12-2001

Modeling Intersections of Geospatial Lifelines

Ramaswam Hariharan

MODELING INTERSECTIONS OF GEOSPATIAL LIFELINES

By

Ramaswamy Hariharan

B.E. Anna University, 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, 2001

MODELING INTERSECTIONS OF GEOSPATIAL LIFELINES

By Ramaswamy Hariharan

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, 2001

Modeling moving objects involves spatio-temporal reasoning. The continuous movements of objects in space-time captured as discrete samples form geospatial lifelines. Existing lifeline models can represent the movement of objects between samples from most likely location to all possible locations. This thesis builds on a model called lifeline bead and necklace that captures all the possible locations of moving objects. Beads are 3-dimensional representations of an object's movements and a series of beads form a necklace. The extent of finding the possible locations is constrained by the speed of movement of the objects.

Intersections of lifelines occur when two or more objects meet in space-time. Lifeline intersections are determined by testing the intersection of lifeline beads and necklaces. This thesis introduces a computational model that calculates the intersection of beads and necklaces. This model reduces the complexity of calculating 3-dimensional intersections to 2-dimensional simple geometric figures.

The bead and necklace model is useful to various applications such as tracking of individuals, animals, vehicles, soldiers passing through hazardous spills, or the analysis of criminal activities. Lifeline bead intersections is applicable to interesting situations such as finding out the possibility of two individuals present in a meeting or traveling together, the incidence of an individual with environmental hazards, and determining the spatial and temporal clusters for disease patterns.

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. Kathleen Hornsby and Dr. Kate Beard, for their support and guidance.

I would like to specially thank Dr. Kathleen Hornsby who always encouraged me in all my efforts towards this thesis with valuable suggestions that were very helpful in the successful outcome of this thesis.

This research work was partially supported by a grant from the National Institute of Environmental Health Sciences, NIH, under grant number 1 R 01 ES09816-01.

My colleagues at the Department of Spatial Information Science and Engineering were great friends and guides who helped me a lot in adjusting to the new environment and adapting myself to the department. Special mention goes to Craig Miller, Thomas Windholz, Jim Farrugia, Andrea Rodríguez, Chris Frank, Boonsap Witchayangkoon, and James Wilusz. My sincere and heartfelt thanks to Chitra who was always there for any kind of support.

I would like to thank my professors at Anna University, where I had the first opportunity to step into professional studies, for all their help in making my pursuit for higher education a successful one.

I am very grateful to my parents and my family, especially my mother, who wholeheartedly supported all my efforts in life and being responsible for my present status. Not but the least, I always enjoyed the warmth of Maine, not through the weather but through the people who I felt were always kind and helpful.

I also thank all my Indian friends at the University of Maine campus who have made my stay here very memorable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Incorporating time into geographic information systems (GISs) has been requested for a long time. A workshop on Time in Geographic Space (Egenhofer and Golledge 1994) expressed concerns and needs for cognitive representations of geographic space and time, developing formal systems for spatio-temporal reasoning, and bridging the gap between human and formal systems. Handling geographic data would become better when it is viewed as a series of snapshots, that is, capturing change in geographic data with time. Langran (1992) provided a comprehensive list of application areas that would improve significantly by including time in addition to the two-dimensional geographic space. She also discussed how a temporal GIS could better explain patterns, trends, and changes in geographic data. Better visualization and data integrity checks (Abraham and Roddick 1999) are two other critical aspects for using time in GIS. The need for integrating space and time is very important to solve the problems addressed by their respective areas (Sellis 1999). While the spatial and temporal communities have started working together, there still remain a number of unsolved research issues such as representation of space and time, models and languages for spatio-temporal database management systems, graphical user interfaces, and query processing. This thesis develops a conceptual and computational model for representing and querying moving objects.

1.1 Background of the Thesis

People move through geographic space. Although their movements are always continuous, the movements can also be perceived as being discrete depending upon the temporal granularity of movement. For example, a student biking to the University of Maine, walking to a classroom, and traveling by bus during weekends describes continuous movements. Over a considerable period of time, however, the student might have moved to California for a job. This movement from Maine to California would be perceived as a discrete jump. Although both movements are captured over time, they differ greatly in semantics when queries and analyses are performed over them. A continuous movement is particularly challenging because individual typically make frequent changes in their direction and speed.

Research in tracking moving objects has been pursued for many years. Hägerstrand (1970) provided a theoretical framework for capturing individual's movement in space over time by constructing a space-time prism. Miller (1991) and Forer (1998) developed models using the space-time prism to measure an individual's accessibility to an environment. This geometric approach has also been used in an investigation of accessibility by public transport (O'Sullivan *et al.* 2000). Early attempts were made by Rugg (1973) to represent the itinerary (i.e., a sequence of activities taking place in a given set of locations over time) of people's movement in forests as maps to study their recreational behavior. The Lund school of time geography used a *timeline*, which is defined as a continuous line that traces temporal variations in positions of an object. Since the concept of a timeline occurs more often in a non-spatial context of referring to some events or attributes as a function of time, Odland (1998) called the timeline a *lifeline*, which describes the most likely space-time points of an object moving continuously from sample point *A* to sample point *B*. Odland's lifeline is a series of straight lines connecting all the sample points; therefore, his lifeline may not consider all possible locations occupied by an object in between two sample points. We introduce the concept of a *geospatial lifeline bead* (Mark *et al.* 1999; Hariharan and Hornsby 2000; Hornsby and Egenhofer (in press)) as a set of all possible geographic locations that an object could have occupied between two sample points *A* and *B*, given the maximum velocity at which the object would move between those two points. We assume this movement is continuous between two consecutive observations. The bead concept forms a strong basis for representing and querying geospatial lifelines.

1.2 Motivation for Research

Geospatial lifelines provide a way of referring back in time or extrapolating into the future to analyze the locations of individuals that participate in a certain event. Individuals meeting each other or traveling together in space can be modeled as the intersection of two or more lifelines. Intersections of lifelines are relevant for such analyses as alibi testing whether two individuals could have met in space-time.

Let us consider the following hypothetical example. Two persons are accused of theft in a shopping mall in a busy part of New York City. The New York Police Department wants to solve this problem by checking on the alibis of the accused. They have the information of their spatio-temporal whereabouts before and after the crime occurred. Let a_1 be the location of accused person *A* at time t_1 before the crime occurred and a_2 be the location at time t_2 after the crime occurred (Figure 1.1). Similarly, let b_1 be the location of person B at time t_3 before the crime occurred and b_2 be the location at time t_4 after the crime occurred (Figure 1.1). With this information the police have to figure out whether the two suspects could have been present at the crime spot at the same time.

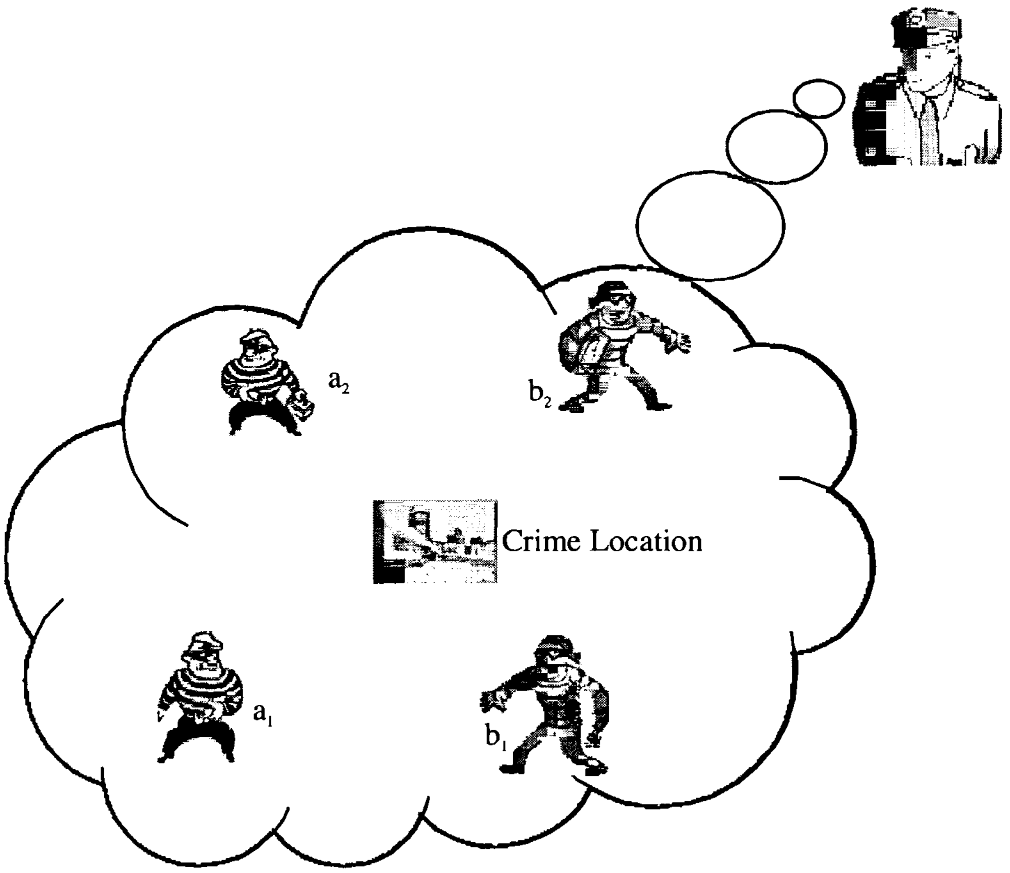


Figure 1.1 Example showing the positions of two suspects around the crime location.

Different spatio-temporal models can be designed for the movement of these individuals in space-time between the two locations. First, the suspects' movements could be modeled as *lifeline steps* (Figure 1.2a) that capture discrete changes in location. The step model assumes the individual's location to remain constant between two sample points. The lifeline has the appearance of a step where each step captures the change in location over time. Typically this model is used in capturing residential histories where the temporal resolution ranges from months to years. The application of this model to the above example will be ineffective. Alternatively, the suspects' movements can be modeled as *lifeline threads* (Figure 1.2b) that represent the *continuous* movement of an object at discrete time intervals. In this case, an interpolation method is applied to successive space-time samples to determine space-time locations between samples. The slope of the thread determines the speed of movement. The steeper the slope, the higher is the speed of movement. The interpolation gives the values between two space-time samples that are not actual recorded values, but are the *most likely* values of space-time locations. As a result, it is possible that an actual space-time location does not coincide with the interpolated space-time location, thus introducing uncertainty into the model. Hence the lifeline thread model may not capture the possibility of criminals being present in the crime spot.

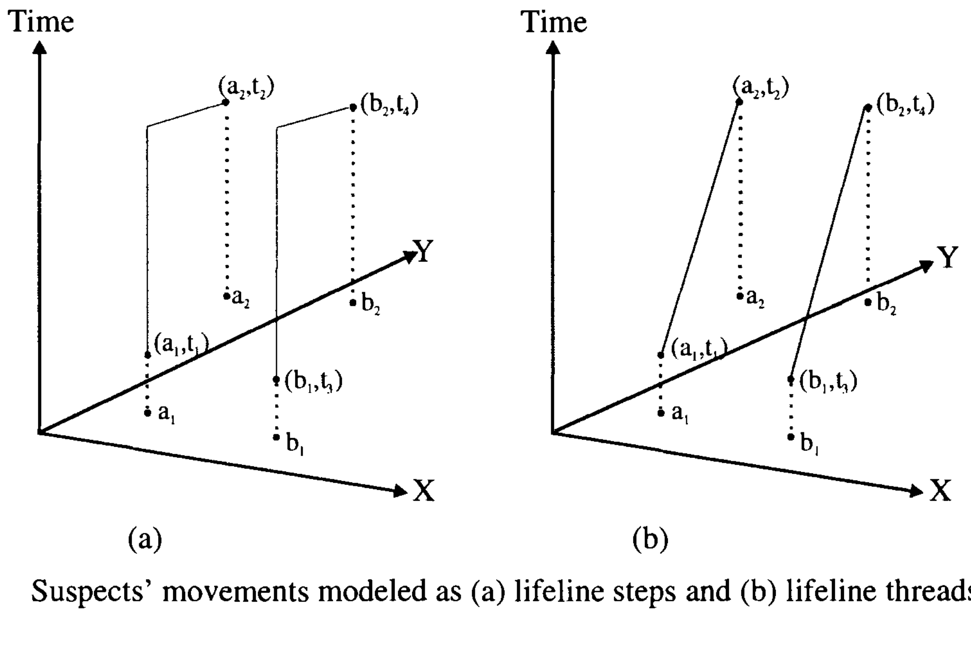


Figure 1.2 Suspects' movements modeled as (a) lifeline steps and (b) lifeline threads.

Another approach is to model each suspect's movement as a *lifeline bead* (Figure 1.3) that captures *all the possible* movements between two locations, thus eliminating the uncertainty associated with the lifeline thread model. Also, in many cases of spatio-temporal reasoning, the set of all possible values is required rather than most likely locations. To find out whether the two suspects could have met in space-time, it is not enough to form a single connection between a pair of samples as in the case of a lifeline thread, since it may exclude the location of where the individuals actually met. Modeling the movements using lifeline beads might reveal the possibility of the suspects being present in the crime spot. This thesis examines the lifeline bead model in detail and shows how useful this model is for supporting interesting queries over geospatial lifelines.

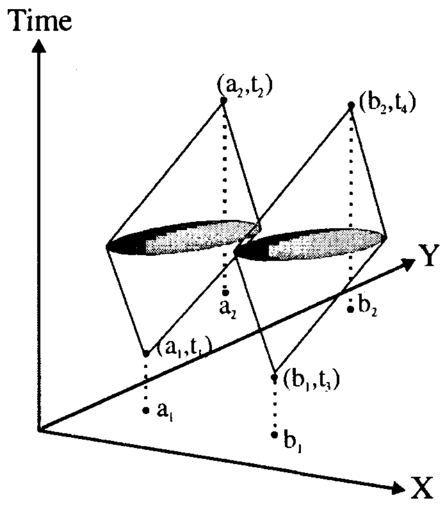


Figure 1.3 Suspects' movements modeled as lifeline beads.

1.3 Research Questions

To capture the continuous movement of physical objects, we can use discrete spatio-temporal samples at regular or irregular time intervals. The resolution of spatio-temporal data is very important for the application at hand. Environmental health applications need data with a resolution of minutes to years. These discrete samples give the location and time of movement only at the defined sample points. However, many research questions arise to answer the spatio-temporal whereabouts of an individual in between the sample points. The following queries are challenging research questions that the approach developed in this thesis can answer:

- What are all the possible geographic locations occupied by an individual at time *t*?
- Could two individuals have met?
- Could two individuals have met at time *t*?
- What are all the possible locations at which two individuals could have met?
- Was it possible that two individuals met at time *t* at location *s*?
- How many times would the two individuals have met?

To process such queries, one needs models and methods that can consider all possible space-time locations at which an individual could have been between two measured space-time samples.

1.4 Goal and Hypothesis

The goal of this thesis is to develop a conceptual and computational model for continuous movement of objects that enables spatio-temporal query. This model, called *geospatial lifelines* is aimed at representing the complete movement of an object traveling from point *A* to point *B*, provided there is a certain amount of knowledge about speed heuristics. The maximum speed at which an object moves between the origin and destination is critical to define the model precisely. Such a model would require a 3-dimensional representation, with *x* and *y* for the spatial extent and *z* for time. All possible locations between two consecutive sample points are captured by a 3-dimensional solid.

The possibility for individuals meeting in space and time is expressed by the geometric intersection of their corresponding solids.

Three-dimensional intersection of such solids is expected to be computationally expensive. Also, lack of 3D representations in current commercial systems forces the need for alternative representations. Hence, we resort to a set of operations on 2D models.

The hypothesis of this thesis is:

To determine whether two lifeline beads intersect, a finite set of operations in 2-dimensional space produces the same logical result as a 3-dimensional model of lifeline beads.

1.5 Scope of the Thesis

Our methodology is based on general assumptions. The lifeline bead concept is formed assuming the movement of an object in an isotropic medium, that is, movement is assumed to be the same in all directions. Constraints, such as movements in transportation networks, are not taken into consideration. The model works well even when it is constrained, but becomes more general. Another assumption is the knowledge of the maximum speed of an object's movement. A person can walk, or travel in a car or on a bike. The speed is not constant throughout the individual's movement. If, however, the maximum speed is taken into account, the variation in speed is not taken into account. This decreases the probability of an individual occupying a particular geographic location by exaggerating the positions occupied.

1.6 Organization of the Thesis

The remainder of this thesis is organized as follows:

Chapter 2 discusses the evolution of spatio-temporal models that have been proposed to represent moving objects. This chapter explains how the present GIS technologies can support research work on spatio-temporal databases.

Chapter 3 reviews the mathematics of cones that lay the foundation for representing the geospatial lifeline model explained in the subsequent chapters. Elements of solid geometry and plane geometry are used to explain the formation of cones and their sections.

Chapter 4 introduces the concept of lifeline beads and necklaces. Different types of beads and their formations are explained. How the bead model supports various query operations is discussed later in the chapter.

Chapter 5 deals with the intersection of lifeline beads, its significance, and computational models that are required for testing intersections. A novel method is proposed which can test the intersections of geospatial lifelines. Finally, the extensions of beads to necklaces, which are series of beads, are also discussed.

In chapter 6, the implementation of a prototype showing different kinds of beads, their intersections, and tests for intersections is discussed. Visual C++ with the OpenGL Application Programming Interface (API) is used as the implementation platform.

Chapter 7 summarizes the thesis with conclusions and scope for carrying out further research work in this area.

Chapter 2

Spatio-Temporal Models of Movement

Hägerstrand's (1970) Time Geography has been very influential for spatio-temporal modeling. His concept of space-time has provided the framework for today's work on spatio-temporal GIS using geometric approaches. Forer (1998) and Miller (1991) used this framework to develop concepts for space-time accessibility. Many of the concepts presented in this thesis have their roots in Hägerstrand's work. This chapter reviews relevant foundations in geography, GIS, and computer science.

