Dense matching and image segmentation using projective geometry.

David John O'Connell

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/1631

This online database contains the full-text of PhD dissertations and Masters’ theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
Dense Matching and Image Segmentation using Projective Geometry

David John O'Connell

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirements for the
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
2002

©2002 David John O’Connell
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-75800-1
Abstract

Dense matching and image segmentation are fundamental image analysis operations. These operations are required by many computer vision applications. Artificial view-synthesis, 3D scene reconstruction, token tracking and augmented reality are examples of applications that rely heavily on these primitives. The speed and accuracy of such applications rely on the quality of the matching and segmentation. As a result, solutions to these problems have been widely researched. However, due to the difficulty of these problems, there is no universal solution.

Most solutions to these two problems make certain assumptions. First, dense matching and image segmentation are often viewed as separate problems. Second, most image segmentation techniques operate on only a single image. This introduces a technique that simultaneously performs image segmentation and dense matching of planar surfaces in a stereo pair of images. Using three matched points from an arbitrary plane, and four other matched points, a projective mapping, known as a homography, is calculated. This homography is used to iteratively grow a region in both images. The result is a matched and segmented plane.

Practical tests comparing the computation time of this method to traditional matching techniques are presented. These results are used to motivate the use of the planar technique as a primary step for reducing the overall computation time for dense matching and image segmentation.
## Contents

Abstract iv

1 Introduction 1
  1.1 The Origin of Computer Vision 1
  1.2 Overview of Computer Vision 2
  1.3 Motivation of Thesis 3
  1.4 Plane Matching and Segmentation 3
  1.5 Overview of Thesis 4

2 Image Formation and Image Processing Basics 5
  2.1 Geometric Transformations in Image Formation 5
     Euclidean Transformations 6
     Affine Transformations 6
     Projective Transformations 7
  2.2 Projective Transformations 8
  2.3 The Pinhole Camera Model 9
  2.4 Epipolar Geometry and the Epipolar Constraint 10
  2.5 Summary 11

3 Overview of Image Segmentation 12
  3.1 Definition and Classifications of Segmentation 12
  3.2 Algorithm Classifications 13
  3.3 Global Techniques 15
     Thresholding 15
     Multithresholding Techniques 16
     Other Global Techniques 16
## CONTENTS

3.4 Boundary-Detection Techniques ........................................... 17
  Edge Detection .......................................................... 17
  Edge Linking ............................................................. 18
  Active Contour Models ................................................... 19
  Other Boundary-Detection Techniques .................................. 20
3.5 Region Detection .......................................................... 20
  Early Region Detection .................................................. 20
  Advanced Region Segmentation Techniques .......................... 21
3.6 Integrated Hybrid Techniques ............................................. 23
3.7 Conclusion .................................................................. 23

4 Correlation-Based Matching Techniques ............................... 25
  4.1 Definition of Point Matching ........................................... 25
  4.2 Anatomy of a Matching Algorithm ................................... 26
  4.3 Correlation Functions .................................................. 27
  4.4 Sparse Matching ......................................................... 27
  4.5 Review of Existing Technique .......................................... 28
    Exhaustive Approach with the Epipolar Constraint .............. 28
    Dense Matching Using Iterative Improvement Algorithms .... 29
    Dynamic Programming .................................................. 30
    Integer Programming .................................................... 31
  4.6 Computational Expense of Correlation Measures ................. 31
  4.7 Analysis of Dense Matching Using the Epipolar Constraint .... 32
  4.8 Conclusion .................................................................. 34

5 Matching and Segmentation using Projective Geometry .......... 35
  5.1 Overcoming the Empirical Nature of Conventional Techniques .. 35
  5.2 Plane Matching .......................................................... 36
  5.3 Calculating the Plane Homography ................................... 36
  5.4 Dense Matching and Region Segmentation ........................ 39
  5.5 Calculating the Seed Region and Control Points ................ 40
  5.6 Growing the Region .................................................... 41
  5.7 Finding the Initial Matches .......................................... 42
  5.8 Discussion of Homography Growing .................................. 43
  5.9 Experiments .............................................................. 46
List of Figures

2.1 Box Before and After a Euclidean Transformation ........................................... 6
2.2 Box Before and After an Affine Transformation .............................................. 7
2.3 Box Before and After a Projective Transformation .......................................... 8
2.4 The Pinhole Camera Model ................................................................. 9
2.5 Epipolar Geometry ................................................................. 11

4.1 A stereo pair of images displaying several matches ........................................ 26
4.2 Example Patterns for Interest Points in a 5 × 5 neighbourhood: a) dot, b) corner, c) junction .............................................................. 28
4.3 Dynamic Programming Performs Matching by Finding a Path Between S and T .......................................................... 30
4.4 The Maximum Weighted Matching Approach ............................................. 32

5.1 A Highly Planar Scene ................................................................. 37
5.2 Three Points Defining a Planar Region ..................................................... 37
5.3 Basic Steps for Plane Matching/Segmentation ........................................... 39
5.4 Growth Vector for Control Point ............................................................ 42
5.5 Scenario 1 - Edge Detection Properly Plants the Seed ................................ 43
5.6 Scenario 2 - Edge Detection Improperly Plants the Seed ................................ 44
5.7 The lighthouse represents a non-homogeneous region .................................. 45
5.8 Stereo Pair of a Simple Plane, 576 × 768 Pixels Each ................................... 47
5.9 Stereo Pair of Building Model, 576 × 384 Pixels Each ................................... 48
5.10 Stereo Pair of a Church, 576 × 384 Pixels Each ......................................... 48
5.11 Seed Region and Growth for Text Plane 1 ................................................. 50
5.12 Seed Region and Growth for Text Plane Test 2 ......................................... 51
5.13 Seed Region and Growth for Wall Scene Test 1 ......................................... 52
5.14 Seed Region and Growth for Wall Scene Test 2 ......................................... 53
5.15 Seed Region and Growth for Wall Scene Test 3 ......................................... 54
LIST OF FIGURES

5.16 Seed Region and Growth for Church Scene Test 1 ........................................ 56
5.17 Seed Region and Growth for Church Scene Test 2 ........................................ 57
5.18 Seed Region and Growth for Church Scene Test 2 ........................................ 58
List of Tables

4.1 Common Correlation Functions ................................................. 27
4.2 Mathematical Operations Performed for Different Correlation Measures .......... 33

5.1 Results for Simple Plane Tests .................................................. 49
5.2 Results for Wall Tests ............................................................... 53
5.3 Results for Church Tests ............................................................ 55
Chapter 1

Introduction

This chapter presents the motivations for this thesis. First, the evolution of computer vision is discussed. Then, the separation of computer vision into high and low-level activities is illustrated with several examples. Following this, the low-level primitives of image segmentation and point matching are introduced. Finally this chapter outlines the layout for the rest of this thesis.

1.1 The Origin of Computer Vision

Computer vision originated from the two separate, but related, fields of artificial intelligence (AI) and robotics. The study of AI came from the desire for computers to solve problems that required more than simple calculations. Researchers in AI develop problem solving techniques that imitate intelligent behavior. A common example of an AI application is a chess playing program. The ability to play chess well is usually considered a measure of intelligence. Therefore, a chess playing program could be considered intelligent. One of the basic AI paradigms is the intelligent agent model. In this model the intelligent agent is able to perceive aspects of its environment and react based on these perceptions.

The field of robotics is concerned with creating machines that can perform physical activities. Early robots were either guided by a human controller, or performed their function without considering change to their environment. To be useful for many applications, the robot must be able to work autonomously within its environment. Such a robot is an intelligent agent that must make intelligent decisions based on the state of its environment. Thus, the field of robotics has become coupled with AI to create robots that can operate autonomously in a dynamic environment.

Consider the design of such an autonomous robot. The aforementioned intelligent agent model dictates that there is a dependence of a robots actions on its perceptions. Thus, the quality of the
perceptions can determine how effectively the robot reacts to the environment. This motivates the use of many different types of stimuli. These include tactile, olfactory, pressure, temperature and visual information.

There is no best type of information for a robot to perceive. The information needed is highly dependent on the task of the robot. In fact, many different senses may be required to perform certain tasks. One sense people rely heavily on is vision. Thus, it is natural to adapt this sense for use by an intelligent agent. This need to process visual information has lead to the conception, and continues to drive, the field of computer vision.

1.2 Overview of Computer Vision

Computer vision is the study of the extraction of information from visual stimuli, as well as the interpretation and uses of this information. There are several different facets to this field. These are illustrated by the following questions [26]:

1. What information should be extracted from the outputs of the visual sensors?

2. How is this information to be extracted?

3. How should this information be represented?

4. How must the information be used to allow a robotics system to perform its tasks?

The first three questions deal with low-level aspects of computer vision. The prototypical vision system has one or more cameras used to capture images. Primitive low-level operations are needed to extract information from these images. This involves determining what to extract, developing extraction algorithms and defining a representation for the resultant data. Commonly used low-level information includes edges, regions, texture, depth and the correspondence between pixels in multiple photographs of the same scene, known as point matching.

The final question introduces the notion of high-level vision. This is where the aforementioned primitives are used in an application such as robotics, augmented reality, medical imaging or motion tracking. For example, a robot would use several of these primitive operations in planning a path through a room. To avoid obstacles the robot may use region extraction to identify objects and a depth map to avoid the objects. Augmented reality is a field similar to virtual reality. Virtual reality submerges an individual into a artificially modeled 3D looking scene. Augmented reality uses computer generated objects to enhance a person's view of the real world. To do this, the application must identify a region where the virtual object is to be inserted. For an image sequence, this location may need to be matched over multiple images.
1.3 Motivation of Thesis

One of the most fundamental operations of computer vision is matching. This is a very difficult and expensive operation, yet it is critical to the success of many high-level applications. Another fundamental operation is region segmentation. This is a first step to breaking an image into its composite objects. This thesis explores a technique for simultaneously segmenting and matching entire planar surfaces by region growing.

1.4 Plane Matching and Segmentation

Matching is used to define a relationship between corresponding points in multiple images of the same scene. These correspondences are important for many applications of computer vision, such as augmented reality and path planning. Thus, the efficiency and robustness of these methods is very important. Classical methods take two images, left and right, and attempt to match each point individually. For an $n \times m$ image the algorithm will run $nm$ times. Similarly, classical image segmentation techniques must perform a membership test for each pixel in the image. Since the computation time for most classical methods is high, it is not feasible for many real-time applications.

