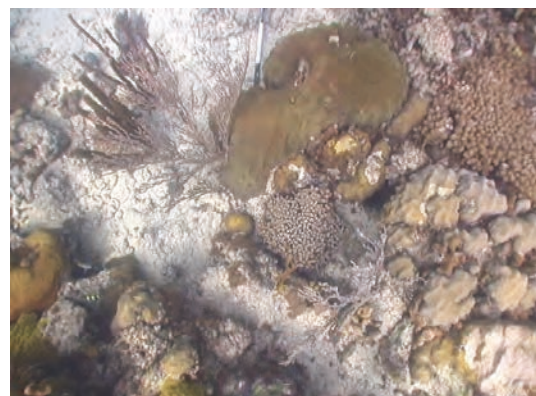
I_{155}

Figure 4.18: **Reef experiment – Cross-over.** Query frame I_{204} and frames I_{52} and I_{155} were successfully determined as corresponding to the same region of the scene, defining a loop closure.

differences. The error was 0.095, higher than in the previous experiment. This is expected, since low contrast and high blurriness in underwater imaging decreases the quality of image features.

We compared the result with K-means vocabulary, using the same number of visual words as in the OVV in its final stage. The average error in case of K-means vocabulary is 0.0978, indicating that OVV yields slightly better results in case of underwater imaging.

4.3.3   Tortugas Experiment

The second underwater experiment presented here was acquired in the Tortugas, Florida Keys, using an ROV of the UoM. The sequence, containing 1,000 frames of 720×530 pixels, depicts a region comprised mainly by rocks and sand. The sequence is characterized by repetitive textures, allowing us to test the OVV algorithms in presence of increased visual aliasing[6].

Figure 4.19 illustrates the estimated 3D model containing 125,850 vertices along with the camera trajectory. The online vocabulary was initialized using the feature tracks in the first 20 frames. During scene reconstruction, the vocabulary went through 15 updates, containing 6,644 visual words in its final form.

After frame indexing, the similarity matrix in Figure 4.20 shows a higher degree of noise due to visual aliasing. This fact is also indicated by an increased average error of 0.14 between the visual similarity matrix and the overlap ground truth. The average in case of K-means is 0.17, showing that OVV is less sensitive to visual aliasing.

The effect of visual aliasing on image similarity estimation is better outlined in

---

[6]The visual aliasing problem corresponds to scenes with poor or repetitive textures and is characterized by the fact that different regions of the scene appear similar to the camera.
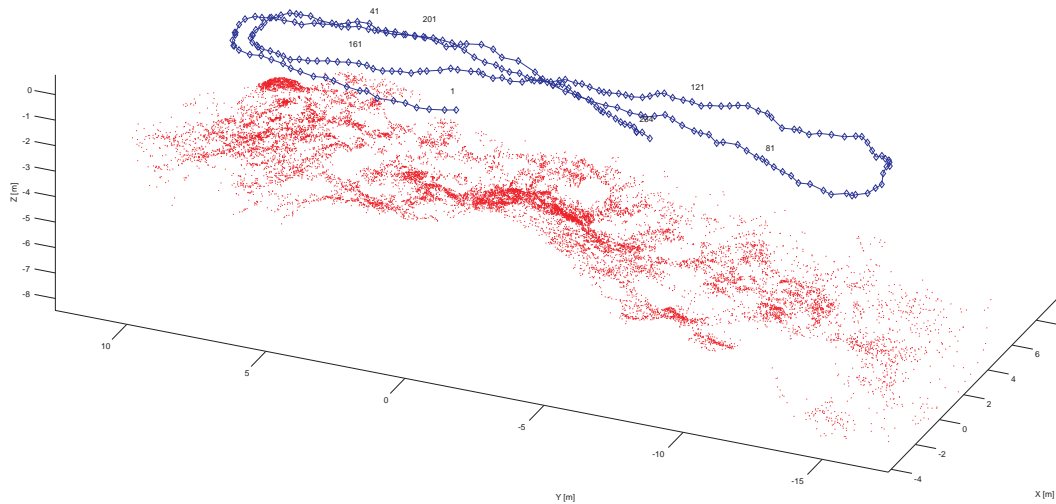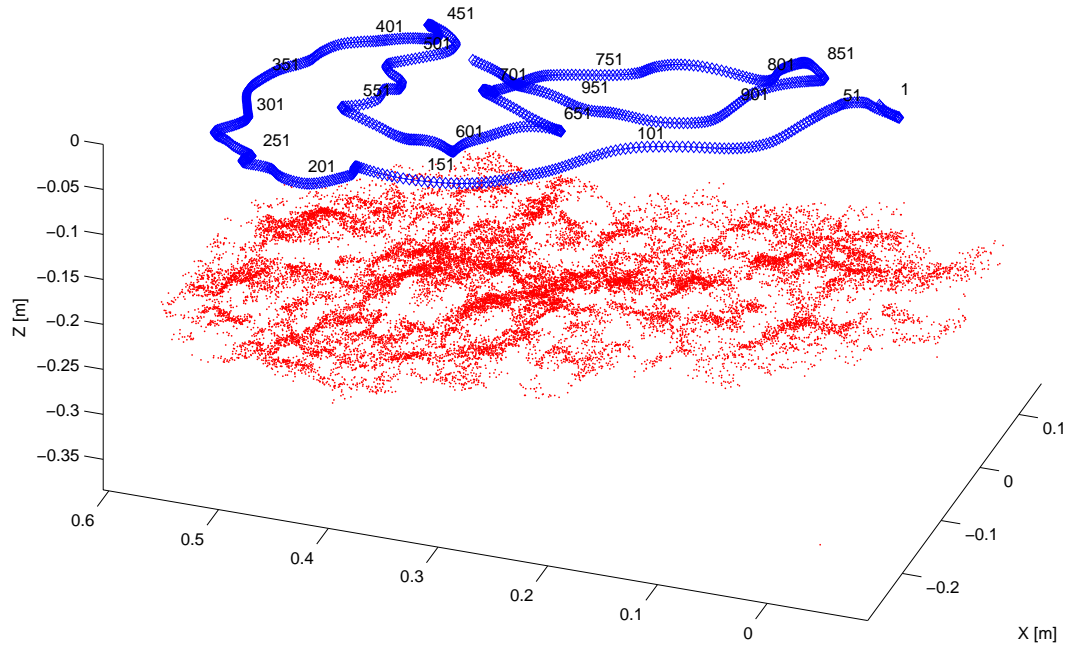
Figure 4.19: **Tortugas experiment – 3D model and camera trajectory.** The scene model shown in red contains $\simeq 126,000$ vertices. The trajectory of the camera (blue) presents some partial overlaps.

Figure 4.21, where non-overlapping frames show a certain degree of visual similarity. While this does not affect the detection of cross-over at frame I_{156}, cross-overs corresponding to frames with small overlap (*e.g.* I_{500}) are not detectable.

Figure 4.22 illustrates some of the pairs of images, corresponding to cross-overs in the camera trajectory.

4.3.4 Urban Experiment

Here, we discuss the loop closure detection for the urban experiment presented in Section 3.7.7. The visual vocabulary was generated and the images were indexed during the scene reconstruction. The final vocabulary contains 7,182 words. The resulting similarity matrix, shown in Figure 4.23, points out a cross-over between
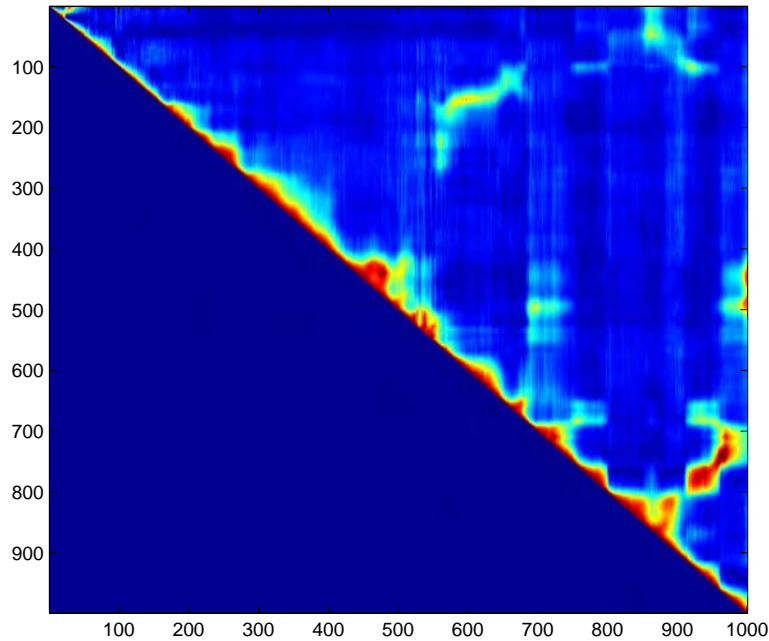
Figure 4.20: **Tortugas experiment – Image similarity matrix.** The slightly brighter background on the upper right side of the similarity matrix denotes a certain amount of visual aliasing. However, overlapping frames are clearly distinguishable.
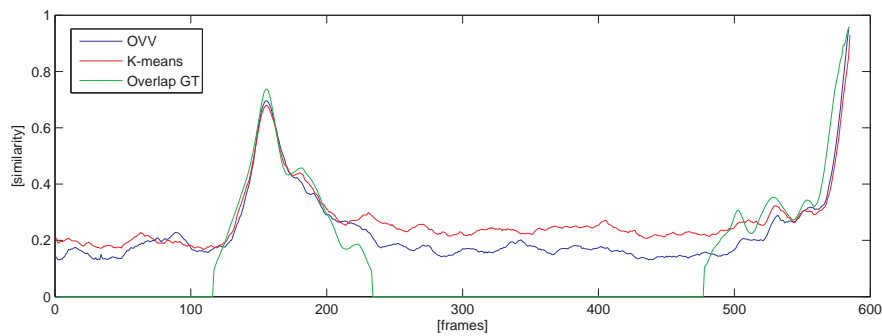
Figure 4.21: **Tortugas experiment – Image similarity for query image** I_{589}**.** A comparison of OVV (blue) and K-means (red) with overlap ground truth (green) points out the effect of visual aliasing: in both cases, non-overlapping images show a certain degree of visual similarity, while OVV proves to be slightly less susceptible to this phenomenon. The cross-over at frame I_{156} is easily detectable.

the first and last frames of the sequence. The situation is exemplified in Figure 4.24, where a query for frame I_{960} denotes a visual similarity of 0.8 with frame I_{45}. Figure 4.25 confirms that the two frames correspond to a loop closure.

