GPU RAY TRACED RENDERING AND IMAGE FUSION BASED VISUALIZATION OF URBAN TERRAIN FOR ENHANCED SITUATION AWARENESS

by
LING LING SIK

M.S. University of Central Florida, 2010

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the College of Graduate Studies at the University of Central Florida Orlando, Florida

Spring Term
2013

Major Professor:
Pattanaik Sumanta
ABSTRACT

Urban activities involving planning, preparing for and responding to time critical situations often demands sound situational awareness of overall settings. Decision makers, who are tasked to respond effectively to emergencies, must be equipped with information on the details of what is happening, and must stay informed with updates as the event unfolds and remain attentive to the extent of impact the dynamics of the surrounding settings might have. Recent increases in the volumes of geo-spatial data such as satellite imageries, elevation maps, street-level photographs and real-time imageries from remote sensory devices affect the way decision makers make assessments in time-critical situations. When terrain related spatial information are presented accurately, timely, and are augmented with terrain analysis such as viewshed computations, enhanced situational understanding could be formed. Painting such enhanced situational pictures, however, demands efficient techniques to process and present volumes of geo-spatial data. Modern Graphics Processing Units (GPUs) have opened up a wide field of applications far beyond processing millions of polygons. This dissertation presents approaches that harness graphics rendering techniques and GPU programmability to visualize urban terrain with accuracy, viewshed analysis and real-time imageries. The GPU ray tracing and image fusion visualization techniques presented herein have the potential to aid in achieving enhanced urban situational awareness and understanding.

Current state of the art polygon based terrain representations often use coarse representations for terrain features of less importance to improve rendering rate. This results in reduced geometrical accuracy for selective terrain features that are considered less critical to the visualization or simulation needs. Alternatively, to render highly accurate urban terrain, considerable computational effort is needed. A compromise between achieving real-time rendering rate and
accurate terrain representations would have to be made. Likewise, computational tasks involved in terrain-related calculations such as viewshed analysis are highly computational intensive and are traditionally performed at a non-interactive rate. The first contribution of the research involves using GPU ray tracing, a rendering approach, conventionally not employed in the simulation community in favor of rasterization, to achieve accurate visualization and improved understanding of urban terrain. The efficiency of using GPU ray tracing is demonstrated in two areas, namely, in depicting complex, large scale terrain and in visualizing viewshed terrain effects at interactive rate.

Another contribution entails designing a novel approach to create an efficient and real-time mapping system. The solution achieves updating and visualizing terrain textures using 2D geo-referenced imageries for enhanced situational awareness. Fusing myriad of multi-view 2D inputs spatially for a complex 3D urban scene typically involves a large number of computationally demanding tasks such as image registrations, mosaickings and texture mapping. Current state of the art solutions essentially belongs to two groups. Each strives to either provide near real-time situational pictures in 2D or off-line complex 3D reconstructions for subsequent usages. The solution proposed in this research relies on using prior constructed synthetic terrains as backdrops to be updated with real-time geo-referenced images. The solution achieves speed in fusing information in 3D. Mapping geo-referenced images spatially in 3D puts them into context. It aids in conveying spatial relationships among the data. Prototypes to evaluate the effectiveness of the aforementioned techniques are also implemented. The benefits of augmenting situational displays with viewshed analysis and real-time geo-referenced images in relation to enhancing the user's situational awareness are also evaluated. Preliminary results
from user evaluation studies demonstrate the usefulness of the techniques in enhancing operators' performances, in relation to situational awareness and understanding.
I would like to dedicate this dissertation work to my family, for their love and supports, and for inculcating in me the value of learning, and for shaping me in my early life to have the strength and persistence to pursue my dreams.
ACKNOWLEDGMENTS

My greatest thanks go to Dr Pattanaik for all the guidance and supports that he has given me while working on the research. I am grateful for the time he spent in discussing ideas and revising papers. I have learnt so much through my journey of discovery in the world of computer graphics programming. My thanks also goes to Dr Kincaid for the initial encouragement for me to embark on this journey. Also utmost appreciations to Dr Proctor and Dr Graniela for their insightful comments on my research work, and useful advice and endless patience in improving my writing. Also thanks to Dr Tappen for serving on my dissertation committee. The multi-discipline skills and experiences I gained would be valuable beyond this thesis as I continue my journey, be it in the world of research, computer graphics and vision, and modeling and simulation, and even other domains. Last but not at all least, I would like to acknowledge helps and supports rendered by special friends, relatives, and acquaintances who offer words of encouragements and useful advice, and who nudge me forward whenever the journey seems insurmountable.
# TABLE OF CONTENTS

LIST OF FIGURES ......................................................................................................................... xv

LIST OF TABLES ............................................................................................................................. xxvi

CHAPTER 1 INTRODUCTION .............................................................................................................. 1

1.1 Decision Making In Urban Settings ......................................................................................... 1

1.2 Synthetic Urban Terrain For Modeling And Simulation (M&S) And Command And Control (C2) .......................................................................................................................... 3

1.3 3D Urban Terrain As Backdrop For Enhanced Situational Picture ....................................... 4

1.4 Terrain Viewshed Analysis To Aid In Situational Understanding ........................................ 6

1.5 Fusing 2D Images And 3D Terrain .......................................................................................... 7

1.6 Research Objective .................................................................................................................. 9

1.7 Overview ................................................................................................................................ 10

CHAPTER 2 BACKGROUND AND PREVIOUS WORKS .................................................................. 12

2.1 Situational Awareness .............................................................................................................. 12

2.1.1 Using Information Visualization To Aid In Decision Making ........................................ 13

2.1.2 Measuring Situational Awareness ..................................................................................... 16

2.1.3 Situational Awareness For Urban Scenes ......................................................................... 17
2.2 Programmable Graphics Pipeline ................................................................. 20

2.3 Rendering Virtual Scene ............................................................................. 22

2.4 Visualizing Synthetic Urban Terrain ......................................................... 24
   2.4.1 Generating Urban Terrain ...................................................................... 26
   2.4.2 Representing Terrain Relief ................................................................. 29
   2.4.3 Scene Complexity Management ........................................................... 31
   2.4.4 Image Base Rendering .......................................................................... 35

2.5 Accelerated Ray Casting / Tracing Techniques ........................................... 43
   2.5.1 Spatial Database ................................................................................... 44
   2.5.2 Ray Traversal Algorithm ...................................................................... 46
   2.5.3 Space Leaping ....................................................................................... 48
   2.5.4 Maximum Mipmaps ............................................................................. 50

2.6 Fusing 2D Images And 3D Terrain ............................................................ 51
   2.6.1 Texture Mapping For Terrain Generation And Visualization .............. 53
   2.6.2 Image Mosaicking ................................................................................. 55
   2.6.3 3D Reconstruction From 2D Images ................................................... 56
   2.6.4 Transformations Between 3D Geometries And 2D Image Spaces .......... 57
2.6.5 Mapping Texture Onto Terrain Geometries .............................................. 59
2.6.6 Leveraging On Computer Graphics Techniques For Terrain Texture Mapping .... 61
2.7 Summary .............................................................................................................. 66

CHAPTER 3 RAY TRACING FOR ACCURATE VISUALIZATION AND IMPROVED UNDERSTANDING OF URBAN TERRAIN .............................................................................. 68
3.1 Enhancing Ray-casted BlockMap For Accurate Urban Terrain Representation .... 69
3.1.1 Using BlockMap For Interactive Simulation.................................................... 70
3.1.2 Providing Depth Information For BlockMap .................................................... 75
3.2 A More Flexible Approach For BlockMap Implementation................................. 80
3.2.1 Optimizing BlockMap Texture Usage .............................................................. 81
3.2.2 Improving Rendering Performance ................................................................. 84
3.2.3 Encoding Wall Depth Information ................................................................. 88
3.3 Large Scale Urban Terrain Prototype Implementation .......................................... 88
3.3.1 Enhanced BlockMap Implementation For Distant City Block ......................... 90
3.3.2 BlockMap Representation For Individual Building ........................................ 99
3.3.3 Terrain Prototype Implementation ............................................................... 102
3.4 Accelerated Ray Casting Process For Enhanced BlockMap Rendering ............... 111
3.4.1 Accelerating Ray Traversal ................................................................................................. 112
3.4.2 Enhanced Wall Indexing System For Ray Casting ................................................................. 114
3.4.3 Spacing Leaping For Steep Viewing Rays ............................................................................. 117
3.4.4 Spacing Leaping For Oblique Viewing Rays ......................................................................... 120
3.4.5 Adopting Maximum Mipmap Structure For BlockMap Ray Casting ................................. 123
3.5 Ray Tracing For Improved Understanding Of Urban Terrain Using Viewshed Analysis
And Rendering .................................................................................................................................. 125
3.5.1 GPU-Accelerated Viewshed Analysis ...................................................................................... 126
3.5.2 Performance Evaluation ........................................................................................................ 129
3.5.3 Evaluating Effectiveness Of Displays Enhanced With Viewshed Analysis ........................ 131
3.6 Summary .................................................................................................................................. 147

CHAPTER 4 FUSING 2D GEO-REFERENCED IMAGES AND 3D TERRAIN .......................... 151
4.1 Geo-Spatial Information .......................................................................................................... 153
4.1.1 Static Context ....................................................................................................................... 153
4.1.2 Dynamic Context .................................................................................................................. 155
4.2 Fusing 2D Images And 3D Terrain Geometries ..................................................................... 156
4.2.1 Generating A Centralized Texturing System For 3D Geometries ........................................ 158
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.2</td>
<td>Performing 2D to 3D Transformation From Image To Scene Spaces</td>
<td>163</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Relating Information From 2D display to Relevant Geometries</td>
<td>164</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Mapping 2D Image Portions To Respective 3D Geometries</td>
<td>167</td>
</tr>
<tr>
<td>4.3</td>
<td>Prototype Implementations</td>
<td>170</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Texture Mapping of Building Facades Using Geo-Referenced Images</td>
<td>170</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Real-Time Fusion Of Surveillance Imageries In Urban Scenes</td>
<td>177</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Fusing Spatial Information From Disparate Sources</td>
<td>185</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance Evaluation</td>
<td>194</td>
</tr>
<tr>
<td>4.5</td>
<td>Evaluating Effectiveness Of The 2D Displays Enhanced With Fused 2D Map And Video Feeds</td>
<td>196</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Scenario Background</td>
<td>196</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Performance Based Assessment Of The 2D Displays Enhanced With Fused Information Using Experimental Study</td>
<td>197</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Knowledge Based Assessment Of The 2D Displays Enhanced With Fused Information Using Survey Questionnaires</td>
<td>209</td>
</tr>
<tr>
<td>4.6</td>
<td>Summary</td>
<td>215</td>
</tr>
</tbody>
</table>

CHAPTER 5 CONCLUSION............................................................................................................. 217
5.1 Contributions And Lessons Learnt

5.1.1 Accurate Visualization And Improved Understanding Of Urban Terrain

5.1.2 Real-time Fusion Of 2D Images And 3D Terrain

5.1.3 Lessons Learnt

5.2 Research Limitations

5.2.1 Limitations Of BlockMap Representation

5.2.2 Using Real World Data For Information Fusion

5.2.3 Integrating Proposed Techniques With Established Terrain Generation And Visualization Systems

5.2.4 Subject Matter Experts' And Other Inputs For The Comparative User Studies

5.3 Potential Enhancements And Future Works

5.3.1 Integration Of Proposed Solutions As Part Of Large Scale Visualization System

5.3.2 Improvements To The Enhanced BlockMap Representation

5.3.3 Combining BlockMap Rendering With Other Rendering Techniques To Visualize Terrains Of Other Types

5.3.4 Validating BlockMap LOD Integrated Terrain

5.3.5 GPU Ray Tracing For Terrain Analysis
5.3.6 Enhancements For The Fusion Technique ........................................... 239

5.3.7 Potential Use Of The Proposed Fusion Technique In Other Applications .... 240

5.4 Summary .......................................................................................................................... 242

APPENDIX IRB APPROVAL LETTERS ........................................................................... 243

REFERENCES ......................................................................................................................... 247
LIST OF FIGURES

Figure 2-1. Endsley’s Model Of Situation Awareness (Endsley, 1995) ........................................ 13

Figure 2-2. Theoretical View Of Visualization ........................................................................... 15

Figure 2-3. The Visualization Discovery Process (Johnson, Moorhead, Munzner, Pfister, Rheingans, & Yoo, 2005) ................................................................................................................ 16

Figure 2-4. Knowledge-Based And Performance-Based Measurement Techniques ................. 17

Figure 2-5. GPU (NVIDIA) Versus CPU (Intel) Peak Single Precision Floating Point Performance (Skinner and Hutchinson, 2010) ...................................................................................................... 20

Figure 2-6. The Graphics Pipeline ............................................................................................... 21

Figure 2-7. Visualizing the Graphics Pipeline ............................................................................. 21

Figure 2-8. The Programmable Graphics Pipeline .................................................................... 22

Figure 2-9. Rasterization (Left) And Ray Tracing (Right)(Skinner & Hutchinson, 2010) ........ 23

Figure 2-10. Terrain Dataset (From left, Satellite Imageries, Elevation Data And Building Footprints) ........................................................................................................................................ 29

Figure 2-11. From Left, Regular Height Posts, Triangulated Irregular Network (TIN), Right-Triangulated Irregular Network (RTIN) (Bartoň, 2011) ............................................................................................ 30

Figure 2-12. The Five Levels-of-Detail (LOD) Defined By CityGML (Kolbe, Gröger, & Plümer, 2005) ................................................................................................................................................. 32
Figure 2-13. A Typical Scene Graph ................................................................. 34

Figure 2-14. Texture Mapping Onto 3D Geometries ............................................ 36

Figure 2-15. Relief Mapping Using Ray Casting (Policarpo, Oliveira, & Comba, 2005) ........ 38

Figure 2-16. (a) Bump Mapping, (b) Parallax Mapping And (c) Relief Mapping (Policarpo, Oliveira, & Comba, 2005) ........................................................................ 38

Figure 2-17. City Block Rendered Using Polygons (Left) And BlockMap (Right) ............ 40

Figure 2-18. BlockMap (Left) Storage Texture Format And (Right) Rendered Representation Using GPU Ray Casting ................................................................. 41

Figure 2-19. BlockMap GPU Ray-Casting Process ................................................. 42

Figure 2-20. Rendering (Right) Of Height Field (Left) Of The Grand Canyon (Henning & Stephenson, 2004) ................................................................. 44

Figure 2-21. Ray Casting Of The Height Field (Henning & Stephenson, 2004) ............. 45

Figure 2-22. Octree (Top) and Quadtree (Bottom) .................................................. 46

Figure 2-23. The Bresenham Line-Drawing Algorithm ........................................... 48

Figure 2-24. Height Map (Left) And Corresponding 2D Distance Map (Right) ............. 49

Figure 2-25. Space Leaping Using Sphere Tracing .................................................. 49

Figure 2-26. Maximum MipMap Pre-Computation (Tevs, Ihrke, & Seidel, 2008) ........... 50
Figure 2-27. Ray-Height Field Intersection Using The Mipmap Hierarchy Structure (Tevs, Ihrke, & Seidel, 2008)........................................................................................................................................... 51
Figure 2-28. Captured Image (Left) To Be Mapped Onto Geo-Specific Building Geometry (Right)...................................................................................................................................................... 54
Figure 2-29. Extracting Texture Portions (Left) And Rectifying And Mapping Them Onto Polygonal Surfaces (Right)................................................................................................................................................... 54
Figure 2-30. Some Common Geometric Transformations ................................................................................................................................................................................ 58
Figure 2-31. UAV Camera Viewing Frustums (Left) And Rectified Orthogonal Images (Right)60
Figure 2-32. UAV Oblique Camera Viewing Frustum............................................................................................................................................................................ 60
Figure 2-33. Occlusion Handling To Rendering Video Images From A Virtual Viewpoint ...... 63
Figure 2-34. Overall Rendering Pipeline (Owens, Sycara, & Scerri, 2009)............................ 65
Figure 3-1. Rendering Urban Scene With Substantial Polygons Results In Low Refresh Rate... 71
Figure 3-2. Simplification Of Distant 3D Models For Real-Time Rendering To Improve Rendering Performance ................................................................................................................................. 72
Figure 3-3. Terrains Represented Without 3D Details (Top), With Full Geometrical Details (Middle) And With LODs Integrated (Bottom) Populated With Red Spheres................................. 74
Figure 3-4. Visual Effect When Polygonal Spheres (Left) Is Planted Within A City Block Rendered Using Standard BlockMap (Right)........................................................................................................ 76
Figure 3-5. Rendering BlockMap With A Sphere In Scene .......................................................... 77

Figure 3-6. Rendering Objects With BlockMap With Depth Correction ................................. 78

Figure 3-7. Rendering Spheres And BlockMap Terrain Tile Representation In A Common Spatial Space Without (Left) And With (Right) The Proposed Depth Correction Implementation ............................................................................................................................... 79

Figure 3-8. Terrains Represented With BlockMaps For Distant City Blocks Populated With Red Spheres ........................................................................................................................................................................ 79

Figure 3-9. Top View Of A Terrain Populated With A Cluster Of Geo-Typical Buildings (Left) And Corresponding Terrain Scene Graph (Right) ................................................................................................................................. 82

Figure 3-10. BlockMaps For Terrain Divided Into n Tiles (Left) And Individual Buildings Sharing A Common Reference Model (Right) ...................................................................................................................... 82

Figure 3-11. Scene Graph For BlockMap Representation Using Standard BlockMap For n City Blocks (Left) And Using References To The Same Copy Of BlockMap Texture For The Geo-Typical Buildings (Right) ........................................................................................................................................ 83

Figure 3-12. Scene Graph With Multiple Instances Of Similar Building Using Polygon Based Representation For High LOD And BlockMap For Low LOD representation .................................................. 84

Figure 3-13. Approximate Rendering Complexity Versus Distance From The Camera For Polygon Based And BlockMap Representation Of 3D Object ........................................................................................................... 85

Figure 3-14. Different Switching Distances For Buildings With Different Geometrical Complexity (Left) And Screen Footprint (Right) .................................................................................................................................... 85
Figure 3-15. Prototype Terrain Populated With Buildings Of Varying Polygonal Complexities Represented Using Full Polygon Based (Left) And Potential Distant City Blocks In BlockMaps (Right) ................................................................. 86

Figure 3-16. Prototype Terrain Adopting Selective BlockMap Representations For Polygonal Demanding Buildings .................................................................................................................................................. 87

Figure 3-17. Lowest (Top), Medium (Middle) And Highest (Bottom) LODs Of A Modeled Terrain Tile .................................................................................................................................................. 89

Figure 3-18. Enhanced BlockMap Data Structure Encoded in a 1024×128 24-bit RGB Texture 90

Figure 3-19. Valid But Unregistered Wall Texture Colors In Standard BlockMap Due To Sampling Limitations .................................................................................................................................................. 92

Figure 3-20. Multiple Sampling Strategy ........................................................................................................................................................................................................................................... 94

Figure 3-21. Construction Of Enhanced BlockMap Texture .................................................................................................................................................................................................................. 95

Figure 3-22. Filling In Wall Texture Information For Enhanced BlockMap ........................................................................................................................................................................... 97

Figure 3-23. Before (Top) And After (Bottom) Filling Up Empty Wall Texture Texels ............................................................................................................................................. 98

Figure 3-24. City Blocks In Polygons (Left) And BlockMap (Right) .................................................................................................................................................................................................................. 99

Figure 3-25. BlockMap Data Structure For A Single Complex Building .................................................................................................................................................................................................................. 100

Figure 3-26. BlockMap Rendering For Individual Complex Building .................................................................................................................................................................................................................. 101
Figure 3-27. Geometrical Representation (Left) And BlockMap Representations For Far Away (Middle) And Moderate Distant (Right) Viewing ................................................................. 102

Figure 3-28. LOD Node In OpenFlight File .............................................................................. 103

Figure 3-29. Boston Terrain Prototype Spanning Over An Area Of 10x10km Divided Into 1600 250×250m Tiles ..................................................................................................................... 104

Figure 3-30. Viewing Geometrically Complex Buildings At Distances More Than And Less Than 1 km .............................................................................................................................. 106

Figure 3-31. Terrain Prototype Rendered In Line (Top) And Fill (Bottom) Modes .............. 107

Figure 3-32. Views Of The Terrain Using Different Camera Viewing Heights (In Meters) .... 108

Figure 3-33. Performance Comparison Using Average Frame Rate ...................................... 109

Figure 3-34. Amount Of Occlusions Of Randomly Placed Spheres ...................................... 110

Figure 3-35. Determine Traversed Grids Using Bresenham's Line Algorithm (Left) And Using Fix Step Size (Right) ........................................................................................................... 113

Figure 3-36. Rendering Performance Comparing Bresenham's Line Algorithm And Ray Traversal Using Fix Step Size ...................................................................................................... 114

Figure 3-37. Top View (Left) And Visual Effect (Right) Of The BlockMap Ray Casting Process Using Fix Step Size ........................................................................................................... 115

Figure 3-38. Ray Casting Process Using Wall Index Maps Before (Left) And After (Right) Extending Wall Index Cells To Neighboring Units ...................................................................................... 116
Figure 3-39. Wall Index Maps (Upper Row) And Visual Rendering Effects (Lower Row) Before (Left Column) And After (Right Column) Extending Wall Index Cells To Neighboring Units 117

Figure 3-40. Leap Distance Calculations For Steep Viewing Rays .......................................................... 118

Figure 3-41. Applying Leap Distance For Steep Viewing Rays To Slightly Oblique Viewing Rays ......................................................................................................................... 119

Figure 3-42. Visual Effects Before (Left) And After (Right) Adding Intermediate Heights Comparison Checks .......................................................................................................................... 120

Figure 3-43. Space Leaping Calculations For Oblique Viewing Rays With Hit Points Located On BlockMap Ceiling Surfaces ................................................................................................................ 121

Figure 3-44. Space Leaping Calculations For Oblique Viewing Rays With Start Point At BlockMap Side Boundaries ................................................................................................................ 122

Figure 3-45. Rendering Performance With Space Leaping ................................................................................ 123

Figure 3-46. Rendering Performance Using Maximum MipMaps Acceleration Structure ...... 124

Figure 3-47. Combined Fix Step And Space Leaping With Maximum Mipmaps ......................... 125

Figure 3-48. Terrain Height Map (Left) And Polygon-Based Terrain (Right)................................. 127

Figure 3-49. Rendering Viewshed Using Rasterization And Ray Tracing ........................................... 127

Figure 3-50. Rasterization And Ray Tracing For Viewshed Analysis .............................................. 129

Figure 3-51. GPU-Accelerated Viewshed Analysis ................................................................................ 130
Figure 3-52. Frame Rates Of Scene Rendered With Different Number of Viewsheds .......... 131

Figure 3-53. Urban Scene Populated With Ground Vehicles, Surveillance And Firing UAVs 134

Figure 3-54. Visual Display Of Tank As Ground Vehicle .............................................. 134

Figure 3-55. User Interface For Non-Enhanced Display .............................................. 136

Figure 3-56. User Interface For Enhanced Display With Viewshed Analysis ................. 137

Figure 3-57. Performance Assessment For Non-enhanced And Enhanced Displays Presented in Graphic Forms ........................................................................................................ 139

Figure 3-58. Imagery Information For Baseline (Top) And Enhanced (Bottom) Displays To Be Presented To Participants ................................................................. 144

Figure 3-59. Participants' Rating On The Informativeness Of The Enhanced Display ........ 147

Figure 4-1. Proposed Solution ...................................................................................... 158

Figure 4-2. Texture mapping using UV coordinates. ..................................................... 159

Figure 4-3. Creating Terrain Texture Atlas Using The Guillotine Split Placement Algorithm . 162

Figure 4-4. Sub-textures Allocation Using Guillotine Split Placement Algorithm ......... 162

Figure 4-5. Updating Texture Information ..................................................................... 165

Figure 4-6. A GPU-Accelerated Technique For Spatial Mapping .................................. 168

Figure 4-7. A GPU-Accelerated Technique For Spatial Mapping Of Multiple Images ........ 169
Figure 4-8. High Resolution 3D UCF Model ................................................................. 171

Figure 4-9. Screen Captures Of The UCF 3D Model ...................................................... 171

Figure 4-10. Corresponding Texture Atlas (Left) And Given 3D Model To Be Used As "Base-
Skeleton" (Right) .............................................................................................................. 172

Figure 4-11. Texture U And V Information For Respective Screen Captures Of The 3D Building
Model.......................................................................................................................................... 172

Figure 4-12. Updated Building Texture And Resultant 3D Textured Model ......................... 173

Figure 4-13. Terrain Texture Section With Unfilled Texels.................................................. 174

Figure 4-14. Pyramid Of Textures.......................................................................................... 175

Figure 4-15. Combined Texture Atlas Using Pyramid Of Different Resolution Textures ....... 176

Figure 4-16. Updated Building Texture And Resultant 3D Textured Model ......................... 177

Figure 4-17. Heavily Populated Urban Terrain ........................................................................ 178

Figure 4-18. Fusing Acquired Imageries From Static Sensors ............................................. 179

Figure 4-19. Fusing Acquired Imageries From Mobile Sensors ............................................ 180

Figure 4-20. Acquired Images (Top) And Corresponding Texture UV Information (Bottom) For
Terrain Skin Only ...................................................................................................................... 183

Figure 4-21. Identified SIFT Keypoints From The Second Image From Figure 4-20 Before (Left)
And After (Right) Filtering Irrelevant Terrain Data.............................................................. 184
Figure 4-22. Updated Textured 3D Urban Terrain For Only Terrain Skin .......................... 185

Figure 4-23. Fusing 2D Map And Images. ..................................................................... 187

Figure 4-24. Acquired Surveillance Image (left) To Be Fused With A Reference Terrain Map (Right) ................................................................................................................................. 188

Figure 4-25. Challenges Faced By Operators To Identify Locations Of Detected Targets On Reference Map ................................................................................................................................. 189

Figure 4-26. Pairs Of Acquired (Top) And UV Rendered Images (Bottom) ....................... 191

Figure 4-27. Combining Acquired And Background Imageries Using A GPU-Accelerated Technique ................................................................................................................................. 192

Figure 4-28. Enhanced Situational Picture With Real-Time Imageries ............................... 194

Figure 4-29. Performance Improvement Using GPU Accelerated Approach ...................... 195

Figure 4-30. Augmenting Multi-UAVs Surveillance Mission Using Fused Information ......... 197

Figure 4-31. Urban Scene Populated With Ground Vehicles And Surveillance UAVs ........... 200

Figure 4-32. Sensor Footprints Casted On 2D Map .............................................................. 201

Figure 4-33. Ground Vehicles ........................................................................................ 202

Figure 4-34. User Interface Applications Implementation .................................................. 203

Figure 4-35. Baseline (Left) And Enhanced (Right) Situational Displays ......................... 204
Figure 4-36. Enhanced Situational Display With Superimposed Video Imageries. 205

Figure 4-37. Interface Display For Experimental Study. 206

Figure 4-38. Performance Assessment For Enhanced Display In Graphic Forms. 209

Figure 4-39. Imagery Information For Baseline (Top) And Enhanced (Bottom) Displays To Be Presented To Participants (similar to Figure 3-58). 212

Figure 4-40. Participants' Rating On The Informativeness Of The Enhanced Display. 214

Figure 5-1. Research Contributions. 218

Figure 5-2. Contributions Of The Rendering Techniques In Enhancing Situation Awareness. 228
## LIST OF TABLES

Table 2-1. A Comparison Between Surface Graphics And Volume Graphics. (Kaufman, Cohen, & Yagel, 1993) .................................................................................................................................................................. 31

Table 3-1. Performance Assessment For Non-enhanced And Enhanced Displays ................. 138

Table 3-2. Numbers Of Participants With Scores Of 1s And 0s When Using Baseline And Enhanced Displays .............................................................................................................................................. 145

Table 4-1. Storage Bits For U And V Information ........................................................................ 166

Table 4-2. Performance Assessment For Enhanced Display .................................................... 207

Table 4-3. Numbers Of Participants With Scores Of 1s And 0s When Using Baseline And Enhanced Displays .................................................................................................................. 213
CHAPTER 1  INTRODUCTION

1.1 Decision Making In Urban Settings

Increasing world urbanization is driving many activities into urban settings. The ability to simulate and explore interactively within a three dimensional (3D) virtual environment augmented with real-time situational information and terrain analysis functionalities, aids in various time critical tasks. Planning, preparing for and responding to time critical situations such as military operations, humanitarian relief missions, and search and rescue works require the collaboration of relevant participating agents (Blais, 2005). Decision makers, who are tasked to respond effectively to such emergencies, must be equipped with information on details of what is happening and stay informed with updates as the event unfolds and attentive to the extent of impact the dynamics of the situational settings might have. When working under constraints of limited time and incomplete information, decision makers typically rely on experience or "gut feel" rather than structured analysis of the given situation to make an assessment. Such undertakings are error-prone and may lead to catastrophic results due to insufficient situational awareness (Hartel, Smith, & Prince, 1991). On the contrary, when information becomes abundant, information overload may occur. An operator, working under stressful condition, may respond inappropriately to trivial information and ignore the critical ones (Laudy, Mattioli, & Museux, 2006). The impact of cognitive workload to process spatial information is aggravated in urban scenes due to scene complexity.
In addition, advances in information acquisition and remote sensing technologies have increased accessibility to data and potentially influences the way in decision makers make assessments in time-critical situations. The ability to integrate views from different sensory devices increases situational awareness. When augmented with the ability to perform terrain analysis, enhanced situational understanding could be formed. Such analysis aids in determining the relationships of the factors present and form logical conclusions related to mission accomplishment. Decision makers, inundated with myriad of information from various sources, often face challenges in forming accurate situational pictures. According to Endsley (Endsley, 1995), situation awareness begins with perception of elements in the environment which form the basic building blocks for comprehension and projection. Comprehension involves integration of various pieces of information and a determination of their consequence to the main goals. Projection, the highest level of situation awareness, refers to the ability to predict future events based on knowledge of the settings and dynamics of the elements and understanding of the situation.

Modern information technology could be leveraged to enhance situational awareness and understanding. The enhancements are of practical significance if they help in accelerating and improving decision making processes. The aim is to surpass information superiority, with achieving “decision superiority” as the end goal. The decision making processes will have to be accomplished by a combination of human and machine reasoning (Marsh, Quinn, Toth, & Jakubek, 2001). Effective external aids can give decision makers competitive advantages by overcome human limitations and enhance human reasoning.
1.2 Synthetic Urban Terrain For Modeling And Simulation (M&S) And Command And Control (C2)

Synthetic terrains are valuable tools to support applications related to training, planning and decision making. Faced with the changing environment of market forces and enabling technologies, synthetic terrain generation has undergone evolutionary process from time-consuming, manual-intensive, monolithic creation from proprietary vendors to the "on-the-fly", procedural terrain generations, built around standard-based designs for interoperability and reusability. With the readily availability of quality geo-spatial data in today's digital age, the emphasis is on systems that "pull from multiple data providers and push to multiple consumers" within minimal turnaround time (Salemann, Curley, Perkins, & Hubbard, 2006). The ability to visualize or navigate synthetic environments integrated with latest changes can facilitate rapid comprehension of complex urban settings for simulation and command and control systems. This is especially vital for time critical tasks, such as military operations, and search and rescue works, where having timely and accurate situational awareness is imperative for mission success.

