

Deriving an Obstacle-Avoiding Shortest Path in Continuous Space:

A Spatial Analytic Approach

by

Insu Hong

A Dissertation Presented in Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy

Approved January 2015 by the  
Graduate Supervisory Committee:

Alan Murray, Chair  
Michael Kuby  
Sergio Rey

ARIZONA STATE UNIVERSITY

May 2015

## ABSTRACT

The shortest path between two locations is important for spatial analysis, location modeling, and wayfinding tasks. Depending on permissible movement and availability of data, the shortest path is either derived from a pre-defined transportation network or constructed in continuous space. However, continuous space movement adds substantial complexity to identifying the shortest path as the influence of obstacles has to be considered to avoid errors and biases in a derived path. This obstacle-avoiding shortest path in continuous space has been referred to as Euclidean shortest path (ESP), and attracted the attention of many researchers. It has been proven that constructing a graph is an effective approach to limit infinite search options associated with continuous space, reducing the problem to a finite set of potential paths. To date, various methods have been developed for ESP derivation. However, their computational efficiency is limited due to fundamental limitations in graph construction. In this research, a novel algorithm is developed for efficient identification of a graph guaranteed to contain the ESP. This new approach is referred to as the convexpath algorithm, and exploits spatial knowledge and GIS functionality to efficiently construct a graph. The convexpath algorithm utilizes the notion of a convex hull to simultaneously identify relevant obstacles and construct the graph. Additionally, a spatial filtering technique based on intermediate shortest path is enhances intelligent identification of relevant obstacles. Empirical applications show that the convexpath algorithm is able to construct a graph and derive the ESP with significantly improved efficiency compared to visibility and local visibility graph approaches. Furthermore, to boost the performance of convexpath in big data environments, a parallelization approach is proposed and applied to exploit

computationally intensive spatial operations of convexpath. Multicore CPU parallelization demonstrates noticeable efficiency gain over the sequential convexpath. Finally, spatial representation and approximation issues associated with raster-based approximation of the ESP are assessed. This dissertation provides a comprehensive treatment of the ESP, and details an important approach for deriving an optimal ESP in real time.

Dedicated to my parents, Yang Hong and Heejae Jo,

To my friends, especially to Myungchul and Hyungin,

and

To my cats, Harry and Puffy.

Make it so!

## ACKNOWLEDGEMENTS

Over the years of my Ph.D. study, I have received great supports and encouragements from many people. I am using this opportunity to express my gratitude to everyone who supported me throughout the course. Most of all, I must thank to Dr. Alan T. Murray, my adviser. Without years of his patient guidance and support, it would not be possible to make this far. Even in the hardest times of my study, he believed me and supported me, and gave me invaluable advice for advancement. I really appreciate his mentorship for me.

I would like to thank to my committee members, Dr. Sergio J. Rey and Dr. Michael Kuby, for help and advice for my dissertation. It is my great fortune to study in the GeoDa Center and School of Geographical Sciences and Urban Planning.

Also, I would like to thank to my colleagues and friends, including Levi and Kihwan for their support, advice, feedbacks, and friendship. I would like to thank to my dear friends in Korea who have supported me during the hardest years of my life so far.

# TABLE OF CONTENTS

	Page
LIST OF TABLES.....	viii
LIST OF FIGURES.....	ix
CHAPTER	
1 INTRODUCTION.....	1
Background.....	1
Research Objective.....	4
Organization of the Research.....	5
2 EFFICIENT MEASUREMENT OF CONTINUOUS SPACE SHORTEST DISTANCE AROUND BARRIERS.....	7
Introduction.....	7
Background.....	10
Problem Formalization.....	12
Potential Solution Approaches.....	18
Deriving an Efficient Graph.....	22
Application Results.....	29
Discussion and Conclusions.....	33
3 EFFICIENT WAYFINDING IN COMPLEX ENVIRONMENT: DERIVATION OF A CONTINUOUS SPACE SHORTEST PATH.....	35
Introduction.....	35

CHAPTER	Page
Background.....	37
Euclidean Shortest Path.....	40
Convexpath Algorithm.....	41
Improving the Convexpath Algorithm.....	44
Application.....	48
Discussion and Conclusions.....	49
<b>4 SPATIAL FILTERING FOR IDENTIFYING A SHORTEST PATH AROUND OBSTACLES.....</b>	<b>52</b>
Introduction.....	52
Background.....	54
Convexpath.....	57
Spatial Filtering.....	62
Application Results.....	71
Discussion and Conclusions.....	75
<b>5 HIGH PERFORMANCE COMPUTING TO DERIVE AN OBSTACLE- AVOIDING SHORTEST PATH.....</b>	<b>77</b>
Introduction.....	77
Background.....	80
Shortest Path Derivation.....	83
Application Parallelization.....	93
Computational Results.....	99
Conclusions and Discussion.....	105

CHAPTER	Page
6 ASSESSING RASTER GIS APPROXIMATION FOR EUCLIDEAN SHORTEST PATH ROUTING.....	108
Introduction.....	108
Background.....	111
Vector Representation.....	115
Raster Representation.....	120
Study and Empirical Findings.....	124
Conclusions and Discussion.....	130
7 CONCLUSIONS.....	133
Summary.....	133
Future Work.....	135
BIBLIOGRAPHY.....	145



## LIST OF TABLES

Table	Page
4.1 Average Performance Comparison of 30 Origin-destination Pairs Using the Convexpath and Convexpath-sf Algorithm.....	74
5.1 Application Summary.....	102
5.2 Summary of Performance Improvement.....	105

## LIST OF FIGURES

Figure	Page
2.1 Line Segments Between Obstacles and Regional Boundary Vertices.....	18
2.2 Convex Hull for Two Points and an Obstacle.....	23
2.3 Combined Set of Multiple Convex Hulls.....	25
2.4 Flowchart of $G^*$ Generation.....	26
2.5 Pseudo Code for the Convexpath Algorithm.....	27
2.6 Application Area, Graph $G^*$ and Shortest Path.....	30
2.7 Application Results.....	32
3.1 Steps of the Convexpath.....	43
3.2 Violate Convexity Assumption.....	44
3.3 Dividing Approach.....	45
3.4 Steps of the Extension of the Convexpath Algorithm.....	47
4.1 Iterative Convexpath Graph Construction Process.....	63
4.2 Resulting Graph and ESP Using the Convexpath-sf Algorithm.....	71
4.3 Travel Path Planning Contexts, Each Showing One Origin-destination Pair.....	73

Figure	Page
4.4 Average Relative Performance Comparison of 30 Origin-destination Pairs in the Different Obstacle Density Contexts using the Convexpath and Convexpath-sf Algorithms.....	74
5.1 Three Cases of Intersection.....	79
5.2 Convex Hull for Origin, Destination and Impeding Obstacle.....	86
5.3 Line-polygon Overlay.....	88
5.4 Graph Construction using Line-polygon Overlay.....	89
5.5 Spatial Filter Operator.....	91
5.6 Parallel Processing Steps of the Convexpath-parallel Algorithm.....	99
5.7 Arizona State University Campus Buildings.....	100
5.8 Generated Graph using Convexpath along with Optimal ESP.....	101
5.9 Scatterplot of Sequential/parallel Computing Time Ratio against the Number of Arcs in Graph.....	104
6.1 Shortest Path Derivation.....	109
6.2 Convexpath Algorithm Graph and Path Generation Process.....	119
6.3 Impact of Raster Resolution on Obstacle Representation.....	122
6.4 Different Cell Neighbor Definitions.....	123
6.5 Arizona State University Campus Buildings.....	125

Figure	Page
6.6 ESP and Least Cost Path Approximations.....	127
6.7 Path Comparisons.....	129

## CHAPTER 1

### INTRODUCTION

#### 1.1. Background

The shortest path between two locations provides crucial information for spatial analysis, route planning, and location modeling. It has been used as a surrogate measurement of proximity and distance in spatial analysis and location modeling, such as for deriving service areas and assessing behavioral movements (Klamroth 2001a, Jones *et al.* 2010). Furthermore, shortest paths are fundamental knowledge for planning a route for human, robots, ships, and virtual objects (Lozano-Pérez and Wesley 1979, Fagerholt *et al.* 2000, Yap *et al.* 2011). If a movement occurs on a preexisting transportation network and network data is already available, network shortest path algorithms can provide the shortest path quickly and reliably. However, if the availability of this data is limited or movement is not restricted to a transportation network, the Euclidean straight line path between the given locations is commonly used as a stand-in for a true shortest path. Due to its simplicity and availability, the Euclidean straight line path has been widely utilized for assessment of proximity and spatial interaction, such as accessibility analysis in healthcare planning (Phibbs and Luft 1995, Fone *et al.* 2006, Higgs 2009, Jones *et al.* 2010, Cudnik *et al.* 2012).

In continuous space, however, obstacles or barriers that influence movements must be taken into account. Assessment that does not consider these spatial nuisances when deriving the shortest path are biased and often unnavigable (Carling *et al.* 2012).

Deriving the shortest path that avoids obstacles in continuous space is a substantially more complex problem, since continuous space implies an infinite number of options possible for movement. The problem of deriving an obstacle-avoiding shortest path in continuous space has been referred to as the Euclidean shortest path (ESP) problem, and substantial research effort has been devoted to its formalization and solution techniques (Guibas and Hershberger 1989, Hershberger and Suri 1993, Mitchell 1999). It has been proven that transforming continuous space to a discrete network of obstacle vertices is an effective approach to derive the ESP (Wangdahl *et al.* 1974, Lozano-Pérez and Wesley 1979, Viegas and Hansen 1985, de Berg *et al.* 2008). The visibility graph is the most prominent method for ESP derivation and involves the construction of a graph that connects all feasible vertices in a given area. Since Lozano-Pérez and Wesley (1979) initially proposed the visibility graph as a method for collision-free robot path planning, many have attempted to extend the procedure or improve the computational efficiency of the graph derivation process (Asano 1985, Welzl 1985, Asano *et al.* 1986, Rohnert 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996). Furthermore, the local visibility graph (Kim *et al.* 2004, Zhang *et al.* 2005), shortest path map (Mitchell 1999), and Voronoi diagram (Papadopoulou and Lee 1995) approaches have been proposed to solve the ESP problem for spatial analysis, robotics, shipping, location modeling, and more (Takahashi and Schilling 1989, Fagerholt *et al.* 2000, Klamroth 2001b, Zhang *et al.* 2005).

Unfortunately, most existing solution approaches for the ESP problem are computationally inefficient. To construct a graph for the ESP derivation, they have to evaluate all or most vertices for all obstacles in a given region, regardless of the location

of origin and destination points. As a result, deriving the ESP is computationally prohibitive, especially for large-sized spatial problems. Although the local visibility graph approach tries to filter out a portion of the obstacles using proximity-based filtering techniques (Kim *et al.* 2004, Zhang *et al.* 2005, Gao *et al.* 2011), it still requires considerable amount of computing resources and induces the possibility of errors in the resulting path. Therefore, a novel approach is required that is able to intelligently process obstacles and vertices for efficient construction of a graph while still guaranteeing inclusion of the ESP.

In big data environment, even highly efficient methods may have degraded performance, since ESP derivation requires repeated, computationally intensive spatial operations. To overcome scale limitations in practice, advanced computing techniques are necessary to enhance the performance of the new algorithm. Parallelization techniques have been utilized in spatial analysis to address performance and scale concerns (Lanthier *et al.* 2003, Zhang 2010, Anselin and Rey 2012, Rey *et al.* 2013). Parallelization of processes is now mostly performed over multicore CPUs and General Purpose Graphics Processing Unit (GPGPU) environments, rather than supercomputers and grid computing systems (Zhang 2010, Xia *et al.* 2011, Anselin and Rey 2012). These techniques harness the power of multiple cores in CPUs or GPUs in either single or several machines to boost the performance of a given spatial analytical method.

Although the ESP is based on vector representation of GIS data, it is possible to approximate the ESP under raster representation. The least cost path approach has been used to estimate of the ESP in robotics and video game raster environments (Mitchell

1988, Yap 2002, Nash *et al.* 2007, Daniel *et al.* 2010, Yap *et al.* 2011). The benefits of raster representation for path estimation and availability of the technique in many GIS software packages make its use rather common. At issue, however, is whether the ESP approximation in raster method is computationally prohibitive and/or is limited in the quality of the estimated path. Therefore, assessing how to address issues and differences between vector-based ESP and raster-based its approximation approaches is critical.

### 1.2. Research objective

Given the issues in derivation of the ESP using existing approaches, the major objective of this research is to develop a novel method that efficiently computes the ESP by intelligently evaluating given obstacles. This new algorithm will exploit spatial knowledge and GIS functionality to construct a graph that includes the ESP. Intelligent processing of given spatial information using GIS will be a key strategy of this algorithm. Furthermore, there are two secondary objectives of this research: utilizing high performance computing techniques, more precisely parallelization approaches, to address performance and scale issues in big data environment; and assessment and comparison between direct ESP derivation and raster-based ESP approximation.

### 1.3. Organization of the research

This research is structured as follows. In Chapter 2, a new algorithm for the ESP derivation is proposed. The ESP problem is formalized mathematically. Existing methods for the ESP derivation are reviewed, and their limitations are presented and analyzed. A new algorithm for the ESP derivation, referred to as the convexpath algorithm, is



proposed and its optimality is proved. The computational efficiency of visibility graph, local visibility graph, and convexpath is compared with an empirical application.

Chapter 3 relaxes a limiting assumption of the convexpath algorithm for the utilization of convexpath for general ESP problems. The convexity assumption of convexpath, which limits the relative location of nodes in the resulting graph and obstacles, is relaxed to extend the applicability of convexpath. Line-polygon overlay is utilized for construction of a subgraph that guarantees the ESP. Two wayfinding applications are considered to assess the efficiency of improved convexpath.

In Chapter 4, a spatial filtering technique is developed to enhance the performance of the convexpath algorithm in a high density obstacle environment. Although convexpath efficiently produces the ESP, high-density obstacle problems are computationally more demanding, and this results in the degradation of algorithm performance. A spatial filtering technique is proposed for improving computational efficiency while still guaranteeing derivation of the ESP. A wayfinding application results in ASU campus is presented for measuring the improvement.

Chapter 5 develops a parallelized version of the convexpath algorithm to boost the computational performance of convexpath in a big data environment. The convexpath algorithm utilizes several computationally intensive spatial operators iteratively for derivation of the ESP. For parallelization, these steps are restructured. Efficiency gains from this parallelization approach are assessed in wayfinding applications with different obstacle settings.

In Chapter 6, raster representation for ESP approximation techniques are reviewed and compared to vector-based ESP derivation approaches. The least cost path ESP approximation is widely used for route planning of robots and virtual objects. Although the least cost path has several benefits, issues arise regarding to computational efficiency and approximation quality. Assessment is conducted with a wayfinding application using the convexpath algorithm and the least cost path approach.

In the last chapter, Chapter 7, the research results of this dissertation are summarized and conclusions follow. Also, future research directions are suggested.

## CHAPTER 2

### EFFICIENT MEASUREMENT OF CONTINUOUS SPACE SHORTEST DISTANCE AROUND BARRIERS\*

As mentioned in the previous chapter, the objective of this dissertation is developing a novel algorithm for the efficient derivation of the ESP. This chapter formalizes the ESP problem mathematically, and develops a new algorithm for the ESP derivation, providing essential proofs. The new method, convexpath algorithm, exploits spatial knowledge and GIS functionality to construct a graph efficiently.

