

Context-Aware Information Access and Retrieval for Rapid
On-Site Decision Making in Construction, Inspection and
Maintenance of Constructed Facilities

by

Hiam Mayez Khoury

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Civil Engineering)
in The University of Michigan
2009

Doctoral Committee:

Professor Photios G. Ioannou
Associate Professor Vineet R. Kamat, Chair
Adjunct Associate Professor Klaus-Peter Beier
Lecturer John G. Everett

© Hiam Mayez Khoury 2009
All Rights Reserved

To My Lovely Family,
That if it were not for their help and support,
I could not make my way towards my every success.

Acknowledgments

I would like to thank my advisor, Professor Vineet R. Kamat, for his great help and patience throughout my graduate studies at the University of Michigan. His continuous guidance and complete support made my working and learning experience a very special and enjoyable one.

I also want to extend my thanks to the members of my Ph.D. committee, Professors Photios Ioannou, John Everett, and Klaus-Peter Beier, for their support and enthusiasm in providing me with invaluable advice and deep insight and suggestions to improve the quality of my work.

Besides, I would like to express my greatest appreciation to my parents, Mayez and Sara Khoury, my sisters, Lara and Layal Khoury, and my lovely brother Jean Khoury, for the love and encouragement they endowed me during my studies. I would also like to convey my deep gratitude to the support I received from all my friends.

Hiam Mayez Khoury

Table of Contents

Dedication	ii
Acknowledgments	iii
List of Figures	ix
List of Tables	xvii
List of Appendices	xviii
Abstract	xix
Chapter	
1. Introduction	1
1.1. Background	1
1.2. Problem Statement	2
1.3. Research Objectives and Challenges	6
1.4. Research Contributions	8
1.5. Dissertation Outline	9
1.6. References	12
2. Context-Aware Information Retrieval Methodology: Technical Overview	15
2.1. Introduction	15
2.2. Literature Review	16
2.3. Technical Approach	21

2.3.1.	Tracking a User’s Spatial Parameters	23
2.3.2.	User Spatial Context Interpretation	24
2.3.3.	Relevant Contextual Objects Identification	26
2.3.4.	Contextual Information Retrieval and Visualization	26
2.4.	Summary and Conclusions	27
2.5.	References	29
3.	Evaluation of Tracking Systems for Integration in Context-Aware Construction Applications	32
3.1.	Introduction	32
3.2.	Position and Head Orientation Tracking Systems	33
3.2.1.	Position Tracking Systems	34
3.2.2.	Head Orientation Tracking Systems	41
3.3.	Eye Motion Tracking Systems	44
3.3.1.	Human Visual Perception	44
3.3.2.	Existing Gaze Tracking Systems and Limitations	47
3.4.	Summary and Conclusions	49
3.5.	References	51
4.	Evaluation of Standard Product Models and Project Databases for Information Access and Retrieval in Construction Applications	55
4.1.	Introduction	55
4.2.	Background	57
4.3.	Building Information Models	58
4.4.	Database Management Systems	67

4.5. Summary and Conclusions	76
4.6. References	78
5. User Tracking in Outdoor and Indoor Construction Environments	83
5.1. Introduction	83
5.2. Technical Overview on Indoor User Position Tracking Technologies	87
5.2.1. WLAN-based User Position Tracking	87
5.2.2. UWB-based User Position Tracking	93
5.2.3. Indoor GPS-based User Position Tracking	96
5.3. Mechanisms of a User Viewpoint Tracking Application	99
5.3.1. WLAN-based Position and Head Orientation Tracking Application	99
5.3.2. UWB-based Position and Head Orientation Tracking Application	103
5.3.3. Indoor GPS-based Position and Head Orientation Tracking Application	105
5.4. Tracking Systems Comparative Summary	107
5.5. Summary and Conclusions	109
5.6. References	111
6. High-Precision User Spatial Context Interpretation	114
6.1. Introduction	114
6.2. Human Vision and Computer Graphics	115
6.2.1. Computer Graphics Based Frustum Technique	116
6.2.2. Computer Graphics Based Intersection Techniques	119

6.2.2.1.	General Collision Detection Techniques	119
6.2.2.2.	Raycasting Technique	122
6.3.	Accurate Contextual Object Identification	125
6.3.1.	Computing Region of Space Visible to User	125
6.3.2.	Identifying Objects Contained in User's Visible Space	136
6.3.3.	Increasing Precision in Identifying Contextual Objects	142
6.4.	Summary and Conclusions	148
6.5.	References	149
7.	Contextual Information Retrieval and Visualization	151
7.1.	Introduction	151
7.2.	Information Retrieval Process from MS Access Project Databases	152
7.2.1.	MVC Model Component: Database Structure	156
7.2.2.	MVC Controller Component of the Information Retrieval Process	159
7.2.3.	MVC View Component: User Interface	162
7.3.	Contextual Information Retrieval from CIS/2 Product Models	165
7.4.	Contextual Information Visualization	171
7.5.	Summary and Conclusions	175
7.6.	References	177
8.	Validation of Location-Aware Information Retrieval Methodology	179
8.1.	Introduction	179
8.2.	Validation of Accuracy in 3D Spatial User Tracking and Precision in Contextual Objects Identification	180
8.2.1.	GPS-Based Outdoor Experiment	181

8.2.2. WLAN-Based Indoor Experiments	183
8.2.3. UWB-Based Indoor Experiments	191
8.2.4. Indoor GPS-Based Indoor Experiment	194
8.3. Validation of Reliability in Contextual Information Retrieval and Visualization	196
8.3.1. Structural Steel Inspection Operation	197
8.3.2. Maze Inspection Operation	199
8.4. Validation of Ability of the Framework to Track and Retrieve Contextual Project Information: Civil Engineering Laboratory Inspection Operation	202
8.5. Summary and Conclusions	212
8.6. References	213
9. Summary and Conclusions	214
9.1. Summary of the Proposed Methodology	214
9.2. Contributions	216
9.3. Future Work	221
Appendices	233

List of Figures

Figure 1.1 – Manual Information Search from Structural Drawings	3
Figure 1.2 – Automated Context-Aware Information Retrieval of Structural Details	3
Figure 1.3 – Current Pipe and Fire Hydrant Inspection Practice	4
Figure 2.1 – Overview of Location-Aware Information Retrieval Framework	22
Figure 2.2 –Key Components of Location-Aware Information Retrieval Methodology	23
Figure 2.3 – Mobile User’s Viewing Frustum	25
Figure 3.1 – Eye Anatomy	45
Figure 3.2 – Fovea Centralis of the Eye	46
Figure 4.1 – A Sample Product Model	59
Figure 4.2 – IFC Product Model Viewed in Different CAD Tools	61
Figure 4.3 – Data Exchange Scenario with CIS/2	65
Figure 4.4 – CIS/2 Product Model of a Building Structural Frame (Left) and CIS/2 Product of Connection Details with Bolts (right)	66
Figure 4.5 – Hierarchical Database Model	68
Figure 4.6 – Network Database Model	69

Figure 4.7 – Sample Database Table for a Building	70
Figure 4.8 – Relational Database Model	71
Figure 5.1– Yaw, Pitch, and Roll Angles	84
Figure 5.2 – Outdoor Hardware Prototype	85
Figure 5.3 – WLAN Technology	88
Figure 5.4 – Fingerprinting Approach	89
Figure 5.5 – Ekahau Tracking System Components	90
Figure 5.6 – Ekahau Deployment and Calibration	91
Figure 5.7 – Sapphire UWB Tracking System	93
Figure 5.8 – Multilateration Approach	95
Figure 5.9 – Indoor GPS Transmitter (left) and Receiver (right)	96
Figure 5.10 – Indoor GPS Tracking System	97
Figure 5.11 – Triangulation Approach	98
Figure 5.12 – Using the Ekahau SDK Environment	100
Figure 5.13 – Pseudo Code of the Client Application <i>SimpleTrack.java</i> using JAVA SDK	101
Figure 5.14 – Pseudo Code for Creating a Pipe between Two Tracking Applications	102
Figure 5.15 – Creating a Pipe between Ekahau-based Position and Head Orientation Tracking Applications	102
Figure 5.16 – a) Output Results from Sapphire HUB (top), b) Pseudo Code to Extract UWB Position Coordinates (bottom)	104
Figure 5.17 – Retrieving Positioning and Orientation Information from UWB Sapphire HUB and Tracker	105

Figure 5.18 – Pseudo Code to Extract Indoor GPS Position Coordinates	106
Figure 5.19 – Retrieving Positioning and Orientation Information from Indoor GPS and Tracker	107
Figure 6.1 – Cone of Vision	116
Figure 6.2 – Computer Graphics Pyramid of Vision	117
Figure 6.3 – Field-of-View Angles	117
Figure 6.4 – Near and Far Plane Distances of Viewing Frustum	118
Figure 6.5 – First Raycasting Technique	122
Figure 6.6 – Flight Simulator Raycasting Scenario	123
Figure 6.7 – Tree Location Raycasting Scenario	124
Figure 6.8 – World and Local/User Coordinate Systems	126
Figure 6.9 – 3D Cartesian Local or User Coordinate System	126
Figure 6.10 – Region of Space Visible to a Mobile Computing User	128
Figure 6.11 – Geometric Relationship between Line of Sight and Visible Space for a) yaw (α) and b) pitch (β) Rotations	129
Figure 6.12 – Coordinates of Frustum’s Near Plane Vertices	130
Figure 6.13 – Coordinates of Frustum’s Far Plane Vertices	130
Figure 6.14 – Roll, Pitch and Yaw Angles	131
Figure 6.15 – Basic 3D Rotation Matrices	132
Figure 6.16 –Geometric Transformation for a) yaw (α), b) pitch (β), and c) roll rotations (γ)	132
Figure 6.17 – Matrix Multiplication of the Three Rotation Matrices	133
Figure 6.18 – Rotation and Translation Transformations	134
Figure 6.19 – Vertex Coordinates of View Frustum	135

Figure 6.20 – Viewing Frustum Graphical Components	136
Figure 6.21 – Geometric Interference Test	137
Figure 6.22 – Bounding Volume Representations	138
Figure 6.23 – OBBs of Building Objects	139
Figure 6.24 – Interference Detection Using RAPID Technique	140
Figure 6.25 – Visible Objects Identification	141
Figure 6.26 – LOS Raycasting Test	144
Figure 6.27 – Prioritizing Objects of Interest with LOS Raycasting	145
Figure 6.28 – LOS Intersection with Several Aligned Objects	146
Figure 6.29 – Prioritizing Objects of Interest with Rays Cast away from the Line of Sight	147
Figure 7.1 – A Sample Database Building Model	153
Figure 7.2 – The Model-View-Controller Architecture.	154
Figure 7.3 – MS Access Database Structure	157
Figure 7.4 – Join Table and Relationship Keys	158
Figure 7.5 – Retrieval Application from a User Point of View	163
Figure 7.6 – User Interface Information Retrieval Application (1)	164
Figure 7.7 – User Interface Information Retrieval Application (2)	165
Figure 7.8 – CIS/2 Product Model of a Building Structural Frame	166
Figure 7.9 – CIS/2 File Structure Example	167
Figure 7.10 – CIS/2-VRML Translator Snapshot (1)	169
Figure 7.11 – CIS/2-VRML Translator Snapshot (2)	170
Figure 7.12 – Hardware Components of UM-AR-GPS-ROVER	172

Figure 7.13 – i-Glasses SVGA Pro video see-through Head Mounted Display (HMD)	173
Figure 7.14 – Hands-Free Laptop Holder	174
Figure 7.15 – Lightweight Displays for Engineering Applications	175
Figure 8.1 – Aerial View of the GGB Outdoor Experiment	181
Figure 8.2 – Snapshots of Simulated Building Inspection	182
Figure 8.3 – GGB Construction Engineering Laboratory (University of Michigan)	183
Figure 8.4–Plan View of the Testing inside the Construction Engineering Laboratory (Room 1340 GGB Building)	184
Figure 8.5 – Ekahau Calibration and Testing inside the Construction Engineering Laboratory (University of Michigan)	185
Figure 8.6 – Virtual Representation for Indoor Tracking of a Mobile User Using WLAN (Construction Laboratory, GGB, University of Michigan)	186
Figure 8.7– Virtual Representation for a Geometric Interference Test in the Construction Laboratory	187
Figure 8.8 – Virtual Representation for Indoor Tracking of a Mobile User Using WLAN (Second Floor of GGB Building, University of Michigan)	187
Figure 8.9 – Aerial View of the Maze at Disaster City	188
Figure 8.10 – Access Points Deployment around the Maze (Disaster City)	189
Figure 8.11 – Ekahau Positioning at the Maze (Disaster City)	190
Figure 8.12– Maze at Nike Site (NIST)	190
Figure 8.13 – Plan View of the UWB Receivers Setup inside the Maze (NIST)	191
Figure 8.14 – 3D Snapshot Views of a UWB Tracked User inside the Maze	192
Figure 8.15 – UWB Tracking System at the Maze (Disaster City)	193
Figure 8.16 – Screen View of a Response Robot Evaluation Exercise (NIST)	193

Figure 8.17 – 3D Snapshots of a UWB Tracked Robot at Different Locations in the Maze	194
Figure 8.18 – Deployment of Laser Transmitters around the Maze	195
Figure 8.19 – Steel Structure on NIST Main Campus	197
Figure 8.20 – UWB Receiver and Reference Tag Setup at the Steel Structure (NIST)	198
Figure 8.21– Deployment of UWB Receivers around the Steel Structure (NIST)	198
Figure 8.22 – Snapshot of Contextual Information Retrieval and Visualization of Identified Steel Elements	199
Figure 8.23 – Columns Location in the Maze (Nike Site, NIST)	200
Figure 8.24 – Snapshots of Objects Interference Detection	201
Figure 8.25 – Snapshot of Contextual Information Retrieval	202
Figure 8.26 – Floor Plan of CEE Structural Engineering Laboratory	203
Figure 8.27 – Ekahau Testbed within the Laboratory	203
Figure 8.28 – Ekahau Calibration Reference Points	204
Figure 8.29 – Snapshot of Simulated Structural Engineering Laboratory Inspection	205
Figure 8.30 – Snapshot of Third Person View Interference Detection (Scenario 1)	206
Figure 8.31 – Snapshots of Interference Detection (First Person Virtual and Real Views)	207
Figure 8.32 – Snapshot of Contextual Information Retrieval	207
Figure 8.33 – Snapshot of Contextual Information Retrieval (CAD Drawing)	208
Figure 8.34 – Snapshot of Contextual Information Retrieval (Cost Estimation)	209
Figure 8.35 – Snapshot of Third Person View of Interference Detection (Scenario 2)	210
Figure 8.36 – 3D Snapshots of a Tracked User at Different Locations in the CEE Laboratory (Third and First Person Views)	211

Figure 9.1 – Inertial Measuring Unit (IMU) mounted to the user's boot	223
Figure 9.2 – Augmented Reality View	224
Figure 9.3 – Building Damage Inspection	226
Figure 9.4 – Bridge Inspection	227
Figure 9.5 – Inventory Control Scenario	229
Figure 9.6 – Hybrid GPS and WLAN-Based Global Position Tracking in Emergency Response	230
Figure 9.7 – Navigation Guidance in First Response	231
Figure 9.8 – Eye Motion Tracking Tool	232
Figure 9.9 – IR-Based Eye Motion Tracking System	232
Figure A.1 - Ekahau Client Properties Dialog's Status Tab	239
Figure A.2 - Ekahau Client Properties Dialog Displaying Positioning Engines Tab	240
Figure A.3 - Positioning Engine Login in Ekahau Manager Startup	241
Figure A.4 - Ekahau Manager Startup Options	242
Figure A.5 - Signal Strengths from Access Points	242
Figure A.6 - Map Properties Box	243
Figure A.7 - Drawing Tracking Rails	244
Figure A.8 - Ekahau Java SDK output	248
Figure A.9 – Daisy Chain Configuration	251
Figure A.10 – Sapphire Hub Configuration and Management Main Menu	252
Figure A.11 - <i>Sapphire</i> Hub Setup Menu	253
Figure A.12 – a) C-Shape Transmitter Configuration (Top), b) Box-Shape Transmitter Configuration (Bottom)	256

Figure A.13 – Transmitter Mounted on a Tripod	257
Figure A.14 – Box Geometry Observations and Scale Bar	259
Figure B.1 – Graphics Framework	262
Figure B.2 – Default Coordinates Systems in OpenGL and OSG	263
Figure B.3 – Typical Scene Graph Structure	265
Figure B.4 – Pseudo Code for Defining the Coordinates of the Viewing Frustum Corner Vertices	267
Figure B.5 – Viewing Frustum Graphical Components	269
Figure B.6 – Viewing Frustum Scene Graph Structure	271
Figure B.7 – Defining Viewing Frustum	272
Figure B.8 – Pseudo Code for Identifying Contextual Objects using Raycasting	275
Figure C.1 – Database Structure	286
Figure C.2 – Information Retrieval Application User Interface	295
Figure C.3 – Schedule and Cost Estimation Information Retrieval	298

List of Tables

Table 5.1 – Comparative Summary of Indoor Positioning Technologies	108
Table C.1 – Item Table	287
Table C.2 – Activity Table	288
Table C.3 – Schedules Table	289
Table C.4 – Specs Table	290
Table C.5 – CostEstimation Table	290
Table C.6 – OrdersTable	291
Table C.7 – ItemToActivityMapping Table	291
Table C.8 – ImgFiles Table	292
Table C.9 – ItemCadModels Table	292
Table C.10 – ActivityToImgFilesMapping Table	293
Table C.11 – ActivityToCadModelsMapping	293

List of Appendices

Appendix A – Guide to Use WLAN-Based Ekahau, UWB and Indoor GPS Technologies	235
Appendix B – Computer Graphics Concepts and Techniques	261
Appendix C – Information Access and Retrieval Processes from Databases	284
Appendix D – Biography	300

Abstract

The spatial expanse and the dynamic and information intensive nature of typical engineering projects, in particular construction projects, require mobile users (engineers, managers, inspectors etc.) to have rapid and real-time access to a wealth of project information. However, during the course of their normal activities, field personnel typically spend a significant amount of time in manually accessing relevant information, thereby amounting to lost productivity and increased costs.

In an effort to remedy this situation, this dissertation takes the first steps and presents research that investigated methods for high-precision identification and automated retrieval of contextual information in mobile construction engineering applications. The primary contribution of the presented work is the design and implementation of a dynamic user-viewpoint tracking scheme in which mobile users' spatial context is defined not only by their position but also by their Three Degree-of-Freedom (3DOF) head orientation (i.e. line of sight). This allows the identification of objects and artifacts visible in a mobile user's field of view with much higher accuracy than was possible by tracking position alone.

Within this scheme, location-aware computing and head orientation sensing technologies were integrated. For outdoor applications, the Global Positioning System (GPS) and a

magnetic orientation tracker were used to track a user's dynamic viewpoint. For indoor applications, this research explored the applicability of wireless technologies, namely Wireless Local Area Networks (WLAN), Ultra-Wide Band (UWB), and Indoor GPS for dynamic user position tracking in situations where GPS is unavailable. The same orientation device was used indoors. Additionally, this research evaluated the capability of interoperable product models and project databases, such as the Computer Integrated Manufacturing for steel structures (CIMsteel) Integration Standards (CIS/2) product model and Microsoft Access databases for automated context-aware information access and retrieval.

The developed methods and the components of the proposed context-sensing information retrieval methodology have been validated through several experiments conducted at the University of Michigan, the National Institute of Standards and Technology (NIST), and the Disaster City urban search and rescue testbed in Texas.

Chapter 1

Introduction

1.1. Background

Due to its nature, the construction industry requires its personnel to be mobile, to communicate efficiently, and to exchange large volumes of information. Evolving technologies such as location-aware computing and building information models offer significant potential of improving important decision-making tasks on construction sites by providing support for tedious and time-consuming tasks associated with timely and accurate access to project information.

For example, rapid and convenient access to contextual project information, through continuous tracking of engineers, managers, and inspectors' spatial characteristics, can lead to significant cost and time savings due to the accuracy and immediacy with which relevant project information can be made available to field personnel (Aziz et al. 2005).

The primary research question that this dissertation addressed was how to design and implement a context-aware methodology capable of rapidly identifying and retrieving contextual information, and providing it to mobile users in challenging engineering environments such as those found on construction sites.

The designed algorithms were implemented in a prototype application, and track mobile users in dynamic environments using ubiquitous positioning and orientation tracking hardware devices, identify specific entities visible to users at a particular time, and automatically retrieve and help users visualize contextual information.

1.2. Problem Statement

During several phases in the life cycle of a constructed facility, from construction to inspection to maintenance, the incapacity of engineers and inspectors to quickly and precisely gain access to relevant information required to make critical decisions in real-time has considerable consequences. Time is valuable, and constructors and inspectors typically work with stacks of paperwork and unwieldy drawings on harsh and dynamic jobsites, and spend a significant amount of effort in identifying, accessing, and retrieving the information necessary for important decision making tasks. This repetitive, mundane, and seemingly innocuous activity consumes a significant amount of time, and thus money (Cox et al. 2002, Bowden et al. 2004 and Skattor et al. 2007).

For example, an inspector must study several detailed drawings to find the size, shape,

and location of a particular set of steel braces to verify their correct installation on site (Figure 1.1). This non-value adding activity can, for instance, be eliminated if the inspector can identify and retrieve the specific information in context (i.e. structural details of the pertinent steel braces) automatically, while being continuously tracked on the site (Figure 1.2) (Han et al. 2007). The on-site automatic context-aware information retrieval process can also be helpful in rapidly notifying the inspector of any significant deviations such as activities behind schedule or materials not meeting the specifications.



Figure 1.1 – Manual Information Search from Structural Drawings

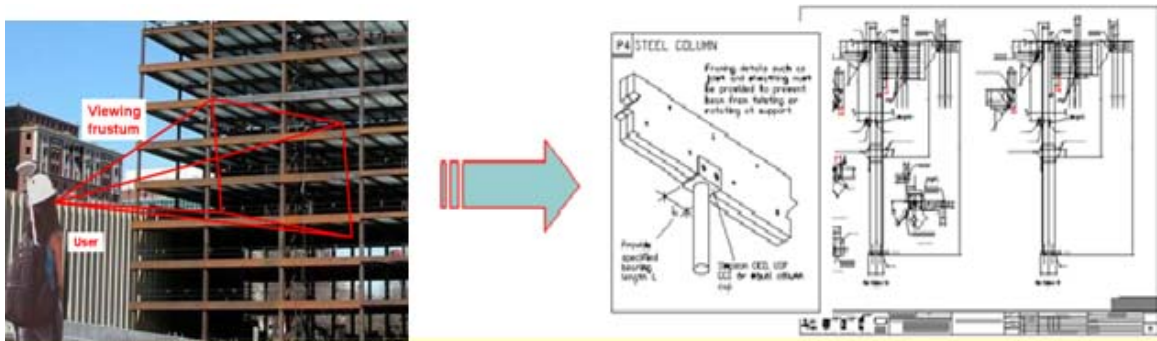


Figure 1.2 – Automated Context-Aware Information Retrieval of Structural Details

Another example can be considered in pipe/valve system inspection and maintenance. Currently, inspectors must perform inspection from a checklist, and then contractors must

conduct pressurized tests and submit charts and technical reports. Finally, inspectors have to verify and record test results and file reports later in the office (Figure 1.3).

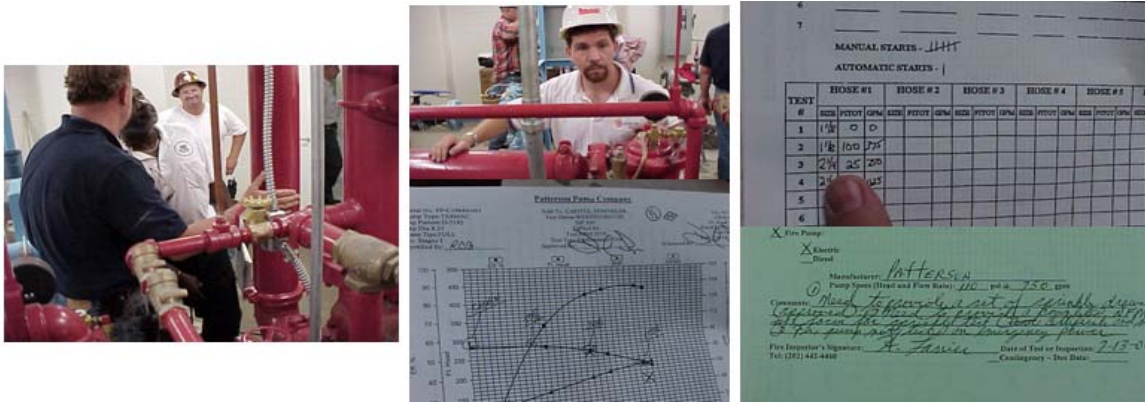


Figure 1.3 – Current Pipe and Fire Hydrant Inspection Practice

This tedious inspection process can, for instance, be eliminated if an integrated context-based flow and archival of information is created among all entities involved in inspection and maintenance in order to rapidly access needed data in real-time.

There are thus clear motivating reasons to investigate a new context-aware methodology that can allow rapid and accurate identification, and retrieval of contextual project information to support decision makers in field construction, inspection, and maintenance tasks.

For the last several years, many research efforts have focused on enabling and advancing information technology such as location-aware computing, by providing a mechanism to deliver pertinent information in order to enhance collaboration and work efficiency. Relevant information can be delivered to any decision maker in real-time by monitoring

the physical and environmental context, and then reconciling the identified context parameters with the available pool of digital information. Such context-aware computing is defined by Burrell et al. (2001) as the use of environmental characteristics such as a user's location, time, identity, profile, and activity to provide information that is relevant to the current context. Context-aware computing can thus potentially enable mobile users to leverage knowledge about various context parameters such as their identity, current task, and location to ensure that they get highly specific information pertinent to the decisions at hand (Schilit et al. 1994).

The relevance of context-awareness for mobile users has been demonstrated in a large number of applications that have been summarized in Aziz et al. (2006). Prior applications of context-aware computing have included fieldwork (Kortuem et al. 1999, Pascoe et al. 1998), museums (Fleck et al. 2002, Lassila et al. 2003), route planning (Marmasse et al. 2002), libraries (Aittola et al. 2003) and tourism (Laukkanen et al. 2002). Examples of other projects that have specifically focused on location based information delivery have included the GUIDE project (Davies et al. 1999), the Mobile Shadow Project (MSP) (Fischmeister et al. 2002), and the Ambience project (Ambience 2004). Previous work on context-aware computing also involved the development of mechanisms for information support on construction sites (Halfawy et al. 2001, Cox et al. 2002, Singhvi et al. 2003, Bowden et al. 2004, May et al. 2005, Aziz et al. 2005, 2006 and Skattor 2007).

The ability to automatically access and retrieve accurate information that is of decision

making context at an arbitrary time and location can significantly increase the productivity of constructors, engineers and inspectors. However, existing context-aware information delivery applications are based on the interpretation of a user's spatial context using position (location) alone. Another major element of the user's spatial context, the three-dimensional orientation of the line-of-sight, is ignored in the computation.

For example, tracking only an engineer's position on a construction site might help determine which floor of a building a mobile user is located on or even which room the user is currently in (Aziz et al. 2005, May et al. 2005). However, this information is not sufficient to conclude which part or section of the room, or what particular component or object in that room the engineer is currently interested in. Therefore, the position as well as the orientation must be considered in the computation to fully interpret a user's spatial context and facilitate the information delivery process.

1.3. Research Objectives and Challenges

In order to address the limitations in the existing state of knowledge, the overall objective of this research was to study the requirements, design, implement, validate, and demonstrate a new spatial context-aware information technology that is capable of providing engineers with accurate prioritized contextual information in real-time for critical decision-making. In order to effectively achieve the identified objective, the research addressed several problems and challenges, and pursued the following tasks:

- Studied the decision-making process of engineers and inspectors in specific scenarios to understand the nature of the decisions that are made and the type and source of information on which those decisions are based at those times. This helped define the requirements of an automated, location-based contextual information access and retrieval methodology.
- Studied the details of involved parameters in tracking a mobile user's spatial context in outdoor and indoor environments. The fully-qualified spatial context of an engineer or inspector can be continually interpreted by tracking not one, but two basic context parameters: 1) the position where the engineer is located; and 2) the direction in which s/he is looking.
- Presented a context-sensing information retrieval methodology capable of accurately and continuously tracking a user's position and orientation at an arbitrary outdoor and/or indoor location, sending this tracking data to the 3D visualization application at each animation frame, reconciling the user's tracked spatial context with available information; and interactively presenting retrieved information to the user.
- Devised methods capable of communicating with position tracking systems and head tracking devices (i.e. magnetic trackers) in real-time to accurately determine the user's location and head orientation in indoor and outdoor environments.
- Designed an algorithm to accurately define, based on tracked position and head orientation, the mobile user's line-of-sight and compute the region of space that is in the field of view at that time.

- Designed a method to overlap the visible region of space with the coordinate system of CAD (and other digital data representing the user's surroundings), and precisely identify objects of contextual interest (i.e. visible to users) at a given time and tracked location.
- Created a 3D scene graph based application to visualize a user's head and body movements (i.e. navigation behavior) in an indoor or outdoor environment, and evaluated the reliability of the tracking data in graphically updating the specific components visible to the user.
- Designed algorithms to query project databases and retrieve information expected to be of relevance to a user at a particular tracked time and location.
- Implemented methods to display retrieved information to mobile users using mobile visualization tools and interaction techniques.
- Validated the designed tracking and retrieval algorithms of the developed context-aware information retrieval framework in several experiments including both outdoor and indoor tests of engineering operations in environments representing conditions such as those found on construction sites.

1.4. Research Contributions

During several phases in the life cycle of a constructed facility and for all types of project information ranging from plans and shop drawings to specifications, schedules, installation guides, and change orders, rapid and convenient access to contextual data is critical. Accurate and effective on-site information retrieval facilitated by the proposed

methodology offers significant potential for higher productivity and faster service in field construction, inspection, and maintenance tasks.

The main contributions of this research therefore primarily lie in: 1) Designing and implementing a dynamic user line-of-sight tracking scheme in which mobile users' spatial context is defined not only by their position (i.e. location), but also by their three-dimensional head orientation (i.e. line-of-sight), 2) Designing algorithms to identify objects and artifacts visible in a mobile user's field of view with much higher accuracy than was possible by tracking position alone, and prioritize relevant identified information, and 3) Implementing methods to provide continuous, context-based contextual information on specific identified components, and interactively visualize retrieved information on mobile devices in arbitrary indoor/outdoor environments.

1.5. Dissertation Outline

This dissertation presents the developed context-aware methodology, the design of ubiquitous tracking algorithms, and the use of interoperable product models and project databases, and interactive visualization tools together with their practical implementation inside a mobile context-sensing visualization scheme.

Each chapter of the dissertation serves as a stand-alone document that describes the major challenges encountered, different choices investigated, details of individual scientific questions answered, and algorithms and methods designed and implemented in

addressing a specific research issue to achieve a particular research objective. Additional implementation details to supplement the information presented in the chapters are provided in several appendices.

In Chapter 2, a detailed literature review on several research efforts conducted in the field of context-aware computing is presented. Additionally, the overall methodology, a context-sensing information retrieval framework with its different mechanisms, is introduced and described. Chapter 3 and Chapter 4 explore several position and orientation tracking systems and project databases for possible implementation within the framework. Chapter 5 describes the details of the adopted tracking technologies investigated in this research to provide real time access to a user's positional and head orientation data and their integration into a 3D visualization tracking application. In Chapter 6, methods to precisely interpret mobile users' spatial context and identify contextual relevant objects using computer graphics are presented. In Chapter 7, methods to access and retrieve contextual information from product models and standard project databases as well as different interactive visualization tools and classes of display devices to present retrieved information to mobile users are presented. Finally, Chapter 8 documents results from several validation experiments that have been conducted to evaluate the functionality and reliability of the individual components of the developed context-sensing information retrieval methodology.

Each chapter of this dissertation has been compiled and written so that it is easily understandable by a wide audience ranging from individuals with prior experience in

wireless positioning technologies, data retrieval, and computer graphics, to those with only basic knowledge about construction and other engineering operations and projects. Each chapter concludes with an extensive list of references (books, papers, and internet URLs) that direct the reader to further resources related to the main topic of the chapter.

1.6. References

Aittola, M., Ryhänen, T., and Ojala, T. (2003). "SmartLibrary—Location-Aware Mobile Library Service", In Proceedings of the Fifth International Symposium on Human Computer Interaction with Mobile Devices and Services, Udine, Italy, 411–416.

AMBIENCE Project - available at:

<http://www.extra.research.philips.com/euprojects/ambience/> (Accessed: June 30, 2006).

Aziz, Z., Anumba, C.J., Ruikar, D., Carrillo, P.M., Bouchlaghem, D.N. (2005). "Context Aware Information Delivery for on-Site Construction Operations", 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universität Dresden, Germany, CBI Publication No:304, 321-32.

Aziz Z, Anumba CJ, Ruikar D, Carrillo P, Bouchlaghem D.N, (2006). "Intelligent wireless web services for construction – A review of enabling technologies", Automation in Construction, Elsevier, 15(2), 113–23.

Behzadan, A.H., and Kamat, V.R. (2007). "Georeferenced Registration of Construction Graphics in Mobile Outdoor Augmented Reality", ASCE Journal of Computing in Civil Engineering, 21(4), Reston, VA, 247-258.

Behzadan, A.H., and Kamat, V.R. (2005). "Visualization of Construction Graphics in Outdoor Augmented Reality", In Proceedings of the 37th Winter Simulation Conference (WSC), IEEE, Piscataway, NJ, 1914-1920.

Bowden, S.L., Dorr, A., Thorpe, A., and Anumba, C.J. (2004). "Mapping Site Processes for the Introduction of Mobile IT", In Proceedings of the ECPPM eWork and eBusiness in Architecture, Engineering and Construction, AA Balkeman Publishers, 491-498.

Burrell, J., and Gay, K. (2001). "Collectively Defining Context in a Mobile, Networked Computing Environment", In Proceedings of the Conference on Human Factors in Computing Systems, Association for Computing Machinery (ACM), New York, NY, 231-232.

Cox S., Perdomo, J., Thabet, W. (2002). "Construction Field Data Inspection Using Pocket PC Technology", In Proceedings of the International Council for Research and Innovation in Building and Construction, Aarhus, Denmark, 243-251.

Davies, N., Cheverst, K., Mitchell, K., and Friday, A. (1999). "Caches in the Air: Disseminating Information in the Guide System", In Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), IEEE, New Orleans, Louisiana, 11–19.

Ekahau, Wi-Fi Based Real-time Tracking and Site Survey Solutions – available at: <http://www.ekahau.com>. (Accessed: February 20, 2007).

Fischmeister, S., Menkhaus, G., and Pree, W. (2002). “MUSA-Shadows: Concepts, Implementation, and Sample Applications; a Location-Based Service Supporting Multiple Devices”, In Proceedings of the Fortieth International Conference on Technology of Object-Oriented Languages and Systems (TOOLS), Sydney, Australia., 71–79.

Fleck, M., Frid, M., Kindberg, T., O'Brien-Strain, E., Rajani, R., and Spasojevic, M. (2002). “From Informing to Remembering: Deploying Ubiquitous Systems in Interactive Museums”, IEEE Pervasive Computing, 1(2), 13–21.

Han, S., Lee, S., Golparvar Fard, M., and Peña-Mora, F. (2007), “Modeling and Representation of Non-Value Adding Activities due to Errors and Changes in Design and Construction Projects.” In Proceedings of the 2007 Winter Simulation Conference, Institute of Electrical and Electronics Engineers (IEEE), Washington, D.C, 2082-2089.

Kamat V.R. and Behzadan A.H. (2006). “GPS and 3DOF Angular Tracking for Georeferenced Registration of Construction Graphics in Outdoor Augmented Reality”, In Lecture Notes in Computer Science (Intelligent Computing in Engineering and Architecture), Ian F.C. Smith (Editor), Springer, New York, NY, 368-375.

Kortuem, G., Bauer M., Heiber, T., and Segall, Z. (1999). “NETMAN: the Design of a Collaborative Wearable Computer System.” ACM/Baltzer Journal on Mobile Networks and Applications (MONET), 4(1), Springer, Netherlands, 49-58.

Laukkanen, M., Helin, H., and Laamanen, H. (2002). ”Tourists on the Move”, In Cooperative Information Agents VI - 6th International Workshop, Lecture Notes in Computer Science, Springer, Germany, 36–50.

Lipman, R., and Reed, K. (2000). “Using VRML in Construction Industry Applications”, In Proceedings of the Fifth Symposium on Virtual Reality Modeling Language (Web3D-VRML), Association for Computing Machinery (ACM), New York, NY, 119-124.

May, A., Mitchell, V., Bowden, S., and Thorpe, T. (2005). “Opportunities and Challenges for Location Aware Computing in the Construction Industry”, In Proceedings of the MobileHCI'05, Austria. ACM Press, 255-258.

Metris Products Website, IGPS- Large Scale Metrology – available at: http://www.metris.com/large_volume_tracking_positioning/igps/ (accessed: September 30, 2007).

Multispectral Solutions, Inc. Website, Sapphire DART System- available at: <http://www.multispectral.com/products/sapphire.htm> (Accessed May 31, 2007)

Pascoe, J., Morse, D.R., and Ryan, N.S. (1998). "Developing Personal Technology for the Field", *Journal of Personal and Ubiquitous Computing*, 2(1), Springer, London, 28-36.

Schilit, B.N., Adams, N., and Want R. (1994). "Context-Aware Computing Applications", *Workshop on Mobile Computing Systems and Applications (WMCSA)*, Santa Cruz, CA, 85-90.

Singhvi, V., Fish, W., and Terk, M. (2003). "Context-Aware Information System for Construction Site Applications", In *Proceedings of the ASCE Construction Research Congress- Winds of Change: Integration and Innovation in Construction*, Honolulu, Hawaii, 981-988.

Skattor, B. (2007). "Design of Mobile Services Supporting Knowledge Processes on Building Sites", In *Proceedings of the World Congress on the Management of eBusiness (WCMeB)*, 10-17.

Chapter 2

Context-Aware Information Retrieval Methodology: Technical Overview

2.1. Introduction

In an information-intensive industry such as construction, rapid and convenient access to project information is crucial in helping engineers, inspectors, and maintenance personnel make critical, real-time decisions. Field personnel typically spend a significant amount of time in manually accessing the information they need for important decision making tasks. Such lost time amounts to lost productivity and thus money (Aziz et al. 2005).

This chapter introduces the state of the art, and describes prior work in context-aware computing technologies. It then presents a technical overview on the proposed context-aware information retrieval framework that enables real-time tracking of mobile users'

fully-qualified spatial context, identification of objects in users' field of view, and automated retrieval and visualization of contextual information in dynamic outdoor and indoor environments such as those found on construction sites.

2.2. Literature Review

Recently, several approaches providing support for context-aware applications have been proposed. Context-awareness (Dey 2001) is interesting in mobile scenarios where the context of the application is highly dynamic, and allows the application to deal with the constraints of mobile devices in terms of network connections and mobile users' characteristics.

Prior applications of context-aware computing have included fieldwork (Kortuem et al. 1999) where a wearable groupware system was designed to enhance the communication and cooperation of highly mobile network technicians in the field with the capabilities for real-time audio-conferencing, transmission of video images back to the office, and context-sensitive access to a shared notebook. In other fieldwork applications (Pascoe et al. 1998), personal computing aids (i.e. hand-held computing devices and stick-e note technology) were developed to help mobile field workers in their tasks.

Context-aware computing has also been applied in museums such as the Exploratorium, an interactive science museum in San Francisco (Fleck et al. 2002). For instance, computing tools (i.e. electronic guidebook and "rememberer" tool) were implemented

and tested to allow users to bookmark exhibits and take pictures of their experiences at the exhibits. Other museum computing applications enabled the utilization by other agents of ubiquitous computing devices, together with semantic web-based techniques, without human guidance or intervention (Lassila et al. 2003). A similar concept has also been applied for tourism (Laukkanen et al. 2002).

Other applications have involved the design of location-aware mobile services for use in libraries, namely SmartLibrary software-based solution (Aittola et al. 2003). This service helped users in searching for books in the main library of the University of Oulu by providing a map-based guidance to books and collections on a PDA.

Another application (Dey et al. 1999) included the design and implementation of a context-aware architectural solution (Conference Assistant) to assist conference attendees. In this solution, context came from many, distributed machines, so-called widgets read sensors. When the user is attending the conference, the application first uses information about what is being presented at the conference and the user's personal interests to determine what presentations might be of particular interest. The application uses the conference attendee's location, the activity (presentation of a Web page or slide) in that location and the presentation details to determine what information to present. After the conference, the retrieval application uses the conference context to retrieve information about the conference (i.e. time when presentations started and stopped, names of the presenters and the presentations, etc.).

Additional case studies involved building an intelligent information system (HARVEST) to aid users in their investigative tasks, such as detecting fraud (Wen et al. 2007). This system offered two unique features that assisted users in their information seeking. First, it worked with users cooperatively to help build a rich investigative context over the course of an investigation. Second, it adaptively selected and evaluated information based on the evolving investigative context. In addition, it externalized the investigative context, which allowed users to track their information needs and directly update such needs if desired.

Other prior applications included designing an infrastructure for context-aware information retrieval to enhance situation awareness in military applications (Bahrami et al. 2007). For military operators, the awareness of the temporal unfolding of events in a situation, together with their understanding of the implications of those events for both ongoing activities and future plans, is clearly a critical element in the effectiveness and adaptability of a military force. This infrastructure provided each user with a customized, mission-oriented system that gives access to the right information from heterogeneous sources in the context of a particular task, plan and/or mission.

The BMW ConnectedDrive project (Bachmann et al. 2000) is another context-aware application. It combined contextual information of an automotive framework according to the three autonomous fields, the vehicle environment, its current driving state parameters, and driver information, interpreted as the vertices of a so-called contextual triangle. In this framework, environment information is acquired by sensors analyzing the

traffic in the front, the side and the back of the car, including technologies like radar or lidar. The vehicle context describes the state of the car, such as its velocity and acceleration, as well as system status information. The role and the influence of the driver on the situation, implies detailed information on his/her point of gaze and interactions with the vehicle.

In the ConnectedDrive system, the driver information is one missing aspect of the contextual triangle. Therefore a new project has been recently created, the BMW SURF project (Hoch et al. 2007). The central idea of this project has been to build a car, which is able to cover all parts of the context triangle and bring these information sources together to create exemplary intelligent vehicle behavior.

Additional related studies have included GeoNotes (Espinoza et al. 2001), and CampusSpace (Ferscha et al. 2001). GeoNotes is a system for abstracting location information for location-aware applications. The system architecture was constructed to support shared information for mobile devices, in particular to leave notes at specific places for other users to be read. A user creates notes and sticks it to certain places, where other users can read them. GeoNotes considered context in a very limited way, and focused on location context only. CampusSpace is a multi-user team awareness framework developed to gather the geographical position using WLAN access points. The idea was based on using signal strength and signal quality to determine spatial proximity either to an access point or another device also equipped with WLAN.

Other examples of projects that have specifically focused on context-based information delivery have included the GUIDE project (Davies et al. 1999) and the Mobile Shadow Project (MSP) (Fischmeister et al. 2002). In the GUIDE project, a context-sensitive tourist guide was developed for visitors to the city of Lancaster. Visitors were equipped with portable GUIDE units which provided interactive services and dynamically tailored Web based information using a city-wide wireless network infrastructure. In the MSP project, the authors used agents to map the physical context to the virtual context. In another project named Ambience (Ambience 2004), a different approach was used wherein the focus was on creating a digital environment that is aware of users' presence, context, and sensitivity, and responds accordingly.

Previous work has involved the use of different wireless technologies on construction sites. For instance, Aziz et al. (2006) developed a prototype application for context-aware information delivery that takes advantage of Appear Networks. Appear makes use of wireless local area networking (WLAN) and employs a layer of context-aware intelligence to proactively provide mobile workers with the relevant tools and mission-critical information they need to work faster and more effectively. Skibniewski and Jang (2006) explored the use of ZigBee networks to be used in civil infrastructure systems. A prototype application was developed for object tracking and monitoring on construction sites to provide an insight into industrial practices in sensor and network-based ubiquitous computing. Teizer et al. (2008) also showed how the Ultra-Wide Band (UWB) wireless sensing technology is capable of determining three-dimensional resource location information in object cluttered construction environments.

Each of the abovementioned research efforts, however, determined the position of a mobile user or physical asset with relatively low accuracy. The uncertainty of the tracked position was not low enough to accurately identify the spatial context with fine granularity. Additionally, all prior work described above focused on developing tracking applications where the spatial context is defined solely by the location (position) of the user or physical asset. Another major attribute, the three-dimensional orientation, is ignored in the computations. This leads to an imprecise and incomplete interpretation of a user's spatial context. A user's 3D head orientation fully describes the user's line of sight (i.e. the direction in which the user is looking) and therefore can interpret a user's spatial context with much greater precision than is possible with position alone.

For these reasons, prior context-aware information retrieval applications do not consider identification of specific objects visible to a mobile user within the realm of a room or section of a building. Instead, they detect and identify other generic information, for example the overall location of the user in the building. As such, none of the prior work described above proposes any methods to automatically retrieve and visualize contextual information on identified visible components in a studied area.

2.3. Technical Approach

This dissertation introduces a dynamic user context-sensing framework that accurately tracks a mobile field user's fully-qualified spatial context, and automatically retrieves prioritized project information of interest to that user, based on the sensed context.

Figure 2.1 summarizes the mechanics of the proposed framework.

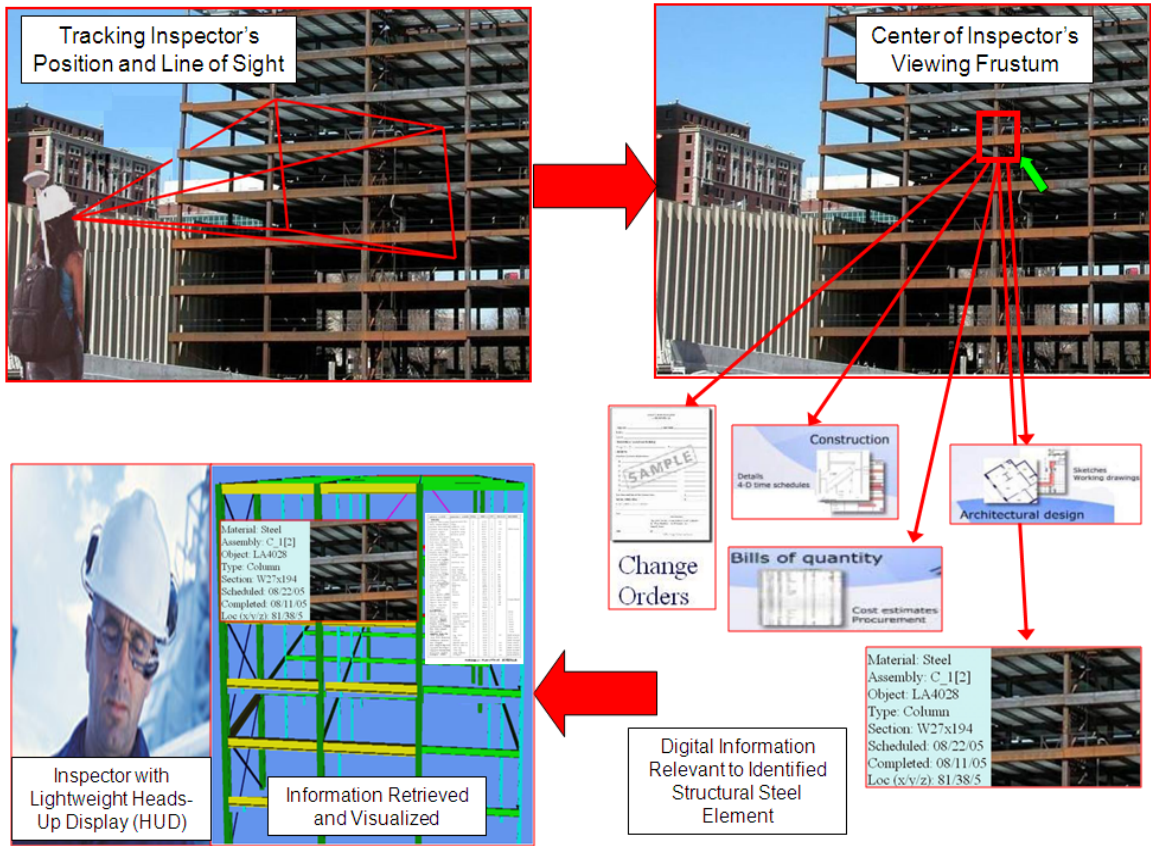


Figure 2.1 – Overview of Location-Aware Information Retrieval Framework

The developed framework consists of the following key components (Figure 2.2) : 1) methods to accurately and continuously track a mobile user’s spatial parameters (i.e. position and orientation) in arbitrary environments; 2) algorithms to interpret a user’s fully-qualified spatial context based on the tracked spatial parameters; 3) algorithms to reconcile a user’s interpreted spatial context with the available pool of project information; and 4) methods to retrieve and interactively present the prioritized project information to the user.

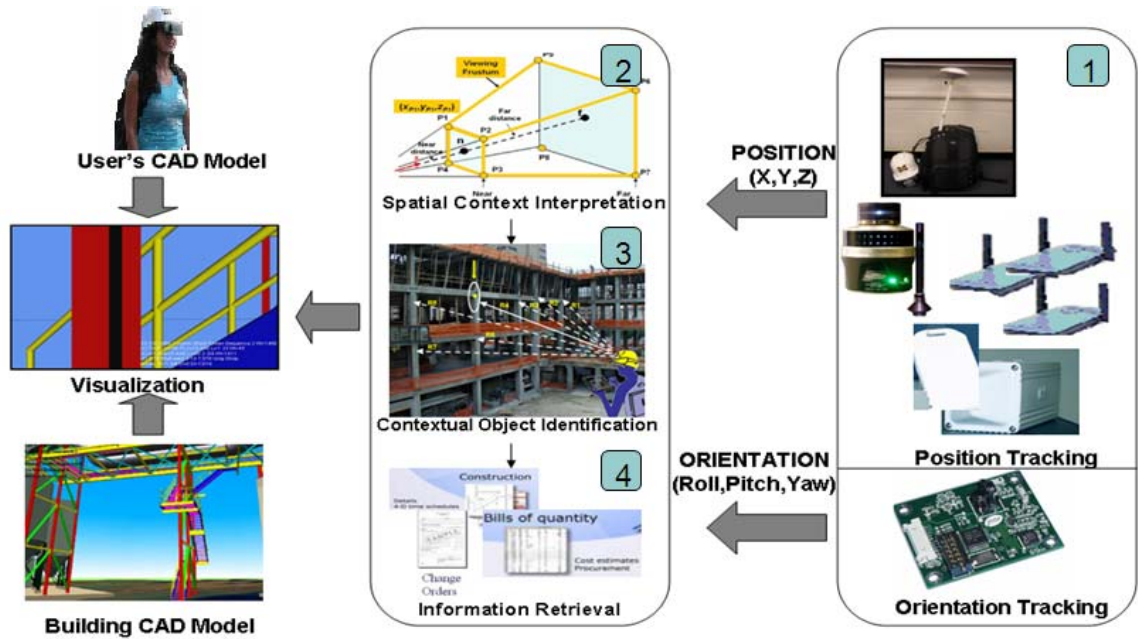


Figure 2.2 –Key Components of Location-Aware Information Retrieval Methodology

All four components are briefly presented in the following subsections and then individually described in detail in Chapters 5 through 7 and Appendices A through C.

2.3.1. Tracking a User’s Spatial Parameters

The first framework component involves accurately tracking a user’s 3D position and orientation in any indoor and/or outdoor environment. The designed tracking scheme encompasses outdoor positioning technologies and indoor wireless tracking technologies, as well as 3DOF magnetic orientation tracking devices.

For outdoor environments, a georeferencing based algorithm previously developed using the Global Positioning System (GPS) and magnetic orientation tracking devices

(Behzadan and Kamat 2005, 2006, and 2007) is used to continuously track a user's dynamic viewpoint. GPS is an attractive option because it does not rely on any pre-installed sensor infrastructure and instead depends on direct satellite communication. However, GPS is unsuitable in situations where the user has a possibility of being indoors because the receivers need a clear line-of-sight to the satellites in order to track position. Therefore, for indoor environments, several wireless technologies such as Wireless Local Area Networks (WLAN) (Ekahau 2004), Ultra-Wide Band (UWB) (Multispectral Solutions 2008), and Indoor GPS (iGPS) (Arc Second 2004) have been investigated and integrated within the designed framework, together with the same set of orientation devices used in outdoor applications.

2.3.2. User Spatial Context Interpretation

By knowing the user's position and orientation that is tracked by the first framework component (tracking module), the user's line-of-sight can be accurately defined and the region of space that is in the field of view at that time can be computed. The region of real space visible to a mobile computing user can be conceptually thought of to be similar to an avatar's viewpoint in a computer graphics application or virtual reality world. In a computer graphics world (e.g. visual simulation), the region of visible virtual space is called the viewing frustum or view frustum, and is typically shaped as a frustum of a rectangular pyramid (Woo et al. 1997). This is graphically shown in Figure 2.3.

Based on the concept of the viewing frustum, the author mathematically derived the formulation for the region of space visible to a mobile computing user. Additional details of this computation are provided in Chapter 6 and Appendix B.

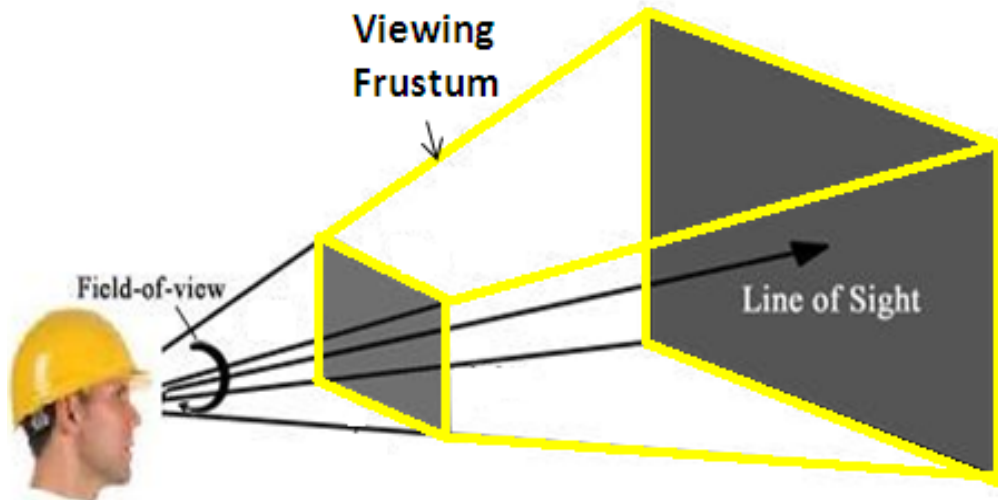


Figure 2.3 – Mobile User's Viewing Frustum

In order to interpret which objects in the user's surrounding environment are currently in context, the computed view frustum is represented as a geometric model (i.e. CAD object) on the computer. Then the coordinate system used to track the user's navigation is reconciled with the coordinate system used to model the user's surroundings in CAD. A CAD design typically uses a local coordinate system without reference to any particular geo-referenced global coordinate system.

The viewing frustum model is then tested for geometric interference with CAD representations of objects in the user's surroundings. The goal of intersection detection in computer graphics is to report any contacts between geometric objects when they occur

(Lin and Gottschalk 1998). In this context, the interference detection enables the determination of whether a corresponding real object is indeed in the user's view at that particular time.

2.3.3. Relevant Contextual Objects Identification

Using the method described above, several objects can be detected in the user's view frustum at any given time. However, the user might be specifically interested in only one or few of the identified visible objects at that time. Such precision in interpreting which specific visible object(s) the user is interested in is achieved by adopting another geometric interference analysis technique known as raycasting (Foley 1990). Further details on the third component of the proposed framework and the raycasting technique in particular are presented in Chapter 6 and Appendix B.

2.3.4. Contextual Information Retrieval and Visualization

Once the specific objects of interest are identified using the algorithms described above, the captured spatial contextual information can be mapped to available data and services (Anumba et al. 2003). In this research, project databases and interoperable product models were used to store, access, and retrieve project information (textual and graphical). For example, the utility of the interoperable CIMsteel Integration Standards (CIS/2) product model was investigated and used. CIS/2 is a logical product model for structural steel building information (Lipman and Reed 2003). CIS/2 has been

implemented by many steel design, analysis, engineering, fabrication, and construction software packages to create a seamless and integrated flow and archival of information among all entities involved in the design and construction of steel framed structures.

For these reasons, the CIS/2 standard model presents a suitable data structure to represent cross-referenced building data for the evaluation of the developed automated information retrieval technique. In addition to CIS/2 product models, other project repositories such as MS Access databases were also deployed on mobile computers as detailed in the validation chapter (Chapter 8). Additional details on other investigated databases and information retrieval algorithms are provided in Chapter 4, Chapter 7, and Appendix C.

In order to interactively display information retrieved using the described methodology to the mobile user, a new class of display and interaction devices are needed. It is important that any information support provided to the user must not come at a safety and efficiency cost. For instance, any display technologies used must unobtrusively present information in a site engineer's or inspector's view without distracting the user from the core task at hand. Several types of mobile computing devices have been investigated and used in this research. Examples are presented in Chapter 7 of the dissertation.

2.4. Summary and Conclusions

This chapter presented first an overview of the state of the art and available context-aware computing technologies together with their limitations. It then described the

overall architecture of a spatial context-aware methodology for automatically tracking, identifying and retrieving, contextual project information to support decision makers such as engineers, inspectors, and maintenance crews on construction project sites.

Detailed information on the different components of the proposed methodology is provided in Chapter 5 through 7 and Appendices A through C. The author has described methods to calculate the region of space visible to a mobile user based on the tracked location and head orientation, prioritize the context of visible objects in the user's view using a raycasting and intersection approach, and retrieve and visualize the prioritized contextual information. Then to demonstrate the ability of the proposed methodology, several validation tests have been conducted in outdoor and indoor environments (Chapter 8).

It is important to note that the proposed methodology is presented as a feasibility study and is independent of any specific technology. With the advance of time, better and more accurate tools can be used and integrated.

2.5. References

- Anumba C. J., Ruikar D., Aziz Z., Carrillo P. M., and Bouchlaghem D. N. (2003). "Towards a Web of Construction Knowledge and Services", In Lecture Notes in Computer Science (ISWC), Springer, Berlin, Germany, 319-334.
- Aziz Z, Anumba CJ, Ruikar D, Carrillo P, Bouchlaghem D.N, (2006). "Intelligent wireless web services for construction – A review of enabling technologies", Journal of Automation in Construction, Elsevier, 15(2),113–23.
- Aziz, Z., Anumba, C.J., Ruikar, D., Carrillo, P.M., Bouchlaghem., D.N. (2005). "Context aware information delivery for on-Site construction operations", Proceedings of the 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universitat Dresden, Germany, CBI Publication No:304, 321-32.
- Bahrami, A., Yuan, J., Smart, P. R. and Shadbolt, N. R. (2007)." Context-Aware Information Retrieval for Enhanced Situation Awareness", In Proceedings of the Military Communications Conference (MILCOM), IEEE, Orlando, Florida, 1-6.
- Behzadan A.H., and Kamat V.R. (2007). "Georeferenced Registration of Construction Graphics in Mobile Outdoor Augmented Reality", ASCE Journal of Computing in Civil Engineering, 21(4), Reston, VA, 247-258.
- Behzadan A.H., and Kamat V.R. (2005). "Visualization of Construction Graphics in Outdoor Augmented Reality", In Proceedings of the 37th Winter Simulation Conference (WSC), IEEE, Piscataway, NJ, 1914-1920.
- Bachmann, T., Naab,K., Reichart, G., and Schraut, M.(2000). " Enhancing Traffic Safety With BMW's Driver Assistance Approach Connected Drive", In Proceedings of the Seventh World Congress on Intelligent Transportation Systems Conference, Turin, no. 2124.
- Burrell, J., and Gay, K. (2001). "Collectively defining context in a mobile, networked computing environment", In Proceedings of the Conference on Human Factors in Computing Systems, Association for Computing Machinery (ACM), New York, NY, 231-232.
- Dey, A. (2001). " Understanding and Using Context", Journal of Personal and Ubiquitous Computing, 5(1), Springer, London, 4-7.

Dey, A., Salber, S., Futakawa, M., and Abowd, G. (1999). "The Conference Assistant: combining context-awareness with wearable computing", In Proceedings of the Third IEEE International Symposium on Wearable Computers, San Francisco, CA, 21-28.

Ekahau, Wi-Fi Based Real-time Tracking and Site Survey Solutions – available at: <http://www.ekahau.com>. (Accessed: February 20, 2007).

Espinoza, F., Persson, P., Sandin, A., Nyström, H., Cacciatore, E., and Bylund, M. (2001). "GeoNotes: Social and Navigational Aspects of Location-Based Information Systems", Ubicomp 2001, In Proceedings of Ubiquitous Computing (UbiComp), Springer, Berlin, 2-17.

Ferscha, A., Beer, W., Narzt, W. (2001). "Location Awareness in Community Wireless LANs", In Kurt Bauknecht, Wilfried Brauer, and Thomas A. Mück (eds.), Proceedings of the GI Jahrestagung, Vienna, Austria, 190-195.

Foley, J.D. (1990). Computer Graphics. Theory and Practice. Addison Wesley.

Hoch, S., Schweigert, M., Althoff, F., and Rigoll, G. (2007). "The BMW SURF Project: A Contribution to the Research on Cognitive Vehicles", In Proceedings of the IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, 692-697.

Kamat V.R., and Behzadan A.H. (2006). "GPS and 3DOF Angular Tracking for Georeferenced Registration of Construction Graphics in Outdoor Augmented Reality", In Lecture Notes in Computer Science (Intelligent Computing in Engineering and Architecture), Ian F.C. Smith (Editor), Springer, New York, NY, 368-375.

Lin, M., and Gottschalk, S. (1998). "Collision Detection Between Geometric Models: A Survey", Proceedings of the IMA Conference on Mathematics of Surfaces - available at: <http://www.cs.unc.edu/~dm/collision.html> (Accessed: December 20, 2006).

Lipman, R., and Reed, K. (2000). "Using VRML in Construction Industry Applications", Proceedings of the Fifth Symposium on Virtual Reality Modeling Language (Web3D-VRML), Association for Computing Machinery (ACM), New York, NY, 119-124.

Metris Products Website, IGPS- Large Scale Metrology – available at: http://www.metris.com/large_volume_tracking_positioning/igps/ (Accessed June 30, 2007).

Pateli, A., Giaglis, G.M, Fouskas, K., Kourouthanassis P. and Tsamakos A. (2002). "On the Potential Use of Mobile Positioning Technologies in Indoor Environments", In the Proceedings of 15th Bled Electronic Commerce Conference -e-Reality: Constructing the e-Economy, Bled, Slovenia, 413-429.

Schilit, B.N., Adams, N., and Want R. (1994). "Context-aware computing applications" Workshop on Mobile Computing Systems and Applications (WMCSA), Santa Cruz, CA, 85–90.

Wen, Z., Zhou, M. and Aggarwal, V. (2007). "Context-aware, adaptive information retrieval for investigative tasks". In Proceedings of the IUI'07, 121–131.

Woo, M., Neider, J., and Davis, T. (1997). OpenGL Programming Guide, 2nd .Ed. Addison-Wesley, Reading, MA.

Xybernaut Website, Product and Services- available at: <http://www.xybernaut.com> (Accessed November, 10 2006).

Chapter 3

Evaluation of Tracking Systems for Integration in Context-Aware Construction Applications

3.1. Introduction

During the course of their normal activities, constructors, engineers, inspectors, and maintenance crews typically work with stacks of paperwork and drawings on dynamic jobsites, and spend a significant amount of time in accessing and retrieving relevant information (Skattor 2007).

Tracking field personnel's movement and head orientation is essential for rapidly and conveniently providing mobile and information-based application services on construction sites. In addition to position and head orientation tracking, monitoring eye motion or gaze tracking also offer several possibilities to refine a user's spatial context.

However, due to the relatively nascent stages of development they are in, these latter technologies were not considered in the implementation phase of this research.

This chapter reviews the latest advances in position, head, and gaze tracking technologies and examines their applicability in context-aware construction applications. Several tracking systems were found to offer a practical solution for use in the proposed methodology (Chapter 2).

3.2. Position and Head Orientation Tracking Systems

In early years, the human-computer interface was dominated by the prevalence of so-called standard peripheral input devices such as keyboards, mice, trackballs, light pens and two/three-axis joysticks in addition to some other forms of input system, such as the graphics tablet, touch screen or, in some cases, speech recognition (Stone 1996).

However, a recent surge of interest in using computers for real-time applications which require using input devices in six degrees of freedom (DOF), namely the Cartesian translation in x, y and z, and Euler orientation in roll, pitch and yaw, has manifested the inadequacies of using traditional computer peripherals (Hongu et al. 2003). Examples of applications requiring accurate six-DOF input include robotics (Cleary and Brooks 1993), computer-aided design (CAD) (Barbič and James 2007), biomechanic analysis for the instrumentation of the human body (Aranov et al. 2004), military applications, such as in-cockpit head tracking (Lu et al. 2002), digital antique cataloguing, and archaeological site

surveys, for the rapid production of digital records of everything from temporarily uncovered sites to preserved bodies.

In the field of construction specifically, all prior applications have been based on exclusively using location-aware tracking systems which only provide the position of mobile users (Aziz et al. 2005). The three dimensional head orientation has been ignored in the computations. In order to automatically provide constructors, engineers, inspectors, and other field users with accurate prioritized contextual information for critical decision-making, six-DOF tracking details obtained from both position and head orientation tracking systems are needed.

The overall objective of the research described in this section was to take the first steps in investigating and evaluating different commercially available position and head orientation tracking systems for possible integration in the newly proposed spatial context-aware information delivery methodology (Chapter 2).

3.2.1. Position Tracking Systems

Mobile computing has evolved over the last several years and has aimed at providing mobile users ubiquitous access to the right information at the right time. The position (i.e. location) of users is an important component of mobile computing that can assist users with their desired goals, and make the workplace more intelligent. One of the most popular research areas in pervasive computing is the development of location-aware

systems. Location-aware techniques, also called positioning technologies are systems in which computing devices provide the users with specific information depending on their location, and enable the design of applications that have the capability to identify a user's location and modify their settings, interfaces, and functionality accordingly (Pateli et al. 2002).

In this research, different location-aware technologies have been evaluated to check their applicability in context-aware applications. For instance, the concept of context-aware information delivery (Aziz et al. 2005), on which this whole research is based, encompasses the creation of a user centered mobile dynamic indoor and outdoor work environment, which has the ability to deliver relevant information to on-site mobile users by intelligent interpretation of their spatial characteristics so that they can take more informed decisions (Schilit et al. 1994).

Global Positioning System (GPS), being a satellite-based navigation system, works very well outdoors but lacks support indoors and in congested areas (El-Rabbany 2002). In addition, unlike outdoor areas, the indoor environment imposes different challenges on location discovery due to the dense multipath effect and building material dependent propagation effect (Xiang et al. 2004).

In contrast to the outdoor positioning technologies that are capable of identifying the location of an object or person in open areas, indoor positioning technologies typically set the constraint of a limited coverage range, such as a building or other confined spatial

area. These technologies are therefore not dependent on any external network. They are dependent on a set of technologies used for transmitting wireless data in closed environments.

The first indoor positioning systems that were developed used infrared sensors (Tseng et al. 2001). Infrared (IR) systems (Aitenbichler and Muhlhauser 2003), are known for their capability of not being able to penetrate walls or other opaque materials. A computing device with an infrared receiver uses these signals to determine its current position. Location of tagged devices determines where receivers should be placed. These devices emit IR light, and if the tagged device is in the same room as a receiver, its position is known. However, for these systems to be effectively used there must be receivers, connected using special wiring, in every room where an asset might be located, which is time-consuming and expensive.

Second, IR-based systems fail to function if the IR signal gets blocked. IR-based location systems are subject to restrictions, such as line of- sight limitations or poor performance with fluorescent lighting or in direct sunlight. Therefore, as intervening objects can easily block infrared signals, IR systems were not employed in this research.

A closest competitive technology to IR systems is Bluetooth (Kodde 2005). Unlike infrared, the line of sight it provides can penetrate through walls or obstacles. Bluetooth, providing ranges of up to 100 meters, is also low power and low processing with an overhead protocol, which makes it ideal for integration into small battery powered

devices. However, Bluetooth positioning technology poses some downsides and problems such as the data rate and security. It only offers data rates of 1 MBps, which provides low rates for data transfer. For this very reason, IR systems are considered by many to be the complimentary technology to that of Bluetooth. The greater range and radio frequency (RF) of Bluetooth make it much more open to interception and attack. On the other hand, Bluetooth still remains the best for short range wireless technology but it lacks efficiency in data transfer and for long range applications, and therefore was not suitable in this research. Radio-based positioning has emerged as a more attractive alternative.

A radio-based technology used for identifying and tracking objects within a few square-meters is Radio Frequency Identification (RFID). An RFID (Ayre 2004) system integrates an antenna with electronic circuitry to form a transponder that, when polled by a remote interrogator, will echo back an identification number. There are two types of RFID systems: active and passive. In a passive RFID system an antenna transmits a radio signal to the RFID tag, which then disturbs the signal in an identified expected way and returns the modified signal to the radiating antenna. The biggest advantages of passive RFID systems are the low cost and long useful life of tags. The biggest disadvantage is that their read distance is very limited; tag can be read at very short distances. On the other hand, active RFID tags have batteries and transmit data either at a regular rate or when activated by other transmitters. They have the advantage of being able to transmit longer distances with smaller antennae, but aren't true location solutions since the distances are still typically only 2-3 meters (Aksoy et al. 2004).

Other researchers such as Skibniewski and Jang (2006) explored the use of ZigBee wireless networks for use in civil infrastructure systems. A prototype application was developed for object tracking and monitoring on construction sites in order to provide insights on industrial practices in sensor and network-based ubiquitous computing. The Zigbee protocol is less complex than Bluetooth, has superior power management (2 AA batteries can have ZigBee module last over years), supports many more nodes per network, has lower latency, lets devices join the network more quickly, and wakes up in milliseconds instead of seconds.

