

2D to 3D Conversion with Direct Geometrical Search and Approximation Spaces

by
Maciej Borkowski

A Dissertation
submitted to the Faculty of Graduate Studies,
in Partial Fulfilment of the Requirements for the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

© by Maciej Borkowski, 24 August 2007

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba R3T 5V6 Canada

2D to 3D Conversion with Direct Geometrical Search and Approximation Spaces

by
Maciej Borkowski

**A Dissertation
submitted to the Faculty of Graduate Studies,
in Partial Fulfilment of the Requirements for the degree of**

**Doctor of Philosophy
in
Electrical and Computer Engineering**

© by Maciej Borkowski, 24 August 2007

Permission has been granted to the Library of the University of Manitoba to lend or sell copies of this dissertation to the National Library of Canada to microfilm this dissertation and to lend or sell copies of the film, and University Microfilms to publish an abstract of this dissertation.

The author reserves other publication rights, and neither the dissertation nor extensive abstracts from it may be printed or otherwise reproduced without the author's permission.

Abstract

This dissertation describes the design and implementation of a system that has been designed to extract 3D information from pairs of 2D images. System input consists of two images taken by an ordinary digital camera. System output is a full 3D model extracted from 2D images. There are no assumptions about the positions of the cameras during the time when the images are being taken, but the scene must not undergo any modifications.

The process of extracting 3D information from 2D images consists of three basic steps. First, point matching is performed. The main contribution of this step is the introduction of an approach to matching image segments in the context of an approximation space. The second step copes with the problem of estimating external camera parameters. The proposed solution to this problem uses 3D geometry rather than the fundamental matrix widely used in 2D to 3D conversion. In the proposed approach (DirectGS), the distances between reprojected rays for all image points are minimised. The contribution of the approach considered in this step is a definition of an optimal search space for solving the 2D to 3D conversion problem and introduction of an efficient algorithm that minimises reprojection error. In the third step, the problem of dense matching is considered. The contribution of this step is the introduction of a proposed approach to dense matching of 3D object structures that utilises the presence of points on lines in 3D space.

The theory and experiments developed for this dissertation demonstrate the usefulness of the proposed system in the process of digitizing 3D information. The main advantage of the proposed approach is its low cost, simplicity in use for an untrained user and the high precision of reconstructed objects.

Keywords: 2D to 3D conversion; 3D object structure; dense matching; camera parameters; epipolar geometry; minimisation; approximation space, genetic algorithm, image processing, 2D matching, rough sets, image segment.

Acknowledgements

I want to thank my supervisor Prof. J. F. Peters for his help and support. This dissertation would have never been finished without his valuable comments, directions and corrections. I also want to extend special thanks to my Ph.D. committee members Prof. D. Gunderson, Prof. W. Lehn and Prof. M. Pawlak for very helpful guidance, astute suggestions and incisive comments throughout my research project.

Special thanks are extended to my wife Hania who supported me in all aspects of my life for the last couple of years when I was working on this dissertation.

I would also like to acknowledge my parents who helped me focus on the work when I was doing a million other things.

I also wish to express my gratitude to the Natural Sciences and Engineering Research Council of Canada (NSERC) and Manitoba Hydro for funding this research project.

Contents

Abstract	iii
Acknowledgements	v
List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Motivation	3
1.2 Known Solutions	4
1.3 Organization of the Dissertation	6
2 Background Theory	8
2.1 Basic Definitions	8
2.2 Photogrammetry	10
2.3 Pinhole Camera Model	13
2.4 Camera Matrices	15
2.5 Camera Calibration	16
2.6 Image Processing Terminology	18
2.7 Hough Transform	19
2.8 Image Warping	21
2.9 Edge Detection	21
2.10 Nonlinear Diffusion	24

2.11	Fundamental Matrix	25
2.12	The Epipolar Geometry	27
2.13	Image Rectification	30
2.14	Rough Sets: Basic Concepts	31
2.15	Rough Set Theory	31
2.16	Approximation Spaces	35
2.17	Genetic Algorithms	36
2.18	Correlation	38
3	2D Image Processing	40
3.1	Coarse Matching	41
3.1.1	Image Quantization	47
3.1.2	Image Segmentation	51
3.1.3	Segment Selection	54
3.1.4	Feature Generation	58
3.1.5	Exhaustive Feature Matching	65
3.1.6	Single Point Standard (Upper Approximation)	70
3.1.7	Interval Standard (Lower Approximation)	74
3.1.8	Genetic Approach for Matching	79
3.1.9	2D Matching with Approximation Spaces	87
3.1.10	Tolerance Relation vs. Equivalence Relation	88
3.1.11	Classical vs. Rough Matching Methods	93
3.2	Point Matching	96
3.2.1	Interest Point Detection	97

3.2.2	Scale invariant interest point detectors	98
3.2.3	Rotation invariant interest point detectors	104
3.2.4	Affine invariant interest point detectors	108
3.2.5	Interest Point Descriptors	111
3.2.6	Matching	122
3.3	Dense Matching	127
3.3.1	3D Line Extraction	131
3.3.2	Dense Matching	134
3.3.3	Inside Point Detection	135
3.3.4	Matching	136
3.3.5	Merging Matches	137
3.3.6	Removing Outliers	138
3.3.7	Results	140
4	2D to 3D Conversion	142
4.1	Cost Function	144
4.1.1	Basic Definitions	145
4.1.2	Cost Function Derivation	153
4.1.3	Search Space	157
4.2	Error minimisation	166
4.3	Comparison with known methods	180
5	Conclusions and Future Work	184
A	Developed Applications	186

A.1	PointMatch	187
A.2	SPoints	189
A.3	VisualBatch	191
A.4	HypoDissertation	194
A.5	Parallel Computations - The Server	196
A.6	Parallel Computations - The Client	198
A.7	TwoViews	199
A.8	2DSearch	201
A.9	BiCuGPU	202
A.10	MeshReduction	206
A.11	TexturedView	208
A.12	UnrollLoops	209
	References	211
	Notation	233
	Glossary	236
	Index	243

List of Tables

1	Symbols used in Coarse Matching section.	41
2	Ranges of probe functions	65
3	Voting table for Algorithm 5	73
4	Colour table	77
5	Overlap table	78
6	\mathcal{Z} table vs. δ parameter	78
7	Probe functions and their corresponding abstract forms.	81
8	Rules for creating genes.	84
9	Symbols used in Coarse Matching section.	97
10	Scale factor estimation for square gradient.	102
11	Interest point detector point match detection rates.	121
12	Symbols used in 2D to 3D conversion algorithm.	145
13	List of all symbols used in the error minimisation algorithm	165
14	Domains of five parameters for location \mathbb{C}_2 for camera 2.	167
15	Comparison of 8-point algorithm and DirectGS.	182
16	Two sample nested “for” loops.	208

List of Figures

1	Overview of the 2D to 3D conversion process	4
2	Overview of the 2D to 3D conversion system.	6
3	Pinhole camera model (left) and camera lens system (right).	12
4	Two images of the same scene with simple camera (left) and more accurate (right)	12
5	Pinhole Camera Model	14
6	Camera calibration result (focal length set to 6 mm).	18
7	Black and white image (left) and corresponding Hough transform (right)	20
8	Sobel masks, horizontal G_x and vertical G_y	22
9	Original image (left) and the result of Sobel line detection (right). . .	23
10	Diffused image (left) and the result of Sobel line detection (right). . .	25
11	Epipolar geometry	28
12	Sample representation of B_*X (dark squares), B^*X (light squares). .	34
13	2D Image Segment Matching Steps	43
14	Image of street water pump (left) and result of 2-bit quantization (right).	50
15	Hydrant image after 7 iterations of (8) (left) and (9) (right)	53
16	Quantized images obtained by iterating (9)	55
17	The result of applying exponential into overlap function	61
18	Preliminary overlap of 2 segments (left), and best overlap (right) . . .	61
19	Three sample steps of segment matching.	62

20	The process of finding the best overlap for $\frac{ P_{one_seg} }{ P_{two_seg} }$ and <i>overlap</i> parameter.	64
21	Generated segments for the Wearever box scene.	70
22	Voting results. \circ good match, $+$ the closest match.	75
23	Voting results: \circ good match, $+$ the closest match.	75
24	Segments for Example 3	76
25	Voting results: ‘interval standard’ (left) and ‘single point standard’ (right).	80
26	The overview of the genetic algorithm.	83
27	The chromosome. Each block contains indices of matched shapes.	84
28	Rough coverage vs. ratio of correct matches for 2,000,000 chromosomes.	88
29	Ratio of correct matches for tolerance and equivalence relation	90
30	Rough coverage vs. ratio of correct matches (zoomed Figure 28).	94
31	Overview of Point Matching steps	97
32	Scale space filtering. Left and middle images - images taken with a zoom factor 1 and 2.88, respectively, right image - middle image rescaled with factor $\sigma = 2.88$	100
33	Harris corners	107
34	Affine invariant interest points	110
35	Results of SIFT detector matching.	122
36	Close-up of two insulators from Figure 35	123
37	Results of ZNCC matching.	125
38	Overview of the 2D to 3D conversion system.	129

39	Overview of the algorithm extracting 3D straight lines.	131
40	Image of an electric power tower and detected edges	133
41	Sample lines and points from line in LD space	134
42	Detected line in LD space and points from original image	135
43	Sample disparity map for the "jet plane" scene.	137
44	3D model of electric power tower	139
45	Overview of the 2D to 3D conversion system.	141
46	Pinhole Camera Model	146
47	Point $p = (x, y, z)$ represented in spherical coordinate system (θ, ϕ, R)	149
48	First camera view (at the origin of the coordinate system) and the second camera view on the sphere of radius R	159
49	Cost function for azimuth and altitude.	166
50	Illustration for the algorithm that determines the Ψ function.	174
51	Cost function minimisation algorithm overview.	179
52	Sample run for 11 point matches of DirectGS and 8-point algorithm.	181
53	PointMatch application screenshots	187
54	SPoints application screenshot	189
55	Sample script created in VisualBatch	191
56	Image quantization described in Section 3.1.1	191
57	Script for point matching described in Section 3.2.6	192
58	Script for segment matching described in Section 3	192
59	HypoDissertation application screenshots	194
60	RServer application screenshot	196

61	TwoViews application (control pane)	198
62	TwoViews application (main pane)	199
63	Closeup for several rays and their closest approach lines	199
64	2DSearch application screenshot	200
65	BiCuGPU application screenshot	202
66	BiCuGPU application screenshots of 8x magnification	202
67	Fragment of a 3D model of a power tower.	206
68	UnrollLoop application screenshot	209

1 Introduction

3D vision has a long history. Euclid is credited with discovering the principles of binocular vision. A 3D device called a stereoscope invented during the early 1830s by Sir Charles Wheatstone [15, 111, 189, 190] made it possible to view a different image with each eye. Photography allowed people to capture images with cameras separated by the same distance as human eyes. The stereoscope made it possible to view objects in such a way that the brain would create a 3D image. The invention of the digital camera made it possible to capture images with remarkable precision. Additionally, such cameras made it easier to extract 3D information coded in flat, 2D images.

Considerable work has been done on solving the problem of extracting 3D information from multiple 2D images. The most common approach uses what is known as epipolar geometry [31, 61, 195]. In this work, the focus is on facilitating 3D descriptions of objects based on 2D images obtained by one or more digital cameras. Usually, there are no constraints on camera position in capturing 2D images. For example, 2D images are obtained with a single camera that records the movements of an object on a rotating turntable (see, *e.g.*, [30]), or two cameras are used with lateral movement (see, *e.g.*, [45]).

In this research, 2D images are obtained by movement of a camera along a path determined by an electric power transmission line. In effect, this research represents a specialization of 2D to 3D image processing methods relative to the problem of stereovision in a mobile robot equipped with one or more cameras. At the same time, there are no limitations on the type of movement the camera undergoes between the

locations where the images are taken. It should be mentioned that this work¹ is a continuation of research that started earlier with the study of 2D camera images [7, 135, 136] that resulted from studies of applications of rough sets [8, 12, 131] based on both classical rough set theory [79] and minor extensions of rough set theory [119, 120, 121, 137]. The focus of this dissertation is on matching points from given images and conversion from 2D images to 3D objects.

The process of acquiring 3D information from the surrounding environment can be classified according to three groups of methods [73]:

- Active or Passive,
- Image Based or Direct,
- Monocular or Multiple View.

An active 3D acquisition group includes methods that introduce a structured source of energy such as light or ultrasonic waves. Passive methods are non-invasive, *i.e.*, the process of recording an image does not alter the surrounding environment in any way. Note that a camera flash unit does not emit structured light, but the direction of illumination (actually the location of shadows) changes with the location of a camera. Therefore, the use of a camera flash can be seen as a factor that alters the environment and with one exception is not considered in this research².

The second group divides the acquisition methods into ones that collect data in form of images and direct methods that do not require such data representation like

¹Point matching algorithms

²The exception is the case where a flash unit is not moving and is independent from the cameras begin used.

range sensors. This project is classified as image based method.

The monocular 3D acquisition method defines a number of views used for the data acquisition process. Interestingly, 3D information can be extracted from a single view. But this does not mean that one image is sufficient to recover the depth. For example, in the case of a range-from-focus approach, several images are taken from one view. For each image, the focus is different. By identifying the sharpest areas in each image, one can obtain information about depth of an underlying scene. Other monocular methods like range from brightness, attenuation or texture use some extra information that must be known *a priori*. This research is classified into the multiple view category. The depth information is retrieved from the differences between images taken from different vantage points in 3D space.

1.1 Motivation

Briefly, the motivation for this research is the need to use image processing to detect deformations in various 3D structures that are common in electric power transmission systems, *e.g.*, steel towers, wooden towers, cross beams, insulators, insulator pins, vibration dampers and so on. The proposed approach to 2D to 3D image processing makes it easier to impose constraints on camera movement. Usually, the search space for 2D to 3D image processing is quite large (*i.e.*, at least 6 dimensions [31, 61, 195]). As a result of the proposed approach, the size of the search space is reduced by one dimension. The benefit of this approach means faster calculations, which is important for robotic inspection of power transmission equipment that is currently being investigated.

In addition, there is no single approach to pixel matching (*i.e.*, discovery of correspondence between pixels), which is needed before 3D information can be extracted from 2D images. In this research, the matching problem is solved using approximation spaces, which underly the basic idea of classical rough sets [79].

1.2 Known Solutions

In this section, a short description of known methods for 2D to 3D conversion is presented as way of establishing a framework for this research.

In general, the process of extracting 3D information from 2D images is performed in two steps. First, the image registration is performed [35, 201]. This operation is performed only in the 2D image domain. Its purpose is to match pixels from two or more images that represent the same point in 3D space. In the second step, the positions of cameras and the location of the points in 3D space are found. This process is depicted in Figure 1.

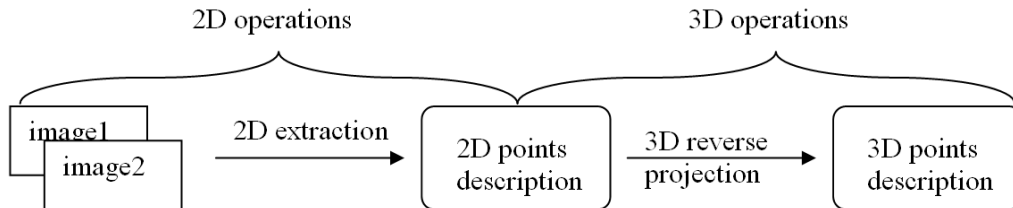


Figure 1: Overview of the 2D to 3D conversion process

The first step in Figure 1 entails finding corresponding pixels in pairs of images. The solution to the problem of identifying corresponding pixels is still being investigated by many researches [201]. At the 2D level, considerable work has been done using various kinds of interest point detectors [153], especially the Harris detector

[74, 113, 175, 177] and differential invariants [50, 113]. The most commonly used method used to find corresponding pixels is the cross-correlation [99].

The most common approach to extracting 3D information from two or more 2D images is by using the epipolar geometry [31, 61, 195] and the fundamental matrix [19, 31, 51, 52, 61, 147, 179, 182]. Pruning outliers (removing false matches from the registration step) is usually achieved by the RANSAC algorithm [34], which can also be used for post processing of extracted 3D models [18].

Some solutions make use of *a priori* knowledge about the scene being reconstructed, such as the parallelism of the walls [18], existence of corners in the scene [92] or the limited movement of the camera [99].

An example of a working system used to extract 3D information from 2D images was introduced by Philip H. S. Torr [178]. Torr's work concentrated on motion segmentation and determining the maximum amount of information that can be gained from two or three flat images. 3D information is extracted from images taken from large sequences of images (movie frames). The differences between the images were small enough to permit image registration using the Harris corner detector and cross-correlation over 9×9 pixel windows. To extract 3D information, Torr estimated the fundamental matrix using an algorithm based on RANSAC.

The main difficulty with most of the algorithms that estimate the fundamental matrix is that while solving for the fundamental matrix one usually does not minimise a physical quantity. Such an approach is prone to special solutions, which are not feasible in practise, but are valid mathematical solutions of the equations. This problem can be solved by formulating the 2D to 3D conversion problem in such a

way that the solution includes minimisation of a physical quantity. An example of such a formulation is the *nonlinear method that minimizes distances between observation and reprojection* [195, 200]. The idea behind this method is to minimize the distance between the selected image points and their reprojections. This approach requires the calculation of the fundamental matrix, the retrieval of 3D coordinates of all points of interest and the reprojection of these points back into an image.

Matthies *et al.* in [99] proposed a new method for dense image registration based on the Kalman filter. The matching of pixels was performed using the sum of squared differences. The authors concentrated only on lateral movement of the cameras. This is an example of a special case known in the literature as *stereovision* [68]. Usually, in stereovision, it is assumed that one camera is translated by a movement with respect to a second camera that is perpendicular to the optical axis of both cameras (lateral movement).

1.3 Organization of the Dissertation

This section briefly describes the organization of this dissertation. The dissertation consists of three main steps (see Figure 2). Each step corresponds to one section in the dissertation.

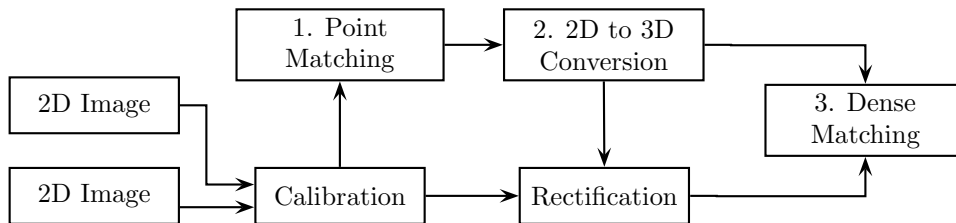


Figure 2: Overview of the 2D to 3D conversion system.

Section 2 gives a concise description of the background topics used for this research. The *calibration* step in Figure 2 denotes the process of undistorting images. The process of removing the distortions caused by the camera can be performed only after the camera is calibrated (see Section 2.5). Finally, the rectification step aligns the epipolar lines with the horizontal edges of both images (see Section 2.13). The rectification step is performed before the dense matching.

In Section 3, 2D image processing methods used in this research are presented. This section consists of three main parts. In Section 3.1, a coarse matching is presented. This step aims at detecting segments in both images and detecting the scale difference and rotation angles between them. Section 3.2 gives a review of existing methods for point matching. This is followed by a description of an algorithm used for matching points from pairs of 2D images. An approach to dense matching is presented in Section 3.3. Dense matching utilises the presence of points on lines in 3D space and relies on information about camera locations and orientations. In practise, dense matching is performed after 2D to 3D conversion.

Section 4 presents the core of 2D to 3D conversion. It incrementally describes the design of a 2D to 3D conversion algorithm. In Section 4.1, a cost function is derived. This section also presents the definitions and mathematics needed for the derivation of a 2D to 3D conversion algorithm. Section 4.2 contains a description of an optimal algorithm for minimisation of the cost function.

2 Background Theory

The basic definitions of technical terms as well as the fundamental methods and theories underlying the approach to 2D to 3D conversion are presented in this section of the dissertation.

2.1 Basic Definitions

In this dissertation, all vectors and point coordinates are always given as column matrices. For example, if P is a point in 3D space, then P is represented as:

$$P = [x \ y \ z]^T = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

The symbol \mathbb{R} denotes the set of all real numbers. If M is a matrix then let m_{ij} denote an element of M , where i denotes the row and j denotes the column of M .

Definition 1. A square matrix M is called a singular matrix if it is not invertible, i.e., if no matrix M^{-1} exists such that $MM^{-1} = M^{-1}M = I$. A determinant of a singular matrix is equal to zero.

Definition 2. A non-singular matrix is a matrix that is not singular.

Definition 3. The Mahalanobis distance is a measure of the distance between two points $x, y \in \mathbb{R}^n$ given by

$$d_M(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)},$$

where Σ is any non-singular n dimensional matrix.

Usually, one of the points denotes a mean value of a set of points derived from a normal distribution and Σ is a covariance matrix of a given normal distribution. In such case the Mahalanobis distance measures the distance from one given point to the centre of mass of a given distribution relative to the scatter of points from the distribution.

The Euclidean distance is a special case of Mahalanobis distance, where the matrix Σ is identity matrix.

Next, definitions of transformations used in this dissertation are presented [61].

Definition 4. A linear transformation is a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that preserves addition and scalar multiplication. For $X, Y \in \mathbb{R}^n$ and $a \in \mathbb{R}$, a function f is linear if and only if

$$f(X + Y) = f(X) + f(Y) \quad \text{and} \quad f(aX) = af(X).$$

Definition 5. An affine transformation \mathcal{A} is a function $\mathcal{A} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ that preserves collinearity and ratios of distances. In n dimensional space, an affine transformation is described by a non-singular matrix $A \in \mathbb{R}^n \times \mathbb{R}^n$ and a vector $T \in \mathbb{R}^n$. For any $X \in \mathbb{R}^n$, an affine transformation \mathcal{A} is given by

$$\mathcal{A}(X) = AX + T. \tag{1}$$

Any point in $X = [x_1 \ x_2 \ \dots \ x_n] \in \mathbb{R}^n$ can be represented in *homogeneous coordinates* as $[\tilde{x}_1 \ \tilde{x}_2 \ \dots \ \tilde{x}_n \ \tilde{x}_{n+1}] \in \mathbb{R}^{n+1}$. For $x_{n+1} \neq 0$ the conversion between

homogeneous coordinates and Cartesian coordinates is performed using

$$x_1 = \frac{\tilde{x}_1}{\tilde{x}_{n+1}} \quad x_2 = \frac{\tilde{x}_2}{\tilde{x}_{n+1}} \quad \dots \quad x_n = \frac{\tilde{x}_n}{\tilde{x}_{n+1}}.$$

A point in homogeneous coordinates such that $x_{n+1} = 0$ represents a point at infinity. Homogeneous coordinates are often used in context of the epipolar geometry (see Section 2.12).

2.2 Photogrammetry

The problem of extracting 3D information from images has its roots in photogrammetry³ [25]. Photogrammetry was developed and mainly used as a method for measuring real-life objects based on images containing objects of interest. Originally, photogrammetry did not use either computers or digital cameras. The main task for photogrammetry is to recover real dimensions of objects being photographed. The problem is the deformation of an object in an image caused by the way light traverses through the camera elements. This section contains a description of important topics from photogrammetry that apply to this project. This section is based on [25] and [192].

The process of image creation assumed in this project is that described by the pinhole camera model (see Section 2.3 for more details). In practise though, due to the camera construction, the assumptions about the pinhole camera model are violated. A *pinhole camera model* is a simplified mathematical model of how images

³Etymology: *photogram-* photograph (from *phot-* + *-gram*) + *-metry*; the science of making reliable measurements by the use of photographs, from [205].

are created. In practise, the aperture of a pinhole camera would have to have infinitely small diameter. This makes the exposure of images impossible. Instead, glass lenses are used that are able to catch a bundle of rays and focus them on one point in an image plane. This makes it possible to form an image on light-sensitive film inside a camera.

The disadvantage of using glass lenses is that a ray of light coming from an object refracts several times. The angle at which a ray of light enters a lens is not the same as the angle at which the ray of light leaves the lens. As a result, one cannot select one point and consider it the centre of the projection and points in the image are recorded with different focal lengths (see Figure 3). The actual focal length f_θ depends on the angle between a principal ray and a given object. The greater the angle θ , the bigger the distortion of an image. In the left part of Figure 4, the image produced by an inexpensive web camera⁴ is shown. The lens distortion causes straight lines to be recorded as non-straight lines. This effect becomes more severe as the distance from a principal point increases. The right part of Figure 4 shows an image taken with a digital camera equipped with a Sony Mavica Carl-Zeiss lens system⁵. The accuracy of the Mavica image is much better than the WebCam camera image in Figure 4. The superiority of the Mavica image is due to the fact that the Mavica camera has a bigger focal length. The resulting Mavica image has a geometry similar to that obtained when using the pinhole camera model. Nevertheless, even images produced by high quality cameras contain some distortions and these distortions need to be removed in order to obtain reliable

⁴Creative® WebCam Ultra NX, resolution 640x480 pixels.

⁵SONY® CD Mavica MVC-CD300 Digital Still Camera, 640x480 pixels.

results in 2D to 3D conversion.

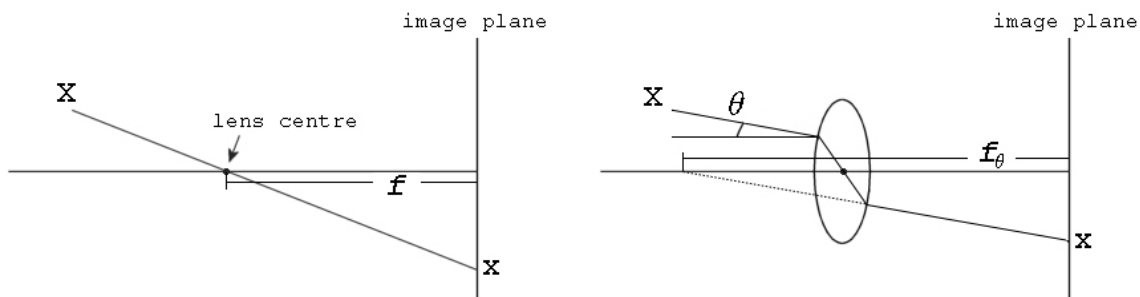


Figure 3: Pinhole camera model (left) and camera lens system (right).

The distortions due to glass lens are corrected using photogrammetry techniques. This process is called camera calibration. To produce an accurate image, a camera is calibrated and an image is corrected. The literature is rich in camera calibration techniques that do not impose conditions on how the images are to be taken [169, 182].



Figure 4: Two images of the same scene with simple camera (left) and more accurate (right)

For this project, full camera calibration is described (see Section 2.5). The reason

for this is that despite the fact that the most common application of the system is to process images taken using more accurate digital cameras, such cameras produce slightly distorted images. The precision required for this project is a crucial factor and small distortions produced by high quality off-the-shelf digital cameras are still a concern.

2.3 Pinhole Camera Model

Geometric equations used for converting flat images into a 3D space assume that images are obtained using a camera that conforms to the pinhole camera model. In practise, digital camera construction violates this assumption and the resulting images contain distortions. The most common result of these distortions is that straight lines are mapped into curves. Section 2.5 describes a standard camera calibration method used to remove these distortions. The notation used here is based on [25], [26] and [192].

Figure 5 shows the *pinhole camera model* (also called *central projection*). The *image plane* is shown on the left-hand side of Figure 5. In digital cameras, the image plane is a CCD (Charged-Coupled Device) matrix used to record an image. By contrast, in classical cameras, images are recorded on light-sensitive film. Regardless of the media, the image plane is considered to be the actual image containing a view of an observed scene. The straight line connecting each point P_i in 3D space with its corresponding image point p_i is called the *projecting ray*. The point where the projecting ray crosses the image plane p_i is described by the coordinates of the pixel representing given point P_i in 3D space. Each projecting ray goes through a special

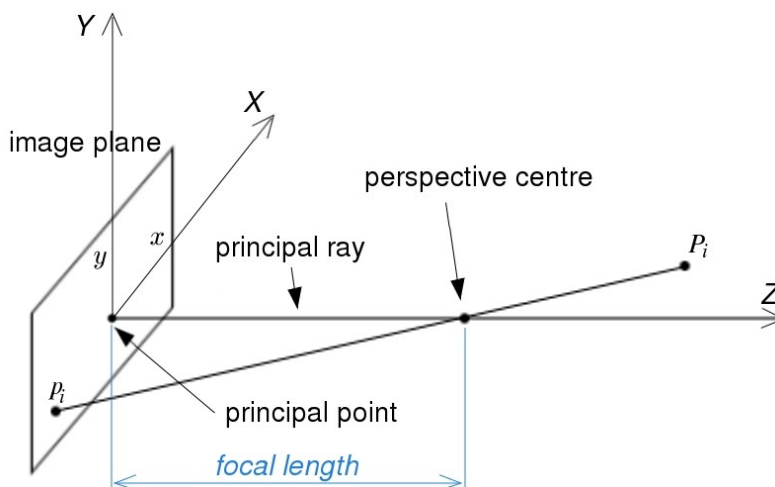


Figure 5: Pinhole Camera Model

point called the *perspective centre*, the *lens centre* or the *pinhole lens*. The projecting ray that is perpendicular to the image plane is called the *principal ray*. It crosses the image plane at a point called the *principal point*. The distance f between the image plane and the perspective centre is called the *focal length*. The focal length controls the width of a camera view. The shorter the focal length the wider the camera angle, but also the bigger the distortion of an image due to perspective transformation.

Perspective transformation is the transformation that converts coordinates of a 3D point P_i into image plane coordinates p_i . For the sake of simplicity, assume that the world Z coordinate is aligned with the principal ray as shown in Figure 5. The x and y coordinates of the image plane correspond to world X and Y coordinates. Assuming that the origin of the world coordinate system is placed at the perspective centre, the focal length equals f and the coordinates of point P_i are (X_i, Y_i, Z_i) , the

coordinates of a point $p_i = (x_i, y_i)$ are given by

$$x_i = \frac{fX_i}{Z_i}, \quad y_i = \frac{fY_i}{Z_i}.$$

2.4 Camera Matrices

In this section, image mapping from 3D space to a 2D image by a pinhole camera is formally described using matrix notation.

Let k denote an aspect ratio, α a magnification parameter, v_x and v_y a principal point coordinates and s a skew parameter. These parameters do not depend on camera location and orientation (the so-called *internal camera parameters* or *intrinsic parameters*) and they form the calibration matrix

$$\mathbf{K} = \begin{bmatrix} k\alpha & s & v_x \\ 0 & \alpha & v_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (2)$$

Let $\mathbf{R} \in \mathbb{R}^3 \times \mathbb{R}^3$ denote a camera rotation matrix (as defined in (25)). The matrix \mathbf{R} due to Euler's theorem⁶ depends only on three parameters - rotation angles along three coordinates X, Y and Z . Let $T \in \mathbb{R}^3$ denote a translation of the lens centre of a given camera in 3D space (each component of a vector T is responsible for translation along different axis). The three rotation parameters defining the rotation matrix \mathbf{R} and three translation parameters T are called the *external camera parameters* (or *extrinsic parameters*) [22, 61].

⁶See subsection *Rotations In 3D Space* of Section 4 for more details about Euler's theorem.

A camera projection matrix $P \in \mathbb{R}^3 \times \mathbb{R}^4$ is build from internal and external camera parameters [61, 182].

$$P = K \left[R \mid T \right].$$

Let $P_i \in \mathbb{R}^3$ denote a point in 3D space and $p_i \in \mathbb{R}^2$ denote the corresponding point in the image plane. The relation between these two points is shown in

$$\begin{bmatrix} p_i \\ 1 \end{bmatrix} = P \begin{bmatrix} P_i \\ 1 \end{bmatrix}.$$

2.5 Camera Calibration

Due to the fact that digital cameras do not conform to the pinhole camera model, it is necessary to transform digital images. This process (called *camera calibration*) consists of two steps. First, all internal camera parameters describing the actual camera model are identified. Then each digital image is transformed to the required format.

The first step of camera calibration process requires several images of a shape with known geometry. In the software used in [169] the calibration images show a planar checkerboard. Each image is taken from a different angle. Then all the corners in the checkerboard are identified and the internal parameters of a camera are calculated. The model used in the calculations takes into account the focal length of the camera, the principal point and skew coefficient (see (2)) as well as radial and tangential distortion coefficients k_1, k_2, \dots, k_5 (see, *e.g.*, [49, 63]). An

image point with coordinates (x, y) is mapped to a point (x_{un}, y_{un}) in undistorted image using the transformation

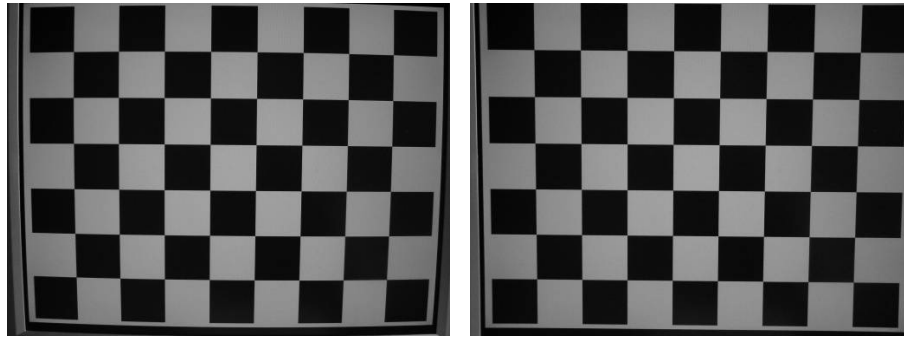
$$\begin{bmatrix} x_{un} \\ y_{un} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2k_3 xy + k_4(r^2 + 2x) \\ k_3(r^2 + 2y) + 2k_4 xy \end{bmatrix},$$

where: $r^2 = x^2 + y^2$.

After all parameters have been identified, an inverse mapping from a distorted to an undistorted image is found. Having found⁷ this transformation, all the images taken by the camera need to be transformed to the undistorted form. The undistorted image will then conform to the pinhole camera model. For experiments for this dissertation, only undistorted versions of images have been used.

The Figure 6 shows the result of calibrating the image taken by the digital camera with the focal length set to 6 mm. The distortion of the original image is visible in the corners of Figure 6(a). Straight lines in 3D space should be mapped to a straight lines in the image plane. But, as can be seen in Figure 6(b), the lines forming a checkerboard are bent. In Figure 6(b) the image from 6(a) was undistorted using the information about the internal camera parameters. This image conforms to the pinhole camera model.

⁷Calibration parameters are found by solving Direct Linear Transform using Levenberg-Marquardt optimization method, see [63] for more details.



(a) Original image

(b) Calibrated image

Figure 6: Camera calibration result (focal length set to 6 mm).

2.6 Image Processing Terminology

This section gives an introduction to the basic definitions of technical terms associated with the classical approach to 2D matching images.

Definition 6. ([48, 62, 94]) **Pixel.** A pixel (also referred to as an *image element*, *picture element*, or *pel*) is the smallest component (a point) of a digital image. A pixel has its fixed location and a colour that can be modified.

In a digital image, a pixel is an element that has a numerical value that represents a greyscale (256 possible values) or RGB intensity value ($256^3 = 16,777,216$ possible values).

Definition 7. ([48, 94]) A **4-Neighbourhood** of a pixel p with coordinates (x, y) is a set of 4 pixels (2 vertical and 2 horizontal neighbours) at coordinates

$$(x - 1, y), (x + 1, y), (x, y - 1), (x, y + 1).$$

Definition 8. ([48]) An **8-Neighbourhood** of a pixel p with coordinates (x, y) is

a set of 8 pixels at coordinates

$$(x - 1, y - 1), (x - 1, y), (x - 1, y + 1), (x, y - 1),$$
$$(x, y + 1), (x + 1, y - 1), (x + 1, y), (x + 1, y + 1).$$

Definition 9. Segment. A segment is a collection of 4-neighbourhood connected pixels that have the same colour.

Definition 10. Image Segmentation. An image segmentation is a process of identifying segments in a digital image. After image segmentation each pixel from an image belongs to exactly one segment.

Definition 11. 2D Convolution Let $I \in \mathbb{R}^k \times \mathbb{R}^l$ be an 2D image and $M \in \mathbb{R}^n \times \mathbb{R}^m$ 2D mask⁸. A 2D convolution of image I with a mask M is a linear combination of elements from mask M and image I

$$Conv(x, y) = \sum_{i=1}^n \sum_{j=1}^m I(x - n/2 + i, y - m/2 + j)M(i, j)$$

2.7 Hough Transform

The most popular algorithm in digital image processing for detecting lines is Hough transform [48, 69]. It transforms points from Cartesian coordinates into sinusoidal functions on the plane. Lines from Cartesian space correspond to intersections of sinusoidal functions in the Hough space. Each point in the Hough space is a counter of points from the original image, which are part of a line defined by a given point

⁸Usually the dimensions of mask M are much smaller than dimensions of the image I .

in the Hough space. For example, in the left part of Figure 7 black and white image is shown. All white points from this image were transformed to Hough space. The right side of this figure shows the same image, but in the Hough space.

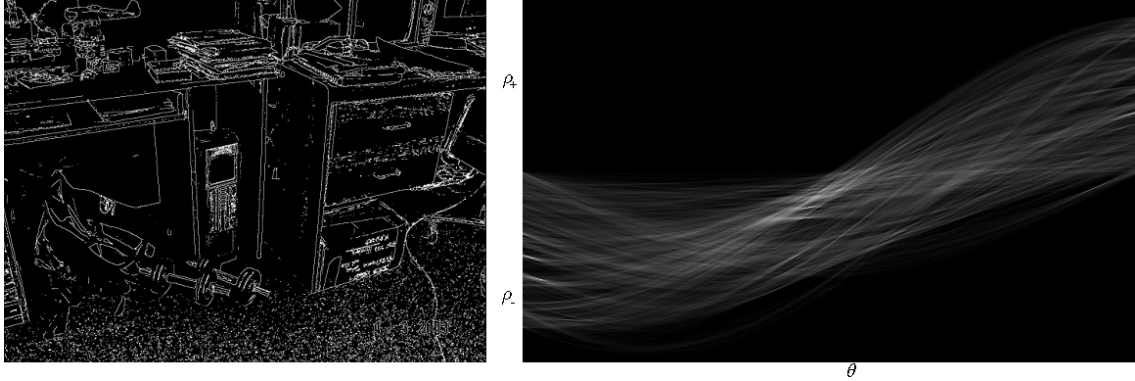


Figure 7: Black and white image (left) and corresponding Hough transform (right)

Let denote by (x_i, y_i) coordinates of an i -th pixel from a given image. The conversion from image space to Hough space (θ, ρ) is given by

$$\rho = x_i \cdot \cos(\theta) + y_i \cdot \sin(\theta).$$

The parameters θ and ρ describe the location of the line from the original image. The ρ parameter is equal to the distance between the line and the origin and θ defines the angle with which the line crosses the vertical axis.

2.8 Image Warping

Image warping is an algorithm for applying geometrical transformations to digital images. Geometrical transformations are used to remove (or add) unwanted geo-

metrical distortions from an image. These transformations are defined by selecting a mesh of points, called tiepoints. One set of tiepoints is located in the input (distorted) image and the second set in the output (corrected) image. A mapping between input and output images is exact at specified tiepoints. If we denote the coordinates of a pixel in the corrected image by (x, y) and coordinates of a pixel in the input image by (\hat{x}, \hat{y}) then the First Order Warp equation [48] is defined by

$$\hat{x} = c_1x + c_2y + c_3xy + c_4,$$

$$\hat{y} = d_1x + d_2y + d_3xy + d_4,$$

and $c_1, c_2, c_3, c_4, d_1, d_2, d_3$ and d_4 are real numbers. First, these parameters are calculated from the above equations and known tiepoints. Then the image is warped using the same set of equations shown above.

In certain cases the warping transformation can be defined not using a tiepoints but explicitly by a transformation matrix.

2.9 Edge Detection

Edge point detection aims at identifying pixels with high variability. Usually, images containing an edge of a 3D shape is represented with areas of different colours on both sides of the edge. In most cases, this results from different lightning conditions on both sides of an edge. For similar reasons, flat areas are represented in an image by uniform patches. These patches do not contain much relevant information relative to the shape of a particular object.

A Sobel filter is a gradient based edge detector that detects edges in an image ([167] as cited by [48]). The Sobel filter consists of two masks that are convolved with the image (see Figure 8).

$$G_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad G_y = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Figure 8: Sobel masks, horizontal G_x and vertical G_y

The application of the Sobel filter results in high values for pixels with high changes of colour in their neighbourhood. In the area where colour values are approximately at the same level, the Sobel filter produces small values. But any digital image is hardly ever ideally uniform, and as a result there are always some edges present in the areas representing flat surfaces.

The Sobel filter works on grey-scaled images, therefore, before detecting the lines the image is converted to grey-scale levels. The formula used for conversion uses the guidelines from [203]. The formula

$$I = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B,$$

converts the RGB colours into a grey-scale in a more natural way for human vision system than the $(R + G + B)/3$ formula.

Denote by $G_x(I)$ the result of convolving the mask G_x with an image I . An image I_{sobel} resulting from applying a Sobel filter to an image I is the square root

of sum of two squared convolved images, *i.e.*,

$$I_{sobel} = \sqrt{G_x^2(I) + G_y^2(I)}.$$

An example of the edge detection result is shown in Figure 9. Edges detected by the Sobel filter are denoted in dark colour. As can be seen, the edge detector selected many points on the flat surface that should not be detected. One solution to this problem could be the thresholding of the image with low threshold, but it would also remove some important parts of the image. What in fact is needed, is a way of removing small details from the areas of low variability, while keeping the sharp edges. The solution for this problem is a "nonlinear diffusion" that is discussed in the next section.

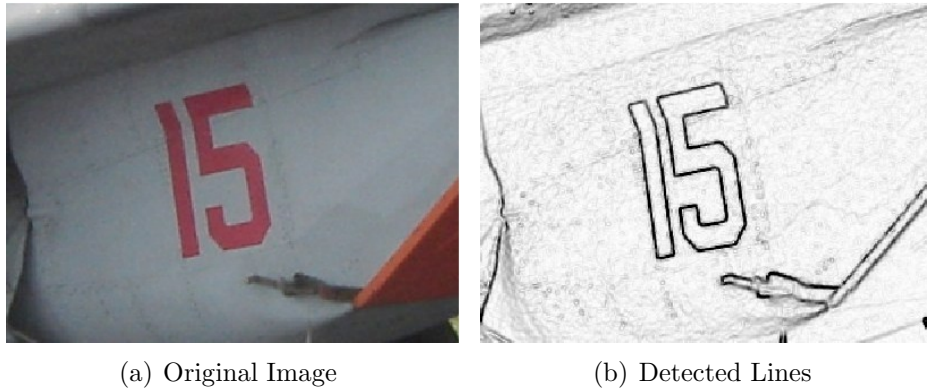


Figure 9: Original image (left) and the result of Sobel line detection (right).

2.10 Nonlinear Diffusion

The idea behind nonlinear diffusion is to smooth relatively uniform areas of the image while keeping the edges. It is based on a physical phenomenon called *diffusion*, a process during which a change of concentration happens spontaneously so that a steady-state of concentration is reached. For image processing applications, the concentration is interpreted as pixel value. During the diffusion process, pixel values are used as a quantities for which an equilibrium is sought. Colours of neighbouring pixels are being mixed and an equilibrium means no change in colour and results in smoother image.

The diffusion process is modelled by a *diffusion equation* [185], also known as *heat equation*

$$\partial_t I = \Delta(D \cdot \nabla I),$$

where t denotes time, I is an image and D is a "diffusion coefficient". The coefficient D controls the diffusion speed. It is usually set as a function of a gradient of an image, see [128, 185, 186]. The gradient is responsible for detecting edges (see the discussion of Sobel filter). By setting D to be a function of the gradient one can control the diffusion speed depending on the proximity of the pixels to the edges. The function suggested by Perona and Malik [128] is

$$g(s^2) = \frac{1}{1 + s^2/\lambda^2}, \quad \lambda > 0,$$

where

$$D = g(\|\nabla I\|^2).$$

As shown in the above equations the diffusion coefficient D is inversely proportional to the square root of the gradient. This means that the diffusion is slow for the pixels belonging to the edges and fast for the pixels belonging to more uniform areas. In what follows, areas with small variability are smoothed and edges are kept unchanged. As a result, the image is smoothed except for the edges. The result of applying the nonlinear diffusion filter to the left part of the image from Figure 9 is shown in Figure 10. The diffusion was performed using the code by Frederico D’Almeida available at [204].

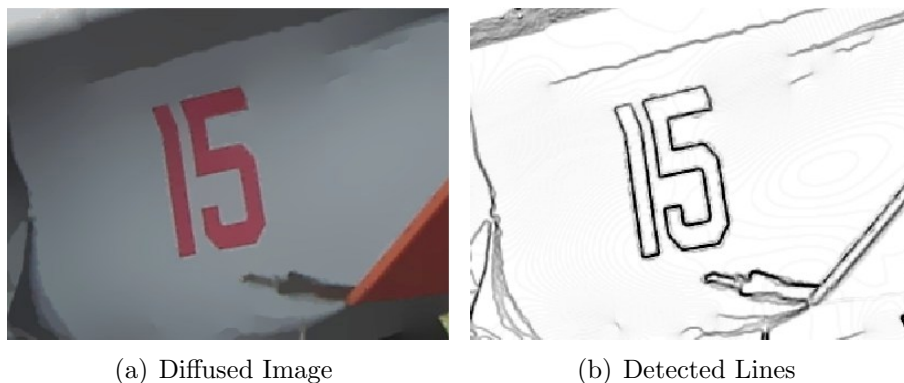


Figure 10: Diffused image (left) and the result of Sobel line detection (right).

2.11 Fundamental Matrix

Denote by $p_{1,i} \in \mathbb{R}^2$ a point in one image and by $p_{2,i} \in \mathbb{R}^2$ its corresponding match in a second image. The fundamental matrix $F \in \mathbb{R}^3 \times \mathbb{R}^3$ is a matrix that satisfies the equation

$$\begin{bmatrix} p_{2,i} \\ 1 \end{bmatrix} F \begin{bmatrix} p_{1,i} \\ 1 \end{bmatrix} = 0, \quad (3)$$

for all point matches.

The fundamental matrix contains information about the relative position and orientation of two camera views. Therefore, given enough matched points, one can calculate the fundamental matrix F using (3). Knowing F , it is possible to reconstruct a 3D scene from given images (see, *e.g.*, [195]).

Estimation of the fundamental matrix is the most common approach for determining the external camera parameters. The popularity of the fundamental matrix approach lies in the fact that it can be used to specify a correspondence between points from two given stereo images without the need to know the projection matrices. The problem of estimating the fundamental matrix is rather complex. There have been many attempts to solve this problem and there are many well studied solutions. Zhang [195] reviews many known methods that fall into the following groups.

- Exact solution with 7 point matches;
- Analytic method with 8 or more point matches;
- Analytic method with rank-2 constraint;
- Nonlinear method minimising distances of points to epipolar lines;
- Gradient-based technique;
- Nonlinear method minimising distances between observation and reprojection.

The fundamental matrix is a singular matrix and has only seven degrees of freedom. This means that two values in the fundamental matrix depend on other seven pa-

rameters. By directly solving (3), one obtains the solution for the nine unknowns. Different algorithms try to constrain external requirements to decrease the space of possible solutions (see, *e.g.*, [33, 199]).

2.12 The Epipolar Geometry

The description of the epipolar geometry presented in this section is based on [61]. This section contains only necessary terms needed to explain how the epipolar geometry is used to perform the dense matching.

The Epipolar Geometry Theory

The epipolar geometry is the geometry between two two-dimensional views of the same 3D scene. The main notion of the epipolar geometry is the fundamental matrix [33, 60, 61, 195, 199, 200] that links the coordinates of two views of one point from 3D space. In order to derive this correlation, consider the scene shown in Figure 11 (based on Figure 9.1 from [61]). This figure shows two pinhole cameras C_1 and C_2 , two corresponding image planes I_1 and I_2 , and a point P in 3D space ($P \in \mathbb{R}^3$). The line connecting two cameras' centres is called a *baseline*. A plane containing the baseline and a point P is called the *epipolar plane* π . The point P is mapped into both image planes. In the first view it maps to a point p_1 and in the second view to a point p_2 .

