12-2003

Query-by-Pointing: Algorithms and Pointing Error Compensation

Farhan Faisal

QUERY-BY-POINTING:

ALGORITHMS AND POINTING ERROR COMPENSATION

By

Farhan Faisal

B.E. R.E.C Silchar, India, 1999

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(in Spatial Information Science and Engineering)

The Graduate School

The University of Maine

December, 2003

Advisory Committee:

Max J. Egenhofer, Professor in Spatial Information Science and Engineering, Advisor

Anthony Stefanidis, Assistant Professor in Spatial Information Science and Engineering

Silvia Nittel, Assistant Professor in Spatial Information Science and Engineering

Lars Kulik, Postdoctoral Research Associate, National Center for Geographic Information and Analysis

QUERY-BY-POINTING:

ALGORITHMS AND POINTING ERROR COMPENSATION

By Farhan Faisal

Thesis Advisor: Dr. Max J. Egenhofer

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Master of Science
(in Spatial Information Science and Engineering)
December, 2003

People typically communicate by pointing, talking, sketching, writing, and typing. Pointing can be used to visualize or exchange information about an object when there is no other mutually understood way of communication. Despite its proven expressiveness, however, it has not yet become a frequently used modality to interact with computer systems. With the rapid move towards the adoption of mobile technologies, geographic information systems (GISs) have a particular need for advanced forms of interaction that enable users to query the geographic world directly. To enable pointing-based query system on a handheld device, a number of fundamental technical challenges have to be overcome. For such a system to materialize we need models stored in the device's knowledge base that can be used as surrogate of real world objects. These computations, however, assume that (1) the pointing direction matches with the line-of-sight and (2) the observations about location and direction are precise enough so that a computational model will determine the same object as what the user points at. Both assumptions are not true. This thesis, therefore, develops an efficient error compensation model to reduce the discrepancy between the line-of-sight of the eye and the pointer direction. The model is

based on a coordinate system centered at the neck and distances measured from neck to eye, neck to shoulder, shoulder to handheld pointer, and the pointing direction. An experiment was conducted using a gyro-enhanced sensor and three subjects who pointed at marked targets in a given room. It showed that the error compensation algorithm significantly reduces errors in pointing with arms outstretched.

ACKNOWLEDGMENTS

I would like to express my sincere thanks and gratitude towards my thesis advisor Dr. Max J. Egenhofer for his encouragement, support, guidance, and patience, and the members of my thesis advisory committee, Dr. Silvia Nittel, Dr. Lars Kulik, and Dr. Anthony Stefanidis, for their support and guidance.

I would like to specially thank Dr. Hans for his willingness to discussions, which helped me in clarifying many doubts. I thank him for his support and guidance.

Thanks to all my colleagues in the SIE department, especially Vibhav, Rui Zhang, Marcus, Akshata, Sabeshan, and David for their numerous stimulating discussions. This research would not have been possible without the huge personal and academic support from Mehwish and I thank her for sharing every moment of this thesis with me.

The work was partially supported by the National Imagery and Mapping Agency under grant number NMA201-01-1-2003.

Finally, I thank all my family members, my parents, sister, and brother-in-law for their love and support through all the years and encouraging me in all my endeavors.

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....ii

LIST OF TABLES.....vi

LIST OF FIGURES.....vii

CHAPTER 1 INTRODUCTION 1

1.1 Example of Query-By-Pointing 2

1.2 Background of Thesis 4

1.2.1 Vision.....6

1.2.2 Pointing 7

1.2.3 Pointing as an Alternative Modality for GIS 8

1.3 A Pointer-Based System for Querying Geographic Information..... 8

1.3.1 Problem Statement 9

1.3.2 Goal.....11

1.3.3 Research Question and Hypothesis.....11

1.3.4 Scope of Thesis 13

1.4 Approach..... 14

1.5 Major Results 15

1.6 Intended Audience 16

1.7 Organization of Thesis 16

CHAPTER 2 DIGITAL TERRAIN MODELS 18

2.1 Modeling Terrain 19

2.1.1 Data Structures.....20

2.1.2 Terrain Models Based on Triangulation 22

2.2 Construction of a TIN 23

2.2.1 An Algorithm for Computing a Delaunay Triangulation..... 25

2.2.2 Computational Complexity.....26

2.2.3 Data Structure for TIN 27

2.3 Summary 28

CHAPTER 3 LINE-OF-SIGHT COMPUTATIONS 29

3.1 Line-of-Sight Computations on TINs 30

3.2 Theoretical Background 31

3.2.1 Digital Models of a Terrain 31

3.2.2 Common Terminology 32

3.2.3 Point Visibility 33

3.3 Existing Algorithm for Point Visibility 34

3.3.1 Horizons on a Two-Dimensional Terrain Model 34

3.3.2 A Divide-and-Conquer Approach 38

3.3.3 A Brute-Force Approach 40

3.3.4 Faster Approach 41

3.4 Prototype For Line-of-Sight Computations 44

3.4.1 User Interface 45

3.4.1.1 Model Formulation Mechanism 45

3.4.1.2 Query-By-Pointing Mechanism 46

3.4.2 Guided Tour 47

3.5 Summary 49

CHAPTER 4 POINTING ERRORS 50

4.1 Errors in Pointing 50

4.2 Problem Formulation and Analysis 51

4.3 Human Figure Model 52

4.4 Pointing Error Compensation 53

4.5 Determination of Correction Factor 55

4.6 Summary 59

CHAPTER 5 ALGORITHM ASSESSMENT 60

5.1 Experimental Setup 60

5.2 Observations 61

5.3 Accuracy of Orientation Sensors when Pointing 63

5.4 Pitch Deviations from Different Pointing Styles 68
5.5 Accuracy Improvement After Computational Error Compensation 74
5.6 Summary 77

CHAPTER 6 CONCLUSIONS AND FUTURE WORK 78
6.1 Summary 78
6.1.1 Algorithms 78
6.1.2 Pointing Error Compensation Method 79
6.1.3 Evaluation of Pointing Error Compensation Method 79
6.2 Conclusions 80
6.3 Future Work 82
6.3.1 Alternate Pointing Methods 82
6.3.2 Alternate Ways of Interacting at a Distance 83
6.3.3 Pointing in Augmented Reality 83
6.3.4 Efficient Surface Approximation Models 84
6.3.5 Mobile Environment 84
6.3.6 Blue Eyed Vision of the Future 85
BIBLIOGRAPHY 86
BIOGRAPHY OF THE AUTHOR 92

LIST OF TABLES

Table 5.1 : Measurements for three different subjects (D, F, C) with center of the neck as the origin.....62

Table 5.2 : True pitch values based on observer and target location (RC).....62

Table 5.3 : Pitch deviation for all subjects for target points (a) 1-5 (TP vs. RC),.....64

(b) 6-9 (TP vs. RC).....65

Table 5.4 : Standard deviations based on the mean value of differences between TP and RC by (1) per subject per target, and (2) per target.....66

Table 5.5 : Deviations of pitch values for all subjects for target points (a) 1-5 (AOS vs. TP),.....70

(b) 6-9 (AOS vs. TP).....71

Table 5.6 : Standard deviations based on the mean value of differences between AOS and TP by (1) per subject per target, and (2) per target.....72

Table 5.7 : Standard deviation of AOS, CP, TP w.r.t true pitch values (RC).....75

LIST OF FIGURES

Figure 1.1 : Example showing a user pointing at a building 3

Figure 1.2 : User pointing at a building 10

Figure 2.1 : The two most commonly used data structures for DTMs: (a) a rectangular grid and (b) a triangular irregular network(TIN).....22

Figure 2.2 : Projection of a TIN.....23

Figure 2.3 : Two triangulations: (a) a Delaunay traingulation (equiangular triangle) and (b) an arbitrary triangulation (long and thin triangle).....24

Figure 2.4 : Updating a Delaunay triangulation: (a) a Delaunay cavity and (b) the reconnection step.....26

Figure 2.5 : A triangulation and its corresponding representation by (a) quadedge and (b) triangular data structures.....27

Figure 3.1 : Before/Behind Subdivision: a<b<c; d<e; f<g w.r.t to the observer point O.....32

Figure 3.2 : Observer at O pointing to A.....33

Figure 3.3 : Determination of target location from a given observation point O.....34

Figure 3.4 : Horizon of an observer point O on a terrain, projected on the X-Y plane in a given direction.....36

Figure 3.5 : Set of segments in the θ - α plane, obtained by projecting the terrain edges, and the corresponding envelope (de Floriani and Magillo 1994).....38

Figure 3.6 : (a) Sweep line status (e8, e7, e6, e5) and (b) sweep line status.....39

Figure 3.7 : Processing and event eduring Atallah's (1989)merging procedure with (a) an intersection point; (b) a right end point, and (c and d) left end points.....40

Figure 3.8 : Visual ray r hits the point P on the terrain.....42

Figure 3.9 : Processing a ray-shooting query on the horizon tree (de Floriani and Magillo 1994).....43

Figure 3.10: Process flow in a TIN model.....45

Figure 3.11: Displaying TIN models.....46

Figure 3.12: Slider representing pointing direction (θ,α).....47

Figure 3.13: Ray-Shooting-Query from an observer location.....48
Figure 3.14: Corresponding details of the observer and target points.....49
Figure 4.1 : Articulated model of the human figure with degrees of freedom (dofs).....52
Figure 4.2 : Pointing error between the eye and the arm.....53
Figure 4.3 : User's coordinate system centered at the neck.....55
Figure 4.4 : Coordinate system centered at the neck N.....56
Figure 5.1 : Subject pointing at target points.....61
Figure 5.2 : Standard deviations per target for TP vs. RC.....67
Figure 5.3 : Standard deviations of points based on elevations TP vs. RC.....68
Figure 5.4 : Standard deviations per target for AOS vs. TP.....73
Figure 5.5 : Standard deviations of points based on elevations AOS vs. TP.....74
Figure 5.6 : Standard deviations from RC for pitch values based on arms outstretched (AOS), corrected pitch values (CP), and telescope pointing (TP).....75

Chapter 1
INTRODUCTION

People typically communicate by pointing, talking, sketching, writing, and typing. This sequence is consistent with the temporal order in which people learn these modalities in their childhood (Owens 1996). Pointing is, therefore, one of the most primitive yet most expressive ways of confronting the unknown. It can be used to visualize or exchange information about an object when there is no other mutually understood way of communication. Despite its proven expressiveness, however, it has not yet become a frequently used modality to interact with computer systems, which rely primarily on typing, sometimes talking, and in few cases on sketching and handwriting (Collins 1988). With a rapid move towards the adoption of mobile technologies, geographic information systems (GISs) have a particular need for advanced forms of interaction that enable users to query the geographic world directly.

Mobile GISs provide the facility to extract spatial information in a dynamic environment. One potential area of application for mobile GIS is to query remote geographic objects by pointing at them using a handheld device (Egenhofer and Kuhn 1998). The objective of this thesis is to evaluate a pointing error compensation method for pointing with a smart pointer in a mobile environment. The foundation of this work is a digital terrain model represented by a triangular irregular network (TIN). Line-of-sight computations on TINs are used to determine the object being pointed at. The location of the user carrying the smart pointer and the two angles representing the pointing direction

of the pointer are the key variables to compute the line-of-sight between the user and the object. A prototype application of a point-and-click interface is implemented to demonstrate the concepts.

1.1 Example of Query-By-Pointing

Imagine this scenario of a user in the year 2005:

Marcus and his friends are touring Boston. They have decided to go to a restaurant located in the downtown area. Once Marcus and his friends get to downtown, they come across a typical large downtown city scene with dozens of restaurants sprawled over a few blocks to choose from. The party walks around, not being able to come to a consensus on a restaurant without having an idea of prices, menus, and seating availability. Marcus has a state-of-art intelligent pointer that combines a global position system (GPS) unit and orientation sensors, yet weighs little more than a cell phone. The pointer includes a detailed surface model of the downtown area and is based on real-time GPS location technology.

Marcus points to a restaurant while activating the pointing device. This device captures the location of the user and the direction in which the user is pointing. The GPS receiver gives an accurate location of the user. The pointing direction coupled with Marcus's location is used to identify the restaurant by computing the line-of-sight between the user and the pointed object. The mobile device with the help of wireless communication technology is linked to the web address for a given restaurant to access menus, rates, and seat availability for Marcus and his friends. He then chooses a restaurant to which all his friends agree.

Figure 1.1: Example showing a user pointing at a building

1.2 Background of Thesis

In computer science the term *interaction* is frequently used to describe the communication between a user and a computer and is referred to as human-computer interaction (HCI) (Helander 1988). HCI is concerned with the joint performance of tasks by humans and machines; the structure of communication between humans and machines; human capabilities to use machines (including the learnability of interfaces); algorithms and programming of the interface itself; engineering concerns that arise in designing and building interfaces; the process of specification, design, and implementation of interfaces; and design trade-offs. HCI thus comprises science, engineering, and design aspects (Hewett *et al.* 1996). Improvements in HCI have led to an enhanced usability and a broad acceptance of computers in everyday life. Today's user interaction involves primarily typing with a keyboard and selecting or drawing with a pointing device, such as mouse or trackball. The future development of HCI is expected to be characterized by new, innovative, and human-centered input devices, which are made possible through the use of portable computers and improved interaction techniques (Shneiderman 1990).

With the advent of wireless technology, the creation and deployment of computing technology is becoming an invisible part of everyday life and commerce (Weiser 1991). The emerging mobile technologies have provided people with the ability to work in novel and previously unanticipated ways. Such developments are at both the level of emerging technological infrastructures for connectivity (e.g., Bluetooth, Sun's Jini, and HP's Jet Send, location pinpointing technologies, 3G and GPRS) and mobile information

appliances (e.g., mobile phones, personal digital assistants (PDAs), and laptop computers). They have the potential of provoking even more radical changes in work practices and encourage an even greater level of mobile work and distributed collaboration (Perry *et al.* 2001).

These advances have provided the basis to expand GIS technology to handheld devices. Mobile GISs can provide access to data anywhere and anytime the user desires. Related operations, such as querying spatial data, performing spatio-temporal analysis or modeling, become possible on the go (Cappelletti 1997). Innovative GIS front ends and interaction techniques have to be developed to make use of the geographic information on mobile handheld devices. Egenhofer and Kuhn (1998) foresee several GIS appliances, such as magic wands and intelligent geospatial pointers, to identify remote geographic objects by pointing at them. These appliances will provide users with opportunities to display and query spatial data anywhere anytime. The mobile handheld device equipped with a global positioning system unit (GPS) can be used to obtain the location of the user carrying a handheld device. Such a device, linked with orientation sensors, captures the direction in which the user is pointing. The position and direction are then matched with a digital terrain model, which is part of the devices knowledge base, to determine the object the user is pointing at.

Such systems must be able to deal with very large spatial data sets. Advanced computational models are needed to integrate different geographic data sources and to respond in real time. Other useful information about the object can be found out about the object with the integration of mobile GIS and the Internet.

1.2.1 Vision

Conventional query languages, such as the SQL, use text-based statements (Egenhofer 1992). They work well within data domains where data can easily be stored in tables, but lack expressiveness and flexibility within more complex domains, such as images, maps, or other spatially related, multi-dimensional data (Egenhofer and Frank 1991). Recent research activities in visual information retrieval systems investigated novel techniques to query spatial data more efficiently (Blaser 2000). With an *"I know it when I see it"* mindset, users feel that there is clear need for a better visual information retrieval system (Gupta *et al.* 1991). Unlike the SQL-based approach, these systems focus more directly on the end result, since an example of a user's query can be used as a formulation of a query statement (Caduff 2003).

As a framework and foundation, we use a pointing based system (Egenhofer and Kuhn 1998) that allows users to formulate a query by pointing at remote geographic objects using a handheld device equipped with a GPS unit and orientation sensors. In order to determine the object being pointed at based on the pointing direction the line-of-sight between the object and the user has to be calculated in the model (for the given terrain) stored in the device's knowledge base. The target point is calculated based on the user location (x, y, z from GPS) and the pointing direction of the handheld device ((θ,α) from the orientation sensors). These computations, however, assume that the line-of-sight of the user holding the device and the direction of the handheld device is the same. This assumption does not necessarily hold true. This work, therefore, develops and evaluates a pointing error compensation method while pointing with arms outstretched and orthogonal to the user's body.

1.2.2 Pointing

People naturally use gestures to communicate. It has been demonstrated that young children can readily learn to communicate with gestures before they learn to talk (Collins 1988) . Pointing is the most natural gesture. It is used where there is no other mutually understood way of communication, such as giving directions to a tourist to a correct address. People point using their hands or fingers; however, it is also possible to point virtually by glancing in a particular direction or at a particular object. Pointing is a modality that focuses on one object or direction at a time. Some people can point by touching the target, while others have to point with their fingers if the target is far away. There can be various problems if a person is pointing at an object in order to help a stranger. The stranger may or may not perceive what the other person is pointing at because of differences in their relative positions and line-of-sight.

Many studies comparing joysticks and other pointing devices exist; however, the domain is typically desktop computing (Gill *et al.* 1991). Non-desktop evaluations, to date, are limited to remote pointing, such as in presentation or home entertainment systems (Westerink 1994). A mouse is a common pointing device, but using it is very different from natural pointing. Pointing is a suitable interaction modality to select visible objects, to initiate process, or to set a focus.

Although pointing is a natural and deeply rooted part of our communication, current interface technology does not take advantage of it. This situation is not a conscious design decision, but results from the evolution of interface technology.

1.2.3 Pointing as an Alternative Modality for GIS

The user interface design process comprises two sets of design decisions, one set determining *what* users can do through the interface and the other determining *how* they can do it (Gellersen 1995). Decisions in the first set are concerned with the types of information that can be exchanged and the operations a user can invoke. The second set of design decisions is concerned with user interface appearance. The appearance is determined by the interaction modalities (spoken, written, gestured) chosen for information exchange and invocation of operations.

Recently, new interaction technologies have become available and affordable, thus enabling new interaction modalities (Thomas *et al.* 1999). This new era of HCI is expected to pave the way for people to use modalities such as pointing. A tourist, for example, might find it easier to point to a building and obtain relevant information about it by using a handheld device rather than trying to ask a stranger who does not understand his or her language. The interaction must be supported by adequate software and hardware technology, leading the user through the possibly complex process of selecting objects by pointing towards them. With recent advances in highly portable mobile computing devices, pointing can become the modality of choice to identify remote geographic objects and to get relevant spatial information about them.

