Plane-Based Close Range Photogrammetric Reconstruction of Buildings

Håkan Fors Nilsson
David Grundberg

March 9, 2009
Master's Thesis in Computing Science, 30 ECTS-credits
Supervisor at CS-UmU: Niclas Börlin
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

This thesis presents theory and a prototype computer application for photogrammetric reconstruction of textured 3D models of buildings. The application uses bounded planes to represent building facades. The planes are approximated from a reconstructed point cloud and initially bounded by the convex hull of the relevant points. Multiple bounded planes may be combined into complex, composite models. The intersection between two or more planes is used to update the bounds of the corresponding planes.

The focus of the thesis has been to create a streamlined operator workflow that reduces operator work time while creating models of sufficient quality. Thus, the main approach is *operator-guided automation* rather than a fully automatic approach. Of course, subproblems are solved automatically wherever appropriate.

Reconstruction results from several buildings of low to high geometric complexity are presented together with the approximate operator work time required for the reconstruction. Furthermore, a time exposure experiment was performed to investigate the effect of the poor lighting conditions common during the winter in northern Sweden.

The results show that the reconstruction is sensitive to a combination of three factors: 1) Low-contrast texture in the foreground, 2) low-contrast texture on the building, 3) poor lighting conditions. However, tripod-mounted cameras and sufficient exposure times are shown to alleviate most of these problems. With images of sufficient quality, the total required operator work time, including photography, is in the order of a few man hours per building.

The thesis concludes with a discussion on how to improve the robustness of the applications, reduce the operator time, and extend the prototype to work with other primitives, e.g. cylinders, as well as predefined composite primitives.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Photogrammetry is the science of measuring objects pictured in photographs. Using methods developed in this science, it is possible to construct 3D models of objects. The aim of this thesis is to create a prototype computer application which enables a user to process digital photographs into 3D models.

1.1 Problem Statement

Oryx Simulations is a company that produces simulation equipment for training purposes. Examples include front loaders, harvesters, and harbour cranes complete with real-life cabins and instruments, allowing an operator to train both basic skills and specific tasks in a safe and conveniently located environment.

The virtual environments are currently being created manually in 3D modelling software. To create a model that closely match a real work site takes many man hours and is expensive. Oryx Simulations is for that reason interested in a process that is more automated and requires less man hours.

1.2 Aims

The primary aim of this thesis is to present a prototype tool for rapid measurements and 3D reconstruction of buildings. Specifically, the following tasks should be considered:

- Image pairing, i.e. determining what images are overlapping and what order to include them in the measurements.
- Automated point matching between paired images.
- Reconstruction of geometric primitives, e.g. planes and cylinders.
- Combining geometrical primitives.
- Reconstruction of the ground using DTM (digital terrain map).

Furthermore, the efficiency of the workflow is a priority in the design of the prototype application. Especially two aspects are considered important:

– All tasks performed by the operator should be made in the application (no external software).

– Operator guided methods should be *aided* by the graphical interface, provide visualization and good overview.

1.3 Related Work

Automatic reconstruction is implemented in Photo tourism: Exploring Photo Collections in 3D[30], where the photos have not been ordered and no calibration information is known. By using heuristics and a lot of computing power, camera poses of these photos may be found without human intervention. A similar paper is [4], where M. Brown and D. G. Lowe also work with unordered, uncalibrated photos. None of these papers explore surface detection or texturing.

Man-made buildings often contain planar surfaces that can be roughly described with a simple linear mapping between photos, e.g. walls or roofs. If two or more of these homographies are known, Q.-T. Luong[20] describes a direct method to obtain the fundamental matrix. Comparing to the 8-point method[11], they find that a) the 8-point method will give better results if there are more than 3 planes, b) their method deteriorates while the 8-point method excels as the number of visible planes is increased and concludes c) that their method can only have usable applications in the case of two planes.

Furthermore in [34], R. Szeliski and P. H. S. Torr experiment with using extra information to improve bundle adjustment between two views. They incorporate facts known from the planes (like point co-planarity) and inter-plane relations (e.g. right angles in intersections and parallelity). Enforcing inter-plane angles removes major error sources like ambiguities in camera orientation, while co-planarity constraints do not do much difference.

The authors of [9] explores the possibilities of semi-automatic fitting of wireframe models to buildings. Although the authors goals are similar to the aims of this thesis, Glüch et al. focus on areal photos. Since areal photos convey less details their methods were not applicable.

On the subject of creating models from point clouds R. Ramamoorthi and J. Arvo[27] describes a tree based method using parameterized operations such as scaling and rotations on user supplied shapes. In [6] A. Fitzgibbon and A. Zisserman proposes using line matching as a complement to point matching. The matched lines provides additional information such as connectivity among the reconstructed points.

1.4 Organization of Thesis

In this thesis, photogrammetry is explained and used to create computer models of real-world environments. The fundamental theory needed to understand the photogrammetric methods is explained in Chapter 2. Furthermore the theoretical workflow of a close range photogrammetric reconstruction is described. In Chapter 3 surface parameterization and modeling is presented. Chapter 4 is more of a “hands on” section with concrete methods of solving the problems described in the previous chapters.

In Chapter 5 implementation details can be found, together with a workflow description and screenshots.

Image acquisition and a description of the camera equipment used is available in Chapter 6, also containing an introduction to the sites that has been shot. The resulting reconstructed models are presented in Chapter 7 together with results from an exposure time experiment

Finally, in Chapter 8, a discussion and analysis of the results and limitations of the thesis can be found. Also suggestions on future works and limitations of the prototype is situated there.

A digital copy of the thesis report in color is available for download at:
`http://www.cs.umu.se/education/examina/Rapporter/ForsNilssonGrundberg.pdf`

Chapter 2

Close Range Photogrammetry

In this chapter, the theory of close range photogrammetry and related topics are introduced.

2.1 Introduction to Projective Geometry

This section is an introduction to basic projective geometry needed to understand the methods described in the later chapters. This section also explains the notation used throughout the thesis. The geometry part of this section is based on [12, Chapter 2] unless otherwise noted.

2.1.1 Points and Lines in $\mathbb{P}^2$

In two dimensional projective space, $\mathbb{P}^2$, a point is represented by a 3-vector

$$\mathbf{x} = \begin{bmatrix} x & y & t \end{bmatrix}^{\mathrm{T}},$$

this is called the *homogeneous representation* or *homogeneous coordinates.* All homogeneous vectors $\mathbf{x} = k \begin{bmatrix} x & y & t \end{bmatrix}^{\mathrm{T}}$, $k \neq 0$ are interpreted as the same object, in this case a point. In $\mathbb{P}^2$ lines are represented by homogeneous 3-vectors like

$$\mathbf{l} = \begin{bmatrix} a & b & c \end{bmatrix}^{\mathrm{T}}.$$

Vectors are represented by boldface letters. A point $\mathbf{x}$ lies on a line $\mathbf{l}$ iff $\mathbf{x}^{\mathrm{T}}\mathbf{l} = 0$ and since the scalar product is commutative, $\mathbf{x}^{\mathrm{T}}\mathbf{l} = \mathbf{l}^{\mathrm{T}}\mathbf{x}$, the scalar product may also be interpreted as a test if a line intersects a given point. The intersection of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$ is the point $\mathbf{x}$ that satisfies $\mathbf{x}^{\mathrm{T}}\mathbf{l_1} = 0$ and $\mathbf{x}^{\mathrm{T}}\mathbf{l_2} = 0$. It is possible to construct such a point using the cross product. The cross product produces a vector that is orthogonal to the input vectors and is written as

$$\mathbf{l_1} \times \mathbf{l_2} = \mathbf{x}.$$

Since points and lines are indistinguishable in $\mathbb{P}^2$, the cross product may be put to other uses as well. For example a line joining the two points $\mathbf{x_1}$ and $\mathbf{x_2}$ may be calculated as $\mathbf{l} = \mathbf{x_1} \times \mathbf{x_2}$.

2.1.2 Conversion to Euclidean Coordinates

A point in $\mathbb{P}^2$ can easily be converted to a point in $\mathbb{R}^2$ as

$$\mathbf{x} = \begin{bmatrix} x & y & t \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \frac{x}{t} & \frac{y}{t} & 1 \end{bmatrix}^{\mathrm{T}} \rightarrow \tilde{\mathbf{x}} = \begin{bmatrix} \frac{x}{t} & \frac{y}{t} \end{bmatrix}^{\mathrm{T}}, \text{for } t \neq 0. \tag{2.1}$$

For $t = 0$, the right hand side of equation (2.1) is undefined. Such points are called points at infinity or *ideal points*. All ideal points lie on the *line at infinity*, $\mathbf{l}_\infty = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$. For example the point $\mathbf{x} = \begin{bmatrix} x & y & 0 \end{bmatrix}^{\mathrm{T}}$ lies on $\mathbf{l}_\infty$, since

$$\mathbf{l}_\infty^{\mathrm{T}}\mathbf{x} = 0 \cdot x + 0 \cdot y + 1 \cdot 0 = 0.$$

Ideal points are not defined in Euclidean space and may be interpreted as directions instead of points. In fact, two parallel lines will actually intersect each other in projective space, but this intersection will lie on the line at infinity. In this thesis vectors representing Euclidean coordinates are annotated with a superscripted tilde sign to distinguish them from projective coordinates.

2.1.3 Second Order Curves in $\mathbb{P}^2$

Parabolas, hyperbolas, ellipses, and circles are all second order curves. These curves and the degenerate cases are all defined by the general equation (where not all coefficients are zero)

$$ax_1^2 + bx_1x_2 + cx_2^2 + fx_1 + gy_1 + d = 0. \tag{2.2}$$

This equation may be turned into a homogeneous equation by substituting x_1 and x_2 so that $x_1 \rightarrow \frac{x}{t}$ and $x_2 \rightarrow \frac{y}{t}$ and then multiplying the equation with t^2 producing

$$t^2\left(a\frac{x^2}{t^2} + b\frac{xy}{t^2} + c\frac{y^2}{t^2} + f\frac{x}{t} + g\frac{y}{t} + d\right) = ax^2 + bxy + cx^2 + fxt + gyt + dt^2. \tag{2.3}$$

The coefficients from the right hand side of equation (2.3) can be stacked into the 3×3 symmetric matrix

$$\mathtt{C} = \begin{bmatrix} a & b/2 & f/2 \\ b/2 & c & g/2 \\ f/2 & g/2 & d \end{bmatrix}.$$

Thus equation (2.2) may be rewritten in a more compact form

$$\mathbf{x}^{\mathrm{T}}\mathtt{C}\mathbf{x} = 0 \tag{2.4}$$

using homogeneous coordinates. Equation (2.4) is the definition of a conic. Conics may represent degenerate curves such as straight lines and points.

2.1.4 3-Dimensional Projective Space

Points in $\mathbb{P}^3$ are represented by homogeneous 4-vectors

$$\mathbf{X} = \begin{bmatrix} x & y & z & t \end{bmatrix}^{\mathrm{T}}.$$

Ideal points in $\mathbb{P}^3$ all lie on the *plane at infinity*, $\boldsymbol{\pi}_\infty = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$. In this thesis points in $\mathbb{P}^3$ are always denoted by a boldface uppercase letter.

2.1.5 Homogeneous Representation of Planes

The homogeneous representation of planes is very similar to the Euclidean space equation

$$\pi_1 x_1 + \pi_2 x_2 + \pi_3 x_3 + \pi_4 = 0.$$

This equation can be extended to include homogeneous points as well, since the multiplication of a scalar leaves the equation unchanged. With the substitutions $x_1 \rightarrow x/t$, $x_2 \rightarrow y/t$ and $x_3 \rightarrow z/t$, we get

$$t(\pi_1 x_1 + \pi_2 x_2 + \pi_3 x_3 + \pi_4) = 0 \Rightarrow \pi_1 x + \pi_2 y + \pi_3 z + \pi_4 t = 0. \tag{2.5}$$

If the plane $\boldsymbol{\pi}$ is defined as $\boldsymbol{\pi} = \begin{bmatrix}\pi_1 & \pi_2 & \pi_3 & \pi_4\end{bmatrix}^{\mathrm{T}}$ and a homogeneous point as $\mathbf{X} = \begin{bmatrix}x & y & z & t\end{bmatrix}^{\mathrm{T}}$ then equation (2.5) can be expressed in terms of inner product

$$\boldsymbol{\pi}^{\mathrm{T}}\mathbf{X} = 0.$$

2.1.6 Lines in $\mathbb{P}^3$

There exists several ways to represent lines in $\mathbb{P}^3$. Two non-coincident points $\mathbf{X}_1$ and $\mathbf{X}_2$ defines the line going through them. Let the matrix W be formed as

$$\mathrm{W} = \begin{bmatrix}\mathbf{X}_1^{\mathrm{T}} \\ \mathbf{X}_2^{\mathrm{T}}\end{bmatrix}$$

so the column space of W^{T} spans the line. In other words the points on the lines can be described as a linear combination of the rows of W,

$$X(\lambda, \mu) = \lambda\mathbf{X}_1 + \mu\mathbf{X}_2.$$

Lines can also be defined by the intersection of two planes. Given two non parallel planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$, form W^* with the planes as its row vectors

$$\mathrm{W}^* = \begin{bmatrix}\boldsymbol{\pi}_1^{\mathrm{T}} \\ \boldsymbol{\pi}_2^{\mathrm{T}}\end{bmatrix}.$$

W^* is similar to W but it is the *null space* of W^* that spans the line of intersection between $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$.

2.1.7 Construction of Plane Coordinate Systems

It is sometimes necessary to create a two-dimensional coordinate system for a plane. This coordinate system describes a mapping from a 2D point in the plane to a 3D point and vice versa.

In $\mathbb{P}^3$ three points define a plane. Since for each of the three points $\boldsymbol{\pi}^{\mathrm{T}}\mathbf{X}_i = 0$, these points can be expressed as a matrix, whose rows are the coordinates of the points. This yields a 3×4 matrix whose null space is

$$\mathcal{N}\begin{pmatrix}\mathbf{X}_1^{\mathrm{T}} \\ \mathbf{X}_2^{\mathrm{T}} \\ \mathbf{X}_3^{\mathrm{T}}\end{pmatrix} = \boldsymbol{\pi}.$$

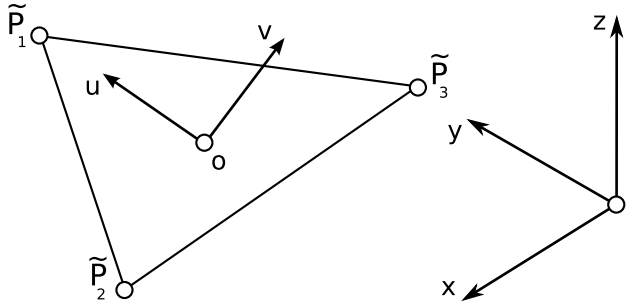


Figure 2.1: Points defining the plane, and its internal basis (**u**,**v**).

To create a coordinate system, a basis for the plane and a reference point is needed. Let the points $\tilde{\mathbf{P}}_1$, $\tilde{\mathbf{P}}_2$ and $\tilde{\mathbf{P}}_3$ define the plane $\boldsymbol{\pi}$. See Figure 2.1. The centroid $\tilde{\mathbf{O}}$ of these points is chosen to map to the origin of the 2D system, i.e.

$$\tilde{\mathbf{O}} = \frac{(\tilde{\mathbf{P}}_1 + \tilde{\mathbf{P}}_2 + \tilde{\mathbf{P}}_3)}{3}.$$

The base vectors for the coordinate system, $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$, are also derived from the points. If $\tilde{\mathbf{U}}$ is chosen as

$$\tilde{\mathbf{U}} = \tilde{\mathbf{P}}_1 - \tilde{\mathbf{O}},$$

$\tilde{\mathbf{V}}$ can be constructed from the Euclidean plane normal $\tilde{\mathbf{N}} = \begin{bmatrix}\pi_1 & \pi_2 & \pi_3\end{bmatrix}^{\mathrm{T}}$ and $\tilde{\mathbf{U}}$, such that $\tilde{\mathbf{V}}$ is orthogonal to $\tilde{\mathbf{U}}$, by letting

$$\tilde{\mathbf{V}} = \tilde{\mathbf{U}} \times \tilde{\mathbf{N}}.$$

The mapping of a 2D point $\mathbf{x}$ in the plane coordinate system to a point $\mathbf{X}$ in world coordinates can be expressed as a matrix-vector multiplication $\mathrm{M}_\pi \mathbf{x} = \mathbf{X}$. The 4×3 matrix M_π is defined as

$$\mathrm{M}_\pi = \begin{bmatrix}\tilde{\mathbf{U}} & \tilde{\mathbf{V}} & \tilde{\mathbf{O}} \\ 0 & 0 & 1\end{bmatrix}$$

in [3]. The pseudo inverse M_π^+ performs the opposite operation, $\mathrm{M}_\pi^+ \mathbf{X} = \mathbf{x}$. If the matrix U is defined as $\mathrm{U} = [\tilde{\mathbf{U}}\ \tilde{\mathbf{V}}]$ then the pseudo inverse can be formed as

$$\mathrm{M}_\pi^+ = \begin{bmatrix}\mathrm{U}^+ \\ \mathbf{0}^{\mathrm{T}}\end{bmatrix} \begin{bmatrix}-\mathrm{U}^+\tilde{\mathbf{O}} \\ 1\end{bmatrix}.$$

2.1.8 Plane-Plane Intersection

As stated earlier, the intersection of two distinct planes is a line in $\mathbb{P}^3$. If $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ is the homogeneous representation of two planes then the line of their intersection is

$$\mathrm{W}^* = \begin{bmatrix}\boldsymbol{\pi}_1^{\mathrm{T}} \\ \boldsymbol{\pi}_2^{\mathrm{T}}\end{bmatrix}.$$

The span of the 2D null space of W^* is the line of intersection. If the planes are parallel then the line of intersection will lie on $\boldsymbol{\pi}_\infty$.