First, consider the case, when only the first mapped point p_1 is known. The point P_i lies on the line passing through the first camera centre C_1 and an image point p_1 , see Figure 11. In other words, point P_i is a point p_1 reprojected back into

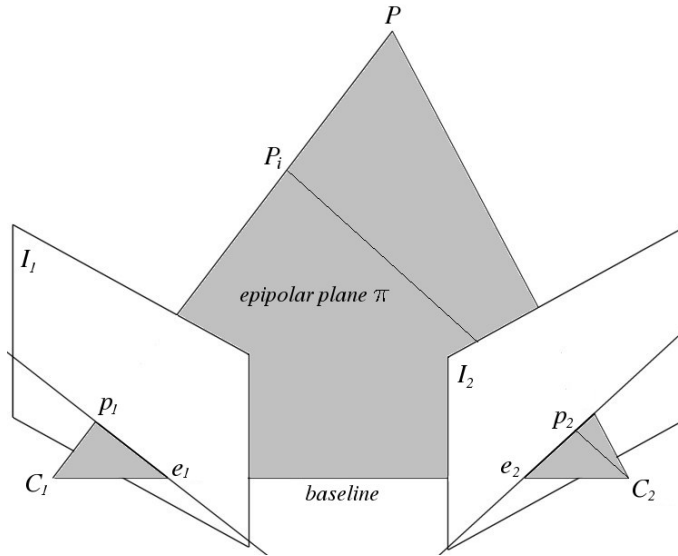


Figure 11: Epipolar geometry

the 3D space. The location of the point P_i in 3D space is not unique. This point can be located anywhere on the line determined by vector $\vec{C_1 p_1}$. One view does not give enough information to determine the location of this point (since it is 2D to 3D mapping). Now, consider where this point is mapped on the second image plane. The line connecting the lens centre C_2 and the point P_i is contained in the epipolar plane π (since both ends of the vector $\vec{C_2 P_i}$ are contained in π). Therefore, the mapping of point P_i on the image plane I_2 lies on the line that is an intersection of epipolar plane π and image plane I_2 . This line is called *epipolar line* and is of a main interest from the point of view of point matching. All epipolar lines intersect in one point called the *epipole*, denoted by e_1 for the first view and e_2 for the second view. This is also a point in which the baseline intersects with both image planes. Notice, the epipole may, but does not necessarily need to lie within image plane of

a camera.

The coordinates of the two points p_1 and p_2 are correlated via the fundamental matrix, see (3).

Application of Epipolar Geometry to Dense Matching

The epipolar geometry can be used as a means in the retrieval of the relative positions of the two cameras. Another popular application is called "dense matching" [177] (see Section 3.3). For this research, the epipolar geometry is used for dense matching. Before the epipolar geometry can be used, the relative position of the two cameras in the 3D space must be known. This implies that the fundamental matrix F is known as well. Using (3) one can calculate the epipolar line in the second view for each point from the first view. The goal is to reduce the search space. In order to reconstruct the 3D scene, one must have the information about the positions of all pairs of points from both images. One of the problems in coarse matching is a big search space. For each pixel from the first view, its corresponding pixel in the second view is searched in two dimensional space. By using (3) one can reduce the dimensionality of the search space from two to one. It is possible because the matching point must lie somewhere on the epipolar line that can be found if the fundamental matrix is known. After determining the epipolar line the problem of matching a given pixel from the first view with the point on the epipolar line is still open. But using epipolar constraint makes this task much simpler and can be handled using correlation. In addition, the error of the match along the epipolar line can be parametrized using a parabola, allowing for obtaining sub-pixel resolution

[99].

2.13 Image Rectification

Image rectification (sometimes called *the epipolar rectification*) is the process of determining a mapping for two given images such that corresponding epipolar lines become collinear and parallel to the horizontal image axes [41].

After the rectification, all corresponding epipolar lines are placed at the same level, *i.e.*, two corresponding epipolar lines have the same vertical position. As a result, images are transformed to a stereo vision view, *i.e.*, images from two cameras are displaced only laterally [2]. This transformation is very important, because it simplifies the problem of dense matching. After finding the epipolar lines, the problem of matching pixels is reduced from 2 dimensions to 1 dimension, where each matching point lies on an epipolar line that is a 1-dimensional subspace of the 2-dimensional space. Parametrization of a line is 1-dimensional, but in general, when moving along a line, both coordinates of a point belonging to a line change. After rectifying the images, the vertical coordinates of matched points do not change and the search for matching point from the second image can be performed by varying the horizontal coordinates only.

Additionally, there are no special requirements for the cameras' locations and orientation for the rectification to be possible. Fusiello et al. [41] show that rectification can always be performed, except for the case where movement of the camera is parallel to the principal ray of the camera.

2.14 Rough Sets: Basic Concepts

This section briefly presents some fundamental concepts in rough set theory that provide a foundation for the image processing described in this dissertation. In addition, a brief introduction to approximation spaces is also given, since approximation spaces are used to solve the 2D matching problem. First, definitions of tolerance and equivalence relations are given.

Definition 12. Tolerance Relation (from [79])

A binary relation $\tau \subseteq X \times X$ is called a tolerance relation if and only if τ is

1. reflexive, an object is in relation with itself $x\tau x$,
2. symmetric, if $x\tau y$ then $y\tau x$.

Definition 13. Equivalence Relation (from [79])

A binary relation $R \subseteq X \times X$ is called an equivalence relation if and only if R is a tolerance relation and is

1. transitive, if xRy and yRz then xRz .

2.15 Rough Set Theory

In this dissertation, the rough set approach introduced by Zdzisław Pawlak [122, 126, 127] provides grounds for concluding to what degree a set of image segment pairs representing a standard covers a set of similar image segment pairs. The term “coverage” is used relative to the extent that a given set is covered by a standard set. An overview of rough set theory and applications is given in [79, 144].

Let U be a non-empty finite set (called a *universe*) and let $\mathcal{P}(U)$ denote the power set of U , *i.e.*, the family of all subsets of U . Elements of U may be, for example, images, image segments, physical objects, or observed behaviours of organisms. A *feature* \mathcal{F} of elements in U is measured by an associated probe function $f = f_{\mathcal{F}}$ whose range is denoted by \mathcal{V}_f , called the *value set* of f ; that is, $f : U \rightarrow \mathcal{V}_f$. There may be more than one probe function for each feature. For example, a feature of an object may be its weight, and different probe functions for weight are found by different weighing methods; or a feature might be colour, with probe functions measuring, *e.g.*, red, green, blue, hue, intensity, or saturation. The similarity or equivalence of objects can be investigated quantitatively by comparing a sufficient number of features by means of probes [118, 130]. In this dissertation, each feature is associated with only one probe function and its value set is taken from the set of real numbers \mathbb{R} . Thus one can identify the set of features with the set of associated probe functions, and hence we use f rather than $f_{\mathcal{F}}$ and call $\mathcal{V}_f = \mathcal{V}_{\mathcal{F}}$ a set of probe function values. If F is a finite set of probe functions representing features of elements in U , the pair (U, F) is called a *data table*, or *information system* (IS).

For each subset $B \subseteq F$ of probe functions, define the binary relation $\sim_B = \{(x, x') \in U \times U \mid \forall f \in B, f(x) = f(x')\}$ ⁹. Since each \sim_B is an equivalence relation, for $B \subseteq F$ and $x \in U$ let $[x]_B$ denote the equivalence class, or *block*, containing x , that is,

$$[x]_B = \{x' \in U \mid \forall f \in B, f(x') = f(x)\} \subseteq U.$$

If $(x, x') \in \sim_B$ (also written $x \sim_B x'$) then x and x' are said to be *indiscernible* with

⁹The symbol \mid denotes "such that", a notation commonly used in set theory [16].

respect to all feature probe functions in B , or simply, B -indiscernible.

Information about a sample $X \subseteq U$ can be approximated from information contained in B by constructing a B -lower approximation

$$B_*X = \bigcup_{x:[x]_B \subseteq X} [x]_B,$$

and a B -upper approximation

$$B^*X = \bigcup_{x:[x]_B \cap X \neq \emptyset} [x]_B.$$

The B -lower approximation B_*X is a collection of blocks of sample elements that can be classified with full certainty as members of X using the knowledge represented by features in B . By contrast, the B -upper approximation B^*X is a collection of blocks of sample elements representing both certain and possibly uncertain knowledge about X . Whenever $B_*X \subsetneq B^*X$, the sample X has been classified imperfectly, and is considered a rough set.

Example 1. Figure 12 shows a sample upper and lower approximation of a bounded region (call it \mathcal{I}) in the sample image. Let U consist of pixels in the image in Figure 12. Let f_i be probe functions that determine the horizontal and vertical locations of given point, *i.e.*, $f_1(x)$ returns the horizontal location of an element x and $f_2(x)$ returns the vertical location of an element x . Define δ -mesh function to be $f_{i,\delta}(x) = \lfloor f_i(x)/\delta \rfloor$. Assume that B contains two probe functions representing

the feature *resolution* that is represented by a so-called δ -mesh [7] function

$$\sim_{B,\delta} = \{(x, x') \in U^2 \mid \forall f_{i,\delta} \in B. f_{i,\delta}(x) = f_{i,\delta}(x')\}.$$

The relation $\sim_{B,\delta}$ partitions U into blocks containing adjacent pixels. All pixels within each square in the black grid are considered to be indiscernible because of the matching probe function values. Each of the darkened squares is entirely contained within \mathcal{I} to form a lower approximation of \mathcal{I} . On the other hand, the combination of squares containing pixels having a lighter-red or dark red colour either partially intersect with or are entirely contained inside \mathcal{I} and, hence, constitute an upper approximation of \mathcal{I} .

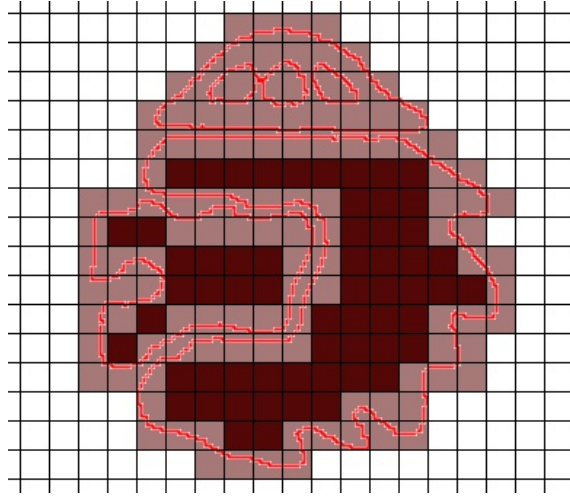


Figure 12: Sample representation of B_*X (dark squares), B^*X (light squares).

2.16 Approximation Spaces

This section gives a brief introduction to approximation spaces. A very detailed introduction to approximation spaces considered in the context of rough sets is presented in [144]. The classical definition of an approximation space given by Zdzisław Pawlak in [125, 122] is represented as a triple (U, F, \sim_B) , where the indiscernibility relation \sim_B is defined for probe functions in $B \subseteq F$ representing features of objects in a universe of objects U (see, *e.g.*, [161]). Let U/\sim_B denote a partition of U defined by \sim_B . It should be observed that any subset X of U can be approximated in U/\sim_B .

Before defining an approximation space a notion of an neighbourhood function and an overlap function are introduced.

The neighbourhood function N defines for every object x a set of similarly defined objects [162]. In effect, N defines a neighbourhood of every sample element x belonging to the universe U (see, *e.g.*, [132]).

The overlap function computes the degree of overlap between two subsets of U . Let $\mathcal{P}(U)$ denote the powerset of U .

A generalized approximation space was introduced by Skowron and Stepaniuk in [163, 164, 168]. A *generalized approximation space* is a system $GAS = (U, F, N, \nu)$ where

- U , a non-empty set of objects.
- F , a set of probe functions representing features of objects in U .
- $N : U \rightarrow \mathcal{P}(U)$, a neighbourhood function such that $x \in N(x)$ for any $x \in U$.

- $\nu : \mathcal{P}(U) \times \mathcal{P}(U) \rightarrow [0, 1]$ is an overlap function.

Specifically, any information system (U, F) and any $B \subseteq F$ naturally defines a parameterized approximation space $PAS_B = (U, F, N_B, \nu)$, where $N_B = [x]_B$, a B -indiscernibility class in a partition of U [130]. The overlap function ν computes the degree of overlap between two subsets of U . Standard rough inclusion (SRI) $\nu : \mathcal{P}(U) \times \mathcal{P}(U) \rightarrow [0, 1]$ is defined in terms of the relationship between two sets as

$$\nu_{SRI}(X, Y) = \begin{cases} \frac{|X \cap Y|}{|X|}, & \text{if } X \neq \emptyset, \\ 1, & \text{if } X = \emptyset. \end{cases}$$

for any $X, Y \subseteq U$. We are interested in Y because we want to see how well Y “covers” X , where Y represents a standard for evaluating pairs of image segments.

Standard rough coverage (SRC) ν_{SRC} is defined as

$$\nu_{SRC}(X, Y) = \begin{cases} \frac{|X \cap Y|}{|Y|}, & \text{if } Y \neq \emptyset, \\ 1, & \text{if } Y = \emptyset. \end{cases}$$

In other words, $\nu_{SRC}(X, Y)$ returns the degree that Y covers X . In the case where $X = Y$, then $\nu_{SRC}(X, Y) = 1$. The minimum coverage value $\nu_{SRC}(X, Y) = 0$ is obtained when $X \cap Y = \emptyset$ (*i.e.*, X and Y have no elements in common).

2.17 Genetic Algorithms

Evolution has been characterized as an optimization process [29, 66, 100]. Darwin [24] observed “organs of extreme perfection” that have evolved. Genetic al-

gorithms (GAs) belong to a class of evolutionary algorithms introduced by John Holland in 1975 [66] as a means of studying evolving populations. A GA has three basic features:

- **Representation:** each population member is coded to binary representation in a gene,
- **Method of Selection:** fitness of each population member is evaluated,
- **Method of Variation (Crossover):** create new population member by combining features from pairs of highly fit individuals.

Crossover is the fundamental operation used in classical genetic algorithms. Mutation is another method used in GAs to induce variations in the genes of a chromosome representing a population member. The basic steps in a genetic algorithm are described as follows. Let P_t denote an initial population of individual structures, each with an initial fitness at time t . Then an iteration begins. Individuals in P_t are selected for mating and copied to a mating buffer C_t at time step t . After that, the individuals in C_t are combined and form a new mating buffer C'_t . Next, a new population P_{t+1} is constructed from P_t and C'_t . A desired fitness is used as a stopping criterion for the iteration in a GA. A representation of a very basic GA that uses only the crossover operation is given in Algorithm 1.

In this dissertation, GA was used for search for matching segments in pairs of images. The details of implementation of GA is given in Section 3.1.8.

Algorithm 1: Basic GA

Input : population P_t , mating pool C_t
Output: evolved population P_T at time T
 $t = 0$;
Initialize fitness of members of P_t ;
while (*Termination condition not satisfied*) **do**
 $t = t + 1$;
 Construct mating pool C_t from P_{t-1} ;
 Crossover structures in C_t to construct new mating pool C'_t ;
 Evaluate fitness of individuals in C'_t ;
 Construct new population P_t from P_{t-1} and C'_t ;
end

2.18 Correlation

Search of the best match for two points by correlation is performed using two windows. One window I_o (original window) is centred at the point in original image for which one searches the match. The second window I_s (search window) is placed in different locations in the second image. The correlation value is calculated using (from [48])

$$C(x, y) = \sum_s \sum_t I_o(s, t) I_s(x + s, y + t). \quad (4)$$

A point (x, y) in the second image for which the value $C(x, y)$ is maximal is assumed to be the best match. The problem with such defined correlation is that it is influenced by the values in any of the considered windows regardless of the correlation between them. The solution is the *normalized cross correlation* [146]. In its general form it is given by

$$NCC(x, y) = \frac{\sum_s \sum_t I_o(s, t) I_s(x + s, y + t)}{\sqrt{\sum_s \sum_t [I_o(s, t)]^2 \sum_s \sum_t [I_o(x + s, x + t)]^2}}.$$

It is equal to the normalized correlation from 4. The effect of normalization is that the correlation increases only when the similarity of the two windows increases, not when the average level of pixels in one window increases.

Another similar measure of similarity of two windows I_o and I_s is the *correlation coefficient* [48]

$$\gamma(x, y) = \frac{\sum_s \sum_t [I_o(s, t) - \bar{I}_o(s, t)][I_s(x + s, y + t) - \bar{I}_s]}{\sqrt{[\sum_s \sum_t [I_o(s, t) - \bar{I}_o(s, t)]^2][\sum_s \sum_t [I_s(x + s, y + t) - \bar{I}_s]^2]},$$

where \bar{I}_o and \bar{I}_s denote averages over the entire window. This measure is also known in the literature as Zero-Mean Normalized Cross Correlation [74, 85].

3 2D Image Processing

This section concentrates on two dimensional digital image processing. This means that all processing described in this section makes no use of any 3D information. All information available at this step are pixel locations and their RGB values. The purpose of this step is to match all pixels from one image with corresponding pixels from the second image. This operation is known in the literature as *image registration* [35, 49, 201]. In this dissertation, the matching is performed in three steps.

First, a coarse matching is performed, see Section 3.1. The main idea behind coarse matching is to determine the approximate angle of rotation and scaling between both images. The subject of matching are not pixels but segments (segments were defined in Definition 9). The matching algorithm extracts segments from the images and detects an optimal angle of rotation and scale difference between them. At this stage no pixels are matched.

In the second step, namely the point matching step, the information from the coarse matching step is used. In preparation for pixel matching the images are rotated and rescaled. At this step not all pixels from the images are being matched. Instead, only several points from each image are selected and correspondences between them are found. This step is described in Section 3.2.

Finally, in the third step, namely the dense matching step, all points from both images are matched. This step uses the information about the relative location and orientation of both cameras in 3D space. This information allows to match pixels by means of epipolar geometry. The benefit of using the epipolar geometry is the

reduction of a search space by one dimension. The dense matching step is described in Section 3.3.

3.1 Coarse Matching

Table 1 presents all symbols used in this section.

Table 1: Symbols used in Coarse Matching section.

Symbol	Description
C	colour difference
O	overlap
A	angle of rotation
Rc	ratio of cardinalities
\mathbb{C}	domain of the colour differences
\mathbb{O}	domain of the overlap
\mathbb{A}	domain of the angle of rotation
\mathbb{RC}	domain of the ratio of cardinalities
Ω	subspace of \mathbb{R}^4 , $\Omega = \mathbb{C} \times \mathbb{O} \times \mathbb{A} \times \mathbb{RC}$
ξ	ideal solution for segment matching problem
Q_{n_i}	quantization algorithm, given in Algorithm 2 n_i - requested number of colours after quantization
Av_I	spatial colour averaging algorithm, given in Algorithm 3
$M_{3 \times 3}$	3 by 3 median filter
C_f	convexity factor
\overline{X}	mean value of expression X
Z	ideal solution for segment matching used in rough set approach
Ch	a chromosome (in Genetic Algorithm)
G	a set of genes
C_m	a codebook (in Lloyd quantization)
C_i, C_j	i -th or j -th segment colour
S_i, S_j	i -th or j -th segment
γ	separation of segment pairs resulting from GA
S	a set of the best genes returned by GA

Considerable work on the application of rough set methods in image processing

has been reported (see, *e.g.*, [7, 65, 138, 173, 213]). This dissertation introduces an approach to matching image segments in the context of approximation spaces (see Section 2.16). The basic model for an approximation space was introduced by Pawlak in 1981 [125], elaborated in [115, 122], generalized in [163, 164, 168], and applied in a number of ways (see, *e.g.*, [47, 133, 139, 140, 165]). An approximation space serves as a formal counterpart of perception or observation [115], and provides a framework for approximate reasoning about vague concepts. Image segmentation (see, *e.g.*, [17, 40, 54, 57, 84, 116, 158, 196, 198]), and the image segment matching problem (see, *e.g.*, [48, 148, 194, 197]) have been widely studied. The goal of an image-matching system is to match the segments from the two given images. Colour and overlap are the two features of image segments that are commonly used to solve the matching problem. To achieve more accuracy in matching image segments, a combination of an evolutionary approach to finding sets of similar segments and approximation spaces are used. The evolutionary approach is realized with a genetic algorithm (GA) that separates collections of image segments into sets of similar image segments. Filtering out GA-produced sets of image segments with the best match is carried out in the context of an approximation space. This approach makes it possible to solve the image segment matching problem with larger sets of features that yield more information about segments. This approach also results in more accurate matching of image segments. An overview of the 2D image segment matching method presented in this dissertation is shown in Figure 13.

The matching process begins by forming a composite of a pair of images, then carrying out colour quantization (step 2 in Figure 13). After that, the quantized

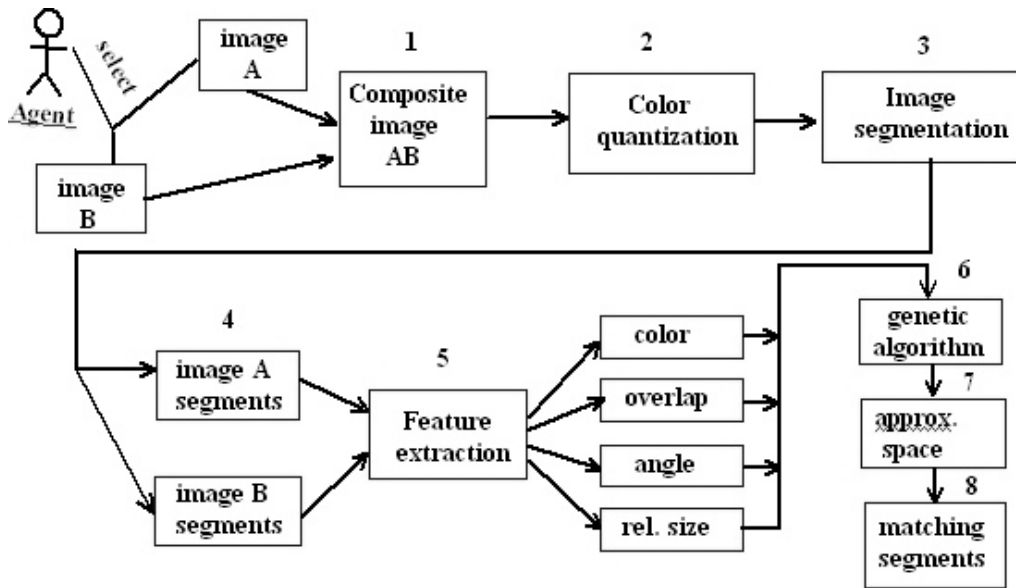


Figure 13: 2D Image Segment Matching Steps

image is segmented and this results in a pair of segmented images. Next, feature values of image segment pairs are obtained in step 5 in Figure 13. Then a GA is applied to a collection of image segment pairs that are separated into sets. After eliminating non-disjoint sets of segment pairs, the coverage of the remaining sets of segment pairs is measured relative to a standard (norm) that is a set of image segment pairs that represent certain knowledge. The end result in step 8 of Figure 13 is a collection of best matching pairs of image segments. This is in keeping with the original view of approximation spaces as counterparts of perception (in this case, approximations provide a framework for visual perception).

The process of matching segments described in this dissertation is based on four parameters described in Section 3.1.4. These parameters are *degree of overlap* between segments, *angle of rotation* between segments, *distance between mean colours*

of segments and *ratio of cardinalities* of both segments. A combination of a genetic algorithm and rough set-based post processing is used to combine the information from all four parameters to find the best matches between the segments.

The problem of finding the match between image segments is not trivial. The four parameters required for matching image segments sometimes contain contradictory information about the quality of match. Thus, it is impossible to find proper matches using only one or two of these parameters. The simplest approach is to find the matches with the smallest distance in the space defined by the four parameters. This space is denoted by Ω and consists of vectors where each coordinate value is the difference of particular parameter values.

Definition 14. Image Segment Parameter Space. Define the space Ω to be a subspace of \mathbb{R}^4 such that

$$\Omega = \mathbb{C} \times \mathbb{O} \times \mathbb{A} \times \mathbb{RC} \subseteq \mathbb{R}^4,$$

where \mathbb{C} , \mathbb{O} , \mathbb{A} and \mathbb{RC} denote domains of the four parameters' values, i.e., the distance between mean colours of segments, degree of overlap between segments, the angle of rotation between segments and the ratio of cardinalities of both segments, respectively.

The match between two points $s, t \in \Omega$ can be calculated as a weighted distance between their parameters' values.

Definition 15. Distance in Image Segment Parameter Space. The distance between two vectors s and t such that $s, t \in \Omega$ is defined to be a distance between

these points in the space Ω weighted by the vector $\omega = (C, O, A, Rc)$.

$$\|s - t\|_{\omega} = \sqrt{C \cdot (s_1 - t_1)^2 + O \cdot (s_2 - t_2)^2 + A \cdot (s_3 - t_3)^2 + Rc \cdot (s_4 - t_4)^2},$$

where $s = (s_1, s_2, s_3, s_4)$ and $t = (t_1, t_2, t_3, t_4)$, and $s_1, t_1 \in \mathbb{C}$, $s_2, t_2 \in \mathbb{O}$, $s_3, t_3 \in \mathbb{A}$ and $s_4, t_4 \in \mathbb{RC}$.

Here, the weight vector (C, O, A, Rc) denotes the importance of each parameter. Each such vector and an ideal vector ξ define a measure of the quality of a match.

Definition 16. Measure of Quality. A measure of quality parametrized by the vector $\omega = (C, O, A, Rc)$ of a match between a segment s and the ideal solution ξ is given by the distance between points s and ξ in Ω space.

$$Q_{\omega, \xi}(s) = \|s - \xi\|_{\omega}.$$

The problem that arises after formulating Definition 16 is caused by the definition of the ideal solution ξ . For the ideal solution ξ , it is possible to define the first two parameters without considering any particular matches. These parameters are the difference in colour $C = 0$ and the overlap between two segments $O = 1$. The remaining two parameters (A and Rc) can be defined only with respect to some set of matches. It does not make sense to define the ideal angle of rotation between segments, since this angle depends on the images and can be different for any pair of images. Therefore, the ideal solution cannot specify the rotation angle. By analogy, the ideal ratio of cardinalities cannot be defined either. In order to make the

remaining two parameters (rotation angle and ratio of cardinalities) not influential, the ω vector must contain zeros in the third and fourth position. As a result, the ideal vector is defined as

$$\xi = (0, 1, 0, 0), \quad \omega = (v, \nu, 0, 0), \quad (5)$$

where $v, \nu \in \mathbb{R}$. Unfortunately, this solution uses only two parameters instead of four. This can lead to wrong classification as shown in Figure 22 or Figure 25 (first row) (\circ denotes the correct match and $+$ denotes the closest match using $Q_{\omega, \xi}$ measure).

An algorithm that uses all four image segment features should generate a set of possible good matches. A genetic algorithm (GA) is the example of such an algorithm. It is possible to design a genetic algorithm (see, *e.g.*, Algorithm 6 and Algorithm 7) that orders image segments. This form of GA is considered an image segment matching algorithm that uses all four features in the image segment feature space.

The next few paragraphs contain detailed information about digital image processing methods that were implemented for coarse matching. These methods are helpful in capturing the information from images needed for pixel matching. The main idea here is to determine the approximate rotation angle and scaling between two images. In order to find these parameters more abstract information from the images can be extracted, like lines, solid areas of approximately the same colour (Section 3.1.1) or similar shapes (Section 3.1.3).

3.1.1 Image Quantization

Quantization has been defined as a process of converting an analog signal to a digital signal [46]. A quantizer is defined as a mapping from an uncountably infinite space of values into a finite set of *output levels*. In the proposed system the source signal is a digital image. Its domain is a finite set (pixels) of integer numbers (colours). Since colours are represented by three components, namely *Red*, *Green* and *Blue*, and each component is described by one byte, the input signal is already finite. Thus, the term quantization is rather used as a mapping from a finite set of numbers to another finite set of numbers, where the cardinality of the destination set is smaller than the source set. In what follows, the Lloyd quantization algorithm [46] has been used, see Algorithm 2.

Algorithm 2: The Lloyd Algorithm [46] (algorithm Q_n)

Input : image I , required number of colours n

Output: optimal codebook with n entries C_{opt}

Initialize codebook C_1 with n entries randomly, set $m = 1$

repeat

 Based on codebook C_m and using nearest neighbour condition, partition the image I into the quantization cells R_m

 Using centroid condition, find optimal codebook C_{m+1} for cells R_m

 Set $m = m + 1$

until *distortion caused by C_m is small enough*

Set $C_{opt} = C_m$

A quantization mapping is usually expressed by a codebook. A codebook is a set of n colours that are used to represent the original image. The mapping is performed by replacing the original colour with the closest colour from the codebook. The optimal codebook of size n is the set of colours that minimizes the distortion

caused by the codebook. Here, the distortion is calculated as the squared difference between all components of the original colour and its nearest neighbour from the codebook.

The Lloyd algorithm consists of main two steps that are repeated until the distortion caused by the codebook is small enough. The first step is the partitioning of the input image based on the current codebook. The partitioning is performed using nearest neighbour condition, *i.e.*, each pixel is assigned to the cell closest to the colour of given pixel. In the second step, a new codebook is created based on the partitioning from the first step. Each codebook entry is replaced by a centroid of all colours of pixels from the corresponding cell.

The main application for image quantization is signal compression. Since the number of colours after quantization is much smaller than the number of colours before, the same image can be represented using a smaller number of bits. In other words a quantization causes some loss of information from the image. It is important to note, that the quantization influences only the information carried by the pixels, not the location or number of the pixels itself.

The typical context for image quantization is deflation of the image size due to some memory constraints. The advantage of using quantization is small deterioration of image's visual content, but significant change in an image's size. The loss of information is an unwanted result or a price paid for the smaller image size.

In this research, the reason for a quantization is different. There are no constraints regarding memory limits nor computational complexity. There is also no interest in decreasing the size of the image either. Quantization is used as an aid

in image segmentation (see Section 3.1.2 for more details). There are two aspects of quantization that are important in this research. One of them is the removal of noise. Nearly every digital image contains some noise. The main cause of the noise is the process of quantization performed by the light sensor mounted in the light sensitive array in a digital camera. These sensors, due to imperfect construction and finite space of imaged colours, map the original colours into their closest digital representation introducing some error. As a result of this mapping, two neighbouring pixels that should have the same value, can have slightly different colours introducing the noise. Even though this noise is very small and not detectable by the human eye, when comparing two images it can have a huge effect on the result of matching. The problem lies in the magnitude of the noise and the magnitude of the signal. Real life images consist of millions of colours. In the captured image we can find almost any transitions between any two selected colours. Usually, these transitions are so smooth, that the magnitude of difference between two neighbouring pixels is comparable to the magnitude of the noise. As a result, it is not possible to detect the noise looking only at two pixels (it is possible when the context or neighbourhood in which the two pixels are considered is introduced). Now, the result of quantization is a smaller number of colours in the image. This means that some colours are merged into one. But, since most useful quantizers try to preserve visual content in the image, in the first place colours that are similar to each other are merged together. The resulting image contains less variations in colour and less noise (it can be argued here, that the image after quantization contains more noise, since it has less original information, but the point is that the noise, caused by imperfection

of light sensitive cells is reduced).

The second aspect of quantization considered in this research is the identification of regions of an image filled with a solid colour. In the steps following the quantization, regions in both images are identified and matched. In the images representing real world scenes the complexity of the scene makes it difficult to identify the same regions in both images. By reducing the number of colours the complexity of the scene is reduced, thus making it possible to form simple shapes.

In Figure 14 an original image of a street water pump is shown. The right part of the image shows the result of 2-bit Lloyd quantization. The quantized image contains only four colours. Not surprisingly, there are no large segments¹⁰. This is because quantization did not make any use of spatial information. In the next section it is shown how to incorporate spatial information into the process of segmentation.



Figure 14: Image of street water pump (left) and result of 2-bit quantization (right).

¹⁰One pixel surrounded by pixels of different colour is also considered a segment, but the purpose of segmentation is creation of much bigger segments.

3.1.2 Image Segmentation

Colour quantization described in the previous section is used as an aid in image segmentation. It works only in the colour space. The actual segmentation needs to take into account spatial information, namely the position of pixels. Only the combination of colour and spatial information leads to identification of image segments. The averaging step fills the gap regarding the use of spatial information. Its only purpose is to average information carried out by pixels representing similar colours. The term *similar* is in this context precisely defined. Assume, that an original image denoted by I_o is given. First, quantization reducing number of colours to n_1 is performed. This step is denoted by (6) and results in a quantized image denoted by $I_{q_{n_1}}$ (symbol Q represents the algorithm 2, where I_o is the input image I , and n_1 is the required number of colours).

$$I_o \xrightarrow{Q_{n_1}} I_{q_{n_1}}. \quad (6)$$

As a result of quantization, the quantized image $I_{q_{n_1}}$ contains only n_1 colours. In the next step, the information from the $I_{q_{n_1}}$ image is used to average the colours among all pixels that are connected, *i.e.*, belong to the same segment.

In a quantized image, regions of pixels of the same colour can be identified. These regions create segments. With each such segment is associated a colour that is an average of all original colours from pixels belonging to this region. This step is denoted below, see also Algorithm 3.

$$I_{q_{n_1}} \xrightarrow{Av_{I_o}} I_{Av_{n_1}}. \quad (7)$$

The image $I_{Av_{n_1}}$ resulting from the above equation has more than n_1 colours, where pixels are grouped into segments. This procedure, namely steps defined in (6) and (7), is repeated. The number of colours gradually decreases in consecutive iterations and the creation of segments can be observed.

Algorithm 3: The Spatial Colour Averaging (algorithm $Av_{I_{s_n}}$)

Input : image I

Output: averaged image I_A

Mark all pixels from I as *not processed*

foreach *not processed pixel* p **in** I **do**

Find segment $S(p)$ in I containing pixel p

Assign to each corresponding pixel in I_A from $S(p)$ an average colour of all pixels from $S(p)$

Set all pixels from $S(p)$ as *processed*

end

The unwanted effect of the algorithm defined this way is that if a segment is created at some step, there are no chances to change it in consecutive steps. In other words, the first quantization plays a crucial role in the entire process. In addition, the resulting image still contains a lot of details (even though the number of colours was reduced). An example of an image processed using seven iterations¹¹ described by the succession of mappings in (8), where numbers n_i for $i = 1, 2, \dots, 6$ are 256, 64, 32, 16, 12, 8 and 4, is shown in left side of Figure 15.

¹¹For $i = 0$ it is assumed that $I_{s_{n_0}} = I_o$, and after each iteration $I_{s_{n_{i-1}}} = I_{Av_{n_{i-1}}}$.

$$I_{s_{n_i-1}} \xrightarrow{Q_{n_i}} I_{q_{n_i}} \xrightarrow{Av_{I_{s_{n_i-1}}}} I_{Av_{n_i}}. \quad (8)$$

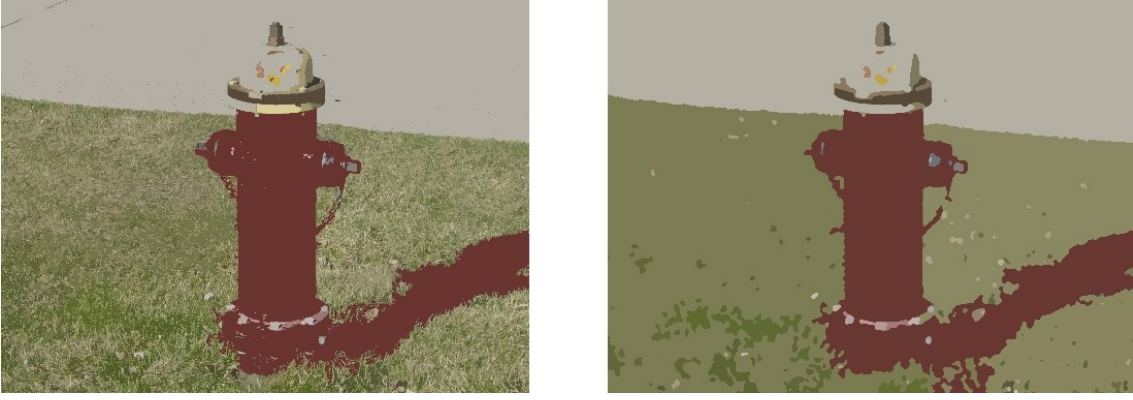


Figure 15: Hydrant image after 7 iterations of (8) (left) and (9) (right)

To make the entire segmentation process more robust and force the creation of bigger segments, one extra step for each stage defined by (8) is added. That is, after the colours are recreated from the original image, a 3 by 3 median filter is used. This causes almost uniform areas to blur even more and allows edges of neighbouring segments to overlap. As a result, all small details from the image are lost, and big uniform segments are formed instead. The final formula describing one step of this iterated algorithm is

$$I_{s_{n_i-1}} \xrightarrow{Q_{n_i}} I_{q_{n_i}} \xrightarrow{Av_{I_{s_{n_i-1}}}} I_{Av_{n_i}} \xrightarrow{M_{3 \times 3}} I_{s_{n_i}}. \quad (9)$$

The $M_{3 \times 3}$ symbol denotes the median filter that is applied to each pixel from an input image. The median filter is applied to 3 by 3 neighbourhood¹² of given pixel

¹²The 3 by 3 neighbourhood differs from the 8-neighbourhood by the fact that a point (x, y) belongs to the 3 by 3 neighbourhood and does not belong to 8-neighbourhood.

$p(x, y)$.

$$M_{3 \times 3}(p) = \text{median}\{p(x-1, y-1), p(x, y-1), p(x+1, y-1), p(x-1, y), \\ p(x, y), p(x+1, y), p(x-1, y+1), p(x, y+1), p(x+1, y+1)\}. \quad (10)$$

In order to find the median, all pixels are sorted by their colour value and the one in the middle (*i.e.*, at the 5-*th* place) is chosen.

The right side of Figure 15 shows the result of iterative algorithm (with the same values as in the previous example), where each step is described by (9). There are still many small segments, but comparing with the corresponding image, where the median filter was not used, their number is greatly reduced.

Figure 16 shows all steps of applying (9). The image in the first row and leftmost column is the original image. The second image in the first row, is a result of 8-bit quantization. The third image shows the result of applying 3 by 3 median filter. In the second row, the second iteration is shown. The leftmost image shows the result of averaging colours in segments from the previous step. The middle image shows the result of 7-bit quantization and the rightmost image shows the result of applying a 3 by 3 median filter. The remaining five rows are organized the same way as the second row, *i.e.*, the quantization codebook uses one less bit in each iteration.

3.1.3 Segment Selection

At this stage it is assumed that a digital image is divided into segments. To increase the chances of identifying the same segments in both images, image quantization is

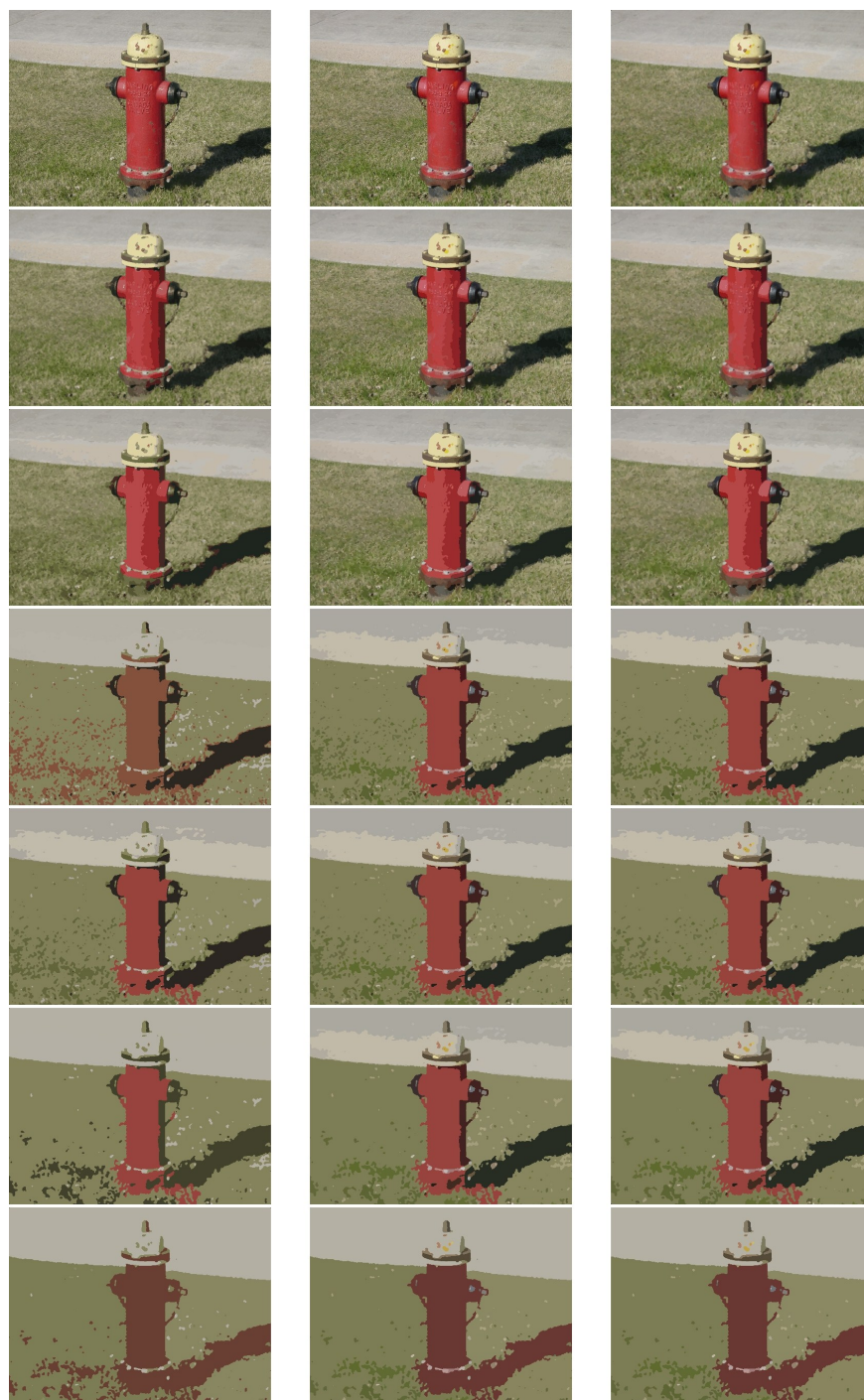


Figure 16: Quantized images obtained by iterating (9)

performed on one large image that is a composite of two individual images placed next to the other. After segmentation of the composite image, the two images in the composite are extracted and the analysis continues on the separate images. In this step, only some segments from all of the segments created so far are selected. The reason for this is the high number of segments and their shape. During a procedure to match shapes (described in Section 3.1.8), only segments from both images that satisfy specific criteria are matched using a GA search. In addition, the matching algorithm requires that each segment satisfies some additional properties. The criteria evaluated in this step are

- **Lower bound on segment size**
- **Upper bound on segment size**
- **Convexity factor**

Segment size is measured by the number of pixels belonging to a given segment. A lower bound on segment size is need for the following reasons. First, if there are not too many pixels, for example less than 10, the pixels can describe only a small number of distinctive shapes. Matching of such shapes is very difficult, since such a small number of pixels does not have enough power to uniquely represent a fairly distinctive shape. Second, if all tiny segments are considered, the search space for matching segments becomes too large. There is a small chance that these tiny shapes can be uniquely matched.

The explanation for upper bound of segment size is motivated by characteristics of most images. Usually, images contain large areas of a solid colour. For outdoor

images it can be the sky, for indoor images it can be the walls of the room the image is shot in. These solid areas function as a background for the given scene. The shape of the background is not unique and it changes due to perspective transformation. By setting an upper bound for segment size, all segments that can be part of the background are filtered out. For this research, this limit is set to be 30% of the entire image area.

The last constraint in matching image segments is the convexity factor that deals with the effect of perspective transformation and filters out shapes that are difficult to match. To get a deeper insight into this problem, consider what detected segments represent and how they differ from image to image. Each image is a 2D representation of a 3D scene. Similarly, segments that are flat, represent 3D objects. The transformation from 3D space into 2D images flattens objects in a sense that the information from different parts of an object is represented in a small area. For example, consider the silhouette of a tripod. Given one segment representing an entire tripod, each leg is separated from the other legs by some background pixels (at least at the bottom of the tripod). Depending on the angle of the camera, some legs can be quite close to each other. The shape of a tripod changes dramatically with the change of view angle. Attempts to match such shapes should be avoided. This example shows that objects that are spread in all three dimensions are separated by some pixels not belonging to the object. This condition is expressed for flat images in terms of convexity. A segment S is *convex* if each point from a straight line connecting any two points in S also belongs to S [48].

Definition 17. Convexity factor. The convexity factor for a segment S is a number $C_f(S)$ between 0 and 1 specifying how many lines between all combinations of points from segment S lie entirely inside the segment S .

$$C_f(S) = \frac{\# \text{ lines entirely inside } S}{\# \text{ all possible lines}}.$$

To filter out segments that are potentially difficult to match, a threshold for a convexity factor is set and only segments greater than the threshold are selected. Based on experiments, a threshold of 0.5 has worked well.

Implementation of an algorithm used to calculate a convexity factor from the definition requires $O(n^2)$ lines to be tested, where n is the number of segment pixels. In order to speed up the calculations the estimation is performed. The estimation process is applied on two levels. First, not all combinations of points are checked. Instead, randomly selected $50 \cdot n$ pairs of points are chosen. Second, instead of checking if an entire line is contained within a segment, only checks for 7 points are performed: middle point of a line, one fourth, three fourths and remaining multiples of $1/8$, namely, $1/8^{th}$, $3/8^{th}$, $5/8^{th}$ and $7/8^{th}$ of the line. In order to calculate each of these points only as few as two additions and two divisions are required, which makes this algorithm very fast with a complexity of $O(n)$.

3.1.4 Feature Generation

In this section, the following four features used for matching segments are elaborated on.

- **degree of overlap** between segments,
- **angle of rotation** between segments,
- **distance between mean colours** of segments,
- **ratio of cardinalities** of segment pairs.

This section describes how these features are extracted from two sets of segments (one set of segments for each image). Section 3.1.8 elaborates on how the actual matching is performed using these features. First, recall that segments are only two dimensional representations of 3D objects. Due to the change of view angle, segments undergo transformation that alters their shape. Therefore, simple comparison of shapes is not enough to pair segments.

Before the matching can start, values for the four features for all combinations of segments from both sets are generated. First two features are generated by an algorithm that tries to find the biggest overlap between two segments.

Overlap. This parameter measures the overlap between two segments. To calculate the overlap, two segments are plotted in one image using the same colour (one_seg denotes the number of pixels belonging to one segment). Pixels that belong to both segments are denoted by a second colour (two_seg denotes the number of pixels belonging to both segments). A measure of the overlap between a pair of image segments is computed using:

$$overlap = e^{-\frac{|P_{one_seg}|}{|P_{two_seg}|}}. \quad (11)$$

where P_i denotes pixels of i -th colour. For $|P_{one_seg}| \neq 0$ and $|P_{two_seg}| = 0$ it is assumed that $\frac{|P_{one_seg}|}{|P_{two_seg}|} = \infty$. In other words, *overlap* measures how well one segment matches the other. The minimum value for *overlap* is zero. In this case, the number of pixels belonging to both segments is equal to zero, which means that the segments do not intersect. A maximum *overlap* = 1 occurs when both segments have the same shape and are located at the same position. In such case $one_seg = 0$ and the overlap equals to $e^0 = 1$. For all other cases, $overlap \in (0, 1)$.

Formula (11) was chosen for two reasons. First, it rescales the range of overlap values from $(0, \infty)$ to interval $(0, 1)$. A finite interval is easier to handle than the infinite one. Second, the exponential function compresses the output of the original $\frac{|P_{one_seg}|}{|P_{two_seg}|}$ function in the range where P_{one_seg} is much greater than P_{two_seg} (for example, where $\frac{|P_{two_seg}|}{|P_{one_seg}|} < 2.5$, see Figure 17). The absolute value of the slope of the overlap function from 11 is much smaller than the slope of the $\frac{|P_{one_seg}|}{|P_{two_seg}|}$ function. This allows for easier comparison of overlap values in the last stage of overlapping, *i.e.*, when there is much more common pixels than not matched ones. The smaller slope means that small changes in the ratios of common/not matched pixels will not cause huge changes of the overlap function.

