AN IMAGE SEGMENTATION BASED METHOD FOR STEREO CORRELATION WITH VIEWPOINTS AT DIFFERENT DEPTH LOCATIONS

By

XIN GAO

(Under the Direction of Hamid Arabnia)

ABSTRACT

This thesis presents a method that uses image segmentation techniques to compute the depth image from a pair of stereo images which are taken from two viewpoints placed along the z-axis in the same scene. Instead of applying the traditional window-based matching methods, this approach uses the similarity of the image segments / components to find the matches, and then computes the depth information from the ratio of the sizes of the matched components.

INDEX WORDS: Image segmentation, Stereo correlation, Stereo matching, Watershed method
AN IMAGE SEGMENTATION BASED METHOD FOR STEREO CORRELATION WITH
VIEWPOINTS AT DIFFERENT DEPTH LOCATIONS

by

XIN GAO

Bachelor of Economics, Fudan University, P.R.China, 1999

A Thesis Submitted to the Graduate Faculty of the University of Georgia in Partial Fulfillment of
the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2005
AN IMAGE SEGMENTATION BASED METHOD FOR STEREO CORRELATION WITH
VIEWPOINTS AT DIFFERENT DEPTH LOCATIONS

by

XIN GAO

Major Professor: Hamid R. Arabnia
Committee: Eileen T. Kraemer
Liming Cai

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2005
ACKNOWLEDGEMENT

First of all, I would like to thank Dr. Hamid Arabnia for his encouragement and constant academic support. I would also like to thank Dr. Liming Cai and Dr. Eileen Kraemer for their many thoughtful suggestions.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Basic Concept</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Overall Procedure</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Common Stereo Correlation Problems</td>
<td>6</td>
</tr>
<tr>
<td>2  RELATED STEREO CORRELATION WORK</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Block/Area-based Matching</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Gradient-based Optimization</td>
<td>14</td>
</tr>
<tr>
<td>2.3 Feature Matching</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Regional/Surface-based Matching</td>
<td>16</td>
</tr>
<tr>
<td>2.5 Dynamic Programming</td>
<td>19</td>
</tr>
<tr>
<td>2.6 Intrinsic Curves</td>
<td>22</td>
</tr>
<tr>
<td>2.7 Graph Cuts</td>
<td>23</td>
</tr>
<tr>
<td>2.8 Summary</td>
<td>24</td>
</tr>
<tr>
<td>3  IMAGE SEGMENTATION TECHNIQUES</td>
<td>26</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>26</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 1 A taxonomy of stereo matching techniques 9

Table 2 Formula for measuring the goodness of local area-based matching. $I_1$ denotes template window, and $I_2$ is the candidate window, $\sum_{(x,y) \in W}$ indicates the summation over the window. 11

Table 3 The list of program I implemented for the stereo correlation method, and their code statistics 80
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fundamental concept of stereo correlation. The two viewpoints are placed along the x-axis, looking in the same direction. The point P in the scene is projected to P1 in the image plane of V1, and it is projected to P2 in the image plane of V2.</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Two arbitrary images of the same scene. The orientations of two cameras may not be the same, and the image pairs need to be rectified into parallel image planes.</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>An example of Half Occlusion. Point B appears in Image1, but it is occluded in Image2, and it is said to be half-occluded.</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>The test result of area-based matching. (a) the reference image; (b) the ground truth image; (c) the result of area-based matching.</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Hierarchical feature-base matching (a) two possibly matched surfaces; (b) the hierarchical structure of the two matched surface; the structure is built in a top-down fashion (surfaces, edges, vertices), and the matching is performed in a bottom-up fashion (vertices, edges, surfaces).</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>The test result of color segmentation based matching. (a) the reference image; (b) the ground truth image; (c) the result of color segmentation based matching.</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>The result of Tao and Sawhney’s algorithm [29]. (a) the left image (reference image); (b) the right image (paired image); (c) the final depth map from their algorithm.</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>The space-disparity map used in Dynamic Programming, intensities in the map represent the respective costs of potential matches (a) The map uses left-right axes (b) the map uses left-disparity axes.</td>
<td>19</td>
</tr>
<tr>
<td>9</td>
<td>3D search space for the intra- and inter-scanline search by dynamic programming. The side faces correspond to the left and right images. Connected edges in each image form sets of intersections (2D nodes); each set is called a 3D node. Selection of a set of paths is done at every 3D node.</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>The test result of area-based matching. (a) the reference image; (b) the ground truth image; (c) the result of dynamic programming.</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>The intrinsic curve map. (a) left and right intensities; (b) the derivatives from (a); (c) the intrinsic curves formed by plotting one against the other.</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>The 3D representation of maximum flow methods.</td>
<td>23</td>
</tr>
</tbody>
</table>
Figure 13 The test result of graph cut/maximum flow methods. (a) the reference image; (b) the ground truth image; (c) the result of maximum flow methods.

Figure 14 The high-pass filter for detecting point discontinuities.

Figure 15 The high-pass filter for detecting line discontinuities. (a) Mask for detecting horizontal lines; (b) Mask for detecting vertical lines; (c) Mask for detecting $+45^\circ$ lines; (d) Mask for detecting $-45^\circ$ lines.

Figure 16 The formation of edges; from top to bottom, the image regions containing the edge, the 1D view of grayscale value of that region, the first-order derivative, and the second-order derivative. (a) a perfect sharp edge; (b) an imperfect blurred edge.

Figure 17 (a) A 3x3 image with the grayscale value associated with each pixel; (b) the cost of each edge segment is marked; (c) an edge of minimal cost is found.

Figure 18 The fundamental concept of stereo correlation. $V_1$ and $V_2$ are two cameras along the z-axis in the same scene; $AB$ and $CD$ are the front surfaces of two objects, and $A_1B_1$ and $B_2B_2$ are the projections of $AB$ on the image planes of $V_1$ and $V_2$, respectively; similarly for $C_1D_1$ and $C_2D_2$ to $CD$. The image taken by $V_1$ is called the reference image (closer one), and the image taken by $V_2$ is called the paired image (further one).

Figure 19 The principal concept of dam construction in the Watershed method. (a) punching a hole at the regional minimum; (b) flooding the topography from below by letting the water rise through the holes with a uniform rate; (c) if two catchments basins are about to merge, a dam is built between them to prevent the merging; (d) after the maximum gradient level is reached, all points except the dam points are flooded, and all dams are built.

Figure 20 Using morphological dilation to construct dams at flooding step n. (a) two components are found at flooding level n-1; (b) 8-connectivity window for dilation; (c) At flooding level n both components will be expended; this is the first round of dilation; (d) dams are built to prevent merging; (e) another round of dilation in level n; (e) more dams are built.

Figure 21 The first scan of component labeling. The gray points are the ones being checked, and the algorithm only checks the upper and left neighbors of the point. (a) none of the neighbors has a label, and a new label is assigned to the gray point; (b) only the upper neighbor has a label, and that label is assigned; (c) only the left neighbor has a label, and that label is assigned; (d) both of the neighbors have the same label, and that label is assigned; (e) two neighbors have different labels, then the upper label is assigned, and the equivalent label handling process is invoked.

Figure 22 The directed graph is used to store the equivalent labels. The equivalent labels handling process will try to group equivalent labels as trees, and meanwhile try to minimize the height of each tree to 1. Finally the graph will become a forest with trees of height 0 or 1, and the number of trees are the number of labels we will use in the
output label map, and each root node represents the minimal labels of a group of equivalent labels in that tree. Figure 23 the properties of a component; (a) a component labeled in the image; (b) basic perimeter = 18; (c) basic size = 23; (d) external perimeter = 14; (e) external size = 26.48

Figure 24 the superimposed images. (a) point $A$ in the reference image corresponds to $A'$ in the paired image, the path from $A$ to $A'$ heads to the origin. (b) The search space of component $AB$ is in gray shade, where corresponding component $A'B'$ may reside.

Figure 25 the relationship between the ratio of the projected length of a line $AB$ and the $z$-coordinate of this line. The two cameras are placed along the $z$-axis at $V_1$ and $V_2$, and focus at the same direction with the same focal length $f$. The baseline between two cameras is denoted as $T$. The line $l$ is projected into the closer image place as $n$, and projected into the further image plane as $m$. We can derive the $z$ coordinate by the ratio of the lengths of $m$ and $n$.

Figure 26 The segmented artificial images and the segmentation maps. (a) minSize=100; (b) minSize=500; (c) minSize=1500; (d) minSize=1900; (e) minSize=2500

Figure 27 The segmented “bean” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=1500

Figure 28 The segmented “scene” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=2500

Figure 29 The segmented “cloud” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2900

Figure 30 The segmented “stone texture” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2700

Figure 31 The segmented “x-ray” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2700

Figure 32 The segmented “yosemite” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2700

Figure 33 The segmented paired “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=1500

Figure 34 The segmented reference “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=2500

Figure 35 The segmented paired “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2100
Figure 36 The segmented reference “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=1700 ................................................................. 66

Figure 37 The segmented paired “palm” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=1900 ................................................................................. 66

Figure 38 The segmented reference “palm” photos and segmentation maps; (a) minSize=100; (b) minSize=900; (c) minSize=2500 .................................................................................... 67

Figure 39 The screen shot of the Java 3D test tool; the reference camera (closer one) is attached to the panel on the left, and the paired camera (further one) is attached to the panel on the right ......................................................................................................................... 68

Figure 40 The stereo correlation result on the parallel cubes in the artificial scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image ......................................................................................................................... 69

Figure 41 The stereo correlation result on two cubes of different distances to the cameras in the artificial scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image ......................................................................................................................... 71

Figure 42 The stereo correlation result on two textured objects with different distances to the camera in the artificial scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image ......................................................................................................................... 73

Figure 43 The stereo correlation result on three cookie boxes with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image ......................................................................................................................... 75

Figure 44 The stereo correlation result on palm trees with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image ......................................................................................................................... 76

Figure 45 The stereo correlation result on various objects with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image ......................................................................................................................... 77

Figure 46 The stereo correlation result on various objects with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image ......................................................................................................................... 78
1 INTRODUCTION

Stereo correlation refers to the problems of extracting three-dimensional structures of a scene from two or more images taken from distinct viewpoints. It has been an intense area of computer vision research for decades. Early work, conducted in the 1970s and early 1980s, was primarily focused on local window-based and feature-based matching. Barnard and Fischler [1] reviewed stereo research through 1981, and surveyed those well-known approaches at that time. By the 1990s, accompanied by the improvement of hardware and software, the majority of the stereo community had switched to the study of stereo correlation in global scale, and stereo research had matured in many ways. Brown and Burschka [2] reviewed the recent advances in computational stereo through 2002, focusing primarily on the correlation methods, methods of occlusion, and real-time implementations of stereo correlation.

In this thesis, I focus on a specific scenario in which two stereo images are taken of the same scene from different viewpoints along the z-axis. In this scenario, the objects in the scene are scaled and located in various places in the stereo image pair; therefore, we can not use the traditional local window-based matching to perform stereo correlation. Instead, I segment the two images at different levels and scales, and then try to match each pair of components extracted from the segmentation process between two images. After all possible matches are found, and the matching scores are computed, the results are put into a global matching process, and a set of optimal matches are selected that do not conflict with each other. Finally, a depth image is computed from the ratio of sizes of matched components, and the result is presented in the form of a grayscale image.
The organization of this thesis is as follows: the rest of Chapter 1 introduces some principal concepts and procedures in stereo correlation, and the possible problems that may occur in the matching process are also discussed. Chapter 2 reviews related work in stereo correlation, covering local methods and global methods. Since the correctness and robustness of the segmentation process is critical to the overall result of the stereo correlation method discussed in this thesis, some of the well-known image segmentation techniques are reviewed in Chapter 3. A new image segmentation based stereo matching technique is introduced in Chapter 4 to deal with the special scenario of two stereo images being taken of the same scene from different viewpoints along the z-axis, and the detailed implementation is also discussed in this chapter. Chapter 5 presents the experimental results of the image segmentation and stereo correlation method I have designed and implemented. Conclusions and future refinements of my work are presented in Chapter 6.