I_{589} $\qquad$ I_{156}

I_{966} $\qquad$ I_{738}

I_{999} $\qquad$ I_{445}

I_{841} $\qquad$ I_{45}
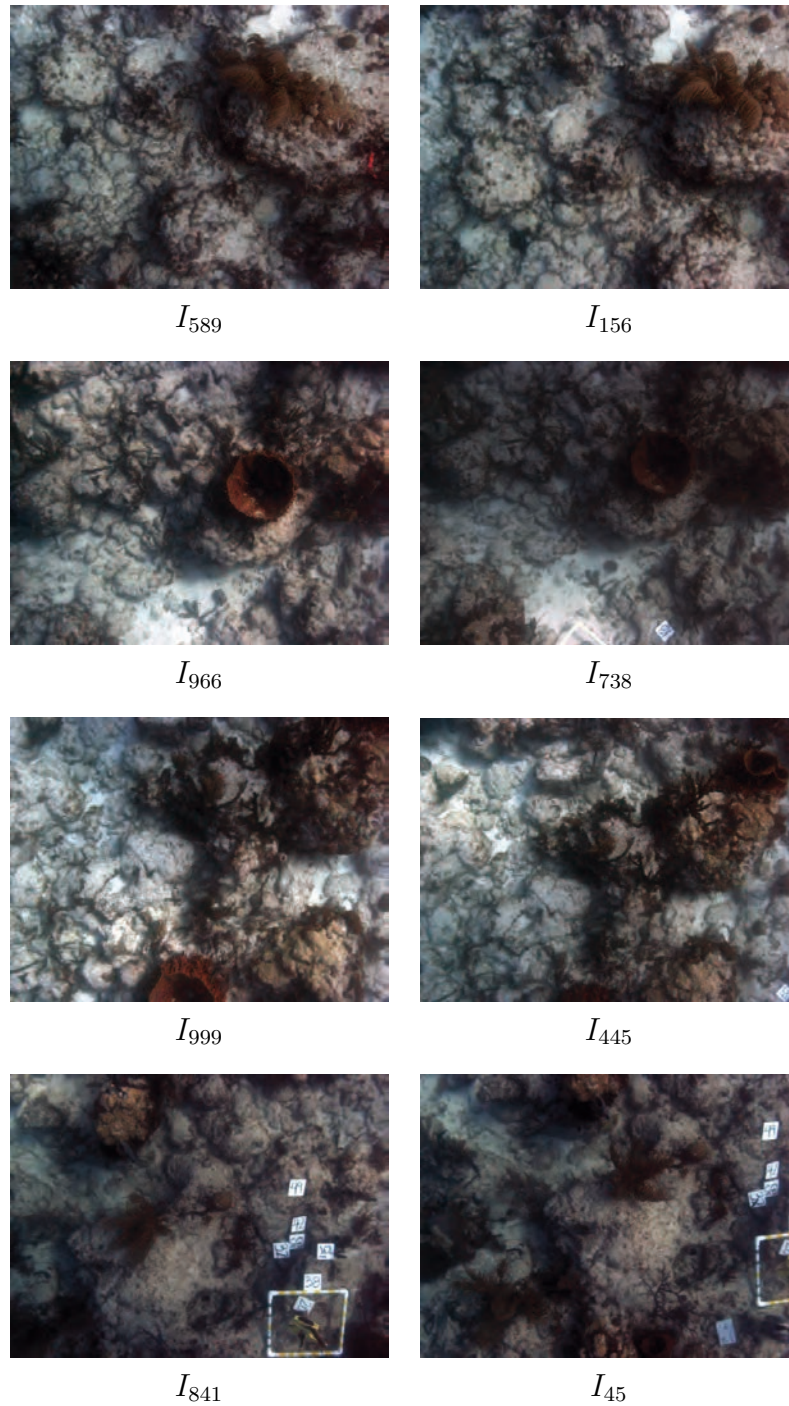
Figure 4.22: **Tortugas experiment – Loop detection.** Pairs of images corresponding to some of the detected cross-overs. Query frames are shown in the left column and their corresponding most similar frames are shown in the right column.
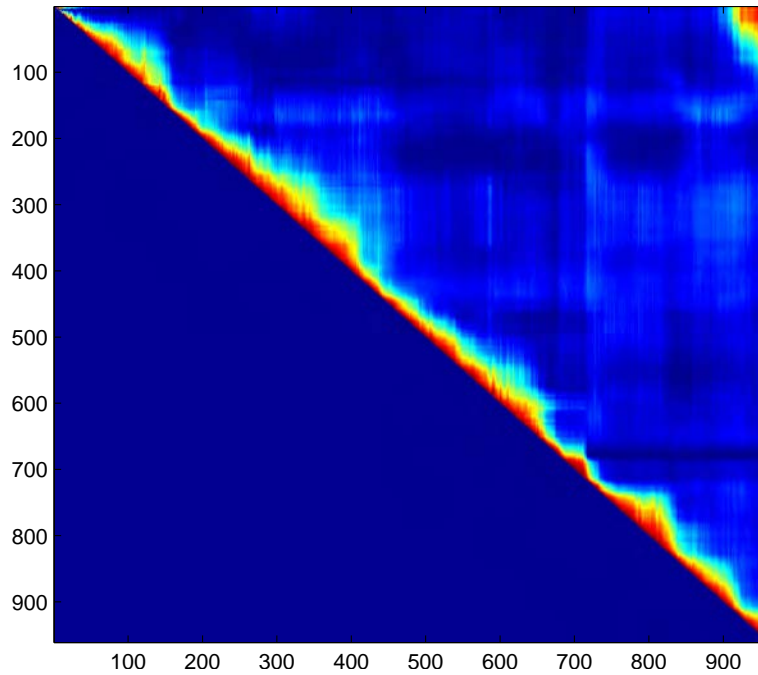
Figure 4.23: **Urban experiment – Image similarity matrix.** The bright region in the upper-right corner of the matrix indicates an overlap between first and last frames of the sequence.
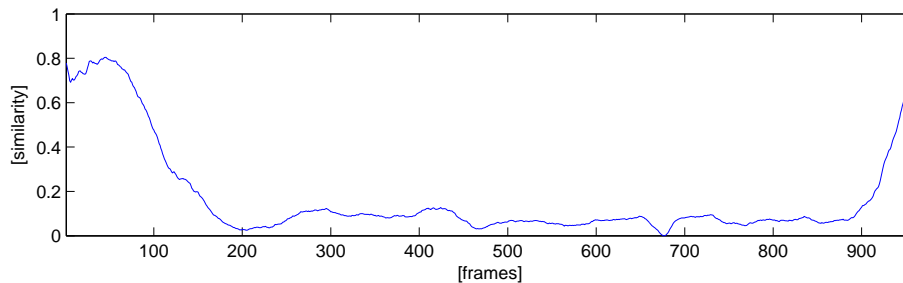
Figure 4.24: **Urban experiment – Image similarity for query image I_{960}.** The plot shows a high degree of visual similarity between frames I_{960} and I_{45}, corresponding to a loop closure.

In the remainder of this section, we present a test that we have carried out in order to assess the capacity of OVV indexing to be extended to other images of the same location. For this, we selected a set of photos from *Google Images* [48] depicting the Unirii Square, taken at different times of day and from various viewpoints. Each photo was then indexed using the generated vocabulary and the most visually similar

I_{960} I_{45}

Figure 4.25: **Urban experiment – Loop detection.** Example of image pair corresponding to the loop closure.

image from the original dataset was extracted. Figure 4.26 illustrates the results. The majority of photos were correctly associated ($\simeq$ 90%). Generally, the cases where OVV did not correctly identify the location were the result of: (i) extreme zooming, where the query pictures show details of the buildings not modeled in the vocabulary due to the limited resolution of the original dataset; (ii) severe obstructions that block most of the visual content modeled in the vocabulary; (iii) extreme lighting changes – pictures taken in the early evening or at night, where most of the visual details are lost due to low contrast. Moreover, in the last row of Figure 4.26 we illustrate an example of poor localization due to HDR processing of the query image.

4.4 Discussion

BoW methods have proved very efficient in detecting cross-overs in visual navigation and mapping, especially under lighting and camera view-point changes, occlusions, etc. However, state of the art visual bag of words methods limit the flexibility of

Figure 4.26: **Urban experiment – Location identification.** *Google Images* photos used as query images (left column) and the most visually similar image from the original dataset (right column). The last row shows an example of poor location identification, due to the post-processing of the query photo.

the visual surveys, as they require strong knowledge of the surveyed area and heavy user intervention.

We have developed a new visual BoW method for loop-closure detection, oriented towards online navigation and mapping. The method uses a novel incremental vocabulary building process that eliminates the need of the offline training stage. The vocabulary is initialized from visual information extracted from a reduced number of images. As the camera moves, the vocabulary is extended to model new visual information corresponding to newly surveyed parts of the scene. In order to decrease redundancy and augment the efficiency of the vocabulary building process, we present a novel vocabulary update criterion, which takes into account the visual information present in the images. In this way, OVV allows for navigation and visual mapping with no *a priori* information of the environment.

The vocabulary building process uses a new method, based on Fisher's linear discriminant, that takes into account the global data distribution. Experiments show that this approach increases both the repetitiveness and the discriminative power of the resulting vocabularies. Also, using LDA during the vocabulary building enables data dimensionality reduction, decreasing the computational time related to vocabulary building and image indexing.

In the context of a constantly changing vocabulary, we propose a new hierarchical feature-cluster association technique, that increases the stability of feature labeling. We show that stable feature labeling is critical in detecting visual similarities between images that are indexed at different vocabulary stages. Also, in this context, we propose a novel incremental image re-indexing method, eliminating the high cost of repeatedly indexing the images as the vocabulary changes.

Consequently, we present some experimental results that show the applicability of the method in cross-over detection for online navigation and mapping in underwater

and land-based environments.

We are currently developing a method for decreasing the complexity of the visual vocabularies by eliminating small, insignificant clusters at the bottom of the hierarchy. This would allow OVV to be applied on larger sequences of images. Also, the computational times related to vocabulary building and image indexing could be reduced by using Graphics Processing Unit (GPU)-based parallel processing.

Chapter 5

Online 3D Model Simplification

Scene reconstruction algorithms approximate the shape of the scene using 3D features such as vertices or lines. These features can be seen as discrete measurements of a continuous model representing the scene. Clearly, the higher the number of the 3D features, the higher the accuracy of the scene structure estimation.

When navigation and mapping algorithms have to deal with large areas, however, the amount of data may prove overwhelming. This is especially the case when Kalman Filter or Global Alignment algorithms are used, in which the complexity of the problem grows with the square of the number of scene features.

The solution to this problem is reducing the number of extracted scene features. The difficulty stands in selecting the 3D features in a way to minimize the impact on the precision of the resulting scene model.

The problem of reducing the complexity of 3D models while maintaining the model precision has been studied by the computer graphics community where it is known as *mesh simplification*. The state of the art in mesh simplification includes a wide range of alternatives. In [78,121], the authors divide the 3D volume into a user-specified grid. Then, the model is simplified by removing all vertices within a grid

cell, maintaining only the most representative vertices. Schroeder *et al.* [129] use a multiple pass simplification, based on a user specified user error threshold. Eck et al. [26] use a wavelet-based approach to create a fast multi-resolution representation of the original surface. A similar multi-resolution approach is employed by Progressive Meshes (PM) [59, 60], a widely used method in real-time 3D rendering. Other authors have proposed the use of color and texture information in addition to the shape in the simplification criteria [45, 46, 150], minimizing visual aliasing due to model simplification.

Unfortunately, all mesh simplification algorithms are inherently offline, in the sense that the entire scene geometry must be available during the simplification process. We propose a novel algorithm that carries out the model simplification sequentially, as the model is being generated. The simplification is done by selecting the vertices that are most representative for the scene geometry, reducing the redundancy in describing 3D shapes. In order to better understand the concept, consider the simple example of Figure 5.1a, which illustrates a 2D profile as the cross section of a 3D relief. By extracting vertices around the edges of the slopes (marked by dots) and applying linear interpolation (dotted lines), a good approximation of the shape is obtained. The algorithm follows this concept, selecting 3D vertices on edges/surface inflexions of the objects present in the scene. Similarly to the interpolation in Figure 5.1a, these vertices provide the basis for surface interpolations that accurately approximate the geometry of the scenes.

As the Online Model Simplification (OMS) process is carried out sequentially, in parallel with the scene reconstruction process, the scene model is not fully known during vertex selection. Instead we approximate the scene geometry using depth maps. From these depth maps, we extract interest points, corresponding to object edges/surface inflexions, hereafter called *geometrical features*. The geometrical fea-
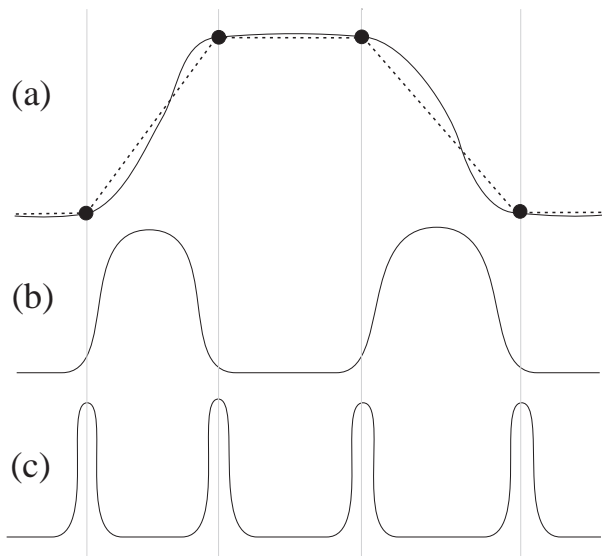
Figure 5.1: **Simple 2D example of feature extraction from a topological point of view.** 4 feature points provide a good initial piece-wise linear approximation of the curved profile (a); absolute value of first derivative (b); the 4 features correspond to the maxima of the response of the second derivative (c).

tures are then substituted by 3D vertices. The result is a small subset of 3D vertices that accurately describes the geometry of the scene.

Figure 5.2 outlines the main modules of the OMS algorithm. There are two parallel pipelines: one computes the structure of the scene (DPR-SfM) and the second extracts the geometric features. The two pipelines are merged in order to select the most representative vertices for the structure of the scene. Hereafter we describe each stage of the vertex selection process.