Figure 18 illustrates the best overlap. For better visualization, the two segments are plotted using different colours. The intersection is denoted by the brightest shade of grey. The left-hand side of Figure 18 shows both segments with their original rotation, scale and position. The right-hand side of Figure 18 shows the two segments with maximum *overlap* = 0.85. Observe that the area occupied by only one segment has been significantly decreased in comparison with the original

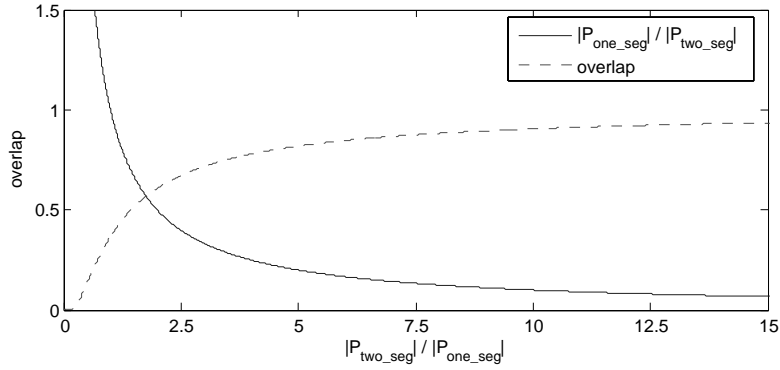


Figure 17: The result of applying exponential into overlap function

configuration.

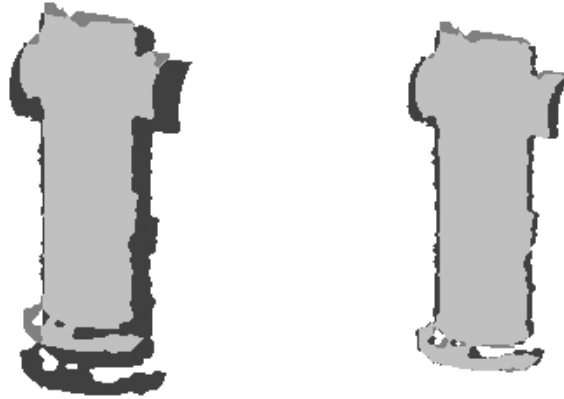


Figure 18: Preliminary overlap of 2 segments (left), and best overlap (right)

Example 2. Figure 19 shows three sample steps out of many steps performed during segment matching of Figure 15. These three steps explain the idea behind the overlap formula introduced in 11.

The first image in Figure 19 shows the first stage when the two segments do not have any pixels in common. The area of the first image is 41083 pixels and the area of the second one is 36447 pixels. Since there are no common pixels $|P_{one_seg}| =$

$41083 + 36447 = 77530$ and $|P_{two_seg}| = 0$. From the assumption for $|P_{one_seg}| \neq 0$ and $|P_{two_seg}| = 0$ the fraction $\frac{|P_{one_seg}|}{|P_{two_seg}|} = \infty$. Because of 11 the overlap is equal to $e^{-\infty} = 0$.

The second image shows one of the intermediate steps, where the two segments have a lot of pixels in common, but also a lot of non-overlapping pixels. The area of the first segment that does not intersect with the second one is 18219 pixels. The area of the second segment that does not intersect with the first one is 13583 pixels. The area of overlap between these segments is 22864 pixels. Therefore, $|P_{one_seg}| = 18219 + 13583 = 31802$ and $|P_{two_seg}| = 22864$. The overlap is equal to $overlap = e^{-\frac{|P_{one_seg}|}{|P_{two_seg}|}} = e^{-\frac{31802}{22864}} = e^{-1.391} = 0.248$.

The third image shows the final result of search for the best overlap. The first segment, was being translated, rotated and rescaled to maximize the overlap function. In this position the overlap is maximum. Here, the $|P_{one_seg}| = 5540 + 1097 = 6637$ and $|P_{two_seg}| = 35350$. Thus, $overlap = e^{-\frac{6637}{35350}} = e^{-0.187} = 0.828$.

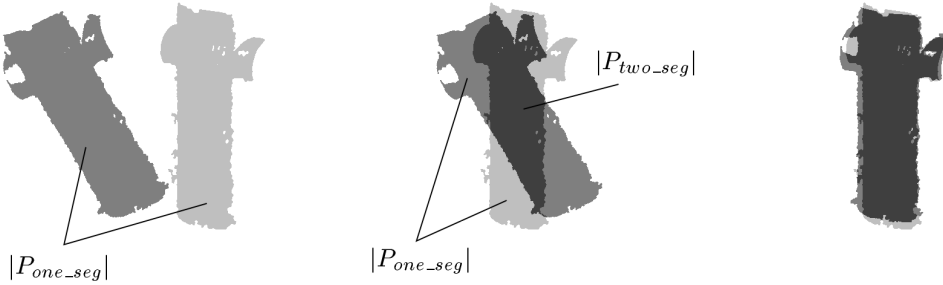


Figure 19: Three sample steps of segment matching.

Figure 20 shows the entire process of finding the best overlap for segments from Figure 15. The horizontal axis denotes the iteration number. In each iteration the

position, rotation and scale for the first segment is altered to minimize the overlap function. In the left part of Figure 20 a ratio $\frac{|P_{one_seg}|}{|P_{two_seg}|}$ is plotted. For the first several iterations it takes on high values compared to the end of the matching process. In fact, the ending of the matching process is more important, since it can detect small differences in segment shapes. Therefore, the overlap function, shown in the right part of Figure 20, is more sensitive to changes in the second half of the matching process. When two segments do not overlap significantly, the overlap function is close to zero. Only after there is a lot of overlap between segments (see the middle image from Figure 19), the overlap function changes more rapidly to emphasize the change in overlap.

Angle. The angle of rotation is the angle by which one segment must be rotated to maximize overlap (11) between two segments.

Colour. The previous two parameters (*overlap* and *angle*) dealt with geometrical properties of segments. The *colour* parameter takes into account the colour of a segment. Recall that all pixels from one segment are assigned the mean value of the colours from the original image. $C_{diff}(i, j)$ denotes the distance between the RGB vectors of colours for a pair of segments. If the i -th segment's colour is denoted by $C_i = (R_i, G_i, B_i)$ and j -th segment's colour is denoted by $C_j = (R_j, G_j, B_j)$, then $C_{diff}(i, j)$ is defined by:

$$C_{diff}(i, j) = |C_i - C_j| = \sqrt{(R_i - R_j)^2 + (G_i - G_j)^2 + (B_i - B_j)^2}.$$

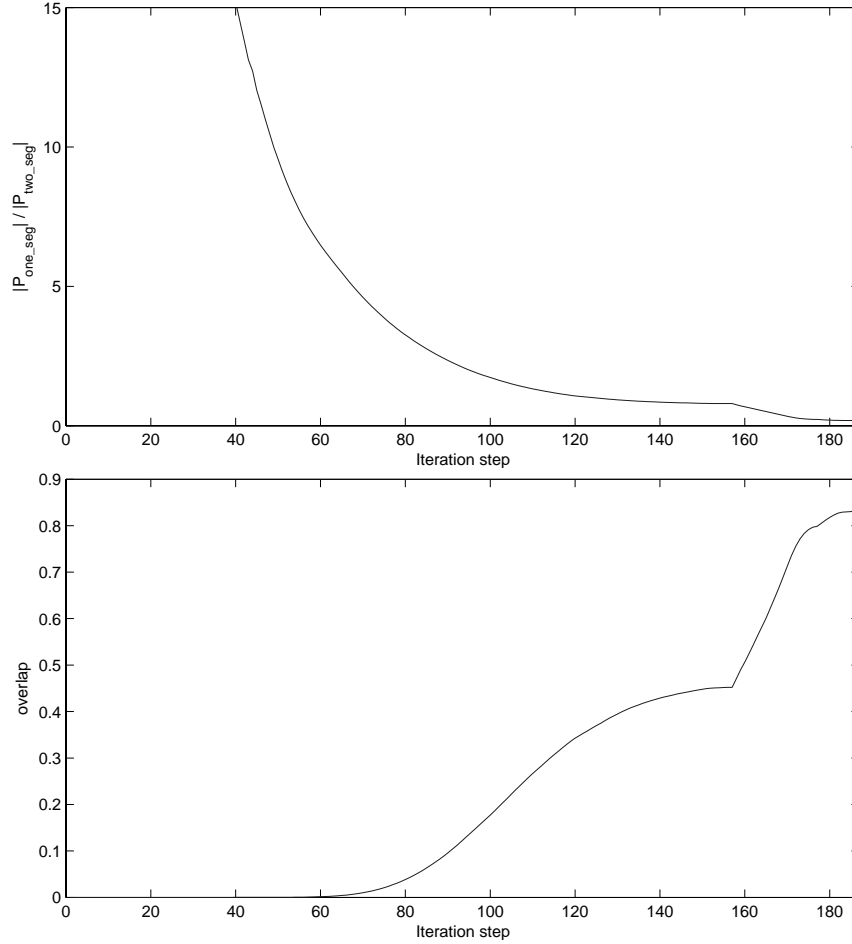


Figure 20: The process of finding the best overlap for $\frac{|P_{one_seg}|}{|P_{two_seg}|}$ and *overlap* parameter.

Ratio of Cardinalities. The *Ratio of Cardinalities* parameter is a measure of the relative size of a pair of segments. Let S_i, S_j denote sets of 4-neighbourhood connected pixels for image segment i and j , respectively. Further, let RC denote a measure of the *Ratio of Cardinalities*. It is defined by:

$$RC(i, j) = \frac{|S_i|}{|S_j|}. \quad (12)$$

A summary of the segment probe functions is given in the table below.

Table 2: Ranges of probe functions

colour	range $[0, 1]$. 0 means identical colours; 1, all channels differ by the maximal value, <i>i.e.</i> , the colour value is 1 if one segments has colour $(0, 0, 0)$ in RGB space and the second segment has colour $(255, 255, 255)$ in RGB space (where for each channel the range of values is from 0 to 255).
overlap	range $[0, 1]$. 0 means no overlap between segments; 1, identical segments (after translation, rotation and scaling).
angle of rotation	range $[-1, 1]$. 0 means no rotation between segments; 1, rotation by 180 degrees, where the sign denotes the direction of rotation.
RC	range $[0, \infty]$. 1 means that both segments have the same area. For $RC \in [0, 1]$ the second segment is greater than the first one. For $RC > 1$ the first segment is greater than the second one.

3.1.5 Exhaustive Feature Matching

The goal of a matching algorithm is to produce a set of segment pairs so that each segment belongs to a different image. Given n_1 segments identified in the first image and n_2 segments identified in the second image, the total number of possible pairs is $n_1 \cdot n_2$. From $n_1 \cdot n_2$ matches, only small number of matching segments corresponds to the correct matches.

A *hypothesis* is a set of image segment matches. A hypothesis is created by assuming that all four parameters for correctly matched segments are in the same range of values. A hypothesis is introduced to allow for grouping of several image segment matches.

Algorithm 4 searches through the space of matches using hypotheses to validate each pair. A hypothesis is characterized by the average rotation angle between segments and average ratio of cardinalities of both segments, where the average is taken with respect to all pairs in the hypothesis. The rotation angle between segments corresponds to the rotation between images and the ratio of cardinalities corresponds to the difference in distances between object and the camera for the two views. Thus, a hypothesis contains only pairs of segments, that are similar to each other with respect to these two conditions. If the difference in a segments' shapes is not caused by the change of viewpoint, then the difference comes from the fact that non-matching segments are being considered. In that case, the values for relative rotation and ratio of cardinalities are random for different pairs. On the other hand, if the difference in these parameters is caused by the change of the view point, it is the same for any two correctly paired segments. This allows for creation of bigger hypotheses with higher probability that each hypothesis contains only correct matches.

The four parameters are denoted by the following tables:

- $C(i, j)$: colour difference,
- $O(i, j)$: overlap,
- $A(i, j)$: angle of relative rotation,
- $RC(i, j)$: ratio of cardinalities.

where (i, j) denotes i -th segment from the first image and j -th segment from the second image, respectively. Next, a brief description of how these parameters are used to evaluate hypotheses, is given.

Ratio of cardinalities. This condition uses the ratio of cardinalities parameter $RC(i, j)$. If all the pairs from a given hypothesis are correct matches, then the value of this parameter for each pair should be in the same range. The minimal and maximal values of $RC(i, j)$ for all pairs from a given hypothesis are found. The minimum and maximum values should be in a $\pm RC_{th}$ range from the mean value of all ratios of cardinality for given hypothesis. If any $RC(i, j)$ value from given hypothesis is outside the interval $[(1 - RC_{th}) * \overline{RC}, (1 + RC_{th}) * \overline{RC}]$ ¹³ then a given hypothesis is not valid and is discarded. Otherwise, the next check is performed.

Angle of rotation. This check assumes that for correct matches the angles of rotation $A(i, j)$ should be similar to each other for all pairs from a given hypothesis. First, the average angle of all angles is calculated (except for the pair being added). Then the rotation angle from the pair being added, is compared to the average angle. If the absolute value of the difference is greater than some threshold A_{th} , then the given hypothesis fails the check and is removed from the system. Otherwise, the next check is performed.

Triangle property. After passing the *Ratio of cardinalities* and *Angle of rotation* checks, a newly added pair in a given hypothesis is checked against the triangle property. This property assures that a newly added pair preserves the order in which any three segments are arranged in a triangle. Given three segments in one image, one can connect the centroids of these segments creating a triangle. The vertices of this triangle can be ordered in clockwise or counter-clockwise order. After repeating the same procedure for corresponding segments in a second image, a second triangle

¹³The \overline{RC} symbol denotes the mean value of all RC parameter from a given hypothesis.

is formed. By checking the order of the vertices in the second triangle, one can validate the correctness of matches. If the order of vertices is not the same, this does *not* mean that the matching is not correct. The order is preserved between two different views if the triangle of interest is face up on the same side. But the centroids of segments need not lie on the plane in the real 3D space. This means that while moving from one view to the second one, the triangle formed by these segments is flipped to the other side, which reverses the order of the vertices. Nevertheless, this effect is very hard to obtain. Notice, that the identified segments would have to have identical shape from both sides. In most cases, the change in position between the two views is too small to cause this to happen. Hence, despite this special case, the power of discriminating bad matches is very useful for this application and is utilized in this check to decrease the number of hypotheses.

In Algorithm 4, the centroid of a segment from a new pair is used to build triangles with all combinations of centroids from the hypothesis. The corresponding triangles for segments from a second image are built as well. If the order of vertices for any of these corresponding triangles do not match, the hypothesis fails the check and is removed from the set M . Otherwise, the algorithm finishes the pruning part and moves to the growing step.

The last part of the matching Algorithm 4 identifies the hypothesis that is the most likely to contain only correct matches. After applying Algorithm 4, the set M consists of many hypotheses that satisfy all conditions. From them, only one hypothesis is selected using the measure of correctness. The measure of correctness is the number of hypotheses a pair is included in. To each pair a number of hy-

Algorithm 4: Matching Segments

Input : tables $C(i, j)$, $O(i, j)$, $A(i, j)$, $RC(i, j)$, centroids of all segments

Output: hypothesis with highest score, sets of matches M

Set the set of all hypotheses $M = \emptyset$, and $N_M = |M|$;

for all segments S_i in the first image **do**

 Create pairs $P_i = \bigcup_j P_{ij}$ with all segments S_j from second image;

 Remove from P_i all pairs P_{ij} such that

$C(i, j) > C_{th}$ or $O(i, j) < O_{th}$;

 Add $N_P = |P_i|$ pairs to N_M existing hypotheses

 producing total of $N_M + N_M \cdot N_P + N_P$ hypotheses;

foreach hypothesis $M_k \in M$ **do**

 Set $\overline{RC} = \sum_{i,j} \frac{RC(i,j)}{|M_k|}$

if $\exists_{RC(i,j)} RC(i, j) \notin [(1 - RC_{th}) \cdot \overline{RC}, (1 + RC_{th}) \cdot \overline{RC}]$ or

$|A(i_{new}, j_{new}) - \text{mean}_{P(i,j) \in M_k \setminus P(i_{new}, j_{new})} A(i, j)| > A_{th}$ or

$P(i_{new}, j_{new})$ changes triangle order **then**

 Remove M_k from M ;

end

end

end

potheses that this pair is included in is assigned. Then to each hypothesis a score is assigned that is the sum of all measures of correctness of all pairs belonging to given hypothesis. The hypothesis with the highest score is selected as the output of Algorithm 4. The list of pairs from the selected hypothesis consists of correctly paired segments from both images.

In the following sections, more sophisticated methods for selecting the optimal set of matches are presented.

3.1.6 Single Point Standard (Upper Approximation)

The goal of an image-matching system is to match segments from two given images. For example, Figure 21 shows generated segments for the Wearever®¹⁴ box scene. The left image in Figure 21 contains 68 segments and the right image contains 51 segments.

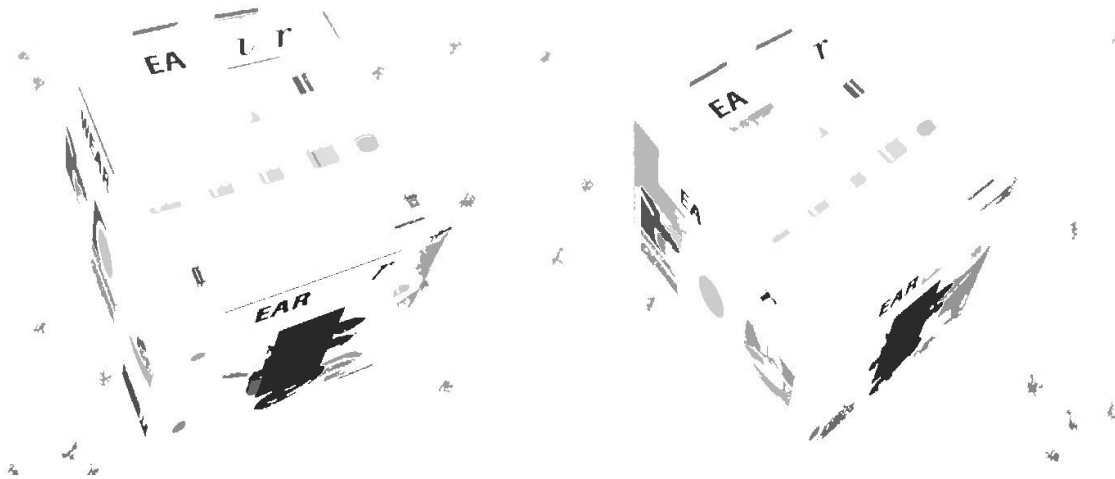


Figure 21: Generated segments for the Wearever box scene.

Let $IS = (U, F)$ be an information system, where U is a set of pairs of image segments, and F is a set of image segment probe functions. The probe functions in F , namely, *degree of overlap*, *angle of rotation*, *distance between mean colours* and *ratio of cardinalities*, are defined relative to two segments. Hence, each probe function value is indexed by two numbers that are the indices of the segments in a pair from U . Most ranges of values for the segment probe functions have been adjusted so that they are in the interval $[-1, 1]$ (see Table 2).

¹⁴Trademark of the WearEver Company, <http://www.wearever.com>

The rough matching is performed relative to the upper approximation of the standard set \mathcal{Z} that is an ideal match of two image segments (see the discussion of an ideal vector in (5)). In other words, \mathcal{Z} is the optimal case for matching two identical image segments and such case may, but does not have to exist in the real data.

The angle of rotation and ratio of cardinalities for a proper match are unknown. Hence, use of these probe functions does not introduce any new information and is not considered in what follows, since the standard for these probe functions is unknown.

All segment pairs are ranked based on the information represented by $B^*\mathcal{Z}$. Different upper approximations can be constructed by changing the equivalence relation and subsets of probe functions used to obtain $B^*\mathcal{Z}$. In the original K-means clustering algorithm [114], data points are arranged so that they are clustered around K centres. In this work, an equivalence relation based on the K-means clustering algorithm has been introduced (see, *e.g.*, [138]), and is summarized in this section. Briefly, two segments S_i and S_j are in relation $\sim_{B,K}$ if and only if the values of all probe functions for S_i and S_j are associated with the same cluster. $\sim_{B,K}$ is formally defined in

$$\sim_{B,K} = \left\{ \begin{array}{l} (S_i, S_j) \in U^2 \mid \forall f \in B, \\ \exists l. 1 \leq l \leq K, f(S_i) \in C_l \wedge f(S_j) \in C_l \end{array} \right\},$$

where C_l denotes the l -th cluster from the set of K clusters. Let the set \mathcal{Z} be defined as:

$$\mathcal{Z} = \{x \in U \times U \mid \text{colour}(x) = 0, \text{overlap}(x) = 1\}. \quad (13)$$

The set \mathcal{Z} consists of matched pairs of segments with probe function values specified in (13). Let $[x]_B$ be a class in the partition of U that is a set of B -indiscernible pairs of image segments containing x . At this point, there is interest in finding the upper approximation of \mathcal{Z} that is described by

$$B^*(\mathcal{Z}) = \{x \mid [x]_B \cap \mathcal{Z} \neq \emptyset\}.$$

Algorithm 5: Matching Segments Using Upper Approximation

Input : set of probe functions $\mathcal{A} = \{C(i, j), O(i, j)\}$
Output: ranking of all segment pairs s_{ij}

```

for (all segments  $S_i$  in the first image) do
  for ( $K=2$  to ( $\#$  of segments in the second image)/2) do
    Perform K-means clustering for each probe function separately
    for (each subset  $B$  of the set of all probe functions  $F$ ) do
      Find  $B^*(\mathcal{Z})$ 
      for (each segment  $S_j$  from the second image) do
        if ( $S_j \in B^*(\mathcal{Z})$ ) then
          | vote for pair  $S_{ij}$ 
        end
      end
    end
  end
end

```

Algorithm 5 gives the steps for ranking segment matches using the upper approximation. To each vote is assigned the same unit weight. Because cases where

two probe functions are used include cases where one probe function is used, the effective weights are greater for cases with multiple probe functions used. Table 3 shows the effective voting weights for the Algorithm 5.

Table 3: Voting table for Algorithm 5

# of probe functions in B	# of votes	effective # of votes
1	1	1
2	1	3

Figures 22 and 23 show sample voting results for two segments. A circle \circ denotes a good match made by visual inspection of the two images, and a cross $+$ denotes the segment that is the closest to the standard \mathcal{Z} . That is, for a given segment i from the first image, a $+$ denotes the segment j_{min} from the second image such that

$$j_{min} = \underset{j}{argmin} |\mathcal{Z} - \{C(i, j), O(i, j)\}|,$$

i.e., the argument j (segment number) for which the value of the expression $|\mathcal{Z} - \{C(i, j), O(i, j)\}|$ is minimum. Figure 22 shows how the information is extracted from the generated probe functions using the upper approximation $B^*(\mathcal{Z})$ of the set \mathcal{Z} . The cross $+$ shows that the best match using the distance between the given three parameters is with segment number 44. However, the correct match is with segment number 18. The number of votes for segment number 18 is higher than the number of votes for segment 44. This means that using this algorithm, segment 18 is more likely to be chosen as the match than segment 44.

The problem that is still to be solved is the high number of segments with high

votes. For example, in Figure 22, segments 18, 20 and 33 have high votes and it is not possible to select the best match. Hence, there is interest in considering the lower approximation $B_*(\mathcal{Z})$ of the set \mathcal{Z} .

3.1.7 Interval Standard (Lower Approximation)

This section presents an extension of the method described in Section 3.1.6. The lower approximation $B_*(\mathcal{Z})$ is derived relative to \mathcal{Z} that is defined as the approximation of a set of image segment pairs that constitute a perfect match. Let δ denote a parameter used to adjust the interval to define the ideal match \mathcal{Z} . The goal is to find δ such that an image segment pair constitutes a match. Hence, an interval interpretation of the probe function values of image segment pairs is introduced. That is, the probe function values associated with image segment pairs in the set \mathcal{Z} are parametrized by a parameter $\delta > 0$. In effect, each probe function value of each image segment pair from \mathcal{Z} belongs to a small interval containing δ . Using this approach, \mathcal{Z} is defined as:

$$\mathcal{Z} = \{x \in U \times U \mid colour(x) \in (0, \delta), overlap(x) \in (1 - \delta, 1)\}, \quad (14)$$

where the probe function values for each image segment pair x in \mathcal{Z} belong to intervals for colour and overlap specified in (14).

For experiments, the parameter δ was set to 0.1. The results for different δ values did not differ significantly from the ones shown here. The formula for calculating the lower approximation is given by

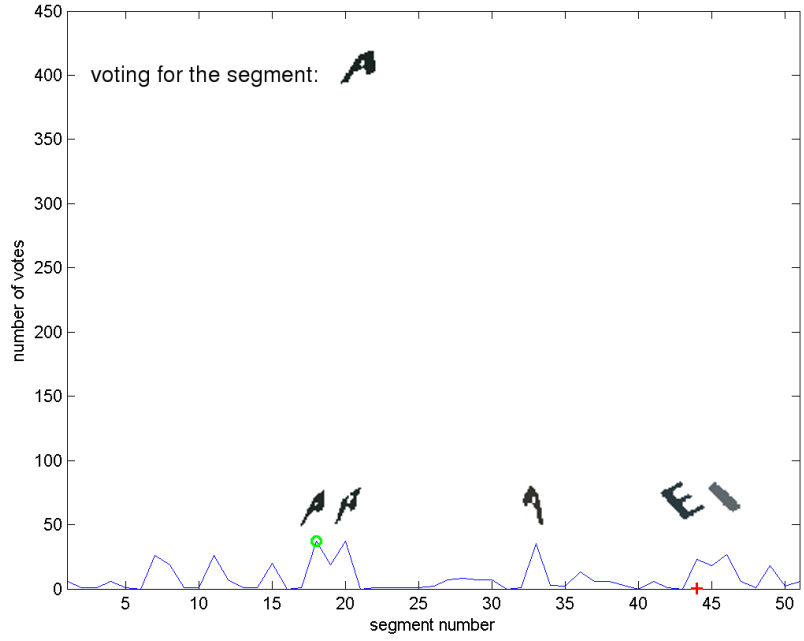


Figure 22: Voting results. o good match, + the closest match.

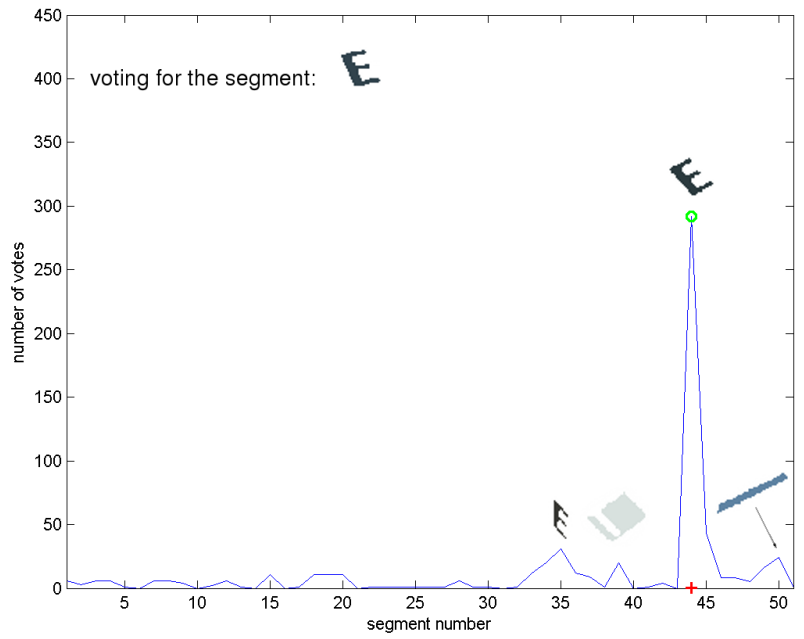


Figure 23: Voting results: o good match, + the closest match.

$$B_*(\mathcal{Z}) = \{x \mid [x]_B \subseteq \mathcal{Z}\}. \quad (15)$$

The new matching algorithm is essentially the same as Algorithm 5, except that a *Find* $B_*(\mathcal{Z})$ operation has been added.

As can be seen from Figure 25, the best results are obtained for the ‘single point standard’. The circle \circ in Figure 25 denotes a good match made by visual inspection of the two images, and a cross $+$ indicates a segment pair that is the closest to \mathcal{Z} . The ‘interval standard’ method fails to yield one segment pair as a good match. Instead, it yields several segments with equally high votes. This means that this method cannot be used by itself as the deciding method for solving the matching problem. However, the interval standard method can be used as an aid, since the correct solution is usually among the segment pairs with the highest votes.

The Figure 24 and the next example is used to explain in more detail the idea of the standard \mathcal{Z} .



Figure 24: Segments for Example 3

Example 3. Left part of Figure 24 shows the first image. It consists of twelve segments created from the letters of a word ”Matching”. Notice, there are only

eight letters in the word "Matching". The remaining four segments are: white area in the letter 'a' (denoted by a.), a dot in 'i' letter (denoted by i·), upper white area in the letter 'g' (denoted by g·) and lower white area in the letter 'g' (denoted by g.).

The right part of the image 24 shows the same letters as the first image. Only, for the second image, they underwent geometrical transformations: image warping, rescaling and rotation. In addition, the brightness of each letter from the second image was randomly altered.

In order to construct the standard \mathcal{Z} , the colour difference and overlap parameters were calculated. The colour difference values are shown in Table 4 and overlap values are shown in Table 5. Values corresponding to proper matches are denoted by bold face font in both tables. For example, the pair of segments 'M' from both images is characterized by the pair (0.070,0.614), where the first number denotes the colour difference and the second number denotes the overlap value for these two segments.

Table 4: Colour table

	M	a	t	c	h	i	i·	n	g	a.	g·	g.
M	0.070	0.574	0.671	0.066	0.572	0.532	0.532	0.639	0.068	0.784	0.784	0.784
a	0.579	0.064	0.707	0.549	0.606	0.492	0.492	0.213	0.647	0.778	0.778	0.778
t	0.654	0.693	0.048	0.595	0.091	0.875	0.875	0.528	0.657	0.773	0.773	0.773
c	0.118	0.539	0.628	0.011	0.532	0.545	0.545	0.596	0.126	0.792	0.792	0.792
h	0.571	0.608	0.179	0.531	0.045	0.753	0.753	0.449	0.585	0.647	0.647	0.647
i	0.481	0.452	0.898	0.564	0.760	0.069	0.069	0.554	0.558	0.483	0.483	0.483
i·	0.482	0.454	0.894	0.565	0.756	0.075	0.075	0.552	0.559	0.477	0.477	0.477
n	0.649	0.305	0.556	0.636	0.449	0.551	0.551	0.108	0.710	0.579	0.579	0.579
g	0.082	0.617	0.679	0.171	0.573	0.507	0.507	0.666	0.078	0.705	0.705	0.705
a.	0.719	0.727	0.800	0.790	0.679	0.542	0.542	0.657	0.768	0.014	0.014	0.014
g·	0.715	0.726	0.799	0.786	0.677	0.540	0.540	0.657	0.764	0.017	0.017	0.017
g.	0.718	0.728	0.800	0.790	0.679	0.542	0.542	0.659	0.767	0.014	0.014	0.014

The creation of standard \mathcal{Z} for given parameter δ is straight forward. From (14), the standard is a set of all segment pairs for which the colour difference and the overlap values are in some interval, *i.e.*, colour difference is less than δ and overlap is

Table 5: Overlap table

	M	a	t	c	h	i	i'	n	g	a.	g'	g.
M	0.614	0.084	0.019	0.058	0.372	0.040	0.001	0.242	0.210	0.001	0.001	0.002
a	0.367	0.383	0.143	0.206	0.221	0.165	0.010	0.497	0.276	0.003	0.005	0.010
t	0.141	0.330	0.832	0.292	0.249	0.001	0.001	0.151	0.153	0.016	0.142	0.203
c	0.181	0.266	0.202	0.722	0.228	0.250	0.043	0.316	0.220	0.069	0.081	0.121
h	0.052	0.411	0.331	0.139	0.714	0.247	0.002	0.549	0.251	0.008	0.004	0.021
i	0.301	0.275	0.001	0.306	0.134	0.716	0.133	0.230	0.166	0.267	0.508	0.423
i'	0.003	0.256	0.001	0.113	0.032	0.216	0.913	0.033	0.018	0.421	0.675	0.807
n	0.333	0.390	0.116	0.368	0.239	0.149	0.012	0.847	0.280	0.001	0.002	0.024
g	0.234	0.268	0.118	0.109	0.155	0.115	0.001	0.091	0.713	0.001	0.001	0.010
a.	0.001	0.045	0.320	0.102	0.028	0.185	0.726	0.001	0.007	0.861	0.818	0.721
g'	0.001	0.060	0.152	0.054	0.015	0.274	0.730	0.003	0.002	0.675	0.931	0.873
g.	0.001	0.108	0.251	0.130	0.042	0.425	0.618	0.337	0.039	0.726	0.865	0.860

greater than $1 - \delta$. For example, for $\delta = 0.1$ there are only two segments satisfying the above requirements. These pairs are (i', i') for which $\text{colour}(i', i') = 0.075 < 0.1$, $\text{overlap}(i', i') = 0.913 > 0.9$, and (g', g') for which $\text{colour}(g', g') = 0.017 < 0.1$, $\text{overlap}(g', g') = 0.931 > 0.9$, see Tables 4 and 5. Therefore, for $\delta = 0.1$ the standard $\mathcal{Z} = \{(i', i'), (g', g')\}$. This means that the segments i' and g' are the most similar segments in both images. The matching of the remaining segments is performed relative to this match.

Table 6: \mathcal{Z} table vs. δ parameter

δ	\mathcal{Z}	(colour, overlap)
0.05	\emptyset	
0.1	$\{(i', i'), (g', g')\}$	$\{(0.075, 0.913), (0.017, 0.931)\}$
0.15	$\{(i', i'), (a., a.), (g', g'), (g., g'), (g', g.), (g., g.)\}$	$\{(0.075, 0.913), (0.014, 0.861), (0.017, 0.931), (0.014, 0.865), (0.017, 0.873), (0.014, 0.860)\}$
0.2	$\{(t, t), (i', i'), (n, n), (a., a.), (a., g'), (g', g'), (g., g'), (g', g.), (g., g.)\}$	$\{(0.048, 0.832), (0.075, 0.913), (0.108, 0.847), (0.014, 0.861), (0.014, 0.818), (0.017, 0.931), (0.014, 0.865), (0.017, 0.873), (0.014, 0.860)\}$
0.3	$\{(t, t), (c, c), (h, h), (i, i), (i', i'), (n, n), (g, g), (a., a.), (g., a.), (a., g'), (g', g'), (g., g'), (a., g.), (g', g.), (g., g.)\}$	$\{(0.048, 0.832), (0.011, 0.722), (0.045, 0.714), (0.069, 0.716), (0.075, 0.913), (0.108, 0.847), (0.078, 0.713), (0.014, 0.861), (0.014, 0.726), (0.014, 0.818), (0.017, 0.931), (0.014, 0.865), (0.014, 0.721), (0.017, 0.873), (0.014, 0.860)\}$

Table 6 shows several sets \mathcal{Z} for different values of parameter δ . The bigger the parameter δ the more matches are included in the standard \mathcal{Z} . This is the

crucial property of the rough set approach, where the definition of an 'ideal match' is not fixed, but can be adjusted based on available information and data. Notice, for smaller δ values the standard \mathcal{Z} is an empty set, which means that the identical segments are not present in both images. On the other hand, bigger δ values produce a standard that contains incorrect matches. Nevertheless, at this stage, the matching correctness is not crucial, in fact, segments $(g.,g\cdot)$ form better match than $(g.,g.)$, since (colour, overlap) values for the first pair are (0.014, 0.865) and for the second pair (0.014, 0.860). The correctness of this match cannot be determined using only colour difference and overlap values, but other parameters must be used as well.

As shown in this example, the δ parameter allows for adjusting how strict the definition of an ideal match is. This was not possible in the Upper Approximation based approach (see (13)).

3.1.8 Genetic Approach for Matching

A genetic approach for segment matching creates a framework for the search based on any set of features extracted from images. Feature values are extracted using different shapes, where a shape is a segment, line or a point in an image. A genetic approach is used, because one of the byproducts of the genetic algorithm is a separation of the space of all possible matches (induced by chromosomes). And this separation is used to match segments in context of approximation spaces.

The central notion of a classical, Darwinian form of genetic algorithm is a gene. A *gene* is a pair of matched feature values for two images. There are many types of features that can be used in the algorithm. More abstract forms (also called *shapes*)

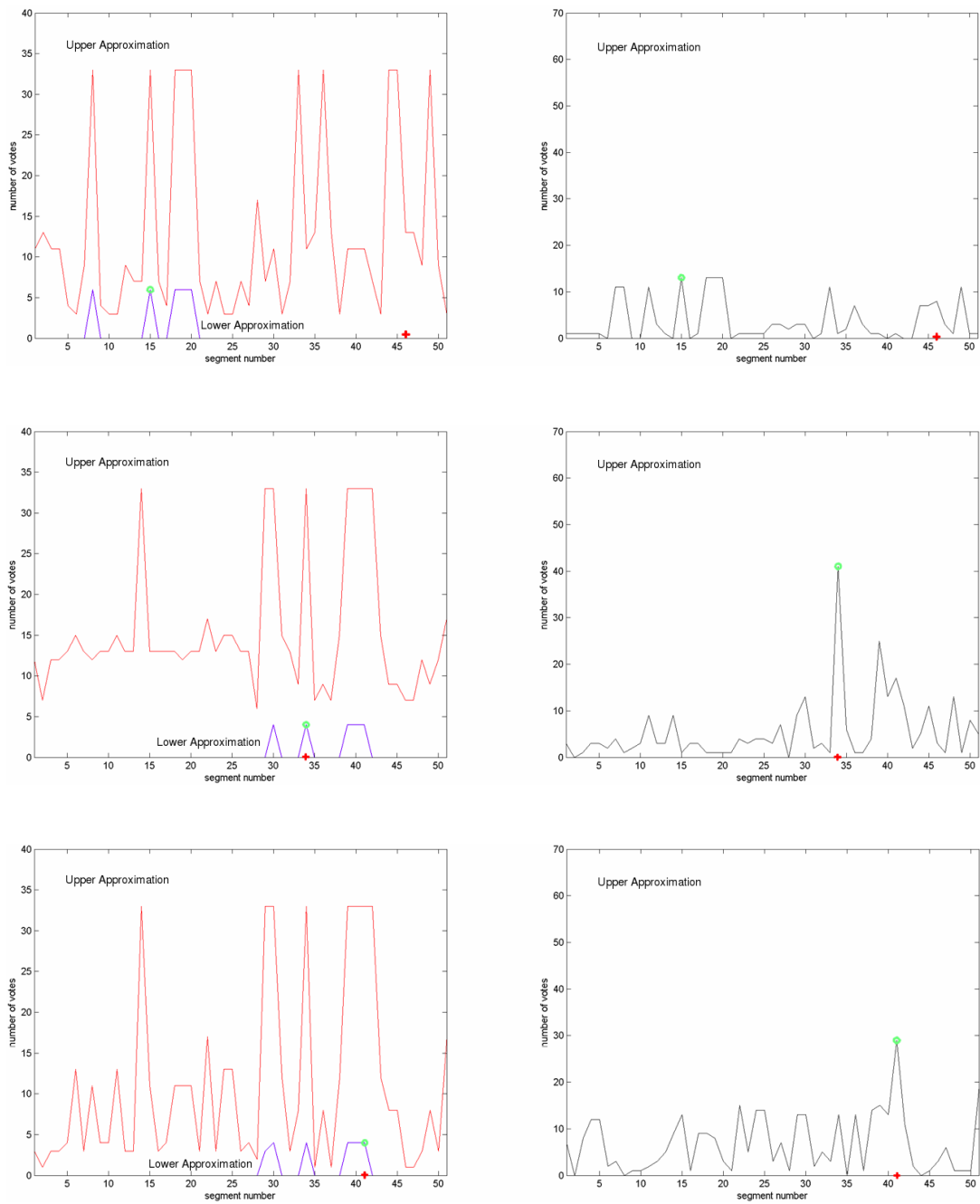


Figure 25: Voting results: ‘interval standard’ (left) and ‘single point standard’ (right).

are sources of features, *e.g.*, a segment has its colour, a line has its length, etc. Three methods of deriving features have been considered so far, and are summarized in Table 7. The simplest form of a gene considered in this research is a pair of segments, where each segment comes from a different image. One of the functions of a gene in genetic approach is to hide the differences between the features for the genetic algorithm. The genetic algorithm does not discern between the genes and processes them the same way.

A *chromosome* is a collection of genes. The set of features creates a chromosome (or *hypothesis* as described in Section 3.1.5). A genetic algorithm tries to select the best chromosome that consists only of correct matches. The term *hypothesis* is used in the context of matching features from the images, since a hypothesis identifies possible matches of features. The term chromosome is used in the context of the genetic algorithm, since a chromosome is a member of the population.

Table 7: Probe functions and their corresponding abstract forms.

Abstract Form	Complexity	Probe Functions
point	simple	cross-correlation
line	moderate	cross-correlation, angle, RC
segment	complex	colour, overlap, angle, RC

The overview of the structures used in Algorithm 6 is given in Figure 26. Three kinds of feature generators are denoted by three paths at the bottom of the image. The tasks of image processing blocks are generation of points, lines and segments. After this step, the identified shapes are passed to the feature extraction blocks. These blocks use selected shapes to generate features. The term ‘feature’ needs

more explanation. Usually, a *feature* is represented by a probe function that maps an observed object in the universe to a value. In this case, a feature is not represented by a probe function defined relative to some aspect of a single image segment or pixel, but, instead, is defined relative to a comparison of a pair of image segments or pixels. In the case of the colour of a pair of image segments, a feature is represented by the difference in the average colours of two segments. In general, a feature probe function value $\mathcal{F}(x, y)$ for a pair of observed objects x and y is a scalar from some pre-defined range $[a, b]$, *i.e.*, $\mathcal{F}(x, y) \in \mathbb{R}$ and $a \leq \mathcal{F}(x, y) \leq b$. In addition, there is one point $\kappa \in [a, b]$ that denotes a value for which objects x and y are not discernible with respect to the given feature. For example, for colours of image segment pairs, the possible range of colour differences can be defined as $[0, 1]$, where 0 denotes two identical colours and 1 denotes the maximum difference of colours allowed by an image's colour depth. In this case, $\kappa = 0$.

The fact that the features are calculated as a difference between probe function values for pairs of digital images is denoted by the 'fusion' box in Figure 26. The procedure represented by the fusion box combines the information from pairs of images to generate feature values.

Next, the description of a chromosome is given (see Figure 27). A gene represents a match between two shapes from a pair of images. This is indicated by a pair of indices of two corresponding shapes. Genes in each chromosome are divided into three blocks: point block, line block and segment block. Each block contains indices of matched shapes of a given type, namely point, line or segment.

The number of genes in each group can be zero. The genetic algorithm does not

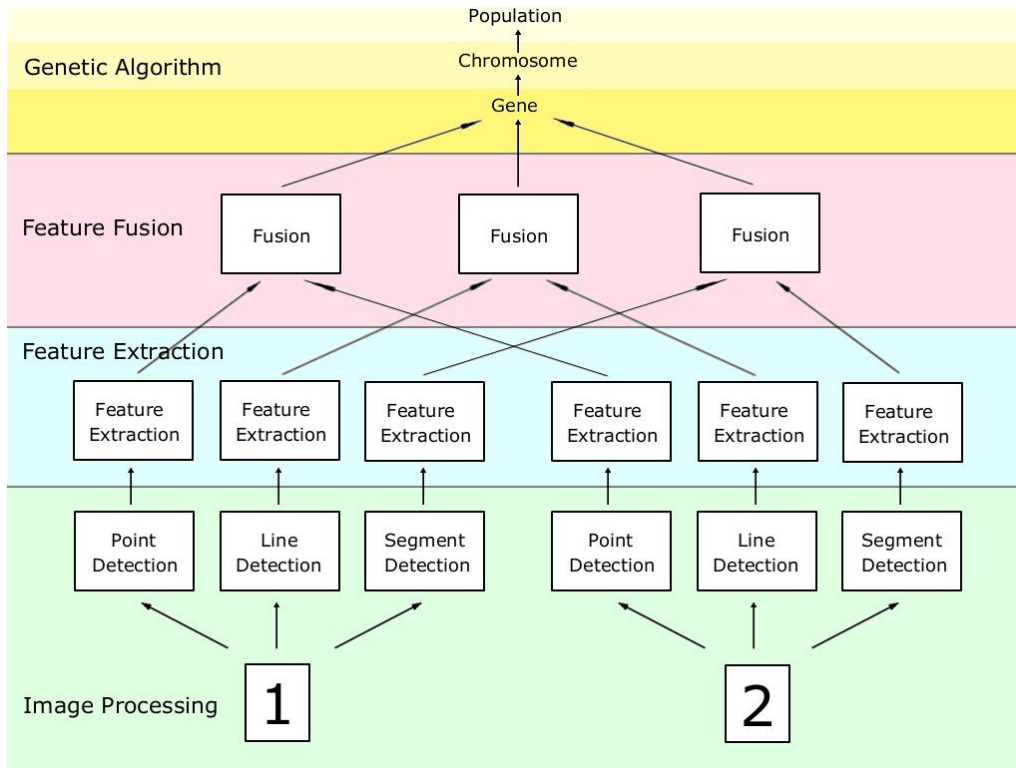


Figure 26: The overview of the genetic algorithm.

discern between different types of genes as long as both halves of the gene are of the same type. The order of the chromosome is the sum of lengths of all blocks, *i.e.*, $np + nl + ns$ (number of points + number of lines + number of segments).

The current version of the genetic algorithm has only image segment genes implemented, since the previous steps generate a number of segments from both images. The experiments show that segments provide enough information to determine the scale and rotation between the images. Therefore, to speed up the calculations point and line blocks are not used and are always empty. If this system was applied to a class of images requiring more features, lines and points can be easily added. The

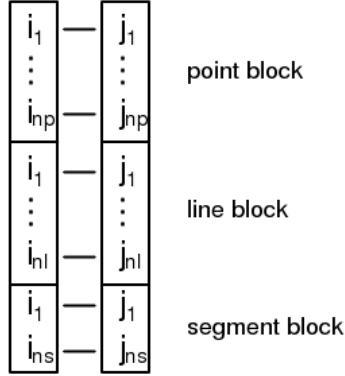


Figure 27: The chromosome. Each block contains indices of matched shapes.

creation of the genes is constrained by the rules, which assures that only reasonable matches are considered. These rules control the colour difference, overlap and the ratio of cardinalities. Let $colour_th$, $overlap_th$, rc_th denote the maximum values allowed for the colour difference, overlap and the ratio of cardinalities for a pair of image segments, respectively. For the ratio of cardinalities parameter, the threshold denotes the the maximum difference of the areas of a pair of segments, *e.g.*, for $rc_th = 2$ it means that $\frac{1}{2} \leq RC \leq 2$. Table 8 gives constraints for feature values during the creation of genes.

Table 8: Rules for creating genes.

Feature	Condition
colour	$\leq colour_th$
Overlap	$\geq overlap_th$
RC	$\geq 1/rc_th \wedge \leq rc_th$

Let Ch denote a chromosome from a population evaluated by a genetic algorithm. Further, let ang , \overline{ang} , rc_th , ang_th denote angle of rotation, average angle of rotation, ratio of cardinalities threshold, and angle threshold, respectively. The

$RC(i_k, j_k)$ parameter was defined in (12) and the \overline{RC} symbol denotes the mean value of ratios of cardinalities for all genes from given chromosome. The subscript k iterates from 1 to the number of genes in given chromosome such that subscripts (i_k, j_k) iterate through all image segments contained in the chromosome, see Figure 27. The fitness of Ch is determined using

$$Fitness(Ch) = \begin{cases} 1 & \text{if } \forall k \ (1 - RC_{th}) \cdot \overline{RC} \leq RC(i_k, j_k) \leq (1 + RC_{th}) \cdot \overline{RC} \\ & \wedge \max_k |ang(i_k, j_k) - \overline{ang}| < ang_{th} \\ & \wedge \text{all genes within chromosome pass the triangle check,} \\ 0 & \text{otherwise.} \end{cases}$$

The fitness function shown above is maximally selective, *i.e.*, it causes chromosomes to survive and reproduce or die and be removed from the population.