1.1 Basic Concept

The fundamental concept of stereo correlation is based on the discovery that an object in the scene will be projected into a unique pair of regions in the image plane from distinct viewpoints, and through perspective projection, objects that have different distances from the viewpoints will be projected differently. For example, two viewpoints are placed at $V_1$ and $V_2$ along the z-axis in Figure 1, looking in the same direction, with the same focal length and orientation. Assume $P$ is a point in 3D space, and it is projected to $P_1$ in the left viewpoint’s image plane, and it is also projected to $P_2$ in the right viewpoint’s image plane, with optical centers at $O_1$ and $O_2$, respectively. If we overlay two optical centers together, we can obtain the disparity of point $P$, which is defined as the distance between $P_1$ and $P_2$; in this case, disparity $d = O_1P_1 + O_2P_2$. We also define baseline $T$ as the distance between $V_1$ and $V_2$, and define the
focal length of a viewpoint as the distance between viewpoint to the image plane of that viewpoint; in this case, focal length $f = O_1V_1 = O_2V_2$. Due to the perspective projection, points of different distances from the viewpoint have different disparities in the image plane, and those closer to viewpoint have larger disparities, and vice versa.

**Figure 1** Fundamental concept of stereo correlation. The two viewpoints are placed along the x-axis, looking in the same direction. The point $P$ in the scene is projected to $P_1$ in the image plane of $V_1$, and it is projected to $P_2$ in the image plane of $V_2$.

We can derive the depth information of the points using the disparities of projected points. Let us move one projection line $V_1P$ from $V_1$ to $V_2$, as depicted in **Figure 1**, and the new line intersects with the image plane at $P_1'$. Two similar triangles, $V_2P_2P_1'$ and $V_2PP'$, are formed, with $PP' = T$ and $P_2P_1' = d$. Deriving from the similarity of triangles, we obtain the following equation: $f \frac{d}{T} = \frac{z}{T}$, and by reforming it, we get the principal formula to compute depth value of a point, depicted in **Equation 1**:

**Equation 1** \[ z = f \times \frac{T}{d} \]
1.2 Overall Procedure

The overall procedure of stereo correlation can be categorized into five steps [3]:

1. Calibration – In the ideal situation, the two viewpoints are identical except for their locations, which means they have the same interior orientations (such as focal length, position of the principal point, distortion characteristics of the lens system, etc.) and exterior orientations (such as optical center position, optical axis direction, etc.). However, in the real world, these characteristics differ from one camera to another, and we need to calibrate two cameras to eliminate or reduce the interferences of those differences [4] [5].

![Figure 2](image)

**Figure 2** Two arbitrary images of the same scene. The orientations of two cameras may not be the same, and the image pairs need to be rectified into parallel image planes.

2. Rectification – Usually rectification is combined with calibration. Consider the situation in **Figure 2**, in which two arbitrary images are taken from the same scene, but the orientations of the cameras are different. The plane defined by point $PO_LO_R$ is called the **epipolar plane**, and the intersection of the epipolar plane with the image plane is called an **epipolar line**. In this case, line $pe$ and $p'e'$ are epipolar lines. Line $O_LO_R$ is defined as the **baseline** $T$, and the intersection of an epipolar line with the baseline is called an **epipole**. In this case, the epipole $e$ corresponds to point $P$ in the left image, and $e'$...
corresponds to the same point \( P \) in the right image. Therefore, the correlated points \( p \) and \( p' \) can be found along the corresponding epipolar lines. By rectifying the image such that the corresponding epipolar lines match with horizontal scanlines, we can simply search along the paired scanlines of both images for the correct matches. We can thus reduce the stereo correlation problem in Figure 2 into the simpler problem in Figure 1, and the computational complexity is reduced dramatically.

3. Preprocessing: besides rectification, sometimes the input image pairs need to be preprocessed before stereo correlation. Preprocessing operations include the following:
   - Enhancement: in some situations, the images need to be processed through a series of enhancement operations before correlation to improve the qualities of the original images, such as noise reduction, sharpening, etc.
   - Transformation: the pixel values of the image can be transformed from one 2D array form into another 2D array form, for example, sign representation. Highly depending on the nature of the problem, such operations can be used to reduce or increase the variation of color values of the original image.
   - Feature Extraction: local and global regional features, such as corners, edges, and boundaries, can be extracted prior to the correlation process, and later be used in the process to improve the efficiency.

4. Matching: after the pair of images is preprocessed, the matching process tries to allocate a correctly corresponding point for each possible point in the reference image. This procedure is the core step of the whole stereo correlation method, and therefore is also the focus of this thesis. The result of this step will be a disparity image in which each point contains the disparity of the reference point on that location.
5. Reconstruction: in this step, a depth image is computed using the disparity information obtained from the previous matching step. The depth image is usually presented in the form of a grayscale image.

1.3 Common Stereo Correlation Problems

Ideally, a point in the reference image can have a corresponding point in the paired stereo image. However, there are many cases in which more than one point from the paired image can be matched to one point from the reference image, or there is no matched point found from the paired image. Different problems can occur in the stereo correlation process, and some common ones are listed below [3] [6].

- **Occlusion**: occlusion occurs when portions of a scene are visible in one image, but not in the other one. As illustrated in Figure 3, points $A$ and $C$ are visible in both images, but $B$ is only visible in Image1, not in Image2. The point $B$ is usually called half occluded [3]. There are some cases in which small narrow objects are in front of some large objects, and they will obstruct some small parts of the large objects in the view. Such occlusion is called complete occlusion, and in this scenario, the corresponding points no longer preserve the same order as the reference points along the scanlines, and the traditional matching process will most likely fail because of it.

- **Repetitive Patterns**: repetitive patterns deceive the matching algorithm into choosing incorrect matches, and without the global information, it is almost impossible to correct such mismatches in the local region.

- **Bland Regions**: some regions do not contain enough information for matching, with the classic example being a featureless wall;
Figure 3 An example of Half Occlusion. Point B appears in Image1, but it is occluded in Image2, and it is said to be half-occluded.

- Perspective Distortion: this kind of distortion occurs as the shapes of objects change when they are viewed from different viewpoints;

- Radiometric Distortion: such distortion results in a constant offset, or an offset multiplied by a gain factor, between pixel values in the stereo pair. These effects are caused due to the differences of the parameters of image acquisition devices, such as gain, bias and gamma factor.

- Specular Reflection: this is caused by the reflectance properties of the object in the scene. Matching algorithms usually assume a Lambertian reflection model (an object reflects light equally in all directions), such that a particular point has the same intensity regardless of the direction it is viewed from. However, this is not always true in the real world.

- Noise: noise is introduced into the matching process in the image acquisition and digitization process. Depending on the nature of the problem to be handled, noise may affect the process significantly.
2 RELATED STEREEO CORRELATION WORK

Stereo correlation for extracting three-dimensional structure from a scene has been studied intensively for decades. One of the major tasks for stereo correlation is to find the correspondence of two points in two or more images taken from distinct viewpoints, and then to estimate the three-dimensional depth information using the disparity value of the reference points.

From the 1970s to the early 1980s, due to the intensive computational complexity of the stereo correlation process and the limitations of computer hardware at that time, research in this area was primarily focused on the matching of sparse-point based features [7]. Meanwhile, some local information based stereo correlation methods had also been developed during that period.

In the later 1980s and 1990s [8], computational resources had been upgraded dramatically, and many stereo correlation methods requiring significant computational power had been developed. Those methods not only consider local matching information, but also combine global information in the matching process. The focus of the stereo correlation methods had also shifted from sparse point matching into generation of dense disparity images during that period. By the mid-1990s [9] [10], stereo correlation research had, in many ways, matured.

Since the later 1990s [2], although the stereo research continues, the major focus of the community has moved into solving specific problems. New algorithms and technologies have been applied to existing methods to generate high quality results and achieve better performance. Research on occlusion removal and real-time stereo implementation has also developed recently [11].
There are roughly two major groups of matching methods being developed: local matching and global matching [2], [3], and a list of these methods are presented in Table 1.

Table 1 A taxonomy of stereo matching techniques

<table>
<thead>
<tr>
<th>Category</th>
<th>Approach</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Methods</td>
<td>Block/Area based matching</td>
<td>Search for maximum score or minimum error over small region, typically using variants of cross-correlation methods [25]</td>
</tr>
<tr>
<td></td>
<td>Gradient based optimization</td>
<td>Minimize a functional, typically the sum of squared differences, over a small region [12], [13], [14].</td>
</tr>
<tr>
<td></td>
<td>Feature matching</td>
<td>Match dependable features rather than intensities themselves [26] [15]</td>
</tr>
<tr>
<td></td>
<td>Regional/Surface based Matching</td>
<td>Use image segmentation [27], [28], [29] and/or other regional information to extract surfaces and use them to guide the matching process [16], [17], [18], [19], [20]</td>
</tr>
<tr>
<td>Global Methods</td>
<td>Dynamic Programming</td>
<td>Determine the disparity surface for scanline as the best path between two sequences of ordered primitives. Typically, order is defined by the epipolar ordering constraints. [31]</td>
</tr>
<tr>
<td></td>
<td>Intrinsic Curves</td>
<td>Map epipolar scanlines to intrinsic curve space to convert the search problem to a nearest-neighbor look up problem. Ambiguities are resolved using dynamic programming [21]</td>
</tr>
<tr>
<td></td>
<td>Graph Cuts</td>
<td>Determine the disparity surface a minimum cut of the maximum flow in a graph. [22] [23] [24]</td>
</tr>
<tr>
<td></td>
<td>Relaxation</td>
<td>Cooperative Algorithm, Relaxation labeling</td>
</tr>
<tr>
<td></td>
<td>Object Space matching</td>
<td>Matching and reconstruct object space models</td>
</tr>
</tbody>
</table>

Local matching only considers the information of a local region in the stereo image pair, and tries to find the best match of the local region within the corresponding local search region. Usually local matching requires less computational complexity, but such methods cannot handle problems such as occlusion and repetitive patterns well. Typical local matching techniques
include local area-based matching [25], gradient-based optimization, feature-based matching [26], and segmentation/regional-based matching [27], [28], [29].

Global matching is usually based on the results of local matching, but it does not only pick the best match found in the local region, but also considers the effects of local match in a global context, and tries to find a globally optimal solution [30]. Usually, global matching requires more computational power, but it can generate results of better quality. Typical global matching techniques include dynamic programming [31], graph cuts, and regional/surface based algorithms.

2.1 Block/Area-based Matching

Block matching methods seek to estimate the disparity for each point in the reference image by comparing a small region about that point (which is called the template window) with a series of regions of the same size (which are called candidate windows) in a corresponding search space in the other image [32]. As stated before, the searching problem can be reduced from a 2D domain into a 1D domain by applying the epipolar constraints in the rectification process. A list of formulas that can be used to measure the goodness of local matching [25] is presented in Table 2.

Ideally, an exact match will generate a score of zero; however due to the problems we discussed before, such as occlusion and projection distortion, we usually expect a minimal value within the search region as the best match. Among all area-based matching formulas listed in Table 2, the sum of absolute differences (SAD) is the simplest and computationally least expensive one. However, it cannot handle the problem that intensity values of two images are shifted by a constant value; SSD and NCC cannot handle this situation either. On the other hand,
the zero mean methods, such as ZSAD, ZSSD, and ZNCC, are designed to deal with such situations.

Table 2 Formula for measuring the goodness of local area-based matching. \( I_1 \) denotes template window, and \( I_2 \) is the candidate window, \( \sum_{(u,v) \in W} \) indicates the summation over the window.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAD (Sum of absolute Differences)</td>
<td>( \sum_{(u,v) \in W}</td>
</tr>
<tr>
<td>ZSAD (Zero mean sum of absolute differences)</td>
<td>( \sum_{(u,v) \in W}</td>
</tr>
<tr>
<td>SSD (Sum of squared differences)</td>
<td>( \sum_{(u,v) \in W} (I_1(u,v) - I_2(x+u,y+v))^2 )</td>
</tr>
<tr>
<td>ZSSD (Zero mean sum of squared differences)</td>
<td>( \sum_{(u,v) \in W} ((I_1(u,v) - \bar{I}_1) - (I_2(x+u,y+v) - \bar{I}_2))^2 )</td>
</tr>
<tr>
<td>NCC (Normalized cross correlation)</td>
<td>( \frac{\sum_{(u,v) \in W} I_1(u,v) \times I_2(x+u,y+v)}{\sqrt{\sum_{(u,v) \in W} I_1^2(u,v) \times \sum_{(u,v) \in W} I_2^2(x+u,y+v)}} )</td>
</tr>
<tr>
<td>ZNCC (Zero mean normalized cross correlation)</td>
<td>( \frac{\sum_{(u,v) \in W} (I_1(u,v) - \bar{I}_1) \times (I_2(x+u,y+v) - \bar{I}<em>2)}{\sqrt{\sum</em>{(u,v) \in W} (I_1(u,v) - \bar{I}<em>1)^2 \times \sum</em>{(u,v) \in W} (I_2(x+u,y+v) - \bar{I}_2)^2}} )</td>
</tr>
</tbody>
</table>

Another problem that cannot be solved by SAD, ZSAD, SSD, or ZSSD, is that the intensity values in one image equal to the intensities in the other image all multiplied by a gain factor. Normalized cross correlation (NCC) and zero mean normalized cross correlation (ZNCC) can be used to solve this problem. In general, NCC and ZNCC can generate better results than SAD and SSD family do, but they require higher computational cost.
The result of area-based matching methods is presented in Figure 4. All area-based matching methods work well on the regions that contain rich texture information; however, if the areas being matched do not contain enough color or grayscale variations to guarantee one (and only one) good match, those methods will fail.