2.1.9 Second Order Surfaces

A seconds order surface, a quadric, is defined by the general equation

$$ax^2 + bx^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2px + 2qy + 2rz + d = 0.$$

This equation can be expressed as a 4×4 symmetric matrix

$$\mathtt{Q} = \begin{bmatrix} a & h & g & p \\ h & b & f & q \\ g & f & c & r \\ p & q & r & d \end{bmatrix}.$$

All homogeneous points on the surface defined by the quadric $\mathtt{Q}$ satisfies

$$\mathbf{X}^{\mathrm{T}}\mathtt{Q}\mathbf{X} = 0.$$

A rank deficient quadric matrix, also called a degenerate quadric, can represent a line, plane, point, or no points at all.

2.1.10 Intersection of a Second Order Surface and a Plane

An example of a degenerate quadric is the elliptic cylinder and is defined by the equation

$$\frac{x^2}{\alpha^2} + \frac{y^2}{\beta^2} = 1. \tag{2.6}$$

This equation may be rewritten to the general form

$$\beta^2 x^2 + \alpha^2 y^2 - \beta^2\alpha^2 = 0,$$

which gives the quadric $\mathtt{Q}$ of rank 3,

$$\mathtt{Q} = \begin{bmatrix} \beta^2 & 0 & 0 & 0 \\ 0 & \alpha^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\beta^2\alpha^2 \end{bmatrix}.$$

The intersection of a quadric $\mathtt{Q}$ and a plane $\boldsymbol{\pi}$ is a point conic on the plane. Since $\mathbf{X}^{\mathrm{T}}\mathtt{Q}\mathbf{X} = 0$ if $\mathbf{X}$ lies on $\mathtt{Q}$ and since $\mathbf{X}$ can be defined as $\mathbf{X} = \mathrm{M}_\pi\mathbf{x}$ one can see that if $\mathbf{X}$ lies on both $\boldsymbol{\pi}$ and $\mathtt{Q}$ then

$$\mathbf{x}^{\mathrm{T}}\mathrm{M}_\pi^{\mathrm{T}}\mathtt{Q}\mathrm{M}_\pi\mathbf{x} = 0 \tag{2.7}$$

where

$$\mathbf{X}^{\mathrm{T}} = (\mathrm{M}_\pi\mathbf{x})^{\mathrm{T}} = \mathbf{x}^{\mathrm{T}}\mathrm{M}_\pi^{\mathrm{T}}.$$

If $\mathtt{C}_\pi$ is defined as $\mathtt{C}_\pi = \mathrm{M}_\pi^{\mathrm{T}}\mathtt{Q}\mathrm{M}_\pi$ then equation (2.7) can be rewritten as

$$\mathbf{x}^{\mathrm{T}}\mathtt{C}_\pi\mathbf{x} = 0$$

which is the definition of a conic. As described in Chapter 2 a conic defines a second order curve in a plane. In the general case this plane is the x-y plane.

The intersection of a cylinder and a plane can result in four different curves: an ellipse, a circle, and two degenerate curves; coincident lines, and two parallel lines. Figure 2.2 shows two of these possibilities. The elliptic curve is the intersection of a plane which is tilted relative to the cylinder. The circle can be considered a special case of the ellipse with $a = b$, that occurs when the plane is perpendicular to the cylinder axis. When the plane is parallel to the cylinder axis, the intersection will consist of two parallel lines. These lines will become incident if the plane become a tangent to the cylinder.

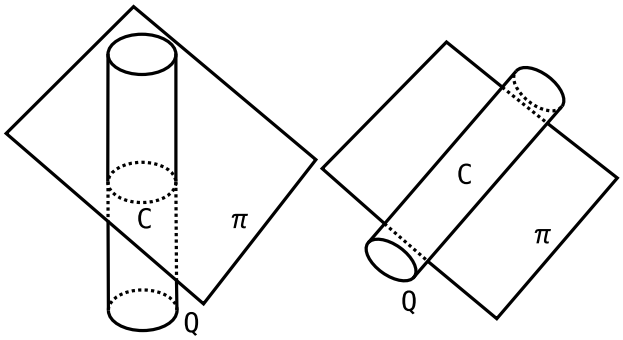


Figure 2.2: Intersection of quadrics and planes with resulting conics.

2.1.11 Classification of Conics

In [36] two approaches to conic classification are described. The first approach determinates what kind of curve the conic represents by partitioning $\mathtt{C}$. Recall that $\mathtt{C}$ is defined as

$$\mathtt{C} = \begin{bmatrix} a & b/2 & f/2 \\ b/2 & c & g/2 \\ f/2 & g/2 & d \end{bmatrix}.$$

Let $\mathtt{C}$ be partitioned into the parts

$$\Delta = |\mathtt{C}|\,,\ I = a + c,\ J = \begin{vmatrix} a & b/2 \\ b/2 & c \end{vmatrix} \text{ and, } K = \begin{vmatrix} a & f/2 \\ f/2 & d \end{vmatrix} + \begin{vmatrix} c & g/2 \\ g/2 & d \end{vmatrix}.$$

With the help of these partitions it is possible to decide which kind of curve the conic $\mathtt{C}$ is representing by consulting Table 2.1.

Table 2.1: Classification table (Purely imaginary solutions are omitted). From [36].

Conic type	Δ	J	$\frac{\Delta}{I}$	K
Hyperbola	$\neq 0$	< 0		
Parabola	$\neq 0$	0		
Ellipse	$\neq 0$	> 0	< 0	
Intersecting lines	0	< 0		
Point	0	> 0		
Distinct parallel lines	0	0		< 0
Coincident lines	0	0		0

The other approach presented is best described as an algorithm and can be found in Appendix A.

2.1.12 Transformations in Projective Space

A transformation or homography is a straight line preserving mapping of $\mathbb{P}^2$ onto itself. A transformation in $\mathbb{P}^2$ is defined as a $\mathrm{H}\mathbf{x} = \mathbf{x}'$ or

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix}.$$

Transformations can be classified by describing what properties of the coordinate system that stay invariant. Here four distinct types of transformations will be discussed as well as their invariants.

The first and the most basic transformation is the *Euclidean transformation*. This transformation is composed of a translation and a rotation and can be written

$$\mathrm{H}_\mathrm{E} = \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}.$$

The R block of H_E is a 2×2 rotation matrix and $\mathbf{t}$ is a translation vector. A Euclidean transformation will not change the length or area of the transformed object. This transformation has three degrees of freedom; the rotation angle and the translation in the x and y directions.

A *similarity transform* is a Euclidean transform with a scale factor σ resulting in

$$\mathrm{H}_\mathrm{S} = \begin{bmatrix} \sigma\mathrm{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}.$$

The scale factor adds one degree of freedom to a total of four. This also affects the invariants of this transformation. Lengths and areas are no longer invariant, but the *ratio* of lengths and the *ratio* of areas are unchanged under this transformation.

The third class of transformations is the *affine transformation* also called an *affinity*

$$\mathrm{H}_\mathrm{A} = \begin{bmatrix} \mathrm{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

and has six degrees of freedom. Two degrees from the translation and four from 2×2 affine block A. The matrix A must be non-singular, i.e. invertible.

Since the affine transformation is not limited to a rotation, the invariant properties are different from the previous two transformations. Orthogonality is not preserved. However parallelism is, i.e. two parallel lines will still be parallel after an affinity.

The fourth and most general form of transformations in $\mathbb{P}^2$ is the *projective transformation* and has eight degrees of freedom. The matrix

$$\mathrm{H}_\mathrm{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

is homogeneous, meaning that it is determined up to a scale, that is why it only has eight degrees of freedom instead of nine. The only requirement on H_P is that it is invertible. The largest difference from the earlier transformations is that a projective transformation does not preserve parallelism.

2.2 Cameras and Camera Models

The pinhole camera is one of the simplest forms of cameras. It is an opaque box with a hole in one of the sides. The image plane is situated at the opposite side of the pinhole. The world outside the box is projected onto the image plane through the hole.

A light sensitive surface that resides on the image plane accumulates the light. This surface could for example be a film or a charge coupled device (CCD) for a digital camera.

The pinhole camera is used as a mathematical model to describe how a camera transforms "real world" 3-dimensional points to 2-dimensional points in an image, or

$$\mathbf{X}_{world} \rightarrow \mathbf{x}_{image}.$$

In the mathematical model used in this thesis, the point where all light rays intersects is called the *focal point* or camera center. This point is analogous to the pinhole. The image plane is modeled in front of the focal point in contrast to the pin hole camera, where the image sensing device is situated behind the pin hole. This simplification lets us ignore the fact that the image actually is turned upside down in the acquisition process.

Let the focal point be the origin of the 3D coordinate system and the Z-axis be the direction in which the camera is facing as seen in, Figure 2.3. Now the mapping from a 3D point with the coordinate (X, Y, Z) to a 2D point (x, y) can be described through this mapping[12, p. 154]:

$$(X, Y, Z) \rightarrow \begin{cases} x = \dfrac{fX}{Z} \\ y = \dfrac{fY}{Z} \end{cases} \tag{2.8}$$

The mapping in equation (2.8) can also be described in matrix form as the *camera matrix*

$$\mathrm{P} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.9}$$

f is the *focal length* of the camera, the distance from the focal point $\mathbf{C}$ to the principal point $\mathbf{p}$. The principal point is the intersection of the image plane with the Z-axis. The parameters f and $\mathbf{p}$ are often given in mm.

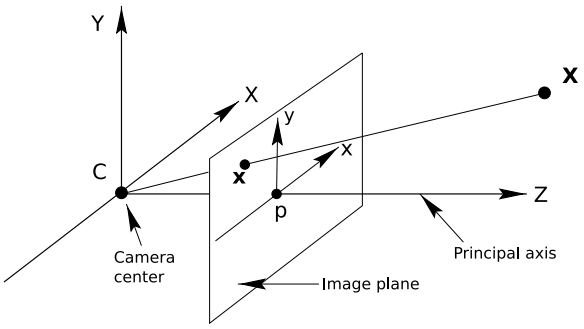


Figure 2.3: Diagram of a mathematical camera model. Coincidentally also shows the pose of a canonical camera. From [12], used with permission.

The mapping in equation (2.8) can now with the matrix P from (2.9) be turned into a simple matrix-vector multiplication $\mathrm{P}\mathbf{X} = \mathbf{x}$.

2.2.1 Internal Parameters

The camera matrix P is often factored into two parts. The first part, K, represents the internal parameters of the camera, e.g. the focal length. This matrix is often called the *camera calibration matrix*. The other part of P represents the external parameters, i.e. the position and rotation of the camera, relative to the world[12, p. 155]. The internal parameters of the mapping in equation (2.8) can be expressed as

$$\mathrm{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The model described so far assumes two things about the configuration of the camera. First that the principal point is the origin of the image. This is often not the case as most digital image coordinate systems have the origin in the top left corner. A translation of each point image coordinate (p_x, p_y) is needed. This leads to a new camera calibration matrix

$$\mathrm{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}.$$

The other assumption made is that the image sensing devices on the detector plane are square. In other words the x-axis and the y-axis in the image plane must be equal in magnitude. This can be remedied by adding a "stretch" factor. Let m_x and m_y be this factor for the x-axis respectively y-axis given in pixels/mm. By including these factors in K the model can compensate for the scaling[12, pp. 156 - 157]. The notation of K is simplified by substituting the products with the new variables α_x, α_y, x_0, and y_0 as

$$\mathrm{K} = \begin{bmatrix} m_x f & 0 & m_x p_x \\ 0 & m_y f & m_y p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The units of α_x, α_y, x_0, and y_0 are in pixels.

The last and final internal parameter in the camera model is the skew factor s. This variable can be thought of as representing a camera where the x- and y-axes in the image plane are not orthogonal. Although this is not likely to be the case for a digital camera it completes the model, letting

$$\mathrm{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2.2.2 External Parameters

In the camera model described above the camera center is in the center of the coordinate system and the camera coordinate system is aligned with the world coordinate system. To enable arbitrary placement of the camera, a set of external parameters are added to the camera model. The rotation of the camera is described by a rotation matrix R and

the translation part is formed from the Euclidean representation of the camera center. The final model

$$\mathrm{P} = \mathrm{KR}[\mathrm{I} \mid -\tilde{\mathbf{C}}]$$

is called the *finite projective camera model*.

2.2.3 Lens Distortion

The finite projective camera model is a linear mapping from a world coordinate $\mathbf{X}$ to an image point $\mathbf{x}$. As seen in Figure 2.3, this mapping is a straight line.

However, a real camera contains lenses that bends this straight line. This error in the model is called lens distortion[12, p. 189]. Lens distortion cannot be described by a linear function, thus it cannot be introduced into the camera model as such. In photogrammetry, it is common to divide lens distortion into two distinct components, *radial distortion* and *tangential distortion*[19].

Radial Distortion

Radial distortion is present in all lens systems. Radial distortion displace image points. The displacement is function of the distance to the principal point $\mathbf{p}$ of the camera. This effect can be seen in ordinary photographs. For example in photographs of buildings, corners near the edges of the photographs can appear to be curved. The diagram in Figure 2.4 shows the effect of this error.
The radial lens distortion can be modeled using the polynomial[21, pp. 297 - 298]

$$\Delta r = k_0 r + k_1 r^3 + k_2 r^5 + k_3 r^7$$

where

$$r = ||\tilde{\mathbf{x}} - \tilde{\mathbf{p}}||_2$$

is the distance from a point to the principal point, and the coefficients k_0, k_1, k_2, and k_3 are determined in a calibration process. The correction function for an image point $\mathbf{x}$ is defined as

$$\mathbf{x}' = \begin{bmatrix} x\left(1 - \dfrac{\Delta r}{r}\right) \\ y\left(1 - \dfrac{\Delta r}{r}\right) \end{bmatrix}.$$

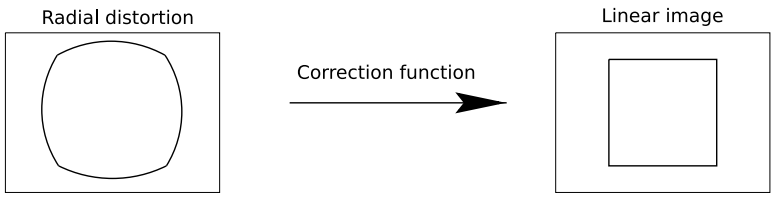


Figure 2.4: Lens distortion and correction. From [12], used with permission.

Tangential Distortion

Unlike the radial distortion, tangential distortion has a decentering effect which varies with direction of the point $\mathbf{x}$ with respect to the principal point. This distortion is an effect of poorly centered lenses in the camera objective. The mathematical model to describe this distortion is more complex and is described in detail in [21, pp. 299 - 300].

2.3 Two-view Geometry

An object imaged in two photos can be measured by *triangulation*, where relative 3D coordinates can be produced. This requires the location and orientation (the *pose*) of both cameras to be known. However, by matching points between the two photos, it is possible to calculate the second pose in relation to the first pose (which could be fixed to any arbitrary value). This *relative orientation* makes it possible to triangulate 3D coordinates.

The epipolar geometry describes the relationship between two views. This relationship depends only on the internal parameters of the cameras and their poses[12, p. 239].

The line between the two focal points ($\mathbf{C}$ and $\mathbf{C}'$) is called a *baseline*. An epipolar plane is defined as the plane spanned by the three points $\mathbf{C}$, $\mathbf{C}'$, and an arbitrary point $\mathbf{X}$. Figure 2.5 depict an epipolar plane.
Each point $\mathbf{X}$ defines an epipolar plane. An epipolar plane intersects each of the views, the line of intersection is called the epipolar line, $\mathbf{l}$ in left view and $\mathbf{l}'$ in the right view. The 2D points in the image planes, $\mathbf{x}$ (left) and $\mathbf{x}'$ (right) must lie on their epipolar lines. This is called the *epipolar constraint*

$$\begin{cases} \mathbf{l}^{\mathrm{T}}\mathbf{x} = 0 \\ \mathbf{l}'^{\mathrm{T}}\mathbf{x}' = 0 \end{cases} .$$

The intersection of the baseline and the image plane is called an epipole. The epipole $\mathbf{e}$ is the projection of the right view's focal point in the left view's image plane. On the opposite image plane the epipole $\mathbf{e}'$ is the projection of the left focal point. All epipolar lines intersect the epipole.

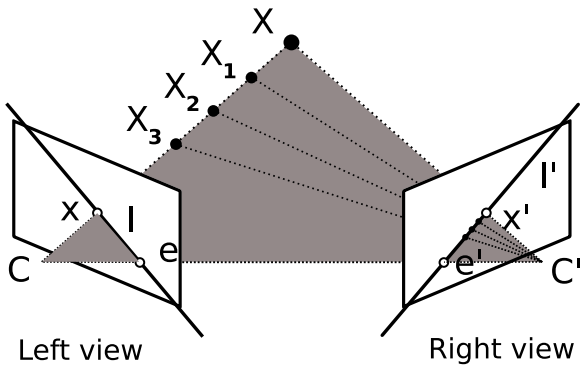


Figure 2.5: An epipolar plane of two views. Illustration by Arne Nordmann, used with permission.

2.3.1 The Fundamental Matrix

The epipolar geometry may be described by a 3×3 homogeneous matrix F. This matrix is called the *fundamental matrix*. For the complete derivation of the fundamental matrix see [21, p. 247] or [12, pp. 246 - 247]. However, one important result is that F is directly dependent on the camera matrices P and P′ as

$$\mathrm{F} = [\mathbf{e'}]_{\times}\mathrm{P'P}^{+}$$

where $[\mathbf{e'}]_{\times}$ is the cross product matrix of $\mathbf{e'}$, and P^{+} is the pseudo inverse of P.