Algorithm 6 is based on the standard procedure for genetic algorithms described in [27]. The only genetic operator implemented is the crossover operation. New genes are not created by Algorithm 6. All unique genes appearing in the population are created before evolutionary iteration starts. The crossover operation cannot split halves of existing genes. Two chromosomes of lengths n_1 and n_2 can only be concatenated to form a new chromosome of a length $n_1 + n_2$ that contains all genes from the original two chromosomes. The repetition of left and right handed parts of the gene within a chromosome is not allowed either. This means that only two chromosomes with different sets of left and right handed parts can mate and create an offspring.

After implementing Algorithm 6, all genes are scored using the Algorithm 7. The

Algorithm 6: Genetic Algorithm

Input : tables $C(i, j)$, $O(i, j)$, $A(i, j)$, $RC(i, j)$, centroids of all segments

Output: ordered set of matches \mathcal{O}

Create the set of genes S using rules from table 8

while (*stop condition is not true*) **do**

 | Apply genetic operator: crossover

 | Evaluate fitness function

 | Remove chromosomes with fitness function below some threshold

end

Order all genes into set \mathcal{O} using *GA based ordering* algorithm, see Algorithm 7

symbol γ denotes separation introduced by the chromosomes. Each chromosome forms a block of genes. Genes within a block (a chromosome) are considered to be indiscernible. Since, different chromosomes can contain the same genes this separation forms a tolerance relation (see Definition 12).

The genes are sorted by the number of chromosomes they appear in. The chromosome that contains the most common genes is selected as the output of the simulation. All sorted genes are returned in the set \mathcal{O} .

Algorithm 7: GA based ordering

Input : set of genes G , separation of this set γ

Output: ordered set of matches \mathcal{O}

Create the set of counters for all genes in the set G

Set initial values of these counters to 0

for (*all blocks from γ*) **do**

 | **for** (*all genes S_i from given block*) **do**

 | Increase counter of gene S_i by 1

 | **end**

end

Return sorted in descending order list of genes $\mathcal{O} = \text{sort}(G)$

3.1.9 2D Matching with Approximation Spaces

This section considers an approach to processing the output from the genetic algorithm 6 within the context of an approximation space. Let S be a set of n best genes returned by Algorithm 6. Let $B(x)$ be a set of genes equivalent to x , and let B_*S be the lower approximation of the set S . There is an advantage in using B_*S as a standard, and measuring how well B_*S “covers” each set. This “normalized” view of S (*i.e.*, S considered in relation to classes in U / \sim_B that are proper subsets of S) makes it possible to select a set of genes covered to the greatest extent by the standard. The steps of this approach to finding the set of best matches are given in Algorithm 8.

Algorithm 8: The algorithm for selecting best matches using rough coverage

Input : set of all possible matches

Output: ordered set of matches

Run the GA to get the separation of the genes

Create and initialize to 0 the rough coverage weights for each gene

Create a set S of top n genes

for (*each set* $[x_i]_B$) **do**

 Evaluate rough inclusion value

$$Rcover([x_i]_B, S) = \frac{|[x_i]_B \cap B_*S|}{|B_*S|}$$

 Increase weights of genes from $[x_i]_B$ by $Rcover([x_i]_B, S)$

end

The results are shown in Figure 28. In the plot from Figure 28 a set S was created from 114 genes. The most important result to observe in this plot is that the rough coverage measure does better than other methods for the number of genes

between 37 and 49. This means Algorithm 8 sorts the genes better than the pure GA represented in Algorithm 6. This also means that the rough coverage measure can be used to find more correct matches than other methods.

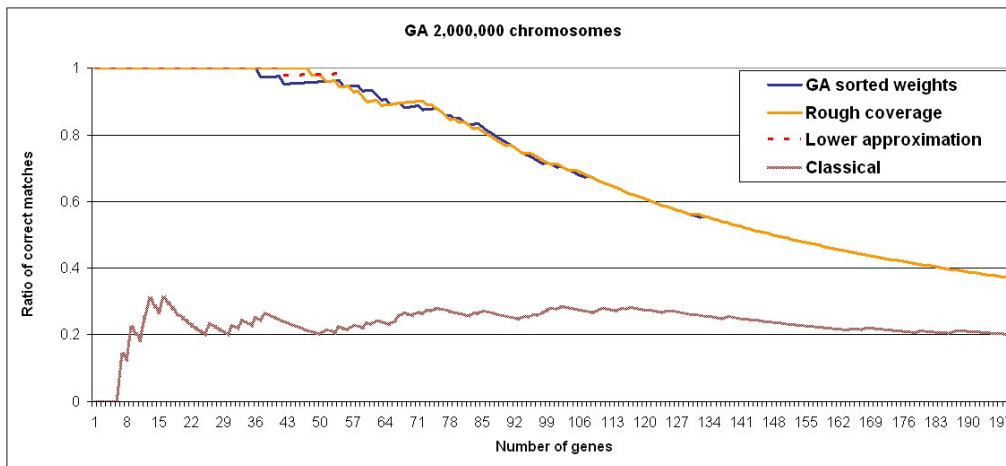


Figure 28: Rough coverage vs. ratio of correct matches for 2,000,000 chromosomes.

3.1.10 Tolerance Relation vs. Equivalence Relation

The output of the GA in Algorithm 6 is a set of hypotheses. In other words, Algorithm 6 produces sets containing pairs of image segments. Each such set (hypothesis) corresponds to one chromosome. The crossover operation in Algorithm 6 produces a chromosome that is a copy of two input chromosomes. Therefore, the resulting separation is a tolerance relation (see Property 1). Algorithm 9 converts the tolerance relation induced by Algorithm 6 into an equivalence relation. This is done by removing the overlapping sets in the separation created by Algorithm 6.

Observe that Algorithm 9 searches for the chromosomes with the highest weight (starting from the longest chromosomes) and removes all chromosomes that have

Algorithm 9: Conversion of tolerance relation to equivalence relation

Input : set of genes G , set of chromosomes Ch , separation γ of set G
expressed by sets $Ch_I \in Ch$

Output: partitioning \sim_B of set G

```
1 Order all genes using GA based ordering, see Algorithm 7
2 Set  $\sim_B$  to  $\emptyset$ 
   /* starting from the longest chromosomes                                     */
4 for (all chromosomes' lengths I) do
5   for (all chromosomes  $Ch_I$  of the length I) do
6     Find the chromosome  $ch_{max} \in Ch_I$  with the highest score and move it
       to  $\sim_B$ 
7     for (all chromosomes  $ch_k$  in  $Ch$ ) do
8       if ( $ch_k \cap ch_{max} \neq \emptyset$ ) then
10        | Remove  $ch_k$  from  $Ch$ 
11        | end
12      end
13    end
14 end
```

non-empty intersection with a given chromosome. The resulting partitioning of all genes C forms an equivalence relation (see Theorem 2).

The down side of converting the tolerance relation induced by Algorithm 6 into an equivalence relation is the reduction of the number of chromosomes. For example, in the case of the system consisting of $\approx 818,000$ chromosomes with 4920 genes where the longest chromosome has length 29, the conversion to equivalence relation decreases the number of chromosomes with cardinality greater than one to 1229. This means that the number of blocks after the conversion is less than 0.16% of the number of tolerance relation separation sets.

Experimental results show that the equivalence relation does not have enough

power to improve the ordering produced by the GA in Algorithm 6. This is due to the small number of blocks in the equivalence relation. Figure 29 shows sample results for 818,000 chromosomes.

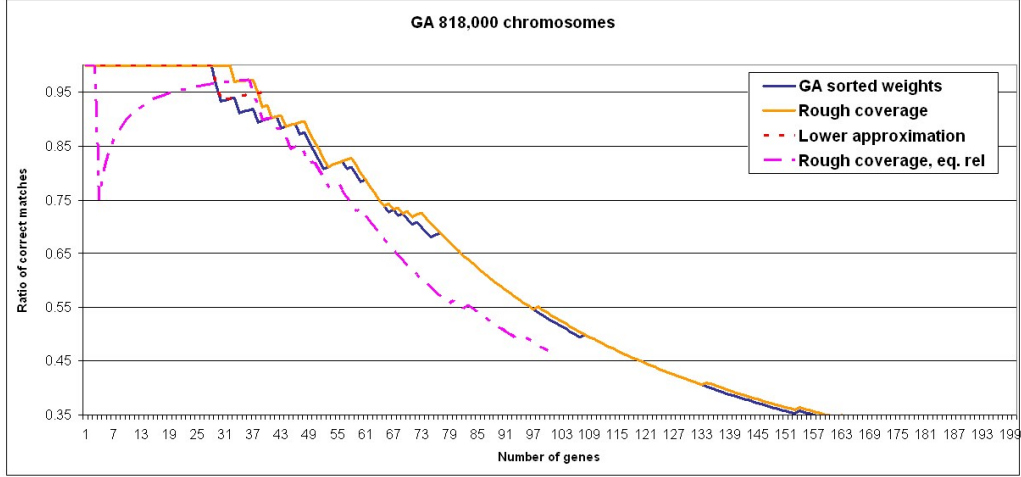


Figure 29: Ratio of correct matches for tolerance and equivalence relation

Next, several properties relating the proposed evolutionary approach with rough set theory are given.

Property 1. The crossover operator in the GA in Algorithm 6 produces a partitioning of the set of genes G .

Proof. Let Ch denote the set of all chromosomes ch_k such that $Ch = \bigcup_k ch_k$. If L denotes the longest chromosome in Ch , then all chromosomes can be grouped into subsets of Ch , namely, $Ch_L, Ch_{L-1}, \dots, Ch_I, \dots, Ch_1$, where $Ch_I \subseteq Ch$ and index I denotes the length of the chromosome¹⁵. Without the loss of generality, consider a

¹⁵The index written with a small letter by a chromosome ch does not indicate the length of the chromosome.

case where a chromosome of a length I is crossed with a chromosome of a length J producing a chromosome of a length $K = I + J$:

$$\text{crossover} : Ch_I \times Ch_J \rightarrow Ch_K, \quad ch_k = \text{crossover}(ch_i, ch_j),$$

where $ch_i \in Ch_I$, $ch_j \in Ch_J$, $ch_k \in Ch_K$, $K = I + J$, $K \geq 2$ and $I, J \geq 1$.

After applying the crossover operator, the chromosomes ch_i and ch_j are not removed from Ch . This means that a gene g_t that belongs originally to ch_i belongs also to the chromosome ch_k .

Now, for any chromosome ch_k of order greater than 1,

$$\forall_{g_t \in ch_k} \exists l \neq k \mid g_t \in ch_l.$$

From the fact that the order of a chromosome ch_k is greater than 1, we have $ch_k = \text{crossover}(ch_i, ch_j)$. What follows is that $l = i$ or $l = j$. \square

The above proof shows that for each chromosome ch_k of order $K > 1$ there exists a chromosome of order smaller than K that is a part of ch_k . Therefore, for each such chromosome ch_k there exist at least two chromosomes that have non-empty intersection with it.

Theorem 1. The GA in Algorithm 6 produces a separation of the set of all genes, which corresponds to a tolerance relation γ .

Proof. Chromosomes ch_k produced by the GA consist of the genes from the set G . Each chromosome can be considered as a separation set of indistinguishable

genes. Thus, they create a separation of the set G . This separation corresponds to the tolerance relation if the relation determined by this separation is reflexive and symmetric. The relation γ is based on the fact that two elements belong to the same chromosome, i.e.

$$x\gamma y \text{ iff } \exists_i \mid x \in ch_i \text{ and } y \in ch_i$$

where $ch_i \in Ch$ and $x, y \in G$. From the definition this relation is symmetric and reflexive.

The separation Ch covers all genes G because Ch includes the set $Ch_1 = G$ of chromosomes of order one, so $Ch_1 \in Ch$ therefore $G \subseteq Ch$.

The relation γ is not an equivalence relation because chromosomes ch may have non-empty intersections (from Property 1). □

Theorem 2. Algorithm 9 converts the separation γ produced by the GA in Algorithm 6 into a partitioning \sim_B .

Proof. Since Algorithm 9 does not create new separation sets, the partition induced by \sim_B is a subset of the separation induced by γ . Thus, there are two conditions that must be satisfied for the partitioning \sim_B to be an equivalence relation:

- all blocks from \sim_B must have empty intersection:

Step 10 from Algorithm 9 assures that all subsets from \sim_B have empty intersection.

- all blocks from \sim_B must cover the entire space of genes G :

The step 4 from Algorithm 9 starts with the longest chromosomes and ends

with the shortest $Ch_L, Ch_{L-1}, \dots, Ch_I, \dots, Ch_1$, where L is the length of the longest chromosome in the system. The shortest chromosome is of length one, *i.e.*, it is a gene. Notice, none of the genes that are not included in the set \sim_B will be removed from the set Ch_1 because their intersection with ch_{max} is empty. This means that in the last iteration of loop 4 all missing genes will be added to the set \sim_B .

□

3.1.11 Classical vs. Rough Matching Methods

Classical 2D image segment matching method is usually limited to two probe functions, namely, colour difference and the overlap between two segments (see, *e.g.*, [48, 94, 197]). This is a severe limitation because these two probe functions do not yield enough information to permit accurate image segment matching. By contrast, in designing genes in chromosomes used in evolutionary 2D segment matching, the number of features and corresponding probe functions associated with a gene can be quite large.

In this study, four parameters for each gene and 2,000,000 chromosomes have been used. Similarly, using the rough coverage methods, the number of probe functions associated with an image segment can be large. The probe functions used in this study are defined in terms of the distance between mean colours of segments, the degree of overlap between segments, the angle of rotation between segments and the ratio of cardinalities of both segments. In addition, rough coverage values represent a comparison between a set representing a norm (*e.g.*, B_*S) and each of the

separation sets containing similar pairs of image segments. This means that information contained in separation sets generated by chromosomes is validated against the information contained in the standard. The higher the rough coverage function the bigger overlap between these two sets and the greater chance that genes belonging to a given separation set are correct matches. In effect, the rough coverage matching method yields better results because it uses more information about the image segments being compared. This is one way to explain the plots in Figure 28.

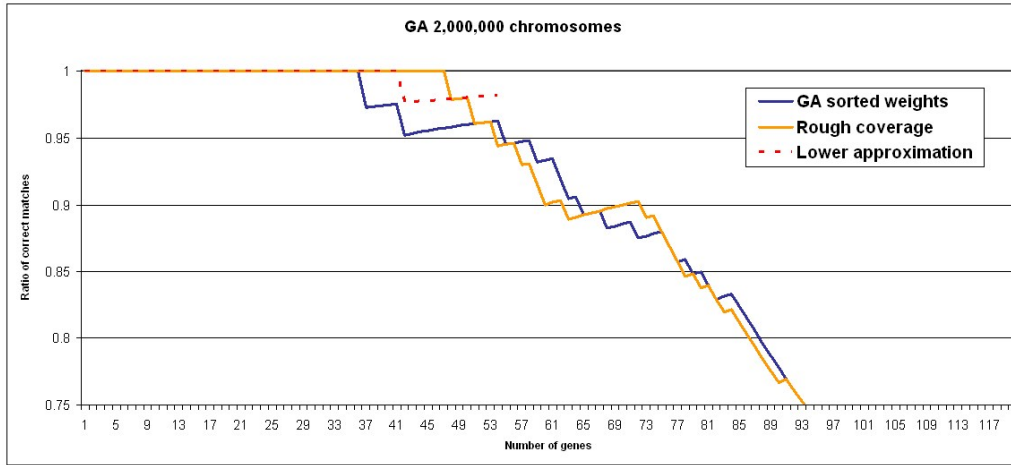


Figure 30: Rough coverage vs. ratio of correct matches (zoomed Figure 28).

In Figure 30, the left upper corner of the plot from Figure 28 is shown. Figure 30 illustrates the advantage of the rough coverage approach compared to the other methods. Recall, that the problem of segment matching is considered in the context of 2D to 3D conversion. The 2D to 3D conversion algorithm takes as an input paired pixels that are generated from paired segments. Any mismatch at the segment matching stage propagates to the pixel matching stage and finally into 2D to 3D conversion. Therefore, a crucial requirement for image segment matching is

to generate as little wrong matches as possible. Figure 30 shows that the rough coverage approach yields the biggest number of correct matches, *i.e.*, the first wrong match occurs after finding 47 good matches. For the weights generated by the GA used in Algorithm 8, the first mismatch occurs after only 36 correct matches. Hence, rough coverage approach reduces the number of mismatches, which improves the robustness of the overall 2D to 3D conversion process.

3.2 Point Matching

In this section an approach to point matching¹⁶ is presented (see [49] and [201] for excellent reviews of this topic). The goal of point matching is to find pairs of matched points from both images. There is no requirement to match all possible points from both images. Instead, it is enough to match only some small number of points. These matches are used for 2D to 3D conversion (see Section 4). Thus, the more points matched the better, but the crucial property of a match is the quality not the quantity.

In the process of point matching, there are three separate problems that need to be considered. The selection of an interest point is the first problem to consider (see Section 3.2.1). The purpose of selecting interest points is to limit the search space and consider only points that are somehow characteristic for an image. Next, calculation of local descriptors must be considered (see Section 3.2.5). The purpose of this step is to extract local information around a detected point and pre-process it so that the image is more suitable for matching. Usually, this entails extracting information about a given point, which is invariant relative to required image transformations. Finally, the most similar points are found using the extracted features. The main goal is to find the best distance measure to match given interest points. A high-level view of this three-step point-matching process is depicted in Figure 31¹⁷.

This subsection contains descriptions of several of the most popular point matching techniques. In general, there are no assumptions about any extra knowledge of

¹⁶A word "point" is used interchangeably with "pixel"

¹⁷Note, that in this dissertation local descriptors are not used, see Section 3.2.6.



Figure 31: Overview of Point Matching steps

approximate locations of prospective matches. Usually, no extra information about the rotation and scaling between given images is given either. The only information needed in this step is contained in the images.

Table 9 presents symbols used in this section.

Table 9: Symbols used in Coarse Matching section.

Symbol	Description
$G(\sigma)$	Gaussian kernel
$L(\sigma)$	an image convolved with a Gaussian kernel
L_i	first derivative of an image L , $i \in \{x, y\}$
L_{ii}	second derivative of an image L , $i \in \{x, y\}$
$DoG(\sigma)$	difference of Gaussian filter
H	the Harris corner detector
(η, ξ)	Gauge coordinates

3.2.1 Interest Point Detection

Since not all points from an image are matched, it is important to select only so-called interest points that are distinctive in an image. Hence, a common approach prior to point matching is to identify interest points. An *interest point* is a point that is easily discernible from its background and has diverse surroundings. For example, matching a white point on a white background is a poor choice, since all neighbouring points of the white point are indiscernible from it. Interest points should also be

easily detectable in an image that is subjected to one or more transformations. Usually rotation and scale changes are considered. Some researchers also consider affine invariant interest points [105]. For image registration, a required transformation is projective transformation. Unfortunately, projective invariants require several lines to be extracted from images [26, 60, 61], which tends to subvert the interest point detection process. In this section, several interest point detectors are discussed.

3.2.2 Scale invariant interest point detectors

The most common approach to detect scale-invariant interest points is through the use of a so called scale-space. The inventor of this approach is A. Witkin, who proposed “Scale-space filtering” in 1983 (see, *e.g.*, [56, 90, 107, 155, 193]). Other sources suggest this idea stems from the work of Taizo Iijima in Japan, in 1959 [187].

In addition to detecting the scale of an image, the spatial location of a point must be found. The most common approaches are the use of the Harris corner detector and the so-called difference of Gaussian approach.

Harris-Laplace detector

A *scale space* is a sequence of images parametrized by a scale. Each scale corresponds to a different resolution of an image. Rescaling is achieved by convolving an image with a Gaussian kernel¹⁸ [107, 191]. The Gaussian kernel is a preferred smoothing mask since Gaussian is the only filter that does not create zero-crossings as the scale increases [3], which means that the smoothing does not introduce any artifacts. In

¹⁸When referred to a convolution mask a word kernel is used interchangeably with word filter in image processing [48].

addition, a Gaussian filter is the only smoothing filter that can be applied (in one dimension) separately in vertical and horizontal direction and produces the same result as if 2D convolution was performed with 2D Gaussian mask. Let $I(x, y)$ denote a point in an image I , where x denotes the horizontal position of a pixel and y denotes the vertical position of a pixel. Let $G(x, y, \sigma)$ be a Gaussian kernel [27, 90], where

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}. \quad (16)$$

The representation $L(\sigma)$ of the image I relative to scale σ is given by a two dimensional convolution (see Definition 11) of an image I and the Gaussian kernel $G(\sigma)$.

$$L(\sigma) = G(\sigma) * I.$$

Let W denote the coordinates of a Gaussian kernel window (where point $(0, 0)$ denotes the centre of a window), the formula for any point at coordinates (x, y) with scale σ is given by

$$L(x, y, \sigma) = \sum_{s,t \in W} G(s, t, \sigma) I(x + s, y + t). \quad (17)$$

The resulting image in a higher scale is a blurred version of the original image. An example showing an image rescaled with factor $\sigma = 2.88$ as well as two images taken with zoom 1 and 2.88 are shown in Figure 32.

The symbols L_i and L_{ii} denote the first and the second derivatives of L in the direction of $i \in \{x, y\}$. The example of the first and second derivatives in the



Figure 32: Scale space filtering. Left and middle images - images taken with a zoom factor 1 and 2.88, respectively, right image - middle image rescaled with factor $\sigma = 2.88$

horizontal direction is

$$L_x(x, y, \sigma) = L(x + 1, y, \sigma) - L(x - 1, y, \sigma),$$

$$L_{xx}(x, y, \sigma) = L(x + 1, y, \sigma) - 2L(x, y, \sigma) + L(x - 1, y, \sigma).$$

Since L is the result of convolving the image I with Gaussian filter $G(x, y, \sigma)$, the derivatives can be calculated [156] by convolving the image with the derivatives of the Gaussian kernel

$$L_{i_1, \dots, i_N}(\sigma) = G_{i_1, \dots, i_N}(\sigma) * I.$$

The notion of a scale-space is used for automatic scale selection. This approach introduced by Lindeberg [90] is based on the fact that local extrema of spatial derivatives in a scale-space correspond to a scale of the image. This means that a scale-space approach allows for determining a magnification factor to each image.

In what follows, relative scale between two images can be found by determining the relative scale factor between the maxima of extrema of spatial derivatives. Mikolajczyk *et al.* [107] proposed four different spatial derivatives that can be used to determine a scale space: square gradient, Laplacian function, difference of Gaussian and Harris corners. Experiments reported in the literature suggest that the best results for detecting a scale of an image are obtained using the Laplacian function [90, 107, 108]. The interest point detector based on Harris corner detector and Laplacian scale detector is called the Harris-Laplacian detector.

To evaluate the advantages of the characteristic scale in the process of matching points, the characteristic scale algorithm has been introduced, implemented and tested as part of the research leading to this dissertation. This was done to assess the accuracy of an estimated scale assuming that the locations of the corresponding points are known. Therefore, to select matched points, information about an actual scale factor between the two images is used. That is, a smaller image is rescaled to match the dimensions of a bigger image. After rescaling, matching points in both images are found using a Zero-Mean Normalized Cross Correlation (for details, see Section 2.18). Then coordinates of matched points from a rescaled image are transformed back into the coordinates of an original image. The characteristic scale is calculated using the original image. The algorithm for calculating the scale factor between two images using the characteristic scale is shown in the Algorithm 10.

Table 10 shows the result of scale factor estimation using the square gradient.

$$F(x, y, s) = s^2(L_x^2(x, y, s) + L_y^2(x, y, s)).$$

Algorithm 10: Determining scale factor for two images

- 1 Rescale smaller image to match the image with bigger resolution (bicubic interpolation is used);
 - 2 Detect Harris corners in both images;
 - 3 Match Harris corners using the Zero-Mean Normalized Cross Correlation;
 - 4 Adjust location of matched points using ZNCC;
 - 5 Rescale back coordinates of matched points from the smaller image into its original scale;
 - 6 Find characteristic scales for matched points.
-

The first column in Table 10 shows the actual scale factor between two images. The second column shows the scale factor calculated using the characteristic scale and the third column shows the percentage of error of an estimation. The last three columns show the number of points for which the characteristic scale was calculated, the number of points for which the matching point was found and the percentage of pairs for which the characteristic scale was calculated, respectively. The bigger image had dimensions 2046 by 1215 pixels, and the smaller image dimensions varied from 2034×1200 to 715×422 pixels.

Table 10: Scale factor estimation for square gradient.

scale	est. scale	scale error	scale found	pairs detected	% found
1	1.0093	0.9%	1129	1783	63.3%
1.1335	1.1592	2.3%	994	1730	57.5%
1.295	1.3119	1.3%	733	1373	53.4%
1.4892	1.4724	-1.1%	453	821	55.2%
1.6904	1.5800	-6.5%	366	767	47.7%
1.9098	1.8253	-4.4%	218	433	50.3%
2.17	2.0563	-5.2%	126	274	46.0%
2.4897	2.2916	-8.0%	38	84	45.2%
2.8804	2.3398	-18.8%	35	90	38.9%

The results in Table 10 confirm good results for small scale change obtained by Mikolajczyk *et al.* [107]. For scale changes up to 1.5, the scale estimation error is no greater than 2.3% and over 50 percent of the pairs used for calculations yielded the characteristic scale. But the key assumption for successful scale space filtering is that pairs of images differ only in scale. For the experiments summarized in table 10, images taken from one viewpoint were considered. For the problem of 2D to 3D conversion, images are taken from different locations. Hence, information in the images is changed. This imposes two additional problems not usually considered by researchers using the scale space approach. First, to find a scale factor between the two images, pairs of points from both images must be known. For this research, the objective of 2D processing is to find point matches. Thus the scale space approach uses information that is being sought. Second, in the case of a projective transformation, information contained in images is different. This influences the characteristic scale and the scale factor between the images. As a result, the accuracy of a calculated scale factor decreases even further. Even though the scale-space approach has been incorporated in several interest point detectors [28, 107, 108, 193], due to the reasons described above it is not used in this research.

Difference of Gaussian (DoG)

The idea behind the "difference of Gaussian" (DoG) [95, 96] detector is similar to that of Harris-Laplacian. For the DoG detector, an image is smoothed using a Gaussian filter (see (16)). Both the spatial and scale-space extrema are found directly from smoothed images. To find a scale-space extremum, more than one image

is necessary. A collection of images is created, where images from this collection are parametrized by a scale factor. All adjacent images are subtracted. They form *difference of Gaussian* images. The formula for a DoG image is:

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

A collection of DoG images is used to find the extrema for scale and space. A search for scale-space extrema is performed in 3D space (two spatial coordinates and a scale). A point is assumed to be extremal if it has a extremum value in its 26-th neighbourhood. A 26-th neighbourhood consists of 9 points from a lower scale, 9 points from a higher scale and the eight-neighbourhood from the current scale.

3.2.3 Rotation invariant interest point detectors

This section contains a description of interest point detectors that are invariant relative to image rotation. The only commonly used point detector that is rotation invariant is the Harris corner detector. For more information about obtaining rotational invariance, see Section 3.2.5.

Harris corner detector

The Harris corner detector [58] is a measure for identifying points in an image for which colour information varies in all directions. By contrast, edges are areas for which pixel colours vary only when crossing an edge as opposed to moving along an edge. The Harris corner detector is a very important tool for image registration, since it makes it possible to identify points that are distinct from the background.

These corners are also rotation invariant. Therefore, they are often used as interest point detectors in image registration. In fact, the Harris corner detector is the most commonly used detector in image processing, for example see [74, 113, 175, 177].

Let $L(x, y, \sigma)$ denote the Gaussian filter as defined in (17). Moreover, let μ be the second moment matrix

$$\mu(x, y, \sigma) = \begin{bmatrix} L_x^2(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_y^2(x, y, \sigma) \end{bmatrix},$$

$\det(\mu)$ denote a determinant of a matrix μ and $Tr(\mu)$ denote the trace of a matrix μ . The parameter k is usually set to 0.04. The cornerness [58, 108] of an image at a given location (x, y) is measured using

$$H = \det(\mu(x, y, \sigma)) - k \cdot Tr(\mu(x, y, \sigma)). \quad (18)$$

The process of finding the corners in an image consists of two main steps. First, the cornerness of each point in the image is found. Second, the points with maximum cornerness in some neighbourhood are selected. These points are assumed to be the corners in the image. The algorithm for finding the Harris corners [58] is shown in Algorithm 11. Figure 33 shows the Harris corners detected for the image of a power tower structure.

Algorithm 11: Harris corner detector

- 1 Calculate the first derivatives in horizontal $\partial I/\partial x$ and vertical $\partial I/\partial y$ directions
 - 2 Square derivatives and smooth them using Gaussian filter G
 $A = (\partial I/\partial x)^2 * G,$
 $B = (\partial I/\partial y)^2 * G,$
 $C = ((\partial I/\partial x)(\partial I/\partial y)) * G.$
 - 3 Create 2x2 matrix
 $M = \begin{bmatrix} A & C \\ C & B \end{bmatrix}.$
 - 4 Calculate the corner measure R for $k = 0.04$
 $H = \det(M) - k\text{Tr}(M) = AB - C^2 - k(A + B).$
 - 5 Select given point as a corner if H is maximum in 3×3 neighbourhood
-

Scale-space adapted Harris corner detector

A scale-invariant Harris corner detector is an adaptation of the classical Harris corner detector to the scale-space [28, 108]. In addition to smoothing an image before calculating the second order matrix, the result is also averaged using another Gaussian kernel σ_I (integration scale). The image smoothing kernel σ_D is called the derivation scale. The scale-adapted second moment matrix is given by

$$\mu(x, y, \sigma_I, \sigma_D) = \sigma_D^2 G(\sigma_I) * \begin{bmatrix} L_x^2(x, y, \sigma_D) & L_{xy}(x, y, \sigma_D) \\ L_{xy}(x, y, \sigma_D) & L_y^2(x, y, \sigma_D) \end{bmatrix}. \quad (19)$$

Cornerness is calculated like in the classical Harris corner detector, see 18, but the scale-space adapted second moment matrix is used.



Figure 33: Harris corners

Comparison of interest point detectors

Because of the usefulness of interest point detectors, the literature is rich in comparisons between them. C. Schmid *et al.* [153, 154] tested different interest point detectors. They divided them into three categories: contour based, intensity based and parametric model based methods. In their work, they compare five detectors: Harris [58], Cottier [23] as cited by [154], Horaud [67] as cited by [154], Heitger [64] as cited by [154] and Förstner [39] as cited by [154]. For the comparison, they use the repeatability rate that is defined as a ratio of a number of repeatedly detected points in the original and transformed image to the total number of detected points.

For rotation invariance, the best result is obtained by the improved Harris detector. But the result strongly depends on localization error. For a sub-pixel resolution, the repeatability oscillates around 0.4 and for an error equal to 1.5 pixels, the repeatability is close to 1. All of these results are independent of the rotation angle with the maximum rotation angle equal to 180 degrees.

For scale change, the best detectors are again the Harris and Cottier detectors. In this case, the results are much worse compared to rotation. For scale factor greater than 2.0 and sub-pixel resolution, the repeatability is smaller than 0.1. For an error of 1.5 pixels, the repeatability drops from around 0.9 (scale ≈ 1.1) to 0.4 (scale ≈ 2) and then it drops below 0.2 for bigger scale factors.

In this research, the scale factor should rarely exceed 2. Nevertheless, the results obtained by C. Schmid *et al.* are not promising for this research. The rotation and scale changes are not the only transformations present for the images taken from different viewpoints. Extra differences are added by different lighting conditions, perspective transformations and occlusions. C. Schmid *et al.* performed more tests, *e.g.*, with respect to illumination and affine transformations. The results are very similar to those with a scale change and a rotation (repeatability from 0.5 to 0.9 for illumination change and from 0.3 to 0.9 for affine transformation and with an error of 1 pixel). This means that if all of these factors are present, namely rotation, scale change, illumination and affine transformation, repeatability is certain to be low enough to make even the best interest point detector useless as the only means of registration. Nonetheless, interest point detectors are still very useful as additional tools for image registration, especially in cases where due to some extra information the ambiguity in registering points can be decreased.

3.2.4 Affine invariant interest point detectors

An affine invariant interest point (AIIP) detector's goal is to select points in images regardless of the affine transformation between the images. Usually, due to the

specific nature of the transformations between two images taken from different positions by a digital camera, the set of all possible affine transformations is narrowed to translation, rotation and independent scaling in the horizontal and vertical directions. Affine invariant interest point detectors are very important in image matching for 2D to 3D conversion, since the projective transformation for the planar surfaces can be approximated¹⁹ by an affine transformation [37, 105, 109].

The AIIP detector implemented as part of the research reported in this dissertation is based on the article by Mikolajczyk and Schmid [105, 108]. This detector uses the multi-scale Harris detector (as described in the previous section). The AIIP algorithm consists of four main steps.

1. The spatial localization of a point is determined by multi-scale Harris detector.
2. The characteristic scale is found. This scale is constructed using a spatial derivative function that is an application of the normalized Laplacian $|\sigma^2(L_{xx}(x, y, \sigma) + L_{yy}(x, y, \sigma))|$.
3. A derivation scale is found (see, *e.g.*, 19). This scale is set to the value that maximizes the ratio of the smallest and greatest eigenvalues of the second moment matrix

$$\sigma_D = s\sigma_I \quad \text{such that} \quad s = \underset{s \in [0.5, 0.75]}{\operatorname{argmax}} \frac{\lambda_{\min}(\mu)}{\lambda_{\max}(\mu)}.$$

4. Finally, the neighbourhood of a detected point is normalized using the second moment matrix. This normalization is performed by warping the image [61].

¹⁹The approximation is performed locally around pixels and it helps in pixel matching.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x_o \\ y_o \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x - x_o \\ y - y_o \end{bmatrix},$$

where $[x_o \ y_o]^T$ is the transformation centre and the matrix A is calculated from the second moment matrix as:

$$A = \mu^{-\frac{1}{2}}(x, y, \sigma_I, \sigma_D).$$

Figure 34 illustrates affine invariant points detected by the above method. Each ellipse denotes an enclosed area after normalization. The size of the ellipse corresponds to the characteristic scale detected for a given point. The size and position of these ellipses are very important for matching points. After normalization, neighbourhoods of the selected points are similar in both images. The only difference between them is the rotation. To match corresponding points, it is necessary to extract rotation invariant descriptors and match them.



Figure 34: Affine invariant interest points

3.2.5 Interest Point Descriptors

Detecting interest points in an image is the first step in solving the image matching problem. To be useful, an interest point detector must detect the same points in a pair of images. The problem of pairing corresponding interest points is still an open problem.

Due to viewpoint change, the information in the neighbourhoods of corresponding pixels is different in both images. Therefore, attempts to match points using the information directly contained in the images may not give satisfactory results. To facilitate matching, an approach widely used in pattern classification²⁰ is applied. Using the SVM approach, an image space is transformed into a feature space. The features in the new feature space are invariant with respect to the transformations. Matching of extracted features is more robust than using information directly contained in an image. In addition, the dimensionality of the new feature space can be significantly reduced (although search space reduction is not the main goal).

The set of features extracted from an image for a given point is called a *descriptor*. A comparison of descriptors is usually performed by calculating the Mahalanobis distance between them [106, 155], see Section 3.2.6. Mikolajczyk *et al.* in [106] compare five different interest point descriptors: SIFT, steerable filters, differential invariants, complex filters and moment invariants. The following subsections contain descriptions of all of these descriptors.

²⁰Support Vector Machines (SVM) [151]

Scale Invariant Feature Transform (SIFT)

The first descriptor is called a SIFT descriptor (Scale Invariant Feature Transform) that was proposed by Lowe [95, 96]. The interest point detector used by Lowe is the *difference of Gaussian*. To ensure rotation invariance, gradient orientations are used. For each detected point, gradient orientation and magnitude are calculated. Gradients for points around a given interest point form a histogram. A peak in this histogram denotes the orientation of a given interest point. Further calculations of the SIFT descriptor are performed relative to this orientation.

The SIFT descriptor is calculated in the following manner.

1. The gradient orientations and magnitudes for points in the neighbourhood of a given point are used. Assume that the neighbourhood was chosen to be an 8×8 pixel wide window around a given interest point. This means that 64 gradient orientations and magnitudes are found.
2. Gradient magnitudes are accumulated over their neighbourhoods. Instead of an 8×8 pixel window, a smaller window such as a 2×2 pixel window is used. Each entry in the new 2×2 window consists of r vectors. Each vector represents one direction. The length of a vector is a sum of all of the magnitudes of the gradients that are closest to a particular direction²¹.
3. To improve descriptor stability, gradient magnitudes are weighted by their distance to the interest point.

²¹If the smaller window size is $n \times n$ for $n \leq 8$, then the descriptor dimension is $n \times n \times r$. Lowe *et al.* determined that the best stability and discrimination was obtained for $n = 4$ and $r = 8$. This produces a descriptor with length 128.

Steerable Filters

Steerable filters were introduced by Freeman and Adelson in [38]. The main advantage in using steerable filters is a reduction in computational complexity in calculating image gradients.

Image gradients are commonly used to define the orientation of an image relative to a given point. Before two image points can be matched, they need to be aligned (rotated) with respect to their orientation angle. Therefore, finding the gradient for a given point is crucial for further matching.

To find a gradient in a given direction, it is necessary to calculate the first derivatives of an image in a given direction. Usually, to find a gradient's orientation, one must find derivatives in several directions. Freeman and Adelson showed that instead of calculating several derivatives, it is possible to calculate only a few base derivatives and interpolate the remaining angles relative to these base derivatives. Usually, basis filters are chosen to be aligned along horizontal and vertical axes. This result is very appealing in the field of digital image processing, since the calculation of derivatives in the horizontal and vertical directions is very fast.

To see how this is accomplished, consider the Gaussian kernel

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}.$$

The derivative of G in direction θ can be calculated interpolating the derivatives of G in the horizontal and vertical directions as:

$$G^\theta(x, y, \sigma) = \cos(\theta) G^{0^\circ}(x, y, \sigma) + \sin(\theta) G^{90^\circ}(x, y, \sigma),$$

where $G^{0^\circ}(x, y, \sigma) = \frac{\partial}{\partial x}G(x, y, \sigma)$ and $G^{90^\circ}(x, y, \sigma) = \frac{\partial}{\partial y}G(x, y, \sigma)$.

In addition, since convolution is a linear operation, the result of convolving an image with a kernel G^θ is identical to the linear combination of two images convolved with kernels G^{0° and G^{90° . That is, if $L(\sigma) = G(\sigma) * I$ and $L^\theta(\sigma)$ denotes the derivative of $L(\sigma)$ in a direction θ , then

$$L^\theta(\sigma) = \cos(\theta) L^{0^\circ}(\sigma) + \sin(\theta) L^{90^\circ}(\sigma).$$

The above equation makes it possible to calculate the image derivatives only once and then use them to interpolate the derivatives in any orientation θ .

Differential Invariants

The term *differential invariants* covers a wide range of invariants based on image derivatives. The foundations of this approach were presented by Koenderink *et al.* in [78] and used by many other researchers [155, 156, 176].

Calculations of differential invariants can be divided into two steps.

1. A so called *local jet* is found. A *local jet* is a set of derivatives relative to a given point. Before the derivatives are calculated, an image is smoothed using the Gaussian kernel. As a result, this process takes into account the derivation

scale:

$$J^N(I)(x, y, \sigma_D) = \{L_{i_1, \dots, i_n}(x, y, \sigma_D) \mid (x, y, \sigma_D) \in I \times \mathbb{R}^+; n = 0 \dots N\}.$$

For example, L_{ii} denotes the Laplacian $L_{ii} = \sum_{i=x,y} L_{ii} = L_{xx} + L_{yy}$ and $L(x, y, \sigma)$ is given in (17).

2. In the second step, a set of differential invariants is created from the members of a local jet.

A set of differential invariants is not unique and depends on a researcher's preferences. Due to the result obtained by Hilbert in 1890 (as cited by [58, 51, 50, 52, 156]), the problem of finding invariants was reduced to a small set of irreducible invariants. Hilbert showed that any invariant of finite order can be expressed as a polynomial in a base of irreducible invariants. As a consequence, most interest is directed toward finding sets of irreducible invariants. For example, for two dimensions, the set of irreducible invariants up to the order two is equal to:

$$L, \quad L_i L_i, \quad L_{ii}, \quad L_i L_{ij} L_j, \quad L_{ij} L_{ji}.$$

As proposed by Schmid and Mohr [155, 156, 157] the differential invariant descriptor consists of all irreducible invariants up to the order of three. It is assumed that the source image is a grey-scale image.

Denote by ϵ_{ij} the 2D antisymmetric Epsilon tensor:

$$\epsilon = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

A vector containing all irreducible invariants up to the order of three is equal to:

$$\nu = \begin{bmatrix} L \\ L_i L_i \\ L_{ii} \\ L_i L_{ij} L_j \\ L_{ij} L_{ji} \\ \epsilon_{ij} (L_{jkl} L_i L_k L_l - L_{jkk} L_i L_l L_l) \\ L_{ii} L_j L_k L_k - L_{ijk} L_i L_j L_k \\ -\epsilon_{ij} L_{jkl} L_i L_k L_l \\ L_{ijk} L_i L_j L_k \end{bmatrix}.$$

This set of descriptors is invariant with respect to image rotation, scale change and small viewpoint variations.

Another approach proposed by Montesinos *et al.* [50, 51, 52] takes into account colour information contained in an image. Instead of using an image coordinate system (x, y) , they calculate invariants in the *Gauge coordinate* (η, ξ) . The term 'gauge condition' comes from physics and denotes a situation where a coordinate frame is oriented in such a way that one or more of the partial derivatives is zero [91, 176]. In digital image processing, one of the directions in a gauge coordinate

system is parallel to the gradient direction $\eta = \frac{\nabla I}{|\nabla I|}$ and the second direction is perpendicular to the first one $\xi \perp \eta$. Montesinos *et al.* proposed five invariants that are suitable for greyscale images [50, 51, 52].

$$I, \quad I_{\eta\eta} + I_{\xi\xi}, \quad I_{\eta}, \quad \frac{I_{\xi\xi}}{I_{\eta}}, \quad \frac{I_{\xi\eta}}{I_{\eta}}.$$

For colour images, Montesinos *et al.* proposed a descriptor consisting of eight elements constructed from gradients of red, green and blue channels.

Invariant Moments

Invariant moments were popularized in the pattern recognition community by the work of Hu in 1962 [70]. Hu proposed seven invariants with respect to rotation. These invariants are based on central moments that are defined in this section (all equations in this section are taken from [48] and [70], if not stated otherwise).

The geometric moment for a given point (x, y) is equal to:

$$m_{p,q} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q I(x, y) \, dx dy. \quad (20)$$

The central moment is equal to:

$$\mu_{p,q} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q I(x, y) \, dx dy, \quad (21)$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}.$$

By normalizing the central moments, these moments become invariant relative to scale change [98]:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \quad \gamma = \frac{p+q}{2} + 1.$$

The formulae for seven invariants introduced by Hu are:

$$\Phi_1 = \eta_{20} + \eta_{02},$$

$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,$$

$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2,$$

$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2,$$

$$\Phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] +$$

$$(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2],$$

$$\Phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}),$$

$$\Phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] +$$

$$(3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].$$

In theory, it is possible to match two points in an original and transformed image using these invariants. But in practise, there are two problems. First, instead of

(21) for calculating central moments, the formula²²

$$\mu_{p,q} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y).$$

is used. It is necessary because the function I for digital images is discrete. In changing the domain from continuous to discrete, the invariant moments partially lose their properties. For example, rotation of an image has the biggest influence on invariant moments due to the finite number of sample points. The second problem is the nature of a projective transformation. As mentioned before, the seven moments are invariant with respect to translation, rotation and scale change, but not projective transformation.

Flusser *et al.* [36] have used moment invariants to match regions detected in two satellite images. The moments they used in the proposed method are invariant to affine changes.

$$\begin{aligned} \Phi_1 &= \frac{1}{\mu_{00}^4} (\mu_{20}\mu_{02} - \mu_{11}^2), \\ \Phi_2 &= \frac{1}{\mu_{00}^{10}} (\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{03}\mu_{21}^3 - 3\mu_{21}^2\mu_{12}^2), \\ \Phi_3 &= \frac{1}{\mu_{00}^7} (\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)), \\ \Phi_4 &= \frac{1}{\mu_{00}^{11}} (\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2 \\ &\quad + 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} - 8\mu_{11}^3\mu_{30}\mu_{03} \\ &\quad - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21}^2 + 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} + \mu_{02}^3\mu_{30}^2). \end{aligned}$$

²²For better approximations of true moments see [89].

Nevertheless, the differences between images from satellites are small compared to the differences taken by a hand-held camera and the authors pointed out several shortcomings of this method.

Similarly to differential invariants, moment invariants have been designed for grey-valued images. Mindru *et al.* in [110] extended this idea to colour images. Recognition rates for a data base with 200 samples varied from 64% to 96%.

The moment invariants presented in this section are invariant with respect to translation, scale change, rotation and affine changes. Suk *et al.* [170] showed, contrary to common belief, that moment invariants that are invariant to projective transformation exist. Unfortunately, such moment invariants can be expressed only in a form of infinite series. Due to their infinite nature and errors caused by approximation using a finite sum, this result has theoretical but not practical importance in image processing.

Complex Filters

Complex filters ([37]) are very similar to moment invariants. Instead of using 20, a formula with complex numbers is used, namely

$$c_{p,q} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x + iy)^p (x - iy)^q I(x, y) dx dy.$$

An example of an application of complex filters to point matching is shown in [152].

Comparison of Interest Point Descriptors

Mikolajczyk and Schmid compared several mentioned earlier descriptors [106]. In addition to descriptors, they also studied cross-correlation for detected interest points. Table 11 shows approximated results (read from graphs from [106]) of the detection rates for different descriptors. A *detection rate* is the number of correctly matched points relative to the number of possible matches. The results in Table 11 correspond to a false positive rate of 0.006. The false positive rate is the total number of false matches divided by the product of the number of database points and the number of image points. Values below 0.012 allow for reliable scene recognition [106].

Table 11: Interest point detector point match detection rates.

descriptor	rotation	scale & rotation	affine	illumination
SIFT	98.8	99.7	83.6	96.9
steerable filters	95.2	96.6	79.7	99.4
moments	91.9	89.5	71.6	94.4
complex filt.	86.2	71.4	52	94.8
diff. inv.	75.4	57.9	44	87.6
cross-correl.	97.1	68.2	40	97.4

The best results are obtained by the SIFT descriptor. For this research, the transformation that is most important as well as the most difficult to deal with is the affine transformation. As can be seen from Table 11, three descriptors, namely SIFT, steerable filters and moments, are superior compared to the other descriptors. Due to its high performance, the SIFT descriptor is considered next.

3.2.6 Matching

An algorithm used for matching points depends on the information that is available for matching point pairs. When interest point descriptors are considered, matching is performed by minimizing the distance between the descriptors. The matching problem can be formulated as a problem of finding the closest vectors in a descriptor's space. The most common approach is to use the Mahalanobis distance [106] (see Definition 3) or Euclidean distance. If pixel values are used instead of descriptors, the most common measure is the sum of squared distances (see (22)) or correlation (see Section 2.18).

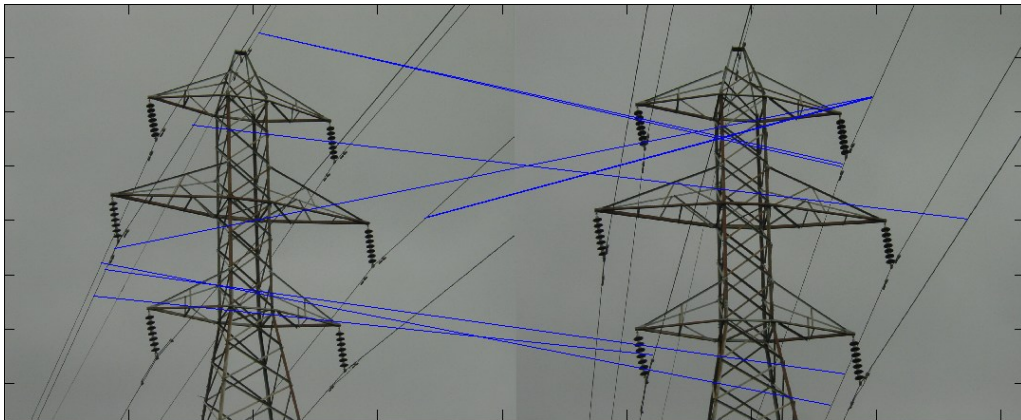


Figure 35: Results of SIFT detector matching.