In the defense domain, three key enablers of battle command are Command and Control (C2) systems, Modeling and Simulation (M&S) systems and Geographical Information System (GIS) (Lehman, 2009). Though C2 and M&S communities use GIS data sources for terrain generations, differences exist in their terrain generation processes and applications. M&S systems traditionally employ static terrain databases for 3D environment modeling while Battle Command Systems (BCS) typically use maps and GIS to serve as backdrops for real-time information displays. As the technologies have advanced in recent years, it has become possible for M&S and BCS systems to share the same GIS-based environment representations. Such
integrations provide C2 decision support with M&S capability to perform course of action analysis, rehearse timing of operations and evaluate mission plans by playing out scenarios defined in the Common Operational Picture (COP) (Stanzione & Lashlee, 2008).

1.3 3D Urban Terrain As Backdrop For Enhanced Situational Picture

Planning and operating activities within urban environments require a sound understanding of the surrounding world often populated with the plethora of man-made structures. Urban terrains, rich in visual and functional complexity as a result of developments due to various cultural factors, are difficult to model (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007). 2D (two-dimensional) maps, lacking in depth information, are inadequate to provide effective representations of such environment. Efficient urban terrain representation and data exploitation in 3D (three-dimensional) world are required to comprehend the built-up environment. Concomitantly, progresses made in GIS (geographic information system), data acquisition techniques such as LIDAR (Light Detection And Ranging) acquisition of 3D data, and improvements made in reconstruction techniques coupled with significant progresses made in procedural city generation using techniques such as fractals, L-systems to construct urban environment, contribute to availability of highly detailed urban models for visualization and other terrain-related analysis tasks (Lin, Jing, & Zhang, 2008) (Kelly & McCabe, 2006). The focus today has shifted from merely textured digital elevation representations primarily for open terrain to geometrically complex representations for urban terrain densely populated with 3D objects (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009). Managing such abundant and rich urban-related information interactively and effectively for terrain-related
tasks, such as real-time rendering from diverse viewpoints, be it the view point of UAV (Unmanned Aerial Vehicle) or that of the pedestrian moving along the street, becomes a challenge for terrain modelers and graphics engineers.

Urban terrains are often populated with numerous small, connected 3D objects such as single or small group of buildings. Each of these 3D objects is usually represented using polygons with different facade color information. The terrain geometries are highly discontinuous, and different views of terrain at disparate heights result in drastically different visual effects of the scene. Ground objects, for example, are often fully visible when viewed orthogonally downwards from the bird eye's views. They are, however, usually occluded when viewed from an oblique angle from a distance. Such terrains, scattered with 3D models, cannot be represented efficiently using classical multi-resolution / Continuous LOD (CLOD) rendering. While multi-resolution texturing and geometric levels of detail may provide a satisfactory solution for close-view objects and moderate degrees of simplification, similar approaches cannot be applied to distant objects (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009).

Typical image generation systems simplify or reduce terrain polygons for distant terrain features to achieve interactive rendering rate. If managed inadequately, real time visualization of such terrain may result in LOD switching popping artifacts which could be further aggravated by different shading effects due to different geometry representations. Switching geometrical representation of the urban setting may also results in erroneous visual effect leading to misleading conclusion. An example of a misleading visual cue occurs when an enemy, hiding in a shadowed sheltered balcony, becomes exposed when the building model switched to lower
LOD models where the balcony are textured and mapped onto building wall. In urban settings where timely detection of activities is essential for missions, such popping artifacts present distraction to already cluttered urban landscape, and are undesirable for urban visualization and simulation.

1.4 Terrain Viewshed Analysis To Aid In Situational Understanding

Likewise in terrain analysis, computational complexity involved in terrain-related calculations such as sensor coverage analysis involving multiple line of sight checks grows in tandem with the resolution and richness of source data. Performing such analysis is vital for mission planning tasks, such as route planning to ensure that designed path do not enter sensor zones of detected enemy forces, and humanitarian relief missions to plan a feasible channel for the rescue team to reach disastrous site. The ability to evaluate effects on terrain-related activities due to natural and man-made features and visualize the effects interactively within a three-dimensional (3D) virtual environment gives competitive edges to decision makers during time critical operations (Medby & Glenn, 2002) (Petitjean, 2011). Terrain analysis such as viewshed analysis for the urban environment is especially critical to built-up areas due to scene complexity and inter-object occlusions. Timely visual indications of detection and firepower ranges from disparate key locations provide decision makers with valuable information, such as optimal hiding locations and detection vulnerabilities.
1.5 Fusing 2D Images And 3D Terrain

Generating urban terrain populated with details such as buildings and skyscrapers presents an array of challenges. Convincing representation of the surfaces of open terrains, where height variations are relatively small, could be visually simulated by simply projecting orthogonal satellite imageries onto the terrain height map. Unlike open terrains, urban terrains, which are often populated with detailed 3D models, such as buildings, require additional functions to handle vertical surfaces. Typical geo-specific urban terrain generation process entails performing building extrusions from footprints and draping satellite imageries over building rooftops. Generic wall textures are then applied to building facades. Examples of terrain generation tools that generate generic wall textures for buildings include Presagis' Terra Vista and Cogent3D's GenesisRTX. These generic wall textures may not match the appearances of actual buildings. As such, these procedural modeling engines fail to reproduce building facades in synthetic form precisely. While enhanced building models with photorealistic wall textures could be constructed by 3D modelers using images taken from oblique angles capturing the building facades, enormous amount of manual work in texture manipulation and mapping are required. Cogent3D's Geosketch, a 3D modeling tool, attempt to streamline the modeling process by importing and projecting oblique and nadir images during building geometries construction. Nevertheless, the image projection algorithms adopted do not consider geometry occlusions from other buildings. Human intervention is required to select appropriate textures to be mapped onto corresponding faces. The approach also does not consider each and every contributing texture parts. Fusing of images is not performed when different parts of a building exterior are partially visible in different input images. An automated solution that extracts and
applies oblique 2D images onto building geometries can streamline the process to texture map building facades. The process would require fusion of 2D images and 3D terrain.

Additionally, video surveillance has grown by leaps and bounds with decreasing cost and better capability of cameras (Sizemore, 2008). The video imagery, rich in information and readily available from an array of spatial sensing devices, if used with prior constructed synthetic terrain as backdrop, has great potential to promote situational assessments. Draping these real time imageries over synthetic 3D terrain puts them into context. The capability to visualize virtual terrains augmented with real-time imageries from varying angles and viewpoints other than viewpoints of captured imageries and videos allows novel insights to be gained (Owens, Sycara, & Scerri, 2009). The approach relieves cognitive efforts of the operators in associating spatial positioning and viewing angle of the sensor devices and scene activities. Such spatial data fusion has great potential to provide substantial performance improvements over single sensor situation assessment functions. Fusing myriad of multi-resolution and multi-view 2D inputs spatially within open terrain typically involves a large number of computationally demanding tasks such as image registrations, mosaicking and calibrations (Brown, Gillbert, Holland, & Lu, 2006) (Se, Firoozfam, Goldstein, & Dutkiewicz, 2010). One approach to fuse aerial video and synthetic terrain is accomplished by projecting perspective trapezoidal video imageries onto the synthetic terrain. The approach works well if the acquired aerial video imageries are captured from high viewing locations and are superimposed over relatively flat terrain, with minimal occlusion among terrain features. Simply mapping oblique or rooftop or street-level view imageries over urban terrain populated with buildings and objects, however, is inappropriate for the urban environment due to scene
complexity and high prevalence of inter-objects occlusions. A target vehicle travelling on a street between two skyscrapers, when viewed at a slight oblique angle, could be well-hidden. When such imageries information is projected over the synthetic terrain, a misleading conclusion would be made. To overcome this limitation, 3D mapping of 2D imageries must be applied only to areas with direct camera line of sight when superimposing imagery and urban terrain data.

1.6 Research Objective

The research objective of this dissertation is to adapt GPU ray tracing rendering technique for improved accuracy and understanding of urban terrain using BlockMap and viewshed rendering, and to achieve real time fusion of 2D images and 3D terrain for enhanced situational awareness. The motivations of the research are multi-fold. First and foremost, increasing urbanization is driving many activities into urban settings. The ability to visualize scene activities updated with the latest situational information potentially aids in various time critical and planning tasks. Some areas where enhanced situation awareness is critical include aviation (Nullmeyer, Herz, & Montijo, 2009), emergency response (Blandford & Wong, 2004), military command and control operations (Gorman, Cooke, & Winner, 2006), and offshore oil and nuclear power plant management (Flin & O'Connor, 2001). The richness and complexity of urban terrain related data also motivates the need for efficient techniques to process these data. Moreover, advances in remote sensing technologies, 3D reconstructions, and the advent of the computer age and today's information age have brought about availability of tons of disparate, fragmented terrain related data. The potential to gain insights from these data by piecing them together motivates the search to formulate efficient techniques to fuse and visualize the information in an insightful
manner. A lot of research have been devoted to achieving fusion of spatial information from various sources (Brown, Gillbert, Holland, & Lu, 2006) (Coorg & Teller, 2000) (Kumar, et al., 2001) (Sawhney, et al., 2002) (Se, Firoozfam, Goldstein, & Dutkiewicz, 2010). Additionally, the advent of programmable GPU pipelines has brought about significant performance improvement and innovation in interactive graphics techniques. One of the research motivations involves exploring the power and feasibility of using programmable GPU to achieve the aforementioned challenges in processing and presenting urban terrain related data. The final motivation is to measure and study the effects of proposed visualization techniques on users' performances in spatial reasoning tasks.

1.7 Overview

This dissertation describes the research works on adapting real-time GPU ray tracing and image fusing techniques for enhanced situation awareness and understanding. The solutions harness graphics rendering techniques and GPU programmability to visualize urban terrain with accuracy, viewshed analysis and real-time imageries. The visualization techniques aid in painting enhanced situational pictures. Two main research areas are deliberated. The first involves using GPU ray tracing for accurate visualization and improved understanding of urban terrain. The second involves real-time fusion of 2D images and 3D terrain.

Chapter 2 provides relevant background information and previous works done pertaining to the research works. Chapter 3 describes research works on enhancing and adapting approaches that leverage on GPU ray tracing to achieve accurate visualization and improved understanding of urban terrain. The research explores using hybrid rendering approaches that comprise of both
rasterization and GPU ray tracing, to achieve accurate visualization and viewshed rendering of urban terrain. Chapter 4 describes research works on real-time fusion of 2D images and 3D terrain. The research aims to rapidly fuse geo-referenced images with 3D terrain to allow the presentation of real-time terrain rendered with current situational pictures. Lastly, Chapter 5 provides a conclusion of research contributions, limitations and potential future works and enhancements.
CHAPTER 2  BACKGROUND AND PREVIOUS WORKS

This chapter provides background information and reviews previous works done that are related to the research. It commences with description on Endsley's definition of situational awareness, which involves perception, comprehension and projection. The chapter moves on to discuss on the benefits of using information visualization to aid in decision making. It then proceeds to introduce the programmable graphics pipeline, and current state of the art terrain rendering and image fusion techniques. The chapter summarizes with identified shortfalls of solutions in terrain visualizations and image fusing techniques.

2.1 Situational Awareness

The advent of the computer age and today's information age brought along new challenges in integrating and interpreting tons of data, made available due to advancements in datalink and internet technologies. Forming a complete picture, to aid in situational awareness using disparate, fragmented information, however, can be extremely challenging. Endsley defines situational awareness as "The perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" (Endsley, 1995). According to Endsley, situation awareness begins with perception of elements in the environment which form the basic building blocks for comprehension and projection. Comprehension involves integration of various pieces of information and a determination of their relevance to the underlying goals. Projection, the highest level of situation awareness, refers to the ability to predict future events based on knowledge of the status and
dynamics of the elements and comprehension of the situation. Figure 2-1 illustrates the three levels of situational awareness (Endsley, 1995).

![Endsley's Model Of Situation Awareness](image)

**Figure 2-1. Endsley's Model Of Situation Awareness (Endsley, 1995)**

### 2.1.1 Using Information Visualization To Aid In Decision Making

Visualization involves forming internal pictures using perceived external information of the environments (Searle, 1983) (Gibson, 1979) (Ntuen, 2009). It is controlled by both internal and external events. While the mind forms conclusions about the environmental settings based on past experiences, external events provide the triggers and signals for building the mental pictures. When graphically portrayed and rendered as displays, these external events have the
potential to aid in gaining a nominal level of situational awareness. In Figure 2-2, situational awareness of external events could be improved using an appropriate information display. For example, when the locations of major events such as explosion are visualized spatially on a centralized 2D map using an information display, enhanced situational awareness could be gained. An internal interpretation of the event in relation to the surrounding environments could be formed by the human mind. The information display aids in gaining a more accurate visualization of the events. It was highlighted by (Norman, 1993) that “The power of the unaided mind is highly overrated. Without external aids, memory, thought, and reasoning are all constrained. But human intelligence is highly flexible and adaptable, superb at inventing procedures and objects that overcome its own limits. The real powers come from devising external aids: it is things that make us smart”.
Through effective visualization, the high bandwidth channel of human visual perception, to process tons of data, is exploited. "People are biologically equipped to make spatial inferences and decisions, and experience refines their ability to do so. Visualizations can bootstrap this facility metaphorically, by mapping elements and spatial relations in the abstract domain onto elements and relations in a concrete visualization" (Johnson, Moorhead, Munzner, Pfister, Rheingans, & Yoo, 2005). Figure 2-3 illustrates the discovery process where raw data is transformed into knowledge with the user actively interacting and exploring via visualization.
2.1.2 Measuring Situational Awareness

Various situational awareness measurement techniques are used to measure situational awareness. They essentially belong to either knowledge based or performance based approach of evaluating situation awareness. Knowledge based techniques assess the subject's level of situational awareness directly while performance based techniques "infer" level of situational awareness from the subject's task performance (Muller, 2010). Figure 2-4 compares knowledge based and performance based techniques when evaluating users' level of situational awareness. It is observed that measurements are obtained at different stages in the process of user cognition (Pritchett, Hansman, & Johnson, 1996).
Whilst performance measures are simple to administer, assumptions made with the relationship between situational awareness and performance are often questioned. Often, situational awareness theory is used to determine if an interface is effective in decision making. As such, there is no necessity to measure acquired level of situational awareness directly. If the objective is to evaluate the effectiveness of the systems of interests, performance based measurements could be used to measure performance indicators such as speed and accuracy in user’s responses. Assumptions on user's likely actions, given their knowledge state, need to be considered when analyzing results collected using knowledge based measurement. On the other hand, performance based measurements give indications of users’ likely performance when systems of interests were to be used.

2.1.3 Situational Awareness For Urban Scenes

Having timely and accurate perception, comprehension and projection of complex settings is essential for enhanced situational awareness and effective decision making. Some areas where enhanced situation awareness is critical include aviation (Nullmeyer, Herz, & Montijo, 2009),
emergency response (Blandford & Wong, 2004), and military command and control operations (Gorman, Cooke, & Winner, 2006). The ability to visualize urban scene activities updated with the latest situational information potentially aids in various time critical and planning tasks. Traditional visualization systems to aid in situational awareness for large scale operations such as military warfare are 2D based. 2D maps are used as backgrounds. 3D visualization could also be used for enhanced situational awareness (Durbin, et al., 1998). Dragon (Durbin, et al., 1998) and JOVE (Feibush, Gagvani, & Williams, 2000) are examples of battlefield visualization systems that extended battlefield visualization into 3D.

A key challenge in most 3D situational awareness applications is the rendering of terrain in real-time. This is because 3D terrains, unlike 2D maps, are complex to render. Accurate visualization of the terrains is important as they are used as central referents for planning activities. Direct from source approach, which uses streamed terrain data to generate "on-the-fly" terrain at client machines, are often used for Command and Control (C2) applications. The approach ensures latest terrain updates if source data from terrain servers are up-to-date. For example, NASA World Wind and Google Earth are used in Army C2 Applications and Systems such as Command Post of the Future (Hieb & Maxwell, 2012). Such "on-the-fly" terrain generation is well suited for applications requiring fast turnaround prototyping without a high amount of details (Wiesner, Brockway, & Stanzione, 2011).

Urban terrains, populated with details such as buildings and skyscrapers, are difficult to visualize. They are often populated with numerous small, connected 3D objects such as single or small group of buildings. Each of these 3D objects is usually represented using polygons with
different facade color information. The terrain geometries are highly discontinuous. Different views of terrain at disparate heights result in drastically different visual effects of the scene. Ground objects are often fully visible when viewed orthogonally downwards from the bird eye's views but are usually occluded when viewed from an oblique angle from a distance. While multi-resolution texturing and geometric levels of detail may provide a satisfactory solution for close-view objects and moderate degrees of simplification, similar approaches cannot be applied to distant objects (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009). Typical approach to render terrain involves using polygon based representations. An alternative rendering approach is to use image-based rendering techniques. Nevertheless, such image-based rendering techniques are not used in the context of battlefield visualization for situational awareness (Kim & Hoffmann, 2003).

To render highly accurate urban terrain, considerable computational effort is needed. A compromise between achieving real-time rendering rate and accurate terrain representations would have to be made. Studies indicate that systems with frame rates at 10 Hz and below cause stress in terms of physiological responses and performance decrements. Achieving frame rate of around 15 Hz is preferred for a lot of tasks, including those that are psychomotor and perceptual in nature (Thropp & Chen, 2006). The ability to visualize information at interactive or real-time rates is important.

Subsequent sections of the chapter discuss previous works done related to rendering solutions and techniques pertinent to the research in providing visualization of accurate and up to date information for urban scenes.
2.2 Programmable Graphics Pipeline

The advent of the Graphics Processing Unit (GPU) allows for faster graphics processing speeds by offloading computationally intensive transform and lighting calculations from the Central Processing Unit (CPU). GPU has become a standard part of almost every commodity computing platforms. The recent increase in peak performance of GPU’s far outpaces the increase in peak performance of CPU’s (Figure 2-5) (Skinner & Hutchinson, 2010).

![Graph showing GPU (NVIDIA) Versus CPU (Intel) Peak Single Precision Floating Point Performance (Skinner and Hutchinson, 2010)](image)

Figure 2-5. GPU (NVIDIA) Versus CPU (Intel) Peak Single Precision Floating Point Performance (Skinner and Hutchinson, 2010)

Figure 2-6 illustrates the standard composition of the graphics pipeline. The rendering process is initiated by the 3D application which sets up and issues the drawing command and sending a sequence of vertices batched into geometric primitives to the graphics system to be rendered (Lindeman, 2008).
In Figure 2-7, the rasterization process of a 3D model is illustrated.

While the graphics operations were traditionally fixed, a paradigm shift started with the introduction of programmable vertex and fragment processing units (Figure 2-8). The GPU
programmable rendering pipeline can be programmed using Shaders, which are computer programs (Lindeman, 2008).

![Figure 2-8. The Programmable Graphics Pipeline](image)

### 2.3 Rendering Virtual Scene

Ray casting / tracing and rasterization are two graphics rendering techniques for image generations of 3D scene (Figure 2-9). Commonly used in graphics rendering, rasterization based rendering involves the process of transforming 3D scene geometries typically defined by collections of triangles, to 2D camera perspective plane for display. Transformed triangles are then painted with details from textures. To determine which surfaces are visible from a camera, z-buffering is exploited. Z-buffer manages image depth coordinates in 3D camera space and
determines which polygonal surfaces are closest to camera observer (Skinner & Hutchinson, 2010).

Ray casting / tracing, on the other hand, mimic the way light photons move in the real world. First introduced to computer graphics in 1982 by Roth (Roth, 1982), ray-casting renders 3D scenes onto a display by following rays of sight from viewpoint of the observer. While contemporary approaches in terrain rendering involves rasterization of triangle meshes generated using elevation data and achieves satisfactory rendering results, ray casting possesses attributes that are suitable for terrain representations and rendering of complex urban environments. Rendering performance using ray casting is output sensitive. It depends on the screen footprint spanned by the rendered objects. It possesses innate attributes that allow it to handle inter-object occlusion efficiently. Ray casting could also be extended to ray tracing (Whitted, 1979) to accurately simulate reflections, refractions, or shadowing. This is achieved by computing new ray vectors after the first intersection with the object surfaces (Figure 2-9). Traditionally used
for high quality rendering, ray casting is a computationally demanding technique. Therefore, rendering using ray casting often lacks interactivity when rendering large datasets. Exhibiting an intrinsic parallelism, in the form of completely independent rays, ray casting can take advantage of the massively parallel architecture of the GPU (Skinner & Hutchinson, 2010).

### 2.4 Visualizing Synthetic Urban Terrain

While digital terrain visualization started in 1960s, advances in computer science and computer graphics have made realistic rendering possible in the recent years. Today, various techniques, such as Computer Aided Design (CAD), Geographical Information System (GIS), image processing, and even digital video technologies, are employed for digital terrain constructions and visualizations. Nevertheless, a number of challenging problems remain (Ervin, 2004).

Recent terrain generation and storage and analysis techniques are geared towards having high-resolution representation and analysis to facilitate tasks related to the urban environment. Unlike open terrain where height variation in landform is relatively low, urban terrain populated with details such as buildings and skyscrapers presents different and additional array of challenges. One distinctive characteristic of the urban terrain is its highly discontinuous height elevation due to the presence of man-made structures such as flyovers, towers and buildings with intricate structures (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009). Insufficient resolution or sampling assumption for representation or analysis, results in misleading or erroneous conclusion when performing terrain-related tasks. Furthermore, fractal models that work splendidly in simulating and generating natural objects such as trees and grass may become
inappropriate when applied at urban scene (Reljic, 2006) (Ervin, 2004). Alternative sets of rules related to architectural structure constraints are deliberated instead.

While accurate depiction of terrain information is critical to planners and decision makers, of increasing importance to achieving situational awareness and understanding is visualizing effects due to terrain features. One such terrain effect is viewshed analysis. Viewshed or visibility analysis from point(s) to area surface typically involves multiple line of sight (LOS) calculations on the terrain digital elevation map (DEM). One usage of viewshed analysis includes determining optimal placements of radio towers. Another area of applications of viewshed analysis includes investigating the impact of placing a tall skyscraper to the urban landscape. Other applications include determining the area of coverage achieved when placing fire observation stations in mountainous regions.

With increasing activities such as military warfare being conducted in the urban environment, providing understanding of the terrain data at hand and usage of high resolution data such as LIDAR data for analysis are crucial to obtain an operational edge. The challenges involve management and presentation of such massive terrain data (Smith & Bishop, 2003). Another challenge involves fusion of spatial information to present the most up to date pictures. Some research devoted to achieving such fusion of spatial information include (Brown, Gillbert, Holland, & Lu, 2006) (Coorg & Teller, 2000)(Kumar, et al., 2001)(Sawhney, et al., 2002)(Se, Firoozfam, Goldstein, & Dutkiewicz, 2010).
The later subsections describe methods of terrain generations and rendering techniques. Section 2.4.1 describes trends and approaches to generating urban terrains. Section 2.4.2 discusses techniques to represent terrain relief, which involve polygon and volumetric representations. Section 2.4.3 describes scene complexity management, which involves techniques to handle and traverse terrain data during real-time rendering. Section 2.4.4 describes some image-based rendering techniques. Image-based rendering techniques render virtual scenes without considering 3D geometries. Commonly used image-based rendering techniques include using billboards to model trees or textured background polygons to create pseudo-3D appearances.

2.4.1 Generating Urban Terrain

Synthetic terrains are valuable tools to support applications related to training, planning and decision making. Faced with the changing environment of market forces and enabling technologies, synthetic terrain generation has undergone evolutionary process from time-consuming, manual-intensive, monolithic creation from proprietary vendors to the "on-the-fly", procedural terrain generations (Salemann, Curley, Perkins, & Hubbard, 2006). Conventional terrain generation involves hand modeling of the terrain using tools such as Autodesk’s 3DS Max or Presagis’ Creator. Such terrain construction process is labor intensive. The terrain generated are typically visually rich and best suited for used in simulation for close combat operations, urban scenarios, first person shooter applications, and architectural walkthroughs (Wiesner, Brockway, & Stanzione, 2011).

With advancement made in computer technology, some forms of automation such as generating roads, extruding building from footprints, and even populating the building interiors with floor
plans become possible. Some of these tools include TerraSim’s TerraTools, TrianGraphics’ Trian3D Builder, and Presagis’ TerraVista. Nevertheless, these terrain databases are static and typically consume a large amount of disk space. With readily accessibility of GIS data, the next terrain database approach, termed “Direct from Source”, involves loading GIS directly into simulation applications. Based on a ARC Advisory Group study, the worldwide market for GIS is forecast to grow 50% over the next five years (Reiser, 2009). Direct from source approach makes possible "on the fly" terrain databases and ensures latest terrain updates if source data from terrain servers are up-to-date. As such, this terrain visualization approach is often used for Command and Control (C2) applications. For example, Google Earth and NASA World Wind are used in Army C2 Applications and Systems such as Command Post of the Future (Hieb & Maxwell, 2012). The geospatial data are often streamed from GIS servers on local networks or the Internet or web-mapping services. They provide a world of data through open standard protocols, such as the Web Map Service (WMS) from the Open Geospatial Consortium. Examples of GIS servers include Microsoft Virtual Earth, ESRI’s ArcGIS Online and ArcGIS Server/Data Appliance, and OpenStreetMap (Wiesner, Brockway, & Stanzione, 2011).

Applications that load source data directly, however, can have performance and correlation issues as the source data is not optimized for runtime applications and processed differently by different applications respectively (Wiesner, Brockway, & Stanzione, 2011). Creating interoperable terrain datasets to be used for simulations, mission command and other decision support systems become a common goal of developers across different information system communities. Correlation among different terrain representations is vital to allow usage and sharing of similar GIS-based environment representations among different applications. There
are many challenges to achieving terrain correlation among different systems. One of them is the disparities in generation processes adopted by different communities. Creating correlated terrain databases generated using different tools is very difficult. One of the challenges is due to different triangulation algorithms used by different terrain generation tools to create polygon based terrains (Hieb & Maxwell, 2012).

GIS datasets, utilized to generate urban terrain, involve satellite imageries, elevation data and 2D vector dataset. The 2D vector dataset defines building footprints and heights. The satellite imageries are draped over elevation data. They are combined to form textured digital elevation representations. Such digital elevation models (DEM)s coupled with corresponding aerial imagery provide coarse representation of the terrain. To approximate detailed buildings, 2D vector dataset containing the building footprints are extruded to create “block-shaped” buildings. Figure 2-10 illustrates satellite imageries, elevation data and building footprints data required to construct urban terrain. Most extrusion systems produce flat roof buildings with generic wall textures. Recent procedural city generation tools have extended rules beyond simply extruding building footprints to include building facade details such as adding windows, facades, and arches geometrically. One example of such terrain generation tools that construct detail building structures include CityEngine procedural city generation tool.
2.4.2 Representing Terrain Relief

3D representations, such as terrain relief, can be accomplished by polygonal or volumetric representations. Polygonal representations focus on defining the surfaces of the 3D objects while volumetric representations define terrain geometries by means of the volumes they occupy.

2.4.2.1 Polygonal Representation

Representing terrain topology in three dimensions typically involves construction of geometrical polygons to approximate the terrain top. One straightforward geometrical height representation involves using evenly spaced height posts, arranged in regular grids, each assigned with a height value. This approach allows easy storage and manipulation using a 2D array as data structure. However, such regular grid height posts representation do not address geometrical complexity distributed asymmetrically, where large areas of flat surfaces are oversampled while uneven surfaces are smoothen as the details are not captured. The alternative solution is to use Triangulated Irregular Network (TIN) representations (Grumet, 2004). A TIN provides
representation of a terrain surface using triangulated faces formed from a network of points located and spaced irregularly in 3D space. Represented by contiguous, non-overlapping triangles, the ability to allocate irregular distribution of vertices provide flexibility in representing surfaces with varying complexity. Alternative representation include using Right-triangulated irregular network (RTIN) which is a compromise between simplicity of regular space grid format and efficiency of TIN and can be represented using a binary tree or a quadtree. (Figure 2-11)

![Figure 2-11. From Left, Regular Height Posts, Triangulated Irregular Network (TIN), Right-Triangulated Irregular Network (RTIN) (Bartoň, 2011)](image)

2.4.2.2 Volumetric Representation

Analogous to an image, volumetric representation involves partition of the space into individual cells or voxels. A voxel-based object models a discrete representation of the original continuous object. The precision of the representation is determined by the volume buffer resolution.

Table 2-1 illustrates comparison made between volume graphics and surface graphics (Kaufman, Cohen, & Yagel, 1993).
Table 2-1. A Comparison Between Surface Graphics And Volume Graphics. (Kaufman, Cohen, & Yagel, 1993)

<table>
<thead>
<tr>
<th>Capability</th>
<th>Surface Graphics</th>
<th>Volume Graphics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Rendering performance</td>
<td>Sensitive to scene and object complexity</td>
<td>Insensitive to scene and object complexity</td>
</tr>
<tr>
<td>2. Memory and processing requirement</td>
<td>Variable - depends on scene and object complexity</td>
<td>Large but constant</td>
</tr>
<tr>
<td>3. Object-space aliasing</td>
<td>None</td>
<td>Frequent</td>
</tr>
<tr>
<td>4. Transformation</td>
<td>Continuous, performed on the geometric definition of objects</td>
<td>Discrete, performed on pixel blocks (windows)</td>
</tr>
<tr>
<td>5. Scan conversion and rendering</td>
<td>Pixelization is embedded in viewing</td>
<td>Voxelization is decoupled from viewing</td>
</tr>
<tr>
<td>6. Boolean and block operations</td>
<td>Difficult, must be performed analytically</td>
<td>Trivial, by using voxelate, voxel-by-voxel operation, aggregation, octrees</td>
</tr>
<tr>
<td>7. Rendering of interior and amorphous phenomena</td>
<td>No, surface only</td>
<td>Yes, rendering of inner structures as well as surfaces</td>
</tr>
<tr>
<td>8. Adequacy for sampled data and intermixing with geometric data</td>
<td>Partially and indirectly (fitting followed by surface rendering)</td>
<td>Supports representation and direct rendering</td>
</tr>
<tr>
<td>9. Measurements (e.g., distance, area, volume, normal)</td>
<td>Analytical, but often complex</td>
<td>Discrete approximation, but simple</td>
</tr>
<tr>
<td>10. Viewpoint dependency</td>
<td>Requires recalculation for every viewpoint change</td>
<td>Precomputes and stores viewpoint-independent information</td>
</tr>
</tbody>
</table>

2.4.3 Scene Complexity Management

Urban scenes often comprise of complex geometries. Visualizing and rendering such scenes at real-time create challenges due to urban terrain geometrical complexity and system memory and processing limitations. Constructing realistic representations of terrains utilizing millions of triangles could easily overwhelms even state of the art systems. Multi resolution representations and rendering algorithms are often employed for visualization systems.
2.4.3.1  Level Of Details For Urban Terrain

Having different level of details (LOD), first introduced by James H. Clark (Clark, 1976), is the core to many approaches to manage scene complexity. This involves having representations with different degree of complexity according to metrics such as object distance from viewer, size and interest. Figure 2-12 illustrates the five LODs defined by CityGML (Kolbe, Gröger, & Plümer, 2005). When visualizing a scene, objects farther away from the observer typically have less visual impact. This is because distant objects usually occupy smaller footprints on rendered screen. An architecturally complex building, for instance, adorned with curves, columns and arches geometrically could be simplified to a block when viewed from a far distance.