2.1 Time Geography

The origin of spatio-temporal models has strong relevance with the socio-economic problems of people. The problems include the necessity for proper guiding and planning, improvements in the quality of human life, and the question of individual's survival in a complicated environment. Regional scientists constantly questioned the optimal livability of an individual in his or her environment considering such problems. This question led to research efforts to solve the problem of livability. Moving from one place to another was often sought as a solution to this problem, especially when the existing place became uncomfortable for living. The spatial distributions of needs, such as schools, educational facilities, universities, and libraries, are contributing factors for people's movements. Models of movement are largely dependent on the aggregate of people's activities in an environment, called *mass probabilistic behavior*, which in turn depends on the independent activities of a person, called the *micro-situation* of the individual. There may be many differences at the micro level depending on the range of organizational and social setup. It is also important to consider the addition of time to a spatial perspective, since the flow of time cannot be separated from the activities in space. Even though an individual performs a multitude of roles, the sequence of activities follows a time log and cannot be misplaced randomly; therefore, an individual must pass every point on the time scale, while being anywhere in space. To model changes in the socio-economic patterns, a time study of activities in space is required. Hägerstrand proposed a socio-economic web model based on this space-time concept. This model attempts to describe an individual's accessibility of space over time.

2.1.1 Space-time path

A *path* describes an individual's movement in space-time (Figure 2.1). The path is represented in space as a two-dimensional plane *XY*, and in time as a third dimension perpendicular to the plane. The movement can be described over years, weeks, or days leading to *a life path*, *a week path*, or *a day path*, respectively. These paths are controlled and influenced by constraints imposed by society, such as physical necessities or private and common decisions.

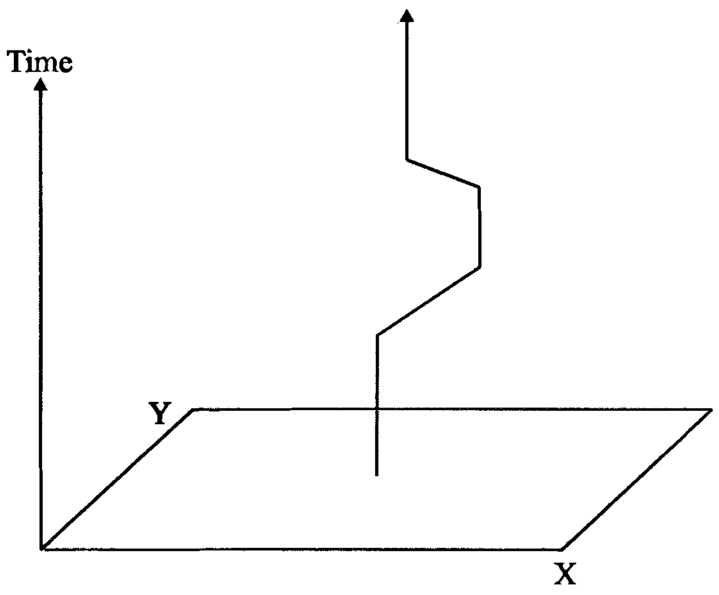


Figure 2.1 Space-time path.

2.1.2 Constraints to a life path

Hägerstrand discussed constraints to a life path that have been broadly classified into capability constraints, coupling constraints, and authority constraints. *Capability constraints* arise out of an individual's biological construction or the tools he or she can command or the distance that can be reached. This space-time is surrounded by concentric *tubes* of accessibility (Figure 2.2). The inner tube is the extent to which an individual's arm can reach. The second tube is the range of his or her communication. This could be the distance that an individual's eye or voice can reach. Telecommunications also forms a part of the second tube. Both these tubes originate from a fixed center, which is the individual's home or office.

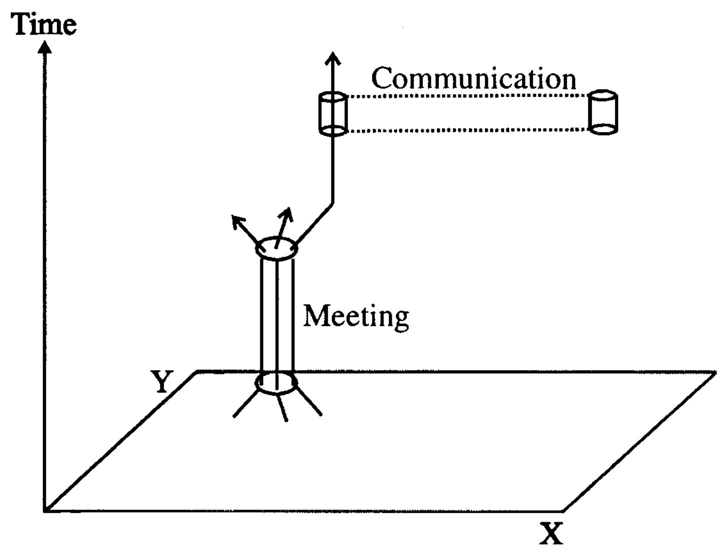


Figure 2.2 Tubes of accessibility showing an individual's ability to move and communicate.

When an individual moves from one place to another, the tube is no longer a cylinder, but becomes a *prism* (Figure 2.3). The extent of an individual's movement is obtained by projecting the prism in the *XY* plane (i.e., the space). The prism grows in size as an individual moves, and shrinks in size when the location becomes fixed. The path of an individual inside the prism is unbroken (i.e., there are no sudden jumps in an individual's movement) and there is no way an individual can perform a backward loop. *Coupling constraints* control the paths taken by an individual inside the prism. These constraints are where the individual stays, how long he or she stays, or who he or she meets. The third type of constraint is called *authority constraints*, which grant authority to domains that are space-time entities, such as township, nation, company, or university controlled by an individual or a group.

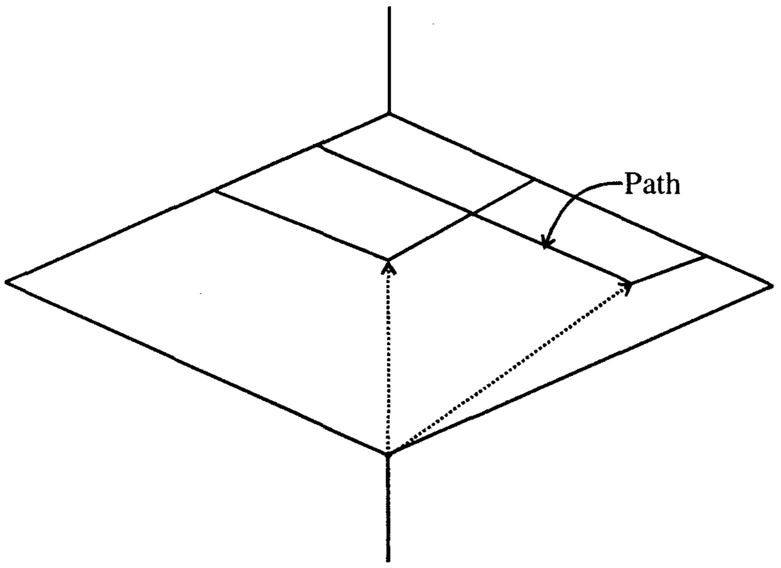


Figure 2.3 Prism showing unbroken paths inside, with arrows indicating their directions.

2.2. Geometric Approaches to the Space-Time Concept

The work of Lund School, particularly Hägerstrand's time geography gave a significant impetus in further developing the space-time model. Two large bodies of work were concentrating on incorporating the space-time concepts.

2.2.1 Metrics of space

The first dealt with defining metrics of space, which can transform physical distance into time separations by human actions. Lenntorp (1976) studied the application of space metrics within a particular context of measuring accessibility in an urban area. His model involved giving inputs to an urban network in the form of transportation characteristics, locations of activities, and hypothetical activity schedules. The output obtained was all the feasible activity schedules, which was regarded as the measure of urban accessibility. The measure of accessibility had to be obtained after involving all those tedious calculations of arriving at the output with the given inputs. This limitation mainly dismissed this body of work as the researcher's choice.

2.2.2 Space-time prism

Another body of work dealt with the geometric interpretation of space-time concepts, that is, a framework to describe and analyze the space-time activities of humans. The immediate outcome was a space-time prism or aquarium, a geometrical construct to represent Hägerstrand's space (two-dimensional) and time (third dimension). The space-time prism (Figure 2.4) determines the feasible set of all locations participated in by individuals in space within a given interval of time. The activities participated by an individual are represented as locations in space by two-dimensional *XY* planar axes, while the time interval of participation is represented by a third dimension, the Z-axis. The constraints needed to arrive at the dimensions of this prism are potential travel velocities, any stop time, etc. The projection of the prism onto the planar *XY* axes generates the activity space. The real-world meaning of this prism is that an individual starts from a location (x_0, y_0) (e.g., his or her work place) at time t_0 and arrives at the same location (x_0, y_0) at time t_1. It is assumed that the individual does not stop anywhere during his or her travel. During the travel time $t = t_1 - t_0$ the individual could have traveled anywhere within the radius defined by his or her travel velocity v. Unlike the metrics of space, which requires a huge amount of data as input, the interpretation of a prism requires only the time available for travel, the locations of origin and destination, and the velocity at which the travel occurs between origin and destination.

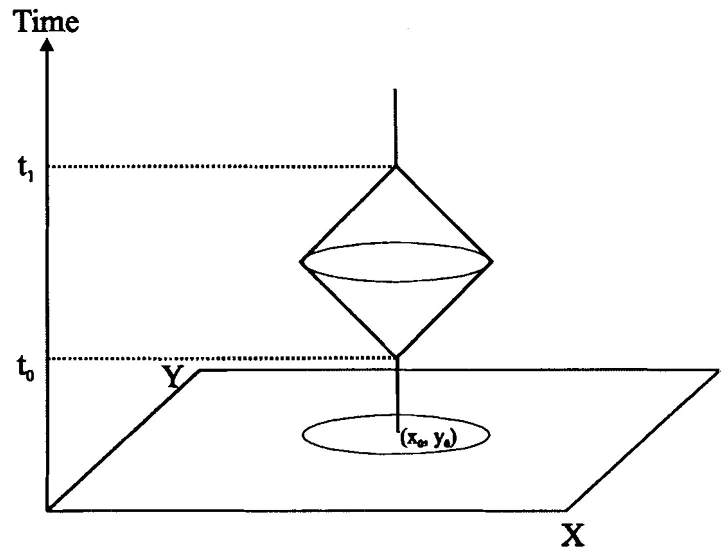


Figure 2.4 Space-time prism.

The concept of prism did not develop further, since the GIS technologies at that time did not offer capabilities to incorporate this concept. Some of the limitations include practical problems in generating a space-time prism for individuals, and the difficulty in obtaining individual's time budget. The space-time prism was revisited by the researchers during 1990s, as these limitations were solved by the emerging spatial technologies. Miller (1991) used this space-time prism to model an individual's accessibility to an environment. He called the prism a *potential path space* (PPS) and the action space as *potential path area* (PPA). Miller explored the requirements for deriving and using space-time prism constructs in a GIS and demonstrated the practical applications of such a derived construct. He proposed a network based on PPA as one of the space-time constructs in which he analyzes various space and time constraints involved in determining multiple paths between two points in a network. These constructs find an extensive application in infrastructure planning, travel modeling, and facility location. (Forer 1998) discussed the practical implementation of space-time concepts in the face of earlier drawbacks and suggested solutions with the available modern and sophisticated GIS technologies.

2.2.3 Geometric models for representing moving objects in space

The space-time prism construct is based on the simple assumption that an individual starts out at a location and comes back to the same location after a time interval without any intermediate stop. In this case the projections of the origin and destination are collocated in two-dimensional space. The set of movements of an object in between the time interval is defined by a circle (Figure 2.5a) with radius r determined by the maximum velocity v of the individual's movement. However, if there are any stops or if the origin and destination are not collocated in space, the geometry of the prism becomes complex (Miller 1991). In this case, the set of movements in space between the time intervals is defined by an ellipse (Moriera *et al.* 1999) (Figure 2.5b). The foci f_1 and f_2 of the ellipse respectively are the origin and destination of movement, while any line of length $l \leq 2a$, where a is the semi-major axis, connecting the foci within the ellipse is a possible trajectory of the object.

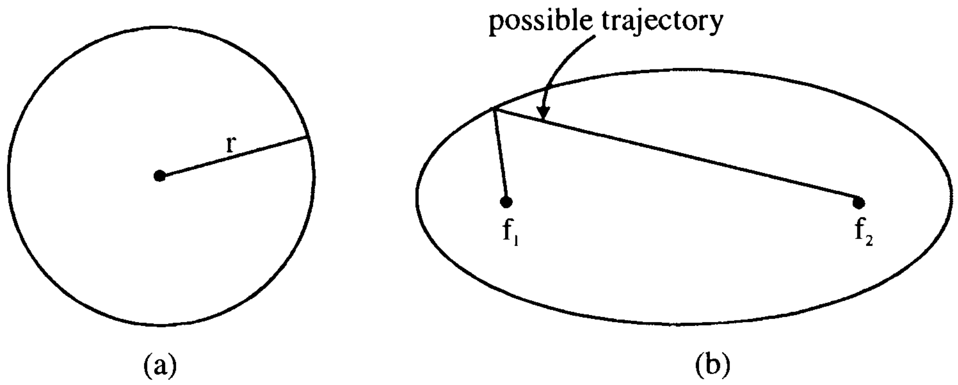


Figure 2.5 Trace of space path as (a) circle and (b) ellipse.

2.3 Types of Geographic Movements

A geographic movement occurs when the characteristics associated with a geographic phenomena change. These movements are accompanied both by changes in spatial and temporal dimensions. The objects in space are often represented as geometric features such as points, lines, areas, and volumes. Abler *et al.* (1971) classified the geographic movements based upon these geometric features. Their classification exhibited movements of similar spatial forms with the spatial dimensions of moves distinct from one another. This classification was helpful in understanding how things move in space, but required temporal characteristics to better understand the dynamics of movement.

Haggett (1990) classified temporal changes as constants, trends, cycles, and shifts. Constants and trends are long-term changes, while shifts and cycles are short-term changes. Time is conceptualized as continuous or mechanical time, and as discrete or body time. For example, seasons are a measure of continuous time, while seconds, minutes, and hours are measures of discrete time. The notion of intervals comes into play when time is viewed as a discrete phenomenon. It refers to how frequently an event occurs over a time interval. This phenomenon helps to capture and better understand the pattern or change in the underlying geographic process. Moriera *et al.* (1999) identified two types of moving objects based on their trajectories. *Free trajectories* have no or few restrictions in movement (e.g., movement of aircraft in space), while *constrained trajectories* (e.g., movement of vehicles in a road network) are based on restrictions in movement.

The values of spatial properties may change continuously or discontinuously over time (Yeh and Vermont 1992). When the changes are continuous, the plot of the spatial properties over time is also a continuous line. When the changes are discontinuous, the value changes instantaneously with time. This change is called an event recording. In between the time interval, the spatial property is either undefined or assigned a null value. The values are constant between two events, which define a time interval.

Classification of movements is typically based on spatial or temporal characteristics. For temporal modeling and simulation of dynamic geographic process, spatial changes must be captured over time (Langran 1992). Yattaw (1999) classified geographic movements based on both temporal and spatial characteristics. The classification is based on the fact that any spatially dynamic geographic process is conceptualized by its patterns and processes of movement through space and time. If a phenomenon moves uninterrupted for a period of time, then it is said to be continuous (e.g., monitoring a vehicle's movement using GPS). If the movement is discontinuous, it is said to be intermittent, since the motion fluctuates or stops periodically (e.g., seismic activities occurring intermittently). If the frequency of movement is periodic and regular, then the motion is cyclic (e.g., rotation of the earth around the sun). Any geographic phenomenon is typically described as points, lines, areas, and volumes. Combined with the temporal descriptors of movements (continuous, intermittent, and cyclic), the geographic movements are grouped into 12 classes. Out of 12 classes, *continuous point movements*, *cyclical point movements,* and *intermittent point movements* are relevant for our discussions in later chapters. Continuous point movements occur when point objects move continuously from beginning to end and uninterrupted through time. Examples are movements of a vehicle, person, or animal monitored using GPS. Cyclical point movements are those in which a point object moves with a regular or predictable frequency. Human migration patterns, multiple residences, and journey to work are examples concerned with cyclical point movements. Intermittent point movements are those infrequent movements of point objects across space without regular intervals. Examples of this movement are residential migration, immigration, leisure travel, and personal action space. Models capturing all the three types of point movements are discussed in later chapters.

2.4 Database Requirements for Moving Objects

Extending the treatment of moving objects in space to include time has been an important focus of recent database research (Sellis 1999). This involves topics relating to moving objects including developing conceptual models that describe how objects move in space and time (Miller 1991; Forer 1998) and designing database systems to handle queries about moving objects (Sistla *et al.* 1997a; Wolfson *et al.* 1999a; Pfoser *et al.* 2000; Forlizzi *et al.* 2000; Wolfson *et al.* 1999b). Other related work has focused on indexing moving objects trajectories (Nishida *et al.* 1998; Saltenis *et al.* 2000) and addressing uncertainties associated with moving objects (Sistla *et al.* 1997b; Pfoser and Jensen 1999). In particular, databases should capture the changing geometries over time so that they better represent the dynamic nature of moving objects. Spatio-temporal databases should facilitate efficient querying and representation of moving objects. Güting *et al.* (1998) defined abstract data types for moving points and regions, which enabled powerful querying of moving objects database. Abraham and Roddick (1999) discussed the essential functions that a spatio-temporal database should perform, such as inventory analysis, scheduling, updates, quality control, and display. Since spatio-temporal databases contain very large amounts of data, the need for developing appropriate indexing methods (Nascimento *et al.* 1999; Tzouramanis *et al.* 1998) for efficient access to moving objects databases is considered as another important database requirement. Constraint databases (Chomicki and Revesz 1997) are convenient ways of representing moving objects by capturing the position of each moving object as a function of time. This approach overcomes the problem of continuous update of objects' positions which overheads the database and updating the database at only given time instants which lowers the query performance.