#### 2.1. Introduction

Proximity and distance are perhaps cornerstone features of spatial analysis. The appropriate distance measure or metric is essential for best reflecting movement behavior, closeness and general spatial relationships. One only need examine any spatial analysis text to observe the significance of distance as it is central to almost all developed and applied methods, models or approaches (e.g. Bailey and Gatrell 1995, Fischer and Getis 1997, Church and Murray 2009, Anselin and Rey 2010, O'Sullivan and Unwin 2010, Rogerson 2010, de Smith *et al.* 2012).

A key issue, of course, is how to measure distance. There are several widely used distance measures in spatial analysis: rectilinear, Euclidean, and network distance.

---

\* This chapter represents a slightly revised version of a paper published in *International Journal of Geographical Information Science*, co-authored with Dr. Alan T. Murray.

Euclidean distance has continued to be a favorite because it is simple, intuitive and easy to apply. If detailed road network characteristics are important, then network based distance could be considered, especially in cases where actual movements or flows occur on a transportation system. However, network datasets are sometimes not available in certain situations. In developing countries or rural areas, network data could be incomplete, insufficient or non-existent (Yao *et al.* 2012). Moreover, there could be limitations in purchasing or acquiring such data. When this is the case, Euclidean distance is widely used as a surrogate for proximity, and in many cases it has proven sufficient with high correlation observed between Euclidean and actual distance (Phibbs and Luft 1995, Fone *et al.* 2006, Jones *et al.* 2010, Carling *et al.* 2012, Cudnik *et al.* 2012). For rural areas, especially in developing countries, Euclidean distance is often the only possible choice because there is no existing road network data or actual movement is not restricted to roads (Stock 1983, Oppong and Hodgson 1994).

Of course, Euclidean distance is limited in many ways when used as a proxy for network or actual travel distance as no generic metric would be expected to be accurate or correct in all or even most cases. The reasons are that local nuances are likely varying and also obstacles that hinder directions of travel exist, such as rivers, mountains, coastlines, etc. as well as airports, military installations, etc. Though there may be some exceptions, these obstacles generally do not allow travel through them. A generic metric like Euclidean distance would therefore be challenged to capture or approximate travel movement behavior with any degree of confidence as it ignores the existence of obstacles/barriers (Martin *et al.* 2002, Jordan *et al.* 2004).

Euclidean distance is important for many aspects of spatial analysis and planning. However, barriers are problematic and likely bias results in many ways, such as service areas of certain types of public facilities (Carling *et al.* 2012). To overcome drawbacks of Euclidean distance, extensions of this metric can reflect spatial patterns more representative of actual travel behavior. Measuring Euclidean distance in the presence of obstacles has been referred to as the Euclidean shortest path (ESP) in computational geometry (Guibas and Hershberger 1989, Hershberger and Suri 1993, Mitchell 1999). It is therefore a recognized problem, with considerable attention focused on its solution. What is lacking to date is a formal mathematical specification of the problem. Further, implementation in a GIS environment provides many operational and efficiency benefits to a range of spatial analytical methods that must rely on metrics like Euclidean distance.

The aim of this paper is to formalize and solve the Euclidean shortest path problem to support various spatial analytical methods within a GIS environment. A mathematical problem formulation is presented to account for the presence of barriers. A solution technique based on the notion of a convex hull is introduced and operationalized in a commercial GIS. Empirical results are presented for analysis in an urban region. The paper ends with discussion and concluding comments.

## 2.2. Background

As mentioned previously, proximity and distance are central to virtually all spatial analytical methods. Bailey and Gatrell (1995), Fotheringham *et al.* (2000), de Smith *et al.* (2012), O'Sullivan and Unwin (2010) and Rogerson (2010) are popular texts illustrating the ubiquitous nature of proximity and distance in a range of spatial analysis approaches,

including spatial autocorrelation, geographically weighted regression, point pattern analysis, geocomputation, spatial interpolation, and exploratory spatial data analysis. A review of Church and Murray (2009) suggests that most location models are dependent on proximity and distance in some way as well. For many reasons the Euclidean metric is a popular and oft relied upon method for deriving proximity and distance. While there are many application domains that could be discussed, healthcare planning is one where there is much interest in the use and appropriateness of Euclidean distance in the analysis of accessibility (Phibbs and Luft 1995, Fone *et al.* 2006, Haynes *et al.* 2006, Jones *et al.* 2010, Cudnik *et al.* 2012). Some have advocated the use of Euclidean distance in urban and/or rural settings (Phibbs and Luft 1995, Jones *et al.* 2010), while others have explored its appropriateness in various contexts (Martin *et al.* 1998, Martin *et al.* 2002, Jordan *et al.* 2004, Higgs 2009, Cudnik *et al.* 2012).

Unobstructed travel is often assumed in most continuous space location models. Obstacles/barriers therefore present a problem because travel between two locations is dependent upon spatial structure. This means that approaches or models that ignore obstacles when they do in fact exist unintentionally introduce errors and/or biases in results, producing incorrect objective values and solutions that likely are not optimal. For example, in non-uniformly distributed rural area, Euclidean distance was found to cause sub-optimal service areas when compared with network distance and travel time (Carling *et al.* 2012). To address this, continuous space models must better represent a study region, and this requires travel obstacles to be explicitly considered (Klamroth 2001a, Bischoff and Klamroth 2007). Katz and Cooper (1981) were among the first to undertake such consideration in location modeling, introducing the barrier restricted Weber

problem. Subsequent work has followed to solve this problem (Aneja and Parlar 1994). Problem extension and solution has continued as well (Larson and Sadiq 1983, Batta *et al.* 1989, Klamroth 2001b). This location modeling work highlights the significance of obstacles/barriers. As GIS has various important roles in location modeling (Church 2002, Murray 2010), a GIS-based approach that exploits spatial knowledge about obstacles in a solution approach is very appealing. Further, it provides context for which shortest distance paths likely are useful in broader modeling efforts.

Research focused on Euclidean shortest paths emerged in computational geometry, with a number of solution techniques proposed that address obstacles/barriers. The most prominent approach is the visibility graph, which connects all mutually visible vertices in a given area (Lozano-Pérez and Wesley 1979). Extensive effort has been devoted to computational efficiency issues in the construction of the visibility graph (Welzl 1985, Asano *et al.* 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996, Kim *et al.* 2004, Zhang *et al.* 2005, Gao *et al.* 2011, Li *et al.* 2011). Another utilized solution approach for the ESP is the shortest path map, which discretizes an area based on shortest paths from a source point (Mitchell 1989, Hershberger and Suri 1993, Mitchell 1996, Hershberger and Suri 1999). For the case where there are complex polygon barriers, Voronoi diagram (Papadopoulou and Lee 1998) and funnel sequence approaches (Ghosh and Mount 1991) have been developed. Most of these approaches are not implemented in GIS environments. Further, existing approaches typically deal with the entire region, making the methods inefficient in various ways. Even though some research has examined the local visibility graph for more efficient search (Kim *et al.*

2004, Gao *et al.* 2011, Li *et al.* 2011), its efficiency and potential for integration in GIS is not particularly promising.

Euclidean distance is a relatively popular measure used in location and spatial analysis, but its appropriateness as a surrogate for actual travel distance/time is questionable when one considers obstacles/barriers that inhibit certain travel routes. Researchers have addressed various problem nuances and solution techniques. Further, a range of methods has been developed to solve the ESP. However, specialized computational geometry techniques for the ESP have limited efficiency when implemented for spatial analysis in a GIS environment. This is due to a lack of explicit mathematical specification of the problem, but also a lack of spatial perspective.

### 2.3. Problem formalization

As discussed previously, of interest in this chapter is mathematically formulating and solving the Euclidean shortest path (ESP) problem in the context of spatial analysis. This problem involves identifying the shortest distance/pathway between two locations that avoids all obstacles/barriers that impede travel. If there were no obstacles between points A and B, then Euclidean distance represents shortest length path. The presence of obstacles between A and B means that direct, straight line travel is not possible. The shortest distance/path between the two points that avoids crossing through any obstacle will necessarily involve a route comprised of one or more intermediate points. The issue then is to determine the necessary intermediate points. This is the so called ESP. While described in the literature, the ESP has not been explicitly formulated mathematically. Consider the following notation:



$l$  = index of intermediate points

$(\tilde{x}_l, \tilde{y}_l)$  = coordinates of intermediate point  $l$

$p$  = number of intermediate points

The decisions associated with the shortest path between A and B involve finding the number of intermediate points,  $p$ , and their location through which the path is to be routed. Formally, this is:

$$\text{Minimize} \quad \sum_{l=1}^{p+1} \sqrt{(\tilde{x}_{l-1} - \tilde{x}_l)^2 + (\tilde{y}_{l-1} - \tilde{y}_l)^2} \quad (1)$$

where  $(\tilde{x}_0, \tilde{y}_0) = (x_A, y_A)$ ,  $(\tilde{x}_{p+1}, \tilde{y}_{p+1}) = (x_B, y_B)$  and  $(x_A, y_A)$  and  $(x_B, y_B)$  are the coordinates of points A and B, respectively. The straight line between two consecutive points,  $l$  and  $l+1$ , is required to not cross any obstacle. Without doubt, this is not a trivial problem, and not readily solvable as presented in (1). Locating a number of points in continuous space while satisfying several conditions is a formidable task. One must locate intermediate points without knowing the exact number of points. As this is a continuous space problem, anywhere in the given study region is a potential location for intermediate points except the interior of obstacles. Further, the constraining conditions are challenging to impose, as all intermediate points have to be located such that connected line segments avoid intersecting all obstacles.

In a GIS environment, obstacles in a study region are represented as polygons.

The formal specification of the obstacles is as follows:

$k$  = index of obstacles (entire set  $K$ )

$$\Omega_k = \{(\hat{x}_{k1}, \hat{y}_{k1}), (\hat{x}_{k2}, \hat{y}_{k2}), (\hat{x}_{k3}, \hat{y}_{k3}), \dots, (\hat{x}_{kn_k}, \hat{y}_{kn_k})\}$$

$n_k$  = number of polygon vertices describing obstacle  $k$

For a polygon, it is implicit that two consecutive vertices are connected by a straight line. Without loss of generality, assume a polygon obstacle,  $k$ , is between points  $A$  and  $B$ . With this, we can define  $\Phi$  as the set containing all obstacle vertices as well as the points  $A$  and  $B$ ,  $\Phi = \{(x, y) \in \Omega_k, (x_A, y_A), (x_B, y_B)\}$ . The significance of this observation is that the feasible continuous space for intermediate points can be reduced. It has been proven that the intermediate points  $(\tilde{x}_l, \tilde{y}_l)$  of the shortest path will consist of points in  $\Phi$ , that is  $(\tilde{x}_l, \tilde{y}_l) \in \Phi$  (Viegas and Hansen 1985). The search for intermediate points, therefore, can be limited to  $\Phi$ . Thus, the problem now not only has a finite number of potential locations to consider, but only includes relevant portions of the study region. The difficulty is accounting for line segments between members of  $\Phi$  that do not intersect the interior of  $\Omega_k$  for any obstacle  $k$ . That is,  $i, j \in \Phi$  such that  $\bar{ij} \cap \text{int}(\Omega_k) = \emptyset$  for any  $k$ , where  $\text{int}(\cdot)$  is the interior region of an obstacle and  $\bar{ij}$  is the line segment connecting vertex  $i$  directly to vertex  $j$ .

An additional consideration is the regional boundary, and that it could inhibit travel. In general, it is assumed that travel outside the regional boundary is prohibited. Let  $R = \{(\bar{x}_1, \bar{y}_1), (\bar{x}_2, \bar{y}_2), \dots, (\bar{x}_m, \bar{y}_m)\}$  represent the boundary of the study region defined by  $m$  vertices. This would need to be accounted for in  $\Phi$  as well. Consider the following additional notation:

A, B = point of origin/destination

$\Gamma$  = set of impeding obstacles

$\Gamma_R$  = set of impeding vertices in  $R$

$\Phi$  = set of all impeding vertices and origin and destination points

$N_j$  = set of vertices in  $\Phi$  that can be connected to vertex  $j$  by an arc segment

$i, j$  = index of vertices in  $\Phi$

$\alpha_{ij}$  = distance from vertex  $i$  to  $j$

$$Z_{ij} = \begin{cases} 1 & \text{if arc from } i \text{ to } j \text{ is on the shortest path} \\ 0 & \text{otherwise} \end{cases}$$

In general,  $\Phi = \{(x, y) \in \Omega_k \mid k \in \Gamma, (x, y) \in \Gamma_R, (x_A, y_A), (x_B, y_B)\}$ . Also, particularly important in this notation is the set  $\Gamma$ , because it consists of only the obstacles that impede travel between given points A and B, not the entire set of obstacles in the study region. Impeding obstacles are not only those that directly inhibit a straight line segment, but also indirectly impeding obstacles that hinder possible pathways. Moreover, if there is any impeding portion of the regional boundary between points A and B or possible pathways, this is included in  $\Gamma_R$ . By applying a restricted search method, irrelevant obstacles and insignificant portions of the regional boundary do not waste computing effort. An important issue here is techniques for detecting direct and indirect impeding obstacles and relevant regional boundary vertices,  $\Gamma$  and  $\Gamma_R$ , that utilize spatial knowledge.

For deriving the shortest path, a graph of the vertices in  $\Phi$ ,  $G$ , can be constructed by linking each vertex to members of the set  $N_j$ . The set  $N_j$  for each vertex in  $\Phi$  consists of vertices that can be connected without intersecting the interior of obstacles or outside of the regional boundary. Thus,  $G$  is a network that represents all feasible path segments to travel from A to B, among which Viegas and Hansen (1985) proved the shortest distance will be found. As will be evident in the next section, there are many graphs  $G$  that are possible. We are interested in the most efficient graph  $G^*$ , where  $G \subset G^*$ . Such an efficient graph for the problem is shown in Figure 2.1a. The regional boundary contains 21 vertices, and three obstacles (16 vertices describe the obstacles). The resulting graph contains 13 arcs connecting 12 vertices.

With the above pre-processing, notation, parameters and decision variables, a formulation of the ESP can be structured based on the graph  $G$  (or  $G^*$ ), for beginning and ending locations A and B. This amounts to a shortest path problem in a network:

$$\text{Minimize} \quad \sum_j \sum_{i \in N_j} \alpha_{ij} Z_{ij} \quad (2)$$

$$\text{Subject to:} \quad \sum_{j \in N_A} Z_{Aj} = 1 \quad (3)$$

$$\sum_{j \in N_B} Z_{iB} = 1 \quad (4)$$

$$\sum_{i \in N_k} Z_{ik} - \sum_{j \in N_k} Z_{kj} = 0 \quad \forall k, k \neq A, B \quad (5)$$

$$Z_{ij} = \{0,1\} \quad \forall i, j \quad (6)$$

The objective function (2) is to minimize the total length of line segments that connect the given points, A and B. Constraint (3) and (4) stipulate flow from a point of origin, A, and to a point of destination, B. Constraint (5) ensures conservation of flow for each

intermediate vertex except the point of origin and destination. Constraint (6) limits decision variables to be binary.