Figure 2-12. The Five Levels-of-Detail (LOD) Defined By CityGML (Kolbe, Gröger, & Plümer, 2005)
The most commonly used techniques to construct models with different LOD involve polygon simplification and reduction. Other methodologies to acquire simplified representations for the models involve having image representations of the models mapped onto simple textured quads. Such approach uses image based rendering techniques and characterizes "imposter" approach (Maciel & Shirley, 1995).

2.4.3.2 Scene Graph To Organize Terrain Structure

Scene graphs are often used to organize the terrain structure in a logical and spatially-related manner. The organization consists of arranging groups of nodes in a hierarchical data structure for efficient rendering. Figure 2-13 illustrates the scene graph of two chairs, a table and a room. The scene graph contains a root node, with children consisting of group nodes that organize geometry and the rendering state. The leaf nodes at the bottommost define the geometry of respective scene objects. In Figure 2-13, the root node has four group node children. The chair group nodes (color-coded in red) have similar child node. Only one chair leaf node is used as the two chairs are identical, though its parent group nodes transform two instances of the chair to different locations.
Using group nodes with child nodes, logical organization of the geometric and state data is defined. Typical scene graphs, such as those encoded by OpenFlight format, provide various different node types with a wide range of functionalities. They include switch nodes that enable or disable selected children, LOD nodes that select relevant child node based on distance from the viewer, and transform nodes that perform transformations on child geometry. Scene graphs encapsulate the graphics primitives and rendering state to be used for visualization. They can be created through a low-level graphical application programming interface (API). OpenSceneGraph is one such graphic toolkit based on the low-level OpenGL API that leverages on the scene graph. It is used to implement visualization applications for virtual 3D environments. OpenSceneGraph allows graphics engineers to develop applications in a standard way in handling graphics tasks such as scene culling and LOD management (Wang & Qian, 2010).
2.4.4 Image Base Rendering

Image-based rendering refers to the rendering of 3D scene using 2D images without the need to construct 3D geometric models explicitly. This section of the dissertation describes some image based rendering methodologies used for surface representations. The BlockMap representation (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007), an image based rendering technique to provide coarse representation of a far away city block, is also elaborated.

2.4.4.1 Texture Mapping

Texture mapping is applied to the model to improve the detailed appearance. The application of texture mapping to 3D graphics is first introduced by Edwin Catmull in 1974 (Catmull, 1974) in his PhD thesis. A texture, an image with rectangular array of pixels, also called texels. Each texel stores the color information and occupies a texture coordinate determined by the width and height of the texture. Texture mapping refers to the application of texture onto a face or polygon, painting details onto the surface (Figure 2-14).
2.4.4.2 Bump Mapping

Though texture mapping enriches surfaces with additional visual information, when viewed from an oblique angle, flatness of the surface becomes apparent with the absence of shading cues associated with surface height variation, such as shadow and highlight. To improve the appearance of 3D depth to texture mapping, one technique involves the use of bump mapping (Blinn, 1978) where 3D attributes are assigned to the textures. This allows surface shading and highlights to be added during rendering to give the illusion of rough surfaces. This is accomplished by varying the pixel intensity in response to a height of the rendered surface point.
In Blinn’s approach, each surface normal is defined according to a bump map. The resultant normal is used when shading the surface to give the appearance of unevenness (Figure 2-16).

2.4.4.3 Parallax Mapping

Parallax mapping (Kaneko, et al., 2001) renders irregular surfaces by displacing the texture coordinates according to the view angle to the surface normal and respective height map value at that point. The displacement of the texture coordinates depend on the steepness of the viewing angle, and/or the depth value at a point. This gives the appearance of depth due to parallax effects when the viewpoint changes (Figure 2-16).

2.4.4.4 Relief Mapping

An enhancement to the parallax mapping technique, relief texture mapping (Oliveira, Bishop, & McAlliste, Relief Texture Mapping, 2000) (Policarpo, Oliveira, & Comba, 2005) creates illusion of depth using ray casting. A depth map stores the distance of the sampled surface points to a reference plane. The intersection point of the viewing ray with the surface is computed using a roof finding technique using ray casting. One technique used is a binary search. Relief texture mapping allows us to correctly render self-occlusion and self-shadowing. Figure 2-15 illustrates the ray casting process to find the intersection point of the ray with the object surface using binary search. The numbers indicate the sequence of points that are used for height comparisons. The intersection point between A and B is located.
Figure 2-15. Relief Mapping Using Ray Casting (Policarpo, Oliveira, & Comba, 2005)

Figure 2-16 illustrates the different rendering effects using various rendering techniques. It is observed that relief mapping gives the most realistic effects, complete with accurate shading, lighting and occlusion effects.

Figure 2-16. (a) Bump Mapping, (b) Parallax Mapping And (c) Relief Mapping (Policarpo, Oliveira, & Comba, 2005)

The aforementioned rendering effects provide mesostructure geometry information onto surfaces. Nevertheless, devoid of 3D geometric information, silhouettes could not be depicted. When viewed at a horizontal angle, the surface flatness becomes evident and illusion breaks.
2.4.4.5 *Billboard*

To create illusion and presence of 3D objects viewed at oblique angles, billboards, which are textured planar polygons, could be used to replace intricate geometry to increase performance. Billboard is often applied to render complex objects such as trees. It uses an axial rotation to transformed itself to face the viewer. Billboards are especially suitable for objects which are cylindrically symmetric such as trees. When allowed to rotate about a point in place of an axis, billboards could be used to represent symmetric objects such as smoke, clouds and bushes (McReynolds & Blythe, 1998).

2.4.4.6 *BlockMap Representation For Urban Environment*

A BlockMap (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007), which is a GPU-friendly simplified representation of urban environment rendered using GPU ray casting, is used to encode and provide coarse representation of far away city blocks. The approach requires minimal polygonal data while preserving the ability to portray building heights and wall textures with computational complexity depending on the BlockMap’s screen footprint.

Storing geometrical data in texture memory, BlockMap is a simple, heightfield like, representation for a set of discrete vertical textured prisms for rendering far-away city blocks while preserving large scale features such as silhouette and color of the buildings. The BlockMap representation is used as a lower LOD representation as it provides a view-independent, simplified representation of the geometrical data, giving full support to visibility queries, and, when built into a hierarchy, offers multi-resolution adaptability (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007). Unlike bump, parallax and relief
mapping, and billboards which work at restricted view angle, BlockMap provides a view-independent solution for a city block representation.

Figure 2-17 illustrates the visual effects of a city block rendered using BlockMap and polygon based representation. It is observed that basic terrain features are faithfully preserved using BlockMap. It requires only polygons to define the bounding faces.

![City Block Rendered Using Polygons (Left) And BlockMap (Right)](image)

*Figure 2-17. City Block Rendered Using Polygons (Left) And BlockMap (Right)*

Standard BlockMap rendering assumes that the majority of buildings has vertical walls. Such geometrical constraints are aligned with the assumption made while acquiring and storing terrain elevation information and during terrain reconstruction and auto-generation. Using the assumption that all buildings have vertical walls, the BlockMap representation stores in a texture memory height, roof and wall color information required to render a portion of the city (Figure 2-18).
Synthetic terrains are often subdivided into tiles. A BlockMap can be used to represent individual tile. Integrated into traditional level-of detail (LOD) modeling, BlockMap can be rendered using distance-dependent resolutions. Such data representation provided by BlockMap structure allows the relevant city block information to be efficiently accessed at rendering time by GPU-based ray casting.

Using ray casting technique, rendering BlockMap requires only polygon based geometries that define the bounding box for each terrain tiles. The ray casting process is activated by the rasterization of visible faces of the bounding box (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007). Figure 2-19 illustrates the side view of a ray casting process.
For each fragment, the algorithm casts rays of sight through the space until a surface, be it a spot on the ground or patch on the roof or side of a building is hit, or when the ray exits the block space defined by the BlockMap. In the former case, the texel color representing the hit surface is fetched and rendered. To determine which surface is hit, the rays are sampled and compared with corresponding values encoded in the height map.

Constructing BlockMap textures require processes of extracting color information from the actual geometrical representations and writing them in corresponding addresses on the BlockMap textures. Filling up roof/ground information is clear-cut by performing orthogonal top view rendering of the terrain tile. To populate wall information for the BlockMap texture, an approach is to perform orthographic renderings of the textured models following its boundary to
compute columns of wall colors. If the number of wall columns exceeds that reserved for vertical texture, the wall texture is resized horizontally (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007). A more elegant GPU accelerated, multi-view sampling strategy is subsequently employed (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009) where multiple views of the actual city block is taken and re-projected onto the BlockMap's prisms. The final color is the result of contributions from all views.

The ray casted BlockMap possess characteristics that is desirable for use in terrain representation. First of all, spatial errors of BlockMap representation are bounded by its resolution and could be estimated. (Cignoni, Di Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007). In addition, to facilitate terrain viewing and navigation over a large area, streaming of BlockMap data could be done to achieve remote visualization of large scale urban terrain. (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009) demonstrate interactive remote visualization of large urban environments using city models, represented using multi-resolution BlockMaps, stored on a remote server.

### 2.5 Accelerated Ray Casting / Tracing Techniques

As spatial data of higher resolution and spanning over larger areas become readily available, the demands for speedy processing of terrain-related data inevitably increase. Programming on graphics cards offers a promising approach to compute volume rendering methods, taking advantage of the parallel architecture of today’s graphics cards. Such modern GPUs have great potential to allow for rapid acceleration of volume rendering applications at interactive and real time frame rates. One technique for rendering volumes is ray casting. Multiple rays may be cast
independently, resulting in full pixel-parallelism. Unoptimized ray casting involves sampling points along the rays to locate intersection points. Various techniques are used to speed up the ray casting process. They involve efficient spatial database management, ray traversal algorithms and space leaping techniques.

2.5.1 Spatial Database

A spatial database refers to a database to manage geometric, geographic, or spatial data. Spatial databases are often optimized to store and query data, typically in the form of points, lines and polygons, in 3D space. To represent terrain heights, digital elevation models (DEM), which contain elevations at points arranged in a grid based data structure, are commonly used. Figure 2-20 illustrates the rendering (right) of the height field (left) of the Grand Canyon (Henning & Stephenson, 2004).

![Figure 2-20. Rendering (Right) Of Height Field (Left) Of The Grand Canyon (Henning & Stephenson, 2004)](image)
While common approach to depict height fields involve converting the terrain information into polygon mesh to be rendered using commodity graphics card, with recent developments of programmable GPUs, ray casting offers rendering of a more concise representation by considering height fields in the original form. Figure 2-21 illustrates a simple ray tracing scheme where one ray per pixel is intersected with the bounding box of the height field.

Spatial partitioning is often done to speed up the processing. Quadtrees and octrees, for example, are spatial data structures that partition a region of space into 4 or 8 equally sized quadrants or octants respectively. These data structures can be represented in the hierarchical tree structures or pointerless representations. A hierarchical tree structure comprises of a root cell,
intermediate cells, and leaf cells while a pointerless representation stores only leaf cells (Figure 2-22) (Frisken & Perry, 2002).

![Octree (Top) and Quadtree (Bottom)](image)

**Figure 2-22. Octree (Top) and Quadtree (Bottom)**

### 2.5.2 Ray Traversal Algorithm

In ray tracing, a traversal algorithm for a grid based terrain return all the cells traversed by the ray using equation 1. The traversal typically terminates once an intersection is found.
Two commonly used, optimized grid traversal algorithms include Bresenham line draw algorithm and Digital Differential Analyzer (DDA).

2.5.2.1 Digital Differential Analyzer (DDA) algorithm

The DDA algorithm interpolates values over an interval between start and end points. Considering a line with a positive slope less than or equal to 1, points along the line are sampled at unit x interval and corresponding y values are computed using equation 2 until the end point is reached.

\[ y_{k+1} = y_k + m \]  \hspace{1cm} (2)

For lines with a slope greater than 1, the role of x and y is reversed where points are sampled at unit y interval and x values are computed using equation 3 until the endpoint is reached.

\[ x_{k+1} = x_k + \frac{1}{m} \]  \hspace{1cm} (3)

Similar computations are carried out to determine sampled points along a line with negative slope. Sample values are rounded off to the nearest integer to correspond to a screen pixel (Watt, 2000).

2.5.2.2 Bresenham Line Algorithm

Considering a line with positive slope less than or equal to 1, the next point on the line could either be point \((x+l,y)\), or point \((x+l,y+l)\) if points are sampled at unit x intervals (Figure 2-23).
To determine which point to be plotted, the Bresenham line algorithm keep track of an error, $\epsilon$. As sampling proceeds from $x$ to $x+1$, $y$ values are increased by an amount equivalent to the slope of the line, $m$. If the difference between the new value and $y$ is less than 0.5, plot $(x+1,y)$ would be plotted, else $(x+1,y+1)$ is selected (Flanagan).

$$y + \epsilon + m < y + 0.5 \quad (4)$$

![Diagram showing the Bresenham line-drawing algorithm](image)

**Figure 2-23. The Bresenham Line-Drawing Algorithm**

### 2.5.3 Space Leaping

Space leaping is the process of advancing to the next sampled point of interest using a larger step, instead of a constant, small increment. A distance map defining the shortest distance from each and every point from any surface. Figure 2-24 illustrates a height map and the corresponding distance map.
Figure 2-24. Height Map (Left) And Corresponding 2D Distance Map (Right)

Figure 2-25 illustrate space leaping using sphere tracing. Using distance map, that indicates the shortest distance from each and every point, sampling along a ray progresses at larger steps and manages to converged within three iterations (Donnelly, 2005).

Figure 2-25. Space Leaping Using Sphere Tracing
2.5.4 Maximum Mipmaps

Two classes of algorithms are used in current state-of-the-art ray traversal methods. The first involves performing ray object intersection checks on height field data. The second uses pre-computed information to achieve fast traversal and precise intersection computation. The second approach comes at the cost of pre-computation time (seconds to minutes) and storage requirements in graphics memory. (Tevs, Ihrke, & Seidel, 2008) utilize a fully subdivided quad-tree acceleration data structure for ray-height-field intersection algorithm using the maximum mipmap. The pre-computation process computes the maximum value of each cell of the height field (Figure 2-26). The mipmap data structure is constructed by utilizing the maximum value of the four underlying samples iteratively. At the finest level, the heights of the four corners of a bilinear patch are stored in one RGBA value.

![Figure 2-26. Maximum MipMap Pre-Computation (Tevs, Ihrke, & Seidel, 2008)](image)

The intersection computation commences at the highest mipmap level. During ray traversal through the height field hierarchy, the algorithm computes intersection of the four planes bounding the volume on the sides and the plane of maximum elevation at the current level. Figure 2-27 illustrates the ray traversal process through the mipmap data structure for a given height field. The graph at the bottom depicts a one-dimensional height field at the finest level.
The tree on the top encodes the maximum mipmap structure. Each node contains the maximum height field value. The black lines in highlighted in bold indicate the order of traversal. The first six steps traverse through the "hill" in cell 4 and 5 to descend to level 1 of the bounding volume hierarchy. Subsequent steps undertake larger strides until cell 12 containing the correct intersection is obtained.

Figure 2-27. Ray-Height Field Intersection Using The Mipmap Hierarchy Structure (Tevs, Ihrke, & Seidel, 2008)

2.6 Fusing 2D Images And 3D Terrain

Hitherto, acquiring 3D geospatial data requires highly sophisticated and expensive analog devices. Datasets employed in terrain generation used to be confined to geo-rectified orthogonal
satellite depicting top view of the environment. Today's digital age has brought about high resolution imagery with amateur digital camera devices. When attached and used with GPS (Global Positioning System), which has become ubiquitous in handheld devices and commercial navigating tools, photogrammetry could be performed. Unmanned Aerial Vehicle (UAV) based remote sensing also contributes to 3D data acquisition. To achieve geo-specific terrain generation inclusive with textured building facades, registering and aligning oblique images onto planes parallel to building facades needs to be done. This necessitates an automated image fusion process to rectify and merge photographs taken from different views into a single composite texture mosaic to be applied onto building facades.

Recent 3D Modeling tools attempt to streamline texture mapping process. Cogent3D Geosketch, for example, imports and projects oblique and nadir images during building geometries construction. Nevertheless, the image projection algorithms adopted do not consider geometry occlusion from other buildings. Human intervention is required to select appropriate textures to be mapped onto corresponding faces. The approach also does not consider each and every contributing texture parts capturing similar patch of interest. Fusing of images is not performed when different parts of a building facade are partially visible in different input images. A system to extract and apply oblique 2D images onto building geometries is to be established to streamline the process.

Additionally, real-time availability and the multitude of spatial data from an array of spatial sensing devices motivates their joint use with prior constructed synthetic terrain as backdrops to facilitate situational assessments. Traditional video, acquired by mobile sensor devices, and
synthetic terrain fusion methodology focus on projecting perspective trapezoidal video imageries onto the synthetic terrain. Simply mapping the rectified trapezoidal imageries over urban terrain populated with buildings and objects, however, is inappropriate for the urban environment due to scene complexity and high occurrence of inter-objects occlusions. Similar to the challenges faced when creating photorealistic buildings using oblique images, 3D mapping of 2D georeferenced images onto terrain geometries, taking into consideration of inter-object occlusions and viewing angles, is to be designed and utilized.

2.6.1 Texture Mapping For Terrain Generation And Visualization

While processes related to terrain generation and visualization seems to overlap or merged due to advancing techniques making real-time procedural terrain generation possible, an unique task central to terrain generation and visualization is texture mapping. Texture mapping could be performed manually or automated. The nature of the process depends on the precision required for texture mapping, and the nature of the datasets such as if the textures are tagged with geo-referencing information or if additional processing is required prior usage.

Considering a captured image as illustrated in Figure 2-28 with texture portions to be mapped onto the building geometries, typical manual approach involves several steps. They include having textures portions extracted, rectified and mapped onto polygonal surfaces (Figure 2-29). The process is laborious (Tan, 2009). The complexity of the mapping process is aggravated when a texture to be mapped onto a face requires additional considerations. For example, the object portions depicted in the texture is occluded or captured in two or more different images. Substantial efforts are required to extract relevant portions, scale and/or merge the image parts.
With 3D terrain and related modeling being increasingly essential for a variety of applications, there is a growing demand towards semi-, if not fully, machine-based interpretation systems for texture mapping. One key challenge involves merging images, which are often taken from various isolated views, to form composite texture prior usage. Another challenge includes relating texture parts to terrain geometries during the mapping process.
2.6.2 Image Mosaicking

Image mosaicking is used to piece together sections of images. In the past, image mosaicking was performed manually. Computational techniques to solve related problems are developed with recent advances made in informative technologies, and increasing availability of satellite imageries. Image mosaicking involves assemblage of partial images to form a composite picture. Central to image mosaicking is the process of image rectification and registration. Image rectification involves the process of capturing multiple imageries of the scene from different perspectives to be mapped onto common plane for image fusion. Image registration essentially consists of various steps (Zitová & Flusser, 2003). They include (1) feature detection to identify salient traits in images, (2) feature matching to determine correspondences between the features in the input and reference imageries, (3) transform model estimation followed by (4) image resampling and transformation. Mosaic of the acquired images is often performed. This is due to limited, instantaneous sensor views of the scene. It generates seamless mosaic of the scene to applied to terrain surfaces.

Mosaicking of the acquired images is often performed to aid in urban terrain texture mapping for terrain generation. Doing so allows perspective images to be rectified onto a common plane prior extraction of relevant parts. The resultant texture is to be applied onto building facades. Much research has been conducted in image mosaicking, particularly to form composite building facade textures. Various solutions involving rectifying images from dynamic or static viewpoints onto different planes are studied. (Zhang & Kang, 2004) rectify, mosaic and refine land-based video sequences onto vertical building facade plane. (Coorg & Teller, 2000), on the
other hand generate spherical mosaics from images acquired from a common optical center. (Nicolas, 2001) uses dynamic images to form mosaic representations of a scene.

In additional to forming composite images for building facades to aid in the virtual scene creation, image mosaicking is also used for situational awareness visualization, which has become increasing essential for a variety of applications. (Kumar, et al., 2001) generate mosaics for video compression, visualization, and archiving. (Se, Firoozfam, Goldstein, & Dutkiewicz, 2010) enhance situational awareness by mosaicking and producing high-resolution 2D map from sequence of video frames. Information from all the frames is displayed simultaneously for enhanced geographical situational awareness. The GeoReferenced Information Manager (GRIM) Enterprise Server (Brown, Gillbert, Holland, & Lu, 2006) generates an automatic mosaic at real-time using a spatial database integrated with sophisticated search tools to retrieve relevant sensor images.

### 2.6.3 3D Reconstruction From 2D Images

3D reconstruction from 2D images involves constructing 3D geometries using a series of images. A 3D scene, when projected onto a 2D image planes, loses the depth information. These depth information can be retrieved by comparing multiple images depicting similar features. Much research has been devoted to this problem in the context of city-scale 3D reconstruction (Antone & Teller, 2002) (Fr`uh & Zakhor, 2004) (Zebedin, 2008), and the findings of these research have been used in virtual city visualization systems such as Google Earth and Microsoft’s Virtual Earth (Agarwal, Snavely, Simon, Seitz, & Szeliski, 2009). The images are typically collected from a structured source such as sequences of aerial photographs or street side imagery captured
by moving plane or vehicle. These mobile vehicles are equipped with sensors such as GPS and Inertial Navigation Units. Availability of camera pose information greatly simplifies the reconstruction. Scene reconstruction in the absence of camera pose information is difficult, particularly when images are acquired from unstructured sources. The images may have been acquired using different cameras, varied timings, tagged with erroneous or lacking in geo-referencing information. Agarwal et al (Agarwal, Snavely, Simon, Seitz, & Szeliski, 2009) have proposed a 3D reconstruction method from collections of photographs from the web using a collection of parallel distributed matching and reconstruction algorithms. Such 3D reconstruction processes takes a collection of acquired images as input and iteratively refines the camera poses and 3D positional information of the scene visible in the images in an offline batch process. Incremental real-time updates and adjustments to the scene geometries base on latest inputs are difficult using such systems.

### 2.6.4 Transformations Between 3D Geometries And 2D Image Spaces

A geometric transformation from 2D image to 3D geometrical spaces is required to transfer texture color onto geometries. Transformations can be performed globally or locally. Global transformations refer to single models that apply to the entire image while local transformations are applied to a sub-part of an object or image. Figure 2-30 illustrates some common global transformations. Common global transformations include affine, perspective and polynomial transformations. Translation, scaled rigid, horizontal shear are examples of the affine transformations while terrain relief is an example of polynomial transformation (Gumustekin, 1999).
2.6.4.1 Forward Texture Mapping

The approach to perform a transformation from acquired 2D image space to 3D model space is termed forward texture mapping (FTM) by (Tan, 2009). Traditionally adopted by graphic artists in modeling 3D buildings, texture mapping using FTM is typically performed manually. FTM does not require acquired images to be geo-referenced. Image mosaicking is carried out when texture portion of interest exists in different images. Computation of registration points for image mosaicking, however, is time consuming. These registration points are usually acquired through correlation maximization functions performed iteratively. Depending on image qualities, human assistance may be required to ensure valid registration points are identified. Human involvement is also necessary for tasks related to texture portion extraction and perform mapping onto respective building facade. The process, though straightforward, is laborious. The time taken to texture a building can easily amount to an hour (Tan, 2009).

2.6.4.2 Reverse Project Texture Mapping

Driven by advances in digital imaging and direct-geo-referencing technologies, coarse transform model estimation between 2D acquired image space and 3D scene model space could be derived
without performing image feature extraction and matching (Enhancing GPS Accuracy, 2010) (Han, 2008). With approximate intrinsic and extrinsic parameters belonging to the camera made available and availability of geometrical scene constructs due to advances in sensing and reconstruction techniques, transforming spatial information from 3D scene model space to 2D acquired image space to aid in texture mapping becomes possible without prior image processing. The approach to aid in texture mapping is termed reverse project texture mapping (RPTM) in (Tan, 2009). Likewise, to overlay UAV imagery onto 3D terrain models, (Owens, Sycara, & Scerri, 2009) use an inverse transform approach which involves reading each and every pixel in the destination image and perform an inverse transform back to the source to sample colors. To perform facade texture extraction for RPTM, fine-alignment and recovery camera pose information is vital. (Tan, 2009) demonstrates that pose estimation can be performed in an interactive way by adjusting the orientation and location input parameters. Unlike FTM, which demands significant computational resources to aid in image alignment, texture portion extraction and texture mapping onto respectively building facades, with accurate camera transform model and scene geometric constructs, RPTM could be performed in a speedy manner.

2.6.5 Mapping Texture Onto Terrain Geometries

While the aforementioned transforming process involve global transformation between 2D image and 3D terrain spaces, local transformations due to variations among 3D objects are not taken into consideration. Figure 2-31 illustrates two rectified unmanned aerial vehicle (UAV) acquired images on a horizon plane orthogonally. Using the assumption that depth variation is relatively
low when viewed from high altitude with camera viewing angle pointing downwards, the acquired images are projected onto common ground plane.

![Figure 2-31. UAV Camera Viewing Frustums (Left) And Rectified Orthogonal Images (Right)](image)

With camera viewing angle pointing downwards, distortion due to differences in heights is considered minimal. When viewed from an oblique angle (Figure 2-32), however, image distortion is magnified.

![Figure 2-32. UAV Oblique Camera Viewing Frustum](image)

There are shadowed regions due to presences of 3D structures such as buildings due to inter object occlusions. This results in occluded shadowed regions that have no line of sight with the
UAV camera. When projected onto the common ground plane, misleading situational assessment may be made. For example, a target vehicle travelling on a road between two skyscrapers when viewed at a slight oblique angle could be well-hidden. When such view is captured by a sensor and respective image information is projected over a common ground plane, a misleading conclusion would be made. To remedy, terrain geometries in 3D space and inter-object occlusion need to be considered when fusing static synthetic terrain data and real-time imageries. Imageries must be applied only to areas with direct camera line of sight when superimposing imageries and urban terrain data.

### 2.6.6 Leveraging On Computer Graphics Techniques For Terrain Texture Mapping

Techniques related to the computer graphics rendering capabilities could be exploited to aid in texture mapping onto terrain geometries.

#### 2.6.6.1 Shadow Mapping

(Sawhney, et al., 2002) presented a solution to render multiple live videos in real-time over 3D models for global awareness by exploiting the real-time rendering capabilities of graphics hardware. The approach renders live videos captured from various views of the environment. The rendering algorithms exploited shadow mapping and projective texturing techniques commonly used in computer graphics (Segal & Korobkin, 1992) (Heidrich, 1999). The techniques are originally designed for shadows and lighting effects. Similar approach is adopted by (Neumann, You, Hu, Jiang, & Lee, 2003) to simultaneously visualize various images projected on common spatial model for enhanced situational awareness. The approach
implements shadow mapping, commonly used in computer graphics, by taking advantage of projective texturing, depth buffering, automatic texture generation, texture compositing and alpha testing (Sawhney, et al., 2002). The algorithm involves creating z-buffer for each surveillance sensor by rendering the scene from the sensors' viewpoint and subsequently rendering the scene from the users' virtual view point and, at the same time, replacing pixels that are "illuminated" by or have line of sight with the surveillance sensors. Figure 2-33 illustrates occlusion handling by (Sawhney, et al., 2002) to render video images from a virtual viewpoint. The flashlight camera uses virtual location that corresponds to the location of the sensor camera. Two depth values for each point in the scene are used for occlusion handling to render multiple live videos. The first depth value corresponds to the z-buffer depth from the sensor camera to the first object intersected. The second depth value is the true depth value for each scene point with respect to the sensor camera.
Nevertheless, without updating the terrain textures permanently, the proposed projective texturing approaches fail to achieve persistent terrain updates. Analogous to flashlight projector systems, projective texture mapping casts captured images onto 3D objects using camera views as projection centers. Unlike image mosaicking where multiple views of the video images at different time frame could be displayed simultaneously, the mapping solutions adopted by (Sawhney, et al., 2002) and (Neumann, You, Hu, Jiang, & Lee, 2003) display information only for current video frames. The instant when sensor viewpoint shifts to a new focus area and moves away from any areas of interest, previous data are removed. While their proposed systems successfully illustrate 3D spatial relationships of activities captured by different sensors,
sequences of activities captured by sensor views at different time frames could not be displayed simultaneously. Imageries captured by a sensor scanning and combing an area when viewed concurrently could potentially provide more insights on targets' intents such as collective behaviors, group movements, and any spatial interactions in contrast to when viewed in isolation.

When video frames are displayed sequentially using "soda straw" views, operators need to rely upon the recollection of past events and activities for analysis. One promising solution to present historical imageries persistently is to update synthetic terrain with geo-referenced imageries. The approach requires synthetic terrain to serve as "base-skeleton". The synthetic terrain is to be skinned to latest acquired imageries. Such method involves fusing and updating terrain textures using real-time geo-referenced images.

2.6.6.2  Rasterization

A method to relate pixels from 2D image to 3D terrain space is to be established to fuse perspective images and update terrain textures. (Owens, Sycara, & Scerri, 2009) use an inverse transform approach to read each and every pixel in the destination image and perform an inverse transformation back to the source to sample colors. The process involves sampling colors from acquired imageries and mapping them onto terrain textures. Pre-constructed terrain is used to render a virtual scene. A list of texel points contained by each triangle is constructed and assigned to the triangle for texture mapping. Using inverse transform approach, the triangles are transformed from the 3D world space and rasterized to the 2D display space. Each texel point in a triangle stores the texture coordinates and corresponding 3D world space coordinates.
In Figure 2-34, fusing UAV imageries into 3D synthetic terrain is achieved by encoding georeferencing information into the synthetic terrain mesh and textures using texture texel values. The resultant terrain contains polygons with textures with texels storing respective texture coordinates and corresponding 3D world space coordinates. For each input image to be baked or "burned" into the terrain textures, relevant terrain mesh triangles and textures are computed by projecting all terrain mesh triangles using the input images perspectives. During the texture baking or image updating process, for each terrain texel that are unfiltered by frustum culling and occlusion test, corresponding texel color from the input image is fetched to be written on the terrain texel (Owens, Sycara, & Scerri, 2009).

