SAMATS
Semi-Automated Modelling And Texturing System

Joe P. Hegarty BSc.

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Philosophy
(MPhil)

Digital Media Centre,
Faculty of Applied Arts,
Dublin Institute of Technology

2006

Research Supervisor:

Dr. James D. Carswell
ABSTRACT

The creation of detailed 3D (three-dimensional) building models has become an area of considerable research over the last couple of decades. The accurate modelling of buildings offers LBS (Location Based Services) applications in planning, cultural heritage, tourism and e-commerce among others. The approach taken by the majority of contemporary modelling systems that use terrestrial imagery taken from arbitrary locations requires the user to carry out manual correspondences across the image set. These correspondences are used for two purposes. Firstly, the correspondences are used to determine the exterior orientation parameters (position and orientation) of the cameras used to capture each image. Secondly, (and more importantly) the correspondences are used to group highlighting primitives (points or lines placed in the images of the image set) that highlight the same building feature. Requiring the user to carry out these correspondences manually is a very time consuming process, which greatly limits the scalability of such systems.

This thesis investigates techniques that reduce the amount of user interaction required to model a building. SAMATS, a Semi-Automated Modelling And Texturing System, has been created to demonstrate these techniques. Modelling systems that automate the modelling process generally add restrictions and place constraints on what building types the system is capable of modelling. These restrictions and constraints make such systems less flexible, greatly reducing that usefulness. SAMATS demonstrates that it is possible to automate the steps that have traditionally been carried out manually while maintaining the system’s flexibility to produce geometrically accurate photorealistic 3D building models of arbitrarily shaped buildings.

SAMATS does not require the user to carry out correspondences manually. SAMATS makes use of georeferenced terrestrial imagery so that the cameras’ exterior orientation parameters are provided. Also, while contemporary
Abstract

modelling systems have required the user to carry out the correspondences manually in order to group highlighting primitives, SAMATS demonstrates how these correspondences can be determined automatically using the georeferencing information provided. This makes it possible to eliminate the manual correspondence step from the modelling process completely, reducing user interaction substantially. Although obtaining accurate positional information for the imagery is still a bottleneck, as this information becomes more readily available with the use of GPS (global positioning system) enabled cameras, digital compasses, and gyroscopic sensors, the reduced user interaction offered by SAMATS’ approach will likely outweigh the burden of providing this additional georeferencing information.
DECLARATION

I certify that this thesis which I now submit for examination for the award of Masters of Philosophy, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for an award in any other Institute or University.

The work reported on in this thesis conforms to the principles and requirements of the Institute’s guidelines for ethics in research.

The institution has permission to keep, to lend or to copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Joe P. Hegarty

________________

Date: ____________
Acknowledgements

ACKNOWLEDGEMENTS

First I would like to thank James Carswell for introducing me to the field of photogrammetry and building reconstruction. He gave me the freedom to choose the direction of this research while providing valuable support and feedback. I would also like to thank Charlie Cullen who offered many helpful suggestions on how to improve the structure of this thesis. I would like to thank everyone who works at the DMC, whose company made the slowest days fly by.

Finally, I would like to thank my family, without whose support I would not have been able to complete this research.
# TABLE OF CONTENTS

ABSTRACT ........................................................................................................... I
DECLARATION ..................................................................................................... III
ACKNOWLEDGEMENTS ................................................................................... IV
TABLE OF CONTENTS ....................................................................................... V
LIST OF FIGURES ............................................................................................... IX
LIST OF TABLES ................................................................................................. XV

1 INTRODUCTION ......................................................................................... 1
  1.1 MOTIVATION ......................................................................................... 3
  1.2 APPROACH ............................................................................................. 4
    1.2.1 System Overview .............................................................................. 5
  1.3 CONTRIBUTIONS ..................................................................................... 6
  1.4 SUCCESS CRITERIA ............................................................................... 7
  1.5 THESIS OUTLINE ................................................................................ 7
  1.6 SUMMARY ............................................................................................... 8

2 BACKGROUND AND RELATED RESEARCH ............................................ 10
  2.1 MODELLING SYSTEM TAXONOMY ....................................................... 11
  2.2 GEOREFERENCED IMAGERY ................................................................. 16
  2.3 CAMERA CALIBRATION ......................................................................... 19
  2.4 STRUCTURE FROM MOTION .................................................................. 20
  2.5 STEREO IMAGERY ................................................................................ 22
  2.6 RANGE SCANNING ................................................................................ 26
  2.7 IMAGE-BASED RENDERING AND MODELLING ..................................... 27
  2.8 SEMI-AUTOMATED MODELLING SYSTEMS ......................................... 30
  2.9 AUTOMATED MODELLING SYSTEMS .................................................. 32
  2.10 TEXTURING .......................................................................................... 34
  2.11 DISCUSSION ........................................................................................ 36
  2.12 CONCLUSIONS .................................................................................... 39

3 SYNTHETIC IMAGE GENERATION .......................................................... 41
  3.1 SCENE CREATION .................................................................................. 41
  3.2 SCENE CAPTURE TOOL ......................................................................... 42
  3.3 SYNTHETIC IMAGERY VS. PHOTOGRAPHIC IMAGERY ...................... 43
    3.3.1 Interior and Exterior Orientation Parameter Accuracy .................... 43
    3.3.2 Image Complexity .......................................................................... 44
  3.4 SUMMARY ............................................................................................... 45

4 EDGE HIGHLIGHTING TOOL ................................................................. 46
  4.1 HIGHLIGHTING EDGES ......................................................................... 46
  4.2 HIGHLIGHTING RULES ......................................................................... 52
# Table of Contents

4.3 DISCUSSION ................................................................. 53  
4.4 CONCLUSIONS .............................................................. 54  

5 MODEL CREATION TOOL .............................................. 55  
  5.1 EDGE DETERMINATION .............................................. 55  
    5.1.1 Line Projection .................................................. 56  
    5.1.2 Triangle Intersection ......................................... 58  
    5.1.3 Correspondence Determination ............................. 61  
    5.1.4 Edge Averaging .................................................. 77  
    5.1.5 Vertex Merging ................................................... 78  
    5.1.6 Secondary Edge Determination ............................. 80  
  5.2 SURFACE STRUCTURE DETERMINATION ...................... 81  
    5.2.1 Surface Determination ....................................... 81  
    5.2.2 Surface Aligning ............................................... 82  
    5.2.3 Surface Triangulation ....................................... 85  
  5.3 EXPORTING ............................................................ 86  
  5.4 DISCUSSION ............................................................ 86  
  5.5 CONCLUSIONS .......................................................... 88  

6 TEXTURE EXTRACTION TOOL .......................................... 89  
  6.1 INITIALISATION ....................................................... 89  
    6.1.1 Triangle Setup .................................................. 90  
    6.1.2 Image Setup ..................................................... 93  
    6.1.3 Image Contribution Determination ........................ 95  
  6.2 TEXTURE DETERMINATION PER TRIANGLE .................... 96  
    6.2.1 Single Image Texture Capture .............................. 97  
    6.2.2 Texture Blending ............................................... 103  
    6.2.3 Triangle Texture Filling ..................................... 104  
  6.3 TEXTURE PACKING ................................................... 107  
  6.4 EXPORTING ............................................................ 110  
  6.5 DISCUSSION ............................................................ 110  
  6.6 CONCLUSIONS .......................................................... 112  

7 EXPERIMENTS AND RESULTS ......................................... 113  
  7.1 JOHN HANCOCK BUILDING ....................................... 113  
    7.1.1 Image Set Acquisition ....................................... 114  
    7.1.2 Edge Highlighting Statistics ................................ 114  
    7.1.3 Results and Analysis .......................................... 116  
  7.2 OFFICE BLOCK ......................................................... 116  
    7.2.1 Image Set Acquisition ....................................... 117  
    7.2.2 Edge Highlighting Statistics ................................ 117  
    7.2.3 Results and Analysis .......................................... 118  
  7.3 BARN ................................................................. 119  
    7.3.1 Image Set Acquisition ....................................... 120  
    7.3.2 Edge Highlighting Statistics ................................ 121  
    7.3.3 Results and Analysis .......................................... 121
Table of Contents

7.4  HOUSE WITH EXTENSION ................................................................. 122
    7.4.1  Image Set Acquisition ......................................................... 123
    7.4.2  Edge Highlighting Statistics ............................................. 123
    7.4.3  Results and Analysis ......................................................... 123
7.5  COTTAGE ................................................................. 125
    7.5.1  Image Set Acquisition ......................................................... 126
    7.5.2  Edge Highlighting Statistics ............................................. 126
    7.5.3  Results and Analysis ......................................................... 127
7.6  PYRAMID ................................................................. 128
    7.6.1  Image Set Acquisition ......................................................... 128
    7.6.2  Edge Highlighting Statistics ............................................. 129
    7.6.3  Results and Analysis ......................................................... 129
7.7  SAMATS STABILITY TESTS ..................................................... 131
    7.7.1  John Hancock Stability Tests ............................................ 132
    7.7.2  Pyramid Stability Tests ..................................................... 135
7.8  DISCUSSION ................................................................. 138
7.9  CONCLUSIONS ................................................................. 138

8  CONCLUSIONS AND FUTURE WORK ........................................ 140
    8.1  RESEARCH QUESTIONS ANSWERED .......................................... 141
    8.2  LIMITATIONS ................................................................. 142
    8.3  CONTRIBUTIONS ............................................................... 143
    8.4  SUCCESS CRITERIA ............................................................ 144
    8.5  FUTURE WORK ................................................................. 146
        8.5.1  Edge Highlighting Tool .................................................. 146
        8.5.2  Model Creation Tool ..................................................... 147
        8.5.3  Texture Extraction Tool ................................................ 149
    8.6  FINAL NOTE ................................................................. 151

BIBLIOGRAPHY ................................................................. 152

APPENDIX A: CODE OVERVIEW .................................................. 156
    SYSTEM AND SOFTWARE REQUIREMENTS ....................................... 156
    TEST DATA SET USE ............................................................. 157
    EDGE HIGHLIGHTING TOOL ............................................................ 157
        Code Overview ............................................................... 158
        Compilation Instructions ..................................................... 158
    MODEL CREATION TOOL ............................................................. 158
        Code Overview ............................................................... 158
        Compilation Instructions ..................................................... 159
    TEXTURE EXTRACTION TOOL .......................................................... 159
        Code Overview ............................................................... 159
        Compilation Instructions ..................................................... 160

APPENDIX B: GLOSSARY ............................................................. 161

APPENDIX C: PUBLICATIONS .......................................................... 167
LIST OF FIGURES

Figure 1.1 This graph shows how modeling systems generally lie on the line separating Flexibility and Level of User Interaction. The MIT City Scanning project [7] [45] [46] is an automated system, which requires little user interaction but is quite restricted on the types of building it is capable of modeling. 3ds Max [3], Maya [2], and SketchUp [1] are all examples of manual systems that are very flexible but require a large amount of user interaction. Façade [9] [10] [11] is a semi-automated system, which lies between the two extremes. The goal of SAMATS is to move as far into the upper left region as possible ........................................... 2

Figure 1.2 By projecting the edges in the two images out to infinity using the correct interior and exterior orientation parameters, their line of intersection (highlighted in bold) represents the location of the edge in world-space .................. 5

Figure 1.3 SAMATS system diagram. The blue text boxes each represent one of the core tools of SAMATS. The red text boxes each represent input required by SAMATS. The green text boxes represent intermediate produces from the tools. The gold text box represents the final textured building model .............................................. 6

Figure 2.1 A taxonomy of modeling systems based on the level of interaction required of the user ................................................................................................................................................. 11

Figure 2.2 A scene created and rendered inside SketchUp [41] ........................................... 13

Figure 2.3 A screenshot from the upcoming Playstation 3 game, The Getaway [40] ........................................................................................................................................................................... 13

Figure 2.4 A screenshot of the Campanile, Berkeley’s clock tower, being modeled in Façade [9] [10] ............................................................................................................................................................. 14

Figure 2.5 A screenshot of three building models produced in the MIT City Scanning Project [7] [45] [46]. The red spheres indicate the spherical mosaics that capture the scene ........................................................................ 15

Figure 2.6 A spherical mosaic from the MIT City Scanning project [7] [45] [46] ...... 15

Figure 2.7 A photograph of the Argus rig [8] .................................................................................. 16

Figure 2.8 A picture of the Ricoh Pro G3 GPS Camera [38] ............................................................ 18

Figure 2.9 A photo of the MTi miniature gyro-enhanced Attitude and Heading Reference System [50] .................................................................................................................................................. 18

Figure 2.10 The calibration rig from Faugeras’ book [16] .............................................................. 20

Figure 2.11 The resulting reconstruction of a scene from [44]. Note that only edges are recovered and these edges are not connected together (i.e. they are floating edges) .................................................................................... 22

Figure 2.12 A photograph of a stereo rig from Faugeras’ book [16] ........................................... 23
List of Figures

Figure 2.13 Debevec’s model-based stereo [9] [11]. The two images at the top are the original key and offset images. If the offset image is projected onto a flat geometric plane representing the building front face, it appears as the warped offset image from the key camera’s point of view. Correspondences can then be determined between the key and warped offset to create a disparity depth map, which can be used to increase the geometric complexity of the rendered model...........25

Figure 2.14 Michelangelo’s David being laser scanned for the Digital Michelangelo Project [27]..........................................................................................................................26

Figure 2.15 Debevec’s view-dependent texture mapping [9] [11]. In the top left image we see the geometric model of a school from viewpoint A. In the upper right image we see the school from the same viewpoint but with the texture captured from this viewpoint applied to the model. Notice how the windows appear to be receded even though the geometric model does not capture this. When the model is viewed from viewpoint B with the texture from viewpoint A (lower left image) the façade appears distorted. When the texture from viewpoint B is used though (lower right image) the illusion of depth is restored..................28

Figure 2.16 Policarpo et al. [37] use a single image (top) to create the reconstructed scene (below). When viewed from a similar viewpoint the scene appears quite convincing..............................................................31

Figure 2.17 Line geometry recovered by Petsa and Karras [36]. Parallel and perpendicular constraints are used to recover the location of lines more robustly...........33

Figure 2.18 Coorg’s median estimation technique [7] used to create the final texture..................................................................................................................35

Figure 3.1 A screenshot of a textured building on a textured ground plane inside 3dsMax ........................................................................................................................42

Figure 3.2 Example image created using the Scene Capture Tool ......................43

Figure 4.1 A screenshot of the Edge Highlighting Tool ........................................46

Figure 4.2 A green skyscraper scene. An image is taken from each of the four cardinal corners of the skyscraper. Each of the building edges is highlighted using a red primary line........................................................................................................47

Figure 4.3 For vertical edges it is trivial to create arbitrarily large angles of intersection between the projected triangle planes ........................................48

Figure 4.4 For horizontal edges near camera level it is more difficult to create large angles of intersection between the projected triangle planes ......................49

Figure 4.5 When the offset camera is placed anywhere on the circle, the cameras will be a fixed distance from each other and the direction of displacement will be is perpendicular to the KeyTarget vector. Every point on the circle will give the same result when recovering the location of a point ........................................50

Figure 4.6 When recovering line features, the perpendicular direction of the displacement must be considered. The accuracy is at a maximum when the displacement is perpendicular to the orientation of the line feature and at a
minimum when the displacement is parallel. At the maximum, the accuracy is the same as when recovering point features, but at the minimum the accuracy is zero... 50

Figure 4.7 The same scene from Figure 4.2. This time red primary lines are used to highlight the vertical edges while yellow secondary lines are used to highlight the horizontal edges... 51

Figure 4.8 When only primary lines are used to highlight the green skyscraper (Figure 4.2) the outline shown to the left is recovered. As can be seen, only the vertical edges have been recovered correctly. None of roofline edges were recovered and only a single baseline edge was recovered, which is clearly not located in the correct location. When proper use of primary and secondary edges is made (Figure 4.7), the outline to the right is recovered... 52

Figure 5.1 This diagram shows the relationship between a pixel in an image and its corresponding 3D ray... 57

Figure 5.2 The images above show the scene from Figure 4.7, with each of the four images above adding an additional set of projected triangles from the next image of the image set... 59

Figure 5.3 The lines of intersection between the projected primary line triangles from Figure 5.2... 60

Figure 5.4 Close-up of the lines of intersection from Figure 5.3... 60

Figure 5.5 Using the ratio of the coverage over the extended line across both triangles as the only criteria for rating each intersection does not provide sufficient information to differentiate between valid and invalid intersections... 62

Figure 5.6 A 2D example of the intersection rating problem. Three cameras are used to locate two points... 63

Figure 5.7 The values returned by Equation 5.5 for various values of ScalingFactor, PowerFactor, and Distance. Note that Distance runs along the x-axis, with the functions y value representing the functions result... 66

Figure 5.8 The three partial comparisons used to determine the similarity between two intersection lines... 67

Figure 5.9 Intersection points when comparing the lines A, B, and C... 69

Figure 5.10 Intersection points when comparing the lines A, B, and C... 70

Figure 5.11 Intersection points when comparing the lines A, B, and C... 71

Figure 5.12 Intersection points when comparing the lines A, B, and C... 72

Figure 5.13 After the intersection rating step, every triangle will store a reference to two other triangles... 73

Figure 5.14 Three triangles, all with the same GSS... 74
List of Figures

Figure 5.15  Four triangles representing a primary edge with three of the triangles forming a core group with a fourth referencing two of the triangles of the core group without being referenced itself........................................75

Figure 5.16  Six triangles forming two separate GSSs. The black line represents the group after merging..........................................................76

Figure 5.17  The intersection lines between all group members are averaged to form the final primary edge..................................................77

Figure 5.18  Connected primary edges need to have the position of their end vertices averaged so that they are coincident at the same location........79

Figure 5.19  Secondary edges are recovered by connecting primary edges whose primary lines where connected during the edge highlighting phase..................80

Figure 5.20  The green outline represents a valid surface in the model, while the red outline is an invalid surface ........................................82

Figure 5.21  The surfaces are on the same side of the edge vector. Therefore the normal of the slave surface needs to be inverted........................84

Figure 5.22  The surfaces are on opposite sides of the edge vector. Therefore the surfaces are correctly aligned..............................................84

Figure 5.23  The normal of the first surface intersected by the ray from the camera must point towards the camera.........................................85

Figure 5.24  An example of a surface being triangulated........................................86

Figure 6.1  Triangle rendered using its own special camera..................................90

Figure 6.2  The camera position \( \mathbf{p} \), the look at point \( \mathbf{l} \) and the \( \mathbf{u} \) vector are used to create the \( \mathbf{V} \) (view) matrix. The vertical field of view VFOV and the aspect ratio of the AABB are used to create the \( \mathbf{P} \) (projection) matrix .........................93

Figure 6.3  (Top) An image of a cottage. (Bottom) The projection of the image onto the reconstructed building model. Note that some areas of the projected image appear brighter than others. The brighter the area, the more the texture will contribute to the final blended texture (6.2.2) .................................................94

Figure 6.4  The view vector points N. The direction of the blue, orange, green, and red surface normals point NE, SE, SW, and NW respectively. The dot product between the view vector and both the blue and red surface normals is positive; therefore the image from this camera cannot contribute to either of these surfaces since they are pointing away from the camera. The dot product between the view vector and both the orange or green surfaces is negative; therefore the image from this camera may contribute to these surfaces. Note that even though an image may pass this test, it may not contribute to a surface’s final texture since the camera may be occluded by part of the building (Section 6.2.1).................................96

Figure 6.5  A simple scene with a camera and two spheres. The camera is able to view the black highlighted portions of each sphere. Points A and B are in this
highlighted portion while point C is not. Although the surface normal at point C is pointing towards the camera, point C is being occluded by the red sphere

Figure 6.6  The scene from Figure 6.5 as viewed from the camera

Figure 6.7  The occlusion map for the camera from the scene in Figure 6.5

Figure 6.8  By using custom depth planes to enclose the model, the [0…1] depth range is more fully utilised

Figure 6.9  Using the near custom depth plane as the origin of a point’s distance from the camera would shift the relative distance considerably. Point A is 32 units from the camera’s focal point. Point B is 44 units from the camera’s focal point. Therefore point A is 32/44 (or 8/11) times the distance of point B from the camera’s focal point. If the near custom depth plane were used, point A would be 2 units away and point B would be 14 units away. Therefore, point A would be only 2/14 (or 1/7) times the distance of point B, which is not the case. Note that even using the standard near clip plane shifts the origin by 8 units. To correct this problem the distance of the near and far custom depth planes are passed to the shaders so that the relative distance can be correctly rescaled

Figure 6.10  In mipmap level 0 there is a small blue texture. The outline of the triangle that uses this texture is highlighted in white. At level 1, the texture is halved in size along each axis with black and blue texels being averaged together to form darker blue texels. The darkening gets progressively worse at higher mipmap levels until at level 3 there is only a single texel, which is the average of all the texels from the original texture

Figure 6.11  Triangle texture surface with the empty regions shown

Figure 6.12  The triangles lower left vertex accesses the triangle texture at the red X. Texels are only filled with colour data if their centre point is inside the triangle. The two blue texels above are the only texels with their centre points inside the triangle. To find the first non-empty texel, a spiral search is performed starting at the texel accessed by the vertex’s texture coordinates

Figure 6.13  The triangle texture varies in colour quite considerable along the bottom edge. By sweeping the edge texels outwards, neither darkening nor colour bleeding will occur at higher mipmap levels

Figure 6.14  The texture packing algorithm simply orders the triangle textures by width. The algorithm then iteratively increases the number of columns until the width becomes equal to or greater than the height. The iteration that has the minimum difference between the width and height is chosen as the best configuration

Figure 6.15  The triangle texture is stored in a 512 x 512 texture. Its packed form is only 212 x 102. The mipmap hierarchy is generated from the triangle texture and the 256 x 256 mipmap level is used since 256 is greater than 212, but less than 424 (twice 212)

Figure 6.16  The packed texture of the barn from Section 7.3
List of Figures

Figure 7.1   The highlighted edges of the John Hancock building test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow .......................... 114

Figure 7.2   Comparison of the original John Hancock building test scene (left) with the reconstructed model (right) ........................................................................................................ 115

Figure 7.3   The highlighted edges of the office block test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow................................. 117

Figure 7.4   Comparison of the original office block test scene (top) with the reconstructed model (bottom) ........................................................................................................ 119

Figure 7.5   The highlighted edges of the barn test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow ................................. 120

Figure 7.6   Comparison of the original barn test scene (top) with the reconstructed model (bottom) ........................................................................................................ 122

Figure 7.7   The highlighted edges of the house with extension test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow. Note that the green edge (edge A) is not visible in any image taken from ground level ........... 124

Figure 7.8   Comparison of the original house with extension test scene (top) with the reconstructed model (bottom) ........................................................................................................ 125

Figure 7.9   The highlighted edges of the cottage test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow ................................. 126

Figure 7.10  Comparison of the original cottage test scene (top) with the reconstructed model (bottom) ........................................................................................................ 127

Figure 7.11  The highlighted edges of the pyramid building test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow ................. 129

Figure 7.12  Comparison of the original pyramid test scene (top) with the reconstructed model (bottom) ........................................................................................................ 130

Figure 7.13  A graph of the results from Table 7.7 ......................................................... 132

Figure 7.14  A bar graph of the results from Table 7.8 ......................................................... 134

Figure 7.15  A bar graph of the results from Table 7.9 ......................................................... 136

Figure 7.16  A bar graph of the results from Table 7.10 ......................................................... 137

Figure 8.1   SAMATS plotted on the Flexibility Vs. Level of User Interaction graph from Figure 2.1 ........................................................................................................ 146

Figure 8.2   Screenshot of the pavement from the ATI Toy Shop technical demonstration that demonstrates parallax occlusion mapping. The 3D structure of the cobble stones is clearly visible even though the pavement is actually made up of flat polygons ........................................................................................................ 150
List of Tables

LIST OF TABLES

Table 2.1 Shows the amount of user interaction required to model a building for each of the 11 modelling systems shown in Figure 2.1 ................................. 12
Table 2.2 Shows the flexibility of the 11 modelling systems shown in Figure 2.1 ...... 12
Table 5.1 A_\text{ intersecting } B_\text{ configuration} .............................................................. 70
Table 6.1 Example of a pixel being blended from 3 surfaces ................................. 103
Table 7.1 John Hancock test scene edge highlighting statistics.................................. 115
Table 7.2 Office block edge highlighting statistics ...................................................... 118
Table 7.3 Barn edge highlighting statistics ................................................................. 120
Table 7.4 House with extension edge highlighting statistics ................................. 123
Table 7.5 Cottage edge highlighting statistics .............................................................. 126
Table 7.6 Pyramid edge highlighting statistics .............................................................. 129
Table 7.7 The results for the John Hancock building test scene adding position error only ........................................................................................................ 132
Table 7.8 The results for the John Hancock building test scene adding orientation error only ........................................................................................................ 133
Table 7.9 The results for the Pyramid test scene adding position error only ............... 135
Table 7.10 The results for the Pyramid test scene adding orientation error only ........... 136
1 INTRODUCTION

The creation of detailed 3D (three-dimensional) building models has become an area of considerable research over the last couple of decades [6] [7] [9] [10] [11] [23] [28] [34] [36] [37] [45] [46] [52] [53] [55]. Now that personal computers are capable of rendering complex 3D scenes, accurate building models are used for more than just specialized applications. Accurate building models are used in e-commerce and LBS (Location Based Services) applications, such as virtual shopping malls [17], virtual tours [4], cultural heritage conservation [33] and city planning [5].

One particular approach to building modelling involves using terrestrial imagery\(^1\) as reference [6] [7] [9] [10] [11] [45] [46] [55]. This thesis investigates techniques for automating the modelling of buildings using such imagery. The system developed to demonstrate these techniques is called SAMATS, which stands for Semi-Automated Modelling And Texturing System.

At a high level, there are three broad categories of modelling system that can be used to create building models (Section 2.1):

- **Manual Systems** – These systems give the user complete flexibility to model almost any structure imaginable. Although these systems are very flexible, both expert knowledge and a large amount of user interaction are required to model anything beyond the most rudimentary of models.

- **Semi-Automated Systems** – These systems generally get the user to highlight building features across an image set and then make correspondences between those features. Features being identifiable points or edges on the building, and the image set being the collection of images of the building taken from differing locations. While often not as flexible as manual systems, semi-automated systems generally reduce the amount of user interaction and the level of expertise required of the user (Section 2.8).

---

\(^1\) Words or phrases highlighted in italics are defined in the Glossary.
1 Introduction

- **Automated Systems** – These systems automate much or even all of the modelling process. As will be seen in Section 2.9, most automated modelling systems impose restrictions and constraints on what building types can be modelled.

Along with the three high level categories of modelling system there are many lower level choices that must be considered when designing a building modelling system that uses terrestrial imagery:

Figure 1.1 This graph shows how modelling systems generally lie on the line separating Flexibility and Level of User Interaction. The MIT City Scanning project [7] [45] [46] is an automated system, which requires little user interaction but is quite restricted on the types of building it is capable of modelling. 3ds Max [3], Maya [2], and SketchUp [1] are all examples of manual systems that are very flexible but require a large amount of user interaction. Façade [9] [10] [11] is a semi-automated system, which lies between the two extremes. The goal of SAMATS is to move as far into the upper left region as possible

- **Camera Quality** – Does the system expect the imagery to be taken using a calibrated camera? Using a calibrated camera makes it easier to determine the exact location of points in world-space.

- **Georeferencing Information** – Do the images need to be georeferenced? Georeferencing information can be used to aid the modelling process. This information can also be used to place the model in some world reference frame, making it easier to drop into an existing city scene.

