

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ESAT
AFDELING PSI
Kasteelpark Arenberg 10 — 3001 Heverlee, Belgium

HARNESSING STRUCTURE-AND-MOTION

Promotor:
Prof. L. VAN GOOL

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de Ingenieurswetenschappen

door

Maarten VERGAUWEN

2006



KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT ESAT
AFDELING PSI
Kasteelpark Arenberg 10 — 3001 Heverlee, Belgium

HARNESSING STRUCTURE-AND-MOTION

Jury:
Prof. P. Van Houtte, voorzitter
Prof. L. Van Gool, promotor
Prof. L. Van Eycken, assessor
Prof. P. Wambacq, assessor
Prof. Y. Willems, assessor
Prof. T. Moons (K.U.Brussel)
Prof. P. Cignoni(Univ. of Pisa, Italy)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de Ingenieurswetenschappen

door

Maarten VERGAUWEN

U.D.C. 681.3*I4

2006

©Katholieke Universiteit Leuven - Faculteit Ingenieurswetenschappen
Arenbergkasteel, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en /of openbaar gemaakt worden door middel van druk, fotocopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

Wettelijk depot D/2006/7515/76

ISBN 90-5682-742-1

Voorwoord

Dat mijn doctoraat uiteindelijk het daglicht heeft gezien, is te danken aan vele mensen. In de wetenschap dat ik ze nooit allemaal kan vermelden en altijd wel iemand zal vergeten, wil ik hier toch een poging wagen.

Om te beginnen wil ik prof. Van Gool bedanken. Hij bood me zoveel jaren geleden de mogelijkheid om als assistent in zijn groep onderzoek te doen. Ik wil hem bedanken voor zijn vertrouwen en de kansen die ik kreeg, alsook voor dat beetje extra druk om de tekst te schrijven. Soms heeft een mens een duwtje nodig.

Ik ben prof. Van Eycken, prof. Wambacq en prof. Willems van mijn leescomité en prof. Moons van de jury dank verschuldigd voor het nauwgezet nalezen van mijn doctoraatstekst en voor de vele geïnteresseerde vragen die ze me stelden.

Prof. Cignoni, Paolo, thank you very much for accepting to be a jury member. We have collaborated in the Epoch project and I hope we'll continue this fruitful collaboration in the future.

Ik heb me altijd thuis gevoeld in onze VISICS onderzoeksgroep. Vele resultaten die in dit werk worden beschreven zijn tot stand gekomen door teamwerk. Ik wil daarom graag Frank Verbiest, Geert Willems, Bert Deknuydt, Kurt Cornelis, Egemen Özden, Tinne Tuytelaars en alle andere huidige en vroegere VISICS collega's bedanken voor de prettige samenwerking.

Dankjewel mama en papa, Kwinten, Karen, Lukas en peter Wout, Herman en Moniek, Leen, Roel, Sammeman en Fienemie. Ook mijn vrienden van Cantico wil ik graag vermelden. Jullie vriendschap is een stimulans en jullie interesse in m'n onderzoek dwingt me om er op een begrijpelijke manier over te communiceren.

Tenslotte gaat mijn dank uit naar mijn Elsje. Dankjewel voor je steun, voor je interesse en voor je aandringen om aan deze tekst te beginnen en vol te houden.

Abstract

This dissertation deals with the problem of recovering the camera calibration and the 3D reconstruction of a scene from uncalibrated images of this scene, usually referred to as *Structure and Motion* recovery or SaM. This work gives an overview of SaM theory and techniques. It discusses several design choices, implementation issues and algorithmic improvements to arrive at a robust and efficient system that can deal with many problems and issues that typically arise. Many real-life examples are given as well as applications in a wide gamut of fields: from laboratory conditions, over cultural heritage applications to planetary exploration.

In deze verhandeling wordt het probleem bestudeerd van het berekenen van de camera calibratie en 3D reconstructie van een scène uit ongecalibreerde beelden van die scène, een techniek die *Structure and Motion* of SaM wordt genoemd. In dit werk wordt een overzicht gegeven van de theorie achter SaM en van belangrijke technieken die ervoor nodig zijn. Verschillende ontwerpkeuzes worden besproken, alsook implementatie aspecten en algoritmische verbeteringen om te komen tot een robuust en efficiënt systeem dat vele problemen en speciale gevallen aankan die regelmatig voorkomen. Vele voorbeelden uit de praktijk worden getoond die komen uit een brede waaier van toepassingsgebieden: van laboratorium omstandigheden, over cultureel erfgoed tot planetaire exploratie.

Notations

To enhance the readability some notations and naming conventions used throughout the text are shortly summarized here.

For matrices bold face fonts and capitals are used (e.g. $\mathbf{A}$). Vectors are represented by lowercase bold face fonts ($\mathbf{t}$). Two-dimensional geometrical entities like 2D points and lines are represented by lowercase type face fonts ($\mathtt{m}$, $\mathtt{l}$). Three-dimensional entities like 3D points and planes are represented by capital type face fonts ($\mathtt{M}$, $\mathtt{\Pi}$). Scalar values will be represented as a. $\mathbf{A_{ij}}$ refers to the element of matrix $\mathbf{A}$ on the ith row and jth column.

$\simeq$	equivalence up to scale ($A \simeq B \Leftrightarrow \exists \lambda \neq 0 : A = \lambda B$)
$\mathbf{A}^T$	transpose of matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	inverse of square matrix $\mathbf{A}$ (i.e. $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$)
$\mathbf{A}^{\dagger}$	pseudo-inverse of non-square matrix $\mathbf{A}$ (i.e. $\mathbf{A}^{\dagger} = \left(\mathbf{A}^T A\right)^{-1} \mathbf{A}^T$)
$[\mathbf{a}]_{\times}$	skew-symmetric matrix (3×3) of the vector $\mathbf{a}$

Moreover, the following symbols are reserved for the following entities:

$\mathtt{M}$	world point (4-vector)
$\mathtt{m}$	image point (3-vector)
$\mathbf{H}$	homography (3×3 matrix)
$\mathbf{F}$	fundamental matrix (3×3 rank 2 matrix)
$\mathbf{E}$	essential matrix (3×3 rank 2 matrix)
$\mathtt{e}$	epipole
$\mathbf{K}$	calibration matrix (3×3 upper triangular matrix)
$\mathbf{R}$	rotation matrix
$\mathtt{t}$	translation vector
$\mathbf{P}$	projection matrix (3×4 matrix)
$\mathbf{\Omega}^*$	dual absolute quadric (4×4 rank 3 matrix)
ω^*	image of the dual absolute quadric (3×3 matrix)
$\mathbf{I}$	the identity matrix
$I(x, y)$	image intensity (in case of color images: either of the three colorbands R, G or B)

Contents

1 Introduction **1**
- 1.1 Main Contributions . . . 1
- 1.2 SaM Applications . . . 2
- 1.3 Outline of the Thesis . . . 3

2 Calibration **5**
- 2.1 Camera Model . . . 5
  - 2.1.1 Pinhole Camera . . . 5
  - 2.1.2 Decomposition of P . . . 6
  - 2.1.3 Non-Linear Distortions . . . 6
- 2.2 Calibration . . . 7
  - 2.2.1 Internal Calibration . . . 7
  - 2.2.2 External Calibration . . . 9
- 2.3 A Stereo Setup . . . 11
  - 2.3.1 The Teleman Stereo Head . . . 13

3 Structure And Motion **15**
- 3.1 The Geometry of Two Images . . . 15
  - 3.1.1 Epipolar Geometry . . . 15
  - 3.1.2 The Fundamental Matrix . . . 17
  - 3.1.3 Computation of the Fundamental Matrix . . . 17
- 3.2 Features . . . 21
  - 3.2.1 Simple Interest Points . . . 21
  - 3.2.2 Affine Invariant Features . . . 24
  - 3.2.3 SIFT and SURF Features . . . 25
  - 3.2.4 Matching Point Configurations . . . 26
- 3.3 RANSAC . . . 27
  - 3.3.1 Dealing with Outliers: the RANSAC Algorithm . . . 27
  - 3.3.2 F-Guided Matching . . . 30
- 3.4 Computing Homographies . . . 31
  - 3.4.1 Computing Homographies with RANSAC . . . 31
  - 3.4.2 Layered 3D Reconstruction . . . 32
- 3.5 Multiple View Geometry . . . 33

3.6 Structure and Motion . . . 34
3.6.1 Initialization Step . . . 34
3.6.2 Projective Pose Estimation . . . 38
3.6.3 Updating Structure . . . 40
3.6.4 Global Minimization . . . 40
3.7 The Projective Ambiguity . . . 42
3.8 Scene Constraints . . . 45
3.9 Self-Calibration . . . 46
3.9.1 The Absolute Conic . . . 46
3.9.2 Practical Constraints . . . 48
3.9.3 Coupled Self-Calibration . . . 50
3.10 Multi-View Camera Calibration . . . 51
3.10.1 Introduction . . . 51
3.10.2 Setup . . . 51
3.10.3 Point Detection . . . 53
3.10.4 Calibration Algorithm . . . 53
3.10.5 Results . . . 54
3.11 Conclusion . . . 56

4 Model Selection **61**
4.1 Introduction . . . 61
4.2 Problems with Planar Scenes . . . 61
4.3 Detecting Planar Scenes . . . 62
4.3.1 GRIC . . . 64
4.4 More GRICs . . . 65
4.5 Long Video Sequences . . . 69
4.5.1 Video Frames . . . 69
4.5.2 Long Sequences and Subsequences . . . 72
4.5.3 Intermediate Frames . . . 75
4.5.4 Blurry Frames . . . 76
4.6 Essential Matrices . . . 78
4.6.1 The Essential Matrix . . . 79
4.6.2 Cameras from the Essential Matrix . . . 80
4.6.3 Computation of the Essential Matrix . . . 81
4.6.4 Planar Degeneracy . . . 84
4.7 Dense Matching . . . 86
4.7.1 Rectification . . . 86
4.7.2 Stereo Matching . . . 91
4.7.3 Linking Stereo Pairs . . . 96
4.8 Conclusion . . . 98

5 SaM and Space Robotics 99
5.1 Introduction . . . 99
5.2 Viable Setup . . . 99
5.2.1 The ETS-VII Satellite . . . 99
5.2.2 Station Setup . . . 101
5.2.3 VBRC Experiments . . . 102
5.2.4 Enhancing the Image Quality . . . 103
5.3 Calibration Experiments . . . 104
5.3.1 Invariant-Based Feature Extraction . . . 104
5.3.2 Eye-hand Calibration . . . 106
5.4 Pose Estimation Experiments . . . 107
5.4.1 Calculating Pose from Known Markers . . . 107
5.4.2 Insertion of GPF Into a Hole . . . 109
5.5 Taskboard Reconstruction . . . 110
5.5.1 Calibration of 3-Point Markers . . . 110
5.5.2 Taskboard Reconstruction . . . 112
5.6 Robot Calibration Experiments . . . 113
5.6.1 Robot Calibration . . . 113
5.6.2 Repeatability Study . . . 116
5.7 Conclusion . . . 116

6 SaM and Planetary Exploration 119
6.1 Introduction . . . 119
6.2 Rover Mission Study . . . 120
6.2.1 System Setup . . . 120
6.2.2 Utilization . . . 122
6.3 Calibration . . . 123
6.3.1 Using Markers? . . . 123
6.3.2 Strategy . . . 124
6.3.3 Relative Calibration . . . 124
6.3.4 Pan-Tilt Calibration . . . 125
6.3.5 Synthetic Calibration Experiment . . . 129
6.4 3D Terrain Modeling . . . 129
6.4.1 Disparity Map Generation . . . 129
6.4.2 Digital Elevation Map . . . 130
6.5 System Test . . . 132
6.5.1 Calibration and 3D Modeling . . . 132
6.5.2 Simulator . . . 134
6.5.3 Results . . . 136
6.6 Aerobots . . . 137
6.6.1 Introduction . . . 137
6.6.2 Aerobot Systems . . . 137
6.7 Overall Aerobot Software System . . . 138
6.8 ILP and 3D Reconstruction . . . 139
6.8.1 ILP Overlap . . . 141

6.8.2 ILP Structure And Motion . . . 142
6.8.3 ILP Dense Matching . . . 143
6.9 ILP and Storage . . . 144
6.9.1 Image Richness Index . . . 144
6.9.2 DEM Combination and Compression . . . 144
6.10 Tests Of ILP with Simulated Images . . . 145
6.10.1 Authoring Test Cases . . . 146
6.10.2 Running Simulated Test Cases . . . 149
6.10.3 Results . . . 150
6.11 Tests of ILP with Real Images . . . 154
6.11.1 Images from a Video . . . 154
6.11.2 Aerobot Hardware . . . 155
6.11.3 Images from the Digital Camera . . . 157
6.12 Conclusion . . . 160

7 SaM and Cultural Heritage 163
7.1 Introduction . . . 163
7.2 Timeline into the present . . . 164
7.2.1 Combining Image and Model Based Rendering of an Archaeological Site . . . 164
7.2.2 Acquisition of Imagery . . . 168
7.2.3 Grouping of Images . . . 169
7.2.4 Structure and Motion and Dense 3D . . . 170
7.2.5 Registration . . . 172
7.2.6 Image-Based Rendering . . . 175
7.2.7 Refining cameras . . . 176
7.2.8 Results . . . 177
7.3 3D Webservice . . . 178
7.3.1 System Overview . . . 182
7.3.2 Automatic Reconstruction Pipeline . . . 188
7.3.3 Global Image Comparison . . . 191
7.3.4 Self-Calibration . . . 192
7.3.5 Reconstruction . . . 194
7.3.6 Dense Matching . . . 199
7.3.7 Results . . . 201
7.4 Conclusion . . . 203

8 Conclusion 209
8.1 Summary . . . 209
8.2 Discussion and Future Work . . . 211

A Passive 3D Reconstruction: Basics 225
A.1 Introduction . . . 225
A.2 Homogeneous Coordinates . . . 225
A.2.1 Points . . . 225

A.2.2 Infinity 226
A.2.3 Lines 226
A.3 Camera Models 228
A.3.1 Image Coordinates 228
A.3.2 The Linear Pinhole Model 229
A.3.3 Non-Linear Distortions 233
A.4 Deformation of Planar and 3D Objects 235
A.4.1 Deformation of Planar Objects under Projection 235
A.4.2 Stratification of Transformations of 3D Objects 236
A.4.3 Transformations and the Projection Equation 238

Chapter 1

Introduction

Mankind in its continuous strive to improve its fate has learned early that information about the world that surrounds us is vital. Throughout the ages, systems have been developed that can acquire precise measurements of objects and scenes in this world. The industrial and modern age brought more complexity to these systems. If one wants to acquire 3D positional measurements, for instance, many systems are at one's disposal: one can buy active systems like lasers, based on time-of-flight or triangulation principles, or an apparatus using structured light. Passive systems are available as well, like calibrated stereo rigs or photometric stereo setups.

The fact that 3D measurements can be obtained from images taken from different positions has been known since the nineteenth century. The calibrated stereo rigs we spoke of are based on this principle. As a first extension to this technique, the setup with two cameras can be replaced by one where only a single camera is needed which moves and observes a static scene. The structure of the scene can be recovered from correspondences if the camera calibration and its motion are known, hence the name *Structure from Motion*.

Over the last years computer vision researchers have taken the idea a step further. What if nothing is known about the moving camera? No calibration of the internal parameters or of the camera's motion is available. It turns out it is possible to employ the images this camera takes to recover both the structure of the scene and the calibration (internal and external) of the recording camera. This explains the name of these techniques, commonly referred to as *Structure and Motion*.

1.1 Main Contributions

The theory behind Structure and Motion has been developed over many years. A dedicated researcher can implement SaM algorithms using the many publications that exist in literature. In practice, however, a straightforward implemen-

tation of the theory will only yield satisfying results in well chosen, relatively easy to process situations. It has been the goal of this work to broaden the applicability of SaM algorithms. In this section we will summarize the main contributions we believe have been realized in this work.

- **Longer sequences**: if one wants to process video sequences of several minutes, the amount of frames increases very quickly. The nature of video sequences with frames taken from viewpoints that are close together also gives rise to a need for other strategies than simple processing of consecutive frames with a standard SaM pipeline. Both enhancements are needed to deal with casually rather than purposely taken video data.
- **Larger baselines**: consecutive images taken by non-experts are often still quite far apart. Sometimes difficult recording conditions simply disallow to take pictures that are close together. Dealing with such large baseline conditions calls for other matching strategies, using e.g. more invariant features.
- **Degenerate cases**: standard SaM techniques fail for degenerate camera motions or scenes. Unfortunately, these are common. Examples are cameras moving on a straight line or on a perfect circle or the observation of a planar scene. We have developed strategies to cope with these degenerate cases, making use of a-priori information or of non-degenerate parts of the recorded data.
- **Non-sequential imagery**: SaM was and still is mostly used to process sequential imagery. But several applications can be envisaged where images are recorded non-sequentially. Then images need to be related with other images than just their predecessor. In order to improve the calibration, the same is true if the camera moves in a zigzag manner.

1.2 SaM Applications

As will become clear in the last three chapters of this thesis, we have applied SaM techniques in a wide gamut of application fields: from laboratory conditions over cultural heritage applications to space robots and planetary exploration. The way the images are recorded and the image content itself differ from field to field, giving rise to specific algorithms. We have opted for an eclectic approach where several recent developments, also by other researchers, are incorporated into the SaM pipeline to arrive at the necessary robustness and flexibility. Below is a list of projects and applications in which SaM techniques were used. For each of them we explain the specific problems one encounters if straightforward SaM is employed and the improvements we had to implement. For more details we refer to the relevant chapters.

The European IST projects *Attest* and *Inview* made heavy use of SaM. In the former, the consortium studied the feasibility of 3D-TV, including the upgrade

of existing 2D material to 3D. In the latter an image-based rendering system was developed that could be fed with images from both laboratory and hand-held camera footage. For both projects, long video sequences of several minutes needed to be processed. In *Inview*, several recordings were made in a zigzag fashion, requiring non-sequential image matching for better calibration.

In our lab we built a multi camera setup. Calibration of such a system can be tedious, so a user-friendly calibration approach was developed. Using a laser pointer in dark conditions, detection of features and matches is straightforward. The calibration of the cameras from these matches is clearly a non-sequential problem in which matches with all already reconstructed cameras are used to calibrate additional cameras.

Our lab has been active in many projects with the European Space Agency. *Viable* was a joint project with the Japanese Agency, executed on a Japanese satellite. SaM algorithms were used here to guide the robot on-board this satellite in order to execute several tasks or to improve its calibration. Several projects dealt with vision and planetary exploration. The availability of images from different positions, makes it possible to compute 3D information of the planetary surface. This enables scientists to plan rover missions on Earth and simulate the actual motion from telemetry, as was shown in the *Robust* project. In *Aerobots* SaM techniques were used to reconstruct cameras and depth maps from images on board a flying platform. These planetary vision systems must be as robust as possible. Degenerate cases, like planar scenes, will be encountered often and must be dealt with.

Several applications and examples come from the cultural heritage field. For the archeological museum of Ename, for instance, we created an extra layer for an existing QT-VR movie. For this we had to deal with degenerate cases and non-sequential imagery. In the scope of the *Epoch* project, we developed a 3D webservice, allowing cultural heritage professionals to upload images to a server at ESAT which computes a 3D reconstruction. Because one has no control on which kind of imagery people upload, we had to dig deep in our bag of tricks to make SaM überhaupt possible. One can expect large baseline conditions, non-sequential imagery and recordings with degenerate parts.

Although not elaborated on anywhere else in this work, we like to mention that our work has been used by our group's spin-off company Eyetronics in two major Hollywood film productions. For *King Arthur* [26], a valley in the Irish countryside was recorded and reconstructed in 3D in order to construct a virtual frozen lake, used in several seconds of footage. In *The Brothers Grimm* [29], pictures of a well were used to reconstruct it in 3D to be used as a depth cue for better visual effects, incorporating a small monster climbing out of the well.

1.3 Outline of the Thesis

SaM algorithms aim at recovering both the camera parameters and the scene geometry. It is therefore imperative to have a good understanding of camera

models and their calibration. Chapter 2 gives a very brief overview of the camera model we employ in this work and describes some calibration algorithms that can be used to recover the camera parameters with specific calibration techniques. A more thorough explanation of the camera model and a stratification of 2D and 3D geometry can be found in appendix A.

In chapter 3 we describe the standard, sequential SaM pipeline. The epipolar geometry is computed for sequential image pairs using different features. The RANSAC algorithm and its improvements are explained, as well as a direct application in the form of the combination of epipolar geometry and homographies yielding layered 3D reconstructions. Afterwards projective 3D reconstructions are obtained that can be upgraded to metric using self-calibration algorithms. This chapter ends with the description of a multi-view camera calibration system, for which both real and artificial results are given.

Chapter 4 first introduces degenerate cases and model selection algorithms to detect these. The application that deals with long video sequences is directly related to this. Afterwards, essential matrices and a stable algorithm to recover them are described. We end this chapter with dense matching algorithms that yield dense 3D models of the scene.

Chapter 5 is the first chapter that deals with a specific application of SaM algorithms. It describes the work that was done on the Japanese satellite ETS-VII. The vision algorithms mostly focus on calibration because of the nature of the project which targeted on space robotics.

In chapter 6 we employ SaM techniques in planetary exploration. Two ESA projects are described. The first deals with a stereo head with two degrees of freedom for which we need to recover the calibration and which we employ to reconstruct the surrounding terrain in 3D to perform planning, simulation and control of a planetary rover. The second project is about planetary aerobots that record a planet's terrain using images and reconstruct it as they go. The system setup and specific techniques to deal with this situation are described and results from three different kinds of input stimuli are shown.

Chapter 7 explains our efforts in the field of cultural heritage. First we describe a pipeline we implemented to add a contemporary layer of images to an existing QT-VR movie of an archaeological site. Image-based rendering techniques were employed in order to do so. This chapter concludes with the description of the culmination of our work on SaM algorithms: a fully automatic 3D reconstruction webservice which allows users to upload images to a server and download 3D reconstructions that are automatically generated on this server.

Finally, in chapter 8 we come to a conclusion about our work and indicate possible future work.

Chapter 2

Calibration

In this work several techniques will be described that perform calibration and 3D reconstruction from camera images. In this chapter we briefly introduce the camera model we employ. Afterwards we explain how cameras can be calibrated with standard calibration procedures using known calibration objects and we will show some concrete examples. At the end of this chapter we discuss the simplest passive 3D reconstruction setup: a stereo pair. For the sake of completeness a more thorough derivation of the camera model and a stratification of 2D and 3D geometry can be found in appendix A.

2.1 Camera Model

The camera model that will be used throughout this text is the linear pinhole model. This camera model is inspired by the *camera obscura*: a black box, one side of which is punctured to yield a small hole and with photosensitive paper on the opposite side.

2.1.1 Pinhole Camera

The mathematical set-up of the pinhole model is shown in Fig. 2.1. It shows a 3D point $\mathtt{M}$, projected in a camera with center $\mathtt{C}$. The local coordinate system of the camera is located in the world using a translation $\mathbf{t}$ and rotation $\mathbf{R}$, also referred to as the camera external parameters or extrinsics. The projection $\mathtt{m}$ of $\mathtt{M}$ in the image is the intersection of the image plane with the ray formed by connecting $\mathtt{C}$ with $\mathtt{M}$. The equation for this projection is

$$\begin{aligned} \mathtt{m} &\simeq \mathbf{K}\left(\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right)\mathtt{M} \\ &\simeq \mathbf{P}\mathtt{M} \end{aligned} \tag{2.1}$$

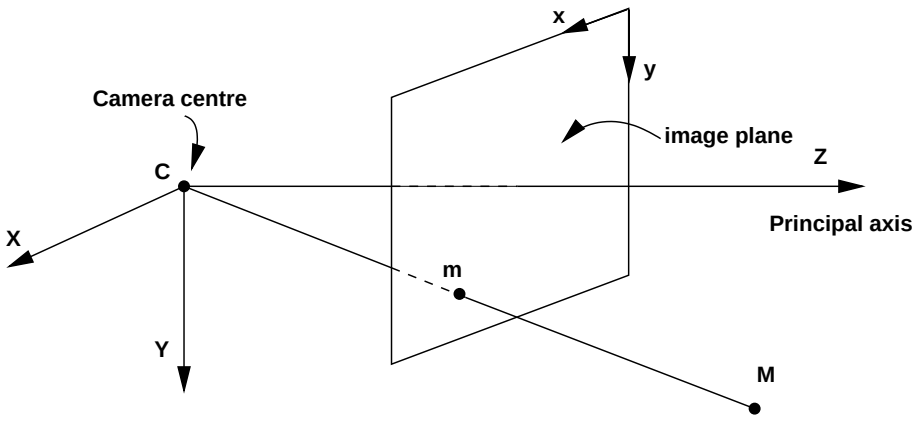


Figure 2.1: The pinhole camera model

where $\mathbf{K}$ is the matrix holding the internal parameters (or intrinsics) of the camera, namely the focal length in pixels in x and y direction α_x and α_y, the skew s and the principal point (u_x, u_y).

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & u_x \\ 0 & \alpha_y & u_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

2.1.2 Decomposition of P

If only the 3×4 projection matrix $\mathbf{P}$ is known, it is possible to retrieve the intrinsic and extrinsic parameters from it. This process is called *decomposing* $\mathbf{P}$ into $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$ and is done as follows. Inspect the upper left 3×3 sub-matrix of $\mathbf{P}$. This matrix is formed by multiplying $\mathbf{K}$ and $\mathbf{R}^T$. The inverse of this matrix is $\mathbf{R}\mathbf{K}^{-1}$. The QR-decomposition of a 3×3 matrix of rank 3 decomposes the matrix into two matrices: an orthonormal matrix on the left and an upper triangular matrix on the right. This is exactly the decomposition we are looking for since $\mathbf{R}$ is a rotation matrix and thus orthonormal and $\mathbf{K}$ is by definition an upper triangular matrix, so $\mathbf{K}^{-1}$ is as well. When $\mathbf{K}$ and $\mathbf{R}$ are known, $\mathbf{t}$ is easily computed by pre-multiplying the fourth column of $\mathbf{P}$ with $-\mathbf{R}\mathbf{K}^{-1}$.

2.1.3 Non-Linear Distortions

The projection model we described is linear which means it can only model linear effects. Images taken by real cameras experience non-linear deformations or distortions which make the simple linear pinhole model inaccurate. The most important and best known non-linear distortion is *radial distortion*. It is typically modeled using a Taylor expansion with up to 3 coefficients κ_1, κ_2, κ_3. Section A.3.3 gives a mathematical description.



Figure 2.2: Calibration objects in 3D (left) and 2D (right)

2.2 Calibration

As explained before a perspective camera can be described by its intrinsic and extrinsic parameters. The process of determining these parameters is known as *camera calibration*. We make a distinction between intrinsic or internal calibration and extrinsic or external calibration, also known as pose-estimation.

2.2.1 Internal Calibration

Several techniques have been developed to recover the intrinsic parameters of a camera. Typical calibration procedures extract the camera parameters from a set of known 3D-2D correspondences, i.e. a set of 3D coordinates with corresponding 2D image coordinates. In order to easily obtain such 3D-2D correspondences, these calibration techniques employ calibration objects, like the ones displayed in Fig 2.2. These objects have in common that they contain easily recognizable markers. A typical calibration algorithm therefore follows the following steps

- Construct a calibration object (either 3D or 2D). Measure the 3D positions of the markers on this object.
- Take one or more (depending on the algorithm) pictures of the object with the camera to be calibrated.
- Extract and identify the markers in the image.
- Fit a linearized calibration model to the 3D-2D correspondences found in the previous step.
- Improve the calibration with a non-linear optimization step.

Tsai's and Zhang's Methods

As mentioned, both 2D and 3D calibration objects can be used to perform the calibration. In his famous article Tsai [95] describes an algorithm to calibrate a camera with either 2D or 3D calibration objects. He follows the procedure explained above. Using his technique it is possible to retrieve not only the intrinsic but also the extrinsic parameters of the camera. Using the 2D calibration object, some degenerate setups can appear however, the most important one consisting of the calibration plane being parallel to the image plane.

A flexible technique has been described by Zhang [108]. A 2D calibration object is moved several times and an image is taken by the camera every time. The internal camera calibration is easily extracted from these images. Jean-Yves Bouguet has made a Matlab implementation of this algorithm available on the Internet [9].

Calibration in Action

The *Teleman* project is a robotics project of the European Space and Technology Center (ESTEC) of the European Space Agency in which I was responsible for the VISICS involvement. Part of this project was the construction and calibration of a stereo head.

The left image of Fig 2.3 shows the Graphical User Interface (GUI) of the calibration package. An image of a calibration object, taken by the left and right camera of the stereo head are shown. The calibration package first preprocesses the images with a thresholding algorithm to identify the black and white squares. This delivers the 2D coordinates of the known 3D points of the calibration object, as shown in the right image of Fig 2.3. The software then calibrates the intrinsics and extrinsics of both cameras using a two step approach: first Tsai's algorithm delivers an initialization and then a non-linear optimization minimizes the reprojection error.

Because the cameras are looking at the same calibration object, their calibration is recovered in the same coordinate system. This means that some error-statistics can be performed in 3D. Fig 2.4 shows a 3D reconstruction of the calibration object's points using the recovered calibration. Because the rendering is orthographic, the coplanar points should (and do) coincide when viewed by a camera located above the object and looking straight down. The distance between the points can be measured in 3D as well. In this particular example, we recover a mean distance of 13.39mm with a standard deviation of 0.15mm. The ground-truth distance between the points is 13.40mm.

Calibrating Radial Distortion Only

For some computer vision tasks like object recognition, complete calibration of the camera is not necessary. However, radial distortion in the images makes these tasks more difficult. That is why one sometimes only wants to remove the radial distortion in the images. In order to remove it, the parameters of

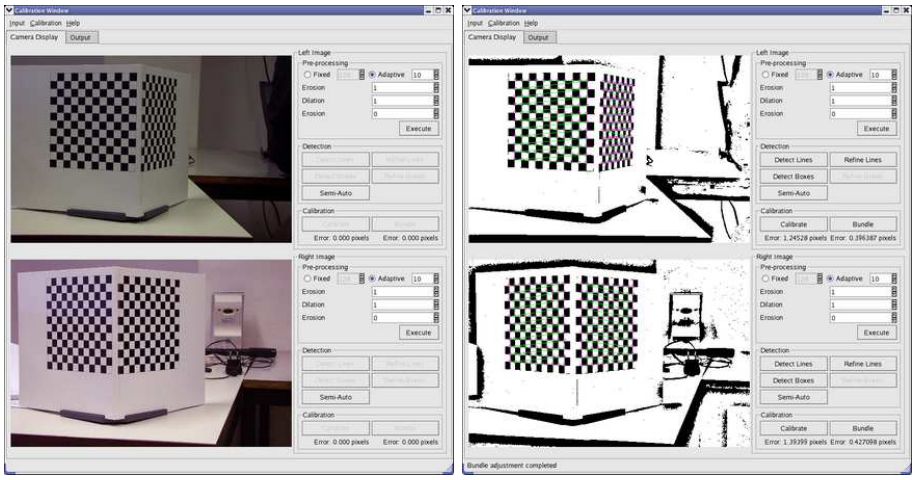


Figure 2.3: The GUI of the *Teleman* calibration package (left). The black and white squares are automatically detected and the calibration of the cameras is computed from the recovered 3D-2D correspondences.

the distortion need to be known. A simple but efficient way of retrieving the parameters without a full calibration is explained below.

Inspect Fig. 2.5. The picture on the left shows a clearly distorted image and we want to undistort it. The first step in the computation of the distortion parameters consists in the detection of edges. In this example we used the well-known Canny edge detector [11]. The user then identifies a set of curves which he recognizes as straight lines in the world but which are bent as a result of the radial distortion. These lines are most apparent near the borders of the image because there the radial distortion has more influence. For each line, points are sampled along it and a straight line is fitted through this data. If we want the distortion to vanish, the error consisting of the sum of the distances of each point to their line should be zero. Hence a non-linear minimization algorithm like Levenberg-Marquardt is applied to this data. The algorithm is initialized with the distortion parameters set to zero. At every iteration, new values are computed for these parameters, the points are warped accordingly and new lines are fitted. The algorithm stops when it converges to a solution where all selected lines are straight (i.e. the resulting error is close to zero). The resulting unwarped image can be seen on the right of Fig. 2.5.

2.2.2 External Calibration

The calibration techniques of section 2.2.1 are quite intricate because they recover the internal and sometimes the external calibration of a camera. Once a camera has been internally calibrated however, computing its external calibration is a much simpler problem. The computation of the external parameters

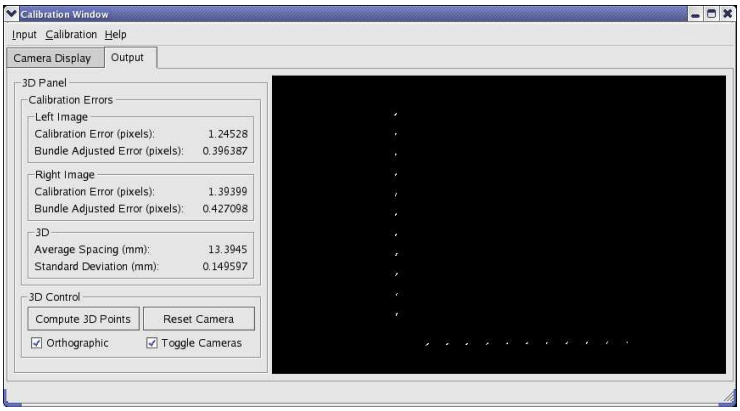


Figure 2.4: Results of the calibration: a 3D reconstruction of the calibration points is shown as well as some error-statistics.

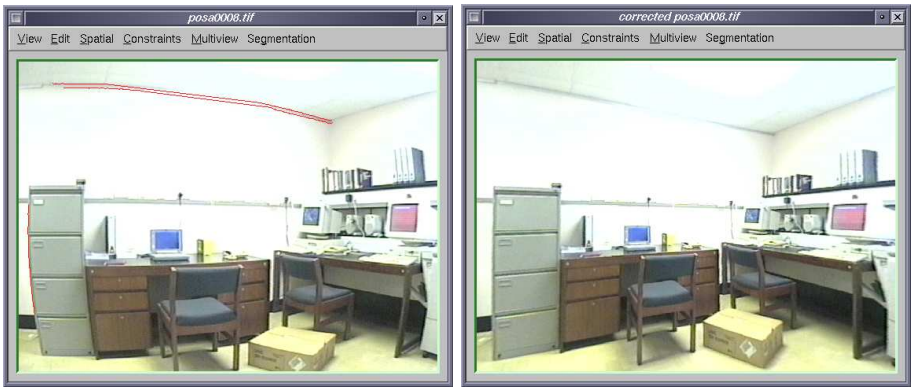


Figure 2.5: Left: distorted image with indicated curves that should be straight lines Right: undistorted image. The lines are now straight

(rotation and translation) of a camera based on a set of 3D-2D correspondences is also referred to as *Pose Estimation*. The computation of the camera-pose can be done if as few as 3 3D-2D correspondences are available. Already in 1841 Grunert [34] proposed a solution to this *Three Point Perspective Pose Estimation* problem. A review of Grunert's and other solutions can be found in [35]. Grunert proved that the problem boils down to solving for the roots of a fourth-degree polynomial, so up to 4 possible solutions can be found for the extrinsic calibration from the minimum amount of correspondences. If more than 1 real solution exist, the solutions can be checked with one extra 3D-2D correspondence to retain the correct one. This Euclidean pose estimation was used in several applications, including the Aerobot project (chapter 6) and the 3D Webservice (chapter 7). In paragraph 3.6.2 we will explain an algorithm

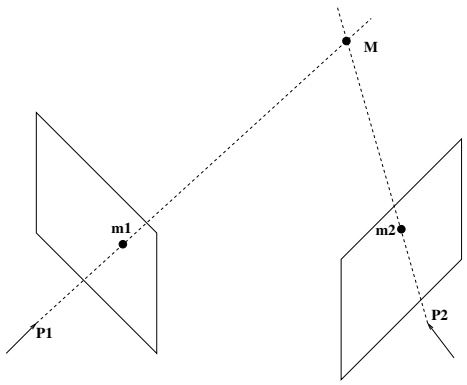


Figure 2.6: The principle behind stereo-based 3D reconstruction is very simple: if the point M is projected into two images, yielding the projections $\mathtt{m}_1$ and $\mathtt{m}_2$, then, starting from these projections, the 3D position of M is found as the intersection of the two projection rays.

that can be used to perform projective pose estimation. It can compute the projective projection matrices **P** from 6 3D-2D correspondences.

2.3 A Stereo Setup

With camera parameters and their calibration introduced, we discuss what can be done if we combine two cameras into a stereo setup. It is the simplest setup one can use to reconstruct scenes in 3D in a passive way. Suppose we have two images, taken at the same time and from different viewpoints. The situation is illustrated in Fig. 2.6. The principle behind stereo-based 3D reconstruction is simple: if the point M is projected into two images, yielding the projections $\mathtt{m}_1$ and $\mathtt{m}_2$, then, starting from these projections, the 3D position of M is found as the intersection of the two projection rays. Repeating such a procedure for several points yields the 3D shape and configuration of the objects in the scene. Note that this procedure – sometimes referred to as *triangulation* – requires complete knowledge of both the internal and the external parameters of the cameras. Due to noise in the localization of the image points, the backprojected rays are not guaranteed to intersect, we will come back to this problem in section 3.6.

The whole method is seen to be based on finding the projection of points in both images. More specifically, if a point in the left image is selected, the problem is to find the corresponding point in the right image, assuming one exists (the point could be invisible to the right camera). Solving this *correspondence problem* is the hard part of stereo. In order to reconstruct scenes in 3D using a passive stereo setup, one has to solve "the two C's": the *calibration* of the cameras needs to be known, as well as *correspondences* between the cameras.

In order to derive some mathematical formulas for triangulation, a simpli-

fied stereo setup is considered in which the two cameras have coplanar image planes, parallel axes, identical intrinsic parameters and the same X axis. If the coordinates of a pair of matching 2D image points are known, the coordinates of the 3D point can be easily computed, the most important one of which is Z, corresponding to the depth. Due to the simplified setup, this coordinate can easily be computed from the x-coordinate of the points x_1 and x_2 as

$$Z = \frac{bf}{(x_1 - x_2)} \tag{2.3}$$

The resolution of the localization of the 2D points is limited to the order of one pixel. This sets an upper bound on the depth of points that we can measure accurately. This upper bound can only be increased by increasing either b or f or both. However, a larger baseline or a larger focal length both infer less overlap between the two images, and thus a smaller part of the world that can be reconstructed in 3D.

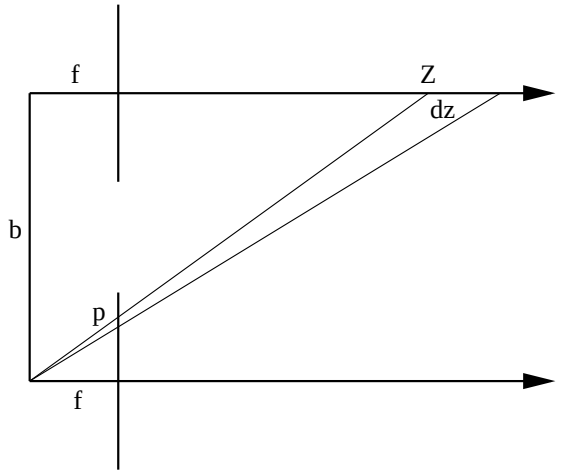


Figure 2.7: A localization error in the second image has an influence on the depth estimation of the 3D point. The larger Z, the larger this error.

The accuracy of the depth estimation also decreases with increasing depth. Figure 2.7 shows a top view of the simple stereo setup. Let us assume that the point in the left image has been detected correctly but that an error of p pixels has been made in the localization in the second image. The resulting depth error dZ at depth Z is then given by Eq. 2.4, which clearly indicates that the error dz increases for increasing Z.

$$dZ = \frac{pZ^2}{(bf - pZ)} \tag{2.4}$$

Figure 2.8 illustrates the decreased reconstruction quality one obtains when

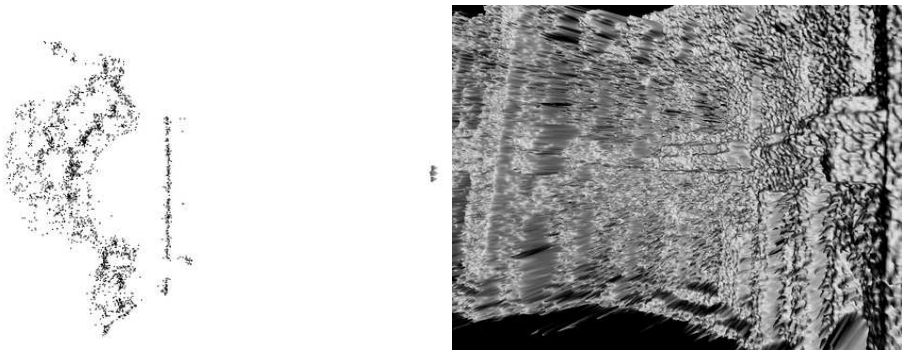


Figure 2.8: Left: top view of a point reconstruction of a scene. The baseline between the cameras on the right-hand side is very small w.r.t. the distance to the scene. Right: untextured view of the dense reconstruction of the scene. The accuracy in the depth direction is limited. The noise is clearly visible.

the cameras are too close together. The reconstruction depicted here was made with the webservice, explained in chapter 7. The left image of Fig 2.8 shows a top view of a point reconstruction of the scene with the cameras on the right-hand side clearly very close together with respect to the size of and the distance to the scene. The right image shows the degraded quality of the dense reconstruction with clearly large errors in the depth direction of the cameras.

2.3.1 The Teleman Stereo Head

Chapters 5 and 6 will describe the work that has been performed in the scope of several projects with the European Space Agency ESA. A project that is not mentioned in these chapters is *Teleman*. In this ESA-study we built a stereo head, which we calibrated with the techniques of section 2.2.1. The goal of the *Teleman* system, to be employed in space robotics, was to use stereo images to perform measurements and to compute 3D models of robotic taskboards on-board spacecrafts. In a typical operation, the stereo head is calibrated once. Afterwards it is moved to several positions to acquire 3D measurements. This requires an extra registration step every time the stereo head is moved in order to align the coordinates of the reconstructed 3D points with the world coordinate system.

Some triangulation measurements are shown in Fig 2.9. Three points on the right plane are indicated in both the left and right image, so their 3D coordinates can be computed. This allows us to compute distances between the points as well as the parameters of the plane going through these points. In Fig 2.9 we computed the distance between points 0 and 2 as being 133.7mm (ground truth 134.0mm). The distance between points 0 and 1 should also be 134.0mm but the measurement is a bit worse (132.9mm). This error can be explained by

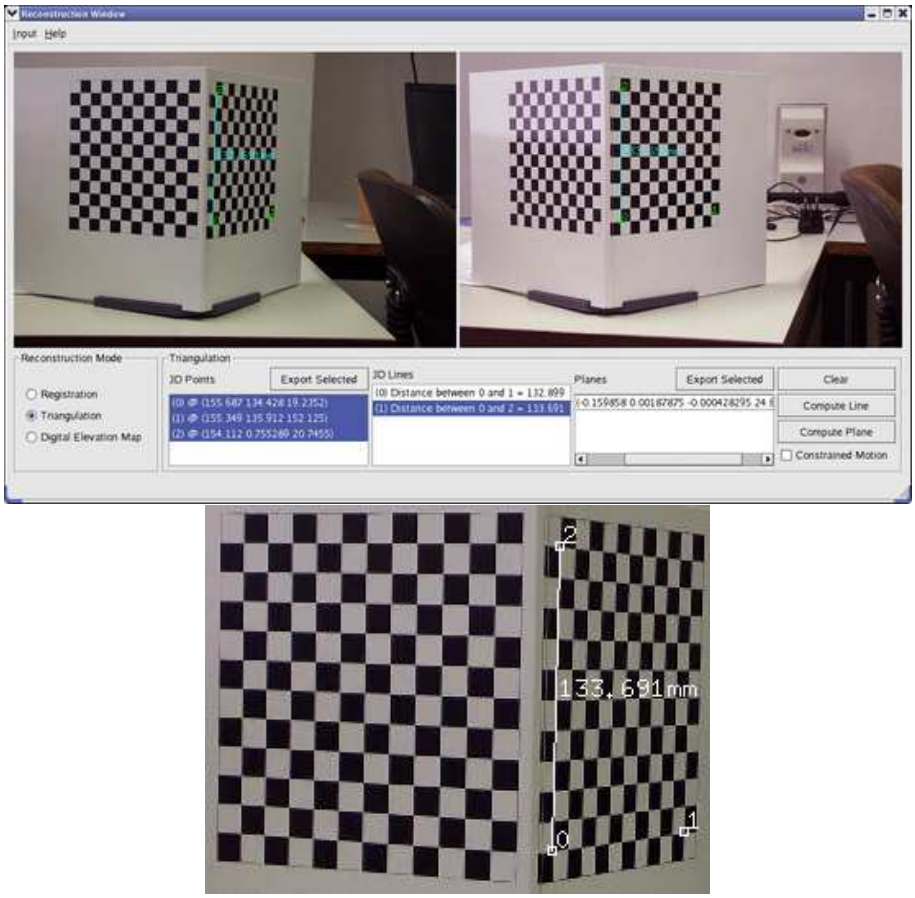


Figure 2.9: Measurements via triangulation in the *Teleman* project. Top: view of the triangulation GUI. Bottom: detail of the left image

two reasons. First the plane 0-1-2 is not frontal in the left image, which makes localization of the points less accurate. Furthermore, point 1 is clearly further away from the left camera. As explained in Eq. 2.4, the accuracy of a stereo system decreases for larger distances.

Fig 2.9 also shows the recovered parameters A, B, C, D of the plane-equation $Ax + By + Cz + Dw = 0$. They are (-0.156, 0.0019, -0.0004, 24.6436). As expected, B and C are very small w.r.t. A ($\leq 1\%$) because we are dealing with a plane that is parallel to the YZ-plane. The x-position of the plane is $\frac{24.6436}{0.156} = 154.16$mm. The true distance between the right plane of the calibration object and the chosen world frame origin is 154.0mm.

Chapter 3

Structure And Motion

Stereo imagery from calibrated stereo heads has been used for reconstruction purposes for many years. In the last decade insights, mathematical descriptions and techniques have been developed to reconstruct 3D information from uncalibrated imagery, i.e. sets or sequences of images without knowledge of the intrinsic or extrinsic parameters of the camera that recorded them. Since both the 3D structure and the camera calibration must be recovered, these techniques are referred to as *structure and motion* (SaM). In this chapter we will give an overview of the most important aspects of SaM. It has always been our intention to build 3D reconstruction systems with a very high degree of autonomy, taking efficiency into account as well. Therefore we will not simply focus on the theory but mostly on the algorithmic applicability in a real system.

This chapter consists of three consecutive parts. Sections 3.1 up to 3.4 deal with matching two views. In sections 3.5 and 3.6 we discuss how other views can be added in order to create a projective 3D reconstruction. Sections 3.7 up to 3.10 describe self-calibration techniques that upgrade the result to metric reconstructions.

3.1 The Geometry of Two Images

As described in 2.3, stereo image pairs can be used to reconstruct corresponding points in 3D. In this section we will take a closer look at the setup and we will discover that there exists an underlying geometric structure that can be employed, called **epipolar geometry**.

3.1.1 Epipolar Geometry

Consider again the simple stereo setup we described in 2.3. We explained there that in order to reconstruct points in 3D we need the calibration of the two cameras and corresponding image points. If a point $\mathtt{m}_1$ is given in one image and we want to find the corresponding point $\mathtt{m}_2$ in the other image, the concept of

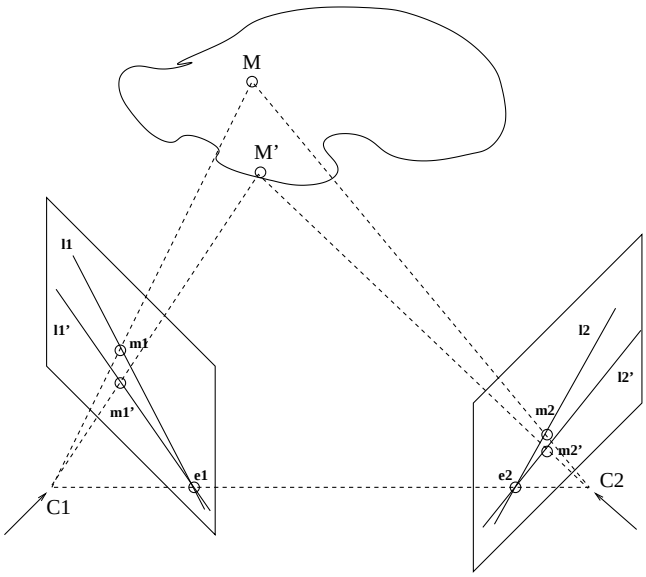


Figure 3.1: All points on the 3D line $\mathtt{M} - \mathtt{C}_1$ are projected onto $\mathtt{m}_1$ in image 1. This means that the corresponding point of $\mathtt{m}_1$ in the second image must lie on the projection of this line in this image. We call this line ($\mathtt{l}_2$) the epipolar line of $\mathtt{m}_1$. The three 3D points $\mathtt{M}$, $\mathtt{C}_1$ and $\mathtt{C}_2$ form a plane which intersects the two image planes in the two corresponding epipolar lines $\mathtt{l}_1$ and $\mathtt{l}_2$. Another 3D point $\mathtt{M}'$ forms another plane and intersects in two other epipolar lines $\mathtt{l}'_1$ and $\mathtt{l}'_2$. Because the baseline always lies in the planes, all epipolar lines intersect in the so-called epipoles: the intersection of the baseline with the images.

epipolar geometry, illustrated in Fig. 3.1, limits the search for the corresponding point in the second image to the epipolar line $\mathtt{l}_2$ which is the projection of the backprojected ray formed by the first camera center $\mathtt{C}_1$ and $\mathtt{m}_1$. This is a strong constraint indeed, because it eliminates an entire degree of freedom. Analogously, starting from a point $\mathtt{m}_2$ in the second image, the backprojected ray between $\mathtt{C}_2$ and $\mathtt{m}_2$ is projected into image 1 and yields the epipolar line $\mathtt{l}_1$.

In fact every 3D point $\mathtt{M}$ defines a plane together with the two camera centers $\mathtt{C}_1$ and $\mathtt{C}_2$. The two epipolar lines $\mathtt{l}_1$ and $\mathtt{l}_2$ are found as the intersection of this plane with the images. Any other 3D point in this plane is therefore projected onto these two epipolar lines. In other words, all points on $\mathtt{l}_1$ have a corresponding point on $\mathtt{l}_2$ (provided this correspondences exists, which it does not in the case of occlusion). Any 3D point $\mathtt{M}'$ outside this plane, together with $\mathtt{C}_1$ and $\mathtt{C}_2$ defines another plane, yielding another pair of epipolar lines. The set of all these planes forms a pencil around the line between $\mathtt{C}_1$ and $\mathtt{C}_2$ (the baseline). We call the intersection of this line with the images ($\mathtt{e}_1$ and $\mathtt{e}_2$ respectively) the epipoles. An epipole is in fact the projection of the other camera center in an

image. Since this line is part of all the planes, the intersection of this line with each image (the epipole) lies on all epipolar lines. This means that all epipolar lines in the first image intersect in $\mathtt{e}_1$ and all epipolar lines in the second image intersect in $\mathtt{e}_2$ as shown in Fig. 3.1.

3.1.2 The Fundamental Matrix

The fact that all points on epipolar line $\mathtt{l}_1$ have corresponding points in the other image that have to lie on the corresponding epipolar line $\mathtt{l}_2$ can be nicely expressed using the *fundamental matrix*. The derivation of this relation can be found in [39]. If the points $\mathtt{m}_1$ and $\mathtt{m}_2$, located in image 1 and 2 respectively, are corresponding points, then the following relation holds

$$\mathtt{m}_2{}^T \mathbf{F} \mathtt{m}_1 = 0 \tag{3.1}$$

If the calibration of the two cameras is given by the projection matrices $\mathbf{P_1}$ and $\mathbf{P_2}$, then $\mathbf{F}$ can be found as

$$\mathbf{F} \simeq (\mathbf{P_2}{}^T)^\dagger \mathbf{P_1}{}^T [\mathtt{e}_1]_\times \tag{3.2}$$

with † denoting the pseudo-inverse.

Equation 3.1 is the well-known expression of the relationship between two corresponding points $\mathtt{m}_1$ and $\mathtt{m}_2$ via the *fundamental matrix* $\mathbf{F}$. Given this 3×3 matrix and either $\mathtt{m}_1$ or $\mathtt{m}_2$ the corresponding epipolar line in the other image is easily computed as

$$\begin{aligned} \mathtt{l}_2 &\simeq \mathbf{F}\mathtt{m}_1 \\ \mathtt{l}_1 &\simeq \mathbf{F}^T\mathtt{m}_2 \end{aligned}$$

Since $\mathbf{F}$ was defined as $(\mathbf{P_2}{}^T)^\dagger \mathbf{P_1}{}^T [\mathtt{e}_1]_\times$ and $[\mathbf{a}]_\times \mathbf{a} = 0$ for all vectors $\mathbf{a}$, this means that

$$\mathbf{F}\mathtt{e}_1 = \mathbf{0} \tag{3.3}$$

This can only hold if the fundamental matrix is rank-deficient, i.e. has a maximal rank of 2. Equation 3.3 defines the epipole $\mathtt{e}_1$ as the right nullvector of $\mathbf{F}$ and we see that all epipolar lines $\mathtt{l1}$ must go through the epipole since

$$\mathtt{m}_2^T \mathbf{F} \mathtt{e}_1 = 0$$

for all points $\mathtt{m}_2$. Analogously, the epipole $\mathtt{e}_2$ in image 2 is the left nullvector of $\mathbf{F}$.

3.1.3 Computation of the Fundamental Matrix

In this paragraph we will explain how the fundamental matrix can be computed.

Linear Computation

It is clear from its definition in Eq. 3.2 that $\mathbf{F}$ can be computed if the projection matrices of the two cameras are known. If this calibration is not known, Eq. 3.2 cannot be used and we must find other means to compute it. A possibility is to employ a limited set of known point correspondences to compute the fundamental matrix. Indeed, Eq. 3.1 shows that every corresponding point pair $(\mathtt{m}_1, \mathtt{m}_2)$ delivers one constraint on the 9 elements of the 3×3 matrix $\mathbf{F}$. Since Eq. 3.1 is a homogeneous equation, $\mathbf{F}$ only needs to be computed up to a scale, so only 8 constraints are necessary.

Let us assume we have 8 corresponding point pairs at our disposal. A possible way to obtain these is by human interaction where an operator indicates 8 corresponding points. We define the vector $\mathbf{f}$ as

$$\mathbf{f} = \begin{bmatrix} F_{11} & F_{12} & F_{13} & F_{21} & F_{22} & F_{23} & F_{31} & F_{32} & F_{33} \end{bmatrix}^T$$

which holds the elements of $\mathbf{F}$. If $\mathtt{m}_1 = \begin{bmatrix} x_1 & y_1 & w_1 \end{bmatrix}^T$ and $\mathtt{m}_2 = \begin{bmatrix} x_2 & y_2 & w_2 \end{bmatrix}^T$ then Eq. 3.1 boils down to

$$\begin{bmatrix} x_2x_1 & x_2y_1 & x_2w_1 & y_2x_1 & y_2y_1 & y_2w_1 & w_2x_1 & w_2y_1 & w_2w_1 \end{bmatrix} \mathbf{f} = 0 \quad (3.4)$$

Every correspondence gives us one equation like Eq. 3.4. We can now stack all 8 equations and thus form the matrix $\mathbf{A}$ where

$$\mathbf{A}\,\mathbf{f} = 0 \quad (3.5)$$

This system of equations is easily solved by Singular Value Decomposition (SVD) [32]. Applying SVD to $\mathbf{A}$ yields the decomposition $\mathbf{USV}^T$ with $\mathbf{U}$ and $\mathbf{V}$ orthonormal matrices and $\mathbf{S}$ a diagonal matrix containing the singular values. These singular values are positive and in decreasing order. Therefore in our case σ_9 is guaranteed to be identically zero (8 equations for 9 unknowns) and thus the last column of $\mathbf{V}$ is the correct solution for $\mathbf{f}$ (at least as long as the eight equations are linearly independent, which is equivalent to all other singular values being non-zero). The matrix we obtain from $\mathbf{f}$ in this way however will, in the presence of noise, not satisfy the rank-2 constraint. This means that there will be no real epipoles through which all epipolar lines pass. A solution to this problem is to take as $\mathbf{F}$ the closest rank-2 approximation of the solution coming out of the linear equations.

Non-Linear Computation

The 8-point algorithm described before is linear and hence very popular. We know however that $\mathbf{F}$ only holds 7 degrees of freedom ($\mathbf{F}$ is defined up to a scale factor and its determinant is zero), so 7 correspondences should suffice to compute it. Since the extra constraint consists of enforcing that the rank of $\mathbf{F}$ should be 2, which is a non-linear constraint, the computation ultimately boils down to solving a non-linear equation.

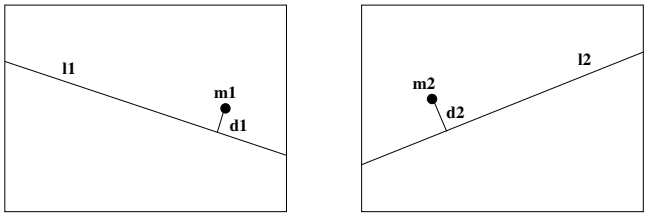


Figure 3.2: The geometric error we want to minimize is the sum of the distances d_1 and d_2 of the points to the respective epipolar line.

A similar approach as in the previous section can be followed to characterize the right null-space of the system of linear equations originating from the seven point correspondences. Since we are one constraint short, the solution $\mathbf{f}$ must be a linear combination of the last two singular vectors in $\mathbf{V}$.

$$\mathbf{f} = \mathbf{v_8} + \lambda \mathbf{v_9}$$

This means the fundamental matrix can be written as

$$\mathbf{F} = \mathbf{F_8} + \lambda \mathbf{F_9}$$

with $\mathbf{F_8}$ and $\mathbf{F_9}$ the matrices corresponding to $\mathbf{f_8}$ and $\mathbf{f_9}$ respectively. We now have to search for the unknown λ which makes the rank of the matrix 2. The rank of $\mathbf{F}$ is only 2 if its determinant is 0. Hence

$$\det(\mathbf{F_8} + \lambda \mathbf{F_9}) = a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0 = 0$$

which is a polynomial of degree 3 in λ . This can simply be solved analytically. There are always 1 or 3 real solutions. The special case where $\mathbf{F} = \mathbf{F_9}$ (which is not covered by this parametrization) is easily checked separately, i.e. $\mathbf{F_9}$ should have rank-2. If more than one solution is obtained then one extra correspondence is needed to obtain the true fundamental matrix.

Using More Matches

The human operator can of course indicate more than the minimum needed amount of matches. We can make use of all these matches to compute the fundamental matrix. The easiest way to do so is to write down the equation for each match and stack all these equations in the matrix $\mathbf{A}$ of Eq. 3.5. The fundamental matrix, obtained by solving the SVD of this $\mathbf{A}$ is the best solution in a least-squares sense. If many matches are available, the amount of rows of $\mathbf{A}$ can become quite large which makes computation of the SVD an expensive task. However, if $\mathbf{Af} = 0$ then $\mathbf{A}^T\mathbf{Af} = 0$ as well. Hence, by taking the SVD of the matrix $\mathbf{A}^T\mathbf{A}$ which always has 9×9 elements, we can also solve for $\mathbf{F}$.

The scheme described above gives equal weight to every match. Also, we actually solve for $\mathbf{F}$ by searching for the solution which minimizes an algebraic

error (the least squares error of the algebraic equation 3.1). The error we actually want to minimize is a geometric error, expressing the distance of the points to the corresponding epipolar lines, d_1 and d_2 in Fig. 3.2. In homogeneous coordinates the distance between a point `m` and a line `l` is given by

$$d = \frac{(m_x l_x + m_y l_y + m_w l_w)^2}{l_x^2 + l_y^2 + l_w^2} \tag{3.6}$$

A possible way of minimizing this error is by employing Reweighed Least Squares. This technique solves for $\mathbf{F}$ in an iterative way. The SVD of the matrix $\mathbf{A}$ gives us a first solution for $\mathbf{F}$. Then, we compute the residual distance for each match as shown in Fig. 3.2. Every row of $\mathbf{A}$ is then divided by the residual of its respective point match, measured in the images with Eq. 3.6. We then solve again for $\mathbf{F}$ by computing the SVD of the reweighed $\mathbf{A}$. This means that correspondences with small residuals will have more impact on the data and that *worse* correspondences (i.e. with higher residuals) will have a smaller impact. After a couple of iterations this algorithm converges to a better estimate of $\mathbf{F}$, minimizing the geometric instead of the algebraic error.

Another possibility to take into account multiple matches for the computation of $\mathbf{F}$ is a non-linear optimization. The linear SVD solution can be used as an initialization to a Levenberg-Marquardt algorithm or another non-linear optimization scheme which explicitly minimizes the geometric error. Because the error function is known and quadratic, its derivatives can easily be written down. Together with the good initialization, this will help the Levenberg-Marquardt algorithm to quickly converge onto the solution which minimizes the geometric error.

Conditioning

If we compute the fundamental matrix by solving an SVD containing rows like the ones given in Eq. 3.4, we will quickly run into problems. The result of an SVD is the best solution to the system of equations in a least squares sense. This only works stably if the different columns of the matrix $\mathbf{A}$ stacking the equations have approximately the same magnitude. Typically we deal with images of 1000×1000 pixels or more. This means the coordinates of the extracted features are in the order of (1000, 1000, 1). The third, homogeneous, coordinate w is clearly much smaller than the other two. This means the different columns of the matrix $\mathbf{A}$ will have the following order of magnitude.

$$\begin{bmatrix} x_2x_1 & x_2y_1 & x_2w_1 & y_2x_1 & y_2y_1 & y_2w_1 & w_2x_1 & w_2y_1 & w_2w_1 \\ 1e6 & 1e6 & 1e3 & 1e6 & 1e6 & 1e3 & 1e3 & 1e3 & 1 \end{bmatrix}$$

It is clear that an error on the ninth element in the solution vector will have a much smaller impact on the total least squares residual of the system. This means the solution will be biased and hence unusable. A way around this problem is to make sure the data is *conditioned* before use. Optimal conditioning

algorithms can be found in the literature but in practice a simple scaling of the 2D coordinates of the image points to the range [-1,1] yields satisfactory results. The input data is transformed by a matrix $\mathbf{T}$, according to the image width ($2w$) and height ($2h$):

$$\begin{aligned} c_x &= \frac{w}{2} \\ c_y &= \frac{h}{2} \\ s &= \frac{1}{c_x + c_y} \\ \mathbf{T} &= \begin{bmatrix} s & 0 & -sc_x \\ 0 & s & -sc_y \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.7}$$

The transformed points $\mathtt{m}' \simeq \mathbf{T}\mathtt{m}$ are filled into Eq. 3.5 where the matrix $\mathbf{A}$ is now well conditioned (all columns have the same order of magnitude). The resulting fundamental matrix $\mathbf{F}'$ is then transformed into real image space

$$\mathbf{F} \simeq \mathbf{T_2}^T \mathbf{F}' \mathbf{T_1}$$

where $\mathbf{T}_1$ and $\mathbf{T}_2$ are obtained as in Eq. 3.7 by filling in the parameters for the first and the second image respectively.

3.2 Features

As explained in chapter 2, one of the important prerequisites for 3D reconstruction is the detection of correspondences between images. The 7- and 8-point algorithms for the computation of the fundamental matrix, aimed at facilitating the search for correspondences, also need initial correspondences. For a human operator, this may seem an easy task. Unfortunately, the algorithms we feed to a computer still lack the kind of global scene understanding which underlies much of human performance.

Anyway, the problem can at least be made easier by focusing on salient structures in the images. Features that stand out have a better chance to be detected robustly, with good repeatability and accurate localization. This section discusses examples of such *interest points*.

3.2.1 Simple Interest Points

The interest point extraction and matching process consists of several steps. One type of interesting features would be points that can automatically be extracted and are stable across views, such that they can be matched. This is the *feature detection* part of the approach. Then, the points are characterized through measures calculated from the intensity or color patterns in their immediate

surroundings. The choice of such surrounding or *feature region* can be quite involved as well. The measures extracted from the region are used for the *feature description*. The matching is based on the comparison of the descriptors for features detected in the different images. For the same descriptors, one can still opt for different *feature matching* strategies.

We will come to possible schemes for detecting interest points soon. Once such points have been selected, one of the simplest descriptions would be the intensities in their surrounding *per se*. In order to compare these, one can use the sum-of-squared-differences (SSD) or normalized cross-correlation(NCC) for the matching, as explained next.
Consider a window W_I in image I and the corresponding region in image J which is found by transforming the window W_I with transformation $\mathbf{T}$. The dissimilarity between two image regions based on SSD is given by

$$D = \iint_W [J(\mathbf{T}(x,y)) - I(x,y)]^2 v(x,y) dxdy \tag{3.8}$$

where $v(x,y)$ is a weighting function. Typically $v(x,y) = 1$ or it is a Gaussian. In practice, the integral is computed as a summation of the squared differences of pixels in a window in the images with widths w and heights h.

$$D = \sum_{x=-w}^{x=w} \sum_{y=-h}^{y=h} [J(\mathbf{T}(x,y)) - I(x,y)]^2 v(x,y) \tag{3.9}$$

Note that this value gives the dissimilarity, i.e. a figure which states how much intensities in the windows in the two images differ.

While having the advantage of being very easy to compute, the SSD has the major drawback of being influenced by the global intensity of an image. Indeed, if one image is taken under bright and the second under dark conditions, then two matching windows will still yield a dissimilarity which can be rather large due to the intensity shift between the images. The NCC comparison function does not have this problem. It gives a similarity value defined as

$$S = \frac{\iint_W \left(J(\mathbf{T}(x,y)) - \overline{J}\right) . \left(I(x,y) - \overline{I}\right) v(x,y) dxdy}{\sqrt{\iint_W \left(J(\mathbf{T}(x,y)) - \overline{J}\right)^2 v(x,y) dxdy . \sqrt{\iint_W \left(I(x,y) - \overline{I}\right)^2 v(x,y) dxdy}}} \tag{3.10}$$

where

$$\begin{aligned} \overline{I} &= \frac{\iint_W (I(x,y)) dxdy}{\iint_W dxdy} \\ \overline{J} &= \frac{\iint_W (J(\mathbf{T}(x,y))) dxdy}{\iint_W dxdy} \end{aligned}$$

are the mean intensities of the windows W and $\mathbf{T}(W)$ in the images. This time a similarity measure is computed which indicates how well the image windows

correspond. Taking into account the mean intensity and the variance of the windows makes this measure stable against global intensity changes.

The SSD (eq. 3.8) and NCC (eq. 3.10) measures suggest matches when they are small or large, resp. Apart from being able to match features in the first place, we want them to be as unique as possible, i.e. they should be discriminant. Features that look like many others will yield many false matches. In the context of feature tracking, Shi et al. [77] proposed their *KLT* feature scheme, thereby suggesting a good feature detection procedure. We employ local derivatives of the image intensity to define the matrix $\mathbf{Z}$ as

$$\begin{aligned} \mathbf{Z} &= \int\int_W \begin{bmatrix} \frac{\partial J}{\partial x} \\ \frac{\partial J}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial J}{\partial x} & \frac{\partial J}{\partial y} \end{bmatrix} v(x,y)dxdy \\ &= \int\int_W \begin{bmatrix} \left(\frac{\partial J}{\partial x}\right)^2 & (\frac{\partial J}{\partial x})(\frac{\partial J}{\partial y}) \\ (\frac{\partial J}{\partial x})(\frac{\partial J}{\partial y}) & \left(\frac{\partial J}{\partial y}\right)^2 \end{bmatrix} v(x,y)dxdy \end{aligned} \tag{3.11}$$

According to Kanade et al, a feature can be tracked well if the matrix $\mathbf{Z}$ defined in Eq. 3.11 is well conditioned. Practically, these are points that can be detected with good precision, and hence also yield a good basis for matching. Kanade et al. [77] impose that the smallest eigenvalue of $\mathbf{Z}$ should be larger than a threshold.
Harris [36] arrived at similar results, using the eigenvalues of this matrix in a different manner to extract a *corner response* function

$$R = det(\mathbf{Z}) - k(trace(\mathbf{Z}))^2 \tag{3.12}$$

where Harris suggests the value $k = 0.04$. To extract only well conditioned features, local maxima of the corner response function are selected that are above-threshold. Sub-pixel precision can be achieved by quadratic approximation of the neighborhood of a local maximum. A typical choice for $v(x, y)$ in this case is a Gaussian with $\sigma = 0.7$.

Harris corners are features that are typically very well localized in the image. In order to match features between different images, a local neighborhood of 7×7 pixels or more (depending on the image size) is extracted around the features in both images and the SSD or NCC is computed between these windows. Fig. 3.3(left) shows an image with 1000 extracted Harris corners. As this example shows, one should be very careful with the selection of the corners. Typically one selects an upper limit N of features. If these are just the N strongest (based on the corner response function), one stands the risk of extracting corners in one part of the image only where the gradients are very strong. This can clearly be seen on the left in Fig. 3.3 where most of the corners are found in the trees. A better option is to select the M strongest corners with $M = 0.7N$. For the other 30%, the image is divided in different regions and the strongest not yet selected features in each area are added to the list. The result of this algorithm is shown on the right in Fig. 3.3. The better distribution of the features is clearly visible.



Figure 3.3: Extracted Harris points on an image without forcing distribution over the image (left) and with distribution enabled (right)

3.2.2 Affine Invariant Features

The image descriptors of the aforementioned schemes simply consisted of the intensities in square regions around the detected feature points, the patterns of which are then compared. Such schemes are rather fragile as soon as images are taken from appreciably different viewing directions. The intensity patterns within square regions of fixed sizes will start to contain intensities from surface patches that are quite different. Indeed, when looking at the same surface patch from such a novel viewpoint, it will look quite deformed and direct intensity pattern matching will break down. Rather than using the SSD or NCC measures directly on intensities taken from fixed regions, as previously proposed for *small baseline* conditions, the shape of the regions and the feature descriptors from them have to be designed more carefully under *wide baseline* conditions. Wide baseline conditions will increase both the geometric and photometric changes between views. In order to be robust against these changes, one needs a sufficient level of geometric and photometric invariance.

A short note on photometric changes is in order here. What is meant is that the same point of a scene can change its intensity and / or color from one view to another. First, the illumination conditions may change. For instance, one image can be taken while the scene is flooded by sun light, and the next under an overcast sky. Moreover, the local material may reflect light differently towards the camera if this has moved. Highlights are an example of such a directional effect, and often cause problems to matching algorithms. In general, it is required that one achieves at least some level of invariance under such photometric changes as well.

Recently features have been proposed that are invariant under affine transformations, to cope with geometric changes, and under photometric transformations, to cope with changing illumination conditions. Examples of such features are Intensity- or Edge-based features [97], combined with generalized color moments [61].

3.2.3 SIFT and SURF Features

Starting from fixed square regions with their intensities as descriptors, we have made our way into schemes that adapt to the viewing conditions, in order to cater for wide baseline conditions, changing illumination, etc. The use of invariance comes at a price, however. Not only get the descriptors less discriminant (they are insensitive to more change by design), but they are also more difficult to extract and compute. This latter element tends to bring in more noise and lowers the robustness and speed of extraction. Therefore, schemes that are not invariant under more than image scaling and rotation in principle, sometimes turn out to be the better option in practice nevertheless, even if the camera rotates in 3D quite a bit.

Features

Recently Lowe [58] proposed SIFT features, for instance. These use square regions, but the descriptors are more robust to image changes than the direct use of intensities or colors. The SIFT acronym stands for *Scale Invariant Feature Transform*. SIFT features are scale and rotation invariant and are up to some degree robust (though not mathematically invariant) against changes in 3D viewpoint and illumination. SIFT features can be extracted relatively quickly. The extraction of SIFT features follows the two standard steps of feature detection and region description. The former localizes features in the image as the extrema of a difference-of-Gaussian function and extracts square regions around these points, enhanced with a reference orientation obtained from the image gradient directions in a local image window. The latter is based on image gradient histograms.

Building further on the idea of fast and robust feature extraction and description, Bay et al. [7] developed SURF (*Speeded-Up Robust Features*) features, which are also rotation and scale invariant. The localization is done by searching extrema in both the scale and spatial dimension of the determinant of an approximated Hessian of image intensities. This can be done very efficiently by applying box filters and using integral images. Similar to SIFT, a reference orientation is computed and the descriptor is extracted as a weighted sum of intensity contrasts in x and y direction at different positions in the region, yielding a feature vector of 64 elements. The sign of the Laplacian (computed as the trace of the Hessian) can be used as a quick check to discern two types of features.

The strength of a SURF feature is given by the determinant of its Hessian. Similar to Harris corner extraction, one stands the chance of localizing many strong features in a small part of the image. That is why for use in Structure and Motion algorithms, we again allow the algorithm to select the strongest features in the image and force it to detect a significant amount of other features, equally distributed over the image.

Matching

In order to match SIFT or SURF features a different scheme needs to be used than for image windows with NCC or SSD. Two features can be compared by computing the distance between their normalized feature vectors. One could set a threshold on this distance, similar to NCC or SSD matching. However, Lowe found that it is better to use a different strategy in which for every feature in the first image not only the best but also the second best matching feature in the second image is searched. If the ratio in matching score between these two is large, the match is very stable and a good candidate. In practice, matching can be done very efficiently using KD-trees and Approximate Nearest Neighbour searches [63].

The 3D webservice we will describe in chapter 7 makes use of SURF features. We have found them to be better localized than SIFT features, which is absolutely vital for Structure and Motion algorithms. The added rotational and scale invariance has helped us to match images that are quite far apart and could not be matched with ordinary, Harris and NCC based techniques.

3.2.4 Matching Point Configurations

So far, we have focused entirely on matching points between views. But one could also use other features, e.g. straight line segments, or special configurations of features, e.g. configurations of points. A nice example is given by quadruples of collinear points. Such point alignments can be matched via the use of the well-known cross-ratio. This descriptor is not merely affine invariant, but even projective invariant.

If four points $\mathtt{m}_1$,$\mathtt{m}_2$,$\mathtt{m}_3$,$\mathtt{m}_4$ are collinear, then their cross-ratio, defined as

$$Cr(\mathtt{m}_1, \mathtt{m}_2, \mathtt{m}_3, \mathtt{m}_4) = \frac{\Delta_{13}\Delta_{24}}{\Delta_{14}\Delta_{23}} \tag{3.13}$$

is projectively invariant with $\Delta_{ij} = \sqrt{\left(\frac{x_i}{w_i} - \frac{x_j}{w_j}\right)^2 + \left(\frac{y_i}{w_i} - \frac{y_j}{w_j}\right)^2}$ the Euclidean distance between points i and j. Based on this invariant, quadruples of collinear points can be matched between views.

Fig. 3.4 shows a practical example of the cross ratio's use on images taken by the robot on-board the Japanese Engineering Test Satellite ETS-VII. More information on the specifics of this application are given in chapter 5. The cross-ratio is one of many possible projective invariants one can use. The major drawback of such invariants is the amount of points, lines, ... one has to combine to compute them and the restrictions like collinearity or coplanarity. For instance the cross-ratio requires the simultaneous extraction of 4 collinear points. The so-called 5 point cross-ratio needs 5 non-collinear but coplanar points. Without prior information on which points may be collinear or coplanar, many combinations of points would have to be tried.

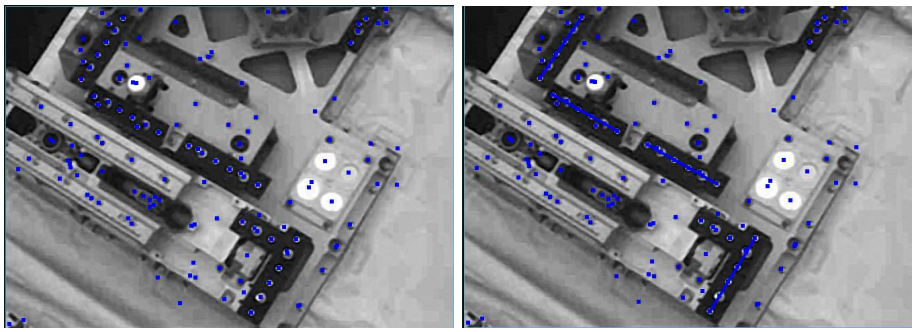


Figure 3.4: Extracted points on an image of the ETS-VII satellite (left). Identified markers where collinear points with the correct cross-ratio are found (right).

3.3 Computing Epipolar Geometry from Automatically Found Correspondences

In section 3.1 we explained that for every image pair an underlying structure exists which is called epipolar geometry and which can be expressed by means of the fundamental matrix **F**. This matrix can be recovered if the calibration of the image pair is known or from a set of corresponding points in both images. Section 3.2 explained how features can be matched between different images. In this section we will demonstrate how to employ these matches to compute the fundamental matrix.

3.3.1 Dealing with Outliers: the RANSAC Algorithm

Linear and non-linear estimation of the fundamental matrix is only possible if all input data is reliable. If only one incorrect match is present in the set of equations building **A** of Eq. 3.5, the computed fundamental matrix will be biased by this incorrect match and unusable. These wrong matches are called *outliers*. Since correspondences are computed by the computer based on NCC, SSD or comparison of feature vectors, the probability of mistakes is high and the presence of outliers cannot be ignored. Hence, we need a robust way to deal with these outliers. There exists a well-known algorithm to do just that: given a data-set polluted by outliers, this algorithm is capable of computing the underlying structure and at the same time identifying the inliers and outliers in the data-set. The algorithm is called *RANSAC* which stands for RAndom SAmpling Consensus and was proposed by Fischler and Bolles [25] in 1981.

The RANSAC algorithm is in essence an iterative scheme. It is based on the assumption that a certain percentage of the input data-set consists of consistent inliers and that the outliers are just random mistakes in the data-set. In every iteration a set of samples is taken out of the data-set randomly. The amount of samples that is selected is equal to the minimum number we need to compute

the underlying structure. In the case of the fundamental matrix, this is 8 if we use the linear algorithm or 7 if we employ the non-linear algorithm. Every time the structure is computed using this random set of samples and the support amongst the rest of the data-set is computed. In other words, in every iteration we assume that all 7 or 8 selected samples are inliers, and hence the computed structure is correct. We determine which of the remaining matches in the data-set are inliers for this structure and which are outliers. This is done by checking if matching points are on each other's epipolar line. A few pixels error can be tolerated because of the influence of noise. In reality there will be many iterations in which the set of samples contains at least one outlier and thus for which the computed structure is incorrect. However, the support for such an incorrect structure will be smaller than for the true solution. The iterative procedure is repeated until a stop-criterion has been met. The solution with the largest support (i.e. the largest number of inliers) is selected as the true solution.

