Three-Dimensional Reconstruction and Layout Planning for Industrial Automation

Nikolaos Georgis

Submitted in
Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy
of
Surrey University

Department of Electronic and Electrical Engineering
University of Surrey
Guildford, Surrey GU2 5XH, U.K.

December, 1994
Abstract

This thesis is concerned with the mathematical aspects related to the optimal cutting of an object whose three-dimensional shape has been accurately and robustly reconstructed using appropriately developed computer-vision tools.

First, a brief introduction to various one- and two-dimensional packing problems is presented. The Constrained Rectangle Packing problem which allows for defects to be modeled is then formulated and an efficient algorithm for solving it is presented. The two-stage stock-cutting problem according to which a set of rectangular pieces of prespecified dimensions are to be cut from a general shape object with general shape holes or defective regions is then investigated. It is shown how mathematical morphological operators can be used in order to determine the optimal shifting for a given cutting pattern and proved that the problem of obtaining the optimal cutting pattern is \textit{NP}-hard. However, the optimal solution to the unconstrained problem using mathematical programming is proposed. For the general problem good sub-optimal solutions are obtained using the technique of simulated annealing.

Stereo-vision techniques are then employed for the accurate shape determination of the object to be cut. A three-dimensional reconstruction technique based on projective geometry is formally analysed and guidelines for its robust application are given. Finally, emphasis is placed on the correspondence problem, which becomes very difficult in the case of non-coplanar features and cameras set at 90 degrees from each other. It is shown how to cast the problem into an optimisation framework and a branch and bound algorithm is used in order to obtain the optimal solution. For increased robustness a Hough-Transform-like algorithm is also suggested.

Both synthetic and real experimental results are presented throughout the thesis in order to illustrate the validity and usefulness of the proposed algorithms.
Στή γιαγιά Ειρήνη και τόν παππού Γιώργο
(To my grandparents Irene and George)
Acknowledgments

I would like to express my gratitude to all the folks in the Vision, Speech and Signal Processing Group who have generously given their guidance, help and encouragement during the course of my research.

I would particularly like to thank my supervisors Doctor Maria Petrou and Professor Josef Kittler not only for providing the opportunity for this research but also for their continuous support, advice and encouragement.

The work described in this thesis was part of collaborative research between University of Surrey, University of Genova, Technostone and Computer Recognition Systems Ltd for the EEC BRITE-EURAM Programme (contract number BREU-CT91-0496). The financial support of the BRITE-EURAM project number BE 4097 and an SERC studentship are gratefully acknowledged.
## Contents

1 Introduction ............................................................... 1  
1.1 The Stock-Cutting Problem .................................... 1  
1.2 Scope of this work .................................................. 3  
1.3 Outline of the thesis ............................................... 4  

2 Packing Problems ...................................................... 10  
2.1 Introduction .......................................................... 10  
2.2 Bin Packing .......................................................... 11  
  2.2.1 One-Dimensional Bin Packing ............................ 12  
  2.2.2 The Knapsack Problem .................................... 13  
  2.2.3 Two-Dimensional Bin Packing or Rectangle Packing ... 14  
  2.2.4 Conclusions ................................................... 16  
2.3 Floorplanning ......................................................... 16  
  2.3.1 Introduction ................................................... 17  
  2.3.2 Literature survey ............................................ 17  
    2.3.2.1 Analytical algorithms ............................... 17  
    2.3.2.2 Graph-theoretic approaches ....................... 19  
  2.3.3 Polish expressions .......................................... 20  
    2.3.3.1 Optimisation using Simulated Annealing ....... 22  
    2.3.3.2 Genetic Algorithms ................................. 23  
  2.3.4 Experimental results and Comparison ................. 26  
  2.3.5 Conclusions ................................................... 26  
2.4 Rectangle Packing ................................................. 27  
  2.4.1 Introduction ................................................... 27  
  2.4.2 Representation ............................................... 28  
    2.4.2.1 Solution representation ............................ 28  
    2.4.2.2 Calculation of the cost function .................. 29
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.3 Optimisation</td>
<td>31</td>
</tr>
<tr>
<td>2.4.4 Experimental results</td>
<td>33</td>
</tr>
<tr>
<td>2.4.5 Conclusions</td>
<td>36</td>
</tr>
<tr>
<td>2.5 Conclusions</td>
<td>37</td>
</tr>
<tr>
<td>3 Generalised Stock-Cutting Problem</td>
<td>41</td>
</tr>
<tr>
<td>3.1 Problem Formulation</td>
<td>42</td>
</tr>
<tr>
<td>3.2 Determination of optimal translation</td>
<td>45</td>
</tr>
<tr>
<td>3.2.1 Morphological Operations</td>
<td>45</td>
</tr>
<tr>
<td>3.2.2 Optimal translation for one type of rectangles</td>
<td>48</td>
</tr>
<tr>
<td>3.2.3 Optimal translation for a given cutting pattern</td>
<td>50</td>
</tr>
<tr>
<td>3.3 Searching for the optimal pattern</td>
<td>52</td>
</tr>
<tr>
<td>3.3.1 Integer Programming Formulation</td>
<td>52</td>
</tr>
<tr>
<td>3.3.2 Simulated Annealing</td>
<td>55</td>
</tr>
<tr>
<td>3.4 Choosing the rotation angle ( \theta )</td>
<td>56</td>
</tr>
<tr>
<td>3.5 Experimental Results</td>
<td>57</td>
</tr>
<tr>
<td>3.6 Conclusions</td>
<td>59</td>
</tr>
<tr>
<td>4 Three Dimensional Reconstruction</td>
<td>62</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>62</td>
</tr>
<tr>
<td>4.2 The Projective Geometry Approach</td>
<td>64</td>
</tr>
<tr>
<td>4.2.1 Introduction to Projective Geometry</td>
<td>64</td>
</tr>
<tr>
<td>4.2.1.1 Cross-ratio of four points</td>
<td>64</td>
</tr>
<tr>
<td>4.2.1.2 Cross-ratio of a pencil of four coplanar lines</td>
<td>65</td>
</tr>
<tr>
<td>4.2.1.3 Projective Coordinates on a plane</td>
<td>66</td>
</tr>
<tr>
<td>4.2.1.4 Cross-ratio of a pencil of four planes</td>
<td>67</td>
</tr>
<tr>
<td>4.2.1.5 Preservation of the cross-ratio by perspective projection</td>
<td>68</td>
</tr>
<tr>
<td>4.2.2 Projective Reconstruction using planar points</td>
<td>69</td>
</tr>
<tr>
<td>4.2.2.1 Finding the viewing line</td>
<td>69</td>
</tr>
<tr>
<td>4.2.2.2 Reconstructing the point ( P )</td>
<td>70</td>
</tr>
<tr>
<td>4.3 Experimental Results</td>
<td>70</td>
</tr>
<tr>
<td>4.3.0.3 Finding the viewing line ( OP )</td>
<td>72</td>
</tr>
<tr>
<td>4.3.0.4 Finding the other viewing lines</td>
<td>73</td>
</tr>
<tr>
<td>4.3.0.5 Recovering the position of the line segment ( PQ )</td>
<td>73</td>
</tr>
<tr>
<td>4.3.0.6 Recovering the shape of a stone</td>
<td>73</td>
</tr>
<tr>
<td>4.4 Conclusions</td>
<td>78</td>
</tr>
</tbody>
</table>
## CONTENTS

5 Projective Geometry Based Reconstruction 82
5.1 Error Analysis .................................................. 83
  5.1.1 Error in the calculation of the projective coordinates ........... 83
    5.1.1.1 How many cross-ratios? ......................................................... 88
  5.1.2 From the projective coordinates to the 3D coordinates of \( P_1 \) .... 90
    5.1.2.1 Transforming to the local coordinate system ........... 90
    5.1.2.2 Calculating \( P_1 \) .................................................. 93
    5.1.2.3 Back to the world coordinate system ........... 97
  5.1.3 Determining the 3-D coordinates of \( P \) ........... 97
  5.2 Implications of the error analysis ................... 100

6 Solving the Correspondence Problem 103
6.1 Casting the problem into an optimisation one ................... 104
  6.1.1 The epipolar constraint .................................................. 104
  6.1.2 The Fundamental matrix .................................................. 105
  6.1.3 Casting Matching into Optimisation ................... 105
6.2 Deriving the cost function .................................................. 106
  6.2.1 Fitting a line to a set of points .................................................. 106
  6.2.2 Finding the point of intersection of \( N \) lines ................... 108
6.3 Exploring the configuration space .................................................. 108
  6.3.1 Exhaustive Search .................................................. 110
  6.3.2 From Permutations to Combinations ................... 110
  6.3.3 Branch and Bound Algorithm .................................................. 112
  6.3.4 Hough Transform .................................................. 116
  6.3.5 Randomised Hough Transform .................................................. 116
6.4 Experimental Results .................................................. 118
  6.4.1 Synthetic Data .................................................. 118
  6.4.2 Real Data .................................................. 120
6.5 Discussion .................................................. 121

7 Conclusions 128
  7.1 Overview and major contributions of this work ................... 128
  7.2 Future Directions .................................................. 130

A Functions of \( n \) \( m \)-tuples of Random Variables 132
## List of Figures

1.1 A granite to be reconstructed in 3D is placed on a platform. ..................... 2

2.1 A perfect square. ........................................................................................................ 11

2.2 Linear Programming approach to the knapsack problem. .......................... 14

2.3 First Fit Decreasing Height 2D packing. ................................................................. 15

2.4 Slicing structures and Polish expressions. .............................................................. 21

2.5 Simulated Annealing Moves M1, M2 and M3. .................................................. 23

2.6 Crossover Operators CO1, CO2 and CO3. ............................................................. 25

2.7 Formation of the “LABR” string. ............................................................................. 28

2.8 An invalid configuration. .......................................................................................... 29

2.9 Compaction Algorithm. ............................................................................................ 31

2.10 The Move operator. .................................................................................................. 32

2.11 The Exchange operator. .......................................................................................... 33

2.12 Packing of 18 blocks. ............................................................................................. 33

2.13 Percentage of invalid configurations. ...................................................................... 35

2.14 Performance of Simulated Annealing.................................................................... 35

2.15 Experimental Results. ............................................................................................ 36

3.1 The 2-stage Stock Cutting Problem. ................................................................. 43

3.2 Pictorial Illustration of the Erosion Chain Rule. ................................................ 47

3.3 CPU times for erosion. ............................................................................................ 47

3.4 An example of determining the \( C_r \) matrices. ...................................................... 51

3.5 An example of a constraint matrix. ......................................................................... 55

3.6 Calculation of the area utilisation over angle. ..................................................... 57

3.7 Experimental results. ............................................................................................... 58

3.8 Investigation of the Rotation angle. ........................................................................ 58

3.9 Determination of the optimal translation and pattern. ........................................ 59
LIST OF FIGURES

4.1 The problem of 3D vision .......................................................... 63
4.2 Cross-ratio of four lines .......................................................... 65
4.3 Projective coordinates $k_1$, $k_2$ and $k_3$ of a point ..................... 67
4.4 Cross-ratio of a pencil of four planes ......................................... 68
4.5 The cross-ratio is preserved by perspective projection ................. 68
4.6 Each of the reference planes contains four reference points ........... 70
4.7 Setup for the experiment .......................................................... 71
4.8 Reconstruction of PQ ............................................................ 74
4.9 Drawing of the platform (all dimensions in mm) ......................... 75
4.10 First viewpoint ................................................................. 76
4.11 Second viewpoint ............................................................... 76
4.12 Third viewpoint ................................................................. 77
4.13 Fourth viewpoint ............................................................... 77
5.1 Determination of $k_2$ ........................................................... 84
5.2 Regions of instability for determination of $k$ ................................. 89
5.3 Reference points used for experiment ....................................... 90
5.4 Transforming from the Reference Plane Coordinate system to the World Coordinate system ............................................. 92
5.5 Determination of $AP_1$ given $k_2$ ............................................ 93
5.6 Regions of instability for determination of $P_{1w}$ ......................... 96
5.7 Regions of instability for determination of $P_{1y}$ ......................... 96
5.8 Worst case for reconstruction of $P_1$ ......................................... 96
5.9 Transforming $P'_i$ to the world coordinate system ....................... 98
5.10 Determination of the 3D coordinates of point $P$ ......................... 99
5.11 Setup of reference planes and points for minimum sensitivity to noise 101
6.1 Matching features from left to right ....................................... 104
6.2 Determination of the point of intersection of 200 lines .................. 109
6.3 Determining correspondences ............................................... 111
6.4 All possible configurations .................................................... 118
6.5 A synthetic example for matching 30 points ............................... 119
6.6 Experimental results based on real images ................................. 122
6.7 Left image is transformed using the 4 reference points ................. 123
6.8 Illustration of the error amplification factor ............................... 124
List of Tables

2.1 Size of configuration spaces ................................................. 29
2.2 Simplex vs Topological Sorting .............................................. 34
3.1 Experimental results for two kinds of tiles ............................ 59
4.1 Coordinates of Reference points ................................. 72
4.2 The equations of the four viewing lines .................... 73
4.3 Experimental Results ............................................................ 74
4.4 Platform corner coordinates ........................................ 75
4.5 3D reconstruction of the stone .......................................... 78
5.1 The 2 independent cross-ratios ............................................. 91
6.1 Possible matches and cost evaluation ............................ 112
6.2 Probabilities for a given matching ................................. 118
6.3 The accumulator for the synthetic example .............. 120
6.4 Relative measurement of distances compared to the ground truth 123
"I made this letter longer than usual because I lack the time to make it shorter"

Pascal, Provincial Letters XVI
Chapter 1

Introduction

Bin-packing usually refers to the problem of fitting a collection of objects into well defined regions so that they do not overlap [6]. This can be viewed from an engineering point of view as making efficient use of material, space or time.

The applications of the bin-packing problem are numerous. Vehicle loading with a given weight limit or division of work into time shifts [6] are just two examples of everyday encountered bin-packing problems. Applications of bin-packing in parallel computers have also been published [20]. Scheduling a set of tasks on parallel systems in order to achieve the maximum system utilisation has been modeled as a two-dimensional rectangle-packing problem. By far the most important stimulus to the research on bin-packing has been the stock-cutting or lay-planning problem according to which some piece of material has to be cut to certain shapes so that the waste is minimised. Examples include cutting steel in aerospace or shipbuilding [16], cutting glass plates, leather cutting in shoe manufacturing, cutting of garment in the clothes industry, cutting wood plates to make furniture or paper board to make boxes. Unfortunately, the bin-packing problem is an example of a large scale optimisation problem which is known to be \( \mathcal{NP} \)-hard \(^1\) [10]. Nevertheless, due to its significant practical importance it has received great attention and efficient approximation algorithms for obtaining good sub-optimal solutions have been developed.

1.1 The Stock-Cutting Problem

In this thesis we study packing problems in general, with emphasis on stock-cutting applied to the problem of optimally cutting a granite block viewed by four cameras.

\(^1\)\( \mathcal{NP} \)-hard and \( \mathcal{NP} \)-complete are terms introduced in the early 1970's to symbolise a class of problems for which a polynomial-time algorithm to obtain the optimal solution is not likely to exist.
1.1. THE STOCK-CUTTING PROBLEM

Granite blocks are approximately $3 \times 2 \times 2$ m$^3$ and each takes 3–4 days to be cut into slabs using the most modern machines. As the process is so slow and expensive it is very important for the manufacturer that the optimal way in the sense of usable area of the cut slabs and utilisation of the cutting equipment is found for each particular block of granite.

For the purpose of inspection each block is placed on a specially constructed platform and viewed from four cameras around it, placed at approximately 90° angular distance from each other (figure 1.1). A factory worker marks with distinct markers the points on the surface of the stone that are excessively protruding or receding from the surface (usually the flattest side of the stone is placed at the bottom). The four cameras are placed robustly several meters away from the crane that moves the stones, and roughly so that each camera “sees” three sides of the stone and two sides of the platform on which the stone sits. The exact dimensions of the platform are known. The top plane of the block is viewed by all four cameras but any side plane is viewed by only two adjacent cameras. So, the problem we are interested in, is effectively trying to reconstruct an uneven surface which is viewed by two cameras, call them left and right cameras. In particular, our task prior to cutting is to automatically identify the most significant points on the surface of each side of the stone which have been marked by the worker, to solve the correspondence problem between the two cameras that see the same face, and find the 3D coordinate position of each point and reconstruct the 3D shape of the stone.
1.2. SCOPE OF THIS WORK

Three major areas of research are involved in addition to bin-packing: feature detection, feature matching and 3D object reconstruction within the factory floor environment which imposes significant constraints on the methods used. The novelty that these constraints bring to the problem is the necessity to perform 3D matching from a stereo pair at approximately 90° apart, the inability to use accurate and sensitive camera calibration aids in an environment where people and machines are meant to handle granite blocks rather than sensitive electronic equipment, the impracticality of using any calibration approach in a rapidly changing conditions of the environment (e.g. ambient temperature affecting camera parameters), and the requirement of the producer for 1% accuracy in the volume measurement of the block with minimum capital investment that excludes range finders etc.

1.2 Scope of this work

The purpose of this thesis is twofold in the sense that both the above mentioned problems, namely the stock-cutting and the three dimensional reconstruction problem, are investigated.

First, a clear presentation of packing and stock-cutting problems is given. Existing algorithms for obtaining the optimal or good sub-optimal solutions are in-depth analysed and compared. However, many questions remain unanswered in the already published work. For example, how can one take into account constraints like defective regions, or can one model general shape objects to be cut. Is there inherent flexibility of the existing algorithms to be generalised into higher dimensions? One goal of this thesis is to provide answers to these questions. Novel algorithms which can take into account several constraints (like defective regions) and model general shape objects easily generalised into higher dimensions are presented and thoroughly tested.

Secondly, emphasis is given on the determination of the shape of the object to be cut by means of four cameras placed at 90 degrees around it. Computer vision techniques are applied here with several research areas being addressed. The already existing and widely used technique of camera calibration [31] for three-dimensional reconstruction was found inadequate due to the large number of reference points needed. Moreover, Projective Geometry based approaches [26] did not seem to be robust enough. We hope the formal sensitivity analysis, presented here, will enlighten the projective reconstruction technique and the guidelines given will be a good scientific tool for its future use. Finding
1.3. OUTLINE OF THE THESIS

corresponding features in two images, the so called correspondence problem, is also addressed. The camera separation of 90 degrees and the non-coplanar features makes it difficult for already existing matching algorithms to work [24]. In spite of this, the correspondence problem is casted into an optimisation framework and robust algorithms to solve it are suggested. A detailed outline of the contents of this thesis is given next.

1.3 Outline of the thesis

We start in chapter 2 by carefully examining various one-dimensional packing problems, like the bin packing [8] and the Knapsack [23] problems. These problems are well known for their complexity in the field of Operations Research, and many approximation algorithms for obtaining sub-optimal solutions are reviewed. Packing in two-dimensions is then examined. First, the floorplanning and placement problems, well known in the design of VLSI chips, are formulated and it is shown how good sub-optimal solutions can be obtained fast by using either deterministic or stochastic algorithms. Approaches which use linear programming or integer programming [30] formulation and eigenvalue-based [14] and graph-theoretic approaches [22, 4, 18] are all part of the deterministic techniques presented and compared. On the other hand simulated annealing [1, 21] and genetic algorithms [17] are the two stochastic approaches investigated for the approximate solution of floorplanning and placement problems [7]. The limitations of these techniques soon become obvious as none of them can take into account fixed rectangles which can, for example, model defects on the surface of the main rectangle to be packed. The final section of chapter 2 is therefore devoted to the formulation of the Constrained Rectangle Packing problem according to which rectangular boxes are packed into a single orthogonal bin so that the uncovered area is minimised [11]. Defective regions can be effectively modelled as fixed boxes. Moreover, rotation of the rectangles is allowed and also the relative position of boxes can be prespecified and no uniform distribution of the sizes of rectangles is assumed. A solution is then presented by using efficient topological sorting algorithms and the stochastic technique of simulated annealing.

All of the two-dimensional packing algorithms presented in chapter 2 could deal with rectangular pieces only. The necessity of designing an algorithm which could handle general shape objects to be packed (or cut) was becoming clear. Chapter 3 concentrates on the Generalised Stock Cutting Problem according to which an object of general shape with defective regions or holes of general shape is cut into rectangular pieces [12]. It
1.3. OUTLINE OF THE THESIS

is shown how morphological operators [15] can be used to obtain the optimal solution of the problem when the cutting pattern is specified. The problem of determining the cutting pattern is then attacked. It is shown that the task of finding the optimal pattern is \( \mathcal{NP} \)-hard as it is equivalent to solving a set-packing problem [3]. It is shown, however, that careful formulation of the problem can lead to a special case of the set-packing problem which can be solved by using the simplex algorithm [9]. However, if constraints on the maximum number of rectangles of each type to be used exist, the problem cannot be optimally solved and therefore the efficient technique of simulated annealing is employed in order to obtain good-quality sub-optimal solutions in reasonable time. Experimental results on real images illustrate the validity and usefulness of the proposed algorithms. It should be noted that the proposed algorithm for solving the Generalised Stock Cutting Problem as well as the algorithm for solving the Constrained Rectangle Packing can be very easily generalised to higher dimensions. Indeed, they have been successfully applied for packing cubes and parallelepipeds in the 3D space.

No matter how powerful packing or stock-cutting algorithms exist, they would be useless for the full automation of industrial environments if a robust and accurate way of determining the shape of the object to be cut was not available. For this purpose computer vision techniques are adopted in this thesis. In chapter 4 a review of the two most widely used approaches for reconstructing the shape of an object is presented. The first technique [31] is based on determining the internal camera geometric and optical characteristics and the position and orientation of the camera, what is known as the camera calibration intrinsic and extrinsic parameters respectively. Relatively accurate reconstruction is shown to be achieved by the aid of many reference points. On the contrary, a technique based on Projective Geometry [25, 26] is shown to require no camera calibration parameters and satisfactory results can be obtained in most cases with as few as 6 reference points. However, the satisfactory performance of the projective-reconstruction algorithm is shown experimentally to suffer from instability and major sensitivity to input noise.

A detailed-sensitivity analysis of the Projective-Geometry based reconstruction technique is therefore undertaken [13] in chapter 5. All intermediate stages of the algorithm are formally analysed and many cases where great sensitivity to noise is expected are discovered. First, in the determination of the so called projective coordinates there are many avoidable configurations of the reference points which can lead to numerical un-
stable results. Moreover, it is shown which regions of the image are likely to be unstable during the determination of the viewing lines and the actual 3D reconstruction of points. Guidelines for the proper positioning of the reference points are also given.

In chapter 6, the well known correspondence problem in stereo vision is dealt with. Although much work has been done in the past on this problem [24, 5, 28, 27, 29, 19, 2, 32], many difficulties arise when the features to be matched are non-coplanar and the two cameras are set at 90 degrees from each other. It is shown how to cast the correspondence problem into an optimisation problem and a cost function for evaluating possible solutions is derived. Two efficient algorithms are then presented. One is based on a branch and bound which can be used due to our monotonically increasing cost function. For increased robustness to noise a Hough-Transform like algorithm is presented next. Both algorithms are shown to obtain satisfactory solutions on both synthetic and real images and the relative merits of each one are highlighted.

Finally, we conclude in chapter 7 where the main contributions of this thesis are clearly outlined and future research work towards many possible directions is suggested.

References


REFERENCES


REFERENCES


Chapter 2

Packing Problems

First, a brief introduction to various one- and two-dimensional packing problems is presented. Floorplanning problems, which are of great importance in the design of VLSI chips are then studied and the techniques of simulated annealing and genetic algorithms are used to obtain good quality solutions. Finally, the Constrained Rectangle Packing problem is formulated and an efficient algorithm for solving it using the technique of simulated annealing is presented. Experimental results are also included.

2.1 Introduction

Packing problems go back to 1932 when Abe [2] made an attempt to find a perfect square, in other words, to cover a square simply and without gaps with a finite number of squares which are all different from each other. Although he did not manage to solve the problem, he gave many examples of rectangles fully covered by different squares. It was not until eight years later that Brooks et al [6] presented the perfect square shown in figure 2.1 and the methodology used, based on the electrical theory of networks, in order to generate it. The problem of finding the perfect square made of the least number of squares was solved by Duijvestijn [12].

In this chapter we mainly concentrate on more general packing problems [16]. In section 2.2, after an introduction to one-dimensional packing problems, some approximation algorithms are examined. The main ideas from the one-dimensional algorithms are then applied into higher dimensions.

In section 2.3 the floorplanning and placement problems, well known in the design of VLSI chips, are formulated and a review of the most commonly used algorithms is presented. The main body of this section is devoted to the most recent techniques of sim-
A formulation of the two-dimensional Bin Packing problem or Rectangle Packing problem is discussed in section 2.4 and a solution to the Constrained Rectangle Packing problem which is of greater interest in practical applications is proposed. A set of search operators is proposed which can be used by the simulated annealing algorithm to obtain good-quality solutions. Experimental results and comparisons between different approaches are also included.

2.2 Bin Packing

This chapter is primarily concerned with the investigation of two-dimensional packing problems. However, a brief description of one-dimensional bin packing problems will be given first. After the formulation of the one-dimensional bin packing problem an attempt is made to familiarise the reader with the jargon used by computer scientists, mathematicians and operation researchers who have extensively studied this problem. Indeed, since it was shown [15] that packing problems in general are \( \mathcal{NP} \)-hard and there is no hope to solve them in polynomial time, a lot of attempts have been made to design and analyse simple and efficient approximation algorithms. We conclude with the introduction of simple algorithms for the two-dimensional bin packing problem.
2.2. BIN PACKING

2.2.1 One-Dimensional Bin Packing

Given a list of \( n \) items of sizes \( s_1, \ldots, s_n \) (with \( 0 < s_i < 1 \)), the one-dimensional bin packing problem is to find the smallest number of bins in which these items can be packed, subject to the constraint that the total size of the items packed in any one bin cannot exceed 1. In other words, given a set \( U = \{ u_1, u_2, \ldots, u_n \} \) of \( n \) items and a size \( s_i \in \mathbb{R}, 0 < s_i < 1 \) for each item \( u_i \in U \), the objective of the one-dimensional bin packing problem is to find a partition of \( U \) into disjoint sets \( U_1, U_2, \ldots, U_k \), where the sum of sizes of the items in each \( U_i \) is no more than 1, such that \( k \) is minimum. Since the problem is known to be \( \mathcal{NP}-\)hard [15], simple approximation algorithms are worth considering.

The First Fit (FF) algorithm suggests placing the items into the unit-capacity bins one at a time, in order of increasing index. Given an infinite sequence of \( B_1, B_2, \ldots \) bins, which are empty at the beginning, the item \( u_i \) is placed into the lowest-indexed bin for which the sum of the sizes of the items already in the bin does not exceed \( 1 - s_i \).

The Next Fit Decreasing (NFD) algorithm is an improved version of the First Fit algorithm. The items on the list are first re-indexed so that \( s_1 \geq \ldots \geq s_n \). Then they are allocated to bins in that order. As in the First Fit algorithm, a new bin is opened whenever there is not enough space for the current item in the most recently opened bin. Worst case analysis of this algorithm can be found in [3] and the average case analysis for uniformly and randomly distributed data in [11] and [30] respectively.

The Best Fit algorithm is similar to the First Fit, except that the current item is placed in the bin into which it will fit with the smallest gap left over. The Worst Fit algorithm places an item in the non-empty bin with the biggest gap, starting a new bin if the biggest gap is not big enough. The Almost Worst Fit algorithm tries the second largest gap first, and then proceeds as does Worst Fit, which surprisingly makes a difference. Many other algorithms for one-dimensional bin packing problems have been studied. For a survey see [7].

It is important to realise, that some of the above algorithms do not require the whole sequence of item sizes to be known before packing can proceed. Indeed, the First Fit algorithm can perform packing as soon as an item is available. These are referred to as “online” bin packing algorithms. On the other hand, the Next Fit Decreasing algorithm requires all sizes to be known, because sorting has to be performed. Algorithms, which belong to this category are known as “off-line” bin packing algorithms. On-line algorithms tend
to have worse performance than off-line algorithms, because the latter can preview and rearrange the items before packing them.

Recently, important generalisations of the bin packing problem have been investigated [13],[14] where the size of bins is allowed to vary. According to this model, which is much more realistic for practical problems, there is a fixed collection of bin sizes and the objective is to minimise the total space of the packing.

In [14], off-line algorithms for variable-sized bin packing were analysed. The most complicated of those, the First Fit Decreasing Least Size (FFDLS), was proved to produce a packing whose total space is asymptotically bounded by $4/3$ times the optimum. A fully polynomial-time off-line approximation scheme has been devised in [29] using a linear programming formulation of the problem.

