9-1-2009

Automated 3D object modeling from aerial video imagery

Prudhvi K. Gurram

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Dissertation is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact rit.scholarworks@rit.edu.
Automated 3D Object Modeling from Aerial Video Imagery

by

Prudhvi K. Gurram

M.S. Electrical Engineering,
Rochester Institute of Technology, USA, 2008
B.E. Electronics and Communication Engineering,
National Institute of Technology Karnataka, India, 2003

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Chester F. Carlson Center for Imaging Science Rochester Institute of Technology

September 2009

Signature of the Author ____________________________________________

Accepted by ____________________________________________
Coordinator, Ph.D. Degree Program Date
The Ph.D. Degree Dissertation of Prudhvi K. Gurram has been examined and approved by the dissertation committee as satisfactory for the dissertation required for the Ph.D. degree in Imaging Science.

Dr. Eli Saber, Dissertation Advisor

Dr. Harvey Rhody

Dr. Sohail Dianat

Dr. Ferat Sahin

Dr. Joseph Voelkel

Date
Title of Dissertation:
Automated 3D Object Modeling from Aerial Video Imagery

I, Prudhvi K. Gurram, hereby grant permission to Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature __________________________________________ Date
Automated 3D Object Modeling from Aerial Video Imagery

by

Prudhvi K. Gurram

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

Research in physically accurate 3D modeling of a scene is gaining momentum because of its far reaching applications in civilian and defense sectors. The modeled 3D scene must conform both geometrically and spectrally to the real world for all the applications. Geometric modeling of a scene can be achieved in many ways of which the two most popular methods are - a) using multiple 2D passive images of the scene also called as stereo vision and b) using 3D point clouds like Lidar (Light detection and ranging) data. In this research work, we derive the 3D models of objects in a scene using passive aerial video imagery. At present, this geometric modeling requires a lot of manual intervention due to a variety of factors like sensor noise, low contrast conditions during image capture, etc. Hence long time periods, in the order of weeks and months, are required to model even a small scene. This thesis focuses on automating the process of geometric modeling of objects in a scene from passive aerial video imagery. The aerial video frames are stitched into stereo mosaics. These stereo mosaics not only provide the elevation information of a scene but also act as good 3D visualization tools. The 3D information obtained from the stereo mosaics is used to identify the various 3D objects, especially man-made buildings using probabilistic inference provided by Bayesian Networks. The initial 3D building models are further optimized by projecting them on to the individual video frames. The limitations of the state-of-art technology in attaining these goals are presented along with the techniques to overcome them. The improvement that can be achieved in the accuracy of the 3D models when Lidar data is fused with aerial video during the object identification process is also examined.
To my mom and dad who made me what I am today...
To
The Inhabitants of SPACE IN GENERAL
And H.C. IN PARTICULAR
This Work is Dedicated
By a Humble Native of Flatland
In the Hope that
Even as he was Initiated into the Mysteries
Of THREE Dimensions
Having been previously conversant
with ONLY TWO
So the Citizens of that Celestial Region
May aspire yet higher and higher
To the Secrets of FOUR FIVE OR EVEN SIX Dimensions
Thereby contributing
To the Enlargement of THE IMAGINATION
And the possible Development
Of that most rare and excellent Gift of MODESTY
Among the Superior Races
Of SOLID HUMANITY

– Edwin A. Abbott, FLATLAND: A Romance of Many Dimensions, 1885
Acknowledgements

I would like to express my sincere gratitude to my advisor Dr. Eli Saber for his steadfast guidance and support throughout the period of my doctoral studies. He has been very patient and was always there to keep me focused on the final goal. He helped me in developing confidence for independent research, which was my personal goal in PhD. This work would not have been possible without him. I would also like to thank Dr. Harvey Rhody who initiated 3D computer vision research at the Center and supported my work through his helpful insights into this field. My graduate studies at RIT started in the Department of Electrical Engineering where Dr. Sohail Dianat hired me as his research assistant. I cannot possibly express my gratitude to Dr. Dianat. He taught me the importance of diligence and work ethic. His incredible mathematical and MATLAB skills are awe-inspiring and I will always remain deeply indebted to him for helping me hone mine. I would like to thank Dr. Ferat Sahin who introduced me to the field of Artificial Intelligence and helped me in formulating a crucial part of my PhD thesis. I would also like to thank Dr. Joseph Voelkel for serving as the external chairman of my PhD thesis committee and providing valuable feedback on my dissertation. I have been fortunate to be financially supported by a grant from the National Geospatial Intelligence Agency (NGA) and by the Laboratory for Imaging Algorithms and Systems (LIAS).

Special thanks go to Jason Faulring who provided me with the airborne data used in this dissertation as well as support for pre-processing step of my thesis. Matthew Montanaro took time away from his work to run the DIRSIG model and create data sets for me. Jim Stefano always provided support for the computers and other systems in the Electrical Engineering lab and made sure that my work went well without any interruptions. Steve Lach was very kind in providing insights into 3D computer vision and my thesis. I cannot express enough gratitude to Archana Devasia. Not only did she proofread all my manuscripts but also provided me with constant guidance over the last six years for everything in my life. Mustafa Jaber was always there to provide useful comments on my manuscripts and thesis work. Aaron Gerace and Matthew Montanaro were always there to listen to my complaints, bolster my confidence and encourage me on. I owe you guys for my positive outlook towards the end of my work here. Xiaofeng Fan listened to my ideas and critiqued them. To Karl Walli, the glass is always half full and it is with this attitude that he advised me time and again. This work would not have been possible without the help of these amazing individuals.

I would like to thank the faculty and staff of Center for Imaging Science (CIS) and Department of Electrical Engineering (EE) for their support over the last six years. Even
though all the teachers I had for courses were not directly involved in my thesis, each one had a part in formulating my thought process. I am glad I took all the courses I did.

I have been blessed with a wonderful family - my parents, my brother and my sister who provided me with encouragement at every step of my life. I thank my family and friends for supporting me throughout my doctoral studies.
Contents

Abstract I

Table of Contents VI

List of Figures VIII

List of Tables XII

1 Introduction 1
1.1 Research Goals .................................................. 2

2 Background 5

3 Approach 13
3.1 Pre-processing of Video Frames ................................. 14
3.2 Ray Interpolation .................................................... 18
  3.2.1 Geometry ....................................................... 18
  3.2.2 Existing Fast PRISM Algorithm .............................. 22
  3.2.3 Problem with Fast PRISM .................................. 23
  3.2.4 Segment-based Mesh Design ................................ 25
  3.2.5 Modified PRISM Algorithm ................................ 26
3.3 3D Object Identification ......................................... 30
  3.3.1 Theory of Bayesian Networks ................................. 32
  3.3.2 Bayesian Network Structure ................................. 33
  3.3.3 Feature Extraction ............................................ 34
    3.3.3.1 Elevation Information from Visual Imagery .......... 39
    3.3.3.2 Elevation Information from Lidar Data ............... 44
    3.3.3.3 Edge and Corner Information from Visual Imagery . 45
    3.3.3.4 Surface Area from Visual Imagery .................. 46

VI
# List of Figures

1.1 Standard scene generation process for DIRSIG .......................... 2  
3.1 Block Diagram of the approach used for modeling of 3D objects from aerial images .................................................. 14  
3.2 Relationship between world coordinate system and image coordinate system 15  
3.3 Relationship between image coordinates in pixels and image coordinate system .................................................. 17  
3.4 Geometry of a linear pushbroom camera and a pin-hole camera ........ 18  
3.5 Fixed lines (scanlines of a pushbroom sensor) in a video frame .......... 19  
3.6 Overlapped region between the fixed lines in two successive frames .... 20  
3.7 Interpolation of corresponding points in two frames on to the mosaic .... 21  
3.8 Steps in fast PRISM algorithm .................................................. 24  
3.9 Motion parallax between a helipad which is at greater height than the ground 24  
3.10 Mosaic formed from the two frames ......................................... 25  
3.11 Triangulation using edges and segments .................................... 26  
3.12 Steps in modified PRISM algorithm ........................................ 29  
3.13 Handcrafted structure of Bayesian network to fuse visual and Lidar information .................................................. 35  
3.14 Handcrafted structure of Bayesian network to use visual information only . 36  
3.15 Original video frames captured by an airborne sensor ................. 36  
3.16 Left and right mosaics built from airborne video imagery ................ 37  
3.17 Stereo mosaics in red-cyan anaglyph form ............................... 37  
3.18 Nadir mosaic built from airborne video imagery .......................... 38  
3.19 Rasterized elevation map from Lidar point cloud ........................ 38  
3.20 Material map generated by DIRSIG ......................................... 38  
3.21 Truth map used for training the BN ......................................... 38  
3.22 Segmented mosaics ($h_s = 15, h_c = 10, M = 20$) .................... 40
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.23</td>
<td>Segment maps of the mosaics ( h_s = 15, h_c = 10, M = 20 )</td>
<td>41</td>
</tr>
<tr>
<td>3.24</td>
<td>Elevation map generated from stereo mosaics</td>
<td>43</td>
</tr>
<tr>
<td>3.25</td>
<td>Projection of Lidar data points on to the fixed baseline</td>
<td>45</td>
</tr>
<tr>
<td>3.26</td>
<td>Corner extraction process for each surface in the nadir mosaic</td>
<td>46</td>
</tr>
<tr>
<td>3.27</td>
<td>The conical representation of HSV color space</td>
<td>47</td>
</tr>
<tr>
<td>3.28</td>
<td>Entropy map of nadir mosaic ((9 \times 9 \text{ window}))</td>
<td>48</td>
</tr>
<tr>
<td>3.29</td>
<td>Entropy map of Lidar elevation data ((9 \times 9 \text{ window}))</td>
<td>48</td>
</tr>
<tr>
<td>3.30</td>
<td>Flowchart for 3D modeling of a building</td>
<td>57</td>
</tr>
<tr>
<td>3.31</td>
<td>Modeling a T corner during post processing step</td>
<td>60</td>
</tr>
<tr>
<td>3.32</td>
<td>Final model of a building with T corners</td>
<td>60</td>
</tr>
<tr>
<td>3.33</td>
<td>Modeling of corners at different elevations during post processing step</td>
<td>61</td>
</tr>
<tr>
<td>3.34</td>
<td>Final model of a building with corners at different elevations</td>
<td>61</td>
</tr>
<tr>
<td>3.35</td>
<td>Final model of a building with different corners at same elevation</td>
<td>61</td>
</tr>
<tr>
<td>3.36</td>
<td>Final model of a building with different corners at different elevations</td>
<td>61</td>
</tr>
<tr>
<td>4.1</td>
<td>Two frames created artificially with motion parallax between objects</td>
<td>63</td>
</tr>
<tr>
<td>4.2</td>
<td>Source triangles from fast PRISM in two consecutive frames</td>
<td>63</td>
</tr>
<tr>
<td>4.3</td>
<td>Source triangles from modified PRISM in two consecutive frames</td>
<td>64</td>
</tr>
<tr>
<td>4.4</td>
<td>Left mosaic and destination triangles using fast PRISM</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>Left mosaic and destination triangles using modified PRISM</td>
<td>64</td>
</tr>
<tr>
<td>4.6</td>
<td>Mosaics illustrating the improvement between the two methods</td>
<td>65</td>
</tr>
<tr>
<td>4.7</td>
<td>Six frames from an indoor video sequence</td>
<td>65</td>
</tr>
<tr>
<td>4.8</td>
<td>Mosaics built from indoor video</td>
<td>66</td>
</tr>
<tr>
<td>4.9</td>
<td>Stereo mosaic pairs of the indoor scene</td>
<td>66</td>
</tr>
<tr>
<td>4.10</td>
<td>CAD models of the box built from the stereo mosaics</td>
<td>66</td>
</tr>
<tr>
<td>4.11</td>
<td>Six frames from an airborne video sequence</td>
<td>67</td>
</tr>
<tr>
<td>4.12</td>
<td>Parts of right mosaics built from airborne imagery</td>
<td>68</td>
</tr>
<tr>
<td>4.13</td>
<td>CAD models of the building built from the stereo mosaics</td>
<td>68</td>
</tr>
<tr>
<td>4.14</td>
<td>Frames from an airborne data sequence</td>
<td>69</td>
</tr>
<tr>
<td>4.15</td>
<td>Right mosaics built from airborne imagery</td>
<td>70</td>
</tr>
<tr>
<td>4.16</td>
<td>Nadir mosaic built from airborne video imagery over a scene</td>
<td>71</td>
</tr>
<tr>
<td>4.17</td>
<td>Segmentation maps generated from nadir mosaic for different color band-</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>widths in mean shift algorithm</td>
<td></td>
</tr>
<tr>
<td>4.18</td>
<td>Material map of the scene generated by DIRSIG</td>
<td>72</td>
</tr>
<tr>
<td>4.19</td>
<td>Original truth map generated from the material map of the scene</td>
<td>72</td>
</tr>
<tr>
<td>4.20</td>
<td>Pseudo truth maps generated from segmentation maps for training the BN</td>
<td>72</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>4.21</td>
<td>Pseudo truth maps generated for different segmentation parameters</td>
<td></td>
</tr>
<tr>
<td>4.22</td>
<td>Classification maps generated by the BN using visual information only</td>
<td></td>
</tr>
<tr>
<td>4.23</td>
<td>Plot of quality metric vs. color bandwidth for the case of ( W_{\text{building}} = 1 ) and rest of the weights 0</td>
<td></td>
</tr>
<tr>
<td>4.24</td>
<td>Plot of quality metric vs. color bandwidth for the case of ( W_{\text{building}} = 0.5, W_{\text{trees}} = 0.5 ) and rest of the weights 0</td>
<td></td>
</tr>
<tr>
<td>4.25</td>
<td>Plot of quality metric vs. color bandwidth for the case of ( W_{\text{building}} = 0.1, W_{\text{trees}} = 0.9 ) and rest of the weights 0</td>
<td></td>
</tr>
<tr>
<td>4.26</td>
<td>Classification maps generated by the BN using visual and Lidar information</td>
<td></td>
</tr>
<tr>
<td>4.27</td>
<td>Plot of quality metric vs. color bandwidth for the case of ( W_{\text{building}} = 1 ) and rest of the weights 0</td>
<td></td>
</tr>
<tr>
<td>4.28</td>
<td>Plot of quality metric vs. color bandwidth for the case of ( W_{\text{building}} = 0.5, W_{\text{trees}} = 0.5 ) and rest of the weights 0</td>
<td></td>
</tr>
<tr>
<td>4.29</td>
<td>Plot of quality metric vs. color bandwidth for the case of ( W_{\text{building}} = 0.1, W_{\text{trees}} = 0.9 ) and rest of the weights 0</td>
<td></td>
</tr>
<tr>
<td>4.30</td>
<td>Nadir mosaic built from the new data set</td>
<td></td>
</tr>
<tr>
<td>4.31</td>
<td>Truth map for the new data set</td>
<td></td>
</tr>
<tr>
<td>4.32</td>
<td>Classification map generated by the trained BN (using visual and Lidar information) for the new data set</td>
<td></td>
</tr>
<tr>
<td>4.33</td>
<td>Bayesian network structure learnt from data using Particle Swarm Optimization</td>
<td></td>
</tr>
<tr>
<td>4.34</td>
<td>The buildings map of the scene generated by the BN</td>
<td></td>
</tr>
<tr>
<td>4.35</td>
<td>The example building used to illustrate the object modeling process</td>
<td></td>
</tr>
<tr>
<td>4.36</td>
<td>Initial 3D model of a building in the scene</td>
<td></td>
</tr>
<tr>
<td>4.37</td>
<td>Individual video frames in which the example building is visible</td>
<td></td>
</tr>
<tr>
<td>4.38</td>
<td>Segmented individual video frames in which the example building is visible</td>
<td></td>
</tr>
<tr>
<td>4.39</td>
<td>Matched point pairs of corners of a building in video frames</td>
<td></td>
</tr>
<tr>
<td>4.40</td>
<td>Matched point pairs of corners of a building in magnified video frames</td>
<td></td>
</tr>
<tr>
<td>4.41</td>
<td>Minimization of the distances between projected corners and actual corners in the video frames</td>
<td></td>
</tr>
<tr>
<td>4.42</td>
<td>Final 3D model of the building after local optimization</td>
<td></td>
</tr>
<tr>
<td>4.43</td>
<td>Building 2 used to illustrate the object modeling process</td>
<td></td>
</tr>
<tr>
<td>4.44</td>
<td>Minimization of the distances between projected corners and actual corners in the video frames</td>
<td></td>
</tr>
<tr>
<td>4.45</td>
<td>Final 3D model of the building after local optimization</td>
<td></td>
</tr>
<tr>
<td>4.46</td>
<td>A building used to illustrate the object modeling process</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>List of nodes/features used in BN (with visual and Lidar information)</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>List of nodes/features used in BN (with visual information only)</td>
<td>37</td>
</tr>
<tr>
<td>3.3</td>
<td>Labels used to transform material map to truth map</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Utilities assigned by an expert to make decisions on classification</td>
<td>52</td>
</tr>
<tr>
<td>4.1</td>
<td>True positive and false positive rates for building class using visual information only</td>
<td>73</td>
</tr>
<tr>
<td>4.2</td>
<td>True positive and false positive rates for grass class using visual information only</td>
<td>75</td>
</tr>
<tr>
<td>4.3</td>
<td>True positive and false positive rates for trees class using visual information only</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>True positive and false positive rates for asphalt class using visual information only</td>
<td>76</td>
</tr>
<tr>
<td>4.5</td>
<td>True positive and false positive rates for building class using visual and Lidar information</td>
<td>78</td>
</tr>
<tr>
<td>4.6</td>
<td>True positive and false positive rates for grass class using visual and Lidar information</td>
<td>78</td>
</tr>
<tr>
<td>4.7</td>
<td>True positive and false positive rates for trees class using visual and Lidar information</td>
<td>78</td>
</tr>
<tr>
<td>4.8</td>
<td>True positive and false positive rates for asphalt class using visual and Lidar information</td>
<td>81</td>
</tr>
<tr>
<td>4.9</td>
<td>True positive and false positive rates for new data set using the trained BN</td>
<td>83</td>
</tr>
<tr>
<td>4.10</td>
<td>Final error statistics for each building</td>
<td>94</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Research in 3D scene reconstruction is gaining momentum because of its far reaching applications not only in the defense sector but also in the civilian sector. Military applications include target area simulation and moving target detection in a 3D scene. Non-military applications include 3D maps and assessment of degree of damage in case of natural disasters. The other applications include 3D visualization of human body in medical imaging and robotic vision. For most of these applications, the reconstructed 3D model of a scene must conform both geometrically and spectrally to the real world. This is so, because for most of the remote sensing applications, the synthetic imagery generated using these 3D models is very useful in providing accurate physical properties of the real world.

The Digital Imaging and Remote Sensing Image Generation (DIRSIG) model is a tool capable of creating this desired physically realistic synthetic imagery ([1] and [2]). A simple illustration showing the inputs used to build the 3D scene for DIRSIG is shown in Figure 1.1. Given a pre-defined scene, it generates an accurate representation of what a modeled passive electro-optical (EO) or active Lidar sensor would detect under specified conditions by modeling the relevant physical processes in the imaging chain. It is currently being used in both academia and industry. However, while it is fairly easy to model different sensor designs, atmospheric conditions and geometries for a given scene, the current process for actually generating a DIRSIG scene is quite involved. As a case in point, the creation of the MegaScene documented by Ientilucci [1] took a team of researchers well over a year to complete. The solution to this problem is to automate the process of 3D scene generation for DIRSIG.

To achieve a spectrally-accurate 3D scene, first 3D geometrical information is extracted from raw imagery (see CAD Models in Figure 1.1). Then, all the facets in this 3D model
are assigned spectra which are extracted from hyperspectral imagery of the scene. The raw imagery is obtained from various sensors ranging from active Lidar sensor to passive video camera. The 3D information from the active sensors like Lidar is apparent, but the 2D images from the passive sensors like digital or video cameras have to be processed to extract 3D information from them. Though the geometric models generated using Lidar are more accurate than the ones generated using passive imagery, Lidar is not used widely, because of the expensive equipment required to collect the data and difficulty in obtaining the data at any time and place. Hence, we use passive airborne imagery to generate 3D geometric models of a scene.

This dissertation describes the algorithms and the methods developed to automate the 3D building modeling process in semi-urban scenes from aerial video imagery. The limitations of the algorithms and the passive data are identified. In order to overcome these limitations, passive imagery is fused with Lidar point cloud. This would not only automate the scene generation process to a further degree but also improve the accuracy of the models. The optimization techniques used to increase the accuracy of the different processes in the system are also explained in detail.

1.1 Research Goals

This section lists all the research goals and objectives that the author aims at achieving by the end of his doctoral studies. They are listed in the order of the steps followed to model 3D objects from aerial video frames.
1.1. RESEARCH GOALS

1. The first goal is to process the aerial video frames to obtain stereo mosaics. The video frames are pre-processed in order to remove the relative/absolute rotational effects between the frames. Next, the 3D translational coordinates of the sensor position for each of the frames are used to stitch them into two mosaics which act as a stereo pair.

2. The second goal is 3D object identification and modeling from the stereo mosaics. This step involves identifying all the regions corresponding to buildings and trees. While tree regions are just identified in the scene, all the surfaces belonging to each of the polyhedral buildings in the scene are grouped together and modeled into a 3D CAD model. The initial elevation information for each of the surfaces is obtained from the stereo mosaic pair.

3. The third goal involves the identification of limitations of using passive imagery to extract and model 3D objects in a scene. The limitations are overcome by using the information from Lidar data and fusing it with the information from passive video imagery in order to build the 3D models of the buildings in a scene.