Stop-Criterion

An important question when using RANSAC is when to stop the iterative process. We want to be reasonably sure that we have explored all our options without having to compute $\mathbf{F}$ from all possible sets of 7 points which is clearly combinatorially impossible. As explained in the previous paragraph, we only need to select one set of matches in which not a single outlier is present[1]. If this is guaranteed to occur, we also know for sure the $\mathbf{F}$ matrix will be computed and identified correctly from its support amongst the other matches. The amount of trials is thus clearly dependent on the percentage of inliers in the data set (ι). If we select p matches from the data-set, the chance that we have selected p correct matches is ι^p. If we perform this process n times, the chance that we have selected only good matches becomes

$$P = 1 - (1 - \iota^p)^n$$

Suppose we want to be 95% certain that we have at least one set of p inliers, then the number of tries n (depending on ι) is given by Eq. 3.14 where P stands for the certainty we want to achieve (e.g. 95%). Some values are shown in table 3.3.1.

$$n = \frac{\log(1 - P)}{\log\left(1 - \iota^p\right)} \tag{3.14}$$

Preemptive RANSAC

Since RANSAC's introduction in 1981, several improvements have been suggested to make it more stable and faster. I came up with a very powerful extension, called *preemptive* RANSAC. The most time-consuming part per iteration of

[1] Actually this is over-optimistic. If we select a set of 7 correct matching points that are all located close together in the same neighborhood of the image, the computed $\mathbf{F}$ matrix will only be correct for this neighborhood and not for the rest of the image.

ι	95%	90%	80%	70%	60%	50%	40%	30%	20%
n	3	5	13	35	106	382	1827	13692	233963

Table 3.1: Number of tries n we need to select 7 matches from the data-set to be 95% certain that we have selected at least 1 set of 7 inliers, depending on the percentage of inliers in the data-set ι.

the RANSAC algorithm is typically not the computation of the structure from the minimal sample set but rather the determination of the in- and outliers. For computing the fundamental matrix with RANSAC (called F-RANSAC in short) for instance, the distance between a point and an epipolar line has to be computed twice for every correspondence in the data set (once for image 1 to image 2 and once vice versa). This is a very expensive operation. However, if we can quickly determine that a certain solution is computed from a contaminated sample set, it doesn't make sense to compute all in- and outliers. The probability distribution of the matches in the data-set is described by a binomial distribution with the inlying fraction ι as the probability. Suppose that we have already checked y matches from the data-set. The expected amount of inliers is of course $y\iota$. The standard deviation of the distribution is given by

$$\sigma^2 = y\iota(1-\iota)$$

We define m as the expected amount of inliers minus a 'safety margin' of 2σ.

$$\begin{aligned} m &= y\iota - 2\sigma \\ &= y\iota - 2\sqrt{y\iota(1-\iota)} \end{aligned} \tag{3.15}$$

If we have fewer than m inliers, the probability that this solution is not good is about 97% and hence it can be discarded. For example, if we assume a (low) inlying fraction of 20% and after having checked 16 matches for support we still haven't found a single inlier, the probability that this solution is wrong is 97.18% so we can stop checking. This saves a lot of processing time which would otherwise have been spent on checking the rest of the data-set. In order to apply this preemptive approach, checking for support needs to be done on random matches in the data-set because otherwise the solution will be biased towards the matches that appear first.

Selecting a good a-priori value for ι is not always an easy task. However, it is possible to adapt this value during computation. Therefore, we typically start with a low inlying fraction (e.g. 20%). If, during RANSAC processing, we detect a solution with a support of $\kappa > \iota$, we immediately know for certain that the inlying fraction is at least κ. We can thus recompute the number of trials we need and also the minimum amount of inliers given by Eq. 3.15 will be higher, thus wrong solutions are detected sooner during further iterations of the RANSAC algorithm.

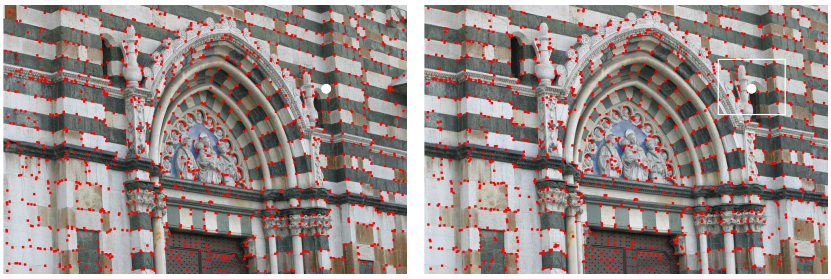


Figure 3.5: For small-baseline cases, the original search for matches is limited to a neighborhood in the second image around the coordinates of the point in the first image.

Global Transformation

If we use simple SSD or NCC to compare feature points between (small-baseline) images, a point in one image might have several resembling points in the other image. In order to limit the number of mismatches for the initial F-RANSAC step, we limit our search to a window in the other image centered around the same coordinates as the point in the first image, as shown in figure 3.5. This approach limits the number of matching work we have to perform and also reduces the number of mismatches. This limitation of course only works if small-baseline conditions apply since then feature points don't move too much between images. This approach also fails if a global shift and/or rotation exists between the two images. This happens frequently if the photographer does not point the camera at the same 3D point between consecutive images. In order to deal with this common situation, we compare the center part of the first image with equally large windows at different positions in the second, using NCC values at highly subsampled levels. The highest NCC score yields a global shift which brings the two images closer.

3.3.2 F-Guided Matching

In the original matching step, we limited our search for a correspondence to a window centered around the same coordinates as the point in the first image, taking into account the global shift. However, points in the foreground of the scene can still experience quite large coordinate changes, so that these will not be picked up. This is unfortunate because we want to find as many correspondences between the images as we can. Points that move more than the original matching region were not matched originally but now that we have computed the epipolar geometry, we can search for more matches. Indeed, the localization limitation can be lifted. By lowering the NCC or SSD threshold for matching and allowing for an epipolar distance of about 3 pixels (depending on the image size of course), this *F-guided matching* step delivers many extra matches.

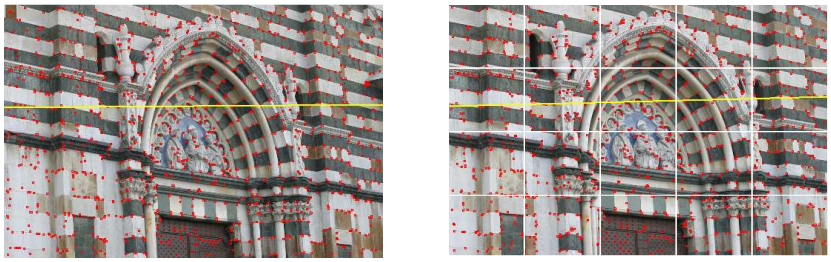


Figure 3.6: F-Guided corner matching. In order to find the corresponding point in the second image, we only have to search along the epipolar line. If we divide this image in tiles, only the features inside the tiles that are crossed by the line have to be checked.

A naive approach to F-guided matching would be to select a feature in the first image, go over all features in the second image, check if this feature is on the epipolar line and then compute the NCC value. A much more efficient approach is to divide the second image into tiles of say 50×50 pixels. Features in the second image are assigned to the tile they are located in. If we then want to search for a match for a feature in the first image, we compute the epipolar line in the second image and determine the tiles this line crosses. Only features in these tiles are possible candidates for matching. Fig. 3.6 illustrates the procedure. When the epipolar line is closer to the boundary of a tile than the distance threshold, we include the tile on the other side of the boundary as well.

3.4 Computing Homographies

Planar scenes can undergo several types of linear deformations, a list of which the reader can find in section A.4.1. The most general, the projective transformation or *homography* is defined in Eq. A.8. In this section we explain how to compute a homography matrix from point correspondences using H-RANSAC.

3.4.1 Computing Homographies with RANSAC

There are two important cases for which an homography matrix can explain the transformation between two images. The first is the already mentioned case of a single plane, viewed by two arbitrary cameras. The second is a setup where two cameras share the same camera center but are rotated w.r.t. each other. For both cases the homography can be computed if the calibration parameters and, for the first case, the parameters of the plane, are known.

Just like in the case of the fundamental matrix, the homography matrix can also be recovered from correspondences. To this end, we will employ the RANSAC algorithm again. Suppose we are dealing with a planar scene or

purely rotating cameras, then Eq. A.8 gives the mathematical formulation of the relation between two matching points described by the homography or projective transformation matrix **H**. We want to compute the matrix **H** such that Eq. A.8 is valid for all matching points. Note that knowledge of **H** gives a much stronger constraint on matches than knowledge of **F**. While **F** only predicts the epipolar line on which the corresponding point must lie, **H** completely defines the position of the corresponding point, i.e. there is no degree of freedom left.

The procedure to determine a homography from correspondences is very similar to the computation of epipolar geometry. We start by extracting a set of possible matches between the two images. This is the data set from which we will select random samples to compute **H**. Next comes the linear computation of **H** from a minimal set of matches. There are 9 elements in the 3x3 matrix **H** which is determined up to scale so **H** has 8 degrees of freedom. Every point match ($\mathtt{m}_1$,$\mathtt{m}_2$) delivers 2 constraints on **H** since both the x and y coordinate of $\mathtt{m}_2$ can be computed from **H** and $\mathtt{m}_1$. If we define the vector **h** similarly to **f** as

$$\mathbf{h} = \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{21} & H_{22} & H_{23} & H_{31} & H_{32} & H_{33} \end{bmatrix}^T$$

and given that every point match ($\mathtt{m}_1$,$\mathtt{m}_2$) delivers two equations

$$\begin{aligned} \frac{x_2}{w_2} &= \frac{\mathbf{H}_1\mathtt{m}_1}{\mathbf{H}_3\mathtt{m}_1} \\ \frac{y_2}{w_2} &= \frac{\mathbf{H}_1\mathtt{m}_1}{\mathbf{H}_3\mathtt{m}_1} \end{aligned}$$

with $\mathbf{H}_i$ the i-th row of **H**, then every point match gives us two row vectors

$$\begin{bmatrix} x_1w_2 & y_1w_2 & w_1w_2 & 0 & 0 & 0 & -x_1x_2 & -y_1x_2 & -w_1x_2 \\ 0 & 0 & 0 & x_1w_2 & y_1w_2 & w_1w_2 & -x_1y_2 & -y_1y_2 & -w_1y_2 \end{bmatrix}$$

that can be stacked to yield

$$\mathbf{A}\,\mathbf{h} = 0$$

Since every point delivers two equations, we only need 4 points to build a linear system with 8 equations. In other words the minimum sample in H-RANSAC consists of 4 points as opposed to 7 for **F**. The criterion that decides whether a match is an in- or outlier computes the distance between $\mathtt{m}_2$ and $\mathbf{H}\mathtt{m}_1$. If this distance is smaller than a threshold, the match is an inlier for **H**. Analogously to the computation of **F**, the matrix **A** needs to have well-conditioned columns. We will therefore condition the points with **T** given by Eq. 3.7. The resulting $\mathbf{H}'$ matrix must then be de-conditioned as follows

$$\mathbf{H} \simeq \mathbf{T}_2^{-1}\mathbf{H}'\mathbf{T}_1$$

3.4.2 Layered 3D Reconstruction

The main theme in this work is the recovery of life-like 3D reconstructions of scenes and objects. In section 2.3 we explained that the accuracy of depth estimation decreases the further the object is away from the camera. For parts



Figure 3.7: Left: model of the Namur Citadel without background. Right: combined view of both foreground 3D object and background texture at infinity.

of the scene that are in the far background, there is no hope of obtaining an accurate depth value. This results in a 3D model of the foreground floating around in black space, without any reference to its background. However, because the baseline between the cameras is typically negligibly small compared to the distance to the far-away parts of the scene, the transformation of these pixels from one image to another can be described with a homography as it is like the camera has only rotated for such parts. This allows us to render the foreground in 3D against a backdrop, consisting of the background.

An example of this was created using imagery recorded on the Namur Citadel during excavation under the surface of its parking lot. The left image of Fig 3.7 shows the 3D model of the foreground, without any reference to the background in which this model resides. In order to compute this background, the parts of all images that were not matched consistently with the 3D foreground were used to compute the homography of the plane at infinity. The inlying pixels were put on a virtual cube around the 3D object. When a new view of the scene was requested, first a view of the background was created by rendering the virtual cube with the camera always in its center, i.e. without taking into account the translational part of the camera. On top of this rendered image, we then rendered the 3D foreground. The right image of Fig. 3.7 shows a result of the rendering.

3.5 Multiple View Geometry

The previous sections dealt with finding features and correspondences between them in two views. The concept of epipolar geometry was derived and it was shown that the fundamental matrix can be computed from a set of correspondences that might be contaminated with outliers.

Just like the epipolar geometry describes constraints between features in a

setup of two views looking at the same scene, other constraints can be written down for a configuration of three views. A very elegant way to describe this setup is by means of the *trifocal tensor*. Whereas the epipolar constraint is the "simplest" constraint (in the sense that it involves the smallest number of views) that must hold between corresponding points in different images, so is the trifocal constraint the "simplest" constraint that must hold between corresponding lines in different images. Indeed if only two views of the scene are given, then any pair of lines in the two views might be corresponding lines because they can always be interpreted as being the images of the intersection line of their backprojecting planes. To verify whether this is indeed the case, a third view is needed. In fact the trifocal tensor allows to predict where the corresponding line is to be found in this image. In other words it allows to transfer lines from two images to a third one. There also is a trifocal relation between matching points in three views.

Relations between lines and points for constellations of more than 3 views exist too. The relation between 4 views for instance can be described by means of a quadrifocal tensor. We will not go into this here. The reader is directed to [62] and [21] for more information on these topics.

3.6 Structure and Motion

Many researchers have studied trifocal, quadrifocal or further tensors for 3D reconstruction. We will follow a more pragmatic approach, using computations in a projective frame using epipolar geometry only. The output of this algorithm will consist of projection matrices for the cameras and 3D coordinates of points in a projective frame. Hence we solve the problem of finding both *Structure and Motion* (SaM) from image information only. The structure of the algorithm we employ is given in table 3.6. The different steps in the algorithm are explained in the following paragraphs.

3.6.1 Initialization Step

Table 3.6 shows that the first two steps we employ consist of the computation of the epipolar geometry between the first two images and the initialization of the structure and motion. The former is done using the algorithms explained in the previous sections. The combination of F-RANSAC, non-linear optimization and F-guided matching delivers a good estimation of the fundamental matrix $\mathbf{F}_{12}$ and a set of matching image points.

Initializing the Projective Frame

If we want to initialize the structure, we need first to define initial values for the projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$. If these projection matrices were given, the fundamental matrix would be readily computed from these using Eq. 3.2. Vice versa, if the fundamental matrix is known, some constraints on the projection

Step 1.	Compute the epipolar geometry for the first image pair
Step 2.	Initialize structure and camera pair
Step 3.	For every consecutive image, do
	Step 3.1 Compute matches between this and the previous image
	Step 3.2 Estimate the projection matrix
	Step 3.3 Update structure and initialize new points
Step 4.	Global minimization of all results

Table 3.2: SaM algorithm

matrices can be written down. If we choose the first projection matrix to be located in the origin of the world, and to have an identity matrix as the left 3x3 part, then the two projection matrices can be initialized as

$$\mathbf{P}_1 = [\mathbf{I}_{3\times 3}|\mathbf{0}_3] \quad (3.16)$$

$$\mathbf{P}_2 = [[\mathtt{e}_{12}]_\times \mathbf{F}_{12} - \mathtt{e}_{12}\pi^T | a\mathtt{e}_{12}] \quad (3.17)$$

with $\mathtt{e}_{12}$ the epipole in the second image (the projection of the first camera center in the second image) and π an arbitrary 3D plane. The exact derivation is beyond the scope of this text; details can be found in [39]. Note that there are 4 degrees of freedom left in Eq. 3.17: the 3 coefficients of the plane π and the scale factor a. This can be logically explained since the fundamental matrix has 7 degrees of freedom (9 elements - 1 for scale - 1 for rank 2 constraint) and a projection matrix has 11 degrees of freedom (12 elements - 1 for scale). This means that Eqs. 3.16, 3.17 can be used to initialize a projective frame for the reconstruction but cannot be employed to compute the projection matrices of consecutive images. If one were to use Eqs. 3.16, 3.17 to initialize the second and third image, then this projective reconstruction would again be defined up to 4 degrees of freedom and thus would not be in the same projective frame as the first two cameras.

Selecting the Best Initialization Pair

In practice, it is not always a good idea to choose the first two images in the sequence for initialization. Other image pairs might be better suited. Which

criterion could one use to determine if an image pair is better suited for initialization of the projective frame than others? Such a pair must fulfill two requirements. First, many correspondences must exist between the two images, in order to start with plenty of 3D reconstructed points. Secondly, the two images should not be too close together for well-conditioned initialization. The first requirement is easy to check. The second is harder because in a projective frame we cannot use metric qualities like the distance between cameras. We use the following algorithm.

- Compute the best fitting homography between every consecutive image pair.
- Compute the median residual error of every match w.r.t. this homography.

The homography computation finds the homography that fits the correspondences between two images best in a least squares sense, as explained in paragraph 3.4.1. No RANSAC procedure is used, just one computation of $\mathbf{H}$. The median residual error is found as the median of all distances between points in the second image and the transformation via $\mathbf{H}$ of the matching point in the first image.

$$median\{D(\mathbf{H}\mathtt{m}_\mathtt{i}, \mathtt{m}_\mathtt{j})\}$$

For every consecutive image pair a combination of the number of matches and the median residual error is computed and the image pair which maximizes this value is selected for initialization. In our implementation we opted for a simple multiplication of the number of matches and the residual.

Triangulation

When the initializing cameras are chosen and the first two cameras are computed with Eq. 3.16, 3.17, all correspondences we found in the F-guided matching step can be reconstructed in the initialized frame. Fig. 3.8 illustrates the process. Due to noise and the margin we allowed around the epipolar line in the F-guided matching step, it is not guaranteed that the two backprojected rays lie in the same plane. Which 3D point must we choose? In a metric frame this can be done by computing the closest 3D point to the two rays (M in Fig. 3.8). In a projective frame we cannot use a metric quality like distances in 3D. The only measure we can safely minimize is the distance in the images. Two options are available to us. Either we first compute the two 2D points $\mathtt{m}_{\mathtt{1h}}$ and $\mathtt{m}_{\mathtt{2h}}$ which are the points closest to the original points $\mathtt{m}_1$ and $\mathtt{m}_2$ but this time lying on each others epipolar line. This *Hartley-Sturm* adaptation of the points guarantees that the backprojected lines coincide. Another approach is simpler but effective. We search for the 3D point which, projected into the images with $\mathbf{P}_1$ and $\mathbf{P}_2$, lies as closely as possible to $\mathtt{m}_1$ and $\mathtt{m}_2$ respectively. This means

$$\begin{aligned} \mathtt{m}_1 &\simeq \mathbf{P}_1\mathtt{M} \\ \mathtt{m}_2 &\simeq \mathbf{P}_2\mathtt{M} \end{aligned}$$

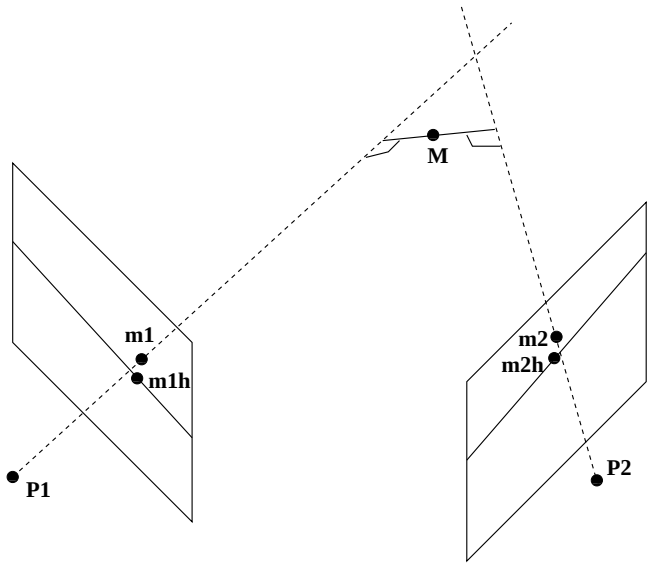


Figure 3.8: Triangulating 2D correspondence $\mathtt{m}_1$, $\mathtt{m}_2$ to yield the 3D point $\mathtt{M}$. In a metric frame we can minimize the 3D distance to the backprojected rays. In a projective frame we have to minimize the distance of the projection of $\mathtt{M}$ to $\mathtt{m}_1$ and $\mathtt{m}_2$

This can be written linearly for both points $\mathtt{m}_1 = (x, y, w)^T$, $\mathtt{m}_2 = (x', y', w')^T$ and both projection matrices $\mathbf{P}_1 = (P_1, P_2, P_3)^T$ and $\mathbf{P}_2 = (P'_1, P'_2, P'_3)^T$ as

$$\begin{array}{ll} P_3\mathtt{M}x = P_1\mathtt{M} & P_3\mathtt{M}y = P_2\mathtt{M} \\ P'_3\mathtt{M}x' = P'_1\mathtt{M} & P'_3\mathtt{M}y' = P'_2\mathtt{M} \end{array} \tag{3.18}$$

If we follow the same approach we used to compute $\mathbf{F}$ and $\mathbf{H}$, we can compute $\mathtt{M}$ linearly by computing the SVD of the matrix $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} wp_{11} - xp_{31} & wp_{12} - xp_{32} & wp_{13} - xp_{33} & wp_{14} - xp_{34} \\ wp_{21} - yp_{31} & wp_{22} - yp_{32} & wp_{23} - yp_{33} & wp_{24} - yp_{34} \\ w'p'_{11} - x'p'_{31} & w'p'_{12} - x'p'_{32} & w'p'_{13} - x'p'_{33} & w'p'_{14} - x'p'_{34} \\ w'p'_{21} - y'p'_{31} & w'p'_{22} - y'p'_{32} & w'p'_{23} - y'p'_{33} & w'p'_{24} - y'p'_{34} \end{bmatrix} \tag{3.19}$$

where $\mathbf{P}_1 = [p_{ij}]$ and $\mathbf{P}_2 = [p'_{ij}]$.

Actually we don't want to minimize the algebraic error of Eq. 3.18 but are more interested in minimizing the true reprojection error $\sum D(\mathbf{P}_i\mathtt{M}, \mathtt{m}_\mathtt{i})$. This boils down to finding $\mathtt{M}$ which minimizes

$$E = \left(x - \frac{P_1\mathtt{M}}{P_3\mathtt{M}}\right)^2 + \left(y - \frac{P_2\mathtt{M}}{P_3\mathtt{M}}\right)^2 + \left(x' - \frac{P'_1\mathtt{M}}{P'_3\mathtt{M}}\right)^2 + \left(y' - \frac{P'_2\mathtt{M}}{P'_3\mathtt{M}}\right)^2 \tag{3.20}$$

This non-linear computation can be performed by computing the solution of Eq. 3.18, which we call $\tilde{\mathtt{M}}$. Then we solve another system of equations:

$$\begin{array}{cc} \frac{P_3\mathtt{M}x}{P_3\tilde{\mathtt{M}}} = \frac{P_1\mathtt{M}}{P_3\tilde{\mathtt{M}}} & \frac{P_3\mathtt{M}y}{P_3\tilde{\mathtt{M}}} = \frac{P_2\mathtt{M}}{P_3\tilde{\mathtt{M}}} \\ \frac{P'_3\mathtt{M}x'}{P'_3\tilde{\mathtt{M}}} = \frac{P'_1\mathtt{M}}{P'_3\tilde{\mathtt{M}}} & \frac{P'_3\mathtt{M}y'}{P'_3\tilde{\mathtt{M}}} = \frac{P'_2\mathtt{M}}{P'_3\tilde{\mathtt{M}}} \end{array} \tag{3.21}$$

Equations 3.18 are not the same as Eq. 3.21 because the top two equations are divided by a different scale factor ($P_3\tilde{\mathtt{M}}$) than the bottom two ($P'_3\tilde{\mathtt{M}}$). Iterating this approach a number of times will let $\mathtt{M}$ and $\tilde{\mathtt{M}}$ converge and finally yield the solution which minimizes the true reprojection error.

3.6.2 Projective Pose Estimation

After initialization we can start adding other cameras to the projective frame. For this we use the matches we have computed between consecutive images using the F-RANSAC and F-guided matching algorithms. The situation we are in now is depicted in Fig. 3.9. In this figure we have three images with corresponding projective projection matrices and a set of 3D points in the same projective frame with corresponding 2D points. We also have matches between the last image and the new image which we want to add to the projective frame.

For every match between points $\mathtt{m_{i-1}}$ and $\mathtt{m_i}$ in the last image $i-1$ and the new image i we check if the point $\mathtt{m_{i-1}}$ has a corresponding 3D point $\mathtt{M}$. If it does, the 3D-2D correspondence $\langle \mathtt{M}, \mathtt{m_i} \rangle$ is a possibly correct piece of information which we can employ to compute the projection matrix $\mathbf{P}_i$ in the existing projective frame. If we have enough of these 3D-2D correspondences, the projection matrix can be computed linearly. Every 3D-2D correspondence yields 2 equations: the x and y coordinate of the projection of the 3D point in the image ($\mathbf{P}_i\mathtt{M}$) must coincide with the x and y coordinate of the point $\mathtt{m_i}$. Since a projection matrix has 11 degrees of freedom (12 elements - 1 for scale) and every correspondence yields 2 equations, we need 6 correspondences to compute $\mathbf{P}_i$ linearly (actually we only need 5 plus one equation in either x or y of a sixth). Of course, we will have to deal with possible outliers again. Since the fundamental matrix only checks if corresponding points are on each other's epipolar line, possible mismatches along this line can still appear. Thus wrong 2D-2D matches between $i-1$ and $i-2$ might have been reconstructed or wrong 2D-2D matches between i and $i-1$ appear. Both give rise to wrong 3D-2D correspondences. This means we will have to use the RANSAC paradigm once again to distinguish the inliers from the outliers and select the projection matrix with the largest support.

The linear algorithm in the inner RANSAC loop again solves for the SVD of a matrix $\mathbf{A}$. This time the matrix is stacked with 2 rows per 3D-2D correspondence $\mathtt{M} = (X, Y, Z, W)^T$, $\mathtt{m} = (x, y, w)^T$.

$$\begin{bmatrix} wX & wY & wZ & wW & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -xW \\ 0 & 0 & 0 & 0 & wX & wY & wZ & wW & -yX & -yY & -yZ & -yW \end{bmatrix}$$

RANSAC also needs a criterion to check if, given a computed P-matrix, a correspondence is an inlier or an outlier. A naive approach would be to simply

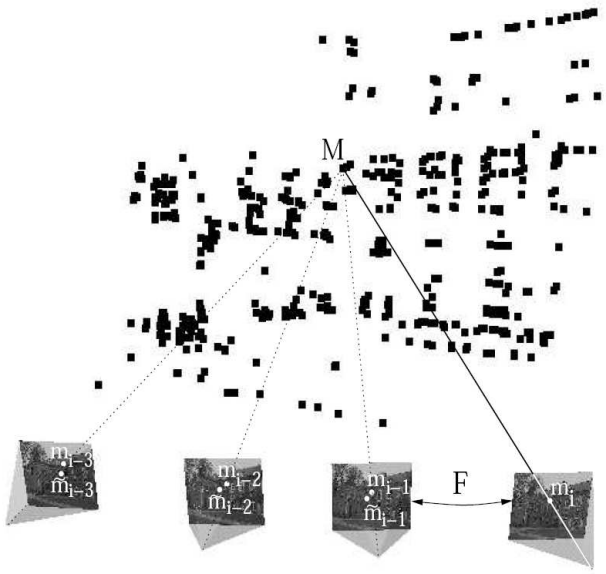


Figure 3.9: The 3D point $\mathtt{M}$ has been seen in and reconstructed from images $i-3$ to $i-1$ where it projects into $\mathtt{m}_{i-3}$ to $\mathtt{m}_{i-1}$. An F-RANSAC and F-Guided matching algorithm between images $i-1$ and i delivers a set of matches between these images. The match $\langle \mathtt{m}_{i-1}, \mathtt{m}_i \rangle$ provides a possible 3D-2D correspondence $\langle \mathtt{M}, \mathtt{m}_i \rangle$. If enough of these correspondences are available, the projection matrix $\mathbf{P}_i$ can be computed in the existing frame using RANSAC.

project the 3D point $\mathtt{M}$ into the new image i and check if the distance between the projection and the 2D point, $D(\mathbf{P}_i\mathtt{M}, \mathtt{m}_i)$, is below a threshold. This is not a good idea. All incorrect 3D points lying on or close to the backprojected ray through $\mathtt{m}_i$ would then always be judged to be inliers.. An alternative could be to triangulate the 3D point from $\mathtt{m}_i$ and $\mathtt{m}_{i-1}$ using $\mathbf{P}_i$ and $\mathbf{P}_{i-1}$ and then check the distance between the reconstructed point and the original point $\mathtt{M}$. In order to do so, however, we would again need to measure the distance between points, which we cannot use because our reconstruction is only valid up to any projective transformation. Hence, we will need to employ information in all other images. We backproject the 3D point again but this time using the 2D points in all images ($\mathtt{m}_i$, $\mathtt{m}_{i-1}$, $\mathtt{m}_{i-2}$, ...) and all projection matrices ($\mathbf{P}_i$, $\mathbf{P}_{i-1}$, $\mathbf{P}_{i-2}$, ...). We use Eq. 3.19 but this time with a matrix $\mathbf{A}$ with more than 4 rows, namely 2 times the number of images, including the new image i. This clearly is an overdetermined system and solving it will yield the 3D point $\tilde{\mathtt{M}}$ that minimizes the projection-residual in the images in a least-squares way. We then check the residual errors $D(\mathbf{P}\tilde{\mathtt{M}}, \mathtt{m})$ in all images. Only if this residual is below a threshold in all images will we accept the 3D-2D correspondence as an inlier for the computed $\mathbf{P}_i$. This algorithm only measures distances in the images and

not in 3D space. Since the images are our observations of the world, feature points and projections of 3D points should coincide in them and measurements of discrepancies between their locations are allowed.

3.6.3 Updating Structure

Every time a new projection matrix has been added, the structure of 3D points must be updated. Every new projection matrix has been computed making use of 3D-2D correspondences. Once $\mathbf{P}_i$ is known, we can update the position of the existing 3D points if a new observation of them is available in the form of the correspondence $< \mathtt{M}, \mathtt{m_i} >$. Eq. 3.19 with the extended matrix $\mathbf{A}$, containing rows from all images the point is visible in, is used again to compute the new position of $\mathtt{M}$. It is best to use the iterative scheme of Eq. 3.21 to minimize the true reprojection error.

Feature matches will exist between image i and $i-1$ that have no 3D-2D correspondence, i.e. for which no consistent match existed between $i-1$ and $i-2$. These points can now be instantiated in the projective frame using triangulation with projection matrices $\mathbf{P}_{i-1}$ and $\mathbf{P}_i$. This instantiation is vital because otherwise only points visible in the first images will be used for pose-estimation. Once they would fall out of view, new cameras could not be added anymore.

3.6.4 Global Minimization

Once the structure and motion has been obtained for the whole sequence, it is recommended to refine it through a global minimization step. A maximum likelihood estimation can be obtained through bundle adjustment [93, 81]. The extracted image points are the only piece of information that can be assumed to be fixed. The parameters of the different cameras and the reconstructed 3D points have been computed from them and the goal is to find the projection matrices and 3D points for which the mean squared distances between the observed image points and the projected image points is minimized. For m views and n points the following criterion should be minimized:

$$\sum_{k=1}^{m}\sum_{i=1}^{n} D(\mathtt{m}_{ki}, \mathbf{P}_k \mathtt{M}_i)^2$$

If the image error is zero-mean Gaussian then bundle adjustment is the Maximum Likelihood Estimator. Although it can be expressed very simply, this minimization problem is huge. For a typical sequence of 20 views and 2000 points, a minimization problem in more than 6000 variables has to be solved. A straight-forward computation is obviously not feasible. However, the special structure of the problem can be exploited to solve the problem much more efficiently. The observed points $\mathtt{m}_{ki}$ being fixed, a specific residual $D(\mathtt{m}_{ki}, \mathbf{P}_k \mathtt{M}_i)^2$ is only dependent on the i-th point and the k-th camera view. This results in a

sparse structure for the normal equations. Using this structure the points $\mathtt{M}_i$ can be eliminated from the equations, yielding a much smaller but denser problem. Views that have features in common are now related. For a long sequence where features tend to only be seen in a few consecutive views, the matrix that has to be solved is still sparse (typically band diagonal). In this case it can be very interesting to make use of sparse linear algebra algorithms, e.g. [6].

As we have seen in paragraph 2.1.3, images can suffer from radial distortion. If possible, the camera projection model should also take this distortion into account. Unfortunately, as explained in paragraph 2.1.3 the camera model proposed in Eq. A.6, A.7 puts the radial distortion parameters between the extrinsic and linear intrinsic parameters of the camera. At this moment we have only computed a projective reconstruction of the scene and cameras and we have no clue what the intrinsics of the cameras are. Therefore, it is impossible to estimate the radial distortion parameters with the proposed model. A way around this is to use a crude estimation of the internal parameters matrix $\mathbf{K}_e$ for the time being. This matrix can be defined as

$$\mathbf{K}_e = \begin{bmatrix} \frac{w+h}{2} & 0 & \frac{w}{2} \\ 0 & \frac{w+h}{2} & \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

with w and h the width and height of the image in pixels. The radial distortion is then applied as follows. The matrix $\mathbf{P}'$ is defined as

$$\mathbf{P}' \simeq \mathbf{K}_e^{-1}\mathbf{P}$$

The 3D point is projected via $\mathbf{P}'$. There the radial distortion formula of Eq. A.6 is applied and the resulting point is multiplied on the left with $\mathbf{K}_e$ again.

To end this section, a few more notes on the parametrization of the cameras are due. Projective cameras can easily be parameterized using 11 parameters. One can determine the largest element (in absolute value) in the 3×4 projection matrices, store this one for reference and use the 11 remaining elements to parameterize the matrix. Once the intrinsic parameters of the cameras are known, we can execute a Euclidean bundle adjustment. Euclidean cameras, however, are more difficult to deal with. First there is the matter of the internal parameters. For the focal length, the aspect ratio, the principal point and the skew, three situations are possible

- *known* value, e.g. we know that the skew of a digital camera is 0
- *fixed* value, e.g. we know that a camera with the same lens has the same radial distortion parameters for all pictures
- *free* value, e.g. a sequence where the operator zoomed in during recording has a different focal length for every image.

Depending on the specific situation, another choice can be made. For instance when the user has not touched the zoom, the focal length and the principal point

can be considered fixed. Finally there is the matter of the extrinsic parameters of a camera. These comprise the rotation and translation. The latter is easily described using 3 coordinates but the former is harder to parameterize. The most comfortable way to describe a rotation is by means of a rotation matrix. Such a matrix, however, contains 9 elements although a rotation only holds 3 degrees of freedom. Hence, it is not a good idea to employ rotation matrices to parameterize rotations in a minimization process. Another possibility is the use of Euler angles. This representation sometimes suffers from so-called *gimbal lock*, a situation in which two rotational axes of an object point in the same direction[2]. A way out is to use *quaternions*. These are vectors with 4 elements with norm 1. This is still one degree of freedom too much. If the initialization for the bundle adjustment is good enough, we can follow the so-called *small-angle* approach, where the orientation of a camera is described with three deviations (small angles) from the original orientation. This approach has the advantage of using the minimum amount of parameters and the derivatives of the error w.r.t. these parameters are quite easy to express [91].

3.7 The Projective Ambiguity

The images have been related to each other in a projective frame. After the projective bundle adjustment step, we have recovered the projective projection matrices $\mathbf{P}_k$ for every image and a set of 3D points $\mathtt{M}_i$ that, when projected via those projection matrices, result in the smallest residual error w.r.t. the extracted feature points $\mathtt{m}_{ki}$.

Unfortunately, since the only true information we have consists of these image points, the resulting reconstruction is only valid up to any projective transformation in 3D. As we have derived in paragraph A.4.3: for every image point that is a projection of a 3D point, the equation

$$\mathtt{m}_{ki} \simeq \mathbf{P}_k \mathtt{M}_i$$

must hold. Inserting an arbitrary 4×4 projective transformation yields the same image point:

$$\begin{aligned} \mathtt{m}_{ki} &\simeq \mathbf{P}_k \mathbf{T} \mathbf{T}^{-1} \mathtt{M}_i \\ \mathtt{m}_{ki} &\simeq (\mathbf{P}_k \mathbf{T})(\mathbf{T}^{-1} \mathtt{M}_i) \\ \mathtt{m}_{ki} &\simeq \mathbf{P}'_k \mathtt{M}'_i \end{aligned}$$

and thus the solution consisting of projection matrices $\mathbf{P}'_k$ and 3D points $\mathtt{M}'_i$ is equally valid and cannot be discerned from other reconstructions based on image feature measurements only.

[2]The reason for this is that Euler angles evaluate each axis independently in a set order. For instance, first the object travels down the X axis. When that operation is complete it then travels down the Y axis, and finally the Z axis. The problem with gimbal lock occurs when one rotates the object down the Y axis, say 90 degrees. Since the X component has already been evaluated it doesn't get carried along with the other two axes. What winds up happening is the X and Z axis get pointed down the same axis.



Figure 3.10: One of the images of a sequence taken in Mechelen. The grey wall on the right is part of a medieval building.

Because the reconstruction can be transformed by any projective transformation and still be valid, this means that many properties (like orthogonality, relative distances, parallelism) are not preserved. For the reconstruction result to be useful, we need to upgrade it from projective to metric, a process called *self-calibration*. Fig. 3.10 shows one of the images of a sequence that was recorded in the town of Mechelen in Belgium. It shows part of the remaining wall of a medieval building (on the right), called a "Spijker" which was originally a warehouse but performed many other functions later. The resulting projective reconstruction is shown in the top row of Fig. 3.11. It is clear that, although projective properties like planarity are preserved, this reconstruction is unusable. We want to upgrade it to metric, the result of which is shown in the bottom row. The projective deformation is actually quite mild in this example because the plane at infinity is already quite far from the reconstruction compared to its size. Some views of the reconstruction, deformed by another projective transformation where the plane at infinity is placed elsewhere, are shown in Fig. 3.12. The reconstructed scene nor the cameras can be recognized in these views, which shows the devastating effects a projective transformation can have.

How can the upgrade from a projective to a metric reconstruction be done? Different techniques can be employed, depending on the information one disposes of. In paragraph 3.8 we will explain how self-calibration can be performed if certain constraints on the scene are known. Paragraph 2.2.1 and 2.2.2 explained how to calibrate intrinsics and extrinsics of cameras. If no information on the scene or camera motion is known, a self-calibration technique which imposes constraints on the intrinsics of the cameras can be used. This is explained in paragraph 3.9.

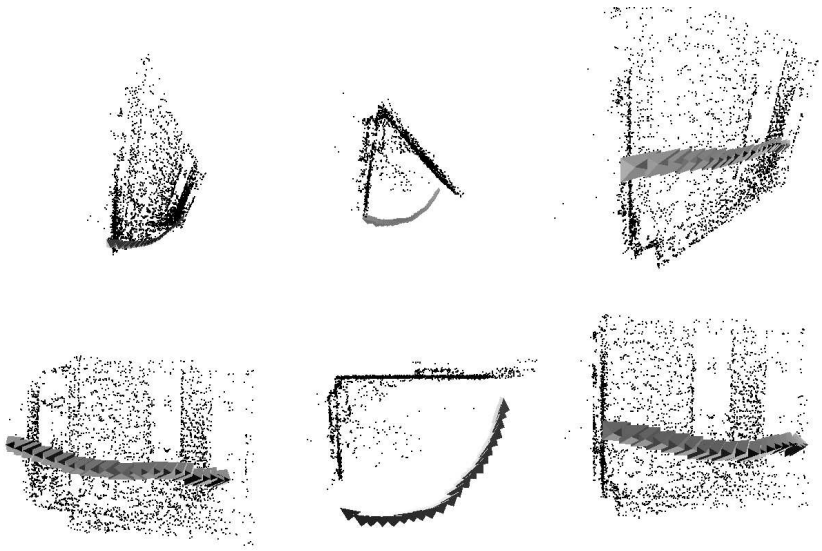


Figure 3.11: Top row: Projectively deformed reconstruction, output of the projective structure and motion algorithm. Planar regions are still planar but metric quantities like angles and (relative) distances are clearly not preserved. Bottom row: Metric reconstruction, obtained via self-calibration.

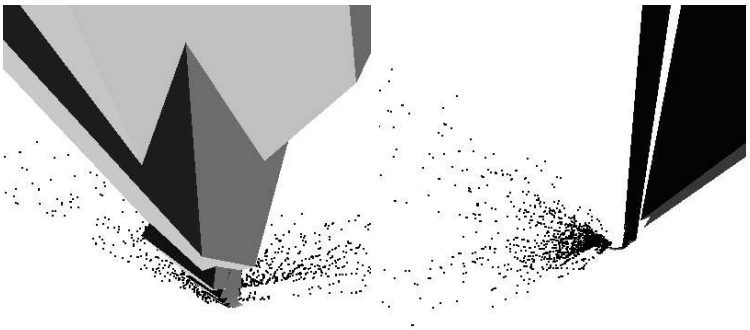


Figure 3.12: Two views of the reconstruction of the sequence of the wall in Mechelen, deformed by another projective transformation. Scene nor cameras can be recognized in these views. Although the 3D looks nothing like reality, still all projections of 3D points into the images via the projection matrices coincide neatly with the extracted feature points.

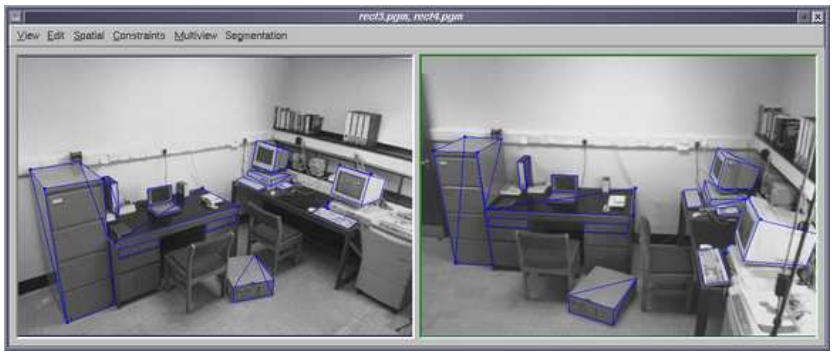


Figure 3.13: Scene-constraints are indicated on the images via a GUI.

3.8 Scene Constraints

We will first discuss how knowledge of the scene can be exploited to perform the upgrade from a projective to a metric reconstruction. The goal is to find new projection matrices $\hat{\mathbf{P}}_k$ and 3D points $\hat{\mathtt{M}}_i$ for which certain constraints imposed on these hold while minimizing the mean squared distances between the observed image points $\mathtt{m}_{ik}$ and the reprojected image points $\hat{\mathtt{m}}_{ik}$. The constraints we can think of here are based on properties that are only preserved in reconstructions, valid up to a metric transformation, such as orthogonality of lines or planes, parallelism of lines or planes (actually an affine property) and relative distances between point pairs. Imposing these constraints means that one has to have knowledge of where they apply. One could think of techniques that search for these properties automatically but here we will assume that a human operator indicates the constraints via a GUI. When the different constraints have been indicated, a minimization algorithm is run which imposes these constraints.

Calibration based on scene constraints is a time-consuming process because all constraints have to be identified by an operator. Moreover, there is a risk of constraints being incompatible. However, in some cases we have no other choice. For instance, if only two images of a scene are given, taken by two cameras with different intrinsic parameters, the techniques that will be explained in paragraph 3.9 and that don't exploit any such constraints cannot be applied. An example is shown in Fig. 3.13, depicting an indoor scene, filmed by two surveillance cameras. The two images are far apart, so for matching them wide-baseline matching techniques have to be used or manual interaction is necessary. Superimposed on the images Fig. 3.13 shows the information the operator has provided. Parallel and orthogonal planes and lines have been indicated. A projective reconstruction of indicated regions, based on the recovered epipolar geometry can be seen in Fig. 3.14 on the left. The constraints, indicated in Fig. 3.13 have been used to upgrade this projective reconstruction to metric, the result of which is shown on the right in Fig. 3.14.

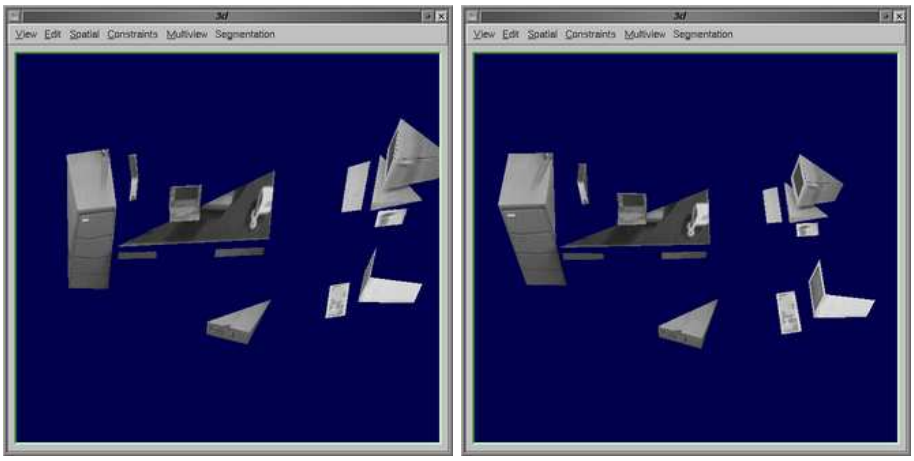


Figure 3.14: Projective (left) and metric reconstruction of the indoor scene. The metric reconstruction is found by upgrading the projective one via a global minimization which takes into account the indicated constraints.

3.9 Self-Calibration

In the previous paragraph we explained how to upgrade the reconstruction from projective to metric using scene constraints. Manual interaction is necessary for this approach. In this section an approach will be given which allows for completely automatic self-calibration. If we cannot employ constraints on the scene (because this is a manual process) or on the camera path (because it was not measured), then the only way to upgrade the reconstruction to metric is by imposing constraints on the intrinsic parameters of the camera. Indeed, in the projective reconstructions all cameras are described by projection matrices. This description does not take into account the fact that a real-life camera took those images and that we have some extra information: typically it is the same camera that was used to record the sequence; the camera might be digital (with zero skew); etc.

After a brief overview of the equations that govern conics and quadrics in projective geometry, we introduce a specific conic (the *absolute conic*) that can be used for self-calibration. Afterwards we will explain a practical algorithm and give more information on how to self calibrate different sequences, recorded with the same camera.

3.9.1 The Absolute Conic

In projective 2D and 3D geometry quadratic expressions of homogeneous points amount to the concept of conics and quadrics respectively.

A 2D point $\mathtt{m}$ belongs to a conic $\mathbf{C}$ and a 3D point $\mathtt{M}$ to a quadric $\mathbf{Q}$ if

respectively

$$\mathtt{m}^T\mathbf{C}\mathtt{m} = 0 \quad (3.22)$$
$$\mathtt{M}^T\mathbf{Q}\mathtt{M} = 0 \quad (3.23)$$

where the conic is represented by a 3×3 matrix $\mathbf{C}$ and the quadric by a 4×4 matrix $\mathbf{Q}$. The dual $\mathbf{C}^*$ of the conic and $\mathbf{Q}^*$ of the quadric can be defined as the sets of 2D lines $\mathtt{l}$ and 3D planes $\mathtt{\Pi}$ that have to satisfy respectively

$$\mathtt{l}^T\mathbf{C}^*\mathtt{l} = 0 \quad (3.24)$$
$$\mathtt{\Pi}^T\mathbf{Q}^*\mathtt{\Pi} = 0 \quad (3.25)$$
$$(3.26)$$

The transformation of a conic or its dual with a projective 2D transformation $\mathbf{H}$ can be written as

$$\mathbf{C}' \simeq \mathbf{H}^{-T}\mathbf{C}\mathbf{H}^{-1} \quad (3.27)$$
$$\mathbf{C}^{*\prime} \simeq \mathbf{H}\mathbf{C}^*\mathbf{H}^T \quad (3.28)$$

The projection of a dual quadric $\mathbf{Q}^*$ in a camera with projection matrix $\mathbf{P}$ yields a dual conic $\mathbf{C}^*$ defined as

$$\mathbf{C}^* \simeq \mathbf{P}\mathbf{Q}^*\mathbf{P}^T \quad (3.29)$$

We can employ these concepts in the self-calibration process. As explained before we want to impose constraints on the intrinsic parameters of the cameras. Unfortunately, the projection matrices we have recovered are only valid up to an arbitrary projection matrix, making it impossible to separate the intrinsics from the extrinsics with the QR-technique described in paragraph 2.1.1.

In literature, the *absolute conic* was introduced because of its specific form. The *absolute conic* is a conic which resides in the plane at infinity. For the sake of simple notation, we typically employ its dual, the *dual absolute quadric* (DAQ)[3] [94] which we denote with the symbol $\mathbf{\Omega}^*$. In Euclidean space, this DAQ is defined as

$$\mathbf{\Omega}^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.30)$$

Projecting this dual quadric into an image with a Euclidean pinhole camera, as defined in Eq. 2.1, yields the *image of the dual absolute quadric* ω^* which is computed with Eq. 3.29 and the fact that a rotation matrix is orthonormal

$$\omega^* = \mathbf{P}\mathbf{\Omega}^*\mathbf{P}^T \quad (3.31)$$

[3]The name *dual* absolute quadric is in fact an abuse of terminology. Since the absolute conic is not a quadric, the DAQ is not strictly spoken a dual quadric. However, since the DAQ is constructed as the set of tangent planes to the absolute conic which resides in the plane at infinity, the term *dual* is more or less applicable.

$$\begin{aligned}
&\simeq \mathbf{K}\left[\mathbf{R}^T|-\mathbf{R}^T\mathbf{t}\right]\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0\end{bmatrix}\left[\mathbf{R}^T|-\mathbf{R}^T\mathbf{t}\right]^T\mathbf{K}^T\\
&\simeq \mathbf{K}\mathbf{R}\mathbf{R}^T\mathbf{K}^T\\
&\simeq \mathbf{K}\mathbf{K}^T
\end{aligned} \tag{3.32}$$

In other words, the projection of the dual absolute quadric is described by parameters of the internal calibration matrix of the projection matrices only.

In reality, the projection matrices we have recovered with projective structure and motion are only valid up to an arbitrary projective transformation $\mathbf{T}$ and so is the recovered 3D structure of this space. Hence, the DAQ is not in its canonical position given by Eq. 3.30 but in a transformed position

$$\mathbf{\Omega}^{*\prime} \simeq \mathbf{T}\mathbf{\Omega}^*\mathbf{T}^T \tag{3.33}$$

However, since the images were taken in the real world, the projection of the dual absolute quadric $\mathbf{P}\mathbf{\Omega}^{*\prime}\mathbf{P}^T$ is still in the same position given by Eq. 3.32 because $\mathbf{P}' = \mathbf{P}\mathbf{T}^{-1}$

$$\begin{aligned}
\omega^{*\prime} &\simeq \mathbf{P}'\mathbf{\Omega}^{*\prime}\mathbf{P}'^T\\
&\simeq (\mathbf{P}\mathbf{T}^{-1})(\mathbf{T}\mathbf{\Omega}^{*\prime}\mathbf{T}^T)(\mathbf{T}^{-T}\mathbf{P}^T)\\
&\simeq \mathbf{P}(\mathbf{T}^{-1}\mathbf{T})\mathbf{\Omega}^{*\prime}(\mathbf{T}^T\mathbf{T}^{-T})\mathbf{P}^T\\
&\simeq \mathbf{P}\mathbf{\Omega}^*\mathbf{P}^T\\
&\simeq \omega^*
\end{aligned}$$

In other words, we will employ the following steps in our self-calibration approach.

- Select a set of constraints on the internal parameters $\mathbf{K}$ of the camera.
- Translate these constraints to constraints on $\mathbf{\Omega}^*$ itself via Eq. 3.31.
- If enough constraints are given, $\mathbf{\Omega}^*$ (9 linearly independent elements) can be computed.
- When $\mathbf{\Omega}^*$ is known, the transformation$\mathbf{T}$ between it and the canonical position of Eq. 3.30 is readily computed from Eq. 3.33.
- The inverse of this transformation is exactly the transformation that brings our projective reconstruction into metric space!

3.9.2 Practical Constraints

The algorithm described above allows us to translate constraints on the intrinsic parameters to constraints on the DAQ and in this way to compute the transformation that brings us to metric space. The question now is which constraints

are suitable and how to formulate them, so the DAQ is easy to compute. Because of the stability and efficiency of SVD-based methods, we would like to impose constraints that lead to linear equations in the elements of $\mathbf{\Omega}^*$. How this can be done is explained in this paragraph.

We begin with normalizing the projection matrices such that the focal length is in the order of 1 and the principal point is close to the origin. We obviously don't know the focal length yet, so the following normalization is proposed:

$$\mathbf{P}_N = \mathbf{K}_N^{-1}\mathbf{P}$$

with

$$\mathbf{K}_N = \begin{bmatrix} w+h & 0 & \frac{w}{2} \\ 0 & w+h & \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

with w and h the width and height of the image. The principal point (u, v) typically lies near the center of the image, so the normalization is fine in that respect. The focal length (f) deserves some more explanation. If we approximate it with $w + h$, then this corresponds to a camera with a (35mm equivalent) focal length of 63mm, or with an opening angle of about 30 degrees. If we allow for a deviation of a factor 3, then cameras with focal lengths from 20 to 180mm (82 to 11 degrees) are OK. This covers the range of most used lenses. The aspect ratio (r) is taken to be around 1 and the skew (s) is 0, which is true for all digital cameras. Making this a priori knowledge more explicit and estimating reasonable standard deviations one could for example get $f \approx rf \approx 1 \pm 3$, $u \approx v \approx 0 \pm 0.1$, $r \approx 1 \pm 0.1$ and $s = 0$. It is now interesting to investigate the impact of this knowledge on ω^*:

$$\omega^* \simeq \mathbf{K}\mathbf{K}^T = \begin{bmatrix} f^2+s^2+u^2 & srf+uv & u \\ srf+uv & r^2f^2+v^2 & v \\ u & v & 1 \end{bmatrix} \approx \begin{bmatrix} 1 \pm 9 & \pm 0.01 & \pm 0.1 \\ \pm 0.01 & 1 \pm 9 & \pm 0.1 \\ \pm 0.1 & \pm 0.1 & 1 \end{bmatrix}$$

Comparison of elements of $\mathbf{K}\mathbf{K}^T$ gives us several possible constraints: elements (1,1) and (2,2) should be equal to element (3,3) (focal length = 1) and should be equal to each other (aspect ratio = 1). Elements (1,3) and (2,3) should be zero (principal point in origin), as well as element (1,2) (skew = 0). Since these are constraints on ω^*, we can translate them to constraints on $\mathbf{\Omega}^*$ via Eq. 3.31. The equations should be weighted according to the uncertainty on the result. This gives the following weighted equations:

$$\begin{aligned} \frac{1}{9\nu}\left(P_1\mathbf{\Omega}^*P_1^T - P_3\mathbf{\Omega}^*P_3^T\right) &= 0 \\ \frac{1}{9\nu}\left(P_2\mathbf{\Omega}^*P_2^T - P_3\mathbf{\Omega}^*P_3^T\right) &= 0 \\ \frac{1}{0.2\nu}\left(P_1\mathbf{\Omega}^*P_1^T - P_2\mathbf{\Omega}^*P_2^T\right) &= 0 \\ \frac{1}{0.01\nu}\left(P_1\mathbf{\Omega}^*P_2^T\right) &= 0 \end{aligned} \tag{3.34}$$

$$\begin{aligned} \frac{1}{0.1\nu}\left(P_1\mathbf{\Omega}^* P_3^T\right) &= 0 \\ \frac{1}{0.1\nu}\left(P_2\mathbf{\Omega}^* P_3^T\right) &= 0 \end{aligned}$$

with P_i the ith row of $\mathbf{P}$ and ν a scale factor initially set to 1 and in later iterations to $P_3\mathbf{\Omega}^* P_3$.

An estimate of $\mathbf{\Omega}^*$ can be obtained by solving the above set of equations for all views through linear least-squares. The rank-3 constraint should be imposed by forcing the smallest singular value to zero. This scheme can be iterated until the factors converge.

The practical method that we described here is a linear method. In order for it to return exact results, the imposed constraints must all hold. In practice, this not always the case. The most important parameter to estimate is the focal length and it happens that image sets are recorded with varying focal lengths. Since the linear algorithm assumes all focal lengths to be the same, it only recovers one focal length. This needs then be used as an initialization for a non-linear bundle adjustment process, as described in section 3.6.4, in which the focal lengths for all cameras can be estimated independently.

3.9.3 Coupled Self-Calibration

We will see in later chapters that sometimes different (sub)sequences have been recorded. One could of course self-calibrate the different subsequences separately but if they were recorded with the same camera and the same settings, it is possible to combine the equations derived above into a larger system of equations. Each subsequence has been reconstructed into its own projective frame, thus the $\mathbf{\Omega}^*$ for each subsequence is different. However the fact that the intrinsic parameters of the cameras were the same allows us to solve for the different $\mathbf{\Omega}^*$ while retaining a coupling between the subsequences.

Every subsequence can first be transformed thus that the first camera $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$. From Eq. 3.31, 3.32 we see that we can write

$$\mathbf{\Omega}^* = \begin{bmatrix} \mathbf{K}\mathbf{K}^T & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix}$$

and thus the equations of Eq. 3.34 can be written in matrix form as

$$\begin{bmatrix} \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} \end{bmatrix} = 0$$

with $\mathbf{k}$ a vector containing 6 coefficients representing the matrix $\mathbf{K}\mathbf{K}^T$, a is a 3-vector and b a scalar and $\mathbf{C}$ and $\mathbf{D}$ are matrices containing the coefficients of the equations. Note that this can be done independently for every 3D subsequence.

If the sequence is recorded with constant intrinsics, the vector $\mathbf{k}$ will be common to all subsequences and one obtains the following coupled self-calibration

equations:

$$\begin{bmatrix} \mathbf{C}_1 & \mathbf{D}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{C}_2 & \mathbf{0} & \mathbf{D}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}_n & \mathbf{0} & \mathbf{0} & \dots & \mathbf{C}_n \end{bmatrix} \begin{bmatrix} \mathbf{k} \\ \begin{bmatrix} \mathbf{a}_1 \\ b_1 \end{bmatrix} \\ \begin{bmatrix} \mathbf{a}_2 \\ b_2 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \mathbf{a}_n \\ b_n \end{bmatrix} \end{bmatrix} = 0 \tag{3.35}$$

The most important feature is that through the coupling this approach solves for the intrinsics using all information together which allows to get good results even for the shorter subsequences. When $\mathbf{\Omega}^*$ has been computed for every subsequence, they can each be upgraded to metric by computation of the transformation that brings it to its canonical position.

3.10 Multi-View Camera Calibration

3.10.1 Introduction

If one wants to capture the 3D information of non-rigid, moving objects or persons in a passive way, we have no other choice than to employ more than one camera observing the scene. These cameras record the moving object from different positions simultaneously and in a synchronized manner. From the combination of these images, a 3D reconstruction of the object can be made for all time instances, effectively creating a 4D reconstruction.

Several techniques exist to compute such reconstructions. A well known technique is called *Shape from silhouettes* and employs the *visual hull* algorithm. The idea behind this algorithm is shown in Figure 3.15. The object (in this case a teapot) is recorded from several viewing positions. A background-foreground segmentation is created. From this segmentation, we know the area in every image in which the object is projected. The backprojection of the foreground area from one image yields an irregular cone-shaped volume in space in which the object has to reside, as shown in the second image of Figure 3.15. The same reasoning holds for a second view, which means that the foreground object must reside in a space delimited by the intersection of both backprojected cone-shaped volumes. The more views are added to the setup, the closer the resulting volume (called the *visual hull*) resembles the real object.

3.10.2 Setup

The camera setup of a system designed to compute 3D reconstructions of articulated objects of course depends on the specifics of the input data. If one is e.g. primarily interested in the dense reconstruction of the face of a moving person, several cameras will be targeted on the space where we expect this face

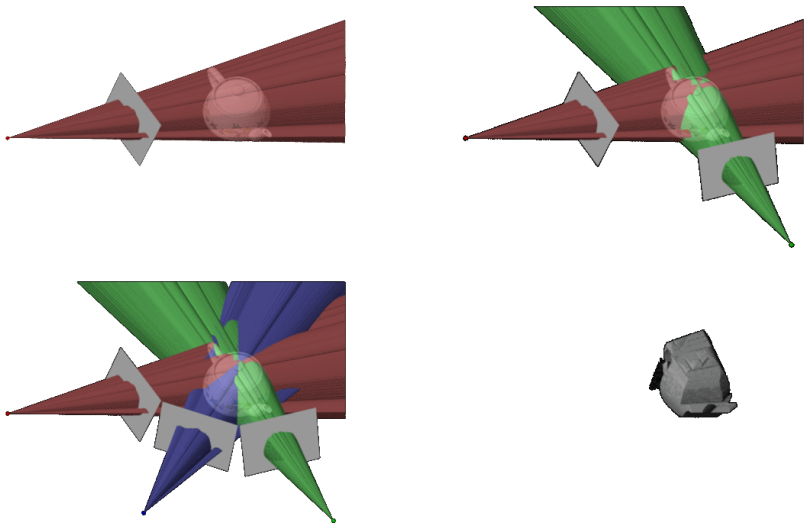


Figure 3.15: Visual Hull Principle: An object is projected into several images. The backprojection of the foreground area in one image yields a cone-shaped volume in space in which the object resides. The intersection of the volumes from all images yield the visual hull: an approximation of the original object.

to appear. If the person is known to always face approximately the same way (e.g. while teaching), several cameras will be recording the front of the person while only a few others perpendicularly to these in order to recover a rough 3D shape in these directions.

No matter how the cameras are set up, the calibration of these cameras is a vital prerequisite for the visual hull computation. A straightforward calibration mechanism consists of making use of a 2D or 3D calibration pattern and apply Tsai's or Zhang's calibration algorithm on the cameras. However, we need the external calibration of all cameras w.r.t. to one coordinate system. This means that the calibration pattern should preferably be visible in all cameras at the same time in order to calibrate them in the same coordinate system. If the cameras are looking from very diverse positions, this is not trivial to achieve.

Therefore, Svoboda et al. [89] proposed a calibration algorithm which makes use of a simple moving laser pointer the projection of which is detected in the different cameras. This yields automatic point matches between the cameras which can be employed to calibrate the cameras. In our lab we have implemented a similar approach, with a different self-calibration part, which makes use of the theory described in section 3.9. The calibration technique was used in visual hull computations [16] and virtual viewpoint computation algorithms [28].

We assume that the cameras are synchronized, either by means of a hardware trigger (yielding the best results) or in software by identifying a specific

event (e.g. turning off the light) in all camera streams. In the latter case, the synchronization is only correct up to half the framerate, 20 ms in the case of PAL video.

3.10.3 Point Detection

Before we explain the calibration algorithm, a short word about the point detection is in place. During the recording for calibration, all light in the lab is extinguished. A person moves the laser pointer in the scene, making sure he covers the entire volume that is visible in the cameras. This is vital for a stable calibration. If an image contains only points located in a small neighborhood, the calibration of the corresponding camera will not be accurate for other parts in the image.

Because the laser pointer is recorded in darkness, extraction of its projection in the images seems a trivial matter. However, due to abrupt or fast motion, the projection of the laser pointer can become several pixels wide. In such a case, typically the center of gravity of the projected area is taken as the projection of the laser pointer. If the motion of the pointer is too fast during the entire recording, the localization of the projections is not accurate which influences the calibration.

3.10.4 Calibration Algorithm

The calibration algorithm for the multi-view camera setup follows an approach very similar to the uncalibrated structure and motion algorithm. From the time synchronization we know that a projection of the laser pointer in one camera at time t corresponds to the projection of the pointer in all other cameras at the same time. In other words, all recorded calibration images yield point correspondences between the cameras. This makes our life much easier since we don't have to search for features and feature matches. A process which would be very hard in the typically wide-baseline multi-camera setups.

The calibration algorithm is outlined below.

1. For every pair of cameras the set of correspondences is used to compute the epipolar geometry robustly with the F-RANSAC algorithm. This process is very fast because of the high inlier ratio of the data (typically around 95 percent)

2. All image pairs are ranked according to the amount of matches

3. The image pair with the highest amount of matches is chosen to initialize the projective structure

4. The image with the largest number of matches to any of the two initial views is chosen as the third view and the projective camera of this camera is computed in the frame of the first two using all inlying correspondences between this view and the first two views.

5. The fourth view is selected in the same way and its projective camera is computed from matches with all already reconstructed views.

6. This process is repeated until no cameras remain to be computed.

7. A projective bundle-adjustment step minimizes the total reprojection error.

8. The self-calibration algorithm of section 3.9 is applied to the projective reconstruction and this reconstruction is upgraded to metric.

9. A final Euclidean bundle-adjustment step minimizes the reprojection error, taking into account other constraints one might want to impose.

Some remarks can be made about this procedure. First, because the amount of cameras is limited (typically up to 20 cameras are used), the fundamental matrix and inlying matches for all image pairs can be computed without ending up in a huge combinatorial problem. Second, due to the well-behaved and stable laser-point detection process, the inlier ratio is typically very high, which improves the speed of the RANSAC estimation of $\mathbf{F}$ and $\mathbf{P}$ very much. Third, for the estimation of the projective camera of a view, we take into account all matches with all already reconstructed cameras. This makes the process much more stable and less prone to error-accumulation. Fourth, the self-calibration procedure assumes the same intrinsic parameters for all cameras, the most important of which is of course the focal length. In reality the cameras can have very different focal lengths. Due to the uncertainty we attribute to the focal length (see Eq. 3.34), this is taken into account. The self-calibration only yields an initialization of the Euclidean reconstruction. The Euclidean bundle-adjustment step starts from this initialization and allows to impose or lift different constraints, like the fact that the focal lengths can differ.

3.10.5 Results

Real Life Calibration

The calibration algorithm described above has been used by several researchers in our institute. Due to the scarcity of cameras, these are employed for several research topics. This means that the setup changes very often, requiring a new calibration every time.

Figure 3.16 shows the result of a calibration of ten cameras in our vision lab. The left image shows the reconstructed cameras. The size of the cameras is directly proportional to the focal length. It is clear that a large variation (a factor of two) in focal length exists in the setup. The right image shows the reconstructed points in 3D without the cameras.



Figure 3.16: Calibration of a real multiview camera setup. The left image shows the reconstructed cameras with a clearly large variation in focal length. The right image shows the reconstructed 3D points (depicted at another scale). The points are clearly connected by curves: the path the laser pointer traveled during recording.

Synthetic Experiment

In order to test the calibration, we generated artificial data. A thousand points were projected in ten cameras with different intrinsic and extrinsic parameters. Two cases were studied: one with and one without radial distortion. For both cases a data-set was generated with the projections as given by the projection equation, and another with added white noise with a standard deviation of 1.15 pixels.

Figures 3.17, 3.18, 3.19, 3.20 and Table 3.3 show the result of this synthetic experiment. It is known in calibration theory that certain camera parameters can be interchanged, in the sense that a change in one can be compensated by a change in another, for example the focal length and the position of the camera in the direction of its optical axis. This means that a simple comparison of calibration parameters is not a perfect measure of the calibration result. Therefore, we interpret the results by comparing the 3D reconstruction of the image projections to the original 3D points used to generate these projections. If these 3D points span the working volume of the system (i.e. the volume in space visible in all or at least several images), the error in the reconstruction is representative of the error on the results of the techniques using the obtained calibration.

The reconstruction of the 3D points is of course only valid up to a metric transformation. This means that we must first search for the metric transformation which brings the reconstruction as close as possible to the original points. The error for every transformed reconstructed point is then calculated as the

Radial Distortion	Noise	Mean Reprojection Error
No Distortion	No Noise	0.001
$\kappa_1 = 2.18$, $\kappa_2 = -42.68$	No Noise	0.02
No Distortion	$\sigma = 1.15$	1.13
$\kappa_1 = 2.18$, $\kappa_2 = -42.68$	$\sigma = 1.15$	1.17

Table 3.3: Mean reprojection error (in pixels) of the final bundle-adjustment step of the several synthetic experiments

distance to the corresponding original point. This distance of course depends on the norm of the original coordinates. That is why we show a relative error, computed as the absolute error divided by the mean norm of the original coordinates.

The first two synthetic experiments deal with *perfect* data, i.e. without influence of noise. The first uses the linear pinhole camera model only. During calibration, the computation of F- and P-matrices does not suffer from non-linear effects and hence finds 100% inliers (as it should). The result is identical to the original data as can be seen in the first row of Table 3.3 and in Figure 3.17 where we notice relative errors of about 1E-4%. The second experiment is already a bit harder because the computation of F- and P-matrices does not take into account radial distortion. This means that the initialization to the bundle adjustment step will not be perfect and this step will have to estimate the non-linear distortion parameters. It succeeds because we obtain a mean reprojection error of 0.02 pixels and relative position errors of 0.03% as seen in Figure 3.18.

For the third and fourth experiment, zero-mean white noise with $\sigma = 1.15$ was added to the projections. The calibration results of course suffer from this noise. However, the mean reprojection error after bundle-adjustment is 1.13 and 1.17 respectively, which is of the same order as the standard deviation of the noise. Figures 3.19 and 3.20 show relative position errors of about 0.5%.

3.11 Conclusion

In this chapter we introduced SaM algorithms and explained how a metric 3D reconstruction can be recovered from images of a scene only. All SaM algorithms need sufficient, well localized, high-quality features that are detected and matched throughout the image set. Standard algorithms, including my own pipeline, for a long time made use of simple features like Harris corners. We found that the very recent scale and rotational invariant features like SURF are superior to those. One still needs to make sure of a good distribution in the image, though.

The two-view setup and its underlying geometry was explained, as well as the RANSAC algorithm which allows for its robust computation in the presence of outliers. My preemptive RANSAC algorithm speeds up the process significantly

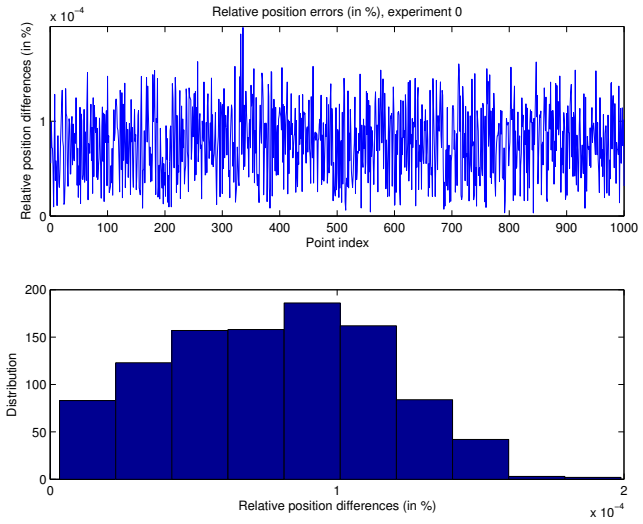


Figure 3.17: Relative Error between the reconstructed 3D points and the original 3D points for the experiment without radial distortion and without noise

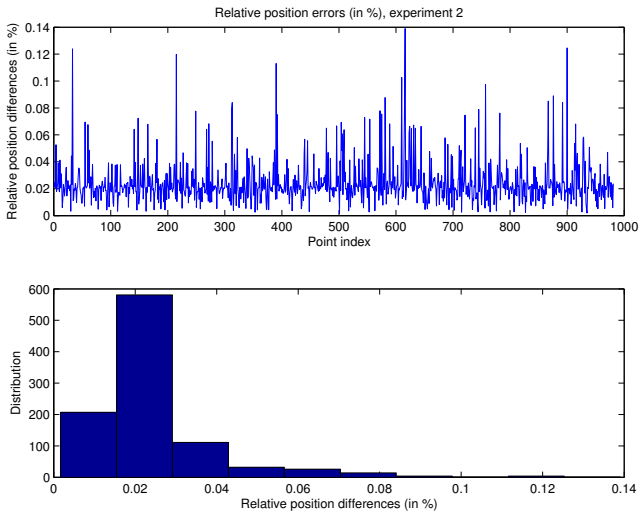


Figure 3.18: Relative Error between the reconstructed 3D points and the original 3D points for the experiment with radial distortion and without noise

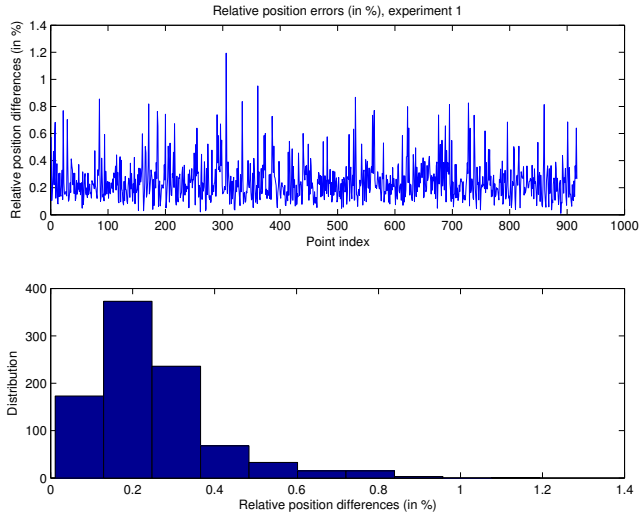


Figure 3.19: Relative Error between the reconstructed 3D points and the original 3D points for the experiment without radial distortion and with noise

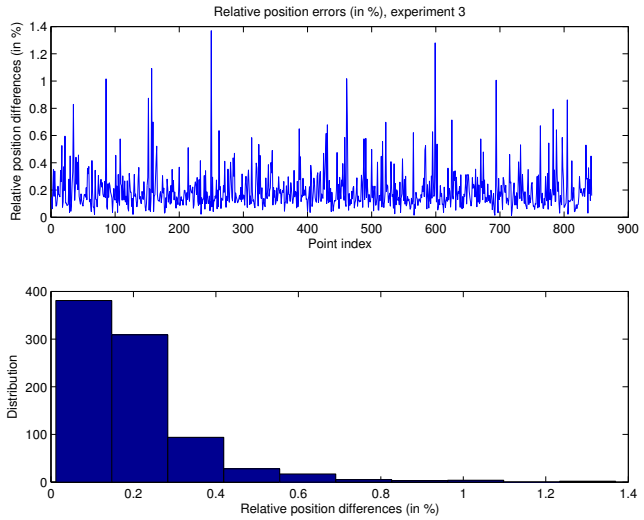


Figure 3.20: Relative Error between the reconstructed 3D points and the original 3D points for the experiment with radial distortion and with noise

because the time needed for checking solutions decreases and simultaneously the updated estimate of the inlying fraction decreases the necessary number of tries. In practice one often has to match two images with a global shift. My estimate of the global transformation makes matching with simple Harris corners still possible. When the fundamental matrix has been computed, my binning approach increases the speed of the F-guided matching algorithm. If a scene consists of a 3D foreground and a (2D) background at infinity, my approach using the H-RANSAC algorithm can place the 3D model inside a virtual cube, textured with this background.

Matching pairs of images is only the first step towards a 3D reconstruction. We need to recover the calibration parameters of the cameras as well. We introduced a multiview reconstruction system in a projective framework which I implemented keeping in mind robustness (e.g. using conditioning in all steps requiring an SVD) and efficiency (e.g. employing the preemptive scheme in P-RANSAC). The self-calibration algorithm that upgrades the projective reconstruction to metric the concept of the dual absolute quadric and allows for multiple (sub-)sequences with the same intrinsics to be calibrated together. Finally, we discussed a multi-view camera setup and its calibration, which can be employed for 3D reconstruction using visual hulls. My calibration algorithm makes use of all concepts described in this chapter and can calibrate the camera setup in a very short time even if a large variation in intrinsic parameters is present.

Chapter 4

Model Selection

4.1 Introduction

At this point we are capable of reconstructing 3D points and cameras from images only. The approach we follow starts with establishing pairwise relations between images by means of the fundamental matrix, then moves on to the computation of a projective reconstruction, and finally upgrades the result to metric. Unfortunately, it happens every so often that so-called critical motions of the camera or critical surfaces in the scene are encountered during recording. Often occurring examples are linear motions or planar surfaces, respectively. These situations will cause our algorithms to fail and we need to be able to deal with them. In [42] the general problem of critical surfaces is described. Different classes of critical motion sequences that cause problems for self-calibration are described in [87, 86, 88, 45]. First we will describe the problems we encounter when dealing with planar scenes. Then we will introduce the concept of the essential matrix and we will explain how to compute it.

4.2 Problems with Planar Scenes

The most common critical surface we encounter in SaM is a planar scene. It might come as a surprise that such a simple scene (a plane) can cause problems for our algorithms, but mathematically it can easily be understood. In paragraph A.4.1 the reader can find the linear transformations a planar object can undergo. Table A.1 shows an overview of them. A general projective transformation can be explained by a so-called planar homography. This means that the transformation from one image to another can be modeled by a 3×3 matrix for which the mathematical expression is given by equation A.8. Since this matrix is only determined up to scale, an homography has 8 degrees of freedom (DOF).

Let us now inspect the case of the computation of a fundamental matrix $\mathbf{F}$. If only a plane is visible in both images, all point correspondences can be

modeled via an homography **H**. It was explained in paragraph 3.4.1 that, since every point correspondence yields 2 equations (one in x and one in y), only 4 points are needed to compute **H**. All other correspondences can be predicted from these 4. This would suggest that only 4 independent point matches can be used to compute **F** but it was seen in paragraph 3.1.3 that we need at least 7 correspondences. This means that there are 3 DOF in **F** we cannot determine without information of points outside the plane. If no such points are available, these remaining DOF are filled by noise on the extracted features. Fig. 4.1 shows an example of the Spijker in Mechelen we already encountered in paragraph 3.7. Two images of the wall are shown in which clearly only a planar part is visible. The top and bottom image-pairs depict the same images but with a different fundamental matrix. Both matrices were computed with the same RANSAC algorithm. The only difference is a small perturbation on one of the parameters (i.e. the outlier distance which was changed from 2 to 2.2 pixels). The resulting F-matrices are quite different which is apparent from the epipolar lines. If one compares the epipolar lines in any of the image pairs however, points on corresponding lines are always in correspondence. This means that both solutions are equally valid and no claim can be made as to the quality of the result without information outside the plane. One can intuitively understand this as follows. Take an arbitrary pencil of lines in one image. If an homography describes the relation between the two images, then every line can be transformed to the other image and all points on the line in the first image will lie on the transformed line in the other.