Tiny ZigBee sensors can be used to replace RFID tags, which transmit data. However, the ZigBee protocol is designed to operate over a radio defined by the IEEE 802.15.4 standard for the physical and data link protocol layers. As such, ZigBee also inherits the disadvantages of this 802.15.4 standard, which is a 2.4 GHz direct sequence spread spectrum radio. While a good general purpose radio standard, 802.15.4 is not particularly well suited to applications with significant distances between nodes, operation within buildings or other high blockage environments, or operation in high interference environments. There are proprietary implementations that implement the ZigBee networking protocol over another type of radio, but these variations are entirely proprietary and discount all of the advantages.

In the last few years, WLAN radio-signal-based positioning systems, supported by underlying Radio Frequency (RF) and Infra Red (IR) transmission technologies has seen enormous expansion and is expected to continue this trend due to the fact that it is an

economical solution providing convenient connectivity and high speed links, and can be implemented with relative ease in software (Hightower and Borriello 2001).

The distance over which RF and Infra Red IR waves can communicate depends on product design (including transmitted power and receiver design) and the propagation path, in particular in indoor environments. IR, blocked by solid objects, such as walls, metal, and even people, constitutes a main limitation. For this reason, most WLAN systems use RF, because radio waves can penetrate many indoor walls and surfaces.

RF-based WLAN covers a large area and is not restricted by line of sight issues. The range of a typical WLAN node is about 100 m (Wang and Liu 2005). A WLAN system can support a large number of nodes and vast physical areas by adding access points to extend coverage. This means using access points to cover an area in such a way that their coverages overlap each other. This can allow users to navigate around and move from the coverage area of one access point to another without even knowing they have, and at the same time seamlessly maintain the connection between their node and available access point(s). In some situations, interactions with typical building objects can affect the propagation of energy, and thus the range and coverage of the system,

However, WLAN is considered as appealing because it allows enhanced connectivity, extended coverage, and is particularly useful when mobile access to data is necessary. Additionally, user flexibility and portability can easily be reconfigured while requiring no cable infrastructure (CISCO 2002). For the above reasons, WLAN was investigated and

implemented in the presented research (Chapter 5). A proper WLAN architecture framework provides a structure to develop, maintain, and implement an acceptable operation environment, and can support implementation of automated testbed experiments conducted to continuously track mobile users.

UWB, on the other hand, is a lower power solution that benefits from causing minimal interference to other systems and UWB networks operating in the same frequency bands, and even supports multiple independent networks. UWB systems have therefore an inherent immunity to detection and interception. Additionally, the low frequencies included in the broad range of UWB frequency spectrum have long wavelength, which allows UWB signals to penetrate a variety of materials. For instance, Teizer et al. (2008) demonstrated how the UWB wireless sensing technology is capable of determining three-dimensional resource location information in object cluttered construction environments. For the aforementioned advantages, the proposed research tested and implemented the UWB technology (Chapter 5) for possible integration within the overall framework.

The last indoor tracking technology that has been investigated for possible use in the proposed context-aware framework is Indoor GPS. This system focuses on exploiting the advantages of GPS for developing a location-sensing system for indoor environments. As mentioned earlier, the GPS signal does not typically work indoors because the signal strength is too low to penetrate a building (Chen and Kotz, 2000). Nevertheless, Indoor GPS solutions can be applicable to wide space areas where no significant barriers exist. A GPS-like navigation signal is generated by a number of transmitters. This signal is

transferred through a wireless network connection providing mobility to the operator (Kang and Tesar 2004). Additionally, Indoor GPS is a rugged technology offering superior operating ranges, with accuracies in the range of few centimeters (Barrientos 2005). Another key advantage of indoor GPS is the 360-degree coverage.

In a subsequent chapter (Chapter 5), the technical characteristics of the WLAN, UWB, and Indoor GPS indoor wireless technologies introduced above are explained. Additionally, the extent to which each technology can be used to accurately calculate the positional context of a user in congested dynamic environments such as those found on construction sites is highlighted.

3.2.2. Head Orientation Tracking Systems

As mentioned in previous chapters, interpretation of a user's spatial context using position alone results in an incomplete and imprecise interpretation of spatial context. In order to interpret the fully qualified context, the position as well as three-dimensional orientation must be considered in the computation. Therefore, available head orientation trackers need to be investigated for possible use in the framework together with position tracking systems.

In the past, there have been a variety of mechanical, ultrasonic, optical, gravimetric and magnetic head-trackers available (Ferrin 1991). Mechanical trackers are capable of very

good accuracy, resolution, and interference immunity, but they have extremely limited range and tend to encumber the user.

Ultrasonic time-of-flight trackers can have relatively large range at low cost, but the need to wait for echoes to fade out before initiating a new measurement can cause low update rates, particularly when tracking large volumes (Foxlin et al. 1998).

Optical head-trackers have been built using cameras or other position-sensitive optical devices to track beacons, and using laser range measurement techniques. This method, however, tends to provide reasonable range, accuracy and resolution, but suffers from line-of-sight restrictions, reflections and very high cost (Welch et al. 2001).

Gravimetric trackers (typically using a fluid-filled inclinometer and a compass) are usually provided as a built-in component of very low-cost consumer Head Mounted Displays (HMDs). These systems are sensitive to magnetic interference, and suffer from a high level of false orientation readings caused by linear accelerations (Foxlin et al. 1998).

The most common technology today is magnetic tracking, which is convenient because it does not have the line-of-sight problems of optical and ultrasonic systems. The biggest problems with magnetic systems are distortions caused by metal objects (Nixon et al. 1998), and a very rapid decrease in accuracy and resolution with distance (Livingston and State 2001). Magnetic trackers are subject to large amounts of error and jitter. Despite

their lack of accuracy, magnetic trackers are popular because they are robust and place minimal constraints on user motion. Electromagnetic fields can travel through minor obstructions like human beings with no problem. Magnetic trackers are on the lower end of the spectrum, and can be much cheaper than any other solutions. A key portion is that the processing is not complex compared to other solutions, so the computational equipment demands are slight. In this case, accuracy is considered as reasonable, and effective for the price (Baratoff and Blanksteen 2001).

Inertial orientation trackers were also investigated in this research (Foxlin et al. 1998). They are truly sourceless, and are unaffected by almost any environmental interference. The transmitting of body tracking data via wireless means would also eliminate tethers. Additionally, the use of advanced micro-machined inertial sensors and application specific integration circuits would minimize the encumbrance of the user. Therefore, they can be used in large workspaces because they do not need transmitting source to perform tracking and there is no hardware or cabling between computer and tracker. The user would be free to move around in the real world with no restrictions. Inertial orientation trackers are also known for their low latency. They use accelerometers and gyroscopes to instantaneously derive orientation changes either by performing integration directly on the accelerometer angular rate or double integration on the gyroscopic information. Although comparatively cheap, fast and accurate, inertial trackers suffer from drift error accumulation (3-10 degrees per min) following rapid motion or long tracking periods (Baratoff and Blanksteen 2001). Moreover, the effect of gravity on accelerometers and gyroscopes induces an erroneous downward acceleration force on the tracking device.

It was thus decided to use a magnetic orientation tracker, specifically the TCM5 magnetic orientation tracker (PNI 2005). This tracker includes a built-in compass thereby avoiding manual calibration and error accumulation (Behzadan and Kamat 2005). The TCM5 orientation tracker employs solid-state magnetic field sensors which measure compass heading through a full 360 degrees of rotation. The tracking device is placed at the highest point inside the user's helmet, directly above the top of their head, and parallel to their forward line of sight.

3.3. Eye Motion Tracking Systems

3.3.1. Human Visual Perception

From the beginning of time humans have tried to explain the complex process of vision. Recorded studies of human vision date back at least to the time of Aristotle (Szaflarski 2007). In this section, initial steps in the process of human vision are presented before information about eye motion tracking systems is provided in the next subsection.

Vision is a complicated process that requires numerous components of the human eye and brain to work together (Szaflarski 2007). The eyes gather impressions, but the brain interprets them. Each eye is moved by the coordinated use of six small, strong muscles, called the extraocular muscles, which are controlled by specific areas in the brain. The brain is always trying to make sense out of what it sees. If the eyes see something that the brain can't figure out, the mind "corrects" the picture automatically (Ge et al. 2006).

The main structures of the eye are the iris, lens, pupil, cornea, retina, vitreous humor, and optic nerve (Figure 3.1).

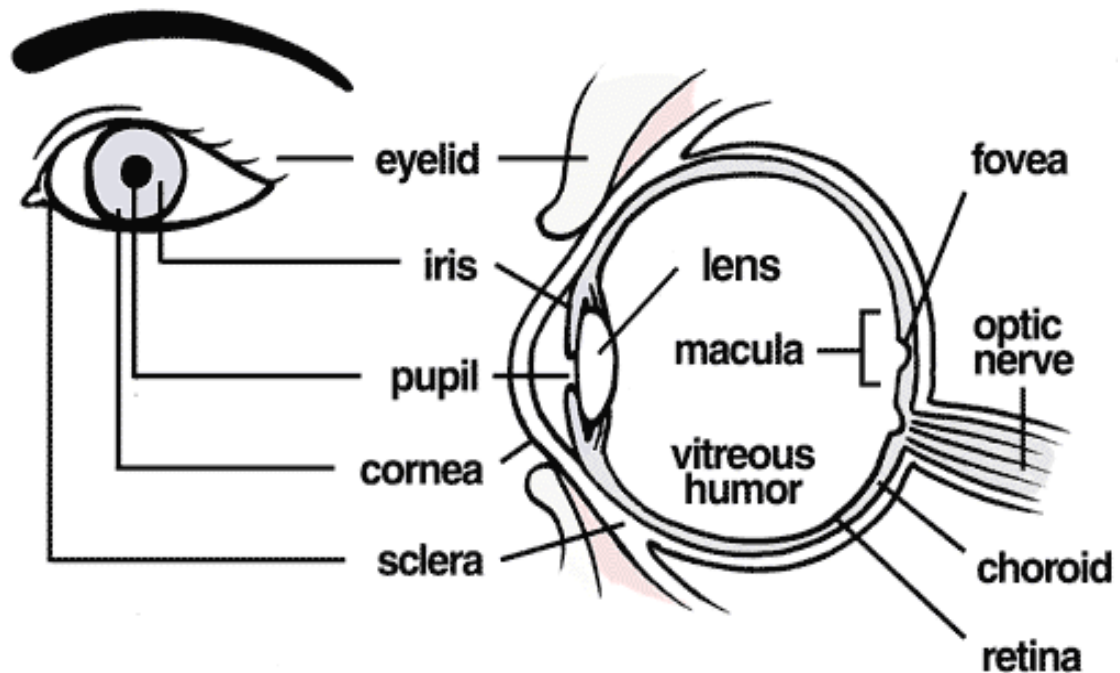


Figure 3.1 – Eye Anatomy

The normal eye thickens the lens to see near objects and flattens the lens to see distant ones (Rempel et al. 2007). The initial step of this fascinating and powerful sense is carried out in the retina of the eye. The retina is the sight center of the eye, where the most important part of seeing takes place (Nippon et al. 2007). It is a light-sensitive layer at the back of the eye that covers about 65 percent of its interior surface.

However, the picture formed on the retina has no meaning until it is sent by tiny electric discharges to the brain (Laeng et al. 2007). Specifically, the photoreceptor neurons (called photoreceptors) in the retina collect the light and send signals to a network of

neurons that then generate electrical impulses that go to the brain by the optic nerve (Figure 3.1). The brain then processes those impulses and gives information about what is seen. Therefore, the actual perception of a scene is constructed by the eye-brain system in a continuous analysis of the time-varying retinal image.

In the middle of the retina, is a small dimple called the fovea or fovea centralis (Figure 3.2).

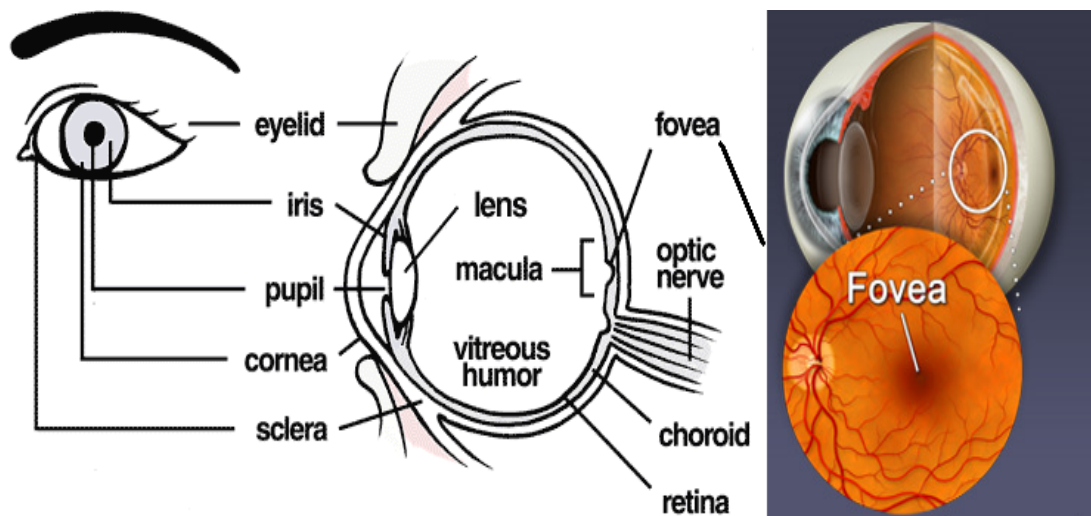


Figure 3.2 – Fovea Centralis of the Eye

The fovea provides the highest resolution and sharpest vision. Sharp images can only be seen with the very small fovea, which covers only 1 degree of the entire field of view. Therefore, the user permanently scans the environment by moving different parts onto the fovea through eye movements. The eyeball is continuously moving, so that light from the object of primary interest falls on this region before it is generated into electrical impulses and sent to the brain. These random movements can be measured with eye

motion tracking devices. In the next subsection, eye motion or gaze tracking devices are explained together with associated limitations.

3.3.2. Existing Gaze Tracking Systems and Limitations

Eye tracking consists of automatically following the eye movement and computing the gaze direction. Eye movements are necessary for scanning different regions of the visual field without having to turn the entire head and for focusing on objects at different distances (Cadik 2004). A variety of old eye-gaze (eye motion) tracking techniques have been reported in Yang and Sheena (1975).

According to the very complete analysis made by Yang et al. (2002), research in this field really started in the beginning of the 90s. A short list includes Electra-Oculography (EOG) (Kaufman et al. 1993) and Limbus, Pupil and Eyelid Tracking (Myers et al. 1991, Ebisawa 1998, Hutchinson et al. 1998, Colombo et al. 1995, Collet et al. 1997, Hu and Qiu 1994, and Stiefelhagen 1994). It includes as well Contact Lens Method, Corneal and Pupil Reflection Relationship (Hutchinson et al. 1998, Ebisawa 1998, and Hu and Qiu 1994), Purkinje Image Tracking, Artificial Neural Networks (Baluja and Pomerlo 1994) and Head Movement Measurement (Ballard and Stockman 1992, Horpresert et al. 1996, Stiefelhagen 1994, and Gee and Cipolla 1994).

Since then, the eye motion tracking technology became more accurate and less cumbersome. In parallel, the understanding and the modeling of the gaze behavior

improved, together with an extension to wide application domains such as psychiatry, cognitive science, behavioral analysis, medicine and human computer interaction (HCI) (Ciger et al. 2004).

Although EOG can potentially provide the gaze direction (Kaufman et al. 1993 and Krapichler et al. 1998), computer vision systems, especially with infrared illumination, have much better results. Latest developments achieved acceptable precision and stability by using purely mathematical (Li et al. 2004 and Kaur et al. 2003) or connectionist (Zhu and Ji 2004) approaches. These recent improvements allowed the whole setup to be more flexible and adapted for desktop, large projection screen, or Head Mounted Display (Lin 2002).

However, existing gaze tracking technologies or techniques have some limitations (Ciger et al. 2004). First, voluntary and precise control of the gaze, using eye motion tracking technologies, is tiring and a passive use of gaze is preferable to the active control (Krapichler et al. 1998). Second, these techniques cause what is called perpetual motion. While voluntary eye saccades (1 to 40 degrees of the visual angle) correspond to the visual search, micro-saccades (< 1 degree) still occur when the eye is focusing on a target. Despite the fact that the vision is suspended during saccades or eye blink, only few systems are able to really distinguish the fixation phases (Sibert and Jacob 2000).

Third, what the user is looking at is not necessarily what s/he wants to interact with. Although some studies tried to integrate statistical analysis to better recognize the user's

focus of attention (Salvucci and Anderson 2000), tracking exclusively eye movement is not natural and therefore head movements need to be tracked as well. However, coupling gaze tracking devices and techniques with head orientation tracking devices (Section 3.2.2) does not solve the problem because large head movements are still impossible, and many systems place the respective device on the head.

For all the aforementioned drawbacks of eye motion tracking technologies, using gaze in addition to head orientation for interaction in dynamic environments, such as construction sites, was not covered within the scope of this research.

3.4. Summary and Conclusions

The author has studied several positioning technologies and head orientation tracking devices to track mobile users' fully-qualified spatial context in outdoor and indoor congested environments such as those found on construction sites. Additionally, the feasibility of using eye motion tracking systems was evaluated. However, for all the drawbacks described above, the integration of such technologies in the proposed framework was not considered.

The research presented in this chapter introduced different positioning technologies (GPS, IR, Bluetooth, RFID, ZigBee, WLAN, UWB, and Indoor GPS) and many head orientation tracking techniques (mechanical, ultrasonic, optical, and gravimetric, inertial

and magnetic head-trackers) that can be used for tracking mobile users on construction sites.

Based on this study, it was found that WLAN, UWB, and Indoor GPS as position tracking systems and magnetic trackers as head orientation tracking systems offer the most promise for accurately computing mobile users' spatial parameters.

In Chapter 5, technical features and detailed information on these studied technologies, and their applicability in a context-aware information delivery framework are evaluated and compared.

3.5. References

Aitenbichler, E., and M. Muhlhauser. (2003). "An IR Local Positioning System for Smart Items and Devices", In Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), .334-339.

Aksoy Y., Chan I., Guidry K., Jones J., and Wood C. (2004). "Materials and Asset Tracking using RFID: A Preparatory Field Pilot Study", FIATECH Smart Chips Project Report- available at: <http://www.fiatech.org> (Accessed June 15, 2006).

Aranov, V. Y., Sholukha, V. A., Van Sint Jan, S. (2004). "Biomechanical Analysis and Visualization Tool of Human Morphological and Motion Data", In Proceedings of the International Conference on Computer Graphics and Vision (Graphicon), Moscow, Russia- available at: <http://www.graphicon.ru/> (Accessed March 15,2009).

Ayre, L. 2004. RFID and Libraries [online]-available at: http://galecia.com/included/docs/position_rfid_permission.pdf (Accessed: May 18, 2006).

Aziz, Z., Anumba, C.J., Ruikar, D., Carrillo., P.M., and Bouchlaghem.,D.N. (2005). "Context-aware information delivery for on-Site construction operations", Proceedings of the 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universitat Dresden, Germany, CBI Publication No:304, 321-327.

Ballard, P., and Stockman, G.C. (1992). "Computer Operation via Face Orientation, Pattern Recognition", In Proceedings of the 11th Computer Vision and Applications Conference.

Baluja, S., and Pomerlau,D. (1994). "Non- Intrusive Gaze Tracking Using Artificial Neural Networks", CMU Technical Report, CMU-CS-94-102, School of Computer Science, Carnegie Mellon University.

Baratoff, G., and Blanksteen, S. (2001). "Tracking Devices"- available at: <http://www.hitl.washington.edu/scivw/EVE/I.D.1.b.TrackingDevices.html>(Accessed February 25, 2006).

Barbič, J.; and James, D.L.(2007): "Time-Critical Distributed Contact for 6-DoF Haptic Rendering of Adaptively Sampled Reduced Deformable Models", In Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SESSION: Real-time simulation, 171-180.

Ciger, J., Herbelin, B., and Thalmann, D. (2004). "Evaluation of Gaze Tracking Technology for Social Interaction in Virtual Environments", 2nd Workshop on

Modelling and Motion Capture Techniques for Virtual Environments, CAPTECH04, 9-11.

Cleary, K., and Brooks, T. (1993). "Kinematic Analysis of a Novel 6-DOF Parallel Manipulator", In Proceedings of the IEEE International Conference on Robotics and Automation, Atlanta, GA, 708-713.

Collet, C., Finkel, A., and Gherbi, R. (1997). "CapRe: A Gaze Tracking System in Man-Machine Interaction", In Proceedings of the IEEE International Conference on Intelligent Engineering Systems.

Colombo, C., Andronico, S., and Dario, P. (1995). "Prototype of a Vision-Based Gaze-Driven Man-Machine Interface", In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.

Ebisawa, Y. (1998). "Improved Video-Based Eye- Gaze Detection Method", In Proceedings of the IEEE IMTC, Hamamatsu.

El-Rabbany, A. (2002). Introduction to GPS: The Global Positioning System, Artech House Inc., Norwood, MA.

Ge, S., Saito, T., Wu, J.L., and Iramine, K. (2006). "A Study on Some Optical Illusions Based Upon the Theory of Inducing Field", Neuroscience J. 1, 4205-4208.

Gee, A.H., and Cipolla, R. (1994). "Determining the Gaze of Faces in Images", Technical Report, CUED/FINFENG/ TR 174, Department of Engineering, University of Cambridge.

Ferrin, F. (1991). "Survey of helmet tracking technologies", In Proceedings of SPIE, 1456, 86-94.

Horpresert, T., and Yacoub, Y., and Davis, L.S. (1996). "Computing 3-D Head Orientation from a Monocular Image Sequence", In Proceedings of the Second International Conference on Automatic Face and Gesture Recognition.

Hu, B., and Qiu, M. (1994). "A New Method for Human-Computer Interaction by using Eye Gaze", In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics.

Hutchinson, T.E., White, K.P., Martin, W.N., Reichert, K.C., and Frey, L.A. (1998). "Human Computer Interaction Using Eye Gaze Input", In Proceedings of the IEEE Trans. on Systems, Man, and Cybernetics, 19(6), 1527-1534.

Kaufman, A.E., Bandopadhyay, A., Shaviv, B.D. (1993). "An Eye Tracking Computer User Interface", Proceedings of the Research Frontier in Virtual Reality Workshop, IEEE Computer Society Press, 78-84.

- Kaur, M., Tremaine, M., Huang, N., Wilder, J., Gacovski, Z., Flippo, F., and Sekhar, C.M. (2003). "Where is .it? Event synchronization in gaze-speech input systems". In Proceedings of the 5th International Conference on Multimodal Interfaces (ICMI).
- Krapichler, C., Haubner, M., Engelbrecht, R., and Englmeier, K. (1998). "VR interaction techniques for medical imaging applications", *Computer Methods and Programs in Biomedicine*, 56(1), 65-74.
- Laeng, B.K., Waterloo, S.H., Johnson, S. J., Bakke, T., Lag, S. S., Simonsen, and Hogsæet, J. (2007). "The Eyes Remember It: Oculography and Pupillometry During Recollection in Three Amnesic Patients", *J. Cog. Neuroscience*, 19, 1888-1904.
- Lin, C. (2002). "An eye behavior measuring device for VR system". *Optics and Lasers in Engineering*, 38(6), 333-359.
- Lin, C., Huan, C., Chan, C., Yeh, M., and Chiu, C. (2004). "Design of a computer game using an eye-tracking device for eye's activity rehabilitation", *Optics and Lasers in Engineering*, 42(1), 91-108.
- Sibert, L.E., and Jacob, R.J.K. (2000). "Evaluation of eye gaze interaction". In CHI 2000 Conference on Human Factors in Computing Systems.
- Lu, G.; Jiang, C.; and Geng, Y. (2002). "Adaptive Fuzzy Control of High Accuracy Airborne Optical-Electronic Tracking System", *Journal of Acta Armamentarii*, 23(4), Zahongguo Binggong Xuehui Publishers, 533-535.
- Nippon, Gananka, G., and Zasshi, G. (2007). "Physiology and Pathology of Visual Information Processing", *Neuroscience J.*, 111, 160-191.
- Nixon, M.A., McCallum, B.C., Fright, W.R., and Price, N. B. (1998). "The Effects of Metals and Interfering Fields on Electromagnetic Trackers", *Presence Journal*, 7(2), 204-218.
- Myers, G.A., Sherman, K.R., and Stark, L. (1991). "Eye Monitor", *IEEE Computer Magazine*, 14-21.
- Pateli, A., Giaglis, G.M., Fouskas, K., Kourouthanassis, P. and Tsamakos, A. (2002). "On the Potential Use of Mobile Positioning Technologies in Indoor Environments", In the Proceedings of 15th Bled Electronic Commerce Conference -e-Reality: Constructing the e-Economy, Bled, Slovenia,
- Rempel, D., Willms, K., Anshel, J., Jaschinski, W., and Sheedy, J. (2007). "The Effects of Visual Display Distance on Eye Accommodation, Head Posture, and Vision and Neck Symptoms", *J. Neuroscience*, 49, 830-838.

Salvucci, D. D., and Anderson, J. R. (2000). "Intelligent gaze-added interfaces". In CHI 2000 Conference on Human Factors in Computing Systems.

Schilit, B.N., Adams, N., and Want R. (1994). "Context-aware computing applications", Workshop on Mobile Computing Systems and Applications (WMCSA), Santa Cruz, CA, 85-90.

Stiefelhagen, R. (1996). "Gaze Tracking for Multimodal Human-Computer Interaction", Diplomarbeit, Universitiit Karlsruhe.

Szaflarski, D. M. (2007). "How We See: The First Steps of Human Vision", Sch. Neuroscience J. 11, 67-86.

Welch, G., Bishop, G., Vicci, L., Brumback, S., Keller, K., and Colluci, D. (2001). "High-Performance Wide-Area Optical Tracking: The HiBall Tracking System, Presence: Teleoperators and Virtual Environments, 10(1), MIT Press.

Xiang, Z., Song, S., Chen, J., Wang, H., Huang, J., and Gao, X. (2004). "A Wireless LAN-Based Indoor Positioning Technology", IBM Journal of Research and Development, 48 (5/6), 617-626.

Young, L.R., and Sheena, D. (1975). "Survey of Eye Movement Recording Methods", Behavior Research Methods and Instrumentation, 7(5), 397-429.

Zhu, Z., and Ji, Q. (2004). "Eye and Gaze Tracking for Interactive Graphic Display", Machine Vision and Applications, 15(3), 139-148.

Chapter 4

Evaluation of Standard Product Models and Project Databases for Information Access and Retrieval in Construction Applications

4.1. Introduction

The construction industry is inevitably characterized as one which generates vast amounts of graphical and textual data. Effectively managing such an enormous volume of information to ensure its accuracy and availability in a timely manner is crucial to the successful completion of any project (Boddy et al. 2007). Inaccuracies in information access and retrieval can invariably lead to project delays, uneconomical decisions, accidents, or even the complete failure of a project. Therefore, the efficient and timely communication of relevant data, and an improved and integrated flow of information are critical concerns to all members of the construction industry, in particular the project participants involved in the facility construction.

For a typical construction information retrieval situation, an inspector for example, would want to find all available information about one construction activity, i.e. a building wall. The inspector would perhaps find the drawings in a Computer-Aided Design (CAD) file, the specifications in a text document, the cost estimates in a spreadsheet, the schedule in a particular application format, the contracts in a text document, and price quotes in different websites or other sources. In this case, the major task would be how to index, integrate and retrieve information from these different unorganized data sources.

Recent years have seen the emergence and development of a plethora of visual software modeling tools and modeling standards and methods (Fahdah 2008). For instance, standard product models and project information systems, by representing the whole building information in one coherent virtual model, can facilitate the exchanging, sharing and delivery of pertinent construction information, as well as enhance work efficiency and collaborative processes (Halfawy et al. 2001, Aziz et al. 2005).

In this chapter, the author capitalizes on these existing methodologies and evaluates the applicability of several interoperable product models and database management systems for automated context-aware information access and retrieval in construction applications.

4.2. Background

During the early 1980s, Information Technology (IT) was almost exclusively used to support activities which could be categorized as creation of new information. The use of CAD started to proliferate, but still the emphasis was on support for the creation and the viewing of data (Bjork 1999). The support for information retrieval and making information available in the construction industry came in the form of very expensive plotters enabling the plotting of drawings which emulated manually produced drawings. Construction managers, inspectors and other site personnel were mainly occupied with information retrieval and communication with co-workers. In the early years, there was relatively little IT support for retrieval and communication tasks, especially in construction (Aziz et al. 2005).

There has therefore been a need for an effective communications infrastructure that facilitates seamless inter-working between the disparate professionals involved in construction projects (Amor and Anumba 1999). Such an infrastructure needs to be based on interoperable information and communications technologies, and should facilitate information interchange between members of the project team and across stages in the project lifecycle from construction to inspection to maintenance. It has been suggested that the central kernel of this communications infrastructure should be inhabited by a shared construction project model (Anumba et al. 1997) in the form of integrated product models and project databases (Kiviniemi 2005a, b).

There are several views within the construction industry and the research community on what constitutes an integrated project database. Some see it simply as an amorphous collection of all the information relating to a project, irrespective of the medium of storage (paper drawings and specifications, CAD files, etc.) or the method of dissemination of the project information (Fisher et al. 1997). Others see it in terms of a single database which holds all the information on a project and which is accessible to all members of the project team (Niemiöja 2005). Some others view the integrated project database as an integration of product models, which hold information relating to the building product, and process models, which hold information regarding the construction and business processes required to translate the product information into a physical product - the constructed facility (Kiviniemi 2005a, Amor and Anumba 1999, and Bjork and Penttila 1989).

The next sections present common integrated product models and project databases that were investigated in this research and viewed as computer-interpretable models of buildings enabling efficient information sharing between different project parties across the life-cycle of a constructed facility.

4.3. Building Information Models

Building Information Models (BIMs) have become an active research area in Computer Integrated Construction over the last ten to twenty years. From an industrial perspective, the rise of the trend towards BIMs and model based engineering is due to the inadequate

interoperability in the industry. Gallaher et al. (2004) indicated that US\$15.8B is lost annually in the U.S Capital Facilities Industry due to the lack of interoperability. Today, BIMs are seen as the main facilitators of integration, interoperability, collaboration and process automation. A BIM is a digital information structure of the objects that make up a building, captured in the form, behavior and relations of the parts and structural components within this building. BIMs can have several pieces of information including CAD drawings, project specifications and schedule, bill of materials, change orders, and other information pertaining to any construction entity of the building (Chin et al. 2008, Tse et al. 2006)(Figure 4.1). It serves then as a shared knowledge resource for information about a building, forming a reliable basis for decisions during its whole lifecycle, i.e. from inception onwards (NBIMS 2006).

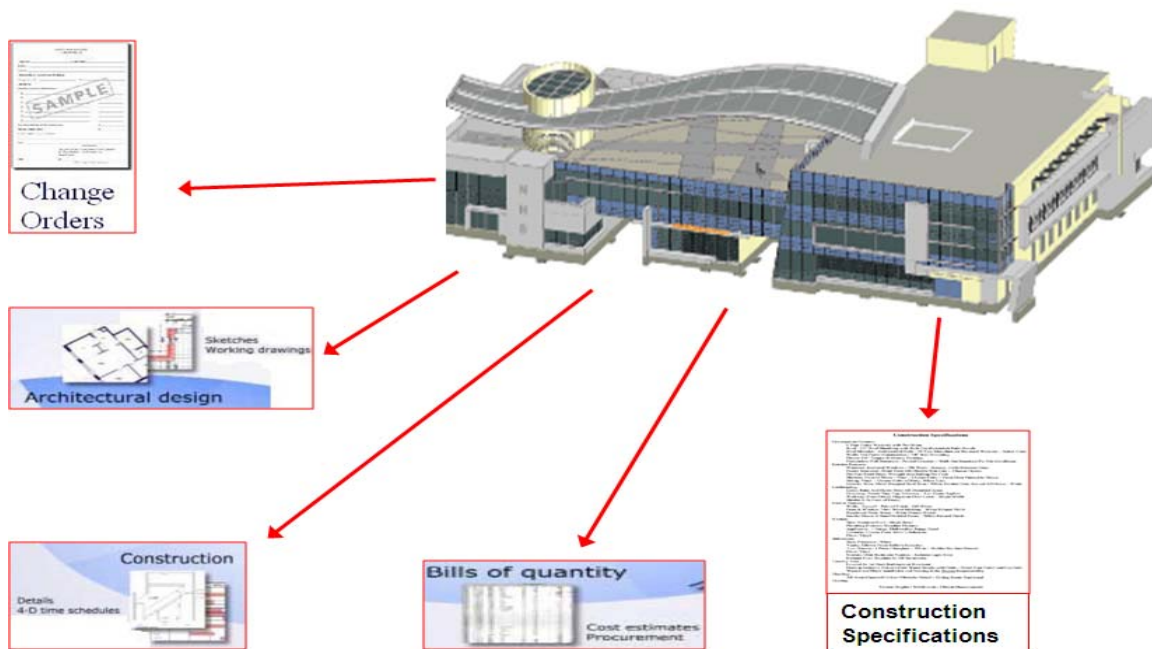


Figure 4.1 – A Sample Product Model

In the design process, for instance, the production of separate drawings is replaced by the construction of a virtual building. A drawing in the traditional sense is merely a view to the model at a chosen moment, as the whole building and its environment are included in one single product model. Everything is modeled using three-dimensional building components. The model is then maintained and revised throughout the project from start to hand-over. Additionally, the use of one virtual building facilitates and automates the data transfer between different applications used in construction, inspection and maintenance processes (Hakkinen 2007).

Recently, there has been an upsurge in the commercial interest for building product modeling, through an initiative by several large end users of commercial CAD systems to start to define the object classes needed in building product modeling (Howard and Bjork 2007). The research concerning such models was envisaged as early as the late 1970s (Eastman 1979) but started to gain more momentum around 1985, when the ISO STEP (ISO 10303:1992) standardization project started. STEP stands for Standard for the Exchange of Product Data (Step Tools, Inc. Website 2008) and tries to solve the engineering data exchange and sharing needs of all branches of industry (Owen 1993). Early attempts at building standardization within STEP included the global AEC reference model (Gielingh 1988) and the building systems model (Turner 1990). In the mid 1990s the product modeling standardization for the building domain was taken over by an industry consortium called the International Alliance for Interoperability (IAI), buildingSMART.

The first version of IAI/buildingSMART industry foundation classes (IFC) was released in 1997. IFC is an international specification for product data exchange and sharing for Architecture, Engineering, Construction and facilities management (AEC/FM). IFC enables interoperability between the computer applications for AEC/FM (IAI 2003). A subset of IFC is approved as ISO/PAS 16739.

Over the last decade, IFC has matured as a standard BIM in supporting and facilitating interoperability across the various phases of the construction life cycle. IFC, showed good prospect for meeting industry requirements and was adopted by many construction professionals (Wix et al. 2007). IFCs are data elements that represent parts of a facility and contain relevant information about those parts. IFCs are used by applications such as CAD tools to assemble computer readable facility models that contains all information of the parts and their relationships (Figure 4.2).

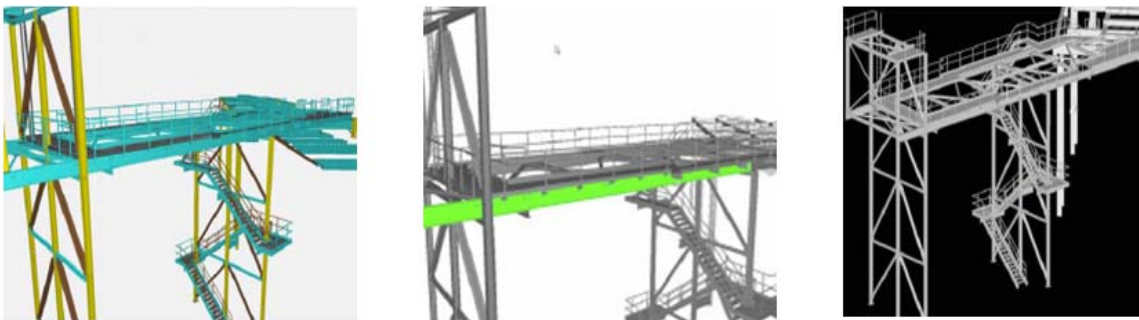


Figure 4.2 – IFC Product Model Viewed in Different CAD Tools

Previous research efforts focused on examining the applicability of the IFC and presenting an architecture for achieving the requirements for IFC-based construction product libraries (Owolabi et al. 2003). Work by Owolabi et al. (2003) also presented the

implementation of the architecture and envisioned it as a multi-tiered, service-based, message-oriented, client-server application, which is a hybrid solution combining the elements of Web services and multi-tier applications. Other researchers such as Halfawy et al. (2002) proposed and employed various aspects of the aforementioned architecture for IFC related implementations. They recommended IFC-based messaging for systems interoperability and efficient data exchange.

Froese (2003) highlighted the need for information and data exchange through IFC-based systems. His work showed the integration of model-based integrated systems together with web-based technologies such as document management, workflow management and e-commerce applications. Some of the general characteristics of integrated systems are that they cluster many functional views around an overall task of building, maintaining, and interacting with an integrated project database. They adopt a model-based approach: all information is structured around an object-oriented data model of the project. Generally, the product model (the physical components of the built facility) plays a central role, but this is interlinked with many other types of project information. Finally, most of the information contained within the system can be exchanged with other systems.

In 1999, another standard, called aecXML was established within IAI. aecXML is intended to be used to facilitate information exchange of AEC data on the Internet. The whole concept, initially developed by Bentley, was introduced on August 13, 1999 (The XML Cover Pages Website 2008). aecXML's further development is being directed by

the aecXML Working Group, an independent coalition of interested parties (The aecXML Working Group 1999)

It is a framework for using the eXtensible Markup Language (XML) standard for electronic communications in the architectural, engineering and construction industries. It includes an XML schema to describe information specific to the information exchanges between participants involved in designing, constructing and operating buildings, plants, infrastructure and facilities. The various software applications used by these participants can transfer messages formatted according to the aecXML schema to coordinate and synchronize related project information. In addition, a standard aecXML specification can also facilitate e-commerce between suppliers and purchasers of equipment, materials, supplies, parts and services based on that same technical information (aecXML white paper 1999).

The aecXML system is designed for all the non-graphic data involved in the construction industries. Much of the data within the scope of the aecXML system is currently stored as arbitrarily formatted text authored in word processors, or as spreadsheets or databases. The aecXML system provides a set of keywords and named data attributes, so that all users will employ the same naming logic and grouping, and software will be able to make use of the data without it needing to be interpreted by humans and manually re-entered in each program's required form. Therefore, aecXML uses IFC to create a vendor-neutral means to access data generated by building information modeling (Cheng et al. 2002).

IFC and aecXML, however, are not the only open standards for interoperability in the building industry. This is primarily because IFC, in particular, is not yet complete in its representation of all the disciplines. It serves the needs of most architectural, general contracting, and some facilities management tasks, but still lacks a sufficiently complete representation of structural engineering to be effectively used by structural engineers or inspectors. Another open standard has emerged that caters to at least one significant segment of the structural engineering field, that of steel construction. This is the CIMsteel Integration Standards (CIS/2) standard. Since 2003, CIS/2 has been firmly established as the de facto interoperability standard for structural steel (Crowley and Watson 2000, Eastman et al. 2005).

The CIMsteel Integration Standards Release 2 (CIS/2) is a logical product data model for structural steel building information (Lipman and Reed 2003). It is the result of the pan-European Eureka EU130 CIMsteel Project and has been endorsed by the American Institute of Steel Construction (AISC) as the technical basis for their Electronic Data Interchange initiative (Kamat and Lipman 2006). The objective was to decrease steel construction timelines, eliminate duplication of effort, and enable software applications to exchange data electronically.

CIS/2 deals with information about the steel structure throughout its analysis, design, detailing, and fabrication life cycle (Figure 4.3). CIS/2 structures can thus be represented as analysis, design, or manufacturing models (Kamat and Lipman 2006). Information on any of these models can coexist in the same CIS/2 file. An analysis model of a steel

structure consists of nodes and elements and supports several static and dynamic analysis methods. A design model represents a steel structure as a design assembly to allow member and connection design. A design assembly can be partitioned into other simpler design assemblies and eventually into design parts and design joint systems. The design parts and joint systems respectively form the conceptual representations of a basic piece of steel and a basic joint system. A manufacturing model in CIS/2 represents a steel structure as manufacturing assemblies for the purpose of detailing, production planning, and manufacturing.

As such, CIS/2 has been implemented by many specific software packages to create a seamless and integrated flow and archival of information among all entities involved in all life cycle stages of steel framed structures (Reed 2002). Any software application can seamlessly have CIS/2 import and/or export capabilities.

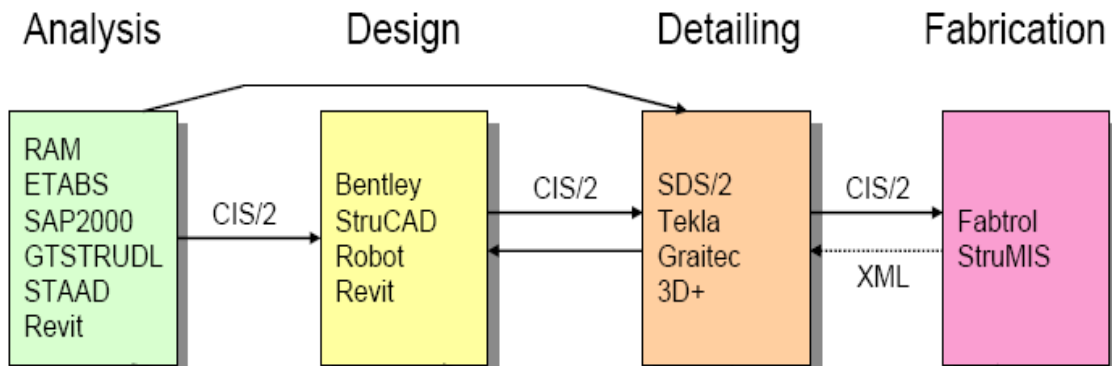


Figure 4.3 – Data Exchange Scenario with CIS/2

[Images courtesy of Mr. Robert Lipman, NIST]

The geometry of the structure is only one property of the product data model. Other attributes of the structure include how parts are combined into assemblies and structures, analysis loads and reactions, material types, connection details, associations between members and drawings, and modification history. Therefore, the CIS/2 standard provides data structures for multiple levels of detail ranging from frames and assemblies to nuts and bolts (Figure 4.4).

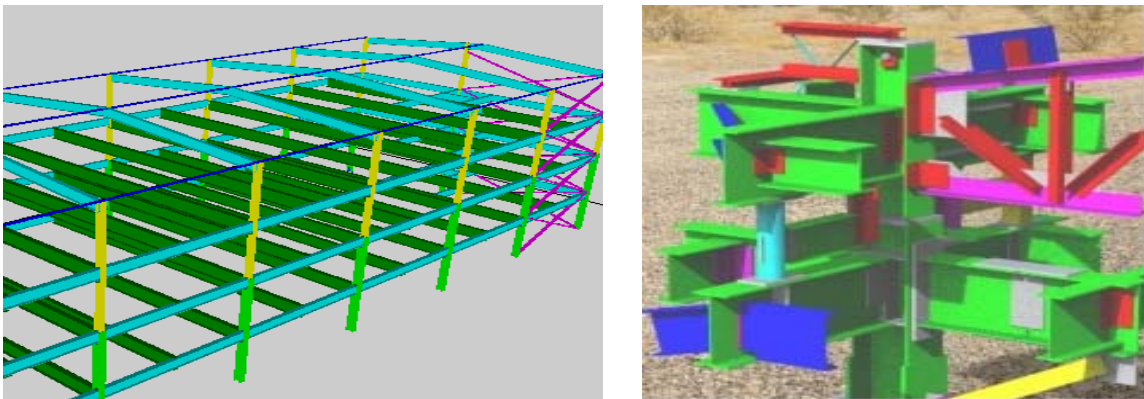


Figure 4.4 – CIS/2 Product Model of a Building Structural Frame (Left) and CIS/2 Product of Connection Details with Bolts (right)
[Images courtesy of Mr. Robert Lipman, NIST]

The product data model is defined by a schema, STEP as well, that details how all the information in the CIS/2 standard is represented and how each entity relates to each other. It also defines rules for using various entities and the information those entities contain.

The CIS/2 standard has been demonstrated to be a very effective communication tool and was successfully deployed on a mobile computing platform at the National Institute of Standards and Technology (NIST) (Saidi et al. 2005). For these reasons, in this research,

the utility of the interoperable CIS/2 product model was investigated and is presented as a suitable data structure to represent cross-referenced building data for the evaluation of the designed automated information retrieval technique (Chapter 7).

4.4. Database Management Systems

Despite the recent upsurge of BIM tools, contractors, engineers, managers, etc. still use standard databases management systems (DBMS) while they transition to BIM. A database is a well organized collection of information. A DBMS gives the software tools needed to manage data in a flexible manner. It includes various features which help to add, modify or delete information or data from the database (Date 2003). One of the most important features of a DBMS is asking questions or queries about the information stored in the database and generating reports or forms analyzing selected content. Information stored in databases must be organized into related record types through a process known as database design. The DBMS that is chosen must be able to manage different relationships, which is where database models come into play. The schema of each database is defined according to a specific model. The four most common types of organizations are the hierarchical model, network model, relational model and object model (Ullman and Widom 1997).

A hierarchical data model (Kamfonas 1992) is a data model in which the data is organized into a tree-like structure (Figure 4.5). The structure allows repeating

information using parent/child relationships: each parent can have many children but each child only has one parent.

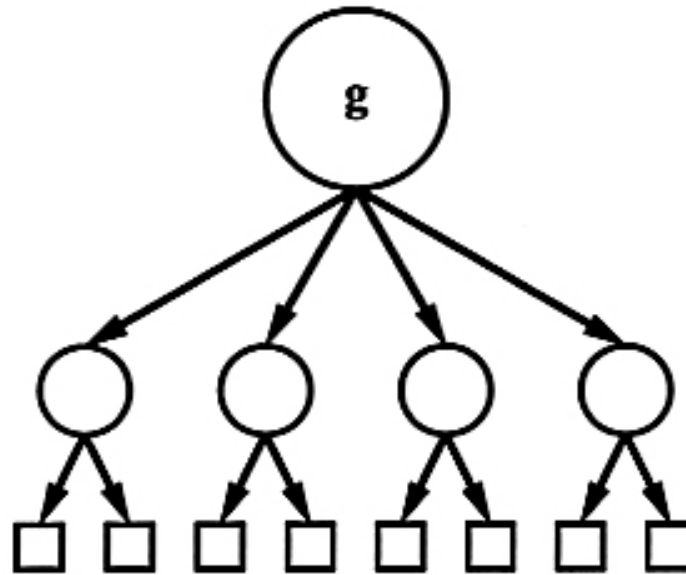


Figure 4.5 – Hierarchical Database Model

In hierarchical databases, attributes of specific records are listed under an entity type and entity types are connected to each other through one-to-many relationships, also known as 1:N mapping. Originally, hierarchical relationships were most commonly used in mainframe systems, but with the advent of increasingly complex relationship systems, they have now become too restrictive and are thus rarely used in modern databases. Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s (Andany et al. 1991).

If any of the one-to-many relationships are compromised, for example a child node having more than one parent node, the database structure switches from the hierarchical

to a network model (Figure 4.6). In the 1970s, the Conference on Data Systems Languages (CODASYL) formally defined the network model (Bachman 1973).

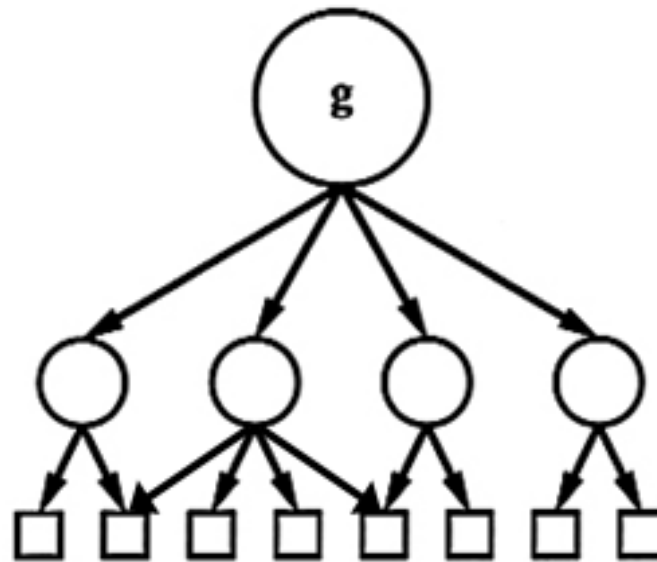


Figure 4.6 – Network Database Model

In the network model of a database it is possible for a record to have multiple parents, making the system more flexible compared to the strict single-parent model of the hierarchical database (Figure 4.5). The model is made to accommodate many to many relationships, which allows for a more realistic representation of the relationships between entities. Even though the network database model gained popularity and was widely implemented, it is now rarely used because of the availability of more competitive models that possess the higher flexibility and productivity demanded in today's ever advancing period. Therefore, this model was eventually displaced by the relational model, which offered a higher-level, more declarative interface.

Working under the assumption that file systems (which often use the hierarchical or network models) are not considered databases, the relational database model is the most commonly used system today.

The relational model was invented by E.F. Codd (Codd 1970) as a general model of data, and subsequently maintained and developed by Chris Date and Hugh Darwen among others (Date and Darwen 2000). In a relational model, the design of the records is organized around a set of tables. Each table is broken down into additional components (Figure 4.7) such as a row in a table. Each row is referred to as a record. Each column in a table is a category of information referred to as a field. One item of data is called a data value.

The diagram shows a table titled 'dbo_building : Table' with the following data:

ID	Object	Material	Type	Scheduled	Completed
1	Column1	Concrete	Column	09/10/2006	09/02/2006
2	Column2	Concrete	Column	09/10/2006	09/02/2006
3	Column3	Concrete	Column	09/16/2006	09/10/2006
4	Column4	Concrete	Column	09/16/2006	09/10/2006
.....
10	Beam1	Concrete	Beam	09/22/2006	09/21/2006
▶

Annotations in the diagram:

- Field:** An arrow points to the 'Type' column header.
- Data Value:** An arrow points to the '09/02/2006' value in the 'Completed' column of the first row.
- Record:** An arrow points to the entire third row of the table.

Figure 4.7 – Sample Database Table for a Building

A key piece to the relational database concept is the field in which the data value uniquely identifies the record, i.e. *ID* in Figure 4.7, and therefore all respective data values remain unique to each record. This field is referred to as the primary key. In

Figure 4.8, *Object ID* in *Object Table*, *Activity ID* in *Activity Table*, *Schedule ID* in *Schedule Table*, and *Estimation ID* in *CostEstimation Table* are primary keys. Those keys are used for mapping or cross-referencing the tables to each other. A table that contains the parent or primary information (i.e. *Schedule Table*) can be linked to the appropriate row(s) in a table that contains child information (i.e. *Info Type Table*) based on a common indexed field of the two tables (*Schedule ID*), called primary key in the parent table and foreign key in the child table. This way, keys represent the data in the tables and their relationships.

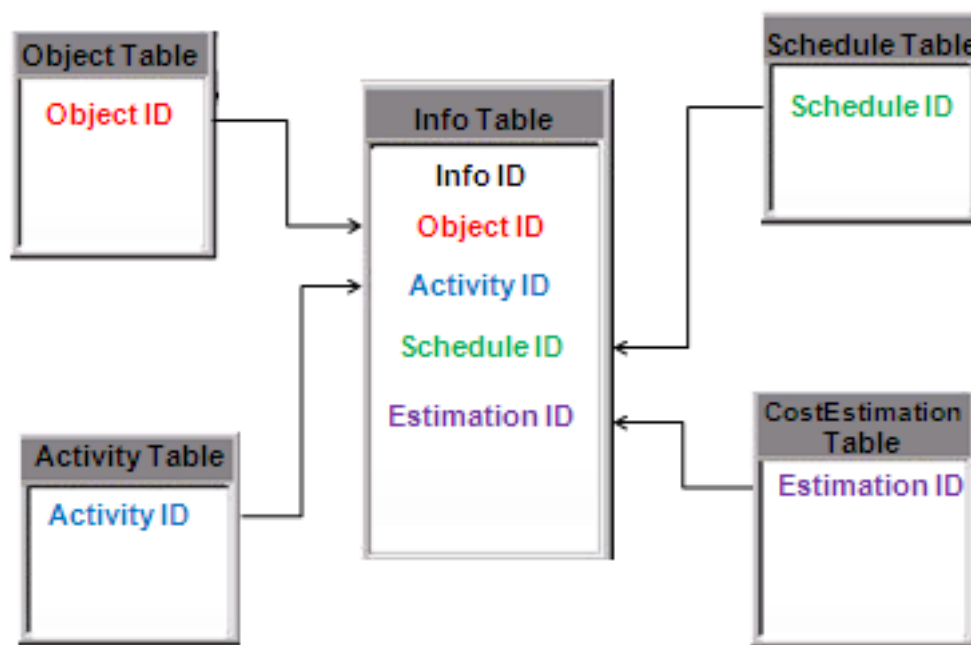


Figure 4.8 – Relational Database Model

There are three main types of relationships, One-to-One, One-to-Many, and Many-to-Many relationship. A one-to-one relationship exists when the primary record will have only one related record. Another determining factor is both fields using the relationship

are primary key fields: If the related data value in both tables must be unique then there can only be one matching record, thus a one-to-one relationship.

A one-to-many relationship is the most common type of relationship. A one-to-many relationship exists when the primary record can have many related records. A one-to-many relationship can be defined either using an embedded foreign key column, or using a join table. An embedded foreign key is a column defined in the table on the many side of the relationship that holds a key to the table in the one side of the relationship. A join table is a table whose sole purpose is to store mapping values between the key and foreign key of the two tables involved in the relationship.

A many-to-many relationship refers to a relationship between two tables in which the first one may contain a parent row for which there are many children in the second one and vice versa. Because most DBMSs only support one-to-many relationships, it is necessary to implement such relationships physically via a third junction table, with two one-to-many relationship. In this case the logical primary key for the junction table is formed from the two foreign keys (i.e. copies of the primary keys of both tables).

In *The Third Manifesto* (first published in 1995), Date and Darwen show how the relational model can work together with the object model and accommodate certain desired object-oriented (OO) features, leading to what is called the object-relational database (ORD) (Stonebraker et al. 1999). An ORD is similar to a relational database, but with an object-oriented behavior: objects, classes and inheritance are directly

supported in database schemas and in the query language (Booch 1994). ORDs allow the user to embed new classes of data objects into the relational data model abstraction. It supports extension of the data model with custom data-types and methods. Therefore, by synthesizing both models, the ORD takes advantage of both the performance and robustness of the relational model semantics and flexibility of the object model.

Relation is still the fundamental abstraction, with integration of OO features (Ullman and Widom 2002). All records in a table are structurally identical in that they all consist of a fixed number of values of specific data types stored in columns or fields that are named as part of the table's definition. The most important distinction between relational database tables and object-relational database tables is the way that ORD columns are not limited to a standardized set of data types. In addition to defining the structure of a table, integrity constraints can be included in its definition.

The leading DBMSs are based on the relational model. A relational database management system (RDBMS) is a program that allows creating, updating, and administering a relational database. Most commercial RDBMSs use the Structured Query Language (SQL) to access the database. In this section, Oracle, Microsoft's SQL Server and Microsoft Access RDBMSs have been evaluated for possible implementation of relational principles in the design of the database needed to fulfill the information access and retrieval component requirements of the proposed methodology (Chapter 2). Additionally, in order to ensure flexibility in the database design, relational principles

were then coupled with OO features, using an OO programming language, to achieve an ORD (Chapter 7).

Oracle (Loney 2004) is a multi-user database management system. It is a software package specializing in managing a single, shared set of information among many concurrent users. Oracle is one of many database servers that can be plugged into a client/server equation. Oracle works to efficiently manage its resource, a database of information, among the multiple clients requesting and sending data in the network. Oracle has many important features that make it an efficient database server choice for client/server computing. Oracle supports all major operating systems for both clients and servers. In the latest version, innovative features such as self-tuning database components, sophisticated partitioning schemes and data compression have maintained Oracle's reputation for performance, scalability and optimal use of hardware resources for industry applications. With Data Guard, Real Application Clusters (RAC) and Oracle Advanced Security (ASO), Oracle has shown itself to be a leader in database availability and security as well.

On the other hand, Microsoft SQL Server (Agrawal et al. 2004) is a relational model database server produced by Microsoft. It provides an environment used to generate databases that can be accessed from workstations, the web, or other media such as a personal digital assistant (PDA). In the nine years since release of Microsoft's previous SQL Server product (SQL Server 2000), advancements have been made in performance, the client IDE tools, and several complementary systems packaged all together. SQL

Server supports different data types, including primary types. SQL Server also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). A database can also contain other objects including views, stored procedures, indexes and constraints, in addition to tables, along with a transaction log. Additionally, an SQL Server database can contain a maximum of 231 objects, and can span multiple OS-level files with a maximum file size of 220 TB. The main mode of retrieving data from an SQL Server database is querying for it. The query declaratively specifies what is to be retrieved. It is processed by the query processor, which figures out the sequence of steps that will be necessary to retrieve the requested data. The sequence of actions necessary to execute a query is called a query plan. There might be multiple ways to process the same query. SQL Server possesses the ability to choose the plan that is supposed to yield the results in the shortest possible time.

Microsoft Access (Allison and Berkowitz 2005), in particular, provides users with simple and highly flexible DBMS solutions in the online database market today, mainly in terms of compatibility and sharing. Additionally, MS Access, installed as part of the Microsoft Office suite, is considered cheaper compared to other databases such as Microsoft SQL server and Oracle which are quite expensive (i.e. Oracle 25-user pricing is around \$87,000 and Microsoft SQL server 25-user pricing is around \$ 45,000) (New York Times Company 2008). To add further, Access allows relatively quick and fast development because of very good GUI design tools, and high level integration of GUI design and data objects.

When it comes to security, MS Access is limited to security in terms of username/password on the database. If the database design needs to be secured to prevent changes, Access databases can be locked or protected by converting the database to an .MDE file. It is also subject to Windows Server security on the file itself as well as the folder it resides in (Wikipedia 2008). MS Access uses standards that enable applications to work in large and harsh environments (DatabaseDev Website 2008). Therefore, it constitutes a good and cheap means for storing large volumes of information which needs to be held for a long time and retrieved flexibly using varying criteria and sometimes in formal reports. It allows the user to store, retrieve, sort, analyze and print information stored in the database. It also allows defining input screens, and adding cross-referencing to make sure that data is internally consistent. Additionally, MS Access is feasible if multiple users, and especially non-technical users, will use the system, and will need to update the contents of the data store at one time.

For the above reasons and for the purpose of this research, MS Access DBMS was adopted as the software tool to create and design project databases (Chapter 7).

4.5. Summary and Conclusions

The author has studied several types of building information models and project information systems for access and retrieval of contextual information on identified objects in users' field of view.

The research presented in this chapter introduced three building information models (IFC, aecXML, and CIS/2) and three database management systems (Oracle, Microsoft SQL Server, and Microsoft Access) that can be used for information support on construction sites. Based on this study, it was found that CIS/2 as a product model and MS Access as a database management system can be presented as suitable project repositories to represent cross-referenced building data for the evaluation of the designed automated information retrieval component of the proposed methodology .

In Chapter 7, technical features and detailed information on these studied technologies, and their applicability in a context-aware information delivery framework are evaluated.

4.6. References

- Agrawal, S., Chaudhuri, S., Kollar, L., Marathe, A., Narasayya, V., and Syamala, M, “Database Tuning Advisor for Microsoft SQL Server 2005”, Proceedings of the 30th International Conference on VLDB, Toronto, Canada, 2004, pp. 1110–1121.
- Allison, C.L., and Berkowitz, N. (2005). SQL for Microsoft Access, Edition: illustrated, Wordware Publishing, Inc., Plano, TX.
- Amor R., and Anumba, C.J. (1999) “A Survey and Analysis of Integrated Project Databases”, In Proceedings of the Concurrent Engineering Construction (CEC): Challenges for the New Millennium, CIB Publication 236, Espoo, Finland, 217-227.
- Andany, J., Leonard, M., and Palisser, C. (1991). “Management of Schema Evolution in Databases”, In Proceedings of Very Large Databases Conference, Barcelona, Spain, 161-170.
- Anumba C. J., Baron G., and Duke A. (1997): “Information and Communications Technologies to Facilitate Concurrent Engineering in Construction”, BT Technology Journal, 15(3), 199-207.
- Aziz, Z., Anumba, C.J., Ruikar, D., Carrillo, P.M., Bouchlaghem, D.N. (2005). “Context aware information delivery for on-Site construction operations”, Proceedings of the 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universitat Dresden, Germany, CBI Publication No:304, 321-32.
- Bachman, C.W. (1973). “The Programmer as Navigator”, ACM Turing Award lecture, 16(11), Communications of the ACM, 653-658.
- Björk, B. C. (1999). “Information Technology in Construction – domain definition and research issues”, International Journal of Computer Integrated Design and Construction, 1(1), SETO, London, 1-16.
- Bjork, B.C., and Penttila, H. (1989). “A Scenario for the Development and Implementation of a Building Product Model Standard”, Advances in Engineering Software, 11(4), 176-186.
- Boddy, S., Rezgui, Y., Cooper, G. and Wetherill,, M. (2007). “Computer integrated construction: A review and proposals for future direction”, Advances in Engineering Software, 38(10), 677-687.

Booch, G., Maksimchuk, R. A, Engel, M. W., Brown, A., Conallen, J. Houston, K. A. (1994), Object-Oriented Analysis and Design with Applications, Addison Wesley, Reading, MA.

buildingSMARTalliance Website, National BIM Standard™ – available at:
<http://www.buildingsmartalliance.org/nbims/> (Accessed July 15, 2007)

Cheng, J., Trivedi, P., and Law, K.H. (2002). “Ontology Mapping between PSL and XML-Based Standards For Project Scheduling”, In Proceedings of the 3rd International Conference on Concurrent Engineering in Construction, Berkeley, CA, 143-156.

CIS/2 Website, CIMsteel Integration Standards Release 2 – available at:
<http://www.cis2.org/>. (Accessed June 15, 2006)

Codd, E.F. (1970). “A Relational Model of Data for Large Shared Data Banks”, Communications of the ACM, 13 (6), 377–387.

Crowley, A., and Watson, A. (2000). “CIMsteel Integration Standards Release 2”, SCI-P-268, The Steel Construction Institute, Berkshire, England.

Date, C. J., and Darwen, H. (2000). Foundation for Future Database Systems: The Third Manifesto, 2nd Ed., Addison-Wesley Professional, Reading, MA.

Date, C. J. (2003). Introduction to Database Systems. 8th Ed., Addison-Wesley, Reading, MA.

Eastman, C., Wang, F., You, S.J., and Yang, D. (2005). “Deployment of an AEC Industry Sector Product Model”, Computer-Aided Design, 37-12, 1214-1228.

Eastman, C. (1999). “Building Product Models: Computer Environments Supporting Design and Construction”, CRC Press, Boca Raton, FL.

Eastman, C. (1979). “The representation of design problems and maintenance of their structure”, in: Latcombe (Ed.), Application of AI and PR to CAD, IFIPS Working Conference, Grenoble, France, North-Holland, Amsterdam, 335–337.

Fahdah, I. (2008). Distributed IT for Integration and Communication of Engineering Information for Collaborative Building Design, Ph.D. Dissertation, University of Nottingham, Nottingham, UK.

Fisher N., Barlow R., Garnett N., Finch E., and Newcombe R. (1997): “Project Modelling in Construction”, Thomas Telford Ltd, London.

Froese, T. (2003). “Future directions for IFC-based interoperability”, Journal of ITCON, 8, 231-246 – available at : <http://www.itcon.org/2003/17> (Accessed December 14, 2006).

Gallaher et al. (2004). “Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry”, National Institute of Standards and Technology (NIST).Gaithersburg, MD.

Gieling, W. (1988). “General AEC reference model. ISO TC 184/SC4/WG1doc. 3.2.2.1”, TNO report BI-88-150. Delft, the Netherlands.

Hakkinen, T.M. (2007). “Sustainable building related new demands for product information and product model based design”- available at:
http://www.itcon.org/data/works/att/2007_2.content.06036.pdf (Accessed June 23, 2008)

Halfawy, M., Pouria, A., Foese, T.(2002). “Developing message-based interoperability protocols for distributed AEC/FM Systems”, In Proceedings of the CIB w78 conference- available at: <http://itc.scix.net/data/works/att/w78-2002-61.content.pdf>. (Accessed May 15, 2008)

Halfawy, M., and Froese, T. (2001). “Leveraging information technologies applications in the Canadian AEC/FM Industry”, In the Conference Proceedings of the Canadian Society of Civil Engineers, Victoria, BC, Paper A22.

Harrington, J. L. (2000). Object-oriented Database Design Clearly Explained. Edition: illustrated, Morgan Kaufmann Publishers Inc., San Francisco, CA.

Howard, R., Bjork, B.C. (2007). “Building information modeling – experts’ views on standardization and industry development”, Journal of Advanced Engineering Informatics, 22 (2), 271–280.

IAI - International Alliance for Interoperability Website, Industry Foundation Classes- available at: <http://www.iai-tech.org/>. (Accessed July 12, 2006)

ISO 10303-42. (2000). “Industrial automation systems—Product data representation and exchange—Part 21: Integrated generic resource: Geometric and topological representation”, ISO/IEC, Geneva, Switzerland (with Technical Corrigendum 1, 2001).

Kamat, V.R., and Lipman, R.R. (2006). “Evaluation of Standard Product models for Supporting Automated Erection of Structural Steelwork”, Journal of Automation in Construction, 16, 232-241.

Kamfonas, M. J. (1992). Recursive Hierarchies: The Relational Taboo! , The Codd and Date's Relation Journal, 4(5), 10.

Kiviniemi, Arto (2005a). “Integrated product models in life-cycle evaluation and management of the built environment”, In: Proceeding of the International workshop on lifetime engineering of civil infrastructure, Ube, Yamaguchi, Japan.

Kiviniemi, Arto (2005b). "Requirements Management Interface to building product models", Ph.D. Dissertation and CIFE Technical Report TR161, Stanford University – available at: <http://cife.stanford.edu/online.publications/TR161.pdf> (Accessed March 23, 2008).

Kroenke, D.M. (1997). Database Processing: Fundamentals, Design, and Implementation, Prentice-Hall, Upper Saddle River, NJ.

Lipman, R.R., K.A. Reed. (2003). "Visualization of Structural Steel Product Models", Electronic Journal of Information Technology in Construction, 8, Royal Institute of Technology, Stockholm, Sweden, 43-50.

Loney, K. 2004. Oracle Database 10g: The Complete Reference, Osborne, McGraw-Hill, New York, NY.

Niemioja, Seppo (2005). "Architect's product model based design, Guidelines. ProIT Product model data in construction process", Confederation of Finnish Construction Industries RT., 57 (In Finnish).

Reed, K.A. (2002). "Role of the CIMsteel integration standards in automating the erection and surveying of constructional steelwork", 19th International Symposium on Automation and Robotics in Construction (ISARC), NIST, Gaithersburg, MD, 15–20.

Saidi, K.S., Lytle, A.M., Scott, N.A., and Stone, W.C. (2005). "Developments in automated steel construction", NISTIR 7264, National Institute of Standards and Technology, Gaithersburg, MD.

STEP Tools, STEP ISO10303, STEP Tools Inc. - available at: <http://www.steptools.com/library/standard/2007> (Accessed July 8, 2008)

Stonebraker, M., Brown, P., and Moore, D. (1999). Object-Relational DBMSs: Tracking the Next Great Wave, 2nd Ed., Morgan-Kaufman Publishers. San Francisco.

Tse, K., Wong, A., Wong, F., Leung, Y., and Riese, M. (2006). "Building Information Modelling – a case study of building design and management", In: Proceedings of the 31st AUBEA Conference, University of Technology, Sydney.

Turner, J. (1990). "AEC Building Systems Model, ISO TC184/SC4/WG1", Doc. N363. Working paper, University of Michigan, Ann Arbor, MI.

Ullman, J.D., and Widom, J. (2002). A First Course in Database Systems, Prentice Hall, Upper Saddle River, NJ.

Ullman, J.D., and Widom, J. (1997). A First Course in Database Systems, Prentice Hall, Upper Saddle River, NJ.

Wix, J., Liebich, T., Karud, O.J., Bell, H., Häkkinen, T., and Huovila, P. (2007). “Integration of Performance based building standards into business processes using IFC standards to enhance innovation and sustainable development”, Guidance Report: IFC Support for Sustainability, EU project 2006-2008 under “Structuring the European Research Area”.

XML Cover Pages Website. “aecXML Working Group - Architecture, Engineering and Construction” – available at : <http://xml.coverpages.org/aecXML.html> (Accessed July 9, 2008)

Chapter 5

User Tracking in Outdoor and Indoor Construction Environments

5.1. Introduction

In recent years, the need for localization and tracking have been rapidly expanding in many fields (Kuusniemi and Lachapelle 2004), and currently offers significant potential of improving manual processes and supporting important decision-making tasks in the construction field (Kirisci et al. 2004) by allowing rapid access to a wealth of project information.

Considering the dynamic nature of typical construction projects, mobile users (e.g. construction engineers, inspectors, etc.) need to be constantly tracked outdoors as well as indoors. By capitalizing on the ability to accurately track mobile users in any indoor and/or outdoor environment, the presented research has designed and implemented a

dynamic spatial context-sensing framework that can allow the identification of construction entities and artifacts visible in a user's field of view at any given time and location, as well as contextual information retrieval (Chapter 2). A context-aware data delivery and information retrieval framework retrieves and displays graphical and textual data to the users based on their latest position, line of sight and the level of detail they request. As a result, tracking a user's fully-qualified spatial context is a crucial task in almost all applications designed to retrieve and deliver context aware information with high-precision on a continuous basis, such as construction applications.

However, as noted in previous chapters, in order to interpret a site engineer's or an inspector's fully-qualified spatial context, another parameter in addition to the position is required. This parameter is the user's three-dimensional head orientation, which is the direction in which the user is looking. It is defined by three angles named yaw, pitch, and roll that can be described using the same notation typically used to define the orientation of an aircraft in flight (Figure 5.1).

