Qualitative geometric scene understanding using planar surfaces in multiple views.

Jacob Buhl

Master thesis in Vision, Graphics and Interactive Systems

Aalborg University, Summer 2011

AALBORG UNIVERSITET

The Faculties of Engineering, Science and Medicine
Department of Electronic Systems
Frederik Bajers Vej 7
Phone: +45 96 35 86 00
http://es.aau.dk

Title:
Qualitative geometric scene understanding using planar surfaces in multiple views.

Project period:
University of cental Florida:
November 1st 2010 - June 30, 2011
Aalborg University:

August 1st, 2011 - August 17, 2011

Group members:
Jacob Buhl

Supervisor:
Thomas B. Moeslund (Aalborg Univertity)
Dr. Mubarak Shah (University of central Florida)

Number of copies: 3
Number of pages: 81

Appended documents:

(10 appendix, 1 DVD)
Total number of pages: 146

Finished: August 2011

Abstract:

This project deals with the problem of automatically generating a qualitatively geometric scene understanding using planar surfaces in multiple views. Geometric scene understanding are useful for, among other things scene analysis's and autonomous robots. Much research have been done on the multiple view qualitative geometric scene understanding methods and the single image qualitative methods. This thesis focus on the generating a qualitative geometric scene understanding using multiple frames but without doing reconstruction. The system consists of four module which are evaluated quantitatively at the end of each module. "Pre-processing", which detect blur and noise in each frames and discard bad frames. "Feature extraction", which extract feature point correspondences between two frames using SIFT and KLT. The extracted feature points are improved with respect to the epipolar geometry to archive correct and accurate correspondences. "Layer extraction", which extract the planar surfaces in two views using an extended RANSAC. The initial layer segmentation are improved using graph cut. "Geometric scene understanding", which estimates layer connections, relative depths, and orientations. These features are then combined using simple to generate a qualitative geometric scene understanding. The entire system is evaluated qualitatively on two videos. It is concluded that accurate layer segmentation is vital for the system performance. Provided with accurate layer segmentation is the system able to generate a good and precise geometric scene understanding. This project makes the following contribution, an extension to RANSAC and a simple and intuitive reasoning algorithm for generating a qualitative geometric scene understanding using homographies and layer segmentations.

Preface

This report constitutes the master's thesis of Jacob Buhl at the Vision, Graphics and Interactive Systems specialization at the Department of Electronic Systems, Aalborg University. The project work spanned the period of November 1, 2010 to 30 June, 2011 at University of Central Florida and from August 1 2011 to August 17, 2011 at Aalborg University.

The thesis describes the development of a system that generates a qualitative geometric scene understanding using multiple frames of a scene. The thesis consists of an analysis, where the problem of geometric scene understanding and the use of planar surfaces in multiple views are analyzed, which leads to a System Specification for the system. This is followed by chapters that describe the structure of the system and the design of its underlying modules. The system is qualitatively evaluated on two video sequences, which leads to a conclusion on the project, followed by suggestions on future work that can be done to improve the system. This is followed by appendices which explains theory used to support the description of the design of the modules. Lastly, the sources used throughout the thesis are listed in the Bibliography chapter.

The author would like to thank Dr. Shah and Fulbright, for making it possible to conduct research at University of Central Florida.

Reading Instructions

Some words and expressions that are used throughout the thesis, are written correctly the first time they are used, followed by an abbrevation stated in parentheses. This abbrevation is used hereafter. For instance, Scale Invariant Feature Transform (SIFT) will be written as SIFT after the first occurrence. Throughout the thesis are vectors written as $\mathbf{v}$, matrices as M, and homogeneous representation as $\widetilde{hr}$. References are written as numbers in square brackets, e.g. [1]. This number then corresponds to the number written in the Bibliography chapter, where information about the source, such as the name of the author, publication and the publisher, etc. can be found.

The Internet sources are included on the attached CD in the back of this thesis. Some of the datasets used for validation and the validation result can also be found on the CD.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

One of the ultimate goals of computer vision is to understand an unconstrained real world scene. A computer needs to interpreting the scene structure, objects and events to completely understand the scene. Humans and animal are very good at these tasks, but it is very difficult for a computer to perform them accurately. Different branches of computer vision work with different aspect of scene understanding as the question

What does it mean to understand a visual scene?

can not be answered explicitly.

Figure 1.1: *High level of detail in a real world scene*

One popular branch of scene understanding are segmentation. Segmentation is the process of labeling objects in the scene, such as for Figure 1.1 road, humans, costume bike, ect,[1, 2, 3]. These object are large object and each contains many levels of details, which we as humans are able to distinguish and label very quickly. For example looking to the left of image we can quickly tell that there are two men, one black one white, both with a camera and jeans on.

To solve this labeling process, the algorithm would be trained on a set of labeled data and then be required to recognize similar objects in unknown scenes. This type of classification do not recognize the context which the object appear. Therefore, is it very difficult to distinguish an image of an image of an object from an image of the actual object.

This semantic approach to scene understanding is somewhat superficial to the underlying structure. To rally understand the scene it is necessary to obtain some clues about the geometric structure of the scene, e.g. 3D information. This includes answering questions such as "What region of the image is near and what region is far?" and "What regions is free space, walkable surfaces?". The geometric understanding is part of the scene reconstruction aspect of scene understanding. The 3D reconstruction from images is a widely explored research area in the computer vision community[4, 5, 6, 7, 8, 9, 10].

Figure 1.2: *3D reconstruction into a potemkin city.[10]*

This thesis deals with the problem of recovering geometric scene understanding.

1.1 Initial problem

The introduction leads to the following initial problem:

How can a geometrical scene understanding be obtained?

This question is answered in the following chapter through an analysis of the problem.

Part I

Analysis

Before setting up the requirements of the system, the use of geometric scene understanding in general and related work, are analyzed. First, the use and related work of geometric scene understanding is covered. Next, the camera model, projection from 3D to 2D and the parameters that influenced the recorded images are covered. Last, the advantages and use of planar surfaces in multiple views are covered in detail. Based on the knowledge of geometric scene understanding, camera models and the geometry of planar surfaces, the System Specifications can be defined.

Chapter 2
Geometric scene understanding

Geometric scene understanding is a broad and very popular research area. It includes all forms of simple obstacle avoidance to dense reconstruction of cities. In this section a general introduction to the use of geometric scene understanding is provided, along with an overview of some of the research.

The principle of geometric scene understanding is to obtain a geometric representation of a scene along with some input data. The geometric scene understanding methods can be divided into two groups based on the method used to obtain the input data. The first group is active methods. Active methods use interaction with the object/scene that requires input data. The list of active methods includes, among others, moving light sources, colored visible light, time-of-flight lasers to microwaves or ultrasound. The second group is passive methods which do not interact but measure the radiance reflected or emitted from the scene. One typical passive sensor is a camera, which is the only one considered in this thesis.

This section covers some of the applications and domains that use geometric scene understanding. One domain that requires geometric scene understanding is autonomous robots – mobile robots, which are able to move autonomously through an environment from its current position to a goal position. These can be beneficial in civilian applications such as search- and rescue activities, wide-area environment monitoring, disaster recovering, as well as planetary exploration and military operations. To do this the robot must be able to freely navigate on cluttered an unstructured outdoor environments without any collisions. The robot does not need metric geometric information about the objects, only the approximate location and the area they occupy.

Further more the robot needs the ability to interact with the scene, such as picking up an object. This type of application demands more accurate geometric information about the object, as the robot must determine the best way grip it. Among the more accurate geometric scene understating methods is reconstruction. Reconstruction is the process of generating a 3D model of the scene.

In the following section some of the previous work on obtaining a geometric scene understanding is covered.

2.1 Previous work

One of the very first attempts to do geometrical scene understanding using images was back in 1963 by Lawrence G. Roberts[11]. He uses the perspective projection of lines in a single image, together with decomposition of 3D objects into known 3D primitives. The final results display the three-dimensional structure with all the hidden lines removed from any point of view. This was the first attempt to construct a complete scene understanding system. The system was designed for a very constrained environment, only consisting of textureless polyhedral shapes.

Newer approaches have less constraints such as [12] proposed by Saxena et al. They only

assume that the scene is constructed of small planes. Saxena et al. uses a Markov Random Field on "superpixels" to infer a set of "plane parameters" that capture both the 3D location and orientation, see Figure 2.1.

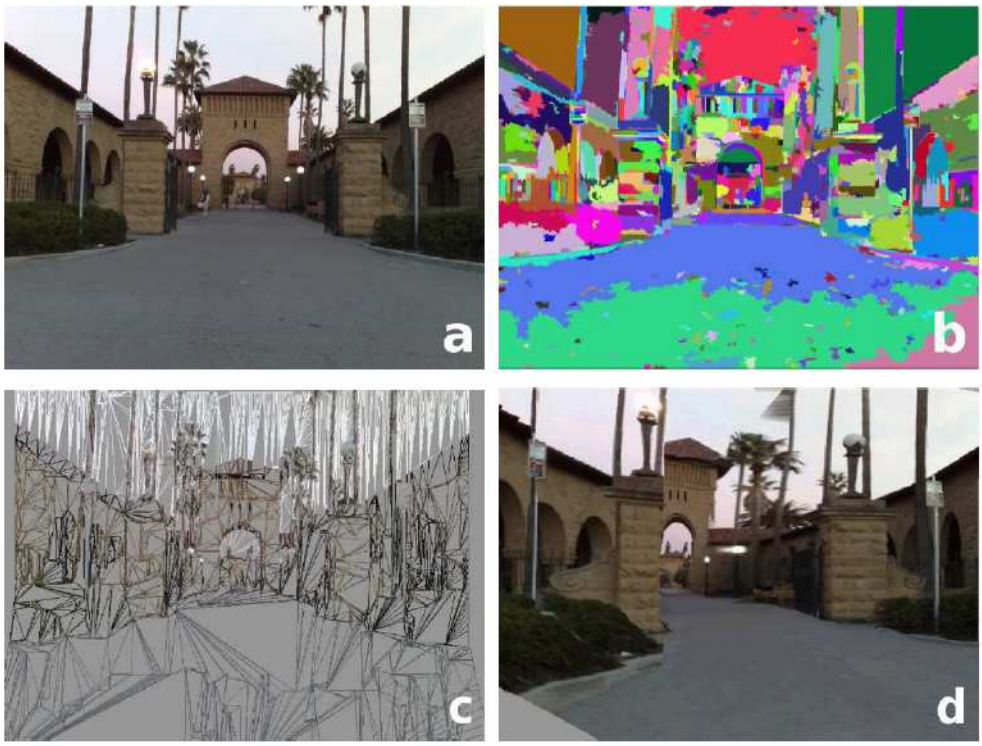


Figure 2.1: *(a) An original image. (b) Oversegmentation of the image to obtain "superpixels". (c) The 3D model predicted by the algorithm. (d) A screenshot of the textured 3D model.[12].*

Recently a new approach, Blocks World, was proposed by Gupta et al. [13], which is inspired by the work from Roberts. The method results in a qualitative physical representation of objects, see Figure 2.2. In this representation, objects have volume, mass and relationships. They use single monocular images where a "interpretation-by-synthesis" approach is applied. They start from an empty ground plane, where they progressively "build up" a physically-plausible 3D interpretation of the image. Their contribution is to incorporate the physical constraints which follows by the use of volumes.

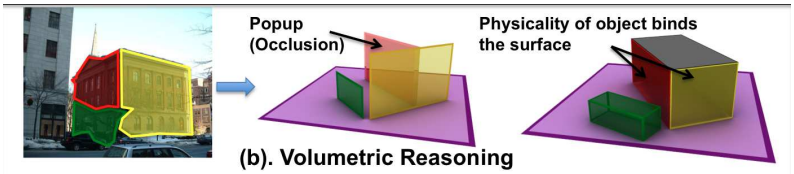


Figure 2.2: *Volumetric reasoning about a scene.[13]*

The methods mentioned so far all do geometric scene understanding from a single image, a more widely explored area is the multi view methods. The multiple view methods include reconstruction and structure from motion[14, 6, 15]. In structure from motion a points 3D location is estimating given it can be seen from multiple cameras or views points.

There exists three classes of reconstruction: projective, affine and euclidean reconstruction[15]. All three methods require multiple views of the scene or some known 3D distances. These views can

either be from a single moving camera or from multiple cameras. Many of them also require some sort of camera calibration. For uncalibrated cameras it is only possible to do the reconstruction up to an unknown projective deformation.

One of the simplest reconstruction methods is triangulation. Triangulation can be used to estimate the 3D point, when camera location and orientation are known[6]. The 3D point is found as the intersection of two 3D rays from the camera centers C_1 and C_1' through the image locations x and x', see Figure 2.3. It is not possible to obtain a unique solution through triangulation if camera centers are not known.

If the camera location and orientation are not known then perspective reconstruction can be performed. In perspective reconstruction, only the point correspondence x and x' are needed. The perspective reconstruction for uncalibrated cameras use Fundamental matrix and a known homography to estimate the camera matrices P, P', as Equation 2.1. Appendix E provides a description of the fundamental matrix. Using the 3D to 2D projection $\widetilde{\mathbf{x}} = P\widetilde{\mathbf{p}}$ and $\widetilde{\mathbf{x}'} = P'\widetilde{\mathbf{p}}$ the 3D point p can be determined.

The projective reconstruction can be converted into an euclidean if the camera calibration is know. The camera calibration can be obtained using self-calibration explained in Section B.2.4.

$$P_0 = [I|0] \text{ and } P_1 = [H_\pi|e'] \qquad [\cdot] \quad (2.1)$$

Where:

P_0, P_1: Camera matrices.	$[\cdot]$
H_π: Known homography for planar surface π.	$[\cdot]$
e': Epipole	$[\cdot]$

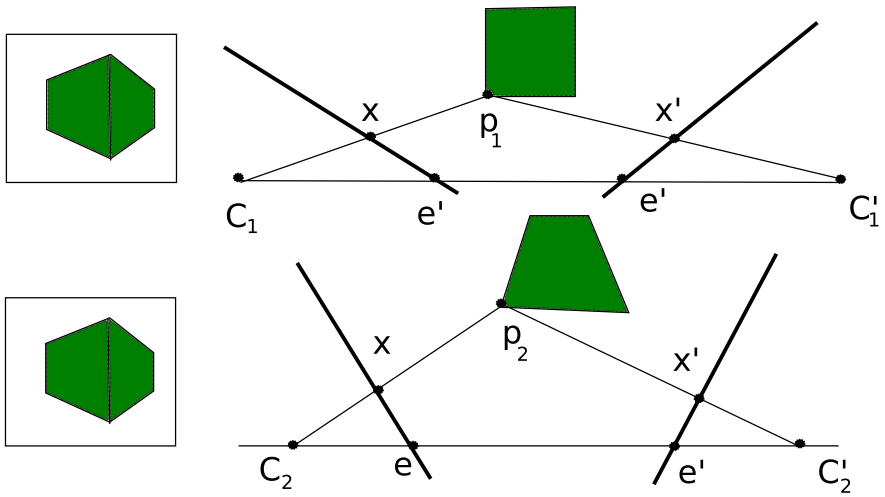


Figure 2.3: *Illustration of two scenes and four different camera locations that provide two identical images.*

Parodi et. al uses the geometrical constraints provided by the location of vanishing points to obtain an affine reconstruction[16]. The use of vanishing points is motivated by the perspective distortion. One major effect of this distortion is that a set of parallel lines (real world) converges into a single point in the image plane, see Figure 2.4. The Affine reconstruction, is a reconstruction that is only known up to a set of independent scaling factors along each axis[15].

A more useful system can be constructed by assuming that the camera is calibrated. The calibration is done up to an unknown focal length which can be obtained from (finite) vanishing

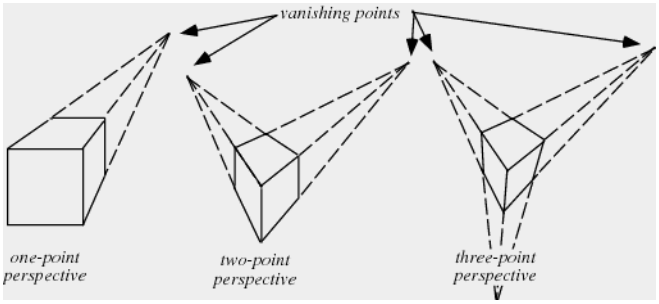


Figure 2.4: *One, two, and three vanishing point geometry*

points[17, 6]. Then, given two points with known distance, a fully euclidean reconstruction of the scene can be created.

Until now only two views have been used to create a geometric scene understanding. When more than two views are available, extended feature tracks can be obtained. From these tracks it is possible to recover the structure and motion using a process called factorization. For the orthography model, Section B.2.3, the shape and motion can be recovered simultaneously using a singular value decomposition [18]. For motion models other than pure orthography, such as the scaled orthographic, it is necessary to extend the method of [18]. A disadvantage for factorization is that all points must be visible in all images. Another disadvantage of regular factorization is that it cannot deal with perspective cameras. Christy and Horaud perform an initial affine reconstruction to correct for the perspective effects [19].

The result of factorization is a sparse 3D point cloud for all the feature tracks. To make the result more dense the simpler triangulation method can be used.

The most accurate method to recover structure and motion is Bundle adjustment[20]. Bundle adjustment performs robust non-linear minimization of the re-projection errors[21]. The Bundle adjustment calculates the re-projected feature location x using the point (track index) i and the camera pose index j, see Equation 2.2.

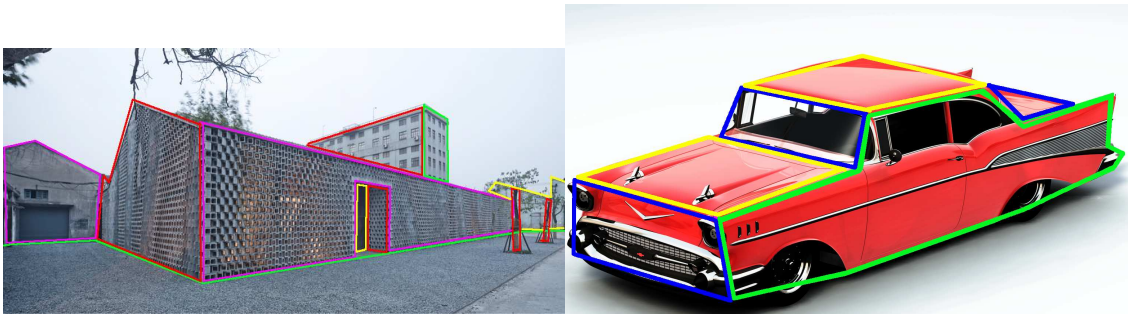
$$\mathbf{x}_{ij} = f(\mathbf{p}_i, \mathbf{r}_j, \mathbf{c}_j, \mathbf{k}_i) \qquad [\cdot] \quad (2.2)$$

Where:

$\mathbf{x}_{ij}$: Feature location for point i for camera j.	$[\cdot]$
$\mathbf{p}_i$: 3D pose for point i.	$[\cdot]$
$\mathbf{r}_j$: Camera j rotation.	$[\cdot]$
$\mathbf{c}_j$: Camera j translation.	$[\cdot]$
$\mathbf{k}_j$: Camera j intrinsic parameters.	$[\cdot]$

The previous multi-frame methods work on unconstrained scenes, but often a planar surface constraint is used to simplify the process. The planar surface constraints are motivated by many reasons. It is constrained and compact representation and man-made objects are often constructed by piecewise planar and near-planar surfaces, see Figure 2.5.

The assumption of planar surfaces makes homography valuable transformation. Google uses homographies and 3D reconstruction to generate 3D models of objects, as Google Earth is capable of generating a 3D model of entire cities[22]. H. L. Chou et. al. in [23] uses uncalibrated camera and homographies to compute the Fundamental matrix between two frames. They use the Fundamental matrix for 3D reconstruction in the projective space. Given multiple projective



(a) (b)

Figure 2.5: *Examples of planar or near planar man-made objects.*

reconstructions, they then map to a common projective space and thereby obtain the 3D model. 3D reconstruction and homographies are also used in forensic science. J. Wright et al. in [24] estimate the source of blood splatter using coplanar ellipses. They point out that a typical blood stain is an ellipse which depends on the angle between the surface and the path of the blood drop.

2.2 Summery

The most explored area of geometric scene understanding is reconstruction from two or multiple images. These methods can be used to calculate an quantitative 3D model of the scene. It is only possible to reconstruction to a projective space if the camera calibration is not known. Therefore, requires many of multi view methods a camera calibration. Almost all the research in the multiple view methods focus precision[19, 17, 6].

Given the nature single image and the loss of depth motivate for more quantitative approaches for geometric scene understanding. Obtaining a geometric scene understanding from a single image is often very computational expensive for unconstrained scenes. This is also the case for the resent single image method proposed is Blocks World. This, methods uses an extensive frame work of classifiers to obtain an qualitatively scene understanding represented by simple 3D blocks.

Common for both the single and multiple image method are that many of them assumes planar surfaces. It is motivated by the fact that man-made object often consist of planar or near planar surfaces. In the following chapter the advantages which follows use of planar surfaces and multiple views are analyzed.

Chapter 3
Geometry of Planar Scenes

In this Chapter, a brief overview of several technical terms and algorithms related to planar scenes in multiple views are given. Planar scenes are often used in geometric scene understanding because it simplifies and provide valuable information. Therefore, does this chapter also serve as a basic for the frame work developed in Part 2.

First, homography is defined along with an overview of methods to obtain and decompose it. Next, different methods used to determine all planar surfaces in a scene are covered. A 2D projection of a planar surface is also referred to as a layer.

Appendix B describes basic concepts of a camera. First, the transformation from world point to image coordinate is derived for different camera models. The simpler pinhole model is the most used camera model in literature. The pinhole model represent the ideal camera where each world point is related with the image plane through one ray starting at the nodal point of the camera.

The appendix also describes two different transformations, orthography and perspective. Orthography is a simpler model where the distance to the world point is disregarded. Often projective transformation is preferred as it accounts for the depth. The projective 3D to 2D transform is described by a camera matrix.

3.1 What is a homography

Homography is a collineation [1] , e.g. an invertible transformation of points and lines on the projective plane. It is represented by a 3×3 matrix H, which transforms a random homogeneous point on the view plane of a planar surface from one image $\widetilde{\mathbf{x}}$ to another $\widetilde{\mathbf{x}'}$ up to a scale λ, see Figure 3.1 and Equation 3.1.

$$\widetilde{\mathbf{x}'} = \lambda H \widetilde{\mathbf{x}} = \lambda \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \widetilde{\mathbf{x}} \qquad [\cdot] \qquad (3.1)$$

Where:

$\widetilde{\mathbf{x}}$: Planar surface point in first image. $[\cdot]$

$\widetilde{\mathbf{x}'}$: Planar surface point in second image. $[\cdot]$

H: Homography between image one and two $[\cdot]$

Let the camera locations in the two images be $C = [I|\mathbf{0}]$ and $C' = [R|\mathbf{t}]$. Where C is the reference and C' is located with a rotation R and translation $\mathbf{t}$ relative to C. $\mathbf{x}_\pi = [xyz]$ is a 3D

[1] Collineation is an one-to-one map from one projective space to another.

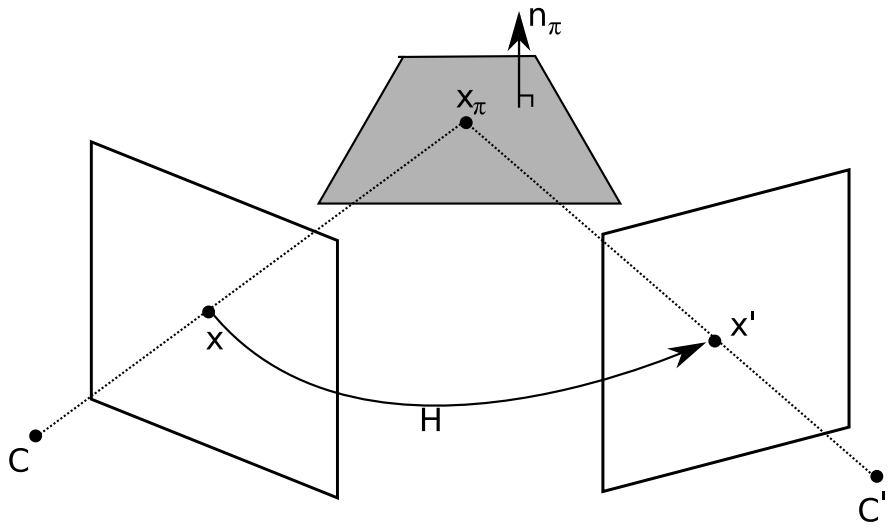


Figure 3.1: *Illustration of homography H between two images and a planar surface π. $\mathbf{x}$ and $\mathbf{x}'$ are the projections of point $\mathbf{x}_\pi$ on to the view plane. C and C′ are the respective cameras locations for image one and two.*

point located on the plane π with surface normal $\mathbf{n}_\pi$ and distance d from the first camera. $\mathbf{x}$ and $\mathbf{x}'$ are its projections. Given Equation B.5 the projection can be expressed as Equation 3.2, where p represents different points on the 3D line between the camera C and the x_π, see Figure 3.1.

The value of p that satisfies the relationship $\mathbf{x}_\pi = [\mathbf{x}^T\ p]^T$ is $\frac{-{\mathbf{n}_\pi}^T}{d}\mathbf{x}$ which is given from the plane equation. Equation 3.4 is reached by substitution p with $-\mathbf{n}_\pi x$ for the second image, see Equation 3.3[15].

$$\widetilde{\mathbf{x}} = K\ Cx_\pi = K\,[I|\mathbf{0}]\,\mathbf{x}_\pi \qquad [\cdot] \qquad (3.2)$$

$$\widetilde{\mathbf{x}'} = K'\ C'x_\pi = K'\,[R|\mathbf{t}]\left[\mathbf{x}_\pi^T\ -\frac{{\mathbf{n}_\pi}^T}{d}x\right]^T \qquad [\cdot] \qquad (3.3)$$

$$H = K'(R - \mathbf{t}\frac{{\mathbf{n}_\pi}^T}{d})K^{-1} \qquad [\cdot] \qquad (3.4)$$

Where:

$\widetilde{\mathbf{x}}$, $\widetilde{\mathbf{x}'}$: Homogeneous representation of x and x'.	$[\cdot]$
K, K': Calibration matrix for camera one and two.	$[\cdot]$
C, C': Camera location for image one and two.	$[\cdot]$
R, $\mathbf{t}$: Camera rotation and translation relative to C.	$[\cdot]$
$\mathbf{n}_\pi$: Surface normal of surface π.	$[\cdot]$

3.2 Relationship between homography and fundamental matrix

This section covers the differences and relations between homography and fundamental matrix. Appendix E provides a description of the fundamental matrix and epipolar geometry.

The main difference is that homography depends on the normality of a planar surface where the fundamental matrix depends on the epipolar geometry. They both are a 3×3 matrix but the fundamental matrix only have rank 2.

C. Sagüés et. al. [25] derive the fundamental matrix from two homographies using Equation E.2. Given the two homographies $H_{\pi 1}$ and $H_{\pi 2}$ a mapping from the image to itself exist, $H = H_{\pi 1}\ H_{\pi 2}^{-1}$. Under this mapping the epipole remains fixed such that $e' = He'$ is valid. The epipole can be determined by the eigenvector of the non unary eigenvalue. Given the epipole the fundamental matrix can be determined through either of the two equation 3.5 and 3.6.

$$F = e' \times H_{\pi 1} \quad [\cdot] \quad (3.5)$$
$$F = e' \times H_{\pi 2} \quad [\cdot] \quad (3.6)$$

In the previous sections the homography has been derived and related to the Fundamental matrix. Next, the different methods for estimating the homography are described.

3.3 Homography estimation

This section covers the different methods for obtaining homography. The different cost functions which can be used to evaluate homography are described in Appendix D "Cost functions".

3.3.1 Basic DLT algorithm

The Direct Linear Transform (DLT) algorithm is a simple homography estimator which uses point correspondences. Using homogeneous coordinates the mapping between $\widetilde{\mathbf{x}} \rightarrow \widetilde{\mathbf{x}'}$ can be expressed as Equation 3.7, where λ is a non-zero scale factor. The function $G(\cdot)$ normalizes the vector with respect to the third element, such that homogeneous representation is obtained for $\mathbf{x}'$.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = G\left(H\widetilde{\mathbf{x}}\right) = G\left(\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\right) \quad [\text{px}] \quad (3.7)$$

Where:

$\widetilde{\mathbf{x}'} = [u\ v\ 1]^T$	[px]
$\widetilde{\mathbf{x}} = [x\ y\ 1]^T$	[px]
H: 3×3 Homography.	$[\cdot]$
λ: Scale factor.	$[\cdot]$
$G(\cdot)$: Normalizes the vector with respect to the third element.	$[\cdot]$

Equation 3.7 can be rewritten such that Equation 3.8 and 3.9 are obtained.

$$u = \frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + h_9} \quad [\text{px}] \quad (3.8)$$
$$v = \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + h_9} \quad [\text{px}] \quad (3.9)$$

Multiplying and subtracting the left part of Equation 3.8 and 3.9 with the right part results in Equation 3.10 and 3.11.

$$-h_1 x - h_2 y - h_3 + (h_7 x + h_8 y + h_9)u = 0 \quad [\text{px}] \quad (3.10)$$
$$-h_4 x - h_5 y - h_6 + (h_7 x + h_8 y + h_9)v = 0 \quad [\text{px}] \quad (3.11)$$

Equation 3.10 and 3.11 can then be written in matrix form such that Equation 3.12 is obtained.

$$A_I\ \mathbf{h} = 0 \qquad [\text{px}] \qquad (3.12)$$

$$A_I = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & u \cdot x & u \cdot y & u \\ 0 & 0 & 0 & -x & -y & -1 & v \cdot x & v \cdot y & v \end{bmatrix} \qquad [\text{px}] \qquad (3.13)$$

$$\mathbf{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^1 \qquad [\text{px}]$$

$$A\ \mathbf{h} = 0 \qquad [\text{px}] \qquad (3.14)$$

Each point correspondence provides two equations, therefore a minimum of four point correspondences are necessary to solve for homography, as it has 8 DoF. Given four points, no three points can be collinear, (i.e. lie on the same line)[15].

Each point correspondence $(\mathbf{x_i}, \mathbf{x_i'})$ provides a 2×9 matrix A_I, given n points the matrices can be stacked such that a $2n \times 9$ matrix A is obtained, see Equation 3.14. If more than four point correspondences are used and the points are exact then there will exist an unique solution. This unique solution exists because A will have rank 8. In practice, point correspondences are inexact given noise etc, therefore A will not be rank deficient as it will have rank 9[26]. In this case, no non-zero solution to Equation 3.14 exists. The best estimate of homography is obtained by minimizing a cost function with respect to $\mathbf{h}$, see Appendix D. The minimization is constrained by $||\mathbf{h}|| = 1$ such that a non-zero solution is found.

Using Largrange multipliers and algebraic distance the solution to $\mathbf{h}$ can be found as unit eigenvector of $A^T A$ corresponding to the smallest eigenvalue of A. Consequently, the solution can be found through Singular Value Decomposition, SVD, of A[27]. The obtained solution is dependent on the origin and scale of the coordinate system in the image[15][26], so a normalization step is introduced.

Normalization

Hartley in [26] describes a normalization step using transformation of the point correspondences such that $\hat{\mathbf{x}}_\mathbf{i} = T\mathbf{x_i}$ and $\hat{\mathbf{x}}_\mathbf{i}' = T'\mathbf{x_i'}$. The normalization can be divided into two parts: transformation of points, and transformation of homography. The process of estimating the homography is

1. Transform the image coordinates according to $\hat{\mathbf{x}}_\mathbf{i} = T\mathbf{x_i}$ and $\hat{\mathbf{x}}_\mathbf{i}' = T'\mathbf{x_i'}$
2. Estimate homography $\hat{H}$ using point correspondences $\hat{\mathbf{x}}_\mathbf{i}$ and $\hat{\mathbf{x}}_\mathbf{i}'$.
3. Set $H = T'^{-T}\hat{H}T$

Hartley found that the best transformations T and T' are the ones that translated points so that their centroid is at the origin and then points are scaled isotropically then the average distance from the origin is equal to $\sqrt{2}$.

The translation $\mathbf{t} = [t_1\ t_2]$ is calculated as the mean of image coordinates $X = [\mathbf{x_1}\ \mathbf{x_2} \dots \mathbf{x_i}]$. The scale s is calculated using Equation 3.15 and the final transform T is obtained by Equation 3.16.

$$s = \sqrt{2}/_{mean(dist(X - \mathbf{t}))} \qquad [\cdot] \qquad (3.15)$$

$$T = \begin{bmatrix} s & 0 & -s \cdot t_1 \\ 0 & s & -s \cdot t_2 \\ 0 & 0 & 1 \end{bmatrix} \qquad [\cdot] \qquad (3.16)$$

3.3.2 Line features

So far only point correspondences have been used to calculate the homography H. In this section, lines are used and in the following section, points and lines are combined.

A line in 2D can be represented as equation $ax + by + c = 0$, where a, b and c are the line parameters. The line l can be represented as a vector $\mathbf{l} = [a\ b\ c]^T$. The line is considered homogeneous. Any scalar multiple of the vector is the same line. Therefore, the line only has 2 DoF. Given a line in two images represented as $\widetilde{\mathbf{l}}$ and $\widetilde{\mathbf{l}'}$. Let $\widetilde{\mathbf{x}}$ be a point on line $\widetilde{\mathbf{l}}$ and $\widetilde{\mathbf{x}'}$ on line $\widetilde{\mathbf{l}'}$. Then, $\mathbf{x}$ lies on the line $\widetilde{\mathbf{l}}$ if and only if $\widetilde{\mathbf{x}}^T\widetilde{\mathbf{l}} = 0$ or $\widetilde{\mathbf{l}}^T\widetilde{\mathbf{x}} = 0$. The line homography Equation 3.18 is obtained by substituting Equation 3.1, $\widetilde{\mathbf{x}'} = H\widetilde{\mathbf{x}}$ in Equation 3.17.

$$\widetilde{\mathbf{l}}^T\widetilde{\mathbf{x}} = 0 \text{ and } \widetilde{\mathbf{l}'}^T\widetilde{\mathbf{x}'} = 0 \qquad [\cdot] \qquad (3.17)$$

$$\widetilde{\mathbf{l}'} = H^T\widetilde{\mathbf{l}} \qquad [\cdot] \qquad (3.18)$$

Where:

$\widetilde{\mathbf{l}}$: Homogeneous line in image one. $[\cdot]$

$\widetilde{\mathbf{l}'}$: Homogeneous line in image two. $[\cdot]$

Following the same derivation as of DLT, Section 3.3.1, matrix A_i for lines is given as Equation 3.19. Where $\widetilde{\mathbf{l}} = [a\ b\ 1]^T$ and $\widetilde{\mathbf{l}'} = [u\ v\ 1]^T$. The homography is obtained exactly as in DLT just with the matrix A computes as Equation 3.19.

$$A_I = \begin{bmatrix} -a & 0 & au & -b & 0 & bu & -1 & 0 & u \\ 0 & -a & av & 0 & -b & bv & 0 & -1 & v \end{bmatrix} \qquad [\cdot] \qquad (3.19)$$

Where:

$\widetilde{\mathbf{l}} = [a\ b\ 1]^T$: Line in image one. $[\cdot]$

$\widetilde{\mathbf{l}'} = [u\ v\ 1]^T$: Line in image two. $[\cdot]$

In the same way as a normalization improves the estimated homography in the presence of noise, with the point correspondences, it also improves the line homography. Zeng et al. proposed in [28] the following method for normalized line-based homography estimation. The method consists of two transformations instead of only one as in DLT normalization. Given a set of lines $L = [\mathbf{l}_1\ \mathbf{l}_2\ \ldots \mathbf{l}_i]$, where $\mathbf{l}_i = [a_i\ b_i\ c_i]^t$. Then, the normalization consists of a translation T_1 and a scaling T_2 constructed as follows:

$$T_1 = \begin{bmatrix} 1 & 0 & -t_1/t_3 \\ 0 & 1 & -t_2/t_3 \\ 0 & 0 & 1 \end{bmatrix} \qquad [\cdot] \qquad (3.20)$$

$$[a'_i\ b'_i\ c'_i]^T = T_1[a_i\ b_i\ c_i] \qquad [\cdot]$$

Where:

$t_1 = \sum_i a_i,\ t_2 = \sum_i b_i,\ t_3 = \sum_i c_i$: Sum of line components. $[\cdot]$

Given the tranlated line $[a'_i\ b'_i\ c'_i]$, the scaling transform T_2 is defined as:

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} \qquad [\cdot] \qquad (3.21)$$

Where:

$$s = \left[\frac{\sum_i \left(a_i'^2 + b_i'^2 \right)}{2 \cdot \sum_i c_i'^2} \right]^{1/2} : \text{Scaling factor. } [\cdot]$$

The final normalizaion of a line $\widetilde{\mathbf{l}}$ becomes:

$$\hat{\mathbf{l}} = T_2 \, T_1 \, \widetilde{\mathbf{l}} \qquad [\cdot]$$

3.3.3 Lines and Points

The DLT algorithm described in Section 3.3.1 can easily be extended to use a combination of points and line correspondences. A_I for points in Equation 3.13 and matrix A_I for lines in 3.19 both have the same size such that they can be stacked into the same matrix A. The complex part is to normalize a set of correspondences which contains both points and lines. In section 3.3.3 the point normalization from Section 3.3.1 is used to normalize lines, point-centric. In Section 3.3.3 the line normalization from Section 3.3.2 is used to normalize points, line-centric.

Point-centric

The point-centric approach uses the point normalization transforms T and T' to normalize the line correspondences. T and T' are calculated as Equation 3.16 from point correspondences of the set x_i containing both point and line correspondences. The mapping between $\hat{\mathbf{l}}$ and $\mathbf{l}$ is given by Dubrofsky et. al. in [29]. They use the fact that given two points $\widetilde{\mathbf{x_1}} = [x_{11} \; x_{12} \; 1]$ and $\widetilde{\mathbf{x_2}} = [x_{21} \; x_{22} \; 1]$ on a line $\mathbf{l}$, the line can be formulated as the cross product, see Equation 3.22.

$$\mathbf{l} = \widetilde{\mathbf{x_1}} \times \widetilde{\mathbf{x_2}} = \begin{bmatrix} x_{12} - x_{22} \\ x_{21} - x_{11} \\ x_{11}x_{22} - x_{12}x_{21} \end{bmatrix} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} \qquad [\cdot] \qquad (3.22)$$

The normalized line $\hat{\mathbf{l}}$, can be obtained by applying the point normalization to Equation 3.22, see see Equation 3.23.

$$\hat{\mathbf{l}} = T\widetilde{\mathbf{x_1}} \times T\widetilde{\mathbf{x_2}} \qquad [\cdot] \qquad (3.23)$$

After expanding and simplifying Equation 3.23, Equation 3.24 is obtained. Substitution Equation 3.22 in 3.24 provides the point-centric normalization of line $\mathbf{l}$ as Equation 3.25.

$$\hat{\mathbf{l}} = s \begin{bmatrix} x_{12} - x_{22} \\ x_{21} - x_{11} \\ s(x_{11}x_{22} - x_{12}x_{21}) + t_1(x_{22} - x_{12}) + t_2(x_{11} + x_{21}) \end{bmatrix} \qquad [\cdot] \qquad (3.24)$$

$$\hat{\mathbf{l}} = s \begin{bmatrix} l_1 \\ l_2 \\ sl_3 + t_1l_1 + t_2l_2 \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ st_1 & st_2 & s^2 \end{bmatrix} \begin{bmatrix} l_1 \\ l_2 \\ l_3 \end{bmatrix} \qquad [\cdot] \qquad (3.25)$$

Line-centric

The line-centric approach uses the line normalization transforms T_1,T_2 and T_1', T_2' to normalize the point correspondences. T_1, T_2 and T_1', T_2' are calculated as Equation 3.20 and 3.21 from line correspondences of the set $\mathbf{x_i}$ containing both point and line correspondences. There are given two lines $\widetilde{\mathbf{l_1}} = [l_{11}\ l_{12}\ 1]$ and $\widetilde{\mathbf{l_2}}$, and the intersection point x which is a member of $\mathbf{x_i}$. The intersection point can be found as the cross product between $\widetilde{\mathbf{l_1}}$ and $\widetilde{\mathbf{l_2}}$.

$$\mathbf{x} = \widetilde{\mathbf{l_1}} \times \widetilde{\mathbf{l_2}} = \begin{bmatrix} l_{12} - l_{22} \\ l_{21} - l_{11} \\ l_{11}l_{22} - l_{12}l_{21} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad [\text{px}] \qquad (3.26)$$

The transformation that the line normalization applies to the points is derived as for point-centric, which give rise to Equation 3.27.

$$\hat{\mathbf{x}} = (T_2\ T_1\widetilde{\mathbf{l_1}}) \times (T_2\ T_1\widetilde{\mathbf{l_2}}) \qquad [\text{px}] \qquad (3.27)$$

After expanding and simplifying Equation 3.27, Equation 3.28 is obtained. Substitution Equation 3.26 in 3.28 provides the point-centric normalization of line $\mathbf{x}$ as Equation 3.29.

$$\hat{\mathbf{x}} = \begin{bmatrix} s(l_{12} - l_{22}) \\ s(l_{21} - l_{11}) \\ l_{11}l_{22} - l_{12}l_{21}) + {}^{t_1}/_{t_3}(l_{12} - l_{22}) + {}^{t_2}/_{t_3}(l_{21} - l_{11}) \end{bmatrix} \qquad [\cdot] \qquad (3.28)$$

$$\hat{\mathbf{x}} = \begin{bmatrix} sx_1 \\ sx_2 \\ x_3 + {}^{t_1}/_{t_3}x_1 + {}^{t_2}/_{t_3}x_2 \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ {}^{t_1}/_{t_3} & {}^{t_2}/_{t_3} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad [\cdot] \qquad (3.29)$$

3.3.4 Robust Estimation

The algorithms discussed so far all require some form of correspondences as input. The algorithms are only robust to low noise related to the position of the correspondences. Obtaining perfect and correct matches between features in two images is a tricky task and errors might occur, which will effect the homography.