Also projective pose estimation is problematic for precisely the same reasons. In paragraph 3.6.2 we saw that, since a projection matrix **P** has 11 degrees of freedom (3×4 minus one for scale), and every 3D-2D correspondence yields two equations, we need at least 6 points to compute **P** (actually we only need 5 plus one equation in either x or y of a sixth). If all points are on a plane, however, the correspondences can be predicted if the homography is known, thus if 4 correspondences are given. So again, there are 11-8=3 DOF left in **P** which we cannot determine if we only see a plane.

4.3 Detecting Planar Scenes

A first step in solving the problem with planar scenes is the detection of this situation. Since images of such a scene can be transformed into each other via a homography, one could compute such an **H** matrix and inspect the residual error. If this error is below a threshold, one could consider the scene to be planar. However, it is unclear which value one should choose for this threshold. A better and more stable approach is to take into account not only the residual error but also the degrees of freedom. This can be done by computing and comparing values of the Geometric Robust Information Criterion for different models. This will be explained in the following paragraphs.



Figure 4.1: An image pair depicting a planar scene. The top and bottom pair show epipolar lines of the resulting F-matrices computed with the same algorithm but with slightly different parameters. The influence of noise on the result is apparent. Although the results are clearly different, both are equally valid which can be verified by comparing points on corresponding epipolar lines.

4.3.1 GRIC

The problem of determining whether a scene is planar or not, based on correspondences between images is a specific example of a more general problem which can be stated as follows. If one has a data-set and different models that can explain the data, then which model should one choose? Already in the middle ages a Franciscan friar called William of Ockham [5] shed some light on this. His theory, known as *Occam's Razor* states

> One should not increase, beyond what is necessary, the number of entities required to explain anything.

In essence this theory means that if two or more theories explain the observations equally well, one should always choose the simplest one.

For the case of multiple view geometry, Torr proposed a mathematical description of this principle that is completely general [92], based on Akaike's information criterion [1] and Rissanen's minimum description length [74]. It calculates a score function for each model taking into account the number of inliers, outliers, the residuals, the standard deviation of the error, the dimensionality of the data, the number of the parameters and dimensionality of the model. The general formula is given by

$$GRIC = \sum \rho(e_i^2) + (nd\ln(r) + k\ln(rn)) \tag{4.1}$$

where

$$\rho(e_i^2) = min\left(\frac{e_i^2}{\sigma^2}, 2(r-d)\right) \tag{4.2}$$

and

- n is the number of data (inliers + outliers)
- e_i is the residual for every correspondence i. For an homography $\mathbf{H}$ for instance this is the distance between point $\mathtt{m_2}$ and the transformed corresponding point $\mathbf{H}\mathtt{m_1}$.

  $$e_i = D(\mathbf{H}\mathtt{m_1}, \mathtt{m_2}) + D(\mathbf{H}^{-1}\mathtt{m_2}, \mathtt{m_1})$$

  For a fundamental matrix $\mathbf{F}$ e_i is given by the sum of the distances to the epipolar lines

  $$e_i = D(\mathtt{l_2} \simeq \mathbf{F}\mathtt{m_1}, \mathtt{m_2}) + D(\mathtt{m_1}, \mathtt{l_1} \simeq \mathbf{F}^T\mathtt{m_2})$$

- σ is the standard deviation on the measurement error.
- r is the dimensionality of the data or the amount of observations. In the case of 2 views, r is 4 (two times two coordinates).
- k is the number of model parameters. For a homography k is 8, for a fundamental matrix it is 7.

- d is the dimensionality of the structure. For a homography d is 2 since the points are in a (two-dimensional) plane, for a fundamental matrix it is 3 since the points can now be in the entire 3D world.

Inspecting Eq. 4.1 we clearly identify two parts. The first part is the goodness of the fit. How well does a model describe the data? This is given by the residual error of the corresponding matches. Of course, this residual alone does not suffice since a complicated model that contains a simpler model always explains the data better since it has more degrees of freedom. In the case of a planar surface for instance, the 3 remaining degrees of freedom of a computed fundamental matrix are estimated from noise in the data, allowing for a lower residual score.

That is why the GRIC model has a second part ($nd\ln(r) + k\ln(rn)$) which takes into account the so called *parsimony* of the model. This means that the more complex a model is, the higher this penalty term will become. If we want to choose a more complex model over a simpler one, it should explain the data a lot better, to the extent that its residual error should be so much lower than the residual error of the simpler model that this reduction nullifies the extra cost associated with the higher complexity.

Let us apply this algorithm to the problem of a planar surface visible in a pair of images. We execute the following algorithm.

- Compute the fundamental matrix between the two images, as well as the inliers and outliers using F-RANSAC.
- Compute a non-linear optimization of $\mathbf{F}$ using the inliers only, e.g. with a MLE (Maximum Likelihood Estimator) [2, 37].
- Compute a planar homography matrix $\mathbf{H}$ using all the inliers of $\mathbf{F}$. The latter is absolutely necessary because if only inliers for $\mathbf{H}$ are used to compute it, the H-GRIC value will always be lower than the F-GRIC because of its lower complexity.
- Compute H-GRIC and F-GRIC with Eq. 4.1.
- Compare both GRIC values. If H-GRIC is lower than F-GRIC, then the preferred model to explain the data should be $\mathbf{H}$. This can mean that either the observed scene is planar or that the camera underwent a pure rotation around its own camera center. If the F-GRIC value is lower, then enough 3D information was visible in the scene to compute the F-matrix correctly.

4.4 More GRICs

As explained before, not only the computation of epipolar geometry but also projective pose estimation suffers from dominant planar surfaces. One might

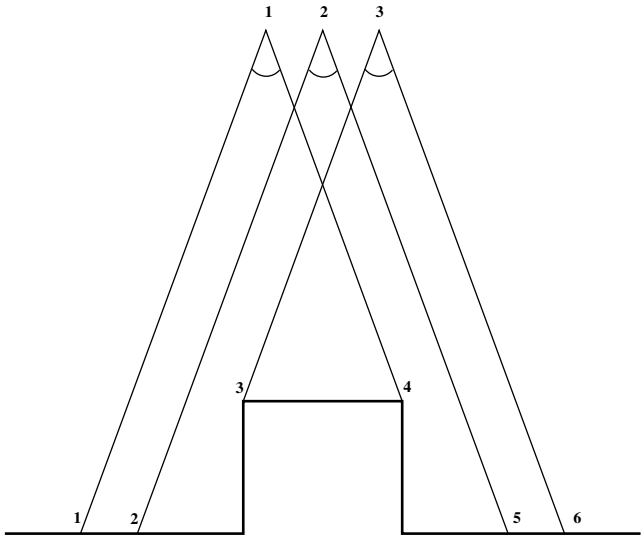


Figure 4.2: The F-GRIC of camera pairs 1-2 and 2-3 is lower than the H-GRIC due to enough 3D information visible in the cameras. However only the planar part 3,4 is seen in common in the three cameras which makes unique projective reconstruction of the cameras impossible.

hope that the comparison of F-GRICs and H-GRICs tells us when enough 3D information is visible in the scene and when this is not the case. However, observe the situation depicted in Fig. 4.2. Three cameras are looking at a scene with plenty of 3D information. The surface between points 2 and 4 is visible in cameras 1 and 2. The surface between points 3 and 5 is visible in cameras 2 and 3. This means that both F-GRICS will yield a lower penalty than their corresponding H-GRIC. However, when looking at the camera triplet, only the surface part between points 3 and 4 is seen in all views. This part is planar and hence the projective reconstruction of the cameras cannot be uniquely determined.

If we want to find out whether there is enough 3D information in the scene for projective reconstruction, we have to take into account the information common in 3 views. This can be done by comparing the HH-GRIC and the PPP-GRIC of such a triplet. This is done as follows.

- Compute a projective reconstruction of the triplet, e.g. by computing epipolar geometry between two images; instantiating the projective frame from these and then performing a projective pose-estimation of the third camera. This also yields a set of triplet-correspondences: features that are matched throughout all three images.

- Compute the best-fitting planar homography between image 1 and 2 and between image 2 and 3, using the same triplet correspondences.

	HH-GRIC	**PPP-GRIC**
r	6	6
k	$2 \times 8 = 16$	$3 \times 11 - 15 = 18$
d	2	3

Table 4.1: Parameters in the GRIC formula 4.1 for HH-GRIC and PPP-GRIC

	F-GRIC	**H-GRIC**	**PPP-GRIC**	**HH-GRIC**
pair 1-2	2332.8	3711.83		
pair 2-3	1424.03	2247.66		
triplet			1435.908	843.01

Table 4.2: The F-GRIC and H-GRIC for the two image pairs show us that fundamental matrix computation is possible. The planar common part however leads to a lower HH-GRIC than PPP-GRIC value which tells us that projective reconstruction of this triplet is not possible.

- Compute the HH-GRIC and PPP-GRIC from these results. Table 4.1 gives an overview of the value of the different parameters in the GRIC formula 4.1. The number of model parameters for PPP-GRIC is $3 \times 11 - 15$ which comes from 3 projection matrices with 11 parameters each but up to a general projective transformation with 15 parameters.

- If the resulting HH-GRIC is lower than the PPP-GRIC then only a planar surface is seen in common in the three images and no projective reconstruction can be computed.

A concrete example of the situation of Fig. 4.2 is shown in Fig. 4.3. A triplet of images shot at the Department of Electrical Engineering of the K.U.Leuven in Heverlee depicts a wall with an extrusion. When one observes the images, it is clear that this planar extrusion is the only part common to the three images. The two consecutive image pairs 1-2 and 2-3 have a larger part in common which yields enough 3D scene to compute the epipolar geometry robustly as can be seen in Fig. 4.4 for the first pair. The resulting GRICs are gathered in table 4.2. Since the value in the first column, denoting the F-GRIC is lower than the H-GRIC in the second column for both image pairs, the epipolar geometry can be estimated correctly. When we compute the projective projection matrices, however, we see that the inliers are only located on the planar extrusion as can be seen in Fig. 4.5. Due to the planar common part, the PPP-GRIC is higher than the HH-GRIC which means that robust estimation of the projective projection matrices of these images is not possible.

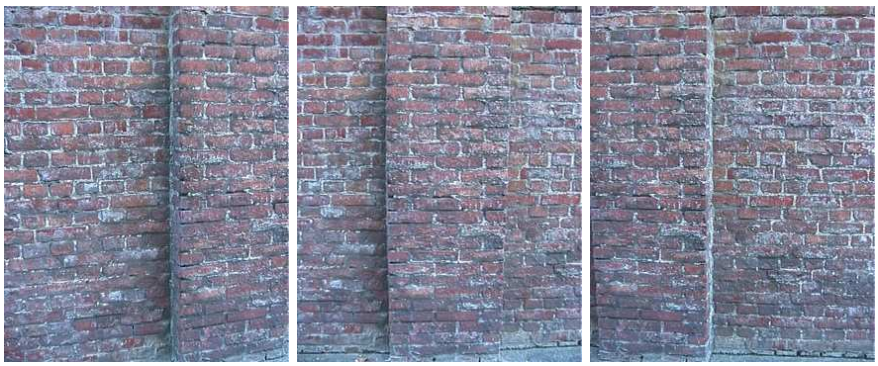


Figure 4.3: Triplet of images with a concrete example of the situation of Fig. 4.2. The common part between the three images is the planar extrusion.

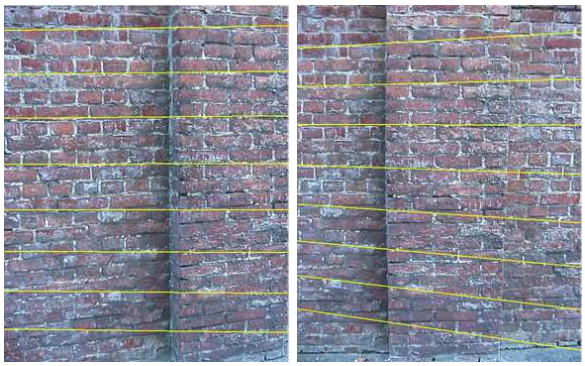


Figure 4.4: Epipolar geometry for the first image pair. Enough 3D information is seen in common to compute the fundamental matrix robustly.

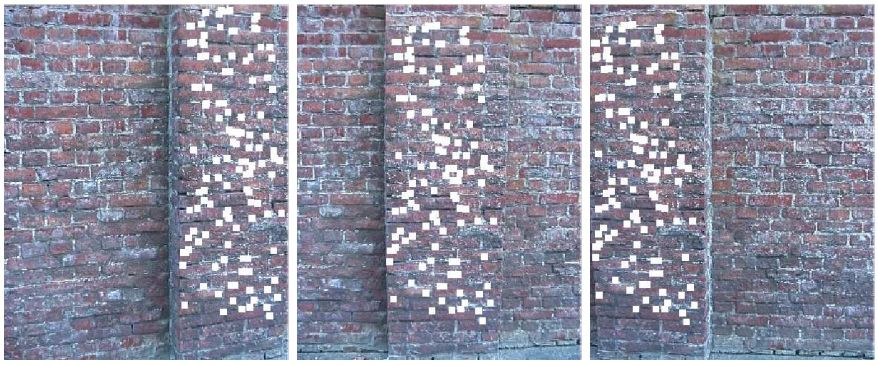


Figure 4.5: Inliers for projective projection matrix estimation. These inliers are clearly only located on the planar extrusion.

4.5 Long Video Sequences

The previous discussion about model selection and the usage of GRICs can play its role in the computation of the structure and motion of long video sequences as will be explained below. Video sequences of 1000 frames or more require specific algorithms and techniques if one wants to retrieve the camera calibration and 3D scene reconstruction from them. Two important differences can be seen which distinguish video sequences from sequences of images taken with a still camera. First, video frames are typically much closer together and second, videos of a thousand frames or more yield many more images to process than a typical still image sequence. These two observations have their impact on the algorithms.

4.5.1 Video Frames

Video recordings yield image sequences with frame rates of 25 (PAL) or 30 (NTSC) frames per second. If the video camera does not move at very high speed (which would cause blurring effects anyway, more on which in paragraph 4.5.4) this means that consecutive video frames are typically very close together and thus very much alike. This gives us a huge advantage for the computation of matching points between images because such points are typically located close to the coordinates in the previous image and very little distortion has to be dealt with. This makes it possible to *track* features rather than match them. In section 3.2.1 we described how to find good features to track with the formula of KLT-features [77].

Unfortunately, the fact that consecutive images are so similar also has its drawbacks. Inspect Fig. 4.6. This figure shows two consecutive frames of a video sequence of the roof of the Antonine Nymphaeum in Sagalassos, Turkey. If we attempt to compute the epipolar geometry between these images we notice that, just like in the case with planar scenes, this process is very sensitive to noise. Two possible solutions of the epipolar geometry are shown in Fig. 4.6 and both are equally valid.

The images shown in Fig. 4.6 depict a scene with plenty of 3D information. Yet we cannot compute the epipolar geometry stably because of the closeness of the two camera centers. The translation between the cameras is very small. This means that the camera setup resembles that of a purely rotating camera and hence the relation between corresponding pixels can also be explained by an homography as shown in Fig. 4.7.

The computation of epipolar geometry breaks down if images are too close together. The same is true for projective camera estimation where newly computed 3D points and cameras suffer from large uncertainties due to the small baseline between cameras (see section 2.3). This means we have to find a technique to increase the baseline between images that are used for structure-and-motion recovery. A possible solution is not to deal with consecutive images but to select every n-th frame of the video sequence. This is not a good idea because it is impossible to choose a value of for n that will work for all video sequences.

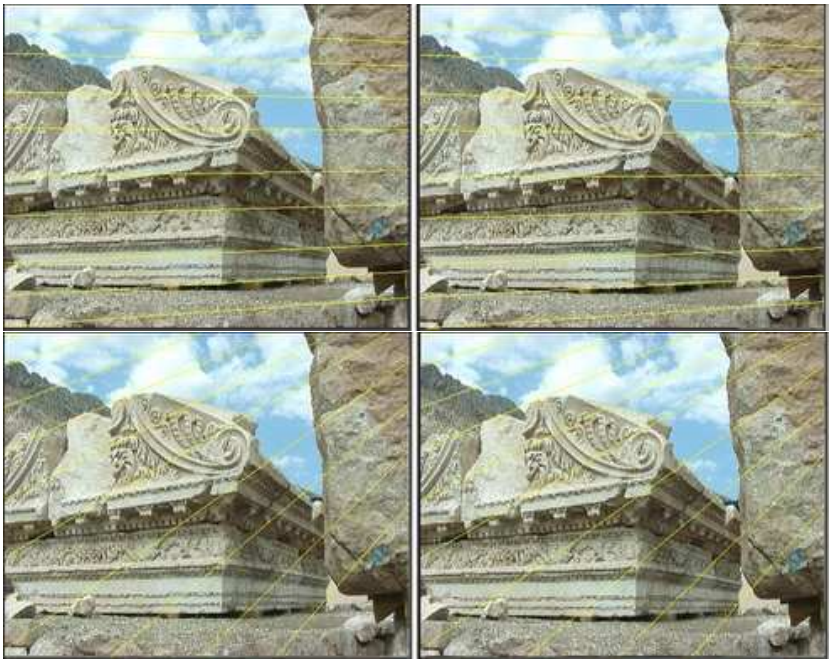


Figure 4.6: Two consecutive frames of a video sequence are so close together that the fundamental matrix cannot be uniquely determined.

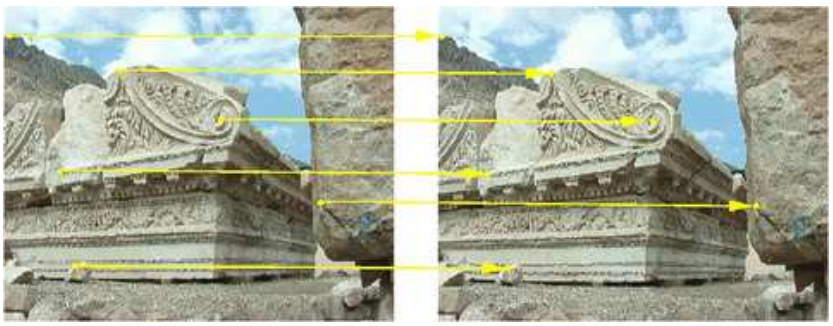


Figure 4.7: Two consecutive frames of a video sequences are so close together that the motion resembles that of a pure rotation. This means that corresponding points can be related by an homography.

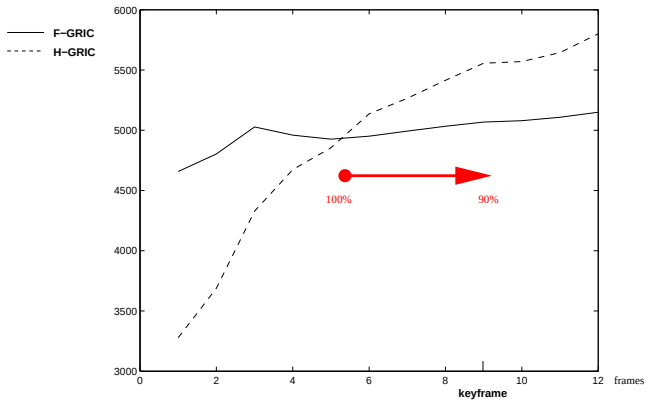


Figure 4.8: While the F-GRIC (full line) is higher than the H-GRIC (dashed line) we continue traversing in the sequence. At the cross-over, we store the amount of inliers for **F**. We continue traversing in the sequence while the amount of inliers is higher than 90% of the stored value. When it drops below 90%, we select that image as the new keyframe.

It would depend on the velocity of the camera, the frame rate and the distance to the scene. These properties could even change during the recording of the video. Imagine a camera filming a scene, standing still every once in a while and then continuing. The value of n would have to be adapted constantly.

We can however make use of the GRIC-concept, explained above. In order to do so we introduce the concept of *keyframes*. We select the first frame in the sequence as the first keyframe and track features to the second frame. We estimate the fundamental matrix and an homography between these frames and compute the corresponding F-GRIC and H-GRIC values. If the H-GRIC is lower than F-GRIC, the homography describes the data better than F and we track the features to the next frame in the sequence where we perform the same process now between the first and the third frame. As long as the H-GRIC is lower, we continue to advance in the sequence. Once the F-GRIC yields the lowest penalty, the epipolar geometry can be reliably estimated and we could select the frame where this happens as the new keyframe. However, in order to reduce the processing time on the structure and motion estimation, we progress a little further in the sequence. For every frame we track the features and estimate epipolar geometry w.r.t. the previous keyframe. While more than 90% of the inliers at the crossing-point are still matched, we continue. When we drop under this 90%, we select this frame as the new keyframe. Figure 4.8 explains the process.

4.5.2 Long Sequences and Subsequences

The technique of selecting keyframes in a video sequence combines the best of two worlds. It allows tracking features instead of matching them by exploiting closeness of consecutive video frames. Due to the degeneracy of close frames for structure and motion estimation, only a subset of frames is selected. Typically every 5th frame is selected for computation which also makes this problem 5 times less computation- and memory-intensive. Even so, due to accumulation of measurement errors, long video sequences have problems with drift [14]. The drift on the projective reconstruction can become quite substantial. This causes problems for the self-calibration techniques explained in paragraph 3.9. If the projective projection matrices at the input of the algorithm are not in the same projective coordinate system, the self-calibration techniques will fail because the pose of the DAQ cannot be found consistently in all images. To alleviate this problem we split the long video sequence into subsequences.

Subsequences

In order to alleviate drift on the projective reconstruction, we split the original long video sequence into shorter subsequences. Typically these subsequences have a length of about 100 frames. With a mean value of one keyframe per 5 frames this yields 20 images per subsequence. The computation of keyframes, fundamental matrices and projective projection matrices is done independently for every subsequence. The tracking of features however is done before on the original long sequence to ensure that the same features are used in all subsequences.

Subsequences with shorter length suffer less from drift and typically self-calibration is possible for all of them. However, since all subsequences are part of a global sequence, recorded with a single camera, the intrinsic parameters of the camera do not change if no zooming is allowed. This means the coupled self-calibration approach of section 3.9.3 can be used to retrieve these intrinsics. The result of this process is the pose of the DAQ and hence a metric reconstruction of every subsequence separately. This means that every subsequence is reconstructed in its own metric frame. If we want to put the subsequences together into the same coordinate system, we have to compute the metric transformation (or similarity transformation, see A.4.2) between all these subsequences, a process we call *gluing* them together.

Gluing Subsequences

In order to glue the subsequences together, we make sure that a certain overlap exists between consecutive subsequences. We typically take an overlap of 10 frames. Fig 4.9 shows this schematically. As explained before, the first frame of a sequence is always picked as a keyframe. Additionally, we always include the last frame of a subsequence as a keyframe as well.

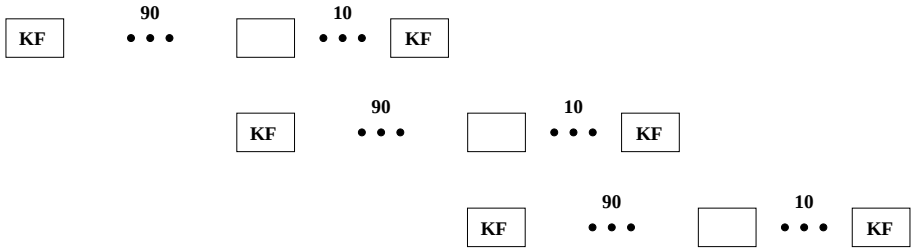


Figure 4.9: The original long video sequence is split in subsequences of 100 frames each. The overlap between subsequences is 10 frames. The first and last frame of every subsequence is always picked as keyframe.

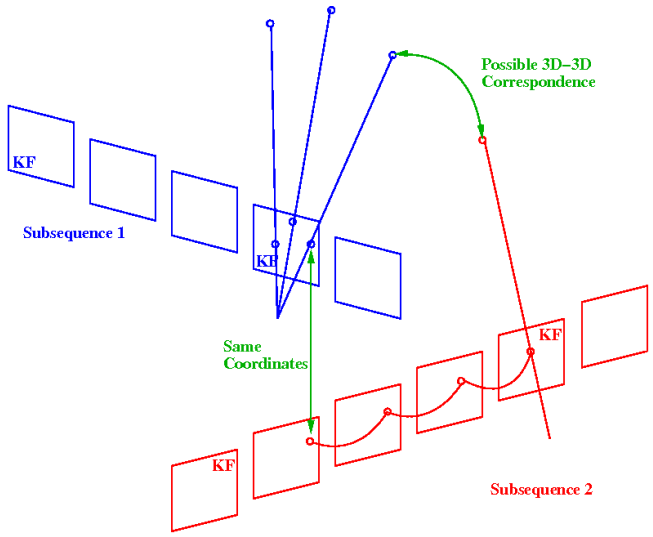


Figure 4.10: The two subsequences have been reconstructed in 3D. A 3D point in the first subsequence has been reconstructed from 2D features in keyframes. We select a frame that is also present in the second subsequence. We take the feature with the same 2D coordinates and follow it until we end up in a keyframe of the second subsequence. If the tracked feature is reconstructed, this 3D point is a possible correspondence of the original 3D point.

The gluing algorithm we employ is illustrated in Fig. 4.10 and works as follows. We consider two subsequences with an overlap of 10 images. For every subsequence, the keyframes have been reconstructed into its own metric coordinate system. This means metric cameras, 3D points and their corresponding 2D points in the images. Because of the overlap of 10 images and the fact that the feature tracking was done on the entire video sequence, these 2D feature points are the same in the overlapping images. Let us consider a 3D point in the first subsequence which projects onto a 2D point in a keyframe which is part of the overlapping 10 images. The same image is also part of the second subsequence but is not necessarily a keyframe. The 2D feature is found on the same position and might have been tracked further in this second subsequence. We follow the track until we reach the next keyframe in this subsequence. If the tracked 2D feature is part of the 3D reconstruction of this subsequence, then the original 3D point and the newly found 3D point form a possible 3D-3D correspondence between the two reconstructed subsequences.

The approach explained above gives us a set of possible 3D-3D correspondences which we will use to compute the metric transformation between two consecutive subsequences. Not all correspondences are inliers however, which means we have to resort to the robust RANSAC technique again to retrieve this transformation. The metric transformation consists of 7 parameters, 3 for rotation, 3 for translation and 1 for scale. Every 3D-3D correspondence yields 3 equations (the X, Y and Z coordinates should be identical after transformation), which means the minimum sample size is 3.

Every sample of 3 3D-3D correspondences allows us to compute a metric transformation linearly. Since a linear method cannot impose the non-linear constraints that the elements of the upper left 3×3 matrix of the transformation form a rotation matrix, the computed values for R_{ij} are not guaranteed to form a true rotation matrix which should be orthonormal with determinant 1. A typical solution to obtain a real rotation matrix is by using QR decomposition [32]. The drawback of this technique is that the extracted orthogonal matrix has no real physical significance. We therefore opt for *Polar decomposition* [79] which gives us the closest possible orthogonal matrix, where closeness is measured with the Frobenius norm.

The RANSAC algorithm checks for support by determining inliers and outliers for the current solution. A straightforward criterion to do so would be to transform all points of one subsequence to the other using the computed transformation and compute the distance between the transformed point and its corresponding point. If this distance is lower than a threshold, the point is an inlier. However, it is unclear which value one should take for the 3D distance threshold. That is why we go back to the images. We still transform the 3D points of one subsequence with the computed solution and then project these transformed points into the image using the camera matrices of the other subsequence. This projection is then compared to the 2D feature points that were used to reconstruct the original 3D point in the first place. If the distance between the projection and the original point is lower than a threshold (typically

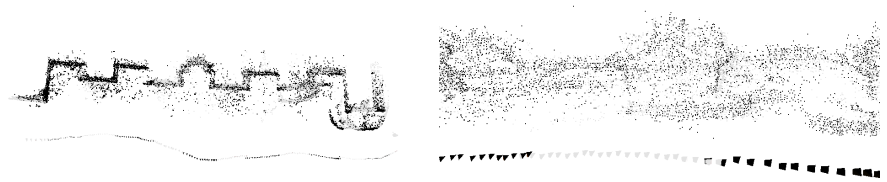


Figure 4.11: Reconstruction of keyframes in subsequences for the *Nymphaeum* sequence. The left view shows an entire overview, the right is a detail.

one or two pixels), then the point is an inlier.

When all subsequences are put in the same coordinate system, a full non-linear optimization on the results can be executed. This bundle-adjustment updates the poses of the cameras, the intrinsics and the position of all 3D points, minimizing the global reprojection error. Care should be taken that common keyframes between subsequences only get one camera matrix and that all inliers of the 3D-3D correspondences relate to the same 3D point. All these constraints lead to index-hell but once implemented properly, the keyframes of the entire video sequence are reconstructed in 3D. Fig. 4.11 shows the result for the *Nymphaeum* sequence. Subsequences (cameras and points) are alternatingly shown in light-gray and black. The detailed view on the right shows 22 cameras in the gray subsequence. This means that in this specific case approximately every fifth frame was chosen as a keyframe.

4.5.3 Intermediate Frames

Certain applications are not satisfied with the calibration of only a subset of all original frames. The creation of a 3D movie from a video sequence for instance requires a depth map for all original frames. This means the camera pose has to be computed for all frames, not only the keyframes. Retrieving the camera calibration for intermediate frames (frames between keyframes) can be done as follows.

Remember that a feature tracker was executed on the entire original video sequence. This means that we have correspondences between all frames, and hence between keyframes and intermediate frames. From the set of 3D-2D correspondences in the keyframes, used to reconstruct the scene, we obtain 2D points in the keyframes for which we have 3D information. The feature tracks yield correspondences between these points and 2D points in the intermediate frames, which gives rise to 3D-2D correspondences for the intermediate frames.



Figure 4.12: Reconstructed points and cameras shown in alternating light-gray and black for alternating subsequences of the *globe* sequence

This means we have enough information to execute a Euclidean pose estimation process (see paragraph 2.2.2).

In our experience an easier method exists which immediately computes the poses with a non-linear method. Because keyframes are typically not too far apart, we can initialize the pose of intermediate cameras by interpolation between the closest keyframes. The translation is easily found and for the rotation one can use quaternions. Another simple method is the one described in [3] where the transformation of camera i, located between keyframes 1 and 2 is computed as

$$\mathbf{T}_i = \exp(a \log(\mathbf{T}_2) + (1-a) \log(\mathbf{T}_1))$$

where a is the ratio describing how far i is located between 1 and 2 (e.g. for the third frame of seven frames between the two keyframes, a is 3/7). The non-linear pose-estimation can then simply consist of a Euclidean bundle-adjustment process where the 3D points and intrinsic camera parameters remain fixed and only the extrinsics are updated.

A result for the *globe* sequence is shown in Fig. 4.12. The top images of Fig. 4.13 show a top view and a detailed view of the recovered cameras for the *Nymphaeum* sequence. The bottom images depict a frame and the full calibration of the *Zaragossa* sequence.

4.5.4 Blurry Frames

When video sequences are recorded in a freehand style, one can never totally exclude the possibility of blurred frames. This blurriness is typically motion blur which is the result of sudden motions of the camera. Blurred frames cause problems for structure and motion techniques mostly because the amount of tracked features drops dramatically when such a frame is encountered. It is

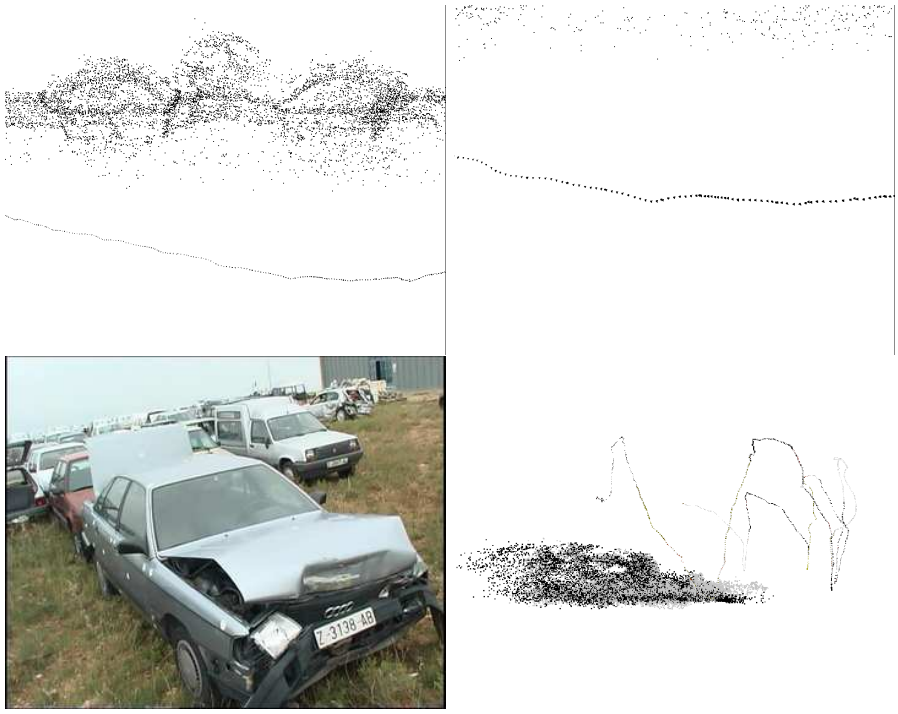


Figure 4.13: Top: Reconstruction of all frames in subsequences for the *Nymphaeum* sequence. Bottom: One frame of the *Zaragossa* sequence (left). Reconstructed points and cameras shown in alternating light-gray and black for alternating subsequences (right)

therefore best to try to detect these frames beforehand and deal with them. The simplest solution is of course to simply discard them.

A possible technique to spot blurred frames automatically is based on comparing the image *sharpness* of nearby frames. The value for the sharpness of an image could e.g. be computed as the mean square of horizontal and vertical intensity gradients, evaluated as finite differences:

$$S = \frac{\sum_{i=0}^{W-1}\sum_{j=0}^{H-1}(I(i+1,j)-I(i,j))^2 + (I(i,j+1)-I(i,j))^2}{WH}$$

The value of such a sharpness measure depends on the image content and thus no absolute threshold can be used to detect blurry frames. However, if one compares the sharpness of frames in each other's neighborhood, then conclusions can be drawn. Due to motion blur, the gradient strength in blurred images is significantly lower than in the other images. Fig. 4.14 shows two frames of the *vase* sequence. The second image suffers from motion blur which is apparent in the zoom-in on a high-frequency part of the image. Fig. 4.15 shows a plot of

the sharpness values for the first 150 frames of the sequence. A significant drop is noticeable around frames 32, 39 and 100. The images in Fig. 4.14 are frames 38 and 39 respectively. The blurred frames are detected and can be removed from the sequence before further processing.

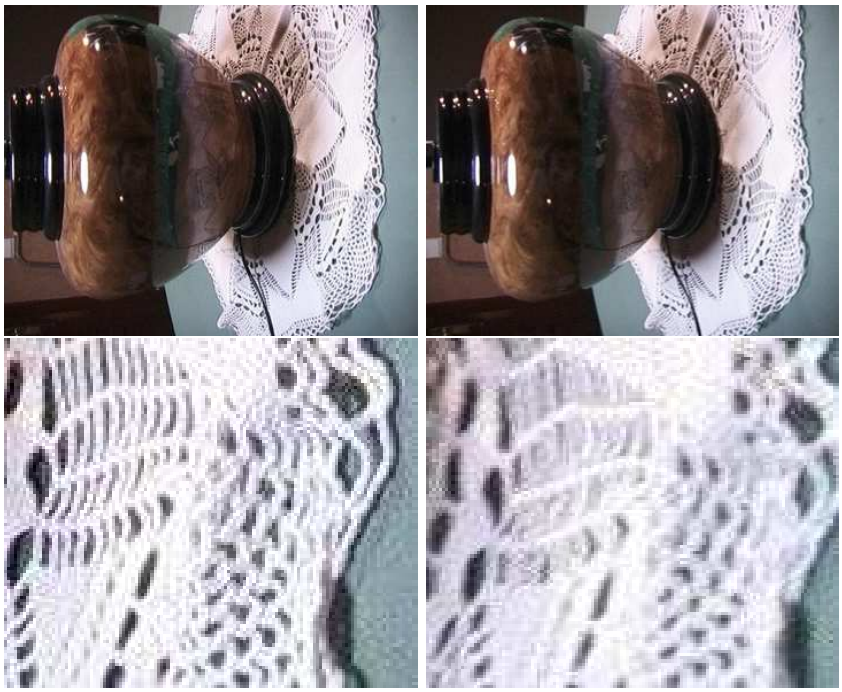


Figure 4.14: Frames 38 and 39 of the *vase* sequence (top). A zoom-in on a high-frequency part of the image shows that frame 39 suffers from motion-blur (bottom).

4.6 Essential Matrices

Until now we have worked with cameras for which we did not know the internal calibration in advance. This meant we had to use the intermediate steps of projective reconstructions and self-calibration to recover the entire calibration of these cameras. If, however, we know the intrinsic parameters of the cameras, either through a calibration or self-calibration algorithm, more possibilities arise. In chapter 3 the concept of the fundamental matrix was derived for an image pair with cameras with unknown internal calibration. This 3×3 matrix imposes a constraint on matching features in that they should lie on each-other's epipolar line. When the internal calibration of the cameras is known, we can *upgrade* this concept to that of an *essential* matrix. Historically these essential matrices

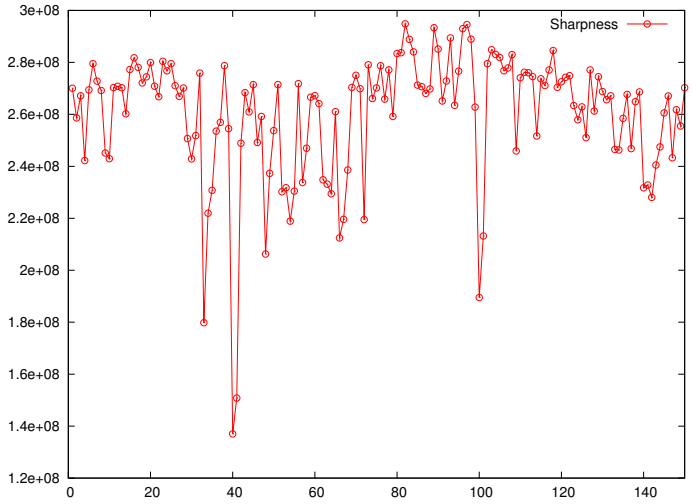


Figure 4.15: The sharpness plot of the first 150 frames of the *vase* sequence shows a noticeable drop around frames 32, 39 and 100.

were derived first [55] but since the fundamental matrix is more general than the essential matrix we have introduced it first.

4.6.1 The Essential Matrix

If the intrinsic parameters of a camera are given, we can compute the so-called *normalized* coordinates of image points. Suppose that an image point $\mathtt{m}$ is the projection of a 3D point $\mathtt{M}$. With the projection equation 2.1 we have

$$\mathtt{m} \simeq \mathbf{K}\left[\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right]\mathtt{M}$$

If the intrinsic parameters of the camera are known, we can define the normalized coordinates $\mathtt{q}$ as

$$\begin{aligned} \mathtt{q} &\simeq \mathbf{K}^{-1}\mathtt{m} \\ &\simeq \left[\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right]\mathtt{M} \end{aligned} \tag{4.3}$$

The essential matrix $\mathbf{E}$, defined as

$$\mathbf{E} \simeq [\mathbf{t}']_\times \mathbf{R}' \tag{4.4}$$

with $\mathbf{R}' = \mathbf{R}^T$ and $\mathbf{t}' = -\mathbf{R}^T\mathbf{t}$, can be shown to have the following property

$$\mathtt{q}_2^T\mathbf{E}\mathtt{q}_1 = 0 \tag{4.5}$$

Comparing Eq. 4.5 with Eq. 3.1 shows that the relationship between $\mathbf{E}$ and $\mathbf{F}$ is

$$\mathbf{E} \simeq \mathbf{K}_2^T\mathbf{F}\mathbf{K}_1 \tag{4.6}$$

Since the essential matrix can be computed from the relative rotation and translation between the two cameras but is only determined up to scale, it has five degrees of freedom. This means there are two extra constraints that have to be enforced w.r.t. the fundamental matrix. These constraints are that two of its singular values have to be equal and that the third is zero which can be proven from the eigenvalue decomposition of a skew-symmetric matrix [32]. The property that the first two singular values are equal boils down to constraints instead of one. For proof we refer the reader to [23] where the equivalence with the Kruppa equations [50] is shown.

4.6.2 Cameras from the Essential Matrix

From Eq. 4.4 we see that the essential matrix is constructed only from the relative extrinsic parameters of the cameras of the two images. It is possible to extract them again if the essential matrix is known. For this one makes use of the matrices $\mathbf{Z}$ and $\mathbf{W}$ defined in Eq. 4.7

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.7}$$

If the singular decomposition of $\mathbf{E}$ is $\mathbf{U}diag(1,1,0)\mathbf{V}^T$, there are two possible factorizations of $\mathbf{R}' = \mathbf{R}^T$ and $\mathbf{t}' = -\mathbf{R}^T\mathbf{t}$ [39]

$$[\mathbf{t}']_\times = \mathbf{U}\mathbf{Z}\mathbf{U}^T \tag{4.8}$$
$$\mathbf{R}'_1 = \mathbf{U}\mathbf{W}\mathbf{V}^T \tag{4.9}$$
$$\mathbf{R}'_2 = \mathbf{U}\mathbf{W}^T\mathbf{V}^T \tag{4.10}$$

Using the factorizations of Eq. 4.8, 4.9 and 4.10 we can now extract the relative rotation and translation from the essential matrix. For the two rotations this immediately follows from the equations. For the translation we follow this reasoning: Because $[\mathbf{t}']_\times\mathbf{t}' = 0$, it follows that $\mathbf{U}\mathbf{Z}\mathbf{U}^T\mathbf{t}' = 0$, thus $\mathbf{t}'$ must be the third column of $\mathbf{U}$, $\mathbf{u}_3$, the singular vector corresponding to the third, zero singular value. However, the sign of $\mathbf{t}'$ cannot be determined which gives rise to two possible solutions for $\mathbf{t}'$: $\mathbf{t}'_1 = \mathbf{u}_3$ and $\mathbf{t}'_2 = -\mathbf{u}_3$. Notice that, since the essential matrix can only be recovered up to scale, the scale of the translation is unknown as well. Only the direction can be recovered.

This means that there are four possible solutions to the relative rotation and translation between the two cameras: $(\mathbf{t}'_1, \mathbf{R}'_1)$, $(\mathbf{t}'_2, \mathbf{R}'_1)$, $(\mathbf{t}'_1, \mathbf{R}'_2)$ and $(\mathbf{t}'_2, \mathbf{R}'_2)$. Be aware that the extracted rotation and translation are actually the inverse values we have to fill in in the projection equation since we defined $\mathbf{R}' = \mathbf{R}^T$ and $\mathbf{t}' = -\mathbf{R}^T\mathbf{t}$. The four solutions are depicted in Fig. 4.16. The camera in the middle is the first camera, put in the origin. The four solutions in the order given above correspond to the pairs consisting of the middle camera with the bottom right, bottom left, top right and top left camera respectively. The geometric interpretation is as follows. The translations $\mathbf{t}'_1, \mathbf{t}'_2$ are each other's

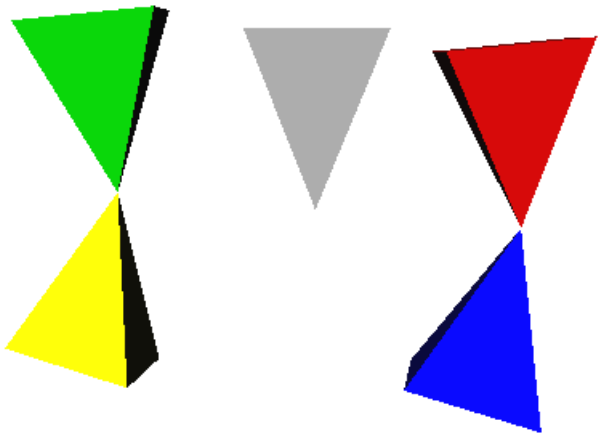


Figure 4.16: From the SVD of the essential matrix $\mathbf{E}$, four solutions for the relative translation and rotations can be computed. In this figure, the middle camera is always the first camera, the inverse rotation and translation of the second camera are given by $\mathbf{R}'$ and $\mathbf{t}'$. The four possible solutions $(\mathbf{t}'_1, \mathbf{R}'_1)$, $(\mathbf{t}'_2, \mathbf{R}'_1)$, $(\mathbf{t}'_1, \mathbf{R}'_2)$ and $(\mathbf{t}'_2, \mathbf{R}'_2)$ correspond to the bottom right, bottom left, top right and top left camera respectively.

opposite. The rotation matrix $\mathbf{R}'_2$ is found as $\mathbf{R}'_1$ rotated 180 degrees around the axis connecting the two camera centers.

4.6.3 Computation of the Essential Matrix

The relative rotation and unsigned direction of translation between the cameras of a camera pair can directly be extracted from the essential matrix of this pair. This has tempted computer vision researchers to make use of it since many years. There are many techniques to compute the essential matrix from point matches and three of them will be explained here.

8 and 7 Point Algorithm

The most straightforward method to compute the essential matrix is the simple, linear 8-point algorithm we introduced in paragraph 3.1.3 to compute the fundamental matrix. This is the oldest algorithm and suffers from the fact that no constraints on the matrix are enforced. The seven point algorithm enforces the rank-2 constraint. It is non-linear and boils down to finding the roots of a third degree polynomial.

We can employ exactly the same algorithm to compute the essential matrix. Either we follow the same procedure as in paragraph 3.1.3 to compute the

fundamental matrix with the 8- or 7 point algorithm. Then we can extract the essential matrix from **F** using Eq. 4.6. An alternative algorithm immediately employs the intrinsic parameters to compute the normalized coordinates with Eq. 4.3. Then we solve for **E** directly with the 8- or 7 point algorithm.

Both methods described here suffer from the same problem. They don't enforce the extra constraints an essential matrix should fulfill. The 7 point algorithm only requires the rank of the matrix to be 2. The two remaining singular values of the matrix are not necessarily equal. The following algorithms enforce this constraint as well.

6 Point Algorithm

In 1990 Faugeras and Maybank showed [24] that the constraint on equal singular values of the essential matrix can be written as the following equation on **E**

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0 \tag{4.11}$$

How can we employ the constraint of Eq. 4.11 to compute **E** from 6 matches? A possible strategy is explained in [68]. As a first step we write down the 6 homogeneous equations 4.5 in the 9 elements of **E**. The SVD of this 6x9 matrix yields the right nullspace of this matrix, consisting of 3 vectors. The real solution for **E** cannot be computed just from this SVD and is of course a linear combination of these three vectors. The scale factors for this linear combination are called λ_1, λ_2 and λ_3. If we insert this solution (with the three unknowns λ_i into Eq. 4.11, this yields 9 homogeneous polynomial equations of degree 3 in λ_1, λ_2 and λ_3. Since **E** can only be computed up to scale, we can only recover ratios of λ_i:

$$x = \frac{\lambda_1}{\lambda_3} \quad y = \frac{\lambda_2}{\lambda_3} \tag{4.12}$$

Since the 9 equations are of degree 3 in λ_i, we can write them down using the substitution in Eq. 4.12, yielding terms in $x^3, x^2y, x^2, xy^2, xy, x, y^3, y^2, y, 1$. Thus, we have 9 equations in 9 unknowns, which can be solved for linearly. The solution for x and y is used to compute λ_i from Eq. 4.12, setting λ_3 arbitrarily to 1.

The advantage of this approach is that it is a linear method to compute the essential matrix, taking into account the constraints on **E** explicitly. Only 6 point pairs are needed instead of 7 or 8 which reduces the amount of necessary tries in the RANSAC algorithm as can be seen from formula 3.14.

5 Point Algorithm

We can now compute the essential matrix from 8, 7 or 6 point matches. However, we know that **E** has only 5 degrees of freedom, so we search for a method that can compute it from 5 point matches only. Unfortunately, no such algorithm exists that is linear. In [68] a non-linear 5 point algorithm is explained. Nister employs the same technique in [65] to solve for the essential matrix.

The algorithm follows the same steps as the 6-point approach. Five equations on the 9 elements of $\mathbf{E}$ now yield a nullspace of 4 vectors, from which the solution is described as a linear combination in four parameters λ_1, λ_2, λ_3,λ_4. A similar substitution is performed as in Eq. 4.12.

$$x = \frac{\lambda_1}{\lambda_4} \quad y = \frac{\lambda_2}{\lambda_4} \quad z = \frac{\lambda_3}{\lambda_4} \tag{4.13}$$

From this a linear system is obtained in variables x^3, x^2y, x^2z, x^2 etc. We can apply a Gauss-Jordan elimination on these 9 equations to yield the following form:

	x^3	x^2y	x^2z	x^2	xy^2	xyz	xy	xz^2	xz	x	y^3	y^2z	y^2	yz^2	yz	y	z^3	z^2	z	1
(a)	1									.	.	.	.	.	.	.	.	.	.	.
(b)		1								.	.	.	.	.	.	.	.	.	.	.
(c)			1							.	.	.	.	.	.	.	.	.	.	.
(d)				1						.	.	.	.	.	.	.	.	.	.	.
(e)					1					.	.	.	.	.	.	.	.	.	.	.
(f)						1				.	.	.	.	.	.	.	.	.	.	.
(g)							1			.	.	.	.	.	.	.	.	.	.	.
(h)								1		.	.	.	.	.	.	.	.	.	.	.
(i)									1	.	.	.	.	.	.	.	.	.	.	.

(4.14)

In the rest of the discussion $[n]$ is used to denote an n-th degree polynomial in z. Let us start by eliminating xz^2 from equations (h) and (i) in Eq. 4.14. which is done by subtracting z times (i) from (h), yielding a new equation

$$x[1] + y^4[1] + y^2[2] + y[3] + [4] = 0 \tag{4.15}$$

In the same way we can eliminate other factors by computing

$$z(g) - (f)$$
$$z(d) - (c)$$

which yields the same form as Eq. 4.15. If we compute $(e) - y(g) - [0](g)$, then we obtain an expression like

$$x[0] + y^4[0] + y^3[1] + y^2[2] + y[3] + [3] = 0 \tag{4.16}$$

The same form is obtained from $(b) - x(g) - [0](g)$. A linear combination of the two equations of form 4.16 can eliminate the term in y^4 which gives us an equation of the form

$$x[0] + y^3[1] + y^2[2] + y[3] + [3] = 0 \tag{4.17}$$

Finally, one more equation is obtained from (4.16)

$$(a) - x(d) - [0](d) - y[0](e) - [1](e) - [2](g) - [0] \tag{4.18}$$

Combining the three equations of the form (4.15) and the two equations (4.17) and (4.18) gives us five equations in the variables $x, y^3, y^2, y, 1$ where every coefficient is a polynomial in z. The stacked equations look like Eq. 4.19

$$\begin{aligned} x[1] + y^3[1] + y^2[2] + y[3] + [4] &= 0 \\ x[1] + y^3[1] + y^2[2] + y[3] + [4] &= 0 \\ x[1] + y^3[1] + y^2[2] + y[3] + [4] &= 0 \\ x[0] + y^3[1] + y^2[2] + y[3] + [3] &= 0 \\ x[3] + y^3[2] + y^2[3] + y[4] + [5] &= 0 \end{aligned} \tag{4.19}$$

This homogeneous system of equations only has a non-trivial solution if the determinant of the 5x5 matrix is zero. This constraint leads to a 13-th degree polynomial in z, denoted as $p(z) = 0$. Thus, if we compute the roots of the polynomial $p(z)$, e.g. by means of a Sturm Chain [30] or by computing the eigenvalues of p's companion matrix [18], we compute possible solutions for z. For every solution we solve for x and y using system 4.19, from which we compute λ_i by means of Eq. 4.13 and setting $\lambda_4 = 1$. This then gives us the linear combination of the vectors of the nullspace and yields a possible solution for $\mathbf{E}$. In [65], the technique described above is slightly adapted to yield two 11th degree polynomials of which the roots give possible solutions for $\mathbf{E}$. For the implementation of this algorithm, we had to resort to the Maple [49] software to find the coefficients of all terms in matrix (4.14), perform the Gauss-Jordan elimination and the computation of the coefficients of the polynomials. Once correctly instructed, Maple produces a c-file with the generic code to construct these polynomials for a given set of 5 correspondences.

4.6.4 Planar Degeneracy

Bearing in mind the discussion we had previously about degenerate cases, one may wonder how the computation of the essential matrix suffers from such scenes. We are particularly interested in planar scenes because they are the most common form of degenerate surfaces. If we have knowledge of the intrinsic parameters of the camera, we can employ the techniques described above to compute the essential matrix from 5, 6, 7 or 8 point correspondences. Because the 7 and 8 point algorithms are basically the same as the F-matrix algorithm, they don't work on planar scenes either. The 6-point algorithm, while interesting due to the reduction of the needed correspondences, will also fail on purely planar scenes. The 5 point algorithm however is in general unaffected by the planar degeneracy and will yield good results.

It has been shown [56, 59, 64] that two possible solutions exist for the essential matrix of two cameras looking at a plane. This is not a problem because for only one of the two solutions do all reconstructed points lie in front of the cameras. This so called cheirality principle [38] allows us to distinguish the correct solution from the false one. This works provided that not all the points are closer to one of the cameras than to the other.

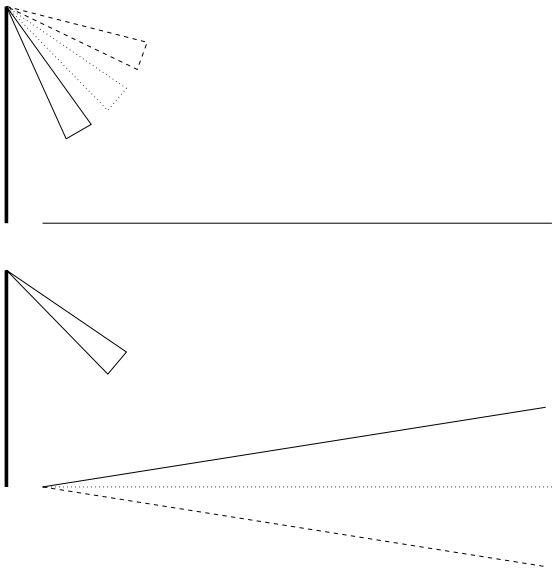


Figure 4.17: Combining different segments. The top view shows the setup of cameras looking at a (possibly planar) scene from different tilt angles. This is equivalent to looking from the same position at a scene consisting of repetitions of the original scene at different positions.

Sometimes, the specifics of the camera setup allow for an optimized algorithm. Take for example the case of a stereo camera setup, mounted on a pan and tilt axis. This system allows to record stereo pairs in all directions from one position. In order to calibrate the relative position and orientation between the two cameras, we can employ images of an unknown environment. If this environment is planar, the 6, 7 and 8 point algorithms don't work. However, we can think of a different strategy. If we take multiple image pairs with different pan and tilt angles the relative pose of the cameras does not change. This means we can regard this setup as if the cameras don't move but the scene does, as illustrated in Fig. 4.17. For every tilt angle we compute a set of possible matches between feature points. If we now combine these different sets into one and feed this to the standard F-RANSAC algorithm, using the simple 7-point algorithm, then we can compute the fundamental matrix, even if the terrain is planar because the combination made an artificial scene, consisting of different planes. The essential matrix is easily derived from Eq. 4.6 and the extrinsics of the cameras are computed from Eq. 4.8, 4.9, 4.10.

4.7 Dense Matching

The result of the SaM procedures consists of camera calibration and sparse 3D point reconstruction. This is only part of the story. In order to obtain convincing 3D models, we must reconstruct much more 3D points, i.e. obtain a *dense* reconstruction of the scene. To do so, we must search for dense matches between the images. In paragraph 4.7.2 we will examine a possible algorithm for the standard technique to perform dense matching between a pair of images: *stereo matching*. Because an image sequence consists of multiple images and thus multiple image pairs, the depth information, computed from the image pairs must be integrated, an approach for which will be explained in section 4.7.3. First, however, we will explain a standard technique, called *rectification* to simplify dense matching by taking into account knowledge on the epipolar geometry.

4.7.1 Rectification

In section 3.1.1 we introduced the concept of epipolar geometry for an image pair. It delivers pairs of corresponding epipolar lines. This concept will reduce our dense matching efforts tremendously because a pixel in the first image only needs to be compared to pixels on the corresponding epipolar line in the other image. In general epipolar lines can lie in all directions. This hampers the implementation of search algorithms that want to take advantage of the knowledge of the epipolar geometry. A pre-processing step, called *rectification* makes our life a lot easier. The idea is that, since pairs of epipolar lines can be found, it is a good idea to rewarp the images such that these corresponding lines now coincide with (horizontal) image scanlines. Then correspondences must only be searched along scanlines with the same y coordinate in the two images and hence only differ in horizontal displacement, called *disparity*.

Planar Rectification

Different algorithms exist to rectify pairs of images. The most popular kind of rectification is *planar* rectification in which an optimal plane is searched for. The two images I_1 and I_2 are projected onto this plane with a homography. Image I_1 is warped to image I_1' using homography $\mathbf{H_1}$ and I_2 is warped to I_2' with $\mathbf{H_2}$. The resulting image pair should have corresponding epipolar lines that are horizontal. This boils down to a fundamental matrix of the following form

$$\mathbf{F}' \simeq \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Let $\mathbf{F}$ be the (known) fundamental matrix of the original pair I_1, I_2. The relation between $\mathbf{F}$ and $\mathbf{F}'$, using the two homographies is given by Eq. 4.20.

$$\mathbf{F} \simeq \mathbf{H_2}^T \mathbf{F}' \mathbf{H_1} \tag{4.20}$$

Eq. 4.20 does not define the two homographies $\mathbf{H_1}$ and $\mathbf{H_2}$ uniquely and existing algorithms differ in the choice of the remaining degrees of freedom. The standard approach is to consider the baseline between the two images of the stereo setup and selecting a plane parallel to this baseline. The remaining degrees of freedom correspond to the orientation of the plane and the distance to the baseline. Different techniques exist to find the optimal plane w.r.t. a certain criterion, for instance from information on the two optical axes: construct the two epipolar planes, containing the baseline and the optical axes. Select the orientation of the plane such that its normal bisects the angle between these two planes. The distance can be defined such that no compression of pixels is performed. A different approach that minimizes the image deformation is given in [57].

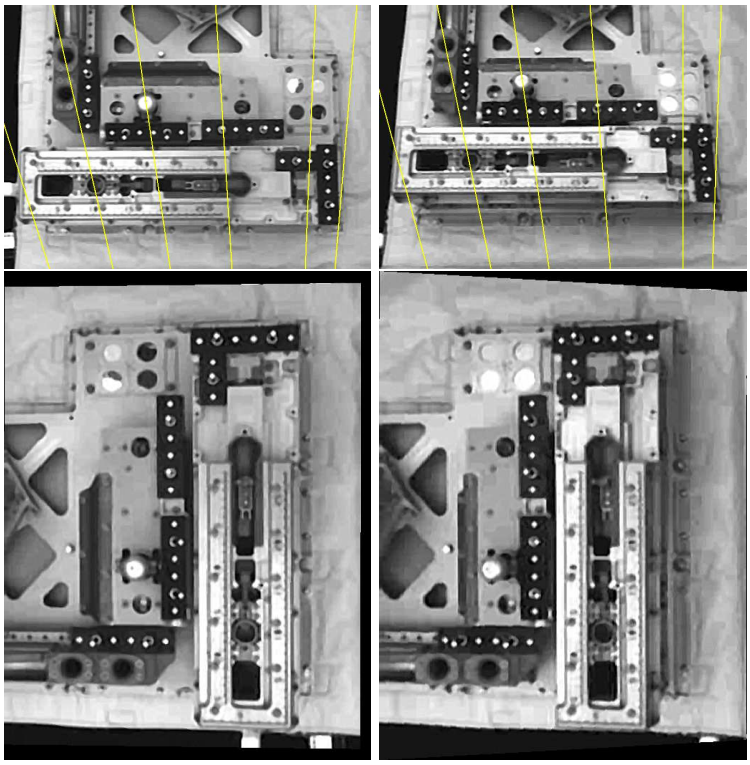


Figure 4.18: Epipolar geometry of the original pair (top). Planar rectified images (bottom)

The top of Fig. 4.18 shows two images of an image sequence of the workcell of the ETS-VII satellite (see chapter 5) with some epipolar lines. At the bottom the resulting planar rectified images are shown. One can check that corresponding epipolar lines are now horizontal and at the same y-coordinate.

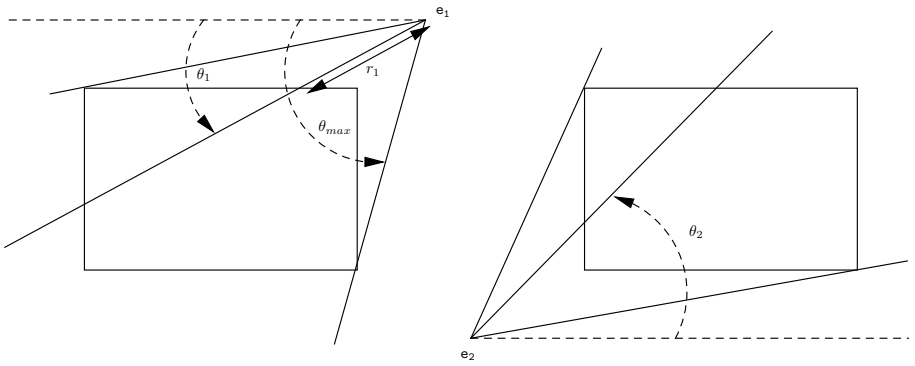


Figure 4.19: Polar Rectification. The epipolar line with angle θ_1 in image 1 corresponds to the epipolar line with angle θ_2 in image 2. The pixels with distance r_1 from the epipole are written on a scanline of the left rectified image and the pixels on the second epipolar line fill the corresponding scanline on the right rectified image.

Polar Rectification

The planar rectification method has several advantages. First, it is relatively easy to compute. The search for the plane is the hardest part. Once it has been located, the rest of the algorithm is straightforward. Second, since the images are only transformed with a planar homography, all projective invariants remain valid. This means e.g. that straight lines in the original images are still straight in the resulting rectified images.

Unfortunately, planar rectification has a major weakness: it cannot handle all configurations of image pairs. A typical example is the setup in which the camera moves forward. In this case the epipoles are located in the image. If a

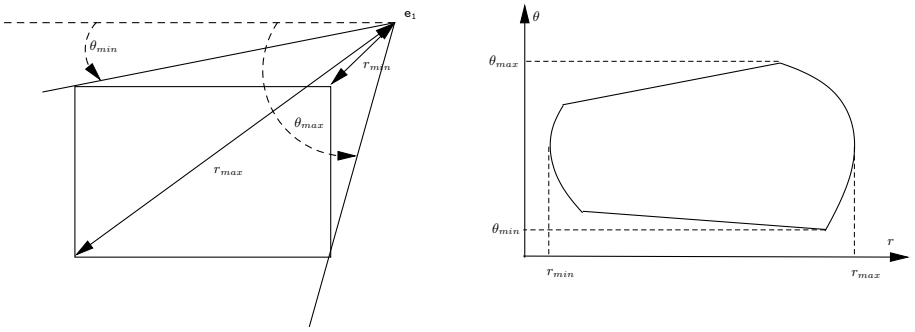


Figure 4.20: Polar Rectification. The size of the rectified image depends on the position of the epipole which fixes θ_{min}, θ_{max}, r_{min} and r_{max}.

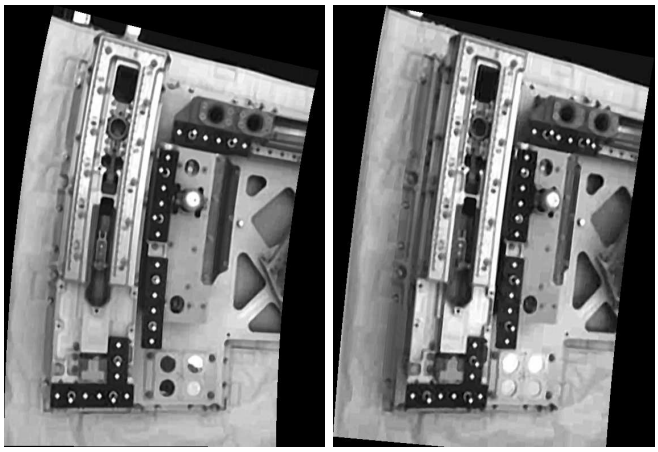


Figure 4.21: Polar rectified images of the *Viable* images. The polar rectification clearly breaks projective invariants: straight lines are no longer straight.

plane has to be chosen which is parallel to the baseline and we don't want to lose any image information, then the resulting rectified images are doomed to become infinitely large. In [71] an algorithm is described which can handle general camera configurations. This method, called *polar rectification* only makes use of knowledge on the oriented fundamental matrix between images (for more information on oriented epipolar geometry, we refer the reader to [53]). The algorithm essentially considers a pencil of epipolar lines through the epipole of one image and the corresponding lines through the other epipole and is illustrated by Fig 4.19. The pixels that are encountered on the epipolar line with angle θ_1 in image 1 are written in a scanline of the first rectified image. Analogously the corresponding scanline in the rectified second image is formed by using pixels on the corresponding epipolar line with angle θ_2. Of course only epipolar lines that intersect both images need to be taken into account. The images are thus traversed in a fashion not unlike *polar* coordinates: with an angle θ describing which epipolar line we take and a distance r between the pixel and the epipole. Fig. 4.20 illustrates this: the rectified version of an image on the left has pixels between θ_{min} and θ_{max} and between r_{min} and r_{max}. In practice, the algorithm is a somewhat more involved than one is lead to believe, depending on the location of the epipoles w.r.t. the images. However, the technique can deal with general configurations of cameras.

An example of a polar rectified image pair is shown in Fig 4.21. For well-behaved setups, the result of this approach does not differ much from the planar rectification. The big advantage however is that all camera configurations (including forward motions) can be dealt with. An example of such a setup is shown in Fig. 4.22 which depicts a forward motion in a narrow street in the Béguinage in Leuven. The epipolar geometry is shown on top with epipolar

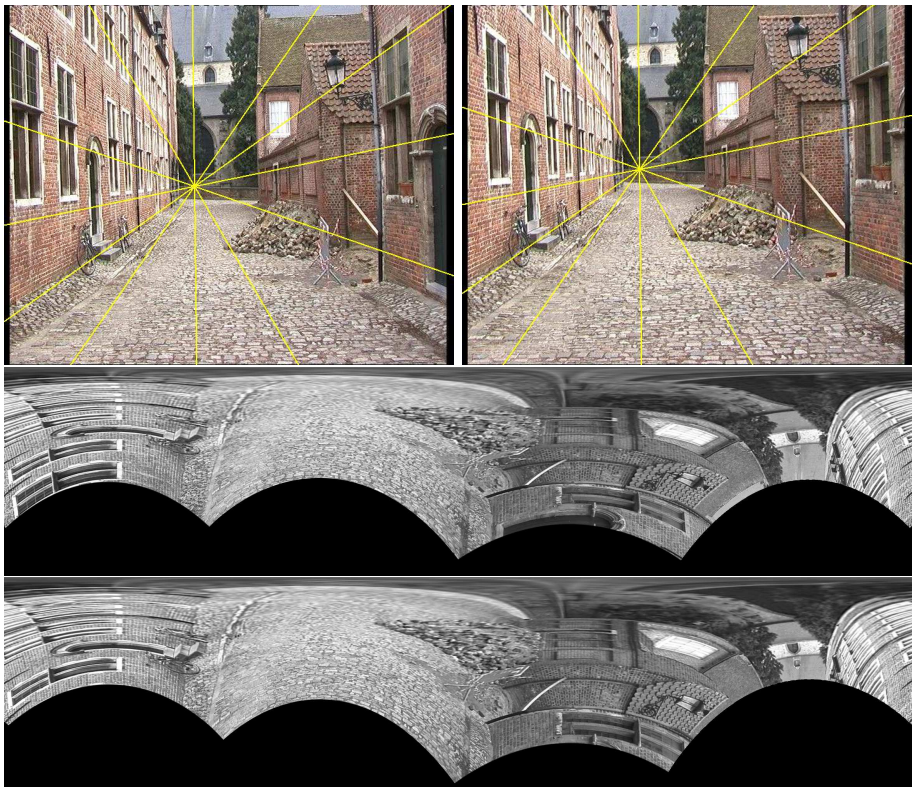


Figure 4.22: Epipolar geometry of a pair of images of the *Béguinage* sequence (top). The epipoles are clearly located in the image, which poses problems for planar rectification algorithms. Polar rectified images yield a viable alternative (bottom)

lines which clearly intersect inside the image, i.e. with an epipole in the image. The resulting polar rectified images are shown underneath and are 90 degrees turned for better visualization, hence corresponding lines are now vertical. Because all epipolar lines coincide in the epipole, and the epipole is inside the image, all angles θs have a pixel which is at $r_{min} = 0$. This can be seen from the rectified images which have a full horizontal line at the top (the θ axis is now horizontal because of the 90 degree rotation).

Compared to the planar rectification method, polar rectification has one drawback. Since it describes the image in polar coordinates, the resulting images are no longer formed by a standard camera and can't be computed from the original images with a projective transformation. This means that projective invariants are not valid anymore. This is already clear in Fig. 4.21 where straight lines (e.g. the edges of the markers) are no longer straight.

4.7.2 Stereo Matching

Dense stereo matching is the technique which tries to find as many pixel matches as possible between the two images of a stereo pair. A straightforward method would be to scrutinize every pixel of the first image independently and simply search for the best matching pixel in the second image. The criterion for matching can then be SSD, NCC or another window-based correlation measure. This approach has the advantage of simplicity and is easy to parallelize. However this approach will yield many-to-one matches and has unpredictable results if different pixels are similar.

Constraints

This is why stereo algorithms typically take into account several constraints to make the matching more robust. Typical constraints are

- *Epipolar constraint*: if the epipolar geometry is known, matching pixels must only be searched on corresponding epipolar lines. This is why rectification (paragraph 4.7.1) is so important because it rewarps the images such that epipolar lines become corresponding scanlines and allows the stereo algorithm to only compare pairs of scanlines.
- *Uniqueness constraint*: a 3D point cannot be projected to two 2D points in one image. The stereo algorithm should therefore enforce that a pixel in the second image can only be matched to one pixel in the first image and vice versa.
- *Ordering constraint*: if the camera centers are on the same side of all planes tangent to the parts of the scene visible to both cameras, which is commonly the case, the ordering of pixels along epipolar lines is preserved. Because of this constraint, dynamic programming searching techniques [8] can be employed to perform the stereo search.
- *Disparity range constraint*: from the set of reconstructed 3D points which result from the Structure and Motion algorithm, a minimum and maximum depth value can be deduced. These values can be translated to a minimum and maximum disparity range outside which the stereo algorithm does not have to search.
- *Smoothness constraint*: probably the most important constraint because it alleviates the problem of relatively untextured areas in the image. This constraint enforces the disparity to evolve smoothly along the epipolar line except at abrupt changes like depth discontinuities or edges.

Matching Matrix

In order to abide to all the constraints listed above, a global search has to be performed which searches for the optimal global solution to the matching problem.

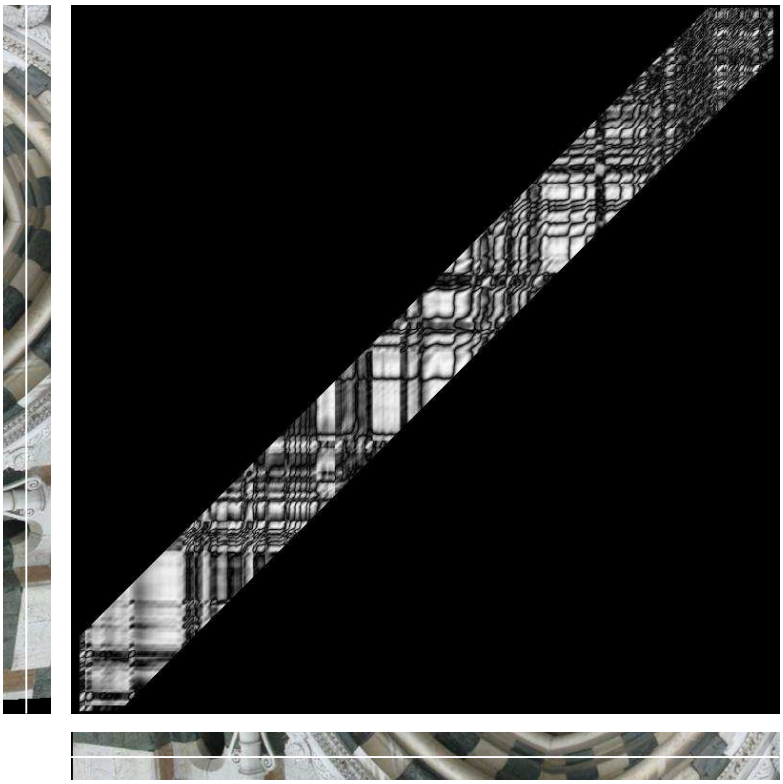


Figure 4.23: Matching matrix of two scanlines of the Prato sequence. For every pixel on the left scanline, all pixels on the right scanline within the disparity range are compared using NCC. This value is put in the correct place in the matrix.

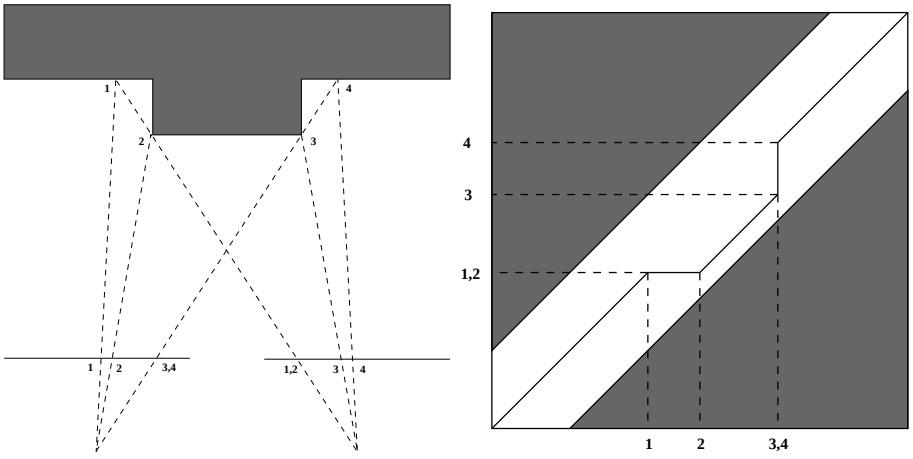


Figure 4.24: The impact of occlusions on the path in the matching matrix. The left figure shows a top view of a scene recorded in two images of a stereo pair. The right figure shows the path in the matching matrix. Point 3 and 4 coincide in the left image, which means that all pixels between 3 and 4 in the second image are not visible in image 1. Hence, a vertical line is found in the path between these points. Similarly points 1 and 2 are the same point in image 2 which means the area between them is occluded in this image and the horizontal piece of the path is explained.

An example of such a global search, implemented as a dynamic programming algorithm is explained in [100]. The most important concept of the matching is of course still the idea that matching pixels should resemble each other. Therefore, all pixels of one scanline are compared to the pixels of the pixels on the corresponding scanline in the other image, using NCC. Taking into account the disparity range constraint, only pixels with a disparity between d_{min} and d_{max} have to be matched. Fig. 4.23 shows a matching matrix of two scanlines of a stereo pair of the Prato sequence. Only the pixels with a disparity between -72 and 47 are compared, values we deduced from the depth range of the reconstructed 3D points used for self-calibration. The goal of the stereo algorithm is now to search for the optimal path inside this matching matrix, taking into account the other constraints.

The algorithm explained in [100] searches for the optimal path in the matching matrix of Fig. 4.23. To traverse the matrix, per step a path can only move one pixel to the right, diagonally to the top-right or to the top. Going back to the left or to the bottom is ruled out because of the ordering constraint. If the path makes a step to the top, this represents an occlusion in the first image, which means this pixel is visible in the second image but not in the first. A step to the right represents an occlusion in the second image while a step to the top-right means a continuous surface. Figure 4.24 illustrates this.

Cost Function

To determine the optimal path, a cost function is assigned to every path which is defined as follows:

$$C(path) = \sum_{\text{matching } x_1,x_2} D(x_1, x_2) + \sum_{\text{left occl}} \beta_{occl} + \sum_{\text{right occl}} \beta_{occl} \quad (4.21)$$

First we go look at all pixels in the first scanline. Some of these pixels have matching pixels in the other scanline and others are occluded. The first term is the sum of all *dissimilarities* for the matching pixels x_1 and x_2. The dissimilarity can be defined as $1 - NCC(x_1, x_2)$ with NCC the normalized cross correlation of two windows centered around the two pixels with a typical size of 5×5 or 7×7. The second term gives a penalty β_{occl} to all occluded pixels. Finally we look at all the pixels in the second scanline and add a penalty β_{occl} for all occluded pixels. The value of the parameter β_{occl} determines how much weight we give to occlusions. If this value is high, we give a high penalty to an occlusion, forcing the path to be smooth. Lower values give more weight to the matching term, accepting only well matched pixels but increasing the amount of occlusions and thus unmatched pixels. We typically use a value of 20 for β_{occl}. The stereo algorithm will then calculate the cost for the different paths and select the cheapest one as the optimal solution. Dynamic programming and backtracking techniques are used to increase the performance. A typical improvement consists in making the disparity range symmetric by shifting one of the epipolar lines left or right. This improves the memory-allocation efficiency.

Hierarchical Stereo

The complexity of the search algorithm described above is $O(WH\delta^2)$ with W and H the width and height of the image and δ the disparity range. Large disparity ranges therefore have a fateful influence on the execution time. The chance on mismatches within large disparity ranges increases as well. This is the reason why the stereo algorithm is executed in a hierarchical way. The rectified images are subsampled a number of times, depending on the disparity range. At the highest (i.e. smallest) level, the disparity range is limited to e.g. $[-5, 5]$. The optimal path is searched for at this level and the solution is used as an initialization for the next level. At this next level we limit the disparity range again to a fixed width of e.g. $[-5, 5]$ around the initialization. By executing this algorithm for all levels, using the solution of the previous level as the initialization for the next, we can compute the optimal path at the last level with a complexity of $O(WH)$. As an extra advantage we ensure that, by starting the matching with the smallest image, large disparity values can be dealt with too. A downside of this technique is of course that a possible mismatch at the highest level can never be undone in the next levels if this mismatch is more than 5 pixels (for a fixed range of $[-5, 5]$) wrong in the next level.