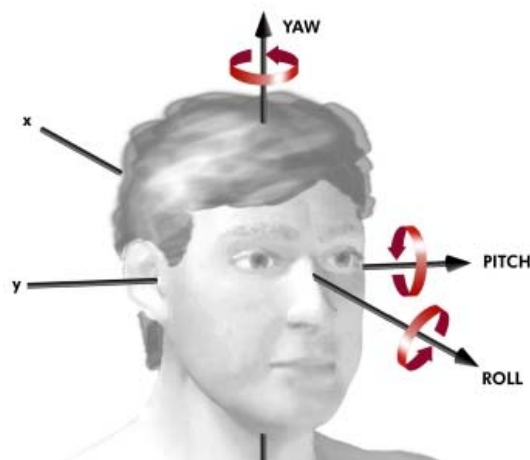


Figure 5.1– Yaw, Pitch, and Roll Angles

As shown in Figure 5.1, the yaw represents the rotation in the horizontal plane, pitch is the rotation in the vertical plane parallel to the forward direction, and roll is the rotation in the vertical plane perpendicular to the forward direction.

In this chapter, the framework's first component, ubiquitous user tracking scheme, is presented. This scheme encompasses positioning as well as orientation tracking devices for tracking a user's spatial context parameters in outdoor and indoor environments.

For outdoor applications, positioning techniques have been investigated and validated in recent work reported in Behzadan and Kamat (2007). The outdoor positioning technologies were integrated within an outdoor AR platform (UM-AR-GPS-ROVER). The hardware configuration consists of a georeferencing based algorithm developed using the Global Positioning System (GPS) and magnetic orientation tracking devices to track a user's dynamic viewpoint (Figure 5.2).



Figure 5.2 – Outdoor Hardware Prototype

As mentioned in Chapter 3, the orientation tracker, a TCM5 magnetic device, includes a built-in compass, and employs solid-state magnetic field sensors which measure compass heading through a full 360 degrees of rotation. The tracker employs proprietary hard and soft iron correction algorithms to calibrate out magnetic anomalies for repeatable, high-resolution measurement in challenging environments (PNI 2008). The tracking device was placed at the highest point inside the user's helmet, directly above the top of the head, and parallel to the forward line of sight (Figure 5.2).

The GPS measures the user's position as longitude (x), latitude (y), and altitude (z). The magnetic tracker, on the other hand, measures the orientation of the user's head (and thus line of sight) in the form of yaw (α), pitch (β), and roll (γ) angles. These six measurements fully define the user's outdoor location and line of sight at any given time.

However, GPS technology is not suitable for indoor applications because it becomes ineffective when there is no continuous straight signal path between the satellites and a receiver. Therefore, other feasible techniques of user position and orientation tracking in indoor enclosed environments had to be investigated. The author thereby explored the applicability of wireless technologies, namely Wireless Local Area Networks (WLAN), Ultra-Wide Band (UWB), and Indoor GPS for dynamic user position tracking in situations where GPS is unavailable. By tagging users with appropriate receivers/tags and deploying a number of nodes (access points, receivers, transmitters, etc.) at fixed positions indoors, the location of tagged users can conceptually be determined and continuously tracked.

The objective of this chapter is to describe three key wireless technologies applicable for indoor positioning, portray and compare the technical characteristics of these technologies, and highlight the extent to which each technology can be used to accurately calculate the positional context of a user in congested dynamic environments such as those found on construction sites.

While these technologies are capable of streamlining tracking, data access and retrieval processes, their performances differ from the manufacturers' specifications when utilized on construction sites due to issues, such as interference, data reading range, data accuracy, interoperability of hardware and software, and memory limitations (Kiziltas et al. 2008). In addition, although these technologies can save time and effort, they can also add new tasks that need to be performed prior to, during, or after the utilization of a technology in the field. Hence, a thorough understanding of both the technological capabilities and process implications of these technologies is needed to be able to utilize them effectively during construction, inspection and maintenance of facilities.

5.2. Technical Overview on Indoor User Position Tracking Technologies

5.2.1. WLAN-based User Position Tracking

A WLAN is a flexible data communication system implemented as an extension to or as an alternative for a wired LAN within a building or an area. The technology mainly

consists of a set of access points communicating wirelessly, using electromagnetic waves, to any WLAN-enabled device (Figure 5.3).

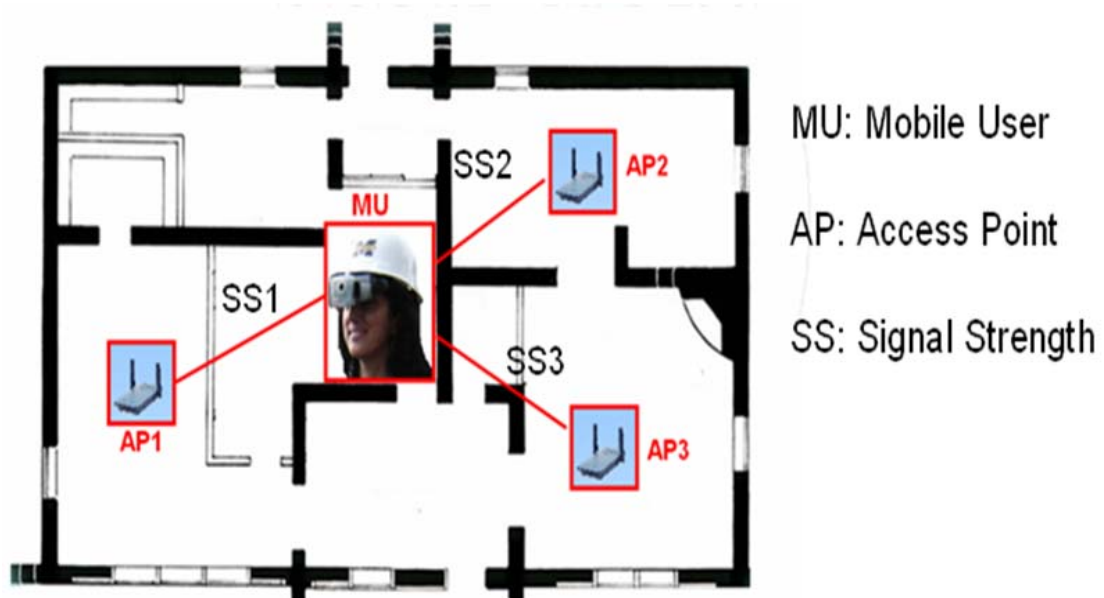


Figure 5.3 – WLAN Technology

One set of conducted experiments (Chapter 8) to obtain location information in this study was based on a WLAN based position system called Ekahau manufactured by the Finnish company Ekahau Inc. (Ekahau Website 2007). The underlying approach used for determining users' position in the Ekahau tracking system is the fingerprinting technique (Figure 5.4).

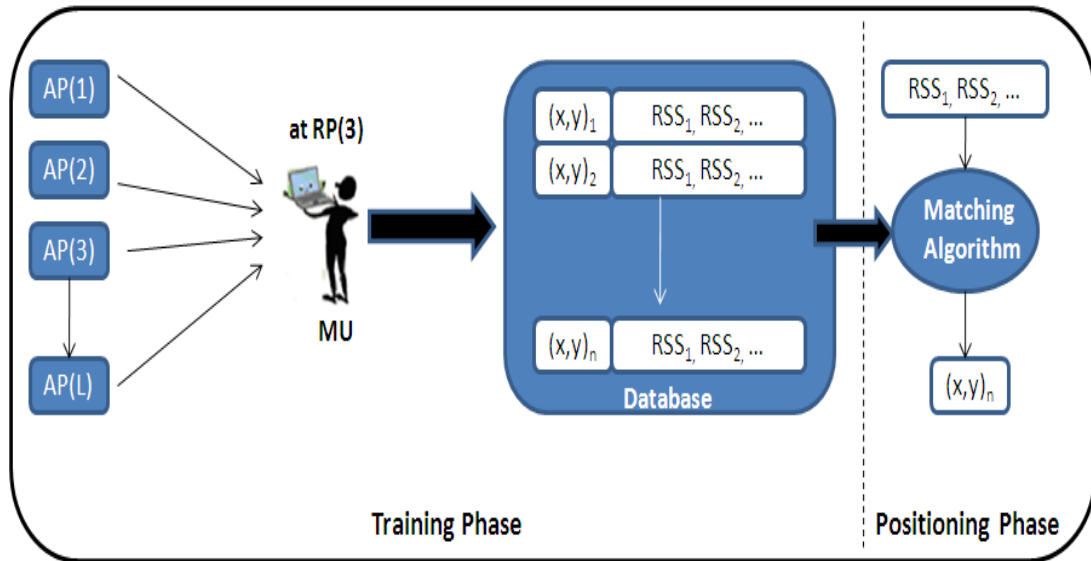


Figure 5.4 – Fingerprinting Approach

As shown in Figure 5.4, location fingerprinting consists of two phases: ‘training’ and ‘positioning’ (Li et al. 2006). The objective of the training phase is to build a fingerprint database. In order to generate the database, reference sample points (RPs) must first be carefully selected. Locating a mobile user (MU) at one RP location, the Received Signal Strengths (RSS) of all the access points (AP) are measured. From such measurements the characteristic feature of that RP (its RSS) is determined, and is then recorded in the database. This process is repeated at another RP, and so forth until all RPs are visited. In the positioning phase, the MU measures the RSS at a place where it requires its position. The measurements are compared with the database using an appropriate search/matching algorithm. The outcome is the likeliest location of the MU.

In the process of determining mobile users’ location using the aforementioned fingerprinting technique, the Ekahau tracking system relies heavily upon many

components, namely the Ekahau Client, the Ekahau Positioning Engine (EPE), the Ekahau Manager, and the Ekahau Application Suite (EAS) (Figure 5.5).

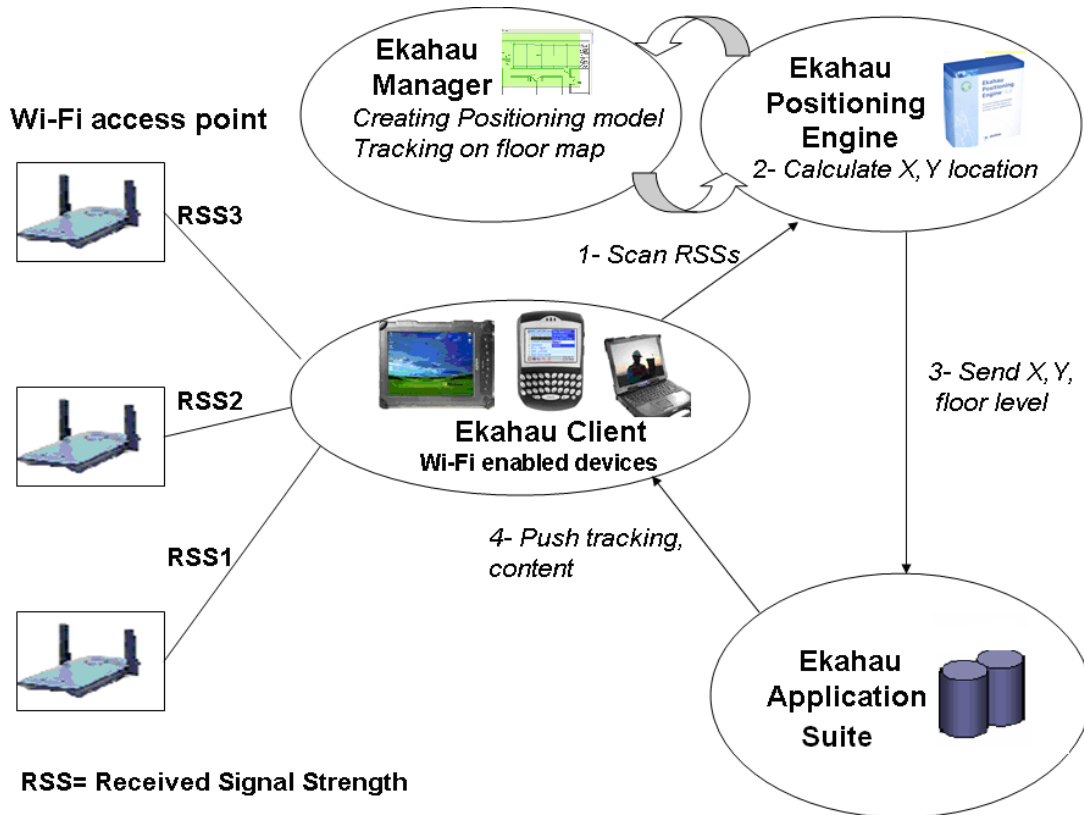


Figure 5.5 – Ekahau Tracking System Components

The Ekahau Client is a free downloadable piece of software that is responsible for sending the recorded Received Signal Strength (RSS) WLAN signals to the EPE. These signals play a major part in determining users' position and are also used by the Ekahau Manager application to do *SiteCalibration*.

The Ekahau Manager is mainly responsible for the patented process called *SiteCalibration* (Figure 5.6).

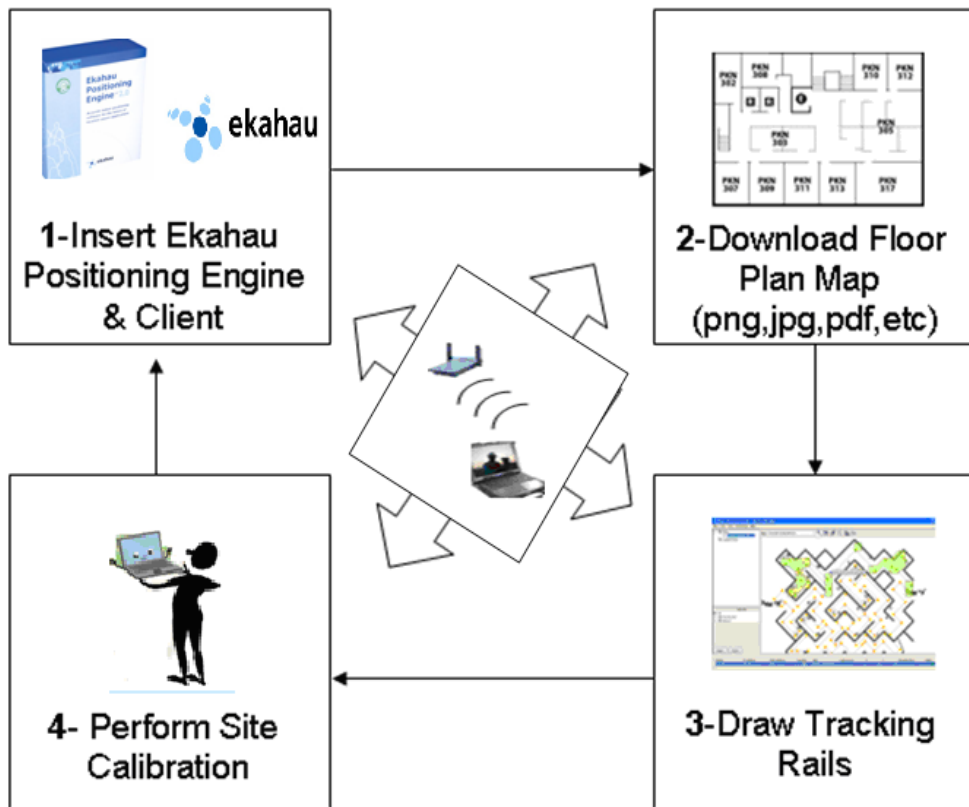


Figure 5.6 – Ekahau Deployment and Calibration

First, it creates a model of the desired area in the form of an accurate floor map image (can be in JPG, PNG or BMP format). Then, *tracking rails* are drawn on top of the map to symbolize mobile users' walking paths. In order to turn this final version of the map into a positioning model, Ekahau Manager requires explicit calibration according to the training phase of the fingerprinting technique (sample points are recorded at many RPs in the walking paths). The floor map used by the Ekahau Manager can be seen as an implicit location model when calibrated as a positioning model. The map does not show hierarchical or containment relationships between objects in a building, but is used to create a coordinate system of each floor area.

The Ekahau Manager can also perform various analyses of the WLAN signals it receives and process them to provide instant live tracking of a connected device.

The EPE, being the centerpiece of the Ekahau Tracking System, is responsible for keeping track of all the devices, and updating their position based on the information it continuously receives from them. The EPE is a 2.5 dimensional positioning solution. It cannot give a complete 3D coordinate of the estimated position. What it can compute is mainly the 2D coordinates (x, y) and the floor the device is on.

The EAS is a set of helpful tools, examples, and easy programming interface for third-party applications to quickly utilize EPE location information.

Based on all the aforementioned system components, the Ekahau tracking process operates as follows:

First, a model of the desired space is created using Ekahau Manager. Then, areas/rooms are scanned and Radio Frequency (RF) parameter measurements obtained at different RPs using Ekahau Client (power loss, multipath phase, etc.) are recorded. The measurements with their location are then saved to a database/EPE (i.e. Received Signal Strengths, RSS indicate power loss, and this loss is translated into location).

Having the positioning model created and calibrated, the second phase of the fingerprinting approach, positioning comes into play. When a WLAN-enabled mobile device/user moves in the area, its RF parameter measurements are reported to the EPE.

The device/user location is estimated by matching the RF parameters against the location fingerprints in the database. The software uses patented algorithms and scene analysis on the signals to compute a location estimate. The whole process results in a positioning estimate that can be as accurate as 1 to 2 meters under optimal conditions (LaMarca et al. 2005).

5.2.2. UWB-based User Position Tracking

The second tracking system studied in this research is the Sapphire *DART* Ultra-Wide Band (UWB) *Digital Active Real Time Tracking* system (Multispectral Solutions Website 2007). It is designed for the tracking of personnel and/or equipment. A system is defined as one processing hub, four or more receivers, one or more reference tags, and multiple tags for individual assets (Figure 5.7).

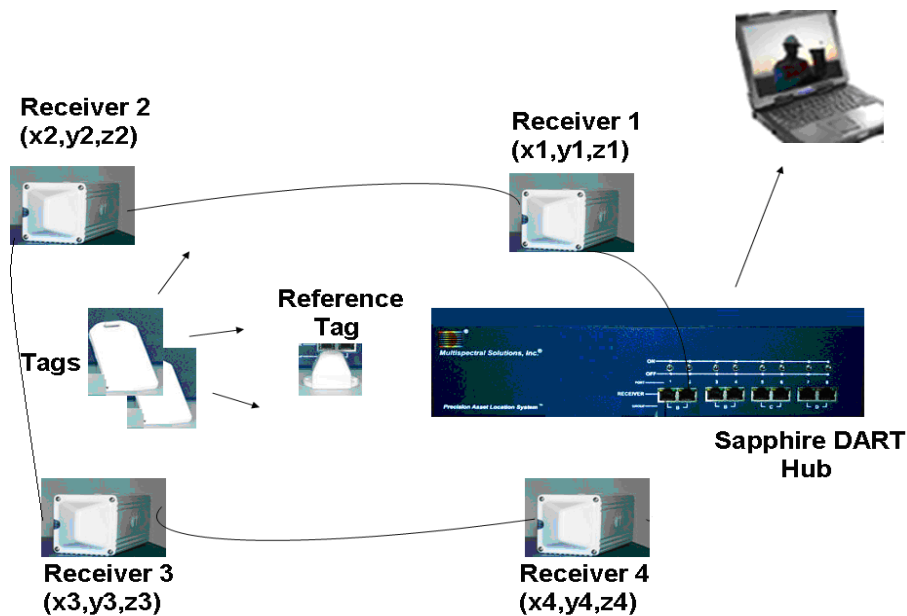


Figure 5.7 – Sapphire UWB Tracking System

The system uses short pulse or UWB technology to determine the precise location of UWB radio frequency identification (RFID) tags and operates as follows:

Each tag repeatedly sends out a packet burst consisting of a short train of UWB pulses, each pulse having an instantaneous bandwidth of over 1 GHz. Since individual tags are not synchronous, and the packet bursts are of extremely short duration, the probability of tag packet collision is very small allowing for the simultaneous processing of hundreds to thousands of tags in a local area.

These transmitted UWB pulse trains are received by one or more *Sapphire DART* UWB receivers which are typically located around the periphery of the area of coverage at known locations. Reception by three or more receivers permits accurate 2D localization, while reception by four or more receivers allows for precise 3D localization. Each receiver uses a highly sensitive, very high speed, short pulse detector to measure the precise time at which a tag packet arrives at its antenna. The extremely wide bandwidth of the UWB pulses permits the receivers to measure these times-of-arrival to sub nanosecond precision.

In order to determine the actual tag position from these measurements, the *Sapphire DART* Hub/Processor, using calibration data from the *Sapphire DART* UWB reference tag, determines the differential times-of-arrival (DTOA) between receiver pairs from these individual receiver measurements and implements an optimization algorithm to determine the location using a multilateration or hyperbolic positioning technique (Figure 5.8). In general, N receivers provide $N-1$ halves of two-sheeted hyperboloids which are

approximated by hyperbolas. In this case, given four receivers, the tag is located along three DTOA hyperbolas (i.e. $DTOA_{1-2}$, $DTOA_{1-3}$, and $DTOA_{1-4}$) at their intersections. Since the speed of light is approximately 0.98 feet per nanosecond, these DTOA are readily converted into the appropriate measurement distances (Multispectral Solutions Website 2007) and the tag location is determined.

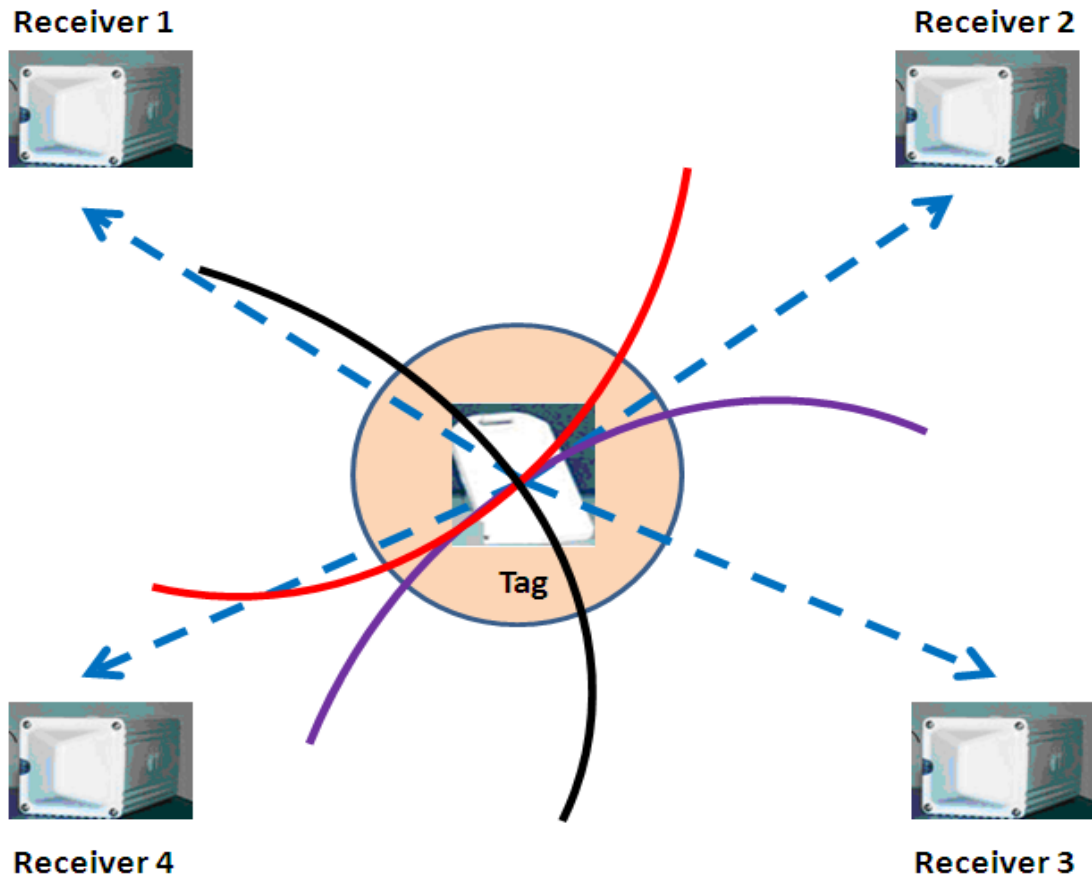


Figure 5.8 – Multilateration Approach

The outputs resulting from a UWB tracking system application are provided from the hub to the client machine in the following format:

`<Data Header>, <tag #>, <X>, <Y>, <Z>, <battery>, <timestamp>, <unit><LF>`

“Data Header” represents the tag dimensional information. There are many expected values for the data header but the one of interest in this research is R which reflects the 3D calculation for x , y , and z . “tag #” is the tag ID. “X, Y, Z” are the calculated tag coordinates in feet or meters with respect to a user supplied origin. “Battery” is the tag’s low battery indicator (range value from 0-15, where 15 represents a fully charged battery). “Timestamp” represents the hub system time. “Unit” is a Virtual Group ID. The tag location data is computed from the time of flight measurements of the receivers within the virtual group. “LF” is a Line Feed character (with ASCII code = 0x0A), to terminate a location data string. Among all the output information, the “Data Header”, “tag#” and “X, Y, Z” are of primary importance for the purpose of this research.

5.2.3. Indoor GPS-based User Position Tracking

Indoor GPS (Metris Website 2007) is the third tracking system studied in this research. The system is mainly defined by transmitters and receivers (Figure 5.9).



Figure 5.9 – Indoor GPS Transmitter (left) and Receiver (right)
[Images courtesy of Dr. Kamel Saidi, NIST]

A battery operated transmitter uses laser and infrared light to transmit one-way position information and elevation to the receiver. With the addition of a second transmitter of known location and orientation, users can calculate the position of the receiver in the base coordinate system. By adding two more transmitters, the system can have four laser transmitters having its accuracy maximized. The GPS-like navigation signal is transferred through a wireless network connection providing mobility to the operator (Kang and Tesar 2004). As in satellite-based GPS, this one-way signal path is created from transmitters to the receiver, allowing an unlimited number of receivers to continuously and independently calculate positions whenever two or more transmitters are in view (Figure 5.10).

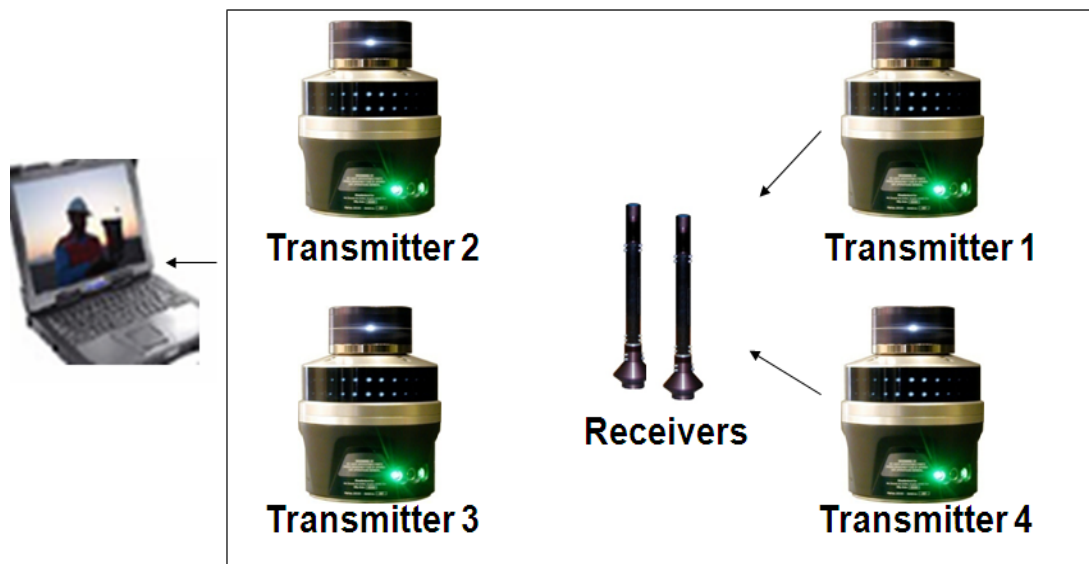


Figure 5.10 – Indoor GPS Tracking System

A receiver in the measurement volume, having photodiodes inside its module, detects and processes the laser and infrared light signals from each visible transmitter. The 3D position of the optical receiver is then calculated by the process of triangulation (Figure

5.11). Triangulation (Lähteenmäki et al. 2001) is used, if the angles to known locations are given. The two angles (α and β) are used to determine the line-of-sights to each of the known locations. These lines are unique in the two-dimensional space and intersect in the desired position.

Therefore, given the angular information from at least two transmitters, and provided with the position and orientation of each transmitter, a unique 3D position within the measurement volume can be calculated. The indoor GPS eliminates the recurring problem of accidentally interrupting a laser beam during measurement that requires the operator to begin the measurement again.

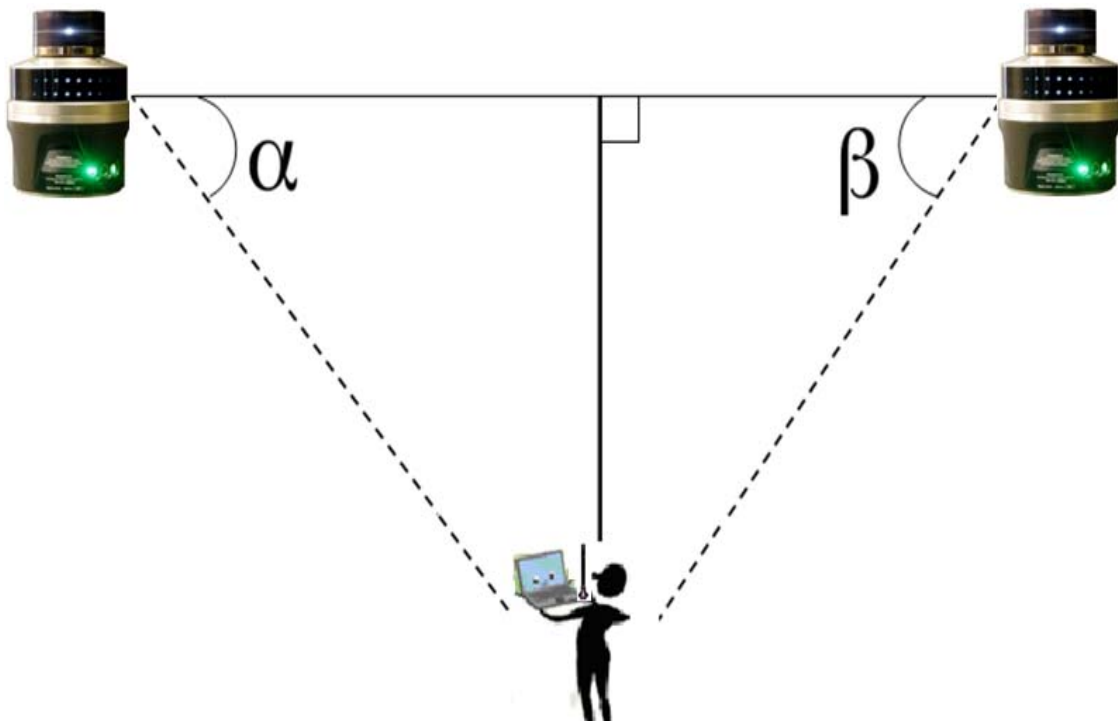


Figure 5.11 – Triangulation Approach

Additional details on setting up and using the WLAN-based Ekahau system, UWB and Indoor GPS with all their different components are presented in Appendix A. The reader is recommended to refer to this appendix for additional details on each technology.

5.3. Mechanisms of a User Viewpoint Tracking Application

In this research, the objective of the first component of the proposed methodology was to track both the mobile user's position and head orientation in real-time. However, positioning information is obtained from one source, i.e. the three different indoor positioning technologies described above, and orientation information is obtained from another source, i.e. a software process communicating with a magnetic orientation tracker (Behzadan and Kamat 2006). Therefore, there was a need to extract location information and then combine it with the orientation information within the same application.

5.3.1. WLAN-based Position and Head Orientation Tracking Application

In this subsection, positioning information (X, Y, floor level) is obtained from the Ekahau tracking system (Ekahau Website 2007) and orientation information (roll, yaw, and pitch) from the magnetic orientation tracker (Behzadan and Kamat 2006).

In order to extract location information, part of the Ekahau solution, the EAS (Figure 5.5) containing different examples and client applications, is used. It is constantly fed by the

Ekahau Positioning Server with position data. One of the ways to serve location information to the EAS is to use a Java Software Development Kit (Java SDK) (Ekahau Website 2007). The Ekahau Java SDK utilizes TCP sockets to connect to the Positioning Engine and provides quick and effective way for accessing location information, either from a local or remote computer. Using the SDK (Figure 5.12) requires a working knowledge of the Java programming language and Java 2 Platform 1.4.2.

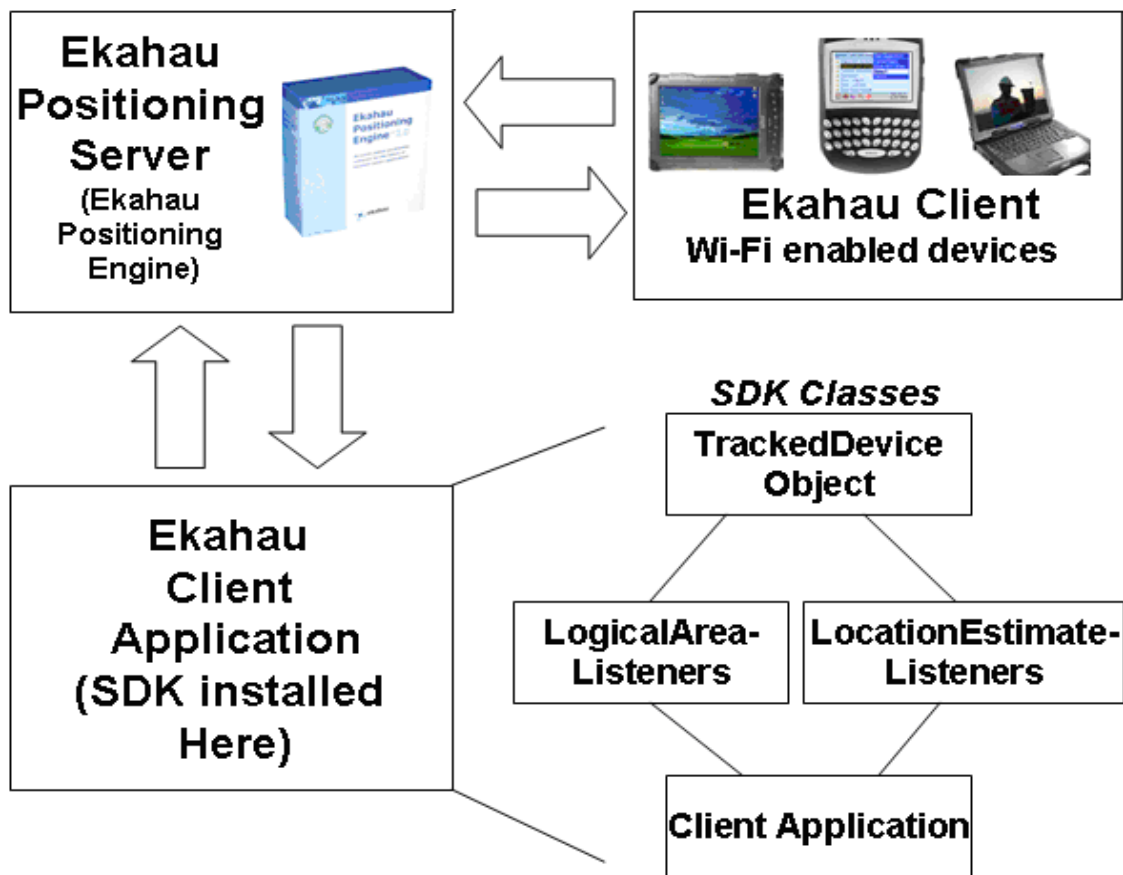


Figure 5.12– Using the Ekahau SDK Environment

The TrackedDevice class is the key class for Ekahau Java SDK's functionality. The Positioning Engine's TrackedDevice objects represent wireless devices, so one object needs to be created for each physical device which is to be tracked. After recording a Positioning Model and saving it in the Positioning Engine with Ekahau Manager, TrackedDevice objects are used to return the coordinates, timestamp, status, speed, map, and any "logical area" information. The client receives the information via two main kinds of Listener classes. The Listener interfaces are LocationEstimateListener used to obtain automatic location updates, and StatusListener used to get information about the device's state, whether it has been detected or not. Figure 5.13 presents pseudo code reflecting one of the Ekahau ready examples or client applications (*SimpleTrackExample.java*) using the JAVA SDK to obtain the 2D coordinates of a device/user (x, y coordinates).

- Connect to Positioning Engine on port 8548 using connect()
- Create a TrackedDevice instance using a Device Object
 1. Create a Device Instance
 2. Retrieve a Device Object using FindDevice method of the PositioningEngineClass
 3. Create a TrackedDevice Object
- Add a LocationEstimateListener for a TrackedDevice object
 1. LocationEstimateListener receives LocationEstimate objects
 2. LocationEstimate contains location : getLatestLocation()
 3. Use addLocationEstimateListener method and receive automatic location updates
- Add a status Listener for a TrackedDevice object
 1. Use addStatusListener method to get the Tracked Device's State
- Start tracking by calling the setTracking(true) method
- Get all location values
- Stop Tracking by calling the setTracking(false) method
- Disconnect from Positioning Engine using disconnect()

Figure 5.13 – Pseudo Code of the Client Application *SimpleTrack.java* using JAVA SDK

Knowing that positioning information (x, y, floor level) is directly accessible using JAVA SDK (Ekahau Website 2007), the next step is to combine it with the orientation information (roll, yaw, and pitch) in a single application as reflected in the pseudo code shown in Figure 5.14 and in Figure 5.15.

- Initialize C++ tracker application to start obtaining head orientation values (roll, pitch and yaw)
- Get positioning values
 1. Open pipe to invoke Ekahau JAVA SDK and Return pointer to a stream using popen ()
 2. Read output (x, y, Floor level) of the invoked program using the reading access character string mode “r”
- Output all positioning and head orientation values (x, y, Floor level and roll, pitch, yaw)
- Close pipe using _pclose()

Figure 5.14 – Pseudo Code for Creating a Pipe between Two Tracking Applications

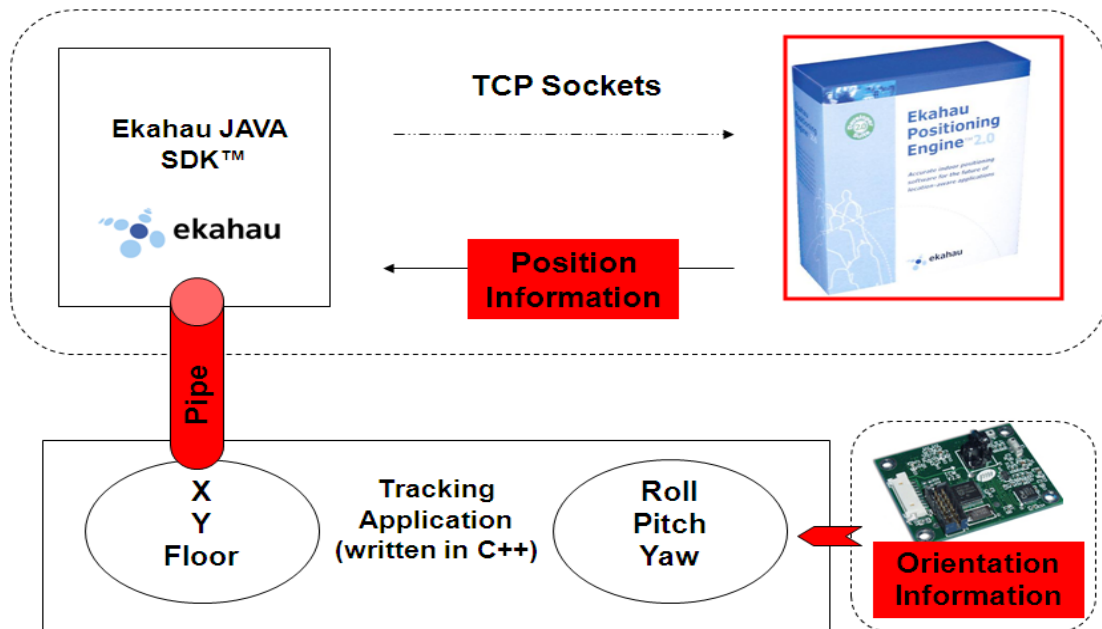


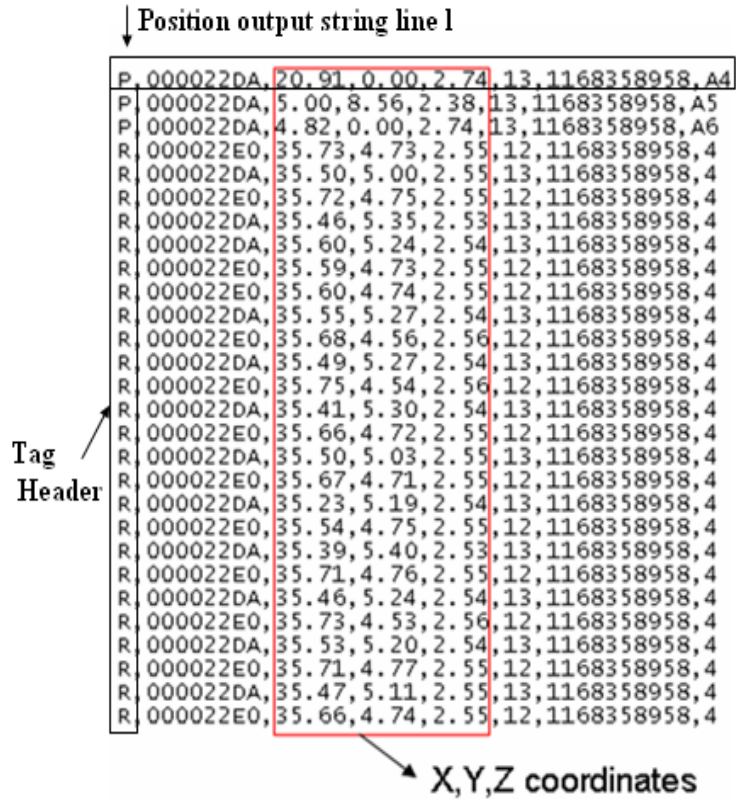
Figure 5.15 – Creating a Pipe between Ekahau-based Position and Head Orientation Tracking Applications

This was achieved by creating a “pipe” between the JAVA application communicating with Ekahau and the C++ application communicating with the magnetic orientation tracker.

5.3.2. UWB-based Position and Head Orientation Tracking Application

Similar to the WLAN-based tracking application, location information (X,Y,Z) was extracted from one positioning technology, UWB (Multispectral Solutions Website 2007) and then combined with the orientation information (roll, yaw, and pitch) received from the tracker within the same application.

TCP sockets were used to connect directly to the Hub and provide a quick and effective way for accessing location information from a local computer/laptop. This was achieved by opening a socket connection given the IP address of the hub and the port number, reading values (Figure 5.16a.) according to the *Sapphire* output format defined in Section 3.3.2 and then performing a string manipulation on the values extracted to obtain the x, y, and z coordinates (Figure 5.16b.). Only values corresponding to a tag header **R** were extracted (n1, n2, and n3).



Open socket connection "s" to Sapphire HUB (IP address and port number)

- Receive each position output string l from s
- If (l contains tag header R meaning 3D readings X,Y,Z)
 - Do a string manipulation and extract array s[i], a set of 3 position coordinates from each string l
 - Convert the array of position strings s to array of position integers f and Assign n1, n2, n3 to each of the position integer values f

for(i=0 → i=3)

string s[i] → f[i]

n1=f [1], n2=f [2] & n3= f [3]

Figure 5.16 – a) Output Results from Sapphire HUB (top), b) Pseudo Code to Extract UWB Position Coordinates (bottom)

The positioning values were then used together with orientation values received from the magnetic tracker and both were integrated in one application to retrieve tracking information (Figure 5.17).

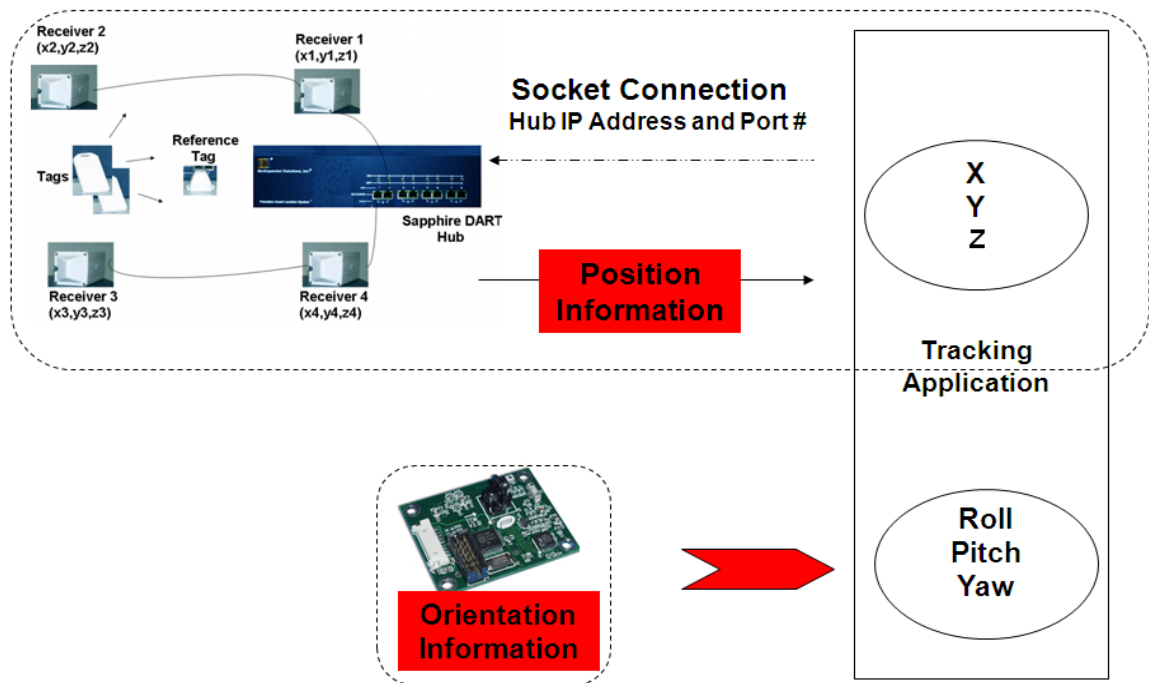


Figure 5.17 – Retrieving Positioning and Orientation Information from UWB Sapphire HUB and Tracker

5.3.3. Indoor GPS-based Position and Head Orientation Tracking Application

Similar to the previous tracking applications, the user's position and orientation were continuously obtained from two different sources, in this case the Indoor GPS and magnetic tracker.

TCP sockets were used to connect directly to the Indoor GPS system application and provide a quick and effective way for accessing location information from a local computer/laptop. This was achieved by opening a socket connection given the IP address and the port number, reading values in byte string format, performing a byte manipulation on those values and extracting the x, y, and z coordinates (Figure 5.18).

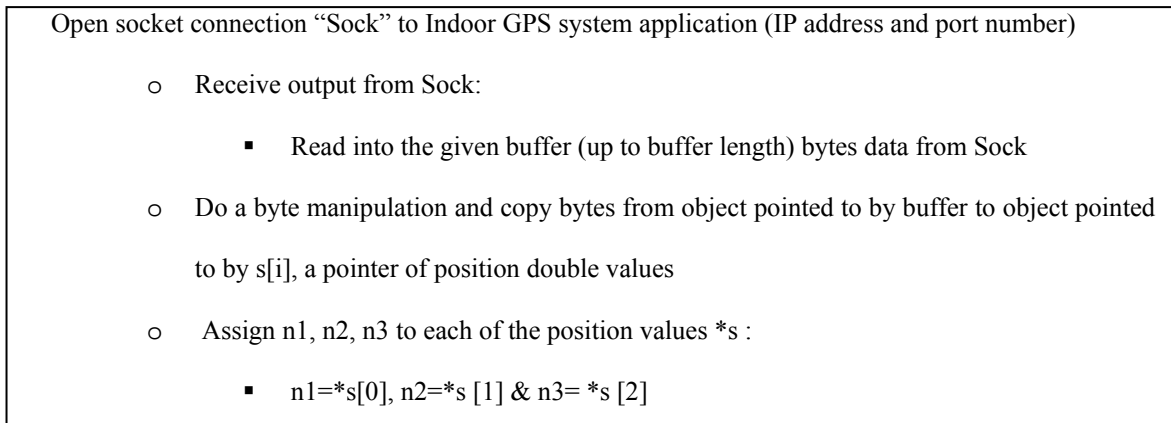


Figure 5.18 –Pseudo Code to Extract Indoor GPS Position Coordinates

The positioning values were then used together with orientation values received from the magnetic tracker and both were integrated in one application to retrieve tracking information (Figure 5.19).

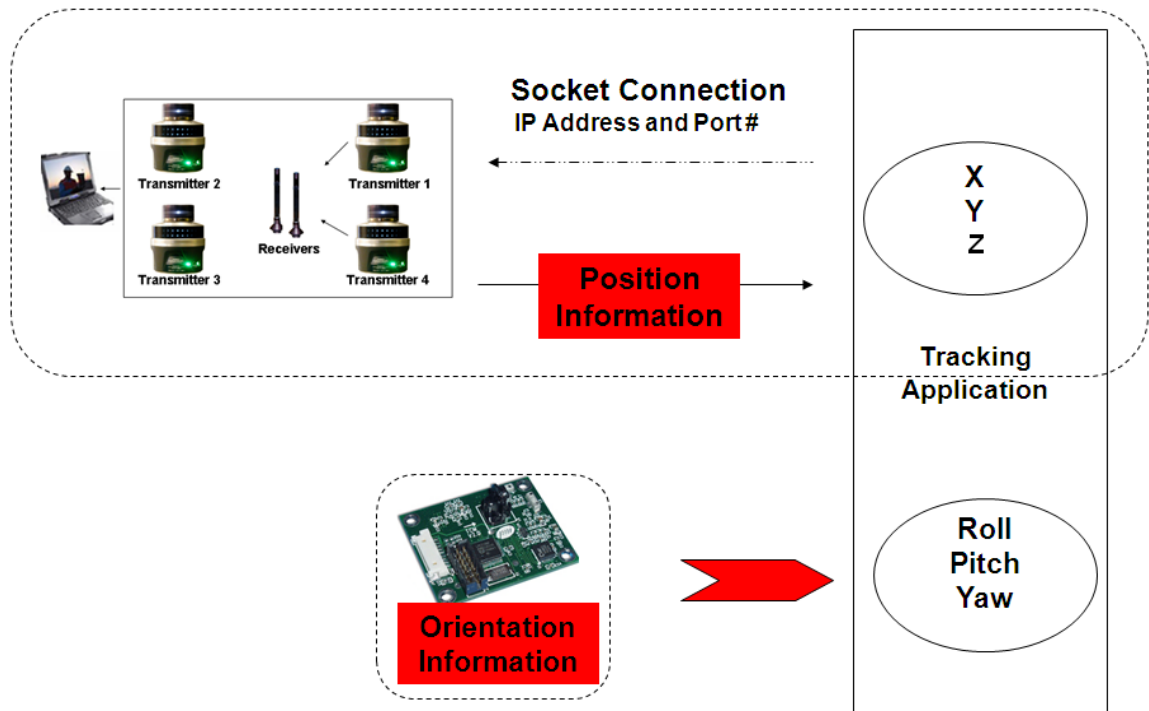


Figure 5.19 – Retrieving Positioning and Orientation Information from Indoor GPS and Tracker

5.4. Tracking Systems Comparative Summary

The discussion in the previous sections highlighted the potential applicability of positioning technologies, mainly WLAN, UWB and Indoor GPS, for positioning in indoor environments.

In order to demonstrate their feasibility for position tracking and compare their technical characteristics, experiments were conducted in several indoor environments. These experiments are described in detail in Chapter 8.

While these technologies share some common traits, they also have some significant differences based on an analysis of their technological aspects (e.g. line of sight requirement), as well as implementation details (calibration, equipment deployment, cost, etc.). The major differences are summarized in Table 5.1.

Table 5.1 – Comparative Summary of Indoor Positioning Technologies

	Line of Sight	Position Uncertainty	Range	Calibration	Deployment and Cost
Indoor GPS	Needed (receiver-transmitter)	Very Low (1-2 cm)	60 m	Needed (few sampling points)	Quite Easy but Very Expensive
UWB	Needed (receiver-reference tag)	Low (10-50 cm)	10 m	Not needed	Quite Easy but Expensive
WLAN (Ekahau)	Not needed	Medium (1.5-2 meters)	10-100m	Needed (time-consuming)	Easy and Economical

For instance, WLAN-based tracking systems such as Ekahau are economical and equipment deployment mainly consists of placing access points in the tracked area. However, the area needs to be calibrated first (several sample points are required at different locations) which is an arduous and often challenging task, in particular in dynamic construction environments. Although the range of a typical 802.11 b WLAN node is 100 m (Wang and Liu 2005), the technology does not provide the desired accuracy (1.5 to 2m) needed to locate mobile users and identify their spatial context with high-precision.

On the other hand, UWB and Indoor GPS require significant time and effort to deploy all required stations around the coverage areas (Appendix A), in particular on dynamic construction sites. Additionally, both technologies are relatively expensive. For instance, a full UWB system with 4 receivers (any antenna type), one processing hub, four cables (150') and eleven tags cost about \$15,000. Individual receivers (any antenna type) are \$2,195 each. The hub costs around \$5,195 and individual 1Hz tags are \$40 each, and higher power tags range from \$120 to \$125 each. An Indoor GPS system with four transmitters and 1 receiver costs up to \$ 45,000 (transmitters cost \$ 10,000 each). While the Indoor GPS technology range (60m) is better than that of UWB (10 m), it depends on a clear line of sight and some calibration points are needed unlike UWB. However, both technologies offer centimeter level positioning accuracy with Indoor GPS positioning offering significantly higher precision.

5.5. Summary and Conclusions

The author has studied several tracking technologies, and designed methods and algorithms to track mobile users in congested environments such as those found on construction sites. The research described in this chapter presented the GPS technology and compared three different wireless technologies (WLAN, UWB, and Indoor GPS) that can be used for tracking mobile users' position on outdoor and indoor construction sites. As for tracking users' head orientation, a magnetic tracker was adopted for both outdoor and indoor use.

The decision on using one indoor positioning technology over another should be based on important technical criteria (e.g. calibration, line of sight, etc.) in addition to other logistic issues such as availability, the prevailing legal situation (e.g. permitted bandwidth), and the associated implementation costs. However, based on the circumstances expected to be encountered in the intended deployment environment (i.e. indoor construction sites), the Indoor GPS positioning technology, coupled with the magnetic orientation tracker, was found to offer the most promise for indoor user tracking due to the low level of uncertainty in the reported user position (1 to 2 cm) compared to that of WLAN (1.5 to 2 m) and UWB (10 to 50 cm).

5.6. References

Aziz, Z., Anumba, C.J., Ruikar, D., Carrillo., P.M., and Bouchlaghem, D.N. (2005). "Context-aware information delivery for on-Site construction operations", Proceedings of the 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universitat Dresden, Germany, CBI Publication No:304, 321-327.

Behzadan, A.H., and Kamat, V.R. (2007). "Georeferenced Registration of Construction Graphics in Mobile Outdoor Augmented Reality", ASCE Journal of Computing in Civil Engineering, 21(4), Reston, VA, 247-258.

Behzadan, A.H., and Kamat, V.R. (2006). "GPS and 3DOF Angular Tracking for Georeferenced Registration of Construction Graphics in Outdoor Augmented Reality", 13th EG-ICE Workshop on Intelligent Computing in Engineering and Architecture, Ascona, Switzerland, 368-375.

Behzadan, A.H., and Kamat, V.R. (2005). "Visualization of Construction Graphics in Outdoor Augmented Reality", In Proceedings of the 37th Winter Simulation Conference (WSC), IEEE, Piscataway, NJ, 1914-1920.

Cisco Systems. (2002). "Wireless Local-Area Networking"- available at: http://www.sat-corp.com/products/PDF/CISCO_WLAN_Overview.pdf. (Accessed July 15, 2006).

Ekahau, Wi-Fi Based Real-time Tracking and Site Survey Solutions – available at: <http://www.ekahau.com>. (Accessed February 20, 2007).

Kamat, V. R. (2003). "VITASCOPE: Extensible and Scalable 3D Visualization of Simulated Construction Operations". Ph.D. dissertation, Department of Civil and Environmental Engineering, Virginia Tech, Blacksburg, VA.

Kang, S., and Tesar, D. (2004). "Indoor GPS Metrology System with 3D Probe for Precision Applications", In Proceedings of the ASME International Mechanical Engineering Congress (IMECE), Anaheim, CA.

Khoury, H. M., and Kamat, V. R. (2008a). "High-Precision Identification of Contextual Information in Location-Aware Engineering Applications", Advanced Engineering Informatics, Elsevier Science, New York, NY (Tentatively Accepted for Publication-In Re-Review).

Khoury, H. M., and Kamat, V. R. (2008b). "Evaluation of Positioning Tracking Technologies for User Localization in Indoor Construction Environments", Proceedings

of the 15th Workshop of Intelligent Computing in Engineering and Architecture (EG-ICE), Plymouth, UK.

Kiziltas, S., Akinci, B., Ergen, E., Tang, P., and Gordon, C. (2008). "Technological Assessment and Process Implications of Field Data Capture Technologies for Construction and Facility/ Infrastructure Management", *Electronic Journal of Information Technology in Construction (ITcon)*, Vol. 13, 134-154.

Kuusniemi, H., and Lachapelle, G. (2004b). "GNSS Signal Reliability Testing in Urban and Indoor Environments", In *Proceedings of the National Technical Meeting (NTM)*, San Diego, CA, 210-224.

Lähteenmäki, J., Laitinen, H., and Nordström, T. (2001). "Location Methods", *VTT Information Technology*- available at: <http://location.vtt.fi/source/technologies.html> (Accessed July 15, 2006)

LaMarca, A., Hightower, J., Smith, I., and Consolvo, S. (2005). "Self-Mapping in 802.11 Location Systems", In *Proceedings of the Seventh International Conference on Ubiquitous Computing Ubicomp, Lecture Notes in Computer Science*, Springer, Germany, 87-104.

Li, B., Salter, J., Dempster A.G., Rizos, C. (2006). "Indoor positioning techniques based on wireless LAN", In *Proceedings of the First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*, Sydney, Australia, 13-16.

Metris Products Website, IGPS Large Scale Metrology - available at: http://www.metris.com/largevolume_tracking_positioning/igps/ (Accessed July 24, 2007)

Multispectral Solutions, Inc. Website, Sapphire DART System- available at: <http://www.multispectral.com/products/sapphire.htm> (Accessed May 31, 2007)

OpenSceneGraph Website, Introduction to OpenSceneGraph – available at: <http://www.openscenegraph.org/projects/osg/wiki/About/Introduction> (Accessed November 18, 2005).

Schilit, B.N., Adams, N., and Want R. (1994). "Context-aware computing applications", *Workshop on Mobile Computing Systems and Applications (WMCSA)*, Santa Cruz, CA, 85–90.

Skibniewski, M. J., and Jang, W.S. (2006). "Ubiquitous Computing: Object Tracking and Monitoring in Construction Processes Utilizing ZigBee Networks," In *Proceedings of the 23rd International Symposium on Automation and Robotics in Construction (ISARC)*, Tokyo, Japan, 287-292.

Teizer, D.J., Venugopal, M., and Walia, A. (2008). "Ultra Wideband for Automated Real-time Three-Dimensional Location Sensing for Workforce, Equipment, and Material Positioning and Tracking", In Proceedings of the 87th Transportation Research Board Annual Meeting, Washington, DC.

Wang, J., and Liu, J. (2005). "Interference Minimization and Uplink Relaying at 3G/WLAN Network", In Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'), Towson, Baltimore, 388-395.

Chapter 6

High-Precision User Spatial Context Interpretation

6.1. Introduction

This chapter describes the second and third components of the developed framework (Chapter 2), which consist of demonstrating how a mobile user's spatial context can be interpreted with a level of precision that is sufficiently high to identify on-site contextual objects in location-aware engineering applications. Interpreting a user's spatial context requires 1) defining the region of space visible to a user based on the user's position and orientation that is tracked by the first framework component (Chapter 5) (Section 6.3.1), and aligning the computed volume of space with CAD objects to test for collision detection and determination of all visible objects at a particular time (Section 6.3.2), and 2) prioritizing the context by identifying specific relevant objects using high precision geometric intersection techniques (Section 6.3.3).

In order to implement the research described in this chapter, a computer graphics toolkit based on the concept of the *Scene Graph*, namely OpenSceneGraph (OSG) within Visual C++ .NET, was adopted (OpenSceneGraph Website 2005) for both region of virtual space computation and geometric interference analysis techniques.

The developed methods have been validated by conducting several outdoor and indoor experiments, where a user was accurately tracked followed by the interpretation of site object(s) in the user's view (Chapter 8).

6.2. Human Vision and Computer Graphics

The eventual consumer of all 3D computer graphics applications is the human visual system (Montalvo 1979). The very reason for the rise in the use of graphics in the first place is deeply embedded in the way humans perceive visual information. Therefore, the first question that poses itself is: Why should computer graphics concern itself with human vision? What is it about visual perception that allows this seemingly instantaneous communication from computers to humans? The answer is to be found in the way the visual system is structured (Chapter 3, Section 3.3.3) and in what it was designed to perceive.

6.2.1. Computer Graphics Based Frustum Technique

In the real world, when human beings look at the world around them, objects too far to the left or right or too far above or below them are not visible (O'Rourke 2003). They are outside the range of their peripheral vision, or field of view. The human field of view (for each eye) is often approximated by a cone-shaped area, called a cone of vision (Figure 6.1).

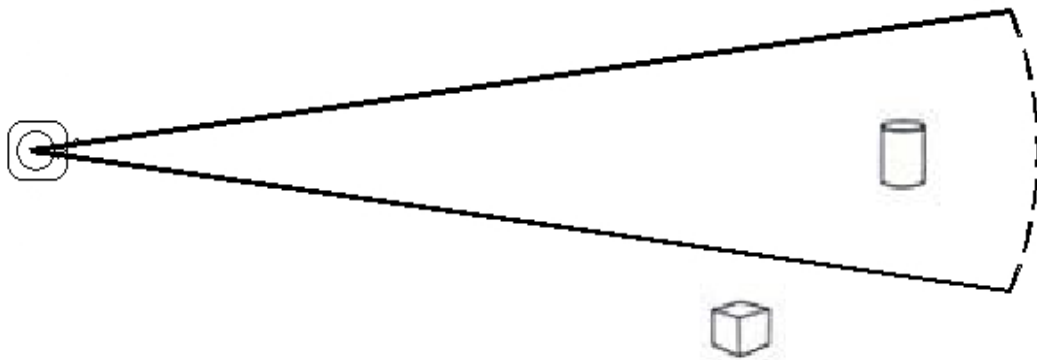


Figure 6.1 – Cone of Vision

Objects that lie within the cone of vision (such as the cylinder in Figure 6.1) are visible, whereas objects that lie outside the cone of vision (such as the cube) are not visible.

On the other hand, in computer graphics, the cone of vision becomes a pyramid of vision in order to conform to the rectangular shape of computer monitor screens (O'Rourke 2003). This pyramid of vision is called the viewing frustum or view frustum, and is typically shaped as a frustum of a rectangular pyramid (Woo et al. 1997) (Figure 6.2).

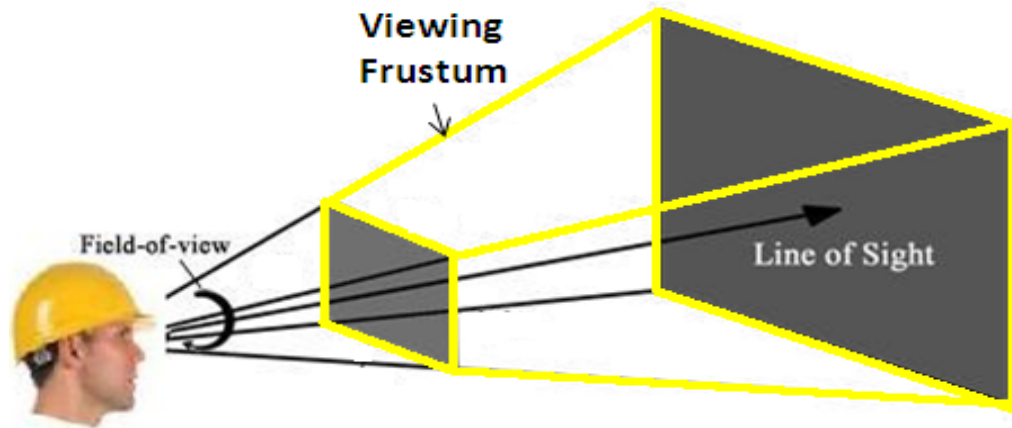


Figure 6.2 – Computer Graphics Pyramid of Vision

One of the factors that determine the shape of this pyramid is the angle usually referred to in computer graphics as the field-of-view angle or simply the field of view (FOV) measured in degrees and spanning from one corner of the viewport to the other (Figure 6.3).

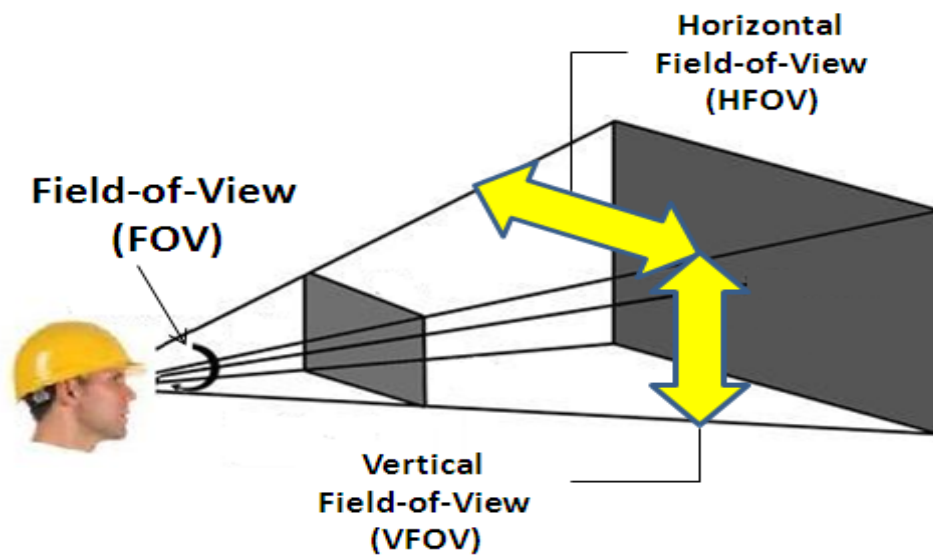


Figure 6.3 – Field-of-View Angles

Since the viewport is a rectangle and not a circle as is the case of the cone of vision, the FOV value is, in fact, split in two different values across the horizontal and vertical directions. As depicted in Figure 6.3, the horizontal field of view (HFOV) is the angular extent of the user's visible space when observed in plan view and the vertical field of view (VFOV) is similarly the angular extent of visible space in side view. Typical FOV values are defaulted to 45, but can be decremented or incremented at any interval up to 120 degrees. But the human eye really sees a larger field of view, close to 180 degrees.

When any scene is viewed, either through a photographic camera or with eyes, some objects appear in focus and some appear out of focus. Computer graphics tools reproduce this effect through a parameter called depth of field, which is given by two numbers: the nearest and the farthest distances at which objects will be rendered and seen (Figure 6.4).

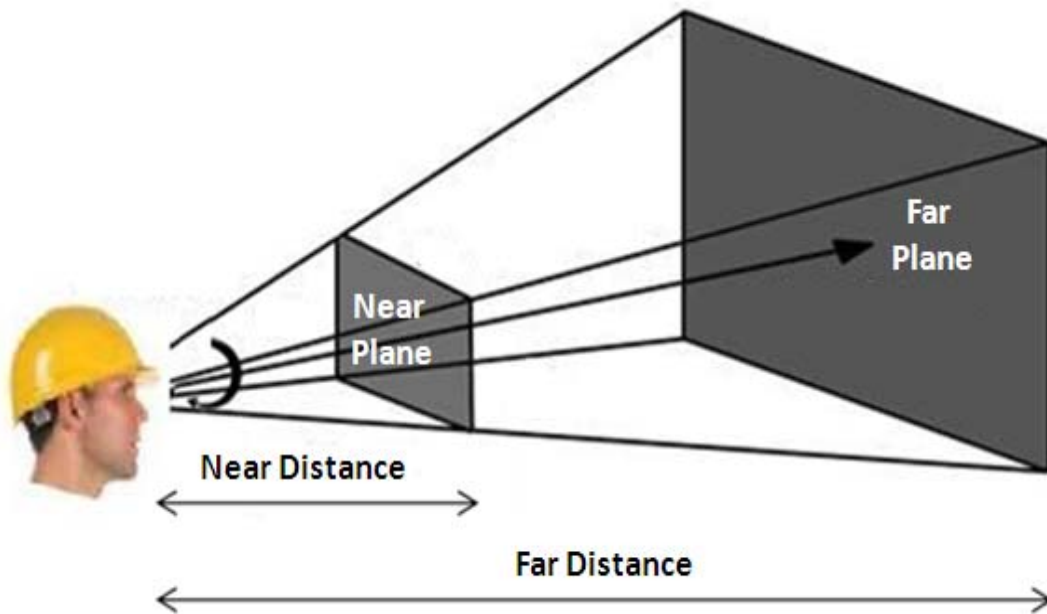


Figure 6.4 – Near and Far Plane Distances of Viewing Frustum

In the above picture, the planes that cut the frustum perpendicular to the viewing direction at the near and far distances are called the near plane and the far plane respectively or what are often called the near and far clipping planes. The idea of clipping is that there is a rectangular window that defines the portion of a scene or model that is visible, much as a window in a wall defines what portion of the outside world can be seen (McConnell 2005). Therefore, objects closer to the user than the near plane or beyond the far plane are assumed to be out of sight and context. Typically in computer graphics, the near plane is chosen close to the user's viewpoint and the far plane is placed infinitely far away so all objects within the frustum are considered to be of interest regardless of their distance from the user. However, for the purpose of this research, near and far distances are dependent on the user's choice, and how near and far the user expects the most extreme case objects to be. Therefore, the choice is purely empirical and made based on the knowledge of what really is needed to be contextual and of interest.