In what follows, the SIFT descriptor is used to detect and match points in two images²³ of power towers (see Figure 35). In the experiments, the implementation by [183] is used (see also [95, 96, 97]). The matching of descriptors is performed using the Euclidean distance (as suggested by [95] and cited by [106]).

²³The resolution of each image is 2048 by 1536 pixels

The results of experiments indicate that the SIFT descriptor does not have enough discriminative power to be used as a point detector for power tower images. For the images in Figure 35, the SIFT descriptor is not able to find one accurate match (blue lines join matched points). This means that the point matching step would not benefit from using this descriptor. As indicated by Table 11, the SIFT descriptor is the best one from considered six interest point descriptors. Based on the fact that the SIFT descriptor is not able to match points of power tower images and is the best of tested descriptors, it is concluded that the use of other descriptors would not be beneficial for the process of point matching.

As shown in the previous paragraphs, interest point descriptors do not have enough discriminative power to be used for point matching for power tower images. The reason for this stems from the fact that power tower images do not contain many details. Most areas such as sky or steel are plain (fairly featureless). On the other hand, the areas with high variability such grass and trees can change between images due to wind conditions.

Next, consider the close up of two insulators given in Figure 36.

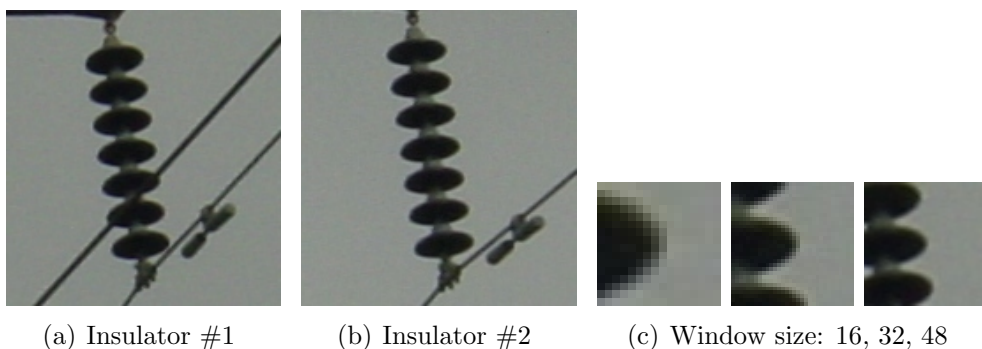


Figure 36: Close-up of two insulators from Figure 35

Matching part of the insulator is very difficult, since an insulator consists of several circular plates that are identical. Consider, for example, the close-ups shown in Figure 36(c). When using a 16×16 pixel window (the first one from the left in Figure 36(c)), it is impossible to tell which part of the insulator it shows. After doubling the window size to 32×32 (this is the middle image in Figure 36(c)), it is possible to determine if the window is centred on the first or last plate, but it is still impossible to determine which insulator is shown in the window. Even after increasing the window size by another 16 pixels in each direction, the situation is hardly improved. The process of increasing the window size increases the information that is used from the image. In effect, the matching is less precise, because the differences between both images start to be influential. At the same time the matching is more robust to the changes in viewpoint because it is harder to mismatch a big portion of an image. As a general rule, the bigger the windows used for matching, the more robust to the change of the viewpoint is the matching process. The smaller the windows, the more precise the point matching, but at the same time the number of false matches increases. In addition, the bigger the window, the bigger the computational cost.

As suggested by [166], the SSD metric is one of the most discriminative among several metrics tested by the author, namely, cross-correlation, zero mean cross-correlation, χ^2 test, Kolmogorov-Smirnov distance, Jeffrey divergence and Earth Mover distance. Due to its simplicity in implementation, many researchers choose the SSD metric for point matching [35, 83, 102, 103, 112, 129, 166, 174]. Other authors prefer the Zero-Mean Normalized Cross Correlation (ZNCC) measure due

to its robustness relative to illumination changes [14, 83, 85, 85, 87, 88, 171, 172]. In this project, the point matching step uses ZNCC (see Section 2.18).

As a point detector, the Harris corner detector is used. First, all Harris corners in both images are found. Then, to speed up the calculations, matching is performed using several different-sized windows. Matching starts with a small 5×5 pixel window. All pairs of points for which the ZNCC is smaller than some threshold are deemed putative matches and are kept for subsequent iterations. In what follows, the window size is increased by 5 pixels in each iteration and all pairs from the current iteration are checked again. All pairs for which the ZNCC is bigger than some threshold are removed from the set of putative matches. The iteration stops when the window size is bigger than 40 pixels. At the end of the matching process, all pairs of points are checked against the triangle property (see Section 3.1.5) and those pairs not satisfying this condition are removed.

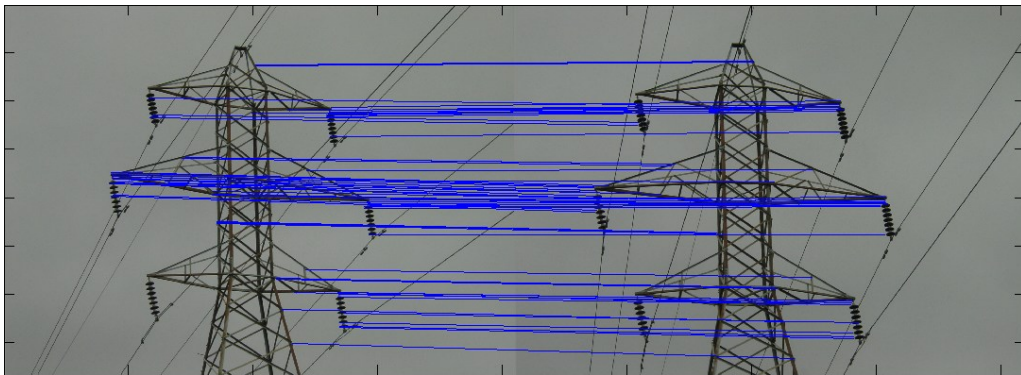


Figure 37: Results of ZNCC matching.

Sample results of the proposed point matching procedure are shown in Figure 37. It can be observed that the proposed point matching method is very robust because

the information about scale change and rotation between the two images is known from the previous step, namely, coarse matching. This means that two images can be rotated and rescaled before ZNCC matching is attempted. As a result, matched areas are very similar and the matching is very robust. For a maximal window size of 20 pixels, the percentage of false matches is below 10% and for a maximal window size of 40 there are no false matches (but the number of matched points is much smaller). In the former case, 70 point matches were found and in the latter only 13.

3.3 Dense Matching

The dense matching problem has received a great deal of attention by the Computer Vision Society, since it is a crucial step in 3D reconstruction from digital images. Various researchers have studied different approaches such as dynamic programming [6, 77, 177] or Voronoi diagrams [174]. A common dense matching technique is based on a region growing procedure [85, 86, 102, 174]. First, using some additional constraints, only a small number of matches are determined with higher accuracy. Then the matched areas are expanded preserving the original matching continuity. The last step in the proposed dense matching algorithm is based on this scheme.

The framework presented in this section is similar to the one proposed by Bufama *et al.* in [13] in the sense that the calculations are performed separately for edge pixels and non-edge regions. It is easier to impose constraints on edge pixels, making them more robust to mismatches. The process of constraining the search on some group of pixels is the most widely used technique to increase the accuracy of matches. The proposed algorithm limits the search of pixels to be matched to those located on a line in 3D space.

The problem of detecting lines in 3D space is a classical problem in the pattern recognition area. The standard method for detecting lines in the 2D space is a Hough transform (see Section 2.7). In 3D space, the Hough transform is used to detect planes, see for example [184]. Lines in 3D space can be detected using the Hough transform twice. This approach was described before by D. Katsoulas in his Ph.D. dissertation [75] and also in [4, 76]. These authors work with range data, *i.e.*, points in 3D space for which all three coordinates are already known. The proposed

dense matching approach differs from the earlier approach due to the fact that the points considered are not converted to 3D space.

An approach to a solution to the problem of noiseless reconstruction of 3D objects dominated by sets of line segments is obtained in two stages. In the first stage, points on lines in 3D space are found and matches corresponding to these points on lines are identified. In the second stage, identified matches are used as a seed points in dense matching of two views. Additional constraints control the process of dense matching in such a way that the original information from lines is propagated through the entire image without the “noise” (*e.g.*, blurring or extraneous parts of the original image). The end result is a noiseless 3D image. The problem of reducing the noise of range data representing the power structures is crucial for the process recovering an accurate model of power system structures to facilitate detection of deformities (*e.g.*, tilting due to winter ground frost heaving and twisting of tower structures due to wind action). Melzer *et al.* in [104] study the problem of fitting range points belonging to the power line to a Mecenary curve. This problem is similar to the problem considered in this section, but again here, the range data are not available and a disparity maps are used instead.

This dissertation proposes a new dense matching method that utilizes the presence of straight lines in the images. The goal is to robustly extract the structure shown in the image rather than to recover locations of all the pixels in the image. The main application for this algorithm is the extraction of 3D data from the images of electric power transmission towers. In most cases, such images are taken from the ground and do not contain any textures. And the textures contain most information

needed for the dense matching process. In addition, the background is usually much brighter than the tower, making the use of colour information very difficult. Using the algorithm proposed here it is possible to successfully extract the structure of a tower despite of the mentioned problems.

The dense matching is performed after the point matching, 2D to 3D conversion and rectification of both images have been completed in a 2D to 3D conversion system.

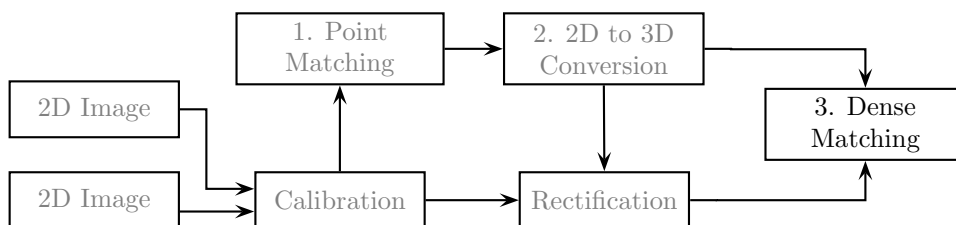


Figure 38: Overview of the 2D to 3D conversion system.

In the point matching step in Figure 38 (see also Section 3.2), several points are matched from two given images. These matches are used in 2D to 3D conversion to recover extrinsic camera parameters, *i.e.*, location and orientation of the cameras. Once the extrinsic camera parameters are known, one can use epipolar geometry to reduce the dimensionality of the dense matching problem [41]. In this step, a substantially larger number of pixels (compared with point matching) is matched.

The dense matching approach introduced in this dissertation consists of the following three main steps.

1. Edge detection;
2. Matching using lines in 3D;

3. Dense matching using seeds from the step 2.

Starting with edge detection, points of interest are selected from both images. Note that, information about a structure is usually contained in the edges of a structure. In a 2D image, 3D edges are usually projected onto curves with areas of different colours on both sides. The best way to detect these edges is to find all edges in the images. In the first step in dense matching, only edge points are processed.

In the second step (matching lines in 3D), a disparity map is constructed and depth information is used to filter out points that were mismatched (see Section 3.3.1 for a formal definition). Without any post-processing, the disparity map is very noisy, *i.e.*, many points are mismatched resulting in the wrong location in the 3D space. The proposed algorithm aims at extracting the 3D information from the images, assuming that there are a lot of straight lines in the images. The algorithm extracts straight lines in the 3D space, which makes it possible to determine the 3D structure of an object. Generally, it is difficult to extract lines in 3D space. Therefore, the extraction of lines is performed in two stages. First, lines are extracted in the image space, *i.e.*, in the 2D image plane. Then, for the points that have been extracted, lines are extracted using depth information.

In the third step in dense matching, information about correct matches for straight lines is propagated further from the edges. Correctly matched edges are used as a seed points in the process of dense matching.

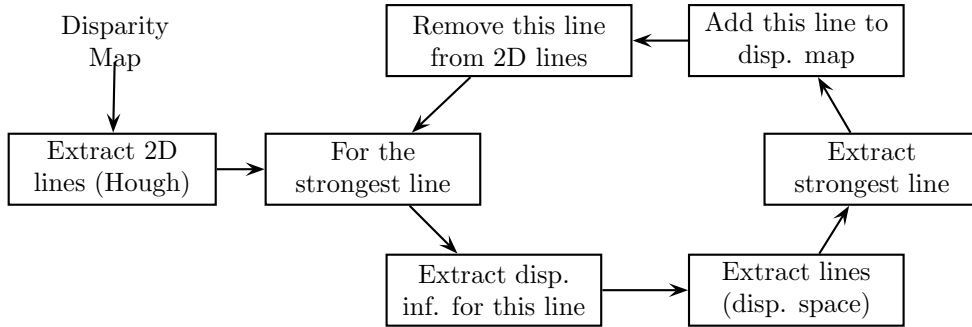


Figure 39: Overview of the algorithm extracting 3D straight lines.

3.3.1 3D Line Extraction

The input for this stage of the dense matching algorithm are points selected in the previous step, *i.e.*, edge points in both images. In this step, these points are matched using both, the information contained directly in the image and the depth information.

First, the tentative matches between the two images are found. The matching algorithm was chosen to be a form of template matching. The simplest (regarding the computational power) and the most discriminative [166] measure is the sum of squared differences. The formula for calculating the SSD is:

$$SSD = \sum (i_{image} - j_{target})^2, \quad (22)$$

where i and j denote the source image and target intensities, respectively. The rotation of the image has a huge impact on the performance of the SSD measure, but since there is no rotation present (due to the image rectification), the SSD measure is well suited for the pixel matching. A pair of matched points, where for

the point in the left image its counterpart in the right image is found, is called a “left to right” match or *LR* match. The match in reverse direction is called “right to left” match or *RL* match. Finding a match for each pixel is equivalent to finding a *disparity map*. A disparity is a difference of matching pixel’s horizontal coordinates [71]. A disparity map is an image, in which each pixel’s value is a disparity of a corresponding pixel from the second image. A sample disparity map for a jet plane scene is shown in Figure 43. Notice, the further the points from the camera the smaller the disparity.

After calculating the disparity, the main loop of the algorithm begins (see Figure 39 for the overview of presented algorithm). First, a Hough transform for the entire image is calculated. This allows to detect the lines that appear to be straight lines in the 2D original image (see Figure 41(a) for a sample line). It does not mean that these are straight lines in the 3D space yet. The process of selecting 3D straight lines is two fold. First, the lines in the image plane are selected. These lines form a pool *OL* of 2D lines in the original image. During extraction of a line from the original image only one coordinate of a pixel’s location is saved. In other words, all pixels belonging to a line are projected onto one dimensional space aligned with the original line. As a result, one dimension is dropped and the points belonging originally to a line are considered as a one dimensional set of points.

Next, for the strongest line *L* in the pool the following steps are performed. First, the disparity information about all the points belonging to *L* is extracted. The disparity introduces one extra number for each point, which raises the dimensionality of the set *L* by one. The resulting set is again two dimensional, and it contains also

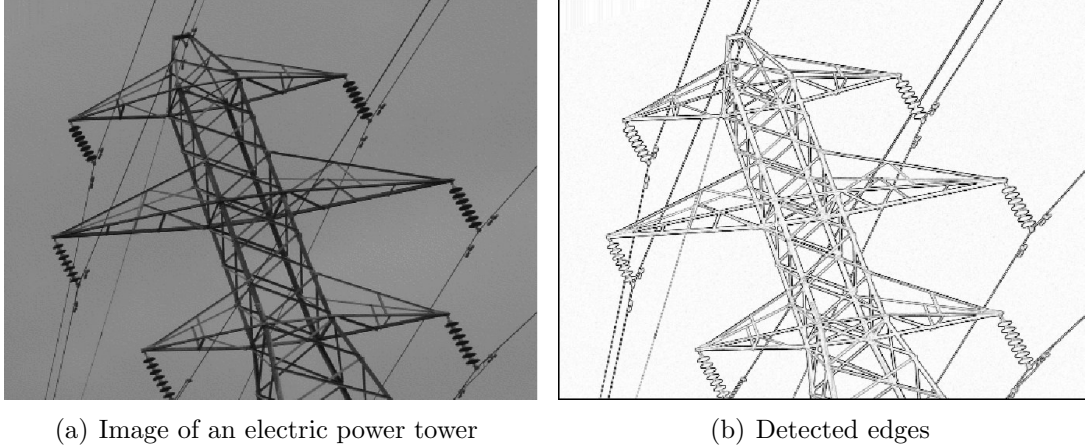


Figure 40: Image of an electric power tower and detected edges

depth information. Since one dimension denotes the position on the line and the second dimension denotes the disparity, this new two dimensional space is called a LD space (Line-Disparity space). Figure 41(b) shows an example of a line from Figure 41(a) converted to the LD space. Since all the points from L were extracted along a line, some of them correspond to a real line in the 3D space. Therefore, the disparities of these points should form a line in this new two dimensional space. In the next step, the Hough transformation is applied to this set. The strongest line L' is extracted (see Figure 42(a)) and all the pixels belonging to this line are considered to be correct matches. Only points belonging to this line are added to the final disparity map. All points considered to be correct matches are shown in Figure 42(b). Finally, all the points from L that gave rise to the points in L' are removed from OL . This process is repeated until the set OL is empty.

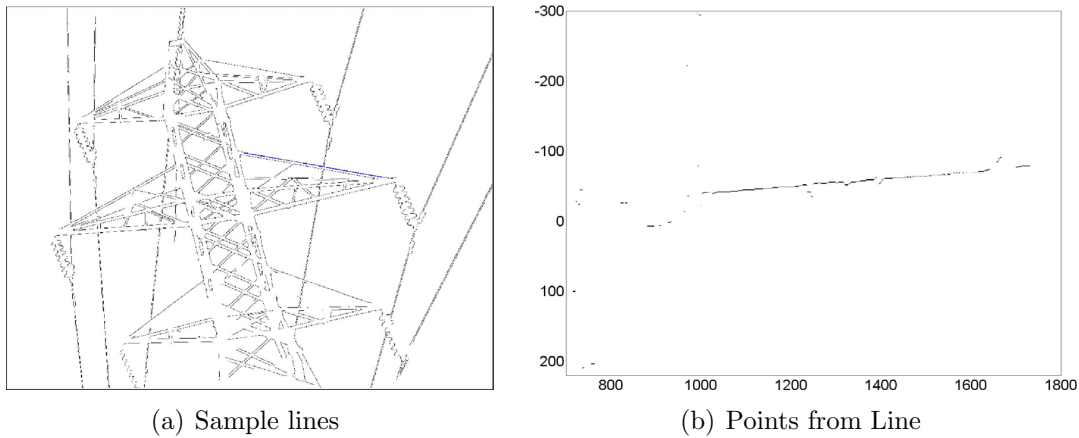


Figure 41: Sample lines and points from line in LD space

3.3.2 Dense Matching

The previous step removed all the points from disparity map that did not belong to some line in the 3D space. It is assumed that the remaining points are correctly matched points and belong to the structure of the object being recovered. Next, the dense matching is performed. The points from the previous step are used as a seeds for the dense matching algorithm. In each iteration, new points that satisfy certain condition are assumed to be good matches and are added to the final disparity map. The matching is performed with the continuity constraint, namely, the disparity of a new matched point cannot differ from the disparity of neighbouring points by a given threshold. The overview of this process is shown in Algorithm 12.

3.3.3 Inside Point Detection

The proposed dense matching method uses two constraints: (1) the edge points were matched correctly, (2) the disparity changes continuously for non-edge points. Since

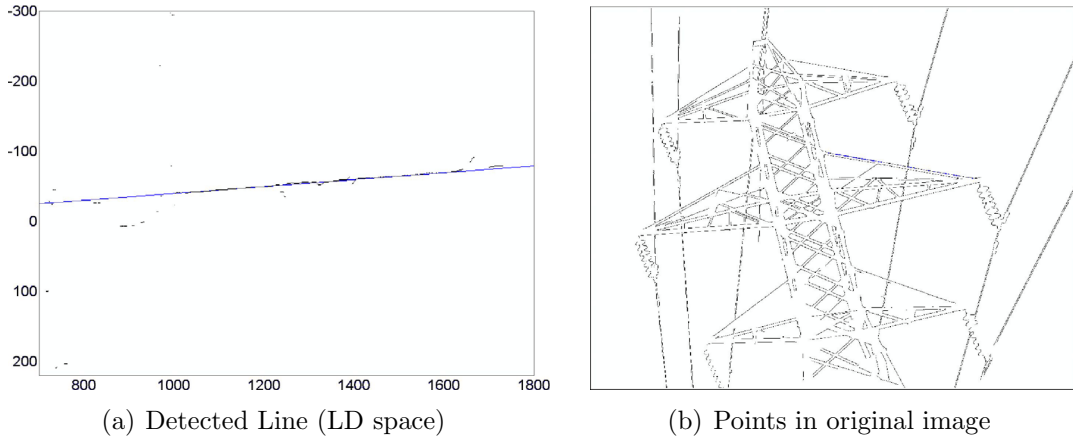


Figure 42: Detected line in LD space and points from original image

the edges usually surround plain areas, non-edge points are called the *inside points*. In this step, the inside points that are going to be added to the output disparity map are selected. In order to satisfy the second constraint, the disparities for new inside points cannot differ significantly from the disparities of already existing points. A set of new prospective inside matches is created. This set consists of only points that have at least one point in their 8-neighbourhood (see Definition 8) for which a disparity is known. An 8-neighbourhood of each new prospective match determines the boundaries of a new disparity, *i.e.*, if we denote by $N_8(x)$ the 8-neighbourhood for point x , then the new disparity $D(x)$ satisfies the inequality:

$$\min(N_8(x)) - th \leq D(x) \leq \max(N_8(x)) + th,$$

where th denotes the maximum difference threshold (for the experiments for this project th was set to 4). In other words, the new disparity cannot differ from the maximum/minimum disparity from the 8-neighbourhood by more than th .

Algorithm 12: Dense Matching

Input : edge point disparity map, pair of 2D images

Output: dense disparity map

```
1 while Not all points matched do
2   Find new inside points with known neighbourhoods for LR and RL;
3   /* in parallel */
4   begin
5     Match new inside points in LR;
6     Match new inside points in RL;
7   end
8   Merge LR and RL matches;
9   Remove outliers for LR and RL;
10 end
```

3.3.4 Matching

Once the set of new perspective matches with the boundaries of the disparities is selected, the actual matching begins. The matching is performed using the sum of squared differences, see (22). In order to achieve good accuracy the matching is performed with sub-pixel resolution. Before each SSD comparison the images are supersampled using the bicubic interpolation procedure. In order to speed up the processing, this step is performed in parallel on several machines. More precisely, the workload of all points to be matched (both “left to right” and “right to left” matches) is divided into available computers.

3.3.5 Merging Matches

Up to this point, all calculations were performed separately for “left to right” and “right to left” images. In this step, these matches are merged together. For two

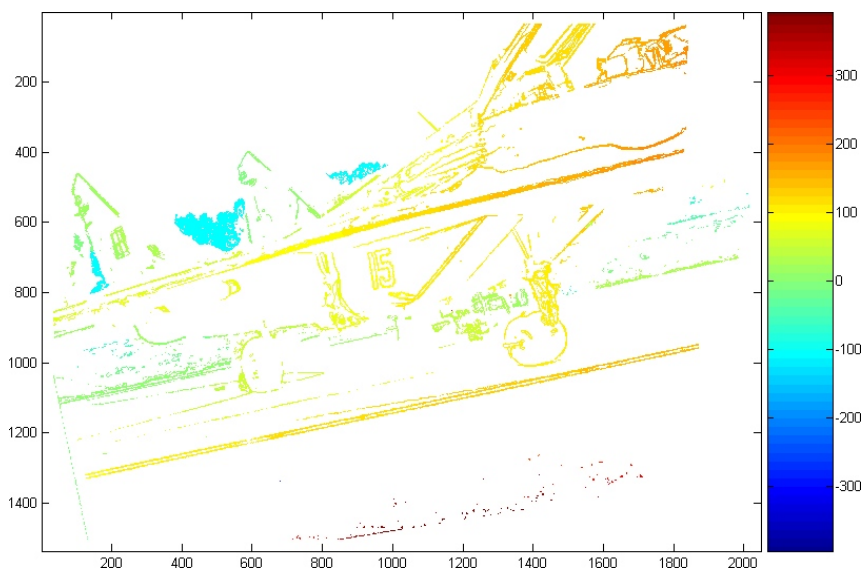


Figure 43: Sample disparity map for the "jet plane" scene.

corresponding points from two given images, matching a point from the left image with a point from the right image should result in the same disparity as when matching a point from the right image with a point from the left image. Denote by $M_{LR}(x_L)$ a match of a point x_L from the left image and $M_{RL}(x_R)$ a match of a point x_R from the right image, then for the correctly matched point x'_L the following equation holds:

$$M_{RL}(M_{LR}(x'_L)) = x'_L,$$

or equivalently if $D_{LR}(x_L)$ denotes the disparity of x_L and $D_{RL}(x_R)$ denotes the disparity of x_R then for the correctly matched point x'_L the following equation holds:

$$D_{LR}(x'_L) = -D_{RL}(M_{LR}(x'_L)).$$

Due to the fact that the observed scene is slightly different in both images the above equation may never be satisfied. Thus, a point x'_L is considered to be a correct match if it satisfies:

$$|D_{LR}(x'_L) + D_{RL}(M_{LR}(x'_L))| < 2.$$

The constraint from the above inequality is applied to validate the “left to right” and “right to left” matches.

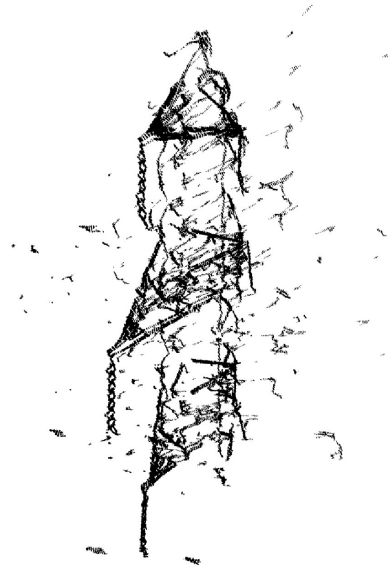
3.3.6 Removing Outliers

In the last step of dense matching, the outliers are removed. Outliers are defined to be a points for which the disparity differs from the disparity of its neighbouring points by more than a threshold D_{th} . The neighbourhood was chosen to be a circle of a radius with 4 pixels. The threshold changes for the three iterations of this step. The values of D_{th} are 8, 1 and 0.1. All points classified as outliers are removed from the final disparity map.

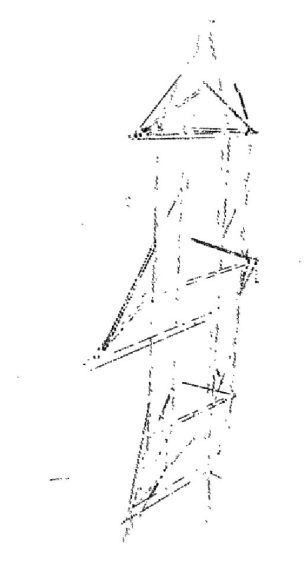
An iterative process of the preceding four steps is repeated until no more points can be added to the final disparity map.

3.3.7 Results

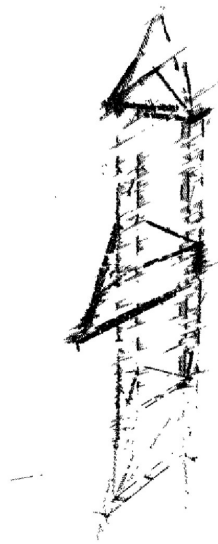
The 3D line extraction algorithm was applied to the 'tower' scene shown in Figure 40(a). The results of the matching without the 3D line extraction are shown in Figure 44(a). There is a lot of noise present in the extracted data. Noise means that the position of the extracted point does not correspond to the actual location of this point in the 3D space. The image in Figure 44(a) shows the tower from



(a) Without post-processing



(b) After 3D extraction



(c) After dense matching

Figure 44: 3D model of electric power tower

the direction approximately perpendicular to the principal ray of the first camera. Mismatch during the matching phase results in incorrect disparity. The disparity corresponds to the depth of the points. Therefore, error is the most influential in the direction parallel to the principal ray. As can be seen in Figure 44(a), the structure of the tower is spread along the direction parallel to the principal ray. There is quite a large number of points that, even though they belong to the tower, are located in the space outside the tower's structure.

The image in Figure 44(b) shows a 3D model of a tower that was reconstructed using 3D line extraction. The point of view was selected approximately perpendicular to the principal ray of the first view. In this case, there is no noticeable noise present in the data. The number of reconstructed points is much lower, but all of the reconstructed points are placed in the right position in 3D space.

Finally, the image in Figure 44(c) shows the same model as in the two previous cases, but after dense matching. There are many more points reconstructed from the tower structure compared to the previous step. In the dense matching step, the constraint about being part of a line in 3D space does not hold any more. As a result some points are slightly shifted away from the tower. Therefore, the dense matching step should be iterated only long enough to reconstruct as many points as are needed to recover the entire structure of a tower.

4 2D to 3D Conversion

This dissertation considers an approach to solving the problem of extracting 3D objects from 2D images, which is part of a growing literature on 2D to 3D conversion (see, *e.g.*, [4, 11, 13, 22, 44, 75, 141, 160, 184, 200]). The proposed solution to this problem uses 3D geometry rather than the fundamental matrix widely used in 2D to 3D conversion (see, *e.g.*, [33, 199]). The proposed approach to 2D to 3D conversion uses what is known as direct geometrical search (DirectGS), where distances between the reprojected rays for all image points are minimised. DirectGS focuses on the extraction of 3D information (depth information) from a two dimensional images. Figure 45 gives an overview of the steps leading to a 2D to 3D transformation. The approach to 2D to 3D conversion presented in this dissertation is an extension of the author’s earlier work on segment matching [10] and dense matching [11].

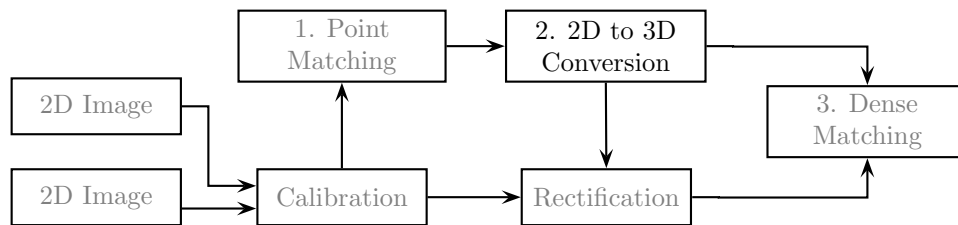


Figure 45: Overview of the 2D to 3D conversion system.

Briefly, the input information for step 2 in Figure 45 consists of pixel matches from a pair of sample images. The output from step 2 contains information about the external camera parameters, *i.e.*, the relative location and orientation of the two cameras in 3D space. This approach utilizes the pinhole model for a camera (see Section 2.3).

It is assumed that the cameras used to obtain 2D images have been calibrated, *i.e.* the internal camera parameters are known. The information used in this algorithm is the focal length of a camera. In other words, it is assumed that the focal length is known. It is worth mentioning that the assumption about knowing the focal length of the camera is not a limiting condition. Digital images taken with off-the-shelf cameras are too distorted to be used in 2D to 3D conversions without modifications. Therefore, the camera calibration process (see Section 2.5) must be performed and one of the by-products of the calibration process is the estimation of the focal length.

The goal of the proposed approach is to find the relative position of two cameras in a 3D world. This means that six parameters must be found, namely, three values for the translation along X, Y, Z coordinates, two for the relative difference of pan and tilt, and one for the rotation around the principal axis (formal definitions are given in Section 2.3). In fact, the recovery of the relative location of the cameras can be determined only up to a scale factor (see Condition 2). Instead of three parameters for the location, only two are used. The three coordinates X, Y and Z can be expressed using two parameters *azimuth* and *altitude*. Thus, the search is performed in a five dimensional space. By use of some specific properties of the problem formulation, this dissertation proposes even further reduction of the search space to four dimensions.

Notice that it is not important whether the images are taken by one camera that changes its position or by two cameras located in different positions. The only constraint is that the scene is unchanged between the two shots and that the two images are taken with the cameras having the same internal parameters (or both

cameras' internal parameters must be known). Keeping this in mind, the terms *one camera at different positions* and *two cameras at different positions* are used interchangeably. Following the commonly used practise in computer vision literature [61], the position and orientation of a camera is termed the *camera pose*.

DirectGS is based on the minimisation of the reprojection errors. Once all the parameters for both cameras are fixed, one is able to reproject the rays connecting the principal point and each pixel into 3D space. If the cameras were placed at the positions from where the images were originally taken, each pair of rays would cross in one point that is the actual point in the 3D space represented in both images by a given pixels. If on the other hand, the cameras are placed at some other locations, these rays do not cross. In such a case, it is possible to find the smallest distance between these rays. The sum of squared distances for each known pair of points provides the basis for a cost function. The main contribution of this section is twofold. First, a definition of an optimal search space for solving the 2D to 3D conversion problem is given. Second, an efficient algorithm that minimises the reprojection error is proposed.

4.1 Cost Function

This section contains the derivation of the cost function for estimation of positions of the camera views and the locations of points belonging to an object being reconstructed in 3D space without use of the fundamental matrix. The correct relative cameras' positions are found by minimising the cost function over the search space defined in this section. The cost function is the sum of squared shortest distances

between two rays reprojected for corresponding pixels into 3D space. In Section 4.2, an optimal algorithm for searching for this minimum is presented.

4.1.1 Basic Definitions

Notation

- The notation \overrightarrow{AB} , where A and B are points in the \mathbb{R}^n space, denotes a directed vector in \mathbb{R}^n space ($n = 2$ or $n = 3$). The beginning of the vector is at the point A and the end at the point B .
- The double indexing in the subscripts denotes the camera number k (where $k = 1$ or $k = 2$) and the point match pair i , where $0 < i < N$. For example, the point p_{ki} denotes the i -th point from the k -th camera. When there is no need to distinguish between cameras, the first index is omitted and the remaining index denotes a point match pair.

Customized Pinhole Camera Model

In this dissertation, a slightly different version of the pinhole camera model is used (*e.g.*, see Section 2.3). Instead of placing the image plane on the other side of the perspective centre, it is placed on the same side as the imaging object, as shown in Figure 46. Moving the image plane between the object and the perspective centre facilitates drawing rays starting at the perspective centre and passing through the image points. In addition, the proposed camera setup includes the following structure.

Table 12: Symbols used in 2D to 3D conversion algorithm.

Symbol	Description
P_i	i -th point in the 3D space
p_i	i -th point in the 2D image plane
p'_i	rotated point p_i
\bullet_{ki}	$k = 1, 2; i = 1, \dots, n; i$ -th point from k -th image
N	number of point matches
f	focal length of both cameras
$[v_x \ v_y \ v_z]^T$	point or vector in the 3D space
L_k	lens centre of the k -th camera
\mathcal{P}	principal point
θ	azimuth
ϕ	altitude
γ	rotation around the principal ray
R	distance from the origin to given point
ϑ	pan
φ	tilt
S_{ki}	parametrization of the i -th ray from the k -th view
τ_{ki}	parametrization variable for line S_{ki}
$\hat{\tau}_{ki}$	parametrization for the closest approach point A_{ki}
D_i	cost function for the i -th rays
$R_\alpha, R_\beta, R_\gamma$	rotation matrices
T_{ki}	$T_{ki} = p'_{ki} - L'_k$ directional vector for projecting ray S_{ki}
$V_1 \times V_2$	cross product between vectors V_1 and V_2
U_i	$U_i = T_{1i} \times T_{2i}$ common perpendicular vector to projecting rays S_{1i} and S_{2i}
W_i	$W_i = S_{1i}(t_{1i}) - S_{2i}(t_{2i})$ vector from point on line S_{2i} to point on line S_{1i}
A_{ki}	closest approach point for S_{ki} , A_{1k} is the closest point from S_{1i} to the line S_{2i}
\mathbb{S}	search space for the DirectGS algorithm
M	set of points from one image

- Image plane is perpendicular to the principal ray,
- origin of the image plane coordinates is set at the principal point,
- origin of the world coordinate system is set at the lens centre,
- world Z axis is parallel to the principal ray, and
- world X and Y axes are parallel to the x and y image plane coordinates.

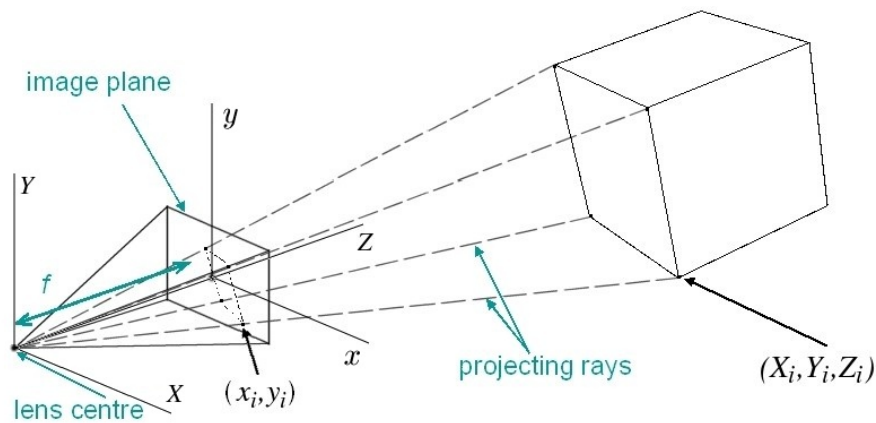


Figure 46: Pinhole Camera Model

Determining image coordinates from 3D coordinates is accomplished using:

$$x_i = \frac{fX_i}{f + Z_i}, \quad y_i = \frac{fY_i}{f + Z_i}.$$

The above formula do not take into account translations of the lens centre and rotations of the vector \vec{f} that is the vector from the lens centre to the principal point. Translations and rotations are explained in Section 4.1.1.

Reprojecting Rays Into 3D Space

Having defined the camera model, the next step is to reproject rays back into 3D space. For each given point match, two rays are reprojected. One ray is reprojected for each camera.

Let N denote the number of point matches. The image coordinates of the i -th matched point are given by (x_i, y_i) . The image plane is placed at $Z = -f$, where f is the focal length of the camera. The location of the lens centre is $[0\ 0\ 0]^T$ and the location of the i -th point²⁴ in the 3D space is $[x_i\ y_i\ -f]^T$.

$$L = [0\ 0\ 0]^T, \quad p_i = [x_i\ y_i\ -f]^T.$$

The i -th ray is parameterized by the function:

$$\text{Ray}_i(\nu) = [0\ 0\ 0]^T + \nu \overrightarrow{Lp_i} = [\nu x_i\ \nu y_i\ -\nu f]^T,$$

where the ν parameter is a positive real number. In general, each ray can be parameterized as the i -th ray directional vector $\overrightarrow{RD_i}$ multiplied by the parametrization coefficient ν :

$$\text{Ray}_i(\nu) = \nu \overrightarrow{RD_i} = [\nu rd_x\ \nu rd_y\ \nu rd_z]^T.$$

In this case, $rd_x = x_i$, $rd_y = y_i$ and $rd_z = -f$.

The set of N rays for each camera is called a *pencil of rays*. Definition 18 introduces a very useful notion of a spread of points.

²⁴The location of the i -th point in the image plane

Definition 18. The **diameter of a pencil of rays** at the distance L is the diameter of the smallest sphere in which all points belonging to each ray at the distance L from the lens centre can be enclosed.

Translation and rotation of the cameras

Up to now, the lens centre was located at the origin of the coordinate system and the principal ray was parallel to the Z axis. In order to find the relative locations and orientations of cameras, it is necessary to translate the cameras and change the cameras' directions. Notice that a camera's position is determined by its lens centre. On the other hand, the camera's orientation is determined by the vector \overrightarrow{LP} , where L is the lens centre and \mathcal{P} is the principal point. These two parameters, *i.e.*, the camera's location and orientation, are independent and can be modified independently.

First, consider the location of the camera. In order to facilitate the calculations (see the Condition 2), the position of the camera is not given in Cartesian coordinates, but in spherical coordinates [180, 188]. In order to cover the entire \mathbb{R}^3 space, three parameters are needed to uniquely describe the position of a point. These parameters are the azimuth θ , the altitude ϕ and the distance R from the point to the origin. The azimuth is an angle between a projection on the XZ plane of a line connecting given point and the origin of the coordinate system and the Z axis measured in such a way that a point on the positive Z axis has the azimuth $\theta = 0$ and a point on the positive X axis has the azimuth $\theta = \frac{\pi}{2}$. In other words, the azimuth increases when rotating counter-clockwise around the Z axis (this is also called the

right-handed coordinate system [82, 180]). The range of values for the azimuth is therefore $[-\pi, \pi)$ (closing the left side of the range interval was chosen arbitrary). The altitude is an angle between a line connecting given point and the origin of the coordinate system and the projection of this line on the XZ plane measured such that for positive Y values the altitude is negative, for negative Y values the altitude is positive and for $Y = 0$ the altitude is zero (see Figure 47). The range for the altitude parameter is $(-\frac{\pi}{2}, \frac{\pi}{2})$.

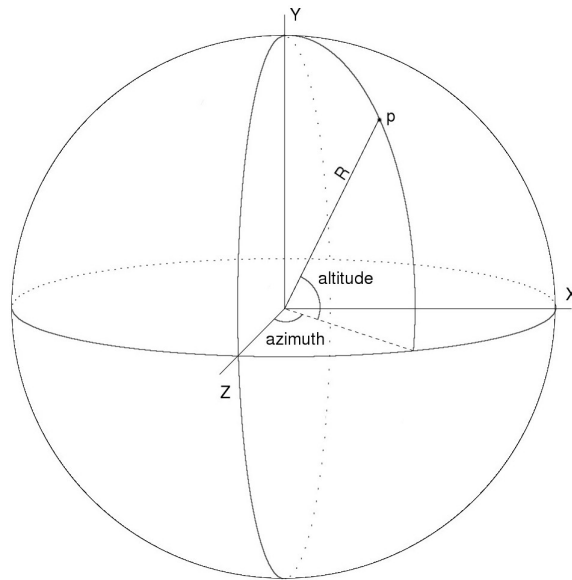


Figure 47: Point $p = (x, y, z)$ represented in spherical coordinate system (θ, ϕ, R) .

The conversion from Cartesian to spherical coordinates and vice versa is carried out using:

$$\theta = \arctan\left(\frac{x}{z}\right) \quad \phi = \arctan\left(\frac{-y}{\sqrt{x^2 + z^2}}\right) \quad R = \sqrt{x^2 + y^2 + z^2},$$

and

$$x = R \sin \theta \cos \phi \quad y = -R \sin \phi \quad z = R \cos \theta \cos \phi.$$

In this dissertation, the terms *azimuth* and *altitude* are used to describe the location of a camera in 3D space (together with the distance R shown in Figure 47). Specification of the orientation of a camera can be accomplished using only two parameters, pan and tilt. More precisely, pan ϑ is the rotation of the camera around the Z axis. A positive pan turns in a counter-clockwise direction. Tilt φ denotes the rotation around a new X axis (after rotation around Z axis). Again, a positive tilt corresponds to a counter-clockwise direction. One last parameter (rotation around the principal ray denoted by γ) is needed to uniquely describe the orientation of a camera.

Rotations In 3D Space

All rotations shown in this dissertation are performed using *Euler Angles* [82]. An Euler Angle results from the rotation performed around one of the coordinate axes. The theorem [82] by Leonard Euler (1707-1783) states that any orthonormal coordinate frame can be related to any other orthonormal coordinate frame by at most of three rotations. This means that any rotation in 3D space can be performed by at most three rotations using Euler Angles. This sequence of rotations is called the *Euler Angle Sequence*. An alternative method of performing rotations in 3D space can be carried out using quaternions [82]. A unit quaternion is the preferred method of rotation by many researchers [68]. Since calculations for this dissertation did not suffer from limitations resulting from Euler Angles, all rotations were performed

using Euler Angles.

The rotation around a selected axis in the 3D space is performed using rotation matrices:

$$R_\beta = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \quad R_\alpha = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & \sin \alpha \\ 0 & -\sin \alpha & \cos \alpha \end{bmatrix}. \quad (23)$$

Matrix R_β is used to rotate around the Y axis by an angle β and matrix R_α is used to rotate around the X axis by an angle α .

The composition of two rotations results in one matrix

$$R_{\beta\alpha} = R_\beta R_\alpha = \begin{bmatrix} \cos \beta & -\sin \alpha \sin \beta & \cos \alpha \sin \beta \\ 0 & \cos \alpha & \sin \alpha \\ -\sin \beta & -\sin \alpha \cos \beta & \cos \alpha \cos \beta \end{bmatrix}, \quad (24)$$

that yields a rotation around two specified axes. Notice that matrix multiplication is not commutative. The rotation is performed first around X axis (by α) and second around new Y axis (by β).

In order to rotate a point $v = [v_x \ v_y \ v_z]^T$, this point must be multiplied on its left-hand side by the rotation matrix.

$$v' = R_{\beta\alpha} v = \begin{bmatrix} \cos \beta & -\sin \alpha \sin \beta & \cos \alpha \sin \beta \\ 0 & \cos \alpha & \sin \alpha \\ -\sin \beta & -\sin \alpha \cos \beta & \cos \alpha \cos \beta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} =$$

$$\begin{bmatrix} \cos \beta v_x - \sin \alpha \sin \beta v_y + \cos \alpha \sin \beta v_z \\ \cos \alpha v_y + \sin \alpha v_z \\ -\sin \beta v_x - \sin \alpha \cos \beta v_y + \cos \alpha \cos \beta v_z \end{bmatrix}.$$

The rotation matrix from (24) is used to change the location and orientation of the camera. In the former case, the pair of parameters (β, α) is replaced by the pair (θ, ϕ) and in the latter case by (ϑ, φ) . The last operation to consider is rotation around the principal ray. This rotation around the Z axis is given by:

$$R_\gamma = \begin{bmatrix} \cos \gamma & \sin \gamma & 0 \\ -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

To model rotation of a camera around all 3 axes, the pan and tilt matrix $R_{\vartheta\varphi}$ is multiplied by the matrix R_γ to obtain a matrix $R_{\vartheta\varphi\gamma}$:

$$R_{\vartheta\varphi\gamma} = R_{\vartheta\varphi}R_\gamma. \quad (25)$$

4.1.2 Cost Function Derivation

The input for a cost function is the full information about the positions of a pair of cameras. This information is used to reconstruct the projecting rays for all pairs of points from a pair of camera views. Then, the squared distances between corresponding projecting rays are found and summed. The procedure for rotating points, determining the projecting rays and finding the distances between them is discussed next. The list of all symbols used in the derivation of the proposed cost function is

shown in Table 12.

Denote by p_{ki} the i -th point from the k -th camera. Since the location and the orientation of the first camera is fixed, the 3D coordinates of the i -th point for the first camera are:

$$p'_{1i} = [x'_{1i} \ y'_{1i} \ z'_{1i}]^T = [x_{1i} \ y_{1i} \ -f]^T. \quad (26)$$

The lens centre for the first camera has the coordinates $[0 \ 0 \ 0]^T$. For the second camera, the point coordinates have to be rotated and translated to match the required camera location and orientation²⁵. First, the point coordinates are rotated to reflect the camera's pan and tilt.

$$p'_{2i} = [x'_{2i} \ y'_{2i} \ z'_{2i}]^T = R_{\theta\phi}(R_{\vartheta\varphi\gamma}[x_{2i} \ y_{2i} \ -f]^T + [0 \ 0 \ R]^T). \quad (27)$$

Then, the camera is translated to the surface of the sphere of radius R and rotated to its final destination. The lens centre for the second camera is determined by rotating the point $[0 \ 0 \ R]^T$ by azimuth and altitude angles.

$$L'_2 = R_{\theta\phi}[0 \ 0 \ R]^T. \quad (28)$$

Next, the projecting rays are determined. Since the i -th projecting ray for the k -th image must go through the k -th lens centre and the point p'_{ki} , the direction of the ray is equal to the difference of the points through which it has to pass.

$$T_{ki} = [t_{x,ki} \ t_{y,ki} \ t_{z,ki}]^T = \overrightarrow{p'_{ki}L'_k} = p'_{ki} - L'_k. \quad (29)$$

²⁵For justification of this approach see Conditions 1 and 2.

The line can be parametrized as:

$$S_{ki}(\tau_{ki}) = p'_{ki} + \tau_{ki}T_{ki} = [x'_{ki} \ y'_{ki} \ z'_{ki}]^T + \tau_{ki} \begin{bmatrix} t_{x,ki} \\ t_{y,ki} \\ t_{z,ki} \end{bmatrix}, \quad (30)$$

where $\tau_{ki} > 0$ is the parametrization variable.