### 2.2.2 The Knapsack Problem

For the sake of complicity, a variation of the one-dimensional bin packing problem the Knapsack Problem will be described next [28]. The problem scenario is as follows: A person is planning a hike and has decided to carry various items of total weight less than $W$. If $w_i$ is the weight of object $i$ then the weight limitation can be expressed by

$$\sum_{i=1}^{n} w_i s_i \leq W \quad (2.1)$$

where $s_i$ is set to 1 to indicate that item $i$ has been selected, and to 0 otherwise. If $v_i$ is the relative value of item $i$, determined by the hiker in comparison to the values of the other objects, then the problem is to choose the values of $s_i$, so that the total value

$$\sum_{i=1}^{n} v_i s_i \quad (2.2)$$

is maximal.

Many similarities between bin packing problems and this problem are apparent. To begin with, both problems are $\mathcal{NP}$-hard [15]. In bin packing there is an infinite number of bins to be used, whereas just one knapsack is to be filled in the knapsack problem. A perfect packing of the knapsack, in terms of weight, does not necessarily mean an optimal solution to the problem. It is the sum of the relative values of packed items that make this problem difficult to solve.

Linear programming techniques could be employed if the value of $s_i$ could lie anywhere in the interval from 0 to 1. This can be illustrated graphically, as shown in figure 2.2.
2.2. BIN PACKING

If each item is represented in the plane by a point having coordinates \((w_i, v_i)\) an approximate solution can be obtained as follows: Rotate clockwise a ray with the origin as pivot point and with starting position being the \(v_i\) axis. Points which are swept out by the ray correspond to items which are selected. Rotation of the ray stops when upon the selection of an item \(i\) the sum of the weights of selected items exceeds the weight limitation. If \(s_i\) could take all real values between 0 and 1, the optimal solution to this problem could have been obtained by choosing the value \(s_i\) as the fractional part of the weight of item \(j\) which would make the sum exactly \(W\). In practice, since the weight carried does not have to be exactly \(W\), the technique described above is all that is needed.

An extension of the knapsack problem into two-dimensions will be examined in section 2.4. Then the Rectangle Packing Problem will be reformulated as a 2D knapsack problem with prime constraint that the sum of the rectangle areas should not exceed the area of the 2D bin (the rectangle to be covered). The relative value \(v_i\) of each rectangle is simply its area. Therefore, by maximising the sum of the relative values of the selected rectangles it is ensured that maximum area utilisation of the bin is achieved.

2.2.3 Two-Dimensional Bin Packing or Rectangle Packing

Extension of the one-dimensional packing problem into two dimensions results in the two-dimensional Bin Packing problem or simply rectangle packing. In one version of the two-dimensional bin packing problem the items \(s_i\) are rectangles with height \(h_i\) and width \(w_i\). The goal is to pack them in a vertical strip of width \(C\), so as to minimise the total height of the strip needed. The rectangles must be packed orthogonally, that is, all rectangles must have their width parallel to the bottom of the strip.
A number of problems in computer science and operations research can be adequately represented as a packing of rectangles in a strip of a finite fixed width and infinite length. Examples of such problems are: (a) scheduling in a multiprogrammed computer system, where the width is the main storage size and the length is processing duration; (b) stock cutting (sheets of metal, cloth, paper), where the dimensions are those of the original roll.

A number of strip packing algorithms are based on a "bottom up-left justified" (BOTTOM-LEFT for short) packing rule. According to this rule, items are packed in turn, each item being placed as near to the bottom of the strip as it will fit and then as far to the left as it can go at that bottom-most level. Preordering may be performed resulting in four different algorithms, depending on how the rectangles are initially re-indexed. If BL stands for BOTTOM-LEFT then BLIW, BLIH, BLDW and BLDH stand for the algorithm with preordering by increasing width, increasing height, decreasing width and decreasing height respectively.

Another group of packing algorithms, which try to achieve better asymptotic worst case ratios was studied by Tarjan [8]. These algorithms are based on a different type of packing rule, suggested by Golan [18] and called "level algorithms". Most of them try to apply knowledge from one-dimensional bin packing. It is obvious that if all the rectangles had the same height, the two-dimensional problem would reduce to the one-dimensional case with all items being placed in rows or "levels". An example of how level algorithms produce packings as a sequence of levels is shown in figure 2.3 where items are first preordered by decreasing height and then placed in levels from left to right.
2.2.4 Conclusions

Most algorithms analysed in this section are popular mainly because of their mathematical tractability rather than their attractiveness from a practical point of view.

It should be clear by now that two-dimensional packing is much more difficult than packing in one dimension. If not, a trivial example will persuade the reader: Consider the simple 1-D problem: Given 7 items of sizes \(0.1, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\) one can easily decide that the minimum number of unit size bins to be used is 4. One possible answer would be to optimally pack \(0.3, 0.7\) in the first bin, and \(0.4, 0.6\) in the second one. The third and fourth bins could be left partially packed \(0.1, 0.8\) and \(0.5\) but it would make no difference to the overall number of bins used.

In the 2D problem, assume that the same numbers represent the length of the sides of the squares and that we have at our disposal an infinite number of unit area squares (2D bins) which we may cover. In order to find the minimum number of bins required, one might propose the following solution as optimal: "Two bins are required since \(0.1^2 + 0.3^2 + 0.4^2 + 0.5^2 + 0.7^2 = 1.0\) and \(0.6^2 + 0.8^2 = 1.0\)." It does not take a long time to realise that the proposed solution is not valid since there is no way to fit these squares legally. Indeed, even though the areas add up to the bin area, this is not sufficient to ensure that legal packing can be achieved. It is the additional constraints, like conditions for non-overlapping that make the two-dimensional problem difficult. Hence, a mathematical model is required to represent and evaluate the cost of proposed solutions. Some more complex formulations will be examined in the next two chapters. It should be noted, however, that complex algorithms are not tractable mathematically and most of them are studied from an experimental point of view rather than mathematical.

2.3 Floorplanning

Floorplan design is the first stage of VLSI circuit layout. The objective is to allocate spaces to a given set of circuit modules in the plane so that the weighted sum of the area of the bounding rectangle containing the modules and the total wiring required to interconnect the modules are minimised.
2.3.1 Introduction

Circuit modules are usually rectangular and can either have fixed height and width or variable. In the first case modules are referred to as rigid and in the latter, when dimensions can change within certain limits, the modules can be classified as flexible.

Such flexibility in VLSI design represents the designer’s ability to manipulate the module’s internal structure at the early stages of design. For example, if a module consists of a set of 12 flip-flops, these may be arranged in a single row 12 by 1, or two rows in 6 by 2 arrangement, or in 3 by 4, depending on which configuration gives the best silicon utilisation and minimum wiring length. It is important to minimise the wiring since it represents delays. Moreover, minimising the wiring means less silicon area is required for routing. Floorplan design is a generalisation of the classical cell placement problem in which no modules are flexible.

In section 2.3.2 a survey of the most widely used deterministic algorithms will be presented. In section 2.3.3 we present a formulation of the floorplanning problem, based on a representation where Polish expressions are used to represent slicing structures. The floorplan optimisation using simulated annealing and genetic algorithms will be examined in sections 2.3.3.1 and 2.3.3.2 respectively.

2.3.2 Literature survey

There are analytical approaches as well as graph-theoretical approaches to the floorplanning problem. Both of them will be briefly discussed in this section. The stochastic approaches will be discussed in section 2.3.3.

2.3.2.1 Analytical algorithms

Recently, an analytical approach to floorplanning was proposed in [31]. Like other methods this algorithm provides a suboptimal solution. However, this suboptimal solution is obtained by the consecutive solution of subproblems of smaller size, for which the optimal solution is found. Each subproblem is solved as a 0-1 mixed integer programming problem as follows: There are \( N \) rectangular modules of width \( w_i \) and height \( h_i \). To prevent overlapping of any pair of modules \( i \) and \( j \) it is required that at least one of the following inequalities holds:

\[
x_i + w_i \leq x_j, \quad i \text{ is to the left of } j
\]
2.3. FLOORPLANNING

\[
x_i - w_i \geq x_j, \quad i \text{ is to the right of } j
\]
\[
y_i + h_i \leq y_j, \quad i \text{ is below } j
\]
\[
y_i - h_i \geq y_j, \quad i \text{ is above } j
\]

(2.3)

where \((x_i, y_i)\) and \((x_j, y_j)\) denote the position of the lower left corners of the modules \(i\) and \(j\) with respect to the centre of coordinates.

In order to ensure that at least one of the above inequalities always holds two additional 0-1 integer variables \(x_{ij}\) and \(y_{ij}\) are introduced. Additionally, \(W\) and \(H\) are defined such that we always have \(|x_i - x_j| \leq W\) and \(|y_i - y_j| \leq H\). \(W\) and \(H\) can either be the maximal width \(W_{\text{max}}\) and height \(H_{\text{max}}\) of the chip allowed or if \(W_{\text{max}}\) and \(H_{\text{max}}\) are not known then \(W = \sum_{i=1}^{N} w_i\) and \(H = \sum_{i=1}^{N} h_i\) can be used.

The following system of linear inequalities for any pair of modules \(i\) and \(j\) can be constructed:

\[
x_i + w_i \leq x_j + W(x_{ij} + y_{ij})
\]
\[
x_i - w_i \geq x_j - W(1 - x_{ij} + y_{ij})
\]
\[
y_i + h_i \leq y_j + H(1 + x_{ij} - y_{ij})
\]
\[
y_i - h_i \geq y_j - H(2 - x_{ij} - y_{ij})
\]

(2.4)

It is not hard to see that for each of the four possible choices for \((x_{ij}, y_{ij})\), which are \((0,0), (0,1), (1,0)\) and \((1,1)\), only one of the above inequalities is active since the others hold for any permitted values of \((x_i, y_i)\) and \((x_j, y_j)\). The integer programming model can be completed by imposing a few additional constraints. For example, if the width \(W\) of the chip is known and a minimum height of \(h\) floorplan is required then the problem can be formulated as: \textit{minimise } \(h\) \textit{subject to the constraints} (2.4) and (2.5).

\[
x_i \geq 0, \ y_i \geq 0, \quad \forall i \in N
\]
\[
x_i + w_i \leq W, \quad \forall i \in N
\]
\[
h \geq y_i + h_i, \quad \forall i \in N
\]

(2.5)

The solution for the mixed integer linear programming model can be obtained by standard mathematical programming software. However, since the solution time grows exponentially with the number of integer variables needed, it is impractical to solve
problems of big size. A way to overcome this limitation was proposed in [31]. First, as few as ten modules are selected and a solution is obtained. It can be shown [31] that the partial floorplan can be modelled with fewer rectangles, thus reducing the number of variables required in the integer programming problem. A new group of modules is then added to the partial floorplan and the optimal solution is again obtained. This technique is known as successive augmentation and results in linear execution times, since at each step the partial solution is reformulated, thus reducing the number of variables and constraints in the mixed integer programming problem.

Other deterministic algorithms have been investigated. However, most of them tend to minimise only the wiring needed to interconnect modules. A quadratic objective function is normally used [5], which represents the sum of the squares of the distances between connected devices:

\[ \Gamma = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \]  

(2.6)

where \( c_{ij} \) is a symmetric matrix representing the connectivity between devices \( i \) and \( j \), and \( x_i, y_i, x_j, y_j \) are the coordinates of some reference point on devices \( i \) and \( j \) respectively, and \( N \) the total number of devices. Hall [20] has devised a novel method to minimise the above function by using eigenvalues.

However, algorithms of this kind do not take into account module size, shape and routing channel width. Since they assume zero-area point modules the solutions do not correspond to the placement problem. Mapping the modules to exact positions requires a lot of post processing and many arbitrary decisions to be made. Due to these problems, sometimes it is very difficult to obtain good solutions for large circuits.

2.3.2.2 Graph-theoretic approaches

Most graph theoretic algorithms for solving the floorplanning problem are based on a technique known as rectangular dualisation. A planar graph is normally used to represent the layout. Dualisation refers to the process of finding the dual of a graph, that can be drawn in the form of a rectangular dissection [27].

Bhasker [4] has published a linear algorithm for obtaining a rectangular dual, but it can only process planar triangulated graphs. Jabri [22] has developed an efficient algorithm that transforms an arbitrary graph representing a placement of rectangular blocks, into

\[ \text{selection is normally based on connectivity information} \]
one suitable for rectangular dualisation. This algorithm makes use of efficient techniques in graph processing such as planar embedding.2

2.3.3 Polish expressions

In this section a formulation of the floorplanning problem using Polish expressions will be examined.

Given \( n \) modules there is a triplet of numbers \((A_i, r_i, s_i)\), where \( A_i \) represents the area of the module \( m_i \) and \( r_i, s_i \), with \( r_i \leq s_i \), specify the limits of its aspect ratio. It follows that a module \( m_i \) is a rigid module iff \( r_i = s_i \). Modules can have either fixed or free orientation, in which case rotation of the module by 90 degrees, which is equivalent to interchanging its dimensions, is allowed. If \( w_i \) and \( h_i \) are the width and height of module \( m_i \), and \( O_1 \) and \( O_2 \) the set of modules with fixed and free orientation respectively, we must have:

\[
\begin{align*}
  w_i h_i &= A_i \quad (2.7) \\
  r_i &\leq h_i/w_i \leq s_i \text{ if } m_i \in O_1 \quad (2.8) \\
  r_i &\leq h_i/w_i \leq s_i \text{ or } 1/s_i \leq h_i/w_i \leq 1/r_i \text{ if } m_i \in O_2 \quad (2.9)
\end{align*}
\]

A floorplan for the given \( n \) modules consists of the minimum bounding rectangle, \( R \), which must have an aspect ratio between two given numbers \( p \) and \( q \), with \( p < q \). Partitioning of \( R \) by \( n - 1 \) horizontal and vertical line segments results in the creation of \( n \) non-overlapping rectangular regions \( r_i \). For a realisable floorplan, each region \( r_i \), of width \( x_i \) and height \( y_i \) should be large enough to accommodate its corresponding module \( m_i \), after rotating if \( m_i \in O_2 \) and reshaping if module \( m_i \) is flexible. The first term of the objective function is the area of \( R \) which can be expressed in terms of the sum of the areas of the rectangular regions, i.e.,

\[
A = \sum_{i=1}^{n} x_i y_i \quad (2.10)
\]

It is reminded that one of the floorplan design goals is to minimise the wire length in addition to minimising the area of the bounding rectangle. This demand is incorporated in the objective function as follows: Let \( d_{ij} \) represent the Manhattan distance between the centres of the corresponding regions \( r_i \) and \( r_j \), and \( c_{ij} > 0 \) a cost associated with each connection between modules \( m_i \) and \( m_j \). The cost \( c_{ij} \) is set to 0 if no connection exists between the corresponding modules, otherwise to a value which provides information

---

2Planar embedding refers to the process of deleting a minimal set of edges from a given graph in order to planarise it (i.e. to make it possible to be drawn on a plane without any intersection of its edges).
about the wiring density between the pair of modules. The addition of the weighted wirelength component results in the following objective function:

$$\psi = A + \lambda W = \sum_{i=1}^{n} x_i y_i + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} d_{ij}$$  \hfill (2.11)

The constant $\lambda$ is normally specified by the user so that the relative importance of total area and weighted wirelength can be controlled appropriately.

Figure 2.4: (a) A slicing structure, (b) A non-slicing structure, (c) The slicing tree of the slicing structure, (d) Illustration of the two operators: + and *.

Before we explain how this function can be minimised, it is essential to give some insight into the representation used to describe the cutting or slicing of the bounding rectangle. A slicing structure (figure 2.4a) is a rectangle dissection that can be obtained by recursively cutting rectangles into smaller rectangles by vertical or horizontal lines. By contrast, figure 2.4b shows a non-slicing structure. A slicing structure can be described by a binary tree, called a slicing tree (see figure 2.4). Each leaf corresponds to a rectangular region produced by the slicing process and each node of the tree is labeled either "*" or "+", corresponding to either a vertical or a horizontal cut, respectively. By postorder traversing\(^3\) the slicing tree the expression $a_1 a_2 \ldots a_{2n-1}$ of length $2n - 1$ and elements from $\{1, 2, \ldots, n, *, +\}$ is obtained. Elements that belong in $\{*, +\}$ are the operators and in $\{1, 2, \ldots, n\}$ are the operands. This expression, which is known as Polish expression.

---

\(^3\)In postorder traversal of a tree the left subtree is visited first, then the right subtree and then the root.
can be normalised so that every module appears exactly once, no two consecutive "*" or two consecutive "+" are in the sequence and the number of operands in any position in the expression is greater than the number of operators. For example, the expression "34 + 21 * 5 + +" is a normalised Polish expression which represents a realisable configuration of 5 modules.

2.3.3.1 Optimisation using Simulated Annealing

The physical process of annealing involves first increasing the temperature of a solid to a value at which the solid melts and the careful decreasing of the temperature until the particles arrange themselves in the ground state of the solid where the energy of the system is minimal [1]. “Careful decreasing” in the above definition means spending enough time at each temperature enabling the system to achieve equilibrium so that the resulting configuration has not many defects in the form of high-energy, meta-stable, locally optimal structures. Instantaneous decrease of the temperature is known as the process of quenching which is the converse of annealing.

Kirkpatrick et al [25] suggested that good results to combinatorial optimisation problems can be obtained by simulating the process of annealing. In their analogy between a many-particle physical system and a combinatorial optimisation problem, a solution corresponds to a state of a physical system and its cost is the energy at this state. A control parameter is then introduced which plays the role of temperature. The optimal solution of the problem is equivalent to reaching the ground state of the physical system.

Three types of moves are defined in order to manipulate a given normalised Polish expression:

M1: Swap two adjacent operands.

M2: Complement a chain of operators.

M3: Swap two adjacent operand and operator.

A pictorial illustration can be seen in figure 2.5. In the first example M1 operates upon the "14 * 235 + *+". It chooses at random two adjacent operands (2 and 4) and swaps them. The resulting Polish expression, which is "12 * 435 + *+", is then manipulated by the M2 operator. The "+ *+" is the randomly chosen chain of operators which is complemented, producing "12 * 435 * +*". Finally, M3 is applied and chooses to swap operand 5 and its adjacent operator "+".
It is clear that M1 and M2 always produce a normalised polish expression. However, extra care should be taken when applying M3 since it may produce a Polish expression which is not normalised, however this can be tested efficiently every time M3 is applied.

2.3.3.2 Genetic Algorithms

Genetic algorithms are effective search algorithms based on ideas from natural selection and genetics developed by John Holland [21].

To apply genetic algorithms to an optimisation problem, a finite length string, the chromosome, is constructed over some finite alphabet in order to represent all possible valid states in the configuration space of the problem. A population of these strings is maintained at any one time. Genetic algorithms operate on populations of strings enforcing a Darwinian survival of the fittest amongst them. In every generation, a new set of strings is created using parts of the fittest strings of the previous generation. It has been proved theoretically and empirically that these algorithms provide robust search in complex spaces [21],[19].

Genetic algorithms differ from traditional search techniques in three ways:

- they work with a coding of the problem parameter set, rather than with the actual parameters themselves.

- they search from a population of possible solutions.

- they use probabilistic transition rules, rather than deterministic.
Normally, \( n \) binary coded strings of length \( s \) are randomly generated at the beginning of the search. Each string, which represents a set of physical parameters for the system, has a fitness value associated with it. Fitness in genetic algorithms corresponds to the objective function in classical optimisation theory.

Three operations are then used to modify existing populations.

Reproduction: It is the process by which individual strings are copied to the following generation according to their fitness. Highly fit strings have greater chance to survive and to pass onto the next generation.

Crossover: Two randomly selected strings are mated and produce new solutions which will create the new generation.

Mutation: It is a random alteration of the value of a string position. This operation, when used sparingly, introduces diversity to the search and ensures that potentially useful information in a string position is not lost forever.

Following the same formulation of the floorplanning problem using Polish expressions, we will describe a different approach to solving it using genetic algorithms with punctuated equilibria [10]. The overall structure of the algorithm is as follows: There are \( N \) processors connected by an interconnection network with sufficient connectivity. A set of \( n \) possible solutions is assigned to each of \( N \) processors. The set assigned to each processor is its subpopulation. There are \( E \) major iterations called epochs. During an epoch each processor, disjointly and in parallel, executes a Genetic Algorithm in its subpopulation. After each processor has completed \( G \) generations there is a phase during which each processor copies randomly selected subsets of its population to neighbouring processors.

Three types of crossover operators, CO1, CO2, CO3 are used in order to produce an offspring from two Polish expressions (fig. 2.6). Whenever a crossover operation is to be performed one of the three crossover operators is selected randomly. The first operator, CO1, first copies the operands from one parent into the corresponding positions of the offspring. It then copies the operators from the second parent, by making a left-to-right scan, to complete the offspring. CO2 starts out by copying the operators from one parent and then completes the construction of the offspring by copying the operands from the second parent. Finally, CO3, first copies the operators from a parent to the corresponding positions in the offspring (as in CO2). An operator is then randomly selected from the first
parent and the operands associated with it are copied unchanged into the corresponding positions in the offspring. The remaining operands required to complete the offspring are brought in from the second parent by making a left-to-right scan. For example, if the Polish expression for the first parent is "1456 + + 87 * 32 * + + +" and for the second parent is "268 * + 5 + 4 + 13 + + +" then CO3 first copies the operands "* + + * + + *" from the first parent as shown in figure 2.6). Secondly, it randomly chooses the last "+" operator from parent 1 and finds all the operands associated with it (8, 7, 3, 2 the encircled subtree of the first parent slicing tree). It then copies them into the corresponding positions of the offspring (dashed arrows in figure 2.6). Finally, the remaining operands (6, 5, 4, 1) are copied from the second parent.

Figure 2.6: The three crossover operators (CO1, CO2 and CO3) and the slicing structure for parent 1.

Three types of mutation operators are used. These are the M1, M2, M3 operators which were described earlier in this chapter and used in the simulated annealing algorithm.

A score of every string is simply the value of the cost function. Then the fitness can be calculated by [10]:

\[
\text{fitness}(x) = \frac{(\mu_s - \text{score}(x)) + \alpha \sigma_s}{2\alpha \sigma_s} \tag{2.12}
\]

where \(\mu_s\) is the mean of the scores, \(\sigma_s\) is the standard deviation and \(\alpha\) a parameter.
2.3. FLOORPLANNING

2.3.4 Experimental results and Comparison

We have tried both simulated annealing and genetic algorithms using Polish expressions. Several experiments have shown that genetic algorithms perform consistently better than the simulated annealing approach [10]. Given approximately the same number of strings to be examined the results using genetic algorithms were significantly better both in terms of average cost of the solutions found and the best-found solution.

The slicing structure is restrictive in area utilisation. When there are constraints on the shape of building blocks (i.e. some blocks may be fixed), the wasted space may turn out to be very large in some instances.

For example the perfect square shown in figure 2.1 could never be obtained using the Polish expression representation. This is because all squares are of different size and 100% area utilisation must be achieved, which is the optimal solution. The representation using Polish expressions and slicing structures could not handle that since at least two squares should be modelled as a rectangle without any wastage. However, this is impossible since all squares are of different size.

2.3.5 Conclusions

In this section several types of floorplanning/placement algorithms were discussed. Simulated annealing is currently the most popular amongst researchers and the best amongst the algorithms available in terms of the placement quality. However, it takes an excessive amount of computational time. Improvements have been made concerning mainly the cooling schedule resulting in faster execution speeds. Parallel implementations of the simulating annealing algorithm seem to be very efficient.

The other class of algorithms discussed here, which seem to compete with the simulated annealing algorithm in the quality of the produced solutions and execution time, is the genetic algorithms. It is an extremely efficient search and optimisation technique for problems with a large configuration space. The search through the solution space is inherently parallel, which is a major advantage of the genetic algorithms. However, this parallelism can be a potential problem, and unless a clever representation scheme is devised to represent the physical placement as a genetic chromosome, the algorithm may prove ineffective.
2.4 Rectangle Packing

The two-dimensional bin-packing problem or rectangle packing (RP) problem can be viewed as a generalisation of the classical \( \mathcal{NP} \)-hard [15] bin packing problem which has been extensively studied in one dimension.

2.4.1 Introduction

The objective here is to pack arbitrarily dimensioned rectangular boxes into a single orthogonal bin without overlapping, so that the uncovered area is minimised. The boxes can be either of fixed or free orientation and must have their edges parallel to the bin edges. More formally:

Given a finite set \( B = \{ b_1, \ldots, b_n \} \) of \( n \) rectangular objects and a width \( w_i \in \mathbb{R} \) and height \( h_i \in \mathbb{R} \) for each \( b_i \in B \), our task is to determine the Euclidean coordinates of the lower left-hand-side corner \( x_i, y_i \) for each \( b_i \) within a rectangular bin of width \( W \) and height \( H \), \( 0 < w_i \leq W \), \( 0 < h_i \leq H \) such that the uncovered area

\[
A = W \times H - \sum_{i=1}^{n} m_i \cdot w_i \cdot h_i
\]  

(2.13)

is minimised, where \( m_i \) is a binary variable introduced to indicate mapping information. It is set to one to indicate that an object is legally placed inside the bin, and to zero otherwise. Various constraints may be imposed. Some of the boxes may be fixed, thus they have a predetermined position within the bin and cannot be moved. Other boxes may have a fixed relative position with respect to others or their orientation may be fixed. The incorporation of these constraints leads to the Constrained Rectangle Packing (CRP) problem [17].

A few approximation algorithms for solving the bin-packing problem in two dimensions have been presented, however none of them allows for constraints to be imposed. In [9] an algorithm for packing squares is presented, and the assumption that the sizes of the squares are uniformly distributed in \([0,1]\) is made, which is not always valid. The same assumption is also made in [24], however this algorithm can pack rectangles as well as squares. In [23] an attempt is made to reshape rectangles before packing so that maximum area utilisation can be achieved. Although manipulation of the rectangle dimensions is acceptable in task scheduling, it is not allowed in most other applications. In this section we present a new efficient representation, which can easily handle the various
2.4. RECTANGLE PACKING

Figure 2.7: A simple example illustrates the formation of the "LABR" string.

constraints and a solution using the stochastic technique of simulated annealing [25]. An explanation of this technique was given in section 2.3.3.1.

The necessary ingredients of a simulated annealing algorithm [1] are: (a) an appropriate problem representation, (b) a transition mechanism which allows the generation of a new possible solution from a previous one, and (c) a cooling schedule. All these aspects will be discussed next.

2.4.2 Representation

The problem representation consists of a representation of the solution space and an expression for evaluating the cost function at each configuration. All the expressions used to represent the problem and to evaluate the cost function should be easy to manipulate so that the overall efficiency of the algorithm is high.

2.4.2.1 Solution representation

In order to represent a current configuration a string $S = s_1s_2 \ldots s_l$, constructed over the quaternary alphabet $V = \{L, A, B, R\}$, of length $l = n(n - 1)/2$ is used. Each character $s_i$ in the string $S$ represents the relative position of $b_j$ with respect to $b_k$, $P(b_j, b_k)$, defined as follows:

$$s_i = P(b_j, b_k) \in \{L, A, B, R\} : s_i = \begin{cases} L & \text{if } x_f + w_f \leq x_s, b_f \text{ is to the Left of } b_s \\ A & \text{if } y_s + h_s \leq y_f, b_f \text{ is Above } b_s \\ B & \text{if } y_f + h_j \leq y_s, b_f \text{ is Below } b_s \\ R & \text{if } x_s + w_s \leq x_f, b_f \text{ is to the Right of } b_s \end{cases}$$

where $f = \min(j, k)$, $s = \max(j, k)$, and $i = n(f - 1) - \frac{f(f+1)}{2} + s$, with $j, k, n, f, s \in I$.

From the above definition it follows that $1 \leq f < s \leq n$ and $P(b_k, b_j) = P(b_j, b_k)$.

Let us illustrate how the string $S$ can be constructed for the configuration shown in figure 2.7. Since there are 5 rectangles, the length of the string is $l = 5 \cdot 4/2 = 10$ characters.
Every character represents the relative position between any two rectangles. Therefore, the first character, $s_1$, is set to $B$ since $b_1$ is below $b_2$. The second character, $s_2$, describes the relative position of $b_1$ with respect to $b_2$ and is set to $L$, and all remaining characters are set in a similar way.

For a given configuration there may be more than one string representation (for example a certain rectangle $b_j$ may be both below & and to the right of $b_k$ causing $\mathcal{P}(b_j, b_k)$ to be either $B$ or $R$). This ambiguity is not important, as we may use any of these strings and still reproduce the same configuration.