6.2.2. Computer Graphics Based Intersection Techniques

6.2.2.1. General Collision Detection Techniques

The goal of collision detection is to report any contacts (or containments) between geometric objects when they occur (Lin and Gottschalk 1998). The geometric models may be polygonal objects, splines, or algebraic surfaces.

The collision detection problem has been encountered in computer-aided design and machining (CAD/CAM), robotics and automation, manufacturing, and most importantly computer graphics, animation and computer simulated environments. Many of the tasks performed in the latter domain involve contact analysis and spatial reasoning among static or moving objects. For instance, a virtual environment (i.e. walkthrough) creates a computer-generated world, filled with virtual objects. Such an environment should give the user a feeling of presence, which includes making the images of both the user and the surrounding objects feel solid. For example, the objects should not pass through each other, and things should move and behave as expected (Kamat 2003).

Such actions require accurate collision detection, if they are to achieve any degree of realism. Other applications for collision detection in computer graphics are based on the view frustum clipping and culling methods used when setting camera views for a 3D scene and optimizing rendering processes (Assarsson and Stenström 2001). Clipping determines whether virtual objects (or polygons) are within the frustum at all (Section 6.2.1). Culling involves cutting out virtual objects lying outside the viewing frustum, and not displaying and rendering them.

It was found in this research that the collision detection concept lying behind these latter methods, can be adopted in the proposed methodology (Chapter 2) as a first step to disregard objects not colliding with the frustum and instead detect visible objects for the purpose of extracting contextual information .

In order to achieve this objective, a number of bounding volumes have been used for collision detection between geometric models, in particular general polygonal models (Lin and Gottschalk 1998). A bounding volume is an approximation of complex objects by simplified geometries for faster processing and more efficient geometrical operations. Typical examples of bounding volumes include axis-aligned boxes and spheres and they are chosen for their fast overlap tests. Other structures include cone trees, k-d trees and octrees (Samet 1989), sphere trees (Hubbard 1993, Quilan 1994), trees based on S-bounds (Cameron 1991) etc. More recent work has focused on tighter-fitting bounding volumes (Tropp et al. 2006).

Cohen et al. (1995) have presented algorithms and a system, I-COLLIDE, based on spatial and temporal coherence, for large environments composed of multiple moving objects. The number of object pair interactions is reduced to only the pairs within close proximity by sorting axis-aligned bounding boxes (AABBs) surrounding the objects. Gottschalk et al. (1996) have presented a fast algorithm and a system, called RAPID, for interference detection based on oriented bounding boxes (OBB), which approximate geometry better than do AABBs. Hudson et al. (1997) have presented another system, called V-COLLIDE. It is a collision detection library for large dynamic environments and unites the n-body processing algorithm of I-COLLIDE and the pair processing algorithm of RAPID. It is designed to operate on large numbers of static or moving polygonal objects to allow dynamic addition or deletion of objects between time steps.

In Section 6.3.2, all the steps involved in identifying objects in a mobile user's field of view using the pair processing algorithm of RAPID are presented.

6.2.2.2. Raycasting Technique

Raycasting is the use of ray-surface intersection tests to solve a variety of problems in computer graphics (Foley 1995).

The first raycasting algorithm was presented by Arthur Appel in 1968 for scene rendering purposes (Macey 1999). In this case, an image is thought of as a screen-door with each square in the screen being a pixel (Figure 6.5). The idea behind the algorithm is to shoot rays from the eye, one per pixel, and find the closest object blocking the path of that ray. This is then the object the eye normally sees through that pixel (Objects A, B, and C in Figure 6.5).

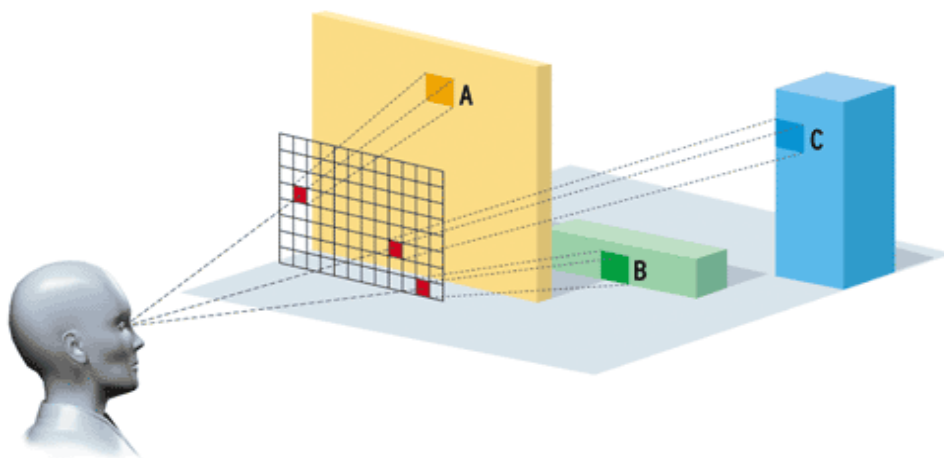


Figure 6.5 – First Raycasting Technique

Based on the above concept, raycasting can also refer to a technique for hidden surface removal (Adams et al. 2005).

Object shading effect is another computer graphics concept where raycasting process lends itself conveniently (Permadi 2009). For instance, when an object is farther away from the viewer, the object should appear less bright. In this case, a ray can be cast from the viewer and the distance to the object to be rendered is obtained. Using a color matching algorithm and mapping the result into an intensity-distance table (object intensity gets smaller with distance), the object will be correctly shaded and rendered. Similar computer graphics concepts, making use of the raycasting technique, include Level-of-Detail (LOD) effects changing with distance to viewer.

Another typical need of raycasting involves obtaining the current Height Above Terrain (HAT) information in a flight simulator or a driving simulator (Drugge 2005)(Figure 6.6).



Figure 6.6 – Flight Simulator Raycasting Scenario

This height information is determined by firing a vertical line segment from the aircraft or vehicle towards the terrain/ground and calculating the distance between the aircraft or vehicle and the intersection point on the ground (Kamat 2003).

This same concept applies for finding the correct height (z) value on a terrain model to place trees or other static objects on a terrain. The height information is determined by firing a vertical line segment at a location (x,y) through the terrain and then determining the intersection point on the ground (Figure 6.7). This intersection data is used to position the tree or other static object.



Figure 6.7 – Tree Location Raycasting Scenario

Based on the abovementioned cases, the concept of raycasting in this chapter, mainly involves casting imaginary/virtual lines originating at the user's viewpoint and heading inside the viewing volume to test for intersection with geometric representations of

visible objects, i.e. nearest objects blocking the path of rays along or closer to the line of sight (Section 6.3.3).

6.3. Accurate Contextual Object Identification

6.3.1. Computing Region of Space Visible to User

By knowing the user's position and orientation that is tracked by the first framework component (tracking module) (Chapter 5), the user's line-of-sight can be defined and the region of space that is in the field of view at that time can be computed. In this section, based on the concept of the viewing frustum explained in Section 6.2, the author mathematically derived the formulation for the region of space visible to a mobile computing user at any time.

First, the notion of coordinate systems is introduced. In all the 3D graphics applications carried out in this research, a 3D Cartesian world coordinate system similar to the one adopted in OSG was used (Figure 6.8).

In addition to the x and y-axis known in 2D graphics, there is the z-axis which extends vertically up from the mobile user into positive space and down from the user into negative space. As shown in Figure 6.8 and similar to any graphical CAD object, the mobile user CAD model has, beside the world coordinate system, a local coordinate system (Figure 6.9) as well.

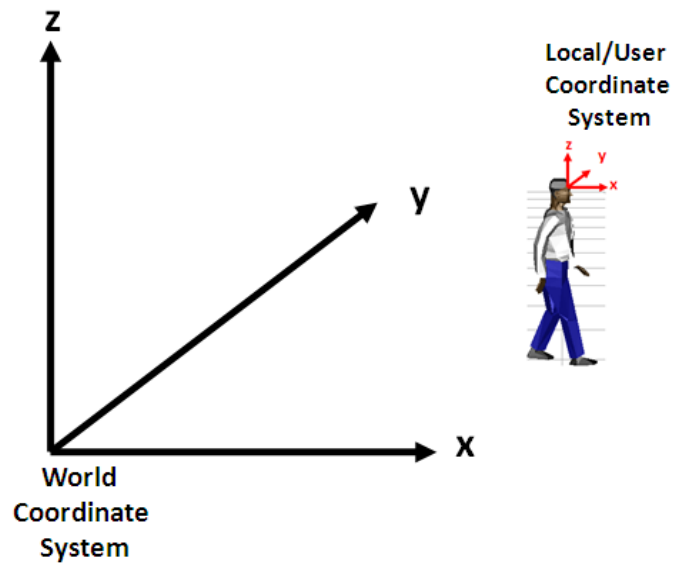


Figure 6.8 – World and Local/User Coordinate Systems

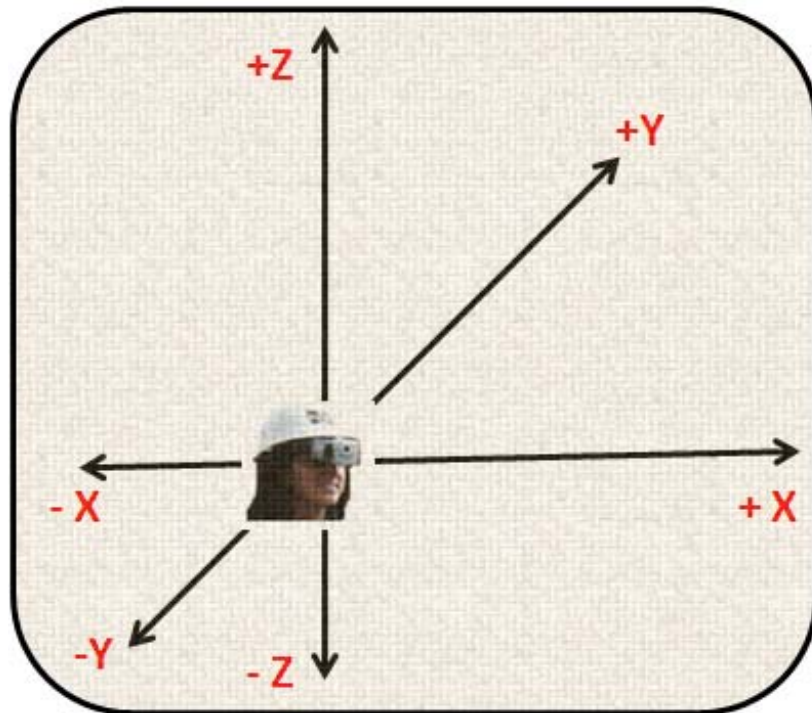


Figure 6.9 – 3D Cartesian Local or User Coordinate System

The local coordinates are usually used for rotating the object/user, and the world coordinates are used for positioning the user or any other CAD object in the world or in a 3D level graphical scene. It is crucial to understand that when the user is moving, what occurs is not the actual movement of the user in space, moving from one point to another. In fact, the respective local coordinate systems are being modified by applying transformations on them and moving them either positively or negatively along any one or more of the three axes of the world coordinate system (X, Y, or Z). Therefore, all movement in 3D is based on these geometrical transformations. There are rules to these transformations, specifically the order in which transformations should be performed.

Based on the previous explanation, the region of space visible to a mobile computing user is graphically shown in Figure 6.10. The user's line of sight or viewing direction is aligned with the x-axis, left and right user's movements along the y-axis, and upward and downward movements along the z-axis. The user's viewpoint is at the world position (x, y, z), the distances from the viewpoint to the near plane (P1-P2-P3-P4) and far plane (P5-P6-P7-P8) are n and f respectively, and the field of view angles in both directions are HFOV and VFOV respectively. If both angles are equal, then the width and height of each of the near and far planes become equal.

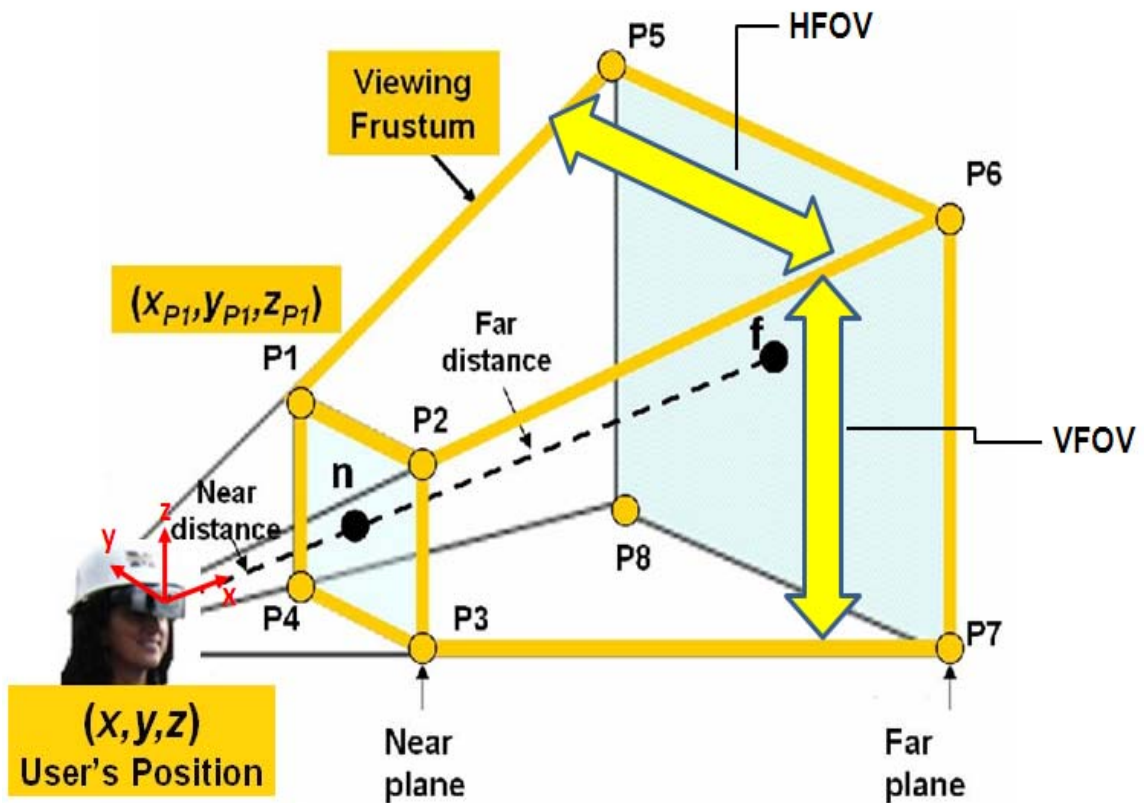


Figure 6.10 – Region of Space Visible to a Mobile Computing User

The frustum computation procedure was divided into two main steps:

A first step included computing the coordinates of the pyramid's corner vertices ($P1$ - $P8$) based on just the user's position, n , f , HFOV and VFOV. Figure 6.11 illustrates the (x, y, z) coordinates of points $P1$ - $P4$ computed based on the geometric relationships between the user's line of sight and the visible space in plan and side views (Figure 6.12).

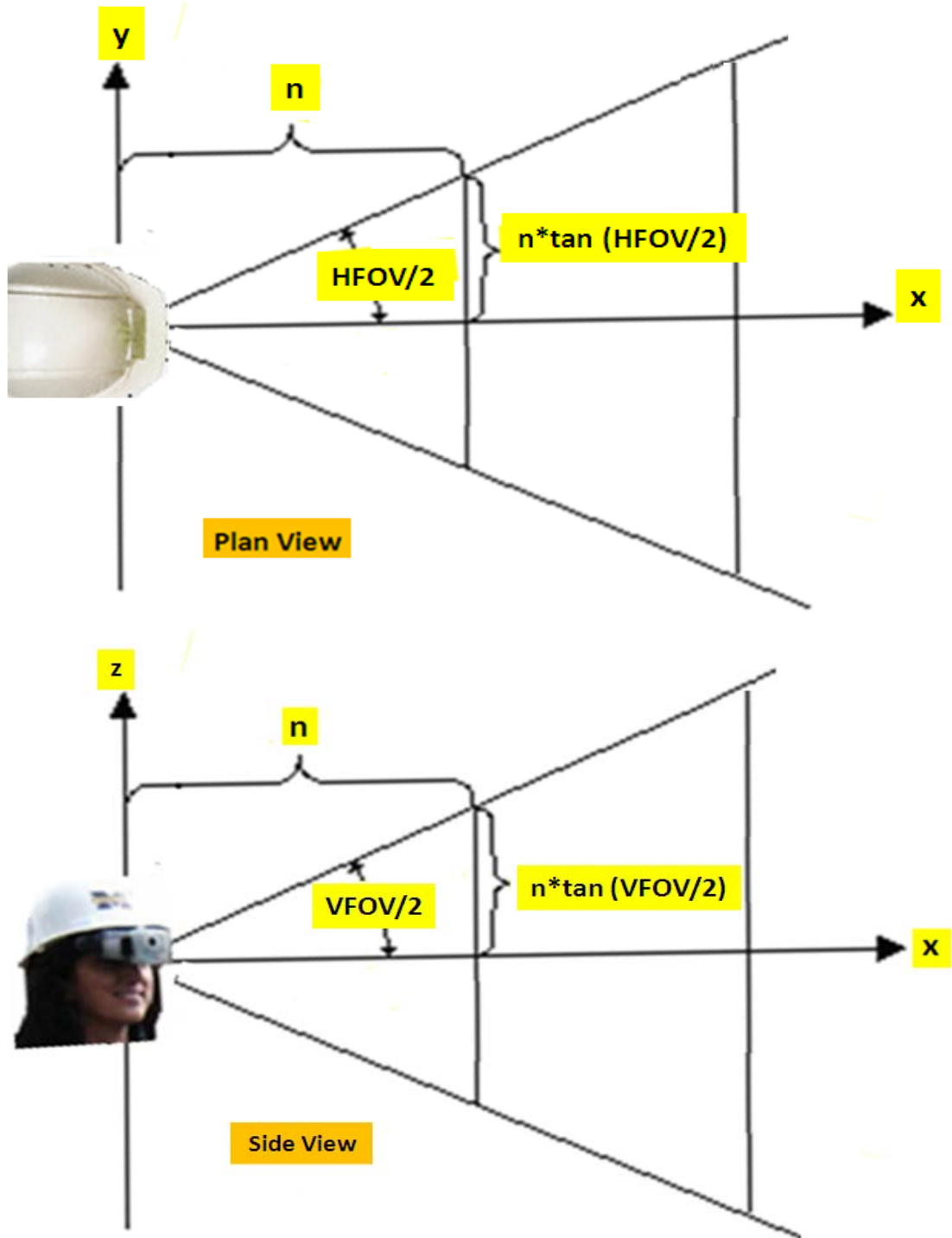


Figure 6.11 – Geometric Relationship between Line of Sight and Visible Space for a) yaw (α) and b) pitch (β) Rotations

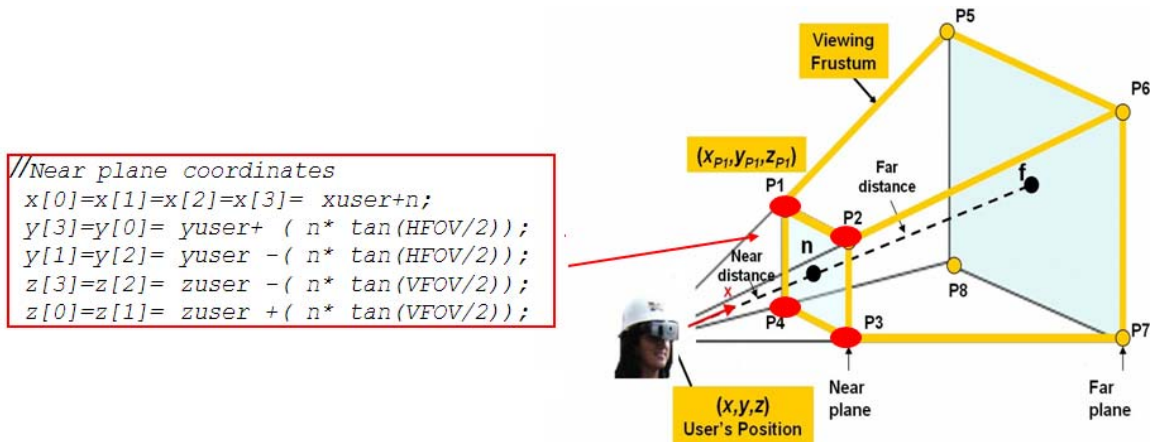


Figure 6.12 – Coordinates of Frustum’s Near Plane Vertices

The other coordinates of the four corner vertices on the far plane are computed in a similar way with clipping distance f instead of n (Figure 6.13).

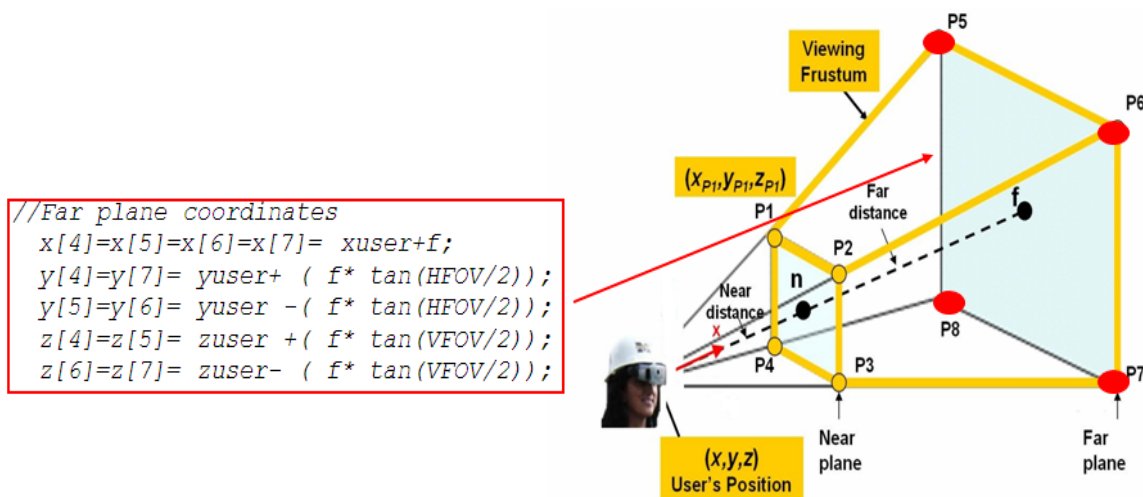


Figure 6.13 – Coordinates of Frustum’s Far Plane Vertices

The second step involved adjusting the previously computed coordinates by taking into account the user’s head orientation movements at any instant in time. As mentioned in Chapter 5, those movements are defined by three rotation angles, yaw, pitch and roll,

symbolically represented by α , β , and γ respectively as shown in Figure 6.14. Yaw is the rotation around the vertical axis (left or right movement), pitch is the rotation around the side-side axis (up or down motion), and roll is the rotation around the front-to-back axis.

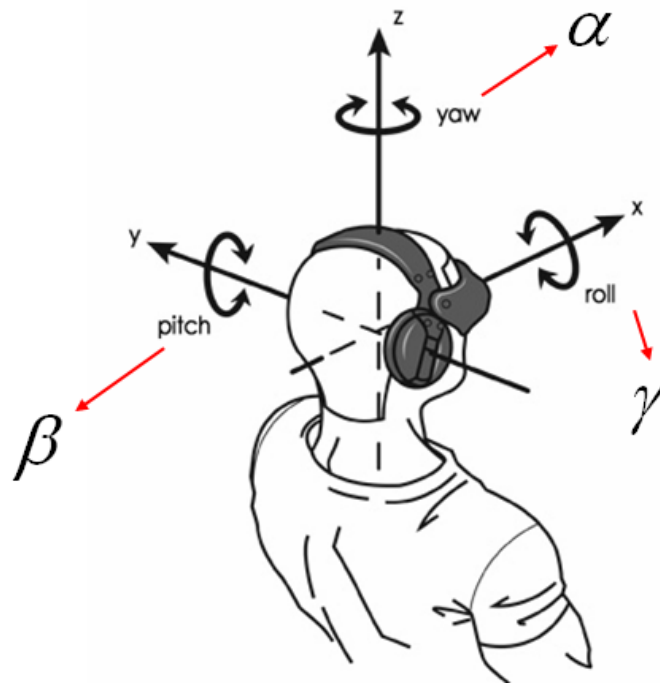


Figure 6.14 – Roll, Pitch and Yaw Angles

Therefore, in order to rectify the coordinates and apply orientation transformations, three basic rotation matrices in three dimensions are needed (Figure 6.15). These matrices represent counterclockwise rotations of an object/user relative to fixed coordinate axes, by angles of α , β , and γ around the z , y , and x axes respectively (R_z , R_y , R_x). The direction of the rotation is always determined by the right-hand rule: R_x rotates the z -axis towards the y -axis, R_y rotates the x -axis towards the z -axis, and R_z rotates the x -axis towards the y -axis (Woo et al. 1997).

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_Z(\alpha)$$

$$R_Y(\beta)$$

$$R_X(\gamma)$$

Figure 6.15 – Basic 3D Rotation Matrices

Figure 6.16 a and b depict each of the geometric relationship between the line of sight and the visible space for yaw (α) and pitch (β) rotations and Figure 6.16 c represents graphically the geometric transformation for roll (γ) rotation.

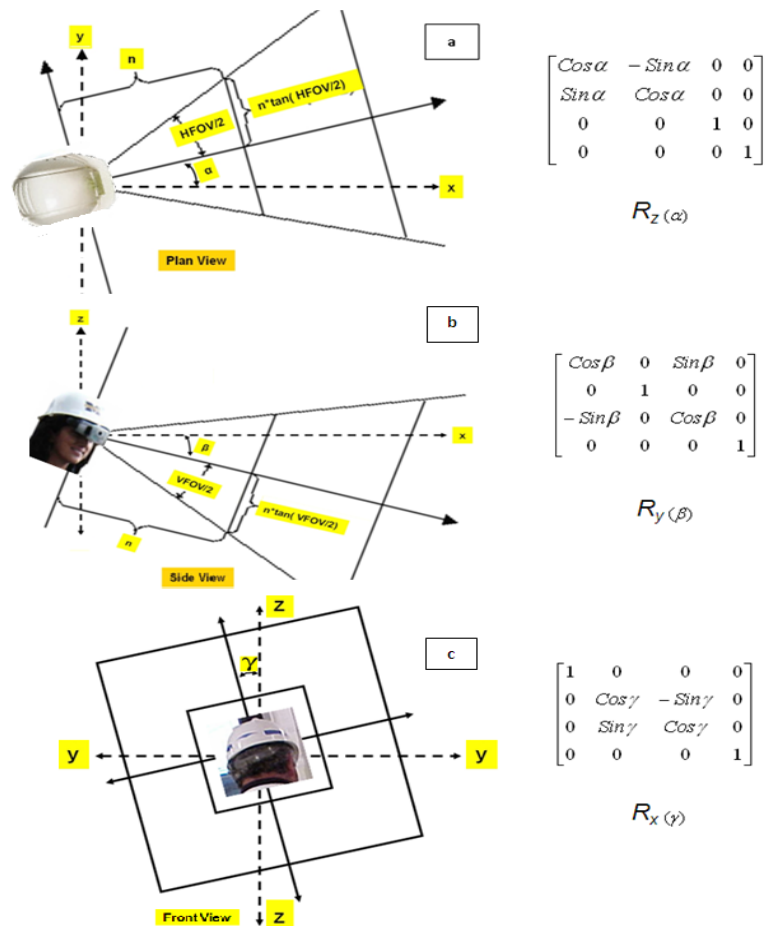


Figure 6.16 –Geometric Transformation for a) yaw (α), b) pitch (β), and c) roll rotations (γ)

The mobile user can perform rotations around all three axes leading to a matrix multiplication of the three individual rotation matrices as follows (Figure 6.17):

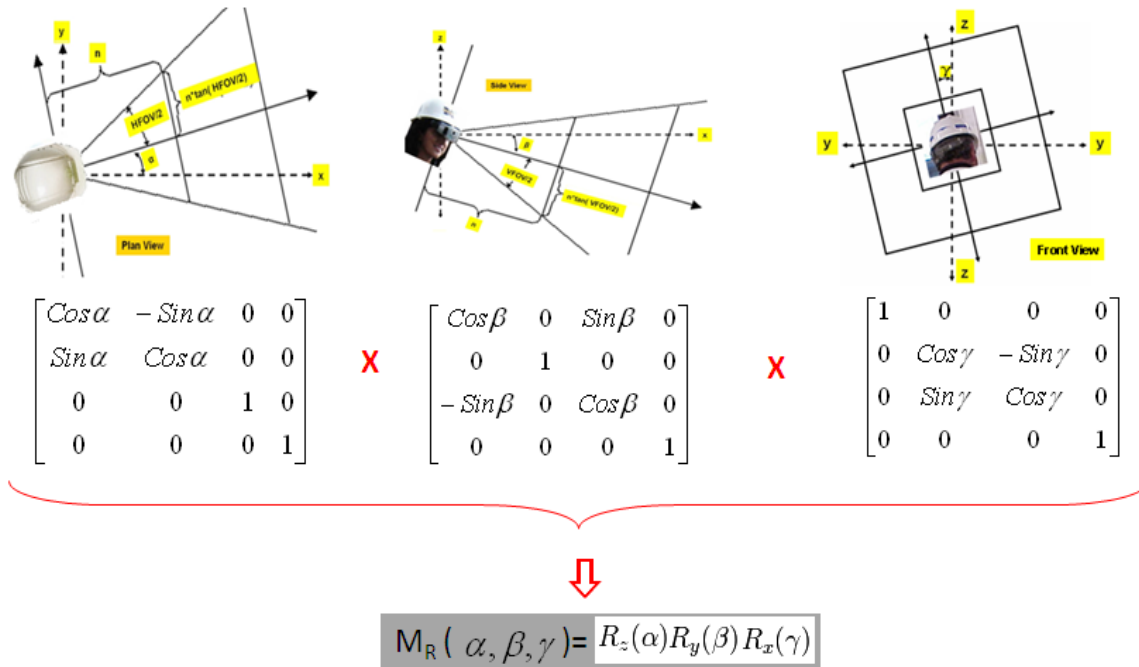


Figure 6.17 – Matrix Multiplication of the Three Rotation Matrices

An important question that arose was: What if the mobile user is at a location (x,y,z) different than global (0,0,0) position, which is always the case, and is moving his head and changing his viewing angle to identify contextual objects? The outcome is that the user is rotating around an imaginary origin, located at (0, 0, 0), as if it is a planet rotating around the sun and not around its center. Therefore, rotation transformations are performed around the center of the given world coordinate system. This was virtually achieved by translating the user (or frustum) from his current location to (0,0,0) by applying (-x,-y,-z), rotating its local coordinate system around the different axes (M_r as

shown in Figure 6.17) and then translating back to his location by applying (+x,+y,+z) (Figure 6.18). The user still resides at (0, 0, 0) on its local coordinate system.

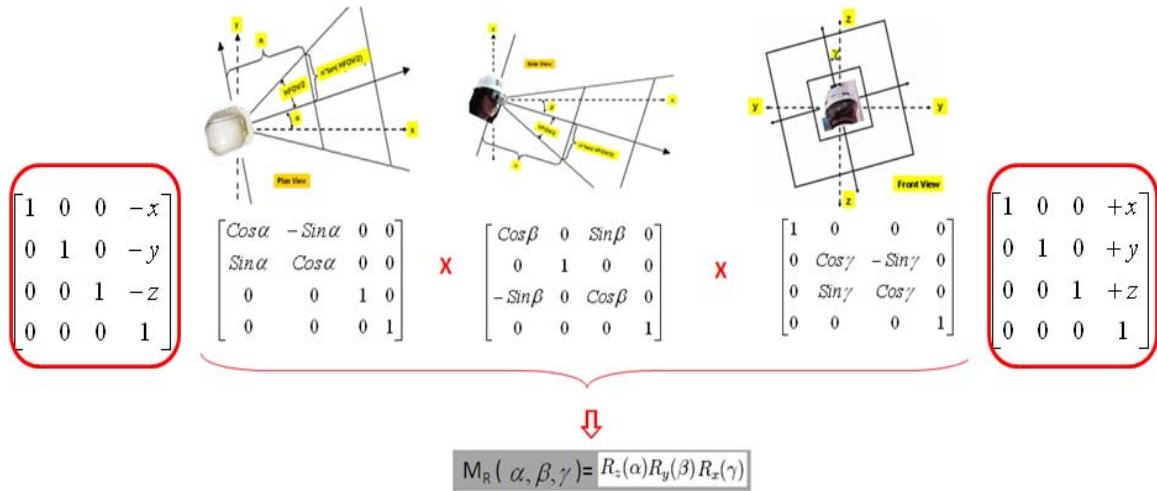


Figure 6.18 – Rotation and Translation Transformations

Based on the aforementioned trigonometric computations, interpreting the region of space visible to the user at a given time involves computing the coordinates (x_n, y_n, z_n) of each of the frustum's corner vertices (i.e. P1 through P8) as a function of the tracked spatial context parameters, i.e. the user's position (x, y, z) and the three-dimensional orientation of the line of sight (α , β , and γ) as well as near and far distances (n and f) and field of view angles (*HFOV* and *VFOV*).

Figure 6.19 presents the equations derived to calculate the coordinates (x_{P1}, y_{P1}, z_{P1}) of vertex P1. Similar equations can be derived for the other vertices. Thus, by computing the coordinates of all eight frustum vertices (P1-P8), it is possible to accurately calculate the region of space that is in the user's context at a particular time.

$$\begin{aligned}
x_{P1} &= \left[\cos\left(\frac{HFOV}{2}\right) \times (n \cos\beta \cos\gamma + x) - \sin\left(\frac{HFOV}{2}\right) \times n(\cos\alpha \sin\gamma + \sin\alpha \sin\beta \cos\gamma) \right] \times \left/ \begin{array}{l} \cos\left(\frac{HFOV}{2}\right) \times \\ \cos\left(\frac{VFOV}{2}\right) \end{array} \right. \\
&\quad \cos\left(\frac{VFOV}{2}\right) - \cos\left(\frac{HFOV}{2}\right) \times n(\cos\alpha \sin\beta \cos\gamma - \sin\alpha \sin\gamma) \times \sin\left(\frac{VFOV}{2}\right) \\
y_{P1} &= \left[\cos\left(\frac{HFOV}{2}\right) \times (n \sin\beta + y) + \sin\left(\frac{HFOV}{2}\right) \times n \sin\alpha \cos\beta \right] \times \left/ \begin{array}{l} \cos\left(\frac{HFOV}{2}\right) \times \\ \cos\left(\frac{VFOV}{2}\right) \end{array} \right. \\
&\quad \cos\left(\frac{VFOV}{2}\right) + \cos\left(\frac{HFOV}{2}\right) \times n \cos\alpha \cos\beta \times \sin\left(\frac{VFOV}{2}\right) \\
z_{P1} &= - \left[\cos\left(\frac{HFOV}{2}\right) \times (n \cos\beta \sin\gamma - z) + \sin\left(\frac{HFOV}{2}\right) \times n(\cos\alpha \cos\gamma - \sin\alpha \sin\beta \sin\gamma) \right] \times \left/ \begin{array}{l} \cos\left(\frac{HFOV}{2}\right) \times \\ \cos\left(\frac{VFOV}{2}\right) \end{array} \right. \\
&\quad \cos\left(\frac{VFOV}{2}\right) - \cos\left(\frac{HFOV}{2}\right) \times n(\cos\alpha \sin\beta \sin\gamma + \sin\alpha \cos\gamma) \times \sin\left(\frac{VFOV}{2}\right)
\end{aligned}$$

Figure 6.19 – Vertex Coordinates of View Frustum

The last step involved graphically drawing the pyramid within a scene graph application, OSG specifically, based on the computed coordinates of the frustum's vertices. The frustum consists of a pyramid tip (mobile user's location), base, top, and rectangular and triangular faces (Figure 6.20). For example, the pyramid base is illustrated by the four points (P5-P8), the top triangular face is represented by the three points (P1, P2, and P_{user}), etc. In drawing the different parts of the pyramid, vertices are specified in a counterclockwise order. Then the whole drawn frustum is added to the scene so that all position and orientation transforms can be applied to it.

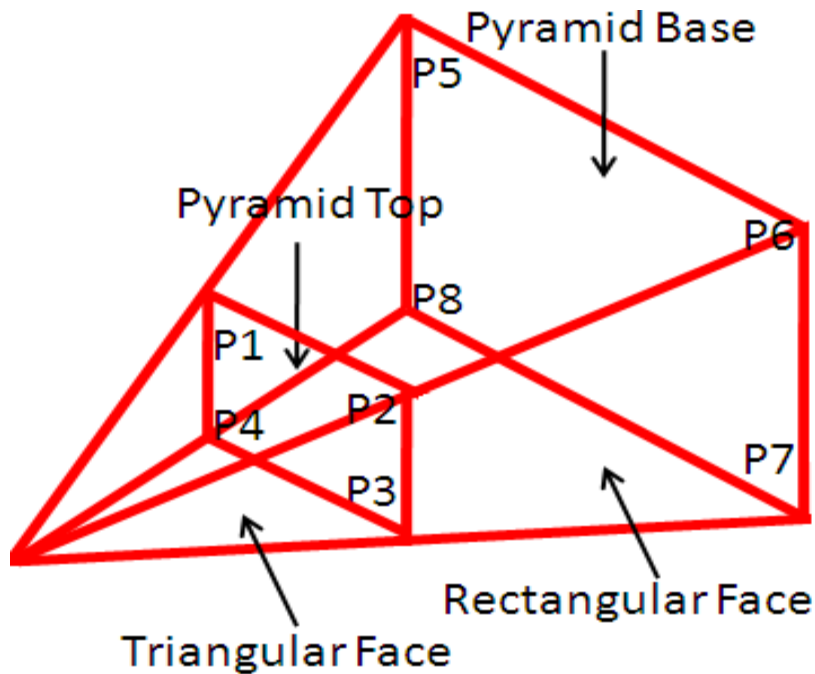


Figure 6.20 – Viewing

Frustum Graphical Components

6.3.2. Identifying Objects Contained in User's Visible Space

In order to interpret which objects in the user's surrounding environment are currently in context (i.e. visible), the computed view frustum was represented as a geometric model (i.e. CAD object) on the computer (Section 6.3). This model was then tested for geometric interference with CAD representations of objects in the user's surroundings using general collision detection techniques (6.2.2.1). This was done after reconciling (i.e. overlapping) the coordinate system used to track the user's navigation with the coordinate system used to model the user's surroundings in CAD. In this particular case, if the graphical representation of any object in the user's surroundings intersects with the geometric representation of the user's view frustum, then it can be concluded with

certainty that the corresponding real object(s) is indeed in the user's view at that particular time (Figure 6.21).

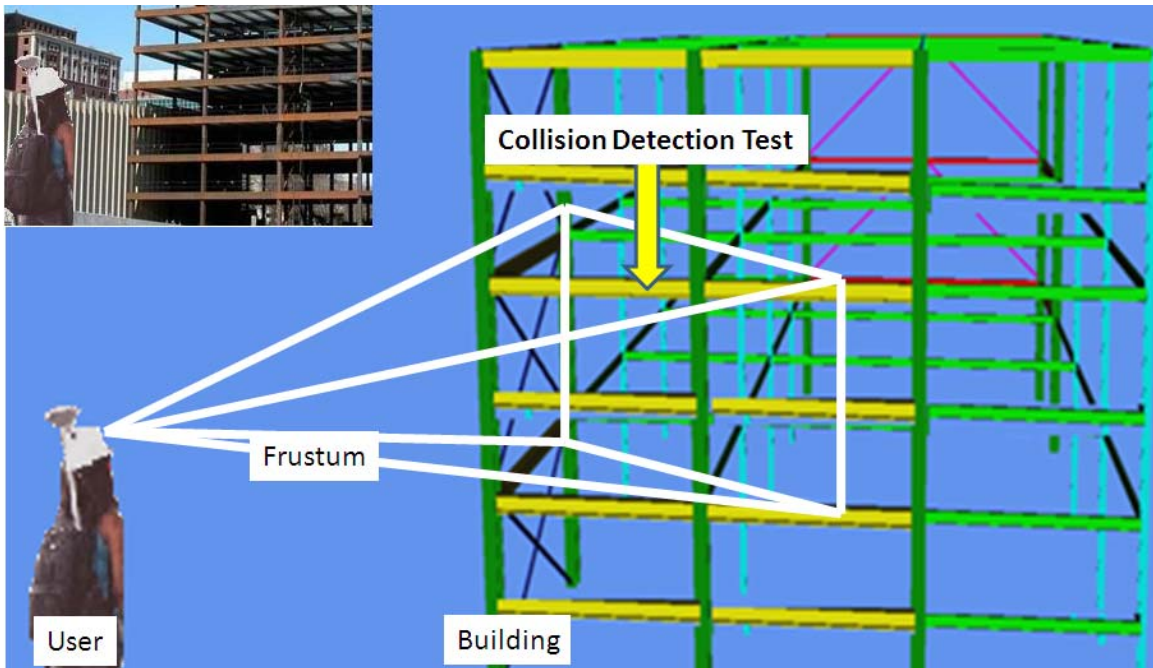


Figure 6.21 – Geometric Interference Test

One of the common collision detection techniques is V-Collide (Hudson et al.1997). It is a robust and accurate polygon interference detection library for pairs of unstructured polygonal models. V-Collide uses a three-stage collision detection architecture:

- An axis aligned bounding box test, called N-body or prune test, eliminates pairs of scene objects that cannot possibly collide and finds possibly colliding pairs of objects,
- An oriented bounding box test finds possibly colliding pairs of objects, and
- An exact and accurate test determines whether or not a pair of object triangles overlaps.

For the purpose of the proposed methodology, the first algorithm is not taken into account and therefore, it was decided to consider RAPID collision detection system (Gottschalk et al. 1996) which only covers the second and third stages of V-Collide.

In this case, and as mentioned in Section 6.2.2.1, the two-stage technique constructs imaginary bounding volumes called OBBs around each scene building object, which approximate geometry better than do AABBs (Figure 6.22).

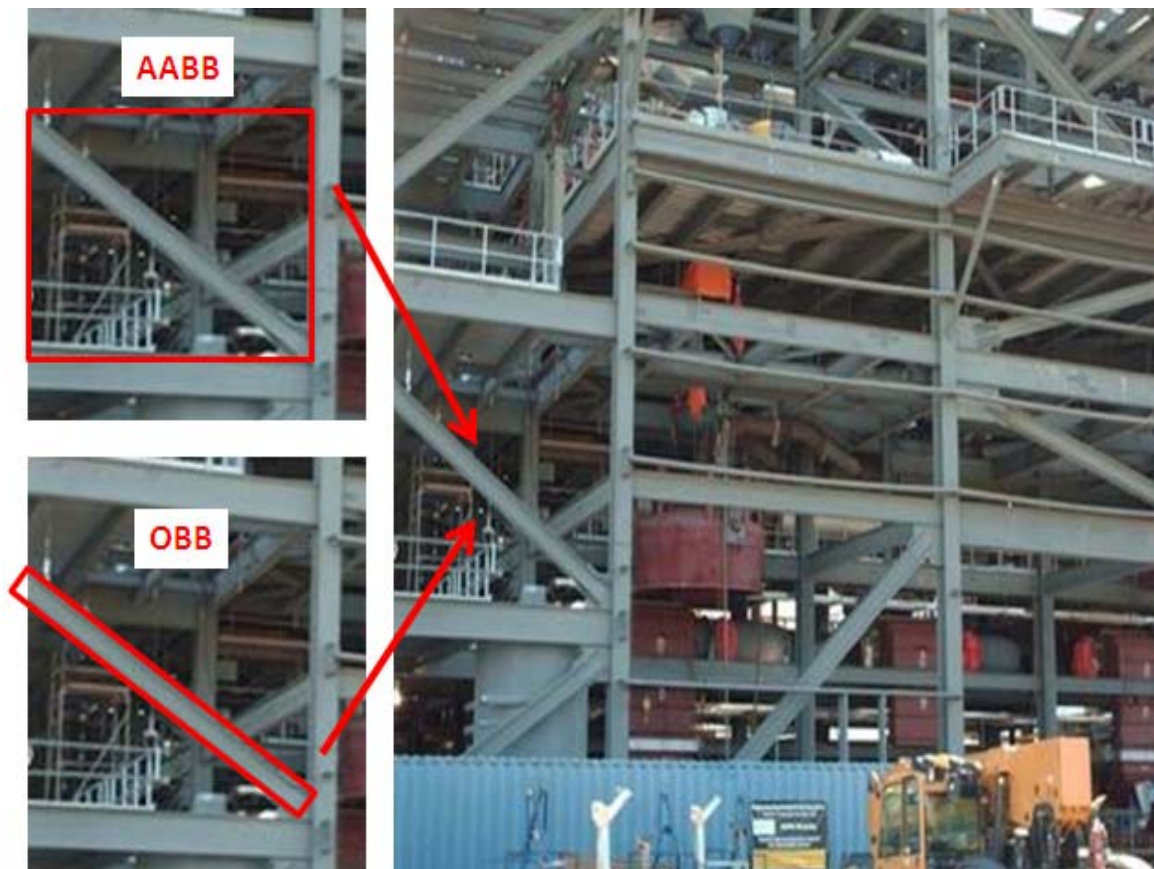


Figure 6.22 – Bounding Volume Representations

Unlike AABBs that are always aligned with the coordinate axes, OBBs can be oriented in any direction such that the resulting volume encloses the bounded object as tightly as possible. Since OBBs are constructed taking an object's orientation into account, the size and amount of free space inside the enclosing bounding box always remain constant. As a result, OBBs describe tight fitting bounding volumes for all enclosed object orientations. Figure 6.23 shows the OBBs of the building front façade structural objects.

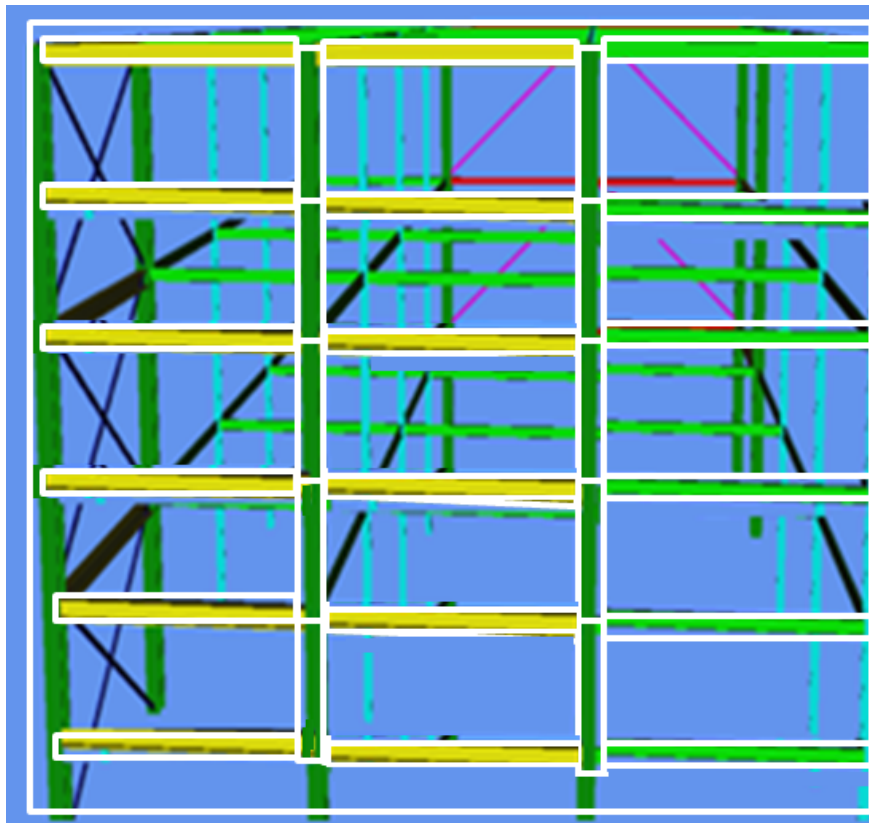


Figure 6.23 – OBBs of Building Objects

Each frustum-building object pair is then subjected to narrow-phase geometric tests to determine all existing contacts. More specifically, overlap tests are performed each time a check is desired between the moving viewing frustum and OBBs of static visible building

objects. If the viewing frustum does not intersect the respective bounding volume, it cannot intersect the object contained in the volume. Otherwise, if intersection is detected, the overlap tests are then followed by exact and accurate geometric primitive intersection tests (i.e. intersection between pair of triangles) to confirm collisions and determine exact point of contact. These intersection tests produce a list of objects that are identified and displayed.

Figure 6.24 depicts all the intersections between the viewing frustum and building objects. In this case, seven objects were identified as visible and contextual (Figure 6.25), i.e. B3, B4, B9, B10, C9, C10, and C11.

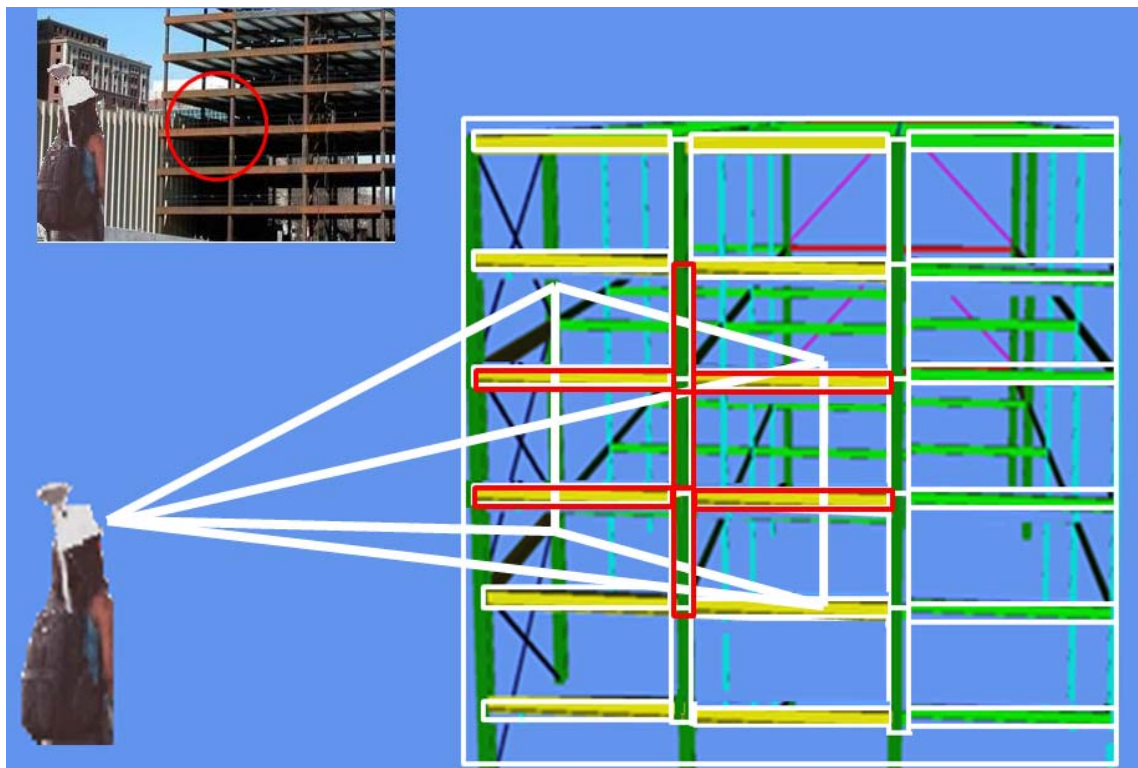


Figure 6.24 – Interference Detection Using RAPID Technique

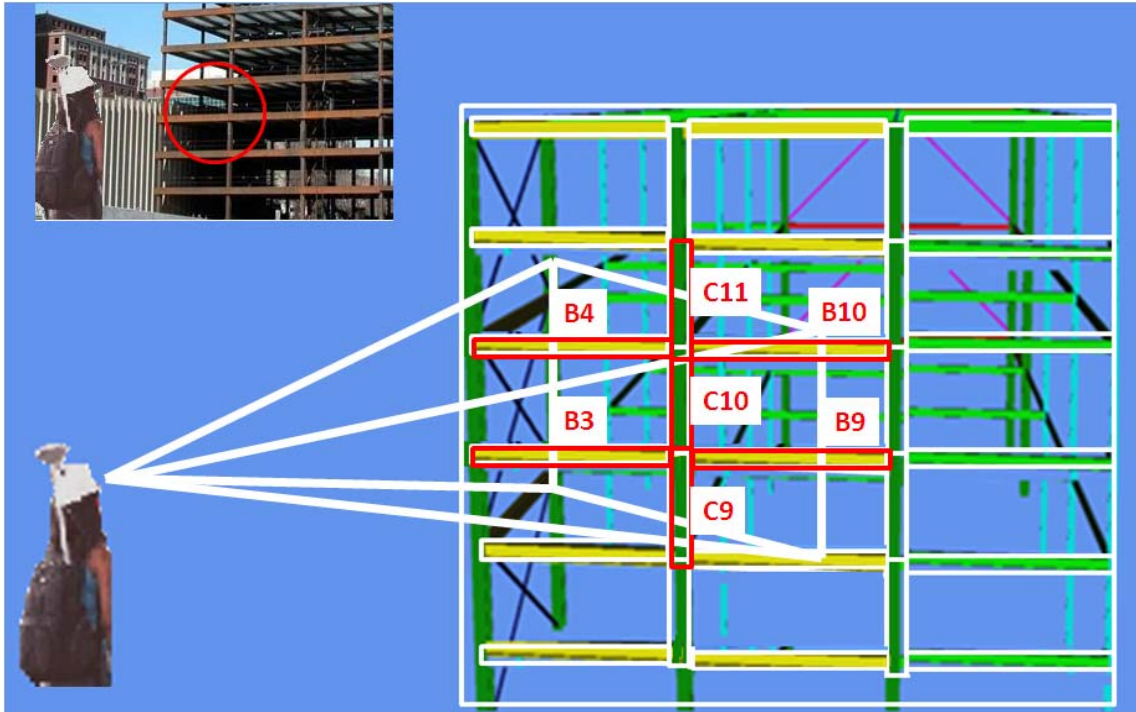


Figure 6.25 – Visible Objects Identification

The complexity of the required computation is directly proportional to the 3D graphical building model (i.e. number and types of objects contained). However, it is the mobile user's choice to define his viewing volume and limit the number of visible objects. As for the object types, the model might often contain complex shapes. In order to obtain bounding volumes of complex objects, a common way is to break the objects/scene down using a scene graph or more specifically bounding volume hierarchies (BVH) like e.g. OBB trees (Gottschalk et al.1996). The basic idea behind this is to organize a scene in a tree-like structure where the root comprises the whole scene and each leaf contains a smaller subpart.

The general collision detection test can prove very useful when a mobile user is inspecting a part of a building that is yet to be constructed. In this case, the aforementioned two-stage technique virtually computes the intersections thereby allowing contextual information retrieval about all entities yet to be built.

6.3.3. Increasing Precision in Identifying Contextual Objects

Using the method described in Section 6.3.2, several objects can be detected in the user's view frustum at any given time. However, the user might be specifically interested in only one or few of the identified visible objects at that time. Such precision in interpreting which specific visible object(s) the user is interested in can be achieved by adopting another geometric interference analysis technique, namely raycasting (Section 6.2.2.2). As mentioned earlier, the objective is to graphically test for intersection between the rays cast and geometric representations of the visible objects, and then prioritize the identified contextual objects. These methods were implemented by using OSG, mainly the functions provided by the *osgUtil* intersection modules (OpenSceneGraph Website 2005). Below is a brief description of the basic classes and methods used:

osg::LineSegment provides a means to define the rays. Line segments consist of two *osg::Vec3* instances - one to define a start point and one to define the end point. When invoked, intersection detection is performed along this ray. *osgUtil::Hit* provides access to the basic values associated with a intersection detection test. A hit contains the information available when a ray intersects geometry in the scene.

osgUtil::IntersectVisitor::HitList is deemed useful when a segment (or multiple segments) intersect any number of geometry instances within a scene (or with the same geometry multiple times.). In this case, for each segment that intersection detection is performed, a list is created. This list contains the *Hit* instances associated with each intersection detected. If no intersections are detected, the list remains empty.

The base of raycasting intersection testing method is *osgUtil::IntersectVisitor*. *IntersectVisitor* provides the ability to handle collision detection between objects within a scene graph and is an essential part of most visual simulations (Tomlinson 2007). It is created and applied to initiate intersection detection tests between a ray and geometry instances in a scene. A list of line segments to use for intersection detection is maintained. For each of these segments, a list of hits (*osgUtil::IntersectVisitor::HitList* instance) is created.

Using the above classes, the raycasting technique adopted in this research comprised primarily of the following steps:

First, the ray or rays are created as a *LineSegment* with start and stop coordinates. Then, those *LineSegment(s)* are added to a created *IntersectVisitor*. The next steps involve starting a traversal with the *IntersectVisitor* at a start node (e.g. the root), mentioning a *Hit* with objects or bounding spheres of objects in view (identified in Section 6.3.2) , using a hit list to retrieve the resulting hits from the test, and finally getting hold of the nodes (and coordinates) for the identified hits.

Given that the user can only focus on a very small area at any point in time using the eye's fovea (Section 3.3.1), the aforementioned raycasting intersection testing procedure was first used to cast one ray along the line of sight (LOS), LOS ray (Figure 6.26). The goal was to optimize the interference test by having the search process terminate on the first positive intersection(s) of the line of sight with one or more objects out of those identified in Section 6.3.2. In this case, the LOS ray was created as a *Line Segment* defined by two 3D vectors, begin and end positions. The begin position is set as the user's viewpoint (tracked x,y,z in Chapter 5) and the end position set as the center point of the user's view frustum far plane. This center point is computed using the coordinates of the four corners vertices (P5-P8) obtained in Section 6.3.1 (Figure 6.26).

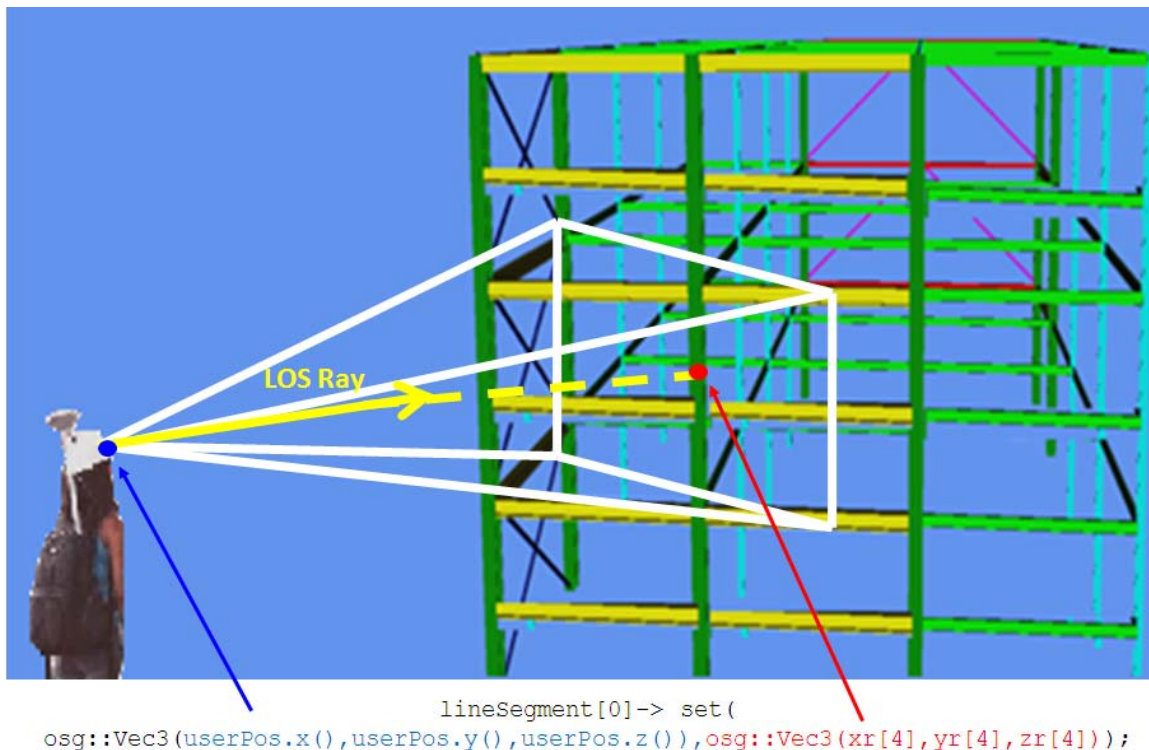


Figure 6.26 – LOS Raycasting Test

If intersection is detected, the LOS ray color switches to red (Figure 6.27) and the hit node/object is identified, C10 in this case (as depicted from Figure 6.25).

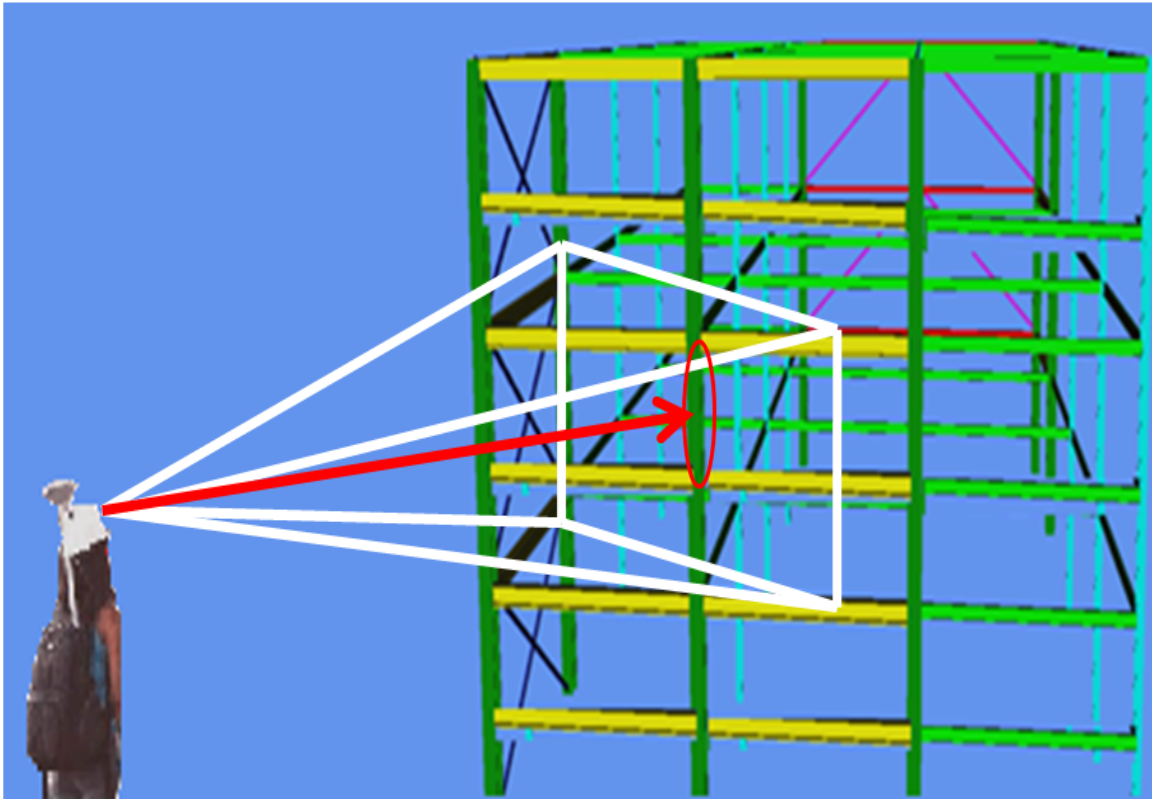


Figure 6.27 – Prioritizing Objects of Interest with LOS Raycasting

In some situations, rays cast along the line of sight detect more than one node/object along the same segment line (Figure 6.28). In this case, hits are sorted by depth with the first one being the foremost, and thereby intersecting and identifying the closest relevant object.

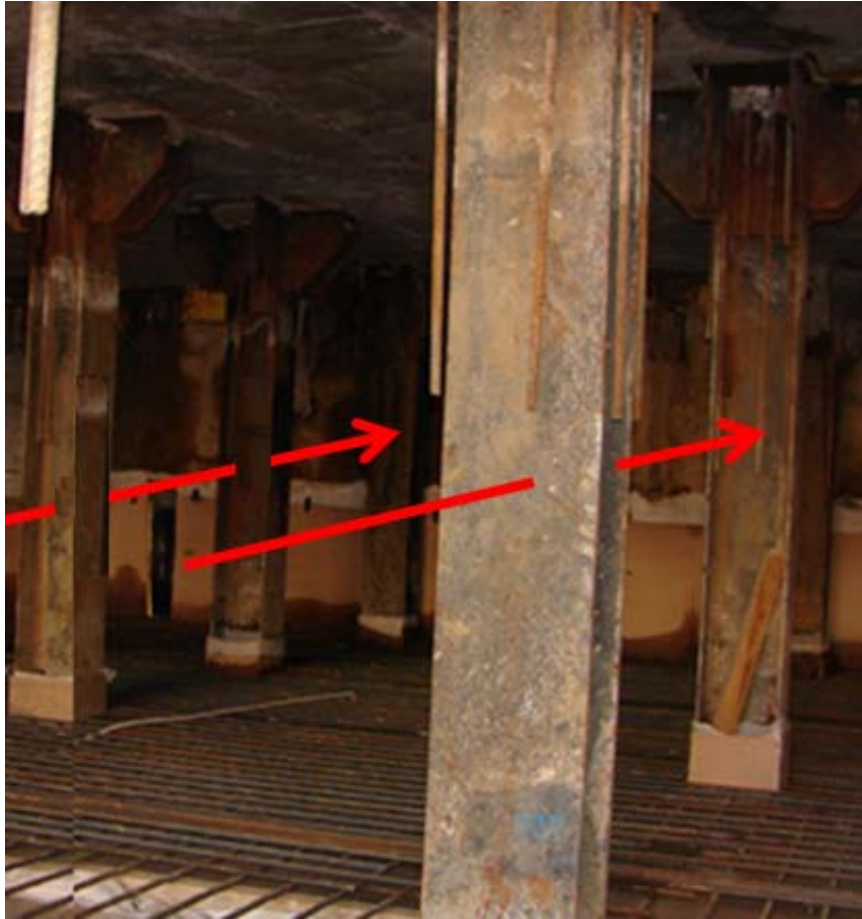


Figure 6.28 – LOS Intersection with Several Aligned Objects

In other situations when the LOS ray does not detect any object, the area of interest is then approximated with a set of rays cast away from the line of sight by an angle (i.e. 10 degrees) smaller than half the field of view angles used in the first collision detection step (Section 6.3.2). This process is presented in Figure 6.29, which depicts a user (i.e. inspector) looking at several objects such as columns, beams, etc. and each intersecting a different object.

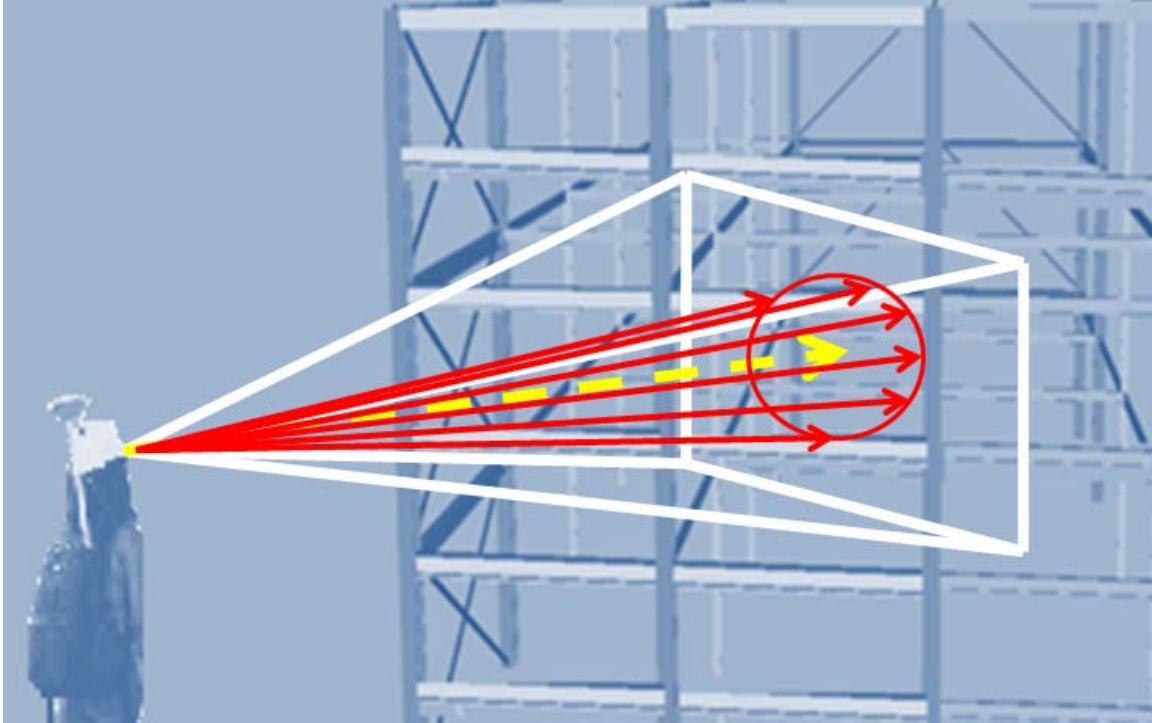


Figure 6.29 – Prioritizing Objects of Interest with Rays Cast away from the Line of Sight

If a construction scene is complicated with different and complex geometrical shapes of objects, then the *Intersector* traverses its target scene graph and tests nodes bounding spheres against the *Line segments*. If an intersection hit is encountered with the bounding sphere, the test can then become more fine-grained and each child node can be tested for an intersection until the leaf geometry node is reached, then data on the collisions detected can be stored such as pointers to node, position of intersection, etc.

Additional details on how to create and load the viewing frustum in a graphical scene, test it for intersection with CAD objects, and prioritize the identified visible objects using geometric interference analysis techniques are covered in Appendix B.