2
1 Introduction

- **Fidelity Vs. Speed** – Do we want the final model tuned for rapid rendering or high visual fidelity? If the model is to be used as a stand-alone model, special techniques can be used to improve the realism of the rendered model at the cost of greater rendering complexity. However, if the model is to be used in a city scene, it would be desirable to keep the rendering complexity as low as possible since a large number of other objects would also need to be rendered.

1.1 Motivation

From the three high level categories of modelling system presented above, it is clear that an ideal modelling solution would provide the flexibility found in manual systems yet have the minimal user interaction found in automated systems. This is what semi-automated systems try to achieve, but generally need to make compromises on both sides (Figure 1.1). As will be seen in Sections 2.8 and 2.9, systems that require more user interaction are, in general, more flexible than those that require less user interaction. The amount of user interaction also has a direct correlation to the length of time required to model a building. As a result of this observation:

- The first guiding principle of the SAMATS design was to reduce user interaction to an absolute minimum, while still providing the flexibility of the more user intensive systems.

A disadvantage of many of the systems found during research is that they require the user to have expert knowledge in order to use them. This greatly reduces the accessibility of such systems. As a result of this observation:

- The second guiding principle of the SAMATS design was to make it as straightforward to use as possible, so that anyone could create a textured building model with minimal supervision.

Note that the first guiding principle of reduced user interaction lends itself to this second guiding principle, since reduced user interaction results in the need for reduced prior training overall.
1.2 Approach

Many contemporary modelling systems are able to reduce user interaction by setting restrictions and placing constraints on what the system can model. However, these restrictions and constraints can have the undesirable affect of making such systems less flexible. In order to avoid this situation SAMATS uses a different approach to reduce user interaction. Instead of setting restrictions and placing constraints on what SAMATS can model, additional information requirements are placed on the input data. This results in a system that is flexible like manual/semi-automated systems, while still requiring only minimal user interaction.

SAMATS thus places two additional requirements on the input data:

- **Rectified Imagery** – The images need to be taken using a calibrated camera so that the interior orientation parameters of the camera are known.

- **Georeferenced Imagery** – The images need to be georeferenced so that the exterior orientation parameters of the camera are known for each image.

The camera needs to be calibrated since it is much easier (and more reliable) to determine the location of a point relative to the camera when the interior orientation parameters are known. Non-calibrated cameras could be used, however, considering the relative ease with which cameras can be calibrated (Section 2.3), there would be little reason to introduce such unnecessary error.

Images need to be georeferenced for two reasons:

- Firstly, (and most importantly) the exterior orientation parameters are used to eliminate the manual correspondence step required by the majority of contemporary modelling systems.

- Secondly, if the model is intended to be georeferenced it eliminates the need to place control targets of known position in the scene.
SAMATS determines the location of building edges based on the following principle; consider two images of the front of a building taken from differing locations (Figure 1.2). If the exact position and orientation of the camera is known for each image, then the exact location of any edge visible in both images can be determined.

1.2.1 System Overview
SAMATS uses three tools to create a geometrically accurate, photorealistic model of a building (Figure 1.3):

1) The **Edge Highlighting Tool** is used to highlight building edges in the images of the image set.

2) The **Model Creation Tool** is used to create the *geometric model* of the building.

3) The **Texture Extraction Tool** is used to apply the façade texture information to the model.
1 Introduction

![SAMATS System Diagram](image)

Figure 1.3 SAMATS system diagram. The blue text boxes each represent one of the core tools of SAMATS. The red text boxes each represent input required by SAMATS. The green text represent intermediate produces from the tools. The gold text box represents the final textured building model

1.3 Contributions

The goal of this thesis is to demonstrate modelling techniques that reduce the amount of user interaction required to create complex 3D building models using georeferenced terrestrial imagery. The techniques proposed in this thesis towards achieving this overall goal are based on the following ideas:

- **Correspondence Determination** – By using georeferenced imagery and the automated correspondence determination algorithm as described in Section 5.1.3, it is possible to eliminate the manual correspondence step required by the majority of contemporary modelling systems.

- **Vertex Merging** – By identifying connections between edges in the edge highlighting phase as described in Section 5.1.5, the recovered
floating edges can be connected together to create the edge outline model of the building.

- **Surface Determination** – By treating the edge outline model like a graph and finding cycles in the graph that are co-planar as described in Section 5.2.1, the surfaces of the building model can be determined.

- **Surface Aligning** – By recursively aligning the normals of adjacent surfaces as described in Section 5.2.2, the surface normals of the entire model become coherent.

- **Primary and Secondary Edges** – By determining the locations of strong candidate edges while ignoring weak unessential edges as described in Section 4.1, a more accurate building model is created.

- **Triangle Texture Filling** – By leaving an empty border around each triangle texture and setting empty texels to the colour of the closest non-empty texel inside the triangle as described in Section 6.2.3, the triangle’s texture does not darken at high mipmap levels.

- **Texture Packing** – By packing all of the triangle textures into a single texture map as described in Section 6.3, the building model can be rendered more efficiently.

### 1.4 Success Criteria

In order to gauge to what extent SAMATS has succeeded in achieving its goal, SAMATS will be evaluated against two criteria:

1) In what ways has SAMATS been able to streamline the contemporary building modelling workflow?

2) To what extent has SAMATS retained the flexibility to model a wide variety of building types?

These questions will be answered in Section 8.4 once both the system and experimental results have been presented.

### 1.5 Thesis Outline

The remainder of this thesis is divided into the following sections:
Chapter 2 investigates the state of the art in the various disciplines that contribute to a photogrammetry-based modelling system.

Chapter 3 describes how the synthetic image sets used to develop the techniques demonstrated by SAMATS are created.

Chapter 4 describes the first of three tools that make up the core of SAMATS, the Edge Highlighting Tool. This tool is used to highlight edges of interest in the images of the image set.

Chapter 5 describes the Model Creation Tool, which is a fully automated tool that uses the output from the Edge Highlighting Tool to generate the geometric model of the building.

Chapter 6 describes the Texture Extraction Tool, which extracts the façade information from the images of the image set, blends contributing images together, packs the façade data into a single texture and assigns the texture to the model.

Chapter 7 investigates the use of SAMATS to model a variety of building structures, as well as examining SAMATS’ tolerance to error in the exterior orientation parameters.

Chapter 8 presents conclusions and indicates future work to expand the scope of SAMATS.

1.6 Summary
This chapter gave a brief overview of the building modelling field, followed by a brief description of the three broad categories of modelling systems, touching on the advantages and disadvantages of each. Some of the lower level requirements of a terrestrial-based modelling system were then presented demonstrating the range of choices available when designing such a system.
1 Introduction

Next, the initial motivation for pursuing this research was presented followed by an overview of the work that will be presented in this thesis.

Although some initial justification for undertaking this research was presented in this chapter, the next chapter expands on this examining the relevant fields that contribute to a photogrammetry-based modelling system. The research from this thesis is also placed in context to the previous work from these fields.
2 BACKGROUND AND RELATED RESEARCH

Creating photorealistic 3D building models using photogrammetry-based techniques relies on the combination of techniques developed in many distinct fields. Nine of the most significant topics that directly relate to the building modelling problem will be examined in this chapter:

- **Georeferenced Imagery** – Discusses techniques to facilitate the acquisition of imagery of known position and orientation.

- **Camera Calibration** – Describes how the mapping between image pixels and their corresponding 3D ray can be determined.

- **Structure from Motion** – Investigates the original Structure from Motion problem as well as more recent advances.

- **Stereo Imagery** – Looks at stereo-based techniques, highlighting their strengths and weaknesses.

- **Range Scanning** – Presents some of the cutting edge work from this field, both building centric and non-building centric.

- **Image-Based Rendering and Modelling** – Describes the main image-based techniques, and why they were not pursued in this research.

- **Semi-Automated Modelling Systems** – Investigates the advantages and limitations of user operated modelling systems, as well as identifying the requirement for human interaction.

- **Automated Modelling Systems** – Investigates techniques that automate steps in the modelling process that have traditionally been carried out manually, as well as the limitations they impose.

- **Texturing** – Discusses a number of approaches to adding façade texture information to models.

Before looking at these individual fields, a taxonomy of modelling systems is presented based on the level of user interaction required for their use.
2. Background and Related Research

2.1 Modelling System Taxonomy

Based on the level of user interaction required, modelling systems can be divided into three broad categories (Figure 2.1):

![Diagram of Modelling System Taxonomy]

In Table 2.1 above, tasks are divided into the following three categories:

1) **Image Acquisition** – The image acquisition task involves the amount of effort required of the user to capture suitable imagery of a particular building for the particular modelling system. Note that it was assumed that aerial imagery is readily available, as it is in North America.

2) **Modelling/Registration** – This task involves the amount of effort required of the user to both register the images so that they are in some relative or absolute reference frame, as well as the length of time required to highlight features in the images.

3) **Texturing** – The final task evaluates the amount of effort required of the user to texture the final building model, assuming the model will be textured.
2 Background and Related Research

Note that each task is marked out of five with a zero signifying no effort and a five signifying maximum effort. A dash (-) is used to signify that the particular task is not supported by the system.

<table>
<thead>
<tr>
<th>System Task</th>
<th>Noronha</th>
<th>Lee</th>
<th>Zlata.</th>
<th>Li</th>
<th>Zhao</th>
<th>MIT</th>
<th>Policarpo</th>
<th>Façade</th>
<th>SketchUp</th>
<th>Max</th>
<th>Maya</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquisition</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Modelling</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Texturing</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2.1 Shows the amount of user interaction required to model a building for each of the 11 modelling systems shown in Figure 2.1

Similarly to how the tasks were divided up in Table 2.1, tasks are divided into two categories in Table 2.2, although this time the systems flexibility at carrying out each task is assessed. Note that each task is marked out of five with a zero signifying no effort and a five signifying maximum effort. A dash (-) is used to signify that the particular task is not supported by the system.

<table>
<thead>
<tr>
<th>System Task</th>
<th>Noronha</th>
<th>Lee</th>
<th>Zlata.</th>
<th>Li</th>
<th>Zhao</th>
<th>MIT</th>
<th>Policarpo</th>
<th>Façade</th>
<th>SketchUp</th>
<th>Max</th>
<th>Maya</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelling</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Texturing</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2.2 Shows the flexibility of the 11 modelling systems shown in Figure 2.1

- **Manual Systems** – These systems give the user complete flexibility to model almost any structure imaginable. Examples of such systems include:
  
  - 3D Studio Max by Autodesk [3].
  - Maya by Alias [2].
  - SketchUp by @Last Software [1].

Figure 2.2 shows a detailed scene created and rendered in SketchUp. Some of the most realistic city models seen in computer graphics
2 Background and Related Research

have been created using manual systems, such as the London city model created by Sony London Studio for their upcoming PlayStation 3 game, The Getaway [40] (Figure 2.3). Although these systems are very flexible, both expert knowledge and a large amount of user interaction are required to model anything beyond the most rudimentary of models.

Figure 2.2 A scene created and rendered inside SketchUp [41]

Figure 2.3 A screenshot from the upcoming PlayStation 3 game, The Getaway [40]
2 Background and Related Research

- **Semi-Automated Systems** – These systems generally get the user to highlight features across the image set and then make correspondences between those features. Debevec presents such a semi-automated system called Façade [9] [10] (Figure 2.4). While generally not as flexible as manual systems (Façade is unable to model domes or columns) the amount of user interaction and the level of expertise required to use such systems is reduced.

![Figure 2.4 A screenshot of the Campanile, Berkeley's clock tower, being modelled in Façade [9] [10]](image)

- **Automated Systems** – These systems automate much or even all of the modelling process. The MIT City Scanning project [7] [45] [46] makes use of a robotic rig [8] (Section 2.2) to capture *spherical mosaics* (i.e. 360° images) around the MIT campus (Figure 2.5 and Figure 2.6), which are then used to create models of the buildings in the area. Although the system requires very little user interaction (the robotic rig needs to be supervised and each mosaic requires a single correspondence identified by the user), the system is limited to modelling simple shaped buildings by identifying the rooflines of buildings and extruding downwards to ground level. A significant limitation of this simplistic approach is that it is impossible to model geometric façade features (e.g. balconies or porches). As will be seen in Section 2.9, most automated modelling systems impose restrictions and constraints on what can be modelled.
2 Background and Related Research

An ideal modelling solution would provide the flexibility of manual systems, or even semi-automated systems, yet have the minimal user interaction found in automated systems.

Figure 2.5  A screenshot of three building models produced in the MIT City Scanning Project [7] [45] [46]. The red spheres indicate the spherical mosaics that capture the scene

Figure 2.6  A spherical mosaic from the MIT City Scanning project [7] [45] [46]
2.2 Georeferenced Imagery

Obtaining georeferenced terrestrial imagery is not a rapid process at present. There have been a number of attempts to accelerate the process but these systems generally require custom built rigs.

De Couto [8] used an automated rig called Argus to acquire georeferenced imagery using a navigation system and a digital camera. Argus was capable of determining the pose of each image within 1-2 degrees of attitude and 2-7 meters in position (Figure 2.7).
2 Background and Related Research

Coorg [7] and Teller [45] [46] went on to use Argus to create spherical mosaics around the MIT campus, but required an additional registration step carried out by the user to identify correspondences in each mosaic in order to refine the exterior orientation parameters of the rig to obtain the accuracy required to reconstruct the building models. After the registration process the direction of a projected ray would be accurate to within 0.05 degrees, resulting in a 2.5 centimetre error in the predicted location of a 3D point 25 meters from the camera. Requiring an additional registration step is undesirable, but considering only a single registration point was required for each spherical mosaic and each mosaic is composed of many individual images, the cost of registration is reduced considerably. The accuracy obtained was demonstrated to be sufficient for recovering the basic shape of the buildings in the area (Section 2.9).

Zlatanova and van den Heuvel [55] make use of GPS and an inertial tracker to obtain rough estimates of the camera’s position. Coupled with a video camera which tracks line features in real-time and compares these features against an a priori reconstructed 3D model, less than one metre positional accuracy was achieved. The need for a rough model a priori to image capture is a major limitation of the approach, since a rough model is likely to be one of the desired products from a modelling system.
Cheaper solutions are becoming available, such as GPS cameras like the Ricoh Pro G3 GPS Camera [38] (Figure 2.8) and digital orientation devices, such as the MTi miniature gyro-enhanced Attitude and Heading Reference System [50] (Figure 2.9). Although the accuracy of these devices is not sufficient for building modelling at present (the Ricoh Pro G3 GPS Camera only provides three meters absolute accuracy), it is likely to improve over time making such systems viable replacements to the higher cost rigs used to date.

In this thesis, it was clear that obtaining sufficiently accurate georeferenced imagery posed a problem. To avoid committing a large amount of resources into tackling this problem, the decision was taken to use synthetic imagery instead.

Several of the systems reviewed have carried out experiments with both synthetic imagery and photographic imagery test sets [11] [44] [47]. In no case was there mention of noticeable differences in the results between the two types of test sets. For this reason, it was felt that even though it would have
2 Background and Related Research

been desirable to demonstrate the techniques from this thesis using photographic imagery, demonstrating the techniques using only synthetic imagery should not detract from their suitability with photographic imagery.

In order to gauge the georeferencing accuracy requirements of an image set, a series of tests that introduce error into the exterior orientation parameters were carried out (Section 7.7). This gives an estimate of what level of georeferencing accuracy is required of an image set for it to be suitable for use with SAMATS.

2.3 Camera Calibration

Camera calibration is the process of determining the internal geometry and optical characteristics (interior orientation parameters) of a camera. A camera is said to be calibrated if the mapping between pixels in the image and the direction of rays from the camera’s focal point are known.

Although the decision was made to use synthetic imagery, and synthetic imagery generally does not need to be calibrated, since an idealised pinhole camera can be used to render the imagery, this section looks at how a camera can be calibrated in order to remove any distortion from the image and to resolve the mapping between image pixels and their corresponding ray directions.

Faugeras [16] presents a thorough explanation of the mathematics describing a pinhole camera as well as both a linear and a non-linear approach to determining the interior and/or exterior orientation parameters of a camera. However, neither the mathematics nor the methods for resolving the interior/exterior orientation parameters are presented in this thesis.
Debevec [9] presents a practical approach to camera calibration. Prior to recovering the interior orientation parameters of the camera using a calibration rig (Figure 2.10 shows the type of calibration rig used by Debevec), the radial lens distortion of the camera is removed. Radial lens distortion makes straight lines in world-space appear curved in image-space. If the camera is calibrated a number of times from differing viewpoints, one finds that the recovered interior orientation parameters are considerably more consistent if the radial distortion is removed prior to calibration.

Since synthetic imagery is used in our case, no camera calibration is required. However, it is unlikely that this would present much difficulty, since the problem of camera calibration is largely resolved with many established approaches.

2.4 Structure from Motion

Given two images of a scene, it is possible to determine the position and orientation of the cameras used to take each image if a sufficient number of correspondences are identified in both images.
2 Background and Related Research

Faugeras [16] presents a thorough overview of the mathematics describing the structure from motion approach, presenting a detailed analysis of Kruppa’s 1913 proof, which states that five point correspondences are needed across two images to determine the position and orientation of the two cameras as well as the positions of the five points relative to some relative world reference frame. Although it is theoretically possible, the five point solution is very sensitive to noise. In practice, eight points are required to determine a robust solution.

Ullman [47] was one of the first to investigate the structure from motion problem, reasoning that any set of elements undergoing a 2D transformation which has a unique interpretation as a rigid body moving in space, should be interpreted as such a body in motion.

Dellaert et al. [11] demonstrate that it is possible to solve for the structure from motion problem without the need to carry out correspondences manually. Point features are highlighted manually without making correspondences between them. An initial guess is made for the correspondences. It is not feasible to attempt all combinations of possible correspondences, so a Monte Carlo Expectation Maximisation algorithm is used to guide the refinement of the correspondences. This is shown to generally converge after about 100 iterations. The main limitation of this work is that all point features have to be visible in every image, which means the presented approach is not able to model an object completely.
Taylor and Kriegman [44] extend the feature correspondence from points to lines (Figure 2.11). This work recovers some good results, but like the previous approaches suffers from two problems:

- Correspondences between features must still be carried out manually.
- The recovered features are independent of each other and are not suitable for representing solid objects.

### 2.5 Stereo Imagery

Stereo rigs extract 3D positional information from scenes by taking two images of the scene from known differing locations, matching corresponding features in the two images, and then triangulating the features’ positions relative to the rig (Figure 2.12).
The distance between the two cameras, known as the baseline distance, plays an important role in determining how accurately the camera-space location of features can be measured. The stability of the correspondence determination algorithm is also very dependent on the baseline distance.

The accuracy of the feature measurements improves with larger baseline distances, since a larger baseline distance results in larger disparity between the feature measurements in the two images. Slight errors in the image measurements are amplified at smaller baseline distances.

However, increasing the baseline distance makes it more difficult to automatically determine the correspondences, since features in the images will appear more and more different at increased baseline distances due to foreshortening effects and possible occlusions.

Faugeras’ book [16] presents a number of constraints that can be used to improve the accuracy of the feature measurements including:
2 Background and Related Research

- **The Disparity Constraint** – This constraint states that surfaces must be locally smooth.

- **The Geometric Constraint** – This constraint assumes that surfaces are planar almost everywhere except across discontinuities.

Petsa and Karras [36] make use of parallel line and surface perpendicular constraints to improve the accuracy of line feature measurements for a building. This has the disadvantage of assuming that the building being modelled is rectilinear in shape.

Debevec [9] [11] presents a novel adaptation of the standard stereo algorithm called model-based stereo, which adds fine detail to the geometric model produced using the Façade semi-automated modelling system. Stereo algorithms are generally unable to resolve the correspondences across two images that are taken relatively far apart from each other since foreshortening and occlusion effects start occurring. Debevec demonstrates that images that appear quite different generally appear quite similar when projected onto an approximation of the scene geometry. Therefore by projecting the *offset image* onto a rough model and then reprojecting onto the *key image* plane, a warped offset image is created (Figure 2.13). Then correspondences can be identified along the *epipolar lines* of the key and warped offset images. The *depth map* created can then be used to add depth to the rough model using standard image-based rendering algorithms (Section 2.7).
2 Background and Related Research

![Figure 2.13 Debevec's model-based stereo [9] [11]. The two images at the top are the original key and offset images. If the offset image is projected onto a flat geometric plane representing the building front face, it appears as the warped offset image from the key camera’s point of view. Correspondences can then be determined between the key and warped offset to create a disparity depth map, which can be used to increase the geometric complexity of the rendered model.]

Although features can be reasonably measured using the above techniques, these techniques have two major shortcomings:

- They are generally restricted to two images when determining the location of features. It would be desirable to use all available information from all images when determining the location of a feature.

- Stereo rigs have a fixed baseline distance. It would be desirable to capture individual images from arbitrary locations to increase the accuracy of the features’ recovered locations.

These problems exist because the correspondences are carried out in the image-space of the photographs. If the correspondence determination step was carried out in world-space, many of these problems may be avoided.
2.6 Range Scanning

The use of range scanners, typically laser scanners, to obtain range imagery has been an active area of research over the last decade. Laser scans have produced highly accurate geometric models, especially when used to capture human sized objects such as statues.

The Digital Michelangelo Project [27] is a great example of what is achievable using laser scanning (Figure 2.14). The number of points measured, and the accuracy to which they are determined gives this technique an unparalleled ability to capture micro details.

Laser scanning has also been used in the modelling of urban areas [28] [53] and areas of architectural heritage [33].

Two recent approaches to automating the modelling of urban scenes include the work of Zhao et al. [53]. This work was able to accurately model a street scene by attaching a laser scanning unit to a road vehicle. However, the
2 Background and Related Research

models produced were \textit{voxel}-based so were not suitable for integration with standard geometry-based rendering engines.

On the other extreme of the spectrum, Li et al. [28] were able to produce geometric models of buildings. However, the models produced where extremely rudimentary, simply tracing out the footprint of the building and extruding upward by the average height.

Even though laser scans can measure points very accurately, they do have several disadvantages including:

- Laser scanning equipment is extremely expensive when compared to a mid to high range camera.
- Photographic data is still required in order to add façades to the models produced.
- A massive amount of data is produced which needs to be further processed to a) convert the data into a mesh representation and b) significantly reduce the geometric complexity of the model so that it can be viewed in real-time.

2.7 Image-Based Rendering and Modelling

Image-based rendering uses two or more images of a scene to create novel views of that scene by combining the images using suitable warping and blending operations. Often accurate correspondences are required between images in order to determine depth in the scene or a rough 3D model is used to improve the quality of the renderings, this related field is known as image-based modelling.
Debevec uses a technique called view-dependent texture mapping [9] [11], which projects different textures onto a model of the building depending on the user’s viewpoint (Figure 2.15). This technique can produce very realistic results, since each texture captures the micro geometry of the building correctly when viewing the model from a similar viewpoint. However, the
2 Background and Related Research

memory requirements of this technique can be much larger than using a single texture per surface.

Szeliski [42] presents an overview of the latest techniques (as of 2001) noting that although many new techniques have been developed, all are limited in their own way. One technique, the volumetric representation, has many degrees of freedom that make it difficult to determine the best reconstruction. While using the layered representation, it can be difficult to determine the correct number of planes that should be used, as well as determining the correct pixel assignments.

Snavely et al. [39] present the Photosynth system that allows the browsing and organising of large photo collections of popular sites using image-based rendering, image-based modelling, and user-interface techniques. The photo collections are browsed using a 3D interface, using morphing techniques to provide smooth transitions between photos. Although the system has the ability to organise hundreds of photographs and allow the user to navigate them in an intuitive manner, the interface is very stylised showing the images as floating billboards, which lessens the sense of immersion.

Uyttendaele et al. [48] present an approach to create virtual tours of real world locations along designated paths allowing the user to rotate the view 360 degrees at any point on the path. Although visually very accurate, the user is still locked to the recorded paths and is unable to choose arbitrary viewing locations.

Zitnick et al. [54] present a system that allows the user to seamlessly change their viewpoint while a video is playing. The user is also able to freeze the video while changing the viewpoint. This system makes use of automated high-quality viewpoint interpolation using video from relatively few high-resolution cameras.
2 Background and Related Research

When no rough model is provided, the above techniques are only able to produce convincing reconstructions from viewpoints close to where the original images were taken. AlthoughDebevec’s view-dependent texture mapping [9] [11] is much more robust, there is still the increased memory footprint required of image-based rendering techniques.

2.8 Semi-Automated Modelling Systems

As mentioned in Section 2.1, semi-automated modelling systems attempt to speed up the modelling process, while still making it possible to model a wide variety of building structure types.

Debevec [9] [10] presents Façade, a semi-automated modelling system that requires the user to create block primitives, highlight edges in the images, and carry out correspondences between the highlighted edges in the images and the edges of the block primitives. The user is also able to add constraints to the block primitives, such as block A must be centred on block B, or set a symmetry constraint so that features that may not be visible can be modelled. These constraints help to simplify the modelling process.

Policarpo et al. [37] present a technique for modelling an urban scene using a single image (Figure 2.16). The system works by first defining a ground plane. All objects visible in the image are then constructed by placing user defined box primitives on top of each other. When the scene is viewed from a perspective similar to that of the original image, the scene appears quite convincing. Clearly, as the view is moved further from the original viewpoint, artefacts begin to appear in the rendered reconstruction.
An interesting approach is not to reconstruct the scene using a semi-automated approach, but to supervise an automated modelling system (Section 2.9). Lee et al. [23] use an automated modelling system, which recovers rectilinear shapes, but adds a user supervised step at the end where the user can connect these shapes together to form more complex buildings, using only a few mouse clicks.

Although many of these systems have been used to create accurate building models with reduced user interaction when compared to the manual systems (Section 2.1), all suffer from the same major problem. The tasks that are performed by the user are difficult to automate. Therefore, additional information needs to be provided to the system, so that it can automate these difficult tasks.
2.9 Automated Modelling Systems

Automated modelling systems have the potential to model large areas in a short length of time, since human interaction is minimised or even eliminated completely. This is due to the fact that human interaction is generally the main time limiting factor in most modelling systems. Unfortunately, all systems examined suffer from some set of limitations.

Using spherical mosaics captured using the Argus robotic rig [8], Coorg [7] and Teller [45] [46] use a global registration step to refine the position and orientation of each mosaic. The registration step is the only manual step carried out by the user. The vertical façades of buildings are recovered as follows:

1) The azimuths of the building faces are determined in each mosaic.

2) Using a space-sweep algorithm, the offset of each façade is determined.

Although this approach works well for some datasets it does suffer from several limitations:

- Firstly, spurious façades may be generated due to interactions between different façades.

- Secondly, façades can be missed if viewed by too few nodes.

- Thirdly, (and most significantly) is the assumption that the roofline edges and baseline edges are horizontal. Buildings situated on a hill or buildings with sloped roofs would not be modelled correctly.

Petsa and Karras [36] improve the feature measurements recovered from a stereo rig by setting constraints (Figure 2.17). For example, they state that lines should be either perpendicular or parallel to all other lines. Although good results were obtained for each stereo-pair, there was still no way of combining results from pairs of images taken from arbitrary locations and the
resulting model consisted of floating edges, which are not suitable for solid object rendering.

Zlatanova and van den Heuvel [55] look at the problem of extracting line features from close range imagery using a rig fitted with GPS and a video camera that can track edge features and compare these features against a rough model of the building created a priori. This automated extraction system was able to recover some features, but they found that highly accurate exterior orientation measurements were required, as well as a reasonably accurate rough model, in order to resolve the correspondences accurately.

Chou and Teller [6] discuss an approach to determine edge correspondences using georeferenced imagery by treating new possible correspondences as hypothetical. They only promote correspondences to confirmed when sufficient supporting evidence is discovered. This system uses a state
2 Background and Related Research

machine to track matching hypotheses in various states of certainty. The state machine evolves its state in response to new evidence. However, no implementation details or experimental results were presented.