It should be noted that since the relative placement $\mathcal{P}(b_k, b_j)$ can take any value for any $(b_j, b_k)$ pair, there will be many invalid configurations among the $4^{n(n-1)/2}$ that arise. For example if $\mathcal{P}(b_k, b_j) = L$ and $\mathcal{P}(b_j, b_m) = L$ it follows that $\mathcal{P}(b_k, b_m) \neq R$ (see figure 2.8). According to experimental results, it seems that the number of invalid solutions exceeds the number of the valid ones when $n$ is large. This is illustrated in table 2.1, where $\text{Ratio}$ is defined as the number of valid solutions divided by the number of invalid. Therefore, it is important to restrict the search inside the valid configuration space during the optimisation process. This will be further discussed in section 2.4.3.

### 2.4.2.2 Calculation of the cost function

Before the cost function can be calculated it is necessary to convert the relative placement of objects, as described by the string $S$, into an absolute placement. This means that the
2.4. RECTANGLE PACKING _________________________________________________

$\bar{x}_i$, $\bar{y}_i$ coordinates of every object $b_i$ have to be determined. This can be done very easily, by employing a compaction algorithm.

A compaction algorithm can be thought of, as a way of bringing all objects as close as possible, by minimising their minimum enclosing rectangle. Their edges must be parallel to the bin edges, and there should be no overlapping between them. Finally, all the adjacencies specified by the relative placement description string $S$ should remain exactly the same. The compaction problem can be formulated as a linear programming problem and solved using the simplex method [31]. However, substantial storage is required in order to build the linear programming tableau. Furthermore, a long time is required for problems of big size. For this reason, we have developed a more efficient compaction algorithm which falls in the general category of Topological Sorting Algorithms (TSA) [26].

A two-dimensional compaction process moves the objects in both the vertical and horizontal directions of the bin area simultaneously. Since all objects are rectangular, a true “two-dimensional” algorithm is not needed. The compaction problem can be broken down into two “one-dimensional” problems, one in the horizontal and one in the vertical direction. Without loss of generality the compaction algorithm just for the horizontal direction is described next. First, every rectangle is checked to determine whether it has a neighbouring rectangle to the left or not. If it does not, then it is pushed onto the stack and is classified as a leftmost object. Next, the $\bar{x}_i$, $\bar{y}_i$ coordinates of all objects must be calculated by taking into account the relative left-to-right placement and the widths or heights in the case of rotated objects. Additionally, since some of the objects are fixed, their $\bar{x}_i$ coordinate must not change. An algorithm to satisfy the above requirements is proposed in figure 2.9.

Given the $\bar{x}_i, \bar{y}_i, w_i, h_i$ for all n modules, it is easy to determine the value of the binary variable $m_i$,

$$m_i \in \{0, 1\} : m_i = \begin{cases} 0 & \text{if } \bar{x}_i + w_i > w_b \\ 0 & \text{if } \bar{y}_i + h_i > h_b \\ 1 & \text{otherwise} \end{cases} \quad (2.14)$$

and then to calculate the value of the cost function according to equation 2.13.
2.4.3 Optimisation

To restrict the search to the valid subspace of the problem’s state space an efficient search algorithm has been developed. This employs three operators which generate a new valid configuration from a previous one, the Move, Exchange and Rotate operators. They operate directly upon the string $S$ and the object dimensions $w_i, h_i$, thus enabling the algorithm to reach any point in the valid configuration space. In order to generate a new solution from the current one, one out of the three operators is chosen at random. From the above operators only the Move is the one which may introduce invalid configurations. However, we define it in such a way that it preserves the validity of the configuration.
2.4. RECTANGLE PACKING

BEGIN (Move)
Choose \( b_i, b_j \) at random, \( i \neq j \)
\[ \mathcal{P}(b_i, b_j) = \text{random}(L, A, B, R) \]
FOR every \( k \in \{1, \ldots, n\} \)
   IF \( (i > k \text{ AND } j > k) \)
   OR \((i < k \text{ AND } j < k)\)
\[ \mathcal{P}(b_k, b_i) = \mathcal{P}(b_k, b_j) \]
ELSE
\[ \mathcal{P}(b_k, b_i) = \text{Not}(\mathcal{P}(b_k, b_j)) \]
END IF
END FOR
END

Figure 2.10: The Move operator.

Before we explain how the three operators work, let us define two more operators, the Not:
\[
\text{Not}(\mathcal{P}(b_i, b_j)) = \begin{cases} 
L & \text{if } \mathcal{P}(b_i, b_j) = R \\
A & \text{if } \mathcal{P}(b_i, b_j) = B \\
B & \text{if } \mathcal{P}(b_i, b_j) = A \\
R & \text{if } \mathcal{P}(b_i, b_j) = L 
\end{cases} 
\]
and Swap:
\[
\text{Swap}(\mathcal{P}(b_i, b_j), \mathcal{P}(b_k, b_m)) \Rightarrow t = \mathcal{P}(b_i, b_j); \mathcal{P}(b_i, b_j) = \mathcal{P}(b_k, b_m); \mathcal{P}(b_k, b_m) = t 
\]

The Rotate operator simply interchanges the width \( w_i \) and height \( h_i \) of a randomly selected object \( b_i \). The Move operator chooses two objects \( b_i, b_j \) at random, where \( i \neq j \) and moves randomly \( b_i \) to one of the four sides of \( b_j \). Then \( \mathcal{P}(b_i, b_j) \) is assigned its value drawn from \( \{L, A, B, R\} \), and \( b_j \) inherits all the topological attributes of object \( b_i \) as shown in figure 2.10. The Exchange operator provides a convenient way of exchanging the positions of two objects (figure 2.11).

With the representation and the robust search operators introduced, the technique of simulated annealing can be employed in order to minimise \( A \). Initially we start from a random valid configuration and apply randomly one of the three operators to generate a new configuration. The proposed change is accepted if it reduces \( A \). However, if \( A \) increases it may still be accepted with a probability of \( e^{-\Delta A/T} \), where \( \Delta A \) is the cost increase, and \( T \) is the control parameter. The value of \( T \) is initially set to a large
BEGIN (Exchange)
Choose $b_i, b_j$ at random, $i \neq j$
Not$(P(b_i, b_j))$
FOR every $k \in \{1, \ldots, n\}$
  IF $(i > k \text{ AND } j > k)$
    OR $(i < k \text{ AND } j < k)$
    Swap$(P(b_k, b_i), P(b_k, b_j))$
  ELSE
    Not$(P(b_k, b_i))$
    Not$(P(b_k, b_j))$
    Swap$(P(b_k, b_i), P(b_k, b_j))$
END_IF
END_FOR
END

Figure 2.11: The Exchange operator.

(a) Sliced Rectangle. (b) Final Solution.

Figure 2.12: Packing of 18 blocks (two of them fixed) obtained after 50 iterations.

enough value, to ensure that all proposed configurations are equally acceptable. It is then
decreased according to: $T_{new} = 0.9 \times T_{old}$. This cooling schedule has been determined
experimentally according to [1].

2.4.4 Experimental results

We have implemented our algorithm in C and run it on a SPARC2 workstation. First
the linear programming and the topological sorting algorithm (TSA) were compared.
Both algorithms obtained the expected solution of the problem. The substantial speed
improvement of the topological ordering algorithm over the simplex method is obvious
in Table 2.2. In order to test our algorithm, we constructed an artificial problem whose optimal solution is known. The rectangle shown in Figure 2.12a was sliced into 20 smaller rectangles. The dimensions of these rectangles were fed into our algorithm. The solution obtained after 50 iterations (in less than one minute on a SPARC2 workstation) for the constrained problem (Figure 2.12b), with objects $b_8$ and $b_{15}$ fixed, achieves 92% area utilisation. For the unconstrained case utilisations up to 95% were obtained. The performance of the simulated annealing algorithm can be seen in Figure 2.14 and a pictorial illustration can be found in Figure 2.15. Since we are not aware of any experimental results or benchmark data for the two-dimensional Constrained Rectangle Packing problem no comparisons with other algorithms can be made.

Finally, a few implementation details should be observed. To achieve maximum efficiency, the over-constrained string representation can be relaxed. This means that object positions which can be inferred from the relative position to other objects, do not have to be explicitly specified. This can be done by the introduction of a fifth element in our alphabet $V$. Thus, $\mathcal{P}(b_i, b_j) = X$ means that the relative position of block $b_i$ with respect to $b_j$ can be inferred from other information given concerning the configuration represented by the string. This relaxed representation in conjunction with use of linked lists, can improve the execution speed of the algorithm.
2.4. RECTANGLE PACKING

Figure 2.13: Percentage of invalid configurations.

Figure 2.14: Performance of Simulated Annealing.
2.4.5 Conclusions

We have presented an algorithm for solving the two-dimensional Constrained Rectangle Packing problem. Experimental results indicate that by the use of the fixed-size string representation and careful stochastic search of the configuration space, sub-optimal solutions can be obtained in relatively short time even for problems of big size. Moreover,
additional constraints can be applied very easily, taking advantage of the inherent flexibility of our modeling. A generalisation of our algorithm in $d$-dimensions is trivial. A three-dimensional version has been implemented and packed successfully cubes and parallelepipeds. Before we close this section, let us highlight once again the major advantage of the algorithm. Indeed, the efficient search in the valid configuration subspace makes it much faster than other algorithms. Although, it is computationally infeasible to determine the actual percentage of the invalid configuration space for large numbers of $N$, it is possible to estimate it using random sampling. It is then clear that any algorithm that does not limit the search into the valid subspace would be wasting computational power, proposing and evaluating mostly invalid configurations. Almost 99.9% of the configuration space are invalid configurations, as it can be seen in figure 2.13.

2.5 Conclusions

In this chapter we have examined various packing problems. Although the one-dimensional packing problems were investigated first, their extension into higher dimensions was the major topic of this work. Perhaps the main conclusion to be drawn, is that all the problems which were investigated, belong to a class of problems which are known to be $NP$-hard. This means that one cannot find a polynomial-time algorithm to solve them, no matter how hard one tries.

The major direction of the research in this area is towards finding efficient approximation algorithms which can give good solutions. Simple approximation algorithms which are mathematically tractable are mainly studied by computer scientists and mathematicians. However, these algorithms tend to be of no practical value, since their simplicity results in bad quality results. On the other hand, engineers and operations researchers tend to develop somewhat more complicated algorithms whose performance can only be analysed experimentally.

We have used the stochastic techniques of simulated annealing and genetic algorithms which have been reported to achieve good results in some combinatorial optimisation problems. Indeed, the simulated annealing performed very well in most problems tried, but its major disadvantage is the long computational time required to obtain a good quality solution. Genetic algorithms, tend to perform slightly better than simulated annealing, provided a good representation scheme is used.
References


REFERENCES


REFERENCES


Chapter 3

Generalised Stock-Cutting Problem

In this chapter we formulate the two-stage stock-cutting problem according to which a set of rectangular pieces of prespecified dimensions are to be cut from a general shape object with general shape holes or defective regions. We show how mathematical morphological operators can be used in order to determine the optimal shifting for a given cutting pattern. It is then proved that the problem of obtaining the optimal cutting pattern is NP-hard and a solution to the unconstrained problem using mathematical programming is proposed. However, for the general problem good sub-optimal solutions can be obtained using the technique of simulated annealing. Experimental results are also included.

Introduction

In the two-dimensional stock cutting problem, a single rectangular piece, the stock sheet, is to be cut into a finite number of rectangular pieces of given size and value. The objective is to maximise the sum of the values of the pieces cut in the final product. If the value of each piece is a function of its area, then the maximisation problem defined above can be thought of as a minimisation of the un-utilised area.

Linear programming techniques have been used by Gilmore and Gomory [7], but the number of variables and constraints required make it impossible to solve big problems without the introduction of heuristics. Cristophides and Whitlock [3], presented an efficient algorithm for solving a special case of the constrained cutting problem. In their formulation all cuts go from one edge of the rectangle to another, what is known as a "guillotine" type of cut. A tree search strategy with a dynamic programing procedure and a transportation routine are then used to obtain the optimal pattern of cutting. Other researchers have also considered guillotine cutting problems. Wang [11] proposed two combinatoric methods that generate constrained cutting patterns by means of horizontal
and vertical building of rectangles. These methods can efficiently solve small problems and provide good approximate solutions for larger problems.

There is at least one attempt to solve the non-guillotine cutting problem by Beasley [2] who formulated the problem as a 0-1 integer programming model and solved it using the branch and bound algorithm. Lagrangian relaxation and sub-gradient optimisation were used in order to obtain tight bounds, thus optimising the performance of the branch and bound algorithm. Good results were reported for moderately sized problems.

In spite of the increasing popularity of the rectangular stock-cutting problem no work has been done so far in the more general case where the shape of the material to be cut is not rectangular. In this chapter we will show how mathematical morphological operators can be used efficiently to solve general-shape two-dimensional stock-cutting problems, where the object to be cut is not constrained to have a regular shape. We shall only deal with the modified two-stage guillotine cutting problem, according to which rectangles must be produced in two stages, first by cutting through horizontally and then vertically, as shown in figure 3.1. Our model allows for the existence of general-shape defects on the object surface. It also takes into account the length and width of cutting, \( l_c \) and \( w_c \) respectively. In most applications \( l_c = w_c \) but for some non-isotropic materials this may not be the case, therefore we will assume that \( l_c \neq w_c \).

### 3.1 Problem Formulation

Let \( S \) be an \( L \times W \) matrix representing the two-dimensional object to be cut:

\[
s_{xy} = \begin{cases} 
1 & \text{if } s_{xy} \text{ represents an object point} \\
0 & \text{if } s_{xy} \text{ represents defective area or background}
\end{cases} \quad (3.1)
\]

where \( x, y \) are integers. Without loss of generality, we can assume that the first and last row and column of \( S \) has at least one non-zero element, i.e.:

\[
\sum_{y=1}^{W} s_{y1} \neq 0, \quad \sum_{y=1}^{W} s_{yL} \neq 0, \quad \sum_{x=1}^{L} s_{1x} \neq 0, \quad \sum_{x=1}^{L} s_{Wx} \neq 0 \quad (3.2)
\]

If this is not the case, we can determine the four extremal points of the region, namely the leftmost, topmost, rightmost and bottommost points, \((x_{\min}, y_{\min})\), \((x_{\max}, y_{\min})\), \((x_{\min}, y_{\max})\), \((x_{\max}, y_{\max})\) respectively

\[
x_{\min} = \min(x \mid (x, y) \in S) \quad y_{\min} = \min(y \mid (x, y) \in S) ,
\]
\[
x_{\max} = \max(x \mid (x, y) \in S) \quad y_{\max} = \max(y \mid (x, y) \in S) . \quad (3.3)
\]
These define what is known as the Minimum Enclosing Rectangle or the Bounding Box of the object. Then we extract the sub-matrix of size \((L = x_{\text{max}} - x_{\text{min}} + 1) \times (W = y_{\text{max}} - y_{\text{min}} + 1)\) enclosed by:

\[
x_{\text{min}} \leq x \leq x_{\text{max}}, \quad y_{\text{min}} \leq y \leq y_{\text{max}}.
\]

Figure 3.1: The 2-stage Stock Cutting Problem. Only the rectangles numbered from 1 to 14 can be utilised.

The 2-stage stock-cutting problem can then be defined as follows: Given a binary matrix \(S\) representing the object to be cut as defined by equation 3.1, and a set \(R\) of \(m\) smaller rectangular pieces of specified same length and different widths, \(R = \{(l, w_1), (l, w_2), \ldots, (l, w_m)\}\), and a maximum number \(b_i\) of pieces of type \(i\) to be cut from \(S\), cut object \(S\) so that the following quantity is maximised:

\[
\tilde{z} = \sum_{i=1}^{m} \xi_i l w_i, \tag{3.4}
\]

where \(\xi_i\) is the number of pieces of type \(i\) cut from \(S\). Clearly, since the length \(l\) is the same for all pieces, it can be omitted from the above expression of \(\tilde{z}\) which can be redefined as:

\[
z = \sum_{i=1}^{m} \xi_i w_i, \tag{3.5}
\]

with \(\xi_i \leq b_i, 1 \leq i \leq m, \xi_i \in \mathbb{Z}^+, i \in \mathbb{Z}^+\), where \(\mathbb{Z}^+\) is the set of positive integers.
3.1. PROBLEM FORMULATION

The object $S$ is defined only for integer values of $x$ and $y$, therefore we will only consider integer values for $l, w_i$. As it can be seen from figure 3.1, there are three parameters to be determined in order to obtain the optimal solution:

- The optimal translation of the cutting scheme (shown by the dark grid in figure 3.1), has to be specified by determining the optimal distances $dx, dy$ from the leftmost and topmost point of the minimum enclosing rectangle.

- Since there are $m$ different types of pieces of identical length, we define the one-dimensional cutting pattern as:

\[
\pi = [\pi(1), \pi(2), \ldots, \pi(k)] \quad \text{where} \quad \pi(j) \in \{1, 2, \ldots, m\}, \quad \forall j \in \{1, 2, \ldots, k\}
\]

\[
\text{and} \quad 1 \leq k \leq \left\lfloor \frac{W}{\min_{1 \leq i \leq m}(w_i) + w_e} \right\rfloor \quad (3.6)
\]

where $\left\lfloor \cdot \right\rfloor$ means integer part of. Clearly, $\pi$ is an ordered $k$-tuplet made up from the various types of rectangles to be cut, each of which may appear in the $k$-tuplet more than once. The cutting pattern $\pi$ has to be determined as part of the optimisation process. There may be as many as $\left\lfloor \frac{W}{\min_{1 \leq i \leq m}(w_i) + w_e} \right\rfloor^m$ possible configurations to be examined and that is what makes the problem hard to solve.

- The rotation parameter, not shown in figure 3.1. It may be necessary to rotate object $S$ by an angle $\theta$ before cutting it in order to obtain optimal results. In what follows, we shall first try to solve the problem for fixed angle $\theta$ and later on we shall use a heuristic to determine the best rotation parameter.

The rest of this chapter is organised as follows: In section 3.2 an efficient way to obtain the optimal $dx, dy$ values by which the cutting structure has to be shifted given the cutting pattern will be presented. In section 3.3 we shall prove that the problem of obtaining the optimal cutting pattern is in general $NP$-hard. However, a solution to the unconstrained problem can be obtained by using linear programming techniques. For the constrained case a stochastic algorithm for obtaining the optimal cutting pattern $\pi$ will be described. It is based on the technique of simulated annealing which has been successfully used in similar problems [5],[6]. In section 3.4 the procedure for determining the rotation angle $\theta$ will be described and finally, experimental results, including execution times, can be found in section 3.5.
3.2 Determination of optimal translation

Efficient techniques for determining the optimal translation of the cutting structure for one kind of tiles and for different tiles for a prespecified cutting pattern will be given in sections 3.2.2 and 3.2.3 respectively. First, a brief discussion of the algebra of binary morphology and some useful operators which we will be using in the rest of this chapter will be given.

3.2.1 Morphological Operations

Mathematical morphological transformations apply to sets of any dimensions, like the Euclidean \( \mathbb{R}^2 \) or its discrete equivalent, the set of pairs of integers \( \mathbb{Z}^2 \), to which we shall refer from now on.

The primary morphological operations are dilation and erosion. If \( A \) and \( B \) are sets in \( \mathbb{Z}^2 \) with elements \( a \) and \( b \) respectively, \( a = (a_x, a_y) \) and \( b = (b_x, b_y) \) being pairs of coordinates, then the dilation of \( A \) by \( B \), denoted by \( A \oplus B \), is the set of all possible vector sums of pairs of elements:

\[
A \oplus B = \{ c \in \mathbb{Z}^2 | c = a + b, \ \forall a \in A, \ \forall b \in B \} \quad (3.7)
\]

Although, dilation is commutative, that is

\[
A \oplus B = B \oplus A, \quad (3.8)
\]

the first set \( A \) is normally referred to as the set undergoing processing, and the second set \( B \) as the structuring element.

Erosion is the morphological dual of dilation. It combines two sets by using containment as its basis set. The erosion of \( A \) by \( B \) is the set of all elements \( x \) for which \( x + b \in A, \ \forall b \in B \). It is denoted by \( A \ominus B \) and it is formally defined by

\[
A \ominus B = \{ x \in \mathbb{Z}^2 | x + b \in A, \ \forall b \in B \} \quad (3.9)
\]

Dilation can also be represented as a union of translates, whereas erosion as an intersection of negative translates as shown by equations 3.10 and 3.11 below:

\[
A \oplus B = \bigcup_{a \in A} B_a \quad (3.10)
\]

\[
A \ominus B = \bigcap_{b \in B} A_{-b} \quad (3.11)
\]
These representations help us to understand better how these operations can be implemented by digital computers. If we assume for simplicity that we have only one type of tiles to be cut from object $S$, then we may say that $A$ in the above expressions corresponds to the object $S$ to be cut and $B$, what is known as the eroding element, is the shape $R_0$ to be cut from $S$.

After this definition of erosion it becomes clear that all the points in $S \ominus R_0$ represent possible positions of cutting $R_0$ from $S$, ensuring that no defective or background points will be present in it.

In order to perform erosion by an element $(l, w)$, $l \times w$ intersections of negative translations are required, which is computationally expensive. However, we can take advantage of the chain rule of erosion [8]:

$$A \ominus (B \oplus C) = (A \ominus B) \ominus C$$

(3.12)

It can be seen that the above rule only holds if the structuring element is decomposable through dilation. Fortunately, this is not a problem in our case because a rectangular structuring element $R_0$ can very easily satisfy this condition as it is the result of the dilation of one of its columns $R_{oc}$ by one of its rows $R_{or}$, i.e. $R_0 = R_{oc} \oplus R_{or}$. By doing so we improve the complexity from $O(l \times w)$ to $O(l + w)$ and clearly the speed advantage is $(l \times w)/(l + w)$. In practice, it is not even needed to scan every single pixel in a row; once a background pixel is found we can simply abandon checking the remaining pixels. A pictorial illustration of the chain rule can be found in figure 3.2. In this example and in all our applications, the origin of the coordinate system is assumed to be at the top left corner of the image. The speed advantage of the chain rule is clear in the graph shown in figure 3.3 where we plot the CPU time needed to perform erosion versus the size of the single type tiles to be cut with and without the chain rule used. It should be noted, that although erosion is not a commutative operation it does not make any difference whether erosion is performed first by $R_{or}$ or $R_{oc}$, i.e.:

$$(S \ominus R_{or}) \ominus R_{oc} = (S \ominus R_{oc}) \ominus R_{or}$$

(3.13)

This can be shown as follows: Using the chain rule of erosion (equation 3.12) we have:

$$(S \ominus R_{or}) \ominus R_{oc} = S \ominus (R_{or} \oplus R_{oc})$$
3.2. DETERMINATION OF OPTIMAL TRANSLATION

Dilation is commutative (equation 3.8), therefore:

\[ S \odot (R_{or} \oplus R_{or}) = S \odot (R_{or} \oplus R_{or}) \]

and finally using the chain rule of erosion again we obtain:

\[ S \odot (R_{or} \oplus R_{or}) = (S \odot R_{or}) \odot R_{or} \]

Figure 3.2: Pictorial Illustration of the Erosion Chain Rule. Erosion by a rectangle (a) is equivalent to first eroding by a column (b) and then by a row (c).

Figure 3.3: The dashed line represents the CPU time required to perform erosion by a rectangle and the solid line to perform erosion by applying the chain rule.
3.2. DETERMINATION OF OPTIMAL TRANSLATION

3.2.2 Optimal translation for one type of rectangles

We shall now show how morphological operators can be used in order to solve the stock cutting problem when the shape to be cut is at a fixed orientation. First we shall consider the relatively simple problem of cutting just one type of rectangle, \( R_0 \) with length \( l \) and width \( w \), from \( S \). Let \( E_0 \) be the result of erosion of \( S \) by \( R_0 \):

\[
E_0 = S \ominus R_0
\]

Define two matrices \( C \) and \( O \) with elements \( c_{yx} \) and \( o_{yx} \) and dimensions \( W \times (w + w_c) \) and \( L \times (l + l_c) \) respectively:

\[
c_{yx} = \begin{cases} 
1 & \text{if } (y - x) \text{ mod } (w + w_c) = 0 \\
0 & \text{otherwise}
\end{cases}
\]

where \( 1 \leq x \leq (w + w_c) \) and \( 1 \leq y \leq W \).

\[
o_{yx} = \begin{cases} 
1 & \text{if } (y - x) \text{ mod } (l + l_c) = 0 \\
0 & \text{otherwise}
\end{cases}
\]

where \( 1 \leq x \leq (l + l_c) \) and \( 1 \leq y \leq L \).

Then the \( dx, dy \) values, for which the maximum number of tiles \( \xi_0 \) can be cut, are given by:

\[
a_{dy} dx = \max(a_{yx}), \quad \text{with } \xi_0 = a_{dy} dx
\]

where \( a_{yx} \) are the elements of matrix \( A \) defined by:

\[
A = C^T E_0 O
\]

with the superscript \( T \) denoting transpose. By this process we effectively partition the eroded matrix \( S \) into tiles (with the cutting width included) and add all the partitions. Clearly, the maximum element of the result corresponds to the number of tiles which had the particular element inside the eroded \( S \) and thus if the top left corner of a tile is placed at that position, the whole tile will be inside the non-eroded \( S \).

Let us now illustrate it by means of a numerical example: \( L = 7 \) and \( W = 6 \) are the length and width of the minimum enclosing rectangle of an object \( S \). If \( l = 3 \), \( w = 2 \) are the dimensions of the rectangle \( R_0 \), and the length and width of cutting \( l_c = w_c = 1 \), it is
3.2. DETERMINATION OF OPTIMAL TRANSLATION

required to find the optimal \( dx, dy \) values so that the wastage is minimised.

\[
S = \begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
\end{pmatrix}, \quad \text{and} \quad R_0 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix},
\]

\[
E_0 = S \ominus R_0 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix},
\]

\[
C^T E_0 \ O = \begin{pmatrix}
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
\end{pmatrix} \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix} \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{pmatrix}
\]

Then,

\[
A = C^T E_0 \ O = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & 2 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}
\]

It can be seen that \( \max(a_{yx}) = 2 \) for \((x, y) = (2, 2)\) or \((3, 2)\). The number \( a_{22} = a_{23} = 2 \) indicates that two pieces \( R_0 \) can be cut if shifting by the specified \( x, y \) values is performed. Clearly, there is not a unique optimal solution in every problem.

To summarise, to obtain the optimal translation in the vertical and horizontal direction the following 5 steps are needed:

**Step 1:** Represent the object by a binary matrix.

**Step 2:** Obtain \( E_0 \) by eroding the object matrix by the tile matrix.

**Step 3:** Form matrices \( C \) and \( O \) according to equation 3.15.

**Step 4:** Calculate \( A \) according to equation 3.18.

**Step 5:** The indices of the maximum element of \( A \) indicate the optimal shift required.
3.2.3 Optimal translation for a given cutting pattern

The methodology used to solve the 2-stage stock cutting problem for one type of rectangle, \( R_q \), can be generalised to the case where \( m \) different types of rectangles are to be cut from \( S \). Again, it is enough only to consider translation (shifting) of the cutting scheme within:

\[
1 \leq dx \leq l + l_c, \quad 1 \leq dy \leq w + w_c
\]

(3.19)

but in this case \( w \) is defined as the minimum tile width, i.e.:

\[
w = \min_{1 \leq i \leq m} (w_i).
\]

The matrix \( C \) now has to be redefined as \( C_{i\pi} \) to indicate its dependence on the cutting pattern \( \pi \) and on the different type of rectangle \( R_i \) we consider each time. Hence, for a given cutting pattern, we define a series of \( m \) matrices \( C_{i\pi} \) with elements \( c_{i\pi yx} \) defined for:

\[
1 \leq y \leq W, \quad 1 \leq x \leq w + w_c,
\]

(3.20)

according to the following:

\[
c_{i\pi yx} = 1 \quad \text{if} \quad \pi(j) = i \quad (3.21)
\]

for:

\[
y = \sum_{d=1}^{j} w_{\pi(d)} - w_{\pi(j)} + (j - 1) w_c + x,
\]

\[
y \leq W, \quad 1 \leq j \leq k
\]

\[
c_{i\pi yx} = 0 \quad \text{otherwise}.
\]

We shall demonstrate the above by means of a numerical example. An object of width \( W = 28 \) is to be cut into \( m = 3 \) different types of tiles of the same length \( l = 6 \), and widths \( w_1 = 4, w_2 = 6, w_3 = 3 \). Determine the three \( C_{i\pi} \) matrices given that the cutting pattern \( \pi \) is “132312” and the length and width of cutting is \( l_c = w_c = 1 \). Note that the maximum possible length of the cutting pattern is \( [W/(w_3 + w_c)] = 7 \) because tile type 3 has the minimum width. The \( C_{i\pi} \) matrices have dimensions \( 28 \times 4 \) and are shown in figure 3.4 (elements not shown are zero).