The following relations between the point $\mathbf{x}$ in the left view and the epipolar line $\mathbf{l'}$ in the right view can be described in the mapping

$$\mathbf{l'} = \mathrm{F}\mathbf{x}.$$

The corresponding mapping from a point in the right view to an epipolar line in the left view is found through F^{T} and thus

$$\mathbf{l} = \mathrm{F}^{\mathrm{T}}\mathbf{x'}.$$

From these equations one can derive that for each matched point pair, $\mathbf{x} \leftrightarrow \mathbf{x'}$,

$$\mathbf{x'}^{\mathrm{T}}\mathrm{F}\mathbf{x} = 0 \tag{2.10}$$

will hold. Since $\mathbf{l'} = \mathrm{F}\mathbf{x}$ equation (2.10) can be rewritten as

$$\mathbf{x'}^{\mathrm{T}}\mathbf{l'} = 0$$

which holds iff $\mathbf{x'}^{\mathrm{T}}$ lie on $\mathbf{l'}$ the epipolar line. This again is the epipolar constraint.

2.3.2 The Essential Matrix

The essential matrix[16] E is a 3×3 matrix with similar properties as F. But it removes the calibration information from the equations. E only has five degrees of freedom compared to the seven degrees in F [12, p. 257].

Consider a point $\mathbf{x}$ that has been mapped with P through $\mathrm{P}\mathbf{X} = \mathbf{x}$ and that P is defined as $\mathrm{P} = \mathrm{KR}[\mathrm{I} \mid -\tilde{\mathbf{C}}]$. The point $\mathbf{x}$ must be transformed to its *normalized form* $\hat{\mathbf{x}}$[12, p. 257]. To convert a point to its normalized form one multiplies it with the inverse of the calibration matrix used to take the photograph, i.e.

$$\hat{\mathbf{x}} = \mathrm{K}^{-1}\mathbf{x}.$$

This normalization operation can be thought of as removing all the calibration information from the point $\mathbf{x}$. With normalized coordinates the same mappings as with F can be performed with E i.e.

$$\hat{\mathbf{x}}'^{\mathrm{T}}\mathrm{E}\hat{\mathbf{x}} = 0.$$

The relation still requires that the point correspondence $\hat{\mathbf{x}} \leftrightarrow \hat{\mathbf{x}}'$ is known.

2.4 Error Minimization and Estimation

Error minimization or optimization is often applied when trying to solve a problem where finding the solution will take hours, days or even longer. Methods used to perform the minimizations depend on the nature of the problem. For this reason there exist a myriad of different solvers with different strengths and shortcomings.

The ideal situation would be if the measured data exactly fit into the model. This is usually not the case, and many of the algorithms previously mentioned fit data to a model by minimizing an error measure. One example is the epipolar geometry, where the matching points contain measurement errors.

Consider a model $f(x)$, that depend on some parameter x. From a measurement b it may not be possible to estimate $\hat{x}$ so that $f(\hat{x}) = b$. But it is possible to construct r so that $r(\hat{x}) = f(\hat{x}) - b$ and by minimizing this function an estimation $\hat{x}$ can be made.

The most common error measure is the sum of the squared distances. The squared distance might not be the most intuitive error measurement but it is optimal if the errors are normal distributed[25, p. 246].

If the problem one is trying to solve is (partially) differentiable then a Newton method can be used like Levenberg-Marquardt or Gauss-Newton. These methods need a starting point, this is called the initial guess. This initial guess is often found using a direct solution. The optimization method iteratively changes the parameters to minimize the error, thus improving the solution. More information about non-linear optimization methods can be acquired in [25].

In some scenarios, the measurements contains wildly outlying data points that can not be modeled. In e.g. the calculation of the epipolar geometry, the matching points may be mismatched. Fitting all data points then become impossible. One has to classify the measurements as inliers or outliers. The model is then fitted to the inliers only. Algorithms that classify points are called *robust methods*.

2.4.1 Linear Least Squares Approximation

The *least squares* method is used to approximate an overdetermined system of equations. Usually this means fitting a linear model to a set of measured data points[1, p. 469]. Given n measured data points $(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$ to be fitted against the 2 parameter model $y = \alpha + \beta x$ one can form a series of equations

$$\begin{aligned} \alpha + \beta x_1 &= y_1 \\ \alpha + \beta x_2 &= y_2 \\ &\vdots \\ \alpha + \beta x_n &= y_n. \end{aligned}$$

This equation system can be rewritten in matrix form

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

or $\mathbf{Ax} = \mathbf{b}$. If $n > 2$ the system is said to be overdetermined, there are more equations that unknowns. An overdetermined system cannot be solved directly unless the measured points are collinear[26, p. 105].

When there is no solution to $\mathrm{A}\mathbf{x} = \mathbf{b}$ an approximation of $\mathbf{x}$ may be sought. The least squares approximation is the solution $\hat{\mathbf{x}}$ that minimizes $||\mathrm{A}\mathbf{x} - \mathbf{b}||_2^2$. To find $\hat{\mathbf{x}}$ the *normal equations* are formed as

$$\mathrm{A}^{\mathrm{T}}\mathrm{A}\hat{\mathbf{x}} = \mathrm{A}^{\mathrm{T}}\mathbf{b}$$

and solved for $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = (\mathrm{A}^{\mathrm{T}}\mathrm{A})^{-1}(\mathrm{A}^{\mathrm{T}}\mathbf{b}).$$

2.4.2 Orthogonal Distance Regression

Consider a cloud of points, which we know approximately lie on a common plane. We want to estimate an equation for such a plane. More specifically, if these points are projected to a plane, how do we find the plane that minimizes the distances between the original positions and the projections? If the measurement we want to minimize is the sum of the (squared) distance of each point, the problem is called *orthogonal distance regression* (ODR).

Raleigh Quotient

In order to solve the ODR problem, we will need the *Raleigh quotient*, defined as[33]

$$R(\mathrm{A}, \mathbf{x}) = \frac{\mathbf{x}^{\mathrm{H}}\mathrm{A}\mathbf{x}}{\mathbf{x}^{\mathrm{H}}\mathbf{x}}$$

where A is an Hermitian matrix and $\mathbf{x}^{\mathrm{H}}$ is the Hermitian conjugate of an arbitrary vector $\mathbf{x}$. All symmetric real matrices are Hermitian. For real values of $\mathbf{x}$ the Hermitian conjugate is simply $\mathbf{x}^{\mathrm{T}}$. For the purposes of this thesis we will only be considering real values.

Consider A fixed. Now the only independent variable of the quotient is $\mathbf{x}$. It can be shown that R reaches its minimum value when $\mathbf{x}$ equals the eigenvector of A, corresponding to the smallest eigenvalue[33].

Minimization Problem

The orthogonal distance between a Euclidean point $\tilde{\mathbf{X}}$ and its projection on the plane $\boldsymbol{\pi}$ is

$$dist(\boldsymbol{\pi}, \tilde{\mathbf{X}}) = \frac{|\pi_1 x + \pi_2 y + \pi_3 z + \pi_4|}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}}. \tag{2.11}$$

The fitted plane will however always contain the centroid point $\bar{\tilde{\mathbf{X}}}$, thus we can describe π_4 as

$$\pi_4 = -(\pi_1\bar{x} + \pi_2\bar{y} + \pi_3\bar{z}).$$

Substitute $\boldsymbol{\pi}$ with the set $\{\tilde{\mathbf{n}}, \pi_4\}$, where $\tilde{\mathbf{n}} = (\pi_1, \pi_2, \pi_3)$ is unknown. Substituting π_4 in the right-hand side of equation (2.11) we get

$$dist(\boldsymbol{\pi}, \tilde{\mathbf{X}}) = \frac{|\pi_1(x - \bar{x}) + \pi_2(y - \bar{y}) + \pi_3(z - \bar{z})|}{\sqrt{\pi_1^2 + \pi_2^2 + \pi_3^2}},$$

which can be formulated with vectors if it is squared, as

$$dist^2(\boldsymbol{\pi}, \tilde{\mathbf{X}}) = \frac{(\tilde{\mathbf{n}}^{\mathrm{T}}(\tilde{\mathbf{X}} - \bar{\tilde{\mathbf{X}}}))^2}{\tilde{\mathbf{n}}^{\mathrm{T}}\tilde{\mathbf{n}}}.$$

We want to add up the squared distance of all points, and minimize this sum. The minimization problem for n points is

$$\min_{\tilde{\mathbf{n}}} \sum_{i=1}^{n} dist^2(\boldsymbol{\pi}, \tilde{\mathbf{X}}_i) = \min_{\tilde{\mathbf{n}}} \left[(\tilde{\mathbf{n}}^{\mathrm{T}}\tilde{\mathbf{n}})^{-1}\tilde{\mathbf{n}}^{\mathrm{T}} \left(\sum_{i=1}^{n} (\tilde{\mathbf{X}}_i - \bar{\tilde{\mathbf{X}}})(\tilde{\mathbf{X}}_i - \bar{\tilde{\mathbf{X}}})^{\mathrm{T}} \right) \tilde{\mathbf{n}} \right].$$

Rewriting the problem by constructing a matrix

$$\mathrm{K} = \begin{bmatrix} \tilde{\mathbf{X}}_1 - \bar{\tilde{\mathbf{X}}} & \tilde{\mathbf{X}}_2 - \bar{\tilde{\mathbf{X}}} & \dots & \tilde{\mathbf{X}}_n - \bar{\tilde{\mathbf{X}}} \end{bmatrix}$$

consisting of all translated Euclidean points as column vectors. The problem formulation is now

$$\min_{\tilde{\mathbf{n}}} \frac{\tilde{\mathbf{n}}^{\mathrm{T}}\mathrm{K}\mathrm{K}^{\mathrm{T}}\tilde{\mathbf{n}}}{\tilde{\mathbf{n}}^{\mathrm{T}}\tilde{\mathbf{n}}}.$$

Since $\mathrm{K}\mathrm{K}^{\mathrm{T}}$ is symmetric, the problem now visibly consists of a Raleigh quotient.

Thus we want to find the minimum of the Raleigh quotient $R(\mathrm{K}\mathrm{K}^{\mathrm{T}}, \tilde{\mathbf{n}})$. As expressed earlier, the Raleigh quotient is minimized by the eigenvector of $\mathrm{K}\mathrm{K}^{\mathrm{T}}$ corresponding to the smallest eigenvalue.

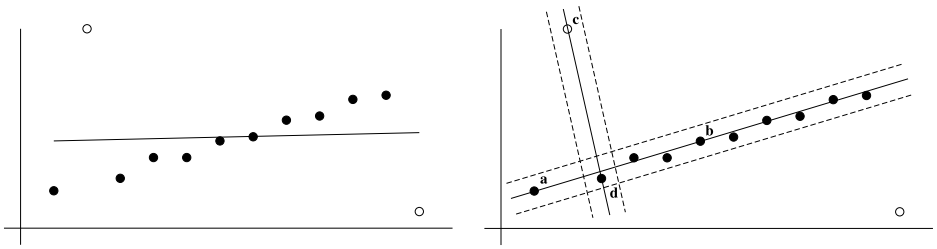
Let $\mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}$ be the singular value decomposition of K. The *eigenvalue decomposition* of $\mathrm{K}\mathrm{K}^{\mathrm{T}}$ can then be written as $\mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}(\mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}})^{\mathrm{T}} = \mathrm{U}\Sigma^2\mathrm{U}^{\mathrm{T}}$. U is the eigenvector matrix and Σ^2 is the diagonal eigenvalue matrix. The eigenvectors of $\mathrm{K}\mathrm{K}^{\mathrm{T}}$ are stored in the columns of U, sorted in descending order by their eigenvalue[33, p. 355].

In conclusion, the ODR problem is solved by constructing the K matrix and calculating the SVD of it. The vector minimizing the objective function is the last row of U, or simply $\mathbf{u}^3 = \boldsymbol{\pi}$.

2.4.3 Robust Methods

The presence of outliers can seriously perturb fitting, especially when minimizing squared distances. Consider the simple 2-parameter problem of fitting a linear equation to a set of points. One point, to the eye clearly not part of any line, could seriously skew the fitted function. Thus the fitting is not *robust* in the presence of outliers.

A *robust method* tries to identify the outliers and only fit to measurements that are actually part of the expected function. There are several methods available, least median of squares, M-estimators etc. One reoccurring method in photogrammetry is



(a) Fit on all points (b) Two RANSAC samples (a and b, c and d)

Figure 2.6: Naive fitting versus robust fitting. From [12], used with permission.

RANSAC (*RANdom Sample Consensus*) which will be described in more detailed in the following section.

RANSAC

RANSAC is an iterative method for robust estimation. RANSAC needs three inputs: First a model which the data is to be fitted against. Furthermore it needs two thresholds t and T. t is the maximum distance a data point can be within and still be considered an inlier. T is the minimum number of inliers needed for the solution to be considered a good approximation of the data. The third and final input is the data to be fitted against the model.

Lets use a 2D line like the one in Figure 2.6 as an example. The model to describe the line is $y = ax+b$ the input is a set of noisy point data, i.e. the points are all supposed to lie on the line, but they do not. RANSAC's objective is to determine the parameters a and b which makes the line pass a close as possible to all the inlier points. A point is deemed to be an inlier if the orthogonal distance is shorter than t^2 units from the line.

The first step in RANSAC is to randomly select a number of data points from the input set. The number of points vary with the model, two points is the smallest amount of data points to define a line, so in this case two points are selected, this is called a sample. From this sample the parameters a and b are calculated. Figure 2.6(b) shows two lines from two samples.

When the sample has been fitted to the model the next step is to decide which of the data points in the input are inliers and outliers. Again this varies with the model, in the case of the 2D line the orthogonal distance squared is the preferred measure. The distance from data point i to the line is $d_{i\perp}^2$. By forming a simple classification function

$$\begin{cases} d_{i\perp}^2 & < & t^2 & \text{inlier} \\ d_{i\perp}^2 & \geq & t^2 & \text{outlier} \end{cases}$$

each point i is classified either as an outlier or an inlier.

The inliers are collected in a set called the *consensus set*, S. If the consensus set is large enough, $|S| > T$, then the approximation of the model is sufficient for the next step. Otherwise RANSAC discards the consensus set and selects a new random sample and the process is repeated.

When the consensus set is large enough a final approximation is made of the data. All the data points in the consensus set is used to make a better approximation of the parameters. In the case of the 2D line an ODR approach would yield the final result.

On a side note, it is often a good idea to include one more input threshold in RANSAC, N which is the maximum number of iterations, if no sufficiently good solutions has been found after N attempts RANSAC simply produces a solution from the largest consensus set found.

Adaptive RANSAC

The adaptive RANSAC is an extended version of the regular RANSAC. This approach uses a statistical model to determinate the values of the thresholds instead of having them fixed. Variable thresholds is very useful when the size and signal to noise ratio of data varies a lot.

In [12, pp. 119 - 121] a model for selecting the size of N based on the probability of selecting a sample free of outliers is derived. p is the probability that at least on of the

N samples selected will be free of outliers. The authors recommend choosing $p = 0.99$. N is defined as

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}$$

where s is the number of data points in each sample. ϵ is defined as

$$\epsilon = 1 - \frac{|S|}{|D|}$$

where $|S|$ and $|D|$ is the size of the consensus set respectively the size of the data set. The value of T, the limit on the consensus set size, can also be chosen in an adaptive way

$$T = (1-\epsilon)|D|.$$

The final threshold t can be approximated given that one has knowledge about the distribution of the noise that is included in the input. It is shown how this can be done in [12, p. 120].

2.5 Photogrammetric Reconstruction

This section is an introduction of the photographic reconstruction process, which results in a point cloud of discrete points. The necessary steps are presented, together with a presentation of the parameters required for each step.

Network Design The number of overlapping photos and how the photos are shot are critical to the quality of the reconstruction. It is vital to capture the object completely and accurately. Failure to plan ahead may lead to difficulties that cannot be corrected without going back and shooting the site again.

Calibration, On-site Photographing and Rectification Calibration is used to determine the lens distortion and (in our case) to determine the internal parameters of each camera used. On-site photographing is the actual photographing of the site, while rectification removes the lens distortion from the images. These topics are only briefly discussed in this thesis.

Matching Corresponding points between images need to be matched. The robustness of this step affects the quality of the following steps, since the data acquired here are used throughout the other steps.

Determining Epipolar Geometry An essential or fundamental matrix is calculated using the matched points acquired in the previous step.

Relative Orientation The camera matrices are estimated using the epipolar geometry. Since it is not possible to know the size or location of the depicted objects, some of the parameters in the camera matrices will be fixed.

Triangulation Calculates the 3D coordinates of matched points. Beside the matched points, this also requires the camera matrices.

Registration Combine the triangulated coordinates to a common coordinate system.

Error minimization Minimize the combined error of all the 3D coordinates, this is usually referred to as *bundle adjustment*.

Absolute Orientation Use control points with known coordinates to transform the Euclidean reconstruction to an external coordinate system.

2.5.1 Network Design

It is important to make sure the cameras not only capture the material needed but also be set in correct angles to make triangulation possible and ease feature point matching. Some factors are competing, i.e. to get a reliable triangulation a wide baseline is needed, while the automatic feature point matching is simplified by a narrow baseline. An object need to be shot considering both aspects. The shots are often taken moving around the object. In order to get many feature points, there must be photos taken close to each other, only moving a short distance along the object (narrow baseline). Wide baseline shots are optimally taken at distances 1/4 of the circumference, i.e. at 90 degree angle between the principal axes.

It should be noted that there are often objects blocking the view of the sought object, this should also be considered when planning.

The rules how to layout shots involve heuristics rather than strict rules. Lerma et al.[13] describes the 3x3 rules. These rules consists of three categories all with three basic rules in each[5].

The geometric rules are to prepare control points (plumb lines and distances), place shots in a circle around the object (wide baseline, preferably at a height half of the object) and arranging stereo views properly (narrow baseline, avoiding degenerative configurations).

The photographic rules stress the need to have a controlled camera calibration (no zooming, no auto-focus), planning for illumination (or artificially changing it on site) and selecting the right camera for the work (wide angle preferred).

The organizational rules state advice about bookkeeping. First, a proper sketch must be done, with camera positions and directions decided. Secondly, while photographing, documenting at least the calibration and camera position is necessary for cataloging. The third rule is reminder to verify everything before leaving the site.