Figure 2-34. Overall Rendering Pipeline (Owens, Sycara, & Scerri, 2009)
2.7 Summary

This chapter provides some pertinent background information related to the research. Details on the current state of the art terrain rendering and image fusion techniques are described. Some limitations of current techniques are also discussed. The chapter commences with section 2.1 that presents information on situational awareness, importance of information visualization to aid in decision making, and techniques to measure situational awareness. It moves on to section 2.2, which introduces the programmable graphics pipeline. The programmable graphics pipeline has made possible programmable flexibility to a graphics pipeline. Section 2.3 describes ray tracing and rasterization, the two main graphics rendering techniques to produce 3D virtual scenes.

With increasing activities carried out in the urban environment, providing visualization and understanding of the terrain data at hand and usage of high resolution data for analysis are crucial for planning and decision making. In section 2.4, various techniques and challenges to rendering synthetic urban terrain are presented and discussed. Current state of the art polygon based terrain rendering solutions result in reduced geometrical accuracy for selective terrain features that are considered less critical to the visualization or simulation needs. Likewise, computational intensive tasks involved in terrain-related calculations such as viewshed analysis are traditionally performed at a non-interactive rate. Chapter 3 presents literature review and the approaches that leverage on GPU ray tracing to achieve accurate visualization and improved understanding of urban terrain using enhanced ray cast BlockMap and viewshed rendering.

Lastly, section 2.6 presents and discusses the current state of the art fusing solutions for 2D images and 3D terrain. Current state of the art solutions essentially belongs to two groups. Each
strives to either provide near real-time situational pictures in 2D or off-line complex 3D reconstruction for subsequent usages. Chapter 4 presents literature review specific to this work and the solution to fuse geo-referenced images onto 3D terrain at interactive rate. The solution allows the presentation of real-time terrain rendered with latest situational pictures, using most current surveillance images.
CHAPTER 3  
RAY TRACING FOR ACCURATE VISUALIZATION 
AND IMPROVED UNDERSTANDING OF URBAN TERRAIN

Current state of the art polygon-based terrain representations often use coarse representations for terrain features of less importance to improve rendering rate. This results in reduced geometrical accuracy for selective terrain features that are considered less critical to the visualization or simulation needs. Alternatively, to render highly accurate urban terrain, considerable computational effort is needed. A compromise between achieving real-time rendering rate and accurate terrain representations would have to be made. Likewise, computational tasks involved in terrain-related calculations such as viewshed analysis are highly computational intensive and are traditionally performed at a non-interactive rate. Exhibiting an intrinsic parallelism in the form of independent rays, ray casting and tracing can take advantage of the parallel architecture of the GPU.

This chapter presents approaches that leverage on GPU ray tracing, a rendering approach, conventionally not employed in the simulation community in favor of rasterization, to achieve accurate visualization and improved understanding of urban terrain. The efficiency of using GPU ray tracing is demonstrated in two areas, namely, to depict complex, large scale terrain and visualize viewshed terrain effects at interactive rate. Hybrid rendering approaches using both rasterization and GPU ray tracing are adopted to achieve accurate visualization and viewshed rendering of urban terrain. The first half of the chapter delineates the approach to achieving highly accurate terrain representations for interactive simulation using BlockMap, a simplified representation of the distant city blocks rendered using GPU ray casting. The latter part of this chapter describes another approach to adopt ray tracing, a technique widely used for realistic
rendering by computer graphics community, to render multi-view real-time viewshed analysis to aid in situational understanding of terrain effects. Such potentially helpful real-time viewshed analysis is lacking in most decision support applications.

3.1 Enhancing Ray-casted BlockMap For Accurate Urban Terrain Representation

The section of this chapter commences with delineating the benefits of employing BlockMap representation (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007) (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009) for virtual simulation to achieve both high rendering rate and geometric accuracy for urban terrain representation. Limitations of current BlockMap implementations for simulation applications are also delineated. Enhancements are made to overcome these limitations of current BlockMap representation. The following list records improvements made with Enhanced BlockMap representation and rendering approaches.

1. Represent and visualize simulation entities and BlockMaps together in common spatial space by computing and updating depth information for each and every BlockMap surface fragment (to be described in section 3.1),

2. Adopt a more flexible utilization of BlockMap by representing individual, complex buildings, and assigning different LOD switch values for different buildings to optimize computational resources and improve rendering performance (to be described in section 3.2) and

3. Accelerate ray casting process for BlockMap rendering to achieve high rendering rate (to be described in section 3.4).
The efficiency of aforementioned techniques is demonstrated by integrating the Enhanced BlockMap representation with a large OpenFlight urban terrain. Real-time rendering performance of proposed rendering techniques, implemented using a GPU shader-based OpenSceneGraph rendering application, is evaluated. Geometric accuracy of terrain using Enhanced BlockMaps for distant city block representations is also measured.

3.1.1 Using BlockMap For Interactive Simulation

Typical urban terrain represented using polygon based data necessitates generation of a substantial amount of geometrical data to depict the scene accurately. Compromises to terrain representation accuracy and rendering rate need to be made. This is due to limited computational resources to satisfy the insatiable demand to represent geo-specific and highly detailed models within urban scenes. Figure 3-1 illustrates a typical urban scene modeled with high geometrical accuracy which comprises of 739,000 polygons rendered at 20 Hz.
To maintain high rendering rate for real-time visualization, allocating more computational resource power to terrain features of importance, such as those nearer to observer viewpoint is commonly done by simplifying or omitting rendering objects of lower importance. This is usually accomplished by switching out or simplification of distant 3D models within the terrain. Figure 3-2 illustrates similar urban scene depicted in Figure 3-1 with three levels of details. The distant objects are simplified to reduce the number of polygons. Doing so allows the scene to be rendered at a 30 Hz. The number of polygons present in the scene also reduces from 739,000 to 292,000.
While simplification of polygons belonging to distant 3D objects aids in reducing computational processing, switching geometrical representation of the urban setting leads to erroneous or misleading visual cues. An object behind a building, for instance, obstructed from view may become exposed if the visualization system has switched to lower LOD model where the building block is no longer rendered and presented in the urban scene. Unlike real–time 3D exploration of urban terrain where geometrical simplified distant objects provide approximate but reasonably compelling visual cues of terrain information, in simulation where virtual entities are to be placed in the scene, geometric accuracy is especially crucial to depict spatial relationships among the 3D objects. The level of spatial and imagery accuracy required for urban terrain representations depends on the context of the problem and related tactical actions that are planned. Snipers related simulation, for example, typically requires simulation using...
virtual terrain with high spatial and imagery accuracy while helicopter rescue simulation requires low spatial but high imagery accuracy. Blast damage assessment analysis, on the other hand, usually requires high spatial and low imagery accuracy (Pigeon, 2002). With increasing demand for correlated terrain for federated simulation, the ability to share and provide visualization of similar urban terrain with a high level of spatial accuracy and textures at interactive rate seems attractive. Providing urban terrain with a high level of spatial accuracy allows simulation entities to share similar notion of locations of environmental elements. Doing so also allow the simulation entities to execute interaction in a correlated environment.

While adopting simplification of polygons belonging to distant 3D objects aids in reducing computational processing and providing good visual relationships among terrain details, detail spatial accuracy is compromised. To assess spatial accuracy of the urban terrain representation when different LODs are adopted, numerous red spheres are placed randomly on the terrain and their amounts of occlusions due to terrain objects are measured. The result is compared to that when the red spheres are placed on similar terrain when full geometric details are included. Figure 3-3 illustrates image portions of respective terrains populated with randomly placed red spheres.
Image analysis is then performed on screen captures of the terrains. The numbers of red pixels, that is, un-occluded red portions of the red spheres are found to be 9331, 7136 and 8642 for the terrains without 3D objects, with full geometrical detailed 3D objects, and with LOD integrated and distant simplified 3D objects respectively. The percentages of occluded portions for red
spheres are 23.5 and 7.4 for full geometrical details and LODs integrated versions respectively. Thus, 16.1% of red spheres that are supposed to be occluded become exposed due to object simplifications. Though visual difference between displays of the terrains with full geometrical details and LOD integrated simplified version is minimal, degraded spatial accuracy is evident when LODs integrated version is used.

One promising solution to achieve visualization of urban terrain with a high level of spatial and imagery accuracy involves using an enhanced version of the BlockMap. The BlockMap (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007) is a GPU-friendly simplified representation of urban environments that could be used to encode coarse representations of blocks of terrain buildings when rendering far-away city blocks. They are stored in textures and rendered through GPU ray casting. BlockMap rendering requires minimal polygon based data while preserving the ability to portray building heights and wall textures. The approach exploits the advantages offered by traditional polygon based representation for close view objects, and image based rendering approaches for far away objects, in rendering large scale urban terrain. BlockMap possesses many characteristics that make it well suited for use in current state-of-the-art visualization system. The subsequent sections of this chapter delineate some enhancements made to current BlockMap implementations for use in visual simulation system.

3.1.2 Providing Depth Information For BlockMap

Unlike real–time 3D exploration of urban terrain where geometrical simplified distant objects provide approximate but reasonably compelling visual cues of terrain information, in simulation
where virtual entities are to be placed in the scene, geometrical accuracy is especially crucial to depict spatial relationships among the 3D objects. While BlockMap representation facilitates real-time visualization of virtual cities, without modeling the geometrical constructs of the virtual terrain explicitly, inaccuracy results when the scene to be rendered comprises of both polygon-based objects and BlockMap representations that share common spatial space. Figure 3-4 illustrates the rendering effects when polygonal spheres are planted within a city block rendered using BlockMap. It is observed that sections of the spheres are occluded by the bounding box of the BlockMap. The green sphere, for example, is entirely occluded by the BlockMap’s bounding faces. Erroneous visual effects results when polygon based objects, such as simulated entities, are placed in the scene.

Figure 3-4. Visual Effect When Polygonal Spheres (Left) Is Planted Within A City Block Rendered Using Standard BlockMap (Right)

Figure 3-5 illustrates the side view of the ray casting process of a BlockMap with a polygonal 3D object (represented by a red sphere) planted in the scene. To handle the visibility problem in selecting which polygonal faces to render, z-buffer based technique is used. The technique renders terrain portion represented by the BlockMap since its faces are nearer to observer viewpoint as indicated by the blue, dotted lines. The objects indicated in black bold lines are
visualized. This paints an erroneous picture where the yellow portion of the red sphere is occluded by the objects contained in the BlockMap despite the fact that the “real” position of the buildings or terrain features is further away from the observer viewpoint than the portion of the red sphere.

![Point of view](image)

*Figure 3-5. Rendering BlockMap With A Sphere In Scene*

To remedy, the actual depths of the first visible surfaces within the BlockMap are computed using the hit positions of the buildings and object surfaces encoded within the BlockMap during the ray casting process. The calculated values are used to update the depth values of the respective pixels. In Figure 3-6, the blue, dotted lines indicate the updated surfaces of the BlockMap to be rendered and visualized. An accurate depiction of the scene resulted with the pixel depths for the BlockMap faces updated using actual real-world locations of the terrain features. The portion of the red sphere that is supposed to be exposed to the observer would be rendered “in-front” of the buildings. The black bold lines indicate surfaces that would be
rendered. Polygon based representation objects such as simulated entities moving into BlockMap spatial space could be rendered accurately consequently.

![Point of view](image)

**Figure 3-6. Rendering Objects With BlockMap With Depth Correction**

Figure 3-7 illustrates the visual effects of rendering 3D spheres of different sizes and BlockMap terrain tile representation in common spatial space, without and with the proposed depth correction implementation. It is observed that the spatial relationships and occlusions among the BlockMap terrain features and that of the spheres are presented accurately after using the proposed depth correction implementation. Sections of the spheres, erroneously occluded when rendered without BlockMap depth correction are correctly rendered consequently. For example, the smallest sphere indicated in green become exposed and rendered correctly above terrain ground when BlockMap depth correction is performed.
Similar urban terrain scene populated with red spheres for occlusion test in section 3.1 is also rendered using BlockMaps. In the rendering, distant city blocks are enhanced with depth correction. The image portion is as illustrated in Figure 3-8.

Image analysis is then performed on screen capture of the terrain using BlockMaps. The numbers of red pixels, that is, un-occluded red portions of the red spheres are found to be 7070. The percentage of occluded portions for red spheres is 24.2. This is much closer to 23.5% of
occluded portions when full geometric details are used, as compared to only 7.4% of occluded portions when 3 LODs are used. As such, it is concluded from the evaluation that terrain using Enhanced BlockMaps for distant city block representations can accurately model inter-object occlusion effects.

3.2 A More Flexible Approach For BlockMap Implementation

Providing accurate spatially represented BlockMaps allows polygon based simulation entities and BlockMaps that are sharing common spatial space to be rendered correctly. In addition, updating valid spatial information for the BlockMap terrain surfaces facilitates co-existence of polygon based terrain 3D objects and BlockMap representations. This allows utilization of a flexible BlockMap rendering approach for select geometrically-dense sections of the terrain. For example, BlockMap representation for selective complex buildings can be adopted while polygon based representation for relatively polygonally less demanding terrain surfaces is retained. Extending the image based rendering approach adopted by BlockMap rendering, similar rendering methodology is applied to individual buildings. This section highlights the significance and advantages of equipping with the ability of rendering individual buildings using BlockMaps. The key benefits of employing BlockMap representation for individual buildings are as follow.

1. Promote sharing of geo-typical building models in scene, thus optimizing usage of texture memory space.

2. Improve rendering performance by assigning unique LOD switching distances to individual buildings, which in turn, facilitates a more flexible distribution of the
rendering computational resources to address asymmetric scene complexity distribution innate to the urban scene.

3. Encode building wall depth information onto BlockMap representation.

3.2.1 Optimizing BlockMap Texture Usage

To isolate different modeling processes and promote efficient memory usage and model reuses, external referencing mechanism is often used during urban terrain modeling and rendering process. These external references allow proxy components within the scene to reference to other 3D models. Using external referencing facilitates easy viewing of an entire scene while minimizing memory usage by loading only portions of relevant scene during modeling and rendering process. When external referencing is used in scene graphs during runtime rendering, only one copy of the model is loaded and maybe reused with multiple instances of the 3D model. Such approach is useful when the urban terrain is populated with numerous similar building types.

Figure 3-9 illustrates terrain populated with cluster of similar geo-typical buildings. A typical scene graph optimized for texture memory usage is shown on the right where one copy of the geo-typical building is loaded.
For BlockMap construction, however, should the terrain be divided into $n$ number of tiles using standard BlockMap design, $n$ BlockMap textures are required. On the other hand, if BlockMap representation is used for individual building, only one BlockMap texture for the geo-typical building is required for the entire terrain (Figure 3-10).

*Figure 3-9. Top View Of A Terrain Populated With A Cluster Of Geo-Typical Buildings (Left) And Corresponding Terrain Scene Graph (Right)*

*Figure 3-10. BlockMaps For Terrain Divided Into n Tiles (Left) And Individual Buildings Sharing A Common Reference Model (Right)*
Figure 3-11 illustrates the scene graphs for BlockMap representation using standard BlockMap for \( n \) city blocks and using a common reference building model.

Since terrain skins are relatively flat, polygon based representation for terrain skins can be retained even for coarse representation. Such design is analogous to typical scene graph populated with instances of similar external references of a single building copy as shown in Figure 3-9. Similar scene graph could be used. In the scene graph, multiple instances of building referencing to a common building 3D model is used. The model is constructed with different LOD versions, consisting of polygon based representation for high LOD and BlockMap for low LOD representation (Figure 3-12).
3.2.2 Improving Rendering Performance

As rendering BlockMap requires activations of ray casting operations for respective screen pixels spanned by the rendering objects on screen, computation complexity for BlockMap rendering depends on the screen footprint spanned by the BlockMap bounding box of the represented object rather than the object’s complexity. It could be observed from Figure 3-13 that while rendering complexity of a polygon based 3D object is invariant to its distance from the camera viewpoint, rendering complexity of a 3D object represented using BlockMap representation decreases as the distance from the camera viewpoint increases. There is a switching distance from the camera viewpoint beyond which representing the object using BlockMap is computationally less demanding than using polygon based representation.
Different buildings shall have different optimal switching distances due to different complexities unique to each 3D model. Figure 3-14 illustrates how the BlockMap switching distance for each building model is affected by the polygon count and BlockMap screen footprint.

This necessitates the need for using different switching distances for different 3D models during real-time visualization so as to ensure optimal efficiency in computational resource utilization.
Current BlockMap design for terrain block representation employs common and universal switching distances for all terrain blocks. The approach fails to address potentially asymmetric nature of the terrain polygonal distribution when rendering the urban environment on the screen. This is especially so for terrains where scenes comprise of patches of open spaces populated with clusters of towns or scattered buildings with varying polygonal complexities.

Figure 3-15 illustrates the rendering of a prototype terrain populated with buildings of varying polygonal complexities using full polygon-based (left) and hybrid (right) representation. The screenshot on the right illustrates the current approach for distant terrain tiles to switch to potential BlockMap representations (indicated in red). More terrain tiles, achieved by using BlockMap representations for distant city blocks, could be rendered consequently. The number of polygons also reduces from 3 to 2 million when hybrid representation is employed.

![Figure 3-15. Prototype Terrain Populated With Buildings Of Varying Polygonal Complexities Represented Using Full Polygon Based (Left) And Potential Distant City Blocks In BlockMaps (Right)](image)

Drawing on the aforementioned method, described in section 3.1.2 in updating the BlockMap depth information to handle occlusions and achieve co-existence of polygon based and BlockMap representations in similar spatial space, BlockMap representations could be applied to separate, individual buildings with different switching distances assigned to different object models. When appropriate switching distances are assigned, a more optimal distribution of
computational resources for scene rendering is achieved. In such terrain scenes, BlockMap representations are activated when building models occupy “small enough” screen areas to become computationally favorable to be represented by BlockMaps while larger or minimal polygon based objects are rendered in polygon based forms.

Figure 3-16 illustrates the visual effect of utilizing a more flexible representation approach where distant terrain tiles are rendered using image-based BlockMap representations, as indicated within the boxes in red, while terrain objects close to observer viewpoint are rendered using polygon based rendering approach. For moderate distant, complex building terrain models, BlockMap rendering is adopted only for polygonally demanding objects. The regions of the terrain in the transitional zone are rendered using such mixture of varied representations. They consist of terrain skins and less polygonal demanding buildings, rendered using polygons, and relatively more geometric complex buildings, rendered using BlockMaps. A more optimal distribution of computational resources for scene rendering is achieved consequently. The number of polygons used reduces to less than a million consequently.

![Figure 3-16. Prototype Terrain Adopting Selective BlockMap Representations For Polygonal Demanding Buildings](image)
3.2.3 Encoding Wall Depth Information

An additional advantage of adopting BlockMap representations for individual buildings is the ability to encode building wall depth information on the BlockMap texture. The details are elaborated in section 3.4.2.

3.3 Large Scale Urban Terrain Prototype Implementation

This section of the chapter delineates my implementation approach to utilize BlockMaps to aid in representing and rendering a large scale urban terrain. The OpenFlight file, used for my work, models the city of Boston over an area of 10 km x 10 km (kilometer) and contains information of a heavily populated urban terrain organized in a grid tiling structure. The terrain consists of over 60,000 buildings extruded from 2D footprints. Satellite imagery with 0.5 m (meter) resolution for roof and the ground is draped over the building polygons while generic buildings wall textures are mapped over the building sides. Two polygon based represented versions of the terrain would be used for comparison with the BlockMap enhanced terrain. The first contains only one LOD with all geometric details modeled. The second uses three LODs as illustrated in Figure 3-17.
Three terrains are considered for the purpose of this dissertation. They include the followings:

1. Full polygon based with detail geometries,
2. Full polygon based with three LODs and
3. Hybrid of polygon based for the highest LOD and BlockMap representation for lower LOD
3.3.1 Enhanced BlockMap Implementation For Distant City Block

As illustrated in section 3.2, enhanced BlockMap with depth-corrected surfaces provides an attractive alternative to traditional polygon based representation as coarse representation for distant city blocks. They aid in achieving high spatial, geometric accuracy without compromising rendering rate. This section of my dissertation conveys the implementation approach and some proposed solutions to enhance standard BlockMap data structure.

3.3.1.1 Data Structure

Figure 3-18 illustrates my enhanced BlockMap data structure adopted for distant view rendering.

![Figure 3-18. Enhanced BlockMap Data Structure Encoded in a 1024x128 24-bit RGB Texture](image)

The approach enhances the original BlockMap data structure (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007) (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, &
Scopigno, 2009) by adding additional features to encode more detail wall surfaces and adopt accelerated ray casting algorithms to improve rendering efficiency.

The following list contains enhancements made to the original BlockMap data structure:

1. Improved indexing system to encode more wall columns information
2. More columns of wall color texels encoded
3. Additional minimal object surface distance map from the ceiling for ray casting acceleration

The first 128×128 texels contain color information for building roofs/ground. The next 128×128 texels contain information for building height map, wall offset map and distance map, each stored in a different color channel. The approach to encode BlockMap data is described in section 3.3.1.2.

While performing GPU-based ray casting, the height map values are referred to determine if the ray hit a roof or wall. If the roof or ground is hit, the corresponding roof or ground texel is rendered. If a building wall is hit, the value from wall offset map is used to access a texel from the next 512×128 texels which contains relevant wall color information for rendering building sides. Extending the BlockMap data structure where a BlockMap is sliced into four parts horizontally, each with its own parameterization, further subdivisions are done to the tile vertically to obtain 4×2 sub-tiles. A wall offset map is constructed to perform wall offset mapping. The wall offset map is divided into 4×2 sub-tiles, each of them containing wall offset values, ranging from values of 1 to 255. The wall offset map is used to fetch the wall texel
colors from respective 256×32 sub-wall texture spaces for buildings located in corresponding sub-tile locations.

Dividing the tile into 4×2 sub-tiles allows more wall information to be encoded onto the BlockMap data structure. The next 128×128 texels store the surface normal map used for shading effect during rendering. Similar to the original BlockMap data structure (Cignoni, Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007), the distance map stores the minimal inter-building distance for each location represented by respective texel. This information is used to speedup ray casting by space leaping (Donnelly, 2005). The last 128×128 texels contain additional data, added to the original BlockMap structure. They are used to accelerate the ray casting process. They encode closest surface distance from the BlockMap ceiling for space leaping.

3.3.1.2 BlockMap Construction

One challenge encountered during BlockMap texture formation arises when constructing BlockMap texture at high resolution. Valid but unregistered wall texture color may result due to screen capture resolution limitation and occlusion (Figure 3-19). This shortfall is addressed by filling up unregistered wall texture using neighboring wall texel colors. The approach is described in step 9 for the enhanced BlockMap encoding process.

*Figure 3-19. Valid But Unregistered Wall Texture Colors In Standard BlockMap Due To Sampling Limitations*
To accommodate geometrically complex building facades in BlockMap encoding process, a two-pass batch encoding process is adopted. The process approximates and projects irregular wall surfaces onto vertical prisms to allow representation of terrain tiles populated with intricate complex buildings. The multiple-view sampling approach used by (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009) was adopted. The two pass batch-processing approach allows heights and wall locations to be established prior inserting information for wall textures. The approach does away with the assumption made for building walls to be perfect vertical prisms. Indented wall surfaces are projected onto approximated walls during the second processing pass.

In the encoding process, the roof/ground and height texels information is first constructed using roof/ground color and positional data captured from all views. Another rendering pass is performed using the height information obtained from the first pass. The second rendering pass captures the wall offset map addresses to be used to fill up the wall texture information onto the BlockMap data structure. The process to construct the BlockMap texture is as delineated below:

1. Obtain a set of images of the terrain tile to be represented by the BlockMap from multiple views taken from camera positioned at locations distributed on the hemisphere with the geometry block placed at the center as shown in Figure 3-20. The square boxes indicate camera positions from which the scene would be rendered.

2. For each corresponding view and using similar viewing settings of the camera positions used in step 1, the spatial and surface normal information of the geometries is captured and stored.
3. Using the spatial information captured in step 2, for each discrete point on the x, y plane of the geometry block, the maximum height is determined and stored in the height map. The texel colors from images obtained in step 1 that correspond to the maximum height for each x, y coordinate are then used to construct the first 128×128 texels of the BlockMap for building roofs/ground (Figure 3-21).

4. Using the surface normal information captured in step 2, the surface normal map is constructed.

5. Using the height map, a distance map is computed that stores the minimal inter-building distance field.

6. An edge detection filter is then applied to the height map to detect sudden changes in height, indicating the presence of a wall structure. The 128×128 texels area depicting the
edges is divided into 4×2 tiles, each with their own set of indices to be used to populate the wall texture texels portion of the BlockMap as described in Steps 7-8.

Figure 3-21. Construction Of Enhanced BlockMap Texture

7. Using information obtained from step 3 and 6 and for each corresponding view with similar viewing settings used in step 1, the BlockMap is ray cast using the GPU ray casting process described in section 3.3.1.1. The height map (obtained from step 3) is used to determine if the ray hits any obstacle. Whenever a ray hits a wall structure, the hit location is used to access the wall offset map. For each fragment, instead of using the wall offset address obtained from the wall offset map to fetch the wall texel color as in
the actual BlockMap rendering process, the wall offset value is rendered onto the fragment. The results from step 1 and 7 are subsequently used to build the wall texture portion of the BlockMap.

8. The wall texture space of the BlockMap is then filled using the color information from step 1 and corresponding address information obtained in step 7. For each and every texel from pairs of images obtained from step 1 and 7 with similar viewing setting, the wall offset address is retrieved by reading the texel value obtained from step 7. Then, the respective texel color obtained from step 1 is written onto the wall texture portion indicated by the wall offset address. In the case where an address has more than one corresponding color information (occurs when a particular patch is captured by various views), the contribution of all colors is averaged and written onto the address location (Figure 3-22).
9. Due to sampling resolution limitation or inter-building occlusion, some wall texture texels may not be sampled. To fill up any valid but empty space, the neighboring wall texture texel colors are utilized. Figure 3-23 illustrates the wall texture texels before and after filling up valid and empty spaces using neighboring wall texture texel colors as indicated in the top and bottom texture screenshots respectively.
10. Additional processed data are stored in the BlockMap texture to aid in speedy ray-casting process. The processed data involves computing the closest surface distance map from BlockMap ceiling for space leaping.

The aforementioned approach to generate coarse representations of city blocks for coarse LOD representations requires minimal human inputs. Construction of BlockMap data, using the aforementioned LOD construction process, for a single tile using 100 screenshots of the scene at 512 by 512 pixels requires less than few minutes. As the time taken for BlockMap encoding is relatively invariant to the geometrical complexity, the potential to achieve multiple-fold improvement in processing time is especially promising for complex terrain. Figure 3-24 illustrates the visual effect of a city block rendered using BlockMap and polygon based representation.
3.3.2 BlockMap Representation For Individual Building

This section describes the data structure used for individual building BlockMap representation and delineates the process to construct BlockMap for individual building. Representing individual buildings allow complex geometrical structures, such as curves, columns and arches, to be registered during the encoding process.

3.3.2.1 Data Structure

Figure 3-25 illustrates the BlockMap data structure. It is used for the rendering of individual, complex buildings. The 768×256 24 bit RGB texture contains geometrical information for an individual, complex building. The first 256×256 texels contain color information for building roof. The next 256×256 texels contain information for building walls textures. The last 256×256 texels contain height map, wall offset and depth maps encoded in each of the color channels. Similar to the original BlockMap rendering approach for terrain tiles, while performing GPU-based ray casting, the height map values are referred to determine if the ray hit a roof. If a roof is hit, the corresponding roof texel color is rendered. If a wall is hit, the
corresponding value from the wall offset map is used to determine which specific wall is hit. Unlike the original BlockMap rendering, instead of simply fetching the wall texture texel for rendering, additional step is required to fetch the respective wall depth value to be used to determine if additional ray casting is required to acquire the final texel color to be read from the wall color map for rendering.

![BlockMap Data Structure For A Single Complex Building](image)

**Figure 3-25. BlockMap Data Structure For A Single Complex Building**

### 3.3.2.2 Data Capture And Encoding Process

The first pass to encode BlockMap texture for external reference buildings is similar to the steps performed to encode BlockMap for far away city blocks. However, in addition to computing the maximum height for the geometry block, depth information for each wall surface is computed. Just as in the encoding process for distant city block, an image processing approach is applied on
the height map. The image processing detects edges to be used to parameterize the wall offset map for their references during BlockMap ray casting.

Figure 3-26 illustrates the BlockMap rendered with depth information encoded for building facade for a 3D building model as illustrated in Figure 3-27. In addition to the original BlockMap rendering approach that conveys building height information, building facade details are also portrayed.

![Figure 3-26. BlockMap Rendering For Individual Complex Building](image)

Figure 3-27 illustrates the various visual effects from different combination of encodings and ray casting algorithms. The first image from the left displays the screen capture of the original building using polygon based representation. The second image illustrates the effects of the encoding algorithm to approximate and project wall information onto vertical prism for far distance viewing. The last image displays the rendering effects with building wall depth encoded for moderate distance display. It could be observed that most of the prominent building facade features are faithfully preserved. Slight visual deviations from actual geometric details of
the building are evident. Nevertheless, when used as distant presentation, BlockMap achieved its primary goal, that is, to provide coarse representation so as to fill up empty but valid space while achieving interactive rendering rate.

Figure 3-27. Geometrical Representation (Left) And BlockMap Representations For Far Away (Middle) And Moderate Distant (Right) Viewing

3.3.3 Terrain Prototype Implementation

The aforementioned rendering techniques were implemented using a hierarchical tile-based scene graph defined in OpenFlight™ format. The OpenFlight format is the defacto industry standard format used in visual simulation applications. The OpenFlight file stores the logical, hierarchical scene description to be used by image generators for real-time 3D scenes rendering. The OpenFlight format consists of "nodes" such as LOD, SWITCH, EXTERN that are used for real-time simulations. The LOD node stores values for the "switch in", "switch out" and "center location" parameters. The "switch in" and "switch out" values define the range of distances away from the rendering camera point during which any child node data of the LOD node would be displayed. The "center location" defines the reference location of which the distance from the
camera rendering point is measured. Figure 3-28 illustrates the hierarchical tree structure with LOD node (left) and the corresponding rendering effects (right).

![Diagram of LOD Node in OpenFlight File]

Figure 3-28. LOD Node In OpenFlight File

The OpenFlight file used to illustrate the BlockMap rendering approaches is divided into 250×250 m terrain blocks, each to be switched in at a 10 km from the camera rendering point. This is accomplished by dividing the terrain into 250×250 m tiles and adding each as a child of an LOD node with “switch in” and “switch out” distances of 10 km and 0 km respectively. The "center location" for each tile stores the center point of the tile (Figure 3-29).
The implementation is designed to replace the polygon based terrain tile representations with BlockMaps, when the 250×250 m terrain tiles are beyond 2 km from the camera rendering point. The "switch in" and "switch out" distances of the terrain tiles were modified to 2 km and 0 m, and the "switch in" and "switch out" distances of 10 km and 2 km were used for the BlockMap representations respectively. Detailed 3D models are also embedded in the terrain. Due to their geometrical complexities, polygon based representations of these models are switched out to be replaced by computational less demanding BlockMaps beyond a distance of 1 km. This distance is shorter than the distance when geometry blocks switched to BlockMap representations for the
250×250 m terrain tiles. The co-existence of polygon based and image based representations for 3D objects within similar viewing range (1 km to 2 km) from the viewer illustrates my approach to adopt BlockMap rendering for geometrically more complex terrain features selectively.