There are existing methods which perform all the above tasks. The most important goal in this research is to automate this entire chain of processes as much as possible while still maintaining the same quality or improving the quality of the results obtained in terms of accuracy and/or speed. To this effect, improvements have been made in the method used to build stereo mosaics from video frames and hence the first goal has been achieved. The automation of second and third goals has been achieved by using probabilistic inference through Bayesian Networks (BN) to fuse the information from stereo mosaics and Lidar point cloud. The accuracy of the 3D models of the buildings is improved by using local optimization techniques.

The following original contributions have been made by this work to the knowledge base in photogrammetry and 3D computer vision fields while achieving the research goals listed above:

1. A segment-based mesh design for aerial triangulation without any prior 3D knowledge of the scene has been developed. This new design helps in avoiding visual artifacts in the parallel-perspective stereo mosaics that are built using ray interpolation. Consequently, the errors in the final 3D models of buildings are reduced.

2. A novel method to control the input parameters of vision algorithms like color segmentation using the data-driven probabilistic inference in Bayesian networks has
been designed. This method automates the 3D object identification process and precludes the need for manual intervention to set the accurate input parameters for best quality of the final 3D building models.

The rest of the dissertation is organized as follows. Chapter 2 provides a comprehensive look at the literature on 3D object modeling from multiple images and explains the problems existing in the present techniques. This chapter also discusses what problems are dealt with in this work and what improvements are made in the existing techniques. Chapter 3 presents our approach and how each step deals with the existing problems and helps in automating the system. Chapter 4 illustrates the results obtained in detail. Chapter 5 concludes this dissertation by listing out the contributions of this thesis to the fields of photogrammetry and computer vision. In this chapter, future work that can be done in this field is also discussed. Final sections of appendices provide derivations of some mathematical expressions used in the intermediate chapters.
Chapter 2

Background

A variety of methods are available in the market today to extract 3D information from 2D images. The most widely used method is stereo vision which refers to the ability to infer information on the 3D structure of a scene from two or more images captured by a sensor from different viewpoints [3]. It is very popular because it gives us the capability to visualize the 3D structure in the scene without actually extracting it. Another method is to use feature-based motion analysis, which exploits the finite changes in a series of images (video frames) induced by the relative motion of the world and camera to extract 3D structure [3]. Yet another method uses the shading in a single image to extract 3D structure of objects in the image [3].

Many researchers have worked on different camera geometries for stereo vision. One of the popular camera geometries for stereo vision has been circular projective geometry which is used in building panoramic mosaics. Earlier, panoramic mosaics were built using special sensors which rotated about their optical centers. But such systems have limitations on imaging conditions [4]. So, better panoramic mosaicing systems were developed for hand-held general motion of the camera [4]. In [5], [6], [7], the authors combined stereo geometry with panoramic mosaics to extract 3D information for a full rotation of 360° in a scene. But we are more concerned with extracting 3D information from airborne video of a scene in which case, the motion of the sensor is more translational rather than rotational. A translational stereo system usually has a stereo rig with two cameras placed at a particular angle with each other (to satisfy stereo geometry). But this hardware formerly required for a stereo system is no longer indispensable, as new algorithms can extract desired 3D information from two or more images of a scene captured by a single camera. A good review of many effective methods are found in the literature for automatic reconstruction of a 3D scene from images can be found in [8].
Every object in the scene requires two or more views for such techniques [8] to be successfully applied on them. Hence, a large scene requires many images to cover the entire area. To effectively deal with this problem, a video camera maybe attached to an aircraft and flown over the scene. However this results in hundreds of frames that need to be processed. Also, in order to extract 3D coordinates of a particular object, we must identify the frames in which that object is present. There is also an additional constraint that the object has to exhibit good disparity with respect to the baseline for fairly accurate results. Thus the frames need to be indexed and used for the retrieval of 3D data of any object. This is a very tedious process to implement for an extended scene due to the large number of frames and objects involved in the scene. To alleviate this issue, we have chosen to stitch the frames to form a mosaic which is easier to handle than individual frames. Two mosaics are built from these frames, in such a way that they form a stereo pair for the entire scene. The stereo mosaics facilitate automation of 3D object modeling for four main reasons. First, if we build a nadir mosaic as explained in [9], the scene is already orthorectified along the direction of motion of the sensor. Second, since we know exact epipolar geometry of the assumed linear pushbroom camera, it is less complex to find putative matches between corresponding features in the stereo mosaics. Third, since there are no rotational effects involved in the stereo mosaics, it is straightforward to match the corresponding features and extract the depth information based on the disparity. Finally, a human operator can always look at the visual evidence using stereo glasses (like anaglyph glasses) and confirm the shape and geometry of the 3D objects modeled from the stereo mosaics.

The process of stitching large number of video frames to form a mosaic is very simple once corresponding points are matched in successive frames. But such mosaics do not serve our purpose because they are not seamless, and the apparent motion parallax information between the frames is often lost in the mosaics. To address this, Zhu et al. [10] proposed Parallel Ray Interpolation for Stereo Mosaicing (PRISM) to develop these two stereo images from airborne video. Each of the two images is a mosaic built from a series of video frames which when viewed together as a stereo system, form left and right stereo images. Zhu's method is very effective in building seamless stereo mosaics. This method uses the apparent motion parallax existing between successive video frames to build better mosaics. An overview of Zhu's method is given later on in this dissertation to give the readers a good perspective on the changes we made to this method. The first research goal of this thesis is to improve upon the triangulation method used in building the mosaics in [10].

Usually triangulation is done as a part of mesh development which is used in a va-
riety of video applications especially for finding local spatial transformations between video frames. The main step in our algorithm is also to find spatial transformation (affine) between the video frames and the mosaics. For affine transformation, we need three corresponding control points between the source image and destination image [3], [11]. As affine transformation is a linear transformation, the entire region inside each triangle defined by the three points can be transformed as a single block. So triangulation of the control points is a major step in the process of building the mosaics.

Different methods of triangulation or mesh design are available in the literature. We can group the mesh design strategies into four main types: 1) regular mesh, 2) hierarchical mesh, 3) content-based mesh, and 4) knowledge-based mesh designs [12]. In [13], the authors used regular mesh design to find local affine transformations between video frames for video encoding application. In [14], hierarchical mesh design was used to recursively divide the image into triangles of various sizes and orientations to determine a non-linear local geometric distortion for image registration. In [15], hierarchical mesh design was employed to estimate motion in the video using triangular patches. In [10], the authors suggest a triangulation method developed by Morris and Kanade [17] to be applied to their method. However they describe an optimum triangulation method given the prior knowledge of 3-D points of the object. In [18], [19], the authors used a previously defined 3-D model to synthesize an output image given an input image. But unlike [17], [18] and [19], in our case we do not have such prior 3-D knowledge about the objects in the scene. We looked more closely at the content-based mesh design techniques for our purposes. In [20], mesh was developed by placing the nodes or vertices of the triangles at high gradient points (corners) in the first frame and these nodes are followed in the successive frames. But this constraint does not guarantee that the triangles do not cross over edges of the objects in the image. In [21] a combination of the hierarchical and content-based mesh design techniques was used to optimize the mesh.

None of the methods discussed above can satisfy our requirement that the triangles do not cross over from one planar facet to another (This requirement is explained later on in Section 3.2.3). So we have developed a method which does not require any prior information about the 3D scene and uses the content readily available in the images. Our method is different from previous methods in two ways - First, we use segmentation to recognize the planar surfaces in the image and design a regular mesh in each segment; Second, after the triangles in each segment are transformed from the source image to destination mosaic, independent of other segments, we make sure that there are no gaps between the transformed segments in the mosaics. The accuracy of the 3-D information extracted from the stereo mosaics is closely related to the visual quality of the stereo mosaics.
Once the stereo mosaics are built, stereo geometry can be used to extract the height information of different surfaces of the objects in the scene. In this thesis, we are trying to geometrically model the man-made objects in the scene i.e. man-made buildings. This process comprises of four important parts. The first part is identification and recognition of the building surfaces. The second part is identification of 2D features from the mosaics which describe these building surfaces. The third part is combining these 2D features with height information obtained from stereo mosaics to come up with the 3D features which provide building primitives. The final part involves optimizing the parameters of the building primitives to produce accurate 3D building models.

A very good review of building reconstruction methods from (a) images alone, (b) laser data alone and (c) images and laser data combined is provided in [8]. As explained in [8], extraction and modeling of man-made objects like buildings in 3D space consists of two important steps:

1. Detection and recognition of the objects in the scene - different control paradigms are used for this purpose like bottom-up (data-driven), top-down (model-driven) and mixed approaches like hypothesize-and-test or hypothesize-and-verify.

2. Measurement of geometric information - the position, orientation and size of the objects have to be measured after recognition. The objects can be described using (a) boundary representation, (b) constructive solid geometry and (c) spatial enumeration like voxels.

We will summarize the existing methods that tackle 3D object modeling and their problems. We are also going to provide a brief overview of how our proposed approach can tackle the existing problems in automation of 3D object extraction and modeling process. Haala [8] proposes a process of step-by-step filtering of the primitives from a low level to a high level. His work starts with extraction of 2D line segments separately from the stereo images. They are transformed into 3D line segments using the disparity map obtained using stereo geometry. This constitutes the first filtering step. These 3D line segments are combined to form 3D rectangles and second filtering step is executed by applying meaningful thresholds on angles, surface areas etc. Finally, the last filtering step is applied to combine the 3D rectangles into a saddleback type building primitive. Accurate parameters describing the saddleback building are obtained by optimizing the error between the building primitive and a saddleback wireframe model. But this method has obvious shortcomings. First, this work deals with only one type of building model i.e., saddleback type. Second, all the thresholds applied during each filtering step are deterministic and will fail if there is any noise in the images or the disparity map extracted.
from the images. Third, this process expects the vision algorithms to detect and recognize the building surfaces to work perfectly.

In [22], Henricsson and Baltavias start the 3D building modeling process by extracting 2D line segments and identifying the segments or surfaces on either side of the line segments. The 2D line segments from one image (master image) are matched to the corresponding line segments in other stereo images using constraints like compliance with epipolar geometry, color attributes of the surfaces identified on either side of the line segments. Thus 3D line segments are generated and are combined to form hypotheses of planes. Similarity measures like coplanarity, proximity, orientation and color attributes are used to produce polyhedral building models. This work stresses the importance of interaction between 2D features and their 3D counterparts. However, the authors assume the availability of color information in the images, good edge extraction methods to produce perfect edges in the master image.

Fischer et al. [23] base their 3D building modeling process on a hierarchical structure. At each level, the models in 2D space and 3D space are coupled to one another. Along with the models, the constraints and quality measures are also laid out in the hierarchical structure. The top level consists of the scene in 3D space and images in 2D space. Next lower level consists of objects like buildings in 3D space and their corresponding aspect in images. Further below, different surfaces of buildings and their corresponding surface segments in images are coupled. Further down, features like edges and corners of the buildings are coupled in 3D and 2D space. The lowest level consists of the basic features of voxels in 3D and pixels in 2D. The authors use the data to move from bottom to the top of the hierarchical structure while making sure that the constraints are satisfied at each level.

Baillard and Zisserman [24] extract 2D line segments in an image and find their correspondences among six images. Using these correspondences, 3D line segments are obtained. For each of these 3D line segments, left and right half planes are determined from the left and right surfaces in the image. The actual slopes of these planes are obtained by using certain geometric constraints like the plane equation and orientation parameters of the camera. Then all the 3D line segments and 3D planes are grouped together under the constraints of collinearity and coplanarity. Verification of the surfaces is done by using a similarity measure over all the images.

In [25], 3D buildings are modeled by Suveg and Vosselman using 2D ground plans and images. First, the ground plan is projected on to the images and height information is extracted. Then, the building in the 2D ground plan is divided into rectangles and 2D corners are extracted. These 2D corners are used to come up with 3D corners. 3D corners
are used as primitives and the geometric information of these 3D primitives is refined using two measures - contour measure and texture measure. Finally, all 3D primitives are refined simultaneously using the relations between these primitives as constraints.

All the above mentioned algorithms were developed by assuming good performance of certain feature extraction algorithms like edge detectors and accurate identification of homogeneous surfaces. This is not true in all cases. For instance, the problem of solar shadow is very difficult to solve using these methods since edges can not be detected easily. Even the methods which extract 3D information using solar shadows - a whole problem in itself - require the position of Sun. The extracted 3D building would have a completely different shape if one of the edges is missing. This would also be the case if we assume an edge to be present when in reality it is not.

The existing algorithms also recognize all surfaces belonging to a building in a deterministic way at a local level which is selected by the user by employing a Region-Of-Interest (ROI) tool. However, the algorithms would fail when there is noise either in the images or in the orientation parameters of the sensor. Recognition of all the surfaces belonging to a building by using information over the entire image or scene is related to the object identification and recognition part of the 3D object modeling process.

The problems we face in 3D object modeling can be solved by combining the models obtained from aerial video imagery with models from Lidar data. However, how does one decide which feature to extract from which data and use it in the modeling process? How does one deal with this uncertainty? How do we avoid overfitting the model to one scene and hence avoid its failure when applied on another scene? A really good method to use the Hypothesize-and-Verify approach under uncertainty of evidence/data is using Bayesian statistical methods. Bayesian Networks (BNs) provide an excellent tool to exploit the causal relationships between the variables that arise in 3D reconstruction. BNs have been extensively used in the last few years for 3D object modeling.

Investigators at the University of Massachusetts, Amherst and University of Southern California have been the frontrunners in this research of applying BNs to identify, recognize and model 3D buildings using images and elevation data obtained from Lidar point cloud or by using stereo geometry on multiple images. Ascender II system developed in UMass [26], [27], [28] uses BNs to control vision algorithms and obtain the best reconstructed scene using multiple images. The authors use prior knowledge to handcraft the belief networks and use ground truth data (visual information provided by humans) to learn the parameters of the networks. These networks are developed at a hierarchical level where each level represents the scene at one detail level. The structure of the belief network is also learned using the data using Cheng’s algorithms [29]. Utility theory is used
Background

to make decisions regarding the classes of the objects at each level. This research group has been very successful in controlling the vision algorithm used in 3D modeling using BNs. The group from Southern California use Expandable Bayesian Networks (EBN) [30] to represent the evidence features which vary according to number of images available and any other data forms like Lidar data. Each hypothesis is verified using these belief networks and thus the uncertainty in the evidence or data available is handled. However, all the existing methods assume the prior and accurate knowledge of different surfaces/regions in the scene from which different features are extracted to be used in the BNs. Additionally, the former group also assumes the availability of orthorectified scenes with accurately registered dense elevation map obtained through sources like Lidar data. The problem with the above methods is that seldom can one identify all the building surfaces without any errors in edge detection in natural scenes because such identification would require extensive manual intervention. And, there is always some error in the registration between Lidar point cloud and the visual imagery which are collected at different times.

The work presented in this dissertation solves all the above mentioned problems as follows:

1. Detection and recognition of the objects in the scene - Bayesian Networks (BNs) are used to identify different 3D objects in the scene like buildings, tree regions, etc. by using probabilistic inference on the features extracted from the stereo mosaics and the nadir mosaic. So the models developed are not overfitted to a particular data set which would have been the case if deterministic thresholds were applied on the features like elevation of the surfaces. The probabilistic inference in BNs is also used to set the input parameters of segmentation algorithm for accurate identification of homogeneous surfaces in the scene. The input parameters are set in such a way that best classification results are obtained from the BNs. BNs are also used to fuse information from video imagery and Lidar data. Since it is a supervised classification method, uncertainties in the Global Positioning System (GPS) measurements and registration between visual data and Lidar data are handled well. Thus, 3D object identification process is automated along with the process of identification of homogeneous surfaces in the scene.

2. Measurement of geometric information - The initial geometry of the identified 3D buildings is available from the stereo mosaics. The accuracy of this geometry is improved by projecting the 3D building surfaces onto the individual video frames and optimizing the 3D positions of the corners of the building using iterative optimiza-
tion techniques like Levenberg-Marquardt (LM) algorithm [31].

The complete end-to-end system to reconstruct 3D CAD buildings from aerial video imagery is discussed in the next chapter.
Chapter 3

Approach

In this chapter, a complete approach that is used to extract and model 3D objects from aerial video frames is presented. The block diagram illustrating various steps involved in this process is shown in Figure 3.1. As mentioned in the research goals, the ultimate goal is to automate all the steps involved in this process while maintaining the speed and accuracy of the results obtained. The inputs to the developed system are the individual video frames captured by the airborne sensor, the exterior orientation parameters and the viewpoints of the sensor associated with each video frame. The other camera parameters like the focal length, detector size, position of the principle point, etc. are also assumed available to the user. In the first step, the video frames are pre-processed to remove the rotational effects of the sensor. At this point the video frames available seem to have been captured by a sensor with only translational motion. These video frames are stitched into three mosaics of which two of them (called left and right mosaics) form a stereo pair. Elevation map of the scene is obtained from the stereo mosaics. The third mosaic (also called nadir mosaic) is used as the base image for orthorectification and geo-referencing of extracted 3D points, lines and planes. 3D object reconstruction process from images involves two important parts: object identification and object modeling. The object identification part comprises of detecting and recognizing different objects in an urban scene like buildings, trees, grass, parking lots and terrain. This step uses the available though inaccurate elevation maps from different sources (like stereo mosaics and Lidar data) as well as other features like area, edge and corner information, color and texture information to classify different regions in to their respective object classes. Once the building surfaces are identified, we move on to object modeling part. This part involves local optimization of the 3D geometry of the identified man-made objects i.e. buildings. Even though elevation information is available from stereo mosaics, this step is essential because the accuracy of
the elevation map obtained from stereo mosaics depends on the view angle chosen for the fixed lines to build stereo mosaics (see Section 3.2). Each step is discussed in detail in the following sections.

Figure 3.1: Block Diagram of the approach used for modeling of 3D objects from aerial images

### 3.1 Pre-processing of Video Frames

Video frames are captured by an airborne sensor at different viewpoints and orientations. Different viewpoints are generated by translational motion of the aircraft while orientation at each viewpoint is determined by roll and heading of the aircraft. Frames with different orientations demonstrate displacement of objects in the scene not only due to motion parallax between different surfaces in the scene but also due to tilt of the sensor [32]. The basic idea of stereo geometry is to determine the heights of various surfaces by measuring the disparity exhibited by them from the baseline. Disparity is caused by motion parallax between surfaces of a scene. Accurate measurement of motion parallax is possible when displacement of the surfaces is caused only by translational motion of the sensor. To effectively deal with this problem, the video frames are corrected for rotation before they are stitched into mosaics.

The Exterior Orientation (EO) parameters of the camera are given by the navigational system installed aboard the aircraft along with the sensor. These include the position (center) of the camera in world coordinates \((X_C, Y_C, Z_C)\) and the orientation of the camera in terms of three angles \((\omega, \phi, \kappa)\). If an accurate navigational system is not available, there are commercial software programs which can perform bundle adjustment using the video frames, Digital Elevation Map (DEM) from any source (for example, Lidar data over the region), and ground control points and generate fairly accurate EO parameters for each
image. The sensor is calibrated for the Interior Orientation (IO) parameters prior to the flight using a target with known 3D points and commercially available camera calibration tools. Thus, focal length of the camera \( f \), offset of the principle point from the center of the focal plane \((\delta x_p, \delta y_p)\) and distortion parameters are obtained. We are assuming that the images are already corrected for distortion [32] and we are not going to discuss that particular aspect here. The size of each detector element \( d \) in the sensor is usually specified by the vendor. The relationship between world coordinate system described by \( X, Y, \) and \( Z \) axes and the image coordinate system described by \( x, y, \) and \( z \) axes is shown in Figure 3.2. Orientation correction matrix can be developed from this relationship which is mathematically described by collinearity equations [32].