2.5.2 Matching

The image correspondence problem is the problem of finding out which and how the pictured elements in one photo relates to the elements in another photo. More precisely, finding pairs of points between the images that belong to each other. Most seeing creatures are hardwired to solve this problem, hence it is tempting to think of this as a trivial problem. Unfortunately it turns out to be a difficult problem.

The usual approach to this problem is to first find places in an image – *feature points* – that have distinct *features*[10]. There is something special with the image intensity at these coordinates with surrounding neighborhood. Something that discerns it from other points in the same image, while still making it possible to match with features from another image. The detectors not only find feature points, they also describe the points in some way, providing extra data for the matching process.

Once these points have been *detected*, one finds point correspondences in two photos by comparing feature points between the two images. There are a lot of different schemes

to *match* feature points. Since different feature point detectors neither find features at the same places nor have the same kind of descriptor data, the match method is often specialized on the characteristics of one particular feature point detector.

The difficulties with detection and matching includes but are not limited to perspective distortion and illumination changes. Some feature point detection and matching schemes also take into account properties of the whole image – they do not blindly just look at the individual points.

So what signifies a good feature point? Förstner[7] defined four properties that a good feature point should have.

Distinctness A point should be distinct in relation to its neighborhood in the image.

Invariance A feature should be invariant to geometrical transformations, e.g. a change in scale should not affect the feature point. This is to make it possible to match points in photos where the perspective has changed.

Stability A feature should be expected to be visible in other photos.

Seldomness While the distinctness criteria states that a point should be unique with respect to its neighbors, the seldomness property states that it should be unique with respect to the whole image. This is to avoid repetitive patters in the photo which could lead to false matches.

Feature Detectors

The *scale-invariant feature transform* method, SIFT, covers both feature point detection and feature matching. Here follows a short description based on [18].

In order to find the same features at different zoom levels, the image is resampled in different resolutions. This is often described as an image pyramid. This makes the feature matching scale invariant. Each layer in the pyramid is called an *octave*.

Furthermore, each octave consists of a *scale space*, where the image is exposed to different levels of Gaussian blur. Each level is subtracted to the next level, resulting in new levels stored in the space of difference-of-Gaussian (*DoG space*).

To find points of interest, the DoG space is searched for extreme points, i.e. points that are stronger than their neighbors in the octave and stronger than the same point in adjacent levels. Points may be rejected if they are not distinct enough, e.g. have too low contrast or they are situated along an edge.

In order to make the features rotation invariant, SIFT calculates both an intensity gradient and orientation grid local to the point. An orientation histogram is calculated from the grids, where cells close to the point are weighted higher than points far away. The highest peak of the orientation histogram determines the dominant orientation.

The feature descriptors are created by calculating a new gradient and orientation histogram relative to the dominant orientation. The values of the histogram is stored in a feature vector.

Matches are formed as pairs of feature vectors. Each pair is graded according to the Euclidean distance between the two vectors, where a short distance signifies a close match.

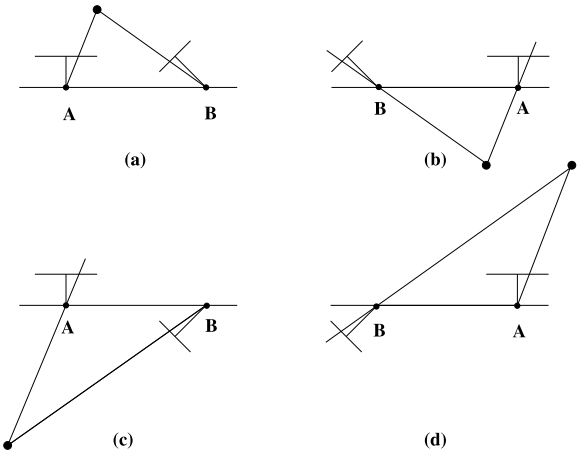


Figure 2.7: The four different solutions to the relative orientation problem given an essential matrix. From [12], used with permission.

2.5.3 Determining Epipolar Geometry

The essential matrix relation between two images is estimated from their matched points, together with the calibration matrices. If the calibration matrices are unavailable, the fundamental matrix can be computed instead.

Determining the epipolar geometry consists of finding the essential or fundamental matrix that satisfies the equation $\hat{\mathbf{x}}'^{\mathrm{T}}\mathrm{E}\hat{\mathbf{x}} = 0$ or $\mathbf{x}'^{\mathrm{T}}\mathrm{F}\mathbf{x} = 0$ for all matched points. Additionally, one algebraic property must be satisfied. The fundamental matrix must have rank 2[12, p. 280], and the essential matrix furthermore needs specific singular values[12, p. 257]. Calculating E and F is described in [32] and [12, Chapter 11] respectively.

2.5.4 Relative Orientation

To decide the relative orientation is to calculate the two camera matrices from a given epipolar geometry. Since it is not possible to determine neither the scale nor absolute pose from the geometry, the first camera matrix is fixed to the origin facing the z-axis. See Figure 2.3. This camera is said to be *canonical*. The second camera may be orientated in any direction, but since the scale is unknown the position is constrained to be exactly one unit from the origin. Given the essential matrix, there are four different camera configurations possible[12], see Figure 2.7 for an illustration. These camera matrices can be constructed using the SVD of the essential matrix.

2.5.5 Triangulation

Consider two views with one camera matrix and one 2D point for each view, and that the points have been matched to each other. We want to find the 3D position of the point seen in both views.

The *back-projection* of an image point is a line going through both the camera focal point and the image plane. The back-projection of each view's point will result in two lines. By estimating the intersection of these two lines, a 3D coordinate can be found. If the measurements in the views contain errors, the rays will most likely not intersect.

There are two approaches to solve this problem, either by minimizing the *object-space error* or *image-space error*.

Back-projection

Given camera matrix P and homogeneous image coordinate $\mathbf{x}$, the back-projected line can be described as a linear combination of the camera center and a 3D coordinate $\mathbf{X}$ such that $\mathrm{P}\mathbf{X} = \mathbf{x}$ for the image point. The function

$$\mathbf{l}(\alpha) = \mathbf{C} - \alpha \underbrace{\mathbf{P}^{+}\mathbf{x}}_{\mathbf{T}} \tag{2.12}$$

spans all homogeneous points that are part of this line[12, p. 161].

It is difficult to solve the geometric problem of forward intersection in homogeneous space and one need to rewrite this function to Euclidean space. The resulting function is

$$\tilde{\mathbf{l}}(\alpha) = \tilde{\mathbf{C}} - \alpha\tilde{\mathbf{T}}.$$

Forward Intersection

Forward intersection minimizes the object-space error by picking the point closest to both lines. The image points are regarded as free from errors.

Given the back-projected lines $\tilde{\mathbf{l}}_1$ and $\tilde{\mathbf{l}}_2$, solve the minimization problem

$$\min_{\alpha_1,\alpha_2,\tilde{\mathbf{X}}} \left|\left|\tilde{\mathbf{X}} - \tilde{\mathbf{l}}_1(\alpha_1)\right|\right|_2^2 + \left|\left|\tilde{\mathbf{X}} - \tilde{\mathbf{l}}_2(\alpha_2)\right|\right|_2^2. \tag{2.13}$$

This formulation exploits the fact that each norm describes the distance between the point $\tilde{\mathbf{X}}$ and the closest point on that line.

An alternative formulation considers the difference vector $\tilde{\mathbf{d}} = \tilde{\mathbf{l}}_1 - \tilde{\mathbf{l}}_2$. This vector must be perpendicular[24, p. 183] to both $\tilde{\mathbf{l}}_1$ and $\tilde{\mathbf{l}}_2$. This assumption is only applicable when the number of lines are exactly two. The former problem hence has an advantage since it easily can be expanded to a multitude of lines.

Looking at minimization problem (2.13), the line functions can be expanded, giving

$$\min_{\alpha_1,\alpha_2,\tilde{\mathbf{X}}} \left|\left|\alpha_1\tilde{\mathbf{T}}_1 + \tilde{\mathbf{X}} - \tilde{\mathbf{C}}_1\right|\right|_2^2 + \left|\left|\alpha_2\tilde{\mathbf{T}}_2 + \tilde{\mathbf{X}} - \tilde{\mathbf{C}}_2\right|\right|_2^2.$$

The norms can be joined together to form a combined norm expression

$$\min_{\alpha_1,\alpha_2,\tilde{\mathbf{X}}} \left|\left|\begin{matrix}\alpha_1\tilde{\mathbf{T}}_1 + \tilde{\mathbf{X}} - \tilde{\mathbf{C}}_1 \\ \alpha_2\tilde{\mathbf{T}}_2 + \tilde{\mathbf{X}} - \tilde{\mathbf{C}}_2\end{matrix}\right|\right|_2^2,$$

the content of which can be written in matrix form as

$$\underbrace{\begin{bmatrix}\tilde{\mathbf{T}}_1 & \mathbf{0} & \mathrm{I} \\ \mathbf{0} & \tilde{\mathbf{T}}_2 & \mathrm{I}\end{bmatrix}}_{\mathrm{A}} \underbrace{\begin{bmatrix}\alpha_1 \\ \alpha_2 \\ \tilde{\mathbf{X}}\end{bmatrix}}_{\mathbf{p}} - \underbrace{\begin{bmatrix}\tilde{\mathbf{C}}_1 \\ \tilde{\mathbf{C}}_2\end{bmatrix}}_{\mathbf{b}}.$$

In conclusion the problem can be rewritten in the common form

$$\min_{\mathbf{p}} ||\mathrm{A}\mathbf{p} - \mathbf{b}||_2^2$$

and be recognized as a common linear least squares problem. The solution to a linear least squares problem can be found by solving the normal equations

$$\mathrm{A}^{\mathrm{T}}\mathrm{A}\mathbf{p} = \mathrm{A}^{\mathrm{T}}\mathbf{b}.$$

Algebraic Solution

In contrast to the geometric formulation, this method requires no projected line.

Given the camera matrices P and P′ with image points $\mathbf{x}$ and $\mathbf{x}'$ respectively, find the point $\mathbf{X}$ such that

$$\mathbf{x} = \mathrm{P}\mathbf{X}$$
$$\mathbf{x}' = \mathrm{P}'\mathbf{X}$$

This only works when the cameras and points are known exactly, otherwise the system will not have any (non-trivial) solutions. One can approximate a solution anyway using Direct Linear Transform (DLT), minimizing algebraic error rather than a geometric. The downside is that the algebraic error is dependent on the coordinate system, hence translating or otherwise transforming the coordinates before triangulation affects the result.

The algebraic formulation involves the cross product of two $\mathbb{P}^2$ vectors:

$$\mathbf{x}_1 \times \mathbf{x}_2 = \begin{bmatrix} y_1t_2 - t_1y_2 \\ t_1x_2 - x_1t_2 \\ x_1y_2 - y_1x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} & -t_1 & y_1 \\ t_1 & & -x_1 \\ -y_1 & x_1 & \end{bmatrix}}_{[\mathbf{x}_1]_\times} \mathbf{x}_2.$$

The $[\mathbf{x}_1]_\times$ matrix has two linearly independent rows, only two equations are needed to solve $\mathbf{x}_1 \times \mathbf{x}_2 = \mathbf{0}$.

The original formulation of the problem can be written as

$$[\mathbf{x}]_\times \mathrm{P}\mathbf{X} = \mathbf{0}$$
$$[\mathbf{x}']_\times \mathrm{P}'\mathbf{X} = \mathbf{0}$$

where there is one unnecessary equation in each cross product. Rewrite the system to remove these, introducing the 4×4 matrix A such that

$$\mathrm{A}\mathbf{X} = \mathbf{0}.$$

If the points correspond exactly to each other, the null space of A will be one-dimensional. However, since $\mathrm{P}, \mathbf{x}, \mathrm{P}'$ and $\mathbf{x}'$ are not known exactly, the system has only the trivial solution $\mathbf{X} = \mathbf{0}$.

We can however solve the problem

$$\min_{\mathbf{X}} ||\mathrm{A}\mathbf{X}|| \text{ subject to } ||\mathbf{X}|| = 1$$

using DLT.

2.5.6 Registration

The coordinates created by triangulation is relative to the camera coordinate system. Registration puts the triangulated data and their cameras into a common coordinate system. One way of accomplishing this is *sequential relative orientation*.

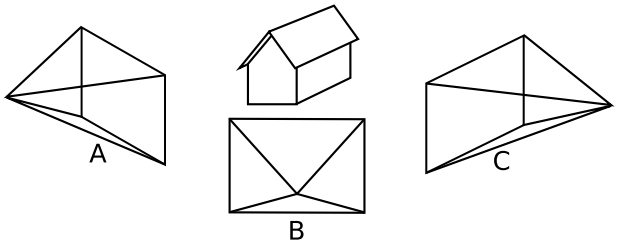
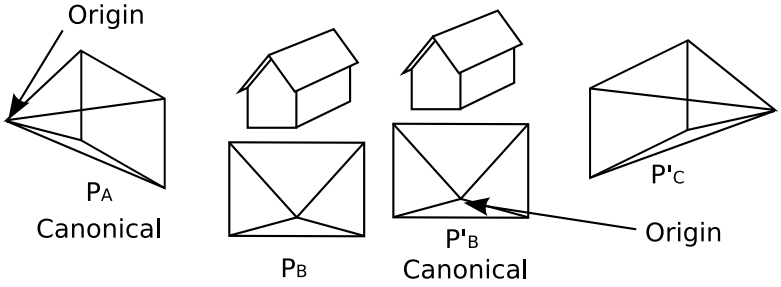


Figure 2.8: Three views of a house. The orientations AB and BC is known. Since we know that B is the same camera in both orientations, the AC orientation can be determined.



(a) A and B (b) B and C

Figure 2.9: The two pairs of relative orientation.

Sequential Relative Orientation

Consider two relative orientations, the first between views A and B, and the second between views B and C. See the views illustrated in Figure 2.8. By using the relative orientation found between B and C, we want to find the location of view C in the coordinate system of A and B.

In Figure 2.9 camera matrices P_A and P_B for relation AB, and matrices P'_B and P'_C for BC are introduced. These matrices are determined with previously described relative orientation methods. Hence P_A and P'_B are canonical, and the corresponding baselines to P_B and P'_C have a fixed length of 1.

We want to find a homography H that both satisfies $P_B = P'_B H$ and scales the baseline in BC to fit relative to AB. The scaling can be determined by comparing a common point between the views, and the unscaled homography can be found through DLT. Combining these two results gives the final homography. This homography may then be used to find the missing camera matrix $P_C = P'_C H$.

Since the camera matrices are approximated, using the resulting AC orientation to project points might give large image-space errors. More accurate orientation can be achieved by using more information. If there are plenty of points common in all views, one might prefer to use points instead of the camera matrices. This also requires that the points are all not co-linear.

Consider correctly matched points common to all three views. The points are stored as homogeneous columns in a matrix X for AB, and the corresponding matrix X' for BC. The projection on C should be $P_C X = P'_C H X = P'_C X'$ and thus $HX = X'$ or

$\mathrm{X} = \mathrm{H}^{-1}\mathrm{X}'$. The homography H can now be determined by X and X′ alone.

2.5.7 Bundle Adjustment

Bundle adjustment is the process of minimizing the projection error over all images by moving the 3D point coordinates and camera parameters.

Consider one 3D point $\tilde{\mathbf{X}}$ projected into a view described by P. The function $q(\mathrm{P}, \tilde{\mathbf{X}})$ applies the camera projection P to $\tilde{\mathbf{X}}$, the result being a 2D point. Furthermore the true position $\tilde{\mathbf{x}}$ of the 2D point is known. Taking the difference between the projected point $q(\mathrm{P}, \tilde{\mathbf{X}})$ and the true point $\tilde{\mathbf{x}}$ gives us the residual

$$r(\mathrm{P}, \tilde{\mathbf{X}}) = q(\mathrm{P}, \tilde{\mathbf{X}}) - \tilde{\mathbf{x}}.$$

Minimizing this residual may be written as

$$\min_{\mathrm{P}, \tilde{\mathbf{X}}} f(\mathrm{P}, \tilde{\mathbf{X}}) = \min_{\mathrm{P}, \tilde{\mathbf{X}}} \frac{1}{2} \left|\left| r(\mathrm{P}, \tilde{\mathbf{X}}) \right|\right|^2.$$

Expanding the concept to several views and points, with m points, n views, and V is the set of all view-point pairs, the complete minimization problem can be described as

$$\min_{\mathrm{P}, \tilde{\mathbf{X}}} \sum_{(i,j) \in V} \frac{1}{2} \left|\left| r_j(\mathrm{P}_i, \tilde{\mathbf{X}}_j) \right|\right|^2$$

where

$$r_j(\mathrm{P}_i, \tilde{\mathbf{X}}_j) = q(\mathrm{P}_i, \tilde{\mathbf{X}}_j) - \tilde{\mathbf{x}}_j.$$

The problem may be rewritten to a standardized form

$$\min_{\mathrm{P}, \tilde{\mathbf{X}}} \frac{1}{2} \left|\left| R(\mathrm{P}_1, \ldots \mathrm{P}_n, \tilde{\mathbf{X}}_1, \ldots \tilde{\mathbf{X}}_m) \right|\right|^2$$

by introducing a vector function

$$R(\ldots) = \operatorname*{concat}_{(i,j) \in V} r_j(\mathrm{P}_i, \tilde{\mathbf{X}}_j).$$

The problem may be solved by a gradient method like Gauss-Newton or Levenberg-Marquardt .

Chapter 3

Surface Parameterization

Geometric surface primitives must be parameterized and bounded before they can be turned into models of objects. Before the bounding process is described a few modeling representations are described.

The best representation of a model depends on its purpose. If a model is to be used in a physics simulation, the volume, mass, center of gravity of the object must be known. However, if a model is produced for visualization purposes, information about its mass becomes redundant. This leads to a number of different representations of models which focuses on describing different features.