![Image](image_url)

**Figure 4** The test result of area-based matching. (a) the reference image; (b) the ground truth image; (c) the result of area-based matching

One of the improvements for area-based matching is to use Left-Right Consistency Checking. Assume we start with a template window on the left image, and find a correlation window on the right image; now we switch the left and right image by letting the correlation window be the new template window, and search a corresponding region on the left image. If the best correlation window found in the second step matches the template window found in the first step, then we have enough confidence to say it is a good match, and the matching point represented by this template window and correlation window passes the consistency check and is marked as a good match. This method is also a good way to detect occluded areas.

Another possible improvement is to use the Adaptive Window techniques [33]. The selection of window size is the central problem for area based matching. We need a large enough window to obtain sufficient information for texture-less regions, but also need a small enough window to avoid the interference of projection distortion. The adaptive window method allows
us to adjust the window size depending on local variations in intensity and disparity. The algorithm is outlined here:

1. Initial disparity estimation, $d_0(x, y)$ is computed using an area-based method, such as SSD;
2. For each point $(x, y)$, a window is chosen to provide an estimate of the disparity increment, $\Delta d$, with the lowest uncertainty. Then the estimate is updated by $d_{i+1}(x, y) = d_i(x, y) + \Delta d(x, y)$. The window is a rectangular window that can be expanded in four directions, and it can be chosen by applying the following rules:
   a. A 3x3 window is first centered at that reference pixel, then the disparity is incremented and the uncertainty in disparity is computed by a statistical model within the window;
   b. The window is expanded by one pixel in one direction, and the uncertainty for the expanded window is computed again. If expansion increases the uncertainty value, the expansion on that direction is stopped.
   c. The direction resulting in the lowest increase in uncertainty is chosen, and window is expanded by one pixel in that direction.
   d. Iterate back to step (a) until the expansion cannot be performed in either direction, or the window size reaches a pre-defined limit.
3. Iterate back to step 2 until the disparity estimate converges to a pre-defined score, or the maximum number of iterations has been reached.

One more possible approach to improve the performance is to use the Hierarchical Method [34]. The idea is to build a pyramid for each image with different resolution levels, and then do a coarse-to-fine matching. The matching begins with the coarsest resolution level, and we can use any available matching algorithms to do this. After we find the disparities of pixels
on this level, they will be propagated to subsequent finer level as the initial disparity approximation. In this way, the initial disparity is refined towards the finer resolution levels in the pyramid, and we can therefore reduce the size of the search space significantly. One drawback of this method is that if the initial disparity estimation is not correct, the error will be propagated through the finer resolution levels.

2.2 Gradient-based Optimization

Gradient-based methods, or optical flow [12], seek to determine small local disparities between two images by formulating a differential equation relating to motion and image brightness. There is an assumption for the gradient-based methods, that the image brightness of a point in the scene is constant between the pair of stereo images. The horizontal translation of a point from one image to the other is computed by a simple differential equation [13]:

\[ (\nabla_x E) v + b + E_t = 0 \]

Where \( \nabla_x E \) denotes the horizontal component of the image gradient, \( E_t \) denotes the temporal derivative (here referring to the intensity differences between left and right stereo images), \( v \) denotes the translation between the two images, and \( b \) denotes the radiometric bias.

Only translation in the direction of the gradient at a given point may be estimated accurately. Therefore, an additional constraint is needed to achieve reliable results. If the disparities vary smoothly over a small window of pixels, then the disparity at a point may be estimated using least squares on the system of linear differential equation. This equation can be transformed into the form of SSD we introduced in the previous section.

Similar to area-based matching, gradient-based correlation methods are also sensitive to depth discontinuities, since the supporting regions for these methods contain points that come from different depths.
2.3 Feature Matching

Feature-based methods seek to overcome the sensitive problems of area-based and gradient-based methods on regions of depth discontinuities. Feature-based methods usually limit the depth estimation in the supporting regions where specific reliable features can be extracted; therefore, the results generated by feature-based methods are usually sparse depth maps [35].

In the 1970s and 1980s, the feature-based methods received significant attention, largely due to their efficiency and the limitations of computational powers during that period. However, due to the need for dense depth maps and the improvement of computational hardware, many new algorithms have been developed to improve the efficiency of area-based methods, and interest in feature-based method has declined in the last decade.

Figure 5 Hierarchical feature-base matching (a) two possibly matched surfaces; (b) the hierarchical structure of the two matched surface; the structure is built in a top-down fashion (surfaces, edges, vertices), and the matching is performed in a bottom-up fashion (vertices, edges, surfaces).

A hierarchical feature-based method has been proposed in [26]. There are four types of features of different levels: line, vertices, edges, and surfaces. Matching begins at the lowest level of the hierarchy (surface), and proceeds to the highest level (lines). The idea of this feature-based hierarchical framework is similar to the area-based hierarchical framework. It first matches on the coarsest level (surface) to obtain estimation of disparities, and uses them as the initial values in the finer level, depicted in Figure 5.
1. The algorithm first extracts edges from the paired images, and builds the feature hierarchy graph based on the relationships between features. Relationships include structural (i.e., connectivity), perceptual (i.e., parallel, collinear, and proximate), and incompatibility relations (i.e., intersects, overlaps, touches). All potential features in the hierarchy are stored as hypotheses in a relational graph, and inconsistent groupings are detected and removed from the graph.

2. Feature matching is then performed between relational graphs of the stereo images, starting from the surfaces and proceeding to lines. The constraint here is that for two surfaces to be matched, their component edges must also be matched; this constraint is also applied to finer levels. Once a higher level match is confirmed, the components belonging to the matched features are no longer included in the search for other lower-level matches.

2.4 Regional/Surface-based Matching

Surface-based matching is similar to feature-based matching. It first segments the images, and then matches the segmented regions between the paired images. Birthfield and Tomasi [36] segment images into small planar patches determine the correspondence of those patches. This reduces the sensitivity to the depth discontinuities that occur in the paired images. However, these planes are likely to be slanted rather than face the cameras directly. The relationship between segments are modeled by six parameter affine transformations, such that

\[
\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = A \cdot \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + d,
\]

where \( A = \begin{bmatrix} 1 + d_{xx} & d_{xy} \\ d_{yx} & 1 + d_{yy} \end{bmatrix} \), and \( d = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \).

Equation 3

where \((x_1, y_1)\) and \((x_2, y_2)\) are the coordinates of corresponding points in the left and right image, respectively. Vector \(d\) defines the translation of a segment between frames; matrix \(A\) defines the in-plane rotation, scale, and shear transformations between frames. The parameters
can be computed directly from spatial-temporal intensity gradients. The result of this segmentation method is depicted in Figure 6.

![Figure 6](image)

**Figure 6** The test result of color segmentation based matching. (a) the reference image; (b) the ground truth image; (c) the result of color segmentation based matching

Cohen and Vinet [28] incorporate the hierarchical framework approach into regional based matching. For each image, they maintain a structure for the segments corresponding to different scales used in segmentation. Then the algorithm begins matching on the coarsest segmentation level [20]. If a match is confirmed, then the pair of segments is removed from the search list. The components of remaining segments will then be added into the search space, along with their unmatched parents, and matching is performed in the search space in an iterative fashion. The algorithm stops when there are no more segments left in the search space, or the segments cannot be divided further and cannot match to any other segments in the search space.

Bleyer and Gelautz [27] suggest a new algorithm that uses color segmentation to handle large un-textured regions and precise localization of depth boundaries. Each segment is assigned into a layer, which can be extracted by mean-shift-based clustering of depth planes. Z-buffering enforces visibility and allows the explicit detection of occlusions, and an efficient greedy algorithm searches for a local minimum of the cost function. Each time it adjusts only the depth estimation by one layer by consulting its neighbors’ depth information. Layer extraction and assignment are alternately applied.
H. Tao and H. Sawhney [29] suggest another alternative way that combines the global criterion and segmentation-based matching. They combine a color segmentation based representation and neighborhood depth hypothesizing method in a local search algorithm. More important than computational considerations, this approach enforces depth smoothness in homogeneous color regions and also makes it possible to infer reasonable depth for unmatched regions. A straightforward local greedy search algorithm is used in this method, and it tests all the neighboring depth hypotheses of each segment while all other segments are kept fixed. The neighborhood depth hypothesis that gives the best global matching score is recorded. After all segments have been tested, their depths are updated by choosing from the initial depth and the best neighborhood hypothesis according to the matching scores. The stereo correlation result is presented in Figure 7:

![Figure 7](image)

**Figure 7** The result of Tao and Sawhney’s algorithm [29]. (a) the left image (reference image); (b) the right image (paired image); (c) the final depth map from their algorithm

This process is performed iteratively until either the total number of segments with depth changes is small or the number of iterations exceeds a certain value. Once the image-disparity matching volume is computed, a plane is fitted for each color segment. An iterative fitting process is adopted to reduce the effect of outliers. The depth of every pixel is decided by picking the best matching score; then a plane is fitted in a segment. In the next iteration, the depth of each pixel is chosen within a given range of the fitted plane by finding the best matching score in that range. The plane parameters are updated accordingly based on this depth. This process
iterates several times until the plane parameters converge to a certain degree. To generate a hypothesis for a segment from a neighboring segment, the plane parameters are replaced using those of the neighboring segments. Then residual disparity for each pixel is found by searching around the plane and smoothing within the segment.

2.5 Dynamic Programming

Dynamic Programming [37], like the divide-and-conquer method [38], solves problems by combining the solutions of sub-problems. Dynamic programming is applicable when the sub-problems are not independent, that is, when sub-problems share sub-problems. In this context, a divide-and-conquer algorithm does more work than necessary, repeatedly solving the common sub-problems. A Dynamic Programming algorithm solves every sub-problem just once and then saves its answer in a table, thereby avoiding the work of re-computing the answer every time the sub-problem is encountered. Dynamic programming is typically applied to optimization problems, in which there are many possible solutions, and each solution has a value, and we wish to find a solution with the optimal (minimum or maximum) value. Such a solution is an optimal solution to the problem, and there may be several solutions that achieve the optimal value.

![Figure 8](image)

**Figure 8** The space-disparity map used in Dynamic Programming, intensities in the map represent the respective costs of potential matches. (a) The map uses left-right axes. (b) The map uses left-disparity axes.
In the stereo matching process, the epipolar monotonic ordering constraint allows the global cost function to be determined as the minimum cost path through a disparity-space image (DSI), shown in Figure 8. The cost of the optimal path is the sum of the costs of the partial paths, obtained recursively.

There are two ways to construct a DSI, shown in Figure 8. In Figure 8 (a), the axes are defined as the left and right scanlines, and dynamic programming is used to determine the minimum cost path from the lower left corner pixel to the upper right corner of the DSI. In Figure 8 (b), the axes are defined as the left scanline and the disparity range, and dynamic programming is used to determine the minimum cost path from the first column to the last column. The Intra-Scanline search is trying to find a path that gives us minimal cost. If we use DSI shown in Figure 8 (a), the matches will appear along the diagonals, and occlusions will appear along the horizontal and vertical lines along the minimal cost path.