Planar surfaces are abundant in many scenes. This fact can be exploited to quickly match and segment a large portion of the image. The general steps for a method to segment and match the image are as follows:

1. Identify and match a small set of interesting image pixels including points, corners and junctions

2. Identify and match three points from a planar surface in the scene

3. Using the points from the first two steps calculate a projective mapping transforming the planar surface from the left image to the right image

4. Grow the planar region as long as the mapping function is valid

The technique introduced in Chapter 5 of this thesis uses this strategy to simultaneously match and segment the image. Most of the classical point matching and image segmentation techniques approach these problems separately. Thus, the technique presented in Chapter 5 gives a uniquely unified view of these two difficult vision problems.
1.5 Overview of Thesis

This thesis consists of six chapters. General information on geometric transformations and the image formation is provided in Chapter 2. Euclidean, affine and projective transformations are introduced with an emphasis on projective transformations. The image formation process using the pinhole camera model is also discussed here. In addition, the epipolar geometry, describing relationships between two images, is presented. The third chapter gives an overview of the image segmentation problem. A mathematical criteria for image segmentation is introduced and several categories of segmentation techniques are discussed. Chapter 4 introduces the point matching problem and discusses traditional correlation-based techniques. The inherent computation problem is discussed motivating the planar matching technique presented in this thesis. The use of projective geometry to simultaneously match and grow the image is discussed in Chapter 5. The function for mapping of a plane from the left image to the right image is derived and applied in a planar region growing algorithm. Results are also presented in this chapter. Finally Chapter 6 offers some concluding remarks and discusses future improvements for this algorithm.
Chapter 2

Image Formation and Image Processing Basics

The field of vision relies heavily on geometric transformations. Three types of transformations are introduced in this chapter: Euclidean, affine and projective. Projective transformations are used extensively in this thesis. Thus, they are presented in more detail.

The aforementioned geometry is used when modeling the image formation process. A commonly used model is the pinhole camera model. Image formation based on this model is presented in this chapter.

Many vision applications require multiple perspectives of the same scene. The epipolar geometry describes some basic relationships between these multiple images. This chapter introduces the epipolar geometry and the epipolar constraint.

2.1 Geometric Transformations in Image Formation

Modeling the image formation process relies heavily on geometry. An image formation model is a collection of geometric transformations mapping 3D scene points to 2D image pixels. Three types of geometric transformations are common in the image formation process.

1. Euclidean
2. Affine
3. Projective

This section will provide a brief overview of the differences between these transformation classes.
Euclidean Transformations

Euclidean geometry is used to describe real-world geometry [41]. Euclidean transformations have many invariants. These transformations only change the location and orientation of objects in space. Shape, distances and angles are all preserved by Euclidean transformations. Think of a real-world object, such as the box in figure 2.1. A Euclidean transformation may change the position and orientation of the box; however, the dimensions of the box remain the same.

\[
\frac{|AB|}{|AC|} = \frac{|A'B'|}{|A'C'|}
\]  

(2.1)

where \(| \cdot |\) represents the typical Euclidean distance measure.

Figure 2.1: Box Before and After a Euclidean Transformation

Affine Transformations

Affine transformations are less restrictive than Euclidean transformations [26]. Distances and angles are no longer preserved. However, many geometric properties are still preserved. For example, parallel lines remain parallel, and lines that intersect still intersect after applying the affine transformation. In addition ratio measures of distance are preserved.

Consider the affine transformation of a box in figure 2.2. The parallel sides of the box remain parallel, but the box is clearly distorted. The following line ratio measure is preserved by this transformation.
Note that Euclidean transformations also preserve these ratios. In fact, the Euclidean transformations are a subset of affine transformations.

![Figure 2.2: Box Before and After an Affine Transformation](image)

**Projective Transformations**

Projective transformations have fewer invariants than affine transformations. Only points, lines and planes are preserved by projective transformations. Cross-ratios, or ratios of ratios, are preserved by projective transformations.

Consider the projective transformation of the box in figure 2.3. Sides of the box that were previously parallel are no longer parallel after the projective transformation. This is apparent in the sides along the depth of the box. The following line cross-ratio measure is preserved by the projective transformation

\[
\frac{|AC| \times |BD|}{|AD| \times |BC|}
\]  

(2.2)

Both affine and Euclidean transformations are a subset of projective transformations. Thus, cross-ratios are also preserved by these two classes of transformations.

More details on Projective Transformations are presented in the next section.
2.2 Projective Transformations

A projective transformation, also known as a *collineation*, is a linear transformation of points in homogeneous coordinates. The following function defines the mapping from a projective space $\mathbb{P}^n$ to the projective space $\mathbb{P}^m$

$$ Q \sim Wp $$

(2.3)

where $W$ is the $(m + 1) \times (n + 1)$ non-singular projective matrix and $p$ is the point to which the transformation is being applied.

In projective space equality is defined up to a scale factor. This equality is represented by the following equation.

$$ p \sim p' \quad \text{iff } \exists \lambda \neq 0 \text{ such that } p = \lambda p' $$

(2.4)

where the symbol $\sim$ stands for equality up to a scale factor.

A projective basis for $\mathbb{P}^n$ is a collection of $n + 2$ points such that no $n + 1$ of them are linearly dependent. For example, the standard basis for $\mathbb{P}^2$ is:
\[ p_1 = (0, 0, 1)^T \]
\[ p_2 = (1, 0, 0)^T \]
\[ p_3 = (0, 1, 0)^T \]
\[ p_4 = (1, 1, 1)^T \]  

Any projective transformation that provides a \( \mathbb{P}^n \to \mathbb{P}^n \) mapping is known as a homography. Geometrically, a homography can be considered as a change of basis.

Only the properties of projective transformations used in this thesis are presented here. For a more thorough introduction to the use of projective geometry in vision see [48].

### 2.3 The Pinhole Camera Model

Before developing an algorithm to extract information from an image, it is important to describe how the image was formed. In computer vision and image processing this corresponds to a geometric image formation model. A good model should accurately describe how the objects in a scene are represented as the pixels of the image. A commonly used model in computer vision is the pinhole camera model. A distinguishing property of this model is the foreshortening property. This means that an object will appear smaller in the image as it is moved away from the camera.

![Figure 2.4: The Pinhole Camera Model](image)

Using the pinhole camera model, light reflected off an object is projected onto the image plane as in figure 2.3. A space point, \( P \), is projected along the ray passing through the optical point.
O. The intersection of this ray with the image plane determines the coloring of the pixel \( p \). This process is defined by the following three step transformation.

1. A 3D Euclidean transformation that transforms the object from the world coordinate system to the coordinate system of the camera.

2. A 3D-2D projective Transformation that defines how the objects are projected onto the image plane. After this transformation pixels are in a normalized coordinate system.

3. A 2D-2D affine transformation that transforms the points from the normalized coordinate system to pixel coordinates.

Out of the three transformations used the projective transformation preserves the least amount of information. Thus, the whole process can be considered a projective mapping. Only points, lines and planes are guaranteed to be preserved during the image formation process. The fact that planes are preserved in the image formation process is a key to the matching and segmentation technique presented in this thesis.

2.4 Epipolar Geometry and the Epipolar Constraint

Many vision applications require multiple perspectives of the same scene. In particular, many vision applications use a stereo pair of images. The epipolar geometry describes the relationships between a stereo pair of images. These relationships are used in many vision applications, such as matching and 3D scene reconstruction.

Consider the two image planes shown in figure 2.5. In this image space point \( P \) is projected to \( p \) and \( p' \) in the first and second images respectively. The point \( p' \) has the special property that is lies on the line that is the projection of the ray \( PO \) on the second image, and similarly for \( p \) on the line that is the projection of the ray \( PO' \) in the first image. These projected lines are known as epipolar lines and they bind the projection of a point in one image plane to a line in another [26]. This geometric property is captured in a \( 3 \times 3 \) matrix known as the fundamental matrix, denoted \( F \). Consider the fundamental matrix, \( F \), that defines the epipolar geometry from the first image to the second image. The expression \( Fp = (A, B, C) \) yields the coefficients of the line containing \( p' \) in the second image. Using the fact that \( p' \) must lie on the line \( Fp \) the following relationship, known as the epipolar constraint, is derived.

\[
p'^T Fp = 0 \tag{2.6}
\]
Figure 2.5: Epipolar Geometry

Each of the epipolar lines for an image intersect at a unique point known as the epipolar point. This can be seen in figure 2.5 where $e$ and $e'$ are the epipolar points of the first and second images respectively.

The epipolar geometry can be used to simplify many stereo vision problems, provided $F$ is known. Fortunately, there are several methods for calculating $F$, based on as little as eight matched points. Methods for robustly calculating the fundamental matrix are given in [6] and [35]. From this point on the calculation of the epipolar geometry will be assumed as a pre-processing step.

2.5 Summary

This chapter discussed the basic background required for this thesis. The Euclidean, affine and projective geometries were introduced. The projective transformation and several of its properties were also introduced. These classes of geometry were then used to describe the pinhole camera model. This camera model is commonly used to describe the image formation process.

Finally, the epipolar geometry was introduced. This geometry describes a set of properties of stereo image pairs. In particular, the epipolar constraint binds the correspondence of point $p$ in the left image to a line in the right image.
Chapter 3

Overview of Image Segmentation

Image segmentation is a very difficult problem. However, segmented images are often required for applications such as augmented reality, robot path planning, medicine and token tracking. As such, there has been much research done on this problem. This chapter introduces the basic theory of image segmentation and discusses some of the techniques for solving this problem.

3.1 Definition and Classifications of Segmentation

An image segmentation is the grouping of image pixels based on some homogeneity criteria. Depending on the technique the criteria may be different. Some methods are based on pixel statistics such as intensity. Other methods try to enclose regions within a boundary. Since different methods rely on different criteria, the results vary from one method to another.

Consider what happens if you ask a person to segment an image of a scene into regions. Most of the time the person will easily segment the image into similar regions. If you ask many people to do this you will notice that the results will differ, but in general most will be acceptable. This shows two problems of segmentation. First, a precise definition of an image segmentation is needed. Second, there must be some criteria for evaluating the quality of a segmentation.

One way to define an image segmentation is as follows [37, 54]. Consider the binary predicate, $P(X)$, that can assume the values TRUE or FALSE. A segmentation of grid $X$ into $X_1, X_2, ..., X_n$ must satisfy the following criteria:

1. $\bigcup_{i=1}^{N} X_i = X$

2. $X_i, i = 1, 2, ..., n$ is connected

3. $P(X_i) = TRUE$ for all $i = 1, 2, ..., n$
4. \( P(X_i \cup X_j) = FALSE \) for all \( i \neq j \)

The first criterion says that the collection of all segments in the image must make up the whole image. The second ensures that each region’s points are reachable from any other point in that region. The binary predicate in the last two criteria are used to ensure that regions are as big as possible. If not, two regions \( i \) and \( j \) could be merged into one.

Many segmentation techniques implicitly use this definition, but this is only one such formalization. Not all segmentation techniques use these criteria. For example, techniques such as thresholding, to be discussed later, produce a segmentation based on homogeneity, but do not guarantee connected regions.