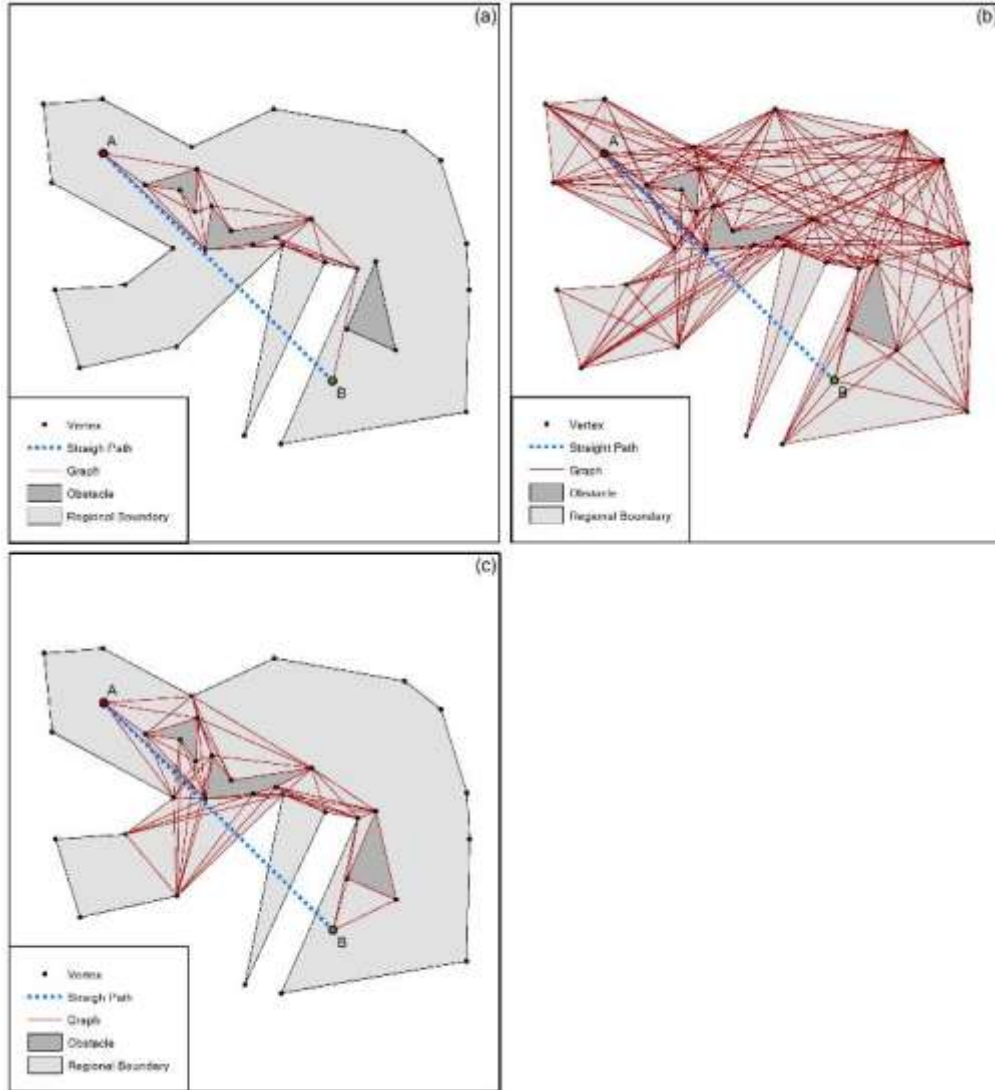


Figure 2.1. Line segments between obstacles and regional boundary vertices:  
(a) Graph  $G^*$ ; (b) Visibility graph; (c) Local visibility graph

## 2.4. Potential solution approaches

As mentioned previously, several solution approaches for the ESP have been developed. Perhaps the most popular methods are visibility graph and local visibility graph. A fundamental concept of the visibility graph is linking mutually visible vertices to each other. A visibility graph is constructed by evaluating whether a line segment is possible between each pair of vertices in the study region as well as beginning-ending points (Lozano-Pérez and Wesley 1979). Of course at issue is whether a straight line (line segment) between two points does not intersect any obstacle and remains in the study region. If so, such points are considered visible to each other, and the line segment is included as an arc of the visibility graph. With the visibility graph of nodes and arcs,  $VG$ , the shortest path can be determined using a shortest path algorithm.

An important question is how  $VG$  relates to the above described problem, and its associated characteristics. In terms of resulting graphs, in general  $VG \approx G$ . One major difference is that  $VG$  includes all vertices of the obstacles and the region boundary. Thus,  $\Phi^{VG} = \{(x, y) \in \Omega_k \ k \in K, (x, y) \in R, (x_A, y_A), (x_B, y_B)\}$ , assuming only one origin-destination pair for comparison purposes. It should be noted, however, that  $VG$  typically includes all considered origin-destination pairs, whereas the above problem description has been simplified for a single origin-destination pair. The significance of the  $VG$  distinction is that  $|\Phi^{VG}| > |\Phi|$ . That is, the number of vertices in the  $VG$  is notably larger than  $G^*$ , because  $G^*$  only requires impeding obstacles and relevant portions of the boundary. While there has been attention devoted to reducing the number of edges in  $VG$  (Rohnert 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996), all approaches

effectively utilize most vertices in the study region. What is new, unique and different in this research is the recognition that all obstacles and the entire regional boundary need not be considered when evaluating a given origin-destination pair. Spatial knowledge can be exploited to select only relevant obstacles and portions of the regional boundary.

The comparative difference is highlighted in Figure 2.1b. Recall that there are 37 vertices in total. Figure 2.1 suggests that for an origin-destination pair,  $VG$  is much larger than  $G^*$ . This is due to the substantially larger number of identified arcs. In this case,  $VG$  has 226 arcs (Figure 2.1b) while  $G^*$  has only 13 arcs (Figure 2.1a).

Another potential solution approach for the ESP is the local visibility graph and it differs in noteworthy ways from the visibility graph. A graph is generated in a similar fashion as the visibility graph, but the local visibility graph attempts to filter obstacles (see Zhang *et al.* 2005). By utilizing several spatial queries for a given origin/destination pair (Zhang *et al.* 2005, Gao *et al.* 2011, Li *et al.* 2011), the local visibility graph tries to exploit proximity-based information. Once relevant/impeding obstacles are identified, a local visibility graph,  $LVG$ , is generated by evaluating visibility between pairs of vertices. A significant concern, however, is how to detect relevant/impeding obstacles. Zhang *et al.* (2005) suggest a circle based search method for spatial query that can be applied for the ESP. A search circle is generated centered on the midpoint of two given points; its diameter is the Euclidean distance between the two points. Any obstacles intersecting the circle are considered relevant, forming the set  $\Gamma^{LVG}$ . Thus,  $\Gamma^{LVG} \subseteq K$ . Most of the local visibility graph literature does not consider issues of intersection with the regional boundary. However, the set  $\Gamma_R^{LVG}$  of vertices in  $R$  within the search circle can be assumed

in order to remain consistent with the above discussion. Thus,  $\Phi^{LVG} = \{(x, y) \in \Omega_k \mid k \in \Gamma^{LVG}, (x, y) \in \Gamma_R^{LVG}, (x_A, y_A), (x_B, y_B)\}$ . The local visibility graph has a benefit of reduced size compared to the visibility graph, because  $|\Phi^{LVG}| \leq |\Phi^{VG}|$ . However, there are several drawbacks to the local visibility graph. First, the local visibility graph possibly violates a fundamental constraint of ESP by failing to detect indirectly impeding obstacles. Such a case can happen if the size of the obstacle is larger than the search circle. Second, the reduction in graph size is not necessarily substantial. Proximity-based filtering methods possibly select irrelevant obstacles for  $\Gamma^{LVG}$ . The likelihood increases as the distance between two points increases. In extreme cases, it is possible that the search circle covers most of the study region, if the origin and destination are on opposing sides of the region. All local visibility graph methods have similar limitations in detecting impeding obstacles (Gao *et al.* 2011, Li *et al.* 2011). Furthermore, the local visibility graph is equally limited in detecting impeding regional boundary vertices. In fact, there is not any discussion/recognition in most local visibility graph methods about regional boundary issues.

Returning to the example shown in Figure 2.1, the comparative difference for the *LVG* can be seen. Figure 2.1c depicts the *LVG* in this case with 94 arcs. This is fewer than the 226 arcs needed in the *VG* (Figure 2.1b) but more than the 13 arcs for  $G^*$  (Figure 2.1a).

While the visibility graph and the local visibility graph are popular for solving the ESP, limited spatial knowledge is utilized. Further, in the latter case, significant problems could be encountered that would produce invalid results if the *LVG* is relied upon. It is



possible to exploit spatial knowledge, and GIS offers much potential for this as well as operational benefits for general usage and application in spatial analysis.

## 2.5. Deriving an efficient graph

In previous sections notation was defined associated with an efficient graph,  $G^*$ , through which the optimal ESP can be found. Figure 2.1 supports efficiency inferences for the example problem, as  $G^*$  is substantially smaller in size than the  $VG$  and  $LVG$ . In this section, details on how to efficiently find  $\Gamma$  and  $\Gamma_R$ , and derive  $G^*$  using GIS functionality are provided. That is, we would like the smallest and most efficient graph possible,  $G^*$ . The most important consideration for efficient solution of the ESP is detecting direct and indirect impeding obstacles in the set  $\Gamma$ . Obstacles that impede a straight line segment can be easily found. However, more complex spatial knowledge is required to find indirect impeding obstacles. Also important is addressing impeding vertices on the regional boundary.

The convex hull is an important concept in computational geometry. It can be an effective way to exploit spatial knowledge for filtering obstacles and regional boundary vertices. Assume there is a finite point set  $N$ . A convex hull is the intersection of all convex sets containing  $N$ , or the smallest and unique convex polygon that contains all points (see de Berg *et al.* 2008). By definition, the length of the boundary of a convex hull is the minimum possible. Further, we assume here that A and B are on the convex hull's boundary.

**Theorem 1:** The optimal Euclidean shortest path between two points separated by a single contiguous obstacle will be on the convex hull boundary.

**Proof:** Consider two points, A and B, and obstacle  $k$  that impedes straight line travel between A and B., as shown in Figure 2.2. Without loss of generality, the regional boundary is ignored as it is sufficiently distant from these spatial objects and does not impact travel in any way. Suppose there exists a point outside the convex hull through which a shorter path exists, and A and B are on the hull. If this were true, the convex hull would not be the minimum length possible. This would contradict the definition of a convex hull, so is not possible. Alternatively, suppose there exists a point inside the convex hull through which a shorter path exists. Such a situation would necessarily create a non-convex path around the obstacle. By the triangle inequality, this would increase the distance traveled to get from A to B compared to the shortest distance on the convex hull. This too is a contradiction, so any point travelled through on the Euclidean shortest path must be along the convex hull when A and B are on the convex hull. ◻

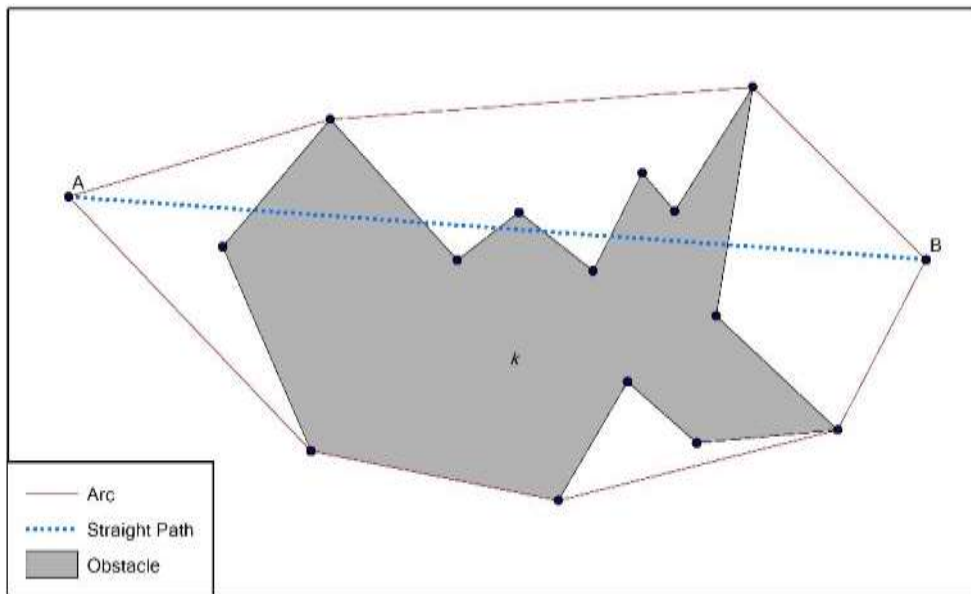


Figure 2.2. Convex hull for two points and an obstacle

The significance of Theorem 1 is that an algorithm is possible for selectively identifying vertices and arcs to include in a graph through which the optimal Euclidean shortest path may be found. However, it should be noted that here we consider cases only that origin and destination points are on the convex hull's boundary. Recall that  $\Omega_k = \{(\hat{x}_{k1}, \hat{y}_{k1}), (\hat{x}_{k2}, \hat{y}_{k2}), \dots, (\hat{x}_{kn_k}, \hat{y}_{kn_k})\}$  reflects the vertices of obstacle  $k$  and  $\Phi = \{(x, y) \in \Omega_k, (x_A, y_A), (x_B, y_B)\}$ . Viegas and Hansen (1985) proved that the optimal Euclidean shortest path is comprised of arcs in  $VG$  obtained from  $F$ . However, it is clear from Theorem 1 that a much smaller graph,  $G^*$ , is possible, one that includes the optimal Euclidean shortest path. Again, Figure 2.1 demonstrated empirically the comparative reduction in graph size possible.

As the boundary of the convex hull contains the shortest path around an obstacle, it can be utilized for detecting indirectly impeding obstacles to be included in  $\Gamma$ . Let there be several obstacles between points A and B, as shown in Figure 2.3a. In this case, only  $k_1$  and  $k_2$  impede the straight path. However, the convex hull for each direct impeding obstacle intersects one or more other obstacles (Figure 2.3b). If hull line segments intersecting other obstacles are replaced with associated convex hulls, a combined set of hulls results (Figure 2.3c and d). This then provides an approach for detecting direct and indirect impeding obstacles to be included in the set  $\Gamma$  (and  $\Gamma_R$ ).

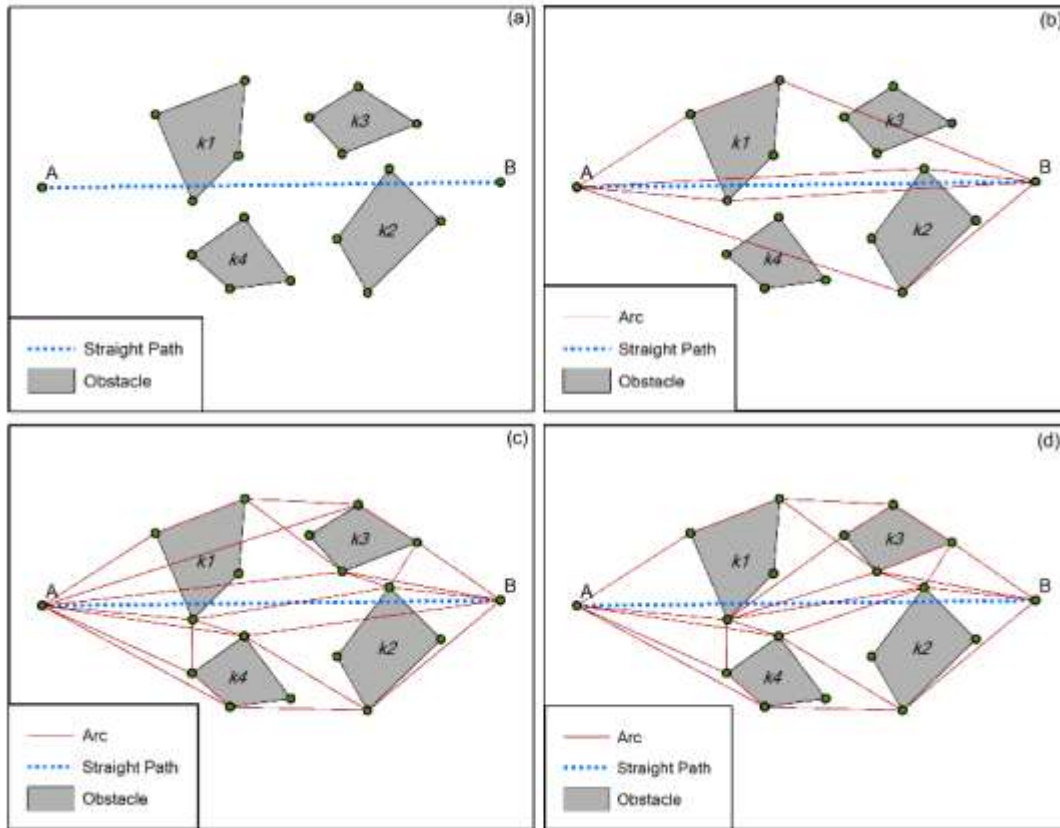


Figure 2.3. Combined set of multiple convex hulls: (a) Detection of direct impeding obstacles; (b) Separate convex hulls for each direct impeding obstacle; (c) Combined set of convex hulls for direct impeding obstacles; (d) Combined set of convex hulls for direct and indirect impeding obstacles.