We then can extend the algorithm into Inter-Scanline search [31]. Note that the points belonging to the same edge but on different scanlines should also be correlated to each other. The problem of obtaining a correspondence between edges under inter-scanline consistency constraints can be transformed into the problem of finding a set of paths in a 3D space composed by a stack of 2D planes for intra-scanline search. Figure 9 illustrates this 3D space. The side faces of the space correspond to the right and left images of a stereo pair. The cost of a set of paths is defined as the sum of the costs of the individual paths in the set. The goal is to obtain an optimal set of paths that minimize the total cost as inter-scanline constraints.
Figure 9 3D search space for the intra- and inter-scanline search by dynamic programming. The side faces correspond to the left and right images. Connected edges in each image form sets of intersections (2D nodes); each set is called a 3D node. Selection of a set of paths is done at every 3D node.

One of the principal advantages of dynamic programming, or any global search method, is that it provides global support for local regions that lack texture or would otherwise be matched incorrectly. These local regions present little difficulty for a global search since any cost function (e.g., intensity difference or variance) in these regions is low [19]. Another problem that global search seeks to resolve is occlusion. This is more difficult since a cost function applied near an occlusion boundary is typically high. Methods for dealing with such difficulty replace matching costs at occlusion boundaries with a small fixed occlusion cost.

The principal disadvantage of dynamic programming is the possibility that local errors may be propagated along the scanline, corrupting other potentially good matches. Horizontal streaks caused by this problem may be observed in many of the disparity map results reported in the literature, as shown in Figure 10.
Proposed by Tomasi and Manduchi [21], intrinsic curves use a different representation of image scanlines to do global matching. Intrinsic curves are defined by plotting the intensities of scanline pixels against their respective derivatives. This mapping ignores the translation between left and right image, therefore the mapping is invariant to the disparity differences. Thus in the ideal case, matching pixels should be mapped to the same points along a common intrinsic curve; however, in the general case, due to noise, occlusion, and projection distortion, matching pixels do not always map to exactly the same points, as shown in Figure 11:

The disparity search problem can be transformed into the intrinsic curve space, and the optimal solution is converted into finding nearest neighbors along two intrinsic curves. Ambiguities are resolved by maximizing a global metric using dynamic programming. Once
matching has been achieved in intrinsic curve space, and inverse mapping is applied to determine disparities.

The major advantage of the intrinsic curve approach is its invariance to disparity. The nearest-neighbor distances between points on two curves are not directly affected by the disparities between two image spaces. Therefore, it can be applied to the situation in which we do not know the initial estimate of the disparity value. More importantly, hierarchical multi-resolution levels of image can be used in intrinsic curves, which can reduce the computational cost without the serious impact of errors introduced in starting levels.

2.7 Graph Cuts

The Graph Cut method is also called the maximum flow method [22], and they are used to incorporate both horizontal and vertical continuity constraints.

![Figure 12](image)

**Figure 12** the 3D representation of maximum flow methods.

Let us define a graph $G = (V, E)$, where $V$ is the vertex set, and $E$ is the edge set. The vertex set can be constructed through disparity-space image (DSI). The graph axes correspond to the image’s horizontal and vertical axes and the disparity range. An edge is defined to be the link between two adjacent vertices, or the link from source to any point of zero disparity, or the
link from any point having maximum disparity to the sink. By stacking disparity-space images at different levels, we can have a 3D representation of maximum flow, as shown in Figure 12.

Internally, each node is six-connected with its neighbors, and each node has an associated cost which can be computed by dynamic programming, and each edge has an associated flow capacity that is defined as a function of the costs of the adjacent nodes it connects to. There is a limited amount of flow that can be sent from the source to the sink. The capacity is defined to be infinity for both the source and the sink. Maximum flow problems can be transformed into finding minimum cuts of the graph, which is analogous to the best path along a pair of scanlines determined by dynamic programming extended to tree dimensions [24]. Therefore, the disparity estimations associated with the minimum cut are not only consistent across one scanline, but also consistent globally throughout the image.

![Figure 13](image)

Figure 13 The test result of graph cut/maximum flow methods. (a) the reference image; (b) the ground truth image; (c) the result of maximum flow methods

2.8 Summary

All of the research in stereo correlation I reviewed in this chapter assumes that two images are taken by the cameras placed along the x-axis, and the objects in the scene have similar sizes and shapes. This assumption is also applied to most of the current research in stereo correlation, and it guarantees that two images can be rectified to let the corresponding epipolar
lines match with horizontal scanlines; the search of correct matches then can be performed along
the paired scanlines.

However, if two cameras are placed along the z-axis with the same orientation pointing to
the direction of the z-axis, the sizes and shapes of the objects in the scene will be changed
significantly, and they cannot be rectified to match along the scanlines. The previous assumption
no longer holds, and therefore most of current research in stereo correlation is failed to be
applied to such a scenario. This scenario is depicted in Figure 18, and it will be explained in
detail in the Chapter 4. To design an effective stereo correlation method to be applied to such a
scenario is the focus of this thesis, and this method is also discussed in detail in the Chapter 4.
3 IMAGE SEGMENTATION TECHNIQUES

Image segmentation subdivides an image into its constituent regions or components. The level to which the subdivision is carried depends on the problem being solved. Segmentation of nontrivial images is one of the most difficult tasks in image processing. Image segmentation can be performed at different scales and levels, and the level and accuracy of segmentation are critical to the stereo correlation method I designed.

3.1 Introduction

Image segmentation algorithms are generally based on one of two basic properties of the intensity values of an image: discontinuity and similarity [39]. Usually, pixels belonging to the same object have similar intensity values, and pixels belonging to different objects have different intensity values. The principal approaches of using intensity discontinuity are to partition an image based on abrupt changes in intensity, such as edges or boundaries of the objects in the image. Then an edge-linking process is gone through and boundaries of the objects are extracted. The principal approaches of using intensity similarity are to partition an image into regions having similar intensity values; thresholding, region growing, region splitting and merging methods belong to this category.

The discontinuities in grayscale images can be presented in different forms, such as points, line, and edges. There are many sophisticated techniques for detecting these three basic forms of grayscale level discontinuities, and edge detection in particular has been the core of segmentation algorithms for many years.
The most common way to detect isolated points in the image is to perform a high-pass filter with a mask shown in Figure 14, and the point to be detected is usually placed in the center of the mask [39].

Figure 14 The high-pass filter for detecting point discontinuities

Then the gradient value of that point is computed using Equation 4:

\[
G = \sum_{i=1}^{9} w_i g_i
\]

where \(w_i\) is the mask coefficient, and \(g_i\) is the grayscale value of the point and its neighbors. This high-pass filter is one form of Laplacian operator. Note that the absolute values of gradient of points in a uniform area are zero or very close to zero, while the absolute values of gradient of points around edges are usually much bigger.

To detect lines, we can use the combination of four masks, shown in Figure 15:

Figure 15 The high-pass filter for detecting line discontinuities. (a) Mask for detecting horizontal lines; (b) Mask for detecting vertical lines; (c) Mask for detecting \(+45^\circ\) lines (d) Mask for detecting \(-45^\circ\) lines.

The mask in Figure 15 (a) can be used to detect horizontal lines, since the maximum gradient value can be obtained when a single horizontal line comes across the middle row of the mask. A similar idea is applied to the masks in Figure 15 (b), (c), and (d), which can be used to detect vertical lines, \(+45^\circ\) lines, \(-45^\circ\) lines, respectively. The four masks can be applied to an
image individually. For any given point in the image, if the absolute gradient value computed by
a certain mask is the biggest, it means the point lies more likely on the line detected by that mask.
Therefore, to summarize the results from four individual mask operations, we can select the
maximum absolute gradient value and store it as the gradient value of a given point.

To detect edges in an image, we first take a look of the nature of edges. An edge can be
defined as a set of connected pixels that lie on the boundary between two regions. Compared to
a regional boundary, an edge is a local concept whereas a regional boundary is more a global
idea. Let us take a look at the formation of an edge, shown in Figure 16:

![Figure 16](image)

**Figure 16** The formation of edges; from top to bottom, the image regions containing the edge,
the 1D view of grayscale value of that region, the first-order derivative, and the second-order
derivative. (a) a perfect sharp edge; (b) an imperfect blurred edge.
Ideally, a perfect edge is a set of connected pixels, each of which is located at an orthogonal step transition in grayscale values, as depicted in Figure 16 (a). However in practice, due to the inaccuracy and imperfection of optics, sampling, and other image acquisition techniques and devices, the edges yielded are blurred, with the degree of blurring being determined by factors of the quality of the image acquisition system. The edge are more closely modeled as having a “ramplike” 1D representation, as the one shown in Figure 16 (b).

If we compute the first-order derivative and the second-order derivative of the 1D representation of the edge, we can find that in the first-order derivative view, the pixels located on an edge have larger positive gradient values, and the pixels not on an edge have gradient values of zero or very close to zero; in the second-order derivative view, the sign of the gradient values changes from one side of the edge to the other side of it. If we move from a dark region to a light region, the sign of the second-order derivative changes from positive to negative; if we move from a light region to a dark region, the sign changes from negative to positive. Therefore, we can not only detect the edges, but also characteristics of regions with second-order derivative gradient values.

3.2 Graph Based Method

We can use a global approach for edge detection and linking by presenting edge segments in the form of a graph and searching the graph for low-cost paths that correspond to significant edges. We can define a directed graph G=(E, V), where V is a set of nodes in the graph which represent the neighbor pixels in the image, and E is a set of edge segments between each neighbor pixels, with a cost assigned to each of them. To be consistent, an edge is defined as a sequence of connected edge segments. The goal of this method is to connect the edge segments
to form an edge, such that the total cost of this edge is minimal. The cost of each edge segment can be computed with **Equation 5**:

**Equation 5** \( c(p,q) = H - [f(p) - f(q)] \)

where \( H \) is the highest grayscale value in the entire image, and \( p, q \) are 4-connected neighbor pixels, and \( f(p), f(q) \) return the grayscale values of \( p \) and \( q \), respectively. To simplify the problem, we consider an image of size 3x3, and we will try find the edge of minimal cost from the top border of the image to the bottom border, depicted in **Figure 17**:

![Figure 17](image)

**Figure 17** (a) A 3x3 image with the grayscale value associated with each pixel; (b) the cost of each edge segment is marked; (c) an edge of minimal cost is found.

We first compute the costs of edge segments between neighbor pixels. There are two edge segments being computed between two neighbor pixels, from either direction. However, since we just consider the edges from top to the bottom, we only compute some of the edge segments we will use. After the costs are computed, an all-pairs-shortest-paths algorithm, such as Dijkstra’s algorithm [37], can be applied to look for the edge(s) with minimal costs. This method usually performs well in the presence of noise.
3.3 Thresholding

Objects in the image usually have different grayscale colors against the background. If we represent the image with a histogram, the background color can be displayed as one major peak in the histogram, and the colors of the objects can be displayed as other peak(s) in the histogram. Therefore, we can segment out the objects by their colors which can be located at the peaks in the histogram. The goal of locating peaks can be achieved by thresholding techniques.

There are many techniques for determining the threshold. The simplest one is to partition the image histogram by using a single global threshold, and segmentation is then accomplished by scanning the image pixel by pixel and labeling each one as object or background, depending on whether the grayscale value of the pixel is greater or less than the threshold. One way to determine the single threshold is described below:

1. Select an initial estimate threshold value, which can be an arbitrary value within the color range in the histogram
2. Separate the pixels in the image into two groups, where one group contains pixels with grayscale colors less than or equal to the threshold, and another group contains pixels with grayscale colors larger than the threshold;
3. Compute the average of the grayscale values of pixels for both groups;
4. A new threshold is then computed as the average of the averages of two groups;
5. Repeat steps 2 to 4 until the difference between the new threshold and the old one is smaller than a predefined parameter.

3.4 Region-Based Segmentation

There are two major techniques for region-based segmentation: region growing and region splitting/merging. Region growing is a procedure that groups pixels or sub-regions into
larger regions based on the similarities of pixels. The basic approach is to start with a set of “seed” pixels, and form initial regions with those seeds. Then the neighbor pixels are checked with the seeds, and those having similar properties to the seeds are grown into the regions. Selecting one or more starting seeds often can be based on the nature of the problems; however, if such prior information is not available, we can use a gradient image to find possible seeds. As we know, uniform areas possibly belong to the same regions; therefore, we can choose the pixels with minimal gradient values as seeds. To select the similarity criteria, we can use color properties for colored images, grayscale features for grayscale images, or/and the spatial properties such as moments or texture.

Region splitting and merging is the reverse procedure of region growing. Instead of finding seeds and growing regions, region splitting and merging subdivides an image into a set of arbitrary disjointed regions, and then merges and/or splits the regions based on the similarity criteria. The Quad-tree method can be applied in region splitting and merging. The image is first split into adjacent regions along the quad-tree, and the adjacent regions that meet the similarity criteria are merged.