2.5. Summary

With the advent of technology, GIS is now capable of supporting spatio-temporal modeling. While the research emphasis is more on integrating spatial and temporal databases, the spatio-temporal databases have concerns of their own. This problem requires better conceptual models to capture spatio-temporal scenarios and efficiently handle spatio-temporal databases. The space-time prism concept evolved as a strong foundation for representing moving objects. The literature review lays a strong emphasis on the fact that improved technologies could better incorporate this prism concept. This thesis builds on the prism concept for capturing moving objects. The following chapter discusses relevant foundations for the concepts introduced in the subsequent chapters.

Chapter 3

Mathematical Foundations for Lifelines

Solid geometry forms a strong foundation for the mathematical treatment of prisms. Since a prism represents planar coordinates in the *XY* plane and time in the *Z*-axis perpendicular to the *XY* plane, the sections of prisms form an important part of the computational analysis of space-time models. Since a prism inherits the geometric properties of a cone, the well-established concepts, construction, and properties of cones are discussed in this chapter. Equations representing cones in 3-dimensional coordinate space are useful foundations for discussing the concepts introduced in the subsequent chapters.

3.1 Cones

Let C be a given curve in a plane P and O a point that does not lie on P. The non-closed curve with a *vertex* O and *directrix* C is the surface obtained by the union of all lines that join O with points of C (Figure 3.1). The perpendicular distance h between point O and curve C gives the *altitude*. If C is a simple closed curve, then the term *cone* refers to the solid enclosed by the surface generated in this way. The line segment L generating the surface is called the *generatrix*.

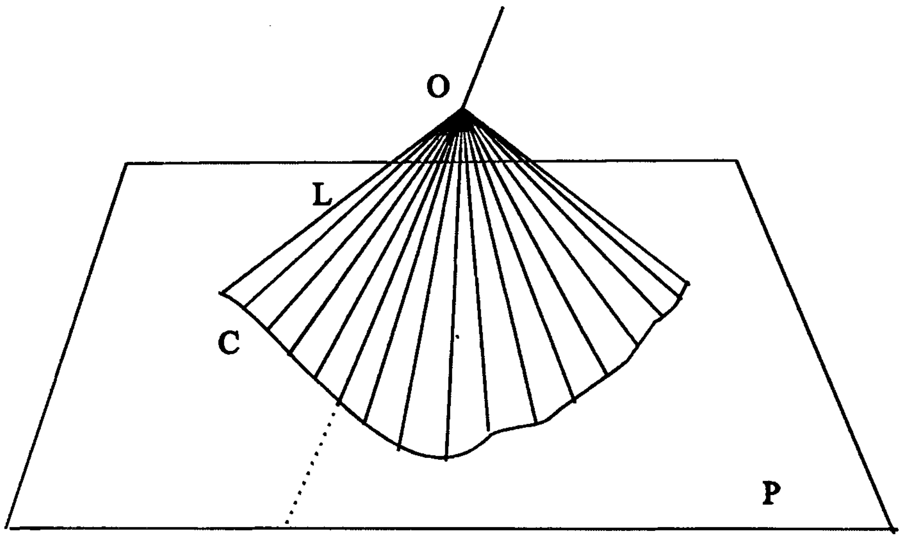


Figure 3.1 Cone.

If the curve C of the cone is a circle with radius r, then the cone is called a *circular cone* (Figure 3.2). The line segment joining the vertex of a cone to the center of its base is called the *axis* of the cone. A circular cone is further called a *right cone* (Figure 3.3a) or *oblique cone* (Figure 3.3b), depending upon whether the axis is respectively perpendicular or oblique to the base.

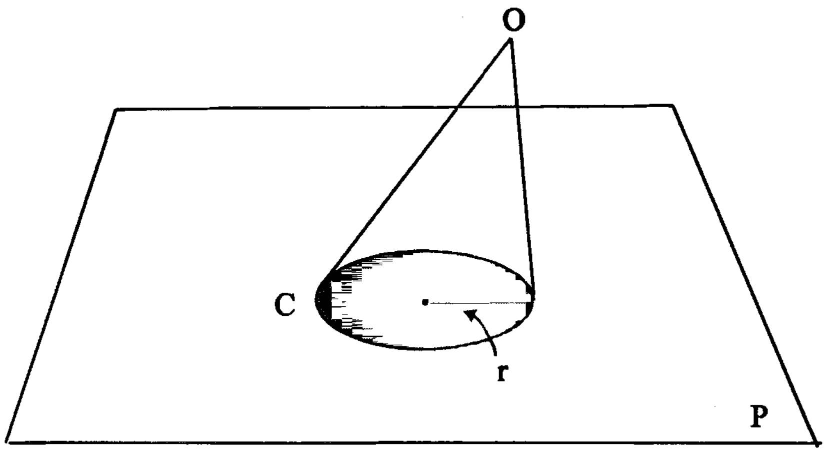


Figure 3.2 Circular cone.

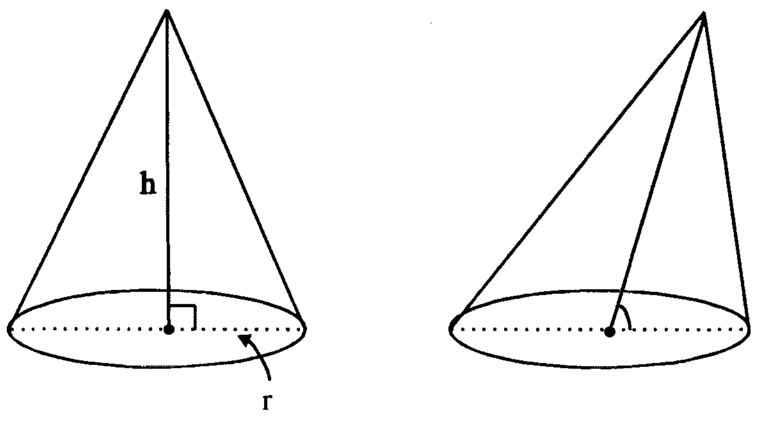


Figure 3.3 Two types of cones (a) right cone and (b) oblique cone.

If r is the radius of the base, h is the altitude, then the length l (Equation 3.1), lateral area A_l (Equation 3.2), total surface area A_s (Equation 3.3), and volume of the cone v (Equation 3.4) can be calculated.

$$l = \sqrt{r^2 + h^2} \tag{3.1}$$

$$A_l : \pi rl = \pi r\sqrt{r^2 + h^2} \tag{3.2}$$

$$A_s : \pi r(l + r) = \pi r(r + \sqrt{r^2 + h^2}) \tag{3.3}$$

$$v = \tfrac{1}{3}\pi r^2 h \tag{3.4}$$

Let us imagine a vertical line, and a second line intersecting it at some angle θ. We will call the vertical line the *axis*, and the second line the *generator*. The angle θ between them is called the *vertex angle*. Now imagine grasping the axis between thumb and forefinger on either side of its point of intersection with the generator, and twirling it. The generator will sweep out a surface, called a *double cone* (Figure 3.4). A double cone has an upper half and a lower half and they are joined at a single point, called the *vertex*.

A double cone is thus completely determined by its vertex angle. In all the subsequent discussions the term cone refers to a double cone.

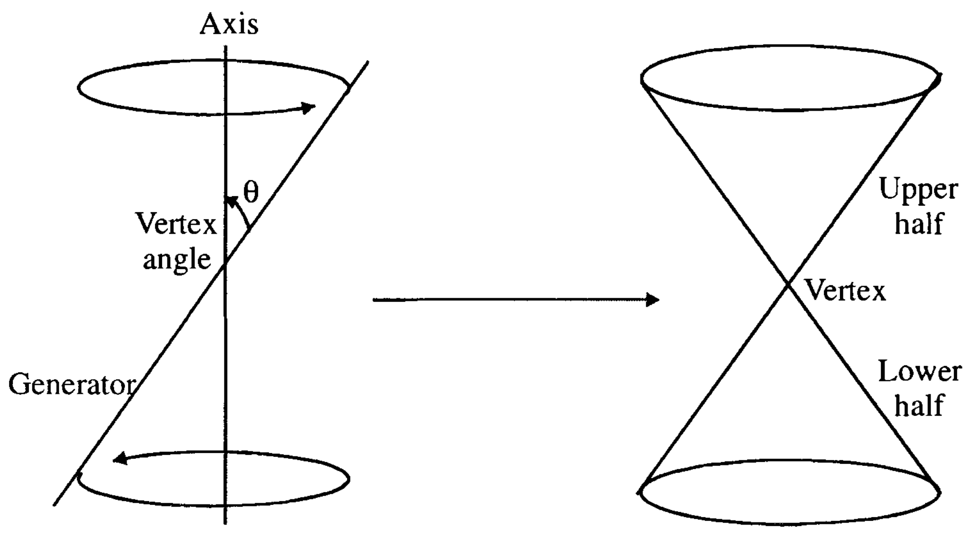


Figure 3.4 Double cone.

3.2 Sections of a Right Cone

When a right cone is cut by a plane parallel or oblique to its base plane, sections of the cone are obtained. The cut can be made by one or more planes along the length of the cone. If the cut is made by a single plane, the resulting curve formed by the intersection of the plane with the right circular cone (commonly represented as a double cone) is called a *conic section or conics.* Different types of conic sections such as ellipse, parabola, or hyperbola result, depending on the angle of the cutting plane relative to the cone. Conics may also be described as plane curves that are the paths (loci) of a point moving so that the ratio of its distance from a fixed point (the focus) to the distance from a fixed line (the directrix) is a constant, called the *eccentricity* of the curve. If the eccentricity is zero, the curve is a circle; if equal to one, a parabola; if less than one, an ellipse; and if greater than one, a hyperbola.

Special case occurs, if the intersecting plane has a greater angle to the vertical than does the generator of the cone. In this case the plane must cut right through one of the halfcones, thus resulting in a closed curve called an *ellipse* (Figure 3.5a). If the plane is perpendicular to the axis (i.e., horizontal) then the conic section is a circle (Figure 3.5b).

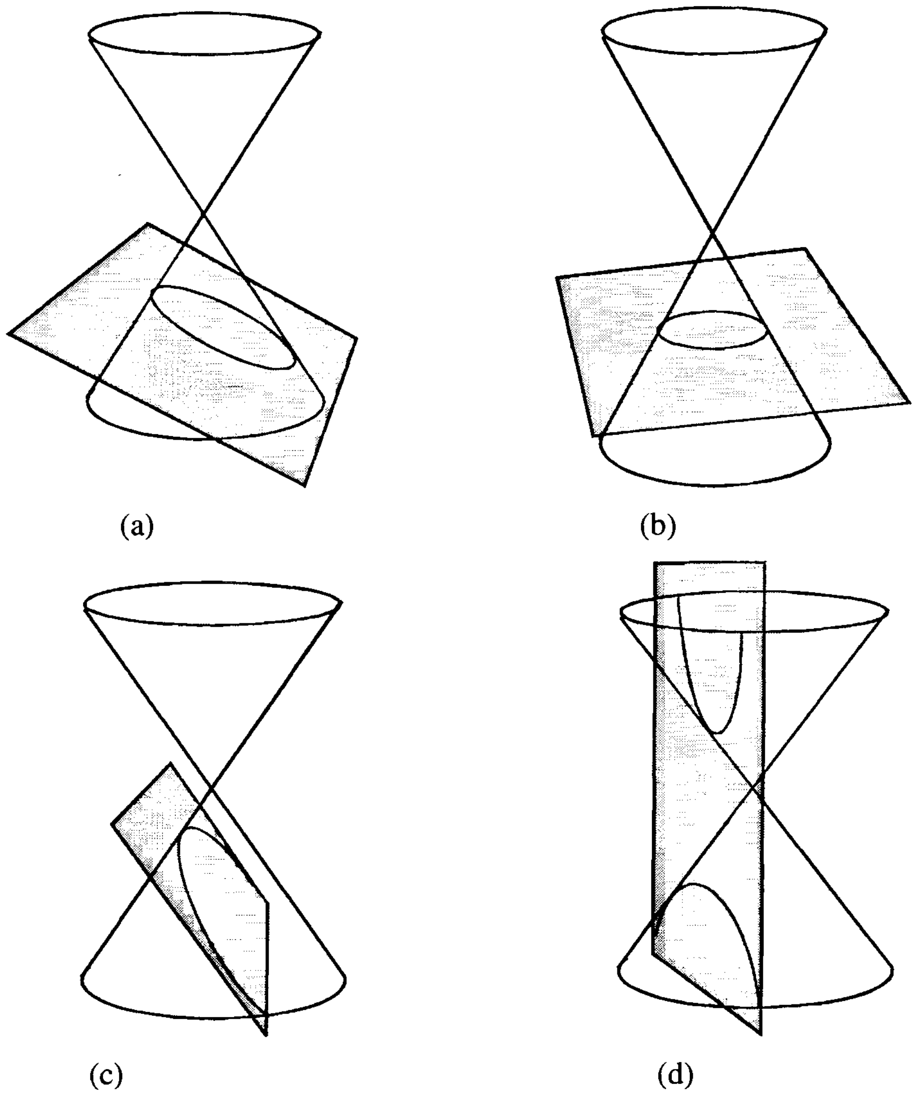


Figure 3.5 Conic sections: (a) ellipse, (b) circle, (c) parabola, and (d) hyperbola.

The algebraic representations of the conic sections may be derived by considering the plane geometry of these sections. The definition of an ellipse is given by the set of all points in the plane, the sum of whose distances from two fixed points, called the *foci*, is a constant. An ellipse has two axes of symmetry, the larger of which is the *major axis* and the smaller the *minor axis* (Figure 3.6). The two points at the ends of the ellipse on the major axis are called the *vertices*. The length of the major axis is equal to the sum of the distances from any point on the ellipse to its foci. If we call the length of the minor axis $2b$ and the distance between the foci $2c$, then the Pythagorean Theorem yields the relationship $b^2 + c^2 = a^2$. This leads to the standard equation for an ellipse centered at the origin with a and b as semi-major and semi-minor axes, respectively (Equation 3.5).

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{3.5}$$

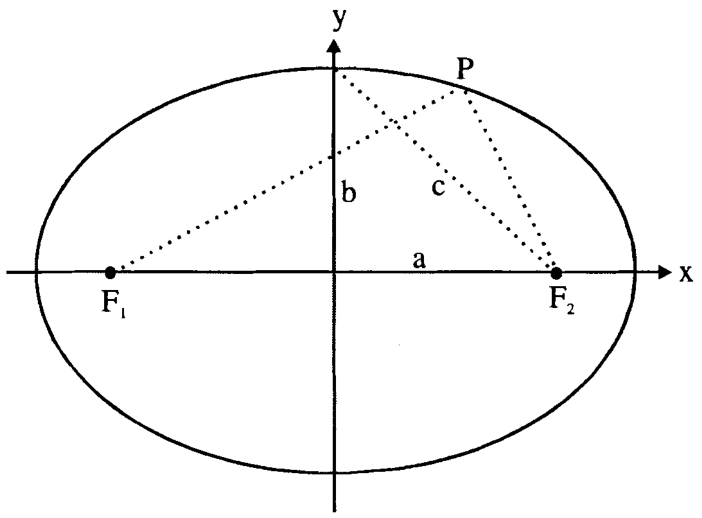


Figure 3.6 Standard ellipse.

3.3 Summary

This chapter reviewed the mathematics of cones in detail. Different classifications of cones are mentioned, out of which a right cone is used later. The sections of a right cone called *conics* are understood from the perspectives of both three-dimensional geometry and planar geometry. The properties and uses of these conic sections are better explained by their planar geometric definitions. The next chapter introduces two new concepts, called *a lifeline bead and a necklace*, which have the mathematical foundations explained in this chapter.

Chapter 4

Lifeline Necklaces and Beads

In this chapter, we explain the geometric construction of beads and the relevant mathematics for representing the beads. Example query operations that can be performed on the beads and necklaces are discussed in the later part of this chapter.

4.1 Lifeline Beads

A lifeline bead captures the set of all possible locations that an object can feasibly pass through when traveling from one location to another through a set of geometric constraints.

4.1.1. Formation of a bead

A bead is a three-dimensional geometric construct, formed by a non-empty intersection between a lower and an upper halfcone (Figure 4.1) constructed from two different cones. After the formation of a bead, the lower halfcone is referred to as the *upper halfbead* and the upper halfcone becomes the *lower halfbead.* The lower halfbead models the space-time points observed from a specified origin (x_0, y_0, t_0), while the upper halfbead captures the space-time points at which the individual could have been while approaching the second observation sample (x_1, y_1, t_1), where $t_0 \le t \le t_1$. The maximum speed v at which an individual travels determines the apex angle of both halfbeads.

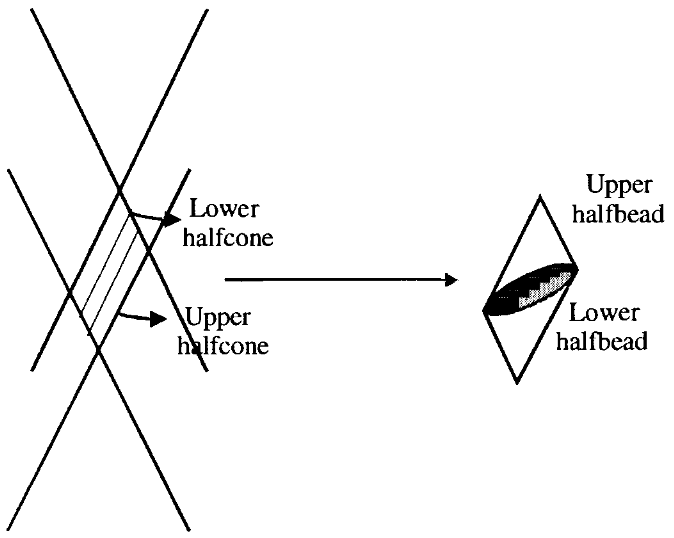
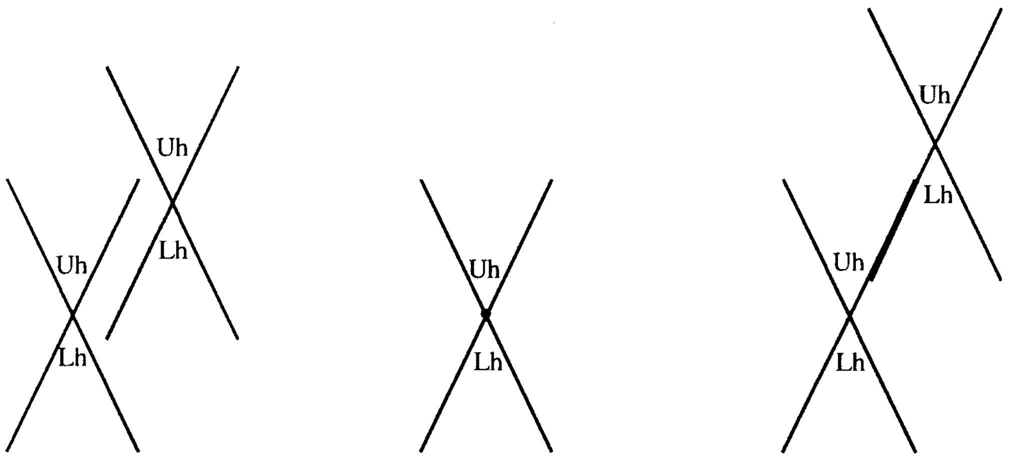


Figure 4.1 Bead formed from non-empty intersection between upper and lower halfcones.