5.1   Depth Map Computation

The first step for obtaining the depth map is the computation of the 2D optical flow $\mathbf{v} = [u \, v]^T$ from pairs of images. The GDIM method used for this step was proposed by Negahdaripour *et al.* [100,106], and later generalized to take advantage of color in addition to intensity information for improved robustness and estimation accuracy
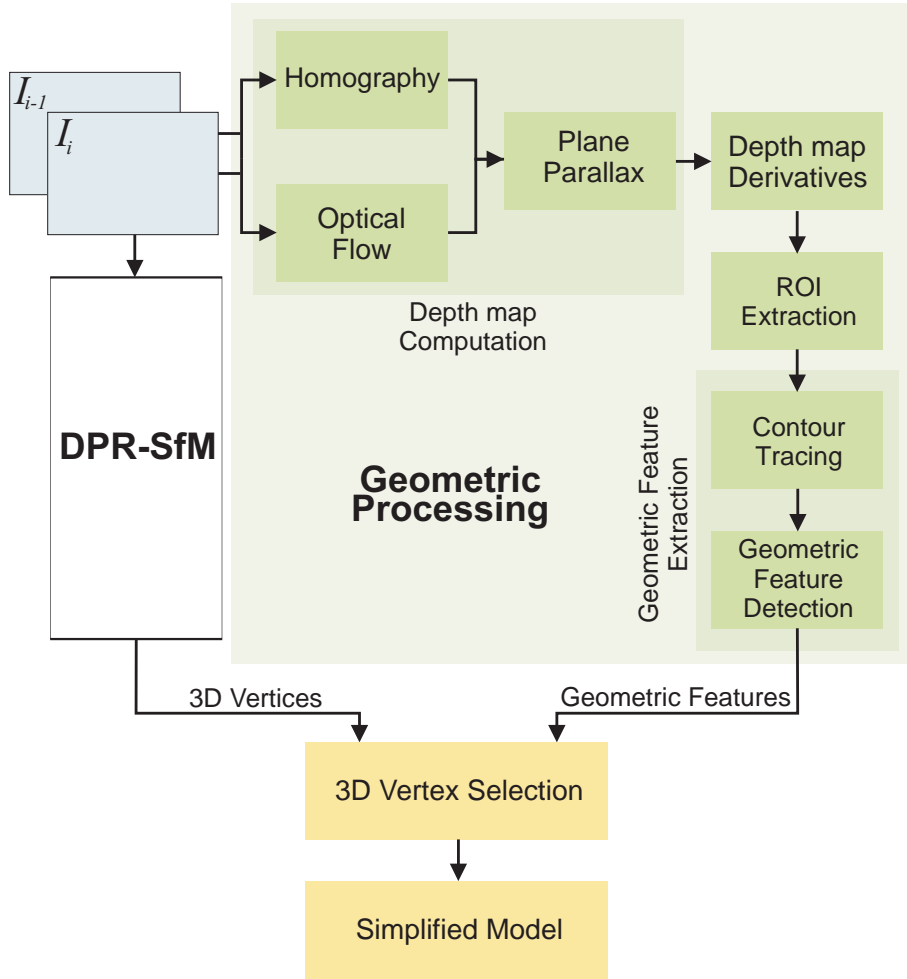
Figure 5.2: **Flowchart of the OMS algorithm.** The geometric processing (right) runs in parallel with the DPR-SfM pipeline (left). In the geometric processing block, first the depth map is obtained using plane parallax. Then, the regions of interest corresponding to edges of objects, are segmented from areas of local maxima of the depth map second derivative. Finally, geometric features, are then extracted from the regions of interest. The geometric features are used to select the vertices, generated by the DPR-SfM pipeline, that are the most representative for the scene.

[102] (see Section 2.1.2). The computed optical flow for each pair of consecutive frames $\{I_{i-1}, I_i\}$ provides an estimate of local disparities for depth computation.

Given the optical flow, an approximation of the depth map can be computed. Our previous proposals use Longuet-Higgins differential image motion model [107, 108]. However, this approach is computationally expensive, requiring iterative scene depth

and camera motion estimations. Here, we propose a fast, closed-form solution using plane-parallax. First, the homography ${}^{i}H_{i-1}$ is computed using all the correspondences between the two frames. This homography embodies the disparity induced by the camera motion on the average scene plane. From here we can obtain the parallax of the scene that represents a direct measurement of the depth variations of the scene $(\widehat{D})$:

$$\widehat{D}_i = (p_i - {}^{i}H_{k-1} \cdot p_{i-1}) - \mathbf{v}_k$$

where p_{i-1} and p_i are the image point coordinates of frames I_{i-1} and I_i respectively.

5.2 Depth Map Derivatives

In order to extract the geometric features, we consider two types of regions of interest: (i) object edges and (ii) surface inflexions, both of which correspond to large absolute values of the second derivative of the depth map (see Figure 5.1c) and will be called *edges* hereafter.

The second derivative of the depth map (D) is approximated by:

$$D''(x, y) = \frac{1}{N} \Sigma_{k=1}^{N} D(x, y) * LoG(\sigma_k)$$

where $*$ is the convolution operator, $LoG(\sigma_k)$ is the Laplacian of Gaussian with standard deviation $\sigma_k = m \cdot k$ and m is a predefined constant. D'' computed in this way is less sensitive to noise compared to the standard second derivative using local differences, while still providing high responses on the edges of the surfaces (Fig. 5.3b).
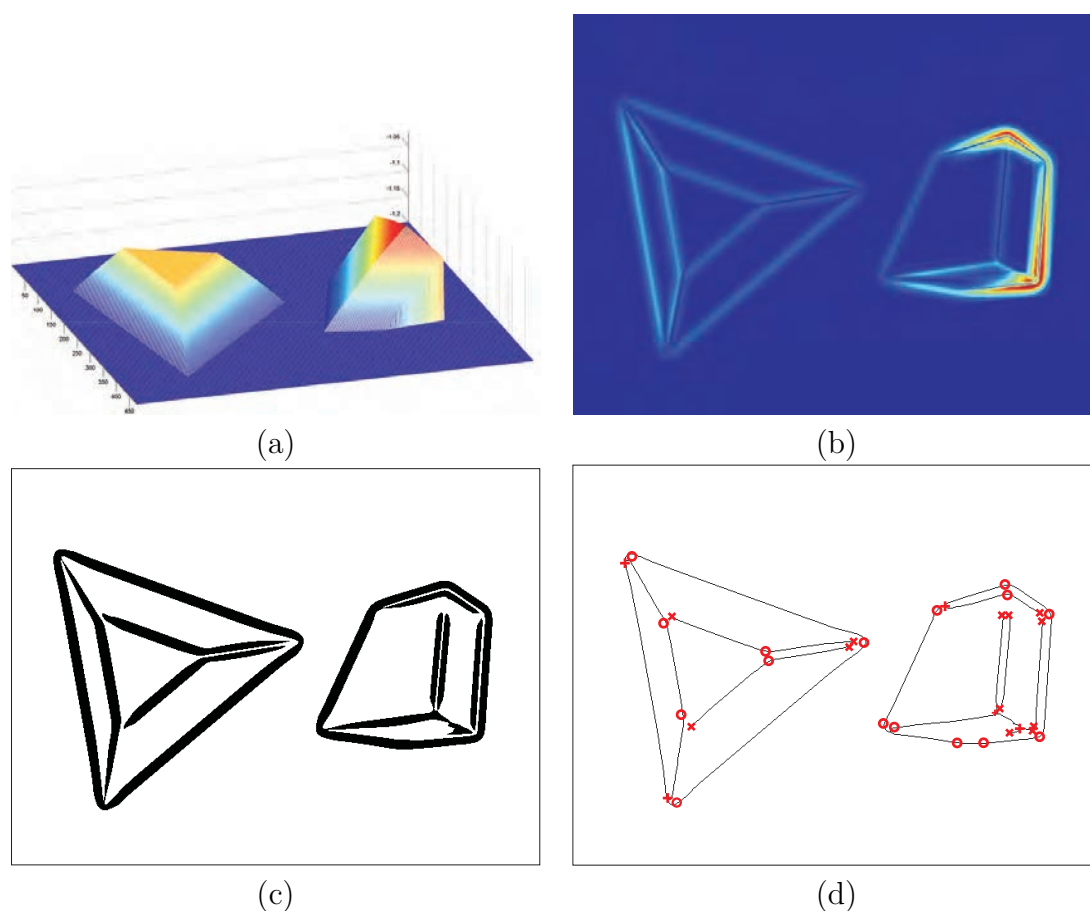
Figure 5.3: **Main steps of the OMS algorithm.** (a) depth map of the scene, (b) computation of the second derivative, (c) RoI extraction and (d) edge traces along with the extracted geometric features: " × " corresponds to line ends, " + " represents line junctions and " ∘ " denotes high curvature points.

5.3 RoI Extraction

As mentioned earlier, the regions of interest correspond to those areas where D'' has high absolute values. In order to extract these regions, a binarization using a constant threshold could be applied. However the steepness and the area of the slopes influence the magnitude and width of the peaks in D''. In this case, applying a fixed binarization would either not detect certain edges or would over-evaluate others. In order to obtain a more suitable binarization, D'' is locally normalized using:

$$\widehat{D''}(x,y) = \frac{D''(x,y) - \overline{w}_n(x,y)}{\sqrt{\overline{v}_n(x,y) - \overline{w}_n^2(x,y)}}$$

where

$$\overline{w}_n(x,y) = \frac{\sum_{i=x-n}^{x+n} \sum_{j=y-n}^{y+n} D''(i,j)}{(2n+1)^2}$$

and

$$\overline{v}_n(x,y) = \frac{\sum_{i=x-n}^{x+n} \sum_{j=y-n}^{y+n} (D''(i,j))^2}{(2n+1)^2}$$

Figure 5.3c shows the regions of interest after normalization and binarization.

5.4 Geometrical Feature Extraction

In order to minimize geometrical redundancy, only relevant edge points are extracted. To detect the edges, a thinning algorithm is applied to the regions of interest [71]. The result is a pixel-wide trace line following the edge (hereafter called *traces*), with each pixel corresponding to the local maxima of D'' (see Figure 5.3d).

Three types of geometrical features are defined along the traces:

- line end points

- line junction points

- high curvature points

Line end-points and line junction points are extracted by convolving the trace image with specific kernels taking into account 8-neighbor-connectivity. The curvature of the trace line along each point p is obtained by computing C_p within a $2n+1$ band along the line [24], with:

$$C_p = \frac{1}{(2n+1)} \sum_{i=p-n}^{p+n} e^{(-d_{ip}^2)(1-cos(\phi_p-\phi_i))}$$

where ϕ_p and ϕ_i represent the angles of the line normals at points p and i respectively; and d_{ip} represents the Euclidean distance between p and i.

High curvature points are extracted by locating local maxima of C_p where $C_p > t_c$. The threshold t_c is imposed in order to avoid false positives due to image aliasing.

Figure 5.3d illustrates the extracted geometric features: line junctions, line-ends and high-curvature points.

5.5   Selection of 3D Vertices

In order to obtain a reliable 3D reconstruction, the algorithm substitutes geometric features with vertices. In previous works we have proposed the direct selection of image features using the geometric criteria [107, 108]. However, as the selection is done prior to vertex reconstruction, there is no guarantee that all the selected features will provide reliable reconstruction.

Performing the selection process on the 3D vertices rather than on the image features overcomes this limitation. Substitution of each geometric feature with a vertex is carried out using a criteria based on two measurements: the uncertainty of the 3D location of the vertex and the "distance" between the geometric feature and the vertex. As the extraction of geometric features takes place in the images, in order to have a common frame, 3D vertices are represented by their image projections. Therefore, the score of substituting the geometric feature g with vertex X, in frame I_i is given by:

$$s_F(g, X) = (1 - \overline{\Sigma}_{\Delta x}) \cdot cos(\frac{\pi}{2} \cdot \frac{\|\Pi_i \cdot X - g\|}{max_G}) \tag{5.1}$$

where max_G is a pre-established maximum substitution distance. $\overline{\Sigma}_{\Delta x}$ is the uncertainty of vertex X normalized among all the possible candidates of g (see Appendix A). The use of the cosine function in eq. (5.1) applies a nonlinear weight that rewards vertices which are close to the geometric feature and penalizes those towards the outer radius max_G.