### 3.5 Watershed Method

The Watershed method uses many of the concepts of the other approaches, and often produces more stable segmentation results, including generating continuous segmentation boundaries. The principal concept of the Watershed method is to visualize the gradient image in three dimension view, and to find the catchment basins in the image with dams surrounding them. This method will be described in detail in the next chapter.
4 STEREO CORRELATION PROCESS

4.1 Introduction

The majority of research in stereo correlation has focused on stereo images taken from two viewpoints placed along the x-axis. In that case, objects or regions in the scene are projected into two image planes with similar shapes and sizes. After the process of rectification, the epipolar lines of stereo images are aligned with scanlines, and we can assume there is no significant distortion along the y-axis. Therefore, by searching along scanlines, we can probably find correct matches for those non-occluded pixels or features; furthermore, by measuring the disparity along the x-axis, we can predict the depth information of those matched points. This is the fundamental assumption of scanline-search based correlation methods, such as dynamic programming methods, graph-cut methods, or feature-based methods. Meanwhile, the non-occluded objects will be projected into image planes through perspective projection without too much distortion on both the x-axis and the y-axis. Local window-based matching methods take this assumption and guarantee that, by measuring the differences between the intensities of the pixels in two windows of the same size on both images, we can find correct matches for non-occluded points. Again, we can still calculate the depth information of the matched points using their disparity values along the x-axis.

However, if we place two cameras along the z-axis in the scene, as depicted in Figure 18, objects are distorted substantially in their sizes and locations through perspective projection, although they still have similar shapes. In Figure 18, two cameras are placed at \( V_1 \) and \( V_2 \) in the scene, and \( AB \), \( CD \) are the front surfaces of two objects with different distances to the camera. \( AB \) is projected into \( A_1B_1 \) and \( A_2B_2 \) in the image planes of camera \( V_1 \) and \( V_2 \), respectively; similarly, \( CD \) is projected into \( C_1D_1 \) and \( C_2D_2 \) in the image planes of \( V_1 \) and \( V_2 \). The image
taken by $V_1$ is called the reference image (closer one), and the image taken by $V_2$ is called the paired image (further one).

![Diagram of stereo correlation](image)

**Figure 18** the fundamental concept of stereo correlation. $V_1$ and $V_2$ are two cameras along the z-axis in the same scene; $AB$ and $CD$ are the front surfaces of two objects, and $A_1B_1$ and $B_2B_2$ are the projections of $AB$ on the image planes of $V_1$ and $V_2$, respectively; similarly for $C_1D_1$ and $C_2D_2$ to $CD$. The image taken by $V_1$ is called the reference image (closer one), and the image taken by $V_2$ is called the paired image (further one).

Without losing generality, let us assume $AB$ and $CD$ are line segments; furthermore, let us assume that the projections of $AB$ and $CD$, $A_1B_1$ and $C_1D_1$, are aligned in the same scanline in the reference image. However, we cannot assume $A_2B_2$ and $C_2D_2$ still reside in the same scanline in the paired image. That is, if we rectify the paired images based on line $A_1B_1$ and $A_2B_2$, then $C_1D_1$ and $C_2D_2$ will not be in the same scanline. Therefore, the fundamental assumption used by the scanline-search based algorithms will not hold in such a scenario, and those correlation methods, such dynamic programming and graph-cuts, will fail to apply to this scenario.
Secondly, the assumption for local window-based matching algorithms will not hold either, since the objects vary substantially in size and location, and the errors can be significant in the measurement of differences between the intensities of the pixels in the correlation windows.

On the other hand, if we measure the scaling ratios from $A_1B_1$ to $A_2B_2$, and from $C_1D_1$ to $C_2D_2$ in Figure 18, we will notice that they shrink in different ratios, and $C_2D_2$ shrinks more from $C_1D_1$ than $A_2B_2$ from $A_1B_1$. In general, objects closer to the camera will shrink more than the ones further away when the camera moves back along the z-axis. We can derive the relationship between the depth information and the scaling ratio of paired objects, and we will do so later in this thesis.

Based on the above discovery, I perform stereo matching in object space. In my thesis, I only consider the situation of two stereo images, taken by the cameras at different locations along the z-axis. The orientations and other internal characteristics of the cameras are assumed to be the same; that is, the two cameras have the same focal lengths, and they also have the same orientations with the z-axis. The overall process can be summarized as follows:

1. Segment the reference image which is taken by a closer camera (in Figure 18, the image taken by $V_1$), and label the segments as components/objects. I use the image taken by the closer camera as the reference image since most of the pixels in this image will also appear in the paired image taken by the further camera; however, if we take the image taken by the further camera, many pixels in it will be occluded in the other one.

2. Use the information of each component obtained in step 1 to guide the segmentation process on the paired image (in Figure 18, the image taken by $V_2$). The information of a component includes its spatial position, and its color distribution, etc.
3. By using the combination of different parameters, we may find a set of components in the paired image that can be possibly matched with the reference component. Each possible pair is then added into the candidate matches space.

4. Due to the consideration of performance, a greedy strategy is applied to choose an optimal solution from the candidate matches space, and the solution contains a set of matches. The matches shouldn’t conflict with each other (or within a tolerable range of overlapping).

5. A depth image is constructed from the optimal solution, and the depth information is computed from the ratio of the sizes of paired components.

4.2 Segmenting on Reference Image

As stated in the previous section, the stereo correlation method I designed is performed in the object space. To do so, I use image segmentation techniques to segment and extract objects from the reference image. Compared to other segmentation techniques, the Watershed method usually generates more robust results, and I use it to do the segmentation on the reference image. The segments then are labeled as components / segments, and for each of them, I will try to find zero or more possibly matched components in the paired image. One way to perform such matching is to apply segmentation on both images, and then try to match the extracted components from both images. However, due to the large variation of object sizes and noises which may occur frequently in real images, the direct segmentation on both images will not be able to give us the same results on the same objects; and the matching process will give us more mis-matches than correct matches.

An approach is proposed here to address this problem, which uses the color similarities of the matched components in both images to guide the segmentation process. Instead of
segmenting both images directly, I only apply the segmentation process on the reference image, and then use the spatial information and color distribution of each extracted component, to guide the segmentation process on the paired image. I just need to segment a certain part of the paired image by applying those criteria. After a set of components are extracted from that region, possible matches between the reference component and these components are computed and added into the candidate matches space.

One of the principal applications of watershed segmentation is in the extraction of nearly uniform objects (bloblike) from the background. Regions characterized by small variations in intensities have small gradient values, and the Watershed method use this characteristic of gradient image to do the segmentation. To segment the reference image, I first generate a gradient image from the original color image, and then apply the Watershed method onto it. At each flooding stage of the Watershed method, I use the component labeling method to extract components, and build dams between merging components.

4.2.1 Creating Gradient Image

Instead of storing the intensity values, gradient images contain the variance information of the intensity values of an image. In my case, I use a first-order differential equation to derive gradient values. Regions with significant changes in intensity values have higher gradient values in the output, and regions containing uniform intensity values have lower gradient values. As we know, those regions with significant changes usually refer to the boundaries of the objects in the scene, and those uniform regions usually refer to the bodies of the objects. By segmenting along the boundaries, i.e. segmenting along the regions with higher gradient values, we can effectively extract the objects out of the scene.
To generate the gradient image from the reference image, we can first convert the color image into a grayscale image, and then compute first-order differences of the gradient intensities. There are two major approaches to performing the color-to-grayscale conversion. One way is to use the average values of the red, green, and blue part of the pixels; another is to use the maximum values among these three parts as the grayscale intensities. However, we lose some valuable information by converting from color image to grayscale image, no matter which converting method we use.

In my approach, I do not convert the color image into a grayscale one; instead, I split the image into three sub-images using the color parts of the pixels individually. For example, one image just represents the red parts of all pixels; similarly, one for blue, and one for green. Then I apply the gradient operation on each sub-image, and at last, I combine the three gradient images together by picking up the highest gradient value for each pixel. This approach preserves more color variance information in the final gradient image, and can generate more accurate results.

The Roberts Operator is used to derive the first-order gradient values. This operator provides a simple approximation to the gradient magnitude using **Equation 6**:

\[
G[f(x,y)] = |G_x| + |G_y| = |f(x,y) - f(x+1,y+1)| + |f(x+1,y) - f(x,y+1)|
\]

where \(G_x\) and \(G_y\) can be computed with the following masks:

\[
G_x = \begin{bmatrix}
1 & 0 \\
0 & -1 \\
\end{bmatrix}
\quad\quad\quad G_y = \begin{bmatrix}
0 & -1 \\
1 & 0 \\
\end{bmatrix}
\]

The Roberts Operator computes the differences at the interpolated point \((x+\frac{1}{2}, y+\frac{1}{2})\), and is an approximation to the continuous gradient at that point and not at the point \((x, y)\) as might be expected.

After the final gradient image is generated by combining the gradient images from the three sub-images, a median filtering mask is then applied onto it. In my case, I choose a 3x3
window as the median filtering mask. Since we use the Roberts Operator to generate a gradient image with 2x2 window, the one-pixel-wide border in the original image usually generates about 2-pixel-wide highlighted area in the gradient image, and in a 3x3 window, the highlighted area usually cover 6 pixels out of 9 pixels. Therefore, by using 3x3 median filtering, we can remove small noise pixels effectively, and still preserve the highlighted boundary areas at the same time.

4.2.2 Applying the Watershed Method

As we stated before, the Watershed method incorporates many concepts of other approaches, such as thresholding, region processing, and discontinuity detection. It usually produces more stable segmentation results, including continuous segmentation boundaries. More importantly, this approach also provides a simple framework for incorporating knowledge-based constraints in the segmentation process.

The basic concept of the Watershed method is to visualize a gradient image in three dimensions: two spatial coordinates versus gradient levels along the z coordinate. If we imagine placing a drop of water on a point in the three-dimensional image, there can be three possible situations. The drop of water could stay and accumulate on some points, or it could fall with certainty into a single minimum on some other points, or it could equally likely fall into more than one such regional minima. We can think about the three-dimensional topographical view as the terrain with several basins on the land, and the regional minima are the bottoms of the basins. Points that would make the drop of water fall into a single minimum are called the catchments basin or watershed of that regional minimum, and the points that make the drop of water fall into more than one regional minimum are called divide lines or watershed lines.
Figure 19 The principal concept of dam construction in the Watershed method. (a) punching a hole at the regional minimum; (b) flooding the topography from below by letting the water rise through the holes with a uniform rate; (c) if two catchments basins are about to merge, a dam is built between them to prevent the merging; (d) after the maximum gradient level is reached, all points except the dam points are flooded, and all dams are built.

The principal goal of this segmentation algorithm is to discover the watershed lines, which represent the boundaries of the objects. As illustrated in Figure 19, the idea of this algorithm is fairly simple: suppose we first make a hole on the regional minimum of all the basins in the topological view, as shown in Figure 19 (a), and then we flood the basins from below by letting the water rise through the holes at a uniform rate. We can use gradient values to represent the flooding rate, and they are displayed as the heights of each pixel in the topography, as shown in Figure 19 (b), (c), and (d). Flooding from the minimal gradient to maximal one incrementally, at each flooding level, we just flood the points that have equal or smaller gradient values than that level. If distinct catchments basins are about to merge, a dam is constructed to prevent the merging, as shown in Figure 19 (c). The flood eventually reaches a stage at which
only the dams are visible above the water lines, as shown in Figure 19 (d). The dam boundaries correspond to the watershed lines, and they are used to extract the components/objects from the reference image.

To construct the dams between catchments basins, we use morphological dilation techniques, as shown in Figure 20. The idea is that at each flooding level, we use an 8-connectivity window to expand a basin, depicted in Figure 20 (b).

First, I trace along the contour of the component to be expanded, and check the neighbor points outside of the contour using an 8-connectivity window. The point to be checked is placed in the center of the window, and the possible situations are:

- If there is at least one neighbor point coming from other components, then this point is marked as a dam point;
- If there is no neighbor coming from other components, and if the gradient value of this point is equal or less than current flooding level, then the component is expanded to include this point;
- Otherwise, the point remains unchanged.