Next, we need to find the distance between the two projecting rays S_{1i} and S_{2i} [208]. First, the vector perpendicular to both lines is found. It is the result of the cross product between the vectors T_{1i} and T_{2i} [180]. The formula for calculating the common perpendicular vector U_i is:

$$U_i = \begin{bmatrix} u_{x,i} \\ u_{y,i} \\ u_{z,i} \end{bmatrix} = T_{1i} \times T_{2i} = \begin{bmatrix} t_{y,1i}t_{z,2i} - t_{z,1i}t_{y,2i} \\ t_{z,1i}t_{x,2i} - t_{x,1i}t_{z,2i} \\ t_{x,1i}t_{y,2i} - t_{y,1i}t_{x,2i} \end{bmatrix}.$$

The shortest line joining the two projecting rays is parallel²⁶ to U_i . To find the shortest distance, a vector W_i is used to join a point corresponding to τ_{1i} from S_{1i} to a point corresponding to τ_{2i} from S_{2i} .

$$W_i = \begin{bmatrix} w_{x,i} \\ w_{y,i} \\ w_{z,i} \end{bmatrix} = S_{1i}(\tau_{1i}) - S_{2i}(\tau_{2i}) = \begin{bmatrix} x'_{1i} + \tau_{1i}t_{x,1i} - (x'_{2i} + \tau_{2i}t_{x,2i}) \\ y'_{1i} + \tau_{1i}t_{y,1i} - (y'_{2i} + \tau_{2i}t_{y,2i}) \\ z'_{1i} + \tau_{1i}t_{z,1i} - (z'_{2i} + \tau_{2i}t_{z,2i}) \end{bmatrix}. \quad (31)$$

²⁶ U_i is just a vector defining a direction of the shortest line between the two projecting rays and may have different length than the distance between these lines.

In effect, the shortest vector W_i must be parallel to U_i , *i.e.*,

$$\frac{w_{x,i}}{u_{x,i}} = \frac{w_{y,i}}{u_{y,i}} = \frac{w_{z,i}}{u_{z,i}}. \quad (32)$$

The above equation can be decoupled into two separate equations:

$$w_{x,i} \cdot u_{y,i} = w_{y,i} \cdot u_{x,i}, \quad w_{y,i} \cdot u_{z,i} = w_{z,i} \cdot u_{y,i}.$$

After substituting values of components of W_i from (31) and solving for τ_{1i} and τ_{2i} , one gets the closest approach points (parametrized by $\hat{\tau}_{1i}$ and $\hat{\tau}_{2i}$).

$$\begin{aligned} \hat{\tau}_{1i} = & [(u_{y,i}t_{z,2i} - u_{z,i}t_{y,2i})(x'_{1i} - x'_{2i}) + (u_{z,i}t_{x,2i} - u_{x,i}t_{z,2i})(y'_{1i} - y'_{2i}) + \\ & (u_{x,i}t_{y,2i} - u_{y,i}t_{x,2i})(z'_{1i} - z'_{2i})] / [t_{x,1i}(u_{z,i}t_{y,2i} - t_{z,2i}u_{y,i}) - \\ & t_{y,1i}(u_{z,i}t_{x,2i} - t_{z,2i}u_{x,i}) + t_{z,1i}(u_{y,i}t_{x,2i} - t_{y,2i}u_{x,i})] \end{aligned} \quad (33)$$

$$\begin{aligned} \hat{\tau}_{2i} = & [(y'_{2i} - y'_{1i})u_{x,i} - (x'_{2i} - x'_{1i})u_{y,i} - \\ & \hat{\tau}_{1i}(t_{y,1i}u_{x,i} - t_{x,1i}u_{y,i})] / (t_{x,2i}u_{y,i} - t_{y,2i}u_{x,i}). \end{aligned}$$

The *closest approach* points $A_{ki} = S_{ki}(\hat{\tau}_{ki})$ give the positions on both lines that are the closest points with respect to the other line [26]. Finally, the distance D_i between two projecting rays is the distance between points A_{ki} .

$$D_i = \|A_{1i} - A_{2i}\|. \quad (34)$$

where $\|\bullet\|$ denotes the norm operator, namely $\|[v_x \ v_y \ v_z]\| = \sqrt{(v_x^2 + v_y^2 + v_z^2)}$. The cost function is a sum of squared distances for each point match.

4.1.3 Search Space

As mentioned before, the main idea in the proposed approach to 2D to 3D conversion is to find the relative location of both cameras such that the squared distances between corresponding reprojected rays are minimal. The fact that the locations of the cameras can be determined up to translation and a scale factor introduces redundancy in the solution space. More precisely, the six dimensional search space (namely, three spatial coordinates, and three rotations) contains an infinite number of points yielding the same solution. To construct an algorithm that searches through distinct solutions only, two conditions are introduced. First, the assertion that a solution can be found up to translation and a scale factor, a formal definition of equivalent solutions is given.

Definition 19. Two solutions to the reprojecting rays minimal distance problem are equivalent, if the only difference between them is described by one or more of the following statements.

1. two solutions differ by a scale factor,
2. one solution is a translated version of the second solution,
3. one solution is a rotated version of the second solution.

Next, the condition that removes the redundancy 2 and 3 from Definition 19 is given.

theorem guarantees that such a matrix exists.

In addition, when applied to both cameras, the transformation H does not change the relative translation and orientation between cameras (since translations and rotations do not change the size and shape of transformed figures). This proves that all solutions that differ by a translation and rotation of both cameras are equivalent. \square

Theorem 3 and the above proof show that Condition 1 does not limit the search space, since any point in the $\mathbb{R}^3 \times \mathbb{R}^3$ space could be selected. Fixing the position of the first camera allows for reducing the number of possible cameras' configurations, by omitting all the similar cases that are different only by a translation and rotation of both cameras.

The second condition removes redundancy 1 from Definition 19.

Condition 2. The distance between the first and the second camera is fixed.

Since it is assumed that Condition 1 holds, the distance between cameras is equal to the distance between the second camera and the origin of the 3D space. For the experiments, the distance of the second camera to the origin was arbitrarily set to three times the focal length of the first camera. The set up of both cameras is shown in Figure 48. Theorem 4 below as well as its proof provide a basis for validating Condition 2.

Theorem 4. The choice of any particular distance between the first and second camera does not limit the search space.

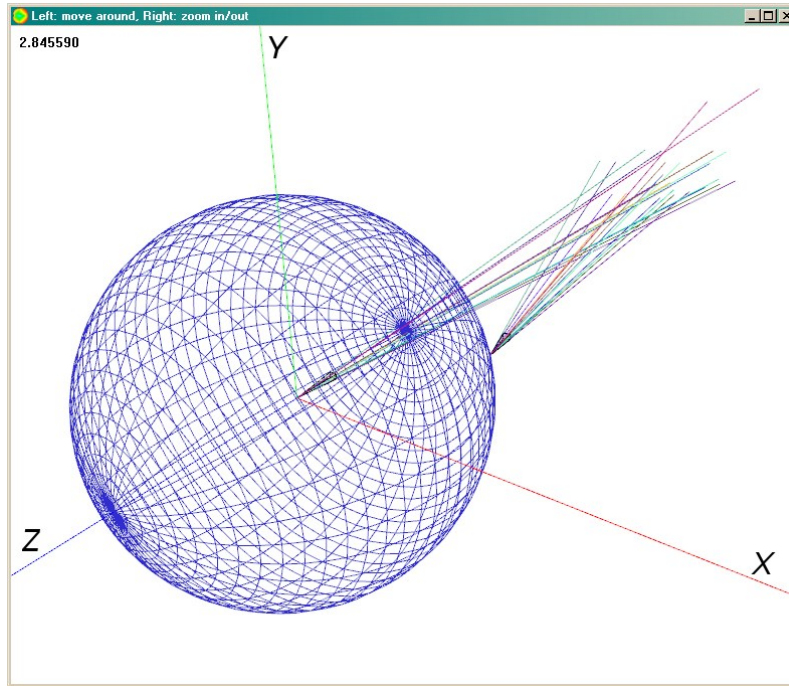


Figure 48: First camera view (at the origin of the coordinate system) and the second camera view on the sphere of radius R

Proof. To prove Theorem 4, it has to be shown that all the solutions that differ only by a distance between the two cameras are equivalent. To prove that, it is enough to show that the distance between each reconstructed point and the origin is proportional to the distance between the second camera and the origin of the coordinate system $|L'_2|$ (it is assumed that Condition 1 holds).

The proof is based on the derivation of the cost function presented in Section 4.1.2 and on the fact that the superposition of two linear functions is a linear function. It will be shown that the solution for the i -th pair of points is given in the form:

$$Sol_i = (A_{1i} + A_{2i})/2 = \alpha R, \quad (35)$$

where A_{ki} is the closest approach point for the k -th view and i -th pair of points (see Section 4.1.2 for more details), and $\alpha \in \mathbb{R}^3$ is a constant, in a sense that it does not depend on R .

From (27), the location of the i -th point from the second view is equal to:

$$p'_{2i} = R_{\theta\phi}(R_{\partial\varphi\gamma}[x_{2i} \ y_{2i} - f]^T + [0 \ 0 \ R]^T) = R_{\theta\phi}R_{\partial\varphi\gamma}[x_{2i} \ y_{2i} - f]^T + R_{\theta\phi}[0 \ 0 \ R]^T. \quad (36)$$

In order to simplify the calculations, the matrix product $R_{\theta\phi}R_{\partial\varphi\gamma}$ is defined by nine parameters r_{mn} , where $m, n \in \{1, 2, 3\}$. Thus,

$$p'_{2i} = \begin{bmatrix} x'_{2i} \\ y'_{2i} \\ z'_{2i} \end{bmatrix} = \begin{bmatrix} r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f + R_{13,\theta\phi}R \\ r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f + R_{23,\theta\phi}R \\ r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f + R_{13,\theta\phi}R \end{bmatrix}, \quad (37)$$

and the p'_{1i} is given in (26).

The direction of the k -th ray is given by (29). Since there is no translation nor rotation for the first camera, the direction of the first ray is equal to

$$T_{1i} = [t_{x,1i} \ t_{y,1i} \ t_{z,1i}]^T = [x_{1i} \ y_{1i} \ -f]^T, \quad (38)$$

and, after substituting (36) in (29), the direction of the second ray is given by:

$$T_{2i} = \begin{bmatrix} t_{x,2i} \\ t_{y,2i} \\ t_{z,2i} \end{bmatrix} = \begin{bmatrix} r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f \\ r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f \\ r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f \end{bmatrix}. \quad (39)$$

After substituting (26), (37), (38) and (39) in (33), the equation (33) becomes

$$\begin{aligned}
\hat{\tau}_{1i} = & [(u_{y,i}(r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f) - u_{z,i}(r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f)) \\
& (x'_{1i} - (r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f + R_{13,\theta\phi}R)) + (u_{z,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f) - \\
& u_{x,i}(r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f))(y'_{1i} - (r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f + R_{23,\theta\phi}R)) + \\
& (u_{x,i}(r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f) - u_{y,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f)) \\
& (z'_{1i} - (r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f + R_{13,\theta\phi}R))]/[x_{1i}(u_{z,i}(r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f) - \\
& (r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f)u_{y,i}) - y_{1i}(u_{z,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f) - \\
& (r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f)u_{x,i}) - f(u_{y,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f) - \\
& (r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f)u_{x,i})]. \tag{40}
\end{aligned}$$

Denote by $N(\hat{\tau}_{1i})$ the numerator of $\hat{\tau}_{1i}$. Then, (41) shows the numerator of $\hat{\tau}_{1i}$ after all terms containing R were grouped together.

$$\begin{aligned}
N(\hat{\tau}_{1i}) = & [-(u_{x,i}(r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f) - u_{y,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f))R_{13,\theta\phi} - \\
& (u_{z,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f) - u_{x,i}(r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f))R_{23,\theta\phi} - \\
& (u_{y,i}(r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f) - u_{z,i}(r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f))R_{33,\theta\phi}]R + \\
& (u_{x,i}(r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f) - u_{y,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f))
\end{aligned}$$

$$\begin{aligned}
& (-f - r_{31}x_{2i} - r_{32}y_{2i} + r_{33}f) + (u_{z,i}(r_{11}x_{2i} + r_{12}y_{2i} - r_{13}f) - \\
& u_{x,i}(r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f))(y_{1i} - r_{21}x_{2i} - r_{22}y_{2i} + r_{23}f) + \\
& (u_{y,i}(r_{31}x_{2i} + r_{32}y_{2i} - r_{33}f) - u_{z,i}(r_{21}x_{2i} + r_{22}y_{2i} - r_{23}f)) \\
& (x_{1i} - r_{11}x_{2i} - r_{12}y_{2i} + r_{13}f). \tag{41}
\end{aligned}$$

The above equation can be rewritten in the form:

$$N(\hat{\tau}_{1i}) = C_1R + C_2,$$

where C_1 and C_2 are constants (in a sense that they do not depend on R). Similarly, (40) can be rewritten in the form:

$$\hat{\tau}_{1i} = \frac{C_1R + C_2}{C_3},$$

where C_3 is another constant and denotes the denominator of $\hat{\tau}_{1i}$ in 40. After rearranging all the terms in C_2 and C_3 , one can observe that $C_2 = -C_3$ and after substituting by $\frac{C_1}{C_3}$ a new constant C_4 , the above equation has the form:

$$\hat{\tau}_{1i} = C_4R - 1. \tag{42}$$

Moreover, one can observe that the formula for the $\hat{\tau}_{2i}$ resembles the above equation:

$$\hat{\tau}_{2i} = C_5R - 1.$$

From (38) and (26), it is apparent that

$$p'_{1i} = T_{1i}. \quad (43)$$

Let $\mathcal{C}_6 \in \mathbb{R}^3$ be a 3D vector that does not depend on R . From (37) and (39), we obtain

$$p'_{2i} = T_{2i} + \begin{bmatrix} R_{13,\theta\phi}R \\ R_{23,\theta\phi}R \\ R_{13,\theta\phi}R \end{bmatrix} = T_{2i} + \mathcal{C}_6R,$$

Thus, after substituting (42) and (43) in (30) and setting \mathcal{C}_7 to be a 3D vector resulting from multiplying vector p'_{1i} by constant C_4 , one gets the formula for the location of the closest approach point for the first ray, namely

$$\begin{aligned} S_{1i}(\hat{\tau}_{1i}) &= p'_{1i} + \hat{\tau}_{1i}T_{1i} = p'_{1i} + (C_4R - 1)p'_{1i} = \\ & p'_{1i} + C_4Rp'_{1i} - p'_{1i} = C_4p'_{1i}R = \mathcal{C}_7R. \end{aligned} \quad (44)$$

Corresponding calculations for the second ray are given in:

$$\begin{aligned} S_{2i}(\hat{\tau}_{2i}) &= p'_{2i} + \hat{\tau}_{2i}T_{2i} = T'_{2i} + \mathcal{C}_6R + (C_5R - 1)T'_{2i} = \\ & T'_{2i} + \mathcal{C}_6R + C_5RT'_{2i} - 1T'_{2i} = (\mathcal{C}_6 + C_5T'_{2i})R = \mathcal{C}_8R, \end{aligned} \quad (45)$$

where \mathcal{C}_8 is a 3D vector that does not depend on R .

Finally, after substituting (44) and (45) in (35), one gets the formula for the

location of the i -th point in the 3D space,

$$Sol_i = (\mathcal{C}_7 R + \mathcal{C}_8 R)/2 = \frac{\mathcal{C}_7 + \mathcal{C}_8}{2} R.$$

This proves that $\alpha = \frac{\mathcal{C}_7 + \mathcal{C}_8}{2}$ (in (35)) is independent of R and the solution is proportional to the distance R between the cameras. Moreover, since \mathcal{C}_7 and \mathcal{C}_8 are 3D vectors, it has been shown that each coordinate of the i -th solution point depends on R . □

A significant result of applying Condition 2 and Condition 1 is the reduction of the search space. The first condition fixes the location and the orientation of the first camera. This leaves six more parameters that need to be used to describe the location and orientation of the second camera. The second condition decreases the search space from six parameters to five. Next, the formal definition of the search space is given.

Definition 20. A **search space** \mathbb{S} is a set of all relative locations and orientations of two cameras. The lens centre of the first camera is set at the origin of the coordinate system, *i.e.*, at point $[0 \ 0 \ 0]^T$. The orientation is also fixed and it is chosen such that the principal ray is parallel to the Z axis pointing into the negative values of the Z axis. The location of the second camera is fixed on the sphere of a radius R , centred at the origin of the coordinate system. The pan, tilt and rotation around the principal ray are unconstrained.

Table 13: List of all symbols used in the error minimisation algorithm

Symbol	Description
\mathbb{M}	set of point matches
\mathbb{I}	initial guess of the azimuth and altitude
θ	azimuth
ϕ	altitude
γ	rotation around the principal ray
R	the radius of the sphere
f	the focal length of both cameras
ϑ	pan
φ	tilt

4.2 Error minimisation

This section contains the description of a very efficient algorithm for finding the minimum of the cost function derived in Section 4.1 (see Algorithm 15). Table 13 contains all the symbols used in this section (in addition to symbols given in Table 12).

The input to Algorithm 15 is the set of point matches \mathbb{M} between two images, focal length f of both cameras, radius R of the sphere on which the second camera is located and an initial guess \mathbb{I} of the azimuth and altitude of the second camera. The initial guess does not have to be very close to the global minimum for the algorithm to converge. Figure 49 shows how the cost function depends on the azimuth and altitude of the second camera²⁷. Basically, choosing any initial point in the front hemisphere results in the proper convergence of the algorithm. Since the location of the blue path shown in Figure 49 depends on the locations of the cameras when the images were taken, the safest initial guess is close to the area where the rays from the

²⁷The cost function shown in Figure 49 was calculated for the 'tower' scene.

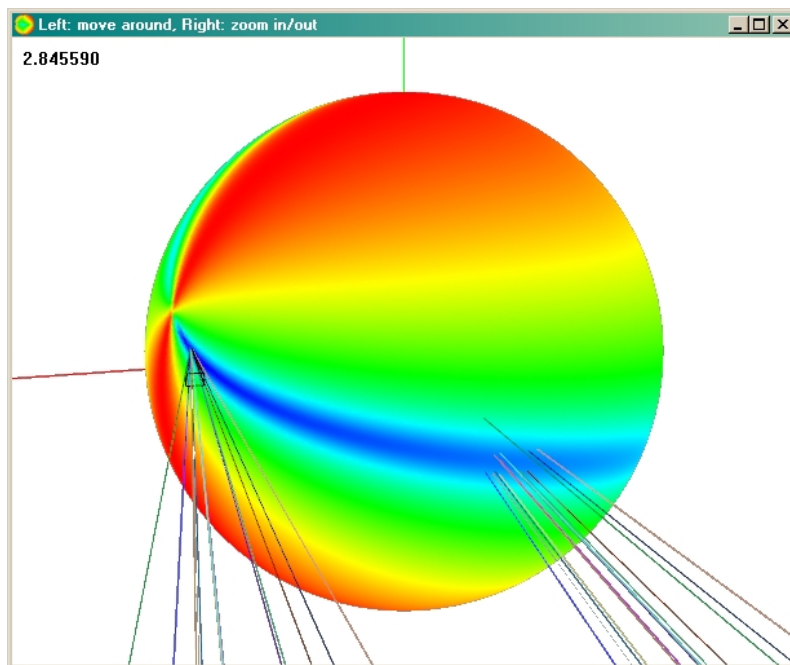


Figure 49: Cost function for azimuth and altitude.

first camera cross with the sphere on which the second camera is located. Since the first camera does not move, a good initial guess is at locations $[0 \ 0 \ R]^T$ and $[0 \ 0 \ -R]^T$. Intuitively, the choice for the location of the second camera depends on whether the second camera is closer or further away from the object being reconstructed. This information can be easily extracted from the images. If not, the algorithm must be run twice with both values and the solution with lower cost function is selected.

The output from Algorithm 15 is the location and orientation of the second camera such that the cost function for this configuration is minimal. Denote by \mathbb{C}_2 the location and orientation of the second camera. \mathbb{C}_2 is a five dimensional vector $\mathbb{C}_2 = [\theta \ \phi \ \gamma \ \vartheta \ \varphi]^T$, where θ and ϕ denote the azimuth and altitude of the second camera, γ its rotation around the principal ray and ϑ and φ its pan and tilt,

respectively. The second camera parameters $\hat{\mathbb{C}}_2$ corresponding to the minimum of the cost function constitute the output of the error minimisation algorithm:

$$\hat{\mathbb{C}}_2 = \underset{\mathbb{C}_2 \in \mathbb{S}}{\operatorname{argmin}} D(\mathbb{C}_2),$$

where $\hat{\mathbb{C}}_2$ denotes the camera's parameters returned by the algorithm and $D(\mathbb{C}_2)$ is the cost function defined in by

$$D(\mathbb{C}_2) = \sum_i D_i^2, \quad (46)$$

where i iterates through all image pairs from the set \mathbb{M} and D_i is the cost function for the i -th point match (see (34)).

The setup for Algorithm 15 starts with defining domains for each parameter. Pairs of attributes (θ, ϕ) and (ϑ, φ) denote the spherical coordinates of a point in 3D space. For the first pair, the sphere radius is equal to R , whereas for the second pair, the sphere radius is not defined, since the angles denote the pan and tilt of the camera. Based on the assumptions from Section 4.1.1, the ranges for these four parameters are given in Table 14. The rotation around the principal ray covers the entire range of 360 degrees and is directed in the clockwise direction.

Table 14: Domains of five parameters for location \mathbb{C}_2 for camera 2.

Parameter	Range of values
θ	$[-\pi, \pi)$
ϕ	$(-\frac{\pi}{2}, \frac{\pi}{2})$
γ	$[-\pi, \pi)$
ϑ	$[-\pi, \pi)$
φ	$(-\frac{\pi}{2}, \frac{\pi}{2})$

From Table 14, one can easily formulate an exhaustive algorithm for searching for the minimum value of the cost function over the space \mathbb{S} . As an aid to understanding the idea behind Algorithm 15, first consider an exhaustive search provided by Algorithm 13.

Algorithm 13: Exhaustive Algorithm for Cost Function minimisation

Input : \mathbb{M} set of point matches
 f focal length of both cameras
 R radius of the sphere on which the second camera is located
Output: Second camera parameters $\hat{\mathbb{C}}_2$

```

1 Set  $D_{min} = \infty$ 
2 forall  $(\tilde{\theta}, \tilde{\phi}) \in [-\pi, \pi) \times (-\frac{\pi}{2}, \frac{\pi}{2})$  do
3   forall  $\tilde{\gamma} \in [-\pi, \pi)$  do
4     forall  $(\tilde{\vartheta}, \tilde{\varphi}) \in [-\pi, \pi) \times (-\frac{\pi}{2}, \frac{\pi}{2})$  do
5       Set  $\tilde{\mathbb{C}}_2 = [\tilde{\theta} \ \tilde{\phi} \ \tilde{\gamma} \ \tilde{\vartheta} \ \tilde{\varphi}]^T$ 
6       if  $D(\tilde{\mathbb{C}}_2) < D_{min}$  then
7          $D_{min} = D(\tilde{\mathbb{C}}_2)$ 
8          $\hat{\mathbb{C}}_2 = \tilde{\mathbb{C}}_2$ 
9       end
10    end
11  end
12 end

```

The exhaustive Algorithm 13 consists of three nested loops. First, the altitude and azimuth of the second camera are fixed. Then the pan and tilt are set and finally the rotation γ . If in each iteration the cost function of the current configuration is minimal, it is saved as the prospective solution. This process is repeated until all combinations of these five parameters are searched. Notice that the order of the three loops is not important and has no influence on the performance and outcome of the algorithm.

The optimization of the exhaustive Algorithm 13 is based on two statements.

Statement 1. The cost function $D(\mathbb{C}_2)$ (as defined in (46)) is a continuous function of \mathbb{C}_2 on \mathbb{S} .

The proof of Statement 1 contains very laborious calculations. Therefore, only an outline of this proof is presented next.

A sum of finitely many continuous functions is a continuous function. Therefore, it is sufficient to show that each D_i is a continuous function on \mathbb{S} . Notice, D_i is a distance between two lines in 3D space. Since one of the lines (S_{1i} - the projecting ray from the first camera) does not change, the change of the distance D_i depends solely on the change of the location and orientation of the second projecting ray S_{2i} in 3D space. The continuity of D_i can be proved using the Cauchy continuity definition. In what follows, for any $\tilde{\mathbb{C}}_2$ from the search space one has to show that for any $\varepsilon > 0$ a $\delta > 0$ exists such that for each $\mathbb{C}_2 \in \mathbb{S}$ where $|\mathbb{C}_2 - \tilde{\mathbb{C}}_2| < \delta$, the inequality $|D(\mathbb{C}_2) - D(\tilde{\mathbb{C}}_2)| < \varepsilon$ holds. To control the 'epsilon' inequality²⁸, it is necessary to control very precisely the location and orientation of the ray reprojected from the second camera. Since the ray reprojected from the first camera is fixed at one position the change of distance between the two rays depends on a change of location of the second ray. Precise control of the location of the second ray is possible if we notice that the second ray is a line defined by two points p'_{2i} and L'_2 , where p'_{2i} is the i -th image point in 3D space and L'_2 is the lens centre of the second camera. From (27) and (28), we see that the location of points p'_{2i} and L'_2 is controlled by two rotation matrices $R_{\theta\phi}$ and $R_{\vartheta\varphi\gamma}$. Since sine and cosine functions

²⁸Controlling means setting the left side of the inequality as small as required.

are continuous, we can always find a δ such that $\theta < \delta$, $\phi < \delta$, $\vartheta < \delta$, $\varphi < \delta$ and $\gamma < \delta$, and all of these parameters yield such a small change in the location of points p'_{2i} and L'_2 that the 'epsilon' inequality holds true. This concludes the proof outline.

Before the second statement can be presented, a definition of a mean ray must be given.

Definition 21. *A mean ray is a ray reprojected into 3D space from the point that is the centre of mass for all points M in a given image. More precisely, a mean ray for the k -th camera is a line passing through the camera's lens centre L'_k and a point p'_{mean} , where*

$$p'_{mean} = R_{k,\theta\phi}R_{k,\vartheta\varphi\gamma}[\bar{x}_k \ \bar{y}_k \ -f]^T + L'_k,$$

and

$$\bar{x}_k = \underset{i}{mean}(x_{k,i}) \quad \text{and} \quad \bar{y}_k = \underset{i}{mean}(y_{k,i}).$$

The second statement allows one to relate the pan and tilt of the second camera to its azimuth and altitude.

Statement 2. If the cost function $D(\mathbb{C}_2)$ defined in (46) reaches its global minimum, then the mean rays for both cameras cross in one point.

Since the formal calculations for proving Statement 2 contain complex calculations, it is hard to see the essence of the proof. Hence, only the outline of the proof of Statement 2 is given here. To present the idea behind this proof, a non-limiting simplification is introduced. For very small changes of the attributes from the vector \mathbb{C}_2 , the change of the location of the closest approach points A_{ki} is also very small. Therefore, the location of the closest approach points for each camera can be

considered as a rigid cloud of points. Consider a case of two clouds of points in 3D space, where each point has assigned a matched point from the other cloud. The task is to translate and rotate one of the clouds such that the sum of the squared distances between corresponding points is minimal. It is not hard to show that the minimum of the sum of squared distances between corresponding points is reached when the two centres of mass of these clouds overlap. If they do not overlap, moving one cloud in the direction of the difference between the centres of mass decreases the SSD between the corresponding points. When considering very small differences, the situation with the cost function $D(\mathbb{C}_2)$ is analogous. This means that the centres of mass of both clouds of closest approach points overlap, and as a result two mean rays cross in one point. This concludes the proof outline for Statement 2.

Based on Statement 2, one can decrease the search space by making sure that the two mean rays always cross. Instead of using the pan and tilt parameters, one parameter can be used. This parameter is a distance from the point where the two mean rays cross to the lens centre of one of the cameras. Since the first camera does not change its location and orientation, it is natural to choose this parameter to be the distance between the point where the two mean rays cross and the lens centre of the first camera. This new parameter is called ψ . The function Ψ that calculates the values of pan and tilt for a given azimuth, altitude and distance ψ has the form

$$[\theta \ \phi \ \gamma \ \vartheta \ \varphi]^T \rightarrow [\theta \ \phi \ \gamma \ \Psi_\vartheta(\theta, \phi, \psi) \ \Psi_\varphi(\theta, \phi, \psi)]^T.$$

Because of Ψ the location and position of the second camera can be determined using only four parameters $[\theta \ \phi \ \gamma \ \psi]^T$. The formula for Ψ is not presented, instead

all the steps needed to determine pan and tilt from three given parameters are given next. It is assumed, that the two mean rays intersect in one point.

First, consider the point where the two mean rays cross. This point is denoted as J when considering the mean ray from the first camera and by Q when considering the mean ray from the second camera. The procedure described below starts from setting the location of the second camera to (θ, ϕ, γ) , and setting pan and tilt to zero. Then the location and orientation of the second camera is modified such that the points J and Q overlap.

- i) Find the centres of mass for points in both images:

$$\overline{M}_{kx} = \frac{\sum_{i=0}^N x_{ki}}{N}, \quad \overline{M}_{ky} = \frac{\sum_{i=0}^N y_{ki}}{N},$$

where $k \in \{1, 2\}$ denotes the camera number, i denotes the point number and N denotes the total number of matched points.

- ii) Determine the location of the point J_1 in 3D space.

$$J_1 = [\overline{M}_{1x} \ \overline{M}_{1y} \ -f], \quad J_2 = \psi \frac{J_1}{\|J_1\|},$$

where $\|\bullet\|$ denotes the length of a vector.

- iii) Determine the coordinates of the point J in the second camera coordinate system.

- Rotate the point J_2 by the second camera rotation matrix.

$$J_3 = R_{\theta\phi} J_2.$$

- Translate the point J_3 by a vector $[0 \ 0 \ R]^T$ (move the second camera such that the lens centre is not on the sphere of a radius R , but at the origin of the coordinate system).

$$J = J_3 - [0 \ 0 \ R]^T.$$

- iv) Determine the location of a point Q on the mean ray of the second camera.

$$Q_1 = [\overline{M}_{2x} \ \overline{M}_{2y} \ -f], \quad Q = \|J\| \frac{Q_1}{\|Q_1\|}.$$

- v) Determine the tilt angle needed to rotate point Q to have the same Y coordinate as the point J , see Figure 50. The subscript YZ and XZ denote the projection of given point into YZ and XZ plane, respectively. Notice, it is not enough to use the tilt angle $\angle J_{YZ} O Q_{YZ}$.

- Determine the angle α by which a point J has to be rotated around the axis Y to have a zero X coordinate.

$$\alpha = \angle J_{XZ} O T.$$

- Rotate points J and Q by an angle $-\alpha$ around the Y axis.

$$J' = R_{-\alpha} J, \quad Q' = R_{-\alpha} Q.$$

Notice that point J' lies on the XZ plane. Hence, $J' = J'_{XY}$.

- Set the tilt angle $\varphi = \angle Q'_{YZ} O J'_{YZ}$.

vi) Rotate point Q so that it has the same Y coordinate as point J .

$$Q_2 = R_\varphi Q.$$

vii) Determine the pan angle θ to be the angle between the projections of points J and Q_2 into the XZ plane:

$$\vartheta = \angle Q_{2,XZ} O J_{XZ}.$$

viii) Set $\Psi_\vartheta(\theta, \phi, \psi)$ to return ϑ and $\Psi_\varphi(\theta, \phi, \psi)$ to return φ .

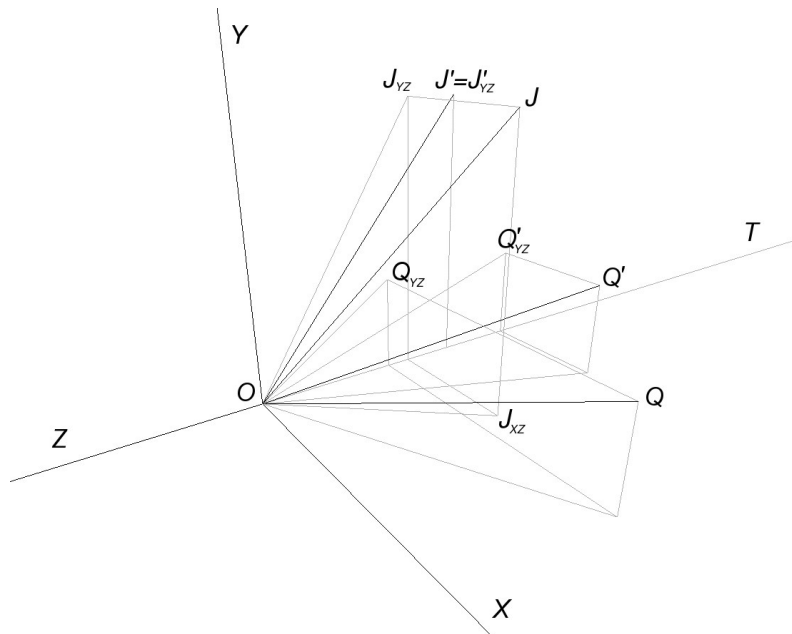


Figure 50: Illustration for the algorithm that determines the Ψ function.

The introduction of the function Ψ permits the reformulation of Algorithm 13

to obtain an exhaustive version with only four parameters (see Algorithm 14).

Algorithm 14: Exhaustive Cost Function minimisation with only 4 parameters

Input : \mathbb{M} set of point matches
 f focal length of both cameras
 R radius of the sphere on which the second camera is located
Output: Second camera parameters $\hat{\mathbb{C}}_2$

```

1 Set  $D_{min} = \infty$ 
2 forall  $(\tilde{\theta}, \tilde{\phi}) \in [-\pi, \pi) \times (-\frac{\pi}{2}, \frac{\pi}{2})$  do
3   forall  $\tilde{\gamma} \in [-\pi, \pi)$  do
4     forall  $\tilde{\psi} \in (0, \infty)$  do
5       Set  $\tilde{\mathbb{C}}_2 = [\tilde{\theta} \ \tilde{\phi} \ \tilde{\gamma} \ \Psi_\vartheta(\tilde{\theta}, \tilde{\phi}, \tilde{\psi}) \ \Psi_\varphi(\tilde{\theta}, \tilde{\phi}, \tilde{\psi})]^T$ 
6       if  $D(\tilde{\mathbb{C}}_2) < D_{min}$  then
7          $D_{min} = D(\tilde{\mathbb{C}}_2)$ 
8          $\hat{\mathbb{C}}_2 = \tilde{\mathbb{C}}_2$ 
9       end
10    end
11  end
12 end

```

Algorithm 14 is much more efficient than Algorithm 13, since it uses only four parameters to optimize the cost function. By introducing a necessity for an initial guess for the azimuth and altitude and by taking advantage of the property of the cost function given in (1), it is possible to convert Algorithm 14 into a non-exhaustive one. The optimization is performed separately for each of the 'for' loops shown in Algorithm 14.

To simplify the description of Algorithm 14, the explanation of several small subroutines is not given until the end of the algorithm.

First, consider the $\tilde{\theta}, \tilde{\phi}$ loop (see Algorithm 13). Using the fact that the cost

function is continuous, one can use a gradient-descent algorithm to search for the minimum. Having the initial guess \mathbb{I} of the azimuth and altitude, a gradient-descent search is guaranteed to find the local minimum around a given initial guess \mathbb{I} . As shown at the beginning of this section, if the initial guess is chosen properly, this minimum is also the global minimum.

A summary of the methods underlying Algorithm 14 is given next.

(DA) **Dense Algorithm.** In this algorithm, the domain for the given parameter is sampled into O equally spaced points. The point that yields the minimum value is returned.

(PFA) **Parabola Fitting Algorithm.** This algorithm uses three points with its corresponding cost function. The points have to be chosen such that the cost function is minimal for the middle point. The parabola is fit into those points and the coordinates for the point yielding the minimum is returned.

(LAA) **Local Adjustment Algorithm.** This is a 1-dimensional version of the gradient descent algorithm. That is, the cost function is calculated at a given point and both neighbours. If the cost function is minimal for one of the neighbours, this neighbour becomes the current point and the algorithm starts over. If not, the current point is returned. Usually, due to small changes in parameters' values, this algorithm exits after one iteration.

Having set the azimuth and altitude, there are only two parameters left to explain, namely, distance ψ and rotation around the principal ray γ . To solve for these parameters, the following heuristic is used. First, the parameter ψ is set to a big

value. The term *big* means that distance ψ is much greater than radius R (for example 50 times). This results in directions of the two mean rays being approximately the same, *i.e.*, parallel. In what follows, the closest approach points are quite far from the lens centres of the both cameras, which artificially increases the cost function. For such a value of ψ , an optimal rotation is found by a dense algorithm^(DA) followed by a parabola fitting^(PFA). Next, the ψ parameter is decreased and the rotation angle is adjusted based on the last optimal value^(LAA) followed by parabola fitting^(PFA). This procedure is continued until the cost function increases above some threshold over the minimum value. Notice that the cost function for the values of ψ approximately equal to the focal length of the first camera is artificially high again. In the case when $\psi \approx f$, the pencil of rays for the first camera has a small diameter and for the second camera has a big diameter (since the distance between the cameras R is much greater than the focal length). This results in a high cost function value and is not correlated with the configuration of points in both images. Since the cost function takes on high values for the extreme ends of the ψ parameter, the search is terminated after the cost function increases above some threshold above the minimum.

Cost function minimisation is given in Algorithm 15. A graphical representation of the proposed algorithm is given in Figure 51. The parameter K denotes the number of samples for the ψ parameter and the parameter th denotes the threshold above which the search for the ψ parameter is terminated. In each call of the CostFn subroutine, the function Ψ is evaluated. As a result the proposed algorithm decouples the search from the three loops given in Algorithm 14.

Algorithm 15: Cost Function Minimization

Input : \mathbb{M} the set of point matches
 f the focal length of both cameras
 R the radius of the second camera sphere
 \mathbb{I} initial values of azimuth θ and altitude ϕ
Output: Second camera parameters $\hat{\mathbb{C}}_2$

```
1 Set the  $(\tilde{\theta}, \tilde{\phi})$  parameters to those from  $\mathbb{I}$ , GlobalMinCF= $\infty$ 
2 Using gradient descent method minimize the cost function for  $(\tilde{\theta}, \tilde{\phi})$ 
3 begin
4   Set  $\tilde{\psi} = Kf$ , LocalMinCF= $\infty$ 
5   for  $\hat{\gamma} = -\pi$ ; step  $\frac{2\pi}{O}$ ; to  $\pi$  do
6     if CostFn( $\hat{\gamma}$ ) < LocalMinCF then
7       | LocalMinCF = CostFn( $\hat{\gamma}$ );  $\tilde{\gamma} = \hat{\gamma}$ 
8     end
9   end
10   $\tilde{\gamma} = \text{FitParabola}(\tilde{\gamma})$ 
11  for  $\tilde{\psi} = (K-1)f$  to  $\tilde{\psi} = f$  do
12     $\tilde{\gamma} = \text{LocallyAdjust}(\tilde{\gamma})$ 
13    while CostFn( $\tilde{\gamma} - \Delta\tilde{\gamma}$ )  $\geq$  CostFn( $\tilde{\gamma}$ )  $\leq$  CostFn( $\tilde{\gamma} + \Delta\tilde{\gamma}$ ) do
14      | if CostFn( $\tilde{\gamma} - \Delta\tilde{\gamma}$ ) < CostFn( $\tilde{\gamma} + \Delta\tilde{\gamma}$ ) then
15        |  $\tilde{\gamma} = \tilde{\gamma} - \Delta\tilde{\gamma}$ 
16      end
17      else
18        |  $\tilde{\gamma} = \tilde{\gamma} + \Delta\tilde{\gamma}$ 
19      end
20    end
21     $\tilde{\gamma} = \text{FitParabola}(\tilde{\gamma})$ 
22    if CostFn( $\tilde{\gamma}$ ) < GlobalMinCF then
23      | GlobalMinCF = CostFn( $\tilde{\gamma}$ );  $\hat{\mathbb{C}}_2 = \tilde{\mathbb{C}}_2$ 
24    end
25    else
26      | if CostFn( $\tilde{\gamma}$ ) >  $th \cdot \text{GlobalMinCF}$  then break
27    end
28  end
29 end
```

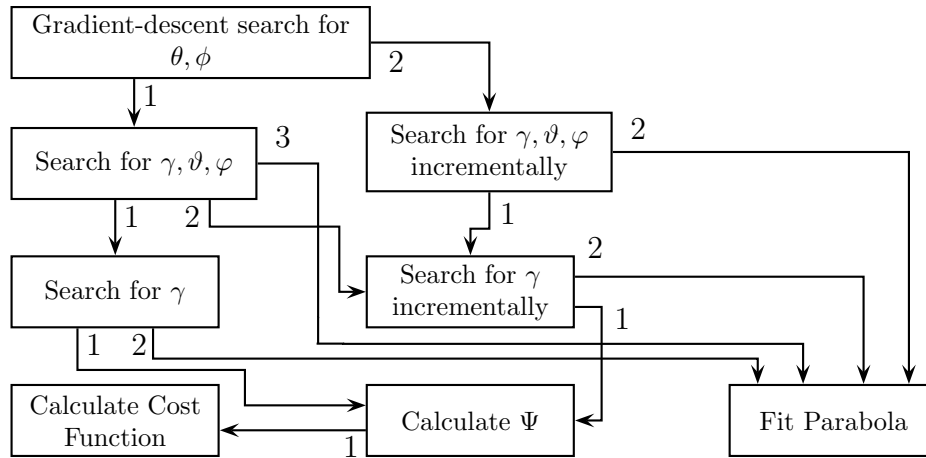


Figure 51: Cost function minimisation algorithm overview.

The proposed optimizations result in very high reduction of the number of checked solutions. In the experiments performed for this project, the time needed to find the minimum cost function did not exceed 1.5 seconds on a Pentium 4, 2.8 GHz processor. With the speed of checking 380,000 points per second, it means that less than 600,000 points were visited. Since the goal is to estimate the values of five parameters, it gives a rough average of only 14 samples per domain for each parameter. This is a very good result considering that the precision used in the algorithm is better than $1e-7$.

4.3 Comparison with known methods

This section contains a comparison of the proposed method of estimating the external camera parameters with the classical 8-point algorithm given by [61] and implemented by [81]. The original algorithm is due to Longuet-Higgins (see [93] as cited by [61]). The 8-point algorithm uses the least-squares method to estimate the

fundamental matrix from (3) using eight or more point matches. The minimisation is performed on the distances between points and their corresponding epipolar lines. The proposed method (DirectGS) does not explicitly estimate the fundamental matrix, but it is possible to calculate it from the two camera matrices.

The criterion used for comparing the two algorithms is an average distance from each point to its corresponding epipolar line (see Section 2.12). For experiments, a pair of images with 13 point matches was used (see Figure 35). In order to determine the precision of each algorithm on different sets of matches, several subsets of point matches were generated from this set of 13 points. Table 15 shows results of the experiments. The first column contains the number of point matches. The second column denotes how many runs were performed for a given number of point matches. The third column, shows the percentage of runs, for which the DirectGS method gave smaller error and the fourth column shows the percentage of runs for which the 8-point algorithm gave smaller error. Finally, the last column shows the percentage of runs for which the DirectGS algorithm did not find the global minimum. Figure 52 shows a sample run of both algorithms for 11 point matches. The three peaks for the DirectGS algorithm denote the cases when the local minimum instead of global was found. For most of the trials, the DirectGS error is much smaller than the 8-point algorithm.

The results from Table 15 show that in majority of cases the DirectGS performs better than 8-point algorithm. It must be noted that for cases when the DirectGS performs better than 8-point algorithm, the error being minimized is twice smaller for DirectGS algorithm. The cases when the DirectGS cannot find the global solution

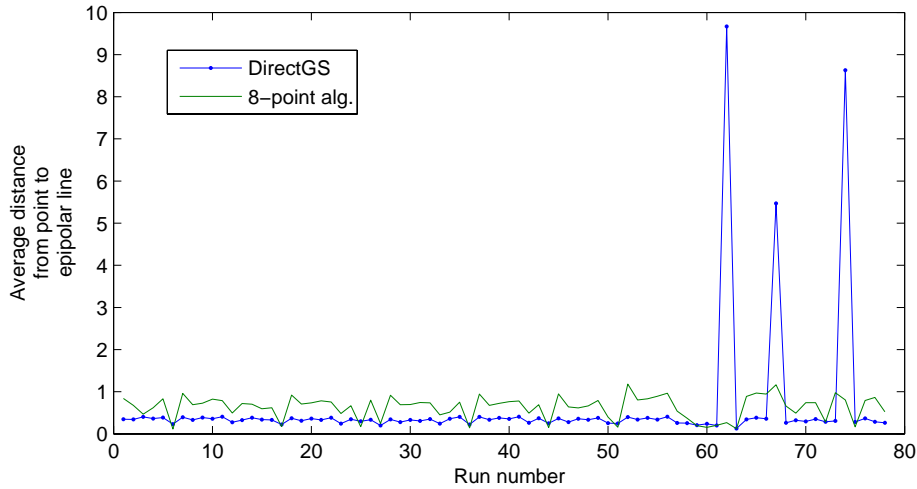


Figure 52: Sample run for 11 point matches of DirectGS and 8-point algorithm.

are caused by an initial guess of azimuth and altitude that is too far from the global minimum. This problem can be solved by choosing the initial guess to be located on the other side of the sphere or by increasing the number of points used in calculations.

The runtimes for both algorithms cannot be precisely compared since the 8-point algorithm was implemented in `Matlab` and DirectGS in `C++`. A rough estimate suggests that the 8-point algorithm is several times faster than the DirectGS algorithm, even though the total calculations for the DirectGS algorithm are below 1.5 second (on Pentium 4, 2.8 GHz processor).

The obtained results confirm that the main goal for this project was achieved. Namely, the precision of the proposed algorithm is better than the 8-point algorithm. The reason is that the DirectGS minimises the function that is the physical quantity, *i.e.*, the distance between the reprojected rays. Most known algorithms, like the 8-point algorithm, minimise the error of the reprojection, but in the image plane. This

means that the cost function is the two dimensional projection of the 3D function. The dimensionality reduction results in a less precise estimation of the true locations of the cameras. On the other hand, the proposed algorithm minimises the actual difference between the two corresponding projection rays in the 3D space. As a result, obtained reconstruction of 3D scene is very precise.

Table 15: Comparison of 8-point algorithm and DirectGS.

# of points	# runs	% of DirectGS	% of 8-point	% of local min.
13	1	100	0	0
12	13	92.3	7.7	0
11	78	82.1	17.9	3.8
10	286	70.6	29.4	14
9	715	58.3	41.7	31

5 Conclusions and Future Work

The work presented in this dissertation generated improvements in several areas of 2D to 3D conversion field. The following subsections summarize the results of this research and point towards areas of improvement.

Segment Matching

This dissertation introduces an approach to using a combination of genetic algorithms and approximation spaces in solving the image segment matching problem. Approximation spaces are used as a form of visual perception of pairs of images, which is a step toward 2D image classification in the case where one of the paired images plays the role of a reference image for matching purposes. Future work includes the extension of the proposed method to various shapes. It should be observed that the Hough transform can be applied to detect shapes like circles or ellipses. The next step is to define a class of shapes, which can be detected in a 2D image so that a 3D shape can be uniquely determined from a disparity map.

Dense Matching

This research introduces an approach to dense matching of 3D object structures that utilises the presence of points on lines in 3D space. A dense matching method that utilizes the presence of straight lines in a reconstructed model is presented. Matching is guided by points belonging to straight lines. In the first stage, lines in 3D space are found and matches corresponding to these lines are identified. In the second stage, identified matches are used as seed points in dense matching of two

views. As a result of the carried out approach to dense matching, a constructed 3D model is almost free from any noise, *i.e.*, only points belonging to a 3D structure with lines are extracted.

2D to 3D Conversion

A solution to the problem of estimating the external camera parameters using 3D geometry, rather the fundamental matrix widely used in 2D to 3D conversion, is another outcome of this research. The carried out geometric method estimates a camera's external parameters based on pairs of images. This method minimises the distance between two reprojected rays for two given point matches. This approach results in an optimal search space that reduces the computational burden required to find the minimum of a cost function. Experiments that have been performed show quite high dependency between the azimuth and altitude in the proposed camera setup. Future work beyond what has been presented includes work on reducing the search space by relating the altitude and azimuth of a second camera.