Matrix \( O \) is defined as it was defined before, in section 3.2.2. Further, we define the erosion \( E_i \) of \( S \) by the structuring element \( R_i \) as:

\[
E_i = S \ominus R_i.
\]

(3.22)

Then we have a different matrix \( A_i \) defined for each type of tile we want to cut:

\[
A_i = C_{i\pi}^T E_i O
\]

(3.23)
3.2. DETERMINATION OF OPTIMAL TRANSLATION

Figure 3.4: An example of determining the $C_{1x}$ matrices.

The idea here is that for mixed sizes of tiles, each time we add up only those partitions of the eroded $S$ which will be present if $S$ were to be cut according to the given cutting pattern. If we add up all matrices $A_i$, weighted by the width of the corresponding tile, we can determine a new matrix $Q$, the indices of the maximum element of which will give us the optimal shifting, and the number $\xi_i$ of tiles type $i$ that will be obtained:

$$q_{yx_{dx}} = \max(q_{yx})$$

(3.24)

where $q_{yx}$ is element of $Q = \sum_{i=1}^{m} A_i w_i$

(3.25)
3.3 Searching for the optimal pattern

In this section we address the problem of determining the optimal cutting pattern as well as the translation of the cutting scheme in both the vertical and horizontal directions at the same time. We shall first transform the problem in hand to a set-packing problem in order to prove its intractability. We shall then examine the unconstrained case (when there is no upper bound to the number of tiles to be cut from each kind) and show how the special properties of the problem make it easy to be solved by using well known linear programming methods like the simplex method [4]. However, the mathematical programming model for the general case can only be solved by using integer programming which is computationally expensive. For this reason the technique of simulated annealing is investigated and shown to produce good quality sub-optimal solutions.

3.3.1 Integer Programming Formulation

We shall now show that the stock cutting problem can be transformed to a series of \((l + l_v)\) set packing problems (hence being \(NP\)-hard). Each problem has \(O(Wm)\) variables and \(O(W)\) constraints. Let us first briefly explain what a set packing problem is. Given a set \(S = \{1, 2, 3, \ldots, m\}\) and a class \(\mathcal{F}\) of subsets of \(S\) "pack" as many of the members of \(\mathcal{F}\) into \(S\) as possible without overlap. For example let \(S = \{1, 2, 3, 4, 5, 6\}\) and \(\mathcal{F} = \{\{1, 2, 5\}, \{1, 3\}, \{2, 4\}, \{3, 6\}, \{2, 3, 6\}\}\). In order to form the integer programming model which describes the problem let us associate a variable \(\delta_i\) with every member of \(\mathcal{F}\)

\[
\delta_i = \begin{cases} 
1 & \text{if the } i\text{th member is packed} \\
0 & \text{otherwise}
\end{cases}
\]

Then the problem can be stated as follows:

Maximise

\[
\delta \equiv \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5
\]

subject to

\[
\begin{align*}
\delta_1 + \delta_2 &\leq 1, \\
\delta_2 + \delta_3 + \delta_5 &\leq 1, \\
\delta_3 + \delta_4 + \delta_6 &\leq 1, \\
\delta_1 &\leq 1, \\
\delta_4 + \delta_5 &\leq 1.
\end{align*}
\]

These inequality constraints arise from the fact that we do not allow any overlap. For example, since element 1 belongs to both the first and the second element of \(\mathcal{F}\), the no
3.3. SEARCHING FOR THE OPTIMAL PATTERN

Overlap condition requires that at most one of these two elements of \( T \) will be packed into \( S \). For this particular example an optimal packing would be \{1, 2, 5\} and \{3, 6\}. The properties of set packing problems are that they are described by maximisation integer programming models with all inequality constraints, where all RHS coefficients are 1 and all other matrix coefficients are either 0 or 1. It has been shown [12] that the optimal solution to set packing problems is always one vertex of the linear relaxation but not necessarily the optimal one.

It is easy to transform the unconstrained stock cutting problem into a set packing problem. First of all we assume that we know \( dx \) and only \( dy \) and the cutting pattern \( \pi \) are needed.

Let \( P_i \) be a series of \( i \) matrices with elements \( p_{iy} \) defined as:

\[
P_i = E_i O
\]

where \( E_i \) is the original image \( S \) eroded by \( R_i \), and \( O \) is the matrix defined by (3.16). Let set \( S \) have \( W \) elements i.e. \( S = \{1, \ldots, W\} \) which are the possible \( y \) positions at which a tile can be cut. We define \( m \) sets of subsets of \( S, \mathcal{F}_i \), one for each type of tile we want to cut. Set \( \mathcal{F}_i \) contains all sets of points of \( S \) which will be covered by a tile of type \( i \) if it is placed with its beginning at any of the possible \( y \) positions of \( S \).

\[
\mathcal{F}_i = \{ \{y, 1 + y, 2 + y, \ldots, w_i + y, w_i + y + 1, \ldots, w_i + y + w_e - 1\} | 1 \leq y \leq (W - w_i - w_e + 1) \}, 1 \leq i \leq m
\]

(3.27)

Therefore, there are \( \sum_{i=1}^{m} (W - w_i - w_e + 1) \) subsets and consequently the same number of variables, let us call them \( \delta_{iy} \), associated with the subsets. Each binary variable \( \delta_{iy} \), if set to 1, indicates that a tile type \( i \) is cut off object \( S \) at position \( y \). The problem can be stated as an integer programming one:

Maximise

\[
\sum_{i=1}^{m} \sum_{y=1}^{W-w_i-w_e+1} w_i \delta_{iy} p_{iyx}
\]

subject to:

\[
\sum_{i=1}^{m} \sum_{d=1}^{W-w_i-w_e+1} a_{idy} \delta_{iy} \leq 1, \quad 1 \leq y \leq W
\]

where

\[
a_{idy} = \begin{cases} 
1 & \text{if } (d \leq y) \text{ and } (d \geq y - w_i - w_e + 1) \\
0 & \text{otherwise}
\end{cases}
\]

(3.30)

where \( x = dx \) which is assumed to be known. The \( W \) constraints shown in equation 3.30 above are imposed in order to avoid overlapping in a similar way as described in the example given earlier.
For example if \( W = 10 \) and there are two kinds of tiles \((m = 2)\) of widths \( w_1 = 3 \) and \( w_2 = 5 \) and the cutting width is \( w_c = 1 \), the sets of subsets corresponding to each type of tile are:

\[
\mathcal{F}_1 = \{\{1, 2, 3, 4\}, \{2, 3, 4, 5\}, \{3, 4, 5, 6\}, \{4, 5, 6, 7\}, \{5, 6, 7, 8\}, \{6, 7, 8, 9\}, \{7, 8, 9, 10\}\} \\
\mathcal{F}_2 = \{\{1, 2, 3, 4, 5, 6\}, \{2, 3, 4, 5, 6, 7\}, \{3, 4, 5, 6, 7, 8\}, \{4, 5, 6, 7, 8, 9\}, \{5, 6, 7, 8, 9, 10\}\}
\]

We introduce 12 binary variables: 7 of them associated with tile type 1 \((\delta_{11}, \delta_{12}, \delta_{13}, \delta_{14}, \delta_{15}, \delta_{16}, \delta_{17})\) and 5 of them associated with tile type 2 \((\delta_{21}, \delta_{22}, \delta_{23}, \delta_{24}, \delta_{25})\). Clearly:

\[
\begin{align*}
\delta_{11} + \\
\delta_{11} + \delta_{12} + \\
\delta_{11} + \delta_{12} + \delta_{13} + \delta_{14} + \\
\delta_{12} + \delta_{13} + \delta_{14} + \delta_{15} + \\
\delta_{13} + \delta_{14} + \delta_{15} + \delta_{16} + \\
\delta_{14} + \delta_{15} + \delta_{16} + \delta_{17} + \\
\delta_{15} + \delta_{16} + \delta_{17} + \\
\delta_{16} + \delta_{17} + \\
\delta_{17} &
\end{align*}
\]

\[
\begin{align*}
\delta_{21} + \\
\delta_{21} + \delta_{22} + \\
\delta_{21} + \delta_{22} + \delta_{23} + \delta_{24} + \delta_{25} &
\end{align*}
\]

\[
\begin{align*}
\delta_{21} + \delta_{22} + \delta_{23} + \delta_{24} + \delta_{25} &
\end{align*}
\]

\[
\begin{align*}
\delta_{22} + \delta_{23} + \delta_{24} + \delta_{25} &
\end{align*}
\]

\[
\begin{align*}
\delta_{23} + \delta_{24} + \delta_{25} &
\end{align*}
\]

\[
\begin{align*}
\delta_{24} + \delta_{25} &
\end{align*}
\]

\[
\begin{align*}
\delta_{25} &
\end{align*}
\]

It is easy to understand the significance of the constraints in (3.31). Consider for example the first constraint. By forcing \( \delta_{11} + \delta_{21} \leq 1 \) it is guaranteed that the first \( y \) position of the object will be covered by either a tile of type 1, a tile of type 2 or will be left uncovered.

The second constraint allows for the second \( y \) position of the object to be covered either by a tile of type 1 or of type 2, cut from the first or second \( y \) position or left uncovered and so on.

Let us now give some useful definitions.

A square matrix \( A \) is unimodular if the determinant of all its singular sub-matrices is 1.

A matrix \( A \) is totally unimodular if every square submatrix of \( A \) is unimodular. It is well known [12] that the optimal solution of an Integer Programming Problem is the optimal solution of the corresponding^1 Linear Programming Problem if \( A \) is totally unimodular.

An \( m \times n \) \((0,1)\) matrix \( A \) is called an interval matrix [10] if in each column the 1's appear consecutively, in other words if \( a_{ij} = a_{kj} = 1 \) and \( k > i + 1 \), then \( a_{ij} = 1 \) for all \( l \) with \( i < l < k \). Interval matrices are totally unimodular [10].

^1The corresponding Linear Programming Problem is obtained by relaxing the integer requirement of variables.
3.3. SEARCHING FOR THE OPTIMAL PATTERN

There are a few interesting properties to our formulation of the stock cutting problem. Not only is it transformed to a set packing problem, but also the coefficient matrix $A$ can be shown to be totally unimodular due to its special triangular structure (figure 3.5) which makes it an interval matrix.

Although, this formulation assumes $dx$ to be known this is not a serious limitation since the same procedure can be applied for all possible $x$ positions in order to find the optimal one. Therefore, the optimal solution to the unconstrained problem can be obtained by just using standard linear programming techniques like the simplex method. However, the situation becomes more complicated if the constraints concerning the maximum number $b_i$ of tiles type $i$ to be cut are imposed. Then, the coefficient matrix is not totally unimodular and specialised algorithms for the set packing problem have to be used [1]. The technique of simulated annealing which will be described in the following subsection can then be used, although optimality is not guaranteed.

3.3.2 Simulated Annealing

Let us consider the configuration space of all possible cutting patterns. Clearly, only an exhaustive search will yield the optimal cutting pattern. Such a search, however, is prohibitively expensive even for moderately sized problems. To demonstrate our ideas
3.4 CHOOSING THE ROTATION ANGLE $\theta$

on the choice of the optimal shifting for a given pattern, we decided to adopt a stochastic approach to solve the full optimisation problem.

Let us assume that the object to be cut $S$ is at a certain orientation $\theta$ with respect to the cutting machine and let us choose a cutting pattern at random. We can find the optimal shifting of this pattern and compute $z$ from (3.5). By altering one element of the cutting pattern, we can create a new one for which we repeat the process. If the new cutting pattern improves $z$, we keep it; if it does not, we accept it with probability proportional to $\exp(\Delta z / T)$, where $\Delta z$ is the change in the value of $z$, defined as $\Delta z = z_{\text{new}} - z_{\text{old}}$, and $T$ is some parameter. Thus we stochastically explore the configuration space by creating a sequence of patterns each of which differs from the previous one by one element only most of the time moving towards a better value of $z$ but also taking some regressing steps which may allow us to escape from local optima. As the process proceeds, by lowering the value of $T$ gradually, we reduce and eventually eliminate the number of regressions we allow ourselves (since they may occur only when $\Delta z < 0$. This is the well known method of simulated annealing [9] and is known to produce good suboptimal solutions.

It should be noted that every time a new pattern is proposed it is not possible to know in advance whether the constraints will be satisfied or not. A possible way to enforce constraints would be simply during the calculation of the cost function not to take into account the exceeding pieces of the kind of tiles that violated the corresponding constraint. Experimental results show that this is an effective way of enforcing constraints.

### 3.4 Choosing the rotation angle $\theta$

In order to determine the optimal rotation angle we should repeat the whole process described in the previous section for every possible angle $\theta$. However, this would make the system extremely slow. We found experimentally that unless the sizes of the different types of tiles to be cut differ significantly, the optimal angle can be specified deterministically by sampling the possible range of its values every $5^\circ$ and computing the area utilisation using one type of tile only and in particular the smallest one.

This is illustrated in figure 3.6.a where the object shown in image 7 was rotated every $5^\circ$ from $-90^\circ$ to $90^\circ$ and the cost function calculated first by using simulated annealing and then by just assuming one kind of tiles, the smallest one. Clearly, both functions indicate where the most promising regions of rotating angles are. Note that there are two
3.5 Experimental Results

To test the algorithms presented in the previous sections we used the 8 images shown in figure 3.7 of resolution 512 by 512 pixels. Experimental results are summarised in table 3.1 where all 8 objects were optimised for different mix of (10 x 10), (10 x 20), and (10 x 30) tiles. The obtained result, the area utilisation, is a percentage derived as follows: 

\[
\text{Utilised Area} / \left( \text{Total Area} - \text{Defective Area} \right)
\]

For all cases the optimal shifting \(dx, dy\) was derived very fast (less than a second\(^3\)) given the cutting pattern, and the determination of the cutting pattern in less than a minute CPU time. Figure 3.8 shows the value of the cost function after rotating the object every 5\(^\circ\) from -45\(^\circ\) to 45\(^\circ\). In this case the smallest element was square so it would be superfluous to search from -90\(^\circ\) to 90\(^\circ\).

\(^3\)CPU time on a SPARC2 workstation.
Figure 3.7: Experimental results are based on a set of eight images.

Figure 3.8: Investigation of the Rotation angle.
3.6 Conclusions

In this chapter we formulated the 2-stage generalised stock-cutting problem of the guillotine type and showed how morphological operators can be used to choose the optimal
shifting for a given cutting pattern, when the object to be cut is of a general shape, and the objects to be cut from it are rectangles of various sizes. The procedure is very efficient and fast. In particular, the determination of the optimal shift takes only about $10^{-6}$ CPU seconds.

The problem of determining both the optimal cutting pattern and translation was solved by converting the problem to a set-packing problem, forming the integer programming model and finally, because of the special properties of the coefficient matrix, solved by just using the simplex method.

The constrained problem, however, can be solved using integer programming techniques which is computationally expensive for most real problems. For this purpose, an algorithm based on simulated annealing was found to be very efficient, yielding good results in less than a minute on a SPARC2 workstation for a 512 by 512 image. Some heuristics for determining the optimal orientation in order to speed up the system were also discussed.

In order to apply the powerful techniques for stock-cutting which have been developed so far in this thesis, an accurate way of determining the shape of the object to be cut is needed. For this purpose, stereo-vision techniques can be used. The rest of this thesis is devoted to several aspects related to the accurate and robust three-dimensional shape reconstruction.

References


REFERENCES


Chapter 4

Three Dimensional Reconstruction

One of the objectives of computer vision is the recovery of the 3D information lost by the process of recording a scene on the 2D image plane. This information cannot be recovered in general from one conventional intensity image only. At least two images are needed obtained either by the same camera and exploiting the principles of motion parallax\(^1\) [9, 8], or obtained by two stationary cameras and exploiting the principles of stereo vision. We are interested here in the latter approach and in particular in the problem of recovering the 3D shape of a block stone viewed from four cameras around it, placed at approximately 90° angular distance from each other. Each camera "sees" only three planes of the stone which has approximately the shape of a parallelepiped. The top plane is viewed by all four cameras but any side plane is viewed by only two adjacent cameras. So, the problem we are interested in, is trying to reconstruct the equation of a plane which is viewed by two cameras, call them left and right cameras.

4.1 Introduction

Stereo (or binocular) vision techniques rely on the fact that if the same object point is seen from two different positions, its three-dimensional position can be recovered. Imagine two optical systems which yield two images \(I(u, v)\) and \(I'(u', v')\) of a scene referenced to what is known as the world coordinate system \(O(x_w y_w z_w)\) as shown in figure 4.1. The main problems to be solved in 3D vision are the correspondence and the reconstruction problems. More specifically:

(a). Given a point \(m\) of coordinates \((u, v)\) in image \(I\), to which point \(m'\) of coordinates \((u', v')\) in image \(I'\) does it correspond?

---

\(^1\)for general motion parallax it can be shown that if two distinct points in 3D have the same image projection then the difference in their image displacements due to a change in the viewpoint depends only on the 3D translation or rotation between the views and the relative depth of the 3D points.
Figure 4.1: The problem of 3D vision.

(b). Given \( m \) and \( m' \) how do we compute in the \( O(x_w y_w z_w) \) coordinate system the coordinates of the point \( M \) of the scene that gave rise to images \( m \) and \( m' \)?

We are not concerned here with the correspondence problem which we assume is solved manually at this stage. We are mainly concerned about the recovery of the 3D position of points and lines from their 2D images by the two cameras. We shall adopt the widely used pinhole camera model according to which the world coordinates of a 3-D point \( \mathbf{M} = [x_w, y_w, z_w]^T \) and its image coordinates \( \mathbf{m} = [u, v]^T \) are related by

\[
\mathbf{s} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}
\]

where \( s \) is an non-zero arbitrary scale factor and \( \mathbf{P} \) the so called projection matrix. This model assumes that the relationship between the world coordinates and the pixel coordinates is linear projective. This is not a bad assumption since with the state of the art of the technology camera distortion is reasonably small [14]. The projection matrix is \( 3 \times 4 \) arbitrary matrix of rank 3 and since it is defined up to an arbitrary scale has 11 degrees of freedom and can be computed from 6 or more points. It can then be decomposed uniquely [10]:

\[
\mathbf{P} = \lambda \mathbf{A} \mathbf{D} \mathbf{E} \mathbf{D}_w
\]

where \( \alpha_u \) and \( \alpha_v \) the horizontal and vertical scaling, \( (u_o, v_o) \) the principal point (where the optical axis intersects the image plane) and \( \gamma \) accounts for the shear of the axes (more specifically \( \gamma = \tan \theta \) where \( \theta \) is the angle between the image axes. The elements of
matrix $A$ are called *intrinsic parameters* and the elements of matrix $D_w$ are called *extrinsic parameters*. A camera is said to be *calibrated* when $A$ is determined.

Camera calibration techniques [13, 3, 7] have been widely used. However, there are many disadvantages as a calibration grid is normally used which is not very practical in most cases. Pre-calibration of the cameras is not reliable since the camera may undergo small mechanical or thermal changes and therefore, the intrinsic parameters might change. A solution to these problems is offered by *Projective Geometry* [11, 12] approaches using invariants which have been employed in 3-D vision for the recovery of planar surfaces [1] and for reconstruction without using any calibration parameters [5].

In section 4.2 we first present an introduction to projective geometry [2], [4], [6] with some fundamental definitions and theorems and show how they can be used in order to solve the reconstruction problem in 3D vision. We also setup a trivial experiment and prove the validity of the method. Finally, in the same section, the 3D shape of a stone is recovered from four images. We draw our conclusions in section 4.4.

### 4.2 The Projective Geometry Approach

#### 4.2.1 Introduction to Projective Geometry

First, a few basic definitions and theorems, which will be used for projective reconstruction of points in 3D, are presented.

**4.2.1.1 Cross-ratio of four points**

If $A$, $B$, $C$ and $D$ are four co-linear points then their *cross-ratio* is defined as:

$$[A, B, C, D] = \frac{CA}{CB} \frac{DB}{DA} \hspace{1cm} \text{(4.1)}$$

where $AB$ is the *directed Euclidean distance* of the two points $A$ and $B$. This means that $AB = -BA$.

If the barycentric representation of the line is used, i.e.: $\vec{b} + \mu \vec{d} = \vec{r}$, where $\vec{r}$ is the position vector of any point along the line, $\mu$ is a parameter taking real values and $\vec{b}$ and $\vec{d}$ are the base and directional vectors of the line respectively, then the cross-ratio can be expressed in terms of the $\mu$ value of every point in the line $l$, that is:

$$[A, B, C, D] = \frac{\mu_C - \mu_A}{\mu_C - \mu_B} \frac{\mu_D - \mu_B}{\mu_D - \mu_A} \hspace{1cm} \text{(4.2)}$$
4.2. THE PROJECTIVE GEOMETRY APPROACH

where \( \mu_A \) is the value of the parameter \( \mu \) in the equation of the line for which point \( A \) is defined and \( \mu_B, \mu_C \) and \( \mu_D \) have similar interpretation.

The cross-ratio is the basic invariant in projective geometry since all other projective invariants can be derived from it. It has been shown [2] that any linear transformation in homogeneous coordinates — like perspective projection, linear scaling, skewing, rotation, translation, etc — preserves this cross-ratio.

4.2.1.2 Cross-ratio of a pencil of four coplanar lines

The cross-ratio of a pencil of four coplanar lines \( l_1, l_2, l_3, \) and \( l_4 \) going through \( O, \) is defined as the cross-ratio \( [A, B, C, D] \) of the points of intersection of the four lines with any line \( l \) not going through \( O. \)

![Figure 4.2: Cross-ratio of four lines.](image)

The invariance of the cross-ratio is illustrated by figure 4.2 where

\[
[A, B, C, D] = [E, F, G, H] = \frac{\sin A\hat{O}C \sin B\hat{O}D}{\sin A\hat{O}D \sin B\hat{O}C}
\] (4.3)

Indeed, if we consider the sine law for the triangles \( \hat{O}DB \) and \( \hat{O}DA, \) we have:

\[
\frac{DB}{\sin B\hat{O}D} = \frac{OD}{\sin O\hat{D}B}
\] (4.4)

and

\[
\frac{DA}{\sin A\hat{O}D} = \frac{OD}{\sin O\hat{A}D}
\] (4.5)

A pencil of lines is a set of lines intersecting at one point.
Similarly, from the triangles $OCA$ and $OCB$ we have:

\[
\frac{CA}{\sin \angle AOC} = \frac{OC}{\sin \angle OAD} \quad (4.6)
\]

and

\[
\frac{CB}{\sin \angle COB} = \frac{OC}{\sin \angle OBD} \quad (4.7)
\]

Upon solving the above four equations for $DB$, $DA$, $CA$ and $CB$ and substituting in equation (4.1), we obtain an expression independent of line $ABCD$, thus equation (4.3) is proven. We denote this characteristic number of the pencil of lines by $[l_1, l_2, l_3, l_4]$.

### 4.2.1.3 Projective Coordinates on a plane

Let A, B, C and D be four coplanar points, no three of them co-linear. These points are said to define a projective coordinate system in the plane, $\mathcal{P}$, they belong to. The projective coordinates $(k_1, k_2, k_3)$ of any point $P$ of $\mathcal{P}$ are the three real numbers defined as:

\[
k_1 = [CA, CB, CD, CP]
\]
\[
k_2 = [AB, AC, AD, AP]
\]
\[
k_3 = [BC, BA, BD, BP] \quad (4.8)
\]

Any point on $\mathcal{P}$ is uniquely referenced by its projective coordinates $k_1$, $k_2$ and $k_3$ with respect to the $\{A, B, C, D\}$ projective coordinate system.

Consider, for example point $P$ in figure 4.3. Given its projective coordinates in the $\{A, B, C, D\}$ projective coordinate system and the Cartesian coordinates of A, B, C and D it is a relatively easy task to determine the Cartesian coordinates of P. First, we recall from equation (4.8) that $k_1$ is the cross-ratio of the pencil of lines $CA$, $CB$, $CD$ and $CP$. Let us draw a line $l$ with equation $b + \mu d = \rho$ which intersects $CA$, $CB$, $CD$ and $CP$ at points $K$, $L$, $J$ and $M$ respectively. Then according to equation (4.2)

\[
k_1 = \frac{\mu_L - \mu_K}{\mu_J - \mu_L} \cdot \frac{\mu_M - \mu_L}{\mu_M - \mu_K} \quad (4.9)
\]

Therefore,

\[
\mu_M = \frac{k_1 \mu_K (\mu_J - \mu_L) + \mu_L (\mu_K - \mu_J)}{k_1 (\mu_J - \mu_L) + \mu_K - \mu_J} \quad (4.10)
\]
4.2. THE PROJECTIVE GEOMETRY APPROACH

Having obtained $\mu_M$, the Cartesian coordinates of $M$ can be found — by replacing $\mu$ by $\mu_M$ in the equation of line $l$ — and thus the equation of the line defined by the points $C$ and $P$. In exactly the same way the equations of lines $AP$ and $BP$ can be obtained. The Cartesian coordinates of $P$ are given by the intersection of the three lines $AP$, $BP$ and $CP$. It is obvious that to obtain the Cartesian coordinates of $P$ any two lines of $AP$, $BP$ or $CP$ are enough, thus only two of $k_1$, $k_2$ or $k_3$ are needed. However, in some cases where degeneration occurs (when point $P$ is collinear with any two reference points) the third value is necessary.

![Figure 4.3: Projective coordinates $k_1$, $k_2$ and $k_3$ of a point.](image)

4.2.1.4 Cross-ratio of a pencil of four planes

The cross-ratio of a pencil\(^8\) of four planes $P_1$, $P_2$, $P_3$ and $P_4$, is defined as the cross-ratio of their four lines of intersection with any plane not passing through the common line of the pencil.

To see this, consider a plane $\mathcal{P}$ which is perpendicular to the common line $l$ of the pencil of planes (figure 4.4). On that plane consider any line $a$ which intersects planes $P_1$, $P_2$, $P_3$ and $P_4$ at points $A$, $B$, $C$ and $D$ respectively. There is a plane passing through $a$ and parallel to line $l$. This plane is perpendicular to plane $\mathcal{P}$ and any line $\tilde{a}$ on it intersects planes $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ and $\mathcal{P}_4$ at points $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ and $\tilde{D}$ which project to points $A$, $B$, $C$ and $D$. From the fact that lines $A\tilde{A}$, $B\tilde{B}$, $C\tilde{C}$ and $D\tilde{D}$ are parallel it is obvious that the cross-ratio of the plane defined by line $\tilde{a}$ and a point $O$ on $l$ will be

$$\frac{\tilde{C}\tilde{A} \cdot \tilde{D}\tilde{B}}{\tilde{C}B \cdot \tilde{D}\tilde{A}} = \frac{CA \cdot DB}{CB \cdot DA} \quad (4.11)$$

\(^8\)A pencil of planes is a set of planes intersecting at a common line
which is independent of the choice of line \( \tilde{a} \) and the plane through it, and independent of the choice of line \( a \), since it is equal to the cross-ratio of the pencil of lines on plane \( P \) which only depends on the pencil of planes \( P_1, \ldots, P_4 \).

![Figure 4.4: Cross-ratio of a pencil of four planes.](image)

### 4.2.1.5 Preservation of the cross-ratio by perspective projection

We are ready to show now that the cross-ratio is preserved by perspective projection. For simplicity we adopt the pinhole camera model with a frontal image plane (figure 4.5).

![Figure 4.5: The cross-ratio is preserved by perspective projection.](image)

We assume that we are viewing down the negative \( z \) axis with the centre of projection being the origin of the coordinate system \( O \). The coordinate axes \( Ox \) and \( Oy \) are parallel to the image plane which is based at a distance \( f \) (the focal length of the camera) away from the origin and in the direction of viewing.
4.2. THE PROJECTIVE GEOMETRY APPROACH

Let us assume that we are observing four points $A, B, C$ and $D$ which belong to the same plane $P$ that is intersected by the optical axis at $O''$. These points project on the image plane to points $a, b, c$ and $d$ respectively. As all the projection lines meet at the centre of projection $O$, each one of the four points $A, B, C$ and $D$ defines a plane with the optical axis, and all these four planes form a pencil of planes with common line the optical axis itself. Thus the cross-ratio of the four lines $O''A, O''B, O''C$ and $O''D$ is the same as the cross-ratio of the lines $O'a, O'b, O'c$ and $O'd$.