Much research has been carried out in the areas of automated roof modelling and complete building modelling using aerial imagery. Noronha and Nevatia [34] give an example of building modelling using aerial imagery, while Ye and Lee [52] give an example of roof modelling using aerial imagery. Although these approaches have been used to accurately model urban buildings and urban building roofs respectively, the techniques used are often not applicable when using terrestrial imagery, since aerial modelling can make use of shadow analysis as well as stereo techniques which are less effective when using terrestrial imagery.

If automated systems were able to use some of the techniques found in semi-automated systems, adopting the flexibility of such techniques in the process, a more streamlined system would result.

2.10 Texturing

Extracting the façade information into textures and applying them to the reconstructed model is a vital step to increasing the realism of the reconstructed model.

Much work has been done investigating the integration of building façade data taken using terrestrial imagery with building models created using aerial imagery. Lee et al. [24] [25] [26] have looked at several techniques to achieve this. Kada [21] looks at a GPU (Graphics Processing Unit) accelerated technique for achieving the same results.

Although the results of the integration can be quite accurate, two separate systems are required to model a single building accurately. The aerial system is needed to create the geometric model, while the façade integration system is needed to create accurate façades.
2 Background and Related Research

Debevec’s view-dependent texture mapping [9] [11] chooses visual fidelity over efficiency, by blending textures that were taken from viewpoints close to the viewer’s current location. This technique is able to capture global illumination and local occlusion effects, at the cost of using a specialised texturing technique and a larger texture memory footprint.

Coorg’s median texture estimation technique [7] chooses to take advantage of established texturing techniques, whereby each surface is assigned a single texture (Figure 2.18). In order to take advantage of the façade information from each contributing image, the images are blended together in three steps:

1) A statistical median technique is applied to each contributing texture to balance the illumination across the images.

2) The texture is sharpened by iteratively refining the pose of each image.

3) An occlusion map is used to remove any occlusions from the images.

Figure 2.18 Coorg’s median estimation technique [7] used to create the final texture
2 Background and Related Research

Models that are intended for use in city models need to have as low a rendering complexity as possible. Coorg’s techniques are good step towards achieving this goal.

2.11 Discussion

This chapter gave a taxonomy of modelling systems based on the level of interaction required of the user, as well as looking at nine of the more significant topics that contribute to the building modelling problem.

Section 2.1 – Modelling System Taxonomy, highlighted the strengths and weaknesses of the three types of modelling system. Automated systems require little to no user interaction but are generally quite inflexible. Semi-automated systems are generally quite flexible but require a moderate level of user interaction. If the user-performed tasks of semi-automated systems could be automated, the result would be a flexible automated system.

Section 2.2 – Georeferenced Imagery, looked at automated approaches to recovering georeferenced imagery. This problem does not have a straightforward solution at present. The most successful approaches have used custom built rigs, which are not readily available and are expensive to build in terms of both time and money. Cheaper, more compact solutions are beginning to appear but do not have the required accuracy at present. As a result of these findings, the decision was made to use synthetic imagery since such imagery is relatively simple to create and would not detract from the modelling techniques demonstrated in this thesis.

Section 2.3 – Camera Calibration, discusses the camera calibration problem that needs to be addressed when using photographic imagery. Although this is not an issue with this research (since synthetic imagery is being used) even if photographic imagery was to be used it is unlikely that camera calibration would represent much of an obstacle since a number of established approaches exist.
2 Background and Related Research

Section 2.4 – Structure from Motion, describes how the location of a set of 5 points in two images could be determined using Kruppa’s 1913 proof. The work of most interest from this section is that of Taylor and Kriegman [44], which demonstrated structure from motion using line primitives. Two of the problems with their approach are that all correspondences needed to be carried out manually and the output of the system is a set of floating edges. Some type of automated correspondence determination and vertex merging is required in order to make such a system suitable for automated building modelling.

Section 2.5 – Stereo Imagery, discusses how terrestrial stereo systems work as well as discussing some of the problems associated with the approach. Since the correspondences are carried out in the image-space of the photographs, keeping the baseline distance small has the negative effect of reducing the measurement accuracy. If the correspondence determination step was carried out in world-space, many of these problems may be avoided. However, this would assume that the exterior orientation parameters of the camera are already known.

Section 2.6 – Range Scanning, looks at how laser scanners can be used to capture high-resolution geometric models of statues and buildings. The two main disadvantages of this approach are the cost of the equipment and the mass of data produced by the scans. Consolidating the data down to a usable level is not a trivial problem, and generally requires user supervision. Although this approach is increasing in popularity, a purely photogrammetry-based approach was pursued in this research.

Section 2.7 – Image-Based Rendering and Modelling, discusses techniques for rendering scenes directly using the original imagery. These techniques suffer from two main problems:
2 Background and Related Research

- Firstly, when no rough model is provided, the above techniques are only able to produce convincing reconstructions from viewpoints close to where the original images were taken.

- Secondly, image-based rendering techniques generally have a much higher memory footprint requirement than geometry-based techniques.

For these reasons, the specialised image based techniques were avoided.

Section 2.8 – Semi-Automated Modelling Systems, looked at a number of modelling systems that require a moderate level of user interaction. Although many semi-automated systems are capable of modelling relatively complex building structures with reduced user interaction, the user interaction that is required is generally difficult to automate. By providing additional information to the system it may be possible to automate these user-performed tasks, transforming the flexible semi-automated system into a flexible automated one.

Section 2.9 – Automated Modelling Systems, highlights some of the approaches that have been taken to reduce user interaction, although all the systems researched suffer from reduced flexibility as a result. In a similar manner to that in which flexible semi-automated systems can become flexible automated systems by automating the user-performed tasks, if the automated modelling systems were enhanced to handle arbitrarily shaped structures, inflexible automated systems could become flexible automated systems.

Section 2.10 – Texturing, discusses a number of techniques that have been used to apply façade information to building models. There are many established techniques for doing this, but if the model is intended for use in a city model, as is the case with this research, additional steps need to be considered in order to reduce the rendering complexity as much as possible. It may be possible to reduce the rendering complexity further by using texture atlases.
2.12 Conclusions

From the issues discussed in this chapter a number of RQs (Research Questions) are raised:

**RQ1** – Can the techniques used in both automated and semi-automated systems be merged to create a flexible automated system?

This is the main question being asked in this research. It is raised as a result of the observations from Sections 2.1, 2.8, and 2.9. The answer to this question is touched on in Section 5.4, with the full discussion postponed to the end of this thesis (Chapter 8) after SAMATS has been fully presented, tested and analysed.

**RQ2** – Can the correspondences between line features be determined automatically?

This is the first of two questions that need to be answered if line primitives are to be used in a fully automated system. This question is raised as a result of the observations from Sections 2.4 and 2.5. One approach to solving this question is presented in Section 5.1.3.

**RQ3** – Can the higher level topology of the model be determined from floating edges?

This is the second of two questions that needs to be answered if line primitives are to be used in a fully automated system. This question is raised as a result of the observations from Sections 2.4 and 2.5, relating to the work of Taylor and Kriegman [44] and Petsa and Karras [36] in particular. Determining the models higher order topology, such as the adjacency between the edges and the surfaces, must be carried out in order to render the model as a solid object. This is examined in Section 5.2.
2 Background and Related Research

**RQ4** – Is there a way of determining the location of edges more accurately by only considering strong candidate edges?

This question is raised as a result of the trade off that exists between measurement accuracy and correspondence determination stability when using stereo systems. When using line primitives instead of point primitives, there is the additional consideration that the direction of displacement also factors in the measurement accuracy. As will be discussed in Section 4.1, prudent selection of contributing edges can alleviate this problem.

**RQ5** – Can textures be consolidated to reduce rendering complexity?

This question is raised as a result of Coorg’s median texture estimation technique [7]. Although this approach favours rendering speed over rendering fidelity, one must consider whether the façade information can be consolidated to reduce the rendering complexity further. One technique to achieve this is presented in Section 6.3.

The next chapter discusses how the synthetic image sets used to test SAMATS are created, as well as comparing and contrasting photographic imagery with synthetic imagery.
3 SYNTHETIC IMAGE GENERATION

As mentioned in Section 2.2, the acquisition of georeferenced terrestrial imagery is currently not a rapid process. Several approaches for automating this process have been developed. However, all approaches involve the use of specialised equipment. Since the aim of this research is to demonstrate techniques which reduce the amount of user interaction required to construct complex 3D building models using georeferenced terrestrial imagery (and not techniques for acquire such imagery) the decision to use synthetic imagery was taken so that the focus of the research would remain on the original aim and not be split across two related but separate problems.

This chapter describes how the synthetic scenes are constructed and how image sets of these scenes are captured using the Scene Capture Tool. The chapter concludes by looking at the main differences between photographic imagery and synthetic imagery, and determines if these differences would have a major effect on the performance of SAMATS when using photographic imagery.

3.1 Scene Creation

The test scenes are created using 3D Studio Max. Each scene consists of a single textured building as well as a textured ground plane (Figure 3.1).

Once the scenes are created they are ready to be exported for use in the Scene Capture Tool. Since SAMATS uses Microsoft’s DirectX API for its visualisation, all models are stored in Microsoft’s native eXtension file format (these files have the .x file extension), since this format allows for easy importing and exporting between the various tools.
3 Synthetic Image Generation

![Figure 3.1 A screenshot of a textured building on a textured ground plane inside 3dsMax](image)

3.2 Scene Capture Tool

The Scene Capture Tool is a support tool that is used to generate rectified georeferenced image sets. However, this tool does not form part of the main SAMATS workflow.

Once the Scene Capture Tool imports a scene, the user is allowed to navigate around the scene in a similar manner to a first-person shooter game such as Doom3 [21] or Half-Life2 [50]. The user moves around the scene with a virtual camera that can be used to capture images of the scene. Along with each image being saved to disk, an entry is made in a properties file that stores the interior and exterior orientation parameters of the camera used to take each image.

Once the desired set of images has been taken the image set is ready for use with SAMATS.
3 Synthetic Image Generation

3.3 Synthetic Imagery Vs. Photographic Imagery

With regards to SAMATS, synthetic imagery and photographic imagery can be compared in two basic categories: (a) interior/exterior orientation parameter accuracy and (b) image complexity.

3.3.1 Interior and Exterior Orientation Parameter Accuracy

With synthetic imagery, the interior and exterior orientation parameters of the camera are known precisely. User specified interior orientation parameters are used to create the corresponding idealised pinhole camera to render the scene. The exact position and orientation of the camera used to render each image is also trivial to recover.

When using photographic imagery, determining the interior orientation parameters is a relatively straightforward procedure (Section 2.3). However, determining the exterior orientation parameters can be much more difficult.

Figure 3.2 Example image created using the Scene Capture Tool
3 Synthetic Image Generation

Systems that do not use correspondences to triangulate the exterior orientation parameters need to use GPS and/or inertial tracking systems to determine their position and orientation (Section 2.2).

The accuracy at which these parameters are known (especially the exterior orientation parameters) is the main differentiator between synthetic imagery and photographic imagery. Closing the gap between the two will be one of the main challenges in transitioning SAMATS to work with photographic imagery. The Stability Tests discussed in Section 7.7 give an estimate of what accuracy is required of the georeferenced image set.

3.3.2 Image Complexity

The world around us has near infinite complexity. Even a simple house is not so simple when one considers the finer details, such as the receded mortar between each brick, the overlapping pattern of roof tiles, or each bar supporting a balcony railing. These details are clearly visible in close range photographic imagery. There are also the secondary objects in a scene such as trees, bushes, cars, people, etc. All of which are even more complex than the building itself.

Real world photographic imagery has a complexity that is difficult to match using synthetic imagery. When a synthetic model of a scene is constructed, often only the main features are captured geometrically. No attempt is made to capture the geometry of each individual brick or each individual bar supporting a balcony railing. The impression of such details is usually represented with the use of textures. Since photographic imagery captures scenes of near infinite geometric complexity, selecting the appropriate features to highlight can be more difficult than when selecting features from synthetic imagery.

However, even though there is a difference in image complexity between photographic and synthetic imagery, it only concerns the feature highlighting phase and the texture extraction phase of the modelling process. The
3 Syntheti c Image Gener ation

modelling phase makes no use of the image data when constructing the
g eometric model of a building (Figure 1.3). The main focus of this research is
concerned with streamlining the geometric modelling phase, so the disparity
between image complexities was not a major concern.

3.4 Summary

This chapter described how synthetic scenes are created and how image sets
of these scenes are captured using the Scene Capture Tool. The main
differences between photographic imagery and synthetic imagery were also
discussed to show why synthetic imagery was used. The next chapter
presents the first of three core tools that make up SAMATS, the Edge
Highlighting Tool.
4 Edge Highlighting Tool

4 EDGE HIGHLIGHTING TOOL

This chapter describes the Edge Highlighting Tool (Figure 4.1), which is used to highlight edges of interest in the images of the image set. Edge highlighting is the only task carried out by the user when using SAMATS to create photorealistic 3D building models.

![Figure 4.1 A screenshot of the Edge Highlighting Tool](image)

4.1 Highlighting Edges

The user highlights edges by first identifying junction points in the images of the image set (e.g. the corner points of a building). Edges are highlighted by connecting junction points together. There are two types of connecting lines that can be used to highlight an edge: (a) primary lines and (b) secondary lines. To explain the difference between primary and secondary lines, let us imagine first that there is only a single type of line, a primary line.
Figure 4.2 below shows four images of a green skyscraper taken from its four cardinal corners. Each edge of the skyscraper is also highlighted using a red primary line.

The reason the entire model should not be highlighted using just primary lines is because it is difficult to recover some edges given the limitations of the input data. Primary lines are well suited to recovering the position of vertical edges because it is possible to create arbitrarily large angles of intersection between the projected triangle planes of these edges (Figure 4.3). However, for horizontal edges, especially near camera level, it is not possible to create arbitrarily large angles of intersection between the projected triangle planes. This makes it difficult to recover the horizontal edges accurately, since slight inaccuracies in the camera’s interior and/or exterior orientation parameters result in large errors in estimated edge location (Figure 4.4).
Secondary lines work by connecting primary lines together, where the use of a primary line would be prohibitive due to the insufficient intersection angle between the projected triangle planes. Since primary lines will generally be used to recover the vertical edges of buildings, secondary lines should then be used to highlight the horizontal baseline (i.e. the buildings footprint) and the roofline. This indicates to the system that these edges should be connected without trying the same recovery technique used for the primary edges.
Systems that recover the location of point features do not have to consider the perpendicular direction of the baseline displacement of a pair of cameras relative to the point target. It is only the absolute perpendicular displacement that contributes; the direction does not affect the result at all (Figure 4.5).

When recovering the location of line features, the perpendicular direction of the baseline displacement is important since the projected planes of line features can be co-planar (Figure 4.6).
When recovering point features, it does not matter which direction the secondary camera is displaced. The angle between the projected lines will be the same, and there is no possibility of co-planarity.

When placed on the circle the offset camera will be a fixed distance from the key camera and will only be displaced perpendicular to the target point.

Figure 4.5  When the offset camera is placed anywhere on the circle, the cameras will be a fixed distance from each other and the direction of displacement will be is perpendicular to the KeyTarget vector. Every point on the circle will give the same result when recovering the location of a point.

If the offset camera is displaced perpendicular to the orientation of the highlighted edge, the maximum angle of intersection is achieved between the projected triangles (green).

If the offset camera is displaced parallel to the orientation of the highlighted edge, the projected triangles will be co-planar (red).

Figure 4.6  When recovering line features, the perpendicular direction of the displacement must be considered. The accuracy is at a maximum when the displacement is perpendicular to the orientation of the line feature and at a minimum when the displacement is parallel. At the maximum, the accuracy is the same as when recovering point features, but at the minimum the accuracy is zero.
**RQ4** asked if it was possible to obtain a more accurate building model by only considering strong candidate edges. This is exactly what primary and secondary lines achieve. Primary lines highlight strong candidate edges, which have been captured from varying viewpoints that have a large perpendicular displacement component relative to the target line feature, while secondary lines, which do not satisfy this requirement, simply connect primary lines together.

Thus, the correct way to highlight the edges of the green skyscraper (from Figure 4.2) would be to highlight the vertical edges in the same way as before, with primary lines. However, the roofline and baseline of the building should be highlighted using yellow secondary lines (Figure 4.7).

Figure 4.7  The same scene from Figure 4.2. This time red primary lines are used to highlight the vertical edges while yellow secondary lines are used to highlight the horizontal edges.
Figure 4.8 shows a comparison between using just primary lines to recover the location of every edge, with that of using a combination of primary and secondary lines. As can be seen, using primary lines alone results in many unrecovered edges.

4.2 Highlighting Rules

With the difference between primary and secondary lines presented, this section describes how primary and secondary lines can be placed in an image.
4 Edge Highlighting Tool

The basic approach to outlining a building across the entire image set is also discussed.

The endpoints of a primary line can be connected (having one or more primary or secondary lines sharing that endpoint) or unconnected (having no other lines sharing that endpoint). A secondary line is used to connect primary lines together and must have each of its endpoints connected to at least one primary line.

Primary edges should be used to recover the core structure of the building, while defining as few edges as possible. Then secondary lines should be used to define all remaining edges. A primary edge must be highlighted in at least three images. This is a requirement of the automated correspondence determination algorithm (Section 5.1.3), although it can be advantageous to define a primary edge in more than three images when trying to recover edges that make poor primary edge candidates (Section 4.1). Secondary edges only need to be defined in a single image, but can be defined in any number of images.

4.3 Discussion

The Edge Highlighting Tool was designed to be as easy to use as possible. The tool has only three primitives: junction points, primary lines and secondary lines. This keeps the complexity of the tool down.

The use of primary and secondary lines to maximise the available information, without introducing unnecessary error greatly improved the accuracy of the models produced (Figure 4.8), providing an answer for RQ4 in the process. Having two different types of highlighting primitives does introduce the problem of which edge should use which primitive. Although this is not a major problem for a human user to solve, automating the highlighting phase would rely on heuristics to guide the choice of primitive type.
The main limitation of the Edge Highlighting Tool is its inability to handle partially highlighted edges. Edges must be highlighted in their entirety in order to contribute to the final geometric model correctly. This can make it difficult or even impossible to model certain structures due to spatial constraints, for example in a tight enclosure it may not be possible to capture the full length of a long wall edge. Using a wide-angle lens would reduce the impact of this limitation but it would still be beneficial to support the use of partially highlighted edges. Section 8.5.2 discusses a possible approach to extending SAMATS to handle partially highlighted edges.

It should be noted that no attempt was made to automate the edge highlighting phase. This was due to the disparity in image complexity that exists between synthetic and photographic imagery (Section 3.3.2). If image-processing methods were used to accelerate the edge highlighting process, no definite conclusions could be drawn, since it would not have been demonstrated that such methods would be applicable to photographic imagery. Section 8.5.1 describes two approaches that could accelerate the edge highlighting task: (a) the first approach would aid the user in the edge highlighting task, while (b) the second would attempt to fully automate the process.

4.4 Conclusions

The Edge Highlighting Tool adequately fulfils the core functionality required of it by allowing the user to place line primitives in the images of the image set. But it is in answering RQ4 (Section 2.12) that the strength of the tool is demonstrated. By using primary and secondary lines to maximise strong data while minimising error prone data, complete models can be recovered which would not have been recovered if all features were treated equally.

The next chapter discusses the second of the tools that make up the core of SAMATS, the Model Creation Tool.
5 MODEL CREATION TOOL

After the user has placed the primary and secondary lines, the Model Creation Tool can use this information, along with the interior and exterior orientation parameters for each image, to create the geometric model of the building. The modelling process is **fully automated**, requiring no user interaction. The model is created in two steps:

1) **Edge Determination** – The edge outline model of the building is created by first determining the locations of the highlighted edges.

2) **Surface Structure Determination** – Finally, the surface structure is determined using only the edge outline model.

5.1 Edge Determination

The edge determination process is performed in six steps. The locations of the primary edges are determined by the first five steps:

1) **Line Projection** – Primary lines are transformed into projected 3D triangles.

2) **Triangle Intersection** – The intersections between the projected triangles are determined.

3) **Correspondence Determination** – The correspondences between the primary lines are resolved.

4) **Edge Averaging** – The final primary edge locations are determined.

5) **Vertex Merging** – Floating edges are connected together when appropriate.

The secondary edges are determined in the final step:

6) **Secondary Edge Determination** – The secondary edges that connect primary edges together are determined.

Each of these steps is described in more detail next.
5 Model Creation Tool

5.1.1 Line Projection

The first step in determining the locations of the primary edges is to project the 2D primary lines to form 3D triangles in world-space. The interior and exterior orientation parameters of the camera are used to project the primary lines from the camera’s focal point, at the correct orientation, out to some large distance relative to the dimensions of the scene being modelled. This is performed for every primary line in each image as follows.

A pixel in the image is defined by its x and y components:

\[ p_i = (x_i, y_i) \]  \hspace{1cm} \text{(Equation 5.1)}

Where,

\( p_i \) is the \( i \)th pixel,
\( x_i \) is the pixel position of the \( i \)th pixel in the x-axis direction,
\( y_i \) is the pixel position of the \( i \)th pixel in the y-axis direction.

The 3D ray corresponding to any point in the image is determined by the following equation:

\[ r_i = (x_i - w/2, h - y_i - h/2, f) \]  \hspace{1cm} \text{(Equation 5.2)}

Where,

\( r_i \) is the ray corresponding to \( p_i \),
\( w \) is the width of the image in pixels,
\( h \) is the height of the image in pixels,
\( f \) is the focal length of the image in pixels.

Note that the image origin is in the upper left corner of the image and that the positive y-axis runs down the image (Figure 5.1).
5 Model Creation Tool

A primary line is defined by its two end points, $\mathbf{p}_{\text{start}}$ and $\mathbf{p}_{\text{end}}$. The rays that correspond to these two points, $\mathbf{r}_{\text{start}}$ and $\mathbf{r}_{\text{end}}$, are used to construct the projected triangle of the primary line. The three vertices of the triangle are defined as follows:

$$v_1 = (x_i, y_i, z_i) \quad \text{(Equation 5.3a)}$$

$$v_0 = \text{Origin} \quad \text{(Equation 5.3b)}$$

$$v_1 = \hat{r}_{\text{start}} \times \text{INFINITE\_DISTANCE} \quad \text{(Equation 5.3c)}$$

$$v_2 = \hat{r}_{\text{end}} \times \text{INFINITE\_DISTANCE} \quad \text{(Equation 5.3d)}$$

Note that INFINITE\_DISTANCE is some large distance relative to the scene being modelled. Ideally, the base of the triangle would be projected out to infinity. However, since the triangle intersection algorithm used in the next section requires triangles of finite area, a large relative distance is used instead.
Although the triangle has been created correctly, it must be transformed from camera-space to world-space. The exterior orientation parameters of the camera are used to first rotate each vertex of the triangle so that the triangle is correctly oriented in the world. Once the vertices are in the correct orientation, each vertex is translated to its current world position. The entire transformation is performed by the following equation:

\[ v' = v \times R_{\text{pitch}} \times R_{\text{yaw}} \times R_{\text{roll}} \times T_{\text{cam}} \]  

(Equation 5.4)

Where,

\[
R_{\text{pitch}} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \omega & \sin \omega & 0 \\
0 & -\sin \omega & \cos \omega & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \quad R_{\text{yaw}} = \begin{bmatrix}
\cos \phi & 0 & -\sin \phi & 0 \\
0 & 1 & 0 & 0 \\
\sin \phi & 0 & \cos \phi & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
R_{\text{roll}} = \begin{bmatrix}
\cos \kappa & \sin \kappa & 0 & 0 \\
-\sin \kappa & \cos \kappa & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \quad T_{\text{cam}} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
x & y & z & 1
\end{bmatrix}
\]

Figure 5.2 below shows the primary lines from Figure 4.7 being transformed to projected triangles.

5.1.2 Triangle Intersection

Once the primary lines have been transformed to 3D triangles, the next step is to determine the intersections between all the triangles. Every triangle stores a list of the triangles it intersects along with the line of intersection it makes with that triangle. The interval overlap method first presented by Möller [32] was used to determine the triangle-triangle intersections.
Figure 5.2. The images above show the scene from Figure 4.7, with each of the four images above adding an additional set of projected triangles from the next image of the image set.

Figure 5.3 and Figure 5.4 show the lines of intersection from the simple skyscraper scene in Figure 4.7. One can see that for even a very simple scene there are a large number of invalid intersection lines. The only valid intersection lines are those located at the four vertical wall edges of the skyscraper.
5 Model Creation Tool

Figure 5.3 The lines of intersection between the projected primary line triangles from Figure 5.2

Figure 5.4 Close-up of the lines of intersection from Figure 5.3

60
5.1.3 Correspondence Determination

Generally, each triangle intersects many other triangles even though only a small number of these intersecting triangles highlight the same primary edge as itself. Semi-automated systems resolve this problem by getting the user to carry out manual correspondences in the highlighting step (Ullman [47] and Taylor and Kriegman [44] from Section 2.4 and Debevec [9] from Section 2.8) so lines that highlight the same edge are grouped together. Once these lines are transformed into projected triangles, the only intersections considered are those between members of the same group. Unfortunately, carrying out the correspondences manually can be a very time consuming process.

Stereo-based systems resolve the correspondences in image-space using image feature matching techniques (Section 2.5). However, these image-space correspondence techniques do not work when images are taken from arbitrary locations since the images of a particular building feature can appear very different when viewed from widely spaced locations due to foreshortening and occlusion effects.

RQ2 asked if it is possible to resolve the correspondences automatically. While stereo-based systems resolve the correspondences in image-space, SAMATS resolves the correspondences in world-space by analysing the lines of intersection between the projected primary lines. Note that it is the georeferencing information from the images that make it possible to defer the correspondence determination between features from their image-space representation (i.e. primary lines) to their world-space representation (i.e. projected triangles). SAMATS determines the correspondences automatically in three steps:
5 Model Creation Tool

1) **Intersection Rating** – Each triangle rates each of the triangles it intersects.

2) **Triangle Grouping** – The intersection ratings are used to group triangles together.

3) **Group Merging** – Sometimes two or more groups can be created that represent the same primary edge. These groups must be merged together.

These steps are discussed further next.

### 5.1.3.1 Intersection Rating

To determine which of the intersections are valid (i.e. where both triangles highlight the same primary edge) every triangle needs to rate each of the triangles it intersects. The triangles can then use these ratings to determine which of the intersecting triangles represent the same primary edge as itself.

![Intersection Rating Diagram](image)

**Figure 5.5** Using the ratio of the coverage over the extended line across both triangles as the only criteria for rating each intersection does not provide sufficient information to differentiate between valid and invalid intersections

An initial approach to rating each intersection might use the coverage of the line of intersection over the extended line across both triangles as the only measure, with greater coverage resulting in a better rating (Figure 5.5). Using
this single attribute does not differentiate valid intersecting triangles from invalid intersecting triangles, since often intersecting triangles that represent different primary edges (i.e. invalid intersections) receive better ratings than those that represent the same primary edge (i.e. valid intersections).

The automated rating process does not rate an intersecting triangle on the quality of the intersection line, but rather on the similarity of the intersection line with other intersection lines.

Figure 5.6  A 2D example of the intersection rating problem. Three cameras are used to locate two points.