1.3 A Pointer-Based System for Querying Geographic Information

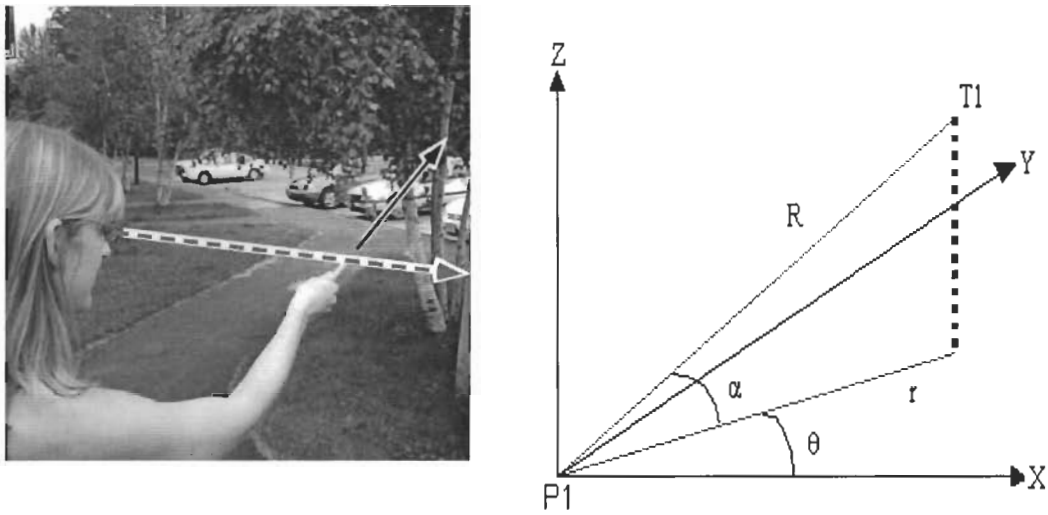
Portable applications are appealing to GIS users. In the future, mobile GIS will extend GIS into the field through wireless communication technology (Harrington 2002). As real time access becomes a reality, mobile GISs will use existing geographic data for more

sophisticated queries and analysis operations. Pointer-based query systems on handheld wireless devices promise to be an appropriate approach to querying an object in an unfamiliar environment. The following sections describe the problem statement, goal, key research questions and hypothesis, and scope of this thesis.

1.3.1 Problem Statement

In order to enable such an innovative human computer interaction, a number of fundamental technical challenges must be overcome. Let P1 be the location of a person holding an intelligent pointer in her hand and T1 the position of the target or object being pointed at by the person in a given terrain (Figure1.1). In order to determine the object being pointed at based on the pointing direction the line-of-sight between the object and the user has to be calculated. Given an observer point P1 and a spherical coordinate system centered at P1, a visual ray R is identified by the pair (θ,α), called a pointing direction, where θ is the horizontal angle, that is, the angle between the projection of ray R on the X-Y plane (r), and the X axis and α is the vertical angle, that is, the angle between a ray R and the projection of ray R on the X-Y plane. With these measurements it is possible to calculate within a terrain model at what object the user is pointing.

These computations, however, assume that (1) the pointing direction matches with the line-of-sight and (2) the observations about location and direction are precise enough so that a computational model will determine the same object as what the user points at. Both assumptions are not true. The thesis, therefore, develops an efficient error computational model to reduce the discrepancy between the line-of-sight of the eye and the pointer direction.



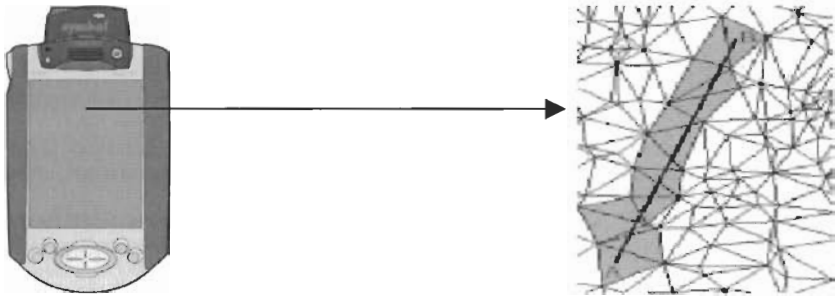


Figure 1.2: User pointing at a building

1.3.2 Goal

The goal of this thesis is to evaluate a pointing error compensation method for pointing with a smart pointer in a mobile environment. The work explores errors in pointing to targets using a handheld device and ways of correcting such errors while pointing with arms outstretched. With the pointing error compensation the measured direction can be transformed such that it then leads to an accurate identification of the object the user pointed to. These calculations rely on line-of-sight algorithms in a digital terrain model for query-by-pointing.

1.3.3 Research Question and Hypothesis

Pointing, to get information about real world objects, promises to enhance interaction in mobile GIS. Many research questions, however, arise about the usefulness of pointing as an alternate modality in mobile GIS and the accuracy of the model to determine the object being pointed at. The following questions are challenging questions that the approach developed in this thesis can answer.

Question 1: *What are the different sources of error while pointing?*

There are lots of different ways a user can point to an object such as with her arms outstretched or by holding the pointer close to the eyes or looking in a different direction while pointing. What are the different types of errors in pointing with a handheld device? What are the appropriate strategies to overcome errors in pointing? These questions are relevant because they help us find the best way to identify the targets in a computational model.

Question 2: *How to correct the errors in query-by-pointing?*

A mathematical model with a coordinate system centered at the neck can be used to correct the errors in pointing with arms outstretched. A pilot study was conducted to calibrate the model. Significant improvements in the result were obtained after applying the correction factor to the data obtained while pointing with arms outstretched. Is this an appropriate strategy to overcome errors in pointing? What are the key variables on which the accuracy of the model depends upon?

Question 3: *What are the different challenges involved in query-by-pointing using a smart pointer?*

Based on the assumption that a real world object can be represented on a handheld device by an accurate surface model and reduction in the pointing error, a user can identify remote objects in the real world by pointing at them. What are the different algorithms involved in the line-of-sight computation based on the pointing direction? The determination of the object from a given user location depends on suitability of the line-of-sight algorithms based on the location of the user and the object in a given surface model.

For correct results the direction of the handheld device, measured with the gyroscope, needs to be corrected such that the pointing direction (θ,α) coincides with the line-of-sight of the user. For this purpose pointing error compensation method has been designed which considers the person's body measurements from neck-shoulder joint, neck-center of the eyes, and the arm length. We hypothesize that:

Pointing error compensation method reduces the discrepancy between the angular values of the line-of-sight over the tip of the pointer and the pointer direction while pointing with our arms extended.

1.3.4 Scope of Thesis

Querying remote geographic objects by pointing at them using a handheld device depends on the user location (from GPS) and the pointing direction (from orientation sensors). The line-of-sight computation between the user and the pointed object does not depend on the distance between the user and the pointed object. This work is based on the assumption that an accurate surface model can represent real world objects. We address the different algorithms involved in the line-of-sight computation in a terrain model based on the pointing direction but the focus was not on developing faster and accurate algorithms for surface modeling. An overview of the algorithms is provided (Chapter 2 and 3) for point visibility computations on a digital terrain model. The prototype for line-of-sight computation further demonstrates its suitability in query-by-pointing. To determine what object the user is pointing at, we assumed that pointing direction of the handheld device matches with the line-of-sight of the eye of the user. This assumption however, does not hold true. We develop a method to reduce the angular shift between the line-of-sight computation of the eye over the tip of the pointer and the pointer direction while pointing with arms outstretched. A descriptive model was considered with a coordinate system centered at the user's neck. The distances measured from; neck to eye, neck to shoulder, shoulder to handheld pointer, and the pointing direction are the key

variables required for compensating errors in pointing. The results of the pilot study conducted using a gyro-enhanced sensor further proved the accuracy of the model. The pointing error compensation method focused on pitch correction.

1.4 Approach

This thesis is concerned with evaluation of a pointing error compensation method to reduce pointing errors. For a pointer-based query system to materialize we need a digital model of the real world, line-of-sight algorithms between the user and pointed object, and methods to compensate for errors in pointing. A large percentage of the knowledge inherent in a real world system is dependent upon the spatial association of system components (Gimblett and Ball 1991). Spatial relations do not exist in the world in any meaningful sense. Rather, they exist in minds, to aid in making sense of the world, and in interacting with it (Mark and Frank 1989); therefore, our concern is with human perception of the real world objects and their representation. Models can be used as a surrogate of the real world. The appropriateness of the model is determined by the sufficiency of information it provides about the system. Mathematically we see a situation similar to a homomorphism, which essentially is a mapping that preserves structure (Frank *et al.* 1997).

The first section of this thesis is concerned with modeling the terrain surface and understanding the algorithms involved for computing the line-of-sight between the observer point (user) and the target point (pointed object) based on the pointing direction in a given terrain model. The terrain model is a mesh of interlocking triangles that provides an efficient approximation of the terrain surface. In the second phase the

research focuses on developing a method to reduce the errors in pointing with arms outstretched. A pointing error compensation method was developed to correct the errors in pointing. The results of the pilot study were used to evaluate the pointing error compensation method.

1.5 Major Results

The survey about the pointing behavior of people using a pointer equipped with orientation sensors provided new insights about the errors in pointing. It showed that the pointer should be held close to the user's eyes before pointing to an object. When using it with arms outstretched, however, often big deviations occur between the line-of-sight of the eye and the pointer direction. In order to correct for such errors, we suggested a method based on a coordinate system centered at the neck and distances measured from the neck to eye, neck to shoulder, shoulder to handheld point, and the pointing direction. The results of the experiment conducted using orientation sensors calibrated the model and demonstrated that it reduced such errors in pointing with arms outstretched.

Pointing promises to be an appropriate modality to formulate a spatial query. To enable such a system we need a better understanding of the algorithms involved in query-by-pointing. There are many algorithms available for computing the line-of-sight between the user and the object in a given TIN model; however, many algorithms are of theoretical interest and not of much practical significance because of difficulty in implementation. We explain the algorithms and complexities involved in computing the line-of-sight between the two points. The prototype of line-of-sight computation based on

the pointing direction specified by the user demonstrates the suitability of point visibility algorithms for query-by-pointing.

1.6 Intended Audience

The intended audiences of this research are researchers, developers, and practitioners in all areas of interactive mobile information system. This thesis may be of interest to GIS professionals working in developing new user interfaces for future spatial information technologies. This thesis may also be of interest to researchers concerned with alternative, multi-modal forms of human-computer interaction.

1.7 Organization of Thesis

The remainder of the thesis is organized into five chapters.

Chapter 2 reviews the terrain modeling techniques for the generation of the TIN model. It discusses the algorithm involved in computing the TIN model from a set of sample data points and the complexities involved in it.

Chapter 3 addresses the problem of computing the line-of-sight in a TIN model. The algorithm discussed is important for understanding the concept of query-by-pointing to query remote objects using a handheld device based on the concept of homomorphism. It also describes the design and implementation of a prototype for computing line-of-sight between the user and the target point based on the pointing direction

Chapter 4 provides the background for the approach to error detection, methods for dealing with such errors in the line-of-sight terrain model, and the assessment of the

implication of errors in pointing to a target using a handheld device by a user. We introduce a mathematical correction model for correcting errors in remote pointing using a handheld device with arms outstretched.

Chapter 5 describes the experiment conducted in a room using gyro-enhanced sensors and three participants to evaluate the model, and compare the pitch values after applying the error correction factor with the pitch values based on the user and target location in the given room.

Chapter 6 summarizes the findings of this thesis, draws conclusions, and presents future research directions.

Chapter 2

DIGITAL TERRAIN MODELS

A *terrain* can be described as an extent of ground, region, or territory (Petrie and Kennie 1991). It is part of the earth's surface. The five major features of terrain distinguished in military maps are hills, saddles, valleys, ridges, and depressions (Military 2001). Terrain also comprises of cliffs, overhangs, and other constructed features, such as cuts and fills, which result from the cutting-through of high areas and the filling-in of low areas to form a level bed for a road. *Surface modeling* is a general term to describe the process of representing a physical or artificially created surface by means of a mathematical expression. *Terrain modeling* is a particular category of surface modeling that deals with the specific problems of representing the surface of the Earth.

Digital representations of the terrain are central elements of the mapping process. Unlike in surface modeling, where a unique mathematical expression can often be used to define the feature of interest, it is difficult in terrain modeling to define precisely the structure of the terrain by a single global mathematical function. Nowadays, the modeling techniques are also used to create digital design models of proposed structures, such as roads and buildings. The ability to accurately model a given terrain will help solve line-of-sight computations (Chapter 3) on a given terrain to identify the object the user is pointing at. The following section describes algorithms and data structures used to model a given terrain.

2.1 Modeling Terrain

A digital terrain model (DTM) is a digital representation of a portion of the Earth's surface (Peucker 1977). It is a representation of the continuous surface of the ground by a large number of selected points with known coordinates in an arbitrary coordinate field (Laflamme and Miller 1958). Obtaining a DTM is a three-step process. The first step consists of acquiring three-dimensional coordinates that represent the area to be surveyed. The second step involves division of the terrain surface into simple sub-regions (e.g., triangles). The third step determines a piecewise polynomial function that describes a terrain approximation for each sub-region.

DTMs have been in existence for decades. They have been applied widely in geoscientific applications since the 1950s. The early work of Miller and La Flamme (1958) was concerned specifically with the use of cross-sectional data to define the terrain. Since then, several other terms, such as digital elevation model (DEM), digital height model (DHM), and digital ground model (DGM), have been coined to describe the surface. Although in practice these terms are often presumed to be synonymous, they often refer to quite distinct concepts. DTM is a more complex and all-embracing concept involving not only heights and elevations, but also other geographical elements and natural features, such as rivers, ridges, hills, and mountains (Petrie and Kennie 1991). They have become major constituents of geographical information processing. In GIS, DTMs provide an opportunity to model, analyze, and display phenomena related to topography or other surfaces (Weibel 1997).

The place of DTMs among real-world applications has been constantly evolving,

adapting to the changing needs of a multi-discipline workplace. Initially, DTMs were developed to work with large-scale mapping projects. As the use of DTMs grew, they became specialized to include a variety of applications, such as landscape architecture and mechanical part modeling.

Many interesting application problems on terrain involve visibility computations. Describing a terrain through visibility information applies to geomorphology, line-of-sight communication problems, and navigation and terrain exploration (de Floriani and Magillo 1993). Problems that can be solved based on visibility consist of determining whether a given object located on a terrain is visible from a viewpoint located on the terrain. The choice of data sources and data structures for modeling a given terrain is critical for solving visibility problems.

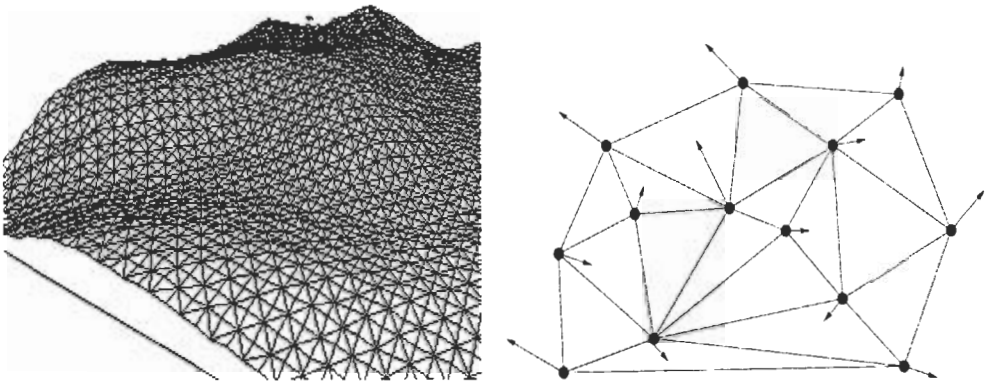
2.1.1 Data Structures

The acquisition of accurate three-dimensional coordinates that represent the surface of the terrain is a vital stage in the process of terrain modeling. It is possible to form such models using a range of different techniques. The particular technique used will depend on factors such as the size of the area to be surveyed, the required accuracy of the data, and the type of information that will eventually be extracted from the model (Petrie and Kennie 1991). The DTM data derived from ground surveys, photogrammetric data capture, or from any other source must be structured to enable handling by subsequent terrain modeling.

There are two common groups of approaches for terrain modeling techniques: (1) using a rectangular grid or (2) using a triangulation. In the rectangular grid approach, the

data comprising the terrain model are measured or collected in the form of a rectangular grid. Grids present a matrix structure that record relations between data points implicitly (Figure 2.1a). The shortcoming of this regular grid-based approach is that the distribution of data points is not related to the characteristics of the terrain itself. If the data-point sampling is conducted on the basis of a regular grid, then the density must be high enough to accurately portray the smallest terrain features present in the area being modeled. If this is done, then the density of the data collected will be too high in most areas of the model, leading to unnecessary and redundant data (Kostli and Sigle 1986).

The triangulation approach is being used increasingly in terrain modeling (Figure 2.1b). This method, referred to as triangular irregular network (TIN), is often used in terrain and surface modeling (Milne 1988). The triangular models represent the ground surface as series of non-overlapping, contiguous triangles with a data point at each node. The heights of additional points can be determined by interpolation. The triangulated method, therefore, overcomes the grid-based difficulties, providing a much more efficient method for representing surface terrain. With the use of TINs it is also possible to accurately define irregularities, such as sharp ridges and embankments.



(a) (b)

Figure 2.1: The two most commonly used data structures for DTMs: (a) a rectangular grid and (b) a triangulated irregular network (TIN).

2.1.2 TIN Models Based on Triangulation

A terrain model is represented by a set of vertices *v*, a set of edges *e*, and a set of triangular faces *f* (Figure 2.2). The three-dimensional coordinates of the original data points are assigned to the vertices. Each edge connects two vertices and is the intersection of exactly two faces. The terrain is approximated by the polyhedron consisting of the triangles. Inside the triangles, the surface is assumed to be planar.

The major drawback of terrain models based on triangulation is the irregular shape of triangles generated from a single set of randomly located measured data points. Also, a triangulation often took an exorbitant time to execute in a computer (Petrie and Kennie 1991). These problems have been overcome, using either the Delaunay triangulation method (Delaunay 1934) or the radial sweep algorithm (Mirante and Weingarten 1982) to generate triangles from a given dataset.