Based on convex hulls, an efficient algorithm for solving the ESP is proposed, referred to here as *convexpath*. Convexpath derives a graph  $G^*$  enabling the shortest path/distance between two points to be found by utilizing a series of the convex hulls for impeding obstacles/vertices. The steps of the convexpath algorithm are detailed in Figure 2.4. Convexpath evaluates an origin-destination pair using a straight line between them. If there are any obstacles intersecting the straight line, they are considered as directly impeding obstacles, and included in set  $\Gamma$ . A convex hull of origin-destination points and

each obstacle in  $\Gamma$  is generated in an iterative fashion. If any arc in the convex hulls intersects with another obstacle, that arc is substituted by an additional convex hull. If the obstacle is not in  $\Gamma$ , it is considered indirect impeding and included in  $\Gamma$ . Non-crossing line segments from the origin and destination to the vertices of the segment convex hull are added. This continues until there are no more impeding obstacles. If the initial straight line or arcs of the convex hulls intersect with the regional boundary, a boundary induced obstacle results. Boundary induced obstacles are considered impeding vertices,  $\Gamma_R$ . The vertices in  $\Gamma$  and  $\Gamma_R$  define the resulting graph  $G^*$ , and with this the shortest path/distance is calculated using Dijkstra's shortest path algorithm. The entire process is depicted as pseudo-code in Figure 2.5, and all operations in convexpath use standard GIS functions.

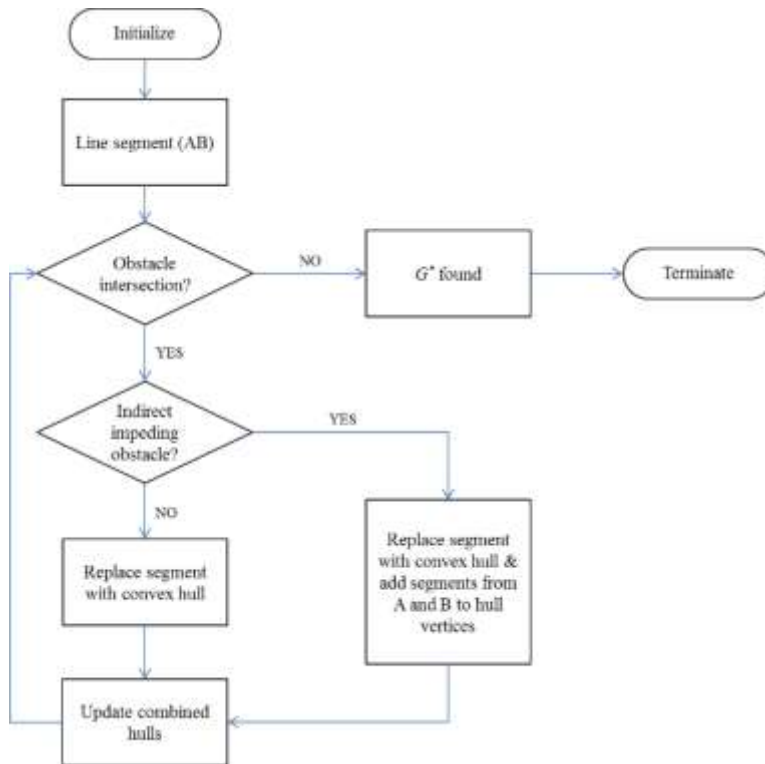


Figure 2.4. Flowchart of  $G^*$  generation

```


function convexpath
generate straight line segment for origin-destination points
if straight line crosses obstacle(s) then
    generate convex hull for each intersecting obstacle
    while segments cross obstacles do
        select segment impeded by obstacle
        generate convex hull for intersecting obstacle and segment and end points
        replace segment with convex hull
    if indirect obstacle then
        generate non-crossing segments from origin and destination to convex hull
    identified vertices and segments are nodes and arcs in graph  $G^*$ 
    calculate shortest path in  $G^*$  using shortest path algorithm
    return shortest distance
else
    return Euclidean distance

```

Figure 2.5. Pseudo code for the convexpath algorithm

**Theorem 2:** The optimal Euclidean shortest path between two points with multiple obstacles inhibiting travel is contained in graph  $G^*$ .

**Proof:** Lozano-Pérez and Wesley (1979) suggested (and Viegas and Hansen 1985 give proof) that the shortest path from A to B goes through one or more vertices of obstacles  $k \in K$ . Thus, the search may be limited to  $\Phi = \{(x, y) \in \Omega_k \mid k \in K, (x_A, y_A), (x_B, y_B)\}$ . What remains is to prove is that no vertices or arcs on the Euclidean shortest path have been omitted in  $G^*$ . Suppose vertex  $v$  is on the optimal shortest path, but  $v \notin G^*$ . Given A, B and obstacle  $k_l$ , from Theorem 1 there would be two potential shortest pathways on the convex hull for  $\{A, B, k_l\}$  if A to obstacle  $k_l$  is not

impeded by another obstacle. Call the last vertex on obstacle  $k_l$  for each pathway  $\alpha$  and  $\gamma$ . Note that  $\alpha, \gamma \in G^*$  by convex hull algorithm. If  $\overline{\alpha B}$  and  $\overline{\gamma B}$  do not intersect another obstacle, i.e.  $\overline{\alpha B} \cap \text{int}(k') = \emptyset$  and  $\overline{\gamma B} \cap \text{int}(k') = \emptyset$  for  $k' \in K$ , then  $v$  cannot be on the shortest path due to the triangular inequality as  $\overline{\alpha v} + \overline{vB} \geq \overline{\alpha B}$  and  $\overline{\gamma v} + \overline{vB} \geq \overline{\gamma B}$ . Alternatively, if  $\overline{\alpha B} \cap \text{int}(k') \neq \emptyset$  or  $\overline{\gamma B} \cap \text{int}(k') \neq \emptyset$  for another obstacle  $k''$ , then by Theorem 1 the shortest distance would be on the convex hull boundary for the intermediate vertex, B and the obstacle (e.g.,  $\{\alpha, B, k''\}$  and/or  $\{\gamma, B, k''\}$ ). Both cases contradict that vertex  $v, v \in G^*$ , could be on the Euclidean shortest path as the convexpath algorithm accounts for all possible shortest path options through the combined convex hulls. 

There are many significant aspects of the convexpath algorithm. First, it is finite in terms of the number of operations required. It will terminate after a finite number of iterations. Second, the resulting graph,  $G^*$ , contains the optimal Euclidean shortest path. Finally, the convex hull based approach is very efficient. As only some of the vertices are utilized for the convex hulls, the size of the  $G^*$  is smaller than  $G$ , as  $G^* \subseteq G$ .

The convexpath method exploits spatial knowledge to solve the ESP in a GIS environment. It finds  $\Gamma$  and  $\Gamma_R$  efficiently and precisely, and generates a minimal sized graph  $G^*$  for use in shortest path calculation.

## 2.6. Application results

To demonstrate the operational efficiency of the convexpath algorithm, a portion of the Tampa, Florida region is considered. The region is part of school districting work involving the author in an effort to reduce bus transportation costs. The interest here is

finding the optimal Euclidean shortest path (ESP) between an origin and destination. Six convex obstacles are in the interior of the regional boundary. The shortest distance between an origin and destination is sought that avoids obstacles. An origin and destination, the regional boundary, and obstacles are depicted in Figure 2.6. The convexpath, visibility graph, and local visibility graph approaches are implemented using C# .NET and ArcObjects 10. The analysis is carried on an Intel i5 personal computer (2.80GHz) with 12 GB memory.

For assessment, a number of different origins and destinations are considered. In total, the analysis examines 2,853 different origin-destination instances of the ESP. The results for the three methods are compared in terms of computing time and graph size (the number of vertices and arcs).

To illustrate differences between the three methods, the shortest path/distance for the problem is presented in Figure 2.6 based on the convexpath approach. In this case, the convexpath algorithm identified the graph,  $G^*$ , as 16 vertices and 20 arcs in size. In contrast, the visibility graph approach found  $VG$  with 1,010 vertices and 33,104 arcs and the local visibility graph approach determined  $LVG$  to be 117 vertices and 919 arcs in size. In percentage terms,  $VG$  is over 6,000% and  $LVG$  is over 600% larger than  $G^*$ . Such differences in graph size have direct implications for computation. The total computing time required for the convexpath algorithm was 4.03 seconds, including graph identification and shortest path solution. In contrast, the visibility graph approach requires 6,231.79 seconds and the local visibility graph approach requires 61 seconds. Again, in percentage terms, this translates to over 150,000% more total computing time



for the visibility graph approach and over 1,400% more total computing time for the local visibility graph approach when compared to the convexpath algorithm.

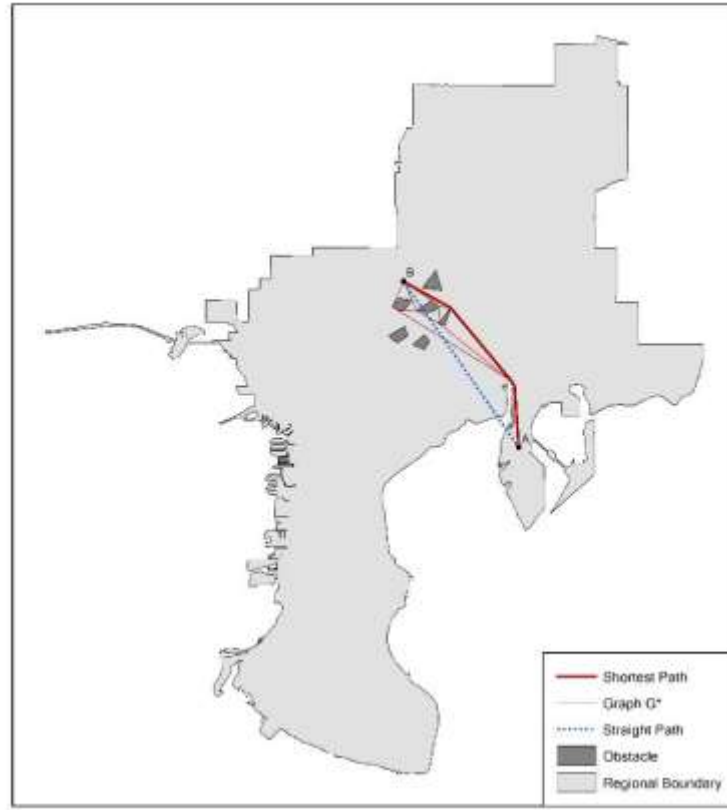


Figure 2.6. Application area, graph  $G^*$  and shortest path

Similar findings were observed for the other origin-destination pairs. Statistically, the 2,853 different cases are summarized based on observed minimum, mean and maximum values. This is reported for number of vertices, number of arcs, and total computing time. Figure 2.7 gives these findings for each method. In terms of graph size and computing time, the convexpath algorithm appears to perform well in most cases. The only exception is that there is a minimum case (Figure 2.7a) where graph size is the

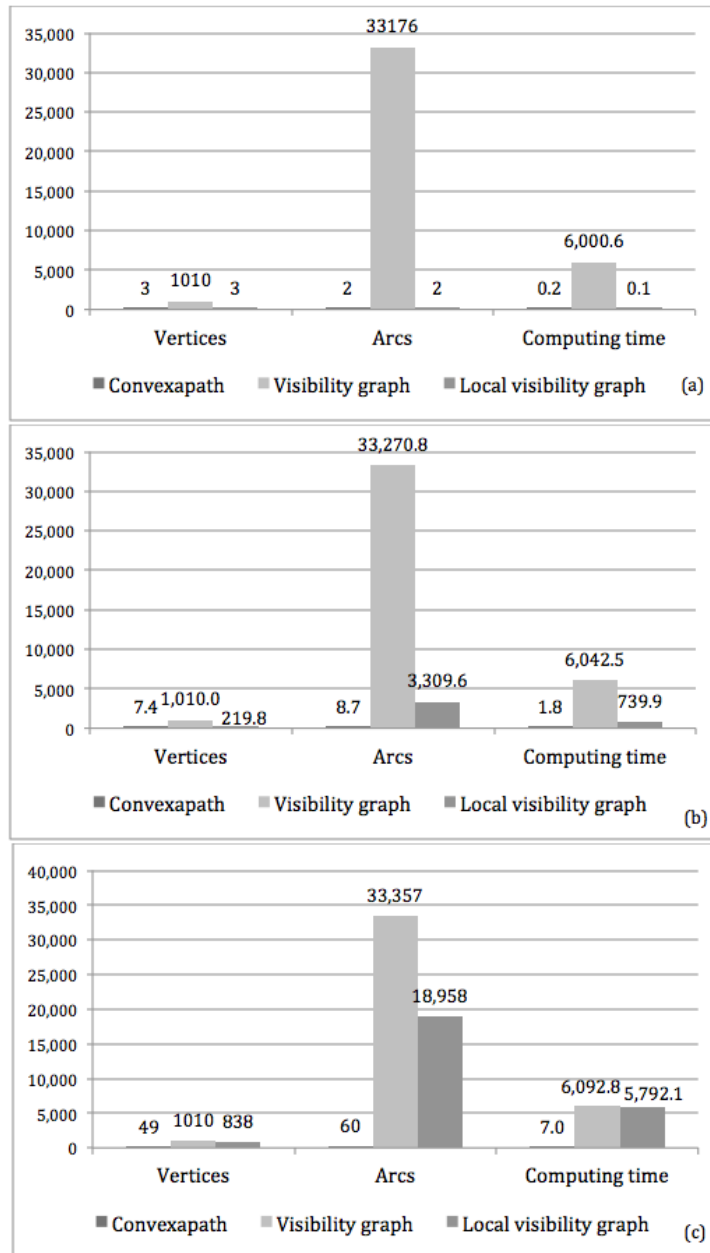


Figure 2.7. Application results: (a) Minimum; (b) Mean; (c) Maximum

same for the convexpath and local visibility graph approaches. Otherwise, the visibility graph always requires a larger graph and more time to process. For example, Figure 2.7b

illustrates that the visibility graph always had substantially larger graphs (33,270.8 arcs on average) and needed 6,042.5 seconds of total computing time on average. Whereas total computing time is 1.8 seconds on average for the convex path approach and 739.9 seconds for the local visibility graph approach. One final point is that the local visibility graph approach is unable to correctly identify the shortest distance in 19 of the 2,853 cases. This is due to the previously noted limitations associated with the proximity search circle.