In appendix SIFT, F and KLT Feature Point Tracker G two methods for obtaining correspondences are described. A wrong correspondence is two points $\mathbf{x}$ and $\mathbf{x}'$ which do not belong to the same real world point $\mathbf{p}$. Points a correct correspondences is often referred to as an inlier where a wrong correspondence is an outlier. This section describes two robust methods to determine both inliers and outlier.

RANSAC

Random Sample Consensus, also known as RANSAC, was introduced in 1981 by Fischler et al.[30] for fitting a model to experimental data. RANSAC is capable of handling a significant percentage of incorrect correspondences, which makes it ideal for homography estimation. Consequently, it is also one of the most used methods for robust homography estimation[15, 31, 32, 33].

RANSAC is an iterative algorithm, which estimates a model using minimum number of points, and then expands this set. There are many variants of the RANSAC algorithm, see Algorithm 1 for one common variant.

Algorithm 1 RANSAC algorithm

function RANSAC($X_{k\times n},\ \tau,\ m,\ p$)
X : n data correspondences $(x_{1,1}, \ldots, x1, k \rightarrow x_{2,1}, \ldots, x2, k)$
τ : Error distance threshold
m : Minimum number of correspondences in subset
p : Probability for one correct subset

$nBest \leftarrow 0$ ▷ Number of inliers for best model
$N \leftarrow 1$ ▷ Number of iterations
$i \leftarrow 0$ ▷ Current iteration
while $i < N$ **do**
$X_{sub} \leftarrow randsubset(X,\ m)$ ▷ Select random subset of size m from X
$model \leftarrow fitmodel(X)$ ▷ Fits the selected model to the subset.
$inliers \leftarrow Distfn(X,\ \tau)$ ▷ Evaluate error distance for each correspondence.
if $length(inliers) > nBest$ **then**
$Best\ Inliers \leftarrow inliers$
$nBest \leftarrow length(inliers)$
end if
end while
$model \leftarrow fitmodel(XBest\ inliers)$ ▷ Fit model to all inliers of best model.
return $Best\ inliers$, $model$
end function

To determine if a point is within the predefined tolerance, a threshold τ is used. The value of θ is set such that an accepted point is also a true inlier with the probability α. In practice, the values of θ are commonly set empirically[15].

The number of iterations N is determined such that it is insured that with a probability p at least one set of random samples do not contain any outliers. N can be calculated if the probability u, the probability that a sample is an inlier, is known. Then $v = 1 - u$ is the probability of observing an outlier. Given this, and solving Equation 3.30 with respect to N the minimum number of iterations can be determined by Equation 3.31. If the probability u is unknown it can adaptively be determined along with N.

In the case of homography the minimum number of correspondences required to calculate H is four, so $m = 4$.

$$1 - p = (1 - u^m)^N \qquad [\text{Percent}] \qquad (3.30)$$

$$N = \frac{log(1-p)}{log(1-v)^m} \qquad [\cdot] \qquad (3.31)$$

Where:

p: Probability of observing as set of samples with no outliers.	[Percent]
u: Probability of inlier.	[Percent]
$v = 1 - u$: Probability of an outlier.	[Percent]
m: Number of samples in a set.	$[\cdot]$
N: Number of iterations.	$[\cdot]$

RANSAC can also be used to distinguish different planes in a scene using homography. This approach is called layer extraction and is covered in Section 3.5. Next another method robust method is described.

Least Median of Squares Regression

The RANSAC method uses the distance to sort out outliers. This is one way to deal with the problem, that the sum of squared errors such as the algebraic cost function, Appendix D.1, is very sensitive to outliers. Another way is to use another error measurement which is less sensitive, such as the absolute value.

A popular alternative is the Least Median of Squares (LMS) which replaces the sum with the median of the squared residuals[34]. This error measurement only provides a better estimate if the percentage of outliers are below 50 %. Otherwise, the median distance would be to an outlier correspondence. Compared to RANSAC the LMS does not need any predefined thresholds or prior information about the sample distribution (RANSAC parameters θ and u).

3.4 Homography decomposition

Homography is given by Equation 3.4, which shows that the homography is composed by the camera location, intrinsic parameters and surface normal. Homography decomposition, decomposes the homography into these camera locations and surface normal given the intrinsic parameters. The decomposition is obtained through SVD in [35] and [36].

In both methods the eigenvalues from either H or $H^T H$ are used to obtain 8 solutions for R, $\mathbf{t}$ and $\mathbf{n}_\pi$. From the eight solutions, six are disregarded based on different constraints. The correct solution can be determined by considering a third view or second plane. Homography decomposition requires that the calibration matrix for both cameras are known[37], as $K'^{-1}HK = R - \mathbf{t}\mathbf{n}_\pi{}^T$ which is the matrix decomposed. Through out the rest of this section H will be the homography where the influence of the calibration matrix has been removed.

3.4.1 Faugeras decomposition

Faugeras et. al [35] use SVD of H, followed by Equation 3.32 which relate the diagonal matrix Σ to some new variables.

$$\begin{aligned} H &= U\ \Sigma\ V^T && [\cdot] \\ \Sigma &= R_\Sigma + \mathbf{t_\Sigma}\mathbf{n_\Sigma}{}^T && [\cdot] \end{aligned} \tag{3.32}$$

Where:

$$\begin{aligned} &U = [\mathbf{u_1}\ \mathbf{u_2}\ \mathbf{u_3}] : . && [\cdot] \\ &U = diag\,(\lambda_1\ \lambda_2\ \lambda_3) : \text{Diagonal matrix with eigenvalues.} && [\cdot] \\ &V = [\mathbf{v_1}\ \mathbf{v_2}\ \mathbf{v_3}] : . && [\cdot] \end{aligned}$$

The components R_Σ, $\mathbf{t_\Sigma}$, $\mathbf{n_\Sigma}$ can easily be computed since Σ is a diagonal matrix. First T_Σ is found through elimination in the three vector equations produced by Equation 3.32.

The orthogonal matrix constraint on the rotation matrix is used to linearly solve for $\mathbf{n_\Sigma}$, this is described in more detail in [35]. Equation 3.33, 3.34 and 3.35 relates R_Σ, $\mathbf{t_\Sigma}$, $\mathbf{n_\Sigma}$ with the final decomposition elements R, $\mathbf{t}$ and $\mathbf{n}$.

$$R = U\ R_\Sigma\ V^T \quad [\cdot] \tag{3.33}$$

$$\mathbf{t} = U\ \mathbf{t_\Sigma} \quad [\cdot] \tag{3.34}$$

$$\mathbf{n} = V\ \mathbf{n_\Sigma} \quad [\cdot] \tag{3.35}$$

3.4.2 Zhang decomposition

Zhang et. al [36] decompose the homography by computing R, $\mathbf{t}$ and $\mathbf{n}$ from the eigenvalues of $H^T\ H$, see Equation 3.36.

$$H^T H = V\ \Lambda^2\ V^T \qquad [\cdot] \tag{3.36}$$

Where:

$$\Lambda = diag\,(\lambda_1\ \lambda_2\ \lambda_3): \text{ Diagonal matrix with eigenvalues.} \quad [\cdot]$$
$$V = [\mathbf{v_1}\ \mathbf{v_2}\ \mathbf{v_3}]: \ . \quad [\cdot]$$

The eigenvalues Λ are used to calculate four components $\mathbf{v'_1}$, $\mathbf{v'_3}$, ζ_1 and ζ_3, see [36] for description of these equations. First the normalized translation $\mathbf{t_0}$ and the surface normal $\mathbf{n}$ is computed using Equation 3.37 and 3.38. In both equations the numerators and denominators share the same sign, which give 8 equations.

$$\mathbf{t_0} = \pm\frac{\mathbf{v'_1} \pm \mathbf{v'_3}}{\zeta_1 \pm \zeta_3} \qquad [\cdot] \tag{3.37}$$

$$\mathbf{n} = \pm\frac{\zeta_1\mathbf{v'_1} \pm \zeta_3\mathbf{v'_3}}{\zeta_1 \pm \zeta_3} \qquad [\cdot] \tag{3.38}$$

The rotation matrix R can then be computed as $R = H\left(I + \mathbf{t_0}\ \mathbf{n}^T\right)^{-1}$.

The previous sections describe the principle and methods for obtaining and decompose homography. Next the process of determining all the layers present in a scene is covered.

3.5 Layer extraction

The goal of layer extraction is given two or more images to segment all planar surfaces. Each 2D projection of a planar surface is referred to as a layer. In Figure 3.2 the different layers are marked with different colors. Layer extraction is a common step in many computer vision tasks such as geometric scene understanding, scene reconstruction, motion analysis[38, 39], visual navigation[40], and object detection and recognition.

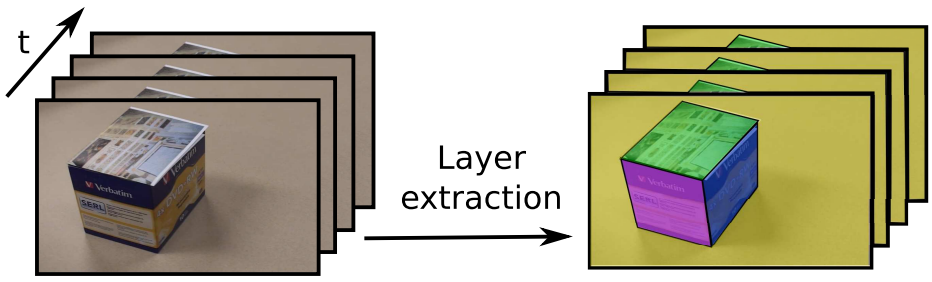


Figure 3.2: *The input frames are shown on the left side and on the right side the layer extractions are visualized as a different color for each layer.*

Layer extraction can be divided into three subproblems, shown below. These three subproblems are very tightly coupled as the motion is used to determine the pixels in the layer, and the pixels in a layer are used to determine the motion.

1. **Segmentation**, which pixels belong to the layer?

2. **Motion**, what is the layer motions?

3. **Number of layers**, how many layers are present in the video?

Throughout this section different approaches to solving these problems are described. The previous work is grouped accordingly to the approach it uses.

3.5.1 EM approach

The Expectation Maximization (EM) algorithm is a natural approach to solve the coupled problem of layer extraction[39, 41, 42]. The EM algorithm is described in detail by Gupta and Chen in [43]. EM is a method for finding maximum likelihood or maximum a posteriori (MAP). MAP estimates parameters in a statistical model, where the model depends on unobserved latent variables.

In the case of layer extraction the video data is formulated as mixture model, i.e. a mixture of Gaussians. Each of the layers are represented as one of the mixture components. The EM algorithm consists of an iterative expectation and maximization step, and then the minimum descriptor length MDL principle[44, 45] is used to estimate the number of layers in the video. The EM algorithm is dependent on a good initialization of the number of layers and motions associated with each layer. Without this, it is possible that the EM algorithm does not converge or converge to a local minimum. An example of the initialization is given by Weiss and Adelson in [41].

3.5.2 Dominant approach

This is an iterative top-down approach of the layer extraction problem. The idea behind the dominant approach is very intuitive. For each iteration the dominant layer is extracted using the dominant motion estimation[46, 47]. The dominant motion can be estimated with a robust estimator[48], i.e. RANSAC, Section 3.3.4. Alternative Black and Anandan propose in [49] a framework for the robust estimation of optical flow. They also proposed a method using affine and piecewise smooth motion fields to robustly estimate multiple motions[47]. The estimated dominant layer is then segmented out and the process is repeated. For each iteration the most dominant layer left in the image is segmented out along with its motion. Ideally the iterations are repeated until all pixels or feature points are assigned a layer. The disadvantage of this approach lies in the assumption of a dominant layer which, in practice, does not always exist.

3.5.3 Grouping approach

The grouping approach was introduced to overcome the disadvantage of the dominant approach. It is a bottom-top compared to the top-down of the dominant approach. The idea is that regions of the same layer will have similar motion and therefore can be grouped together. The approach is usually initialized by calculating the motion (homography) between small blocks i.e. $16 \times 16px$. Xiao and Shah propose the use of a normalized graph cut to cluster the different patches into layers[50, 51, 52]. Altunbasak et al. [53] use k-means clustering to achive the final layers. Compared to the other approaches this does not require any input such as the number of layers. Furthermore, it does not assume anything about the given sequence, such as the dominant approach. However, given the nature of this algorithm it suffers from noisy motion estimates and it ignores the global spatial and temporal constraints[54].

3.6 Summary

Through this section the use and advantages of planar surfaces has been described. It is possible to represent the pixel transform between two images, with different viewpoints, using a perspective transform also called homography. Homography has 8 DoF and is estimated up to a scale. Multiple algorithms exist for the purpose of estimating homography from correspondences between two images. The most used method is normalized DLT, based on point correspondences. However, other types of correspondences can be used such as lines or a combination of lines and points. Normalized DLT estimate the homography through the eigenvalue of a matrix constructed from normalized correspondences. The normalization step is performed to insure independence on the origin and scale of the coordinate system.

More precise algorithms exist, which uses different cost functions, such as geometrical, Reprojection, and Sampson error, and non-linear optimization.

Homography is also used to determine then number of planar surfaces in the scene, also called layer extraction. This is because each planar surface have an unique homography. One method often used is RANSAC which assumes that one layer is more dominant than others. Other methods uses graph-cut, in combination with a grouping strategy, for layer extraction.

Chapter 4
System Specification

In this chapter, the requirement specification for the system is described. First, the problem of geometric scene analysis is described. Next, limitations for the system are presented. This is followed by the problem, which this thesis will focus on. Lastly, the requirements for the system are presented.

4.1 Problem description

A lot of work has been done to obtain accurate 3D reconstructions given multiple images A common problem with these methods is that they have a high computational cost. Single image methods also exist. However, these cannot provide 3D information. Therefore, they commonly use many constraints and some complex framesworks to obtain the geometric scene understanding.

To our knowledge, none or little work has been done on the area between qualitative single image apporaches, such as Blocks world, and reconstruction methods, such as SFM. Therefore, this thesis will focus on this area. It is desired to obtain a simple and quantitative geometric scene understanding using two or more images. It is not desired to use reconstruction nor calibration. Instead, homography will be used to extract all the planar surfaces and then use their interrelationship to obtain a qualitative geometric scene understanding.

The problem description leads to the problem statement, the limitations of the system and the requirements for the system.

4.2 Problem statement

The analysis leads to the following goal. The goal for this thesis is:

To develop a simple system that can obtain a qualitative geometric scene understanding, from planar surfaces observed in two views, without doing reconstruction and calibration.

4.3 Limitations

This section describes the limitations of the system. These are described in the following by a set of assumptions for the input data:

A1 The scene is static.

A2 The scene must primarily consist of planar or near-planar surfaces.

A3 The planar surfaces are not texture-less.

Assumption A1 ensures that the only motion between frames are camera movement. Similar to many other geometric scene understanding approaches, the scene is assumed to consist of planar surfaces (assumption A2). Assumption A3 ensures that feature selection and matching is possible for the given planar surface.

4.4 Requirements

As described earlier, little work has been done within the areas addressed by this thesis. Thus, it is also difficult to set reasonable quantitative requirements for the entire system. The problem has also been addressed by Gupta et al.[13]. Thus, the next chapter will describe the requirements for each module. Chapter 10, "System validation" shows two qualitative results for the entire system.

Part II

System

This part covers system design and implementation. First the system context and structure are addressed. Then each module is described in more detail and validated against the module requirements.

Chapter 5
Program Design

In this chapter the overall structure of the system implemented in this work is described. First the context for which the system should operate in is described with a system context diagram. After this the internal structure of the system is described using a flow chart, including the data interfaces between each module. Last a number of assumptions are introduced followed by the requirements for each module and for the system.

5.1 System Context

In this section the outside interfaces of the system designed in this work are described and illustrated in Figure 5.1 Scene analysis.

According to the requirements, the system is only allowed to use monocular video. Therefore the input of the system is a sequence of images, which follows the assumptions in Section 4.3.

The output of the system is where the left, right, front and top of an object are marked, also the ground and sky are determined.

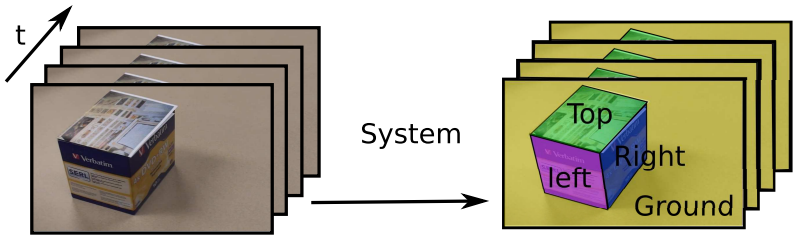


Figure 5.1: *A system context diagram*

5.2 System Structure

In this section the internal structure of the system is described. This is done using a flow chart, where interactions between the modules of the system are illustrated. The chart can be seen in Figure 5.2. Even though the pipeline of this system is similar to the ones seen in other multiple view geometric scene understanding approaches, is each module designed specifically for this system

As already mentioned the input for the system is a sequence of frames, which is pre-processed to improve their quality. After pre-processing, the frames that have been accepted are passed on to the layer extraction module. The frames and extracted layers are passed on to the final stage.

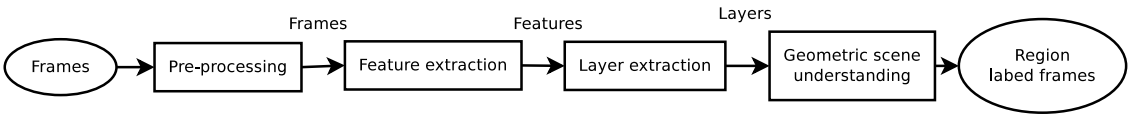


Figure 5.2: *A flow chart of the system*

The scene analysis stage is where the frames and layers are combined into the final output, where layers are assigned a label.

In this work the planar surfaces can be obtained in a simpler manner using homography and layer extraction, see Section 3.5. The final module, "Geometric scene understanding", estimates orientation and relationship between all layers. Given the the orientation and relationship, it is possible to construct a quantitative geometric scene understanding using simple reasoning. The reasoning is also described in the last module, but not fully implemented due to time constraints on this thesis.

Below is a short description of each module and its purpose. The contribution from this work is written with italic.

Pre-processing , the purpose of this module is to select frames, with good quality, which are suitable for feature extraction.
Multiple methods for detection of noise and blur and methods for improving contrast and brightness are thoroughly compared.

Feature extraction , the purpose of this module is to extract features form the image which can be used for layer extraction and geometric scene understanding.
KLT and SIFT features are compared based on their precision. Furthermore, each correspondence are corrected with respect to epipolar geometry between the two images.

Layer extraction , the purpose of this module is to obtain the number of planar surfaces, and the pixels corresponding to each layer.
In this work an extension to the normal RANSAC approach is proposed. The normal and extended RANSAC is compared both on accuracy and time. Furthermore, this module uses a combination of RANSAC and Graph-cut to achieve a dense segmentation.

Geometric scene understanding , the purpose of this module is giving layers, homographies, and features to construct a geometric scene understanding.
This module propose a simple reasoning based on relative depth and orientation between layer.

5.3 Modul Specification

The following requirements are set up for the modules. The requirements written in italic are the ones that the modules and the system will be tested against. The order of the appearance of the requirements and their numbers correspond to the order and numbers that will be used in the module tests.Some of the requirements are followed by an elaboration on how certain details of the requirement in question was determined. If no elaboration is provided then the value is set to a reasonable value.

M.1 Pre-processing must:

1. *Detect low quality frames with recognition rate of* 100%. If frames are not suitable for tracking then the frames are not usable in the remaining system.
2. *Detect medium quality frames with recognition rate of min.* 75%. Medium quality frames can be used, but it is not preferred based on accuracy considerations.

M.2 Feature extraction must:

1. *75% of all matched feature points must have an accuracy of 0.25 px or less.*
   The more precise the matched feature point are the more precise a homography can be estimated. For more information on the actual impact see [55] which covers an error analysis of the impact that feature point location errors has on the homograph.
2. *Maximum 5% incorrect matches.*
   It is not desired to have any incorrect matches between feature points. A match are considered incorrect if the error are more than one pixel.

M.3 Layer extraction must:

1. 90% *of the planar surfaces which have an area over 2500px and contains more than 15 feature points must be detected.*
2. *All detected layers must on average support minimum 90% of the ground truth layer.*
3. *All detected layers must on average support maximum 10% outside the ground truth layer.*
   It is crucial that all the layers of an scene are found and extracted accurately as these are used to generate the scene understanding.

M.4 Geometric scene understanding must:

1. *All the layers must be classified correctly as either a vertical or horizontal layer.*
2. *Minimum 75% of all layer connections must be determined.*
3. *Maximum 5% of determined layer connections may be incorrect.*
4. *0 % percent of left sides may be classified as right sides and vice versa.*
5. *Maximum 25 % percent of left and right sides may be classified as center sides and vice versa.*

Given the module specification, the design and verification of each module is described in the following chapters.

Chapter 6
Pre-processing

In this chapter, the "Pre-processing" module is described. In Figure 6.1, the placement of this module in the system is illustrated. The task of the module is to detect distorted frames with low quality, so that accurate feature matching can be achieved in the next module. Given a distorted frame, accurate feature matching can be impossible. Therefore, it is necessary to minimize the amount of distorted frames. Some of the different distortions that can affect the video quality are covered in Section B.3, Chapter "Camera".

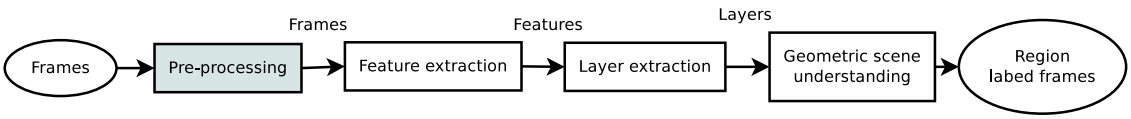


Figure 6.1: *The contexts in which the "Pre-processing" module is placed. The module gets a raw video as input, and outputs video of good quality.*

Figure 6.2 shows a framework that is capable of detecting and removing blur and noise from the frames. Ideally, it is desired to improve the quality of distorted frames in a video. Although, given the purpose of the pre-processing step and the amount of frames in a normal video (30 fps) it is chosen to simply disregard the low quality frames rather than improving them. This decision is also based on the fact that especially deblurring has a very high computational cost[6].

By disregarding the blurry and noisy frames, both the deblurring and denoising steps can be removed. The implemented pre-processing module can be described by the flowchart in Figure 6.3. The pre-processing module with three different steps; detection of blur, detection of noise and contrast and brightness enhancement. Each of these steps are described and evaluated individually in the following sections, 6.1, 6.2 and 6.3. In the end of this chapter, the combined preprocessing module is validated.

As described in the Module Specification, Section 5.3, the requirements for this module are:

M.1 **Pre-processing** must:

1. *Detect low quality frames with a recognition rate of min.* 100%.
2. *Detect medium quality frames with a recognition rate of min.* 75%.

In order to detect distortion in frames, a number of approaches have been described in Section B.3. It is chosen to implement multiple methods for both blur detection and noise detection to find the best candidates. Each of the pre-processing steps and chosen methods are described in the following sections. Lastly, the combined "Pre-processing" module is validated against the Module Specification.

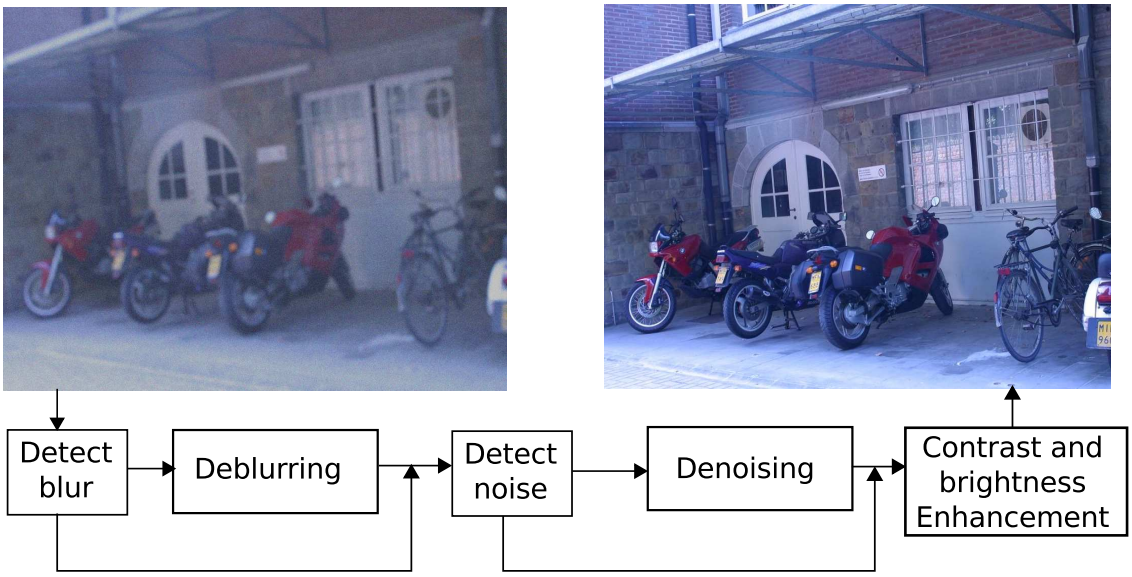


Figure 6.2: *The left image shows distorted image input, and the right image shows the desired output. Below the two images, the different steps of the pre-processing module are illustrated.*

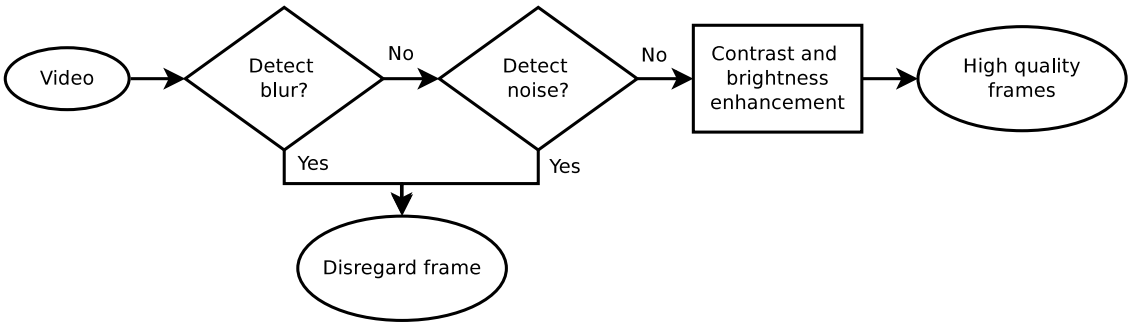


Figure 6.3: *The input is a video, and the output is only frames with high quality. Each frame is tested for blur and noise. If either is present then the frame is disregarded. Finally the contrast and brightness of the frame is improved.*

6.1 Detect blur

The task of the Blur detection step is to determine, whether an image is blurred and to which extent.Three different methods are implemented and evaluated. These methods are chosen based on the principle difference in the approach used.

The implemented approaches are: a simple edge based approach, a Haar wavelet transform and Perceptible Blur. Each of these methods are described in the following sections and verified aginst each other at the end of this section.

6.1.1 Simple edge based

This simple edge based score, builds on the assumption that blurred images will have wider edges and smaller gradients. The edges are determined using a discrete differentiation operator, such as the Sobel operator on the intensity image. The algorithm for the simple edge based blur score is:

1. Convert RGB image to YCbCr. (YCbCR is used because it is commonly used in image compression)
2. Filter Y channel with the Sobel operator.
3. Count Pixels with intensity above zero, Cp, and sum all intensity pixels, Sp.
4. Calculate score as $edge_{score} = {}^{Sp}/_{Cp}$.

6.1.2 Haar Wavelet transform

Tong et al. propose in [56] the use of Haar Wavelet transform (HWT) blur detection. The HWT is a simpe case of the Daubechies wavelet[57], and is also known as **D2**. The HWT decomposes a given image iteratively for different scales, into a hierarchical pyramid-like structure, see Figure 6.4. The input image can be seen in Figure 6.5a. The result of every iteration is halfed in size. The decomposition consists of four different components; LL_i, HL_i, HH_i and LH_i. HH_i is horizontal high-pass/vertical high-pass; HL_i is horizontal high-pass/vertical low-pass; LH_i is horizontal low-pass/vertical high-pass. LL_i is iteratively split. Figure 6.4b shows the level 3 HWT image decomposition, which is used in this method.

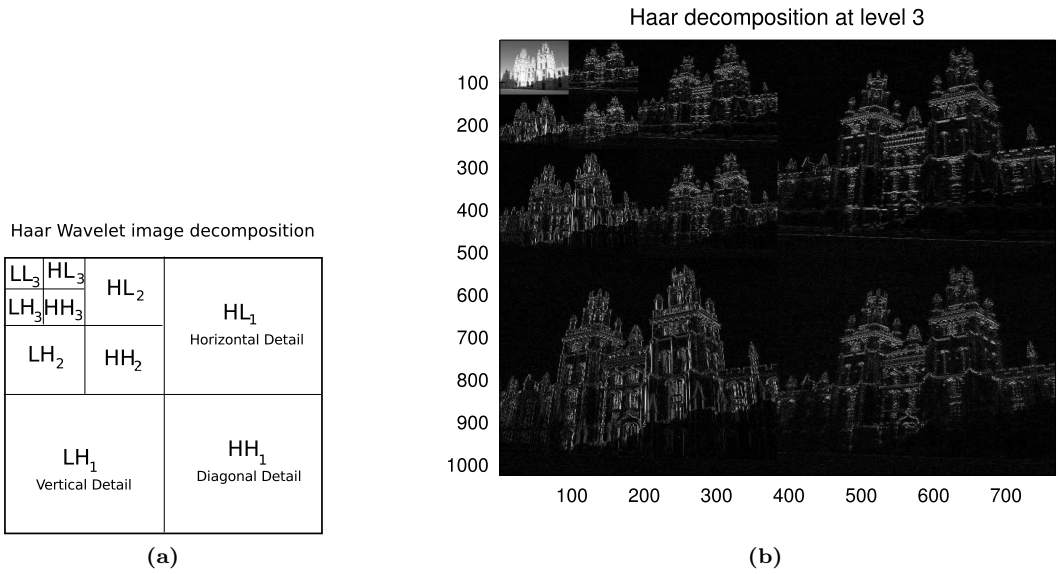


Figure 6.4: *Figure (a) shows the principle of the decomposition and the different components. HH_i is horizontal high-pass/vertical high-pass; HL_i is horizontal high-pass/vertical low-pass; LH_i is horizontal low-pass/vertical high-pass. LL_i is iteratively split as shown[56]. Figure (b) shows a level 3 haar image decomposition of the luminance component of the image.*

Given the HWT decomposition of the image, the edges are detected by the following algorithm:

1. Level 3 HWT decomposition of luminance component.
2. Calculate edge map for each scale $i = (1, 2, 3)$ as $Emap_i = \sqrt{LH_i^2 + HL_i^2 + HH_i^2}$
3. Find local maximum using sliding window, where the window size is given as $2^i \times 2^i$. This will produce equally sized images, $Emax_i$.

The images $Emax_i$ correspond to the intensity of the edge at the given scale, so that a large value corresponds to a sharper edge. An image block (k, l) is defined as an edge block

Edge type	$Emax_1$	$Emax_2$	$Emax_3$
Dirac-Structure	Highest	Middle	Lowest
Astep-Structure	Highest	Middle	Lowest
Gstep-Structure	Lowest	Middle	Highest
Roof-Structure	Lowest	Middle	Highest
	Lowest	Highest	Middle

Table 6.1: *Effect of HWT on different types of edges.*

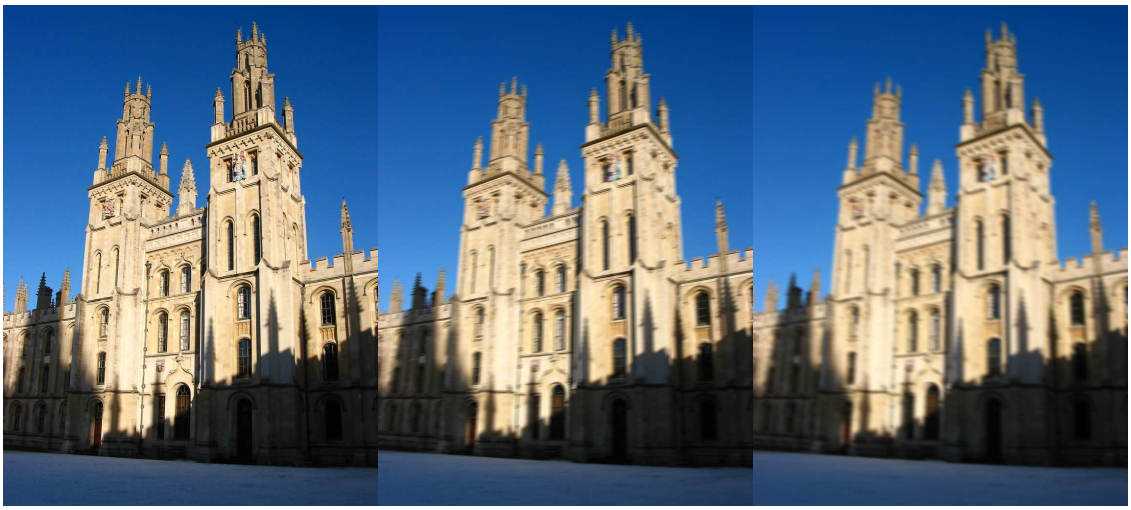
if $max_{i=1,2,3}(Emax_i(k,l)) > Threshold$. The impact that different edges have on $Emax_i$ is examined, and the result is shown in Table 6.1. Given this, different rules are generated so that the different edge types can be distinguished from each other.

The blur score is calculated as Equation 6.1 which is defined as the ratio between the total number of Roof-Structure or Gstep-Structure N_{rg} edges and the sub-part of them where $Emax_1 < threshold$, N_{bgr}.

$$blur\ score = \frac{N_{bgr}}{N_{gr}} \qquad [\cdot] \qquad (6.1)$$

6.1.3 Perceptible Blur

Crete et al. propose a method, which is independent of edge detection [58]. Instead, the method discriminates between different levels of blur perceptible on the same picture, e.g what is the difference between the current image and the image with additional blur. The idea is that an already blurred image will not change significantly when exposed to additional blur, in contrast to a sharp image, see Figure 6.5.



(a) *Original image* **(b)** *Blurred image* **(c)** *Re-blurred image*

Figure 6.5: *Three images that illustates the idea of Perceptible Blur.*

The method uses the intensity variations between the input image and a blurred image to determine, whether the input image was originally blurred. Figure 6.6 shows a flowchart of the

blur score estimation. The calculations are done independently in the vertical and horizontal direction. In the following, the calculation for the horizontal direction is described:

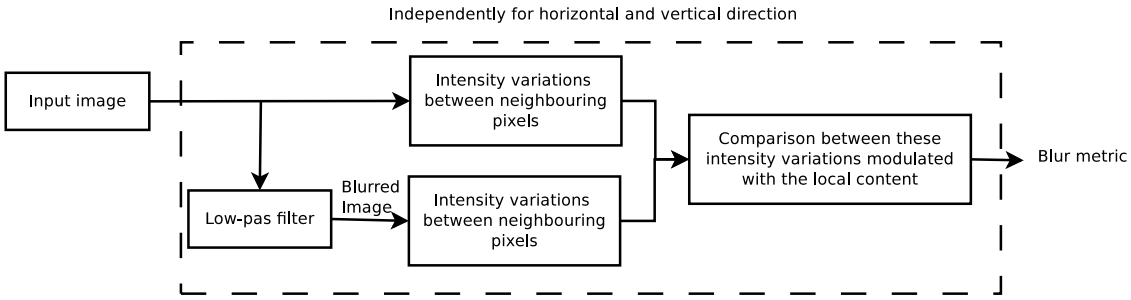


Figure 6.6: *Flowchart for obtaining the Perceptible Blur score[58].*

- **Low-pas filter**, the image is blurred with a uniform nine element kernel $k = {}^{[1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]}/_{9}$.
- **Intensity variations**, are calculated as the absolute difference between the image and the image shifted one pixel horizontally, $I\Delta_{input}$ and $I\Delta_{blur}$.
- **Comparison**, is performed as $blur_{horizontal} = \frac{\sum I\Delta_{input} - \sum max(0,(I\Delta_{input} - I\Delta_{blur}))}{\sum I\Delta_{input}}$.
- **Blur score**, is calculated as $max(blur_{horizontal}, blur_{vertical})$.

6.1.4 Validation

The three methods are evaluated in two different tests. The first test is on a binary ISO 12233 test char, see Figure 6.7a. The second test is on the Oxford data set consisting of images from Flicker[59].

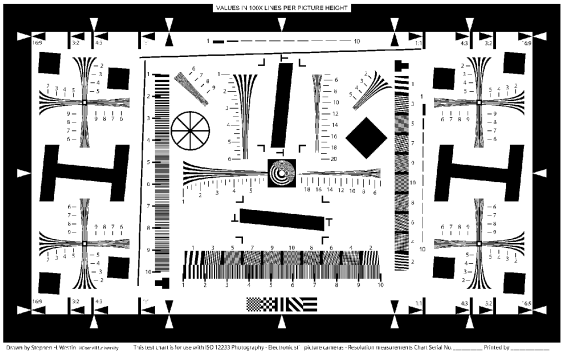
ISO chart test

This test measures the ability of the score to represent different degrees of blur. The ISO test chart is a binary image with size 2000×1249. This image may not represent a natural image but is chose because no or very little blur is present in the image. The image consists of different textures, shapes and numbers, see Figure 6.7a. The ISO test chart is exposed to motion blur. Motion blur can be expressed with a length and angle of direction. For the ISO test 18 random lengths and directions is used. The lengths are limited to an interval of 1 to 25 where the direction can be any angle from 1 to 360 degrees.

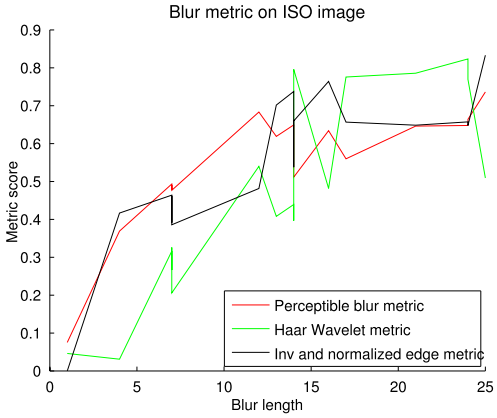
Figure 6.7b shows the calculated scores against the motion blur length. The edge score is normalized and inverted so that it is comparable with the other scores. It can be seen from the figure that all three scores are capable of representing the degree of blur in the interval of [1; 15].

Blur detection on natural images

This test measures the ability of the score to estimate the degree of blur in natural images. The natural images used for this test are from the Oxford flicker dataset[59]. The images are taken with different cameras and in different conditions. Since the images are natural, blur might occur in some of them. To eliminate the most blurry images from the dataset, the three scores are run on the dataset. For each score, a new subset is generated by removing the 1500 images with the



(a) *This test chart is for use with ISO 12233 Photography.[60]*



(b) *Blur scores on blurred ISO test chart. The Chart is blurred with 18 different length and directions of motion blur The lengths are limited to an interval from 1 to 25 where the direction can be any angle between 1 and 360. The edge score is normalized and inverted, so it is comparable with the other scores.*

Figure 6.7: *Test using blur on the ISO test chart.*

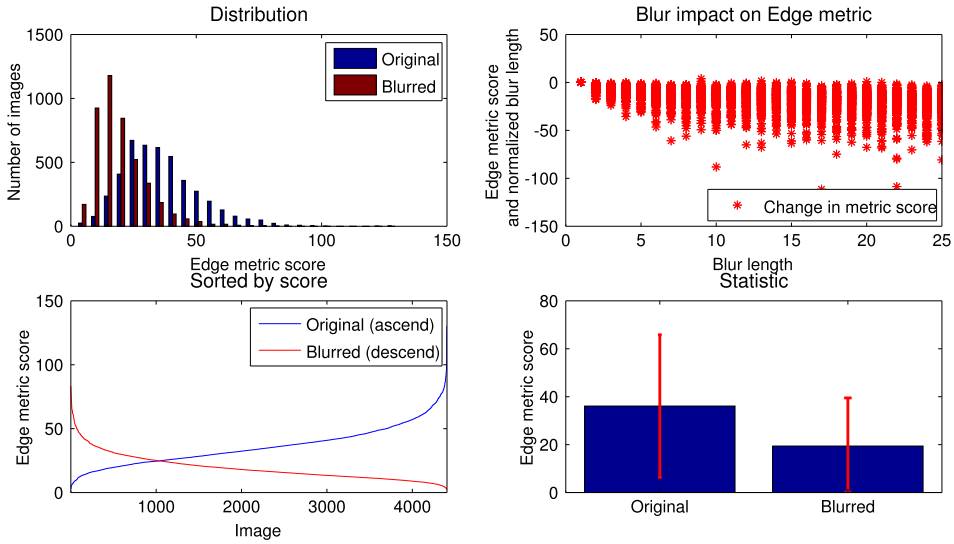


Figure 6.8: *The top left image shows the distribution of the estimated Haar Wavelet blur score for both the input and blurred image. The top right image shows the difference in blur scores (input-blurred) against the blur length. The bottom left image shows a sorted graph of the Edge blur score. The bottom right image shows the mean value and the 95% confidence interval of the Haar Wavelet blur score for both the input and the blurred image.*

highest scores. A common dataset for all three is then generated from all the unique images in all three subsets. The final dataset contains 4406 images, which is 657 less than the original dataset.

Given the new dataset, the test is performed as described below:

- Calculate scores on the input image.

- Blur input image with random motion blur (same as the ISO chart test).
- Calculate scores on the blurred image.