An orientation matrix is developed in this dissertation assuming a right-handed, passive rotation system. Right-handed system implies that if the right hand thumb is pointing along any axis away from the origin, the rotation angle is positive in the direction in which the rest of the fingers curl. Passive rotation system implies that the angles are considered in such a way that it is the axes and not the vectors of points that are being rotated by given angles. The rotations are relative, which suggests that first rotation would be about \( X \) axis by \( \omega \). The second rotation would be about the transformed \( Y \) axis \((\gamma_\phi)\) by an angle of \( \phi \) and the third rotation would be about the twice-transformed \( Z \) axis \((\zeta_{\omega\phi})\) by an angle of \( \kappa \). More information on orientation systems used in different applications can be found in [32] and [33]. The rotation matrix about each axis is developed with respect to

![Figure 3.2: Relationship between world coordinate system and image coordinate system](image)
3.1. PRE-PROCESSING OF VIDEO FRAMES

points and not with respect to axes. Reader should understand that every vector shown here represents a point in 3D space and not the axes. Initially, we start with a point in the world coordinate system and transform the point on to the image plane. Let the world point to be projected on to the image plane be \( P_{\text{world}} = (X, Y, Z)^T \). As the rotations are performed about the center of the camera given by \( T = (X_C, Y_C, Z_C)^T \), \( P_{\text{world}} \) has to be shifted to a coordinate system with \( T \) as origin, before being rotated. Once the point is shifted, the rotations \( \omega, \phi \) and \( \kappa \) are applied successively as shown in Equations (3.1) to (3.3). The rotation matrices are given in Equation (3.4).

\[
P_\omega = R_\omega( P_{\text{world}} - T )
\]

\[
P_{\omega\phi} = R_\phi P_\omega = R_\phi R_\omega( P_{\text{world}} - T )
\]

\[
P_{\omega\phi\kappa} = R_\kappa P_{\omega\phi} = R_\kappa R_\phi R_\omega( P_{\text{world}} - T ) = R( P_{\text{world}} - T )
\]

\[
R = R_\kappa R_\phi R_\omega =
\begin{bmatrix}
\cos \kappa & \sin \kappa & 0 \\
-\sin \kappa & \cos \kappa & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\cos \phi & 0 & -\sin \phi \\
0 & 1 & 0 \\
\sin \phi & 0 & \cos \phi
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 1 \\
0 & \cos \omega & \sin \omega \\
0 & -\sin \omega & \cos \omega
\end{bmatrix}
\] (3.4)

The Interior Orientation (IO) parameters can be put in a matrix form and used to project the rotated point vector on to the image or focal plane [11]. The relationship between image coordinates in pixels \((x_{im}, y_{im})\) and image coordinates in distance units \((x, y)\) can be seen in Figure 3.3. If there is no offset, the principle point is given by \(((M - 1)/2, (N - 1)/2)\) pixels. With the offset, the principle point of the image plane (pixels) is given by Equation (3.5). The negative sign in the y-coordinate is due to the fact that the \(y_{im}\) coordinate axis and \(y\) coordinate axis are in opposite direction to each other. Note that a scaling of \(1/d\) provides the offsets in terms of pixels.

\[
(x_p, y_p) = \left( \frac{M - 1}{2} + \frac{\delta x_p}{d}, \frac{N - 1}{2} - \frac{\delta y_p}{d} \right)
\] (3.5)

The \(z\) coordinate of any point on the focal plane is \(-f/d\) pixels, as it is in the negative \(Z\) direction (see Figure 3.2). The IO matrix is given by Equation (3.6). The homogeneous image coordinates of the world point \( P_{\text{world}} = (X, Y, Z) \) are given by Equation (3.6) [11].
3.1. PRE-PROCESSING OF VIDEO FRAMES

![Diagram showing image coordinates and coordinate systems](image)

**Figure 3.3:** Relationship between image coordinates in pixels and image coordinate system

\[
K = \begin{bmatrix}
-f/d & 0 & x_p \\
0 & f/d & y_p \\
0 & 0 & 1
\end{bmatrix}
\]  
(3.6)

\[
P_{im} = \begin{bmatrix}
kx \\
yy \\
k
\end{bmatrix}
= KR (P_{world} - T)
\]  
(3.7)

The first video frame is taken as the base image of the series and the rest of the frames are corrected so that they have the same orientation as the first frame. Or all the frames can be corrected to remove all the rotational effects in them. Consider any 3D point \(P_{world}\) mapped onto the first image \((i = 1)\) of the series. As it is considered as the base image, rotation matrix will be an identity matrix. For any other frame \(i\), the rotation angles are given by \(\Delta \omega_i = \omega_i - \omega_1, \Delta \phi_i = \phi_i - \phi_1, \Delta \kappa_i = \kappa_i - \kappa_1\). The viewpoints of the sensor for frame 1 and frame \(i\) are located at \(T_1\) and \(T_i\) respectively. The image coordinates in both the frames are given by Equations (3.8) and (3.9). After going through the transformations shown in Equations (3.10) and (3.11), we can see that the image coordinates in frame 1 and transformed frame \(i\) differ only by translational vector \((T_i - T_1)\). Thus we have successive images with just translational motion of the sensor.

\[
P_{im_1} = K(P_{world} - T_1)
\]  
(3.8)

\[
P_{im_i} = KR_i (P_{world} - T_i) = R_{\Delta \kappa_i} R_{\Delta \phi_i} R_{\Delta \omega_i} (P_{world} - T_i)
\]  
(3.9)

\[
R_i^{-1}K^{-1}P_{im_i} = (P_{world} - T_i)
\]  
(3.10)
3.2 Ray Interpolation

Ray interpolation is explained in detail and in its most generalized form in [10]. But for the completeness of this thesis, we will go through only the basic steps of the algorithm for the cases involving 2D and 3D translation of the sensor. Before moving on to the fast algorithm, we provide a basic understanding of the geometrical set-up and the mathematics involved in this method.

3.2.1 Geometry

In [10], the authors emulate a linear pushbroom camera using a pin-hole video camera. A linear pushbroom camera executes a parallel-perspective projection of the real world points on to the focal plane - parallel projection along the dominant motion direction of the sensor (along-track) and perspective projection in the perpendicular direction to the motion of the sensor (across-track) [34]. But a video camera executes a perspective-perspective projection of the world points on to the focal plane. The basic purpose of the PRISM algorithm is to convert the perspective-perspective projection into a parallel-perspective projection so that we can use the apparent motion parallax of the objects to extract 3D structure. Figure 3.4 shows the projections through a linear pushbroom camera (a) and a video camera (b).

\[ P'_{im_j} = AP_{im_j} = KR_i^{-1}K^{-1}P_{im_j} = K(P_{world} - T_i) \] (3.11)
A video camera is attached to the aircraft and flown over the scene. Images are captured by the camera at different viewpoints and orientations. Each viewpoint of the camera has 6 DOF (Degrees Of Freedom) - 3 degrees for the camera center which defines the position of the camera in the world coordinate system and 3 degrees for the orientation of the camera which defines the angles made by the camera coordinate system with respect to the world coordinate system. The camera center \((T_x, T_y, T_z)\) (same as the camera center \((X_C, Y_C, Z_C)\) in Section 3.1) and the three orientation angles \((\omega, \phi, \kappa)\) with respect to world \((X, Y, Z)\) axes at each viewpoint of the sensor are obtained from a navigational system aboard the aircraft. These are called the extrinsic camera parameters or the exterior orientation parameters. All the video frames are rectified to remove the rotational effects of the sensor on the images before applying ray interpolation. The orientation correction procedure is explained briefly in [10] and in detail in [9]. Usually the average flying height of the aircraft is very large compared to the change in the flying height and relief change in the scene. So the translational motion of the sensor can be considered to be two dimensional, ignoring the changes in the third dimension \((Z)\). The new algorithm is explained for the case of 2D translational motion of the camera initially and is extended to 6 DOF generalized sensor motion later on in Section 3.2.5.

A fixed line is defined at a distance of \(d_y/2\) on every frame (i.e. at every viewpoint). This is the scanline of a pushbroom camera i.e. this is the angle at which the imaginary linear pushbroom camera is looking at the world at any viewpoint of the sensor. Hence it is called fixed line because the pixels on this line are directly copied on to the mosaics. One fixed line is defined on either side of the viewpoint (one for the left stereo mosaic and the other for the right stereo mosaic) as shown in Figure 3.5. This view angle is user defined, and can be changed to have a more oblique view and less occlusion problem [10].

Let us consider the left mosaic for the sake of discussion. As we are aware of the translation parameters of the camera between two successive frames, we can easily find
the overlapped region in the two frames. In order to stitch the two frames together, our region of interest in the overlapped region exists between the two fixed lines in the two successive frames as shown in Figure 3.6.

For every column of pixels in the overlapped region between the two fixed lines, an imaginary viewpoint is constructed by ray interpolating backwards using the existing viewpoints. The interpolation is carried out in such a way that at that viewpoint of the sensor, that column of pixels is the fixed line or the imaginary pushbroom sensor’s scanline. The interpolated viewpoint \( (T_{xi}, T_{yi}) \) is given by Equation (3.12).

\[
T_{yi} = T_y + \frac{y_1 - d_y/2}{y_1 - y_2} S_y, \quad T_{xi} = T_x + \frac{y_1 - d_y/2}{y_1 - y_2} S_x, \quad (3.12)
\]

where \( (T_x, T_y) \) is the initial viewpoint or the initial position of the sensor, \( (T_x + S_x, T_y + S_y) \) is the next viewpoint of the sensor as shown in Figure 3.6, \( (x_1, y_1) \) and \( (x_2, y_2) \) are the positions (on the focal plane) of the corresponding points in the overlapped region in first and second frames respectively. The projection of the interpolated viewpoint on the focal plane \( (t_{xi}, t_{yi}) \) is given by Equation (3.13).

\[
t_{xi} = \frac{F T_{xi}}{H}, \quad t_{yi} = \frac{F T_{yi}}{H}, \quad (3.13)
\]

where \( F \) is the focal length of the sensor and \( H \) is the average distance between the sensor and the scene. Since we are assuming that the airborne sensor has only 2-D translation \( H \) is nothing but the flying height of the sensor above the scene (Z direction). The coordinates of the pixel \( (x_i, y_i) \) in the frame, corresponding to the interpolated viewpoint of the camera, are given by Equation (3.14). The final coordinates of the point on the left mosaic \( (x_l, y_l) \) are given by Equation (3.15). The geometry can be seen in Figure 3.7. Right stereo mosaic is also built in a similar fashion.
If the flying height is not very large compared to the change in the Z dimension of the scene, the third dimension of the sensor translation cannot be ignored with this data sequence. The new triangulation presented in this section can be extended for use even with generalized motion of the sensor with 6 DOF. Using the available Exterior Orientation (EO) parameters, the orientation of all the video frames is corrected and the images are rectified in such a way that they are absolutely nadir as explained in [10] and [9]. After the image rectification, motion of the sensor is limited to 3D translational motion. Ray interpolation technique to stitch parallel-perspective stereo mosaics for 3D translational motion of the sensor is also described in [10]. The geometry needed for such a sensor motion is an extension to that described in Section 3.2.1. Similar to Equation (3.12), the three dimensional interpolated viewpoint \((T_{xi}, T_{yi}, T_{zi})\) is given by Equation (3.16) [10].

\[
T_{yi} = T_y + \left( \frac{y_1 - \frac{d_y}{2}}{y_1 - y_2} \right) \left( \frac{FS_y - y_2 S_z}{FS_y - \frac{d_y}{2} S_z} \right) S_y, T_{zi} = T_z + \frac{S_z}{S_y} (T_{yi} - T_y), T_{zi} = T_z + \frac{S_z}{S_y} (T_{yi} - T_y)
\]

where \(T_z\) is the Z coordinate of the initial viewpoint of the sensor, \(T_z + S_z\) is the Z coordi-
nate of the next viewpoint of the sensor. The coordinates of the pixel \((x_i, y_i)\) in the frame, corresponding to the interpolated viewpoint of the camera, are given by Equation (3.17). The final coordinates of the point on the left mosaic \((x_l, y_l)\) are given by Equations (3.13), (3.15), (3.16) and (3.17).

\[
x_i = \frac{x_1 - FS_y Z_i}{1 - \frac{S_z}{Z_i}}, y_i = \frac{dy}{2}, Z_i = \frac{FS_y - \frac{dy}{2}S_z}{y_1 - \frac{dy}{2}}
\]  

(3.17)

3.2.2 Existing Fast PRISM Algorithm

It would be computationally intensive to compute the mosaicing coordinates of all the individual pixels in the overlapping region. So a fast approach to the same algorithm has been suggested in [10]. The steps in the fast PRISM algorithm are as follows:

1. To make sure that the mosaic is seamless any two successive frames are stitched at the center of the overlapped region. For any two successive frames - frame \(k\) and frame \(k + 1\), the fixed lines are chosen as described in Section 3.2.1.

2. To stitch the two frames at the center of the overlapped region, stitching lines are chosen at the center of the overlapped region. Ideally, without any motion parallax, these stitching lines must be straight lines in both the frames. The stitching lines would be located at \(dy/2 + (t_y^{k+1} - t_y^k)\) in frame \(k\) and at \(dy/2 - (t_y^{k+1} - t_y^k)\) in frame \(k + 1\). Control points are chosen at regular intervals on the stitching line in frame \(k\).

Now corresponding control points are found on the stitching line in frame \(k + 1\). The approximate positions of the control points in frame \(k + 1\) are given by the translation parameters of the sensor. For every control point \(P_i^k, i = 1, 2, \ldots\) in frame \(k\), the approximate position of the corresponding control point \(P_i^{k+1}, i = 1, 2, \ldots\) in frame \(k + 1\) is given by \(P_i^{k+1} = P_i^k + (t_x^{k+1} - t_x^k, t_y^{k+1} - t_y^k)\). The accurate positions of the corresponding points are found using a matched filter. A small window of the gradient map around the control point in frame \(k\) is used as the correlation filter. The target image is a larger area of the gradient map around the approximate position of the corresponding point in frame \(k + 1\). As said earlier, if there is no motion parallax, the control points in the two frames must be separated by just the sensor’s translational parameters. Because of motion parallax, instead of a stitching line, we have a stitching curve (not a straight line) in frame \(k + 1\). These are called matching curves. The mosaic coordinates \(Q_i\) of the control points on the matching curve are found using Equations (3.12)-(3.15). So now we have a stitching curve through the interpolated control points on the mosaic.
3. We choose control points on the fixed lines in both the frames. The $y$ coordinate of these control points is determined by the fixed line. The $x$ coordinate of each control point is given by averaging the $x$ coordinates of two successive control points on the stitching lines. There are two sets of control points on the fixed lines: $R_{1i}, i = 1, 2, \ldots$ in frame $k$ and $R_{2i}, i = 1, 2, \ldots$ in frame $k+1$. These points are already on the fixed lines and translate on to the mosaics without any interpolation. As a result, the corresponding control points on the mosaic $S_{1i}$ and $S_{2i}$ are given by $R_{1i}$, $R_{2i}$, translation parameters $(t_x^k, t_y^k)$ of frame $k$ and $(t_x^{k+1}, t_y^{k+1})$ of frame $k+1$.

4. The first set of source triangles are formed by $R_{1i}$ and $P_k^i$ in frame $k$ while the corresponding destination triangles on the mosaic are formed by $S_{1i}$ and $Q_i$. The second set of source triangles are formed by $R_{2i}$ and $P_{k+1}^i$ in frame $k+1$ while the corresponding destination triangles on the mosaic are formed by $S_{2i}$ and $Q_i$. The source triangles are warped to the destination triangles using affine transform. Since the control points on the stitching curve $Q_i$ are common for both sets of destination triangles, the stitched mosaic is seamless.

The step by step process can be seen in Figure 3.8. All the features shown in the figures are marked for clarity of understanding and do not represent the real control points.

### 3.2.3 Problem with Fast PRISM

Fast PRISM works very well when the inter-frame displacement is small i.e. the overlapped region does not include a whole object like a building. But when the inter-frame displacement is large which is usually the case with airborne video, an entire building can fall into a single overlapped region. The problem with the method discussed in the previous section, is that the triangulation does not take the spatial features of the image into account. So if the vertices of any triangle lie on two surfaces with different motion parallax, the triangle will be warped in an unusual way on to the mosaic. The objects in the scene get disfigured because of the warping. For example, consider the situation illustrated in Figure 3.9. Here we have an imaginary helipad (H) over a background of a surface which is lower than the helipad surface. One of the vertices of a source triangle (marked red) is on the ground while one of the other two vertices of the same source triangle is on the helipad. As the helipad is at a greater height than the ground, when the scene is being imaged by an airborne sensor moving to the right, the helipad exhibits motion parallax and moves more to the left than the ground below it. Hence, the mosaic coordinates for these control points look as shown in Figure 3.10. As a result, the source
3.2. RAY INTERPOLATION

(a) Fixed Lines and Overlapped region
(b) Stitching lines and control points
(c) Matching curves
(d) Control points on the fixed lines
(e) Source triangles
(f) Destination triangles

Figure 3.8: Steps in fast PRISM algorithm

Figure 3.9: Motion parallax between a helipad which is at greater height than the ground

triangles from the two frames get warped in an unwanted fashion and generate visual artifacts as illustrated in Figure 3.10.
3.2.4 Segment-based Mesh Design

To avoid the visual artifacts, we have developed a new mesh-design technique which makes sure that none of the triangles cross over from one surface to another. In that way none of the affine transformations would warp the shapes of the objects in an undesired fashion. The steps of the algorithm and the method are explained in the following subsection. The basic idea of our algorithm is that two adjacent surfaces with different color or texture may be at different heights and hence may exhibit motion parallax. Even if the two surfaces of different color/texture are at the same height, it would just increase the number of computations but would not produce any adverse effects. Such a distinction between adjacent surfaces can be made using image segmentation.

One might wonder that edges can be easily extracted from the overlapped region and used for this purpose, but usually the edge detection algorithms detect a common edge between two surfaces. The control points on the common edge may belong to either of the two surfaces. The transformations for the triangles defined by these control points are determined based on their movement in successive frames. So the triangle belonging to one surface might get warped according to the parallax defined by the adjacent surface. For instance, consider the two surfaces in Figure 3.11. In Figure 3.11(a), edges are used to distinguish between the surfaces. So triangles in both the surfaces have a common control point on the common edge. If the edge belongs to surface 1, then the triangle in surface 2 gets warped on to the mosaic according to the movement of surface 1 which is exactly what we are trying to avoid. In Figure 3.11(b), the inner boundaries of the segments are used as edges of the surfaces. So the triangle in surface 2 has control points belonging
3.2. RAY INTERPOLATION

3.2.5 Modified PRISM Algorithm

1. The fixed lines and stitching lines are chosen on each of the two successive frames - frame $k$ and frame $k + 1$ as described in Sections 3.2.1 and 3.2.2 (Fig. 3.12(a)).

2. The overlapped region between the fixed line and stitching line in frame $k$ is segmented as shown in Fig. 3.12(b). Very effective segmentation algorithms can be found in [35], [36] and [37]. A convex hull [38] is formed around each segment to obtain the significant points which enclose the segment completely (Fig. 3.12(c)). However, readers should note that the segments themselves need not be convex in nature. This algorithm works on all surfaces without any constraints. We perform Delaunay triangulation [38] on these significant points (Fig. 3.12(d)). These triangles constitute our source triangles from frame $k$. The corresponding control points (or tie points as they are referred to in photogrammetry literature) in frame $k + 1$ for the vertices of the source triangles in frame $k$ are obtained using a matched filter just like in fast PRISM. The mosaicing coordinates of the vertices are obtained using ray

![Figure 3.11: Triangulation using edges and segments](image)
3.2. RAY INTERPOLATION

interpolation ((3.12)-(3.15)). These vertices form the destination triangles.

3. The mosaic is on a regularly sampled grid. It is a standard image processing tech-
nique to back project a pixel in the destination coordinate system (mosaic) using
inverse affine transformation to the source coordinate system (individual frames)
and use some kind of interpolation to obtain the source position (in frame \( k \)) from
where the image information for that particular pixel on the mosaic is acquired. In
our case, we are using nearest neighbor interpolation. But before the interpolation
step, these source positions are tested to find if they lie in the same segment as the
one we are processing now. Only if the source position lies in the same segment,
will the pixel in the mosaic be filled with information from frame \( k \). There are two
reasons for this test:

- To avoid using the affine transformation of source triangles on the pixels be-
  belonging to neighboring segments but enclosed by these source triangles (see
  Fig. 3.12(g)) which would negate the basic idea of our method.
- To prevent the present segment of the mosaic from obtaining information from
  neighboring segments. Due to noise in the images, errors made in matching
  the vertices of source triangles in the two successive frames result in errors in
  the positions of the vertices of destination triangles.

4. The vertices of all the source triangles that lie on the stitching line in frame \( k \) are
grouped together and ordered from top to bottom on the stitching line. The corre-
sponding points in frame \( k + 1 \) are similarly grouped and ordered. Piece-wise
linear interpolation is done between every two consecutive points in frame \( k + 1 \)
and all the corresponding points are connected using straight lines. This results in
the matching curve in frame \( k + 1 \) (Fig. 3.12(e)). This matching curve also forms one
boundary while the fixed line forms the other boundary for the overlapped region
in frame \( k + 1 \) to be stitched to the mosaic. Again Steps 2 and 3 (i.e. segmenta-
tion, triangulation and ray interpolation) are implemented on the region between
the matching curve and fixed line in frame \( k + 1 \) (see Fig. 3.12(e)) and frame \( k + 1 \) is
stitched to the mosaic. Since the control points on the stitching curve in the mosaic
are common to both frames \( k \) and \( k + 1 \), the stitched mosaic is seamless.

5. Some of the pixels are left vacant in Step 3 for the reasons explained in same step.
These are termed as “orphan” pixels (Fig. 3.12(h)). Segment information is not
available on the mosaics. We do not know which segment’s affine transformation
can be used to find the source pixel positions corresponding to these orphan pixels.
However, a constraint inherent to the basic method of ray interpolation is used to obtain information about these orphan pixels. The mosaics are parallel-perspective mosaics with perspective projection in $x$ direction and parallel projection in $y$ direction as shown in Fig. 3.6. Hence motion parallax is not considered in the $x$ direction while stitching the mosaics. Any row of pixels (same $x$ coordinate) in the mosaic stitched from a frame is obtained from a single row of pixels with same $x$ coordinate even in the frame. Now by rearranging the terms in (3.12)-(3.15), we get (3.18). All the terms in that equation are known except $(x_1, y_1)$. For every orphan pixel, the nearest filled pixel in the same row of the mosaic is found and the source coordinate $x_1$ for that pixel is obtained. Using (3.18), $y_1$ is found and thus the source point for the orphan pixel $(x_1, y_1)$ is obtained. The image information is picked from that source pixel in the video frame to fill the orphan pixel in the mosaic. A part of the final mosaic is shown in Fig. 3.12(i).

$$y_1 = y_l - t_y - (x_l - t_x - x_1) \frac{S_y}{S_x}$$

(3.18)

All the steps are illustrated with two frames of real airborne imagery in Figure 3.12. Again, all the features shown in the figures are marked only for clarity of understanding and do not represent the real control points.