## 2.7. Discussion and conclusions

The comparative results highlight substantial improvements in computational capabilities possible using the convexpath algorithm for solving the ESP optimally. The reason for this can be attributed to graph size, as the convexpath exploits spatial knowledge for intelligently detecting direct and indirect impeding obstacles as well as impeding regional boundary vertices,  $\Gamma$  and  $\Gamma_R$ . As a result, the graph through which a shortest path is found is significantly smaller than is possible using the visibility graph or local visibility graph approaches. By exploiting appropriate spatial knowledge in a GIS environment, the convexpath algorithm is able to solve the ESP efficiently and effectively.

It was noted in the paper that the visibility graph approach typically solves the ESP for a set of origins and destinations simultaneously. In the application considered here, it would be possible to apply the visibility graph approach to all origin-destination combinations (2,853 in total) at the same time. Doing so would require approximately 10,000 seconds, suggesting that spends about 3.47 seconds per case. Taking this into

account and summing the convexpath solution time for each of the 2,853 instances would total approximately 5,000 seconds. This is still about half of the total computation time compared to the visibility graph approach. For the local visibility graph approach, total computing time is approximately 2,083,585 seconds. Not particularly comparable to either of the other methods, and would not be a practical solution approach for large sized instance of the ESP. Moreover, the local visibility graph approach may not be able to actually find the optimal shortest path/distance in certain situations.

The problem of finding the shortest path between two points in the presence of obstacles is referred to as the Euclidean shortest path (ESP) problem. Numerous solution approaches have been developed for this problem. However, the ESP has not been formally defined to date. To address this, a mathematical formalization of the ESP as a spatial optimization problem was presented in this research. A new solution approach for the ESP was developed, named convexpath. By utilizing convex hulls, the convexpath algorithm exploits appropriate spatial knowledge for identifying direct and indirect impeding obstacles as well as impeding vertices in the region boundary. The convexpath algorithm constructs a minimal sized graph, and as a result is very computationally efficient both to identify and solve for a shortest path. The application results highlighted this effectiveness.

## CHAPTER 3

### EFFICIENT WAYFINDING IN COMPLEX ENVIRONMENT: DERIVATION OF A CONTINUOUS SPACE SHORTEST PATH\*

In Chapter 2, the convexpath algorithm is developed. However, it has one assumption that limits its applicability to general ESP problems: the convexity assumption. This chapter develops extension of the convexpath algorithm that relaxes the convexpath assumption. To address the issue regarding to the assumption, a line-polygon overlay operator is used to generate a subgraph that guarantees inclusion of the ESP.

#### 3.1. Introduction

In many situations, wayfinding tasks are conducted in complex environments with obstacles that inhibit travel. Travel movement of the blind in an urban area, routing for shipping and robots, and emergency evacuation from building structures can be considered instances of such travel, and research effort has been devoted to deriving an efficient path across a landscape (Lozano-Pérez and Wesley 1979, Golledge *et al.* 1998, Fagerholt *et al.* 2000, Qin *et al.* 2004, Kwan and Lee 2005, Bekker and Schmid 2006, Szymanski *et al.* 2006, Guven *et al.* 2012).

An important wayfinding task is finding the shortest path to a destination given the spatial configuration of an area. If movements were restricted to the transportation

---

\* This chapter represents a slightly revised version of a paper presented in *6th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, co-authored with Dr. Alan T. Murray.

network, the shortest path can be calculated using various network shortest path algorithms, but this requires a network composed of nodes and arcs. If this is not the case, a different solution approach is required as there are infinite number of pathways to consider. Euclidean straight line travel is the most widely used estimation, because of simplicity of calculation and no need for auxiliary data. However, Euclidean straight line travel generally fails to capture spatial nuisances, including obstacles such as mountains, rivers, coastlines, building structures and so on. To account for this, an alternative approach for the derivation of a valid and meaningful shortest path is required.

Finding the shortest path that avoids obstacles over continuous space is referred to as the Euclidean shortest path (ESP) problem in computational geometry (Guibas and Hershberger 1989, Hershberger and Suri 1993, Mitchell 1999). It is a well-known problem, with substantial attention having been paid to its solution. Most prominent are the visibility graph (Lozano-Pérez and Wesley 1979, Welzl 1985, Asano *et al.* 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996), local visibility graph (Kim *et al.* 2004, Zhang *et al.* 2005, Gao *et al.* 2011, Li *et al.* 2011), shortest path map (Mitchell 1999), and Voronoi diagram approach (Papadopoulou and Lee 1995, Papadopoulou and Lee 1998). However, existing solution techniques are limited in many ways due to the significant computational requirements necessary.

Hong and Murray (2013a) proposed a new method for solving the ESP based on GIS functionality and spatial knowledge. Their method, the so called convexpath, exploits spatial knowledge by utilizing the notion of a convex hull. The convexpath algorithm constructs a series of convex hulls for an origin and destination to find

impeding obstacles, and in the process better focuses attention on good the spatial possibilities for a potential shortest path. It has significant computational advantages over existing methods, making it appealing for real time travel support environments. However, convexity assumption potentially limits more general utilization of the convexpath algorithm for wayfinding tasks.

The aim of this research is to extend the convexpath algorithm for solving the ESP problem relaxing previous assumption of convexity. Optimality conditions for the new method are established, and details regarding implementation in a commercial GIS are given. Application results are presented to demonstrate the effectiveness of the approach. The paper ends with concluding comments.

### 3.2. Background

As mentioned in the previous section, the shortest path to a destination is an essential element in various wayfinding tasks. Route finding in robots and shipping requires the Euclidean shortest path for various reasons. For robots, Lozano-Pérez and Wesley (1979) proposed a method to calculate a collision-free shortest path using the visibility graph approach. Different solution techniques have been proposed, such as the ‘MAKLINK’ graph using midpoints of free links of buffered polygonal obstacles (Habib and Asama 1991, Qin *et al.* 2004), pheromone signals (Szymanski *et al.* 2006) and the biomimetic ‘slime mold’ strategy (Bhattacharya and Gavrilova 2007) for micro robot swarms. For shipping, several shortest path strategies have been suggested: utilizing the visibility graph or its variations (Fagerholt *et al.* 2000); applying Voronoi diagrams (Bhattacharya and Gavrilova 2007); utilizing buffers and minimum bounding rectangles

of the obstacles (Tsou 2010); and dividing the given area by a square grid (Chang *et al.* 2003, Bekker and Schmid 2006, Zhang *et al.* 2011).

Efficient wayfinding in the interior of the large buildings also requires a shortest path. Although the structure of buildings confines movements inside, it can be considered continuous space. Wayfinding strategies for movements in buildings has pursued different approaches compared to the ESP, such as constructing pre-defined graphs based on a fixed sensor network (Güven *et al.* 2012), center points of doors and corridors (Lee *et al.* 2010) and centerlines of corridors and major doors (Kwan and Lee 2005).

Wayfinding for the visually impaired can be divided into indoor and outdoor environments. For outdoor navigation, most approaches find the shortest path based on an existing network (Golledge *et al.* 1998, Loomis *et al.* 2001, Loomis *et al.* 2005, Wilson *et al.* 2007). In indoor environment, difficulty in self-positioning arises due to the impenetrability of a GPS signal through building materials. Therefore, many approaches have focused on new self-positioning strategies, such as RFID tags (Kulyukin *et al.* 2006), Wi-Fi (Riehle *et al.* 2008) and ultrasound location systems (Ran *et al.* 2004, Riehle *et al.* 2008, Kurata *et al.* 2011). Limited work explicitly considers obstacle-avoiding shortest paths.

The ESP problem has been extensively studied in computational geometry, as noted previously, with a number of well-developed solution approaches proposed. The most widely utilized method is the visibility graph. It was proposed by Lozano-Pérez and Wesley (1979) for a collision-free shortest path, and subsequent research has been followed (Welzl 1985, Asano *et al.* 1986, Rohnert 1986, Ghosh and Mount 1991,



Pocchiola and Vegter 1996). The visibility graph consists of vertices of obstacles, the regional boundary and the given origin/destination points, with arcs connecting mutually visible vertices (Lozano-Pérez and Wesley 1979, de Berg *et al.* 2008). The construction of the visibility graph is done by evaluating all vertices, including the origin and destination points, in the given region (de Berg *et al.* 2008). Only mutually visible vertices and their arc are included in the resulting graph. There are several sweep-based algorithms that can improve the complexity of graph construction. Research on reducing the total number of resulting arcs in the visibility graph has been pursued (Rohnert 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996). However, the efficiency of the visibility graph is inevitably limited as it still has to consider most vertices in the region, regardless of the location. Not all obstacles nor the entire regional boundary actually impede potential pathways between vertices. There is substantial room for improving efficiency by filtering out irrelevant obstacles and/or portions of the regional boundary. The challenge is how to do this efficiently and effectively.

The local visibility graph (LVG) approach results from such a consideration. By filtering out irrelevant obstacles, the local visibility graph tries to reduce the number of vertices considered in graph construction (Zhang *et al.* 2005). The local visibility graph is built only with select impeding obstacles for given intermediate points. Therefore, an important issue is techniques to filter out irrelevant obstacles. Proximity-based spatial knowledge has been utilized for the filtering process, such as static or dynamically changing search circles (Zhang *et al.* 2005, Gao *et al.* 2011, Li *et al.* 2011) and convex hull search (Kim *et al.* 2004). Although LVG methods archive some efficiency in graph construction, there are still several critical drawbacks or limitations as a solution approach

(Hong and Murray 2013a). First, the benefit of filtering obstacles is likely not significant for the LVG. Proximity-based filtering methods tend to include non-impeding obstacles, and this tendency increases as the distance between two points is increased. Second, it is possible that impeding obstacles are actually omitted, if the obstacles are larger than the search circles. Third, there is virtually no consideration of the regional boundary in the local visibility graph literature. Even though search methods can be easily extended to account for vertices of the regional boundary, such approaches still have similar limitations.

### 3.3. Euclidean shortest path

The ESP problem can be considered a spatial optimization problem for finding the shortest path between two points. Hong and Murray (2013a) formalized this problem mathematically. Consider a region with a number of obstacles, a regional boundary, and points A and B. The shortest path between points A and B would be:

$$\text{Minimize } \sum_{i=1}^{p+1} \sqrt{(\tilde{x}_{i-1} - \tilde{x}_i)^2 + (\tilde{y}_{i-1} - \tilde{y}_i)^2}$$

where  $(\tilde{x}_l, \tilde{y}_l)$  is the coordinate of intermediate vertices of the shortest path, with  $(\tilde{x}_0, \tilde{y}_0) = (x_A, y_A)$ ,  $(\tilde{x}_{p+1}, \tilde{y}_{p+1}) = (x_B, y_B)$ . Let  $K$  denote the set of obstacles and  $\Phi$  the set of vertices of  $K$  as well as origin/destination points. As Viegas and Hansen (1985) proved, the intermediate vertices of the shortest path will only consist of vertices of the obstacles. Also, vertices of the regional boundary, denoted as the set  $R$ , should be considered as well, because the regional boundary can impede potential pathways.

Of course only a portion of the vertices in the sets  $\Phi$  and  $R$  will comprise the actual shortest path between the two points. An important issue then is a method for finding actual impeding obstacles and the relevant portions of the regional boundary. A GIS-based filtering method that exploits spatial knowledge can be utilized for pre-processing. After appropriate and necessary vertices are found, a graph,  $G^*$ , can be constructed, and will contain the ESP. This approach is more computationally efficient than other potential solution techniques, such as the visibility graph and the local visibility graph.

#### 3.4. Convexpath algorithm

Hong and Murray (2013a) proposed a new method based on the concept of  $G^*$  for the ESP problem. They utilized the notion of a convex hull for finding impeding obstacles and significant portions of the regional boundary for constructing a graph  $G^*$ . The authors classified obstacles into three groups; direct impeding obstacles (DIOs), blocking a straight line path between two points; indirect impeding obstacles (IIOs) that obstruct potential pathways around DIOs; and boundary induced obstacles (BIOs) that consist of portions of the regional boundary blocking direct or indirect paths. They proved the ESP is always in the  $G^*$ .

The graph  $G^*$  is constructed using the following steps. First, generate a straight line between two given points, then evaluate obstacles with the straight line to find DIOs (Figure 3.1a). Second, create a convex hull for each DIO and origin and destination points. If any arcs in these convex hulls intersect DIOs, replace that arc with a new convex hull (Figure 3.1b). Third, any new obstacle intersecting arcs is designated as IIO,

and replaced by a new convex hull constructed from segment endpoints and the obstacle. Additionally, new line segments from the origin/destination points to vertices of the new convex hull are constructed. Repeat the third step until there are no intersections between arcs and obstacles (Figure 3.1c). Finally, convert resulting arcs to a graph  $G^*$  (Figure 3.1d). The ESP can be easily found using any of the shortest path algorithms applied to  $G^*$ .

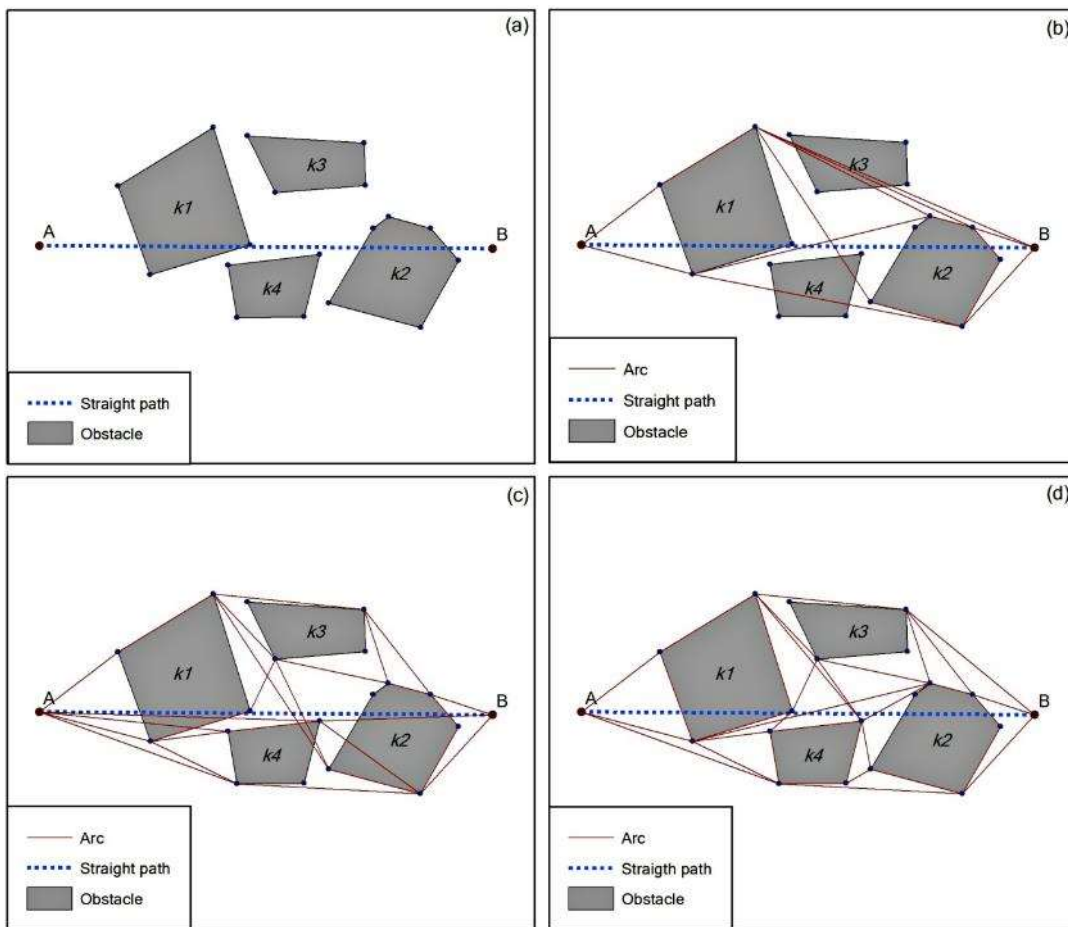


Figure 3.1. Steps of the convexpath

There are several benefits of the convexpath algorithm (Hong and Murray 2013a). A major benefit is its computational efficiency. By utilizing spatial knowledge, convexpath can find the ESP with higher efficiency than other existing solution approaches. Furthermore, it is based on GIS functionalities, so it can be easily integrated in a variety of spatial analytical methods.