Given a feature g in frame I_i, s_F is computed for all vertices whose projections fall within a radius of max_G. The vertex with the highest score s_F is considered the substitute of g. This selection process is carried out for all geometric features.

Using the substitution criteria show in eq. (5.1), OMS creates a tradeoff between vertex reconstruction precision and geometric approximation. As the Online Model Simplification process runs in parallel with the DPR-SfM, the two processes can be seen as a single SfM module, whose output is a reduced yet accurate scene model.

Obtaining a simplified model directly, without the necessity of generating the full model as an intermediate step, the computational and memory costs are drastically reduced, allowing reconstruction of more complex and larger scenes.

5.6 Experimental Results

The experiments reported in this section are aimed at evaluating the OMS algorithm. We are concerned with two aspects of OMS: (i) efficiency – its capacity to reduce the number of vertices in the 3D model and (ii) accuracy – the precision loss after model simplification. In each experiment, the evaluation was carried out by comparing the model containing the complete set of features (full model) with the simplified model. In order to provide the basis for comparison, we only tagged the vertices selected by OMS, without removing any vertices. In this way, vertices corresponding to the simplified model represent a subset of the vertices comprising the full model.

We define the efficiency of the OMS as follows:

$$\xi = \frac{N_{full} - N_{OMS}}{N_{full}} \cdot 100$$

where N_{full} is the number of vertices in the full model and N_{OMS} is the number of vertices in the simplified model.

We quantify the simplification accuracy using the average Hausdorff distance [98] between simplified model and the full model. As the Hausdorff distance is metric, we represent the error as percentage of scene depth.

We extend the analysis of OMS by comparing it with Progressive Meshes [60]. We have chosen this algorithm for 3 reasons: (i) it is a widely used algorithm in computer graphics and hardware-based rendering, (ii) it reduces the model complexity by selecting the most geometrically representative vertices, similarly to our algorithm and (iii) it allows the user to manually set the number of vertices in the simplified model, thus providing common basis for comparison.

Hereafter, we present some of the results we have obtained from image sequences representing outdoor and underwater environments.

5.6.1   Rocks Experiment

In this experiment, a set of rocks with various photometric (texture) and geometric (size and shape) properties were imaged on a planar concrete background (see Fig. 5.4a). During acquisition, the camera was oriented towards the ground with little pitch and roll movement, and rotated around its optical axis so that it maintained a constant orientation with respect to the motion direction (*i.e.* simulating the motion of a survey platform). The sequence consists of 360 images of 694×519 pixels.

After applying DPR-SfM on the sequence, the resulting full model shown in
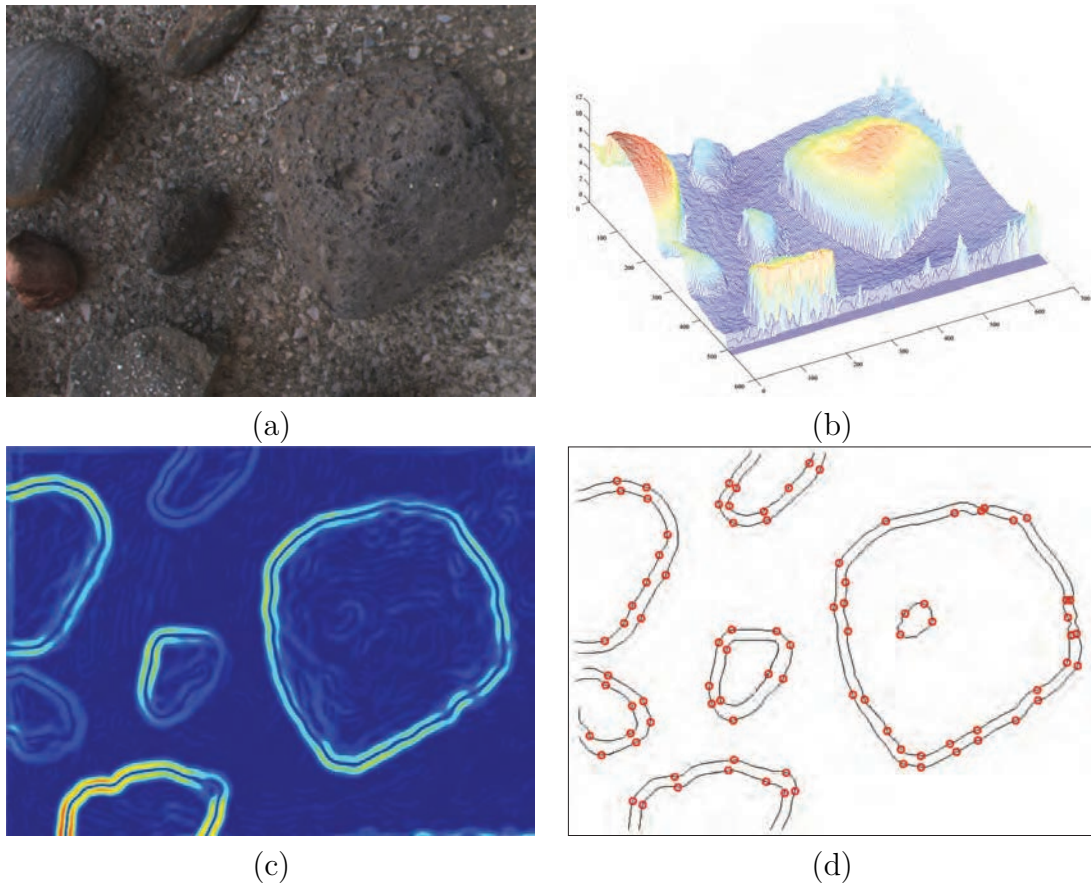
Figure 5.4: **Rocks experiment – Geometrical feature extraction.** (a) sample image from the sequence, (b) depth map, (c) second derivative and (d) object contours and the extracted geometrical features).

Figure 5.5a contains 14,000 vertices. When we performed model simplification in parallel with DPR-SfM, OMS introduced an overhead of 0.11s/frame in the model update step, as the latter requires the computation of the vertex covariance. The optical flow computation times are highly dependent on the image resolution. For high resolution images, we use subsampling prior to optical flow computation, resulting in significant gains in computational times with minimal loss of depth map precision. For this image sequence, we obtained an average of 1.2s/frame for depth map computation without subsampling. The rest of the steps for the geometrical feature extraction and vertex selection stages had small computational costs, averaging a

(a)

(b)

Figure 5.5: **Rocks experiment – 3D Structure.** (a) full model containing 14,000 vertices, (b) simplified model using the geometric criteria containing 432 vertices and (c) texture rendering of the simplified model.

total of 0.1s/frame.

OMS allows the user to specify the maximum substitution distance max_G as an input parameter (see eq. (5.1)). As max_G is represented in image pixels, we avoid the resolution dependency by defining the user parameter as percentages of image width. In order to assess the influence of max_G on the outcome of OMS, we generated the simplified model using different values of the parameter and compared the results with PM. In each case, we set the number of vertices in PM to be the same as those of the OMS model.

Table 5.1: **Rocks experiment – Comparison between OMS and PM.** Low values of max_G result in a highly simplified model, where PM is more efficient than OMS. Increasing the value of max_G, improves the accuracy of OMS to a point where the difference between OMS and PM is small. Values of max_G above 4.3 do not bring any significant improves neither in efficiency nor accuracy of OMS.

max_G	Vertices	Efficiency [%]	Accuracy OMS [%]	Accuracy PM [%]
0.7	309	97.8	1.31	0.77
1.4	398	97.2	0.96	0.73
2.9	424	97.0	0.87	0.73
4.3	432	96.9	0.86	0.72
5.8	446	96.8	0.86	0.72

Table 5.1 shows the results of the experiment. Using low values of max_G limits the amount of geometrical features that are substituted by vertices. This increases the efficiency of the model simplification at the expense of accuracy. Increasing the value of max_G highly improves the accuracy of OMS to a point where the results of OMS and PM are similar. This shows that our OMS approach is nearly as accurate as the batch PM algorithm. Figure 5.5b illustrates the simplified model using $max_G = 4.3$.

5.6.2 Coral Head Experiment

The data for this experiment were acquired during one of the coral reef UoM ROV surveys. The dataset consists of $2,000$ frames of 512×384 pixels, corresponding to a coral head of 1m in height and its surroundings (see Figure 5.6a).

The aim of this experiment was to test the OMS algorithm using real, geometrically complex underwater scenes under extreme, changing lighting conditions. For this, the sequence was chosen to include both almost flat regions and high 3D structure regions, with images affected by sun flickering, scattering, blurring and decreased image contrast due to light attenuation.

During OMS process, the lighting conditions reduced the accuracy in optical flow
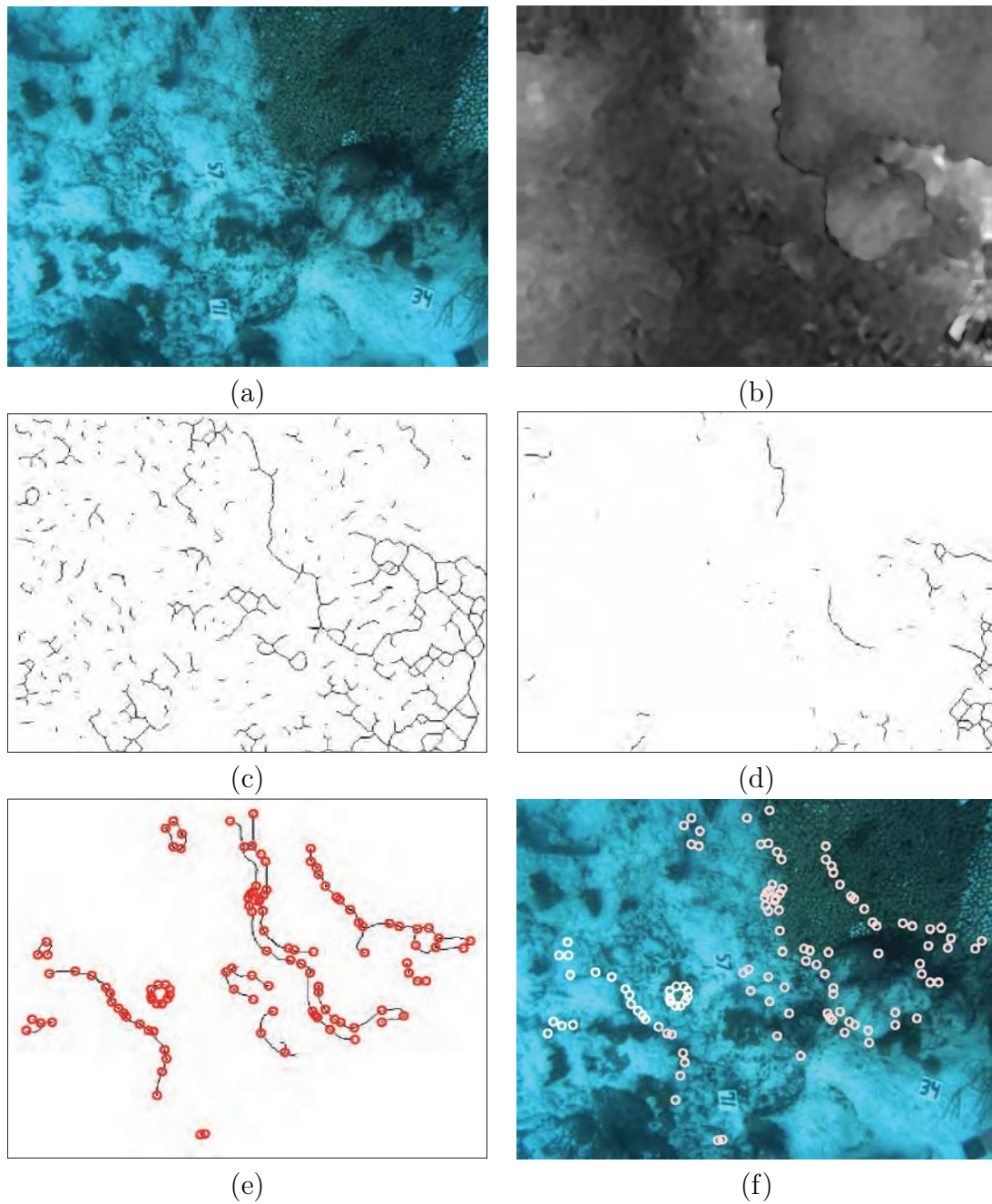
Figure 5.6: **Coral head experiment – Geometrical feature extraction.** (a) sample image from the sequence; (b) depth map; (c) using local differences for second derivative and simple binarization using a simple threshold the result is very noisy; (d) increasing the binarization threshold results in edge loss; (e) edge trace and geometrical features extracted using proposed method; (f) geometrical features on the input image.

computation, increasing the noise in the depth map computation (see Figure 5.6b). The high level of noise in the depth map made the detection of edges more difficult. By computing the second derivative using local differences and applying a fixed binarization threshold in Figure 5.6c, the result is very noisy, with few extracted edges corresponding to true scene edges. However, applying the locally normalized binarization method, the effect of noise was highly reduced, resulting in a correct estimation of the edges (Figure 5.6e) and the geometrical features (Fig. 5.6f).