Figure 6.8, 6.9 and 6.10 show the result for the three methods, i.e. the simple edge based, the Haar Wavelet and the Perceptible Blur scores. The figures show four different plots. The top left image shows the distribution of the estimated blur scores for both the input and blurred image. The top right image shows the difference in blur scores (input-blurred) against the blur length. The bottom left image shows a sorted graph of the blur score. The bottom right image shows the mean value and the 95% confidence interval both input and blurred image.

From the distribution of scores and the statistics, it can clearly be seen that the Haar wavelet and the Perceptible blur scores provide a significantly better result than the simple edge based score. Given the simple edge based score, it is not possible to distinguish between blurred and non-blurred images. The edge score also has another disadvantage as it is not normalized.

From the top right figures, it can be seen that the Perceptible Blur scores consistently provide a higher score on the burred images. However, Haar Wavelet is capable of providing a higher score for the blurred image, which can make it easier to detect them.

Neither the Haar Wavelet or the Perceptible Blur scores provide a significantly better result than the other. It is therefore chosen to combine the two methods by multiplying their scores, see Figure 6.11. This is motivated by the fact that the two methods calculate the blur metric in two very different ways. Thus, it is assumed that when one method fails the other will succeed. A threshold for the score is chosen as $\tau_{blur} = 0.25$, which is based on the distribution, statistics and the intersection point in then bottom left graph.

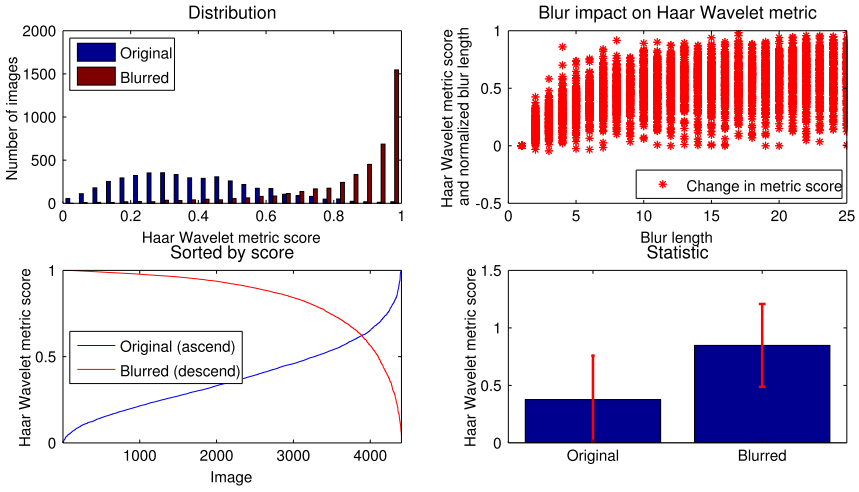


Figure 6.9: *Top left shows the distribution of the estimated Haar Wavelet blur score for both the input and the blurred image; top right shows the difference in blur scores (input-blurred) against the blur length; Bottom left shows a sorted graph of the Haar Wavelet blur score; Bottom right shows the mean value and the 95% confidence interval of the Haar Wavelet blur score for both input and blurred image.*

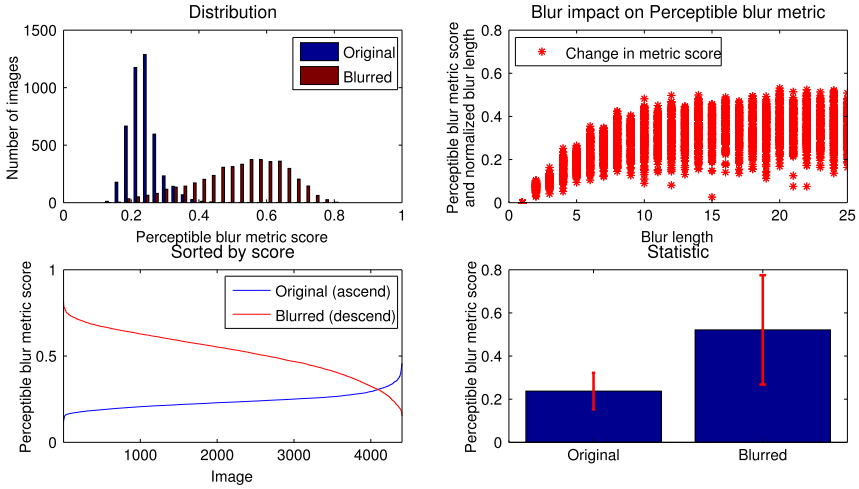


Figure 6.10: *Top left shows the distribution of the estimated Perceptible blur score for both input and blurred image; top right shows the difference in blur scores (input-blurred) against the blur length; Bottom left shows a sorted graph of the Perceptible blur score; Bottom right shows the mean value and the 95% confidence interval of the Perceptible blur score for both input and blurred image.*

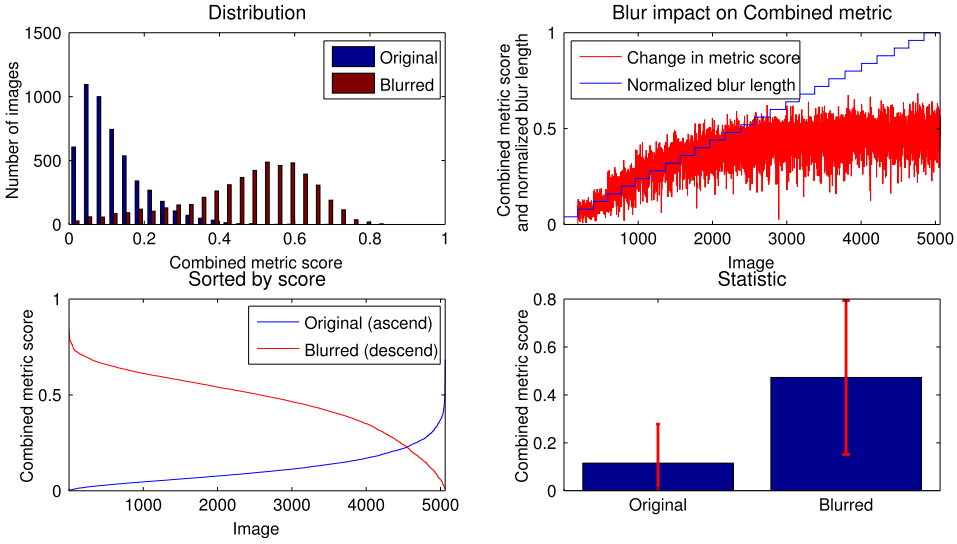


Figure 6.11: *Top left shows the distribution of the estimated combined blur score for both input and blurred image; top right shows the difference in blur scores (input-blurred) against the blur length; Bottom left shows a sorted graph of the Perceptible blur score; Bottom right shows the mean value and the 95% confidence interval of the combined blur score for both input and blurred image.*

6.2 Detect noise

In this section, three simple spatial methods are described and evaluated. Although the methods are spatial, temporal information is also given.

6.2.1 Cascaded horizontal-vertical shifted differences (CHVSD)

This approach finds the difference between the original and a shifted image. This process is cascaded for both the horizontal and the vertical direction[61]. The standard deviation of noise, σ, is then found by taking the median of the cascaded differences, and dividing it by 0.6745. The constant 0.6745 makes the estimate unbiased for the normal distribution[62].

6.2.2 Wavelet domain estimation

This approach is based on the Daubechies wavelet[57]. A wavelet is a wave-like oscillation, which can be used to extract information from signals, e.g. audio and images. Donoho et al. propose to use the Daubechies wavelet to estimate the standard deviation of the noise in an image [63]. This is done by a cascading convolution of the noisy image with a 6th order Daubechies wavelet. σ is then found by taking the median of the cascaded differences and dividing it with 0.6745.

6.2.3 Immerkaer

This approach was proposed by Immerkaer in [64]. The noisy image $X_{n\times m}$ is convolved with the kernel K, see Equation 6.2.

$$\sigma = \sqrt{\frac{\pi}{2}} \frac{1}{6\ n\ m} \sum_{i,j=1}^{n,m} |(X * K)_{i,j}| \qquad [\cdot] \tag{6.2}$$

$$K = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad [\cdot]$$

Where:

$X_{n\times m}$: $n \times m$ input image. $[\cdot]$
K: Immerkaer Kernel. $[\cdot]$

6.2.4 Validation

The three methods for estimation noise in an image are compared in this section. The test is performed on 5063 different natural images from the Oxford flicker dataset[59]. The test is described below:

1. Estimate σ of noise in original image, σ_o, pixel values are between 0 and 1.
2. Add random Gaussian noise,σ_t with between 0 and 0.0585 to the original image.
3. Estimate σ of noise in the new image, σ_n
4. Calculate error as $\epsilon = abs(\sigma_n - (\sigma_o + \sigma_t))$

Figure 6.12 and 6.13 shows two sub-images from an original image and the corresponding noisy image. It can clearly be seen that there are added noise to Figure 6.12b and 6.13b.

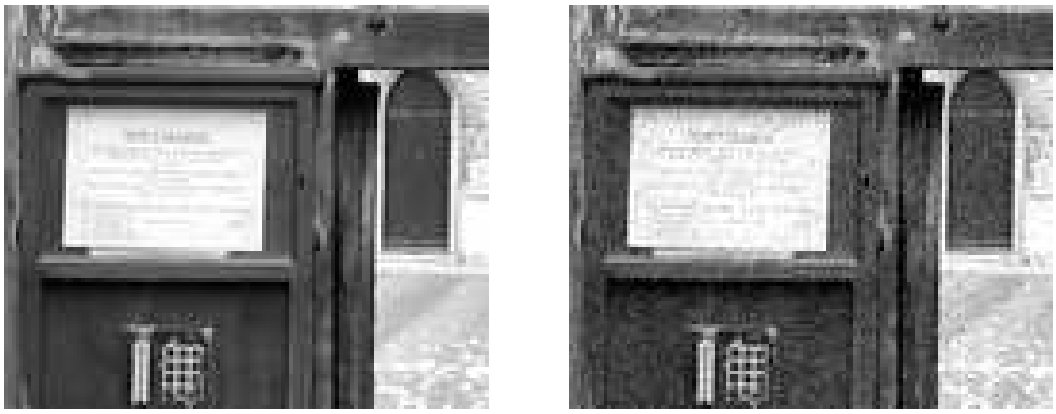
Figure 6.14a shows a histogram of the absolute error, and Figure 6.14b and Table 6.2 shows the mean and 95% confidence interval. It can be seen that CHVSD and Wavelet perform almost

similar with a mean around 0.005, where Immerkaer performs worst with mean 0.0078. Wavelet method chosen for noise detection base on the statistic.

Figure 6.15 shows the sorted noise estimates for Wavelets against the ground truth. The Figure clearly shows that Wavelet averagely underestimate the present noise. Multiplying the the estimate by a constant $k = 1.2$ improves the noise estimate by 50%, such that the error has mean 0.0032 and 95% confidence interval $[0; 0.0087]$. It shall be noted that multiplication by k also improves the other two methods.

Method	Mean	95% confidence interval
CHVSD	0.0052	[0; 0.013]
Wavelet	0.0048	[0; 0.0124]
Immerkaer	0.0078	[0; 0.0171]

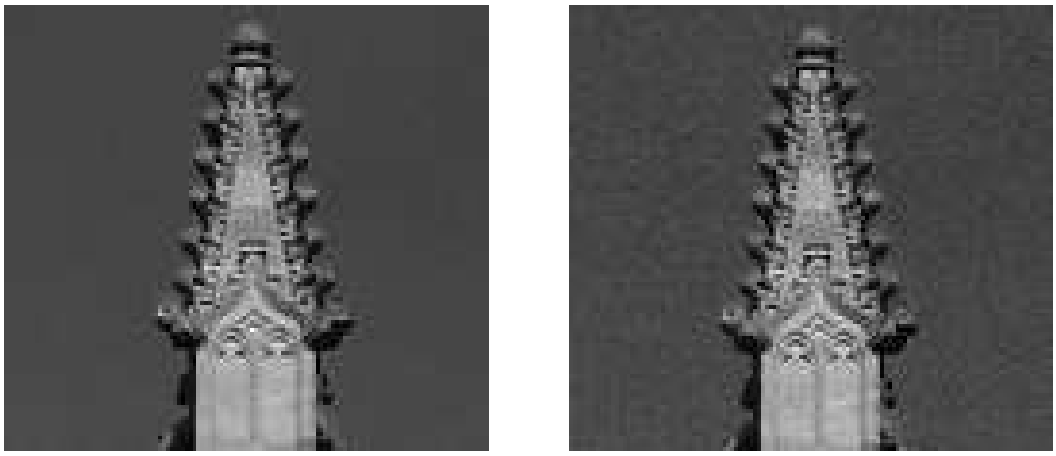
Table 6.2: *Error statistic for the error for absolute noise σ estimates for the three methods.*



(a) *Original image*

(b) *Image with applied noise $\sigma = 0.04$. Estimated noise for the three methods are* 0.03, 0.032, 0.029.

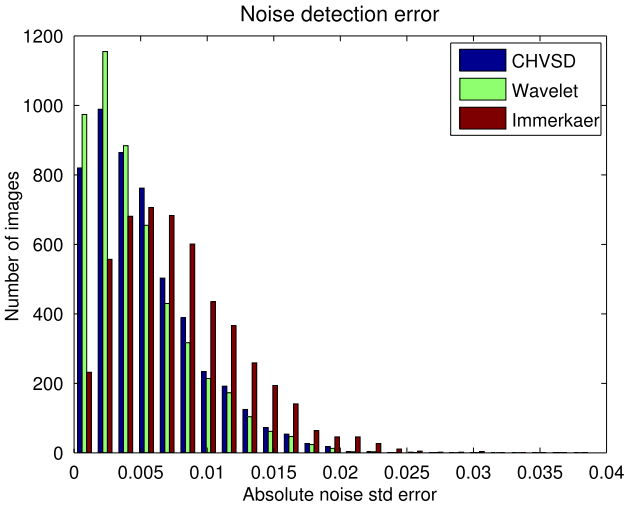
Figure 6.12: *Original and noisy image*



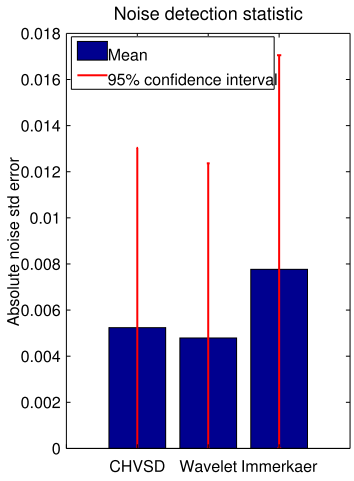
(a) *Original image.*

(b) *Image with applied noise $\sigma = 0.02$. Estimated noise for the three methods are* 0.016, 0.016, 0.011.

Figure 6.13: *Original and noisy image*



(a) *Histogram of the absolute σ error for the three methods.*



(b) *Absolute σ error statistic for the three methods. The shown statistic are mean value and 95% confidence interval*

Figure 6.14: *Distribution and error statistics for three noise detection methods*

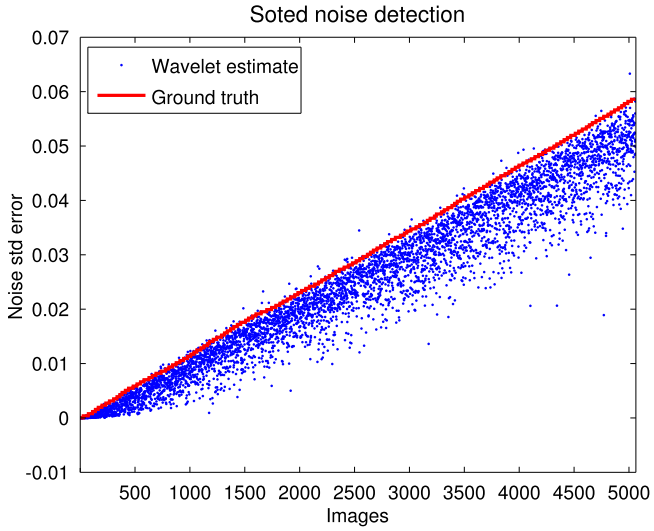


Figure 6.15: *Graph of the estimated σ and the ground truth for the Wavelet method. All data is sorted with respect to the ground truth σ.*

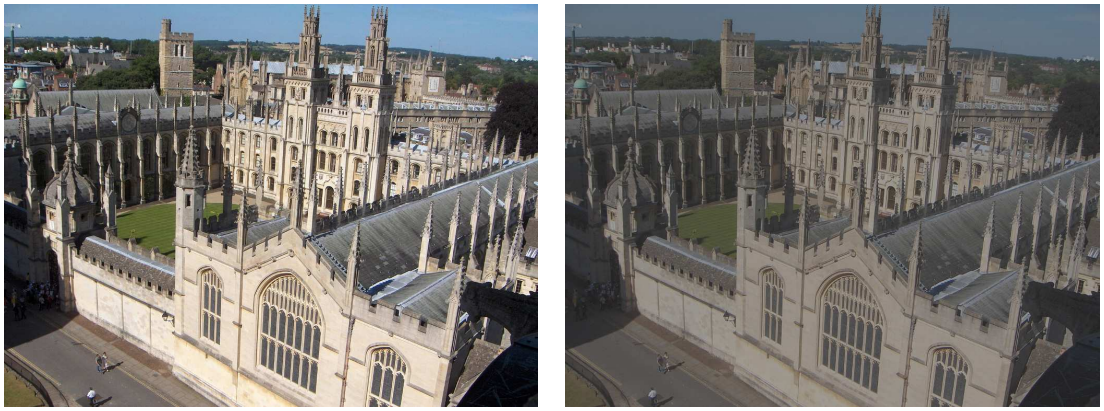
6.3 Contrast and brightness enhancement

The contrast and brightness of an image can be modeled as a multiplication and addition, Equation 6.3. α and β are commonly called gain and bias and they control the contrast and brightness of the image. Both α and β can be constant or depend on $\mathbf{x}$. Figure 6.16 shows the original image and a low brightness and contrast version of it.

$$g(\mathbf{x}) = \alpha f(\mathbf{x}) + \beta \qquad [\cdot] \tag{6.3}$$

Where

$\mathbf{x}$: Image coordinate x, y	[px]
$f(\cdot)$: Input image.	$[\cdot]$
$g(\cdot)$: Output image.	$[\cdot]$
α: Gain	$[\cdot]$
β: Bias	$[\cdot]$



(a) *Original image.* **(b)** *Low contrast and brightness image.*

Figure 6.16: *Original and low brightness and contrast image.*

There are different approaches to contrast and brightness enhancement. One approach is to stretch the histogram of the image such that the darkest and brightest pixels are mapped to pure black (0) and white (255), see Figure 6.17. Another approach is to calculate the average pixel value and transform the image such that it approaches pure gray (128).

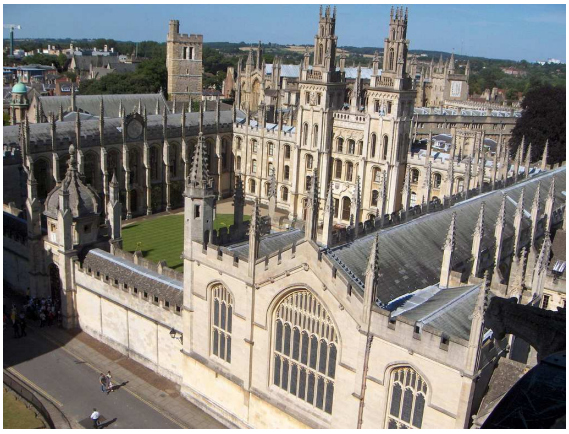


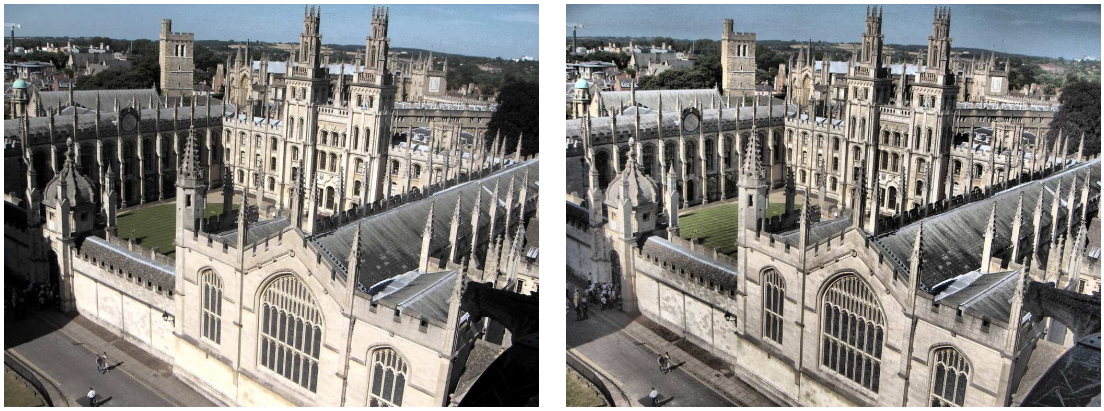
Figure 6.17: *Contrast and brightness enhancement using image stretching on RGB image.*

A third and popular method is histogram equalization. Histogram equalization is used in image comparison processes (because it is effective in detail enhancement) and in the correction of non-linear effects[65, 66]. Histogram equalization is usually performed on the intensity component of

an HSI image (Hue, Saturation, and Illumination). Equalizing the illumination component will improve contrast and brightness of the image, see Figure 6.18a. All though histogram equalization improves the contrast it will also, especially in homogeneous areas, amplify the noise which might be present in the image.

To prevent amplification of noise in homogeneous areas an adaptive histogram equalization can be used. The adaptive histogram equalization is performed in small blocks and a contrast factor that prevents over-saturation of the image, see Figure 6.18b.

Histogram equalization uses the cumulative distribution function which is also the image's accumulated normalized histogram. It is desired to create a transformation of the form $y = T(x)$ to produce a new image y, such that its CDF will be linearized across the value range.



(a) *Normal histogram equalization.*

(b) *Adaptive histogram equalization, with block size* 8×8 *and contrast factor = 0.02*

Figure 6.18: *Contrast and brightness enhancement through normal and adaptive histogram equalization of illumination component of HSI. Pixel values are in the interval 0 to 1.*

It can be seen from Figure 6.18, 6.17 and 6.19 that stretching provides a good result without amplifing noise. It is therefore chosen to use image stretching as contrast and brightness enhancement method.

(a) *Image streching.*

(b) *Histogram equalization.*

(c) *Adaptive histogram equalization.*

Figure 6.19: *Three sub images from the top right corner of Figure 6.17, 6.18a and 6.18b. Each sub image is* $100 \times 100px$ *and show a homogeneous area.*

Label	Estimated	Ground truth Low (146)	Medium (114)	High (85)
Low	66.09% (228)	64.04% (146)	33.77% (77)	2.19% (5)
High	33.91% (117)	0% (0)	31.62% (37)	68.38% (80)

Table 6.3: *Result of "Pre-processing" module. The number in parentheses are the number of frames.*

6.4 Module validation

The "Pre-processing" module are validated on its ability to detect low quality images in a video sequence. The ideal test would include a dataset with ground truth values of blur and noise. It has not been possible to locate such a dataset, thus it have been chosen to manually generate a dataset. The dataset consist of one video, with a 345 frames. The video is recorded by a hand held camera. Thus, it contains sequences with motion blur due to shake. The ground truth have been generated by three persons, which have classified frames as high, medium or low quality. Figure show an example of a high and low quality frame, and Figure 6.21 shows location of the different frames. The video contains 85 high quality frames, 114 medium and 146 low.

(a) *High quality frame.* **(b)** *Low quality frame.).*

Figure 6.20: *Example of high and low quality frames in generated dataset.*

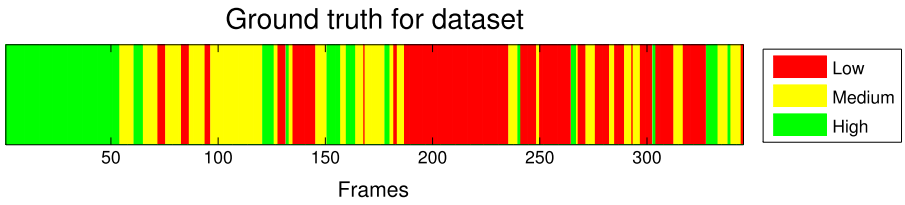


Figure 6.21: *Ground truth classification video into high, medium, and low. High quality is shown as green, medium as yellow and low as red. The video contains 85 high quality frames, 114 medium and 146 low.*

Figure 6.22 shows the labeling of frames determined by the "Pre-processing" module. A total of 117 frames have been determined as high quality frames. Table 6.3 shows the amount of frames from each group that are labeled as high or low quality.

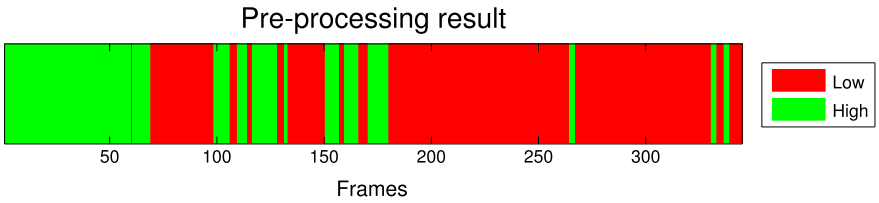


Figure 6.22: *Result of "Pre-processing" module. Location of high low quality frames is shown with green. The video contains 117 high quality frames, 228 low.*

6.5 Module conclusion

The "Pre-processing" module, uses a combination of Haar Wavelet and perceptible blur for detection of blur. It further more uses a wavelet method for detection of noise. Finally image stretching is performed to improve contrast and brightness.

The module validation shows that the "Pre-processing" module is able to detect 100% of the low quality frames, and 67.54% of the medium quality frames. The module is also able to correctly detect 94.11% of the high quality frames.

It can therefore be concluded that the "Feature extraction" module satisfy the requirement about 100% detection rate for low quality frames. The other requirement about 75% detection rate of medium quality frames are not satisfied, with the current thresholds of 0.25 for blur an 1/256 for noise. Reducing the blur threshold to 0.2 will improve the detection of medium quality frames to 93.86%. Decreasing the threshold will also result in a decreases of correctly detect high quality frames to 88.24%. Given the small increase of (5.87%) false positive compared to the high decrease of (26.32%) false-negative, it is chosen to use 0.2 as threshold for blur detection.

Given the pre-processed video sequence the next step is to determine correspondences between two frames.

Chapter 7
Feature extraction

In this chapter, the "Feature extraction" module is described. In Figure 7.1, the placement of this module in the system is illustrated. The task of the module is to detect features that can be used in the "Layer extraction" module. Given a sequence of frames, feature points must be defined and matched between subsequent frames. In Section 3.3, "Homography estimation", different features that can be used for homography estimation are described. It is chosen to use points as these are simple features to find and match.

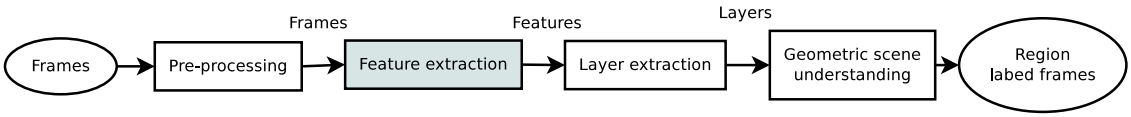


Figure 7.1: *The contexts in which the "Feature extraction" module is placed. The module gets a video as input, and outputs feature point correspondences.*

Figure 7.2 shows a flow chart of this module. This module is capable of robustly detecting accurate feature point correspondences. First, feature points are found and matched between adjacent frames. Next, mismatches are detected and removed. Finally, all feature point locations are improved to achieve higher accuracy.

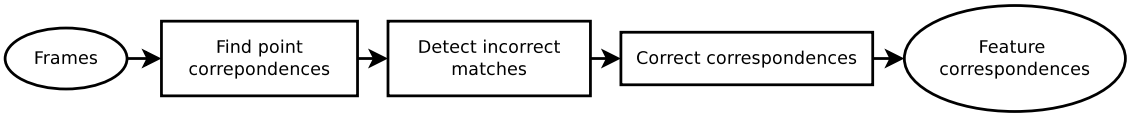


Figure 7.2: *Flow chart of the "feature extraction" module. First, feature point correspondences are found, and then any mismatches are removed. Finally, the point locations are improved.*

As described in the Module Specification, Section 5.3, the requirements for this module are:

[M.2] **Feature extraction**:

1. *75% of all matched feature points must have an accuracy of 0.25 px or less.*
2. *Maximum 5% incorrect matches.*

7.1 Find feature correspondence

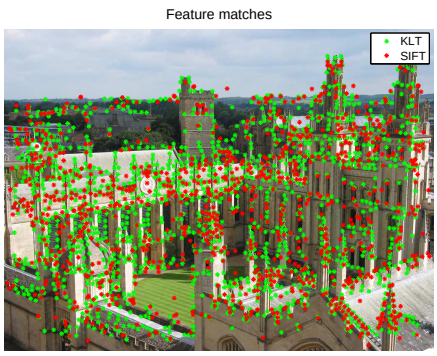
Different methods for feature point estimation and matching have been proposed over the years. Two very popular methods are the Kande Lucast Thomasi (KLT) feature point tracker[67] and

the Scale Invariant Feature Transform (SIFT)[68]. A detalied description of both methods can be fund in Appendix G "KLT Feature Point Tracker" and F "SIFT".

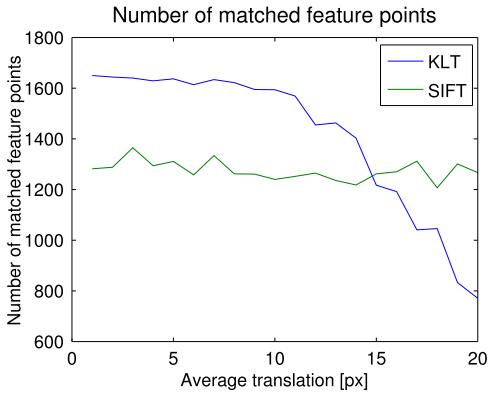
In this section, the two methods are compared to investigate their accuracy. The test is performed using one image, where feature points are selected. The features are then matched against a translated version of the image. The translation in the x and y directions are equal and is determined by a normal distributed random value with mean 1,2,...,20 and variance 1. 20 images are generated for for each mean value, e.g for 20 image with mean 1, 20 image with mean 2. It is chosen to use translation under the assumption that the change between adjacent frames is small.

Both KLT and SIFT depend on different variables. A detailed description and an analysis of each parameter can be found in Appendix G and F. KLT only depends on the minimum eigenvalue, which is set to 750. SIFT depends on several variables; the edge and contrast threshold are set to 5 and 0.02, respectively and for matching, a threshold of 0.75 is chosen. The implementations of KLT and SIFT used in this thesis are done by [69] and [70], and a Matlab interface has been written for the KLT code.

Figure 7.3a shows the matched KLT and SIFT features, exposed to a 10.26 px translation in x,y. It can be seen that both are able to detect features uniformly over the interesting regions of the scene. Figure 7.3b shows the average amount of features matched for different lengths of translation. It can be seen that SIFT remains constant, where KLT decreases as the translation increases.



(a) *Matched KLT and SIFT features on the original image, here with a 10.26px translation.*



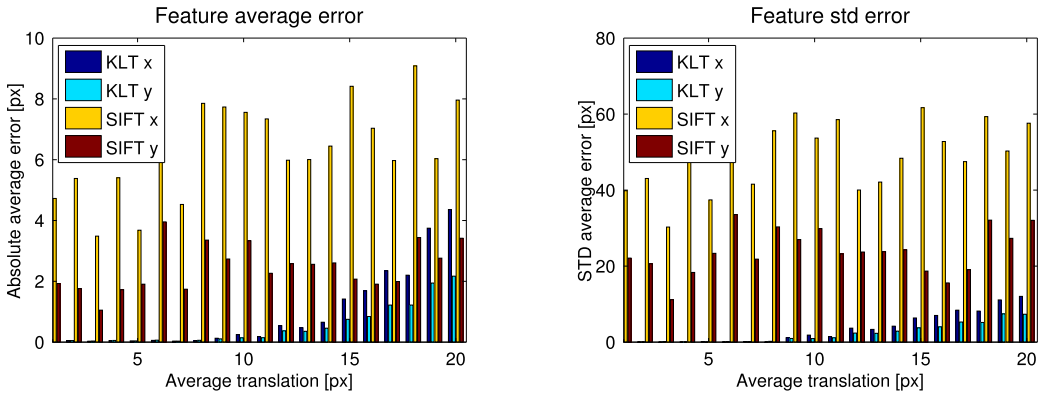
(b) *Average number of features compared to the average translation.*

Figure 7.3: *Location and average amount of KLT and SIFT features.*

Furthermore, it can be seen from Figure 7.4 that KLT performs significantly better for low translations than SIFT. The average error of SIFT remains constant over the entire translation range, where KLT has a very low average error for translations below 10 px. Similar characteristics can be observed from the standard deviation (SD) values.

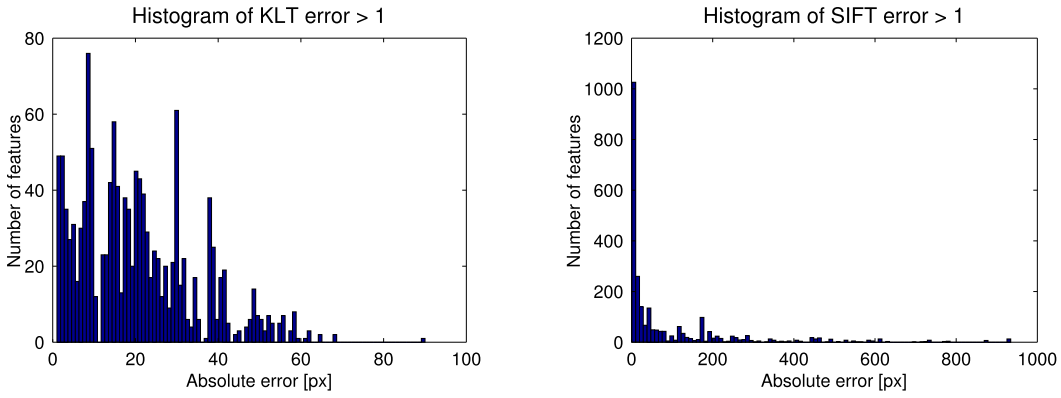
From the SD values it can also be seen that SIFT has a large deviation for all translations. This behavior indicates that SIFT makes some incorrect matches, which results in high errors. This is also supported by Figure 7.5 which shows two histograms; one for the KLT errors and one for SIFT errors (only errors ¿ 1 are shown). For SIFT it can be seen that there are some very large errors, which can only be caused by incorrect matches. The KLT is more robust to large mismatches as it searched in a close neighborhood of the original point, where SIFT searches through all feature points.

Given the occurrence of mismatches, it is desired to detect and remove incorrect matches.



(a) *Average absolute error* **(b)** *SD of absolute error*

Figure 7.4: *The two figures show the average and SD of the absolute matching errors using 20 different translations on the image.*



(a) *KLT* **(b)** *SIFT*

Figure 7.5: *The two figures shows the histogram of every absolute matching errors in the 20 different translations used. Only the errors that are larger than one pixel are shown.*

7.2 Detect incorrect matches

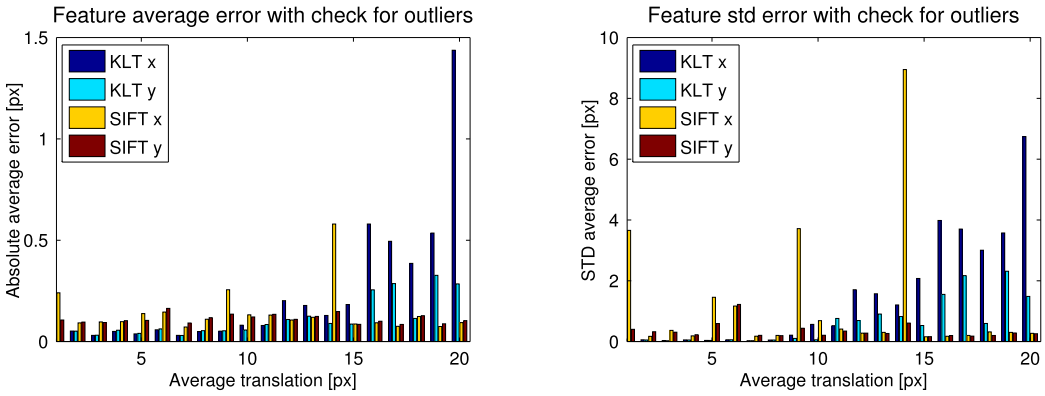
The goal for this step is to minimize the number of outliers to improve the matching performance. One simple method to find outliers is to use the epipolar error related with the epipolar geometry, i.e. the fundamental matrix, see Appendix E "Fundamental matrix". The fundamental matrix is robustly estimaed using RANSAC. Given $\widetilde{\mathbf{x}'}^T F\widetilde{\mathbf{x}} = 0$ derived from Equation E.1, each point correspondence can be determined as an inlier or an outlier.

In practice, it is unlikely that the points correspond precisely with the epipolar geometry. Therefore, a small threshold is used to determine outliers, see Equation 7.1.

$$|\widetilde{\mathbf{x}'}^T F\widetilde{\mathbf{x}}| > \tau \rightarrow outlier \tag{7.1}$$

Given the fact that outliers exist, RANSAC is used to robustly estimate the fundamental matrix with the eight point algorithm introduced by Christopher Longuet-Higgins in 1981[71]. The principle of the eight point algorithm is similar to the estimation of homography, see [15, 72] for more details.

The mismatches of the previous step are removed and the new results can be seen in Figure 7.6. It can clearly be seen that especially the performance of SIFT has improved considerably. The average absolute error has increased with 74.38% and with 77.77% for the KLT x and y coordinate respectively. For SIFT the increase is even higher 97.79% and 95.43%. Also, the SD has improved considerably, where there only are a few very high deviations for the SIFT feature. For KLT, it can still bee seen that larger translations result in a more inaccurate matching, see Figure 7.7 which shows the number of outliers compared to average translation length.



(a) *Average absolute error* **(b)** *SD of absolute error*

Figure 7.6: *The two figures shows the average and SD of the absolute matching errors using 20 different translation lenghts on the image. The matching result has been improved by removing outliers.*

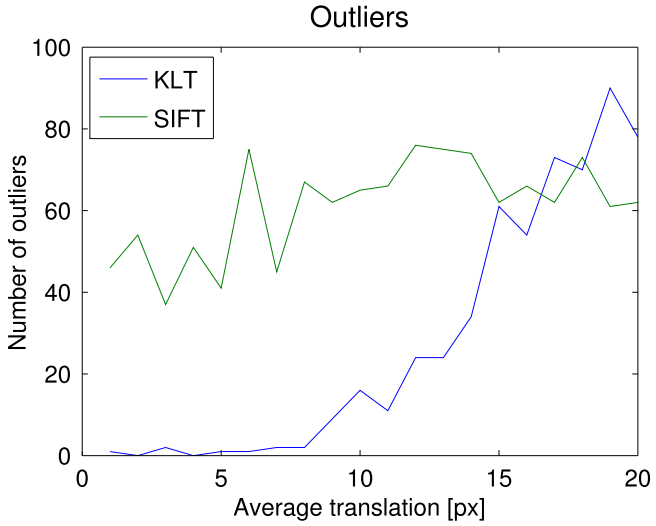


Figure 7.7: *Number of outliers determined using the fundamental matrix and a threshold equal to 0.01 for both KLT and SIFT.*

Given the correct matches, each point correspondences can be corrected with respect to the epipolar geometry constraint, so that the epipolar error is minimized.

7.3 Correction of point correspondence

As described in the previous section, the epipolar constraint $\widetilde{\mathbf{x}'}^T F\widetilde{\mathbf{x}} = 0$ can be used to determine inliers and outliers, but it can also be used to correct feature point locations. It is desired to find the corrected points $\widetilde{\hat{\mathbf{x}}}$ and $\widetilde{\hat{\mathbf{x}}'}$ so Equation 7.2 are minimized. $d(\cdot,\cdot)$ is the Euclidean distance between the points.

$$C(x,x') = d(\widetilde{\mathbf{x}},\widetilde{\hat{\mathbf{x}}})^2 + d(\widetilde{\mathbf{x}'},\widetilde{\hat{\mathbf{x}}'})^2 \text{ subject to } \widetilde{\hat{\mathbf{x}}}^T F\widetilde{\hat{\mathbf{x}}'} = 0 \qquad [\text{px}] \qquad (7.2)$$

Equation 7.2 can be minimized using a numerical minimization method such as Levenberg-Marquardt[73]. However, if the error distribution can be assumed Gaussian, then $\hat{x}$ and $\hat{x}'$ are Maximum Likelihood Estimates (MLE) of correct correspondence and the optimal solution to Equation 7.2 can be found explicitly[15].

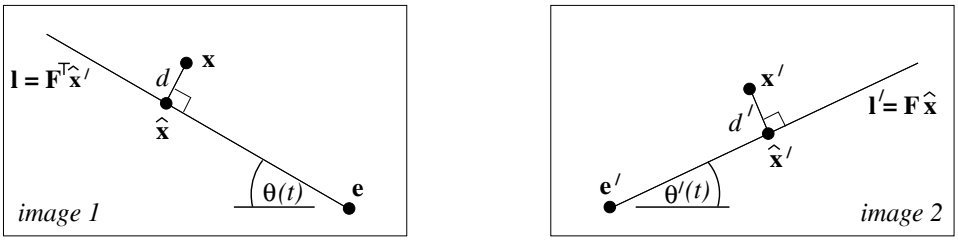


Figure 7.8: *The points $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{x}}'$ lie on a pair of corresponding epipolar lines. The optimal $\widetilde{\hat{\mathbf{x}}}$ and $\widetilde{\hat{\mathbf{x}}}'$ is founds as the intersection points between the epipolar lines and the ortogonal distance to the measured points $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{x}}'$.[15].*

The minimization of Equation 7.2 can be reformulated as a search for one parameter. The following discussion relates to Figure 7.8.