An empty intersection between a lower and an upper halfcone leads to the formation of *degenerate beads*. Several types of degenerate beads occur:

- An *empty bead* results from disjoint halfcones (Figure 4.2a). The interpretation is that the individual could not reach the destination with the given travel speed.
- A *point bead* results from two halfcones that meet along a point (Figure 4.2b). The interpretation is that the point is the individual's sample point.
- A *line bead* results from two halfcones that meet along the rim (Figure 4.2c). The interpretation of this scenario is that the individual could reach the destination through the shortest path, traveling at the maximum speed.



Uh = Upper halfcone Lh = Lower halfcone

(a) (b) (c)

Figure 4.2 Degenerate beads: (a) empty bead, (b) point bead, and (c) line bead.

4.1.2. Formalization of beads

If (x_0, y_0, t_0) is the origin of the bead, (x_1, y_1, t_1) is the destination of the bead, and θ is the apex angle of the upper or lower halfbead, then the bead is represented as the intersection of a lower halfbead (Equation 4.1a) and an upper halfbead (Equation 4.1b).

$$(x-x_0)^2+(y-y_0)^2 \le \tan^2\theta(t-t_0)^2 \qquad (4.1a)$$

$$(x-x_1)^2+(y-y_1)^2 \le \tan^2\theta(t_1-t)^2 \qquad (4.1b)$$

However, it is appropriate to represent the equations of halfcones in terms of travel speed v, since speed heuristics of travel are often known (Equations 4.2a and 4.2b).

$$(x-x_0)^2+(y-y_0)^2 \le v^2(t-t_0)^2 \qquad (4.2a)$$

$$(x-x_1)^2+(y-y_1)^2 \le v^2(t_1-t)^2 \qquad (4.2b)$$

Four abbreviations are used in the subsequent sections for describing beads: the time difference between the destination and origin, referred to as Δt_{10} (Equation 4.3a), the x- and y-differences between destination and origin, referred to as Δx_{10} (Equation 4.3b) and Δy_{10} (Equation 4.3c), respectively, and the planar distance between the origin and destination, referred to as d_{01} (Equation 4.3d).

$$\Delta t_{10} = t_1 - t_0 \tag{4.3a}$$

$$\Delta x_{10} = x_1 - x_0 \tag{4.3b}$$

$$\Delta y_{10} = y_1 - y_0 \tag{4.3c}$$

$$d_{01} = \sqrt{\Delta^2 x_{10} + \Delta^2 y_{10}} \tag{4.3d}$$

4.1.3. Right beads

Depending on the origin and destination locations, two types of beads result. If the origin and destination are at the same location (i.e., the two consecutive samples are (x_0, y_0, t_0) and (x_0, y_0, t_1)), then the apexes of a bead are collocated in space, but shifted in time, forming a *right bead* (Figure 4.3). Since the two apex angles from opposite sides are identical, the intersection between two halfcones forms a circle located in the plane $t = \frac{(t_0 + t_1)}{2}$, around the center $\left(x_0, y_0, \frac{(t_0 + t_1)}{2}\right)$, and with a radius $r = \frac{t_1 - t_0}{2} \tan\phi$ (Equation 4.4).

$$(x - x_0)^2 + (y - y_0)^2 = \left(\frac{t_1 - t_0}{2}\right)^2 \tan^2\phi \tag{4.4}$$

The radius of a right cone taken at any time t_i between t_0 and t_1 depends on the location of t_i—either in the lower halfcone (Equation 4.5) or in the upper half cone (Equation 4.5b).

$$r_i = (t_i - t_0)\tan\phi \ \ if \ \ t_0 \le t_i \le (\frac{t_0 + t_1}{2}) \qquad (4.5a)$$

$$r_i = (t_1 - t_i)\tan\phi \ \ if \ \ (\frac{t_0 + t_1}{2}) \le t_i \le t_1 \qquad (4.5b)$$

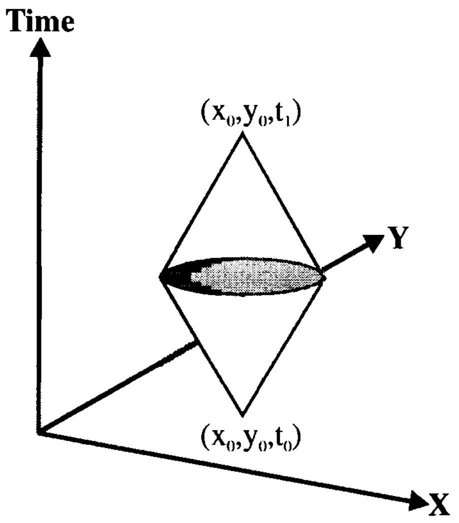


Figure 4.3 Right bead.

4.1.4. Oblique beads

If a bead's origin and destination are at different locations (i.e., the two consecutive space-time samples are x_0, y_0, t_0 and x_1, y_1, t_1 such that $x_0 \neq x_1$ or $y_0 \neq y_1$), then the apexes of the bead are not spatially aligned, forming an *oblique bead* (Figure 4.4). Since the apex angles reflect the maximum travel speed, they must be large enough to cover the distance from the origin to the destination (i.e., $\phi > \arctan(\frac{d_{01}}{\Delta t_{01}})$). Otherwise, the set of all possible movements collapses into a single line if the apex angle (i.e., the travel speed) offers exactly one path to travel the distance d_{01} within time Δt_{01} ($\phi = \arctan(\frac{d_{01}}{\Delta t_{01}})$) (see line bead in Section 4.1.1). If the apex angles are smaller ($\phi < \arctan(\frac{d_{01}}{\Delta t_{01}})$), there cannot exist a movement within the given constraints, because it would be impossible to travel the distance d_{01} to reach the destination within the allotted time Δt_{01} (see empty bead in Section 4.1.1). All subsequent considerations exclude the degenerate beads.

For an oblique bead, the two halfbeads intersect in a classic section— an ellipse— that is located in the plane orthogonal to the line from the origin (x_0, y_0, t_0) to the destination (x_1, y_1, t_1) and runs through the bead's center $\left(\frac{(x_0+x_1)}{2}, \frac{(y_0+y_1}{2}, \frac{(t_0+t_1)}{2}\right)$. The projection of an oblique bead into the plane $t = t_n$ also results in an ellipse whose foci are the projections of the origin (x_0, y_0, t_n) and (x_1, y_1, t_n). The ellipse's major axis (Equation 4.6a) and minor axis (Equation 4.6b) depend on the apex angle ϕ, the time difference Δt_{01}, and the planar distance d_{01} from origin to destination.

$$a = \frac{1}{2}\Delta t_{01} \tan\phi \qquad (4.6a)$$

$$b = \frac{1}{2}\sqrt{a^2 - d_{01}{}^2} \qquad (4.6b)$$

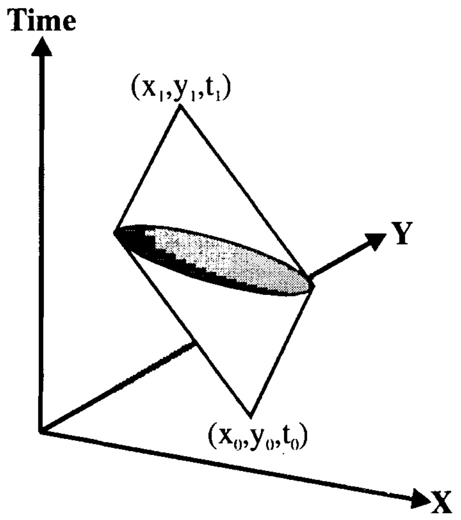


Figure 4.4 Oblique bead.

4.2 Lifeline Necklaces

Multiple space-time samples taken for the same object lead to a connected sequence of lifeline beads, called a *lifeline necklace* (Figure 4.5). The sequence of beads in a necklace reveals more details about the path of movement. For example, different travel speeds of an individual could be determined from the beads, since each bead differs in its dimensions because of different travel speed in that portion of movement. Three temporally consecutive space-time samples (x_0, y_0, t_0; x_1, y_1, t_1; and x_2, y_2, t_2 with $t_0 < t_1 < t_2$) form two consecutive lifeline beads, b_i and b_j, such that b_i's destination is the same as b_j's origin. The bead b_j is then called the successor of b_i. Except for the first and last bead of a necklace, each bead within a necklace has exactly one successor (Equation 4.7a) without any gap or overlap between them. At the same time, non-consecutive lifeline beads of the same lifeline necklace cannot meet or overlap (Equation 4.7b).

$$\forall b_i, \exists b_j : \mathit{successor}(b_i, b_j) \ \mathit{with} \ b_i \neq b_j \tag{4.7a}$$

$$\forall b_i, b_j, b_k : b_i \cap b_k = \varnothing \wedge b_j \cap b_k = \varnothing \ \mathit{with} \ b_i \neq b_k \wedge b_j \neq b_k \wedge \mathit{successor}(b_i, b_j) \tag{4.7b}$$

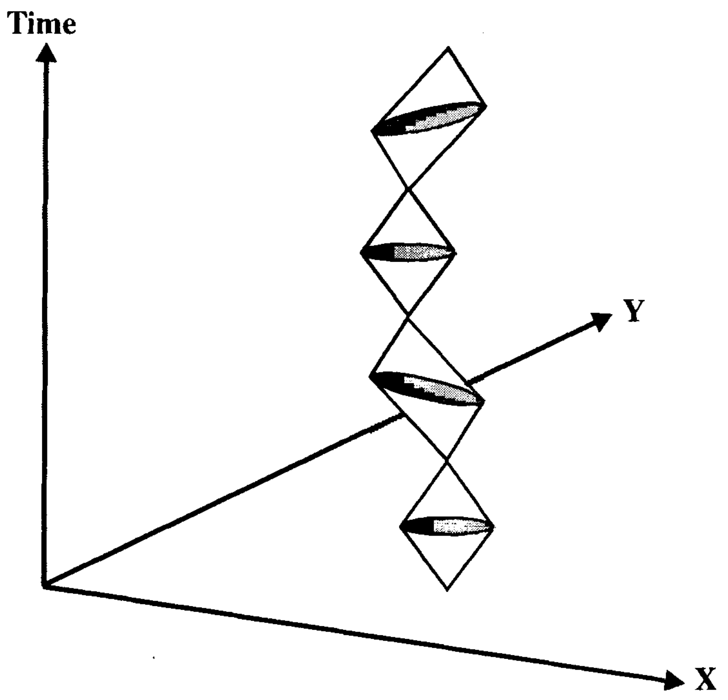


Figure 4.5 Lifeline necklace.

4.3. Query Operations on Beads and Necklaces

The necklace and bead model provides a computational foundation for performing queries on moving objects. Operations on a necklace (i.e., individual lifeline) are undertaken by analyzing whether their component beads satisfy certain geometric constraints. The queries can be over an individual bead, or a pair of beads, or an individual lifeline necklace, or a pair of lifeline necklaces. We illustrate each case with an example in the following discussions.

Example 1:

Query operation on a single bead. When could *A* have been at location *s*?

This query is aimed at finding the time(s) or time intervals during which an individual *A* could have occupied location *s*. First, we locate *s* in the *XY* plane and then extend a vertical line from the *XY* plane, parallel to the time axis. The intersection of this vertical line with the bead gives the time(s) or time interval during which *A* could have occupied position *s*. An empty intersection (Figure 4.6) means that *A* did not occupy position *s*. A non-empty intersection means that *A* could have just occupied *s* at time *t* (Figure 4.7a), or that *A* could have occupied *s* at times t_1 and t_2 or *A* could have occupied *s* for the entire time interval between t_1 and t_2 (Figure 4.7b).

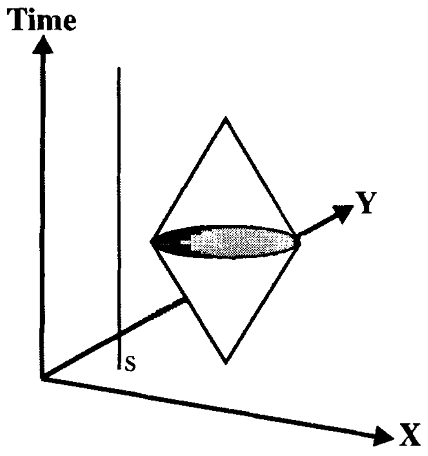


Figure 4.6 Empty intersection between bead and vertical line.

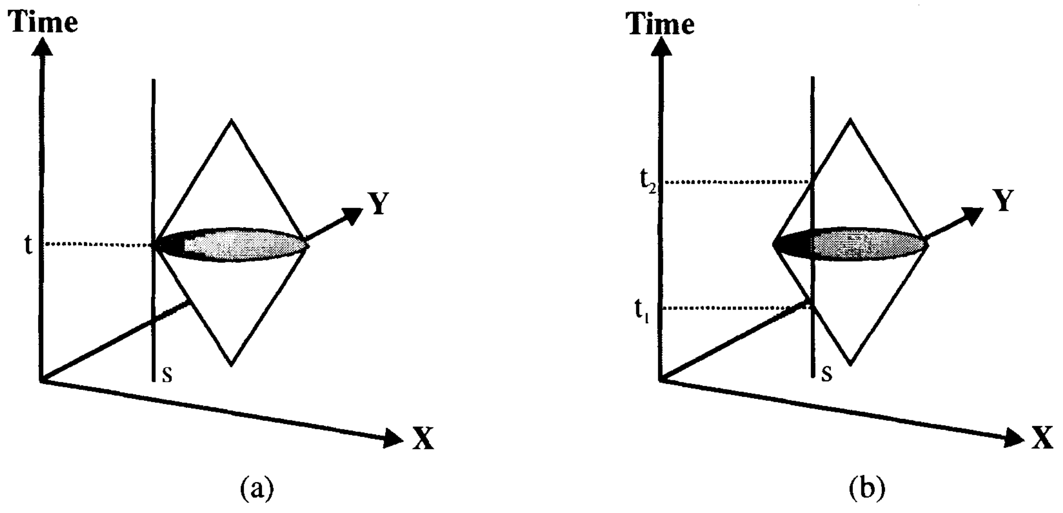


Figure 4.7 Non-empty intersection between bead and vertical line: (a) vertical line and bead *touch* each other and (b) vertical line and bead *overlap* with each other.

Example 2:

Query operation on a pair of beads. Is it possible that individuals *A* and *B* met?

This query involves testing the intersection of two beads corresponding to the movement of individuals *A* and *B*. If the beads intersect, then these individuals could have met. The common likely geographic location where *A* and *B* could have met is the projection of the intersecting portion of the beads on the *XY* plane, denoted by *s*, and the corresponding time interval is t_1 and t_2 (Figure 4.8). If the individuals did not meet *s* is empty; *s* is a point if the individuals could have met at some location for a moment; and *s* is a line if the individuals could have met for a continuous period of time while traveling together.

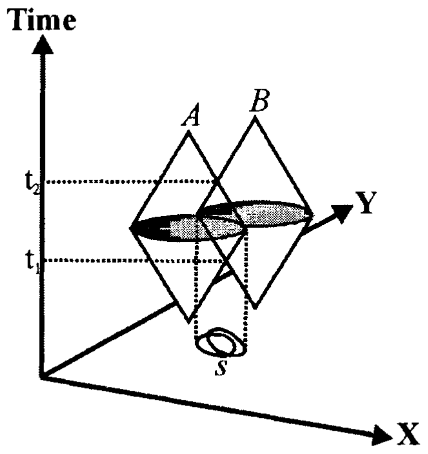


Figure 4.8 Intersection between a pair of beads.

Example 3:

Query operation on a single necklace. At what time(s) could *A* have been at location *s*?

The discussions for the query operations on a single bead also apply for the query operation on a single necklace, except that the vertical line parallel to the time-axis may intersect with more than one bead (Figure 4.9).

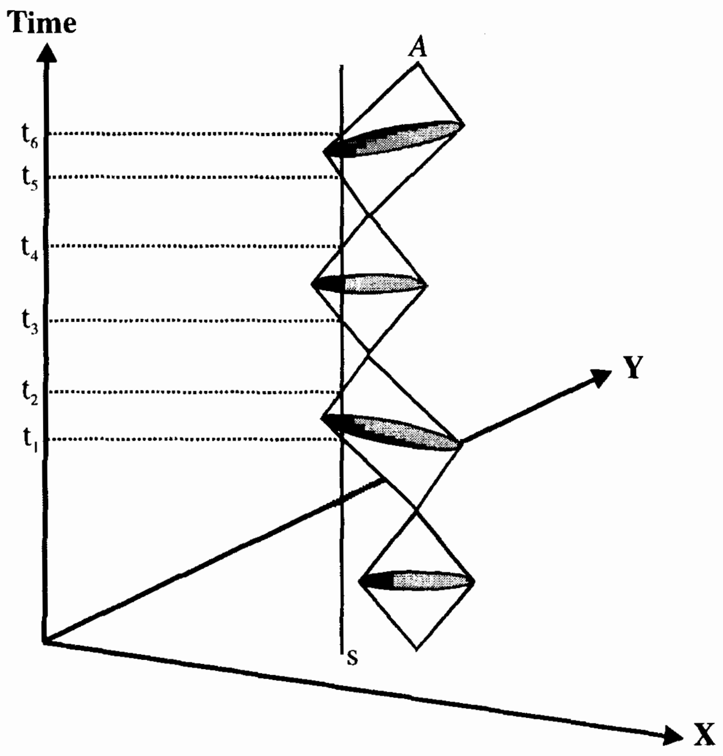


Figure 4.9 Intersection between vertical line and necklace.