The full scene model, shown in Figure 5.7a contains 15,000 vertices. Using OMS, the number of vertices was reduced to 641 (Fig. 5.7b), resulting in a decrease of 95.7% in model complexity. The OMS algorithm highly reduced the number of vertices in the close-to-planar regions, while maintaining the model complexity in the regions with high 3D structure (*i.e.* the coral head in the center). The error introduced by the OMS algorithm was 1.15% while the error introduced by PM simplification was 0.92%. This shows that OMS has good performance under challenging conditions, with an accuracy comparable with PM.

5.6.3   Coral Reef Experiment

In this section, we discuss the result of OMS on the dataset presented in Section 4.3.2. The aim of this experiment was to assess the efficiency of OMS under a dense set of vertices. For this, we increased the number of extracted image features to 5,000 features/frame. This resulted in a 3D scene model that contains 62,322 vertices. After applying OMS, the model was reduced to only 762 vertices, with a simplification efficiency of 98.8% (see Figure 5.8). This shows that, as expected, the complexity of the simplified model depends only on the shape of the scene. This offers a significant advantage over regular SfM modeling, where the number of vertices depends on image resolution, number of extracted image features, image
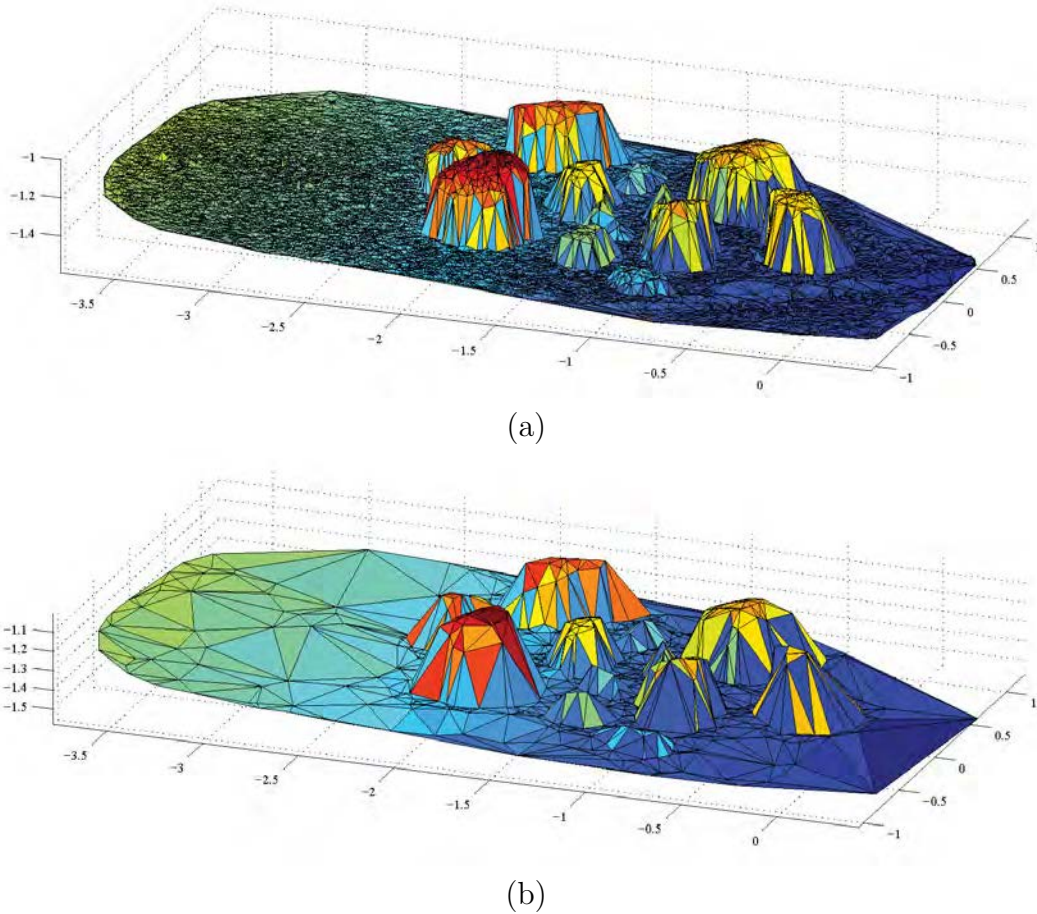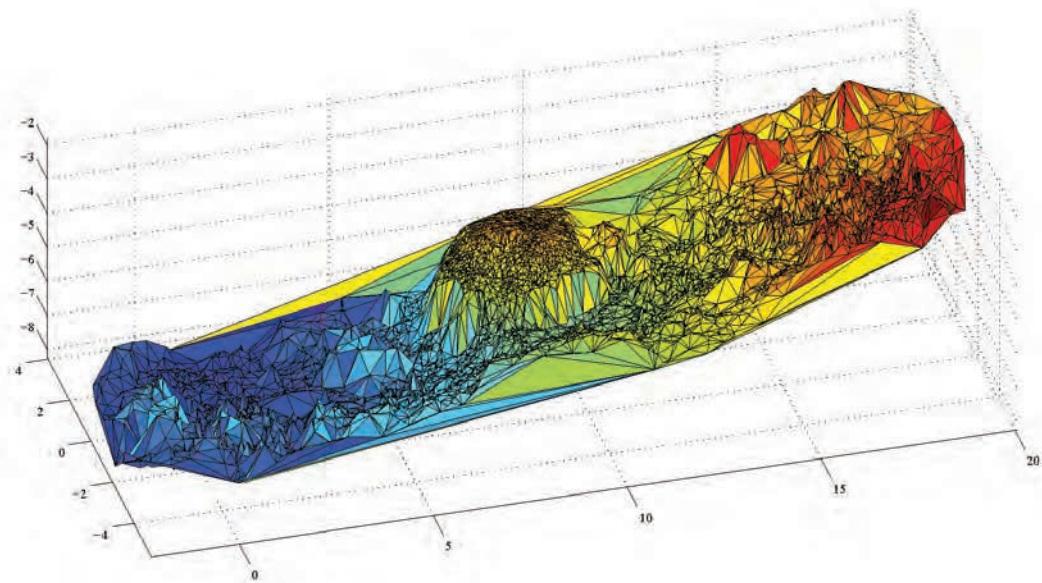
(a)
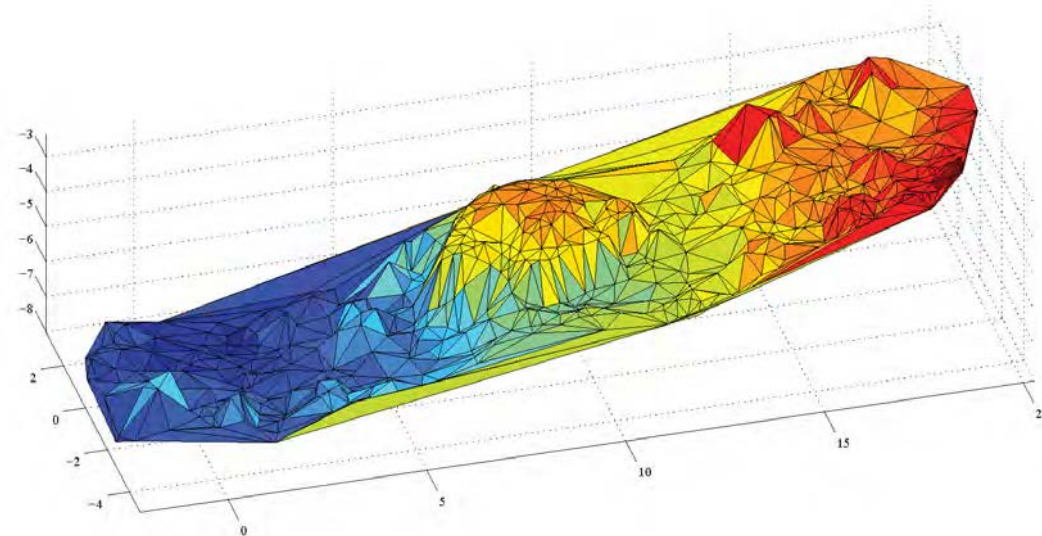
(b)

Figure 5.7: **Coral head experiment – 3D Structure.** (a) full model containing 15,000 vertices, (b) reduced model using the geometric criteria containing 641 vertices.

content, etc.

The error introduced by OMS simplification was 1.17%, while in the case of PM was 1.01%.

(a)

(b)
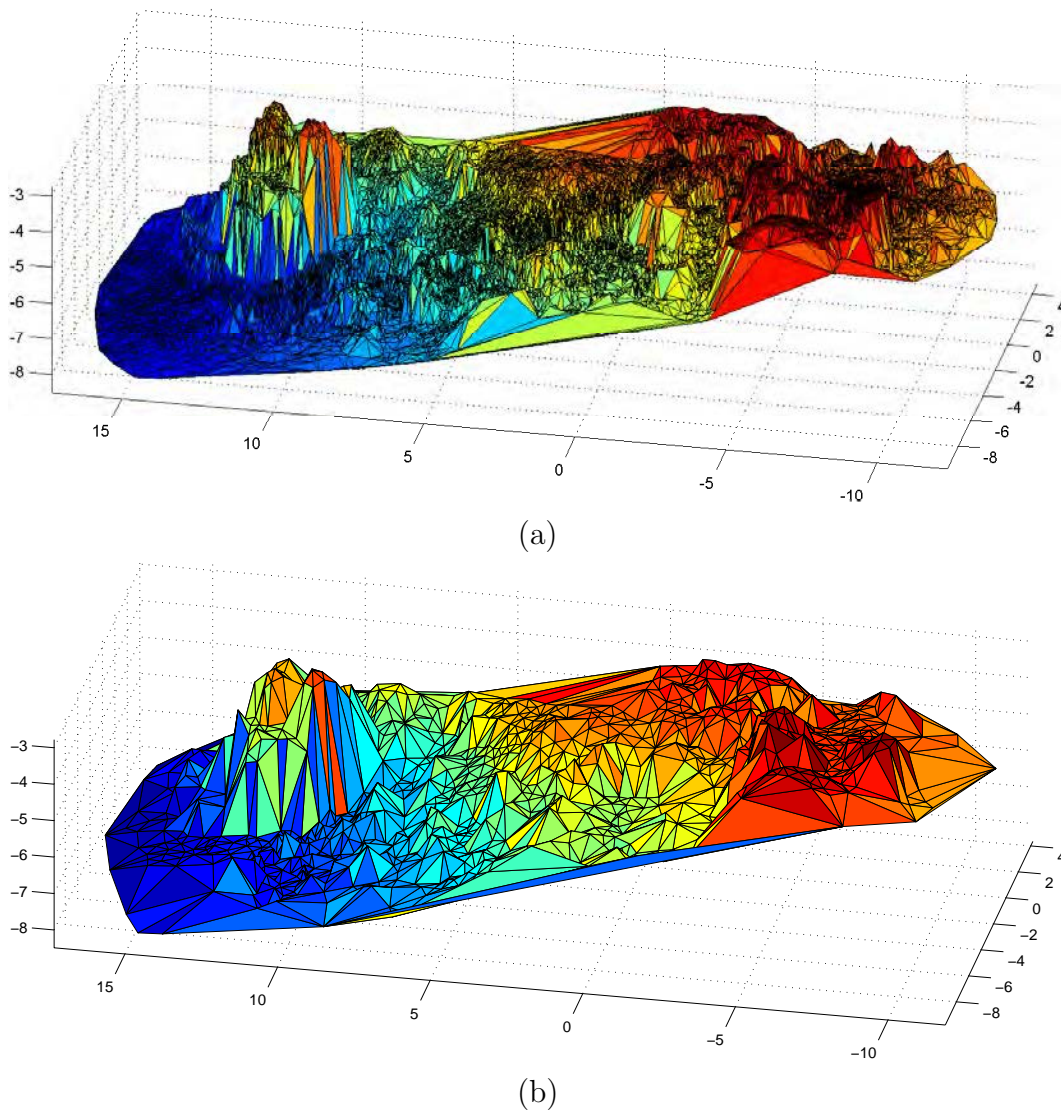
Figure 5.8: **Coral reef experiment – 3D Structure.** (a) the full model contains a dense set of 62,322 vertices, (b) applying OMS, the resulting model is highly simplified to only 762 vertices.