Note that even if line $OO''$ was not the optical axis but simply another viewing line, the cross-ratio between any four lines defined with the help of any four of the points and using the fifth as origin, would still be preserved because lines $OA, OB, OC, OD$ and $OO''$ have a common point (the centre of projection) and thus they always define a pencil of planes.

4.2.2 Projective Reconstruction using planar points

In this section we briefly describe how projective geometry techniques can be used to reconstruct a point $P$ in the world coordinate system, given its left and right image coordinates as well as the exact position and correspondences of a set of eight reference points $\{A, B, C, D\}$ and $\{G, H\}$, consisting of two sets of 4 coplanar points, $\{A, B, C, D\}$ and $\{G, H\}$. The technique will be further explained by means of a numerical example which can be found in the next section.

4.2.2.1 Finding the viewing line

First, the equation of the viewing line $OP$ (figure 4.6) has to be determined. Consider the first set of reference points $\{A, B, C, D\}$ and their projections to the left image plane $\{a, b, c, d\}$. The projective coordinates $k_1$, $k_2$ and $k_3$ of image point $p$ with respect to the $\{a, b, c, d\}$ projective coordinate system can be determined according to equation (4.8).
4.3. EXPERIMENTAL RESULTS

If \( P_1 \) is the intersection of the viewing line \( OP \) with the \( ABCD \) plane, the coordinates of \( P_1 \) can be determined. In view of the result of section 4.2.1.5, the projective coordinates of \( p \) in the image plane with respect to \( \{ a, b, c, d \} \) are the same as the projective coordinates of \( P_1 \) in the \( ABCD \) plane with respect to \( \{ A, B, C, D \} \) because of the cross-ratio invariance under perspective projection [6]. Therefore, since the exact positions of \( A, B, C \) and \( D \) are known, the method described in section 4.2.1.3 can be used to determine the world coordinates of \( P_1 \).

In a similar way the coordinates of point \( P_2 \) can be calculated. The two points \( P_1 \) and \( P_2 \) are enough to define uniquely the equation of the viewing line \( OP \).

4.2.2.2 Reconstructing the point \( P \)

In the previous section we showed how the viewing line for the left image can be obtained. Working in exactly the same manner the right viewing line \( O'P \) can also be determined. Then, it is trivial to find \( P \) as it is the point of intersection of the two viewing lines.

4.3 Experimental Results

In order to test the projective geometry approach to 3D reconstruction we set up the scene shown in figure 4.7. This scene is very similar to the real problem we are interested in, namely the reconstruction of a stone block. The stone blocks will be placed on a parallelepiped platform which resembles the box of figure 4.7. The objective of this experiment is to exemplify the method which will be used in the experiments with real stones and check its accuracy. For this purpose we shall determine the 3D world coordinates of the line segment \( PQ \) without knowing any camera calibration parameters.
Point \( Q \) is one of the vertices of the box and \( P \) is a point chosen at random on the top plane of the box.

The coordinates of all 8 points, \( \{A, B, C, D, E, F, P, Q\} \), and their projections to the left and right images, \( \{a, a', b, b', c, c', d, d', e, e', f, f', p, p', q, q'\} \) respectively, are known. All known coordinates are given in table 4.1. The left-to-right correspondences were found by hand but generally they could have been obtained by some of the methods given in chapter 6. A set of six reference points \( \{A, B, C, D, E, F\} \) will be used in order to determine the position of the line segment \( PQ \). Four of them, \( \{A, B, C, D\} \), belong to the same plane, namely plane 1, and form a projective coordinate system. Points \( \{C, D, E, F\} \) belong to a different plane and they will be used to set up a different projective coordinate system. These two coordinate systems will help us calculate the world coordinates of points \( P \) and \( Q \). The known values of the coordinate positions of these points will be used to check the accuracy of the method.
### Table 4.1: Coordinates of Reference Points (image coordinates measured with 1 pixel accuracy and world coordinates with 0.1 cm accuracy).

<table>
<thead>
<tr>
<th>Point</th>
<th>Left x</th>
<th>y</th>
<th>Right x</th>
<th>y</th>
<th>World x</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>139</td>
<td>116</td>
<td>161</td>
<td>54</td>
<td>12.9</td>
<td>0.0</td>
</tr>
<tr>
<td>B</td>
<td>147</td>
<td>422</td>
<td>125</td>
<td>344</td>
<td>12.9</td>
<td>0.0</td>
</tr>
<tr>
<td>C</td>
<td>438</td>
<td>491</td>
<td>207</td>
<td>474</td>
<td>12.9</td>
<td>16.3</td>
</tr>
<tr>
<td>D</td>
<td>439</td>
<td>163</td>
<td>250</td>
<td>145</td>
<td>12.9</td>
<td>16.3</td>
</tr>
<tr>
<td>E</td>
<td>598</td>
<td>111</td>
<td>545</td>
<td>166</td>
<td>0.0</td>
<td>16.3</td>
</tr>
<tr>
<td>F</td>
<td>594</td>
<td>410</td>
<td>490</td>
<td>481</td>
<td>0.0</td>
<td>16.3</td>
</tr>
<tr>
<td>P</td>
<td>330</td>
<td>108</td>
<td>320</td>
<td>93</td>
<td>6.8</td>
<td>6.3</td>
</tr>
<tr>
<td>Q</td>
<td>314</td>
<td>70</td>
<td>421</td>
<td>74</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

4.3.0.3 Finding the viewing line OP

Since the positions of the images $a$, $b$, $c$, $d$ and $p$ are known in the left image, the projective coordinates of $p$ in the $\{a, b, c, d\}$ projective coordinate system can be calculated:

$$k_1 = \begin{bmatrix} a_p, b_p, c_p, d_p \end{bmatrix} = 1.45$$

$$k_2 = \begin{bmatrix} a, b, c, d \end{bmatrix} = 1.17$$

$$k_3 = \begin{bmatrix} b, c, a, d \end{bmatrix} = 0.59$$

Using the cross-ratio invariance property, we know that the projective coordinates of $P_1$ in the plane defined by the points $A$, $B$, $C$ and $D$ are the same as those of $p$ in the plane defined by the points $a$, $b$, $c$ and $d$, that is:

$$[CA, CB, CD, CP_1] = [ca, cb, cd, cp] = 1.45$$

$$[AB, AC, AD, AP_1] = [ab, ac, ad, ap] = 1.17$$

$$[BC, BA, BD, BP_1] = [bc, ba, bd, bp] = 0.59$$

Since the coordinates of $A$, $B$, $C$ and $D$ are known, as well as the projective coordinates of $P_1$, it is possible to determine the exact position of $P_1$, which turns out to be $(12.90, 10.68, 15.49)$.

In a similar way, the position of $P_2$ can be found. This time the vertices of plane 2 of the parallelepiped will be used, namely $C$, $D$, $E$ and $F$. The coordinates of $P_2$ are $(20.41, 16.30, 17.42)$. 
4.3. EXPERIMENTAL RESULTS

<table>
<thead>
<tr>
<th>Viewing line</th>
<th>Base vector</th>
<th>Direction vector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>OP</td>
<td>12.9</td>
<td>10.7</td>
</tr>
<tr>
<td>OQ</td>
<td>12.9</td>
<td>9.8</td>
</tr>
<tr>
<td>O'P</td>
<td>12.9</td>
<td>24.5</td>
</tr>
<tr>
<td>O'Q</td>
<td>12.9</td>
<td>35.1</td>
</tr>
</tbody>
</table>

Table 4.2: The equations of the four viewing lines.

The equation of the viewing line, which goes through the left origin $O$, $P_i$, $P_j$ and $P$ can be calculated, since the position of $P_i$ and $P_j$ has been determined.

4.3.0.4 Finding the other viewing lines

The equations of the remaining three viewing lines, namely: $OQ$, $O'P$ and $O'Q$ can be found by using the same procedure as the one described in the previous section. The base and direction vectors of all four viewing lines can be found in table 4.2.

4.3.0.5 Recovering the position of the line segment $PQ$

In order to reconstruct the line segment $PQ$, the equations of the two planes defined by the points $\{P_1, P_2, Q_2\}$ and $\{P'_1, P'_2, Q'_2\}$, are first determined (see figure 4.8). The normals to the two planes $n$ and $n'$ as well as the scalars $k$ and $k'$ are given below.

\[
\begin{align*}
    n &= (9.99, 11.20, -6.26) \\
    k &= -87.73 \\
    n' &= (-6.65, 8.46, -27.72) \\
    k' &= -378.60
\end{align*}
\]

Having obtained the equations of the left and right planes it is straightforward to find the equation of the line of intersection of them namely $PQ$. The point of intersection of the viewing line $OP$ and the plane intersection line $PQ$ is obviously the point $P$ whose coordinates have been found to be $(6.84, 6.14, 13.94)$. In a similar way the coordinates of $Q$ are determined: $(-0.21, -0.08, 13.83)$. Table 4.3 summarises the experimental results.

4.3.0.6 Recovering the shape of a stone

In this section we shall apply the techniques described in the previous sections in order to estimate the 3D shape of a stone, given its four images separated by 90° from each
4.3. EXPERIMENTAL RESULTS

Figure 4.8: Reconstruction of PQ.

<table>
<thead>
<tr>
<th>Point</th>
<th>World Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>P calculated</td>
<td>6.8 6.1 13.9</td>
</tr>
<tr>
<td>P ground truth</td>
<td>6.8 6.3 13.9</td>
</tr>
<tr>
<td>Q calculated</td>
<td>-0.2 -0.0 13.8</td>
</tr>
<tr>
<td>Q ground truth</td>
<td>0.0 0.0 13.9</td>
</tr>
</tbody>
</table>

Table 4.3: Experimental Results (note that the ground truth was measured with 0.1cm accuracy).

other as shown in figures 4.10, 4.11, 4.12 and 4.13. Note that the resolution of the images is 768 by 512 pixels. The stone is placed on a platform whose diagram can be found in figure 4.9.

Let us define our world coordinate system to be centered at $B$ with $BD$ the $x$ axis, $BH$ the $y$ axis and $BA$ the $z$ axis. Then, since the dimensions of the platform are known the coordinates of all 8 vertices of the platform in $B(xyz)$ system can be obtained and the results are shown in table 4.4. These 8 points, namely $A, B, C, D, E, F, G$ and $H$, whose world coordinates $x$, $y$ and $z$ are known, can be used as our reference points. Indeed, they form 4 projective coordinate systems: $\{A, B, C, D\}$, $\{C, D, E, F\}$, $\{E, F, G, H\}$ and $\{G, H, A, B\}$. Projective coordinate systems $\{A, B, C, D\}$ and $\{G, H, A, B\}$ can be seen by the first camera and can be used to find the equation of viewing lines for the first image.

Similarly, projective coordinate systems $\{C, D, E, F\}$ and $\{E, F, G, H\}$ can be used for the second image, projective coordinate systems $\{E, F, G, H\}$ and $\{G, H, A, B\}$ for the
4.3. EXPERIMENTAL RESULTS

![Figure 4.9: Drawing of the platform (all dimensions in mm).](image)

<table>
<thead>
<tr>
<th>Point</th>
<th>World Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>x 0   y 0  z 610</td>
</tr>
<tr>
<td>B</td>
<td>x 0   y 0  z 0</td>
</tr>
<tr>
<td>C</td>
<td>x 2955 y 0  z 610</td>
</tr>
<tr>
<td>D</td>
<td>x 2955 y 0  z 0</td>
</tr>
<tr>
<td>E</td>
<td>x 2955 y 3100 z 610</td>
</tr>
<tr>
<td>F</td>
<td>x 2955 y 3100 z 0</td>
</tr>
<tr>
<td>G</td>
<td>x 0   y 3100 z 610</td>
</tr>
<tr>
<td>H</td>
<td>x 0   y 3100 z 0</td>
</tr>
</tbody>
</table>

Table 4.4: Platform corner coordinates (measured with 0.1cm accuracy).

third image, and finally projective coordinate systems \{G, H, A, B\} and \{A, B, C, D\} for the fourth image.

The same techniques described in the earlier experiment can be applied now in order to obtain the 3D information of the stone we need. We shall try to obtain the width, height and breadth of the stone and compare them with the already known values which were obtained by physically measuring the stone. We shall try to use exactly the same points as the ones that were used by the workman who measured the stone. For the object of this experiment we assume that the correspondences are given so we can proceed to the reconstruction stage. We have implemented the algorithms in C language, run on a SPARC2 workstation and obtained the results shown in table 4.5.

The numbers obtained are in about 5% discrepancy with the measured values. One
4.3. EXPERIMENTAL RESULTS

Figure 4.10: First viewpoint.

Figure 4.11: Second viewpoint.
4.3. EXPERIMENTAL RESULTS

Figure 4.12: Third viewpoint.

Figure 4.13: Fourth viewpoint.
4.4 Conclusions

In this chapter we have presented a method for reconstructing points in the 3D space, given their left and right images, which does not require any camera calibration parameters. Indeed, no camera intrinsic or extrinsic parameters (pixel size, focal length, determination of the optical centre, position of the camera) are required if projective geometry based techniques are used.

The accuracy of the technique primarily depends on the accuracy with which the coordinates of the reference points are measured. It also depends on the camera resolution, since the higher the resolution the more accurately the projective coordinates of every point can be calculated. The mathematics involved is trivial making the method a very easy technique to implement. Thus, the algorithm is very fast, with typically 1000 points reconstructed in less than 3 seconds CPU time on a SPARC2 workstation.

source of error is the poor correspondence between the points of the two images. Indeed, the correspondence problem, even when it has been solved manually, becomes very difficult when the cameras are separated by 90 degrees. The two views are very different from each other and corresponding pixels on the left and right images of the same feature are very difficult to be identified within pixel accuracy.

However, by far the most significant source of error is the ground "truth" itself. The measured dimensions of the stone, are not really uniquely determined. Different people obtain different measurements. To prove this we tried the algorithm on a known point on the platform. The known coordinates were (620, 0, 610) and we obtained (608.01, 8.72, 604.94) which indicates an error of less than 0.5% (note that the error was calculated in terms of the linear dimensions of the platform which was approximately 3000mm, thus for example the percentage error for the z coordinate is given by (610-604.94)/3000 which can be found to be 0.16%).

<table>
<thead>
<tr>
<th>Dimensions (mm)</th>
<th>Width</th>
<th>Breadth</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculated</td>
<td>1425</td>
<td>2490</td>
<td>1366</td>
</tr>
<tr>
<td>Ground truth</td>
<td>1520</td>
<td>2600</td>
<td>1410</td>
</tr>
</tbody>
</table>

Table 4.5: 3D reconstruction of the stone (note that the ground truth was measured with 2cm accuracy).
The accuracy of the technique described in this section can be further increased if more reference points are used. Indeed, more reference points would define more projective coordinate systems which means that more than 2 points would be available in order to reconstruct each viewing line. Least square techniques could then be used in order to find the equation of the viewing line. However, this option is not of great interest in an industrial environment, where, as mentioned before, the number of reference points to be used should be as small as possible.

Since the projective-geometry based reconstruction technique seemed able to reconstruct accurately points in 3D, several test images were tried and the results were compared to the ground truth which was made available to us by the factory workers who manually measured the coordinates of the points in 3D or the relative distances between them. It was then noted that although the technique could reconstruct with better than 1% accuracy in most cases, there were a few situations where the error exceeded 1000% or in some cases even 2000%. It was then obvious that there were cases where the method degenerated and as a result it was very sensitive to input noise.

To test this, it was decided to setup a simple experiment. The 3D world coordinates of 1000 points were generated by sampling a 3 × 3 × 3 meter cube every 10cm³. These coordinates were then projected onto the left and right images by using appropriate transformation matrices which simulated two cameras set up at a height of 7 meters separated by 90 degrees and looking towards the cube. The projected coordinates were then sampled at sub-pixel accuracy. We repeated the same procedure for 6 reference points which simulated the platform on which the stone is placed. We then reconstructed the points using projective geometry and compared the values of the reconstructed coordinates of the 1000 points with the known values. No significant errors were observed. We repeated the experiment, but this time Gaussian noise of 1 pixel standard deviation was added to the image coordinates of all points (including the 6 reference points). It was then observed that some regions of the cube could be reconstructed with great accuracy which was comparable to the accuracy obtained without added noise. Nevertheless, some other regions were reconstructed with great error and a few others with huge error. Apparently, different regions of the 3D object to be reconstructed had different immunity to noise, with some of them being extremely sensitive.

It was then obvious, that if one wanted to use the projective-geometry based reconstruction technique certain things should be taken into account which could guarantee
robust 3D reconstruction. The task of further discovering the limitations and applicability constraints of the technique seemed to be necessary and this will be the topic of the next chapter.

References


Chapter 5

Projective Geometry Based Reconstruction

Projective geometry approaches to 3-D vision have been used before for the recovery of planar surfaces [2, 3, 1] and for reconstruction without using any calibration parameters [4]. The Projective Geometry approaches rely on the knowledge of the coordinates of two sets of 4 coplanar points in the world coordinate system as well as their projections onto the image plane. Although, in general, the minimum number of 8 reference points is needed one can easily see that just 6 reference points are enough in special cases. This special configuration can be obtained if the two sets of 4 coplanar points have 2 points in common; in particular these common points should lie in the line of intersection of the two reference planes.

A good option offered to us is that of using the platform on which the stone stands. The platform is a robust structure (designed to support stones of granite) and can be made to have some desired characteristics like to be a right angled parallelepiped with well defined vertices, and faces painted with distinct colours for easy identification. The world coordinate position of each vertex is assumed to be known. Thus in each image, we shall have a set of 6 reference points (points A, B, C, D, E and F in figure 1.1) forming two planes intersecting along a line in the image.

The method of Projective Geometry can therefore be employed for the 3D reconstruction of the granite block. The user requirement, however, is that the reconstruction error should be minimal in terms of the linear dimensions of the block. Given that there is always some inaccuracy in the estimation of the location of vertices in an image and certainly inaccuracy in the construction of the platform (in terms of which the world coordinate system is defined) and the measurement of its vertices (the reference points),
it is very important to check whether the adopted method can be relied upon to cope with uncertainties. Detailed experimentation we performed with settings where ground truth was known, showed that there were cases where the error exceeded 2000%! This motivated us to perform careful error analysis for the various stages of the Projective Geometry method the results of which guided us in choosing the best possible configuration for our application. A detailed error analysis for the Projective Geometry approach will be presented and next we describe the way its results guided us in solving our problem given the constraints imposed by the factory environment.

5.1 Error Analysis

Although the 3D reconstruction technique based on Projective Geometry [6, 7] described in the previous chapter is accurate and computationally simple, it is very sensitive to noise. The point is that the equations involved are non-linear and thus the propagation of error is not straightforward. Indeed, in non-linear equations it is often the case that the error in the computed quantity is not only a function of the error in the measured quantity times a constant, but it also depends on the computed value itself. Thus, there may be ranges of values for which the error is unacceptably amplified. We shall discuss here the way the error propagates at each stage of the reconstruction process, starting from the estimation of the error in the calculation of the projective coordinates on the image plane, and finishing with the estimation of the error in the calculation of the 3D position of point $P$ [5].

5.1.1 Error in the calculation of the projective coordinates

Let us consider the projections of the four reference points on the image plane $a$, $b$, $c$ and $d$ and the projection $p$ of the point whose 3D position we want to determine. The first projective coordinate of $p$ with respect to the Cartesian coordinates of $a$, $b$, $c$ and $d$, computed from the pencil of lines with vertex $a$ (figure 5.1), will be derived next.

Let the position vectors of $a$, $b$, $c$, $d$, and $p$ be

$$
\begin{align*}
\vec{a} &= \begin{bmatrix} a_x & a_y \end{bmatrix}^T \\
\vec{b} &= \begin{bmatrix} b_x & b_y \end{bmatrix}^T \\
\vec{c} &= \begin{bmatrix} c_x & c_y \end{bmatrix}^T \\
\vec{d} &= \begin{bmatrix} d_x & d_y \end{bmatrix}^T
\end{align*}
$$
Figure 5.1: Determination of $k_2$ by calculating the cross-ratio of the pencil of lines $ab$, $ac$, $ad$ and $ap$.

\[ \vec{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}^T \]

where \((a_x, a_y), (b_x, b_y), (c_x, c_y), (d_x, d_y),\) and \((p_x, p_y)\) are the Cartesian coordinates of points \(a, b, c, d\) and \(p\) respectively with the superscript \(T\) denoting transpose. Then the equation of the lines $ab$, $ac$, $ad$ and $ap$ in the image plane can be written as:

\[
\begin{align*}
\vec{p}_{ab} &= \vec{a} + \mu_{ab}(\vec{b} - \vec{a}) \\
\vec{p}_{ac} &= \vec{a} + \mu_{ac}(\vec{c} - \vec{a}) \\
\vec{p}_{ad} &= \vec{a} + \mu_{ad}(\vec{d} - \vec{a}) \\
\vec{p}_{ap} &= \vec{a} + \mu_{ap}(\vec{p} - \vec{a})
\end{align*}
\]

where $\vec{p}_{ij}$ is the position vector of any point along the line defined by points \(i\) and \(j\), and $\mu_{ij}$ takes real values. Let $\vec{p}_R = \vec{b}_R + \mu_R \vec{d}_R$ be the equation of any line on the same plane not going through $a$. In order to calculate $k_2$, the $\mu$ values of the intersection points of this line with the lines $ab$, $ac$, $ad$ and $ap$ must be found: Consider first the point of intersection of line $ab$ with $\vec{b}_R + \mu_R \vec{d}_R$, let us call its position vector $\vec{j}$. The point of intersection of the two lines, $\vec{j}$, must satisfy the equations of lines $ab$ and $\vec{b}_R + \mu_R \vec{d}_R$:

\[
\begin{align*}
\vec{b} - \vec{a} &= \mu_{ab}(\vec{b} - \vec{a}) \\
\vec{b} - \vec{a} &= \vec{b}_R + \mu_R \vec{d}_R
\end{align*}
\]

Therefore,

\[ \vec{a} + \mu_{ab}(\vec{b} - \vec{a}) = \vec{b}_R + \mu_R \vec{d}_R \]
or in vector component form:

\[ a_x + \mu_{ab} (b_x - a_x) = b_{Rx} + \mu_R d_{Rx} \]
\[ a_y + \mu_{ab} (b_y - a_y) = b_{Ry} + \mu_R d_{Ry} \]

Solving for \( \mu_R \) of point \( j \), let us call it \( \mu_j \), we get:

\[ \mu_j = \frac{a_x b_y - a_y b_x + b_{Ry} b_y - b_{Rx} b_y - b_{Ry} a_x + b_{Rx} a_y}{d_{Rx}(b_y - a_y) - d_{Ry}(b_x - a_x)} \]  (5.1)

Similarly \( \mu_k, \mu_l \) and \( \mu_m \) can be calculated:

\[ \mu_k = \frac{a_x c_y - a_y c_x + b_{Ry} c_y - b_{Rx} c_y + b_{Ry} a_x - b_{Rx} a_y}{d_{Rx}(c_y - a_y) - d_{Ry}(c_x - a_x)} \]  (5.2)
\[ \mu_l = \frac{a_x d_y - a_y d_x + b_{Ry} d_y - b_{Rx} d_y - b_{Ry} a_x + b_{Rx} a_y}{d_{Rx}(d_y - a_y) - d_{Ry}(d_x - a_x)} \]  (5.3)
\[ \mu_m = \frac{a_x p_y - a_y p_x + b_{Ry} p_y - b_{Rx} p_y + b_{Ry} a_x - b_{Rx} a_y}{d_{Rx}(p_y - a_y) - d_{Ry}(p_x - a_x)} \]  (5.4)

Then the projective coordinate \( k_3 \) is given by the cross-ratio:

\[ k_3 = \frac{(\mu_l - \mu_j)(\mu_m - \mu_k)}{(\mu_m - \mu_j)(\mu_l - \mu_k)} \]  (5.5)

Substituting the values of \( \mu_j, \mu_k, \mu_l \) and \( \mu_m \) to equation (5.5) we obtain:

\[ k_3 = \frac{(a_x c_y - a_y c_x + c_x p_y + c_y p_x - a_y c_x - p_x c_y)(a_x b_y - a_y b_x + b_x d_y - b_y d_x - a_y b_x + a_x d_y)(a_x c_y - a_y c_x + c_x d_y + c_y d_x - d_x c_y - a_y c_x)(a_x b_y - a_y b_x + b_x p_y - b_y p_x - a_y b_x + a_x p_y)}{(a_x b_y - a_y b_x + b_x d_y - b_y d_x - a_y b_x + a_x d_y)(a_x c_y - a_y c_x + c_x d_y + c_y d_x - d_x c_y - a_y c_x)(a_x b_y - a_y b_x + b_x p_y - b_y p_x - a_y b_x + a_x p_y)} \]  (5.6)

We note that the above expression for \( k_3 \) can be written in terms of some determinants:

\[ k_3 = \frac{k_{2a} k_{2b}}{k_{2c} k_{2d}} \]  (5.7)

where:

\[ k_{2a} = \begin{vmatrix} a_x & c_x \\ a_y & c_y \end{vmatrix} + \begin{vmatrix} c_x & p_x \\ c_y & p_y \end{vmatrix} + \begin{vmatrix} p_x & a_x \\ p_y & a_y \end{vmatrix} \]  (5.8)
\[ k_{2b} = \begin{vmatrix} a_x & b_x \\ a_y & b_y \end{vmatrix} + \begin{vmatrix} b_x & d_x \\ b_y & d_y \end{vmatrix} + \begin{vmatrix} d_x & a_x \\ d_y & a_y \end{vmatrix} \]  (5.9)
\[ k_{2c} = \begin{vmatrix} a_x & c_x \\ a_y & c_y \end{vmatrix} + \begin{vmatrix} c_x & d_x \\ c_y & d_y \end{vmatrix} + \begin{vmatrix} d_x & a_x \\ d_y & a_y \end{vmatrix} \]  (5.10)
\[ k_{2d} = \begin{vmatrix} a_x & b_x \\ a_y & b_y \end{vmatrix} + \begin{vmatrix} b_x & p_x \\ b_y & p_y \end{vmatrix} + \begin{vmatrix} p_x & a_x \\ p_y & a_y \end{vmatrix} \]  (5.11)
The above equation can be further simplified if we define the \( i \)th order \( CD \) function:

\[
CD(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} \left| \begin{array}{cc} a_{ix} & a_{iy} \\ a_{iy} & a_{iy} \end{array} \right|, \quad \text{with } j = ((i + 1) \mod n) \tag{5.12}
\]

where \( a_i \) are 2-D vectors, i.e.:

\[
a_i = \begin{bmatrix} a_{ix} & a_{iy} \end{bmatrix}^T \tag{5.13}
\]

Then, clearly,

\[
\begin{align*}
k_{ab} &= CD(a, c, p) \\
k_{ac} &= CD(a, b, d) \\
k_{ad} &= CD(a, c, d) \\
k_{bd} &= CD(a, b, p)
\end{align*}
\]

and

\[
k_2 = \frac{CD(a, c, p) CD(a, b, d)}{CD(a, c, d) CD(a, b, p)} \tag{5.14}
\]

We note that the expression for \( k_2 \) is independent of \( \overline{b}_R \) and \( \overline{d}_R \), therefore, independent of the chosen random line. This was expected as it is a direct implication of the cross-ratio invariance. Let us now assume that each of the reference pairs of coordinates can be estimated with error normally distributed with zero mean and covariance matrix

\[
\Sigma = \begin{pmatrix} \sigma_{xx}^2 & \sigma_{xy} \cr \sigma_{yx} & \sigma_{yy}^2 \end{pmatrix}. \tag{5.15}
\]

Then, it can be shown (for proof see appendix A) that the variance of the error distribution in the value of \( k_2 \) is given by:

\[
\sigma_{k_2 k_2}^2 = \left[ \left( \frac{\partial k_2}{\partial a_{ix}} \right)^2 + \left( \frac{\partial k_2}{\partial a_{iy}} \right)^2 + \cdots \right] \sigma_{xx}^2 + \left[ \left( \frac{\partial k_2}{\partial a_{ix}} \right)^2 + \left( \frac{\partial k_2}{\partial a_{iy}} \right)^2 + \cdots \right] \sigma_{yy}^2 + 2 \left[ \frac{\partial k_2}{\partial a_{ix}} \frac{\partial k_2}{\partial a_{iy}} + \frac{\partial k_2}{\partial b_{ix}} \frac{\partial k_2}{\partial b_{iy}} + \cdots \right] \sigma_{xy} \tag{5.16}
\]

Applying this formula using \( k_2 \) given by (5.14) we can derive an expression for the error in \( k_2 \) which depends on the location of point \( p \) on the image plane. For simplicity, we shall assume that the errors in determining the positions of points on the image plane in the \( x \) and \( y \) directions are uncorrelated, thus \( \sigma_{xy} = 0 \). First, the coefficient which multiplies
5.1. ERROR ANALYSIS

\( \sigma^2_{z_y} \) will be derived. In order to determine its first term, \( \frac{\partial k_2}{\partial a_x} \), we can use the rules of differentiation and also the following special property of the \( CD \) function:

\[
\frac{\partial}{\partial a_{x}}(CD(a_1, a_2, \ldots, a_n)) = a_{jy} - a_{xy} \tag{5.17}
\]

\[
\frac{\partial}{\partial a_{xy}}(CD(a_1, a_2, \ldots, a_n)) = a_{xy} - a_{jx} \tag{5.18}
\]

where \( j = ((i + 1) \mod n) \)

\( k = ((i - 1 + n) \mod n) \)

With the help of the above two equations and equation (5.14) we get:

\[
\frac{\partial k_2}{\partial a_x} = -\frac{k_{2a}}{k_{2c}k_{2d}}(b_y - d_y) + \frac{k_{2b}}{k_{2c}k_{2d}}(c_y - p_y) - \frac{k_{2c}k_{2b}}{k_{2c}k_{2d}}(c_y - d_y) - \frac{k_{2c}k_{2b}}{k_{2c}k_{2d}}(b_y - p_y)
\]

Similarly,

\[
\frac{\partial k_2}{\partial b_x} = -\frac{k_{2a}}{k_{2c}k_{2d}}(d_y - a_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(p_y - a_y)
\]

\[
\frac{\partial k_2}{\partial c_x} = \frac{k_{2b}}{k_{2c}k_{2d}}(p_y - a_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(d_y - a_y)
\]

\[
\frac{\partial k_2}{\partial d_x} = -\frac{k_{2a}}{k_{2c}k_{2d}}(a_y - b_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(a_y - c_y)
\]

\[
\frac{\partial k_2}{\partial p_x} = -\frac{k_{2a}}{k_{2c}k_{2d}}(a_y - c_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(a_y - b_y)
\]

Therefore,

\[
\left( \frac{\partial k_2}{\partial a_x} \right)^2 + \left( \frac{\partial k_2}{\partial b_x} \right)^2 + \left( \frac{\partial k_2}{\partial c_x} \right)^2 + \left( \frac{\partial k_2}{\partial d_x} \right)^2 + \left( \frac{\partial k_2}{\partial p_x} \right)^2 =
\]

\[
= \left( \frac{k_{2a}}{k_{2c}k_{2d}}(b_y - d_y) + \frac{k_{2b}}{k_{2c}k_{2d}}(c_y - p_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(c_y - d_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(b_y - p_y) \right)^2 +
\]

\[
+ \left( \frac{k_{2a}}{k_{2c}k_{2d}}(d_y - a_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(p_y - a_y) \right)^2 + \left( \frac{k_{2b}}{k_{2c}k_{2d}}(p_y - a_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(d_y - a_y) \right)^2 +
\]

\[
+ \left( \frac{k_{2a}}{k_{2c}k_{2d}}(a_y - b_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(a_y - c_y) \right)^2 + \left( \frac{k_{2b}}{k_{2c}k_{2d}}(a_y - c_y) - \frac{k_{2a}k_{2b}}{k_{2c}k_{2d}}(a_y - b_y) \right)^2
\]

A similar expression for the coefficient which multiplies \( \sigma^2_{z_y} \) can be derived by following the same procedure as above.

\[
\left( \frac{\partial k_2}{\partial a_y} \right)^2 + \left( \frac{\partial k_2}{\partial b_y} \right)^2 + \left( \frac{\partial k_2}{\partial c_y} \right)^2 + \left( \frac{\partial k_2}{\partial d_y} \right)^2 + \left( \frac{\partial k_2}{\partial p_y} \right)^2 =
\]
5.1. ERROR ANALYSIS

The coefficients which multiply \( a^y \) are the error amplification factors. As long as these factors are less or equal to 1 the error is damped but when these factors exceed 1, the error is amplified. We can derive similar expressions for the other two projective coordinates of \( p \).