6.4. Summary and Conclusions

In this chapter, the author has described methods to interpret, with high-precision, the user's fully qualified spatial context. The chapter first presented an overview of computer graphics coordinate systems. Then it described, based on the mobile user's tracked location and head orientation, all the different geometrical computations to define the visible volume of space at a particular time. The region of space was conceptually thought of to be similar to an avatar's viewing volume in a computer graphics application or virtual reality world, often called the viewing frustum. Finally, the chapter introduced different collision detection techniques to test for interference between both frustum and CAD objects, and accurately determine all contextual relevant elements.

In order to illustrate the developed methods in this chapter, several validation tests have been conducted in outdoor and indoor environments (Chapter 8).

6.5. References

Adams, B., Keiser, R., Pauly, M., Guibas, L., Gross, M., and Dutre, P. (2005). “Efficient Raytracing of Deformable Point-Sampled Surfaces”, In Proceedings of the 2005 Eurographics Conference, 677–684.

Assarsson, U., and Stenström, P. (2001). “A Case Study of Load Distribution in Parallel View frustum Culling and Collision Detection”, In Lecture Notes in Computer Science, 663.

Cameron, S. (1991). “Approximation Hierarchies and S-Bounds”, In Proceedings of the Symposium on Solid Modeling Foundations and CAD/CAM Applications, Austin, TX, 129-137.

Cohen, J., Lin, M., Manocha, D., and Ponamgi, M. (1995). “I-collide: An Interactive and Exact Collision Detection System for Large-Scale Environments”, In Proceedings of the ACM Interactive 3D Graphics Conference, 189-196.

Drugge, M. (2005). OpenSceneGraph, Advanced – available at:
ftp://148.187.224.51/out/uvaretto/GraphicsReference/Lecture2-OSG_Advanced.pdf
(Accessed February 18, 2009).

Foley, J. D., Van Dam, A., Feiner, S. K., and Hughes, J. F. (1995). Computer Graphics: Principles and Practice, Addison-Wesley, Reading, MA.

Gottschalk, S., Lin, M. and Manocha, D. (1996). “Obb-Tree: A Hierarchical Structure for Rapid Interference Detection”, In Proceedings of the ACM Siggraph, 171-180.

Hubbard, P.M. (1995). “Collision Detection for Interactive Graphics Applications”, In Proceedings of IEEE Transaction on Visualization and Computer Graphics, 218-230.

Hudson, T., Lin, M., Cohen, J., Gottschalk, S., and Manocha, D. (1997). “V-Collide: Accelerated Collision Detection for VRML”, In Proceedings of the VRML Conference, 119-125.

Kamat, V. R. (2003). “VITASCOPE: Extensible and Scalable 3D Visualization of Simulated Construction Operations”, Ph.D. Dissertation, Department of Civil and Environmental Engineering, Virginia Tech, Blacksburg, VA.

Lin, M., and Gottschalk, S. (1998). “Collision Detection Between Geometric Models: A Survey”, Proceedings of the IMA Conference on Mathematics of Surfaces - available at:
<http://www.cs.unc.edu/~dm/collision.html> (Accessed: December 20, 2006).

- Macey, J. (1999). "Ray-tracing and other Rendering Approaches", Lecture Notes, MSc Computer Animation and Visual Effects, University of Bournemouth.
- McConnell, J. (2005). Computer Graphics: Theory into Practice, Jones & Bartlett, Sudbury, MA.
- Montalvo, F.S. (1979). "Human Vision and Computer Graphics", In the Proceedings of SIGGRAPH Conference, 121-125.
- Clark, R. N. (2007). Notes on the Resolution and Other Details of the Human Eye – available at: <http://www.clarkvision.com/imagetdetail/eye-resolution.html> (Accessed February 20, 2009).
- NPS OSG Tutorials Website, Intersection Tests – available at: <http://faculty.nps.edu/jasullivan/osgTutorials/osgIntersect.htm> (Accessed June 18, 2006).
- OpenSceneGraph Website, Hierarchy of osg::Util Classes – available at: <http://openscenegraph.sourceforge.net/documentation/OpenSceneGraph/doc/doc++/osgUtil/HIER.html> (Accessed June 18, 2006).
- OpenSceneGraph Website, Introduction to OpenSceneGraph – available at: <http://www.openscenegraph.org/projects/osg/wiki/About/Introduction> (Accessed November 18, 2005).
- O'Rourke, M. (2003). Principles of three-dimensional computer animation, 3rd Ed., W. W. Norton & Company, Inc, New York, NY.
- Permadi, F. (2005). Ray-Casting Tutorial– available at: <http://www.permadi.com/tutorial/raycast/index.html> (Accessed June 18, 2006).
- Quinlan, S.E. (1994). "Efficient Distance Computation Between Non-Convex Objects", In Proceedings of International Conference on Robotics and Automation, 3324-3329.
- Samet, H. (1989). Spatial Data Structures: Quadtree, Octrees and Other Hierarchical Methods, Addison Wesley, Reading, MA.
- Stroustrup, B. (2009). The C++ Programming Language–available at: <http://www.research.att.com/~bs/C++.html> (Accessed February 10, 2009).
- Tropp, O.A., Shimshoni, T.I., and Dobkin, D.P. (2006). "Temporal Coherence in Bounding Volume Hierarchies for Collision Detection", International Journal of Shape Modeling, 12(2),159-178.
- Woo, M., Neider, J., and Davis, T. (1997). OpenGL Programming Guide, 2nd .Ed. Addison-Wesley, Reading, MA.

Chapter 7

Contextual Information Retrieval and Visualization

7.1. Introduction

This chapter describes the fourth component of the developed framework, contextual information retrieval and visualization. Having identified contextual objects with high-precision (Chapter 6), specific information of interest can then be retrieved and interactively presented to a mobile user.

The information-intensive nature of construction projects requires the site personnel to have on-demand access to construction project data such as plans, drawings, schedules, specifications, etc. The flow of information, among parties involved in the project, is important to avoid problems and delays, and to save time on construction projects. The unprepared and dynamic nature of a construction site, and the on-site work difficulties, also necessitate the use of intelligent ways to support on-site construction personnel.

Standard product models and databases have been facilitating the exchanging, sharing and delivery of pertinent construction information, as well as enhancing work efficiency and collaborative processes (Halfawy et al. 2001, Aziz et al. 2005).

This chapter evaluates the capability of the CIMsteel Integration Standards (CIS/2) product model (Lipman and Reed 2003), and project information systems such as Microsoft Access databases (Allison and Berkowitz 2005) for improving traditional, and manual information retrieval processes and automatically retrieving contextual information and presenting it to construction site personnel.

7.2. Information Retrieval Process from MS Access Project Databases

In this section, a MS Access project database was designed as an option for demonstrating information retrieval capabilities developed in this research for use in AEC applications. The goal is to enable efficient information sharing between different project parties across the life-cycle of a constructed building facility.

To achieve this objective, a building was completely described in one coherent model, from which users can extract all the information they need, and to which they can add information they contribute (Figure 7.1).



Figure 7.1 – A Sample Database Building Model

In this case, where lots of data need to be presented to the user, it was found in this research that it is preferable to separate data and user interface concerns, so that changes to the user interface do not impact the data handling, and that the data can be reorganized without changing the user interface. Therefore, the Model-View-Controller (MVC) architectural pattern (Holub 1999) was adopted to solve this problem and decouple data access and business logic from data presentation and user interaction, by introducing an intermediate component: the controller. MVC consists thereby of three different parts: Model, View and Controller (Figure 7.2).

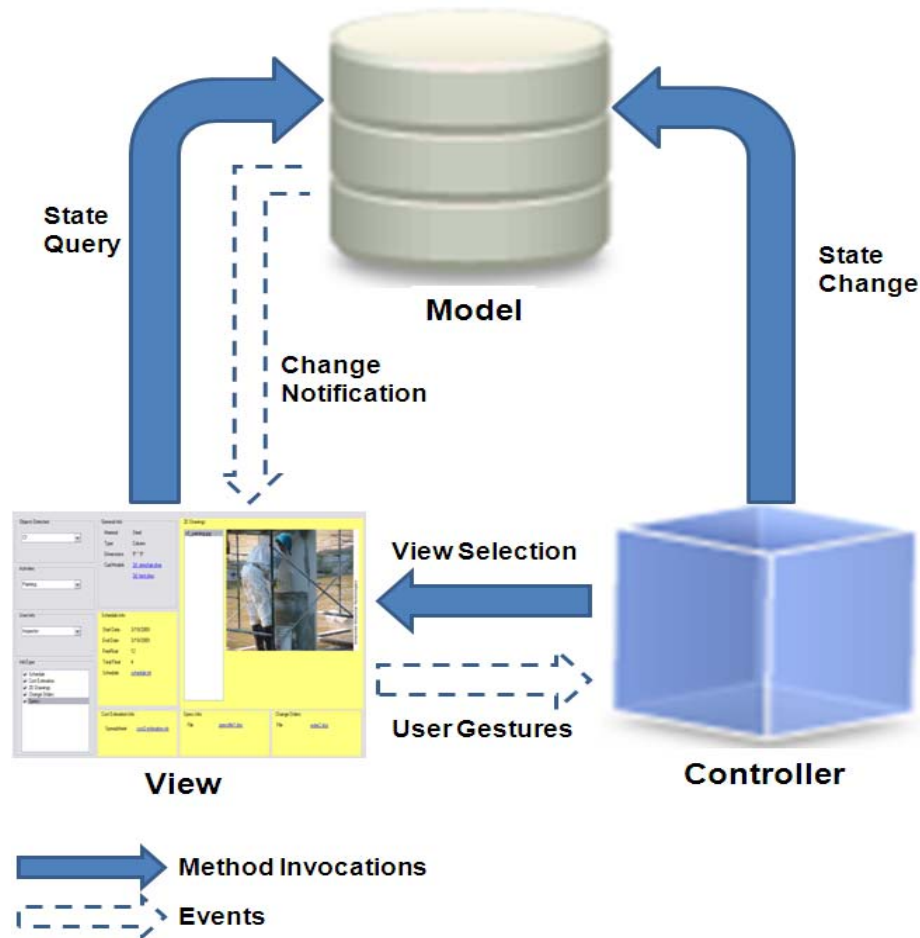


Figure 7.2 – The Model-View-Controller Architecture.

- The model is the part of the component that encapsulates the application's data structures, i.e. MS Access database tables. It often provides routines to manage and manipulate this data in a meaningful way in addition to routines that retrieve the data from the model. In general, the underlying data access technique should be encapsulated in the model. In this way, if, for example, the application is to be moved from one system to another system covering a different construction project, the model is the only element that needs to be changed, not the view or the controller.

- The view is the part of the component that is used to render the data from the model in a manner that is suitable for user interaction. The view pulls data from the model (which is passed to it from the controller) and feeds the data into a template which is populated and presented to the user. The view does not cause the data to be modified; it only displays data retrieved from the model.
- The controller is responsible for responding to user actions. It determines what request is being made by the user and responds appropriately by triggering the model to manipulate the data and passing the model into the view.

As such, MVC's control flow (Figure 7.2) starts when the user interacts with the user interface in some way (user gestures, i.e. mouse event). Then, the controller handles the input event from the user interface, often via a registered handler or callback. The controller notifies the model of the user action, possibly resulting in a change in the model's state. A view uses the model indirectly to generate an appropriate user interface. The view gets its own data from the model. The model and controller have no direct knowledge of the view. The user interface waits for further user interactions, which restarts the cycle. Therefore, successful use of the MVC pattern isolates business logic (controller) from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other.

The next subsections cover the three different MVC components as used in this research to meet the information retrieval requirements of the proposed methodology.

7.2.1. MVC Model Component : Database Structure

Figure 7.3

As mentioned in Chapter 4, an object-relational database structure using MS Access was adopted. Using the C# programming language, the database model was structured to contain classes and methods to access and retrieve different types of information about specific building objects residing in the database tables. Following this approach, tables were created and each class in the model was set to map its own table in the database (). Each class contains object properties to define each field inside its mapped database table. For example, considering an instance of class *Object*, the statement *Object.ObjectID* allows to get the value of the property *ObjectID* which is stored into the *ObjectID* field of the mapped database table.

As depicted in Figure 7.3, the main database tables are 1) *Item* houses the different building objects or structural entities and their attributes like material, type, dimensions, etc., 2) *Activity* takes into account the multilayered nature of a construction *Item* and covers several *Item* construction activities (i.e. concrete pouring, reinforcement, finishing, curing, etc.), 3) *ImgFiles*, *ItemCadModels*, *Schedules*, *Specs*, *CostEstimation* and *Orders* each reflects a different type of information, and most importantly 4) *ItemtoActivityMapping* maps all information to item-activity pairs. This latter table stores instances of the other classes in the form of a join table containing several foreign keys. Those keys define the one-to-many (or many-to-one) relationships to other tables (Figure 7.4).

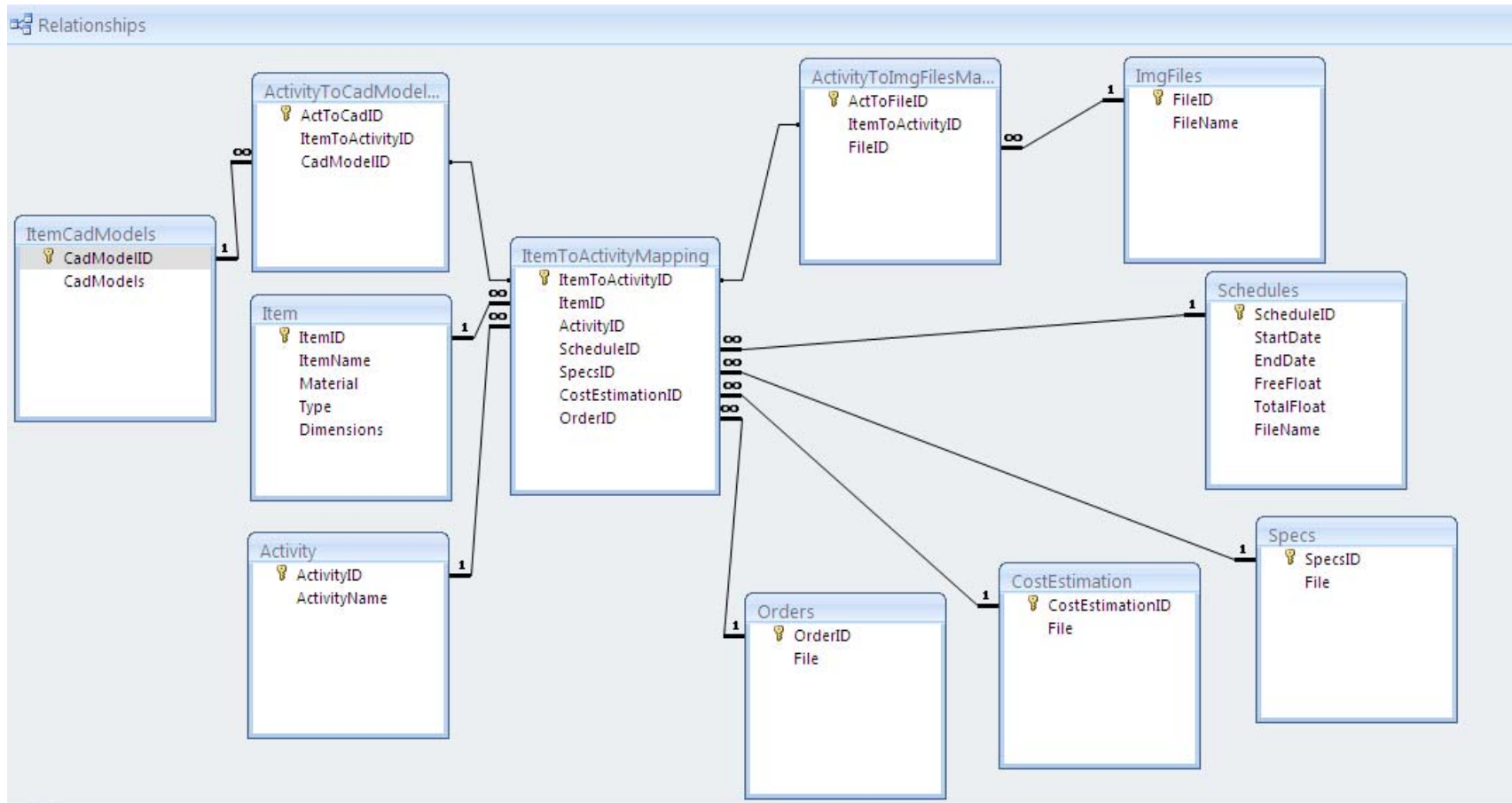


Figure 7.3 – MS Access Database Structure

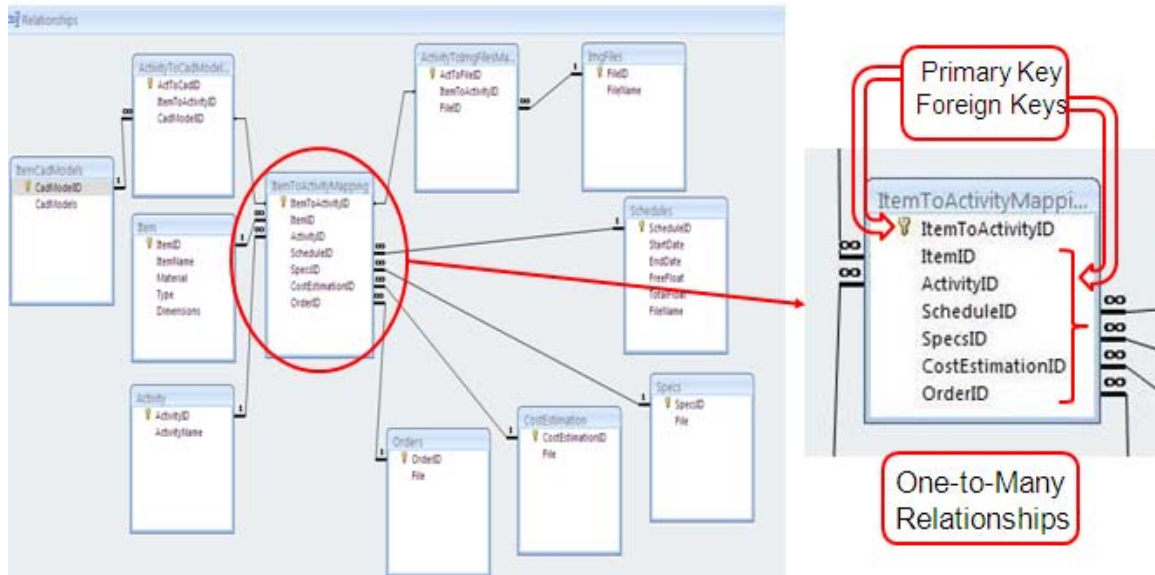


Figure 7.4 – Join Table and Relationship Keys

As mentioned in Chapter 4, a foreign key is a column defined in the table on the “many” side of the relationship (i.e. join table) that holds a key to the table in the “one” side of the relationship (i.e. each of the other tables). In Figure 7.4, *ItemToActivityID* is the join table primary key referencing a set of information for an item-activity pair. For example, for a certain *ItemToActivityID*, which refers to concrete pouring (i.e. *Activity.ActivityID=1*) of building item C1 (i.e. *Item.ItemID=10*), information on schedule, specifications, cost estimation, change orders, etc. can be automatically retrieved by using the foreign keys (*ScheduleID*, *SpecsID*, etc.), referring to respective tables and extracting all the properties values (Refer to the classes *DBController* and *ReflectionController* in Section 7.2.2). Additional details on the different database tables and mapped classes are provided in Appendix C.

7.2.2. MVC Controller Component of the Information Retrieval Process

The controller manages the communication of data and the rules used to manipulate the data to and from the model, in this case the Access database (Section 7.2.1). This was achieved by creating the following classes:

- *AppController*: It is the main class running the application and it contains 2 main functions: 1) *generateObjectsInfo* takes a vector of strings as a parameter, i.e. names of detected contextual building objects (Chapter 6), and 2) *Run* which is called to display the user interface (Section 7.2.3) .
- *ReflectionController*: This class was created to implement dynamic mapping functionality by taking advantage of the reflection-oriented programming which is a functional extension to the object-oriented programming paradigm (Sullivan 2001). The emphasis of the reflection-oriented paradigm is dynamic program modification, which is determined and executed at runtime. For instance, data or class objects can be created based on the underlying database using many code generation tools. However, it is time-consuming to manually and explicitly set all properties as well as get property values for several objects of different classes. Reflection can avoid code redundancy and save time by dynamically extracting objects (classes) from the tables that are returned from the database. Using this method, there is no need to use a code generator to generate the code but rather to loop through the columns in a table and dynamically match the column names to the property names of the object. In this case, since reflection works at runtime, a

convention was initially adopted where the name of a class object was set to have the same name of the mapped database table and the properties of the object are nothing but the columns in that same table (Section 7.2.1). For example, instead of having to manually define all *Activity* object properties (i.e *Activity.ActivityID*, *Activity.ActivityName*, etc.), another fast and dynamic way using reflection is to loop over query results (check *DBController* class below), get the object property name and value then pass them to the *SetProperty()* function after calling it. This whole reflection process happens in *DBController* after each SQL query statement and applies to all class objects as follows:

```
DataTable dt = ConvertDataReaderToDataSet(dataReader);

Type tp = typeof(Item);
refController = new ReflectionController(tp);

foreach (DataRow row in dt.Rows)
{
    foreach (DataColumn col in dt.Columns)
    {
        string field = col.ToString();
        string val = row[col].ToString();

        refController.SetProperty(tp.GetProperty(field),val);
    }
}
```

- *DBController*: This class is responsible for interacting with the MS Access database and retrieving requested information. This was first achieved by using a Microsoft Application Programming Interface (API) called OLE DB (Object Linking and Embedding, Database) (MSDN Website 2009) and defining the connection string to display the data source name then connecting to the database as follows:

```

public DBController()
{
    try
    {
        connection = new OleDbConnection (@"Provider=
Microsoft.Jet.OLEDB.4.0;
Data Source=.\db\InfoRetrievalDB.mdb;Persist
Security Info=False");

        connection.Open();
        AppData.dbController = this;
    }
    catch(Exception ex)
    {MessageBox.Show(ex.Message.ToString());}
}

```

After connecting to the database, a second step comprised retrieving stored information using methods called queries. Queries provide the capability to manipulate and combine data from multiple tables and place specific conditions on the data retrieved. Most users rely on queries because they only need a certain group of records or fields at any one time, i.e. records that adhere to a specific set of criteria are retrieved. For example, a query can be developed to retrieve general design information about specific items identified to be in the user's field of view. The query was built and executed by attributing the *ItemName* in the SQL *SELECT* statement to the identified contextual building entity string variable *itemName* as follows:

```

command = new OleDbCommand();
command.Connection = connection;
command.CommandText = "SELECT * FROM Item WHERE ItemName=?";
parameter = new OleDbParameter("ItemName",itemName);

```

A record set was then created as an instance of the *OleDbDataReader* class, item properties obtained using reflection, all handles closed, and then data retrieved:

```

private OleDbDataReader dataReader;
command.Parameters.Add(parameter);
dataReader= command.ExecuteReader();
DataTable dt = ConvertDataReaderToDataSet(dataReader);

Type tp = typeof(Item);
refController = new ReflectionController(tp);

foreach (DataRow row in dt.Rows)
{
    foreach (DataColumn col in dt.Columns)
    {
        string field = col.ToString();
        string val = row[col].ToString();

        refController.SetProperty(tp.GetProperty(field),val);
    }
}

dataReader.Close();
Item item = (Item)refController.getObj();

```

- *FileController*: This class is responsible for anything related to the input/output (I/O) library, in particular handling files. This includes reading, streaming and displaying to the user files such as CAD drawings, images, Primavera and Word files, and Excel spreadsheets.

7.2.3. MVC View Component: User Interface

As mentioned earlier, there is always more to an information system than a database. Other programs are responsible for communication and for managing user interfaces. In this case, it is the MVC View component using a variety of classes to display information on the Windows form and make it visible to the user. Figure 7.5 reflects the application interface visualization process from a point of view of an observer, i.e. mobile user.

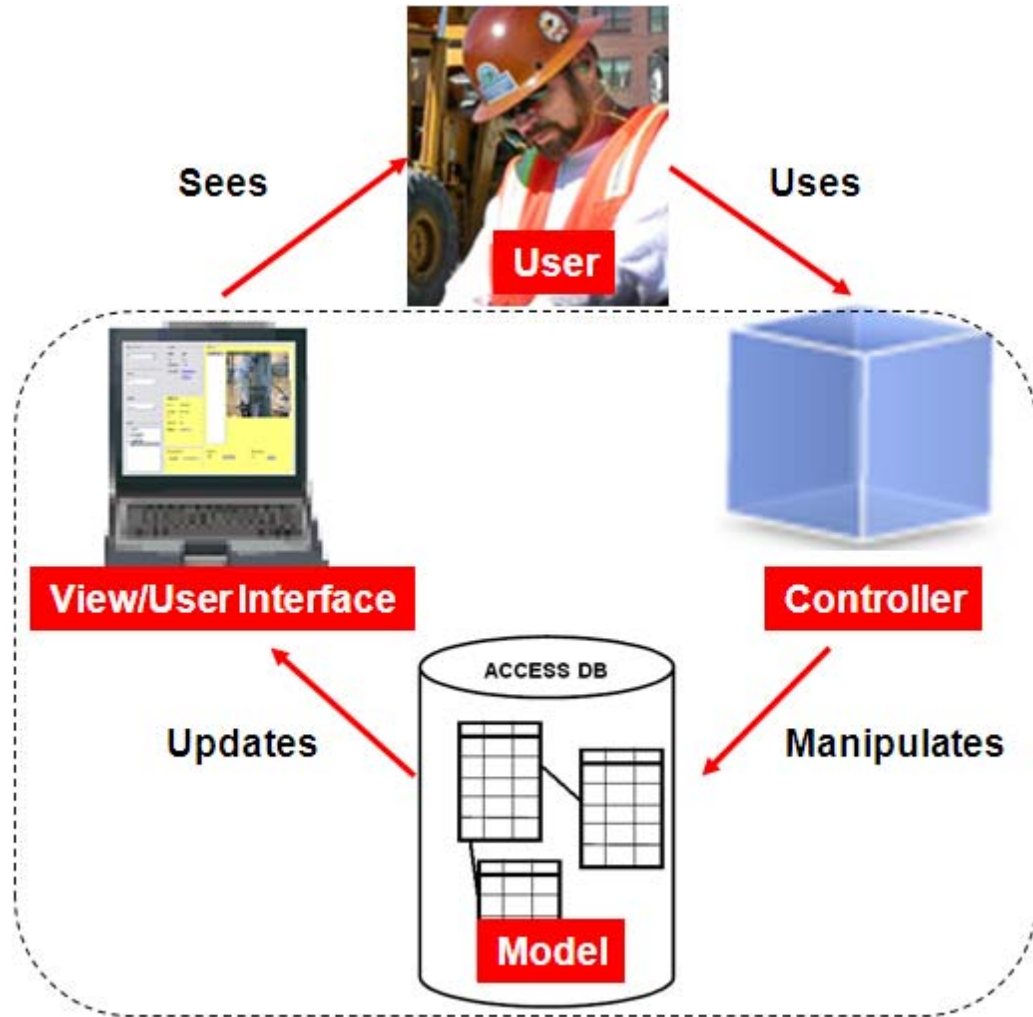


Figure 7.5 – Retrieval Application from a User Point of View

Therefore, the View component consists of several classes implementing the non-DBMS parts of the final system. Those are control classes that use indirectly the database model by sending function calls to the controller in order to generate and manipulate the final Windows form. As depicted in Figure 7.6, each of the group boxes (*Object Detected*, *Activities*, *User Info*, *General Info*, *Schedule Info*, etc.) constitutes a control class. Details on the controls classes are provided in Appendix C.



Figure 7.6 – User Interface Information Retrieval Application (1)

The information is displayed on the user interface in the following order:

First, detected contextual building objects (Chapter 6) are automatically presented in a drop-down menu. The user chooses one object (i.e. C1) and proceeds to choose one activity (i.e. painting) for which s/he is interested in getting information about at this particular time. Following this, a list of different information types (schedule information, cost estimation information, etc.) is ranked or sorted according to the user's function, in this case an inspector. Then the user has the choice to pick one or more types of information and have them displayed. In Figure 7.7, the user opted to only have the drawings, schedule, and cost estimation information retrieved and displayed as links to CAD drawings, primavera files or Excel spreadsheets.

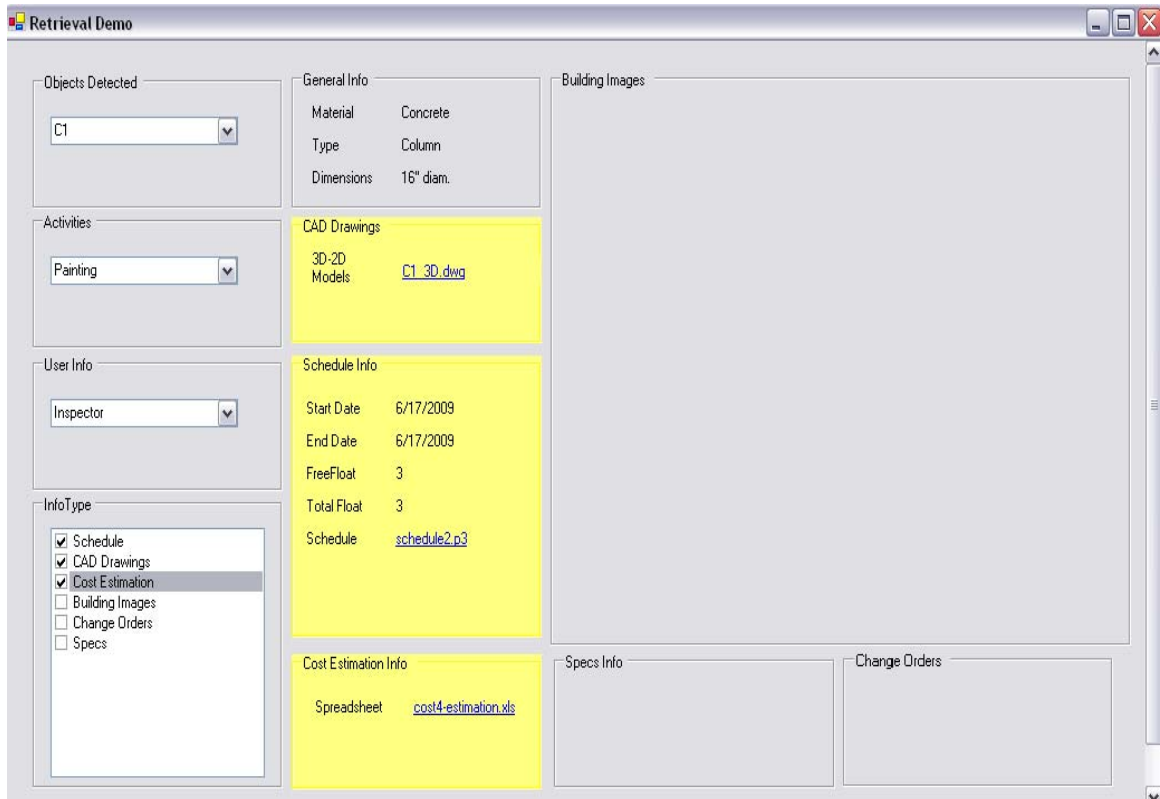


Figure 7.7 – User Interface Information Retrieval Application (2)

Depending on the nature of a project and information involved in it, and given that an MVC approach was adopted, the whole retrieval concept presented in this section can be easily extended to a more advanced database system such as Microsoft SQL.

7.3. Contextual Information Retrieval from CIS/2

Product Models

In this section, the utility of the interoperable CIS/2 product model (Lipman and Reed 2003) is presented as a suitable data structure to represent cross-referenced building data

for the evaluation of the designed and proposed automated information retrieval technique.

As a mobile user is moving on the jobsite, relevant information on visible and identified entities at a particular instant in time can be retrieved from the underlying product model. Therefore, the primary contribution of the research presented in this section lies in gathering as-built information from the ambient environment and then cross-referencing recognized construction entities with an existing CIS/2 building product model (Figure 7.8) for automated information support on construction sites.

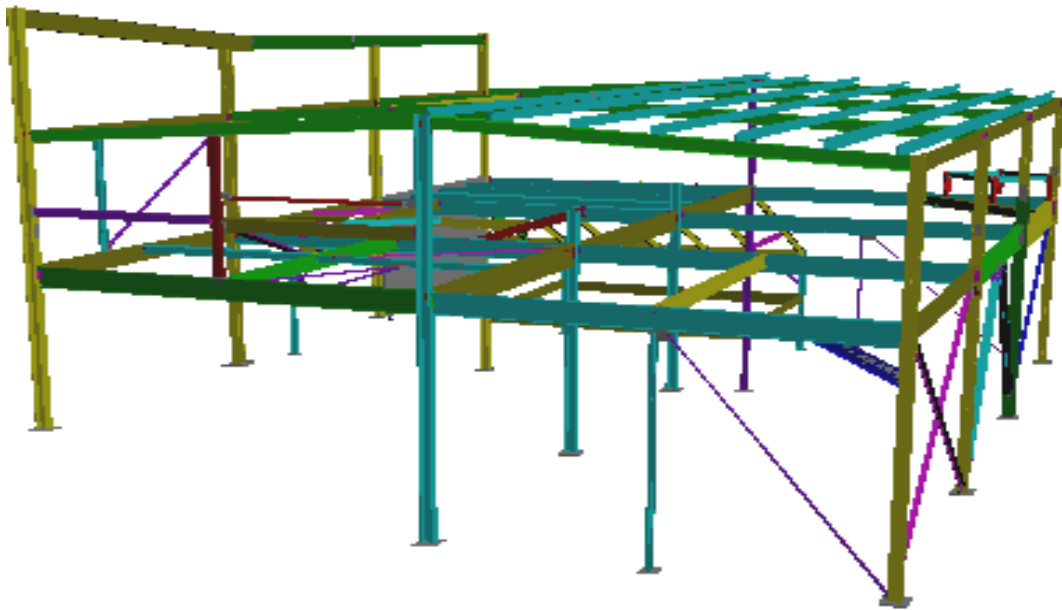


Figure 7.8 – CIS/2 Product Model of a Building Structural Frame
[Image courtesy of Mr. Robert Lipman, NIST]

As mentioned in Chapter 4, CIS/2 deals with information about the steel structure throughout its analysis, design, detailing, and fabrication life cycle (Reed 2002). The

geometry of the structure is only one property of the product data model. Other attributes of the structure include how parts are combined into assemblies and structures, analysis loads and reactions, material types, connection details, associations between members and drawings, and modification history. Figure 7.9 shows a sample of a CIS/2 file illustrating how steel parts are combined into assemblies and structures.

```
#43= LOCATED_PART(92,'92','brace',#42,#33,#20);
#42= (COORD_SYSTEM('', 'Part CS', $, 3)
      COORD_SYSTEM_CARTESIAN_3D(#40)COORD_SYSTEM_CHILD(#18));
#40= AXIS2_PLACEMENT_3D('Part axes', #34, #38, #36);
#34= CARTESIAN_POINT('Part origin', (0., 0., 0.));
#38= DIRECTION('Part z-axis', (0., 0., 1.));
#36= DIRECTION('Part x-axis', (1., 0., 0.));
#18= COORD_SYSTEM_CARTESIAN_3D('', 'Assembly CS', $, 3, #17);
#17= AXIS2_PLACEMENT_3D('Assembly axes ', #11, #15, #13);
#11= CARTESIAN_POINT('Assembly origin ', (720., 540., 120.));
#15= DIRECTION('Assembly z-axis ', (-0.37139068, 0., 0.92847669));
#13= DIRECTION('Assembly x-axis ', (0.92847669, 0., 0.37139068));
#33= (PART(.UNDEFINED., $)PART_PRISMATIC()PART_PRISMATIC_SIMPLE(#21, #26, $, $)
      STRUCTURAL_FRAME_ITEM(92, '92', 'brace')STRUCTURAL_FRAME_PRODUCT($)
      STRUCTURAL_FRAME_PRODUCT_WITH_MATERIAL(#27, $, $));
#21= SECTION_PROFILE(1, 'W14X158', $, $, 5, .T.);
#26= POSITIVE_LENGTH_MEASURE_WITH_UNIT
      (POSITIVE_LENGTH_MEASURE(258.48791), #3);
#3= (CONTEXT_DEPENDENT_UNIT('INCH')LENGTH_UNIT()NAMED_UNIT(#1));
#1= DIMENSIONAL_EXPONENTS(1., 0., 0., 0., 0., 0., 0.);
#27= MATERIAL(1, 'GRADE50', $);
#20= LOCATED_ASSEMBLY(92, '92', 'brace', #18, $, #19, #10);
#18= COORD_SYSTEM_CARTESIAN_3D('', 'Assembly Coordinate System', $, 3, #17);
#17= AXIS2_PLACEMENT_3D('Assembly axes ', #11, #15, #13);
#11= CARTESIAN_POINT('Assembly origin ', (720., 540., 120.));
#15= DIRECTION('Assembly z-axis ', (-0.37139068, 0., 0.92847669));
#13= DIRECTION('Assembly x-axis ', (0.92847669, 0., 0.37139068));
#19= ASSEMBLY_MANUFACTURING(92, '92', 'brace', $, $, $, $, $, $, $);
#10= STRUCTURE(1, 'cis_2', 'Unknown');
```

Figure 7.9 – CIS/2 File Structure Example

The file is represented in the Standard for the Exchange of Product model data (STEP) format (ISO 10303:1992, STEP Website 2008). Each CIS/2 entity instance is assigned a number indicated by a pound (#) sign. The name of the CIS/2 entity then appears in upper case letters. Every entity has a number of fields that contain text strings, numeric values,

Boolean values, references to other entities, or null values indicated by a dollar sign. For instance, the #43 CIS/2 entity, together with all its sub-entities, encompasses information about the steel part location within the assembly. The #33 instance refers to information about the steel part properties, such as cross-section, length, and material, and the #20 instance points to the location of the assembly within the structure.

One major step in using CIS/2 in the retrieval process consisted of using the NIST CIS/2 to Virtual Reality Modeling Language (VRML) translator (Lipman 2002) to convert CIS/2 files to their corresponding VRML (Virtual Reality Modeling Language) representation. The intermediate conversion of CIS/2 files to VRML was adopted for the following reasons (Kamat and Lipman 2006). First, the VRML translator provides a visual interface to the underlying CIS/2 data thereby providing a means to visualize the represented steel structure in a 3D virtual world. Second, implementing a VRML file parser that can traverse the described scene graph to extract member information is relatively straightforward compared to the implementation of a full-fledged CIS/2 file parser. Thus, the parsing and interpretation of CIS/2 information contained in a converted VRML file, yield information for each steel member contained in the represented structural steel frame, including but not limited to the name and the geometry of each member.

Any CIS/2 file can provide data structures for multiple levels of detail ranging from frames and assemblies to nuts and bolts. Therefore, prior to translating the CIS/2 file to a VRML file, the user can specify what type of information is of interest to him in order to

expedite the translation process and only get the needed information. Figures 7.10 and 7.11 are snapshots of the CIS/2-VRML translator showing its different information level options.

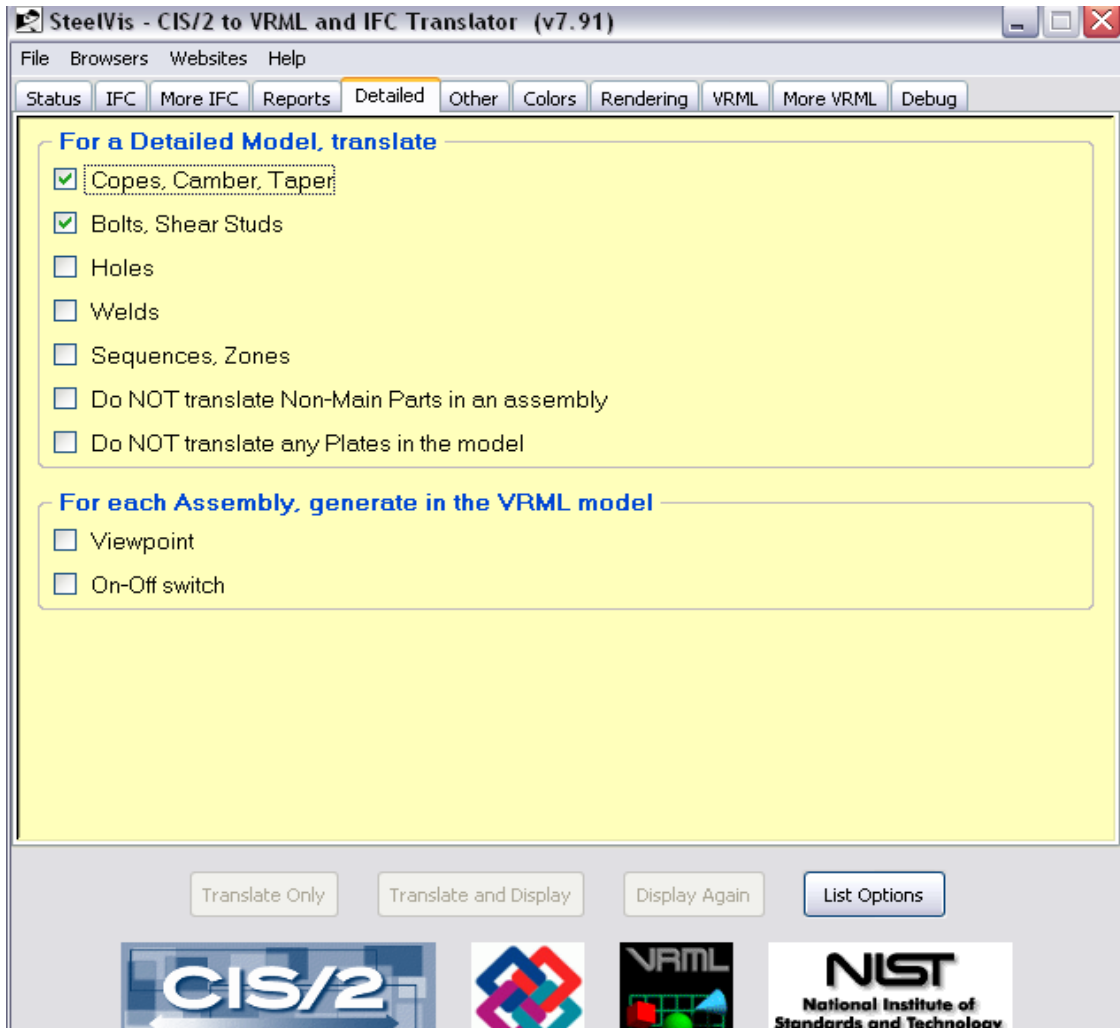


Figure 7.10 – CIS/2-VRML Translator Snapshot (1)

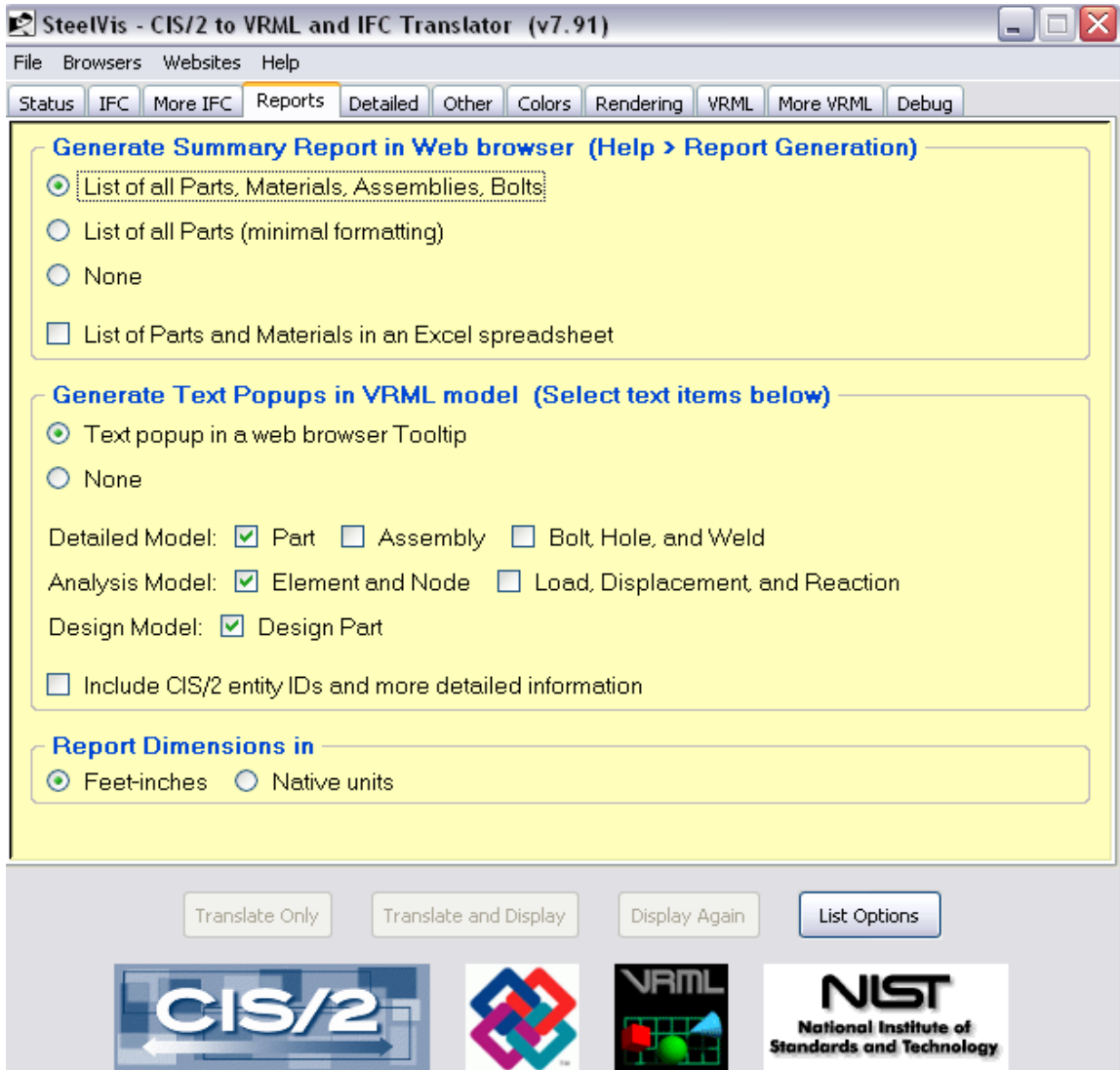


Figure 7.11 – CIS/2-VRML Translator Snapshot (2)

After getting the desired translated VRML file, the next step included using a VRML reader-writer or loader application (Fisher 2005) that automatically parses all of the VRML nodes (together with the attached information), and then translates back to OSG nodes. This is mainly taken care of in the VRML loader by creating several callback actions (i.e. *cbAction*) for the VRML nodes as follows:

```

SoCallbackAction cbAction;
cbAction.addPreCallback(SoGroup::getClassTypeId(), preSeparator, this);
cbAction.addPostCallback(SoGroup::getClassTypeId(), postSeparator,
this);
cbAction.addPreCallback(SoVRMLTransform::getClassTypeId(),preTransform,
this);
cbAction.addPostCallback(SoVRMLTransform::getClassTypeId(),
postTransform, this);
cbAction.addPreCallback(SoVRMLShape::getClassTypeId(), preShape, this);
cbAction.addPostCallback(SoVRMLShape::getClassTypeId(), postShape,
this);
cbAction.addPreCallback(SoVRMLPositionInterpolator::getClassTypeId(),
prePositionInterpolate, this);
cbAction.addPostCallback(SoVRMLPositionInterpolator::getClassTypeId(),
postPositionInterpolate, this);
cbAction.addPreCallback(SoVRMLOrientationInterpolator::getClassTypeId()
, preOrientationInterpolate, this);
cbAction.addPostCallback(SoVRMLOrientationInterpolator::getClassTypeId(
), postOrientationInterpolate, this);
cbAction.addPreCallback(SoTexture2::getClassTypeId(), preTexture,
this);
cbAction.addPreCallback(SoLight::getClassTypeId(), preLight, this);
cbAction.addTriangleCallback(SoVRMLGeometry::getClassTypeId(),
addTriangleCB, this);
cbAction.addLineSegmentCallback(SoVRMLGeometry::getClassTypeId(),
addLineSegmentCB, this);
cbAction.addPointCallback(SoVRMLGeometry::getClassTypeId(), addPointCB,
this);

```

Once the conversion is performed, OSG nodes (building objects) are tested for interference using intersection techniques (Chapter 6), and then information attached to the nodes is automatically retrieved for each of the detected contextual steel objects.

7.4. Contextual Information Visualization

In all described case studies, retrieved contextual information was visualized and interactively presented to the user through mobile wearable displays, devices and tools, some of which are part of the mobile visualization platform developed by Behzadan and Kamat (2007). Figure 7.12 presents the hardware setup used in the design of this prototype.



Figure 7.12 – Hardware Components of UM-AR-GPS-ROVER
[Behzadan and Kamat, 2007]

The components and interactive devices that were used for visualization purposes, are primarily a Head Mounted Display (HMD), in particular i-Glasses SVGA Pro video see-through HMD, a laptop computer, a keyboard and a touchpad (Figure 7.12). The main computing task is performed by the laptop computer secured inside the backpack and the rendered graphical scene is displayed to the user through the HMD (Figure 7.13) installed in front of the hard hat.

During regular site operation, it may often be necessary for the user to change computer or component settings, or to select a new mode of operation after one task is complete. Therefore, it is desirable to allow the user to do this without having to temporarily halt operation and remove the backpack to physically access the laptop. This was achieved by selecting components such as a touchpad and a miniature keyboard to accommodate user

input and provide full interactivity with the laptop, even though the user is not physically accessing the laptop (Behzadan and Kamat 2007).



Figure 7.13 – i-Glasses SVGA Pro video see-through Head Mounted Display (HMD)

The above prototype was also used to take advantage of its AR features in presenting retrieved information to mobile users on construction sites. AR is the superimposition of computer-generated graphics over the user's view of the real world. Subsets of the retrieved information can be interactively augmented onto the user's view to provide automated real-time information support and assist in decision-making tasks.

Besides using the aforementioned hardware prototype, in particular its HMD component, to interactively present retrieved information, a hands-free laptop holder was adopted in many of the experiments conducted (Figure 7.14) so that information is directly displayed on the laptop screen instead of the HMD.



Figure 7.14 – Hands-Free Laptop Holder

This laptop holder gives the ability to use the laptop with both hands free, to type and change settings while walking or standing. It can work with a laptop, notebook, or tablet computer of any size, and can be adjustable to fit a person of any height. It is quick and easy to put on and take off and ergonomically designed for comfort. It is truly mobile and portable, and there is no need for a laptop cart or stand. Most importantly, taking notes with a full size keyboard while having the ability to move around can be of great value to almost any mobile user.

Other lightweight head-mounted displays have been explored for their suitability and possible future integration within the proposed framework. Examples include wearable displays manufactured by LitEye (2007) and Inition (2006). Several models of displays are especially designed for rugged applications in harsh environments such as construction (Figure 7.15), and can be used with lightweight wearable computers.



Figure 7.15 – Lightweight Displays for Engineering Applications

7.5. Summary and Conclusions

The objective of this step of the research work was to develop methods for directly comparing relevant designed data in product models and project databases with the construction entities positively identified to be in a user's field of view at a given time instant, and then retrieving and visualizing contextual information. The author has devised retrieval methods using two types of project information systems, namely CIS/2 product models and MS Access databases.

In order to demonstrate the applicability of the adopted project databases and product models in a context-aware information delivery framework, experiments were conducted at NIST indoors, in the maze at the NIKE site, outdoors at the steel structure on the main campus, and at the University of Michigan indoors, in the Structural Engineering Laboratory (Chapter 8). The results of the experiments reflected how retrieving

contextual information from project databases in real-time can save time and improve work productivity in dynamic environments such as those found on construction sites.

Additional details on all retrieval algorithms are presented in Appendix C. The reader is recommended to refer to this Appendix to gain a better understanding of all the designed methods.

7.6. References

Allison, C.L., and Berkowitz, N. (2005). *SQL for Microsoft Access*, Edition: illustrated, Wordware Publishing, Inc., Plano, TX.

Aziz, Z., Anumba, C.J., Ruikar, D., Carrillo, P.M., Bouchlaghem., D.N. (2005). "Context aware information delivery for on-Site construction operations", Proceedings of the 22nd CIB-W78 Conference on Information Technology in Construction, Institute for Construction Informatics, Technische Universitat Dresden, Germany, CBI Publication No:304, 321-32.

CIS/2 Website, CIMsteel Integration Standards Release 2 – available at: <http://www.cis2.org/>. (Accessed June 15, 2006)

Halfawy, M., and Froese, T. (2001). "Leveraging information technologies applications in the Canadian AEC/FM Industry", In the Conference Proceedings of the Canadian Society of Civil Engineers, Victoria, BC, Paper A22.

Holub, A. (1999). "Building User Interfaces for Object-Oriented Systems", Java World – available at: <http://www.javaworld.com/javaworld/jw-07-1999/jw-07-toolbox.html> (Accessed February 10, 2009).

ISO 10303-42. (2000). "Industrial automation systems—Product data representation and exchange—Part 21: Integrated generic resource: Geometric and topological representation", ISO/IEC, Geneva, Switzerland (with Technical Corrigendum 1, 2001).

Kamat, V.R., and Lipman, R.R. (2006). "Evaluation of standard product models for supporting automated erection of structural steelwork", *Journal of Automation in Construction*, 16, 232-241.

Lipman, R.R., K.A. Reed. (2003). "Visualization of Structural Steel Product Models", *Electronic Journal of Information Technology in Construction*, 8, Royal Institute of Technology, Stockholm, Sweden, 43-50.

LitEye, Products and HMD Technology - available at: <http://www.liteye.com> (Accessed December 5, 2007)

MSDN Website, MSDN Library – available at: <http://msdn.microsoft.com/en-us/library/default.aspx> (Accessed March 15, 2009).

Reed, K.A. (2002). "Role of the CIMsteel integration standards in automating the erection and surveying of constructional steelwork", 19th International Symposium on Automation and Robotics in Construction (ISARC), NIST, Gaithersburg, MD, 15–20.

STEP Tools, STEP ISO10303, STEP Tools Inc. - available at:
<http://www.steptools.com/library/standard/2007> (Accessed July 8, 2008)

Sullivan, G. T. (2001). "Aspect-Oriented Programming using Reflection and Metaobject Protocols", 44(10), Communications of the ACM, 95-97.

Chapter 8

Validation of Location-Aware Information Retrieval Methodology

8.1. Introduction

The information presented in this chapter presents the validation exercises for the research contributions described in individual chapters of this dissertation that each focus on a specific aspect of the research. In particular, the designed methodology and the implemented software and hardware framework were critically evaluated using the following criteria:

- Accuracy in 3D spatial user tracking in outdoor and indoor environments
- Precision in visible components identification and detection
- Reliability in contextual information retrieval and visualization

In addition to validating each individual aspect of the research during its progress, the overall ability of the location-aware information retrieval framework to ubiquitously track and provide mobile users with relevant contextual information was validated.

8.2. Validation of Accuracy in 3D Spatial User Tracking and Precision in Contextual Objects Identification

In order to validate the ability of the proposed location-aware methodology to track mobile users based on their latest position and head orientation data and precisely identify contextual objects, several experiments were conducted at the University of Michigan, the National Institute of Standards and Technology (NIST), and Disaster City at Texas A&M University. In all experiments, the user was allowed to navigate in different outdoor and indoor environments, and freely change the head orientation.

The objective of this part of the validation exercise was to ascertain that the tracking devices are fully capable of obtaining the user's real time position and head orientation, and that the application is robust enough to update the contents of the user's viewing frustum based on these values and then detect and identify visible objects in real-time.

As discussed in Chapter 5, the tracking methods developed in this research, have been implemented in many outdoor and indoor environments using positioning technologies, namely the Global Positioning System (GPS), Wireless Local Area Networks (WLAN), Ultra-Wide Band (UWB) and Indoor GPS, and head orientation tracking devices. Continuous communication with the tracking technologies to obtain real-time spatial data of the user was very critical since the viewing frustum or more specifically the pyramid coordinates had to be constantly and virtually updated during the course of the user's navigation in the specified coverage area. All conducted indoor and outdoor tracking experiments are described in the next subsections.

8.2.1. GPS-Based Outdoor Experiment

The first validation experiment was conducted outside the G. G. Brown (GGB) laboratory building at the University of Michigan. This building houses several engineering departments including Civil, Mechanical, and Chemical engineering. Figure 8.1 shows the aerial view of this outdoor experiment.



Figure 8.1 – Aerial View of the GGB Outdoor Experiment

The objective of this experiment was to simulate an inspector surveying the building, and evaluate whether the different sections of the building can be automatically identified based on the user's spatial context. During the experiment, the user's position and orientation were continuously obtained from the developed mobile computing backpack's GPS and magnetic tracker respectively (Behzadan and Kamat 2007). Selected snapshots

of both third- person virtual and real views taken during the conducted experiment are shown in Figure 8.2.

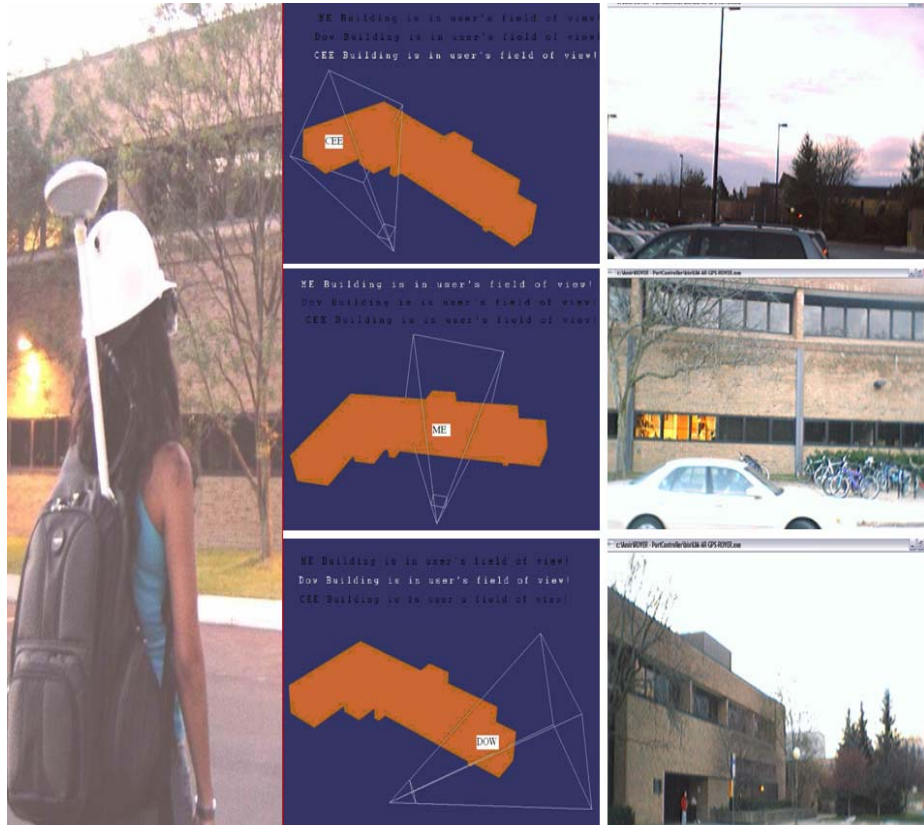


Figure 8.2 - Snapshots of Simulated Building Inspection

The near and far plane distances were assumed to be 1 and 100 m respectively, and both the HFOV and VFOV angles were chosen to be 45 degrees (typical values for computer graphics applications). Based on this information, the eight coordinates of the truncated pyramid (i.e. viewing frustum) were computed. The frustum was then aligned with a 3D VRML model of the GGB building and geometric interference tests were performed. Each time the user moved on the site, the intersection between the current frustum and sections of the building was computed and reported. It was observed that as the user was

moving around the building, the computer was correctly interpreting which building segment was in the view at each time instant.

8.2.2. WLAN-Based Indoor Experiments

Another set of validation experiments was conducted using a WLAN based positioning system called Ekahau manufactured by the Finnish company Ekahau Inc. (Ekahau Website 2007).

A first experiment was conducted at the Construction Engineering Laboratory located in the GGB building at the University of Michigan (Figure 8.3).



Figure 8.3 – GGB Construction Engineering Laboratory (University of Michigan)

Full wireless coverage is provided from university access points deployed in many locations in the College of Engineering (CoE) buildings across campus. In this case,

signals were detected from three access points deployed in the laboratory at different corners (Figures 8.4).

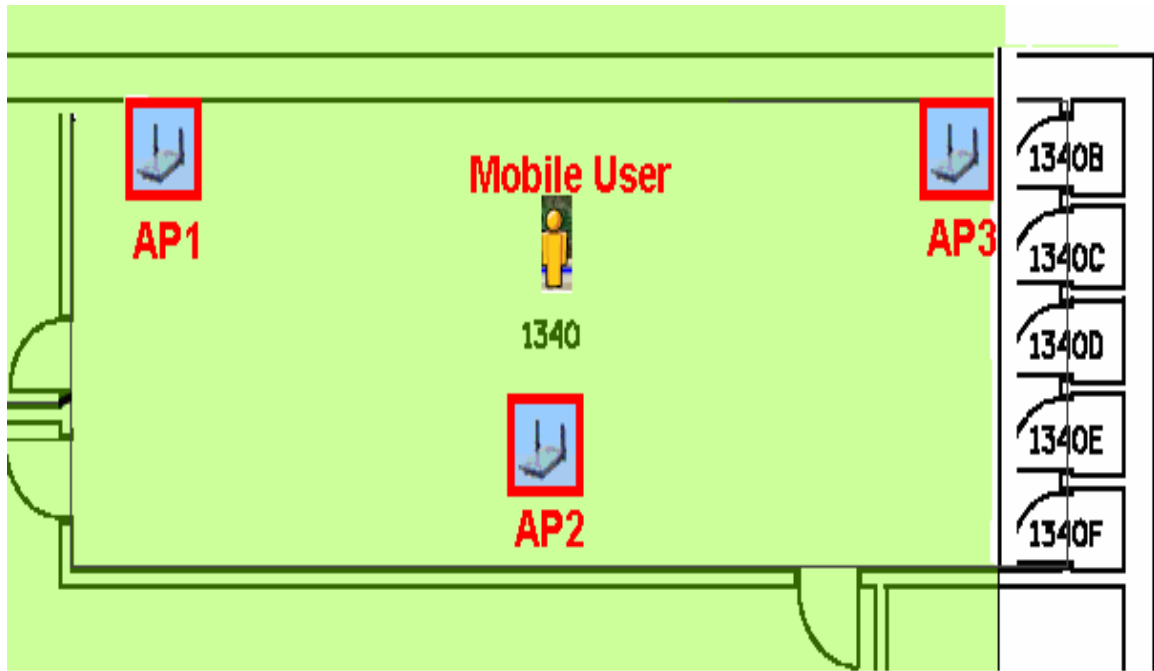


Figure 8.4 – Plan View of the Testing inside the Construction Engineering Laboratory (Room 1340 GGB Building)

Walking paths or tracking rails were then drawn on top of the laboratory floor map and calibration was performed by taking many sample points at different reference locations on the tracking rails (Figure 8.5).

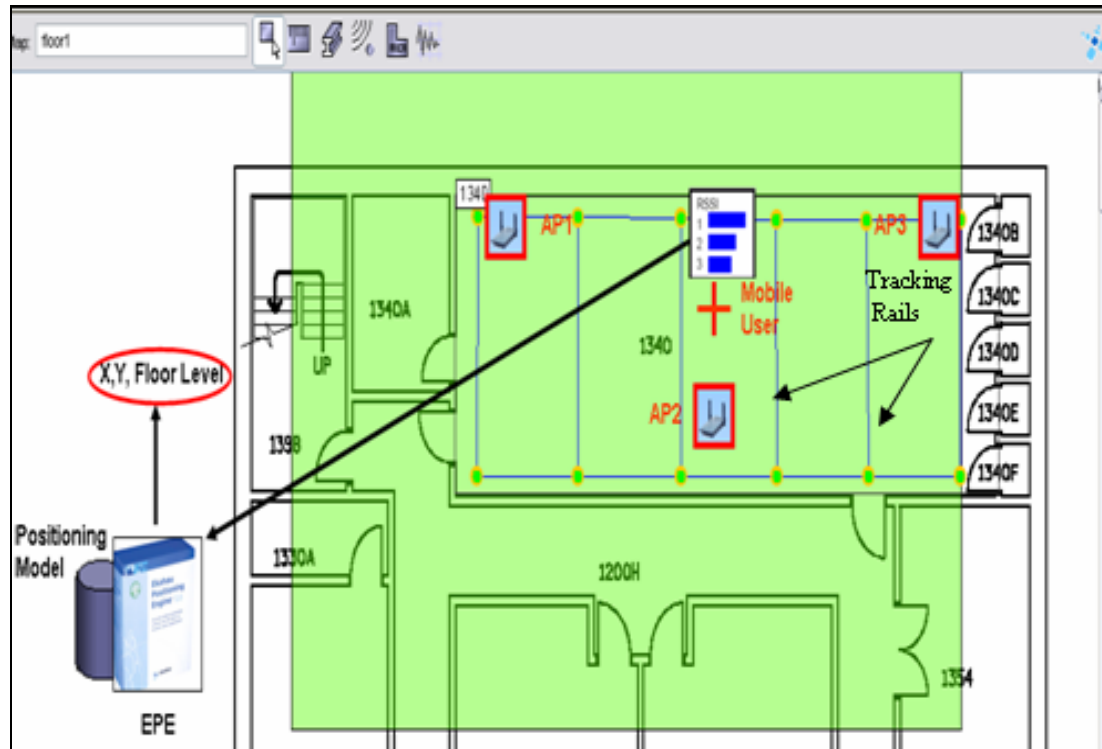


Figure 8.5 – Ekahau Calibration and Testing inside the Construction Engineering Laboratory (University of Michigan)

The objective was to track a mobile user's position and head orientation in real-time as s/he walked inside the laboratory. Based on Chapter 5, this was achieved by combining the positioning information (X, Y, floor level) obtained from Ekahau (Ekahau Website 2007) and orientation information (roll, yaw, and pitch) obtained from the orientation in one tracking application .

The second step in the experiment encompassed interpreting the volume of space visible to the tracked mobile user. The near and far plane distances were assumed to be 0.5 and 2 m respectively, and both the HFOV and VFOV angles were chosen to be 45 degrees. Based on this information, the eight coordinates of the viewing frustum were computed. Each time the user moved on the site, the geometric interference between the current

frustum and VRML models of objects or other users was computed, and then specific relevant objects were identified using raycasting (Chapter 6).

In order to visualize how the mobile user (the author in this experiment) is being continuously tracked in the laboratory using the tracked user's position and head orientation, a 3D environment with sufficient underlying computer graphics support to allow the manipulation of entities in a 3D scene was needed. As mentioned in Chapter 6, a computer graphics toolkit namely OpenSceneGraph (OSG) within Visual C++ .NET, was adopted (OpenSceneGraph Website 2005) (Appendix B).

Selected snapshots of virtual views taken during the experiments, conducted on the first floor of GGB Building (Construction Engineering Laboratory) are shown in Figures 8.6 and 8.7.

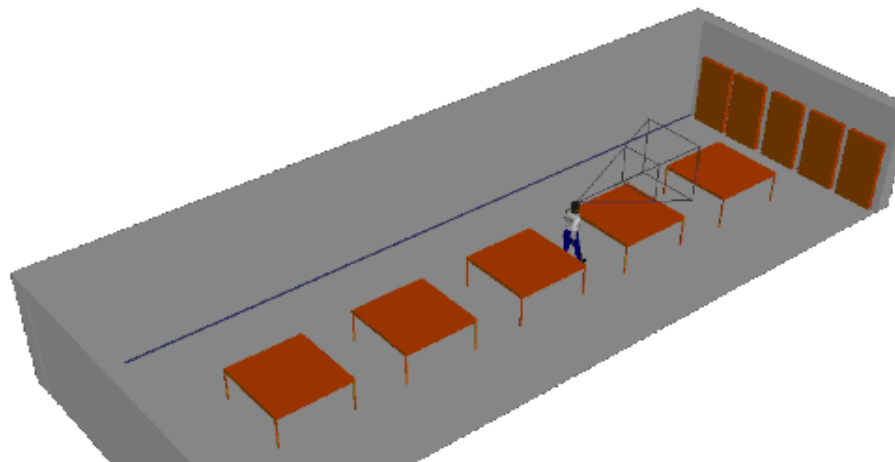


Figure 8.6 – Virtual Representation for Indoor Tracking of a Mobile User Using WLAN (Construction Laboratory, GGB, University of Michigan)

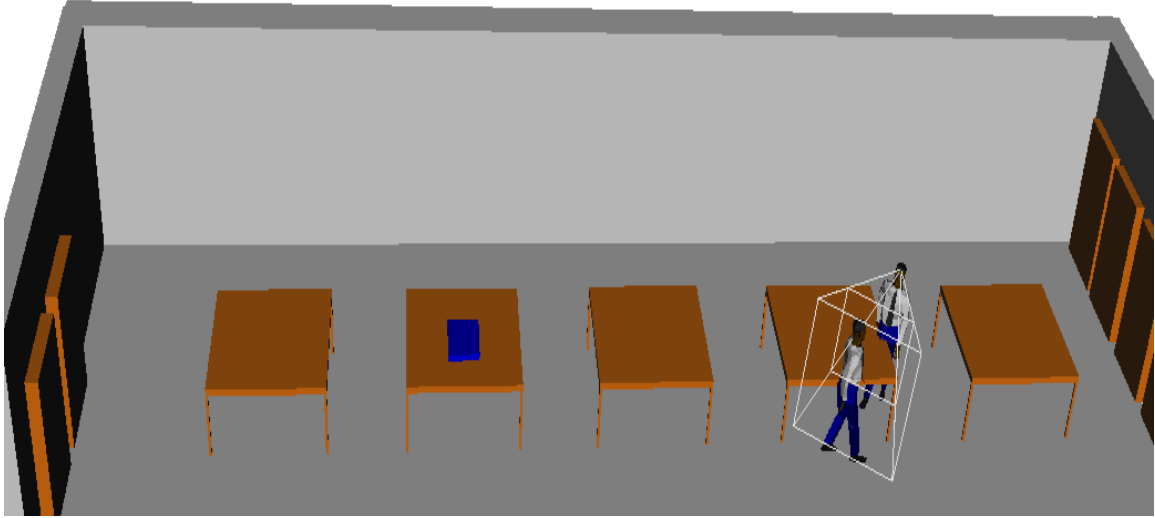


Figure 8.7 – Virtual Representation for a Geometric Interference Test in the Construction Laboratory

A similar experiment using Ekahau was conducted on the second floor of GGB building, namely the Civil Engineering Department (Figure 8.8).