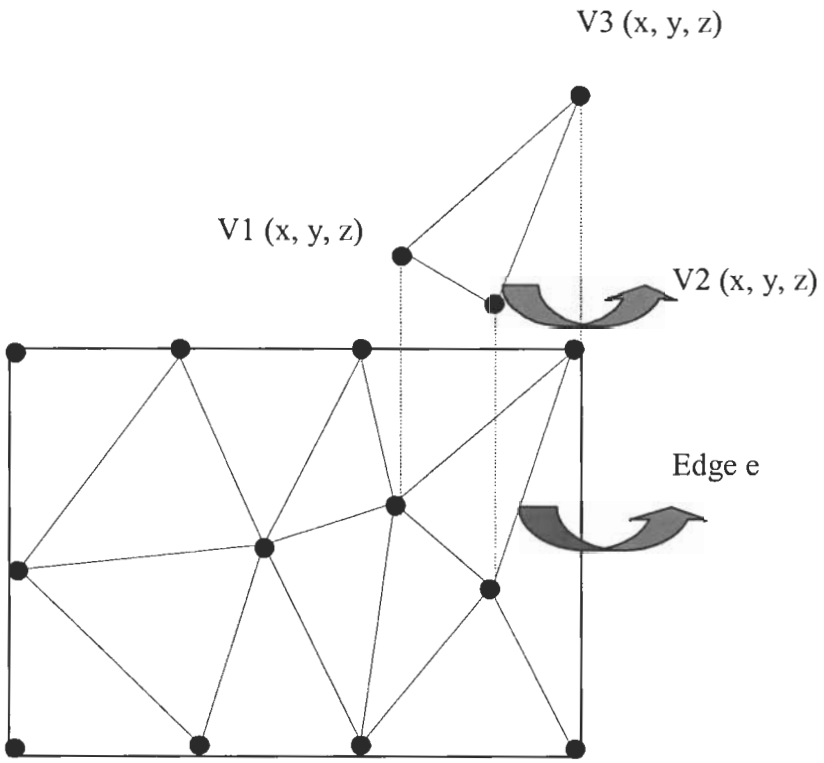


Figure 2.2: *Projection of a TIN.*

2.2 Construction of a TIN

The radial sweep algorithm was devised by Mirante and Weingarten (1982). The input data are in the form of randomly located points with x, y, and z coordinates. The point nearest to the centroid of the dataset is selected as the starting point for the triangulation. From this central point the distances and bearings to all other points in the dataset are calculated and the points are ordered by bearing. The radiating line (i.e., plane swept by a radius from a center point) to each point is established, and a long thin triangle is formed by connecting a line between the new point and the previous point. Although this algorithm results in non-overlapping triangles, the shapes and connections between triangles are undesirable. For many applications, a good triangulation is one without thin

or elongated triangles (McCullogh 1983); therefore, the other most common optimization criterion for triangulation, the Delaunay criterion (1934), is often used for modeling the terrain. The Delaunay triangle has a unique vertex and no other vertex within the structure lies within the circle centered at this vertex. This method has several advantages over other triangulation methods:

- The triangulation is independent of the order in which the points are processed.
- The triangles are as equi-angular as possible, thus reducing potential numerical precision problems created by long skinny triangles (Figure 2.3).
- It ensures that any point on the surface is as close as possible to a node.

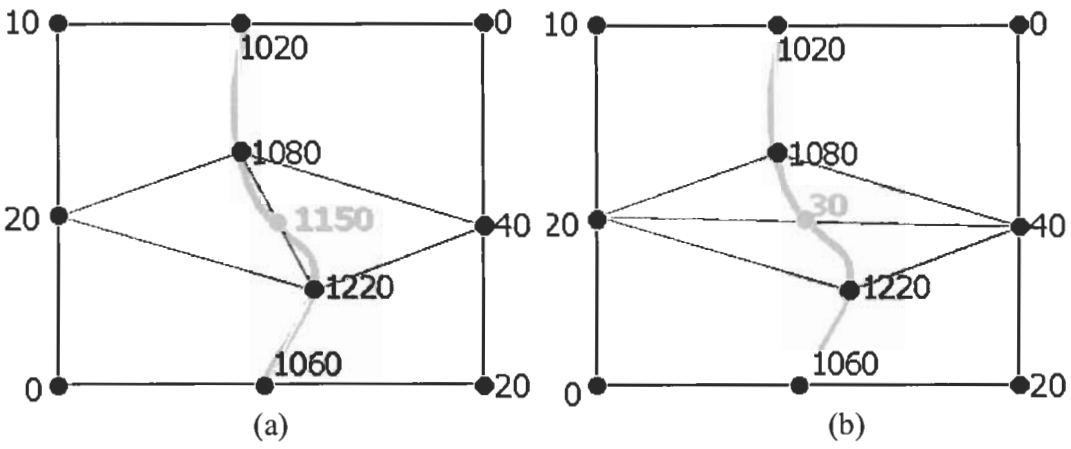


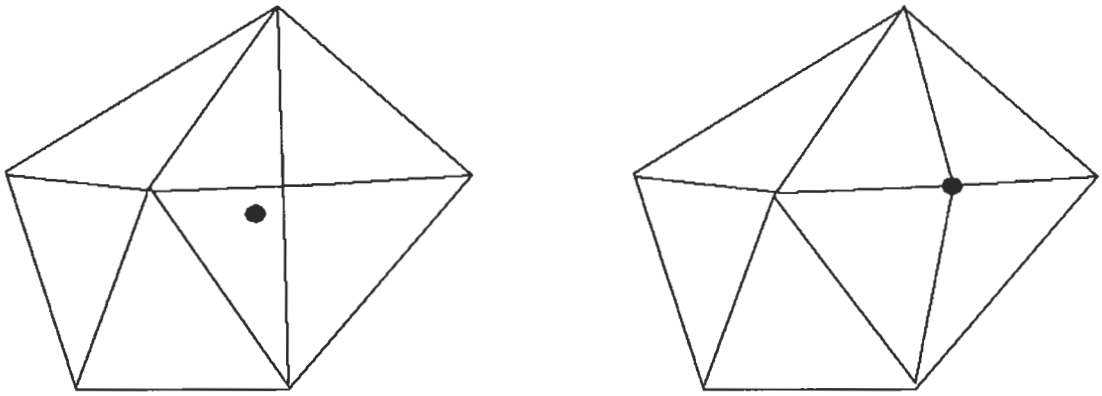
Figure 2.3: Two triangulations: (a) a Delaunay triangulation (equiangular triangle) and (b) an arbitrary triangulation (long and thin triangle).

2.2.1 An Algorithm for Computing a Delaunay Triangulation

There are many Dealunay triangulation algorithms, some of which have been surveyed and evaluated by Fortune (Fortune 1992). Several optimal-time algorithms for Delaunay triangulations have been proposed in the literature. The divide-and-conquer algorithm (Guibas and Stolfi 1985) and the sweep-line algorithm (Fortune 1987), achieve optimal time complexity. Alternatively, a family of incremental algorithms has been used in practice because of their simplicity and robustness.

The *Bowyer Watson algorithm* is often used for building a Delaunay Triangulation for its simplicity. This is an incremental algorithm, meaning that points are added one at a time into an existing triangulation and, although described for two dimensions, can be easily extended to three or more. When a new point is inserted anywhere within the bounding box the topology around the inserted point is updated. All triangles whose circumcircles contain the inserted point are removed and the resulting cavity is triangulated by linking the inserted point to all vertices of the cavity boundary (Figure 2.4).

For an efficient implementation of the Bowyer-Watson algorithm, the information about coordinates of the original points, the neighboring elements, and the center of the n-dimensional circumsphere are permanently stored and updated during processing for each geometrical entity (i.e., point, edges, and triangle). This simple linking scheme automatically guarantees the Delaunay property of the new elements.



(a) (b)

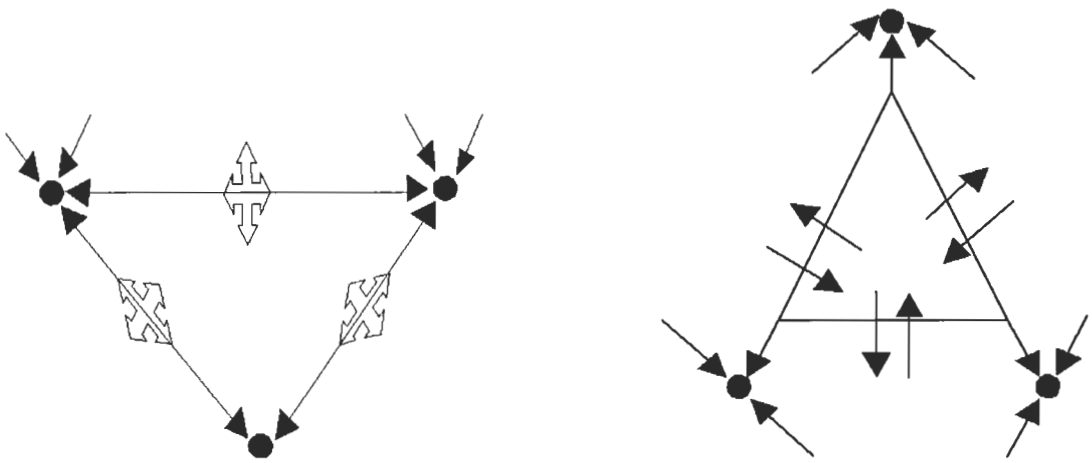
Figure 2.4: Updating a Delaunay triangulation: (a) a Delaunay cavity and (b) the reconnection step.

2.2.2 Computational Complexity

The efficiency of an algorithm is measured by its time and storage complexity. Assuming a computational model time refers to the number of steps, as a function of number of points N, needed to complete the computation. Storage refers to the amount of storage space needed and is also measured as a function of n. To define the complexity of the algorithm used one need to highlight, what components are dependent on the number of points and, therefore, play an important part in the complexity equation. The time T needed to generate a triangulation for N points is $O(N^2)$. The optimal time is O (n log n). To improve the time complexity of the algorithm it is then necessary to implement a data structure that allows an efficient search of the first element to be deleted independently from the way the points are sorted.

2.2.3 Data Structure for TIN

Data structures are a way to organize data in a computer. Common examples of data structures are arrays, stacks, priority queues, and trees (Weiss 1995). Several data structures have been proposed for efficiently representing geometric models as a basic set of functionalities, such as Boolean operations and point location (de Berg 1997). There are two popular ways of storing TINs: one is based on triangles and the other based on points and their neighbors. The two structures are simplified versions of data structures that can store arbitrary planar subdivisions, such as doubly connected edge list or quad-edge structures (Figure 2.5) commonly used in GIS and computational geometry (de Berg 1997; Dobkin and Tal 1995).



(a) (b)

Figure 2.5: A triangulation and its corresponding representation by (a) quadedge and (b) triangular data structures.

2.3 Summary

Digital terrain models (DTMs) are a major constituent of geographic information processing. One of the common digital terrain models is a triangulated irregular network (TIN). This chapter explored the individual elements of DTM techniques for the generation of TINs. We also discussed ways to represent a TIN in a data structure and reviewed an algorithm to construct a Delaunay Triangulation. Building a triangulation for a set of points can be seen as an efficient way of defining the relationship for geometric data sets. The particular method for building triangulations is considered optimal in regard to the shape of the elements generated, because the triangle shapes are as close as possible to those of regular triangles. This is an important property in spatial interpolation and visualization. Despite the fact that the Bowyer-Watson algorithm is not the fastest available, it has been preferred for simple algorithm design and ease of integration. This chapter reviewed the method to generate a TIN model that will be used for demonstrating the query-by-pointing concept in a given terrain in the next chapter.

Chapter 3

LINE-OF-SIGHT COMPUTATIONS

Visibility problems on terrain are concerned with the computation of visibility information from a viewpoint, which can lie outside or inside the domain, or solving optimization problems with the use of visibility information. Examples of optimization problems related to visibility are finding the minimum number of towers of a given height necessary to view an area of the terrain or determining intervisibility between two points located on the terrain (de Floriani *et al.* 2003). Line-of-sight computation problems consist of finding a visibility network connecting two or more sites such that every two consecutive nodes of the network are mutually visible. Applications include the location of fire towers, radar sites, television or telephone transmitters, path planning and navigation (Nagy 1994). With recent advances in computer hardware and mobile GIS it might be possible to identify an object on a terrain by pointing at it using a handheld device in a given TIN model (Egenhofer and Kuhn 1998). Visibility computation algorithms are the basis for solving query-by-pointing using a handheld device. The remainder of this chapter describes in detail an algorithm used for computing the line-of-sight between objects and the user. The design and implementation of a prototype for computing line-of-sight between the user and the target point based on the pointing direction is also discussed.

3.1 Line-of-Sight Computations on TINs

Visibility problems on terrains can be divided into computation of *visibility structures*, which provides information about the portions of the terrain visible or invisible from a given view point, and *visibility queries*, which determine whether a given entity located on the terrain is visible from a given viewpoint (de Floriani and Magillo 1999). The basic visibility structure for a terrain is the viewshed, which is the collection of the surface portions visible from a viewpoint. Another visibility structure is the horizon of a viewpoint, which determines the farthest portion on the terrain that is visible from a viewpoint for every radial direction around it in the X-Y plane.

A visibility query related to a point involves the line-of-sight computation to determine whether it is visible from a viewpoint. It is a fundamental step in identifying objects in a given terrain by pointing at them. Given a terrain elevation model E and two spatial objects represented by two points A and B, this work concentrates on the visibility test between A and B. We want to compute if there is any obstacle on E blocking the visibility between A and B. In this context the visibility test can be expressed as the following spatial query: Is there an intersection between the line segment AB and any of the triangles on E? We need specialized data structures that can query very large amounts of spatial data.

Line-of-sight algorithms must be efficient in identifying the object being pointed at in the terrain model. Visibility problems considered in this work operate on the basis of a point of view located on the terrain.

3.2 Theoretical Background

In this section, some basic notions about Digital Terrain Models, the definition of the point visibility problem, and its existing solutions are introduced. The common terminology and definitions used in the visibility problems on the terrain are also discussed.

3.2.1 Digital Models of a Terrain

A Delaunay triangulation has an important property for visibility computation: it is possible to define a partial order relation (called the before/ behind relation), with respect to any point inside the domain of the subdivision (de Berg *et al.* 1991). Given a planar subdivision $\sum$ and a point O in the plane, an edge e1 of $\sum$ is said to be before an edge e2 (and e2 behind e1) with respect to O if and only if there exist a ray r emanating from O and intersecting first e1 and then e2 (de Floriani and Magillo 1994). Figure 3.1 shows different situations of the before/behind relation.

Given a subdivision with the before/behind relation, visibility information can be incrementally computed for each face or edge by taking into account only the configuration of that portion of the terrain formed by those faces and edges which come before in the order. Visibility problems on a terrain are classified into point, line, and region visibility on the basis of the dimensionality of their output information.

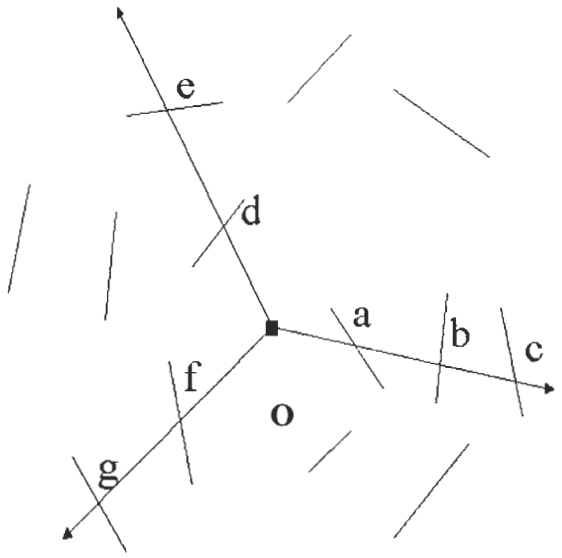


Figure3.1: Before/Behind Subdivision: a<b<c; d<e; f<g w.r.t to the observer point O.

3.2.2 Common Terminology

A candidate point is any point P = (x, y, z) belonging to or above the terrain. Given an observation point (an arbitrary candidate point) O, and a spherical coordinate system centered at O, a visual ray R is identified by the pair (θ,α), called view direction, where θ is the horizontal angle or the angle between the projection r of ray R on the X-Y plane with the X-axis, and α is the vertical angle or angle between ray R and the X-Y plane (Figure 3.2).

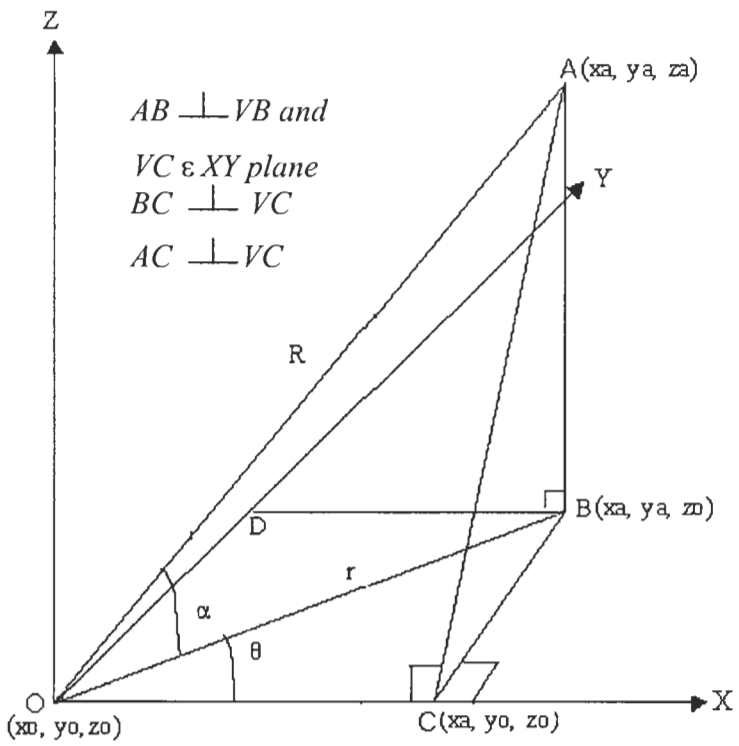


Figure 3.2: Observer at O pointing to A.

Point visibility problems compute the set of points, chosen in a candidate set, visible from a predefined observation point, whereas line visibility problems compute curves on the terrain with special visibility characteristics with respect to an observation point, such as the computation of a horizon.

3.2.3 Point Visibility

Two points are mutually visible when the straight-line segment joining them lies above the terrain and touches them at its two extremes. Figure 3.3 shows how sample terrain data is projected on to the X-Y plane.

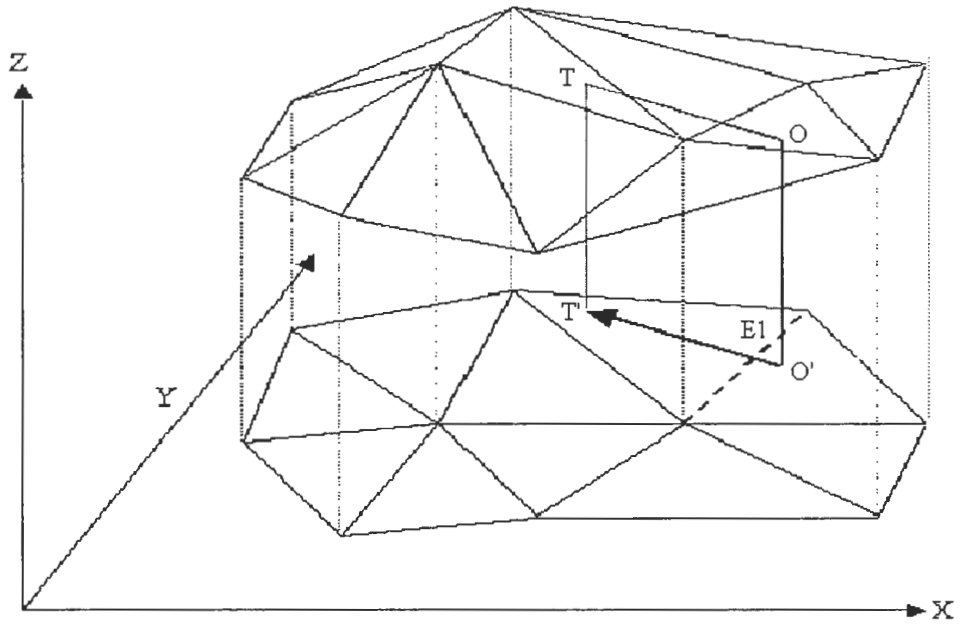


Figure 3.3: Determination of target location from a given observation point O.