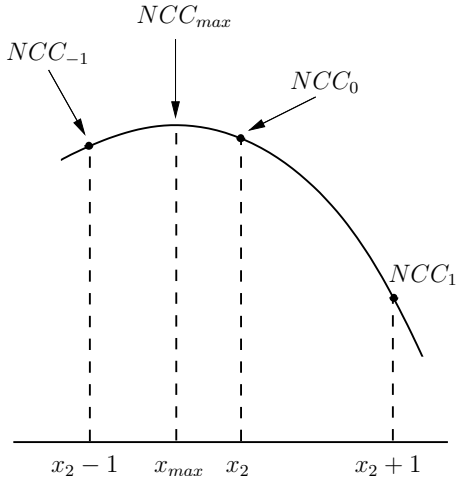


Figure 4.25: The optimal path of the dynamic programming solution says pixel x_1 in the left scanline matches to x_2. We compute the NCC with $x_2 - 1$ and $x_2 + 1$ as well and fit a parabola through the 3 NCC values. The position on the scanline x_{max} where the parabola reaches its maximum is taken as the subpixel match of x_1.

Post Processing

Even though the dynamic programming algorithm searches for the optimal solution of the path, some post-processing on this result is still possible. The step size of the matching in the algorithm is one pixel. We can improve this to sub-pixel accuracy by combining the information of neighboring matched pixels. A possible technique to refine the match between pixels x_1 and x_2 consists in matching x_1 with $x_2 - 1$ and $x_2 + 1$ using NCC. A parabola can then be fitted through the three values NCC_0, NCC_{-1} and NCC_1 and the x value where this parabola reaches its maximum can be selected as the floating point, sub-pixel match of x_1 on the second scanline. This technique is illustrated in Fig. 4.25.

The output of the stereo algorithm is a disparity map between the rectified images. This means that in order to find the matching pixels between the original images, the transformations depicted in Fig. 4.26 have to be performed. The pixel (x_1, y_1) is transformed to its rectified coordinates (x_{1r}, y_{1r}). A lookup in the disparity map tells us how many pixels we have to move to the left or to the right on the same scanline and yields the matching pixel in the second rectified image with coordinates $(x_{2r}, y_{2r} = y_{1r})$. Inverse rectification of these coordinates then yield the coordinates (x_2, y_2) of the matching pixel in the second image. Applying these transformations time after time can be a time-consuming task. If this lookup has to be done regularly, it is better to construct two displacement maps from the disparity map. These maps hold the x and y displacement of the pixels in the first image that have to be added to the

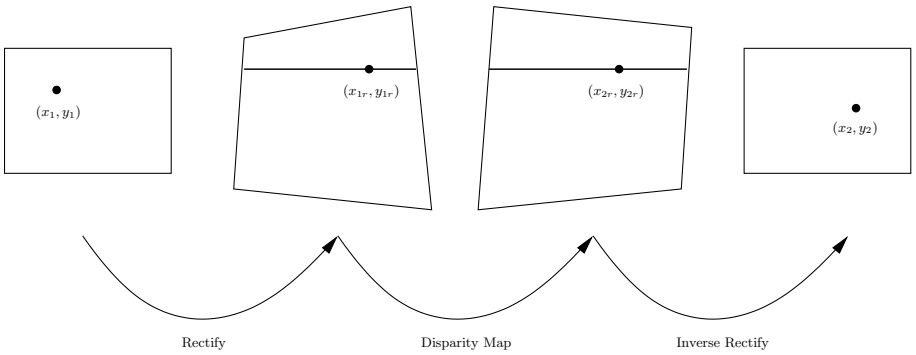


Figure 4.26: The transformations one has to apply to find the matching pixel in the second image comprise of a rectification step, a lookup in the disparity map and an inverse rectification.

original coordinates to end up with the coordinates of the matching pixel in the second image.

4.7.3 Linking Stereo Pairs

Section 4.7.2 explained how to compute dense matches between pairs of images. The input of Structure and Motion algorithms however typically consists of multiple images for which SaM computes the cameras and a 3D point cloud reconstruction. This means that there are multiple camera pairs to be considered. In this section we will explain an algorithm, proposed in [47], that allows to link the information of several stereo pairs together in order to create dense depth maps.

Consider a set of images, taken in sequential order and processed by the Structure and Motion algorithms described before. For every image pair, consisting of two consecutive images in the sequence, a dense stereo matching process can compute dense disparity maps or displacement maps. Because the consecutive images are relatively close together, they are similar which helps the matching process to yield accurate correspondences. Unfortunately, a 3D reconstruction based on the dense matches of such an image pair will be fairly inaccurate, due to the relatively small baseline between the views, as we can understand from Eq. 2.3. Large baselines yield smaller errors on the 3D reconstruction. However, matching wide baseline views poses many difficult problems. Because the images are taken further apart, they are less similar which makes them harder to match. A matching algorithm using simple comparison techniques like NCC will have a hard time finding correspondences because the affine deformation of the surfaces is not taken into account. Also, because of the wider baseline, the chance of occlusions increases and thus more image points will not be matched.

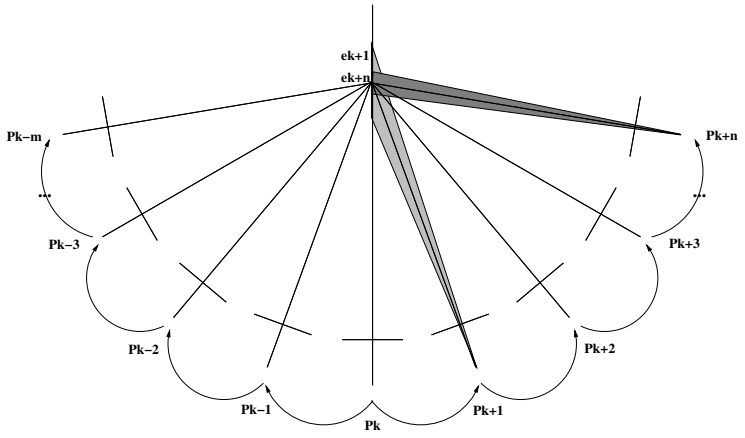


Figure 4.27: The linking algorithm illustrated. The figure shows a central view k. The 3D point is initiated from the match with view $k+1$ and the error e_{k+1} is computed. From the matches between views $k+1$ and $k+2$, the matching point in view $k+2$ is found and the 3D point and error band are updated. This process is repeated until no more match is found or until the reconstructed point from the new view falls outside the error margin. The linking is executed for both views to the right and the left of the central view.

The algorithm that is proposed in [47] tries to retain the best of both worlds by doing small-baseline matching but combining several of these matches to increase the reconstruction baseline. The concept is illustrated in Fig. 4.27. Each image k in the sequence we want to compute a dense depth map for is selected once as central view. The displacement maps between views k and $k+1$ are used to instantiate the 3D points between matching pixels in these views. Because of the small baseline we expect many corresponding points but also a large error distance e_k. After this initialization step, we move one view to the right. We use the displacement maps between views $k+1$ and $k+2$ to check whether reconstructed 3D points have a match in image $k+2$. For those that do, we reconstructed the 3D point from images k and $k+2$. If this 3D point lies within the error bound e_k we accept it, update the position of the point accordingly and also update the error to e_{k+1}. If the reconstructed 3D point falls outside e_k, we no longer have a consistent match and we break the link. This process is repeated until we hit the rightmost image or until no more 3D points are linked. The process is then repeated for the images on the left side of the central view, updating the already reconstructed points but also instantiating new points, that were not reconstructed between images k and $k+1$.

The linking approach of the dense stereo matches combines the strong points of small- and wide-baseline matching. However, it is only applicable if the set of images is recorded and processed sequentially. If this is not the case, the

prerequisites are not fulfilled and the algorithm gives bad results. This can for example be the case if the images don't form a sequence but just a random set in which it is not certain that consecutive images are also close. Another problem arises when two consecutive images are very close together so that 3D points are badly instantiated, which the linking algorithm is not able to recover from. Other approaches are able to deal with multiple images at the same time and combine the information in the different images in one go to yield dense matches. Approaches exist that make use of the increasing processing power of GPUs [15] or that combine multiple views, often taken from all around the object, like voxel carving [51], photo hulls [80] and level sets [22]. In [83] a Bayesian multi-view matching approach was developed where, the wide-baseline stereo problem is addressed from a probabilistic point of view, using pixel-based PDE matching algorithms [4, 85]. In chapter 7 we will describe a web-based reconstruction service that is capable of processing non-sequential imagery and recovering the structure and motion. In section 7.3.6 we will discuss the results of two of the cited dense multi-view matching algorithms.

4.8 Conclusion

In this chapter the important practical problem of degenerate cases was introduced of which a planar scene is the most common and thus most important. We explained how Torr's GRIC can help us to distinguish between planar and non-planar scenes. F- and H-GRICs lie at the basis of my system that recovers the SaM from long video sequences. I devised an intricate scheme with sub-sequences in which keyframes are found using the aforementioned GRICs. I can identify and discard blurry frames before processing which improves robustness. My coupled self-calibration algorithm recovers the metric reconstruction and my gluing algorithm knits the sequences together once more, allowing to compute the calibration of all intermediate frames as well, a necessity for 3D TV applications.

When the intrinsic parameters of the cameras are known, one can compute the essential matrix, rather than the fundamental matrix. If a 5-point algorithm is used, this method does not suffer from planar degeneracy. I implemented Nister's rather intricate 5-point method but also showed that in specific cases, like a tilting stereo head, one can get away with the much simpler 7-point method if one combines the data of different tilts.

When the calibration of the cameras has been computed from the original feature sets, we can improve the realism of the resulting reconstruction by searching for many more matches and hence 3D points. The linking approach described in this chapter combines the results of dense stereo processes, run on consecutive image pairs. The stereo algorithm we employ was written by a master thesis student under my supervision [100].

Chapter 5

SaM and Space Robotics

5.1 Introduction

The VISICS research group has been active in several projects of the European Space Agency. Six of those were carried out for the Automation and Robotics Section (TEC-MMA) of the European Space research and Technology Center (ESTEC) in The Netherlands. Because of the robotics interest of this ESA section, the focus was mostly on the use of vision for the enhancement of space robotics. In this chapter we will discuss the work and the results of the VIABLE project.

The VIABLE consortium consisted of the Belgian companies TRASYS-SPACE and Space Applications Services and the K.U.Leuven departments PMA and ESAT-PSI. This project was the first collaboration between ESA and its Japanese counterpart, the National Space Development Agency (NASDA). In the scope of the project, several robotic experiments were performed on the Japanese research satellite ETS-VII. Section 5.2 describes the specifics of the robotic and vision setup. Sections 5.3 and 5.4 discuss the calibration issues and pose-estimation experiments, respectively. Sections 5.5 and 5.6 describe experiments on the 3D reconstruction of the taskboard and on robot calibration.

5.2 Viable Setup

5.2.1 The ETS-VII Satellite

In VIABLE VISICS was responsible for the vision-based robotic control (VBRC) experiments. These were executed in conjunction with the joint Visual Interactive Autonomy Bi-Lateral Experiments (VIABLE) of ESA and NASDA on the Japanese research satellite ETS-VII [106] of which Fig. 5.1 shows an artistic impression. The satellite consists of a large target (named Orihime) and a small chaser satellite (named Hikoboshi) that can be separated for rendez-vous ex-



Figure 5.1: The Japanese ETS-VII satellite, consisting of a large target and a small chaser satellite. The bottom part of the target satellite contains the robot and the taskboard on which the VBRC experiments were conducted.

periments. The names were actually chosen as a reference to a Japanese myth in which a cowherd and a weaver are separated by the gods and can only come together once per year. The bottom part of the target satellite in Fig. 5.1 shows the space robot and taskboard on which the VBRC experiments were conducted.

This robot arm is a 6-DOF robot manipulator, equipped with two sets of cameras. The VIABLE experiments had access to a taskboard that allows several tasks to be executed by the manipulator. A grapple-fixture can be attached to the end effector. In order to guide the robot during experiments, the taskboard contains sets of 3-point calibration markers with known 3D positions in the taskboard reference frame. The top image of Fig 5.2 shows a top-view close-up of a such a calibration marker. The bottom image shows a side-view of the typical use of this marker: two 3-point markers stand side-by-side. One marker consists of a black background on which three white circles are painted. The middle circle is larger and on it a black cylinder is constructed on top of which another white circle is painted.

The ETS-VII on-board vision system consists of two sets of cameras. The arm monitor camera (AMC) is mounted on the first joint and the arm hand camera (AHC) is mounted on the end effector of the robot arm. Each set contains two cameras, one primary and one redundant unit. Both can be utilized as stereo head with 60 mm baseline. Each camera records a grey level image with 668×480 pixel resolution with fixed focal length. The images are compressed

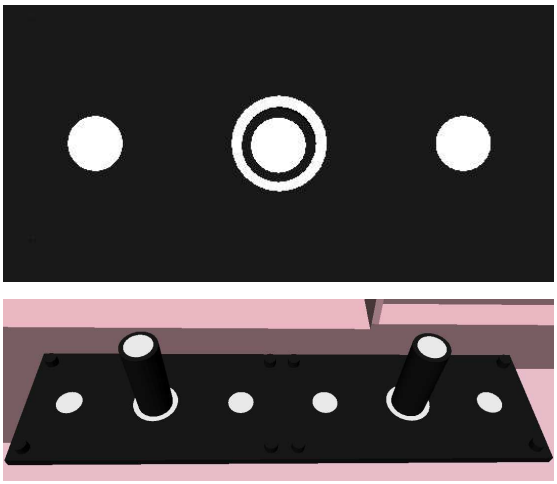


Figure 5.2: Top view of a three point marker (top). Side view of a typical use of two of these markers side-by-side (bottom).

with JPEG by a factor of 8.6 to yield a frame rate of 4 images per second on the video downlink. Two NTSC video channels allow access to two camera images simultaneously.

5.2.2 Station Setup

An important aspect of the VBRC experiments was the capability of the vision tools to allow operator intervention during execution of a vision task. Image processing and computer vision are processes with possibly many sources for errors that cannot all be modeled beforehand. Therefore a user-friendly graphical interface was developed to assist with the VBRC tasks. The interface allowed the operator to interact with the vision system to guide and help the automatic processing. While the human operator is very good at interpreting the scene and recognizing qualitative information, the vision system is good at precise quantitative measurements when given the appropriate input data. An important aspect of the graphical interface was the elimination of possible errors. The majority of the VBRC experiments on the satellite required on-line image processing. Communication with the satellite happened during windows of approximately 20 minutes. Any mistake would inevitably lead to the loss of precious operation time. Therefore, the interface was programmed to only allow for a limited set of actions, depending on the status of the experiment. For instance, during the pose estimation experiments (explained in section 5.4.1), the GUI would be launched with a newly acquired image. Then the operator had only one option: the delineation of a ROI (region of interest). Only upon successful completion of this task, the GUI would enable the detection of ellipses,

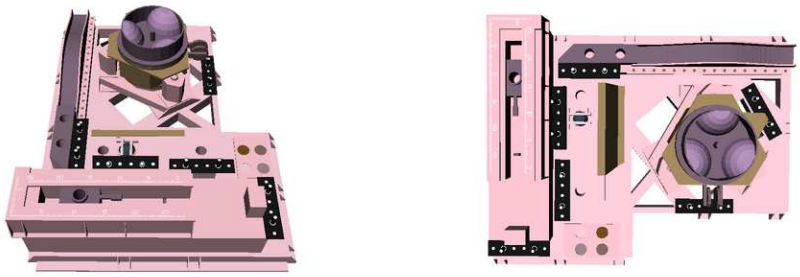


Figure 5.3: Mockup of the real ETS-VII taskboard including all visually significant parts.

and so on.

The VIABLE experiments were performed on the ETS-VII satellite commanded from the base-station in Tsukuba. Operating such a satellite is not a trivial task. All commands undergo several simulation tests before authorization can be obtained to execute them on the real system. In order to simulate and verify the VIABLE experiments in our own lab, before making the trip to Japan, a photo-realistic 3D digital model of the taskboard and the robot was constructed. This model served as reference for the IA path planning (in ROBCAD) and the VBRC visual simulator (in OpenInventor[66]). The model allowed for the realistic visual simulation of all aspects of the experiments.

Verification of this simulation was performed with a mockup taskboard of scale 1:1. It contained all visually significant parts and served as a realistic testbed for the VBRC experiments. Fig. 5.3 shows two views of the mockup taskboard, containing the calibration markers, holes, slider, sinusoidal surface and other features of the real taskboard.

5.2.3 VBRC Experiments

The VBRC experiments were designed to

- enhance the image quality to allow for better visual control,
- refine calibration of the system components (intrinsic camera parameters, eye-hand calibration) based on the calibration markers,
- perform on-line pose estimation procedures and guide the robot by automatically refining the robot pose,
- aid the operator during the experiments with visual clues using augmented reality techniques,

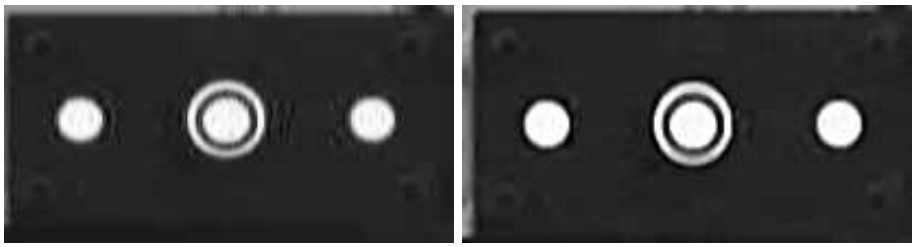


Figure 5.4: A 3-point marker before (left) and after (right) the enhancement filters. The edges are much sharper and the contrast is more outspoken. The ringing around the edges has disappeared.

- provide material for post-mission robot calibration and testing of advanced methods for uncalibrated vision experiments.

5.2.4 Enhancing the Image Quality

The first set of experiments evaluated the impact of the imaging conditions in space (degradation of the images due to noise, image compression, direct sunlight, etc.) and derived parameters for image preprocessing. These experiments were performed beforehand, using images of the real system that were sent to our lab. Analysis of these images yielded a set of parameters for image-enhancement filters. Evaluation led to the following filter sequence:

1. **a median filter.** This non-linear filter effectively removes spikes and noise in the image but preserves the edges. It was chosen for its capacity to remove the ringing that typically occurs around the edges of an image when JPEG compression is used. Because the JPEG-ringing was quite severe, a window-size of 5×5 pixels was used for most images.

2. **a binomial filter.** This low-pass filter smoothes the image to remove noise. It has the advantage over standard mean filtering that its frequency response has no ripples.

3. **a sharpening filter**. This unsharp-masking filter cancels the smoothing of the edges caused by the previous filter.

4. **radial distortion compensation.** This filter undoes the quite severe radial distortion of the images.

5. **aspect ratio compensation.** This procedure restores the original aspect ratio of the image which was changed due to the conversion to NTSC.

These preprocessing filters were applied to all incoming images before further processing. Fig. 5.4 shows a detail of an image of a 3-point marker before and after the enhancement filters.

5.3 Calibration Experiments

Online calibration was one of the crucial needs in the VIABLE project because no a priori calibration or in the best case a pre-flight version of the intrinsic camera parameters, the eye-hand, or robot pose was available. Only approximate calibration parameters could be obtained from the specification documents and from a limited set of images taken while the system was still on the ground. We therefore designed a set of calibration experiments that verified and refined the approximate calibration from images during the flight segment. These experiments made use of an invariant-based feature extraction procedure which is discussed in paragraph 5.3.1. This procedure was also used in other experiments (like pose estimation or robot calibration).

5.3.1 Invariant-Based Feature Extraction

The online calibration relied on the 3-point markers and their given 3D position on the taskboard. One of the employed techniques in this task was to reliably extract image features and find the corresponding 3D features. Correspondences were found robustly and completely automatically by exploiting viewpoint invariant relations. Two strategies were used.

Cross-Ratio Strategy

If enough markers were visible, a strategy could be employed that uses the viewpoint-invariant properties of the cross-ratio of 4 collinear points. The algorithm defined lines as the 6 points belonging to a pair of neighboring 3-point markers. The strategy implemented the following steps

- Marker points were extracted as ellipse centers. Although, strictly spoken these centers are not projectively invariant, the ellipses are so small, that the projective foreshortening has very little effect.
- Collinear points were found and the cross-ratio of all sets of 4 collinear points was computed. Because this can become a huge combinatorial problem, the solution of which could never be done inside the communication windows of 20 minutes, a ROI was selected by the operator, delineating the area in which the markers were present.
- Lines were sought as follows. When a set of 4 collinear points yielded a cross-ratio corresponding to one of a group of 4 out of the 6 points of a line, this set cast a vote for this line. Lines were found if enough votes were cast.
- In the next step, the lines were identified. Since all lines were in one plane, their intersection points could be searched. The cross-ratio of these points together with 3 marker points, yielded the identification of the lines.

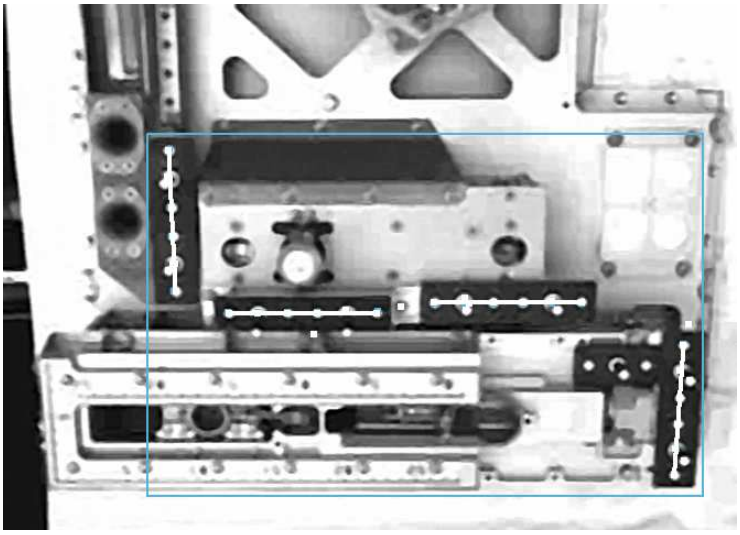


Figure 5.5: Marker points (found as ellipse centers) are grouped into lines using viewpoint invariant relations. The lines for each marker block are superimposed (in white) over the image for visual confirmation. Correspondences between the lines and marker blocks are also computed which yields 3D-2D relations.

- From the marker and line identification, all extracted marker points could be labeled, as shown in Fig. 5.5.

Common Tangent Points Strategy

If the camera was closer to the taskboard and only one 3-point marker was visible, a different approach was used. Because the ellipses could be extracted more reliably in this case, we could use them (and not only their center points) to find enough 3D-2D correspondences. The fact was exploited that tangent points of two ellipses with a common tangent are invariant under projective transformations. Actually, all bitangent lines, and not only those of ellipses or conics, are projectively invariant [98] but our proof will only show invariance for conics because this is the property that was employed.

Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be two conics (ellipses in our case). The set of lines, tangent to these conics is given by the dual conics $\mathbf{C}_1^*$ and $\mathbf{C}_2^*$. A common tangent line $\mathtt{l}$ to both conics therefore has to satisfy both equations

$$\mathtt{l}^T \mathbf{C}_1^* \mathtt{l} = 0 \tag{5.1}$$

$$\mathtt{l}^T \mathbf{C}_2^* \mathtt{l} = 0 \tag{5.2}$$

If the line $\mathtt{l}$ is transformed by the projective transformation $\mathbf{H}$, then the transformed line is given by

$$\mathtt{l}' \simeq \mathbf{H}^{-T} \mathtt{l}$$

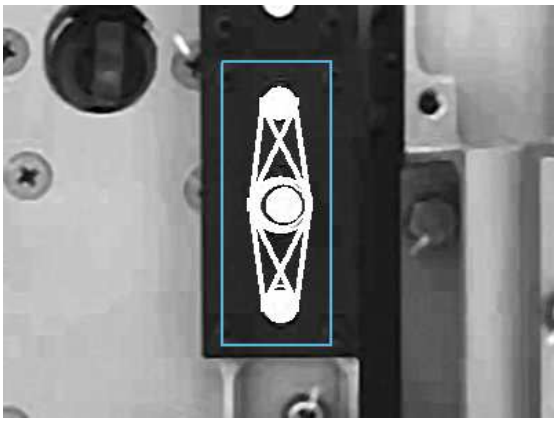


Figure 5.6: Common tangent lines of ellipses yield tangent points that are invariant under projective transformations. The ellipses and their common tangent lines are superimposed (in white) over the image for visual confirmation. This yields 2D-3D relations.

But since the relation between the dual conic and its transformation $\mathbf{C}^{*\prime}$ is given by Eq. 3.28 as

$$\mathbf{C}^{*\prime} \simeq \mathbf{H}\mathbf{C}^{*}\mathbf{H}^{T}$$

Eqs 5.1, 5.2 become

$$\mathbf{1}'^{T}\mathbf{H}\mathbf{H}^{-1}\mathbf{C}_1^{*\prime}\mathbf{H}^{T}\mathbf{H}^{-T}\mathbf{1}' = \mathbf{1}'^{T}\mathbf{C}_1^{*\prime}\mathbf{1}' = 0 \tag{5.3}$$

$$\mathbf{1}'^{T}\mathbf{H}\mathbf{H}^{-1}\mathbf{C}_2^{*\prime}\mathbf{H}^{T}\mathbf{H}^{-T}\mathbf{1}' = \mathbf{1}'^{T}\mathbf{C}_2^{*\prime}\mathbf{1}' = 0 \tag{5.4}$$

So we see that a line, tangent to both conics, is still tangent when the line and the conics are transformed by a projective transformation.

Since the common tangent lines are projectively invariant, so are the common tangent points. By computing these tangent points between the three coplanar circles, we obtain many identified 2D points, yielding 3D-2D correspondences. In Fig. 5.6 the common tangent point and lines are superimposed over one of the 3-point markers.

5.3.2 Eye-hand Calibration

For robot guidance from images the relative transformation between the cameras and the robot tip frame – the *eye-hand calibration* – has to be known. Procedures using calibration objects are known in literature [78, 96, 102] but thanks to the specific setup of the Viable taskboard and robot, a procedure could be developed that was specially targeted towards the ETS-VII robot. When the robot executed the procedure to grasp the grapple-fixture (GPF), it came into contact with the taskboard in a predefined position and orientation. In this specific pose, the cameras were approximately aligned with 3-point markers. These

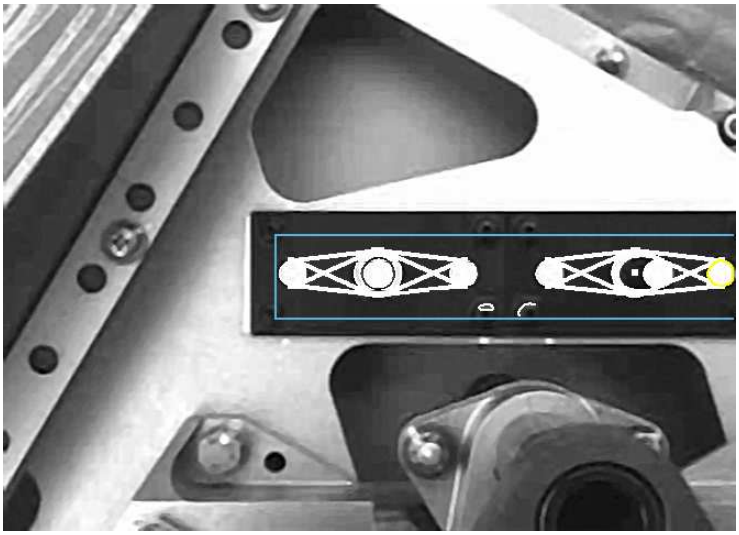


Figure 5.7: Setup for the eye-hand calibration experiment. The robot is touching the GPF and the AHC is above a 3-point marker. The common tangent points are found and based on these 2D-3D relations the camera pose is computed. This yields the eye-hand calibration of the camera.

markers were large enough in the image in order to be exploited to compute the camera pose with the second technique explained in section 5.3.1. Based on the computed camera pose and the fixed robot pose, the relative transformation from one to the other (i.e. the eye-hand calibration) could be calculated. Fig. 5.7 shows a screenshot of the image and the extracted tangent points during the operation of this calibration experiment.

5.4 Pose Estimation and On-Line Robot Guidance Experiments

Several experiments concerning pose estimation (using calibration markers) and on-line robot guidance were performed during the flight segments. These are explained in sections 5.4.1 and 5.4.2 respectively.

5.4.1 Calculating Pose from Known Markers

A first experiment consisted of calculating the robot pose from the known 3-point markers. The robot moved to a position where different markers were visible. Using the invariant relations described in paragraph 5.3.1, 3D-2D relations were found. These relations were the input for a robust camera pose estimation algorithm as described in section 2.2.2. An immediate verification

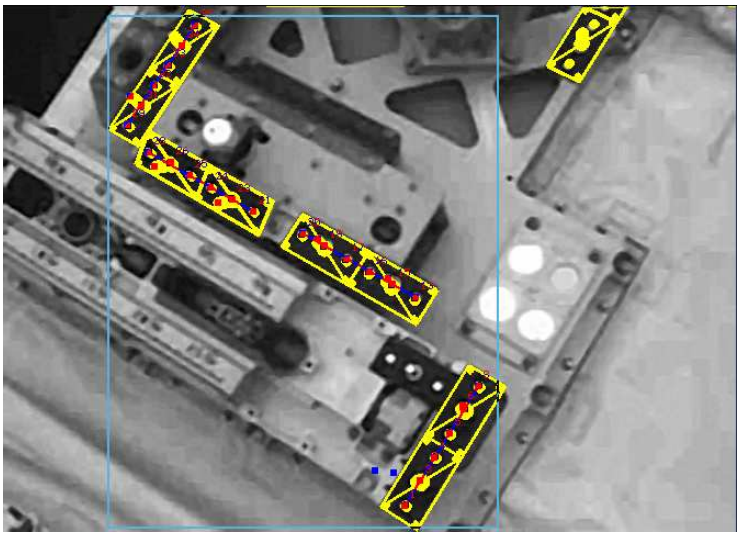


Figure 5.8: From automatically found 2D-3D relations the camera pose is computed. Parts of the model are superimposed over the real image and give a very good and intuitive verification of the calibration accuracy.

of the current calibration status and the accuracy of the computed position could be supplied to the operator by superimposing parts of the given CAD-model with the actual images, using the calculated position. An example of this superimposition can be seen in Fig. 5.8. A few notes on these results are in place.

- During the experiments, the intrinsic parameters of the camera were assumed to be known from a pre-flight calibration. If this had not been the case, a complete camera calibration (with both intrinsic and extrinsic parameters) could also have been performed. However, due to the planarity of the markers and their non-optimal distribution over the image, this was guaranteed to be problematic.

- The fact that the augmented reality image fits so well is proof of the accuracy of the calibration. While the CAD model of the workcell consisted of all markers, the pose estimation of Fig 5.8 only used the markers on the left side of the figure. Still, the marker on the right, close to the GPF is also perfectly superimposed on the image. This would not have been the case if the intrinsic parameters of the camera had been wrong.

A second step in this experiment consisted in moving the camera to a position much closer to one of the 3-point markers. The robot was intentionally positioned in a pose not perfectly above the marker. The second invariant method of paragraph 5.3.1 was used to calculate the camera pose, and using

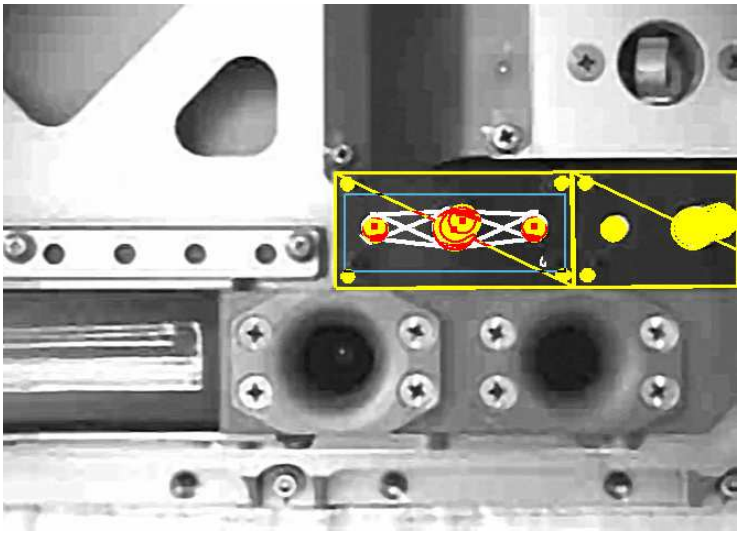


Figure 5.9: From automatically found 3D-2D relations the camera pose is computed. Verification of the result is possible by reprojecting parts of the CAD-model in the image.

the eye-hand transformation also the robot pose. Parts of the model were reprojected into the actual image to verify the calculation (Fig. 5.9). Using the computed pose, a relative translation and orientation change were computed to position the robot perfectly above the 3-point marker. The original image and the resulting image after execution of the movement are shown in Fig. 5.10.

5.4.2 Insertion of GPF Into a Hole

The ETS-VII robot had the possibility to attach a grapple-fixture (GPF) to its end effector and insert it into different holes and a slider on the taskboard. Usually, positioning of the robot was done manually by the operator who used the artificial markers as a visual clue. Actually, the right image of Fig. 5.10, where the inner and outer ring of the central marker are aligned, corresponds to the correct robot position for GPF insertion.

During the VIABLE experiments we showed that positioning could be done using the image of the hole or slider only. This is especially important for the case of the slider because its exact position is unknown due to possible previous motions. Using an ellipse-fitting algorithm the hole or slider was extracted and the center point was found. Because the robot approaches the slider perpendicularly, the perspective foreshortening effect on the ellipse is minimal and the center point can be assumed invariant. From the intrinsics of the camera and the size of the ellipse, the height of the robot could be estimated. This allowed the algorithm to compute a relative update of the current pose to position the GPF

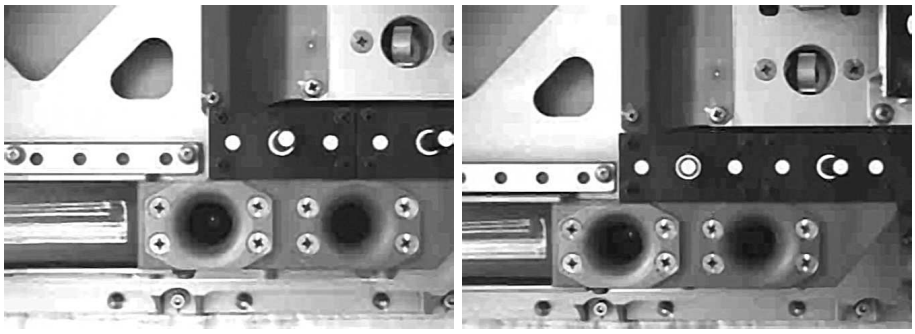


Figure 5.10: After a relative motion from the current position (left image), automatically computed by the vision-tools, the AHC ends up perfectly above the 3-point marker (right image). This is verified visually by the fact that the central marker tip is centered perfectly within the outer marker ring.

perfectly above the hole or slider. The image was augmented with the current impact point of the GPF (the point where the GPF would hit the taskboard if it were lowered from its current position) and the estimated impact point after relative motion. During operations the robot was deliberately mispositioned above both hole and slider. The algorithm managed to automatically update the pose to allow insertion. Fig. 5.11 shows both current (misplaced) and estimated impact point. Fig. 5.12 shows an image after the pose update. The robot is actually already descending towards the slider.

This insertion experiment is of course a poor man's version of true *Visual Servoing* algorithms. These algorithms constantly update the pose of the robot based on new observations in the images and their deviation from expected values [13, 40]. However, in the case of ETS-VII there was no possibility to directly interact with the robotic controller: only robot poses could be commanded which had to been authorized by both an on-ground simulator and a human operator first.

5.5 Taskboard Calibration and Reconstruction

5.5.1 Calibration of 3-Point Markers

The 3-point markers on the taskboard were important for vision-based algorithms. The calculation of the camera pose from 2D-3D relations, found by the algorithm, needed the exact 3D coordinates of these markers. Experiments were designed which could retrieve this information.

Because a good estimate of the 3D coordinates of the markers was supplied to us by NASDA, a quick and easy check on the consistency of this data could be performed. We moved the robot over the taskboard to different positions for which different markers were visible in the images. We computed the camera

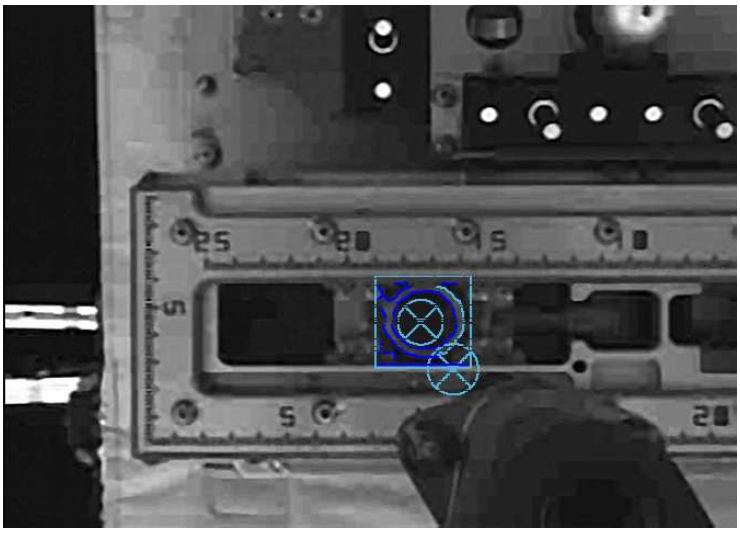


Figure 5.11: The vision system computes the current impact point of the GPF. The center point of the slider is extracted and the relative movement is computed to position the GPF above the slider. The predicted impact point is shown to fit into the hole.

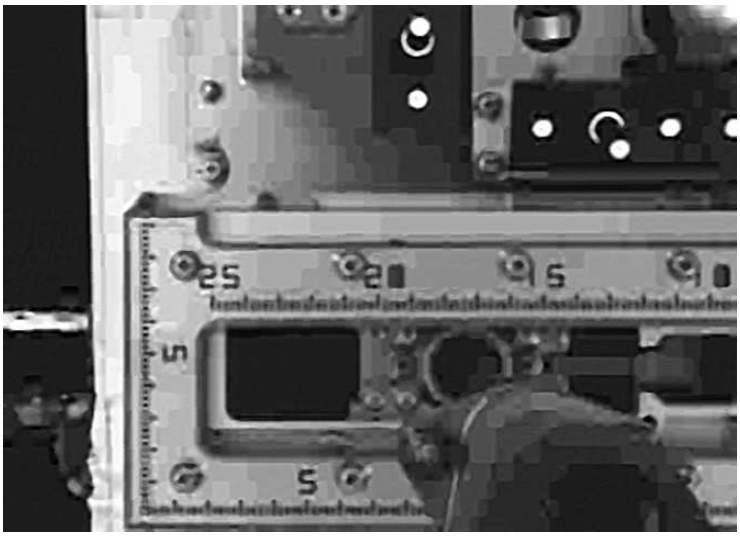


Figure 5.12: After the update in order to bring it above the slider hole, the robot can start its descent to insert the peg.

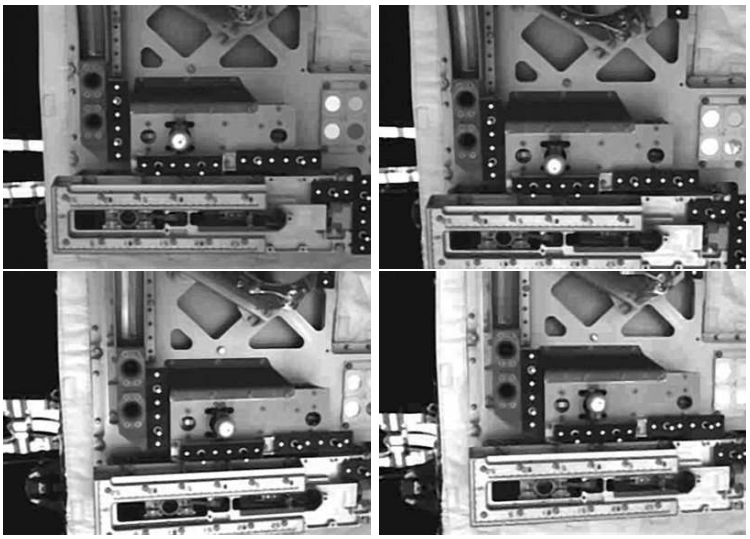


Figure 5.13: Some input images of the uncalibrated taskboard reconstruction algorithm.

pose based on the markers and reprojected the given 3-point markers in the original image. The estimated mean reprojection error was below a pixel which confirmed the consistency of the marker positions.

The coordinates of the 3-point markers could also be explicitly retrieved from images. This is what was done in another experiment. Three different images, taken from three different poses, showed the same 3-point markers. This time, the camera pose was not computed from these markers but rather retrieved from the given pose of the robot and the eye-hand calibration. Based on the identification of the markers given by the invariants, multiple-view matches were found. The markers could then be reconstructed in 3D by triangulation. The resulting data was consistent with the given 3D information up to the accuracy of the reconstruction of 2.23 mm in x, 1.45 mm in y and 0.84 mm in z.

5.5.2 Taskboard Reconstruction

Previous chapters discussed techniques for obtaining 3D reconstructions of scenes from nothing more than uncalibrated imagery. We applied these techniques on images of the ETS-VII taskboard. A set of images was recorded during one flight segment for this purpose. Fig. 5.13 shows some of the input images. The structure and motion result is shown in Fig. 5.14.

In the resulting camera calibration, the focal length is 885.665 (pixels) and the principal point is located at (335,233). The original pre-flight calibration computed a focal length of 935 and the principal point at (334, 239). The deviation is 5.3% for f and 1.5% for the principal point. The resulting ra-

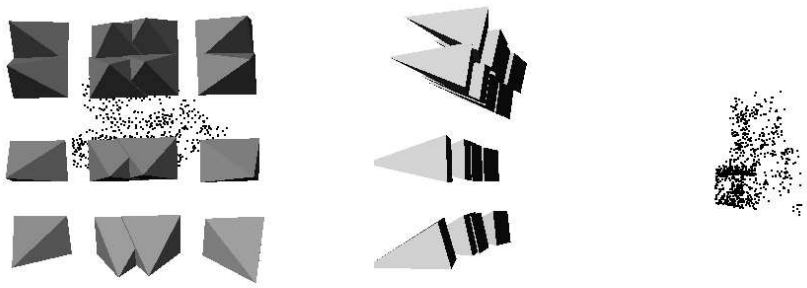


Figure 5.14: Point and camera reconstruction as output of the Structure and Motion process, run on the images of the ETS-VII taskboard.

dial distortion parameters were (0.00124481, -0.0130958). These are very small which is as expected because the images were first warped to undo the radial distortion effects, as explained in section 5.2.4.

Fig. 5.15 shows some results of the dense reconstruction of the slider-area of the taskboard. The figure shows views of the reconstruction without any manual refinement. In a post processing step it is possible to obtain reconstructions of parts of the taskboard by human interaction in the image only, using the computed depth data. Several spikes and other artefacts can be noticed in these images. These are due to the nature of the scene with large untextured or similarly colored areas, and to the lighting conditions in space which changed during recording of the images.

5.6 Robot Calibration Experiments

5.6.1 Robot Calibration

In the area of robotic control, calibration is absolutely vital. One can only move the robot from one pose to another if all aspects of the robot are known. The most important information consists of the length of the robot limbs and the joint offsets. When these are known, a readout of the joint values can easily be transformed into a Cartesian pose using a forward kinematics model. Robot calibration is a procedure that aims at improving the robot accuracy by modifying the robot positioning software, rather than changing or altering the design of the robot or its control system[76]. The procedure that is followed to obtain this goal is to position the robot in different poses, trying to excite all possible modes; then to measure these poses with a measurement system and

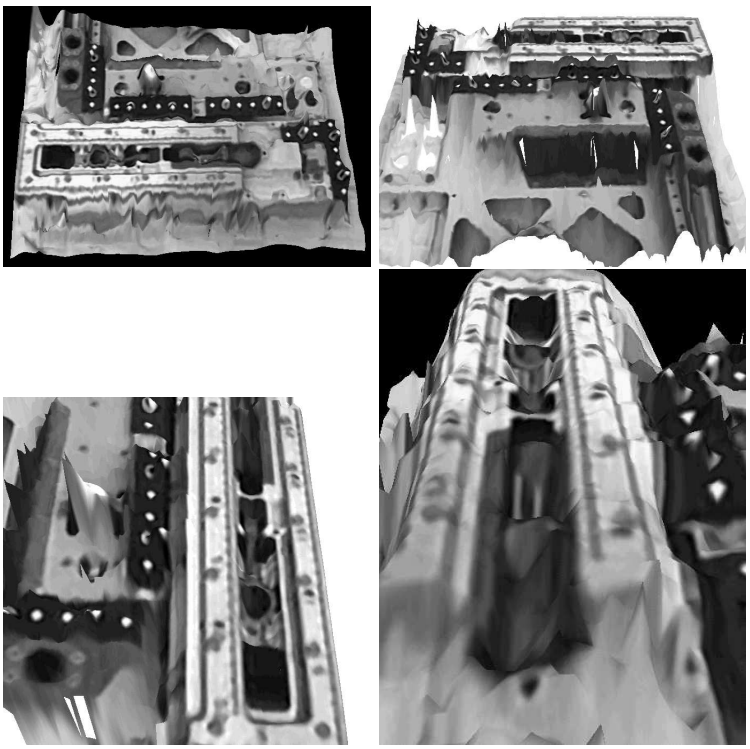


Figure 5.15: Different views from the dense reconstruction of the slider-area of the ETS-VII taskboard.

finally to compute the difference between these measured poses and the pose computed from the joints telemetry by the forward kinematics model of the robot. If all modes are excited sufficiently this allows to identify updates to be made to the current model. Standard robot calibration procedures obtain pose measurements from external measuring systems. In the case of the ETS-VII robot no such system was available. Instead we computed the robot poses from the 3-point markers as explained in paragraph 5.4.1.

Since the taskboard on which all VBRC experiments were conducted was placed in one corner of the ETS-VII satellite, we could excite only a limited range of values in joint space. Table 5.1 shows the excited versus the full range of the joints of the robot. Figure 5.16 illustrates the small variation of the orientation in space, which was to be expected because the camera (on last robot joint) must be targeting the workcell in order to employ the pose-estimation techniques.

Ten parameters were identified in the robot calibration procedure: four link

	J1	J2	J3	J4	J5	J6
actuated range	95.800	57.920	69.590	73.220	45.190	267.320
full range	300.000	200.000	185.000	220.000	170.000	300.000

Table 5.1: Actually covered range versus the full range of the joints in degrees

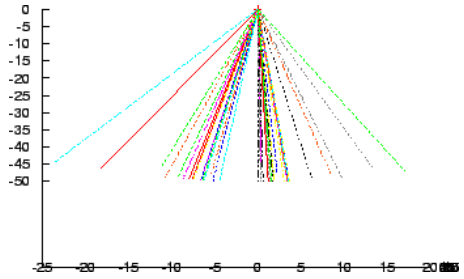


Figure 5.16: The orientation for all experiments show a limited range which is not beneficial for robot calibration.

lengths and six joint offsets. Their nominal and estimated values are shown in table 5.2. It is possible to use the calibrated model for calculating the end position of the robot. Using the same joint values for robot actuation, the generated Cartesian position should be closer to the measured values. Calculating the differences in x, y, z between the measurement points, the original calculated points and the calculated points after calibration gives the results shown in table 5.3. As expected, range and standard deviation are hardly affected, the mean value of the difference is however reduced. This is the only improvement the calibration can reach when the errors are mostly noise instead of showing a systematic behavior.

	L1	L2	L3	L4
Nominal	870	630	-267	277
Identified	838.87	631.66	-230.62	292.64

	O1	O2	O3	O4	O5	O6
Nominal	0	0	0	0	0	0
Identified	-0.048	-0.032	-0.004	-0.013	-0.017	0.016

Table 5.2: Nominal versus identified values of the four links and six joint offsets in mm

value	nominal-measured			calibrated-measured		
Range	113.316	37.420	23.080	82.041	42.428	23.746
Mean	9.3284	6.5935	-2.5777	-0.23776	0.40250	-0.99509
Std	21.0261	7.3233	4.5685	16.5512	9.0863	4.9108

Table 5.3: Range, mean and standard deviation of the differences in x, y, z between the measurement points, and the forward-kinematics points before and after calibration

5.6.2 Repeatability Study

During the last temporal window of the on-line experiments data was gathered to perform a brief repeatability study of the robot. The arm was moved to a certain position where three 3-point markers were visible. Then the robot was commanded to move a certain distance in a direction and return to the same position. This procedure was executed for three orthogonal directions and each time an image was taken by the AHC. Using the second procedure from paragraph 5.3.1 the camera pose and, using the eye-hand transformation, also the robot pose were computed.

The result of the computations is displayed in table 5.4. The first pose is the reference pose. Poses 2, 3 and 4 are poses obtained after a to-and-fro motion in x, y and z direction respectively. For each pose the computed position of the camera is shown. This data clearly shows a significant difference in the x and y direction. It indicates a repeatability error of a few mm in the robot movements. This is not unexpected because also the encoded joint values show very large deviations of more than 0.2 degrees per joint.Because of the very limited amount of motions and repetitions no statistical conclusions could be drawn from this data.

pos	x	y	z
pos 1	415.496	152.332	435.001
pos 2	419.604	152.325	435.071
pos 3	417.232	151.359	434.834
pos 4	417.920	154.069	435.442

Table 5.4: Result of the repeatability study

5.7 Conclusion

This chapter described the use of SaM techniques and algorithms for space robotics applications. The Japanese ETS-VII satellite served as a real-life testbed in space. Because of the dramatic lighting effects and the artefacts

due to compression, the images needed to be enhanced before processing. A pipeline of standard enhancement filters was implemented to this end.

Because of the binary nature of the images, most of the vision-based robotic control experiments on the satellite made heavy use of geometric invariants. I implemented techniques based on the cross-ratio of four collinear points or the common tangent points of ellipses. The robot carrying the camera was guided and controlled by my pose-estimation algorithms using these techniques.

Even though the images contained little texture, I was still able to run my uncalibrated SaM algorithms on them. The resulting camera calibration was accurate up to approximately 5% but the dense matching result was rather poor, as could be expected. Finally, some robot calibration experiments were performed by our colleagues of PMA, taking my computed camera poses as input.

Chapter 6

SaM and Planetary Exploration

6.1 Introduction

In 1996 the world was in awe when pictures of the *Mars Pathfinder* [31] mission arrived. The adventures of the rover *Sojourner* were covered in great detail by the media. This successful mission was repeated with two much smaller rover-missions, *Spirit* and *Opportunity*. The concurrent *Polar lander* mission was lost during descent.

Inspired by the success of the American rover missions, the ESTEC section of the European Space Agency sponsored a feasibility study, called PSPE, an acronym for Payload Support for Planetary Exploration. In this project an end-to-end system was developed for a planetary exploration mission by means of a small tethered rover. Space Applications Services in Zaventem was the prime contractor. Sub-contractors included the research groups VISICS and PMA of the K.U.Leuven, its spin-off company OptiDrive and the German companies Van Hörner & Sulger and DLR. Sections 6.2 up to 6.5 describe the vision aspects of such a mission. A completely new and somewhat futuristic approach to planetary exploration is the idea of using flying robotic platforms or *aerobots*. ESTEC's *Aerobots* project studied the possibilities of such a platform, focussing on the computer vision aspects and especially on the 3D reconstruction capabilities of the system. Sections 6.6 up to 6.11 describe this work performed by a team consisting of Space Applications Services and VISICS.

6.2 Rover Mission Study

6.2.1 System Setup

The PSPE study investigated the possibilities, problems and solutions of a planetary exploration mission, making use of a tethered rover commanded by a stationary lander. The PSPE system was designed to mimic an actual landing and exploration mission. It therefore consisted of three important parts working together as shown in Fig. 6.1.

- The Planetary Rover. In the spirit of the current *Faster, Cheaper, Better* approach to space exploration, ESA opted for the use of a small and simple rover: the *Nanokhod* [73]. This small vehicle is designed to carry instruments in the immediate surroundings of a lander. Based on an initial Russian design, the Nanokhod has been further developed by ESA and is the most probable candidate for an ESA rover mission. It is equipped with a tether cable, providing the rover with power and data connection to the lander which allows for a very high ratio instrument-mass/rover-mass. The rover consists of two connected tracks, able to traverse a wide variety of soil-structures, and a payload-cab which is subdivided into four chambers, in each of which an instrument can be placed to perform in-situ experiments. All chambers can be put in contact with the soil, a process called *docking*, to extract samples.

- The Planetary Lander. This is in essence the landing shell in which the rover, all communication equipment, computational and electrical power are stored. In PSPE the lander was not built explicitly. The most vital parts of it from an operational point of view were present though, namely the Imaging Head, for terrain reconstruction and rover localization, the On-Board computer and the Control System for both Nanokhod and Imaging Head.

- The On Ground Control System. All robotic exploration missions are guided and observed on Earth. The OGCS contained the necessary software to perform 3D reconstruction of the terrain from imagery, path planning, rover control and visualization of the received telemetry.

For the computer vision aspects of the project, the Imaging Head was vital. It was used for recording images from which a reconstruction of the planetary terrain was computed as will be explained in the following sections. The Imaging Head was also used for controlling the motion of the rover. For this purpose, the team of DLR made use of the Light Emitting Diodes on the payload cab of the rover (visible in Fig. 6.2). By lighting the LEDs in a predefined order and taking the difference with a background image, the LEDs and therefore the pose of the rover cab could be determined [52]. The Imaging Head consisted of a stereo head, mounted on a unit which allowed for pan and tilt motions and which was approximately 1.5 meters high. The two cameras of the stereo head were

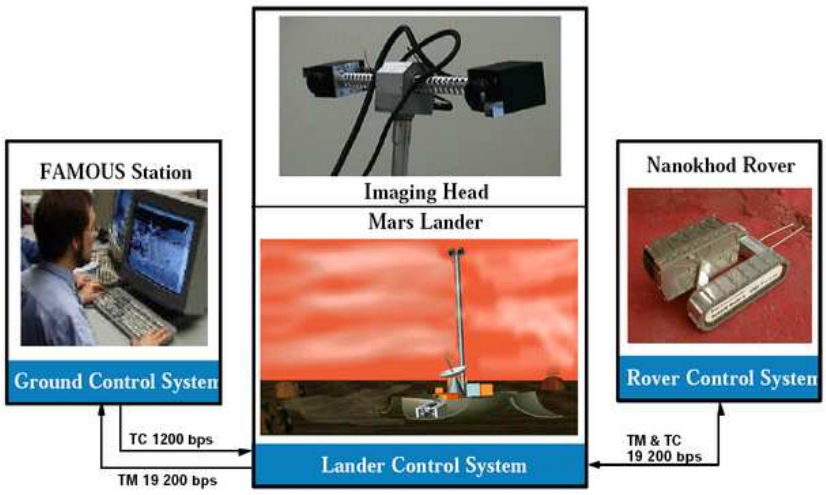


Figure 6.1: The three PSPE subsystems working together



Figure 6.2: The PSPE rover *Nanokhod*

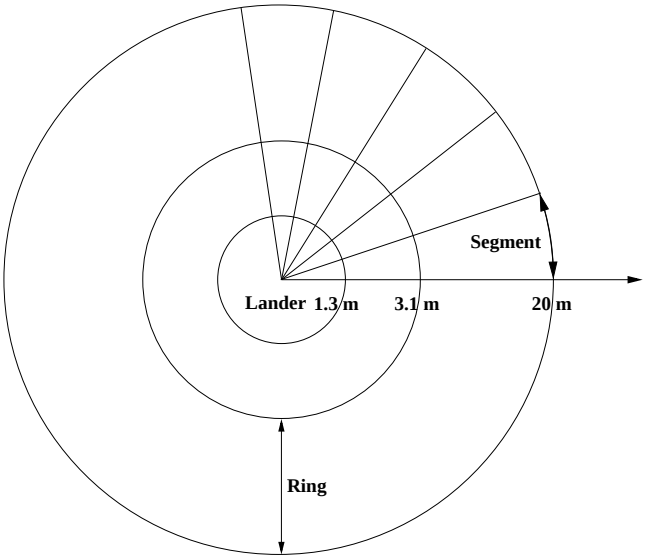


Figure 6.3: Segmentation of the terrain into segments and rings

space approved 1024×1024 CCD cameras constructed by DLR-Berlin, based on the ROLIS lander camera of the ROSETTA mission [60]. The stereo head had a baseline of 0.5 meter. A picture of it with attached cameras and cabling is shown in the center of Fig. 6.1.

6.2.2 Utilization

The PSPE study investigated the following typical utilization scenario of the lander and rover. As soon as possible after the landing of the planetary system the Imaging Head would be deployed. Because of the strain on the parts during launch and landing, the Imaging Head needs to be recalibrated. To accomplish this, it takes images of the terrain which are sent to earth where the calibration is performed using these images. From the same images a 3D reconstruction of the terrain is then computed. Since the cameras have a limited field of view (23×23 degrees for the PSPE cameras) the entire environment is not recorded at once but it is segmented into rings according to the tilt angle and each ring is divided into segments according to the pan angle of the Imaging Head (see Fig. 6.3). The outermost boundary of the recorded terrain lies at twenty meters from the camera. For each of the segments a stereo image pair is recorded and sent to Earth. The values of the actual pan and tilt angles can be read out from the encoders of the pan-tilt motors (accurate up to the possible influence of backlash) and are transmitted together with the corresponding images.

The 3D reconstruction of the planetary terrain, obtained on earth, is then used by a team of scientists to indicate possible interesting sites for the Nanokhod

to visit and extract and analyze samples from. A path-planning procedure is applied to compute a route for the Nanokhod to follow which minimizes a cost, combining different factors like the risk of tipping over, the risk of losing visual contact, etc. After thorough verification through simulation the route is uplinked to the lander. The control system on the lander observes the Nanokhod via stereo images, using the differencing technique mentioned above. This allows for the system to control the rover such that it executes the uploaded tasks. The results of the analysis of the samples together with the telemetry of the rover are sent back to the ground station where the latter is visualized in a 3D simulation environment.

In sections 6.3 and 6.4, the Imaging Head calibration process and the reconstruction of the planetary terrain in 3D are explained in more detail. Results of a system test in a planetary testbed at ESTEC are presented in section 6.5.

6.3 Calibration

Every planetary mission is a high-risk operation. During launch and landing, the lander and its contents are subject to extreme forces. The mechanical properties of the Imaging Head are likely to have been affected by mechanical and thermal effects. For high accuracy equipment, such as the Imaging Head, a small change in these mechanical properties results in a large degradation of the results, unless the new properties can be estimated. The cameras themselves are built so that the intrinsic parameters during the mission can be assumed identical to the parameters obtained through pre-mission calibration.

6.3.1 Using Markers?

Traditional calibration algorithms rely on known calibration objects with well-defined optical characteristics in the scene. By taking pictures of these artificial objects, the pose of the cameras can be computed, yielding the extrinsic calibration of the stereo rig. An example of this was given in section 2.2.1. There are two reasons why this scheme was not suited in the case of PSPE where the Imaging Head is deployed on a distant planet. First there is the problem of where to place the calibration objects. One needs to be absolutely sure of the pose of these objects for the calibration to have any meaningful result. It is of course impossible to add objects to the terrain, so one has to think of placing calibration markers on the lander itself. A typical lander consist of a cocoon which opens after landing, comparable to an opening flower. The markers could be applied to the opening petals. However, one is never sure of the exact position of these petals and hence of the markers. Even if one did dispose of accurate markers on the lander, a second problem arises. During the design of the Imaging Head, robustness was a very important issue and therefore the number of moving items was minimized. Thus, a fixed focus lens was chosen. Since the accuracy of stereo decreases with the square of the distance, the cameras are

focused on the far range to gain as much accuracy in the far regions as possible. As a consequence, the images of near regions are blurred. Since the markers would be on the lander, images of the markers would always be blurred, reducing the accuracy of the calibration up to the point where the markers are useless. It is clear that standard calibration algorithms could not be used for PSPE. A new strategy had to be developed that only uses images of the (unknown) terrain to calibrate the Imaging Head.

6.3.2 Strategy

The calibration procedure that was implemented for the PSPE project is able to calibrate the Imaging Head using images of the terrain only. This means that the images which are sent down from the planet to earth to reconstruct the terrain, can also be used for calibrating the Imaging Head. Therefore, the only overhead the terrain based calibration causes on transmission is due to the (small) overlapping parts we require between recorded rings and segments. The calibration of the extrinsic (mechanical) properties of the Imaging Head is split into two parts which are executed consecutively. First the relative transformation between the two cameras is computed. This is explained in Section 6.3.3. Once this relative calibration is performed, a procedure can be executed which computes the transformations between the cameras and the lander. This boils down to computing the pan and tilt axes of the pan-tilt unit. Section 6.3.4 deals with this problem.

6.3.3 Relative Calibration

The relative transformation between the two cameras of the Imaging Head can be computed from images of the terrain only. The algorithm to achieve this uses the concept of the essential matrix as explained in section 4.6.

Epipolar Geometry Estimation

Since the relative transformation between the cameras does not change when the different segments of the terrain are recorded, we can combine different measurements of the epipolar geometry to yield one accurate solution. This procedure was explained in section 4.6.4. It uses the 7-point algorithm to compute the fundamental matrix but due to the combination of images from all rings and segments does not suffer from a possible planar degeneracy. The combination further improves the stability of the F-matrix algorithm. Even if some areas around the lander contain little texture, there will most probably be enough textured areas that are recorded in different image sets. This ensures that the extracted matches cover the entire image area, which is vital for accurate epipolar geometry estimation.

Relative Transformation Computation

Once the essential matrix between the two views is computed, the relative transformation (position and orientation) between the two cameras can be calculated up to the baseline (i.e. the distance between the two cameras). First the essential matrix is obtained from Eq 4.6. The relative rotation and translation are then recovered using the procedure described in section 4.6.2. The absolute value of the distance between the two cameras, i.e. the baseline, cannot be computed from the images. It is unlikely, however, that this value will deviate much from the original value. Furthermore, since all measurements for 3D reconstructions and localization will consistently be carried out using the same stereo head, this has only very little influence. The only case where absolute measurements are really needed is for estimating tether consumption and the absolute size of obstacles to overcome. In these cases even a deviation of a few percent would be smaller than the uncertainty induced by other factors. The computed values for rotation and translation are used as an initialization for a non-linear Levenberg-Marquardt minimization which computes the values that minimize the reprojection error. The result is an estimation of the relative transformation between the two images that best fits the observations in the recorded images.

6.3.4 Pan-Tilt Calibration

Both the 3D terrain reconstruction and rover localization algorithms need to bring all measurements from the images into a single frame. It is not sufficient to have the relative calibration between the cameras only, but also the location of the pan- and the tilt-axis are needed. Since these values can also change due to the stress that occurs during launch and landing, we need to calibrate these axes.

The recovery of the pan- and the tilt-axis is more complicated than the relative calibration but can also be achieved from the image data, at least if some overlap exists between neighboring image pairs. This is guaranteed in the pan direction due to the fixed vergence set-up that does not allow 100% overlap between two views of the same stereo pair (and thus yields overlap with neighboring pairs). For the tilt axis care should be taken to foresee a sufficient overlap between the rings of Fig. 6.3.

Relative Transformations Between Views

To calibrate the pan and tilt axes, stereo images of the same ring and the same segment are used respectively. Especially the overlap between consecutive stereo images will come out to be important in the strategy.

For the calibration of the tilt axis, a stereo image of the outer ring of a certain segment is recorded. The Imaging Head is commanded to execute a tilt motion and to record a stereo image of the second ring. One has to make sure that there is sufficient overlap between the two image-pairs. This setup is

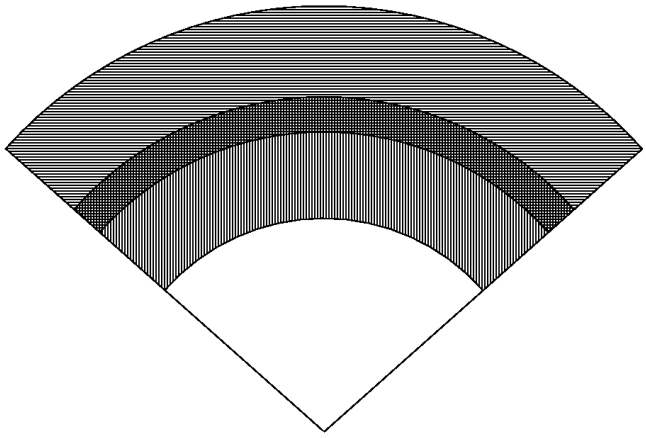


Figure 6.4: Symbolic representation of the setup for the computation of a relative transformation for a tilt motion

shown in Fig. 6.4. The area shaded horizontally is visible in the first pair. The area shaded vertically is visible in the second. The overlapping area is of course visible in both pairs.

Corresponding features in the images of the first image pair can be found as explained in chapter 3. In PSPE we used Harris corners and NCC of image windows for comparison. Because we have recovered the relative transformation between the two cameras, we can reconstruct the features in 3D in the Imaging Head frame. The same is done in the second image pair. Because of the overlap, some of the features will be visible in both image pairs. We can find correspondences between these features, using robust matching techniques on the two images of the left or the right camera. The corresponding features yield 3D-3D correspondences which allow us to align the reconstruction of the second pair with that of the first pair with a similar technique as the one explained in section 4.5.2. This yields the relative transformation between the first and second Imaging Head frame. Fig. 6.5 illustrates this procedure. Since only a motion around the tilt axis took place between the recording of the first and second stereo pair, which is a pure rotation around a 3D axis, we can recover this axis using the eigenvalue decomposition of the relative rotation. The eigenvectors $\mathbf{X}$ and corresponding eigenvalues λ of a matrix $\mathbf{R}$ have the property that

$$\mathbf{RX} = \lambda\mathbf{X} \tag{6.1}$$

For rotation matrices two of the three eigenvalues are complex conjugates and the third is 1. The eigenvector corresponding to this last eigenvalue is the axis of rotation because Eq. 6.1 shows that for $\lambda = 1$ all points $\mathbf{X}$ remain the same under rotation with $\mathbf{R}$.

In order to retrieve the pan axis we also want to compute the relative trans-

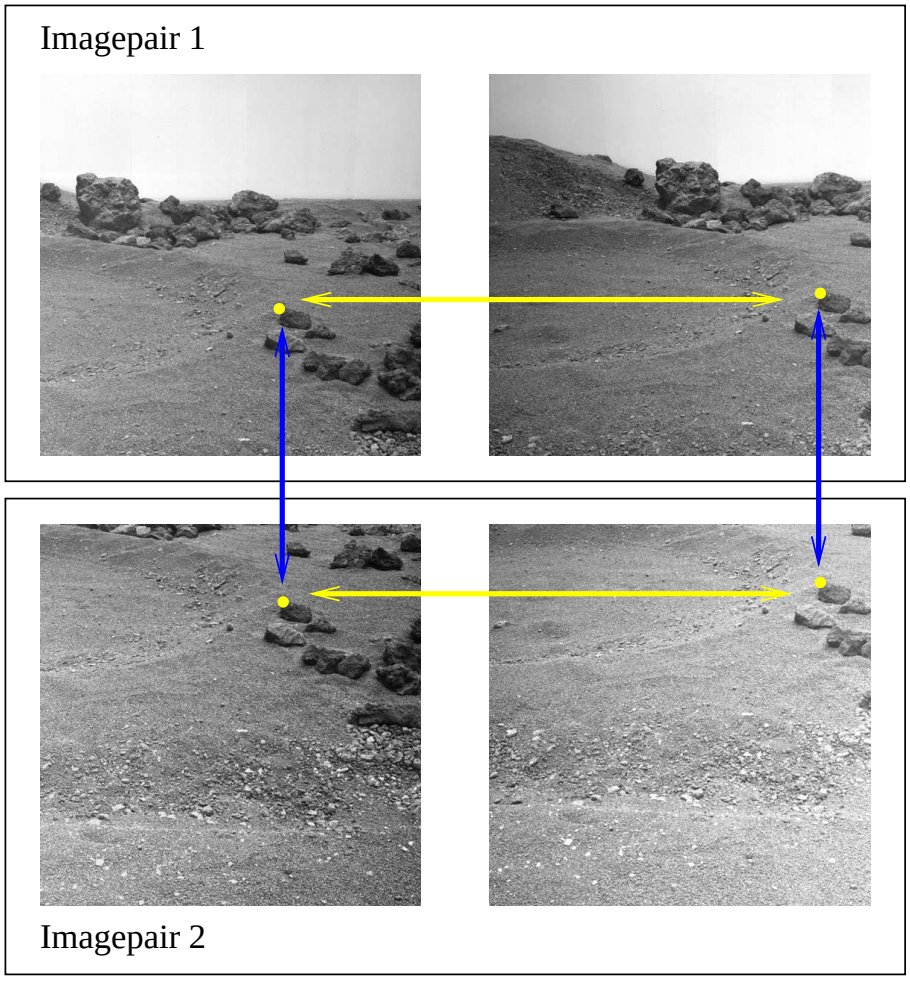


Figure 6.5: Different correspondences to compute the tilt axis. The upper two images are the left and right image of a stereo pair, taken for a tilt angle of 24 degrees. The lower images are the left and right image of a stereo pair, taken for a tilt angle of 14 degrees. The bright arrows denote correspondences between images of the same stereo pair. The dark arrows denote correspondences between left and right images respectively.

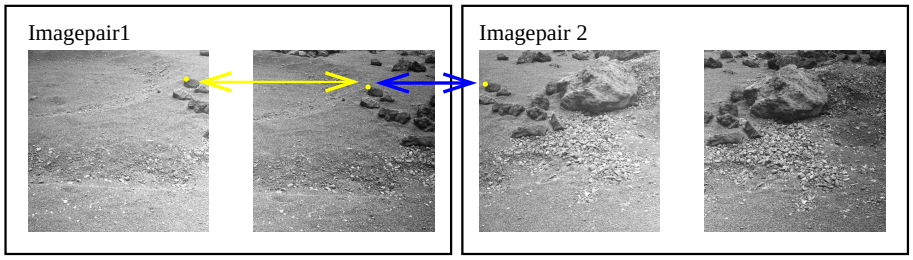


Figure 6.6: Matching between different views for a pan motion.

formation between views. This computation is now slightly different. The setup is shown in Fig. 6.6. It is clear that in this case there are almost no features that are present in all 4 images of the two views. Due to the verging of the two cameras however, there are some features that can be seen in one stereo view and in one image of the other image pair. One of these features is represented by a bright dot in Fig. 6.6. Again we find corresponding features between the left image of the first pair and the right image of the second pair, shown in Fig. 6.6 as the bright arrow. Because the features are visible in both images of the first stereo view, they can be reconstructed in 3D. They are also visible in one image of the second view and correspondences are found (the dark arrow in Fig. 6.6). One then applies a pose estimation of the camera of the second pair in which the features are visible, yielding the pose of this camera in the frame of the first pair. Since only a motion of the pan axis took place between the recording of the first and second stereo view, the pan axis can be computed from this result using the eigenvalue decomposition of the rotation matrix.

Actual Calibration of Pan and Tilt Axes

During the acquisition of the data one tries not to change the pan angle if a pure tilt rotation is executed and vice versa. In order to record all the stereo pairs, the stereo head executes several pan and tilt motions. Due to limitations on the repeatability of any real system, there will be deviations from the desired angles. This means that the computation of the tilt axis will not be correct because the algorithm computes the real rotation axis, which is not the true tilt axis if there is an –even small– pan component. But there is a solution to this problem. In the second step a good approximation of the pan axis was found, so if we account for the small deviations of pan, which are read out from the encoders, with the current computed value of the pan axis, we can recompute the tilt axis more accurately. This in turn allows us to update the pan axis etc. We can repeat this iterative procedure until the solution for the axes has converged. In reality three iterations have proven to be sufficient.

6.3.5 Synthetic Calibration Experiment

The calibration algorithm has been tested on artificial data. A virtual planar scene with texture from a real image from Mars was constructed and pairs of images were generated with a visualization toolkit. We chose a planar scene to show that our algorithms could deal with this degenerate case. First the relative calibration between the two cameras was computed. During calibration, data of 9 image pairs was combined and 2591 corners were matched to calculate the relative transformation. We could compare the result with the ground-truth value of the relative transformation. For comparison, the rotational part of the relative transformation is represented as a rotation around an axis (l, m, n) with a certain angle θ. The ground truth was

$$(l_0, m_0, n_0) = (0, 1, 0) \text{ and } \theta_0 = 10^\circ$$

The computed values were

$$(l, m, n) = (0.000397, 0.99951, 0.00018) \text{ and } \theta = 10.093^\circ$$

The angle between (l, m, n) and (l_0, m_0, n_0) was 0.0252 degrees. The difference between θ and θ_0 was 0.093 degrees. Both values are small, meaning that the rotation was estimated accurately because of the combination of data.

The pan and tilt axes were calibrated from the same data. The angle between the computed tilt axis and the ground truth was 0.138 degrees. The angle between the computed pan axis and the ground truth was 0.618 degrees. The larger error for the pan axis can be explained from the fact that only correspondences between three images are used to estimate it while correspondences from four images can be exploited to compute the tilt axis.

6.4 3D Terrain Modeling

After the calibration of the Imaging Head is performed, the process of generating a 3D model or models of the planetary terrain can start. This modeling is vital to accomplish the goal of planetary exploration. Its input are all images of the terrain and the calibration of the Imaging Head. The output of the terrain modeling can have different forms but the most important is the Digital Elevation Map (DEM). In the following sections we will describe the different steps that are performed to obtain such a DEM.

6.4.1 Disparity Map Generation

On each pair of images recorded by the Imaging Head, a stereo algorithm is applied to compute the disparity maps from the left image to the right and from the right image to the left. Disparity maps are an elegant way to describe correspondences between two images if the images are **rectified** first. As explained in section 4.7.1 the process of rectification re-maps the image pair to

standard geometry with the epipolar lines coinciding with the image scan lines. The correspondence search is then reduced to a matching of the image points along each image scan-line. The result (the disparity map) is an image where the value of each pixel corresponds with the number of pixels one has to move to the left or right to find the corresponding pixel in the other image. Details on the employed stereo algorithm were given in section 4.7.2.

Due to the fixed vergence of the cameras, very large disparity ranges (from -100 up to 500 pixels) can appear between two images of an image pair, especially in the very near and far regions of the terrain. This forces the stereo algorithm to search in a very large search range and causes problems with memory management and speed as well as stability because mismatches are more likely in such large search ranges. To deal with this problem we transform the images in such a way that the disparity range is reduced. The key to this reduction is the realization that there is one degree of freedom left in rectified images. Since all matching epipolar lines coincide with scanlines, we can shift all pixels on one scanline to the left or the right as a matter of an offset. The stereo algorithm will then find other disparities. Since most terrains are laid out horizontally, we can approximate the terrain in the first order by a plane. Hence we compute the homography that fits the corresponding features of the calibration process best. Using this homography, we shift every scanline to the left or right so that a point in the middle of the scanline and in the plane would have zero disparity. Because the terrains typically don't differ too much from the plane, the disparity search range can be greatly reduced. Fig. 6.7 shows an example.

6.4.2 Digital Elevation Map

A digital elevation map or DEM can be seen as a collection of points in a *top view* of the 3D terrain where each point has its own height or elevation. The algorithm proposed for generating regular DEMs in the PSPE project fills in a top view image of the terrain completely, i.e. a height value can be computed for every pixel in the top view image, except for pixels that are not visible in the IH because of occlusions. These occlusions are found in a very simple way. The terrain is divided into cells: the pixels of the DEM. For each cell the stereo pair image is selected in which the cell would be visible if it had a height of zero. A vertical line is drawn and the projection of this line in the left and right image of the stereo pair is computed. Fig. 6.8 illustrates the algorithm that is used to determine the height of the terrain on that line.

The two images on the left and right are the disparity images of the left and right stereo image respectively. We display the untransformed images here because they are smaller. The solid lines $\mathbf{L_1}$ and $\mathbf{L_2}$ are the projections of the vertical line in both disparity images. Now imagine placing a light where the left camera is. This light shines on the vertical line which throws a shadow on the terrain. In the left image this shadow of course has the same projection as the line itself. In the right image however this is not the case. The projection of the

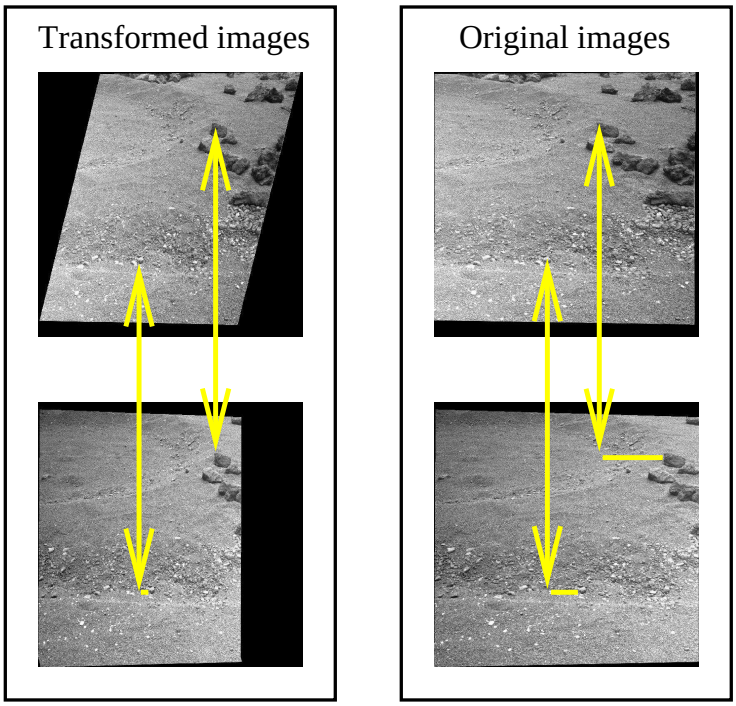


Figure 6.7: Transforming rectified images decreases disparity ranges by a large factor. On the left the disparity for two rocks in the transformed images is clearly much smaller than on the right in the original rectified images.

shadow in this image is the smooth curve $\mathbf{L_1'}$. The part of this curve above the intersection of $\mathbf{L_1'}$ and $\mathbf{L_2}$ is the real part of the shadow (i.e. it would be visible on the terrain). The other part can be seen as the virtual part of the shadow, coming from the part of the vertical line below the surface of the terrain. This shadow-curve can be computed using the disparity in the left disparity image of every pixel of the projected line $\mathbf{L_1}$. The intersection point of the vertical line and the terrain can then be found as the point where the shadow $\mathbf{L_1'}$ intersects the line $\mathbf{L_2}$. Some remarks about this procedure are in place.