Example 4:

Query operation on a pair of necklaces. How frequently could *A* and *B* have met?

A and *B* are two necklaces with their component beads intersecting with each other at various points along the time axis (Figure 4.10). The maximum number of times that *A* and *B* could possibly meet during the time intervals $t_1 - t_2, t_3 - t_4$, and $t_5 - t_6$ is 3. Hence, the frequency of meeting between *A* and *B* could be any number between 0 and 3.

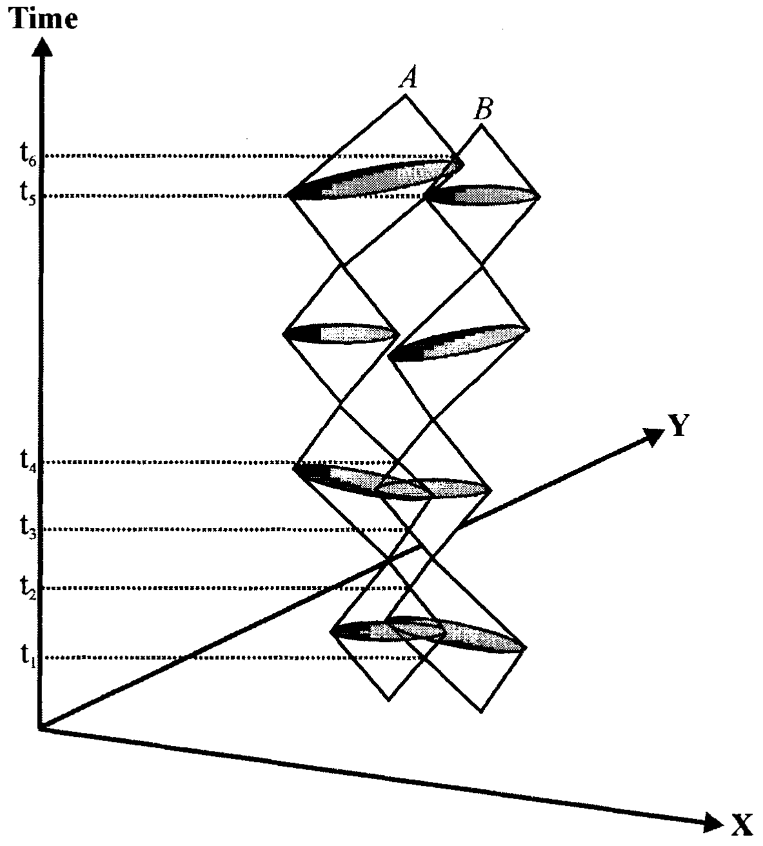


Figure 4.10 Query operation on a pair of necklaces.

4.4 Summary

This chapter introduced lifeline beads as building blocks for necklaces, which model the space-time movements of objects. The mathematical equations describing the beads are explained and example queries that can be performed over geospatial lifelines are presented along with a description of how to satisfy the query using a bead and necklace model. The next chapter discusses the analyses of bead(s) (from single or multiple lifelines) to provide computational solutions to the various queries discussed in Section 4.3 of this chapter.

Chapter 5

Intersections of Lifeline Beads and Necklaces

It is common that people moving in geographic space meet each other, travel together, or stay together. During these encounters people share a common geographic space and frequently occupy this space at the same time. In the computational model of lifelines this means that the geospatial lifelines of these individuals intersect. The foundation for this kind of analysis is the calculation of the intersection of two beads. This chapter introduces the computational model to solve the intersection of beads.

5.1 Relations between Lifeline Necklaces

Modeling multiple geospatial lifelines as necklaces can capture certain topological relations between the lifelines that cannot be described with lifeline steps or threads (see Figure 1.2). For example, the set of possible geographic locations where two individuals could have met at time *t* can be determined by solving the intersections of beads containing time *t*, whereas the lifeline step or thread model returns only a single most likely location.

The intersection of beads falls into one of the categories of relations between two objects modeled with 9-intersection model (Egenhofer 1991). These categories of relations such as disjoint, meet, overlap, equal, contains, inside, covers, and covered_by offer rich semantics in interpreting the relations among individuals moving in space-time. For example, the movement of two individuals *A* and *B* traveling together for some time can be modeled as two beads *touching* each other (Figure 5.1) All these categories of relations reduce to testing whether two beads intersect or not.

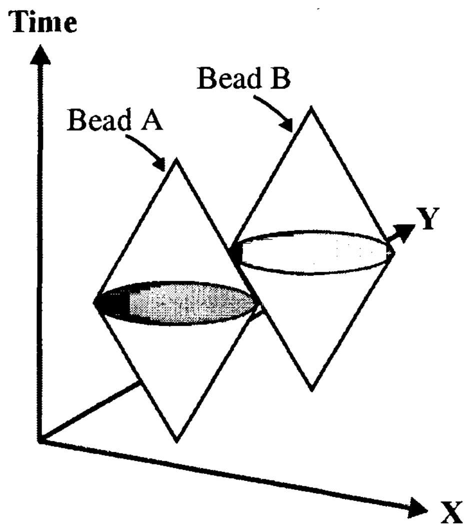


Figure 5.1 Beads *touching* each other.

Intersection of lifeline beads can occur between,

- two right beads (Figure 5.2a),
- two oblique beads (Figure 5.2b), and
- a right bead and an oblique bead (Figure 5.2c)

Solving the intersection of two lifeline beads through analytical operations on the equations (Equations 4.1a and 4.1b) of the four constituent halfbeads is computationally expensive, since each equation for a halfbead is a quadric, whose intersection would require the solution of equations of degree 2. To avoid this we use an approach that involves only the extrema of a bead and, therefore, provides a finite, discrete model for the calculation of bead intersections.

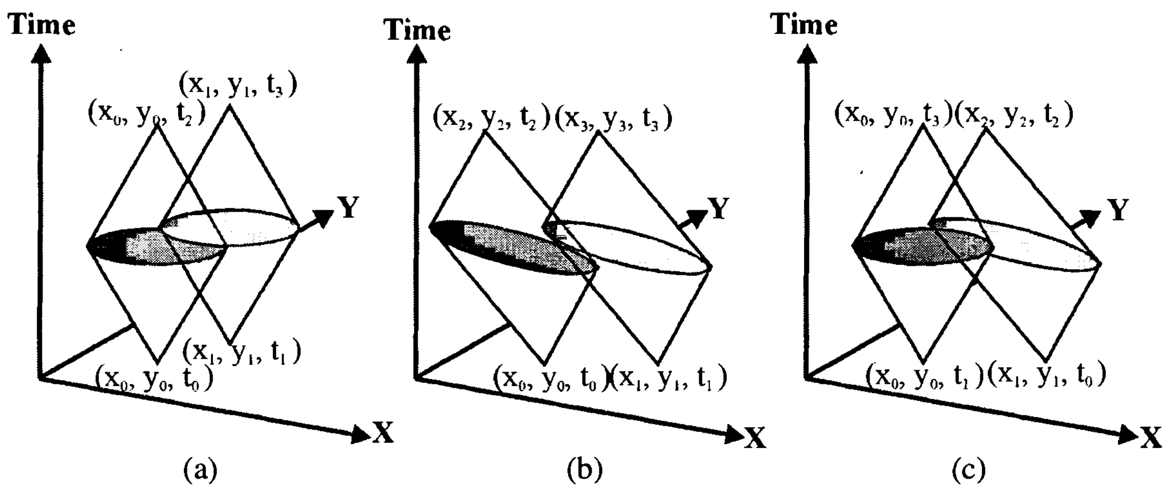


Figure 5.2 Intersections of lifeline beads between (a) two right beads, (b) two oblique beads, and (c) a right bead and an oblique bead.

5.2 Intersection Algorithms

This section introduces necessary algorithms that are relevant for testing intersection of two beads.

5.2.1 Extrema of beads

A right bead has three extrema (Figure 5.3a): the lower apex $\Omega_l = (x_0, y_0, t_0)$, the upper apex $\Omega_u = (x_0, y_0, t_1)$, and the directrix Γ at $z = \frac{(t_0 + t_1)}{2}$. Similarly, an oblique bead contains four extrema (Figure 5.3b): the lower apex $\Omega_l = (x_0, y_0, t_0)$, the upper apex $\Omega_l = (x_1, y_1, t_1)$, the lower directrix Γ_l at $z = \frac{(t_1 + t_2)}{2} - \frac{d_{01}}{2\tan\phi}$ (Equation 5.1a), and the upper directrix Γ_u at $z = \frac{(t_1 + t_2)}{2} + \frac{d_{01}}{2\tan\phi}$ (Equation 5.1b).

$$(x - x_0)^2 + (y - y_0)^2 = \frac{\Delta^2 t_{01}}{4} \tan^2 \phi \tag{5.1a}$$

$$(x - x_0)^2 + (y - y_0)^2 = \frac{\Delta^2 t_{01}}{4} \tan^2 \phi \tag{5.1b}$$

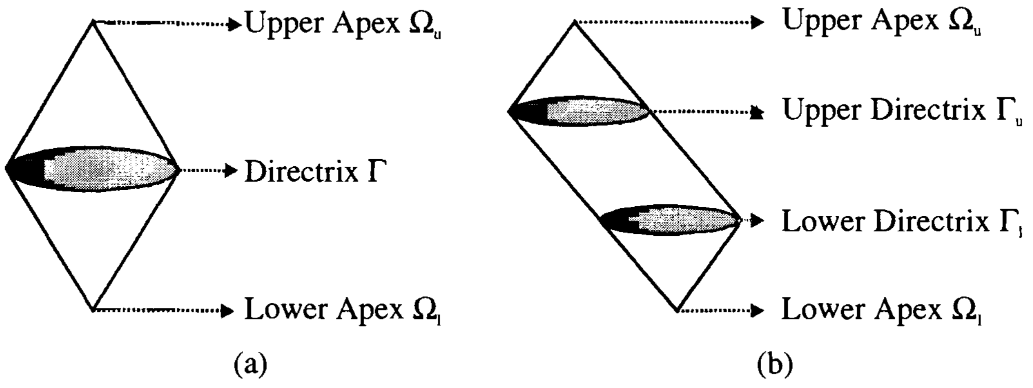


Figure 5.3 Extrema of a lifeline bead (a) for a right bead with its lower and upper apex and its directrix; and (b) for an oblique bead its lower and upper apex and its lower and upper directix.

These extrema completely describe a bead, such that it can be constructed uniquely, because a bead is a convex hull and the extrema contain all the points necessary to form exactly that convex hull. Right beads and oblique beads are formed from their extrema in a similar way. A right bead is formed by constructing a set of straight lines from the lower apex Ω_l to the directrix Γ, and another set of straight lines from the upper apex Ω_u to Γ. An oblique bead involves four sets of straight lines: (1) from Ω_l to the lower directrix Γ_u, (2) from Ω_l to the upper directrix Γ_u, (3) from Ω_u to Γ_u, and (4) from Ω_u to Γ_l. Testing the intersection of beads can be reduced to an evaluation of whether any of the extrema intersects with the target bead or not. To determine whether an intersection exists it is sufficient to find a single case of an intersection between an extremum and the opposite bead, whereas determining that no intersection exists requires the verification of all extrema.

Given the different extrema for right and oblique beads, there are three different scenarios for determining the intersection of two lifeline beads:

- the intersection of two right beads A and B, involving two sets of three elements each, $\{\Omega_l(A), \Gamma(A), \Omega_u(A)\}$ and $\{\Omega_l(B), \Gamma(B), \Omega_u(B)\}$;
- the intersection between a right bead A and an oblique bead B involving the comparison of $\{\Omega_l(A), \Gamma(A), \Omega_u(A)\}$ with $\{\Omega_l(B), \Gamma_l(B), \Gamma_u(B), \Omega_u(B)\}$; and
- the intersection between two oblique beads A and B, based on two sets of $\{\Omega_l(A), \Gamma_l(A), \Gamma_u(A), \Omega_u(A)\}$ and $\{\Omega_l(B), \Gamma_l(B), \Gamma_u(B), \Omega_u(B)\}$.

5.2.2 Tests for intersections using slices

A test for intersections between two beads needs to determine whether any of *A*'s extrema are located inside the target bead *B*, or *vice-versa*. Each extremum $\varepsilon(A)$ has its corresponding time t_ε. If $t_\varepsilon(A)$ lies outside of *B*'s bounds along the z-axis, then the two beads cannot intersect. Otherwise, one constructs a horizontal *slice* (i.e., the intersection of bead *B* with a plane $z = t_\varepsilon(B)$) (Figure 5.4) and tests whether this slice intersects $\varepsilon(A)$. The geometry of a slice through a bead can take three shapes, depending on the time at which the bead is sliced (*A* and *B* can be systematically exchanged to obtain the cases for testing *B*'s extrema). A slice is

- a point in the case $t_\varepsilon(A) = t(\Omega_l(B))$ or $t_\varepsilon(A) = t(\Omega_u(B))$;

- a circle, if B is either a right bead and $t(\Omega_l(B)) < t_\varepsilon(A) < t(\Omega_u(B))$, or if B is an oblique bead and $t(\Omega_l(B)) < t_\varepsilon(A) < t(\Gamma_l(B))$ or $t(\Omega_u(B)) < t_\varepsilon(A) < t(\Gamma_u(B))$; or
- a lens, as the intersection of two circles if B is an oblique bead and $t(\Gamma_l(B)) < t_\varepsilon(A) < t(\Gamma_u(B))$

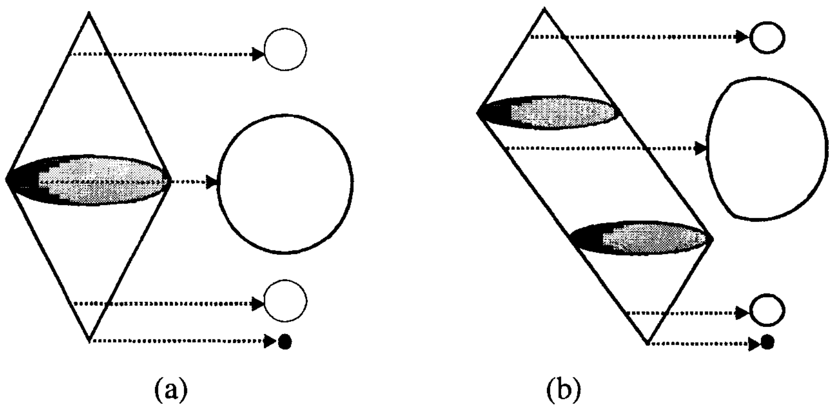


Figure 5.4 Slices for (a) a right bead forming points or circles and (b) an oblique bead forming points, circles, or lenses.

With two types of extrema (points and circles) and three types of slices (points, circles, and lens) there are five unique types of geometric operations to test intersections (Figure 5.5).

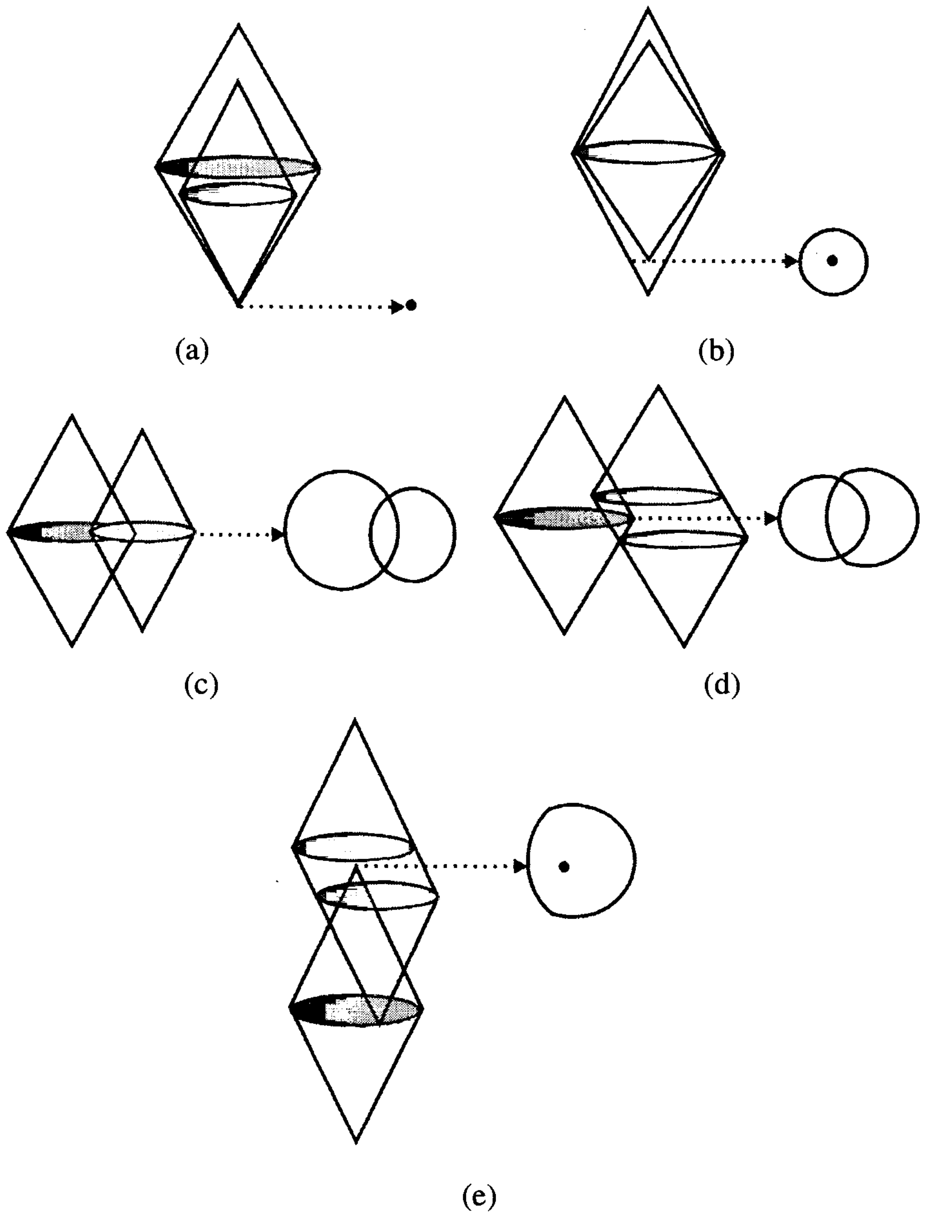


Figure 5.5 The intersections between two beads can be reduced to the intersection between (a) two points, (b) a point and a circle, (c) two circles, (d) a circle and a lens, and (e) a point and a lens.