Figure 20 (a) depicts the result from flooding the topography to level n-1, and we obtain two components, shown in black, by flooding the two catchment basins. We then raise the flooding level to n, and both of the components are expanded. The adjacent regions with equal or lower gradient values than n are flooded and merged into the catchment basin, as depicted in gray and light gray colors in Figure 20 (b), (c), (d).
Figure 20 Using morphological dilation to construct dams at flooding step n. (a) two components are found at flooding level n-1; (b) 8-connectivity window for dilation; (c) At flooding level n both components will be expended; this is the first round of dilation; (d) dams are built to prevent merging; (e) another round of dilation in level n; (e) more dams are built.
Figure 20 (c) through (f) show the dilation process at the flooding level \( n \). In Figure 20 (c), we first walk along the contour of each component, and expand it by one pixel using an 8-connection window, as shown in gray colors in the figure. The dilation is only applied to the neighbor pixels with equal or lower gradient values. By expanding the component by one pixel each round, I guarantee that each component has equal chance to expand to its neighbor regions. After the first round of dilation is done, two components are now connected with each other, as shown in Figure 20 (c). To prevent this from happening, I mark four points as dam points (the crossed blocks), as shown in Figure 20 (d), and assign a dam value to them, which is the maximal gradient value plus one. After all components are expanded by one pixel in the first round, a second round of dilation begins, as shown in light gray colors in Figure 20 (e). More dam points are marked to prevent the component from merging, as shown in Figure 20 (f). Eventually, all the adjacent pixels with equal or lower gradient values than the flooding level are flooded or marked as dams.

Assume we begin the flooding level from the minimum gradient value to the maximum gradient value. At the flooding level \( n-1 \), let \( C(n-1) \) denote the set of components (catchment basins) extracted from that level, and \( c(n-1,i) \) is one of the components. Taking the example in Figure 20 (a), we call the two components as \( c(n-1, a) \) and \( c(n-1, b) \). At the flooding level \( n \), we first convert the gradient image into binary image by marking the points with equal or lower gradient values than \( n \) as white, and leave other points as black; then we label the components in the result binary image. Let \( Q(n) \) denotes the set of components we get from the binary image, and \( q(n, i) \) is one of them. In Figure 20 (e), the whole piece of black, gray, and light gray regions (including dam points) is labeled as one component. Let it be \( q(n, a) \). We consider the
component $c(n-1, i)$ as a set of coordinates of points belonging to that component; and the same for $q(n, i)$. From Figure 20 (e), we can find three important properties of $C(n)$ and $Q(n)$:

- For each component $c(n-1, i)$ found in flooding level $n-1$, there is one, and only one, superset of it existing in $C(n)$; that is, a component $c(n, j)$ contains all the points in $c(n-1,j)$;
- A set $q(n, i)$ either equals to a component $c(n, i)$, or is a superset of two or more $c(n, i)$, but any point in $q(n, i)$ cannot belong to two or more $c(n, i)$ at the same time.
- If the set $q(n, i)$ contains two or more subsets (or components) from the previous flooding level, it means those components are going to merge in the flooding level $n$, and dams must be built between them to prevent the merging.

Based on the above discovery, I developed an efficient algorithm of watershed method. The process of the algorithm is:

1. Initialize the flooding level at the minimal gradient value;
2. Find marker candidates in the smoothened gradient image on the flooding level $n$. A marker is a region in the image that has some uniform characteristics, such as similar gradient values, similar texture information, etc. In my case, I use the uniform gradient values as a criterion in finding markers. A marker is a region with gradient values below the points surrounding it, and it must meet a certain minimal size requirements. By choosing smaller minimal size, usually we may get more markers with less fluctuation within each of them; however, the result may appear to be over-segmented. On the other hand, if we choose a bigger minimal size threshold, we may get fewer catchment basins, but the fluctuation within each one increases, and the result may appear to be under-segmented. Markers can be found on different
flooding levels. To extract a marker on a given flooding level $n$, I first convert the gradient image (as a grayscale image) into a binary image based on the flooding level $n$; that is, I mark pixels with equal or lower gradient values as white, and I mark other pixels as black. I then apply a component labeling method on the binary image to extract the components. $Q(n)$ denotes the set of the components I extract, and those components are the candidates of markers. This process is performed repeatedly at each flooding level.

3. After we extract marker candidates at the flooding level $n$, I compare those components in set $Q(n)$ with the components in set $C(n-1)$, which is the set of components representing catchment basins found in the flooding level $n-1$. For each component $q(n, i)$, I check how many components in $C(n-1)$ it contains. Based on the discoveries stated previously, if a component $c(n-1, i)$ is a subset of a component $c(n, i)$, and $c(n, i)$ is a sub-set of $q(n, i)$, then $c(n-1, i)$ is also a subset of $q(n, i)$; that is, $c(n-1, i)$ is only overlapped with one $q(n, i)$. It is sufficient to check only one pixel on component $c(n-1, i)$, and if that pixel also belongs to the component $q(n, i)$, then the component $c(n-1, i)$ is a subset of $q(n, i)$. There are three possible relationships between $c(n-1, i)$ and $q(n, i)$:

- If $q(n, i)$ contains none of the components from $C(n-1)$, then $q(n, i)$ is a new marker found on flooding level $n$, and it is put into $C(n)$;
- If $q(n, i)$ contains exactly one component from $C(n-1)$, then $q(n, i)$ is an expanded version of $c(n-1, i)$; $q(n, i)$ is put into $C(n)$ to replace $c(n-1, i)$;
- If $q(n, i)$ contains two or more components from $C(n-1)$. In this case, those components are about to merge, and some dams must be built to prevent this from
happening. Those components in \( C(n-1) \) will be input into the dam building process I described before to construct the dams, and then the dilated components are put into \( C(n) \).

4. Increase the flooding level to \( n+1 \), and iterate back to step 2, till the flooding level exceeds the maximal gradient value.

5. Now all dams are built, and the gradient values of dam points become maximal gradient value plus one. The components in \( C(n) \) are final segmentation result of the Watershed method.

### 4.2.3 Labeling Components

The component labeling algorithm is used to assign a unique label to the pixels within the same connected component, and then extract those components from the binary image. I developed an algorithm to perform the process efficiently.

The principal concept is a two-phase scan on the image [40]. The first phase of image scanning gives each foreground pixel a label, and it also marks the labels belonging to the same component as equivalent labels. Then a label refining process is performed to merge the equivalent labels; after that, the second scan is processed to assign the refined labels to each pixel, and output the result component map.

![Figure 21](image.png)

**Figure 21** The first scan of component labeling. The gray points are the ones being checked, and the algorithm only checks the upper and left neighbors of the point. (a) none of the neighbors has a label, and a new label is assigned to the gray point; (b) only the upper neighbor has a label, and that label is assigned; (c) only the left neighbor has a label, and that label is assigned; (d) both of the neighbors have the same label, and that label is assigned; (e) two neighbors have different labels, then the upper label is assigned, and the equivalent label handling process is invoked.
In the first scan, the algorithm goes from left to right, top to bottom, and for each foreground pixel, it checks its upper and left neighbor pixels, and assign a label to it, as shown in Figure 21. If none of its neighbors has a label assigned, the current point is assigned a new label; if only one of the neighbors has a label, then the current point is assigned with the same label of its neighbor; if both of the neighbors have labels, and the labels are the same, then the current point is also assigned with the same label; if the labels of its neighbors are different, the label of the upper neighbor is assigned to the current point, and an equivalent label handling process is invoked to set up the relationship of those two different labels of its neighbors.

Figure 22 The directed graph is used to store the equivalent labels. The equivalent labels handling process will try to group equivalent labels as trees, and meanwhile try to minimize the height of each tree to 1. Finally the graph will become a forest with trees of height 0 or 1, and the number of trees are the number of labels we will use in the output label map, and each root node represents the minimal labels of a group of equivalent labels in that tree.

I used a directed graph to accelerate the equivalent labels handling process. Each node of the graph is a data structure that contains a label represented by the node, a minimal label that node is equivalent to, and a list of its children labels equivalent to this node, as depicted in Figure 22. The children list is empty if the node doesn’t represent a minimal label; that is, the node’s label is same as its minimal label. Those nodes are shown as the roots of each tree in Figure 22. At the initial step, each node has a minimal label the same as its own label, and an
empty list of children labels. For each equivalent label pairs, the process first compares the minimal equivalent labels associated with the node of those paired labels, and gets the node of the smaller minimal label, called node $S$, and gets the node of the larger minimal label, called node $L$. Then we replace the minimal labels of $L$ and all of its children nodes of $L$ to the label of $S$, and add $L$ and its children into $S$’s children list, meanwhile, clear the children list of $L$. After the first scan, all pixels are assigned with a label, and a directed graph is built, with each node associated with an equivalent minimal label; what’s more, each node of those minimal labels contains a list of its children labels that is equivalent to that minimal label. In this way, we can traverse the graph in both directions.

Then the algorithm goes into a label refining step. New labels are calculated based on the total amount of minimal label used, and for each minimal label, a new sequential label is assigned. After a new minimal label is associated with each label in the graph, a second scan on the image is performed, and each pixel is assigned by the new minimal label. The labeling map is then created, and each region is represented by a component object.

All the components share the same labeling map to reduce the memory requirement. Besides the labeling map and a label assign to the component, a component has the following properties:

![Figure 23](image.png) the properties of a component; (a) a component labeled in the image; (b) basic perimeter = 18; (c) basic size = 23; (d) external perimeter = 14; (e) external size = 26.
• Basic Perimeter: is counted as the number of points on the component, and those points are adjacent with other points not on that component using a 4-connectivity window, or adjacent with border of the image

• Basic Size: is counted as the number of points on the component.

• External Perimeter: to calculate external perimeter, I do a contour track using an 8-connectivity window on the outer boundaries of the components. By doing this, I pretend the component is a fully filled region without holes in it. The contour track has a similar effect to a 4-connectivity window, and finds out the points on the outer boundaries.

• External Size: is counted as the number of points within the external perimeter of the component, including the points on the external perimeter. By doing this, I also pretend the component is a fully filled region without holes.

• Elongation: is computed by the ratio of height and width of the minimal rectangular region that can contain the component.

• Compactness: compactness is computed by the ratio of square of the external perimeter and the external size.

After all components are extracted, and their properties are computed, a minimal size constraint is applied on the external size property, and the components with a external size less than the constraint are removed, and labels used by them are released. At last, a set of components are output as the result of component labeling process.

4.3 Searching for Paired Components

As stated before, most of the pixels in the reference image will also appear in the paired image; therefore, there exists a region in the paired image that can be correctly matched with
each reference component extracted from the reference image. Assuming we can extract that region from the paired image as a paired component, I will define some matching criteria depending on the characteristics of matched components.

The first and most important characteristic is the pixel similarity. Two correctly matched components should contain similar pixels in the relative spatial context. This characteristic is widely used in most of stereo correlation researches, and the usage of the local correlation window is based on this characteristic. However, two stereo correlation windows with the same size cannot be applied directly onto the situation of this paper since regions are scaled significantly in different ratio between paired images, and the correlation window will have different sizes. Instead, I will first shrink the region covered by the reference component to the size of the paired component. Since we know that reference components are equal or bigger than paired ones, I can preserve more information by shrinking the reference component than by expanding the paired component. The image resizing is done by Java built-in facilities. A correlation window is then used to cover the whole region, and only the pixels on the positions covered by both components are measured. The ZSAD (Zero mean sum of absolute differences) method is used to compute the pixel similarity score:

\[
\text{Equation 7} \quad P = \left( \sum_{(u,v) \in W} \left| (I_1(u,v) - \overline{I}_1) - (I_2(u,v) - \overline{I}_2) \right| \right) / 255
\]

where \( u \) and \( v \) are the relative coordinates of pixels in the region, \( W \) is the set of pixels covered by both components, \( I_i(u, v) \) is the value of a color component of the pixel at position \((u, v)\). The score is computed for each of the three color components and it will be scaled into the range of [0, 1], since we know the largest possible difference is 255 for each color component. Then the average of the three scores is calculated as the final score as the pixel similarity score between the reference component and paired component. Computing the pixel similarity score is
an expensive operation since it involves resizing an image; therefore, this computation is performed after other threshold constraints which can reduce the number of components used for matching.