Our modified PRISM algorithm can be applied to the 3D sensor translation case as explained above except for Step 5. The assumption we made in the Step 5 of our algorithm to fill the orphan pixels in the mosaics, no longer holds true because of change in the scale from frame to frame. So, for the case with three dimensional translational motion of the sensor, we make a small change in this last step. We use two-dimensional interpolation to determine the source pixel position in the original image from which the image content for the orphan pixel in the mosaic has to be picked. The reason for this is that the 2D piecewise linear interpolation provides a good approximation to the affine transformation that is being used to transform the images to stitch them into mosaics. Readers should understand that we are interpolating in the source pixel position space but not in the intensity value space in the mosaics.

There are two reasons for choosing the 2D piecewise linear interpolation to fill the orphan pixels. First, affine transformation is a linear transformation representing the scaling, rotation and translation of the source pixels in individual images to the destination pixels in the mosaic. If the orphan pixel in the mosaic is surrounded by the pixels belonging to the same surface (or segment), then the 2D linear interpolation accurately captures the affine transformation of the surface. Second, if the orphan pixels are between two
surfaces or near the edges of the surface, the 2D linear interpolation captures the transformations of both the surfaces while finding the corresponding source pixels. Thus, it provides a smooth transition from surface to surface in the mosaics. But, the orphan pixels do not necessarily exist in between regularly spaced grid points of the mosaic. To tackle this problem effectively, Sequential Linear Interpolation (SLI) technique described in [39] is used. The source pixel position is given by

\[
x_{s_0} = f \{ (x_{m_o}, y_{m_o}), (x_{m_i}, y_{m_i}), x_{s_i} \},\ y_{s_0} = g \{ (x_{m_i}, y_{m_i}), (x_{m_i}, y_{m_i}), y_{s_i} \}
\]

(3.19)
where \( (x_{so}, y_{so}) \) is the source pixel position corresponding to the orphan pixel \((x_{m_0}, y_{m_0})\) in the mosaic (left or right). \((x_{mi}, y_{mi}), i = 1, 2, 3, 4\) are the grid points on the mosaic picked according to the SLI technique [39] and \((x_{si}, y_{si}), i = 1, 2, 3, 4\) are their corresponding known source pixel positions. SLI technique implements a weighted average of function values \((x_{si}, y_{si})\) at known grid points \((x_{mi}, y_{mi})\) to determine the function value at the unknown position \((x_{m_0}, y_{m_0})\). The weights are proportional to the distances between the grid points \((x_{mi}, y_{mi})\) and \((x_{m_0}, y_{m_0})\), thus capturing the surrounding transformation(s) while interpolating for the source pixel positions.

### 3.3 3D Object Identification

Once the stereo mosaics are built, the objects in the scene, especially man-made buildings can be identified and modeled in three dimensions. Elevation information can be obtained from the pair of stereo mosaics. The 2D features important to construct 3D models of objects like the edges and corners are extracted using nadir mosaic. For this purpose, the nadir mosaic is segmented. Building surfaces can be identified by applying a threshold on the elevation map extracted from the stereo mosaics. Edges and corners of each of these building surfaces are obtained from the boundary of its segment in the nadir mosaic. These 2D corner points are then geo-referenced using the elevation information as explained in Appendix C. Finally the geo-referenced 3D corners (in object space) of all the surfaces of each building are put together to produce the complete 3D model of the building. This process is explained in [9] and [40]. There are two major problems that have been observed while identifying and modeling 3D buildings using the method described above. They are as follows:

1. Inaccuracy in 3D geometry: The elevation map obtained from a single pair of stereo mosaics is sparse. For a denser elevation map, more pairs of stereo mosaics should be built at varying view angles of the fixed line. But that process increases the number of computations to be done. The resolution of the elevation map extracted from the stereo mosaics also depends on the view angle of the fixed line. A single pixel of disparity leads to a change of \((H/d_y)\) units of elevation, where \(H\) is the average flying height of the airborne sensor over the terrain and \(d_y\) is the distance between the two fixed lines (in pixels) used while building the stereo mosaics.

2. Problem with vision algorithms: Different homogeneous surfaces in a scene are identified using segmentation algorithms. The boundaries of each of these surfaces enable vision algorithms to identify the edges and corners of those surfaces. The
efficacy of these vision algorithms may be seriously compromised due to noise in the images, solar angle and other parameters like sensor position and orientation.

It is very difficult to identify the different surfaces in a scene and group the surfaces belonging to a particular building just by applying hard thresholds on the elevation information and the number of corners or by applying constraints on the angles at the corners (like right angles only). These problems have been identified and presented in detail in [9] and [40]. In order to deal with these issues and achieve the second and third research goals, a method based on Bayesian networks and local optimization is proposed in the next few subsections.

As discussed previously in the background section and above, the deterministic feature operators cause the models to be overfitted to the data and hence lead to problems when the same algorithm is applied on other data sets. The uncertainty in data like sensor position, solar angle and noise in the images also lead to wrong hypotheses if only passive video imagery is used. There is also a possibility of obtaining more evidence/data about the scene from other sources like still-image sensors which can capture images of the same scene at a different time and active sensors like Lidar sensor which would provide accurate 3D information in the form of a point cloud. In order to deal with this problem, the building hypotheses need to be verified with all the available data. A 3D reconstruction system can be considered as an intelligent system trying to decipher different objects in the scene. This system basically tries to emulate human beings such as ourselves who can very easily perform this task due to the extensive training we receive throughout our lives. Similarly, machines need to be trained to perform this task.

The different homogeneous surfaces in the nadir mosaic are identified by using segmentation algorithm. Every segmentation algorithm requires certain input parameters to be specified to perform well on a data set. This problem can be solved by humans by looking at the data and manually setting the input parameters. But this is a tedious process and it is not trivial to decide on these parameters just by visual confirmation when a lot of data is involved. Even after the segmentation input parameters required to identify homogeneous surfaces are fixed according to the data set, if deterministic inference is used (on the features extracted from these homogeneous surfaces) for object classification, the thresholds will hold only for a particular data set and will fail for others.

Bayesian Networks (BN) provide a powerful tool which can be trained like human beings and avoid overfitting the 3D models to the data. At the same time, BNs can fuse different data sets from different sensors and also handle the uncertainty in the data well.
3.3.1 Theory of Bayesian Networks

A Bayesian network is a graphical model in which each node is annotated with quantitative probability information [41], [42], [43], [44]. It provides a graphical representation of the joint probability distribution table of many variables by taking the dependence among these different variables into consideration. The advantages of Bayesian networks are as follows [43]:

1. The dependencies among all variables are encoded in the Bayesian network. It is easier computationally, to infer from the Bayesian network than the joint probability distribution table.

2. A Bayesian network is encoded with causal semantics along with probabilistic semantics. It is easier to combine prior knowledge with data.

3. This cause-effect relationship between the variables can provide a better understanding of the problem domain (for instance, 3D object modeling). It is easier for an expert to intervene and predict the effects of such an intervention.

4. Bayesian networks along with Bayesian statistical methods provide a good method to avoid overfitting of data.

Properties of Bayesian networks [44]:

1. A set of random variables form the nodes of the Bayesian network.

2. Nodes are connected to other nodes using arrows. If X is connected to Y, X is parent of Y. X has a direct influence on Y. The causal relationship between the node/variables X and Y can be modeled by making X as the parent of Y.

3. Any node $X_i$ has a conditional probability distribution given its parents ($P(X_i|\text{Parents}(X_i))$). The joint probability distribution of all the random variable/nodes can be derived from the knowledge of only these small set of conditional probability distributions. The parents of $X_i$ are the nodes/variables which have a direct causal effect on $X_i$.

4. There are no directed cycles in the Bayesian networks. So it is called Directed Acyclic Graph (DAG).

The probability (conditional or prior) associated with each node is known as a parameter associated with the node, at least in our case in which each node assumes only a finite number of values and whose probabilities only have their minimal constraints. Bayesian probabilities consider the uncertainty in these parameters while estimating them.
from data. All child nodes are conditionally independent given their parent nodes. Hidden nodes can be introduced in BNs to make causal dependencies explicit or to satisfy the constraints of conditional independence [45]. Since no information is available about these hidden nodes during the training phase, Expectation-Maximization (EM) algorithm is used in the present work to learn the parameters of the BNs with hidden nodes ([46], [47], [43] and [44]). This process is explained in detail later on in Section 3.3.4.

In this section, an algorithm for using probabilistic inference to determine input segmentation parameters and to identify 3D objects from aerial video imagery is described. In order to improve the accuracy of the identification process, information from Lidar data is fused with the visual imagery in a BN. This algorithm consists of two main parts: First, design of BN structure using expert knowledge and computer vision algorithms to extract different features being used in the BN and second, Bayesian analysis (learning of BN parameters and inference) of these features for classification.

### 3.3.2 Bayesian Network Structure

The Bayesian Network handcrafted with expert knowledge of the author for 3D object identification is shown in Figure 3.13. This network is designed to use information from both visual imagery (stereo mosaics) and Lidar data. The nodes/features are listed in Table 3.1. The number of states in which each feature can occur are also given in the same table. The BN designed to use information from only visual imagery is shown in Figure 3.14. The nodes and number of states of each node are listed in Table 3.2. Two hidden nodes are introduced in these networks to describe the ”goodness” of the geometrical plane fit from visual information and Lidar information. These nodes are given two states - good fit and bad fit.

Considerable amount of data is required to train the BNs so that they perform well. We generated the training data using the DIRSIG tool available at RIT. There are three parts to the data generated using this tool.

1. Aerial video frames with the camera position and orientation parameters for each frame
2. Lidar point cloud
3. Material map which is used to generate the truth map for training and testing the BN

The other camera parameters like focal length, size of the detector can be given as inputs to the tool. More information on how to use the DIRSIG tool can be found in [1] and
### Table 3.1: List of nodes/features used in BN (with visual and Lidar information)

<table>
<thead>
<tr>
<th>Node no.</th>
<th>Node description</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Region (Building, Grass, Trees, Asphalt and Misc.)</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Visual plane fit (hidden node)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Lidar plane fit (hidden node)</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Ratio of no. of right angle corners to total no. of corners</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Ratio of no. of 45°/135° angle corners to total no. of corners</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Average height of the surface (from stereo mosaics)</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Minimum height of the surface (from stereo mosaics)</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>Maximum height of the surface (from stereo mosaics)</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>Percentage of inliers during plane fit (from stereo mosaics)</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>Average height of the surface (from Lidar data)</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>Minimum height of the surface (from Lidar data)</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>Maximum height of the surface (from Lidar data)</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>Percentage of inliers during plane fit (from Lidar data)</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>Area of the region</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>Average hue of the region</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>Average saturation of the region</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>Average entropy of the region (from nadir mosaic)</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>Average entropy of the region (from Lidar data)</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>Maximum edge length</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>Minimum edge length</td>
<td>5</td>
</tr>
<tr>
<td>21</td>
<td>Total number of corners</td>
<td>4</td>
</tr>
</tbody>
</table>

The process of extraction of various features from visual imagery (Stereo mosaics and Nadir mosaic) and from Lidar data is explained in the next subsection.

#### 3.3.3 Feature Extraction

Some of the original video frames of a scene simulated using DIRSIG can be seen in Figure 3.15. This video was collected by a sensor flying at an average height of 182 m over the terrain. The distance between the two fixed lines (for ray interpolation) is chosen to be 70 pixels. The left and right mosaics are shown in Figure 3.16. Figure 3.17 presents the stereo mosaic pair in red (left) - cyan (right) anaglyph form. The elevation changes due to the disparity between the left and the right mosaics at any elevation other than 182 m can be observed by using red-cyan stereo/anaglyph glasses. As mentioned earlier in this chapter, nadir mosaic (fixed line angle = 0°) is built along with left and right mosaics. It (shown in Figure 3.18) is used as the base image for building the CAD models since it is very easy to orthorectify and geo-reference 3D points using it (see Appendix C).

Since it is difficult to represent the 3D point cloud of Lidar data using 2D images, only the elevation coordinates (Z) of Lidar data have been rasterized and shown in Figure 3.19.
Figure 3.13: Handcrafted structure of Bayesian network to fuse visual and Lidar information
Figure 3.14: Handcrafted structure of Bayesian network to use visual information only

Figure 3.15: Original video frames captured by an airborne sensor

The raster/grid is generated by projecting the Lidar point cloud on to a plane at a distance of 182 m (the average flying height of the imaginary linear pushbroom camera used to
### 3.3. 3D OBJECT IDENTIFICATION

<table>
<thead>
<tr>
<th>Node no.</th>
<th>Node description</th>
<th>States</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Region (Building, Grass, Trees, Asphalt and Misc.)</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Visual plane fit (hidden node)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Ratio of no. of right angle corners to total no. of corners</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Ratio of no. of $45^\circ / 135^\circ$ angle corners to total no. of corners</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Average height of the surface (from stereo mosaics)</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Minimum height of the surface (from stereo mosaics)</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Maximum height of the surface (from stereo mosaics)</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>Percentage of inliers during plane fit (from stereo mosaics)</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>Area of the region</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>Average hue of the region</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>Average saturation of the region</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>Average entropy of the region (from nadir mosaic)</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>Maximum edge length</td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>Minimum edge length</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>Total number of corners</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.2: List of nodes/features used in BN (with visual information only)

![Figure 3.16: Left and right mosaics built from airborne video imagery](image)

(a) Left mosaic

(b) Right mosaic

Build stereo mosaics) at a nadir angle. The projections are carried out as though the points have been imaged by a linear pushbroom camera. The viewpoints of this pushbroom camera are same as the interpolated viewpoints generated during ray interpolation.

An important advantage of DIRSIG tool is that it can generate a material ID for each of
the image points in the object space along with the video frames. It can generate an entire strip which has accurate material IDs for all the points in the nadir mosaic. This material map in its original form can be seen in Figure 3.20. This material map is converted into a truth map required for our training. Figure 3.21 displays the truth map after relabeling different regions according to Table 3.3.

In order to extract the features of the objects in the scene like corners and edges, dif-
different planar surfaces in the image have to be recognized. Hence segmentation is used to identify all the homogeneous surfaces in the image. There are many efficient color segmentation algorithms in the literature ([35], [36], [37] and [48]). From here on, any reference to segmentation is meant as the mean-shift image segmentation algorithm described in [48]. Mean shift algorithm is used to cluster multimodal space by detecting the modes of the density function used to represent the data. Yet, this technique does not make any assumption about the shape of the density function. Mean shift image segmentation algorithm is a very useful low-level image processing tool. As explained in [48], there are three user-defined parameters that are given as inputs to the segmentation algorithm. The first is the spatial bandwidth \( h_s \) which determines the size of the kernel in spatial domain. All the points within this kernel will be grouped together as one cluster. The second is the color bandwidth \( h_c \) which determines the size of the kernel in range/color domain. The third is the minimum spatial size \( M \) of each cluster. Any cluster of size smaller than \( M \) will be eliminated by merging in to the closest cluster. The minimum spatial cluster size is kept fixed at 20 pixels. The segmentation algorithm is not very sensitive to the spatial bandwidth [48] and hence it is set at a level used in the original paper i.e. \( h_s = 15 \). The segmentation algorithm is quite sensitive on the other hand, to the color bandwidth due to varying textures and small variations in natural images like aerial images. The choice of this parameter will always depend on the final goal of the application in which this algorithm is being used. An algorithm to automatically set the color bandwidth parameter is explained later on in this chapter during the discussion of inference of BN. But for the purpose of feature extraction description we have used a color bandwidth of \( h_c = 10 \).

Segmented left, right and nadir mosaics and their respective segment maps are shown in Figures 3.22 and 3.23.

3.3.3.1 Elevation Information from Visual Imagery

Elevation from visual imagery is obtained from the stereo mosaics that were stitched from individual video frames. From [10], the mathematical model of the generalized parallel-
3.3. 3D OBJECT IDENTIFICATION

(a) Left mosaic

(b) Right mosaic

(c) Nadir mosaic

Figure 3.22: Segmented mosaics \((h_s = 15, h_c = 10, M = 20)\)

perspective stereo mosaics is given by Equations 3.20 and 3.21, where \((x_l, y_l)\) and \((x_r, y_r)\) are the corresponding point pair (from left and right mosaic respectively), \((X, Y, Z)\) is the 3D point in the object space, \((T_{xl}, T_{yl}, T_{zl})\) and \((T_{xr}, T_{yr}, T_{zr})\) are the camera viewpoints for the columns in the left and right mosaics respectively where the point is visible. The focal length of the camera is \(F\) and the average flying height of the camera over the average terrain is \(H\). The complete derivation of these equations along with illustration of geometry is presented in Appendix A.

\[
(x_l, y_l) = \left( F \frac{X - T_{xl}}{Z - T_{zl}} + F \frac{T_{xl}}{H} - \left( \frac{Z - T_{zl}}{H} - 1 \right) \frac{d_y}{2} \right)
\]

(3.20)

\[
(x_r, y_r) = \left( F \frac{X - T_{xr}}{Z - T_{zr}} + F \frac{T_{xr}}{H} + \left( \frac{Z - T_{zr}}{H} - 1 \right) \frac{d_y}{2} \right)
\]

(3.21)

If there is a large variation in the \(Z\) translational component of the camera motion, then the scale of the corresponding points will be different in left and right mosaics. But if the variation is small as is usually in aerial imagery, then the scales are same and hence the stereo mosaics can be used for stereoscopic viewing. This also makes the stereoscopic matching by automated algorithms simple. The mosaic displacement between two corre-
3.3. 3D OBJECT IDENTIFICATION

Figure 3.23: Segment maps of the mosaics ($h_s = 15, h_c = 10, M = 20$)

Sponding points in left and right mosaic can be defined as

$$(\Delta x, \Delta y) = (x_r - x_l, y_r - y_l)$$

(3.22)

The depth of this point can be derived from Equations 3.20 and 3.21 as

$$Z = H \left(1 + \frac{\Delta y}{d_y}\right) + T_z$$

(3.23)

where $T_z$ is the average deviation of the camera height given by

$$T_z = \frac{T_{zl} + T_{zr}}{2}$$

(3.24)

The scaled baseline functions of the stereo mosaics in $x$, $y$ and $z$ directions are given by Equations 3.25, 3.26 and 3.27 respectively. For any point at a depth of $H$ (average flying height of the camera), mosaic displacement $\Delta y = 0$. There is only a disparity of $d_y$ between the left and right mosaics. This plane at which there is zero disparity is called the fixation plane. At the fixation plane, the scaled baseline functions transform to Equations
3.3. 3D OBJECT IDENTIFICATION

3.28, 3.29 and 3.30. The elevation of any point from this fixation plane is given by Equation 3.31. Here \((t_{xl}, t_{yl}, t_{zl})\) and \((t_{xr}, t_{yr}, t_{zr})\) are the projections of camera viewpoints \((T_{xl}, T_{yl}, T_{zl})\) and \((T_{xr}, T_{yr}, T_{zr})\) on the focal plane of the imaginary linear pushbroom camera and hence, can be called as scaled camera viewpoints.

\[
b_x = t_{xr}(y_r) - t_{xl}(y_l) = t_{xr}(y_l + \Delta y) - t_{xl}(y_l) = t_{xl}(y_l + d_y + \Delta y) - t_{zl}(y_l) \tag{3.25}
\]

\[
b_y = t_{yr}(y_r) - t_{yl}(y_l) = t_{yr}(y_l + \Delta y) - t_{yl}(y_l) = t_{yl}(y_l + d_y + \Delta y) - t_{yl}(y_l) \tag{3.26}
\]

\[
b_z = t_{zr}(y_r) - t_{zl}(y_l) = t_{zr}(y_l + \Delta y) - t_{zl}(y_l) = t_{zl}(y_l + d_y + \Delta y) - t_{zl}(y_l) \tag{3.27}
\]

\[
b_x = t_{xr}(y_r) - t_{xl}(y_l) = t_{xr}(y_l) - t_{xl}(y_l) = t_{xl}(y_l + d_y) - t_{zl}(y_l) \tag{3.28}
\]

\[
b_y = t_{yr}(y_r) - t_{yl}(y_l) = t_{yr}(y_l) - t_{yl}(y_l) = t_{yl}(y_l + d_y) - t_{yl}(y_l) \tag{3.29}
\]

\[
b_z = t_{zr}(y_r) - t_{zl}(y_l) = t_{zr}(y_l) - t_{zl}(y_l) = t_{zl}(y_l + d_y) - t_{zl}(y_l) \tag{3.30}
\]

\[
h = H - Z = -H \frac{\Delta y}{d_y} - T_z \tag{3.31}
\]

In order to find the corresponding point pairs in stereo mosaics, the epipolar geometry of the parallel-perspective stereo mosaics is required. The epipolar constraints of parallel-perspective stereo images are quite different from the epipolar constraints of the perspective-perspective stereo images. This difference is explained in detail in Appendix B where the derivations of the epipolar constraints for 3D, 2D and 1D translational motion of the airborne sensor are also presented. Equations 3.32, 3.33 and 3.34 show the constraints for 3D, 2D and 1D case respectively. For any point in the left mosaic, the corresponding point in the right mosaic will be on an epipolar curve defined by [10]

\[
\Delta x = \frac{b_x \Delta y + b_z d_y \left( x_l - \frac{t_{zl} + t_{zr}}{2} \right) / F}{\Delta y + d_y - b_z d_y / 2F} \tag{3.32}
\]

\[
\Delta x = b_x \frac{\Delta y}{d_y + \Delta y} \tag{3.33}
\]

\[
\Delta x = 0 \tag{3.34}
\]