From the epipolar constraint, it is known that the points must lie on the corresponding epipolar lines $\mathbf{l}$ and $\mathbf{l}'$ in the two images. Thus, the optimal solution $\widetilde{\hat{\mathbf{x}}}$ and $\widetilde{\hat{\mathbf{x}}}'$ must lie on the epipolar lines $\mathbf{l}$ and $\mathbf{l}'$. All point correspondences on $\mathbf{l}$ and $\mathbf{l}'$ will satisfy the epipolar constraint. The pair of points that minimizes the squared distance sum of Equation 7.2 are the points on $\mathbf{l}$ and $\mathbf{l}'$ that lie closest to $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{x}}'$. Consequently, the distance $d(\widetilde{\mathbf{x}},\widetilde{\hat{\mathbf{x}}})$ can be reformulated as $d(\widetilde{\mathbf{x}},\mathbf{l})$, which represents the perpendicular distance from the point $\widetilde{\mathbf{x}}$ to the line $\mathbf{l}$. A similar expression holds for $d(\widetilde{\mathbf{x}'},\widetilde{\hat{\mathbf{x}}'})$, so Equation 7.3 is obtained.

$$d(\widetilde{\mathbf{x}},\mathbf{l})^2 + d(\widetilde{\mathbf{x}'},\mathbf{l}')^2 \qquad \left[\text{px}^2\right] \qquad (7.3)$$

$\mathbf{l}$ and $\mathbf{l}'$ range over all choices of corresponding epipolar lines in the two images. This can be reformulated by parametrizing the pencil of epipolar lines in the first image by t. Thus, an epipolar line is written as $\mathbf{l}(t)$. The corresponding epipolar line $\mathbf{l}'$ is found using the fundamental matrix. Thus, the minimization is now reduced to the search of t that minimizes Equation 7.4.

$$\min_t C = d(\widetilde{\mathbf{x}},\mathbf{l}(t))^2 + d(\widetilde{\mathbf{x}'},\mathbf{l}'(t))^2 \qquad \left[\text{px}^2\right] \qquad (7.4)$$

The problem can be simplified by applying a rigid transformation to the fundamental matrix. First, the rigid transformation translates $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{x}'}$ to the origin, see Equation 7.5.

[1]The origin in homogeneous coordinates is $(0,0,1)^T$.

$$F_T = T'^{-T}\ F\ T^{-1} = \begin{bmatrix} 1 & 0 & -x' \\ & 1 & -y' \\ & & 1 \end{bmatrix}^{-T} F \begin{bmatrix} 1 & 0 & -x \\ & 1 & -y \\ & & 1 \end{bmatrix}^{-1} \qquad [\cdot] \qquad (7.5)$$

Next, the epipoles are transformed to $(1, 0, f)^T$ and $(1, 0, f')$ respectively, see Equation 7.6. The transformation is obtained using two rotation matrices, determined by the scaled epipoles $\mathbf{e} = (e_1, e_2, e_3)^T$ and $\mathbf{e}' = (e'_1, e'_2, e'_3)^T$ of F_T. The scale is defined so that $e_1^2 + e_2^2 = 1$ is satisfied for $\mathbf{e}$ and similar for $\mathbf{e}'$.

$$F_R = R'\ F\ R^T = \begin{bmatrix} e'_1 & e'_2 & \\ -e'_2 & e'_1 & \\ & & 1 \end{bmatrix}^T F_T \begin{bmatrix} e_1 & e_2 & \\ -e_2 & e_1 & \\ & & 1 \end{bmatrix} \qquad [\cdot] \qquad (7.6)$$

Multiplying the epipoles with their respective rotation matrix yields $R\mathbf{e} = (1, 0, e_3)^T$ and $R'\mathbf{e}' = (1, 0, e'_3)^T$.

Applying these two rigid transforms will not affect the distance and therefore not change the minimization problem. However, $F(1, 0, f)^T = (1, 0, f')F = \mathbf{0}$ will be valid and the fundamental matrix will therefore take a special form, see Equation 7.7.

$$F_R = \begin{bmatrix} ff' & -f'c & -fd \\ -fb & a & b \\ -fd & c & d \end{bmatrix} \qquad [\cdot] \qquad (7.7)$$

Given the epipole in the first image $(1, 0, f)^T$ and a point $(0, t, 1)^T$, the corresponding epipolar line is computed as the crossproduct, Equation 7.8. Consequently, the squared distance is computed as Equation 7.9.

$$\mathbf{l}(t) = (0, t, 1) \times (1, 0, f) = (tf, 1, -1) \qquad [\text{px}] \qquad (7.8)$$

$$d(\widetilde{\mathbf{x}}, \mathbf{l}(t))^2 = \frac{t^2}{1 + (tf)^2} \qquad [\text{px}^2] \qquad (7.9)$$

Furthermore, the corresponding epipolar line $\mathbf{l}'$ is found as described in Equation 7.10 and the distance is found as described in Equation 7.11.

$$\mathbf{l}'(t) = F(0, t, 1)^T = \begin{bmatrix} -f'(ct + d) \\ at + b \\ ct + d \end{bmatrix} \qquad [\text{px}] \qquad (7.10)$$

$$d(\widetilde{(}x'), \mathbf{l}'(t))^2 = \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \qquad [\text{px}^2] \qquad (7.11)$$

The total distance function $s(t)$ with respect to t is obtained by substituting Equation 7.9 and 7.11 in 7.4.

$$s(t) = \frac{t^2}{1 + (tf)^2} + \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \qquad [\text{px}^2] \qquad (7.12)$$

	Mean	SD
Original	0.0118	4.7510e-04
Corrected	3.9095e-16	1.1637e-31

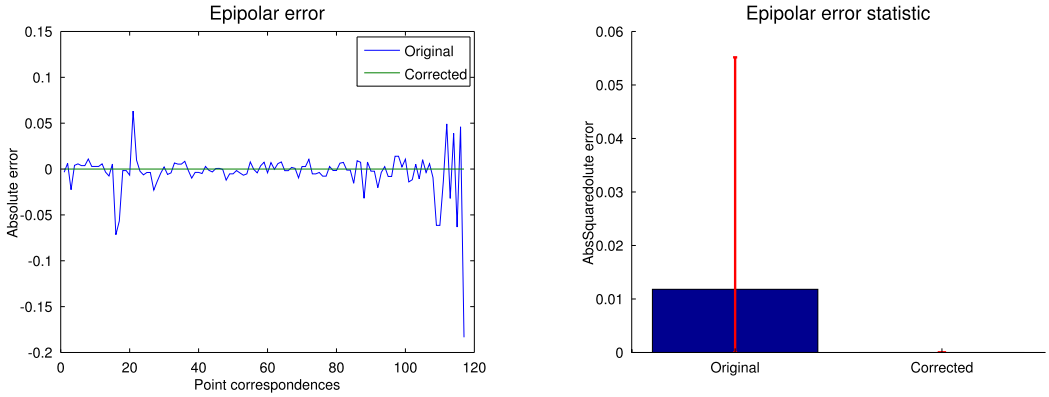
Table 7.1: *Statistic for epipolar error for original and corrected point correspondence.*

Solving Equation 7.12 for the minimum solution can be achieved by deriving it and colleting the two terms over a common denominator. The resulting numerator, Equation 7.13, is a six degree polynomial of t and will therefore provide six solutions (three minima and three maxima). As the solutions are obtained as the roots, both real and complex solutions may exist.

$$0 = t((at+b)^2 + f'^2(ct+d)^2)^2 - (ad-bc)(1+f^2+t^2)^2(at+b)(ct+d) \quad [\text{px}^2] \tag{7.13}$$

The global minimum is determined by evaluation $s(t)$, for the real part of each solution. Let t_{min} represent the solution to $s(t)$ that yields the global minimum. The two epipolar lines $\mathbf{l}$ and $\mathbf{l}'$ are then defined by Equation 7.8 and 7.10. $\widetilde{\mathbf{x}}x$ and $\widetilde{\hat{\mathbf{x}}}'$ can then be defined as the point on $\mathbf{l}$ and $\mathbf{l}'$ that are closest to the origin. For a line $\mathbf{a} = (a_1, a_2, a_3)^T$, the point closest to the origin is computed as $(-a_1a_3, -a_2a_3, a_1^2 + a_2^2)^T$. The obtained points $\widetilde{\hat{\mathbf{x}}}$ and $\widetilde{\hat{\mathbf{x}}}'$ need to be transfered back to the original coordinate system using $T^{-1}R^T\widetilde{\hat{\mathbf{x}}}$ and $T^{-1}R^T\widetilde{\hat{\mathbf{x}}}'$.

Figure 7.9a shows the epipolar error for the original correspondences and the corrected correspondences. Figure 7.9b and Table 7.1 show the mean and 95% prediction interval for the epipolar error.



(a) *Epipolar error for original and corrected point correspondences.*

(b) *Epipolar error statistic, mean and 95% prediction interval.*

Figure 7.9: *Epipolar error and for original and corrected point correspondences.*

7.4 Module validation

The "Feature extraction" is validated using an image and a known homography. The image is then transformed with the homography and feature points and correspondences are estimated. The estimated correspondences are validated based on their location in the second image, x_2. The ground truth is calculated by transforming the feature point in the first image using the homography.

$$Error = |x2 - g(H\widetilde{\mathbf{x}}_1)| \qquad [\text{px}] \tag{7.14}$$

Where:

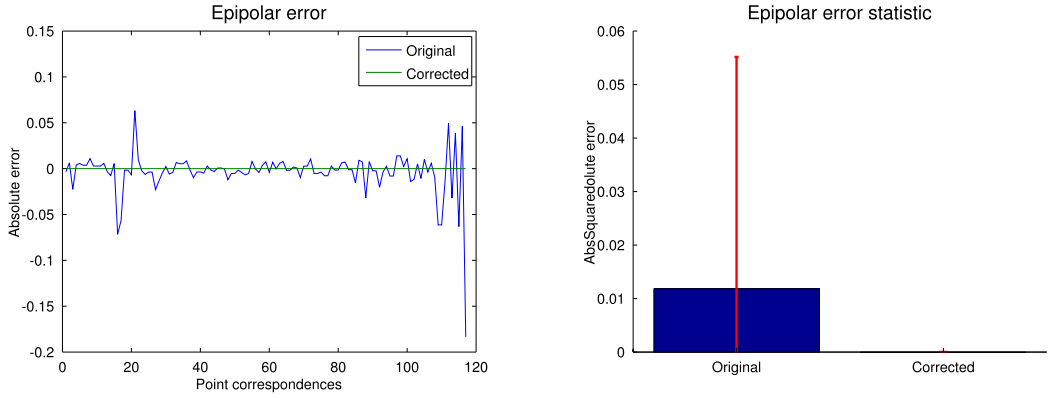
x_1, x_2: Estimated location for feature point correspondence.	[px]
H: Homography	[[·]
$g(\cdot)$: Function that normalize the coordinate by its z component.	[[·]

The homography are calculated by its eight DoF, see Appendix C. Each of the variables is either determined by a normal distribution $\mathcal{N}(\mu, \sigma)$ or a uniform distribution $\mathcal{U}(min, max)$.

$$\begin{aligned}
\theta \quad &\in \mathcal{N}(0, 0.15)[\text{rad}] \\
\phi \quad &\in \mathcal{U}(0, 2\pi)[\text{rad}] \\
\lambda_1, \lambda_2 \quad &\in \mathcal{N}(1, 0.05)[\cdot] \\
t_1, t_2 \quad &\in \mathcal{N}(0, 15)[\cdot]
\end{aligned}$$

Furthermore, the transformed image is subjected to $\mathcal{N}(0, 1)$ pixels of Gaussian noise. The test is run on 100 randomly selected images from the Oxford image dataset. All of the 100 images have passed the pre-processing step, and thus they can be considered of good quality.

Of the 100 pictures, a total of 34633 features are matched. 83.66% of them have an error below 0.25 px for both x and y. Only 3.693% of all matches have an error above 1 px. Figure 7.10a shows a histogram of all errors above one. It can be seen that some very large errors occur. Figure 7.10b show the CDF of all errors, where only the interval 0 to 1 px error are shown. The median error for x is 0.049 px and for y it is 0.056 px, where the average error for errors below 0.25 px are 0.056 px for x and 0.061 for y.



(a) *Histogram of all errors above 1.*

(b) *CDF of all errors (Shown for the interval 0 to 1).*

Figure 7.10: *Result of "Feature extraction" module.*

Figure 7.11 shows an example, where all green lines are matches with an error below 0.25 px, yellow lines are matches with an error between 0.25 and 1 px, and red lines are matches with an error above 1 px.

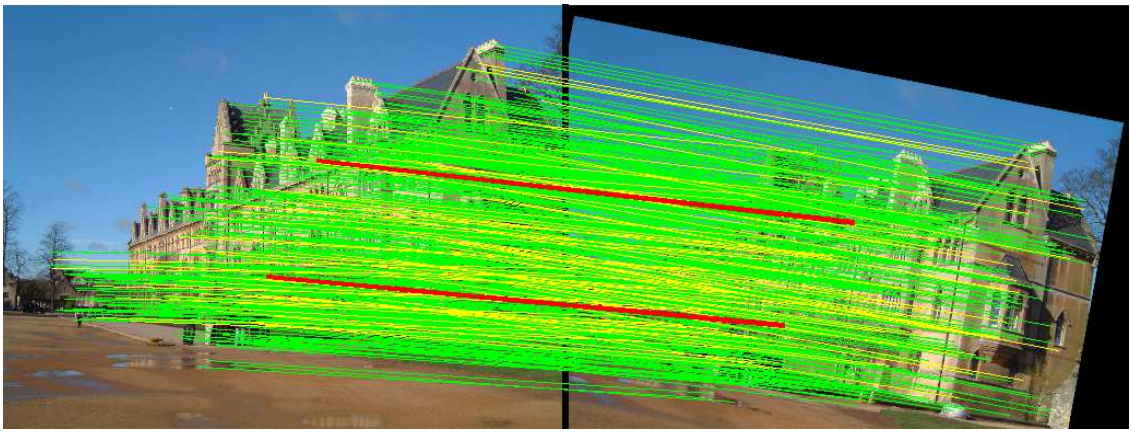


Figure 7.11: *Two images are shown, where the right is the transformed image. Green lines are matches with an error below 0.25 px, yellow lines are matches with an error between 0.25 and 1 px, and red lines are matches with an error above 1 px. The red lines are drawn more thick to make them more visible.*

7.5 Module conclusion

The "Feature extraction" module uses both SIFT and KLT to extract and match feature points between frames. Furthermore, it uses the fundamental matrix to determine outliers and improve the remaining correspondences.

The module validation shows that the "Feature extraction" module is able to estimate the feature point correspondences with an accuracy of maximum 0.25 px for 83.66% compared to the 75% in the requirements. The amount of outliers are found to be 3.693%, where the requirement state a maximum of 5%. It can therefore be concluded that the "Feature extraction" module satisfy both requirements.

Chapter 8
Layer extraction

In this chapter the "Layer extraction" module is described. In Figure 8.1 the placement of this module in the system is illustrated. The task of this module is to determine all the planar surfaces, given feature correspondences. Given two views with different camera locations, the 2D mapping of a planar surface can be represented by homography. Chapter 3 "Geometry of Planar Scenes" provides an analysis of homography and different approaches for layer extraction. This module takes point correspondences between two frames as input, and the module output is segmented layers in both frames.

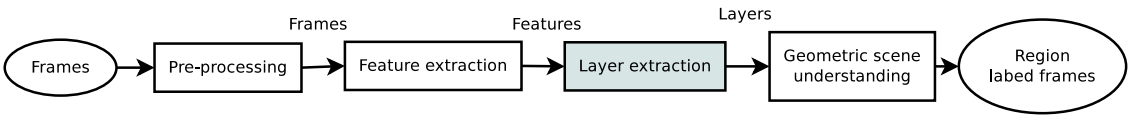


Figure 8.1: *The contexts in which the "Layer extraction" module is placed. The module gets feature point correspondences as input, and outputs the different layers visible in the scene.*

Layer extraction can be divided into three subproblems which are segmentation, motion estimation, and determining the number of layers.

1. **Segmentation**, which pixels belong to the layer?
2. **Motion**, what is the layer motions?
3. **Number of layers**, how many layers are present in the video?

First a simple case where only one layer and all point correspondences are contained is analyzed. In this case the simple normalized DLT algorithm can be used to estimate the motion, e.g. homography. Feature points only provide a sparse segmentation of the layer, but by assuming convexity, can a dense segmentation be obtained, using the convex hull.

The next case is a scene which is composed of multiple layers. Given multiple layers it is unknown which layers contain which points. Thus, a robust approach must be used such as graph cut or RANSAC. Graph cut belongs to the grouping approaches where RANSAC belongs to the dominant approaches, both described in Section 3.5 "Layer extraction".

The main difference of the grouping and dominant approaches is their initial view of the problem. Grouping approaches initialize layers e.g. for every four point correspondences or every point and patch around it[50, 51, 52]. The final layers are obtained by grouping the small layers together. The dominant approaches start out by assuming only one layer and then determines which point correspondences that can be considered inliers to that layer. This process is then repeated until all points are associated with a layer.

The advantage of the graph cut approaches in [50, 51, 52] is that they solve all three problems simultaneously. The disadvantage is that it has a high computational cost, mainly from the image alignment and graph cut.

In this module a new frame work is proposed where RANSAC and graph cut are combined. Furthermore, an extension to the normal RANSAC approach for robust estimation of layers is proposed.

Figure 8.2 shows a flowchart over the "Layer extraction" module developed in this thesis. First, correspondences are processed by an extended RANSAC algorithm. The extended RANSAC algorithm detects and assigns correspondences to a layer. Next, an initial layer shape is estimated. Given the initial shape all points in it are removed and the processes is repeated. When all layers and their initial shape are found graph cut is used to improve the shape. The result of the graph cut method is the final layers. Allthough the module does not directly use more than two frames it can easily be modified to use the information provided by previously detected layers.

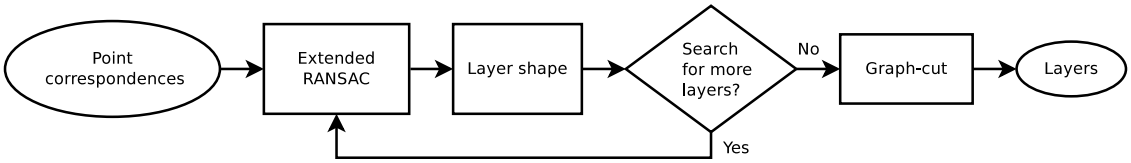


Figure 8.2: *Flow-chart over layer extraction algorithm.*

As described in the Module Specification, Section 5.3, the requirements for the "Layer extraction" module are:

1. 90% of the planar surfaces which have an area over 2500px and contains more than 15 feature points must be detected.
2. All detected layers must on average support minimum 90% of the ground truth layer.
3. All detected layers must on average support maximum 10% outside the ground truth layer.

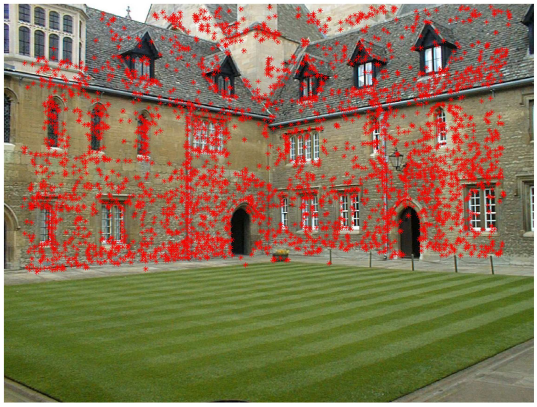
8.1 Extended RANSAC

The assignment of correspondences to new layers is handled by an extended RANSAC

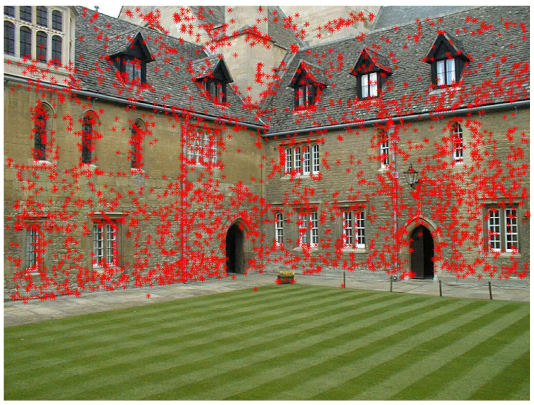
RANSAC is widely used as a robust estimator of homography. The normal approach starts by selecting four random points. Next, it estimates homography using the normalized DLT algorithm. Then, all point correspondences which satisfy this homography is determined by evaluating the geometric cost function and comparing it to a threshold. This procedure is repeated N times. N is calculated as Equation 3.31. For each execution the points corresponding to the most dominant layer are determined. Therefor the RANSAC algorithm must be executed once for each layer there must be extracted.

Figure 8.3 shows two views of a normal scene. There are 2529 point correspondences found between the two images, represented as red stars. For this scene the normal RANSAC algorithm produces the result seen in Figure 8.4. The maximum value for N is set to 2000, and the threshold to 0.0001. The algorithm is stopped when there are less than 15 inliers in a layer. The overall execution time was 11 seconds and 34 layers was extracted.

It can be seen from Figure 8.4 that the normal RANSAC algorithm performs very well when there are many points and the view point changes are large. For the scene shown in Figure 8.4 the average displacement in x,y direction between the two views are 15 and 29 px. Figure 8.5 shows

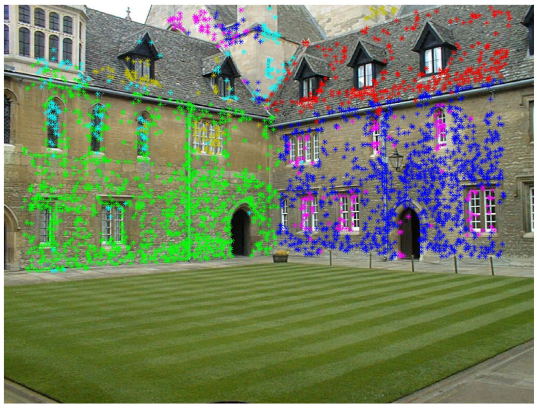


(a) *First view of scene.*



(b) *Second view of scene.*

Figure 8.3: *Image from two different view point of the scene. The view point changes in this example are large. The average displacement in x,y direction between the two views are 15 and 29 px.*



(a) *First six layers marked with different colors.*

(b) *Last nine layers marked with different colors.*

Figure 8.4: *The two figures shows the 13 layers which were extracted by the normal RANSAC algorithm.*

an example with only 1010 point correspondences and a small change in view point. The average point displacement in x, y direction between views is 16, 7 px. 11 layers were extracted in 26 seconds. From the figure it can be seen that the normal RANSAC algorithm performs very poorly on this scene. The algorithm is not able to distinguish layers from each other. This is mainly due to the small difference in homographies. If there are no view point change and therefore only a camera rotation then all points satisfy the same homography.

In this thesis it is proposed to improve the normal RANSAC result by introducing a constraint on the random sampling. The constraint is that points on the same plane are more likely to be located close to each other. This constraint is incorporated such that a random point and the 100 closest neighbors are found. From this subset, five random points instead of four are sampled. With five points it is possible to verify that the points actually satisfy a homography. The five selected points must therefore have a geometric error below the predefined threshold. If not, they are disregarded and five new points are sampled from the subset.

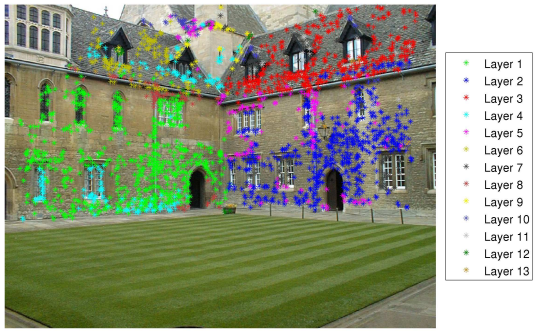
The extended method is more prone to choosing points on the same plane and the maximum N can be reduced to 1000. In Figure 8.6, the results of the two previous scenes are shown for the extended RANSAC algorithm. It can be seen that the extended RANSAC provides a much better

(a) *Example with Low amount of feature points.*

(b) *Nine layers extracted from the scene. Each layer is marked with a different color.*

Figure 8.5: *The two figures show the same scene where points belonging to same layer are marked with the same color.*

result given the scene in Figure 8.6b. For the scene in Figure 8.6a the two algorithms perform equally well. For the total execution time the extended RANSAC algorithm was 32 seconds which is one second faster than the normal RANSAC algorithm. Thus, it is chosen to use the extended RANSAC instead of the normal RANSAC. The extended RANSAC algorithm is also stopped when there are less than 15 inliers in a layer.



(a) *High amount of points and large view point change. Average displacement in x,y direction is 15, 29 px.*

(b) *Low amout of points and small view point change. Average displacement in x,y direction is 16, 7 px.*

Figure 8.6: *The two figures shows the layers extracted using the extended RANSAC algorithm.*

As it can be seen on the images in Figure 8.6 many small layers are estimated. To prevent this an additional stop criteria is added. There must be more than 5% of the initial number of correspondences left before it is tried to extract a new layer.

Next, the initial shape of a layer is determined using the assigned points to that layer.

8.2 Layer shape

Given the initial assignment of points to a layer, it is desirable to obtain the initial layer shape and thereby also the layer mask. The layer mask is a binary image which determines which pixels are associated with the layer. The initial layer shape is used in the next block where it is improved further.

In this work it is chosen to estimate the layer shape using an alpha shape, see Appendix H "Estimate layer shape" for more details. Instead of using a constant alpha value it is shown that a dynamic alpha value provides a better result. The alpha value is determined from the distance

between points assigned to the layer. In Figure 8.7 three different amounts of points are used to estimate the shape of the two facades. It can clearly be seen in Figure 8.7c that the more dense the distribution is, the more accurate the shape fits the actual layer. Figure 8.7a shows a case with a low number of points. Given the low number the distance between points becomes greater. The figure shows that the calculated alpha value is able to provide a good estimate of the two layer shapes and still insure that they stay separated.

(a) *Using 75 points and* $\alpha = 47$

(b) *Using 200 points and* $\alpha = 38$

(c) *Using 800 points and* $\alpha = 33$

Figure 8.7: *Estimated alpha shapes using different number of points distributed uniformly over the purple area. The alpha value is calculated as described in Appendix H.*

The estimated initial layer shape must contain at least 8 points. If there are more than 8 points then the layers shape, correspondences, and homography is saved as a new layer. Finally all inliers are removed from the set of correspondences from which the extended RANSAC algorithm execute on.

The estimated initial shape does not handle small regions very well. Thus, an improvement of the layer shape is needed. The next block described the improvement of layers using graph cut.

8.3 Graph cut

In this block the initial layer shapes are improved using graph cut. Graph cut was first introduced to the computer vision community in 1989 by Greig, Porteous, and Seheult[74]. Graph cut is used to efficiently solve different low level computer vision problems which can be expressed as an energy minimization. Graph cut uses the maximum-flow and minimum-cut theory. The first implemented algorithm to solve this theory was introduced in 1955, by Lester R. Ford and Delbert R. Fulkerson[75].

Although graph cut has become very popular in literature there are some limitations of the method. Graph cut can only be used to determine the global optimum for binary labeling, such as foreground/background image segmentation. Olga Veksler presented in [76] an extension to the graph cut algorithm that can find approximate solutions for multi label graph cut problems. Another limitation is that the graph cut algorithm is biased towards producing a small contour[77]. Therefore, the graph cut is not ideal for extracting a thin object, such as a light pole. The graph cut algorithm used in this thesis was originally implemented by Olga Veksler et al.[76, 78, 79, 80].

In this section, graph cut is used to expand the initial layer. This method is similar to the one presented by Xiao and Shah in [50]. The problem can easily be formulated into the graph cuts framework, as a bipartitioning problem of a node set. The energy function can be expressed with two terms E_{data} and E_{smooth}:

$$\begin{aligned} E &= E_{data}(f) + E_{smooth}(f) && [\cdot] \\ &= \sum_{p \in \mathcal{P}} D(p, f_p) + \sum_{(p,q) \in \mathcal{N}} V(p,q) \cdot T(f_p \neq f_q) && (8.1) \end{aligned}$$

Where:

$E_{data}(f)$: Data energy for labeling f.	$[\cdot]$
$E_{smooth}(f)$: Smoothness energy for labeling f.	$[\cdot]$
$D(p, f_p)$: Data penalty function.	$[\cdot]$
$\mathcal{P}$: A the set of pixels in the image.	$[\cdot]$
$V(p, q)$: Smoothness penalty function.	$[\cdot]$
$\mathcal{N}$: A four-neighbor system.	$[\cdot]$
f_p: Label of a pixel.	$[\cdot]$
$T(\cdot)$: 1 if its argument is true and 0 otherwise.	$[\cdot]$

Figure 8.8 shows an illustration graph with 9 nodes/pixels. The data term is the penalties on the t-links between source and sink. The smoothness term is the penalties on the interconnections between nodes in a four neighbor system (yellow lines). The thick red and blue lines illustrate the label of the node. The green line is optimal graph cut.

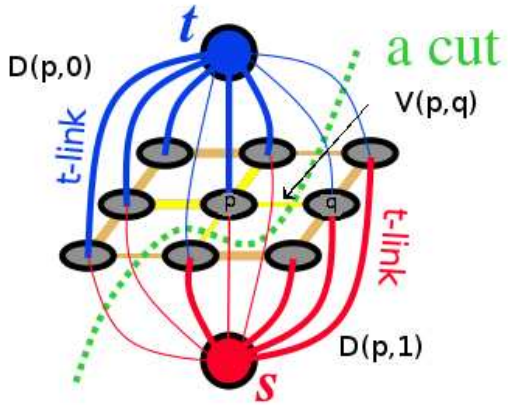
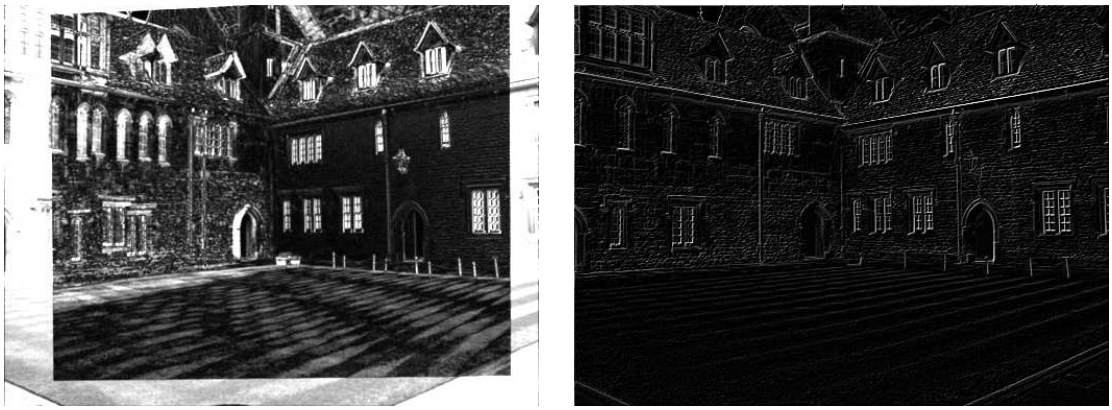


Figure 8.8: *Illustration of graph cut for a graph with nine nodes. The thick blue and red lines illustrate the label of the node. The yellow lines represent the links in a four neighbor system of node p. The smoothness penalty function determines the penalties of the yellow interconnections where the data penalty function determines the t-link penalties. The green line is optimal graph cut.*

In this bipartitioning problem, the label f_p is either 0 or 1. If $f_p = 1$ then p is part of the layer, otherwise it is 0. The data penalty function $D(p, f_p)$ is the penalty for assigning either label 0 or 1 to pixel p. The data penalty function is defined as an approximate unit step function.

$$D(p, f_p) = \begin{array}{ll} \tan^{-1}(\delta(p)^2 - \tau) + \pi/2 & \text{if } f_p = 0, \\ \pi/2 - \tan^{-1}(\delta(p)^2 + \tau) & \text{if } f_p = 1, \end{array} \qquad [\cdot] \qquad (8.2)$$

Where $\delta(p)$ is the absolute image difference of the first frame and a wrapped version of the second. The second image is wrapped with the estimate homography for the layer. Ideally, the absolute image difference is zero if the pixels satisfy the homography and thereby the layer. In practice differences will occur when noise is present. Thus, the two images are smoothed with a 3×3 kernel. The approach described in [50] only use gray scale images. This is expanded to RGB images in this thesis by adding the difference and gradients for all the color channels. Figure 8.9a



(a) *Normalized sum of absolute image difference for RGB between image I_1 and I_2.*

(b) *Sum of RGB maximum gradients for I_1 and I_2*

Figure 8.9: *Normalized absolute image difference and gradients.*

shows the normalized sum of absolute intensity difference for RGB. The second image is wrapped with the homography for the wall layer of Figure 8.6a.

τ is an empirical threshold which can also be used to minimize the effect of noise. The value of τ is increased in the presence of noise. In the pre-processing module the images were controlled for noise. When $\delta(p)^2 = \tau$ then $D(p,1) = D(p,0) = \pi/2$, this is therefore called the critical value. Xiao and Shah use a constant value of $\tau = 64$. Consequently, pixels that has an difference above 8 are more likely not to be part of the layer. In this thesis a variable value of τ is used. The value of τ is calculated as squared sum of the median and mean value divided by two, see Equation 8.3. It is assumed that there are a lot more pixels inside the initial shape which belong to the layer. Thus, the median value is a better approximation of the average difference error for the layer. If the layer mask does not satisfy the layer homography, $\delta(mask)$ will contain many high values. Consequently, this will also result in a very high τ which then results in all pixels being added to the layer. To prevent this the maximum value of τ is set to $50^2 = 2500$.

$$\tau = max(\left(\frac{median(\delta(mask)) + mean(\delta(mask))}{2}\right)^2, \tau_{max}) \qquad [\cdot] \qquad (8.3)$$

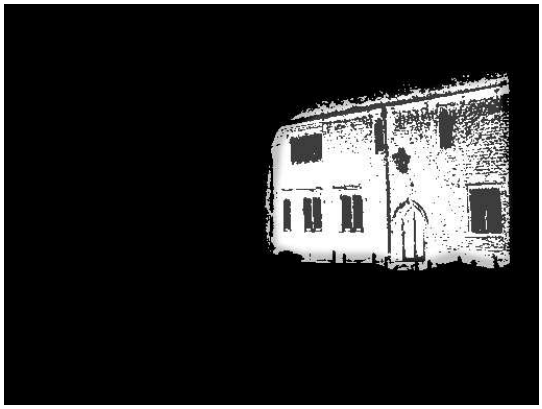
Where:

τ: Critical value.	$[\cdot]$
τ_{max}: Maximum critical value.	$[\cdot]$
$\delta(\cdot)$: Absolute difference image.	$[\cdot]$
$mask$: Pixels of the layer mask.	$[\cdot]$

The smoothness penalty function $V(p,q)$ is calculated for a four-neighbor system. It is defined such that it penalizes changes in intensities among the four neighbors of p, e.g. the gradient of a pixel. Figure 8.9b shows the normalized sum of RGB gradients.

$$V(p,q) = \begin{matrix} 3\lambda \\ \lambda \end{matrix} \quad \begin{matrix} \text{if } \max(|I_1(p) - I_1(q)|, |I_n(p) - I_n(q)|) < \sqrt(\tau) \\ \text{otherwise} \end{matrix} \qquad [\cdot] \qquad (8.4)$$

I_1 and I_n are the two images for which the homography is calculated. The value of λ determines the smoothness of the segmented layer. A large λ will cause a more smooth and connected segmentation. The value of λ is set to the critical value of the data penalty function in Equation 8.2.

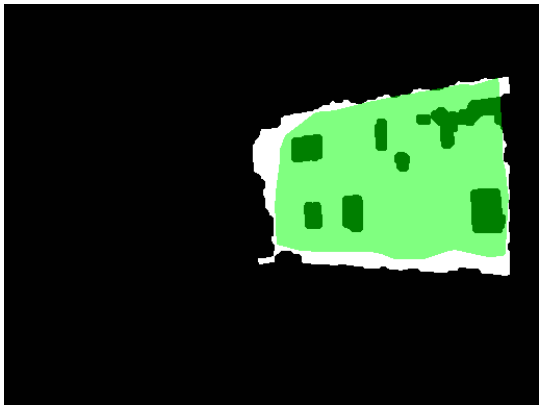


(a) *Normalized data energy for $D(p, 1)$*

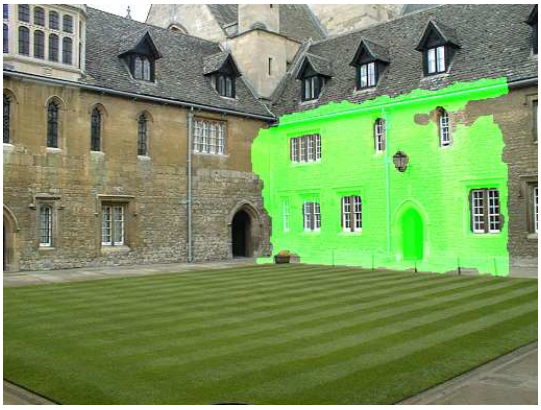
(b) *Normalized data energy for $D(p, 0)$ (Only 10 are added instead of 1000 for better visual)*

Figure 8.10: *The two figures shows data energy. For this layer τ is calculated to 1008.1.*

The layer mask estimated in the previous section is used as an initial labeling of nodes in the graph. However, the partitioning using graph cuts cannot guarantee the gradual expansion or shrinking of a region. To ensure a gradual expansion or shrinking only a local neighborhood around the layer mask must be considered. This can be implemented by convolving the mask with a Gaussian kernel. The layer mask is convolved with a normalized Gaussian kernel with variance 7.5. The higher the variance is the more aggressive is the expansion. The result after convolution is referred to as β. The data penalty function for $D(p, 0)$ is added 1000 where $\beta = 0$ and $D(p, 1)$ is set to 0. This ensures that this part of the picture always will be labeled with label 0. For the region where $\beta > 0$ is $D(p, 0)$ added with $1 - \beta$ and $D(p, 1)$ with β. Figure 8.10 shows the $D(p, 0)$ and $D(p, 1)$.



(a) *Layer mask after graph cut. The white area is the final layer and the green transparent layer is the initial shape.*



(b) *The final layer extracted for the wall surface.*

Figure 8.11: *Resulting improved layer shape.*

To ensure complete expansion of the layer the graph cut approach performed iteratively until the change of layer mask is less than 2.5% or less than 7 iterations. These values are determined through experiments.

Finally layers are deleted or merged together to obtain the final layer segmentation. Layers are merged together if the squared difference between homographies are less than 10 and the two

layer overlap at least 25%. A layer is deleted if another layer overlaps with more than 80%.

In the following section the combined module is validated.

8.4 Module validation

This section describes the module verification. The module is tested on the eight pair of images. Each of these pairs will be refereed to as test case one to eight. Figure 8.12 show the first image from test case three and six.

(a) *First image of test case three.* **(b)** *First image of test case six.*

Figure 8.12: *First image of test case three and six.*

For each image pair a rough ground truth segmentation of layers is manually generated for both images. Figure 8.13 shows the rough ground truth segmentation for the images shown in Figure 8.12. The ground truth segmentation is generated with the requirement of minimum 15 correspondences and have an area above 2500 px in mind.

(a) *Ground truth layers for test case three.* **(b)** *Ground truth layers for test case six.*

Figure 8.13: *Ground truth layers for test case three and six.*

It is controlled for all ground truth layers how many feature point it contains. Table 8.1 shows the number of point correspondences found in each layer for each test case. In total there are 31 ground truth layers which contains more than 15 points.

Figure 8.14 shows the detected and extracted layer for test case three and six. The detected and extracted layers for all test cases can be seen in Appendix I. This appendix, also shows the

Test Case	Layer 1	Layer	Layer 3	Layer 5	Layer 6	Layer 7	
1	1251	1106	25	119	299	78	54
2	1024	593	376	21			
3	96	172	155	16	3		
4	144	39	0	38	1	1	
5	96						
6	360	562	197				
7	276	84	47	21	17	18	
8	1775	1837	296				

Table 8.1: *Number of point correspondences in each layer.*

interim results for the extended RANSAC algorithm.

(a) *Layers extracted for test case three.* **(b)** *Layers extracted for test case six.*

Figure 8.14: *The two figures shows the detected and extracted layers for test case three and six.*

It is determined that a ground truth layer is detected if one of the extracted layers support more than 25% of the area. Furthermore, the extracted layer must not support another ground truth layer with a higher percentage. The support percentage for the ground truth layer is calculated in both images and the maximum value is selected.

It is determined that 80.6% of all ground truth layers were detected. The detected layers have an average support percentage of 72.7%. Furthermore it was calculated how much of the detected layer that were outside (overflow) of the supported ground truth layer. The average overflow percentage is 16.7% percent.

8.5 Module conclusion

The "Layer extraction" module combines an extended version of RANSAC and graph cut. The extended RANSAC is used to detect and assign point correspondences to a layer. Then, the initial layer shape is determined using Alpha shapes. The alpha value is variable and dependent on the distance between the points associated to layer. The initial shape is then improved using graph-cut.

From the module validation it can be concluded that the module is not able to satisfy any of the three requirements. The module is only able to detect 80.6% of all the ground truth layers. Those layers which were detected layers are on average supported 72.7%. It must be noted that

the layers extracted do not contain all the small objects which the rough ground truth layers do, see Figure 8.13b and 8.14b. Thus, it is expected that the average support percent is a littler higher.

The average overflow of 16.7% is mainly due to textureless areas. A textureless area will have an error of 0 in the difference images as all pixels in this region can be mapped to all others. Consequently, will all homographies support this area and graph cut will expand the layer into these regions, see Figure 8.14a.

Chapter 9
Geometric scene understanding

In this chapter, the "Geometric scene understanding" module is described. In Figure 9.1, the placement of this module in the system is illustrated.