The second characteristic is the shape similarity. Again, two correctly matched components should have similar shapes, and this can be measured by the elongation property of the components. A difference of elongations is computed using following formula:

**Equation 8** \[ E = \left( \frac{|E_r - E_p|}{T_E} \right) \]

where \( E_r \) is the elongation value of the reference component, \( E_p \) is the elongation value of the paired one, and \( T_E \) is a heuristic threshold parameter. \( T_E \) will be assigned by the user, and the matches with larger elongation differences than the threshold are rejected. The final elongation difference is then scaled into the range of \([0, 1]\) according to the elongation threshold parameter. This threshold is applied before the previous pixel similarity computation.

The third characteristic is the similarity of the compactness values. As I stated in the previous section, I use the external perimeter and external size to compute the compactness value of a component, and thus avoid the interferences of noises and holes. A difference of compactness is computed using following formula:

**Equation 9** \[ C = \left( \frac{|C_r - C_p|}{T_C} \right) \]

where \( C_r \) is the compactness value of the reference component, \( C_p \) is the compactness value of the paired one, and \( T_C \) is the threshold parameter assigned by the user. Similar to the threshold for the elongation difference, this threshold parameter rejects the matches with larger compactness differences than it, and thus is able to reduce the set of components for the pixel similarity computation. The final compactness difference is also scaled into the range of \([0, 1]\) according to the compactness threshold parameter.
After calculating the above three characteristics, I then define the overall matching score between two components through a linear combination of them:

**Equation 10** \[ M = \frac{(w_p \times P + w_E \times E + w_C \times C)}{(w_p + w_E + w_C)} \]

where \( w_p, w_E, w_C \) are the weights for pixel similarity score, elongation difference, and compactness difference, respectively. Those weights are determined by the user, and passed in as parameters; usually the pixel similarity score should have higher weight than others. The value of matching score \( M \) is scaled to a range of \([0, 1]\). The smaller the score, the better the match it represents. In the ideal situation, if two components match with each other perfectly, the score will be zero.

After we defined how to compute the matching scores between components, we will then focus on how to find the matched components. A naive approach is first to segment both images, and then to try all possible matches among two sets of components. However, we tend to over-segment the image to minimize the probabilities of one component containing more than one surface, and it usually ends up with hundreds to thousands of components for an image of size 500*400. If we compute the matching score on each component from the reference image against each component from the paired image, there would be more than millions of scores to be computed. Instead, I make use of the spatial relationship between the reference component and the paired component, and only allow the reference component to match with other components located in a certain region.

To describe the spatial relationship, let us first assume the origins of the image planes of both paired images reside at the center of the images, and the x-axis of both point to the right. Let us assume there is a point \( A \) in the reference image, and the corresponding point in the paired image is \( B \). If we superimpose the two images together with both origins overlapped and x-axes
pointing to the same direction, we can obtain an image showing in Figure 24. If we draw a linear path from $A$ to $B$, we can discover the path that leads toward to the origin of the superimposed image. This is true because if we try to move the camera to an infinite distance, all the points in the scene converge to the origin. Furthermore, since the $x$ and $y$ coordinates have the same scale ratio, it guarantees that $A$ moves to $B$ in a line towards the origin. In other words, for every point $A$ in the reference image, the correct corresponding point $A'$ in the paired image will reside in the rectangular region $ACOD$, as depicted in the Figure 24 (a). If we consider points $A \ (x_1, y_1)$, $B \ (x_2, y_2)$ as the left-top and right-bottom corner of the rectangular region, respectively, that covers a component in the reference image, shown as the gray region in Figure 24 (b), the search space takes place in the following rectangular region: $(\min(x_1, 0), \ min(y_1, 0)) – (\max(x_2, 0), \ max(y_2, 0))$. Only the components residing in this region are possibly matched with the reference component. By doing this, I can reduce the size of searching space significantly, and hence improve performance.

![Figure 24](image)

**Figure 24** the superimposed images. (a) point $A$ in the reference image corresponds to $A'$ in the paired image, the path from $A$ to $A'$ heads to the origin. (b) The search space of component $AB$ is in gray shade, where corresponding component $A'B'$ may reside.

After figuring out the search window, we can perform an image segmentation process on the region of search window to extract a set of components that are possibly matched with the
reference component. However, instead of segmenting the image region within the search region, I use color characteristics to extract components since correctly matched components should have the same or similar color distribution. First, I compute the color distribution of the pixels covered by the reference component, that is, the standard deviation of the pixels. I compute one standard deviation for each of the three color components, denoted as $S_r$, $S_g$, and $S_b$; and I also obtain the average value for each of the color components, denoted as $A_r$, $A_g$, and $A_b$. Then in the search window in the paired image, I use these color distribution constraint to pick up pixels; that is, the pixels with all of their three color components meet the following condition:

\[
\begin{align*}
    A_r - m \times S_r & \leq V_r \leq A_r + m \times S_r \\
    A_g - m \times S_g & \leq V_g \leq A_g + m \times S_g \\
    A_b - m \times S_b & \leq V_b \leq A_b + m \times S_b
\end{align*}
\]

where $V_r$, $V_g$, $V_b$ are the values of the red, green, and blue parts of a pixel in the search window, respectively. The variable $m$ is a multiplier, and it is changed from 0.5 to 3, with step-size of 0.5. For each given $m$, I can obtain a set of pixels in the search window from the paired image that meets Equation 11. Then a binary image is generated by converting those pixels into white, and converting other pixels into black. I pipeline this binary image through a component labeling process, and extract zero or more components that are possibly matched with reference component. By adding up all possibly matched components under different values of multiplier $m$, I can obtain a set of possibly matched components for each reference component.

4.4 Generating an Optimal Solution

The computed scores from the previous step are stored in an external hash table data structure, where the key for a cell is a reference component, and there is another hash table stored as the value of the cell. The child hash table contains all possibly matched components with their matching score pairs. It is not possible to use all matches in the output, since some of
the matches are exclusive to each other, and many of them are false matches. Therefore, we need to pick up a subset of matches from all possible combinations. The goal is to pick up as many correct matches as possible, and meanwhile to avoid conflicts. As we know, correct matches tend to have smaller match scores, so the guideline is in favor of matched components with smaller scores.

We know that the exclusive component matches cannot be picked at the same time. First of all, we know the reference components are extracted by segmenting on the reference image. Therefore, the reference components will not be overlapped with each other. But this is not true for the paired components on the paired image that match with the reference components; we don’t use image segmentation to extract those paired components, and they may be overlapped with each other. Therefore, some of the paired components cannot co-exist simultaneously. That is, those paired components are exclusive to each other. There are two types of exclusion: absolute exclusion and partial exclusion. Absolute exclusion happens among the paired components that match with the same reference component. Given a reference component, it cannot be fully matched with more than one region at the same time; that is, in the external hash table data structure, if we pick one pair of matched components (one reference component and one paired component), then all other paired components stored in the cell of that reference component key are absolutely exclusive from this match even if they do not overlap with the paired component we just picked up; and thus are removed from the matching pool for further selection. Partial exclusion happens among the paired components that match with different reference components. In that case, they may be overlapped with each other. To avoid the problem of over-excluding good matches, I use a tolerance threshold; that is, unless the ratio of the number of the overlapped pixels over the number of all pixels in each component exceeds a
certain threshold, the overlapping is considered tolerable, and the two components are considered being able to co-exist with each other. The threshold is usually chosen as 10%, and if the ratio exceeds the threshold, then the two components are considered exclusive to each other, and cannot be picked up in the same solution.

A Greedy strategy is used to choose good matches from all possible ones. The match with the highest score is first chosen from the candidate pool, and the absolutely exclusive matches are removed from the pool. This can be done easily with the external hash table data structure, since we just need to remove a cell with the reference component as the key. Then a overlap-checking process goes through the remaining candidates, and those partially exclusive matches are removed from the candidate pool. The algorithm keeps iterating on the candidate matching pool, choosing the match with the highest score, and removing the exclusive matches from the pool. The process keeps working until no more matches are left in the candidate pool, and a solution is constructed with the set of the pairs of matched components.

4.5 Constructing Depth Image

Using the solution obtained from the previous step, we are able to reconstruct the 3D scene, and generate a depth image. A depth image has the same width and height as the reference image, but instead of storing the color information of a pixel at each \((x, y)\) location in the reference image, the value at each \((x, y)\) location in the depth image represents the depth information from that point to the camera. To be consistent with the reference image, the depth image is in the plane formed by the x-axis and y-axis, where the origin is placed in the center of the depth image, and the values in the depth image are the z coordinates of the assigned points.

As described in the beginning of this chapter, if a component is closer to the camera, the size of it changes more in two cameras (shrinks more from closer camera to further one); and the
component further from the camera changes less in its size. As we know, the paired components in the solution can be considered as the regions in the paired images which are projected from the same surface in the scene. Therefore, by measuring the ratio of the sizes of the paired components, we can possibly find out the distance information from the camera to that surface. To derive the relationship between the ratio of sizes of paired components and the depth information of the points covered by the components, we will first look at one simpler situation – the relationship between the ratio of line length and the z-coordinates of the points on the line, as depicted in Figure 25.

**Figure 25** the relationship between the ratio of the projected length of a line $AB$ and the z-coordinate of this line. The two cameras are placed along the z-axis at $V_1$ and $V_2$, and focus at the same direction with the same focal length $f$. The baseline between two cameras is denoted as $T$. The line $l$ is projected into the closer image place as $n$, and projected into the further image plane as $m$. We can derive the z coordinate by the ratio of the lengths of $m$ and $n$.

Two cameras are placed at $V_1$ and $V_2$ in Figure 25, and they have the same focal length $f$. The distance between $V_1$ and $V_2$ is defined as the baseline, denoted as $T$. By the definition, the image at axis $Y_2$ is the reference image, and the one at axis $Y_1$ is the paired image. Assume there
is a line segment \( l \), we get the projection of line \( l \) in the reference image as \( n \), and in the paired image as \( m \). From Figure 25, we can obtain following equations:

\[
\begin{aligned}
\text{Equation 12} & \quad \begin{cases}
\frac{f}{n} = \frac{f + z}{l} \Rightarrow f \times l = n \times (f + z) \\
\frac{f}{m} = \frac{f + z + T}{l} \Rightarrow f \times l = m \times (f + z + T)
\end{cases}
\end{aligned}
\]

Combining the above two equations, we get \( n \times (f + z) = m \times (f + z + T) \)

Therefore, we get the relationship between the ratio of lengths of projections and the distance information of line \( l \) as:

\[
\text{Equation 13} \quad \frac{n}{m} = \frac{f + z + T}{f + z} = 1 + \frac{T}{f + z}
\]

From another point of view, we can consider the line \( m \) and \( n \) as the heights of a paired of rectangular components. If the ratio of the heights follows the above formula, the ratio of the widths of the paired components should also follow it. Let \( s \) and \( t \) denote the sizes of the reference component and the paired one, respectively, we can then derive the following equation:

\[
\text{Equation 14} \quad \frac{s}{t} = \left( \frac{n}{m} \right)^2 = \left( 1 + \frac{T}{f + z} \right)^2
\]

Re-arrange the above equation, finally we get:

\[
\text{Equation 15} \quad (f + z) = \frac{T}{\sqrt{\frac{s}{t}} - 1}
\]

Using this formula, I can compute the distance in the direction of the \( z \)-axis from the camera at \( V_2 \) to the points covered by the reference image. The algorithm iterates on the set of selected component pairs from the previous step, and for each pair, I use the external sizes of the components to compute the distance information. After the distances of all available points are computed, the result depth image is generated and saved on to disk.
5 EXPERIMENTS

In this chapter I present the experimental results that are generated by my implementation, and the technical details of the implementation as discussed in the previous chapter. My implementation is written in java, and the Java Advanced Imaging™ (JAI) and Java 3D™ packages are used. The implementation contains two major parts, the Java 3D scene viewer, and the implementation of the stereo correlation algorithm. The 3D scene viewer presents a scene with two cameras being located along the z-axis, and the virtual scene is generated with Java 3D™ [41]. The viewer also provides a function to grab snapshots of the scene from both cameras, and save them into image files. The algorithm implementation provides most necessary functions such as image color conversion, component labeling, image segmentation, stereo correlation, except the color image resizing which is provided by Java Advanced Imaging™.

5.1 Introduction

There are two categories of .image resources used in my experiments, the artificial images generated from the 3D scene viewer, and the photos of the real scene. The key criterion is to segment out appropriate regions from both images, which is used in the matching process. As we can see later, the segmentation implementation works well for artificial images, since the cameras are placed perfectly along the z-axis, looking at the same direction along the z-axis, and the noise introduced by the image acquisition devices do not exist in artificial images. Therefore, without the interference of incorrect segmentations, we can focus on the stereo correlation portion of my implementation. The correlation process generates good correlation results on artificial images with correct segmentation.
The experiment first presents the results of image segmentation from various image sources; then a screenshot of my Java 3D scene viewer is posted in the following section; in the last section, the correlation results from different sets of stereo images are listed.