Once the 3D stereo geometry and epipolar constraints have been determined, the depth/elevation map can be extracted from the stereo mosaics. First the boundary of each of the segments in the nadir mosaic is extracted and the corresponding putative matches
are found in left and right stereo mosaics using correlation. Correlation or matched filter technique is sufficient to find the putative matches since there is only dominant 2D translation between the mosaics. Then epipolar constraints are applied to remove the bad matches and keep only the good matches between the 3 mosaics. Elevation at each of the boundary point on the surface is found using the Equation 3.31. This elevation map for that particular surface is further optimized using the RANSAC (RANdom SAMple Consensus) plane fit algorithm [11] in which a least squares plane equation of the form shown in Equation 3.35 is fit to the existing 3D points (height information and 2D position in nadir mosaic) while removing the outliers. In this equation, \((x_n, y_n)\) represent the coordinates of each boundary point in the nadir mosaic and \((B1, B2, B3, B4)\) are the coefficients of the plane equation. The height information for all the pixels on the boundary of the surface, is found using this plane fit as shown in Equation 3.36.

\[
B1(x_n) + B2(y_n) + B3(h) + B4 = 0 \tag{3.35}
\]

\[
h_{fit}(x_n, y_n) = -\frac{(B4 + B1(x_n) + B2(y_n))}{B3} \tag{3.36}
\]

In the final step, orthorectified and geo-referenced 3D points corresponding to the boundary pixels are obtained as shown in Appendix C where \(h_{fit}(x_n, y_n)\) is used as the height of each point \((x_n, y_n)\). An example elevation map of the scene is shown in Figure 3.24. Notice that it is noisy because we have not yet set the optimal segmentation parameters. The elevation map is dependent on the boundary of each surface which in turn is dependent on the segmentation parameters. We obtain the average height, minimum height and maximum height of each surface. These 3 features form nodes in our BN. We also use the ratio of the final number of inliers after RANSAC plane fit to the total number of boundary points of that segment/surface as another node in the BN. The features of minimum height, maximum height and average height will help in classifying building and tree regions from terrain (grass and asphalt). The number of inliers during plane fit will help in classifying buildings from trees since building surfaces will have more inliers than the tree surfaces.
3.3. 3D OBJECT IDENTIFICATION

3.3.3.2 Elevation Information from Lidar Data

In order to fuse the information from Lidar data with that from the visual imagery, Lidar data has to be registered with the visual imagery which is nadir mosaic in this case. The knowledge of camera viewpoint \((T_{xi}, T_{yi}, T_{zi})\) for each column of the nadir mosaic makes this registration possible. The depth of the fixed baseline at which the stereo mosaics have zero mosaic displacement is \(H\). If the nadir mosaic is assumed to be a flat surface at the fixed baseline, the position of each pixel in the object space at the fixed baseline is also known. A 2D grid of size same as that of the nadir mosaic is created and hence the georeferenced position of each grid point in this 2D grid is known. Each Lidar data point can be projected on to the fixed baseline and the grid point on which it falls can be determined using Equation 3.37. Any 3D Lidar data point \((X, Y, Z)\) when projected using the camera viewpoint \((T_{xi}, T_{yi}, T_{zi})\) falls at \((X_g, Y_g)\) on the fixed baseline and is given by

\[
X_g = T_{xi} - \frac{T_{xi} - X}{T_{zi} - Z} H, Y_g = T_{yi}
\]  

(3.37)

This projection which is exactly opposite to the orthorectification process can be seen in Figure 3.25. The elevation due to Lidar data at each grid point on the 2D grid and hence on the nadir mosaic is set equal to the elevation corresponding to the Lidar data point which was projected on that grid point. If two or more projected points occupy the same grid point, then the Lidar point with the maximum height is taken into consideration. Thus the Lidar data points corresponding to each of the pixels in the nadir mosaic are determined. Once this is done, for each surface in the nadir mosaic, all the 3D Lidar points lying inside that surface (on the 2D grid) are used and a 3D RANSAC least squares plane fit [11] is performed. This will remove the outliers (which exist because of inaccurate segmentation and registration). We can obtain the average height, minimum height, maximum height and percentage of inliers for that surface. These are used as features corresponding to Lidar data. For example, Figures 3.18 and 3.19 show the nadir mosaic and rasterized Lidar data registered to the nadir mosaic. As in the case of nodes corresponding to the elevation information from visual imagery, the nodes of minimum height, maximum height and average height from Lidar data will help in classifying building and tree regions from terrain (grass and asphalt). The number of inliers during plane fit will help in classifying buildings from trees since building surfaces will have more inliers than the tree surfaces.
3.3.3 Edge and Corner Information from Visual Imagery

The edge and corner information are found using a modified version of the algorithm written by Peter Kovesi [49]. First, the boundary of each of the surfaces is extracted by using morphological image processing technique of erosion and subtraction. Once the boundary pixels of the surface are identified, this algorithm traces through the boundary and identifies all the pixels at which the boundary deviates from a straight line. The amount of deviation that can be tolerated is given as an input to the algorithm depending on the noise in the image. These pixels are deemed as corners. Figure 3.26 displays the important steps in corner detection for a surface. These corners are orthorectified using the elevation information from the stereo images. Edges can be easily obtained by fitting a line passing through two consecutive corners in the object space. The angles at each of these corners can also be found. Maximum edge length (in m), minimum edge length (in m), total number of corners, ratio of number of L corners (corners with right angles) to total number of corners, and ratio of number of 45°/135° corners to total number of corners are used as features in the BN. The nodes related to edge length can be used to differentiate between buildings and other classes like trees and grass. Similarly, buildings have clearly defined L corners or 45°/135° corners while the surfaces belonging to other classes do not. Since, majority of the corners of polyhedral building surfaces have to be L/45°/135° corners, the ratio of the number of these corners to total number of corners in case of buildings must be high compared to the ratio of the same in other cases like trees or grass.
3.3.3.4 Surface Area from Visual Imagery

The orthorectified plane equation for each surface can be used to find the area of the surface. At each pixel on the surface in the nadir mosaic, the absolute area covered by that pixel is given by \((Z/F)^2\) where \(Z\) is the distance between the camera position and the orthorectified point represented by the pixel in object space and \(F\) is the focal length of the camera. Thus the total area of the surface is obtained by adding together the area covered by all the pixels in the surface. Even though this is an approximation of the true area, it serves as a good estimate and can be used as a feature in the BN since the same feature operator is used over all the data sets. Surface area corresponding to grass or asphalt regions is very high compared to building surfaces which is in turn high compared to
3.3. 3D OBJECT IDENTIFICATION

3.3.3.5 Color information from Visual Imagery

Color information is very useful in identifying between trees/vegetation and building surfaces. Color information in images can obtained by converting them from RGB space to HSV (Hue, Saturation, Value) space. Figure 3.27 illustrates the HSV space. The Value part which represents the brightness of any pixel is not used because the object identification process should not be affected by illumination changes. For each surface, mean values of Hue and Saturation are determined and used as features in the BN. The grass and tree regions have a green hue while the building and asphalt regions do not. Tree regions have more saturation than the grass regions even with the same hue. Thus, these two features can be used to classify trees, grass and building/asphalt regions.

![Figure 3.27: The conical representation of HSV color space](image)

3.3.3.6 Entropy

Entropy is a measure of information that is provided on an average, by a source. Two kinds of entropy are used as features in our BN. The first kind is the entropy that is determined from visual imagery. The entropy that is calculated at any pixel from the grayscale version of nadir mosaic measures the illumination homogeneity of a region around that pixel. If there is a rapid change of gray values, as would be in the case of tree regions and edges, the entropy measure would be high. On the other hand, the entropy measure for building surfaces will be low. Entropy can be determined as shown in Equation 3.38.
where \( X \) is a random variable representing the gray values in the image. The more homogeneous a surface, the less the value of entropy. The entropy map for the nadir mosaic measured over \( 9 \times 9 \) window is shown in Figure 3.28. From this image, one can observe that the entropy is very low over building surfaces while it is very high over tree regions.

\[
H(x) = - \sum_{x \in X} p(x) \log_2 p(x)
\]  

(3.38)

![Figure 3.28: Entropy map of nadir mosaic (9 × 9 window)](image)

The second kind of entropy is the one that is determined from Lidar elevation information. In this case, entropy over a small window represents the variation of height in that region. Entropy of Lidar elevation is also determined using Equation 3.38 and in this case, \( X \) is a random variable representing the height information of the scene. If the entropy value is high, the region is a tree region, otherwise it is a plane surface like a building or vegetation or a parking lot. Figure 3.29 displays the entropy map of Lidar elevation data measured over \( 9 \times 9 \) window.

![Figure 3.29: Entropy map of Lidar elevation data (9 × 9 window)](image)

### 3.3.4 Bayesian Network Parameter Learning

Once feature extraction is completed for all the surfaces identified by the segmentation algorithm, the truth map shown in Figure 3.21 is used to learn the parameters of the BN. This process constitutes training the BN. The parameters are the conditional probability tables of a child node given the parent nodes as explained earlier in this section. Every node has parameters defined by \( P(X_i | Pa(X_i)) \) where the random vector \( X = X_i, i = 1, 2, ..., n \) represents the nodes or features of BN and \( Pa_i = Pa(X_i), i = 1, 2, ..., n \)
3.3. 3D OBJECT IDENTIFICATION

represent the parent nodes of $X_i$. The parent nodes have a causal effect on the child nodes.

These parameters along with a given structure $S$ (e.g. Figures 3.13 and 3.14) define the joint probability distribution for the set of nodes [43]. This is given by

$$p(x) = \prod_{i=1}^{n} p(x_i | pa_i)$$  \hspace{1cm} (3.39)

The structure and the conditional probabilities/parameters of BN together encode the joint probability distribution $p(x)$. After training of the BN, all the parameters are known. Then the probability of any node/variable/feature in the BN given new evidence can be computed. This computation of probability of a node of interest is called probabilistic inference [43].

Parameter learning in BNs can be considered to be of 4 types ([43] and [44]):

1. Known structure and fully observable data
2. Known structure and incomplete data
3. Unknown structure and fully observable data
4. Unknown structure and incomplete data

The structure of the BN in this work has been built using expert knowledge of the causal relationships between different features obtained from imagery. This constitutes the known structure case. All the evidence nodes (all nodes except region node) are discretized by quantizing the data to certain number of levels (shown in Tables 3.1 and 3.2) by using equal frequency bins. If the data for all the nodes in the BN is available, then it would constitute the known structure and fully observable data case. Assuming Gaussian approximation for the parameter vectors of the nodes, the parameters can be learned by using Maximum Likelihood (ML) estimation. In this case, learning amounts to counting the number of times each node occurred in each state in all the samples of data available and updating the probabilities/parameters of each node. For example, the parameter/conditional probability that a node $X_i$ is in state $x_i^k$ and its parents $pa_i$ are in a configuration $pa_i^j$ given the data $D$ and structure $S$, is given by

$$p \left( x_i^k | pa_i^j, D, S \right) = \frac{\# \left( X_i = x_i^k, pa_i = pa_i^j \right)}{\# \left( pa_i = pa_i^j \right)}$$  \hspace{1cm} (3.40)

A formal conditional probability method is being used, which typically requires random sampling. This is really not the case here; it is more reasonable to think of these terms as
conditional proportions that will be useful in making decisions under uncertainty.

But if all the nodes are not observable, it is the case of known structure but incomplete data. The BN in this work falls under this category, because of the hidden nodes in the structure. There is no data observed for the nodes of visual plane fit and Lidar plane fit. The popular methods used to learn the parameters of the BN with known structure but incomplete data are Monte-Carlo methods (like Gibbs sampling) and Expectation-Maximization (EM) algorithm. Monte-Carlo methods are more accurate but slower compared to EM algorithm. EM algorithm is most widely used method for Gaussian approximation of the parameter vectors in practical problems.

EM algorithm ([46], [47], [43] and [44]) consists of the following major steps:

1. The algorithm is started with a random set of initial parameters/conditional probabilities.
2. Expectation step - Given the current set of parameters $p_s$, the expected counts for all the states of all the nodes are calculated as follows

$$E(N_{ijk}) = \sum_{l=1}^{N} p \left( x^k_l | pa_i^l, p_s, S \right)$$

where $l = 1, \ldots, N$ represents summation over all the data samples available in the training data set and $N_{ijk}$ represents the count for a node $X_i$ in state $x^k_l$ and its parents $pa_i$ are in a configuration $pa^l_i$.
3. Maximization step - Given the expected counts, ML estimation is performed to update the parameters as follows

$$p \left( x^k_l | pa^l_i, D, S \right) = \frac{E(N_{ijk})}{\sum_{k=1}^{r_i} E(N_{ijk})}$$

where $r_i$ is the number of states of the node $X_i$.
4. Steps 2 and 3 are repeated until the difference between the parameters in successive iterations is less than a threshold or until a maximum number of iterations is reached.

3.3.5 Bayesian Network Inference

In order to determine the expected counts in step 2 of EM algorithm, the states of the hidden nodes are estimated based on current parameters or conditional probabilities. This is also called as probabilistic inference. Many BN inference techniques exist out of which
Join tree inference/Junction tree inference [44] is used in this work. Probabilistic inference is the process of determining posterior probabilities of a desired node given evidence about the other nodes. The posterior probability of a region \( R \) belonging to a particular class \( p \) with the evidence about all the other nodes \( ev \) provided, is given by

\[
P(R_p|ev) = \sum \sum \cdots \sum p(x|ev)
\]  

where the summation is performed over all the nodes except the region node. This is marginalization of all the irrelevant nodes in the BN. But this step is a very time-consuming process if it is performed directly on the joint probability distribution table. Hence a more efficient and yet exact inference algorithm called as Join/Junction tree inference is used. Join/Junction tree inference algorithm is a type of clustering algorithm. In this method, the individual nodes in a multiply-connected network are clustered together to form a polytree network [44]. After this step, the posterior probabilities of all the nodes for which no evidence is provided in the query sample are determined by carefully considering the clustered nodes. A detailed explanation with examples for clustering algorithms can be found in [44].

### 3.3.6 Decision Theory

The theory used to make appropriate decisions based on the inference from the BN is called decision theory. Decision theory is defined as ‘probability theory + utility theory’ [50], [42], [44]. The BN structure along with the learned parameters can be used in probabilistic inference. This gives us the posterior probabilities of any node we require. This is known as probability theory. Utility theory determines how these posterior probabilities are used to make decisions. Here the decision to be made is the choice of class to which a particular region belongs to. For this purpose, utility theory similar found in [50] is used. According to [50], the decision with the maximum expected utility is accepted. Expected utility of a decision that a region \( R \) belongs to class \( q \) is given by

\[
EU(DR_q|ev) = \sum_{p=1}^{r_q} U(DR_q|R_p) P(R_p|ev)
\]  

where \( U(DR_q|R_p) \) is the utility of the decision that a region \( R \) is identified as class \( q \) while it actually belongs to class \( p \) and \( P(R_p|ev) \) is the probability that the region \( R \) belongs to class \( p \) with all the evidence about the features provided to the BN. The decision with Maximum Expected Utility (MEU) is chosen as the final decision. This is shown in
### 3.3. 3D OBJECT IDENTIFICATION

<table>
<thead>
<tr>
<th>Decision ($q$)</th>
<th>Class ($p$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Building</td>
</tr>
<tr>
<td>Building</td>
<td>$u_{11}$</td>
</tr>
<tr>
<td>Grass</td>
<td>$u_{21}$</td>
</tr>
<tr>
<td>Trees</td>
<td>$u_{31}$</td>
</tr>
<tr>
<td>Asphalt</td>
<td>$u_{41}$</td>
</tr>
<tr>
<td>Misc.</td>
<td>$u_{51}$</td>
</tr>
</tbody>
</table>

Table 3.4: Utilities assigned by an expert to make decisions on classification

Equation 3.45

$$\alpha = \arg \max_q (EU (DR_q|ev))$$  \hspace{1cm} (3.45)

The utilities are based on expert preferences and are of the form shown in Table 3.4. In this table, $U (DR_q|R_p) = u_{qp}$. In this work, the utilities are chosen as shown in Equation 3.46. This is equivalent to choosing the class with maximum a posteriori probability given all the evidence about other features as shown in Equation 3.47.

$$u_{qp} = \begin{cases} 
1 & \text{if } q = p \\
0 & \text{otherwise} 
\end{cases}$$ \hspace{1cm} (3.46)

$$\alpha = \arg \max_p P (R_p|ev)$$ \hspace{1cm} (3.47)

### 3.3.7 Choice of Segmentation Input Parameters

The set of segmentation input parameters can be chosen from a set of combinations of the same by analyzing the quality of the object identification results or classification results. This process amounts to selection of input parameters of vision algorithms based on probabilistic inference and decision making. The first step in this direction would be to define a quality metric for classification results.

The most popular quality metric that is used in data classification is the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR). Before going into TPR and FPR, one needs to understand the meaning of true positives and false positives. True positives (TP) are equivalent to hits in target detection. If a pixel whose true class is $i$ and is identified by a classifier as belonging to class $i$, it is called a hit and this is a true positive. False positives are also called as false alarms in target detection. If a pixel whose true class is not $i$ and is identified by the classifier as belonging to a class $i$, it is a false alarm and this is a false positive. True positives and False positives of a class $i$ can
be defined by Equations 3.48 and 3.49 respectively.

\[ tp_j = \begin{cases} 
1 & \text{if} \ Pixel_j \in i \text{ and } D P_{ij} \\
0 & \text{otherwise}
\end{cases} \quad (3.48) \]

\[ fp_j = \begin{cases} 
1 & \text{if} \ Pixel_j \notin i \text{ and } D P_{ij} \\
0 & \text{otherwise}
\end{cases} \quad (3.49) \]

Here \( Pixel_j \) represents a pixel at position \( j \) in the image and \( D P_{ij} \) represents the decision made by the BN that the pixel at position \( j \) belongs to class \( i \). TPR represents the rate of the true positives and FPR represents the rate of the false positives. They are defined by Equations 3.50 and 3.51 respectively.

\[ TP_i = \frac{1}{P_i} \sum_{j \in TM_i} tp_j \quad (3.50) \]

\[ FP_i = \frac{1}{N_i} \sum_{j \notin TM_i} fp_j \quad (3.51) \]

Here \( P_i \) is the total number of class \( i \) pixels in the truth map (or training data) and \( N_i \) is the total number of pixels that do not belong to class \( i \) in the truth map. \( TM_i \) represents all the locations of class \( i \) pixels.

A good classifier is not defined by just the TPR or the FPR. Even if a classifier has a high true positive rate or hit rate, it might suffer from a high false positive rate or false alarm rate. The effectiveness of a classifier is always judged by the trade-off between TPR and FPR. For good performance, a classifier should follow two important rules viz.:

1. \( TPR > FPR \)
2. \( TPR = 1-FPR \)

Ideally a perfect classifier has \( TPR = 1 \) and \( FPR = 0 \). This implies that it has 100% true positives and 0% false positives/alarms. If \( TPR = FPR = 0.5 \) (50%), it is as good as tossing a coin and selecting the class of a region. It is crucial that the two rules listed above be satisfied by the BN because all the surfaces identified as building regions in this step will be grouped together and accurate 3D buildings will be modeled from them in the next step.

This quality metric is defined at pixel level and not at region level which is the main focus of this work. The features that are used in the BN are extracted from regions and not single pixels. In spite of this, the quality metric defined above can be used to determine
the quality of region classification and also set the input parameters of segmentation algorithm. One should observe that the segmentation algorithm does not segment the surfaces ideally according to the true surfaces in the material/truth map. A surface which contains pixels belonging to two different classes is assigned to the class to which majority of its pixels belong to and a pseudo truth map is generated for training the BN. Mathematically, this can be explained as,

$$\text{Class}_R_j = \arg \max_i \left( N_j \in \text{Class}_i \right)$$

(3.52)

Thus the generated pseudo truth map depends on the segmentation input parameters. The statistics of this region will be used to train the BN by ascribing the region to the class which satisfies Equation 3.52 even though it may be made up of pixels belonging to different classes. Let us assume that this region is classified according to the pseudo truth map during the testing phase. The quality metric is determined based on the classification map and the original truth map and not the pseudo truth map. Hence, a flawed pseudo truth map will lead to poor performance of the classifier and hence a bad quality classification map. It is possible for two different classes in the training data to have similar features. As a result their distributions may not be well separated leading to misclassification by the BN. A quality metric defined based on probabilistic inference of the BN, such as the one described above penalizes such incorrect segmentation. On the other hand, good segmentation (more accurate input parameters) leads to a better pseudo truth map, better classification and hence higher quality metric. Thus good segmentation is rewarded by the probabilistic inference.

During the training phase, different color bandwidths ($h_c$) are used to segment the nadir mosaic. Features are extracted from the different pseudo truth maps generated from the segmentation maps. The BN is trained separately using the statistics of the features for each color bandwidth ($h_c$). The segmentation setup is rewarded by the probabilistic inference and decision theory by using the quality metric ($Q_M$) shown in Equation 3.53 such that the color bandwidth $h_{c_i}$ that generates a segment map which is the closest to the truth map is selected. As this metric increases, it implies that the classifier has better performance since same classifier (same structure of BN) is being used in all the cases. But one has to make sure that the TPR and FPR for each class follow the two performance criteria mentioned above or atleast close to them.

$$h_{c_i} = \arg \max_{h_{c_k}} \sum_{i=1}^{N} W_i \left( TP^k_i - FP^k_i \right)$$

(3.53)
Here $k$ is a variable representing different color bandwidths, $i = 1, 2, \cdots, N$ represents the various classes. The weights $W_i$ can be varied depending on the importance given to different classes by the user. For example, if the final purpose of the BN is to identify the building regions and reconstruct accurate 3D models of the buildings, the user might not give importance to how well the other regions are classified. In such a case, $W_{\text{building}} = 1$ while the other weights can be set to 0. The range of $(TP - FP)$ for each class is $[-1, +1]$. The value $+1$ represents the case of $TPR = 1$ and $FPR = 0$, which is a perfect classifier as explained earlier. If all the $N$ classes are equally weighted ($= 1$), ideally the metric shown in Equation 3.53 must be $N$ in which case, the segment map is exactly equal to the truth map and the classes are perfectly separated in the feature space and perfectly classified by the BN.