However, the convexpath has one assumption that limits general utilization: the convexity assumption. Both origin and destination points have to be on the boundary of the resulting convex hulls. Such an assumption cannot hold in many wayfinding tasks. Therefore, a relaxation of the convexity assumption is important and necessary for broadly use and application.

### 3.5. Improving the convexpath algorithm

Let us assume there are two points and a single obstacle as shown in Figure 3.2. It is obvious the convexity assumption is not satisfied, and the convexpath algorithm cannot find the shortest path for the given points. We therefore propose an extension of the convexpath algorithm that relaxes the convexity assumption by dividing obstacles.

Consider point A and B and a single obstacle satisfying the convexity assumption as in Figure 3.3. The obstacle can be divided by a straight line between two points (Figure 3.3a). If a convex hull for the origin/destination point and each divided obstacle is carried out, then further processing is possible. After eliminating convex hull segments that intersect with original obstacle, connect remaining feasible pairs of the convex hull vertices and the origin/destination points (Figure 3.3b). The resulting graph contains a

convex hull for two points and the undivided obstacle. Therefore such a graph will contain the shortest path.

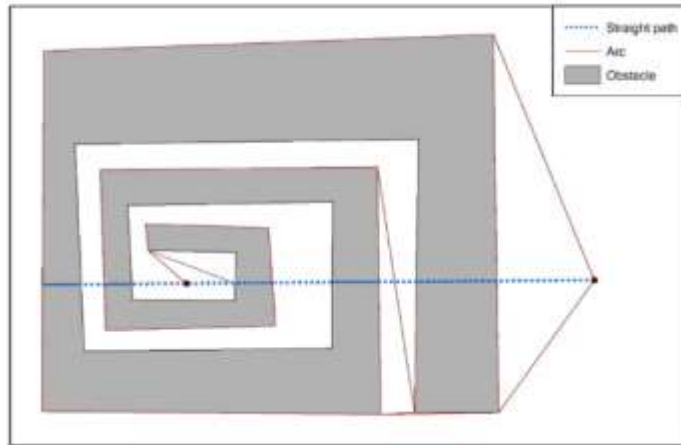


Figure 3.2. Violate convexity assumption

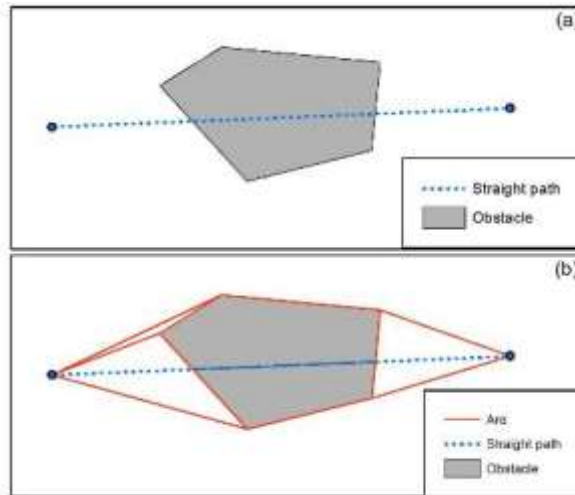


Figure 3.3. Dividing approach

This approach can be applied for the abovementioned situation, with an extended straight line for complete division of an impeding obstacle like Figure 3.3. A convex hull is generated for each divided obstacle and the origin/destination points, and convex hull arcs that intersect original obstacle are removed. Finally, connect all feasible vertices of remaining convex hulls to construct the resulting graph,  $Ge^*$ . The graph  $Ge^*$  will always contain the ESP for two points.

**Theorem 1:** The optimal Euclidean shortest path between two points blocked by a single polygonal obstacle will be on  $Ge^*$ .

**Proof:** Assume the two points and a single obstacle as in Figure 3.3 violating the convexity assumption. A series of convex hulls can be constructed using divided obstacle and two points by abovementioned steps. Vertices of the convex hulls form all possible pathways from the origin to the destination that is impeded by each sliced obstacle.

Therefore, the graph  $Ge^*$ , which contains all feasible paths between those vertices, will contain the shortest path between two points. Suppose the obstacle vertex  $k$ ,  $k \notin Ge^*$ , is forming one of the intermediate points of the ESP. If vertex  $k$  is between vertex  $a$  and  $b$ ,  $a, b \in Ge^*$ , then  $k$  cannot be on the shortest path given the triangle inequality,  $\overline{akb} > \overline{ab}$ .

Q.E.D.

The steps for deriving  $Ge^*$  for multiple obstacles are detailed in Figure 3.4. Evaluate the obstacles with a straight line between two points, and find DIOs. Check whether the convexity assumption holds. If the assumption holds, follow the steps described in Hong and Murray (2013a) If not, select obstacles and divide them using an extended straight line between two points (Figure 3.4a). Then a convex hull for each

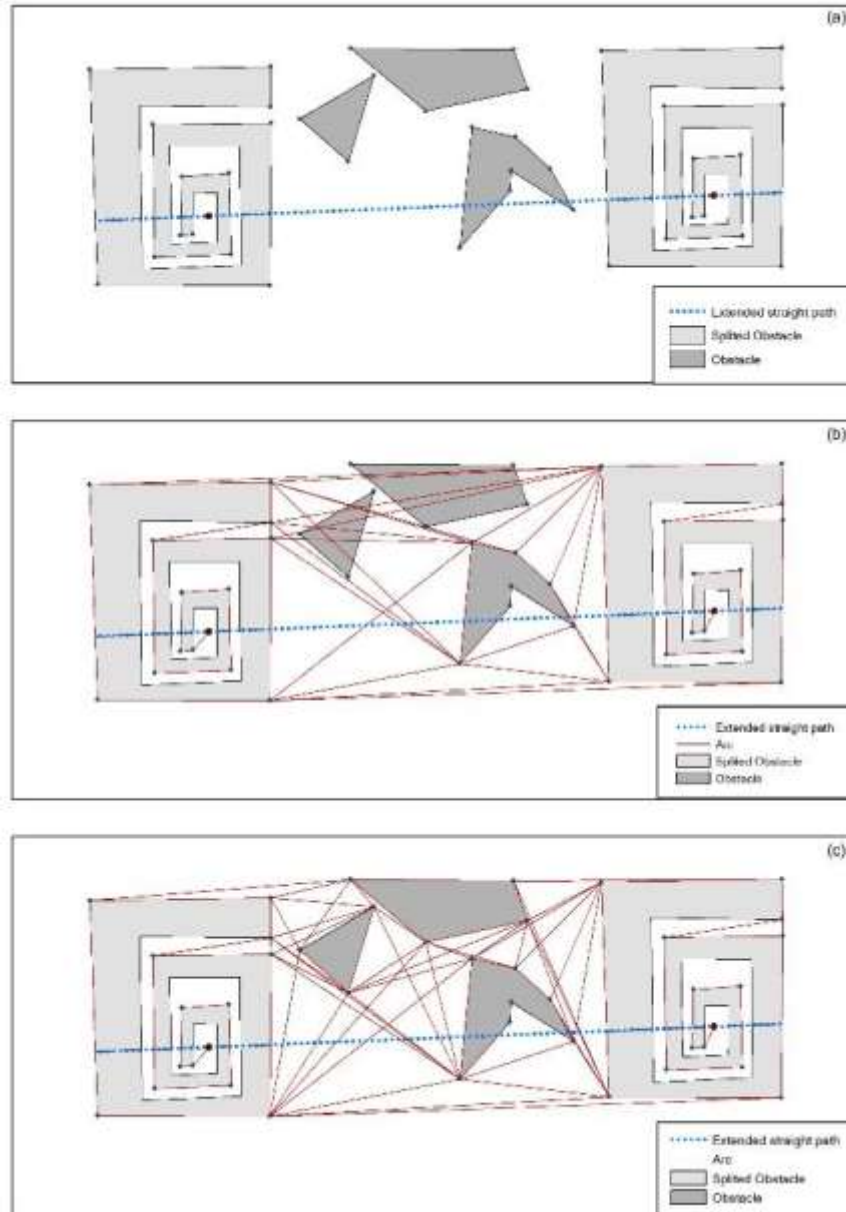


Figure 3.4. Steps of the extension of convex path

divided obstacle and either origin and destination point is generated. Boundary arcs of the convex hulls crossing the original DIOs are removed. Vertices of resulting convex hulls are linked to each other if feasible (Figure 3.4b). If any arc intersects a new obstacle, consider the obstacle an IIO, and substitute that arc with a new convex hull. Additional



arcs are created from vertices of the newly created convex hull to origin and destination points (Figure 3.4c). For boundary-induced obstacles, every step is identical, except arcs that go outside of the regional boundary are eliminated from the resulting graph. Based on theorem 1 and theorems in Hong and Murray (2013a), the resulting graph  $Ge^*$  can be considered as extended version of the graph  $G^*$  for the multiple obstacles. It therefore contains an optimal shortest path between two points.

The improved convexpath algorithm relaxes the convexity assumption by dividing violating obstacles. It finds relevant obstacles and constructs a graph  $Ge^*$  enabling the ESP problem to be efficiently solving using GIS functionalities.

### 3.6. Application

The operational efficiency of the extension of the convexpath algorithm for solving the ESP is examined for two wayfinding situations: a campus area and an interior of a building. The campus area can be considered a continuous space with obstacles (buildings). The buildings of the Arizona State University (ASU) Tempe Campus are utilized for wayfinding tasks. There are 184 buildings, but after processing to include a 3-foot buffer around each building polygon in order to avoid collisions, the number of obstacles to 179. For wayfinding in a building interior, a floor of an office building is used where a 1-foot buffer for every obstacle is applied. This results in 44 obstacles including walls, partitions and tables. The regional boundary is represented as obstacles in both cases. For each situation, three different wayfinding tasks are considered, and all 6 cases violate any convexity assumptions. Origins, destinations, obstacles and the resulting shortest paths are depicted in Figures 3.5 and 3.6. The extended convexpath

algorithm is compared with the visibility graph and the local visibility graph approaches. The three methods are implemented using C# .Net and ArcObject 10. The analysis is conducted on an Intel i5 personal computer with 2.80 GHz CPU and 12GB memory.

The number of vertices, arcs and computing time are compared. For the ASU campus area, the convexpath method required 220, 423 and 83 vertices, and 772, 1,554 and 461 arcs. In contrast, the visibility graph needed 283,991, 284,122 and 284,057 arcs and 7,653 vertices. As the visibility graph always utilizes all vertices in the region, the number of vertices for each case is identical, and the number of arcs is almost identical too. The local visibility graph contained 395, 1,837 and 833 vertices and 3,019, 22,731 and 10,423 arcs for the corresponding O/D pairs. In terms of computing time, the convexpath algorithm derived the shortest path in 74, 499 and 33 seconds for the three problem instances. The visibility graph took 1,260,001, 1,250,813, and 1,255,407 seconds, which is approximately 348.7 hours. The local visibility graph required 2,287, 69,531 and 13,468 seconds, respectively. In percentage term, the visibility graph takes about 250,810 ~ 3,810,397% and the local visibility graph 3,114 ~ 40,878% more time than the convexpath method for the individual wayfinding tasks. In the building interior cases, the overall tendency is similar. The convexpath algorithm needed 221, 243 and 223 vertices and 1,085, 1,190 and 982 arcs for the three problem instances. The visibility used all vertices in the region, total 612 including the origin and destination points for each case, and 7,241, 7,244 and 7,248 arcs. The local visibility graph for the interior cases required 494, 541 and 494 vertices and 3,518, 4,151 and 3,669 arcs, respectively. Again in percentage terms, the visibility graph is 799 ~ 1,333% less efficient than the convexpath

and the local visibility graph requires 488 ~ 809% more computing time than the convexpath algorithm.

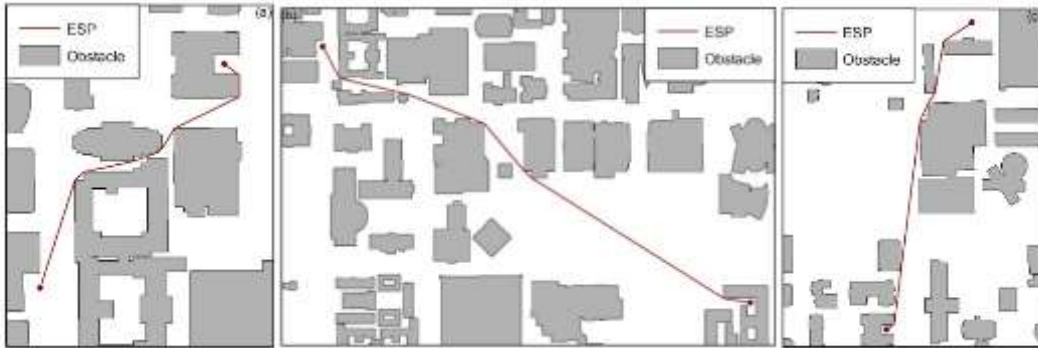


Figure 3.5. Wayfinding cases of ASU campus area



Figure 3.6. Wayfinding cases of building interior

### 3.7. Discussion and conclusions

The results demonstrate substantial advances in computing capability using extended convexpath algorithm for optimally solving the ESP problem. The significance of such improvement is a capability to solve real world problems in real time. The convexpath method exploits spatial knowledge to find the DIO and IIO efficiently, and constructs a graph for path derivation simultaneously. The number of vertices has only a slight impact on the performance of the convexpath algorithm, unlike the visibility and the local visibility graphs. The visibility graph must consider all obstacle vertices and the regional boundary. Therefore its performance is greatly affected by the number of the vertices. The local visibility graph must also evaluate all vertices of selected obstacles. Even if the local visibility graph detects less obstacles (9 obstacles in Figure 3.7a) than the convexpath algorithm (20 obstacles in Figure 3.7b), resulting computing time is much larger, as the local visibility graph considers almost 10 times more vertices. However, the convexpath algorithm only considers relevant vertices of DIOs, IIOs and BIOs. As most real world wayfinding tasks correspond to complex environment, approaches that are not highly efficient will not be practical.

While the visibility graph approach is able to solve an entire set of origin/destination pairs once. It still takes far more computing time than the total time required for the convexpath algorithm. In the case of the ASU campus, the convexpath algorithm only requires 0.07% percent of the computing time of the visibility graph for the three origin-destination pairs. In the case of the building interior environment, the convexpath algorithm needs approximately 30% computing time of the visibility graph,

due to vertices in the region. The local visibility graph also requires a large amount of computing time compared to the convexpath algorithm. For the interior environment, it requires almost 2 times more effort than the visibility graph.