Figure 5.6 shows the basis of the intersection rating algorithm in 2D. In the figure there are three cameras, A, B, and C, there are two points being modelled, X and Y, and there are six lines, two from each camera through the points being modelled, \( A_X, A_Y, B_X, B_Y, C_X, \) and \( C_Y \). Each line intersects every other line, even though the only valid intersections are those between lines with matching subscripts. Note that some of the intersections are off image and that we ignore intersections between lines emanating from the same camera. One should note that the invalid intersections are spaced quite randomly apart while the valid intersections have three points of intersection.
5 Model Creation Tool

coincident at one location. The automated rating algorithm uses this principle of valid intersections being similar to each other when calculating the rating for each intersecting triangle.

This is why there must be at least three primary lines used to highlight each primary edge. If only two primary lines were used to highlight each primary edge there would only be a single valid intersection line for each primary edge, so comparing intersection lines would be pointless. When using three primary lines to highlight each primary edge, three intersection lines are produced. By comparing intersection lines, valid correspondences can be determined. The first step is to assign a rating to each intersecting triangle. The algorithm works as follows.

Let \( \mathbf{TS} \) represent the set of all triangles in the scene.

\[
\mathbf{TS} = \{ t_1, t_2, t_3, \ldots, t_{n-1}, t_n \}
\]

Let \( t_i \) represent some triangle in the set \( \mathbf{TS} \).

\[
t_i \in \mathbf{TS}
\]

Let \( \mathbf{T}_i \) represent the set of triangles from \( \mathbf{TS} \) that \( t_i \) intersects and let \( t_j \) represent some triangle in the set \( \mathbf{T}_i \).

\[
t_j \in \mathbf{T}_i
\]

For each triangle \( t_j \) in the set \( \mathbf{TS} \) we need to rate each of the \( t_j \) triangles in the set \( \mathbf{T}_i \). Since we know that there are at least three triangles per primary edge, we know that at least two of the intersecting triangles are valid matches. We call the two valid \( t_j \) triangles \( t_{j1} \) and \( t_{j2} \). Note that there may be more than two valid intersecting triangles, although that fact is not important at this stage.

If \( t_{j1} \) is a valid match with \( t_i \) and \( t_{j2} \) is a valid match with \( t_i \), then \( t_{j1} \) and \( t_{j2} \) must be valid matches with each other. This implies that \( t_{j2} \) and \( t_{j1} \) intersect. Therefore, when determining the rating of any \( t_j \), we only need to consider...
5 Model Creation Tool

triangles that are in both $t_i$’s intersecting triangles set and $t_j$’s intersecting triangles set, i.e. $T_i \cap T_j$. Note that only a sub-set of this set will contain valid intersections, since often two valid triangles will intersect the same invalid triangles.

The intersecting triangle $t_j$ can now be given a rating based on the triangles in the set $T_i \cap T_j$.

Let $t_k$ represent some triangle in the set $T_i \cap T_j$:

$$t_k \in T_i \cap T_j$$

Each triangle $t_k$ in this set intersects both $t_i$ and $t_j$. Therefore, we can use the three intersection lines, $l_{ij}$, $l_{ik}$, and $l_{jk}$, to give $t_k$ a rating. Intersection lines are evaluated based on 3 properties:

- **Distance between their Midpoints** – The value returned for the distance between their midpoints is in the range (0…1] and is described by the following equation:

$$\frac{1}{1 + (\text{ScalingFactor} \times \text{Distance}^{\text{PowerFactor}})} \quad (\text{Equation 5.5})$$

Both ScalingFactor and PowerFactor should be greater than zero. The PowerFactor is used to control the shape of the falloff curve. The ScalingFactor is used to shift the curve depending on the units of measurement being used. Figure 5.7 shows the changes in the function’s shape depending on the values for ScalingFactor and PowerFactor used.
Figure 5.7  The values returned by Equation 5.5 for various values of ScalingFactor, PowerFactor, and Distance. Note that Distance runs along the x-axis, with the functions y value representing the functions result.

- **Difference in Orientation** – The value returned for the measure of the two lines relative orientation is calculated using the absolute value of the dot product between the lines’ unit vectors and is in the range [0…1]. For two lines \(A\) and \(B\) the equation is as follows:

\[
\left| \hat{A} \cdot \hat{B} \right| \quad \text{(Equation 5.6)}
\]

- **Difference in Length** – Finally, the value returned for their difference in length is in the range [0…1] and is described by the following equation:

\[
\frac{\max(\|A\|, \|B\|) - |A - B|}{\max(\|A\|, \|B\|)}
\quad \text{(Equation 5.10)}
\]
5 Model Creation Tool

Once all these partial comparisons have been performed, the final rating for the lines is simply the product of the three partial comparisons. The value is in the range \([0 \ldots 1]\). Figure 5.8 illustrates each of these tests.

![Figure 5.8 The three partial comparisons used to determine the similarity between two intersection lines](image)

Every triangle \(t_k\) in the set \(T_i \cap T_j\) is given a rating based on the comparison of the three intersection lines \(l_{ij}, l_{ik},\) and \(l_{jk}\). There are three comparisons that can be made: \(l_{ij}\) with \(l_{ik}\), \(l_{ij}\) with \(l_{jk}\), and \(l_{ik}\) with \(l_{jk}\).

The product of these three tests is used to determine the rating of each triangle \(t_k\) in the set \(T_i \cap T_j\). The product is used in favour of the sum in order to keep the ratings in the range \([0 \ldots 1]\).

Once every \(t_k\) has been given a rating there are three logical options for assigning a rating to \(t_j\):

- Assign \(t_j\) the weighted sum of all the ratings in the set \(T_i \cap T_j\). This has proved unfavourable since this would include triangles that are invalid. If there are a large number of low scoring invalid triangles in a particular \(T_i \cap T_j\) set, the \(t_j\) will be given a poor rating even if it is a valid triangle.
5 Model Creation Tool

- A second option would be to assign the weighted product of the ratings. This is a poor choice for the same reasons as assigning the weighted sum. Further, the problem is amplified greatly when taking the product since there are almost always a few poor scoring invalid matches (one would expect invalid matches to receive poor scores), which forces the rating to zero, making the rating useless.

- The option that was found to work best is to assign the best-rated triangle in the set \( T_i \cap T_j \). If \( t_i \) is a valid intersecting triangle for \( t_i \), the best-rated triangle is almost always a valid intersecting triangle for both \( t_i \) and \( t_j \), which we’ll refer to as \( t_k \). When storing the rating for each \( t_p \), a reference to the \( t_k \) triangle responsible for this rating is also stored. This triangle is required for the triangle grouping step described in the next section.

Before moving onto the triangle grouping section, an example of how the Intersection Rating algorithm determines valid intersections will be presented using the 2D example presented previously in Figure 5.6. In the 2D case, triangles become lines and lines of intersection become points of intersection. Also, intersection points are only compared based on their distance from each other.

This example determines which of the lines emanating from camera \( B \) is a valid intersecting line for \( A \) (i.e. the line emanating from camera \( A \) through point \( Y \)). Therefore, \( t_i \) is set to \( A \). Next, the lines that \( t_i \) intersects need to be examined (i.e. the set \( T_i \)). \( T_i \) contains the lines \( \{ B_X, B_Y, C_X, C_Y \} \). Each of the lines in \( T_i \) must be given a rating. \( B_X \) is examined first (i.e. \( t_i \) is set to \( B_X \)). Next, the lines that \( t_i \) intersects need to be examined (i.e. the set \( T_j \)). \( T_j \) contains the lines \( \{ A_X, A_Y, C_X, C_Y \} \). If \( t_j \) is a valid intersecting line of \( t_i \) then the only other valid intersecting lines must intersect both \( t_i \) and \( t_j \) (i.e. the lines in the set \( T_i \cap T_j \)). \( T_i \cap T_j \) contains the lines \( \{ C_X, C_Y \} \). Each of these lines is given a rating in order to determine what rating to give \( t_j \). \( C_X \) is examined first (i.e. \( t_k \) is set to \( C_X \)). \( t_k \) is given a rating by comparing the three intersection points \( \{ A_X-B_X, A_Y-C_X, B_X-C_X \} \). Figure 5.9 highlights these intersection points.
As can be seen in Figure 5.9 there is a reasonable distance separating the three intersection points. This will result in the \( t_k = C_X \) case getting a low rating.

The \( t_k = C_Y \) case is given a rating by comparing the intersection points \( \{ A_Y - B_X, A_Y - C_Y, B_X - C_Y \} \). Figure 5.10 highlights these intersection points.
In Figure 5.10 the intersection points are spaced even further apart than in the $t_k = C_X$ case. This results in the $t_k = C_Y$ getting an even lower rating than the $t_k = C_X$ case. $t_j = B_X$ will be given the $t_k$ rating, which is the maximum value of all the $t_k$ cases (i.e. $t_k$ is set to $C_X$). This is still a low rating but is exactly what one would expect and want since $B_X$ is not a valid intersecting line of $A_Y$. Note that along with the rating stored with $B_X$, a reference to the line responsible for this rating is also stored (i.e. $t_k' = C_X$).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>$A_Y$</td>
</tr>
<tr>
<td>$t_j$</td>
<td>$B_Y$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>${ B_X, B_Y, C_X, C_Y }$</td>
</tr>
<tr>
<td>$T_j$</td>
<td>${ A_X, A_Y, C_X, C_Y }$</td>
</tr>
<tr>
<td>$T_i \cap T_j$</td>
<td>${ C_X, C_Y }$</td>
</tr>
</tbody>
</table>

Table 5.1 $A_Y$ intersecting $B_Y$ configuration
Now that the $t_i = B_X$ case has been given a rating, the next $t_i$ in the set must be considered (i.e. $t_i = B_Y$). The configuration is summarised in Table 5.1. The $t_k = C_X$ case is given a rating by comparing the intersection points $\{A_Y-B_Y, A_Y-C_X, B_Y-C_X\}$. Figure 5.11 highlights these intersection points.

There is a reasonable distance separating the three intersection points in Figure 5.11, similar to the $A_Y$, $B_X$, and $C_X$ case. This will result in the $t_k = C_X$ case getting a low rating.

The $t_k = C_Y$ case is given a rating by comparing the intersection points $A_Y-B_Y$, $A_Y-C_Y$, and $B_Y-C_Y$. Figure 5.12 highlights these intersection points.
In Figure 5.12 the three points $A_Y$, $B_Y$, and $C_Y$ are almost coincident at the same location. The magnified view shows that due to floating-point error and errors in the recorded interior and/or exterior orientation parameters, intersections are rarely exactly coincident. The $t_k = C_Y$ case will get a very high rating, which is not surprising since all the subscripts match, indicating valid intersections between the three. The $t_i = B_Y$ case will be given this $t_k' = C_Y$ rating. This is much higher than the previous $t_k = C_X$ rating.

Continuing with the two remaining $t_j$, one finds that the $t_j = C_X$ case would receive a low rating due to the fact that $C_X$ is an invalid intersecting line of $A_Y$. However, the $t_i = C_Y$ case would get a very high rating. In fact, the $t_i = C_Y$ case would get the same rating as the $t_i = B_Y$ case, since $t_k' = B_Y$. The intersecting points being compared would be between the lines $A_Y$, $B_Y$, and $C_Y$ again. $B_Y$ and $C_Y$ are equally valid intersecting lines to $A_Y$. Only one can be chosen as the best match however (which ever is examined first, in this...
Model Creation Tool

case $B_y$). This is why along with the best match we always store a reference to the line (or triangle in the 3D case) responsible for the rating ($C_y$ in this case).

At this point every triangle will have a reference to its best matching triangle and the triangle responsible for that match. This information is used in the next step to group triangles together.

5.1.3.2 Triangle Grouping

After the intersection rating step, every triangle $t_i$ will have a reference to its highest rated triangle $t_j$ (i.e. the highest rated $t_j$ triangle in its intersection set $T_i$) and a reference to the triangle responsible for this rating $t_k$ (i.e. the $t_k'$ of $t_j$). This relationship is shown in Figure 5.13.

![Figure 5.13](image)

Figure 5.13 After the intersection rating step, every triangle will store a reference to two other triangles

Triangles are placed into groups based on the referencing structure that emerges from the intersection rating step. Essentially, the grouping process is performed in two steps:

1) The GSS (*Group Scope Set*) of each triangle is determined.

2) The GSSs are used to determine the groupings between the triangles.
The GSS for a triangle $t_i$ is a list of triangles that contains the triangle itself (in this case $t_i$), the GSS for $t_j$ (the best-rated $t_j$ of $t_i$) and the GSS for $t_k$ (the $t_k'$ of $t_j$). The GSS can only hold a single instance of each triangle. This ensures that the recursive algorithm used to create a GSS is terminating. Not every triangle will have the same size GSS. The size of these sets will vary depending on the number of triangles used to represent each primary edge, as well as the relationship between their lines of intersection.

The simple case arises when a primary edge is represented by three triangles. In this configuration each triangle $t_i$ refers to the other two triangles as either its $t_j$ triangle or as its $t_k$ triangle. In such a situation all three triangles have identical GSSs containing the three triangles (Figure 5.14). This configuration is called a core group. Every group, including groups made up of more than three triangles, will be structured around a core group.

If there are more than three triangles representing a primary edge there can be two broad types of complex configuration. The first complex configuration involves four or more triangles that represent the same primary edge, which consists of a core group plus other triangles that reference triangles in the core group but which are not referenced themselves. Only the members of the core group have identical GSSs while the other triangle(s) have GSSs containing the core group plus themselves (Figure 5.15). This results in the
real group consisting of four or more triangles even though this is not apparent from the GSSs of the core group or from the GSSs of the other triangles if there are more than four in total.

![Figure 5.15](image)

Figure 5.15 Four triangles representing a primary edge with three of the triangles forming a core group with a fourth referencing two of the triangles of the core group without being referenced itself

In Figure 5.15 we see a situation with four triangles representing a single primary edge. Three of the triangles form a core group while the fourth triangle references two of the triangles from the core group but is not referenced itself. In this case, the GSSs of the core group members do not contain all members of the group. However, the GSS of the fourth triangle does.

Note that if a fifth triangle was added that also referenced two members from the core group, no one GSS from any of the triangles would encompass the entire group.

The second complex configuration involves six or more triangles that represent the same primary edge which form two or more simple or complex configurations which do not reference each other. In this configuration each group is solved independently and then the groups are merged as a post-process (Figure 5.16).
The second step in the grouping process is to use the GSSs to sort the triangles into groups. The grouping algorithm runs in two phases:

1) In phase one, only triangles that have three triangles in their GSS are processed, i.e. the simple cases. Each triangle as well as its GSS members are assigned a new group. The first phase places the majority of the triangles into groups. Only unreferenced triangles like those shown in Figure 5.15 remain.

2) In phase two, these remaining triangles are assigned to an existing group provided their rating with the group is within some minimum threshold.

It may not be possible to assign a group to every triangle for a number of reasons. The user may not have used three primary lines to highlight a particular primary edge or there may be too great an error to group some primary lines together either due to an error in the camera’s interior and/or exterior orientation parameters or an error in primary line placement by the user. In such cases these triangles are marked as invalid.

5.1.3.3 Group Merging

The final step in the grouping process is group merging. This is required because sometimes a primary edge may be represented by six or more
5 Model Creation Tool

triangles, which form two or more self-contained groups with no inter-group referencing (Figure 5.16). If the groups were left the way they are, there would be two primary edges representing the same building edge instead of just one. The merging step simply compares each group to each other group by first comparing the core members. If it is found that the rating between these triangles is within some minimum threshold, the algorithm goes on to test every combination of group members to guarantee that they: (a) all intersect and (b) the lowest ranking observed is within some minimum threshold. If these two criteria are met, the two groups are merged.

5.1.4 Edge Averaging

Once all triangles have been assigned a group the primary edges must be determined for each group. This is obtained from the weighted average of all the intersection lines between all group members (Figure 5.17).

![Image of intersection lines and averaged edge]

Figure 5.17 The intersection lines between all group members are averaged to form the final primary edge.

RQ2 asked if it was possible to determine the correspondences between the line primitives automatically. When carried out by the user, the correspondence task can be very time consuming. SAMATS has shown one
5 Model Creation Tool

approach to answering this question by utilising georeferencing information from the images of the image set.

5.1.5 Vertex Merging

RQ3 asks if it is possible to determine the higher level topology of the model from floating edges. The first step in determining the higher level topology of the model is to determine the connectivity between the primary lines.

In the previous steps, the locations of the primary edges have been determined independently from each other. In some cases this is acceptable, since many primary edges will not connected to any other primary edge. However, sometimes primary edges are connected. This is indicated in the edge highlighting step by having two or more primary lines share the same junction point.

All primary edges that are connected need to have their connected endpoints coincident at the same location. This is achieved by creating a mapping between every primary line and every primary edge, and also between every primary line endpoint and every primary edge vertex. Determining the mapping between the primary lines and their corresponding primary edges is trivial, since all primary lines that contribute to a primary edge are contained in the group that created the edge. Determining which primary line endpoint corresponds to which primary edge vertex is more involved.

Firstly, the relationship between the two primary line junction points is determined. The major axis of the primary line is determined by enclosing the line in an AABB (axis-aligned bounding box) and marking the dominant axis as the major axis. Note that if the two axes are the same length the x-axis is marked as the dominant axis. The junction points of each primary line are marked as either positive or negative depending on their position relative to each other.
5 Model Creation Tool

To determine the corresponding vertex for each junction point, the edge is viewed using the exterior orientation parameters of the image camera the primary line was highlighted in. From this viewpoint, the edge is aligned with the primary line in the image. By marking each vertex as either positive or negative from this perspective, the mapping between the vertices and the junction points is complete.

Primary lines that share the same endpoint map to primary edges that should share the same vertex. By identifying these connections, any vertices that should be coincident are made coincident by averaging their positions (Figure 5.18).

![Image](image_url)

Figure 5.18 Connected primary edges need to have the position of their end vertices averaged so that they are coincident at the same location
5.6 Secondary Edge Determination

In order to obtain the complete edge outline of the model, the secondary edges need to be determined. The secondary edges are recovered using the same mapping information obtained during the vertex merging step. Firstly, the secondary line’s endpoints are determined. Secondly, the corresponding vertices for these endpoints can be resolved. A new group is created for each secondary line using these vertices as the secondary edge’s endpoints. The edge outline model is then complete (Figure 5.19).

Figure 5.19 Secondary edges are recovered by connecting primary edges whose primary lines were connected during the edge highlighting phase.
5.2 Surface Structure Determination

At this point the floating edges have been connected together so that the adjacency between the edges is known. The final step in recovering the higher level topology of the model is to use the edge outline model to resolve the surface structure of the model, as raised by RQ3.

Even though the edge outline of the model has been determined, there is still no surface data associated with the model. The model is only defined in terms of vertices and lines, but not in terms of surfaces and the triangles that make up each surface. Determining this surface structure information is broken into three steps:

1) **Surface Determination** – Determines the co-planar surfaces of the model.

2) **Surface Aligning** – Aligns all of the surface normals so that they are consistent across the model.

3) **Surface Triangulation** – Triangulates each surface of the model so that the model renders more efficiently.

These three steps are discussed in detail below.

5.2.1 Surface Determination

Surfaces are determined by treating the model as a graph, with the models vertices representing the nodes of the graph and the primary and secondary edges representing the edges of the graph. Each surface corresponds to a cycle in the graph, but not every cycle in the graph corresponds to a surface, as illustrated in Figure 5.20.
From the example above we can see that that vertices (1-5-4-9-10-6-1) form a cycle in the graph, but do not correspond to a surface in the model since these vertices do not all lie on the same plane. However, the vertices (2-3-8-7-2) are both a cycle in the graph and a surface in the model.

There are two main assumptions made in order to determine the surfaces from the vertices and edges:

- The model being created is assumed to be a *closed mesh*.
- The number of surfaces associated with each vertex is equal to the number of edges connected to it.

Surfaces are determined by finding the shortest cycles in the graph where all the vertices are co-planar.

### 5.2.2 Surface Aligning

Once all the model’s surfaces have been determined, the direction of the normal vector for each surface must be resolved. The first step is to determine the adjacency of the surfaces, since surfaces are aligned in pairs. Once the surface adjacencies have been determined, one of the surfaces of the model is flagged as the master surface, with all other surfaces flagged as slave surfaces.
5 Model Creation Tool

All slave surfaces that are adjacent to the master are aligned, becoming themselves masters in the process, then all slave surfaces adjacent to these newly flagged master surfaces are aligned, becoming masters themselves. The process continues recursively until all surfaces have been flagged as masters.

The aligning step uses the fact that adjacent surface pairs are attached along one of their edges. This edge can act like a hinge between the two surfaces making it possible to rotate one of the surfaces about this hinge so that the two surfaces are co-planar. If then the surfaces are transformed so that they are perpendicular with the z-axis (axis coming out of the page) with the hinge between them aligned with the x-axis (axis going to the right), we notice that the interior of one surface is above the hinge while the interior of the other surface is below the hinge.

Using this fact, each surface pair is aligned by transforming both the master surface and the slave surface so that their surface normals are aligned with the z-axis and the edge vector between them is aligned with the x-axis. Then, each surface is checked to see if its interior is above or below the hinge edge. If both surfaces are on the same side of the hinge edge they are misaligned so the normal of the slave surface is flipped. If the two surfaces are on opposite sides, the two surfaces are already aligned, see Figure 5.21 and Figure 5.22.

Even though the model’s surfaces have been determined at this stage there may be a problem with the model’s normals (i.e. they may all be pointing the wrong way). This is due to the fact that a random surface was chosen as the master surface at the beginning of the surface aligning step, but it was not determined whether or not this normal points the correct way. Since the building model is assumed to form a closed mesh, the images are assumed to have all been taken from inside the model or outside the model, never a mix of the two.
Therefore, by casting a ray from one of the cameras so that it intersects the model, and examining the orientation of the first intersected triangle's normal, one can easily decide whether to flip all the normals or not. If the dot
product between the normal and the direction of the camera is positive, the
normals need to be flipped, otherwise they are correctly orientated (Figure
5.23).

Figure 5.23 The normal of the first surface intersected by the ray from the camera must point
towards the camera

5.2.3 Surface Triangulation
Once each surface has been determined and aligned, each surface must be
decomposed into triangles. The surfaces in the model can be either convex
or concave, although the surfaces should not contain holes. The algorithm
used to triangulate each surface can be found in Joseph O’Rourke’s Computational Geometry book [35]. Firstly, each surface is orientated so that it is perpendicular with the z-axis. Secondly, the z-coordinate is ignored and the triangulation process treats the surface as if it was a 2D surface.

RQ3 asked if the topology of the model could be determined from the floating edges alone. SAMATS recovers the higher level topology of the model in three steps: Vertex Merging (Section 5.1.5), Surface Determination (Section 5.2.1), and Surface Aligning (Section 5.2.2). With the surface structure of the model resolved, the model is suitable for use in a standard geometry-based rendering engine.

Figure 5.24 An example of a surface being triangulated

5.3 Exporting
The last step is to export the building model for use either as a flat shaded model or as input to the Texture Extraction Tool. The model is exported, in Microsoft’s eXtension file format (i.e. a .x file).

5.4 Discussion
RQ2 asked the question if the correspondences between line features can be resolved automatically. By using georeferencing information, Section 5.1.3 demonstrated an approach to solving this problem by resolving correspondences in world-space rather than image-space.
5 Model Creation Tool

While Coorg’s modelling system [7] reconstructs buildings by identifying the roofline of buildings, SAMATS recovers the location of all highlighted edges. This makes the system much more flexible, being able to model any polyhedral shaped structure.

**RQ3** asked if the topology of the model could be determined from the floating edges alone. Both Taylor and Kriegman [44] and Petsa and Karras [36] systems were limited to recovering the location of floating edges only. SAMATS recovers the higher level topology of the model in three steps:

1) Vertex Merging (Section 5.1.5).
2) Surface Determination (Section 5.2.1).
3) Surface Aligning (Section 5.2.2).

With the surface structure of the model determined, the model is suitable for use in a standard geometry-based rendering engine.

While Chou and Teller [6] theorise the use of a global model to resolve correspondences, SAMATS resolves the problem on the local level. The correspondences between each edge are determined without considering the system as a whole. This makes the system a lot simpler and hence easier to automate and still produces good results as will be seen in the experiments and results chapter (Chapter 7).

SAMATS also does not require a rough model of the building in order to determine the correspondences automatically unlike Zlatanova and van den Heuvel’s system [55]. Their system is also limited to using only two images when determining the location of an edge, while SAMATS can use any number of images.

There are currently two main limitations to the modelling phase of the system:
5 Model Creation Tool

- Firstly, there is a lack of support for partially highlighted edges. This means that unless the entire span of an edge is visible in an image the edge should not be highlighted. This can make it difficult or even impossible to model buildings enclosed in tightly confined spaces, since it may not be possible to capture the entire length of some edges in such situations. Section 8.5.2 discusses a possible approach to overcome this problem.

- Secondly, the Surface Structure Determination algorithm makes the assumption that cycles in the graph that are coplanar are valid. The number of surfaces associated with each vertex is equal to the valency of the vertex (i.e. number of edges connected to the vertex). Currently, the algorithm uses the first set of surfaces that satisfy the co-planar constraint as being the valid surfaces, which is not always correct. Section 8.5.2 discusses possible improvements that could alleviate this limitation.

RQ1 asked if it is possible to take the flexibility of a semi-automated system and combine it with the reduced user interaction found in automated systems. Both RQ2 and RQ3 were raised as a result of RQ1. Therefore by presenting solutions to both RQ2 (automated Correspondence Determination) and RQ3 (Surface Structure Determination) an answer to RQ1 was demonstrated. SAMATS automates tasks that are usually performed by the user in semi-automated systems, resulting in a flexible automated system.

5.5 Conclusions

SAMATS successfully demonstrated an approach to automating the correspondence task that to date has been primarily resolved by the user. Along with the Surface Structure Determination from floating edges, the Model Creation Tool answers questions RQ1, RQ2, and RQ3 posed by the review.
6 TEXTURE EXTRACTION TOOL

Coming into this chapter, we have an accurate model of the building, or to be precise, we have an accurate geometric model of the building. There is still data contained in the image set that has not yet been used to increase the model’s realism (i.e. the building’s façades). The aim of the texture extraction process is to extract façade data from the images and use this data to produce a photorealistic 3D building model. SAMATS, like Coorg’s approach [7], favours efficiency over high fidelity since the models produced will likely be used in larger city models. The texture extraction process is **fully automated** and can be broken into four steps:

1) **Initialisation** – Performs all miscellaneous setup.

2) **Texture Determination per Triangle** – Creates a texture for each triangle of the model.

3) **Texture Packing** – Packs each triangle texture into a single texture.

4) **Exporting** – Creates the final model with the packed texture associated with it.

Each of these steps will be described next.

6.1 **Initialisation**

The initialisation step performs all the miscellaneous actions required for the Texture Determination per Triangle and Texture Packing steps. Initialisation is performed in three steps:

1) **Triangle Setup** – Positions the viewing camera to render each triangle.

2) **Image Setup** – Determines the matrix required to project each image onto the model.

3) **Image Contribution Determination** – Determines which images contribute to each triangle texture.
Each of these steps is discussed below.

6.1.1 Triangle Setup

Each triangle in the model is represented as a triangle object, with each triangle object being processed independently of the others. Triangles need to be viewed from a particular position and orientation in order to capture the façade information correctly.

The camera is positioned so that each triangle is rendered with its longest side aligned with the bottom of the view. This results in the triangle being enclosed in the smallest possible bounding box. The camera is positioned at a specific distance so that there is a buffered area around the triangle, the mipmap buffer area (Section 6.2.3). Figure 6.1 shows how a triangle should appear when viewed using this camera.

The triangle is rendered from this viewpoint by creating the corresponding WVP (world view projection) matrix.