Even if some errors are introduced by the BN due to inseparability of the classes, they will be same over all the BNs trained using different segmentation maps, since the same feature operators are being used in all the cases. Irrespective of these errors, maximizing the metric illustrated in Equation 3.53 will lead to the best segmentation input parameters.

### 3.4 3D Object Modeling

Once the building surfaces are identified by the BN, all the surfaces belonging to a single building can be grouped together by using morphological image processing techniques like connected components. The geometry of these planar surfaces was already determined and used as features in classification during the object identification process. This geometry includes the plane equation derived from the 3D corners of the surfaces. The 3D corners are obtained by orthorectifying the 2D corners detected in the nadir mosaic using height information from stereo geometry.

The height information obtained from stereo mosaics is not accurate because of the resolution of the depth deviation from the fixation plane. The depth deviation from the fixation plane can be derived from Equation 3.23. It is given by

$$
\Delta Z = Z - H = H \frac{\Delta y}{d_y} + \bar{T}_z
$$

(3.54)

where $\Delta y$ is the mosaic displacement between the right stereo mosaic and left stereo mosaic, $H$ is the average flying height of the airborne sensor over the fixation plane, $d_y$ is the distance between the fixed lines in left and right mosaics and $\bar{T}_z$ is the average camera height deviation. If we assume that the camera has only 2D translational motion and
hence $T_z = 0$, the depth deviation from the fixation plane is given by

$$\Delta Z = Z - H = H \frac{\Delta y}{d_y}$$  \hspace{1cm} (3.55)

For every pixel mosaic displacement exhibited by the stereo mosaics, the depth deviation from the fixation plane would be $H/d_y$. So the height of each corner point of a surface is always quantized into levels defined by $n(H/d_y), n = 0, 1, 2, \cdots$.

Thus we can see that the height information is inaccurate for each of the 2D corners. The 3D corners obtained after orthorectification of the 2D corners using this inaccurate height information are not precisely referenced/placed in the object space. This in turn leads to inaccurate plane fit and an erroneous 3D model of the buildings in a scene. In spite of these inaccuracies in the geometry of a scene, BNs perform well during region identification as explained in the previous section. Thus, there is a need for improvement in the accuracy of the 3D geometry of the buildings. The flowchart shown in Figure 3.30 illustrates the method of accurately modeling a 3D building.

In the first step, all the corners of all the surfaces of the building are projected on to the original individual video frames using the camera position and orientation parameters of each of those frames. This projection of a 3D corner point given by $P_{\text{world}} = (X, Y, Z)$ is shown in Equation 3.56. This is the same formulation derived in Equations 3.1 - 3.9 (Section 3.1).

$$P_{\text{im}} = KR(P_{\text{world}} - T_i)$$  \hspace{1cm} (3.56)

Here $K$ is the interior camera parameter matrix defined by Equation 3.6, $R$ is the orientation matrix of the camera and $T$ is the 3D viewpoint of the camera defined by $T = (X_{ci}, Y_{ci}, Z_{ci})$ at frame $i = 1, 2, \cdots, N$. In order to optimize the 3D points $P_{\text{world}}$, this projection is formulated in homogeneous coordinates [11]. This linearizes the Equation 3.56 as shown in Equation 3.57. The image coordinates of the projected corner points in Euclidean space are obtained by $(\hat{x}, \hat{y}) = (k\hat{x}/k, k\hat{y}/k)$.

$$k \begin{bmatrix} P_{\text{im}} \\ 1 \end{bmatrix} = \begin{bmatrix} k\hat{x} \\ k\hat{y} \\ k \end{bmatrix} = KR \left( I_{3\times3} \right) \left( P_{\text{world}} \\ 1 \right)$$  \hspace{1cm} (3.57)

For each building, the surfaces are represented by index $s = 1, 2, \cdots, S$, where $S$ is the total number of surfaces in the building identified during the classification process. Each surface has corner points represented by index $j = 1, 2, \cdots, N_s$, where $N_s$ is the total number of corner points in a surface $s$. All the frames in which the building is visible are
determined by selecting all the frames in which all the corners of the building are within the field of view of the video frames. This can be mathematically represented as

\[ i = fr, \text{iif } \forall i, 1 \leq \hat{x}_i \leq N, 1 \leq \hat{y}_i \leq M \]  

(3.58)

where \( fr \) is the number of the frame under consideration, \((N, M)\) is the size of each of the individual frames. The next step is determining the corresponding points for the corners of the surfaces in the individual frames. For this all the frames are segmented using the
mean shift segmentation algorithm with same input parameters that were used during object identification. The surface in the individual frame which has maximum overlap with the projected 3D surface (by projecting 3D corners) is considered as a match. The 2D corners of the surface in the video frame are determined using the same method described in Section 3.3.3.3. Then the distances between each of the projected 3D corners and all the 2D corners of the surface are measured. The pairs of the points with minimum distances are considered as matches. This process is continued for all the surfaces and all the video frames in which the building is visible.

After obtaining the corresponding point pairs for each surface, the 3D position of each corner in the object space is optimized in such a way that the sum of squares of distances between its projected 2D position and the corresponding actual 2D position in the video frames. This is an unconstrained non-linear optimization problem. We used the Levenberg-Marquardt (LM) algorithm [31] to solve the optimization problem. The algorithm for optimizing the corner points of a surface is as follows:

1. For each corner point, project the present estimate of the 3D position \((X, Y, Z)\) on to each video frame \(i\) given by Equation 3.58 and obtain the projected 2D position \((\hat{x}_j^i, \hat{y}_j^i)\) of the corner point in each of those frames.

\[
\begin{pmatrix}
 k\hat{x}_j^i \\
k\hat{y}_j^i \\
k
\end{pmatrix}
= KR_i \begin{pmatrix}
 I_{3x3} & -T_i \\
 X_j \\
 Y_j \\
 Z_j \\
1
\end{pmatrix}, i = 1, 2, \ldots , N \tag{3.59}
\]

2. Find the distance between the projected corner point and its corresponding actual corner \((x^i_j, y^i_j)\) in each frame.

\[
f^j_i = \sqrt{(x^i_j - \hat{x}_j^i)^2 + (y^i_j - \hat{y}_j^i)^2}, i = 1, 2, \ldots , N \tag{3.60}
\]

3. Find the sum of squared distances calculated in Step 2.

\[
g_j = \sum_{i=1}^{N} \left(f^j_i\right)^2 \tag{3.61}
\]

4. Minimize the error function or sum of squared distances \(g_j\) (in squared pixels) with respect to the 3D corner point \((X_j, Y_j, Z_j)\) using Levenberg-Marquardt [31] optimization algorithm to obtain accurate 3D corner point \((X^{opt}_j, Y^{opt}_j, Z^{opt}_j)\).
5. Repeat steps 1 - 4 until all the corners of each surface are optimized.

Unfortunately all the surfaces in the nadir mosaic from which 3D corners are determined initially and all the surfaces in the video frames are not segmented exactly in the same way. So some of the projected corners do not have a matching 2D corner in the video frame. In such cases, the error function shown in Equation 3.61 does not get minimized. This is the reason why each corner is optimized separately. Finally, in order to make the plane fit more robust, once all the corners of the surface are optimized, a least squares RANSAC plane fit [11] is used to determine the plane equation through the optimized corners. The form of the plane equation is shown in Equation 3.62. By doing this, the outliers (incorrectly matched corners) are removed. Using this robust plane equation, the new \( Z \) coordinate of the 3D corner \( j \) i.e., \( Z_{j}^{\text{fit}} \) is estimated using Equation 3.63, thus obtaining the accurate 3D corners \((X_{j}^{\text{opt}}, Y_{j}^{\text{opt}}, Z_{j}^{\text{fit}})\) of the surface.

\[
B1(X_{j}^{\text{opt}}) + B2(Y_{j}^{\text{opt}}) + B3(Z_{j}^{\text{opt}}) + B4 = 0 \tag{3.62}
\]

\[
Z_{j}^{\text{fit}} = -\left(\frac{B4 + B1X_{j}^{\text{opt}} + B2Y_{j}^{\text{opt}}}{B3}\right) \tag{3.63}
\]

Since the optimization for the 3D corners is performed based on the distance between the projected version and the actual 2D corner at pixel level, the corners on the common edge between two surfaces (like a T corner) do not coincide with each other in the object space after optimization. Modeling the complete building from the accurate 3D planes, their edges and corners requires some final post processing steps.

Consider the common edge between two surfaces of a roof as shown in Figure 3.31. One can see that this building has two T corners. After optimization of all the 4 corners on each surface, the roof structure might turn out to look like the middle image in Figure 3.31 in the object space. This is due to the fact that the T corner is optimized as part of each surface separately. Each T corner is divided into two different corners and there is a gap in the roof structure which is not actually true. So a least squares line is fitted through the 4 points as shown in Figure 3.31 (right most image) but using only the \( X \) and \( Y \) coordinates and not the elevation (\( Z \) coordinate). Then all the 4 points are projected on to this fitted line while keeping their original elevation information.

The distance between the closest point pairs is calculated in 2D space (\( X \) and \( Y \) coordinates after projection). If it is less than a threshold preset by the user, then the two corners are merged by considering the average of the two points in the pair as the accurate \( X \) and \( Y \) coordinates of the T corner all the while preserving the original elevation information.
The distances between the $Z$ coordinates of the points in a pair are tested to see if they are below a certain threshold. If they are, the $Z$ coordinates of the two points in a pair are also averaged to obtain the accurate 3D position of the T corner. Now, as one can see, two points of each surface are fixed. The plane is modeled in such a way that the T corners are fixed but it is a least square fit between the other two corners of each surface. Hence, the elevation of the other two corners of each surface will change according to the new plane equation as shown in Equation 3.63. The final rooftop is illustrated in Figure 3.32. If the distance between the $Z$ coordinates is not below the preset threshold, then the corners are not T corners. They are considered to be corners of surfaces at different elevations and hence a vertical facet is dropped from the higher elevation surface to the lower elevation surface as shown in Figures 3.33 and 3.34. Also, since the roof top of a building cannot be floating in free space, vertical facets are dropped from each edge to the level of an adjacent surface which is lower than the surface in consideration as shown in Figures 3.32 and 3.34 to obtain the final models of the buildings.

If the distance between the points in 2D space is not below the preset threshold, the corners are again two different corners. The test on the $Z$ coordinate similar to the one described above will reveal if the corners are on the same level as shown in Figure 3.35 or at two different levels as shown in Figure 3.36.

The post-processing technique described here is applicable to any polyhedral building with any number of surfaces on the roof. Thus a complete 3D model of a building can be reconstructed from a series of aerial video frames. The results presented in the next
3.4. 3D OBJECT MODELING

Figure 3.33: Modeling of corners at different elevations during post processing step

Figure 3.34: Final model of a building with corners at different elevations

Figure 3.35: Final model of a building with different corners at same elevation

Figure 3.36: Final model of a building with different corners at different elevations

Chapter describe the outputs expected at each level of the system including stereo mosaics, region identification and finally 3D object modeling. Optimization results obtained during object identification and modeling will also be detailed in the next chapter.
Chapter 4

Results and Discussion

In this chapter, the results obtained by using the approaches developed during this research work are presented. Section 4.1 provides the output of accomplishing the first research goal i.e., improvements made in the process of building stereo mosaics from aerial video imagery to remove the artifacts generated by fast PRISM algorithm and thus enabling accurate 3D modeling of objects in the scene. Section 4.2 provides the results for 3D object identification process using BN along with the automated decision process for setting best input parameters to be used by segmentation algorithm to identify the surfaces of various objects. In the same section, the analysis of improvement in the accuracy of object identification with the fusion of Lidar data and visual imagery is performed. Section 4.3 illustrates the results of automated iterative optimization process to improve the accuracy of the 3D models of buildings identified by the BN. Thus, Sections 4.2 and 4.3 provide the final results of accomplishing the second and third research goals.

4.1 Stereo Mosaics

To give the readers a good perception of the improvement that can be obtained using the modified PRISM as compared to the fast PRISM, we present the results on four sets of data. The complexity of the data increases from Set 1 to Set 4.

4.1.1 Image Set 1

Data set 1 consists of just two frames artificially created in Microsoft Paint. The results for this set are illustrated in Figures 4.1 - 4.6. The two frames can be seen in Figure 4.1. The blue rectangle exhibits motion parallax while the other objects do not. First, the source triangles are generated using regular triangulation of fast PRISM method. The fixed lines,
4.1. STEREO MOSAICS

Figure 4.1: Two frames created artificially with motion parallax between objects

Figure 4.2: Source triangles from fast PRISM in two consecutive frames

The matching curve along with the mesh in both the frames are demonstrated in Figure 4.2. The source triangles created using segment-based mesh design of modified PRISM method along with the fixed lines and the new matching curve are displayed in Figure 4.3. The process of stitching the frames into mosaic is clearly illustrated for both the methods by plotting the destination triangles in Figures 4.4 and 4.5. The readers should observe the unwanted warping of some of the source triangles in to destination triangles in Figures 4.4(a) and 4.4(b). The left mosaic built using fast PRISM can be observed in Figure 4.6(a) while the left mosaic built using modified PRISM can be observed in Figure 4.6(b). We can see the visual artifacts generated by fast PRISM and regular triangulation in Figure 4.6(a) but they are eliminated by our modified PRISM. But if there is no motion parallax, both methods generate mosaics of same quality.

4.1.2 Image Set 2

Data set 2 consists of a series 100 frames captured indoors using a Nikon D70 camera with 50 mm lens. The test objects were taped to the wall and the translation in the camera was generated on an optical bench. The translation of the camera is about 1 cm in y direction and ±1 mm in x direction from frame to frame. Six of the frames in this sequence are
shown in Figure 4.7 The improvement made by the modified PRISM can be seen in Figure 4.8. Figures 4.9(a) and 4.9(b) present the stereo mosaics stitched using the fast PRISM algorithm and modified PRISM algorithm respectively.

As explained in the appendix, visual artifacts in the stereo mosaics lead to errors in the reconstructed 3D CAD (Computer Aided Design) model of an object. For instance, in
4.1. STEREO MOSAICS

Figure 4.6: Mosaics illustrating the improvement between the two methods

(a) Mosaic built using fast PRISM
(b) Mosaic built using modified PRISM

Figure 4.7: Six frames from an indoor video sequence

In this example, the left edge of the rectangular box in the left mosaic has been deformed by fast PRISM. The camera was placed at an average distance of $H = 70$ cm from the scene and the distance between the two fixed lines is chosen to be $d_y = 70$ pixels. So, one pixel error in the left mosaic would change the height of that particular point or decrease the depth of the point by $\delta Z = H / d_y = 1$ cm. The relative error in the depth of the object ($\delta Z / Z$) is about $1.4\%$. These errors in heights of few points on a facet can be overcome by using RANSAC (RANdom SAmple Consensus) plane fit algorithm [22]. But extra corners and edges are created in the model. A single facet in real world is transformed into two facets in the reconstructed model as can be seen in Figure 4.10. This would not only affect the geometrical properties of the object but also make the process of spectral assignments to the facets very difficult in the sense that it would be extremely difficult to register these erroneous models to the hyperspectral imagery from which spectral information is obtained. But with modified PRISM, such errors are avoided and we are one step closer to the goal of automatic reconstruction of a spectrally accurate 3D model from 2D images.
4.1. STEREO MOSAICS

4.1.3 Image Set 3

Data set 3 consists of a video captured by an airborne sensor developed indigenously by the Laboratory for Imaging Algorithms and Systems (LIAS) group at Rochester Institute
Figure 4.11: Six frames from an airborne video sequence

of Technology (RIT). The sensor is called Wildfire Airborne Sensing Program (WASP) Lite. The video was captured over RIT at an average height of 3000 ft and a frequency of 3 frames per second. Though it has the capability to produce color video, we used only one of the channels in our experiments on this data set. Figure 4.11 displays some of the frames from the airborne video sequence. In Figure 4.12 we can observe the improvement in the right mosaic. The building circled has been disfigured from its actual shape by the fast algorithm. The distance between the two fixed lines is $d_y = 150$ pixels. The relative error in the depth of the point on the roof of the building would be $\delta Z / Z = 0.66\%$ and the error in the depth (and height) of the point would be 20 ft. The 3D geometry of the building is deformed with extra facets on one of the vertical walls of the building as seen in Figure 4.13.

4.1.4 Image Set 4

Data set 4 consists of a video sequence captured by WASP-LT over RIT. Three narrow band filters were used to capture Red, Green and Blue information of the scene. This set was collected by the sensor at an average flying height of 253m from the scene. As the flying height is not very large for this data set, the third dimension of the sensor translation cannot be ignored with this data sequence. 2D piecewise linear interpolation is used to fill up orphan pixels as explained in Section 3.2.5.

Some of the original images from the data sequence and the orientation corrected im-
ages are shown in Figure 4.14. The right mosaic built using fast PRISM on these images is shown in Figure 4.15(a). It should be observed that a part of the building in the video sequence is missing in the mosaic. The same right mosaic built using our modified PRISM is shown in Figure 4.15(b). The distance between the fixed lines is \( d_y = 70 \) pixels. Even though the shape of the building is unaltered by the missing part in Figure 4.15(a), the height information extracted using this mosaic would have error of about \( \Delta Z = H \left( \Delta y / d_y \right) = 253 \left( \frac{3}{70} \right) \approx 10 \) m for 3 pixels missing from the right mosaic. Our method corrects that artifact and provides both accurate shape and height information required for 3D scene reconstruction process.
4.2 3D Object Identification

In this section, the results pertaining to the 3D object identification part of the algorithm are presented. BN is used for region classification. Bayes Net Toolbox for MATLAB®[51] is used for creating, training and testing the Bayesian Networks in this work. The results are organized as follows. First, the region classification results using only visual information are presented. Then, the region classification results using both visual and Lidar information are discussed. Finally, a qualitative (visual) and quantitative comparison is made between the performance of the classifier using visual information only and the classifier using both visual and Lidar information.

As explained in the previous chapter, the spatial bandwidth $h_s$ and the minimum spatial size of each cluster for the mean shift segmentation algorithm are set at 15 and 20 respectively. In order to pick the best color bandwidth that gives the best classification results, the color bandwidth is varied from $h_c = 2$ to $h_c = 20$ in steps of 2. This results in 10 different sets of input parameters for the segmentation algorithm. The best color bandwidth to be used for the object identification process is picked according to Equation 3.53. First, the segmentation maps, pseudo truth maps and the classification maps are shown for a particular range of the color bandwidth values to point out the variation in the quality of the region identification.
The nadir mosaic of the scene used in this work is shown here in Figure 4.16. The segmentation maps generated from this nadir mosaic using the color bandwidths of $h_c = 2, 10, 20$ can be seen in Figure 4.17. These images reveal that as $h_c$ is increased, neighboring surfaces with little variation in color will be merged together as one single surface in accordance with the mean shift segmentation algorithm. This trait is clearly visible over the tree regions. The material map and original truth map for training the BN are shown in Figures 4.18 and 4.19 respectively. The pseudo truth maps generated using Equation 3.52 on the segmentation maps and original truth map for each of the color bandwidths 2, 10 and 20 are shown in Figure 4.20.

Figure 4.21 illustrates the difference in pseudo truth maps generated using Equation 3.52. Note that there is very little difference between the pseudo truth maps generated for $h_c = 2$ and $h_c = 10$ while the pseudo map generated for $h_c = 20$ is significantly different. This results in considerable disparity in the final quality of the classification map which is the output of the BN for the segmentation parameters under consideration. Apart from
these obvious differences, the surfaces in the segmentation map of \( h_c = 2 \) case are quite different from the surfaces in the segmentation map of \( h_c = 10 \) and \( h_c = 20 \) as can be seen in Figure 4.17. This will lead to different statistics of features extracted which in turn will result in different classification results.

4.2.1 Using Visual Information only

The BN structure shown in Figure 3.14 with nodes defined in Table 3.2 is used to identify the different classes of a region i.e., buildings (1), grass (2), trees (3), asphalt (4) and miscellaneous (5). This structure is formulated using information only from visual
4.2. 3D OBJECT IDENTIFICATION

Figure 4.18: Material map of the scene generated by DIRSIG

Figure 4.19: Original truth map generated from the material map of the scene

(a) $h_c = 2$

(b) $h_c = 10$

(c) $h_c = 20$

Figure 4.20: Pseudo truth maps generated from segmentation maps for training the BN

data i.e., stereo mosaics and nadir mosaic. The classification maps generated by the BN
for the segmentation parameters of \( h_c = 2, 10, 20 \) are illustrated in Figure 4.22. In the classification map of \( h_c = 2 \), many parts of the tree regions are identified as building regions. This can be attributed to the fact that each tree has been segmented into many homogeneous surfaces and each small surface has properties similar to that of building regions like higher elevation and very low entropy values. In the case of \( h_c = 10 \), many grass regions are identified as asphalt. This is due to the fact that the height information and corner information are very similar in the two classes. In the case of \( h_c = 20 \) many grass/asphalt regions are identified as building regions because of the training received by the BN from the pseudo truth map shown in Figure 4.21 (c). Many tree regions are also identified as grass in this case. This is because, whole trees are segmented as one surface here and the elevation information obtained from the boundary of the segment shows that the trees have zero elevation above the surrounding grass regions. All these visual results are reflected in the values of the quality metrics calculated using Equations 3.48 - 3.51. The quality metrics calculated for each of the classes over all the test cases of varying segmentation parameters are tabulated in Tables 4.1 - 4.4.