Figure 3.7. Comparison between convexpath and LVG

The ESP is an essential element for various kinds of wayfinding tasks. Substantial research effort has been paid to its solutions. However, existing approaches have critical limitations. The convexpath method has significant computational advantage over existing techniques, but it is possibly limited by the convexity assumption. By strategically dividing obstacles, the extension of the convexpath method can be applied for general ESP problem solution without any limitations, and still remains as effective as the original convexpath method. The application results clearly demonstrate the performance superiority of the convexpath algorithm.

## CHAPTER 4

### SPATIAL FILTERING FOR IDENTIFYING A SHORTEST PATH AROUND OBSTACLES\*

In Chapter 2 and 3, the convexpath algorithm is developed and extended to general ESP problems. In high obstacle densities, however, the efficiency of the convexpath algorithm can be degraded due to evaluation of non-impacting obstacles to the ESP. This chapter develops a spatial filtering approach to eliminate such obstacles from evaluation and graph construction, while still preserving the optimality guarantee of the resulting path discussed in Chapter 2.

#### 4.1. Introduction

The shortest path between two points in continuous space around obstacles is fundamentally important for route planning, spatial analysis, and location modeling. It can be used for measuring proximity and distance, for deriving service areas and for studying behavioral movement (Hong and Murray 2013a, 2013b). The continuous space shortest path that avoids obstacles has been referred as the Euclidean shortest path (ESP) (Guibas and Hershberger 1989, Hershberger and Suri 1993, Mitchell 1999). It is a well recognized problem that continues to be of interest due to its practical relevance. Several solution techniques have been developed to solve the ESP, including visibility graph (Lozano-Pérez and Wesley 1979), local visibility graph (Zhang *et al.* 2005), shortest path

---

\* This chapter represents a slightly revised version of a paper submitted to *Computational Geometry: Algorithm and Applications*, co-authored with Dr. Alan T. Murray and Levi J. Wolf.

map (Mitchell 1999), and Voronoi diagram (Papadopoulou and Lee 1995) to name a few. Solution approaches for the ESP focus on generating a graph through which a path can be formed. As the ESP is a continuous space problem, there is no pre-defined network. However, research has demonstrated that a network may be derived that contains the optimal shortest path (Wangdahl *et al.* 1974, Viegas and Hansen 1985, Mitchell 1999, de Berg *et al.* 2008), allowing for the transformation of a continuous space problem to one where travel occurs on a discrete network. While an important advance, this network based transformation for the ESP has limited capability for real-time shortest path identification due to fundamental limitations in efficient graph construction. An issue is that current approaches must consider all or most obstacle and boundary vertices in a given region, regardless of the location of the origin and destination. Although substantial attention has been devoted to enhance efficiency and filter out unnecessary obstacles, real-time assessment and route planning is generally not possible with these approaches.

To address computational limitations, Hong and Murray (2013a, 2013b) proposed the convexpath algorithm to efficiently solve the ESP problem. The algorithm is based on derived spatial knowledge and geographic information system (GIS) functionality to efficiently generate a graph proven to contain the optimal ESP. The convexpath algorithm exploits spatial properties of convex hulls around an impeding obstacle, suggesting vertices and arcs to be included in a discrete graph. This approach is generally efficient, requiring minimal computing time. An exception, however, is the case when obstacle density is high, as the resulting graph tends to consider all obstacles. While still an improvement compared to alternatives like the visibility graph, near real-time response

and application may be limited when the number of obstacles is relatively high in a region.

A spatial filtering technique is developed in this chapter to achieve greater efficiency for real-time path solution across continuous space with high obstacle density. The derived graph size and computing requirements are dramatically reduced as a result of this filtering approach yet optimality conditions are maintained. This paper is structured as follows. An overview of the problem and literature is given. The convexpath algorithm and important spatial operations are described. A new solution approach based on spatial filtering is then detailed. This is followed by empirical assessment to identify an ESP using a number of routing and planning settings. Finally, concluding comments are provided.

## 4.2. Background

The shortest path between two points reflects important information about behavior and movement but it is also a measure of proximity and distance. It is heavily relied upon in spatial analysis and is central to most metrics, test and models (Bailey and Gatrell 1995, Fotheringham *et al.* 2000, O'Sullivan and Unwin 2010, Rogerson 2010, de Smith *et al.* 2012). In healthcare planning, as an example, the shortest path is an indicator of accessibility to various services (Phibbs and Luft 1995, Fone *et al.* 2006, Higgs 2009, Jones *et al.* 2010, Cudnik *et al.* 2012). The most efficient or shortest path is therefore relied upon as a representative pattern of movement when actual behavior is not known, but also may be used in a predictive or prescriptive manner. Paths of travel (and distance) must undoubtedly take into account geographic obstacles/barriers to movement lest they



be biased, inaccurate or wrong (Klamroth 2001b, Bischoff and Klamroth 2007, Carling *et al.* 2012). Research has attempted to appropriately account for obstacles in order to eliminate potential errors or biases in analysis (Katz and Cooper 1981, Larson and Sadiq 1983, Batta *et al.* 1989, Aneja and Parlar 1994, Klamroth 2001b), but challenges remain.

The Euclidean shortest path through complex environments is an essential part of navigation. (Lozano-Pérez and Wesley 1979) suggested a method to derive a collision-free shortest path using a visibility graph. Other approaches, such as pheromone signals (Szymanski *et al.* 2006), slime mold strategy (Schmickl and Crailsheim 2007) and the MAKLINK approach (Habib and Asama 1991), have also been developed to support robotic wayfinding. Several approaches have been detailed for trans-oceanic shipping, to estimate arrival times based on a shortest path avoiding various obstacles impeding a route. Approaches to address this include the visibility graph (Fagerholt *et al.* 2000), Voronoi diagrams (Bhattacharya and Gavrilova 2007), buffers and minimum bounding rectangles (Tsou 2010), and tessellation of the area using a square grid (Chang *et al.* 2003, Bekker and Schmid 2006, Zhang 2010).

Research on wayfinding in the interior of buildings as an aid to the visually impaired takes a somewhat different approach. For building interiors, most approaches assume a pre-defined (existing) network (Kwan and Lee 2005, Lee *et al.* 2010, Guven *et al.* 2012). Wayfinding for the visually impaired in outdoor environments tends to assume an existing network as well (Golledge *et al.* 1998, Loomis *et al.* 2005). Unfortunately, self-positioning technology such as GPS is not readily available inside most buildings

(Ran *et al.* 2004), and obstacle information is not generally known beyond architectural design.

Most solution methods for the ESP rely on the visibility graph. The visibility graph was proposed by Lozano-Pérez and Wesley (1979) and subsequent research has suggested alternative graph generation techniques (Asano 1985, Welzl 1985, Asano *et al.* 1986, Rohnert 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996). The visibility graph attempts to connect vertices of obstacles, regional boundary vertices, and origin and destination vertices. If two vertices are mutually visible (unobstructed), the vertices are connected by an arc and the arc (and vertices) is included in the resulting graph. It has been proven that the visibility graph includes the shortest path that avoids obstacles between any two points in the graph (Viegas and Hansen 1985, de Berg *et al.* 2008). While several improvements have been suggested in the literature (Welzl 1985, Ghosh and Mount 1991, Pocchiola and Vegter 1996), the efficiency of the visibility graph approach is limited for practical and/or real-time usage. The reason for this is that it must evaluate all or most vertex pair combinations in the region (Hong and Murray 2013a). To make the visibility graph approach more efficient, filtering techniques have been proposed. The local visibility graph, as an example, utilizes proximity-based filters to reduce the number of obstacles evaluated (Kim *et al.* 2004, Zhang *et al.* 2005, Gao *et al.* 2011, Li *et al.* 2011). Unfortunately, the local visibility graph approaches remain computationally prohibitive and may omit the optimal ESP for an origin and destination pair (Hong and Murray 2013a).

Derivation of a graph has proven to be a necessary approach to solve the ESP in support of continuous space movement. While much progress has been made to more efficiently generate a graph in various ways, real-time navigation and wayfinding remain an elusive goal for ESP application, and continues to be an important area of research. Proposed in this chapter is an important step in this direction. An approach to substantially enhance efficiency in graph derivation is detailed that enables only necessary portions of a study region to be considered based on spatial knowledge and GIS functionality.

### 4.3. Convexpath

As noted above, a common approach for solving the ESP is the visibility graph. However, it is computationally intensive, limiting its usefulness for real-time navigation and wayfinding as well as making it ineffective in big data environments. To address these limitations, Hong and Murray (2013a) proposed an algorithm, referred to as *convexpath*, to derive a graph containing the ESP. The convexpath algorithm is more efficient than the visibility graph because it intelligently exploits spatial knowledge and GIS functionality. The notion of a convex hull is utilized in graph construction, effectively eliminating unnecessary vertices and edges from being evaluated and included in the graph when they are not part of the ESP.

Consider the following definition (de Berg *et al.* 2008, Hong and Murray 2013a):

**Definition (convex hull):** Given a set  $S$  in  $R^n$ , the *convex hull* of  $S$ , denoted  $CH(S)$ , is the collection of all convex combinations between elements in  $S$ .

In two dimensions this means that the convex hull is the smallest unique polygon that contains a specific set of objects and it constitutes the minimum bounding perimeter. A property of convex hulls is that for any two points  $v_1$  and  $v_2$  that lie on the boundary,  $v_1, v_2 \in CH(S)$  and  $v_1, v_2 \cap \text{int}(CH(S)) = \emptyset$  where  $\text{int}(\cdot)$  is the function identifying the interior of a set,  $\overline{v_1 v_2} \in CH(S)$ , the line segment  $\overline{v_1 v_2}$  is completely contained in the hull and no portion of the line segment extends beyond or outside of the hull.

A property attributable to the convex hull as used to derive the ESP is the following theorem.

**Theorem 1:** Given  $v_o$  and  $v_d$ , origin and destination locations respectively, with straight line travel impeded by object  $k$ , the ESP lies on  $CH(v_o, v_d, k)$ .

**Proof:** Hong and Murray (2013a) detail a proof of this theorem. However, the minimal perimeter property discussed above is sufficient to prove the result. Given that the convex hull has the minimal perimeter enclosing the vertices, origin, and destination, paths on its boundary dominate all other paths exterior to the hull. ■

Also found in Hong and Murray (2013a) is an approach to derive the ESP for the case of multiple obstacles based on an iterative application of convex hulls to resolve any case where a hull boundary segment is impeded by obstacles in the region as well as the supporting proof of optimality.

Hong and Murray (2013b) detailed an extension to account for a situation where an origin and/or destination vertex resides in the interior of any resulting convex hull. Given  $v_o, v_d$ , and  $k$ , then this is the case of  $v_o \cap \text{int}(CH(v_o, v_d, k)) \neq \emptyset$  and/or  $v_d \cap$

$int(CH(v_o, v_d, k)) \neq \emptyset$ . Vector overlay can resolve this complication. Consider the following definition:

**Definition (line-polygon overlay):** For some polygon  $k$  and origin and destination points  $v_o, v_d$ , the *line-polygon overlay* procedure splits a polygon into two or more disjoint, boundary-sharing faces with  $\overline{v_o v_d}$  such that  $\cup_j f_j = k$ .

Line-polygon overlay is an algorithm helpful in the derivation of the ESP. Let a straight line segment  $l$  with end points  $v_o$  and  $v_d$ , the origin and destination respectively, be obstructed by polygon  $k$ . This means that  $l \cap int(k) \neq \emptyset$ . In addition, let either  $v_o$  or  $v_d$  be *contained* within  $CH(k)$ . It is assumed that  $v_o$  and/or  $v_d$  do not both lie on the resulting convex hull, so for some  $v_o, v_d$ , and  $k$ , then  $v_o \cap int(CH(v_o, v_d, k)) \neq \emptyset$  and/or  $v_d \cap int(CH(v_o, v_d, k)) \neq \emptyset$ . The line-polygon overlay operation then divides the polygon  $k$  into multiple faces  $f_j$  using line segment  $l$  such that  $\cup_j f_j = k$  with face edges formed by  $l$  (or segments of  $l$ ,  $\hat{l} \subseteq l$ ) and edges of  $k$  (or segments of edges defining  $k$ ) such that the convexpath algorithm can generate feasible paths using the faces instead of  $k$  itself. The algorithm detailed in Hong and Murray [26] to generate the ESP then constructs  $CH(v_o, f_j)$  and  $CH(v_d, f_j)$  for each face  $f_j$  (or sub-polygon), ensuring that the resulting graph includes both  $v_o$  and  $v_d$ . Any edge  $l'$  on the identified hulls that intersects the interior of the obstacle,  $l' \cap int(k) \neq \emptyset$ , is not included in the resulting graph.

**Theorem 2:** Given  $v_o$  and  $v_d$  obstructed by object  $k$  and that  $v_o$  and/or  $v_d$  do not lie on the convex hull,  $v_o \cap int(CH(v_o, v_d, k)) \neq \emptyset$  and/or  $v_d \cap int(CH(v_o, v_d, k)) \neq \emptyset$

$\emptyset$ . The line-polygon overlay operation applied to polygon  $k$  and line segment  $l$  defined as  $\overline{v_o v_d}$  enables a graph to be derived containing the ESP.

**Proof:** A full proof is provided in Hong and Murray (2013b). However, consider the case when  $v_o \in \text{int}(CH(k))$  and/or  $v_d \in \text{int}(CH(k))$ . Without loss of generality, let the line  $l$  between  $v_o$  and  $v_d$  divide  $k$  into some finite countable set of faces  $f_j$ . By Viegas and Hansen (1985) we know that some collection of edges connecting vertices between vertices on  $f_j$  will contain the ESP. Then, by Hong and Murray (2013a) we have that the shortest path around  $f_j$  must lie on convexpath derived graph around them. However, the splitting line introduces *new* vertices into these obstacles at the intersections of  $l$  and  $f_j$ . Therefore, we can use the convexpath graph around  $f_j$  induced by the  $l$  and remove all links between vertices introduced in the splitting process. As this procedure satisfies the conditions presented in Hong and Murray (2013a), it is guaranteed to produce a graph sufficient for the ESP to be both optimal and feasible for the obstacle being divided. ■

Hong and Murray (2013b) prove that the convexpath graph for any set of obstacles  $K$ , where the convex hull for some obstacles contain  $v_o$  and/or  $v_d$ , includes the ESP. The reason why this technique is needed is because no convex hull between the obstacle and the interior vertex will contain an edge connecting the interior vertex to the vertices of the polygon. To handle this, the central realization of the proof is that  $k$  can be exchanged for  $\cup f_j$ , and the convexpath graph of  $\cup f_j$  is then sufficient for ESP

optimality.<sup>1</sup> This is because  $\cup f_j$  derived using the dividing line  $l$  is exactly the original obstacle  $k$ . Feasibility and optimality conditions for this case are identical to the original problem because the convexpath graph is valid for all  $f_j$ . That is, feasibility and optimality conditions are preserved because the polygon-line overlay procedure keeps all conditions sufficient for an ESP solution intact

Using the above theorems, the convexpath algorithm derives a graph by iteratively constructing convex hulls around impeding obstacles and the origin and destination vertices. The graph expands incrementally by replacing obstructed arcs of a hull with a sub-convex hull. Hong and Murray (2013a, 2013b) proved that the resulting graph guarantees inclusion of the ESP. Empirical results have demonstrated that the convexpath algorithm is up to 60 times faster than visibility graph and local visibility graph approaches.