Firstly, the dimensions of the minimum AABB are determined. To minimise the size of the AABB the triangle’s longest edge is aligned with the AABB’s
6 Texture Extraction Tool

major axis. The mipmap buffer area is accounted for by adding twice the mipmap buffer zone size to the x and y components of the AABB.

The WVP matrix is created by concatenating three other matrices: the W (world) matrix, the V (view) matrix, and the P (projection) matrix. The W matrix is used to move the model around in the world. Since the model is already in the correct reference frame the W matrix is set to the identity matrix. The V matrix transforms the scene to camera-space (i.e. the scene objects are positioned relative to the camera). In camera-space the camera is positioned at the origin pointing straight down the positive z-axis. Finally, the P matrix transforms the scene from 3D camera-space to 2D image-space.

The V matrix is constructed by setting the camera’s position \( p \), look at point \( \text{lap} \), and up vector \( \text{up} \).

\( \text{lap} \) is calculated using the following equation:

\[
\text{lap} = \frac{\text{aabb\_lower\_left}}{2} + \frac{\text{aabb\_x\_axis}}{2} \times AABB\_WIDTH + \frac{\text{aabb\_y\_axis}}{2} \times AABB\_HEIGHT
\]

(Equation 6.1)

Where,

\( \text{lap} \) is the look at point (vector),

\( \text{aabb\_lower\_left} \) is the position of the lower left corner of the AABB in world-space (vector),

AABB\_WIDTH is the width of the AABB (scalar),

AABB\_HEIGHT is the height of the AABB (scalar),

\( \text{aabb\_x\_axis} \) is the AABB’s x-axis direction in world-space (vector),

\( \text{aabb\_y\_axis} \) is the AABB’s y-axis direction in world-space (vector).
Texture Extraction Tool

\( \mathbf{p} \) is determined using the following equations:

\[
P = \text{lap} + L \times \text{normal} \quad \text{(Equation 6.2)}
\]

\[
L = \frac{\text{AABB}_\text{HEIGHT}}{2 \times \tan(VFOV / 2)} \quad \text{(Equation 6.3)}
\]

Where,

\( \mathbf{p} \) is the camera position (vector),

\( \text{normal} \) is the triangle’s unit normal vector (vector),

\( L \) is the distance of the camera position from the look at point (scalar), and

\( VFOV \) is the vertical field of view to be used (scalar), which is generally set to \( \pi/4 \) (or 45 degrees). Note that this value is totally separate from the interior orientation parameters of the camera used to capture the original image set. This camera is only used to capture each triangles’ texture information.

\( \text{up} \) is simply the positive y-axis of the AABB in world-space:

\[
\text{up} = \text{aabb}_\text{y\_axis} \quad \text{(Equation 6.4)}
\]

The \( \mathbf{P} \) (projection) matrix is constructed using the VFOV (vertical field of view) and the ASPECT\_RATIO of the AABB:

\[
\text{ASPECT\_RATIO} = \frac{\text{AABB}_\text{WIDTH}}{\text{AABB}_\text{HEIGHT}} \quad \text{(Equation 6.5)}
\]

Figure 6.2 shows a diagram of how these values relate to the triangle shown in Figure 6.1.
6.1.2 Image Setup

Similarly to the Triangle Setup, each image in the image set is represented as an image object, with each image object being processed independently of the others. An image can be projected onto the model using the image camera’s interior and exterior orientation parameters. In order to project an image onto the model, the texture coordinates of each vertex in the model need to be set so that the correct portion on the image is accessed. The texture coordinates are determined by viewing the building model from the image camera’s viewpoint. From the image camera’s viewpoint, the outline of the building in the original image and the outline of the building model overlap. By transforming the model to image-space using the image camera’s WVP matrix, the texture coordinates for each vertex can be determined. By using
these texture coordinates to access the original image the image is effectively projected onto the model (Figure 6.3).

The position $\mathbf{p}$, the up vector $\mathbf{up}$, the vertical field of view VFOV, and the aspect ratio ASPECT_RATIO can all be determined directly from the interior
and exterior orientation parameters of the image camera. The look at point \( \text{lap} \) is determined by first rotating the unit z-axis vector using the yaw, pitch and roll of the image camera. This vector is then added to \( \mathbf{p} \) to give the look at point \( \text{lap} \).

The \( \text{WVP} \) matrix transforms the vertices into image-space, which has a range from \((-1, 1)\) in the upper left corner to \((1, -1)\) in the lower right corner. Texture coordinates have a range from \((0, 0)\) in the upper left corner to \((1, 1)\) in the lower right corner. To convert the \( \text{WVP} \) matrix to a \( \text{WVPT} \) (world view projection texture) matrix, the \( \mathbf{T} \) (texture) matrix is simply concatenated to the end:

\[
\begin{bmatrix}
0.5 & 0 & 0 & 0 \\
0 & -0.5 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0.5 & 0.5 & 0 & 1
\end{bmatrix}
\]

(Equation 6.6)

Now the x and y components of the transformed vertices can be used as texture coordinates to index into the image texture.

### 6.1.3 Image Contribution Determination

The final step of Initialisation is to determine the number of potential images that contribute to each triangle’s texture. For any particular image, only about half of the triangles that make up the model are visible if the model is closed, which is assumed to be the case. About half of the triangles should be facing the camera (i.e. front-facing), while the rest will be facing away from the camera (i.e. back-facing). This implies that about half of the triangles can be culled away from having any one image as a candidate texture source. An image is stored as a potential contributor to a triangle if the dot product between the triangle’s normal vector and the image camera’s view vector is greater than or equal to zero (Figure 6.4).
6.2 Texture Determination per Triangle

Once all the Initialisation has been performed each triangle’s texture contribution can be determined. This process is broken into three steps:

1) **Single Image Texture Capture** – Determines each image’s texture contribution to a triangle.

2) **Texture Blending** – Blends all the image contributions together for a single triangle.

3) **Mipmap Buffer Filling** – Fills the area around the triangle texture so that the texture does not darken at high mipmap levels.
6 Texture Extraction Tool

Each of these steps is explained in detail below.

6.2.1 Single Image Texture Capture

The first step in determining a triangle’s final texture is to store each image’s contribution in a separate render surface. Each image contribution is determined in a number of steps. Firstly, an occlusion map is created using the image camera’s WVP matrix. An occlusion map is an image of the model from the image camera’s viewpoint, but instead of storing colour information, the depth of each pixel is encoded in the image [15].

Figure 6.5 A simple scene with a camera and two spheres. The camera is able to view the black highlighted portions of each sphere. Points A and B are in this highlighted portion while point C is not. Although the surface normal at point C is pointing towards the camera, point C is being occluded by the red sphere.

Figure 6.5 shows a scene consisting of two spheres and a camera. Both spheres are inside the camera’s view frustum, but the smaller red sphere is in front of the larger blue sphere, occluding a large portion of the blue sphere. In the scene there are three points indicated, each on the surface of one of the spheres. Points A and B are visible in the camera image, but point C is not. Figure 6.6 shows the scene from the camera’s viewpoint.
6  Texture Extraction Tool

Figure 6.6  The scene from Figure 6.5 as viewed from the camera

Figure 6.7  The occlusion map for the camera from the scene in Figure 6.5
To determine if a point is visible in the camera image, the depth of every point at each pixel is determined, this is the occlusion map (Figure 6.7). Points close to the camera appear darker than points far from the camera.

Points B and C are represented by the same image pixel. To determine which is visible the distance of each point from the camera is encoded using the same scheme used for the occlusion map. This distance is then compared to the points corresponding pixel in the occlusion map. If the distance is less than or equal to the value in the occlusion map, the point is visible. Point B has the same distance as that stored in the occlusion map; therefore it is visible by the camera. However, point C has a higher value (further distance) than that stored in the occlusion map. Therefore, it is not visible by the camera. The occlusion map is used to prevent the projection of the image onto the model from appearing on surfaces that are blocked (or occluded) by other surfaces closer to the image camera.

One of the main problems with occlusion maps is there limited depth accuracy. Leaving the occlusion map depth range the same as the image camera’s depth range wastes precious measurement points. The solution is to use a technique presented by Michal Valient [49], which maximises the available range by using near and far custom depth planes. The near custom depth plane is positioned just in front of the model, while the far custom depth plane is positioned just behind the model (Figure 6.8). This way the model spans the majority of the available [0…1] depth range.
The triangle is rendered using its own WVP matrix (Section 6.1.1). The occlusion map and the image are projected onto the model using the image camera’s WVPT matrix (Section 0). The colour information at each surface element is stored in the RGB channels while the contribution is stored in the alpha channel.

The contribution of a surface element depends on the following factors:

- **Scaling Factors:**
  - The inverse distance of the surface element from the image camera.
  - The relative orientation between the surface element normal and the vector connecting the surface element and the camera.

- **Boolean Factors:**
6 Texture Extraction Tool

- Is the surface element occluded.
- Is the surface element behind the camera.
- Is the surface element outside the image's projected frustum.

Figure 6.9 Using the near custom depth plane as the origin of a point's distance from the camera would shift the relative distance considerably. Point A is 32 units from the camera's focal point. Point B is 44 units from the camera's focal point. Therefore point A is 32/44 (or 8/11) times the distance of point B from the camera's focal point. If the near custom depth plane were used, point A would be 2 units away and point B would be 14 units away. Therefore, point A would be only 2/14 (or 1/7) times the distance of point B, which is not the case. Note that even using the standard near clip plane shifts the origin by 8 units. To correct this problem the distance of the near and far custom depth planes are passed to the shaders so that the relative distance can be correctly rescaled.

The inverse distance of the surface element from the image camera is transformed to the range [0…1], with surface elements near the camera having values close to one, while surface elements near the far custom depth plane having values close to zero. Note that the near custom depth plane is not used in the depth calculation since it would incorrectly shift the depth results (Figure 6.9). The relative orientation between the surface element normal and the normalised vector connecting the surface element and the
camera is determined by taking the dot product between the two. The result is clamped to the range [0…1] (i.e. negative values are not used). At this point the contribution for a surface element will lie in the range [0…1].

The three Boolean tests either leave the value as it is or set it to zero. Determining if the surface element is occluded is performed as follows:

1) Read the value stored in the occlusion map using the surface element texture coordinates.

2) Compare this value to the distance between the surface element and the camera.

3) If the distance between the surface element and the camera is further than the value stored in the occlusion map, there must be another surface occluding the surface element. Therefore its contribution is set to zero; otherwise its contribution should be left unchanged.

The mathematics of image projection are symmetric, therefore the image gets projected in front of the camera as desired, but the inverted image is projected behind the camera. To prevent surfaces behind the camera from receiving the projection, the position of the surface element relative to the camera is determined. If the surface element is behind the camera, its contribution should be set to zero, otherwise it should be left unchanged. The relative position of the surface element is determined by examining the dot product between the camera’s view direction and the vector connecting the camera and the surface element. If the dot product less than or equal to zero, the surface element is behind the camera, therefore its contribution is set to zero; otherwise its contribution should be left unchanged.

The final Boolean test determines whether the surface element is outside the image camera’s projected frustum. This is determined simply by checking the texture coordinates of the surface element. If its texture coordinates are outside of the range [0…1] the surface element’s contribution is set to zero; otherwise it is left unchanged.
6.2.2 Texture Blending

Debevec’s view-dependent texture mapping approach [9] [11] keeps the texture contributions for each surface separate. When texturing each surface, the textures that were captured closest to the current viewpoint are blended together and used. Although the results from this technique are good, there is a considerable increase in the rendering complexity over the standard single texture mapping technique. RQ5 asks if textures can be consolidated to reduce the rendering complexity of a model.

The first step towards answering this question is to condense each triangle’s contribution from many textures to a single texture. Section 6.2.1 described how to capture the texture of a single image for a triangle in a render surface. Once all the texture surfaces for a triangle have been created, these surfaces need to be consolidated into a single surface by blending them together. The blending is performed per pixel using the values stored in the alpha channels as the contribution weightings. Firstly, the sum of the alpha values across all the contributing surfaces is determined. Each surface’s contribution is equal to the pixel colour multiplied by its alpha value divided by the alpha sum (Table 6.1 for an example).

<table>
<thead>
<tr>
<th>Function Channel</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>Total Alpha</th>
<th>S1 Contrib.</th>
<th>S2 Contrib.</th>
<th>S3 Contrib.</th>
<th>Final Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>0.3</td>
<td>0.8</td>
<td>0.7</td>
<td>1.8</td>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Red</td>
<td>0.5</td>
<td>0.6</td>
<td>0.5</td>
<td>0.1000</td>
<td>0.0833</td>
<td>0.2667</td>
<td>0.1944</td>
<td>0.5444</td>
</tr>
<tr>
<td>Green</td>
<td>0.6</td>
<td>0.7</td>
<td>0.6</td>
<td>0.1000</td>
<td>0.3111</td>
<td>0.2333</td>
<td>0.6444</td>
<td></td>
</tr>
<tr>
<td>Blue</td>
<td>0.6</td>
<td>0.6</td>
<td>0.5</td>
<td>0.1000</td>
<td>0.2667</td>
<td>0.1944</td>
<td>0.5611</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1 Example of a pixel being blended from 3 surfaces

In the example in Table 6.1 there are three surfaces: S1, S2, and S3. S2 is slightly redder and greener than the other two surfaces, while S3 is slightly less blue than the other two. One should also note that the S2 contribution has the largest alpha value, followed by S3, with S1 contributing the least. Each component value of the final colour is between the maximum and minimum of the corresponding surface component values, as one would expect, with
the alpha value of each surface weighting the final colour proportionately towards its colour value. Note that the final colour of each pixel has its alpha value set to one.

### 6.2.3 Triangle Texture Filling

Although the image contributions have been consolidated into a single texture, the texture is not suitable for use when mipmapping is enabled in the rendering engine. A mipmap is a size hierarchy of a single texture, with each level in the hierarchy halving the dimensions of the previous level. Mipmaps are used to prevent aliasing artefacts when rendering a textured object from a distance. If the triangle texture was left the way it is, the texture would darken at higher mipmap levels when the hierarchy is generated using bilinear filtering, as mentioned in Alex Evans GDC 2005 presentation [14]. Figure 6.10 demonstrates how darkening occurs. To prevent this darkening effect there are three options available:

![Figure 6.10](image)

*Figure 6.10* In mipmap level 0 there is a small blue texture. The outline of the triangle that uses this texture is highlighted in white. At level 1, the texture is halved in size along each axis with black and blue texels being averaged together to form darker blue texels. The darkening gets progressively worse at higher mipmap levels until at level 3 there is only a single texel, which is the average of all the texels from the original texture.
1) **Fill the packed texture with the average colour of the non-empty triangle texture texels** – A packed texture is a texture that stores all the triangle textures for the model (Section 6.3). Before copying in the individual triangle textures, the entire packed texture could be set to the average colour of the non-empty triangle textures’ texels, i.e. texels with a non-zero alpha value. Although this would be a viable option if all triangles were of similar colour, colour bleeding artefacts appear along the edges of triangles at high mipmap levels if the triangles differ in colour considerably.

2) **Fill each triangle texture with the average colour of the non-empty triangle texture texels** – This is similar to setting the colour of the entire packed texture to the average colour of the non-empty triangle textures’ texels, although each triangle texture is processed independently of the others. The entire triangle texture is first set to the average colour of the non-empty triangle texture’s colour texels. Then the triangle texture overwrites this average colour filled texture so that the original non-empty texels have their original value but all empty texels are now set to the average colour. Although this would be a viable option if the individual triangle textures did not vary in colour, colour bleeding occurs at edges that differ from the average colour of the triangle.

3) **Clamp the edges of the triangle to fill the empty texels** – In order to eliminate all bleeding artefacts the edges of the triangle texture sweep outwards to fill the empty texels completely.

The Texture Extraction Tool implements the third option to eliminate texture darkening while also not introducing colour bleeding artefacts as follows:

![Triangle texture surface with the empty regions shown](image)
6 Texture Extraction Tool

Firstly, the surface is broken into 7 regions (Figure 6.11). Texels in the bottom-left, bottom-right, and top regions sample the triangles respective corner texels. These corner texels are determined by using the triangle’s WVPT matrix to transform the vertices to texture coordinate image-space. On some graphics hardware, triangles can be rasterised differently than from the DirectX specification [30] resulting in neighbouring texels being sampled by mistake. This can be a problem if the texel being sampled is an empty texel. To correct for this a spiral search is performed to find the nearest texel that has a non-zero alpha value (Figure 6.12).

![Figure 6.12](image_url)

Figure 6.12  The triangles lower left vertex accesses the triangle texture at the red X. Texels are only filled with colour data if their centre point is inside the triangle. The two blue texels above are the only texels with their centre points inside the triangle. To find the first non-empty texel, a spiral search is performed starting at the texel accessed by the vertex’s texture coordinates.

Texels in the bottom region sample the first non-empty texel above their location. Texels in the left and right regions trace the path from their location...
6 Texture Extraction Tool

along the inverse slope of the respective triangle sides. Figure 6.13 shows a screenshot of a triangle texture before and after it has been filled.

![Before vs After texture filling](image.png)

Figure 6.13 The triangle texture varies in colour quite considerable along the bottom edge. By sweeping the edge texels outwards, neither darkening nor colour bleeding will occur at higher mipmap levels.

6.3 Texture Packing

While texture blending (Section 6.2.2) was a first step to answering RQ5, by consolidating each triangle’s texture contributions from many textures down to a single texture, each triangle still had its own texture. Texture packing advances the response to RQ5 further by consolidating all triangle textures into a single texture map. This makes it possible to render the entire building model in a single draw call using hardware that does not suppose multitexturing, which reduces the rendering complexity considerably. Note that the mipmap buffer zone is created around the triangle texture because the triangle textures were to be packed into a single packed texture. If there was no mipmap buffer zone, adjacent triangles in the texture would bleed into each other at higher mipmap levels.

The texture packing algorithm takes a list of triangle objects as input, each with an AABB, and determines the position of each AABB to form a tightly
packed square. The algorithm runs iteratively until the minimum area square is discovered (Figure 6.14).

Along with the arrangement of the triangle textures, the required scaling factor and the UV coordinates for each vertex are determined. Note that all triangles retain their relative size, thus creating an authalic texture map. The dimensions of the packed texture are generally larger than those of the individual triangle textures. However, the texel resolution of the individual triangle textures in the packed texture is generally reduced due to the large number of triangle textures stored.
6 Texture Extraction Tool

For this reason, a mipmap hierarchy is created for each triangle texture to reduce its size while utilising all available texture information. The mipmap level used in the packed texture must have a texel resolution greater than or equal to the width of the packed triangle texture, but less than twice the texel resolution of the width of the packed triangle texture (Figure 6.15). Since the packed texture will be authalic, the choice of mipmap level only needs to be performed once since all triangles will use the same mipmap level.

By sampling the chosen mipmap level using bilinear filtering, 100 percent of the width colour information is utilized. Not all of the height colour information may be utilized since there may be fewer height samples (102 height samples in Figure 6.15) than half the mipmap height dimension (128 texels in Figure 6.15). This means that colour information from some of the texels will be lost, although due to the small amount of such lost information, the effect is negligible. Figure 6.16 shows the final packed texture of the barn from Section 7.3.
6.4 Exporting

The final step simply sets the texture coordinates of the model to index into the packed texture correctly, sets the texture as the model’s material, and exports both the model and the texture to the output directory.

6.5 Discussion

The Texture Extraction Tool automatically extracts the façade information from the images of the image set and stores the information in a form that makes it suitable for rapid rendering.
RQ5 posed the question of how small can the rendering complexity attributed to texture mapping be made. Debevec’s view-dependent texture mapping approach [9] [11] requires a custom texturing technique, which would require additional computation on the CPU, although it may be possible to implement this functionality on the GPU using shaders. One unavoidable cost of this technique is the increased memory requirement for the additional textures. The technique also requires each triangle to be rendered separately, and in multiple passes if the hardware does not support multi-texturing, which increases the rendering complexity considerably.

By not only blending triangle textures together, but also consolidating all triangle textures into a single packed texture, the rendering complexity due to texture mapping is reduced substantially. By having only a single texture for the model, the model can be rendered in a single draw call, even on hardware that does not support multi-texturing.

The triangle texture filling has no effect on the rendering complexity, but it is required to eliminate texture darkening when mipmapping is enabled, which is generally the case. Three options were examined with the edge sweep method providing the best results.

The main limitation of the texture blending technique is that since the geometric model created by the Model Creation Tool is used when capturing each images contribution, errors present in the model will result in shifts in the triangle textures’ colour information. This results in blurred triangle textures once all the contributions are blended together since each contribution will likely be shifted differently. Although this is not too noticeable when using synthetic imagery since the interior and exterior orientation parameters are known precisely, the effect would probably be more noticeable when using photographic imagery. One possible solution would be to use only a single contributing image for each triangle texture, blending two or more images together only when absolutely necessary.
Two features from Coorg’s texturing approach [7] could also be used:

- Firstly, Coorg uses a statistical median technique for each contributing texture to balance the illumination across the images. This would be important when using photographic image sets since the lighting conditions often change from photo to photo. This was not required when using synthetic imagery since a simple Lambertian diffuse lighting model [13] was used to illuminate the scene, therefore the colour of each surface does not change with respect to viewing angle.

- Secondly, Coorg determines which pixels in the individual image contributions are occlusion objects by comparing the pixel values in the individual images to those of the final image contribution. The greater the difference, the more likely the pixel in the individual image is an occluding object. These pixels can be given a lesser weighting or can be removed completely when recreating the final image contribution. This was also not required since the test scenes all consisted of a single textured building.

6.6 Conclusions

Along with the Texture Blending step, the Texture Packing step shows that the textures for an entire model can be consolidated into a single texture, reducing the rendering complexity in the process. Although there are a number of issues that can arise when using the texture blending technique (e.g. the possibility of blurry textures), these effects would be more obvious when using photographic image sets.

The next chapter tests SAMATS using a set of six test scenes, with each scene introducing a new structural feature that must be considered in order to model the structure correctly. The stability of SAMATS is also tested in the next chapter.
7 Experiments and Results

7 EXPERIMENTS AND RESULTS

This chapter tests SAMATS using six test scenes created using the Scene Capture Tool described in Chapter 3. Each test scene introduces an additional structural feature that must be considered in order to model the building correctly. Three aspects of each test scene will be examined.

1) **Image Set Acquisition** – Image set acquisition discusses what special considerations were made when capturing the image set of the particular scene. The number of images required is also recorded.

2) **Edge Highlighting Statistics** – The length of time required to model a building is dependent on the number of junction points, primary lines and secondary lines needed to be placed in the images of the image set, which are recorded in this section. The total time required to place these primitives is also recorded.

3) **Results and Analysis** – Finally, the quality of the reconstructed model is examined in this section.

After examining the six test buildings, the stability of SAMATS will be examined. This is achieved by successively adding error to the camera’s exterior orientation parameters to determine the maximum error tolerable by SAMATS for a particular building type. The most simplistic test scene (i.e. John Hancock building test scene) and the most complex test scene (i.e. pyramid test scene) are used as the test cases for the stability tests.

7.1 John Hancock Building

The first test scene is a reconstruction of the famous John Hancock building in Chicago. The model is very simplistic consisting of only 6 sides. The only notable feature of this building is that the building tapers as it ascends so that the building has a larger base area than roof area (Figure 7.2).
7 Experiments and Results

7.1.1 Image Set Acquisition
For this type of building only 4 images were required, each taken from the four cardinal corners of the building looking up at roughly 30 degrees so that the full length of the building is visible in each shot.

![Image](image.png)

Figure 7.1 The highlighted edges of the John Hancock building test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow.

7.1.2 Edge Highlighting Statistics
The edge highlighting phase of the modelling process required the user to highlight 3 primary lines in each image. These corresponded to the three vertical edges visible in each image. This resulted in 12 primary lines being placed in total. Each one of these primary lines required 2 junction points that resulted in 24 junction points being placed in total. Finally, 8 secondary
7 Experiments and Results

lines were placed to highlight the roofline and the baseline of the building (Figure 7.1). Note that each secondary edge was only highlighted in a single image. If every secondary edge were highlighted in every image a total of 16 secondary lines would have been placed but since this additional information is redundant it was not added.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Junction Points</th>
<th>Primary Lines</th>
<th>Secondary Lines</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>24</td>
<td>12</td>
<td>8</td>
<td>2 Minutes 50 Seconds</td>
</tr>
</tbody>
</table>

Table 7.1 John Hancock test scene edge highlighting statistics

Figure 7.2 Comparison of the original John Hancock building test scene (left) with the reconstructed model (right)
7.1.3 Results and Analysis

The John Hancock building test scene took less than three minutes to highlight and appears almost identical to the original. The most time consuming part of the highlighting phase involved the precise placement of the junction points. Connecting these points using primary or secondary lines was a relatively quick process. The main noticeable difference between the test scene model and the reconstructed model is the resolution of the façade surface. The final texture size used is 1024 x 1024 pixels, which is the same size as the texture used for the original model. However, the original texture has a single building face placed diagonally across the entire texture. This single façade is used for all four faces of the original model resulting in a higher resolution façade than the reconstructed model, since SAMATS repeats the façade four times in the texture resulting in lower resolution façades. Even with this reduction in façade resolution, the final model is very similar to the original (Figure 7.2).

Note that although the roof texture in the reconstructed model matches the original roof texture (black), this is purely coincidental. Any surface or part of a surface that is not visible in any of the images of the image set appears as black.

Also note that the differences in lighting conditions between the original model and the reconstructed model are due to the fact that the Scene Capture Tool uses four directional lights to light the scene when creating the image set. These lighting values are then stored in the reconstructed model’s texture causing the variation in texture illumination.

7.2 Office Block

The second test scene is of an L-shaped office block (Figure 7.4). The approach to modelling this build is the same as that for modelling the John Hancock building. The vertical edges are modelled using primary lines while
Experiments and Results

Secondary lines are used to connect the primary lines together, highlighting the roofline and the baseline of the building.

7.2.1 Image Set Acquisition

The main notable feature of this building is the concave portion defined by its L-shape. Care must be taken to ensure that the vertical edge on the inside of this concave portion appears in 3 images (edge A in Figure 7.3). An additional 3 images must be taken to recover the remaining primary edges, in particular the primary edge on the outer side of the L-shape (edge B in Figure 7.3) is not visible in any of the images that capture the primary edge on the inner side of the L-shape and vice-versa. This results in a total of 6 images.

![Figure 7.3](image)

Figure 7.3 The highlighted edges of the office block test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow

7.2.2 Edge Highlighting Statistics

Table 7.2 summarises the number of primitives placed in the edge highlighting phase as well as the time required to place all of the primitives (Figure 7.3).
7 Experiments and Results

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Junction Points</th>
<th>Primary Lines</th>
<th>Secondary Lines</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>46</td>
<td>23</td>
<td>12</td>
<td>5 Minutes 17 Seconds</td>
</tr>
</tbody>
</table>

Table 7.2 Office block edge highlighting statistics

7.2.3 Results and Analysis

Unlike the John Hancock building, the original office building model is purely geometry based. All of the surface detail is created geometrically (i.e. there are no textures associated with the model). Only the vertex colour of each vertex is used to add variation to each surface’s appearance. Even though the original model is made up of more than 20 thousand triangles and the model created by SAMATS consists of just 20 triangles, there is very little difference between the two models (Figure 7.4).

Note that the occlusion map feature of SAMATS is also exercised when modelling the office block. The L-shape of the building makes it possible for some of the surfaces of the building to face the camera even though the camera is being occluded by other parts of the building. If an occlusion map was not used during the texture extraction phase, incorrect texture information could be stored with these surfaces.
7 Experiments and Results

Figure 7.4 Comparison of the original office block test scene (top) with the reconstructed model (bottom)

7.3 Barn

The next test scene is of a barn (Figure 7.6). There are a number of features to this structure that are important to note in order to model it correctly. Firstly, each side of the barn consists of a surface with 5 sides. This makes it necessary to connect primary lines together in order to capture the side structure correctly. Second, at the front of the barn the roof is raised in the middle. Primary lines must be used to capture this structural feature even
7 Experiments and Results

though there is only a limited view of the edges of this feature, making it a less than ideal primary line candidate.