3.3 Existing Algorithm for Point Visibility

Many practical algorithms (Boissonnat and Dobrindt 1992; de Floriani 1989; Lee 1991) as algorithms of theoretical interest (Preparata and Shamos 1985), have been proposed for computing point visibility. We first discuss the concept of *horizon,* which serves as a good background for understanding the algorithm. Finally we discuss a "brute-force approach" and a faster approach for computing point visibility in a TIN model.1

3.3.1 Horizons on a Two-Dimensional Terrain Model

The computation of the horizon of an observation point on a terrain is a well-defined problem in computational geometry, and several algorithms have been developed for its

solution (Atallah 1989; de Floriani and Magillo 1993; Hershberger 1989) . Given a terrain M, and a viewpoint $V = (x_0, y_0, z_0)$, the horizon of the terrain with respect to viewpoint V is a function $\beta = h(\theta)$, defined for $\theta \varepsilon [0,2\pi]$, and for every radial direction θ, $h(\theta)$ is the maximum value α such that each ray emanating from V in a view direction (θ,β), with $\beta > \alpha$, does not intersect the terrain (de Floriani and Magillo 1993, 1999). This implies that the horizon of the terrain provides for each radial direction the minimum elevation that must have a visual ray emanating from the viewpoint in the given direction to pass above the surface of the terrain. Figure 3.4 shows an example of a horizon on a two-dimensional terrain model.

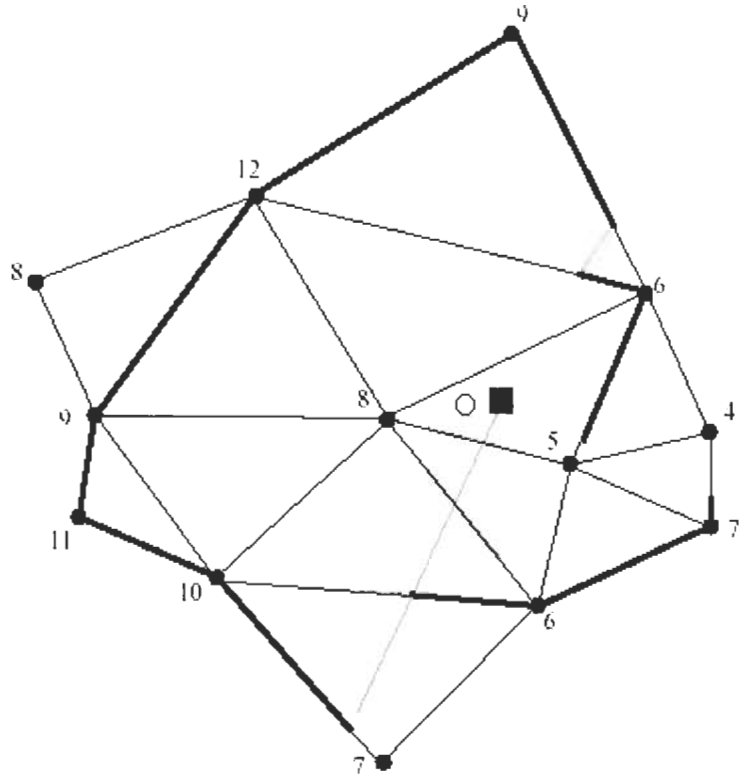


Figure 3.4: Horizon of an observation point O on a terrain, projected on the X-Y plane in a given direction.

From a given observer point, the bearing to all the other points in the data set is first calculated and then the points are sorted and placed in order by bearing. Figure 3.2 shows that from any given two points we can derive the relationship between coordinates and the view direction (θ,α). From Figure 3.2 we have equation 3.1 a-d:

$$tan\ \theta = \frac{BC}{OC} \tag{3.1a}$$

$$tan\ \theta = \frac{ya - yo}{xa - xo} \tag{3.1b}$$

$$tan\ \alpha = \frac{AB}{OB} \tag{3.1c}$$

$$\tan \alpha = \frac{(za - zo)}{\sqrt{(xa - xo)^2 + (ya - yo)^2}} \tag{3.1d}$$

There is a relationship from a given observer point O to all the points on the terrain model. To reduce the horizon on a polyhedral terrain model to the upper set of segment, the edges of the terrain are expressed in a spherical coordinate system, centered at the viewpoint, and with only the two angular coordinates. This transformation produces a set of segments in the θ-α plane. Given a collection of segments in the plane, if the segments can be regarded as opaque barriers, then their upper envelope consists of the portions of the segments visible from a point. The upper envelope maps any x value, in the segment having a maximum angle α value over x (if such segment exists). Figure 3.5 shows an example of a horizon on a polyhedral terrain and the corresponding upper envelope of segments. It has been shown (Edelsbrunner 1989) that the complexity of the upper envelope of p segments in the plane is O(p α (p)) and, therefore, the complexity of the horizon of a polyhedral terrain with n vertices is equal to O(n α (n)). The upper envelope of p segments can be computed by either static divide-and-conquer approach or by a dynamic incremental one. In the next section the divide and conquer algorithm (Atallah 1989) is discussed.

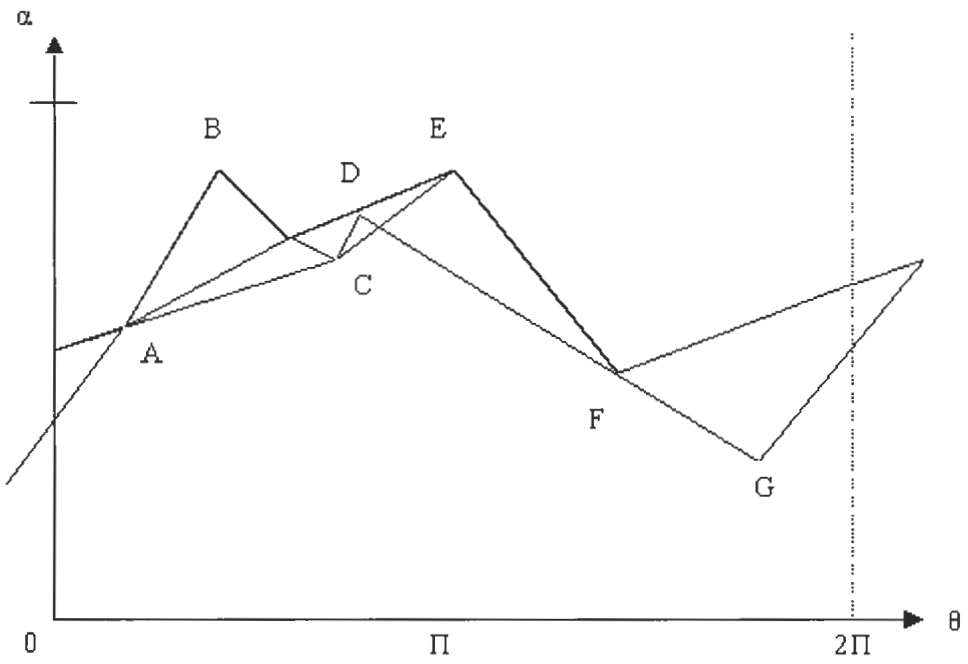
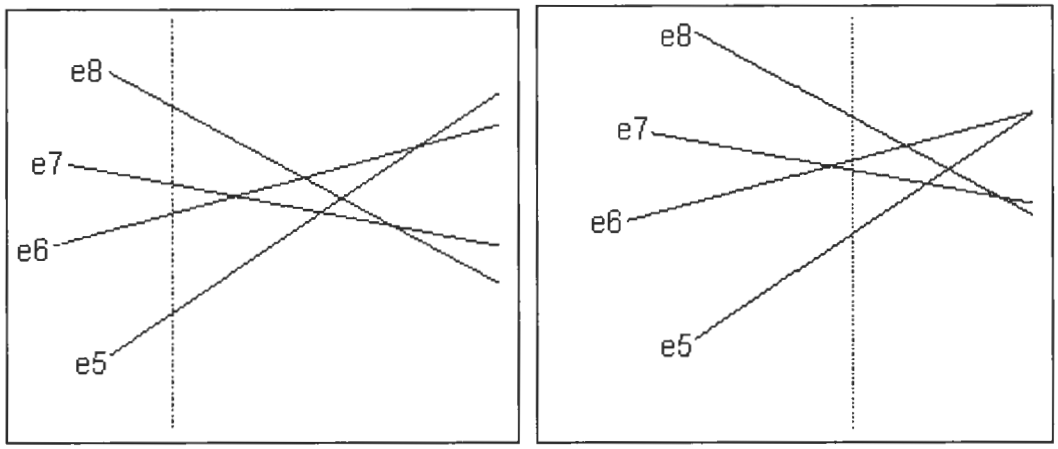


Figure 3.5: Set of segments in the θ - α plane, obtained by projecting the terrain edges, and the corresponding envelope (de Floriani and Magillo 1994).

3.3.2 A Divide-and-Conquer Approach

The algorithm for computing the upper envelope of a set of segments uses a sweep line algorithm for reporting all intersecting segment pairs having a maximum value $\beta = h_v(\theta)$ in the terrain model. The divide part recursively splits the set of segments into two halves. The conquer part merges the results through a sweep line technique for intersecting two chains of segments. The sweep line algorithm moves a vertical line *r* from left to right through the (θ-α) plane and reports all intersecting segment pairs. The events are represented by the vertices of the two given envelopes, and the intersection

points between them. At any event the sweep line status maintain the segments intersecting the sweep line *l* in sorted order, from top to bottom. Figure 3.6 illustrates the sweep line status of a pair of intersecting segments at different intervals.



(a) (b)

Figure 3.6: (a) Sweep line status (e8, e7, e6, e5) and (b) sweep line status (e8, e6, e7, e5).

The *status* of the sweep line is the set of segments intersecting it. The status changes while the sweep line moves to the right, but not continuously. The current status of the sweepline is represented by the pair of segments, one each partial envelope, that are intersected by the sweep line, ordered according to their height. Only at particular points is an update of the status required. These points are called *event points* of the plane sweep algorithm. In this algorithm, the event points are the endpoints of the segments and the intersection points. The event may be an intersection point or a right end point or a left end point (Figure 3.7). By computing the upper envelope of such segments, we obtain a

function that associates with each direction θ the segment having maximum azimuth in direction θ (i.e., the horizon or function $h_v(\theta)$ for $\theta[0, 2\pi]$ from a given view point V).

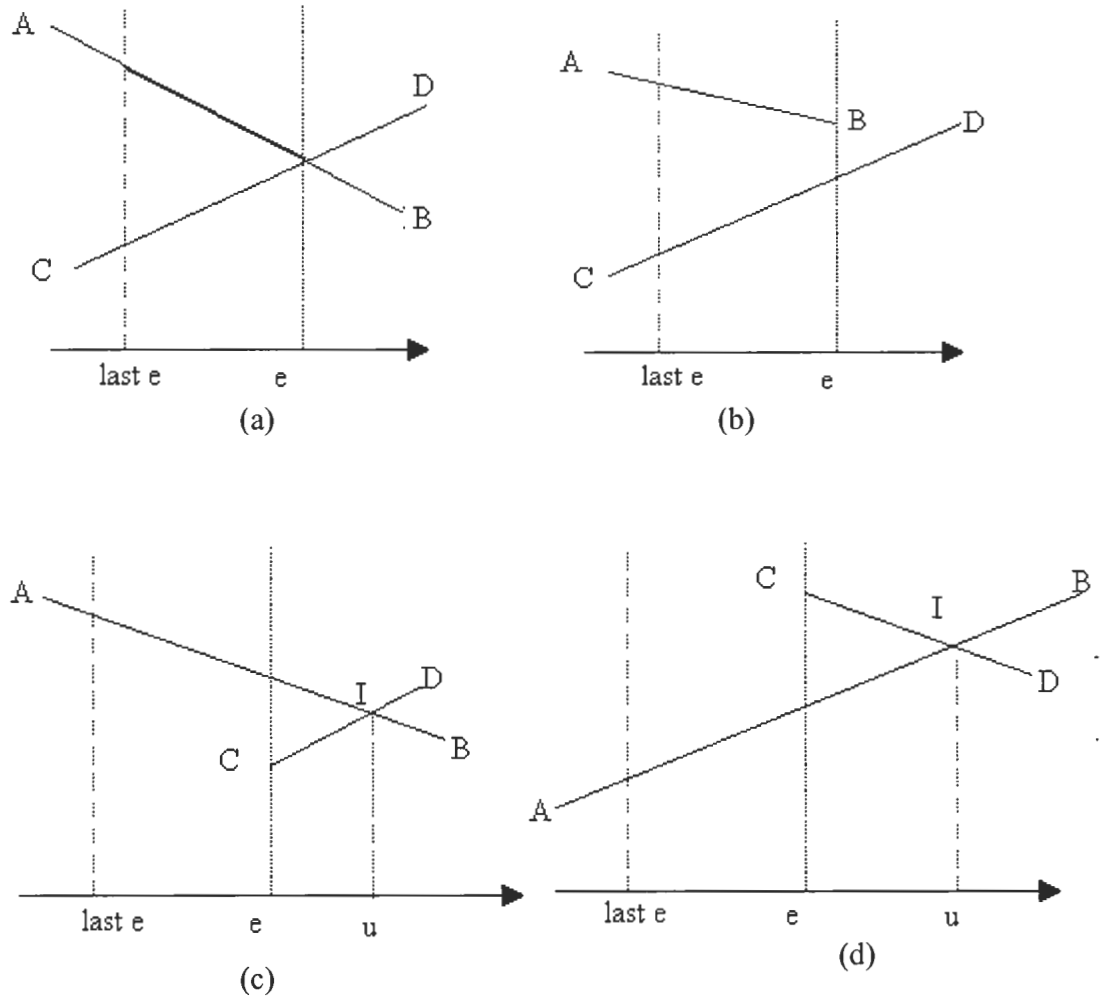


Figure 3.7: Processing an event e during Atallah's (1989) merging procedure with (a) an intersection point; (b) a right end point, and (c and d) left end points.

3.3.3 A Brute-Force Approach

Given a polyhedral terrain model, a point of observer location O, and a view direction (θ,α), the ray shooting problem consists of determining the first face of the polyhedral

terrain model hit by a ray emanating from an observer point O. The mutual visibility of two points observer O and the target point T reduces to computing the intersection of the projection on the X-Y plane of segment s = OT, with the edges of the polyhedral terrain model. At each intersection point P between segment s and an edge e1 of the polyhedral terrain model (PTM) we test whether s lies above the edge of PTM corresponding to e1. If s is above the corresponding terrain edge at any such intersection point, then O and T are mutually visible (Figure 3.3). In general this process has a linear time complexity, in the worst case it is the number of edges of PTM, which is O(n), where n is the number of vertices of the TIN model.

3.3.4 Faster Approach

The second approach preprocesses the terrain model with respect to the observer point O and builds a data structure on which the problem of computing the visibility of a point P from O can be solved in logarithmic time. The data structure has been proposed by Cole and Sharir (1989). The horizon tree data structure (Figure 3.8) is a balanced binary tree in which every node corresponds to a subset of edges and stores a partial horizon. The root corresponds to the whole set of edges of the terrain. Each left child corresponds to the half of the edges, associated to its parent, that are closest to the observer point O, whereas each right child corresponds to the remaining half. Every node of the tree stores the partial horizon (Section 3.3.1) computed on the edges associated with the left child. An example of a horizon tree is illustrated in the Figure 3.4. A horizon tree can be computed in optimal time, since each partial horizon can be computed by a single application of the algorithm of Atallah.

A ray-shooting query, represented by a view direction (θ, α), can be answered by descending the horizon tree, starting at the root. For any node v visited, the value $\beta=h_v(\theta)$, is computed and compared with α. If $\alpha>\beta$, then the visual ray r, identified by (θ,α), passes above the current horizon, and the search continues in the right subtree of v; otherwise r passes below the current horizon, and the search continues in the left subtree. At the end of the process two consecutive horizons are found such that the visual ray passes above the first one, but below the second (de Floriani and Magillo 1994). In descending tree T, one node is visited at each level. For each node v the interval of the horizon associated with v containing ray r must be located. Figure 3.9 illustrates processing of a ray-shooting query on a horizon tree.

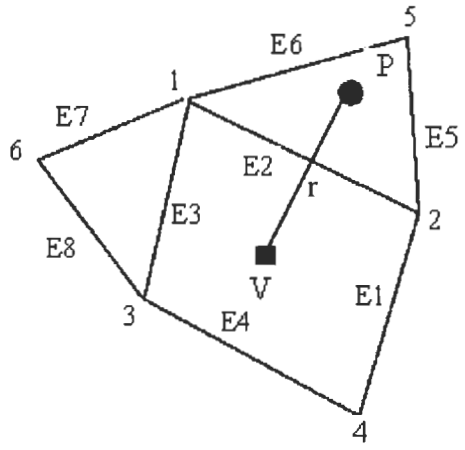


Figure 3.8: Visual ray r hits the point P on the terrain.

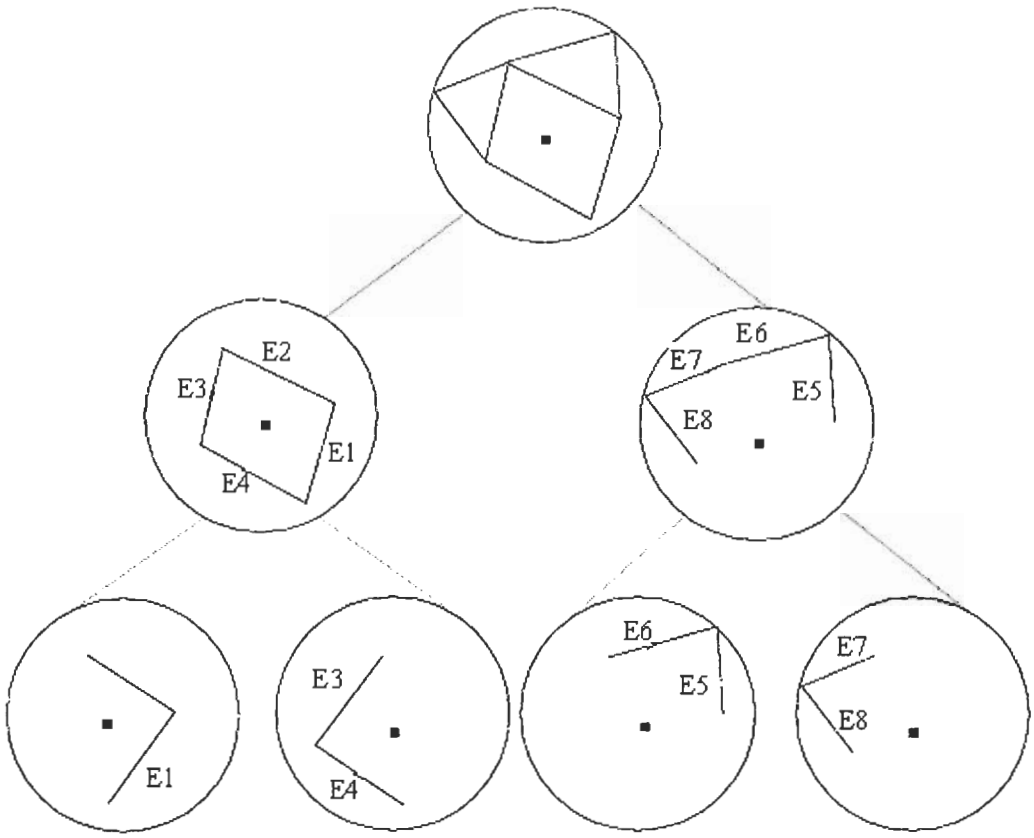


Figure 3.9: Processing a ray-shooting query on the horizon tree (de Floriani and Magillo 1994).