- A point P_0 intersects with another point P_1 if the distance between them is zero (Equation 5.2)

$$P_0 \cap P_1 \;\; if \;\; d(P_0, P_1) = 0 \tag{5.2}$$

- A point P_0 intersects with a circle C_1 if the distance from P_0 to the center of C_1 is less than or equal to the circle's radius (Equation 5.3). This test also applies to the reverse case, the intersection of a circle with a point P_0.

$$P_0 \cap C_1 \;\; if \;\; d(P_0, center(C_1)) \le r(C_1) \tag{5.3}$$

- A point P_0 intersects with a lens L_1 if the distance from P_0 to the centers of each of the lens's circles, C_{10} and C_{11}, is less than or equal to the circles' corresponding radii (Equation 5.4)

$$\begin{aligned} P_0 \cap L_1 \;\; if \;\; & d(P_0, center(C_{10})) \le r(C_{10}) \;\; and \\ & d(P_0, center(C_{11})) \le r(C_{11}) \end{aligned} \tag{5.4}$$

- A circle C_0 intersects with another circle C_1 if the distance between the two circle's centers is less than the sum of the circle's radii (Equation 5.5).

$$C_0 \cap C_1 \;\; if \;\; d(center(C_0), center(C_1) \le r(C_0) + r(C_1) \tag{5.5}$$

- A circle C_0 intersects with a lens L_1 (formed by the intersection of circles C_{10} and C_{11}) if the area of the triangle *ABC* (Figure 5.6) formed by the centers of circles C_0, C_{10}, and C_{11}, as vertices should be less than or equal to the sum of the area of triangles *AaC*, *CcB*, and *BbA* (Equation 5.6). Since the lens must be non-empty, the distances between the centers of C_{10} and C_{11} must be less than the sum of radii around C_{10} and C_{11}.

$$C_0 \cap L_1 \; if \; area(\Delta ABC) \le (area(\Delta AaC) + area(\Delta CcB) + area(\Delta BbA)) \tag{5.6}$$

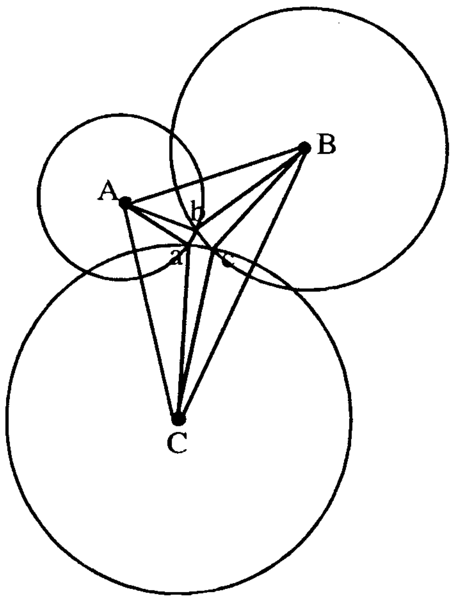


Figure 5.6 Intersection of a circle C_0 with center C and a lens formed by circles C_{10} and C_{11} with centers A and B respectively.

A bead-intersection algorithm (Figure 5.7) iterates over a reference bead's extrema and determines for each extremum the slice through the target bead taken at the same z-value, and then tests whether the extremum intersects with the geometric figure of that slice.

```
algorithm testExtremaWithBead (extrema, bead): Boolean
input: The set of extreme parts of a bead (extrema) and a bead.
output: True if at least one of the extreme parts intersects with the bead, otherwise false.
method:
    intersect := false
    for each $e \in$ extrema do
      if not intersect then
        s := slice (bead)
        if type (e) = point and type (s) = point then intersect := distance (e, s) = 0 end if
        if type (e) = point and type (s) = circle then
              intersect := distance (e, center(s)) $\leq$ radius (s) end if
        if type (e) = point and type (s) = lens then
              intersect := distance (e, center(upperBead(s))) $\leq$ radius (upperBead(s)) and
                           distance (e, center(lowerBead(s))) $\leq$ radius(lowerBead(s)) end if
        if type (e) circle and type (s) = point then
              intersect := distance (s, center(e)) $\leq$ radius (e) end if
        if type (e) circle and type (s) = circle then
           intersect := distance (center(e), center(s)) $\leq$ radius (e) + radius (s) end if
        if type (e) = circle and type (s) = lens then
              intersect := (area ($\Delta$ BbA) + area($\Delta$ CcB) + area($\Delta$ AaC)) $\leq$ area($\Delta$ ABC)
        end if
      end if
    end for;
    return (intersect)
end testExtremaWithBead
```

Figure 5.7 Algorithm *testExtremaWithBead.*

If none of the reference bead's extrema intersect with the target bead, then the same test needs to be performed in the reverse direction (Figure 5.8), taking the extrema of the target bead and testing whether they intersect with the reference bead at the extrema's z-values. If at least one extremum is found to intersect with a slice, then the two beads intersect.

```
algorithm testBeadBeadIntersection (a, b): Boolean
input: Two beads.
output: True if they intersect, otherwise false.
method:
    if testExtremaWithBead (extrema (a), b) then intersect := true
       else intersect := testExtremaWithBead (extrema (b), a)
    end if
    return (intersect)
end testBeadBeadIntersection
```

Figure 5.8 Algorithm *testBeadBeadIntersection.*

5.3 Intersections of Lifeline Necklaces

Intersections of lifeline beads are the basis for testing intersections of lifeline necklaces. If at least a pair of beads from two necklaces intersect, then the two necklaces intersect. The bead-necklace intersection algorithm (Figure 5.9) tests for the intersection of a bead from the reference necklace with the beads in the target necklace. Only beads from the target necklace that overlap with the reference bead's time interval are considered for intersection. The intersection test is iterated through all the beads present in the reference bead.

```
algorithm testBeadNecklaceIntersection (b1, n): Boolean
input: A bead b1 and a lifeline necklace n.
output: True if the bead b1 intersects with at least one bead of n, otherwise false.
method:
    intersect := false
    wb := intervalZ (b1)
    wn := intervalZ(n)
    if overlap (wb, wn) then
        for each b2 ∈ n do
            if overlap (wb, intervalZ (b2)) then
                if not intersect then
                    intersect := testBeadIntersection (b1, b2) end if
            end if
        end for;
    end if
    return (intersect)
end testBeadNecklaceIntersection
```

Figure 5.9 Algorithm *testBeadNecklaceIntersection.*

If none of the beads intersect, then the same test needs to be performed in the reverse direction (Figure 5.10), by taking each bead from the target necklace and testing for intersections with the beads of the reference necklace. The test returns true if at least a bead from the reference necklace intersects with a bead from the target necklace or *vice-versa.*

```
algorithm testNecklaceIntersection (n1, n2): Boolean
input: Two lifeline necklaces, n1 and n2.
output: True if at least one pair of beads intersects, otherwise false.
method:
    intersect := false
    for each $b1 \in n1$ do
        if not intersect then
            intersect := testBeadNecklaceIntersection (b1, n2) end if
    end for;
    for each $b2 \in n2$ do
        if not intersect then
            intersect := testBeadNecklaceIntersection (b2, n1) end if
    end for;
    return (intersect)
end necklaceIntersectionTest
```

Figure 5.10 Algorithm *NecklaceIntersectionTest.*

5.4 Discussion

Testing intersections of lifeline beads using 3-dimensional representations of beads is cumbersome and computationally expensive as it involves solving equations of degree 2. On the other hand, using extrema and slices of beads presents a simple solution for the intersection problem through a discrete set of operations. The extrema are finite in number for a particular bead–three for a right bead and four for an oblique bead. Hence, the maximum number of operations in using extrema for testing intersections is as follows:

- six intersection operations between two right beads,
- seven intersection operations between a right bead and an oblique bead, and
- eight intersection operations between two oblique beads.

Each of these operations requires testing the intersection of slices of beads. The slices of beads are points, circles, and lenses, which are planar geometric figures. Hence, testing the intersection of slices is a geometric operation performed on 2-dimensional figures. A sequential test of these operations determines whether two beads intersect or not. Following the evidences from the above facts we can say that testing the intersection of two lifeline beads using the extrema and slices involves at maximum eight 2-dimensional geometric operations. This result supports the hypothesis of this thesis:

To determine whether two lifeline beads intersect, a finite set of operations in 2-dimensional space produces the same logical result as a 3-dimensional model of lifeline beads.

5.5 Summary

This chapter presented algorithms for calculating whether two beads or two necklaces intersect. A general approach to solve intersections using slices of beads is proposed and algorithms to solve intersection tests using slices are sketched out. These algorithms are then used as a foundation for testing intersections of lifeline necklaces. The next chapter describes the implementation of a prototype using the algorithms discussed in this chapter to test the intersections of beads and necklaces.

Chapter 6

Prototype for Constructing and Querying Lifeline Beads and Necklaces

This chapter describes the design and implementation of a prototype for querying lifeline beads and necklaces. The prototype provides a strong proof of the bead and necklace concepts (Chapter 4) and confirms the computational model (Chapter 5) for testing intersections of lifeline beads and necklaces. It has a graphical user interface that allows the user to construct beads and necklaces and to perform interactive queries on them. The following sections discuss the architecture, class structure, implementation constraints, and user interface. Finally, the use of the prototype is illustrated with an example.

6.1 Prototype Architecture

The implementation architecture is designed to support drawing in a 3-dimensional environment and to allow for user interaction. The building block for multiple necklaces is a single necklace, which in turn is built with a series of beads. This building block hierarchy benefits from object-oriented design methodology and its features. To draw beads and necklaces we use the Open Graphics Library (OpenGL) (Woo *et al.* 1999), a cross-platform standard supporting powerful 3D rendering. This library provides many robust functions for modeling features in 3D. The implementation platform was Visual

C++ (VC++) (Horton 1998), since object-oriented techniques are central to the effectiveness of all the tools provided by VC++ for *Windows programming*. The Microsoft Foundation Class (MFC) library provided with VC++ makes Windows programming easier. Also, VC++ is chosen to take advantage of OpenGL as an application programming interface (API). The application program has a user interface that passes the input from the user to the lifeline constructor, which in turn calls OpenGL functions to render the 3-dimensional lifelines (Figure 6.1). The architecture is implemented by defining two types of classes: classes to handle beads and necklaces, and classes to manage the user interface.

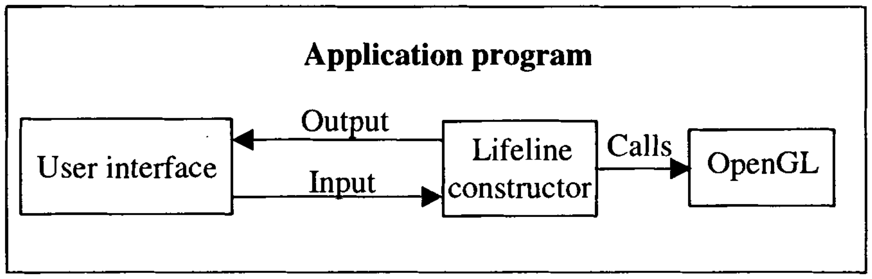


Figure 6.1 Implementation architecture.

6.1.1 Bead and necklace classes

Structures are defined to store the data necessary for a bead and a necklace (Figure 6.2). For the bead, the structure *LBeadInfo* stores two vertices (i.e., origin and destination), the bead-type (i.e., oblique or right), and the travel speed. The necklace structure is defined as *LNecklaceInfo*, which stores the necklace identifier and the set of beads that make up the necklace. The set of beads is stored as a vector array of type *LBeadInfo*.

```
struct LBeadInfo
{
   Vertex l_Origin,            // bead origin vertex
   Vertex l_Destination,       // bead destination vertex
   BeadType l_Type,            // type of bead
   float l_fSpeed              // user speed for bead
};

struct LNecklaceInfo
{
   CString l_sIdentifier,// necklace identifier
   vector <LBeadInfo> l_vBead,// array of BeadInfo
   int l_iNoOfBeads,      // number Of beads
   Clr m_Color,           // color of necklace
};
```

Figure 6.2 Structure for bead and necklace.

Class *LBead* derived from *LBeadInfo* is defined for creating a new lifeline bead (Figure 6.3a). It has a method *LBead::CalculateParameters* that computes the minimum speed required for constructing an oblique bead. A minimum speed is not required for a right bead, since a right bead can be constructed with any travel speed. The above method also calculates the time coordinate of an oblique bead's upper and lower directrices and a right bead's directrix (Section 5.2.1). Each bead is drawn by calling the method *LBead::Draw*.

A lifeline necklace is implemented as the class *LNecklace*, which is derived from *LNecklaceInfo* (Figure 6.3b). The method *LNecklace::AddBead*, takes as argument *LBeadInfo* and adds a bead to the set of beads in the necklace. The class necklace has a draw method *LNecklace::Draw* that draws the necklace by iterating through the set of beads. In subsequent discussions, lifeline necklaces and lifelines refer to one and the same object.

To handle multiple necklaces, class *LNecklaceSet* is defined (Figure 6.3c). It stores all the necklaces in a vector class of type *LNecklaceInfo*. Similar to adding a bead to a necklace, a necklace can be added to a necklace set using the method *LNecklaceSet::AddNecklace.* Given the necklace identifier, the method *LNecklaceSet::Delete* deletes a necklace from the set of necklaces. Another method *LNecklaceSet::GetNecklace* is defined to access a particular necklace from the set. This class has its own draw method called *NecklaceSet::Draw* to draw all the necklaces in the set. The draw methods in all the above classes call the OpenGL rendering functions defined in a file *glfunctions.h.* In addition, the Graphics Language Utility Toolkit (Glut) library functions (Wright and Sweet 1999) are used for drawing.

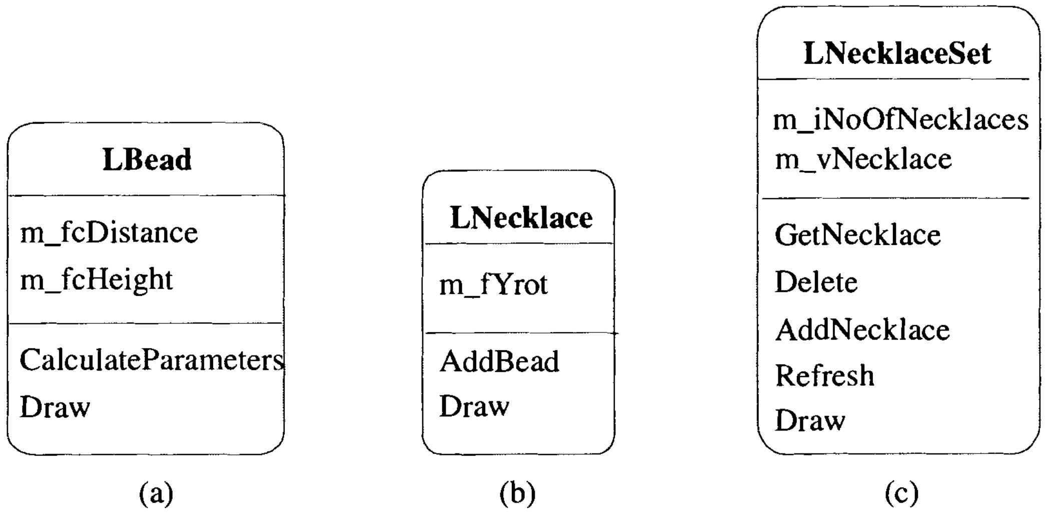
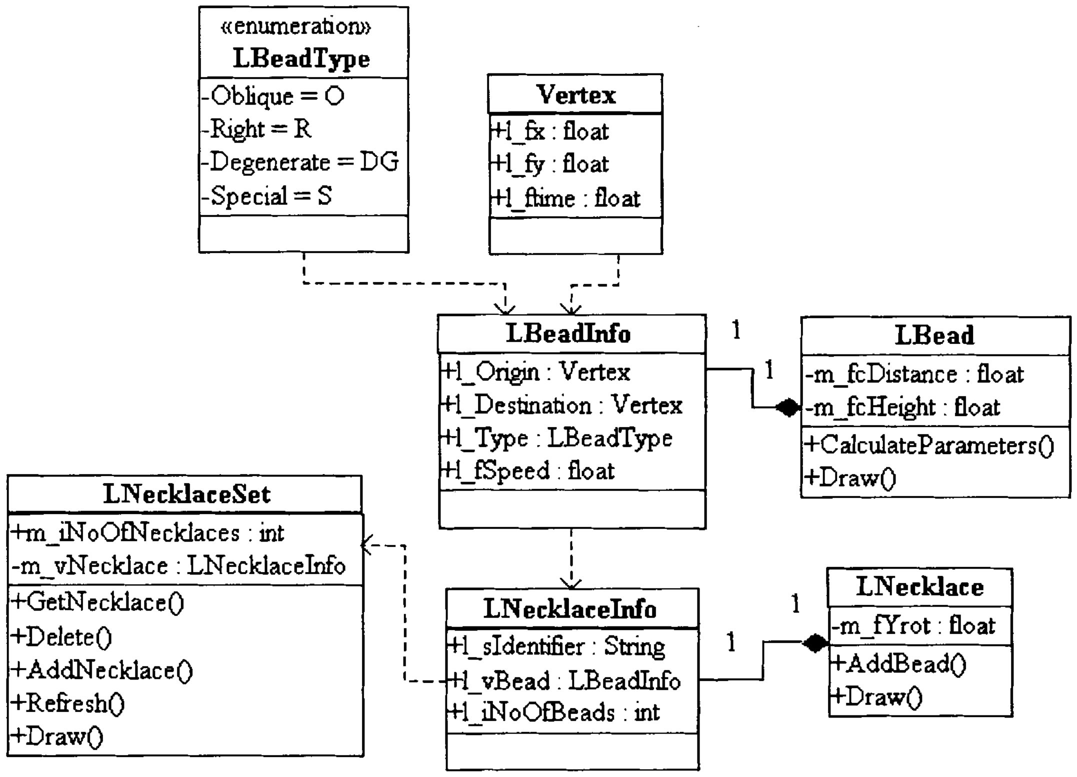


Figure 6.3 Classes for (a) bead, (b) necklace, and (c) necklaceset.