Different segmentation techniques perform in different ways. In fact, there is no universal criteria that all techniques follow. Thus, it is useful to group methods into categories. The following section describes some different classification schemes and adopts one for the purpose of this survey.

### 3.2 Algorithm Classifications

Image segmentation is a very difficult, yet very important, problem in image analysis. As a result, many papers describing many methods have been published[28][34][64][47]. It is thus useful to classify such methods based on their common properties. The number of methods in existence makes classification very difficult. There is no universal classification system. Some of the commonly-used algorithm classifications are presented here.

Healy et al. [36] classify techniques as global, region homogeneity and boundary-finding. The global techniques are those that assign a pixel’s membership based on information from the entire image. Region homogeneity groups pixels based on some criteria of similarity in a neighborhood around the pixel. Finally, boundary-finding techniques locate the region boundaries of the image. Fu and Mui [28] and Gonzalez [32] use the same classification, but refer to global techniques under the more specific name of thresholding.

In his paper comparing different methods, Zhang [79] adopted a classification scheme that enumerates algorithms as either boundary-finding or region detection techniques. The author further classifies them as parallel or sequential. A parallel method contains modules that do not need input from other modules, while the modules in a sequential segmentation algorithm may depend on input from another module. The result is the following four classifications:

- parallel boundary-based techniques
CHAPTER 3. OVERVIEW OF IMAGE SEGMENTATION

- sequential boundary-based techniques
- parallel region based techniques
- sequential region based techniques

Haralick and Shapiro [34] differentiate clustering from segmentation by the mathematical space they are performed in. In clustering, the grouping is done in measurement space, while segmentation performs in the spatial domain. They use this idea to classify segmentation algorithms into the following six categories:

- measurement space guided spatial clustering
- single linkage region growing schemes
- hybrid linkage growing schemes
- centroid linkage region growing schemes
- spatial clustering schemes
- split and merge schemes

The above classifications use different properties of segmentation at varying levels of detail. To keep it simple, this survey will use four categories based on the classifications of Healy, Gonzalez and Fu and Mui. The fourth category corresponds to hybrids between the three natural categories. The result is the following classification.

- Clustering and Thresholding (Global)
- Boundary-Detection Techniques
- Region Detection Techniques
- Integrated Hybrid Techniques

Since there are many papers on segmentation, it is difficult to cover all of them in one survey. The goal of this survey will be to introduce some techniques from each category. This will illustrate the properties of each class. It also serves to introduce some of the work within the classes.
3.3 Global Techniques

Global segmentation techniques are those that exploit the global properties of an image. When deciding which region a pixel belongs to, these algorithms use properties based on all the pixels in the image, not just the surrounding neighborhood. The two most popular global methods are thresholding and clustering.

Thresholding

Thresholding, also known as binarization, is an image segmentation technique that is used to separate the foreground objects of an image from the background. The problem is formalized as follows [64]. Consider the image function \( f(x, y) \) with gray levels \( G = 0, 1, \ldots, I - 1 \) where \( f \) is an \( N \times N \) function and \( (x, y) \) are the spatial coordinates of the pixel. Now let \( t \subset G \) be a threshold and \( B = \{b_0, b_1\} \) be a pair of binary gray levels with \( b_0, b_1 \subset G \). The result of the thresholding is defined by the following equation:

\[
    f_s(x, y) = \begin{cases} 
    b_0 & \text{if } f(x, y) < t \\
    b_1 & \text{if } f(x, y) \geq t 
    \end{cases} 
\] (3.1)

The goal of a thresholding algorithm is to find the optimal threshold \( t^* \) based on some criteria.

One of the earliest researchers to examine thresholding for segmentation was Doyle [22]. Doyle introduced a technique known as the p-tile method. In this method, it is assumed that the percentage of image area occupied by objects is known. The threshold is then defined as the largest number that maps at least \( (100 - p)\% \) of the pixels into objects in the thresholded image. The problem with this approach is that the percentage of area occupied by objects is seldomly known. Thus, a method that can automatically choose a threshold is more desirable. One common approach is to analyze the histogram of an image. The histogram is a diagram that represents the distribution of pixel intensities. If there is a clear difference between the background and objects of the image (i.e., there exists a clear separation in the range of pixel intensities) then the histogram will be bi-modal. If this is the case, the threshold may be chosen as the valley between the two modes in the histogram [56]. Often, images do not exhibit this nice bi-modal property, so more sophisticated methods for choosing the threshold are necessary. For example [63] describes a method that looks for “shoulder” points by analyzing the concavity of the histogram.

One method [78] uses points with high gradient values as aids in choosing a good threshold. Points with high gradient values will be approximations of the threshold. This is intuitive because the threshold represents the intensity transition between object and background points, and high
gradient points are points on a transition between the object and the background.

Another set of approaches are based on information theory [57, 58, 39, 31]. These methods consider the gray level histogram of an image as an l-symbol source and use information theory to choose a threshold based on some image statistics, known as entropies.

Rosenfeld and Smith [61] introduced a thresholding technique that uses relaxation. Probabilities for the classification of each pixel as light or dark are given. These probabilities are iteratively updated based on the value of their neighbors. After the iterations reach a steady state the threshold can easily be chosen as the one that agrees with the probabilities.

Multithresholding Techniques

The problem with the thresholding techniques previously discussed is that they only separate the image into background and foreground regions. Oftentimes, it is desired to have a more detailed segmentation of the image. Multi-thresholding techniques find more than one threshold. A simple example would be the case where the histogram has \( n \) clear modes. Sometimes multi-thresholding techniques are natural extensions of the single thresholding techniques. For example, in the mode method [56] \( n - 1 \) thresholds can be chosen; one from each valley between the nodes.

One method designed specifically for multi-thresholding is the amplitude segmentation method due to Boukharoula et al. [7]. This method defines a curvature function based on the cumulative distribution function of the image. The thresholds and gray levels of each region are then determined by identifying the zeros in this function.

Wang and Haralick [72] have defined a recursive multi-thresholding technique. In this technique, pixels are first identified as edge or non-edge pixels. This process is then recursively iterated using the histogram for each class.

Other Global Techniques

One other popular global segmentation technique is clustering. One popular clustering method [33] introduced by Hansen and Elliott views the image pixels as a lattice that represents a Markov Random Field (MRF). This method is an image restoration method that classifies pixels into clusters based on the statistics of \( n \) neighboring pixels. The goal of achieving the best segmentation is transformed into finding the maximum a posteriori (MAP) probability. Hansen and Elliott's solution to this problem is an iterated algorithm using a recursive dynamic programming method. Geman and Geman [30] have applied a different algorithm to this problem. Random changes are made in each neighborhood based on the local conditional probability distribution. These changes are
based on a temperature parameter $T$. High temperatures add a high coupling between pixels in a neighborhood, while low temperatures lead to a low coupling. This corresponds to greedy (non-random) changes for $t = 0$ and purely random changes for $t = \infty$. A simulated annealing process is used to create a convergent iterating algorithm. Geiger and Girosi [29] developed an iterative and parallelizable model. Using a Weak Membrane model they obtained three field equations that can be solved by con-jointly using gradient descent and deterministic annealing. Bouman and Shapiro [8] have presented a method that uses a novel multiscale random field (MSRF) replacing the MAP with an SMAP derived from the novel estimation criteria. A non-iterative technique based on both fine to coarse and coarse to fine scale linking is developed. This technique performs in $O(mn)$ time where $m$ is the number of clusters and $n$ is the number of pixels.

Another clustering approach uses neural networks. Vilarino et al. [73] use a Hopfield neural network whose goal is to minimize an objective function. This objective function is based on image properties such as intensity and texture. The neural network will segment the image by finding the clusters of points which minimize this objective function.

Another global technique uses normalized cuts [67]. This methods sets up the image as a directed graph. The graph is then partitioned into cuts that define segments in the image. Ishikawa and Geiger [38] detect junctions in the image and then apply the cut method to obtain a better segmentation.

### 3.4 Boundary-Detection Techniques

Another method for segmenting an image is to search for the boundaries that enclose the object. If closed boundaries can be identified then the image has been segmented. Boundaries in an image correspond to a transition between objects in the image, or the object and the background. Edge detection, another low-level operation, is often used to search for these transitions. Since edge detection usually fails to find closed contours, it is not useful for segmentation by itself. The information found by the process of edge detection is still useful and is often used as a pre-processing step for segmentation. It is thus useful to provide a brief summary of classical edge detection.

#### Edge Detection

Edge detection was one of the first image-analysis problems studied. The problem consists of finding areas with a large difference in intensity between adjacent pixels. These areas of large intensity change correspond to edges in the image, and in many cases boundaries between regions. The idea
of large changes in intensity corresponds to derivatives in calculus. The problem of edge detection thus became developing methods which would perform an image derivative. One of the first edge detectors was the Roberts Cross [60] operator. In this method, Roberts applied an image mask to each pixel. This mask estimated the sum of the magnitude of the two directional derivatives $D_{45} f$ and $D_{135} f$. After this, Sobel and Prewitt [56] each introduced a mask that estimates the gradient vector. Another common derivative estimation used is the Laplacian.

The problem is that these methods are ad hoc. They simply estimate the value of a derivative and have been shown to work reasonably well through experimentation. The performance of these image masks was tested as an afterthought. In the landmark paper [12], Canny introduced three mathematical criteria the ultimate edge detection algorithm should meet: good detection, localization and uniqueness of response. The first criteria states that the number of false edges detected should be as low as possible, while the number of true edges detected should be as high as possible. Localization states that the location of marked edges should be as close to the true edge as possible. The last criteria, uniqueness of response, is used to ensure that only one edge is marked for each true edge point. Canny derived an edge detector by maximizing the first two criteria under the constraint of the third criterion. Spacek transformed Canny's criteria into a form where all three criteria are maximized. His paper [70] details this new solution. Deriche [20, 21] discovered a number of elegant recursive solutions using Canny's criteria. In Canny's solution finite image filters applied to some neighborhood of the image were used. Deriche used infinite image filters, in combination with the Z transform, to create a number of recursive solutions.

One boost for the use of edge information has been presented by Elder [24]. In this paper Elder claims that edges are enough to code the information of an image. This means that edge information is definitely an asset to the process of segmentation.

This has been a brief overview of the area of edge detection. For a good introduction to the theory of edge detection see [26, 46]. A more practical introduction that introduces and evaluates different methods can be found in [80]

**Edge Linking**

An obvious way to segment images is to try and link the broken edge chains to form closed contours. Doing this produces a collection of closed contours, each encompassing a region in the image. This process is known as *edge linking*.

Early attempts at edge linking tried to group edges to form larger edge chains, but failed to yield perfectly closed contours. Grouping with multi-scale smoothness criteria has been used in [23, 44, 65]. Some sequential methods that can track contours in a Bayesian framework are
CHAPTER 3. OVERVIEW OF IMAGE SEGMENTATION

presented in [17]. Parallel methods that measure saliency are introduced in [66].