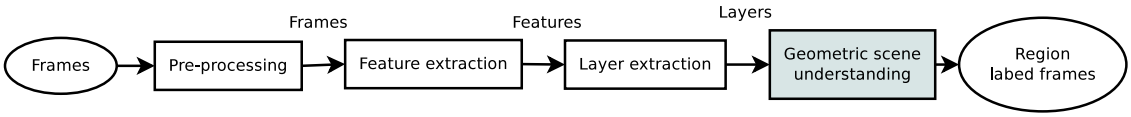


Figure 9.1: *The contexts in which the "Geometric scene understanding" module is placed. The module gets the visible layers as input, and outputs the geometric scene understanding.*

The goal for this module is, given the extracted layers, to determine a geometric understanding of the scene. Given the poor result from the previous module, compared to the litterature, it is chosen to use ground truth segmentations of layers for the development and verification of this module. A qualitative geometric scene understanding must be obtained without doing the reconstruction. Instead, a method based on simple intuitive reasoning is proposed.

Block World uses multiple classifiers to determine segments interrelationship and orientation. These groups are "in-front-of", "behind", "support" , "supported", "left", "center", and "right".

In this thesis only the layers and their homography is estimated so far. With layers only little geometric information about the scene is provided. The provided information is that each layer represents one single planar surface in the scene. In comparison, the approach used in Blocks World leads to multiple segmentations of the same image where segments can contain more than one planar surface. Consequently, the layers provide a better segmentation of an image under the assumption of planar surfaces.

To extract more geometrical information it is desired to determine the layers interrelationship. Thus, all layer connections are determined along with a relative depth estimate. Finally the geometrical scene understanding is determined through simple reasoning. Given initial estimated geometrical scene understanding it is possible to determine vertical layers and then estimate their orientation. Figure 9.2 shows a flowchart of this module. In the following sections are each block described.

As described in the Module Specification, Section 5.3, the requirements for the "Geometric scene understanding" module are:

1. *All the layers must be classified correctly as either a vertical or horizontal layer.*
2. *Minimum 75% of all layer connections must be determined.*
3. *Maximum 5% of determined layer connections may be incorrect.*
4. *0 % percent of left sides may be classified as right sides and vice versa.*

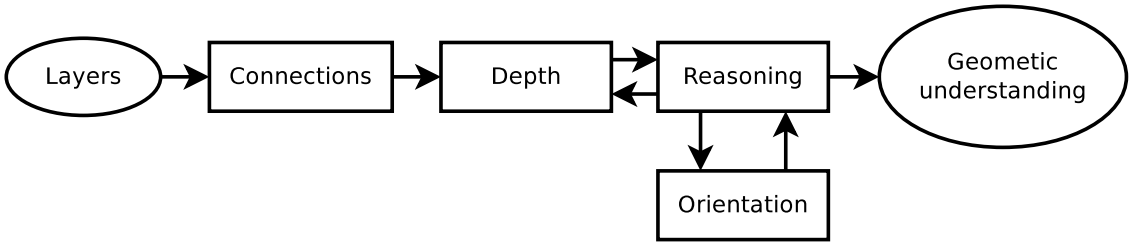


Figure 9.2: *Flow chart over the "Geometric scene understanding" module. First connections and relationships are determined. Next, the relative orientation and depth are determined.*

5. *Maximum 25 % percent of left and right sides may be classified as center sides and vice versa.*

9.1 Connections

This block estimates all intersections between layers. Since Layers represent planar surfaces the intersection between two layers can be represented by a line. A combination of two methods are used to determine the intersection line between two layers.

The first method is investigate the common area of the layer masks. Figure 9.3a and 9.3b shows the ground truth layer masks of the ground and wall layer of Figure 9.5a. The common area is obtained by dilating each layer with a 20 px disk kernel and a simple AND operation, see Figure 9.3c. If a common area exist then it can be concluded that there might be a connection between the two layers.

(a) *Ground layer mask.*

(b) *Wall layer mask.*

(c) *Common area of dilated masks.*

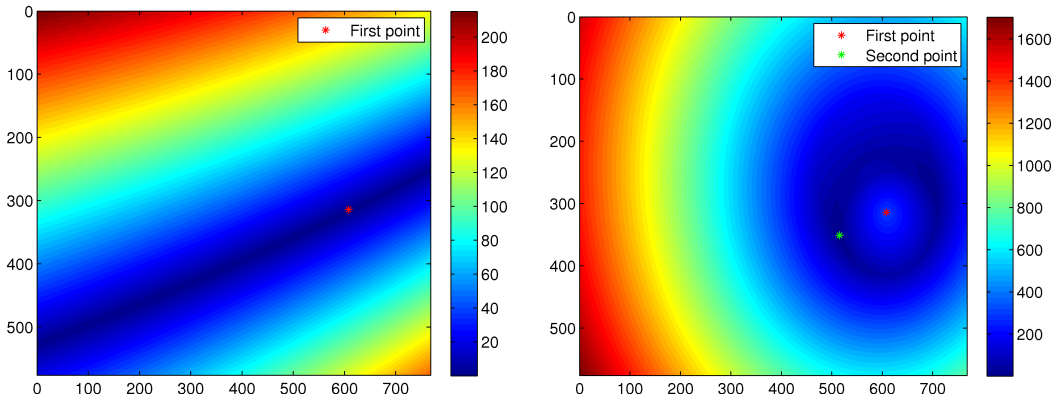
Figure 9.3: *Figure (a) and (b) shows the ground truth layer masks for the ground and wall layer of Figure 9.7. Figure (c) shows the common area of the two layers after dilation with a 20 px disk kernel.*

The second method estimates the precise intersection line. The intersection line is defined as the set of points that satisfy both homographies. It is necessary to estimate at least two points, to derive the intersection line. The first point is determined by minimizing Equation 9.1 over the entire image. Figure 9.4a shows the cost function evaluated over the entire image and the location of the optimal solution. It can clearly be seen that the cost function is minimized along a line.

$$ip_1 = [x\ y\ 1]^T = \min_{x,y} H_{\pi 1}[x\ y\ 1]^T - H_{\pi 2}[x\ y\ 1]^T \quad x, y \in [1 \rightarrow width, 1 \rightarrow height] \tag{9.1}$$

Where:

ip_1: First intersection point. [px]
$H_{\pi 1}$: Homography of first layer [[·]
$H_{\pi 2}$: Homography of second layer [[·]



(a) *Cost function for first point.* **(b)** *Cost function for second point.*

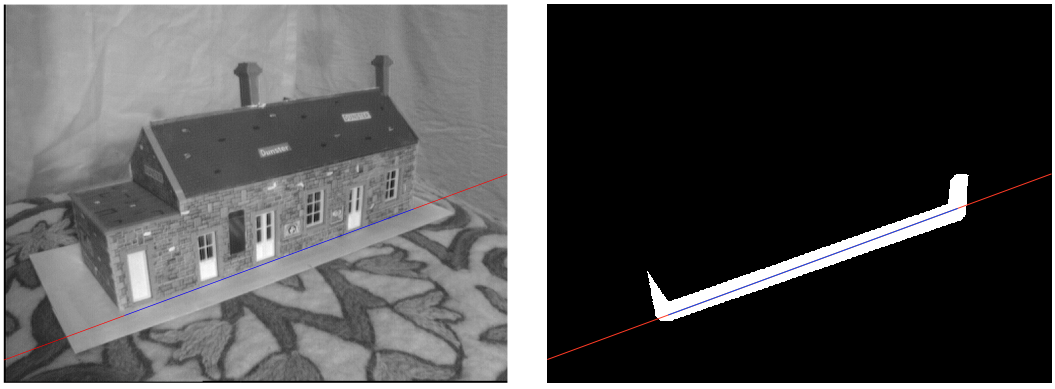
Figure 9.4: *Evaluation over the entire image for both cost functions used to determine the first and second point of the intersection line.*

The second point is determined by minimizing the same cost function and expanding it with a distance cost. The distance cost is added such that the second point does not equal the first point. It is chosen to set the distance cost as Equation 9.2. Figure 9.4b shows the cost function evaluated over the image and the determined location of the first and second point.

$$distance\ cost = (||ip_1 - ip_2|| - 100)^2 \tag{9.2}$$

Where:

ip_2: Second intersection point. [px]



(a) *Image* **(b)** *Common area*

Figure 9.5: *The infinite (red) and finite (blue) intersection line between two layers are shown. The lines are shown on the image and common area of the dilated masks.*

The estimated intersection line is shown as the red line in Figure 9.5 which is infinite. The start and end point on the line are determined using the common area found in the initial test.

Figure 9.5b shows the common area and the infinite (red) and finite (blue) intersection lines. The finite line is determined as the minimum and maximum x,y coordinate of the common area on the intersection line minus the size of the dilation kernel. The finite line is also shown in Figure 9.5a as the blue line.

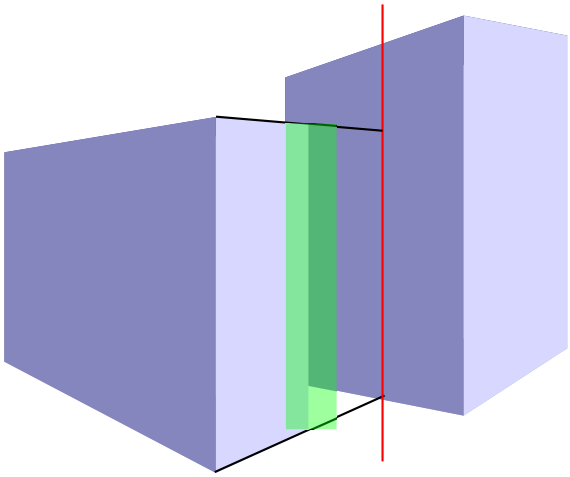


Figure 9.6: *The figure illustrates the case where the estimated intersection line (red) does not correspond to the common area (green).*

The estimated intersection is disregarded if the infinite intersection line and the common area do not satisfy any point. Figure 9.6 illustrates an example of this case. The infinite planar surfaces of the two layers do intersect but the intersection is invalid. Furthermore it can be concluded that the left object must be occluding the right object.

If two layers intersect then the start and end point of the intersection line is stored and the intersection is marked with 1 in a $n \times n$ matrix, where n is the number of layers. For the layers in Figure 9.7a the connection matrix looks like this

$$\begin{array}{c} \\ Ground \\ Wall \\ Roof \end{array} \begin{array}{c} \begin{array}{ccc} Ground & Wall & Roof \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \end{array}$$

9.2 Depth

The relative depth ordering can be obtained either through occlusion boundaries or projective geometry. It is chosen to use projective geometry as this is a simple and robust method as well as providing an advantage in the reasoning step.

In Section 3.1, "What is a homography", is it described that a world point X_π can be parameterized by p given the projected image coordinate x, such that $X_\pi = [x^T p]^T$. Equation 3.1 can be rewritten using this parameterizing to Equation 9.3. Where, H_π is the homography for the planar surface π and e' is the epipole of the second image[81, 15]. It should be noted that x', e', and $H_\pi x$ are collinear.

$$\widetilde{\mathbf{x}}' = H_\pi \widetilde{\mathbf{x}} + pe' \tag{9.3}$$

(a) *Wall of building.* **(b)** *Ground surface.*

Figure 9.7: *Relative depth of two different layers. Yellow points belongs to the layer ($|p| < 2$), the green points are on one side of the layer ($p > 2$), and red points are on the other side ($p < -2$)*

From this it follows that if a point correspondence x, x' belongs to the layer π then $p = 0$ must be satisfied. In the case of oriented projective geometry the sign of p determines which side of the plane the point is located on. Consequently, p can be viewed as the relative depth with respect to the layer π. Figure 9.7 shows two different layers where the yellow point belongs to the layer ($|p| < 2$), the green points are on one side of the layer ($p < -2$), and red points are on the other side ($p > 2$).

It is not known which side of the layer that will have a positive or negative p value. "in-front" and "behind" is defined such that a point closer to the view point is considered "in-front" and a point further away is "behind". For horizontal layers is the ground used as reference such that in front is towards the sky and behind is down in the ground. Thus, it is necessary to have an initial geometric scene understanding relate point to the correct sides, which is provided by the reasoning block. This is described next.

9.3 Reasoning

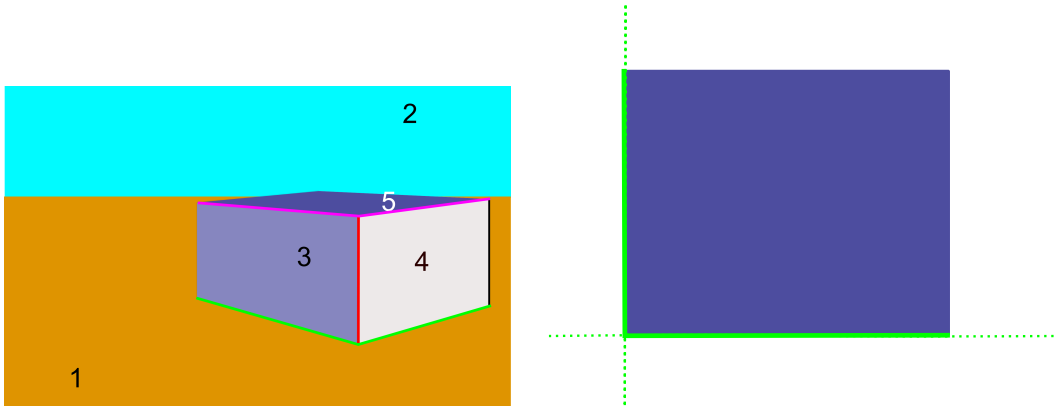
This block determines the geometric scene understanding using simple reasoning on the layer connections and relative depth. First the geometric reasoning is described for a simple scene. Then throughout this section the scene becomes more complex.

Simple geometric reasoning

Figure 9.8 shows the perspective and top view of a simple scene. The scene consists of a ground plane, sky, and a single box object.

Given the simple scene, the ground and sky can be determined as the layers which are behind (below) all other layers. The relative depth constraint is satisfied for both the ground and sky, therefore the layer connections to the image borders are also used. The sky will more likely be connected to the top border of the image where the ground is more likely to be connected to the bottom border. Furthermore only the ground will have connections to other layers.

The vertical layers of an object can be determined from visible connections with the ground. In this scene each side of the object has a visible connection marked with a green line. Consequently, the two layers are classified as vertical layers, and likewise for their intersection (red line). Next, all layers connected to the vertical layers are determined. In this case only layer 5. A layer which



(a) *Perspective view of the simple scene. The visible intersections are illustrated by colored lines.*

(b) *Top view of the simple scene. Colored lines illustrate visible layers and dotted lines illustrate the associated infinite plane. The relative depth is determined with respect to the infinite plane.*

Figure 9.8: *Perspective and top view of the simple scene.*

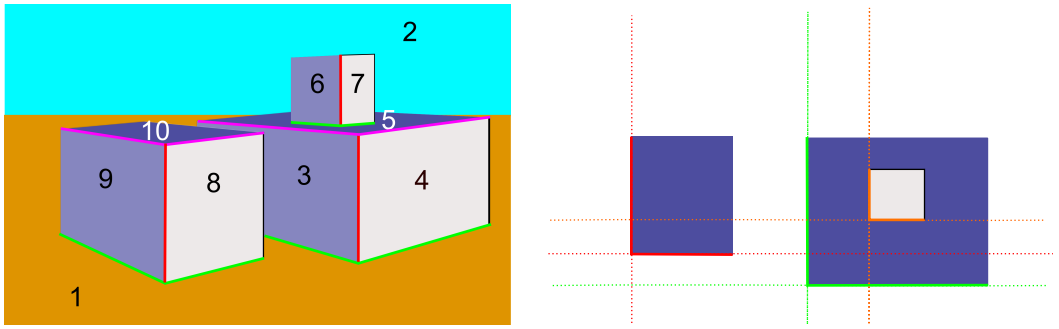
is not ground or a vertical layer but has connections to a vertical layer is defined as a new support layer (e.i. a roof).

Finally, the orientation of each vertical layer is determined with respect to the vertical intersection, see Section 9.4.

The simple scene only contains one object. Next a more complex scene with multiple objects are analyzed.

Medium geometric reasoning

Figure 9.9 shows perspective and top view of the medium scene. Two objects are added to the simple scene to generate the medium scene. The medium scene consists of two non-connected object groups. The first object group contains two objects which are placed on top of each other. The second group contains a single box as in the simple scene.



(a) *Perspective view of the medium scene. The visible intersections are illustrated by colored lines.*

(b) *Top view of the medium scene. Colored lines illustrate visible layers and dotted lines illustrate the associated infinite plane. The relative depth is determined with respect to the infinite plane.*

Figure 9.9: *Perspective and top view of medium scene.*

Given the scene in Figure 9.9a, the ground and sky are determined as in the simple case. The

two object groups are distinguished from each other by the fact that they only share connections to the ground layer. The scene can therefore be decomposed in a recursive approach where the first step is to determine the vertical layers connected to the ground. Next all layers that are supported by the vertical layers are determined. These layers represent new support layers. The process is then repeated such that all vertical layers connected to the new support layers are determined. Given the medium scene the obtained decomposition is:

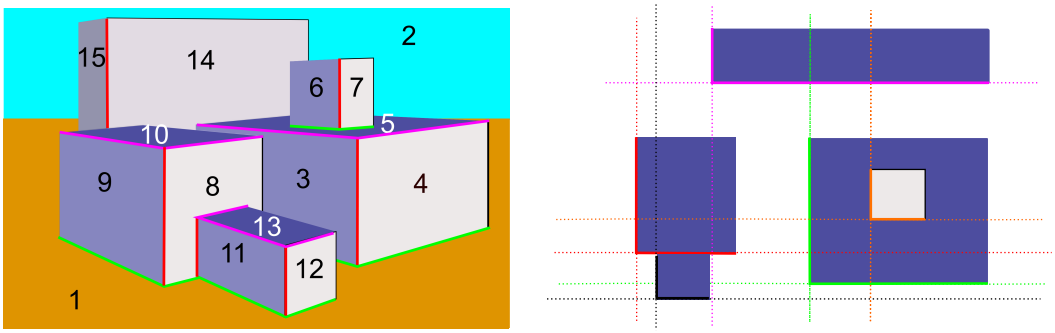
1. a) Vertical layers 3 and 4 which support
   b) Support layer 5 which supports
   c) Vertical layer 6 and 7 which support
   d) None
2. a) vertical layers 8 and 9 which support
   b) Support layer 10 which supports
   c) None

The decomposition is verified using the relative depths (i.e. if layer 5 supports 6 and 7 then they both must be above layer 5). Furthermore, the relative depth is used to determine the relative location for each object group. Figure 9.9b shows a top view of the medium scene where the dotted lines illustrate the infinite planes. The relative depths are determined with respect to these infinite planes. From this it can be concluded that the right object group is placed on the right side of the left object.

So far all object groups have had visible intersection lines with the ground. In the next scene the final reasoning is described which deals with cases where the ground intersection is not known.

Difficult geometric reasoning

Figure 9.10 shows the perspective and top view of the difficult scene. Two objects are added to the medium scene in Figure 9.9 to generate the complex scene. The complex scene consists of three non connected object groups. The first group is similar to the one in the medium scene, see layer 3 to 7. The second group consist of two objects which are located in front of each other, see layer 8 to 13. The third group is a single object located behind the first and second object group.



(a) *Perspective view of difficult scene. The visible intersections are illustrated by colored lines.*

(b) *Top view of the complex scene. Colored lines illustrate visible layers and dotted lines illustrate the associated infinite plane. The relative depth is determined with respect to the infinite plane.*

Figure 9.10: *Perspective and top view of difficult scene.*

For the medium scene, the recursive decomposition was initiated by the vertical layers visible ground intersections. For the difficult scene this approach will not detect the third object group,

as no ground intersections are visible. Thus, a second stage of decomposition is performed, to group the remaining layers based on their connections.

Given the difficult scene the first decomposition provides the following object groups:

1. a) Vertical layers 3 and 4 which support
   b) Layer 5 which supports
   c) Vertical layer 6 and 7 which support
   d) None
2. a) vertical layers 8, 9, 11, and 12 which support
   b) Layer 10 and 13 which support
   c) None

The second stage of decomposition provides information that layers 14 and 15 belong to the same object. It is possible to determine that layer 14 and 15 represent vertical surfaces based on the relative depth and the orientation of the vertical intersection line. The orientation of the vertical intersection line is compared with the vertical intersection lines of other object groups.

Layer 13 is connected with layer 8 but layer 8 does not satisfy the relative depth constraint which was proposed as a verification of the medium scene. Consequently, it is not possible to verify the scene using the depth constraint.

9.4 Orientation

In this block, the orientation of a vertical layer is determined. The orientation of the horizontal layers are of little interest. This is because it is either the ground or dependent upon the vertical layers that support it.

The orientation of a layer can be determined from the corresponding vanishing points, see Figure 9.11. If the vanishing point is located on the right side of the layer it is a left side, and vice versa for the right side. If the vanishing points are close to infinity the layer represents a frontal side. Vanishing points can be determined from straight lines in an image[82, 83, 6, 15, 84]. Instead of determining the straight lines, the orientation can be determined by assuming either that the horizontal line is near horizontal or vertical planes are near orthogonal on the ground.

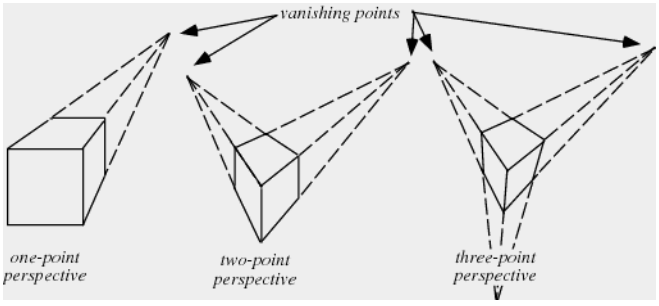


Figure 9.11: *One, two, and three vanishing point geometry*

In this work it is assumed that the vertical planes are near orthogonal on the ground, which is a reasonable assumption in man-made environments (see Figure 9.12). Given this assumption the intersection line will be orthogonal to the horizon line and the orientation of a layer can be determined by its horizontal intersection line with the support layer (see Figure 9.13).

Figure 9.12: *Examples of vertical layers which are orthogonal to the support layer.*

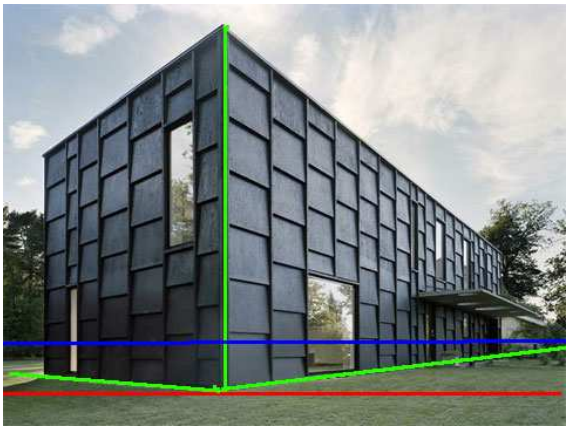


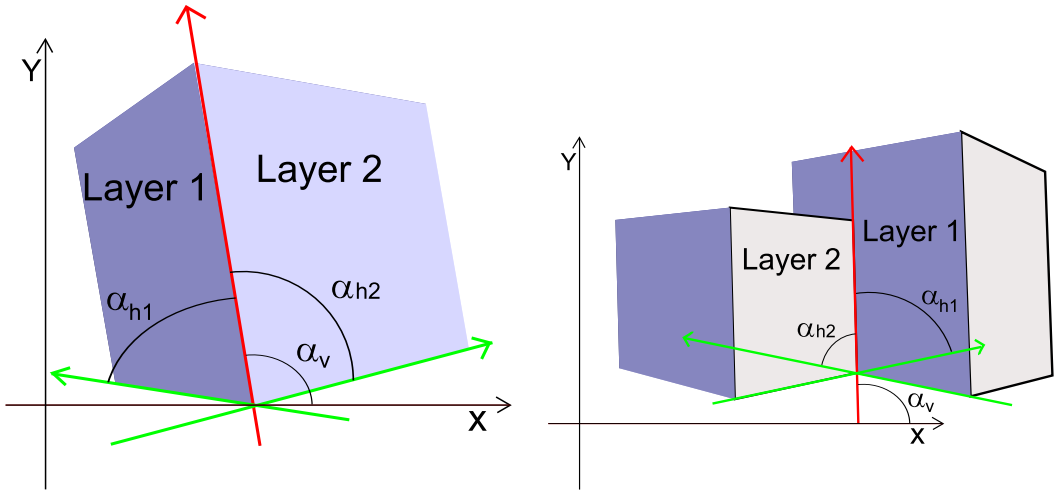
Figure 9.13: *The figure shows a house consisting of two layers. The intersection lines between each layer and the ground is shown as green lines. The Blue line is the horizon line found from two vanishing points. The red line is the x axis of the image. It can be seen that the vertical intersection is orthogonal on the horizon line.*

Figure 9.14 shows two examples of layer relationship given layer orientation. In Figure 9.14a a left-right case is shown. This corresponds to a corner of a building. Figure 9.14b shows a right-left case which correspond the intersection of two blocks. The orientation of a layer is determined by the angle between the vertical intersection (red line) and the horizontal direction vector (green lines). The horizontal direction vector is defined as the direction from the point with lowest y-coordinate to the other point. Given this definition, the angle of a right oriented layer α_{h2} will be negative, the angle for a left oriented layer will be positive α_{h1}. If the angle is close to 90° it is considered a frontal side. Equation 9.4 describes the classification.

$$Orientation = \begin{array}{rl} |\alpha| > 80° : & Center \\ \alpha < 0° : & Right \\ \alpha > 0° : & Left \end{array} \tag{9.4}$$

Where:

α: Angle between vertical and horizontal intersection line. [deg]



(a) *Left-right orientated layers* **(b)** *Right-left orientated layers*

Figure 9.14: *Illustration of left-right and right-left orientated layers and the angles between vertical and horizontal intersection lines. Green lines represent horizontal intersection lines between the layer and the support. The red line represent the vertical intersection line between the two layers.*

Some scenes consist of multiple vertical intersections lines. Instead of using each of them separately, the median orientation is used as a reference. This provides a more robust result in case of a badly estimated intersection line.

9.5 Module validation

This section describes the module verification. The module is tested on images where ground truth layer segmentations are determined. Both images from the Oxford multi-view dataset and images taken specifically for this thesis. In total the module is tested on eight pairs of images. Table 9.1 shows the result of the estimated orientation for all layers and Table 9.2 shows the layer intersections. Only two of the tests contain horizontal layers which are all classified correct. For the orientation test an error 1 is when a left side is estimated as a right side or vice versa. An error 2 is a left or right side which is estimated as a center or vice versa.

It should be noted that it was only possible to estimate the ground layer in four of the tests. In the other four tests the layer orientation is estimated in respect to the lowest edge which is found by differentiation of the the layer mask with respect to y. These lines are not counted as layer intersections in Table 9.2 as the ground layer has not been determined.

	Number	Percent
Error 1	2	8.7%
Error 2	8	34.8%
Correct	13	56.5%
Total	23	100%

Table 9.1: *Result of estimating orientation of layers. Left sides which are estimated as a right side is considered an error 1. Left and right sides which are estimated as a center side is considered an error 2.*

Below is one of the best results shown and described, (the image is from the Oxford dataset). The other seven tests can be seen in Appendix J and on the CD.

	Number	Percent
Incorrect	0	0%
Missing	4	20%
Correct	16	80%
Total	20	100%

Table 9.2: *Result of estimating connections between layers.*

Figure 9.15 shows one of the images and the ground truth layers which the module is verified upon. The homography for each layer is estimated using point correspondences. A point correspondence is associated with a layer if it is located inside the layer mask. Only one example is shown here but is tested on multiple scenes.

(a) *Image from Oxford multi-view dataset*

(b) *Ground truth layer segmentation into 7 layers*

Figure 9.15: *Example of Oxford multi-view dataset, and ground truth layer segmentation.*

Figure 9.16a shows the estimated layer connections. The estimated connection matrix is:

$$\begin{array}{c} \\ 1(Wall) \\ 2(Roof) \\ 3(Roof) \\ 4(Wall) \\ 5(Ground) \\ 6(Wall) \\ 7(Wall) \end{array} \begin{array}{c} \begin{array}{ccccccc} 1(Wall) & 2(Roof) & 3(Roof) & 4(Wall) & 5(Ground) & 6(Wall) & 7(Wall) \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Given the connection matrix the following decomposition of the scene is obtained.

Ground is layer 5 which supports following objects:

1. Object 1
    a) 1st Vertical: Layer 1 and 4
    b) 2nd support: Layer 2 and 3
    c) 2nd Vertical: Layer 6 and 7 (which is supported by layer: 2 and 3)

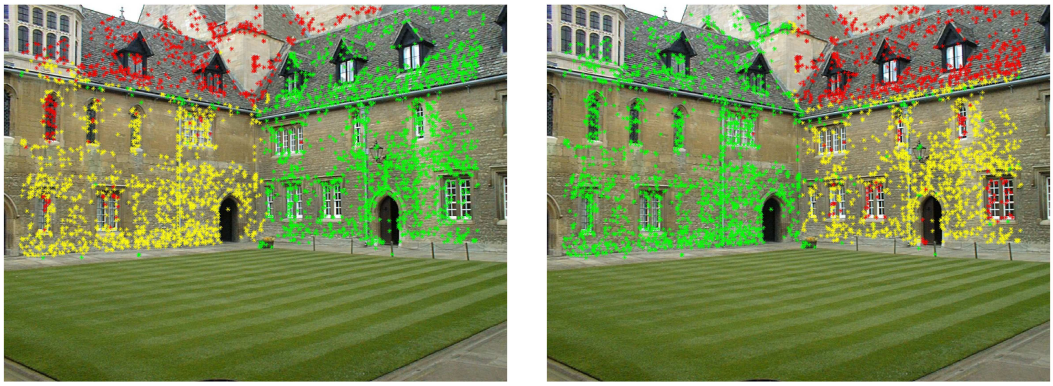
(a) *Estimated layer connections* **(b)** *Estimated layer orientation*

Figure 9.16

Figure 9.16b shows the estimated orientation of all the vertical layers. The figure also shows the two vertical intersection lines and the corresponding orthogonal line. Each of the vertical layers along with their classification and angle with respect to the average orientation are listed below.

- Layer 1: Right (-78°)
- Layer 4: center (95°)
- Layer 6: Left (45°)
- Layer 7: Right (-27°)

Figure 9.17 shows the relative depth estimates for layer 1 and 4.



(a) *Relative depth with respect to layer 1.* **(b)** *Relative depth with respect to layer 4.*

Figure 9.17: *The Figure shows two relative depth estimates. Points on the plane are marked with yellow, points that are in front are green, and points behind are red.*

Given all this, the following scene model in Figure 9.18 can be determined. Due to time limitations, it has not been possible to implement all the reasoning. Therefore, is Figure 9.18 drawn manually but it could be generated automatically from the scene decomposition and reletive depths.

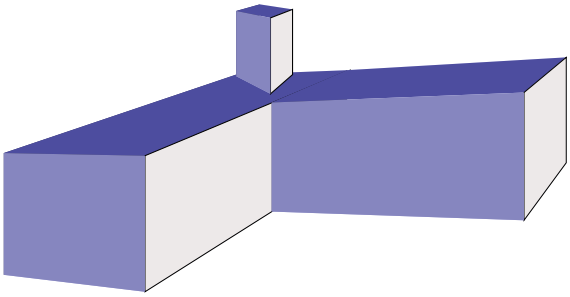


Figure 9.18: *Obtained geometric scene understanding for scene in Figure 9.15a. (note: the figure is drawn manually but could have been created automatically)*

9.6 Module conclusion

The "Geometric scene understanding" module uses homographies and layer masks to estimate relative depth and layer connections. Given these, it then uses a simple initiative reasoning to decompose the scene into objects. Each object is composed from vertical and support layers. The orientation of each vertical layer is determined.

From the orientations in Table 9.1 it can be seen that the module estimates the true orientation 56.5% of the time which is lower than the required 75%. Also the percentage of error 1 is above the required 0%. This is mainly due to the fact that it is difficult to estimate the orientation only from the ground intersection. To improve the orientation result further the upper intersection line (if any) is used. Using the lower and upper intersection line it is possible to estimate a vanishing point (see Figure 9.19). If it is not possible to determine these intersection lines, geometric clues such as straight lines can be used to estimate the vanishing point.

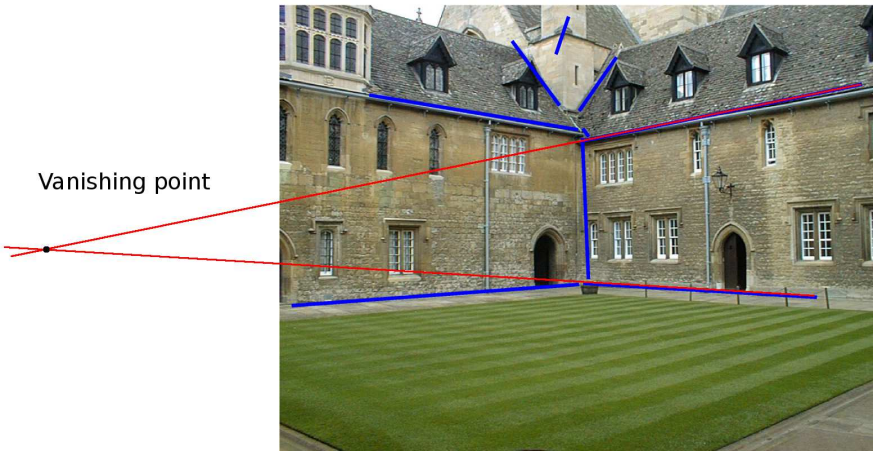


Figure 9.19: *Improvement of orientation using vanishing point.*

From Table 9.2 it can be seen that the module is able to detect 80% of all intersection lines and with out estimating any incorrect lines. The missing intersection lines are mainly due to error in one of the homographies used.

It can be concluded that it is possible to obtain a satisfying geometrical scene understanding when correct homographies and layer masks are provided. It can also be concluded that an improved orientation estimator is needed. Next, chapter describes the combined system verification.

Part III

Verification

This part covers the verification of the system designed in Part II. First, the system is tested and the results are compared to the requirements. Next, the final conlusion of this thesis is presented along with a comment on limitations and future work.

Chapter 10
System validation

In this chapter, the system is described. The system has been validated on two video sequences of semi-natural scenes, i.e. scenes that are natural but set up specifically to validate this system, see Figure 10.1. The first and second video consist of 96 and 118 frames, respectively.

As the "Layer extraction" and "Geometric scene understanding" modules only use two frames and not multiple frames, it is chosen to select two random frames. The random frames are selected from the pre-processing result and must have an average correspondence disparity of minimum 15px.

(a) *Frame from the first video* **(b)** *Frame from the second video*

Figure 10.1: *One frame from each of the videos used for the system.*

The result of both validation tests are described blow, followed by a discussion.

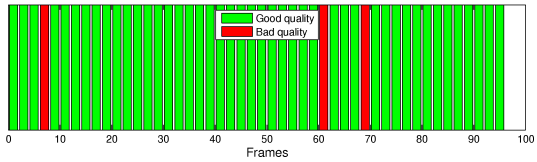
10.1 Validation on video 1

This section shows the result of the system validation on a video of the scene shown in Figure 10.1a. The scene consists of one object, which has five planar surfaces (visible in the video), i.e. ground, left, right and top of box, and the left side of a CD.

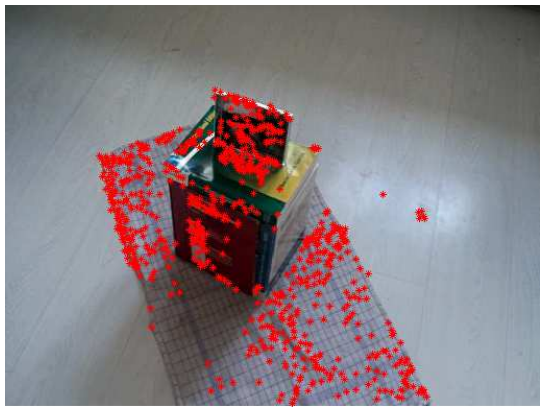
Figure 10.2a shows the pre-processing result, where 84 frames were accepted. Figure 10.2b shows all the feature point correspondences extracted from the two random frames. There were extracted 645 correspondences.

Figure 10.3 shows the six estimated layers from the extended RANSAC algorithm and the final four improved layers.

Figure 10.4 shows the estimated layer connections and the orientation of the vertical layers.

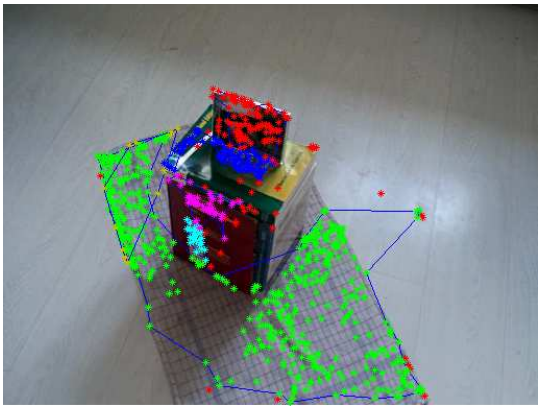


(a) *Pre-processing result; 84 frames classified as good.*

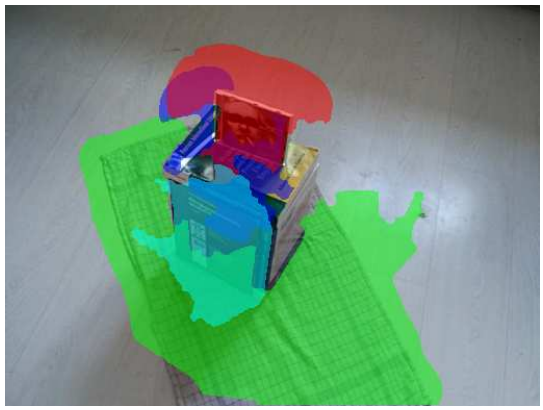


(b) *Feature extraction result 645 feature correspondences extracted.*

Figure 10.2: *Pre-processing and feature extraction result for video 1.*

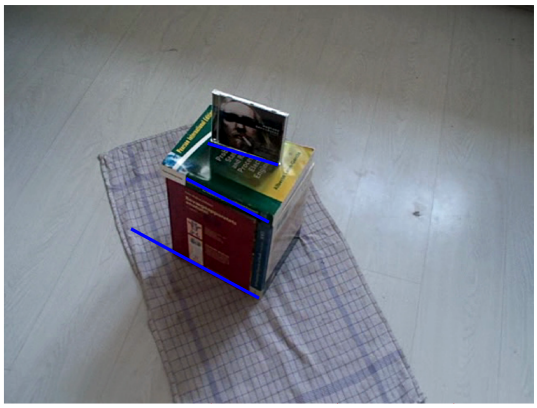


(a) *Layers estimated by extended RANSAC.*

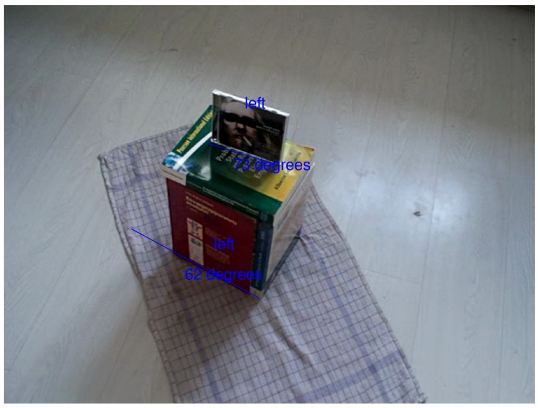


(b) *Final estimate of Layers.*

Figure 10.3: *Estimated layers by extended RANSAC and the final estimated layers for scene 1.*



(a) *Estimated layer connections.*



(b) *Orientation of vertical layers.*

Figure 10.4: *Estimated layer connection and orientation of the vertical layers for scene 1.*

Below, the estimated object graph wich are the outputtet by the system:

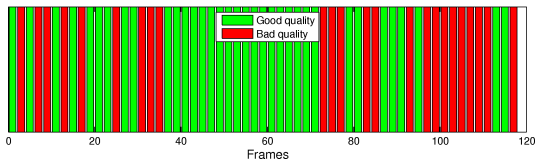
Build graph
Object 1:

Layer 1 support: layer: 4 (left),
Layer 2 support: layer: 3 (left),

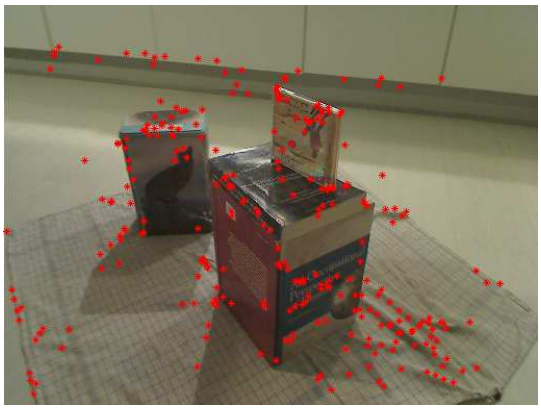
10.2 Validation on video 2

This section shows the result of the system validation on a video of the scene seen in Figure 10.1a. The scene consists of two objects; a box and a box with at CD on top. In total, there are eight visible planar surfaces throughout the video.

Figure 10.5a shows the pre-processing result, where 70 frames were accepted. Figure 10.5b shows all the feature point correspondences extracted from the two random frames. 285 correspondences were extracted.



(a) *Pre-processing result; 70 frames classified as good.*



(b) *Feature extraction result; 285 features extracted.*

Figure 10.5: *Pre-processing and feature extraction result for video 2.*

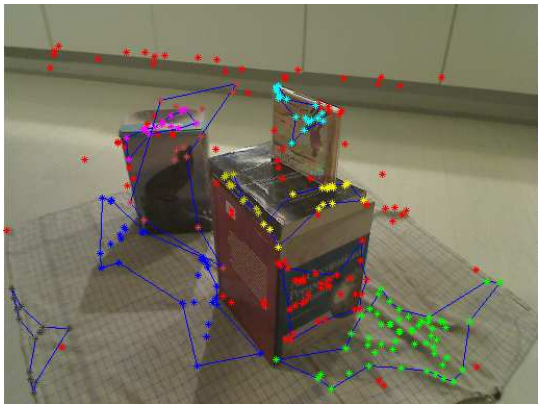
Figure 10.6 shows the nine estimated layers from the extended RANSAC algorithm and the final six improved layers.