A horizon tree can be computed in optimal $O(n\alpha(n)\log n)$ time, since each partial horizon can be computed by a single application of the algorithm of Atallah (1989) for determining the horizon of a point of view on a TIN model.

3.4 Prototype For Line-of-Sight Computations

The prototype demonstrates the suitability of point visibility algorithms for query-by-pointing. The building block for query-by-pointing is a surface model representing real world objects to be queried. The line-of-sight computation on a TIN model is based on the pointing direction (Figure 3.10). The location of the user (x, y, z) and the pointing direction (θ and α angles) are given as input to the TIN model. The pointing direction and the user location is used to compute the horizon from the given user location (i.e. is the farthest portion on the terrain that is visible from an observer point for every radial direction around it in the X-Y plane). This information is used to preprocess a terrain model into a horizon tree data structure that supports fast responses to ray shooting queries to determine the first point on the model hit by a query ray emerging from a point in a specified direction.

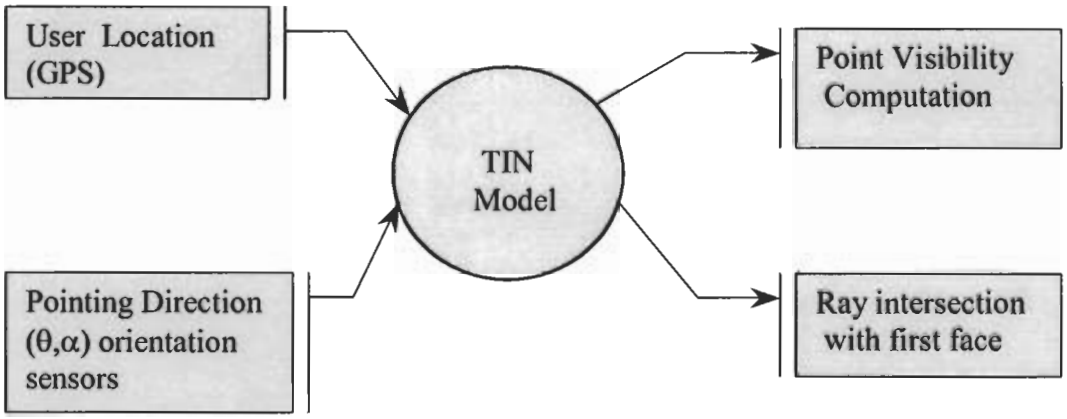


Figure 3.10: Process flow in a TIN model.

3.4.1 User Interface

Input to the prototype, interactive building of a TIN model, and line-of-sight computation on the model are the main features of the user interface. It consists of a text area for displaying the triangulation, a set of buttons used for line-of-sight computation on the TIN model, two slider bars representing the pointing direction, and a separate window displaying the location (x, y, z) of the observation and the target point. Triangulation of sample input points is enabled through mouse clicks in the viewer window. The input of sample data points can also be read from a file and is displayed as a menu item in the application. The user interface implements a model formulation mechanism, and the query-by-pointing mechanism for the results. The following section describes the two mechanisms.

3.4.1.1 Model Formulation Mechanism

The TIN model represents the surface as series of non-overlapping contiguous triangles with a data point at each node of the triangle. The heights of additional points inside the triangle can be determined by spatial interpolation based on height value of the three

vertices of the triangle. The sample input points are passed as mouse click event in the text area (Figure 3.11). As soon as four points are clicked in the text area the application checks the circum circle property and displays the delaunay triangulation in the given text area. The input of sample data points can also be read off from a file by choosing the appropriate menu command.

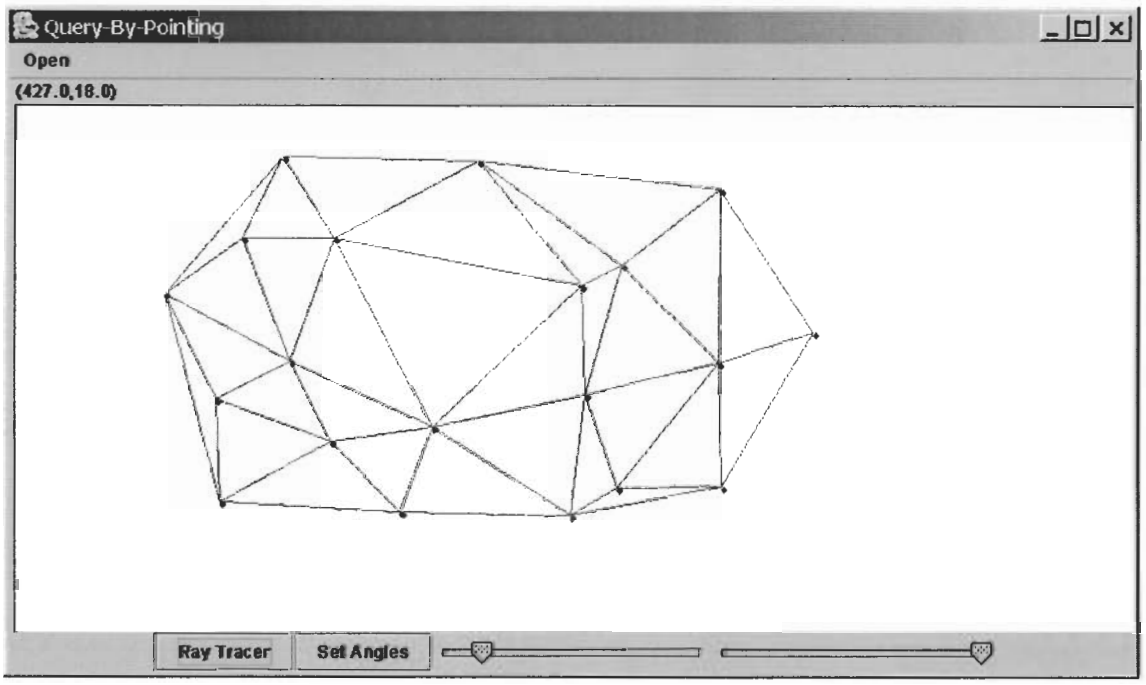


Figure 3.11: Displaying TIN models.

3.4.1.2 Query-By-Pointing Mechanism

The user initiates a query by changing the values of the two-slider bars representing the pointing direction (θ,α) (Figure 3.12). The application can determine the first face of the triangle hit by a ray emanating from an observer point P, based on the value from the two-slider bars. The observer point P can be selected on the TIN model by clicking on the

model after setting the pointing direction. A ray traces a path according to the information provided and hits a face of a triangle in the given model.

Figure 3.12: Slider representing pointing direction (θ, α).

3.4.2 Guided Tour

This guided tour explains the steps involved in a typical pointing query. The task of query-by-pointing consists of three steps: (1) *configuration of the surface model*, (2) *adjusting the pointing direction*, and (3) *presentation of the results*. In the first step, the user configures the TIN model by clicking on the window provided (Figure 3.11). The second step consists of adjusting the pointing direction, that is, the user adjusts the slider bars representing pointing direction. For a handheld device the input would be taken from gyro-enhanced sensors. When the parameters are adjusted a ray emanates from a given observer point and hits the first face of the triangle visible from the given observer location (Figure 3.13). The observer location is represented by the first click on the DTM. This face can further be queried to identify the object being hit by the ray.

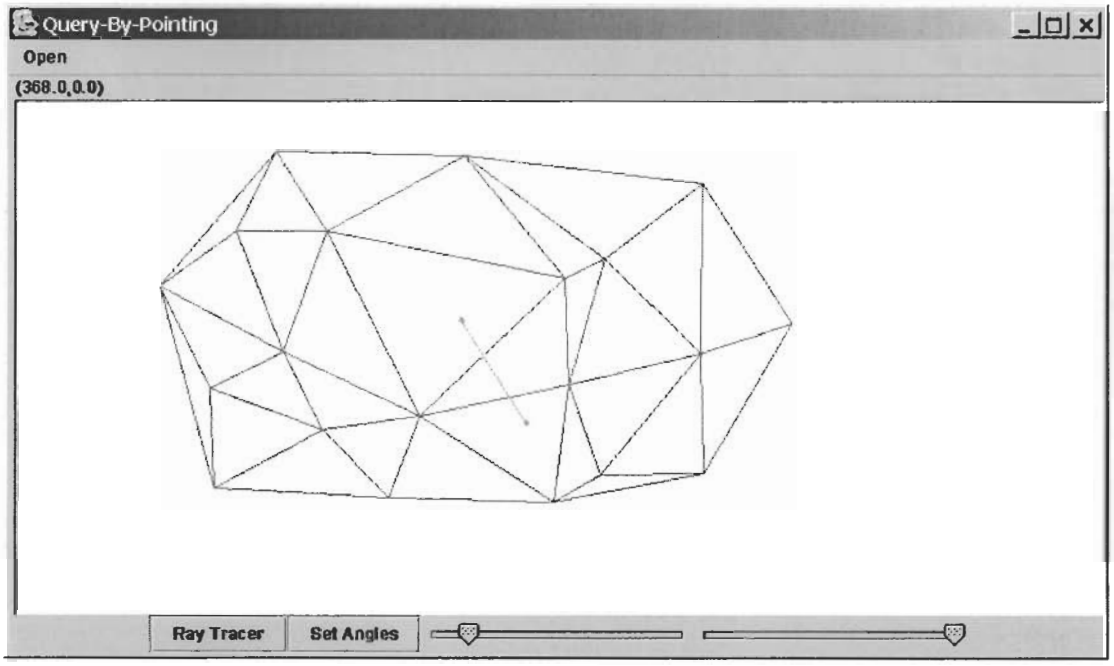


Figure 3.13: Ray-Shooting-Query from an observer location.

The value of the slider bars can be changed and the ray-shooting query can again be performed on the same model. Another TIN model can also be constructed by clicking on the text area provided. The details of the ray shooting query is presented in a separate pop up window that gives information about the location of the observer and the target point located in a given face of the triangle (Figure 3.14). It also gives information about the height of the ray when it crosses any intermediate blocking edges and the height of the point at which the ray crosses a given blocking edge.

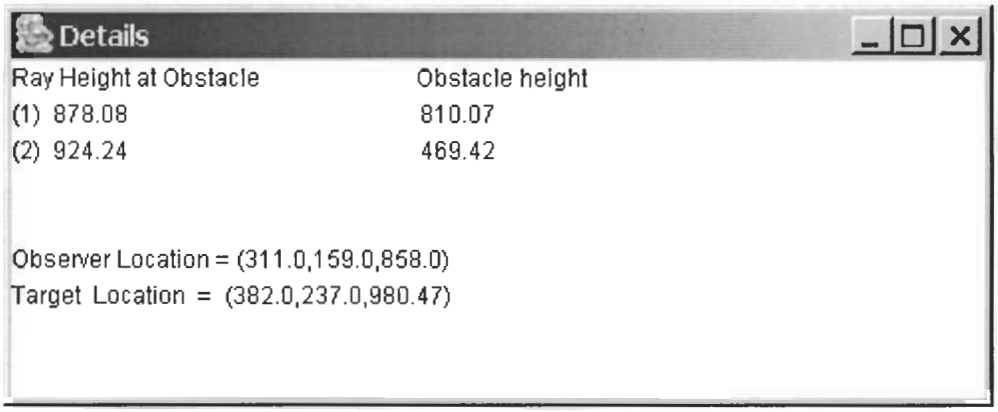


Figure 3.14: Corresponding details of the observer and target points.

3.5 Summary

In this chapter, the problem of computing point visibility on digital terrain models was addressed. The algorithms required for computing point visibility were discussed in detail. These algorithms are important for understanding and computing the line-of-sight between the user and the object at which the user might point using a mobile device. This information is important for identification of the object a person is pointing at. The prototype architecture, design and specification were discussed to understand the data flow and the interaction between the application program and the user.

Chapter 4

POINTING ERRORS

There are various sources of error while using a handheld device (equipped with a GPS unit and orientation sensors) for querying remote geographic objects by pointing at them. There can be errors in the GPS unit, used to provide the location of the user or in recording the exact pointing direction from the orientation sensors. There can also be an error in the line-of-sight of the user and the pointer direction. For query-by-pointing using point visibility algorithms (Chapter 3) we need to reduce the pointing errors to correctly identify a potential target. Several studies have shown that pointing to visual targets without movements of the user is complex by itself. It is well known that people make consistent errors when asked to point to visual targets in space (Soechting and Flanders 1989) .

This chapter develops a method for error compensation while pointing using a handheld device with arms outstretched.

4.1 Errors in Pointing

Pointing and clicking requires good hand-eye coordination, which like any other physical task takes time and practice to learn. Many research and commercial systems have investigated isometric joysticks and other pointing devices, however, the domain is typically desktop computing (Barrett *et al.* 1995). Pointing with a handheld pointing device is a function of the distance and the width of the target (Mackenzie 1992). Several

studies have shown that subjects undershot faraway targets in a systematic way, whereas they sometimes overshot nearby targets (Soechting and Flanders 1989). For query-by-pointing the user should be able to point at targets with reasonable accuracy so as to identify the target correctly. The error in pointing is also dependent on the level of detail the user wishes. For example, if a user is pointing to a window of a far away building then the degree of pointing error is more critical compared to when the user is pointing at a building as a whole at the same distance. The pointing should be more accurate when users point at a window than pointing at the building.

4.2 Problem Formulation and Analysis

In this study, we were primarily interested in the difference between the line-of-sight of the eye and the pointer direction. Deviations between the two can lead to wrong identification of the potential target; therefore, the pointing behavior of subjects with the pointer held close to the eyes and with their arms outstretched was studied in order to reduce the error in pointing.

Mathematically, solving the pointing error problem corresponds to finding the angle the eye makes with the X-Y plane while pointing at a potential target. This angle can be compared with the pitch values obtained while pointing with arms outstretched. The corrected pitch values can be used as an input for line-of-sight computation on a TIN model stored in a handheld device.

4.3 Human Figure Model

The human body has 40 degrees of freedom (Philips *et al.* 1993). Figure 4.1 shows the articulated model of the human figure. Pointing to a potential target by a human can be achieved by using different degrees of freedom in his or her elbows, wrists, and shoulder.

Searching a three dimensional space for a solution based on so many degrees of freedom is complex. In order to reduce the complexity of finding a solution for errors in pointing in three-dimensional space, the following assumption is imposed as a constraint to find a solution for pointing errors:

> *The arm is held straight such that the line-of-sight of the eye over the tip of the pointer coincides with the direction of the pointer while focusing on potential targets.*

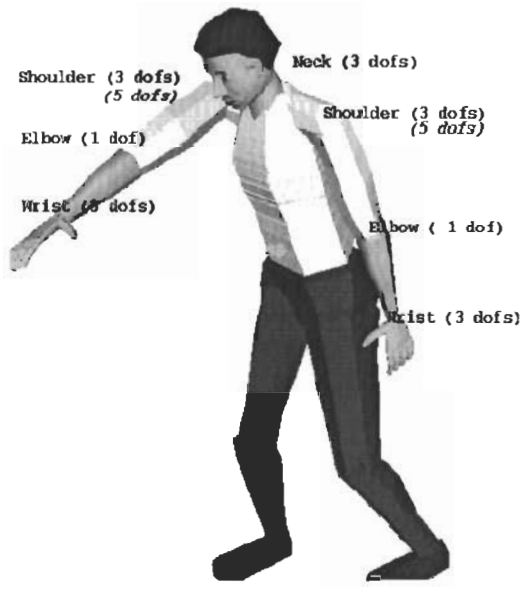


Figure 4.1: Articulated model of the human figure with degrees of freedom (dofs).

4.4 Pointing Error Compensation

The pointing error correction problem involves computation of a correction factor that reduces the angular difference between the line-of-sight of the eyes over the tip of the pointer and the pointer direction. Figure 4.2 shows the error in pointing to potential targets with the user's arm outstretched.

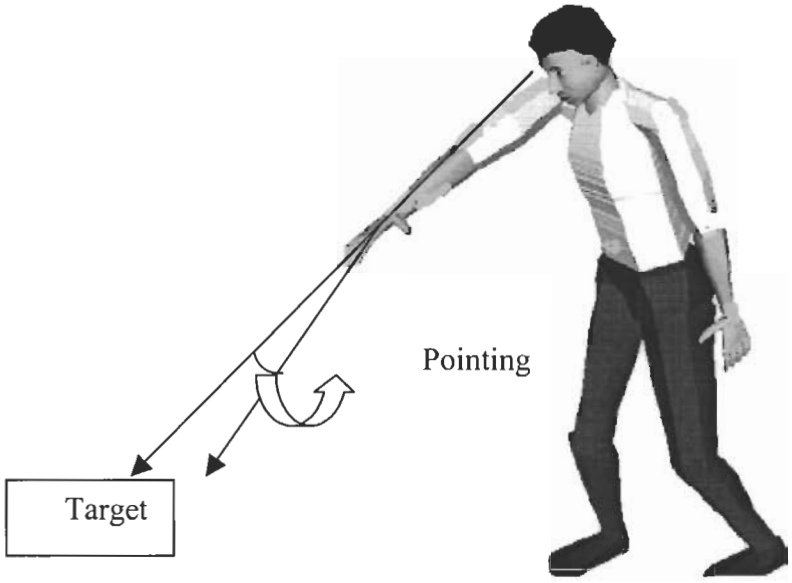


Figure 4.2: Pointing error between the eye and the arm.

To solve this problem, a pointing error correction method was developed that computes the angle between the ray emanating from the eye and the X-Y plane. A descriptive model was considered with a coordinate system centered at the user's neck between two shoulders (Figure 4.3). The problem addressed in the following section is: given a coordinate system centered at the neck, the distances from neck to eye, neck to

shoulder joint, shoulder joint to handheld point, and the pointing direction (θ,α), we wish to compute the pitch (vertical angle α), between the eye and the X-Y plane.

The following notation will be used in the model.

SP: distance between shoulder and the pointer.