5.2 Image Segmentation Experiment Results

The image segmentation technique I implemented is the Watershed method. As I explained before, the process is able to segment the image in different levels and scales, depending on the choice of minimal size of components.

![Image Segmentation Results](image1.png)

Figure 26 The segmented artificial images and the segmentation maps. (a) minSize=100; (b) minSize=500; (c) minSize=1300; (d) minSize=1900; (e) minSize=2500

As I depicted in Figure 26, if the minimal size of components is small, there are more regions to be segmented from the image; if we increase the minimal size threshold, the number of segments reduces; when the minimal size is big enough, there will be no segment extracted from the image.

Following are the results of segmentation experiment on real photos:
Figure 27 The segmented “bean” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=1500

Figure 28 The segmented “scene” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=2500
Figure 29 The segmented “cloud” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2900

Figure 30 The segmented “stone texture” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2700
Figure 31 The segmented “x-ray” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2700

Figure 32 The segmented “yosemite” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2700
Figure 33 The segmented paired “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=1500

Figure 34 The segmented reference “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=500; (c) minSize=2500
In Figure 33 and Figure 34, two images of the same scene are presented. The image in Figure 33 is taken by a camera that is further away from the scene, and the image in Figure 34 is taken by the same camera that is closer to the scene. Since there are noises and errors introduced by the image acquisition devices, the segmentation results are different even in the same region of the scene. More importantly, the objects are scaled differently based on their distances to the cameras, and the minimal size thresholds to be used to segment the object region should be also different in the stereo image pairs. For example, the cookie box on the right side of the scene in the paired image is segmented as a whole piece with a minimal size of 2100, but it is split into several pieces when we choose a smaller threshold. The same box can be segmented out as a whole piece from the reference with a threshold of 2500. The ideal match should be applied to the box component extracted from the paired image with minimal size of 2100, and the box component extracted from the reference image with minimal size of 2500.

The similar situation is presented in Figure 35 and Figure 36.

![Figure 35](image)

**Figure 35** The segmented paired “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=2100
Figure 36 The segmented reference “cookie” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=1700

Figure 37 The segmented paired “palm” photos and segmentation maps; (a) minSize=100; (b) minSize=700; (c) minSize=1900
Figure 38 The segmented reference “palm” photos and segmentation maps; (a) minSize=100; (b) minSize=900; (c) minSize=2500

5.3 Java 3D Virtual Scene Viewer

Java 3D is an addition to Java for displaying three-dimensional graphics. I used this package to construct a virtual scene, and then place two objects into the scene in different locations. After this is done, two cameras are associated with the scene, and they are placed along the z-axis with one camera closer to the objects, and the other one further away. My viewer then attaches each camera to one panel in the same frame window. The viewer also provides a function to save the snapshots from both cameras into image files. There is a screenshot of the viewer presented in Figure 39, where the reference camera (closer one) is associated with the panel on the left, and the paired camera (further one) is associated with the panel on the right.
5.4 Stereo Matching Experiment Results

In this section I present the experiment results of my stereo correlation implementation. As I described in the previous chapter, the stereo correlation process uses the output from the Watershed method, and then tries to find the optimal matches among two images. Each set of the results contains 6 images, including original image pairs, the matching results (with the matched components from both images rendered with same colors), and the generated depth images. The correlation results on the artificial images are first presented below, and those image sources are generated from my Java 3D viewer.

In Figure 40, there are two parallel cubes in the artificial scene, and the two cubes are placed into the virtual world such that the front faces of both cubes have the same z-coordinates. Ideally, the front-face regions should be segmented out and matched with each other, and the z-coordinates of the points on front faces should be computed the same. Therefore, in the result depth image, the front-face regions should have the same grayscale colors. The matched
components are shown in **Figure 40** (c) and (d) with the same colors, and the computed z-coordinates for two front-face regions in the depth image are very close, as we expect.

**Figure 40** The stereo correlation result on the parallel cubes in the artificial scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image.
There are two cubes in the scene in Figure 41, and they are placed in different z-coordinates. The cube on the left is further away, while the right one is closer to the camera. Since each face of the cube has a different color, the borders of the faces show significant changes in the gradient image. This guarantees that the faces can be segmented out separately. In the segment matching results depicted in Figure 41 (c) and (d), the faces of both cubes are matched correctly, and the depth values of the faces on the right cube are closer than those on the left cube. The grayscale values in the depth image are brighter in the right cube than those in the left cube, as we expect.
Figure 41 The stereo correlation result on two cubes of different distances to the cameras in the artificial scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image.
In Figure 42, the two objects in the virtual scene are textured, and the cube on the right is much closer than the globe on the left. The Watershed segmentation implementation segments the two objects slightly differently, and some of the regions are matched incorrectly, as displayed in Figure 42 (c) and (d). In general, the regions on the right object are shown brighter than those on the left object, which is closer to the expected result.
Figure 42 The stereo correlation result on two textured objects with different distances to the camera in the artificial scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image.
The following sets of results are generated from the real photos. The camera is controlled manually to take pictures in two different positions along the z-axis, pointing roughly to the same direction. Errors are introduced by the differences of light source, the roughness of camera orientations because of manual control, and the inaccuracy of the image acquisition process. Due to the errors introduced in the image pair, some of the matched components are incorrect, for example, the middle cookie box is mistakenly matched with the background, while some are good matches, as depicted in Figure 43 (c) and (d).
The stereo correlation result on three cookie boxes with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image.

The scene presented in Figure 44 is more complicated, since the palm trees contain irregular regions, such as the leaves. Some of the palm trees are segmented and matched correctly, while some are not.
Figure 44 The stereo correlation result on palm trees with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image.
Figure 45 The stereo correlation result on various objects with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image.
Figure 46 The stereo correlation result on various objects with different distances to the camera in the real scene. (a) the original reference image; (b) the original paired image; (c) the matching result on the reference image; (d) the matching result on the paired image; (e) the depth image.
6 CONCLUSIONS AND FUTURE WORK

In this thesis I present an image segmentation-based stereo correlation method that can be applied to the stereo images which are taken by two cameras placed along the z-axis. In such a set of stereo images, traditional local window-based correlation methods are no longer applied, since the objects are scaled significantly and may be placed into different locations. Instead, I first segment the reference image into components with similar characteristics, and then search the paired image to find matched components. The depth information of the pixels in the matched components can be computed by the ratio of the sizes of the components.

However, the pixels belonging to the same component can only have the same depth values, although the surface plane of the component may not be parallel to the image plane. Future study can be conducted in this direction by extracting more plane information where the components reside from the shapes of components and their transformations, as well as the shadows and other characteristics.

Another possible research focus is to apply an elastic local correlation window in this case and scan through the edges to find more reliable pixel-wise matches. Using such a elastic matching window, we can locate some corner points that can be matched with higher confidence. Those points can be used as a marker to direct the image segmentation and correlation process, and hopefully generate better results.
Appendix

List of Programs

Table 3 The list of program I implemented for the stereo correlation method, and their code statistics

<table>
<thead>
<tr>
<th>File</th>
<th>#Total</th>
<th>#Code</th>
<th>#Comment</th>
<th>#Blank</th>
</tr>
</thead>
<tbody>
<tr>
<td>\image\BinaryImage.java</td>
<td>106</td>
<td>63</td>
<td>29</td>
<td>14</td>
</tr>
<tr>
<td>\image\ColorImage.java</td>
<td>128</td>
<td>82</td>
<td>25</td>
<td>21</td>
</tr>
<tr>
<td>\image\GrayscaleImage.java</td>
<td>127</td>
<td>78</td>
<td>37</td>
<td>16</td>
</tr>
<tr>
<td>\image\ImageBase.java</td>
<td>75</td>
<td>38</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td>\io\ImageStorage.java</td>
<td>86</td>
<td>55</td>
<td>17</td>
<td>15</td>
</tr>
<tr>
<td>\stereo\StereoMatcher.java</td>
<td>812</td>
<td>535</td>
<td>176</td>
<td>104</td>
</tr>
<tr>
<td>\stereo\StereoMatcherColor.java</td>
<td>749</td>
<td>517</td>
<td>135</td>
<td>99</td>
</tr>
<tr>
<td>\test\io\ImageStorageTest.java</td>
<td>104</td>
<td>76</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>\test\stereo\StereoMatcherTest.java</td>
<td>149</td>
<td>102</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>\test\util\CombinerTest.java</td>
<td>116</td>
<td>84</td>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>\test\util\convert\BinaryConverterTest.java</td>
<td>71</td>
<td>51</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>\test\util\convert\ColorConverterTest.java</td>
<td>57</td>
<td>38</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>\test\util\convert\GrayscaleConverterTest.java</td>
<td>134</td>
<td>91</td>
<td>30</td>
<td>13</td>
</tr>
<tr>
<td>\test\util\HistogramTest.java</td>
<td>160</td>
<td>121</td>
<td>20</td>
<td>19</td>
</tr>
<tr>
<td>\test\util\InverserTest.java</td>
<td>100</td>
<td>70</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>\test\util\ResizerTest.java</td>
<td>91</td>
<td>67</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>\test\util\segment\ComponentTest.java</td>
<td>201</td>
<td>112</td>
<td>47</td>
<td>42</td>
</tr>
<tr>
<td>\test\util\segment\LabellerTest.java</td>
<td>60</td>
<td>37</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>\test\util\segment\SegmenterTest.java</td>
<td>187</td>
<td>134</td>
<td>29</td>
<td>24</td>
</tr>
<tr>
<td>\test\util\SharpenerTest.java</td>
<td>235</td>
<td>161</td>
<td>50</td>
<td>24</td>
</tr>
<tr>
<td>\test\util\SmootherTest.java</td>
<td>71</td>
<td>51</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>\util\Combiner.java</td>
<td>154</td>
<td>133</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>File Name</td>
<td>Lines</td>
<td>Comments</td>
<td>Blank</td>
<td>Other</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------</td>
<td>----------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>\util\convert\BinaryConverter.java</td>
<td>30</td>
<td>26</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>\util\convert\ColorConverter.java</td>
<td>79</td>
<td>60</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>\util\convert\GrayscaleConverter.java</td>
<td>311</td>
<td>226</td>
<td>46</td>
<td>40</td>
</tr>
<tr>
<td>\util\Cropper.java</td>
<td>67</td>
<td>54</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>\util\Histogram.java</td>
<td>413</td>
<td>285</td>
<td>70</td>
<td>58</td>
</tr>
<tr>
<td>\util\Inverser.java</td>
<td>76</td>
<td>58</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>\util\Resizer.java</td>
<td>204</td>
<td>156</td>
<td>17</td>
<td>33</td>
</tr>
<tr>
<td>\util\segment\Bound.java</td>
<td>129</td>
<td>74</td>
<td>38</td>
<td>17</td>
</tr>
<tr>
<td>\util\segment\Component.java</td>
<td>328</td>
<td>227</td>
<td>68</td>
<td>45</td>
</tr>
<tr>
<td>\util\segment\Labeller.java</td>
<td>246</td>
<td>188</td>
<td>41</td>
<td>30</td>
</tr>
<tr>
<td>\util\segment\Segmenter.java</td>
<td>620</td>
<td>464</td>
<td>88</td>
<td>76</td>
</tr>
<tr>
<td>\util\Sharpener.java</td>
<td>257</td>
<td>186</td>
<td>31</td>
<td>40</td>
</tr>
<tr>
<td>\util\Smoother.java</td>
<td>107</td>
<td>86</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>\world\ViewFrame.java</td>
<td>244</td>
<td>176</td>
<td>29</td>
<td>39</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>7084</strong></td>
<td><strong>4962</strong></td>
<td><strong>1221</strong></td>
<td><strong>948</strong></td>
</tr>
<tr>
<td><strong>Percentage</strong></td>
<td><strong>70.05%</strong></td>
<td><strong>17.24%</strong></td>
<td><strong>13.38%</strong></td>
<td></td>
</tr>
</tbody>
</table>
REFERENCES


41 A. Walsh, D. Gehringer, Java 3D API jump-start, Prentice Hall, 2002