In figures 5.2a, 5.2b and 5.2c we fixed the positions of the reference points and allowed the position of \( p \) to scan the whole plane. We mark with black the regions where amplification of the error is expected due to \( k_i, k^2 \) and \( k^3 \) respectively. Figure 5.2d is simply an overlap of the previous three figures showing in black the regions where at least one projective coordinate is very sensitive to noise. Finally, the most meaningful result is shown in 5.2e where the median of the first three figures is plotted, indicating the regions where 2 projective coordinates are unstable. Notice that apart from three small regions around points \( a, b \) and \( c \) where two projective coordinates are with amplified error, in all other places \( p \) has at least two projective coordinates which can be calculated reliably, and this is enough for the determination of the position of point \( P \) in the 3D space.

5.1.1.1 How many cross-ratios?

In the previous section we investigated the three cross-ratios, namely \( k_1, k_2 \) and \( k_3 \) as defined by equation (4.8). However, by permuting the reference points \( a, b, c \) and \( d \) (see figure 5.3) one can obtain \( 4! = 24 \) distinct values cross-ratios as shown in table 5.1. It should be stressed, however, that there are only 2 independent cross-ratios [8] as illustrated in table 5.1 where all cross-ratios can be written in terms of \( k_1 \) and \( k_2 \). For each configuration the \( k_i \) cross-ratio was obtained as well as the associated amplification factors for the errors \( \sigma_{ax}^2 \) and \( \sigma_{ay}^2 \), \( e_{bx} \) and \( e_{by} \) respectively. It was noticed that although all 24 cross-ratios have distinct values, they could be divided into pairs with identical noise behaviour. That is because only the choice of the first line of the pencil of the four makes a difference as far as the expected error is concerned. That can also be seen in figures 5.2a-c where the most unstable regions were across the first line of the pencil of lines used in the determination of \( k_1, k_3 \).
5.1. ERROR ANALYSIS

(a) Error in $k_1 (\varepsilon_{k_1})$
(b) Error in $k_2 (\varepsilon_{k_2})$
(c) Error in $k_3 (\varepsilon_{k_3})$
(d) Max of $\varepsilon_{k_1}, \varepsilon_{k_2}, \varepsilon_{k_3}$
(e) Median of $\varepsilon_{k_1}, \varepsilon_{k_2}, \varepsilon_{k_3}$

Figure 5.2: Regions of instability for determination of $k$. 


Another interesting thing is the angle $\theta$ formed between the line which goes through the origin of the pencil of lines and point $p$, and the first line of the pencil of lines. For example, in the determination of $k_{24}$ defined as $[da, db, dc, dp]$ this angle would be $\theta = \theta_{da}$. The closer this line is to 90° the more likely the associated cross-ratio is to be robust. Indeed, for $k_{24}$ $\theta = 82.4^\circ$ is the closest to 90° and the amplification factor in $x$ and $y$ was the smallest (0.000008, 0.000018). As a rule of thumb, one could choose the two cross-ratios which minimise $|\theta - 90^\circ|$ instead of calculating all the errors associated with the 24 cross-ratios and choose the ones with the smallest expected errors.

5.1.2 From the projective coordinates to the 3D coordinates of $P_1$

Let us say that of the three projective coordinates of $p$ computed in the previous stage, $k_1$ and $k_2$ are the most reliable. Point $P_1$ on the plane defined by points $A$, $B$, $C$ and $D$ has the same projective coordinates and the problem now is to find its 3D Cartesian coordinates from the knowledge of the 3D Cartesian coordinates of $A$, $B$, $C$ and $D$ and $k_1$ and $k_2$.

5.1.2.1 Transforming to the local coordinate system

Since points $A$, $B$, $C$, $D$ and $P_1$ belong to the same plane, a translation $T$ and a rotation $R$ transformation can be found from the world coordinate system to a local coordinate system defined in such way that its origin is at $A$, its $y$ axis is along the line $AB$ and the $x$ axis is normal to the reference plane (figure 5.4). The five points in the new local
5.1. ERROR ANALYSIS

<table>
<thead>
<tr>
<th>$i$</th>
<th>Cross-Ratio</th>
<th>Angle ($\theta$)</th>
<th>$k_i$</th>
<th>$e_{kx}$</th>
<th>$e_{ky}$</th>
<th>Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$ab, ac, ad, ap$</td>
<td>37.3°</td>
<td>-0.812147</td>
<td>0.000280</td>
<td>0.000303</td>
<td>$t_1$</td>
</tr>
<tr>
<td>2</td>
<td>$ab, ad, ac, ap$</td>
<td>37.3°</td>
<td>1.812147</td>
<td>0.000280</td>
<td>0.000303</td>
<td>1 - $t_1$</td>
</tr>
<tr>
<td>3</td>
<td>$ac, ab, ad, ap$</td>
<td>9.6°</td>
<td>-1.231304</td>
<td>0.000643</td>
<td>0.000697</td>
<td>$1/t_1$</td>
</tr>
<tr>
<td>4</td>
<td>$ac, ad, ab, ap$</td>
<td>9.6°</td>
<td>2.231304</td>
<td>0.000643</td>
<td>0.000697</td>
<td>$((t_1 - 1)/t_1)$</td>
</tr>
<tr>
<td>5</td>
<td>$ad, ac, ab, ap$</td>
<td>27.4°</td>
<td>0.448168</td>
<td>0.000026</td>
<td>0.000028</td>
<td>$t_2/(t_1 - 1)$</td>
</tr>
<tr>
<td>6</td>
<td>$ab, ad, ac, ap$</td>
<td>27.4°</td>
<td>0.551832</td>
<td>0.000026</td>
<td>0.000028</td>
<td>1/(1 - $t_1$)</td>
</tr>
<tr>
<td>7</td>
<td>$ba, bc, bd, bp$</td>
<td>98.4°</td>
<td>0.242547</td>
<td>0.000003</td>
<td>0.000038</td>
<td>$t_2$</td>
</tr>
<tr>
<td>8</td>
<td>$ba, bd, bc, bp$</td>
<td>98.4°</td>
<td>0.757453</td>
<td>0.000003</td>
<td>0.000038</td>
<td>1 - $t_2$</td>
</tr>
<tr>
<td>9</td>
<td>$bc, ba, bd, bp$</td>
<td>8.7°</td>
<td>4.122917</td>
<td>0.000996</td>
<td>0.010964</td>
<td>$1/t_2$</td>
</tr>
<tr>
<td>10</td>
<td>$bc, bd, ba, bp$</td>
<td>8.7°</td>
<td>-3.122917</td>
<td>0.000996</td>
<td>0.010964</td>
<td>$((t_2 - 1)/t_2)$</td>
</tr>
<tr>
<td>11</td>
<td>$bd, ba, bc, bp$</td>
<td>27.7°</td>
<td>-0.320213</td>
<td>0.000010</td>
<td>0.000115</td>
<td>$t_2/(t_2 - 1)$</td>
</tr>
<tr>
<td>12</td>
<td>$bd, bc, ba, bp$</td>
<td>27.7°</td>
<td>1.320213</td>
<td>0.000010</td>
<td>0.000115</td>
<td>1/(1 - $t_2$)</td>
</tr>
<tr>
<td>13</td>
<td>$cb, ca, cd, cp$</td>
<td>9.2°</td>
<td>-3.348414</td>
<td>0.008946</td>
<td>0.016649</td>
<td>$t_1/t_2$</td>
</tr>
<tr>
<td>14</td>
<td>$cb, cd, ca, cp$</td>
<td>9.2°</td>
<td>4.348414</td>
<td>0.008946</td>
<td>0.016649</td>
<td>$((t_2 - t_1)/t_2)$</td>
</tr>
<tr>
<td>15</td>
<td>$ca, cb, cd, cp$</td>
<td>16.8°</td>
<td>-0.298649</td>
<td>0.000071</td>
<td>0.000132</td>
<td>$t_2/t_1$</td>
</tr>
<tr>
<td>16</td>
<td>$ca, cd, cb, cp$</td>
<td>16.8°</td>
<td>1.298649</td>
<td>0.000071</td>
<td>0.000132</td>
<td>$((t_1 - t_2)/t_1)$</td>
</tr>
<tr>
<td>17</td>
<td>$cd, ca, cb, cp$</td>
<td>41.4°</td>
<td>0.770031</td>
<td>0.000025</td>
<td>0.000047</td>
<td>$t_1/(t_1 - t_2)$</td>
</tr>
<tr>
<td>18</td>
<td>$cd, cd, ca, cp$</td>
<td>41.4°</td>
<td>0.229969</td>
<td>0.000025</td>
<td>0.000047</td>
<td>$t_2/(t_2 - t_1)$</td>
</tr>
<tr>
<td>19</td>
<td>$db, dc, da, dp$</td>
<td>37.8°</td>
<td>-1.392421</td>
<td>0.000263</td>
<td>0.000595</td>
<td>$(t_0 - t_1)/(t_0 - 1)$</td>
</tr>
<tr>
<td>20</td>
<td>$db, da, dc, dp$</td>
<td>37.8°</td>
<td>2.392421</td>
<td>0.000263</td>
<td>0.000595</td>
<td>$((t_1 - 1)/(t_0 - 1))$</td>
</tr>
<tr>
<td>21</td>
<td>$dc, db, da, dp$</td>
<td>55.2°</td>
<td>-0.718174</td>
<td>0.000070</td>
<td>0.000158</td>
<td>$(t_0 - 1)/(t_0 - t_1)$</td>
</tr>
<tr>
<td>22</td>
<td>$dc, da, db, dp$</td>
<td>55.2°</td>
<td>1.718174</td>
<td>0.000070</td>
<td>0.000158</td>
<td>$((t_1 - 1)/(t_0 - t_1))$</td>
</tr>
<tr>
<td>23</td>
<td>$da, dc, db, dp$</td>
<td>82.4°</td>
<td>0.582013</td>
<td>0.000008</td>
<td>0.000018</td>
<td>$(t_1 - t_2)/(t_1 - 1)$</td>
</tr>
<tr>
<td>24</td>
<td>$da, db, dc, dp$</td>
<td>82.4°</td>
<td>0.417987</td>
<td>0.000008</td>
<td>0.000018</td>
<td>$(t_2 - 1)/(t_1 - 1)$</td>
</tr>
</tbody>
</table>

Table 5.1: There are $4! = 24$ possible distinct values of cross-ratios of which are independent (note that $k_1 = t_1$ and $k_7 = t_2$).

Coordinate system have position vectors given by:

\[
\begin{align*}
\vec{A}' &= R(\vec{A} - \vec{T}) \\
\vec{B}' &= R(\vec{B} - \vec{T}) \\
\vec{C}' &= R(\vec{C} - \vec{T}) \\
\vec{D}' &= R(\vec{D} - \vec{T}) \\
\vec{P}' &= R(\vec{P} - \vec{T})
\end{align*}
\]

This can make the mathematical analysis much easier. When the coordinates of point $P'_1$ have been determined, the position of $P_1$ can be found by transforming back to the world
5.1. ERROR ANALYSIS

The rotation matrix $R$ and the translation matrix $T$ can be calculated as follows: Imagine the coordinate system $O'x'y'z'$ which has origin $O' = A$, axis $O'z'$ the normal to the reference plane $ABCD$, axis $O'y'$ the vector defined by the difference $B - A$ and obviously axis $O'x'$ the cross product of $O'y'$ by $O'z'$. Then, the task of finding the appropriate $R$ and $T$ for making all $z$s zero is reduced to transforming from $O'x'y'z'$ to our world coordinate systems $Oxyz$. Therefore, $T$ is simply $A$ and $R$ is given by:

$$R = \begin{pmatrix} O'x'_x & O'y'_x & O'z'_x \\ O'x'_y & O'y'_y & O'z'_y \\ O'x'_z & O'y'_z & O'z'_z \end{pmatrix}^{-1}$$

where $O'x'_x$ is the $x$ component of vector $O'x'$ and similar interpretation holds for all other entries in this matrix. Vectors $O'z'$, $O'y'$ and $O'x'$ are unit length vectors defined by:

$$O'z' = (\widehat{O'z} - \widehat{A}) \times (\widehat{B} - \widehat{A})$$
$$O'y' = (\widehat{B} - \widehat{A})$$
$$O'x' = O'y' \times O'z'$$

and the wide hat over the vectors means that they are normalised.
5.1. ERROR ANALYSIS

5.1.2.2 Calculating $P_1$

Let us now assume that the four points $A$, $B$, $C$ and $D$ are the four coplanar points with zero $z'$ and that the $k_1$ and $k_2$ projective coordinates of $P_1$ are known. We shall show how the $x'$ and $y'$ coordinates of $P_1$ can be calculated.

First, since $A$ is known we can find the equation of the line which goes through $A$ and $P_1$ (figure 5.5). Let $\vec{p}_R = b_R + \mu_R \vec{d}_R$ be the equation of any line on the same plane not going through $A$. The values of $\mu_J, \mu_K$ and $\mu_L$ can be found using the corresponding equations to (5.1) applied on the $x'O'y'$ plane. Then we can solve for $\mu_M$:

$$\mu_M = \frac{k_2 \mu_J (\mu_L - \mu_J) + \mu_K (\mu_J - \mu_L)}{k_2 (\mu_L - \mu_k) + (\mu_J - \mu_L)}$$

Substituting $\mu_J, \mu_K$ and $\mu_L$ in (5.20) we obtain an expression for $\mu_M$. The Cartesian coordinates of $M$ can then be found:

$$M_{x'} = b_{x'} + \mu_M d_{x'}$$

(5.21)

$$M_{y'} = b_{y'} + \mu_M d_{y'}$$

(5.22)

The next step is to find the equation of line $AP_1$. This is straightforward since the Cartesian coordinates of two points, $A$ and $M$, of the $AP_1$ line are known. Therefore, the base and direction vectors of the line $AP_1$ are:

$$b_{AP_1} = A_{x'}$$

(5.23)
5.1. ERROR ANALYSIS

\[ b_\text{AP}_{\text{i},\text{j}}' = A_y' \]  
\[ d_\text{AP}_{\text{i},\text{j}}' = M_y' - A_y' \]  
\[ d_\text{AP}_{\text{i},\text{j}}' = M_y' - A_y' \]

However, the expressions we obtain for \( \vec{d}_{\text{AP}}' \) this way are still dependent on \( b_R \) and \( d_R \). To get rid of these two parameters and thus have a result independent of the choice of the random intersecting line we need to normalise the direction vector by dividing both its \( x \) and \( y \) components by the \( x \) component. This yields:

\[ d_\text{AP}_{\text{i},\text{j}}' = \frac{A_y' C_x D(B, A, D) + C_y' C_x D(B, D, A) + k_2(B_y' C_x D(C, A, D) + A_y' C_x D(A, C, D))}{A_y' C_x D(B, A, D) + C_x C_x D(B, D, A) + k_2(B_x' C_x D(C, A, D) + A_x' C_x D(A, C, D))} \]

We notice now that the parameters of the intersecting line do not enter in the formula. Since \( k_1 \) is also known the equation of the line \( CP_1 \) can be calculated in a similar way.

\[ d_\text{CP}_{\text{i},\text{j}}' = \frac{1}{C_y' C_x D(A, C, D) + B_y' C_x D(A, D, C) + k_1(A_y' C_x D(B, C, D) + C_y' C_x D(C, B, D))} \]

Then the coordinates of \( \vec{P}_1' \) can be found by finding the intersection of the two lines \( AP_1 \) and \( CP_1 \).

\[ P_{\text{i},\text{j}}' \text{ } d = k_1 k_2 A_x' (B_y' C_y' - B_y' C_x' + C_x' D_y' - C_y' D_x' + D_x' B_y' - D_y' B_x') \]
\[ - k_2 B_y' (A_y' C_y' - A_y' C_x' + C_x' D_y' - C_y' D_x' + D_x' A_y' - D_y' A_x') \]
\[ + C_x' (A_x' B_y' - A_y' B_x' + B_x' D_y' - B_y' D_x' + D_x' A_y' - D_y' A_x') \]  
\[ P_{\text{i},\text{j}}' \text{ } d = k_1 k_2 A_x' (B_y' C_y' - B_y' C_x' + C_x' D_y' - C_y' D_x' + D_x' B_y' - D_y' B_x') \]
\[ - k_2 B_y' (A_y' C_y' - A_y' C_x' + C_x' D_y' - C_y' D_x' + D_x' A_y' - D_y' A_x') \]
\[ + C_x' (A_x' B_y' - A_y' B_x' + B_x' D_y' - B_y' D_x' + D_x' A_y' - D_y' A_x') \]

with \( d = k_1 k_2 (B_x' C_y' - B_y' C_x' + C_x' D_y' - C_y' D_x' + D_x' B_y' - D_y' B_x') \)
\[ - k_2 (A_y' C_y' - A_y' C_x' + C_x' D_y' - C_y' D_x' + D_x' A_y' - D_y' A_x') \]
\[ + (A_x' B_y' - A_y' B_x' + B_x' D_y' - B_y' D_x' + D_x' A_y' - D_y' A_x') \]

where all coordinates that appear in these formulae refer to the local coordinate system defined on plane \((A, B, C, D)\). Rewriting (5.27), (5.28) and (5.29) as a combination of appropriate third order \( C D \) functions yields:

\[ P_{\text{i},\text{j}}' \text{ } d = A_x' k_1 k_2 \text{ } C D(B, C, D) - B_y' k_2 \text{ } C D(A, C, D) + C_x' \text{ } C D(A, B, D) \]
5.1. ERROR ANALYSIS

\begin{align*}
P_{1y'} \delta &= A_{y'} k_1 k_2 CD(B, C, D) - B_{y'} k_2 CD(A, C, D) + C_{y'} CD(A, B, D) \tag{5.31} \\
\delta &= k_1 k_2 CD(B, C, D) - k_2 CD(A, C, D) + CD(A, B, D) \tag{5.32}
\end{align*}

To investigate the effect of the error in the measured positions of the reference points on the determination of the position of \( P \), we can proceed in a way similar to the one described in the previous section. After the straightforward, but tedious, application of simple Geometric and Algebraic reasoning, we can derive formulae similar to formula (5.16) for the error in the calculation of \( P_{1x'} \) and \( P_{1y'} \) introduced by the error in the actual positions of the reference points \( A, B, C \) and \( D \), assuming that the values of \( k_1, k_2 \) and \( k_8 \) are known accurately. Although these expressions can be further simplified by setting \( A_{x'} = A_{y'} = B_{x'} = 0 \), because of the choice of the local coordinate system, they are still too big to be published here.

Then, we repeat the process we followed for the construction of figure 4: As point \( p \) scans the image plane we compute at each position the values of \( k_1, k_2 \) and \( k_3 \) for the given set of reference points. Since we are interested in the error introduced by the inaccuracy in the reference points, we ignore the fact that \( k_1, k_2 \) and \( k_3 \) are themselves computed with some error and we put their values into the formulae we derived for the amplification factors and calculate them assuming that the error in all coordinate positions of the reference points is the same.

Figure 5.6a shows the various regions in the image plane where the coefficient of \( \sigma_{xx}^2 \) for \( P_{1x'} \) is within a certain range. White are the regions where the amplification factor is less than 1, so they are the stable regions. Each shade corresponds to the amplification factor incremented by 1 as we move away from the white region, with the very dark regions corresponding to error amplification factor more than 10. Figure 5.6b illustrates the coefficient of \( \sigma_{yy}^2 \) for \( P_{1x'} \), whereas in figure 5.6c the maximum of the two coefficients is plotted to give us an indication of the expected error in the determination of the \( x' \) coordinate of \( P \). Similar analysis is illustrated in figure 5.7 for the determination of the \( y' \) coordinate of \( P \), and finally, in figure 5.8 the regions where both \( x' \) and \( y' \) coordinates of \( P \) are expected to have maximum robustness to noise are shown in white. Similar analysis can be performed to find the instability regions for \( P_2 \) but the only difference with the above analysis is the reference points used.
5.1. ERROR ANALYSIS

Figure 5.6: Regions of instability for determination of $P_{1x'}$.

(a) Coefficient of $\sigma_{xx}^2$  (b) Coefficient of $\sigma_{yy}^2$  (c) Worst case

Figure 5.7: Regions of instability for determination of $P_{1y'}$.

(a) Coefficient of $\sigma_{xx}^2$  (b) Coefficient of $\sigma_{yy}^2$  (c) Worst case

Figure 5.8: Worst case for reconstruction of $P_{1}$. 
5.1. ERROR ANALYSIS

5.1.2.3 Back to the world coordinate system

After calculating the Cartesian coordinates as well as the error associated with \( P_{ix'} \) it is essential to transform them to the world coordinate system, \( Oxyz \) (figure 5.9). This is a straightforward process, since \( P_{ix}, P_{iy} \) and \( P_{iz} \) can be obtained by

\[
\begin{align*}
P_{ix} &= r_{11} P_{ix'} + r_{12} P_{iy'} + T_x \\
P_{iy} &= r_{21} P_{ix'} + r_{22} P_{iy'} + T_y \\
P_{iz} &= r_{31} P_{ix'} + r_{32} P_{iy'} + T_z
\end{align*}
\]

where \( r_{ij} \) are the components of \( R^{-1} \). We note that \( P_{ix} \) is a function of one pair of random variables, namely the \((P_{ix'}, P_{iy'})\) pair. Therefore, the covariances of the error distribution in the values of \( P_{i} \) can be estimated by:

\[
\begin{align*}
\sigma_{P_{ix}P_{ix}} &= r_{11}^2 \sigma_{P_{ix'}P_{ix'}} + r_{12}^2 \sigma_{P_{iy'}P_{iy'}} + r_{11}r_{12} \sigma_{P_{ix'}P_{iy'}} \\
\sigma_{P_{ix}P_{iy}} &= r_{21}^2 \sigma_{P_{ix'}P_{ix'}} + r_{22}^2 \sigma_{P_{iy'}P_{iy'}} + r_{21}r_{22} \sigma_{P_{ix'}P_{iy'}} \\
\sigma_{P_{ix}P_{iz}} &= r_{31}^2 \sigma_{P_{ix'}P_{ix'}} + r_{32}^2 \sigma_{P_{iy'}P_{iy'}} + r_{31}r_{32} \sigma_{P_{ix'}P_{iy'}} \\
\sigma_{P_{iy}P_{ix}} &= r_{11}r_{21} \sigma_{P_{ix'}P_{ix'}} + r_{12}r_{21} \sigma_{P_{iy'}P_{iy'}} + r_{11}r_{21} \sigma_{P_{ix'}P_{iy'}} \\
\sigma_{P_{iy}P_{iy}} &= r_{21}r_{22} \sigma_{P_{ix'}P_{ix'}} + r_{22}r_{22} \sigma_{P_{iy'}P_{iy'}} + r_{21}r_{22} \sigma_{P_{ix'}P_{iy'}} \\
\sigma_{P_{iy}P_{iz}} &= r_{31}r_{21} \sigma_{P_{ix'}P_{ix'}} + r_{32}r_{21} \sigma_{P_{iy'}P_{iy'}} + r_{31}r_{21} \sigma_{P_{ix'}P_{iy'}}
\end{align*}
\]

Since \( P_{ix}, P_{iy} \) and \( P_{iz} \) are linearly dependent on \( P_{ix'}, P_{iy'} \) which to the approximation we are using here are jointly normal, the resultant distribution is normal. Hence their joint density is to the linear limit assumed here

\[
\mathcal{N}(P_{ix}, P_{iy}, P_{iz}; \sigma_{P_{ix}P_{ix}}, \sigma_{P_{iy}P_{iy}}, \sigma_{P_{iz}P_{iz}}, \sigma_{P_{ix}P_{iy}}, \sigma_{P_{ix}P_{iz}}, \sigma_{P_{iy}P_{iz}})
\]

5.1.3 Determining the 3-D coordinates of \( P \)

Having computed the 3D coordinates of \( P_1 \), the intersection of the left viewing line, \( l \), with the first reference plane, we can repeat the process and calculate the 3D coordinates of \( P_2 \), the intersection of the viewing line with the second reference plane. Similarly, the coordinates of \( P'_1 \) and \( P'_2 \), the intersection of the right viewing line, \( l' \), with the first and second reference plane respectively, can be found. Therefore, the equations of both the left and right viewing lines can be found:

\[
\begin{align*}
\vec{L} &= \vec{P}_1 + \mu_L(\vec{P}_3 - \vec{P}_1) \\
\vec{L'} &= \vec{P}'_1 + \mu_L'(\vec{P}'_3 - \vec{P}'_1)
\end{align*}
\]
where \( \mathcal{L} \) is any point on the left viewing line \( l \) and \( \mathcal{L}' \) is any point on the right viewing line.