![Edge A](image)

Figure 7.5 The highlighted edges of the barn test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow

7.3.1 Image Set Acquisition

In order to capture the horizontal raised roof edge (edge A in Figure 7.5) using primary lines, four images were taken of this edge from the front of the barn. Four images were taken to facilitate the recovery since the edge was a poor primary edge candidate, with more primary lines improving the chance of capturing the edge correctly. An additional two images were taken from the rear of the barn, resulting in six images in total.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Junction Points</th>
<th>Primary Lines</th>
<th>Secondary Lines</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>50</td>
<td>37</td>
<td>11</td>
<td>6 Minutes 46 Seconds</td>
</tr>
</tbody>
</table>

Table 7.3 Barn edge highlighting statistics
7.3.2 Edge Highlighting Statistics
Table 7.3 summarises the number of primitives placed in the edge highlighting phase as well as the time required to place all of the primitives.

7.3.3 Results and Analysis
The main difficulty in modelling this structure was in capturing the raised roof geometry correctly. Similarly to the previous two test scenes, the main difference between the original model and the reconstructed model is the difference in façade resolution (Figure 7.6). The surfaces that suffer the greatest reduction in façade detail are the roof surfaces, although this is due to a different reason. Since the roof surfaces are at grazing angles when viewed from the image cameras’ viewpoints, only a limited surface area is visible in any one image. The surface detail captured from these limited views is actually lower than the maximum resolution supported by the packed texture. This results in the roof surfaces appearing less detailed than the wall surfaces. While increasing the dimensions of the packed texture would improve the wall surface detail, it would not improve the roof surface detail. The roof detail is limited by the original image set data, not the texture consolidation process.
7.4 House with Extension

The next test scene is of a house with an extension (Figure 7.8). Although similar to the office block test scene there is one important difference, it is not possible to capture the horizontal edge connecting the roof of the extension to the rear house face in any image taken from the ground (edge A in Figure 7.7). This edge must be highlighted in order to maintain the constraint that all surfaces must be planar. To achieve this, an optional manual step was added to the Model Creation Tool that allows the user to connect primary edges.
7 Experiments and Results

together, in effect creating secondary edges directly in world-space. Note that this is an optional step that is not required by the majority of structures.

7.4.1 Image Set Acquisition
Similarly to the office block test scene, care must be taken to ensure that both the vertical edge on the inside of the concave portion of the extension (edge B in Figure 7.7) and the vertical edge linking the adjacent roof corner of the extension to the main roof (edge C in Figure 7.7) appear in 3 images. An additional image is taken from the front left of the house, totalling 7 images.

7.4.2 Edge Highlighting Statistics
Table 7.4 summarises the number of primitives placed in the edge highlighting phase as well as the time required to place all of the primitives.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Junction Points</th>
<th>Primary Lines</th>
<th>Secondary Lines</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>51</td>
<td>34</td>
<td>14</td>
<td>8 Minutes 58 Seconds</td>
</tr>
</tbody>
</table>

Table 7.4 House with extension edge highlighting statistics

As mentioned previously an additional edge needed to be highlighted in the Model Creation Tool. This simply involved connecting the desired two vertices together by dragging the mouse between the two.

7.4.3 Results and Analysis
As with the previous test scenes, the main difficulty in modelling this test scene was deciding on the locations to capture each of the images of the image set. To make it possible to model this structure an enhancement was made to the Model Creation Tool, enabling the user to define secondary edges directly in world-space.
Figure 7.7  The highlighted edges of the house with extension test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow. Note that the green edge (edge A) is not visible in any image taken from ground level.

The main differences between the original test scene model and the reconstructed model are again in the façade resolution. Also, since the extension roof is not visible from any image, the roof of the extension is black.

Also note that a portion of the white wall texture that appears just above edge A is blacked out since this portion of the wall is not visible in any of the images.
7 Experiments and Results

Figure 7.8 Comparison of the original house with extension test scene (top) with the reconstructed model (bottom)

7.5 Cottage

The fifth test scene is of a cottage (Figure 7.10). Like the barn test scene, special considerations must be taken when deciding which roof edges should be highlighted using primary lines and which should be highlighted using secondary lines.
7 Experiments and Results

7.5.1 Image Set Acquisition

Even though this is the most complex structure modelled so far, with careful choice of edge type, the model was created using only 9 images. The problem areas of this structure are the two vertical edges in the concave section of the cottage (edges A and B in Figure 7.9) and the two horizontal roof edges (edges C and D in Figure 7.9).

![Figure 7.9 The highlighted edges of the cottage test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow](image)

7.5.2 Edge Highlighting Statistics

Table 7.5 summarises the number of primitives placed in the edge highlighting phase as well as the time required to place all of the primitives.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Junction Points</th>
<th>Primary Lines</th>
<th>Secondary Lines</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>64</td>
<td>32</td>
<td>23</td>
<td>9 Minutes 04 Seconds</td>
</tr>
</tbody>
</table>

Table 7.5 Cottage edge highlighting statistics

Even though roof edges C and D are not ideal primary line candidates, by choosing these edges as primary edges rather than the diagonal roof edges, six less primary edges needed to be highlighted and recovered.
7 Experiments and Results

Note that all edges of this model, like the previous test scenes, were highlighted in less than 10 minutes.

7.5.3 Results and Analysis

Similar to the previous test scenes, the main noticeable difference between the original model and the reconstructed model is the façade detail (Figure 7.10).

Figure 7.10 Comparison of the original cottage test scene (top) with the reconstructed model (bottom)
7 Experiments and Results

7.6 Pyramid
The final test scene to be modelled is considerably more complex than the previous test scenes. Unlike the previous test scenes that only represented the exterior of a structure, this test scene consists of the outside of a pyramid that transitions into an interior tomb. Very few modelling systems have demonstrated the flexibility to handle exterior to interior transitions. Since SAMATS does not make any assumptions regarding the shape of the structures it models, exterior to interior transitions are handled transparently, provided the entire structure forms a closed mesh.

There are quite a number of features that must be considered when modelling the pyramid (Figure 7.11). The peak edge in the tomb was treated as a primary edge although it was not possible to connect this primary edge to the vertical primary edges of the tomb walls due to the tight confinement of the tomb area. Similarly to the house with extension test scene, these connections had to be made in the Model Creation Tool. We could have used primary lines for the entire roof section, which would have removed the need to add the connections in the model creation phase, although this would have required many more images to be taken. It was quicker and easier to add the connections in the modelling phase, rather than capture more images of the tomb area and highlight the required edges in the edge highlighting phase. Care was also needed when connecting the doorway of the tomb with the exterior of the pyramid.

7.6.1 Image Set Acquisition
To capture the exterior geometry of the pyramid, four images were taken from the four cardinal corners of the pyramid. Three additional images were captured of the doorway, resulting in seven images taken from the outside of the pyramid. Due to the confined space inside the pyramid, 15 images were needed to capture the interior geometry correctly, for a total of 22 images.
7 Experiments and Results

7.6.2 Edge Highlighting Statistics

Table 7.6 summarises the number of primitives placed in the edge highlighting phase as well as the time required to place all of the primitives.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Junction Points</th>
<th>Primary Lines</th>
<th>Secondary Lines</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>106</td>
<td>66</td>
<td>30</td>
<td>17 Minutes 06 Seconds</td>
</tr>
</tbody>
</table>

Table 7.6 Pyramid edge highlighting statistics

Figure 7.11 The highlighted edges of the pyramid building test scene. Primary lines are highlighted in red and secondary lines are highlighted in yellow.

7.6.3 Results and Analysis

The final model produced is very similar to the original model with all textures being applied correctly. Note that some of the tomb surfaces were not visible in any of the images so those portions of the surface are black. Also, due to the large number of textures that are being blended together for some of the surfaces (e.g. the floor inside the tomb), some areas appear slightly blurred. This is due to textures being applied at grazing angles, similar
to the problem with the barn roof (Section 7.3.3). Slight errors in the reconstructed model result in texture contributions miss aligning.

It took just over 17 minutes to model the pyramid model. With the possible exception of Façade [9], none of the modelling systems reviewed in Sections

Figure 7.12  Comparison of the original pyramid test scene (top) with the reconstructed model (bottom)
7 Experiments and Results

2.8 or 2.9 would be able to model the pyramid structure correctly. Even though it may be possible to model this structure using Façade, a very large number of block primitives would be required to fill the pyramid structure while leaving the tomb area empty. It would also be very difficult to place these block primitives and make correspondences between these block primitives, due to the limited surface visibility from any one image.

7.7 SAMATS Stability Tests

This section tests the tolerance of SAMATS’ Model Creation Tool to errors in the cameras’ exterior orientation parameters. This gives an estimate of the accuracy requirements of a real world image set in order for it to be suitable for use with SAMATS. Both the John Hancock building test scene and the Pyramid test scene are used as test cases. Two sets of tests are performed on each test scene.

1) **Positional Error** – This test adds a random positional error within a defined threshold to each image’s camera location, e.g. a one meter error would mean that the camera is placed somewhere in the 2 x 2 x 2 meter box centred at the camera’s original location.

2) **Orientation Error** – This test adds a random orientation error within a defined threshold to each image’s camera axes, e.g. a one degree error would mean that the camera is randomly rotated up to one degree around each of its three local axes.

In order to determine the average success rate when experiencing a particular error, ten runs are performed with each run introducing a new random error. The average success rate is given by the number of times the model is successfully reconstructed over the number of runs performed. The first test scene examined is the John Hancock building test scene, which is followed by the Pyramid test scene.
7 Experiments and Results

7.7.1 John Hancock Stability Tests

The first series of tests performed on the John Hancock building test scene determines the maximum position error that can be tolerated when reconstructing the geometric model of the building.

<table>
<thead>
<tr>
<th>Test</th>
<th>Error (meters)</th>
<th>0.5</th>
<th>1.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>10/10</td>
<td>9/10</td>
<td>7/10</td>
<td>6/10</td>
<td>1/10</td>
</tr>
</tbody>
</table>

Table 7.7 The results for the John Hancock building test scene adding position error only

Figure 7.13 A graph of the results from Table 7.7
7  Experiments and Results

From the results shown in Table 7.7 it can be seen that SAMATS begins to have difficulty in creating the building model when the error is greater than 0.5 meters. Note that even with an error as great as 2.0 meters, 60 percent of the time the model can be reconstructed. However, at 2.5 meters error or greater, the model is rarely reconstructed correctly.

SAMATS is able to cope with the reasonably high position error for this test case because the building being reconstructed is so large. The error introduced by the position error can only be as large as the error itself (i.e. if the recorded position of the camera is exactly 1 meter from its actual position, its projected rays will miss their respective targets by a maximum of 1 meter). Since the primary edges are over 100 meters from each other, an error in the order of a meter is tolerable.

The second series of tests performed on the John Hancock building test scene determines the maximum orientation error that can be tolerated when reconstructing the geometric model of the building.

<table>
<thead>
<tr>
<th>Test</th>
<th>Error (degrees)</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>8</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Total</td>
<td>8/10</td>
<td>6/10</td>
<td>3/10</td>
<td>2/10</td>
<td>0/10</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.8  The results for the John Hancock building test scene adding orientation error only
From the results shown in Table 7.8 it can be seen that SAMATS is very sensitive to errors in the cameras’ orientations. As little as a 0.1 degrees shift about each local axis is sufficient to cause SAMATS to fail in two of the ten tests to create the geometric model of the building. At 0.5 degrees of error, none of the ten tests succeeded in creating the geometric model of the building.

While the position error can only introduce an error up to its own value, the error introduced by the orientation error increases linearly with the distance between the target point and the image camera. The dimensions of the John Hancock building are roughly 100 meters in width by 100 meters in length by 344 meters in height. Because of the size of these dimensions the image cameras are roughly 400 meters from the building in order to capture a full view of the building. When the image camera is 400 meters from the target,
7 Experiments and Results

an error of 0.3 degrees results in the projected ray passing 2.09 meter from the target. The 2.0 meters position error had a 60 percent success rate while the 2.5 meters position error had only a 10 percent success rate, which implies that the 30 percent success rate of the 0.3 degree orientation error is as expected.

7.7.2 Pyramid Stability Tests

The first series of tests performed on the pyramid test scene determines the maximum position error that can be tolerated when reconstructing the geometric model of the structure.

<table>
<thead>
<tr>
<th>Test</th>
<th>Error (meters)</th>
<th>0.02</th>
<th>0.035</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>10/10</td>
<td>9/10</td>
<td>5/10</td>
<td>2/10</td>
<td>0/10</td>
</tr>
</tbody>
</table>

Table 7.9 The results for the Pyramid test scene adding position error only

From the results shown in Table 7.9 it can be seen that when using the pyramid test scene SAMATS is very sensitive to errors in the cameras’ locations. At an error of 3.5 centimetres, one out of ten tests fail. At an error of 20 centimetres all tests fail.

135
The second series of tests performed on the pyramid test scene determines the maximum orientation error that can be tolerated when reconstructing the geometric model of the structure.

<table>
<thead>
<tr>
<th>Test</th>
<th>Error (degrees)</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>5</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>6</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>8</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Total</td>
<td>10/10</td>
<td>3/10</td>
<td>2/10</td>
<td>2/10</td>
<td>0/10</td>
<td></td>
</tr>
</tbody>
</table>

Table 7.10  The results for the Pyramid test scene adding orientation error only
From the results shown in Table 7.10 it can be seen that SAMATS is even more sensitive to errors in the cameras’ orientations when using the pyramid test scene than when using the John Hancock test scene. Any error over 0.05 degrees has a serious impact on the reliability of SAMATS to create the geometric model correctly.

There are two reasons for the pyramid test scene being more sensitive to errors than the John Hancock building test scene;

1) **Fine Detail** – In the John Hancock building test scene only four primary edges were recovered with each primary edge being at least 100 meters from the other three. In the pyramid test scene, there are primary edges that are adjacent less than 1 meter from one another.

2) **Number of Primary Edges Recovered** – As the number of primary edges to be recovered increases, the chance of recovering the model correctly decreases since it is more likely that one or more primary edges will not be recovered correctly. The John
Hancock building test scene has only four of its edges designated as primary edges while the pyramid test scene has 21 edges designated as primary edges. If even one of these edges fails to be recovered, the entire model is unrecoverable.

7.8 Discussion

One of the limitations with SAMATS as it exists at present relates to the texture consolidation step. As can be seen with the interior floor of the pyramid test scene textures can appear blurred if either the exterior orientation parameters have error or the user miss places primary lines in the edge highlighting phase.

The second concern regards the accuracy requirements of the exterior orientation parameters. Although centimetre accuracy is possible using custom built rigs such as Argus [7] [8] or that used by Zlatanova and van den Heuvel [55], these rigs are not readily available. As discussed in Section 2.2, smaller more affordable options are beginning to appear although it may be several years before such devices have the accuracy required by SAMATS.

Although the georeferencing accuracy requirements are high, when this requirement is met, SAMATS is a flexible system that lies on the divide between being semi-automated and automated. By eliminating the manual correspondence step found in the majority of contemporary modelling systems, as well as automatically determining the surface structure of the model from the edge outline model, SAMATS has demonstrated that it is possible to reduce the user interaction required of a flexible system.

7.9 Conclusions

The stability tests have shown that SAMATS is sensitive to errors in the image cameras’ exterior orientation parameters. These tests have shown that SAMATS requires centimetre accuracy in positional information and better than 0.1 degrees accuracy in orientation information in order to recover building models successfully.
7 Experiments and Results

However, when these requirements are met, the six test scenes have shown that SAMATS is able to model a wide array of structure types making it a flexible system. The time required to model each of the six test scenes was reasonably short due to the automation of the correspondence step and the surface structure determination.

The next chapter presents the final conclusions and suggestions for extending the scope of SAMATS.
8 CONCLUSIONS AND FUTURE WORK

There are many different approaches to creating 3D building models. One of the most popular approaches utilises aerial imagery stereo-pairs. Noronha and Nevatia [34] give an example of building modelling using aerial imagery while Ye and Lee [52] give an example of roof modelling using aerial imagery. Two of the main disadvantages of using such imagery are that the imagery is not freely available in many countries (Ireland being one such country). Also, the resolution of the façade textures is usually relatively poor.

A technique that is gaining in popularity is that of laser range scanning. Although the models generated using this technique can be very detailed, there are a number of disadvantages at present (Section 2.6).

A third popular modelling technique involves the use of terrestrial imagery to create accurate 3D building models. The two main advantages of this technique are:

- Terrestrial Imagery is **Readily Available**.
- The Façades of the Buildings are **Captured Accurately**.

Debevec’s Façade [9] makes use of such terrestrial imagery to reconstruct accurate building models. One of the disadvantages with using Façade is that quite a large amount of user interaction is involved in the modelling process. Coorg’s system [7] automates most of the modelling process, but sets restrictions and constraints on what the system is capable of modelling in order to achieve these goals.

SAMATS (Semi-Automated Modelling And Texturing System) was created to investigate techniques that automate the steps that are carried out manually in contemporary modelling systems. The goal of SAMATS was to create a system that was flexible like semi-automated systems, while having the
8 Conclusions and Future Work

reduced user interaction of automated systems. SAMATS achieves this by requiring the images to be georeferenced. This georeferencing information is then used to automate the correspondence step found in the majority of contemporary modelling systems.

8.1 Research Questions Answered

The background and related research chapter (Chapter 1) raised a number of questions that guided this research. The main research question (RQ1) asked if it was possible to take the best features of both automated and semi-automated systems to create a flexible automated system. In answering this question another two research questions were raised.

By automating two of the tasks that are usually performed by the user in semi-automated systems (RQ2 and RQ3), SAMATS answers RQ1 by demonstrating such a system.

RQ2 asked if there was a way of automating the correspondence step found in the majority of contemporary modelling systems. This correspondence step being the most time consuming task in the modelling process. Section 5.1.3 presented a solution to this problem by utilising georeferencing information from the image set. While stereo systems resolve correspondences in image space, SAMATS demonstrated an approach to resolving the correspondences automatically in world-space. By automating this step, the most time consuming task in the modelling process has been eliminated.

RQ3 asked if the higher order topology of the model could be determined from the floating edges. Both Taylor and Kriegman [44] and Petsa and Karras [36] systems were limited to recovering the location of floating edges only. SAMATS recovers the higher order topology of the model in three steps; vertex merging (Section 5.1.5), surface determination (Section 5.2.1), and surface aligning (Section 5.2.2). With the surface structure of the model
8. Conclusions and Future Work

resolved, the model is suitable for use in a standard geometry-based rendering engine.

**RQ4** asked if it was possible to obtain a more accurate building model by only considering strong candidate edges. This is exactly what primary and secondary lines achieve. Primary lines highlight strong candidate edges, which have been captured from varying viewpoints that have a large perpendicular displacement component relative to the target line feature. On the other hand secondary lines simply connect primary lines together, since they do not satisfy this requirement. Without differentiating between strong and weak edge candidates the reconstructed models would be either skewed or unrecoverable.

**RQ5** posed the question of how small can the rendering complexity attributed to texture mapping be made. By not only blending triangle textures together but also consolidating all triangle textures into a single packed texture, the rendering complexity due to texture mapping is reduced substantially. By having only a single texture for the model, the model can be rendered in a single draw call, even on hardware that does not support multi-texturing.

### 8.2 Limitations

The main limitation of SAMATS at present is that it has not yet been used with real world image sets. From the stability tests discussed in Section 7.7 it was shown that SAMATS is sensitive to errors in both position and orientation. Although this is no reason to think that using a real world image set which meet the requirements of SAMATS would introduce additional problems, the requirements of SAMATS are still quite high with respect to the accuracy attainable by standard positional equipment today.

SAMATS’ lack of support for partially highlighted or partially occluded edges is also a limitation that could limit its use in the real world. This can make it
8 Conclusions and Future Work

difficult or even impossible to model certain structures due to spatial constraints. For example, in a tight enclosure it may not be possible to capture the full length of a long wall edge. Using a wide-angle lens would reduce the impact of this limitation but it would still be beneficial to support the use of partially highlighted edges. Section 8.5.2 discusses a possible approach to extending SAMATS to handle partially highlighted edges.

The Surface Determination algorithm makes a number of assumptions that may not be correct in all circumstances. For example, the algorithm assumes that the shortest co-planar cycles in the edge outline model are valid surfaces. Some structures may have geometry that conflict with this assumption. Section 8.5.2 discusses possible improvements that could alleviate this limitation.

The main limitation of the texture blending technique is that since the geometric model created by the Model Creation Tool is used when capturing each images contribution, errors present in the model will result in shifts in the triangle textures’ colour information. This results in blurred triangle textures once all the contributions are blended together since each contribution will likely to be shifted differently. Section 8.5.3 discusses possible improvements that could alleviate this limitation.

8.3 Contributions

The goal of this thesis was to demonstrate modelling techniques that reduce the amount of user interaction required to create complex 3D building models using georeferenced terrestrial imagery. The techniques proposed in this thesis towards achieving this overall goal were based on the following ideas:

- **Correspondence Determination** – By using georeferenced imagery and the automated correspondence determination algorithm as described in Section 5.1.3, it is possible to eliminate the manual correspondence step required by the majority of contemporary modelling systems.
8.4 Success Criteria

In Section 1.4 two criteria questions were raised to assess to what extent SAMATS has succeeded in achieving its goals:

1) In what ways has SAMATS been able to streamline the contemporary building modelling workflow?

2) To what extent has SAMATS retained the flexibility to model a wide variety of building types?

In relation to the first question, SAMATS achieves its primary speedup from the elimination of the correspondence step found in the majority of contemporary modelling systems. Add to this the automation of the Vertex Merging, Surface Determination and Surface Aligning steps and the result is a significant reduction in the length of time required to complete the modelling.
workflows. With regard to the level of user interaction metric defined in Section 2.1, SAMATS scores a four overall consisting of a score of two for acquisition, two for modelling/registration, and zero for texturing. Since SAMATS does not use real world imagery it was decided to award SAMATS a score of two for acquisition, matching that of the MIT City Scanning project [7] [45] [46], since this system is the closest applicable approach for SAMATS. As stated in Section 2.2, after registration the Argus rig produces centimetre position accuracy and 0.05 degrees orientation accuracy, which is sufficient for use with SAMATS as demonstrated in Section 7.7. SAMATS is given a score of two for modelling/registration due to the need for the user to place line primitives. SAMATS’ score is one less than Debevec’s Façade [9] [10] [11] since no correspondences need to be made between the line primitives. Finally, SAMATS is given a score of zero for texturing since no user interaction is required for this step.

In relation to the second question, SAMATS has been able to retain most of the flexibility found in manual or semi-automated systems. SAMATS is capable of modelling any polyhedral shaped structure provided no surfaces contain holes. With regard to the flexibility metric defined in Section 2.1, SAMATS scores a six overall consisting of a score of three for both modelling and texturing. SAMATS scores a three for modelling since it lies between the flexibility of the MIT City Scanning project presented by Teller et al. [7] [45] [46], which is limited to modelling rectilinear shaped building that scored a two and Debevec’s Façade [9] [10] [11], which is capable of modelling reasonably complex building structures that would be more difficult to model using only line features that scored a four. SAMATS is given a score of three for texturing since its texturing method is comparable to that of the MIT City Scanning project that also scored a three.

Therefore, using the above information SAMATS can be plotted on the Flexibility Vs. Level of User Interaction graph (Figure 8.1). The graph shows that SAMATS has managed to push into the green area of the graph.
achieving a better Flexibility / User Interaction Ratio \((3/2)\) than Façade \((4/3)\), although not as good a ratio as the MIT City Scanning project \((5/3)\).

![Figure 8.1 SAMATS plotted on the Flexibility Vs. Level of User Interaction graph from Figure 2.1](image)

### 8.5 Future Work

The next step in developing SAMATS is to test the system using real world imagery. Currently, SAMATS has only been used on synthetic images where the exact exterior and interior properties of the camera are known. Achieving such precision in the real world may prove difficult without specialized surveying equipment. New techniques will be required to facilitate the gathering of the georeferenced images required by SAMATS in order for this system to be utilized effectively in the real world.

The following subsections will discuss possible improvements for each of the three tools that make up the core of SAMATS.

#### 8.5.1 Edge Highlighting Tool

The Edge Highlighting Tool is the only tool that requires user interaction. As a result, this tool has the highest potential for improved efficiency. Edge
detected could be added to this tool to reduce the level of user interaction required to use the system. Two levels of automation could be explored.

1) **Edge Highlighting Assistance** – Having edge detection could greatly increase the pace at which edges could be highlighted. Using simple Canny edge detection to identify all straight edges the user would be able to highlight edges of interest simply by clicking on an edge with either the right or left mouse button depending on the whether the user wanted to place a primary line or a secondary line.

2) **Complete Automation** – It may be possible to extend the edge highlighting feature so that the entire edge highlighting phase would be completely automated. By using heuristics such as the orientation of the edges relative to the vertical world vector, edges could be designated as either primary or secondary. It is likely that this process would not be 100 percent accurate since some structures require special consideration (e.g. the raised roof of the barn from Section 7.3). This initial guess of edge types could be performed initially with the user making amendments as required.

The other main limitation with the Edge Highlighting Tool is its lack of support for partially highlighted edges. Although this task would be carried out in the edge highlighting phase, to support this feature the majority of the modifications would be applied to the Model Creation Tool, therefore this improvement will be discussed in the next section (Section 8.5.2).

### 8.5.2 Model Creation Tool

As mentioned in the previous section, one of the limitations with SAMATS at present is its lack of support for partially highlighted or partially occluded edges. Adding this feature would require changes to the Edge Highlighting Tool, but primarily change to the Model Creation Tool. The main changes that would be required are as follows.

1) **Line Projection** – At present, primary lines are projected to form 3D triangles. Since the primary edges are fully highlighted the valid intersections represent the full length of the primary edge. If partially highlighted lines were supported, there is no guarantee that the valid intersections would mark the full length of the primary edge or that all partially highlighted edges would intersect (imagine a primary edge highlighted in two images, in image A the top third of
8 Conclusions and Future Work

the edge is highlighted. In image B the bottom third is highlighted. When these lines are projected they will not intersect each other. The projected lines will only intersect if they both highlight the same portion of the edge in each image. To ensure that all lines contribute the primary lines are transformed to form planes instead of triangles, the plane being defined by the triangle. This ensures that all valid lines intersect each other.

2) Intersection Rating – Although the shift from triangles to planes ensures that all valid lines intersect, almost every line intersects, valid or not, which would drastically increase the number of intersections examined. Since plane intersections are being examined instead of triangle intersections, the way intersections are compared would need to be re-examined. Firstly, intersection length is meaningless since if two planes intersect, the line of intersection has infinite length. Intersection would be based on the distance between their closest points and their difference in orientation. In order to determine the length of a primary line the expanse of all the lines projected onto the average intersection line between all the contributing planes would need to be considered, with the length of the primary edge being the full expanse.

Note that there would definitely be special cases that would need to be considered if such a system was possible at all. For example, how would two edges that lie on the same line in world-space but are separated by a gap be handled. The partial edge system might assume that they represent a single long edge when in fact they represent two separate edges.

3) Secondary Edge Highlighting – If partially highlighted edges were supported, how would secondary edges be handled? Should these edges support partially highlighting? If so, how can one determine which primary edges are connected by such a secondary edge? An easier solution may be to enforce that secondary edges be fully highlighted although the primary edges they are connected to could be partially highlighted.