Figure 10.7 shows the estimated layer connections and the orientation of the vertical layers.

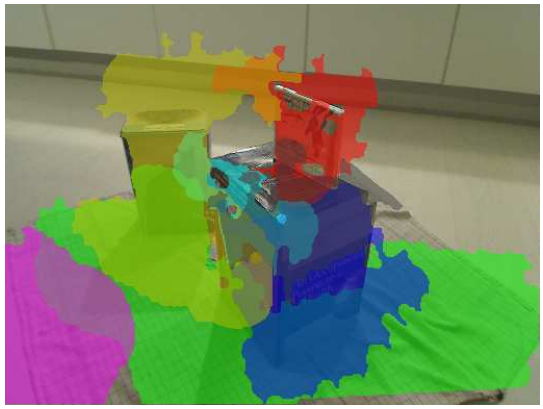
Below, the estimated object graph wich are the outputtet by the system:

Build graph
Object 1:

Layer 1 support: layer: 2 (Right),6 (left),
Layer 4 support: layer:
Layer 7 support: layer:
Layer 3 support: layer:

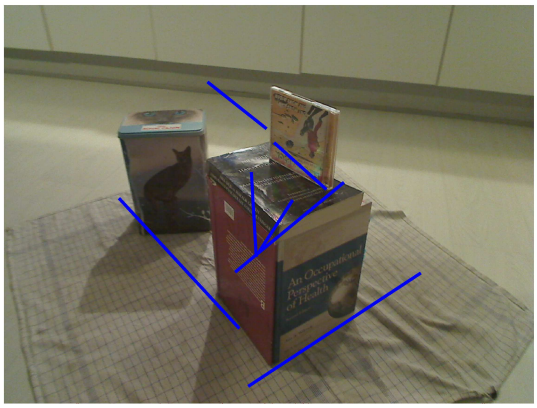


(a) *Layers estimated by extended RANSAC.*



(b) *Final estimate of Layers.*

Figure 10.6: *Estimated layers by extended RANSAC and the final estimated layers for scene 1.*



(a) *Estimated layer connections.*

(b) *Orientation of vertical layers.*

Figure 10.7: *Estimated layer connection and orientation of the vertical layers for scene 1.*

10.3 Discussion

This section will provide a discussion about the obtained results in the two validation tests. It can be seen from the first video that the system provides a good geometric scene understanding. The system is able to detect the layer connections between all layers and estimate the true orientation. The system is also able to distinguish vertical and horizontal layers from each other.

From the second video, it can be seen that the system provides a bad scene understanding. This is manly due to the bad layer extraction. It can be seen on Figure 10.6 that the extended RANSAC is able to detect most of the layers. It can also be seen that especially the ground layer is divided into multiple layers. After improving the layers, some are merged together. This almost results in all ground layers being merged together (green and purple left). The merging also results in the left wall layer of the two object being merged together into layer 6. Consequently, the two objects are now considered as one object, which supports the rest of the layers. The CD (layer 3) is also classified as a horizontal layer because it has a connection with layer 2.

Chapter 11
Conclusion

The goal of this project was to develop a system that could generate a qualitative geometric scene understanding using planar surfaces in multiple views.

First, the problem of geometric scene understanding was investigated thoroughly in the Analysis. To simplify the task, the use of planar surfaces in two views was analyzed. It was found that homography represents the transformation of pixels from the first image to the second image. Through a comparison of the layer extraction algorithms based on homography, it was chosen to use a combination of RANSAC and graph cut to extract the planar surfaces.

After this, the Program Design was defined. Here, the system was split into four modules, for which the external interfaces and requirements were defined. The four modules are "Preprocessing", which estimates the amount of blur and noise in a frame, "Feature extraction", which determines very accurate feature point correspondences, "Layer extraction", which determines all the visible planar surfaces and the corresponding segmentation, and "Geometric scene understanding", which combines layer connections, orientation and relative depths using simple reasoning into a geometric scene understanding.

The modules were designed and evaluated separately and described in the chapters following the Program Design. After this, the entire system was qualitatively evaluated in the System Validation. The System Validation results showed that the system was able to estimate a correct geometric scene understanding if accurate layers were provided. This conclusion is supported by the result of the "Geometric scene understanding" module validation.

From the module validation of the "Layer extraction" module, it can be concluded that the extended RANSAC algorithm is able to detect nearly all layers. It can also be concluded that the final improvement using graph-cut is not able to provide a satisfying layer segmentation.

The system works under the following assumptions:

- The scene must consist of static planar surfaces.
- Each planar surface must contain enough feature correspondences so that accurate layer extraction can be achieved.

This work makes the following contribution to the problem of qualitative geometric scene understanding: A simple and intuitive algorithm that, when provided with homographies and layer segmentations is able to determine a good qualitative geometric scene understanding. In this thesis, it was also proposed to use an extended sampling approach for RANSAC. The new extended RANSAC provided better results than with normal sampling for the case of layer segmentation.

Suggestions on future work for improving the system are given in the following chapter.

Chapter 12
Future Work

This section describes possible improvements and extensions of the system. First, improvement of each module are described followed by improvements of the entire system.

Pre-processing The "Pre-processing" module can be improved by implementing the noise and blur removal blocks. Implementing these will result in images of better quality, and make the system more robust towards low quality video.

Feature extraction The feature extracted in this module already have a very high accuracy. Thus, an improvement could be a faster feature such as Speeded Up Robust Features (SURF)[85]. Another improvement is to be able to detect and match lines between frames.

Layer extraction If lines can be detected and matched between frames in the feature extraction module, they can be combined with points for better RANSAC homography estimation.

As mentioned in the conclusion, it is necessary to improve the final layer segmentation. One way to improve the segmented layer is through alpha-beta swap, which enables graph cut for multi label problems. The main idea of the alpha-beta swap algorithm is to successively segment all α pixels from β pixels with graph cuts and the algorithm will change the $\alpha - \beta$. This will provide a near-optimal layer segmentation, where layers do not overlap.

Another improvement is to extend the module to handle multiple frames so that layer extraction can be performed for an entire video.

Geometric scene understanding Due to time limitations, it has not been possible to implement all the reasoning in this module. Thus, this module can be improved by implementing the missing reasoning. This is especially the reasoning about relative depth. The estimation of the orientation also needs to be improved so that better classification between left, right and center can be archived. The improved orientation can be determined from vanishing points. These vanishing points can either be estimated from layer connection or straight lines inside the layer. If lines are used for homography estimation, these are already available and RANSAC can be used to robustly estimate the three most dominant vanishing points.

Entire system The entire system can be improved by incorporating more of the temporal information available in the video, as this will provide the system information about all layers seen in the scene and not only the ones visible in two frames. The system can also be improved by introducing a visual representation of the obtained geometrical scene understanding. The visual representation could be implemented using a VRML model.

Part IV

Appendix

Appendix A
Blocks World

This section describes main principle of the Blocks World method for geometric scene understanding. The Blocks World paper is an extension of multiple other papers published by Derek Hoiem and/or Martial Hebert[86, 87, 88, 89].

The first step of the Block World method is to classify segments. This work was published in [86]. Here they try to construct the surface layout by labeling the image into geometric classes. They have analyzed 155 pictures and found that more than 97% of all pixels belong to one of these types: horizontal (support) surfaces, nearly vertical surfaces, or the sky. From this, they introduce three classes: support, vertical, and sky.

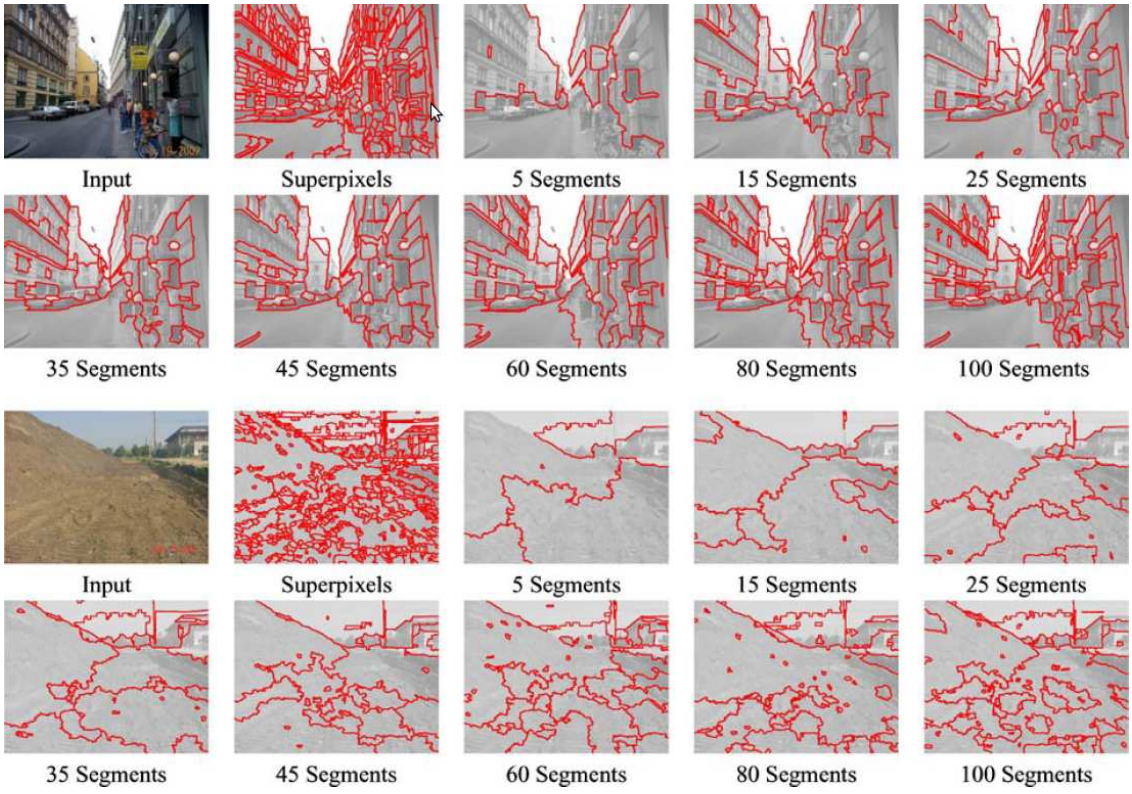


Figure A.1: *Examples of multiple segmentations of two different images.[86]*

The support class covers roads, grounds, lakes, etc. while the vertical class contains walls, cliffs, the curb sides, people, trees, etc. Given the large variety of the vertical class, it is divided in to subclasses; planar surfaces facing to the "left", "center", or "right" of the viewer and non-planar

surfaces that are either "porous" or "solid". The "porous" class contains tree leaves, shrubs and chain link fences, all surfaces which do not have a solid continuous surface. The "solid" class is non-planar but have a solid continuous surface, such as a ball.

A three level representation of the image is used. Level one is pixels, two is superpixels and three is multiple segmentations. Superpixels and segmentations are estimated using the graph-based segmentation technique of Felzenszwalb and Huttenlocher where code is available online[90].

The graph based segmentation, splits an image into n_s segments. With no prior information about the scene, 15 different segmentations of the input image is generated. The following number of segments are used: $n_s \in 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100$. Each of these segmentations provides a different view of the image (see Figure A.1). All the segmentations are put into a "bag" of segments.

All the segmentations are put into a "bag" of segments. Each segment is then assigned a probability that belongs to each of the classes. The probability is based on different clues such as material, location, texture gradients, shading, vanishing points, etc. The probability is calculated using classifiers for each class. Boosted decision trees are used for each classifier as they provide automatic feature selection and limited modeling of the joint statistics of data. The calculated probabilities are then propagated down to superpixel level, using the assign probability for all segments that contain the superpixel.

Figure A.2 shows the assigned probability for the given input image.

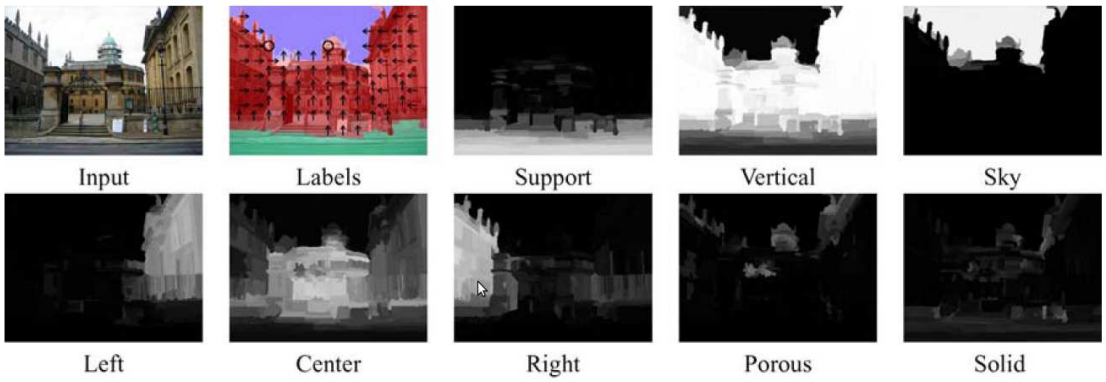


Figure A.2: *Results and assigned probability over the entire image for each class.[86]*

The next step of the block world method is the work described in [13]. The goal is to represent the occupied space with a 3D block relative to the view point. They use a convexity assumption to generate a 3D catalog similar to the vertical subclasses of [86]. The 3D catalog and its 2D projections are shown in A.3. The labeling into these new subclasses can be achieved using the already calculated probabilities and segmentations from [86].

The goal of a block representation is to use physical constraints such as the gravitational force acting on each block and the extent to which a block can support other blocks. To achive this, the density of a block must be determined. Guta et al. uses a classification similar to the one used to label segments. They use three different classes of density: light, medium and high. Trees are light, humans are medium and buildings are high.

The density and occlusion information is used to determine the block relationships which are represented using two attributes: one for the relative depth and one for the mechanical configuration. The relative depth is used as the absolute depth can not be obtained from single image. The depth attribute has three classes; "in-front of", "behind", "no-relationship". The mechanical configuration also has three classes: "supports", "supported by" and "no-relationship".

They use four constraints to reach a qualitative geometric scene understanding:

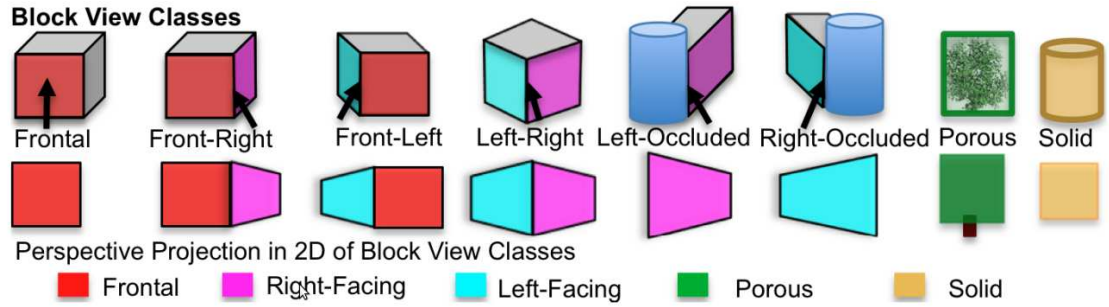


Figure A.3: *Catalog over vertical subclasses in 3D and there 2D projection.[13]*

1. **Static Equilibrium**. Given Newtons's first law, all forces and torques acting on a block must cancel out.
2. **Support Force Constraint**. A supporting object should have enough strength to support the forces of the supported objects.
3. **Volume Constraint**. All the objects in the world must have finite volumes and cannot inter-penetrate each other.
4. **Depth Ordering Constraint**. The depth ordering of the objects with respect to the camera viewpoint should be consistent with the projected regions in the image.

$$C(B_i) = F_{geometry}(g_i) + \sum_{S \in ground,sky} F_{contacts}(g_i, S) + F_{intra}(S_i, g_i, d) + \sum_{j \in bloks} F_{stability}(g_i, S_{ij}, B_j) + F_{depth}(g_i, S_{ij}, D) \quad [\cdot] \quad \text{(A.1)}$$

$C(\cdot)$:	Cost function.	$[\cdot]$
B_i:	Block candidate.	$[\cdot]$
g_i:	estimated block-view class.	$[\cdot]$
S_i:	Region associated with the block.	$[\cdot]$
d:	Estimated density of the block.	$[\cdot]$
S_{ij}:	Represent support relationships.	$[\cdot]$
D:	Represents partial-depth ordering.	$[\cdot]$
$F_{geometry}(\cdot)$:	Agreement of the estimated block view class and surface layout from [86].	$[\cdot]$
$F_{contacts}(\cdot)$:	agreement of geometric properties with ground and sky contact.	$[\cdot]$
$F_{intra}(\cdot)$:	Physical stability within the block.	$[\cdot]$
$F_{stability}(\cdot)$:	Physical stability with respect to other blocks.	$[\cdot]$
$F_{depth}(\cdot)$:	Agreement between the 2D projection of blocks and the image.	$[\cdot]$

Instead of solving the problem as the minimization of one large cost function (see Equation A.1), it is divided into four stages, see Figure A.4. Each of the four stages on the right are described below. The final computed score is then returned to the top-level search procedure and the best block is chosen. When a block is chosen, all segments that are completely or mostly contained by the selected block are removed from bag of segments.

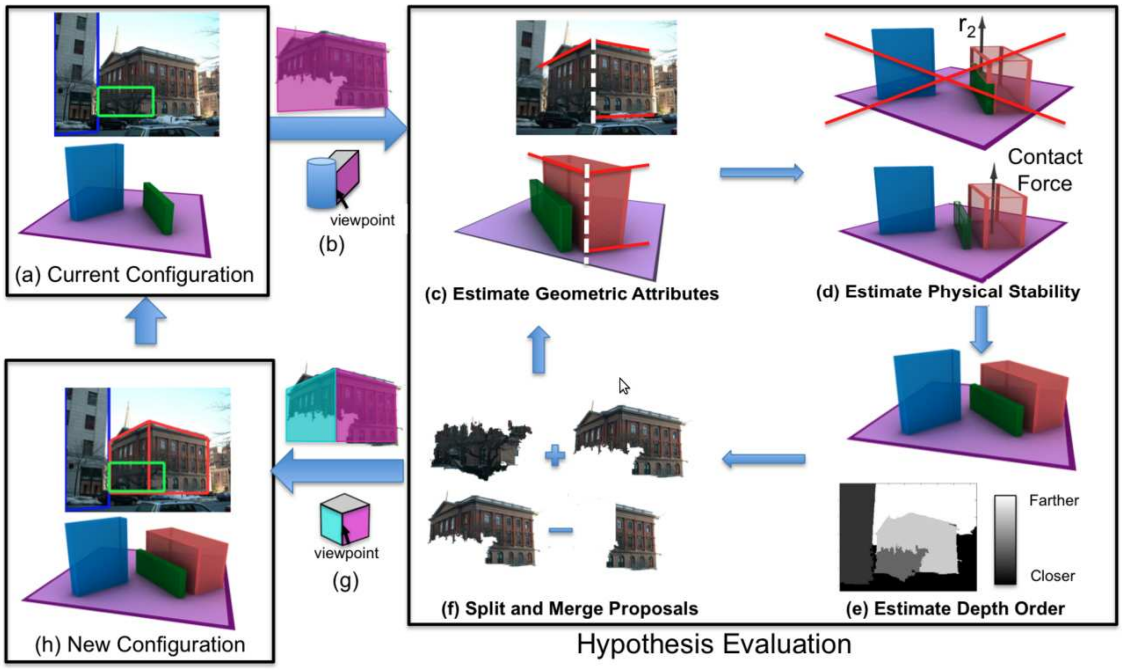


Figure A.4: *Shows the principle of estimating the cost of placing a block at a given position.[13]*

Estimate geometry

The estimation of block geometry cost, Figure A.4c, consists of the block type and location of a fold edge (if any exist) must be determined.

The fold edge location can be determined by introducing a fictive fold edge that divide the segment, see Figure A.5. Moving the fictive edge along the horizontal axe a probability can be calculated for each location. The final estimate of edge locations is chosen as the one with highest probability.

Combined with this is the evidence from ground and sky contact points. For estimating the contact points likelihood term, they use the constraints of perspective projection. A line is fitted to the ground and sky contact point and it is verified if the slope corresponds to the perspective view of the block type.

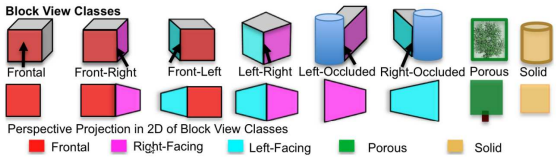


Figure A.5: *Principle of fictive edge (green line). The edge is moved along the horizontal direction.*

Estimating Physical Stability

The physical stability of a block, Figure A.4d, can be divided in to three different terms: internal stability, stability and constraints from Block Strength.

The internal stability is calculated as the change in potential energy. The change is calculated by rotating the segment by a small degree around each face both clockwise and counter-clockwise and use the change in height, δh_s, for each superpixel s in that segment $\mathbf{s_i}$, see Equation A.2. $p(d_s = c)$ is the probability for assigning density class c to superpixel s. This is given from the density classification. The constant m_c is specific to the density class.

$$\Delta P = \sum_{c \in [light, medium, heavy]} \sum_{s \in \mathbf{s_i}} p(d_s = c) m_c \delta h_s \qquad [\cdot] \qquad \text{(A.2)}$$

Where:

ΔP: Change in potential energy.	$[\cdot]$
c: Density class.	$[\cdot]$
$\mathbf{s_i}$: Segment i, vector of superpixels.	$[\cdot]$
s: Superpixel.	$[\cdot]$
m_c: Constant for density class.	$[\cdot]$
δh_s: Change in height for superpixel s.	$[\cdot]$

The stability is calculated from the gravity and the support contact surface for the given block. First the support contact surface is determined, then the torque due to gravitational force for each superpixel and the resultant force on the contact surface.

The constraints from Block Strength are that a lighter density block can not support a heavier block. Therefore the lighter block must occlude the heavier blocks contact points with the supporting ground.

Extracting Depth Constraints

The depth constraints are similar to the constraints from Block Strength. The stage construct of relative depth map are used in the next stage for split and merge. The global depth map is derived from pairwise depth constraints on blocks.

Creating Split and Merge Proposals

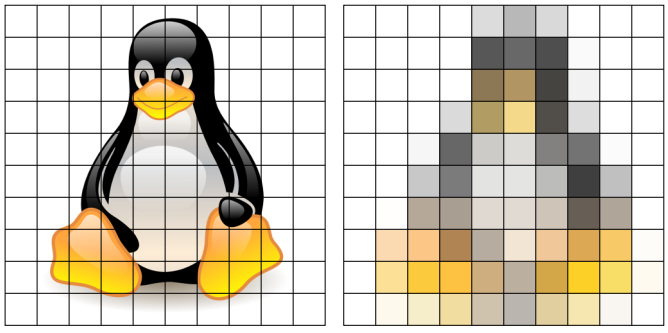
Compared to other methods Gupta et al. introduce a merge and split stage. This stage makes the system more robust to poor segmentations. The merge step is performed if two blocks share the same occluding block, e.g. if there are a tree in front of the house. The split step is performed if the algorithm of [86] with high probability disagree with the inferred geometry in the geometry stage. The new block(s) are compared with the old blocks, with respect to the estimated geometry and physical stability.

Appendix B
Camera

This chapter describes the basic concepts of a camera and parameters that influence the recorded image/video. These concepts create the basis for the Chapter 3,
"Geometry of Planar Scenes" and Appendix E, "Fundamental matrix". First, a basic description of what a camera is will be given. Next, the camera model is derived along with the perspective transformation from 3D to 2D. Last, some of the distortion that can occur on the recorded image data are analyzed.

B.1 What is a camera

Normal cameras work with the light of the visible spectrum where other types of cameras work with other portions of the electromagnetic spectrum, like inferred cameras. The principle stays the same for cameras whether they use visible or non-visible light. The most common camera is an area camera and is the only type handled in this report. Most cameras use an optical lens to focus more light to project it on a two dimensional sensor array, also refereed to as view or image plane, see Figure B.1. It can be seen on Figure B.1 that the light from the scene is quantized into a finite number of cells in the view plane, called pixels. In a digital camera, the cells are constructed with a charge-coupled device (CCD) or CMOS array. The human eye is constructed with a very dense array of cells with photo-reactive molecules on a curved surface called the retina [91].



(a) **(b)**

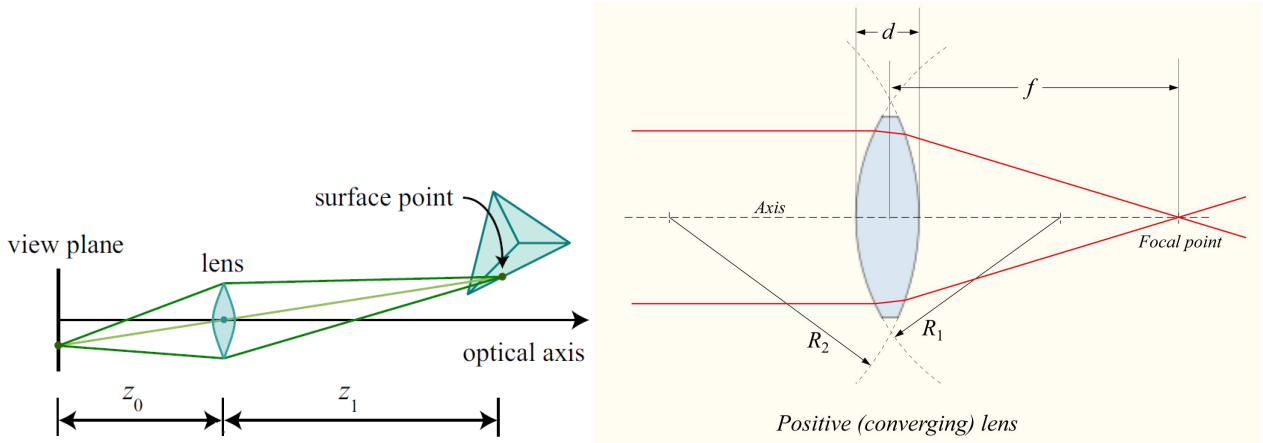
Figure B.1: *(a) Original image projected on the camera sensors. (b) How the image is read by the camera sensors.*

B.2 Model

The goal for a camera model is to mathematically model the basic geometry of projection of 3D points, curves, and surfaces onto a 2D image plane. In this section two different types of camera models are described.

B.2.1 Thin Lens Model

The thin lens model is a simple model for a camera using a lens. As mentioned earlier almost all cameras use a lens to minimize the shutter time by directing more light rays to the view plane, see Figure B.2a. A lens model can be very complex, especially for a compound lens[1]. The thin lens model ignores optical effects due to the thickness of lens, d and simplifies ray tracing calculations. The focal length, f, of a thin lens is given by the Equation B.1 [92]. R_1 and R_2 are the radii of the two surfaces on the lens, see Figure B.2b, and n is the index of refraction of the lens material. More generally, for a thin lens Equation B.2 applies. z_0 is the distance between lens and view plane and z_1 is the distance to the object from the lens.



(a) *Principle of lens, the light rays spread out from the object and the lens directs them to one single point in the view plane.[91]*

(b) *Illustration of thin lens parameters, where R_1 and R_2 are the radii of the two curvatures of the lens.[93]*

Figure B.2: *Principle and parameters of a thin lens*

$$\frac{1}{f} \approx (n-1)\left[\frac{1}{R_1} - \frac{1}{R_2}\right] \qquad [1/\text{mm}] \qquad \text{(B.1)}$$

$$\frac{1}{f} = \frac{1}{z_0} + \frac{1}{z_1} \qquad [1/\text{mm}] \qquad \text{(B.2)}$$

Where:

f:	Focal length	[mm]
n:	Index of refraction of the lens material.	[·]
R_1, R_2:	Radii of curvature of the two surfaces.	[mm]
z_0:	Distance from lens to view plane.	[mm]
z_1:	Distance from lens to object.	[mm]

[1] A compound lens is an array of simple lenses with a common axis. This allows more optical aberrations to be corrected than is possible with a single lens

B.2.2 Pinhole Camera Model

A pinhole camera is a simplification of the thin-lens camera and widely used in computer vision. In a pinhole camera all rays intersect in the same point (nodal point), see Figure B.3. In comparison to the thin-lens model, where light emitted in different directions from the object are focused into one point in the view plane. The pin hole model only accounts for the direct ray.

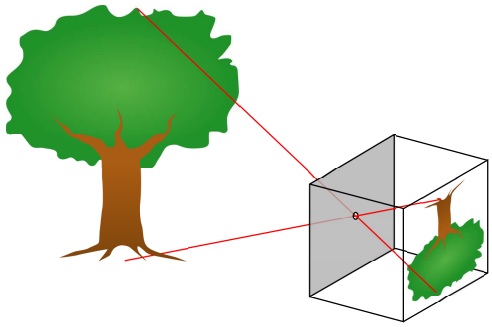


Figure B.3: *Principle of a pinhole camera where all rays intersect in the same point (nodal point). [93]*

B.2.3 Orthography

An orthographic is a simple model where the effect of the distance to the object is not modeled. This basically means that the z component of the coordinate $\mathbf{p} = (x, y, z)$ is dropped. Equation B.3 expresses the orthographic projection using the homogeneous representation. The orthographic projection model approximates the model for long focal length lenses and objects whose depth is shallow relative to their distance to the camera [94].

In practices the real world coordinate of $\mathbf{p}$ are measured in meters where the view plane point $\mathbf{x}$ are measured in mm (ultimately measured in pixels). For this reason a scale is applied to the orthographic model. The scaling can be different from object to object if they are modeled independently, and from frame to frames for objects approaching the camera. The scaled orthographic projection is a popular model for reconstructing the 3D shape of objects far away from the camera, as it simplifies certain computations [6].

$$\widetilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \widetilde{\mathbf{p}} \qquad [\text{m}] \tag{B.3}$$

Where:

$\widetilde{\mathbf{x}} = [x\ y\ 1]^T$: Homogeneous coordinate for 2D view plane point $\mathbf{x}$. [px]

$\widetilde{\mathbf{p}} = [x\ y\ z\ 1]^T$: Homogeneous coordinate for 3D real world point $\mathbf{p}$. [m]

B.2.4 Perspective

The most used projective transformation is the true 3D perspective transformation [6]. This projection divides the x and y by the z component. A perspective projection can be represented with homogeneous coordinates by P which is a 3×4 camera matrix, see Equation B.4. After projection, it is no longer possible to recover the distance of the 3D point from the image [6].

$$\widetilde{\mathbf{x}} = P\widetilde{\mathbf{p}} \qquad [\cdot] \tag{B.4}$$

Camera calibration

The camera matrix is used to represent the perspective transformation for homogeneous coordinates. The camera matrix is a 3×4 matrix with 11 DoF. The camera matrix can be broken down into two groups of parameters: internal (5 DoF) and external (6 DoF) [15]. The internal parameters are expressed as the calibration matrix K and the external parameters as camera 3D rotation R_C and 3D translation $\mathbf{t_c}$ in the world. The camera matrix is defined in Equation B.5.

Camera calibration was first studied in photogrammetry by Brown in 1971[95]. Camera calibration has since been a widely studied topic both in photogrammetry and computer vision. Among the methods that require some known object such as a calibration board are [96, 97].

The methods which do not require any prior knowledge are known as self-calibration[98, 15]. The camera self-calibration is computed in two steps. In the first step, the epipolar transformation (Fundamental matrix) is found, see Chapter E. The second step is to solve the Kruppa equations which requires three or more movements of the camera. Kruppa's equations link the fundamental matrix to the image w of the absolute conic U, where conic w determines the camera calibration. The absolute conic U is invariant under rigid motions which ensures that w is independent of the camera position and orientation.

One special case of self-calibration uses the assumption of pure rotational motion[99, 100, 101]. This case uses overlapping images recorded from rotation around the cameras nodal point. A very accurate estimate of the focal length can be obtained using a $360deg$ motion. Here the accuracy will be proportional to the total number of pixels in the panorama image.

Homographies (see Chapter 3) have previously been used with success to generate panorama images from multiple images[102, 103, 6]. The resulting panorama image might suffer from differences in exposure and small misalignments, such as Figure B.4. Burt and Adelson proposed the use of Multi-band Blending in 1983[104] which preserves the details, see Figure B.5.

Figure B.4: *Panorama image with differences in exposure and small misalignments.[102]*

Figure B.5: *Panorama after multi-band blending.[104]*

Homographies have also been used by Zhang in [105] and Chuan et. al. in [106] to estimate

the calibration matrix. They both use the relationship between homography, calibration matrix and 3D pose described as the perspective transformation, Equation B.4.

The calibration matrix is defined as an upper triangular matrix, see Equation B.6. Here α_x and α_y represent the focal lengths in the x and y directions respectively, $(x_0,\ y_0)$ is the principal point on view plane and s is a skew parameter. A few assumptions can be made to reduce the number of DoF to 7. Assuming that the pixels are square will result in $\alpha_x = \alpha_y = \alpha$. The skew parameter can in most cases be negligible [15]. Setting the principal point to the center of the image, e.g., $(x_0,\ y_0) = (W/2,\ H/2)$, where W and H are the image height and width, can result in an usable calibration matrix with a single unknown, i.e., the focal length f [6].

$$P = K \left[\begin{array}{c|c} R_C & \mathbf{t_c} \end{array} \right] \qquad [\cdot] \qquad \text{(B.5)}$$

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad [\cdot] \qquad \text{(B.6)}$$

Where

K: Camera calibration matrix	$[\cdot]$
R_C: Camera 3D rotation	$[\cdot]$
$\mathbf{t_c}$: Camera 3D translation	$[\cdot]$
$\alpha_x,\ \alpha_y$: Focal in x and y direction	[px/m]
$x_0,\ y_0$: Principal point	[px]
s: Skew parameter	$[\cdot]$

B.3 Distortion

There are many types and sources of distortion in image recording. In this section the two most influential types are described: noise and blur. The relationship between the perfectly clear image and the observed image is described with Equation B.7[107, 6]. In chapter 6, "Pre-processing", some of the different methods used to remove noise and deblur images are described.

$$o(x, y) = spf(x, y) * s(x, y) + n(x, y) \qquad [\cdot] \qquad \text{(B.7)}$$

Where

$o(x, y)$: Observed image	$[\cdot]$
$s(x, y)$: Perfect clear image	$[\cdot]$
$spf(x, y)$: Point spread function (SPF)	$[\cdot]$
$n(x, y)$: Noise model	$[\cdot]$

B.3.1 Noise

Image noise is a random variation in color and/or brightness, caused by the image sensor. The noise is introduced in different steps of the image capture process. Some of the different types of noise are: Amplifier noise, Salt-and-pepper noise, Shot noise, Quantization noise, Non-isotropic noise[6, 108, 109, 110, 111].

- Amplifier noise can be modeled as an additive Gaussian model. It is independent of pixel location and intensity. The amplification noise can be different for each color channel as the amplification varies[6].
- Salt and pepper noise is an impulsive noise. It can be seen as bright pixels in dark areas and vice versa. This type of noise can be caused by dead pixels, bit errors in transmission, analog-to-digital converter errors, etc. Salt and pepper noise can be reduced by interpolating around dark/bright pixels[111].
- Shot noise is dominant in brighter areas of the image as there can be variation in the amount of light sensed. This noise is also known as "photon shot noise"[110].
- Quantiation noise is introduced when the analog current is quantized into discrete levels. It can be modeled as an uniform distribution and can signal dependent[111].
- Non-isotropic is a noise that can have a significant orientation, i.e. row noise or column noise.

The final amount of noise is an image depending on the incoming light, shutter time, and sensor gain. Image noise is most commonly modeled with a Gaussian model but in low light condition a Poisson model may be more appropriate[6]. The noise is added to the perfect image to generate the observed image, see Equation B.7.

Noise estimation has been a popular research topic for many years as it is often desired to detect the amount of noise present in an image or signal. It is often assumed that the image noise can be modeled as an Gaussian distribution with zero mean and standard deviation (std) σ[6]. Thus, the amount of noise can be represented by the std σ.

Given video data the different methods of estimation noise can be divided into three different groups: Spatial, Temporal and Spatial-Temporal. The most explored approach is the spatial[63, 64, 112, 113, 114, 115], followed by the spatial-temporal[116, 117, 118].

- Spatial video denoising methods is when image noise reduction is applied to each frame individually.
- Temporal video denoising methods is when noise between frames is reduced. Motion compensation may be used to avoid ghosting artifacts when blending together pixels from several frames.
- Spatial-Temporal video denoising methods use a combination of spatial and temporal denoising.

Liu, Szeliski, Kang et al. [119] estimates the noise level function (NLF) which predicts the overall noise variance at a given pixel and the brightness of the pixel. An alternative procedure to estimate image noise is to calculate the variance of each pixel, given multiple frames of the same static scene.

If multiple frames of a static scene can not be obtained the noise can be estimated as the variance in regions of near-constant value[119].

B.3.2 Blur

Blur is modeled as a kernel convolution between the perfect image and a point spread function (SPF)[6], see Equation B.8. Automatic blur detection is highly desirable due to the high computational cost of deblurring. The goal is to determine whether a given image is blurred and to determine to what extent the image is blurred. Given this information a decision on whether image needs deblurring or not can be made.

$$o(x,y) = spf(x,y) * s(x,y) \qquad [\cdot] \qquad \text{(B.8)}$$

The blur detection methods can be divided into indirect and direct methods[56]. Indirect methods rely on the estimation of the blur function $spf(\cdot)$, which is closely related to the deblurring process. Consequently, it is also associated with a very high computational, which is undesired.

The direct methods determines blur by examining some discriminative features, i.e. edges. Edges are commonly used as both the edge type. Its sharpness will change when exposed to blur. Tong et al. propose in [56] to use Haar wavelet transform (HWT) to determine to what extent an image is blurred. The method is based on an edge sharpness analysis. HWT is used to detect and determine different types of edges.

Other direct methods use the discrete cosines transform (DCT) to determine blur[120, 121, 122]. DCT is desired to use because it is a bi-product of JPEG and MPEG compression. The DCT approaches rely on frequency analysis since blurred images will contain less high-frequency. DCT is usually performed in a block processing manner and therefore the DCT detection methods are able to detect locally blurred images.

Crete et al. in [58] argue that the edge based methods are sensitive to the threshold choice to detect and classify edges and the presence of noise which can mislead the edge detection. As a solution they propose a method where blur is detected using different levels of blur on the same picture.

Different types of blur exist. The two main types motion and lens blur are described in the following sections.

MotionBlur

Motion blur occurs closely combined to the shutter time. The shutter time controls the amount of light let into the cells and thereby records a singe frame. If the shutter time is too large the scene might change duing recording. The changes can be objects moving and/or camera movement. Changes result in multiple world point being mapped to one image point. Motion blur coursed by camera movement will usually affect the whole image[15] if the scene is static. In a dynamic scene it can be impossible to focus on both the moving object and the static background, see Figure B.6. The most important fact is that motion blur can change from frame to frame in a sequence and make post processing difficult.

(a) *Effect of motion blur on background.*

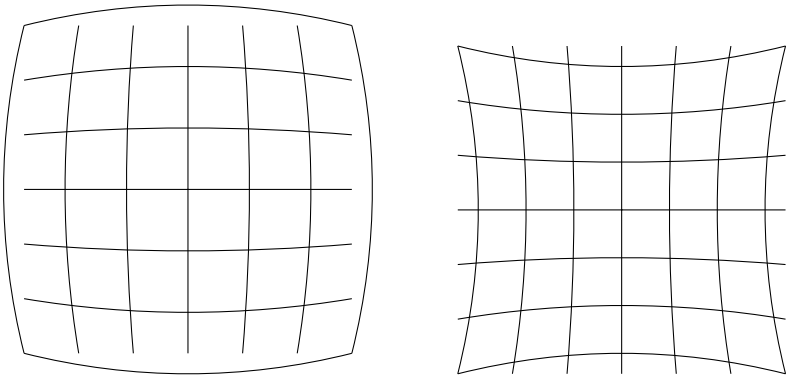
(b) *Effect of motion blur on moving objects.*

Figure B.6: *Two cases of different motion blur*

Lens distortions

Lens distortions depend on multiple factors, such as lens blur and radial distortion. Lens blur can occur when the image is out of focus e.g. one world point is projected to multiple pixels.

The radial distortion can be divided into to different types, Barrel and Pincushion distortion. In "barrel distortion", image magnification decreases with distance from the principal point, see Figure B.7a. In "pincushion distortion", image magnification increases with the distance from the principal point, see Figure B.7b.



(a) *Barrel distortion.*

(b) *Pincushion distortion.*

Figure B.7: *Illustration of barrel and pincushion distortion on a square image*

B.4 Summery

This chapter describes basic concepts of a camera. First, the transformation from world point to image coordinate is derived for different camera models. The simpler pinhole model is the most used camera model in the literature. The pinhole model modes the ideal camera where each world point is related with the image through one ray starting in the nodal point.

Two different transformations, Orthography and Perspective, from world point to image coordinate is analyzed. Orthography is a simpler model where the distance to the world point is disregarded. The projective transformation is widely used, and is described by a camera matrix. A camera matrix can be divided into intrinsic and external parameters. The intrinsic parameters must be determined through calibration but given these it is possible to recover the external parameters given multiple images.

Furthermore, different types of image distortion have been analyzed. The image distortions can in general be described with two types, noise and blur. Noise can occur from different internal processes in the camera such as quantization. Blur is dependent on motion and shutter speed. Given fast motions a high low shutter speed must be used to prevent blur. A high quality image contains low amount of both noise and blur.

The next chapter describes the benefits and uses planar surfaces in multiple view geometry. The camera model and the projection from 3D to 2D is a crucial part.