Based on this earlier work, methods that attempt to detect closed contours have been developed. Alter [1] studied the application of shortest path algorithms for edge linking. Elder and Zucker [25] use a similar idea, but define specific criteria. Image contours are represented locally as a set of tangent vectors, augmented by image intensity estimates. A Bayesian approach that determines the probability of a tangent pair being contiguous is developed by considering properties of the extended tangents. The six most probable contiguous contours for each tangent are considered and a weighted digraph is formed. The job is now minimizing a path through the digraph using a method such as Dijkstra's algorithm.

There are many other segmentation techniques that use edge information, but do not try to close the detected edge chains. Some of these methods are as follows.

Active Contour Models

Another way to find the boundaries of a region is to use Active Contour Models. These techniques are used to deform splines around the region boundary. To obtain closed regions an edge linking technique can be used on the spline.

The use of active contour models for image segmentation was popularized by Kass et al.[40]. Their model, referred to as snakes, use an energy minimizing spline that is deformed by constraint forces. In their paper they describe three forces:

• Internal forces modeling the bending of the contour

• Image forces representing information such as edges

• Constraint forces to provide any additional external forces

Lobregt and Vergever[43] have improved this model by designing a dynamic snake model for 2D images. This model reduced vertex clustering and shrinking problems observed in the original model. The previously mentioned techniques require the initial specification of an entire spline. In Neuenschwander, et. al., [50, 51] a technique requiring the specification of only the two end points is introduced. The boundary is then detected from the end points to the center, mimicking the behavior of closing a zip-lock bag.

One drawback of the above mentioned methods is that they must be supplied with a good initial estimation of the contour. In most cases, this amounts to a manual step where the user places the initial contours in the image. To be useful in many applications the method is required to be fully automatic. Cohen's [16] balloon model is a first step toward automatic segmentation
using deformable models. An additional force, similar to that of an inflating balloon, is used to strengthen the model. This force allows the initial boundary approximation to be further away from the actual boundary. This additional force also keeps the deformable contour from being attracted by weaker, superfluous edges.

An excellent survey of deformable image contour models is provided in [47].

Other Boundary-Detection Techniques

Ma and Manjunath introduced the idea of edge flow [45]. For each pixel, a measure known as the edge energy is calculated. This measure gives the strength of the response to the edge filter and the direction of the edge. This flow is propagated in the direction of the edge energy. This flow should point to an edge, thus an edge occurs at an equilibrium point, or said differently points were the edge flows of two adjacent pixels flow into each other.

Cox, et. al., [17] present a method where multiple hypotheses for the contour are tracked. In this method the hypotheses are kept in a tree with the nodes representing the probability that the given edge will be part of the contour. Each trace of the tree from the root to a leaf represents a unique contour. From these hypotheses a prediction of how the contour will progress is made.

3.5 Region Detection

Region detection methods group pixels into regions based on some criteria of homogeneity amongst neighboring pixels. Thus, each pixel in a region shares common properties with other pixels in the region.

Early Region Detection

One of the earliest methods of region detection is region growing. In this method $n$ seeds are chosen, growing at most $n$ regions in the image. Growth occurs by iteratively expanding the region to encompass adjacent pixels. The region keeps growing until no pixels neighboring the current region are similar enough to be added. An example of similarity criteria that may be used is a threshold between the seed and the pixel in question. The problem with this method is that it is not stable. The choice of initial seeds very much affects the final segmentation. If the seeds are very badly chosen then the final segmentation will be poor; however, good initial seeds can produce good segmentations.

$^1$Note that this threshold is not the same as the global threshold technique introduced earlier. It is based on the difference of a single pair of pixels.
CHAPTER 3. OVERVIEW OF IMAGE SEGMENTATION

Another basic method is the split and merge method. This technique starts with an arbitrary segmentation. Based on this segmentation, splitting and merging is iteratively performed. Once an iteration occurs where no splitting or merging takes place the image segmentation is finished. The splitting step ensures that no region is larger than it should be, while the merging step ensures that there are no small adjacent regions with the same properties.

Advanced Region Segmentation Techniques

The two methods described above have the advantage that they are easy to implement. Both work well on simple images, but poorly on real images. Thus, more sophisticated methods for region segmentation exist. It is interesting to discuss these methods since many segmentation methods use these techniques within the whole.

One division of the region segmentation algorithms uses the cooccurrence matrix. This matrix represents the frequency of change between two intensities in adjacent pixels. The element \((i,j)\) of this matrix denotes how often the transition from intensity \(i\) to intensity \(j\) occurs when moving to an adjacent pixel. Weszka and Rosenfeld [77] define a busyness measure based on this matrix. The measure is evaluated for each gray level \(s\) in the image, and the gray level \(s'\) that minimizes the busyness measure provides a threshold (similar to multi-thresholds). \(^2\)

Deravi and Pal [19] have developed a co-occurrence matrix method that determines the conditional probability of transition from one region to another. This probability is denoted as \(P_c(s)\). The idea they use is as follows. The smaller the value of \(P_c(s)\), the less the chance of a transition in the boundary between two possible regions, thus by minimizing the value of \(P_c(s)\), a good value for the threshold is obtained. To extend this idea to multithresholding, the set of minima of \(P_c(s)\) over all gray levels \(s\) is chosen as the threshold. In [53], Pal and Pal introduce the idea of contrast between regions as well as intensity change. Their co-occurrence matrix measure is thus based on both contrast and intensity. Thresholds are found by increasing the gray level and marking thresholds at maximums in the contrast measure. A merging technique is then used to eliminate single-intensity-valued segments, provided they fit nicely with another segment. A second algorithm incorporates size into the algorithm eliminating the merging phase described above.

Another group of region detection methods are based on a tree structure (called a hyperstack). Each level of the tree corresponds to a level of detail, called pixel in 2D and voxel in 3D, where the root would represent the whole image and the leaves would represent smaller regions. These methods work either fine to coarse or coarse to fine. The fine to coarse has four steps [75].

\(^2\)Once again this threshold is based on a non-global quantity since the matrix is calculated from the adjacent pixels.
CHAPTER 3. OVERVIEW OF IMAGE SEGMENTATION

1. blurring

2. linking

3. root labeling

4. downward projection

In the first stage blurring filters are applied to remove detail in the image, then child-parent linkages are established between pixels at different levels. During the root labeling phase pixels at the top most level, and those with weak parent linkages are marked as roots. After this, the downward projection step traverses the tree, labeling each leaf as a child. Traditional linkage criteria [74, 18] assign exactly one parent to each child; however, probabilistic methods [3] can assign multiple parents to each child, where each parent has a given probability of being the true parent of that child.

The coarse to fine method relies on hierarchical data clustering [76, 69, 68]. In this method, an initial segmentation, defining the root nodes, is chosen. These root nodes are then merged to form bigger segments. The simple way to do this is to merge the regions based upon a simple threshold, usually gray level difference and minimum area, but since real images contain noise, this is not the best method in practice. In [3], the maximum number of regions is specified by the user. This method then applies an iterative process of node re-linking and grouping of eight neighboring roots. When the number of regions is equal or below the user specified number the segmentation is complete. Vincken et al. [75] proposed a hierarchical method that uses statistics. Instead of keeping one parent-child linkage, a child may be linked to multiple parents with a probability assigned to each. Some number $n$ of the strongest linkages are kept. Once the roots are reached, the tree is parsed choosing the segmentation that has the strongest statistics. This prevents the segmentation tree from choosing the wrong parent in the case where the proper parent may not have the strongest linkage to the child at that point.

In [4], a merging criteria is used to minimize a step-wise criteria $C_{ij}$. This criteria determines the wellness of the merging of two adjacent regions. The author presents two methods for doing this. The first one simply minimizes a function, effectively maximizing the fitness of the segmentation. The second considers the minimization of two types of error hypothesis, rejecting $H_o$ when $H_o$ is true, and accepting $H_o$ when $H_o$ is false.
3.6 Integrated Hybrid Techniques

So far, this paper has introduced segmentation techniques from three different categories. These categories have different advantages and disadvantages. For example, region oriented techniques use image statistics over a neighborhood, so they are more robust to noise. A weak point of these techniques is localization. Conversely, boundary-finding techniques that use edge information tend to have better localization, but do not handle noise as well. Therefore some researchers have attempted to fuse methods in different categories to try and achieve the benefits of both.

In 1990, Pavlidis and Liow [55] presented a method for integrating region growing and edge detection. Two steps are used to find the best possible segmentation. First, boundary elimination is performed. This eliminates boundaries which do not correspond to an actual edge. After this step contour modification is applied. In contour modification the boundary is moved to correspond best with the actual edge.

Chehikian [14] has introduced another method that integrates region and boundary-detection techniques. In this paper, region points are defined on a lattice equal to that of the image, with contours located between image pixels. This model is transformed into a lattice twice the size of the image, where the even lattice points correspond to the region pixels and the rest to contour pixels. A B-Spline interpolation operator is defined. The operator locates the boundaries between pixels, that are in turn used in an algorithm that combines this information with region information.

Bozma and Duncan [11, 9, 10] introduce the notion of two segmentation modules that compete in an $n$ player non-cooperative game. Each player then tries to minimize his objective function. Using game theory, this corresponds to the problem of finding the Nash equilibrium for the system. Chakraborty and Duncan [13] used this frame to formulate a non-zero sum (any gain is not at other players expense) game that fuses region based and gradient based segmentation methods. The benefit of this mechanism is that any segmentation technique that can be modeled with the appropriate objective function can be used.

3.7 Conclusion

Image segmentation is an integral part of image analysis and computer vision. By itself, image segmentation partitions the image into regions of similarity. As an image analysis primitive, segmentation is used as part of higher-level operations. Examples of higher-level operations that use computer vision are medical imaging, lip-reading and image retrieval.

The difficulty in identifying a unique definition imposing all segmentation techniques to provide
the same output was discussed. A criteria defining a segmentation was presented. Numerous classification schemes for segmentation algorithms were presented. For the purpose of this survey, four categories were chosen: global techniques, boundary-detection, region detection and hybrid techniques. Some common examples from each category have been described.
Chapter 4

Correlation-Based Matching Techniques

As humans, we rely on two perspectives of a scene to perceive 3D data. Similarly, many computer vision applications use two or more images of a scene to recover similar information. Multiple images of the same scene provide a rich set of information not available from single images, including 3D scene structure and information about occluded objects.

In their raw form, a pair of images provide little data about the scene. To make use of the stereo data, it is necessary to understand how the images are related to each other. In the absence of high-level data, both images are merely a collection of pixels. Thus, the basic relationship between a pair of images is the correspondence between their pixels. The process of finding this correspondence is called point matching. This chapter introduces the problem of point matching and examines several types of techniques.