A Developed Applications

This appendix contains short descriptions of main applications developed²⁹ by the author for this dissertation. The McCabe complexity was calculated using [207]. The McCabe complexity informs about the complexity of a code by calculating branches in the code. Therefore, the more branching (*if* statements) the higher McCabe complexity. Programs with McCabe complexity above 50 are considered to be very complex programs [20].

In addition, to the applications presented in this appendix, 393 `Matlab` scripts/programs were developed consisting of the total of 14019 lines of code (plus 4447 comment lines). The statistics for `Matlab` code were calculated using [206].

²⁹Using C or C++ language

A.1 PointMatch

Program Name	PointMatch, ver. 1.1
Platform	Windows
User Interface	GUI
Programing Language	C++, OpenGL, Matlab
Size, McCabe Complx.	6 classes, 1979 lines of code, 171

Overview

The PointMatch application's main goal is to provide a Graphical User Interface for warping images, see Section 2.8. The user can open two images in two separate windows and create several tiepoints in both images. Then, using the mouse, all tiepoints' locations can be adjusted to match corresponding pixels from the second image. Finally, the image is triangulised by Delaunay triangulation and displayed (see Figure 53 for example).

Features

- Delaunay triangulation (assembly optimised)
- OpenGL texture rendering

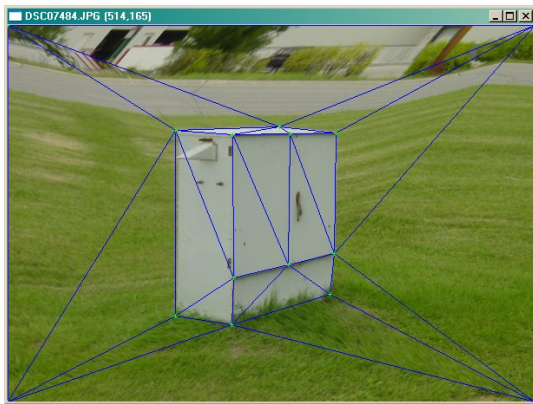
Screenshots



(a) Original left image



(b) Original right image



(c) Warped left image



(d) Right image with tiepoints

Figure 53: PointMatch application screenshots

A.2 SPoints

Program Name	SPoints, ver. 1.16
Platform	Windows
User Interface	GUI
Programing Language	C++, OpenGL
Size, McCabe Complx.	7 classes, 1267 lines of code, 86

Overview

This application positions points on the sphere, such that the distance between the closest two points is maximised. The user can define the number of points and the program, in the iterative manner, maximises the distances between the points. The maximisation is performed based on the distance of each point from its closest neighbourhoods. The number of points from the neighbourhood can be changed dynamically. In each iteration, each point is moved into the opposite direction than the sum of all vectors from a given point to all its neighbourhoods. Point's coordinates can be saved to an external file.

Features

- No limitation for the number of points
- OpenGL rendering
- Saving/loading points to/from a file
- Variable number of neighbours used for calculations
- Assembly optimised

Screenshots

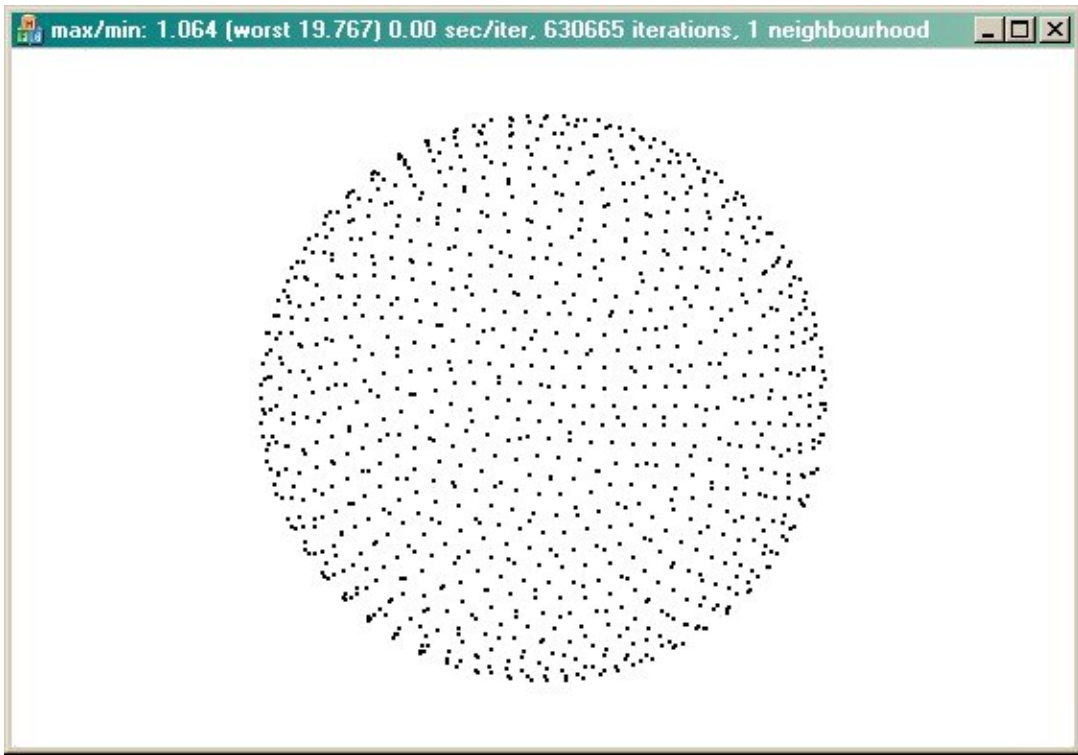


Figure 54: SPoints application screenshot

A.3 VisualBatch

Program Name	VisualBatch, ver. 1.37
Platform	Windows
User Interface	GUI
Programing Language	C++
Size, McCabe Complx.	14 classes, 4320 lines of code, 403

Overview

The VisualBatch application is the environment for building complex systems from small programs. There are several types of applications recognized by VisualBatch. The program can be any executable (exe) file, **Matlab** script, **Java** program or MS-DOS batch file. The user can define a sequence in which given programs are to be executed, including branching and conditional branching. Branching allows for parallel processing on multiprocessor machines and conditional branching allows to set up loops, such that certain calculations are performed only after required results are generated. In addition, the VisualBatch application can be run in batch mode (using command line arguments), which allows for forming even more complex systems. Figure 55 shows an example of program prepared for batch processing. Figures 56, 57 and 58 contain three selected steps from the 2D to 3D conversion process.

Features

- Support for exe files, **Matlab** scripts, batch files, **Java** files
- Saving/Loading project files
- Command line parameters (for batch processing)
- Saving output in a log with **Matlab** errors highlighting
- Branching and conditional branching

Screenshots

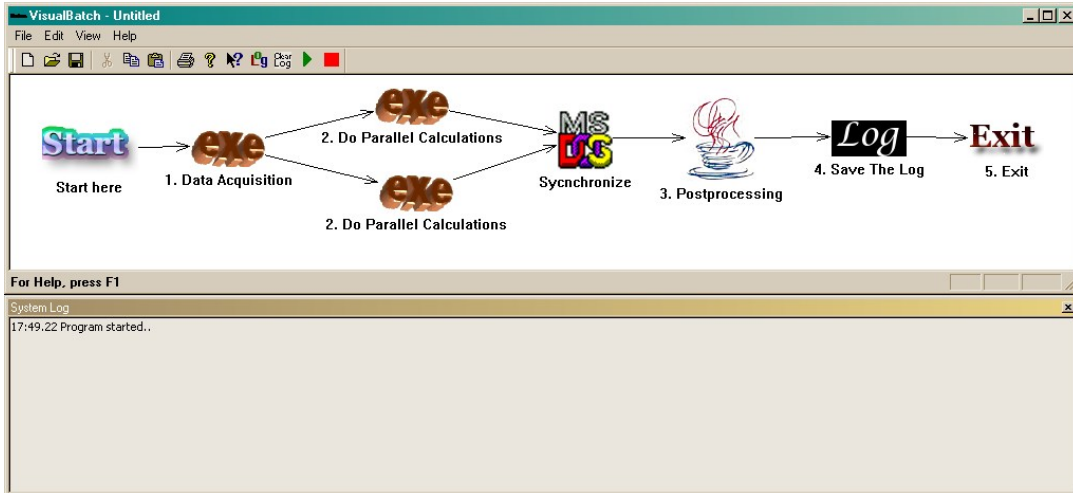


Figure 55: Sample script created in VisualBatch

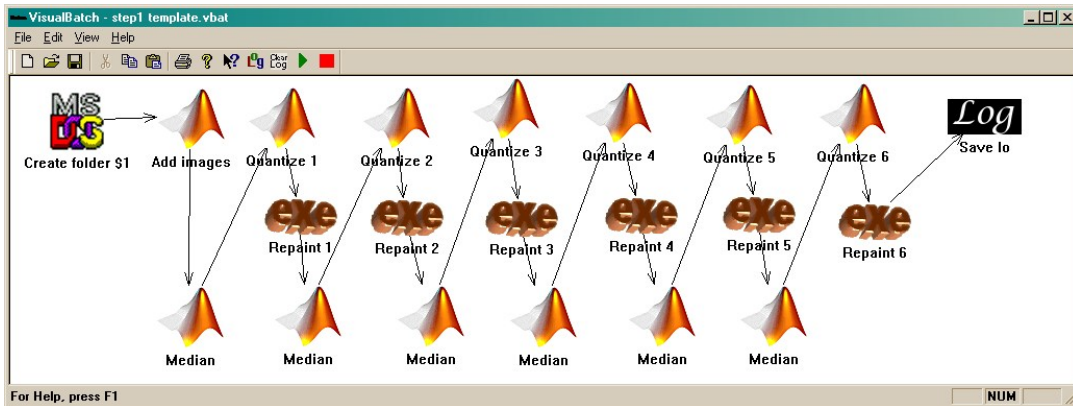


Figure 56: Image quantization described in Section 3.1.1

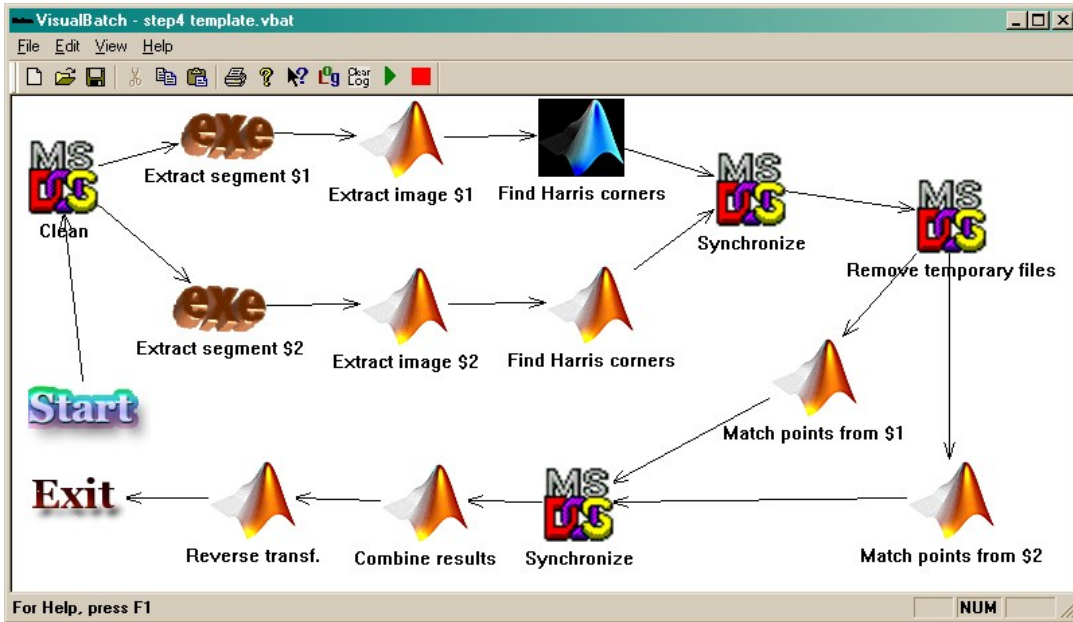


Figure 57: Script for point matching described in Section 3.2.6

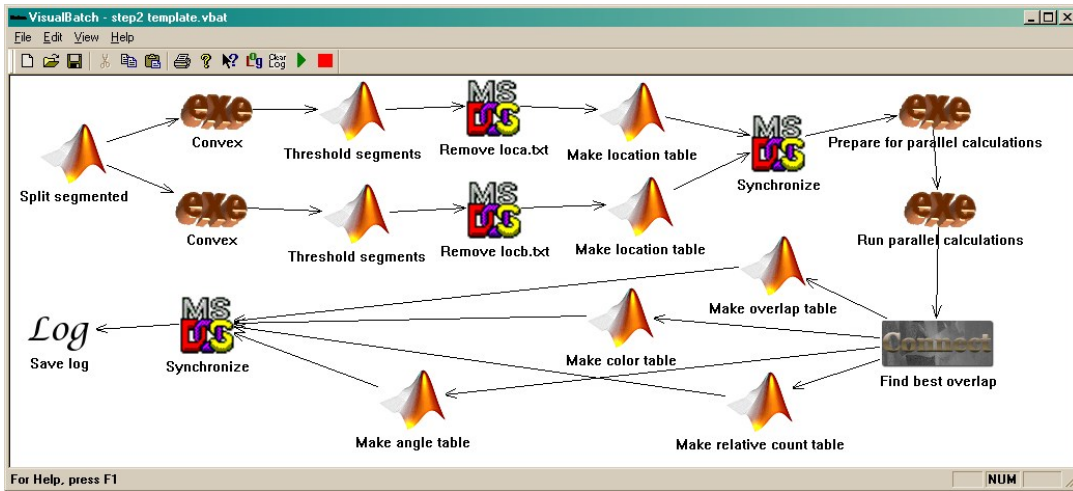


Figure 58: Script for segment matching described in Section 3

A.4 HypoDissertation

Program Name	HypoDissertation, ver. 1.3
Platform	Windows
User Interface	GUI
Programing Language	C++
Size, McCabe Complx.	16 classes, 4072 lines of code, 522

Overview

The HypoDissertation application implements the genetic algorithm approach for segment matching described in Section 3.1.8. The input consists of six files containing centroids of segments, colour table, overlap table, relative count table and angle table. Then, the genes are created and the population is build in an iterative manner. The user has a full control over the cost function that controls the population size. The algorithm that selects the solution contains several options including rough coverage method (see Section 3.1.9). In addition, the relation created by the genetic algorithm can be transformed to an equivalence relation, as described in Section 3.1.10.

Features

- Saving/Loading project files
- Saving/Loading settings to/from .ini file

Screenshots

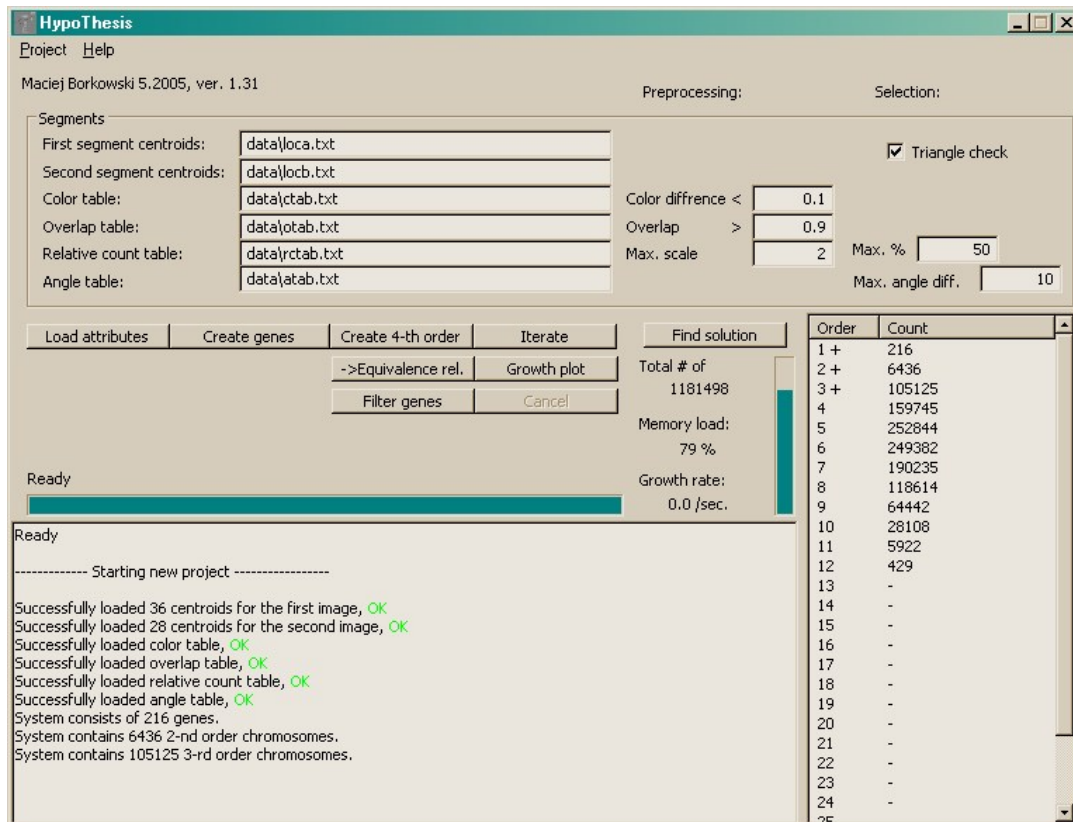


Figure 59: HypoDissertation application screenshots

A.5 Parallel Computations - The Server

Program Name	Overlap, ver. 1.08, RServer, ver. 2.18, DMServer, ver. 1.14a
Platform	Windows
User Interface	GUI
Programing Language	C++
Size, McCabe Complx.	Overlap: 13 classes, 2692 lines of code, 234 RServer: 15 classes, 3053 lines of code, 275 DMServer: 16 classes, 3227 lines of code, 398

Overview

The project called Parallel Computations is a set of classes that can be used for implementing parallel calculations. The server side is responsible for loading the (possible large) set of points for which some cost function is to be evaluated. Then these points are randomised and divided into blocks. After a clients connect to the server, blocks are sent out to the clients. The server keeps track of which blocks have been processed and which have not. In order to avoid delays in a case when some machines become overloaded, the same block can be sent to several machines and is marked as processed after the quickest machine returns the results. The server assures that the workload is optimally distributed and that all results match the points for which they were calculated (in case of broken connections).

For this dissertation, the Parallel Computations classes were used for three projects: Overlap for the calculation of the best overlap (as described in Section 3.1.4), the RServer for calculations of a cost function described in Algorithm 13 and the DMServer for dense matching (see Section 3.3).

Features

- Randomisation of points for each block - better load balancing
- Sending/receiving blocks implemented as a transaction (like in databases) - results in data consistency even in case of broken connections
- Estimates total elapsed time

- Build in ping command to test the connection speed
- Customisable block size
- Accepts up to 1024 clients

Screenshots

The screenshot displays the RServer application window titled "Parallel Computations". The interface is divided into several sections:

- System Panel:** Contains an "Initialise" button, "Jobs waiting: 1842", "Total power: 796.18", "Jobs away: 23", and "Jobs ready: 1422/3267". It also shows "Speed: 0.7 jobs/sec, for the last 2:20" and "Time left: 43:22".
- Server Panel:** Includes a "Port: 6446" field, a "Server" button, and a "Select All" button.
- Client Status Table:** A table with columns: Client, Status, Total, Min, Avg, Max, Jobs, and Ping. It lists 13 clients, most with a "working" status.
- Log Window:** Shows a sequence of events including "Loading jobs..", "32667 jobs loaded.", "Permutating jobs..", and "Each job contains 10 points.", followed by a list of client connection and disconnection messages with timestamps.

Client	Status	Total	Min	Avg	Max	Jobs	Ping
grid01	working	2.5	2.41	21.75	63.11	36	0.007
grid02	working	1.4	9.39	35.38	68.34	19	0.007
grid05	working	1.6	10.50	34.17	110.61	22	0.007
grid03	working	1.2	12.59	32.32	61.38	15	0.007
grid08	working	1.6	12.27	37.33	106.11	23	0.007
grid04	working	3.8	3.28	14.33	42.86	53	0.007
grid07	working	1.2	17.06	45.23	125.00	17	0.007
grid10	working	1.4	10.70	36.89	100.14	20	0.007
grid09	working	1.8	8.11	30.31	170.72	26	0.007
starbuck	working	3.2	3.17	17.99	93.59	45	0.007
morris	working	3.3	3.83	15.71	54.75	47	0.007
dauphin	working	0.9	13.13	80.88	314.45	13	0.007
elsa	working	1.8	4.59	18.65	66.97	26	0.000
elsa	working	1.6	6.44	21.20	48.23	23	0.000

Figure 60: RServer application screenshot

A.6 Parallel Computations - The Client

Program Name	OverlapClient, ver. 1.15, RClient, ver. 2.0 DMClient, ver. 1.18
Platform	Windows, Linux (Intel and Solaris), PocketPC
User Interface	Command line
Programming Language	C
Size, McCabe Complx.	OverlapClient: 1000 lines of code, 158 RClient: 1287 lines of code, 158 DMClient: 1213 lines of code, 148

Overview

The project called Parallel Computations is a set of classes that can be used for implementing parallel calculations. The client side receives the data, performs necessary calculations and send the results back to the server.

For this dissertation, the Parallel Computations classes were used for three projects: Overlap for the calculation of the best overlap (as described in Section 3.1.4), the RServer for calculations of a cost function described in Algorithm 13 and the DMServer for dense matching (see Section 3.3).

Features

- Multi-platform support
- Implements communication protocol to exchange data with PC Server in a safe manner

A.7 TwoViews

Program Name	TwoViews, ver. 1.19
Platform	Windows
User Interface	GUI
Programing Language	C++
Size, McCabe Complx.	11 classes, 3381 lines of code, 269

Overview

The TwoViews application allows for visualization of the geometry of two cameras in 3D space. The program shows two cameras in 3D space with all projecting rays corresponding to two sets of points from both cameras. In addition, the smallest approach between two corresponding rays is drawn. The user can modify all cameras' parameters. The cost function is visualized on the sphere, where the azimuth and altitude of each point correspond to the azimuth and altitude of the second camera.

Features

- Fast OpenGL rendering
- Import of camera parameters from an external file
- Cost function visualization for variable azimuth/altitude or pan/tilt
- Visualization of the epipolar plane

Screenshots

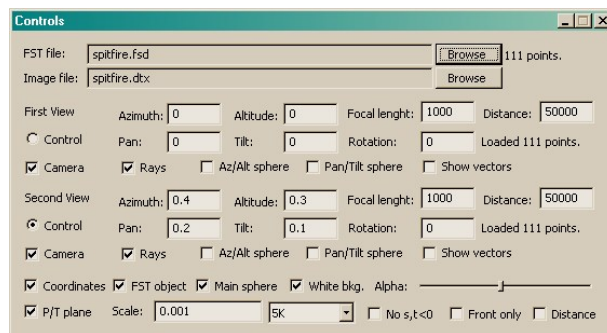


Figure 61: TwoViews application (control pane)

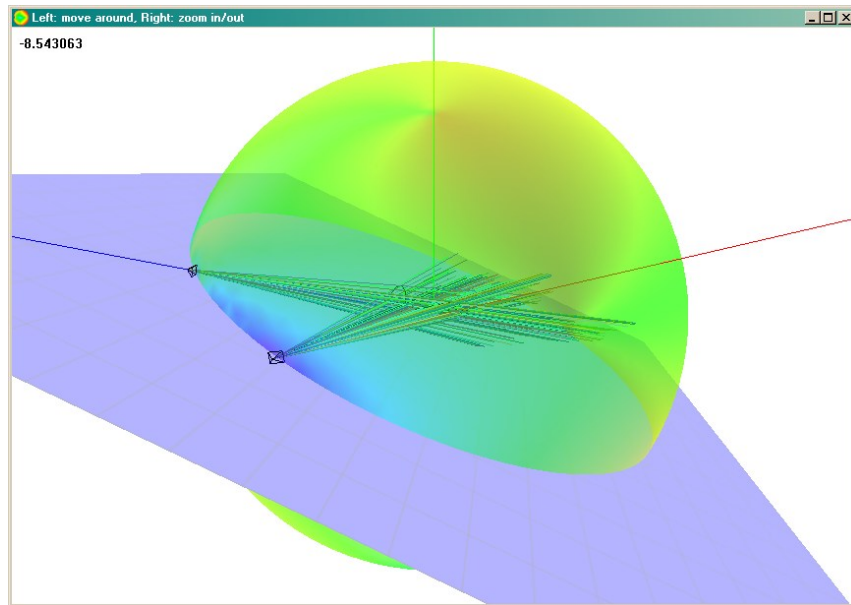


Figure 62: TwoViews application (main pane)

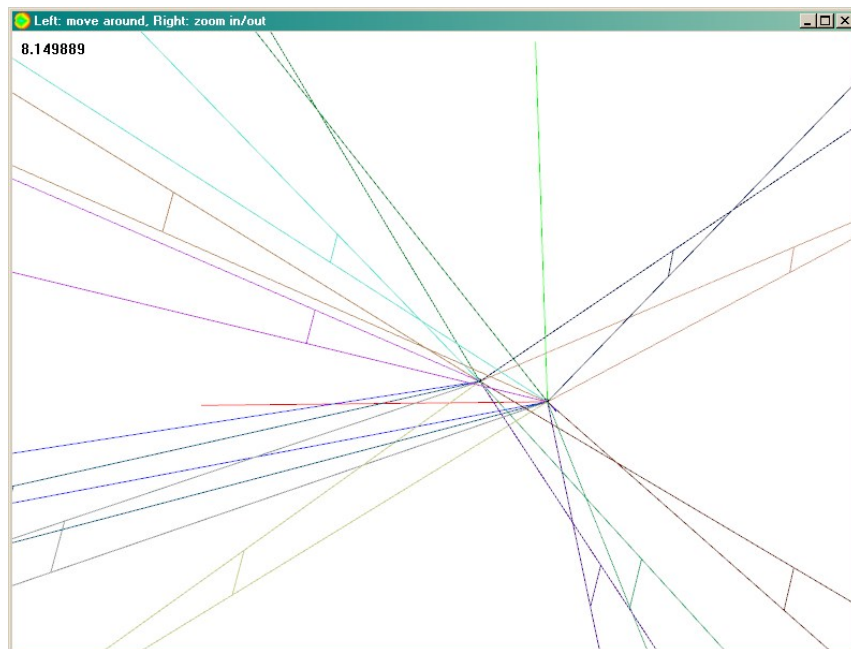


Figure 63: Closeup for several rays and their closest approach lines

A.8 2DSearch

Program Name	2DSearch, ver. 2.07
Platform	Windows
User Interface	Command line
Programing Language	C
Size, McCabe Complx.	900 lines of code, 116

Overview

The 2DSearch program implements the calculations of the cost function described in Section 4.2. This program is based on the RClient application described in A.6 (but no TCP/IP functionality was implemented).

Screenshots

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

Z:\>2DSearch

2D Search version 2.06
Intel compilation.
Entering main loop...
images filename: tower1_1.dtx
XbYb loop: 1e-007
XcYc loop: 1e-007
Zc loop : 180
Zc initial range: 3.1415
Reading file: tower1_1.dtx..
number of points: 13
Mass centre for the left image: <-37.89,-210>
Mass centre for the right image: <-85.32,-198.7>
Focal length: 5983.14, radius: 15000. # of jobs: 4096
Done: 100%
132138 calls processed.
Elapsed time: 0.4 sec, 9351.6 points/sec
Saving to files: z:\res.out and z:\res.p7.out.
Looking for precise solution..
Cost function reduced from 4.1329445 to 2.8452106
374715 calls processed.
Elapsed time: 0.7 sec
Saving to file: z:\solution.out.
Main loop exited...

Z:\>_
```

Figure 64: 2DSearch application screenshot

A.9 BiCuGPU

Program Name	BiCuGPU, ver. 1.2
Platform	Windows
User Interface	Command line
Programing Language	C, OpenGL, GLSL
Size, McCabe Complx.	1564 lines of code, 96 GLSL code: 527 lines of code, 0

Overview

The BiCuGPU application is an implementation of the bicubic interpolation on Graphical Processing Unit (graphic card). The goal is to speed up the calculations and perform the interpolation on the processor belonging to the graphic card, not the Central Processing Unit. The programming language used for programming the GPU is the OpenGL Shading Language (GLSL). One of the limitations of the GLSL is the absence of any loop statements. Therefore, on the GPU the bicubic interpolation has to be implemented without any loops. The hardware limitation is a very small total length of the executable code that can be uploaded into the graphic card. Therefore, the bicubic interpolation, had to be split into nine subprograms that were run separately for each colour channel. Despite the fact that the number of arithmetic operations was ten times greater than in CPU implementation, the GPU implementation was faster. This means that the GPU code is over ten times faster than the CPU implementation.

Features

- Use of the OpenGL interface to access hardware acceleration
- Over ten times more processing power than CPU
- Access to massive parallel processing units (4 processors)

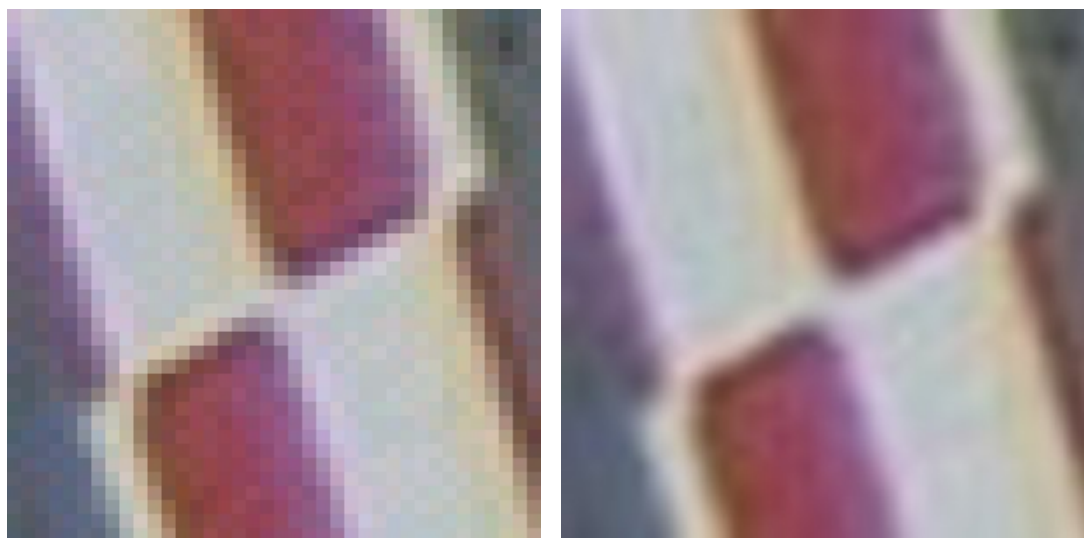
Screenshots

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

Z:\>BiCuGPU input.tga 8
Flipping image horizontally.. OK
glTexImage2D(): [PASS]
Created a 140 by 133 floating point texture.
glTexImage2D(): [PASS]
Created a 1120 by 1064 floating point texture.
glTexImage2D(): [PASS]
Created a 1120 by 1064 floating point texture.
glTexImage2D(): [PASS]
Created a 1120 by 1064 floating point texture.
glFrameBufferTexture2DEXT(): [PASS]
First pass done!
Second pass done!
Third pass done!
Fourth pass done!
Fifth pass done!
Sixth pass done!
Seventh pass done!
Eighth pass done!
Ninth pass done!
Image processed in 0.31 sec.

Z:\>
```

Figure 65: BiCuGPU application screenshot



(a) Resized image

(b) Bicubic interpolation

Figure 66: BiCuGPU application screenshots of 8x magnification

Sample code

The following is the GLSL code for the ninth pass of bicubic interpolation.

```
uniform sampler2DRect Texture0,Texture1,Texture2;
uniform sampler2DRect TextureOut;
uniform float scale,offx,offy;

void main(void)
{
    float y0,y1,y2,y3,y10,y11,y12,y13,y20,
    float y21,y22,y23,y120,y121,y122,y123;
    float c6,c7,c8,c9,c14,c15;
    vec2 texCoord=gl_TexCoord[0].st;
    vec2 tCoord=(texCoord+vec2(offx,offy))/scale;

    float t=texture2DRect(Texture2, gl_TexCoord[0].st).y;
    float u=texture2DRect(Texture2, gl_TexCoord[0].st).z;

    y0 = texture2DRect(Texture0, tCoord).z;
    y10 = texture2DRect(Texture1, texCoord).y;
    y20 = texture2DRect(Texture1, texCoord).z;
    y120= texture2DRect(Texture1, texCoord).w;

    texCoord.x+=scale;tCoord.x+=1.0;
    y1 = texture2DRect(Texture0, tCoord).z;
    y11 = texture2DRect(Texture1, texCoord).y;
    y21 = texture2DRect(Texture1, texCoord).z;
    y121= texture2DRect(Texture1, texCoord).w;

    texCoord.y-=scale;tCoord.y-=1.0;
    y2 = texture2DRect(Texture0, tCoord).z;
    y12 = texture2DRect(Texture1, texCoord).y;
    y22 = texture2DRect(Texture1, texCoord).z;
    y122= texture2DRect(Texture1, texCoord).w;

    texCoord.x-=scale;tCoord.x-=1.0;
    y3 = texture2DRect(Texture0, tCoord).z;
    y13 = texture2DRect(Texture1, texCoord).y;
```

```

y23 = texture2DRect(Texture1, texCoord).z;
y123= texture2DRect(Texture1, texCoord).w;

c6=3.0*(y13-y10)-2.0*y120-y123;
c7=2.0*(y10-y13)+y120+y123;
c8=3.0*(y1-y0)-2.0*y10-y11;
c9=3.0*(y21-y20)-2.0*y120-y121;

float y0123=y0-y1+y2-y3, y1213=y12-y10-y11+y13;
float y2223=y22-y23, y2120=y21-y20;
c14=-6.0*y0123+4.0*y2120+3.0*y1213+2.0*(y2223-y120-y121)
      -y122-y123;
c15=4.0*y0123 - 2.0*(y1213+y2120+y2223)+y120+y121+y122+y123;

float p=texture2DRect(Texture2, gl_TexCoord[0].st).x+
      t*t*t*(c15*u+c14)*u*u+t*(((c7*u+c6)*u+y120)*u+y10)+
      t*t*(u*c9+c8);

p=clamp(p,0.0,255.0);

vec4 m=texture2DRect(TextureOut, gl_TexCoord[0].st);
gl_FragColor = vec4(p,m.y,m.z,0.0);
}

```

A.10 MeshReduction

Program Name	MeshReduction, ver. 1.24
Platform	Windows
User Interface	GUI
Programing Language	C++, OpenGL
Size, McCabe Complx.	5 classes, 1274 lines of code, 177

Overview

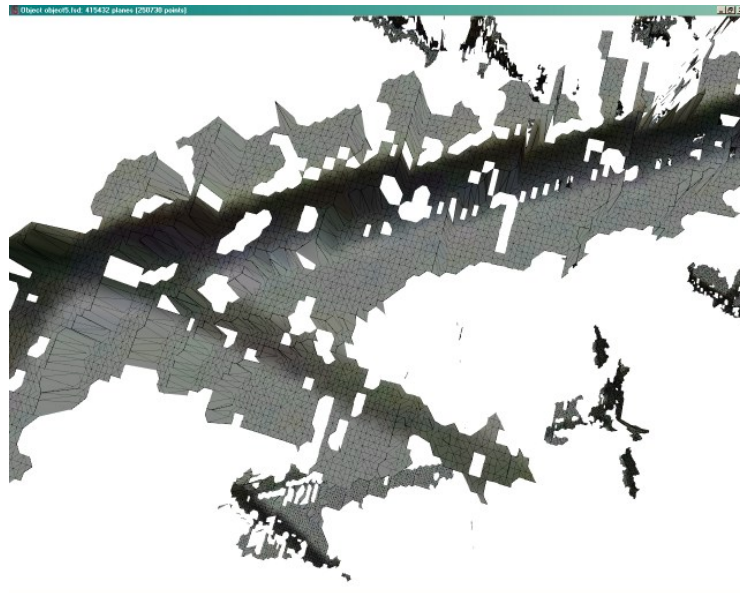
The goal of the MeshReduction application is to decrease the number of triangles forming an object. After dense matching (see Section 3.3), the reconstructed model consists of several hundreds of thousands of triangles. This application uses the fact that for flat surfaces the number of triangles forming the surface can be reduced without any visual degeneration of the shape. The process of removing triangles that do not introduce any valuable information to the shape consists of the following steps.

1. Detect inside/boundary points.
2. Detect all edges that can be removed.
3. Find edges, for which adjacent triangles are almost parallel (lies on the same plane).
4. Remove selected edges.
5. Recreate triangles.
6. Remove unused points.
7. Update texture information.

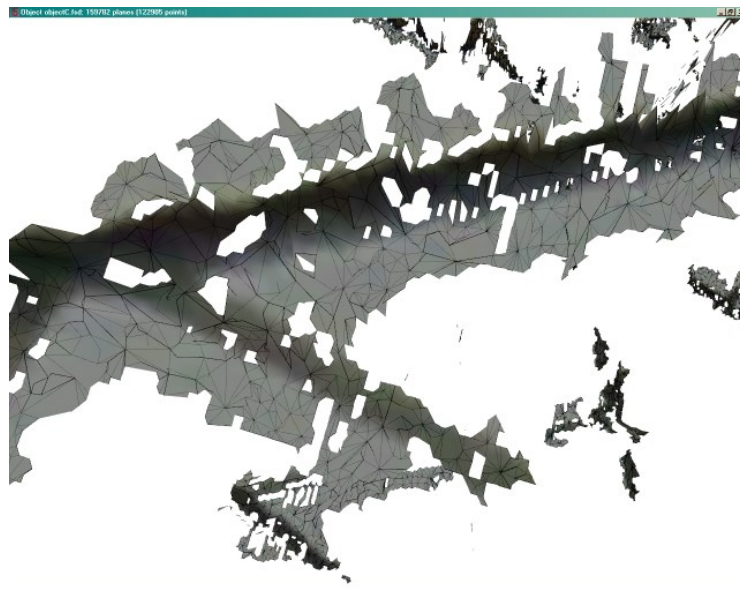
Features

- Removes triangles from the model and updates texture mapping

Screenshots



(a) Original dense model



(b) Model from 67(a) with reduced number of triangles

Figure 67: Fragment of a 3D model of a power tower.

A.11 TexturedView

Program Name	TexturedView, ver. 1.12
Platform	Windows
User Interface	GUI
Programing Language	C++, OpenGL
Size, McCabe Complx.	7 classes, 1326 lines of code, 101

Overview

The TexturedView application renders a 3D model with a texture overlapped on it.

Features

- OpenGL fast rendering
- Optimised for object consisting of hundreds of thousands of triangles

Screenshots

See screenshots for the MeshReduction application in A.10.

A.12 UnrollLoops

Program Name	UnrollLoops, ver. 1.0
Platform	Windows
User Interface	GUI
Programing Language	C++
Size, McCabe Complx.	3 classes, 347 lines of code, 10

Overview

The UnrollLoops application unrolls the “for” loops for C/C++ code. This program was used as a help tool in creation of GLSL code for the bicubic interpolation (see A.9). No nesting loops is allowed, therefore, nested loops have to be unrolled in two stages (see screenshots section below).

Screenshots

The screenshot in Figure 68 shows the second step of unrolling two nested for loops from Tab. 16.

```
for (int j=0; j<4; j++)
  for (int k=0; k<4; k++)
  {   val1.Format("cell[%i][%i]",j,k);
      val2.Format("%i",cell[j][k]);
      ReplaceString(&str, val1, val2);}

```

Table 16: Two sample nested “for” loops.

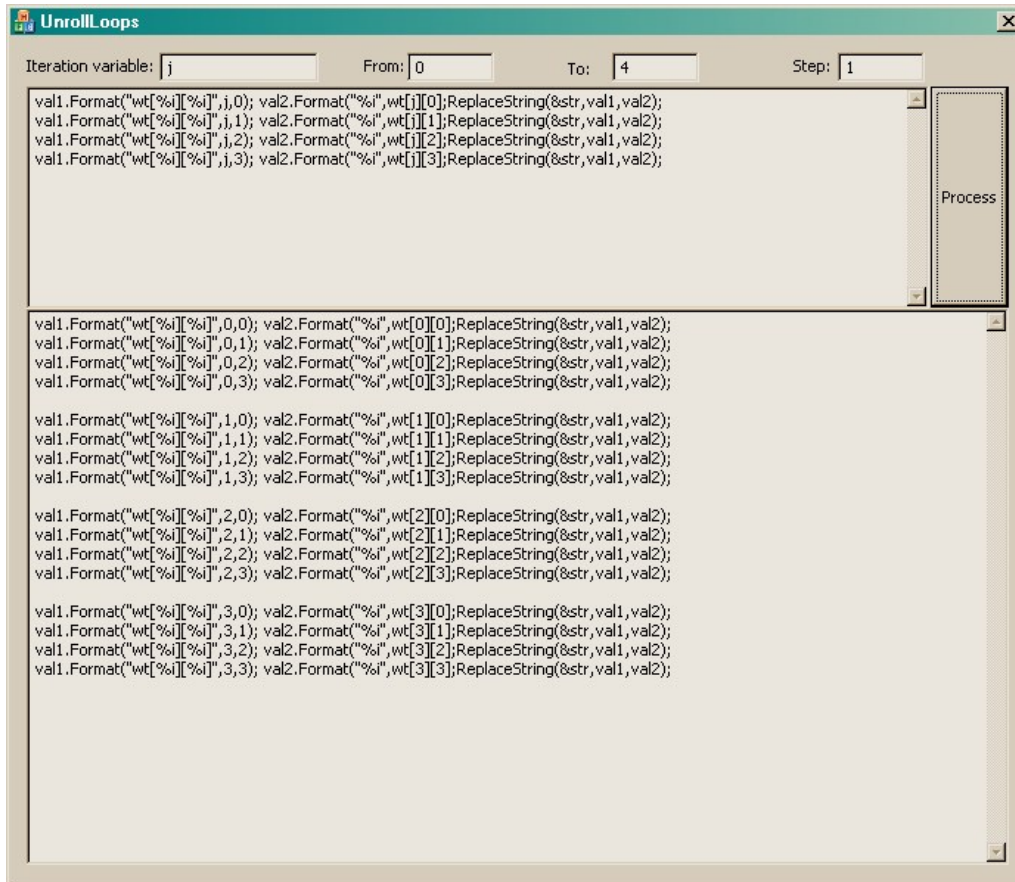


Figure 68: UnrollLoop application screenshot

References

- [1] Atkinson, K.B.: *Developments In Close Range Photogrammetry - 1*, Applied Science Publishers Ltd, London, 1980.
- [2] Barnard, S.T., Fischler, M.A.: Computational Stereo, *Computing Surveys* 14(4), 1982, 553–554.
- [3] Basu, M.: Gaussian-Based Edge-Detection Methods - A Survey, *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews* 32(3), 2002, 253.
- [4] Bhattacharya, P., Liu, H., Rosenfeld, A., Thompson, S.: Hugh-transform Detection of Lines in 3-D Space, *Pattern Recognition Letters* 21(9), 2000, 843–849.
- [5] Bishop, C.M.: *Neural Networks For Pattern Recognition*, Oxford Univ. Press, 1995, 354.
- [6] Bobick, A.F., Intille, S.S.: Large Occlusion Stereo, *Kluwer Academic Publishers* 33(3), Hingham, USA, 1999, 181–200.
- [7] Borkowski, M.: *Digital Image Processing in Measurement of Ice Thickness on Power Transmission Lines: A Rough Set Approach*, M.Sc. Dissertation, supervisor: J.F. Peters, Department of Electrical and Computer Engineering, University of Manitoba, 2002
- [8] Borkowski, M.: Signal Analysis Using Rough Integrals, *Rough Sets and Current Trends in Computing* LNAI 2475, Springer-Verlag, Berlin, 2002, 218–225.
- [9] Borkowski, M., Peters, J.F.: Direct Geometrical Search, 2D to 3D conversion, *International Journal of Computer Vision*, 2007, *submitted*.
- [10] Borkowski, M., Peters, J.F.: Matching 2D image segments with genetic algorithms and approximation spaces, *Transactions on Rough Sets V*, LNCS 4100, 2006, 63–101.

- [11] Borkowski, M., Peters, J.F.: Dense Matching of Two Views: Establishing Correspondences Between Points in 3D Lines, *IEEE Transactions on Image Processing*, 2007, *submitted*.
- [12] Borkowski, M., Peters, J.F.: Approximating Sensor Signals: A Rough Set Approach, *Proceedings of Canadian Conference on Electrical and Computer Engineering (CCECE'02)* 2, 2002, 980–985.
- [13] Boufama, B., Ghanem, K.: Achieving efficient dense matching for uncalibrated images, *Proceedings of the IEEE International Conference on Image Processing*, Italy, 2005, 1069–1072.
- [14] Boufama, B., Jin, K.: Towards a fast and reliable dense matching algorithm, *The 15th International Conference on Vision Interface*, Calgary, Canada, 2002.
- [15] Brewster, D. Sir : *The Stereoscope, Its History, Theory, And Construction With Its Application To The Fine And Useful Arts And To Education*, London: John Murray, Albemarle Street, 1856, reprinted by Rudolf Kingslake, The Fountain Press, London, 1971.
- [16] Bronshtein, I.N., Semendyayev, K.A., Musiol, G., Muehlig, H.: *Hanbook of Mathematics*, 4th Ed., Springer, Berlin, 2004, 292.
- [17] Caelli, T., Reye, D.: On the classification of image regions by colour, texture and shape, *Pattern Recognition* 26, 1993, 461–470.
- [18] Cantzler, H.: *Improving Architectural 3D Reconstruction By Constrained Modelling*, Doctoral Dissertation, Institute of Perception, Action and Behaviour School of Informatics University of Edinburgh, 2003, 117–120.
- [19] Chang, M.M.Y., Wong, K.H.: 3D Model Reconstruction by Constrained Bundle Adjustment, *Proceedings of the 17th International Conference on Pattern Recognition (ICPR04)*, 2004, 1.
- [20] Charney, R.: Programming Tools: Code Complexity Metrics, <http://www.linuxjournal.com/article/8035>

- [21] Cheffins, O.W, Chisholm, N.W.T.: Engineering And Industrial Photogrammetry, In [1], 149–180.
- [22] Chen, Q., Medioni, G.: Efficient Iterative Solution to M-View Projective Reconstruction Problem, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)* 2, 1999, 2056.
- [23] Cottier, J.C.: Extraction et appariements robustes des points d'intérêt de deux images non étalonnées, Internal Report, INRIA, Rhone-Alpes, 1994.
- [24] Darwin, C.: *On the Origin of the Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*, Murray, London, 1859.
- [25] Dowman, I.J., Scott, P.J.: Photogrammetric Theory, Techniques And Problems, In [1], 15–37.
- [26] Duda, R.O., Hart, P.E.: *Pattern Classification and Scene Analysis*, John Wiley & Sons, USA, 1973, 379–404.
- [27] Duda, R.O., Hart, P.E., Stork, D.G: *Pattern Classification*, Second Edition, John Wiley & Sons, USA, 2001, 259, 262, 373–377, 621, 625.
- [28] Dufournaud, Y., Schmid, C., Horaud, R.: Matching Images with Different Resolutions, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, South Carolina, USA, 2000, 612–618.
- [29] Eshelman, L.J.: Genetic algorithms, *T. Back, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation*, Oxford University Press, Oxford, UK, 1997.
- [30] Esteban, C.H., Schmitt, F.: Multi-Stereo 3D Object Reconstruction, *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission*, Paris, France, 2002.