In theory, the two viewing lines \( l \) and \( l' \) must be coplanar because they intersect at point \( P \). However, in general this is not true, mainly due to various errors introduced in the first two steps of the projective reconstruction process. A simple way of checking coplanarity is to calculate the volume, \( V \), of the parallelepiped defined by the vectors \( P_1P_2 \), \( P_1P'_2 \) and \( P_2P'_1 \) which is given by

\[
V = \begin{vmatrix}
P_{2x} - P_{1x} & P_{2y} - P_{1y} & P_{2z} - P_{1z} \\
P'_{2x} - P'_{1x} & P'_{2y} - P'_{1y} & P'_{2z} - P'_{1z} \\
P'_{1x} - P'_{1x} & P'_{1y} - P'_{1y} & P'_{1z} - P'_{1z}
\end{vmatrix}
\] (5.36)

If the volume is zero the viewing lines are coplanar and the coordinates of \( P \) are given by solving the simultaneous equations

\[
\bar{P}_1 + \mu_L (\bar{P}_2 - \bar{P}_1) = \bar{P}'_1 + \mu_{L'} (\bar{P}'_2 - \bar{P}'_1)
\]

Note, that there are three equations in two unknowns, \( \mu_L \) and \( \mu_{L'} \). This means that for the equations to be meaningful there must be at least one pair of the equations which are independent, and the remaining equation must be a combination of these two.

We, now, deal with the general case where \( l \) and \( l' \) are asymptotic. In this case the coordinates of \( P \) can be estimated by finding the midpoint of the minimum-length line segment, \( LL' \), which has one of its endpoint on \( l \) and the other on \( l' \) (see figure 5.10). The direction vector of the line segment \( LL' \), \( \vec{d} \), is the cross product of the direction vectors of
5.1. ERROR ANALYSIS

Figure 5.10: Determination of the 3D coordinates of point $P$.

Let $l$ and $l'$. Therefore,

$$\vec{d} = (\vec{P}_2 - \vec{P}_1) \times (\vec{P}'_2 - \vec{P}'_1)$$ (5.37)

The vector difference $(\vec{L} - \vec{L}')$ should result in $\vec{d}$ scaled by a constant $\alpha$. Therefore, using (5.34) and (5.35) we obtain

$$\left[ \vec{P}_1 + \mu_L(\vec{P}_2 - \vec{P}_1) \right] - \left[ \vec{P}'_1 + \mu_{L'}(\vec{P}'_2 - \vec{P}'_1) \right] = \alpha \vec{d}_x$$

The constant factor $\alpha$ can easily be eliminated from the above set of equations by dividing the first two by the third one by parts. The resulting set of equations is linear in $\mu_L$ and $\mu_{L'}$ and it can be solved for them in terms of the coordinates $P_1, P_2, P'_1, P'_2$. The case where $l$ and $l'$ are parallel needs special care. However, this case never occurs in stereo vision applications, since it directly implies parallel viewing lines. Having obtained the values for $\mu_L$ and $\mu_{L'}$, the coordinates of $P$ can be found by:

$$\vec{P} = \frac{1}{2} \left[ \vec{P}_1 + \mu_L(\vec{P}_2 - \vec{P}_1) + \vec{P}'_1 + \mu_{L'}(\vec{P}'_2 - \vec{P}'_1) \right]$$ (5.38)

where $\mu_L$ and $\mu_{L'}$ can be shown to be:

$$\mu_L, d = d_{x_1}d_{y_2}b_{x_2} + d_{y_2}d_{y_1}b_{x_2} + d_{x_1}d_{y_2}b_{y_2} - d_{x_1}d_{y_2}b_{x_2} - b_{x_1}d_{y_2}d_{y_2} -$$

$$- b_{y_1}b_{x_2}d_{y_2}b_{y_2} + b_{x_1}b_{x_2}d_{y_2}d_{y_2} - b_{y_1}b_{y_2}d_{x_2}d_{x_2} + d_{y_1}d_{y_1}b_{x_2} - d_{y_1}d_{y_1}b_{y_2}$$ (5.39)

$$\mu_{L'}, d = d_{x_1}d_{x_1}b_{x_2} + d_{x_1}d_{y_1}b_{y_2} - d_{x_1}d_{y_1}b_{y_2} + d_{y_1}d_{y_1}b_{x_2} - d_{y_1}d_{y_1}b_{y_2} +$$

$$+ d_{y_1}d_{x_1}b_{x_2} - d_{y_1}d_{x_1}b_{y_2} + b_{x_1}d_{y_1}d_{y_2} + d_{y_1}d_{y_1}d_{x_2} + b_{y_1}d_{y_1}d_{y_2}$$ (5.40)

$$d = d_{x_2}d_{y_1}d_{y_2} - d_{x_2}d_{y_1}d_{y_2}$$

with $d_{x_2}$ indicating the $x$ component of the direction vector of $l$ as given by (5.34), i.e.: $d_{x_2} = P_{x_2} - P_{1x}$, and similar interpretation holding for $d_{y_1}, d_{x_2}, d_{y_2}, d_{y_2}$ and $d_{x_2}$.
5.2. IMPLICATIONS OF THE ERROR ANALYSIS

Let \( P_i \) be a random variable normally distributed, i.e.:

\[
N(P_{1x}, P_{1y}, P_{2x} ; \sigma_{P_{1x}P_{1y}}^2, \sigma_{P_{1x}P_{2x}}^2, \sigma_{P_{1y}P_{2x}}^2, \sigma_{P_{1x}P_{2y}}^2, \sigma_{P_{1y}P_{2y}}^2, \sigma_{P_{1x}P_{1y}P_{2x}}^2, \sigma_{P_{1x}P_{1y}P_{2y}}^2, \sigma_{P_{1x}P_{2x}P_{2y}}^2)
\]  (5.42)

and similarly for points \( P_2, P'_1 \) and \( P'_2 \). Then, it is possible to find the variance of the distribution of \( P \) which being a linear combination of normally distributed random variables it will be normal, with parameters \( \sigma_{P_xP_x}^2, \sigma_{P_yP_y}^2, \sigma_{P_xP_y}^2, \sigma_{P'_xP_x}^2, \sigma_{P'_xP_y}^2, \sigma_{P'_xP'_x}^2 \) and \( \sigma_{P'_xP'_y}^2 \). From equation (5.39) it can be seen that

\[
\sigma_{P_xP_x}^2 = \frac{1}{4} \left[ (1 + \mu_x^2)\sigma_{P_xP_x}^2 + \mu_y^2\sigma_{P_yP_x}^2 + (1 + \mu_y^2)\sigma_{P_yP_y}^2 + \mu_x^2\sigma_{P_yP_y}^2 + \mu_y^2\sigma_{P_xP_y}^2 \right]
\]  (5.43)

As parameters \( \mu_x \) and \( \mu_y \) vary between 0 and 1 only for points that lie between points \( P_1 \) and \( P_2 \) and \( P'_1 \) and \( P'_2 \) respectively, it is obvious from the above expression that unless \( P \) lies in the space between the two reference planes, \( \mu_x \) and \( \mu_y \) will be larger than 1 and we shall have error amplification factor significantly greater than 1.

5.2 Implications of the error analysis

In order to apply the method of Projective Geometry to 3D reconstruction we need to take the following cautionary steps to avoid the introduction of large errors:

(a). It is best if the two reference planes are apart from each other and intersect along a line well away from the area of viewing. As reference points have to be visible on the image, this requirement implies that a setting like the one shown in figure 5.11 is appropriate. However, such an arrangement of reference points is not possible in the case of the granite stone reconstruction. We propose instead to use two sets of reference planes; planes \( ABCD \) and \( AEFB \) for all those points that fall on the right half of the image, and planes \( AEFB \) and \( CFEF \) for all those points which are on the left half of the image (as shown in figure 5.10). Such a setting would reduce the source of error amplification described in section 5.1.3.

(b). Provided the accuracy of the measurements of positions on the image plane is known, the error with which each of the projective coordinates of point \( p \) is computed can be estimated and the most reliable coordinates of point \( p \) may be used each time. There are only three small patches on the image plane where there is only one reliable projective coordinate.

(c). Once the projective coordinates of a point have been found, and given the uncertainty in the measurement of the 3D position of the reference points, the error in the
Figure 5.11: Setup of reference planes and points for minimum sensitivity to noise.

3D position of point $P$ can be estimated. This stage, however, is the most difficult to handle as it seems that unless the points are projected within a region more or less surrounded by the reference points, we are bound to have amplification of the error. The only thing we can do is to try to monitor it carefully. One can envisage the situation where each side of the granite block is reconstructed with the help of several points. Points which are unreliable then can be dropped out of the process and only points with acceptable accuracy are kept for the reconstruction stage.

Alternatively, one may consider several sets of reference points and compute the position of each point under consideration using all of these sets and every time keep the most reliable set. For the problem of granite block reconstruction, however, this is not easy. Many calibration and reference points are an impractical luxury in a stone processing plant. It seems more practicable to attempt to reconstruct each surface of the stone using several points some of which will have to be discarded during the process.

References

REFERENCES


Chapter 6

Solving the Correspondence Problem

In order to obtain three-dimensional depth information using the techniques described in the previous chapter it is essential to find the corresponding pairs of points between the left and right images. This is the well known correspondence problem in stereo vision. In the previous chapters it was assumed that correspondence was somehow available and no attempts were made to determine it. However, this is a very difficult problem in computer vision especially in the case of well separated cameras and non-coplanar features.

There are basically two different approaches to the correspondence problem: feature based and region based. Region based techniques [9, 23] use correlation among intensity patterns in the local neighbourhood of a pixel in one image with intensity patterns in the corresponding neighbourhood of a pixel in the other image [11]. Feature based methods [14, 7] match higher level entities such as edges, line segments or corners. These methods tend to be more efficient in terms of both computational time and performance since higher level information is employed. Other existing stereo matching algorithms can be found in [3, 21, 2].

The problem of matching coplanar features in two perspective views has already been addressed in several papers (for example see [15, 4, 18, 16, 22, 10, 1, 24]). However, when the features are not coplanar and the two views are at a very large angle the problem becomes very difficult to solve. Many invariants which can be calculated if features are coplanar are not valid when dealing with the general case of non-coplanar features. For example, the ordering of the points in the convex hull of a set of corresponding points [15] will be different for the two images in the general case and co-linearity of features is not anymore guaranteed.
In this chapter we show how to cast the point matching problem into an optimisation framework and suggest efficient ways of obtaining the optimal solution. Experimental results are also included.

6.1 Casting the problem into an optimisation one

In this section it will be shown how a solution to the correspondence problem can be found by casting it into an optimisation problem and obtaining its optimal solution. First, some fundamental issues arising from the geometry of the imaging system will be considered.

6.1.1 The epipolar constraint

The most important constraint which can be employed in order to reduce the number of potential matches is a geometric constraint imposed by the imaging system itself, known as the epipolar constraint [6].

A point $m$ in the left image (see figure 6.1) may have been produced by all the points that lie on the infinite half-line which is defined by points $m$ and $M$, let us call it viewing line $mM$. A direct consequence of this observation is that all possible matches $m'$ of $m$ in the right image should lie on the projection of the viewing line $mM$ on the right image, which is called the epipolar line [6]. The epipolar line goes through the point $E'$, which is the intersection of the line defined by the two optical centres $O$ and $O'$ with the right
image. Point \( E' \) is called the epipole of the right camera with respect to the left and all epipolar lines go through this. The plane defined by the two optical centres \( O \) and \( O' \) and the point \( M \) is known as the epipolar plane.

The epipolar constraint is perhaps the most important geometric constraint. Indeed, if this constraint is not taken into account a point \( m \) from the left image could be put in correspondence with any point \( m' \) in the right image. Thanks to the epipolar constraint we can reduce a two-dimensional search into a one-dimensional one [6] since all correspondences of any point \( m \) are to be found on the corresponding epipolar line. As expected it is a symmetric constraint since all the possible matches of a point \( m' \) in the right image should lie on the corresponding epipolar line in the left image.

6.1.2 The Fundamental matrix

The epipolar geometry in the case where no camera calibration parameters are needed can be algebraically represented by the product:

\[
\begin{bmatrix} m_x' & m_y' & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} = \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix}
\]

where matrix \( f_1, \ldots, f_9 \) are the nine elements of the so called fundamental matrix [6] which describes the correspondence between points from left and right images. The fundamental matrix has 9 matrix elements but only their ratio is important, which leaves 8 degrees of freedom. In addition since the determinant of \( F \) is zero the fundamental matrix has 7 degrees of freedom. If camera calibration parameters are not known then the fundamental matrix is the key concept if one wants to proceed only from image measurements. Work has been done on computing the fundamental matrix from point correspondences [8, 20, 17]. A stability analysis of the fundamental matrix can be found in [12].

6.1.3 Casting Matching into Optimisation

A solution to the correspondence problem is possible if we consider the cross-ratio invariance which arises if we assume the pinhole camera model for both cameras. Consider for example the points which form an 'M'-shaped pattern in figure 6.1. Assume that the 3D coordinates of four coplanar points (forming the reference plane) are known as well as their projections to the left and right image. If the projections of the 'M'-shaped pattern
on the left and right images have been identified, our task is to match the points from the left image with the points on the right image.

First, the projective coordinates of all points in the projective coordinate system that is formed by the four reference points can be determined. Then, the points can be reconstructed in the right image to form what is shown in figure 6.1 as the virtual image. This is possible because of the cross-ratio invariance and because the pinhole camera model has been assumed. It can be further noted that all the lines that are formed by the corresponding pair of points of the virtual and right image go through the same point $E$. This point is the projection of the optical centre of the left camera to the right image and as mentioned earlier is called the epipole.

The technique we propose for solving the correspondence problem is based on the above observations and tries to match points in such a way that all epipolar lines go through the same point.

### 6.2 Deriving the cost function

Before proceeding any further let us deal with the simple problem of deriving a cost function which given a set of coplanar lines will be able to give us an indication of how close they intersect. First, let us examine the "dual" problem of fitting a line to a set of points in 2D.

#### 6.2.1 Fitting a line to a set of points

Given a set of $N$ points $\tilde{p}_i = (x_i, y_i)$ in $\mathbb{R}^2$ one can calculate the parameters $\alpha$ and $\beta$ of the line $y_i = \alpha x_i + \beta$ which best fit the $N$ points as follows:

- First find the base $\tilde{b}$ and direction vector $\tilde{d}$ of the required line:
  - The base vector, $\tilde{b}$, can be obtained by calculating the expected value of the $N$ points, $\tilde{p}_{\text{cx}}$ and $\tilde{p}_{\text{cy}}$, i.e.:
    
    \[
    \begin{align*}
    b_x &= \frac{1}{N} \sum_{i=1}^{N} x_i \tag{6.1a} \\
    b_y &= \frac{1}{N} \sum_{i=1}^{N} y_i \tag{6.1b}
    \end{align*}
    \]

  - Calculate the covariance matrix $\Sigma$ of the $N$ points:

\[
\Sigma = \begin{pmatrix}
\sigma_{xx}^2 & \sigma_{xy}^2 \\
\sigma_{yx}^2 & \sigma_{yy}^2
\end{pmatrix}
\tag{6.2}
\]
6.2. DERIVING THE COST FUNCTION

where

\[
\sigma_{xx}^2 = \frac{1}{N} \sum_{i=1}^{N} \left( x_i - \frac{1}{N} \sum_{i=1}^{N} x_i \right)^2
\]  
(6.3a)

\[
\sigma_{yy}^2 = \frac{1}{N} \sum_{i=1}^{N} \left( y_i - \frac{1}{N} \sum_{i=1}^{N} y_i \right)^2
\]  
(6.3b)

\[
\sigma_{xy} = \frac{1}{N} \sum_{i=1}^{N} \left[ \left( x_i - \frac{1}{N} \sum_{i=1}^{N} x_i \right) \left( y_i - \frac{1}{N} \sum_{i=1}^{N} y_i \right) \right]
\]  
(6.3c)

- Find the eigenvalues of the covariance matrix and sort them according to their size. It can be shown that the smallest and largest eigenvalues, \( \lambda_1 \) and \( \lambda_2 \) respectively, are given by

\[
\lambda_1 = \frac{1}{2} \left( \sigma_{xx}^2 + \sigma_{yy}^2 - \sqrt{\sigma_{xx}^4 - 2\sigma_{xx}^2\sigma_{yy}^2 + \sigma_{yy}^4 + 4\sigma_{xy}^2} \right)
\]  
(6.4a)

\[
\lambda_2 = \frac{1}{2} \left( \sigma_{xx}^2 + \sigma_{yy}^2 + \sqrt{\sigma_{xx}^4 - 2\sigma_{xx}^2\sigma_{yy}^2 + \sigma_{yy}^4 + 4\sigma_{xy}^2} \right)
\]  
(6.4b)

- The direction vector \( \vec{d} \) is given by the eigenvector corresponding to the largest eigenvalue which can be shown to be

\[
d_x = \frac{\lambda_2 - \sigma_{xy}^2}{\sigma_{xy}}, \quad d_y = 1
\]  
(6.5)

- From \( \vec{b} \) and \( \vec{d} \) obtain \( \alpha \) and \( \beta \) by

\[
\alpha = \frac{d_y}{d_x}, \quad \beta = \frac{d_y b_x - d_x b_y}{d_x}
\]  
(6.6a)

(6.6b)

- Finally, a measure of how "good" the line fits the points can be given by the product of the two eigenvalues \( \lambda_1 \) and \( \lambda_2 \) scaled by \( N^2 \), i.e.:

\[
N^2 \lambda_1 \lambda_2 = N^2 \left( \sigma_{xx}^2 \sigma_{yy}^2 - \sigma_{xy}^4 \right)
\]  
(6.7)

This product gives us an indication of the spread of the points around the line. Consequently, the smaller this product is, the better the fit is. It can be seen easily that the cost function can be written in terms of the \( x \) and \( y \) coordinates of the \( N \) points as follows:

\[
f(x_i, y_i; i = 1, \ldots, N) = \sum_{i=1}^{N} (x_i - \bar{x})^2 \sum_{i=1}^{N} (y_i - \bar{y})^2 - \left[ \sum_{i=1}^{N} (x_i - \bar{x}) (y_i - \bar{y}) \right]^2
\]
6.3 Exploring the configuration space

It is clear that if there exist \( N \) points on the left image, \( p_1, p_2, \ldots, p_N \) and \( M \) points on the right image, \( p'_1, p'_2, \ldots, p'_M \) and \( B \) is the number of matches required, the number of possible solutions to be examined is:

\[
P(M, B) C_B^N = \frac{M!}{(M - B)!} \frac{N!}{(N - B)! B!}
\]

where \( P(M, B) \) symbolises the number of ordered \( B \)-tuplets that can be created from \( M \) points and \( C_B^N \) symbolises the number of possible combinations of \( N \) points in sets of \( B \) elements.
6.3. EXPLORING THE CONFIGURATION SPACE

(a) A set of 200 line segments

(b) The $\alpha$ and $\beta$ parameters of the supporting lines

Figure 6.2: Determination of point of intersection of the supporting lines by fitting a line in the $\alpha$, $\beta$ space.
6.3.1 Exhaustive Search

One may try to solve the correspondence problem by generating all the possible matchings and evaluating each one. The matching which minimises equation (6.9) could be chosen as the best one.

Let us demonstrate the technique by means of an example (see figure 6.3). The four reference points \( C, D, E \) and \( F \) shown in figure 6.3a can be used in order to determine the projective coordinates of points 1, 2, 3 and 4. These points can be reconstructed onto the right image thus obtaining points 1', 2', 3' and 4' shown in figure 6.3b in white squares. In order to solve the correspondence problem all possible matches have to be generated and evaluated as shown in table 6.1. It is then clear that the best solution is the correct one with the cost function of 4. It is interesting, however, to consider permutation 1243 with a cost of 7. This is a very good suboptimal solution arising from the fact that both points 3 and 3' as well 4 and 4' lie on the same line. If the four points were exactly co-linear our technique would not be able to discriminate between the two solutions.

The size of the configuration space of the problem is given by (6.10) and it is prohibitively large for most practical problems to be searched exhaustively. Therefore, more efficient ways of exploring the configuration space and obtaining the optimal solution are required.

6.3.2 From Permutations to Combinations

There is a simple way to reduce the number of possible solutions of the optimisation problem to

\[
C_B^M C_B^N = \frac{M!}{(M - B)! B!} \frac{N!}{(N - B)! B!}
\]

(6.11)

where \( C_B^M \) symbolises the number of possible combinations of \( M \) points in sets of \( B \) elements. This number is smaller by a factor of \( B! \) compared to the one in (6.10). This can be accomplished if we consider that for any two sets of \( B \) points, one on the left image and the other on the right, there is only one way of matching them, therefore, it is not necessary to consider all permutations.

Indeed, if we sort the two sets of points from top to bottom, for example, the only allowable match is the one which matches the top-most points of the two sets, the second top-most points and so on. Any other match would result in lines that connect the matched points to intersect in the area between the two sets — something that contradicts
6.3. EXPLORING THE CONFIGURATION SPACE

(a) Left image. Note the four reference points C, D, E and F. Points 1, 2, 3 and 4 have been identified.

(b) Right image. The white squares are the transformed points from the left image (virtual points) and the white circles are the points identified on the right image.

Figure 6.3: Determining the correspondences of four points (1, 2, 3, 4) from left to right image using a set of four reference points (C, D, E, and F).
6.3. EXPLORING THE CONFIGURATION SPACE

<table>
<thead>
<tr>
<th>Permutation</th>
<th>$E_x$</th>
<th>$E_y$</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>-1513</td>
<td>-244</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1243</td>
<td>-1551</td>
<td>-285</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>1324</td>
<td>203</td>
<td>178</td>
<td>7710537</td>
<td></td>
</tr>
<tr>
<td>1342</td>
<td>470</td>
<td>257</td>
<td>351</td>
<td></td>
</tr>
<tr>
<td>1432</td>
<td>310</td>
<td>210</td>
<td>1371</td>
<td></td>
</tr>
<tr>
<td>1423</td>
<td>203</td>
<td>151</td>
<td>4183181</td>
<td></td>
</tr>
<tr>
<td>2134</td>
<td>234</td>
<td>193</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>2143</td>
<td>235</td>
<td>192</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>2314</td>
<td>226</td>
<td>203</td>
<td>657</td>
<td></td>
</tr>
<tr>
<td>2341</td>
<td>238</td>
<td>192</td>
<td>3033</td>
<td></td>
</tr>
<tr>
<td>2431</td>
<td>249</td>
<td>179</td>
<td>868</td>
<td></td>
</tr>
<tr>
<td>2413</td>
<td>242</td>
<td>189</td>
<td>2399</td>
<td></td>
</tr>
<tr>
<td>3214</td>
<td>241</td>
<td>200</td>
<td>807</td>
<td></td>
</tr>
<tr>
<td>3241</td>
<td>570</td>
<td>270</td>
<td>253</td>
<td></td>
</tr>
<tr>
<td>3124</td>
<td>203</td>
<td>171</td>
<td>2228592</td>
<td></td>
</tr>
<tr>
<td>3142</td>
<td>283</td>
<td>205</td>
<td>793</td>
<td></td>
</tr>
<tr>
<td>3412</td>
<td>274</td>
<td>205</td>
<td>2553</td>
<td></td>
</tr>
<tr>
<td>3421</td>
<td>202</td>
<td>145</td>
<td>637131</td>
<td></td>
</tr>
<tr>
<td>4231</td>
<td>283</td>
<td>214</td>
<td>29220</td>
<td></td>
</tr>
<tr>
<td>4213</td>
<td>287</td>
<td>234</td>
<td>15806</td>
<td></td>
</tr>
<tr>
<td>4321</td>
<td>202</td>
<td>84</td>
<td>184723130</td>
<td></td>
</tr>
<tr>
<td>4312</td>
<td>289</td>
<td>241</td>
<td>846</td>
<td></td>
</tr>
<tr>
<td>4132</td>
<td>285</td>
<td>222</td>
<td>5738</td>
<td></td>
</tr>
<tr>
<td>4123</td>
<td>202</td>
<td>83</td>
<td>178594582</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Possible matches and cost evaluation. The coordinates of the epipole $E$ are also shown.

the assertion that all such lines meet at the epipole. However, the size of the configuration space is still extremely large for big problems.

6.3.3 Branch and Bound Algorithm

Because of the geometry of our problem we can use the Branch and Bound algorithm [19]. The essential ingredients of the branch and bound algorithm are a tree-structured representation of the configuration space and a monotonically increasing cost function.

As we know roughly the direction along which the epipole is expected to be, we first sort the points along a direction orthogonal to that and index each set accordingly. From now on we shall assume that all left and right points, $(p_i, p_j)$, are sorted with their indices $i$ and $j$ indicating the order.
First a tree is constructed with the first pair of corresponding points, say \((p_1, p'_1)\), as its root. Then \((N - 1)(M - 1)\) children nodes, corresponding to the \((p_i, p'_j)\), \(2 \leq i \leq N, 2 \leq j \leq M\) matches, should be generated. In a similar fashion for every node that corresponds to a \((p_i, p'_j)\) match, \((N - i)(M - j)\) children nodes are attached until \(i = N\) or \(j = M\), which is our node generation terminating condition. Note that because of the observation made in section 6.3.2 once we assume, for example, the point 2 from the left image is matched with point 5 on the right image, the branches of this node will only include the options of point 3 from the left matching points 6, 7, ... , M on the right so that the matching lines do not intersect. Thus, the node in the tree which will represent the match \((p_2, p'_5)\) will have no children nodes. This tree structure is repeated for all possible matches (i.e. for root nodes \((p_1, p'_2), (p_1, p'_3)\) etc).

Depth-first search of the above tree would yield the optimal solution, however, this would be equivalent to the exhaustive search as described in the previous section. However, once partial solutions are worse than the best solution already found, the search of all children nodes can be abandoned early if we have a non-decreasing cost function. We prove next that the cost function is indeed non-decreasing. However, we can understand this intuitively, as the cost function we use measures the spread of the points which represent the intersecting lines in the dual (parameter) space: The addition of an extra line to the previous set of lines will always increase the area of the cluster of intersection points or leave it unchanged.

Thus, we satisfy the monotonicity criterion and a branch and bound algorithm can be used efficiently in order to search only a subspace of the original configuration space and still guarantee that the optimal solution can be obtained. Thus, we start generating complete solutions, keeping track of the best one found so far and give up exploring any path as soon as its cost becomes greater than the best found. Using this technique we are guaranteed that the the optimal solution will be found.

**Proof that the cost function is monotonic**

We shall now prove mathematically that the cost function, equation (6.9), increases monotonically. Therefore, the addition of an extra point \(p_{N+1}\) to the already existing \(N\) points will result in a value greater or equal to the one resulting from the original \(N\) points, i.e.:

\[
f(\alpha_i, \beta_i; i = 1, \ldots, N) \leq f(\alpha_i, \beta_i; i = 1, \ldots, N + 1)\]
Equation (6.9) can be rewritten for \( N + 1 \) points:

\[
\frac{1}{2} \sum_{i=1}^{N+1} \sum_{j=1}^{N+1} \left[ (\alpha_i - \bar{\alpha})(\beta_j - \bar{\beta}) - (\alpha_j - \bar{\alpha})(\beta_i - \bar{\beta}) \right]^2
\]

where \( \bar{\alpha} \) and \( \bar{\beta} \) are the recalculated for \( N + 1 \) points mean values for \( \alpha_i \) and \( \beta_i \) respectively.