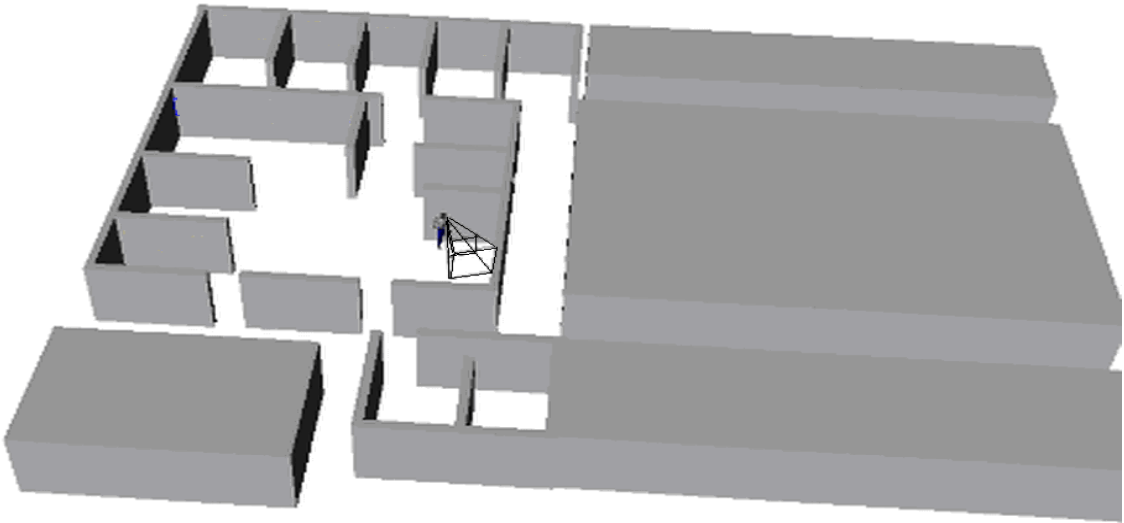


Figure 8.8 – Virtual Representation for Indoor Tracking of a Mobile User Using WLAN (Second Floor of GGB Building, University of Michigan)

Another indoor experiment was performed at Disaster City (Figure 8.9) as part of the response robot evaluation exercises for Urban Search-and-Rescue (US&R) conducted by the National NIST team, of which the author was a member. Disaster City is one of the most comprehensive emergency responses training facility available today. It is a 52-acre training facility designed to deliver the full array of skills and techniques needed by urban search and rescue professionals. As part of the Texas Engineering Extension Service (TEEX) at Texas A&M University, the facility features full-size collapsible structures that replicate community infrastructure, including a strip mall, office building, industrial complex, assembly hall/theater, single family dwelling, train derailment and three rubble piles (TEEX Website 2007).



Figure 8.9 – Aerial View of the Maze at Disaster City

These response robot evaluation exercises for US&R teams introduce emerging robotic capabilities to emergency responders within their own training facilities, while educating robot developers about the necessary performance requirements and operational constraints to be effective. Several of those exercises were specifically performed at the “maze” in the assembly hall/theater (Figure 8.9).

Similar to the earlier experiments, three access points were placed in the hall according to the layout on Figure 8.10. Then, as the user/robot was navigating in the maze, the Ekahau Engine updated the user’s location, and the x, y, and z coordinates were continually displayed as shown in the lower-right corner of Figure 8.11.

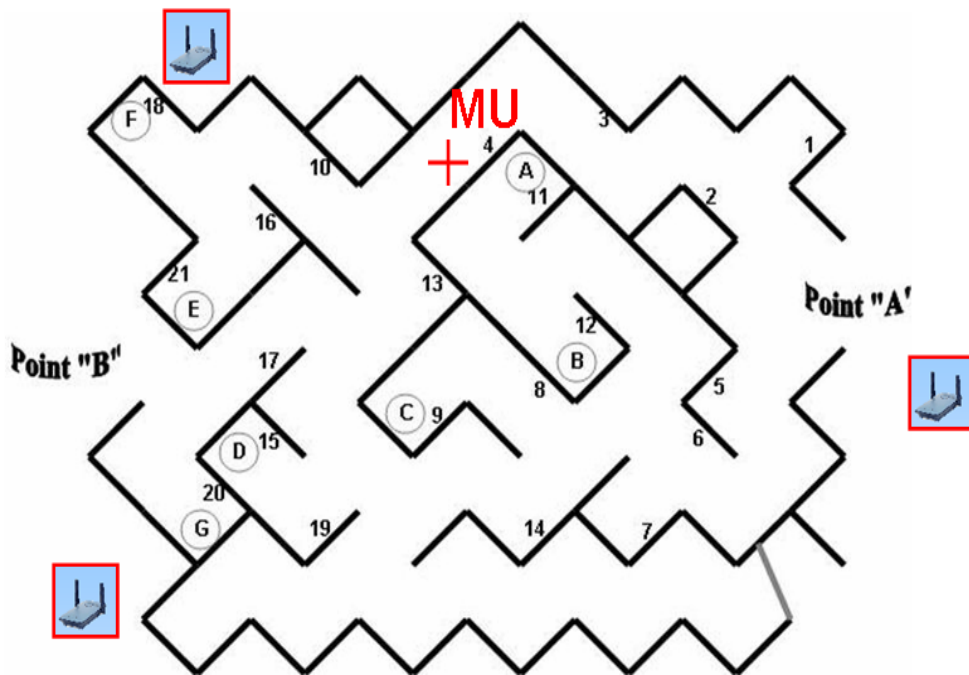


Figure 8.10 – Access Points Deployment around the Maze (Disaster City)

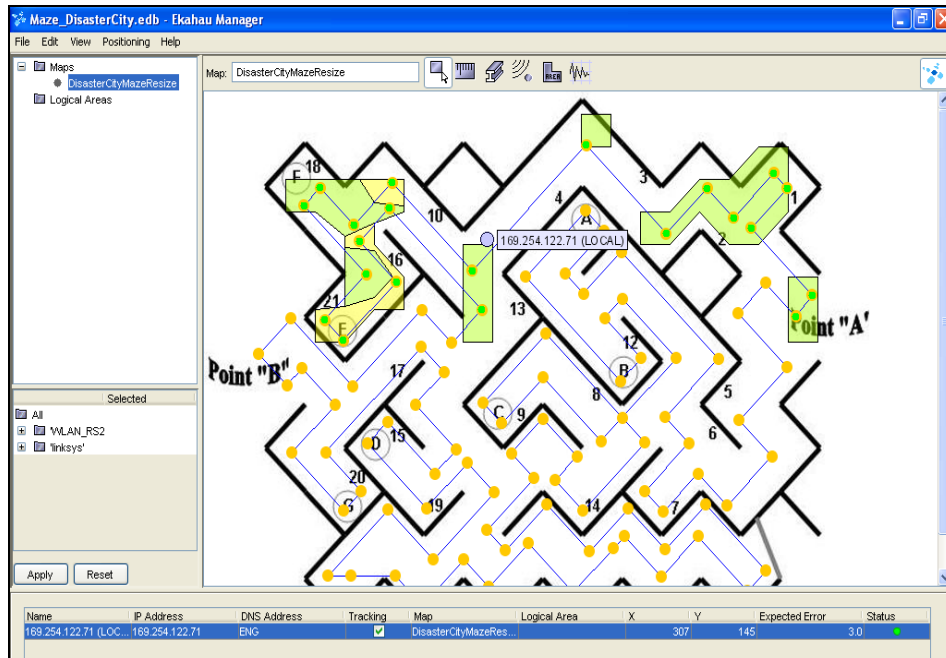


Figure 8.11 – Ekahau Positioning at the Maze (Disaster City)

An experiment similar to the one conducted at Disaster City was performed at NIST specifically in the “maze” at the former NIKE missile base barracks building adjacent to the main campus (Figure 8.12). The testbed used was the same; three access points were used and the user navigated around the “maze” and collected position information.



Figure 8.12– Maze at Nike Site (NIST)

The results of this set of WLAN experiments indicated that the Ekahau tracking system overall achieved a positioning accuracy of approximately 1.5-2m.

8.2.3. UWB-Based Indoor Experiments

Another set of experiments was performed using the UWB tracking system (Multispectral Solutions Website 2007). One of the validation experiments was conducted in the same “maze” at NIST.

Figure 8.13 is a plan view of the test setup (i.e. six receivers were deployed around the maze). The positioning values were used together (Chapter 5) with orientation values received from the magnetic tracker and both were integrated in the graphical OSG application to visualize (in 3D) how the user was moving in real-time around the maze and trying to identify objects in his field of view, in this case two columns located within the maze (Figure 8.14).

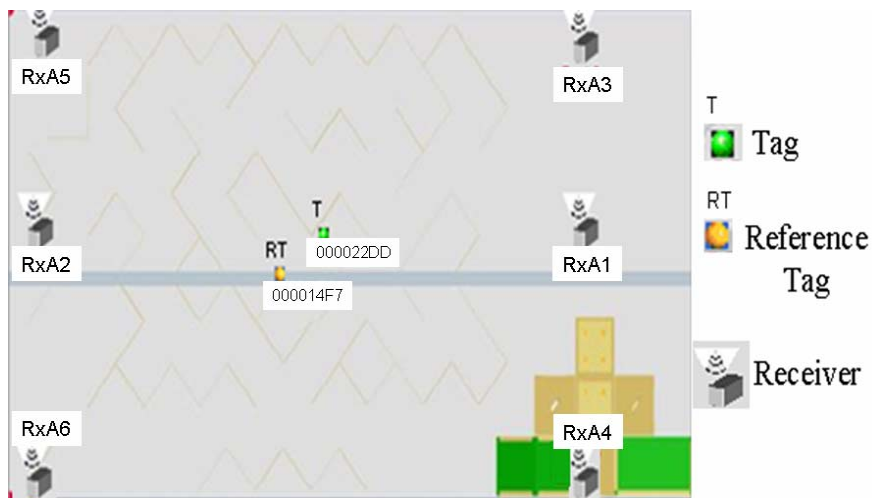


Figure 8.13 – Plan View of the UWB Receivers Setup inside the Maze (NIST)

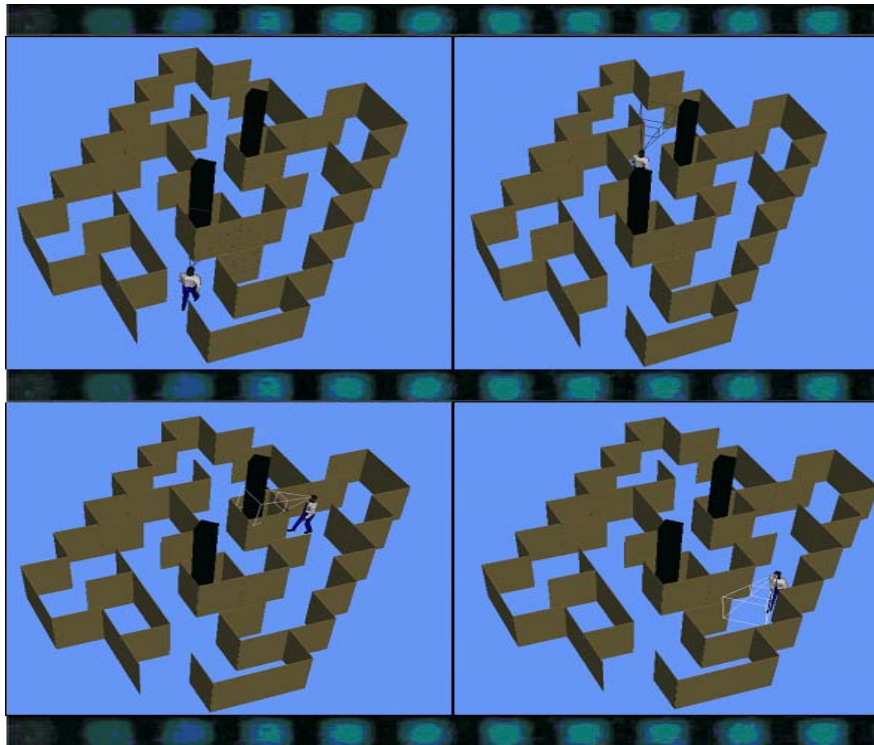


Figure 8.14 – 3D Snapshot Views of a UWB Tracked User inside the Maze

As noted previously, many of the indoor experiments were performed at Disaster City as part of the response robot evaluation exercises for US&R to introduce emerging robotic capabilities to emergency responders within their own training facilities. Besides the WLAN-based experiment conducted in the assembly hall where the maze is located, another experiment was also conducted in the same hall at Disaster City, but using the UWB DART system instead of the Ekahau tracking system. Six receivers were deployed in the hall as shown in Figure 8.15. Tags were placed on top of robots in order to track robots' 3D location. Figure 8.16 illustrates one of the Response Robot Evaluation Exercises conducted by an emergency responder at the maze. Figure 8.17 is a 3D representation of the robot navigation path as it was moving in real-time in the maze from Point A to Point B.

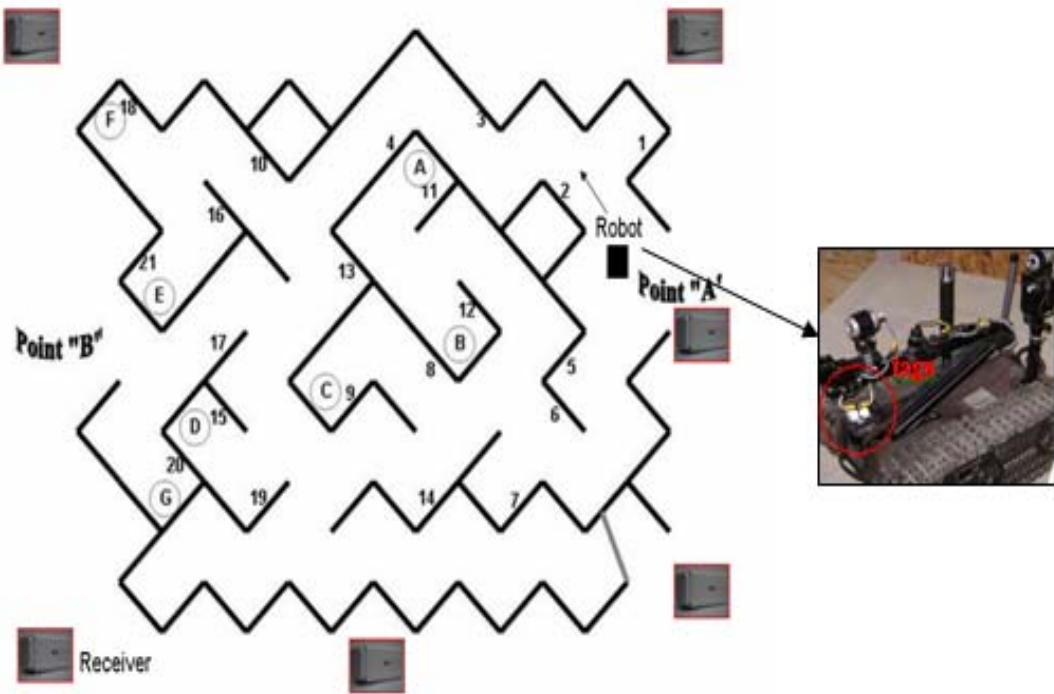


Figure – 8.15 - UWB Tracking System at the Maze (Disaster City)

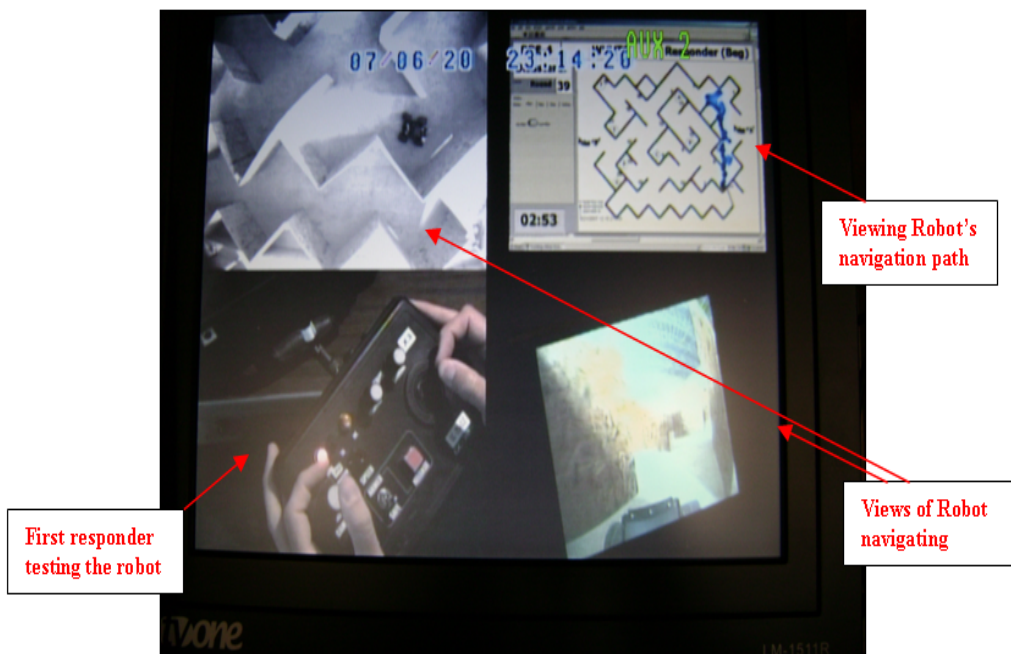


Figure 8.16 – Screen View of a Response Robot Evaluation Exercise (NIST)

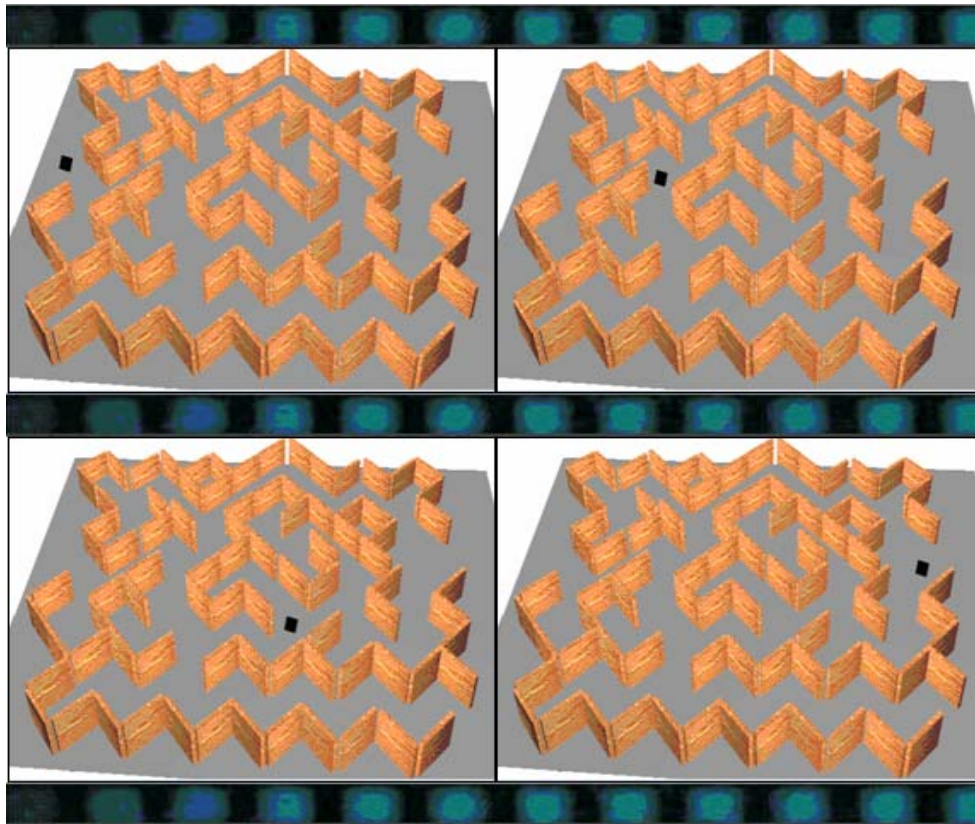


Figure 8.17 – 3D Snapshots of a UWB Tracked Robot at Different Locations in the Maze

The results of the described experiments using UWB based positioning at Disaster City and NIST indicated that the UWB Tracking system overall achieved an accuracy that fluctuated between 10 to 50 cm.

8.2.4. Indoor GPS-Based Indoor Experiment

Another experiment was conducted at NIST, inside the “maze” using the Indoor GPS (Kang and Tesar 2004, Metris Website 2007). Similarly, the goal of this experiment was to simulate a mobile user such as a construction engineer or inspector navigating around and surveying the building, and determine the extent to which the user’s spatial context

can be accurately and continuously tracked in order to precisely identify objects in the field of view.

In this case, four transmitters were deployed inside and around the area of the maze as shown in Figure 8.18, and one receiver and the orientation tracker were mounted on the mobile user who was navigating inside the maze.

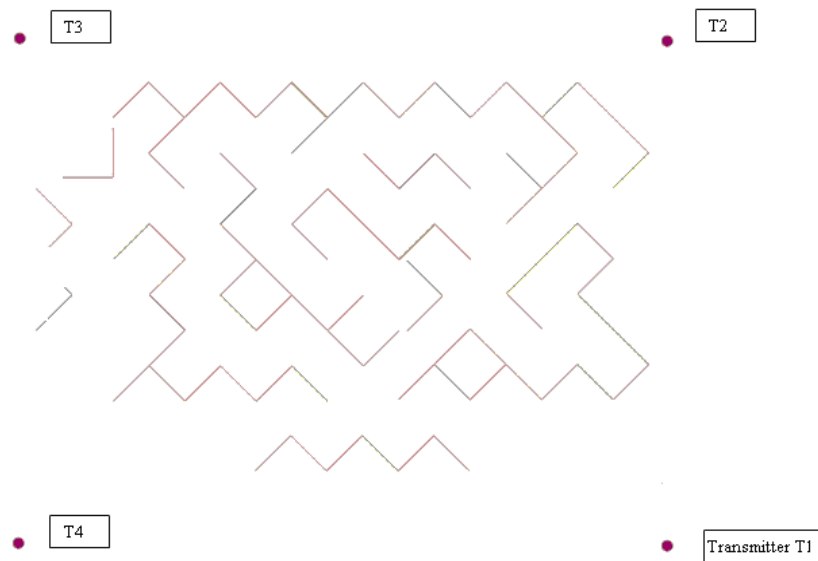


Figure 8.18 –Deployment of Laser Transmitters around the Maze

The user's position and orientation were continuously obtained from the Indoor GPS and magnetic tracker, and similar to the UWB-based experiment, the tracked values were used in the 3D OSG application to visualize the path of the user inside the maze.

Given that the maze covers an area of 13x8 m, the near and far plane distances were considered to be 1m and 3m respectively, and both the HFOV and VFOV were chosen to be 45 degrees.

Similar to the experiments in Figure 8.14, the eight coordinates of the truncated pyramid were computed using the developed method. Geometric interference tests were then performed and detection reported.

The results of the experiments indicated that the Indoor GPS tracking system consistently achieved a positioning uncertainty that fluctuated between 1 and 2 cm.

In all the previous experiments using the three different technologies, the obtained results demonstrated that tracking a mobile user's three-dimensional orientation in addition to the position is an effective way of increasing precision in the interpretation of the user's fully qualified spatial context.

8.3. Validation of Reliability in Contextual Information Retrieval and Visualization

As discussed in Chapter 7, information retrieval in the context of this research is defined as the ability of the proposed methodology to extract in real-time, information from product models and databases about contextual objects identified to be in user's field of view at a specific instant. In order to validate this aspect of the developed context-aware framework, an outdoor structural steel inspection operation and an indoor operation were simulated on NIST's main campus and in the "maze" respectively.

8.3.1. Structural Steel Inspection Operation

A test was conducted at NIST, specifically at the steel structure on the main campus (Figure 8.19). The goal of this experiment was to evaluate whether contextual information identified based on the user's spatial context can be retrieved from a CIS/2 product model as was described in Chapter 7.



Figure 8.19 – Steel Structure on NIST Main Campus

In this case, four Ultra Wide Band (UWB) receivers and one reference tag (Figure 8.20) were deployed around the area of the steel structure as shown in Figure 8.21, and one UWB tag and the orientation tracker were mounted on the mobile user who was navigating around the steel structure. As described in previous experiments, the user's position and orientation were continuously obtained from the UWB system and magnetic tracker.

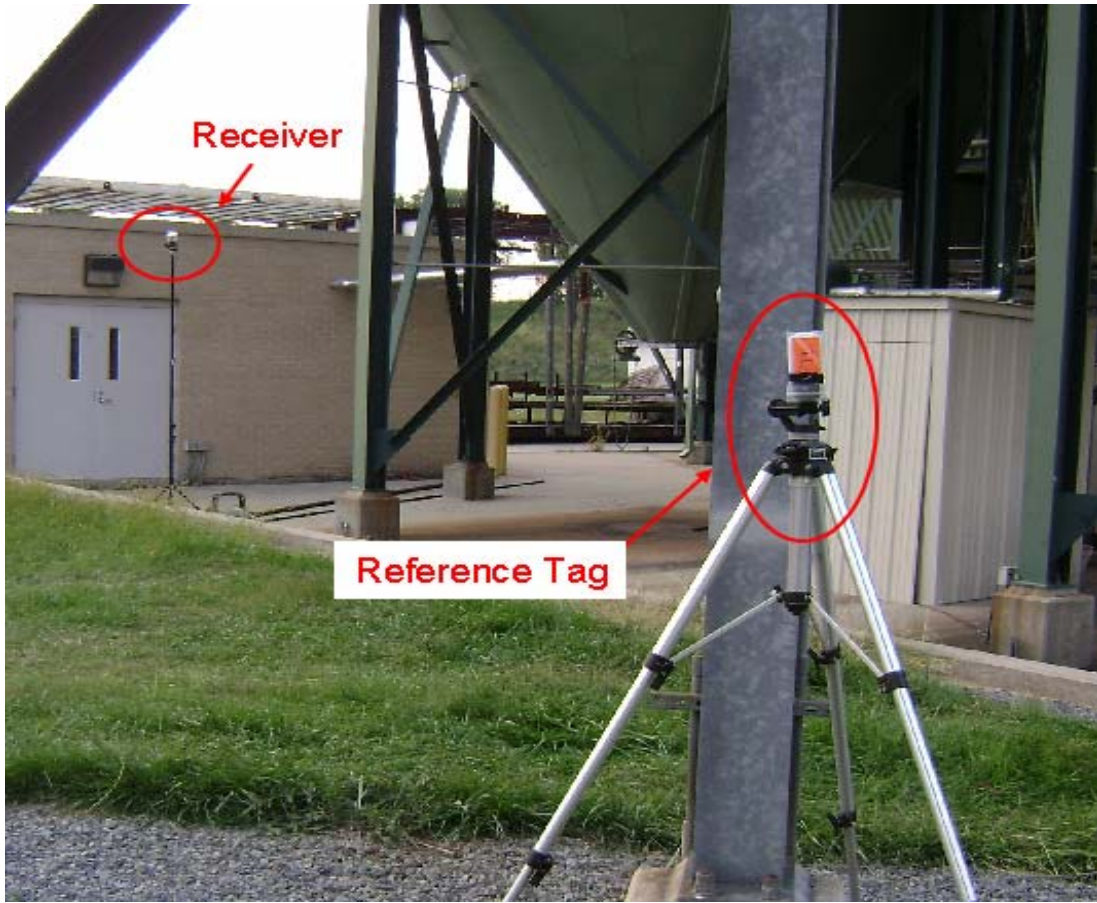


Figure 8.20 – UWB Receiver and Reference Tag Setup at the Steel Structure (NIST)

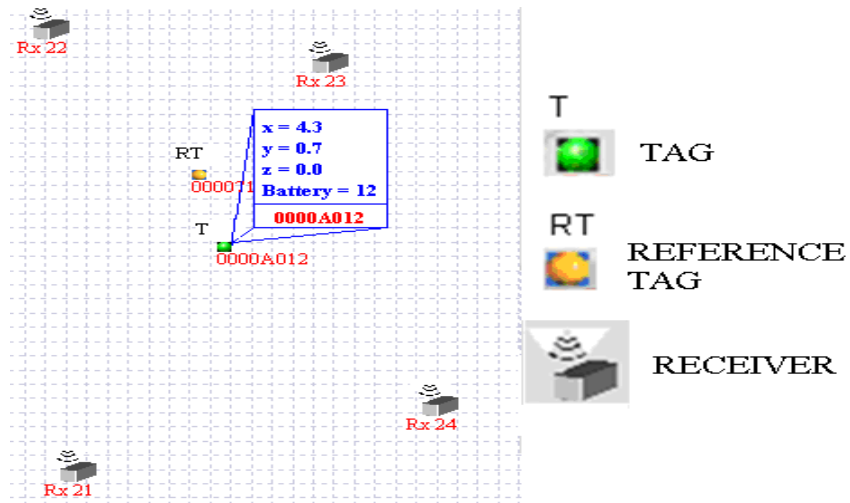


Figure 8.21– Deployment of UWB Receivers around the Steel Structure (NIST)

As the user moved on the site, geometric intersection tests between the computed frustum and virtual steel members were conducted and reported. Having identified specific steel members in the user's field of view, contextual information was then automatically retrieved from the converted CIS/2 model (Figure 8.22).

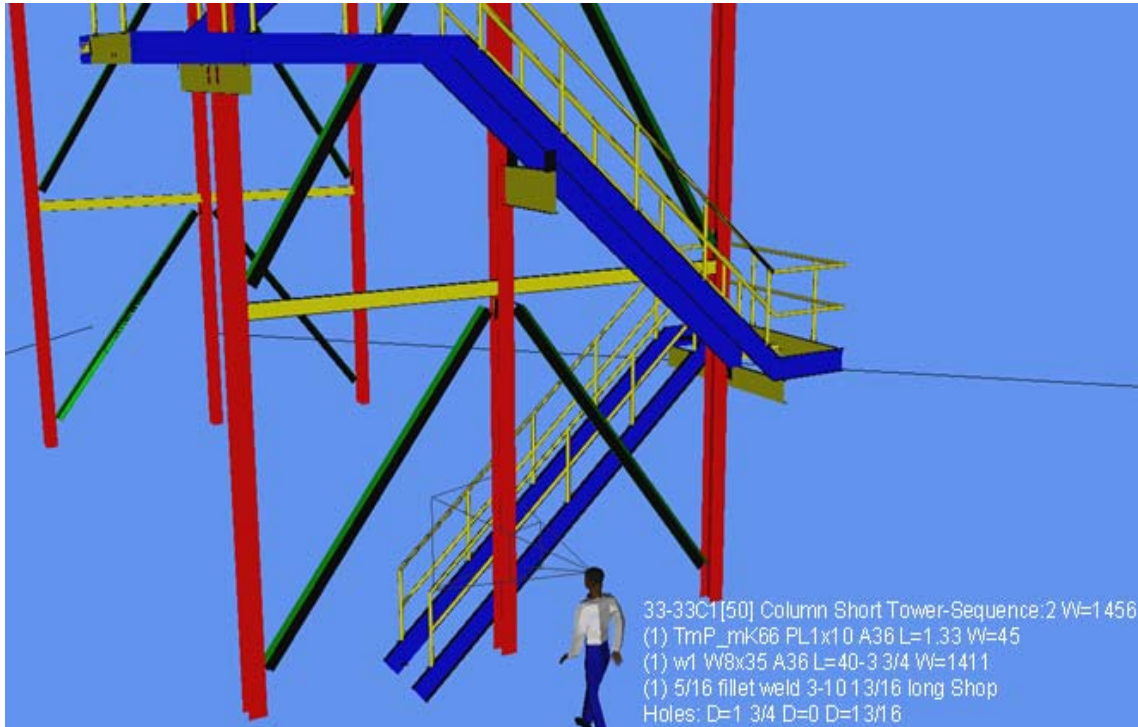


Figure 8.22 – Snapshot of Contextual Information Retrieval and Visualization of Identified Steel Elements

8.3.2. Maze Inspection Operation

A validation experiment was conducted at NIST, in the “maze” at the former NIKE missile base barracks building adjacent to the main campus. The main goal of this experiment was to evaluate whether contextual information on different objects in the maze (Figure 8.23) can be retrieved from project databases and presented to users based on their identified spatial context.

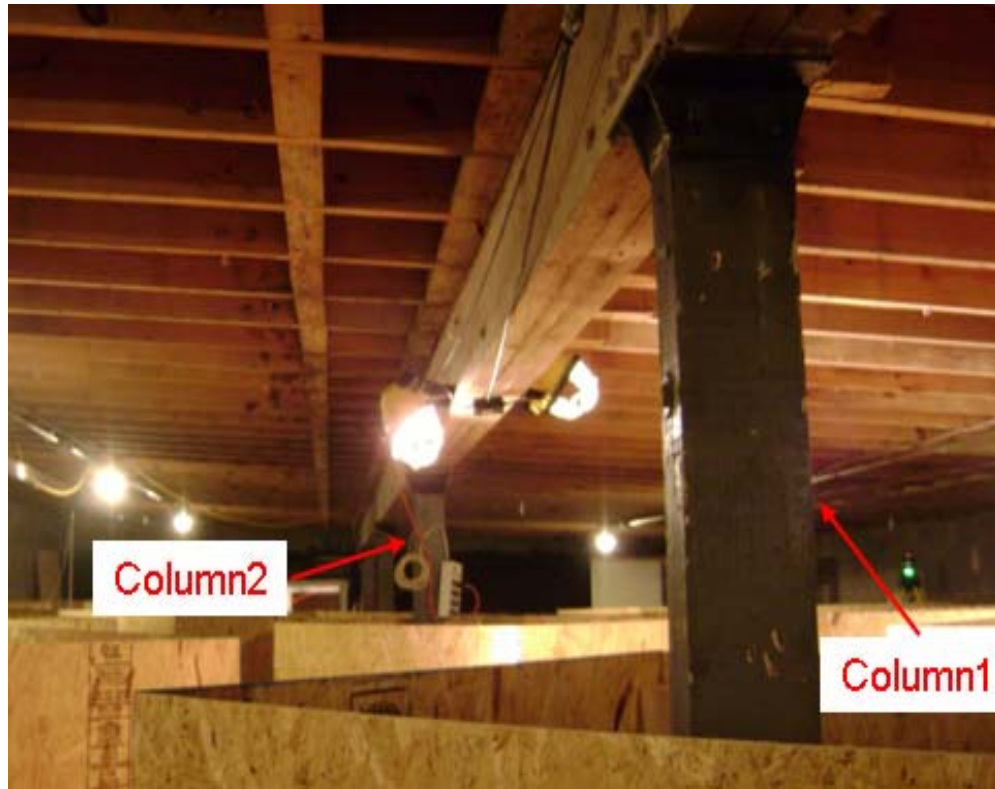


Figure 8.23 – Columns Location in the Maze (Nike Site, NIST)

The experiment simulated a mobile user moving in the maze, and inspecting its different parts. Having detected the interference between the current frustum (obtained based on tracked position and orientation data) and visible columns (Figure 8.24), contextual information was then automatically retrieved from the project database (Figure 8.25). In this case, the database is a MS Access database (Chapter 7) including all the details pertaining to the structural elements located inside the maze.



Figure 8.24 – Snapshots of Objects Interference Detection

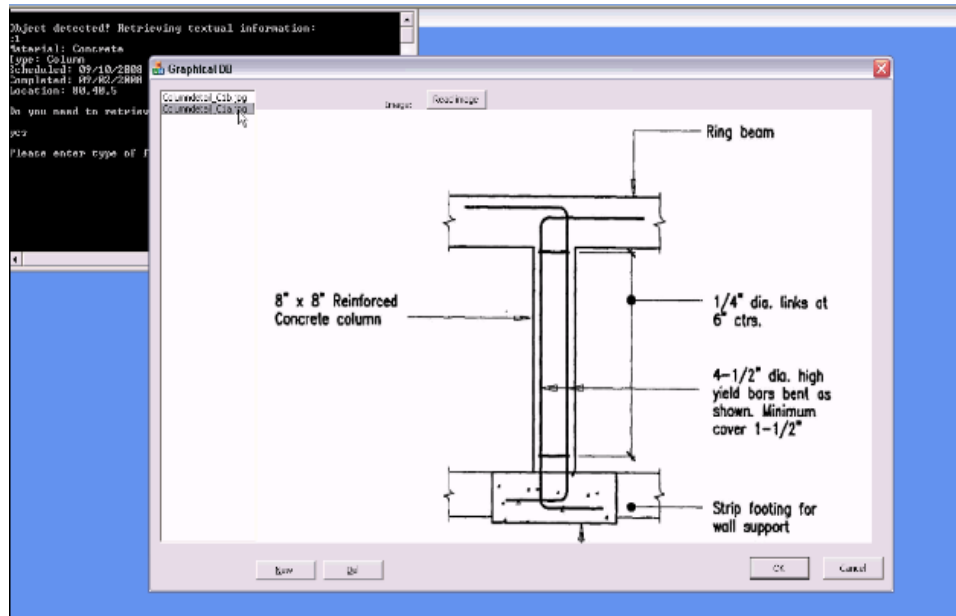


Figure 8.25 – Snapshot of Contextual Information Retrieval

8.4. Validation of Ability of the Framework to Track and Retrieve Contextual Project Information : Civil Engineering Laboratory Inspection Operation

In order to validate the overall ability of the proposed framework, an experiment, using Ekahau positioning system and magnetic orientation trackers, was conducted in the Structural Engineering Laboratory on the first floor of the GGB building at the University of Michigan (Figure 8.26). The WLAN indoor positioning testbed has an area of 40 ft by 25 ft and contains various structural elements including concrete walls and columns, steel columns and beams (Figure 8.27).

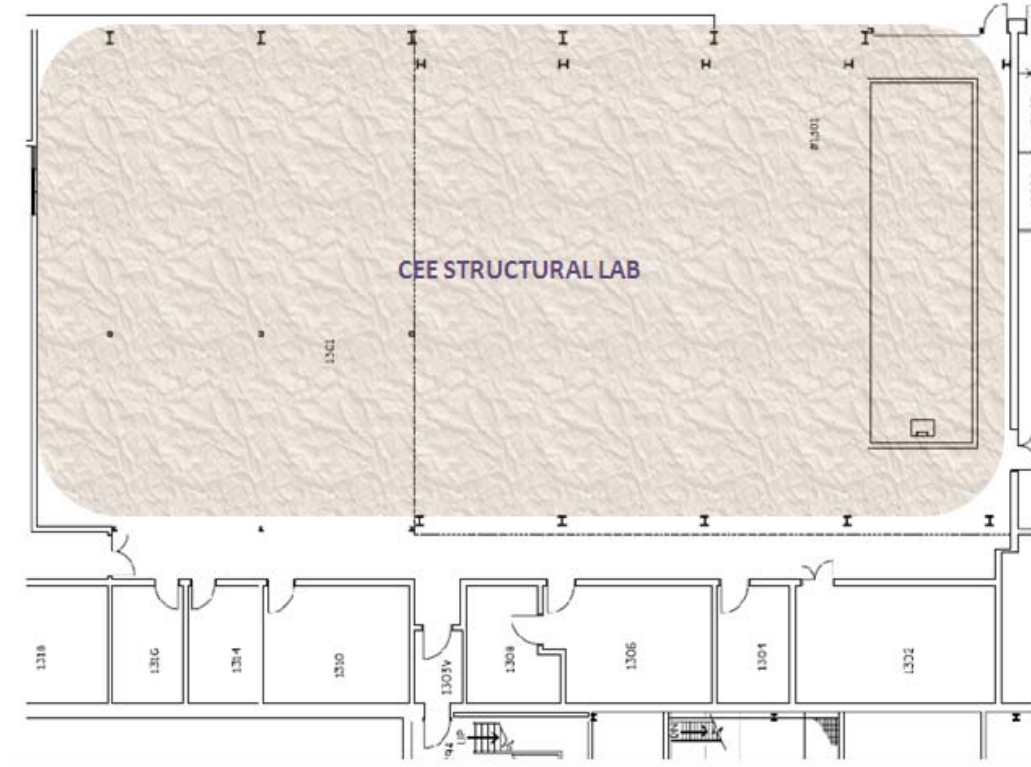


Figure 8.26 – Floor Plan of CEE Structural Engineering Laboratory

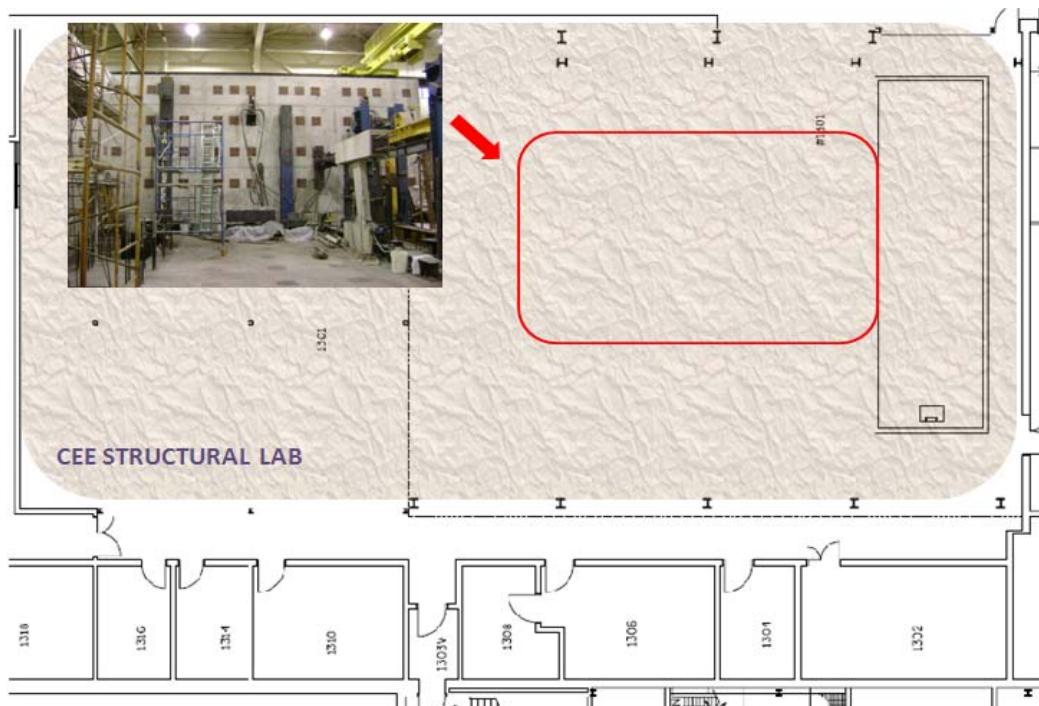


Figure 8.27 – Ekahau Testbed within the Laboratory

As mentioned in Chapter 5, area calibration and data collection follow the training phase of the fingerprinting approach. The signal strength measuring device (laptop or any other WLAN-enabled device) is set at a predetermined reference point (RP) position and data is collected from each access point in the area and saved to the database. The user then advances to the next RP and collects signal data again. In this experiment, several RPs were chosen. The dots shown in Figure 8.28 are different RPs located on tracking rails (Chapter 5). A fingerprint database (Ekahau Engine), comprising of all RPs was generated. Within the database, a fingerprint consists of measurements representing the received signal strengths obtained from each access point at that RP. As stated earlier in this chapter, university access points deployed in the different buildings across campus, as part of a wireless network, were used.

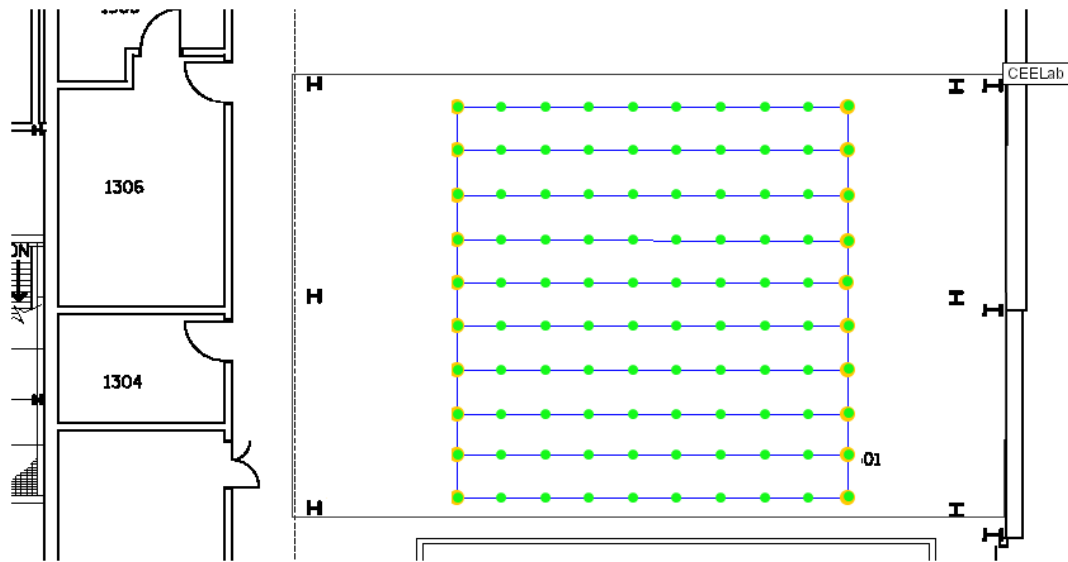


Figure 8.28 – Ekahau Calibration Reference Points

Having created the fingerprint database, signal strength data was collected at each mobile user's location while inspecting the different structural objects of the laboratory. The

method of data collection is similar to that of the RPs during the training phase. Data collected was then compared against the saved fingerprints and the user's position was determined. Based on this tracked position and orientation (obtained from a magnetic orientation tracker), as well as near and far plane distances (0.5 and 3 m respectively) and both the HFOV and VFOV angles (20 degrees), the eight coordinates of the truncated pyramid or viewing frustum were computed (Chapter 6). The frustum was then aligned with a 3D VRML model of objects in the laboratory, for example the concrete wall shown in Figure 8.29. In this case, the user was looking at a wall and then "locked" his context by pressing a key to temporarily stop the processing of tracking information. This "frozen spatial context" state allowed the user to move his head freely and look at the computer screen without having his interpreted context change continually.

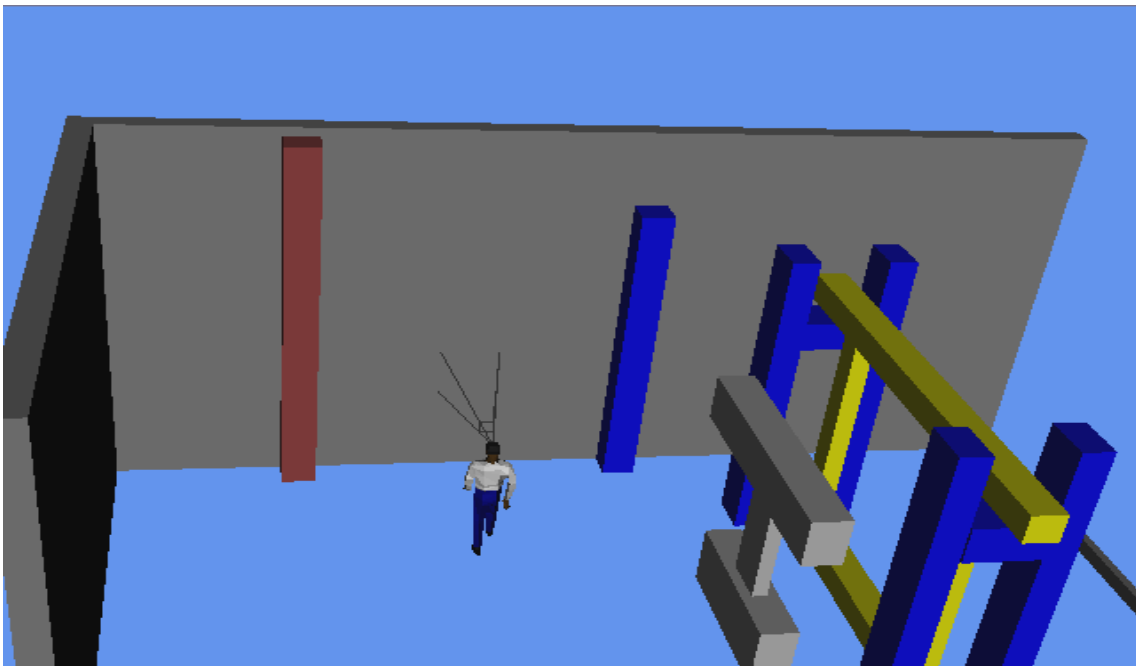


Figure 8.29 – Snapshot of Simulated Structural Engineering Laboratory Inspection

Within the determined and frozen context above, the next step included performing interference detection tests (Chapter 6). First, the intersection between the current frustum and objects was computed using general collision detection techniques, and then the raycasting technique was used to increase precision in object identification. For instance, a virtual ray was cast along the line of sight first (Figure 8.30 and Figure 8.31) to check for intersection and when it was reported, contextual information was retrieved (Figure 8.32).

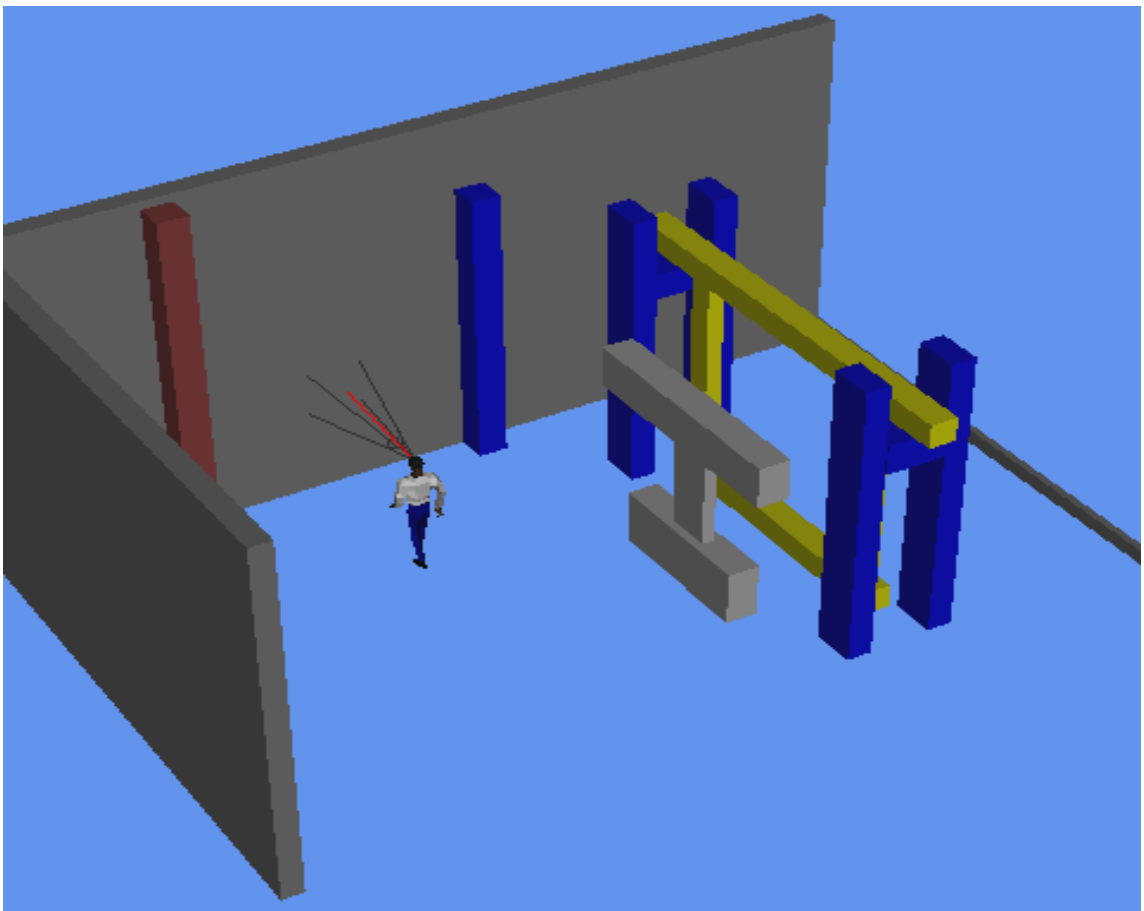


Figure 8.30 – Snapshot of Third Person View Interference Detection (Scenario 1)

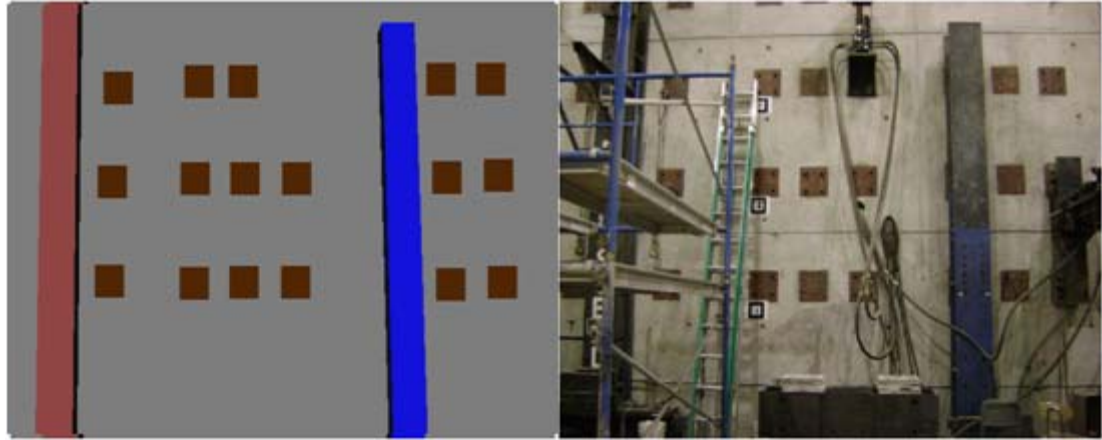


Figure 8.31 – Snapshots of Interference Detection (First Person Virtual and Real Views)

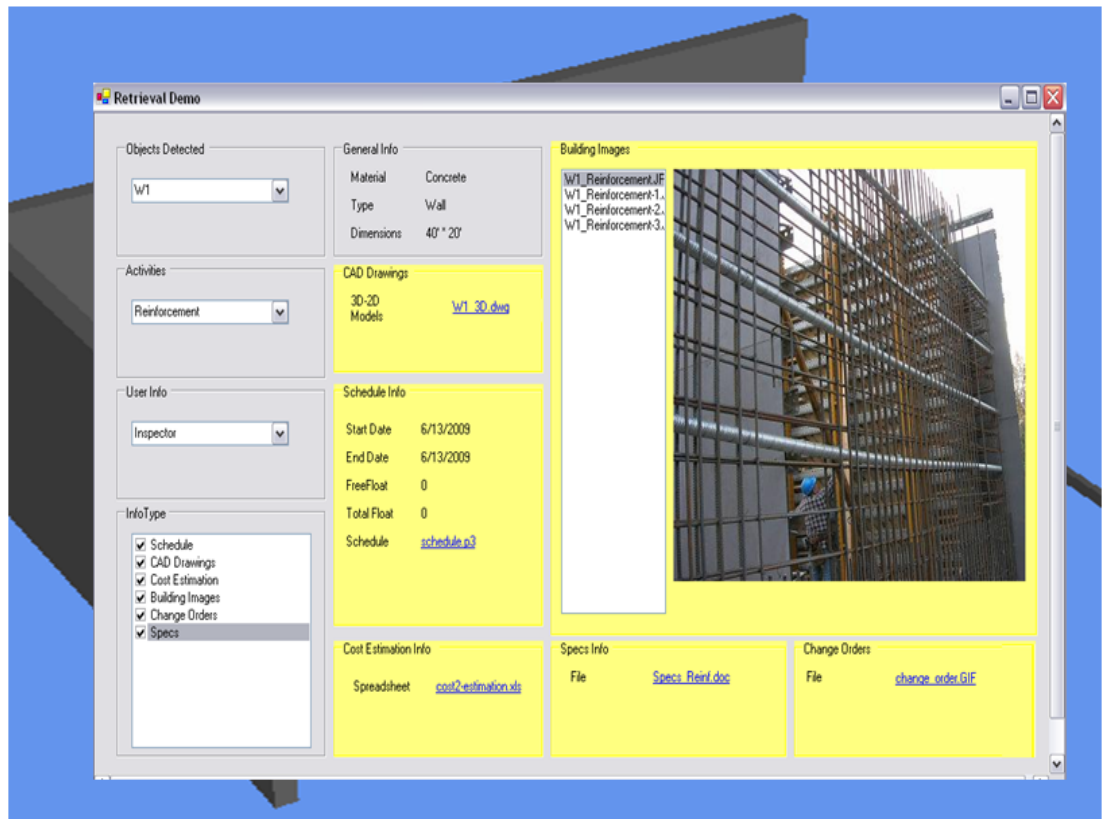


Figure 8.32 – Snapshot of Contextual Information Retrieval

As mentioned in Chapter 7, specific information on identified object(s) for a specific activity can be prioritized based on the user's function, and then retrieved. In the above retrieval user interface (Figure 8.32), the user, in this case an inspector opted to retrieve different types of information, i.e. construction schedule, building images, CAD drawings, etc. for the reinforcement placement and tying activity of the detected wall W1. Building images are automatically displayed on the form (Figure 8.32). Other type of information such as CAD drawings and cost estimation appear after the user clicks on the respective links (Figure 8.33 and Figure 8.34).

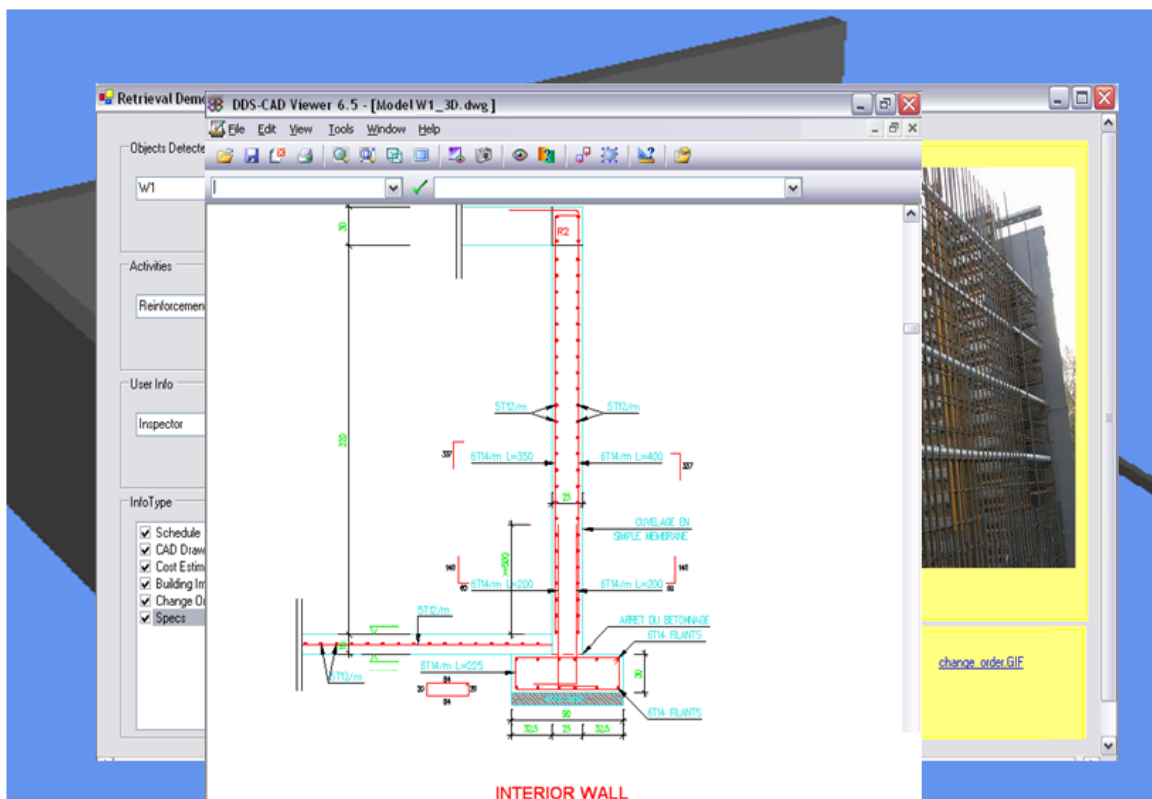


Figure 8.33 – Snapshot of Contextual Information Retrieval (CAD Drawing)

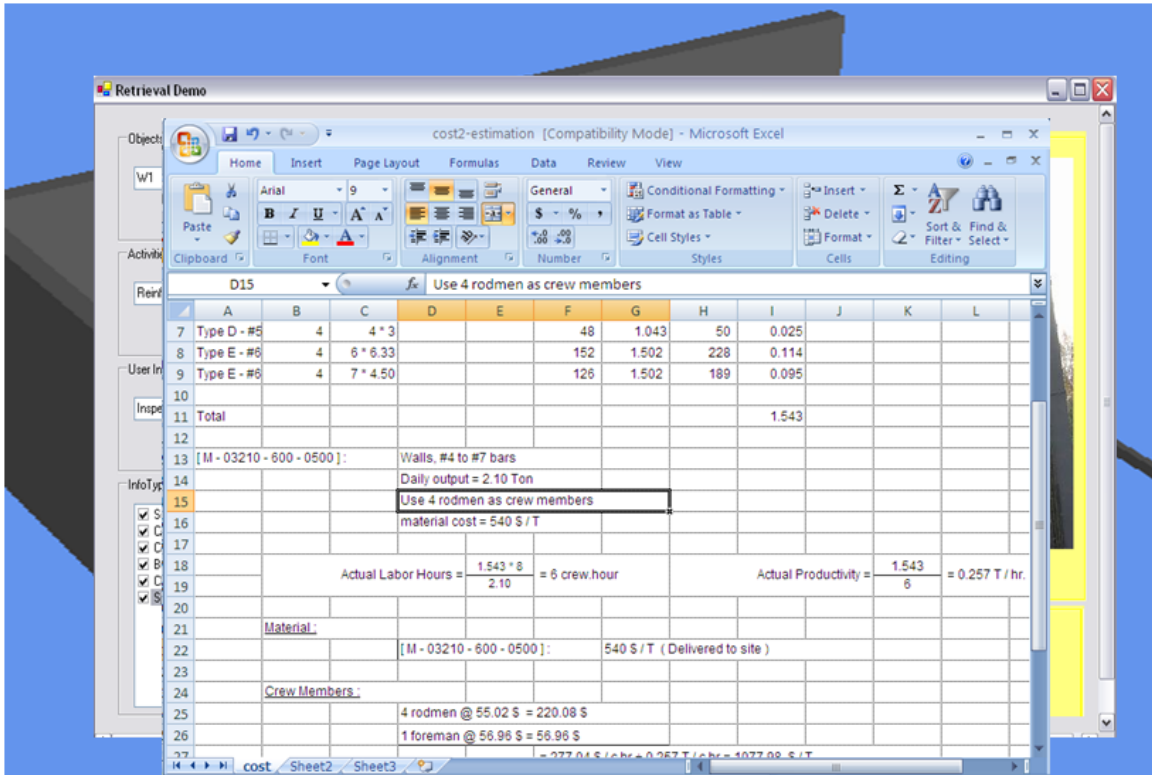


Figure 8.34 – Snapshot of Contextual Information Retrieval (Cost Estimation File)

Once information had been retrieved at this specific location, the mobile user pressed another key to unlock the context and resume the processing of tracking information. Figure 8.35 shows another location of the mobile user. Similar to all the previous steps, the context was locked, the viewing frustum was computed and then interference detection tests were performed.

In this case, two objects were identified along the line of sight, wall W1 and column C2. The user can then choose which object to retrieve information for from the drop-down menu named *Objects Detected* (Figure 8.32). However, C2, for example, at the time of inspection, might not have been built. Therefore the inspector can either decide to

retrieve information about the other detected object (wall W1) or check the schedule information for C2 and conclude whether there is any time delay.

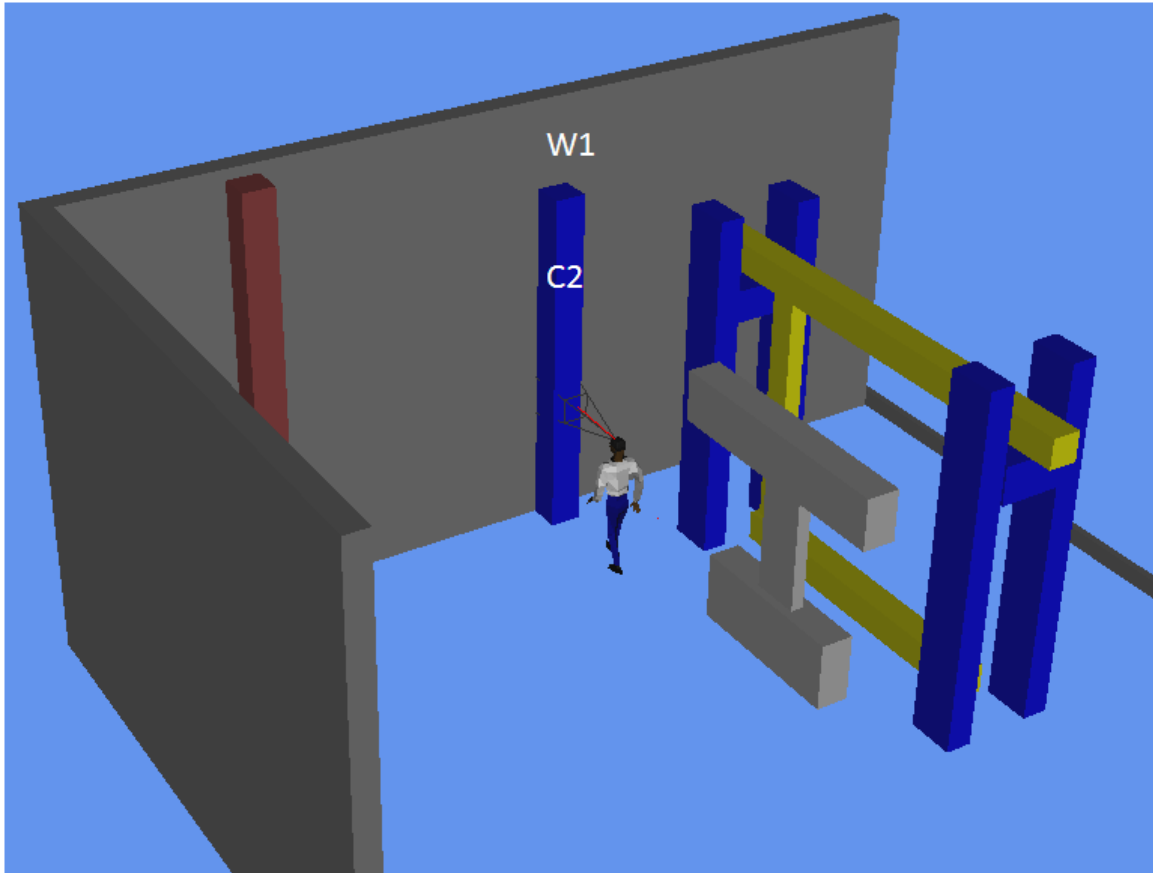


Figure 8.35 – Snapshot of Third Person View of Interference Detection (Scenario 2)

The user was continuously navigating in the CEE laboratory and inspecting the different structural elements according to the aforementioned inspection process. Figure 8.36 is a 3D representation, from a third and first person views, of the mobile user at different locations in the laboratory.

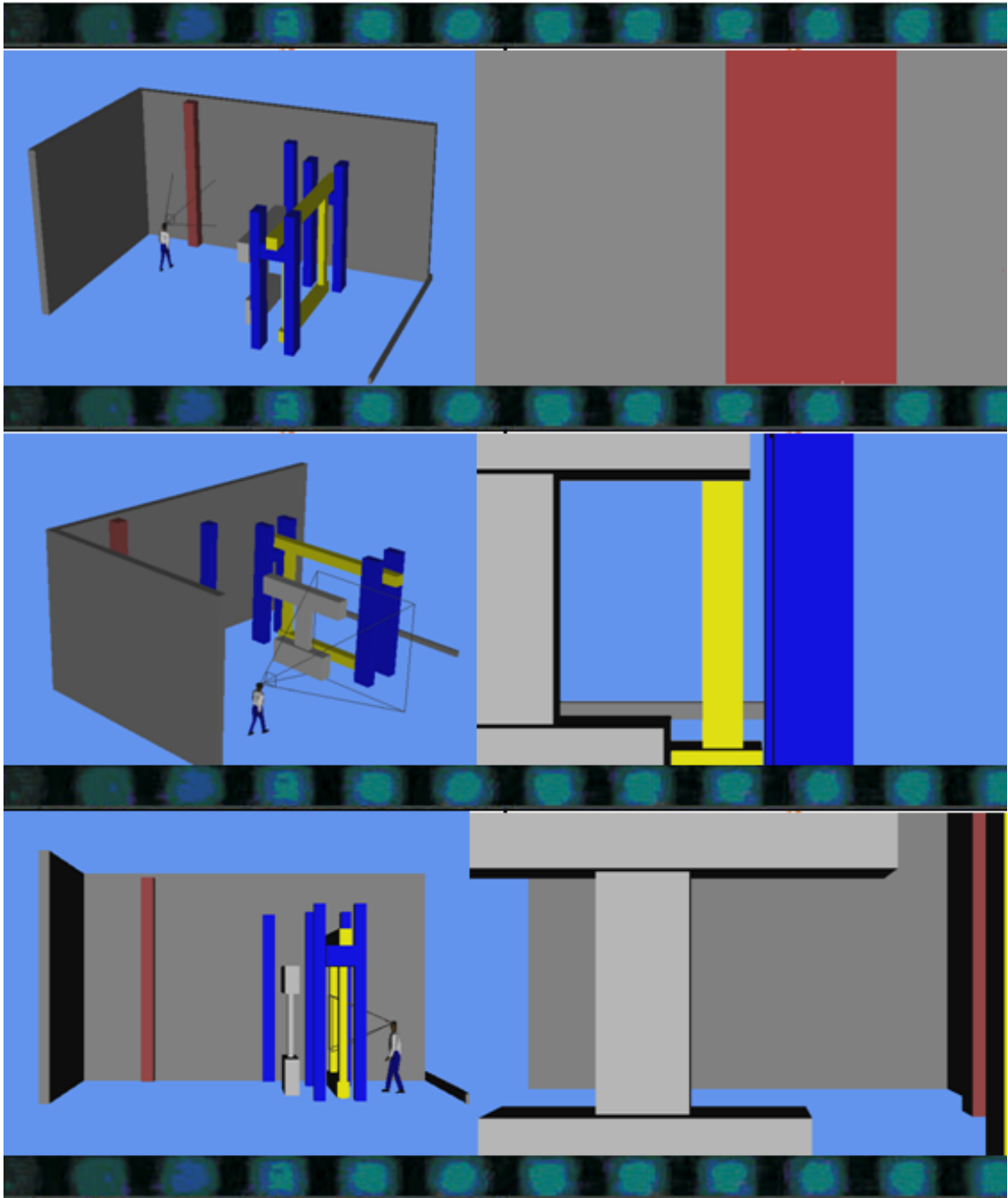


Figure 8.36 – 3D Snapshots of a Tracked User at Different Locations in the CEE Laboratory (Third and First Person Views)

8.5. Summary and Conclusions

Tracking mobile users accurately in harsh and dynamic indoor and outdoor environments such as those found on construction sites, precisely identifying visible project components, and retrieving contextual project information in real-time can help decision makers gain better insight and knowledge of operations and tasks while saving time and effort, and improving productivity.