EN: distance between eye and the neck.

NS: distance between neck and the shoulder.

P_{xy}: projection of point P on the X-Y plane.

P_{zy}: projection of point P on the Z-Y plane.

θ: the horizontal angle between the projections of SP, that is, SP_{xy} with the X-axis.

α: the vertical angle between SP and the X-Y plane.

ψ: the angle between the handheld pointer, eye, and neck (i.e. $\angle PEN$).

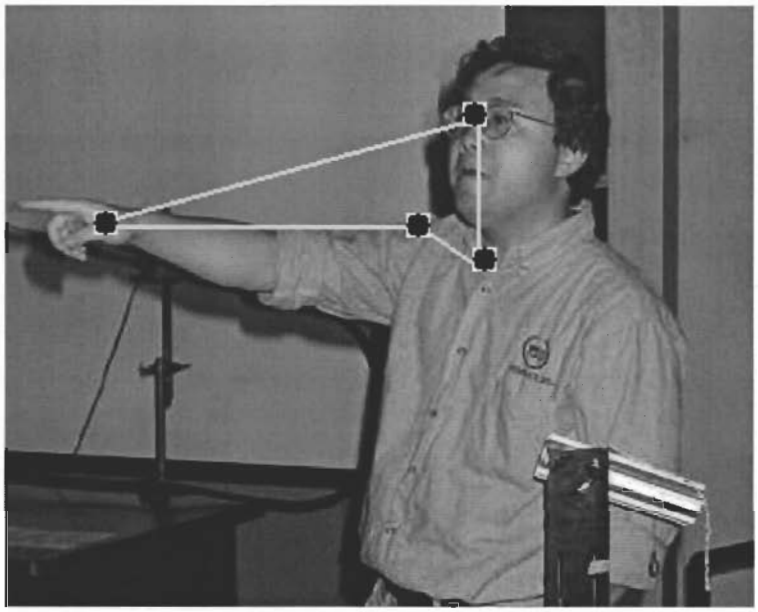


Figure 4.3: User's coordinate system centered at the neck.

4.5 Determination of Correction Factor

Given the distances SP, EN, NS and the pointing direction that comprises of pitch (angle made with X-Y plane) and yaw (angle between the direction of the pointer tip and the axis perpendicular to it), the correction factor can be derived as follows. Figure 4.3 shows the mathematical model with neck as the origin.

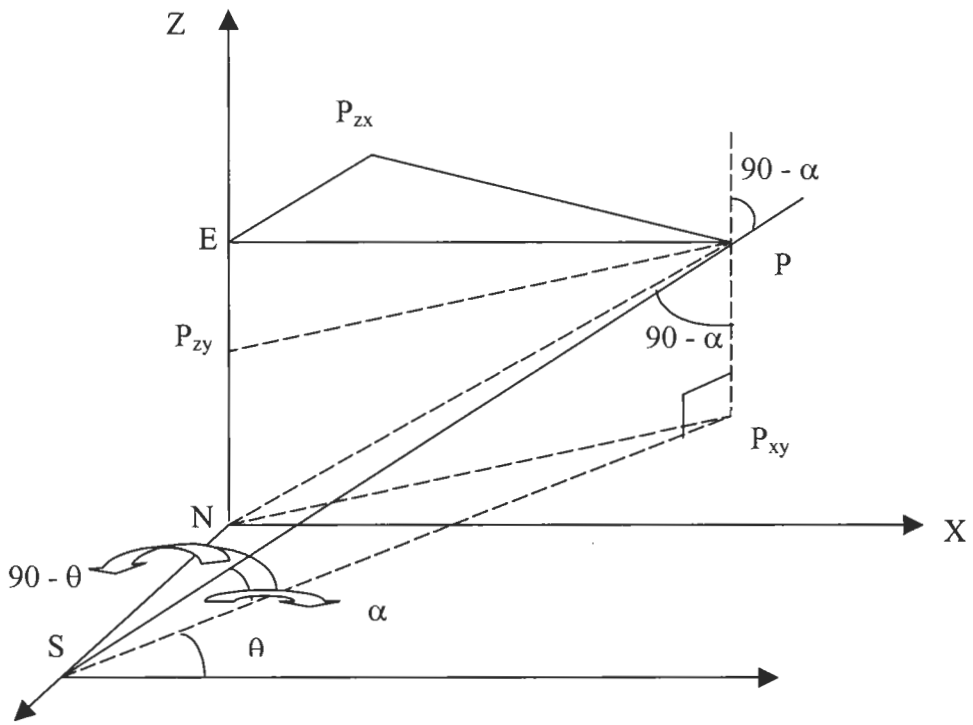


Figure 4.4: Coordinate system centered at the neck N.

From ΔSPP_{xy} we have:

$$SP_{xy} = SP * \cos(\alpha) \tag{4.1}$$

$$PP_{xy} = SP * \sin(\alpha) \tag{4.2}$$

From ΔNSP_{xy} we have:

$$NP_{xy}^{\ 2} = NS^2 + SP_{xy}^{\ 2} - 2 * NS * SP_{xy} * \cos(90 - \theta) \tag{4.3}$$

From ΔNPP_{xy} we have:

$$NP = \sqrt{PP_{xy}^{\ 2} + NP_{xy}^{\ 2}} \tag{4.4}$$

From the figure 4.5, we know that $EP_{zy} = EN - NP_{zy}$

$NP_{zy} = PP_{xy.}$ Therefore $EP_{zy} = EN - PP_{xy.}$

From $\Delta EP_{zy}P$ we have:

$$EP = \sqrt{EP_{zy}^{\ 2} + PP_{zy}^{\ 2}} \tag{4.5}$$

Since $PP_{zy}=NP_{xy}$,

$$EP = \sqrt{EP_{zy}^{\ 2} + NP_{xy}^{\ 2}} \tag{4.6}$$

From ΔENP we have:

$$NP^2 = EN^2 + EP^2 - 2 * EN * EP * \cos(\Psi) \tag{4.7}$$

$$\Psi = Cos^{-1}\left(\frac{EN^2 + EP^2 - NP^2}{2 * EN * EP}\right) \tag{4.8}$$

Thus, we can compute correction factor as follows:

$$CF = 90 - \psi$$

Where $\theta \in [0,2\pi]$ *and* $\alpha \in \left[-\pi/2, \pi/2\right]$

The angle made by the eye with the X-Y plane is 90- ψ.

The pointing error compensation algorithm is shown below:

Computation Program for Compensating Errors in Pointing

Program: Pointing Error Compensation for pointing using a smart pointer with arms outstretched: Numeric
Input: Pitch (α), yaw (θ) obtained form the orientation sensors and the distances from the neck to the eye (EN), shoulder joint (NS), and the distance from the shoulder joint to the handheld point (SP).
Output: Compensation factor (CF), angle made by the eye while looking at the target with the X-Y plane.
method:

(1) From the triangle made by the projection of the handheld point P onto the X-Y plane (P_{xy}) with the shoulder joint (S) and the handheld point P we have:

$$SP_{xy} := SP * \cos(\alpha)$$

$$PP_{xy} := SP * \sin(\alpha)$$

(2) From the triangle made by the neck (N), shoulder joint (S), and the projection of pointer P onto the X-Y plane (P_{xy}) we have:

$$NP_{xy}^{\ 2} := NS^2 + SP_{xy}^{\ 2} - 2 * NS * SP_{xy} * \cos(90 - \theta)$$

(3) From the triangle made by the neck (N), handheld point (P), and the projection of pointer P onto the X-Y plane (P_{xy}) we have:

$$NP := \sqrt{PP_{xy}^{\ 2} + NP_{xy}^{\ 2}}$$

(4) From the triangle made by the handheld point (P), eye (E), and the projection of handheld point P onto the Y-Z plane (P_{yz}) we have:

$$EP := \sqrt{EP_{zy}^{\ 2} + P_{zy}P^2}$$

(5) From the triangle made by the eye (E), neck (N), and the handheld point (P) we have:

$$\Psi := \arccos\left(\frac{EN^{\ 2} + EP^{\ 2} - NP^{\ 2}}{2 * EN * EP}\right)$$

(6) Compute correction factor (CF), the angle made by the eye while looking at target with the X-Y plane:

$$CF := \pi/2 - \Psi$$

4.6 Summary

For query-by-pointing the user should be able to point at targets with reasonable accuracy so as to identify the target correctly. This chapter describes a method for error compensation while pointing with arms outstretched. This chapter also presents a computational program to minimize the angular difference between the line-of-sight of the pointer and the eyes while using a handheld device to identify remote geographic objects by pointing at them. The computational algorithm reduces the 3-dimensional problem to 2-dimensions, thus simplifying the problem.

Chapter 5

ALGORITHM ASSESSMENT

This chapter, describes a pilot study with a gyro-enhanced orientation sensor and three participants to perform pointing at specified targets in a given room. The results for the experiment were used to test the pointing error compensation method developed earlier (Section 4.5) for pointing using a handheld device.

5.1 Experimental Setup

The goal of the experiment was to track the difference between the line-of-sight of the eye over the tip of the pointer and the direction of the pointer. We conjectured that pointing is more accurate if the pointer is held close to the eyes than with the arms outstretched due to lesser difference in the respective line-of-sight. All participants held a gyro-enhanced sensor between their thumb and forefingers. The gyro is used for pitch (angle made with the X-Y plane), yaw (angle between the direction of the pointer tip and the magnetic north), and roll (rotation along axis perpendicular to the pointer tip) measurements. We assumed that the pen mounted on the pointer might be more stable in focusing on targets for accurately measuring the angles (Figure 5.1). Nine targets points were marked at three different corners of the room (411 X 360 X 240 cm^3). The subjects pointed at the targets and the observations for pitch and yaw were made. All pointing movements were made using the right arm. Subjects stood up straight and made pointing movements in two conditions: (1) pointer held close to their eyes and (2) pointer held in a hand with the arm outstretched such that the line-of-sight of the eye over the tip of the

pointer coincided with the direction of the pointer. The subject stood at a fixed position facing the target point in the room. Restrictions were imposed on head and eye movements such that the head was always held straight.

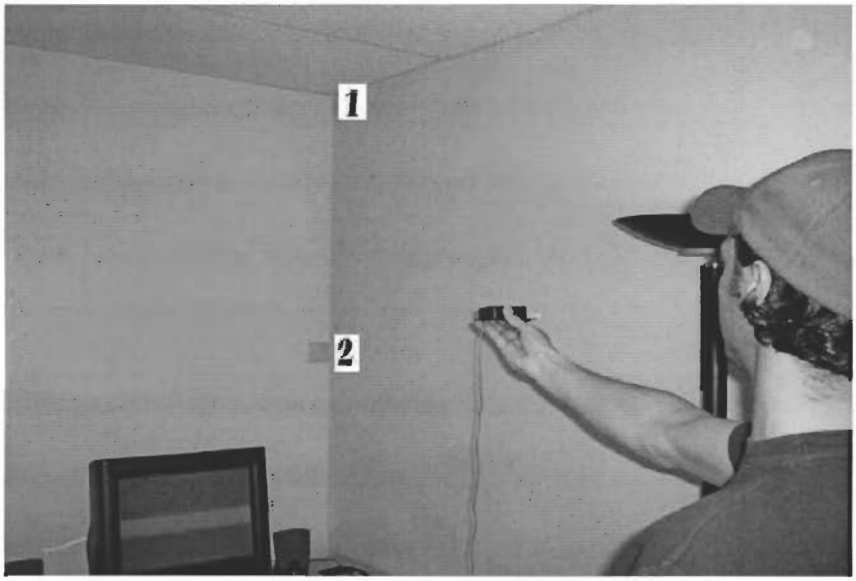


Figure 5.1: Subject pointing at a target points.

5.2 Observations

To analyze the pointing data in a spherical coordinate system with the origin at the middle of the neck, measurements were taken of distances from the middle of the neck to the center of the eyes, shoulder joint, and from tip of the shoulder to the handheld pointer for all subjects. Table 5.1 shows the measurements for the different subjects pointing to specified targets in a room.

Subjects	Distances from eye-neck (EN) (cm)	Distances from neck-shoulder (NS) (cm)	Distances from shoulder-handheld pointer (SP) (cm)
D	26	27	69
F	20	21	64
C	21	21	69

Table 5.1: Measurements for three different subjects (D, F, C) with center of the neck as the origin.

The room measurements were also taken to calculate the true pitch values based on subject and target location. Table 5.2 shows the pitch values and distances of different target points based on subject location.

Target Points	True pitch values based on observer and target location (RC) (degrees)	Distance from the observer location (cm)
1	16.50	225.96
2	-8.50	219.05
3	-39.20	279.17
4	10.56	349.76
5	-5.32	345.34
6	-27.10	386.28
7	10.38	355.01
8	-6.85	351.71
9	-26.70	391.04

Table 5.2: True pitch values based on observer and target location (RC).

5.3 Accuracy of Orientation Sensors when Pointing

The orientation sensors can handle maximum angular velocity (accuracy of rate readings) of 300 degrees per second. Although the sensors provide a good dynamic response, they are more prone to errors while pointing to potential targets. The small variations in pitch values can be significant if the target is located at a far away distance (Fitts 1954). We conducted the experiment by holding the sensors close to the eyes (called *telescope pointing*) and compared the pitch values with the values based on the location of subject and target in the room (called *room coordinates*). Each subject pointed three times to nine different targets. Table 5.3 shows the preliminary results from pointing at various marked targets by subjects.

Target Points	Subject Observations	Telescope pointing (TP Pitch) (degrees)	Room Coordinate (RC Pitch) (degrees)	Difference of TP and RC (degrees)
	D1	19.20		2.70
	D2	17.90		1.40
	D3	17.70		1.20
	F1	20.00		3.50
1	F2	21.00	16.50	4.50
	F3	19.00		2.50
	C1	20.60		4.10
	C2	18.20		1.70
	C3	23.10		6.60
	D1	-4.50		4.00
	D2	-4.50		4.00
	D3	-4.70		3.80
	F1	0.00		8.50
2	F2	1.50	-8.50	7.00
	F3	4.50		4.00
	C1	-3.10		5.40
	C2	-4.20		4.30
	C3	-2.20		6.30
	D1	-37.50		1.70
	D2	-37.50		1.70
	D3	-37.20		2.00
	F1	-31.50		7.70
3	F2	-29.50	-39.20	9.70
	F3	-32.50		6.70
	C1	-38.50		0.70
	C2	-38.70		0.50
	C3	-41.50		-2.30
	D1	11.50		0.94
	D2	11.40		0.84
	D3	11.80		1.24
	F1	16.50		5.94
4	F2	15.50	10.56	4.94
	F3	14.50		3.94
	C1	14.20		3.64
	C2	14.40		3.84
	C3	14.50		3.94
	D1	-3.50		1.82
	D2	-3.50		1.82
	D3	-3.70		1.62
	F1	-3.00		2.32
5	F2	-1.00	-5.32	4.32
	F3	-2.50		2.82
	C1	0.40		5.72
	C2	-2.90		2.42
	C3	-2.90		2.42

Table 5.3: Pitch deviation for all subjects for target points (a) 1-5 (TP vs. RC), and (b) 6-9 (TP vs. RC)

Table 5.3 Continued

Target Points	Subject Observations	Telescope pointing (TP Pitch) (degrees)	Room Coordinate (RC Pitch) (degrees)	Difference of TP and RC (degrees)
	D1	-23.30		3.80
	D2	-25.70		1.40
	D3	-24.50		2.60
	F1	-21.00		6.10
6	F2	-17.50	-27.10	9.60
	F3	-19.00		8.10
	C1	-25.90		1.20
	C2	-24.70		2.40
	C3	-23.70		3.40
	D1	11.50		1.12
	D2	10.90		0.52
	D3	11.10		0.72
	F1	12.00		1.62
7	F2	11.80	10.38	1.42
	F3	12.50		2.12
	C1	13.80		3.42
	C2	10.30		-0.08
	C3	14.30		3.92
	D1	-4.20		2.65
	D2	-4.30		2.55
	D3	-4.80		2.05
	F1	-3.50		3.35
8	F2	-1.50	-6.85	5.35
	F3	-3.00		3.85
	C1	-2.90		3.95
	C2	-3.90		2.95
	C3	-2.10		4.75
	D1	-23.10		3.60
	D2	-25.10		1.60
	D3	-24.50		2.20
	F1	-15.50		11.20
9	F2	-18.50	-26.7	8.20
	F3	-17.50		9.20
	C1	-24.70		2.00
	C2	-23.30		3.40
	C3	-23.30		3.40

We used the results to observe the deviation in pitch values while holding the pointer close to the eyes with true pitch values based on measurement of observer and target location. Table 5.4 shows the standard deviation of pitch values based on the mean of differences of pitch values based on pointer held close to the eyes and location of observer and target in the room. The standard deviation is calculated first by deviations of each subject per target and also on deviations per target by all subjects. Figure 5.2 shows standard deviations per target.

Target Points	Subjects	Mean (TP -RC) per subject per target	Standard deviation per subject per target	Mean (TP-RC) per target	Standard deviation per target
	D	1.76	0.81		
1	F	3.50	1.00	3.13	1.64
	C	4.13	2.45		
	D	3.93	0.11		
2	F	6.50	2.29	5.25	1.57
	C	5.33	1.00		
	D	1.80	0.17		
3	F	8.03	1.52	3.15	3.71
	C	-0.36	1.67		
	D	1.06	0.20		
4	F	4.94	1.00	3.25	1.72
	C	3.80	0.15		
	D	1.75	0.11		
5	F	3.15	1.04	2.80	1.27
	C	3.52	1.90		
	D	2.60	1.20		
6	F	7.93	1.75	4.28	2.81
	C	2.33	1.10		
	D	0.78	0.30		
7	F	1.72	0.36	1.64	1.24
	C	2.42	2.17		
	D	2.41	0.32		
8	F	4.18	1.04	3.49	1.01
	C	3.88	0.90		
	D	2.46	1.02		
9	F	9.53	1.52	4.97	3.36
	C	2.93	0.80		

Table 5.4: Standard deviations based on the mean value of differences between TP and RC by (1) per subject per target, and (2) per target.