The metrics for the miscellaneous class are not shown here. This is because, in some
segmentation cases, the pseudo truth maps generated using Equation 3.52 do not have any segments containing pixels of the miscellaneous class and hence there is no training data for this class. The TPR and FPR for such cases do not reflect the true quality metric. Also, the miscellaneous class is inconsequential for later stages of this work.

The TPR and FPR for the building and tree classes follow the two performance criteria discussed in Section 3.3.7. The grass and asphalt class do not follow the second criterion. But this is the best the classifier can perform with the existing training data. Another important observation is that the quality metrics are not convex in nature with variation in segmentation parameters. This is because of the fact that, for each case of segmentation, the features of any one of the classes are being ill-represented by the segments. For example, in the case of $h_c = 20$, trees are not well represented because the height extracted from the boundary of the segments will be the same height as the terrain (grass class).

In order to choose the correct set of segmentation parameters, the user has to define the weights in Equation 3.53 according to the significance assigned to each class. This again depends on the final purpose for which the classification map will be used. In this work, the final purpose is 3D modeling of buildings. Hence, in the first case, $W_{building}$ is
4.2. 3D OBJECT IDENTIFICATION

Table 4.2: True positive and false positive rates for grass class using visual information only

<table>
<thead>
<tr>
<th>$h_c$</th>
<th>$TP$</th>
<th>$FP$</th>
<th>$TP - FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.7723</td>
<td>0.0575</td>
<td>0.7148</td>
</tr>
<tr>
<td>4</td>
<td>0.7399</td>
<td>0.0706</td>
<td>0.6693</td>
</tr>
<tr>
<td>6</td>
<td>0.5871</td>
<td>0.0523</td>
<td>0.5348</td>
</tr>
<tr>
<td>8</td>
<td>0.5583</td>
<td>0.0946</td>
<td>0.4638</td>
</tr>
<tr>
<td>10</td>
<td>0.4741</td>
<td>0.0609</td>
<td>0.4132</td>
</tr>
<tr>
<td>12</td>
<td>0.5178</td>
<td>0.0874</td>
<td>0.4304</td>
</tr>
<tr>
<td>14</td>
<td>0.5032</td>
<td>0.0926</td>
<td>0.4107</td>
</tr>
<tr>
<td>16</td>
<td>0.5156</td>
<td>0.1280</td>
<td>0.3876</td>
</tr>
<tr>
<td>18</td>
<td>0.5583</td>
<td>0.1046</td>
<td>0.4537</td>
</tr>
<tr>
<td>20</td>
<td>0.6821</td>
<td>0.1301</td>
<td>0.5520</td>
</tr>
</tbody>
</table>

Table 4.3: True positive and false positive rates for trees class using visual information only

<table>
<thead>
<tr>
<th>$h_c$</th>
<th>$TP$</th>
<th>$FP$</th>
<th>$TP - FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9033</td>
<td>0.0860</td>
<td>0.8173</td>
</tr>
<tr>
<td>4</td>
<td>0.8711</td>
<td>0.0764</td>
<td>0.7947</td>
</tr>
<tr>
<td>6</td>
<td>0.9099</td>
<td>0.1034</td>
<td>0.8066</td>
</tr>
<tr>
<td>8</td>
<td>0.8951</td>
<td>0.0806</td>
<td>0.8145</td>
</tr>
<tr>
<td>10</td>
<td>0.9111</td>
<td>0.0968</td>
<td>0.8143</td>
</tr>
<tr>
<td>12</td>
<td>0.8841</td>
<td>0.0810</td>
<td>0.8031</td>
</tr>
<tr>
<td>14</td>
<td>0.8415</td>
<td>0.0658</td>
<td>0.7757</td>
</tr>
<tr>
<td>16</td>
<td>0.7879</td>
<td>0.0646</td>
<td>0.7233</td>
</tr>
<tr>
<td>18</td>
<td>0.8267</td>
<td>0.0596</td>
<td>0.7672</td>
</tr>
<tr>
<td>20</td>
<td>0.7648</td>
<td>0.0523</td>
<td>0.7125</td>
</tr>
</tbody>
</table>

assigned a value of 1 while the other weights are set to 0. For such a case, the plot of the quality metric $QM$ vs. the color bandwidth $h_c$ is illustrated in Figure 4.23. As can be seen from this figure (data point circled in red), the best segmentation input color bandwidth which has the maximum quality metric is $h_c = 10$. This classification map has already been presented in Figure 4.22 (b).

If equal importance is given to building and tree classes, then the weights are set as $W_{\text{building}} = 0.5$ and $W_{\text{trees}} = 0.5$ with the remaining weights being zero. In this case also, the best quality is given by the color bandwidth input of $h_c = 10$ as can be seen from the plot in Figure 4.24. If the identification of tree regions is assigned greater importance as compared to that of building region, the weights can be set to $W_{\text{building}} = 0.1$ and $W_{\text{trees}} = 0.9$, resulting in the best color bandwidth of $h_c = 2$ (from Figure 4.25). This classification map has been illustrated in Figure 4.22 (a).
4.2. 3D OBJECT IDENTIFICATION

<table>
<thead>
<tr>
<th>$h_c$</th>
<th>TP</th>
<th>FP</th>
<th>TP − FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.7555</td>
<td>0.0312</td>
<td>0.7243</td>
</tr>
<tr>
<td>4</td>
<td>0.6025</td>
<td>0.0599</td>
<td>0.5466</td>
</tr>
<tr>
<td>6</td>
<td>0.5015</td>
<td>0.1316</td>
<td>0.3699</td>
</tr>
<tr>
<td>8</td>
<td>0.3917</td>
<td>0.1407</td>
<td>0.2510</td>
</tr>
<tr>
<td>10</td>
<td>0.4471</td>
<td>0.1374</td>
<td>0.3097</td>
</tr>
<tr>
<td>12</td>
<td>0.4599</td>
<td>0.1618</td>
<td>0.2981</td>
</tr>
<tr>
<td>14</td>
<td>0.5329</td>
<td>0.1692</td>
<td>0.3636</td>
</tr>
<tr>
<td>16</td>
<td>0.4110</td>
<td>0.1375</td>
<td>0.2736</td>
</tr>
<tr>
<td>18</td>
<td>0.5130</td>
<td>0.1448</td>
<td>0.3682</td>
</tr>
<tr>
<td>20</td>
<td>0.5026</td>
<td>0.0918</td>
<td>0.4109</td>
</tr>
</tbody>
</table>

Table 4.4: True positive and false positive rates for asphalt class using visual information only.

![Plot of quality metric vs. color bandwidth](image)

Figure 4.23: Plot of quality metric vs. color bandwidth for the case of $W_{building} = 1$ and rest of the weights 0

4.2.2 Using Visual and Lidar Information

The BN structure shown in Figure 3.13 with nodes defined in Table 3.1 is used to identify the different classes of region i.e., buildings (1), grass (2), trees (3), asphalt (4) and miscellaneous (5) similar to the previous subsection. The difference is that this structure fuses the information from visual data i.e., stereo mosaics and nadir mosaic and information
from Lidar data in order to identify different classes of a region. The classification maps generated by the BN for the segmentation parameters of $h_c = 2, 10, 20$ are illustrated in Figure 4.26. The building regions are better identified for the cases of $h_c = 2$ and $h_c = 10$ than for the case of $h_c = 20$. This is because, usually the height information provided by Lidar data is very accurate and can be the most important feature for classification. Due to the merging of building and terrain regions for this case as shown in Figure 4.21, the classes are not well separated in training data resulting in poor classification results for $h_c = 20$. For the identification of grass and asphalt regions, the case of $h_c = 2$ is superior to $h_c = 10$. The quality metrics are presented in Tables 4.5 - 4.8.

Again for this subsection, the results of the miscellaneous class are not shown because the training data for this class is not present in all the cases of segmentation due to the usage of pseudo truth maps to train the BN. Even though TPRs for the building class are convex for different segmentation cases, the quality metrics are not because the other classes also play a role in determining the FPRs of the building class. Again, in order to choose the correct set of segmentation parameters, the user has to define the weights in Equation 3.53 according to the importance ascribed to each class. Similar to the case
### 4.2. 3D OBJECT IDENTIFICATION

<table>
<thead>
<tr>
<th>$h_c$</th>
<th>$TP$</th>
<th>$FP$</th>
<th>$TP - FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9478</td>
<td>0.0273</td>
<td>0.9204</td>
</tr>
<tr>
<td>4</td>
<td>0.9651</td>
<td>0.0499</td>
<td>0.9151</td>
</tr>
<tr>
<td>6</td>
<td>0.9576</td>
<td>0.0346</td>
<td>0.9230</td>
</tr>
<tr>
<td>8</td>
<td>0.9609</td>
<td>0.0476</td>
<td>0.9133</td>
</tr>
<tr>
<td>10</td>
<td>0.9604</td>
<td>0.0151</td>
<td>0.9453</td>
</tr>
<tr>
<td>12</td>
<td>0.9590</td>
<td>0.0197</td>
<td>0.9393</td>
</tr>
<tr>
<td>14</td>
<td>0.9424</td>
<td>0.0225</td>
<td>0.9199</td>
</tr>
<tr>
<td>16</td>
<td>0.9433</td>
<td>0.0481</td>
<td>0.8952</td>
</tr>
<tr>
<td>18</td>
<td>0.9189</td>
<td>0.0996</td>
<td>0.8194</td>
</tr>
<tr>
<td>20</td>
<td>0.9188</td>
<td>0.1236</td>
<td>0.7952</td>
</tr>
</tbody>
</table>

Table 4.5: True positive and false positive rates for building class using visual and Lidar information

<table>
<thead>
<tr>
<th>$h_c$</th>
<th>$TP$</th>
<th>$FP$</th>
<th>$TP - FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.8969</td>
<td>0.0583</td>
<td>0.8386</td>
</tr>
<tr>
<td>4</td>
<td>0.8389</td>
<td>0.0604</td>
<td>0.7785</td>
</tr>
<tr>
<td>6</td>
<td>0.6739</td>
<td>0.0424</td>
<td>0.6314</td>
</tr>
<tr>
<td>8</td>
<td>0.6633</td>
<td>0.0912</td>
<td>0.5720</td>
</tr>
<tr>
<td>10</td>
<td>0.6224</td>
<td>0.1110</td>
<td>0.5114</td>
</tr>
<tr>
<td>12</td>
<td>0.6735</td>
<td>0.1262</td>
<td>0.5473</td>
</tr>
<tr>
<td>14</td>
<td>0.6058</td>
<td>0.1542</td>
<td>0.4517</td>
</tr>
<tr>
<td>16</td>
<td>0.6954</td>
<td>0.1157</td>
<td>0.5797</td>
</tr>
<tr>
<td>18</td>
<td>0.6841</td>
<td>0.0668</td>
<td>0.6173</td>
</tr>
<tr>
<td>20</td>
<td>0.7147</td>
<td>0.0784</td>
<td>0.6362</td>
</tr>
</tbody>
</table>

Table 4.6: True positive and false positive rates for grass class using visual and Lidar information

<table>
<thead>
<tr>
<th>$h_c$</th>
<th>$TP$</th>
<th>$FP$</th>
<th>$TP - FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.9552</td>
<td>0.0291</td>
<td>0.9261</td>
</tr>
<tr>
<td>4</td>
<td>0.9516</td>
<td>0.0273</td>
<td>0.9243</td>
</tr>
<tr>
<td>6</td>
<td>0.9527</td>
<td>0.0280</td>
<td>0.9248</td>
</tr>
<tr>
<td>8</td>
<td>0.9512</td>
<td>0.0278</td>
<td>0.9234</td>
</tr>
<tr>
<td>10</td>
<td>0.9504</td>
<td>0.0277</td>
<td>0.9227</td>
</tr>
<tr>
<td>12</td>
<td>0.9483</td>
<td>0.0252</td>
<td>0.9231</td>
</tr>
<tr>
<td>14</td>
<td>0.9420</td>
<td>0.0227</td>
<td>0.9193</td>
</tr>
<tr>
<td>16</td>
<td>0.9471</td>
<td>0.0224</td>
<td>0.9248</td>
</tr>
<tr>
<td>18</td>
<td>0.9457</td>
<td>0.0226</td>
<td>0.9231</td>
</tr>
<tr>
<td>20</td>
<td>0.9419</td>
<td>0.0225</td>
<td>0.9195</td>
</tr>
</tbody>
</table>

Table 4.7: True positive and false positive rates for trees class using visual and Lidar information
of the classifier based on visual information, if the building regions are the only objects of interest then the weight $W_{\text{building}}$ is assigned a value of 1 while all other weights are set to 0. In such a case, the plot of the quality metric $QM$ vs. the color bandwidth $h_c$ is illustrated in Figure 4.27. As can be seen from this figure (data point circled in red), the best segmentation input color bandwidth which has the maximum quality metric is $h_c = 10$. This classification map has already been presented in Figure 4.26 (b).

If equal importance is attached to the building and tree classes, the weights are set as $W_{\text{building}} = 0.5$ and $W_{\text{trees}} = 0.5$ and the rest of the weights are set to 0. Again for this case, the color bandwidth of $h_c = 10$ performs the best as can be seen from the plot in Figure 4.28. If a higher importance is assigned to the identification of the tree regions as compared to that of the building regions, the weights can be set as $W_{\text{building}} = 0.1$ and $W_{\text{trees}} = 0.9$, resulting in the best color bandwidth of $h_c = 2$ (from Figure 4.29). The classification map has been illustrated in Figure 4.26 (a).
4.2.3 Comparison of the Two Classifiers

As we can see from Figures 4.22 and 4.26 or from the quality metrics presented in Tables 4.1 to 4.5, the BN which uses both visual and Lidar information performs better than that which uses only visual information in identifying the 3D objects. Since the focus is only on building regions then, quantitatively, the best $QM$ for the BN which uses both visual and Lidar information is about 0.9453 whereas the best $QM$ for the BN which uses visual information only is about 0.7700. Even in the case of other classes, it is seen that the BN based on both visual and Lidar data is noticeably superior compared to that based only on visual data. This is because, Lidar data provides very accurate height and entropy information irrespective of the solar angle, color of the buildings, occlusions due visual sensor position, etc. which are vital in differentiating between building, terrain and tree classes. Thus, incorporation of Lidar information with visual data adds significant value to the region identification process.
4.2.3D OBJECT IDENTIFICATION

<table>
<thead>
<tr>
<th>$h_c$</th>
<th>$TP$</th>
<th>$FP$</th>
<th>$TP - FP$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.7459</td>
<td>0.0279</td>
<td>0.7180</td>
</tr>
<tr>
<td>4</td>
<td>0.5917</td>
<td>0.0416</td>
<td>0.5501</td>
</tr>
<tr>
<td>6</td>
<td>0.6322</td>
<td>0.1333</td>
<td>0.4988</td>
</tr>
<tr>
<td>8</td>
<td>0.2148</td>
<td>0.1114</td>
<td>0.1034</td>
</tr>
<tr>
<td>10</td>
<td>0.4548</td>
<td>0.1562</td>
<td>0.2987</td>
</tr>
<tr>
<td>12</td>
<td>0.3382</td>
<td>0.1144</td>
<td>0.2238</td>
</tr>
<tr>
<td>14</td>
<td>0.2873</td>
<td>0.1250</td>
<td>0.1623</td>
</tr>
<tr>
<td>16</td>
<td>0.3037</td>
<td>0.0584</td>
<td>0.2453</td>
</tr>
<tr>
<td>18</td>
<td>0.4559</td>
<td>0.0983</td>
<td>0.3576</td>
</tr>
<tr>
<td>20</td>
<td>0.2970</td>
<td>0.0578</td>
<td>0.2393</td>
</tr>
</tbody>
</table>

Table 4.8: True positive and false positive rates for asphalt class using visual and Lidar information

Figure 4.27: Plot of quality metric vs. color bandwidth for the case of $W_{building} = 1$ and rest of the weights 0

4.2.4 Using Trained Bayesian Networks on Other Data Sets

In order to prove the robustness of the trained BN, it is applied on a different (test) data set for object identification. One can observe that the features like elevation information from visual imagery, elevation information from Lidar data, edge and corner information and surface area of regions are used in absolute units and hence do not depend on the
4.2. 3D OBJECT IDENTIFICATION

Figure 4.28: Plot of quality metric vs. color bandwidth for the case of $W_{\text{building}} = 0.5$, $W_{\text{trees}} = 0.5$ and rest of the weights 0

sensor intrinsic parameters and the flying height of the sensor. The color information and entropy on the other hand are not used in absolute units and hence there are two important constraints that need to be applied in order to use the BN as is on a new data set, without modifying the priors. They are:

1. A sensor with same color characteristics as the one used during training needs to be employed. This will ensure that the data discretization thresholds adopted during training for hue and saturation features can be used with the new data set too.

2. The new data set must be collected by a sensor with the same intrinsic parameters like focal length, detector size, flying at approximately the same height. This will ensure that the thresholds used for visual entropy can be applied to the new data set as well.

A new data set (aerial imagery) was generated by an airborne sensor under the above mentioned constraints. The best segmentation input parameter for building identification was found to be $h_c = 10$ during the training phase. The same input parameters are used to test the trained BN on the new data set. The TPR and FPR for the building and tree
The classes are shown in Table 4.9. The nadir mosaic, the truth map and the classification map are shown in Figures 4.30, 4.31 and 4.32. Quantitatively and qualitatively, it can be seen that the trained BN can perform at the same level on new data sets as it did on training data set. This proves the robustness of the training data and the training methods used to learn the parameters of the BN.

In order to enhance the robustness of the BNs and hence improve the classification results, the statistical causal relationships among the features can be learnt from known data. For this purpose, an optimization technique called Particle Swarm Optimization (PSO) [52] has been used. The complete algorithm of structural learning of BNs using PSO can be found in [53]. For our purpose, 16 particles were used for 2000 iterations to search for the best structure of the network which represents the causal relationships among the features in the best possible way. The structure obtained can be seen in Figure 4.29.

Table 4.9: True positive and false positive rates for new data set using the trained BN

<table>
<thead>
<tr>
<th>Class</th>
<th>TP</th>
<th>FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building</td>
<td>0.9260</td>
<td>0.0035</td>
</tr>
<tr>
<td>Trees</td>
<td>0.9679</td>
<td>0.0170</td>
</tr>
</tbody>
</table>

Figure 4.29: Plot of quality metric vs. color bandwidth for the case of $W_{building} = 0.1$, $W_{trees} = 0.9$ and rest of the weights 0
4.33. This figure illustrates that the physical causal relationships assumed by the expert (seen in Figure 3.13) are not same as the statistical causal relationships among the features. From this structure, one can observe that the elevation information from stereo images are closely related to each other both physically and statistically. Similarly, the elevation features from Lidar data are also closely related. Some of the features which were picked in the handcrafted network like the corner information and edge information are separated from the region node by three levels of nodes. This shows that these features will have less impact on region classification if information about the area of the region is known. Further research is warranted for structural learning of the BNs. It will not be discussed here as it is out of the scope of this work.
4.3 3D Object Modeling

The color bandwidth $h_c = 10$ has been chosen as the best segmentation input parameter from the previous section. The building surfaces identified by the BN are separated out from the other surfaces as shown in Figure 4.34. All the surfaces belonging to each separate building can be detected by using the morphological image processing technique of connected components.
Building 1:

The building shown in Figure 4.35 is used as an example to describe the complete process of 3D object modeling. There are two surfaces on the roof of this building and vertical facets from the roof to the terrain. For each surface, the initial 3D positions of the corners in the object space are available along with the plane equation. If the initial building elevation measurements from the stereo mosaics are used, the 3D model of the building would be as shown in Figure 4.36. Here the X, Y and Z coordinates are in referenced to a local grid and the units are in meters.

All the 8 corners of the building (4 for each of the two surfaces that constitute the roof) are projected onto individual video frames. Only those frames in which the building is visible are selected as demonstrated in Section 3.4. They are shown in Figure 4.37. These video images are also segmented using the same set of segmentation input parameters used for building identification ($h_s = 14, h_c = 10, M = 20$). The segmented images are illustrated in Figure 4.38. The initial estimates of the 3D corners are projected onto individual frames and matched with their corresponding points. The matched point pairs on some of the image frames are shown in Figure 4.39. The red stars represent the projected 3D corners while the blue circles represent the actual 2D corners detected from the video frames. The distances between the projected corners and actual corners can be seen in the
For each point, the sum of the distances between the projected corner and the actual corners in the images is minimized using LM algorithm, by varying the 3D position of the corner in the object space. The starting point for the algorithm is provided by the initial estimate of the 3D position for each corner. The error plots or the distance plots for all the 8 corners of the two surfaces after each iteration of the LM algorithm are illustrated in Figure 4.41. The maximum residual error after the optimization is about 9 pixels for one of the corners. But this is the sum of squares of distances between the projected corner and actual detected corner in pixels over 14 frames. The mean squared distance is $(9/14)$ pixels which is within one pixel error. A least squares plane is fitted through the optimized 3D corners of each surface. Here a RANSAC least squares plane fit algorithm produces the same results as the simple least squares plane fit algorithm because there are no outliers (bad matches). After the post-processing step, the 3D model of the building will be as
4.3. 3D OBJECT MODELING

Figure 4.39: Matched point pairs of corners of a building in video frames

Building 2:

The 3D model of another building circled in Figure 4.43 is presented in Figure 4.45. The residual errors for the corners after each iteration of the LM algorithm are plotted in Figure 4.44. Note that one of the corners still has a huge error after optimization. This is due to the fact that one of the projected corners is matched to a different point in the video frames because of noise in the segment maps. This noise is a result of the close proximity of the building to a tree and consequently, a part of the tree is segmented along with building surface in some of the video frames. Hence, there is no minimization of the error due to this ill-matched point pair. This anomaly can be eliminated by employing the RANSAC plane fit algorithm and performing a final evaluation of accurate 3D corners.
using Equations 3.62 and 3.63. Thus an accurate 3D model of the building as shown in Figure 4.45 can be obtained after the post-processing step.