In this chapter, the validation experiments were mainly grouped into two main categories including 1) accurate 3D user tracking and precise object identification, and 2) reliability of the context-aware framework to retrieve and visualize contextual information. Additionally, the overall ability of the context-aware information retrieval framework to continuously track users, identify objects in their field of view, and retrieve and visualize contextual information in real-time was validated through the set of experiments conducted at the University of Michigan.

However, it is important to note that factors such as accuracy, sensitivity, and practicability of each of the different framework components, in particular the contextual objects identification task (Chapter 6), were not part of this feasibility study, and need to be researched in the future.

8.6. References

Behzadan, A.H., and Kamat, V.R. (2006). “GPS and 3DOF Angular Tracking for Georeferenced Registration of Construction Graphics in Outdoor Augmented Reality”, 13th EG-ICE Workshop on Intelligent Computing in Engineering and Architecture, Ascona, Switzerland, 368-375.

Ekahau, Wi-Fi Based Real-time Tracking and Site Survey Solutions – available at: <http://www.ekahau.com>. (Accessed: February 20, 2007).

Kang, S., and Tesar, D. (2004). “Indoor GPS Metrology System with 3D Probe for Precision Applications”, In Proceedings of the ASME International Mechanical Engineering Congress (IMECE), Anaheim, CA.

Metris Products Website, IGPS Large Scale Metrology - available at: http://www.metris.com/largevolume_tracking_positioning/igps/ (Accessed July 24, 2007)

Multispectral Solutions, Inc. Website, Sapphire DART System- available at: <http://www.multispectral.com/products/sapphire.htm> (Accessed May 31, 2007)

OpenSceneGraph Website, Introduction to OpenSceneGraph – available at: <http://www.openscenegraph.org/projects/osg/wiki/About/Introduction> (Accessed November 18, 2005).

Texas Engineering Extension Service (TEEX) Website, Disaster City- available at: <http://www.teex.com/teex.cfm?pageid=USARprog&area=USAR&templateid=1117> (Accessed: June 4, 2007).

Chapter 9

Summary and Conclusions

9.1. Summary of the Proposed Methodology

This dissertation documented the research that led to the design and implementation of a location-aware information retrieval methodology to improve important decision-making tasks on construction sites and other engineering applications by providing support for tedious and time-consuming activities associated with timely and accurate access to project information.

In an information-intensive industry such as construction, rapid and convenient access to project information is crucial in helping engineers, inspectors, and maintenance personnel make critical, real-time decisions. Field personnel typically spend a significant amount of time in manually accessing the information they need for important decision making

tasks. Such lost time amounts to lost productivity and thus money. Conceptually, relevant information can be delivered to any decision maker in real-time by monitoring their physical and environmental context, and then reconciling the identified context parameters with the available pool of digital information.

As noted in Chapter 1, 2 3, and 4, previous studies have focused on enabling and advancing information technology by providing a mechanism to deliver pertinent information in order to enhance construction collaboration and work efficiency. All prior context-aware information delivery research and applications, however, have focused on developing tracking applications where the spatial context was defined solely by the location (position) of the user. Another major attribute, the 3D head orientation, was ignored in the computations. A user's three-dimensional head orientation fully describes the user's line of sight (i.e. the direction in which the user is looking) and therefore can define a user's spatial context with much greater precision than is possible with position alone.

For these reasons, prior context-aware information retrieval applications do not consider identification of specific objects visible to a mobile user within the realm of a room or section of a building. Instead, they detect and identify other generic information, for example the overall location of the user in the building (e.g. floor). Additionally, prior research does not incorporate algorithms and methods to automatically retrieve and visualize contextual information on identified visible components in the studied area.

In this dissertation, the above limitations were successfully addressed. The design and development of ubiquitous position and orientation tracking methods, computer graphics based techniques, data retrieval and visualization tools together with their practical implementation and integration inside a 3D mobile tracking and visualization tool were addressed. The results of the research have been validated by tracking mobile users in many outdoor and indoor environments and retrieving and presenting contextual information in real-time.

As mentioned earlier, the proposed methodology is independent of any specific technology. With the advance of time, better and more accurate tools can be used and integrated.

9.2. Contributions

The main contribution of this research to the body of knowledge lies in designing, implementing, and demonstrating the ability of accurately and ubiquitously tracking mobile users' fully qualified spatial context, and precisely retrieving, delivering and presenting information in real-time in order to assist decision making tasks in dynamic environments such as those found on construction sites. This was achieved by developing a dynamic user-viewpoint tracking scheme in which mobile users' fully-qualified spatial context is defined. Additionally, methods were developed to identify objects and artifacts visible in a mobile user's field of view with much higher accuracy than was possible by tracking position alone. Methods were also implemented to provide continuous, location-

based contextual information on specific identified components and interactively visualize retrieved information in an arbitrary indoor/outdoor environment.

Therefore, this dissertation described the details of individual scientific questions addressed, and several methods studied, developed, and implemented in contributing to different research objectives. The following list describes the individual research challenges summarized in Chapter 2 and addressed in the following chapters:

- Accurate position and orientation tracking in outdoor and indoor environments (Chapter 5): Evaluating, designing and implementing methods to accurately track mobile users in congested and dynamic environments using the Global Positioning System (GPS), Wireless Local Area Networks (WLAN), Ultra-Wide Band (UWB), and Indoor GPS positioning devices as well as Three Degree-of-Freedom (3DOF) orientation tracking devices (Chapter 3).
- Continuous computation of region of space visible to mobile users (Chapter 6): Developing and implementing methods to compute a mobile user's viewing frustum based primarily on the tracked spatial parameters.
- High-precision identification of contextual objects (Chapter 6): Devising methods to precisely interpret a mobile user's fully-qualified spatial context and identify visible objects using geometric interference analysis techniques.
- Contextual information retrieval and visualization (Chapter 7): Designing methods to retrieve prioritized project information from project databases and product models (Chapter 4) and interactively present it to the user.

In Chapter 2, initial steps toward a ubiquitous location-aware information retrieval methodology designed for information support in construction and other engineering applications were introduced. *The primary contribution of the research presented in this chapter is to describe several existing context-aware applications and then go on describing the overall proposed location-aware information retrieval framework that enables real-time tracking of mobile users' full-qualified spatial context, identification of objects in users' field of view, and automated retrieval and visualization of contextual information in dynamic outdoor and indoor environments such as those found on construction sites.*

In Chapter 3, different positioning and orientation tracking technologies to continuously track mobile users in outdoor and indoor environments were evaluated. For outdoor environments, GPS was used to continuously track a user's position. However, GPS is unsuitable in situations where the user has a possibility of being indoors because the receivers need a clear line-of-sight to the satellites in order to track position. Therefore, *the primary contribution of the research presented in this chapter is to evaluate, together with orientation sensing tools, the different wireless positioning technologies in order to achieve accurate user tracking for information support in construction and other engineering applications. Eye motion tracking technologies were introduced as well but currently deemed not effective for the purpose of the proposed methodology.*

In Chapter 4, standard product models, project databases and visualization tools to retrieve, deliver, and interactively present information to mobile users in engineering

applications were described. Evolving technologies such as interoperable product models, project databases and interactive display devices and tools offer significant potential of improving traditional, time-consuming, manual information retrieval processes, and supporting important decision-making tasks in the field. Therefore, *the primary contribution of the research presented in this chapter is to evaluate the different project databases, product models and visualization tools in order to rapidly and conveniently provide mobile users with context-aware information to support on-site decision-making tasks.*

In Chapter 5, the details of tracking algorithms necessary to track position and 3D orientation (i.e. yaw, pitch, and roll) of the user's viewpoint in indoor environments were discussed. The overall tracking scheme designed encompasses both outdoor positioning technologies and indoor wireless tracking techniques. For outdoor environments, a georeferencing based algorithm previously developed using GPS and magnetic orientation tracking devices was used to continuously track a user's dynamic viewpoint. However, GPS is unsuitable in situations where the user has a possibility of being indoors (Chapter 3) because the receivers need a clear line-of-sight to the satellites in order to track position.. Therefore, in this chapter, tracking algorithms based on different wireless positioning technologies, namely WLAN-based Ekahau, UWB and Indoor GPS systems as well as magnetic orientation tools, were designed and implemented for adoption in indoor location-aware information delivery applications. The results of experiments conducted (Chapter 8) highlighted that based on the circumstances expected to be encountered in the intended deployment environment (i.e. indoor construction

sites), the Indoor GPS positioning technology was found to offer the most promise of accurately tracking mobile users' position due to the low level of uncertainty in the reported user position (1 to 2 cm) compared to that of WLAN (1.5 to 2 m) and UWB (9 to 50 cm). *The primary contribution of the research presented in this chapter is to present the different algorithms designed and implemented to track mobile users in construction and other engineering applications.*

In Chapter 6, methods were developed to calculate the region of space visible to a mobile user based on the tracked location and head orientation (Chapter 5) and then prioritize the context of visible objects in the user's view using geometric interference analysis techniques such as a raycasting approach. *The primary contribution of the research presented in this chapter is thereby to describe, using computer graphics techniques, the methods designed to interpret the mobile user's spatial context and accurately identify relevant visible objects .*

In Chapter 7, two types of project information systems, namely CIS/2 product models and MS Access databases were implemented to retrieve contextual information about identified objects in users' field of view. Related experiments (Chapter 8) were conducted at the National Institute of Standards and Technology (NIST) indoors, in the maze at NIKE site, outdoors at the steel structure on the main campus, and in the Structural Engineering Laboratory on the first floor of the GGB building at the University of Michigan. Results highlighted the potential of the aforementioned information repositories in saving time and improving work productivity on construction sites in

particular. *The primary contribution of the research presented in this chapter is to describe using specific project databases and product models, the different methods designed to rapidly and conveniently provide mobile users with context-aware information to support on-site decision-making tasks.*

In Chapter 8, validation exercises evaluating each of the individual research components described in Chapters 5-7 of this dissertation were presented. In addition to validating each individual aspect of the research during its progress, the overall ability of the location-aware information retrieval methodology to ubiquitously provide mobile users with relevant contextual information was validated by conducting an indoor experiment in the Structural Engineering Laboratory at the University of Michigan. The designed methodology and the implemented software and hardware framework were critically evaluated from the point of view of accuracy in 3D spatial user tracking in congested environments, precision in visible components identification and detection, and ability of the framework to retrieve and visualize contextual project information.

9.3. Future work

In this research, the author has developed a spatial context-aware methodology that focuses on automatically identifying and retrieving contextual project information for on-site decision-making in field applications such as construction. As described in earlier chapters, GPS based positioning can accurately locate a user outdoors, provided a clear view of the sky is available whereas wireless based positioning can locate a user indoors,

provided a network of wireless base stations (access points, receivers, transmitters, etc.) already installed at fixed locations. Since the orientation tracker used in this research is magnetic-based, it can work both outdoors as well as indoors.

However, both positioning options are not feasible on their own because construction engineers and other field personnel typically operate in a mixed outdoor and indoor environment, i.e. no assumptions can be made about when they will be outdoors or indoors. Therefore, all components of this proposed framework must be self-contained and not rely on pre-installed networks of sensors, especially inside the same buildings that an engineer might be entering into.

In order to address the current drawbacks, a context-sensing framework that will accurately track a mobile user's 3D spatial context in an arbitrary indoor/outdoor environment, using a novel combination of GPS and wireless positioning technologies, will be developed. A network of base stations will be rapidly deployed at outdoor locations having a clear view of the sky to allow for GPS connectivity. Each such base station will be mounted with a long-range WLAN router to create an ad-hoc wireless local network. The exact global position of each router is automatically known from the GPS signals. Mobile users in the vicinity will be positioned relative to the WLAN routers mounted on the base stations. The users' global positions can then be computed by reconciling their local positions relative to the routers with the global GPS positions of the routers themselves. This scheme will thus provide uninterrupted global position tracking for any user regardless of their indoor or outdoor location. In addition, this

methodology has other distinct advantages; it is rapidly deployable and reconfigurable. The GPS-tracked WLAN base stations can be set up or moved at any given time to service users in different local areas.

Another possible positioning application that can be used is part of an ongoing project at the University of Michigan. It is based on the combination of GPS and Personal Dead Reckoning (PDR) systems. The concept behind it is that the position of the mobile user is given by the GPS whenever available. When the GPS is unavailable, the position will be given by the PDR. Then the position will be switched back to the GPS position when GPS is reconnected.

The PDR used is the Personal Odometry System (POS) developed by Professor Borenstein and his research group at the University of Michigan. It transmits the location of a mobile user relative to a known starting position. It works by using an Inertial Measuring Unit (IMU) mounted to the user's boot (Figure 9.1) and connected to a portable laptop computer.



Figure 9.1 – Inertial Measuring Unit (IMU) mounted to the user's boot

PDR algorithms correct the drift of the accelerometers in the IMU with every step, thereby preventing the accumulation of errors. Most importantly, this PDR system requires no sensors that must be preinstalled in the work environment as is the case of the developed framework in this research.

Another future research plan includes integrating Augmented Reality (AR) within the current framework. AR is a technique in which computer generated data (i.e. graphics, text, diagrams, etc.) are embedded into the views of the real environment (Figure 9.2).



Figure 9.2 – Augmented Reality View

Automated retrieval of information within an AR environment can serve as an alerting mechanism that can compare the as-built and as-designed information and notify the site

personnel of any significant deviations, like activities behind schedule and materials not meeting the specifications. The as-built automated retrieved information could also assist further by applications that can real-time suggest reconfigurations of priorities to the user if such deviations are detected.

Many application areas can benefit from the suggested hybrid position tracking framework together with magnetic orientation sensing tools and have the contextual information automatically retrieved and visualized within AR environments.

As was described in Chapter 1 and 2, steel inspection is one application area where inspectors, based on their tracked spatial parameters, can automatically retrieve detailed drawings contained in the product models or project databases to check and verify whether steel members are correctly installed at the site. However, in the previous inspection operation, the inspector was only moving outdoors. By using the ubiquitous context-sensing framework to track the inspector and comparing the as-built and contextual as-designed information within an AR environment, site personnel, whether moving outdoor or indoors, can be easily notified of any significant deviations. The above shows clearly that information retrieval within an AR context is very useful in progress monitoring and productivity measuring and can save valuable inspection time. By doing so, more frequent inspections are being promoted and the amount of time spent to detect problems in the construction site such as lack of materials, construction mistakes, etc., lessens which increases response time.

Another application area includes a building inspection scenario in the aftermath of a disaster (e.g. earthquake) (Figure 9.3). An inspector surveys a building for signs of visible damage. In this example, the inspector's position and line-of-sight are tracked using GPS and magnetic orientation trackers. The inspector, however, surveying a damaged building cannot afford to lose GPS connectivity simply because other structures block the view of the satellites at certain locations around the building. In this case when GPS becomes unavailable, the position will be given by the PDR for example.

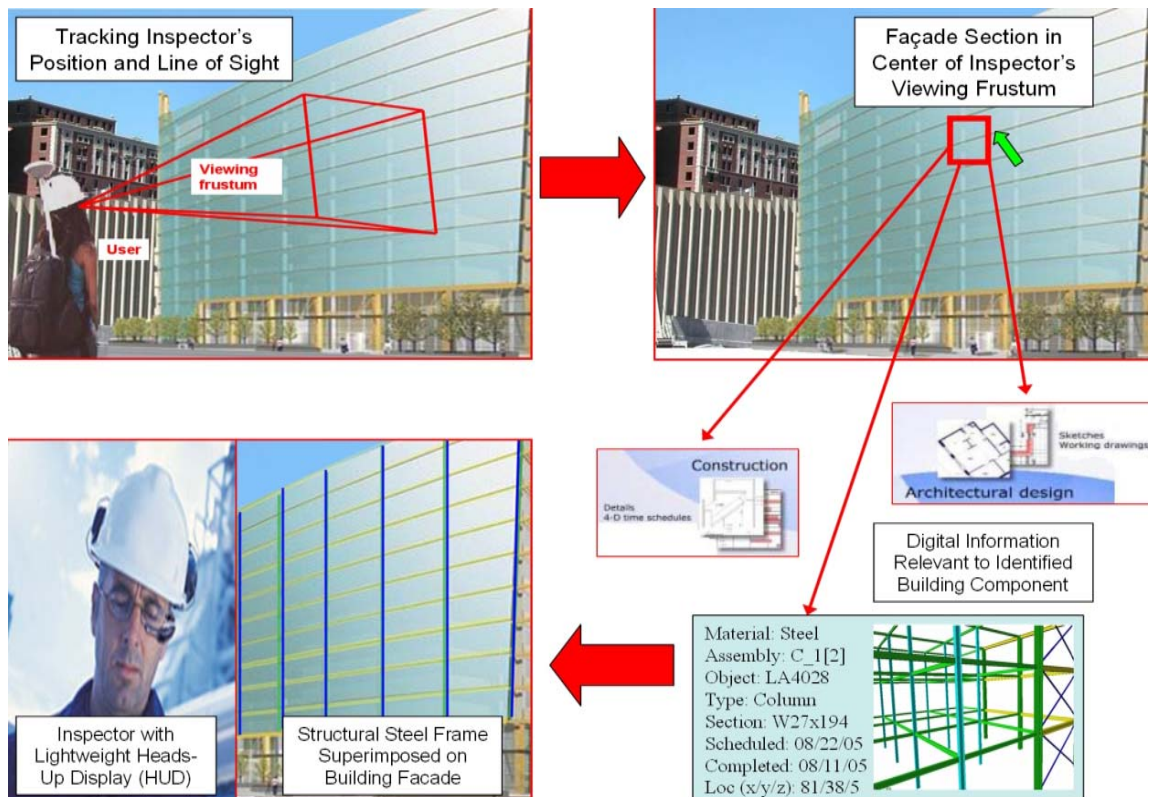


Figure 9.3 – Building Damage Inspection

During the survey, based on the interpreted spatial context, relevant information about the building's structural frame, construction details, and other information can be retrieved

are presented. For example, as depicted in the lower left picture in Figure 9.3, using AR, the building's hidden structural frame can be superimposed on its façade to allow the study of critical connection locations that need detailed inspection. As the inspector approaches these connections, structural details on those connections can be automatically retrieved and displayed based on the interpreted and refined spatial context.

A third possible inspection application is bridge inspection (Figure 9.4).

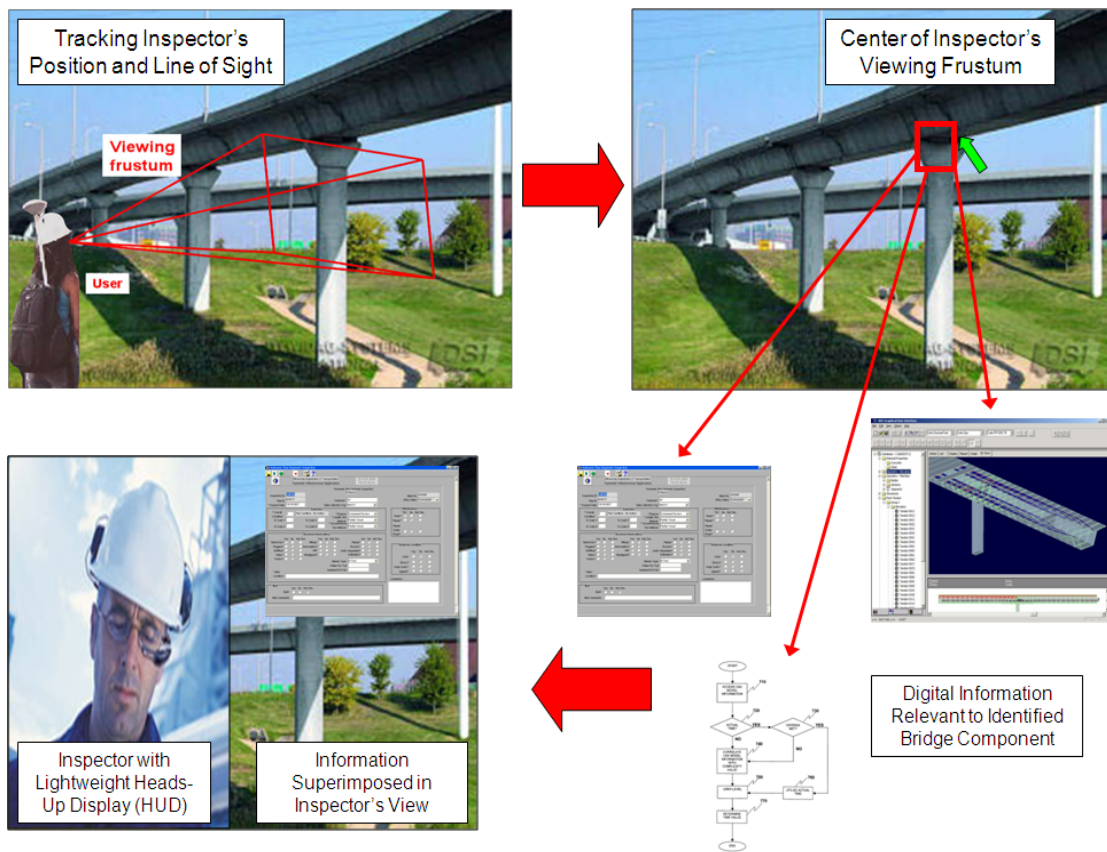


Figure 9.4 – Bridge Inspection

Bridge inspectors regularly revisit the same bridges (on a yearly basis) for damage history documentation. Given the nature of their work and the frequency of their visits, automatic retrieval of relevant contextual information residing in the database from past years and its interaction within an AR context can be found very valuable in terms of assessing the damage. The latter includes, detecting the differences between the current real information obtained based on the inspector location on the site and past information pertaining to the same part of bridge but contained in the product model of the structure. This procedure will contribute in avoiding the tedious manual task that bridge inspectors have to perform each year as well as reduce the time needed to do so.

This application is part of the ongoing NIST Technology Innovation Program (TIP) projects recently announced to develop advanced sensing technologies that would enable timely and detailed monitoring and inspection of the structural health of bridges as well as roadways and water systems.

The ubiquitous tracking and information retrieval algorithms and methods can be readily applied to other engineering domains such as inventory control and emergency response. Inventory control is another possible application area (Figure 9.5) taking advantage of the hybrid GPS-PDR tracking framework as well as AR.



Figure 9.5 – Inventory Control Scenario

In this case, by switching to PDR indoors, people on construction sites can get a complete inventory as well as the location of assets which will result in cost savings. Moreover, individuals will waste much less time looking for lost things, and each other in crowded places. In the process of performing this task, the user's performance in terms of speed, accuracy, and reliability may be enhanced through the use of timely and appropriate additional information. However, interrupting the process to refer to printed manuals or plans can be distracting and confusing. Having all manuals, documents and plans in a database, say a product model, and being able to retrieve real-time this information and interacting with it in an AR environment is deemed to enhance the performance of the user on this indoor task.

In the emergency response scenario, a firefighter simply cannot rely on wireless base stations (routers, receivers, etc.) already installed inside a building because those stations themselves might have get destroyed in the disaster being responded to. Therefore, an alternative solution includes using GPS-tracked WLAN base stations set up in the vicinity of the experimental building (Figure 9.6).

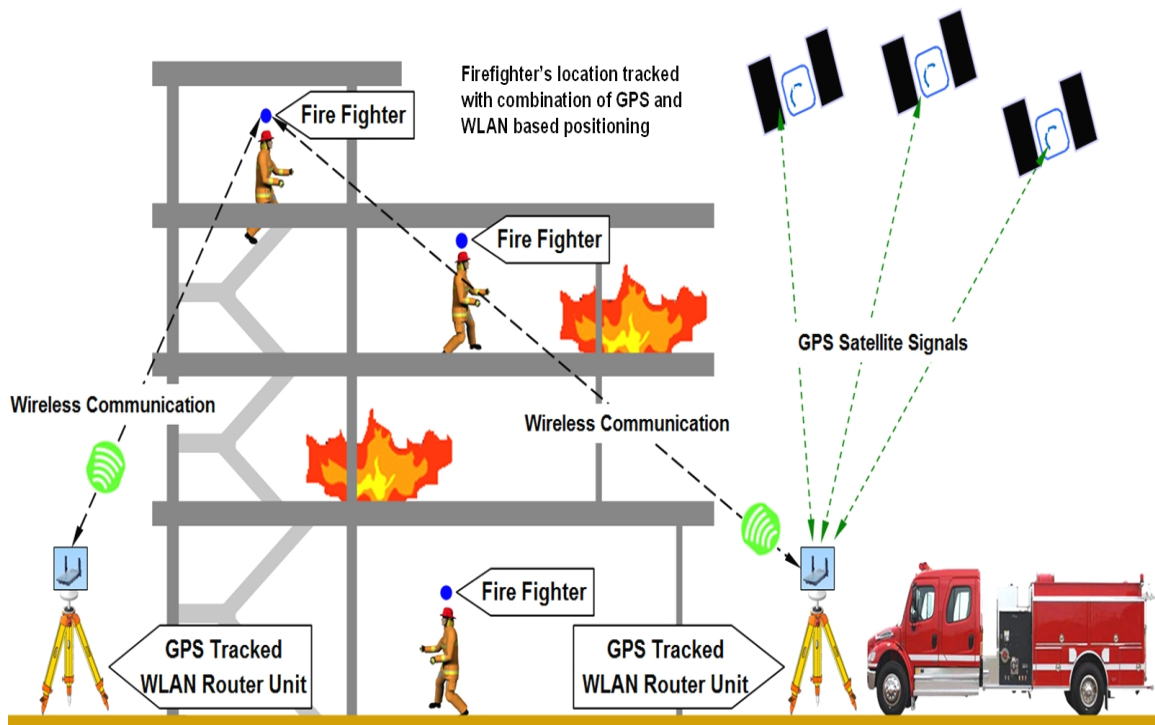


Figure 9.6 – Hybrid GPS and WLAN-Based Global Position Tracking in Emergency Response

The firefighter(s) entering the building will wear lightweight helmet-mounted displays connected to wearable computers that will communicate wirelessly to the established base WLAN base stations. The helmets will also be fitted with magnetic orientation trackers to track the orientation of the line of sight. Based on the tracked spatial context, the computer will superimpose onto the firefighter view of the real environment the

contextual navigation and building information such as floor plans, exit locations, location of critical assets as well as providing the firefighter's current position and orientation in the building (Figure 9.7). This will be achieved by representing the firefighter as an avatar using a world-in-miniature paradigm.



Figure 9.7 – Navigation Guidance in First Response

Another solution to help firefighters in the effective planning of response and rescue strategies lies on using the GPS-PDR combination.

As noted in Chapter 3, eye motion tracking systems (Figure 9.8) were not considered within the scope of this research for certain issues. However, future research plans can

include designing and implementing methods to identify objects by taking into account not only the user's head orientation but also the user's eye movement.



Figure 9.8 – Eye Motion Tracking Tool

This is commonly accomplished through computer vision techniques imaging pupil and corneal reflection of infra-red light (Figure 9.9).

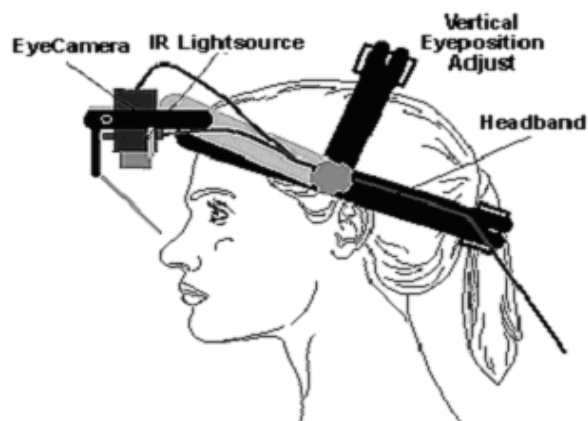
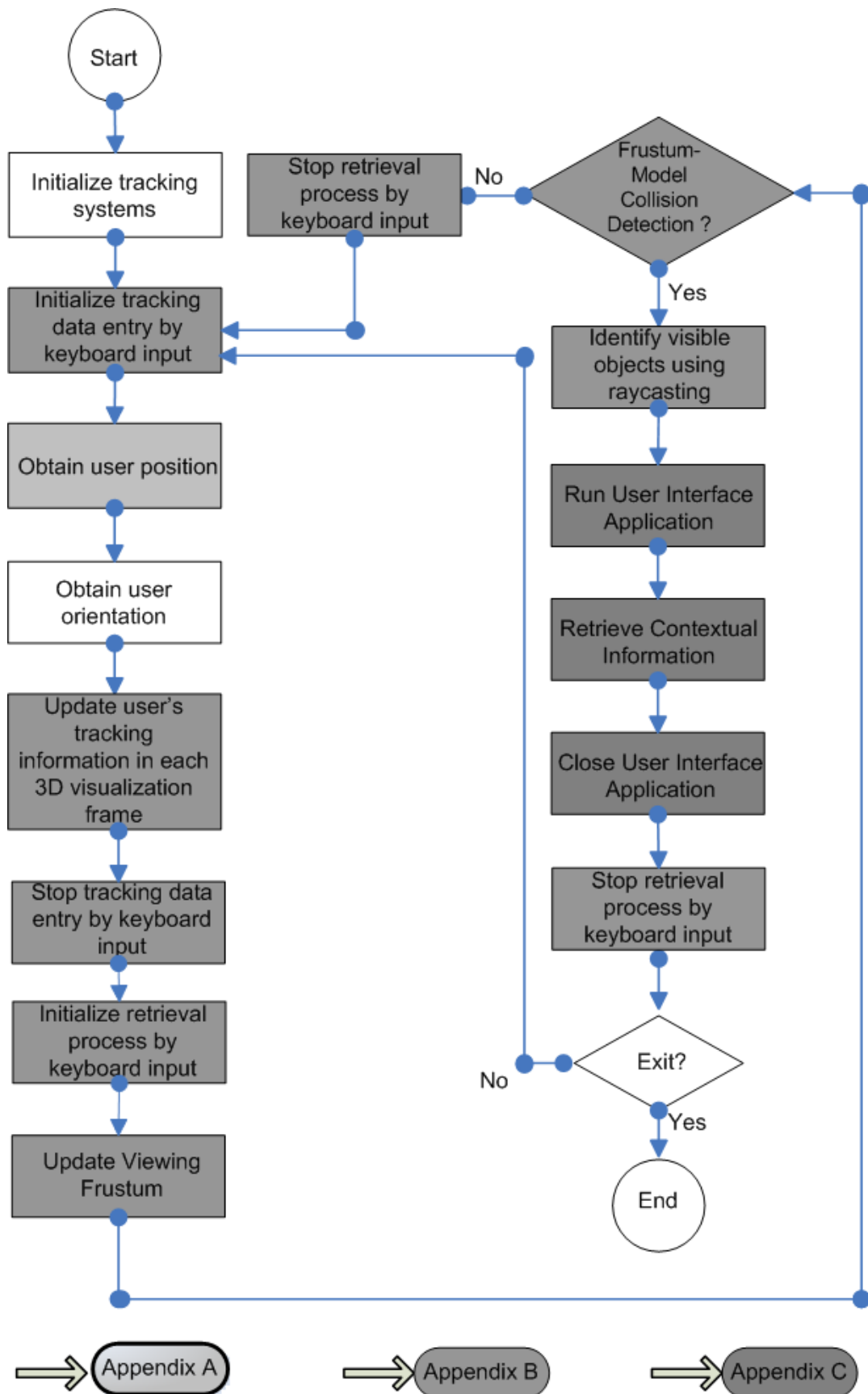


Figure 9.9 – IR-Based Eye Motion Tracking System

Appendices

The following Appendices are provided to supplement the discussion of topics introduced in previous Chapters. Appendix A presents a guide to use WLAN-based Ekahau, UWB, and Indoor GPS positioning technologies. Appendix B describes all the main computer graphics concepts and techniques adopted in this research to define the viewing frustum, and identify and prioritize contextual objects. Appendix C explains in detail all the steps involved in accessing and retrieving information from project databases, using MS Access as an example information repository.

The flowchart, shown below, describes the whole process of tracking mobile users in real-time, and identifying and retrieving contextual information. The components of the flowchart fall into three categories, each corresponding to a different appendix.



Appendix A

Guide to Use WLAN-Based Ekahau, UWB, and Indoor GPS Technologies

In this Appendix, the detailed steps to setup and use wireless positioning technologies to extract position information are documented. As described in Chapters 5, three location-tracking wireless technologies, namely, WLAN, UWB and Indoor GPS were adopted.

The presented information is expected to serve as a guide to help readers gain a better understanding of the whole position tracking process using the aforementioned technologies.

WLAN-based Ekahau System Guidelines

Ekahau RTLS (Real-Time Location System) is a continuous monitoring platform that allows system users to know where their wireless assets/devices are currently located.

The platform is software based; with the central Positioning Engine running off Windows or Linux servers, and software clients available for the tracking of laptops and/or handheld devices (2000/XP/2003, Pocket PC, Windows Mobile, Symbol). The platform operates on standard Wi-Fi networks without requiring proprietary antennas or other hardware such as base stations. It does not need to know the location of the base stations and it works with 802.11a, b, g, and now n gear. As described in Chapter 5, individual components of the platform include the central Ekahau Positioning Engine (EPE), which utilizes strength of signal indicators from user-selected contributing access points in the Wi-Fi network in order to determine an object's or user's location, and the Ekahau Manager, for creating the positioning model and updating it and saving it to the engine. Other components include optional and specific applications that can leverage the platform's capabilities to provide location-based services. Finally, a Java-based SDK and socket-based API are provided, allowing developers to integrate positioning capabilities within remote location applications. In this research work, the Java-based SDK was used to extract positioning information.

E

Before any experiment can be carried out, there are some guidelines that must be explained so that readers have a better understanding of how the actual testing (Chapter 8) was done. In the subsequent sections, the Ekahau technology used and the setup of its components together with the test routines are explained.



To setup the Ekahau tracking system, the user has to:

- . Install the Positioning Engine, Ekahau Manager and Ekahau Client on a laptop PC with at least Windows®2000, Windows® XP Professional, 1 GHz processor, 1 GB RAM, and 200 MB HD space
 1. Double-click Install_Engine_win32.exe.
 2. Follow the on-screen instructions and select the components you wish to install on the target computer (typically Full Install).
- . Install the Ekahau Manager and Ekahau Client on a laptop PC with at least Windows®2000, Windows® XP Professional, 1 GHz processor, 1 GB RAM, and 200 MB HD space
 1. To locate a wireless PC, the Ekahau Client service has to be installed on the device while the Client is running. (The PC should be typically equipped with a Wi-Fi adapter)
 2. In case of Windows® 2000, Windows® XP Professional, or Windows® 2003 Server computer, the Positioning Engine installer is used as described above, but only Ekahau Client and the included Wi-Fi drivers are installed.
- Get at least 3 Wi-Fi Access Points (any 802.11a/b/g model)
- . Get a map or floor plan of the area (in PNG, JPG, or BMP format)


As was described in chapter 5, Ekahau Client allows Ekahau Manager to retrieve signal data for the Calibration process (typically from a Wi-Fi adapter installed on a local computer), and allows the Positioning Engine to retrieve signal data from client devices for positioning.

This is why Ekahau Client must be installed and running on the Ekahau Manager laptop and on each client device that is to be tracked. The Windows® 2000, Windows® XP Professional, and Windows®2003 Server version of Ekahau Client provides *Ekahau Client Controller / Properties* dialog that can be used for:

- Viewing the Client status

NOTE: Red Client icon () on system tray means the Client service is not running. A yellow exclamation mark () means an error. Click the icon to view error status.

- Starting and stopping the Ekahau Client service
- Configuring the allowed or restricted Positioning Engines
- Enabling or disabling the writing of an Ekahau Client log file

To access Ekahau Client Properties dialog, the user must click the Ekahau icon () in Windows system tray (Figure A.1). If the icon is not displayed, the user must start the Client Controller by selecting:

Start > Programs > Ekahau > Client >Ekahau Client Controller.

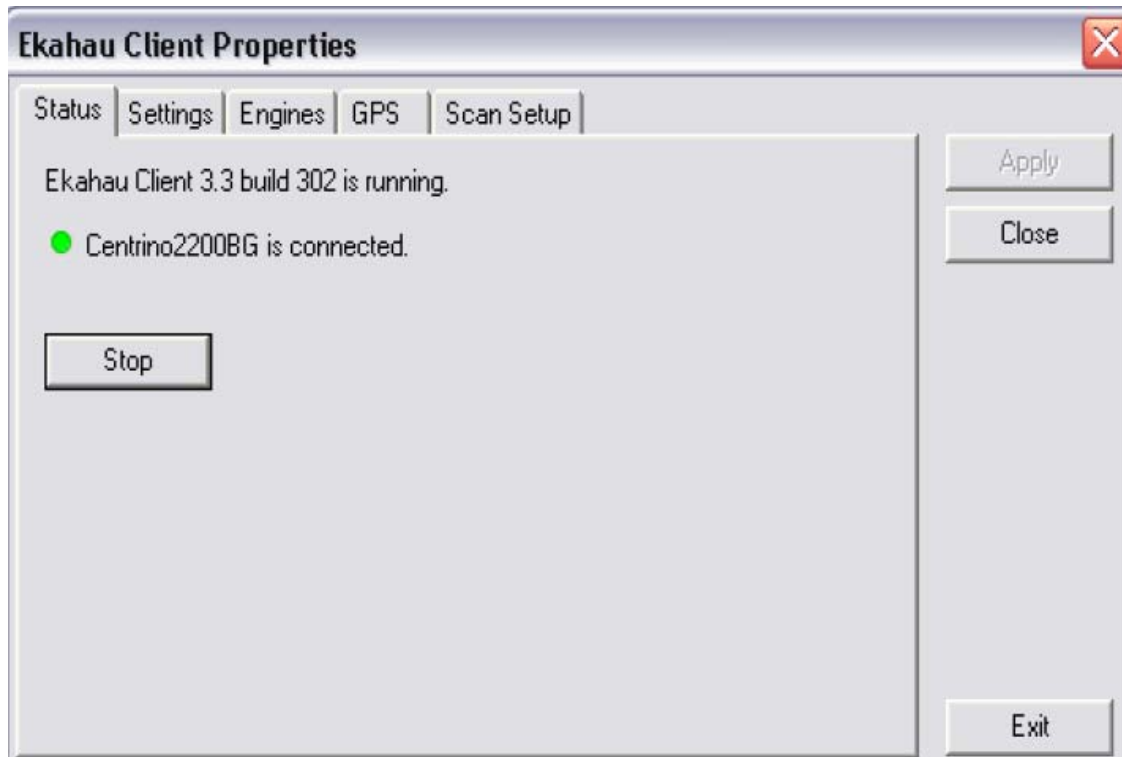


Figure A.1 - Ekahau Client Properties Dialog's Status Tab

Ekahau Client users can select the Positioning Engines that are allowed or disallowed to locate them. When using the default Positioning Engine settings (set automatically during the installation), all available Positioning Engines typically appear in the *Recently Discovered Positioning Engines* listbox (Figure A.2). The user can then uncheck the *Accept* checkbox to block positioning requests from that specific Positioning Engine. On the other hand, to allow requests from known Positioning Engines only, the user can uncheck the *Accept by default* checkbox. Positioning Engine settings can be saved by clicking the *Add...* button, selecting the name of the Positioning Engine from the drop-down listbox, and modifying the settings as appropriate.

If Positioning Engine communication settings were manually modified then the user may need to manually configure the Positioning Engine by clicking the *Add...* button, selecting *<custom settings>* from the drop-down listbox, and filling in or modifying the Positioning Engine details.

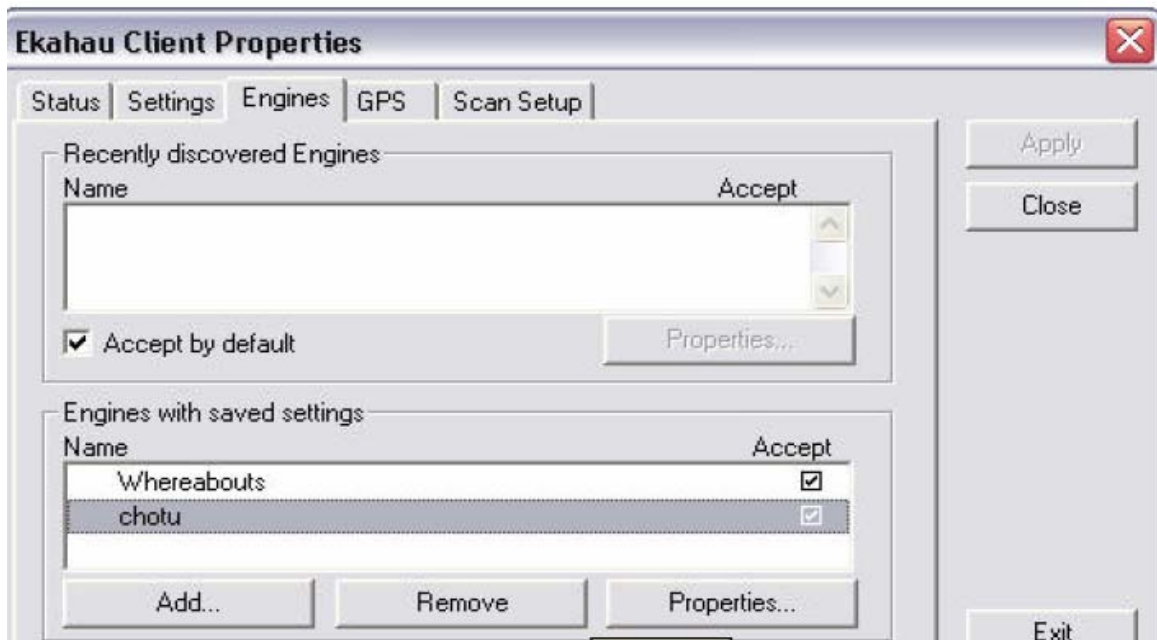


Figure A.2 - Ekahau Client Properties Dialog Displaying Positioning Engines Tab

After having installed the Ekahau Client and setting the necessary properties, the user needs to create a positioning model using Ekahau Manager by following the steps below:

- Double-clicking the Ekahau Manager shortcut.
- Providing the IP address and password of the Positioning Engine: If the Positioning Engine is installed on the same computer, default IP address and password are used by clicking *OK*. Otherwise, the correct name or IP address of the computer that runs the Positioning Engine are provided then *OK* clicked

(empty value is used for password unless Positioning Engine was configured to use special password) (Figure A.3).



Figure A.3 - Positioning Engine Login in Ekahau Manager Startup

- Creating a new Positioning Model or opening an existing Positioning Model (Figure A.4) either from a file or the Positioning Engine database (i.e. *New Positioning Model*). When opening a saved Positioning Model from the Positioning Engine, the user has to make sure that the Positioning Engine is running.

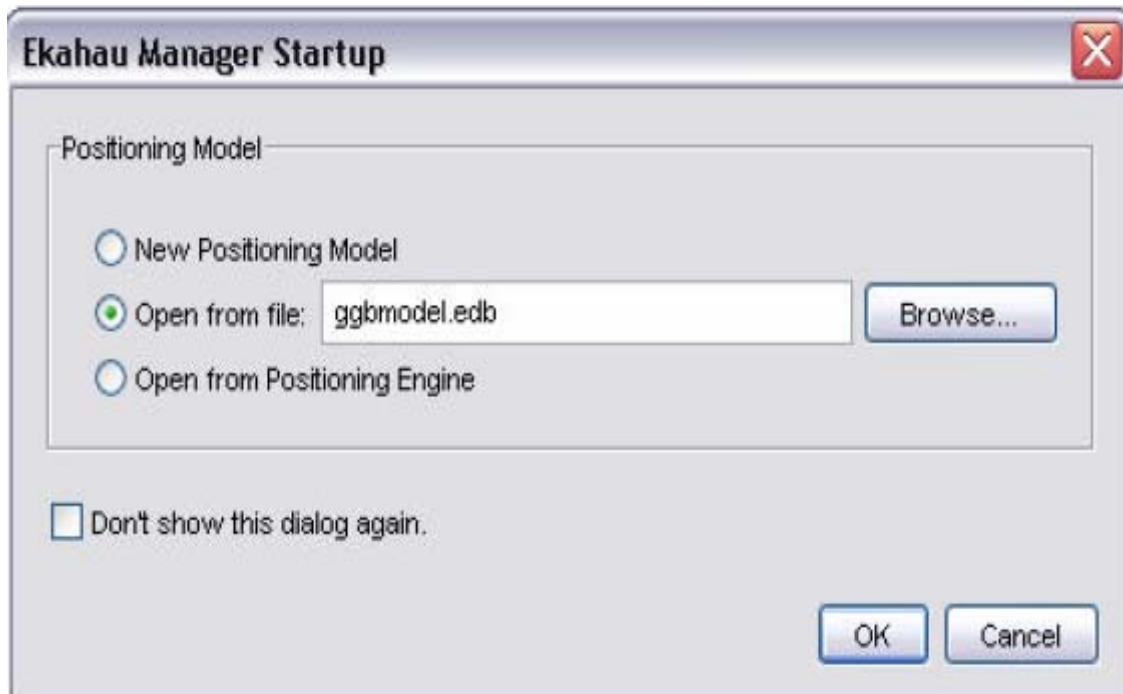


Figure A.4 - Ekahau Manager Startup Options

- Selecting *View > Signal Strengths* and checking the blue bars indicating signal strength (RSSI) values from active Access Points. (Figure A.5).

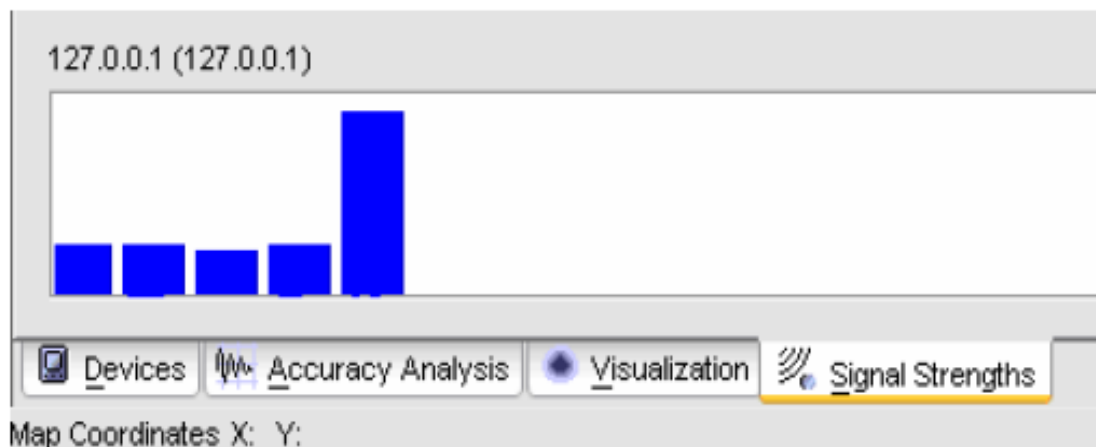


Figure A.5 - Signal Strengths from Access Points

- Loading and configuring a map of the coverage area: a map image for creating the Positioning Model is needed (supported map formats are PNG, JPG, and BMP). To import a map, *New Map...* from the *File* menu is selected. Alternatively, the user can right-click the *Maps* folder in the left window, select *New Map...* from the context menu and get a suitable map image (Figure A.6).

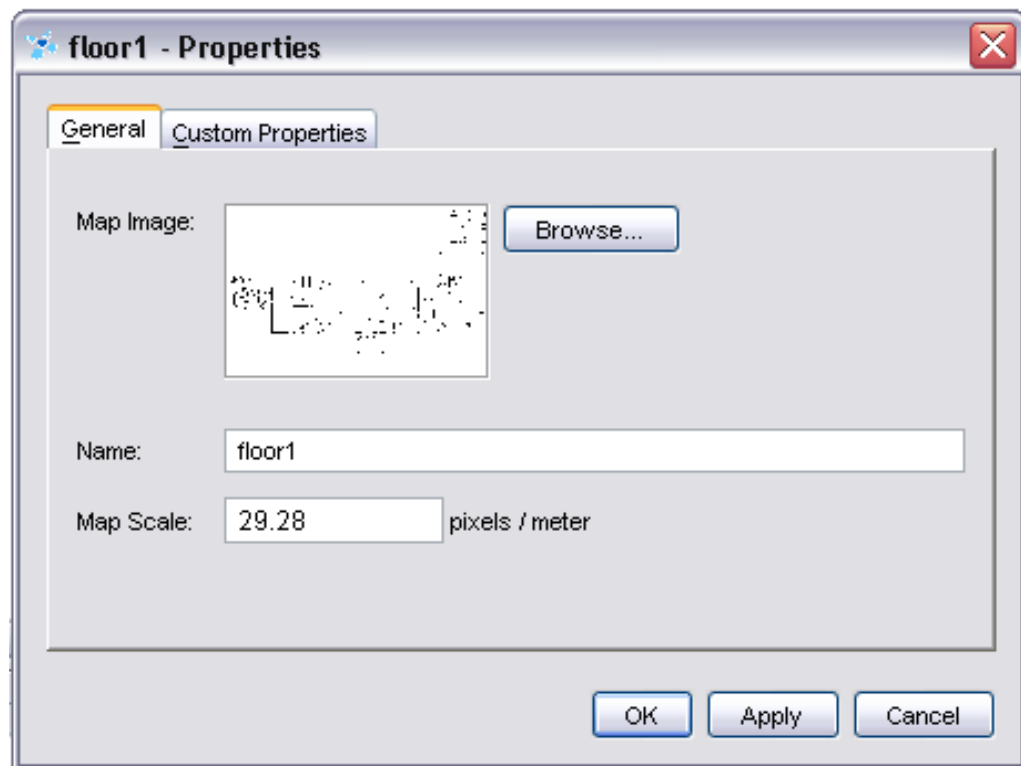


Figure A.6 - Map Properties Box

- Defining the correct map scale for each map image to ensure that Ekahau Positioning Engine functions accurately: If the user knows how many map pixels equal the map's selected distance unit (meter/foot) then the value in the *Map Scale* field can be typed. Otherwise, the user must click the blinking *Measure* tool and click a point on the map – for example the corner of a wall or another

measurable object. Then the user must click another point on the map, at least 3 meters (10 feet) apart from the starting point, walk to the marked location on site, measure the real-world distance between the two points with a measuring tape, and type it in the lower text field to set the map scale. Figure A.6 illustrates what the user sees when trying to open the map with the map scale included.

Next steps include performing a site survey/site calibration using Ekahau Manager. In this case, the user must:

- Display signal strength information in the *bottom view* by selecting *View > Signal Strengths*.
- Take the laptop and walk around the area while observing the number of detected signals and their signal strengths (RSSI) in Ekahau Manager. If necessary, the user has to move existing or add “dummy” Access Points to have at least 3, preferably 4 or more access point signals (blue bars) available in each location.
- Open a map and draw Tracking Rails using the Rail tool (🚂) once signal is detected from at least 3 access points (Figure A.7).

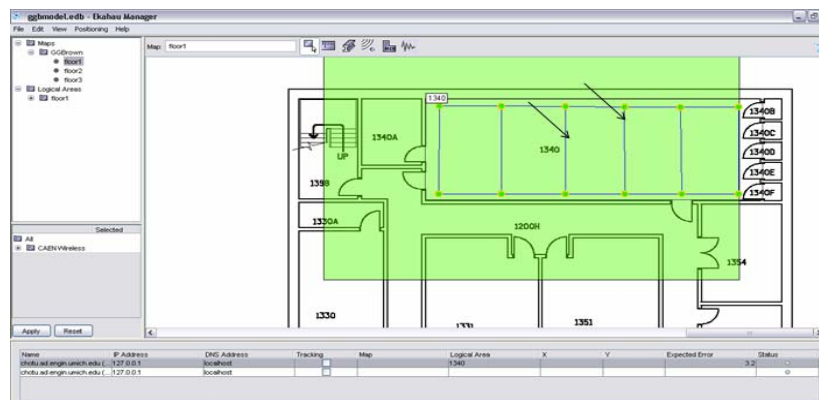




Figure A.7 - Drawing Tracking Rails

- Take the laptop and walk again around the area, stopping on the nearest tracking rail every 5-10m (15-30 ft), moving the mouse pointer to the current location on the rail with calibration tool () selected and clicking the map to record sample point (SP). It is important to note that the user does not have to move away from the current location until the record dialog disappears (in 10–20 s) and while remaining in the current location, the user has to turn around slowly a full 360° to record the signals from all directions without body affecting the readings. This associated received access point signals with the user's physical location on the map. Next, another sample is taken after waiting 2 seconds for the positioning software to update itself. This procedure is repeated for each sample point. When all the samples are taken, the sampling positioning model can then be saved not only to the Manager application but also to the Positioning Engine (select *File > Save to Positioning Engine*).

One important step in the whole process of creating the positioning model using Ekahau Manager is to draw and edit logical areas. Logical areas are user-drawn areas (polygons) that can have a name and other location-based properties. Logical areas are used to determine whether a client device is within a given area. This is achieved by:

- Activating the *Logical Area* tool by clicking the icon ()
- Clicking the map, moving the mouse, and clicking again until the desired area is drawn.

- Right-clicking the area in Browser View to *name* the area or to define *custom properties* for the area. The most probable Logical Area is displayed for each client device in *the Devices Tab* when the *Tracking* checkbox is selected for the device.

EPE offers a way for serving location/positioning information to local or remote applications using Ekahau Java SDK. The Ekahau SDK (Software Development Kit) Java package *com.ekahau.sdk* utilizes TCP sockets to connect to the Positioning Engine and provides quick and effective way for accessing location information, either from a local or remote computer. All what is needed is to import the Ekahau SDK package into the Java application and follow the provided examples. Using the SDK requires a working knowledge of the Java programming language and Java 2 Platform 1.4.2. The Positioning Engine's *TrackedDevice* objects represent wireless devices, so one object needs to be created for each physical device which is to be tracked. After recording a positioning model and saving it in the positioning engine with Ekahau Manager, *TrackedDevice* objects are used to return the coordinates, timestamp, status, speed, map, and any "logical area" information.

In order to use the SDK through an available java example, the user is first recommended to follow the steps below:

- Creating a J2SE 1.4.2 development environment.
- Adding *sdk.jar* in the classpath. The required classes are located in the *lib directory*, and the *sdk.jar* contains a manifest file which causes all the other required *.jar* files to be also added.

- Adding the *conf* directory into the classpath as it contains the configuration files required for running the Java example.

Following the aforementioned steps, Java applications can then be compiled with Ekahau SDK. In this case, the user has to:

- Open *SimpleTrackExample.java* from the *src\examples* folder under the *sdk* folder.
- Compile the sample program (using cmd window, `javac SimpleTrackExample.java` can be typed)
- Run the program; if the test application is running locally (Positioning Engine is installed and running on the same computer), then the following command can be used:

```
java examples.SimpleTrackExample
```

SimpleTrack Code Example:

```
import com.ekahau.sdk.*;
import java.util.*;
import java.io.*;
/**
 * Tracks a device until the user interrupts the action.
 * The device IP is given using a command line parameter.
 */
public class SimpleTrackExample
{
/**
 * @param args give the IP address as a command line parameter
 */

public static void main(String[] args) throws EngineException,
IOException {
String ipAddress = args.length > 0 ? args[0] : "127.0.0.1";//
local
//Connect to Positioning Engine
PositioningEngine.connect();
// Find the given device:
Device[] devices = PositioningEngine.findDevice(
"NETWORK.IP-ADDRESS ", ipAddress);
```

```

if (devices.length == 0) {
System.err.println("Device with IP address " + ipAddress
+ " not found.");
} else {
TrackedDevice device = new TrackedDevice(devices[0]);
// Create a simple handler for locations for one time usage:
LocationEstimateListener estimateListener = new
LocationEstimateListener() {
public void handleLocationEstimate(LocationEstimate
locationEstimate,
TrackedDevice device) {
// Change getLatestLocation to getAccurateLocation if
// you want a bit more accurate but a few 12 seconds delayed
// location.
Location loc = locationEstimate.getLatestLocation();
System.out.println("Location for the device is X:"
+ loc.getX() +
" Y:" + loc.getY());
}
};
StatusListener statusListener = new StatusListener() {
public void handleStatus(Status status, TrackedDevice
device) {
System.err.println("Device status: " + status);
}
};
// Add listeners and start tracking
device.addLocationEstimateListener(estimateListener);
device.addStatusListener(statusListener);
device.setTracking(true);
// Wait until the user inputs something
BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
in.readLine();
// Stop tracking and disconnect from the Positioning Engine:
device.setTracking(false);
PositioningEngine.disconnect();
}
}
}

```

Compiling and running the above program leads to the following output (Figure A.8).

```

*****
Output: Thu May 10 16:23:01 EDT 2007    User Position: X:4850.67431640625 Y:99.8
5266876220703
*****
Output: Thu May 10 16:23:07 EDT 2007    User Position: X:4833.876953125 Y:96.280
9060058594
*****
Output: Thu May 10 16:23:13 EDT 2007    User Position: X:4828.49072265625 Y:94.3
8910675048828
*****
Output: Thu May 10 16:23:19 EDT 2007    User Position: X:4834.599609375 Y:106.81
09399414062
*****

```

Figure A.8 - Ekahau Java SDK Output

In order to access positioning information through the JAVA SDK from a C++ application, a “pipe” needs to be created between the two different programming languages as shown below:

```
if (!(in = _popen("java examples.SimpleTrackExample ", "r")))
{ exit(1); } //open the pipe

if (fgets(buff, sizeof(buff), in) != NULL) { Position();}
//close the pipe
_pclose(in);
```

Based on the code above, the pipe is created using the `_popen` function. The `_popen()` function executes the command specified by the string *command*. It creates a pipe between the calling program and the executed command, and returns a pointer to a stream that can be used to either read from or write to the pipe. In this case, a pipe is being opened and positioning values are read from the JAVA SDK application through the pipe. The character string *mode* which specifies the type of access requested is "r". It allows the calling process to read the spawned command's standard output via the returned stream. A stream opened by `_popen()` should be closed by `_pclose()`.

UWB-based System Guidelines

Ultra-wideband (UWB) is a radio technology that can be used at very low energy levels for short-range high-bandwidth communications by using a large portion of the radio spectrum. As was mentioned in previous chapters, the Sapphire *DART* Ultra Wideband *Digital Active Real Time Tracking* system is designed for the tracking of personnel and/or

equipment. A system is defined as one (1) processing hub, four (4) or more receivers, one (1) or more reference tags, and multiple tags for individual assets/users. *Sapphire DART* uses short UWB pulse to determine the precise location of a UWB radio frequency identification tag

The *Sapphire DART* system is a precision tracking system and requires a degree of precision in installation. Specifically, the receiver and reference tag locations are critical to insure optimum performance. The accuracy of these positions will directly influence the accuracy of the results.

The H651 Processing hub must be placed within 1000 feet of the first receiver in the receiver chain. The Hub must also be placed in a location where either LAN or computer connection is available. A minimum of three receivers are needed to identify the x-y positions of one or more users/objects/assets. The ideal installation is a rectangular box with receivers in each of the four corners, installed as high as possible with the antenna of each receiver pointing towards the center of the box. After having placed the hub and located the receivers, the reference tag needs to be positioned in the area to be monitored in a position that is easily seen by each of the receivers. The optimal location for the reference tag is in the center of the box formed by the receivers. Each of the receivers must have a direct line of sight with the reference tag. The reference tag must be a position where it cannot easily be moved or obstructed during operation.

Next, cables need to be placed to connect receivers. Cable lengths, which limit the distance between the receivers, should not exceed 300 feet between each receiver. Cable

end breakage during installation is a common trouble-shooting problem. Cables must be shielded CAT 5E, 26AWG (or higher gauge) is recommended for best results. Receivers can be connected directly to a hub port connection or connected in a daisy chain to other receivers (Figure A.9). For all the experiments conducted using the UWB technology, the daisy chain configuration was adopted. Each receiver receives 48VDC from the previous receiver (or from the hub if it is the first receiver in the line) and passes the 48VDC on the next receiver in line via the CAT 5E cable.

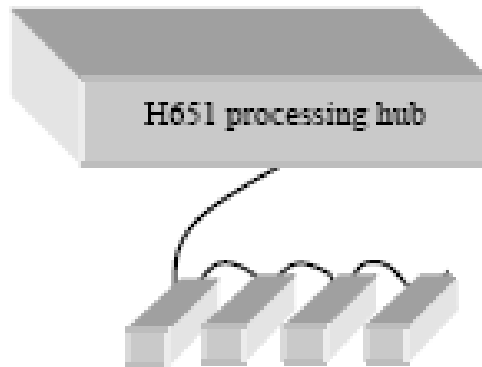


Figure A.9 – Daisy Chain Configuration

In setting up the receivers and connecting them in a daisy chain, major cautions and steps need to be taken into account:

- The hub port connector switch on the front panel hub should be in the OFF position prior to connecting cables to connector in order to avoid electrical damage to the hub and receiver.
- The first cable end (Ethernet connector) should be plugged into one of the hub connector ports and the other end to the “IN” connector of the first receiver in the

chain. Then the second cable is plugged, on one end, to the “OUT” connector of the first receiver and on the other end to the “IN” connector of the second receiver and so on so forth.

- After all receivers are connected through cables, the hub port connector should be switched on the front of the processing hub to the ON position.

Configuring and retrieving data requires a computer connection to the processing hub. The connection from the mobile computer to the hub is done through a wireless network with default IP settings for the hub (IP address: 192.168.1.202, Subnet mask: 255.255.255.0, and Default gateway address 192.168.1.1). Now that the computer is connected to the hub; the *Sapphire DART* can be configured for tracking one or more tags. To achieve this, first, the hub network parameters are configured through the configuration menu (A.10) by opening a web browser (e.g. Internet Explorer 6.0 or Netscape 7.1) to <http://192.168.1.202>.

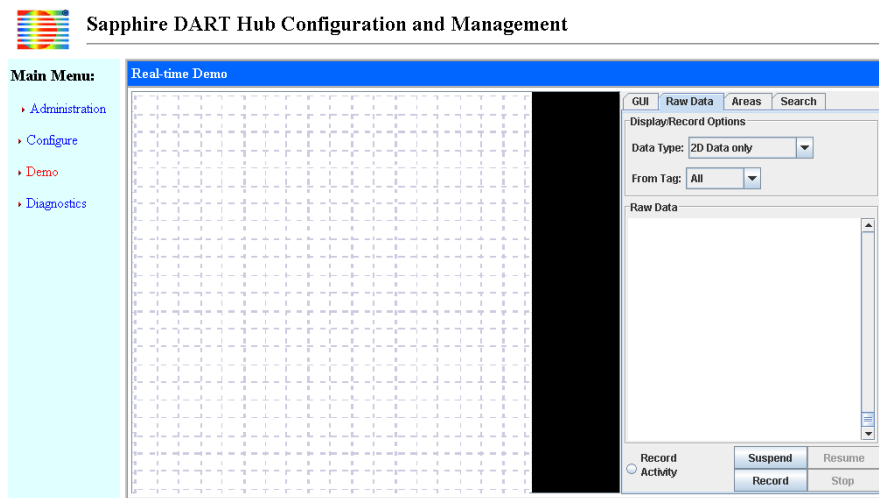


Figure A.10 – Sapphire Hub Configuration and Management Main Menu

Then, a user-defined coordinate system is established, receiver and reference tag positions configured, and boundary parameters configured. For tag localization, the positions of the receivers and reference tag must be established for the system to operate properly.

The user must define an origin and measure the x, y, and z positions of each receiver and reference (in feet or meters) with respect to that origin. Therefore, from the *Hub Configuration and Management* main menu, select *Configure*; next select the *Hub Setup* tab within the *Configure* menu (Figure A.11) then enable and enter location data (x, y, and z) for each active receiver and finally save the receiver positions to the hub.

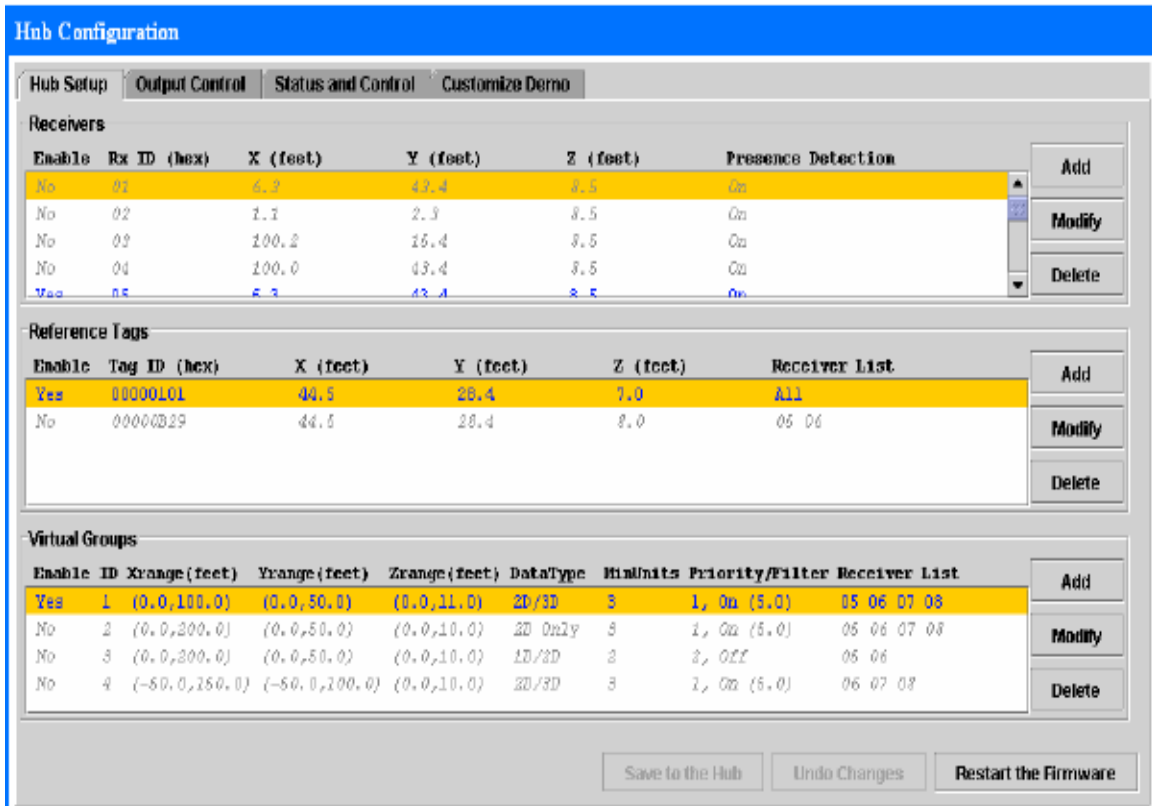


Figure A.11 - *Sapphire* Hub Setup Menu

A *Sapphire DART* system requires the use of one or more reference tags. These are necessary in order to establish a common time base among the receivers. For a reference tag to be useful, it is necessary for it to be in a location that is unobstructed from at least two receivers. Preferably, a position is chosen such that many receivers have such an unobstructed path. Many reference tags may be used, but it is generally a good idea to use as few as necessary. However, it is necessary that each receiver must be associated with at least one reference tag, and that each reference tag has more than one receiver associated with it. From the *Hub Configuration and Management* main menu, select *Configure*; next select the *Hub Setup* tab within the *Configure* menu (Figure A.10), enable, enter location data (x, y, and z) for each reference tag, and *Save to the Hub* on the *Hub Setup* menu.

After setting up the UWB equipment, a last step includes opening a socket connection to the Hub and retrieving x, y, and z coordinates. As mentioned in Chapter 5, the outputs resulting from a UWB tracking system application are provided from the hub to the client machine in the following format:

```
<Data Header>, <tag #>, <X>, <Y>, <Z>, <battery>, <timestamp>, <unit><LF>
```

For instance, as shown in the code below, only output information corresponding to "R" data header value and tag # "000022DD" is retrieved. Then, given that the user is interested in just obtaining <x>, <y>, <z> values, a string manipulation is performed on the incoming information output string and a string-float conversion method is used to obtain the final coordinates:

```

SocketClient s("192.168.16.3", 5117);
string l = s.ReceiveLine();
float n1,n2,n3;
if (l.find_first_of ("R")==0&& l.find("000022DD")==2)
    {
        l.erase (0,11);
        l.erase (15,16);

        a[0]= l.substr(0,5);
        a[1]= l.substr(6,4);
        a[2]= l.substr (11,4);
        for ( int i =0 ; i <3 ; i++)
            f[i]= atof (a[i].c_str());

        n1=f[0];
        n2=f[1];
        n3=f[2];
    }

```

Indoor GPS-based System Guidelines

As described in previous chapters, Indoor GPS is mainly a combination of laser transmitters, receivers, and computing systems designed to create and distribute high accuracy spatial information throughout a work area. Like GPS, positioning information is available in real time and there is no limit to the number of receivers allowed in the workspace—all working independently, yet simultaneously. Unlike GPS, this real time data can achieve sub-millimeter level accuracy.