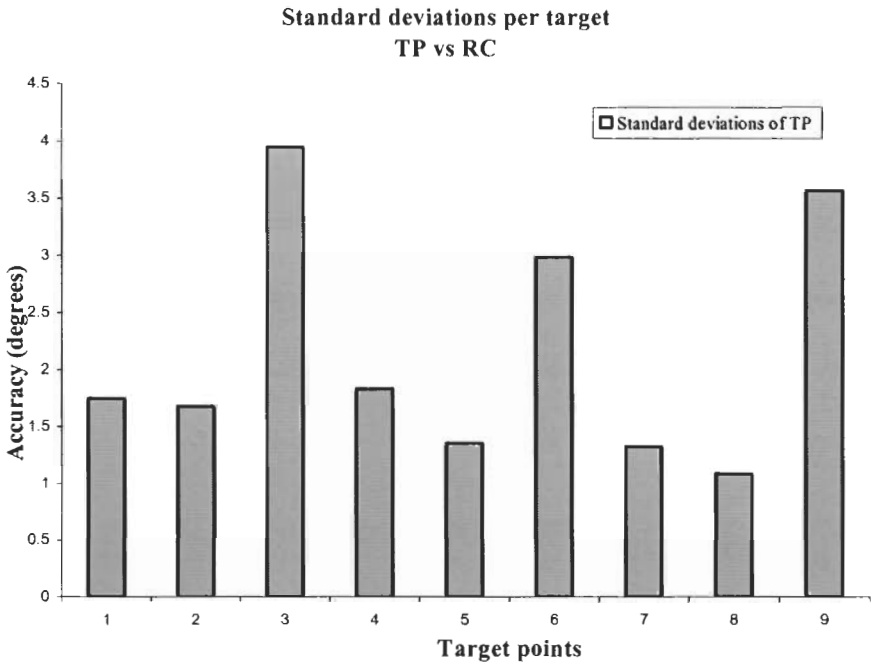


Figure 5.2: Standard deviations per target for TP vs. RC.

The pitch values vary with subjects, but follow a similar pattern. The deviations are maximum for target points (3,6,9) and lesser for target points (1,2,4,5,7,8) from the true pitch values. Figure 5.3 shows standard deviations based on elevations of the targets.

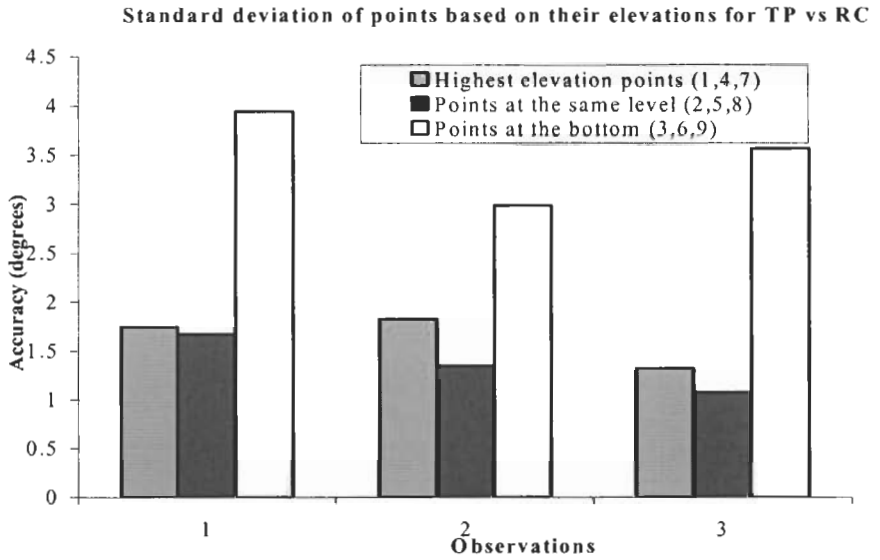


Figure 5.3: Standard deviations of points based on elevations TP vs. RC.

We can see from Figure 5.3 that pointing at a target point located at a higher elevation is more accurate than pointing downwards while holding the pointer close to the eyes. However, the deviations are not too large. This suggests that the line-of-sight of the eye over the tip of the pointer coincides better when the pointer is held close to the eyes. People can point in different ways. Another test was conducted by holding the arm straight and pointing with arms outstretched

5.4 Pitch Deviations from Different Pointing Styles

When we assume that the errors in pointing while holding the pointer close to the eyes (telescope pointing) are due to the slight difference in the respective line-of-sight of the pointer and the eye, the pitch deviation when pointer is held away from the eyes would be

far more compared to the earlier case. Data from the measurements of pitch values while pointing with arms outstretched (AOS) was compared with the pitch values obtained while holding the pointer close to the eyes (TP). Table 5.5 shows the variation of pitch values for all subjects.

Target Point	Subject observations	TP Pitch (degrees)	AOS pitch (degrees)	Difference of AOS and TP (degrees)
1	D1	19.20	32.50	13.30
	D2	17.90	28.60	10.70
	D3	17.70	29.80	12.10
	F1	20.00	39.10	19.10
	F2	21.00	37.00	16.00
	F3	19.00	37.50	18.50
	C1	20.60	18.10	-2.50
	C2	18.20	28.30	10.10
	C3	23.10	22.10	-1.00
2	D1	-4.50	6.30	10.80
	D2	-4.50	4.50	9.00
	D3	-4.70	7.10	11.80
	F1	0.00	13.50	13.50
	F2	1.50	10.00	11.50
	F3	4.50	10.50	15.00
	C1	-3.10	6.90	10.00
	C2	-4.20	2.30	6.50
	C3	-2.20	5.20	7.40
3	D1	-37.50	-25.60	11.90
	D2	-37.50	-28.50	9.00
	D3	-37.20	-26.60	10.60
	F1	-31.50	-16.50	15.00
	F2	-29.50	-20.50	9.00
	F3	-32.50	-22.50	10.00
	C1	-38.50	-37.30	1.20
	C2	-38.70	-29.50	9.20
	C3	-41.50	-39.40	2.10
4	D1	11.50	26.80	15.30
	D2	11.40	22.2	10.80
	D3	11.80	23.10	11.30
	F1	16.50	28.00	11.50
	F2	15.50	26.00	10.50
	F3	14.50	29.50	15.00
	C1	14.20	13.20	-1.00
	C2	14.40	18.70	4.30
	C3	14.50	14.50	0.00
5	D1	-3.50	10.60	14.10
	D2	-3.50	7.70	11.20
	D3	-3.70	7.40	11.10
	F1	-3.00	13.50	16.50
	F2	-1.00	13.00	14.00
	F3	-2.50	10.50	13.00
	C1	0.40	0.60	0.20
	C2	-2.90	5.30	8.20
	C3	-2.90	5.40	8.30

Table 5.5: Deviations of pitch values for all subjects for target points (a) 1-5 (AOS vs. TP), and (b) 6-9 (AOS vs. TP)

Table 5.5 Continued

Target Points	Subject Observations	TP Pitch (degrees)	AOS pitch (degrees)	Difference of AOS and TP (degrees)
6	D1	-23.30	-9.30	14.00
	D2	-25.70	-13.40	12.30
	D3	-24.50	-13.20	11.30
	F1	-21.00	-9.00	12.00
	F2	-17.50	-8.50	9.00
	F3	-19.00	-7.50	11.50
	C1	-25.90	-21.90	4.00
	C2	-24.70	-16.50	8.20
	C3	-23.70	-14.80	8.90
7	D1	11.50	23.20	11.70
	D2	10.90	22.10	11.20
	D3	11.10	23.90	12.80
	F1	12.00	27.50	15.50
	F2	11.80	29.00	17.20
	F3	12.50	28.00	15.50
	C1	13.80	17.90	4.10
	C2	10.30	18.20	7.90
	C3	14.30	18.40	4.10
8	D1	-4.20	8.60	12.80
	D2	-4.30	7.90	12.20
	D3	-4.80	8.80	13.60
	F1	-3.50	12.00	15.50
	F2	-1.50	11.50	13.00
	F3	-3.00	10.00	12.00
	C1	-2.90	2.60	5.50
	C2	-3.90	4.10	8.00
	C3	-2.10	7.80	9.90
9	D1	-23.10	-11.30	11.80
	D2	-25.10	-10.90	14.20
	D3	-24.50	-10.30	14.20
	F1	-15.50	-11.00	4.50
	F2	-18.50	-7.50	11.00
	F3	-17.50	-6.50	11.00
	C1	-24.70	-17.10	7.60
	C2	-23.30	-13.10	10.20
	C3	-23.30	-17.10	6.20

Table 5.6 shows the standard deviation of pitch values based on the mean of differences of pitch values based on pointing with arms outstretched and pointer held close to the eyes. The standard deviation is calculated first by deviations of each subject per target and also on deviations per target by all subjects. Figure 5.4 shows the deviations from true pitch values.

Target Points	Subjects	Mean (AOS -TP) per subject per target	Standard deviation per subject per target	Mean (AOS-TP) per target	Standard deviation per target
	D	12.03	1.30		
1	F	17.86	1.64	10.70	7.74
	C	2.20	6.88		
	D	10.53	1.41		
2	F	13.33	1.75	10.61	2.73
	C	7.96	1.81		
	D	10.50	1.45		
3	F	11.33	3.21	8.66	4.40
	C	4.16	4.38		
	D	12.46	2.46		
4	F	12.33	2.36	8.63	6.06
	C	1.10	2.81		
	D	12.13	1.70		
5	F	14.50	1.80	10.73	4.79
	C	5.56	4.64		
	D	12.53	1.36		
6	F	10.83	1.60	10.13	2.96
	C	7.03	2.65		
	D	11.90	0.81		
7	F	16.06	0.98	11.11	4.84
	C	5.36	2.19		
	D	12.86	0.70		
8	F	13.83	1.44	11.50	3.12
	C	7.80	2.20		
	D	13.40	1.38		
9	F	8.83	3.75	10.07	3.37
	C	8.00	2.02		

Table 5.6: Standard deviations based on the mean value of differences between AOS and TP by (1) per subject per target, and (2) per target.

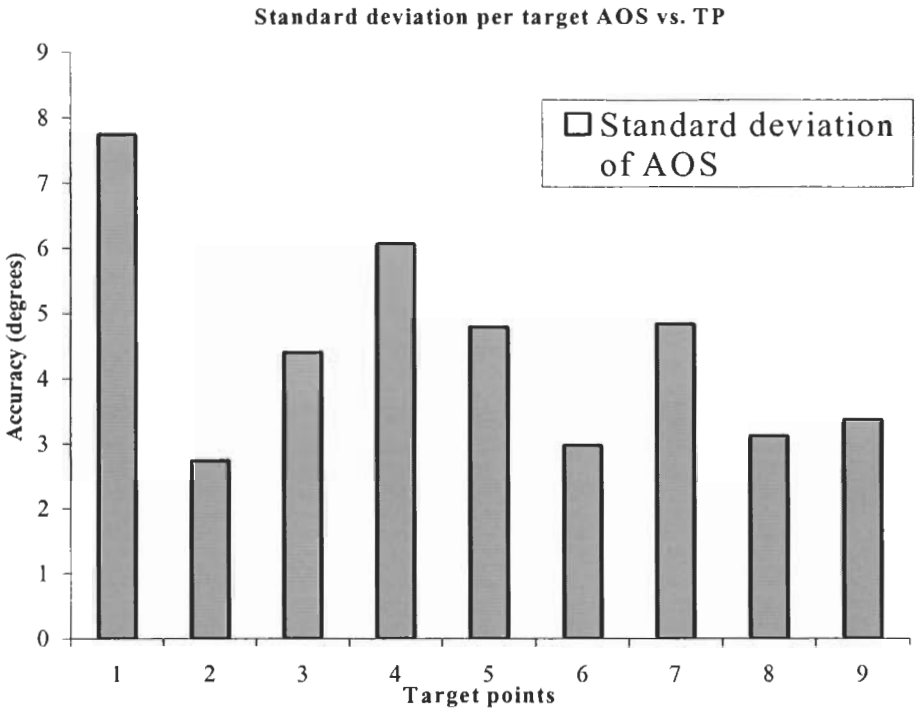


Figure 5.4: Standard deviations of pitch values per target for AOS vs. TP.

We can see that the variations are higher when pointing with arms outstretched than holding the pointer close to the eyes. The plot follows a different pattern while pointing at various targets with arms outstretched than telescope pointing. The deviation is least while pointing at target points (2,6,8) and is maximum while pointing at target points (1,4,7) (Figure 5.5). We can conclude that the probability of errors increases while pointing at target points located at a higher elevation than pointing downwards with arms outstretched. The pointing error compensation algorithm was applied to the data obtained while pointing with arms outstretched.

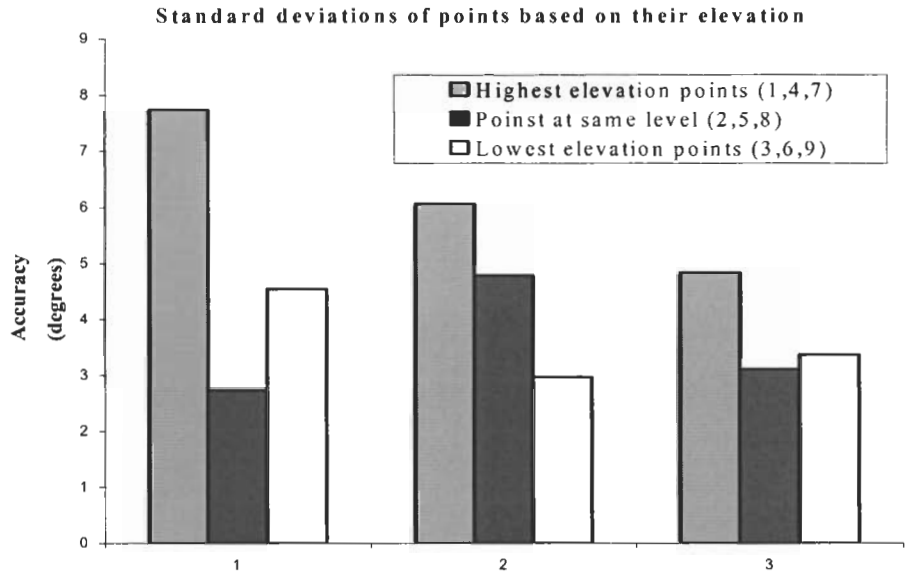


Figure 5.5: Standard deviations of points based on elevations AOS vs. TP.

5.5 Accuracy Improvement After Computational Error Compensation

The error correction was applied to the data obtained while pointing with arms outstretched (AOS) and compared with true pitch values (RC) and also with values obtained while holding the pointer close to the eyes (TP). Table 5.7 shows the standard deviation of the corrected pitch values (CP), AOS pitch values, and TP pitch values with respect to the values based on user and target location (RC).

Target Point	Standard deviation of AOS pitch values w.r.t RC SD (AOS vs. RC)	Standard deviation of CP pitch values w.r.t RC SD (CP vs. RC)	Standard deviation of TP pitch values w.r.t RC SD (TP vs. RC)
1	16.29	3.75	9.47
2	17.17	5.82	4.58
3	14.57	5.17	5.87
4	13.90	3.90	9.68
5	14.94	3.27	6.42
6	15.99	5.44	3.95
7	14.21	2.18	8.46
8	16.20	3.86	4.70
9	16.38	6.37	3.08

Table 5.7: Standard deviation of AOS, CP, TP w.r.t true pitch values (RC).

The plot (Figure 5.6) shows the standard deviations of the pitch values from the true pitch values for three different cases. The pitch values after the correction factor is applied are much closer compared to the values obtained while pointing with arms outstretched.

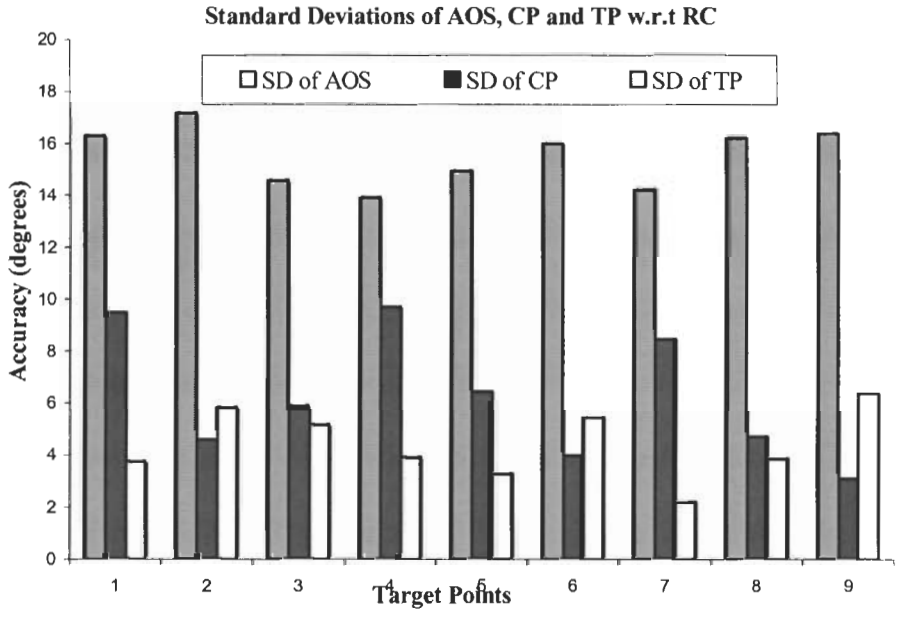


Figure 5.6: Standard deviations from RC for pitch values based on arms outstretched (AOS), corrected pitch values (CP), and telescope pointing (TP).

The standard deviation of the corrected values is less for all the target points compared to values obtained while pointing with arms outstretched. With three subjects, we got some significant results. We found a novel way of correcting the error in pointing using a handheld device with arms outstretched by measuring the angles and distances between shoulder, neck, and eyes. The results of the experiment were in close agreement to the expected results of the error computational model. The pitch values after the correction factor was applied were much closer to actual pitch values based on the room coordinates. Although there is still a slight difference between the pitch values, it might be the result of errors in measurement of distances between the neck, shoulder and the eyes. Also the pitch values of the gyroscope varied in the range of few degrees and hence it was difficult to record the exact pitch value while pointing at potential targets. The errors might also be due to the difference in the angular displacement of the head with the pointer angular movement while pointing on a potential target.

The errors in pointing to targets at a higher elevation were much more while pointing with arms outstretched than in telescope pointing. Where as while pointing at target points located at a lower elevation the results from telescope pointing showed more deviation than while pointing with arms outstretched. We conclude that since it is easier to move the wrist upward than downwards while holding the pointer close to the eyes the probability of error increased for target points located at a lower elevation while holding the pointer close to the eyes. In the case of pointing with arms outstretched the results are less accurate while pointing at target points located at a higher elevation as the pull of gravity is much more when the arm is held out straight and pointing at a higher angle than while pointing downwards. We varied the distances between the eye, neck and arm

length. The corrected pitch values varied in the range of few degrees but the change was not too large. Thus we can conclude that even for a taller or shorter person the pointing error compensation method would still hold good. We conclude that in order to guarantee accurate results while pointing with arms outstretched, the vertical head-and-eye orientation over the tip of the pointer should move in synchronization while focusing on potential targets. The hypothesis of this thesis is defined as: Pointing error compensation method reduces the differences between the line-of-sight and the pointer direction while pointing with arms outstretched. We supported our hypothesis by conducting an experiment using orientation sensors and three subjects who pointed at specified targets in a given room. The results from the pilot study confirmed our hypothesis.

5.6 Summary

This chapter describes the experiment conducted to calibrate and test the pointing error compensation model. The goal of the experiment was to find out the focus of attention target while pointing with arms outstretched. The results from the experiment showed less deviation from the true pitch values in comparison to the values obtained while pointing with arms outstretched. The errors in pointing to targets at a higher elevation while pointing with arms outstretched were greater than the ones in telescope pointing. However, for points located at a lower elevation the results from telescope pointing showed more deviation than while pointing with arms outstretched. In order to guarantee accurate results while pointing with arms outstretched the angular head-and-eye orientation over the tip of the pointer should be proportional to the pointer angular motion while focusing on potential targets.