Figure 3-30 illustrates the implementation where geometrical representation is employed for all buildings that are less than 1 km from camera rendering point. Terrain tiles that are beyond 2 km from the rendering point, on the other hand, are represented by BlockMap rendering approach. For terrain features residing at distances of range from 1 km to 2 km, geometrical complex buildings (Boston city hall and Massachusetts State House) are rendered using image based BlockMap approach, while relatively polygonally less demanding buildings are rendered in polygon based geometries. This addresses the asymmetric polygonal distribution characteristic of urban terrain for efficient rendering. It is observed from Figure 3-30 that the geometrically more complex Boston city hall and Massachusetts State House are rendered using BlockMaps (as illustrated in the left screenshots in line and fill modes). They, however, switched to polygon based representation when viewed at a distance of less than 1 km (as showed by the screenshots in the right column). When rendered in fill modes (bottom row), the buildings geometrical details are successfully portrayed and visualized, be it represented in BlockMaps or polygons.
To integrate image-based BlockMap rendering within traditional hierarchical terrain structure seamlessly, the bounding polygons for BlockMap were created as children of LOD nodes with "switch in" and "switch out" values of 10 km and 2 km respectively. The switching values complement with the values used for the logical, hierarchical scene description of the urban terrain. Shader-based functions (implemented using glsl) for the BlockMap ray casting algorithm are associated with the BlockMap bounding polygons. During rendering, ray casting function would be activated during the rasterization of the faces. When loaded collectively on
my implemented shader-enabled OpenSceneGraph application, these traditional polygon based geometry blocks coupled with shader-enabled faces for BlockMaps orchestrate an artifact-free urban scene interactively as shown in Figure 3-31.

Executing on a 1.73GHz Intel i7 platform equipped with 6 GB of RAM and a NVIDIA GeForce GT 435M 2GB graphics card, the performance of the aforementioned approach in representing and rendering the large-scale Boston urban terrain is compared with the rendering rate using fully traditional geometrically representations with 1 and 3 LODs. Amount of occlusions of randomly placed spheres by terrain features at corresponding camera viewing heights is also computed. Different rendering camera points directing towards the center of the terrain with varied viewing heights are used for performance assessments (Figure 3-32).
As shown in Figure 3-33, the results indicate that the performance of the hybrid rendering approach is better than the rendering using polygon based geometry only representation with full details. The performance of the hybrid rendering at low camera viewing heights, however, is worse than that when polygon based terrain with LODs is used. High rendering rates are observed when viewing the urban terrain rendered using hybrid rendering approach when camera points vertically downwards, from high view locations. This is due to the activation of the BlockMap acceleration algorithm respectively when viewing rays are directed vertically downwards. The BlockMap acceleration algorithm implementation details would be described in section 3.4.
The majority of the buildings used in the terrain prototype is devoid of geometrically complex facades. As such, the potential of using BlockMap rendering, to handling extremely complex urban terrain beyond the prototype terrain, is promising. This is because the rendering performance of BlockMap depends on the screen footprint spanned by the BlockMap bounding box rather than the scene complexity.

Figure 3-34 illustrates the amount of occlusions of randomly placed spheres caused by terrain features, using various camera viewing heights as illustrated in Figure 3-32. It could be observed that, in general, amount of occlusions are substantial when camera viewing heights are low. As
camera viewing heights increase, more spheres become visible. The amount of computed occlusions, when the scene is rendered using hybrid representations, matches closely to that when rendered using terrain with full geometric details. The amount of occlusions of the spheres is significantly lesser when the scene is rendered using terrain with three LODs.

![Graph](image.png)

**Figure 3-34. Amount Of Occlusions Of Randomly Placed Spheres**

In view of the high level of spatial accuracy achieved almost matching that when full geometric detail terrain is used while maintain high rendering rate comparable to typical multi-level LODs terrain, rendering using BlockMaps for urban terrain demonstrates attractive rendering approach well suited for use in visual simulation system. With the advancements in GPU technology and increasing demands for correlated simulation environment with complex urban terrains modeled with both high visual and spatial accuracy, the enhanced depth corrected BlockMap
representations, whose computational complexity depend on rendering screen footprints, offer great potential for use in future urban terrain representations.

### 3.4 Accelerated Ray Casting Process For Enhanced BlockMap Rendering

To reduce per-fragment operations for the Enhanced BlockMap rendering, acceleration techniques are integrated into the ray casting operations. The approach involves optimizing ray traversal and utilizing empty-space skipping techniques.

Space leaping is employed to speed up volume rendering, where multiple rays along view of sight are casted and sampled to determine points of intersections. Skipping contiguous sequence of empty cells provides significant improvement in processing and rendering. To achieve speedy ray casting processing, various techniques are adopted to compute the ray leaping distances. Concept used for the space leaping techniques (Donnelly, 2005) for ray casting process in 3D spaces is adopted in the research work. To maximize storage space for BlockMap texture, 2D planar inter-object space map and minimum distance from the surface from BlockMap ceiling are used in place of 3D map where each pixel stores closest surface displacement values in 3D spatial spaces. The subsequent sections detail my approach to achieve significant performance improvement in rendering BlockMap.

To benchmark the effectiveness of the proposed algorithms, the 8×8 km (kilometer) prototype Boston terrain described in section 3.3.3 is divided into grids, each to be encoded in BlockMap. The terrain is rendered entirely using BlockMaps for performance assessment. The commonly used Bresenham's line algorithm is used as a baseline comparison for the performance
assessment. The improvements in rendering rate when wide views of the terrain at various camera viewing heights are recorded, compared and discussed. Rendering rate at different camera viewing heights as described in section 3.3.3 are used in the assessment.

The followings are my main contributing techniques made to accelerate ray casting:

1. Accelerating ray traversal using fix step size and taking slant factor into consideration,
2. Adopting space leaping for steep and oblique viewing rays and
3. Accelerating ray casting using maximum mipmaps data structure

### 3.4.1 Accelerating Ray Traversal

Common ray traversal algorithms such as Bresenham's line algorithm and Digital Differential Analyzer (DDA) attempt to compute grid indices to retrieve terrain data. To optimize the traversal process, the algorithms focus on advancing along the axis along which the directional vector progresses at larger magnitude using unit step. During each iteration, condition checks are done to determine if sampling along the ray need to be incremented along the other axis. In my implementation, a fix step size is used for ray traversal. The height information in BlockMap is encoded in 2D grids. When viewing rays are steep, the rays traverse through relatively fewer cells. As such, larger sampling interval should be adopted by viewing rays that are steeper. This is accomplished by computing a steep factor based on the viewing ray's steepness. The sampling rates along the rays are multiplied by this factor. Unlike the algorithm for Bresenham's line algorithm, using a fix step size while traversing through the ray simplifies computations without the need to check for condition before progressing onto the next cell. (Figure 3-35) Using
nearest-neighbor texture filtering, color information for the relevant texels could be retrieved immediately without the need to identify the indices to respective texels.

Figure 3-35. Determine Traversed Grids Using Bresenham's Line Algorithm (Left) And Using Fix Step Size (Right)

Figure 3-36 illustrates the performance comparison using my approach and Bresenham's line algorithm when performing ray traversal. Significant performance improvement is observed for viewing rays that are steep for both approaches since the sampling rates become larger after applying the slant factor. In addition, it is observed that the rendering performance of the proposed approach is better when compared to that when standard Bresenham's line algorithm is used.
3.4.2 Enhanced Wall Indexing System For Ray Casting

Due to the discrete nature of the sampling method with fixed step size adopted for the ray casting operations, fine structures may not be registered and rendered. The visual effect using such ray casting approach produces incorrect results. Figure 3-37 illustrates the top view of rays that are cast towards a wall structure. While rays indicated in green successfully register the wall structure indicated in gray, those in red fail to do so. The diagram, on the right, illustrates the visual effect where the vertical strips arise where sampling points along respective viewing ray fails to register any cell with wall structure index. Respective roof colors are used for wall rendering in the event when ray does not intersect with any wall indexing cell.
One solution to circumvent aforementioned shortfall of the ray casting process using fixed step size is to reduce the sampling intervals along the rays. Other approach is to consider all grids intersected by the viewing rays. These solutions, nevertheless, introduce computational overhead, which reduce rendering performance.

An enhanced wall indexing system for ray casting is adopted. The system extends the indexed grids into the neighboring cells. This allows discrete sampling points along the rays to intersect with cells containing wall structure indices. Figure 3-38 shows the ray casting process using wall index maps before and after extending valid wall index cells to neighboring units. Extending valid wall index cells to neighboring units ensure their intersections with points along viewing rays sampled at discrete intervals.
Figure 3-38. Ray Casting Process Using Wall Index Maps Before (Left) And After (Right) Extending Wall Index Cells To Neighboring Units

Figure 3-39 illustrates the wall index maps and visual rendering effects before and after extending wall index cells to neighboring units. The red strips, that were present formerly, are eliminated consequently.
3.4.3 Spacing Leaping For Steep Viewing Rays

Figure 3-40 illustrates the side view of the ray casting process when viewing rays are relatively perpendicular to the horizon. The starting point for the ray stepping is first initialized as the intersection point of the ray with the BlockMap bounding box. Using the assumption that there is no terrain height difference between the intersection points of the ray with the BlockMap ceiling and the terrain surface if the viewing ray is directed vertically downwards, the leap distance is set to be the vertical distance, $x$. 
Using $x$ as leap distance, improvement in BlockMap rendering rate could be achieved. The improvement occurs when camera view point is directed at the terrain vertically downwards. Number of samplings along rays of sight decreases tremendously from points spaced evenly along the view rays to only points close to the surfaces. When extending this approach to slightly tilted vertical viewing rays, performance improvement extends to a wider range of viewpoints. However, artifacts result consequently. Closer inspection on the occurrences of visual artifacts is done. The investigation indicates that the initial assumption, that there is no terrain height difference between the points on the actual hit terrain surface and that of the hit point on the terrain surface if the viewing ray is vertically downwards, no longer hold (Figure 3-41).
To remedy, an additional check is performed to determine if the condition holds before adjusting the leap distance. This is achieved by sampling two additional points, namely the predicted hit point and midpoint between the predicted hit point and BlockMap ceiling and determine if the height information for them is consistent. In the case where the condition fails, leap distance is updated using the algorithm for oblique rays. The details to compute leap distances for oblique viewing rays are discussed in the next subsections.

Figure 3-42 illustrates the visual effects after adding this check into the algorithm. It is observed that building walls that are not captured during the ray casting process due to setting inappropriate leap distances are successfully registered after implementing the checks.
3.4.4 Spacing Leaping For Oblique Viewing Rays

Unlike determining space leaping distance for steep viewing rays, optimizing ray casting for oblique rays is more challenging due to a higher incidence of inter object occlusion, thus limiting leaping process to consist of a series of small "hops" rather than a huge "jump" to potential surfaces of intersections.

If the intersection point is on the ceiling of the BlockMap bounding block, the leap distance is determined using minimum inter-object distance and minimum object surface distance from BlockMap ceiling maps (Figure 3-43). In my algorithm, the equations are formulated using the notion that the target intersection point would be larger than the minimum of the distances, a's and b's. They lie along the view ray vector and are computed using minimum inter-object distance and BlockMap height. The resultant distance is then compared with the respective value for minimum surface distance from BlockMap ceiling map. The maximum of the two values is used as the leap distance.
The equations used for the computations are as follow:

\[ m = \text{minimum inter-object distance} \]

\[ d = \sqrt{(\text{ray}.x^2 + \text{ray}.y^2)} \]

\[ a = \frac{m}{d} \]

\[ b = \frac{(p_1.z - \text{ht})}{\text{ray}.z} \]

\[ c = \min(a, b) \]

\[ s = \text{minimum object surface distance from BlockMap ceiling at point } (p.x, p.y) \]

\[ \text{leap distance} = \max(c, s) \]

If the viewing ray is oblique and the intersection point is on the sides of the BlockMap bounding block, similar equations could be used (Figure 3-44).
The space leaping process is also extended for each intermediate point during the ray casting process. Performance improvements made is presented in Figure 3-45. It could be observed that the introduction of leap distance results in performance improvement for selective, low camera viewing heights.
3.4.5 Adopting Maximum Mipmap Structure For BlockMap Ray Casting

A fully subdivided quadtree acceleration data structure for ray-height-field intersection algorithm using the maximum mipmap (Tevs, Ihrke, & Seidel, 2008) is also adopted for BlockMap ray casting for performance assessment. Figure 3-46 illustrates the results obtained. It could be observed that better performance for low camera heights could be observed when maximum mipmap acceleration structure is employed. At high camera heights where viewing rays are steep, traditional line sampling approach using fix step traversal results in better performance. This is so as the line sampling requires significantly fewer steps to reach the terrain hit location when viewing rays are steeper.
The fix step and space leaping techniques are also combined with maximum mipmaps accelerated approach. Figure 3-47 illustrates the performance obtained.
3.5 Ray Tracing For Improved Understanding Of Urban Terrain Using Viewshed Analysis And Rendering

While accurately representing and presenting geometric details of 3D urban terrain potentially enhances situational awareness in complex urban scenarios, of equal and increasing significance are terrain analysis related tasks. Viewshed analysis, which refers to areas with visibility from an observation point or points, is one basic terrain analysis capability. They are used in a wide variety of applications. For example, viewshed visualization can aid in general analysis of an urban area. They assist in activities such as terrorism and crime prevention. It can present vulnerable areas exposed to attackers (VanHorn & Mosurinjohn, 2010). Computational intensity
of such analysis, nevertheless, is often significant, particularly when high resolution representations for urban terrain are used. A high-performance visualization engine, that exploits the massively parallel computation architecture of modern GPUs, is especially well suited to handle rendering and computation of large-scale complicated 3D geospatial environments. This section of the paper delineates the approach to achieve rendering multiple viewshed analysis using GPU computation. Rendering performance are then presented. Related performance improvements for decision makers operating in cognitively demanding scenarios are then evaluated via user experimental studies.

3.5.1 GPU-Accelerated Viewshed Analysis

A hybrid approach involving rasterization and ray tracing is adopted in the prototype implementation. Two copies of terrain geometries data in the form of polygon-based representation and terrain height map are used (Figure 3-48). While rendering using BlockMaps where terrain heights and color information are to be encoded within each and every BlockMaps, only height information is encoded to be used for ray tracing for viewshed analysis.
Figure 3-48. Terrain Height Map (Left) And Polygon-Based Terrain (Right)

Figure 3-49 illustrates the process to render viewshed using rasterization and ray tracing.

Figure 3-49. Rendering Viewshed Using Rasterization And Ray Tracing
3.5.1.1 Initial Visibility Calculations Using Rasterization

The scene is rendered using rasterization technique on polygon-based terrain. Rasterization transforms and projects relevant terrain polygons onto the screen. During the process, z-buffering is used to paint only closest 3D polygon based faces while "culling" technique discards terrain polygons not within the viewing frustum from the rendering camera. Such rasterization technique efficiently performs initial visibility calculations from observer point of view. Using barycentric interpolation for the terrain vertices, spatial information of the terrain surfaces for each and every rendered pixel could be obtained during triangle rasterizations. They are to be used as one of the pair of end points for inter-visibility checks. The other end point would be the point of interest (Figure 3-50).

3.5.1.2 Terrain Viewshed Analysis Using Ray Tracing

To compute and determine visibility from point of interest(s) for each of every terrain surface shown as pixels on screen, ray tracing is performed on terrain height map. Using ray tracing for each and every rendered terrain pixel, flexibility in selecting areas of interests for consideration is allowed. For example, performing inter-visibility checks could be done only for terrain grounds. Such selective processing allows optimization. Unlike inter-visibility computation approach using depth map for shadow mapping, using ray tracing for inter-visibility checks from multiple points of interests require only one rendering pass.

Figure 3-50 illustrates the process to render viewshed on terrain geometries. The dotted lines indicate rasterization of the terrain geometries where 3D polygons are transformed to 2D display. The red and green lines, on the other hand, are vector rays used to determine visibilities from
point of interest. The inter-visibility checks are performed using ray tracing on terrain height map. Lines that are highlighted in red indicate visibilities with point of interest while others, that are highlighted in green, refer to surfaces occluded from point of interest.

![Diagram](image.png)

*Figure 3-50. Rasterization And Ray Tracing For Viewshed Analysis*

### 3.5.2 Performance Evaluation

The GPU accelerated viewshed analysis and rendering is implemented in the pixel shader of modern graphics cards. Visibility information is color-coded and blended with the terrain texture texel colors to indicate their visibility from the point(s) of interest(s). Figure 3-51 illustrates the rendering of multiple viewshed analysis, computed and rendered with respect to several UAVs (indicated using blue icons). The visibility indications are blended with the terrain geometries, to
be visualized in 3D. Viewshed results indicating ground visibility with a selected UAV of interest (highlighted using translucent blue sphere) are rendered with shades of blue. In addition, shades of red and yellow are also added to indicate the amount of combined visibility from the UAVs. The greater the shade of yellow, the higher the amount of combined visibility, when viewed from UAVs. Such visualization could aid in determining area of vulnerability against adversity.

Figure 3-51. GPU-Accelerated Viewshed Analysis
Figure 3-52 illustrates the rendering performance with different number of viewshed analysis. The results are collected based on a prototype application executed on a commodity laptop. It could be observed that the viewshed analysis could be performed at interactive rate, even when several points of interests are considered.

![Graph showing frame rates vs. number of viewsheds]

*Figure 3-52. Frame Rates Of Scene Rendered With Different Number of Viewsheds*

### 3.5.3 Evaluating Effectiveness Of Displays Enhanced With Viewshed Analysis

A research study, that uses both knowledge based and performance based assessment methods, to evaluate the effectiveness of displays enhanced with viewshed analysis in presenting information to operators is conducted. In the study, enhanced interfaces are simulated using the proposed viewshed rendering techniques. To evaluate users’ performance improvement when operating with enhanced interfaces, an experimental study is carried out. Additionally,
knowledge based assessment of the enhanced interfaces is made using survey questionnaires. The survey also collects subjective ratings on the informativeness of the enhanced displays.

3.5.3.1 Performance Based Assessment Of The Enhanced Interface Using Experimental Study

To assess users' performance improvements made when real-time viewshed analysis is used, a comparative user study is carried out. The study involves using simulated scenarios. The scenarios comprise of Human-UAVs team performing ground targets elimination. Control interfaces are used to provide enhanced situational understanding to the operators. The objective of the experimental study is to measure the amount of performance improvements made by operators when using the enhanced displays in terms of speed and accuracy.

Two statistical hypothesis tests are made:

H1\textsubscript{0}: Time taken to determine correct relationship in terms of visibility between a group of weapon loaded UAVs and targets when using enhanced display is not less than that when using baseline display.

H1\textsubscript{1}: Time taken to determine correct relationship in terms of visibility between a group of weapon loaded UAVs and targets when using enhanced display is less than that when using baseline display.

H2\textsubscript{0}: Number of attempts made to determine correct relationship in terms of visibility between a group of weapon loaded UAVs and targets when using enhanced display is not less than that when using baseline display.
H2: Number of attempts made to determine correct relationship in terms of visibility between a group of weapon loaded UAVs and targets when using enhanced display is less than that when using baseline display.

A within-subjects repeated-measures design is planned. Two sets of data are collected from each participant from two different treatments. The treatments comprise of participants taking up the roles of operators to issue commands for the accomplishment of the simulated missions. The first treatment involves providing participants with baseline control interfaces while the second involves using enhanced control interfaces. The scenarios, consisting of forty static ground vehicles positioned at various locations, two surveillance (in white) and five Unmanned Aerial Vehicles (UAVs) (in blue) performing surveillance tasks and target detection and elimination, are crafted (Figure 3-53). The operator is required to activate firing controls on the firing UAVs. The UAV flight routes are fixed. The operators have controls to alter the UAV "look at" target spot and to trigger weapon release.
The ground vehicles consisted of several land-based ground vehicles. Figure 3-54 illustrates the visual appearance of a tank to be used as ground vehicles.
Figure 3-55 illustrates the baseline user interface to be used for the experimental study. The user interface comprises of a situational 2D map displayed on the left, simulated UAV video stream shown bottom right with selected relevant information displayed on the top right. The information include the number of entities alive, attempts made, frame indices and distance between target spot and selected UAV. The UAV icons indicated in blue are the firing UAVs while those indicated in white are surveillance UAVs. 3D mapping of real-time images acquired from the 2 surveillance UAVs would be performed on the 2D map. Doing so allows up-to-date situational pictures to be presented. The fusing techniques described in chapter 4 would be used for the mapping process. Texture portions depicting "not as current" information are displayed with darker shades. The difference in shades allows indication of their potentially obsolete status. Highlighting their status allows operators to be more informed. Such information is useful when determining if vehicles depicted in image portions remain alive at similar spots.
Visual appearances of the situational displays to illustrate the scenarios with and without viewshed analysis would be simulated. Two user interfaces with and without viewshed analysis are to be compared in the experimental studies. Figure 3-56 illustrates the simulated situational display augmented with viewshed analysis. Viewshed results indicating ground visibility with selected UAV are rendered with shades of blue. In addition, shades of red and yellow are also added to indicate the amount of combined visibility from the UAVs. The greater the shade of yellow, the higher the amount of combined visibility from UAVs.
Eight participants are recruited for the experimental studies. During the experiment, participants are tasked to detect and eliminate targets with direct line of sight with the selected UAV as quickly as possible while at the same time making as little mistakes as possible. Participants could select to view from any of the firing UAVs by mouse click control. The selected UAVs would be highlighted using a blue sphere. To view at any targeted spot, participants need to select the relevant spot on the 2D situational display. The simulated UAV video stream, as shown bottom right, would be updated accordingly. To activate firing control from the selected UAV at the targeted spot, participants are required to key press on the space bar on the keyboard.

A ten minute study brief is provided to prepare the eight participants for the experiment. Prior actual simulation runs, trial runs are given to participants to allow them to get familiarize with
the interfaces. Two simulation runs are performed for each participant. To remove scenario familiarity effects, scenarios with different target locations are generated. To remove learning effects, sequences in the simulation runs are different for the participants. At the end of the section, participants are asked to indicate their preferred interfaces for the assigned tasks.

The data would be recorded by the application which simulates the displays. During simulations, the mouse click events are registered. This allows the computations and recordings of users' number of attempts and time taken to identify UAV-target pairs and eliminate targets. Table 3-1 shows the results obtained from the study. Average times taken to eliminate all ground targets are 247 seconds and 298 seconds, while the numbers of attempts are 45 and 49 for enhanced and non-enhanced displays respectively.

<table>
<thead>
<tr>
<th>Participants</th>
<th>Enhanced Display</th>
<th>Non-Enhanced Display</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time taken (in seconds)</td>
<td>Number of attempts</td>
</tr>
<tr>
<td>1</td>
<td>243</td>
<td>41</td>
</tr>
<tr>
<td>2</td>
<td>376</td>
<td>48</td>
</tr>
<tr>
<td>3</td>
<td>242</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>246</td>
<td>42</td>
</tr>
<tr>
<td>5</td>
<td>237</td>
<td>45</td>
</tr>
<tr>
<td>6</td>
<td>256</td>
<td>44</td>
</tr>
<tr>
<td>7</td>
<td>246</td>
<td>43</td>
</tr>
<tr>
<td>8</td>
<td>133</td>
<td>54</td>
</tr>
<tr>
<td>Average</td>
<td>247</td>
<td>45</td>
</tr>
</tbody>
</table>

Using paired two sample t-tests for times taken and attempts made to eliminate all targets, p-values of 0.046 and 0.042 are obtained respectively. The null hypotheses, $H_{10}$ and $H_{20}$, that speed and accuracy to determine correct relationship in terms of visibility between a group of weapon loaded UAVs and targets when using enhanced display is not higher than that when
using baseline display, are rejected at $p = 0.05$, in favor of the alternative hypotheses, $H_1$ and $H_2$. Post-hoc power analysis is also performed using data collected after the experiments. The power (1-$\beta$ error probability), that is, the probability that the effects of speed and accuracy improvement could be detected using the implemented experiment setups are found to be 0.544 and 0.566 respectively. The beta errors are large and outside typical range of 0.2. Figure 3-57 illustrates the results presented in graphic forms. Relatively large variances for time taken to determine correct relationships in terms of visibility between a group of weapon loaded UAVs and targets are observed. This may be attributed to the fact that the untrained participants are equipped with varied level of spatial reasoning skills. Nevertheless, improvement in participants' speed and accuracy is observed for most. Statistical inferences and recommended future studies would be discussed in section 3.6.

All participants deemed the additional of viewshed analysis aid in determining the visibility between the selected UAVs and respective visible ground areas. Some expressed that additional of multiple viewshed display results in cluttering of visual information for untrained users. One
participant pointed out that the scenario settings do not bring out the full potential usefulness of viewshed analysis. He commented that the majority of the randomly placed ground vehicles are rather exposed and have line of sight with the UAVs more often than not. He highlighted that the scene is rather heavily populated with 40 ground vehicles. This has resulted in the users able to fire at targets within proximity that are exposed and not occluded by the buildings. In his opinion, improvement in operator performances using enhanced interfaces augmented with viewshed rendering would be significant should the targets be strategically placed at obscure locations.

3.5.3.2 Knowledge Based Assessment Of The Enhanced Interface Using Survey Questionnaires

A knowledge-based assessment is also made using survey questionnaires. The survey provides additional insights on the informativeness of the enhanced displays. Participants answer questions on scene activities based on information depicted in the simulated perspective sensor views and 2D maps. The first research objective of the survey aims to compare the levels of accuracy in assessing inter-object visibility among different objects in the scene when using baseline and enhanced displays augmented with viewshed analysis. The second research objective aims to evaluate if the enhanced display has allowed participants to achieve perfect scores when assessing inter-object visibility among different objects. In the survey, users’ ratings are also collected to measure subjective informativeness of the displays.

Three statistical hypothesis tests are made:

\[ H_{30}: \text{The participants' mean score in answering the questions related to assessing inter-object visibility among different objects in the scene is not higher when information are} \]
presented using enhanced displays when compared to that when information are presented using baseline displays.

$H_3$: The participants' mean score in answering the questions related to assessing inter-object visibility among different objects in the scene is higher when information are presented using enhanced displays when compared to that when information are presented using baseline displays.

$H_{40}$: Number of participants receiving a score of 1 in answering the questions related to assessing inter-object visibility among different objects in the scene is not higher when information are presented using enhanced displays when compared to that when information are presented using baseline displays.

$H_{41}$: Number of participants receiving a score of 1 in answering the questions related to assessing inter-object visibility among different objects in the scene is higher when information are presented using enhanced displays when compared to that when information are presented using baseline displays.

$H_{50}$: The displays enhanced with viewshed rendering do not allow participants to achieve perfect scores when assessing inter-object visibility among different objects.

$H_{51}$: The displays enhanced with viewshed rendering have allowed participants to achieve perfect scores when assessing inter-object visibility among different objects.
A survey is conducted to gather users' responses. This survey is also used to collect users' responses to evaluate fused display as described in section 4.5.3. In the survey, each participant is required to answer four questions on scene activities.

They include:

Q1: Match the targets on the photos and 2D map (based on information presented using baseline interface)

Q2: Select vehicle(s) that have direct line of sight with the explosion (based on information presented using baseline interface)

Q3: Match the targets on the photos and 2D map (based on information presented using enhanced interface)

Q4: Select vehicle(s) that have direct line of sight with the explosion (based on information presented using enhanced interface)

Answers collected for Q2 and Q4 attempt to give insights to benefits of viewshed rendering while answers collected for Q1 and Q3 attempt to give indications of the effectiveness of enhanced display using fusing strategy described in chapter 4. The answers from Q2 and Q4 are of interest in this section of the dissertation. Each question is based on two simulated aerial photos and a corresponding 2D map depicting an urban scene. Three ground vehicles are placed in the urban scene. The vehicles are captured by the aerial photos, and their locations are indicated on the 2D map.
Two scenarios, A and B, are crafted. Questions on scene activities related to the two scenarios are asked. Figure 3-58 illustrates imagery information for the questions. Four diagrams: A1, A2, B1 and B2 are generated. The lettering labels are used to identify the scenario to be used. The numbering labels refer to the display types (baseline or enhanced) to be presented. The enhanced displays have the 2D situational display maps updated with superimposed photos and enhanced with viewshed rendering. The superimposed photos are mapped based on the proposed fusing strategy described in Chapter 4. Those who successfully select the right combination of vehicle(s) with direct visibility of the site of interest are awarded with a score of 1 for Q2 and Q4, a zero score otherwise.
106 college students, who are above 18 years of age, participated in the survey. The sample comprises of an equal mix of both genders. Half of the participants are presented with diagrams A1 followed by B2 while the other half are given diagrams B1 and A2. No pre-test trainings are provided. However, detail instructions and sample questions with correct answers are provided for references. Two different scenarios are presented to each participant to remove scenario
familiarity effects. Additionally, half of the participants are presented with different scenarios for similar kinds of interfaces.

The mean scores obtained from the answers on questions related to the information presented using baseline and enhanced displays are 0.632 and 0.726 respectively. A p-value of 0.071 is obtained from two sample t-Test for the mean scores. As such, null hypothesis, H3, that the participants’ mean score in answering the questions related to assessing inter-object visibility among different objects in the scene is not higher when information are presented using enhanced displays when compared to that when information are presented using baseline displays, cannot be rejected at $p = 0.05$. Post-hoc power analysis is performed using data collected after the experiments. The power (1-$\beta$ error probability), that is, the probability that the effects of accuracy improvement could be detected is found to be 0.430. The beta error is large and outside typical range of 0.2.

Additionally, an alternative test using chi square is made. Table 3-2 illustrates the numbers of participants who obtained scores of 1s and 0s when information are presented using baseline and enhanced displays.

<table>
<thead>
<tr>
<th></th>
<th>Number of Participants Who Score 1s</th>
<th>Number of Participants Who Score 0s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline Display</td>
<td>67</td>
<td>39</td>
</tr>
<tr>
<td>Enhanced Display</td>
<td>77</td>
<td>29</td>
</tr>
</tbody>
</table>

$\chi^2$ value of 2.165 and $p$ value of 0.141 is obtained. As such, using chi square test, null hypothesis, $H4_0$, that number of participants receiving a score of 1 in answering the questions
related to assessing inter-object visibility among different objects in the scene is not higher when
information are presented using enhanced displays when compared to that when information are
presented using baseline displays, cannot be rejected at $p = 0.05$.

To answer the second research question if the enhanced displays with viewshed rendering have
allowed participants to achieve perfect scores when assessing inter-object visibility among
different objects, a one sample t-test is performed to test if achieved score is equal to 1. $P$ value
of less than 0.001 is obtained. The power (1-$\beta$ error probability), that is, the probability that the
effects of accuracy improvement could be detected are found to be larger than 0.999. The null
hypothesis, $H_{50}$, that the displays enhanced with viewshed rendering do not allow participants to
achieve perfect scores when assessing inter-object visibility among different objects, cannot be
rejected. Statistical inferences and recommended future studies are discussed in section 3.6.