The above equation can be rewritten in terms of \( N \) points plus an extra positive quantity resulting from the \((N + 1)\)th point:

\[
f(\alpha_i, \beta_i; i = 1, \ldots, N + 1) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[ (\alpha_i - \bar{\alpha})(\beta_j - \bar{\beta}) - (\alpha_j - \bar{\alpha})(\beta_i - \bar{\beta}) \right]^2 + \\
\sum_{i=1}^{N} \left[ (\alpha_i - \bar{\alpha})(\beta_{N+1} - \bar{\beta}) - (\alpha_{N+1} - \bar{\alpha})(\beta_i - \bar{\beta}) \right]^2
\]

(6.12)

We note that the last term of equation (6.12) is always positive. Therefore, to prove that equation (6.12) is always greater or equal to equation (6.9) it suffices to show that the first term of equation (6.12) cannot be less than equation (6.9). This is true because equation (6.9) gets its minimum value for \( \bar{\alpha} \) and \( \bar{\beta} \). Consider a function of two variables \( a \) and \( b \) and \( 2N \) independent variables \( \alpha_i \) and \( \beta_i \) with \( i = 1, \ldots, N \) defined by:

\[
f(a, b; \alpha_i, \beta_i, i = 1, \ldots, N) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left[ (\alpha_i - a)(\beta_j - b) - (\alpha_j - a)(\beta_i - b) \right]^2
\]

The minimum of \( f(a, b; \alpha_i, \beta_i, i = 1, \ldots, N) \) can be obtained by solving the simultaneous equations:

\[
\frac{\partial f(a, b)}{\partial a} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left[ (\alpha_i - a)(\beta_j - b) - (\alpha_j - a)(\beta_i - b) \right] (\beta_i - \beta_j) = 0 \quad (6.13a)
\]

\[
\frac{\partial f(a, b)}{\partial b} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left[ (\alpha_i - a)(\beta_j - b) - (\alpha_j - a)(\beta_i - b) \right] (\alpha_j - \alpha_i) = 0 \quad (6.13b)
\]

Solving the above equations for \( a \) and \( b \) yields:

\[
a = \frac{1}{N} \sum_{i=1}^{N} \alpha_i, \quad b = \frac{1}{N} \sum_{i=1}^{N} \beta_i
\]

which by definition are \( \bar{\alpha} \) and \( \bar{\beta} \) respectively. Note that \( f(a, b; \alpha_i, \beta_i, i = 1, \ldots, N) \) has a minimum at \( a = \bar{\alpha} \) and \( b = \bar{\beta} \) because its Hessian matrix, \( H \), is positive semi-definite. Indeed, \( H \), which is defined by

\[
H = \begin{bmatrix}
\frac{\partial^2 f(a, b)}{\partial a^2} & \frac{\partial^2 f(a, b)}{\partial a \partial b} \\
\frac{\partial^2 f(a, b)}{\partial b \partial a} & \frac{\partial^2 f(a, b)}{\partial b^2}
\end{bmatrix}
\]
6.3. EXPLORING THE CONFIGURATION SPACE

can be found to be

\[ H = \begin{bmatrix} \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_j)^2 & \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_j)(\beta_j - \beta_i) \\ \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_j)(\beta_j - \beta_i) & \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_j)^2 \end{bmatrix}. \]  

(6.14)

Matrix \( H \) is positive semi-definite if its two principal determinants, \( \det H_{11} = |h_{11}| \) and \( \det H_{22} = \det H \), are greater or equal to zero [5]. Therefore,

\[ \det H_{11} = \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_j)^2 \]  

(6.15)

and

\[ \det H = \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_j)^2 \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_j)^2 - \left( \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_j)(\beta_j - \beta_i) \right)^2 
\]

\[ = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} [(\alpha_i - \alpha_j)(\beta_k - \beta_l) - (\alpha_i - \alpha_j)(\beta_j - \beta_i)(\alpha_k - \alpha_l)(\beta_i - \beta_k)] 
\]

\[ = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_j)^2(\beta_k - \beta_l)^2 - 
\]

\[ - \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_j)(\beta_j - \beta_i)(\alpha_k - \alpha_l)(\beta_i - \beta_k) 
\]

\[ = \frac{1}{2} \left( \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_j)^2(\beta_k - \beta_l)^2 + \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_j)^2(\beta_j - \beta_i)^2 - 
\]

\[ - 2 \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_j)(\beta_j - \beta_i)(\alpha_k - \alpha_l)(\beta_i - \beta_k) \right) \]  

(6.16)

Changing the dummy indices in the last two terms of the above equation yields:

\[ \det H = \frac{1}{2} \left( \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_j)^2(\beta_k - \beta_l)^2 + \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_k)^2(\beta_j - \beta_l)^2 - 
\]

\[ - 2 \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} (\alpha_i - \alpha_j)(\beta_j - \beta_l)(\alpha_k - \alpha_l)(\beta_i - \beta_k) \right) \]  

(6.17)

It can be easily shown that \( \det H_{11} \geq 0 \) and \( \det H \geq 0 \) since they are sum of squares. Therefore, \( H \) is positive semi-definite and the vector \((\tilde{\alpha}, \tilde{\beta})\) causes the function \( f(a, b; \alpha_i, \beta_i, i = 1, \ldots, N) \) to get its minimum value.

If the modified mean values \( \tilde{\alpha} \) and \( \tilde{\beta} \) which result from the addition of an extra point is different from \( \tilde{\alpha} \) and \( \tilde{\beta} \) the cost function will increase. This proves our argument about
the monotonicity criterion. Indeed, we have shown that the addition of extra lines will always increment our cost function. In some special cases, when the $\alpha$ and $\beta$ parameters of the line happen to coincide with the mean values of the cluster, the cost function will remain unchanged, but under no circumstances will it get a value less than the previous one. Therefore, the branch and bound algorithm can be used in order to search only part of the configuration space and still guarantee that the optimal solution will be obtained. The size of the sub-space searched depends on how fast a good sub-optimal solution is obtained, thus establishing better bounds and reducing the search space.

### 6.3.4 Hough Transform

The success of a matching algorithm strongly depends on the robustness to noise [25]. It is well known that Hough Transform techniques are noise resistant [13]. Hence, in this section we investigate the applicability of a Hough-Transform-like algorithm to be used in order to search the configuration space and still guarantee to find the optimal solution. Essential ingredients for a Hough-Transform-like algorithm are an accumulator space and a voting scheme. Both will be described next. The accumulator in our case is a two-dimensional space with dimensions $N$ by $M$ and every bin $b_{ij}$ indicates the confidence in point $p_i$ from the left image corresponding to $p'_j$ from the right image.

All bins are set initially to zero. Then, all possible valid triplets of correspondences are generated and the cost is evaluated according to equation (6.9). If the calculated cost for a specific triplet is less than a prespecified value, $t_1$, the three appropriate bins are incremented by one. Finally, after all possible triplets have been examined, the accumulator array is checked to yield the matches. First, the maximum element is found and the corresponding match is the one with the highest probability. The row and the column of the accumulator array containing the match are then removed, since one-to-one matches only are allowed. The same procedure is repeated until all rows and columns of the array have been removed or until the value of the bin is less than a value, $t_2$, which is the minimum probability required for accepting a valid match.

### 6.3.5 Randomised Hough Transform

There may be practical cases where applying the algorithm described in the previous section would be very inefficient due the large number of triplets which have to be generated. Therefore, a randomised scheme could be used instead. Note, however, that the solution may be a sub-optimal one as optimality cannot be guaranteed anymore.
6.3. EXPLORING THE CONFIGURATION SPACE

The idea is to generate random triplets until one with a cost less than \( t_i \) is found. Then, two of the corresponding pairs should be kept and all triplets which can be formed by adding another possible matching pair exhaustively tested. The last two steps should be repeated until a certain number of triplets has been examined. Finally, the best matching found is extracted from the accumulator using the same technique as described in the previous section.

If, however, we start choosing triplets at random, we may be exploring parts of the configuration space which is very scarcely populated by valid triplets. To avoid this, we estimate first how often each triplet is expected to arise in a legitimate configuration. It is easy to see that the frequency by which each correspondence, \((p_i, p'_j)\) for \(i = 1, \ldots, N\) and \(j = 1, \ldots, M\), occurs is given by:

\[
P(p_i, p'_j) = \frac{C^{i-1}_{i}C^{j-1}_{j} + C^{N-i}_{i}C^{M-j}_{j} + (i-1)(j-1)(N-i)(M-j)}{\sum_{i=1}^{N} \sum_{j=1}^{M} \left[ C^{k-1}_{k}C^{l-1}_{l} + C^{N-k}_{k}C^{M-l}_{l} + (k-1)(l-1)(N-k)(M-l) \right]}
\]

\[
= \frac{C^{i-1}_{i}C^{j-1}_{j} + C^{N-i}_{i}C^{M-j}_{j} + (i-1)(j-1)(N-i)(M-j)}{3C^N_iC^M_j}
\]

(6.18)

Let us illustrate this point by means of an example (see figure 6.4) where the three terms in the numerator of equation 6.18 come from. There are \(N = 8\) points on the left and \(M = 10\) points on the right. It is easy to calculate by how many valid triplets of correspondences the \((p_5, p'_6)\) match contributes. Note that the \((p_5, p'_6)\) match separates the cluster points in two sets, \(A\) and \(B\). There are three possible ways of adding the remaining two matches in order to complete a valid triplet:

- **Both matches in Set A**: There are \(C^4_2C^7_2 = 126\) possible matches belonging in this category.

- **Both matches in Set B**: There are \(C^5_2C^4_2 = 3\) possible matches belonging in this category, namely: \(((p_6, p'_5), (p_7, p'_i)), ((p_8, p'_6), (p_9, p'_i))\) and \(((p_7, p'_5), (p_8, p'_6))\).

- **One in Set A, one in Set B**: There are \(4 \times 7 = 28\) single matches in Set A and \(3 \times 2 = 6\) single matches in Set B which multiplied together yield 168 possible matches in this category.

Since all three possible ways of generating triplets yield distinct solutions (without any overlap) the total number of triplets in which the match \((p_5, p'_6)\) participates in, is the sum \(126 + 3 + 168 = 297\). (see table 6.2 for a complete list of probabilities). Thus, to initialize the Randomised Hough Transform we choose our triplets according to the above probability density function.
6.4 Experimental Results

The matching algorithms described above have been implemented in the C programming language and tested on both synthetic and real data.

6.4.1 Synthetic Data

First, the \((x, y, z)\) world coordinates of 30 points were randomly generated, distributed uniformly inside a cube of side 3 meters. Then, they were projected both onto the left and right image by using appropriate transformation matrices which simulated two cameras at a height of 7 meters looking at the cluster of points and with their viewing lines at 90
6.4. EXPERIMENTAL RESULTS

Figure 6.5: A synthetic example for matching 30 points.

degrees (figure 6.5). In addition, 4 reference coplanar points, none three colinear, were projected both onto the left and right image (not shown in figure 6.5).

Although this is a trivial example for the branch and bound algorithm, because there are no outliers, it served as a good test for the Hough Transform algorithm. Indeed, we run the Randomised Hough Transform, by generating valid triplets according to the probabilities given by equation 6.18 as described in section 6.3.5, which yielded the optimal matching after generating 20 thousand triplets out of a possible total of about 50 million. The final accumulator space is shown in table 6.3. Note, although there are many votes associated with wrong matches (mainly due to quantisation errors), the optimal matches are eventually obtained.

The usefulness and robustness of the Randomised Hough Transform algorithm is further illustrated by the addition of 10 outliers to the initial points. Then, the accumulator space remains the same for the valid matches and with zero or very small number of votes for the outliers. We also run the branch and bound algorithm for the case of having outliers which yielded the optimal matching after searching just 600 nodes of the tree. The time it took to execute the algorithms on a SPARC10 workstation is less than 1 CPU second for both examples.
Table 6.3: The accumulator for the synthetic example. The matches sorted by confidence level are: (20-20), (25-25), (18-18), (28-28), (6-6), (29-29), (23-23), (7-7), (10-10), (17-17), (9-9), (4-4), (19-19), (21-21), (9-9), (11-11), (12-12), (13-13), (14-14), (15-15), (16-16), (22-22), (24-24), (26-26), (27-27), (5-5), (1-1), (2-2), (3-3) and (30-30).

6.4.2 Real Data

The algorithms described in this paper have been successfully applied on many real images too. In this section we describe how the face of the stone seen by two cameras (figures 6.6a and 6.6b) was reconstructed in 3D.

Initially, the features, in our cases the circular marks painted on each face, were extracted by first performing edge detection to both images (see figures 6.6c and 6.6d). Then the edge strings were processed by a rule-based system which took into consideration all prior knowledge concerning our features (such as size, shape etc) in order to reject most of the edges. In the final stage of processing the grey-levels of neighbouring to the
edge pixels were taken into consideration in order to obtain the set of features shown in figures 6.6e and 6.6f. For displaying purposes the corresponding features have the same label (note that these are not the labels of the sorted sets used by the branch and bound algorithm). In addition, there are 7 and 6 outliers on the left and right image respectively. The centroids of the left features were obtained and using the 4 reference points A, B, C and D we obtained their virtual images on the right image (see figure 6.7). Then, the branch and bound algorithm was used and the optimal matching was obtained in less than 1 minute CPU time on a SPARC workstation.

The matched points were reconstructed in 3D taking into consideration the error analysis presented in the previous chapter. Every individual point was reconstructed by using the reference planes which were less sensitive to noise. A graphical illustration of the sensitivity to noise using the three different reference planes, ABCD, CDEF and AB EF can be seen in figure 6.8. These images show pictorially the error amplification coefficients which were computed for the particular example shown here. For example by looking at images 6.8a, 6.8c and 6.8e it is obvious that for the reconstruction of a point in the left part of the left image we should use the reference planes ABCD and AB EF.

Finally, the Euclidean distance between some of these points was calculated and compared to the ground truth (see table 6.4) which was obtained by manually measuring the distances within 1cm accuracy. Note that the percentage error shown in table 6.4 was calculated in terms of the width of the stone which was 2.8 meters.

6.5 Discussion

In this chapter the correspondence problem was cast as an optimisation problem. A cost function, suitable for evaluating every possible solution, was derived. Unfortunately, obtaining solution to this combinatorial optimisation problem by exhaustive search was found to be a difficult task for most real problems. However, careful ordering of the configuration space, based on geometrical constraints, was proved to be efficient.

An even more efficient technique, in the case of small number of outliers, was found to be a branch and bound algorithm. It was then, first shown how to create a tree-structured representation of the configuration space and secondly we formally proved that the cost function increases monotonically, both being essential ingredients for a branch and bound algorithm.
Figure 6.6: Experimental results based on real images.
Figure 6.7: Left image is transformed using the 4 reference points.

<table>
<thead>
<tr>
<th>Line Segment</th>
<th>Measurement (cm)</th>
<th>Reconstruction Error</th>
<th>Ground Truth (cm)</th>
<th>Absolute Difference (cm)</th>
<th>Percentage Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1P_2$</td>
<td>45</td>
<td>16.47</td>
<td>47</td>
<td>2</td>
<td>0.7%</td>
</tr>
<tr>
<td>$P_1P_3$</td>
<td>67</td>
<td>7.85</td>
<td>67</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>$P_2P_3$</td>
<td>82</td>
<td>14.40</td>
<td>83</td>
<td>1</td>
<td>0.3%</td>
</tr>
<tr>
<td>$P_3P_4$</td>
<td>37</td>
<td>6.28</td>
<td>38</td>
<td>1</td>
<td>0.3%</td>
</tr>
<tr>
<td>$P_3P_5$</td>
<td>38</td>
<td>3.05</td>
<td>39</td>
<td>1</td>
<td>0.3%</td>
</tr>
<tr>
<td>$P_3P_6$</td>
<td>47</td>
<td>5.73</td>
<td>45</td>
<td>2</td>
<td>0.7%</td>
</tr>
<tr>
<td>$P_3P_7$</td>
<td>53</td>
<td>1.15</td>
<td>53</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>$P_4P_5$</td>
<td>66</td>
<td>6.21</td>
<td>67</td>
<td>1</td>
<td>0.3%</td>
</tr>
<tr>
<td>$P_4P_6$</td>
<td>139</td>
<td>1.02</td>
<td>142</td>
<td>3</td>
<td>1.0%</td>
</tr>
<tr>
<td>$P_4P_7$</td>
<td>38</td>
<td>4.68</td>
<td>39</td>
<td>1</td>
<td>0.3%</td>
</tr>
<tr>
<td>$P_5P_6$</td>
<td>65</td>
<td>10.1</td>
<td>66</td>
<td>1</td>
<td>0.3%</td>
</tr>
<tr>
<td>$P_5P_7$</td>
<td>88</td>
<td>5.65</td>
<td>91</td>
<td>3</td>
<td>1.0%</td>
</tr>
<tr>
<td>$P_5P_8$</td>
<td>117</td>
<td>3.67</td>
<td>119</td>
<td>2</td>
<td>0.7%</td>
</tr>
<tr>
<td>$P_6P_7$</td>
<td>75</td>
<td>3.76</td>
<td>77</td>
<td>2</td>
<td>0.7%</td>
</tr>
<tr>
<td>$P_6P_8$</td>
<td>77</td>
<td>16.70</td>
<td>80</td>
<td>3</td>
<td>1.0%</td>
</tr>
<tr>
<td>$P_7P_8$</td>
<td>45</td>
<td>5.58</td>
<td>45</td>
<td>2</td>
<td>0.7%</td>
</tr>
</tbody>
</table>

Table 6.4: Relative measurement of distances compared to the ground truth which was measured manually with a ±1 cm accuracy. The reconstruction error is the sum of the $x$ and $y$ errors in the two reference planes as described in the previous chapter (section 5.1).
Figure 6.8: Illustration of the amplification factor of the error associated with the 3 reference planes.
However, particularly in the presence of noise and large number of outliers, robustness in a matching algorithm is of primary importance. A robust Hough-Transform-like technique, is therefore suggested. Also, the proposed randomised scheme has been found to be suitable for large scale problems.

Finally, experimental results and comparisons illustrate the usefulness of the suggested algorithms. Although, much work has been done previously on matching, the algorithms presented here can easily deal with non-coplanar features and with wide camera separation with no camera calibration parameters available. The only assumption is that the positions of the left and right images of four coplanar points, none three co-linear, are known. The technique degenerates when more than a pair of corresponding points lie on the same epipolar. This ambiguity can be resolved easily by the use of a properly positioned third camera.

References


REFERENCES


Chapter 7

Conclusions

In this thesis the \( \mathcal{NP} \)-hard [3] lay-planning or stock-cutting problems have been identified as a very important class of packing problems. Careful investigation of already existing algorithms has been performed and new efficient algorithms to overcome known limitations have been suggested. The problem of accurately and robustly determining the 3D shape of the objects prior to cutting has also been addressed and a formal methodology for robust reconstruction and feature matching has been developed.

In this final chapter, an overview of the thesis as well as the major contributions are given first and later, possible directions of future research are suggested.

7.1 Overview and major contributions of this work

First, in chapter 2, packing problems were introduced and a detailed survey of already existing algorithms both deterministic and stochastic was presented. An efficient rectangle packing algorithm which could take into account several constraints was then developed and its performance was demonstrated both theoretically and experimentally [4].

In chapter 3 the stock-cutting problem has been investigated and the necessity for developing an algorithm which could deal with objects of general shape to be cut soon emerged. The generalised stock-cutting problem which could take into account defective regions or holes on the material to be cut was formally defined. It was shown how morphological operators [7] can be a very efficient tool towards solving the problem. It was also shown how to transform the problem to a set-packing problem [1] and hence proved that it was \( \mathcal{NP} \)-hard. Nevertheless, a formulation of the problem as an integer-programming model with a totally unimodular constraint matrix enabled us to obtain the optimal solution to the unconstrained problem using the simplex algorithm [2]. A
stochastic technique was used to obtain good sub-optimal solutions for the general case in which constraints concerning the maximum number of tiles to be cut exist [5]. The techniques developed in this chapter can easily be generalised for the optimal cutting of 3D objects. Indeed, if the shape of the 3D object is known, and given that with the existing cutting equipment it can only be cut in slabs, the cutting problem of a 3D object reduces to a 2D cutting problem which can even be reduced to a series of 1D cutting problems, one for each possible orientation of the block. Thus, it was not necessary for the 3D cutting problem to be specifically considered. Instead we concentrated on defining the 3D shape of the block that is to be cut.

The purpose of chapter 4 was the investigation of the suitability of already existing stereo-vision techniques to the problem of reconstructing the three-dimensional shape of the object to be cut. Techniques based on determining the camera calibration parameters [10] were found inappropriate due to the large number of calibration points needed. Therefore, a technique based on Projective Geometry [8, 9], which did not need any camera calibration parameters seemed to be attractive due to its relative high accuracy and the small number of reference points needed. However, it was found experimentally that in some cases great sensitivity to noise was to be expected.

Therefore, a detailed sensitivity analysis was undertaken in chapter 5. Formal mathematical analysis of every single step of the Projective geometry based technique leads to a set of rules that have to be obeyed if robust and accurate reconstruction in 3D is to be expected when applying it in real life problems [6]. Up to this point in order to reconstruct in 3D it was assumed that the corresponding features in the left and right images were given, i.e. the correspondence problem had somehow been solved.

Nevertheless, automatic solutions to the correspondence problem, which becomes extremely difficult in the case of non-coplanar features and cameras set at 90 degrees, were the subject of chapter 6. The correspondence problem was cast as an optimisation problem and a cost function for evaluating possible solutions was derived. It was formally proved that the cost-function was increasing monotonically and this enabled the use of a branch and bound algorithm in order to optimally solve the optimisation problem and thus obtain a solution to the matching problem. A Hough Transform based algorithm was also suggested for solving the optimisation problem which seemed to perform more robustly in the case of many outliers. A randomised scheme of the Hough Transform was also briefly discussed. Results on synthetic and real data demonstrated the usefulness of
7.2 Future Directions

It is hoped that this thesis provides a clear and consistent presentation of the steps taken towards the automation of the Lay-Planning process of objects whose shape is robustly determined by using stereo-vision techniques. Nevertheless, there is a number of problems which are interesting prospects for future work.

In chapter 3 the novel combination of morphological operators with an integer-programming formulation leads to an efficient stock cutting algorithm for objects of general shape with holes or defective regions of general shape. The algorithm could obtain the optimal pattern of cutting rectangular tiles. These ideas can be further extended to deal not only with patterns of rectangular tiles but any general shape. Although morphological operators can easily handle general shapes, the problem of rotating individual objects prior to cutting has to be further investigated as it is an $\mathcal{NP}$-hard large-scale optimisation problem.

Guidelines for robust and accurate three-dimensional reconstruction by using as few as 6 reference points were given in chapter 5. However, more reference points and/or planes could lead to even more accurate reconstruction of points in the three-dimensional space. Careful consideration should be given, however, into the way one can use the extra reference points and the contribution that each individual subset should have for the final reconstruction. Perhaps, possible subsets of reference points could have a weighted contribution (according to the expected immunity to noise presented in chapter 5) which could lead to an even more accurate and robust reconstruction.

It was mentioned earlier in chapter 6 that in the case of matching features that lie on the same epipolar lines the proposed algorithms would not be able to distinguish the correct solution. Careful modification/extension of the algorithms could incorporate information obtained by a third viewpoint and resolve any ambiguities.

References


Appendix A

Functions of \( n \) \( m \)-tuples of Random Variables

If \( g = g(x_{11}, x_{21}, \ldots, x_{m1}, x_{12}, x_{22}, \ldots, x_{m2}, \ldots, x_{1n}, x_{2n}, \ldots, x_{mn}) \) and \( h = h(x_{11}, x_{21}, \ldots, x_{m1}, x_{12}, x_{22}, \ldots, x_{m2}, \ldots, x_{1n}, x_{2n}, \ldots, x_{mn}) \) are two functions of \( n \) \( m \)-tuples of random variables, then the covariance \( \sigma_{gh}^2 \) as well as the variances \( \sigma_{gg}^2 \) and \( \sigma_{hh}^2 \) of \( g \) and \( h \) can be estimated in terms of the variance and covariance of \( x_i \) and \( y_i \). More formally,

\[
\begin{align*}
g &= g(x_{di}) \forall d, i \\
h &= h(x_{di}) \forall d, i \\
\omega &= (\bar{x}_{di}) \forall d, i \\
E\{x_{di}\} &= \bar{x}_{di} \forall d, i \\
E\{(x_{di} - \bar{x}_{di})(x_{bj} - \bar{x}_{bj})\} &= \sigma_{x_{di}x_{bj}}^2 \forall i, d, b \\
E\{(x_{di} - \bar{x}_{di})(x_{bj} - \bar{x}_{bj})\} &= 0 \forall i, j, d, b, i \neq j
\end{align*}
\]

If \( g \) and \( h \) are represented by their second-order Taylor series expansion about the point \( \omega \)

\[
\begin{align*}
g &\approx g(\omega) + \sum_{d,i} \frac{\partial g}{\partial x_{di}} |_\omega (x_{di} - \bar{x}_{di}) + \frac{1}{2} \sum_{d,i,j} \frac{\partial^2 g}{\partial x_{di}\partial x_{dj}} |_\omega (x_{di} - \bar{x}_{di})(x_{dj} - \bar{x}_{dj}) \quad (A.2) \\
h &\approx h(\omega) + \sum_{d,i} \frac{\partial h}{\partial x_{di}} |_\omega (x_{di} - \bar{x}_{di}) + \frac{1}{2} \sum_{d,i,j} \frac{\partial^2 h}{\partial x_{di}\partial x_{dj}} |_\omega (x_{di} - \bar{x}_{di})(x_{dj} - \bar{x}_{dj}) \quad (A.3)
\end{align*}
\]

Taking expectation values of both sides of (A.2) and (A.3):

\[
\begin{align*}
E\{g\} &\approx g(\omega) + \frac{1}{2} \sum_{d,i} \left( \sigma_{x_{di}x_{di}}^2 \sum_j \frac{\partial^2 g}{\partial x_{dj}\partial x_{dj}} |_\omega \right) \quad (A.4) \\
E\{h\} &\approx h(\omega) + \frac{1}{2} \sum_{d,i} \left( \sigma_{x_{di}x_{di}}^2 \sum_j \frac{\partial^2 h}{\partial x_{dj}\partial x_{dj}} |_\omega \right) \quad (A.5)
\end{align*}
\]

Multiplying (A.2) by (A.3) and keeping up to the second order terms yields:

\[
g h \approx g(\omega)h(\omega) + \sum_{d,i,j} \frac{\partial g}{\partial x_{di}} |_\omega \frac{\partial h}{\partial x_{dj}} |_\omega (x_{di} - \bar{x}_{di})(x_{dj} - \bar{x}_{dj}) + \sum_{d,i,j} \frac{\partial^2 g}{\partial x_{di}\partial x_{dj}} |_\omega (x_{di} - \bar{x}_{di})(x_{dj} - \bar{x}_{dj}) \quad (A.6)
\]

132
Taking expectation value of (A.6):

\[
E\{gh\} = g(\omega) h(\omega) + \sum_{d,b} \left( \sigma_{x_b x_b}^2 \sum_i \frac{\partial^2 h}{\partial x_{di} \partial x_{bi}} \right) + 
\]

\[
+ \frac{1}{2} g(\omega) \sum_{d,b} \left( \sigma_{x_b x_b}^2 \sum_i \frac{\partial^2 h}{\partial x_{di} \partial x_{bi}} \right) + 
\]

\[
+ \frac{1}{2} h(\omega) \sum_{d,b} \left( \sigma_{x_b x_b}^2 \sum_i \frac{\partial^2 g}{\partial x_{di} \partial x_{bi}} \right) \quad (A.6)
\]

Multiplying (A.4) by (A.5) and keeping only up to second order terms we obtain

\[
E\{g\} E\{h\} = g(\omega) h(\omega) + 
\]

\[
+ \frac{1}{2} g(\omega) \sum_{d,b} \left( \sigma_{x_d x_d}^2 \sum_i \frac{\partial^2 h}{\partial x_{di} \partial x_{bi}} \right) + 
\]

\[
+ \frac{1}{2} h(\omega) \sum_{d,b} \left( \sigma_{x_b x_b}^2 \sum_i \frac{\partial^2 g}{\partial x_{di} \partial x_{bi}} \right) \quad (A.8)
\]

By definition

\[
\sigma_{gh}^2 = E\{(g - E\{g\}) (h - E\{h\})\} = 
\]

\[
= E\{gh\} - E\{g\} E\{h\} \quad (A.9)
\]

Substituting (A.7), (A.8) into (A.9) we obtain:

\[
\sigma_{gh}^2 = \sum_{d,b} \left( \sigma_{x_d x_b}^2 \sum_i \frac{\partial g}{\partial x_{di}} \frac{\partial h}{\partial x_{bi}} \right) \quad (A.10)
\]

The variances of \( g \) and \( h \), namely \( \sigma_{gg}^2 \) and \( \sigma_{hh}^2 \), can be calculated directly from (A.10):

\[
\sigma_{gg}^2 = \sum_{d,b} \left( \sigma_{x_d x_d}^2 \sum_i \frac{\partial g}{\partial x_{di}} \frac{\partial g}{\partial x_{bi}} \right) \quad (A.11)
\]

\[
\sigma_{hh}^2 = \sum_{d,b} \left( \sigma_{x_b x_b}^2 \sum_i \frac{\partial h}{\partial x_{di}} \frac{\partial h}{\partial x_{bi}} \right) \quad (A.12)
\]