- Occluded regions are detected easily. This is the case for cells that are not visible in both stereo images. The height value of these cells cannot be computed and these cells get a certain predefined value in the DEM which marks them as unseen.
- This particular scheme makes it possible to generate regular digital elevation maps at any desired resolution, interpolating automatically if needed.
- For the parts of the terrain close to the boundary of a ring, different parts of the vertical line will be projected in different stereo views. There-

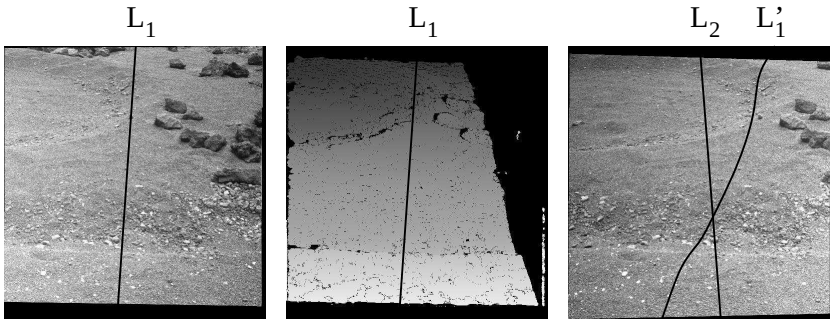


Figure 6.8: Digital Elevation Map generation in detail. The figure contains the left rectified image (left), the corresponding disparity map (middle) and the right rectified image (right). The 3D line L corresponding to a point of the DEM projects to L_1 resp. L_2. Through the disparity map the *shadow* of L_1 on the right image can also be computed. The intersection point of L_2 and L_1 corresponds to the point where L intersects the surface.

fore it is possible that information from two different stereo views has to be combined to find the intersection. This is not a problem because the transformation between the views can easily be computed from their calibration.

- The algorithm makes use of the intersection of lines in rectified images. This means that the polar rectification algorithm of section 4.7.1 cannot be employed because as was shown in Fig. 4.21 projective invariants like straight lines are broken.

6.5 System Test

A test and demonstration of the complete system was performed at the ESA-ESTEC test facilities in Noordwijk (The Netherlands) during the ASTRA 2002 conference. At ESTEC access to a planetary testbed of about 7 by 7 meters was available.

6.5.1 Calibration and 3D Modeling

The Imaging Head was set up next to the testbed. Its first task was the recording of the terrain according to the setup shown in Fig. 6.3. A mosaic of the pictures taken by this process can be seen in Fig. 6.9.

The autonomous calibration procedure was launched and it computed the extrinsic calibration of the cameras based on the images. Once the calibration had been computed the system rectified the images and computed dense disparity maps. Based on these, a Digital Elevation Map was constructed. The result



Figure 6.9: Mosaic of images of the testbed taken by the stereo head.

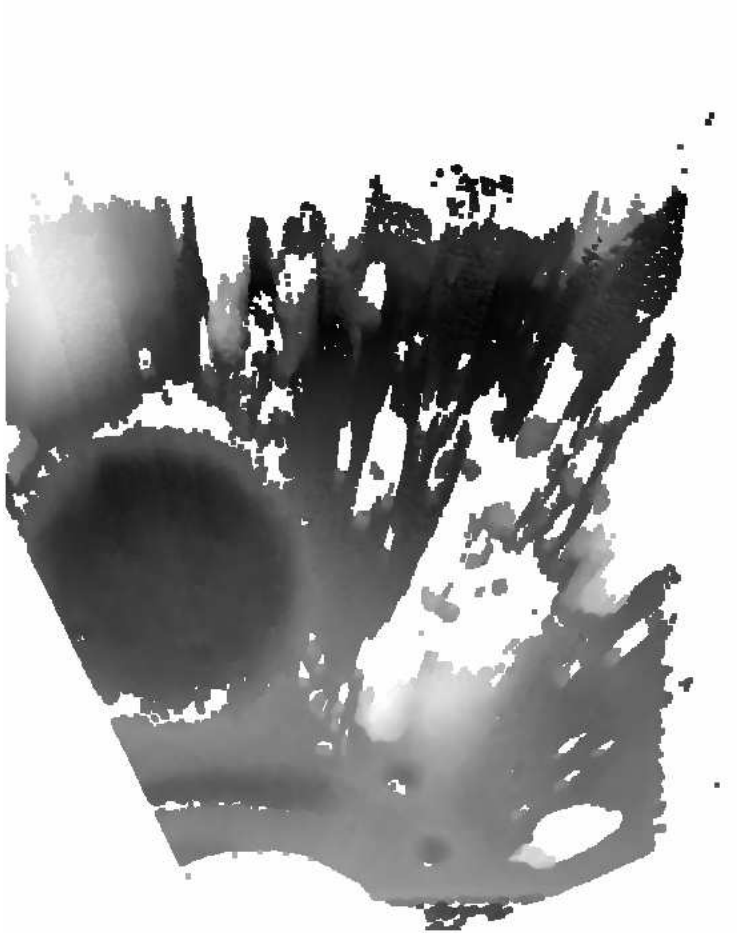


Figure 6.10: Digital Elevation Map of the ESTEC planetary testbed. A significant amount of cells is not filled in because they are located in occluded areas. Darker pixels are lower, lighter pixels have a larger height. One can clearly detect the crater in the middle left of the DEM and the hill behind it.

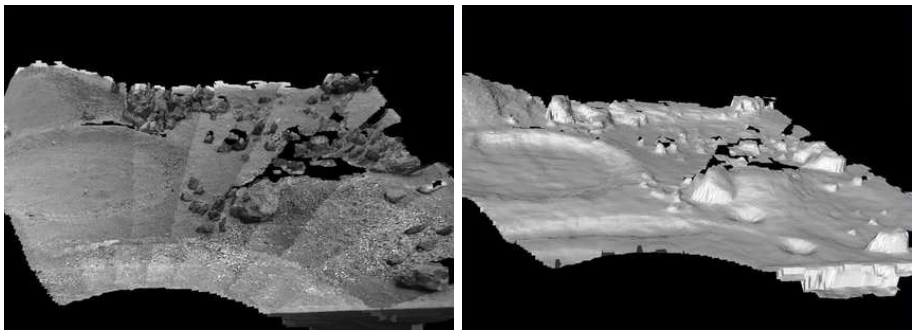


Figure 6.11: A textured and a non-textured view of the reconstructed model of the ESTEC planetary testbed.

can be seen in Fig. 6.10.

Because of the relatively low height of the Imaging Head (approximately 1.5 meters above the testbed) and the big rocks in the testbed, a large portion of the Digital Elevation Map could not be filled because of occlusions. The Digital Elevation Map of Fig. 6.10 was then used to construct a textured triangulated mesh model. Some views of this model can be seen in Fig. 6.11. A striping effect is visible in the texture of the model. This is due to a hardware problem of the cameras which caused the images to suffer from an horizontal intensity gradient. Since the 3D model combines texture data from all images, the boundaries are visible.

6.5.2 Simulator

Once the reconstruction was complete, the model was used to plan a trajectory for the Nanokhod rover. At that time, another component of the system came into play: the *Simulator*. This component allowed for the operator to simulate low-level commands on the Nanokhod and Imaging Head in a Virtual Reality environment. The operator could indicate the desired trajectory as a set of *waypoints* by clicking in a top view or a 3D view of the terrain. The position of the rover was computed quickly with a picking action, in which the Z-buffer of the renderer is used to quickly determine the intersection of the backprojected ray through the clicked position and the terrain. A simple gravitation model was used to predict the orientation of the Nanokhod on the terrain after a motion.

An overview of the GUI of this component is shown in Fig. 6.12. This GUI was highly configurable: the operator could select which views he wanted as well as their size. In Fig. 6.12 The GUI is divided into 4 windows. The upper two show what the left and right camera of the stereo head are seeing. The bottom right window is a 3D interaction window and the bottom left window shows a picture of what a camera, mounted in the cab of the Nanokhod would see.

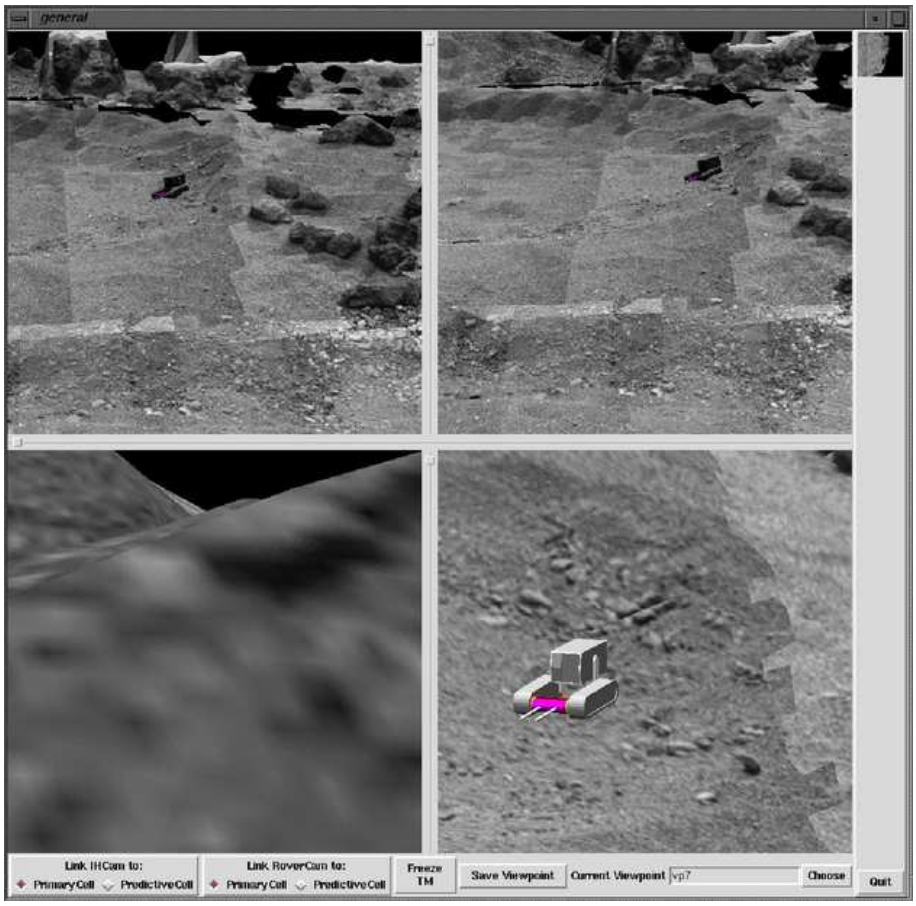


Figure 6.12: Overview of the simulator. The upper two windows simulate the view of the two cameras. The bottom right window is a 3D interaction window and the bottom left window gives a view from the cab of the rover.

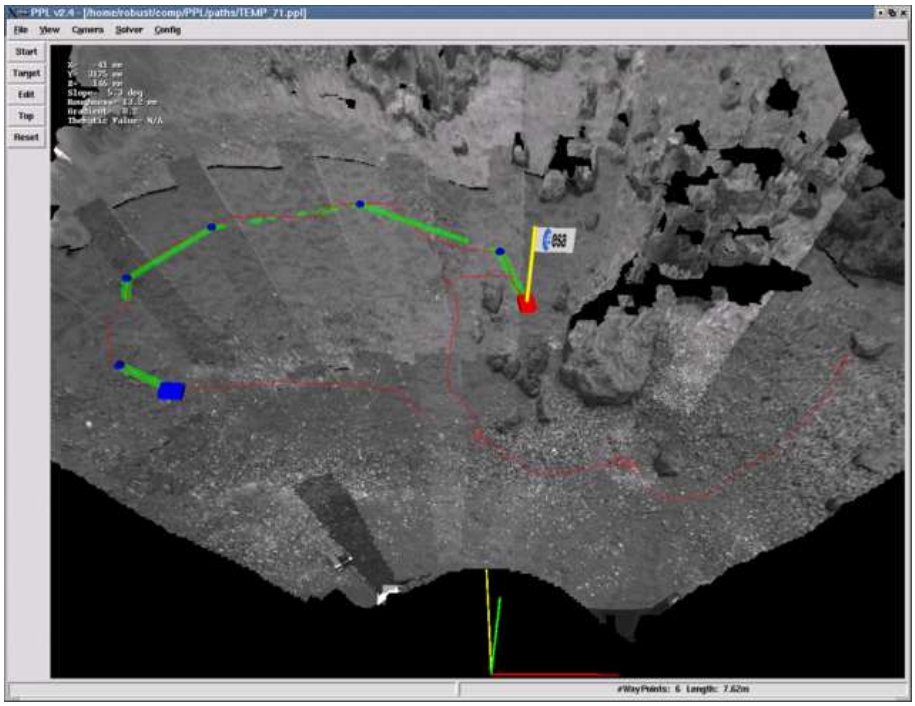


Figure 6.13: Results of the Astra 2002 experiment. On the 3D planetary terrain a path consisting of connected waypoints which the rover should visit is shown. The rover executed this path of approximately 6.7 meters autonomously during 5 hours. The reported telemetry is displayed as well and clearly coincides with the programmed path.

The Simulator component was written in TCL-TK [103]. This flexible interpreted language allowed for easy communication with other components of the system as well as an intuitive interface with the 3D rendering package VTK [75]. The Simulator could also be used to play back actual telemetry received from the planetary system and compare this to the desired trajectories. The choice for TCL proved to be valid here. The playback commands were sent by the communication component in such a format that they could be directly executed by the Simulator in a *safe interpreter*. No additional parsing or syntax checking methods were necessary.

6.5.3 Results

When the trajectory was simulated, the commands were uploaded to the lander system which executed them autonomously. The pose of the rover was observed using the Imaging Head. The lander needed the calibration parameters,

computed on the ground station for this and made use of the LEDs that are present on the Nanokhod. The telemetry, computed by the lander could then (after downlink to the ground station) be played back by the simulator and compared with the simulated pose of the Nanokhod. To show the autonomy of the system, the OGCS was installed in the conference center: another building of the ESTEC compound. During one of the conference sessions a long path was planned along the crater of the terrain and the commands were uploaded to execute it. Each time the rover was localized, its position was reflected in the virtual environment of the simulator component. The rover continued to move autonomously during the night along the crater and traveled a distance of about 6.7 meters. The next day the simulator showed that the rover arrived where planned (after 5 hours of travel) the day before. Fig. 6.13 shows the planned path and the actually followed trajectory. The deviations between these two are due to the specifics of the rover localization and control which only required a perfect match between real and planned rover-pose at waypoints. In between those points some leniency was allowed and the rover's heading was only adjusted when the discrepancy became too large.

6.6 Aerobots

6.6.1 Introduction

The Mars Polar Lander and Beagle missions have shown that atmospheric entry and especially landing on another planet is a high-risk operation. This section describes another approach to planetary exploration by means of aerial platforms or *aerobots*. The robotics section of ESA issued a study on the use of vision algorithms on aerobots. The remaining sections of this chapter will describe the aerobot system that has been built in the scope of this project as well as the setup and results of the tests performed at ESTEC's planetary testbed. In section 6.6.2 we will point out some general information and remarks on the possible use of aerobots for planetary exploration and in section 6.7 we explain the design of the system we developed in this project. Sections 6.8 and 6.9 describe the most important part of the system: the Imaging and Localization Package (ILP), capable of performing the computer vision processing needed by the aerobot. Finally, sections 6.10 and 6.11 discuss results for simulated and real operation.

6.6.2 Aerobot Systems

Aerial platforms can become an integral part of surface exploration missions on planets or moons with an atmosphere, like Venus, Mars or Titan. One of the most immediate applications for aerobots is ultra-high resolution imaging over extensive areas of the planet. Planetary Aerobot missions could prove very useful in the automatic detection of geological features and selection of possible

landing sites for a planetary exploration mission. The aerobot system can travel across many areas on the planet and send the appropriate data back to Earth.

Unfortunately the bandwidth available for data transmission from the planet to Earth is very limited. These bandwidth restrictions would require heavy compression on the images if the aerobot system were to transmit all collected imagery of the planet. This compression would definitely hamper the scientists on Earth to determine the interesting areas. The computer vision algorithms used to reconstruct the terrain in 3D would be disadvantaged as well. It is therefore imperative that the aerobot system has some degree of autonomy and can perform computer vision operations on its own which makes it possible to detect interesting geological features on the spacecraft. During the execution of its mission, the aerobot system has access to the uncompressed imagery, taken by its camera(s). It is therefore recommended to perform all critical computer vision processing on the system itself before the images are polluted by compression. The generation of Digital Elevation Maps is clearly a computer vision process that suffers from compression. The generation of these maps on the aerobot and the transmission of them, together with the compressed reconstructed parts of the images, will lead to a much better understanding of the observed areas, for the same amount of expended bandwidth.

All data collected by the aerobot needs to be correlated with the position from which the measurement was acquired. On long duration missions, the aerobot cannot rely on localization performed by an orbiter or from ground; it must have its own means. Uncalibrated structure and motion algorithms will play a role here, since they allow for simultaneous camera calibration (localization) and 3D reconstruction of the terrain. The specifics of an aerobot mission, such as almost linear motion of the camera, almost planar terrains, etc, require special care in the reconstruction pipeline and the model selection techniques of chapter 4 will have to be employed. Once calibration and reconstruction have been computed, scoring techniques can be applied to the data which now comprises of not only the images but of 3D information as well. The scoring algorithms should be written such that high scores for an image correspond to a large probability for interesting geological features to be found in that specific image. These scores can then be taken into account to decide on an appropriate compression factor, compression scheme and broadcast priority for the images.

6.7 Overall Aerobot Software System

The overall layout of the components of the aerobot system that are relevant is shown in fig. 6.14. Large boxes depict subsystems, rectangles with rounded corners depict commands and normal rectangles represent data that is sent from one component to another.

The aerobot is equipped with a camera system that records the planetary terrain underneath. The Imaging and Localization Package (ILP) requests images from this camera. Images are either taken at regular intervals or at specific

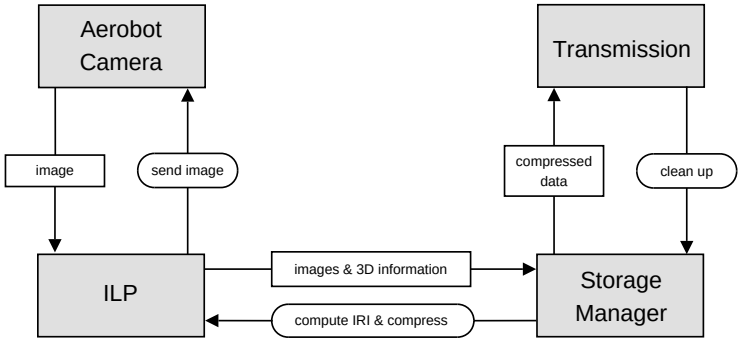


Figure 6.14: Overall layout of the relevant components of the aerobot system. Large boxes depict subsystems, rectangles with rounded corners depict commands and normal rectangles represent data that is sent from one component to another.

times, determined by the ILP (see section 6.8.1). The ILP then processes these images and generates 3D data comprising the camera positions and Digital Elevation Maps of the respective images (see section 6.8.2). This data is sent to the Storage Manager, in charge of managing the acquired and computed data with respect to the available storage on-board. In order to do so, it makes use of functionality provided by the ILP which compresses the data. Whenever the aerobot is in range, it can transmit data to the satellite. Upon successful transmission, the transmitted data can be deleted from the local storage.

The core of the aerobot system is the ILP. It is in charge of several tasks that have to do with image processing. Among these tasks are

1. asking for images and/or determining which images taken by the camera are suitable for further processing,

2. processing recorded images and computing camera poses and 3D information in the form of Digital Elevation Maps,

3. detecting relevant information in the images and DEMs and compressing the data.

The first two tasks together lead to a 3D reconstruction of the aerobot trajectory and planetary terrain and will be explained in section 6.8. The third task is needed by the Storage Manager and is described in section 6.9.1.

6.8 ILP and 3D Reconstruction

The core business of the Imaging and Localization Package is the computation of 3D information from the images that are taken by the on-board camera.

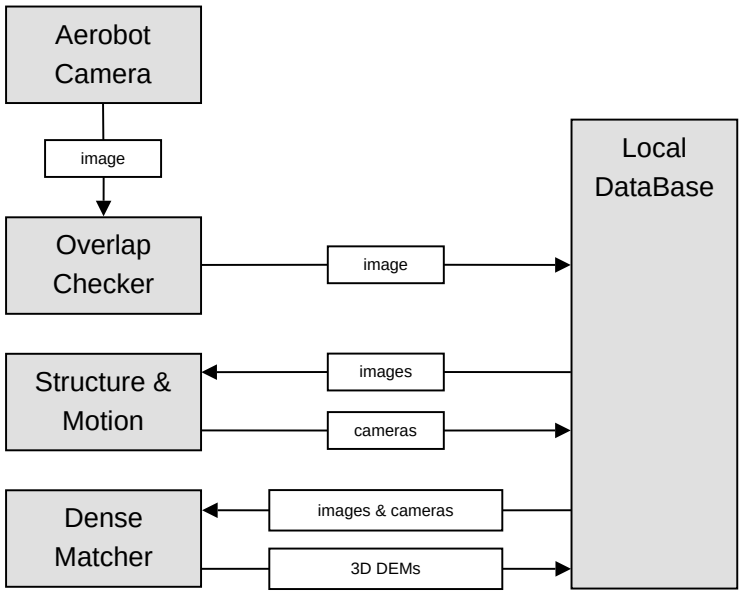


Figure 6.15: A schematic view of how the different subcomponents of the ILP collaborate. Images, received from the Aerobot Camera are the only input of the system. A local database connects all subcomponents.

Fig. 6.15 schematically shows how different subcomponents collaborate to deliver this result. All components are connected to a local database system. The advantage of this design is that local data is not unnecessarily duplicated. Current database systems like *postgresql* [17] furthermore implement notification techniques that can trigger other components when new or updated data is available. The setup of the ILP makes heavy use of this feature. It also allows us to run the different components on different machines that are connected to the database, thus increasing the processing speed.

The sole input data that is used by the ILP consists of images taken by the Aerobot Camera. The ILP Overlap Checker is in charge of determining which images will be used by the other components. It does so based on the overlap of this image with the previously selected image. Section 6.8.1 describes this functionality in more detail. When the first two images have been selected by the ILP Overlap Checker, the ILP Structure And Motion (or SaM) component is automatically notified by the local database and starts processing these images. This component is in charge of computing both the motion of the camera in the form of cameras for every image and a description of the structure of the observed scene in the form of a set of 3D points. Section 6.8.2 describes this component. When cameras are available for two or more images, the ILP Dense Matcher is notified by the local database and computes dense Digital Elevation Maps for the next image. This is done by dense stereo matching between the

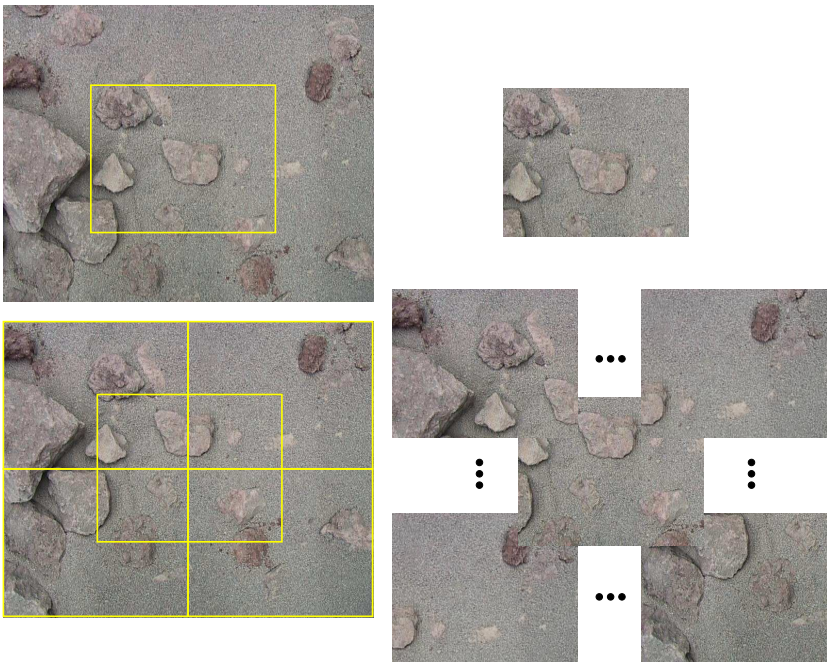


Figure 6.16: Schematic overview of the overlap algorithm. First the center part of the first image is selected. A window with the same size slides over the second image and the NCC value of both windows is evaluated.

new and previous image. Mask files are automatically generated that indicate which part of the image has not been observed by previous images and it is therefore crucial that it be kept by the Storage Manager. More information on the ILP Dense Matcher is given in section 6.8.3.

6.8.1 ILP Overlap

As explained in the previous chapters, Structure and Motion algorithms expect sequences of images as input of their processing. Consecutive images in such sequences should not differ too much since matching them then becomes a very hard problem. Nor should they be too similar, i.e. recorded too close to each other, since this setup typically causes large errors in the resulting calibration and 3D reconstruction. Success of the SaM component therefore greatly depends on the overlap between consecutive images. This is the reason why we decided to implement a first component which is capable of deciding quickly whether a newly recorded image has the right amount of overlap with the previously selected image.

The ILP Overlap Checker component is initialized with the first recorded image and a target overlap percentage, typically around 80-85%. When a new

image arrives the component computes the overlap between this image and the first image. If the overlap is at or below the target percentage, the new image is selected as the next image to be processed by the SaM component and thus inserted in the local database. If the overlap is larger, two actions can be taken. If images are recorded by the Aerobot Camera at regular intervals, the next image is simply awaited and the overlap between the previously selected and the new image is computed. If the Aerobot Camera can be commanded to record an image at a specific time, the optimal time at which we expect to record an image with the target overlap percentage can be easily computed by extrapolating from the previously computed overlaps and time-intervals between the images.

The overlap detection algorithm is based on comparing the two images under scrutiny with a comparison function like Normalized Cross Correlation (NCC). Fig. 6.16 shows a schematic overview of the algorithm. First the center part of the first image is selected. A window with the same size slides over the second image and the NCC value of both windows is evaluated. From the window position for the largest NCC value, it can be computed how many pixels the image content is shifted from the first to the second image, and thus the overlap value can be calculated. This is of course only an approximate value since effects like occlusion will have an impact on the accuracy. Speed of the algorithm is an issue, especially if we want to compute the next optimal time for recording. The images are therefore subsampled a couple of times. A beneficial side-effect of the subsampling is that the chance of local maxima of the NCC values is greatly reduced.

6.8.2 ILP Structure And Motion

The ILP SaM component is a separate process connected to the local database. Every time a new image is accepted by the overlap checker, it is stored in a table in this database and the ILP SaM is automatically notified of this update. The component retrieves the image from the database and processes it. If the process has just started, the second image is awaited and when it arrives an initialization of cameras and structure is performed. For subsequent images, the camera is computed and the structure is updated and extended.

Planar Degeneracy

The use of classical structure and motion techniques [70] was contemplated for this component. However, it is very likely that the aerobot will encounter planar surfaces, fly in a straight line or experience another type of degenerate motion or structure. We showed in chapter 4 that these situations cause trouble for classical techniques. That is why the concept of essential matrices and the 5-point algorithm to estimate them (see section 4.6.3) had to be employed because these do not suffer from the planar degeneracy as explained in section 4.6.4.

To be able to use the essential matrix, the intrinsics of the camera need to

be known. Two possible scenarios can be envisaged. The first assumes that the intrinsics are calibrated before launch and stay constant during launch, flight and deployment, similar to the approach in PSPE. Another possibility makes use of the GRIC computations explained in section 4.4. The process is as follows.

- Take a set of 4 or 5 consecutive images, output of the ILP Overlap Checker. Compute a projective reconstruction with classical structure and motion algorithms, using fundamental matrices and projective pose estimation.
- For consecutive triplets, also compute the best fitting homographies and determine which of the two models fits the data best using PPP-GRIC and HH-GRIC.
- If PPP-GRIC is clearly the better model for the triplets, the projective result from the sequence of images can be self-calibrated.
- This calibration is stored. From now on the derived intrinsics are employed for all images taken by the aerobot.

Windowed Bundle Adjustment

One of the algorithms that is employed in a typical SaM pipeline is a bundle-adjustment step. This is in fact a large non-linear optimization of all reconstructed 3D points and cameras and aims to minimize the global reprojection error in the images. Since a bundle-adjustment step changes both points and cameras, it is typically executed at the end of the SaM process when the information for all images has been computed. Unfortunately, in the case of an aerobot continuously recording images, such a final phase never arrives. Omitting the bundle-adjustment step is not a good idea since it has an important regulating influence on the calibration result. That is why we implemented a windowed version of this bundle adjustment. Since the aerobot typically moves forward all the time and in general doesn't revisit the same area, and since there is an overlap of 80-85% between consecutive images, after approximately 5 images no information of the first image is visible in the latest image. That is why we only minimize the error on points and cameras of the last 5 images that have been processed by the SaM pipeline. All cameras computed before remain fixed. This strategy gives us the best of both worlds: cameras are computed continuously (with a delay of 5 images) and a bundle-adjustment is still performed.

6.8.3 ILP Dense Matching

The windowed bundle adjustment step of section 6.8.2 delivers cameras that will not be updated anymore. These are kept in a separate table of the local database. When two fixed cameras are computed, the calculation of dense Digital Elevation Maps (DEMs) can start. This is the task of a separate process, also connected to the database and automatically notified of the availability of

a new camera. This process retrieves the latest pair of cameras and rectifies them so that epipolar lines become horizontal and coincide with scanlines as explained in section 4.7.1. Thereafter a stereo algorithm is employed on these rectified images. The dense correspondence scheme we employ to construct the disparity maps is the one described in section 4.7.2. The DEM for this image pair is then constructed via simple triangulation. How different DEMs can be combined is topic of discussion in section 6.9.2.

6.9 ILP and Storage

6.9.1 Image Richness Index

All planetary exploration missions need to take into account the limited bandwidth available to send data to Earth. An aerobot system could traverse many kilometers but the required bandwidth to send all recorded data back to Earth would almost certainly not be available. It is therefore imperative for the aerobot system to make decisions on the importance of its recorded data. Once calibration and reconstruction have been computed, scoring techniques can be applied to the data which now comprises of not only the images but of 3D information as well. The scoring algorithms should be written such that high scores for an image correspond to a large chance that interesting geological features can be found in that specific image. These scores, called Image Richness Index (IRI) can then be taken into account to decide on an appropriate compression factor, compression scheme and broadcast priority for the images and DEMs. For sake of demonstration a few simple IRI generators were implemented which assigned a value proportional to the gradient or cornerness content of the image. For real operations scientists should determine criteria by which the image richness can be assigned.

6.9.2 DEM Combination and Compression

The DEMs that have been computed as explained in section 6.8.3 hold for every pixel the distance between the terrain and the camera. Because consecutive images have an overlap of about 80-85% (see section 6.8.1), there exists much duplicated information. In light of the bandwidth constraints this problem must be dealt with. The solution lies in the form of *masks* that are constructed for all images. Fig. 6.17 gives a schematic overview of the process. The dense stereo matching of section 6.8.3 yields a matching pixel in the second image for almost every pixel in the first image of an image pair. For the first image pair all matched pixels in the overlapping area correspond to newly seen 3D points and thus this entire area must be saved. The corresponding mask for the first image is therefore filled where the grey area is visible in the left of fig. 6.17. For the following image pairs, matters are a bit more complicated, as shown on the right of fig. 6.17. Image 2 and 3 overlap again for a large part but only the grey area is new and the mask is only set for this region. The shaded area of image

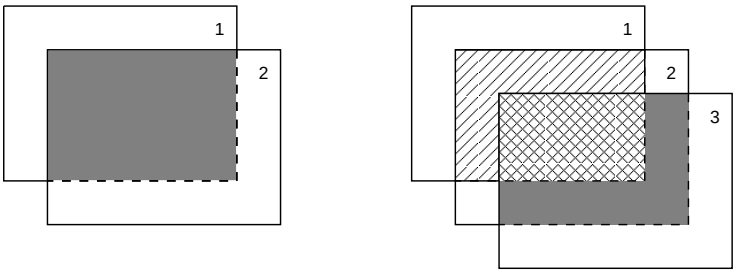


Figure 6.17: For the first two images (left), the mask is set for the area where both images overlap. From three images on, the mask consists only of the newly seen part, depicted in homogeneous gray.

2 is not visible in image 3 but has been reconstructed in image 1. The hatched area is visible in all three images. The already computed DEM of image 1 could be updated with this new information, e.g. using an averaging filter. In our setup we have decided against this because it again delays the time when a final DEM is ready for transmission.

It is not viable to send all recorded and reconstructed data to Earth uncompressed. The masks we have computed can be employed in the compression step. We have chosen to make use of the JPEG2000 standard [90] for compression of our image and DEM data. This standard allows for an arbitrary shaped region of interest (ROI) to be specified. The use of wavelets in the JPEG2000 algorithm allows for the selection of a certain area of an image which should be stored with high quality, while leaving the rest of the image at a lower quality. Thus the algorithm for compressing our image and DEM data first computes the IRI for an image. Based on this value and the relative size of the mask w.r.t. the entire image, a certain amount of bits are accorded to the data. The JPEG2000 algorithm is then executed and compresses the data with respect to this maximum, taking into account the masks as a ROI. Fig. 6.18 shows some results. The top left image is the original and the top middle is the mask. On the top right the JPEG2000 compressed image is shown with ROI enabled. The bottom left image is also JPEG2000 compressed with the same bitrate but without ROI. The last two images show a detail of the bottom left part of the compressed image with and without ROI respectively. It is clear that the ROI-enabled algorithm tries not to spend bits in areas that are not in the ROI. The detailed images clearly show artifacts that are introduced by the JPEG2000 compression without ROI that are not present in the ROI-enabled algorithm.

6.10 Tests Of ILP with Simulated Images

The aerobot system underwent its final acceptance tests at ESTEC on December 14, 2004. In order to test all aspects of the system, both simulated and real

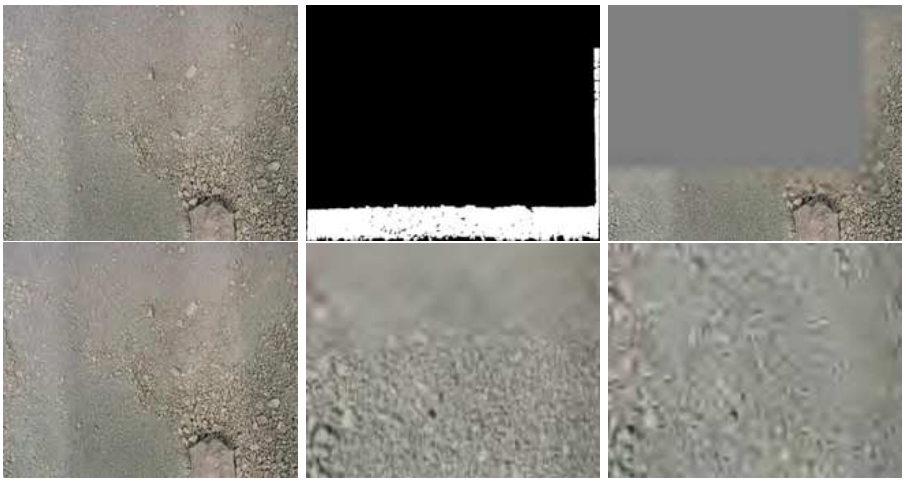


Figure 6.18: The top left image is the original and the top middle is the mask. On the top right the JPEG2000 compressed image is shown with ROI enabled. The bottom left image is also JPEG2000 compressed with the same bitrate but without ROI. The last two images show a detail of the bottom left part of the compressed image with and without ROI respectively.

data was fed to it. The aerobot system has a transparent interface to both the simulated and real acquisition mode. The results of the simulated tests are described in this section while section 6.11 describes the tests on real image data.

6.10.1 Authoring Test Cases

For testing and validation of the system's performance, a simulator was built. In this tool, an operator can author entire test cases. Amongst the parameters that can be chosen are the landscape over which the aerobot is flying, atmospheric conditions, such as fog or haze, lighting conditions, the aerobot trajectory over the landscape, and camera and satellite transmission parameters. Fig. 6.19 shows the GUI interface to author trajectories of the aerobot above a virtual terrain. The bottom left window is a top view on which waypoints can be indicated, the height of which can be selected in the bottom right window. The top window shows a 3D view of the terrain and the trajectory.

For the initial tests at ESTEC a virtual 3D terrain was artificially created. For 3D reconstruction algorithms to be successful and meaningful, enough 3D content and texturing information should be present in the terrain. It is however very difficult to find good 3D data with associated high-resolution top-view imagery. Therefore the virtual 3D terrain was created by texturing 3D MOLA scan data of Mars with an aerial image of Scotland. Some screenshots of this

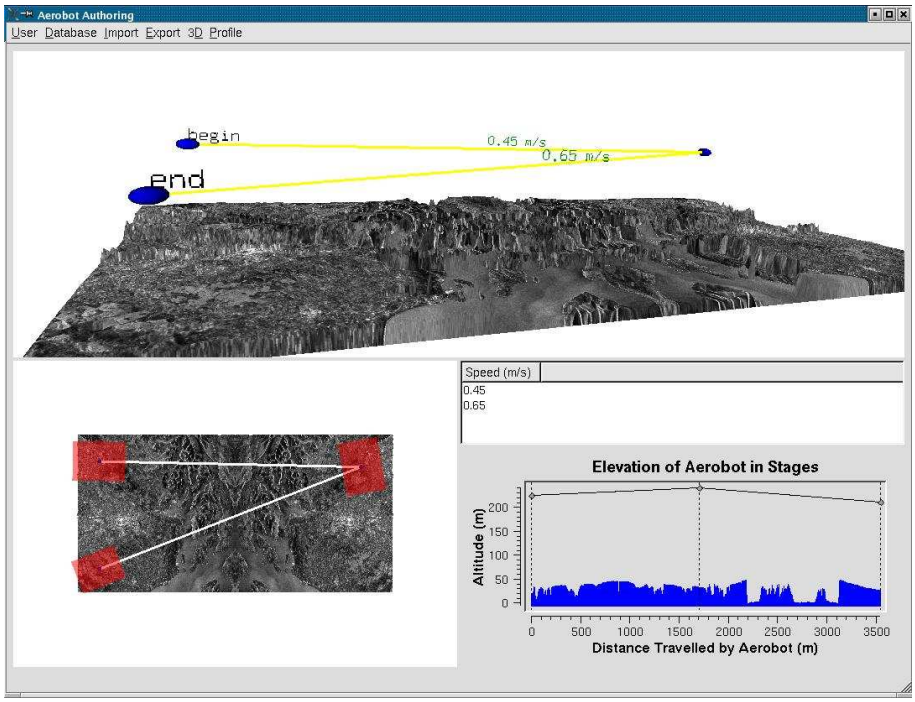


Figure 6.19: Authoring trajectory of the aerobot.

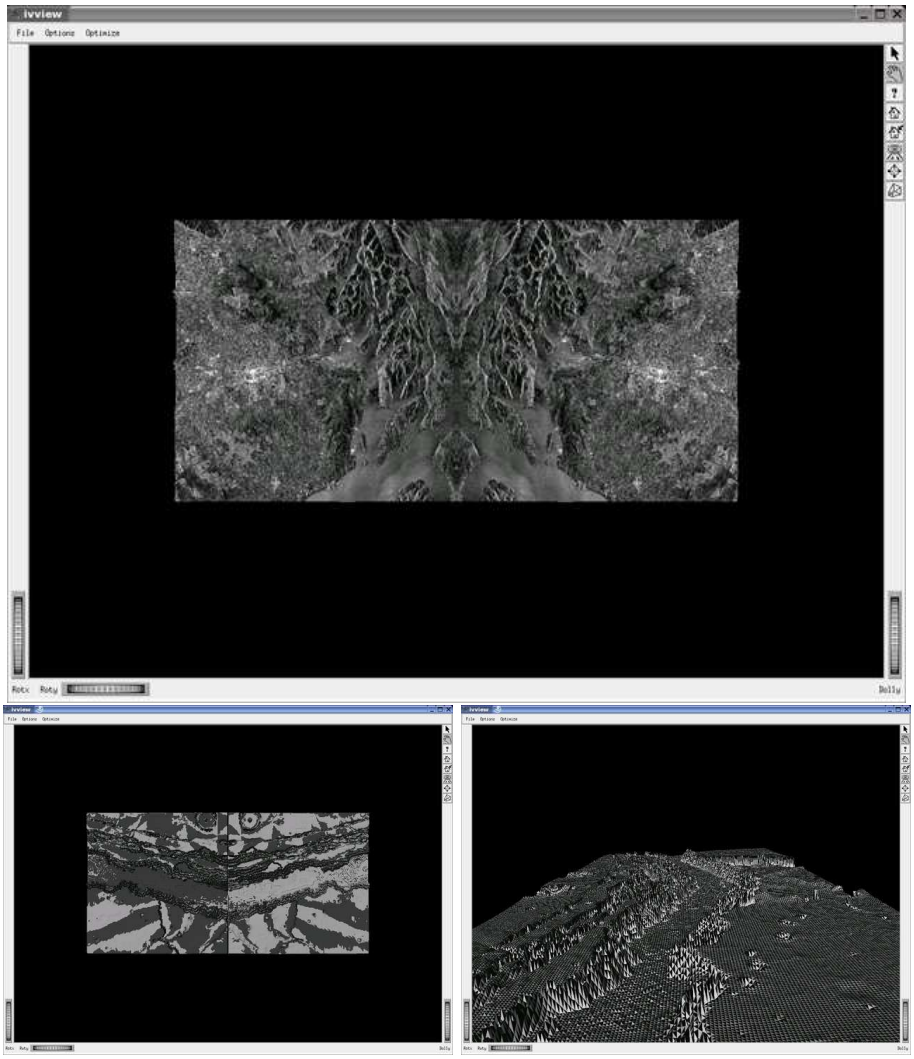


Figure 6.20: Virtual 3D terrain with 3D MOLA data textured with an aerial image of Scotland. The first image shows a top view of the terrain. Because the terrain was rather small, we mirrored it around an edge to obtain a terrain twice the size, hence the symmetric look of it. The second image is the same view without texturing and the third image shows an arbitrary view of the terrain without texturing.

terrain are shown in Fig. 6.20. Both sources of data are publicly available on the Internet. The quality of these inputs satisfies our needs. The disadvantage of this approach is the fact that the texture features do not coincide with real 3D features in the terrain, which makes human interpretation difficult.

6.10.2 Running Simulated Test Cases

The ILP has a transparent interface to both real and simulated cameras. The only difference between the two modes is the way images are created. In real mode, this is of course the task of the camera(s). In simulated mode, this task is performed by the Simulated Image Acquisition. This component takes into account the current simulated position and orientation of the aerobot over the terrain, the camera parameters, illumination and atmospheric conditions and renders photo-realistic images of the terrain. It uses an open source raytracing program for this, called **PovRay** [19]. Some of the generated simulated images of the artificial terrain are shown in Fig. 6.21

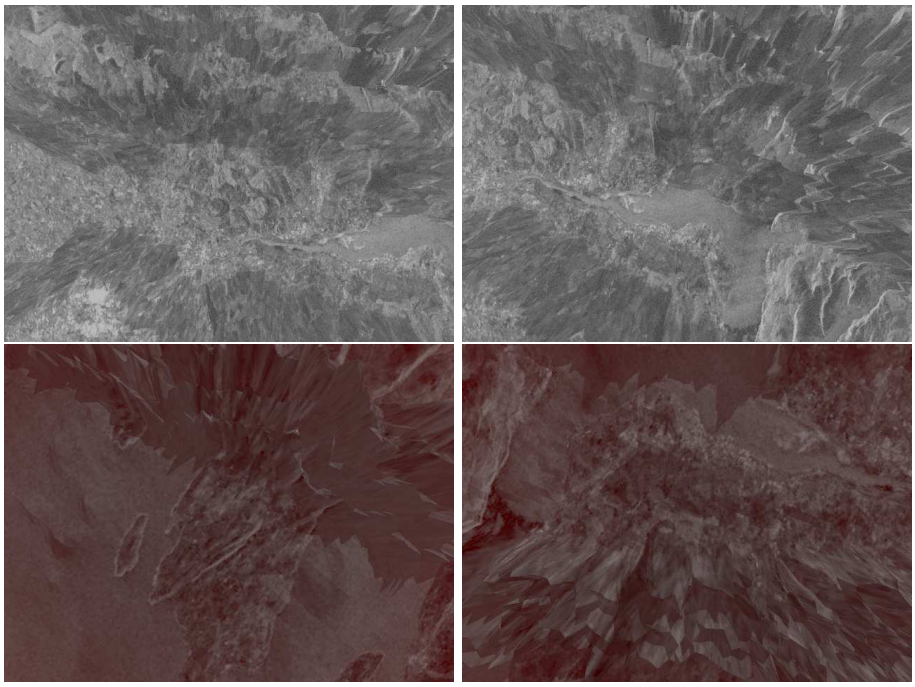


Figure 6.21: Some of the simulated images generated with PovRay. The atmospheric and illumination conditions of the first two were fixed at clear sky, while the two bottom images have been generated with some fog and a reddish spotlight.

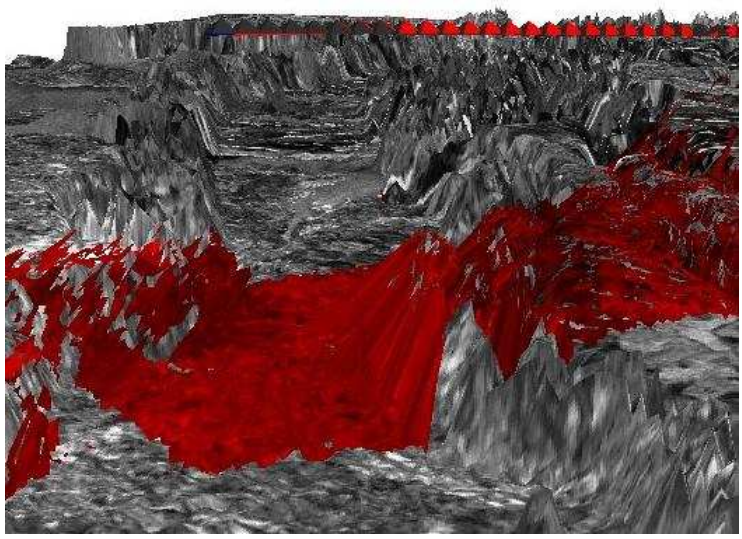


Figure 6.22: Output for experiment 3. The input terrain and trajectory are shown, overlayed by the resulting reconstruction (dark).

6.10.3 Results

Three Experiments

Three of the simulations that were fed to the aerobot system will be described. The trajectories and conditions for these tests were authored and the aerobot system recovered the camera calibration and dense reconstruction. Fig. 6.22 shows this dense reconstruction of the terrain (dark) together with the original virtual terrain. It becomes immediately clear that the resulting reconstruction is very close to the input terrain. Notice how the valley in the virtual terrain almost perfectly coincides with the same part of the reconstructed terrain. Some discrepancies can be seen on the left side of the valley. This is due to the original MOLA data which contained many spikes in this region which caused the texture to be heavily distorted.

Numerical Results and Error Statistics

In order to assess the quality of the resulting reconstructions, we have performed some statistical analysis of the errors between the results and the input. The camera positions and orientations of the reconstructions of all three sequences

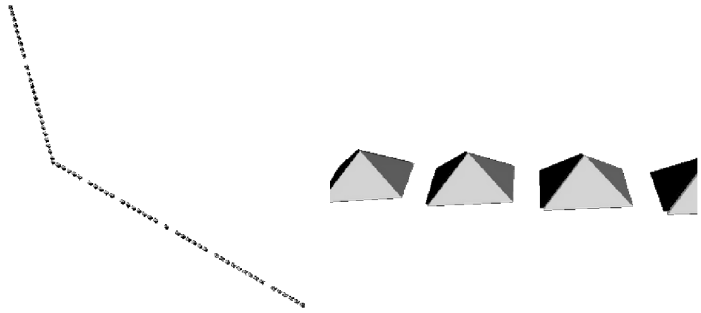


Figure 6.23: Top view and zoomed-in view of the input cameras (gray) and the reconstructed cameras (black) for the first experiment.

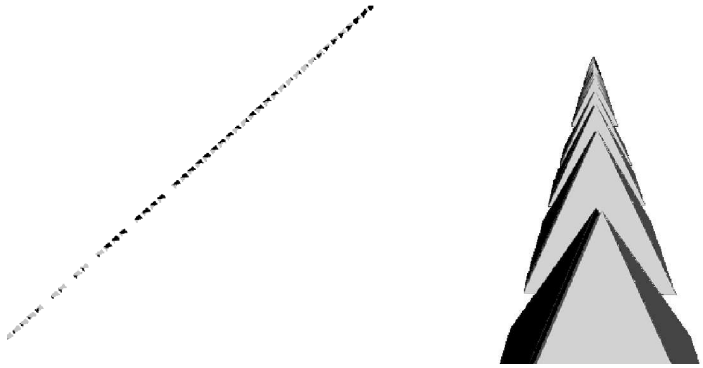


Figure 6.24: Top view and zoomed-in view of the input cameras (gray) and the reconstructed cameras (black) for the second experiment.

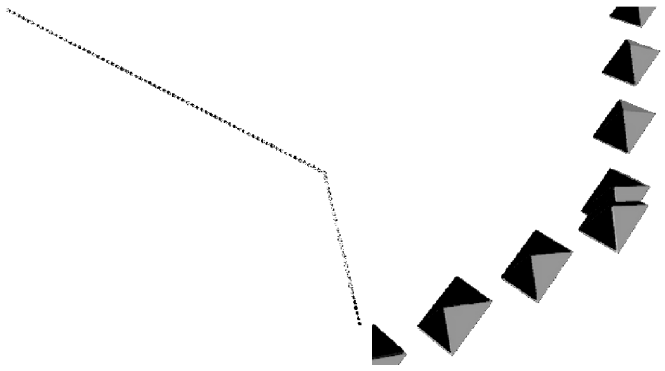


Figure 6.25: Top view and zoomed-in view of the input cameras (gray) and the reconstructed cameras (black) for the third experiment.

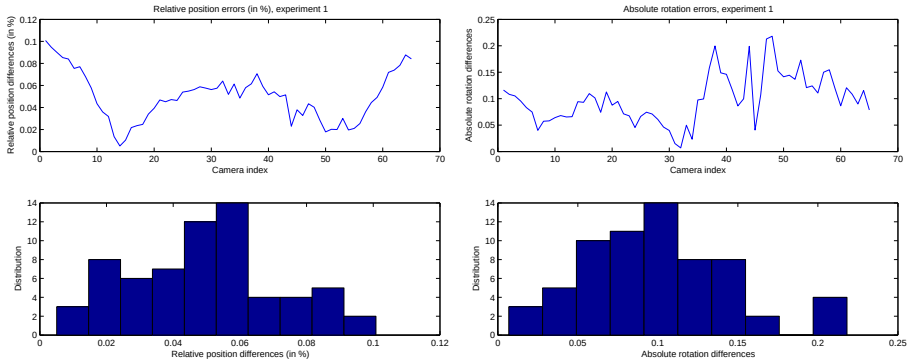


Figure 6.26: Error statistics for experiment 1. The left graph shows the relative positional errors and the right graph depicts the rotational errors.

were compared to their virtual input counterparts. Figures 6.23, 6.24 and 6.25 first show some screenshots of the 3D poses of both the original (gray) and reconstructed (black) cameras. It is already clear that the match is very good. Figures 6.26, 6.27 and 6.28 show graphs for the relative positional and rotational errors on the cameras for all three experiments. A side note must be made here. Since structure-and-motion techniques do not make use of any measurement in the scene, the resulting reconstruction is only valid up to a metric transformation (Euclidean up to scale). In other words, in order to compare both the reconstruction and the input, first the metric transformation between the two reconstructions must be found. This transformation consists of one translation, rotation and scale factor that is applied to all reconstructed cameras in order to bring them into the same coordinate system as the input data. The transformation is found by solving a non-linear optimization process that compares the reconstructed and input cameras and finds the metric transformation that brings the reconstructed data closest to the input data.

Let us consider Fig. 6.26. The graph on the top left shows the relative positional error for all 66 reconstructed cameras. This error was computed as the difference between the input and reconstructed position of each camera. This difference was then divided by the mean magnitude of all input positions to obtain a relative positional error. As one can see in the graph, all errors are smaller than 0.1%. The graph on the bottom left shows the same data but this time represented as a histogram. The graph on the top right depicts the rotational error between the input- and reconstructed cameras. This error was computed as the angle (in degrees) between both optical axes. As is shown in the figure, no rotational error is larger than 0.25 degrees. The graph on the bottom right finally shows the same data represented as a histogram. Table 6.1 shows some statistics on the relative positional errors for all experiments. Table 6.2 does the same for the rotational errors. Both tables shows the mean error and the standard deviation from it.

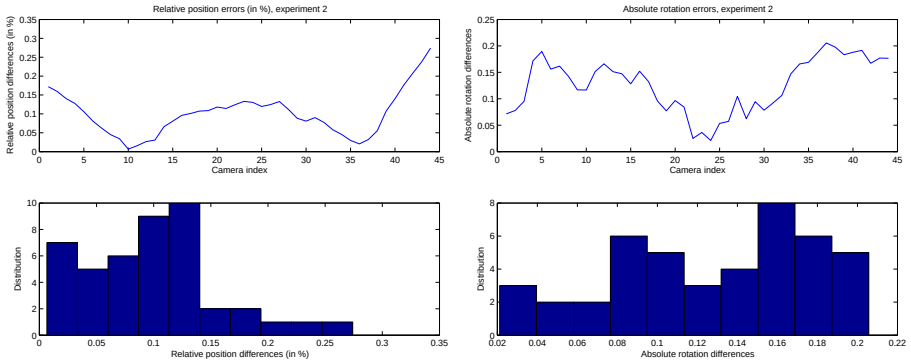


Figure 6.27: Error statistics for experiment 2. The left graph shows the relative positional errors and the right graph depicts the rotational errors.

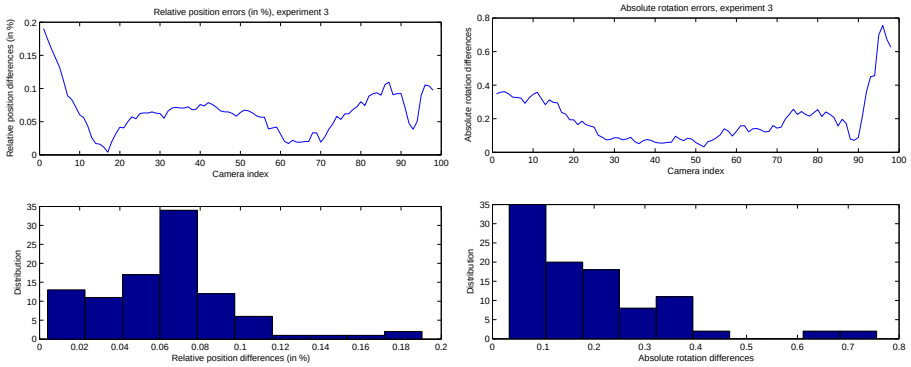


Figure 6.28: Error statistics for experiment 3. The left graph shows the relative positional errors and the right graph depicts the rotational errors.

Experiment	Experiment 1	Experiment 2	Experiment 3
Mean Error (%)	4.96e-02	9.99e-02	6.41e-02
Standard Deviation (%)	2.23e-02	5.82-02	3.35e-02

Table 6.1: Error statistics for the relative positional error on the cameras.

Experiment	Experiment 1	Experiment 2	Experiment 3
Mean Error (deg)	9.95e-02	1.27e-01	1.95e-01
Standard Deviation (deg)	4.58e-02	5.08e-02	1.45e-01

Table 6.2: Error statistics for the rotational error on the cameras.

6.11 Tests of ILP with Real Images

The aerobot system was not only tested on simulated (virtual) data but on real imagery as well. The planetary testbed at ESTEC, which was the testing ground for the PSPE project was used as testing environment in the Aerobot project as well. Two kinds of imagery served as input to the system. First a video recording that was made at the time of the Architectural Design Review of the project is discussed in section 6.11.1. Section 6.11.2 describes the aerobot hardware and on-board camera. The results generated from images of this camera are discussed in section 6.11.3.

6.11.1 Images from a Video

In order to successfully complete any engineering project with a sufficiently large research component, one should aim at testing the critical parts of the project early in the development pipeline. Since the hardware setup, described in section 6.11.2 became available only late in the project, acquiring data with this setup to test the computer vision algorithms while they were being developed was not feasible.

Recording

At the time of the Architectural Design Review, another solution was suggested by the project officer. The ESTEC planetary testbed is located in a room with a high ceiling under which a gantry can be moved. A video camera was placed in a cage suspended on the gantry. This cage was moved across the terrain, simulating the motion of an aerobot. The recorded video was digitally saved onto the computer in DV format and split in files of 1GB for convenience. Fig. 6.29 shows some of the frames of one of the DV files. The frames have been de-interlaced for processing and displaying purposes and have a resolution of 720x576 pixels. The video data that was recorded can be used as an alternative acquisition mode (the video-acquisition-mode, as it is called in the project) next to the simulated mode and the real mode with a flying balloon.

Thanks to this image data acquired early in the project, the team was able to spot problems for the computer vision algorithms at an early stage. To alleviate these problems the special techniques described in sections 6.8.2 were implemented.

Results

The camera calibration of one of the sweeps of the video-acquisition images is shown in Fig. 6.30. The reconstructed cameras are represented by small pyramids looking at the terrain. The feature points that are extracted in the images of the terrain and that are reconstructed in 3D are shown as dots. The results of the calibration are as expected. The cameras are clearly located approximately on one line (the direction of motion of the cage on the gantry) and



Figure 6.29: Some frames of the video acquisition of one of the sweeps of the camera over the planetary testbed in ESTEC

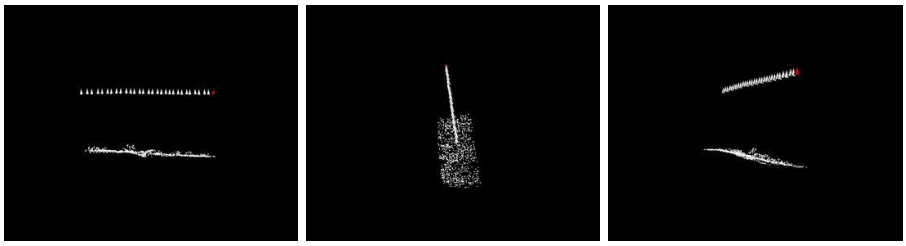


Figure 6.30: The results of the Structure and Motion process on one of the sweeps of the video-acquisition.

at the same height w.r.t. the terrain. Fig. 6.31 shows 3D dense reconstructions of the planetary testbed based on the video-acquisition data. The results are very convincing.

6.11.2 Aerobot Hardware

Although the core of the aerobot system consists of the ILP processing algorithms, a complete hardware platform has been constructed to test these algorithms on planetary testbeds. The hardware platform consists of

- an inflatable envelope, filled with Helium,
- a gondola with room for weights to control the buoyancy of the aerobot,
- a digital still camera equipped with Bluetooth technology for wireless communication,

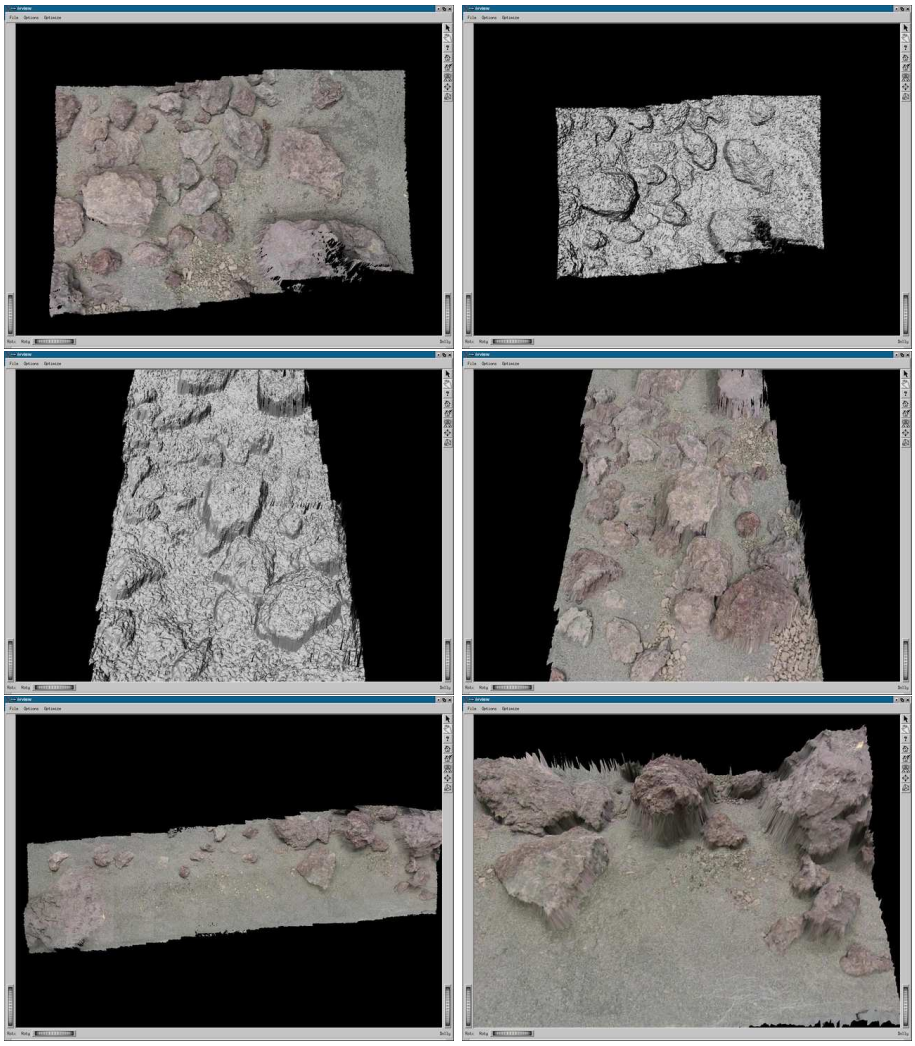


Figure 6.31: The results of the dense 3D reconstruction of the planetary testbed based on the video-acquisition data. Both textured and untextured views are shown to facilitate appreciation of the quality of the reconstruction.



Figure 6.32: The hardware setup of the aerobot system shows the inflatable envelope, filled with Helium, the gondola with camera, the fans and the guide-rope.

- a set of fans on both sides of the envelope to control attitude and speed of the aerobot,
- a guide-rope for more stability and safety.

All components of the hardware platform are clearly visible in Fig. 6.32. Fig. 6.33 shows a close-up of the gondola containing the wireless downward-looking camera.

6.11.3 Images from the Digital Camera

The aerobot system and its hardware system, described in section 6.11.2, was deployed during some preliminary tests a month before the actual review and during the final acceptance as well. At these occasions, the system made different runs over the terrain. More than 5 different reconstructions were created



Figure 6.33: Close-up of the gondola. The camera with wireless capabilities is looking downwards and is fastened in the space foreseen in the gondola.



Figure 6.34: Three consecutive pictures taken by the Sony camera on-board the aerobot system

by the system. A couple of consecutive pictures that were taken by the Sony camera are shown in Fig. 6.34. These pictures have the standard size of NTSC frames, i.e. 640x480 pixels.

Resulting reconstructions, including 3D and camera poses are shown in Fig. 6.35. The screenshots show combined 3D models generated from the DEMs of different images in one sequence. The ROI files described in section 6.9.2 are used as masks. The 3D reconstructions are created automatically and look very convincing. They show that the system is capable of performing as expected.

The strategy of computing and using an ROI for every DEM proved especially useful in a specific sequence. This is shown in Fig. 6.36. The images in the left column are the first and third image in the sequence. During tracking, no features were instantiated on top of the rock in the middle of the image. Therefore the search range of the stereo algorithm of section 6.8.3 is set too small to incorporate the top of the rock in the reconstruction. This is clear in the first ROI image on the top right where a large black hole appears in the place of the rock. Fortunately in the third image some features have been found on top of

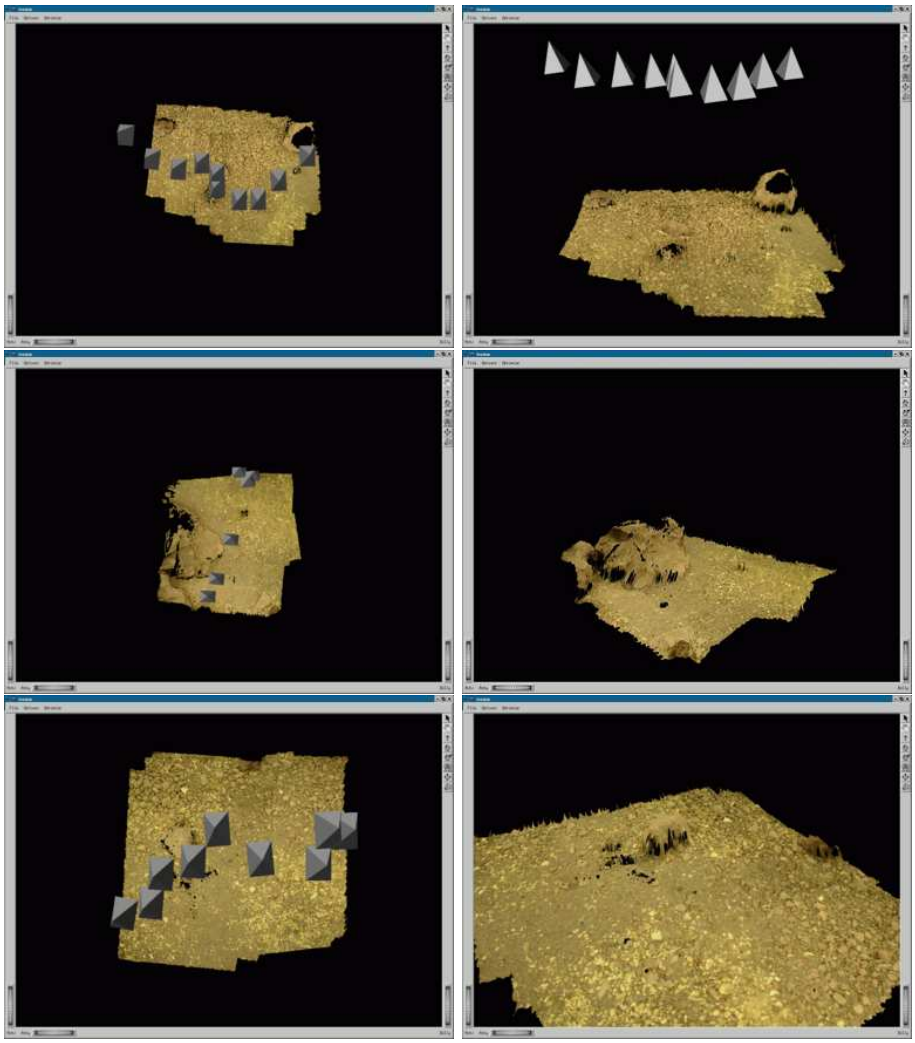


Figure 6.35: Resulting 3D reconstruction and camera calibration for some of the sequences taken by the real hardware setup

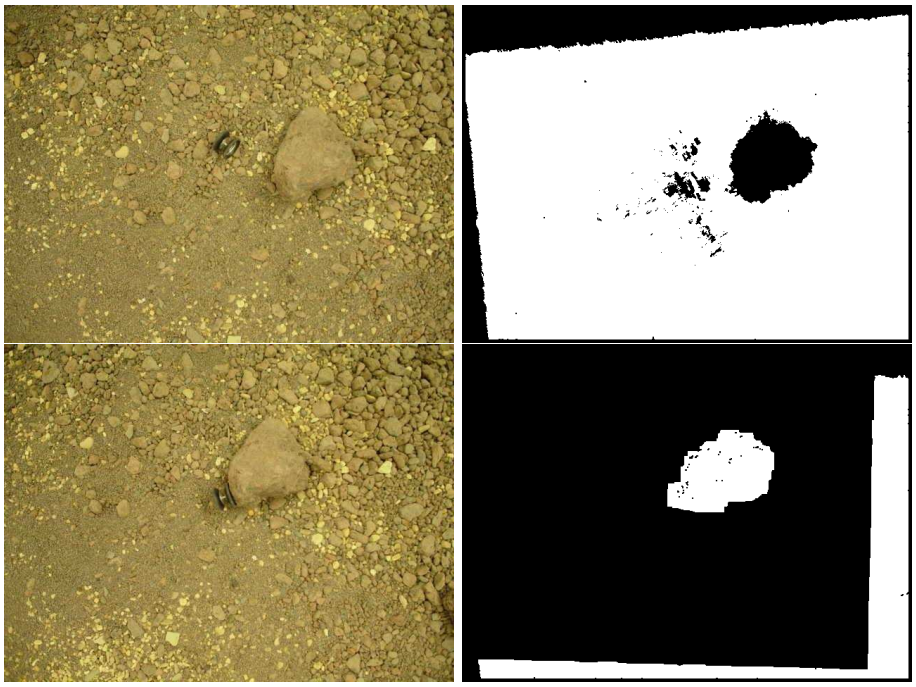


Figure 6.36: Left: first and third image in a real image sequence of the planetary testbed at ESTEC. Right: ROI images of the real images. The top of the rock in the middle of the image is not reconstructed in the first image and appears as a black hole in the ROI. In the third image, the rock is reconstructed for the first time and a white region appears in the ROI.

the rock. This means the search region for the stereo algorithm becomes larger and the top of the stone is reconstructed in 3D. Because the rock had not been reconstructed before, it is included in the ROI of the new image as can be seen in the bottom right image.

6.12 Conclusion

This chapter described two different ways to explore planets autonomously, making use of vision and more specifically of SaM algorithms. Both planetary exploration techniques were the topic of an ESA study performed by our lab as part of international consortia.

In the first ESA study a system was built containing a stereo head, and a small rover traversing the planetary terrain. Images from the stereo head, sent down to Earth, are the input for calibration and 3D reconstruction. Because markers could not be used, I designed and implemented specific calibration

algorithms for the stereo head, including its pan and tilt axes, making use of corresponding feature points between images of the unknown terrain. For the 3D reconstruction of the planetary terrain, I employed the dense stereo matching algorithms described in section 4.7, including some changes to the rectification to deal with the large disparities. The resulting disparity maps were then combined into one digital elevation map. The simulator component, written by myself with some help of a master thesis student under my supervision [43], showed the rover traversing the terrain. Both the simulated trajectory and the telemetry received from the system were displayed for visual confirmation.

The second ESA project studied the feasibility of using uncontrolled Aerobots for planetary exploration. An on-board camera records images of the terrain, which are immediately processed to create 3D DEMs. To this end I designed and implemented an intricate software system with several components, connected via a database. First, my overlap checker quickly decides which images are processed. Then a SaM algorithm, including my implementation of Nister's 5-point algorithm and my windowed bundle adjustment computes 3D points and cameras. When enough cameras are available, the database triggers my dense matching routine which computes DEMs and corresponding masks that are used for efficient compression. The system was tested first with simulated imagery, then with a digital video taken on a gantry over the terrain and finally with images from a camera on-board a real aerobot.

Chapter 7

SaM and Cultural Heritage

7.1 Introduction

Tangible cultural heritage has many enemies, both human and natural. Remember the bombing of our alma mater's library, the destruction of the Buddha statues in Afghanistan or the damage done by acid rain to Michelangelo's David in Firenze, forcing the city to move it indoors. Sometimes time itself is the culprit, especially for works of art, like paintings or sculptures in soft material, that were not created with preservation in mind.

The importance of safeguarding our cultural heritage was understood early-on. During the Middle Ages books and texts of the ancients were minutely copied, allowing us to study these millennia later. Every age employed the technologies it had access to for heritage preservation: manual copying of books and drawings of statues and buildings at first; bookprinting and photography later. With the rise of the digital age we live in, new techniques became available for preservation. Especially laserscanning was and is a popular means (be it very expensive) to acquire accurate 3D models of existing objects, buildings and scenes.

In this chapter we will discuss two applications that employ the techniques explained in previous chapters for cultural heritage preservation. For the first application we implemented an image processing pipeline to extend an existing QT-VR movie of an archaeological site with a layer of its current situation. Section 7.2 explains this application in more detail. The second application is the culmination of all passive 3D reconstruction algorithms in this work. We have built an easy-to-use webservice that allows people in the cultural heritage sector to create 3D models of objects, buildings or sites by simply uploading a set of images to a server and retrieving the result via the Internet. Section 7.3 explains this webservice thoroughly.

7.2 Timeline into the present

7.2.1 Combining Image and Model Based Rendering of an Archaeological Site

Preservation of cultural heritage through digital technology can only be successful if people can experience sites and reconstructions in an intuitive, yet convincingly real way. It also requires ways to get people interested in the past. A possible strategy to do so is highlighting the link of the past with the present. For example, when visitors to an archaeological site can see the link between the archaeological remains and the historical buildings that were standing on that spot, they will be more inclined to explore the story of that site in more detail.

We will extend this methodology to landscapes, and prove that it is possible to show the evolution of a landscape until the present day, without making a virtual model of that present day situation. We will discuss a way to create interactive applications to explore the evolution of a landscape and register a visualization of the current landscape with virtual reconstructions. In this way, we obtain an optimal combination of real and virtual images that shows the evolution and history of the site. The following sections outline the pipeline from data acquisition by oblique aerial photography, over camera calibration and registration of the virtual with the real landscape, to image-based rendering techniques to create the appropriate views. The work we will discuss was done in the scope of the IST project INVIEW. The idea came from Daniël Pletinckx of Visual Dimension who worked for the Ename 974 project at the time and was involved in the project.

Many applications make use of the QuickTime technology provided by Apple, especially to create interactive applications through the QTVR suite [46]. One element of this technology are QTVR object movies that allow interactive access to a two-dimensional matrix of images. One of the ways to use this is to show an object from several different angles. A more sophisticated use of the QTVR object technology is to build interactive 4-dimensional objects, by rotating the object through a horizontal cursor movement, and evolving the object through time through a vertical cursor movement. Since 1999, the Provincial Archaeological Museum of Ename has used 4-dimensional QTVR objects to show the evolution of the entire village over the last ten centuries [69]. In this interactive TimeLine application, virtual models of 12 consecutive periods are visualized. In the TimeLine application a virtual camera spins at a fixed height around a fixed point, yielding 36 regularly spaced views on the geo-referenced virtual model, and this for all 12 historical periods. In this way, the columns of the 36×12 matrix of views show the evolution of the landscape from 1020 to 1780 AD from a specific point of view, with the columns evenly spaced on a predefined circle with a height of 150m and a diameter of 260m. The circle where the virtual camera was positioned is shown in Fig. 7.1. This figure also shows the Ename site.

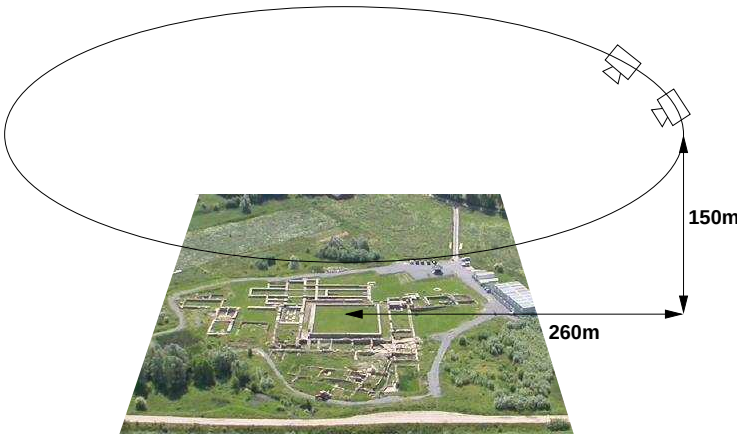


Figure 7.1: The setup of the virtual cameras. They are distributed every 10 degrees on a circle with a radius of 260 meter at 150 meter altitude.

To extend this matrix to the present, as illustrated in Fig. 7.2, we need to take images from exactly the same 36 viewpoints, and this at a height of 150m. This is virtually impossible with straightforward photography however, and we propose an alternative way, based on *image-based rendering*. The result is rewarding. In more than one way, this most recent, modern row of the TimeLine matrix (see the 4th row in Fig. 7.2) is the most intriguing one, as it links the present – experienced by the visitor – to the past. Every element in the current landscape gets a meaning, a story, a reason why it is the way it is. By linking the past to the present, a general audience can be addressed by telling the story of how a modern site and landscape came to be the way it is.

A processing pipeline has been devised to create modern views that are aligned with those already in the TimeLine matrix. The pipeline is shown in Fig. 7.3.

First step - In essence, this pipeline produces interpolated views, starting from a series of photographs of the site. Therefore, a first step consists of acquiring the latter. Since the virtual TimeLine images show the archaeological site from above, a helicopter was used to shoot the photographs. Section 7.2.2 explains this in more detail. The use of a helicopter is only part of the solution. Even when combined with state-of-the art technology such as DGPS or steadycams, it is highly unlikely that one would manage to let any of the images coincide perfectly with one of the 36 target viewpoints as used in the TimeLine views. We have used image-based rendering (IBR) for that purpose. IBR takes a series of close photographs taken from known directions and a 3D model of the scene as input, and then creates a kind of interpolated views, taking the 3D geometry into account.

Second step - Typically IBR is applied based on images taken from many and



Figure 7.2: These 16 images are a small part of the 36×13 TimeLine image matrix, upgraded with present day images (last row). Each column represents one of 36 different viewpoints (in this case arranged along a circular path), and the 12 first rows represent different eras from the past. The last row contains images from our own time, for the same viewpoints and in the form of photo-realistic images, so that visitors can directly relate to them. In this cutout of the complete TimeLine matrix only 4 eras are shown, namely 1020, 1250, 1780 and 2004.

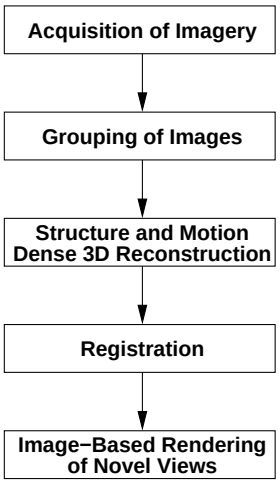


Figure 7.3: Five steps in the pipeline will realize our final goal of making a QTVR object movie of the real site, registered with the virtual views of the existing TimeLine

widely spread directions, so as to be able to generate synthetic views from all around an object. Here, we are only interested in a very limited set of such synthetic views. Hence, we can actually split the task into a series of smaller IBR problems. For each target viewpoint images are selected that were taken in its vicinity. This image grouping step is explained in section 7.2.3.