SAMATS currently resolves for correspondences locally (i.e. each primary edge group is not concerned with any other primary edge group). Although this group independence has made the system easier to design and develop, as discussed during the pyramid stability tests (Section 7.7.2), if a single primary edge fails to be recovered, the geometric model is unrecoverable. The system proposed by Chou and Teller [6] which determines edge correspondences on a global level may be more robust in handling errors. No implementation
8 Conclusions and Future Work

details or experimental results were presented, which implies that the design is purely theoretical.

The surface structure determination algorithm makes the assumption that cycles in the graph that are coplanar are valid. The number of surfaces associated with each vertex is equal to the valency of the vertex (i.e. number of edges connected to the vertex). Currently, the algorithm uses the first set of surfaces that satisfy the co-planar constraint as being the valid surfaces, which is not always correct. By examining the topology as a whole and resolving conflicts globally, this problem could be alleviated.

8.5.3 Texture Extraction Tool

The main problem with the Texture Extraction Tool is that textures can appear slightly blurred if there is either error in the exterior orientation parameters of the image cameras or the user misplaced primary line primitives resulting in the reconstructed model being slightly skewed. There are two approaches that could be used to alleviate this problem. Firstly, the texture for each portion of the model could be restricted to a single image contribution. This would guarantee that the texture would not be blurred although there may still be noticeable transitions across a surface if two or more sources were used for different portions of the surface. Another solution would be to transform the individual texture contributions so that the final contribution appears sharper. The sharpening technique used by Coorg [7] may be applicable.

A technique which would increase the visual fidelity of the models produced by SAMATS could use the view-dependent texture mapping presented by Debevec in [9]. However, this technique would require additional memory requirements as well as multiple textures per surface. SAMATS uses a more simplistic texturing approach which favours performance over maximum quality.
8 Conclusions and Future Work

With advances in GPU (Graphics Processing Unit) technology, the model-based stereo technique described Debevec’s Ph.D. thesis [9] can be adapted to work with a standard geometry-based rendering pipeline. Two possible approaches would be;

1) **Displacement Mapping** – This involves using a texture to hold displacement values which are used to displace vertices along their normal. The depth map recovered using the model-based stereo technique could be converted into a height map. The geometric model would then need to be tessellated to add additional vertices across each surface. By using the height map as a displacement map, fine detail in the building could be rendered by displacing the newly added vertices.

With the soon to be released DirectX 10 specification [17], the ability to create new geometry on the GPU using the Geometry Shader will make this approach even more appealing since only the simple building model would need to be passed to the GPU. The GPU would then be responsible for tessellating the model itself, reducing the amount of vertex data needed to be sent across the Bus.

![Figure 8.2 Screenshot of the pavement from the ATI Toy Shop technical demonstration that demonstrates parallax occlusion mapping. The 3D structure of the cobble stones is clearly visible even though the pavement is actually made up of flat polygons](image)

2) **Parallax Occlusion Mapping** – This technique uses both a height map and a normal map to add detail to a flat surface. Natalya Tatarchuk [43] presents the latest advances to this technique. Similar to the displacement mapping technique above, the depth map recovered using the model-based stereo technique could be converted into a height map. This height map could then be used to generate a normal map. Parallax Occlusion Mapping works by finding where the camera intersects with the height map. Shading is
then done using the recovered texture coordinates at this location. This technique has enormous potential for adding surface detail without the need to tessellate the surface further.

8.6 Final Note
The research presented in this thesis has demonstrated that by utilising information that is becoming more readily available it is possible to automate tasks that were previously preformed by humans. Although the accuracy requirements of SAMATS are quite high with respect to standard georeferencing equipment, the techniques demonstrated have their obvious strengths. By combining the techniques demonstrated here with more traditional approaches, a more robust modelling system may be possible.
BIBLIOGRAPHY


[19] Glassenberg, Sam Z. *Intro to DirectX 10 . . . and why the GPU will never be the same.* Game Developers Conference, Microsoft Sponsored Session, March 2006.


Bibliography


APPENDIX A: CODE OVERVIEW

This appendix discusses the three core tools of SAMATS, as well as their associated code. The following aspects of each tool are discussed:

1) **Code Overview** – This section briefly describes the special techniques or packages that are used.

2) **Compilation Instructions** – This section describes any special steps that must be taken in order to compile each tool project.

The three core tools of SAMATS, as well as their source code, are on the accompanying CD-ROM. Both the Scene Capture Tool and the Stability Test program (the Stability Test program is used to add random error to the exterior orientation parameters of an image set) are also on the accompanying CD-ROM for completeness.

Before describing the code of the three core tools a brief discussion of system and software requirements is presented, followed by a brief section discussing the test data sets.

**System and Software Requirements**

The system requirements for SAMATS are very low. The most demanding tool being the Texture Extraction Tool that can display the geometric model of a building with any of the image contributions projected onto it. An entry level PC would be more than capable of running SAMATS.

However, SAMATS does require the DirectX 9.0 runtime installed. This can be downloaded from the DirectX website [29].

The tool projects were created in Visual Studio .Net 2003 [29]. Although a different IDE (Interactive Development Environment) could be used, the use of .Net is advised since the project files are included on the CD-ROM.
Appendix A: Code Overview

Test Data Set Use
This section describes how the test data sets can be used with SAMATS as well as how new data sets can be created. The test scenes are contained in the \textit{TestSet} folder on the CD-ROM. Each test scene is organised in its own folder in the \textit{TestSet} folder. The test scene folder must contain the following six sub-folders:

- **input\_model** – This folder contains the original test scene. The test scene must be named \textit{model.x}.

- **materials** – This folder contains all the textures used by the original test scene.

- **output\_model** – This folder will contain the final model and well as the packed texture used by this model.

- **properties\_files** – This folder contains the properties files that are generated by the Scene Capture Tool, the Edge Highlighting Tool, and the Model Creation Tool.

- **screenshots** – This folder contains the images of the image set.

- **textures** – This folder contains the individual triangle textures that are used to create the packed texture.

To create a new test scene these folders should be created. If the Scene Capture Tool is to be used to generate the image set only the input model and its associated textures need to be added to the appropriate folders. If a real world image set is to be used, the images should be placed in the \textit{screenshots} folder with the corresponding properties file placed in the \textit{properties\_files} folder.

Edge Highlighting Tool
The Edge Highlighting Tool uses an image properties file (a .ipf file) to load in an image set. Once the image set has been loaded the user can select any of the images to work on. The user can switch between images as needed. Primitives can be placed and deleted from the images. A zoom facility is provided for the precise placement of junction points. When all primitives
Appendix A: Code Overview

have been placed, the highlighted scene can be exported as an edge properties file (a .epf file) for use with the Model Creation Tool.

**Code Overview**

The Edge Highlighting Tool is a relatively simple application. Like the other three tools the program is written in C++. The tool uses the MFC (Microsoft Foundation Classes) Document/View framework, which aids the rapid development of Windows based applications [20].

The Edge Highlighting Tool also makes use of FreeImage [18], an image loading and rendering library.

Aside from the framework and library code, an additional 1,600 lines of code were written from this application.

**Compilation Instructions**

The only additional library required is the FreeImage library, which can be downloaded for free from the web [18] (version 3.0 of this library is included on the CD-ROM).

**Model Creation Tool**

The Model Creation Tool uses an edge properties file (a .epf file) to create the geometric model of the captured building. The Model Creation Tool does not use the image set. The modelling process is completely automated although the tool has break points so that the user can view the intermediate outputs from previous steps. The main break point regards the optional placement of secondary lines in world-space. The geometric model produced is saved in the *output_model* sub-folder of the scene folder, along with a model properties file (a .mpf file).
Appendix A: Code Overview

**Code Overview**
The Model Creation Tool uses the DirectX 9.0 MFC framework. This framework allows for the simple placement of widgets beside the main view window.

Tomas Moller’s Triangle-Triangle intersection code [32] was used to determine the intersections between the projected triangles, and Joseph O’Rourke’s surface triangulation code [35] was used to triangulate the surfaces in the Surface Structure Determination step of the modelling process.

Aside from the framework and library code, an additional 10,000 lines of code were written for this application, making it the most complex of the three.

**Compilation Instructions**
To compile this application the DirectX 9.0 SDK is required. Tomas Moller’s code [32] and Joseph O’Rourke’s code [35] is provided on the CD-ROM (with acknowledgements to both parties).

**Texture Extraction Tool**
The Texture Extraction Tool uses a model properties file (a .mpf file) to load in the geometric model and the image set. Like the Model Creation Tool, this tool is automated although there are break points so that the user can view the intermediate outputs from previous steps. The final model and its packed texture are saved in the output_model sub-folder of the scene folder.

**Code Overview**
The Texture Extraction Tool, like the Model Creation Tool, uses the DirectX 9.0 MFC framework. No other libraries were used.

Aside from the framework code, an additional 5,200 lines of code were written from this application.
Appendix A: Code Overview

Compilation Instructions
Similarly to the Model Creation Tool, the Texture Extraction Tool requires the DirectX 9.0 SDK to compile.
Appendix B: Glossary

APPENDIX B: GLOSSARY

AABB. See Axis-aligned bounding box.

Aliasing. Aliasing occurs when a signal is sampled at too low a frequency. For a signal to be sampled properly, the sampling frequency must be at least twice that of the maximum frequency in the signal being sampled. This is known as the Nyquist frequency.

Authalic texture map. The triangles of a model that use an authalic texture map maintain their relative size. If there are two triangles in the model, the first being twice the area of the second triangle in world-space, the first triangle will access twice as much of the texture as the second triangle. Authalic texture maps often appear better than non-authalic texture maps since all texels are the same size on the model.

Axis-aligned bounding box. An axis-aligned bounding box is the minimum bounding box that encloses a certain object that is aligned with the basis axes of some reference frame (generally the world-space basis axes).

Baseline distance. The baseline distance is simply the distance between two cameras (generally stereo-pairs).

Block primitive. A block primitive is a simple shape (e.g. box, cylinder, etc.) that can be combined with other block primitives to create more complex shapes.

Building feature. A building feature is a significant point or edge that is visible in several images in the image set and makes up part of the outline of a building. Good examples of building features would be the corner points of the building, the wall edges, the baseline and the roofline.

Calibrated camera. Camera calibration is the process of determining the internal geometry and optical characteristics. A camera is said to be calibrated if the mapping between pixels in the image and the direction of rays from the camera’s focal point are known.

Camera-space. Camera-space is the view of the world from the camera’s viewpoint. When using a left-handed coordinate system the camera is placed at the origin looking down the positive z-axis. All scene objects are transformed by the inverse of the camera’s translation and rotation matrices to position them correctly relative to the camera.

Close range imagery. See Terrestrial imagery.
Appendix B: Glossary

Closed model. A closed model has exactly two polygons associated with every edge of the model. Any ray passing through the model will always pass through an even number of surfaces ensuring that the model forms a closed volume.

Closed mesh. See Closed model.

Constraints. A constraint is an observation about a building, which can be used to aid the modelling process. For example, if one knows that the building being modelled has all vertical and horizontal edges, which are perpendicular or parallel to one another, a perpendicular and parallel constraint can be made between all the lines. This allows the system to eliminate any configuration that does not satisfy the constraint, and hence makes the system more robust.

Control target. A control target is a marker of known position in a scene. Control targets can be used to determine the location of the cameras used to capture the scene.

Correspondence. Making correspondences between building features involves grouping features that appear in two or more images that represent the same building feature. Correspondences are carried out for two reasons; firstly, correspondences can be used to determine the interior and/or exterior orientation parameters of the camera(s). Second, since correspondences group features that highlight the same building feature, only members of a group are considered when determining the world-space location of the corresponding feature.

Depth map. A depth map represents the distance of objects from a camera. It is similar to a colour image but instead of colour information being stored at each pixel, the distance of the closest object at that pixel is encoded. Generally depth maps are grey scale images with objects that are further away appearing brighter than objects that are closer to the camera.

Disparity. The disparity of an object across two images is the measure of the objects shift in position relative to the cameras’ image centre points. The greater the disparity, the closer the object is relative to the two cameras.

Epipolar line. Given two cameras, C1 and C2, observing a point m. The plane defined by the focal points of C1 and C2, and the point m is called the epipolar plane. The epipolar lines defined by point m for cameras C1 and C2 are the lines of intersection between the corresponding epipolar plane and the image planes of cameras C1 and C2. The significance of the epipolar constraint is that for a point m visible in C1, the corresponding point in C2 must lie on the epipolar line defined by the focal points of cameras C1 and C2 and the projected image of m in C1’s image plane. This reduces the search
space of the correspondence from a 2D search over the entire image to a 1D search along the epipolar line in $C_2$'s image.

**Epipolar plane.** See **Epipolar line**.

**Expert knowledge.** Expert knowledge is a term used to describe a task that requires considerable training, experience, or knowledge.

**Exterior orientation parameters.** The exterior orientation parameters of a camera are its position $(x, y, z)$ and its orientation $(\omega, \varphi, \kappa)$ in world-space. When known, along with the interior orientation parameters, the 3D ray that goes from the focal point of the camera through any point in the image out to infinity can be determined relative to some world reference frame.

**Feature.** See **Building feature**.

**Flexible system.** A modelling system is said to be flexible if it is capable of modelling a large variety of structures.

**Focal point.** The focal point of a camera is the point where the entire image converges to a single point.

**Geometric complexity.** The geometric complexity of a model is determined by the number of vertices and polygons that make up the model.

**Geometric model.** A geometric model consists of points, lines, triangles, meshes, or higher order surfaces such as B-splines. The geometric model only stores basic material information such as the diffuse, ambient, and specular colour components.

**Georeferenced imagery.** An image is said to be georeferenced if its exterior orientation parameters are known to some world reference frame.

**Group scope set.** A group scope set is a list of triangles that are used to determine valid primary edge contributor. The group scope set for a triangle $t_i$ is a list of triangles that contains the triangle itself (in this case $t_i$), the group scope set for $t_j$ (the best-rated $t_j$ of $t_i$) and the group scope set for $t_k$ (the $t_k$ of $t_j$).

**GSS.** See **Group scope set**.

**Image-space.** Image-space is a 2D representation of a 3D scene from a camera’s viewpoint. Objects that appear inside the view frustum of the camera are transformed from camera-space coordinates to image-space coordinates in the range $(-1, -1, 0)$ to $(1, 1, 1)$. Note that a z-component can be discarded.
Appendix B: Glossary

**Image set.** An image set is a set of images taken of a building. The image set generally captures the building from all angles.

**Inertial tracker.** An inertial tracker is a system of sensors that are used to track changes in motion. The most basic form of inertial tracker involves recording the motion of the wheels of a rig to determine the rig's motion.

**Interior orientation parameters.** The interior orientation parameters of a camera are; the x-coordinate of the centre of projection, the y-coordinate of the centre of projection, the focal length, the aspect ratio, and the angle between the optical axes. When known, these parameters can be used to determine an exact mapping between a point in the image and the corresponding 3D ray being projected from the focal point of the camera through that point out to infinity relative to the camera.

**Key image.** The key image is the term used to describe the primary camera in a stereo rig. Points of interest are first identified in the key image, and then using stereo techniques such as the epipolar constraint the corresponding points are found in the offset image. The offset image is the term used to describe the secondary camera in a stereo rig.

**Metric Camera.** Metric cameras have fixed internal geometry, which makes the image feature measurements very precise. Metric cameras are generally not used to capture terrestrial imagery, although the higher accuracy they provide is usually required when capturing aerial imagery.

**Mipmap.** A mipmap is a hierarchy of textures. Mipmaps are used to reduce aliasing artefacts that can occur when viewing a model from a distance. When a model is viewed from a distance it may only appear as a few pixels on the screen, even though the visible portion of the texture may be several hundred pixels in width and height. In effect, the monitor is resampling the high frequency texture at a very low frequency, which would result in aliasing (see **Aliasing**). To eliminate aliasing the texture signal must be sampled at a higher rate or the frequency of the texture signal must be reduced. Since the resolution of the monitor is generally fixed, the only option is to reduce the frequency of the texture signal, which is exactly what mipmapping achieves.

**Occlusion map.** An occlusion map (also known as a shadow map) is depth map that can be used to determine which scene objects are visible from a given location.

**Offset image.** See **Key image**.

**Photographic imagery.** Photographic imagery is imagery captured using a film or digital camera that captures the visual wave-lengths of light emanating from the objects in the scene.
Appendix B: Glossary

**Pinhole camera.** A pinhole camera is a camera with a pinpoint representing the focal point of the camera. Since the image converges at a single point the images produced by pinhole cameras generally contain no lens distortion. However, having such a small aperture requires the shutter to be open for long periods of time in order to expose the film to sufficient light.

**Rectified image.** A rectified image is an image which has had all distortion removed. A type of distortion that is found in many images is radial lens distortion, which results in lines that are straight in world-space appearing curved in image-space.

**Render surface.** A render surface is an array of memory that represents a raster image.

**Rendering complexity.** The rendering complexity of an object is the length of time required to render the object. In contemporary rendering engines the rendering complexity of an object had a linear link to the geometric complexity of the object, i.e. the number of vertices that make up the object. This is no longer the case since modern rendering engines support special techniques such as parallax occlusion mapping or BRDF (Bi-directional Reflectance Distribution Function) materials, which can be applied to the object to increase the visual fidelity of the rendered object at the cost of increased rendering complexity.

**Restrictions.** A restriction on a system refers to the system’s ability to model certain structures but not other structures. For example, many automated modelling systems represent buildings as rectilinear shapes. Such a system is therefore restricted to modelling rectilinear buildings.

**SAMATS.** SAMATS is the modelling system developed in this thesis. SAMATS stands for Semi-Automated Modelling And Texturing System. SAMATS is composed of three separate tools; the Edge Highlighting Tool, the Model Creation Tool, and the Texture Extraction Tool. Together these tools make it possible to create photorealistic geometrically accurate building models.

**Spherical mosaic.** A spherical mosaic is a set of images captured from the same location in varying directions. While a panoramic image can be constructed using a series of images of each portion of the horizon from the same location, a spherical mosaic captures the entire scene from a single location.

**Stereo-pair.** A stereo-pair of images is two images of the same scene captured from different viewpoints that overlap to some degree. The aerial imagery used by many modelling systems consists of stereo-pairs.
Appendix B: Glossary

**Surface element.** A surface element is a portion of a surface. Since the resolution of a monitor is finite, when rendering a surface discrete points of the surface are sampled. These points are known as surface elements.

**Synthetic imagery.** A synthetic image is a computer generated image of a scene, the scenes themselves generally being composed of texture mapped polygons.

**Terrestrial imagery.** Terrestrial imagery is imagery taken at ground level.

**Texel.** A texel is the abbreviated form of texture element. Texel is a synonym for the pixels that make up a texture but is also used to describe the individual texture elements visible on a model.

**Texture atlas.** A texture atlas is a single texture that is used to hold many smaller textures. Texture atlases are used to eliminate texture swapping that can severely reduce rendering performance.

**Valency.** The valency of a vertex is the measure of the number of edges attached to the vertex.

**Voxel.** A voxel is the three dimensional equivalent of a pixel: a finite volume within 3D space. Some alternative 3D rendering techniques use voxels to render 3D scenes where each voxel may be assigned a colour value.

**World-space.** World-space refers to an objects location in 3D-space.
APPENDIX C: PUBLICATIONS


**SAMATS – Edge Highlighting and Intersection Rating Explained**

Joe Hegarty and James D. Carswell

Digital Media Centre, Dublin Institute of Technology, Aungier St., Dublin, Ireland.
{joe@dmc.dit.ie, jcarswell@dit.ie}

**Abstract.** The creation of detailed 3D buildings models, and to a greater extent the creation of entire city models, has become an area of considerable research over the last couple of decades. The accurate modeling of buildings has LBS (Location Based Services) applications in entertainment, planning, tourism and e-commerce to name just a few. Many modeling systems created to date require manual correspondences to be made across the image set in order to determine the models 3D structure. This paper describes SAMATS, a Semi-Automated Modeling And Texturing System, which has the capability of producing geometrically accurate and photorealistic building models without the need for manual correspondences by using a set of geo-referenced terrestrial images. This paper gives an overview of SAMATS’ components, while describing the Edge Highlighting component and the Intersection Rating step from the Edge Recovery component in detail.

1 Introduction

This research investigates building reconstruction technology for creating geometrically accurate, photorealistic 3D models from terrestrial digital photography for use in LBS (Location Based Services) applications. It is envisioned that the resulting 3D model output from this work be web-enabled and made available to subsequent LBS research endeavors (e.g. for archaeologists, town planners, tourism, e-Government, etc.). Being able to produce 3D building models using terrestrial imagery allows all users to exploit the future commercialization potential of web-based LBS, as demonstrated in [1].

[10] was the first to investigate the principle of structure from motion. [9] builds on these ideas using lines instead of points, although both require correspondences to be made manually across the image set. In fact the majority of semi-automated reconstruction systems require the user to make manual correspondences across the image set in order to reconstruct a model, which is generally a very time consuming task. [3] is one of the most robust systems using this approach which allows the user to create models using a set
of block primitives and by setting constraints on those primitives. A more automated modeling approach involves the modeling of roofs using aerial imagery. Models produced in this way can produce structurally accurate models but fail to capture building façades accurately, although [5], [6], and [7] have looked into the merging of façade textures with models produced from aerial imagery. [2] constructs a large set of 3D building models by using spherical mosaics produced from accurately calibrated ground view cameras fitted with GPS. Although highly automated, this system was limited to modeling simple shaped buildings by simply identifying the rooflines and extruding walls downwards.

SAMATS uses a novel approach to creating building models without the need for manual correspondences to be made. [11] is an example of extracting building and window edges without the need for manual correspondence, although a rough model of the structure being modeled is required in order for this system to work. No prior building model is required by SAMATS. The ability of SAMATS to remove the manual correspondence step found in most modeling approaches is achieved by having all images geo-referenced in the same reference frame. However, the acquisition of geo-referenced terrestrial images is still a serious bottleneck that does not have a straightforward solution. Currently public GPS will give an absolute accuracy of between 1 to 10 meters using a single receiver. This resolution is not technology bound but information restriction bound, with military GPS offering centimeter accuracy. As private industries or other governments create their own satellite networks these restrictions may no longer apply - making the acquisition of accurate geo-referenced imagery as simple as regular imagery. SAMATS does not solve the difficulties in acquiring geo-referenced imagery - it only investigates the usefulness of such imagery in the overall modeling process.

![Fig. 1. SAMATS system diagram. The highlighted steps are the focus of this paper](image)
Appendix C: Publications

This paper gives an overview of the entire SAMATS system, while focusing on the Edge Highlighting component and the Intersection Rating step of the Edge Recovery component. For a detailed description of the other components refer to [4]. Figure 1 shows a systems overview of SAMATS.

2 Modeling

This section describes the process used to model the geometry of a building from a set of geo-referenced images using only simple edge highlighting by the user. The basic concept behind the modeling process is as follows; if one has two images of a scene taken from different locations, and the exact position and orientation of the camera is known for each image (i.e. the exterior orientation parameters $X_o, Y_o, Z_o, \Omega, \Phi$ and $K$) then the exact location of any point visible in both images can be determined. This is illustrated in figure 2.

The modeling process outlined in this section extends this idea by using triangle intersections to find edges rather than line intersections to find points. The modeling process can be split into three main steps; Edge Highlighting, Edge Recovery and Structure Recovery.
2.1 Edge Highlighting

Edge highlighting is the only manual step performed by the user in the modeling process. Primary lines and secondary lines are used to highlight edges in the images. Primary lines are used to recover the position of edges.
directly, determining the core structure of the model. They are responsible for the creation of every vertex in the final model. The endpoints of a primary line can be connected (having one or more primary or secondary lines sharing that endpoint) or unconnected (having no other lines sharing that endpoint). A secondary line is used to connect primary lines together and must have each of its endpoints connected to at least one primary line. In figure 3 the solid black lines represent primary lines while the black dashed lines represent secondary lines.

The reason the entire model is not defined by primary lines is because it is difficult to recover some edges given the input data. Primary lines are well suited to recovering the position of vertical edges because it is possible to create arbitrarily large angles of intersection about the vertical edge axis, as shown in figure 4. However, for horizontal edges near camera level it is not possible to create arbitrarily large intersection angles, making it difficult to recover the horizontal edges accurately since slight inaccuracies in the camera’s intrinsic or extrinsic properties results in large errors in estimated edge location, see figure 5.

Secondary lines work by connecting primary lines, where the use of a primary line would be prohibitive, e.g. the horizontal base line of the building in figure 5. Since the primary lines will recover the vertical edges of the building, the secondary lines simply indicate to the system that these edges should be connected without trying the same recovery technique used for the primary edges.

![Figure 5](image)

**Fig. 5.** For horizontal edges near camera level it is difficult to obtain arbitrarily large disparity angles.
Primary edges should be used to recover the core structure of the building, while defining as few edges as possible. Then secondary lines should be used to define all remaining edges. A primary edge must be highlighted in at least three images, although it can be advantageous to define a primary edge in more than three images when trying to recover edges that make poor primary edge candidates. Secondary edges need only be defined in a single image.

### 2.2 Edge Recovery

After the edges have been highlighted, six automated steps are performed to recover the final edges: Line Projection, Triangle Intersection, Correspondence Recovery, Edge Averaging, Vertex Merging, and Secondary Edge Recovery. Each of these steps is described next.

#### 2.2.1 Line Projection

The first step in determining the positions of the primary edges is to project the 2D primary lines to form 3D triangles. The intrinsic and extrinsic properties of the camera are used to project the primary lines from the cameras position, at the correct orientation out to infinity. This is performed for every primary line in each image, as shown in figure 6 for a scene consisting of 4 images, the final primary edges are highlighted in white.

**Fig. 6.** Projection of primary lines. Primary edges are highlighted in white
2.2.2 Triangle Intersection
Once every 2D primary line has been transformed to a 3D triangle, the next step is to determine the intersections between the triangles. Every triangle stores a list of the triangles it intersects.

2.2.3 Correspondence Recovery
Generally each triangle intersects many other triangles even though only a small number of the triangle intersections have both their parent lines highlighting the same edge. Most systems resolve this problem by performing manual correspondences between the lines so that lines which highlight the same edge are grouped together. Once the lines are converted to triangles the only valid intersections are between members of the same group. This can be a very time consuming process. SAMATS performs this correspondence automatically in three steps; Intersection Rating, Triangle Grouping and Group Merging.

2.2.3.1 Intersection Rating
Every triangle needs to rate each of the triangles it intersects. These ratings can then be used to determine which of the intersecting triangles represent the same primary edge as itself. A naïve approach would simply use the coverage of the line of intersection as the only measure in rating each intersecting triangle, with greater coverage resulting in a better rating. This has proved to be almost completely useless because often intersecting triangles which represent a different primary edge (invalid triangles) receive better rating than those that represent the same primary edge (valid triangles).

The automated rating process does not rate an intersecting triangle on the quality of the intersection line, but on the similarity of the intersection line with other intersection lines. This is the reason for having a 3 primary line minimum when highlighting each primary edge.