5.7 Discussion

Model simplification methods greatly reduce the cost related to processing, storage and representation of complex 3D models. The model simplification techniques proposed in literature are inherently offline, requiring the entire 3D model to be available during simplification. These methods are not adequate for online 3D modeling

applications, which could benefit from on-the-fly model simplification strategies.

In this chapter we presented a novel online model simplification algorithm oriented towards large scene reconstruction algorithms. OMS does not require the geometry of the scene to be available as an intermediate step. The simplification is carried out by analyzing the scene geometry locally, using plane-parallax approximations, and selecting only those 3D vertices that are geometrically representative to elements present in the scene. The vertex selection criteria not only takes into account the geometrical representativeness of the vertices but also their reconstruction accuracy.

Applying OMS in parallel with an SfM algorithm results in highly simplified scene models, which can significantly decrease the computational costs related to mapping.

We show through experimental results that model simplification using OMS has minimal impact on the accuracy of the model, with results similar to state of the art offline model simplification algorithms.

Future work related to Online Model Simplification will focus on multilayered 3D models, containing vertexes classified on geometrical relevance. The result is a multi-resolution representation of the scene similar to PM. This would allow the user to select the level of detail, depending on the specific needs, hardware limitations, etc.

Chapter 6

Conclusions

This thesis presented a complete framework for efficient 3D scene modeling and mosaicing. The objective was to develop an efficient and flexible tool for remote scientific studies that require 3D and visual information under any type of acquisition conditions and scene types. Using such a tool, where no additional sensor information and no special acquisition conditions are required, decreases the complexity and costs related to scientific visual studies.

During the presentation of this work, we mainly focused on underwater scene modeling due to the increased difficulty and additional challenges that are present in this environment. Nevertheless, we show successful results on applying the framework on other types of environments, including land-based and urban scenes.

The core of the framework is based on a novel SfM algorithm – DPR-SfM. The algorithm generates the scene model sequentially, in two stages. In the first stage, a seed model is created using camera motion estimation techniques. The seed model corresponds to the first few frames of the sequence, representing a small subregion of the scene. The second stage extends the seed model in order to cover the entire surveyed area. While the first stage uses classical SFM techniques, the second

stage uses a direct camera-to-scene registration method which increases the accuracy, robustness and flexibility of DPR-SfM. Also, results show that direct camera registration enables the algorithm to quickly recover from scene occlusions and tracking errors (*e.g.* due to excessive blurriness induced by fast camera movements, camera temporary failures, etc.).

Generally, large scale scene modeling algorithms use additional sensor information such as camera position and attitude for accurate results. We show that DPR-SfM achieves the same accuracy with no additional information, increasing its flexibility with respect to other SFM techniques. DPR-SfM can be readily applied on image sequences acquired with any type of camera, both still and video, using natural or artificial lighting (*e.g.* strobe/focus lighting for deep waters).

The direct camera pose registration uses a novel dual RANSAC projective/homography approach which allows the DPR-SfM algorithm to accurately model both planar and non-planar scenes. This is particularly important in underwater and urban scenes, where parts of the scene can have significant parallax while other parts can be perfectly planar. The use of robust estimation methods was also extended to vertex position recovery. We show that using robust estimation for both camera pose and vertex position estimations increases the accuracy and robustness of the method.

DPR-SfM uses an efficient and flexible scene database that enables the parallel use of multiple feature extractors/descriptors, while allowing fast camera registration in case of complex and large scenes. In this context, we employed a Kd-tree scheme for efficient association between image features and model vertices.

The second part of the framework deals with the detection of cross-overs during visual surveys. The novel BoW method (OVV) is oriented towards online navigation and mapping, eliminating significant drawback of state of the art visual vocabulary

algorithms such as strong *a priori* knowledge of the surveyed area and tedious user intervention.

OVV uses an incremental vocabulary building process that eliminates the need for the offline training stage. During the survey, the vocabulary is initialized from visual information extracted from a small number of images, corresponding to the beginning of the sequence. Using a novel vocabulary update criterion which takes into account the visual information present in the images, the vocabulary is updated in order to constantly represent the visual information contained in the scene.

The vocabulary is built using a novel data clustering method. The clustering, based on Fisher's linear discriminant, takes into account the global data distribution rather than local inter-cluster relations. We show that such an approach ensures a more efficient data distribution, increasing both the repetitiveness and the discriminative power of the resulting vocabularies. The discriminative power of the vocabularies is further improved using Linear Discriminant Analysis, which increases the separability of the visual words within the vocabularies. Also, the use of LDA enables data dimensionality reduction, decreasing the computational costs related to vocabulary building and image indexing.

In the context of a constantly changing vocabulary, we propose a new hierarchical feature-cluster association technique, that increases the stability of feature labeling. We show that stable feature labeling is critical in detecting visual similarities between images that are indexed at different vocabulary update steps. Also, to increase the computational efficiency of OVV, we propose a novel incremental image re-indexing method, eliminating the high cost of repeatedly indexing the images as the vocabulary changes.

Finally, we propose a novel Online Model Simplification algorithm oriented towards large scene reconstruction algorithms. OMS simplifies the 3D model sequen-

tially, by analyzing the scene geometry locally, using plane-parallax approximations. During simplification, OMS selects only those 3D vertices that are geometrically representative for elements present in the scene (*e.g.* edges, corners, surface inflections, etc.). The vertex selection criteria not only takes into account the geometrical representativeness of the vertices but also their reconstruction accuracy.

We show through experimental results that model simplification using OMS greatly reduced the complexity of the models while having minimal impact on the accuracy, with results similar to state of the art offline model simplification algorithms.

6.1   Contributions of the Thesis

In this thesis, we have presented a complete framework for 3D scene modeling. Particularly, we focused on developing an accurate and flexible SfM algorithm, online cross-over detection and 3D model simplification. Experimental results presented in this thesis show the efficiency and accuracy of the three modules. Hereafter, we present the main contributions of this thesis:

- In Chapter 3 we proposed a novel SfM algorithm based on direct registration between camera and scene model. While the model is initialized using classical motion-based techniques, the scene model is extended using a new sequential two step approach: (i) the camera pose is obtained from camera-model registration and (ii) using the camera pose, the model is extended to comprise the new information extracted from the camera view. The camera registration uses a novel dual approach that allows reconstruction of both planar/non-planar scenes.

- In Chapter 4 we developed an online cross-over detection algorithm, based on

visual BoW. OVV uses a novel incremental visual vocabulary that eliminates the need of *a priori* knowledge of the scene being surveyed. The vocabulary building process uses an automatic update criterion, based on image content, that reduces the number of vocabulary updates. Also, the vocabulary building uses a novel clustering approach that increases the quality of the vocabulary and data separability while allowing data dimensionality reduction. The natural convergence criterion used during vocabulary building eliminates any user intervention, increasing the ease of use of the method.

Image indexing is carried out by means of a novel indexing method using hierarchical trees. The method increases the stability of the image indexing process in the context of dynamic vocabularies. Furthermore, as vocabularies constantly change, we avoid repeated complete indexing of frames using an efficient incremental re-indexing method that takes into account the changes in the vocabulary.

- In Chapter 5 we propose a novel method for 3D model simplification oriented towards online 3D modeling applications. The method analyzes the scene locally, using plane-parallax, hence not requiring knowledge of the full scene model. Based on plane-parallax approximations, the method selects vertices that are geometrically representative. For this, we present a novel vertex selection criteria that takes into account both geometric relevance and reconstruction accuracy of the vertices.

6.2 Ongoing and Future Work

The work presented in this thesis can be improved and extended in several ways. We present hereafter ongoing and future work directions:

Direct Structure from Motion After camera pose registration, image patches around features can be warped using camera-to-model transformations, reducing the effect of extreme geometric distortions on feature tracking. Also, the scene reconstruction accuracy can be improved, using cross-correlation as a refinement step after feature matching. Finally, the computational time of feature-to-model association can be highly decreased using recent developments in GPU-based parallel processing.

Online Loop Detection The complexity of the visual vocabularies can be reduced by eliminating small, insignificant clusters at the bottom of the hierarchy. This would allow online cross-over detection for larger scenes in a more efficient way. Again, the use of GPU-based parallel processing would highly decrease the computational time related to vocabulary building and image indexing.

Online 3D Model Simplification A new multi-resolution representation of the scene based on vertex geometrical relevance could be developed, similar to PM. Such a representation would allow the user to select the level of detail of the model, depending on the specific needs, hardware limitations, etc.

6.3 Publications

During the development of the work presented in this thesis, the main contributions were presented in the following publications:

- T. Nicosevici, N. Gracias, S. Negahdaripour and R. Garcia. Efficient 3D Modeling and Mosaicing. In *Journal of Field Robotics*, vol. 26, no. 10, pages 759–788, 2009.

- T. Nicosevici and R. Garcia. Online Visual Vocabularies for Robot Navigation

and Mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

- T. Nicosevici and R. Garcia. Online Robust 3D Mapping Using Structure from Motion Cues. In *MTS/IEEE OCEANS Conference*, pages 1–7, 2008.

- T. Nicosevici, R. Garcia, S. Negahdaripour, M. Kudzinava and J. Ferrer. Identification of Suitable Interest Points Using Geometric and Photometric Cues in Motion Video for Efficient 3-D Environmental Modeling. In *IEEE International Conference on Robotics and Automation*, pages 4969–4974, 2007.

- T. Nicosevici, S. Negahdaripour and R. Garcia. Monocular-based 3-D Seafloor Reconstruction and Ortho-mosaicing by Piecewise Planar Representation. In *MTS/IEEE OCEANS Conference*, vol. 2, pages 1279–1286, 2005.

Appendix A

Estimating the Uncertainty of 3D Vertices

The selection of the 3D vertices explained in Chapter 5 is based on computing the first-order approximation of the uncertainty of the 3D points, obtained from noisy measurements of point projections across several views. We follow the approach proposed by Haralick for propagating the covariance matrix when the data and the parameters are implicitly related by the minimization of a cost function [54]. Here, the cost function is represented by the reprojection error shown in eq. (3.3).

For a given feature track, we consider p to be a $2M \times 1$ of noisy measurements, so that $p = p_0 + \Delta p$, where p_0 indicates the ideal noise-free quantities and Δp is random additive noise. Similarly, we consider $P = P_0 + \Delta P$, where P_0 is the vector of ideal noise-free estimates and ΔP is the associated random perturbation induced by Δp. The method assumes the following two conditions:

- The function $E(p, P)$ has finite second partial derivatives.

- The random perturbations Δp are small enough, so that $E(p_0, P_0)$ and $E(p, P)$ can be well related by a first order Taylor series expansion.

Let $\varrho_E(p, P)$ be the gradient of E with respect to P,

$$\varrho_E(p, P) = \frac{\partial E}{\partial P}(p, P)$$

Under the above assumptions, an estimate for the covariance $\Sigma_{\Delta P}$ of the noise in P, is obtained by

$$\Sigma_{\Delta P} = \left(\frac{\partial \varrho_E}{\partial P}\right)^{-1} \cdot \left(\frac{\partial \varrho_E}{\partial p}\right)^T \cdot \Sigma_{\Delta p} \cdot \frac{\partial \varrho_E}{\partial p} \cdot \left(\frac{\partial \varrho_E}{\partial P}\right)^{-1}$$

Given the simplicity of the cost function, analytic expressions for $\frac{\partial \varrho_E}{\partial P}$ and $\frac{\partial \varrho_E}{\partial p}$ are easy to obtain. For the purpose of selecting the 3D vertices with lower uncertainty (detailed in Section 5.5), we consider $\Sigma_{\Delta p} = \sigma^2 \cdot I_{2n}$ where σ is the standard deviation of the reprojection residues obtained from our test data, and I_{2n} is the $2n \times 2n$ identity matrix.