Appendix C
2D Geometric transformations

This appendix presents some different 2D transformations that in combination can lead to the homography. All these transformations are expressed as a 3×3 matrix. The transformations are discussed in detail in [15]. Throughout this appendix $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{x}'}$ will be the 2D point (x, y) represented as a 3D vector (Z_x, Z_y, Z) such that $x = \frac{Z_x}{Z}$ and $y = \frac{Z_y}{Z}$. This representation is called a homogeneous representation of a point and it lies on the projective plane P^2 [123]. x will be the point in the original image, $\widetilde{\mathbf{x}} = [x\ y\ 1]^T$ and $\widetilde{\mathbf{x}'}$ will be the transformed point. In the different sections the transform is applied to the image in Figure C.1

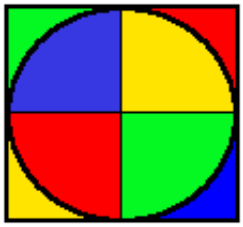


Figure C.1: *Original image used for transformations.*

C.1 Isometry

An Isometry transformation preserves the euclidean relationship between points. The distance between two points in the original and the wrapped image is the same. The same applies for angles between lines and the area of surfaces, see Figure C.2. The Isometry transformation is composed by a 2D rotation (1 dof) and a 2D translation (2 dof) see Formula C.1.

$$\widetilde{\mathbf{x}'} = \begin{bmatrix} R(\theta) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \widetilde{\mathbf{x}} \qquad [\cdot] \qquad \text{(C.1)}$$

Where:

$R(\theta)$: 2×2 Rotation matrix	$[\cdot]$
$\mathbf{t}$: Translation column 2-vector	$[\cdot]$
$\mathbf{0}$: Row of 2 zeros	$[\cdot]$

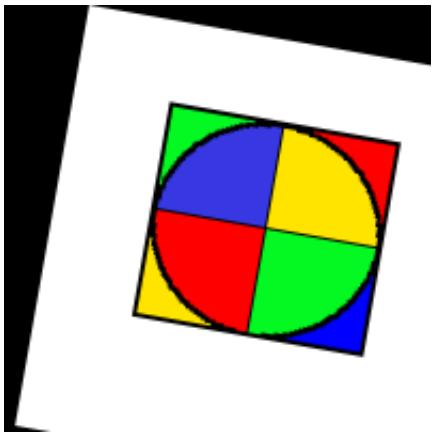


Figure C.2: *Isometry transformation of original image with* $\theta = 10deg$ *and* $t = [40\ 0]^T$.

C.2 Similarity transformation

A Similarity transform is an extension of the Isometry with an isotropic scaling s. Isotropic means that the scaling is invariant with respect to direction. The Similarity transform will not change the shape of an object only scale it, see Figure C.3. Therefore the transformation is invariant with respect to angles and distance ratios, as any scale will cancel out in the ratio. The similarity transformation only affects the actual distance between points. With the isotropic scale the similarity transformation has four dof overall, see Formula C.2.

$$\widetilde{\mathbf{x}}' = \begin{bmatrix} sR(\theta) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \widetilde{\mathbf{x}} \qquad [\cdot] \qquad \text{(C.2)}$$

Where:

s: Isotropic scaling $[\cdot]$

C.3 Affine transformation

An affine transformation is an extension of the similarity transform. In general, an affine transformation is composed of linear transformations (rotation, scaling or shear) and a translation. The extension lies in adding another rotation and scale, such that it consists of two rotations and two non-isotropic scalings, see Formula C.4. A is a 2×2 non-singular matrix. This extension adds two dof. One dof is for the ratio of the scaling parameters and the other dof is for the angle specifying the scaling direction. The affine transformation therefore consists of six dof, see Formula C.3. The affine transformation does not preserve any relationship between angels, point distances and distance ratios. As the affine transformation does not model the perspective changes, parallel lines will remain parallel and the length ratios of these are preserved in the wrapped image, see Figure C.4.

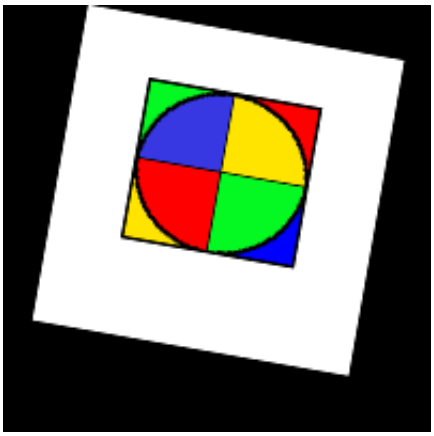


Figure C.3: *Similarity transformation of original image with* $\theta = 10deg$, $t = [40\ 0]^T$ *and* $s = 0.75$.

$$\widetilde{\mathbf{x}}' = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \widetilde{\mathbf{x}} \qquad [\cdot] \qquad \text{(C.3)}$$

$$A = R(\theta)\, R(-\phi)\, DR(\phi) \qquad [\cdot] \qquad \text{(C.4)}$$

Where:

$R(\theta), R(\phi)$: 2×2 Rotation matrices [·]

$D = diag([\lambda_1\ \lambda_2])$: Diagonal matrix with scalings values [·]

The matrix A is thus a concatenation of the rotation by ϕ then scaled in x,y direction by λ_1, λ_2 then rotated back by $-\phi$ and at last rotated by θ.

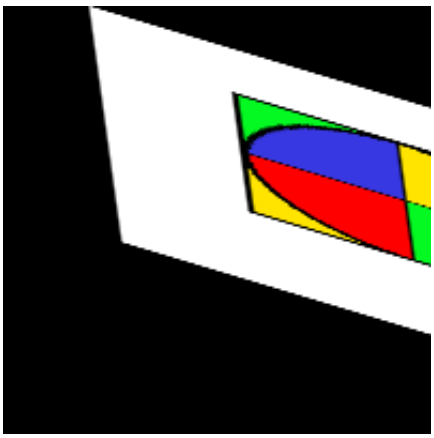


Figure C.4: *Affine transformation of original image with* $\theta = 10deg$, $\phi = -10$, $\lambda_1 = 1.5$, $\lambda_2 = 0.5$ *and* $t = [40\ 0]^T$.

C.4 Projective transformation

A projective transformation is a non-singular linear transformation of homogeneous coordinates. The projective transformations is also the homography. The projective transformation will not

be linear if the coordinates are inhomogeneous, which is the reason for using homogeneous coordinates. The projective transformation is an extension to the affine transformation with the vector $\mathbf{v} = [v_1\ v_2]$ which add additionally two dof, see Formula C.5. In the affine case $\mathbf{v}$ is 0, as this vector is responsible for the non-linear effects of the projectivity. The addition of a non-zero $\mathbf{v}$ makes the scaling from A on points variant to the position of the point in the image. $\mathbf{v}$ also makes the orientation of a line variant to the position in the image. Based on this, none of the invariants from the affine transformation mentioned above hold for the projective transformation, though it still holds that collinear points will stay collinear in the wrapped image, see Figure C.5.

$$\widetilde{\mathbf{x}}' = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{v} & 1 \end{bmatrix} \widetilde{\mathbf{x}} \qquad [\cdot] \qquad \text{(C.5)}$$

Where:

$\mathbf{v}$: Projective row 2 vector $[\cdot]$

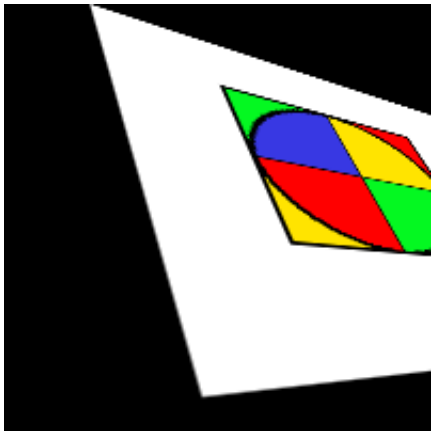


Figure C.5: *Projective transformation of original image with* $\theta = 10deg$, $\phi = -10$, $\lambda_1 = 1.5$, $\lambda_2 = 0.5$ $t = [40\ 0]^T$ *and* $v = [0.003\ \ -0.002]$.

Appendix D
Cost functions

This appendix describes some different cost functions which can be used to obtain a homography estimate. Generally cost functions are used in optimization problems where one wishes to choose the best element from a set of available alternatives. This means solving problems in which one seeks to minimize or maximize a real function (cost function) by choosing the values of variables from within an allowed set. These cost functions are used to find the optimal homography when four or more point correspondence exist between two images.

D.1 Algebraic distance

A simple cost function is the algebraic distance, where the norm $||Ax||$ is minimized with respect to x. The simplest solution is $x = 0$, which is useless, and therefore the constraint $||x|| = 1$ (this 1 is selected arbitrarily) is introduced. The solution to this optimization problem can be found using the Singular Value Decomposition (SVD) analysis of the matrix A. The solution is the solution to the unit singular vector corresponding to the smallest singular value of A. Although, this is a simple and computationally cheap cost function is the quantity being minimized not geometrically/statistically meaningful. It can be improved with the use of normalization, see Section 3.3.1.

D.2 Geometric error

The geometric error also called the transfer function, measures the Euclidean distance between the estimated output and the true output, see Figure D.1. For the homography it measures the distance between the wrapped point and the originally found position in the second image. This distance is also called the transfer error, and can be written as Equation D.1 (for the case with errors only in the second image), where $p_i \leftrightarrow p'_i$ is the point correspondence for point i.

$$\sum_i d\left(p'_i, Hp_i\right)^2 \qquad \left[\text{px}^2\right] \tag{D.1}$$

Where:

p_i: Homogeneous coordinate for point i in first image	[px]
p'_i: Homogeneous coordinate for point i in second image	[px]
H: 3×3 Homography	[·]
$D(\cdot,\cdot)$: The Euclidean image distance between two points	[px]

In the case where both images contain errors the symmetric transfer error and is used. The symmetric transfer error uses both the forward (H) and backward (H^{-1}) transformations, see

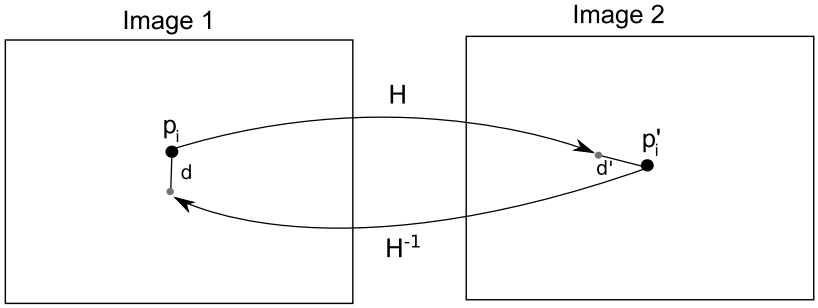


Figure D.1: *Illustration of geometric error, d and d' is the geometric distance between the true points p_i and p'_i and the transformed points $H^{-1}p'_i$ and Hp_i.*

D.2.

$$\sum_i d\left(p'_i, Hp_i\right)^2 + d\left(p_i, H^{-1}p'_i\right)^2 \qquad \left[\text{px}^2\right] \tag{D.2}$$

To minimize Equation D.2, an iterative approach is required. The iterative approaches are usually more accurate than linear algorithms, but it comes with some disadvantages. The disadvantages are that they are slower, risk not converging and present additional problems such as picking initial estimates, step size and stopping criteria.

D.3 Reprojection error

The reprojection error is a geometric error between a projected point and a measured one, see Figure D.2. The reason this cost function is called reprojective, is because it is analogous to estimate the real world point P_i and from the originally found correspondence $p_i \leftrightarrow p'_i$ and then reprojected it to estimate a better matched correspondence $\hat{p}_i \leftrightarrow \hat{p}_i{}'$.

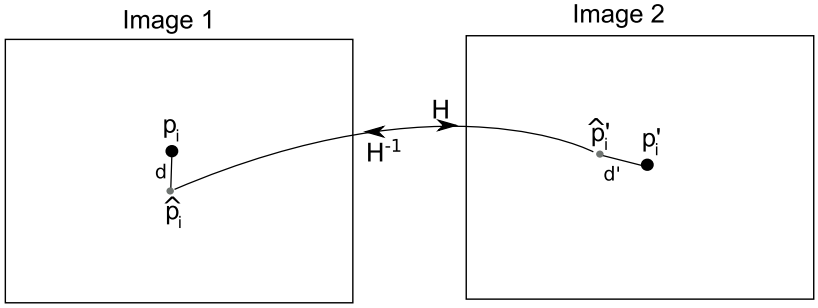


Figure D.2: *Illustration of reprojection error, d and d' is the geometric distance between the true points p_i and p'_i and the estimated corrected points $\hat{p}_i$ and $\hat{p}_i{}'$.*

The cost function not only estimates the homography, but also a correction to the points. The cost function is shown in Equation D.3, which is minimized with respect to $\hat{p}_i$, $\hat{p}_i{}'$ and $\hat{H}$, and is subject to $\hat{p}_i{}' = \hat{H}\hat{p}_i$.

$$\sum_i d\left(p_i, \hat{H}\hat{p}_i\right)^2 + d\left(p_i, \hat{H}^{-1}\hat{p}_i\right)^2 \qquad \left[\text{px}^2\right] \tag{D.3}$$

Where:

$\hat{p_i}$: Corrected homogeneous coordinate for point i in first image [px]
$\hat{p_i}'$: Corrected homogeneous coordinate for point i in second image [px]
$\hat{H}$: 3×3 Homography [·]

D.4 Sampson error

The point correspondence $p_i \leftrightarrow p_i'$ can be formulated as a point in 4D; $G_i = (x_i, y_i, x_i', y_i')$. In the optimal case where all points are perfect the algebraic variety ν_H will pass through these points. A algebraic variety is the set of solutions of a system of polynomial equations. In this case ν_H is a curve in 4D. As the point correspondence is not perfect the reprojective error can be formulated distance between point G_i and the closest point on the variety ν_H $\hat{G}_i$, see Equation D.4 and Figure D.3. The point $\hat{G}_i$ contains the same point correspondence as the corrected points for reprojection.

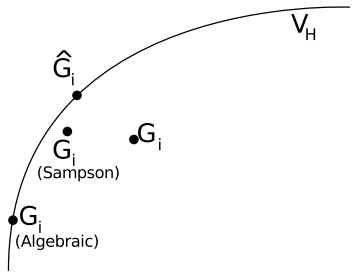


Figure D.3: *The variety ν and points where different errors of the measured noisy point G_i with respect to homography H are reached. The geometric error is reached at $\hat{G}_i$, Sampson's error in G_i (Sampson), and the algebraic error in G_i (Algebraic) [124]*

$$\sum_i ||G_i - \hat{G}_i||^2 \qquad [\text{px}^2] \tag{D.4}$$

G_i: Point correspondence $p_i \leftrightarrow p_i'$ in 4D. [px]
$\hat{G}_i$: Corrected point correspondence in 4D. [px]

(D.5)

In [15] it is mentioned that the point $\hat{G}_i$ can not be estimated directly except via iteration, because of the non-linear nature of the variety ν_H. The Sampson error is a 1 order Taylor approximation to $\hat{G}_i$, and was first used by Sampson in [125] for conics. The derivation of Sampson's error for homographies is described in [15].

Appendix E
Fundamental matrix

Epipolar geometry is the cameras projective geometry between two views, e.g. depending on the intrinsic parameters and the relative pose of the camera. This implies that the epipolar geometry is independent of the actual scene structure. The fundamental matrix F is the algebraic representation of epipolar geometry.

First the basics of epipolar geometry is described in Section E.1. Then, in Section E.2 the fundamental matrix is derived. Finally the differences and relation between homography and the fundamental matrix is described in Section 3.2.

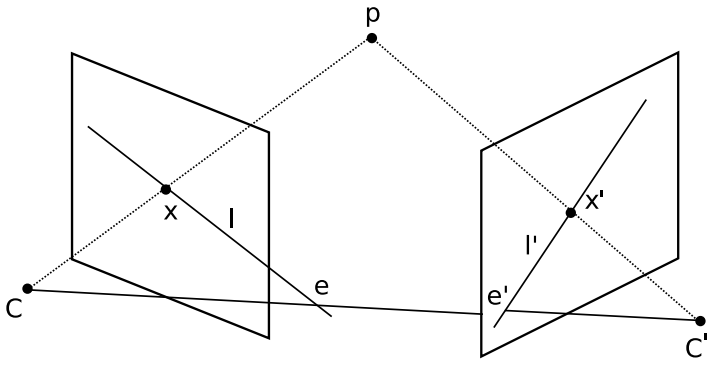


Figure E.1: *Illustration of epipolar geometry, where p is a 3D point in the scene, x and x' is the projection into two images, with projective center C and C'.*

E.1 Epipolar geometry

Epipolar geometry is often used in the search of image correspondents between images. Epipolar geometry is the geometry of the intersection of the image plane with the pencil of planes (epipolar planes) that intersects the baseline C-C'[15]. Figure E.2 shows the case where the cameras are perfectly aligned, i.e. a stereo camera as Bumblebee2 from Point Grey[127]. It can be seen that the search for point correspondence only needs to be done along the horizontal direction. Figure E.1 shows the case where the cameras are not aligned and therefore are independent. The three points C, C' and p' span the epipolar plane that intersect the image planes in the epipolar lines l and l'. It can still be seen that the search for point correspondence only needs to be performed along the epipolar line.

- **Epipole**, is the intersection between the baseline, and the image plane.

[1] A pencil of planes is, the family of planes through a given straight line[126].

- **Epipolar plane**, is the pencil of planes that intersect with the baseline.
- **Epipolar line**, is the intersection between epipolar plane and the image plane. The correspondence between epipolar line in the two images is defined by the epipolar plane. All Epipolar lines intersect in epipole of the image.

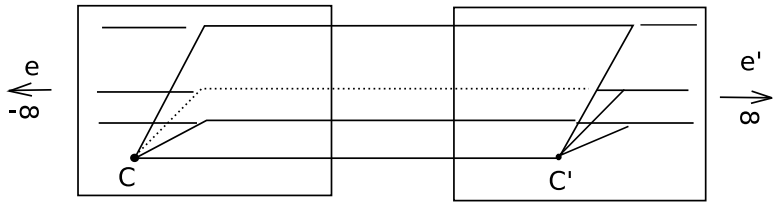


Figure E.2: *Epipolar geometry of stereo camera. Three planes intersection the baseline $C - C'$ and the image plane are shown.*

E.2 What is the fundamental matrix

As mentioned in the introduction to this chapter, the fundamental matrix is the algebraic representation of epipolar geometry. In Figure E.1 it can be seen that the epipolar line l' between e' and x' is the projection of the line going through C and x. Therefore, the point corresponds to x is located on l'. The mapping $x \rightarrow l'$ is defined as the fundamental matrix which leads to Equation E.1[98].

$$l' = F\widetilde{\mathbf{x}} \qquad [\cdot] \quad \text{(E.1)}$$

Given the point x_π on the plane π and its projections x and x' on the image plane, see Figure E.3, the plane must not intersect the baseline, as this would make it invisible on the images. Homography, Equation 3.1, $x' = H_\pi x'$, describes the mapping between corresponding points on the plane π. The epipolar geometry defines the epipolar line as $l' = e' \times x'$. Given these two equation the fundamental matrix is defined as Equation E.2. The cross product between e' and x' can be expressed as a matrix multiplication $|e|_\times x$ where the notation of $|e|_\times$ is given in Equation E.3.

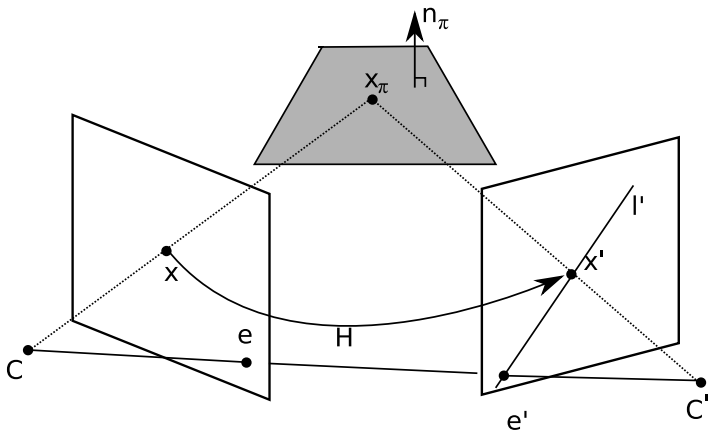


Figure E.3: *Illustration of epipolar geometry and homography using a planar surface.*

$$l' = e' \times H\widetilde{\mathbf{x}} = |e'|_{\times} H\widetilde{\mathbf{x}} = Fx \quad [\cdot] \quad \text{(E.2)}$$

$$F = |e'|_{\times} H \quad [\cdot]$$

$$|e'|_{\times} = \begin{bmatrix} 0 & w & -v \\ -w & 0 & u \\ v & -u & 0 \end{bmatrix} \quad [\cdot] \quad \text{(E.3)}$$

where:

$e' = [u\ v\ w]$: Epipole image one. $[\cdot]$

H_{π}: Homography for plane π. $[\cdot]$

l': Epipolar line in image two. $[\cdot]$

If point x' lines on the line l' then $0 = \widetilde{\mathbf{x}'}^T l'$ must be true. From Equation E.1 it is shown that l' can be replaced with $F\widetilde{\mathbf{x}}$, such that $\widetilde{\mathbf{x}'}^T F\widetilde{\mathbf{x}} = 0$. This, is valid for all points where a correspondence exist in two images. It also implies that the fundamental matrix can be determined only from corresponding points in two uncalibrated cameras[128].

Appendix F
SIFT

This appendix is based on the article "Distinctive Image Features from Scale-Invariant Keypoints" published by David Lowe [68]. The SIFT method was published by David Lowe from the University of British Columbia in 1999. SIFT features are invariant to scale, rotation, affine distortion and they are partially invariant to illumination changes.

F.1 Obtain feature points

The procedure for obtaining SIFT features can be divided into different stages. These stages are described in the following subsections:

- Detection of scale-space extrema
  To detect stable feature points, the local extremas of "scale-spaces" are detected and used as feature points.
- Feature point localization and orientation assignment
  The location of the found feature points in the previous stage are discrete values. In this stage, more exact locations are approximated. Also, local orientations are assigned to the feature points to make them invariant to rotation. Only the stable feature points are wanted, and therefore, the unstable feature points are rejected based on thresholds.
- Descriptor
  The features used to describe the individual points, and thereby used for recognition, are extracted in this stage.

F.1.1 Detection of scale-space extrema

To extract a stable feature point in an image, Lowe has proposed the use of local extrema in differences of Gaussian (DoG). A scale space is a space consisting of images of different scales. The scale-space has to be computed, as there is only one image, and not images of different scales. A scale-space can be computed by convoluting the original image with a Gaussian kernel, see formula F.1, where the kernel is defined as in formula F.2.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad [\cdot] \quad \text{(F.1)}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}} \quad [\cdot] \quad \text{(F.2)}$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \quad [\cdot] \quad \text{(F.3)}$$

where:

x, y: Image coordinates.	$[\cdot]$
σ: Standard deviation.	$[\cdot]$
k: Constant depending on the number of intervals in an octave.	$[\cdot]$

An approach of DoG is described by formula F.3 and Figure F.1. The scale-space is divided into octaves (by doubling σ), and each of these octaves are then divided into a number of intervals s. The value of k in formula F.3 is computed as $k = 2^{\frac{1}{s}}$

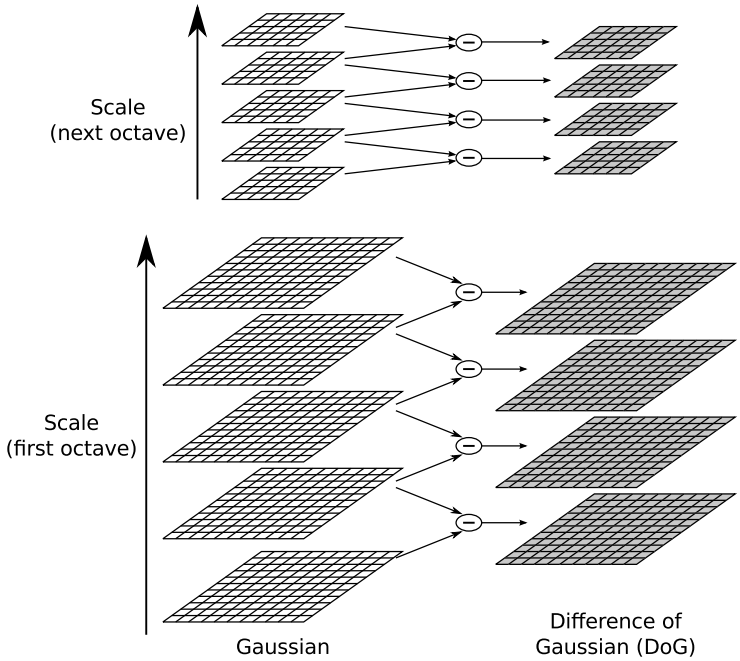


Figure F.1: *The Figure illustrates difference of Gaussian. On the left, the scale-space of images is shown and on the right, the DoG.*

To locate a local extrema, all the points in the neighborhood in the current scale, the scale above and the scale below are examined, see Figure F.2. This also implies that $s + 2$ different scaled images per octave has to be computed. Lowe has examined the effect of s compared to repeatability and the number of feature points in a number of typical images. The result showed that the repeatability is roughly the same for $s > 1$ where it is around 80%. The number of feature points increase as the value of s increases until $s = 3$, where it starts decreasing. Around $s = 2$, the number of feature points is approximately 1750. It is chosen to set $s = 2$, as a higher s will increase the computational complexity and thereby increase the execution time.

F.1.2 Feature point localization and orientation

The location and scale found for the feature point in the previous section is a discrete. To find the exact values for the extremas, a detailed model is fit to approximate the location and scale. The feature points are also assigned a local orientation to make the landmark invariant to rotation. In this section, the unstable feature points are rejected based on two thresholds.

The model used to approximate the location is a 3D quadratic function, which is implemented as a Taylor expansion (up to the quadratic terms) of the scale-space function, see formula F.4.

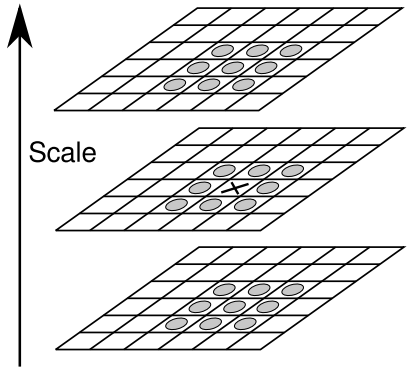


Figure F.2: *An extrema is found by checking all the pixels (green dots) around the sample point (X). The sample point is an extrema if all neighbour pixel values are higher or lower than the sample pixel value in the current, in the above and in the scale below. That is, all the 26 pixels marked by green dots must be higher or lower than the sample point.*

The exact location for the feature point can be found as the point where the derivative of formula F.4 is zero. By solving it with respect to X, an expression of the exact location is found, see formula F.5.

$$D(X) = D + \frac{\partial D^T}{\partial X}X + \frac{1}{2}X^T\frac{\partial^2 D}{\partial X^2}X \quad [\cdot] \quad \text{(F.4)}$$

$$\hat{X} = -\frac{\partial^2 D^{-1}}{\partial X^2}\frac{\partial D}{\partial X} \quad [\cdot] \quad \text{(F.5)}$$

where:

$$X = (x, y, \sigma)^T \quad [\cdot]$$

Feature points with low contrasts are unstable. Formula F.4 can also be used to reject feature points with low contrasts. Figure F.3 shows the impact on the number of feature points that the threshold for $|D(\hat{X})|$ has. It can be seen that the number of feature points decrease rapidly as the threshold increases. This indicates that there are many unstable feature points. Around 0.007, the amount of feature point reaches 1000 and the decrease slows down, which indicates that the remaining feature points are more stable. To ensure stable feature point, a threshold for $|D(\hat{X})|$ is set to 0.02. In Figure F.4a, the feature points are plotted for no threshold and on Figure F.4b, the feature point for a threshold of 0.02 are shown.

It can be seen on Figure F.4b that DoG has a large response around edges. These feature points are unstable and therefore have to be rejected. This can be done by examining a 2×2 Hessian matrix. An unstable feature point along an edge will have a large principal curvature along the edge and a small one perpendicular to the edge. Formula F.6 is used to reject the feature point where the ratio between the two principal curvatures are greater than the threshold on the right side. Figure F.5 shows that the number of feature points stabilize around a threshold at 10. The threshold is set to 5 as this will reject about 15% of the feature point. The rejected feature point is influenced by a near edge. Figure F.4a shows the result without a threshold on the ratio, where Figure F.6 shows the impact, a threshold where $r = 5$ has.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \quad [\cdot] \quad \text{(F.6)}$$

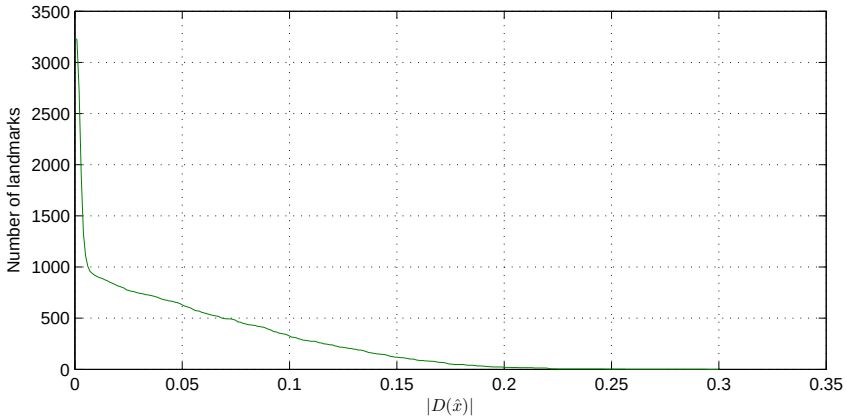
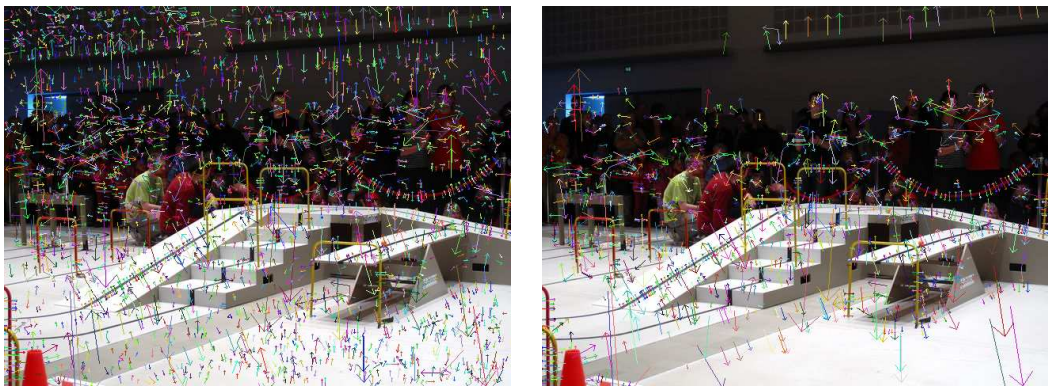


Figure F.3: *The number of found feature point as function of a threshold of* $|D(\hat{X})|$



(a) *No threshold is applied.* **(b)** *Threshold at 0.02*

Figure F.4: *The point of origin for the arrows indicate a feature point location. The orientation and magnitude of the arrow is described in later in this section.*

where:

H: 2×2 Hessian matrix [·]
r: Ratio between principal curvatures [·]

The result of applying both thresholds at the same time is shown in Figure F.6b. It can be seen on the figure that the response along the edges has been reduced, and feature points with low variance have been rejected. For that particular image, the thresholds have reduced the number of feature points from 3271 to 695.

The orientation is found as a peak in an orientation histogram with 36 bins. If there are peaks above 80% of the largest peak, then more feature point with different orientations are assigned to that location. The orientation is calculated using Formula F.8 and the magnitude of the orientation is calculated using Formula F.7. σ of the $L(x, y, \sigma)$ are set as the scale for the feature points found previously. The orientation and magnitude are shown for as the orientation and length of the arrows in Figure F.4 and F.6.

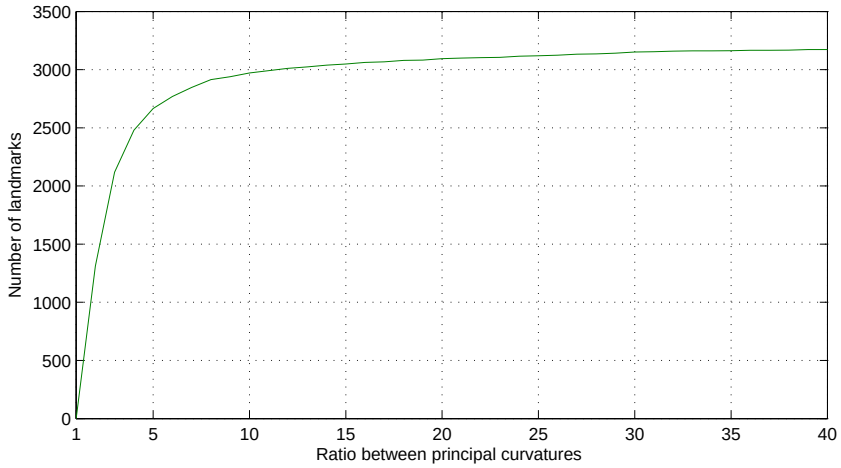


Figure F.5: *The number of found feature point as a function of the threshold on the ratio between principal curvatures*

(a) *Threshold in Formula F.6 applied with* $r = 5$.

(b) *Both Thresholds applied;* $r = 5$ *and* $|D(\hat{x})| < 0.02$

Figure F.6: *The point of origin for the arrows indicate a feature point location. The orientation and magnitude of the arrow is described later in this section.*

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad \text{[pixels]} \quad \text{(F.7)}$$

$$\theta(x,y) = atan(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}) \quad \text{[rad]} \quad \text{(F.8)}$$

F.1.3 Descriptor

The descriptor of a feature point must be invariant to 3D-view point transformations. A simple way of describing the feature points is based on correlation. A better method is based on a model of biological vision. This method was proposed by Edelman, Intrator, and Poggio in 1997. The model is based on complex neurons in primary visual cortex. They have tested these features against a method using correlation with a 3D computer model rotated in depth by 20 degrees. The result showed that the model improved the recognition rate to 94% from 35% [129].

The model uses the gradient magnitude and the orientations, like the calculation of the orientation.

Figure F.7 shows the principle of calculating the descriptor. The arrows on the left figure shows the gradient magnitude and orientation in an area around the feature point. The circle illustrates a Gaussian smoothing function. This is used to assign less importance to the gradients at sample points far away from the feature point. The gradients are represented by a histogram for each 4×4 sub area. Tests have shown that the best result is generated from 8 direction bins per histogram and a histogram matrix of 4×4. On the right figure, a histogram matrix of 2×2 with 8 directions is shown. This results in a feature vector with the length of $4 \times 4 \times 8 = 128$. To improve the invariance to illumination, the vector is normalized.

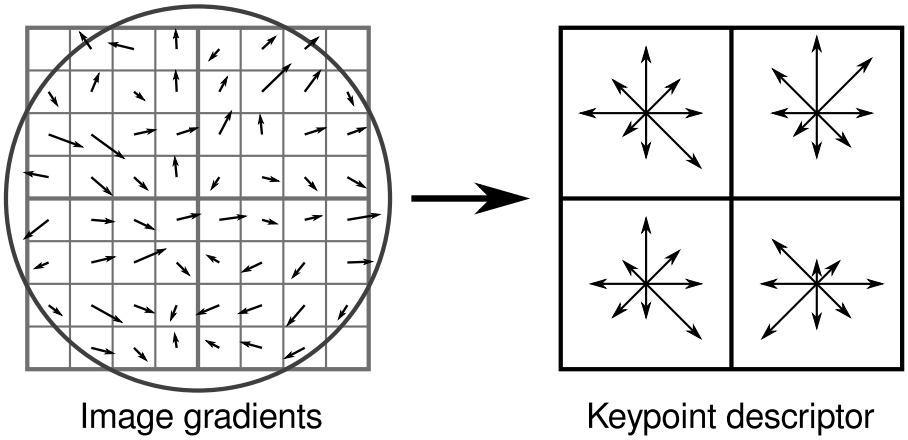


Figure F.7: *On the left figure: The magnitude and orientation of the gradients found around a feature point. The circle illustrates a Gaussian smoothing function. On the right figure: A 2×2 matrix of orientation histograms with 8 directions. These are the descriptor for a feature point.*

F.2 Match descriptors

It is not likely to find a perfect match for a descriptor, especially since it is high dimensional; 128 dimensions. The best match between a descriptor and a group of descriptors can be found by the nearest neighbor method. This method calculates the distance between the descriptors, but because of the high dimensionality, a static threshold for the distance will perform poorly. A more effective solution is to examine the ratio between the nearest neighbor and the second nearest neighbor. If the two neighbors have the same distance, then it is more likely that none of them are a correct match.

This method has been tested by Lowe and it is found that a threshold of 0.8 will reduce the probability of incorrect matches by 90% and only reduce the probability of correct matches by 5%. For this project, the threshold is set to 0.75 as this will reduce the probability of incorrect matches even more, thereby giving more stability to matched feature points.

The algorithm to perform the search for the nearest neighbor is a modification of the k dimensional search tree (the k-d-tree). The k-d-tree does not improve search speed for dimensions over 10 [130]. Therefore, the Best-Bin-First (BBF) version of the k-d-tree is used [131]. The BBF approximates the solution to the nearest neighbor, and thereby reduces the amount of feature points, which must be searched through. It has been chosen to set the search level to 100 feature points, as Lowe found that the search among 200 feature points performs well for a database with

100,000 feature points, and it is assumed that the amount of feature points found in an image will be much lower, see Figure F.6.

Appendix G
KLT Feature Point Tracker

In 1981 the Lucas and Kanade in [132] presented a method to perform efficient template matching in a local neighborhood. The search for a match is performed using gradients weighed by an approximation to the second derivative of the image. Given an image location $u = [x, y]$ and two images $F(u)$ and $I(u) = F(u + d)$ the gradients can be approximated as Equation G.1. Furthermore, the displacement d can be approximated as Equation G.2.

$$F'(u) \approx \frac{I(u) - F(u)}{d} \quad [\cdot] \tag{G.1}$$

$$d \approx \frac{I(u) - F(u)}{F'(u)} \quad [\cdot] \tag{G.2}$$

Where

$I(u)$: Image one [$\cdot$]
$F(u)$: Image two [$\cdot$]
$F'(u)$: Gradient of the image [$\cdot$]
$d = [d_x \; d_y]$: Displacement [px]

The displacement d is found by minimizing Equation G.3, which is the error function, in a given window W.

$$\epsilon = \sum \sum_{W} \left[F(u + d) - I(u)\right]^2 w(u) \quad [\cdot] \tag{G.3}$$

Where

ϵ: Matching error. [$\cdot$]
$W = [w_x \; w_y]$: Window with width $2w_x + 1$ and height $2w_y + 1$. [px]
$w(u)$: Weight function. [$\cdot$]

The minimization of Equation G.3 is to be obtained through Newton-Raphson. The Newton-Raphson algorithm locates zeros of a function thorough iteration. Equation G.3 is differentiated with respect to d and then locate zeros of this function. The derivative of Equation G.3 is found as Equation G.4, where the first order Taylor expansion of $F(u + d) \approx F(u) + g \cdot d$ is used[133].

$$\frac{\delta \epsilon}{\delta d} = \sum \sum_{W} \left[F(u) - I(u) + g(u)^T d\right] g(u) w(u) \quad [\cdot] \tag{G.4}$$

$$g(u) = [g_x \; g_y]^T = [\frac{\delta F(u)}{x} \; \frac{\delta F(u)}{y}] \quad [\cdot]$$

Where

$g(u)$: Gradient vector of $F(u)$. $[\cdot]$

(G.5)

By setting Equation G.4 to zero Equation G.6 can be obtained, this is the tracking equation[133]. This equation must be solved in all iterations.

$$\sum\sum_{W}\left[g(u)g^{T}(u)w(u)d\right]=\sum\sum_{W}\left[\left(I(u)-F(u)\right)g(u)w(x)\right] \qquad [\cdot] \qquad \text{(G.6)}$$

$$Gd=\mathbf{e} \qquad [\cdot]$$

Where

$$G=\sum\sum_{W}\left[g(u)g^{T}(u)w(u)\right] \qquad [\cdot]$$

$$\mathbf{e}=\sum\sum_{W}\left[\left(I(u)-F(u)\right)g(u)w(x)\right] \qquad [\cdot]$$

The matrix G is given as Equation G.7 (with $w(x)$ set to one for simplicity)[67].

$$G=\begin{bmatrix}\sum\sum_{w}g_x^2 & \sum\sum_{w}g_xg_y\\ \sum\sum_{w}g_xg_y & \sum\sum_{w}g_y^2\end{bmatrix} \qquad [\cdot] \qquad \text{(G.7)}$$

(G.8)

Where

$g_x,\ g_y$: Elements of the gradient vector $g(x)$. $[\cdot]$

The rank of G must be two for an unique solution to exist, as the tracking equation provides two equations and two unknowns, $d=[d_x\ d_y]$. This also implies that both eigenvalues of G must be larger than zero. This quality is also used to select features to track (see next section).

The window size W much chosen with respect to accuracy and robustness. A small window size will achive height accuracy but will not be able to track large motions. A large window will act the opposite way, as large motion can be tracked by on the cost of accuracy. In an idealistic situation, this relationship should be satisfied $d_x \leq w_x$ and $d_y \leq w_y$.

To overcome this problem two different methods can be used: varying the window size or varying the image resolution[134]. The later is the most common, as this is the fastest, and can be implemented by building a resolution pyramid of images. The pyramid is constructed in a recursive way such as $G^l(x)$ is computed from $G^{l-1}(x)$.

G.1 Select feature points

The first feature detection algorithm used in KLT tracking was proposed by Tomasi and Kanade in 1991[67]. This algorithm is similar to the Harris corner detector from 1988[135]. The features detected are detected with the main purpose of being robust for tracking, e.g. features are selected with respect to the tracking algorithm and its properties. One property is that a feature can easily be tracked if the (G) has large eigenvalues. Therefore, the features are selected if they are above a defined threshold such as $min(\lambda_1, \lambda_2) \geq \lambda_{thr}$ [136].