3.1 Surface Modeling

A model is a digital approximation of the 3D geometry of a (physical) object. The model is often constructed from a set of geometrical primitives, like blocks, cylinders, and spheres.

Wireframe Models This is the simplest form of the surface models. The wireframe only contain information about the connectivity of its vertices. Since there is no information about faces there is basically no texturing possibilities. Although this form of model has its drawbacks it can be used when trying to visualize complex shapes that are occluded.

Polygon Meshes The polygon mesh is a slightly extended model compared to the wire frame model. In a polygon mesh structure, face information is also stored. This allows for textured models which is an import factor in this application.

3.2 Solid Modeling

An alternative to surface modeling is solid modeling. In [28] and [9] the authors explores the usage of Constructive Solid Geometry for modeling extracted buildings.

3.2.1 CSG

Constructive Solid Geometry (CSG) describes a solid model as a hierarchical structure of primitives, a set of logical operations, and rigid body operations. The primitives can

either be a set of simple geometrical objects, like blocks, spheres, and cylinders, or a set of predefined primitive instances. The logical operations are union, intersection and difference. The rigid body motions are scaling, translation and rotation.

The hierarchical relation is represented using a tree structure where the leafs are the primitives and the nodes are operations. The root of the tree represents the modeled object. This hierarchical structure of this model allows for fast implementations of for example point membership. Point membership is the question whether a given point resides in the solid. Since the model is constructed from a set of primitives and the query whether a point resides inside a primitive is a fast operation, this is done for the primitives and is then propagated up through the tree.

If a point resides in two primitives then it will also be included in the intersection of those two primitives, and if a point is included in one primitive it will also be included in the union of two primitives. On the other hand, some operations cannot be performed on CSG model directly, for example the visualization of a CSG model requires a conversion to a different representation, usually a boundary representation.

3.3 Extent Bounding

To create models from geometrical primitives they need to be bounded. Spheres and ellipsoids for example are bounded (as long as their radii are finite) while lines, planes, and cylinders are infinite in extent. Since planes extent to infinity in all directions most of this section will deal directly with planes. There exists different ways to limit the extent of a plane. One way to do this is to bound the plane using a set of points which lie in the plane. Such points are referred to as member points. The idea is to let these member points span the plane and create a convex hull, that defines the perimeter of the plane. See Figure 3.1.

It is also possible to bound a plane using the intersections of other planes. To completely bound a plane this technique will require at least three non-parallel planes. Furthermore the planes must intersects the given plane. Since the three planes are non parallel they will produce three non parallel lines of intersection in the bounded plane. These lines will in turn intersects in non ideal points in the planes coordinate system. These three points then span the bounded plane.

These two techniques can also be combined, so that a plane can be bound by the span of its member points as well as one or more lines of intersection. This is done by projecting the member points on the calculated line of intersection. The projected points are then added to set of member points. This allows for a new convex hull to be calculated from the updated set of member points.

3.4 Surface Texturing

To make the rendered face more realistic, its surface can be textured. Textures usually consists of some kind of raster data. In an image-based reconstruction, photographs of the reconstructed object is available.

Texturing is usually handled by assigning each vertex a texture coordinate. This texture coordinate is two dimensional and consists of two elements u and v. For this reason this is sometimes also refereed to as uv-mapping. The uv-mapping process for the primitives may be performed by projecting each of the vertices into a suitable image $\mathrm{P}\mathbf{X} = \mathbf{x}$, where $\mathbf{x}$ is the texture coordinate in the selected image.

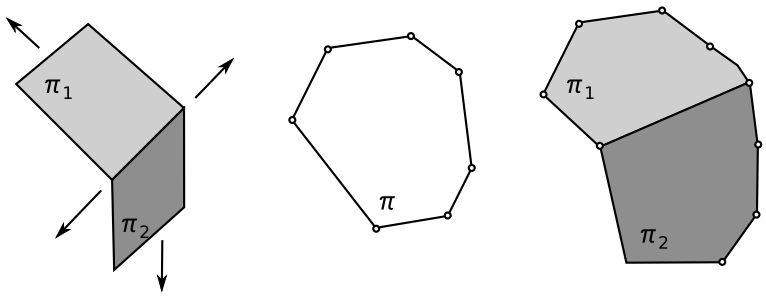


Figure 3.1: From left to right: Plane bounded by intersection, plane bounded by its convex hull, plane bounded by both convex hull and an intersection. The arrows indicate a direction in which the planes extends.

There is usually a large number of photos available as a texture for a single primitive. To choose the optimal two simple criteria have been proposed in [22] to make the selection process automatic. The angle between the camera which took the image and the polygon should be as close to orthogonal as possible. Furthermore, the area of the primitive projected into the image should be as large as possible.

Let α be the angle between the camera and the surface which is being textured. Let A be the percentage of the area the surface covers in the image. Then ω can be constructed as a measurement of how good texture candidate an image is

$$\omega = \beta \sin(\alpha) + \gamma A.$$

Here β and γ are weights. A high value of γ will make the selection function put more emphasis on the area covered, while a large value of β will make the function more biased towards a near perpendicular camera angle.

3.5 Plane Parameterization

A plane can be parameterized by the set

$$\Pi = \{\mathrm{M}, P, C, B, \boldsymbol{\pi}\}$$

where M is the 4×3 matrix defining the mapping from the planes coordinate system to $\mathbb{P}^3$. P is the set of 2D member points belonging to the plane. C is a subset of P and contains all the points in the planes convex hull. B is the set of 2D lines, bounds, in the plane which has resulted from intersections with other planes. Finally the vector $\boldsymbol{\pi}$ is the homogeneous representation of the plane.

Chapter 4

Photogrammetric Reconstruction

The dominant features of buildings can generally be broken down into building blocks of simple geometrical primitives. Such primiteves could be rectangular blocks, cylinders, and planes. Composite objects can be created by combining the primitives.

Photogrammetry is a process that outputs a point cloud without connectivity or relationships between points. Hence it is necessary to label these points according to what primitive they belong to. As an example, an operator may label all points on a wall as being on the same plane. The points may then be fitted to the geometric primitive of the wall, in this case a plane. Combining several of such primitives by relating them to each other, creates a complete composite object. The strategy is illustrated in Figure 4.1.

This chapter first looks at the techniques required to calculate the points. The latter part of this chapter will focus on how to create the geometrical primitives and then how combine these into a surface model. The emphasis is put on combining planes and cylinders.

4.1 Camera Calibration

Before point cloud reconstruction, the calibration matrix and lens distortion for each camera is determined. This is usually done by photographing a special calibration object. A separate, external application is then used to estimate the parameters. Once determined, the calibration parameters can be reused on all photos taken with the camera.

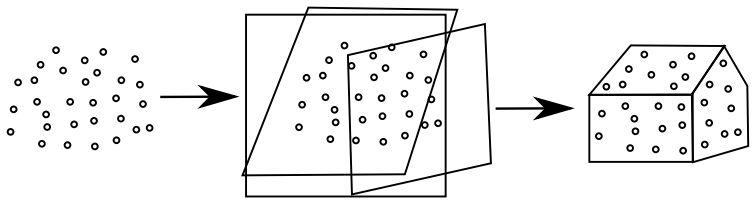


Figure 4.1: From left to right: 3D Points from a reconstruction. Approximated planes from the points, Solid model created from the planes.

4.2 Image Management

The photographs in this thesis are digital still images taken in the range 5 – 300 meters. The sheer number of photos required for a photogrammetric reconstruction is a problem, even for a moderately sized scene. It is therefore required to be able to organize and present this data in the application. A divide and conquerer approach can be applied to this problem. The operator groups the images into smaller groups containing a single building.

The matching of corresponding image pairs, i.e. overlapping areas in one or more images is a fundamental part of a photogrammetric reconstruction. The solution presented in this thesis is that the operator manually pairs together matching photos in each group. The matching process itself in each image pair is automatic, reducing the workload of the user.

As stated earlier, the order of which images are added in the registration process is important. The operator manually registers each photo, where the image groups are used as an aid to the operator.

4.3 User-assisted Labeling

There are several ways to let a user label plane surfaces to be included in the 3D model. They all assume that the user is labeling on the photos, not the reconstructed point cloud.

Directly marking the contour with a 2D polygon This has the downside of being difficult due to occlusion and indistinct corners.

Labeling the feature points in the picture This makes it possible to use the triangulated points from these labeled points to run e.g. ODR or other primitive fitting. The downside is that the labels are discrete points and rerunning feature point detection will not be able to reuse the labels in an effective matter. Letting the user label the 3D cloud points is essentially the same thing as this option.

Using a "paint" tool to mark each wall with a distinct color in every photo By using a separate mask layer, and letting a operator color-code the walls, a program can determine what label a feature belong to by referencing the mask. The

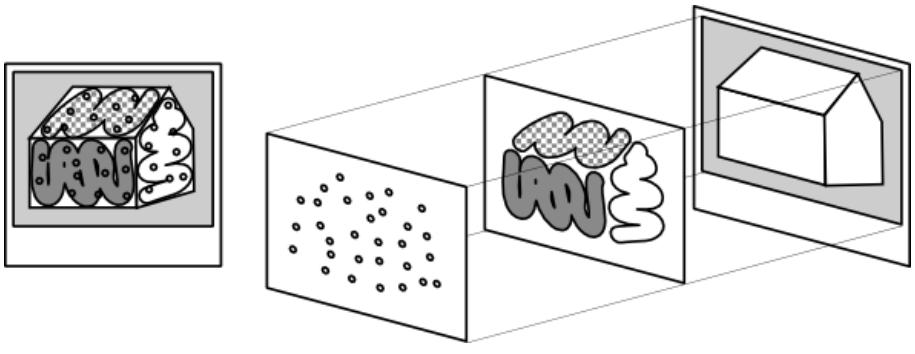


Figure 4.2: Point labeling

user may skip objects that occlude the wall and there is no need to exactly mark the contour of the wall. This paint information could also be useful in texture extraction. See Figure 4.2

4.4 Point Reconstruction

In this section the specific methods used to reconstruct 3D point clouds in this thesis are presented. All measurements and calculations are made by algorithms. Since there will be measurement errors, robust methods are used to estimate a solution. The robust method is aborted if it goes beyond 1000 iterations. When a solution has been estimated, it is shown to the operator, which decides whether or not to register the result.

4.4.1 Image Rectification

Since camera lenses introduce nonlinear errors in the photos, all images were rectified with a separate external application. To simplify the prototype, it only manages rectified images.

4.4.2 Feature Point Detection and Matching

Feature point detection and matching are done with SIFT[18] through a software library called SIFT++. SIFT have many different parameters that can be adjusted to the specific characteristics of the input images. The values used in the implementation by Lowe are sufficient, and often give a large amount of points.

Matching is done pairwise, iterating through all features in the first image, comparing them to the features in the second image. The Euclidean distance between the feature vectors are used to order the features according to similarity. In order to avoid creating pairs where there are multiple similar features, the two pairs with shortest distance are determined. If the difference between these distances is too small, and the match is discarded. Otherwise the pair is significant enough, and a match is created.

4.4.3 Epipolar Geometry

The epipolar geometry is determined with the five-point method by Stewenius at al[32]. This method requires only 5 points as a minimum, and is superior to the other five-point methods known. Unlike similar 5-point methods, it correctly determines situations solely consisting of co-planar points. Up to ten solutions may be returned, due to ambiguities.

RANSAC is used in order to make the fitting robust to outliers. The essential matrix with most inliers is selected, using the *Sampson distance*[12] as distance measurement. This is a geometric image-space error. The distance also determines which solution to use from those returned by the 5-point method.

4.4.4 Relative Orientation and Triangulation

The relative orientation for the essential matrix gives four different solutions for the relative camera matrix (the canonical camera is fixed). See Figure 2.7. In order to determine which camera matrix is most likely to be correct, triangulation is done for each solution. Each solution is a pair of camera matrices: the canonical and the relative

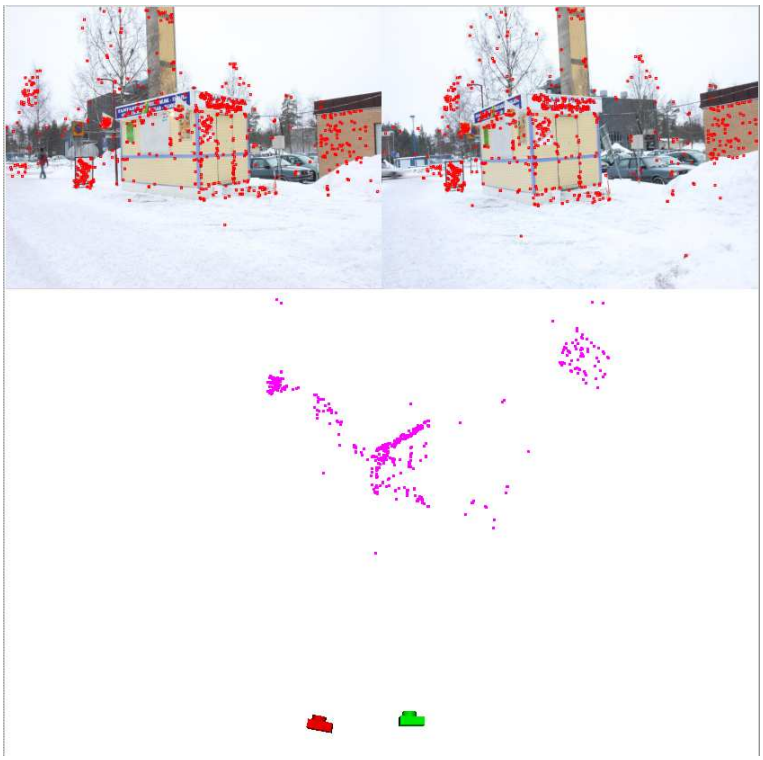


Figure 4.3: Above: Two photos with automatically matched points (red). Below: Triangulated points (magenta) together with representations of the canonical (red) and the relative (green) cameras. Outliers have been removed automatically.

matrix. A successful triangulation can be seen in Figure 4.3. The pair with the highest amount of points in front of both cameras is chosen.

There are several different reasons why points triangulate behind the camera pair. There might be outliers from the match process, since triangulated mismatches may have very deviant positions. Another reason might be that some points are close to the epipole and thus easily perturbed, causing a whirlwind-like effect. These two reasons have nothing to do with the relative orientation. A third reason, and the cases we want to detect and avoid, is that either one of the cameras is orientated in the wrong, opposite, orientation. The points are thus projected correctly to the image plane, but their triangulated position in relation to the camera would be impossible.

4.4.5 Registration

Registration is initiated by picking a starting image pair. The resulting point cloud from the triangulation can be directly saved into the register. In Figure 4.4, the starting pair is AB. The next image pair in the registration tree is BC, then CD and so on.

An image pair, lets say BC, is registered by looking at its triangulated points. If there are at least 3 points common with the register, which in this case consists of points from AB, the registration can be performed. The 3-point limitation is due to the rigid-body

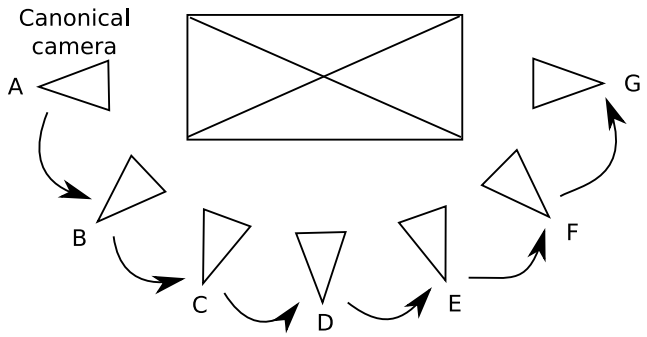
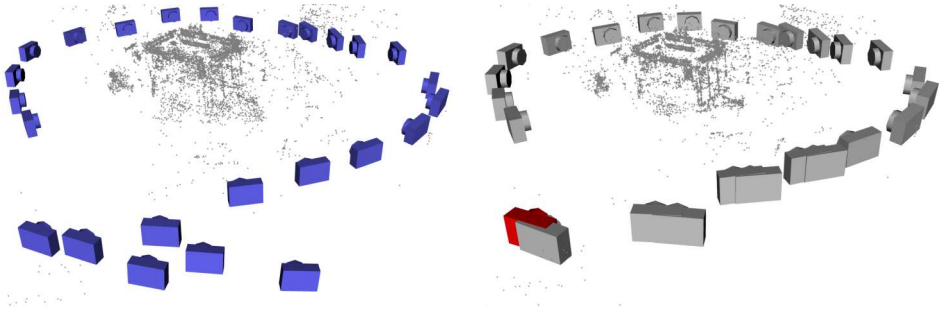


Figure 4.4: Topological view of a sequence of cameras.



(a) Before (b) After

Figure 4.5: The Kiosk shot using the stereo rig. 22 left cameras and 4 right cameras has been registered. After bundle adjustment, the four right-hand cameras move significantly, correctly taking their places in the stereo pairs.

transformation method of sequential orientation. If the rigid-body transformation can be calculated correctly, the extra points in BC can be added, and the common points are merged into *tracks*. A registered point that is visible in both A, B, and C is called to be tracked through these images. If one of the two views in the pair have not already been registered, the camera position is also registered. In the case of registering BC, the camera position of C would be created in the register. If AC would later be registered, only the point information would be updated.

The rigid-body transformation is determined by the method described by Söderkvist and Wedin[31].

4.4.6 Fine-Tuning

The point reconstruction is fine-tuned by minimizing the image-space error with bundle adjustment. The track information from the registration is used during the adjustment, i.e. features that are matched over several images are reconstructed as one 3D point. Both the external parameters of the cameras, and the position of all reconstructed points are adjusted in the minimization. The minimization problem is solved with a Levenberg-Marquardt algorithm.

4.5 Plane Surface Reconstruction

Most buildings have large walls built of bricks or wood. These walls can be measured by fitting a mathematical plane equation to it and model it with a polygon surface with all vertices coplanar. The intrinsic depth structure of the wall – like individual bricks – is not modelled.