The class diagram (Figure 6.4) shows the connectivity among the classes and structures explained in this section.



A┈┈▶B indicates that *B* is dependent on *A* by using the definition of *A*.

A┈┈◆B indicates that *B* is derived from *A*.

- private member

+ public member

Figure 6.4 Bead and necklace classes.

6.1.2 User-interface classes

The user interface classes are handled via the document-view architecture in VC++. The document class *CBeadGUIDocument* stores all the objects that the application requires and its member functions support processing of data. The view class *CBeadGUIView* provides a mechanism for displaying the data stored in the document class. The user interface classes defined in this prototype are the dialog classes derived from *CDialog*. The following discussion briefly describes the user interface classes.

Class *CLifeline* is defined to handle a new lifeline (Figure 6.5a). The method *CLifeline::ChkIpValues* checks the input values for consistency. For example, it checks that the destination time is greater than the origin time. The minimum speed of travel required for consistency check is calculated using the method *CLifeline::ComputeSpeed.* This class calls another class, *CAddDestination*, to add further destinations to the lifeline. Similar to *CLifeline*, *CAddDestination* performs consistency check for the newly entered destination with the previous destination. The class *CSelectLifeline* maintains a list of lifeline identifiers. A particular lifeline can be selected from this set of lifelines (Figure 6.5b). The lifeline with the selected identifier is then used in succeeding operations such as adding destinations or deleting a lifeline.

The query component is implemented through the class *CQuery* (Figure 6.5c). The algorithms discussed in chapter 5 are implemented in this class for testing lifeline intersections. The method *CLifelineQuery::TestBeadIntersection* tests for the intersection of two beads and the method *CLifelineQuery::TestLifelineIntersection* tests for the intersection of two lifelines.

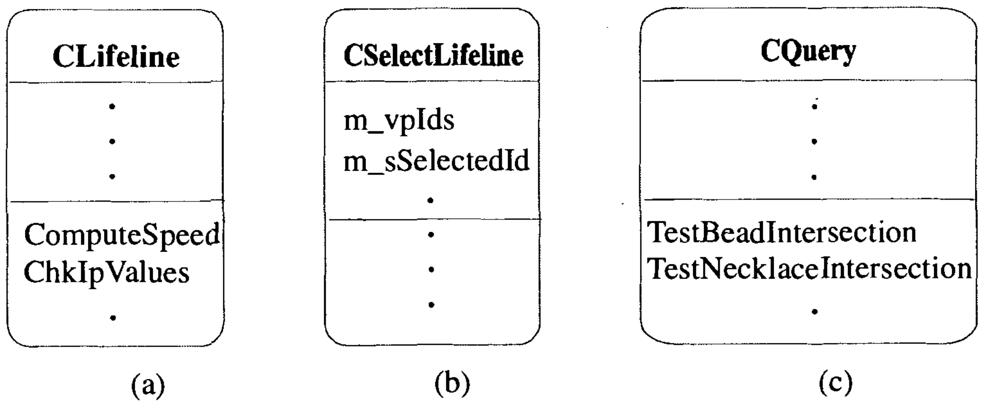


Figure 6.5 User interface classes: (a) CLifeline, (b) CSelectLifeline, and (c) CQuery.

6.1.3 Implementation constraints

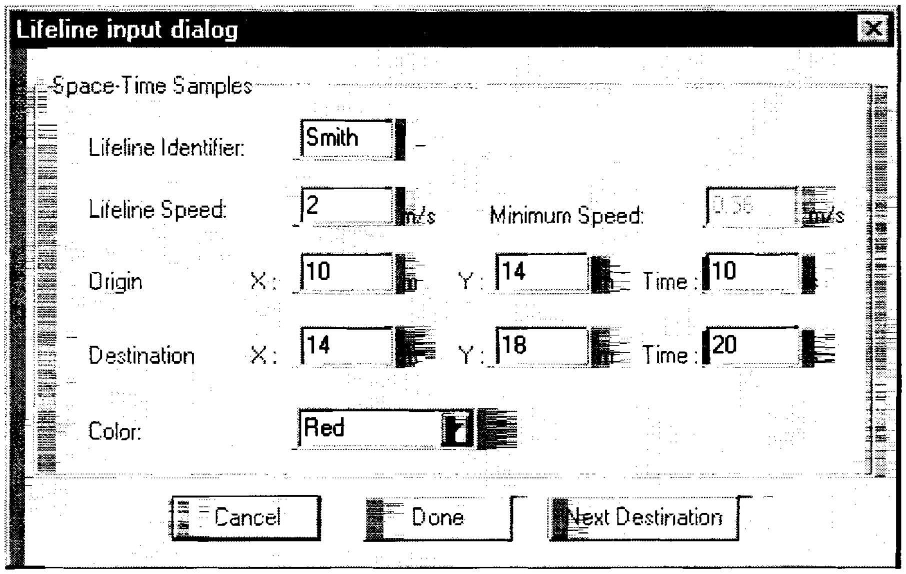
The drawing of an oblique bead involves three operations. The *intersection* between the top and bottom halfbeads gives the region between the two directrices of the bead. The second and third operations add the portion of the bead above the upper directrix and below the lower directrix, respectively. These operations are done by issuing a set of OpenGL commands. The complexity of these operations inhibits the storing of the result of three operations as a single object. This creates problems in recreating the drawing with ease, since each time when a necklace is redrawn these operations are repeated which is time consuming. Also, the drawing is allowed only in the positive *x*, *y*, and *time* axes; and the minimum and maximum coordinates are set between 0 and 100, respectively.

6.2 Description of User Interface

The prototype provides a number of user-interface components. The components are used to create, edit, query, and delete a lifeline and are linked as sub-menus to the menu item *Lifeline* created in the application program. The following sections describe the function of each user interface component.

6.2.1 Creating a new lifeline

The *Lifeline input dialog* (Figure 6.6) allows the user to start a new lifeline by entering an identifier for the lifeline, specifying a travel speed, providing coordinates for origin and destination, and selecting a color. The input values undergo a consistency check where the user-specified speed s_u and the minimum required speed s_{min} (calculated from the input coordinates) are compared. If $s_u < s_{min}$, the user is prompted to either increase the speed or modify the coordinates. With consistent speed and destination the user can either select the option to draw the lifeline or to continue to add more destinations to the lifeline.


Lifeline input dialog

Space-Time Samples

Lifeline Identifier: Smith

Lifeline Speed: 2 m/s Minimum Speed: 0.56 m/s

Origin X: 10 Y: 14 Time: 10

Destination X: 14 Y: 18 Time: 20

Color: Red

Figure 6.6 Lifeline input dialog.

6.2.2. Selecting a lifeline

The user can create as many lifelines as desired. These lifelines are stored for further operations, such as adding more destinations or deleting a particular lifeline. The *Select lifeline dialog* (Figure 6.7) lists the set of lifelines, from which the user can select a lifeline. All further operations are done on the selected lifeline. This dialog is not invoked until the user creates at least one lifeline.

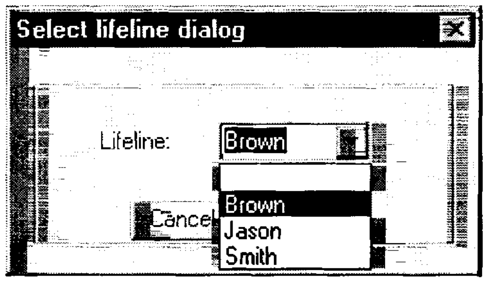
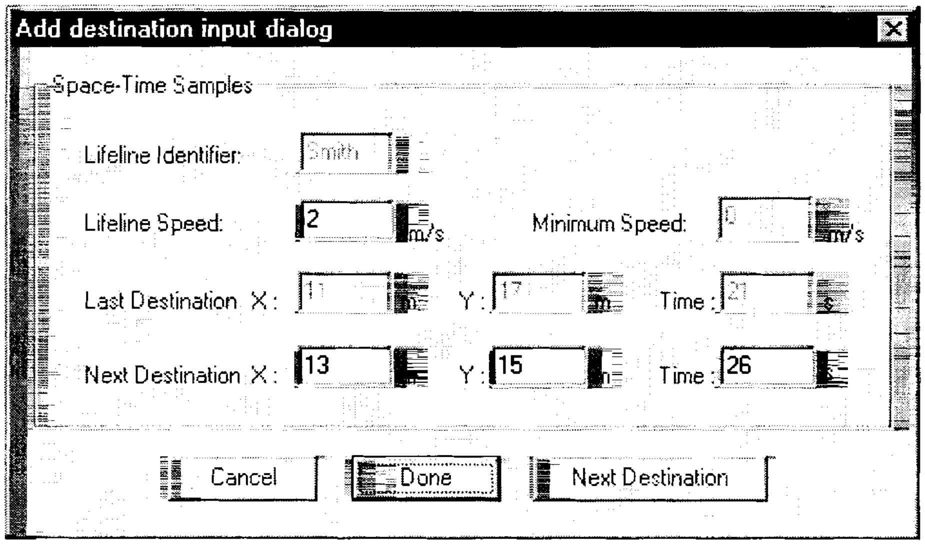


Figure 6.7 Select lifeline dialog that lists the available lifelines.

6.2.3. Adding destinations to a lifeline

To a selected lifeline, the user can add more destinations using the *Add destination input dialog* (Figure 6.8). This dialog displays the currently selected lifeline's identifier, coordinates, and speed of the previous destination entered by the user. The user cannot edit both the lifeline identifier and the previous destination, but can change the speed. The user is required to enter the new destination coordinates. These values are then checked for consistency (Section 6.2.1). The user can continue to add more destinations or opt to draw the lifeline.


Add destination input dialog

Space-Time Samples

Lifeline Identifier: Smith

Lifeline Speed: 2 m/s   Minimum Speed: 0 m/s

Last Destination X : 11 m   Y : 17 m   Time : 21 s

Next Destination X : 13 m   Y : 15 m   Time : 26 s

Figure 6.8 Add destination input dialog to add destinations to a lifeline.

6.2.4. Deleting a lifeline

A lifeline can be deleted from the set of lifelines. The user selects a lifeline using the *Select lifeline dialog* described in Section 6.2.2. The selected lifeline gets deleted from the set.

6.2.5. Querying beads and lifelines

The most important capability offered by the implementation is to allow users to query lifeline intersections. The *Query dialog* (Figure 6.9) is designed as a query builder to construct intersection queries. The dialog lists the available lifelines. When a lifeline is selected the corresponding set of beads is displayed. The dialog allows the user to select two parameters, either two lifelines or two beads, for the query. After selecting the query parameters, the intersection test (see Figure 5.8 and Figure 5.10) is performed and a message box displays the query result.

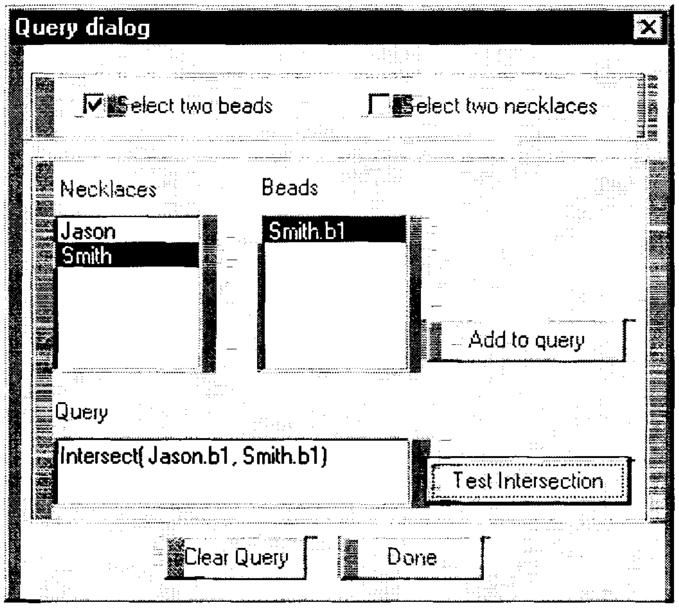


Figure 6.9 Query dialog.

6.3. Illustration of the Prototype

The use of the prototype is illustrated with an example in this section. The example starts with the construction of a new lifeline called *Smith* with a travel speed of 2.5 m/s. Origin and destination coordinates are entered as (15, 17, 12) and (19, 22, 20), respectively, and a red color is selected for drawing. Since the minimum required speed (0.80 m/sec) derived from the destination time and the origin time, is less than the travel speed, this bead passes the consistency test and the bead is drawn (Figure 6.10).

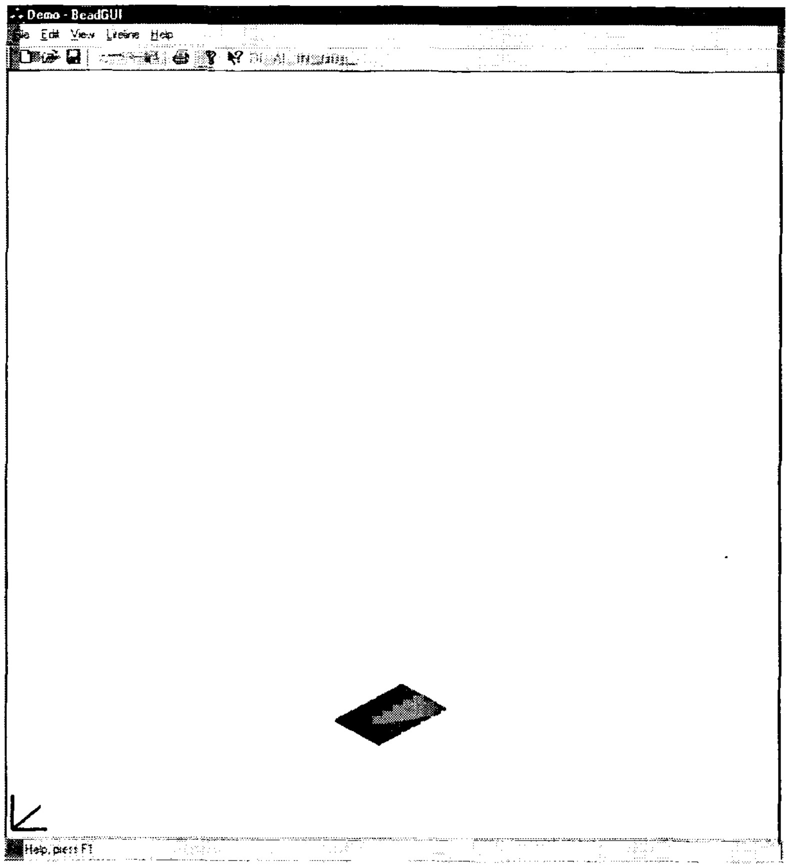


Figure 6.10 Lifeline *Smith* with a single bead.

To add more destinations to the lifeline *Smith*, the user enters the destination values (Table 6.1) using the *Add destination input dialog*. Now the lifeline *Smith* consists of five beads (Figure 6.11).

Destination (m, m, s)	Speed (m/s)
(23, 25, 28)	2.5
(19, 20, 35)	2.5
(19, 20, 43)	1.5
(22, 24, 50)	1.5

Table 6.1 Destination and speed values for *Smith*.

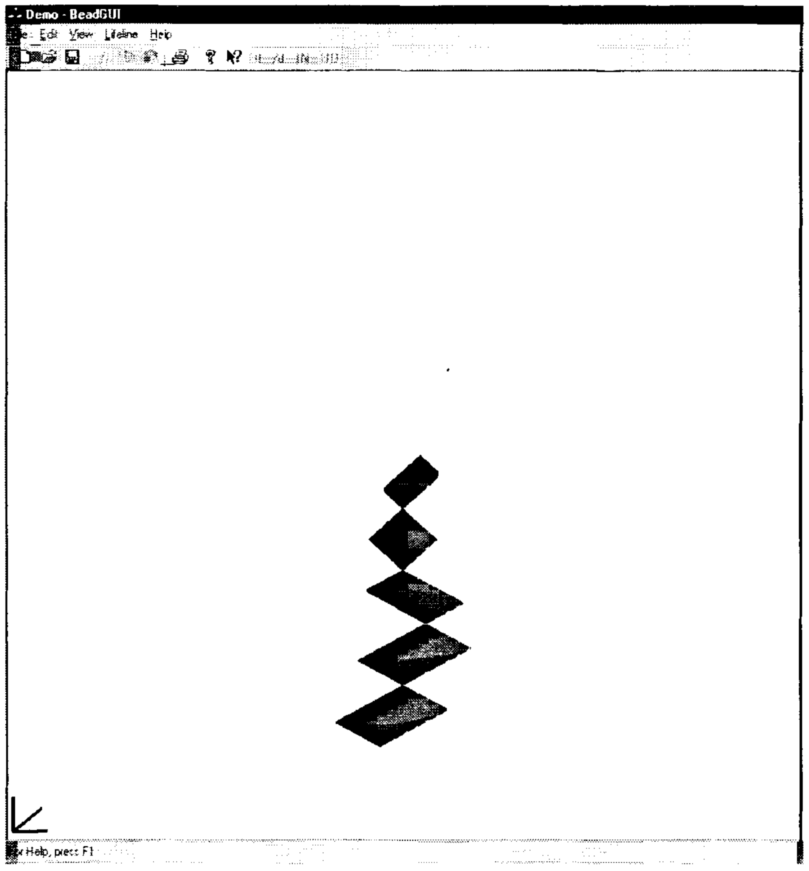


Figure 6.11   Lifeline *Smith* with added destinations.