Third step - In the previous step, the helicopter images have been subdivided into groups, where each group is to be used as the input to an IBR process for some of the target viewpoints. On average four target viewpoints could be rendered per selected group. For the images within each group, the camera parameters are extracted (calibration) and a 3D reconstruction is computed from their point of view, exploiting all the images in the group. This SaM process is the subject of section 7.2.4.

Fourth step - In order to determine the actual target viewpoints in terms of the IBR images that have to be formed, the new images must be registered to the virtual target images. Since only few structures visible in the virtual TimeLine views are still easily recognizable in the real images of today, we opted for a manual process in which an archaeologist indicates points on the virtual reconstructions of the different eras and on the ortho-photo of the actual site. This process is described in section 7.2.5.

Fifth step - Finally, once data from all previous steps have become available, newly rendered images can be produced using the IBR approach explained in section 7.2.6.



Figure 7.4: The helicopter from where the photographs were shot.

7.2.2 Acquisition of Imagery

The Ename TimeLine was created with views on a set of geo-referenced virtual models of the complete village. If we want the real component of the movie to seamlessly match the virtual images, we need images of the real site that are recorded from the same positions. For every time instance, there are 36 images, evenly distributed over a circle with a radius of 260 meter. The virtual cameras are positioned 150 meter high and looking towards the center of the circle on the ground as shown in Fig. 7.1. The easiest way to record images in the real world, taken approximately on this same circle, is by using a helicopter.

On a day with perfect weather conditions in June 2004, the helicopter (Fig. 7.4) arrived at the site. The helicopter pilot was asked to fly the white circle (with a radius of 260 m) shown on the ortho-photo of Fig. 7.5 at an altitude of 150 meter.

An additional navigator made sure the pilot could follow these requirements safely. While the helicopter followed this path as precisely as possible, a photographer – as a third crew member – used a digital photo camera to record the scene, focusing his camera all the time on the center of the circle, somewhere in the middle of the archaeological site. As the helicopter needs a certain speed to fly the circle and the digital camera needs a certain time to store the high resolution image, an average of 15 images were taken per tour, so the consecutive images were quite far apart. Therefore, the helicopter flew the circle several times at slightly different altitudes and images were taken constantly. To ensure that all necessary elements of the site were visible in the recorded images, the digital camera had a larger opening angle than that of the virtual camera of the QTVR movie. Fig. 7.6 shows some of the 130 digital images that were acquired during the helicopter flight.

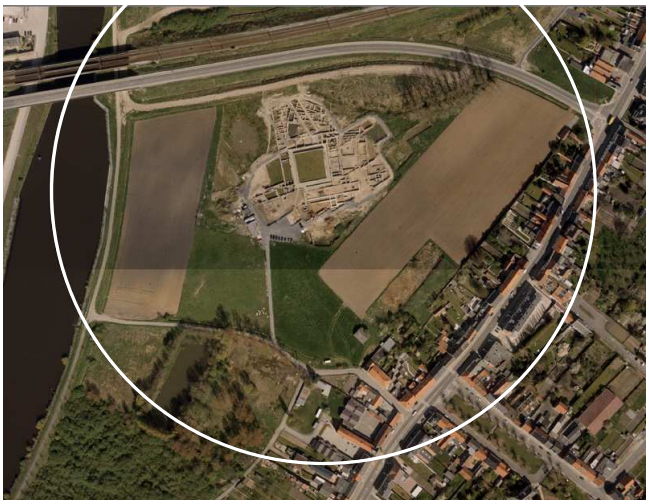


Figure 7.5: The navigator of the helicopter was asked to follow the flight path shown in this figure.

7.2.3 Grouping of Images

The images that were acquired as described in section 7.2.2 show the site from different angles. Consecutive images can be far apart and are therefore not always suited for Structure and Motion (SaM) algorithms. Since the images were recorded during multiple fly-bys, the same content reappears regularly. As explained in section 7.2.1, it is not our goal to calibrate the cameras of all images or of the entire circle in one SaM process; rather several, smaller IBR problems are solved. We want to combine different images taken from viewpoints that are quite close to a target viewpoint and apply SaM to this group of images. The camera range spanned by the selected group of images is large enough to solve the IBR problem for the targeted viewpoint and some of its neighbors. It turned out that an average number of four viewpoints could be rendered for every group of images. The process of selecting the group of images for a target viewpoint was therefore repeated nine times. The major task in the grouping process consists in detecting images that were taken close to each other. Manual selection in the 130 images is, while possible, a very tedious task. A simple but effective algorithm was implemented which does the job.

The algorithm boils down to a comparison of all pairs of images, using a similar technique as described in section 6.8.1. The center part of one image is selected. An area with the same size is extracted from the other image and slides over this image. For every position, both windows are compared using



Figure 7.6: Some of the 130 images taken during the helicopter flight.

Normalized Cross Correlation. The highest correlation gives us a measure of how well these images resemble each other. The images are subsampled to decrease processing time. Fig. 7.7 shows the resulting, symmetric matrix of these values. White points represent a high correlation, black points denote no correlation between the two images given by the x and y coordinate in the matrix. The diagonal is of course perfectly white. One can clearly see the different passes of the helicopter appearing as white lines, connecting consecutive images.

The selection process becomes a much easier task now. If we want to reconstruct the cameras and 3D scene from a certain viewpoint, all we need to do is select one image that is taken near the viewpoint we want. The matrix of Fig. 7.7 then lists the best candidates to match this image with.

7.2.4 Structure and Motion and Dense 3D

The ultimate goal of the pipeline described in section 7.2.1 is to generate new images from specific viewpoints. We have chosen an image-based rendering (IBR) approach to do so. As will be explained in section 7.2.6, this approach produces a kind of high quality interpolation between a set of images recorded in the vicinity of the required viewpoint. In order to do so, the IBR technique needs the intrinsic and extrinsic camera parameters of the recorded images, as

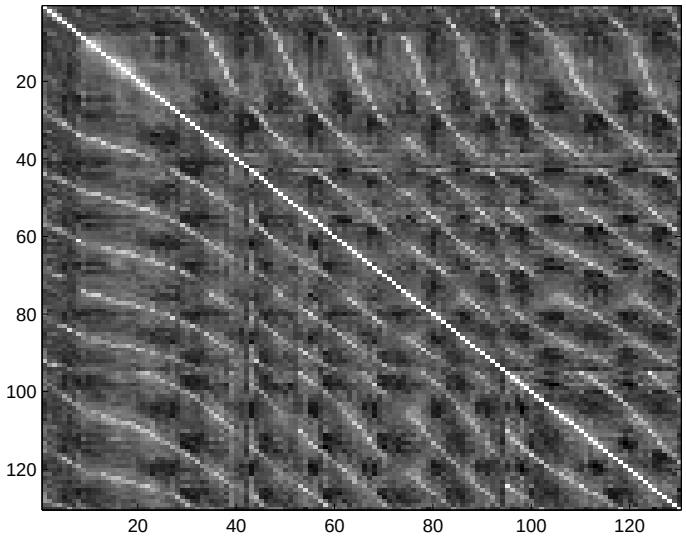


Figure 7.7: Matrix with NCC values. White points represent a high correlation, black points mean that there is no correlation between the two images given by the x and y coordinate in the matrix. This matrix can be used to determine a set of images that were taken close to a given image.

well as a depth map for every image. Obviously the SaM techniques explained in the previous chapters were employed to obtain these results.

Several factors of this specific case will cause standard SaM techniques to fail. For instance, the archaeological site of Ename is not exactly planar but when viewed from a distance of about 300 meter there is hardly any 3D information outside the ground plane, which will certainly cause problems. That is why we were forced once more to make use of the 5-point essential matrix computation scheme and pose-estimation algorithms, as we did in the Aerobots project (see section 6.8.2). The internal camera calibration of the camera was recovered from an extra image sequence of a 3D object we recorded afterwards with the same camera, using the same settings.

For the automatic structure and motion algorithm to succeed, the input images should not differ too much from each other. Techniques exist to deal with wide-baseline cases but these are not applicable here since the result of the structure and motion pipeline will be used by the image-based rendering process of section 7.2.6 which needs images with viewpoints that are close. Fig. 7.8 shows four of seven images in a group used for one of the 9 structure and motion computations mentioned in section 7.2.3. The images are selected by choosing one manually and retrieving the six best matching views from the matrix in



Figure 7.8: Four of the seven images that served as input for one structure and motion process. The images were taken during different passes of the helicopter and are selected automatically using the matrix in Fig. 7.7.

Fig. 7.7.

The resulting point reconstruction and camera calibration can be seen in Fig. 7.9. A top and side view are shown. It is clear that the structure of the scene is very close to planar. Fig. 7.10 shows some views of the dense 3D model that was reconstructed from the images. The dense 3D is far from perfect, due to large homogeneous areas like the fields or due to the large distance to the cameras, like the houses in the background. The quality might not be good enough for ordinary, geometry-based rendering but is sufficient for image-based rendering.

7.2.5 Registration

At this stage in the pipeline of section 7.2.1 we have available separate sets of images – on average one set per four target views – with corresponding 3D reconstructions and camera calibrations. We want to employ these results to create synthetic viewpoints that coincide with the corresponding target viewpoints. Unfortunately, every 3D reconstruction has been computed in its own metric frame. So far, each reconstruction is related to the site coordinate system of

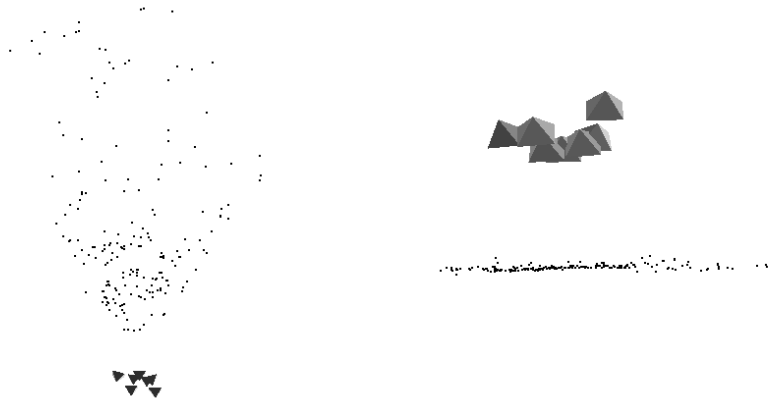


Figure 7.9: The camera calibration of the images and a set of 3D points, representing the scene, retrieved by our structure and motion algorithm. Both a top and a side view are shown. The planar nature of the scene can clearly be recognized.

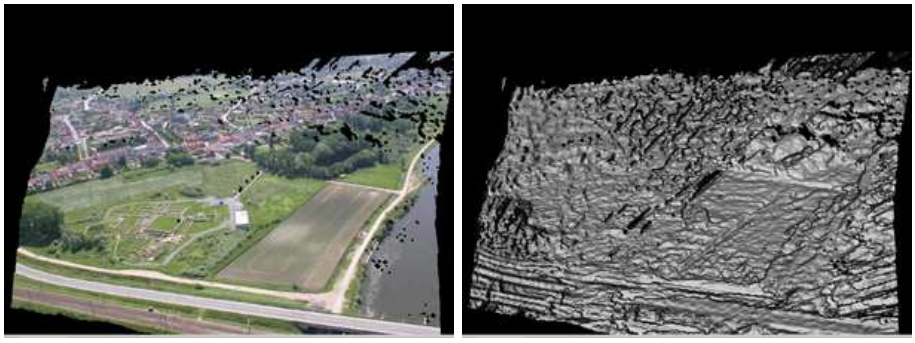


Figure 7.10: Dense reconstruction of the scene. The quality is not perfect, certainly in homogeneous areas or in areas that are far away.



Figure 7.11: Pose estimation with manual interaction. Corresponding points (in red) are indicated both on the ortho-photo (left) and on the image taken from the helicopter (right).

Fig. 7.1 via an unknown metric transformation consisting of a rotation, a translation and a scale factor. If we want to use the reconstruction for image-based rendering, we need to compute these metric transformations. This process is called registration. A possible strategy to find the metric transformation described above consists of following two consecutive steps. First we bring one camera of the structure and motion reconstruction into the target frame, effectively computing the rotational and translational part of the transformation. Then the scale factor between the two frames is computed from one or more indicated distances. Step one boils down to a pose estimation process of one camera that has been computed in the SaM-frame in the target frame. This can be done if at least 3 3D points in the target frame and their corresponding 2D points in the image are known [34]. If more than three correspondences are available, a better approach is to compute an initial solution and then minimize the total reprojection error of all available points with a non-linear optimization step. For the 3D coordinates, we employed a geo-referenced ortho-photo of the archaeological site. A member of the Ename archaeological team was asked to indicate points on this ortho-photo and their corresponding points in the image as well. Fig. 7.11 shows such a process. In the left image the relevant part of the ortho-photo is shown. The right part shows the image taken from the helicopter. The labeled indicated points are shown superimposed.

Step two still needs to determine the scale factor between the two frames. If two 3D points are known in both coordinate systems, the ratio of both distances is the scale factor we are looking for. One of the points could be the center of the camera that has been computed with pose-estimation, since we already know this in both frames. We just need one more 3D point with known coordinates in both frames. The scale factor can be computed from any of the indicated points. The coordinates in the target frame are known from the ortho-photo and the position in the SaM-frame can be found through a lookup in the corresponding

depth map that was computed in section 7.2.4.

7.2.6 Image-Based Rendering

Image-Based Rendering generates novel views by interpolating information from images that are close to the requested view. The advantage over traditional 3D scene model-based rendering is that an exact geometrical description of the scene is not necessary in this approach [54] although approximate geometrical information can be used to improve the results [33, 12]. In our lab we have developed an image-based rendering pipeline [41, 48], and have recently extended it further to incorporate elements from the work of different authors, such as Buehler [10], Pajarola [67] and Evers-Senne [20]. This extended pipeline is used to create each of the 36 target views of the present site (i.e. the last row of the TimeLine matrix), each time exploiting a set of relevant, grouped images (step 2). The image-based rendering step is performed based on the work of Geert Willems [105] of our lab.

In contrast to the first Lumigraphs, the pipeline developed in our lab can deal with an unstructured set of images. All source images have their associated camera pose estimates, which we have computed using Structure and Motion and which were registered with the target frame. The algorithm behaves like an extension of view-dependent texture mapping and uses depth maps generated through SaM as an approximation of the 3D geometry [48].

Rather than using a fixed 3D geometry, Pajarola et al. [67] propose a technique to approximate the 3D information in a non-uniform way. For every depth map a restricted quadtree is built. This quadtree holds the hierarchical representation of the depth map and has the additional property that, no matter which hierarchical level is chosen, the topology of the scene is preserved. This quadtree can be constructed beforehand in an off-line step, reducing the on-line processing time significantly.

The key property of image-based rendering techniques is the combination of different images to render a new virtual view. Not every image that is used for the rendering has the same impact in all areas of the new image. It is desired that the importance of each image depends on the quality of its camera, its 3D reconstruction, etc. evaluated at every pixel of the virtual view. To this end a *blendfield* is constructed. This blendfield describes the relative importance of each camera for every pixel in the virtual view. In order to generate the resulting image, each selected camera projects its own image onto its own quadtree and is rendered from the requested viewpoint. All renderings are combined into one image using the weights in the blendfield.

Although the 36 necessary views can be rendered off-line, one can easily think of other applications where interactivity is important. That is why we have implemented as many operations as possible on the GPU. Fig. 7.12 shows a virtual view, reconstructed from three selected cameras with its corresponding blendfield.



Figure 7.12: A virtual image, rendered from three helicopter images (left). The corresponding blendfield (right) shows the relative importance of each camera at each pixel. For visualization of this blendfield, a grayscale value is assigned to each of the three images used to generate this virtual view: a bright pixel for example denotes a pixel where the influence of the first image in the final rendered image is high. For a pixel with a midtone grayvalue, the second image is more important and for a dark image the third image has the highest influence.

7.2.7 Refining cameras

During processing it became apparent that the calibration of the cameras sometimes needed improvement. For the Structure and Motion process to deliver accurate results feature points need to be tracked throughout the entire sequence of images. The longer the feature tracks, the better the result will be. Unfortunately in this specific case images contain a lot of homogeneous areas, like the green and brown fields and the river. These areas are a nightmare for feature detectors. In order to improve the calibration result a refinement process has been implemented that updates the camera calibration. One camera and its corresponding 3D information is selected as the reference and a virtual viewpoint is chosen. All other images are projected sequentially on the reference 3D scene through their corresponding camera. This result is projected onto the virtual view. The difference between this rendering and the one obtained from the reference view is computed. A Levenberg-Marquardt algorithm aims at minimizing the sum of squared differences between these views by allowing each camera to adjust its position and viewing direction. The necessary derivatives cannot be expressed in a formula and have to be estimated at every step. Fig. 7.13 shows the difference before and after the optimization for one camera. The resulting camera calibration is more accurate than the original because all edges and homogeneous areas have their impact. All projection and difference operations, including the computation of the derivatives, are performed on the GPU, speeding up this minimization process significantly.

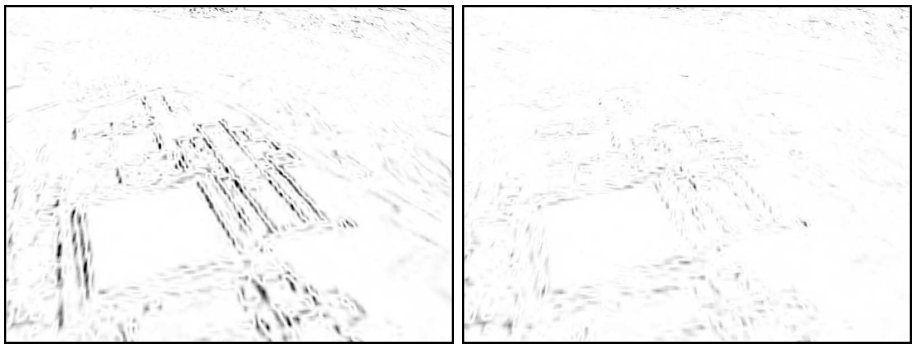


Figure 7.13: The difference image between the reference and one other camera before (left) and after (right) minimization. Black pixels indicate a high intensity difference, white pixels a low difference. The overall intensity error is reduced by a factor of 2.4.

7.2.8 Results

The pipeline of section 7.2.1 was executed for 9 groups of helicopter images. The reconstructions of these sets allowed us to generate the necessary 36 views for the QTVR object movie, yielding an average of four newly generated images per set of helicopter images. Fig. 7.14 shows four of these new views, evenly distributed along the circle.

The new images fit very well with the virtual images of the TimeLine. This can be evaluated by comparing the position of the remaining foundations of the buildings with the walls of the virtual images. Fig. 7.15 shows the last two time periods in the updated TimeLine. The first image is a virtual reconstruction, representing the site as it was around 1780. The second is a photograph of today's situation, generated with the IBR pipeline. Fig. 7.16 shows two viewpoints where the virtual reconstruction of 1780 has been merged with the newly generated images. The buildings of the archaeological site clearly correspond to the foundations in the real images. For the area in the background, including the church (visible in 7.15), the fit is a few pixels off. This can be explained by errors in the registration step. The reference data in this step consisted of an ortho-photo of the archaeological site. This data corresponds to an area in the helicopter images of about 25 percent of the total image. The registration is therefore good for this area but the error in other image parts increases rapidly. The resulting QT-VR movie is on display in the Provincial museum and the archaeological site at Ename where it proved quite popular with the visitors. It was recently shown on "Vlaanderen, vakantieland", a television show by the VRT, the public broadcaster of the Flemish community in Belgium.



Figure 7.14: Four of the newly generated images of the archaeological site

7.3 3D Webservice

The rise of new information technologies has led to the development of many applications in the cultural heritage area. By now, it is commonly accepted that ICT applications in general and 3D reconstruction and rendering technologies in particular carry large potential for this field. Especially the power to acquire accurate and realistic 3D representations of objects and scenes is highly valued, for it offers the possibility to preserve cultural heritage for generations to come. The European Research Network of Excellence EPOCH is a network of about a hundred European cultural institutions joining their efforts to improve the quality and effectiveness of the use of Information and Communication Technology for Cultural Heritage. A research agenda has been composed which clearly indicates needs that are present in this area and how they relate to existing technologies. One of the elements in this agenda is versatile 3D scanning. Scanning technologies should become cheaper and more flexible.

As a core EPOCH partner, our lab is very active in the network and decided to make a part of its expertise available to the cultural heritage professionals. The maturity of the 3D reconstruction techniques gave us the confidence that we could set up a fully automatic reconstruction service that they could easily make use of. The EPOCH network provided the opportunity to collaborate with the Visual Computing Lab of CNR-ISTI (Italy) which has also been active for a long time in the field of 3D reconstruction, albeit in its own specific sub-



Figure 7.15: The newly rendered image for position 10 (bottom) and its virtual counterpart (top). The fit is very good in the area of the archaeological site. The areas near the border of the image, like the church, have a larger error.



Figure 7.16: Two viewpoints of the time period around 1780 have been merged with the corresponding image-based rendered images of the current situation. The fit is remarkably good.

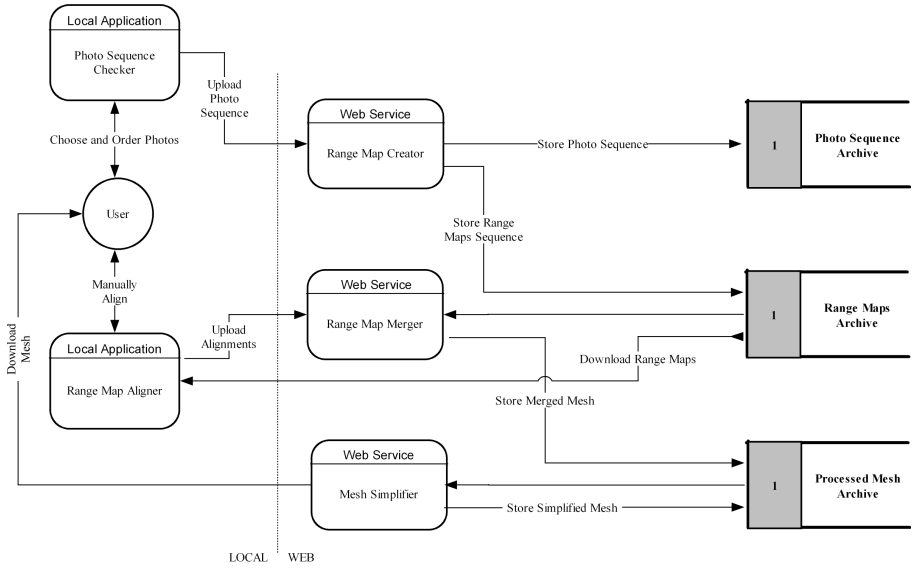


Figure 7.17: The main architecture of the combined 3D system: a local application prepares and checks the photos collected by a user and uploads them to a web service; the web service creates and stores range maps; these range maps are then aligned by the user and finally merged, simplified and exported to various formats

area of active reconstruction with laser scanning. Both labs decided to combine strengths and set-up a low-cost 3D reconstruction pipeline to be used in the cultural heritage field. The idea is that only a digital photo-camera and an Internet connection are necessary prerequisites for a user to reconstruct scenes in 3D. Fig. 7.17 shows a schematic overview of the complete pipeline. It consists of three services that are performed on servers, accessible via the Internet and fed by data from the user and two local applications. Images of the object or scene to be reconstructed are first checked for quality, then uploaded to the first web service which computes dense range maps. A second web service is capable of merging these range maps. The user can also align range maps from different image sequences locally and merge them. The third web service is in charge of mesh creation and simplification. From a development point of view, the first (reconstruction) service has entirely been developed at VISICS. The expertise of CNR-ISTI comes into play in the second and third service.

The remainder of this section describes the first web-service of Fig. 7.17. The outline is as follows. Section 7.3.1 gives an overview of the client-server system that allows users to upload images to a server which computes 3D information from these images. Section 7.3.2 describes the automatic reconstruction pipeline running on the server. Sections 7.3.3, 7.3.4, 7.3.5, 7.3.6 give more detailed information on several core aspects of this pipeline. Section 7.3.7 shows some results we obtained from uploaded images.

7.3.1 System Overview

Fig. 7.18 shows a schematic overview of the client-server setup of the 3D web-service. The client- (or user-) part is located at the top. The server side is at the bottom. On his PC, the user can run two programs, the *upload tool* and the *modelviewer tool*, indicated **A** and **B** respectively. These tools are explained in more detail in paragraphs 7.3.1, 7.3.1. In the upload tool, images can be imported that were taken with a digital camera. Once authenticated, the user can transfer these images (indicated **C**) over the Internet to the EPOCH server at ESAT. There a fully automatic parallel process is launched which computes dense 3D information from the uploaded images. The parallel processes are run on a cluster of Linux PC's (letter **D**). When the server has finished processing, the user is notified by email and the results can be downloaded from the ESAT server by FTP. These results consist of dense depth maps for every image and the camera calibration (letter **E**). The modelviewer tool allows the user to inspect the results. Every reconstructed depth map in the image set can be shown in 3D, unwanted areas can be masked out and the meshes can be saved in a variety of formats.

Upload Tool

The first tool a user of the 3D webservice encounters is the upload tool. This is the program that uploads images to the server via the Internet. Fig. 7.19 shows

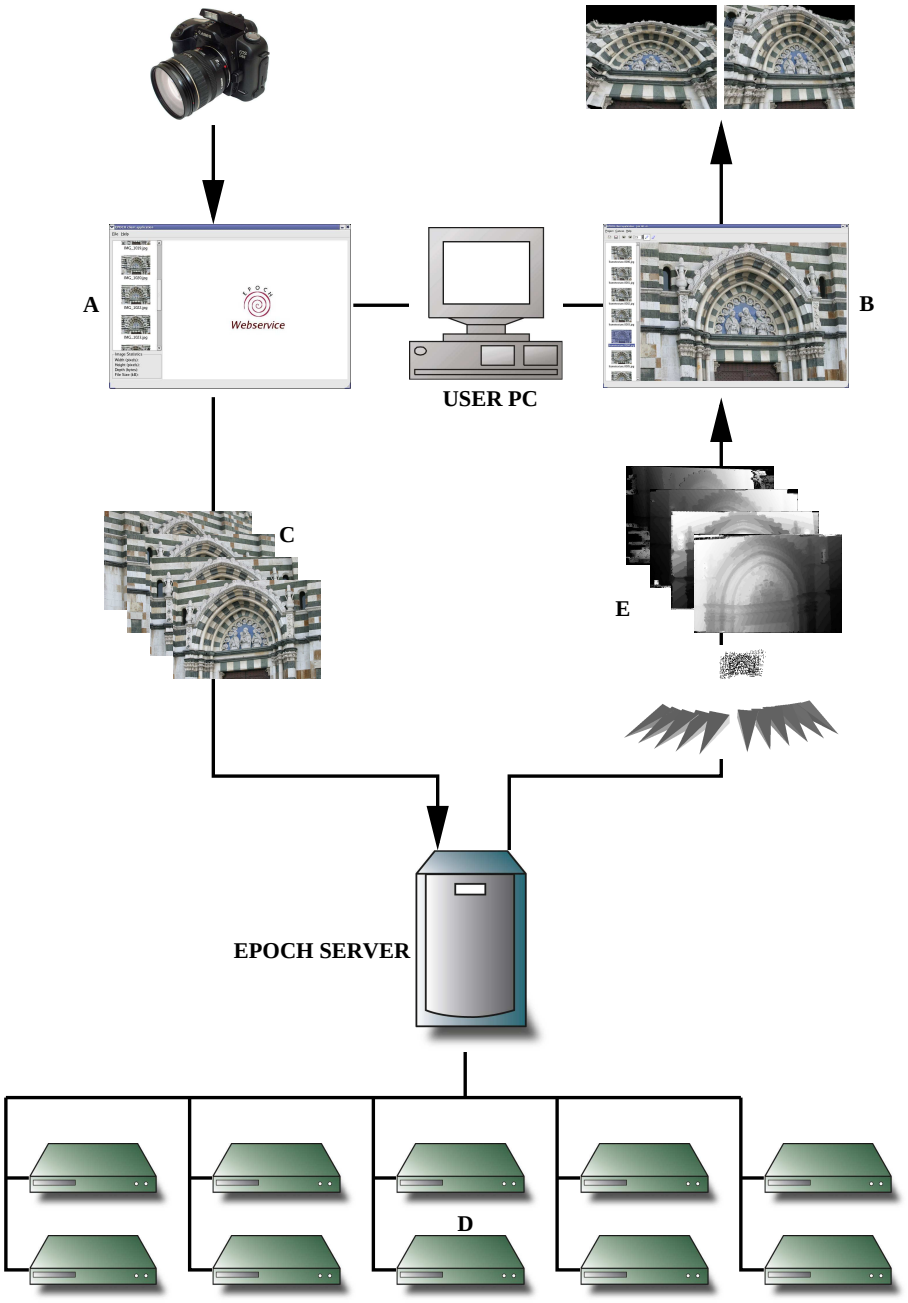


Figure 7.18: Schematic overview of the client-server setup. Images (C) are uploaded from the upload tool (A) on the user side to the server. There they are processed on a PC cluster (D). The results (E) can be downloaded via ftp and visualized on the user PC with the modelviewer tool (B).



Figure 7.19: Upload tool. Thumbnails of the images to be uploaded are shown as well as some information. When satisfied, the user uploads the images to the reconstruction server.

a screenshot of the GUI of this tool. First, the user selects the images which he wants to upload. Thumbnails of these images are shown on the left and a large version of the image is shown in the main panel when a thumbnail is clicked. Images can be removed from the list or extra images can be added. On the bottom left, information on the image such as the number of pixels, the size on disk, and the format are shown. When the user is satisfied with the selected images, he can upload them to the reconstruction server. In order to do so, the user first needs to authenticate himself with his username and password. The upload is started and a progress dialog shows the speed of the upload. In order to limit the upload- and processing time, the user can decide to downscale the images by a certain percentage.

For authentication and uploading, the tool makes contact with the ESAT webserver homes.esat.kuleuven.be, using ordinary HTTP Post, Put and Get commands. On the webserver a set of php scripts implement the necessary authentication and transfer checks. The power of php lies in its multiple add-ons: with a MySQL interface one can operate on a database, filesystem operations are available, as well as all sorts of crypting functions. We give two examples of php-enabled functionality.

- For authentication, we don't want logins and passwords transferred in plain text over the Internet. That is why authentication follows this strategy: First the webserver generates a random challenge, sent to the client.

The client then returns the username in plaintext and a MD5 hash of a combination of the challenge, the username and the password. On the server, the same MD5 hash is generated and compared. If they match, authentication is granted (for a limited amount of time).

- The upload is broken down into separate images. This circumvents the maximum upload size limit on the webserver and also makes the upload less error prone. The size of the uploaded image is returned to the upload tool and should correspond to the size of the local image. If it doesn't, it typically means the bitstream contained one or more special escape characters. If this happens, the image is reuploaded using base64 encoding. This increases the upload size with approximately 35% but alleviates the problem with special characters. When all images are uploaded correctly, they are inserted into the database as a new job.

Modelviewer Tool

Upon completion of the 3D reconstruction, the user is notified that the results are ready. They are stored in a zip file on the ESAT ftp server in a directory with a random name whose parent is not listable. This makes sure the user's reconstructions are safe from prying eyes. The zip file contains the original images, the calibration of the cameras, the dense depth maps and quality maps for every image. The results can be inspected by means of the modelviewer tool, a screenshot of which is shown at the top of Fig. 7.20. The layout of the modelviewer tool purposely resembles that of the upload tool.

A thumbnail is shown for every image in the set. If clicked, a large version of the image is shown in the main window. A triangulated 3D model can be created for the current image, using the depth map and camera of this image. Every pixel of the depth map can be put in 3D and a triangulation in the image creates a 3D mesh, using the texture of the current image.

Every depth map has its corresponding quality map, indicating the quality or certainty of the 3D coordinates of every pixel. The user can select a quality threshold. High thresholds yield models with fewer but more certain points. Lower thresholds yield more complete models including areas with a lower reconstruction quality. If required, certain areas of the image can be masked or grayed out by drawing with a black brush in the image. Areas that are masked will not be part of the 3D model. Homogeneous regions like the sky yield bad reconstruction results and can automatically be discarded by the user. A simple click inside such a region starts a mask-growing algorithm that covers the homogeneous region, as can be seen in Fig. 7.21 where the sky above Paisley Abbey in Scotland is automatically covered by a mask. The resulting 3D model is displayed in an interactive 3D widget. Two viewpoints of the Prato example are shown in the bottom part of Fig. 7.20. The model can be exported in different 3D file formats like VRML2, Open Inventor, OBJ or OpenSG's native file format.

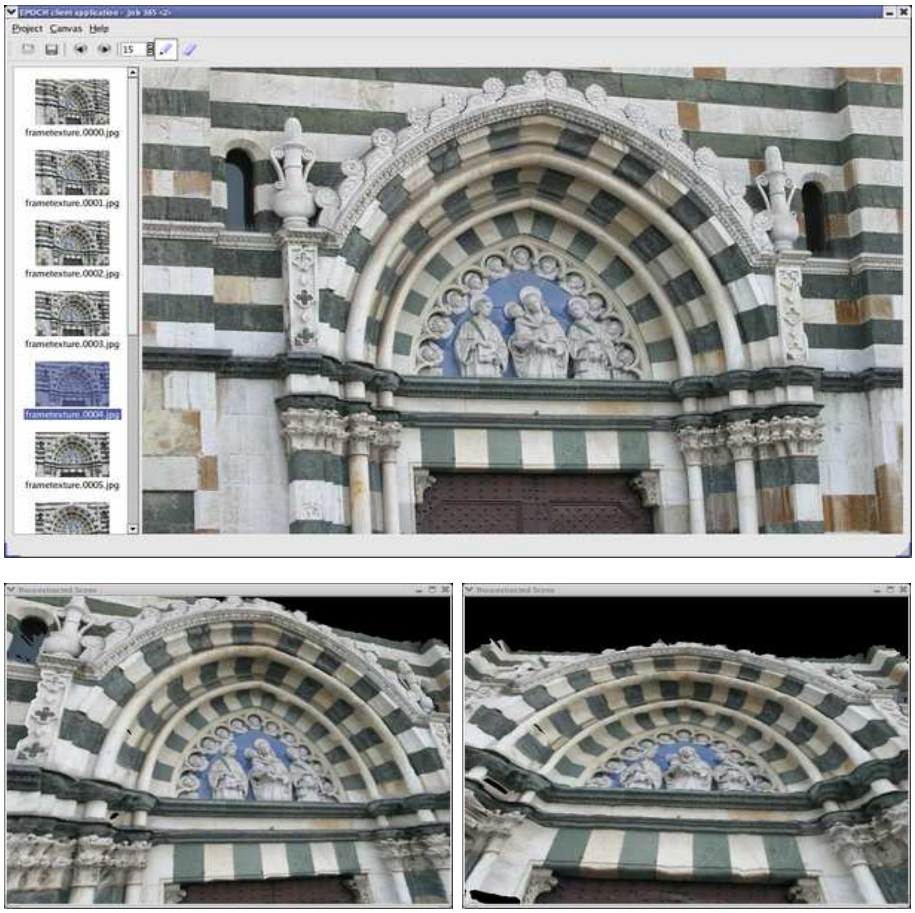


Figure 7.20: Modelviewer tool. The layout of the tool (top) deliberately resembles that of the upload tool, with thumbnails on the left and a large image in the main panel. The tool allows the user to create textured wireframe representations of the depth maps. The bottom two pictures show different views of such a 3D representation.

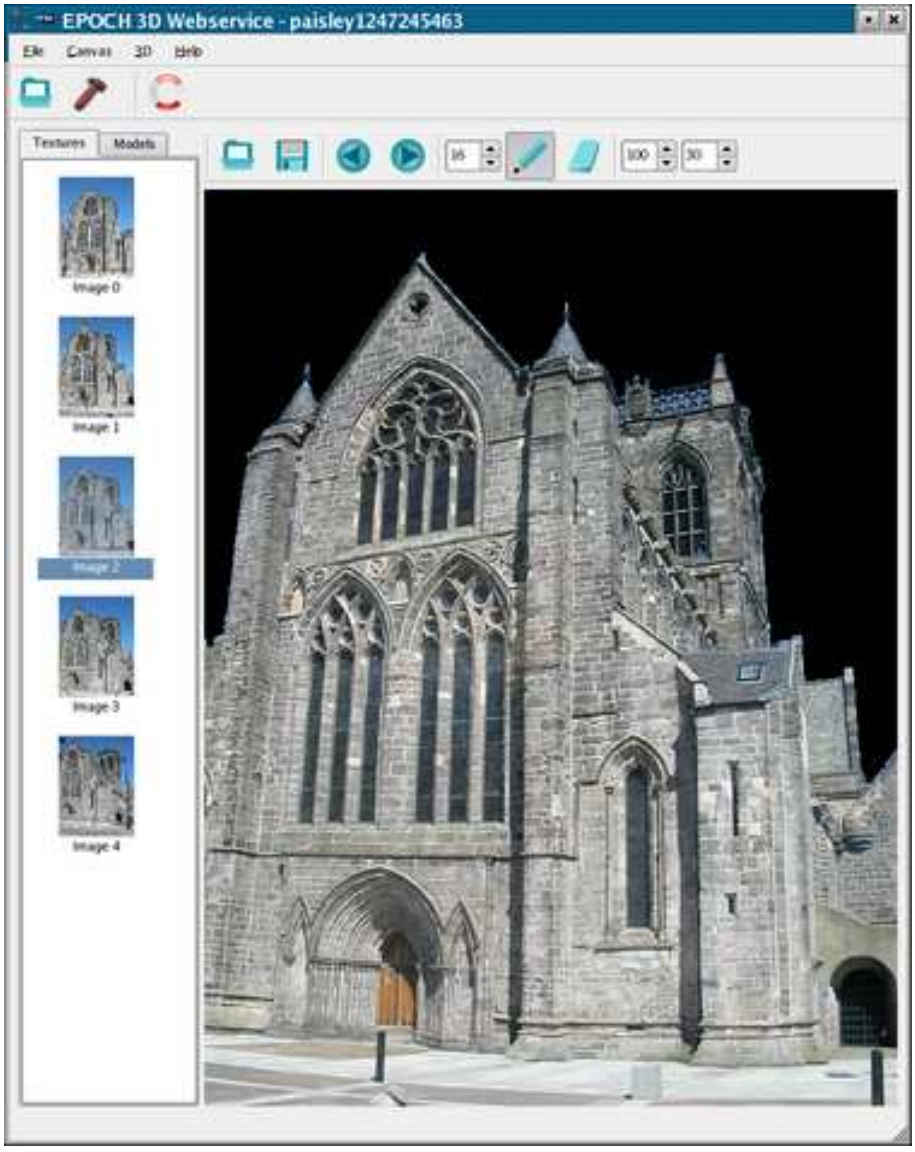


Figure 7.21: View of the Paisley Abbey in Scotland. The user has automatically masked the blue sky with a simple click inside the region.

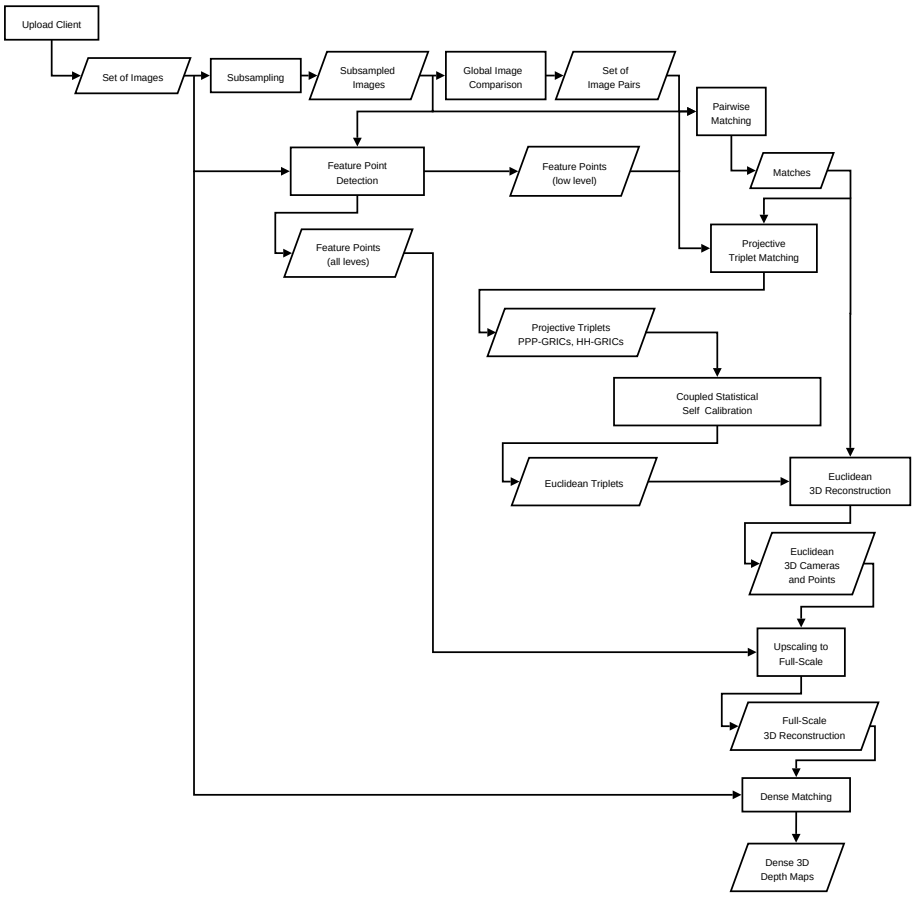


Figure 7.22: Global flowchart of the Reconstruction Pipeline

7.3.2 Automatic Reconstruction Pipeline

The 3D webservice is meant to create 3D reconstructions from images from a wide variety of sources. Because no user interaction is possible once the images have been uploaded, an important prerequisite is the need for robustness and autonomy on the server part. In order to avoid frustration and multiple uploads at the client side, the server should maximize the chance of obtaining a good 3D result, even from sequences of images that are not perfectly suited for the purpose. This requirement has led us to the development of a hierarchical, massively parallelized and opportunistic 3D reconstruction scheme.

A schematic flowchart of the reconstruction pipeline is shown in Fig. 7.22. In this figure rectangles represent procedures or actions. Parallelograms represent data structures. The input of the pipeline is found on the top-left and consists of the set of images the user has uploaded to the server using the upload client.

At the bottom right the result can be seen consisting of dense 3D depth (or range) maps. The pipeline can be seen to consist of roughly four steps:

1. A step that computes a set of image pairs that can be used for matching, including the Subsampling and Global Image Comparison steps. In this step, the images are first subsampled (hence the hierarchical nature of the pipeline). Since images can be uploaded in non-sequential order, we have to figure out which images can be matched. This is the task of the Global Image Comparison algorithm which yields a set of image pairs that are candidates for pairwise matching.

2. A step that performs the Pairwise and Projective Triplet Matching and the Self Calibration. In this step, feature points are extracted on the subsampled images. All possible matching candidates of step 1 are now tried. Based on the resulting pairwise matches, all image triplets are selected that stand a chance for projective triplet reconstruction. This process is performed and the results are fed to the self-calibration routine which finds the intrinsic parameters of the camera.

3. A step that computes the Euclidean reconstruction and upscales the result to full resolution. In this step all image triplets and matches are combined into one 3D Euclidean reconstruction.

4. A step that is responsible for the dense matching, yielding dense depth maps for every image.

Sections 7.3.3, 7.3.4, 7.3.5 and 7.3.6 describe our approach for each of these steps respectively.

Opportunistic Pipeline

Classic Structure And Motion pipelines make use of the fact that the set of input images is taken in a sequential manner. This helps the reconstruction process tremendously because only consecutive pairs of images must be matched for 3D reconstruction. Unfortunately the 3D Webservice described in this chapter cannot rely on this assumption. Users can upload images in non-sequential order or even use images that were taken in a random fashion. This has an impact on the matching step, the reconstruction step and the dense matching step.

Hierarchical Pipeline

In general the quality and accuracy of the resulting depth maps is proportional to the size of the input images. However, computing feature points and matches on large-scale images is very time-consuming and not so stable a process. That is why all incoming images are first subsampled a number of times until they reach a typical size in the order of 1000x1000. As can be seen in Fig. 7.22 most of the Structure And Motion processing is performed on these subsampled images.

It is only in the upscaling step (at the bottom right) that the result is upgraded from the low-resolution to the high input resolution. This hierarchical approach combines a number of advantages. First it is more stable and has a higher chance of success than direct processing of the high resolution images. Indeed, it is easier to find matching features between small images because the search range is smaller and therefore fewer false matches are extracted. On lower-level images feature points are typically more dominant than on higher levels where features also appear in noisy and cluttered areas that are very hard to match. Furthermore processing times decrease when dealing with smaller images. The upscaling step receives a very good initialization from the lower levels. This means that only a small search and optimization needs to be performed.

Parallel Pipeline

The 3D webservice has a rather intricate design with several parts working together. An ESAT server (called *thebes*) is dedicated for the reconstruction application. On this server a couple of services are running:

- a PostgreSQL database, keeping track of users and jobs,
- a python daemon listening to the database triggers, and
- a distributer daemon for parallel processing.

The python daemon listens to the triggers of the database. Whenever a new job arrives and is inserted into the database by the webserver, a new thread is launched that is dedicated to the reconstruction of this specific job. A new directory is created on the local disk of *thebes* in which all intermediate results are stored.

Several operations in the reconstruction pipeline have to be performed many times and independently of each other. Image comparison, feature extraction, pairwise or triplet matching, dense matching, etc are all examples of such operations. The job has to go through several steps and every step can only be started when the previous one has finished. Inside one step, however, the processing is parallelized, using the spare computing power of VISICS clusters, servers and computers. This parallelism must of course be controlled. This is where the distributer daemon plays its part. A schematic view is shown in Fig. 7.23.

The distributer on *thebes* holds a list of machines, together with their processing power and total and currently available memory. Every hour this list is updated and all machines are inspected for availability. On every machine a small daemon runs that listens to a specific TCP port. Whenever the job thread has to launch several processes (e.g. feature detection in all images), it does not do so directly but rather sends every process to the distributer, using a simple bash command like (in the case of the feature detection):

```
echo "/users/visics/vergauwe/bin/$(ESAT_ARCH)/DetectFeatures 0"\
| epoch-queue-submit
```

The submit executable adds this command to the queue in the distributer and returns a unique ID of the process. In the distributer, the status of the queue is checked every second. If it is not empty, the next available computing server is searched and the process is launched on that server. Availability of computing nodes is checked by the distributer using a short query on the TCP port of the remote daemons. A node is considered available if its current load is not too high ($\leq [1.5 \times \#CPUs]$). For machines that are not dedicated to webservice, the amount of webservice-based processes on them must also be lower than $[\#CPUs + 1]$. If the process to be launched needs a minimum amount of memory (e.g. for the dense matching), the availability of this memory on the computing node is checked as well. More intricate load balancing schemes can be envisaged, e.g. taking into account the time of day or the fact that a user is logged in on the console.

If an available node is found, the process command is sent to that node to be executed in a bash shell. All stdout and stderr output is redirected to a file in the directory of the job. When the last command of the process has finished, the output file is automatically renamed to `EPOCH.%d` where `%d` holds the integer with the unique process ID. This is the mechanism whereby the job thread can know of the status of its processes. Every so often it checks for the existence of the output files for its open processes. Once the file exists, it knows the process has ended. The job thread then inspects the output file and sees if the process has succeeded or failed. Failure is mostly due to bad input data, but sometimes an error on a computing node can occur. In this case, the process is simply resent to the distributer.

In order for this intricate scheme to work, we are in need of NFS access to the data directory on *thebes* (where also the output files are written) and the executables. In that respect, note the environment variable `ESAT_ARCH` in the command. The architectures of the computation nodes are diverse and the executables are compiled for every architecture separately. The exact architecture is of course only known at launch time on the node. The environment variable in the command makes sure the appropriate executable is called.

7.3.3 Global Image Comparison

The images that are uploaded to the reconstruction server can be taken in any random order and don't necessarily have to be recorded or uploaded sequentially. This means of course that it no longer suffices to process consecutive images. Unfortunately, matching every possible pair of images is a combinatoric nightmare when the amount of images increases. That's why we want to find out which images can be matched first.

For our pipeline we implemented a simple technique which is fast and still delivers good results. All possible pairs of images are fed to the *Global Image Comparison* algorithm explained in section 6.8.1 and section 7.2.3. This algorithm is based on comparing the two images under scrutiny with a comparison function like Normalized Cross Correlation (NCC). The goal of this step is to

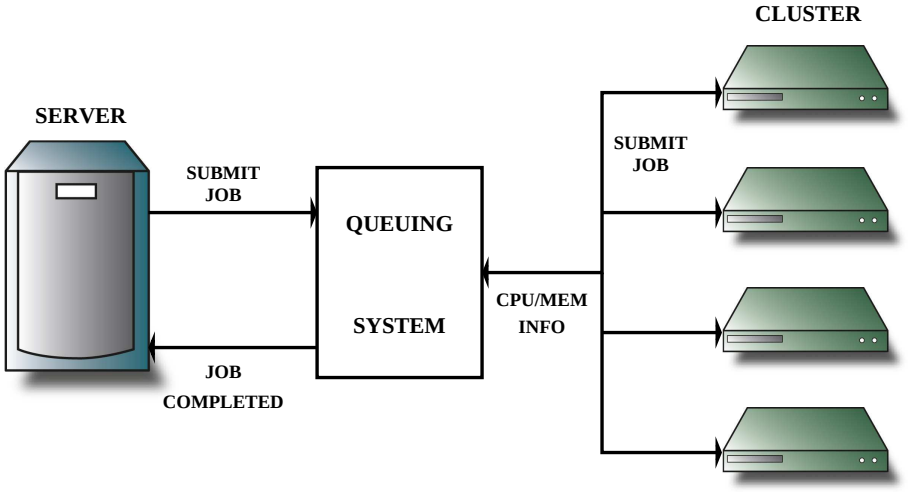


Figure 7.23: Queuing system. The server sends a job to the system. The queue has access to a PC cluster and sends the job to any of the nodes that becomes free.

quickly come up with pairs of images for which matching and computation of epipolar geometry is likely to succeed. All possible image pairs are processed in parallel (as explained in section 7.3.2) and only those pairs with a NCC value that is sufficiently high are kept for real pairwise matching. The shift between the windows that yielded this highest NCC value is stored and used later to facilitate the search for matches between the images.

In section 3.2.3 we discussed SIFT and SURF features which are invariant under scale and rotation and robust under some amount of out of plane rotation. Recently we have replaced our Harris corners with SURF features. This has improved the stability and success rate of the system for wide-baseline images tremendously, as will be shown in the result of Fig. 7.34.

7.3.4 Self-Calibration

Classical SaM techniques

Classic Structure And Motion systems either use a-priori information on internal camera calibration parameters or go through the process of first reconstructing the scene in a projective framework and then upgrading this reconstruction to metric using self-calibration and bundle-adjustment techniques [70, 39].

First feature points are extracted and consecutive images are matched. Epipolar geometry is retrieved and a projective frame is initialized. All consecutive images are matched to their predecessor and every new projective camera is computed in the projective frame of the first pair. As explained by Cornelis et

al [14] all projective reconstructions obtained in a sequential way suffer from drift. In a worst-case scenario, the drift of the recovered solution w.r.t. the true solution can become so substantial that self-calibration is no longer possible because the recovered projective reconstruction is not consistent anymore.

The simple sequential reconstruction approach also has problems with degenerate motions and scenes. Obtaining a unique projective reconstruction of a planar scene, for instance, is mathematically impossible because multiple projective solutions exist. The appearance of different projective frames in the solution causes self-calibration algorithms to fail as well. Also, contrary to the classical techniques, our input images are not always taken in a purely sequential way. It is very disadvantageous if these non-sequential images are only processed sequentially. Images should not just be matched to their direct predecessor to reconstruct their cameras. Very recently Snavely et al. [82] demonstrated a system at Siggraph that is capable of recovering the camera calibration of a non-sequential image set. The paper, however, does not reveal many details about the algorithms behind this system and whether or not it takes into account degenerate motions or scenes.

Triplet Matching

In order to alleviate the problems mentioned in the previous paragraph, a new reconstruction and self-calibration algorithm has been developed which has proved to be stable and rewarding on image sequences that could not be processed with the simple sequential approach. As explained in paragraph 7.3.3 all images in the uploaded set can efficiently be compared to each other. This comparison yields a list of image pairs that are candidates for feature matching. The extracted feature points are matched between each of these image pairs and epipolar geometry is computed. If the matching yields a sufficiently good result, the result is stored. When this pairwise matching step has finished, triplets with enough matches between both images 1 and 2 and images 2 and 3 are withheld. For these triplets a projective reconstruction is computed, in the form of 3 projective cameras and a set of 3D points. All these calculations (image pair matching and image triplet reconstruction) can be performed independently and thus all processing is done in parallel using the queuing system described in paragraph 7.3.2.

Our self-calibration will later make use of the reconstructed triplets. Not only do we want to reconstruct as many triplets as possible, we also need to investigate if they are useful for self-calibration purposes. In order to do so we make use of the Geometric Robust Information Criterion (GRIC). Section 4.4 explained the computation and the use of the two models and their corresponding GRIC values that can be employed to assess the planarity of a scene visible in a triplet of images, namely **PPP-GRIC** and **HH-GRIC**. If a scene is near to planar or if only a planar part is in common between the 3 views of a triplet, the corresponding HH-GRIC will have a lower penalty than the PPP-GRIC, indicating that a model consisting of two homographies better explains the data

than the model using three projection matrices. Such a triplet is therefore not suited for self-calibration and can only be correctly reconstructed if the internal camera parameters have already been recovered, e.g. from other parts of the data.

Statistical Coupled Self-calibration

Coupled self-calibration is a very powerful extension of self-calibration of a single projective sequence. It allows to calibrate reconstructions in different projective frames in one step. We explained the procedure in section 3.9.3. The algorithm is of high interest if only short projective sequences are available. Such sequences do not entail enough information to recover all parameters of the calibration model in a stable way if used by themselves. Coupled in one computation, they will allow for the internal parameters of the camera to be recovered, provided these intrinsics are constant. The most extreme example of a short projective reconstruction is a projective triplet. The calibration algorithm only yields good results if all reconstructions at the input are consistent in their projective frame. With only three views present in each triplet there is a chance that a combination of several triplets contains at least one weakly reconstructed triplet. That is the reason why we opted for a statistical approach.

All reconstructed triplets are possible candidates for coupled self-calibration. First the PPP- and HH-GRIC of each triplet are compared. Only triplets for which the PPP-GRIC value is lower than the HH-GRIC are retained because we are certain that there is enough 3D information in these triplets for self-calibration. From this set of triplets we randomly pick a certain number (7 in the current implementation). The matrix on the left of equation 3.35 is built and we solve for the internal parameters. If the smallest singular value of the SVD which solves this system is small enough the solution is accepted and stored. This step is repeated many ($\sim$1000) times or until all triplet-combinations have been tried, each time picking another random set of triplets. Each solution for the absolute quadric corresponds to a matrix with intrinsic parameters K. We retrieve the top left element of each K, representing the focal length in pixels in x-directions and create a histogram of these values. A typical histogram is shown in Fig. 7.24. This figure shows the typical Gaussian-like distribution of the self-calibration results we obtained from a multitude of experiments. The maximum of the fitted Gaussian is selected and the K-matrix corresponding to this value is chosen. Of course the self-calibration procedure is not meant to yield perfect calibration results but is rather used as a good initialization for bundle-adjustment steps that are executed later.

7.3.5 Reconstruction

Paragraphs 7.3.4 and 7.3.4 explained how we retrieved projective reconstructions of many image triplets and how we used these triplets to compute the intrinsic parameters of the camera that took the images. A straightforward next step is

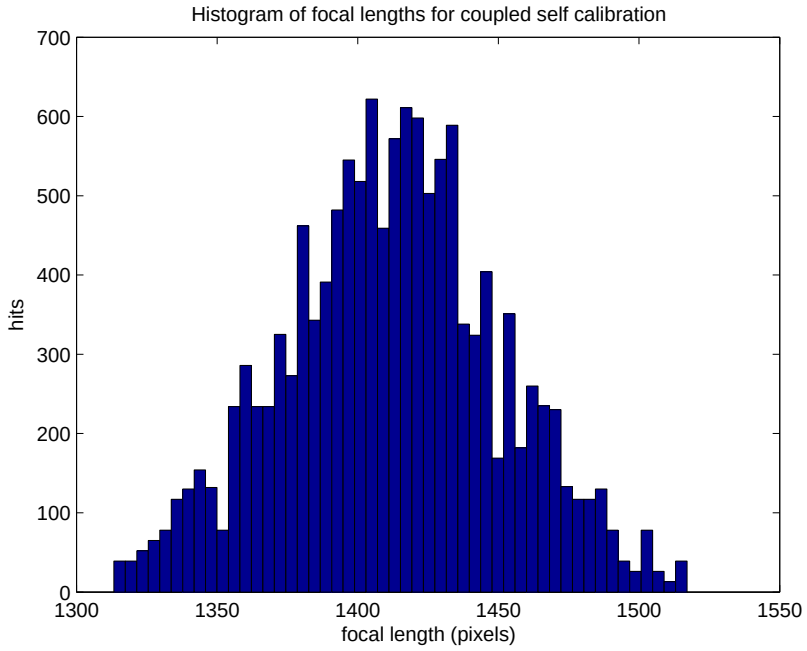


Figure 7.24: Histogram of focal lengths for all solutions of coupled self-calibration from a set of triplets.

to upgrade every triplet from its projective to a metric frame, using the intrinsic parameters $\mathbf{K}$. However, this procedure brings us only halfway towards our true goal: to retrieve the camera calibration and 3D points for all images in the same metric frame because each triplet is still reconstructed in its own metric frame.

In order to bring all information together, we start by searching for the triplet which is best suited for initializing our final 3D reconstruction. This triplet should have a large number of 3D points and have a large $\frac{HH-GRIC}{PPP-GRIC}$ value. When this triplet is found, our reconstruction is initialized. Other images can now be added to the reconstruction, one at a time. This is not done in a purely sequential way where only matches to the previously added image are employed but rather using all matches with all already added images. Using these matches a robust pose-estimation algorithm is executed which computes the camera of the newly added image in the metric frame of the reconstruction. The fact that the pose-estimation can be done in metric space is an advantage because now also images which only observe a planar part of the scene can be reconstructed.

Upscaling the Result

All previous computations were performed on the subsampled images, both for efficiency and stability reasons. It is now time to upgrade the result to the full-scale images that were uploaded, a process we have called *upscaling* the result. An iterative algorithm was implemented. It is executed as many times as needed to go from the small low-level images to the highest level of the original size images. The simplest way is to upscale the resulting calibration. Every upscaling with a factor of two is simply done by scaling the intrinsic parameters of the lower level ($\mathbf{K}_0$) to the higher level ($\mathbf{K}_1$) for every image i. The extrinsic parameters ($\mathbf{R}$ and $\mathbf{t}$) of every camera i and the position of every 3D point j ($\mathtt{M}_j$) can stay the same. This yields the new cameras $\mathbf{P}_i$.

$$\begin{aligned}
\mathbf{K}_{0i} &= \begin{bmatrix} f_{x0i} & s_{0i} & u_{x0i} \\ 0 & f_{y0i} & u_{y0i} \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{K}_{1i} &= \begin{bmatrix} f_{x1i} & s_{1i} & u_{x1i} \\ 0 & f_{y1i} & u_{y1i} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2f_{xoi} & s_{0i} & 2u_{xoi} \\ 0 & 2f_{yoi} & 2u_{yoi} \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{R}_{1i} &= \mathbf{R}_{0i} \\
\mathbf{t}_{1i} &= \mathbf{t}_{0i} \\
\mathtt{M}_{1j} &= \mathtt{M}_{0j} \\
\mathbf{P}_{1i} &= \mathbf{K}_{1i} \left[\mathbf{R}_{1i}^T | -\mathbf{R}_{1i}^T \mathbf{t}_{1i} \right]
\end{aligned} \tag{7.1}$$

This solution can only serve as an initialization for an optimization process because all errors of the low-level reconstruction are upscaled as well. As can be seen in Fig. 7.22 we computed feature points in the images on all levels, i.e. on both subsampled and original images (and possibly intermediate levels too, depending on the size of the originals). The number of extracted features is increased for every consecutive level. These features will help us to optimize the result of Eq 7.1. The algorithm that has been implemented to upscale the resulting reconstruction is iterative and works as follows.

1. Compute a first initialization for the cameras $\mathbf{P}_{1i}$ and points $\mathtt{M}_{1j}$ of the new level based on the level below as explained in equation (7.1).
2. The 3D-2D correspondences that indicate in which images every 3D point is visible remain identical.
3. For every 3D point $\mathtt{M}_{1j}$ perform the following operation:
   - Project $\mathtt{M}_{1j}$ in the images where it is visible using the new cameras $\mathbf{P}_{1i}$.
   - Search the extracted features of the image at level 1 for the feature that is closest to the projected point.

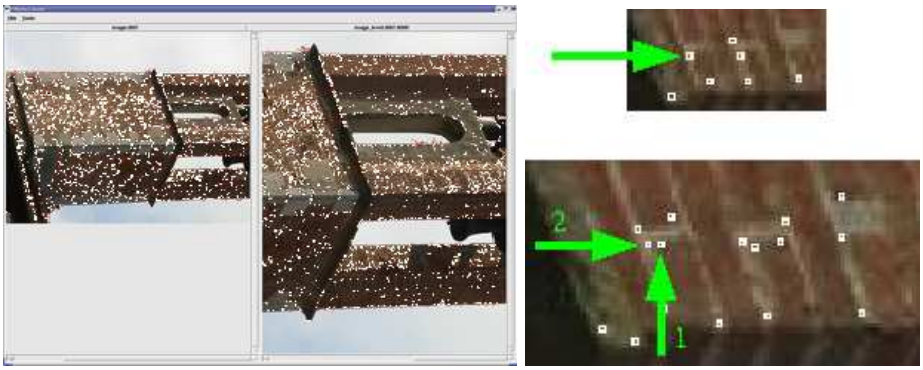


Figure 7.25: Subsampled and original image with extracted features are shown on the left. On the right one part of the images has been cut out. The arrow at the top points to a feature with a 3D point in the low-level image. At the bottom arrow 1 points to the feature in the high-resolution image which is closest to the reprojection of the 3D point using the initialization (eq 7.1). Arrow 2 points to the feature that is selected after 1 iteration of the 3D points and camera parameters.

- Change the 3D-2D correspondence to this closest feature.

4. Perform a Robust Euclidean Bundle Adjustment on all 3D-2D correspondences. This step computes updated values for the 3D points $\mathtt{M}_{1j}$ and cameras $\mathbf{P}_{1i}$. This step is explained in paragraph 7.3.5.

5. Iterate steps 3 and 4. Because of the updated correspondences, the updated values of the points and cameras will yield new projections. Some of these will now be closer to another feature in the feature set. Fig. 7.25 shows an example. This implies a new change to the 3D-2D correspondences and hence again an updated solution. The iteration is stopped when no more updates to the 3D-2D correspondences are made. This is typically reached after 5-6 iterations.

The upscale step described above succeeds in computing the Euclidean projection matrices and 3D points for the full-scale images. Non-linear radial distortion parameters can be estimated as well. The technique has one drawback however. Only originally matched points will be upscaled and no extra points are added to the reconstruction. The reconstructed points are those that are inliers to the epipolar geometry of image pairs. Since at the time of computation of this geometry no information on the radial distortion is available, typically few points near the borders of the image are matched if the images suffer from this kind of distortion. This is because radial distortion has the largest effect on points far away from the center of the image. After the upscaling step, however, we have the parameters describing the distortion to our disposal. This means

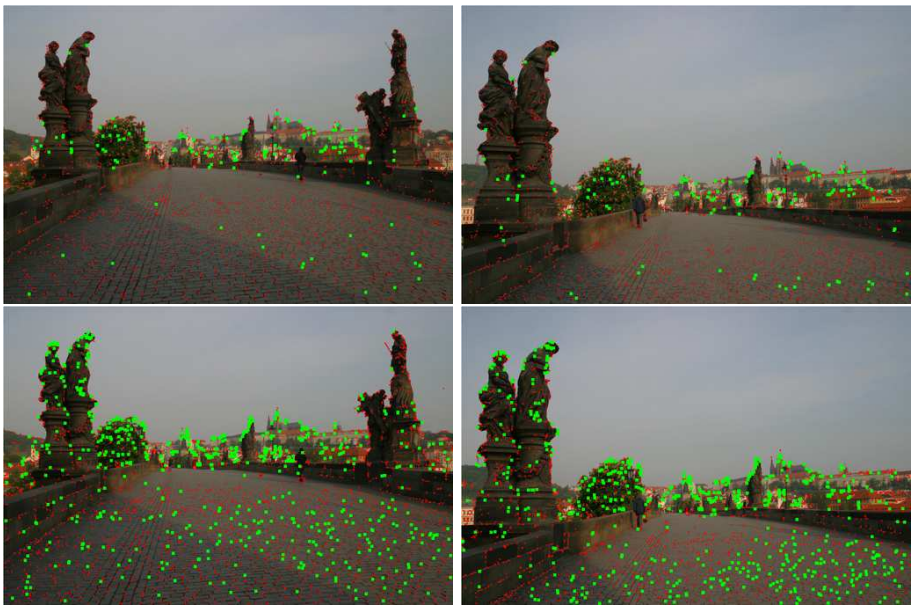


Figure 7.26: The top two images show an image pair matched at the smallest pyramid level without taking radial distortion into account. The bottom image pair shows the matches on the full size images when accounting for radial distortion. Especially the foreground region of the images shows many more matched features.

that now we can search for more matching points, taking into account this distortion which yields many extra points that are also located near the border of the images. Fig. 7.26 shows an example. The top two images show an image pair matched at the smallest pyramid level without taking radial distortion into account. The bottom image pair shows the matches on the full size images accounting for radial distortion. Especially the foreground region of the images shows many more matched features.

Robust Euclidean Bundle Adjustment

The Robust Euclidean Bundle Adjustment of step 4 is an extension of the traditional bundle adjustment techniques. In these least-squares estimators the global reprojection error is minimized. It has been shown in numerous studies that least-squares estimators are vulnerable to the violation of the assumption that all data at the input consists of inliers. Sometimes even when the data contains only one bad sample, least-squares estimates may be completely perturbed. In the upscaling step outliers are inevitable. That is why a robust version of the classical bundle adjustment technique is called for. A popular technique to deal with possible outliers is using so-called *M-estimators* [107]. In short, M-

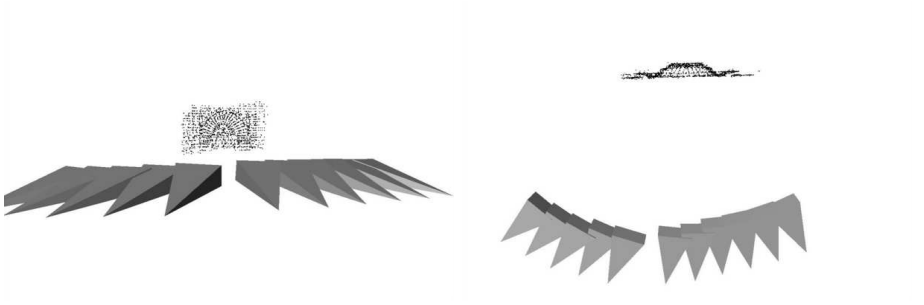


Figure 7.27: Reconstruction of points and cameras for the *Prato* example.

estimators try to reduce the effect of outliers by replacing the squared residuals by another function of the residuals. In our implementation we chose Cauchy's function, also known as the Lorentzian function given in equation (7.2). Instead of minimizing the sum of the residuals $\sum r_i^2$ we now minimize $\sum \rho(r_i^2)$. The parameter c of equation (7.2) is set to 1 which means that the M-estimator starts limiting the influence of outliers when they are more than 1 pixel away from their estimate.

$$\rho(x) = \frac{c^2}{2} \log\left(1 + \left(\frac{x}{c}\right)^2\right) \tag{7.2}$$

Although the resulting 3D points and calibration from the upscaling algorithm are accurate, the number of 3D points can be rather low because the algorithm only removes outliers and never adds 3D points or 3D-2D correspondences. As will be explained in paragraph 7.3.6 some dense matching approaches can use reconstructed feature points as seeds for their dense reconstruction. In order to have enough seeds it is important to increase the number of 3D points. The recovered calibration is used to search for new correspondences in very localized areas (along the epipolar line for the first two images and in a neighborhood around the projection of the 3D point in consecutive images). A typical result, consisting of calibrated cameras and a set of reconstructed 3D points is shown in Fig. 7.27 and Fig. 7.28, showing a sequential and non-sequential image set respectively. The calibration of the non-sequential image set of Fig. 7.28 makes use of non-sequential links between images. After the initialization with the best triplet, new cameras are added using all matches with all already added images. The links that were used in this specific example are shown as lines connecting the camera centers.

7.3.6 Dense Matching

The result of the Euclidean reconstruction is a set of 3D points and cameras, an example of which is shown in Fig. 7.27. This sparse set of points cannot be the end-point of the reconstruction process. The cultural heritage partners expect

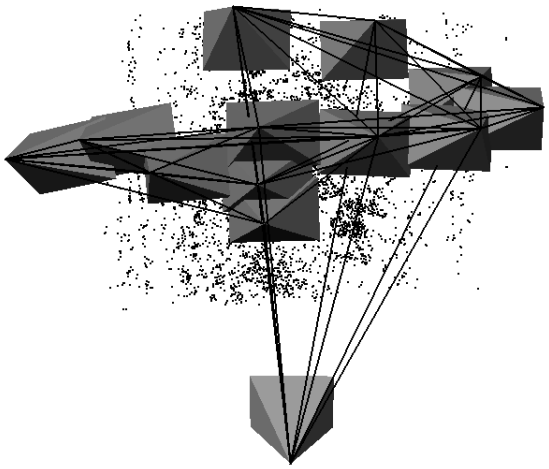


Figure 7.28: Point reconstruction of the *skull* sequence. The images are clearly recorded in a non-sequential order.

dense, textured 3D models. In order to deliver these, a dense matching process between the input images is necessary.

Linked Pairwise Stereo

In section 4.7.2 we described a dense stereo matching technique for image pairs. The resulting disparity maps can be combined into depth maps for every image using the statistical algorithm of section 4.7.3. This approach has proven its merit on sequential image sets, like the Prato example of Fig. 7.27. The quality maps (see section 7.3.1) indicating the confidence we have in the 3D reconstruction of every pixel are easily computed from the linking algorithm. The more consecutive views in which a pixel is matched consistently, the more confident we are about its 3D position, hence the higher the quality measure for this pixel.

As already mentioned, however, we cannot assume that the uploaded image sets have indeed been recorded and uploaded sequentially. Using the linked pairwise stereo method can lead to failure with such non-sequential image sets. If we just process the images pairwise as before the chances of encountering an image pair with extreme disparities or even hardly any overlap are very real. This would immediately break the link between the pairs, resulting in much worse depth maps. A possible strategy would be to search for an optimal sequence connecting the images in the image set in the best order. This is not an easy task. We have to decide which criterion will be used to assess whether two images should be next to each other in the sequence or not, for instance the number of pairwise matches (a result of the algorithm described

in section 7.3.4). Unfortunately, while this number gives a good indication of the matching quality between every pair, it does not give any information on the quality of the resulting 3D reconstruction which could be very bad for close views which have a small baseline. Furthermore finding the optimal path in the image set boils down to solving a Traveling Salesman problem, which is n-p hard and scales very badly. That is why some experiments were done using true multi-view stereo approaches.

Multi-View Stereo

Multi-view stereo algorithms seek to reconstruct the 3D shape of a scene directly from a combination of more than two images. Again, the most difficult step is to find the correspondence between points in different viewpoints. One can make a distinction between two types of techniques. The first are so-called volumetric algorithms. They start with an approximation of the volume of the 3D scene. This can be computed from reconstructed feature points or, more common, with visual hull techniques. The initialization of the volume evolves towards the true shape by enforcing photo-consistency between points that are seen in different images [72, 101].

Other techniques do not make use of a volumetric description of the scene but try to find multi-view correspondences from the images directly [44, 83], possibly initialized from reconstructed feature points. The technique of [83] from our lab builds up a probabilistic model of the correspondences between all views and a reference view, taking into account the possibility of occlusions, color changes, ... and maximize the probability of the solution. This result can then serve as initialization to a pixel-based PDE approach.

Fast direct multi-view methods have been implemented on the basis of GPUs. Graphics boards have become fully featured processing units with dedicated high performance memory and up to 16 parallel pipelines. The introduction of vertex and fragment programs caused a major breakthrough, giving the programmer almost complete control over the GPU. A technique from our lab which uses these GPUs for multi-view stereo was explained in [15].

A non-sequential image set of a skull was recorded and uploaded to the webservice. A point-based reconstruction is shown in Fig. 7.28 which shows the non-sequential nature of the recording. Two textured and one non-textured views of the result of the PDE based method are shown in Fig. 7.29. Different views of the result of the GPU-based method are shown in Fig. 7.30.

7.3.7 Results

The 3D reconstruction webservice has been operational for some months. Several image sets have been uploaded to the service by various users in the cultural heritage field. In this section we show some results. Fig. 7.31 shows some results for the façade of the Cattedrale di Santo Stefano in Prato, Italy. The image set

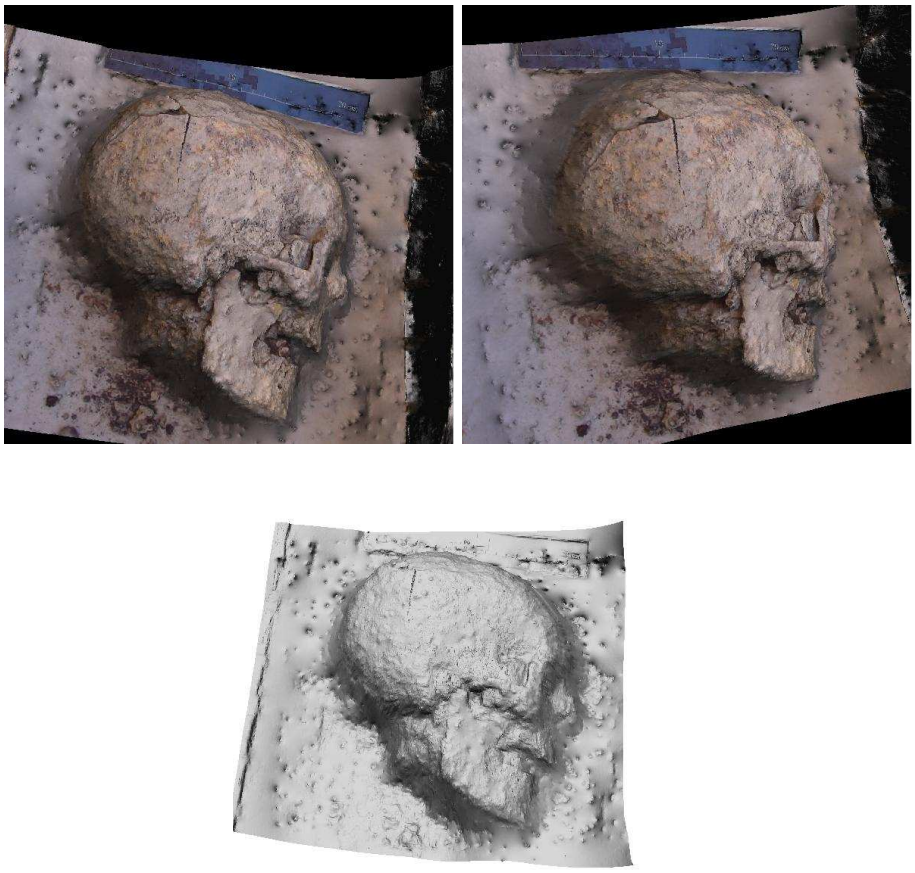


Figure 7.29: Reconstruction of the *skull* sequence made with the PDE-based dense matching method.

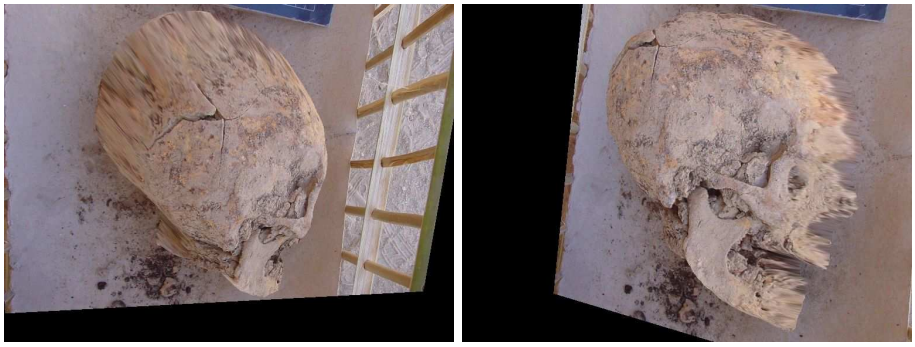


Figure 7.30: Reconstruction of the *skull* sequence made with the GPU-based dense matching method.

was recorded and uploaded by archaeologist Tijl Vereenooghe during an Epoch review in Prato. The image set was sequential as can be seen in Fig. 7.27.

Fig. 7.32 shows resulting depth maps for a stratigraphy at the archaeological excavation of Sagalassos in Turkey. Instead of using textual descriptions of each layer and measuring positions of artefacts with a ruler, the webservice offers a more efficient alternative. Each layer is recorded with a photo camera and these pictures are uploaded to the service. The results describe the layer in 3D, including textures. The second screenshot in Fig. 7.32 shows a top view. The image for which the depth map was computed that is depicted here, was taken from the side. It is clear that occluded areas behind rocks and other obstacles have not been reconstructed and show up as holes in the top view. These should be filled by merging depth maps. This is the goal of the steps in Fig. 7.17 that follow the reconstruction pipeline described in this paper.

Archaeologist Maurizio Forte of CNR-Rome used our webservice at a UNESCO conference in Mexico. He uploaded images of the Templo de los Mascarones (Temple of the Masks), at the Archaeological site of Edzna, which is a Maya site in the peninsula of Yucatan, near Campeche in Mexico. Fig. 7.33 shows two textured and one non textured views of the resulting reconstruction. We draw the reader's attention to the non textured view which shows the 3D relief of the mask and the ornaments much clearer than can be discerned from the original imagery or from the textured 3D views.