The first step toward successful use of the Indoor GPS system is to evaluate the measurement requirements and select a general transmitter arrangement appropriate to the user's needs. There are two general geometric arrangements used with the system, Box and C-shape configurations (Figure A.12). When deciding which geometric arrangements to use, some factors to consider include:

- Overall area of measurement space
- Overall height of measurement space
- Potential line-of-sight obstructions between the transmitter and the sensor.
- Good convergence angles for position measurement

The best arrangements will provide line-of-sight to at least three transmitters with good convergence angle separation for each measurement. For all the experiments conducted using Indoor GPS (Chapter 8), the Box Configuration was used (Figure A.12b).

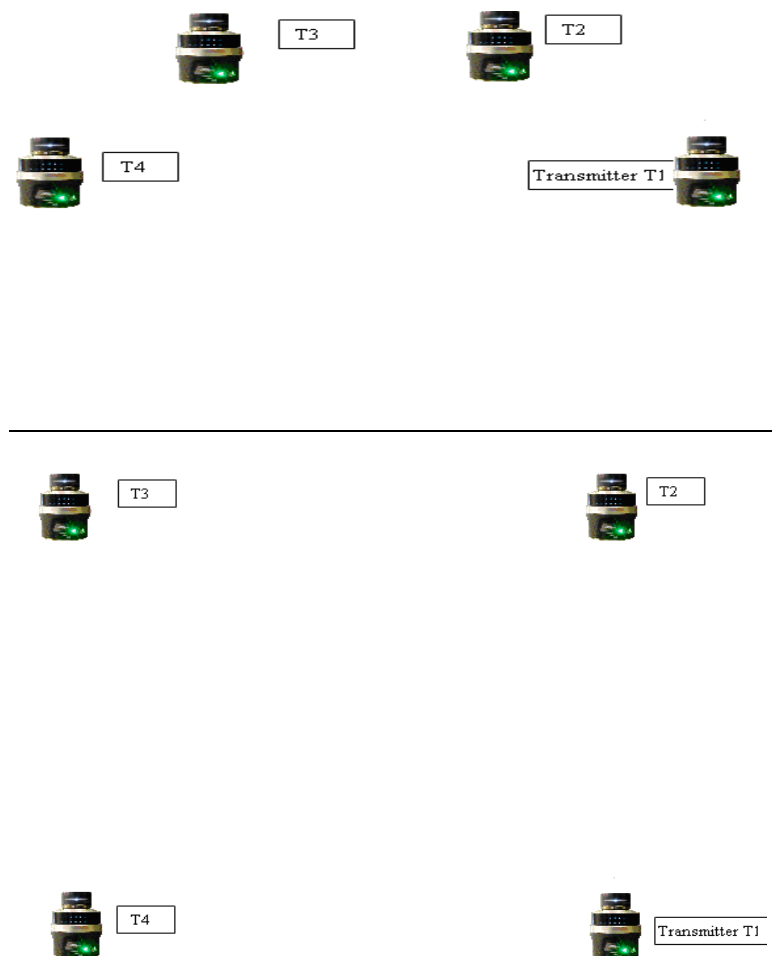


Figure A.12 – a) C-Shape Transmitter Configuration (Top), b) Box-Shape Transmitter Configuration (Bottom)

Each transmitter is placed on a tripod on a stable surface in the geometric arrangement chosen (Figure A.13).



Figure A.13 – Transmitter Mounted on a Tripod

It is not required to get the transmitters in the exact position and orientation, but following the general sequence and orientation will typically yield better overall measurement accuracy. When orienting the transmitters, the indicator lights on the Metrology-Grade transmitters are typically pointed “into” the work area. Although not required, this orientation guarantees that all transmitters will cover the area.

In order for the receiver to properly convert the timing intervals detected from the transmitters to position, the location (x,y,z) and orientation (R_x,R_y,R_z) of all the

transmitters in the work volume must be known. The six degrees of freedom DOF for each transmitter are precisely determined during the setup process. This is a critically important process because the quality of all position measurements performed with the system is primarily determined by the quality of the setup. Fortunately, Indoor GPS makes this process simple. The position and orientation of the transmitters determine how the system interprets the measured angles between the receivers and transmitters –used to calculate the receiver position using triangulation algorithms – after the setup is completed. The system will not be able to properly report the position of a receiver unless the positions and orientations of at least two visible transmitters are determined prior to the calculation. In theory, the angles could be collected-then a setup performed- so that positions could be determined in a post process. However the Indoor GPS system currently requires a setup prior to collecting data.

This setup procedure requires taking multiple samples or observations in the work volume, recording a given number of rays from each transmitter, and then calculating the “best fit” location and orientation of each transmitter (based on all observations) in the work area using a mathematical algorithm termed bundle adjustment. An “observation point” is defined as a stationary location where the system will sample the angles between a detector and the transmitters. The ideal number and location of observation points depends on the geometry (relative position and orientation of transmitters) of the transmitter setup and on the line-of-sight restrictions of the site (e.g. it may be impossible to see all transmitters from every observation).The most general rule in getting a good setup is to use observations that provide the most angular spread in the work area for each

transmitter. A typical layout for observation points consists of a minimum of eight points as shown in Figure A.14. Additionally, the receiver does not need to be placed in exact known locations during each setup observation and does not need to see all transmitters from every observation location.

Measuring angles alone is not sufficient to complete the setup process. A scale of the measurement volume needs to be established by taking observations of a known length or scale bars (Figure A.14). The system needs a minimum of one scale bar but more scale bars generally improve the result of the setup bundling routine.

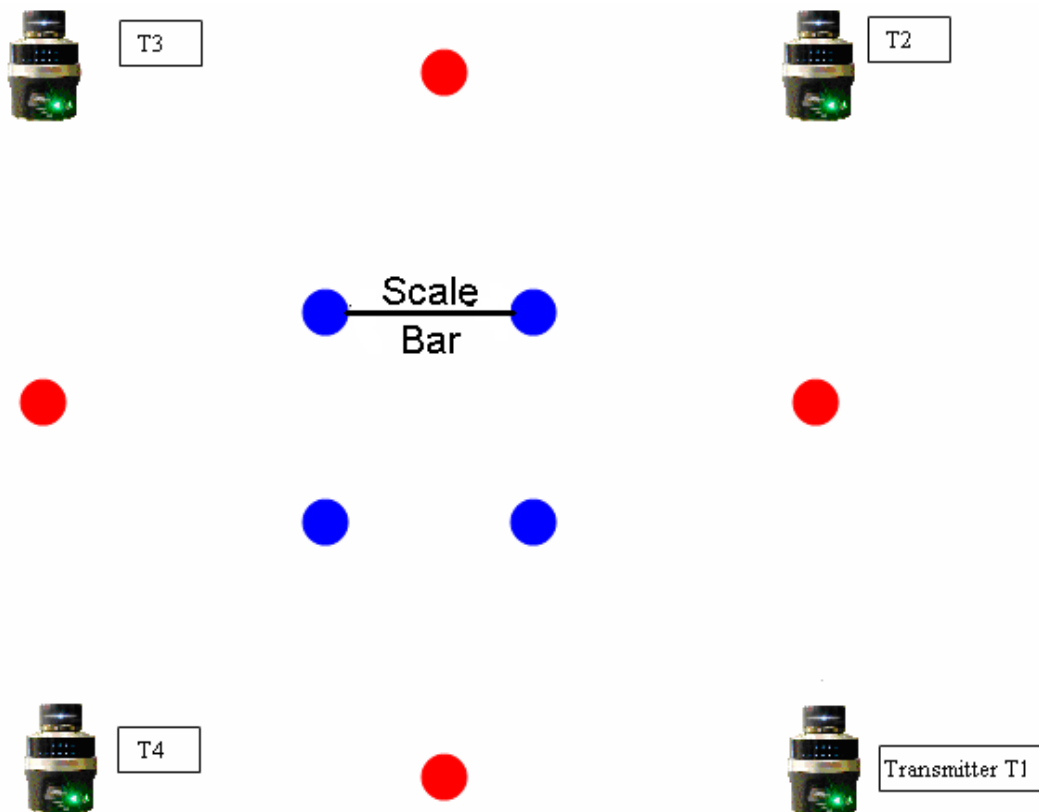


Figure A.14 – Box Geometry Observations and Scale Bar

Once all the observations are taken, the setup calculation routine is performed and the relative position and orientation of all transmitters is calculated. The scale bar is used to set the coordinate frame scale. The results of the setup process are then stored in the Indoor GPS Workspace file. In all the above processes, the Indoor GPS system requires a wireless network to allow communication between all of the receivers and the Data Viewer application of the Indoor GPS Workspace environment installed on the mobile computer.

Following the setup process, tracking information is extracted by first creating a listening socket descriptor on the IGPS address and port and then performing a byte manipulation on the incoming bytes data to get position values:

```
TCPSocket Sock("10.0.0.3", 10000);

char buffer[sizeof(double)*9]= {0};
int bytes=0;
int total_bytes=0 ;

while (total_bytes < sizeof(double)*9)
{
    if ((bytes=(Sock.recv(buffer,sizeof(double)*9)))<=0)
    {
        cout<<"unable to read socket";
    }

    total_bytes+= bytes;
}

memcpy(detector0[0],&buffer[sizeof(double)*0],sizeof(double));
memcpy(detector0[1],&buffer[sizeof(double)*1],sizeof(double));
memcpy(detector0[2],&buffer[sizeof(double)*2],sizeof(double));

n1= *(detector0[0]);
n2= *(detector0[1]);
n3= *(detector0[2]);
```

Appendix B

Computer Graphics Concepts and Techniques

In this Appendix, details on scene graph and computer graphics techniques are presented. Several pieces of pseudo code and code snippets are included to facilitate the discussion of topics introduced in Chapter 6 and to demonstrate how the mobile user's viewing frustum is defined and visible objects identified using computer graphics based intersection techniques.

Additionally, details on using keyboard event handlers within scene graph update callbacks to control certain actions and viewing scene algorithms are provided.

The presented information is also expected to help readers gain a better understanding of the whole process and data flow underlying the final location-aware visualization application.

Scene Graph Concept

A scene graph is a data structure used to organize and manage the contents of hierarchically organized scene data. In the realm of computer graphics, scene graphs are means to create and manipulate hierarchical organization of shapes, groups of shapes, and clusters of groups that collectively define the content of a scene. Scene graphs address low level problems that generally arise in scene composition and management.

In this research, as mentioned in previous chapters, an open source, cross platform graphics toolkit called OpenSceneGraph (OSG) was used to take advantage of scene graphs in effectively creating and managing 3D contents of graphical scenes and consequently use and apply computer graphics-based techniques such as raycasting,. Based around the concept of scene graphs, OpenSceneGraph provides an object oriented framework on top of OpenGL (Figure B.1) freeing the developer from implementing and optimizing low level graphics calls, and provides many additional utilities for rapid development of graphics applications.

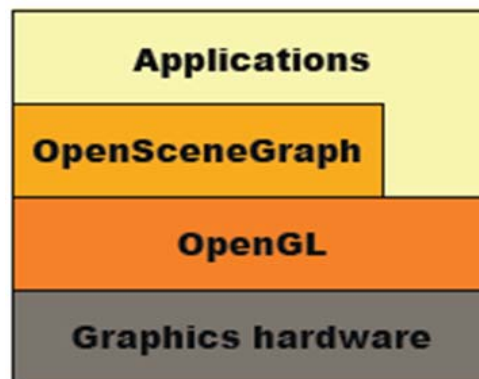


Figure B.1 – Graphics Framework

However, OpenGL and OSG coordinate systems are different (Figure B.2). The default OpenGL coordinate system is:

- **X+** = "to the right"
- **Y+** = "up"
- **Z+** = "out of the screen"

And that of OSG is:

- **X+** = "East"
- **Y+** = "North"
- **Z+** = "Up"

By default, the OSG core does not impose anything on the OpenGL default.

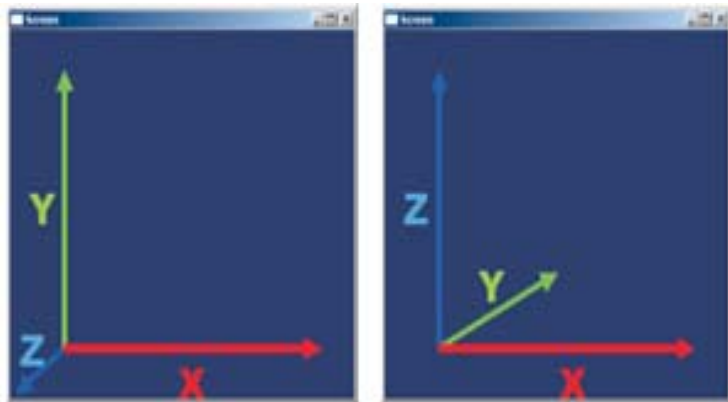


Figure B.2 – Default Coordinates Systems in OpenGL and OSG

OSG Core Classes

Like any scene graph, each OSG scene graph is a collection of nodes in a graph structure or tree connected together via child-parent relationship. A node is an object that can be part of or entirely comprise a scene graph. Each node is a collection of one or more fields (values) and methods that together perform a specific function. Each node encapsulates the semantics of what is to be drawn. A method applied to a parent node will affect all its

child nodes. The nodes are thereby ideally arranged in a hierarchical manner that corresponds spatially and semantically to the modeled scene. Nodes or core classes underlying a typical OSG scene graph tree are as follows:

- *osg::Nodes*: Base class defining all the internal nodes in the scene graph. It mainly consists of:
 - *osg::Group* is used to group objects together. It holds a set of child nodes like *osg::Transform* which is a parent of a basic transform, *osg::PositionAttitudeTransform* setting transforms via *Vec3* position and *Quat* attitude.
 - *osg::Geode* is a geometry node designed to hold geometry data. Geodes are leaves of the scene tree and have no children.
- *osg::Drawables*: Abstract base class for drawable graphics. One of its derived classes, *osg::Geometry*, defines the geometry of objects (i.e. vertices, faces, normals, etc.) and provides functions to manipulate such geometry. *osg::Drawables* can only be grouped and inserted to the scene graph via the leaf node *osg::Geode*.
- *osg::StateSet*: core class encapsulating OpenGL states and attributes and are attached to *osg::Nodes* and *osg::Drawables*.
- *osg::Texture* : core class encapsulating OpenGL texture functionality.

In this research, a scene graph tree was structured using the following OSG classes or nodes. Nodes are attached via links to the highest level node called the root node (Figure B.3).

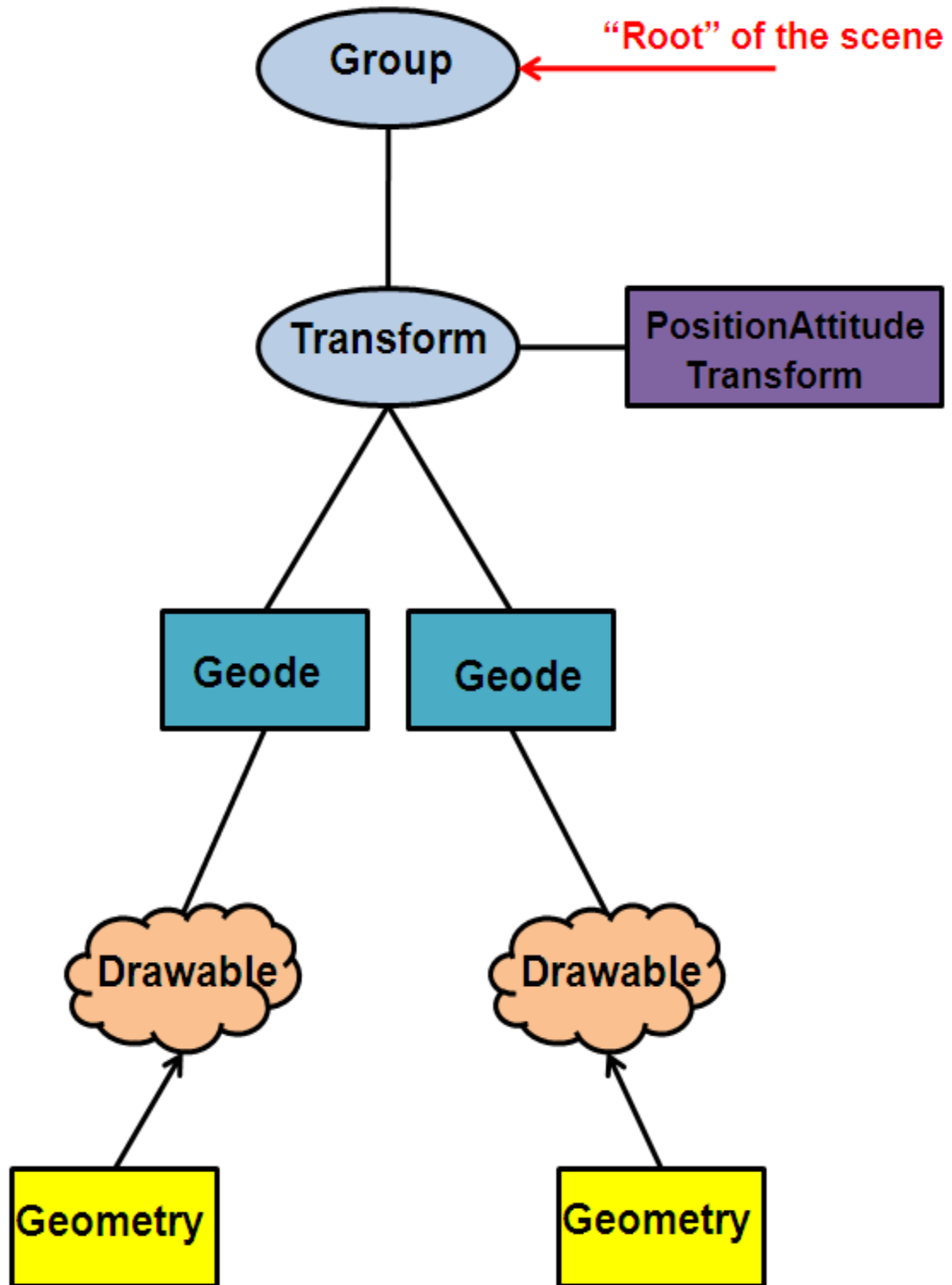


Figure B.3 – Typical Scene Graph Structure

Viewing Frustum Computation using OSG Concepts

Following up on Section 6.3.1 in Chapter 6, this section presents all steps adopted in creating and loading the user's viewing frustum or pyramid using OSG.

As mentioned in Chapter 6, first steps prior drawing and loading the pyramid in OSG included 1) computing the coordinates of the eight corner vertices using the near and far distances, field of view angles (HFOV and VFOV) and the mobile user's tracked position and 2) adjusting the coordinates by taking into account the user's head orientation angles using a set of transformation matrices, in particular rotation matrices. The following pseudo code (Figure B.4) summarizes the latter steps and further details are provided subsequently in code snippets:

```
[1] INPUT near and far distances and horizontal and vertical field of
      views
[2] COMPUTE pyramid or viewing frustum's near and far planes corner
      vertices coordinates based on the user's tracked position, head
      orientation, near and far distances and field of view angles
      • COMPUTE point 1 of the viewing frustum: xr[0],yr[0],zr[0]
        (first corner of the near plane)
      • COMPUTE point 2 of the viewing frustum: xr[1],yr[1],zr[1]
        (second corner of the near plane)
      • COMPUTE point 3 of the viewing frustum: xr[2],yr[2],zr[2]
        (third corner of the near plane)
```

- COMPUTE point 4 of the viewing frustum: `xr[3],yr[3],zr[3]`
(fourth corner of the near plane)
- COMPUTE point 5 of the viewing frustum: `xr[4],yr[4],zr[4]`
(first corner of the far plane)
- COMPUTE point 6 of the viewing frustum: `xr[5],yr[5],zr[5]`
(second corner of the far plane)
- COMPUTE point 7 of the viewing frustum: `xr[6],yr[6],zr[6]`
(third corner of the far plane)
- COMPUTE point 8 of the viewing frustum: `xr[7],yr[7],zr[7]`
(fourth corner of the far plane)

Figure B.4 – Pseudo Code for Defining the Coordinates of the Viewing Frustum Corner Vertices

The code snippets below present three main functions:

- 1) `FarNearCalc` calculates the eight vertices based on `neardist`, `fardist`, `xuser`, `yuser`, `zuser`, `HFOV`, and `VFOV`:

```
void FarNearCalc(double xuser, double yuser, double zuser)
{
//Near plane coordinates
  x[0]=x[1]=x[2]=x[3]= xuser+neardist;
  y[3]=y[0]= yuser+ (neardist* tan(HFOV/2));
  y[2]=y[1]= yuser -(neardist* tan(HFOV/2));
  z[3]=z[2]= zuser -(neardist* tan(VFOV/2));
  z[0]=z[1]= zuser+ (neardist* tan(VFOV/2));

//Far plane coordinates
  x[7]=x[6]=x[5]=x[4]= xuser+fardist;
  y[7]=y[4]= yuser+ (fardist * tan(HFOV/2));
  y[6]=y[5]= yuser -(fardist * tan(HFOV/2));
  z[7]=z[6]= zuser -(fardist * tan(VFOV/2));
  z[4]=z[5]= zuser +(fardist * tan(VFOV/2));
}
```


2) `orientation` gets the final matrix `matresult` based on the multiplication of the three rotation matrices (`rotxmat`, `rotymat`, and `rotzmat`) and translation matrices `transmat` back to and from the origin by `x2,y2,z2` (user's location):

```
void orientation (double x2, double y2, double z2)
{
    roll= dRoll*(3.14/180);
    pitch=dPitch*(3.14/180);
    yaw=dYaw*(3.14/180);

    rot.postmult(rot.rotzmat(yaw,matz),rot.rotymat(pitch,maty),matzy);
    rot.premult(rot.rotxmat(roll,matx),matzy,matxyz);
    matf=matxyz;

    rot.premult (matf,(rot.transmat(-x2, -y2, -z2,matt)),matres);
    rot.premult((rot.transmat(x2, y2, z2,matt)), matres,matresult);
}
```

3) `getcoord` adjusts the coordinates computed in 1) using the matrix obtained from 2).

In this case, `matresult` will be passed as a parameter to the function when called

(`matcoord` is `matresult`):

```
void getcoord (myMatrix matcoord)
{
    for (int i = 0; i <8 ;i++)
    {
        xr[i] = matcoord.mat[0][0]*x[i]+matcoord.mat[0][1]*y[i] +
        matcoord.mat[0][2]*z[i]+ matcoord.mat[0][3]*1 ;
        yr[i] = matcoord.mat[1][0]*x[i]+matcoord.mat[1][1]*y[i] +
        matcoord.mat[1][2]*z[i]+ matcoord.mat[1][3]*1 ;
        zr[i] = matcoord.mat[2][0]*x[i]+matcoord.mat[2][1]*y[i] +
        matcoord.mat[2][2]*z[i]+ matcoord.mat[2][3]*1 ;
    }
}
```

Having computed the coordinates of the viewing frustum corner vertices, the next step consisted of creating the geometry using OSG classes by 1) creating an array of vertices, `pyramidVertices` to store all `Vec3` vertices triples computed earlier in addition to the pyramid's peak vertex triple reflecting the user's viewpoint, and 2) adding the vertices to the geometry, `pyramidGeometry`:

```
osg::Geometry* pyramidGeometry = new osg::Geometry();
osg::Vec3Array* pyramidVertices = new osg::Vec3Array;
for (int i =0 ; i<8 ; i++)
pyramidVertices->push_back(osg::Vec3( xr[i], yr[i], zr[i]));
pyramidVertices->push_back(osg::Vec3( xuser, yuser, zuser));
pyramidGeometry->setVertexArray(pyramidVertices);
```

Based on the vertices defined above, pyramid base, pyramid top as well triangular and rectangular pyramid faces were then specified (Figure B.5).

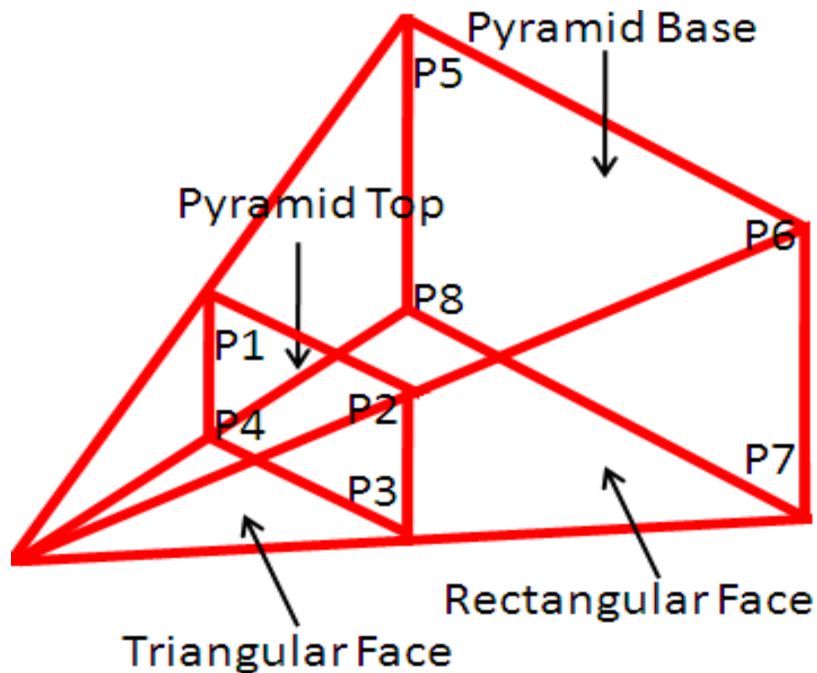


Figure B.5 – Viewing Frustum Graphical Components

For example, points P5-P8 of the pyramid were used to define the base and therefore an instance of the `DrawElementsUInt` class was created together with an `OpenGL PrimitiveSet` (i.e. `LINE_STRIP` in this case). For all the different pyramid faces, each created `PrimitiveSet` was added to the `pyramidGeometry`.

```
osg::DrawElementsUInt* pyramidBase =
new osg::DrawElementsUInt(osg::PrimitiveSet::LINE_STRIP, 0);
pyramidBase->push_back(7);
pyramidBase->push_back(6);
pyramidBase->push_back(5);
pyramidBase->push_back(4);
pyramidGeometry->addPrimitiveSet(pyramidBase);
```

The same procedure was repeated for all the other faces and vertices were each time specified in a counterclockwise order. Below are examples of rectangular and triangular faces, i.e. (P1-P2-P6-P5) and (P4-P9-P3) respectively (P9 being the user's point):

```
osg::DrawElementsUInt* pyramidFaceOne =
new osg::DrawElementsUInt(osg::PrimitiveSet::LINE_STRIP, 0);
pyramidFaceOne->push_back(0);
pyramidFaceOne->push_back(1);
pyramidFaceOne->push_back(5);
pyramidFaceOne->push_back(4);

pyramidGeometry->addPrimitiveSet(pyramidFaceOne);

osg::DrawElementsUInt* pyramidFace1 =
new osg::DrawElementsUInt(osg::PrimitiveSet::LINE_STRIP, 0);
pyramidFace1->push_back(3);
pyramidFace1->push_back(8);
pyramidFace1->push_back(2);
pyramidGeometry->addPrimitiveSet(pyramidFace1);
```

After defining and creating all the faces and sides of the pyramid, the last step included adding the geometry to the scene following the diagram below (Figure B.6):

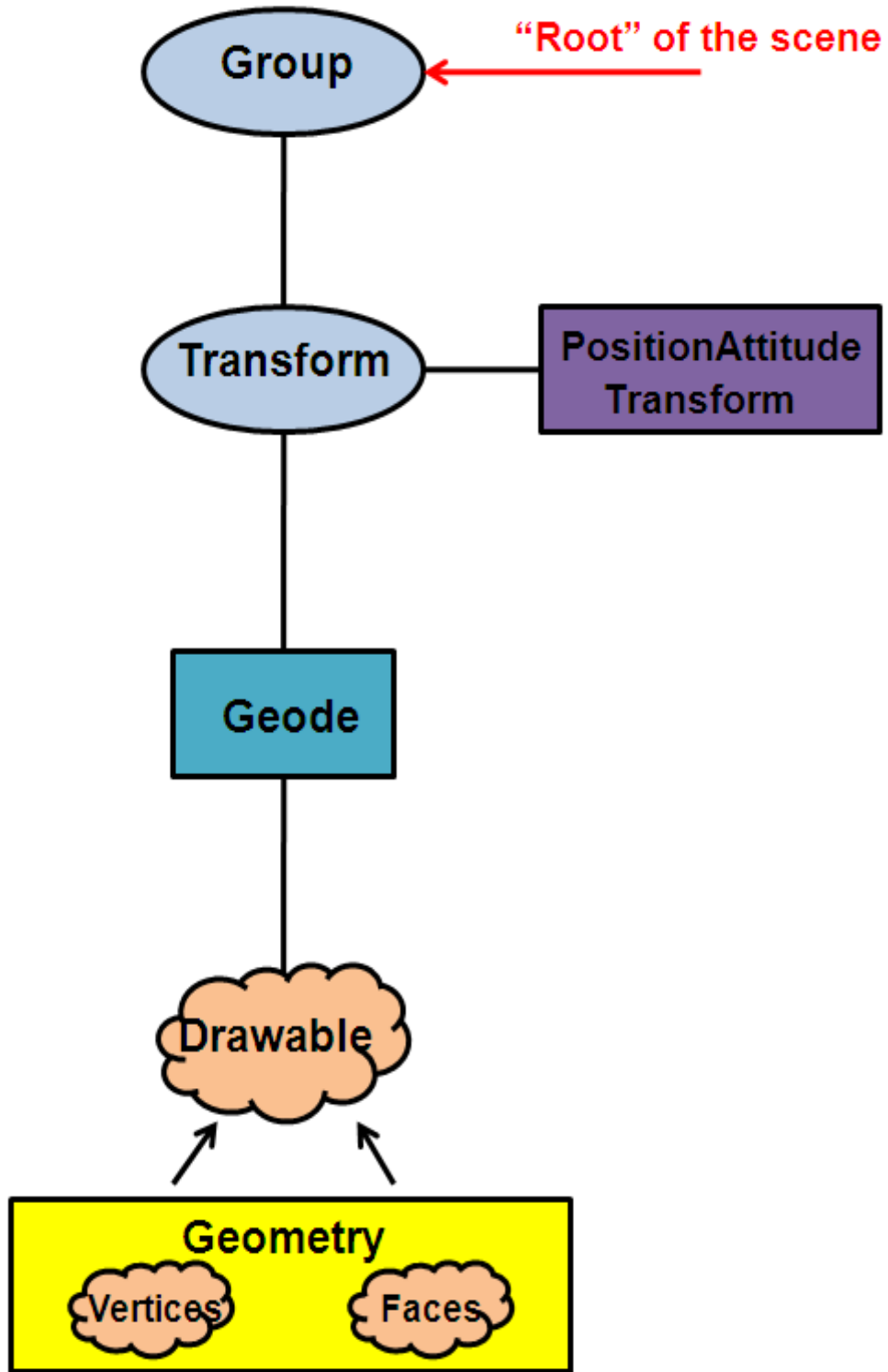


Figure B.6 – Viewing Frustum Scene Graph Structure

The related code consists of 1) creating a geometry node (geode) to hold the pyramid geometry, 2) setting a transform (uXform) to this node so that it is affected by the user's movement and finally 3) adding the transform to the root node:

```
osg::Group* root = new osg::Group();
osg::Geode* pyramidGeode = new osg::Geode();
pyramidGeode->addDrawable(pyramidGeometry);

osg::PositionAttitudeTransform* uXform = new
osg::PositionAttitudeTransform();
osg::Vec3 pposition;
pposition.set(osg::Vec3(xuser,yuser,zuser));
uXform->setPosition(pposition);
uXform->addChild(pyramidGeode);

root->addChild(uXform);
```

All the above steps can be summarized in the following pseudocode (Figure B.7).

```
[1] DRAW pyramid with the tip symbolizing the user's information and
    using the eight computed coordinates
    • STORE tip of pyramid with all other eight coordinates in a 3D
      vector
    • DRAW lines between the different stored coordinates to obtain
      the pyramid base, top and different faces.

[2] SET the drawn pyramid/ viewing frustum tip at the user's location
[3] ATTACH pyramid to this user's transform
[4] LOAD pyramid in the scene by adding the user's transform to the
    root
```

Figure B.7 – Defining Viewing Frustum

Interference Detection Tests

As mentioned in Chapter 6, in order to interpret which objects in the user's surrounding environment are currently visible, the computed viewing frustum was represented as a

geometric model and then tested for geometric interference with CAD representations of building objects using V-Collide collision detection technique, in particular its component RAPID.

RAPID is a robust and accurate polygon interference detection library for large environments composed of unstructured models. It is applicable to polygon soups, i.e. models which contain no adjacency information, and obey no topological constraints. The RAPID library has a very simple user interface. The API requires only that `RAPID.H` and link to `libRAPID.a` be included to be able to use the library. The C++ object into which polygons/objects are loaded is called `RAPID_model`. In this case, for each frustum-building object pair two `RAPID_model` objects were created:

```
RAPID_model *obb1 = new RAPID_model;  
RAPID_model *obb2 = new RAPID_model;
```

Before calling the interference detect routine and running overlap tests, transformations defining the placement of both boxes in world space were constructed as follows:

```
double R1[3][3], R2[3][3], T1[3], T2[3];  
  
R1[0][0] = R1[1][1] = R1[2][2] = 1.0;  
R1[0][1] = R1[1][0] = R1[2][0] = 0.0;  
R1[0][2] = R1[1][2] = R1[2][1] = 0.0;  
  
R2[0][0] = R2[1][1] = R2[2][2] = 1.0;  
R2[0][1] = R2[1][0] = R2[2][0] = 0.0;  
R2[0][2] = R2[1][2] = R2[2][1] = 0.0;  
T1[0] = 1.0; T1[1] = 0.0; T1[2] = 0.0;  
T2[0] = 0.0; T2[1] = 0.0; T2[2] = 0.0;
```

Next, a collision query was performed to determine first contact:

```
RAPID_Collide(R1, T1, obb1, R2, T2, obb2, RAPID_FIRST_CONTACT);
```

If any contact is detected, then exact and accurate geometric primitive intersection tests (i.e. intersection between pair of triangles) are then performed to confirm collisions. In this case, boxes were loaded with triangles using `AddTri()` method. Each load consists of a torus of $2*n1*n2$ triangles (i.e. $n1=n2=50$):

```
for(uc=0; uc<n1; uc++)
    for(vc=0; vc<n2; vc++)
    {
        //generate all tori

        // . . .

        // . . .

        obb1->AddTri(p1, p2, p3, count);
        obb1->AddTri(p4, p2, p3, count+1);
        obb2->AddTri(p1, p2, p3, count);
        obb2->AddTri(p4, p2, p3, count+1);
    }
```

Then again, the RAPID collision routine was performed and collision points confirmed:

```
RAPID_Collide(R1, T1, obb1, R2, T2, obb2, RAPID_ALL_CONTACTS);

for(i=0; i<RAPID_num_contacts; i++)
{
    printf("\t contact %4d: tri %4d and tri %4d\n",
        i, RAPID_contact[i].id1, RAPID_contact[i].id2);
}
```

As mentioned in Section 6.3.3 in Chapter 6, in order to increase precision in identifying relevant contextual objects, an interference detection technique, called raycasting was used. The following pseudo code (Figure B.8) summarizes the steps involved in using this technique:

- [1] DEFINE a set of line segments (rays) with the first endpoint as the user's location and the second endpoint as one of the corners

```

    or any other intermediate points on the far plane of the computed
    viewing frustum
[2]  CREATE an intersectvisitor to do the line segments intersection
    traversals
[3]  ADD all line segments to this intersector
[4]  SET the loaded CAD models to accept the intersector
[5]  CHECK first for (object-line of sight ray) intersection detection
[6]  IF intersection detected
    [7]  IDENTIFY and PRIORITIZE the contextual objects
[8]  END IF
[9]  ELSE
    [10] IDENTIFY the contextual objects within a 20 degrees angle
    away from the line of sight (depends on the visible volume of
    space and field of view angles)
[11] END ELSE

```

Figure B.8 – Pseudo Code for Identifying Contextual Objects using Raycasting

The code snippet below represents the steps followed in identifying objects or nodes with the line of sight ray: 1) A `lineSegment` defined by two `Vec3` points, the first being the user's viewpoint and the second being the centerpoint of the far plane, was created, 2) An instance of the `IntersectVisitor` was created to do the line intersection traversals starting from the root node, 3) A hitlist was created to fill it with all intersections for the line of sight ray at (X,Y), and 4) Hit nodes were identified.

```

lineSegment[0]->set(osg::Vec3(userPos.x(),userPos.y(),userPos.z()),
osg::Vec3(userPos.x()+fardist,userPos.y(),userPos.z()));
osgUtil::IntersectVisitor intersector;

```



```

intersector.addLineSegment(lineSegment[0]);
root->accept(intersector);

hits=intersector.getHitList(lineSegment[0]);
if (!hits.empty())
{
    cout<<"Intersection detected" <<endl;
    for ( int i=1; i < hits.size()-2;i+2)
    {
        osgUtil::Hit Hits = hits.at(i);
        osg::NodePath np = Hits.getNodePath();
        osg::Node *node = (np.size()>=1)?np[np.size()-1]:0;
        node-> getName();
    }
}

```

The same procedure was adopted for all other rays cast away from the line of sight and all detected nodes/objects were stored in a vector of strings.

Update Callbacks and Keyboard Event Handlers

In this research, as the mobile user was moving, position and head orientation tracking information (Appendix A), viewing frustum coordinates, intersection tests and retrieval processes (Appendix C) were continuously updated using callbacks controlled by keyboard event handlers.

Callbacks can be thought of as user-defined functions that are automatically executed depending on the type of traversal (update, cull, draw) being performed. Callbacks can be associated with individual nodes or they can be associated with selected types (or subtypes) of nodes. During each traversal of a scene graph, if a node is encountered that has a user-defined callback associated with it, that callback is executed. In this research, the mobile user's node together with the pyramid node had the same update callback

associated with them. However, these nodes and related processes needed to be controlled in the scene graph based on user keyboard input. Therefore, keyboard event handlers were set up to communicate with the update callback. In this case, two event handlers were created, one to control the processing of position and orientation tracking information obtained from the respective technologies (Appendix A) and the other to control, at each tracked user's context, the computation of the viewing frustum vertices coordinates, intersection tests and contextual information retrieval process (Appendix C).

In order to achieve this communication mechanism between the aforementioned keyboard event handlers and the update callback, a class to store keyboard state information (i.e. `trackingRequest` and `retrievalRequest`) was first created:

```
class userInputDeviceStateType
{
    public:
        userInputDeviceStateType()
            : trackingRequest(false), retrievalRequest(false){}

        bool trackingRequest;
        bool retrievalRequest;
};
```

The next step consisted of creating an event handler class (`keyboardEventHandler`) to keep the keyboard state current based on most-recent keyboard events. The base class `osgGA::GUIEventHandler` was used to define custom actions for GUI keyboard events. Keyboard event handlers were set to have access to the right data. This data is encapsulated in an instance of the created class `userInputDeviceStateType`. Data members (pointers to an instance of `userInputDeviceStateType`) were then added to the event handler as follows:

```

class keyboardEventHandler : public osgGA::GUIEventHandler
{
public:

keyboardEventHandler(userInputDeviceStateType* tids1,
userInputDeviceStateType* tids2)
    {
        userInputDeviceState1 = tids1;
        userInputDeviceState2 = tids2;
    }
    // ...
protected:
userInputDeviceStateType* userInputDeviceState1;
userInputDeviceStateType* userInputDeviceState2;
};

```

The bulk of the `keyboardEventHandler` class is the `handle` method overridden from the base class `osgGA::GUIEventHandler` version. The method takes two arguments, in particular an instance of the `osgGA::GUIEventAdapter` class that allows receiving GUI events. This method, shown below, updates the `userInputDeviceState1` and `userInputDeviceState2` states to indicate requests to:

- 1) Get tracking information and move the user/frustum by pressing 't' then temporarily stop the processing of tracking information and “lock” the user’s context by pressing another key 'c' (`trackingRequest`).
- 2) Update the frustum vertices coordinates, run the intersection tests and retrieve contextual information at each user’s “locked” context by pressing 'r' then stop those processes by pressing 's' (`retrievalRequest`).

```

bool keyboardEventHandler::handle(const osgGA::GUIEventAdapter&
ea, osgGA::GUIActionAdapter& aa)
{
    switch(ea.getEventType())
    {

```

```

case(osgGA::GUIEventAdapter::KEYDOWN):
{
    switch(ea.getKey())
    {

        case 't':
            userInputDeviceState1->trackingRequest = true;
            return false;
            break;

        case 'c':
            userInputDeviceState1->trackingRequest = false;

            return false;
            break;

        case 'r':
            userInputDeviceState2->retrievalRequest = true;
            return false;
            break;

        case 's':
            userInputDeviceState2->retrievalRequest = false;

            return false;
            break;

        default:
            return false;
    }
}

default:
    return false;
}
}
}

```

An update callback class (`updateuserPosCallback`), derived from `osg::NodeCallback` class, was also created. Similar to the `keyboardEventHandler` class, it was designed to have access to the keyboard state data so it can update the scene graph correctly. Therefore, this class has two data members each pointing to the same instances of `userInputDeviceStateType`(`userInputDeviceState1` and `userInputDeviceState2`). These two pointers were then used within the callback. In this case, the callback moves the user/frustum nodes (both being under the same position attitude transform) if the first

keyboard state indicates a user request to extract tracking information (if (userInputDeviceState1->trackingRequest)), if the 't' key is pressed. Additionally, the callback updates the frustum vertices coordinates, runs interference tests and identifies contextual information if the second keyboard state indicates a user request to retrieve information (if (userInputDeviceState1->retrievalRequest)), if the 'r' key is pressed. The class definition is as follows:

```
class updateuserPosCallback : public osg::NodeCallback {
public:

updateuserPosCallback::updateuserPosCallback(userInputDeviceStateType*
userIDevState1,userInputDeviceStateType* userIDevState2)
{

    userInputDeviceState1 = userIDevState1;
    userInputDeviceState2 = userIDevState2;

}

virtual void operator()(osg::Node* node, osg::NodeVisitor* nv)
{

    osg::PositionAttitudeTransform* pat =
        dynamic_cast<osg::PositionAttitudeTransform*> (node);
    if(pat)
    {
        if (userInputDeviceState1->trackingRequest)
        {

            //... Obtain Tracking Info from either Ekahau, UWB, or Indoor
            GPS (Appendix A)

        }
        if (userInputDeviceState2->retrievalRequest)
        {
            //... Update frustum vertices coordinates using the three
            functions FarNearCalc,orientation,and getcoord (Appendix B)

            //... Run Intersection tests, Rapid and raycasting, and
            identify contextual objects (Appendix B)

            //... Retrieve contextual information from either CIS/2 or MS
            Access database (Appendix C)
        }
    }
    traverse(node, nv);
}

protected:
```

```

userInputDeviceStateType* userInputDeviceState1;
userInputDeviceStateType* userInputDeviceState2;

};

```

Having completed the framework for communicating between keyboard and update callback, a subsequent step consisted of creating two instances of the `userInputDeviceStateType` and setting them as arguments for both the constructor of the update callback and that of the event handler as follows:

```

userInputDeviceStateType*tIDevState1=new userInputDeviceStateType;
userInputDeviceStateType*tIDevState2=new userInputDeviceStateType;

uXform->setUpdateCallback
(new updateuserPosCallback(tIDevState1,tIDevState2));

keyboardEventHandler*userEventHandler=new
keyboardEventHandler(tIDevState1, tIDevState2);

```

Configuring the Viewer and Camera

This section covers the final phase in any OSG visualization application, viewing the scene. This consists of first creating a viewer object (`osgProducer::Viewer viewer`), initializing it with standard settings (standard settings include a standard keyboard and mouse interface), assigning the built scene graph to it, and creating the viewer windows to start the required threads:

```

viewer.setUpViewer(osgProducer::Viewer::STANDARD_SETTINGS);
viewer.setSceneData(root);
viewer.realize();

```

Since keyboard event handlers were introduced in this research than an important step included adding the event handler created instance (`userEventHandler`) to the viewer's

```

EventHandlerList:

viewer.getEventHandlerList().push_front(userEventHandler);

```

Next, a viewer simulation loop was set up:

```
while(!viewer.done())
{
    viewer.sync();
    viewer.update();
    viewer.frame();
}
```

Within this loop, methods were invoked on the viewer object for specific purposes:

- `viewer.sync()`: method used to wait for all draw and cull threads to complete.
- `viewer.update()`: main method in the loop used to initiate scene graph traversal to update nodes. Additionally, any node for which an update callback has been set up is also updated after handling keyboard events.
- `viewer.frame()`: method invoked to initiate the cull and draw traversals of the scene.

An update callback was executed each time a scene graph update traversal was performed. Since the code associated with update callbacks happens once per frame and before the cull traversal is executed, the code could be inserted in the main simulation loop between the `viewer.update()` and `viewer.frame()` calls. While the effect of the code is the same, callbacks provide an interface that is easier to update and maintain.

Another way to view the scene was done from a first person view, in this case the mobile user. This was achieved by implementing a camera that continuously follows the user's node, which means positioning a camera with a matrix updated with the correct mobile user's world position and orientation. This matrix is therefore the multiplication of the

rotation matrix, involving the pitch, roll and yaw angles, by the translation matrix (x,y,z position values) . Using the Matrix class methods `makeTranslate()` and `makeRotate()`, the above matrix was computed as follows:

```
cameraRotation.makeRotate(  
osg::DegreesToRadians(dPitch), osg::Vec3(0,-1,0),  
osg::DegreesToRadians(dRoll), osg::Vec3(1,0,0) ,  
osg::DegreesToRadians(dYaw), osg::Vec3(0,0,1));  
  
cameraTrans.makeTranslate( userPos.x(),userPos.y(),userPos.y());  
myCameraMatrix = cameraRotation * cameraTrans;
```

It is important to note that when implementing cameras, translations and rotations become inverted compared to translating objects. This can be thought of as translating and rotating the world objects while the camera stays in place. Therefore, to use the camera position/orientation matrix created above to position and orient the Viewer's camera matrix, the matrix needs to be inverted first:

```
osg::Matrixd i = myCameraMatrix.inverse(myCameraMatrix);
```

Producer and `osg::Matrix` class matrices (like those created above) normally represent a 'Y' up coordinate system. Therefore, in addition to inverting the matrix, a rotation from 'Y' up to 'Z' up is needed so that a world orientation is provided. This was done by rotating -90 degrees about the x-axis.

The `setViewByMatrix()` method of the viewer class was then used, taking as parameter the adjusted matrix, and the method call was placed between the `viewer.update()` and `viewer.frame()` calls within the simulation loop:

```
viewer.setViewByMatrix( (  
Producer::Matrix(i.ptr()))  
* Producer::Matrix::rotate(-M_PI/2.0, 1, 0, 0));
```


Appendix C

Information Access and Retrieval Processes from Databases

In this Appendix, the details of the information retrieval process from projects databases, in this case MS Access database, are documented. As described in Chapter 7, there are three main components underlying this process: database structure design (MVC model component), user interface design (MVC view component), and business logic design (MVC controller component)

MS Access Database Design

As was mentioned in Chapter 7, an object-relational database structure was implemented. This consisted of creating many interconnected tables with object-oriented features, i.e. tables mapped to classes. Figure C.1 shows the database structure presented earlier.

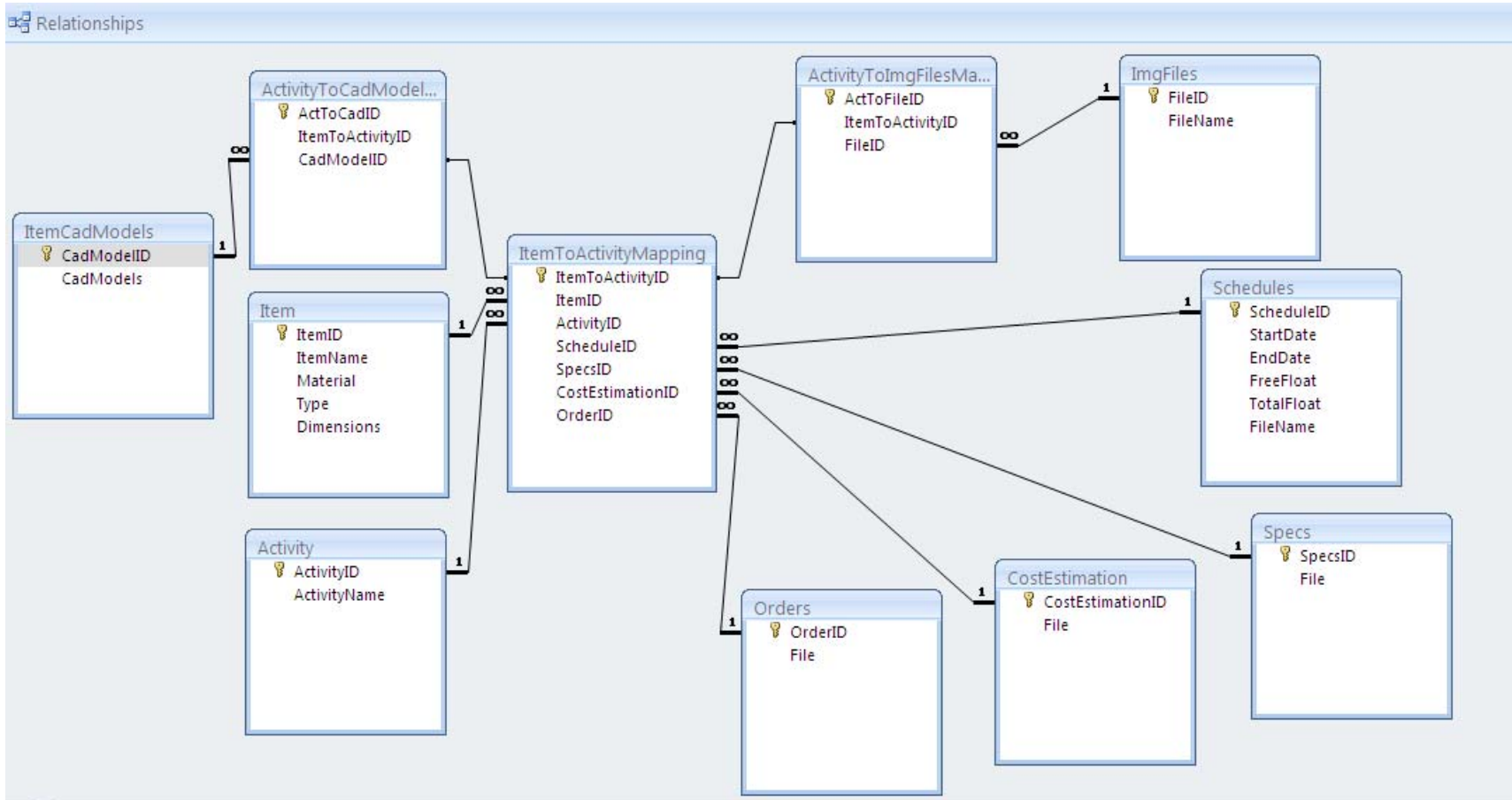


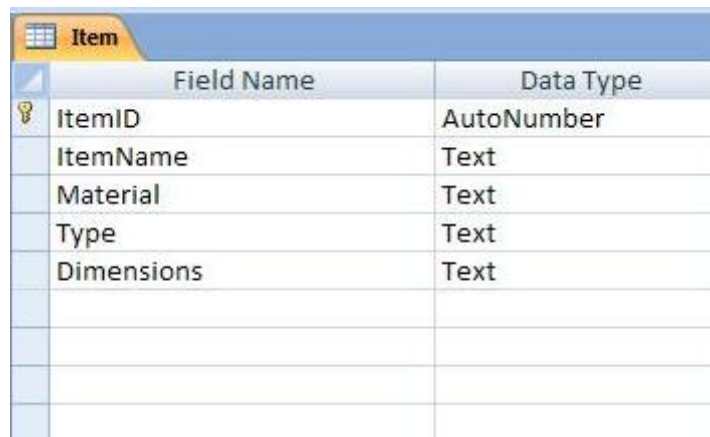
Figure C.1 – Database Structure

MS Access Database Design

As was mentioned in Chapter 7, an object-relational database structure was implemented. This consisted of creating many interconnected tables with object-oriented features, i.e. tables mapped to classes. Figure C.1 shows the database structure presented earlier.

Each of the tables above points to different types of information. For instance, Table C.1, *Item* table, includes general information on building objects (*ItemName*) such as the material, type, and dimensions.

Table C.1 – Item Table



	Field Name	Data Type
🔑	ItemID	AutoNumber
	ItemName	Text
	Material	Text
	Type	Text
	Dimensions	Text

Therefore, general information pertaining to any building object (*Item*) includes material, type, dimensions an *Item* class would look as follows:

```

public class Item
{
    private long itemID;
    private string itemName;
    private string material;
    private string type;
    private string dimensions;

    public long ItemID
    {
        get { return this.itemID; }
        set { this.itemID = value; }
    }

    public string ItemName
    {
        get { return this.itemName; }
        set { this.itemName = value; }
    }

    // . . .
}

```

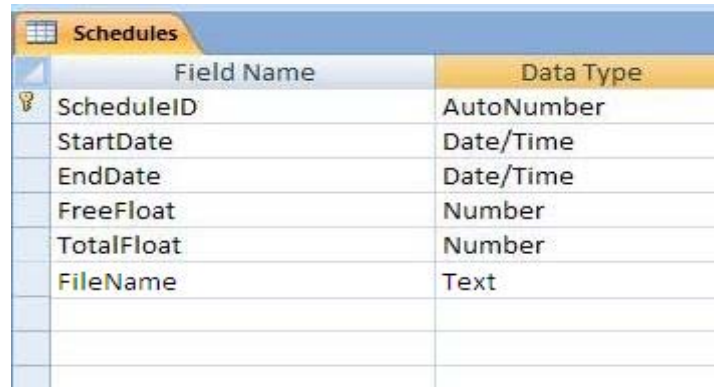
Table C.2, *Activity* table, defines all different *Item* construction activities, i.e. *ActivityName* = painting.

Table C.2 – Activity Table

Activity	
Field Name	Data Type
ActivityID	AutoNumber
ActivityName	Text

Table C.3, *Schedules* table, represents available schedule information about all building objects. This includes information on *StartDate*, *EndDate*, *FreeFloat*, *TotalFloat*, and a schedule file.

Table C.3 – Schedules Table



Field Name	Data Type
ScheduleID	AutoNumber
StartDate	Date/Time
EndDate	Date/Time
FreeFloat	Number
TotalFloat	Number
FileName	Text

The respective *Schedules* class is shown below:

```
public class Schedules
{
    private long scheduleID;
    private string startDate;
    private string endDate;
    private long freefloat;
    private long totalfloat;
    private string fileName;

    public long ScheduleID
    {
        get { return this.scheduleID; }
        set { this.scheduleID = value; }
    }

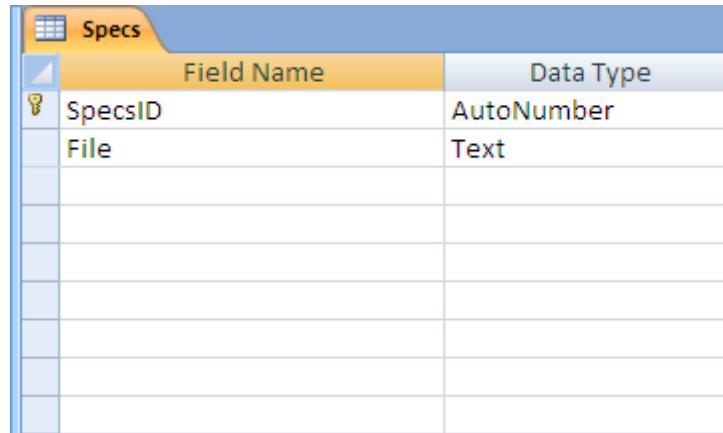
    public string StartDate
    {
        get { return this.startDate; }
        set { this.startDate = value; }
    }

    // . . .

}
```

Table C.4, *Specs* table, consists of many project specification files.

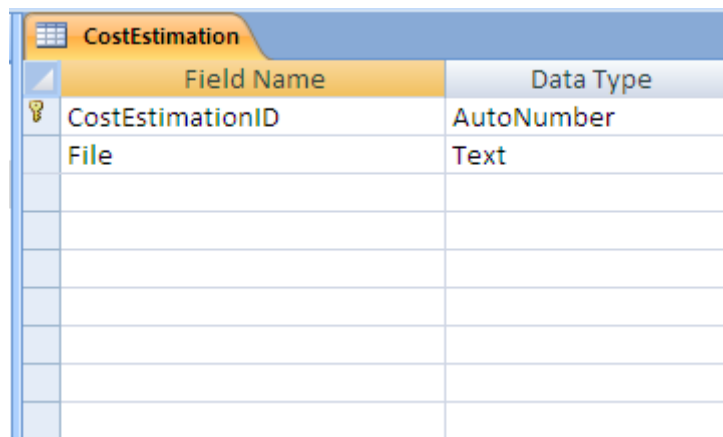
Table C.4 – Specs Table



	Field Name	Data Type
🔑	SpecsID	AutoNumber
	File	Text

Table C.5, *CostEstimation* table, consists of many project cost estimation spreadsheet files.

Table C.5 – CostEstimation Table



	Field Name	Data Type
🔑	CostEstimationID	AutoNumber
	File	Text

Table C.6, *Orders* table, comprises many change orders files and can house other types of files such as faxes, minute papers, etc.

Table C.6 – OrdersTable

Field Name	Data Type
OrderID	AutoNumber
File	Text

Table C.7, *ItemToActivityMapping* table, is the main table of the database. As mentioned in Chapter 7, its role is to map all information from other tables to each item-activity pair. Several one-to-many relationships were thereby created by using the primary key of each table and setting it as foreign key in this table, i.e. *ScheduleID*, *SpecsID*, *CostEstimationID*, and *OrderID*.

Table C.7 – ItemToActivityMapping Table

Field Name	Data Type
ItemToActivityID	AutoNumber
ItemID	Number
ActivityID	Number
ScheduleID	Number
SpecsID	Number
CostEstimationID	Number
OrderID	Number

Besides the schedule, specifications, cost estimation and change orders information, there is also graphical information, building images and CAD drawings. However, since any item-activity pair can have many CAD drawings or pictures, separate tables were created for images and CAD models; *ImgFiles* table (Table C.8) and *ItemCadModels* table (Table C.9).

Table C.8 – ImgFiles Table

ImgFiles	
Field Name	Data Type
FileID	AutoNumber
FileName	Text

Table C.9 – ItemCadModels Table

ItemCadModels	
Field Name	Data Type
CadModelID	AutoNumber
CadModels	Text

In order to map this information to all the textual information residing in the *ItemToActivityMapping* table (Table C.7) for the same item-activity-pair, *ActivityToImgFilesMapping* table (Table C.10) and *ActivityToCadModelsMapping* table

(Table C.11) were created. For example, the table *ActivityToImgFilesMapping* holds the *ItemToActivityMapping* table’s primary key *ItemtoActivityID* as a foreign key and maps to it the *ImgFiles* table’s primary key *FileID*. This way, each item-activity pair has attached to it all textual information as well as several image files. The same applies to CAD drawings.

Table C.10 – ActivityToImgFilesMapping Table

ActivityToImgFilesMapping	
Field Name	Data Type
ActToFileID	AutoNumber
ItemToActivityID	Number
FileID	Number

Table C.11 – ActivityToCadModelsMapping Table

ActivityToCadModelsMapping	
Field Name	Data Type
ActToCadID	AutoNumber
ItemToActivityID	Number
CadModelID	Number

Therefore, based on all the above tables and mappings, the `ItemToActivityMapping` class looks as follows:

```
public class ItemToActivityMapping
{
    private long itemToActivityID;
    private long itemID;
    private long activityID;
    private long scheduleID;
    private long specsID;
    private long costEstimationID;
    private long orderID;
    private string activityName;
    private DataTable imgFiles;
    private DataTable ItemCadModels;
    private DataTable scheduleSet;
    private DataTable costSet;
    private DataTable specsSet;
    private DataTable ordersSet;
    // . . .

    public DataTable ImgFiles
    {
        get { return this.imgFiles; }
        set { this.imgFiles = value; }
    }

    public DataTable ScheduleSet
    {
        get { return this.scheduleSet; }
        set { this.scheduleSet = value; }
    }

    // . . .
}
```

User Interface Design

The information retrieval application user interface, as described in Chapter 7, is illustrated below (Figure C.2):



Figure C.2 – Information Retrieval Application User Interface

A variety of classes to design and manipulate this Windows form were created. Those are called control classes. As depicted in Figure C.2, each of the group boxes (*Object Detected*, *Activities*, *User Info*, *Info Type*, *General Info*, *Schedule Info*, etc.) represents a different control class. For instance, the *ActivitiesInfo* control class consists of a main group box and a drop-down menu box (ComboBox) including all activities for a specific item:

```
public class ActivitiesInfo : System.Windows.Forms.UserControl
{
    public System.Windows.Forms.GroupBox gb_activities;
    public System.Windows.Forms.ComboBox cb_activities;

    // . . .
}
```

On the other hand, the control class *Drawings* mainly consists of a group box grouping a list box, where all available building images are presented, and a picture box where a selected drawing is displayed, i.e. `displayImage(lb.SelectedItem.ToString())`:

```
public class Drawings : System.Windows.Forms.UserControl
{
    public System.Windows.Forms.GroupBox gb_drawings;
    private System.Windows.Forms.ListBox listBox1;
    private System.Windows.Forms.PictureBox pictureBox1;

    // . . .

    private void listBox1_SelectedIndexChanged(object sender,
        System.EventArgs e)
    {
        ListBox lb = (ListBox)sender;
        string fileName = lb.SelectedItem.ToString();
        displayImage(fileName);
    }

    // . . .
}
```

In the case of the main control class *Info Type*, a group box created includes a checked list box containing the different types of information:

```
public class InfoType : System.Windows.Forms.UserControl
{
    private System.Windows.Forms.GroupBox gb_InfoType;
    public System.Windows.Forms.CheckedListBox checkedListBox1;

    // . . .

    public void showRelatedPanels(string clb)
    {
        // . . .

        if(clb.Equals("Schedule"))
        {
            AppData.scheduleInfo.schedulePanel.Visible = true;

            AppData.scheduleInfo.setSchedule(itemToActMapp.ScheduleSet);
            AppData.scheduleInfo.gb_scheduleinfo.BackColor = backColor;
        }

        else
        if(clb.Equals("Cost Estimation"))
        {
```

```

        AppData.costInfo.costPanel.Visible = true;

        AppData.costInfo.setCost(itemToActMapp.CostSet);
        AppData.costInfo.gb_costinfo.BackColor = backColor;
    }

    // . . .

}
}

```

The `showRelatedPanels` function of this class consists of displaying and viewing information on respective interface panels depending on what item of the list is checked, i.e. Schedule, Cost Estimation, CAD Drawings (Figure C.3). This was achieved, as shown in the code above, by calling a controller class method `getItemToActivityMapping` (refer to next section) to get the appropriate information (i.e. `scheduleSet`) about the selected item-activity pair (`itemToActMapp.ScheduleSet`):

```

ItemToActivityMapping itemToActMapp =
AppData.dbController.getItemToActivityMapping(AppData.itemID,
act.ActivityID);
AppData.scheduleInfo.setSchedule(itemToActMapp.ScheduleSet);

```

In this case, `setSchedule()` method of the `ScheduleInfo` control class is used to display the extracted information on the corresponding panel or group box.

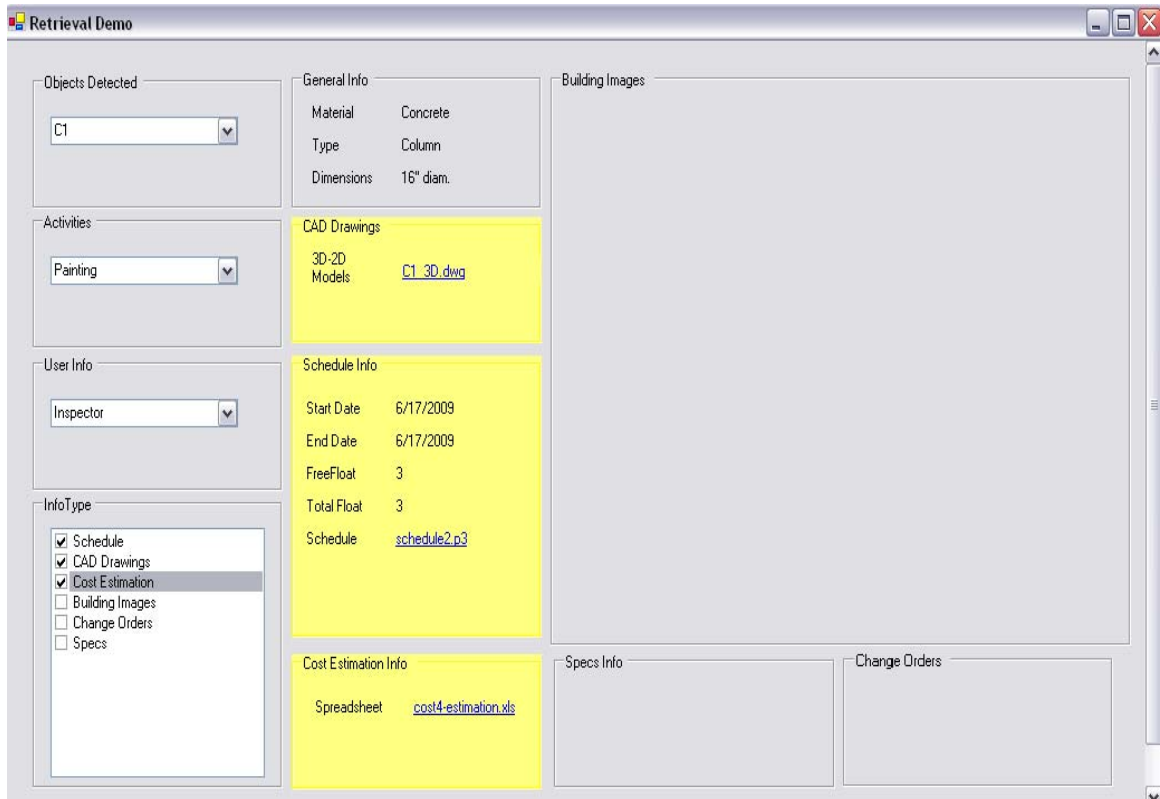


Figure C.3 – Schedule and Cost Estimation Information Retrieval

Database Model - User Interface Communication

As mentioned in Chapter 7, the controller is responsible of the communication of data between the user interface and database model. For this purpose, several controller classes were created, in particular the `DBController` class which makes use of the `ReflectionController` class methods, `setProperty()` and `getObj()`. The code below represents all the methods of the `DBController` class:

```

public DBController()
{
    public Item getItem(string itemName){. . .}
    public DataTable getItemCadModels(long itemID){. . .}
    public string getActivityName(long activityID){. . .}
    public DataTable getActivities(long itemID){. . .}
    public User getUsers(){. . .}
    public string getRanking(string userName){. . .}
    public DataTable getItemActivityOrders(long orderID){. . .}
    public DataTable getItemActivitySpecs(long specsID){. . .}
    public DataTable getItemActivitySchedule(long scheduleID){. . .}
    public DataTable getItemActivityCost(long costID){. . .}
    public DataTable getActivityImgFiles(long itemToActivityID){. . .}
    public Activity getActivityID(string activityName){. . .}
    public DataTable ConvertDataReaderToDataSet(OleDbDataReader
reader){. . .}
    public ItemToActivityMapping getItemToActivityMapping(long
itemID, long activityID){. . .}
}
}

```

An important method is `getItemToActivityMapping`. It consists of:

1) Querying the database to get information on selected item-activity pair:

```

command = new OleDbCommand();
command.Connection = connection;
command.CommandText = "SELECT * FROM ItemToActivityMapping WHERE
ItemID=? AND ActivityID=?";
parameter = new OleDbParameter("ItemID",itemID);
parameter2 = new OleDbParameter("ActivityID",activityID);
command.Parameters.Add(parameter);
command.Parameters.Add(parameter2);
dataReader= command.ExecuteReader();

```

2) Creating a data set using the `ConvertDataReaderToDataSet` method of the same class then automatically getting object properties using the `SetProperty()` and `getObj()` functions of the `ReflectionController` class:

```

DataTable dt = ConvertDataReaderToDataSet(dataReader);
Type tp = typeof(ItemToActivityMapping);
refController = new ReflectionController(tp);

```

```

foreach (DataRow row in dt.Rows)
{
    foreach (DataColumn col in dt.Columns)
    {
        string field = col.ToString();
        string val = row[col].ToString();
        refController.SetProperty(tp.GetProperty(field),val);
    }
}

dataReader.Close();
ItemToActivityMapping itemToActivityMapping =
(ItemToActivityMapping)refController.getObj();

```

3) Get Access to all available information, i.e. activityImgFiles, scheduleSet, costSet, specsSet, and ordersSet using functions of the same class, getActivityImgFiles(), getItemActivitySchedule(), getItemActivityCost(), getItemActivitySpecs(), getItemActivityOrders():

```

DataTable activityImgFiles =
getActivityImgFiles(itemToActivityMapping.ItemToActivityID);
itemToActivityMapping.ImgFiles = activityImgFiles;

DataTable scheduleSet =
getItemActivitySchedule(itemToActivityMapping.ScheduleID);
itemToActivityMapping.ScheduleSet = scheduleSet;

DataTable costSet =
getItemActivityCost(itemToActivityMapping.CostEstimationID);
itemToActivityMapping.CostSet = costSet;

DataTable specsSet =
getItemActivitySpecs(itemToActivityMapping.SpecsID);
itemToActivityMapping.SpecsSet = specsSet;

DataTable ordersSet =
getItemActivityOrders(itemToActivityMapping.OrderID);
itemToActivityMapping.OrdersSet = ordersSet;

```


Appendix D

Biography

Hiam Mayez Khoury was born in Zgharta, Lebanon on January 10, 1981. She earned a Bachelor of Engineering (B.E.) degree in Civil Engineering from the Lebanese American University (LAU), Byblos, Lebanon, in 2004 and a Master of Science (M.S.) degree in Civil and Environmental Engineering (majoring in Construction Engineering and Management) from the University of Michigan, Ann Arbor, MI, in 2005. She pursued her education in Construction Engineering and Management at the University of Michigan, Ann Arbor, MI and earned a Doctor of Philosophy (Ph.D.) degree in Civil and Environmental Engineering in 2009.