To fit a plane to a photographed surface, one has to somehow make it possible to mathematically describe the problem. There are basically two major approaches, either use 2D data (photo measurements) or 3D data (point cloud from reconstruction).

Using 2D data directly is difficult, one could mark the outline of a wall in two photos and triangulate the 3D outline. This is hard to accomplish because the user will not be able to enter the outline consistently over all photos and with enough precision to make for a reliable input. Even if these problems were solved there is still the problem of relative orientation needed for triangulation, which is more reliably solved with traditional reconstruction methods.

On the other hand, using 3D data to model planes is easy since the hard work is already done – the triangulation. A plane equation can be fitted to the reconstructed 3D points using ODR.

Limiting the primitive is necessary since walls have a limited span, unlike planes which are infinite. Possibly the plane could be limited by its outermost points, i.e. the convex hull. Regarding buildings and other block structures, limiting the plane in intersections with other planes is a possible solution.

4.6 Reconstruction of Composite Objects

From the perspective of creating a realistic 3D model of a reconstructed object the pure mathematical representations of the geometrical primitive does not suffice. The geometrical primitives must be bounded and combined with other primitives into composite models. The composite models may later be textured for a more realistic look.

4.6.1 Extent Bounding

Since planes are approximated from measurements which are not exact the approach described in Section 3.3 does not always suffice. For example, if a rectangular box is to be created from a set of planes then only points lying on the surface of the rectangle should be included in the model. If there is a set of outlier among the plane's member points, i.e. lying outside of the rectangular box, they would been included in the model and produced jagged edges on the rectangle instead of straight edges as desired. Figure 4.6 shows an example of such artifacts.

The solution to this problem is to only keep member points which are on the correct half plane. The half planes are created when a line of intersection partitions a plane into two parts. To decide which half plane is the correct one, a membership majority function can be constructed to decide which half is to be kept. Simply put, the half plane with the largest number of member points is deemed to be the correct one. The plane's set of member points can then be updated and a new convex hull can be calculated.

Bounding from Multiple Primitives

When a plane is bounded from several other primitives, special care has to be taken when the convex hull is updated. This is due to the fact that the line of intersection on which the member points are projected upon, might have been bounded in a previous intersection.

Take for example the gable of a saddleback house. This facet of a plane is bounded by four other planes, the two adjacent walls and the two sides of the roof, Figure 4.7. To solve this problem one can keep track of the previous bounds and make sure that only points which are on the correct side of all previous bounds are included in the convex hull.

4.6.2 Texturing

Each model is textured by uv-mapping the points from the convex hull. Each point is projected onto an image and assigned uv coordinates. The image that is selected as texture data is the photograph that contains the most member points.

A surface that spans many photos, i.e. there is no photo in which the entire surface is visible causes problems. One of the problems is that many rendering systems require the whole surface to be mapped to the same raster. Although it is possible to merge images, the result is going to contain ugly seams. This could potentially be solved by using an automatic stitching algorithm. But care has to be taken so that the mapping of 3D points into the merged photo are correct.

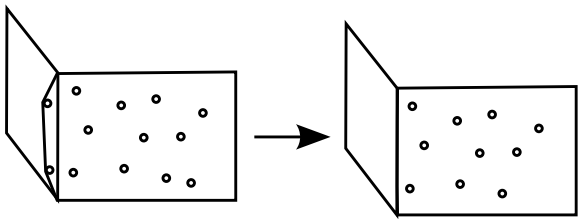


Figure 4.6: Bounded plane showing a jagged edge due to erroneous point data.

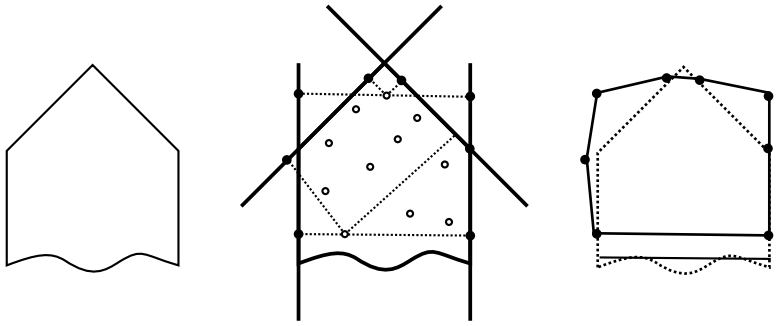


Figure 4.7: Gable of a saddleback house. From left to right: Desired convex hull. Projection trace of two member points (dashed) and lines of intersections (bold). Finally the erroneous convex hull due to unchecked bounding (bold) and the desired hull (dashed).

Chapter 5

Implementation

The prototype application was written for GNU/Linux systems, and features a graphical user interface together with persistant storage. A complete list of all used software is available in Appendix C.

5.1 Development Tools

The choice of programming language for the prototype application was Python. Two major factors were significant in the choice of language: Rapid development, and the ready accessibility to a vast number of libraries. Python is a high-level multi-paradigm scripting language. Since it is a scripting language and has numerous modules to interface third-party libraries, it was deemed suitable.

Development of a GUI (graphical user interface) applications can be time consuming due to the amount of code that must be written, both in regard of GUI widget layout and event handling. Instead of programming the layout using the API, the layout was done graphically using the *Glade Interface Designer*.

5.2 Numerical Calculations

The numerical calculations and algorithms were implemented in m-code, the MATLAB language. Since a lot of algorithms have already been implemented in this language, it was advantageous to be able to use existing implementations.

Although the numerical code originally was written to be interpreted by MATLAB, the Octave software was used to run the code. Octave, unlike the proprietary MATLAB, is free software. This makes it possible to read the source code and create derivative work of the software.

To be able to call Octave and have the calculations returned to the calling script, a Python module was written specifically for the thesis. The module, called Pytave, dynamically links against the Octave libraries. This makes an Octave interpreter reachable through the module Python interface. For convenience, arguments and return values are transparently converted between the native data structures of both languages. The module has been published as free software and is available on the Internet.

Similar modules were also written to interface the SIFT++ and `sba` libraries, which

previously had no Python interface. Bundle adjustment is done with the `sba` implementation from Lourakis and Argyros[17].

5.3 Relational Database

The prototype uses a relational database to store photos, user input, intermediate data, and the final result. A self-contained, server-less relational database manager called `SQLite` 3 was used through a Python module `pysqlite`.

The overall design of the database scheme is diagrammed in Figure 5.1.

5.4 Bundle Adjustment Package

Version 0.2 of the `Bundler` application from [29] has been evaluated, a program derived from the system used in [30]. It was possible to run the demo application with custom input. However, difficulties emerged when trying to adapt the package to the purposes of this thesis. Problems included memory errors both in input parsing and in the data processing. It was deemed too time-consuming to fix these shortcomings.

5.5 Workflow Implementation

Before the photographs can be used in the application the radial distortion must been removed. The calibration data must be specified when loading the images into the application.

Image groups After the images has been loaded the first task the operator performs is to create one or more image groups. Images that are related in some way are placed in the same image group. This could for example be all the images of a single building. Figure 5.2 shows an example of this view. The main view display feature points superimposed on the photograph as red squares.

Image pairing When the photographs has been placed into suitable image groups, the operator pair the images. SIFT feature point matching is performed on each image pair. Figure 5.3 shows an image pair being matched.

Triangulation For each image pair the operator calculates the relative orientation and triangulates the matched points. The operator can preview the result and decide whether it is adequate to be added to the sequential registration, or if needs to be recalculated.

The operator has the possibility to change the minimum degree of separation of the SIFT matching. This can be thought of as how good each match must be to be considered a correct match. Figure 5.4 shows the relative orientation and triangulation results of an image pair.

Bundle adjustment Bundle adjustment can be done as long as the sequential registration has been initialized. The operator would usually run this operation when the sequential registration is done. The view seen in Figure 5.5 offers views of both before and after bundle adjustment has been performed.

Plane fitting The plane fitting consists of two steps. First the operator marks the surfaces of interest in the images. Each surface is marked with the same color in each image. When all the surfaces has been marked the operator can create planes from the labeled points.

The operator has the option of removing points that have been mislabeled or wrongly triangulated.

To combine planes, the operator selects two planes and issues the intersect command. The operator also has the option to export the model to an external 3D modeling application. Figure 5.6 shows a labeled photograph, color coded points, and half completed model.

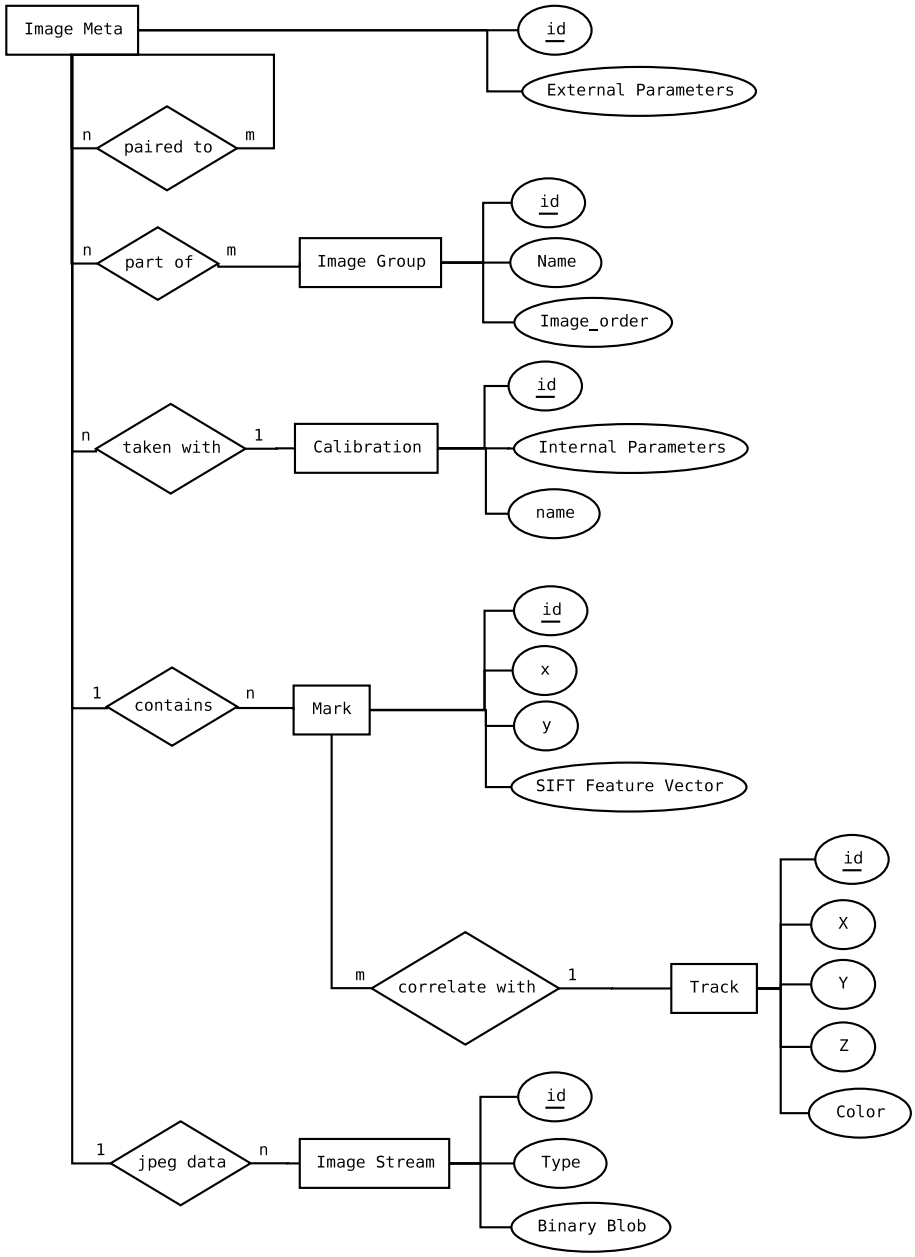


Figure 5.1: ER diagram of the relational database.

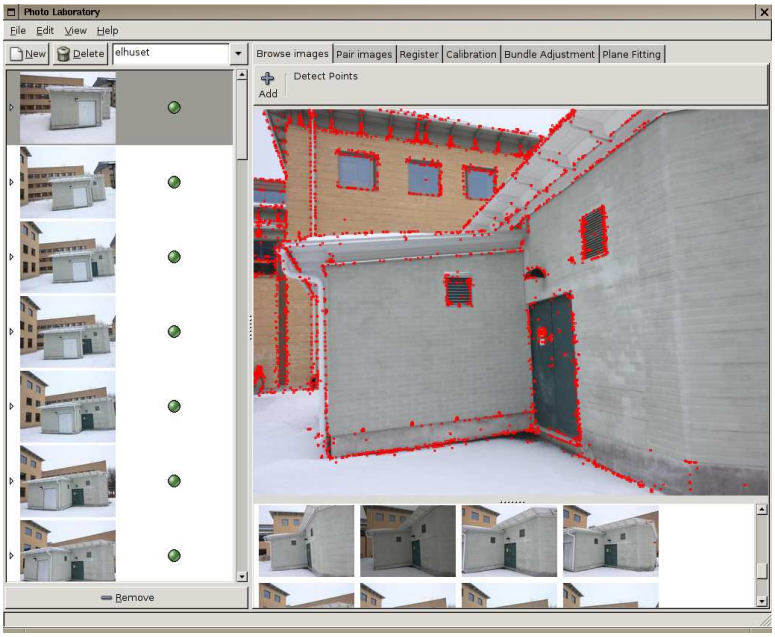


Figure 5.2: View of the image grouping screen. Panel on the left shows images belonging to the selected image group. Top right panel offers a larger view of the select photograph. Bottom right panel shows all available images.

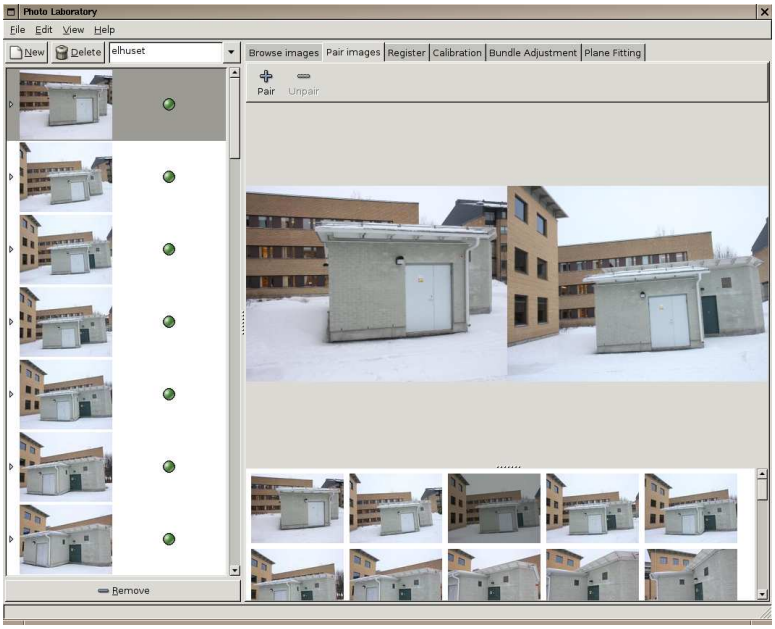


Figure 5.3: View of the image pairing screen. The top right panel displays the images to be paired.

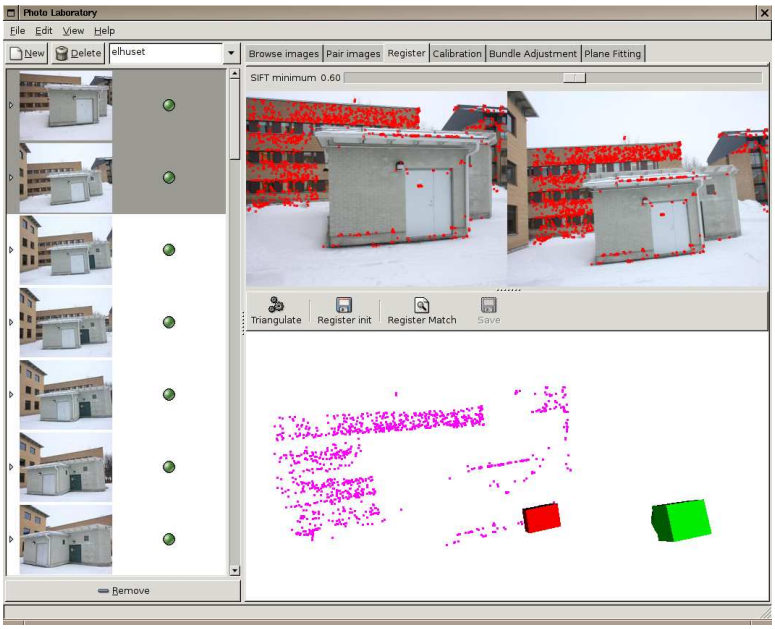


Figure 5.4: View of the triangulation screen. Top right panel displays the *matched* featured points. Lower right panel displays the *triangulated* 3D points and camera positions.

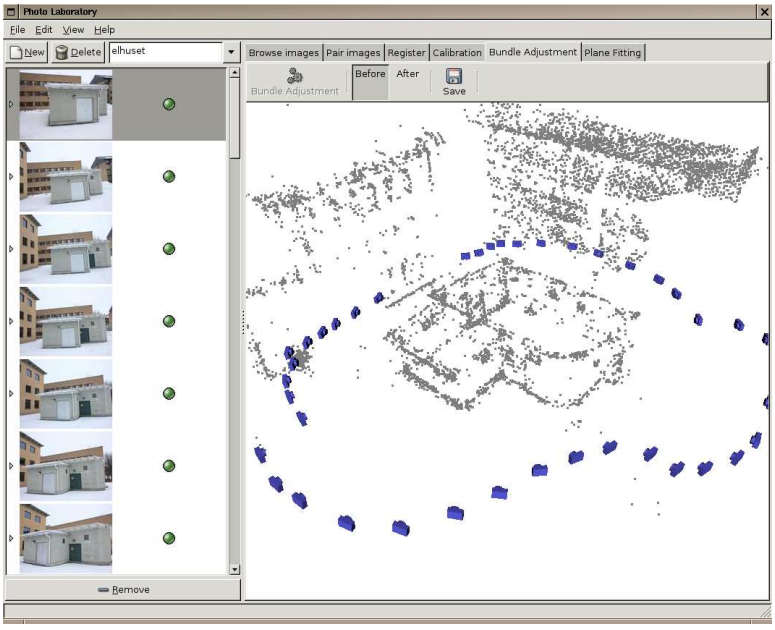


Figure 5.5: View of the bundle adjustment screen. The Right panel shows a view of all camera positions and points added to the sequential registration. The view offers a comparison of results before and after bundle adjustment has been performed.