Finally, Fig. 7.34 gives an example of a true wide-baseline problem that we can solve with the techniques developed for the webservice. A user uploaded only three images of a façade on the San Marco Square in Venice, Italy. Mathematically, this is the smallest number of images one needs for reconstruction and self-calibration. The images, shown in Fig. 7.34, are clearly taken far apart, with an angle between cameras of more than 30 degrees. It proved to be impossible to process these images with our earlier standard SaM pipeline. First, the images differ so much that standard feature matching is doomed to fail due to the affine deformation of local image patches. The SURF features we employ in the webservice pipeline did a much better job. We managed to compute about 300 3-view matches, which was sufficient to calibrate the cameras and have a good idea of the 3D structure of the scene. For the dense matching, we used the normal sequential stereo matching annex linking approach. The resulting 3D model without texture is shown on the bottom right of Fig. 7.34. Not everything in the images has been reconstructed. Only the parts that are visible and matchable in the three views are reconstructed. The result is very convincing and shows the power of the algorithms on such difficult input data.

7.4 Conclusion

In this chapter we described two computer vision applications, based on the 3D reconstruction techniques of the previous chapters and targeted to preservation or presentation of our cultural heritage.

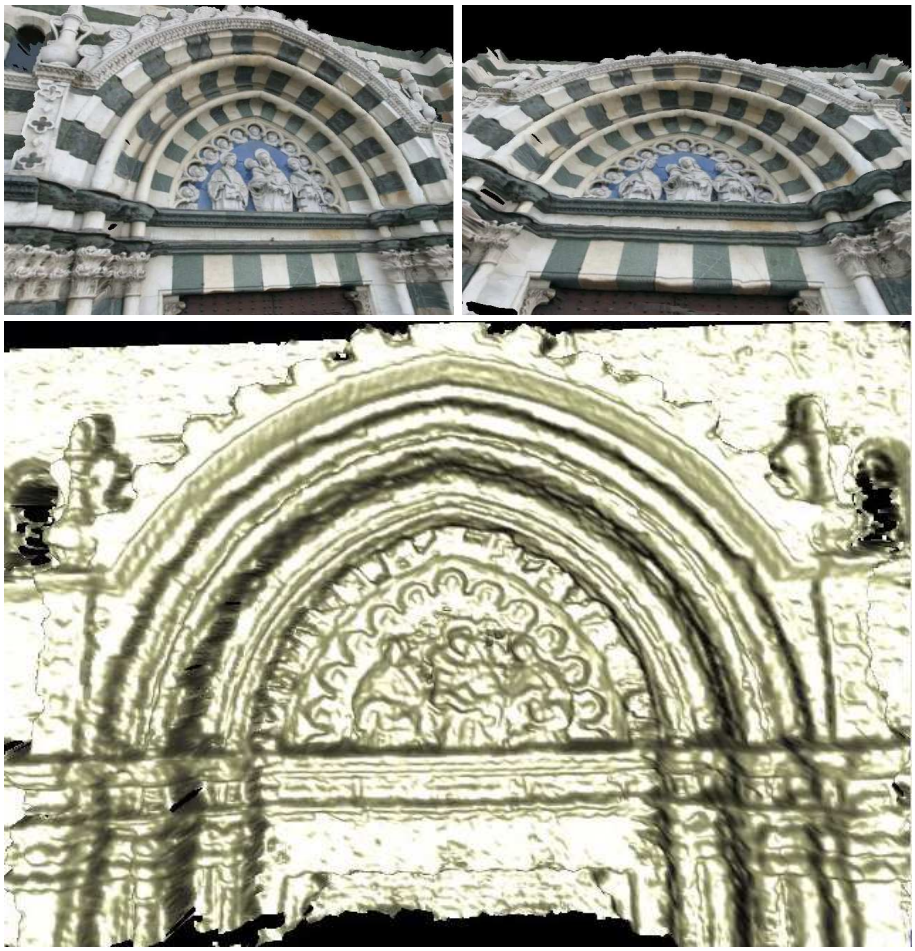


Figure 7.31: Reconstruction of the *Prato* example. The top two views show textured models, the bottom view shows a non-textured version.

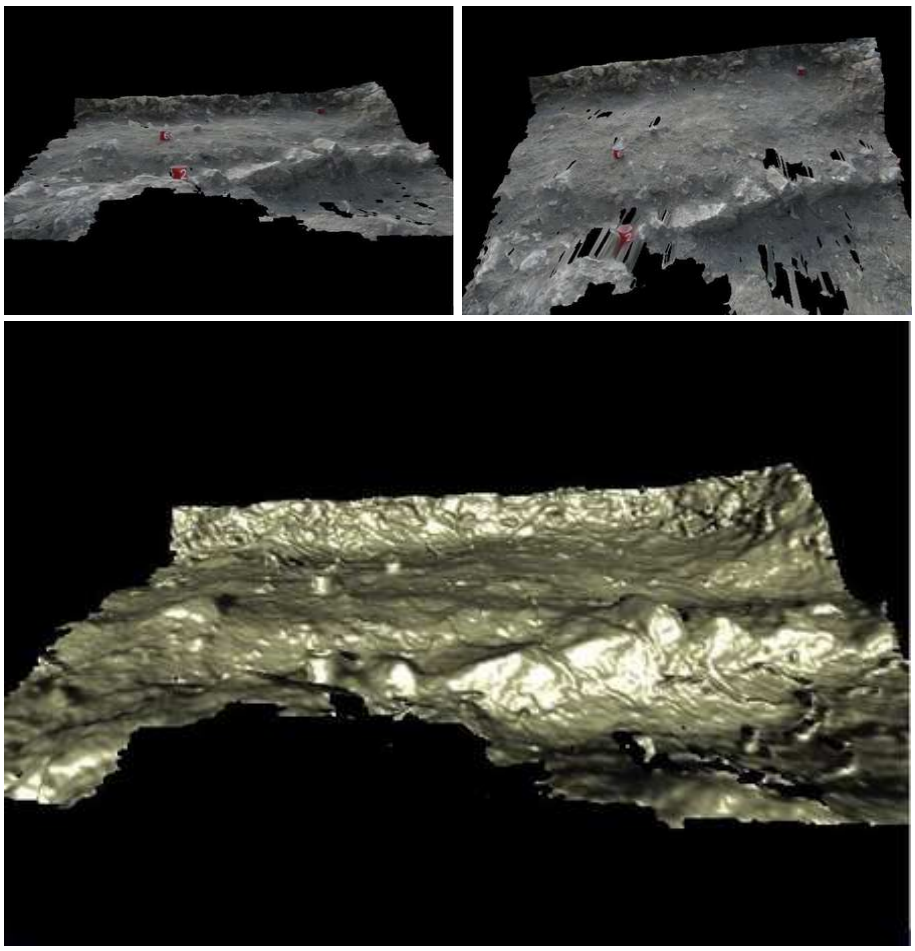


Figure 7.32: Reconstruction of a stratigraphic layer in Sagalassos. The top two views show textured models, the bottom view shows a non-textured version.

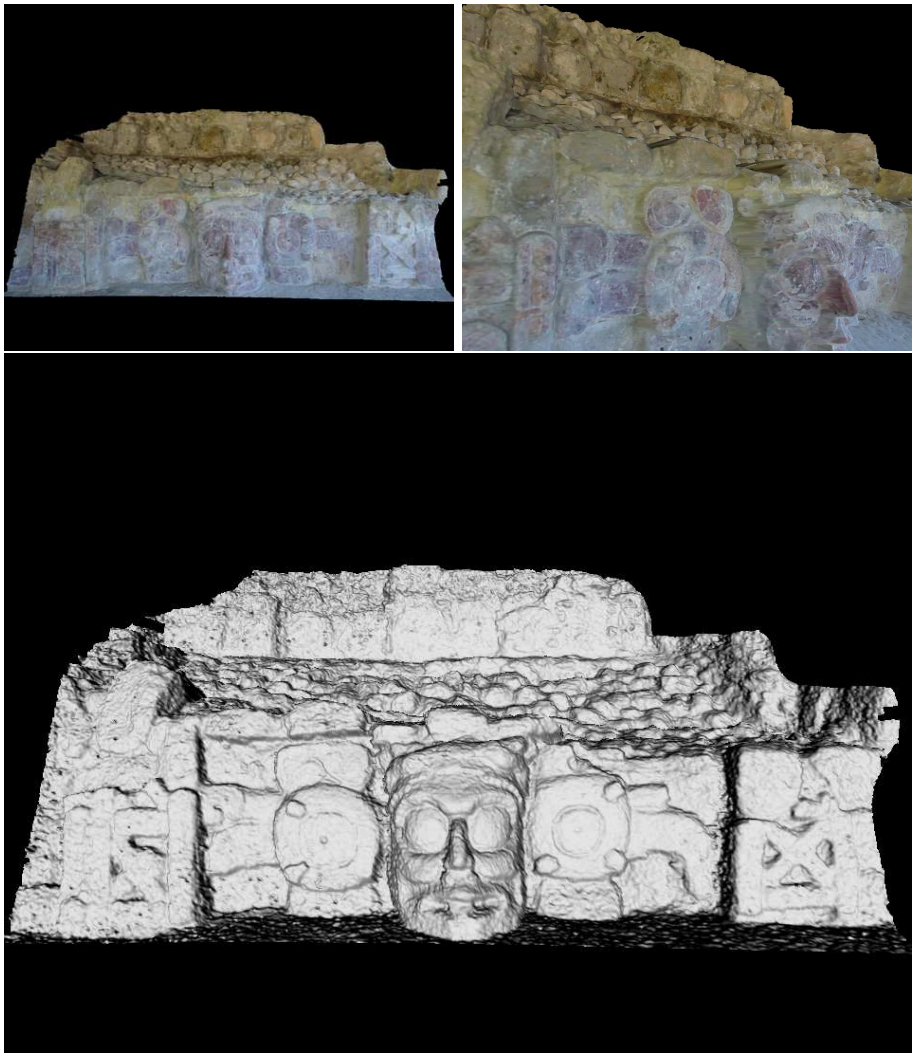


Figure 7.33: Two textured and one non-textured views of the reconstruction of the *Templo de los Mascarones*. The latter gives an especially clear view of the complicated geometry of the scene.

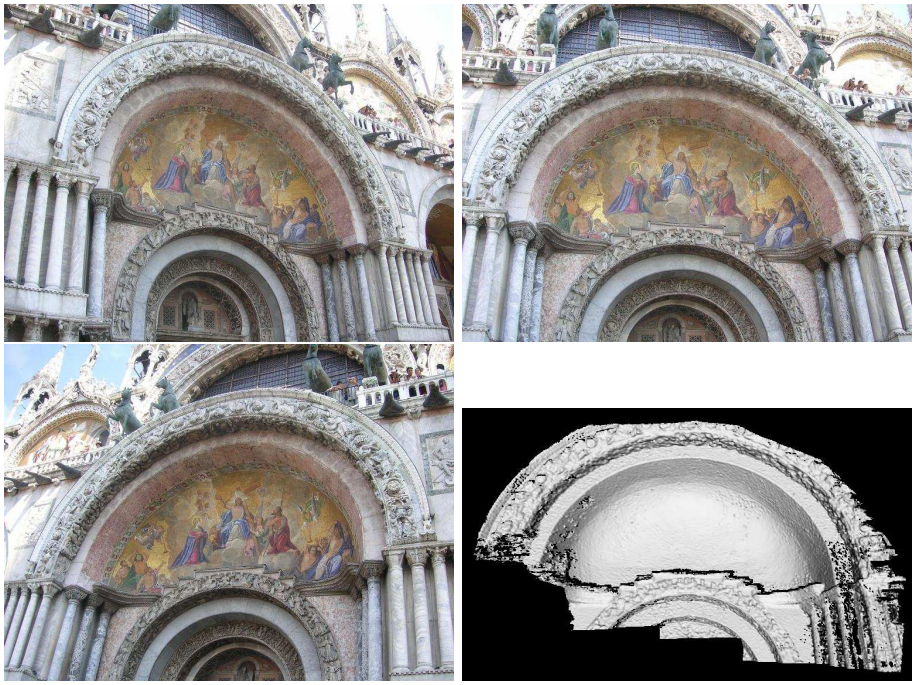


Figure 7.34: Top and bottom left: the three original images of part of the San Marco façade in Venice. This is clearly a difficult wide-baseline problem. Bottom right: resulting dense 3D reconstruction, shown without texture.

First a system was described that added the current time period to an existing QTVR movie of an archaeological site. SaM and IBR techniques were employed on aerial imagery of the site to accomplish this. My fast global comparison technique built a correlation matrix which was used to group the images into sets, suitable for SaM. For each set the camera calibration and point reconstruction was computed with my SaM algorithms, including essential matrix computation to deal with the planarity of the scene. Dense depth maps were computed with our standard stereo and linking approach. The IBR techniques, used to generate the novel views, were implemented by my colleague Geert Willems. The resulting presentation has proved very popular with visitors of the Provincial museum and the archaeological site at Ename. The same principles could clearly be used for many sites, worldwide.

The following section described the first part of a complete 3D reconstruction system for the cultural heritage field. Images uploaded to the service are turned into a 3D reconstruction together with an overview of the camera positions. The system under scrutiny is in charge of computing the camera calibration and dense depth maps from images of an object or a scene. The server automatically computes the results, making use of a cluster of PCs after which the results are made available on an ftp server. Several specifics of the upload-server paradigm had to be taken into account, such as the fact that no user interaction is possible and that input images might not be taken or uploaded in a sequential order. These constraints drove the design of the pipeline. It is automatic, opportunistic, hierarchical and tries to execute as many processes in parallel as possible. Only thanks to the experience that I acquired in multiple projects could this system be built. In this respect I can mention the web-based upload of imagery and my GRIC algorithms, based on my developments in INVIEW, the use of QT for the GUI from several ESA projects like TELEMAN and my choice for a database-driven system and my 5-point E-matrix computation algorithm from AEROBOTS. Of course, not all algorithms were simply lying on the shelf and additional building blocks were needed. My statistical coupled self-calibration and my upscale algorithm for instance were developed specifically for this application.

Chapter 8

Conclusion

8.1 Summary

This dissertation gives an overview of SaM algorithms and shows their use in a wide variety of application domains. The problem behind this work can be stated as *Which information can we retrieve when we have access to a set or sequence of images of a scene, taken from different positions*? During my research and in the scope of many projects, I have developed an extended set of algorithms that, from the input images only, can recover the 3D structure and the calibration of the camera. The number of algorithmic variations can be explained by the wide gamut of application fields as some algorithms are inherently more appropriate than others, depending on the specifics of an application, the recording, the scene, etc.

In the first chapters the theory behind the algorithms has been explained while further in the text the focus shifted towards the applicability, the robustness and efficiency of the techniques that are needed to solve specific problems. The separation is not so strict, however. It was our intent to illustrate the SaM algorithms with concrete examples and to indicate where and why algorithms might fail and how these issues can be resolved.

After I discussed the necessary background information on camera calibration, I introduced SaM algorithms and explained how a metric 3D reconstruction can be recovered from images of a scene only. High-quality features are vital for this and the recent progress on invariant features has been incorporated into my pipeline. Starting from robust two-view matching, introducing my preemptive RANSAC technique, my global transformation computation and my fast F-guided matching algorithm, I moved on to processing multiple views. For this, multiview reconstruction in a projective frame and self-calibration techniques are employed. Not only sequences of images can be processed as was shown when I discussed a multiview camera setup and its calibration.

More often than one would expect, degenerate cases are encountered that cause traditional methods to fail, the most common and thus most important of

which is a planar scene. I explained how Torr's GRIC can help us to distinguish between planar and non-planar scenes. F- and H-GRICs lie at the basis of my system that recovers the SaM from long video sequences. I devised an intricate scheme with sub-sequences in which keyframes are found using the aforementioned GRICs and which uses my coupled self-calibration algorithm to recover the metric reconstruction. My gluing algorithm knits the sequences together once more. I introduced the concept of the essential matrix and showed that, if a 5-point algorithm is used, the computation of this matrix does not suffer from planar degeneracy. I implemented Nister's rather involved 5-point method but also showed that in specific cases, like a tilting stereo head, one can get away with the much simpler 7-point method if one combines the data of different tilts. After we have reached a stage at which the calibration of the cameras and a sparse structure of 3D points has been recovered from the original feature sets, the next logical step consists of improving the realism of the reconstruction by searching for many more matches. Dense stereo matching, combined with a linking algorithm was introduced to this end.

At this point in the text, the basic SaM techniques had been explained and it was time to move on to specific applications, the first of which consisted of the robotic control experiments that I performed in Japan on the ETS-VII satellite. Because of the binary nature of the images, most of the vision-based robotic control experiments on the satellite made heavy use of geometric invariants, like the cross-ratio of four collinear points or the common tangent points of ellipses. These invariants provided the necessary 3D-2D correspondences to perform pose estimation, the results of which were used for robot guidance and later, in an off-line phase, for a robot calibration experiment. Even though the images contained little texture, I was still able to run my uncalibrated SaM algorithms on them with relatively good success.

Another important space application is planetary exploration which was the topic of the next chapter. I described two different ways to explore planets autonomously, making use of SaM algorithms. First I discussed a system with a stereo head controlling a rover. I described specific calibration and reconstruction algorithms running on the same imagery, sent down from the stereo head. The resulting disparity maps were combined into one digital elevation map, used for path planning, simulation, control and visualization. The system was displayed during the ASTRA conference where the audience could see the rover traversing the terrain on a planned path during several hours while it was automatically controlled by the stereo head. A second ESA project studied the feasibility of using uncontrolled Aerobots for planetary exploration. The images from the on-board camera could be immediately processed to create 3D DEMs. To this end I designed and implemented an intricate software system with several components, connected via a database, including an overlap checker, a SaM algorithm with a windowed bundle-adjustment and a dense matching routine with appropriate compression. The system was tested first with simulated imagery, then with a digital video taken on a gantry over the terrain and finally with images from a camera on-board a real aerobot.

At last we came to the chapter describing my work targeted to the preservation or presentation of cultural heritage. First a system was described that added the current time period to an existing QTVR movie of an archaeological site. Since I needed to generate images from precise locations, I combined imagery, shot in a helicopter, using Image Based Rendering techniques. These were fed by the results of my SaM algorithms, making use of my fast global comparison technique to build a correlation matrix and a manual registration technique. The resulting presentation is on display in the Provincial museum and the archaeological site at Ename. The same principles could be used for many sites, worldwide.

The final application in this work is the first part of a complete 3D reconstruction system for the cultural heritage field. The system under scrutiny is in charge of computing the camera calibration and dense depth maps from images of an object or a scene. These images are uploaded to a server and the result, automatically computed on a cluster of PCs, is made available on an ftp server. The webservice application is the culmination of all SaM algorithms I devised and all the experience I acquired in previous projects and applications. It is automatic, opportunistic, hierarchical and tries to execute as many processes in parallel as possible. The webservice is quite popular with cultural heritage professionals and we are investigating possible commercialization.

8.2 Discussion and Future Work

Engineers tend to be conservative in the sense that they don't try to fix what's already working. The SaM algorithms described in this work have proven their merit in several fields. However, the constant flow of new projects have forced me to adapt and rethink my software continuously. Every new application has its own peculiarities which ask for adaptations of existing algorithms or the design of new ones. In retrospect, it were the projects in which I encountered situations that could not be dealt with with (slightly modified) existing algorithms, that pushed the development forward. I can mention the satellite experiments, which came first chronologically and which forced me to learn about pose estimation, invariance and robotic control. The latter seems unrelated to SaM but the knowledge has helped the implementation of the planetary exploration systems as I had to deal with robotic hardware in each of these situations. The INVIEW project forced me to deal with long video sequences. Adaptations needed to be made algorithmically to detect keyframes and to counter drift but also on an implementation level to deal with memory problems. The Aerobot project was another milestone. It put me in a situation where I could no longer ignore the problem of degenerate cases. The implementation of the 5-point algorithm and the GRICs is still essential in my current system. Also the knowledge I gained in using a database as the central core of a software system has helped the setup of the webservice tremendously. That my algorithms have progressed over the years, can be proven by the many image sets that I can now process and that

yielded no result before. A good example of this, are the images of the ETS-VII taskboard from which I could not retrieve the cameras in an uncalibrated way at the time, but which are now processed without any problem, although the dense matching result is still not up to par.

Although we have come a long way, there are still many shortcomings or possible areas of improvement. When we regard the 3D webservice, it is clear that the dense matching component is currently the weakest part. This should not come as a surprise since it has essentially stayed the same since it implementation approximately 6 years ago. Images with nicely textured objects pose few problems but scenes with large untextured but continuous areas don't come out well. One could think of taking into account image gradient information to lock onto strong edges and smooth the disparities stronger across relatively untextured areas. We still employ the sequential linked stereo matching approach. This can suffice for sequential image sets. However, the SaM algorithms can recover the calibration for non sequential image sets and we therefore need a dense algorithm that can deal with these as well. Multiview dense matching approaches exist although most are targeted towards multicamera systems where accurate segmentation methods yield a visual hull to initialize the algorithms. A disadvantage of many methods is the large number of parameters one can tune. Recently Strecha e.a. [84] and Gargallo e.a. [27] have proposed techniques that estimate the optimal parameters together with the dense depth maps.

A necessary second step in the 3D webservice story is the integration of multiple depth maps and of different sequences. All depth maps of one sequence are registered in the same coordinate system but are not yet integrated into one model. In order for this to happen, 3D points from the depth maps must be selected, retriangulation must be performed and the texture must be chosen and integrated to avoid a patchy effect. If a user wants to reconstruct a scene in 3D he will rarely have the opportunity to record it in one sequence and will typically upload several sequences, looking at different parts of the scene. Because every sequence is reconstructed in its own metric frame, a registration step will have to be performed first in order to bring all reconstructions together. This should be done as automatically as possible. A coarse registration step could be done by computing invariant signatures on the 3D geometry of different patches and matching these. Because the recent image features are much more stable under wide-baseline conditions, it can be envisaged to extract and match them in images across sequences. Perhaps a pre-warping technique to compute the feature vectors on fronto-parallel patches could be used [104]. The fine registration afterwards could be a variant of ICP. After registration, the different patches from the different sequences need to be integrated as well. Integration of both geometry and texture will be necessary.

Thanks to the recent powerful image features we can handle larger baseline situations. However, we still rely on the fact that all images in an uploaded set come from the same camera with approximately the same settings. It could be worth investigating whether we can relate images of the same scene taken by different people with different cameras at different times, rather like the appli-

cation of Snavely e.a. [82]. Databases exist with ordered information, including varieties of images, of many historical artefacts. One could mine these databases in a background process, constantly trying to create 3D reconstructions from them. The current self-calibration algorithm, however, is not designed for this since it requires approximately the same camera parameters or at least the same image size. Maybe the EXIF tags, present in most jpegs nowadays, could be of assistance here.

A promising application we are currently investigating is the use of SaM for mobile mapping. The recent boom of GPS-based navigation systems has led to a rise in demand for accurate models of streets and traffic infrastructure. Also, many governments, including the Flemish, have started the construction of a digital reference map of all streets and houses, including traffic signs, sewer lids, street markings, etc. The traditional way of measuring these is with total stations and theodolites, a very time-consuming and hence expensive job. A possible solution is to drive a car through the streets to be modeled and record the surroundings with one or more cameras. Using a few control points, one can bring the reconstruction into the desired coordinate system and measure the position of the landmarks with a simple click in the images. In order for this system to work, several aspects have to be investigated. First we need to obtain a better understanding of which errors are made at every step of the pipeline and what their impact is on the final accuracy of the system. The uncertainty of all reconstruction parameters should be estimated, so one can arrive at an uncertainty estimation per pixel. Actually, knowledge of this uncertainty could also be beneficial for the registration and integration algorithms we mentioned before. We could assign weight factors to different parts of the reconstructed parts that we want to register or integrate. In the case of the mobile mapping application, we need to investigate how other measuring devices can help us in our calibration. Differential GPS systems are cheap but take some time to come to a measurement and are less suitable in cities where the narrow streets and high buildings block the line of sight to many satellites. INS systems are better and come in several forms, from very cheap to extremely expensive. A combination of an initialization from the measurement of a -cheap- INS system and the recorded images as a refinement could yield sufficiently accurate results.

Bibliography

[1] H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[2] John Aldrich. R. A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, 1997.

[3] Marc Alexa. Linear combination of transformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 380–387, New York, NY, USA, 2002. ACM Press.

[4] L. Alvarez, R. Deriche, J. Sanchez, and J. Weickert. Dense disparity map estimation respecting image discontinuities: a pde and scalespace based approach. Technical Report RR-3874, INRIA, 2000.

[5] R. Ariew. *Ockham's Razor: A Historical and Philosophical Analysis of Ockham's Principle of Parsimony*. PhD thesis, University of Illinois, 1976.

[6] Cleve Ashcraft and Roger Grimes. SPOOLES: An object-oriented sparse matrix library. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, March 22–27 1999. `http://www.netlib.org/linalg/spooles/`.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision – ECCV 2006*, volume 3951 of *LNCS*, pages 404–417. Springer, 2006.

[8] R E Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[9] J.Y. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[10] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. *Proc, SIGGRAPH 2001*, pages 425 – 432, 2001.

[11] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[12] J-X. Chai, X. Tong, S-C. Chan, and H-Y. Shum. Plenoptic sampling. *Proc. SIGGRAPH 2000*, pages 307–318, 2000.

[13] Peter I. Corke. *Visual Control of Robots: High Performance Visual Servoing*. Research Studies Press (John Wiley), 1996.

[14] Kurt Cornelis, Frank Verbiest, and Luc Van Gool. Drift detection and removal for sequential structure from motion algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1249–1259, October 2004.

[15] Nico Cornelis and Luc Van Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *CVPR (1)*, pages 1099–1104, 2005.

[16] Stefaan De Roeck, Nico Cornelis, and Luc Van Gool. Augmenting fast stereo with silhouette constraints. In *Proc. of ICPR 2006*, 2006.

[17] Korry Douglas and Susan Douglas. *PostgreSQL (2nd Edition)*. Sams Publishing, 2005.

[18] Alan Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Math. Comput.*, 64(210):763–776, 1995.

[19] Alexander Enzmann, Lutz Kretzschmar, and Chris Young. *Ray Tracing Worlds With Pov-Ray*. Waite Group Pr, 1994.

[20] Jan-Friso Evers-Senne and Reinhard Koch. Image based interactive rendering with view dependent geometry. In *Proc. Eurographics 2003*, Computer Graphics Forum, pages 573–582. Eurographics Association, 2003.

[21] O. Faugeras and B. Mourrain. On the geometry and algebra of the point and line correspondences between n images. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 951, Washington, DC, USA, 1995. IEEE Computer Society.

[22] Olivier Faugeras and Renaud Keriven. Complete dense stereovision using level set methods. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 379–393, London, UK, 1998. Springer-Verlag.

[23] Olivier Faugeras, Quang-Tuan Luong, and T. Papadopoulou. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA, 2001.

[24] Olivier Faugeras and Steven Maybank. Motion from point matches: multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, 1990.

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

[26] Antoine Fuqua. King arthur. `http://www.bvimovies.com/kingarthur/`, 2004.

[27] Pau Gargallo and Peter Sturm. Bayesian 3D modeling from images using multiple depth maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, California*, volume 2, pages 885–891, jun 2005.

[28] Indra Geys and Luc Van Gool. On-line novel view synthesis capable of handling multiple moving objects. In *ICCV-HCI*, pages 150–159, 2005.

[29] Terry Gilliam. The brothers grimm. `http://video.movies.go.com/brothersgrimm/main.html`, 2005.

[30] Andrew S. Glassner. *Graphics Gems*. Academic Press, Inc., Orlando, FL, USA, 1990.

[31] M. P. Golombek, et al. Overview of the Mars Pathfinder mission and assessment of landing site predictions. *Science*, 5344(278):1743–1748, December 1997.

[32] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 1983.

[33] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. *Proc. SIGGRAPH 1996*, pages 43–54, 1996.

[34] J.A. Grunert. Das Pothenotische Problem in erweiterter Gestalt nebst Ueber seine Anwendungen in der Geodäsie. *Grunerts Archiv für Mathematik und Physik, Band 1*, pages 238–248, 1841.

[35] Robert M. Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Noelle. Review and analysis of solutions of the three point perspective pose estimation problem. *Int. J. Comput. Vision*, 13(3):331–356, 1994.

[36] Chris Harris and Mike Stephens. A combined corner and edge detector. In M. M. Matthews, editor, *Proceedings of the 4th ALVEY vision conference*, pages 147–151, University of Manchester, England, September 1988.

[37] John W. Harris and Horst Stocker. *The Handbook of Mathematics and Computational Science*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[38] Richard Hartley. Cheirality. *International Journal of Computer Vision*, 26(1):41–61, 1998.

[39] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[40] K. Hashimoto. *Visual Servoing*. World Scientific, 1993.

[41] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *DAGM'99*, pages 94–101, 1999.

[42] W. Hofmann. *Das problem der Gefährlichen Flächen in Theorie und Praxis -Ein Beitrag zur Hauptaufgabe der Photogrammetrie*. PhD thesis, Fakultät für Bauwesen, Technische Universität München, Germany, 1953.

[43] Tim Janssens. Simulator voor planetair verkenningsvoertuig. Master Thesis Electrical Engineering, K.U. Leuven, 2000.

[44] Hailin Jin, Stefano Soatto, and Anthony Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63(3):175–189, July 2005.

[45] Fredrik Kahl, Bill Triggs, and Kale Aastrom. Critical motions for autocalibration when some intrinsic parameters can vary. *Journal of Mathematical Imaging and Vision*, 13(2):131–146, October 2000.

[46] Susan Kitchens. *The QuickTime VR Book : Creating Immersive Imaging on Your Desktop*. Addison Wesley Longman, 1998.

[47] Reinhard Koch, Marc Pollefeys, and Luc Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 55–71, London, UK, 1998. Springer-Verlag.

[48] Reinhard Koch, Marc Pollefeys, Benno Heigl, Luc Van Gool, and Heinrich Niemann. Calibration of hand-held camera sequences for plenoptic modeling. In *ICCV (1)*, pages 585–591, 1999.

[49] Michael Kofler. *Maple: An Introduction and Reference*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[50] E. Kruppa. Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung. *Kaiser- licher Akademie der Wissenschaften (Wien) Kl. Abt*, IIa(122):1939–1948, 1913.

[51] Kiriakos Kutulakos and Steven Seitz. A theory of shape by space carving. *Int. J. Comput. Vision*, 38(3):199–218, 2000.

[52] K. Landzettel, et al. A micro-rover navigation and control system for autonomous planetary exploration. *Advanced Robotics, Special Issue on Service Robots in Space*, 18(3):285–314, April 2004.

[53] Stéphane Laveau and Olivier D. Faugeras. Oriented projective geometry for computer vision. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 147–156, London, UK, 1996. Springer-Verlag.

[54] M. Levoy and P. Hanrahan. Lightfield rendering. *Proc. SIGGRAPH 1996*, pages 31–42, 1996.

[55] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[56] H. C. Longuet-Higgins. The reconstruction of a plane surface from two perspective projections. In *Proceedings of Royal Society of London*, volume 227 of B, pages 399–410, 1986.

[57] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. Technical Report MSR-TR-99-21, Microsoft Research, 1999.

[58] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[59] Stephen Maybank. *Theory of Reconstruction from Image Motion*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1992.

[60] H. Michaelis, T. Behnke, M. Tschentscher, H.-G. Grothues, and S. Mottola. ROLIS: A small scientific camera system for the ROSETTA-lander. In *Proc. 4th International Conference on Scientific Optical Imaging*, 1999.

[61] Florica Mindru, Theo Moons, and Luc Van Gool. Recognizing color patterns irrespective of viewpoint and illumination. In *CVPR*, pages 1368–1373, 1999.

[62] Theo Moons. A guided tour through multiview relations. In *SMILE'98: Proceedings of the European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, pages 304–346, London, UK, 1998. Springer-Verlag.

[63] D. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. In *CGC 2nd Annual Fall Workshop on Computational Geometry*, 1997.

[64] S. Negahdaripour. Closed-form relationship between the two interpretations of a moving plane. *Journal of the Optical Society of America*, 7(2):279–285, 1990.

[65] David Nister. An efficient solution to the five-point relative pose problem. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 195–202. IEEE Computer Society Press, 2003.

[66] Open Inventor Architecture Group. *Open Inventor C++ Reference Manual*. Addison-Wesley, 1994.

[67] Renato Pajarola, Miguel Sainz, and Yu Meng. Depth-mesh objects: Fast depth-image meshing and warping. Technical Report 03-02, UCI-ICS, 2003.

[68] Johan Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *The Photogrammetric Record*, 15(88):589–599, 1996.

[69] Daniël Pletinckx, Neil Silberman, and Dirk Callebaut. The Saint Laurentius church in Ename and its role in the Francia Media Heritage Initiative. In *Proc. VAST 2001, Virtual Reality, Arachaeology and Cultural Heritage*, pages 197–204. ACM Siggraph, 2002.

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

[71] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. A simple and efficient rectification method for general motion. In *Proc. of Intl. Conference on Computer Vision 1999*, pages 496–501, 1999.

[72] Jean-Philippe Pons, Renaud Keriven, and Olivier D. Faugeras. Modeling dynamic scenes by registering multi-view image sequences. In *CVPR (2)*, pages 822–827, 2005.

[73] R. Rieder, et al. Nanokhod, a miniature deployment device with instrumentation for chemical, mineralogical and geological analysis of planetary surfaces, for use in connection with fixed planetary surface stations. *Lunar and Planetary Science*, XXVI:1261–1262, 1995.

[74] J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.

[75] Will Schroeder, Kenneth M. Martin, and William E. Lorensen. *The visualization toolkit (2nd ed.): an object-oriented approach to 3D graphics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[76] K. Schröer. *Handbook on Robot Performance Testing and Calibration (IRIS-project)*. Fraunhofer IRB Verlag, 1998.

[77] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[78] Y.C. Shiu and S. Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneoustransform equations of the form ax=xb. *IEEE Transactions on Robotics and Automation*, 5:16 – 29, 1989.

[79] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92*, pages 258–264, 1992.

[80] Gregory G. Slabaugh, W. Bruce Culbertson, Thomas Malzbender, Mark R. Stevens, and Ronald W. Schafer. Methods for volumetric reconstruction of visual scenes. *Int. J. Comput. Vision*, 57(3):179–199, 2004.

[81] C. Slama. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, USA, 4th edition, 1980.

[82] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo Tourism: exploring photo collections in 3D. *ACM Trans. Graph.*, 25(3):835–846, 2006.

[83] Christoph Strecha, Rik Fransens, and Luc Van Gool. Wide-baseline stereo from multiple views: A probabilistic account. In *Proc. CVPR 2004*, pages 552–559, 2004.

[84] Christoph Strecha, Rik Fransens, and Luc Van Gool. Combined depth and outlier estimation in multi-view stereo. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2394–2401, 2006.

[85] Christoph Strecha, Tinne Tuytelaars, and Luc Van Gool. Dense matching of multiple wide-baseline views. In *Proc. 2003 International Conference on Computer Vision (2)*, pages 1194–1201, 2003.

[86] Peter Sturm. Critical motion sequences and conjugacy of ambiguous Euclidean reconstructions. In M. Frydrych, J. Parkkinen, and A. Visa, editors, *Proceedings of the 10th Scandinavian Conference on Image Analysis, Lappeenranta, Finland*, volume I, pages 439–446, June 1997.

[87] Peter Sturm. Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 1100–1105, Washington, DC, USA, 1997. IEEE Computer Society.

[88] Peter Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. In *British Machine Vision Conference, Nottingham, England*, pages 63–72, Sep 1999.

[89] T. Svoboda, H. Hug, and L. Van Gool. ViRoom — low cost synchronized multicamera system and its self-calibration, 2002.

[90] David S. Taubman and Michael W. Marcellin. *Jpeg2000: Image Compression Fundamentals, Standards, and Practice*. Kluwer Academic Publishers, 2001.

[91] Camillo Taylor. Minimization on the Lie group SO(3) and related manifolds. Technical report, Dept. of Electrical Engineering Yale University, 1993.

[92] Phil Torr. An assessment of information criteria for motion model selection. In *Proc. CVPR97*, pages 47–53, 1997.

[93] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: A modern synthesis. *Vision Algorithms: Theory and Practice, LNCS*, 1883:298–372, 2000.

[94] Bill Triggs. Autocalibration and the absolute quadric. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 609–614, Washington, DC, USA, 1997. IEEE Computer Society.

[95] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *Radiometry*, pages 221–244, 1992.

[96] Roger Y. Tsai and R.K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5:345–358, 1989.

[97] Tinne Tuytelaars. *Local, invariant features for registration and recognition*. PhD thesis, K.U. Leuven, 2000.

[98] M. Van Diest, L. Van Gool, T. Moons, and E. Pauwels. Projective invariants for planar contour recognition. In *Proc. ECCV*, pages 527–534. Springer-Verlag, 1994.

[99] Luc Van Gool. *Computer Vision I*. K.U.Leuven, Belgium, 2000.

[100] Geert Van Meerbergen, Maarten Vergauwen, Marc Pollefeys, and Luc Van Gool. A hierarchical symmetric stereo algorithm using dynamic programming. *International Journal on Computer Vision*, 47(1):275–285, April 2002.

[101] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proc. CVPR 2005*, pages 391–398, 2005.

[102] C. C. Wang. Extrinsic calibration of a vision sensor mounted on a robot. *IEEE Transactions on Robotics and Automation*, 8:161–175, 1992.

[103] Brent Welch, Ken Jones, and Jeffrey Hobbs. *Practical Programming in Tcl & Tk*. Prentice Hall Professional Technical Reference, 2003.

[104] Geert Willems. Matching of textured 3D-range data using sift features. Master Thesis Artificial Intelligence, K.U. Leuven, 2005.

[105] Geert Willems, Frank Verbiest, Maarten Vergauwen, and Luc Van Gool. Real-time image based rendering from uncalibrated images. In *Fifth International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 221–228, Ottawa, Ontario, Canada, June 2005. IEEE Computer Society.

[106] Woo-Keun Yoon, Toshihiko Goshozono, Hiroshi Kawabe, Masahiro Kinami, Yuichi Tsumaki, Masaru Uchiyama, Mitsushige Oda, and Toshitsugu Doi. Model-based teleoperation of a space robot on ets-vii using a haptic interface. In *Proc. IEEE International Conference on Robotics and Automation 2001*, pages 407–412, 2001.

[107] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing Journal*, 15(1):59–76, 1997.

[108] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *ICCV*, pages 666–673, 1999.

Appendix A

Passive 3D Reconstruction: Basics

A.1 Introduction

In this chapter we will discuss some important basic principles of computer vision that need to be mastered to understand the techniques of 3D reconstruction. More specifically we will start by explaining the image formation process by means of camera models. Both linear and non-linear effects are discussed. A stratification of 2D and 3D geometry is given. Then we will explain how cameras can be calibrated with standard calibration procedures using known calibration objects. At the end of this chapter we will discuss the simplest passive 3D reconstruction setup: a stereo pair.

A.2 Homogeneous Coordinates

Computer scientists often use homogeneous coordinates in their work. While not strictly necessary in the Euclidean world, they make the notation of the equations simpler. In the projective world, homogeneous coordinates are inevitable. The idea behind homogeneous coordinates is to make life simpler. Because of the extra coordinate, transformations of entities (points, lines, conics, quadrics, ...) can be expressed as simple matrix multiplications.

A.2.1 Points

In order to express the coordinates of a point in the plane or in the world, one needs 2 or 3 coordinates respectively. In $\mathbb{R}^2$ and $\mathbb{R}^3$ we can write

$$\mathtt{m}_\mathtt{e} = \begin{pmatrix} x \\ y \end{pmatrix}, \mathtt{M}_\mathtt{e} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}.$$

Expressing the same points in homogeneous coordinates means that we add a third or fourth coordinate to the points respectively, so in $\mathbb{R}^2$ and $\mathbb{R}^3$ the points become

$$\begin{aligned} \mathtt{m}_\mathtt{p} &= \begin{pmatrix} x_h \\ y_h \\ w_h \end{pmatrix} = \begin{pmatrix} ax_h \\ ay_h \\ aw_h \end{pmatrix} \\ \mathtt{M}_\mathtt{p} &= \begin{pmatrix} X_h \\ Y_h \\ Z_h \\ W_h \end{pmatrix} = \begin{pmatrix} aX_h \\ aY_h \\ aZ_h \\ aW_h \end{pmatrix} (a \neq 0). \end{aligned}$$

where we see that a point in homogeneous coordinates, multiplied with any non-zero scale-factor remains the same point.

To write down the homogeneous coordinates of a Euclidean point, we simply add a 1 as the last coordinate. To retrieve the Euclidean coordinates of a point in homogeneous coordinates, we divide by the last homogeneous coordinate to make it 1. This only yields meaningful results if this coordinate is not zero.

$$\begin{aligned} \mathtt{m}_\mathtt{p} &= \begin{pmatrix} x_h \\ y_h \\ w_h \end{pmatrix} = \begin{pmatrix} x_h/w_h \\ y_h/w_h \\ w_h/w_h \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} (w_h \neq 0) \\ \mathtt{M}_\mathtt{p} &= \begin{pmatrix} X_h \\ Y_h \\ Z_h \\ W_h \end{pmatrix} = \begin{pmatrix} X_h/W_h \\ Y_h/W_h \\ Z_h/W_h \\ W_h/W_h \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} (W_h \neq 0) \end{aligned}$$

A.2.2 Infinity

If the last coordinate of a homogeneous point is 0 this point can not be represented in Euclidean space. In projective space, these points are said to be located *at infinity*. In $\mathbb{R}^2$ all points with a zero third homogeneous coordinate form the line at infinity. In $\mathbb{R}^3$, the points with zero fourth coordinate form the plane at infinity. The use of points at infinity will become clear in the next paragraph which deals with lines.

A.2.3 Lines

Lines can of course also be expressed in homogeneous coordinates. A line can be described by two points $\mathtt{m}_1$ and $\mathtt{m}_2$ on it. Any point $\mathtt{m}$ on the line can be expressed by starting at point $\mathtt{m}_1$ and traveling a certain distance λ in the direction $\mathtt{m}_2 - \mathtt{m}_1$. In 2D we write this direction as:

$$\mathtt{m}_2 - \mathtt{m}_1 = \begin{pmatrix} w_1x_2 - w_2x_1 \\ w_1y_2 - w_2y_1 \\ w_1w_2 - w_2w_1 \end{pmatrix} = \begin{pmatrix} w_1x_2 - w_2x_1 \\ w_1y_2 - w_2y_1 \\ 0 \end{pmatrix}$$

(note that before one can add or subtract two homogeneous points, one has to make sure their last coordinate is the same).

In 3D, the equivalent equation can be written down. It is now clear that the direction of the line is a point at infinity. In other words, all points at infinity describe directions of lines. Also, parallel lines have the same direction, so all parallel lines coincide in the same point at infinity.

For 2D lines, an elegant representation of a line can be used which expresses the line as a 3×1 vector. We start from the Euclidean expression of a line

$$ax + by + c = 0$$

This homogeneous equation is unaffected by scaling with any scale-factor w

$$\begin{aligned} awx + bwy + cw &= 0 \\ ax_h + by_h + cw_h &= 0 \end{aligned}$$

The line l can thus be expressed by the vector $l = (a\ b\ c)^T$. The fact that a point $m = (x\ y\ 1)^T = (x_h\ y_h\ w_h)^T$ lies on the line l is easily expressed by enforcing the scalar product between the point and the line vectors to be zero:

$$\begin{aligned} (a\ b\ c) \begin{pmatrix} x_h \\ y_h \\ w_h \end{pmatrix} &= 0 \\ \mathtt{l}^\mathtt{T}\mathtt{m} &= 0 \end{aligned}$$

Thus we see that points and lines have the same representation in the projective plane.

We know that a line can be defined as going through 2 points. If the coordinates of these two points $\mathtt{m}_1$ and $\mathtt{m}_2$ are given, the vector $\mathtt{l}$ describing the line satisfies

$$\begin{aligned} \mathtt{l}^\mathtt{T}\mathtt{m}_1 &= 0 \\ \mathtt{l}^\mathtt{T}\mathtt{m}_2 &= 0 \end{aligned}$$

In other words we search for the vector $\mathtt{l}$ which is orthogonal to both vectors $\mathtt{m}_1$ and $\mathtt{m}_2$. We know that the cross product of these two vectors is indeed a vector orthogonal to the vectors, hence

$$\mathtt{l} = \mathtt{m}_1 \times \mathtt{m}_2$$

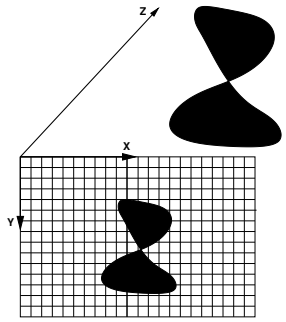


Figure A.1: Image coordinates

or

$$\mathtt{l} = [\mathtt{m}_1]_\times \mathtt{m}_2 \tag{A.1}$$

with $[a]_\times$ the skew-symmetric matrix defined as

$$[a]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{A.2}$$

where the multiplication of the skew-symmetric matrix created from vector $\mathtt{m}_1$ with vector $\mathtt{m}_2$ indeed delivers the cross-product vector $\mathtt{m}_1 \times \mathtt{m}_2$.

A.3 Camera Models

If we want to understand how to reconstruct 3D objects from images, we first need to know how the reverse process works: how is the world transformed by a camera into images. It is clear that this transformation is one from the 3D world ($\mathbb{R}^3$) to the 2D world ($\mathbb{R}^2$). For ease of notation and computation, we employ homogeneous coordinates to denote these transformations.

A.3.1 Image Coordinates

Before we start with the derivation of the pinhole camera model, a quick note about image coordinates is due. In Computer Graphics, image coordinates are typically given in a coordinate system with the origin in the bottom left, an horizontal X-axis pointing right and a vertical Y-axis pointing up. Computer Vision researchers prefer the coordinate system of Fig. A.1. The origin lies at the top left, the Y-axis points down and the X-axis to the right. This setup has several advantages:

- The way X- and Y-coordinates are given coincides with the way an image is read out by a digital camera with a CCD: starting at the top left and reading line by line.

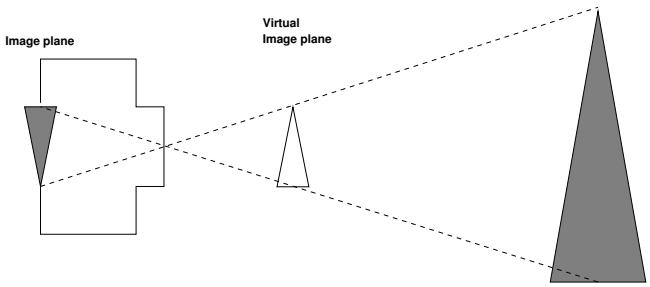


Figure A.2: The *camera obscura* inspired the pinhole model

- The Z-coordinate is pointing away from the image into the world if the coordinate system is to be right-handed. This means it coincides with the depth of a point w.r.t. the camera. Conceptually this is nice because the depth is the typical unknown in 3D reconstruction problems.

A.3.2 The Linear Pinhole Model

The camera model that will be used throughout this text is the linear pinhole model. This camera model is inspired by the *camera obscura*. In its simplest form the camera obscura is no more than a black box of which one side is punctured to yield a small hole. On the opposite side, photosensitive paper is attached. Since light travels in straight lines, all light rays going through the hole and falling on the paper will come directly from the 3D world outside the box. On this paper it forms a *negative* image of the world as can be seen in Fig. A.2. The amount of light that falls into the box through the small hole is very limited. One can increase this by making the hole bigger but then rays coming from different 3D points can fall onto the same point on the image. One way of getting around this problem is by making use of lenses which focus the light. Complicated formulas exist that model the effect of lenses but in essence the behavior can still be linearized to yield the same pinhole model we will discuss. As mentioned, the camera obscura creates a negative image of the world behind the camera-opening. This complicates the mathematical description of the camera. We call the camera-opening the camera center. Let us imagine a *virtual image plane* in front of the camera center at the same distance as the real image plane behind it. The light rays coming from the world and falling onto the camera center intersect this virtual image plane and form a positive image which is clearly shown in Fig. A.2.

Simple Setup

The mathematical set-up of the pinhole model is shown in Fig. A.3. It shows a 3D point `M`. To compute the projection of this point in the image, we connect the camera center `C` with `M` and intersect the image plane with this ray, yielding

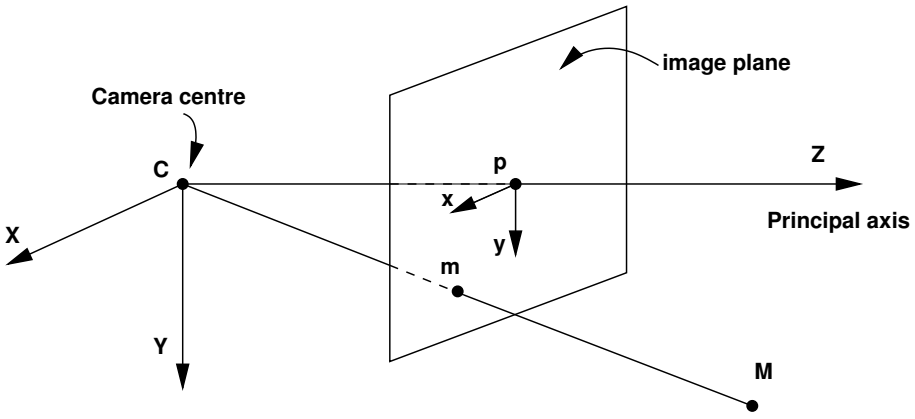


Figure A.3: The pinhole camera model

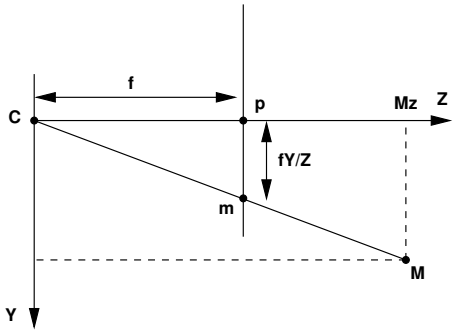


Figure A.4: The pinhole setup, viewed from the direction of the X-axis

the point m. For the time being we assume that the camera center coincides with the origin of the world. The X- and Y-axes are parallel to the image plane and the Z-axis is the principal axis (the line orthogonal to the image plane and through the camera center) and denotes the depth of a point w.r.t. the camera center. The coordinates in the image are for the moment still expressed w.r.t. the point p which is the intersection of the image plane with the principal axis.

If we view the pinhole setup of Fig. A.3 from the direction of the X-axis, we obtain Fig. A.4. The image plane is reduced to a line and the distance between this plane and the camera center is f, the *focal length*. The 3D point M has coordinates $(X, Y, Z)^T$ and we define the 3D point $\mathtt{M_z}$ as the point on the Z-axis with coordinates $(0, 0, Z)^T$. From congruent triangles $\mathtt{M\text{-}M_z\text{-}C}$ and m-p-C we see that the y-coordinate of m in the image plane is $\frac{fY}{Z}$, with Y and Z Euclidean coordinates of M. The x coordinate of m is found analogously from the Y-axis view as $\frac{fX}{Z}$. This means that in this setup, the 3D point $(X, Y, Z)^T$ is projected onto the 2D point $(fX/Z, fY/Z)^T$ or in homogeneous coordinates $(fX, fY, Z)^T$.

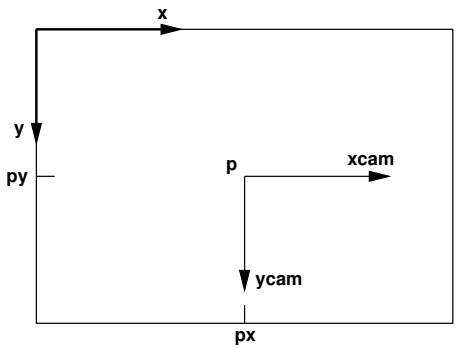


Figure A.5: The coordinates of the principal point must be taken into account

The projection can hence be written in the form of a matrix multiplication as

$$\mathtt{m} \simeq \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{A.3}$$

Eq. A.3 needs to be read from right to left. The 3D point `M` is projected through a projection matrix with 3x4 elements onto the 2D point `m`. All projection equations (of which Eq. A.3 is a simplified version) have this form.

Principal Point Offset

As explained in A.3.1 we prefer the origin of the image coordinate system to be at the top left of the image. This means we have to take into account the offset given by the *principal point*. This is the intersection point of the principal axis (Z) and the image plane. If the coordinates of this point are given by (p_x and p_y) as shown in Fig. A.5, then the projection of the point `M` is $(fX/Z + p_x, fY/Z + p_y)^T$ and the projection equation becomes

$$\mathtt{m} \simeq \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Pixel Coordinates

The coordinates of the point `m` are still expressed in the units of the world coordinate system (meter, millimeter, ...). Since images are nothing more than arrays of pixels, measurements in images are done in pixel coordinates. We therefore want to transform the coordinates to pixels. This is done by multiplication with a matrix which holds the conversion from world-units to pixels:

$$\mathbf{T}_{world}^{pixel} = \begin{pmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where m_1 and m_2 are the scale factors that hold how many pixels go into one world unit. The projection matrix now becomes

$$\begin{pmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & u_x & 0 \\ 0 & \alpha_y & u_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

with α_x and α_y the focal length in pixels for the x and y direction and u_x and u_y the principal point in pixels. In fact there is one more linear term that can be introduced here. Analogue cameras are known to sometimes suffer from *skew*, i.e. the fact that pixels are parallelograms and not rectangles. We can model this by adding a skew factor in the equation.

$$\mathtt{m} \simeq \begin{pmatrix} \alpha_x & s & u_x & 0 \\ 0 & \alpha_y & u_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

This skew-factor s is inversely proportional to the tangent of the angle between the X and Y axis. Digital cameras have rectangular pixels, hence their skew is always zero. The parameters α_x, α_y, s, u_x and u_y define the internal behavior of the camera and are called the *intrinsic parameters* or *intrinsics* of the camera.

Camera Rotation and Translation

Until now we assumed that the camera coordinates coincided with the world coordinates. We want to be able to model cameras anywhere in the world, i.e. with a known translation and rotation w.r.t. the world coordinate system. If the rotation and translation are given by a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 translation vector $\mathbf{t}$ respectively, then the transformation of the coordinate system of the camera in the world is given by

$$\mathbf{T}_{cam}^{world} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

Since any rotation matrix is orthonormal, the inverse transformation is given by

$$(\mathbf{T}_{cam}^{world})^{-1} = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

This means that a point M with coordinates $(X, Y, Z, 1)^T$ in the world coordinate frame, can be expressed in the camera coordinate frame by multiplying it with this inverse matrix.

$$\mathtt{M}_{\mathtt{cam}} = (\mathbf{T}_{cam}^{world})^{-1}\mathtt{M} = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \mathtt{M}$$

Hence, the projection equation becomes

$$\mathtt{m} \simeq \begin{pmatrix} \alpha_x & s & u_x & 0 \\ 0 & \alpha_y & u_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Since the fourth column of the first 3×4 matrix is 0, the fourth row of the matrix with the geometric transformation is of no importance. Bundling the intrinsics in a 3×3 matrix $\mathbf{K}$,

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & u_x \\ 0 & \alpha_y & u_y \\ 0 & 0 & 1 \end{pmatrix} \tag{A.4}$$

the projection equation becomes

$$\begin{aligned} \mathtt{m} &\simeq \mathbf{K}\left(\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right)\mathtt{M} \\ &\simeq \mathbf{P}\mathtt{M} \end{aligned} \tag{A.5}$$

Decomposition of P

If only the 3×4 projection matrix $\mathbf{P}$ is known, it is possible to retrieve the intrinsic and extrinsic parameters from it. This process is called *decomposing* $\mathbf{P}$ into $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$ and is done as follows. Inspect the upper left 3×3 sub-matrix of $\mathbf{P}$. This matrix is formed by multiplying $\mathbf{K}$ and $\mathbf{R}^T$. The inverse of this matrix is $\mathbf{R}\mathbf{K}^{-1}$. The QR-decomposition of a 3×3 matrix of rank 3 decomposes the matrix into two matrices: an orthonormal matrix on the left and an upper triangular matrix on the right. This is exactly the decomposition we are looking for since $\mathbf{R}$ is a rotation matrix and thus orthonormal and $\mathbf{K}$ is by definition an upper triangular matrix, so $\mathbf{K}^{-1}$ is as well. When $\mathbf{K}$ and $\mathbf{R}$ are known, $\mathbf{t}$ is easily computed by pre-multiplying the fourth column of $\mathbf{P}$ with $-\mathbf{R}\mathbf{K}^{-1}$.

A.3.3 Non-Linear Distortions

The projection model we described in A.3.2 is linear which means it can only model linear effects. Images taken by real cameras experience non-linear deformations or distortions which make the simple linear pinhole model inaccurate. The most important and best known non-linear distortion is *radial distortion*. Figure A.6 shows an example of a radially distorted image. Radial distortion is an alteration in magnification from a center to any point in the image, measured in a radial direction from the center of distortion. In other words, the larger



Figure A.6: An example of a radially distorted image

the distance between an image point and the center of distortion, the larger the effect of the distortion. Thus, the effect of the distortion is mostly visible near the edges of the image. This can clearly be seen in Fig. A.6. Straight lines near the edges of the image are no longer straight but are bent. For practical use the center of radial distortion can be assumed to coincide with the principal point, which usually also coincides with the center of the image.

Radial distortion is a non-linear effect and is typically modeled using a Taylor expansion. It can be shown that only the even order terms play a role in this expansion, i.e. the effect is symmetric around the center. The effect takes place in the lens, hence mathematically the radial distortion should be between the external and internal parameters of the pinhole model. The model we propose here follows this strategy. Let us define

$$\mathtt{m_u} = \begin{pmatrix} m_{ux} \\ m_{uy} \\ m_{uw} \end{pmatrix} \simeq \left(-\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

The distance r from the optical axis is then

$$r^2 = \left(\frac{m_{ux}}{m_{uw}}\right)^2 + \left(\frac{m_{uy}}{m_{uw}}\right)^2$$

We now define $\mathtt{m_d}$ as

$$\mathtt{m_d} = \begin{pmatrix} m_{dx} \\ m_{dy} \\ m_{dw} \end{pmatrix} \simeq \begin{pmatrix} \left(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \ldots\right) m_{ux} \\ \left(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \ldots\right) m_{uy} \\ m_{uw} \end{pmatrix} \tag{A.6}$$

The lower order terms of this expansion are the most important ones and typically one does not compute more than three parameters (κ_1, κ_2, κ_3). Finally



Figure A.7: Distorted image (left) and undistorted image (right)

the projection $\mathtt{m}$ of the 3D point $\mathtt{M}$ is

$$\mathtt{m} \simeq \mathbf{K}\mathtt{m}_\mathtt{d} \tag{A.7}$$

When the distortion parameters are known, the image can be undistorted in order to make all lines straight again and thus make the linear pinhole model valid. The original and undistorted version of Fig. A.6 are shown in Fig. A.7.

The model described above puts the radial distortion parameters between the extrinsic and linear intrinsic parameters of the camera. In literature one often finds that the distortion is put on the left of the intrinsics, i.e. a 3D point is first projected into the image via the linear model and then undistorted. Conceptually the latter model is less suited than the one used here because putting the distortion at the end makes it dependent on the intrinsic parameters, especially on the focal length. This means one has to re-estimate the radial distortion parameters every time the focal length changes. This is not necessary in our model. This however does not mean our model is perfect. In reality the center of radial distortion (now assumed to be the principal point) sometimes changes when the focal length is altered. This effect cannot be modeled with our model.

A.4 Deformation of Planar and 3D Objects

In this section we discuss deformations that planar (2D) and 3D objects can undergo. We only study linear deformations, i.e. those deformations that can be described by a multiplication with a 3×3 or a 4×4 matrix in the 2D and 3D case respectively.

A.4.1 Deformation of Planar Objects under Projection

The linear deformations of 2D objects that are described here are those that can be expected from different kinds of projections of these objects. We study

4 cases. The mathematical derivation of the transformation of every case can be found in [99]. Table A.1 gives an overview of the transformation matrices and the effect on the planar objects. The third column of the table shows the effect every transformation can have on a square. The four cases are enumerated below:

1. In case 1 the planar objects are viewed from a perpendicular direction and can only rotate about an axis parallel to this direction, any translation parallel to the object plane is allowed. If the distance between the object and the camera does not change, this case boils down to a *Euclidean* transformation with 3 DOF (1 for rotation, 2 for translation).

2. In case 2, a *similarity* or *metric* transformation is the result of the same setup but the translation can now have a perpendicular component. The extra scale factor in this transformation brings the number of DOF to 4.

3. In case 3 the planar objects can be viewed from any direction but should be at a sufficient distance from the camera so as to make the differences in depth (Z) values small relative to the distance to the camera. The result is an *affine* transformation with 6 DOF.

4. Finally we study the case where the planar objects can be viewed from any direction and can be placed at arbitrary distances from the camera. This gives rise to the *projective* transformation which is the most general. It has 8 DOF (9 elements - 1 of the global scale-factor).

The camera-object configurations become increasingly arbitrary going down this list and the corresponding geometries are of increasing complexity.

All transformations are written for homogeneous coordinates. The projective transformation for instance is defined as

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} \simeq \mathbf{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{A.8}$$

where $\mathbf{H}$ is often called a *planar homography*.

A.4.2 Stratification of Transformations of 3D Objects

A logical next step is to extend this discussion to general 3D objects and describe different transformations 3D objects can undergo. The analogy with the previous paragraph is not entirely valid because there we described transformations on planar objects when projected into a camera. For the transformations on 3D objects, this projection is not part of the discussion. We will not go through the derivation of the formulas. The different transformations for 3D objects are given by table A.2. A few notes:

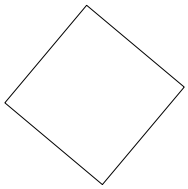
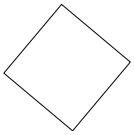
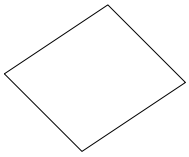
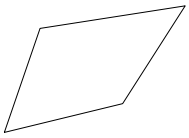
Name	**DOF**	**Transformation**	**Effect**
Euclidean	3	$\begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$	
Similarity	4	$\begin{pmatrix} k\cos\theta & -k\sin\theta & t_x \\ k\sin\theta & k\cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$ $= \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & \frac{1}{k} \end{pmatrix}$	
Affine	6	$\begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & b_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix}$	
Projective	8	$\begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}$	

Table A.1: Effect of different transformations on planar objects, illustrated by the effect on a square

- A similarity transformation is sometimes called a *metric* transformation.
- The 3×3 matrix on the upper left in the Euclidean and similarity transformation is a rotation matrix. These matrices are orthonormal and can be described by three parameters, e.g. three angles of rotation around the three axes. Hence the amount of degrees of freedom (DOG) for the Euclidean and metric transformations are 6 (3 rotation + 3 translation) and 7 (3 rotation + 3 translation + 1 scale) respectively.
- The 3×3 matrix on the upper left in the affine transformation is no longer orthonormal.
- A projective transformation is the most general transformation and has 15 degrees of freedom: 16-1 (homogeneous coordinates describe points up to a scale factor).

A.4.3 Transformations and the Projection Equation

What is the influence of the above mentioned transformations on the camera model we have derived? Let's revisit the projection equation of Eq. A.5. What happens to it if we transform the parts of this equation, e.g. with a projective transformation $\mathbf{T}$?

The 3D point $\mathtt{M}$ is transformed by multiplying it on the left with the matrix $\mathbf{T}$:

$$\mathtt{M}' = \mathbf{T}\mathtt{M}$$

The camera's extrinsics are transformed as well. The frame of the camera was given by the matrix

$$\mathbf{T}_{cam}^{world} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

which needs to be transformed with the transformation $\mathbf{T}$ by multiplying it on the left.

$$\mathbf{T}_{cam}^{world'} = \mathbf{T} \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$$

However, as explained in Eq. A.3.2, A.3.2 the camera equation needs the inverse of $\mathbf{T}_{cam}^{world}$. The transformation of this matrix is

$$(\mathbf{T}_{cam}^{world'})^{-1} = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \mathbf{T}^{-1}$$

Hence, the transformed projection matrix becomes

$$\mathbf{P}' = \mathbf{K} \begin{pmatrix} \mathbf{R}^T | -\mathbf{R}^T\mathbf{t} \end{pmatrix} \mathbf{T}^{-1}$$

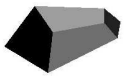
Name	DOF	Transformation	Effect
Euclidean	6	$\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$	
Similarity	7	$\begin{pmatrix} kr_{11} & kr_{12} & kr_{13} & t_x \\ kr_{21} & kr_{22} & kr_{23} & t_y \\ kr_{31} & kr_{32} & kr_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & \frac{1}{k} \end{pmatrix}$	
Affine	12	$\begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	
Projective	15	$\begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{pmatrix}$	

Table A.2: Effect of different transformations on 3D objects, illustrated by the effect on a cube.

This means that the projection equation in the transformed world reads:

$$\begin{aligned} \mathtt{m}' &\simeq \mathbf{P}'\mathtt{M}' \\ &\simeq \mathbf{K}\left(\mathbf{R}^T| - \mathbf{R}^T\mathbf{t}\right)\mathbf{T}^{-1}\mathbf{T}\mathtt{M} \\ &\simeq \mathbf{K}\left(\mathbf{R}^T| - \mathbf{R}^T\mathbf{t}\right)\mathtt{M} \\ &\simeq \mathtt{m} \end{aligned}$$

Thus, the projection of the transformed 3D point through the transformed camera yields the same 2D point as before. In the projective world (the real world transformed with an arbitrary projective transformation), any 3×4 matrix can be seen as a projection equation

$$\mathtt{m} \simeq \mathbf{P}\mathtt{M}$$

where $\mathbf{P}$ can no longer be decomposed meaningfully in a multiplication of intrinsic and extrinsic matrices.

Nederlandse samenvatting

1 Inleiding

In het voortdurend streven om zijn bestaan te verbeteren heeft de mensheid geleerd hoe belangrijk het bekomen van informatie over de wereld die ons omringt wel is. Over de eeuwen heen werden manieren ontwikkeld om nauwkeurige metingen van objecten te bekomen. Als we bijvoorbeeld 3D posities van objecten willen meten, zijn verschillende oplossingen mogelijk: actieve systemen, zoals lasers, die gebaseerd zijn op het tijdsverschil tussen verzenden en ontvangen van een laserstraal of op triangulatie principes. Passieve systemen bestaan ook, zoals gecalibreerde stereo, of fotometrische opstellingen.

Reeds in de negentiende eeuw was het principe bekend dat 3D metingen mogelijk zijn op basis van beelden, opgenomen vanuit verschillende standpunten; het principe dat aan de basis ligt van de hierboven genoemde stereo opstelling. Een eerste uitbreiding bestaat erin om de opstelling met twee camera's te vervangen door één met slechts één enkele bewegende camera die een statische scène observeert.

Gedurende de laatste jaren hebben computer visie onderzoekers dit idee nog verder uitgebreid. Wat als er niets geweten is over de bewegende camera? Als noch de interne calibratie, noch de beweging van de camera gekend zijn? Het blijkt dat het dan toch nog mogelijk is om zowel de calibratie als de 3D structuur van de statische scène terug te vinden. Deze ongecalibreerde 3D reconstructie wordt in het Engels *Structure and Motion* genoemd. Om het leesgemak te verhogen zullen we in deze Nederlands samenvatting de Engelse afkorting SaM gebruiken als we ongecalibreerde 3D reconstructie bedoelen.

Hoewel de theorie van SaM ondertussen goed gekend is, zal een rechttoerechtaan implementatie ervan in de praktijk enkel goede resultaten opleveren voor een handvol, goed gekozen, vrij eenvoudige voorbeelden. Het doel van het onderzoek, beschreven in deze tekst, was om het toepassingsgebied van de algoritmes uit te breiden. De belangrijkste bijdragen hiertoe worden hier opgesomd.

- **Langere sequenties**: als we video sequenties willen verwerken die enkele minuten duren, zal het aantal frames snel toenemen. Het speciale karakter

van frames, die dicht bij elkaar liggen, vraagt om een specifieke verwerkingsstrategie.

- **Grotere baselines**: beelden die worden opgenomen door leken kunnen soms ver uit elkaar liggen. Soms is het zo dat moeilijke opname omstandigheden het onmogelijk maken beelden op te nemen die dicht bij elkaar liggen. De resulterende *wide-baseline* beelden vragen andere algoritmen, zoals het gebruik van meer invariante beeldkenmerken.

- **Gedegenereerde gevallen**: klassieke SaM technieken falen voor gedegenereerde camerabewegingen of scènes. Typische voorbeelden zijn camera's die op een rechte lijn of cirkel bewegen of de observatie van een vlak. We hebben strategieën ontwikkeld om deze gedegenereerde gevallen te overwinnen a.h.v. a-priori informatie of uit niet-gedegenereerde stukken uit de opgenomen data.

- **Niet-sequentiële beelden**: SaM wordt nog steeds vooral gebruikt om sequentiële beelden te verwerken. Om niet-sequentiële beelden aan te kunnen, moeten beelden gerelateerd worden met andere beelden dan alleen hun voorganger. Hetzelfde is waar als we de calibratie van beelden, opgenomen in een zigzag beweging, willen verbeteren.

In de laatste hoofdstukken van dit werk tonen we aan dat SaM technieken kunnen ingezet worden in een wijd spectrum van toepassingen: van laboratorium omstandigheden, langs cultureel erfgoed tot ruimtevaart toepassingen. Elke toepassing kent haar eigen problemen en dus ook specifieke algoritmen. We hebben gekozen voor een eclectische aanpak waarbij verschillende recente ontwikkelingen, ook van andere onderzoekers, in de SaM pijplijn opgenomen werden.

De verschillende toepassingsgebieden kwamen aan bod in vele internationale projecten, waaronder enkele Europese IST projecten en verschillende projecten met de Europese ruimtevaartorganisatie ESA. Graag vermelden we ook dat dit werk gebruikt werd door Eyetronics, de spin-off van onze onderzoeksgroep, in twee grote Hollywood filmproducties. Voor *King Arthur* werd een Ierse vallei in 3D gereconstrueerd en voor *The Brothers Grimm* reconstrueerden we een waterput om gebruikt te worden voor speciale effecten.

2 Camera's en Calibratie

De bedoeling van SaM ligt erin om zowel de camera calibratie als de 3D scène te berekenen. Doorheen de tekst maken we gebruik van een lineair cameramodel dat de projectie van een 3D punt $\mathtt{M}$ naar een 2D punt $\mathtt{m}$ in het beeldvlak van de camera berekent a.h.v. formule 1

$$\begin{aligned} \mathtt{m} &\simeq \mathbf{K}\left(\mathbf{R}^T | -\mathbf{R}^T\mathbf{t}\right)\mathtt{M} && (1)\\ &\simeq \mathbf{P}\mathtt{M} && (2) \end{aligned}$$

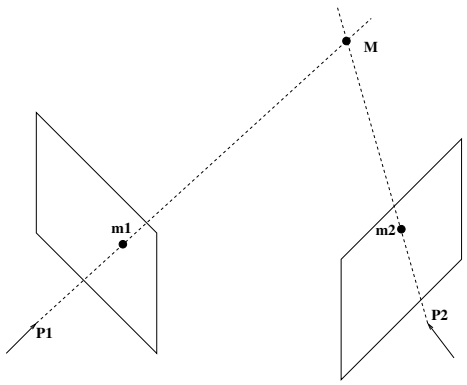


Fig. 1: Het principe achter 3D reconstructie is triangulatie: als het 3D punt `M` projecteert in de twee beelden op $\mathtt{m}_1$ en $\mathtt{m}_2$, dan kunnen we, uitgaande van deze twee projecties, de 3D positie van `M` terugvinden als het snijpunt van de twee teruggeprojecteerde stralen door $\mathtt{m}_1$ en $\mathtt{m}_2$.

waarbij de $\mathbf{R}$ en $\mathbf{t}$ de rotatie en translatie van de camera in de wereld voorstellen (de externe cameraparameters genoemd) en $\mathbf{K}$ de matrix met de interne cameraparameters: de brandpuntsafstand in pixels α_x en α_y, de skew s en het principaal punt u_x, u_y.

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & u_x \\ 0 & \alpha_y & u_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Dit lineaire model wordt uitgebreid met een niet-lineair gedeelte dat de voornaamste niet-lineaire vervorming beschrijft: de radiale distortie. Een gedetailleerde uitleg over het lineaire en niet-lineaire gedeelte van het cameramodel is terug te vinden in appendix A. In de projectieve ruimte blijft formule 2 geldig maar staat $\mathbf{P}$ voor een 3×4 projectiematrix die niet rechtstreeks opgebouwd is uit interne en externe cameraparameters.

Het principe achter 3D reconstructie uit beelden is gebaseerd op dat van stereo reconstructie, geïllustreerd in figuur 1, waarbij een 3D punt `M` kan teruggevonden worden als de snijding van de twee teruggeprojecteerde stralen door zijn projecties $\mathtt{m}_1$ en $\mathtt{m}_2$ in twee beelden. Bemerk dat deze procedure complete kennis van de camera parameters veronderstelt, alsook het vinden van correspondenties tussen de twee beelden. De resolutie van de stereo reconstructie is evenredig met de baseline tussen beide beelden en de brandpuntsafstand van de camera's. Een grotere baseline of brandpuntsafstand verkleint echter het stuk van de scène dat beide camera's zien en dus in 3D kan gereconstrueerd worden. De nauwkeurigheid van de reconstructie is afhankelijk van de verhouding tussen de afstand van het punt tot de camera's en de baseline. Hoe groter die verhouding, hoe groter de onnauwkeurigheid.

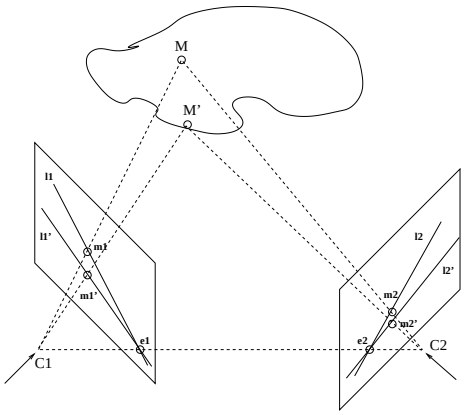


Fig. 2: Alle punten op de 3D lijn $\mathtt{M} - \mathtt{C}_1$ worden geprojecteerd op $\mathtt{m}_1$ in beeld 1. Dit betekent dat het corresponderende punt in beeld twee moet liggen op de projectie van deze lijn ($\mathtt{l}_2$), de epipolaire lijn van $\mathtt{m}_1$.

3 Ongecalibreerde 3D Reconstructie

Bij het begin van een SaM proces bestaat de enige informatie waarover we beschikken uit een aantal opgenomen beelden. Als we het triangulatie proces willen gebruiken om 3D reconstructies te maken moeten we zowel de camera calibratie als correspondenties tussen beelden terugvinden. Eerst zullen we het hebben over hoe we twee beelden met elkaar in correspondentie kunnen brengen. Daarna beschrijven we hoe we beelden toevoegen om een projectieve 3D reconstructie te bekomen. Uiteindelijk bespreken we zelfcalibratie technieken die de projectieve reconstructie opwaarderen tot een metrische reconstructie.

3.1 Relatie Tussen Twee Beelden

Epipolaire Geometrie

Voor een paar van beelden bestaat een onderliggende geometrische structuur die men *epipolaire geometrie* noemt en die verduidelijkt wordt in figuur 2. Alle punten op de 3D lijn $\mathtt{M}-\mathtt{C}_1$ worden geprojecteerd op $\mathtt{m}_1$ in beeld 1. Dit betekent dat het corresponderende punt in beeld twee moet liggen op de projectie van deze lijn ($\mathtt{l}_2$), de epipolaire lijn van $\mathtt{m}_1$. Uit de figuur blijkt verder dat alle correspondenties van punten op de lijn $\mathtt{l}_1$ op de corresponderende epipolaire lijn $\mathtt{l}_2$ moeten liggen. Dit kan uitgedrukt worden a.h.v. de fundamentele matrix $\mathbf{F}$, een 3×3 matrix van rang 2 waarbij voor twee corresponderende punten $\mathtt{m}_1$ en $\mathtt{m}_2$ geldt dat

$$\mathtt{m}_2{}^T \mathbf{F} \mathtt{m}_1 = 0 \tag{4}$$

De fundamentele matrix voor een beeldpaar kan berekend worden uit de calibratie van de camera's, of, als deze niet gekend is, uit voldoende correspondenties

tussen de beelden: 8 voor een lineair, 7 voor een niet-lineair algoritme.

Beeldkenmerken

In de klassieke fotogrammetrie wordt de fundamentele matrix berekend a.h.v. manueel aangeduide correspondenties. Als we dit willen automatiseren, moet een manier gevonden worden om interessante punten automatisch uit de beelden te extraheren en ze in correspondentie te brengen (te *matchen*). Hiertoe worden twee stappen uitgevoerd: kenmerk-extractie en kenmerk-beschrijving waarbij correspondenties gevonden worden waar de beschrijvingen van twee kenmerken goed op elkaar lijken. Verschillende soorten kenmerken en kenmerkbeschrijvingen bestaan.

- **Eenvoudige interesse punten** worden geëxtraheerd als bijvoorbeeld maxima van een hoek-respons functie (de zogenaamde *Harris corners*). Deze kenmerken worden typisch met elkaar vergeleken a.h.v. een beeldvenstertje van enkele pixels groot, gecentreerd rond het geëxtraheerde punt. Als vergelijkingscriterium kan men NCC (genormaliseerde kruiscorrelatie) of SSD (som van gekwadrateerde intensiteitsafstanden) gebruiken.

- **Affien invariante regio's** zijn heel wat stabieler dan eenvoudige punten als de baseline tussen de beelden vergroot. In dat geval vervormt de geprojecteerde scène dramatisch en loopt een vergelijking van beeldvensters rond de punten verkeerd. Affien invariante regio's zijn kenmerken die invariant zijn onder affiene transformaties en dus deze geometrische veranderingen aankunnen.

- **SIFT en SURF kenmerken** werden recent geïntroduceerd als een middenweg tussen de eenvoudige interesse punten en de computationeel intensieve en minder discriminatieve (want meer invariante) affien invariante regio's.

- **Puntconfiguraties** kunnen tenslotte een oplossing bieden voor gevallen waar vele gelijkaardige punten terug te vinden zijn in een beeld. In zo'n geval kan de configuratie van punten een uitweg bieden om toch correspondenties te vinden. Een voorbeeld is de kruisratio van punten die echter alleen projectief invariant is voor vier collineaire of vijf coplanaire punten.