4.1 Definition of Point Matching

The problem of point matching is to find the correspondence of pixels between two images, \( I \) and \( I' \). To illustrate this consider, figure 4.1. In this stereo pair, several matches are marked with crosses. Take, for example, one of the marked window corners in the left image. The corresponding matching pixel is also marked in the right image.

Mathematically, matching can be viewed as the process of finding a function, \( M \), that takes as input the point \( p \) in \( I \) and maps it to \( p' \) in \( I' \):

\[
M(p) \mapsto p'
\]
This function is an $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ mapping where the domain is the possible pixel coordinates in $I$ and the range is the possible pixel coordinates in $I'$. An alternate view to matching is how the position of the pixel has moved between $I$ and $I'$. This is known the disparity of the pixel and the results are shown in a disparity map.

The class of a matching technique is categorized according to the scope of its solution. Sometimes it is only necessary to match interesting features, such as corners, junctions and lines. This is known as sparse matching. Conversely, dense matching is the matching of entire regions or the entire image. Computationally, sparse matching is a simpler problem than dense matching. As such, there exist numerous computationally feasible sparse matching techniques. This thesis deals mainly with dense matching.

4.2 Anatomy of a Matching Algorithm

Traditional matching algorithms typically solve for the matching function, $M(p) \rightarrow p'$, in an empirical manner. This involves a search to select a set of candidate matches and a measure to evaluate them. The goal of the search strategy is to employ heuristics to find a minimal set containing good candidate matches. This may include the application of constraints to the image or the use of a numerical approach to find the optimal candidate. Once the candidate set has been found a correlation function is used to evaluate the candidates. An overview of correlation is presented in the next section. This is followed by a discussion of some common search strategies.
4.3 Correlation Functions

Most traditional pixel matching techniques employ a pixel to pixel comparison strategy. Thus, it is necessary to have some way of evaluating the fitness of potential matches. The most common way is to use a correlation function.

A correlation function is a function that takes, as inputs, two points. A point \((x, y)\) in the image \(I\) and a candidate match \((x', y')\) in the image \(I'\). The output of this function is a measure indicating the fitness of \((x', y')\) as a match to \((x, y)\). This fitness measure is calculated by comparing a neighborhood around both \((x, y)\) and \((x', y')\). The neighborhoods around \((x, y)\) and the actual corresponding point \((x', y')\) should be very similar. Thus, the higher the similarity of these neighborhoods, the better the correlation score. There are many different correlation scores based on different statistics of the neighborhood. Some of the more common correlation measures are listed in table 4.1. Depending on the specific measure the goal may be to either minimize or maximize the correlation score.

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum of Squared Differences</td>
<td>[\text{SSD}((x, y), (x', y')) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} (I(x + i, y + j) - I'(x + i, y + j))^2]</td>
</tr>
<tr>
<td>Sum of Absolute Differences</td>
<td>[\text{SAD}((x, y), (x', y')) = \sum_{i=-n}^{n} \sum_{j=-m}^{m}</td>
</tr>
<tr>
<td>Cross Correlation</td>
<td>[\text{CC}((x, y), (x', y')) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} I(x + i, y + j) \cdot I'(x' + i, y' + j)]</td>
</tr>
<tr>
<td>Zero Mean Normalized Cross Correlation</td>
<td>[\text{ZNCC}((x, y), (x', y')) = \frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} (I(x + i, y + j) - \bar{I})(I'(x' + i, y' + j) - \bar{I}')}{\sqrt{\sum_{i=-n}^{n} \sum_{j=-m}^{m} (I(x + i, y + j) - \bar{I})^2} \cdot \sqrt{\sum_{i=-n}^{n} \sum_{j=-m}^{m} (I'(x' + i, y' + j) - \bar{I}')^2}}]</td>
</tr>
</tbody>
</table>

Table 4.1: Common Correlation Functions

4.4 Sparse Matching

Sparse matching matches feature points in an image. Since these feature points are easily distinguishable from other points, they are also easier to match. Sparse matching can be divided into two steps:

1. Identifying the interest points
2. Matching between these sets of points

Identifying interest points in an image involves finding interest points such as corners, junctions and dots. Figure 4.2 shows examples of different neighborhood configurations around an interest
point. A simple interest operator that identifies such neighborhoods is described by Moravec [49]. Here, the sums of the squares of differences is calculated for the horizontal, vertical and both diagonal directions. The value of the operator is the minimum of these previous values. For interest points, such as dots, junctions and corners, the variance will be high in every direction. Therefore, interest points can be identified where the value of this operator is high.

![Image](image.png)

Figure 4.2: Example Patterns for Interest Points in a 5 × 5 neighbourhood: a) dot, b) corner, c) junction

Once the interest points have been identified, a search strategy, such as those in the following section, is applied to the candidate set. Since the candidate set for sparse matching is much smaller than that for dense matching, typically several hundred, techniques that are not feasible for dense matching may be suitable for sparse matching.

### 4.5 Review of Existing Technique

As a result of the importance of point matching, the literature contains thousands of methods that attempt to solve this problem. However, this is a difficult problem and no universally accepted solution exists. This section will introduce some of the approaches to the point matching problem.

**Exhaustive Approach with the Epipolar Constraint**

Solving the point matching problem in an unconstrained manner would involve a total pixel to pixel comparison between both images. The computation required for this brute force approach is not feasible for even close to real-time solutions. Fortunately, the epipolar geometry can be used to reduce this problem from a 2D search to a 1D search.

The epipolar constraint constrains the point $p'$ in $I'$, corresponding to $p$ in $I$, to lie on a line. This leads to a basic approach for matching pixels, an exhaustive search on the epipolar line. Point $p$ in $I$ is compared to each candidate point, $p'$, lying on the epipolar line in $I'$. The match is the
CHAPTER 4. CORRELATION-BASED MATCHING TECHNIQUES

pair \((p, p')\) with the best correlation score.

Dense Matching Using Iterative Improvement Algorithms

These techniques start with a crude initial matching of the scene and iteratively improve upon it. The goal is to find a pixel mapping that represents a globally optimal solution. Based on the pixels neighbours, and a set of constraints, the pixel is matched in an iterative process.

Relaxation labelling is one such approach [62]. For each match pair \((p, p')\) in the image, the probability of the pair being the proper match is calculated. These probabilities are iteratively updated, based on the probabilities of neighboring pixels, until a steady state is achieved. For example, Christmas, et. al., [15] define a probabilistic method that integrates contextual information using a Bayesian framework. This contextual information is used as evidence to provide a formula that prescribes in a unified and consistent manner how unary relation measurements relating to single entities, binary relation measurements relating to pairs of objects should be brought to bear on the object labelling problem. Here a unary relation is a measure involving only a single pixel and a binary relation involves a pixel and its candidate match. For point matching, examples of binary relations are difference in grey-level intensity, change in relative position and correlation value.

Another popular iterative search approach is simulated annealing. This process mirrors the annealing system used in chemistry. To create perfect crystals, a system of molecules is gradually cooled, staying as close to the equilibrium as possible. The simulated annealing approach solves for the global maximum of a system. To find a globally optimal solution, the iterative process is allowed to take steps that decrease the fitness of the solution. This allows the system to move away from locally optimum solutions. A temperature variable, \(T\), dictates the probability that the system will try a suboptimal solution. As the number of iterations becomes higher, the temperature \(T\) lowers, thus lowering the chances that a fitness decreasing step will be taken. This approach was first applied to point matching by Barnard [2]. Starink [71] offered an improved solution attempting to minimize the effect of matching occluded points. The points to be matched are separated into two sets, one for the left image and one for the right. From the left set a point, \(p\), is taken and matched to a point in the right set, \(p'\), by choosing the best pair \((p, p')\) in terms of correlation measure. Each of these points is removed from their respective sets and the matching continues. Any remaining points are matched to a null point. New matches are assigned in three different ways.

1. Unmatched points are randomly assigned a new correspondence.

2. When the mapping of a point is \(n\) to \(1\) or \(1\) to \(n\), a new match for all but of the
points is randomly chosen

3. When the possibility of rearrangement is 1 then the rearrangement is randomly picked.

The progress of each iteration is measured by an energy function, $E$. Breaking a correspondence decreases $E$, while creating a new correspondence increases $E$ based on the fitness of the match. If $\Delta E$ is positive then the iteration is accepted. If $\Delta E$ is negative there is a probability it will be accepted based on the temperature, $T$. A further study on the fusion of relaxation and annealing techniques is presented in Rangarajan [59].

Iterative improvement techniques have good success in finding the globally optimal solution. The drawback is that the time required to reach this optimal solution is quite large. In fact, sometimes these techniques do not converge at all.

**Dynamic Programming**

Another approach to dense matching uses dynamic programming techniques. The general approach of dynamic programming is to organize an optimization problem to reduce redundant calculations. For the case of point matching, the ordering constraint is exploited. The ordering constraint states that if points have an ordering in the left image they must have the same ordering in the right image. If, for example, point $b$ is between $a$ and $c$ then $b'$ must be between points $a'$ and $c'$.

![Dynamic Programming Diagram](image)

**Figure 4.3: Dynamic Programming Performs Matching by Finding a Path Between $S$ and $T$**

The unidirectional graph in figure 4.3 illustrates the dynamic programming view of matching.
Matching data is stored in a double array indexed over \( i \) and \( j \). Here, \( i \) represents a pixel on the epipolar line in the left image, while \( j \) represents a possible match on the epipolar line in the right image; possible matches are denoted \( P(i,j) \). Matching involves finding the match, \( P(i,j) \), that represents the shortest path from \( S \) to \( T \). The solid or diagonal lines represent possible matches and the value \( F(i,j) \) is the cost, based on the fitness of the match pair \((i,j)\). The better the match, the lower the cost. The dotted lines allow for occlusions with a cost of \( C \). Although a technique such as Dijkstra's shortest path algorithm can be used, dynamic programming methods are more efficient due to the regular structure of the graph. Ohta [52] introduced a method to find this path in a 3D graph. Lloyd [42] presents a two stage method that produces a set of candidate matches between rows, then using the continuity constraint (i.e., the disparity must vary smoothly) chooses the best candidate in the columns.

Dynamic programming techniques decrease computation in the matching process. However, the amount of calculation required is still significant. This reduction of calculation comes at a cost. To eliminate redundant calculations, a large amount of data must be stored. For large images this may present a problem.

**Integer Programming**

Similar to dynamic programming, matching can be done by other, more general graph search approaches. These approaches represent the images as two sets of nodes, \( P \) and \( P' \). Each of the candidates \( p' \) in \( P' \) are attached to a point \( p \) in \( P \) as illustrated in figure 4.4. Weights on the edges between nodes represent the likeliness of a match between the two pixels. To uniquely match each point, the set of edges where the weights are maximum, and no two edges touch the same vertex is chosen. This is a common optimization problem, known as the maximum weighted matching problem. This problem can be solved using an integer programming technique. Fielding and Kam [27] use the "Hungarian Method" providing a polynomial solution to matching.