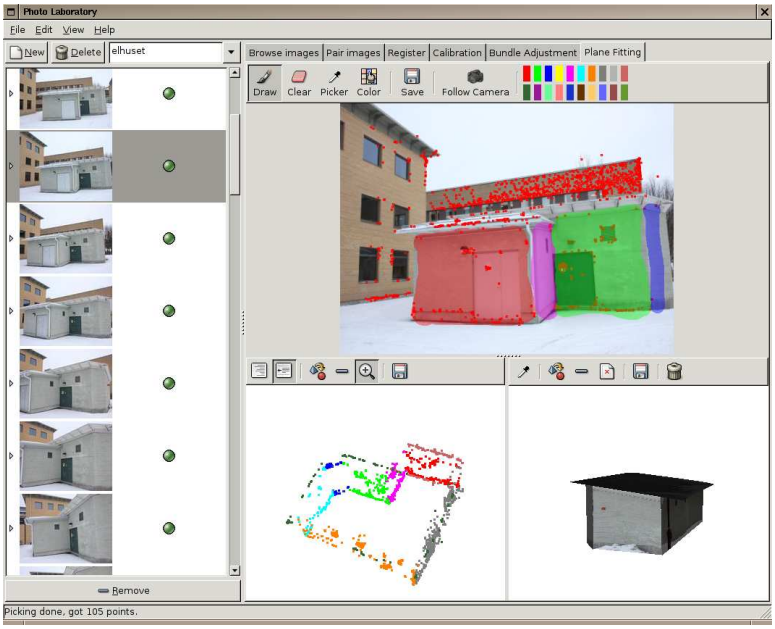


Figure 5.6: View of the plane fitting screen. The top right panel offers the operator the possibility to label surfaces. The lower left panel displays the color coded 3D points marked in the image. Lower right panel shows the textured surfaces.

Chapter 6

Image Acquisition

All images were acquired specifically for the work in this thesis, using digital SLR cameras. The sites were shot using a stereo rig of two cameras, mounted on a tripod. The rig consisted of two Canon Eos 40D digital cameras situated 52 cm apart. A photograph of the used rig can be seen in Figure 6.3.

6.1 Exposure Time Experiment

A series of photographs were taken on each site while changing the exposure time. For each of the two series, 7 different exposure times were used: 1/100, 1/50, 1/25, 1/10, 1/5, 1/2, and 1 second. A remote shutter trigger was employed to minimize movement during exposure.

Two different sites on the university campus were used; the facility Universum (*Universum*) and the MIT building (*MIT*). Photographs of MIT and Universum can be seen in Figure 6.1 and 6.2.

The experiment was performed January 22, 2009 around noon time.

6.2 Site Photography

The photo sets presented in this thesis were shot using a shutter speed of 1/8 of a second. The reconstructions of four sites were selected to be presented in the thesis. These sites are:

- *Origo*, a facility building / night club owned by one of the student unions at Umeå University (Figure 6.4).
- A small *Utility House* located between the MIT building and *Teknikhuset* (Figure 6.5).
- Fantastisk Grill, a small fast food kiosk (*Kiosk*) situated on the Umeå university campus area (Figure 6.5).
- *Minerva*, a secondary education school close to the campus (Figure 6.7).

The front facade of the Minerva building was shot along the opposite of the neighboring road. The Utility House and Kiosk was shot from all sides. At the Origo site the front facade and entrance was shot.

Figure 6.1: Photograph of MIT.

Figure 6.2: Photograph of Universum.

Figure 6.3: Stereo rig used during the photo sessions.

Figure 6.4: Photograph of Origo.

Figure 6.5: Photograph of Utility House.

Figure 6.6: Photograph of the Kiosk.

Figure 6.7: Photograph of Minerva.

Chapter 7

Results

Four different photo sets have been used with the prototype application. The reconstructed models are presented in the end of this chapter.

7.1 Exposure Time Experiment Results

SIFT matching is an important element in the prototype application, getting a lot of correct matches is important to be able to successfully run e.g. sequential relative orientation. Thus to examine what impact shutter speed has on SIFT matching, an experimental photo session on two sites was performed.

For each of the corresponding photos in the stereo rig, feature point matching was performed. Only photos with the same exposure times were matched. The result is shown in Figure 7.1.

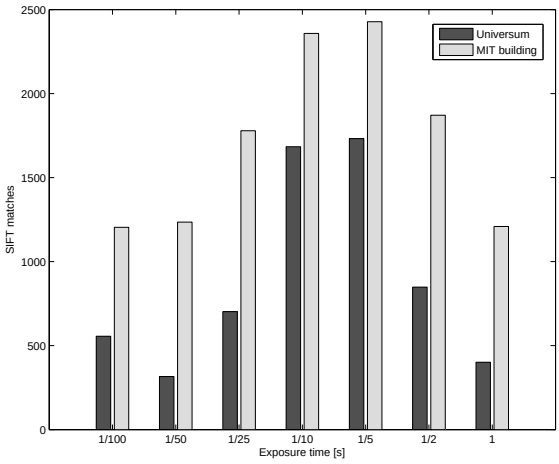


Figure 7.1: Diagram of the number of SIFT matches as a function of the exposure time.

(a) Exposure time 1/100 s.

(b) Exposure time 1/25 s.

(c) Exposure time 1/10 s.

(d) Exposure time 1 s.

Figure 7.2: MIT shot with four different exposure times.

7.2 Reconstruction Results

The number of photographs, reconstructed points, surfaces fitted, and points used in the fitting are shown in Table 7.1 together with the time consumed for each site. The time is measured in wall time (hh:mm). The time measurement only includes triangulation, registering, labeling, and surface fitting.

The Origo photo set did not produce enough feature point matches to be able to do relative orientation. The building and the ground had few matches, while the background – consisting mostly of tree tops – had the majority of the matches. A photo from the set can be seen in Figure 7.3. No further results are presented on this site.

The Utility House (Figure 7.4) was fully reconstructed, with all walls and roofs modeled. A full lapse around the building was completed. The background elements of this photo set was static and close to the utility house. Views of the model, both textured and in wireframe, can be seen in Figure 7.5 and 7.6.

The Kiosk (Figure 7.7) was also reconstructed with a full lapse. The surrounding environment of this site was more complex, with backgrounds depicting far-away objects – some of which was moving. See Figure 7.8 and 7.9.

The front facade (Figure 7.10) of Minerva was reconstructed, creating a complex model. This building was shot with a distance ranging 50 – 100 m, unlike the other buildings that were shot at a distance of 5 – 20 m. The textured model can be seen in Figure 7.11 while 7.12 shows the same reconstruction as a wireframe.

Table 7.1: Comparison of model complexity.

Site name	No. photos	No. points	No. surfaces	No. used points	Time
Utility House	41	11946	9	1328	1:10
Kiosk	24	6909	8	3113	1:45
Minerva	16	4636	12	940	0:45

Figure 7.3: Photograph of Origo.

Figure 7.4: A photograph of the Utility House.

(a) View front.

(b) View from above.

Figure 7.5: Two views of the Utility House.

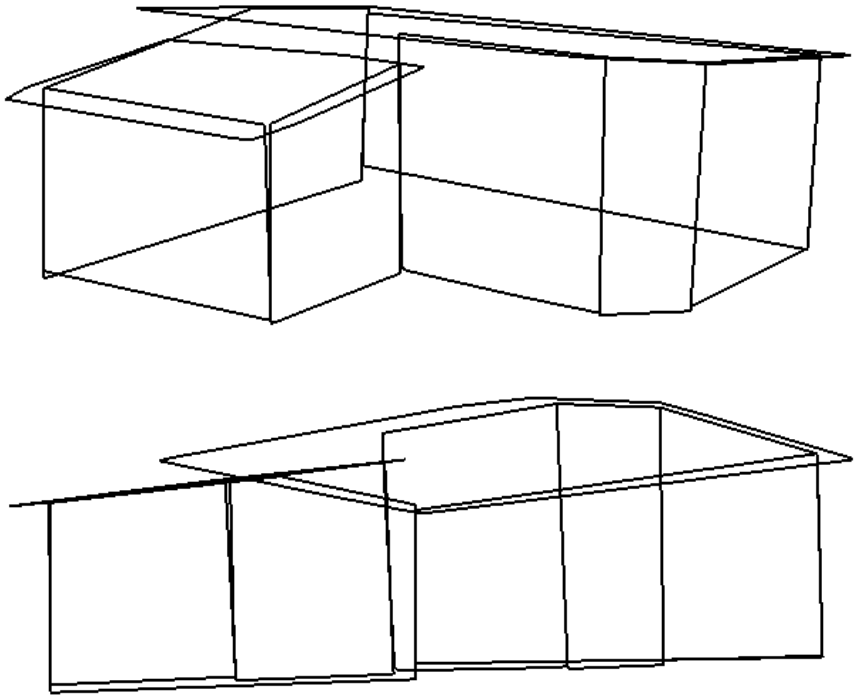


Figure 7.6: Two views of a reconstruction of Utility House.

Figure 7.7: Photograph of the kiosk.

Figure 7.8: Reconstruction of the Kiosk.

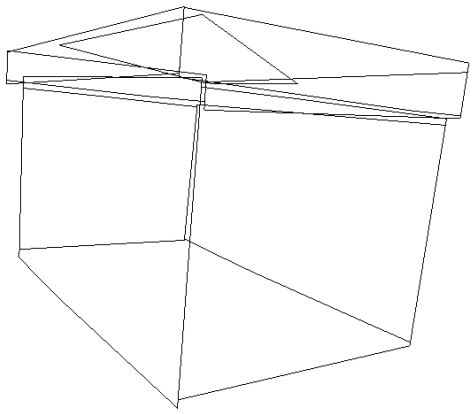


Figure 7.9: Wire frame view of the Kiosk.

Figure 7.10: Photograph of Minerva.

Figure 7.11: Two views of a textured model of Minerva.

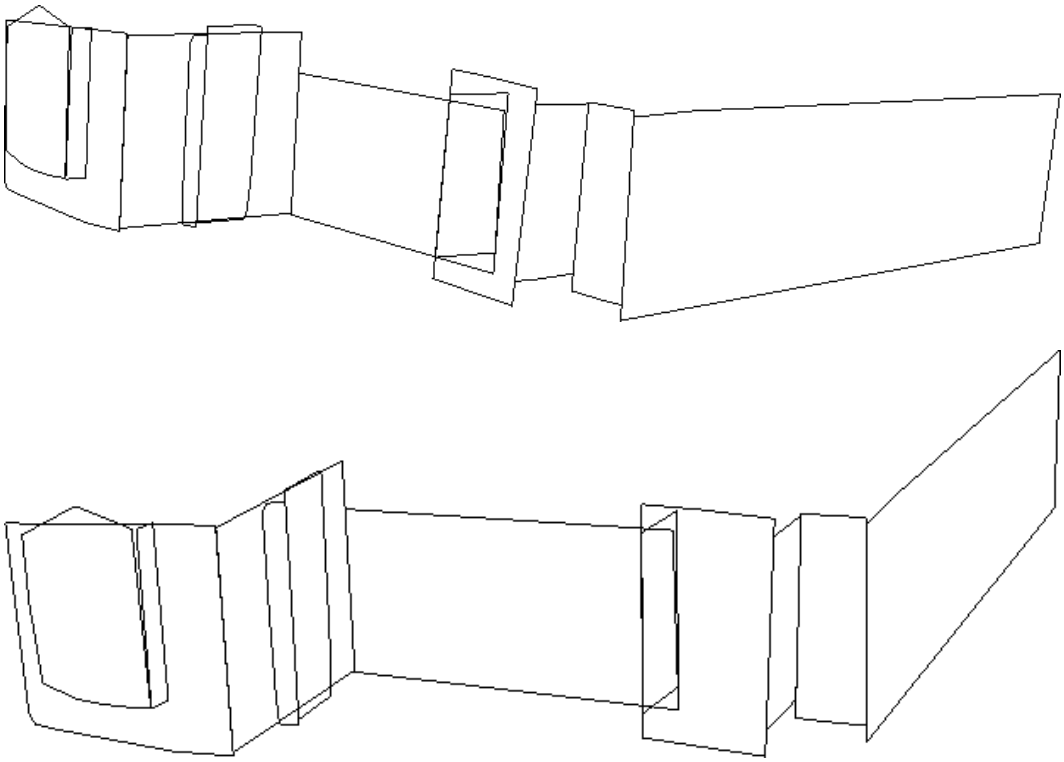


Figure 7.12: Two views of a wireframe model of Minerva.

Chapter 8

Conclusions

The aim of this thesis was broad, both in implementation and theory. Evaluation of methods described in various papers is time consuming and can be theoretically difficult. This limited the number of methods that were thoroughly evaluated. With the broad aim in mind only existing solutions and implementations were employed in the prototype application.

8.1 Feature Extraction Analysis

In the summer season, northern Sweden has sunlight during a majority of the day. It is possible to photograph an out-door object by simply walking with a camera and shooting by hand. In the winter season however, not even noon daylight is always enough to ensure good image quality with a hand-held camera. The shutter speed must be longer, and holding the camera by hand is not stable enough. Hence it is necessary to use a tripod and use an extended exposure time.

From the experimental results shown in Figure 7.1, it was clearly visible that proper lighting conditions improved the number of matches. For reference, four of the photographs of the MIT site shot with different exposure times can be seen in Figure 7.2.

The backgrounds in the photo sets were a significant factor in whether or not the relative orientation was successful, especially if the reconstruction object was depicted only on a smaller fraction of the image area. If the background only consisted of trees and moving objects, the reconstruction process proved more difficult. This was the case for the Kiosk photo set. On the other hand, The Utility House had other buildings in the background and was easier to reconstruct.

Another problem that arises during the winter season is that snow covers the ground around the objects as well as roofs. This leads to fewer feature points around objects which in turn could cause the reconstruction to fail. The Origo photo set (Figure 7.3) suffered from this problem, having few feature points in its surroundings. The backdrop of the photos was both far away and consisted of swaying tree-tops.

8.2 Time Consumption

No clear conclusions can be made from the time consumption presented in Table 7.1. However the different scenes posed different problems. For example the point labeling

step in Minerva set consumed more time than in the other sets. The Minerva building was more complex in shape compared to the Kiosk and the Utility House. Minerva was shot from a longer distance this also contributed in making the labeling process more cumbersome. While geometrical simple, the registration process of the Kiosk was more problematic than the other sites, due to the backdrop of the photo set.

In conclusion, many factors play a role in time consumption of a reconstruction, and it should be noted that the individual operator has a large impact on time. The results presented in Table 7.1 were acquired from different operators.

8.3 Implementation

The application consists of both compiled C++ libraries and interpreted Python and m-code. This way, each problem domain could be treated with its closest matching language. Some algorithms were already implemented in C or C++, and we were able to reuse those implementation. Similar things could be said about using m-code to solve mathematical problems and Python to create the graphical user interface.

Using a relation database scheme instead of developing custom data structures proved advantageous, since saving, reading, and combining data could be delegated to the database manager. An added advantage of using Sqlite is that the database is stored as a single file on the file system, making it easy to move projects between computers.

8.4 Restrictions

One restriction on the prototype application is the lack of support for uncalibrated cameras. This was motivated by the fact that using a calibrated camera makes the approximation of the epipolar geometry better conditioned. Furthermore, the calibration process does not consume much time, and is done only once.

One feature lacking in the prototype is the possibility to model the ground around buildings. This was deemed out of scope for this thesis due to time limitations. This lack of ground modeling capability makes measurements of larger environments difficult.

The choice of only using automatic feature point detection and matching turned out to be less flexible than desired. Allowing for an operator to manually edit the correspondence matching would have been a useful feature in the case of bad input.

Measurements and modeling of planes had a higher priority than cylinders through the work. This caused the cylinder measurement and modeling to be cut out from the application. Though some parts of the theoretical work was kept and can be found in Appendix B.

8.5 Limitations

Lack of texture on walls is a problem that can causes the correspondence matching to fail. This problem can also arise when photographs are taken under badly illuminated conditions.

It is not possible to recalibrate the calibration matrices against a reconstruction. In other words, fixing the reconstruction and minimizing the reprojection error by changing the camera matrices. The implementation of bundle adjustment (`sba`) does not support changing the parameters of the K matrix globally.

Furthermore there is no error estimate presented to the operator in certain steps, instead the operator has to make a judgment if the result of e.g. a triangulation is plausible. No *undo* feature has been implemented in the prototype.

The texturing scheme used in the prototype may introduce visible artifacts in the models. These artifacts results from the projective changes in photographs. To avoid the artifacts, photographs picturing the surfaces orthogonally, in relation to the surface should be used as textures.

8.6 Evaluation of Aims

Image pairing was implemented as a completely operator-lead process. The automated point matching was implemented with SIFT and proved largely successful, given photos of high quality. Some difficulties arose in certain circumstances. Problematic cases included organic, moving backdrops and low-contrast surroundings.

Designing a flexible and usable way for the operator to perform these task raised some interesting questions. For example, how does the operator select the input for a plane reconstruction. The choice in this particular case fell on labeling portions of the image in which the plane is visible. This approach is flexible since 3D points can change if a new triangulation is made. The feature points in an image can also change if the correspondence matching is updated. The photographs however do not change, an area portraying a plane will always do so.