In Figure G.1 the amount of feature points found given the value of λ_{thr} is shown for 25 pictures. The pictures are 25 randomly selected from the Oxford dataset. It can be seen that the

amount of features decrease rapidly low values of λ_{thr} where the amount is more stable for high values. It is chosen to use $\lambda_{thr} = 750$ as threshold as the amount of feature points have stabilized in most of the pictures at this value.

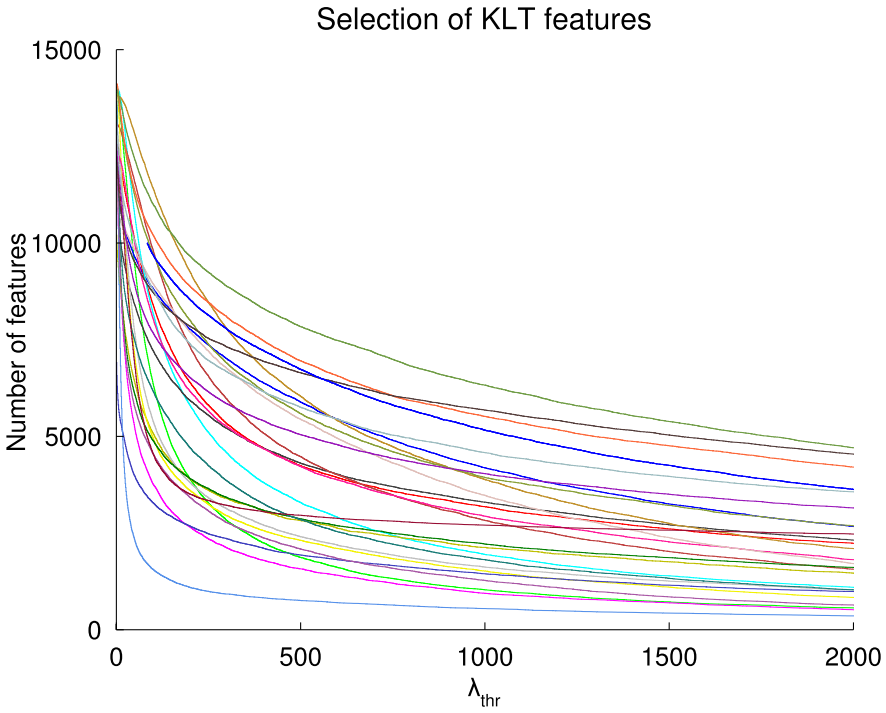


Figure G.1: *The figure shows the number of features found given the value of λ_{thr} for 25 random pictures from the Oxford dataset.*

Appendix H
Estimate layer shape

This appendix describes the process of estimating the layer shape from a set of points $\left[X \right]$ inside the layer. Assuming that the point correspondence is uniformly distributed over the entire layer, the shape of the point correspondence will approximate the shape of the layer.

A simple way of estimating the shape of X is through the convex hull, which results in a minimal convex set containing X. Given the three cases in Figure H.1 it can be seen that the layer shape is not necessarily convex. For the case in Figure H.1b the occlusion of the green layer result causes the layer shape to be non-convex. Representing the green layer by a convex shape, will almost contain the entire blue box.

(a) *Non-convex planar surface and layer.*

(b) *Convex surface but non-convex layer shape, due to occlusion of blue box.*

(c) *Facades of two buildings which share the same 3D planar surface*

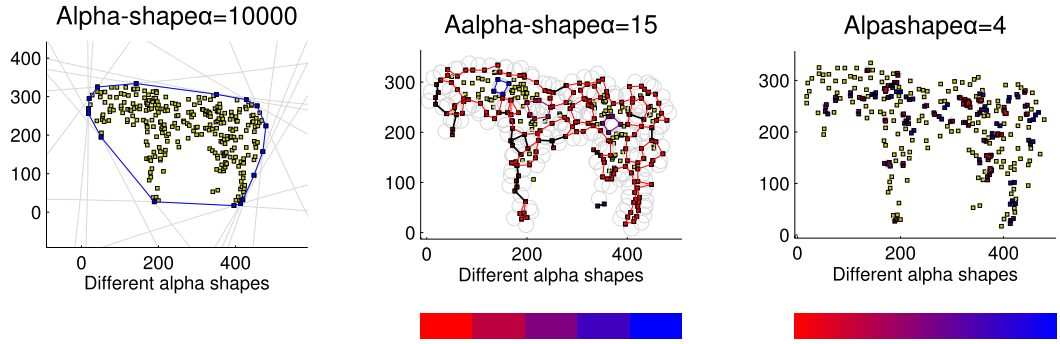
Figure H.1: *Three cases of non-convex layer shapes*

Multiple objects may be on the same planar surface without being connected, (see Figure H.1c). The estimation of inliers and homography may contain correspondence for both objects. The shape estimator must be capable of estimating multiple non-convex shapes one point set, X.

There are multiple methods which can be used to estimate non-convex, non-connected shapes of 2D and 3D point sets; DSAM[137], Characteristic-Hull[138] and Alpha shape[139].

Alpha shape was proposed in 1983 by Edelsbrunner et al. and is widely used in the literature [140, 141, 142, 143]. Alpha shape is a generalization of the convex hull and is a subgraph of the Delaunay triangulation. Given a point set X and an alpha value α, then the alpha shape of X X can be found which is neither necessarily convex nor necessarily connected. In this work the matlab implementation provided by [144] is used.

For large α values the alpha shape is identical to the convex hull of X, see H.2a. As α decreases the alpha shape shrinks and cavities can occur (see Figure H.2b). For a small value of α the alpha shape is empty, see Figure H.2c. The alpha value determines the distance between points which are considered to share the same shape.



(a) **(b)** **(c)**

Figure H.2: *Optained alpha shapes using three different alpha values*

The next problem is how to determine α such that non-connected shapes remain non-connected. Previous alpha values have been determined to satisfy either or both of the two following properties[145].

- All data points are either on the boundary or in the interior of the alpha shape.
- The number of components is equal or less than a given number.

In this work a method which use the distance between points in a close neighborhood, is proposed. Given the case in Figure H.1c with two facades which are non-connected but still share the same homography, some points are distributed uniformly.

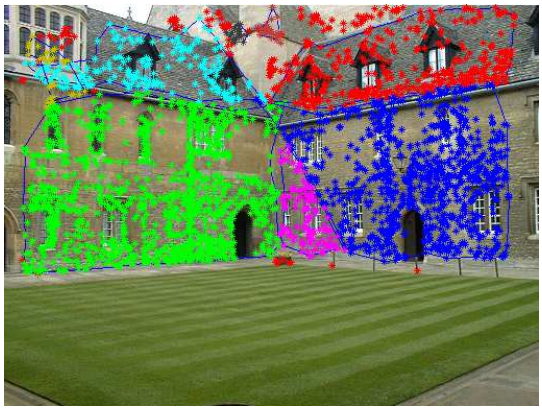
The alpha value is determined to be the value that will collect a given point and the n closest neighbor point in the same alpha shape. The number of points to consider can be constant or it can depend on the number of points available. The disadvantage with a constant n is that with low amount of points will the distance to the n neighbor increases and also the probability for merging non-connected layers. Given a large number of points it can be assumed that the distribution is more dense, and therefore more neighbors can be combined in the same layer. In this work n is determined as 7.5 % of the total number of points, which is found to be a good estimate.

The distance to the n closest neighbors can be determined with a sorted distance matrix. The alpha value is then calculated as the median of the three longest distances in row n. The median is used to make the alpha value more robust to a single outlier.

Appendix I

Layer extraction results

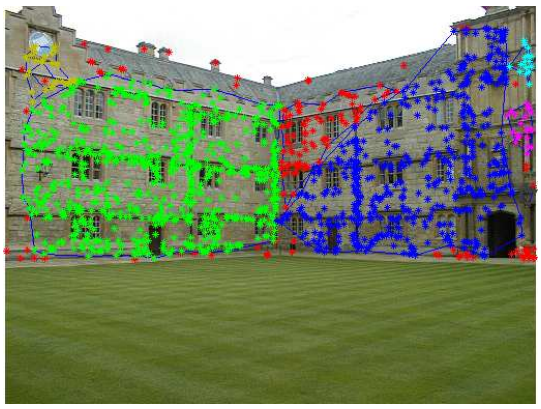
This appendix shows the eight test cases from the "Layer Extraction" module validation, see Section 8.4.



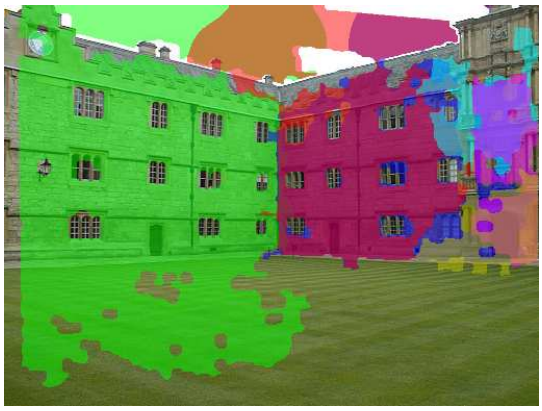
(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*

(b) *Final estimate of layers.*

Figure I.1: *Extended RANSAC with shape and the final improved layers for test case 1. There are no correlation between the colors in figure a and b.*



(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*



(b) *Final estimate of layers.*

Figure I.2: *Extended RANSAC with shape and the final improved layers for test case 2. There are no correlation between the colors in figure a and b.*

(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*

(b) *Final estimate of layers.*

Figure I.3: *Extended RANSAC with shape and the final improved layers for test case 3. There are no correlation between the colors in figure a and b.*

(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*

(b) *Final estimate of layers.*

Figure I.4: *Extended RANSAC with shape and the final improved layers for test case 4. There are no correlation between the colors in figure a and b.*

(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*

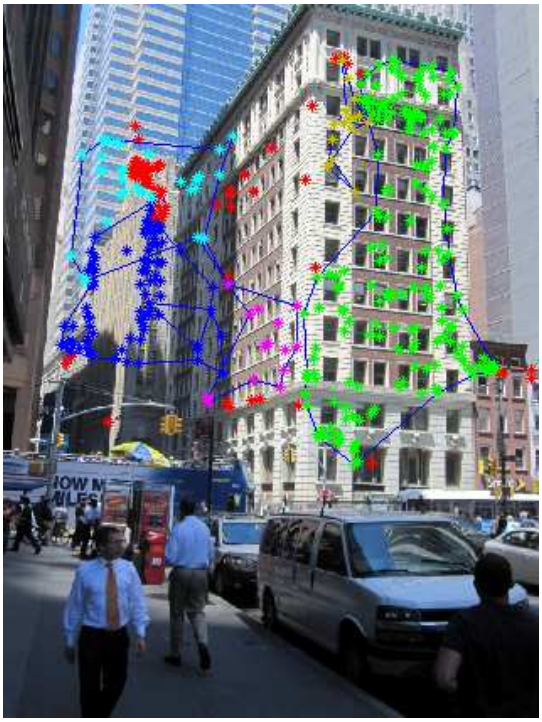
(b) *Final estimate of layers.*

Figure I.5: *Extended RANSAC with shape and the final improved layers for test case 5. There are no correlation between the colors in figure a and b.*

(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*

(b) *Final estimate of layers.*

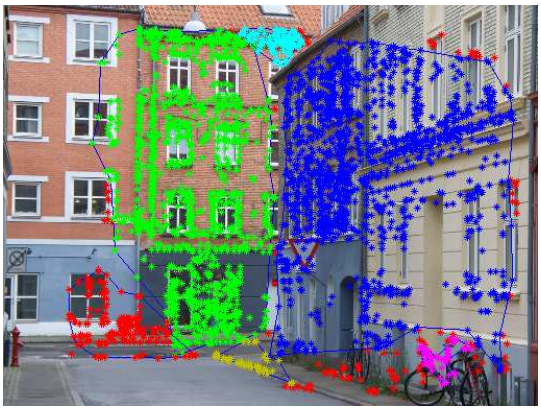
Figure I.6: *Extended RANSAC with shape and the final improved layers for test case 6. There are no correlation between the colors in figure a and b.*



(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*

(b) *Final estimate of layers.*

Figure I.7: *Extended RANSAC with shape and the final improved layers for test case 7. There are no correlation between the colors in figure a and b.*



(a) *Extended RANSAC result. Each layer have a different color and is enclosed by the shape border.*

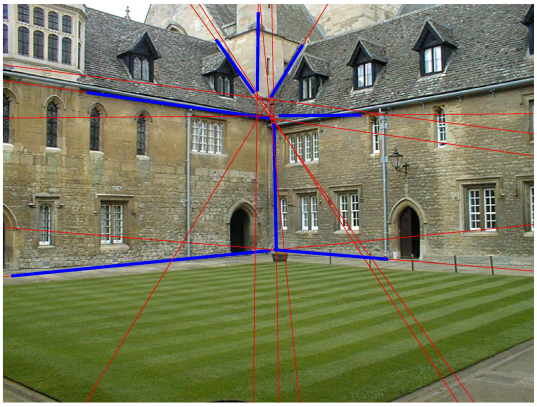
(b) *Final estimate of layers.*

Figure I.8: *Extended RANSAC with shape and the final improved layers for test case 8. There are no correlation between the colors in figure a and b.*

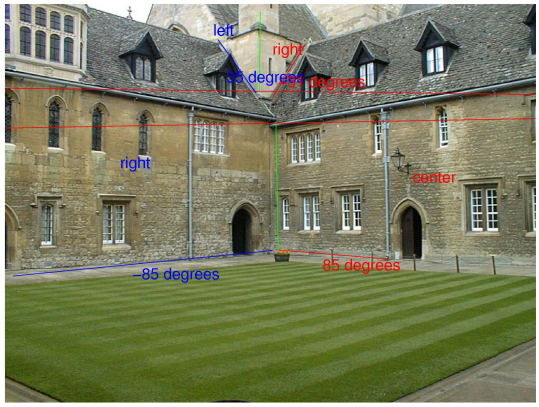
Appendix J

Geometric scene understanding results

This appendix shows the eight test cases from the "Geometric scene understanding" module validation, see Section 9.5.

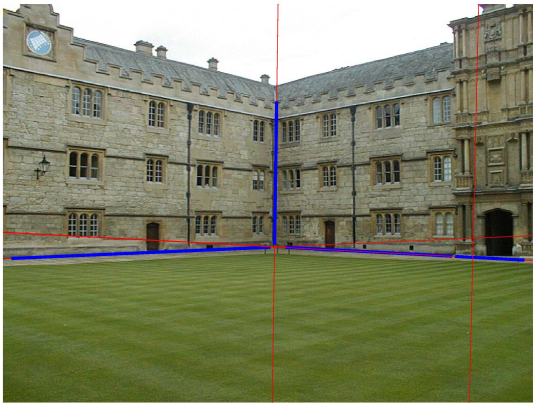


(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*

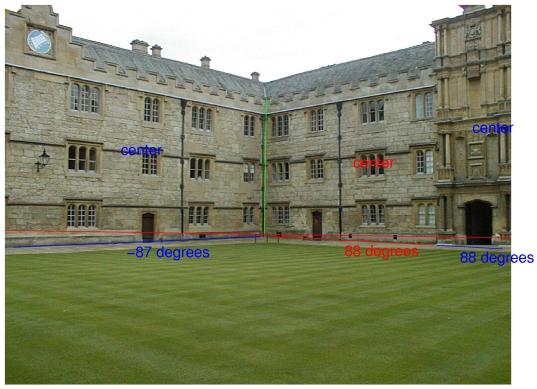


(b) *Estimated orientation.*

Figure J.1: *Connection and orientation of test case 1.*



(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*



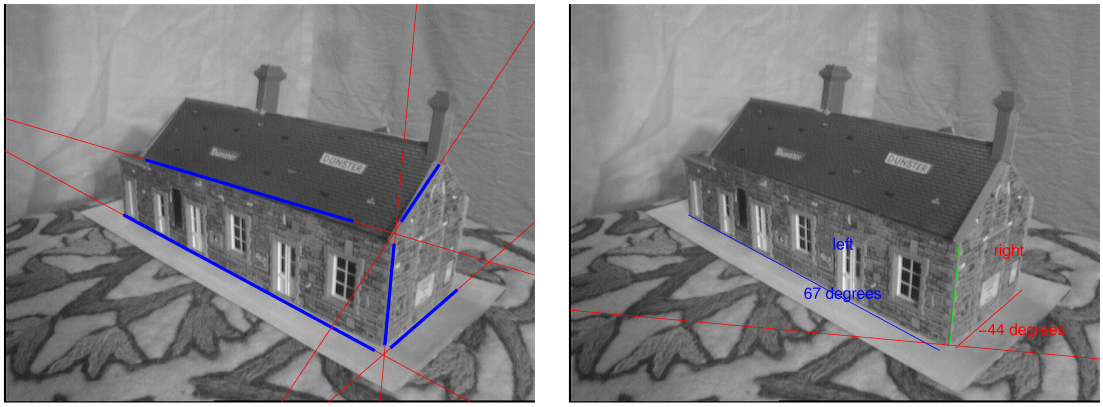
(b) *Estimated orientation.*

Figure J.2: *Connection and orientation of test case 2.*

(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*

(b) *Estimated orientation.*

Figure J.3: *Connection and orientation of test case 3.*



(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*

(b) *Estimated orientation.*

Figure J.4: *Connection and orientation of test case 4.*

(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*

(b) *Estimated orientation.*

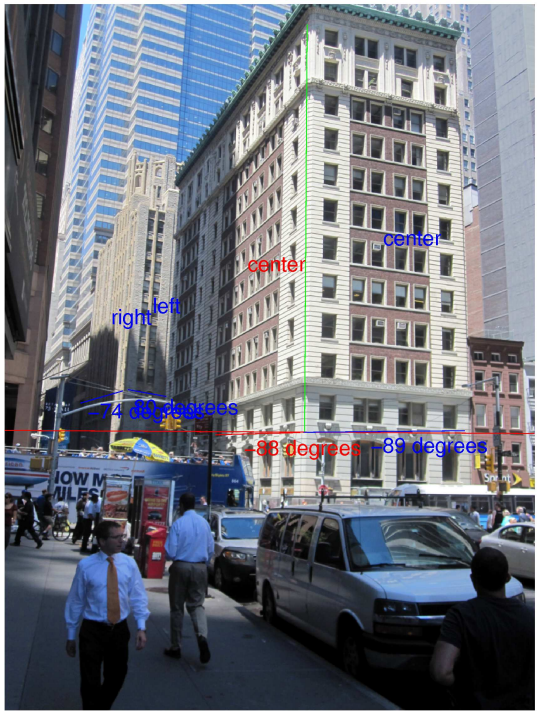
Figure J.5: *Connection and orientation of test case 5.*

(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*

(b) *Estimated orientation.*

Figure J.6: *Connection and orientation of test case 6.*

(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*



(b) *Estimated orientation.*

Figure J.7: *Connection and orientation of test case 7.*

(a) *Layer connections where red lines are the infinity connections and blue lines are the final estimated connections.*

(b) *Estimated orientation.*

Figure J.8: *Connection and orientation of test case 8.*

Bibliography

[1] D. G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.

[2] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, 2002.

[3] L. Amsaleg, P. Gros, and S.-A. Berrani, "Robust object recognition in images and the related database problems." *Multimedia Tools Appl.*, vol. 23, no. 3, pp. 221–235, 2004.

[4] C. Baillard and A. Zisserman, "Automatic reconstruction of piecewise planar models from multiple views." in *CVPR*. IEEE Computer Society, 1999, pp. 2559–2565. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr1999.html#BaillardZ99

[5] A. R. Dick, P. H. S. Torr, S. J. Ruffle, and R. Cipolla, "Combining single view recognition and multiple view stereo for architectural scenes." in *ICCV*, 2001, pp. 268–274. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2001-1.html#DickTRC01

[6] R. Szeliski, *Algorithms and Applications 1st Edition*. Springer, 2011.

[7] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3d model construction for turn-table sequences." in *SMILE*, ser. Lecture Notes in Computer Science, R. Koch and L. J. V. Gool, Eds., vol. 1506. Springer, 1998, pp. 155–170. [Online]. Available: http://dblp.uni-trier.de/db/conf/smile/smile1998.html#FitzgibbonCZ98

[8] H. Jin, S. Soatto, and A. J. Yezzi, "Multi-view stereo reconstruction of dense shape and complex appearance." *International Journal of Computer Vision*, vol. 63, no. 3, pp. 175–189, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijcv/ijcv63.html#JinSY05

[9] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day." in *ICCV*. IEEE, 2009, pp. 72–79. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2009.html#AgarwalSSSS09

[10] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. N. Sinha, B. Talton, L. W. 0002, Q. Yang, H. Stewénius, R. Yang, G. Welch, H. Towles, D. Nistér, and M. Pollefeys, "Towards urban 3d reconstruction from video." in *3DPVT*. IEEE Computer Society, 2006, pp. 1–8.

[11] L. G. Roberts, "Machine perception of three-dimensional solids," *IEEE Transactions on Reliability*, 1963.

[12] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, 2009.

[13] A. Gupta, A. A. Efros, and M. Hebert, "Blocks world revisited: Image understanding using qualitative geometry and mechanics." in *ECCV (4)*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6314. Springer, 2010, pp. 482–496.

[14] S. Ullman, "The interpretation of structure from motion." *Proc R Soc Lond B Biol Sci*, vol. 203, no. 1153, pp. 405–426, 1979 Jan 15.

[15] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[16] P. Parodi and G. Piccioli, "3d shape reconstruction by using vanishing points." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, pp. 211–217, 1996. [Online]. Available: http://dblp.uni-trier.de/db/journals/pami/pami18.html#ParodiP96

[17] E. Guillou, D. Meneveaux, E. Maisel, and K. Bouatouch, "Using vanishing points for camera calibration and coarse 3d reconstruction from a single image." *The Visual Computer*, vol. 16, no. 7, pp. 396–410, 2000. [Online]. Available: http://dblp.uni-trier.de/db/journals/vc/vc16.html#GuillouMMB00

[18] C. Tomasi, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, pp. 137–154, 1992.

[19] S. Christy and R. Horaud, "Euclidean shape and motion from multiple perspective views by affine iterations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1098–1104, 1996.

[20] O. D. Faugeras, Q.-T. Luong, and T. Papadopoulo, *The geometry of multiple images - the laws that govern the formation of multiple images of a scene and some of their applications.* MIT Press, 2001.

[21] B. Triggs, P. F. Mclauchlan, R. I. Hartley, and A. W. Fitzgibbon, *Bundle Adjustment – A Modern Synthesis*, January 2000, vol. 1883. [Online]. Available: http://www.metapress.com/link.asp?id=PLVCRQ5BX753A2TN

[22] Google, "Google earth," 2011, http://earth.google.com.

[23] H.L. Chou and Z. Chen, "A novel 3d planar object reconstruction from multiple uncalibrated images using the plane-induced homographies." 2004, pattern Recognition Letters 25.

[24] J. Wright, A. Wagner, S. Rao, and Y. Ma, "Homography from coplanar ellipses with application to forensic blood splatter reconstruction." *Computational Intelligence in Robotics and Automation*, p. 1:424–429, 2006.

[25] C. Sagüés, A. C. Murillo, F. Escudero, and J. J. Guerrero, "From lines to epipoles through planes in two views." *Pattern Recognition*, vol. 39, no. 3, pp. 384–393, 2006.

[26] R. I. Hartley, "In defense of the eight-point algorithm." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 6, pp. 580–593, 1997. [Online]. Available: http://dblp.uni-trier.de/db/journals/pami/pami19.html#Hartley97a

[27] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1988.

[28] H. Zeng, X. Deng, and Z. Hu, "A new normalized method on line-based homography estimation." *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1236–1244, 2008.

[29] E. Dubrofsky and R. J. Woodham, "Combining line and point correspondences for homography estimation." *ISVC: International Symposium on Visual Computing*, pp. 1236–1244, 2008.

[30] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[31] J.-J. Lee and G.-Y. Kim, "Robust estimation of camera homography using fuzzy ransac." in *ICCSA (1)*, ser. Lecture Notes in Computer Science, O. Gervasi and M. L. Gavrilova, Eds., vol. 4705. Springer, 2007, pp. 992–1002.

[32] E. Vincent and R. Laganiere, "Detecting planar homographies in an image pair," ser. Lecture Notes in Computer Science. IEEE, 2001, pp. 182 – 187.

[33] H. Bay, V. Ferrari, and L. J. V. Gool, "Wide-baseline stereo matching with line segments." in *CVPR (1)*. IEEE Computer Society, 2005, pp. 329–336.

[34] P. J. Rousseeuw, "Least median of squares regression," *American Statistical Association*, vol. 79, no. 388, pp. 871–880, 1984.

[35] O. D. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment." *International Journal of Pattern Recognition and Artificial Intelligence vol 2*, p. 485–508, 1988.

[36] A. Z. Zhang, "3d reconstruction based on homography mapping." 1996, aRPA Image Understanding Workshop.

[37] Z. Zhang, "Three-dimensional reconstruction under varying constraints on camera geometry for robotic navigation scenarios." 1996, electronic Doctoral Dissertations for UMass Amherst.

[38] J. Wang and E. Adelson, "Layered representation for motion analysis." *Computer Vision and Pattern Recognition(CVPR)*, 1993.

[39] A. Jepson and M. Black, "Mixture models for optical flow computation." *Computer Vision and Pattern Recognition(CVPR)*, 1993.

[40] Q. Ke and T. Kanade, "Transforming camera geometry to a virtual downward-looking camera: Robust ego-motion estimation and ground-layer detection." *Computer Vision and Pattern Recognition(CVPR)*, 2003.

[41] S. Ayer and H. S. Sawhney, "Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding." in *ICCV*, 1995, pp. 777–.

[42] Y. Weiss, "Smoothness in layers: Motion segmentation using nonparametric mixture estimation." in *CVPR*. IEEE Computer Society, 1997, pp. 520–526.

[43] M. R. Gupta and Y. Chen, "Theory and use of the em algorithm." *Foundations and Trends in Signal Processing*, vol. 4, no. 3, pp. 223–296, 2010.

[44] J. Rissanen, "Modeling by the shortest data description," *Automation*, vol. 14, pp. 465–471, 1978.

[45] ——, "A universal prior for integers and estimation by minimum description length," *The Annals of Statistics*, vol. 11, no. 2, pp. 416–431, 1983.

[46] M. Irani, B. Rousso, and S. Peleg, "Detecting and tracking multiple moving objects using temporal integration." in *ECCV*, ser. Lecture Notes in Computer Science, G. Sandini, Ed., vol. 588. Springer, 1992, pp. 282–287.

[47] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields." *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[48] P. Rousseeuw and A. Leroy, *Robust regression and outlier detection*, ser. Wiley Series in probability and mathematical statistics. New York [u.a.]: Wiley, 1987.

[49] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow," 1996, p. 231–236.

[50] J. Xiao and M. Shah, "Motion layer extraction in the presence of occlusion using graph cut." in *CVPR (2)*, 2004, pp. 972–979.

[51] ——, "Motion layer extraction in the presence of occlusion using graph cuts." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1644–1659, 2005.

[52] ——, "Accurate motion layer segmentation and matting." in *CVPR (2)*. IEEE Computer Society, 2005, pp. 698–703.

[53] Y. Altunbasak, P. E. Eren, and A. M. Tekalp, "Region-based parametric motion segmentation using color information." *Graphical Models and Image Processing*, vol. 60, no. 1, pp. 13–23, 1998.

[54] J. Chetan, "Scene interpretation in images and videos," 2010, center for Visual Information Technology International Institute of Information Technology.

[55] K. Kanatani and N. Ohta, "Accuracy bounds and optimal computation of homography for image mosaicing applications," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, 1999, pp. 73 –78 vol.1.

[56] H. Tong, M. Li, H. Zhang, and C. Zhang, "Blur detection for digital images using wavelet transform." in *ICME*. IEEE, 2004, pp. 17–20.

[57] P. N. Topiwala, "Review of 'ten lectures on wavelets' (daubechies, i.; 1992)." *IEEE Transactions on Information Theory*, vol. 41, no. 4, p. 1203, 1995.

[58] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas, "The blur effect: perception and estimation with a new no-reference perceptual blur metric," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 6492, Mar. 2007.

[59] V. G. Group, "Data," 2011. [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/

[60] S. H. Westin, "Iso 12233 photography," 2011.

[61] S. I. Olsen, "Estimation of noise in images: An evaluation." *CVGIP: Graphical Model and Image Processing*, vol. 55, no. 4, pp. 319–323, 1993.

[62] P. J. Huber, "Robust statistics," 1981.

[63] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard, "Wavelet shrinkage: asymptopia," *Journal of the Royal Statistical Society, Ser. B*, pp. 371–394, 1995.

[64] J. Immerkær, "Fast noise variance estimation." *Computer Vision and Image Understanding*, vol. 64, no. 2, pp. 300–302, 1996.

[65] R. D. Boyle and R. C. Thomas, *Computer vision: a first course*. Oxford, UK, UK: Blackwell Scientific Publications, Ltd., 1988.

[66] R. Hummel, "Image enhancement by histogram transformation," *Computer Graphics and Image Processing*, vol. 6, no. 2, pp. 184 – 195, 1977. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0146664X77800117

[67] C. Tomasi and T. Kanade, "Detection and tracking of point features," 1991, technical Report.

[68] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," 2004, www.cs.ubc.ca/~lowe/papers/ijcv04.pdf.

[69] S. Birchfield, "Klt: An implementation of the kanade-lucas-tomasi feature tracker," http://www.ces.clemson.edu/~stb/klt, 2007.

[70] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[71] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections." *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[72] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 19, no. 6, 1997.

[73] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *"SIAM Journal on Applied Mathematics"*, vol. 11, no. 2, pp. 431–441, 1963. [Online]. Available: http://dx.doi.org/10.1137/0111030

[74] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 51, no. 2, pp. pp. 271–279, 1989.

[75] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[76] Y. Boykov, O. Veksler, and R. Zabih, "Efficient approximate energy minimization via graph cuts," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1222–1239, November 2001.

[77] A. K. Sinop, "A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm."

[78] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?" *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, February 2004.

[79] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, September 2004.

[80] S. Bagon, "Matlab wrapper for graph cut," 2006.

[81] A. Shashua, "Projective depth: A geometric invariant for 3d reconstruction from two perspective/orthographic views and for visual recognition," in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, may 1993, pp. 583 –590.

[82] F. Schaffalitzky and A. Zisserman, "Planar grouping for automatic detection of vanishing lines and points." *Image Vision Comput.*, vol. 18, no. 9, pp. 647–658, 2000. [Online]. Available: http://dblp.uni-trier.de/db/journals/ivc/ivc18.html#SchaffalitzkyZ00

[83] A. Matessi and L. Lombardi, "Vanishing point detection in the hough transform space." in *Euro-Par*, ser. Lecture Notes in Computer Science, P. Amestoy, P. Berger, M. J. Daydé, I. S. Duff, V. Frayssé, L. Giraud, and D. Ruiz, Eds., vol. 1685. Springer, 1999, pp. 987–994. [Online]. Available: http://dblp.uni-trier.de/db/conf/europar/europar99.html#MatessiL99

[84] J. Kosecká and W. Zhang, "Video compass." in *ECCV (4)*, ser. Lecture Notes in Computer Science, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds., vol. 2353. Springer, 2002, pp. 476–490. [Online]. Available: http://dblp.uni-trier.de/db/conf/eccv/eccv2002-4.html#KoseckaZ02

[85] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008, similarity Matching in Computer Vision and Multimedia. [Online]. Available: http://www.sciencedirect.com/science/article/B6WCX-4RC2S4T-2/2/c2c03b6165996e30312e5b7c7b681155

[86] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image." *International Journal of Computer Vision*, vol. 75, no. 1, pp. 151–172, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijcv/ijcv75.html#HoiemEH07

[87] V. Hedau, D. Hoiem, and D. A. Forsyth, "Recovering the spatial layout of cluttered rooms." in *ICCV*. IEEE, 2009, pp. 1849–1856. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2009.html#HedauHF09

[88] ——, "Thinking inside the box: Using appearance models and context based on room geometry." in *ECCV (6)*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6316. Springer, 2010, pp. 224–237.

[89] D. Hoiem, A. Stein, A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *International Conference on Computer Vision (ICCV)*, October 2007.

[90] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, 2004. [Online]. Available: http://people.cs.uchicago.edu/~pff/segment/

[91] David Fleet and Aaron Hertzmann, "Computer graphics lecture notes," 2006, http://www.dgp.toronto.edu/~hertzman/418notes.pdf.

[92] E. Hecht, *Optics, 4th ed.* Addison Wesley., 2002.

[93] wikipedia, "Images," 2011, www.wikipedia.org.

[94] H. S. Sawhney and A. R. Hanson, "Identification and 3d description of shallow environmental structure over a sequence of images," *CVPR91, Maui, Hawaii*, 1991, pp. 179–185.

[95] D. C. Brown, "Close-range camera calibration," *PHOTOGRAMMETRIC ENGINEERING*, vol. 37, no. 8, pp. 855–866, 1971.

[96] R. Y. Tsai, "Radiometry," L. B. Wolff, S. A. Shafer, and G. Healey, Eds. , USA: Jones and Bartlett Publishers, Inc., 1992, ch. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, pp. 221–244.

[97] K. Gremban, C. Thorpe , and T. Kanade, "Geometric camera calibration using systems of linear equations," in *Proceedings of IEEE Conference on Robotics and Automation (ICRA '88)*, vol. 1, April 1988, pp. 562 – 567.

[98] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank, "Camera self-calibration: Theory and experiments." in *ECCV*, ser. Lecture Notes in Computer Science, G. Sandini, Ed., vol. 588. Springer, 1992, pp. 321–334. [Online]. Available: http://dblp.uni-trier.de/db/conf/eccv/eccv1992.html#FaugerasLM92

[99] G. P. Stein, "Accurate internal camera calibration using rotation, with analysis of sources of error." in *ICCV*, 1995, pp. 230–236. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv1995.html#Stein95

[100] R. I. Hartley, "Self-calibration of stationary cameras." *International Journal of Computer Vision*, vol. 22, no. 1, pp. 5–23, 1997. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijcv/ijcv22.html#Hartley97

[101] H.-Y. Shum and R. Szeliski, "Correction to construction of panoramic image mosaics with global and local alignment." *International Journal of Computer Vision*, vol. 48, no. 2, pp. 151–152, 2002. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijcv/ijcv48.html#ShumS02

[102] M. Brown and D. G. Lowe, "Recognising panoramas." in *ICCV*. IEEE Computer Society, 2003, pp. 1218–1227. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2003-2.html#BrownL03

[103] ——, "Automatic panoramic image stitching using invariant features." *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijcv/ijcv74.html#BrownL07

[104] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics." *ACM Trans. Graph.*, vol. 2, no. 4, pp. 217–236, 1983. [Online]. Available: http://dblp.uni-trier.de/db/journals/tog/tog2.html#BurtA83

[105] Z. Zhang, "A flexible new technique for camera calibration." *Transactions on Pattern Analysis and Machine Intelligence*, p. 22:1330–1334, 2000.

[106] Z. Chuan, T. D. Long, Z. Feng, and D. Z. Li, "A planar homography estimation method for camera calibration." *Computational Intelligence in Robotics and Automation*, p. 1:424–429, 2003.

[107] R. L. Lagendijk and J. Biemond, "Basic methods for image restoration and identification," 1999.

[108] Y. Tsin, V. Ramesh, and T. Kanade, "Statistical calibration of ccd imaging process," *ICCV 2001, Vancouver, Canada*, p. pp. 480–487, 2001.

[109] G. E. Healey and R. Kondepudy, "Radiometric ccd camera calibration and noise estimation," *Transactions on Pattern Analysis and Machine Intelligence*, 1994, 16(3):267–276.

[110] L. W. MacDonald and C. W. Honsinger, "Digital heritage: Applying digital imaging to cultural heritage." *J. Electronic Imaging*, vol. 16, no. 1, p. 019901, 2007.

[111] A. C. Bovik, Ed., *Handbook of image and video processing.* Amsterdam [u.a.]: Elsevier Academic Press, 2005.

[112] J. Sijbers, A. den Dekker, J. V. Audekerke, M. Verhoye, and D. V. Dyck, "Estimation of the noise in magnitude mr images," *Magnetic Resonance Imaging*, vol. 16, no. 1, pp. 87 – 90, 1998.

[113] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 146, no. 2, pp. 80 –84, aug 1999.

[114] Y. Dong, R. H. Chan, and S. Xu, "A detection statistic for random-valued impulse noise." *IEEE Transactions on Image Processing*, vol. 16, no. 4, pp. 1112–1120, 2007.

[115] H. Ji, C. Liu, Z. Shen, and Y. Xu, "Robust video denoising using low rank matrix completion." in *CVPR*. IEEE, 2010, pp. 1791–1798.

[116] M. Ghazal, A. Amer, and A. Ghrayeb, "A real-time technique for spatio-temporal video noise estimation." *IEEE Trans. Circuits Syst. Video Techn.*, vol. 17, no. 12, pp. 1690–1699, 2007.

[117] V. Zlokolica, S. Member, and W. Philips, "Noise estimation for video processing based on spatio-temporal gradients," *IEEE Signal Processing Letters*, vol. 13, pp. 337–340, 2006.

[118] A. Amer, E. Dubois, and A. Mitiche, "Reliable and fast structure-oriented video noise estimation." in *ICIP (1)*, 2002, pp. 840–843.

[119] C. Liu, R. Szeliski, S. B. Kang, C. L. Zitnick, and W. T. Freeman, "Automatic estimation and removal of noise from a single image." *Transactions on Pattern Analysis and Machine Intelligence*, 2008, 30(2):299–314.

[120] X. Marichal, W.-Y. Ma, and H. Zhang, "Blur determination in the compressed domain using dct information," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 2, 1999, pp. 386 –390 vol.2.

[121] E. Kalalembang, K. Usman, and I. Gunawan, "Dct-based local motion blur detection," in *Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2009 International Conference on*, nov. 2009, pp. 1 –6.

[122] M. Saad, A. Bovik, and C. Charrier, "Natural dct statistics approach to no-reference image quality assessment," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 313 –316.

[123] W. C. Graustein, "Homogeneous cartesian coordinates. linear dependence of points and lines." 1930, ch. 3 in Introduction to Higher Geometry.

[124] T. Ondrej Chum, "Evaluating error of homography." 2001, czech Technical University.

[125] P. Sampson, "Fitting conic sections to "very scattered" data: An iterarive refinement of the bookstein algorithm." 1982, cGIP, 18:97–108.

[126] Gellert, W., Gottwald, S., Hellwich, M., Kästner, H., and Künstner, H., *VNR Concise Encyclopedia of Mathematics*, 2nd ed. New York, Van Nostrand Reinhold, 1989.

[127] P. Grey, "bumblebee2 stereo camera," 2011. [Online]. Available: http://www.ptgrey.com/products/bumblebee2/bumblebee2_stereo_camera.asp

[128] W. Chojnacki, M. J. Brooks, A. van den Hengel, and D. Gawley, "Revisiting hartley's normalized eight-point algorithm." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1172–1177, 2003.

[129] T. P. Shimon Edelman, Nathan Intrator, "Complex cells and object recognition," 1997, kybele.psych.cornell.edu/~edelman/Archive/nips97.pdf.

[130] A. Moore, "An intoductory tutorial on kd-trees," 1991, http://www.autonlab.org/autonweb/14665/version/2/part/5/data/moore-tutorial.pdf.

[131] D. L. Jeffrey Beis, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," 1997, http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/cvpr97.pdf.

[132] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision." in *IJCAI*, P. J. Hayes, Ed. William Kaufmann, 1981, pp. 674–679.

[133] S. Birchfield, "Derivation of kanade-lucas-tomasi tracking equation," 1997. [Online]. Available: http://www.ces.clemson.edu/stb/klt/birchfield-klt-derivation.pdf

[134] Jean-Yves Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," 2000. [Online]. Available: http://robots.stanford.edu/cs223b04/algo_tracking.pdf

[135] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[136] J. Shi and C. Tomasi, "Good features to track," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.

[137] H. Alani, C. B. Jones, and D. Tudhope, "Voronoi-based region approximation for geographical information retrieval with gazetteers," *International Journal of Geographical Information Science*, vol. 15, no. 4, pp. 287–306, 2001.

[138] A. Galton and M. Duckham, "What is the region occupied by a set of points?" in *GIScience*, ser. Lecture Notes in Computer Science, M. Raubal, H. J. Miller, A. U. Frank, and M. F. Goodchild, Eds., vol. 4197. Springer, 2006, pp. 81–98. [Online]. Available: http://dblp.uni-trier.de/db/conf/giscience/giscience2006.html#GaltonD06

[139] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane." *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–558, 1983.

[140] P. Stelldinger, U. Köthe, and H. Meine, "Topologically correct image segmentation using alpha shapes," in *Discrete Geometry for Computer Imagery, Proc. DGCI '06*, ser. Springer LNCS, A. Kuba, L. G. Nyúl, and K. Palágyi, Eds., vol. 4245. Springer, 2006, pp. 542–554.

[141] T. J. Cholewo and S. T. Love, "Gamut boundary determination using alpha-shapes." in *Color Imaging Conference*. IST - The Society for Imaging Science and Technology, 1999, pp. 200–204. [Online]. Available: http://dblp.uni-trier.de/db/conf/imaging/cic1999.html#CholewoL99

[142] X. Xu and K. Harada, "Auto-surface reconstruction with alpha shape." in *MVA*, 2002, pp. 380–383.

[143] R. Ohbuchi and T. Takei, "Shape-similarity comparison of 3d models using alpha shapes." in *Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, 2003, pp. 293–302.

[144] U. Schwarz, "ashape: a pedestrian alpha shape extractor," 2005. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/6760

[145] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes." *ACM Trans. Graph.*, vol. 13, no. 1, pp. 43–72, 1994.