Chapter 6

CONCLUSIONS AND FUTURE WORK

This thesis deals with algorithms and errors while pointing with a mobile handheld device. Important issues in this context include computation of a line-of-sight model, and an error correction factor for handheld devices when used for pointing at distant targets to query a given object. This chapter summarizes the thesis work and presents conclusions. Future work on this thesis is also highlighted.

6.1 Summary

The main objective of this thesis was to compensate for errors while pointing with a handheld device for querying remote geographic objects. In this context there were three main areas to investigate: (1) the algorithms involved in the line-of-sight computation for a TIN model, (2) pointing error compensation method, and (3) the evaluation of pointing error compensation method. The following three sections summarize the investigations in each of the three areas.

6.1.1 Algorithms

The algorithms focused on the TIN model and line-of-sight computations. The objective was to obtain an efficient model that can be used as a base for computing line-of-sight model based on pointing direction. There are many algorithms available for computing the line-of-sight between the user and the object in a given TIN model; however many

algorithms are of theoretical interest and not of much practical significance because of difficulty in implementation. The algorithms and the complexities involved in computing the line-of-sight between two points were discussed in detail.

6.1.2 Pointing Error Compensation Method

We considered a spherical coordinate system centered at the neck to compensate for errors in pointing while pointing with arms extended. An error correction factor was computed based on the distances measured from the center of the neck to the center of the eyes, center of the neck to the shoulder joint, and from the shoulder joint to the handheld pointer. The pointing direction (θ,α), were also used for computing the pointing error compensation factor.

6.1.3 Evaluation of Pointing Error Compensation Method

A pilot study was conducted using a gyro based orientation sensor and three participants to perform pointing at specified targets in a given room. The goal of this study was to evaluate the pointing error compensation method. The pitch (α) values were collected for each participant pointing to the same target in a room: (1) with their arms outstretched and (2) by holding the pointer close to the eyes. The error correction factor was computed for different participants by considering a spherical coordinate system centered at the neck and measuring the distances from the neck to the eye, shoulder and the arm length. The error correction factor was applied to the data obtained from the gyroscope when users pointed with their arms outstretched and compared to the actual pitch values based on room coordinates. The result of this experiment gave us an insight into how errors can

be corrected for a user when he or she is pointing with his or her arms outstretched and can be helpful for human computer interaction for designing accurate systems for querying spatial information by pointing using a handheld device.

6.2 Conclusions

In the present study we have investigated the ability to correct errors in pointing using arms outstretched by considering an articulated human model with a spherical coordinate system centered at the neck. Under the assumption that the line-of-sight coincides with the tip of the pointer while pointing at potential targets, our descriptive model gave the best fit for correcting the errors in pointing. The distances from the neck to the eye and shoulder, shoulder to the pointer held in arms outstretched, and the angles obtained from the gyroscope are the key variables involved in computing the correction factor while pointing with arms outstretched. The corrected pitch values varied in the range of few degrees for varying distances between the eye, neck, and the handheld pointer. This showed that the pointing error compensation method gives consistent results for different users. The hypothesis of this thesis is: Pointing error compensation method reduces errors in pointing with arms outstretched. We support our hypothesis by conducting an experiment using orientation sensors and three subjects who pointed at specified targets in a given room. Furthermore, we found that subjects always overshot nearby targets. This result was in good agreement with findings reported earlier in the previous studies (Stratta and Lacquaniti 1997). The descriptive model that we used to correct pointing errors assume that the errors related to pointing are more if there are more degrees of freedom. The pitch values obtained when the pointer was held close to the eyes are much

closer to the actual pitch values based on the room coordinates. Based on the results obtained from the subjects in our experiment, we can conclude that it is very important to reduce the difference in the line-of-sight of the eye over the pointer tip and the pointer direction for query-by-pointing.

The results of the experiment with a gyro-enhanced orientation sensor are promising. The errors in pointing to targets at a higher elevation were greater while pointing with arms outstretched than in telescope pointing, where as while pointing at target points located at a lower elevation the results from telescope pointing showed more deviation than while pointing with arms outstretched. We conclude that in order to guarantee accurate results while pointing with arms outstretched the vertical head-and-eye orientation over the tip of the pointer should move in synchronization while focusing on potential targets. The variations in the pitch values after applying the correction factor could be traced and attributed to the inaccuracies in measuring the distances between arm, eyes, and shoulder. The pitch values from the sensors were also inaccurate as it varied in the range of few degrees while focusing on a potential target. Thus, it was not possible to record the exact pitch values. However, the accuracy of the initial data showed significant improvement after the correction factor was applied. We are thus confident that with accurate distance measurements and precise pitch values from the sensor, the correction model can drastically reduce the difference between the line-of-sight of the eye over the tip of the pointer and the pointer direction.

We can further conclude that point visibility algorithms can be used for query-by-pointing using a handheld device based on the pointing error compensation method.

6.3 Future Work

This section lists a set of possible future research tasks that look attractive for future exploration.

6.3.1 Alternate Pointing Methods

As computerized devices become increasingly ubiquitous and interacting at a distance becomes more common, it will be important to provide interaction techniques that are quick, convenient, and accurate. Pointing with arms outstretched using a handheld device offers an interesting way to interact with mobile GIS. However, further research is required to investigate alternative methods of pointing. The ideal pointing device is a camera since many people are familiar with it. For a more exact selection of potential target by a handheld device, it can be mounted with a digital camera. The user can select a potential target and then record the exact pitch and yaw values. The pointing direction along with user's location can then be used for line-of-sight computation on the TIN model. Many interesting questions in this context are:

- How far is the potential target from the user?
- What is the level of-detail the user is interested in querying a potential target?
- How can advance computational model be developed to respond in real time?

6.3.2 Alternate Ways of Interacting at a Distance

Cameras tracking hand or eye movements will not be able to get more than a very crude estimate of where the user is pointing. Even with a digital camera, the shaking of users hand and the resolution of today's cameras results in an inability to point reliably at anything smaller. Given these human limitations we can use pointing for referencing a broad area of interest. The objects of interest can then be *snarfed* or copied onto the users handheld device so the detailed work can be performed more accurately (Myers *et al.* 2001). The concept of semantic snarfing can be applied to query-by-pointing for mobile handheld devices and can be further investigated. While this approach seems promising it raises challenging questions. How can we generate a cone centered on the line-of-sight from the head to the hand, starting at the hand, and projecting away from the person? How can the area contained within this cone be used as a broad area of interest from which the objects to be queried can be snarfed?

6.3.3 Pointing in Augmented Reality

The basic idea of augmented reality is to superimpose graphics, audio and other sense enhancements over a real-world environment in real-time. The three components needed to make an augmented system work needs a head mounted display, tracking system, and a mobile computing power. For query-by-pointing a user can view where he or she is pointing using the head mounted display. Thus a user can query the object when he or she is sure about the object of interest. Augmented reality is still in an early stage of research and development at various universities and high-tech companies. It can be an accurate

method for querying spatial information using smart head mounted displays that shows the user where he is pointing.

6.3.4 Efficient Surface Approximation Models

Terrains are three-dimensional objects, represented in a two-dimensional domain. For line-of-sight computation using a handheld device to identify constructed objects like buildings, it should be represented using efficient algorithms. Efficient algorithm to build Delaunay tetrahedrizations (i.e., the 3D extensions of triangulations) can be further explored for better representation of a surface. Much preliminary work is still needed to identify general modeling and computational issues that offer a key to a more integrated approach to spatial information handling.

6.3.5 Mobile Environment

Many line-of-sight related problems on terrains still lack of practically satisfactory solutions: we can mention problems such as the update of visibility for a *moving viewpoint*. More research efforts should be spent on finding algorithmic methods and investigating the problem, since they have a high impact on applications and a fundamental importance in the development of information systems of the future.

6.3.6 Blue-Eyed Vision of the Future

The portable smart pointer discussed in this thesis does not exist yet. However our future environment will actively support the use of handheld devices embedded in our environment completely interconnected, intuitive and effortlessly portable. Important questions in this context include:

- How to design a better user interface for human-computer interaction?
- How to efficiently query a large spatial database?
- How to represent man made objects like buildings most accurately for query-by-pointing?
- How to point accurately at targets to reduce the angular shift between the line-of-sight of the eye and the pointer?

BIBLIOGRAPHY

M. Atallah (1989) Dynamic Computational Geometry. in: *IEEE Symposium on Foundations of Computer Science*, Tucson, AZ, pp. 92-99.

T. Baker (1987) Three Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets. in: *Computational Fluid Dynamics*, Honolulu, HI, pp. 255-270.

R. Barrett, E. Selker, and J. Rutledge (1995) Negative Inertia: A Dynamic Pointing Function. in: *Human Factors in Computing Systems*, New York, pp. 316-317.

A. D. Blaser (2000) *Sketching Spatial Queries*. Ph.D, University of Maine, Orono, ME.

J. Boissonnat and K. Dobrindt (1992) On-Line Construction of the Upper Envelope of Triangle in R3. in: *Canadian Conference on Computational Geometry*, Newfoundland, Canada, pp. 311-315.

D. Caduff (2002) *Sketch-Based Queries In Mobile GIS-Environments.* M.S, University of Maine, Orono, ME.

L. Cappelletti (1997) Mobile Computing: Beyond laptops. in: *Proceedings of the 15th Annual International Conference on Computer Documentation*, Salt Lake City, UT, pp. 23-26.

R. Cole and M. Sharir (1989) Visibility Problems of Polyhedral Terrains. *Journal of Symbolic Computation* 7(1): 11-30.

A. Collins (1988) Development During the Transition to Adolescence. in: A. Collins (Ed.), *The Minnesota Symposia on Child Psychology*, Minneapolis, MN, pp. 27-33.

M. de Berg, M. Halperin, D. Overmars, and M. Snoeyink (1991) Efficient Ray Shooting and Hidden Surface Removal. in: *Symposium on Computational Geometry*, New York, pp. 21-30.

M. de Berg (1997) Visualization of TINs. in: P. Widmayer, T. Roos, and M. Kreveld (Eds.), *Lecture Notes in Computer Science*, 1340, pp. 79-97, Springer Verlag, New York.

B. Delaunay (1934) Sur la Sphere Vide Izv Akad. in: *Bulletin of the Academy of Sciences of the U.S.S.R*, pp. 793-800.

D. Dobkin and A. Tal (1995) Visualization of Geometric Algorithms. *IEEE Transactions on Visualization and Computer Graphics* 1(2): 194-204.

H. Edelsbrunner (1989) The Upper Envelope of Piecewise Linear Functions: Tight Bounds on the Number of Faces. *Discrete and Computational Geometry* 4: 337-343.

M. Egenhofer (1992) Why not SQL. *International Journal of Geographic Information Systems* 6(2): 71-85.

M. Egenhofer and A. U. Frank (1991) *Query Languages for Geographic Information Systems.* National Center for Geographic Information and Analysis, Orono, ME.

M. Egenhofer and W. Kuhn (1998) Beyond Desktop GIS. in: *Proceedings of GIS Planet*, Lisbon, Portugal, www.spatial.maine.edu/~max/BeyondDesktopGIS.pdf.

P. Fitts (1954) The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Experimental Psychology* 47: 381-391.

L. de Floriani (1989) A Pyramidal Data Structure for Triangle-Based Surface Description. *IEEE Computer Graphics and Application* 9(2): 67-78.

L. de Floriani and P. Magillo (1993) Computing Visibility Maps on a Digital Terrain Model. in: A. Frank and I. Campari (Eds.), *Lecture Notes in Computer Science* 716, pp. 248-269, Springer Verlag, New York.

L. de Floriani and P. Magillo (1994) Visibility Algorithms on Triangulated Digital Terrain Models. *International Journal of Geographic Information Systems* 8(4): 329-342.

L. de Floriani and P. Magillo (1999) Intervisibility on Terrains. in: P. Langley, M. Goodchild, and D. Rhind (Eds.), *Geographic Information Systems: Principles, Techniques, Management and Applications*, pp. 543-556.

L. de Floriani, E. Puppo, and P. Magillo (2003) Application of Computational Geometry to Geographic Information Systems. *Accessed:* February 2003, citeseer.nj.nec.com/63576.html.

S. Fortune (1987) A Sweep Line Algorithm for Voronoi Diagram. *Algorithmica* 2: 153-174.

S. Fortune (1992) Voronoi Diagrams and Delaunay Triangulations. in: D. Zhu and F. Hwang (Eds.), *Conference on Computing in Euclidean Geometry*, 1, pp. 193-233, Singapore.

A. Frank, M. Egenhofer, and D. Hudson (1997) The Design of Spatial Information Systems. *Accessed:* 2002, www.geog.buffalo.edu/~dougf/spatial_db/595p1.pdf.

W. Gellersen (1995) Modality Abstraction: Capturing Logical Interaction Design as Abstraction from "User Interface For All". in: *Workshop on User Interface for All*, Crete, Greece, http://ui4all.ics.forth.gr/UI4ALL-95/gellersen.pdf.

R. Gill, S. Gordan, S. Dean, and D. McGhee (1991) Integrating Cursor Control into the Computer Keyboard. in: *Proceedings of Human Factors Society*, pp. 256-260.

R. Gimblett and G. Ball (1991) Dynamic Spatial Models and Artificial Worlds: A Perspective on Advances in GIS Modeling into the 21st Century. in: *The Canadian Conference on GIS*, Ottawa , Canada, pp. 996-1008.

L. Guibas and J. Stolfi (1985) Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Transactions on Graphics* 4: 74-123.

A. Gupta, S. Santini, and R. Jain (1997) "In search of Information in Visual Media", *Communications of the ACM*, 40(12): 35-42.

A. Harrington (2002) Mobile GIS Depends on Location. In *Geoworld*. http://www.geoplace.com/gw/2002/0205/0205mgis.asp.

M. Helander (1988) *Handbook of Human-Computer Interaction*, Elsevier Science Publishers, Amsterdam.

J. Hershberger (1989) Finding Upper Envelope of n Line Segments in O (n log n) Time. *Information Processing Letters* 33(4): 169-174.

T. Hewett, R. Baecker, and M. Mantei (1996) Curricula for Human Computer Interaction. *Accessed:* 12/04/2003, http://sigchi.org/cdg/cdg2.html#2_1.

A. Kostli and M. Sigle (1986) The Random Access Data Structure of the DTM. *International Archives of Photogrammetry and Remote Sensing* 26(B4): 128-137.

R. Laflamme and C. Miller (1958) The Digital Terrain Model - Theory and Application. *Photogrammetric Engineering* 24(3): 442-443.

J. Lee (1991) Analyses of Visibility Sites on Topographic Surfaces. *International Journal of Geographic Information Systems* 4(4): 413 - 429.

S. Mackenzie (1991) Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction* 7(1): 91-139.

D. Mark and A. Frank (1989) Concepts of Space and Spatial Language. in: *Auto Carto9*, American Society of Photogrammetry and Remote Sensing, Baltimore, MD, pp. 538-556.

M. McCullogh (1983) Automated Contouring of Faulted Seismic Data. *Computer Graphics* 2(1): 57-66.

P. Milne (1988) Accuracy of Data Processing from Creation of Digital Terrain Model to Contouring and Surface Modeling. in: *Standard and Accuracies*, Institution of Civil Engineering Surveyors, Reading University, London, pp. 14-19.

A. Mirante and N. Weingarten (1982) The Radial Sweep Algorithm for Constructing Triangulated Irregular Networks. *IEEE Computer Graphics and Application* 2(3): 11-21.

B. Myers, C. Peck, D. Kong, R. Miller, and J. Nichols (2001) Interacting at a Distance Using Semantic Snarfing. *Lecture Notes in Computer Science* 2201: 305-312.

G. Nagy (1994) Terrain Visibility. *Computer and Graphics* 18(6): 763-773.

R. Owens (1996) *Language Development: An Introduction*, Allyn and Bacon, New York.

M. Perry, D. O'Hare, A. Sellen, B. Brown, and R. Harper (2001) Dealing with Mobility: Understanding Access Anytime, Anywhere. *ACM Transactions on Computer-Human Interaction* 8(4): 323-347.

G. Petrie and T. Kennie (1991) Terrain Modeling in Surveying and Civil Engineering. in: G. Petrie and T. Kennie (Eds.), *Introduction to Terrain Modeling-Application Fields and Terminology*, vol. 1, pp. 2-5.

T. Peucker (1977) Data Structures for Digital Terrain Models: Discussion and Comparison. in: *International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems*, Cambridge, MA, pp. 1-15.

C. Philips, N. Badler, and B. Webber (1993) Behavioral Control. in *Simulating Humans: Computer Graphics Animation and Control*, Oxford University Press, New York.

F. Preparata and M. Shamos (1985) *Computational Geometry: An Introduction*, Springer Verlag, New York.

B. Shneiderman (1990) Future Directions for Human-Computer Interaction. *International Journal of Human-Computer Interaction* 2(1): 73-90.

J. Soechting and M. Flanders (1989) Sensorimotor Representations for Pointing to Targets in Three-Dimensional Space. *Journal of Neurophysiology* 62(2): 582-594.

M. Stratta and Lacquaniti (1997) Viewer Centered Frame of Reference for Pointing to Memorized Targets in Three-Dimensional Space. *Journal of Neurophysiology* 78(3): 1601-1618.

J. Thomas, W. Buxton, B. Curtis, and T. Landauer (1999) Human Computer Interaction. in: *Human Factors and Computing Systems*, Boston, MA, pp. 253-255.

U. S. Military (2001) Maps, Map Reading, and Land Navigation. *Accessed:* 07/16 2003, http://www.rotc.monroe.army.mil/jrotc/documents/Curriculum/Unit_5/U5C2L4_txt.pdf.

R. Weibel (1997) Digital Terrain Modeling for Environmental Applications. in: *Joint European Conference on Geographic Information*, Vienna, pp. 464-474.

M. Weiser (1991) The Computer of the 21st Century. *Scientific American* 265(3): 66-75.

M. Weiss (1995) Data Structures and Algorithms Analysis. vol. 1, Addison Wesley, Boston, MA.

Westerink (1994) Pointing in Entertainment-Oriented Environments: Appreciation versus Performance. in: *Human Factors in Computing Systems*, New York, pp. 77-78.

BIOGRAPHY OF THE AUTHOR

Farhan Faisal received his undergraduate degree, Bachelor of Engineering in Mechanical Engineering, from R.E.C Silchar, India, in 1999. Thereafter, he worked at Cognizant Technology Solutions (CTS) for one year. At CTS, he held the post of a software developer and worked in many different software application areas. He then joined the University of Maine's graduate program, working as a graduate research assistant with the National Center for Geographic Information and Analysis (NCGIA) since May 2001. Farhan is a candidate for the Master of Science degree in Spatial Information Science and Engineering from The University of Maine in December, 2003.