Figure 3-59 illustrates the participants' rating on the informativeness of the enhanced display.
The survey results indicate that the majority of the participants rate the proposed display to be
informative in most settings.
3.6 Summary

Urban terrains are rich in visual and functional complexity as a result of developments due to various cultural factors. As such, they are difficult to model. Likewise in terrain analysis, computational complexity involved in terrain-related calculations such as sensor coverage analysis involving multiple line of sight checks grows in tandem with the resolution and richness of source data. Modern Graphics Processing Units (GPUs), which can be applied to a wide range of applications beyond processing millions of polygons, could be leveraged to perform purposeful beyond realistic rendering tasks. A high-performance visualization engine, that exploits the massively parallel computation architecture of modern GPUs, is especially well
suited to handle rendering and computation of large-scale complicated 3D geospatial environments. This chapter presents approaches that leverage on GPU ray tracing, a rendering approach, conventionally not employed in the simulation community in favor of rasterization, to depict complex, large scale terrain and visualize viewshed terrain effects at interactive rate.

The former section of this chapter focuses on my research works related to BlockMap. BlockMap, an image based rendering approach, is a GPU-friendly simplified representation of urban environments that could be used to encode coarse representations of distant city blocks. BlockMap rendering requires minimal polygon based data while preserving the ability to portray building heights and wall textures. The approach exploits the advantages offered by traditional polygon based representation for close view objects, and image based rendering approaches for far away objects in rendering large scale urban terrain. In my research, enhancements are made to classic BlockMap rendering and encoding techniques to address shortfalls of the standard BlockMap. The enhancement details are elaborated in section 3.1. The improvements achieve visualization of urban terrain with a high level of spatial and imagery accuracy well suited for use in visual simulation system. One key achievement includes representing and visualizing simulation entities and BlockMaps together in common spatial space. This is achieved by computing and updating depth information for each and every BlockMap surface fragment. Section 3.2 delineates another improvement made which involves adopting a more flexible utilization of BlockMap by representing individual, complex buildings and assigning different LOD switch values for different buildings to optimize computational resources and improve rendering performance. In section 3.3, implementation details are described and demonstrated using a prototyped large scale Boston urban terrain covering an area of 10×10 km. Lastly,
accelerated ray casting techniques are adopted for BlockMap rendering to achieve high rendering rate. The acceleration techniques are described in section 3.4. In view of the advancements in GPU technology and increasing demands for correlated simulation environment with complex urban terrains modeled with both high visual and spatial accuracy, BlockMap representations, whose computational complexity depend on rendering screen footprints, offer great potential for use in future urban terrain representations.

The latter half, section 3.5, of this chapter focuses on adopting ray tracing to render multi-view real-time viewshed analysis. Ray tracing is a technique widely used for realistic rendering, such as shadowing and reflection effects. Analogous to rendering shadowing effects, similar technique is employed to render multi-view real-time viewshed analysis. Such potentially helpful real-time viewshed analysis is lacking in most current decision support applications. A research study, as described in 3.5.3, that uses both knowledge based and performance based assessment methods, to evaluate the effectiveness of rendering viewshed in presenting information to operators is conducted. Results obtained from the performance based assessment indicate that speed and accuracy in determining inter-visibility relationships between the team of UAVs and ground targets are higher when enhanced displays are used at 95% confidence interval. Power analysis, however, indicates that beta errors are large. This corroborates the need for further studies to be conducted to validate the effects. In addition, results obtained from the knowledge based assessment indicate that the participants answered the questions related to assessing inter-object visibility among different objects in the scene more correctly when enhanced display is used. The differences, however, are not statistically significant at 95% confidence interval. Additionally, the enhanced display fails to allow participants to attain
perfect scores when attempting the questions. Nevertheless, results from subjective ratings from
participants indicate that majority rate the enhanced display to be informative in most settings.
Further studies are to be conducted to evaluate and validate the effects. Future research studies
may adopt larger sampler sizes or better qualified participants to reduce sample variances. To
enhance credibility of the study results, subject matter experts' inputs may be sought. A detailed
discussion on limitations of the current studies that were conducted and recommended future
works are given in section 5.2.4.
CHAPTER 4  FUSING 2D GEO-REFERENCED IMAGES AND 3D TERRAIN

According to Endsley, situational awareness refers to "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" (Endsley, 1995). It is “a human experience involving knowing and understanding what is happening around you, predicting how it will change with time, and being unified with the dynamics of your environment” (ESRI, 2008). Recent increases in the volumes of geo-spatial data such as satellite imageries, elevation maps, street-level photographs and real-time imageries from remote sensory devices contribute to the abundance of useful information from diverse sources. With today's ubiquitous sensing technologies, the challenge is to integrate, correlate and fuse data from disparate sources. "Live" imagery information gathered from networks of geo-sensors, when presented jointly with static information such as 3D terrain models or 2D maps, could generate comprehensive situational pictures for commanders and decision makers.

This chapter details my proposed strategy to fuse 2D geo-referenced images and 3D terrain. The fusing solution achieves real-time performance and makes possible insightful visualization of the situational settings virtually in 3D space. The chapter begins with delineating the benefits of performing rapid and autonomous fusion of images and 3D terrain. It proceeds to state various subtasks to be performed. The proposed approach is then delineated. The fusing strategy is accomplished by harnessing graphics rendering techniques and exploiting GPU programmability. The proposed approach utilizes an integrated environment that considers inter-object occlusion and pre-constructed polygon based geometries for texture mapping and merging. It assumes that
acquired images are coupled with corresponding camera parameters. Such an assumption is coherent with the advent of ubiquitous GPS-enabled devices and related technologies. (Enhancing GPS Accuracy, 2010) (Han, 2008).

The followings list my major contributions to fuse 2D geo-referenced images and 3D terrain:

1. Achieving terrain texture updates using 2D geo-referenced images at interactive rate to facilitate rapid terrain generation and to provide enhanced situational picture. Current state-of-the-art solutions are divided into two categories: presenting and superimposing current images on 3D terrain at real-time, or fusing and reconstructing textured 3D geometries offline at a non-interactive rate. Assuming 2D images and 3D scene geometries are available, my solution achieves fusion of geo-referenced images onto 3D terrain at interactive rate. The solution provides real-time visualization of urban terrain rendered with latest situational pictures using most current surveillance images. Such visualization technique aids in presenting enhanced situational pictures to decision makers.

2. Fusing numerous geo-referenced images using GPU accelerated technique at interactive rate. To augment the techniques to achieve contribution 1, a GPU accelerated technique is adopted to speed up terrain texture updating process. The approach demonstrates potential to achieve multiple fold performance improvement when numerous geo-referenced images are to be considered.
4.1 Geo-Spatial Information

Generating spatial contextual awareness involves synthesis of various information. These information typically involves data related to the spatial environment, a cognitive agent, and a cartographic map (Freksa, Klippel, & Winter, 2005). The spatial environment is the physical space where the activities are to be performed. The cognitive agent is the person or entity involved with decision making while the map represent a given state of the environment. The process of using technology to deliver a human experience is referred to as contextual computing (ESRI, 2008). Context refers to "the general situation that relates to it and which helps it to be understood." (Collins COBUILD English Language Dictionary, 1995) Contextual computing aims to improve users' situational awareness and understanding by integrating and presenting relevant information through interfaces in insightful manner (Hu, 2006).

Context awareness systems aim to link and relate dynamics events in the environment with computer systems. The systems integrate the contributions of static, dynamic, and internal context (Li, 2007). Internal context refers to user information such as task assigned, location in the spatial environment and viewing coverage. Static and dynamic contexts are external contributions from other participating entities.

4.1.1 Static Context

Static context refers to digital terrain-related information depicting areas of interests. 2D maps are one commonly used static context. They are traditionally used to support situational awareness. A map visually represents and models a given state of the environment, such as
roads and locations of events. It assists users in reasoning with the given state of the environment using existing mental models (Livnat, Agutter, Moon, & Foresti, 2005) (Scaife & Rogers, 1996).

Another static context commonly used for situational assessment is 3D terrain representations. 2D maps, lacking in depth information, may be inadequate to provide effective representations for some activities. Efficient urban terrain representation and data exploitation in 3D world are required to comprehend the built-up environment. In the course of the last decade, techniques in 3D data collection for 3D city modeling have improved significantly. Recent technological advances contribute to the availability of highly detailed urban models for visualization and other terrain-related analysis tasks (Lin, Jing, & Zhang, 2008) (Kelly & McCabe, 2006). Approaches to perform data collection include aerial and close range photogrammetry, airborne or ground-based laser scanning, mobile mapping and GPS surveying. Historically, 3D city modeling uses input data collected manually on Digital Photogrammetric Workstations (DPW) or analytical stereo plotters. Much research has been conducted toward the automation of 3D object reconstruction. They include 3D building extraction from stereo aerial image pairs and LIDAR points cloud (Doytsher, 2010).

Advancements in building extraction techniques have also speeded up and increased accuracy for building reconstruction. (Huang, Kwoh, Yuan, & Tan, 2006), for example, achieve 60 building extractions per hour using a single IKONOS image coupled with Rational Polynomial Coefficient (RPC) camera model and building shadow information (Grodecki, 2001). Satellite imageries, however, provide texture information for building tops, but provide limited
information for building facades due to sensor look angles. Performing comparatively well in
texture-less areas and on scenes containing repetitive structures, recent LIDAR has become an
accurate, cost-effective alternative technology to collect terrain data. The applications of both
ground laser and airborne LIDAR scanning technology have demonstrated to be fast and
accurate methods to extract building height information. Given that remote sensing data typically
has no texture information for building facades, oblique images capturing of building facades are
needed for generating photo-realistic geo-specific 3D models using texture mapping (Tan, 2009).

Another static context involves geo-referenced photographs and images. Hitherto, acquiring 3D
geospatial data requires highly sophisticated and expensive analog devices. Datasets employed in
terrain generation used to be confined to geo-rectified orthogonal satellite depicting top view of
the environment. Today's digital age has brought about high resolution imageries made available
with amateur digital camera devices. When attached and used with GPS, which has become
ubiquitous in handheld devices and commercial navigating tools, photogrammetry could be
performed. Unmanned Aerial Vehicle (UAV) based remote sensing also contributes to 3D data
acquisition.

4.1.2 Dynamic Context

Dynamic context refers to "live" information on the current user’s environment acquired by
sensors. An example of such information is video imageries. Video surveillance has grown by
leaps and bounds with decreasing price and better capability of cameras (Sizemore, 2008). This
context-rich images and videos offer significant potential to enhance social control and time
critical tasks such as monitoring criminal activities, threats, tactical military situations, disaster
responses, and search and rescue works (Coey & Montgomery, 2002). The imageries acquired by these surveillance sensors, beaming in real time, though informative, often yield “soda straw” views of the situation due to the cameras' narrow field of views. Operators monitoring large-scale video surveillance systems often face challenges forming complete situational pictures from isolated views of the settings. The effect is aggravated in urban environments due to scene complexity and inter-object occlusions. Better insights from these images could be gained when they are presented jointly using static information such as 2D map as backdrops.

4.2 Fusing 2D Images And 3D Terrain Geometries

To achieve geo-specific terrain generation and real-time texture updates inclusive with textured building facades, registering and aligning oblique images onto vertical planes for building facades need to be done. This necessitates an efficient and streamlined fusion process to apply images captured at different views to building and terrain geometries. The process involves rectification and, if required, merging of photographs taken from different views into single composite texture mosaics to be applied onto building facades. If performed at interactive rate, such real-time 3D texture mapping allows synthetic terrain to serve as a platform to integrate real-time surveillance related geo-referenced imageries. Draping these imageries geospatially over synthetic 3D terrains puts them into context.

Figure 4-1 illustrates my proposed approach to accomplish fusion of 2D geo-referenced images and 3D geometries. Most of the subtasks leverage on graphics rendering techniques, and exploiting GPU programmability. The assumptions that scene geometries and camera pose information are available are made. The idea anchors on generating corresponding scene spatial-
related information for each and every pixel shown on the acquired images. This is done by performing rasterization on a virtual scene using available scene geometries and camera pose information. Analogous to the graphic rendering process where color information of the first visible surfaces are rendered, corresponding scene spatial-related information is rendered instead. The process generates address encoded images. By relating pairs of geo-referenced acquired and address encoded images, 3D geometrical faces are updated. A centralized texture UV mapping system is used both for rendering and mapping process. The mapping system involves initializing a blank texture atlas to be updated using geo-referenced images. The details for the subtasks are elaborated in the following subsections. In these sections, descriptions using the wordings, "buildings" and "terrains", refer to synthetic 3D polygon based geometries, with the former referring to selective small sections of urban scenes. Similar solution is applicable to fuse 2D images with individual buildings and terrains.
4.2.1 Generating A Centralized Texturing System For 3D Geometries

Due to advances in sensing and reconstruction techniques, geometrical terrain constructs are made readily available. Current state of the art techniques to represent terrain geometries involve using polygon based models. To improve detail visual appearances of such memory compact representations, textures are applied to the surfaces. Each and every vertex in a polygon is assigned a texture coordinate to define and relate relevant texture portions to geometrical faces. This is achieved by explicit assignment or procedural definition of texture UV values for the terrain geometries.
In my proposed solution, texture UV values are also used as indices to relate texels from captured images to polygon surfaces. The letters "U" and "V" used in "texture UV" are used to describe the coordinates of texels on the textures during texture mapping. "U" and "V" refer to respective texel horizontal and vertical locations on textures. The texture UV coordinate system is used for texture planar space. The system is used to differentiate from the XYZ coordinate system, commonly used for 3D object space to define object coordinates. First introduced by Catmull in 1974 (Catmull, 1974), texture mapping refers to the application of texture onto a face or polygon, painting details onto the surface. This is achieved by mapping selected texels using values ranging from 0 to 1 along u and v axes (Figure 4-2). For each vertex defined for a polygon, a texture UV coordinate is assigned. In Figure 4-2, the texture UV coordinates (indicated in red) are defined for the polygon's vertices to define relevant section of texture to be applied.

![Figure 4-2. Texture mapping using UV coordinates.](image)

The proposed solution uses a centralized texturing system for 3D geometries for texture updating and rendering. The use of texture UV as indices to relate texels from captured images to polygon surfaces circumvents the need to encode spatial information. This eliminates the requirement to
allocate additional indexing and storage system. To generate a texture UV indexing system for terrain geometries, unique texture UV values must be assigned to each and every portion on terrain polygons. Unlike 2D maps where relevant textures are mapped conveniently onto relatively flat planes, 3D terrains require various textures of different sizes to be mapped onto respective geometries. To ensure even and constant texel resolution for all faces within the terrain, a number of different sized texture spaces are allocated. A texture atlas, containing sub-images, each to be mapped onto different parts of the synthetic terrain, is employed to ensure assignments of unique texture UV values to respective terrain faces. Moreover, using a texture atlas, in place of individual images, promotes efficiency. This is because a texture atlas, being handled as a single unit by the graphics hardware, results in less rendering state changes by binding once as they are drawn (Improve batching using texture atlases, 2004).

4.2.1.1 Generating Texture Atlas Using The Guillotine Split Placement Algorithm

An automated and optimized texture allocation system is adopted using the guillotine split placement algorithm (Jylänki, 2012). Unique texture UV values are assigned to the terrain geometries using a two pass geometries texture UV updating operation. In the first pass, each and every polygon of the terrain model is examined to determine the texture portion required to be allocated on the texture atlas. This is done by computing the minimal area spanned by the polygon. The minimal area comprises of a rectangular space. The horizontal axis for the rectangular space is aligned to be parallel to the longest edge in the polygon. The length and width of the rectangular space are then subsequently obtained. Texture space to be allocated for the polygon is related to the computed smallest rectangular space it spans. This ensures even and
constant texel resolution for all faces within the terrain. While running through the terrain polygons, unique polygon indices are assigned to them. The indices, to be assigned correspondingly to the computed rectangular spaces, serve as links back to the polygons. They are used in the second pass of the geometries texture UV updating operation.

The second pass comprises of texture atlas creation using the guillotine split placement algorithm, and geometries texture UV value updates for each and every polygon. A list of sub-images, with spaces defined by the computed rectangular spaces, is initialized. Using the guillotine split placement algorithm, the sub-images are first sorted according to size and the largest is placed earliest. Smaller sub-images are inserted subsequently. Sub-image placement process involves putting each image to a corner of a free rectangle space, and subsequently splitting the remaining free space into two disjoint free rectangles. Remaining sub-images are inserted into the free rectangular spaces iteratively (Figure 4-3). The constructed texture atlas is then resized to an appropriate size, typically to the power of two, as required by the rendering system. Respective texel coordinates defining the area spanned by each sub-image are then computed. The final step involves running through each and every terrain polygon and updating its texture UV values according to its respective sub-texture space on the texture atlas.
Figure 4-3. Creating Terrain Texture Atlas Using The Guillotine Split Placement Algorithm

Figure 4-4 illustrates the result for sub-images allocation using the guillotine split placement algorithm for a 3D model. For illustration purpose, each polygon is color-coded differently to indicate its respectively location on the texture atlas.

Figure 4-4. Sub-textures Allocation Using Guillotine Split Placement Algorithm
4.2.2 Performing 2D to 3D Transformation From Image To Scene Spaces

Image feature extraction and matching are often used to derive camera pose information. Such processes are usually computationally intensive and result in poor updating performance. Driven by advances in digital imaging and direct-geo-referencing technologies, coarse transform model estimation between 2D acquired image and 3D scene model spaces could be derived directly (Enhancing GPS Accuracy, 2010). This makes possible the availability of approximate intrinsic and extrinsic parameters belonging to a mobile camera at real-time. As such, transforming spatial information from 3D geometries space to 2D acquired image space to aid in texture mapping becomes possible without prior image processing. (Owens, Sycara, & Scerri, 2009), for example, adopt an inverse transform approach to read each and every pixel in the destination image and perform an inverse transformation back to the source to sample colors.

My proposed approach leverages on computer graphics rendering techniques to render a virtual scene using known camera parameters and given 3D terrain geometries. In computer graphics, the current state of the art method of rasterization-based rendering involves the process of transforming 3D scene geometries in spatial space, typically defined by collections of polygons, to 2D camera perspective plane for display. Transformed triangles are then painted with details from textures. This is accomplished by the standard function hardware within the graphics pipeline. Such special-purpose system allows for high efficiency. During rasterization, to determine which surfaces are visible from a camera, z-buffering, a technique commonly used in computer graphics, is exploited. The z-buffer manages image depth coordinates and determines which polygon based surfaces are closest to camera observer. Equipped with the ability to be programmable, GPUs make possible harnessing of graphics power for tasks other than graphics
rendering. In my proposed approach, analogous to graphics rendering process where color information of the first visible surface is rendered for each fragment, spatial-related information of corresponding terrain portion is written onto the fragment instead. The process generates corresponding spatial information for each and every texel on the captured images. Using these address encoded images, color information from captured imageries can be related and mapped to respective locations on the texture atlas for updating.

4.2.3 Relating Information From 2D display to Relevant Geometries

Figure 4-5 illustrates a captured image and corresponding address encoded image rendered using available camera parameters of a geometric model. The address encoded image has texel values encoded with related-spatial information. The rendered image contains related spatial values to be used as indices to the image color information for texture updating. For each and every texel from the pair of images from Figure 4-5, the respective texel color obtained from the captured image is written onto the address stated by the corresponding spatial information values.
The proposed solution uses 3D geometries with textured faces updated with texture UV values referring to a texture atlas created using methods as described in section 4.2.1. Using the camera parameters of captured images, virtual scenes of the 3D geometries are rendered. The process generates address encoded images. This is accomplished by writing the texture UV information of the visible faces on screen and saving onto imageries. The texture UV values range from 0 to 1. They are scaled up to the range of 1 to 4096 before writing onto imageries. The 24-bit RGB format is used for storage of texture U and V information, each taking up 12-bit storage. Table 4-1 illustrates the allocation of data bits used for each texel to store texture U and V information.
Table 4-1. Storage Bits For U And V Information

<table>
<thead>
<tr>
<th>Red channel bits</th>
<th>Green channel bits</th>
<th>Blue channel bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 7 8</td>
<td>1 2 3 4 5 6 7 8</td>
<td>1 2 3 4 5 6 7 8</td>
</tr>
</tbody>
</table>

Storage for U information  Storage for V information

For each and every texel from pairs of images, obtained from screen captures of the urban scene and corresponding address encoded images containing texture UV information, the respective texel color obtained from screen captures of the urban scene is to be written onto the address stated by the corresponding texture UV information values. The color information, retrieved by reading the texel values obtained from address encoded images, are used to compute the texture atlas offset addresses. The texture atlas offset addresses are computed using the following equations:

\[
U = uv_1 + \text{mod}(uv_2, 16)
\]

\[
V = uv_3 + \text{floor}\left(\frac{uv_2}{16}\right)
\]

A summary of the algorithm to update the texture atlas is as follows:

1. Initialize texture atlas
2. For every captured image, create corresponding address encoded texture UV rendered image using given camera parameters
3. For each and every pixel from pairs of acquired and corresponding address encoded texture UV-rendered images, perform the followings:
i. **Color** = get pixel value from acquired image

ii. **Position** = get pixel value from address encoded texture UV-rendered image

iii. Update pixel value at location, **Position**, to **Color** on texture atlas

### 4.2.4 Mapping 2D Image Portions To Respective 3D Geometries

In the proposed approach, mapping of 2D image portions onto terrain involves updating relevant portion of the terrain texture atlas that comprises of sub-images mapped onto respective terrain geometries. To perform texture updating, a GPU–accelerated technique that harnesses the performance and programmability of current hardware is adopted. The approach is adopted from a technique originally designed to construct BlockMap texture using a pair of color and wall address encoded input textures (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009). In my solution, in place of wall address encoded texture, an UV rendered image is used for texture construction. Figure 4-6 illustrates the process to update the terrain texture using color and spatial-related information read from acquired and texture UV render images. Using one graphic rendering pass, an array of point primitives are drawn. While performing vertex transformation, each of these points is displaced according to its spatial information that is read from the address encoded texture UV rendered images. At the same time, its color is updated to the values read from respective texel of acquired image. The number of point primitives corresponds to the number of texels used for the input images. During rasterization, the displaced and color-updated point primitives is rendered to form an image using an orthogonal viewport.
The approach is also extended to fuse multiple pairs of color and spatial-related input textures. Figure 4-7 illustrates the process where three acquired and their corresponding address encoded images, each of resolution 256×256, are transformed and combined to form a composite situational picture using one rendering pass. During rendering, 256×256×3 point primitives are drawn. Three layers of 256×256 point primitives are used. This corresponds to three pairs of
input textures, each using 256×256 texels in resolution. During rasterization, the displaced and color-updated point primitives are rendered to form a composite image using an orthogonal viewport.

Figure 4-7. A GPU-Accelerated Technique For Spatial Mapping Of Multiple Images
4.3 Prototype Implementations

The proposed system to fuse 2D images and 3D geometries is demonstrated and evaluated using implemented prototypes in simulated environments. Versatility of the solution is demonstrated in various settings.

4.3.1 Texture Mapping of Building Facades Using Geo-Referenced Images

To illustrate the feasibility of performing semi-autonomous texture mapping of building facades, 20 synthetic screen captures of the University of Central Florida (UCF) library building model captured from various views are used. These images are generated using a high resolution 3D model. Figure 4-8 illustrates the high resolution 3D model used. This 3D model, with detail geometrical structures modeled, is utilized to generate realistic screen captures of the scene. A different 3D model, with low polygon count that is characteristic for real time simulation usage, illustrated in Figure 4-10 for texture mapping, is selected. A different 3D model is selected for texture mapping with intent to illustrate the robustness of the mapping algorithm against slight geometrical differences between actual 3D model captured by geo-reference images and given 3D model to be used as "base-skeleton" for mapping.
Figure 4-8. High Resolution 3D UCF Model

Figure 4-9 illustrates four of the synthetic screen captures. Some of them capture partial visible sides of the building facades while others have parts of the building occluded. These captured images, each showing exclusive portions of the building facades, are to be input into the proposed system. They are to be extracted, rectified and mapped onto given building geometry.

Figure 4-9. Screen Captures Of The UCF 3D Model

A simplified representation of UCF 3D building model, with low polygon count that is characteristic for real time simulation usage, is constructed and input into the proposed system.
(Figure 4-10). An initial blank texture atlas with a resolution of 2048×2048 is created using guillotine split placement algorithm and mapped onto respective polygons.

Using the camera parameters of each corresponding capture images, a virtual scene with the building geometries planted is rendered. The texture UV information of the visible faces is rendered on screen and saved onto imageries. The texture UV values range from 0 to 1. They are scaled up to the range of 1 to 4096 before writing onto imageries. The 24-bit RGB format is used for storage of U and V information, each taking up 12-bit storage (Figure 4-11).

(Figure 4-10. Corresponding Texture Atlas (Left) And Given 3D Model To Be Used As "Base-Skeleton" (Right))

(Figure 4-11. Texture U And V Information For Respective Screen Captures Of The 3D Building Model)
For each and every texel from pairs of images, obtained from screen captures of buildings and corresponding texture UV information imageries rendered using similar camera settings, the respective texel color obtained from screen capture of the buildings are written onto the address stated by the corresponding texture UV information values from the imageries. The color information, retrieved by reading the texel value obtained from Figure 4-11, is used to compute the texture UV address. The respective texel color obtained from Figure 4-9 is then written onto computed texture UV address.

These displaced color texels are combined to form the final image to be mapped onto the building geometry. In the case where an address has more than one corresponding color information (occurs when a patch is captured by various views), the contribution of all colors are averaged, and the results are written onto the address locations. Figure 4-12 illustrates the updated textures and resultant 3D textured model.

Figure 4-12. Updated Building Texture And Resultant 3D Textured Model
Due to discrete sampling rate of the rendered texture UV image, portions of the scene may have unfilled terrain texture texels. This is due to insufficiently sampling on the terrain parts. A section of the terrain texture with unfilled texels is as illustrated in Figure 4-13.

![Figure 4-13. Terrain Texture Section With Unfilled Texels](image)

To remedy, a solution involves generating pyramid of different level of varied sized and resolution terrain textures and consequently combining them. Five different resolutions of terrain textures are created. They include resolutions of 128×128, 256×256, 512×512, 1024×1024 and 2048×2048 (Figure 4-14). When combining the color information of the texels, those from higher resolution textures are given more weights. Figure 4-14 illustrates selected section of intermediate results, obtained by combining textures of lower resolutions. It could be observed that the final 2048×2048 texture section, unlike the image section as shown in Figure 4-13, is free of artifacts with unfilled texels.
Figure 4-14. Pyramid Of Textures

Figure 4-15 illustrates the resultant texture atlas after combining the pyramid of textures with different resolutions.
The mapping approach could also be used to generate texture atlas for lower LOD versions of the 3D model. Figure 4-16 illustrates a textured, low-poly simplified model. The model is well suited for use as a far distant representation. It uses 29 polygons in place of 71 as illustrated in Figure 4-10. The texture atlas is constructed using the proposed mapping approach. The construction uses 20 pairs of screenshots and corresponding generated address encoded images of the high poly version of the model.
4.3.2 Real-Time Fusion Of Surveillance Imageries In Urban Scenes

The proposed solution is also implemented for real-time fusion of surveillance imageries for urban scenes. A heavily populated urban terrain (Figure 4-17), modeled with detail features such as the road network and 3D trees, spanning over an area of 1300m×1100m of a fictitious downtown urban area, is utilized for the prototype implementation. Real-time performances of the various fusing processes are also discussed.
4.3.2.1 Fusing Acquired Imageries From Static Sensors

The proposed system to map real time imageries acquired from static surveillance sensors onto synthetic terrain is evaluated, using an implemented prototype in a simulated environment. Figure 4-18 illustrates the workflow of the proposed system to fuse video imageries acquired from static sensors. The inputs into the system comprise of 3D terrain geometries, acquired images and corresponding camera pose information. Items residing in dotted lines are tasks to be performed prior real time fusion and visualization. They involve constructing blank terrain texture atlas, generating address encoded texture UV rendered images and constructing an indexing texture to relate video pixels to terrain texture atlas. During real time fusion and visualization, the indexing texture would be used to assess relevant video pixels to be painted onto respective terrain geometries as indicated in the box outline in orange on the right. Leveraging on computer graphics rendering techniques, visualization of the updated terrain fused
with real time video imageries is achieved. This is accomplished using an indexing system which relates terrain geometries to video imageries at texel level.

A GPU shader-based OpenSceneGraph rendering application executing on a commodity graphics platform is implemented. A fragment shader to paint terrain geometries with respective video pixel colors is used. Executing on a 1.73GHz Intel i7 platform equipped with 6 GB of RAM and a NVIDIA GeForce GT 435M 2GB graphics card, the application loads a 1024×1024 texture that stores indexing information and inputs and fuses three 256×256 video imageries iteratively at interactive update rate of 60 Hz.

Figure 4-18. Fusing Acquired Imageries From Static Sensors
4.3.2.2  Fusing Acquired Imageries From Mobile Sensors

Unlike fusing imageries acquired from static cameras, indexing information to relate video pixels to terrain texture varies when camera viewpoints change. As such, the UV textures to relate video pixels to terrain texture need to be updated for each and every video frame. Pairs of video images and respectively generated UV textures are then used to update the terrain texture. Figure 4-19 illustrates the workflow to fuse video imageries acquired from mobile sensors. Items residing in boxes indicated in dotted and bold lines are tasks to be processed prior and during real time fusion and visualization respectively.

![Figure 4-19. Fusing Acquired Imageries From Mobile Sensors](image)

The performance of the aforementioned approach in fusing and visualizing urban terrain is evaluated in a simulated environment. Two applications are developed to evaluate the set up.
The first application generates up-to-date terrain texture while the second provides visualization of the urban terrain to operators to aid in situational assessment. In the simulation set-up, the first application models and updates positions of several UAVs and generates screenshots of acquired images and their corresponding texture UV images. Using pairs of images, the terrain texture is updated. The second application renders the updated textured terrain for visualization. Access right to terrain texture for reading and writing is synchronized using a semaphore to be read from shared memory. Three acquired images with resolution of 256×256 are simulated. The texels from these images are iterated to update a 1024×1024 texture with subparts assigned to the terrain geometries using the guillotine split placement algorithm. The two applications are executed on a single commodity machine for performance evaluation. Running on a 1.73GHz Intel i7 platform equipped with 6 GB of RAM and a NVIDIA GeForce GT 435M 2GB graphics card, an update rate of 8-10 Hz is achieved.

Using the aforementioned approach, the time taken to update urban terrain textures is relatively invariant to the geometrical complexity and amount of occlusions. Processing time depends on various subtasks. They include handshaking between the two applications, iterating through each and every pixel from the acquired and generated texture UV images, and consequently updating the terrain texture.