Integer programming solutions are an effective approach to matching as an optimization problem. However, if the candidate matches are not minimized the computation required is still large.

**4.6 Computational Expense of Correlation Measures**

The previous section introduced several of many approaches for the dense matching of points. Each solution differs in the specific search strategy implemented; however, to evaluate candidate matches, each method relies on a correlation measure.

The sample correlation measures presented in table 4.1 show that correlation measures vary
in complexity. Measures such as SSD and SAD are simple to compute, but may not be robust. Measures like ZNCC are more robust, but require more computation.

As shown in the about figure 4.2 the calculation of correlation measures is computationally expensive. There are simpler correlation measures, such as SSD and SAD, but measures such as NCC and ZNCC are more robust and thus more common.

4.7 Analysis of Dense Matching Using the Epipolar Constraint

To analyze the computation required by correlation functions, consider a search along the epipolar line. Furthermore, consider the case where the search is restricted to an $n$ pixel interval in the right image. A complete dense matching will still require $n \times l \times w$ calculations of the correlation measure. Think of a specific example of a $500 \times 500$ image. A disparity of at least thirty pixels between images would not be uncommon. So for each point $p$, thirty pixels along the epipolar line are chosen as candidate matches. To match all pixels in this image would require the calculation of 7.5 million correlation measures! Even if the number of candidate pixels for each point $p$ is reduced to five the correlations necessary would be 1.25 million. An approach to matching other than an empirical search is necessary.
<table>
<thead>
<tr>
<th>Correlation Measure</th>
<th>Operations Calculation</th>
</tr>
</thead>
</table>
| SSD                 | $l \times w$ Subtractions  
                     | $l \times w$ Additions  
                     | $l \times w$ Squares  |
| SAD                 | $l \times w$ Subtractions  
                     | $l \times w$ Additions  
                     | $l \times w$ Subtractions  
                     | $l \times w$ Absolute Values  |
| CC                  | $l \times w$ Multiplications  
                     | $l \times w$ Additions  |
| ZNCC                | $3 \times l \times w$ Additions  
                     | $3 \times l \times w$ Subtractions  
                     | $2 \times l \times w$ Squares  
                     | $l \times w + 1$ Multiplications  
                     | 2 Window Averages  
                     | 1 Square Root  
                     | 1 Division  |

Table 4.2: Mathematical Operations Performed for Different Correlation Measures
4.8 Conclusion

Matching is the process of finding a function \( M(x, y) \) that maps a pixel \((x, y)\) in \( I \) to its corresponding pixel \((x', y')\) in the image \( I' \). The matching of pixels amongst images is a critical operation for many computer vision operations, including synthetic view synthesis, object recognition and augmented reality.

There are two types of matching, sparse and dense matching. Sparse matching is concerned only with identifying and matching interest points. These points include corners, junctions and dots. This problem is relatively simple and many fast and robust methods exist for this purpose. Dense matching involves matching each image pixel. This problem is much more difficult. Search methods, such as dynamic programming, simulated annealing and integer programming, aim to reduce the size of the candidate set, but the calculation of correlation methods is computationally intense and not feasible for real-time dense matching. Improvements must be made when possible.

The next chapter introduces a technique that uses a closed form matching function for planar regions.
Chapter 5

Matching and Segmentation using Projective Geometry

Pixel matching is critical to the success of many computer vision applications. However, as shown in the last chapter, conventional correlation-based methods are not computationally feasible for even close to real-time applications. In addition, the correlation based methods are inaccurate when matching homogeneous regions. Methods to improve the speed and accuracy of dense matching are necessary. Another critical operation to computer vision is region segmentation. This is a first step toward organizing the image into its composite objects. This chapter introduces a method that quickly and simultaneously performs both dense matching and region segmentation for planar regions. First, an algorithm for calculating a projective mapping between a plane in two images is presented. Using this mapping, a region growing technique that both matches and segments the image is introduced. Finally, results comparing this method to the popular correlation-based epipolar line search are presented.

5.1 Overcoming the Empirical Nature of Conventional Techniques

Most traditional pixel matching techniques offer a solution based on two general steps:

1. A search strategy for finding a set of candidate matches

2. An evaluation step that calculates the fitness of each candidate match
CHAPTER 5. MATCHING AND SEGMENTATION USING PROJECTIVE GEOMETRY

This type of approach solves for the matching function, \( M(x, y) \), in an empirical manner. To match a single point, numerous candidate matches are evaluated with a correlation function. Even if the search process is intelligent enough to produce small candidate sets, the calculation of correlation functions is still expensive.

The ideal solution to the matching problem would be one that gives a closed-form function, mapping the entire image \( I \) to \( I' \). This function would map point \( p \) in the image \( I \) to \( p' \) in the image \( I' \). Unfortunately, it is impossible to find such a function. However, it is possible to find a function that maps a plane in image \( I \) to the corresponding plane in \( I' \).

5.2 Plane Matching

Recall that an image formed by the pinhole camera model is the result of a projective mapping. Thus, a closed form function mapping \( I \) to \( I' \) would be a \( \mathbb{P}^2 \to \mathbb{P}^2 \) homography. Since this mapping is between projective spaces, only certain properties are invariant, including points, lines and planes. The plane preserving property of projective transformations is used in this thesis to find a matrix function mapping planar surfaces from \( I \) to \( I' \). For the remainder of this thesis, the mapping used for matching planes will be referred to as the plane homography.

To make use of this geometric property, the scene must be abundant in planes. This is true of many scenes, especially in a man-made environment. Tables, desks, buildings and many other common objects are composed largely of planes. Figure 5.1 is an example of a highly planar scene. By matching these planes quickly and efficiently, the number of pixels needed to be matched by traditional methods is considerably reduced, improving the overall matching performance.

5.3 Calculating the Plane Homography

The plane homography maps points from \( I \) to \( I' \) in the following way:

\[
p' \sim Hp
\]

(5.1)

where \( H \) is a \( 3 \times 3 \) matrix, \( p \) and \( p' \) are points in \( I \) and \( I' \) respectively and \( \sim \) means equality up to a scale factor. Boufama [5] has developed a method that can be used to calculate this homography using a minimum of seven matched points: three points from the projection of an arbitrary plane in the scene and any other four scene points. An overview of this method is presented here.

As mentioned before, the plane homography is represented by a \( 3 \times 3 \) matrix function. Thus, the goal is to find the nine matrix entries that define this function. However, this homography
is homogeneous. This means that there are exactly eight independent parameters to solve for. Consider the notation used in figure 5.2.

Figure 5.2: Three Points Defining a Planar Region

To begin, this method assumes that three points from the scene region have been matched in the left and right images. Plugging these three points into equation 5.1 yields six linear equations;
however, there are eight unknowns. Instead, the three points are used to simplify the equation. Without loss of generality, a new coordinate system is chosen such that:

\[ p_1 = (0, 0, 1)^T \quad p'_1 = (0, 0, 1)^T \]
\[ p_2 = (1, 0, 0)^T \quad p'_2 = (1, 0, 0)^T \]
\[ p_3 = (0, 1, 0)^T \quad p'_3 = (0, 1, 0)^T \]
\[ p_4 = (1, 1, 1)^T \quad p'_4 = (1, 1, 1)^T \]

(5.2)

where \((p_i, p'_i), i = 1...3\) are the three original points. The point \(p_4\) is not used in the simplification, thus does not need to be co-planar with \(\langle p_1, p_2, p_3 \rangle\). Under the new coordinate system, \(H\) becomes

\[
\begin{pmatrix}
\alpha & 0 & 0 \\
0 & \beta & 0 \\
0 & 0 & \gamma
\end{pmatrix}
\]  

(5.3)

This simplified matrix has three parameters, but since \(H\) is not a singular matrix one of these parameters may be set. In this case \(\gamma\) is set to one. The problem now is to find the unknowns \(\alpha\) and \(\beta\). For this, any four matched scene points are required. Consider the point \(P\), in figure 5.2. This point has a projection, \(Q\), onto the virtual scene plane, \(\Omega_{123}\). Note that both \(P\) and \(Q\) project to \(p\) in the left image, but project to \(p'\) and \(q'\) respectively in the right image. Using the homography the coordinates for \(q'\) are

\[ Hp \sim q' \sim (\alpha x, \beta y, t) \]  

(5.4)

where \(\alpha\) and \(\beta\) are unknown. This reduces the problem to finding \(\alpha\) and \(\beta\).

Fortunately, the epipolar constraint provides the additional equations necessary. Note that the point \(q'\) lies on the epipolar line formed by \((e', p')\). This is represented in the equation

\[ (e' \times p') \cdot q' = 0 \]  

(5.5)

where \(\times\) is the cross product and \(\cdot\) is the scalar product. Plugging the coordinates for \(q'\) into equation 5.5 yields the following equation:

\[
(e'_y - t' e'_x) \alpha x + (t' e'_x - e'_zx') \beta y + (e'_x e'_y - y' e'_z) t = 0
\]  

(5.6)

\(^1\)Note here that the plane homography could be solved here by taking four planar points. However, it is more difficult to get four coplanar points without any a-priori knowledge of the plane
This equation has five unknowns, but only four of them are independent. Thus, these equations, and ultimately the plane homography can be solved using four additional scene points. The total number of matched points includes the three original points and the four other scene points.

It is important to note that no assumptions have been made about the initial three points. Particularly, it is not assumed that these points belong to a physical plane in the scene. The method to be presented determines whether or not the three points correspond to a physical plane. However, heuristics can be used to choose these points in a way that increases their probability of belonging to a physical scene plane.

5.4 Dense Matching and Region Segmentation

The previous section showed how to calculate the plane homography. To use this homography a planar region in $f$ must be identified. This thesis presents a method that uses the plane homography to simultaneously segment and match the region. An overview of this process is as follows. From the three planar points used to calculate the homography a seed region is chosen. The boundary of this seed region is broken into control points, used for growing the region. This growing process grows the circle outward as long as the homography is valid. The basic steps for this algorithm is presented in figure 5.3. The details for these steps will be discussed in the following sections.

Region Growing Algorithm

Inputs:
3 Matched Points, denoted $P$
At least 4 Other Matched Points, denoted $Q$

Begin

$H \rightarrow $CalculateHomography$(P, Q)$

$R \rightarrow $CalculateSeedRegion$(P)$

$R' \rightarrow $CalculateControlPoints$(R)$

For Each Control Point

While( Correlation for Homography Matched Point is Good )

Grow ControlPoint()

End

Figure 5.3: Basic Steps for Plane Matching/Segmentation
5.5 Calculating the Seed Region and Control Points

Choosing a proper seed is essential to the success of any region growing algorithm. In choosing the seed for this algorithm three factors were considered:

1. Location
2. Size
3. Shape

The location of the seed is very important. It is critical that this seed lie in the same plane as the initial three points. If the seed lies outside of the plane then no growth will occur.