- [31] Faugeras, O., Luong, Q.T.: *The Geometry of Multiple Images, The Laws That Govern The Formation Of Multiple Images Of A Scene And Some Of Their Applications*, The MIT Press, 2001, 3, 7–9, 582–584.
- [32] Faugeras, O.: From Geometry To Variational Calculus: Theory And Applications Of Three-Dimensional Vision, *Computer Vision for Virtual Reality Based Human Communications*, 1998, Bombay, India, pp. 60–61.
- [33] Feng, C.L., Hung, Y.S.: A Robust Method for Estimating the Fundamental Matrix, *Proceedings of the VII-th Digital Image Computing: Techniques and Applications*, Sun C., Talbot H., Ourselin S. and Adriaansen T. (Eds.), 2003, Sydney, 633–642.
- [34] Fischler, M.A., Bolles, R.C.: Random Sample Consensus: A Paradigm For Model Fitting With Application To Image Analysis And Automated Cartography, *Communications of the ACM* 24(6), 1981, 381-385.
- [35] Fitzpatrick, J.M, Hill, D.L.G., Calvin, R.M. Jr.: Chapter 8: Image Registration, *Handbook of Medical Imaging, Vol. 2: Medical Image Processing and Analysis*, Editors: Milan Sonka and J. Michael Fitzpatrick, SPIE PRESS Vol. PM80, 2000, Bellingham, USA, 449, 488–489, 499.
- [36] Flusser, J., Suk, T.: A Moment-Based Approach to Registration of Images with Affine Geometric Distortion, *IEEE Transactions On Geoscience And Remote Sensing* 32(2), 1994, 383, 384.
- [37] Flusser, J.: Moment Invariants in Image Analysis, *Proceedings of the International Conference on Computer Science ICCS'06* 11, Prague, 2006, 196–199.
- [38] Freeman, W.T., Adelson, E.H.: The Design And Use of Steerable Filters, *IEEE Transactions On Pattern Analysis And Machine Intelligence* 13(9), 1991, 891.
- [39] Förstner, W.: A framework for low level feature extraction, *ECCV*, 1994, 383–394.

- [40] Fu, K.S., Mui, J.K.: A survey of image segmentation, *Pattern Recognition* 13, 1981, 3–16.
- [41] Fusiello, A., Trucco, E., Verri, A.: A compact algorithm for rectification of stereo pairs, *Machine Vision Applications* 12(1), 2000, 16.
- [42] Galpin, F., Morin, L.: Video Coding Using Streamed 3D Representation, *International Conference on Image Processing* 3, 2000, 636–639.
- [43] Galpin, F., Balter, R., Morin, L., Deguchi, K.: 3D Models Coding and Morphing for Efficient Video Compression, *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR04)* 1, 2004, 334–341.
- [44] Garstenauer, H.: *A unified framework for rigid body dynamics*, M.Sc. Thesis, supervisor: Jens Volkert, Johannes Kepler Universität Linz, 2006.
- [45] Gaskett, C.: *Q-Learning for Robot Control*, Doctoral Dissertation, Research School of Information Sciences and Engineering at the Australian National University, 2002.
- [46] Gersho, A., Gray, R.M.: *Vector Quantization And Signal Compression*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992, 133–134, 188–190.
- [47] Gomolinska, A.: Rough validity, confidence, and coverage of rules in approximation spaces, *Transactions on Rough Sets* III, 2005, 57–81.
- [48] Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*, Second Edition, Prentice Hall, USA, 2002, 49–50, 52–54, 66, 116–118, 271–276, 289–302, 519, 539, 577–579, 587–591, 672–675.
- [49] Gottesfeld Brown, L.: A Survey Of Image Registration Techniques, *ACM Computing Surveys* 24(4), 1992, 333.

- [50] Gouet, V., Montesinos, P., Pele, D.: Stereo Matching Of Color Images Using Differential Invariants, *Proceedings of the International Conference on Image Processing*, Chicago, USA, 1998, 152–153.
- [51] Gouet, V., Montesinos, P., Pele, D.: A Fast Matching Method For Color Uncalibrated Images Using Differential Invariants, *Proceedings of the British Machine Vision Conference*, Southampton, 1998, 369.
- [52] Gouet, V., Montesinos, R., Deriche, D.: Differential Invariants for Color Images, *Proceedings of 14 th International Conference on Pattern Recognition*, Brisbane, Austria, 1998, 838–839.
- [53] Grafe, M., Wortman, R., Westphal, H.: AR-Based Interactive Exploration Of A Museum Exhibit Augmented Reality Toolkit, *The First IEEE International Workshop*, Darmstadt, Germany, 2002.
- [54] Grey, R.M., Neuhoff, D.L.: Quantization, *IEEE Transactions on Information Theory* 44, 1998, 1–63.
- [55] Gremban, K.D., Ikeuchi, K.: Appearance-Based Vision and the Automatic Generation of Object Recognition Programs, In [72], 235–239.
- [56] Hansen, B., Morse, B.: Multiscale Registration Using Scale Trace Correlation, *Computer Vision and Pattern Recognition (CVPR'99)*, 1999, 202.
- [57] Haralick, R.M., Shapiro, L.G.: Image segmentation techniques, *Computer Vision, Graphics, and Image Processing* 29, 1985, 100–132.
- [58] Harris, C., Stephens, M.: A Combined Corner And Edge Detector, *Proceedings of the 4th Alvey Vision Conference*, Manchester, 1988, 189-192.
- [59] Hartigan, J.A., Wong, M.A.: A K-means clustering algorithm, *Applied Statistics* 28, 1979, 100–108.

- [60] Hartley, R.: Projective Reconstruction and Invariants from Multiple Images, *IEEE Transactions On Pattern Analysis and Machine Intelligence* 16(10), 1994, 1037.
- [61] Hartley, R., Zisserman, A.: *Multiple View Geometry in Computer Vision*, Second Edition, Cambridge University Press, 2004, 33, 39, 65, 154–157, 239–246, 260, 281–282.
- [62] Heaton, K.G.: *Physical Pixels*, M.Sc. Thesis, MIT, 1994.
- [63] Heikkila, J., Silven, O.: A Four-step Camera Calibration Procedure with Implicit Image Correction, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, 1997, 1106–1112.
- [64] Heitger, F., Rosenthaler, L., von der Heydt, R., Peterhans, E., Kuebler, O.: Simulation of neutral contour mechanism: from simple to end-stopped cells, *Vision Research*, 32(5), 1992, 963–981.
- [65] Hirano, H., Tsumoto, S.: Segmentation of medical images based on approximation in rough set theory, *J.J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.) LNAI 2475*, Springer-Verlag, Berlin, 2002, 554–563.
- [66] Holland, J.H.: *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.
- [67] Horaud, R., Skordas, T., Veillon, F.: Finding geometric and relational structures in an image, *ECCV*, 1990, 374–384.
- [68] Horn, B.K.P.: *Robot Vision*, The MIT Press, USA, 1987, 202–232, 278–283, 299–323, 435–438, 449.
- [69] Hough, P.V.C.: Methods And Means For Recognizing Complex patterns, U.S. Patent 3,069,654, 1962.
- [70] Hu, M.K.: Visual Pattern Recognition By Moment Invariants, *IRE Trans. Inform. Theory* 8, 1962, 179, 185.

- [71] Jähne, B.: *Digital Image Processing*, 6-th revised and extended edition, Springer-Verlag, Berlin, 2005, 221.
- [72] Jain, A.K., Flynn, P.J.: Three-Dimensional Object Recognition Systems, *Elsevier Science Publishers B.V.* 1, 1993.
- [73] Jarvis, R.: Range Sensing for Computer Vision, In [72], 17–56.
- [74] Jung, I.K., Lacroix, S.: A Robust Interest Points Matching Algorithm, *Proceedings of Eighth IEEE International Conference on Computer Vision 2*, 2001, 538
- [75] Katsoulas, D.: *Robust Recovery of Piled Box-Like Objects in Range Images*, Ph.D. Dissertation, Albert-Ludwigs-Universitaet, Freiburg/Breisgau, 2004, 156–158.
- [76] Katsoulas, D.: Robust Extraction of Vertices in Range Images by Constraining the Hough Transform, *F.J. Perales et al. (Eds.) IbPRIA 2003 LNCS 2652*, Springer-Verlag, Berlin, 2003, 363-364.
- [77] Kim, C., Lee, K.M., Choi, B.T., Lee, S.U.: A Dense Stereo Matching Using Two-Pass Dynamic Programming with Generalized Ground Control Points, *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*, 2005, 1075–1082.
- [78] Koenderink, J.J., van Doorn, A.J.: Representation of local geometry in the visual system, *Biological Cybernetics* 55, 1987, 369–370.
- [79] Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: Rough Sets: A Tutorial, *S.K. Pal, A. Skowron (Eds.), Rough Fuzzy Hybridization: A New Trend in Decision-Making*, Springer-Verlag, Singapore, 1999, 1–14.
- [80] Konrad, E., Orłowska, E., Pawlak, Z.: Knowledge Representation Systems, Polish Academy of Sciences Report 433, Institute for Computer Science, 1981.

- [81] Kovesi, P. D.: MATLAB and Octave Functions for Computer Vision and Image Processing, School of Computer Science & Software Engineering, The University of Western Australia, Available from: <http://www.csse.uwa.edu.au/~pk/research/matlabfns/>
- [82] Kuipers, J.B.: *Quaternions And Rotation Sequences: A Primer With Applications To Orbits, Aerospace, And Virtual Reality*, Princeton University Press, USA, 1999, 1–86, 103–128, 357.
- [83] Lan, Z.D., Mohr, R.: Robust Location based Partial Correlation, INRIA Report No. 3186, 1997.
- [84] Lee, S.Uk, Chung, S.Y., Park, R.H.: A comparative performance study of several global thresholding techniques for segmentation, *Computer Graphics and Image Processing* 52, 1990, 171–190.
- [85] Lhuillier, M., Quan, L.: Robust Dense Matching Using Local And Global Geometric Constraints, *Proceedings of the 15th International Conference on Pattern Recognition* 1, Barcelona, Spain, 2000, 968–972.
- [86] Lhuillier, M.: Efficient Dense Matching for Textured Scenes Using Region Growing, INRIA Research Report 03382, 1998, 700–709.
- [87] Lhuillier, M., Quan, L.: Edge-Constrained Joint View Triangulation for Image Interpolation, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* 2, 2000, 218–224.
- [88] Lhuillier, M., Quan, L.: Image Interpolation by Joint View Triangulation, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2, Fort Collins, Colorado, USA, 1999, 139–145.
- [89] Liao, S.X., Pawlak, M.: On image analysis by moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996, 254–266

- [90] Lindeberg, T.: Feature Detection with Automatic Scale Selection, Technical report ISRN KTH/NA/P-96/18-SE, May 1996, Revised August 1998, *Int. J. of Computer Vision* 30(2), 1998, i, 3–6.
- [91] Lindeberg, T.: Scale-Space: A Framework For Handling Image Structures At Multiple Scales, *Proc. CERN School of Computing*, The Netherlands, 1996, 7.
- [92] Liu, Y., Zhang, X., Huang, T.S.: Estimation Of 3D Structure And Motion From Image Corners, *The Journal Of Pattern Recognition Society*, Elsevier Science Ltd., 2003, 1269, 1275.
- [93] Longuet-Higgins, H.C.: A computer algorithm for reconstructing a scene from two projections, *Nature* 293, 1981, 133–135.
- [94] Loog, M., Ginneken, B.v., Duin, R.P.W.: Dimensionality reduction by canonical contextual correlation projections, *T.Pajdla, J. Matas (Eds.) LNCS 3021*, Springer, Berlin, 2004, 562–573.
- [95] Lowe, D.G.: Object Recognition from Local Scale-Invariant Features, *Proceedings of the International Conference on Computer Vision ICCV*, Corfu, 1999, 1150–1152.
- [96] Lowe, D.G.: Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* 60(2), 2004, 95–97.
- [97] Lowe, D.G.: Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image, US Patent 6,711,293 (March 23, 2004). Provisional application filed March 8, 1999. Assignee: The University of British Columbia.
- [98] Maitra, S.: Moment invariants, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Press, 1979, 697.
- [99] Matthies, L., Szeliski, R., Kanade, T.: Incremental Estimation Of Dense Depth Maps From Image Sequences, *Proceedings of Computer Vision and Pattern Recognition*, 1988, 366–368.

- [100] Mayr, E.: *Toward a New Philosophy of Biology: Observations of an Evolutionist*, Belknap, Cambridge, MA, 1988.
- [101] Mees, C.E.K., James, T.H.: *The Theory of the Photographics Process*, Macmillan, UK, 1966.
- [102] Megyesi, Z., Chetverikov, D.: Affine Propagation for Surface Reconstruction in Wide Baseline Stereo, *Proceedings of ICPR 2004*, Cambridge, UK, 2004, 1–4.
- [103] Megyesi, Z., Chetverikov, D.: Affine dense matching for wide baseline stereo, *Proceedings of Grafika 2003*, Budapest, 2003, 109–114.
- [104] Melzer, T., Briese, Ch.: Extraction and Modeling of Power Lines from ALS Point Clouds, *Proceedings of 28th Workshop Austrian Association for Pattern Recognition*, Hagenberg, 2004, 47–54.
- [105] Mikolajczyk, K., Schmid, C.: An affine invariant interest point detector, *Proceedings of the 7th European Conference on Computer Vision*, Copenhagen, Denmark, 2002, 128–142.
- [106] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors, *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR03)*, 2003, 258–259.
- [107] Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points, *International Conference on Computer Vision*, Vancouver, Canada, 2001, 525–531.
- [108] Mikolajczyk, K., Schmid, C.: Scale and Affine Invariant Interest Point Detectors, *International Journal of Computer Vision* 60(1), Kluwer Academic Publishers, 2004, 63–68.
- [109] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schafalitzky, F., Kadir, T., Van Gool, L.: A Comparison of Affine Region Detectors, *International Journal of Computer Vision* 65(1/2), 2005, 44, 46.

- [110] Mindru, F., Moons, T., Van Gool, L.: Recognizing Color Patterns Irrespective of Viewpoint and Illumination, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 1*, Fort Collins, Colorado, 1999, 368–373.
- [111] Model of stereoscope invented in 1839 by Sir Charles Wheatstone: <http://www.geocities.com/CapeCanaveral/8341/stereogr.htm>, See, also, <http://www.kcl.ac.uk/depsta/iss/archives/collect/1wh20-0.html>
- [112] Modersitzki, J., Haber, E.: Cofir: Coarse and Fine Image Registration, *L.T. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, Real-Time PDE-Constrained Optimization*, 2004, 4, 5, 14.
- [113] Montesinos, P., Gouet, V., Deriche, R.: Differential Invariants For Color Images, *Fourteenth International Conference on Pattern Recognition 1*, Brisbane, Australia, 1998, 16–20.
- [114] Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units, *Neural Computation 1*(2), 1989, 281–294.
- [115] Orłowska, E.: Semantics of Vague Concepts, *Applications of Rough Sets. Institute for Computer Science, Polish Academy of Sciences Report 469*, 1982.
- [116] Pal, N.R., Pal, S.K.: A review of image segmentation techniques, *Pattern Recognition 26*(9), 1993, 1277–1294.
- [117] Paton, R.C.: Principles of genetics, *T. Back, D.B. Fogel, Z. Michalewicz (Eds.)*, Handbook of Evolutionary Computation, Oxford University Press, Oxford, UK, 1997.
- [118] Pavel, M.: *Fundamentals of Pattern Recognition*, 2nd Edition, Marcel Dekker, Inc., NY, 1993.
- [119] Pawlak, Z., Peters, J.F., Skowron, A., Suraj, Z., Ramanna, S., Borkowski, M.: Rough Measures, Rough Integrals, And Sensor Fusion, *M. Inuiguchi, S. Hirano, S. Tsumoto (Eds.)*, Rough Set Theory and Granular Computing, Studies in Fuzziness and Soft Computing, 125, Physica Verlag, Berlin, 2003, 263–272.

- [120] Pawlak, Z., Peters, J.F., Skowron, A., Suraj, Z., Ramanna, S., Borkowski, M.: Rough Measures And Rough Integrals, *S. Hirano, M. Inuiguchi, S. Tsumoto (Eds.)*, Lecture Notes in AI, 2002.
- [121] Pawlak, Z., Peters, J.F., Skowron, A., Suraj, Z., Ramanna, S., Borkowski, M.: Rough Measures: Theory And Applications, *S. Hirano, M. Inuiguchi, S. Tsumoto (Eds.)*, Proc. of Int. Workshop on Rough Set Theory and Granular Computing (RSTGC'01), Matsue, Shimane, Japan, 2001, 177–184.
- [122] Pawlak, Z.: Rough sets, *International J. Comp. Inform. Science*, 1982, 341–356.
- [123] Pawlak, Z.: Rough sets and data analysis, *Fuzzy Systems Symposium, Soft Computing in Intelligent Systems and Information Processing*, Kenting Taiwan, 1996, 1–6.
- [124] Pawlak, Z.: Why rough sets?, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems 2*, New Orleans, LA, USA, 1996, 738–743.
- [125] Pawlak, Z.: Classification of Objects by Means of Attributes, *Institute for Computer Science*, Polish Academy of Sciences Report 429, 1981.
- [126] Pawlak, Z.: Rough Sets, *Institute for Computer Science*, Polish Academy of Sciences Report 431, 1981.
- [127] Pawlak, Z.: Rough Sets. Theoretical Reasoning about Data, *Theory and Decision Library, Series D: System Theory, Knowledge Engineering and Problem Solving 9*, Kluwer Academic Pub., Dordrecht, 1991.
- [128] Perona, P., Malik, J.: Scale-Space and Edge Detection Using Anisotropic Diffusion, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(7), 1990, 631.
- [129] Personnaz, M., Sturm, P., Devernay, F.: Design of a Library for Dense Matching, INRIA Report No. 0278, 2003, 7, 23.

- [130] Peters, J.F., Gunderson, D., Henry, C.: Approximation spaces in actor-critic learning with stopping time, *International Journal of Hybrid Intelligent Systems*, 2007, to appear.
- [131] Peters, J.F., Skowron, A., Suraj, Z., Rzasa, W., Borkowski, M.: Clustering: A Rough Set Approach To Constructing Information Granules, *Z. Suraj (Ed.), Soft Computing and Distributed Processing*, Rzeszów, Poland, 2002, 57–61.
- [132] Peters, J.F., Skowron, A., Synak, P., Ramanna, S.: Rough sets and information granulation, *Bilgic, T., Baets, D., Kaynak, O. (Eds.), Tenth Int. Fuzzy Systems Assoc. World Congress IFSA*, LNAI 2715, Istanbul, Turkey, Heidelberg: Springer-Verlag, 2003, 370–377.
- [133] Peters, J.F.: Approximation spaces for hierarchical intelligent behavioral system models, *B.D.-Keplicz, A. Jankowski, A. Skowron, M. Szczuka (Eds.), Monitoring, Security and Rescue Techniques in Multiagent Systems, Advances in Soft Computing*, Heidelberg: Physica-Verlag, 2004, 13–30.
- [134] Peters, J.F.: Approximation space for intelligent system design patterns, *Engineering Applications of Artificial Intelligence*, 17(4), 2004, 1–8.
- [135] Peters, J.F., Ahn, T.C., Borkowski, M.: Obstacle Classification by a Line-Crawling Robot: A Rough Neurocomputing Approach, *Rough Sets and Current Trends in Computing*, LNAI 2475, Springer-Verlag, Berlin, 2002, 594–601.
- [136] Peters, J.F., Ahn, T.C., Borkowski, M., Degtyaryov, V., Ramanna, S., Line-Crawling Robot Navigation: A Rough Neurocomputing Approach, *D. Maravall, D. Zhou (Eds.), Fusion of Soft Computing and Hard Computing Techniques for Autonomous Robotic Systems*, Studies in Fuzziness and Soft Computing, J. Kacprzyk (Ed.), Berlin: Physica-Verlag, 2002.
- [137] Peters, J.F., Skowron, A., Suraj, Z., Borkowski, M., Rzasa, W.: Measures Of Inclusion And Closeness Of Information Granules, *Rough Sets and Current Trends in Computing*, LNAI 2475, Springer-Verlag, Berlin, 2002, 300–307.

- [138] Peters, J.F., Borkowski, M.: K-means indiscernibility relation over pixels, *S. Tsumoto, R. Slowinski, J. Komorowski, J.W. Grzymala-Busse*, LNAI 3066, Berlin, Springer-Verlag, 2004, 580–535.
- [139] Peters, J.F.: Rough ethology, *Transactions on Rough Sets III*, 2005, 153–174.
- [140] Peters, J.F., Henry, C.: Reinforcement learning with approximation spaces, *Fundamenta Informaticae 69 submitted*.
- [141] Peters, J.F., Borkowski, M., Henry, C., Lockery, D.: Monocular vision system that learns with approximation spaces, *Ella, A., Lingras, P., Slezak, D., Suraj, Z.: Rough Set Computing: Toward Perception Based Computing*, Idea Group Publishing, Hershey, PA, 2006, 1–22.
- [142] Peters, J.F., Henry, C., Lockery, D., Borkowski, M., Gunderson, D., Ramanna, S.: Line-Crawling Bots That Inspect Electric Power Transmission Line Equipment, *Autonomous Robots and Agents (ICARA06)*, 2006.
- [143] Petrou, M., Bosdogianni, P.: *Image Processing The Fundamentals*, John Wiley & Sons, Ltd., 2000, 310.
- [144] Polkowski, L.: *Rough Sets. Mathematical Foundations*, Heidelberg: Springer-Verlag, 2002.
- [145] Polkowski L., Skowron, A.: *Rough Sets in Knowledge Discovery 2*, Studies in Fuzziness and Soft Computing 19, Heidelberg: Springer-Verlag, 1998.
- [146] Pratt, W.K.: *Digital Image Processing*, Third Edition, John Wiley & Sons, Inc., 2001, 615, 625–527.
- [147] Reyes-Lozano, L., Bayro-Corrochano, E.: Simultaneous Uncalibrated Multiple View Reconstruction of Points, Lines, Quadrics and Cameras, *Proceedings of the 3rd International Conference on Computer Vision, Pattern Recognition & Image Processing*, Cary, North Carolina, USA, 2003.

- [148] Rosenfeld, A.: Image pattern recognition, *Proceedings of the IEEE* 69(5), 1981, 596–605.
- [149] Roth, G., Whitehead, A.: Using Projective Vision To Automate The Registration Step In Model Building, *Proceedings of Vision Interface*, 2000, 225–232.
- [150] Sahoo, P.K., Soltani, S., Wong, A.K.C.: A survey of thresholding techniques, *Computer Vision, Graphics and Image Processing* 41, 1988, 233–260.
- [151] Schölkopf, B., Smola, A.J.: *Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, Cambridge, MA 2002.
- [152] Schaffalitzky, F., Zisserman, A.: Multi-view Matching for Unordered Image Sets, or How Do I Organize My Holiday Snaps?, *ECCV*, 2002, 414–420.
- [153] Schmid, C., Mohr, R., Bauckhage, Ch.: Evaluation of Interest Point Detectors, *International Journal of Computer Vision* 2(37), 2000, 159, 160, 164, 166, 172–199
- [154] Schmid, C., Mohr, R., Bauckhage, Ch.: Comparing and Evaluating Interest Points, *Proceedings of the Sixth International Conference on Computer Vision*, Bombay, 1998, 230–235.
- [155] Schmid, C., Mohr, R.: Local greyvalue Invariants for Image Retrieval, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5(19), 1997, 530.
- [156] Schmid, C., Mohr, R.: Matching By Local Invariants, Technical Report, INRIA, August 1995, 8–11.
- [157] Schmid, C., Mohr, R.: Combining Greyvalue Invariants With Local Constraints For Object Recognition, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, 1996, 872–874.

- [158] Seemann, T.: *Digital Image Processing Using Local Segmentation*, Ph.D. Dissertation, supervisor: Peter Tischer, Faculty of Information Technology, Monash University, Australia, April 2002.
- [159] Shelton, B.E, Hedley, N.R.: Using Augmented Reality For Teaching Earth-Sun Relationships To Undergraduate Geography Students, Augmented Reality Toolkit, *The First IEEE International Workshop*, Darmstadt, Germany, 2002, 1–3.
- [160] Silva, C.A. dos S.: *3D Motion and Dense Structure Estimation: Representations for Visual Perception and the Interpretation of Occlusions*, Ph.D. Dissertation, Instituto Superior Técnico, Universidade Técnica de Lisboa, Spain, 2001.
- [161] Skowron, A.: Rough sets and vague concepts, *Fundamenta Informaticae XX*, 2004, 1–15.
- [162] Skowron, A., Stepaniuk, J.: Modelling complex patterns by information systems, *Fundamenta Informaticae XXI*, 2005, 1001–1013.
- [163] Skowron, A., Stepaniuk, J.: Information granules and approximation spaces, *Proceedings of the 7th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU98)*, Paris, 1998, 1354–1361.
- [164] Skowron, A., Stepaniuk, J.: Generalized approximation spaces, *Lin, T.Y., Wildberger, A.M. (Eds.), Soft Computing, Simulation Councils*, San Diego, 1995, 18–21.
- [165] Skowron, A., Swiniarski, R., Synak, P.: Approximation spaces and information granulation, *Transactions on Rough Sets III*, 2005, 175–189.
- [166] Smith, P., Sinclair, D., Cipolla, R., Woody, K.: Effective Corner Matching, *Proceedings of BMVC'98 2*, Southampton, UK, 1998, 545–556.

- [167] Sobel, I.E.: *Camera Models And Machine Perception*, Ph.D. Dissertation, Stanford University, Oalo Alto, California, 1970.
- [168] Stepaniuk, J.: Approximation spaces, reducts and representatives, In [145], 109–126.
- [169] Strobl, K., Sepp, W., Fuchs, S., Paredes, C., Arbter, K.: Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
- [170] Suk, T., Flusser, J.: Projective Moment Invariants, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26, 2004, 1364.
- [171] Sun, C.: Multi-Resolution Rectangular Subregioning Stereo Matching Using Fast Correlation and Dynamic Programming Techniques, CMIS Report No. 98/246, Macquarie University, 1998, 3.
- [172] Sun, C.: Multi-Resolution Stereo Matching Using Maximum-Surface Techniques, *Digital Image Computing: Techniques and Applications*, Perth, Australia, 1999, 196.
- [173] Szczuka, M.S., Son, N.H.: Analysis of image sequences for unmanned aerial vehicles, *M. Inuiguchi, S. Hirano, S. Tsumoto (Eds.), Rough Set Theory and Granular Computing*, Springer-Verlag, Berlin, 2003, 291–300.
- [174] Tang, L., Tsui, H.T., Wu, C.K.: Dense Stereo Matching Based on Propagation with a Voronoi Diagram, *Indian Conference on Computer Vision*, 2002.
- [175] Telle, B., Aldon, M.-J.: Interest Points Detection In Color Images, *IARP Workshop on Machine Vision Applications*, Nara, Japan, 2002, 550.
- [176] Ter Haar Romeny, B.M.: Introduction To Scale-space Theory: Multiscale Geometric Image Analysis, Technical Report ICU-96-21, Utrecht University, 1996, 9–10, 15.

- [177] Torr, P.H.S., Criminisi, A.: Dense Stereo Using Pivoted Dynamic Programming, *Cardiff Proceedings of British Machine Vision Conference (BMVC)*, 2002, 1–3, 798.
- [178] Torr, P.H.S., *Motion Segmentation And Outlier Detection*, Ph.D. Dissertation, University of Oxford, Engineering Dept., 1995, 1–7, 32–34, 242–243.
- [179] Triggs, B., McLauchlan, P., Hartley, R., Fitzgibbon, A.: *Bundle Adjustment A Modern Synthesis*, *Vision Algorithms: Theory & Practice*, B. Triggs, A. Zisserman & R. Szeliski (Eds.) LNCS 1883, Springer-Verlag, 2000, 1–5.
- [180] Trim, D.W.: *Calculus And Analytic Geometry*, Addison-Wesley Publishing Company, Inc., 1983, 500–501, 514–520, 526–528, 535–537, 539–540.
- [181] Tuytelaars, T., Van Gool, L.: Matching Widely Separated Views Based on Affine Invariant Regions, *Kluwer Academic Publishers* 59(1), Hingham, MA, USA, 2004, 61.
- [182] Urbanek, M., Horaud, R., Sturm, P.: Calibration of Digital Amateur Cameras, INRIA Report No. 4214, 2001, 24.
- [183] Vedaldi, A.: A Lightweight C++ Implementation Of David Lowe’s Scale Invariant Feature Transforms, <http://www.cs.ucla.edu/~vedaldi>
- [184] Vosselman, G., Dijkman, S.: 3D Building Model Reconstruction From Point Clouds And Ground Plans, *International Archives of Photogrammetry and Remote Sensing XXXIV-3/W4*, Annapolis, MD, 2001, 22–24.
- [185] Weickert, J.: *Anisotropic Diffusion in Image Processing*, Ph.D. Dissertation, Universität Kaiserslautern, 1997, 12–13, 20–21.
- [186] Weickert, J., Haar, B.M., Viergever, M.A.: Efficient and Reliable Schemes for Nonlinear Diffusion Filtering, *IEEE Transactions on Image Processing*, 1998, 399.

- [187] Weickert, J., Ishikawa, S., Imiya, A.: Scale-space has been Discovered in Japan, Technical Report 18, Department of Computer Science, University of Copenhagen, 1997, 162.
- [188] Weisstein, E.W.: Spherical Coordinates, from MathWorld – A Wolfram Web Resource. <http://mathworld.wolfram.com/SphericalCoordinates.html>
- [189] Wheatstone, C.: Contributions to the physiology of vision—part the first. On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London* 128, 1838, 371–394.
- [190] Wheatstone, C.: Contributions to the physiology of vision—part the second. On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London* 142, 1852, 1–17.
- [191] Witkin, A.P.: Scale-Space Filtering: A New Approach To Multi-Scale Description, *Proceedings of ICASSP*, San Diego, CA, 1984, 39, A.1.2.
- [192] Wolf, P.R.: *Elements of Photogrammetry (With Air Photo Interpretation and Remote Sensing)*, McGraw-Hill, 1974, 27–31, 41–46.
- [193] Xin, K., Lim, K.B., Hong G.S.: A Scale-Space Filtering Approach For Visual Feature Extraction, *Pergamon Pattern Recognition* 28(2), Elsevier Science Ltd, 1995, 1145–1146.
- [194] Yang, G.Z., Gillies, D.F.: Computer Vision Lecture Notes, Department of Computing, Imperial College, UK, 2001, <http://www.doc.ic.ac.uk/~gzy>
- [195] Zhang, Z.: Determining the Epipolar Geometry and Its Uncertainty: A review, *International Journal of Computer Vision* 27(2), 1998, 161, 164, 166–168, 171–172, 191.
- [196] Zhang, Y.J.: Evaluation and comparison of different segmentation algorithms, *Pattern Recognition Letters* 18, 1997, 963–974.

- [197] Zhang, C., Fraser, C.S.: Automated registration of high resolution satellite imagery for change detection, Research Report, Department of Geomatics, University of Melbourne, 2003
- [198] Zhang, Y.J., Gerbrands, J.J.: Objective and quantitative segmentation evaluation and comparison, *Signal Processing* 39, 1994, 43–54.
- [199] Zhang, Z., Loop, Ch.: Estimating the Fundamental Matrix by Transforming Image Points in Projective Space, *Computer Vision and Image Understanding* 82(2), 2001, 174-180.
- [200] Zhang, Z., Deriche, R., Faugeras, O., Luong, Q.: A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry, INRIA Report 2273, 1994.
- [201] Zitová, B., Flusser, J.: Image Registration Methods: A Survey, *Image and Vision Computing* 21, 2003, 982.
- [202] Zucchelli, M.: *Optical Flow Based Structure From Motion*, Ph.D. Dissertation, Royal Institute of Technology, Numerical Analysis and Computer Science, Comp. Vision and Active Perception Laboratory, Stockholm, 2002, 4.
- [203] CCIR Rec. 601-2 Encoding Parameters Of Digital Television For Studios (1990), Recommendations of the CCIR, Volume XI - Part 1 Broadcasting Service (Television), 1990, 99.
- [204] <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3710&objectType=FILE>
- [205] Concise Oxford Dictionary (COD10) on CD-ROM, Tenth Edition
- [206] JSankey Windows Software,
<http://www.jsankey.com/software/csl/csl.htm>
- [207] LocMetrics - C#, C++, Java, and SQL,
<http://www.locmetrics.com/index.html>

- [208] Math Forum at Drexel Univeristy,
<http://mathforum.org/library/drmath/view/51755.html>
- [209] Matlab, <http://www.mathworks.com>
- [210] Mesh Factory, <http://www.meshfactory.com>
- [211] Overview of the MPEG-4 Standard,
<http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm>
- [212] Royal Ontario Museum,
<http://www.rom.on.ca/exhibits/egypt/digitalgallery.php>
- [213] WITAS project homepage: <http://www.ida.liu.se/ext/witas/>

Notation

P_i	i -th point in the 3D space
p_i	i -th point in the 2D image plane
p'_i	rotated point p_i
\bullet_{ki}	$k = 1, 2; i = 1, \dots, n; i$ -th point from k -th image
N	number of point matches
f	focal length of both cameras, see 14
$[v_x \ v_y \ v_z]^T$	point or vector in the 3D space
L_k	lens centre of the k -th camera, see 14
\mathcal{P}	principal point, see 14
θ	azimuth
ϕ	altitude
γ	rotation around the principal ray
R	distance from the origin to given point
ϑ	pan
φ	tilt
S_{ki}	parametrization of the i -th ray from the k -th view, see 154
τ_{ki}	parametrization variable for line S_{ki} , see 154
$\hat{\tau}_{ki}$	parametrization for the closest approach point A_{ki} , see 155
D_i	cost function for the i -th rays, see 155
$R_\alpha, R_\beta, R_\gamma$	rotation matrices
T_{ki}	$T_{ki} = p'_{ki} - L'_k$ directional vector for projecting ray S_{ki} , see 153

$V_1 \times V_2$	cross product between vectors V_1 and V_2
U_i	$U_i = T_{1i} \times T_{2i}$ common perpendicular vector to projecting rays S_{1i} and S_{2i} , see 154
W_i	$W_i = S_{1i}(t_{1i}) - S_{2i}(t_{2i})$ vector from point on line S_{2i} to point on line S_{1i} , see 154
A_{ki}	closest approach point for S_{ki} , A_{1k} is the closest point from S_{1i} to the line S_{2i} , see 155
\mathbb{S}	search space for the DirectGS algorithm, see 164
M	set of points from one image
\mathbb{M}	set of point matches
\mathbb{I}	initial guess of the azimuth and altitude
C	colour difference, see 63
O	overlap, see 59
A	angle of rotation, see 63
Rc	ratio of cardinalities, see 64
\mathbb{C}	domain of the colour differences, see 63
\mathbb{O}	domain of the overlap, see 59
\mathbb{A}	domain of the angle of rotation, see 63
\mathbb{RC}	domain of the ratio of cardinalities, see 64
Ω	subspace of \mathbb{R}^4 , $\Omega = \mathbb{C} \times \mathbb{O} \times \mathbb{A} \times \mathbb{RC}$, see 44
ξ	ideal solution for segment matching problem, see 45
Q_{n_i}	quantization algorithm, given in Algorithm 2 n_i - requested number of colours after quantization

Av_I	spatial colour averaging algorithm, given in Algorithm 3
$M_{3 \times 3}$	3 by 3 median filter
C_f	convexity factor, see 58
\bar{X}	mean value of expression X
\mathcal{Z}	ideal solution in segment matching in rough set approach, see 71
Ch	a chromosome (in Genetic Algorithm)
G	a set of genes
C_m	a codebook (in Lloyd quantization)
C_i, C_j	i -th or j -th segment colour
S_i, S_j	i -th or j -th segment
γ	partitioning of segment pairs resulting from GA
$G(\sigma)$	Gaussian kernel, see 99
$L(\sigma)$	an image convolved with a Gaussian kernel, see 99
L_i	first derivative of an image L , $i \in \{x, y\}$
L_{ii}	second derivative of an image L , $i \in \{x, y\}$
$DoG(\sigma)$	difference of Gaussian filter, see 103
H	the Harris corner detector, see 98
(η, ξ)	Gauge coordinates, see 116
∇	gradient operator
Δ	divergence operator

Glossary

altitude

An angle between a line connecting given point and the origin of the coordinate system and the projection of this line on the XZ plane measured in such a way that for positive Y values the angle is negative and for negative Y values the angle is positive. 148

approximation space

A system (U, N, ν) , where U is a universe of objects, N is a neighbourhood function and ν is an overlap function. 34

azimuth

An angle between a projection on the XZ plane of a line connecting given point and the origin of the coordinate system and the Z axis measured in such a way that for positive X values the angle is positive and negative X values the angle is negative. 148

baseline

Line connecting lens centres of two cameras. 27

camera calibration

A process of finding internal camera parameters and removing distortions from an image, caused by the fact that digital cameras do not conform to the pinhole camera model. 16

colour channel

A colour channel is one component from the set of given features in a particular colour model space. For example, in RGB Colour Model, there are three channels: Red, Green and Blue. 117

dense matching

Process of matching points from two images that makes it possible to find matches for all points in both images. 127

direct geometrical search (DirectGS)

Algorithm for determining the relative camera locations and orientations, which minimises the reprojection error in 3D space. 141

disparity

A difference of horizontal coordinates for a pair of matching pixels. 131

disparity map

A disparity map is an image in which each pixel value is a disparity of a corresponding pixel from the second image. 131

epipolar geometry

Epipolar geometry is an intrinsic projective geometry between two views (in this work, the focus is on 2D views of a 3D scene). 27

epipolar line

A projection of the epipolar plane onto camera's image plane. It is a line passing through two corresponding points from two given images. 27

epipolar plane

A plane containing the baseline and a point in 3D space. 27

epipole

A projection of one camera's lens centre on the other camera's image plane. It is a point where all epipolar lines for a given image cross. 27

external (extrinsic) camera parameters

Camera properties that depend on camera location and orientation in 3D space, *i.e.*, camera rotation and translation. 15

focal length

A distance between the image plane and the lens centre. 13

fundamental matrix

Relates image coordinates of corresponding points from two given images. 25

Hough transform

A technique for detecting lines in digital images. 19

image plane

Plane containing the information (image) recorded by the camera. 13

image registration

Process of finding correspondences between two images of the same scene taken at different angles (or times). 40

interest point detector

An algorithm extracting interest points from given image. These points have to be easy discernible from the background, have diverse surroundings and should be easily detectable in an image that undergoes some transformation. 97

internal (intrinsic) camera parameters

Camera properties that do not depend on camera location and orientation, *i.e.*, aspect ratio, magnification, coordinates of the principal point and skew. 15

lens centre

In the pinhole camera model, a point through which all projecting rays forming an image must pass. 147

lower approximation

A collection of blocks of sample elements that can be classified with full certainty as members of given set using the knowledge represented by given subset of probe functions. 33

pan

A rotation of the camera around the Z axis. A positive pan turns in the counter-clockwise direction. 150

pinhole camera model

A mathematical model describing the process of creating images using an ideal camera with zero aperture. 13

principal point

A point where the principal ray crosses with the image plane. 13

principal ray

A ray that is perpendicular to the image plane. 13

projecting ray

A straight line connecting a point in 3D space with its projection in an image plane. Each projecting ray passes through the lens centre. 13

projection matrix

Matrix used to convert 3D coordinates of a given point in 3D space to 2D coordinates of its corresponding point in the image plane. 15

quantization

In image processing, quantization is a method for converting an analog signal to a digital signal [46]. In this dissertation, the term quantization is used as

a mapping from a finite set of numbers (derived from an analog spectrum) to another finite set of numbers, where the cardinality of the destination set is smaller than the source set. 46

Random Sample Consensus (RANSAC)

Algorithm used for estimation of parameters of any system in presence of outliers. 5

rectification

Image rectification (epipolar rectification) is the process of determining a mapping for two given images such that corresponding epipolar lines become collinear and parallel to the horizontal image axes. 30

rough coverage

A measure of how well one set covers (intersects with) another set. 36

rough inclusion

A measure of the degree with which one set is included in another set. 36

rough set

A set X is considered rough if, and only if, $B^*X \setminus B_*X \neq \emptyset$, where B^*X, B_*X denote the upper and lower approximation of X , respectively. 31

scale space

A scale space is a sequence of images parametrized by a scale. 98

standard set

A set containing elements that represent a norm or standard used to evaluate sample sets of elements. 70

tilt

A rotation around a new X axis (after rotation around the Z axis). A positive tilt corresponds to the counter-clockwise direction. 150

uncertainty function

Defines a neighbourhood for each point from the universe U . 35

upper approximation

A collection of blocks of sample elements representing both certain and possibly uncertain knowledge about given set using the knowledge represented by given subset of probe functions. 33

Index

algorithm

- DirectGS, 141, 180, 181
- exhaustive, 168, 169
- gradient descent, 176
- Lloyd, 48
- matching, 40, 46, 56, 65, 68, 76, 131
- minimisation, 167

altitude, 142, 148–150, 153, 165, 166, 168,
170, 171, 175, 176, 181, 184

aperture, 11

approximation

- lower, 33, 34, 74, 87
- upper, 33, 34, 71–73, 79

attribute, 167, 170

average, 39, 51, 54, 66, 67, 82, 84, 106,
179, 180

azimuth, 142, 148–150, 153, 165, 166, 168,
170, 171, 175, 176, 181, 184

baseline, 27, 28

bicubic interpolation, 136

calibration, *see* camera - calibration

camera

- calibration, 7, 12, 13, 16, 142
- external parameters, 15, 16, 26, 129,
141, 179, 184
- extrinsic parameters, 15
- first, 27, 140, 153, 157, 158, 160, 164,
166, 169, 171, 172, 177
- internal parameters, 15–17, 142
- pinhole model, 10, 11, 13, 16, 17, 144
- second, 6, 153, 158, 159, 164–173,
177, 184

CCD, 13

centroid, 48, 67, 68

chromosome, 37, 79, 81–86, 88–94

codebook, 47, 48, 54

constraint

continuity, 134

epipolar, *see* epipolar - constraint

rank-2, 26

continuous, 119, 169, 170, 176

convexity, 56–58

- convolution, 19, 99, 114
- coordinate system
 - Cartesian, 10, 19, 148
 - gauge, 116
 - right-handed, 149
 - spherical, 148, 149, 167
- corneriness, 105
- derivatives, 99–101, 109, 113, 114, 116
- detector
 - DOG, 103
 - edge, 22, 23, 129, 130
 - Harris corner, 4, 5, 98, 101, 104–106, 125
 - Harris-Laplacian, 101
 - improved Harris, 107
 - interest point, 4, 98, 101, 104, 105, 107–109, 111, 112
 - multi-scale Harris, 109
- diffusion, 24, 25
- disparity, 132–135, 137, 138, 140
- disparity map, 128, 130, 132–135, 138, 183
- distance
 - Earth Mover, 124
 - Euclidean, 9, 122
 - Kolmogorov-Smirnov, 124
 - Mahalanobis, 8, 9, 111, 122
- eigenvalue, 109
- eight-neighbourhood, 104
- epipolar
 - constraint, 29
 - geometry, 1, 5, 10, 27, 29, 40, 129
 - line, 7, 26, 28–30, 180
 - plane, 27, 28
 - rectification, 30
- epipole, 28
- Epsilon tensor, 116
- error
 - estimation, 103
 - localization, 107
 - reprojection, 143
- Euler angle, 150, 151
- extrinsic camera parameters, *see* camera
 - external parameters
- feature extraction, 81
- filter

complex, 111, 120
 Gaussian, 99, 100, 103, 105
 Kalman, 6
 median, 53, 54
 Sobel, 22, 23
 steerable, 111, 113, 121
 first order warp, 21
 focal length, 11, 14, 16, 17, 142, 147, 158,
 165, 177
 function
 cost, 7, 143, 152, 156, 159, 165–171,
 175–177, 179, 182, 184
 exponential, 60
 fitness, 85
 Laplacian, 101
 overlap, 35, 36, 60, 62, 63
 gene, 37, 79, 81–83, 85, 91, 93
 genetic
 algorithm, 36, 37, 42, 44, 46, 79, 81–
 85, 87, 183
 operator, 85
 homogeneous coordinates, 9, 10
 Hough transform, 19, 127, 132, 183
 hypothesis, 65–69, 81, 88
 image
 derivatives, 114
 gradients, 113
 plane, 11, 13, 14, 16, 17, 28, 130, 132,
 144, 146, 147, 181
 quantization, 48, 54
 rectification, 30, 131
 registration, 4–6, 40, 98, 104, 105,
 108
 segmentation, 19, 42, 49, 51
 segments, 31, 32, 36, 42–44, 46, 51,
 57, 59, 64, 65, 70–72, 74, 82–85,
 88, 93, 94, 183
 undistorted, 17
 implemented, 46, 83, 85, 101, 109, 179,
 181
 indiscernible, 32–34, 72, 86, 97
 indistinguishable, 91
 insulator, 3, 123, 124
 interpolate, 113, 114
 intrinsic camera parameters, *see* camera
 - internal parameters

invariants
 differential, 5, 111, 114, 115, 120
 projective, 98
 K-means clustering, *see* algorithm - clustering
 kernel, 98–100, 106, 113, 114
 laterally, 1, 6, 30
 least-squares method, 1
 lens centre, 14, 15, 28, 146–148, 153, 157,
 164, 169–171, 173
 local jet, 114, 115, 132
 mapping, 15, 21, 28, 30, 47, 49
 inverse, 17
 matching
 coarse, 7, 29, 40, 46, 126
 dense, 7, 27, 29, 30, 40, 41, 127–131,
 134, 138, 140, 141, 183, 184
 pixel, 4, 40, 46, 94, 131
 point, 7, 28, 40, 96, 97, 120, 123–125,
 129
 rough, 71
 segment, 61, 79, 93, 94, 141, 183
 template, 131
 matrix
 calibration, 15
 fundamental, 5, 6, 25–27, 29, 141,
 143, 180, 184
 identity, 9
 non-singular, 8, 9
 projection, 16
 rotation, 15, 151, 152, 172
 second moment, 105, 106, 109, 110
 singular, 8, 26
 transformation, 21
 mean ray, 170–173, 177
 Mecerary curve, 128
 minimum
 global, 165, 170, 176, 180, 181
 local, 176, 180
 moment
 central, 117–119
 geometric, 117
 invariant, 117, 119
 invariants, 111, 119, 120
 monocular, 2, 3

noise, 49, 128, 138, 140, 184
 noiseless, 128
 nonlinear diffusion, *see* diffusion
 norm operator, 156
 normal distribution, 9
 operator
 crossover, 37, 85, 88, 90, 91
 genetic, *see* genetic - operator
 pan, 142, 150, 152, 153, 157, 164, 166–
 168, 170–172, 174
 perspective
 transformation, *see* transformation -
 perspective
 photogrammetry, 10, 12
 point matches, 26, 147, 165, 180, 184
 polynomial, 115
 powerset, 35
 principal
 point, 11, 14–16, 143, 146, 148
 ray, 11, 14, 30, 140, 146, 148, 150,
 152, 157, 164, 166, 167, 176
 processor, 179, 181
 projecting ray, 13, 14, 153, 169
 projective invariants, *see* invariants - pro-
 jective
 quantization, 42, 47–52, 54
 quaternions, 150
 random, 66
 range-from-focus, 3
 RANSAC, 5
 rectification, *see* epipolar - rectification,
 see epipolar - rectification, *see* epipo-
 lar - rectification
 relation
 binary, 31, 32
 equivalence, 31, 32, 71, 88–90, 92
 indiscernibility, 35
 tolerance, 31, 86, 88, 89, 91, 92
 RGB, 18, 22, 40, 63
 robust, 53, 111, 124–127
 rough
 coverage, 36, 87, 88, 93–95
 inclusion, 36
 sets, 2, 4, 31, 33, 35, 41, 44, 79, 90
 Segment, 19

SIFT descriptor, 112, 121–123
 skew, 15, 16
 Sobel filter, *see* filter - Sobel
 solution
 global, 180
 ideal, 45
 space
 approximation, 4, 31, 35, 36, 42, 43, 79, 87, 183
 line-disparity, 133
 scale, 98, 100, 101, 103, 104, 106
 solution, 156
 SSD, 124, 131, 136, 171
 standard set, 31, 71
 stereoscope, 1
 stereovision, 1, 6
 sub-pixel, 29, 107, 108, 136

 tiepoints, 21
 tilt, 142, 150, 152, 153, 157, 164, 166–168, 170–174
 transformation
 affine, 9, 108, 109, 121
 geometrical, 20, 77
 image, 96
 linear, 9
 perspective, 14, 57
 projective, 98, 109, 119, 120
 triangle property, 67, 125
 Voronoi diagram, 127
 warping, *see* first order warp
 Zero-Mean Normalized Cross Correlation, 39, 101, 124
 ZNCC, *see* Zero-Mean Normalized Cross Correlation