Appendix B

Loop Closing and 3D Model Correction

During a scene reconstruction the 3D map is generated incrementally (see Chapter 3). In a cross-overs situation, there is a certain offset between the two (or more) subregions of the map corresponding to the same region of the scene, mainly due to the drift in camera pose and map estimations, (*e.g.* see Figures 3.23 and 3.38). We aim here to correct this shortcoming by using loop-closure information to correct the 3D models.

As mentioned in Chapter 4, we run OVV in parallel with DPR-SfM. In Figure B.1 we illustrate an example where, during the registration of frame 180, a possible cross-over is detected with frame 31. Before the model correction, the region of the scene corresponding to the cross-over is represented two times in the model – in the subset corresponding to frames 31 and 180, respectively.

Using the cross-over information, we register frame 180 two times. Each time, we use only those feature tracks (and vertices) corresponding to each of the subsets (see Section 3.4). The result is a group of image features that are registered with
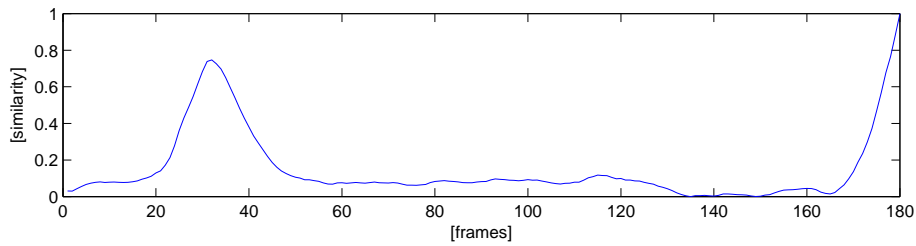
Figure B.1: **Loop closing – Example of loop detection.** Using OVV, a loop closure is detected between frames 31 and 180.
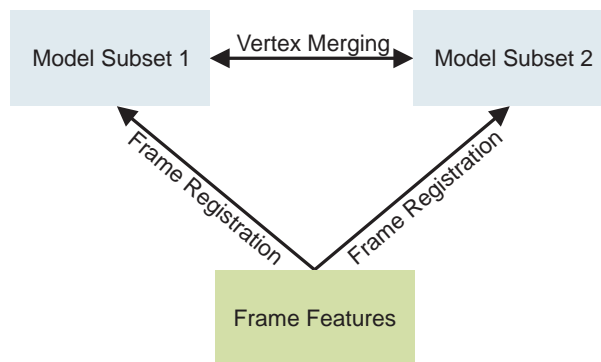
Figure B.2: **Loop closing – Vertex merging.** Registering an image with two subregions of the model corresponding to a cross-over, the resulting vertex pairs corresponding to the same pre-image region are merged.

vertices from both model subsets. In other words, using image-to-model registration, we identify pairs of vertices that correspond to the same pre-image region. Then, for each vertex pair, we merge the corresponding feature tracks and keep the vertex position having the lowest uncertainty (see Appendix A). Doing so (see Figure B.2) allows us to introduce new constrains in the model, corresponding to the detected loop-closure.

In order to correct the 3D model using the newly obtained cross-over information, we apply the BA algorithm presented in [76]. For some examples of models before and after loop-closure detection and model correction, refer to Figures 3.23 and 3.38.

Bibliography

[1] A. Angeli, S. Doncieux, J. A. Meyer, and D. Filliat. Incremental vision-based topological SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1031–1036, 2008.

[2] A. Angeli, S. Doncieux, J. A. Meyer, and D. Filliat. Real-time visual loop-closure detection. In *IEEE International Conference on Robotics and Automation*, pages 1842–1847, May 2008.

[3] X. Armangue and J. Salvi. Overall view regarding fundamental matrix estimation. *Image and Vision Computing*, 21:205–220, 2003.

[4] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *Association for Computing Machinery*, 45:891–923, 1998.

[5] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press,, 1999.

[6] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[7] J. Batlle, T. Nicosevici, R. Garcia, and M. Carreras. Rov-aided dam inspection: Practical results. In *6th IFAC Conference on Manoeuvring and Control of Marine Crafts*, Girona, Spain, September 2003.

[8] H. Bay, T. Tuytelaars, and L. J. Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, pages 404–417, Graz, Austria, May 2006.

[9] P. A. Beardsley, A. Zisserman, and D. W. Murray. Navigation using affine structure from motion. In *European Conference on Computer Vision*, volume 801, pages 85–96, 1994.

[10] P. R. Beaudet. Rotationally invariant image operators. In *IAPR International Conference on Pattern Recognition*, pages 579—-583, Tokyo, Japan, 1978.

[11] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, Inc., 2002.

[12] J. Y. Bouguet. Camera Calibration Toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[13] M. Bowling, D. Wilkinson, A. Ghodsi, and A. Milstein. Subjective localization with action respecting embedding. In *The International Symposium of Robotics Research*, 2005.

[14] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37:855–866, 1971.

[15] M. Brown, R. Hartley, and D. Nister. Minimal solutions for panoramic stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, June 2007.

[16] M. Brown and D. G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *International Conference on 3D Digital Imaging and Modelling*, pages 56–63, 2005.

[17] A. M. Buchanan and A. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 316–322, 2005.

[18] F. Caballero, L. Merino, J. Ferruz, and A. Ollero. Homography based Kalman filter for mosaic building. applications to UAV position estimation. In *IEEE International Conference on Robotics and Automation*, pages 2004–2009, 2007.

[19] D. P. Capel. *Image Mosaicing and Super-resolution.* Springer Verlag, 2004.

[20] J. Castellanos and J. Tardos. *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach.* Kluwer Academic Publishers, 1999.

[21] G. Csurka, C. Bray, C. Dance, , and L. Fan. Visual categorization with bags of keypoints. In *European Conference on Computer Vision*, pages 1–22, 2004.

[22] M. Cummins and P. Newman. Probabilistic appearance based navigation and loop closing. In *IEEE International Conference on Robotics and Automation*, Rome, April 2007.

[23] M. Cummins and P. Newman. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *International Journal of Robotics Research*, 27(6):647–665, 2008.

[24] J. Deschênes and D. Ziou. Detection of line junctions and line terminations using curvilinear features. *Pattern Recognition Letters*, 21:637–649, 2000.

[25] R. O. Duda, P. E. Harta, and D. H. Stork. *Pattern Classification*. Wiley–IEEE, 2nd edition, 2000.

[26] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Siggraph*, volume 29, pages 173–182, 1995.

[27] R. Eustice, O. Pizarro, and H. Singh. Visually augmented navigation in an unstructured environment using a delayed state history. In *IEEE International Conference on Robotics and Automation*, pages 25–32, 2004.

[28] R. Eustice, H. Singh, J. Leonard, M. Walter, and R. Ballard. Visually navigating the RMS Titanic with SLAM information filters. In *Robotics Science and Systems*, pages 57–64, 2005.

[29] R. M. Eustice. *Large-Area Visually Augmented Navigation for Autonomous Underwater Vehicles*. PhD thesis, Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, 2005.

[30] R. M. Eustice, H. Singh, and J. J. Leonard. Exactly Sparse Delayed-State Filters for View-Based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, 2006.

[31] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:485—-508, 1988.

[32] O. D. Faugeras and Q. T. Luong. Camera self-calibration: theory and experiments. *Lecture Notes in Computer Science*, 588:321–334, 1992.

[33] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, volume 24, pages 381–395, September 1981.

[34] R. C. Fisher. The use of multiple measurements in Taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[35] A. W. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision*, pages 311–326, 1998.

[36] S. Fleischer. *Bounded-error vision-based navigation of autonomous underwater vehicles.* PhD thesis, Stanford University, 2000.

[37] S. D. Fleischer. *Bounded-Error Vision-Based Navigation of Autonomous Underwater Vehicles.* PhD thesis, Department of Aeronautics and Astronautics, Stanford University, 2000.

[38] R. Garcia, X. Cufi, and J. Batlle. Detection of matchings in a sequence of underwater images through texture analysis. In Greece Thessaloniki, editor, *IEEE International Conference on Image Processing*, pages 361–364, 2001.

[39] R. Garcia, X. Cufi, and M. Carreras. Estimating the motion of an underwater robot from a monocular image sequence. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1682–1687, Maui, Hawaii, 2001.

[40] R. Garcia, X. Cufí, and V. Ila. Recovering Camera Motion in a Sequence of Underwater Images through Mosaicking. *Lecture Notes in Computer Science*, 2652:255–262, September 2003.

[41] R. Garcia, X. Cufí, R. Prados, A. Elibol, J. Ferrer, M. Villanueva, and T. Nicosevici. Georeferenced Photo-Mosaicing of the Seafloor. *Instrumentation View-Point Journal*, 4:45–46, 2005.

[42] R. Garcia, T. Nicosevici, and X. Cufi. On the way to solve lighting problems in underwater imaging. In *MTS/IEEE OCEANS Conference*, pages 1018–1024, Biloxi, Mississipi,, 2002.

[43] R. Garcia, T. Nicosevici, P. Ridao, and D. Ribas. Towards a Real-Time Vision-Based Navigation System for a Small-Class UUV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, October 2003.

[44] R. Garcia, J. Puig, P. Ridao, and X. Cufí. Augmented state Kalman filtering for AUV navigation. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 4010–4015, Washington D.C., May 2002.

[45] M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proc. Visualization*, pages 99–188, 1998.

[46] C. Gonzalez, P. Castello, and M. Chover. A texture-based metric extension for simplification methods. In *Proc. GRAPP*, pages 69–76, 2007.

[47] Google Earth. `http://earth.google.com/`.

[48] Google Images. `http://images.google.com/`.

[49] N. Gracias. *Mosaic-based Visual Navigation for Autonomous Underwater Vehicles*. PhD thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisbon, Portugal, 2002.

[50] N. Gracias and J. Santos-Victor. Underwater Video Mosaics as Visual Navigation Maps. *Computer Vision and Image Understanding*, 79(1):66–91, 2000.

[51] N. Gracias, S. van der Zwaan, A. Bernardino, and J. Santos-Victor. Mosaic Based Navigation for Autonomous Underwater Vehicles. *IEEE Journal of Oceanic Engineering*, 28(3):609–624, October 2003.

[52] I. S. Gradshteyn and I. M. Ryzhik. *Tables of Integrals, Series, and Products*. Academic Press, 7th edition, 2007.

[53] M. Han and T. Kanade. Reconstruction of a scene with multiple linearly moving objects. In *International Journal of Computer Vision*, pages 285–300, 2004.

[54] R. Haralick. Propagating covariance in computer vision. *Pattern Recognition*, 1:493–498, 1994.

[55] C. G. Harris and M. J. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, Manchester, U.K., 1988.

[56] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[57] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1106–1112, 1997.

[58] A. Heyden, R. Berthilsson, and G. Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image and Vision Computing*, 17(13):981–991, November 1999.

[59] H. Hoppe. Progressive meshes. In *Siggraph*, volume 30, pages 99–188, 1996.

[60] H. Hoppe. View-dependent refinement of progressive meshes. In *Siggraph*, volume 31, pages 189–198, 1997.

[61] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, August 1981.

[62] T. S. Huang and O. D. Faugeras. Some propreties of the e-matrix in two-view motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1310–1312, 1989.

[63] Nav N Go. `http://www.navngo.com/`.

[64] V. Ila, J. Andrade-Cetto, R. Valencia, and A: Sanfeliu. Vision-based loop closing for delayed state robot mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3892–3897, 2007.

[65] J. S. Jaffe, K. D. Moore, J. McLean, and M. P. Strand. Underwater Optical Imaging: Status and Prospects. *Oceanography*, 14(3):66–76, July 2002.

[66] I. Jung and S. Lacroix. Simultaneous localization and mapping with sterevision. In *The International Symposium on Robotics Research*, pages 315–324, 2003.

[67] K. I. Kanatani. Structure from motion without correspondence: General principle. In *DARPA Image Understanding Workshop*, pages 107–116, 1985.

[68] P. M. Kelly. An algorithm for merging hyperellipsoidal clusters. Technical report, Los Alamos National Laboratory, 1994.

[69] E. F. Krause. *Taxicab Geometry*. Dover Publications, 2004.

[70] B. J. A. Kroese, N. A. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appearance-based robot localization. In *Image and Vision Computing*, volume 6, pages 381–391, 2001.

[71] L. Lam, S. Lee, and Y. Suen. Thinning methodologies – a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.