Figure 7 shows the basis of the rating algorithm in 2D. In the figure there are three cameras, A, B and C, there are two points being modeled, X and Y, and there are six lines, two from each camera through the points being modeled, A_X, A_Y, B_X, B_Y, C_X and C_Y. Each line intersects every other line (although some of the intersections are off image) even though the only valid intersections are those between lines with matching subscripts. One should note that the invalid intersections are spaced quite randomly apart while the valid intersection groups have three points of intersection coincident at one location. The automated rating algorithm uses this principle of valid intersections being grouped close together when calculating the rating for each intersecting triangle.
Fig. 7. 2D example of the automated correspondence determination concept

For each triangle $t$, we need to rate each of the triangles $t_i$ in $t$’s intersecting triangles set $T_i$. Since we know that there are at least three triangles per primary edge, we know that at least two of the intersecting triangles are valid matches. We call these valid intersecting triangles $t_{i1}$ and $t_{i2}$. Note that there may be more than two valid intersecting triangles, although that fact is not important at this stage. If $t_{i1}$ is a valid match with $t$, and $t_{i2}$ is a valid match with $t$, then $t_{i1}$ and $t_{i2}$ must be valid matches with each other. This implies that $t_{i2}$ would be an intersecting triangle of $t_{i1}$. Therefore, when determining the rating of any $t_i$, we only need to consider triangles that are in both $t_i$’s intersecting triangles set and $t$’s intersecting triangles set, i.e. $T_i \cap T$. Note that only a sub-set of this set will contain valid intersections.

The intersecting triangle $t_i$ can now be given a ranking based on the triangles in the set $T_i \cap T$. Each triangle $t_k$ in this set intersects both $t_i$ and $t$. Therefore, we can use the three intersecting lines, $l_{ik}$, $l_{ik}$, and $l_{ik}$, to give $t_k$ a
Appendix C: Publications

rating. Intersection lines are evaluated based on 3 properties; the distance between their midpoints, the relative orientations between them, and their difference in length.

The value returned for the distance between their midpoints is in the range [0…1] and is described by the following equation;

\[
\frac{1}{1 + (ScalingFactor \times Distance)^2}
\]

The \textit{ScalingFactor} is used to set the rate at which the value declines with respect to distance. This factor is dependent on the choice of units used to model the building, e.g. if the units are meters and we only want to consider intersection lines with roughly less that 10cm spacing, then setting \textit{ScalingFactor} to about 10 would give a good range. At 1cm the value returned by the equation would be 0.9, at 10cm the value would be 0.5, and at 100cm the value would be 0.09.

The value returned for the measure of the two lines relative orientations is calculated using the absolute value of the dot product between the lines' unit vectors and is also in the range [0…1]. For two lines \(A\) and \(B\) the equation is as follows;

\[
|\hat{A} \cdot \hat{B}|
\]

Finally, the value returned for their difference in length is in the range [0…1] and is described by the following equation;

\[
\frac{\max(A, B) - |A - B|}{\max(A, B)}
\]

Once all these partial tests have been performed, the final rating for the lines is simply the product of the three, which is also in the range [0…1]. Refer to figure 8 for an illustration of each test.

Every triangle \(t\) in the set \(T_i \cap T_j\) is given a rating based on the comparison of the three intersection lines \(l_{ij}, l_{ik}, l_{jk}\). There are three comparisons that can be made, \(l_{ij}\) with \(l_{ik}\), \(l_{ij}\) with \(l_{jk}\), and \(l_{ik}\) with \(l_{jk}\). The product of these three tests is used to determine the rating of each triangle \(t\) in the set \(T_i \cap T_j\). The product is used in favor of the sum in order to keep the ratings in the range [0…1].

Once every \(t\) has been given a rating there are three logical options for assigning a rating to \(t\). Assign \(t\) the weighted sum of all the ratings in the set \(T_i \cap T_j\). This has proved unfavorable since this would include triangles that are invalid. If there are a large number of low scoring invalid triangles in a particular \(T_i \cap T_j\) set, the \(t\) will be given a poor rating even if it is a valid triangle.

A second option would be to assign \(t\) the weighted product of the ratings. This is a poor choice for the same reasons as assigning the weighted sum, only the problem is amplified greatly when taking the product since there are almost always a few poor scoring invalid matches, this forces the \(t\) rating to zero, making the rating useless.
Appendix C: Publications

The option that was found to work best is to assign \( t_j \) the best rated \( t_i \) in the set \( T_i \cap T_j \). If \( t_j \) is a valid intersecting triangle for \( t_i \), the best rated triangle is almost always a valid intersecting triangle for both \( t_i \) and \( t_j \), which we'll refer to as \( t_k \) from here on. When storing the rating for each \( t_j \), a reference to the \( t_k \) triangle responsible for this rating is also stored. This triangle is required for the triangle grouping step described briefly next.

2.2.3.2 Triangle Grouping

After the intersection rating step, for every triangle \( t_i \), every triangle \( t_j \) in \( t_i \)'s intersecting triangles set \( T_i \) will have a rating assigned to it. Also, the \( t_k \) responsible for each \( t_j \)'s rating will be stored along with the rating. This information can then be used to group triangles together where each group represents a primary edge.

Essentially, the grouping process is performed in two steps. Firstly, the GSS (Group Scope Set) of each triangle is determined. The GSS for each triangle is the list of mutually high ranking intersecting triangles. Not every triangle will have the same size GSS. The size of these sets will vary depending on the number of triangles used to represent each primary edge as well as the relationship between their line intersections.

The second step in the grouping process is to use the GSSs to group the triangles into groups. The triangles are ordered based on the size of their GSS’s in ascending order. Triangles with small GSSs form the initial groups. Small GSSs are more tightly coupled which is a desirable property when trying to match triangles together. After the core set of groups is created all remaining triangles are assigned a group, the vast majority being assigned to one of the existing groups with only a small minority forming their own groups.

It may not be possible to assign every triangle to a group for a number of reasons. The user may not have used three primary lines to highlight a particular primary edge or there may be too great an error to group some primary lines together either due to an error in the camera’s intrinsic and/or extrinsic properties or an error in line placement. In such cases the triangles are marked as invalid. For a more detailed explanation of the Triangle Grouping step refer to [4]

2.2.3.3 Group Merging

The final step in the grouping process is group merging. If a primary edge is represented by 6 or more primary lines it may form 2 distinct groups. If the groups were left the way they were, there would be 2 primary edges representing the same building edge instead of just one. The merging step simply compares each group to each other and merges groups which are sufficiently similar.

2.2.4 Edge Averaging

Once all triangles have been assigned a group the primary edges must be determined for each group. This is simply the weighted average of all the intersection lines between all group members.
2.2.5 Vertex Merging
During the edge averaging step, each primary edge will be created totally independently from all other primary edges. In most cases this is acceptable since the majority of primary edges are not connected to any other primary edge. Sometimes however primary edges are connected. This is indicated in the edge highlighting step by having two or more primary lines share the same endpoint.
All primary edges that are connected need to have their connected endpoints coincident. This is achieved by creating a mapping between every primary line and every primary edge, and also between every primary line endpoint and every primary edge vertex. Once the mappings have been made, we can see if any of the primary lines share the same endpoints, which maps to primary edges sharing the same vertex. Once the vertices are identified they are set to the average of their positions.

2.2.6 Secondary Edge Recovery
Secondary edges are determined using the same mapping information obtained during the vertex merging step. Firstly, the secondary lines endpoints are determined. Then the corresponding vertices are determined for these endpoints and a new group is created for each secondary line using these vertices as the secondary edges endpoints. After all secondary edges have been highlighted the outline of the model should be complete.

2.3 Structure Recovery
Even though the outline of the model has been determined there is still no surface data associated with the model. The model is only defined in terms of vertices and lines and not in terms of surfaces and the triangles that make up each surface. Recovering this structural information is broken into three steps. The first step is to determine the models surfaces. This is achieved by treating the model like a graph, with the vertices as the graph nodes and the edges as the graph edges. Surfaces are determined by finding the shortest cycles in the graph where all the vertices are co-planar. All surface normals must then be aligned so that they all point away from the model. This is performed by aligning the normals of neighboring surfaces recursively until all normals are aligned. The final step is to triangulate all the surfaces. The algorithm used to triangulate each surface can be found in [8]. Refer to [4] for a more details.

3 Texture Extraction
Coming into this section, we have an accurate model of the building, or to be exact, we have a geometrically accurate model of the building. There is still data contained in the image set that has not yet been used to increase the models realism, the buildings façades. The texture extraction process takes the façades from the images and applies them to the model. An
Appendix C: Publications

overview of this component is presented next. For a more detailed explanation of the Texture Extraction component refer to [4].

3.1 Overview

The aim of the texture extraction process is to produce a 3D model with photorealistic textures. The texture extraction process can be broken into a number of steps. Firstly, the number of images that will contribute to each triangle is determined using back-face culling. There can be any number of contributing images, with each image’s contribution first being stored in a temporary texture before they are all blended together per-pixel based on the camera-surface distance and orientation. Occlusion maps are used to prevent incorrect façade data being stored with each triangle. All triangles are then packed into a single large texture retaining the relative size of each triangle, thus creating an authalic texture map. The texture coordinates for each triangle are then set to sample the correct region of the texture map, with the texture then being assigned to the model.

4 Conclusions

SAMATS shows that given sufficient information, user input to the modeling process can be reduced significantly. Currently user input is required for the edge highlighting step but since no correspondence is required this step could be automated using edge detection and a set of heuristics to guide the choice between using primary lines or secondary lines. Currently SAMATS has only been used on synthetic images where the exact extrinsic and intrinsic properties of the camera are known. Achieving such precision in the real world would prove difficult without specialized equipment. New techniques will be required to facilitate the gathering of the geo-referenced images required by SAMATS in order for this system to be utilized effectively in the real world.

SAMATS has shown the ability to model rectangular and triangular roofed structures very well, however SAMATS does have trouble modeling certain structures. SAMATS has no special ability to handle curved surfaces, which makes it impossible to model such features completely accurately. Cylindrical column must be replaced by rectangular columns for instance. Another difficulty that can arise is SAMATS’ inability to handle partially highlighted edges. This makes it difficult, and in some cases impossible, to model buildings in tightly confined spaces.

References

1. J.D. Carswell, A. Eustace, K. Gardiner, E. Kilfeather. An Environment for Mobile Context-Based Hypermedia Retrieval, in 13th International Conference on Database and
Appendix C: Publications

Expert Systems Applications (DEXA2002); IEEE CS Press; Aix en Provence, France; September 2002


Appendix C: Publications


**SAMATS - Semi-Automated Modeling And Texturing System (Triangle Grouping and Structure Recovery Explained)**

Joe Hegarty and James D. Carswell

Digital Media Centre, Dublin Institute of Technology, Aungier St., Dublin, Ireland
{[joe@dmc.dit.ie], [jcarswell@dit.ie]}

**Abstract.** The creation of detailed 3D buildings models, and to a greater extent the creation of entire city models, has become an area of considerable research over the last couple of decades. The accurate modeling of buildings has applications in entertainment, planning, tourism and e-commerce, to name just a few. Many modeling systems created to date require manual correspondences to be made across the image set in order to determine the models 3D structure. This paper describes SAMATS, a Semi-Automated Modeling And Texturing System, which has the capability of producing geometrically accurate and photorealistic building models without the need for manual correspondences, by using a set of geo-referenced terrestrial images. This paper focuses on the triangle grouping and structure recovery steps in the modeling component of the system, although an outline of the other components in the system is given.

1 **Introduction**

2D and 3D information visualization using VR modeling is becoming an important area of e-commerce research for today’s web-based location based services (LBS) applications. Examples of exploiting VR navigation for both cultural heritage and environmental applications can be found in [1,2,3]. However, producing visually convincing VR models for these LBS applications requires expert VR knowledge on the part of the system developers. This research investigates building reconstruction technology for creating geometrically accurate, photorealistic 3D models from terrestrial digital photography for use in LBS applications that non-expert VR developers can exploit. It is envisioned that the resulting 3D model output from this work be web-enabled and made available to subsequent LBS research endeavors (e.g. for archaeologists, town planners, tourism, e-Government, etc.). Being able to produce 3D VR building models using terrestrial imagery allows all users to exploit the future commercialization potential of web-based LBS.

[12] was the first to investigate the principle of structure from motion. [11] builds on these ideas using lines instead of points, although both require correspondences to be made manually across the image set. In fact the
majority of semi-automated reconstruction systems require the user to make manual correspondences across the image set in order to reconstruct a model. [5] is one of the most robust systems using this approach allows the user to create models using a set of block primitives and by setting constraints on these primitives. A more automated modeling approach involves the modeling of roofs using aerial imagery. Models produced in this way can produce structurally accurate models but fail to capture building façades accurately, although [7], [8], and [9] have looked into the merging of façade textures with models produced from aerial imagery. [4] constructs a large set of 3D building models by using spherical mosaics produced from accurately calibrated ground view cameras fitted with a GPS device. Although highly automated, this system was limited to modeling simple shaped buildings by simply identifying the rooflines and extruding walls downwards. [13] is an example of extracting building and window edges without the need for manual correspondence, although a rough model of the structure being modeled is required in order for this system to work.

SAMATS uses a novel approach to creating building models without the need for manual correspondences to be made. The ability of SAMATS to remove the manual correspondence step found in most modeling approaches is achieved by having all images geo-referenced in the same reference frame. However, the acquisition of geo-referenced terrestrial images is still a bottleneck that does not have a straightforward solution. Currently public GPS will give an absolute accuracy of between 1 to 10 meters using a single receiver. This resolution is not technology bound but information restriction bound, with military GPS and differential GPS offering centimeter accuracy. As private industries or other governments create their own satellite networks these restrictions may no longer apply - making the acquisition of accurate geo-referenced imagery as easy as regular imagery. SAMATS does not solve the difficulties in acquiring geo-referenced imagery - it only investigates the usefulness of such imagery in the overall modeling process.

\[Fig. 1.\] SAMATS system diagram. The highlighted steps are the focus of this paper.
This paper focuses on the triangle grouping and structure recovery steps in the modeling component of SAMATS, but does give an overview of the other components. For a detailed description of the other components refer to [6]. Figure 1 shows a systems overview of SAMATS.

2 Modeling

This section describes the process used to model the geometry of a building from a set of geo-referenced images using only simple edge highlighting by the user. The basic concept behind the modeling process is as follows; if one has two images of a scene taken from different locations, and the exact position and orientation of the camera is known for each image (i.e. the exterior orientation parameters $X_o, Y_o, Z_o, \Omega, \Phi, \Kappa$) then the exact location of any point visible in both images can be determined. This is illustrated in figure 2. The modeling process outlined in this section extends this idea by using triangle intersections to find edges rather than line intersections to find points. The modeling process can be split into three main steps; Edge Highlighting, Edge Recovery and Structure Recovery.

2.1 Edge Highlighting

Edge highlighting is the only manual step performed by the user in the modeling process. Primary lines and secondary lines are used to highlight edges in the images. Primary lines are used to recover the position of edges directly, determining the core structure of the model. They are responsible for the creation of every vertex in the final model. A secondary line is used to connect primary lines together and must have each of its endpoints connected to one or more primary lines. The reason the entire model is not defined by primary lines is because it is difficult to recover some edges given the input data. Primary lines are well suited to recovering the position of vertical edges because it is possible to create arbitrarily large angles of intersection about the vertical edge axis. However, for horizontal edges near camera level it is not possible to create
Appendix C: Publications

arbitrarily large intersection angles, making it difficult to recover the horizontal edges accurately since slight inaccuracies in the camera’s intrinsic or extrinsic properties results in large errors in estimated edge location. Secondary lines work by connecting primary lines, where the use of a primary line would be prohibitive. Since primary lines will generally be used to recover the vertical edges of a building, secondary lines should then be used to highlight the horizontal wall bases and roof tops, which indicates to the system that these edges should be connected without trying the same recovery technique used for the primary edges. A primary edge must be highlighted in at least three images, although it can be advantageous to define a primary edge in more than three images when trying to recover edges that are poor primary edge candidates. Secondary edges need only be defined in a single image.

2.2 Edge Recovery

After the edges have been highlighted, six automated steps are performed to recover the final edges; Line Projection, Triangle Intersection, Correspondence Recovery, Edge Averaging, Vertex Merging, and Secondary Edge Recovery. Each of these steps is described next.

2.2.1 Line Projection

The first step in determining the positions of the primary edges is to project the 2D primary lines to form 3D triangles. The intrinsic and extrinsic properties of the camera are used to project the primary lines from the cameras position, at the correct orientation out to infinity. This is performed for every primary line in each image.

2.2.2 Triangle Intersection

Once every 2D primary line has been transformed to a 3D triangle, the next step is to determine the intersections between the triangles. Every triangle stores a list of the triangles it intersects.

2.2.3 Correspondence Recovery

Generally each triangle intersects many other triangles even though only a small number of the triangle intersections have both their parent lines highlighting the same edge. Most systems resolve this problem by performing manual correspondences between the lines so that lines which highlight the same edge are grouped together. Once the lines are converted to triangles the only valid intersections are between members of the same group. This can be a very time consuming process. SAMATS performs this correspondence automatically in three steps; Intersection Rating, Triangle Grouping and Group Merging.


Appendix C: Publications

2.2.3.1 Intersection Rating

Every triangle needs to rate each of the triangles it intersects. These ratings can then be used to determine which of the intersecting triangles represent the same primary edge as itself. The automated rating process chosen uses the fact that there must be at least three primary lines, and hence triangles, for each primary edge. Each intersecting triangle is not rated on the coverage of the intersection line it makes, but rather on the similarity of its intersection line with others.

At the end of the intersection rating step, the list of intersecting triangles for each triangle will have a rating. Also, since the rating system is based on comparing intersection lines, a reference to the triangle responsible for the rating is also stored. For example, triangles \( t_i \), \( t_j \), and \( t_k \) all intersect each other. If \( t_j \) is the best rated intersecting triangle of \( t_i \), and it was a comparison between the intersection lines \( l_{ij} \), \( l_{ik} \), and \( l_{jk} \) which were responsible for this rating, then a reference to \( t_k \) will be stored along with this rating for \( t_j \) in \( t_i \)'s intersecting triangles list.

2.2.3.2 Triangle Grouping

After the intersection rating step, for every triangle \( t_i \), every intersecting triangle \( t_j \) will have a rating assigned to it. Also, the \( t_k \) responsible for each \( t_j \) rating will be stored along with the rating. This information can then be used to group triangles together, with each group representing a primary edge.

Essentially, the grouping process is performed in two steps. Firstly, the GSS (Group Scope Set) of each triangle is determined. The GSS for a triangle \( t_i \) is a list of triangles, which contains the triangle itself (in this case \( t_i \)), the GSS for \( t_j \) (the best rated \( t_j \)) and the GSS for \( t_k \) (the \( t_k \) for \( t_j \)). The GSS can only hold a single instance of any triangle. This ensures that the recursive algorithm is terminating. Not every triangle will have the same size GSS. The size of these sets will vary depending on the number of triangles used to represent each primary edge as well as the relationship between their intersection lines.

The simplest case arises when a primary edge is represented by three triangles. In this configuration each triangle \( t_i \) refers to the other two as either its \( t_j \) triangle or as its \( t_k \) triangle. In such a situation all three triangles have identical GSS containing the three triangles, see figure 3.

If there are more than three triangles representing a primary edge there can be three broad types of set configuration. One configuration involves four or more triangles that represent the same primary edge with every triangle having identical GSSs, see figure 4. Another configuration involves four or more triangles that represent the same primary edge but with only a subset of triangles having identical GSSs, while the other triangle(s) have GSSs containing the subset of triangles plus additional triangles. This results in the real group consisting of four or more triangles although the GSSs of some of the triangles will only have a subset of these triangles, see figure 5. The final configuration involves six or more triangles that represent the same primary
Appendix C: Publications

edge but with each triangle having one of two or more GSSs. In this configuration each group is solved independently and then the groups are merged as a post-process, see figure 6. Combination of the above configurations can also occur together.

![Fig. 3. Three triangles, all with the same GSS.](image)

The second step in the grouping process is to use the GSSs to group the triangles into groups. The grouping algorithm runs in two phases. In phase one only triangles that have three triangles in their GSSs are processed. Each triangle as well as its GSS members are assigned a new group. The first phase solves either fully or partially the configurations shown in figures 3, 5, and 6. At the end of this phase the majority of triangles will have been assigned a group. Only triangles which have a configuration similar to that shown in figure 4 or unreferenced triangles like those shown in figure 5 remain.

![Fig. 4. Four triangles, all with the same GSS.](image)
Fig. 5. Four triangles, three of which have the same GSS. The unreferenced triangle has a GSS containing all four triangles.

Fig. 6. Six triangles forming two separate GSSs. The black line represents the group after merging.

In phase two these remaining triangles are assigned a group. If a triangle refers to triangles in an existing group (figure 5), it is added to that group provided that its rating in this group is within some minimum threshold. If a triangle’s GSS has triangles which have not yet been grouped, a new group will be created for these triangles (figure 4). It may not be possible to assign a group to every triangle for a number of reasons. For example, the user may not have used three primary lines to highlight a particular primary edge. Also there may be too great an error to group some primary lines together either due to an error in the camera’s intrinsic and/or extrinsic properties or an error in primary line placement. In such cases these triangles are marked as invalid.

2.2.3.3 Group Merging
The final step in the grouping process is group merging. This is required because sometimes a primary edge may be represented by 6 or more triangles, which form 2 or more self-contained groups with no inter-group referencing (figure 6). If the groups were left the way they were, there
Appendix C: Publications

would be 2 primary edges representing the same building edge instead of just one. The merging step simply compares each group to each other group by first comparing the highest ranked members of each group to each other. If it is found that the ranking between these triangles is within some threshold, the algorithm goes on to test every combination of group members together to guarantee that they; a) all intersect, and b) the lowest ranking observed is within some minimum threshold. If these two criteria are met, the two groups are merged.

2.2.4 Edge Averaging
Once all triangles have been assigned a group the primary edges must be determined for each group. This is simply the weighted average of all the intersection lines between all group members.

2.2.5 Vertex Merging
During the edge averaging step, each primary edge will be created totally independently from all other primary edges. In most cases this is acceptable since the majority of primary edges are not connected to any other primary edge. Sometimes however primary edges are connected. This is indicated in the edge highlighting step by having two or more primary lines share the same endpoint.

All primary edges that are connected need to have their connected endpoints coincident. This is achieved by creating a mapping between every primary line and every primary edge, and also between every primary line endpoint and every primary edge vertex. Once the mappings have been made, we can see if any of the primary lines share the same endpoints, which maps to primary edges sharing the same vertex. Once the vertices are identified they are set to the average of their positions.

2.2.6 Secondary Edge Recovery
Secondary edges are determined using the same mapping information obtained during the vertex merging step. Firstly, the secondary lines endpoints are determined. Then the corresponding vertices are determined for these endpoints and a new group is created for each secondary line using these vertices as the secondary edges endpoints. The outline of the model should be complete.

2.3 Structure Recovery

Even though the outline of the model has been determined there is still no surface data associated with the model. The model is only defined in terms of vertices and lines, and not in terms of surfaces and the triangles that make up each surface. Recovering this structural information is broken into three steps; Surface Determination, Surface Aligning, and Surface Triangulation.
2.3.1 Surface Determination
Surfaces are determined by treating the model as a graph, with the models vertices representing nodes in the graph and the primary and secondary edges representing the edges in the graph. Each surface corresponds to a cycle in the graph, but not every cycle in the graph corresponds to a surface, as illustrated in figure 7.

![Fig. 7](image)

*Fig. 7.* The black outlines represent cycles in the graph. One of the cycles represents a surface (2-3-8-7), while the other does not.

There are two main assumptions made in order to determine the surfaces from the vertices and edges; the model must be closed and the number of surfaces associated with each vertex is equal to the number of edges connected to it. Surfaces are then determined by finding the shortest cycles in the graph where all the vertices are co-planar.

2.3.2 Surface Aligning
Once all the models surfaces have been determined, the normal vector for each surface must be determined. The first step is to determine the adjacency of the surfaces, i.e. which surfaces are adjacent to each other. This is performed because surfaces are aligned in pairs. Once the surface adjacencies have been determined one of the surfaces is flagged as the master surface, while all other surfaces are flagged as slave surfaces. Firstly, all slave surfaces that are adjacent to the master are aligned, becoming themselves masters in the process, then all slave surfaces adjacent to these new master surfaces are aligned, becoming masters themselves. The process continues recursively until all surfaces have been flagged as masters. The aligning step uses the fact that adjacent surface pairs are attached along one of their edges. This edge can act like a hinge between the two surfaces making it possible to rotate one of the surfaces about this hinge so that the two surfaces are co-planar. If then the surfaces are transformed so that they are perpendicular with the z-axis with the hinge between them aligned with the x-axis, we notice that the interior of one surface is above the hinge while the interior of the other surface is below the hinge.
Using this fact each surface pair is aligned by transforming both the master surface and the slave surface so that their surface normals are aligned with the z-axis and the edge vector between them is aligned with the x-axis. Then each surface is checked to see if its interior is above or below the hinge edge. If both surfaces are on the same side of the hinge edge they are misaligned so the normal of the slave surface is flipped. If the two surfaces are on opposite sides, the two surfaces are already aligned, see figures 8 and 9.

Even though the models surfaces have been determined at this stage there maybe a serious problem with the models normals, they may all be pointing inwards instead of outwards. This is due to the fact that a random surface was chosen as the master surface at the beginning of the surface aligning step but it was not determined whether or not this normal points inwards or outwards. Luckily this is not a serious problem since all we have to do to rectify the situation is flip all the surface normals.
2.3.3 Surface Triangulation
Once each surface has been determined and aligned, each surface must be decomposed into triangles. The surfaces in the model can be either convex or concave although the surfaces should not contain holes. There are many factors that can be used to determine how a surface should be decomposed; minimize the number of triangles created, try to keep all triangles equilateral, try to keep all triangles close to equal area. The algorithm used to triangulate each surface can be found in [10]. This algorithm does not take any of these factors into consideration however. Firstly, each surface is orientated so that it is perpendicular with the z-axis. The z-coordinate is then ignored and the triangulation process treats the surface as if it was a 2D surface.

3 Texture Extraction
Coming into this section, we have an accurate model of the building, or to be exact, we have a geometrically accurate model of the building. There is still data contained in the image set that has not yet been used to increase the models realism, the buildings façades. The texture extraction process takes the façades from the images and applies them to the model. An overview of this component is presented next. For a more detailed explanation of the Texture Extraction component refer to [6].

3.1 Overview
The aim of the texture extraction process is to produce a 3D model with photorealistic textures. The texture extraction process can be broken into a number of steps. Firstly, all the triangles are packed into a single large texture retaining the relative size of each triangle, thus creating an authalic texture map. The texture coordinates for each triangle are then set to sample the correct region of the texture map. The number of images that will contribute to each triangle is then determined using back-face culling. There can be any number of contributing images, with each image’s contribution first being stored in a temporary texture before they are all blended together. Occlusion maps are used to prevent incorrect façade data being stored with each triangle. Each triangles final texture is then written out to the large texture map, which is then assigned to the model.

4 Conclusions
SAMATS shows that given sufficient information, user input to the modeling process can be reduced significantly. User input is required for the edge highlighting step but since no correspondence is required this step could be automated using edge detection and a set of heuristics to guide the choice between using primary lines or secondary lines. Testing SAMATS
Appendix C: Publications

with real world imagery is also an area that requires more investigation. Currently the system works reliably for synthetic images with the exact extrinsic and intrinsic properties of the camera known. Achieving such precision in the real world has proven difficult without specialized equipment. New techniques will be required to facilitate the gathering of the geo-referenced images required by SAMATS in order for this system to be utilized effectively outside laboratory conditions.

References

1. J.D. Carswell, A. Eustace, K. Gardiner, E. Kilfeather. An Environment for Mobile Context-Based Hypermedia Retrieval, in 13th International Conference on Database and Expert Systems Applications (DEXA2002); IEEE CS Press; Aix en Provence, France; September 2002
2. J.D. Carswell, M. Bertolotto, N. Mandrak. Applications of Mobile Computing for Fish Species at Risk Management: in Proceedings of International Conference on Environmental Informatics of International Society of Environmental Information Sciences (ISEIS2004); Regina, Canada; August 2004