There is a trade off when choosing the size of the seed. It is desirable to keep the seed small, giving it a greater probability of being entirely inside of the plane. However, bigger seeds can be broken into more control points, giving a more accurate segmentation.

Finally, the shape is an important issue. From a theoretical standpoint, the shape of the seed region does not matter. However, from a computational viewpoint some shapes are easier to grow than others. For example, a triangle is difficult to grow. Growing each of the three sides of the triangle does not cover the entire range of growth. As well, it is difficult to determine how control points from each side should be connected to form a closed region.

For this method a circular seed region, based on the initial three points, is chosen. The starting point of this seed is the circle inscribed in the triangle formed by the three points. The center of this circle lies at the intersection of the perpendicular bisector rays of any of the triangles two angles. The radius of the circumscribed circle is as shown in the following equation.

\[ R = \frac{s \ast (s - a) \ast (s - b) \ast (s - c)}{s} \]  \hspace{1cm} (5.7)

where

\[ a = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \]
\[ b = \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2} \]
\[ c = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} \]
\[ s = \frac{a + b + c}{2} \]

and \((x_1, y_1), (x_2, y_2), (x_3, y_3)\) are the three vertices of the triangle.

The seed is initially planted at the circumcenter of the triangle. The radius of the seed is
\[ r = \frac{1}{n} \times R \quad n > 0 \] (5.8)

The center of the circle is transformed using the homography and its presence in the plane is verified using a correlation measure. If the circumcenter does not lie in the plane another center may be chosen by trying a new center. This new center may be on any arbitrary vector no more than \( \frac{R-2}{2} \) from the circumcenter. This ensures that any new circular seed region will also lie inside the triangle.

Now that the seed region has been calculated, the boundary must be divided into control points. These control points are grown, increasing the size of the region. The control points are equally spaced around the circumference of the circle. The following equations are used to calculate the \( x \) and \( y \) parameters of the control point respectively.

\[ x_{cp} = x_{center} + \cos(\Theta_{inc}) \times r \] (5.9)

\[ y_{cp} = y_{center} + \sin(\Theta_{inc}) \times r \] (5.10)

where \((x_{cp}, y_{cp})\) is the center of the seed region and \( \Theta_{inc} \) is an incremental angle value between 0 and 360 degrees. Control points are calculated by sampling \( n \) angle increments around the circle. The larger the value of \( n \) the more control points the circle will have. There is a trade-off when deciding the number of control points to use. The more control points in the seed region the more accurate the results. However, choosing a lower number of control points reduces the amount of computation required in the growing step, thus minimizing the total cost.

5.6 Growing the Region

Growing a circular region is a relatively simple process. Once the initial seed region has been chosen the growing step occurs. Recall that the circumference of the circle is broken into control points. In image \( I \), each control point is grown along the vector pointing from the center of the circle through the control point, as in figure 5.4.

After each growing iteration the control points are matched to \( I' \) using the plane homography. The quality of the matched control points is evaluated using a correlation measure. If the correlation yields a good score then the control point continues growing. Typically, a threshold, \( t \), is selected as the minimum correlation value representing an acceptable match. The growing may continue as long as the homography matched point is validated by the correlation score.
Occasionally, a mismatch may occur, improperly stopping the control point from growing. To reduce this problem a control point is allowed to continue growing until $n$ consecutive bad matches occur. Once this happens the control point reverts back to the last good match and the growth for that control point stops.

Remember that the three initial points used to calculate the homography are not assumed to lie on a physical plane in the scene. If these points do not correspond to a physical scene plane then little or not growth will occur. In this case the results are disregarded. To minimize this case, a heuristic can be used to choose the points. An example of one such heuristic is discussed in the next section.

5.7 Finding the Initial Matches

To automate this method there must be an algorithm for choosing the initial three points and four additional points. The four general scene points can be acquired through a sparse matching step, but the initial three points are more difficult to identify.

Points on the boundary of the region can be found using edge detection. To find these points one of Deriche's [21] edge detection algorithms was used to find the edge chains for the image. For each edge chain, three points along the edge are taken. These three points, in addition to the points found by the sparse matching algorithm, are then used to calculate the homography. If the region is planar then it is grown, otherwise the correlation measure will not validate the homography and no growing occurs.

Finding the edge of the region does not guarantee that it will be grown properly. There are two possible scenarios, shown in Figures 5.5 and 5.6, that can occur when a planar region is found.

1. The selected edge points form a seed region inside of the planar region

2. The selected edge points form a seed region outside of the planar region
Further work needs to be done to ensure that the seed is always placed inside of the planar region.

Figure 5.5: Scenario 1 - Edge Detection Properly Plants the Seed

5.8 Discussion of Homography Growing

Several new perspectives for the problems of image segmentation and point matching are provided in this thesis. First, the proposed technique provides a unified view of image segmentation and dense matching for stereo vision. The similarity measure for the region growing method is the qual-
Figure 5.6: Scenario 2 - Edge Detection Improperly Plants the Seed
ity of homography matched pixels. Since the pixel correspondence information is used to validate
the region growth, both image segmentation and dense matching are achieved simultaneously.

The segmentation achieved by this method does not depend solely on pixel intensity measures.
Rather, planarity is used as the similarity measure. Therefore, this method can robustly perform
segmentation of regions that contain several different intensity patterns. Many traditional segmen-
tation techniques would segments these areas into separate regions, despite the fact that they may
belong to the same object. The lighthouse in figure 5.7 represents a single region with two main
intensity patterns.

![Lighthouse Image]

Figure 5.7: The lighthouse represents a non-homogeneous region

The majority of the computation time for traditional matching methods is in the evaluation of
correlation measures. To match a single pixel, numerous correlation calculations are often required.
Only one correlation measure per iteration, per control point is needed for the method presented
in this chapter. In most cases, less than one correlation score per match is calculated. In fact, the
upper bound for correlation calculations is one measure per match. This is in the case where a
flood fill growth strategy is used.

This method will not match an entire image; however, entire planes are quickly matched. If the
scene is abundant in planes, a large percentage of the image pixels will be matched. This reduces
the number of pixels that must be matched during the traditional, correlation-based methods,
reducing the overall matching time.
5.9 Experiments

This section presents some experimental results. These results do not take the place of complexity analysis, rather they show how this technique can save time when matching/segmenting real images.

The nature of the technique presented in this chapter is different from the traditional image segmentation and point matching approaches. It cannot be used to segment and match the whole image. However, when planar regions exist, it can be used to quickly segment and match the planes in the image. This makes it impossible to simply compare computation time with existing matching algorithms. Instead, the following quantities will be evaluated:

1. Total number of pixels matched by technique
2. Percentage of pixels matched by this technique
3. Time required to match these pixels
4. Adjusted time in terms of entire image
5. Time saved using the plane matching
6. Visual Evaluation of the Segmentation

The first three quantities measure the raw performance of the plane matching technique. These measures describe what has been matched and how much computation has been required. The final two measures are used to compare plane matching to conventional dense matching techniques. The adjusted time measures how much time the method would have taken if it could have matched every pixel in the image. This is calculated as follows:

$$
Time_{Adjusted} = \frac{Time_{Measure}}{NumberofMatchedPixels} \times TotalPixels
$$

(5.11)

The previous measure shows the matching speed over the entire image. However, the planar matching method has only matched a subset of the image pixels. A more interesting measure is the time saved by using the planar method versus one of the traditional methods.

Three stereo pairs, containing well defined planes, were chosen for testing. The first pair of images is a large textured square plane set against a mostly white background. The large contrast between the plane and the background should ensure that the plane is accurately matched and segmented.

The walls from a stereo pair of a model castle comprise the second test image. Although this is a laboratory scene, it provides more complexity than the first image pair. The walls of the castle
contain several parallel planes that are very close to each other. For example, the plane of the tower is parallel to the rest of the castle, but it juts out from the main part of the wall.

The final scene is a real-world scene of a church. The walls of the church contain many planes, just as many other real-world indoor and outdoor man made scenes. In addition, the walls of the church are very different than the background sky. The transition between the church and the sky provides an obvious departure from the plane and should be recognized by the planar technique of this thesis.

Results for the planar method are compared with the epipolar line search method described in Chapter 4. This method searches for matches on an interval of the epipolar line. The search interval along the epipolar line is set by examining the maximum disparity amongst the interest points, \((x_{\text{max}}, y_{\text{max}})\), from \((x, y)\) in the left image to \((x', y')\) in the right image. The algorithm chooses the point \((x_i, y_i)\) as a starting point, where this point is defined as the point in the right image that lies on the epipolar line and is closest to the coordinates \((x, y)\).

The purpose of these tests is to show the fitness of the planar algorithm, given the three original points. Therefore the three initial points have been manually selected. The additional points required and the epipolar geometry have been calculated as a pre-processing step.

The epipolar line matching was implemented in C, while the plane matching method was implemented in C++. All testing was done using an AMD 533 MHz processor running under Redhat Linux 7.1.

![Figure 5.8: Stereo Pair of a Simple Plane, 576 x 768 Pixels Each](image-url)
Figure 5.9: Stereo Pair of Building Model, 576 x 384 Pixels Each

Figure 5.10: Stereo Pair of a Church, 576 x 384 Pixels Each
CHAPTER 5. MATCHING AND SEGMENTATION USING PROJECTIVE GEOMETRY 49

The Simple Textured Plane

Two different tests were performed using this pair. The first test defined a seed region using the three planar points from corners of the rectangles. This provided the large seed region as shown in Figure 5.11. The second test uses a smaller seed region defined by three arbitrary planar points. Both seed regions were able to match the plane. In both, cases the region grew outside of the texture triangle region. This is because the exterior area matched by the growing process is close to the plane of the rectangle. It is easy to see the quality of these matches.

The seed region used for the first test is larger than that used for the second test. Thus, there are more control points used in this first test. As expected, the use of more control points allows for a more accurate approximation of the region. This can be seen by examining the bottom right of each detected region. The lack of control points in the second test causes the exclusion of part of the region. In addition, the larger number of control points in the first test translates into more computation time.