Since the number of photos required to make a reconstruction is large, at least a couple photos of each building's wall, gaining a good overview of the input data is difficult. The proposed solution to this is to allow the operator to group the images together into smaller groups. The use of photo groups improved the overview of the large input and allowed the operator to focus on the task at hand.

One aspect of the workflow design was to use automatic methods for measurements and calculations. The estimated solution is then visualized to the operator. Using this visualization, the operator can determine whether or not the solution is adequate.

8.7 Future Work

Photogrammetry and computer vision is quite an active area of research, and there exists interesting features on which further study can be made. Here a few approaches to improve the model and the reconstruction process are presented.

8.7.1 Parameterized Primitive Instancing

Primitive instancing is a way to avoid "reinventing the wheel" while modeling. Historically this has been models like bolts and screws in CAD software which can be instantiated by simply entering a diameter, etc. Compared to going through the whole process of modeling the object instancing is much faster. Given a database containing the simplest forms of buildings like hip-, flat-, pent-, and saddleback-roof buildings would, combined with traditional modeling be a good asset. The parameters to these models could be the lengths of the different edges, or a more flexible model, where one specify ratios of lengths and angles between the different pieces in the model.

8.7.2 Fitting Models to Geometrical Primitives

To be able to take full advantage of primitive instancing there must exist a convenient way to fit these models to point clouds measured in a reconstruction. The fitting process could be seen either as a minimization problem, where the application tries to automatically fit the model to a given point cloud, or an operator guided approach. The operator could for example fit parts of the model to a subset of the point cloud and for each step remove degrees of freedom from the model until it is fixed.

8.7.3 Creating Parametrized Models

The parameterized modeling would be a lot more flexible if the operator could create generic models which in turn could be saved in the model database and then be used for instancing. The models could also have a set of basic parameters, e.g. what angles should be fixed, free, or perpendicular. Further parameters could be ratio of lengths and areas.

8.7.4 Extended Modeling Tools

Doors and windows in buildings are often "sunken" into facades. To model this correctly at least 4 or 5 planes including the window or door has to be measured. A more efficient solution to this problem would be to implement an intrude operation. The intrusion operation automatically connects the sunken plane, e.g. window or door with the facade polygon without the need to measure the adjacent planes.

Another useful feature is the opposite operation of an intrusion, i.e. the plane is not sunken into a facade but is extruding from it. These features could thus improve workflow of the application.

8.7.5 Facade Detail Enhancement

One way of extending the level of detail of the facades would be to implement the features proposed by Pascal Müller et. al. in [23]. Here methods to automatically derive high resolution 3D models from a single rectified photograph of a facade is presented. The approach described utilises image analysis to subdivide the facade. The facade is then reconstructed using procedural modeling.

[35] explores the possibilities of using computer vision techniques to identify common architectural details on buildings, particularly on facades. Indentions on the wall surfaces as like windows and portals may be extracted automatically.

8.7.6 Guided Matching

Currently, primitives are created from triangulated SIFT matches. The SIFT matches are very good while creating the relative orientation, but are less than perfect for primitive reconstruction. The point clouds are often sparse, and some walls can have very few SIFT matches. Still the camera poses has been determined, and this could be used to get denser clouds. Since the camera positions are known, the epipolar constraints can be used to make quasi-dense matching[15][14].

Another possibility is to use the plane geometries and try to improve them. A plane visible in two images could be translated into a homography, which could be used to

do a guided template matching[2][8]. This would improve the accuracy of the plane geometry.

Using a stereo rig creates several advantages. Since the relative orientation is fixed in the rig, it is possible to calculate this once and use it for every photo pair. This also improves feature point matching between these photo pairs since the epipolar constraint can be used while matching the points.

8.7.7 Reusing Label Information

In the prototype application the operator label surfaces in photos by coloring them. This information could be put to other uses as well. Consider the image matching problem, if for example a red surface is painted in two photos then matching could automatically be performed between these images. The color information could potentially also be employed as an initial guess for the sequential registration process.

Chapter 9

Acknowledgements

The authors would like to thank Niclas Börlin for his contributions to the thesis. Niclas assisted us by lending a calibrated stereo rig and performed lens rectification on a number of photo sets.

Niclas further contributed with implementations of certain algorithms, and gave input on the organization of the thesis as well as the contents of the report.

We want to thank M. Lourakis for his quick reply regarding a question about the `sba` demo, Hartley and Zisserman for their permission to use figures from [12].

The authors also wishes to thank Arne Nordmann for the permission to use his illustration (Figure 2.5) in the thesis.

References

[1] H. Anton and C. Rorres. *Elemetary Linear Algebra*. John Wiley and Sons, Inc., 9th edition, 2005. Wiley Internationl Edition.

[2] E. P. Baltsavias. *Multiphoto Geometrically Constrained Matching*. PhD thesis, Institute of Geodesy and Photogrammetry, ETH, Zürich, Switzerland, 1991.

[3] N. Börlin and P.-Å. Wedin. A Second Look at Radiostereometry. Technical report, Department of Computing Science, Umeå University, Sweden, 2000.

[4] M. Brown and D. G. Lowe. Unsupervised 3D Object Recognition and Reconstruction in Unordered Datasets. pages 56–63, 2005.

[5] M. Doneus. 3x3 Rules for Simple Photogrammetric Documentation of Architecture. Webpage. `http://www.univie.ac.at/Luftbildarchiv/wgv/3x3.htm`, visited January 14, 2009.

[6] A. Fitzgibbon and A. Zisserman. Automatic 3D Model Acquistion and Generation of New Images from Video Sequences. In *In Proceedings of European Signal Processing Conference*, pages 1261–1269, 1998.

[7] W. Förstner. A Feature Based Correspondence Algorithm for Image Matching. *International Archives of Photogrammetry and Remote Sensing*, 26(3/3):150–166, 1986. Rovaniemi.

[8] A. Gruen. Least squares matching: a fundamental measurement algorithm. In K. B. Atkinson, editor, *Close Range Photogrammetry and Machine Vision*, chapter 8, pages 217–255. Whittles, Caithness, Scotland, 1996.

[9] E. Gülch, H. Müller, and T. Läbe. Integration of Automatic Processes into Semi-Automatic Building Extraction. In *ISPRSGIS99*, pages 177–186, 1999.

[10] C. J. Harris and M. Stephens. A Combined Corner and Edge Detector. In *4th Alvey Vision Conference*, pages 147–151, Manchester, 1988.

[11] R. I. Hartley. In Defense of the Eight-Point Algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580–593, 1997.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2nd edition, 2006.

[13] J. L. Lerma, C. Portalés, and D. Germes. Documenting, Maniging & Visualizing a Huge Digital Photogrammetric Data Set[sic]. Technical report, Dept. Ingeniería Cartográfica, Geodesia y Fotogrametría. Universidad Politécnica de Valencia, 2003.

[14] M. Lhuillier and L. Quan. Match Propagation for Image-Based Modeling and Rendering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:2002, 2002.

[15] M. Lhuillier and L. Quan. Quasi-Dense Reconstruction from Image Sequence. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part II*, pages 125–139, London, UK, 2002. Springer-Verlag.

[16] H. C. Longuet-Higgins. A Computer Algorithm for Reconstructing Scene from Two Projections. *Nature*, 293:133–135, 1981.

[17] M. I. A. Lourakis and A. A. Argyros. The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[18] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[19] T. Luhmann, S. Robson, S. Kyle, and I. Harley. *Close Range Photogrammetry – Principles, Methods and Applications.* Whittles Publising, ISBN 1-870325-50-8, 2006.

[20] Q.-T. Luong. Determining the Fundamental Matrix with Planes: Instability and new Algorithms. In *In Proc. Conference on Computer Vision and Pattern Recognition*, pages 489–494, 1993.

[21] C. McGlone. *Manual of Photogrammetry.* American Society of Photogrammetry and Remote Sensing, ISBN 1-57083-071-1, 5th edition, 2004.

[22] H. Müller. Object-oriented Modeling for the Extraction of Geometry, Texture and Reflectance from Digital Images.

[23] P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3):85, 2007.

[24] W. K. Nicholson. *Elementary Linear Algebra.* PWS-KENT Publishing Company, ISBN 0-534-92189-2, 2nd edition, 1990.

[25] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 2nd edition, 2006.

[26] P. Pohl. *Grundkurs i numeriska metoder.* Liber, first edition, 2005.

[27] R. Ramamoorthi and J. Arvo. Creating Generative Models from Range Images. In *Proceedings of SIGGRAPH 99*, pages 195–204, 1999.

[28] DI. F. Rottensteiner. Semi-automatic Extraction of Buildings Based on Hybrid Adjustment Using 3D Surface Models and Management of Building Data in a TIS. 2001.

[29] N. Snavely. Bundler: Structure from Motion for Unordered Image Collections. Webpage. `http://phototour.cs.washington.edu/bundler/`, visited January 14, 2009.

[30] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring Photo Collections in 3D. In *ACM Transactions on Graphics*, pages 835–846. Press, 2006.

[31] I. Söderkvist and P.-Å. Wedin. Determining the Movements of the Skeleton Using Well-Configured Markers. *Journal of biomechanics*, 26(12):1473–7+, 1993.

[32] H. Stewénius, C. Engels, and D. Nistér. Recent Developments on Direct Relative Orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

[33] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, ISBN 0-9614088-4-7, 3rd international edition, 2005.

[34] R. Szeliski and P. H. S. Torr. Geometrically Constrained Structure from Motion: Points on Planes. In *In European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pages 171–186, 1998.

[35] T. Werner, F. Schaffalitzky, and A. Zisserman. Automated Architecture Reconstruction from Close-Range Photogrammetry. In *In Proceedings on CIPA 2001 International Symposium: Surveying and Documentation of Historic Buildings – Monuments – Sites, Traditional and Modern Methods*, 2001.

[36] D. Zwillinger. *"CRC Standard Mathematical Tables and Formulae"*. 30th edition, 1996.

Appendix A

Conic Classification Algorithm

Note that the algorithm described here ignore imaginary curves. If there are multiple substitutions in the same step these are performed simultaneously.

if $b \neq 0$ **then**
  Let $q = \sqrt{\left(\frac{c-a}{b}\right)^2 + 1} + \frac{c-a}{b}$ and perform the substitutions $x \to qx$ and $y \to qy$.
end if
Now $b = 0$
if $c = 0$ **then**
  Perform the substitutions $x \to y$ and $y \to x$
end if
Now $c \neq 0$
if $g \neq 0$ **then**
  Perform the substitution $y \to y - \frac{g}{2c}$
end if
Now $g = 0$
if $a = 0$ **then**
  if $f \neq 0$ **then**
    Perform the substitution $x \to x - \frac{d}{f}$. Now $d = 0$. Dived the equation with c. The conic is a parabola. On the form $y^2 = \alpha x$ where $\alpha = -\frac{f}{c}$.
  else
    The conic is degenerate.
    if $d = 0$ **then**
      The conic is the line $y = 0$ with multiplicity two.
    else if $d < 0$ **then**
      The conic is the two parallel lines $y = \pm\sqrt{-\frac{d}{c}}$
    end if
  end if
else
  if $f \neq 0$ **then**
    Perform the substitution $x \to x - \frac{f}{2a}$
  end if
  Now $f = 0$
  if $d \neq 0$ **then**
    Divide the equation with d.

Now $d = 1$
if $\text{sgn}(a) \neq \text{sgn}(c)$ **then**
The conic is a hyperbola with $\alpha = \frac{1}{\sqrt{a}}$ and $\beta = \frac{1}{\sqrt{c}}$. If necessary exchange x and y to make a positive.
else if $a \geq 0$ **and** $c \geq 0$ **then**
The conic is an ellipse with $\alpha = \frac{1}{\sqrt{a}}$ and $\beta = \frac{1}{\sqrt{c}}$. If $a > c$ exchange x and y so that $a \leq c$. If $\alpha = \beta$ then the conic is a circle.
end if
else
If a and c has different signs then the conic represents a line, $y = \alpha x$ where $\alpha = \pm\sqrt{-\frac{c}{a}}$. If a and c has the same sign then the conic is a point, the origin.
end if
end if

Since the curve produced by this algorithm is on its normal form the curve is not a true representation of the conic, so for each point x_i derived from the standard equation it must be transformed $x_i \rightarrow x'_i$ so that $x_i'^{\mathrm{T}} \mathtt{C} x'_i = 0$ holds. This is done by reversing the substitutions performed on the equation on the point i.

Appendix B

Cylinder Reconstruction

B.1 Quadric Representation Revisited

A quadric is defined by a symmetric 4×4 matrix where all points on the surface satisfy $\mathbf{X}^{\mathrm{T}}\mathtt{Q}\mathbf{X} = 0$. The *dual quadric* is a function that defines all planes tangent to the surface. If $\mathtt{Q}^*$ is the adjoint matrix of $\mathtt{Q}$, then $\boldsymbol{\pi}^{\mathrm{T}}\mathtt{Q}^*\boldsymbol{\pi} = 0$, where $\boldsymbol{\pi}$ is a homogeneous vector describing a plane tangent to the quadric. The 2D analog is the *dual conic*, with $\mathbf{l}^{\mathrm{T}}\mathtt{C}^*\mathbf{l} = 0$, where the line $\mathbf{l}$ is tangent to the conic $\mathtt{C}$.

A projection of the quadric can be described in simple terms using the dual representation, namely $\mathrm{P}\mathtt{Q}^*\mathrm{P}^{\mathrm{T}} = \mathtt{C}^*$, where P is an arbitrary camera matrix.

The idea here is to use the projected conics to reconstruct the quadric.

For two cameras P, P' with the quadric $\mathtt{Q}$ projected to $\mathtt{C}$ and $\mathtt{C}'$ respectively, we have a system:

$$\begin{cases} \mathtt{C}^* &= \mathrm{P}\mathtt{Q}^*\mathrm{P}^{\mathrm{T}} \\ \mathtt{C}'^* &= \mathrm{P}'\mathtt{Q}^*\mathrm{P}'^{\mathrm{T}} \end{cases}$$

This is the system we want to solve in order to determine the quadric from two views, where it is being projected as a conic.

Now we take advantage of the fact that $\mathtt{Q}^*$ and $\mathtt{C}^*$ are symmetric, and rewrite the system to:

$$\begin{cases} \mathrm{B}\mathbf{v}^* &= \mathbf{c}^* \\ \mathrm{B}'\mathbf{v}^* &= \mathbf{c}'^* \end{cases}$$

$\mathrm{B}\mathbf{v}^* = \mathbf{c}^*$ has 6 equations, but a conic only has 5 degrees of freedom. Let's add the scaling explicitly.

$$\begin{cases} \mathrm{B}\mathbf{v}^* &= \alpha\mathbf{c}^* \\ \mathrm{B}'\mathbf{v}^* &= \beta\mathbf{c}'^* \end{cases}$$

Naturally this can be rewritten as a single matrix system:

$$\begin{bmatrix} \mathrm{B} & \mathbf{c}^* & \mathbf{0} \\ \mathrm{B}' & \mathbf{0} & \mathbf{c}'^* \end{bmatrix} \begin{bmatrix} \mathbf{v}^* \\ \alpha \\ \beta \end{bmatrix} = \mathbf{0}$$

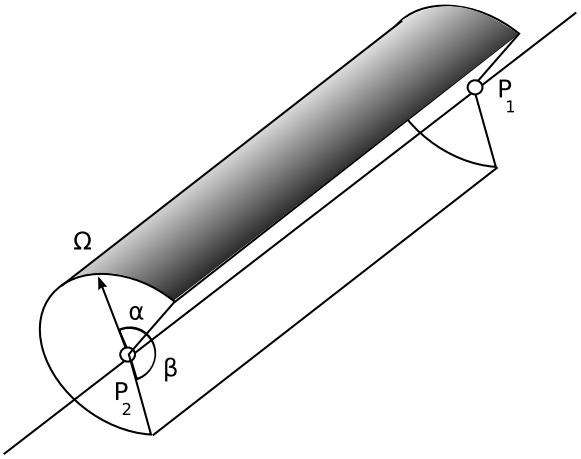


Figure B.1: Parameterized cylinder.

The null space will have several dimensions (there are an unlimited number of quadrics that satisfies these conics), and one must have additional data to get a unambiguous reconstruction.

B.2 Cylinder Parameterization

A cylinder can be parameterized as the set

$$C = \{\mathbf{P_1}, \mathbf{P_2}, \mathrm{Q}, \tilde{\mathbf{\Omega}}, \alpha, \beta\}$$

where $\mathbf{P_1}$ and $\mathbf{P_2}$ is two point lying on the center line of the cylinder in $\mathbb{P}^3$. Furthermore Q is the quadric defining the cylinder. $\mathbf{\Omega}$ is the "direction" vector which defines the base direction and is perpendicular to the base line and its Euclidean 2-norm is the radius of the cylinder. The scalars α and β defines how much of the cylinder which has a surface. Figure B.1 visualizes a parameterized cylinder.

Appendix C

List of Software Libraries

List of software libraries used for the prototype application along with their licences and web address where more information and documentation can be found.

- **Glade Interface Designer** GPL `http://glade.gnome.org/`
- **GTK+** LGPL `http://www.gtk.org/`
- **Octave** GPLv3 `http://www.octave.org/`
- **pysqlite** Custom license `http://pysqlite.org/`
- **Pytave** GPLv3 `https://launchpad.net/pytave/`
- **sba** GPL `http://www.ics.forth.gr/~lourakis/sba/`
- **SIFT++** Custom license `http://vision.ucla.edu/~vedaldi/code/siftpp/siftpp.html` Note: This library has, since this thesis was written, been superseded by a new library called VLFeat, which is distributed under the GPLv2 license.
- **SQLite 3** Public domain `http://www.sqlite.org/`
- **VTK** VTK license (BSD-derived license) `http://www.vtk.org/`
- **essmat5.m** License unknown.
- **ransac.m** Permissive license. From Peter Kovesi's homepage.