[72] P. Lamon, I. Nourbakhsh, B. Jensen, and R. Siegwart. Deriving and matching image fingerprint sequences for mobile robot localization. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1609–1614, 2001.

[73] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European Conference on Machine Learning*, pages 4–15, 1998.

[74] M. Lhuillier Q. Long. Match Propagation for Image-Based Modeling and Rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1140–1146, 2002.

[75] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:122–135, 1981.

[76] M. I. A. Lourakis and A. A. Argyros. SBA: A software package for generic sparse bundle adjustment. *Transactions on Mathematical Software*, 36(1), 2009.

[77] M. I. A. Lourakis and R. Deriche. Camera self-calibration using the singular value decomposition of the fundamental matrix: From point correspondences to 3D measurements. In *Asian Conference on Computer Vision*, pages 403–408, 2000.

[78] K. L. Low and T. S. Tan. Model simplification using vertex clustering. In *Symposium on Interactive 3D Graphics*, pages 75–82, 1997.

[79] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):90–110, 2004.

[80] B. D. Lucas. *Generalized image matching by the method of differences*. PhD thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA., 1984.

[81] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, April 1981.

[82] S. Thrun M. Montemerlo. *FastSLAM: A Scalabale Method for the simultaneous localizaton and mapping problem in robotics*, volume 27. Springer Tracts in Advanced Robotics, 2007. ISBN 978-3-540-46399-3.

[83] Y. Ma, S. Soatto, J. Kosecka, and S. Shankar Sastry. *An Invitation to 3–D Vision: From Images to Geometric Models*, volume 26. SpringerVerlag, 2003.

[84] H. Madjidi and S. Negahdaripour. On robustness and localization accuracy of optical flow computation from color imagery. In *International Symposium on 3D Data Processing, Visualization and Transmission*, pages 317–324, September 2004.

[85] H. Madjidi and S. Negahdaripour. Global alignment of sensor positions with noisy motion measurements. *IEEE Transactions on Robotics*, 21(6):1092–1104, 2005.

[86] P. C. Mahalanobis. On the generalised distance in statistics. In *National Institute of Sciences of India*, volume 2, pages 49–55, 1936.

[87] D. Martinec and T. Pajdla. 3d reconstruction by fitting low-rank matrices with missing data. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 198–205, 2005.

[88] J. Matas, O. Chum, M. Urba, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, pages 384–396, 2002.

[89] L. Matthies and S. Shafer. Error modelling in stereo navigation. *IEEE Journal of Robotics and Automation*, 3(3):239–248, 1987.

[90] G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Interscience, 2004.

[91] P. F. McLauchlan and A. Jaenicke. Image mosaicing using sequential bundle adjustment. *British Machine Vision Conference*, 20(9-10):751–759, 2002.

[92] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 1(3):63–86, January 2004.

[93] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[94] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

[95] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Artificial Intelligence*, pages 1151–1156, 2003.

[96] D. M. Mount and S. Arya. ANN: A Library for Approximate Nearest Neighbor Searching. `http://www.cs.umd.edu/~mount/ANN/`.

[97] T. Mukai and N. Ohnishi. Recovery of motion and structure from optical flow under perspective projection by solving linear simultaneous equations. In *Asian Conference on Computer Vision*, pages 583–590, 1998.

[98] T. Ebrahimi N. Aspert, D. Santa-Cruz. MESH : Measuring errors between surfaces using the hausdorff distance. In *IEEE International Conference in Multimedia and Expo*, pages 705–708, 2002.

[99] N. Nakajima, A. Iketani, and T. Sato. Video Mosaicing for Document Imaging. Technical report, Common Platform Software Research Laboratories, NEC Corporation, 2004.

[100] S. Negahdaripour. Revised Definition of Optical Flow: Integration of Radiometric and Geometric Cues for Dynamic Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(9):961–979, 1998.

[101] S. Negahdaripour and H. Madjidi. Robust optical flow estimation using underwater color images. In *MTS/IEEE OCEANS Conference*, volume 4, pages 2309–2316, San Diego, USA, September 2003.

[102] S. Negahdaripour and H. Madjidi. Robust optical flow estimation using underwater color images. In *MTS/IEEE OCEANS Conference*, volume 4, pages 2309–2316, 2003.

[103] S. Negahdaripour and H. Madjidi. Stereovision imaging on submersible platforms for 3-d mapping of benthic habitats and sea-floor structures. *IEEE Journal of Oceanic Engineering*, 28(4):625–650, 2003.

[104] S. Negahdaripour, R. Prados, and R. Garcia. Planar homography: accuracy analysis and applications. In *IEEE International Conference on Image Processing*, volume 1, pages 1089–1092, September 2005.

[105] S. Negahdaripour, X. Xu, and L. Jin. Direct estimation of motion from sea floor images for automatic station-keeping of submersible platforms. *IEEE Journal of Oceanic Engineering*, 24(3):370–382, July 1999.

[106] S. Negahdaripour, X. Xu, A. Khamene, and Z. Awan. 3-d motion and depth estimation from sea-floor images for mosaic-based station-keeping and navigation of ROVs/AUVs and high-resolution sea-floor mapping. In *Workshop on Autonomous Underwater Vehicles AUV*, pages 191–200, 1998.

[107] T. Nicosevici, R. Garcia, S. Negahdaripour, M. Kudzinava, and J. Ferrer. Identification of suitable interest points using geometric photometric cues in motion video for efficient 3-D environmental modeling. In *IEEE International Conference on Robotics and Automation*, pages 4969–4974, 2007.

[108] T. Nicosevici, S. Negahdaripour, and R. Garcia. Monocular-based 3-D seafloor reconstruction and ortho-mosaicing by piecewise planar representation. In *MTS/IEEE OCEANS Conference*, volume 2, pages 1279–1286, 2005.

[109] J. Oliensis. A multi-frame structure-from-motion algorithm under perspective projection. *International Journal of Computer Vision*, 34:163–192, 1999.

[110] A. Opelt, A. Fussenegger, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. *Lecture Notes in Computer Science*, 3022:71–84, 2004.

[111] K. Park, H. Kima, M. Baek, and C. D. Kee. Multi-Range approach of stereo vision for mobile robot navigation in uncertain environments. *Journal of Mechanical Science and Technology*, 17(10):1411–1422, 2003.

[112] L. M. Paz, P. Pinies, J. D. Tardos, and J. Neira. Large-scale 6-dof slam with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957, 2008.

[113] O. Pizarro. *Large scale structure from motion for autonomous underwater vehicle surveys.* PhD thesis, Massachusetts Institute of Technology, 2004.

[114] O. Pizarro, R. Eustice, and H. Singh. Large area 3D reconstructions from underwater surveys. In *MTS/IEEE OCEANS Conference*, volume 2, pages 678–687, November 2004.

[115] O. Pizarro and H. Singh. Toward large-area mosaicing for underwater scientific applications. *IEEE Journal of Oceanic Engineering*, 28(4):651–672, October 2003.

[116] POV-Ray – Persistence of Vision Raytracer. `http://www.povray.org/`.

[117] F. T. Ramos, B. Upcroft, S. Kumar, and H F. Durrant-Whyte. A probabilistic model for appearance-based robot localization. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2005.

[118] B. S. Reddy and B. N. Chatterji. An FFT-based technique for translation, rotation, and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(8):1266–1271, August 1996.

[119] K. Richmond and S. M. Rock. An operational real-time large-scale visual mosaicking and navigation system. In *MTS/IEEE OCEANS Conference*, Boston, USA, September 2006.

[120] L. Robert and O. D. Faugeras. Relative 3D positioning and 3D convex hull computation from a weakly calibrated stereo pair. In *International Conference on Computer Vision*, pages 540–544, 1993.

[121] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. *Geometric Modeling in Computer Graphics*, pages 455–465, 1993.

[122] P. J. Rousseeuw. Least median of squares regression. *Journal of The American Statistical Association*, 79:871–880, 1984.

[123] Y. Rzhanov, L. Huff, and G. R. Cutter. Seafloor video mapping: modeling, algorithms, apparatus. In *IEEE International Conference on Image Processing*, volume 1, pages 868–871, 2002.

[124] Y. Rzhanov, L. Mayer, S. Beaulieu, T. Shank, S. A. Soule, and D. J. Fornari. Deep-sea Geo-referenced Video Mosaics. In *MTS/IEEE OCEANS Conference*, volume 4, pages 2319–2324, 2006.

[125] J. Salvi, X Armangué, and J. Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1617–1635, July 2002.

[126] P. D. Sampson. Fitting conic sections to 'very scattered' data: An iterative refinement of the Bookstein algorithm. *Computer Vision, Graphics and Image Processing*, 18:97—-108, 1982.

[127] H. S. Sawhney, S. C. Hsu, and R. Kumar. Robust video mosaicing through topology inference and local to global alignment. In *European Conference on Computer Vision*, volume II, pages 103–119, Freiburg, Germany, June 1998.

[128] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps". In *European Conference on Computer Vision*, pages 414–431, 2002.

[129] W. Schroeder, J. Zarge, and W. Lorenson. Decimation of triangle meshes. In *Siggraph*, volume 26, pages 65–70, 1992.

[130] Second Life. `http://secondlife.com/`.

[131] P. Shirley and R. K. Morleys. *Realistic Ray Tracing.* A. K. Peters, Ltd., second edition, 2003.

[132] H. Singh, J. Howland, and O. Pizarro. Advances in large-area photomosaicking underwater. *IEEE Journal of Oceanic Engineering*, 29:872–886, July 2004.

[133] H. Singh, C. Roman, O. Pizarro, R. M. Eustice, and A. Can. Towards High-resolution Imaging from Underwater Vehicles. *International Journal of Robotics Research*, 26(1):55–74, 2007.

[134] J. Sivic. *Efficient visual search of images and videos.* PhD thesis, University of Oxford, 2006.

[135] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80:189–210, 2008.

[136] M. E. Spetsakis and J. Y. Aloimonos. A multiframe approach to visual motion perception. *International Journal of Computer Vision*, 6:245–255, 1991.

[137] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 845–853, 1996.

[138] R. Szeliski. Image mosaicing for tele-reality applications. In *IEEE Workshop on Applications of Computer Vision*, pages 44–53, September 1994.

[139] R. Szeliski and H. Y. Shum. Creating full view panoramic image mosaics and environment maps. In *International Conference on Computer Graphics and Interactive Techniques*, pages 251–258, September 1997.

[140] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization approach. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[141] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In *International Conference on Computer Vision*, volume 1, pages 273–280, 2003.

[142] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment–a modern synthesis. In *International workshop on vision algorithms*, pages 298–372, 1999.

[143] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *International Conference on Computer Vision*, volume I, pages 298–372, Corfu, Greece, September 1999.

[144] T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *British Machine Vision Conference*, pages 736–739, Bristol, UK, 2000.

[145] M. Vergauwen and L. Van Gool. Web-based 3D reconstruction. *Machine Vision and Applications*, 17:321–329, 2006.

[146] J. Wang, R. Cipolla, and H. Zha. Vision-based Global Localization Using a Visual Vocabulary. In *IEEE International Conference on Robotics and Automation*, pages 4230–4235, 2005.

[147] J. Wills, S. Agarwal, and Serge Belongie. A Feature-based Approach for Dense Segmentation and Estimation of Large Disparity Motion. *International Journal of Computer Vision*, 68:125–143, 2006.

[148] G. Wolberg and S. Zokai. Robust image registration using log-polar transform. In *IEEE International Conference on Image Processing*, volume 1, pages 493–496, 2000.

[149] J. Wu, Z. Xu, and P. Shi. A new algorithm of sub-pixels image matching. *High Technology Letters*, 10(1):22–25, 2004.

[150] A. Xu, S. Sun, and K. Xu. Texture information driven triangle mesh simplification. In *Computer Graphics and Imaging*, pages 43–48, 2005.

[151] H. Zhang and S. Negahdaripour. On reconstruction of 3D volumetric models of reefs and benthic structures from image sequences of a stereo rig. In *MTS/IEEE OCEANS Conference*, volume 5, pages 2553–2559, San Diego, USA, September 2003.

[152] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classifcation of texture and object categories: An in-depth study. Technical report, INRIA, 2005.

[153] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[154] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision*, pages 666–673, 1999.

[155] K. Zuiderveld. Contrast limited adaptive histogram equalization. *Graphics gems IV*, pages 474–485, 1994.