4.3.2.3 Fusing Spatial Information Using Only Critical Data

The aforementioned approach involves relating, at the pixel level, rasterized terrain geometries using provided camera parameters and color information of the acquired images. To perform texture mapping onto terrain geometries, the generated texture UV indexing system demands
accurate building geometries and camera pose information for image rasterization. The approach is susceptible to erroneous texture mapping if given information differs from ground truth. Should sensing devices or given terrain data be error prone, the system needs to be augmented with camera post refinement. This demands to solve the correspondence problem by matching points or features captured in one image with the similar points or features in another. To perform image correlation, computation of registration points is done. Such computation, however, is time consuming. The registration points are usually acquired through correlation maximization functions performed iteratively. These computations may result in real time performance degradation. Owing to high likelihood of discontinuity and drastic variations in terrain height, the corresponding problem is aggravated in the urban scene. Identified feature points are likely to become occluded and consequently reappearing in few succeeding frames, thus, making correlation difficult.

To remedy, filtering of data pertaining to special situational assessment needs could be performed. Considering an application that requires surveillance of ground vehicle movements, only activities moving on terrain skin are of interest. For such application with only ground terrain of concern, the aforementioned approach offers flexibility to filter and eliminate processing of texture mapping for building surfaces. This is accomplished by assigning zero texture UV values for vertices of polygons belonging to building sides. Figure 4-20 illustrates the generated texture UV values for acquired images. Only terrain geometries belonging to terrain skins are assigned with valid texture UV values.
Eliminating irrelevant data reduces complexity and computation required for camera pose refinement. A commonly used algorithm designed by David Lowe in 1999 to detect features in images is Scale-invariant feature transform (SIFT) (Lowe, 1999). Applying SIFT operations on the images from Figure 4-20 produces 1546, 1302 and 1276 keypoints respectively. When performing SIFT keypoint extraction on images showing only terrain skin, number of detected keypoints reduces to 70, 118 and 93 for the three images. Figure 4-21 illustrates the keypoints identified using SIFT on the second image, from Figure 4-20 before and after filtering texture portions that are non-terrain skins.
Filtering irrelevant terrain data reduce the amount of processing for point matching tremendously. Since only the geometries belonging to terrain skins, which are relatively parallel to horizon plane, need to be taken into account while solving the corresponding problem, complexity dimension of the problem in point matching is further reduced significantly.

Figure 4-22 illustrates the updated 3D urban terrain with only terrain skin texture portion updated. Although real time texture portions for building facades are not applied to the terrain geometries, situational picture of ground activities could still be provided. One potential future enhancement may involve incorporating camera pose refinement using the aforementioned terrain data filtering approach. The refinement allows employment of lower-cost but error prone sensor tracking devices (Sik & Pattanaik, 2012).
4.3.3 Fusing Spatial Information From Disparate Sources

Visualization involves forming internal pictures using perceived external information of the environments (Searle, 1983) (Gibson, 1979) (Ntuen, 2009). It is controlled by both internal and external events. While the mind forms conclusions about the environmental settings based on past experiences, external events provide the triggers and signals for building the mental pictures of current situational settings. When graphically portrayed and presented insightfully, these external events have the potential to aid in gaining a nominal level of situational awareness.
Traditional visualization systems to aid in situational awareness are 2D based. 2D maps are used as backgrounds. 3D visualization could also be used to augment 2D plan view displays for enhanced situational awareness (Durbin, et al., 1998). Visualization in 3D has played essential roles in training and experimentation related simulation. The potential of using 3D visualization for situational awareness, nevertheless, has not been fully realized. This is particularly so in complex and dynamic terrains (Son, Kerbusch, Appleton, & Meijer, 2010). Urban terrains are often populated with numerous small, disconnected 3D objects such as buildings. The terrain geometries are highly discontinuous, and different views of terrain at disparate heights result in drastically different visual effects of the scene. Ground objects, for example, are often fully visible when viewed orthogonally downwards from the bird eye's views. They are, however, usually occluded when viewed from an oblique angle from a distant. The occlusion effect in 3D environments has a detrimental effect on tasks involving discovery and relating objects. When performing some visual-related tasks, studies indicate that there is no benefit of using 3D based over a 2D based presentation (Carvalho, 2011) (Cockburn & McKenzie, 2002).

This section of the chapter describes a 2D based presentation to visualize 2D map and video imageries integrated using one common display using the proposed fusing technique. The approach reduces the operators' cognitive burden in associating spatial positioning and viewing angle of the sensor devices and scene activities. The fusing strategy utilizes an integrated environment that considers inter-object occlusions. It updates 2D maps with geo-referenced imageries using the assumptions that acquired images are coupled with corresponding camera parameters, and terrain geometrical information are available. The method anchors on using
available synthetic 3D terrain to relate information from 3D imagery views of the scene to 2D map (Figure 4-23).

Figure 4-23. Fusing 2D Map And Images.

Figure 4-24 illustrates an acquired image and a reference terrain map. They are often presented to decision makers using different displays. To facilitate information correlation among the displays, fusing useful information from these two disparate sources would be beneficial. While a clear-cut approach involves simply stretching the image over the reference terrain map using selected control points as anchors, when inter-object occlusions are not taken into considerations, information such as shadowed regions are not presented to the operators. Omitting such information may results in unpleasant surprises such as when troops venturing into a supposed
“cleared” area get ambushed by enemies not captured by the surveillance cameras. Highlighting shadowed regions that are not within line of sight of the cameras allows decision makers to know the “unknowns” and be more prepared.

In the prototype implementation, a virtual urban scenario populated with several ground vehicles is created in a simulated environment. Three surveillance Unmanned Aerial Vehicles (UAVs) patrolling an area of interest are added into the virtual scene. An application to model and update positions of the UAVs is implemented. During the simulation run, screenshots of all acquired images of the UAVs and corresponding camera parameters for respective frames are rendered and recorded.
Figure 4-25 illustrates a situation in which operators are presented UAV video frames and respective positions (indicated by color coded circles), with a task to determine locations of detected targets on reference map. One can jolly well imagine that substantial cognitive efforts are required to orientate the video contents with respect to the base map.

Using the proposed texture updating system, a virtual scene with the associated terrain geometries is rendered. This is done using camera parameters obtained from corresponding captured images. The texture UV information of the visible faces is rendered on screen and
saved onto imageries. The 24-bit RGB format is used for storage of U and V information, each taking up 12-bit storage.

In the simulation, the mission objective is to maintain surveillance of ground vehicle movements. Only activities moving on the terrain skin are of interest. For applications with only ground terrain of concern, the proposed approach offers flexibility to filter and eliminate processing of texture mapping for building surfaces as described in section 4.3.2.3. This is accomplished by assigning zero texture UV values for vertices of polygons belonging to building surfaces. Figure 4-26 illustrates pairs of acquired and texture UV rendered images, for three UAV sensor views at a specific time. Only valid texture UV values are rendered for terrain skins. Image portions depicting buildings are rendered with zero texture UV values. For each and every texel from pairs of images obtained from screen captures of the urban scene and corresponding texture UV information imageries rendered using similar camera settings, respective texel colors obtained from screen captures of the urban scene are written onto the addresses stated by the corresponding texture UV information values retrieved from the texture UV rendered imageries. Instead of simply mapping the acquired images onto common ground plane, by considering 3D terrain geometries when fusing acquired images, a more accurate mapping process is achieved.
The terrain texture is initialized using a base map to serve as a "base-skeleton", to be skinned with the acquired images. In the prototype implementation, where ground surveillance is of interest and for illustration simplicity, only orthogonal top view of the terrain skins is visualized.

To perform texture updating, the GPU–accelerated technique, described in section 4.2.4, is extended to fuse multiple pairs of color and spatial-related input textures. Figure 4-27 illustrates the process where three UAV acquired and background imageries are transformed and combined to form a composite situational picture, using one rendering pass. During rendering, $256 \times 256 \times 3$ point primitives are drawn. While performing vertex transformation, each of these points is displaced according to its spatial information that is read from the texture UV rendered images. At the same time, its color is updated to the values read from respective texel of acquired image.
Three layers of 256×256 point primitives are used. This corresponds to three pairs of input textures, each using 256×256 texels in resolution. During rasterization, the displaced and color-updated point primitives, coupled with background imageries combine to form a composite image using an orthogonal viewport.

Figure 4-27. Combining Acquired And Background Imageries Using A GPU-Accelerated Technique
Figure 4-28 illustrates the updated terrain texture, in the form of terrain map, using acquired images. The color boundaries indicate corresponding instantaneous sensor view footprints. The updated image is applied onto the terrain geometries (in this case, only terrain skins are modeled) for real time updates. This facilitates visualization of the terrain map updated with the latest situational picture. As only terrain texture texels with addresses registered in the texture UV rendered images would be updated, other texture portions that are not captured by any sensors retain previous information. As such, previously updated terrain textures remain, and information persistency is achieved. Texture portions are only to be updated and overwritten when new imageries capturing relevant terrain spots are acquired.

Figure 4-28 also illustrates the updated terrain textures, where several overlapping images are fused with the base map, to provide an overall enhanced situational picture. Texture portions depicting "not as current" information is displayed with different shades, to indicate their potentially obsolete status. Highlighting their status allows operators to be more informed. Such information is used when determining if vehicles depicted in image portions remain at similar spots, or have moved to another location. For example, the positional information of the vehicle indicated inside the green circle in Figure 4-28 is not current and may have moved. Nevertheless, presentations of recent major events that may be fully resolved at a slower rate, such as road accidents and jams, building collapses or craters resulting from massive explosions, allow operators to be aware of key events and their locations of occurrences in the scene. Such information is especially critical for missions such as those involving humanitarian relief agents to plan a feasible channel for a rescue team to reach disastrous site. Fusing such mapping and
real-time terrain information allows operators to make spatial inferences in a concrete visualization.

Figure 4-28. Enhanced Situational Picture With Real-Time Imageries

4.4 Performance Evaluation

The performance improvement made using GPU accelerated technique to displace and color point primitives in updating terrain texture atlas is evaluated. The performance is compared to
that using CPU texel by texel iterating and updating approach. Figure 4-29 illustrates the performance results obtained. Time taken to update terrain texture atlas using different number of surveillance images for both approaches is recorded. The resolution of each surveillance image is $512 \times 512$. 500 frames are simulated for each run. It could be observed in Figure 4-29 that, while performance for both approaches are comparable when few surveillance images are used, significant improvement in speed is achieved by the GPU accelerated technique when the number of surveillance images to be updated increases. The time taken to update terrain texture atlas using 15 surveillance images and GPU accelerated technique is almost four times faster when compared to the CPU solution where texture updating is done sequentially.

![Graph showing performance improvement]

*Figure 4-29. Performance Improvement Using GPU Accelerated Approach*
4.5 Evaluating Effectiveness Of The 2D Displays Enhanced With Fused 2D Map And Video Feeds

A research study, that uses both knowledge based and performance based assessment methods, to evaluate the effectiveness of the fusing strategy in presenting information to operators is conducted. In the study, enhanced interfaces are simulated using fused 2D map and real-time images. To evaluate users’ performance improvement when operating with enhanced interfaces, an experimental study is carried out. Additionally, knowledge based assessment of the enhanced interfaces is made using survey questionnaires. The survey also collects subjective ratings on the informativeness of the enhanced displays.

4.5.1 Scenario Background

Figure 4-30 illustrates related activities and data flow during execution of multi-UAVs surveillance mission. The activities indicated in blue are typical tasks to be performed. During mission execution, video feeds acquired by the sensors on-board of these UAVs are monitored by operators. Using 2D map as the backdrop to provide context, the operators make decisions based on information drawn from the video feeds on current situational settings. A fusing system, with relevant sub-activities indicated in red, could be adopted using the aforementioned fusing technique. The sub-processes involve performing simulation using sensors positional information and available synthetic terrain, and generating address encoded image for each and every acquired image. The address encoded images are used to transfer color information from acquired images onto 2D map. The resultant composite display using fused images potentially aids decision makers in understanding the current situational settings.
Figure 4-30. Augmenting Multi-UAVs Surveillance Mission Using Fused Information

4.5.2 Performance Based Assessment Of The 2D Displays Enhanced With Fused Information Using Experimental Study

The research objective of the experimental study is to measure the amount of performance improvements made by operators when using the enhanced displays in terms of speed and accuracy.
4.5.2.1 Hypotheses

Two statistical hypothesis tests are made:

H$_{10}$: Time taken to identify vehicle locations on reference map is not shorter when using enhanced display than that when using baseline display.

H$_{11}$: Time taken to identify vehicle locations on reference map is shorter when using enhanced display than that when using baseline display.

H$_{20}$: Number of attempts made to identify vehicle locations on reference map when using enhanced display is not less than that when using baseline display.

H$_{21}$: Number of attempts made to identify vehicle locations on reference map when using enhanced display is less than that when using baseline display.

4.5.2.2 Experimental Design

A within-subjects repeated-measures design is planned. Two sets of data are collected from each participant from two different treatments. The treatments comprise of participants taking up the roles of operators to monitor and detect targets placed in simulated virtual environments. The first treatment involves providing participants with baseline control interfaces while the second involves using enhanced control interfaces. Seven students participate in the experimental study. They have no experience in operating real-time surveillance and monitoring system. During the experiment, simulated information related to urban scenarios is presented to participants using different interfaces. They are given tasks to identify the locations of the perceived vehicles. The
aim is to detect the ground vehicles as quickly as possible while making as little mistakes as possible.

4.5.2.3 Simulated Scenarios

Three scenarios are crafted. Each scenario consists of four ground vehicles and three Unmanned Aerial Vehicles (UAVs). The three UAVs perform imagery acquisitions. The ground vehicles are stationary or moving. The scenario settings are based on an urban terrain heavily populated with buildings. The vehicle positions are different for different scenarios. Due to drastic variations and discontinuity in terrain heights, the ground vehicles are likely to become occluded from sensor views and consequently reappear after few succeeding frames. This makes locating them on a reference map difficult (Figure 4-31). The UAVs are simulated to patrol the area of interest in fixed circular paths. Sensor views on-board two of the UAVs are directed almost vertically downwards while the third sensor has its "look at" destination fixed at a targeted location. Having a mixture of varied UAV viewing angles allow the simulated scenarios to be truly reflective of real-world settings where UAV viewing angles typically differ from one another.
Figure 4-32 illustrates the boundary indications of the sensor footprints casted on 2D map. The two box-like footprints (indicated in red and blue) are casted by the two sensors that are directed vertically downwards while the third sensor contributes to the trapezoidal footprint (indicated in yellow) due to its slight oblique downward-looking viewing direction.
The sensors capture scene activities where ground vehicles, of types comprise of fire truck, police car, ambulance and delivery truck are placed. Two are static while others are mobile. (Figure 4-33)
4.5.2.4 User Interface Design And Implementation

Figure 4-34 illustrates the implementation approach to simulate displays for baseline and enhanced interfaces. The process would be performed for each and every scenario. It commences with scenario planning. The outputs from scenario planning include ground vehicle movements and UAV flight routes. These data are used as inputs for simulations. Fixed time-step simulations are performed. Sequences of UAVs' sensor views and corresponding address encoded images are created for all simulation frames. To generate images for UAVs' sensor views, Simulation A is executed where virtual scene comprising of synthetic urban terrain and ground vehicles are rendered using UAVs' sensor camera parameters. To generate corresponding address encoded images, Simulation B is executed where virtual scene comprising of only synthetic urban terrain is rendered. The fusing strategy is then applied to the sequences of images and 2D map. The UAV sensor views images, 2D maps and fused images are subsequently loaded and displayed in the user interface applications. The display rates are adjusted to truly reflect real-world rendering conditions, taking the potential frame to frame delay introduced during fusion processing into consideration. This is achieved by introducing a
time delay for each and every frame during rendering. The user interface applications are used during experimental study.

**Figure 4-34. User Interface Applications Implementation**

Typical tactical situational displays for UAV situational awareness update UAV locations on a 2D map using icons. Corresponding video feeds acquired by sensors on-board the UAVs are
displayed alongside. Operators need to associate UAV locations, their sensor orientations and terrain features so as to make sense of the scene activities. In the experimental study, visual appearances of the situational displays to illustrate the scenarios with and without real-time terrain updates are simulated. Figure 4-35 illustrates the two tactical situational displays to be compared in the experimental study. The right image illustrates the enhanced situational display with simulated surveillance acquired images overlay on 2D map, while the left image illustrates the baseline situational display. Both baseline and enhanced situational displays indicate footprints of the surveillance sensors.

![Figure 4-35. Baseline (Left) And Enhanced (Right) Situational Displays](image)

In Figure 4-36, the boxes, indicated in white, illustrate zoom-in visual appearances of image portions depicting ground vehicles.
Figure 4-36. Enhanced Situational Display With Superimposed Video Imageries.

Figure 4-37 illustrates the user interface used for experimental study. The 2D tactical situational display section shows the enhanced display where surveillance imageries are superimposed on 2D map. In the baseline display, no surveillance imagery is superimposed. Only surveillance viewport footprints are displayed. Surveillance videos are simulated at the bottom of the interfaces. An interactive users' input map is also provided on the right side of the interface. The map allows participants to indicate perceived vehicle locations. The participants need to perform mouse clicks on the 2D maps. When a vehicle location has been successfully identified, a white sphere appears to indicate the correctness. Statistical information related to the number of detected vehicles, attempts made, and simulation time progress are also displayed. Similar features and layouts are used and simulated for the baseline displays. The baseline and enhanced displays differ only in presentation of imagery information for the 2D situational display.
4.5.2.5 Procedure

An introductory session is conducted for each participant. The objective is to allow the participants to get familiarized with the scenarios and the required tasks. A ten minute study brief is provided to prepare the seven participants for the experiment. Prior to actual simulation runs, trial runs are given to participants to allow them to get familiarized with the interfaces. Two simulation runs are performed for each participant. Scenarios with different vehicle locations are generated. They are presented to different participants to remove scenario
familiarity effects. Sequences in the simulation runs are different for the different participants to remove potential bias due to learning effects. The data is recorded by the application which simulates the displays. The mouse click events are registered. This allows the computation and recording of users’ number of attempts made and time taken to click on the correct locations on the reference map. At the end of the session, participants are asked their preferred interfaces for the assigned tasks.

4.5.2.6 Experimental Results

Table 4-2 shows the results obtained from the study involving seven participants. Average times taken to detect all ground vehicles are 340 seconds and 97 seconds, while the numbers of attempts are 36 and 5 for baseline and enhanced displays respectively.

<table>
<thead>
<tr>
<th>Participants</th>
<th>Baseline Display</th>
<th>Enhanced Display</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time taken (in seconds)</td>
<td>Number of attempts</td>
</tr>
<tr>
<td>1</td>
<td>327</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>289</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>459</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>87</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>880</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>156</td>
<td>78</td>
</tr>
<tr>
<td>7</td>
<td>183</td>
<td>10</td>
</tr>
<tr>
<td>Average</td>
<td>340</td>
<td>36</td>
</tr>
</tbody>
</table>

Using paired two sample t-Test for the average of time taken and attempts made to identify vehicle locations, p-values of 0.027 and 0.040 are obtained. As such, null hypotheses (H1₀ and H2₀) that time taken and number of attempts made to identify vehicle locations on reference map are not lesser when using enhanced display than that when using baseline display are rejected at
p = 0.05 respectively, in favor of the alternative hypotheses, H$_1$ and H$_2$. Post-hoc power analysis is also performed using data collected after the experiments. The power (1-β error probability), that is, the probability that the effects of speed and accuracy improvement could be detected are found to be 0.677 and 0.586 respectively. The beta errors are large and outside typical range of 0.2. Statistical inferences and recommended future studies are discussed in section 4.6.

Figure 4-38 illustrates the results presented in graphic forms. Relatively large variances for time taken and number of attempts made to identify vehicle locations on reference map are observed when non enhanced displays are used. This may be attributed to the varied level of spatial reasoning skills of the untrained participants. On the other hand, smaller variances for the time taken and the number of attempts made to identify vehicle locations on reference map are observed when enhanced displays are used. This indicates consistency in achieving comparably better performance when enhanced displays are used. Improvement in participants' speed and accuracy is observed for most. Statistical inferences and recommended future studies are discussed in section 4.6.
In addition, all interviewed participants expressed difficulty and frustration when using the non-enhanced displays. All preferred using the interfaces with the video imagery overlay on the 2D map.

4.5.3 Knowledge Based Assessment Of The 2D Displays Enhanced With Fused Information Using Survey Questionnaires

A knowledge based assessment is also made using survey questionnaires. The survey provides additional insights on the informativeness of the enhanced displays. Participants answer questions on scene activities based on information depicted in the simulated perspective sensor views and 2D maps. The first research objective of the survey aims to compare the levels of accuracy in relating information from sensor acquired images to 2D map when using baseline and enhanced displays. The second research objective aims to evaluate if the enhanced display has allowed participants to achieve perfect scores when relating information from sensor
acquired images to 2D map. In the survey, users’ ratings are also collected to measure subjective informativeness of the displays.

4.5.3.1 Hypothesis

Three statistical hypothesis tests are made:

$H_{30}$: The participants' mean score in answering the questions related to associating scene entities from photos to 2D map is not higher when information are presented using enhanced displays than that when information are presented using baseline displays.

$H_{31}$: The participants' mean score in answering the questions related to associating scene entities from photos to 2D map is higher when information are presented using enhanced displays than that when information are presented using baseline displays.

$H_{40}$: Number of participants receiving a score of 1 in answering the questions related to associating scene entities from photos to 2D map is not higher when information are presented using enhanced displays when compared to that when information are presented using baseline displays.

$H_{41}$: Number of participants receiving a score of 1 in answering the questions related to associating scene entities from photos to 2D map is higher when information are presented using enhanced displays when compared to that when information are presented using baseline displays.
H5₀: The displays enhanced with fused display do not allow participants to achieve perfect scores when associating scene entities from photos to 2D map.

H5₁: The displays enhanced with fused display have allowed participants to achieve perfect scores when associating scene entities from photos to 2D map.

4.5.3.2 Survey Questions

Similar survey, described in section 3.5.3.2, is used to gather users' responses. The answers from Q1 and Q3 are of interest in this section of the dissertation.

The questions are:

Q1: Match the targets on the photos and 2D map (based on information presented using baseline interface)

Q3: Match the targets on the photos and 2D map (based on information presented using enhanced interface)

Those who successfully match vehicles depicted on the photos to indications on the maps are awarded with score of 1, a zero score otherwise. Two scenarios, A and B, are crafted. Figure 4-39 illustrates imagery information for the questions. Four diagrams: A1, A2, B1 and B2 are generated. The lettering labels are used to identify the scenario to be used. The numbering labels refer to the display types (baseline or enhanced) to be presented. The enhanced displays have the 2D situational display maps updated with superimposed photos and enhanced with viewshed rendering.
4.5.3.3 Survey Results

Similar group of college students, described in section 3.5.3.2, participated in answering the questions for the survey. The mean scores obtained from the answers on questions related to the information presented using baseline and enhanced displays are 0.415 and 0.509 respectively. A p-value of 0.085 is obtained from two sample t-Test for the mean scores. As such, null hypothesis, \( H_0 \), that the participants’ mean score in answering the questions related to
associating scene entities from photos to 2D map is not higher when information are presented using enhanced displays than that when information are presented using baseline displays, cannot be rejected at $p = 0.05$. Post-hoc power analysis is performed using data collected after the experiments. The power (1-$\beta$ error probability), that is, the probability that the effects of accuracy improvement could be detected is found to be 0.393. The beta error is large and outside typical range of 0.2.

Additionally, an alternative test using chi square is made. Table 3-2 illustrates the numbers of participants who obtained scores of 1s and 0s when information are presented using baseline and enhanced displays.

<table>
<thead>
<tr>
<th></th>
<th>Number of Participants Who Score 1s</th>
<th>Number of Participants Who Score 0s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline Display</td>
<td>44</td>
<td>62</td>
</tr>
<tr>
<td>Enhanced Display</td>
<td>54</td>
<td>52</td>
</tr>
</tbody>
</table>

$\chi^2$ value of 1.898 and $p$ value of 0.168 is obtained. As such, using chi square test, null hypothesis, $H_{40}$, that number of participants receiving a score of 1 in answering the questions related to associating scene entities from photos to 2D map is not higher when information are presented using enhanced displays when compared to that when information are presented using baseline displays, cannot be rejected at $p = 0.05$. This corroborates the need for further studies to be conducted to validate effects.

To answer the second research question if the enhanced displays with fused information have allowed participants to achieve perfect scores when associating scene entities from photos to 2D
map, a one sample t-test is performed to test if achieved score is equal to 1. P value of less than 0.001 is obtained. The Power (1-β error probability), that is, the probability that the effects of accuracy improvement could be detected are found to be larger than 0.999. As such, the null hypothesis, $H_5_0$, that the displays enhanced with fused information do not allow participants to achieve perfect scores when assessing inter-object visibility among different objects, cannot be rejected. Statistical Inferences and recommended future studies are discussed in section 4.6.

Figure 4-40 illustrates the participants' rating on the informativeness of the enhanced display. The survey results indicate that majority of the participants rate the enhanced display to be informative in most settings.

![Figure 4-40. Participants' Rating On The Informativeness Of The Enhanced Display](image-url)
4.6 Summary

Video surveillance has grown significantly in past decade with decreasing price and better capability of cameras. It offers great potential to enhance social control and time critical tasks. To provide enhanced situational pictures of wide area scenarios and enable timely exploitation of these images by critical players, a novel approach to fuse multi-view imageries spatially with synthetic terrains and visualize them in 3D space is described. Such an approach provides the ability to explore and navigate interactively or to play out hypothetical scenarios using virtual terrains or 2D street maps as backdrops augmented with real-time imageries, and thus allows one to gain novel insights from the captured imageries.

The fusing strategy is accomplished by harnessing graphics rendering techniques and exploiting GPU programmability. In light of availability of geometrical information due to recent advances in 3D acquisition technologies, the proposed approach utilizes an integrated environment that considers inter-object occlusion and building polygon based geometries for texture mapping and merging. The approach assumes that acquired images are coupled with corresponding camera parameters. Such an assumption is coherent with the advent of ubiquitous GPS-enabled devices and related technologies (Enhancing GPS Accuracy, 2010) (Han, 2008).

The solution, as described in section 4.2, achieves fusion of geo-referenced images onto 3D terrain at interactive rate. This allows the presentation of real-time terrain rendered with latest situational pictures, using most current surveillance images. A GPU accelerated technique is also adapted to speed up terrain texture updating process. The approach demonstrates potential, as illustrated in section 4.4, to achieve multiple fold performance improvements when numerous
geo-referenced images are to be considered. In section 4.3, the fusing strategy to perform 3D mapping of 2D images onto synthetic 3D models and terrain is also demonstrated and evaluated using implemented prototypes in simulated environments.

A research study, as described in section 4.5, that uses both knowledge based and performance based assessment methods, to evaluate the effectiveness of the fusing strategy in presenting information to operators is conducted. Results obtained from the performance based assessment indicate with a 95% confidence level that speed and accuracy in locating ground vehicles on a centralized 2D map are higher when enhanced displays are used. The power analysis, however, indicates that the beta errors are large. This corroborates the need for further studies to be conducted to validate the effects. In addition, the results obtained from the knowledge based assessment indicate higher mean score and more correct answers made by participants in relating scene activities in photos and a centralized 2D map when enhanced display is used. The difference, however, is not statistically significant. Additionally, the enhanced display fails to allow participants to attain perfect scores when attempting the questions. Nevertheless, results from subjective ratings from participants indicate that a majority rate the enhanced display to be informative in most settings. Further studies are to be conducted to evaluate and validate the effects. Future research studies may adopt larger sampler sizes or better qualified participants to reduce sample variances. To enhance credibility of the study results, subject matter experts' inputs may be sought. A detailed discussion on limitations of the current studies that were conducted and recommended future works are done in section 5.2.4.
CHAPTER 5 CONCLUSION

This dissertation describes the research works on real-time GPU ray tracing and image fusing techniques for enhanced urban situation awareness and understanding. The solutions harness graphics rendering techniques and GPU programmability to visualize urban terrain with accuracy, viewshed analysis and real-time imageries. Research study results indicate that operators’ task improvements when using the enhanced visualization techniques are evident. Some improvements for the study results, however, are not statistically significant. The rendering and fusing techniques can be applied to a wide range of visualization applications to facilitate enhanced situation awareness and understanding. Examples of systems which would benefit from the techniques include scene reconstruction applications, simulators, surveillance and monitoring systems, search and rescue control centers, and defense command and control stations. This chapter summarizes key contributions and lessons learnt. Limitations and challenges encountered while working on the research are also discussed. Last and not the least, potential enhancements and future works are also included.

5.1 Contributions And Lessons Learnt

The proposed visualization techniques demonstrate potentials in enhancing operators’ situational awareness and understanding of urban scenes. Section 5.1.1 and section 5.1.2 detail the contributions for adapting GPU ray tracing rendering technique for improved accuracy and understanding of urban terrain and achieving real time fusion of 2D images and 3D terrain for enhanced situation awareness respectively. Section 5.1.1.2 describes an approach to map
oblique images onto building sides for improved visual appearances. The approach augments current procedural terrain generation systems. It uses similar techniques to fuse 2D images and 3D terrain described in section 5.1.2. Key lessons learnt in the course of the research are described in section 5.1.3. Contributions in relation to enhanced situation awareness and understanding made possible by aforementioned visualization techniques are also discussed in section 5.1.3. Figure 5-1 illustrates the research contributions towards enhancing visualization techniques that may be applied to both non-real-time and real-time terrain updates.

![Figure 5-1. Research Contributions](image-url)
5.1.1 Accurate Visualization And Improved Understanding Of Urban Terrain

The first contribution involves enhancing and adapting approaches that leverage on GPU ray tracing. Rasterization is currently the most common rendering technique for real-time visualization. Ray tracing, on the other hand, is traditionally employed to render scene with high visual realism, but at a non interactive rate. Today's programmable GPU can very well accelerate ray tracing due to the parallel nature of the problem. In the research, GPU ray tracing is used to aid in achieving accurate visualization and improved understanding of urban terrain. Hybrid rendering approaches using both rasterization and GPU ray tracing are adopted to achieve accurate visualization and viewshed rendering of urban terrain. The implementation methods and details are described in Chapter 3.

5.1.1.1 Contributions To Accurate Modeling And Visualization Of Synthetic Urban Terrain

The synthetic terrain representation is one of the backbone elements for modeling and simulation. Urban terrains are rich in visual and functional complexity due to various cultural factors. Managing and visualizing such abundant and rich urban-related information effectively for terrain-related tasks become a challenge for terrain modelers and graphics engineers.

5.1.1.1.1 Using Enhanced BlockMap For Distant City Blocks To Achieve Accurate Representation And High Rendering Rate

Current state of the art polygon based terrain representations achieve interactive rendering rate using coarse distant terrain representations. Using such simplified representations, nevertheless, result in reduced geometric accuracy for distant 3D objects. While the approach using simplified distant 3D objects offers reasonably compelling visual cues about the terrain features, when
presented concurrently with simulation entities placed in the scene, incorrect depictions become
evident for distant objects. The visual effect is evaluated and measured in section 3.1.1 of
Chapter 3 where 16.1% of randomly placed red spheres become incorrectly exposed due to
object simplifications. To visualize correct spatial relationships among simulated entities and
terrain features, accurate geometrical details of the terrain need to be represented and rendered.
This is particularly so for distant terrain features whose level of details are often compromised in
most rendering systems to achieve high rendering rate.