The next lifeline with identifier *Frank* is created with the origin and destination as (40, 42, 13) and (44, 46, 20), respectively and the travel speed as 3.2 m/s. *Frank* is drawn in yellow color. Table 6.2 lists the coordinates and speed values for adding further destinations to *Frank*, which consists of five beads with the last bead being a line bead (Figure 6.12).

Destination (m, m, s)	Speed (m/s)
(38, 40, 27)	3.2
(42, 43, 34)	3.2
(35, 38, 40)	3.0
(20, 24, 48)	2.56

Table 6.2 Destination and speed values for lifeline *Frank*.

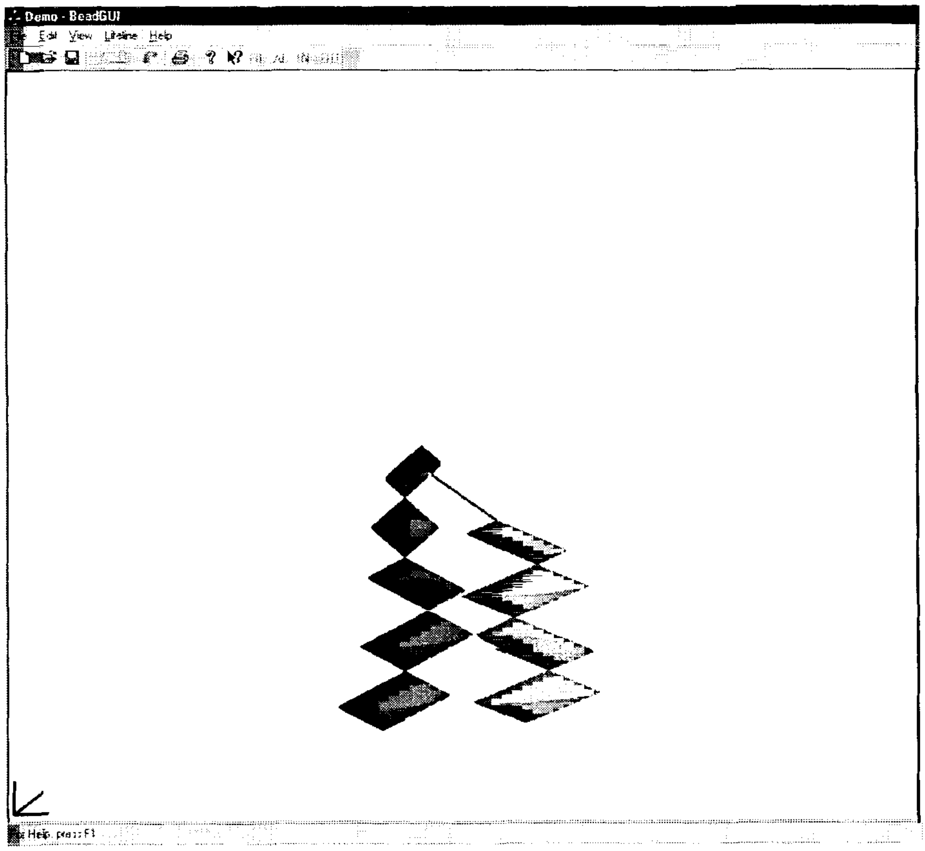


Figure 6.12 Lifelines *Smith* is on the left and lifeline *Frank* is on the right.

Finally, an intersection query is performed between the two lifelines using the *Query dialog*. The query returns the result that the lifelines for *Smith* and *Frank* do intersect (Figure 6.13).

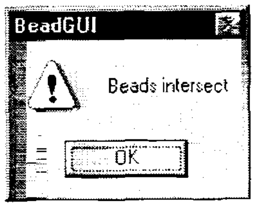

Beads intersect

Figure 6.13 Query result.

6.4. Summary

This chapter described the prototype implementation for querying lifeline necklaces. The architecture and the class structure of the prototype were discussed to understand the data flow and interaction between the application program and the user. The prototype was also used as a test bed for the bead and necklace intersection. The next chapter concludes the thesis with summary and future recommendations to be carried out based on this research work.

Chapter 7

Conclusions

This chapter summarizes the thesis work and presents conclusions. The future directions for research based on this work are highlighted with recommendations.

7.1 Summary

This thesis developed a computational model for testing intersections of lifelines. Lifelines represent individual's movements through a sequence of space-time samples and the maximum travel speed. The lifelines were modeled using beads and necklaces that provided a strong theoretical foundation for the computational model. Beads are three-dimensional constructs that capture all the possible locations of an object moving between two time-stamped samples *A* and *B* at maximum speed *s*. Necklaces are formed by connecting series of samples using beads. The computational model was implemented using algorithms that reduced a 3-dimensional problem to two dimensions, thus demonstrating the simplicity of the model. A software prototype was also developed to create beads and necklaces and test their intersections using the algorithms.

7.2 Conclusions

The following conclusions arise from this thesis:

- Moving objects' paths often intersect with each other. These intersections are captured by analyzing the intersections of the objects' geospatial lifeline beads and necklaces. The computational model introduced in this thesis helps to solve this intersection problem.
- To determine whether two lifeline beads intersect, a finite set of two-dimensional geometric operations using slices of beads produces the same result as the more complicated three-dimensional model of beads.
- The algorithms supporting the computational model reduce the complexity of calculating three-dimensional intersections by using the two-dimensional slices. The algorithms are simple and avoid solving mathematical equations of higher degrees.

7.3 Future Work

This section lists a set of future research tasks that are enabled by this model.

- *Modeling a lifeline bead with an obstacle.*

  Though the bead and the necklace model is complete in representing moving objects, the model assumes isotropic movement (i.e., movement is invariant in all directions) and does not take into consideration constraints in the space in the form of a network or obstacles. For example, if there is a transportation network or a natural barrier such as lake that obstructs movement, the behavior of the model is different. Extensions to the lifeline necklace model are needed to accommodate such settings. For example, the intersection of the obstacle treated as a 3-dimensional cylinder with the bead gives the volume where the spatio-temporal movement cannot exist (Figure 7.1).

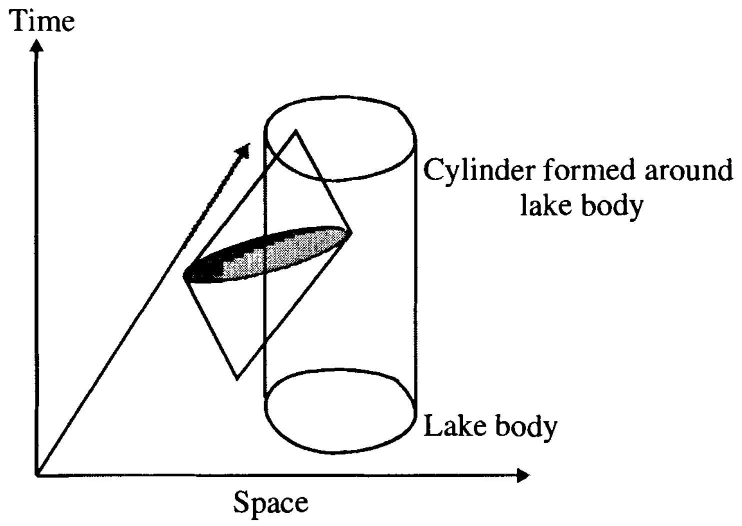


Figure 7.1 A bead with a lake body as an obstacle.

- Formalizing different types of bead relations.

This thesis explored only few types of bead relationship such as disjoint, empty, and overlap. There are many other interesting relations between the beads that offer semantics required for analyzing the beads. All these different types of bead relations can be formalized and presented as a relation set.

- A complete query language.

In this thesis mainly intersection queries are performed over lifelines though the lifeline bead and necklace model can answer numerous other queries. For example, an interesting query would be *what are all the possible locations where two individuals could have met.* This query requires the projection of the common volume of intersections of individual's lifelines into the *XY* plane. The projected area gives the possible locations where the two individuals could have met. This query process requires more computational efforts. Hence a *query language* built on this model can be taken up as future work.

- Performance evaluation.

The computational model introduced in this thesis promises to be inexpensive. A performance test between this computational model and the mathematical equations for three-dimensional constructs to solve intersections can be carried out as future work.

The prototype implemented in this thesis works well with a small data set. However, for large datasets, database issues such as indexing and data storage tailored to this application need to be investigated. Also, the validity of the model should be tested with real datasets collected using GPS receivers.

- Implementation issues with OpenGL.

The complexity of operations in constructing an oblique bead using OpenGL functions lowers the performance of 3D rendering, especially when the screen is refreshed or rotated, or the beads are redrawn. Efforts can be directed in this regard to treat the result of complex operations as a single object. This will enhance the performance of screen refreshing, rotation, and redraw operations. Open geometry (Glaeser and Stachel 1999) is promising in achieving the above task, but other aspects such as modification of the application to suit to the Open geometry environment need to be investigated.

BIBLIOGRAPHY

R. Abler, J. Adams, and P. Gould (1971) *Spatial organisation: The geographer's view of the world.* Englewood Cliffs, New Jersey, Prentice-Hall.

T. Abraham and J. Roddick (1999) Survey of spatio-temporal databases. *Geoinformatica* 3(1): 61-99.

J. Chomicki and P. Revesz (1997) Constraint-based interoperability of spatiotemporal databases, *15th International Symposium on Spatial Databases*, pp. 142-161.

M. Egenhofer (1991) Reasoning about binary topological relations. in: O. Gunther and H. Schek (Eds), *Second Symposium on Large Spatial Databases*, Zurich, Switzerland, Lecture Notes in Computer Science, Vol. 525, pp. 143-160, Springer-Verlag.

M. Egenhofer and R. Golledge (1994) *Time in Geographic Space, Report on the Specialist Meeting of Research Initiative 10.* Technical Report 94-2, Santa Barbara, USA, University of California.

P. Forer (1998) Geometric approaches to the nexus of time, space, and microprocesses: implementing a practical model for mundane socio-spatial systems. in: M. Egenhofer and R. Golledge (Eds). *Spatial and Temporal Reasoning in Geographic Information Systems*: 171-190. New York, Oxford University Press.

L. Forlizzi, R. Güting, E. Nardelli, and M. Schneider (2000) A data model and data structures for moving object databases. in: W. Chen, J. Naughton, and P. Bernstein (Eds), *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*, Dallas, Texas, Vol. 29, pp. 319-330, ACM Press.

G. Glaeser and H. Stachel (1999) *Open Geometry*. Springer.

R. Güting, M. Böhlen, M. Erwig, C. Jensen, N. Lorentzos, M. Schneider, and M. Vazirgiannis (1998) A foundation for representing and querying moving objects. *ACM Transactions on Database Systems* 25(1): 1-42.

T. Hägerstrand (1970) What about people in regional science? *Papers of the Regional Science Association* 14(7): 7-21.

P. Haggett (1990) *The Geographer's Art*. Cambridge, Massachusetts, Basil Blackwell.

R. Hariharan and K. Hornsby (2000) Modeling intersections of geospatial lifelines. in: A. Caschetta (Ed.), *First International Conference on Geographic Information Science*, Savannah, GA, pp. 208-210.

K. Hornsby and M. Egenhofer (in press) Modeling moving objects over multiple granularities. *Annals of Mathematics and Artificial Intelligence.*

I. Horton (1998) *Beginning Visual C++*. Chicago, Illinois, Wrox Press Ltd.

G. Langran (1992) *Time in Geographic Information Systems*. London, Taylor & Francis Ltd.

B. Lenntorp (1976) Paths in space-time environments: A time geographic study of the movement possibilities of individuals. *Lund Studies in Geography* Series B(44).

D. Mark, M. Egenhofer, L. Bian, K. Hornsby, P. Rogerson, and J. Vena (1999) Spatio-temporal GIS analysis for environmental health using geospatial lifelines. in: A. Flahault, L. Toubiana, and A. Valleron (Eds), *2nd International Workshop on Geography and Medicine, GEOMED '99*, Paris, France, pp. 52.

H. Miller (1991) Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Systems* 5(3): 287-301.

J. Moriera, C. Ribeiro, and J. Saglio (1999) Representation and manipulation of moving points: an extended data model for location estimation. *Cartography and Geographic Information Science* 26(2): 109-123.

M. Nascimento, J. Silva, and Y. Theodiridis (1999) Evaluation of Access Structures for Discretely Moving Points. in: M. Böhlen, C. Jensen, and M. Scholl (Eds), *STDBM '99*, Edinburgh, Scotland, Lecture Notes in Computer Science, Vol. 1678, pp. 171-188, Springer.

S. Nishida, H. Nozawa, and N. Saiwaki (1998) Proposal of spatio-temporal indexing methods for moving objects. in: D. Lee, E. Lim, M. Mohania, and Y. Masunaga (Eds), *Advances in Database Technologies, ER '98 Workshops on Data Warehousing and Data Mining, Mobile Data Access, and Collaborative Work Support and Spatio-Temporal Data Management*, Singapore, Lecture Notes in Computer Science, Vol. 1552, pp. 484-495, Springer.

J. Odland (1998) Longitudinal analysis of migration and mobility spatial behavior in explicitly temporal contexts. in: M. Egenhofer and R. Golledge (Eds). *Spatial and Temporal Reasoning in Geographic Information Systems*: 238-259, Oxford University Press.

D. O'Sullivan, A. Morrison, and J. Shearer (2000) Using desktop GIS for the investigation of accessibility by public transport: an isochrone approach. *International Journal of Geographical Information Science* 14(1): 85-104.

D. Pfoser and C. Jensen (1999) Capturing the uncertainty of moving-object representations. in: R. Güting, D. Papadias, and F. Lochovsky (Eds), *6th International Symposium, SSD '99*, Hong Kong, China, Lecture Notes in Computer Science, Vol. 1651, pp. 111-131, Springer-Verlag.

D. Pfoser, C. Jensen, and Y. Theodoridis (2000) Novel approaches in query processing for moving object trajectories. in: A. El Abbadi, M. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K. Whang (Eds), *VLDB 2000, Proceedings of 26th International Conference on Very Large Databases*, Cairo, Egypt, pp. 395-406, Morgan Kaufmann.

R. Rugg (1973) Map Records of Forest Recreational Itineraries. *Journal of Leisure Research* 5 (Winter): 60-66.

S. Saltenis, C. Jensen, S. Leutenegger, and M. Lopez (2000) Indexing the positions of continuously moving objects. in: W. Chen, J. Naughton, and P. Bernstein (Eds), *2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, Vol. 29, pp. 331-342, ACM Press.

T. Sellis (1999) Research issues in spatio-temporal database systems. in: R. Güting, D. Papadias, and F. Lochovsky (Eds), *6th International Symposium, SSD '99*, Hong Kong, China, Lecture Notes in Computer Science, Vol. 1651, pp. 5-11.

A. Sistla, O. Wolfson, S. Chamberlain, and S. Dao (1997a) Modelling and querying moving objects. in: A. Gray and P. Larson (Eds), *Proceedings of the Thirteenth International Conference on Data Engineering*, Birmingham, UK, pp. 422-432, IEEE Computer Society.

A. Sistla, O. Wolfson, S. Chamberlain, and S. Dao (1997b) Querying the uncertain positions of moving objects. in: O. Etzion, S. Jajodia, and S. Sripada (Eds), *Dagstuhl Seminar: Temporal Databases*, Dagstuhl, Germany, Lecture Notes in Computer Science, Vol. 1399, pp. 310-337, Springer.

T. Tzouramanis, M. Vassilakopoulos, and Y. Manolopoulos (1998) Overlapping linear quadtrees: a spatio-temporal access method, *6th ACM International Workshop on Advances in Geographic Information Systems*, pp. 1-7.

O. Wolfson, L. Jiang, P. Sistla, S. Chamberlain, N. Rishe, and M. Deng (1999a) Databases for tracking mobile units in real time. in: C. Beeri and P. Buneman (Eds), *Proceedings of Database Theory - ICDT '99, 7th International Conference*, Jerusalem, Israel, Vol. 1540, pp. 169-186, Springer.

O. Wolfson, P. Sistla, B. Xu, J. Zhou, and S. Chamberlain (1999b) DOMINO: Databases for moving objects tracking. in: A. Delis, C. Faloutsos, and S. Ghandeharizadeh (Eds), *Proceedings of ACM SIGMOD International Conference on Management of Data*, Philadelphia, PA, pp. 547-549, ACM Press.

M. Woo, J. Neider, T. Davis, and D. Shreiner (1999) *OpenGL programming guide.* Addison Wesley.

R. Wright and M. Sweet (1999) *OpenGL SuperBible.* Waite Group Press.

N. Yattaw (1999) Conceptualizing space and time: a classification of geographic movement. *Cartographic and Geographic Information Science* 26(2): 85-98.

T. Yeh and Y. Vermont (1992) Temporal aspects of geographical databases. *Third European Conference on Geographical Information Systems.* Munich, Germany, pp. 320-328.

BIOGRAPHY OF THE AUTHOR

Ramaswamy Hariharan was born in Chennai, India on February 1, 1978. He received his Bachelor of Engineering degree in Geoinformatics from Anna University, India in 1999. The program was affiliated with the Institute of Remote Sensing (IRS), a national sub-center for remote sensing activities. During his stay at Anna University, he was involved in various state government funded GIS projects at IRS. Then he joined the University of Maine's Spatial Information Science and Engineering program as a master's candidate in the fall of 1999. He worked as a research assistant with the National Center for Geographic Information and Analysis (NCGIA). Ramaswamy is a candidate for the Master of Science degree in Spatial Information Science and Engineering from The University of Maine in December 2001.