Robuust Algoritme

Op basis van automatische correspondenties tussen beelden kan de epipolaire geometrie berekend worden. Het valt echter niet uit te sluiten dat een substantieel aantal van de gevonden correspondenties niet correct is. Een robuust algoritme om deze zogenaamde *outliers* te detecteren en te verwijderen, moet dus gebruikt worden. De standaard oplossing hiervoor is *RANSAC*, een iteratief algoritme dat op zoek gaat naar de oplossing met het meeste aantal correspondenties dat ermee in overeenstemming is. Een statistische uitbreiding

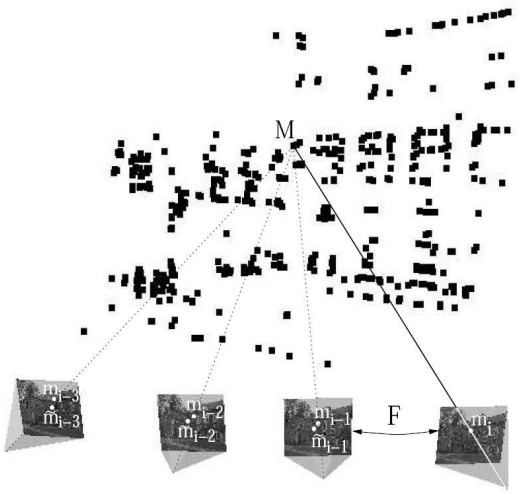


Fig. 3: Het 3D punt $\mathtt{M}$ werd gezien in en gereconstrueerd uit de beelden $i-3$ tot $i-1$. Een F-RANSAC algoritme levert correspondenties op tussen beeld $i-1$ en i. De correspondentie $\langle \mathtt{m_{i-1}}, \mathtt{m_i} \rangle$ geeft ons een mogelijke correspondentie $\langle \mathtt{M}, \mathtt{m_i} \rangle$. Als er voldoende van deze correspondenties voorhanden zijn, kan de projectiematrix $\mathbf{P}_i$ berekend worden.

voor het standaard algoritme maakt de uitvoering veel sneller. Als immers bij het berekenen van de overeenstemmende correspondenties van een oplossing, de eerste -willekeurig gekozen- correspondenties niet overeenstemmen, dan wordt de kans steeds kleiner dat dit de goede oplossing is. Als de kans onder een drempelwaarde zakt, geven we deze oplossing op en gaat het algoritme op zoek naar de volgende.

3.2 Meer Dan Een Beeld

Zoals epipolaire geometrie de relatie tussen corresponderende punten in twee beelden beschrijft, zo bestaan er relaties tussen punten en lijnen in drie of meer beelden. Voor onze SaM pijplijn volgden we echter een meer pragmatische methode die enkel epipolaire geometrie gebruikt en werkt in de projectieve ruimte. Het is immers zo dat, indien de fundamentele matrix tussen twee beelden gekend is, de cameramatrices van beide beelden kunnen berekend worden op vier vrijheidsgraden na. Eens deze matrices gekozen, kunnen alle correspondenties in 3D gereconstrueerd worden a.h.v. triangulatie. Eens deze initialisatie is gebeurd, kunnen nieuwe beelden worden toegevoegd. Dit proces wordt verduidelijkt door figuur 3. Voor een nieuw beeld wordt de fundamentele matrix tussen dit en het vorige beeld berekend. De correspondenties met punten die reeds een corresponderend 3D punt hebben, worden gebruikt om de projectiematrix van het nieuwe

beeld te berekenen. Uiteraard is het nog steeds mogelijk dat deze automatische methode foutieve correspondenties oplevert. Een RANSAC implementatie van het algoritme zal dus alweer noodzakelijk zijn. Aangezien we in de projectieve wereld aan het rekenen zijn, is het verboden om metingen in 3D, zoals de afstand tussen twee punten, uit te voeren omdat deze niet projectief invariant zijn. Daarom is het belangrijk dat alle checks in de beelden zelf gebeuren. Deze zijn immers de enige zekere observatie van de werkelijke wereld waarover we beschikken.

Nadat alle camera's op bovenstaande manier in dezelfde projectieve ruimte zijn berekend, kan een globaal optimisatie proces de oplossing optimaliseren in die zin dat de totale terugprojectiefout van de 3D punten in de camera's t.o.v. de geëxtraheerde punten zo klein mogelijk wordt.

3.3 Zelfcalibratie

De projectieve reconstructie, het resultaat van de vorige stap, is onbruikbaar omdat geen enkele affiene of metrische eigenschap, zoals parallelliteit, hoeken en (relatieve) afstanden, bewaard blijft. Het is daarom noodzakelijk om de projectieve reconstructie naar een metrische op te waarderen. Dit kan op verschillende manieren, afhankelijk van de kennis die men heeft over scène of camera's. In het meest algemene geval echter, zijn zowel scène als camera's volledig onbekend. Dan kunnen enkel beperkingen op de camera parameters aangewend worden om tot een metrische reconstructie te komen. Dit proces noemt men *zelfcalibratie*. Het algoritme dat in dit werk wordt voorgesteld maakt gebruik van de duale absolute kwadriek (DAQ), een entiteit waarvan de plaats in de metrische wereld vastligt en die we in de projectieve ruimte zoeken a.h.v. zijn projectie in de beelden die in de projectieve wereld bewaard blijft.

Over het algemeen worden alle beelden voor SaM opgenomen met dezelfde, digitale camera. Dit betekent dat enkele beperkingen op de cameraparameters kunnen opgelegd worden, zoals het feit dat de skew gelijk moet zijn aan 0, het principaal punt niet verandert en de brandpuntsafstand dezelfde blijft (als tijdens de opname de zoom niet veranderd werd). Deze beperkingen krijgen elk een gewicht naarmate hoe zeker men ervan is en een lineair algoritme berekent de plaats van DAQ in de projectieve reconstructie. Die laat ons toe om de reconstructie naar de metrische wereld te brengen. Een globaal minimisatie algoritme zal alweer de terugprojectiefout minimiseren, waarbij nu nieuwe beperkingen kunnen opgelegd of oude juist opgeheven worden (bv. het toelaten van variërende brandpuntsafstanden).

Een specifiek algoritme werd ontwikkeld om de opwaardering van projectief naar metrisch uit te voeren in het geval men beschikt over verschillende (sub)sequenties die met dezelfde camera zijn opgenomen. In dat geval wordt het gedeelte van de DAQ dat overeenstemt met de interne parameters constant verondersteld voor alle sequenties en kan men die parameters terugvinden door vergelijkingen uit alle sequenties te koppelen. Het resulterende metrische coördinaten stelsel is uiteraard nog steeds anders voor elke sequentie omdat deze

parameters in de DAQ niet aan elkaar gekoppeld kunnen worden.

4 Model Selectie

Op dit punt in de pijplijn zijn we in staat 3D punten en camera's te berekenen vanuit een set van beelden. In de praktijk echter komen zogenoemde kritische camera bewegingen en oppervlakken regelmatig voor. Voorbeelden zijn respectievelijk lineaire bewegingen of vlakke structuren. Deze situaties laten onze algoritmen falen maar omdat we ze regelmatig tegenkomen, moeten we ze het hoofd kunnen bieden. Eerst bespreken we welke problemen we tegenkomen bij vlakke structuren. Daarna tonen we hoe we ze kunnen ontdekken via GRICs, een techniek die we ook gebruiken om lange videosequenties te calibreren. Uiteindelijk introduceren we het concept van de essentiële matrix. Het berekenen van deze matrix met de 5-punts methode lost de problemen met vlakke structuren op.

4.1 Vlakke Structuren

Het meest voorkomende kritische oppervlak voor SaM is een vlakke scène. Dat deze eenvoudige structuur een gedegenereerd geval is, kan vreemd lijken. Nochtans kan een telargument dit vrij eenvoudig uitleggen. In paragraaf A.4.1 kan de lezer een overzicht vinden van de lineaire transformaties die een vlak object kan ondergaan. De meest algemene is de projectieve transformatie of homografie, een 3×3 matrix met 8 vrijheidsgraden. Zoals hierboven vermeld heeft een fundamentele matrix 7 vrijheidsgraden. Zowel een homografie $\mathbf{H}$ als een fundamentele matrix $\mathbf{F}$ kunnen uit puntcorrespondenties worden berekend. Het is echter zo dat een puntpaar voor het eerste geval 2 vergelijkingen levert en voor het tweede geval slechts 1. Dit betekent dat we 4 puntparen nodig hebben voor $\mathbf{H}$ en 7 voor $\mathbf{F}$. Als we met een vlakke scène te doen hebben, legt $\mathbf{H}$ perfect de transformatie van elk punt van het eerste beeld naar het tweede uit. Dat betekent dat we slechts 4 onafhankelijke puntparen kunnen vinden om $\mathbf{F}$ te berekenen. De overige vrijheidsgraden worden dan (verkeerdelijk) ingevuld door de ruis op de gededecteerde punten.

Om exact dezelfde redenen is ook projectieve camera berekening problematisch in het geval van een vlakke scène. Een projectieve projectiematrix $\mathbf{P}$ heeft 11 vrijheidsgraden en elke 3D-2D correpsondentie levert 2 vergelijkingen. We hebben dus 6 punten nodig om $\mathbf{P}$ te berekenen. Als alle punten in een vlak liggen, kunnen alle correspondenties voorspeld worden als de homografie gekend is. Dus opnieuw zijn er $11 - 8 = 3$ vrijheidsgraden over die we niet kunnen invullen.

4.2 GRICs

Als we het probleem met vlakke structuren willen oplossen, moeten we ze eerst kunnen detecteren. Men zou voor een beeldpaar zowel $\mathbf{H}$ als $\mathbf{F}$ kunnen bereke-

nen en selecteren op basis van de overblijvende reprojectiefout. Aangezien **F** echter meer vrijheidsgraden kent, zal zijn reprojectiefout altijd lager liggen dan die van **H**. We moeten dus kiezen tussen beide modellen op een manier die niet alleen in rekening brengt hoe goed het model de data fit maar ook hoe ingewikkeld het model is, een theorie die gebaseerd is op "Occam's razor". Torr introduceerde een mathematische beschrijving van deze theorie voor multiview geometrie die hij GRIC noemde: "Geometric Robust Information Criterion". De formule is

$$GRIC = \sum \rho(e_i^2) + (nd\ln(r) + k\ln(rn)) \tag{5}$$

waarbij $\sum \rho(e_i^2)$ de reprojectiefout voorstelt, met een kleine aanpassing om de invloed van outliers te beperken, en waarbij de term $(nd\ln(r)+k\ln(rn))$ het aantal parameters van het model in rekening brengt. De eerste term zegt m.a.w. hoe goed het model de data beschrijft, terwijl de tweede term zegt hoe ingewikkeld het model is. Als we twee modellen vergelijken, moeten we het model kiezen met de laagste GRIC-kost. Als we dus een ingewikkeld model willen verkiezen boven een eenvoudig model, zal de residuele reprojectiefout veel kleiner moeten zijn om de extra kost van de tweede term te compenseren.

In de meeste gevallen zal de vergelijking van F-GRIC en H-GRIC ons duidelijk maken wanneer vlakke structuren ons in problemen brengen. Men kan echter situaties bedenken waarbij deze beeldpaar-gebaseerde techniek een 3D scène detecteert, terwijl ze ons toch in problemen brengt tijdens de projectieve reconstructie. Figuur 4 geeft een voorbeeld waarbij twee opeenvolgende beeldparen genoeg 3D informatie zien om **F** boven **H** te verkiezen terwijl er toch niet genoeg 3D informatie zichtbaar is in de drie beelden samen om een unieke projectieve reconstructie te bepalen. Om dit geval juist in te schatten moeten we twee andere modellen vergelijken: ofwel een projectieve reconstructie met 3 projectiematrices **P**, ofwel twee opeenvolgende homografieën **H**. In het geval van figuur 4 zal de HH-GRIC een lagere kost hebben dan de PPP-GRIC.

4.3 Lange Video Sequenties

Bovenstaande GRICs kunnen niet alleen gebruikt worden voor detectie van gedegenereerde gevallen. Ze kunnen ook een rol spelen bij de calibratie van lange video sequenties van 1000 beelden of meer.

GRICs

Aangezien videocamera's 25 of 30 beelden per seconde opnemen, liggen opeenvolgende beelden heel dicht bij elkaar. Dit heeft verschillende implicaties. Ten eerste zal het vinden van correspondenties veel eenvoudiger zijn dan voor opnames met een fotocamera. Meestal wordt een tracking algoritme zoals KLT gebruikt in plaats van feature matching. Langs de andere kant heeft het kleine verschil tussen beelden ook een nadeel. Als we figuur 5 bekijken, zien we zowel boven als onder twee opeenvolgende beelden uit een video sequentie van het Antonine Nymphaeum in Sagalassos in Turkije. We berekenden de epipolaire

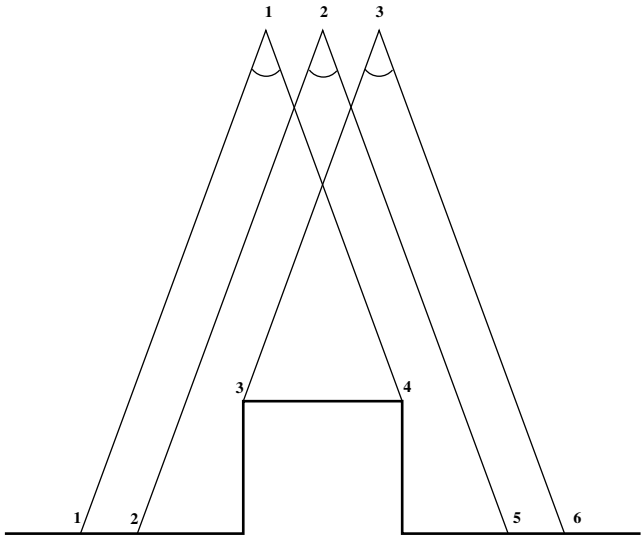


Fig. 4: De F-GRIC van paren 1-2 en 2-3 is lager dan d H-GRIC omdat er genoeg 3D informatie zichtbaar is. Echter, slechts het vlakke stuk (3,4) is gemeenschappelijk in de 3 beelden, wat het onmogelijk maakt om een unieke projectieve reconstructie te berekenen.

geometrie tussen de beelden met hetzelfde algoritme maar met een kleine perturbatie in een parameter. De twee oplossingen voor **F** zijn allebei geldig, wat aantoont dat we **F** niet ondubbelzinnig kunnen berekenen, hoewel er voldoende 3D informatie in beide beelden aanwezig is! De reden hiervoor is dat de translatie van de camera tussen beide beelden zo klein is dat de camerasetup die van een roterende camera benadert, zodat de relatie tussen pixels ook met een homografie kan uitgelegd worden.

Als oplossing voor dit probleem, introduceren we keyframes. We starten met het eerste beeld in de sequentie en tracken features in de volgende beelden. Voor elk beeld berekenen we de F-GRIC en H-GRIC met het eerste beeld. Als de kost voor de F-GRIC lager wordt dan die voor de H-GRIC selecteren we een nieuw keyframe en is de afstand tussen de beelden voldoende voor accurate 3D berekeningen. Om het aantal keyframes te beperken, stellen we het keyframe liever uit zolang we nog 90% van de originele inliers verder tracken. Figuur 6 illustreert het proces.

Subsequenties

Hoewel de keyframe-techniek het aantal beelden voor SaM dramatisch vermindert, blijven er bij lange video sequenties toch nog vele beelden over. In deze gevallen begint het fenomeen van drift te spelen waarbij de accumulatie van meetfouten zo groot kan worden dat de zelfcalibratie-technieken niet meer

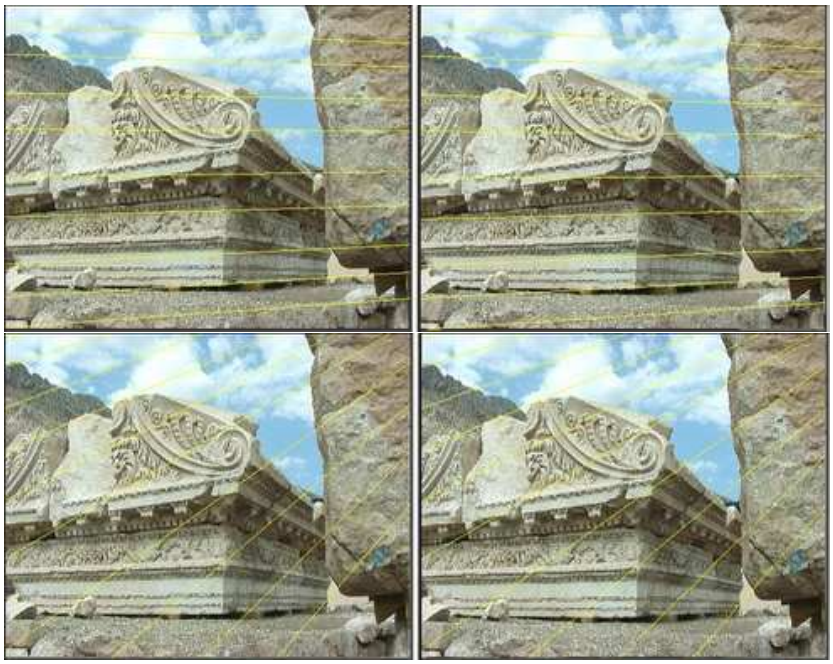


Fig. 5: Twee opeenvolgende beelden uit een video sequentie liggen zo dicht bij elkaar dat de fundamentele matrix niet op een unieke manier kan bepaald worden.

werken. Deze gaan immers uit van een projectief consistente oplossing terwijl de drift het einde van de reconstructie in een substantieel ander projectief frame kan brengen.

Om dit probleem op te lossen, splitsen we de originele sequentie in stukken, zogenaamde subsequenties, waarbij we voldoende overlap voorzien. Elke subsequentie wordt apart in een projectief frame gereconstrueerd. De tracking van de features wordt echter eerst in de volledige sequentie uitgevoerd zodat alle subsequenties in de overlap-gebieden dezelfde features hebben. De resulterende projectieve reconstructies kunnen dan samen gecalibreerd worden a.h.v. het gekoppelde zelfcalibratie algoritme, hierboven beschreven. Dit algoritme berekent de intrinsieke cameraparameters en brengt elke subsequentie in zijn eigen metrisch frame. Om alle subsequenties samen te brengen in één frame, gebruiken we de overlappende beelden. Een groot, globaal, niet-lineair minimisatie proces herleidt daarna de totale reprojectiefout tot een minimum. Op dit moment beschikken we over de calibratie van de keyframes. Als we de calibratie van alle tussenliggende beelden ook wensen te kennen, bv. voor 3D-TV, dan kunnen we die berekenen a.h.v. de gereconstrueerde 3D punten en correspondenties met featuretracks in die beelden. Figuur 7 toont de berekende camera's en 3D punten van de *wereldbol* sequentie.

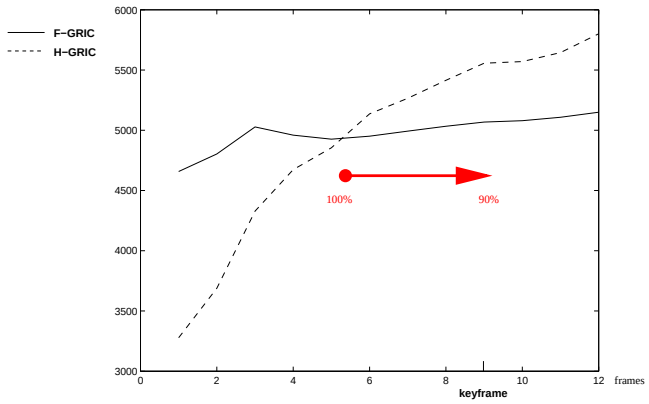


Fig. 6: Zolang de F-GRIC (volle lijn) hoger is dan de H-GRIC (streepjeslijn) gaan we verder door de sequentie. Op het snijpunt bewaren we het aantal inliers voor $\mathbf{F}$ en gaan we verder zolang het aantal inliers hoger is dan 90%. Wanneer we onder de 90% zakken, kiezen we een nieuw keyframe.

4.4 Essentiële Matrices

Totnogtoe werkten we met camera's waarvan de interne parameters niet gekend zijn. Als dit echter wel het geval is, zijn er andere mogelijkheden. Zo kent men bijvoorbeeld het concept van de essentiële matrix $\mathbf{E}$ die men kan gebruiken i.p.v. de fundamentele matrix $\mathbf{F}$ tussen twee beelden als de interne parameters gekend zijn. De essentiële matrix is gedefinieerd als

$$\mathbf{E} \simeq [-\mathbf{R}^T\mathbf{t}]_\times \mathbf{R}^T \tag{6}$$

en kan berekend worden als ofwel de relatieve rotatie en translatie $\mathbf{R}$ en $\mathbf{t}$ tussen de twee camera's gekend is of uit de fundamentele matrix. Er bestaan echter ook algoritmen om $\mathbf{E}$ rechtstreeks uit puntcorrespondenties te berekenen. Philip [68] en Nister [65] beschrijven gelijkaardige niet-lineaire algoritmen om dit uit 5 puntcorrespondenties (het theoretische minimum) te doen. Het voordeel van deze methode is dat ze niet lijdt onder problemen met vlakke structuren. Immers, als beide camera's naar een vlakke scène kijken, dan levert het 5-punt algoritme twee mogelijke oplossingen op. De juiste van de twee kan dan berekend worden via het opleggen van de beperking dat alle gereconstrueerde punten vóór beide camera's moeten liggen.

4.5 Dense Matching

Het resultaat van the SaM procedures bestaat uit de camera calibratie en een set van 3D punten, gereconstrueerd vanuit de geëxtraheerde features. Dit is slechts een deel van het verhaal. Om overtuigende 3D modellen te maken, moeten we veel meer 3D punten reconstrueren om een *dense* reconstructie van de scène



Fig. 7: De gereconstrueerde punten en camera's voor de *wereldbol* sequentie worden gealterneerd getoond in licht grijs en zwart voor gealterneerde subsequenties.

te bekomen. Wij stellen een tweestaps aanpak voor. Eerst berekenen we een dense matching tussen elk opeenvolgend beeldpaar van een sequentie, a.h.v. een hiërarchisch stereo algoritme. Nadien combineren we die stereo resultaten tot dieptekaarten voor elk beeld.

Hiërarchische Stereo

Voor elk opeenvolgend beeldpaar in een gecalibreerde beeldsequentie berekenen we een dense matching a.h.v. een hiërarchisch stereo algoritme. Voor elk punt op een epipolaire lijn in het ene beeld zoeken we een corresponderend punt op de corresponderende epipolaire lijn in het andere beeld. Een dynamisch padzoek algoritme vindt de optimale oplossing die een bepaalde kost minimiseert. Deze kost bestaat uit twee termen. Eerst en vooral moeten matchende pixels goed op elkaar lijken, t.t.z. de genormalizeerde kruiscorrelatie (NCC) moet een zo hoog mogelijke waarde opleveren. De kost voor matching tussen twee pixels x_1 en x_2 die hiermee wordt geassocieerd is $(1 - NCC(x_1, x_2))$. De tweede term is de kost die wordt aangerekend voor het invoeren van een occlusie, t.t.z. een pixel die geen match heeft in het andere beeld. Door de tweede term een hoger gewicht te geven, worden minder occlusies geïntroduceerd en wordt het pad dus vlakker.

Het padzoekalgoritme houdt verder rekening met enkele voorwaarden. Zoals gezegd worden enkel pixels die op elkaars epipolaire lijn liggen gematcht (epipolaire voorwaarde). Verder kan een pixel slechts met één ander pixel overeenkomen (uniciteitsvoorwaarde) en wordt de volgorde van pixels bewaard (volgorde voorwaarde). De gereconstrueerde 3D features leveren een minimum en maximum diepte op waarbinnen de matches zich moeten bevinden (dispariteitsinterval voorwaarde). Figuur 8 toont een match-matrix voor twee epipolaire lijnen uit de

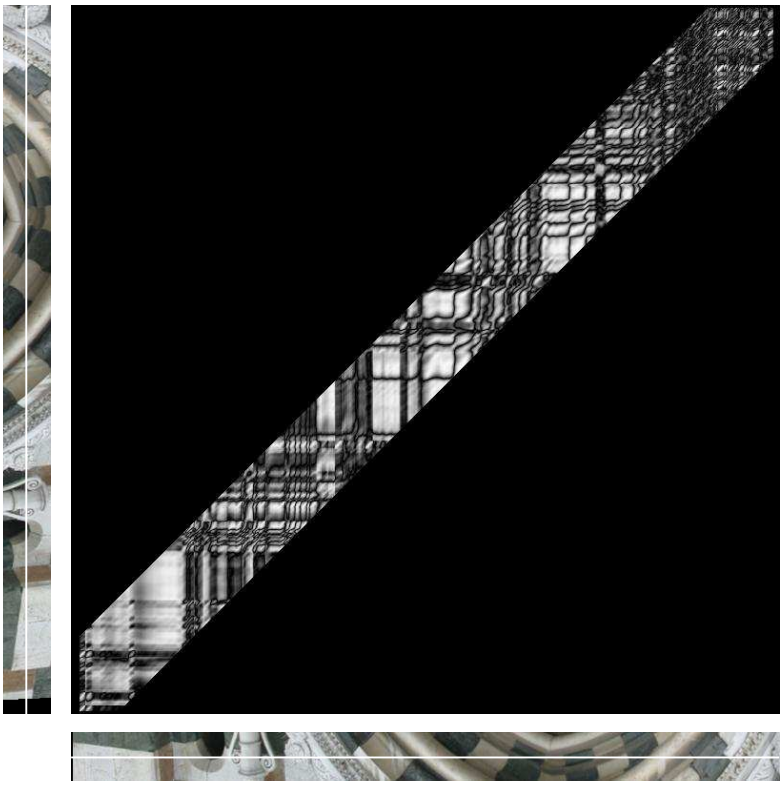


Fig. 8: Match-matrix van twee epipolaire lijnen uit een beeldpaar van de Prato sequentie. Alle pixels op de linkerlijn worden vergeleken met de rechterpixels binnen het dispariteitsinterval.

Prato sequentie. Uit de 3D features halen we dat enkel pixels met een dispariteit tussen -72 en 47 moeten vergeleken worden (dispariteitsinterval). In de match-matrix wordt dan de NCC uitgezet voor elke combinatie en het stereo algoritme zoekt dan het pad met de kleinste kost doorheen de match-matrix, rekening houdend met de andere voorwaarden en de extra kost voor geïntroduceerde occlusies.

Dieptekaarten

Het stereo algoritme berekent dense matches tussen opeenvolgende beeldparen. Deze informatie kan dan samengebracht worden teneinde voor elk beeld een dense dieptekaart te berekenen. Het algoritme daartoe wordt geïllustreerd door figuur 9. Het centrale beeld k is gematcht met beeld $k+1$. Elke match wordt in 3D geplaatst en de onzekerheid e_{k+1} wordt berekend. Beeld $k+1$ is echter ook met beeld $k+2$ gematcht. Alle 3D punten met een match in dit beeldpaar

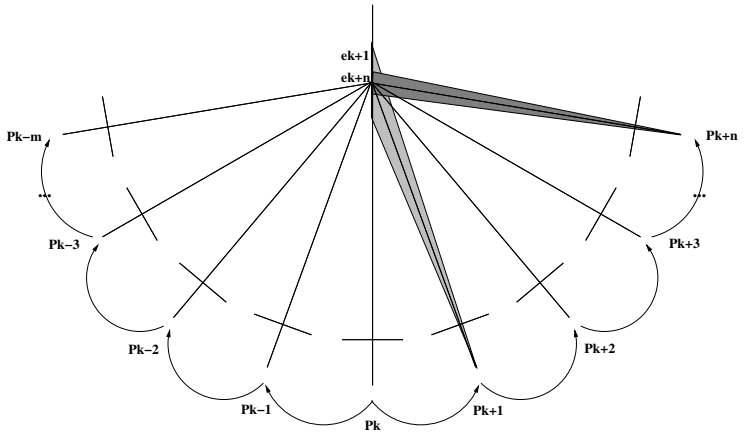


Fig. 9: Dieptekaarten worden berekend door observaties van verschillende stereokaarten samen te brengen om de nauwkeurigheid van de 3D punten te verbeteren.

kunnen worden geüpdated en de onzekerheid kan worden aangepast. Dit proces wordt herhaald tot het punt niet meer gematcht wordt of het gereconstrueerde punt buiten het onzekerheidsinterval komt te liggen. Dit algoritme wordt zowel naar rechts (richting $k+1$) als naar links (richting $k-1$) uitgevoerd om zoveel mogelijk occlusies te vermijden. Deze methode kent enkele voordelen. Ze laat toe om small-baseline stereo algoritmes tussen opeenvolgende beelden uit te voeren en die resultaten te combineren. Gematchte punten uit beelden die verder uit elkaar liggen, geven betere resultaten omdat de fout op de 3D reconstructie kleiner wordt. Nadelig aan deze methode is de noodzaak van een sequentiële input. Als de originele beelden niet in volgorde opgenomen zijn, kent deze methode problemen en moeten we ons heil zoeken in echte multi-view stereo algoritmes.

5 Ruimtevaart Robotica

Het onderzoek dat aan de basis ligt van dit werk werd toegepast in een breed spectrum van toepassingsgebieden. Een eerste hiervan is robotica in de ruimte. Onze onderzoeksgroep heeft enkele projecten met de Europese Ruimtevaartsorganisatie ESA op zijn conto. In het VIABLE project, werden SaM technieken aangewend om een robot op de Japanse satelliet ETS-VII aan te sturen.

5.1 De Satelliet

De ETS-VII satelliet is in feite een testsatelliet om experimenten i.v.m. robotcontrole op uit te voeren. Een robot met 6 vrijheidsgraden kan bestuurd worden

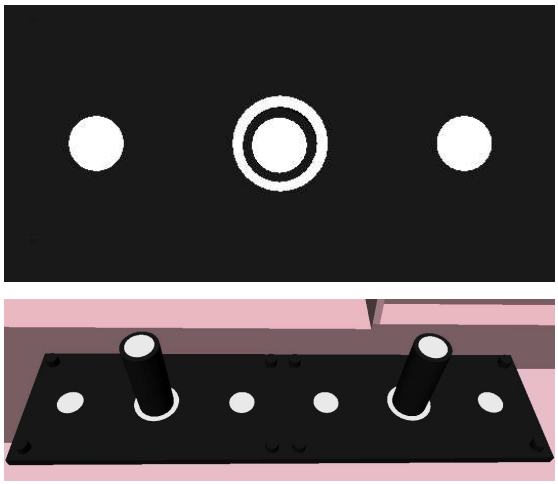


Fig. 10: Bovenaanzicht van een 3-punt marker (boven). Zijaanzicht van het typisch gebruik met twee markers naast elkaar (onderaan).

vanop het basisstation op aarde en heeft toegang tot verschillende satellietonderdelen, waaronder een taskboard dat is uitgerust met verschillende 3-punt calibratie markers. Figuur 10 toont een bovenaanzicht van zo'n 3-punt marker en een zijaanzicht van zijn typisch gebruik, nl. twee markers zij aan zij.

Voor onze experimenten maakten we gebruik van de video camera's die aan de robot waren bevestigd. Communicatie met de satelliet gebeurde tijdens vensters van ongeveer 20 minuten. Alle experimenten moesten op voorhand juist gepland worden om in zo'n tijdslot te kunnen passen.

5.2 Experimenten

Verschillende visie-gebaseerde robot-controle experimenten werden uitgevoerd tijdens ons verblijf in het basisstation in Tsukuba. Het ging om beeldverbeteringsexperimenten die de kwaliteit van de beelden verbeterden zodat we ze beter konden verwerken, calibratieverfijning van intrinsieke parameters en oog-hand calibratie a.h.v. de markers, verschillende on-line robot-controle experimenten a.h.v. markers en andere elementen en tenslotte data-acquisitie experimenten voor latere off-line verwerking.

Beeldverbetering

De kwaliteit van de robotbeelden had zwaar te leiden onder de omstandigheden (bv. direct zonlicht) maar ook onder de gebruikte technologie (compressie, transmissie via een analoge lijn, A/D conversie). Een set van filters werd losgelaten op elk beeld teneinde de beeldkwaliteit te verbeteren. Het ging om achtereenvolgend een mediaan filter, een binomiaal filter, een sharpening filter

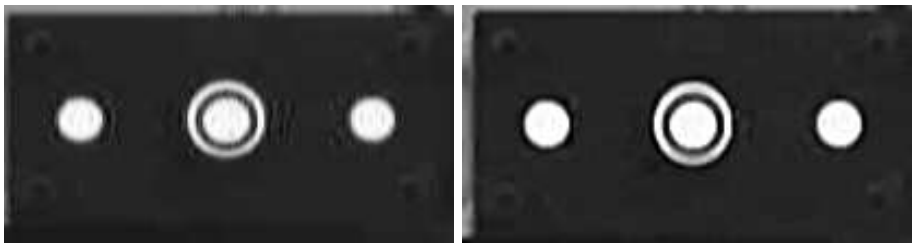


Fig. 11: Een 3-punt marker voor (links) en na (rechts) beeldverbetering.

en radiale distortie en aspect-ratio compensatie. Figuur 11 toont een 3-punt marker voor en na beeldverbetering. De randen zijn veel scherper, het contrast is duidelijker en de ringing rond de randen is verdwenen.

Invariante Features

Verschillende experimenten maakten gebruik van de 3-punt markers. We konden deze automatisch detecteren en identificeren a.h.v. projectieve invarianten. Een eerste projectieve invariant die we gebruikten is de kruisratio van vier collineaire punten. Als er verschillende markers in het beeld aanwezig waren, detecteerden we eerst middelpunten van ellipsen. De kruisratio van alle sets van 4 collineaire punten werd berekend en alleen diegene die voorkwamen in de lijst van op voorhand berekende kruisratio's werden weerhouden. Elke gevonden set van 4 punten bracht een stem uit voor "zijn" lijn. Als voldoende combinaties van punten op een lijn gevonden werden, werd deze lijn weerhouden. De lijnen werden dan geïdentificeerd a.h.v. de kruisratio van hun snijpunten en marker punten. Figuur 12 illustreert het proces.

Een tweede projectieve invariant werd gebruikt als we dichterbij kwamen en slechts weinig markers te zien waren. De invariant bestaat uit de gemeenschappelijke raakpunten van ellipsen. Figuur 13 toont de geëxtraheerde ellipsen en de gemeenschappelijke raaklijnen die ook projectief invariant zijn en dus niet veranderen als de kijkhoek verandert.

Poseschatting

De projectieve invarianten lieten ons toe om markers te identificeren, onder welke hoek we ook naar het taskboard keken. De identificatie leverde steeds genoeg 3D-2D correspondenties op om de pose van de camera te berekenen. Op deze manier kon dan de beweging van de robot gestuurd worden. Figuur 14 toont een typisch voorbeeld. Aan de hand van de markers werd de pose berekend en zo konden verschillende elementen van een CAD model van het taskboard terug in het beeld geprojecteerd worden, inclusief markers die niet gebruikt werden voor de berekeningen. Deze aangevulde realiteitstoepassing gaf de operator een onmiddelijke verificatie van de nauwkeurigheid van de berekende pose.

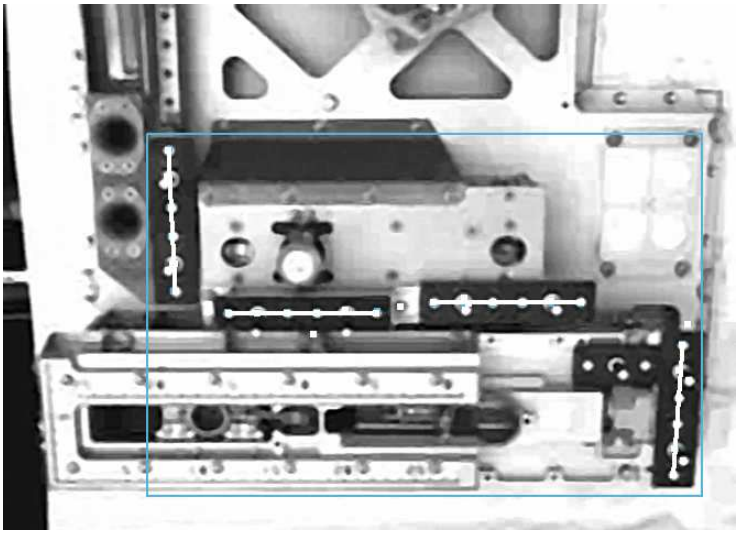


Fig. 12: Marker punten (ellipscentra) worden gegroepeerd a.h.v. kruisratio's. De lijnen die elk een markerblok voorstellen, zijn in wit aangeduid.

Reconstructie

De beelden die werden opgenomen tijdens de communicatievensters werden ook als input gebruikt voor de ongecalibreerde 3D reconstructie algoritmes die hierboven beschreven staan. Het resultaat van de SaM is te vinden in figuur 15. Het verschil tussen de focale lengte van de SaM reconstructie en die van de calibratie voor de vlucht bedraagt ongeveer 5%. Het resultaat van de dense matching is kwalitatief minder goed wegens de grote ongetextureerde stukken en de belichtingsomstandigheden in de ruimte.

6 Planetaire Exploratie

De ESA projecten waarin onze SaM technieken werden toegepast beperkten zich niet tot robotica toepassingen. Geïnspireerd door de Amerikaanse successen, schreef ESA enkele studies uit over planetaire exploratie en het gebruik van visie algoritmen ervoor. Onze onderzoeksgroep was actief in twee zulke studies waarin onze SaM algoritmen gebruikt werden: het PSPE met een planetaire rover en het AEROBOTS project dat luchtschepen bestudeerde.

6.1 Rondrijdende Rovers

Rondrijdende rovers, zoals de Amerikaanse Sojourner, Spirit en Opportunity, kunnen een visie- en navigatiesysteem aan boord hebben. Zolang de energiebronnen het toelaten, kunnen ze dan rondrijden en hun weg bepalen a.h.v. de beelden

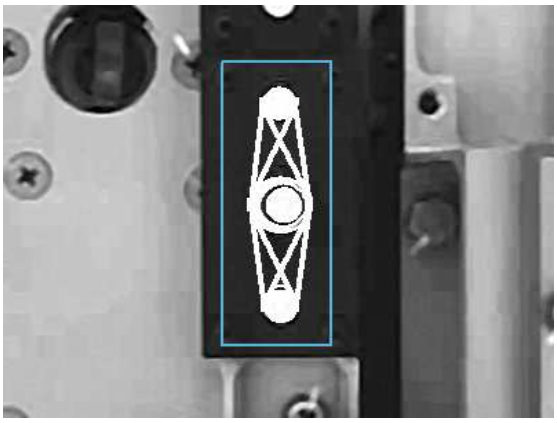


Fig. 13: Gemeenschappelijke raaklijnen van ellipsen leveren raakpunten op die invariant zijn onder projectieve transformaties.

die ze zelf opnemen. Een veel goedkopere oplossing is om een kleine rover steeds in het zicht van de lander te houden die hem met een stereo camera in het oog houdt en stuurt. De rover hoeft dan geen camera's te dragen en krijgt z'n energie vanuit de lander. In het PSPE project bestudeerden we zo'n opstelling. In dit concept maakt de stereo camera eerst beelden van de planetaire omgeving, dewelke naar het grondstation op aarde worden doorgezonden. Daar wordt een 3D reconstructie gemaakt en plannen wetenschappers een traject voor de rover. Dit wordt dan geüpload naar de lander die autonoom de rover dit traject laat volgen.

Calibratie

De stereo camera op de lander was ontworpen met een vrijheidsgraad voor tilt- en pan-bewegingen. Om alle triangulatie metingen in hetzelfde coördinatensysteem uit te voeren, moest de stereo camera eerst gecalibreerd worden. We implementeerden daartoe een set algoritmes die alleen gebruik maakten van de doorgestuurde beelden en die gebaseerd waren op SaM concepten.

Eerst werd de relatieve transformatie tussen beide camera's berekend op basis van feature matches tussen de beelden. De interne parameters van de camera's werden gekend verondersteld, dus kon de essentiële matrix berekend worden waaruit de relatieve transformatie (op de absolute schaal na) volgde. De pan- en tilt-assen werden daarna ook berekend, uitgaande van het doorgezonden beeldmateriaal. Hiervoor moesten correspondenties gezocht worden tussen beeldparen waartussen respectievelijk een pan en een tilt beweging uitgevoerd werd.

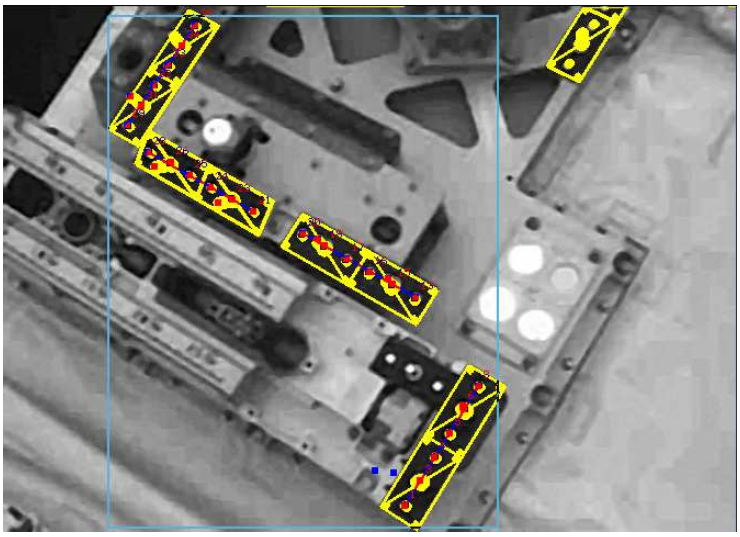


Fig. 14: De camera pose werd berekend uit automatisch gevonden 3D-2D correspondenties. Delen van het 3D CAD model worden over het beeld gelegd en geven een onmiddellijke en intuïtieve verificatie van de nauwkeurigheid.

Reconstructie

Eens de calibratie van de stereo camera gekend, kon het terrein in 3D gereconstrueerd worden. Hiertoe moesten verschillende problemen opgelost worden. Door de vaste relatieve transformatie tussen de camera's bijvoorbeeld, kon het dispariteitsinterval voor heel verre en nabije pixels oplopen tot 600 pixels. Om dit te verhelpen werden de epipolaire lijnen verschoven zodat de dispariteit van het dominant vlak (typisch een horizontaal vlak) op 0 kwam. Dit reduceerde de dispariteit tot aanvaardbare proporties. Ook moest het resultaat van vele stereo recontructies geïntegreerd worden tot één 3D model. Het resultaat van het model, gecreëerd op basis van beelden van het planetaire testbed in ESTEC is te zien in figuur 16. Op dit model werd tijdens de ASTRA 2002 conferentie een traject aangeduid, geüpload naar de lander en uitgevoerd door de rover. De autonome reis van de rover langsheen dit traject duurde ongeveer 5 uur en kon gevolgd worden op een scherm in het conferentie auditorium.

6.2 Luchtschepen en 3D Reconstructie

De Mars Polar Lander en Beagle missies hebben aangetoond dat het landen op een planeet niet altijd van een leien dakje loopt. Een andere aanpak van planetair onderzoek kan erin bestaan om luchtschepen te gebruiken. In het AEROBOTS project van ESA onderzochten we hoe SaM algoritmen een bijdrage kunnen leveren aan zulke systemen. Het concept bestaat erin dat een luchtschip boven een

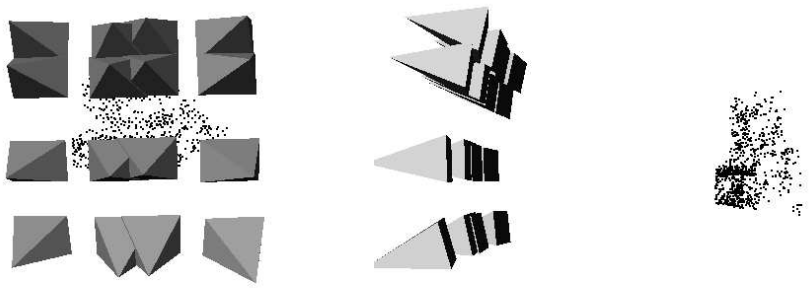


Fig. 15: Gereconstrueerde punten en camera's als output van een SaM proces op de beelden van het ETS-VII taskboard.

planetair oppervlak beelden maakt van dat oppervlak. Deze beelden worden dan lokaal verwerkt om 3D informatie te bekomen in de vorm van dieptekaarten. Deze kunnen dan een goede indicatie geven of er interessante informatie te zien is die doorgestuurd moet worden naar de aarde.

Systeemontwerp

Hoewel het maar een onderdeel is van het totale, ingewikkelde controlesysteem van een luchtschip, lag de nadruk in dit ESA project voornamelijk op de 3D reconstructie. In de ontwerpfase werd beslist om verschillende onafhankelijke componenten te gebruiken die met elkaar in contact stonden via een database. Het voordeel van deze setup was dat de data slechts op één plaats bewaard

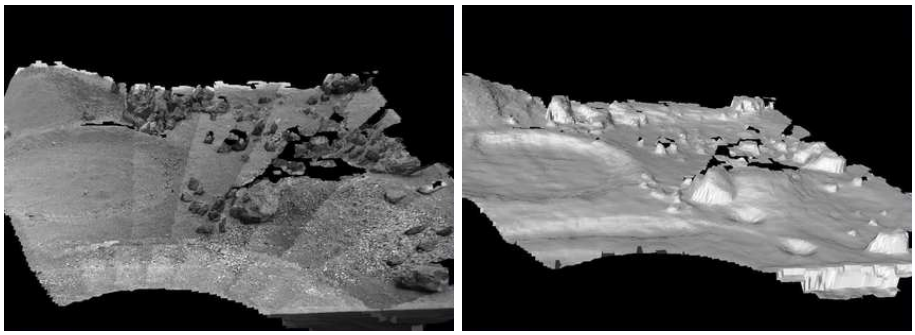


Fig. 16: Een getextureerd en niet-getextureerd aanzicht van het gereconstrueerde model van het planetaire testbed in ESTEC.

moest worden en dat de pijplijn waardoor de data stroomt eenvoudig te implementeren was. Telkens een nieuw beeld beschikbaar kwam, werd het door de database naar de eerste component gestuurd die snel besliste of het beeld bruikbaar was, t.t.z. of er voldoende maar ook niet teveel overlap bestond met het vorige beeld om 3D reconstructie algoritmen op uit te voeren. Als het beeld werd goedgekeurd, werd het naar de SaM component gestuurd die de camera calibratie berekende en in de database bewaarde. Vermits we in een vroege fase van het project over beeldmateriaal beschikten, realiseerden we ons snel dat klassieke SaM technieken niet het verhoopte resultaat konden bereiken wegens problemen met gedegenereerde gevallen, zoals een vlakke scène. Daarom werden algoritmen geïmplementeerd die o.a. het 5-punt algoritme voor de essentiële matrix gebruikten. Als de berekende camera calibratie beschikbaar kwam, werd ze doorgestuurd naar de dense matching die dieptekaarten genereerde, evenals maskers die voor elk beeld aangaven welke delen nieuw waren en welke delen reeds berekend waren in vorige beelden. Deze maskers maakten het mogelijk de informatie op een efficiënte manier te comprimeren a.h.v. het JPEG2000 algoritme.

Resultaten

Het Aerobots systeem werd getest op verschillende soorten input data. Om de nauwkeurigheid van het resultaat te beoordelen, werd een simulator gebouwd die een virtueel luchtschip over een virtueel terrein liet vliegen. Op basis van de virtuele beelden werd dan een reconstructie gemaakt die kon vergeleken worden met het origineel. De resulterende camera posities tonen een afwijking van ongeveer 0.1%. Het rotatie verschil van de optische as tussen het origineel en het resultaat was nooit meer dan 0.25 graden. Een tweede test werd vroeg in het project uitgevoerd op video beelden die opgenomen werden boven het planetaire testbed in ESTEC. Enkele resultaten van de bekomen 3D reconstructie zijn te zien in figuur 17. Tenslotte werd een luchtschip gebouwd, bestaande uit een grote, met Helium gevulde, ballon met een gondel waarin een digitale bluetooth-camera beelden opnam van het terrein. Enkele resultaten van de reconstructie zijn te zien in figuur 18.

7 Cultureel Erfgoed

Ons tastbaar cultureel erfgoed heeft vele vijanden, zowel menselijke als natuurlijke. In dit werk tonen we twee toepassingen die SaM technieken gebruiken om ons cultureel erfgoed beter te bewaren voor het nageslacht.

7.1 Tijdslijn tot Vandaag

Een eerste toepassing had tot doel om de link van het heden met het verleden duidelijk in de verf te zetten. Het provinciaal archeologisch museum te Ename gebruikte al langer een 4D QTVR object om de evolutie van de abdij en het dorp

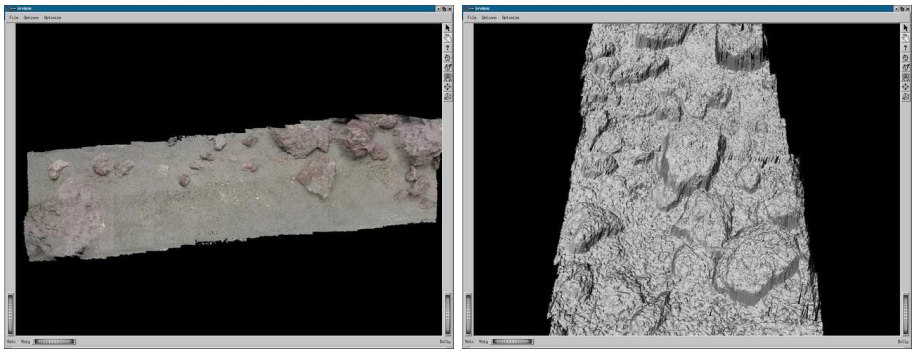


Fig. 17: Enkele resultaten van de dense 3D reconstructie van het planetaire testbed in ESTEC op basis van de video data. Zowel een getextureerd als een ongetextureerd aanzicht worden getoond.

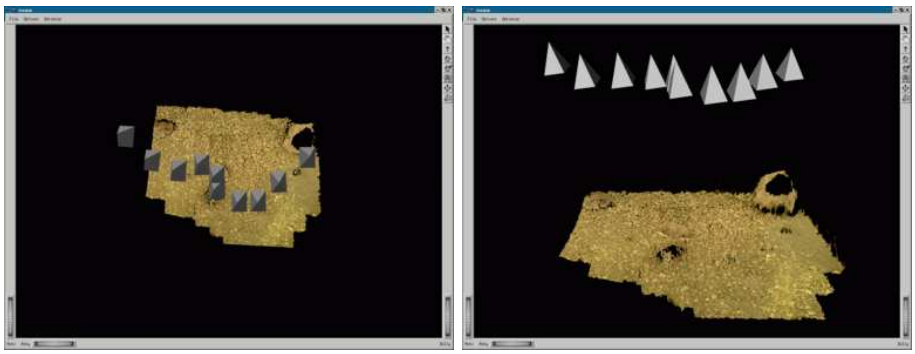


Fig. 18: Resulterende 3D reconstructie en camera calibratie voor een set van beelden, opgenomen met een echt luchtschip.

over de laatste 10 eeuwen te tonen vanop 36 verschillende gezichtspunten. De bezoeker kan met een simpele toetsdruk doorheen de tijd bewegen of een ander gezichtspunt kiezen. Om de link met het heden duidelijk te maken, wilde men een extra laag toevoegen met de huidige situatie. Om dit mogelijk te maken moesten 36 beelden worden aangemaakt, zodanig dat ze overeenkwamen met de standpunten van de camera in de andere tijdvakken. Figuur 19 toont de setup van de virtuele camera's die zich op een hoogte van 150 meter boven en een horizontale afstand van 260 meter van het centrum van de abdij bevinden.

Omdat het onmogelijk is om 36 beelden op perfect de juiste plaats op te nemen met een camera die dezelfde parameters heeft als de virtuele, moesten we een andere strategie bedenken. Er werd besloten om gebruik te maken van Image Based Rendering (IBR) technieken die in staat zijn om nieuwe, virtuele beelden te genereren op basis van andere beelden die in de buurt zijn opgenomen. Daarom werd een helicopter gehuurd van waaruit ongeveer 130 foto's werden

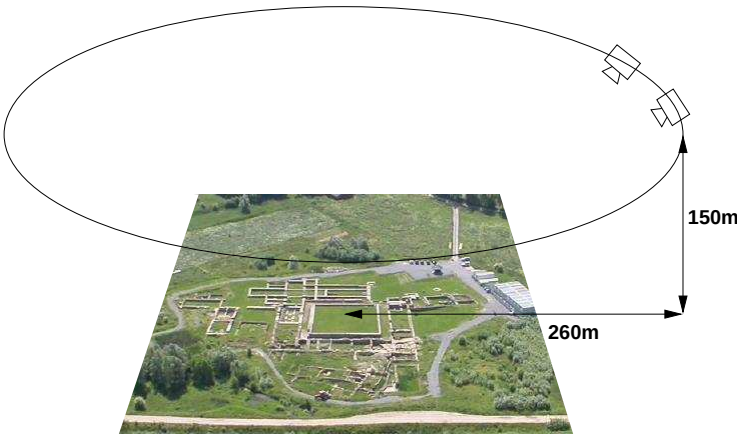


Fig. 19: De setup van de virtuele camera's in de Tijdslijn van het archeologisch museum te Ename. Ze zijn verspreid over een cirkel met een straal van 260 meter op een hoogte van 150 meter.

genomen van de site. De foto's werden automatisch gegroepeerd in sets die geschikt waren om er SaM op toe te passen. Alweer dwong de vrij vlakke scène ons om gebruik te maken van de stabiele methodes om gedegenereerde gevallen te omzeilen. De verschillende reconstructies werden manueel geregistreerd met het coördinatensysteem van de site en een dense matching leverde dieptekaarten op. Samen met de camera calibratie dienden deze als input voor de IBR algoritmen die de 36 nieuwe aanzichten berekenden. Figuur 20 toont een nieuw aanzicht met zijn virtuele tegenhanger. Beide beelden komen heel goed overeen in het gedeelte van de archeologische site. Het gebied aan de rand van het beeld, bijvoorbeeld aan de kerk, heeft een iets grotere fout die te wijten is aan de registratie stap die geen informatie in dit gebied gebruikt.

7.2 3D Webservice

Een laatste toepassing is culminatie van alle 3D SaM technieken en algoritmen in dit werk. In het kader van het Europese project Epoch bouwden we een 3D webservice. Een schematisch overzicht is terug te vinden in figuur 21. Het idee achter de webservice is dat het rekenintensief gedeelte van SaM door een computercluster op ESAT wordt uitgevoerd. De gebruiker neemt een set van beelden op en stuurt deze, nadat hij zich geauthenticeerd heeft, door naar de server op ESAT. Daar worden ze automatisch verwerkt tot een 3D reconstructie, bestaande uit camera's en dieptekaarten. Dit resultaat wordt beschikbaar gesteld via een ftp server en kan gedownload en bekeken worden op de PC van de gebruiker.

De webservice is ontworpen om 3D reconstructies te maken van beelden uit



Fig. 20: Het nieuwe beeld voor positie 10 en zijn virtuele tegenhanger uit 1780.

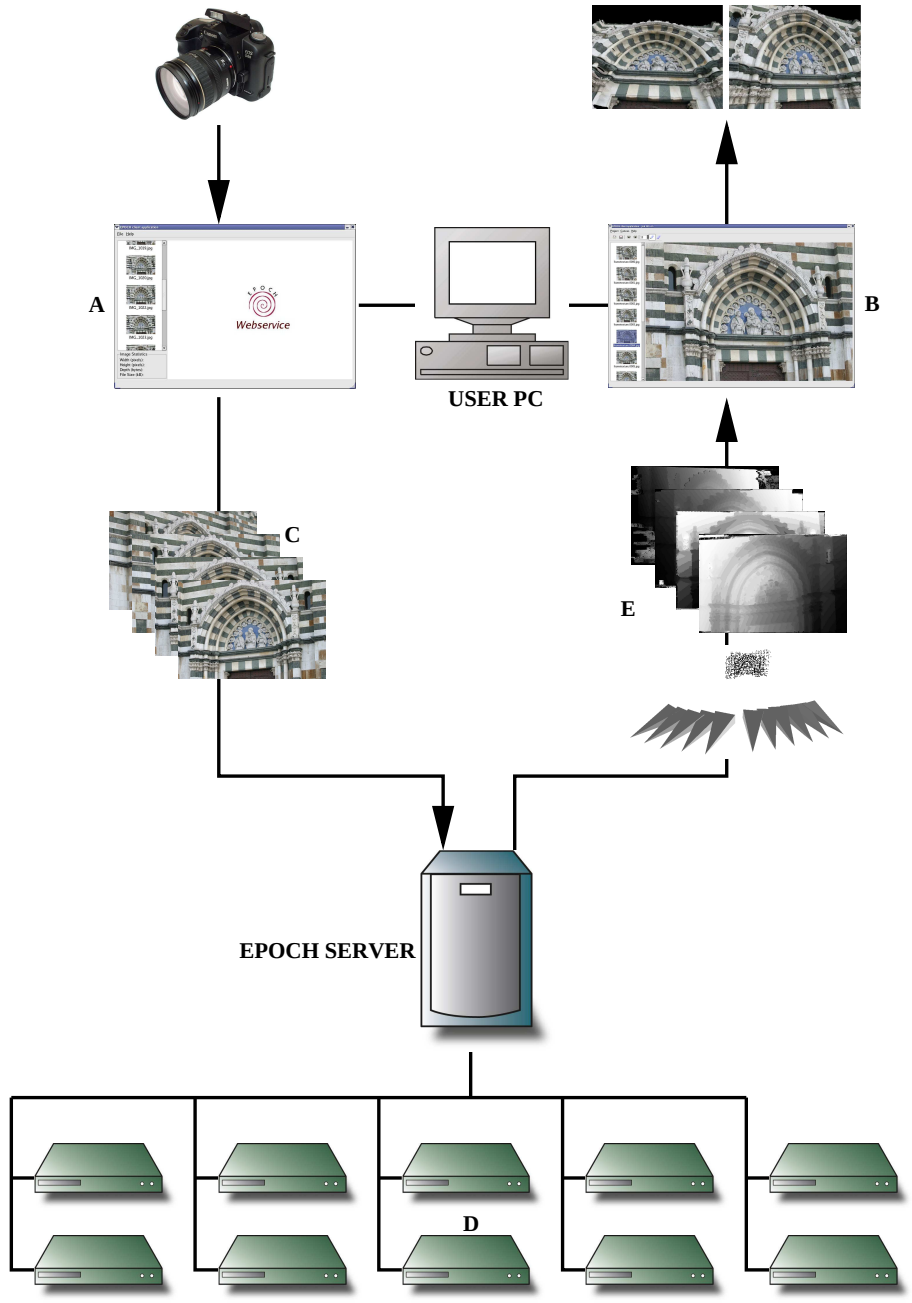


Fig. 21: Schematisch overzicht van de webservice. Beelden (C) worden vai de upload tool (A) geüpload naar de server. Daar worden ze op met behulp van een computercluster (D) geprocessed. Het resultaat (E) can via ftp worden gedownload en gevisualiseerd op de PC van de gebruiker (B).

een brede waaier van toepassingen. Omdat er geen interactie meer mogelijk is eens de beelden geüpload zijn, is robuustheid en autonomie van het systeem een belangrijke voorwaarde. Om frustratie bij de gebruiker en herhaalde uploads te vermijden, moet de server de kans op een goed 3D resultaat maximaliseren. Dit heeft geleid tot een hiërarchisch, geparallelliseerd en opportunistisch 3D reconstructie systeem. De beelden worden eerst op een kleinere resolutie verwerkt omdat dit zowel de rekentijd vermindert als de robuustheid verhoogt omdat de kans op foute matches kleiner wordt. Het systeem houdt er rekening mee dat beelden niet noodzakelijk sequentieel zijn opgenomen of doorgestuurd, wat een impact heeft op het matchen, het reconstrueren van de camera's en de dense reconstructie. Door het gebruik van een cluster van computers die in verbinding staan met een centrale database, kan het systeem verschillende taken in parallel uitvoeren. Dit versnelt de verwerking enorm.

Om de calibratie van niet-sequentiële beelden te berekenen, moeten we uitzoeken welke beelden we met elkaar in correspondentie gaan brengen. Om combinatorische redenen kunnen we niet zomaar elk beeld met elk ander beeld matchen en dus wordt een snel algoritme uitgevoerd waarbij een sterk verkleinde versie van de beelden vergeleken wordt. Dit levert een lijst op van beeldparen en beeldtriplets waarvoor een projectieve reconstructie berekend wordt a.h.v. Surf features. Omdat we met planaire scènes te maken kunnen hebben, berekenen we voor elk triplet ook de PPP-GRIC en HH-GRIC. Alle triplets met voldoende 3D informatie (PPP-GRIC $<$ HH-GRIC) worden samengebracht in een algoritme dat de zelfcalibratie berekend op een gekoppelde en statistische wijze. Daarna wordt de calibratie van alle beelden in hetzelfde metrische frame berekend en wordt de calibratie geüpscaled naar de hoogst mogelijke resolutie. Een dense matching levert dan een dieptekaart voor elk beeld op. Figuur 22 toont het resultaat, bekomen uit een beeldsequentie van de Tempel van de Maskers in Mexico. Het niet getextureerde aanzicht in de figuur geeft een mooi beeld van de ingewikkelde geometrie van de scène. In figuur 23 worden de 3 input beelden getoond van een zeer korte sequentie van de San Marco façade in Venetië. De afstand tussen de beelden is zo groot dat gewone features niet volstonden om de calibratie te berekenen. Het gebruik van de meer stabiele Surf features leverde een mooi resultaat op.

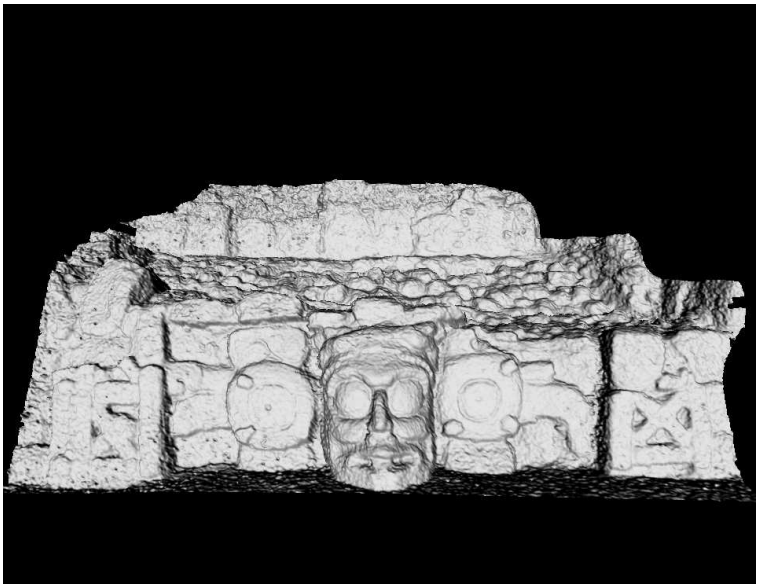


Fig. 22: Niet getextureerd aanzicht van de 3D reconstructie van de *Tempel van de Maskers*. Dit aanzicht geeft een mooi beeld van de ingewikkelde geometrie van de scène..

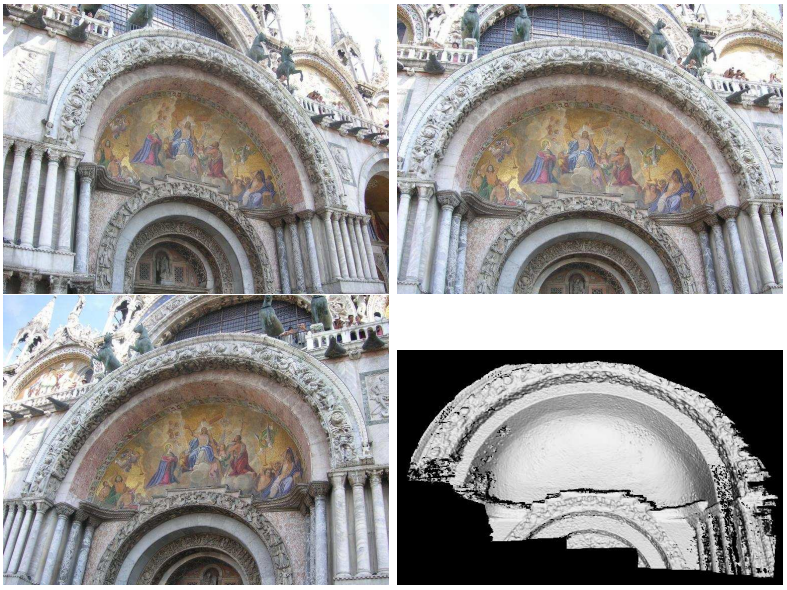


Fig. 23: Boven en links onder: de drie originele beelden van de San Marco façade in Venetië. Dit is duidelijk een moeilijk wide-baseline probleem. Onder rechts: de resulterende 3D reconstructie, getoond zonder textuur.

Curriculum Vitae



Maarten Vergauwen was born on November 7, 1974 in Dendermonde, Belgium. In 1997 he received a Masters degree in Electrical Engineering from the K.U.Leuven.

From 1997 to 2006, he worked towards a Ph.D. under the supervision of prof. Luc Van Gool, in the VISICS lab (part of ESAT-PSI) at the K.U.Leuven. In this period he did research in the field of computer vision in general and Structure and Motion algorithms in particular. He investigated its applicability in many application domains.

He was active in several European projects with academic, space related and industrial partners, among which VIABLE, PSPE and AEROBOTS with the European Space Agency (ESA), the IST projects ATTEST and INVIEW and the European Network of Excellence EPOCH.

He is a guest professor at the Groep-T Engineering School in Leuven where he teaches "3D acquisition" and taught computer vision courses as a guest lecturer at the university of Verona (Italy) and the space and technology center ESTEC of ESA in Noordwijk (The Netherlands).

List of Publications

Articles in International Journals

[1] M. Vergauwen and L. Van Gool, *Web-based 3D Reconstruction Service*, in Machine Vision Applications (MVA), 2006.

[2] M. Vergauwen, G. Willems, F. Verbiest, L. Van Gool and D. Pletinckx, *Combining Image and Model Based Rendering of an Archaeological site*, in Signal Processing: Image Communication, 20(9-10), pp. 891-906, 2005.

[3] K. Landzettel, K. Arbter, B. Brunner, B.-M. Steinmetz, M. Pollefeys, M. Vergauwen, R. Moreas, F. Xu, L. Steinicke and B. Fontaine, *A Micro-Rover Navigation and Control System for Autonomous Planetary Exploration*, in Advanced Robotics, Special Issue on "Service Robots in Space"18(3), pp.285-314, 2004.

[4] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops and R. Koch, *Visual Modeling with a hand-held camera*, International Journal of Computer Vision 59(3), pp. 207-232, 2004.

[5] M. Vergauwen, M. Pollefeys and L. Van Gool: *A stereo-vision system for support of planetary surface exploration*, Machine Vision and Applications 14(1), pp.5-14, 2003.

[6] G. Van Meerbergen, M. Vergauwen, M. Pollefeys and L. Van Gool,*A Hierarchical Symmetric Stereo Algorithm Using Dynamic Programming*, International Journal on Computer Vision 47(1), pp. 275-285, 2002.

[7] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *Automated reconstruction of 3D scenes from sequences of images*, ISPRS Journal Of Photogrammetry and Remote Sensing (55)4, pp. 251-267, 2000.

Articles in Proceedings of International Conferences

[8] G. Willems, F. Verbiest, M. Vergauwen and L. Van Gool, *Real-time Image Based Rendering From Uncalibrated Images*, In Proc.

Fifth International Conference on 3D Digital Imaging and Modeling (3DIM), pp. 221-228, June 2005.

[9] M. Vergauwen, D. Pletinckx, G. Willems and L. Van Gool, *Landscape reconstruction through image-based rendering*, Proc. Int. Workshop on Recording, Modeling and Visualization of Cultural Heritage, 2005.

[10] M. Vergauwen, G. Matthews, L. Van Gool and B. Fontaine, *Advanced Vision Systems for Exploration by a Planetary Aerobot*, Proc. of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS 2005), 2005.

[11] M. Vergauwen, D. Pletinckx, G. Willems, F. Verbiest, L. Van Gool and Truus Helsen, *As Time Flies By: Mixed Image and Model-Based Rendering of an Historical Landscape from Helicopter Images*, in Proc. of 5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST 2004), Brussels, Belgium, December 2004.

[12] L. Van Gool, M. Waelkens, P. Müller, T. Vereenooghe and M. Vergauwen, *Total Recall: A Plea for Realism in Models of the Past, In International Society for Photogrammetry and Remote Sensing (ISPRS)*, XXth ISPRS Congress, Commission V, 2004.

[13] P. Müller, T. Vereenooghe, M. Vergauwen, L. Van Gool and M. Waelkens, *Photo-Realistic and Detailed 3D Modeling: The Antonine Nymphaeum at Sagalassos (Turkey)*, In Computer Applications and Quantitative Methods in Archaeology (CAA): Beyond the artifact - Digital interpretation of the past, 2004.

[14] M. Vergauwen, G. Matthews, L. Van Gool and B. Fontaine, *Imaging and Localization for a Planetary Aerobot*, In Proc. 8th ESA Workshop on Advanced Space Technologies for Robotics and Automation, ASTRA 2004.

[15] M. Pollefeys, L. Van Gool, M. Vergauwen, K. Cornelis, F. Verbiest and J. Tops, *3D Capture of Archaeology and Architecture with a Hand-Held Camera*, The International Archive of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXIV, Part 5/W12, pp. 262-267, 2003.

[16] J. Cosmas, T. Itegaki, D. Green, N. Joseph, L. Van Gool, A. Zalesny, D. Vanrintel, F. Leberl, M. Grabner, K. Schindler, K. Karner, M. Gervautz, S. Hynst, M.Waelkens, M.Vergauwen, M. Pollefeys, K.Cornelis, T. Vereenooghe, R. Sablatnig, M. Kampel, P. Axell and E. Meyns,*Providing Multimedia Tools for Recording, Reconstruction,*

Visualisation and Database Storage/Access of Archaeological Excavations, Proceedings of VAST 2003, Brighton (UK), 2003.

[17] B. Fontaine, L. Steinicke and M. Vergauwen, *Payload support for planetary exploration - monitoring and control of a tethered micro-rover for planetary exploration*, Proc. of 6th International Conference on Climbing and Walking Robots (Clawar), Catania, Italy, 2003.

[18] M. Pollefeys, L. Van Gool, M. Vergauwen, K. Cornelis, F. Verbiest and J. Tops, *Video-to-3D*, Proc. Photogrammetric Computer Vision 2002 (ISPRS Commission III Symposium), International Archive of Photogrammetry and Remote Sensing, Part A, pp. 247-252, 2002.

[19] G. Van Meerbergen, M. Vergauwen, M. Pollefeys and L. Van Gool, *A Hierarchical Stereo Algorithm Using Dynamic Programming*, Proc. IEEE Workshop on Stereo and Multi-Baseline Vision, pp. 166-174, 2001.

[20] M. Vergauwen, M. Pollefeys and L. Van Gool, *A stereo vision system for planetary surface exploration*, Proc. Conference on Optical 3-D Measurement Techniques V, Vienna, pp. 192-199, 2001.

[21] M. Pollefeys, M. Vergauwen, K. Cornelis, J. Tops, F. Verbiest and L. Van Gool, *Structure and motion from image sequences*, Proc. Conference on Optical 3-D Measurement Techniques V, pp. 251-258, Vienna, 2001.

[22] M. Pollefeys, M. Vergauwen, K. Cornelis and L. Van Gool, *3D Acquisition of Archaeological Heritage from Images*, Proc. XVIII International Symposium of CIPA, ISPRS International Archive of Photogrammetry, Remote Sensing and Spatial Information Sciences Vol. XXXIV-5/C7 / CIPA International Archive for Documentation of Cultural Heritage Vol. XVIII-2001, pp. 381-388, 2001

[23] J. Schouteden, M. Pollefeys, M. Vergauwen and L. Van Gool, *Image-based 3D acquisition tool for architectural conservation*, Proc. XVIII International Symposium of CIPA, ISPRS International Archive of Photogrammetry, Remote Sensing and Spatial Information Sciences Vol. XXXIV-5/C7 / CIPA International Archive for Documentation of Cultural Heritage Vol. XVIII-2001, pp. 301-305, 2001

[24] M. Pollefeys, L. Van Gool, M. Vergauwen, K. Cornelis, F. Verbiest and J. Tops, *Image-based 3D Acquisition of Archaeological Heritage and Applications*, Proc. VAST 2001 (Virtual Reality, Archaeology and Cultural Heritage), 2001.

[25] K. Cornelis, M. Pollefeys, M. Vergauwen, F. Verbiest and L. Van Gool, *Tracking based structure and motion recovery for augmented video productions*, Proceedings ACM symposium on virtual reality software and technology - VRST 2001, pp. 17-24, Canada, 2001.

[26] K. Nuyts, J.-P. Kruth, B. Lauwers, H. Neuckermans, M. Pollefeys, L. Qiongyan, J. Schouteden, P. Smars, K. Van Balen, L. Van Gool and M. Vergauwen, *From a conservationist's point of view*, Proc. Conference on Optical 3-D Measurement Techniques V, Vienna, pp. 179-186, 2001.

[27] M. Pollefeys, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops and L. Van Gool, *Virtual Models from Video and Vice-Versa*, keynote presentation, Proc. International Symposium on Virtual and Augmented Architecture (VAA01), B. Fisher, K.Dawson-Howe, C. O'Sullivan (Eds.), Springer-Verlag, pp. 11-22, 2001.

[28] K. Nuyts, J.P.Kruth, B. Lauwers, M. Pollefeys, L. Qiongyan, J. Schouteden, P. Smars, K. Van Balen, L. Van Gool and M. Vergauwen, *Vision on Conservation: VIRTERF* ,Proc. International Symposium on Virtual and Augmented Architecture (VAA01), Springer-Verlag, pp. 125-132, 2001.

[29] M. Vergauwen, M. Pollefeys and L. Van Gool, *A stereo vision system for support of planetary surface exploration*, In B. Schiele, G. Sagerer (Eds.), Computer Vision Systems (Proc. Second International Workshop), LNCS 2095, pp. 298-312, 2001.

[30] M. Vergauwen, M. Pollefeys, R. Moreas, F. Xu, G. Visentin, L. Van Gool and H. Van Brussel, *Calibration, Terrain Reconstruction and Path Planning for a Planetary Exploration System*, Proc. iSAIRAS, 2001.

[31] M. Vergauwen, M. Pollefeys and L. Van Gool, *Calibration and 3D Measurement from Martian Terrain Images*, Proceedings of the 2001 IEEE International Conference on Robotics and Automation, pp 2153-2158, Seoul, Korea, 2001.

[32] M. Vergauwen, M. Pollefeys and L. Van Gool, *Autonomous Operations of a Micro-Rover for Geo-Science on Mars*, Proc. 6th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA 2000), pp. 3.7.1, 2000.

[33] K. Cornelis, M. Pollefeys, M. Vergauwen and L. Van Gool, *Augmented Reality using Uncalibrated Video Sequences*, Proc. Second Workshop on Structure from Multiple Images of Large Scale Environments, pp. 144-160, 2000.

[34] M. Pollefeys, M. Vergauwen and L. Van Gool, *Automatic 3D modeling from image sequences*, invited presentation, International Archive of Photogrammetry and Remote Sensing, Vol. XXXIII, Part B5, pp. 619-626, 2000.

[35] M. Vergauwen, M. Pollefeys, T. Tuytelaars and L. Van Gool, *On Satellite Vision-aided Robotics Experiment*, Proc. International Conference on Robotics and Automation, pp. 4039-4044, IEEE Computer Society Press, 2000.

[36] M. Pollefeys, R. Koch, M. Vergauwen and B. Deknuydt, *three-dimensional scene reconstruction from images*, Proc. SPIE Electronic Imaging, Three-Dimensional Image Capture and Applications III, SPIE Proceedings series, Vol. 3958, pp. 215-226, 2000.

[37] M. Pollefeys, M. Proesmans, R. Koch, M. Vergauwen and L. Van Gool, *Acquisition of detailed models for virtual reality*, Virtual reality in archaeology, BAR international series 843, pp. 71-77, Barcelo J.A., Forte M. and Sanders D.H., eds., 2000.

[38] T. Tuytelaars, M. Vergauwen, M. Pollefeys and L. Van Gool, *Image matching for Wide baseline Stereo*, Proc. International Conference on Forensic Human Identification, October 1999.

[39] L. Van Gool, F. Defoort, J. Hug, G. Kalberer, R. Koch, D. Martens, M. Pollefeys, M. Proesmans, M. Vergauwen and A. Zalesny, *Image-based 3D modeling: modeling from reality*, Proceedings NATO advanced research workshop on confluence of computer vision and computer graphics, NATO science series - 3. high technology, vol. 84, pp. 161-178, August 29-31, 1999.

[40] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *Hand-held acquisition of 3D models with a video camera*, Proc. 3DIM'99 (Second International Conference on 3-D Digital Imaging and Modeling), IEEE Computer Society Press, pp. 14-23, 1999.

[41] M. Vergauwen, R. Koch, T. Tuytelaars and L. Van Gool, *Vision-based Robotics Experiment on ETS VII*, Proc. 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS'99), pp 409-414, 1999.

[42] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *An Automatic Method for Acquiring 3D Models from Photographs: applications to an Archaeological Site*, ISPRS International Workshop on Photogrammetric Measurement, Object Modeling and Documentation in Architecture and Industry, Thessaloniki, International Archive of Photogrammetry and Remote Sensing, Vol. XXXII, Part 5W11, pp.76-80, 1999.

[43] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *Automatic Generation of 3D Models from Photographs,* Proceedings Virtual Systems and MultiMedia (VSMM'98), 1998.

[44] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *Virtualizing Archaeological Sites*, Proceedings Virtual Systems and MultiMedia (VSMM'98), 1998.

[45] L. Van Gool, F. Defoort, R. Koch, M. Pollefeys, M. Proesmans and M. Vergauwen, *3D modeling for communications*, Proceedings Computer Graphics International, pp.482-487, Hannover, 1998.

[46] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *Metric 3D Surface Reconstruction from Uncalibrated Image Sequences*, Proc. SMILE Workshop (post-ECCV'98), LNCS 1506, pp.138-153, Springer-Verlag, 1998.

[47] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *Flexible acquisition of 3D structure from motion*, Proceedings IEEE workshop on Image and Multidimensional Digital Signal Processing'98, pp.195-198, Alpbach, 1998.

[48] M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, *Flexible 3D Acquisition with a Monocular Camera*, Proceedings IEEE Int'l Conf. on Robotics and Automation'98, Vol.4, pp.2771-2776, Leuven, 1998.

[49] M. Pollefeys, M. Proesmans, R. Koch, M. Vergauwen and L. Van Gool, *Flexible 3D reconstruction techniques with applications in archeology*, 26th. Int'l Conf. on Computer Applications in Archeology, Barcelona, 1998.

Master thesis

[50] F. Verbiest and M. Vergauwen, *Robust three-dimensional reconstruction from multiple images*, Master Thesis Electrical Engineering, K.U.Leuven, May 1997.