**Building 3:**

The 3D model of yet another building circled in Figure 4.46 is illustrated in Figure 4.48. The residual errors for the corners after each iteration of the LM algorithm are plotted in Figure 4.47. Note that the small structures on the top of the building can be ignored because of two reasons. Firstly, the stereo mosaics do not provide a high resolution depth map to model these structures separate from the main rooftop structure. Secondly, most of these small structures are merged with the main structure during segmentation in the individual video frames and it is very difficult to obtain matched point pairs for projected corners.
4.3. 3D OBJECT MODELING

The accuracy of the reconstructed 3D building models cannot be expressed in the 3D object space due to unavailability of true 3D information of the buildings. Hence, the accuracy will be represented in the projected 2D image space here. In the 2D image space, the error of modeling or the distance between estimated corner point and true corner point can be specified in terms of pixels. After the optimization process is completed, the

Figure 4.41: Minimization of the distances between projected corners and actual corners in the video frames

Figure 4.42: Final 3D model of the building after local optimization
Figure 4.43: Building 2 used to illustrate the object modeling process

Figure 4.44: Minimization of the distances between projected corners and actual corners in the video frames

The final residuals (at Iteration number 10) in Figures 4.41, 4.44 and 4.47 represent the sum of the squared distances between the projected corner points and the actual detected corner points over all the frames in which the building is visible. This error has already been
4.3. 3D OBJECT MODELING

Figure 4.45: Final 3D model of the building after local optimization

Figure 4.46: A building used to illustrate the object modeling process

shown in Equation 3.61. This sum of squared distances $g_j$ is essentially the residual Sum of Squared Errors (SSE). The Mean Squared Error (MSE) can be obtained for each corner by dividing the SSE by the total number of frames ($N$) in which that particular building is visible. The Root Mean Squared Error (RMSE) for each corner $j$ (following the indexing used in Section 3.4) is obtained by taking the square root of the MSE of the 2D position of the corner point. Mathematically, it can be represented as

$$RMSE_j = \sqrt{\frac{1}{N}g_j(pixels)} \quad (4.1)$$

Table 4.10 provides the error statistics for all the three buildings reconstructed here. The
minimum, maximum and mean RMSE are calculated over all the corners of a building. In this table, the number of frames represents the number of frames in which the building under consideration is visible. The error statistics provided for building 2 are the final values after outlier removal and final plane fitting. From all the presented results, it can
be observed that the maximum RMSE between projected corners and the actual detected corners in 2D space is within a pixel.

Even with such low RMSE, it does not mean that the buildings are modeled accurately in the 3D object space. For example, building 3 has two surfaces which rise up at the center but this particular property of the roof is not captured in the final 3D model of the building even though the final RMSE is very low for all the corner points. This is because, the solar angle and the color information of the two surfaces make it impossible for any segmentation algorithm to detect the two different surfaces on the roof. So the final accuracy of the 3D model of a building reconstructed from visual imagery is limited by the vision algorithms which detect the various features of the building like surfaces, edges and corners.

<table>
<thead>
<tr>
<th>B no.</th>
<th>No. of corners</th>
<th>No. of frames</th>
<th>Min. RMSE</th>
<th>Max. RMSE</th>
<th>Mean. RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>14</td>
<td>0.0734</td>
<td>0.6619</td>
<td>0.1549</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>14</td>
<td>0.0088</td>
<td>0.527</td>
<td>0.1672</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>6</td>
<td>0.0575</td>
<td>0.8892</td>
<td>0.2111</td>
</tr>
</tbody>
</table>

Table 4.10: Final error statistics for each building
Chapter 5

Summary and Conclusions

The main objective of this thesis is to automate as much as possible, the end-to-end process of modeling 3D objects especially buildings in a scene from aerial video imagery as much as possible without compromising on the final quality of the reconstructed 3D models. For this the entire system is divided into three research goals. They have been previously listed in Chapter 1.1. These goals have been achieved as explained below:

1. The first research goal was to build stereo mosaics from aerial video imagery over a large scene automatically. This task has been accomplished using Parallel Ray Interpolation for Stereo Mosaicing (PRISM) technique. The aerial video frames are pre-processed to remove the rotational effects of the sensor movement (roll, pitch and yaw) from the images. The orientation-corrected video frames are then stitched into parallel-perspective stereo mosaics using the 3D translational coordinates of the sensor.

2. The second goal was 3D object identification and modeling from the stereo mosaics. The aim is to automatically identify all the polyhedral building surfaces in the scene and build a 3D model of the building. In order to automate the process of feature detection and geo-referencing of the 3D models, a third mosaic called the nadir mosaic has been built from the aerial video. Color segmentation is used to identify the homogeneous surfaces in the scene. Features like elevation information (from stereo mosaics), edge and corner information (from nadir mosaic), surface area, color information and visual entropy are extracted for each of the homogeneous surfaces. Bayesian Networks have been used to identify the building surfaces using the global (over the entire scene) statistics of the extracted features. The initial 3D models of the buildings are projected onto individual video frames and iterative optimization techniques have been used to automatically improve the accuracy of the 3D building
models.

3. The third goal was to identify the deficiencies of visual information to automatically identify and model 3D objects and overcome these shortcomings by fusing Lidar data with visual imagery. Bayesian Networks are again used for this purpose, resulting in improvement of the accuracy of the outputs of 3D object identification process. The final automated 3D object modeling process using visual information described in this work can be compared to the 3D object modeling technique using only Lidar data described in [54].

The accomplishments described above give a broad overview of the contributions made by this thesis work to an automated system for modeling 3D buildings from aerial video imagery. At a finer level, the following original contributions have been made by this work to the knowledge base in photogrammetry and 3D computer vision fields:

1. A segment-based mesh design for aerial triangulation without any prior 3D knowledge of the scene has been developed. This new design helps in avoiding visual artifacts in the parallel-perspective stereo mosaics that are built using ray interpolation. Consequently, the errors in the final 3D models of buildings are reduced.

2. A novel method to control the input parameters of vision algorithms like color segmentation using the data-driven probabilistic inference in Bayesian networks has been designed. This method automates the 3D object identification process and precludes the need for manual intervention to set the accurate input parameters for best quality of the final 3D building models.

The following journal publications represent the contributions made by this thesis work to the fields of photogrammetry and 3D computer vision:


The following conference publications were also generated by this thesis work:


**Recommendations for future work:**

1. In this work, only one level of BN was pursued because the truth data was available only for this level which classified regions into buildings, trees, grass and asphalt. If truth data like slant roof, flat roof, etc. is available for building surfaces, another level of BN can be included. If more truth data is available about the different types of building primitives, a third level of BN can be included in a hierarchical BN setup. The probabilities can be propagated from one level to the next higher level and the final decisions can be made at the highest level. The probabilistic inference and decision making at the highest level (building level) rather than region level provides more accurate input parameters for vision algorithms like segmentation.

2. In this work, only the input parameters for segmentation were set based on the probabilistic inference of BN. The parameters of other vision algorithms like deviation threshold in edge and corner detection, window size used for calculation of entropy, the elevation threshold to deem a point to be an inlier or an outlier can also be varied and the best set of parameters can be selected based on the final quality of the 3D building models generated.

3. Finally, this work involved the use of handcrafted BNs for the object identification process. These BNs were designed in such a way that they represent the physical causal relationship between different features. But statistically, the relationships vary from those shown in the handcrafted BNs as seen in the results section. So an important future work that can be done is structural learning of BNs [43] using optimization techniques like Particle Swarm Optimization (PSO) [52].
Appendix A

Mathematical Model of the Generalized Parallel-Perspective Stereo Mosaics

Consider the corresponding point pair \((x_l, y_l)\) and \((x_r, y_r)\) (from left and right mosaic respectively) of a 3D point in the object space \((X, Y, Z)\) captured by the sensor at viewpoints \((T_{xl}, T_{yl}, T_{zl})\) and \((T_{xr}, T_{yr}, T_{zr})\) as shown in Figure A.1. The focal length of the camera is \(F\) and the average flying height of the camera over the average terrain is \(H\).

![Figure A.1: Stereo Geometry of the generalized parallel-perspective stereo mosaics under 3D translation of the sensor (Courtesy: Zhigang Zhu et al.)](image)

Figure A.2 shows the projection of this 3D point \((X, Y, Z)\) along the \(X - Z\) plane of the left viewpoint \((T_{xl}, T_{yl}, T_{zl})\). Here the \(Y\) axis is coming out of the paper. Now using the
property of similar triangles,

\[ \frac{F}{x_l - \frac{f}{H} T_{xl}} = \frac{Z - T_{zl}}{X - T_{xl}} \]  

(A.1)

Simplifying the above equation, one can obtain the \( x \) coordinate of the projection of the 3D point on the left mosaic as shown in Equation A.2

\[ x_l = \frac{F}{Z - T_{zl}} \left( X - \frac{Z}{T_{zl}} T_{xl} \right) + \frac{F}{H} T_{xl} \]  

(A.2)

Similarly, the \( x \) coordinate of the projection of the 3D point on the right mosaic is given by Equation A.3

\[ x_r = \frac{F}{Z - T_{zr}} \left( X - \frac{Z}{T_{zr}} T_{xr} \right) + \frac{F}{H} T_{xr} \]  

(A.3)

Figure A.2: Projection of a 3D point along X-Z plane

Figure A.3 shows the projection of the 3D point \((X, Y, Z)\) along the \( Y - Z \) plane of the left viewpoint. Here the \( X \) axis is going into the paper. Again using the property of similar triangles,

\[ \frac{F}{d_y} = \frac{Z - T_{zl}}{Y - T_{yl}} \]  

(A.4)

\[ FY - FT_{yl} = \frac{d_y}{2} (Z - T_{zl}) \]  

(A.5)

From Figure A.3, it can be determined that

\[ T_{yl} \approx \frac{H}{F} \left( y_l - \frac{d_y}{2} \right) \]  

(A.6)
Substituting Equation A.6 in Equation A.5, we can obtain

\[ FY - H y_l + H \frac{d_y}{2} = \frac{d_y}{2} \left( Z - T_{zl} \right) \]  \hspace{1cm} (A.7)

Rearranging the terms in the above equation, the \( y \) coordinate of the projection of the 3D point on the left mosaic is obtained as shown in Equation A.8

\[ y_l = F \frac{Y}{H} - \left( \frac{Z - T_{zl}}{H} - 1 \right) \frac{d_y}{2} \]  \hspace{1cm} (A.8)

Similarly, the \( y \) coordinate of the projection of the 3D point on the right mosaic is given by Equation A.9

\[ y_r = F \frac{Y}{H} + \left( \frac{Z - T_{zr}}{H} - 1 \right) \frac{d_y}{2} \]  \hspace{1cm} (A.9)

![Figure A.3: Projection of a 3D point along Y-Z plane](image)

Combining the Equations A.2, A.3, A.8 and A.9, the mathematical model of the generalized parallel-perspective stereo mosaics is given by Equations A.10 and A.11.

\[ (x_l, y_l) = \left( F \frac{X - T_{xl}}{Z - T_{zl}} + F \frac{T_{xl}}{H} F \frac{Y}{H} - \left( \frac{Z - T_{zl}}{H} - 1 \right) \frac{d_y}{2} \right) \]  \hspace{1cm} (A.10)

\[ (x_r, y_r) = \left( F \frac{X - T_{xr}}{Z - T_{zr}} + F \frac{T_{xr}}{H} F \frac{Y}{H} + \left( \frac{Z - T_{zr}}{H} - 1 \right) \frac{d_y}{2} \right) \]  \hspace{1cm} (A.11)
Appendix B

Epipolar Geometry of the Generalized Parallel-Perspective Stereo Mosaics

In order to extract the depth/elevation map of a scene after building the stereo mosaics, one needs to know the epipolar constraints of the parallel-perspective stereo mosaics under 3D translational motion. The epipolar constraints are used to reduce the search space for finding corresponding point pairs in the left and right stereo mosaics and to remove the outliers in the point matches. To determine the epipolar constraints, consider the mathematical model of the generalized parallel-perspective stereo mosaics given by

\[
(x_l, y_l) = \left(\frac{FX - T_{xl}}{Z - T_{zl}} + \frac{F}{H} \frac{TX_l}{H} - \left(\frac{Z - T_{zl}}{H} - 1\right) \frac{d_y}{2}\right) \tag{B.1}
\]

\[
(x_r, y_r) = \left(\frac{FX - T_{xr}}{Z - T_{zr}} + \frac{F}{H} \frac{TX_r}{H} + \left(\frac{Z - T_{zr}}{H} - 1\right) \frac{d_y}{2}\right) \tag{B.2}
\]

Here \((x_l, y_l)\) and \((x_r, y_r)\) are the corresponding point pair (from left and right mosaic respectively), \((X, Y, Z)\) is the 3D point in the object space, \((T_{xl}, T_{yl}, T_{zl})\) and \((T_{xr}, T_{yr}, T_{zr})\) are the camera viewpoints for the columns in the left and right mosaics respectively where the point is visible. The focal length of the camera is \(F\), the distance between the fixed lines is \(d_y\), and the average flying height of the camera over the average terrain is \(H\). The mosaic displacement between two corresponding points in left and right mosaic can be defined as

\[
(\Delta x, \Delta y) = (x_r - x_l, y_r - y_l) \tag{B.3}
\]
The depth of this point can be derived from Equations B.1 and B.2 as

\[ Z = H \left( 1 + \frac{\Delta y}{d_y} \right) + \mathcal{T}_z \]  \hspace{1cm} (B.4)

where \( \mathcal{T}_z \) is the average deviation of the camera height given by

\[ \mathcal{T}_z = \frac{T_{zl} + T_{zr}}{2} \]  \hspace{1cm} (B.5)

Substituting Equation B.4 in Equation B.1, one can obtain

\[ x_l = F d_y \left( \frac{X - T_{xl}}{H (d_y + \Delta y) + d_y \frac{H}{2F} b_z} \right) + F \frac{T_{xl}}{H} \]  \hspace{1cm} (B.6)

\[ x_l = F d_y \left( \frac{X - T_{xl}}{H (d_y + \Delta y) + d_y \frac{H}{2F} b_z} \right) + F \frac{T_{zl}}{H} \]  \hspace{1cm} (B.7)

The scaled baselines in \( x \) and \( z \) directions can be defined as

\[ b_x = t_{xr} - t_{xl} = \frac{F}{H} (T_{xr} - T_{xl}) \]  \hspace{1cm} (B.8)

\[ b_z = t_{zr} - t_{zl} = \frac{F}{H} (T_{zr} - T_{zl}) \]  \hspace{1cm} (B.9)

From Equations B.9 and B.7, one can obtain

\[ x_l = F d_y \left( \frac{X - T_{xl}}{H (d_y + \Delta y) + d_y \frac{H}{2F} b_z} \right) + F \frac{T_{zl}}{H} \]  \hspace{1cm} (B.10)

\[ x_l = F d_y \left( \frac{X - T_{xl}}{H (d_y + \Delta y) + d_y \frac{H}{2F} b_z} \right) + F \frac{T_{zl}}{H} \]  \hspace{1cm} (B.11)

Simplifying Equation B.11 to obtain \( X \)

\[ X = \frac{H \left( x_l - \frac{FT_{xl}}{H} \right)}{F d_y} \left( d_y + \Delta y + \frac{b_z d_y}{2F} \right) + T_{xl} \]  \hspace{1cm} (B.12)

Modifying the above equation and using Equation B.8, one obtains

\[ F d_y (X - T_{xr}) = H \left( x_l - \frac{FT_{xl}}{H} \right) \left( d_y + \Delta y + \frac{b_z d_y}{2F} \right) - d_y H b_x \]  \hspace{1cm} (B.13)
Substituting Equation B.4 in Equation B.2, one can obtain

\[ x_r = F d_y \frac{X - T_{xr}}{H (d_y + \Delta y) - d_y \frac{H}{2F} b_z} + F \frac{T_{xr}}{H} \] (B.14)

Using Equation B.13 in Equation B.14,

\[ x_r = \left( x_l - \frac{FT_{xr}}{H} \right) \frac{(d_y + \Delta y + \frac{b_z d_y}{2F}) - d_y b_x}{(d_y + \Delta y) - \frac{d_y b_z}{2F}} + F \frac{T_{xr}}{H} \] (B.15)

The mosaic displacement in \( x \) direction is given by Equation B.3. Substituting Equation B.15 in Equation B.3, one obtains

\[ \Delta x = x_r - x_l = \frac{(x_l - \frac{FT_{xr}}{H}) \left( d_y + \Delta y + \frac{b_z d_y}{2F} \right) - d_y b_x}{(d_y + \Delta y) - \frac{d_y b_z}{2F}} + F \frac{T_{xr}}{H} - x_l \] (B.16)

After some algebraic simplification, for any point \((x_l, y_l)\) in the left mosaic the corresponding point \((x_r, y_r)\) will lie on an epipolar curve defined by

\[ \Delta x = \frac{b_z \Delta y + b_z d_y \left( x_l - \frac{b_z + T_{xr}}{2} \right)}{\Delta y + \frac{d_y - b_z d_y / 2F}{2F}} \] (B.17)

One should note that the epipolar constraints of the perspective-perspective stereo are different from those of the parallel-perspective stereo. First, \( \Delta x \) is a nonlinear function of the position of a point in the left mosaic \((x_l, y_l)\) as well as the mosaic displacement in \( y \) direction i.e., \( \Delta y \). So, the epipolar curve on the right mosaic is different for each point \((x_l, y_l)\) on the left mosaic. Any image column with different \( y_l \) coordinates are generated from different viewpoints. Due to this reason, in the case of 3D translational motion of the sensor, the left and right mosaics cannot be aligned even at the points whose depths are \( H \) since \( x_l \neq x_r \) and \( y_l \neq y_r \). Such stereo mosaics cannot be used for stereoscopic viewing.

On the other hand, if the sensor has only 2D translational motion, then \( b_z = 0 \) and the epipolar curve corresponding to the point \((x_l, y_l)\) in the right mosaic is defined by

\[ \Delta x = \frac{b_z \Delta y}{\Delta y} \] (B.18)

In this case, the two mosaics can be aligned for all the points whose depths are \( H \). Usually, the motion of an airborne sensor can be considered to be 2D translational motion since the change in the \( Z \) coordinates of the sensor are very small compared to the depth of the
fixation plane $H$. Hence, the stereo mosaics generated using airborne imagery can be used for stereoscopic viewing as well as stereo matching. If the sensor has only 1D translational motion, then $\Delta x = 0$ and the epipolar curves are just horizontal lines.
Appendix C

Orthorectification of Nadir Mosaic and Geo-referencing of a 3D Point

The nadir mosaic which is built using ray interpolation technique is already orthorectified in the dominant motion direction of the sensor where there is parallel projection involved. This can be understood easily by visualizing a nadir-looking linear pushbroom camera flying over a scene. Figure C.1 illustrates this configuration. As we can see each column of pixels will have the same \( Y \) geo-coordinate as the viewpoint of the pushbroom camera irrespective the height of the point represented by that particular pixel. There is a need to orthorectify the nadir mosaic in the other direction where there is perspective projection involved.

Let us assume the viewpoint of the imaginary linear pushbroom camera in PRISM algorithm is \( (T_{x_i}, T_{y_i}, T_{z_i}) \). The sensor is flying in the direction pointing in to the paper. As shown in Figure C.2, a point (here, a corner of a building) which is at a height of \( Z \)
from the ground appears to be at \((T_{cxg}, T_{cyg}, T_{czg})\) on the ground at a distance of \(H\) from the sensor while it is actually at \((T_{cx}, T_{cy}, T_{cz})\). \(H\) is the average flying height of the sensor from the ground plane and \(Z\) is determined using the stereo information available from the stereo mosaics or is obtained from 3D Lidar point cloud. \(T_{zi}, T_{czg}\) and \(T_{cz}\) are the \(Z\)-coordinates of the sensor, ground plane and the corner of the building in the geo-reference system being used. Using similar triangles, we can write

\[
\frac{T_{cxg} - T_{xi}}{T_{cx} - T_{xi}} = \frac{H}{H - Z} \tag{C.1}
\]

\[
(T_{cxg} - T_{xi})(H - Z) = H(T_{cx} - T_{xi}) \tag{C.2}
\]

\[
T_{cx} = T_{cxg} - \frac{(T_{cxg} - T_{xi})Z}{H} \tag{C.3}
\]

Since the nadir mosaic is orthorectified in the dominant motion direction, \(T_{cy} = T_{yi}\). Thus, we can orthorectify the nadir mosaic and geo-reference any 3D point.
Bibliography