BlockMap, an image based rendering approach, is adopted for the research. BlockMap is a
GPU-friendly simplified representation of urban environments and can be used to encode coarse
representation of blocks of terrain buildings when rendering far-away city blocks. In the
research, enhancements are made to classic BlockMap rendering and encoding techniques to
dress shortfalls of the standard BlockMap. The enhancement eliminates occlusion anomalies
inherent to the classic BlockMap. It is demonstrated in section 3.3 that visualizing urban terrain
using enhanced BlockMap achieves high level of spatial accuracy almost matching that when
full geometric detail terrain is used while maintain high rendering rate comparable to typical
multi-level LODs terrain. Such rendering solution is especially well suited for use in systems
that require high fidelity in representing inter-object relationships. This is achieved by
computing and updating depth information for each and every BlockMap surface fragment (as
described in section 3.1). Doing so also allows the representing and visualizing of simulation
entities and BlockMaps together in common spatial space.
Another improvement involves adopting a flexible utilization of BlockMap. This is done by representing individual, complex buildings and assigning different LOD switch values for different buildings. The solution aids in optimizing computational resources and improving rendering performance (as described in section 3.2). Lastly, accelerated ray casting techniques are used to achieve high rendering rate (as described in section 3.4). The enhanced BlockMap representations, whose computational complexity depend on rendering screen footprints, offer great potential for use in future urban terrain representations. This is especially so in view of the advancements in GPU technology, and increasing demands for correlated simulation environment with complex urban terrains modeled with both high visual and spatial accuracy.

5.1.1.1.2 Perform 3D Mapping To Construct Photorealistic Vertical Wall Surfaces For Close View Objects

Section 5.1.1.1.1 describes improvements made in representational accuracy for distant terrain objects using enhanced BlockMap. Another research area involves enhancing terrain generation process for close distant polygon based representation. To create urban polygon based terrain, procedural modeling is commonly used to speed up the generation process. One of the commonly used methods to generate geo-specific urban terrain entails performing building extrusions from footprints and draping satellite imageries over building rooftops, with generic wall textures applied to building facades. Examples of terrain generation tools that generate generic wall textures for buildings include Presagis' Terra Vista and Cogent3D's GenesisRTX. These procedural modeling engines, that perform auto extrusion and without additional inputs to associate geo-specific textures to building sides, fail to reproduce building facades in synthetic form precisely. The advancements made in remote sensing and 3D reconstruction related
techniques have contributed to availability of huge database of geo-spatial information. Imageries capturing oblique views of building facades could be used to create geo-specific buildings. To do so, a process to identifying relevant texture portions from this huge database of geo-referenced imageries, and associating them to relevant geometries in efficient and rapid manner is needed.

To streamline the process, an approach that involves having a system to extract relevant and apply oblique 2D images onto building geometries is designed. The solution is described in section 4.2. The approach relies on the use of texture UV as indices to relate texels from captured images to polygon surfaces. The solution leverages on computer graphics rendering techniques to render corresponding spatial information of a virtual scene. It uses known camera parameters and given 3D terrain geometries. A centralized texture UV mapping system is used both for rendering and mapping process. The solution creates and updates a centralized texture atlas using several input images. Prototypes involving texturing a 3D model using the mapping approach are implemented and described in section 4.3.1.

5.1.1.2 Real-Time Visualization Of Multiple Viewshed Analysis

As spatial data of higher resolution and spanning over a larger area become readily available, the demands for speedy processing of terrain-related data inevitably increase. In the research, ray tracing, a technique commonly used by computer graphics communities to render realistic imageries, is used to render multiple viewshed analysis at real time. Analogous to rendering shadowing effects, similar technique is employed to render multi-view real-time viewshed analysis. A hybrid approach using rasterization and ray tracing to render viewshed is described
in section 3.5. Such potentially helpful real-time viewshed analysis is lacking in most current decision support applications. Section 3.5.3 describes a research study, that uses both knowledge based and performance based assessment methods, to evaluate the effectiveness of rendering viewsheds in presenting information to operators. Study results indicate that speed and accuracy in determine spatial relationships among 3D entities in the virtual scene are higher when enhanced displays are used. Some test results, however are not statistically significant at 95% confidence level. There is a need for further studies to be conducted to validate the effects. Result details are described in section 3.6.

5.1.2 Real-time Fusion Of 2D Images And 3D Terrain

Painting an enhanced situational picture, using imagery information acquired from disparate sensors, often demands efficient techniques to process volumes of geo-spatial data, and present them in insightfully manner. Similar to the approach adopted to achieve rapid 3D mapping to construct photorealistic vertical wall surfaces, the use of texture UV as indices to relate texels from captured images to 3D terrain polygon surfaces is adopted in the research to fuse 2D images and 3D terrain at real time. The approach, as described in section 4.2, achieves real-time fusion of geo-referenced images and prior constructed synthetic urban terrain. Current state of the art solutions essentially belongs to two groups. Each strives to either provide near real-time situational pictures in 2D or off-line complex 3D reconstruction for subsequent usages. The proposed solution achieves rapid and persistent terrain texture updates in 3D at interactive rate. A GPU accelerated technique, as described in section 4.2.4, is also adopted to speed up terrain texture updating process. The approach, as illustrated in section 4.4, demonstrates great potential
to achieve multiple fold performance improvements when numerous geo-referenced images are to be considered.

In addition, the proposed system to perform 3D mapping of 2D images onto synthetic terrain is also demonstrated and evaluated using implemented prototypes in simulated environments. Section 4.5 describes a research study, that uses both knowledge based and performance based assessment methods, to evaluate the effectiveness of the fusing strategy in presenting information to operators. Study results indicate that speed and accuracy in relating scene activities depicted in images to a centralized 2D map are higher when enhanced displays are used. Some test results, however are not statistically significant at 95% confidence level. This corroborates the need for further studies to be conducted to validate the effects. Result details are described in section 4.6.

5.1.3 Lessons Learnt

Urban activities that involve planning and responding to time critical situations demand sound situational awareness and understanding of overall settings. Recent increases in the volumes of geo-spatial data and their accessibility can influence the way decision makers make assessments in time-critical situations. Modern GPUs, with their availability on commodity computing platform and combined with their high degree of programmability, have opened up a wide field of applications far beyond processing millions of polygons. It is also possible to leverage on GPU to perform purposeful non-visual and analysis related calculations in real time.
5.1.3.1 GPU Ray Traced Rendering And Image Fusion Based Techniques For Enhanced Visualization of Urban Terrain

This dissertation demonstrates great potentials of the computational power and flexibility of GPU computations and programmability to formulate useful and efficient rendering techniques to aid in enhancing urban situation awareness and understanding. Two techniques are worked on. One of the techniques involves using ray tracing to visualize complex distant city blocks and viewshed rendering. Unlike rendering performance for rasterization which is affected by scene complexity, rendering performance for ray tracing is output sensitive. The larger the area is to be rendered using ray tracing, the more rays and processing are required. In both BlockMap and viewshed rendering, ray casting and tracing are used to analyze and render scene of high complexity. Due to the output sensitive nature of ray casting and tracing, rasterization is used in combination with the techniques to achieve efficient rendering. Rasterization is used to render close view objects for BlockMap LOD integrated terrain. It is also used to perform initial visibility test when rendering viewshed. In BlockMap rendering for distant terrain representation, the city blocks are to be rendered occupy relatively small footprints. When rendering viewshed, only selective area of interest, such as ground areas and those within ranges of sensor coverage need to be considered. In urban terrain where scene complexity is high, GPU ray tracing is especially well suited for use in general rendering and analysis purposes. This is especially so when the spatial relationships such as inter-visibility between moving points of interests are highly dynamic due to the high occurrence of inter-object occlusions and abruptness in height differences.
The other technique involves fusing 2D images and 3D terrain. The technique demonstrates its usefulness in various tasks in both offline terrain generations and real-time terrain updates. The approach achieves terrain texture updates at interactive rates and is relatively invariant to scene complexity. Such an approach that fuses and visualizes multi-view imageries mapped spatially onto 3D terrain allows one to gain novel insights from the captured imageries. The solution is especially well suited to be used in urban terrain where high likelihood of inter-object occlusion makes understanding of the situational settings difficult. The geo-spatial information that can be fused may be represented in 2D or 3D, differs in time space, resolutions or even spectral types. The output displays could be rendered in 2D or 3D, depending on visualization needs. The fusing technique can also be applied to augment terrain generation process by performing autotexture mapping of oblique images onto building facades, and to generate textures for low LOD versions of the 3D models.

5.1.3.2 Enhanced Visualization of Urban Terrain For Improved Situation Awareness

As described in section 2.1, Endsley's (Endsley, 1995) theory on situation awareness begins with perception of elements in the environment. An accurate depiction of various entities or players using a virtual environment aids in forming the basic building blocks for comprehension and projection. Typical rendering techniques often compromise terrain geometric accuracy to ensure good rendering performance. The enhanced GPU ray cast BlockMap implementation for distant city block rendering demonstrates the technique's strength to achieve both high geometric accuracy and rendering rate. Accurate spatial relationships among entities and distant terrain features can be visualized using BlockMaps. When augmented with the ability to visualize
viewshed for inter-visibility checks, enhanced situational understanding could be formed. The improvements demonstrated by the participants in a research study described in section 3.5.3, makes it evident that using an enhanced interface helps in achieving enhanced situational understanding. Additionally, according to Endsley, comprehension involves integration of various pieces of information and a determination of their relevance to the underlying goals. The ability to integrate views from different sensory devices increases situational awareness. The proposed technique to fuse 2D images and 3D geometries aids in integration of such disparate and fragmented terrain geometries and imagery information. The improvements demonstrated by the participants in a research study described in section 4.5, makes it evident that using a fused display helps in achieving enhanced situational awareness.

Figure 5-2 illustrates potential influences to increase levels of perception and comprehension of the current situation by the rendering techniques to visualize terrain information. The ability to explore and navigate interactively or to play out hypothetical scenarios using such up to date virtual terrains augmented with real-time updates and analysis allows one to gain novel insights from the captured imageries. This potentially contributes to the ability to achieve projection of future status, and thus provides the capability to achieve the highest level of situation awareness.
Fusing disparate information and depicting spatial relationships among the entities allows enhanced situational awareness and understanding to be gained. By fusing and mapping elements and spatial relations collected from different sources onto a common context, a concrete visualization of the situational settings can be formed. The enhanced interfaces have, to some degree, demonstrate to relieve spatial working memory workload when performing some spatial reasoning tasks. Due to resource constraints, scenarios used for the experimental studies, as described in sections 3.5.3 and 4.5, are crafted to be executed within minutes. Hence, the experimental studies performed on interfaces enhanced with viewshed rendering and fused information focused on investigating efforts on operators’ cognitive workload requiring relatively short term memory. Future works on enhanced interfaces may involve utilizing more complex scenarios. They are to be simulated over longer periods of time to examine efforts on operators'
performances on tasks requiring perceived information retrieval from long-term memory. Painting such enhanced situational pictures of complex scenario using enhanced interfaces could potentially reveal insightful information, such as patterns in group behaviors or movement trends. Such enhanced situational pictures made possible by the enhanced interfaces potentially aid in gaining high levels of perception and comprehension of the settings.

5.2 Research Limitations

While the prototypes implemented using the proposed approaches demonstrate promising results and potentials, several research limitations exist. This section lists the limitations encountered.

5.2.1 Limitations Of BlockMap Representation

Though characterized by compactness in storage, BlockMap, as described in chapter 3, possesses limitations when representing some terrain features. The BlockMap is designed using assumptions that buildings have a prism-like shape and the majority of the terrain objects are buildings. A 2D raster height map is used essentially to define geometric shapes of the urban terrain features. Such representation approach, nevertheless, is incapable of representing shapes with concave side profiles. Examples of terrain features with such concave or indented side profiles include flyovers, bridges, and tunnels. Approaches to overcome this shortfall to depict depressed or indented vertical surfaces while maintain interactive rendering rate need to be formulated. One potential technique may involve extending the proposed hybrid rendering technique. The technique may comprise of using polygon based representation to carve out the general shape of the terrain, and using ray tracing to put in surface details. Such an approach
requires efficient data structure with indexing system to associate terrain surfaces to textures that encode respective surface details. Another alternative technique may involve using rendering using voxel terrain representation. Storage and manipulation of voxel representation, however, may require large amounts of memory. A memory efficient data structure is to be devised to overcome the limitation.

5.2.2 Using Real World Data For Information Fusion

One key limitation while working on the research is the lack of access to real world data to test the fusing techniques. Synthetic screen captures of 3D virtual scene are used extensively to overcome the limitation. The real world data, that are lacking for use in the research, include input information such as geo-referenced imageries, and corresponding LIDAR constructed terrains to be used for 3D mapping of 2D imageries. The proposed fusing strategy has demonstrated robustness against varied geometry data while performing 3D image mapping onto simplified 3D model in simulated environments (as described in 4.3.1). Moving beyond using simulated environments, testing the fusing strategy using real world data could verify its feasibility and effectiveness against potential terrain geometrical errors introduced by 3D reconstruction processes. Due to resources constraint, the research also fails to explore the effectiveness of the proposed 3D mapping technique under diverse sensory and environmental conditions. The work fails to address challenges such as combining imageries with contrasting shades and hues due to different exposures or geo-tagged with erroneous geo-referencing data. Testing the fusing strategy using real world imagery data acquired from actual sensor devices and beyond using simulated environments could verify the strategy’s feasibility to fuse real
world data. Additional camera pose refinement operations may be required to augment the solutions.

5.2.3 Integrating Proposed Techniques With Established Terrain Generation And Visualization Systems

Current state of the art simulation based terrain generation and visualization systems for decision support aim to having multiple clients pulling data from a centralized server. The approach attempts to achieve having a correlated terrain representation among different client systems to facilitate distributed training or network centric operations. The research focuses in advancing visualization techniques to address "gaps" to achieve accurate terrain representation, and render accurate and insightful displays. Approaches to evaluate the solutions, however, are confined to small scale implemented prototypes. Additional issues or insights on the efficiency of the research solutions as parts of larger systems could be drawn if the solutions are integrated and tested with the aforementioned server-clients systems. Some potential issues may involve inadequate network bandwidth between the centralized server and rendering clients or latency introduced by remote sensors performing image acquisition.

5.2.4 Subject Matter Experts' And Other Inputs For The Comparative User Studies

There is a lack of subject matter experts' input and other additional inputs when designing and implementing the comparative user studies. More insights or additional useful considerations could be derived from the inputs. Realistic and accurate simulations of actual interfaces, in use by real world systems, could be used for the comparative user studies. Doing so potentially gathers more indicative results. Given more time and availability of a wider range of human
subjects for the comparative user studies, more participants from different disciplines and diverse backgrounds could be recruited and studies conducted. This enables in depth performance improvement assessments of the proposed interfaces taking into considerations of effects of trainings, participants' experience and other factors. Lastly, the scenarios used for experimental studies are crafted based on common sense and non-expert inputs. More realistic scenarios that leverage on tacit knowledge from related subject matter experts could have being crafted. Using more realistic scenarios when simulating and presenting situational settings to participants during the experimental studies would potentially increase the credibility of the results obtained.

5.3 Potential Enhancements And Future Works

The followings list potential enhancements and future works related to the research.

5.3.1 Integration Of Proposed Solutions As Part Of Large Scale Visualization System

Enhanced situational awareness of the environment requires a sound understanding of the general characteristics of the terrain and the limitations and opportunities the environment offers. Providing a common operating picture to all coordination sites for urban related activities requires a central terrain, updated with the latest changes. The use of streaming geo-spatial data and client-server architecture allow multiple clients to share terrain data and explore city models for simulation and visualization applications. As discussed in section 2.1.3, current solutions, using streamed terrain data to generate "on-the-fly" terrain at client machines, is only well suited for applications requiring fast turnaround prototyping without a high amount of details (Wiesner, Brockway, & Stanzione, 2011).
In the research, the techniques aim to address limitations of current terrain generation and visualization processes. The constraints of typical current urban terrain generation and representation include lack of unique texture mapping, and speedy generation process for building facades. As discussed in 2.4.1, in the case when each and every client machine uses proprietary and unique algorithms and methods in creating terrain geometries, achieving correlated terrain representations becomes a challenge (Hieb & Maxwell, 2012). Future works involve integrating the proposed techniques as parts of large scale server-clients architecture. Their efficiency in addressing current shortfalls of the system could be investigated. A central terrain database could be utilized. It can comprise of both high LOD polygon based and low LOD BlockMap representations. The terrain database can be maintained by a high performance server, which are responsible for generating and updating terrain related information. Doing so off-loads terrain creation processing by the clients. To reduce the amount of data to be streamed to clients, one possible solution may involve streaming only texturing information at real-time, with only occasional critical on-demand terrain geometry updates being pushed to clients. As stated in section 2.4.4.6, to facilitate terrain viewing and navigation over a large area, streaming of BlockMap data could be done for the rendering of distant terrain features. (Benedetto, Cignon, Ganovelli, Gobbetti, Marton, & Scopigno, 2009) demonstrates interactive remote visualization of large urban environments using city models, represented using multi-resolution BlockMaps, stored on a remote server.

BlockMaps provide compact and bounded error coarse representations. They are used as accurate representations for far away distant city blocks and can be seamlessly integrated into hierarchical data structures. When enhanced and updated with surfaces depth corrected,
BlockMaps are especially well suited for use in interactive simulation in large urban environment (as described in 3.2.3). BlockMap data textures could be generated on the terrain server and streamed to multiple clients. With spatial information encoded and easily rendered at real-time, using BlockMaps as coarse terrain representations off-load terrain generation efforts required by client machines. With the constant footprint of BlockMaps, memory management is particularly simple and effective. As stated in section 2.4.4.6, spatial errors of BlockMap representation is bounded by its resolution and could be estimated and likely to be identical among different clients (Cignoni, Di Benedetto, Ganovelli, Gobbetti, Marton, & Scopigno, 2007). When used in concert by all clients, increased correlated terrain representations could be attained. Future works involve using BlockMaps as part of the streamed terrain data in client-server architecture to allow sharing of correlated terrain data among multiple clients.

Additionally, to enable precise building wall surface representations, the proposed centralized texture mapping and rendering system could be adopted in the server-clients system. The fusion and texture updating process could be performed at the terrain server. The updated textures could be streamed to the client machines for rendering. A common polygon based terrain geometries adopting the centralized texture rendering system could be generated. This terrain database potentially can be maintained by the terrain server and distributed and shared among the client machines. In typical urban scenes, massive large scale changes, such as craters caused by explosions or piles of rubble gathered due to collapses of buildings take place less frequently than occurrences of comparatively small-scale changes, such as movements of vehicles or changes in states of any objects of interests. In most situational settings, frequent updates in texture-related information and occasional modifications in geometrical-related information are
sufficient to provide up-to-date situational pictures. The geometrical updates would have to be made for the low LOD BlockMap and high LOD polygon based representations on terrain server. Craters and piles of rubble could potentially be represented as part of the terrain skins with modified heights for both representations. An updated and enhanced situational visualization of an urban setting can be presented to decision makers using different update rates for terrain geometries and texturing information. This can be achieved by periodic on-demand or event triggered updates of the terrain geometries and real-time updates of accompanied terrain textures.

### 5.3.2 Improvements To The Enhanced BlockMap Representation

The proposed approach, as described in 3.3.1.2, in generating BlockMap lower LOD representations provides efficient rendering that requires little human inputs in terrain LOD modeling process. Potential enhancements to streamline the BlockMap LOD modeling process include automating the process of identifying geometrically complex buildings, and assigning the values for LOD parameters based on their geometrical complexity. A value tagged to each building computed by finding the ratio of the building geometrical complexity to the area it spans across rendering screen could be used to assign the LOD “switch in” value. This allows geometrically complex or smaller sized buildings to switch to comparatively more efficient BlockMap representations at a closer distance. Geometrically less complex or large-sized buildings, on the other hand, could be rendered more effectively using polygonal representations at a moderate distance.
BlockMap rendering could also be extended beyond representing individual complex buildings to render any isolated portion of architectural structures, such as for a geometrically complex balcony modeled on a vertically unadorned and simple building side. The wall could be represented using textured polygon while BlockMap is used to represent the geometrically complex balcony. The approach, described in 3.3.2, to encode wall depth for complex buildings assumes that buildings have 4 walls, which are constructed perpendicularly to one another. Additional slices of wall information could be included for buildings comprising of more than 4 walls.

5.3.3 Combining BlockMap Rendering With Other Rendering Techniques To Visualize Terrains Of Other Types

Urban terrains, rich in visual and functional complexity as a result of developments due to various cultural factors, are difficult to model. The research works have leveraged on GPU ray tracing to achieve accurate visualization and improved understanding of urban terrains. Prior research has been devoted to rendering open terrain from height map data using hybrid approach, which may involve polygon based rasterization and GPU ray casting. The approaches, however, fail to encode color information for vertical surfaces. The BlockMap data structure overcomes this limitation by encoding vertical wall color information. This, however, requires additional storage to store wall color information, which may not be efficient in memory usage. Future works possibly involves improving BlockMap wall indexing system to increase flexibility in encoding approach. Other potential enhancements involve incorporating rendering techniques to render objects such as trees or concave structures. Doing so would allow easy adopting of BlockMap representations in other terrain types such as open terrain or sub-urban terrain where
buildings are sparse and large portions of the terrain may include other terrain feature types such as trees and forests.

5.3.4 Validating BlockMap LOD Integrated Terrain

In urban environments, different views of terrain at disparate heights result in drastically different visual effects of the scene. Ground objects, for example, are usually fully visible when viewed orthogonally downwards from the bird eye's views, but are usually occluded when viewed from an oblique angle from a distance. The characteristics of these scene features such as height and size of buildings and other 3D terrain features potentially affect the amount of occlusions. When performing virtual simulations such as those involving UAV surveillance target detection missions, visualizations of correct spatial relationships among simulated entities and terrain features is important. Accurate geometrical details of the terrain need to be represented and rendered. This is particularly so for distant terrain features whose level of details are often compromised in most rendering systems to achieve high rendering rate. It is demonstrated in section 3.3 that visualizing urban terrain using enhanced BlockMap achieves a high level of spatial accuracy almost matching that when full geometric detail terrain is used while maintain high rendering rate comparable to typical multi-level LODs terrain.

Future research to validate visual quality of the BlockMap enhanced terrains involves comparing the levels of alpha and beta error in target identification when visualized using different rendering techniques. The rendering techniques to be compared potentially involve polygon based full detail terrain, polygon based LOD integrated terrain and BlockMap LOD integrated terrain as used in section 3.3. Still imageries depicting scene populated with target positions at
various distances may be provided to participants for the identification process. The collected levels of alpha and beta errors made by participants in target identification must be compared and analyzed. The closer the levels of alpha and beta error using imageries rendered by a candidate technique match to that achieved when using imageries rendered by a full detail terrain, the better the rendering approach in depicting correct scene information is. When evaluating the result collected, potential trade-offs in relation to rendering rates achieved by specific rendering methods, as described section 3.3, need to be taken into account. In addition, one needs to be mindful of the limitation of BlockMap represented terrain where full 3D terrain features such as trees or flyovers may not be represented accurately. Objects, when placed at or in proximity to these features, are likely to be occluded when viewed orthogonally downwards but partially visible when viewed at an oblique angles. Future research on the rendering techniques to address such "concave" structures, need to be carried out as well.

5.3.5 **GPU Ray Tracing For Terrain Analysis**

In terrain analysis, computational complexity involved in terrain-related calculations grows in tandem with the resolution and richness of source data. A high-performance visualization engine, that exploits the massively parallel computation architecture of modern GPUs, is especially well suited to handle rendering and computation of large-scale complicated 3D geospatial environments. The research demonstrates the usefulness of visualizing multi-view real-time viewshed analysis using GPU ray tracing rendering technique. GPU ray tracing, which mimics the behavior of light in the real world, are well suited and can be used to analyze and simulate terrain effects on sensor or communication systems. Similar rendering technique could
be applied to other terrain analysis tasks such as terrain path finding, trafficability of terrain analysis, and assessing terrain impact on communication and related analysis.

5.3.6 Enhancements For The Fusion Technique

The proposed approach to fuse 2D images and 3D geometries assumes accurate camera parameters are fed into the system. Such information fusion is useful in rapid terrain generation and situational awareness visualization applications. Future endeavors involve testing the fusing strategy using real world data to verify its feasibility and effectiveness against potential terrain geometrical errors introduced by 3D reconstruction processes. Other works may involve exploring the effectiveness of the proposed 3D mapping technique under diverse sensory and environmental conditions. Challenges such as combining imageries with contrasting shades and hues due to different exposures or geo-tagged with erroneous geo-referencing data need to be addressed.

One potential enhancement for situational awareness visualization applications involves incorporating camera pose refinement to the fusion process. Doing so increases robustness of the system against error-prone geo-registration components. The proposed solution to perform texture mapping onto terrain geometries demands accurate building geometries, and camera pose information for image rasterization. Such approach is susceptible to erroneous texture mapping if given information differs from ground truth. This demands to solve correspondence problem by matching features from one image with the same features in another image. Such computation of points registration is time consuming. To remedy, filtering of data pertaining to special situational awareness needs could be performed. Eliminating irrelevant data reduces the
complexity and computation required for camera pose refinement. For example, typical surveillance missions are concerned with activities happening on the ground. Imagery information capturing changes or activities occurring on buildings could be disregarded. This can be achieved by assigning invalid or not rendering valid spatial related information for building surfaces when generating address encoded images. Filtering irrelevant terrain data reduces the amount of processing for point matching tremendously. Since only the geometries belonging to terrain skins, which are relatively parallel to horizon plane, need to be taken into account while solving the corresponding problem, complexity dimension of the problem in point matching is reduced significantly.

Other enhancements to aid in rapid terrain generation may involve texture synthesis to create texture portions that are not captured in the input imageries. This can be accomplished using neighboring texture patterns or using rule-based logics. This allows usage of imprecise or incomplete geo-referencing information to aid in robust creation of geo-specific building facades. Similarly, the technique will be useful to achieve rapid terrain generation using incomplete imagery information.

### 5.3.7 Potential Use Of The Proposed Fusion Technique In Other Applications

The fusion techniques that relate 2D images to 3D terrain can be used in a wide range of applications. Examples of systems which would benefit from application of the fusion technique include scene reconstruction applications, simulators, surveillance and monitoring systems, search and rescue control centers, and defense command and control stations. When used in simulation or analysis related applications, the fusing technique could relieve operators’
cognitive workloads in interpreting information from 2D perspective images with respect to a centralized terrain. This is particularly so for those who need to analyze or monitor tons of 2D perspective video images or still photographs. Additionally, the fusing technique could be used to enhance coverage analysis, commonly performed for surveillance route planning related tasks or evaluating strategic points of locations. One software that performs coverage analysis is Systems Tool Kit (STK), formerly termed Satellite Tool Kit, from Analytical Graphics. Instead of simply highlighting areas of visibilities from input routes, additional insights may be gained if simulation of hypothetical scenarios could be included. A simulation could be performed to determine if proposed UAV flight routes are effective in detecting ground entities. The simulation may involve using scenarios, populated with ground entities and UAVs. Screen captures of the virtual scene, using views from the UAVs, could be generated. Using the fusing strategy described in chapter 4, a comprehensive picture, obtained by fusing and putting screen captures onto respective terrain portions, could be painted. This allows insights on potential unobserved or occluded areas of concerns that are highlighted. Visual contrasts or resolutions that potentially affect depiction or detection when viewed from surveillance devices could also be investigated. Similar technique could also be applied for After Action Reviewer (AAR) Tools when reviewing scenario events and players’ actions. A fused display that comprises of multiple perspective views observed by participating players projected on a common display, could possibly aid reviewers in evaluating respective players’ performances while taking into consideration of their situational awareness of the surroundings.
5.4 Summary

The work on this dissertation adapts GPU ray tracing rendering technique for improved accuracy in visualization and understanding of urban terrain, and achieves real time fusion of 2D images and 3D terrains for enhanced situational awareness. The rendering techniques have, to some degree, been proven to enhance urban situational awareness and understanding. The use of viewshed analysis and real-time imagery mapping is used to demonstrate enhanced situation awareness. Evaluation results indicate some enhanced degree of accuracy and performance using the enhanced visualization techniques. Some limitations in the GPU ray casted BlockMap involve the failure to consider full 3D structures. Limitations for image fusion techniques include the need for accurate scene geometries and camera pose information. Future research on GPU ray tracing and image fusion techniques should concentrate on addressing the aforementioned limitations. Additional evaluation on the techniques' efficiency as part of visualization tools should include acquiring subject matter experts' and recruiting more qualified participants for the comparative user studies, validating solutions using real-world data, and investigating the efficiency of the solutions as part of large scale visualization systems.
APPENDIX IRB APPROVAL LETTERS
Approval of Human Research

From:  UCF Institutional Review Board 
FW A00000351, IRB00001138

To:  Ling Ling Shu

Date:  September 04, 2012

Dear Researcher:

On 9/4/2012, the IRB approved the following human participant research until 9/3/2013 inclusive:

- **Type of Review:** UCF Initial Review Submission Form
- **Project Title:** Performance based assessment for situational awareness using enhanced interfaces
- **Investigator:** Ling Ling Shu
- **IRB Number:** SBE-12-68656
- **Funding Agency:**
- **Grant Title:**
- **Research ID:** N/A

The Continuing Review Application must be submitted 30 days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form cannot be used to extend the approval period of a study. All forms may be completed and submitted online at [https://iris.research.ucf.edu](https://iris.research.ucf.edu).

If continuing review approval is not granted before the expiration date of 9/3/2013, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in IRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Sophia Dziegielewski, Ph.D., L.C.S.W., UCF IRB Chair, this letter is signed by:

Signature applied by Joanne Muratori  on 09/04/2012 01:43:04 PM EDT

IRB Coordinator
Approval of Human Research

From: UCF Institutional Review Board #1
FWA00000351, IRB00001138

To: Ling Ling Sik

Date: September 19, 2012

Dear Researcher:

On 9/19/2012, the IRB approved the following minor modifications to human participant research until 09/03/2013 inclusive:

Type of Review: IRB Addendum and Modification Request Form
Modification Type: Surveys have been added to obtain more results from a larger group of participants. Total number of study participants approved is now 500 and a revised Informed Consent has been approved for use.
Project Title: Assessment for situational awareness using enhanced interfaces
Investigator: Ling Ling Sik
IRB Number: SBE-12-08636
Grant Title: N/A
Research ID: N/A

The Continuing Review Application must be submitted 30 days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form cannot be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu.

If continuing review approval is not granted before the expiration date of 09/03/2013, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in iRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Sophia Dziegielewski, Ph.D., L.C.S.W., UCF IRB Chair, this letter is signed by:

Signature applied by Joanne Muratori on 09/19/2012 09:10:52 AM EDT
REFERENCES