It is quite evident in the first region that several control points have grown outside of the plane. In this case, the pixels encompassed in the additional growth lie close enough to the plane that the plane homography validates them as proper matches. For the purpose of matching, this behaviour is desirable, since more points have been inexpensive matched. However, this behaviour is not desirable for image segmentation since the extra points obviously do not belong to the region of the plane.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pixels Matched</th>
<th>Matching Time</th>
<th>Percentage of Image</th>
<th>Adjusted Time</th>
<th>Time Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epipolar Search</td>
<td>168,019</td>
<td>483.00s</td>
<td>76%</td>
<td>636.00s</td>
<td>N/A</td>
</tr>
<tr>
<td>Plane Matching 1</td>
<td>72,992</td>
<td>3.66s</td>
<td>33%</td>
<td>10.00s</td>
<td>206.17s</td>
</tr>
<tr>
<td>Plane Matching 2</td>
<td>68,260</td>
<td>0.96s</td>
<td>31%</td>
<td>3.11s</td>
<td>255.27s</td>
</tr>
</tbody>
</table>

Table 5.1: Results for Simple Plane Tests

Wall Scene

The results of this test are not perfect, but in each case a significant amount of the wall is matched using planes. However, the matching of the tower offers some promise. The control points on the right side stop their growth as they go from the tower to the background. As well, many control points stop their growth at the transition between the wall and the background. Visually, the control points seem stop their growth while the matches are still good. Thus, the problem may lie in the correlation. Even though this match may be acceptable, the correlation measure is not in
Figure 5.11: Seed Region and Growth for Text Plane 1
Figure 5.12: Seed Region and Growth for Text Plane Test 2
the acceptance range to continue growth. This problem may be solved by using a more complex correlation threshold selection method or a more robust correlation measure.

In the wall tests, the regions are under segmented. The expected regions are either the entire castle wall or the tower of the castle.

Figure 5.13: Seed Region and Growth for Wall Scene Test 1
Figure 5.14: *Seed Region and Growth for Wall Scene Test 2*

<table>
<thead>
<tr>
<th>Method</th>
<th>Pixels Matched</th>
<th>Matching Time</th>
<th>Percentage of Image</th>
<th>Adjusted Time</th>
<th>Time Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epipolar Search</td>
<td>161,766</td>
<td>114.67s</td>
<td>73.14%</td>
<td>158.78s</td>
<td>N/A</td>
</tr>
<tr>
<td>Plane Matching 1</td>
<td>62,023</td>
<td>1.53s</td>
<td>28.04%</td>
<td>5.46s</td>
<td>38.50s</td>
</tr>
<tr>
<td>Plane Matching 2</td>
<td>13,566</td>
<td>0.70s</td>
<td>6.13%</td>
<td>11.42s</td>
<td>9.03s</td>
</tr>
<tr>
<td>Plane Matching 3</td>
<td>55,753</td>
<td>1.25s</td>
<td>25.2%</td>
<td>4.96s</td>
<td>35.05s</td>
</tr>
</tbody>
</table>

*Table 5.2: Results for Wall Tests*
Figure 5.15: Seed Region and Growth for Wall Scene Test 3
Church

The church test validates the use of this method for real-world scenes. In the first test, the planar algorithm provides an accurate matching/segmentation of the steeple of the church. Starting from a relatively small seed most of the front steeple of the church has been matched/segmented. Note how the control points stop their growth at the transition between the steeple in the sky. However, on the left side the plane offers a good match for a small section of the side plane of the steeple.

There are two interesting special cases displayed in the third church test. First, the door on the front of the church provides a break in the plane. Note how the control points stop their growth as they approach the door. In addition the stop sign occludes part of the plane. Here the plane grows around the stop sign.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pixels Matched</th>
<th>Matching Time</th>
<th>Percentage of Image</th>
<th>Adjusted Time</th>
<th>Time Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epipolar Search</td>
<td>298,788</td>
<td>1,060.67s.00s</td>
<td>76%</td>
<td>1,395.62s</td>
<td>N/A</td>
</tr>
<tr>
<td>Plane Matching 1</td>
<td>26,223</td>
<td>1.16s</td>
<td>6.67%</td>
<td>17.39s</td>
<td>91.93s</td>
</tr>
<tr>
<td>Plane Matching 2</td>
<td>10,939</td>
<td>0.42s</td>
<td>2.78%</td>
<td>15.10s</td>
<td>92.67s</td>
</tr>
<tr>
<td>Plane Matching 3</td>
<td>21,333</td>
<td>0.97s</td>
<td>5.4%</td>
<td>17.80s</td>
<td>74.76s</td>
</tr>
</tbody>
</table>

Table 5.3: Results for Church Tests

Calculation of the Homography

The computation times for these tests do not include the calculation of the homography. However, the calculation of the homography does not constitute a major part of the algorithm. In fact, the calculation time for the homography was measured between 0.02s and 0.04s for all of the above tests.

5.10 Conclusion

This chapter presented a method of segmenting and matching planar regions. Starting with three matched points from an arbitrary plane, and at least four additional points, a plane homography can be calculated. This homography maps points from the plane in the left image to points in the plane from the right image. This provides a closed-form matching function for the points in this plane. Using the three initial points a seed region is formed and this seed is grown as long as a correlation measure validates the plane homography.

Tests measuring the performance of the planar algorithm, with respect to traditional matching,
Figure 5.16: Seed Region and Growth for Church Scene Test 1
Figure 5.17: *Seed Region and Growth for Church Scene Test 2*
Figure 5.18: Seed Region and Growth for Church Scene Test 2
are presented. Three scenes were used including a simple textured plane, a mock castle model and a real-world church scene. Comparisons with the epipolar search method show this method to perform remarkably well. A large portion of most of the scenes in these images was able to be matched with the planar matching. The performance has shown that the use of this method to match the planar regions will reduce the overall matching time.

The segmentation provided by the method is crude. If only a rough approximation of the region is needed then this method offers a computationally inexpensive solution. Overgrowth can occur when points outside of the region lie close enough to the plane to be validated by the homography. Undergrowth occurs when the correlation score does not validate matches that actually lie within the planar region. Despite this, the segmentation provided is still a good result. Image segmentation is a difficult problem where very few solutions have a high degree of success.
Chapter 6

Conclusion

This chapter will give a brief summary of the topics discussed in this thesis. The results of the method presented in Chapter 5 are discussed in terms of image segmentation and dense matching. Finally, future areas of improvement for this method will be presented.

6.1 Summary

The field of computer vision studies the interpretation and use of visual data. There are many interesting applications that make use of this data. For example, token tracking traces the movement of a token object through a sequence of images. Image synthesis produces new perspectives of a scene from existing photographs. One of the more recent applications of computer vision is augmented reality. Augmented reality overlays virtual objects onto an image of the real scene. For example, a user wearing an augmented reality helmet will see the real world scene, along with some virtual objects. Another application is the addition of virtual objects to a movie sequence.

The aforementioned applications are high-level. These applications rely on several low-level primitive operations. One of these primitives is image segmentation. Segmentation is a first step to dividing an image into its composite objects. This step is necessary for applications such as token tracking and image data-basing. An overview of image segmentation techniques is provided in Chapter 3. Segmentation is broken into three categories: global, boundary finding and region detection techniques. Each class is illustrated with several examples.

Another important low-level primitive is point matching. Many vision operations require two or more images of the same scene. Using such a stereo pair introduces more information including 3D structure and information about occluded objects. However, for this information to be useful it is necessary to know how the two images are related. The problem of point matching is to
CHAPTER 6. CONCLUSION

determine how the pixels in one image correspond to the pixels of another image of the same scene. This problem is discussed in Chapter 4. Point matching techniques using the epipolar geometry, numerical relaxation, dynamic programming and graph theory are discussed in this chapter.

The speed and accuracy of point matching and image segmentation is crucial to many computer vision applications. However, there are currently no universally acceptable methods for either of these tasks. A particular problem of existing methods is computation time. Many applications need point matching and/or image segmentation in or close to real time. However, the methods producing good quality results are computationally complex. Conversely, faster methods are not always accurate. A method that simultaneously and accurately segments and matches a stereo pair of images is presented in Chapter 5 of this thesis.

6.2 Matching and Segmentation of Planes

Chapter 5 of this thesis presents a technique that can be used to simultaneously segment and match the planar portions of an image. Planar regions are common in many man made environments and include walls, tables and many other objects. Quickly performing image segmentation and matching on these parts of an image can significantly reduce the number of pixels to be processed by computationally expensive segmentation and/or matching techniques.

To use this method three points from an arbitrary planar region must first be identified and matched. Using these three points and any four other matched points a projective homography mapping the plane from one image to the other is calculated. If the homography corresponds to a physical scene plane then it is used to grow the planar region. A correlation function is used to validate the growth. The seed for the growth is chosen to lie inside of the triangle formed by the initial three points.

6.3 Results of the Plane Segmentation and Matching

This method was tested in comparison with a traditional epipolar line search matching technique. The results show that this method can calculate the matching for the scene remarkably quicker than the epipolar search method.

The segmentation provided by this method is not perfect. However, image segmentation is a difficult problem and very few solutions are acceptable. Taking this into account, the segmentation provided by this method is comparable to other segmentation techniques in the literature.
There are two types of errors associated with the results of the segmentation provided by this technique. First, the region may keep growing even after crossing the region boundary. This occurs when the non-planar pixels lie close enough to the plane that the homography is a good approximation of the mapping. The other problem is a premature end to the growth. This is caused by a failure in the validation step. The quality of the matches at these prematurely stopped control points is still good, but the validation step fails to affirm this fact.

To automate the selection of the initial three points edge chains are used. It was discovered that there are two cases that occur when creating the seed region using edges. One where the seed lies inside of the intended region and one where it lies outside of the intended region. In the case where the triangle lies inside of the intended region the results are good; however, when it does not lie inside the region it is impossible to find a good seed region.

This technique is good as a first step for matching and image segmentation. By applying this technique to each of the planar regions of an image, a potentially large percentage of the image pixels can be segmented/matched. Following this, one of the classical segmentation or matching techniques can be applied to process the remaining image pixels. The computation for the overall process should be considerably reduced since the planar step is fast.

6.4 Future Work

For the most part, the results presented in this thesis relied on the manual selection of the three planar points. While an edge map is used to automatically identify these three points, there is still a problem to be solved here. The concavity of the edge determines whether or not the seed will actually lie inside of the region. A method to ensure that the seed is always planted inside of the region has yet to be developed.

This method provides a crude image segmentation; however, there are still cases where the seed either grows too far or stops growth prematurely. Solving this problem involves a closer examination of the evaluation step. A translation of the entire correlation window was used to try to improve growing, but was found inefficient.

In this thesis testing was limited to matching/segmenting one region at a time. When using this method to match/segment several regions in the image it is possible that a pixel may belong to more than one region. In this case a method must be developed that assigns each of the pixels to only one region.
Vita Auctoris

Personal Information

Name: David John O'Connell
Date of Birth: February 1st, 1977
Birthplace: Prince Edward Island, Canada

Education

B.Sc., First Class Standing, Double Major in Computer Science and Mathematics
University of Prince Edward Island, 1999

M.Sc., Specializing in Computer Science
University of Windsor, 2002
Bibliography