#### 4.4 Spatial filtering

The convexpath algorithm is highly efficient for identifying a graph and finding the ESP compared to existing alternatives, particularly the visibility and local visibility graph approaches. However, performance tends to degrade as obstacle density increases because the graph expands in a way that ultimately includes obstacles that have no impact on ESP travel. Figure 4.1 illustrates this point; the graph generated using convexpath systematically expands to include all obstacles.

---

<sup>1</sup> Post-processing is necessary to remove arcs that intersect the interior of obstacle  $k$ .

In Figure 4.1a, the graph that results from convex hulls around obstacles  $k_1$  and  $k_2$  includes an arc that intersects obstacle  $k_3$ . This requires more iterations of the convexpath algorithm to resolve infeasible arcs. Thus, the next iteration shown in Figure 4.1b eliminates the intersecting arc with obstacle  $k_3$ , replacing it with corresponding sub-convex hulls, but in doing so introduces one or more arcs in the graph that are obstructed by obstacle  $k_4$ . Figure 4.1c shows resolution of infeasible arcs in the graph with obstacle  $k_4$ , but new arcs are introduced that intersect with obstacles  $k_5$ ,  $k_6$  and  $k_7$ . These are resolved in Figure 4.1d, identifying the graph that avoids all obstacles and includes the optimal ESP. While this process is theoretically correct and fundamentally sound, resolving arcs around obstacles  $k_3$ ,  $k_4$ ,  $k_5$ ,  $k_6$  and  $k_7$  has no actual impact on the ESP for the given origin and destination in this case. This suggests potential for improving convexpath efficiency if obstacles can be eliminated from consideration under certain conditions without impacting the validity of the resulting graph and optimality properties with respect to the ESP.

**Observation 1:** A feasible path  $\hat{P}$  between  $v_o$  and  $v_d$  around obstacles  $K$  satisfies the condition  $\hat{P} \cap \text{int}(K) = \emptyset$ .

**Observation 2:** The optimal Euclidean shortest path  $P^*$  is that path for which  $|P^*| \leq |\hat{P}|$  for all other feasible paths  $\hat{P}$  where  $|\cdot|$  is the operator defining the length of the path.



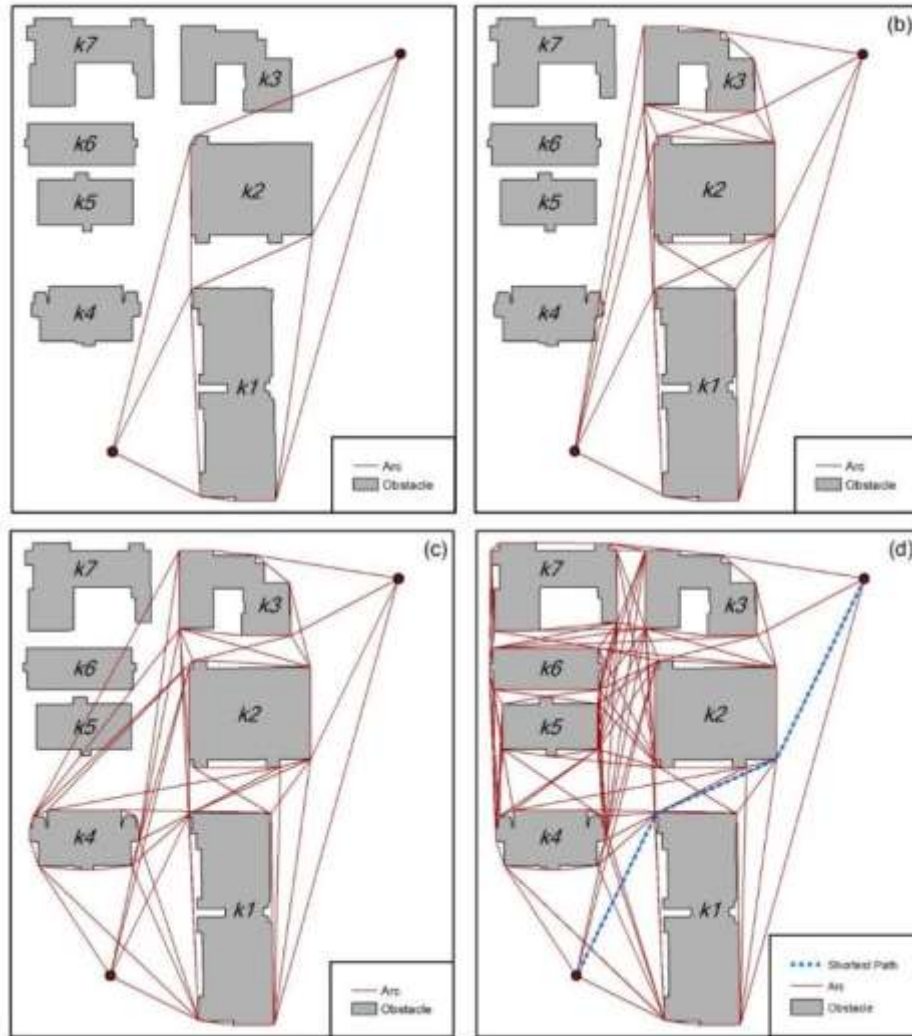


Figure 4.1. Iterative convexpath graph construction process

Thus, a path is the ESP if and only if Observations 1 and 2 are satisfied. Note that nothing in these sufficient conditions for solving the ESP suggests that *all* obstacles define/comprise the graph through which the optimal path will traverse. Rather, it is only necessary to implicitly or explicitly consider all paths and associated obstacles so long as these two conditions are satisfied at termination. The challenge therefore is developing an

approach that can filter out obstacles when they have no impact on the ESP, reflecting their implicit consideration. Interestingly, this is sort of the idea behind the local visibility graph approaches, like Zhang *et al.* (2005), Gao *et al.* (2011), and Li *et al.* (2011) who proposed proximity-based filtering methods to reduce obstacles in graph construction. While these approaches improve computational performance compared to visibility graph, they may exclude the ESP and are less efficient than convexpath (see Hong and Murray 2013a). To seek out and exploit efficiencies through spatial filtering, we detail an approach that enhances convexpath, maintaining its theoretical properties and optimality conditions. This new approach is referred to as convexpath-sf (convexpath-spatial filtering).

The convexpath algorithm generates a graph of edges and vertices,  $G$ , through which the optimal ESP can be found.  $G$  is obtained through an iterative process where an initial graph is found then expanded and/or modified strategically in subsequent iterations. Expansion and modification involves adding vertices, adding edges and removing edges. At any iteration  $t$ , graph  $G_t$  is augmented, continuing the process until no arc intersects any obstacles (as illustrated in Figure 4.1). The initial graph,  $G_0$ , consists of the origin and destination connected by the straight-line segment arc. Subsequent iterations involve replacing arcs that intersect obstacles with their corresponding sub-convex hulls. By construction,  $G_t$  is always a connected graph with one or more paths from  $v_o$  to  $v_d$ , but some arcs may be infeasible if they intersect the interior of an obstacle.

The insight gained from the final graph in Figure 4.1d is that computational effort is being devoted to graph expansion/modification for arcs and obstacles that are actually irrelevant to the optimal ESP. For example, obstacle  $k_7$  does not impact or contribute to the ESP for travel between the given origin and destination. However, convex hulls were derived and arcs added/replaced to deal with this obstacle (see Figures 4.1a, 4.1b and 4.1c). It may therefore be possible to filter out or eliminate obstacles in such cases. To develop a valid and efficient spatial filter, consider how convex hulls around obstacles help to spatially define potential shortest paths.

**Lemma 1:** Given the convexpath graph  $G$  for origin and destination points  $v_o$  and  $v_d$ , a set of identified obstacles  $\Psi$ , and the set of all obstacles  $K$  such that  $\Psi \subseteq K$ , there exists a shortest path  $P^* \in G$  that dominates all other possible paths around obstacles in the set  $\Psi$ , i.e. a path exists in  $G$  that satisfies Observations 1 and 2.

**Proof:** The proof follows from Hong and Murray (2013a). If  $G$  contains the shortest feasible path between  $v_o$  and  $v_d$ , then the path discovered must only consider obstacles within  $k \in \Psi$ . Assume that there exists some shorter path  $P'$  including obstacles  $k \notin \Psi$  that is shorter than  $P^*$ . Consider the graph  $G$  as composed of edges fully contained within  $CH(v_o, v_d, \Psi)$  and edges forming the boundary of  $CH(v_o, v_d, \Psi)$  itself. By Theorem 1, either some fully contained feasible path exists or a boundary path is the shortest feasible path. This implies no shorter path between  $v_o$  and  $v_d$  can exist outside  $G$ , so  $P'$  cannot exist and the assumption is contradicted. Thus, there must exist a path in  $G$  around obstacles in  $k \in \Psi$  that satisfies Observations 1 and 2. ■

With Lemma 1, it is not necessary to explicitly evaluate all obstacles as long as a feasible path has been identified. This establishes a spatial bound on the search space. Considering the boundary paths of  $CH(v_o, v_d, \Psi)$  and the paths contained within  $CH(v_o, v_d, \Psi)$  in the convexpath graph  $G$ , either an interior path is feasible and shortest or a boundary path is feasible and shortest. If the interior path length is large, then  $\Psi$  can be thought of as one large composite obstacle and the shortest path is on the boundary of  $CH(v_o, v_d, \Psi)$ . This means that the search space can be narrowed considerably, as either an interior path or a boundary path of  $G$  will constitute a lower bound on all possible paths.

Let us now consider the construction of a convexpath graph over all iterations. Given a graph  $G_t$  at iteration  $t$ , let  $\delta_t^*$  be the current shortest path in the convexpath graph connecting the origin to the destination that does not intersect any currently identified obstacle  $k \in \Psi_t$ . Thus,  $\delta_t^*$  is a valid shortest travel route around the identified obstacles, and is a lower bound on the optimal ESP as well. This information is helpful for graph construction in subsequent iterations and offers potential as a spatial filter because an obstructed arc can be assessed in terms of its potential to contribute to the optimal ESP. Specifically, assume for iteration  $t$  that graph  $G_t$  has a valid path  $\delta_t^*$  between  $v_o$  and  $v_d$  around the current set of obstacles  $\Psi_t$ , where  $\Psi_t \subseteq K$ . If  $G_t$  includes an edge  $l$  that intersects the interior of an obstacle  $k$  such that  $k \notin \Psi_t$  then  $l \cap int(k) \neq \emptyset$ . Accordingly, based on the convexpath algorithm, this edge must be replaced by a corresponding sub-convex hull,  $CH(v_1, v_2, k)$  where  $l$  is defined as  $\overline{v_1 v_2}$ . However, if the edge  $l$  is not part of  $\delta_t^*$ , then, by definition, any path from the origin to the destination

that includes  $l$  is longer than  $|\delta_t^*|$  and there is neither a reason to include edge  $l$  nor a need to derive its corresponding sub-convex hull. Thus, a spatial filtering approach can be devised based on this insight. This holds for any iteration  $t$ .

**Theorem 3:** Given  $G_t$  and  $G_{t+1}$  for origin  $v_o$  and destination  $v_d$ , the length of the current shortest path  $\delta_t^*$  will not decrease as  $t$  increases. That is,  $|\delta_{t+1}^*| \geq |\delta_t^*|$ .

**Proof:** For some iteration  $t$ ,  $G_t$  contains edges around identified obstacles  $\Psi_t$  such that  $\Psi_t \subseteq K$ .  $\delta_t^*$  is feasible for  $\Psi_t$ . Assume  $\delta_t^*$  is still infeasible with respect to  $k \in K$ , which means that  $\delta_t^* \cap \text{int}(k) \neq \emptyset \forall k$ . The next iteration,  $t + 1$ , infeasible edge  $l \in \delta_t^*$  (where  $l \cap \text{int}(k) \neq \emptyset$ ) is replaced with the boundary of  $CH(l, k)$  and  $k$  is added to  $\Psi_{t+1} = \Psi_t \cup k$ . Next, the algorithm constructs  $G_{t+1}$  avoiding obstacles in  $\Psi_{t+1}$ . There must then be some new shortest path  $\delta_{t+1}^*$  in  $G_{t+1}$  that dominates all other obstacle avoiding paths by Lemma 1 and Hong and Murray (2013a). For each infeasible edge in  $\delta_t^*$ , the new feasible paths in  $G_{t+1}$  cannot be shorter than their corresponding infeasible edges in  $G_t$ . This implies that the shortest path in the next iteration is some path in  $G_{t+1}$  and is at least as long as the current shortest path in  $G_t$ . That is, the length of the shortest path does not decrease as  $t$  increases because infeasible arcs are replaced by their corresponding sub-convex hulls. ■

Due to this non-decreasing property, the shortest paths provide a spatial filter to identify obstacles for evaluation that is sufficient for optimality. The non-decreasing length of shortest paths in each iteration means that shortest paths keep being impeded by at least one obstacle until they become feasible. As the shortest paths are lower bounds for each iteration, it is computationally efficient to evaluate only obstacles that impede

shortest paths compared to evaluating all obstacles that impede any arc of the graph. This *shortest path spatial filter* is utilized to select only the obstacles that impede the shortest path at each iteration. Therefore, the total number of considered obstacles can be significantly reduced.

With optimality guaranteed at each step of the process when the shortest path spatial filter is used, this provides an early termination criterion that guarantees both feasibility and optimality of the final ESP. The algorithm for deriving a graph containing the optimal ESP using spatial filtering follows.

**Algorithm:** convexpath-sf

- 1) Initialization. Given the region and set of all obstacles  $k \in K$ . Set of discovered obstacles  $\Psi = \emptyset$  and graph  $G = \emptyset$ .
- 2) Generate  $\overline{v_o v_d}$ , and add the arc and end points to  $G$ .
- 3) Derive the shortest path  $\delta^*$  from  $v_o$  to  $v_d$  in  $G$ . If arc  $l \cap \text{int}(k) \neq \emptyset$  for any  $l \in \delta^*$ ,  $k \in K$ , then add  $k$  to  $\Psi$ . If nothing is added, terminate process. Optimal ESP found.
- 4) For all  $l \in G$  and  $k \in \Psi$  such that  $l \cap \text{int}(k) \neq \emptyset$ , replace  $l$  in  $G$  with  $CH(l, k)$ . If any end point of  $l$  ( $l = \overline{v_1 v_2}$ ) not on hull boundary, then split  $k$  into faces  $f_j$  using line-polygon overlay and generate hulls for each face as detailed previously.
- 5) Go to step 3.

While implied above, a formal proof of convexpath-sf optimality is now given.

**Theorem 4:** The convexpath-sf algorithm terminates at an optimum feasible shortest path.

**Proof:** Let the shortest path at the last iteration be  $\delta_t^*$  in graph  $G_t$  identified by convexpath-sf. As discussed above,  $\delta_t^*$  is both feasible and the shortest path in graph  $G_t$  at iteration  $t$ . Then, assume that  $G_t$  does not contain the optimal ESP, meaning that some feasible path  $\delta'$  exists such that  $\delta' \notin G_t$ ,  $|\delta'| \leq |\delta_t^*|$ , and  $\delta' \cap \text{int}(K) = \emptyset$  by Observations 1 and 2. In this case, if  $|\delta_t^*| = |\delta'|$ , then  $\delta_t^*$  is equivalent in length to one of many possible optimal paths. By Theorem 3, this must be a lower bound on feasible path lengths, so convexpath-sf must have terminated at one of many potential ESPs and the assumption that  $\delta_t^*$  is not an ESP is untenable. If  $|\delta'| < |\delta_t^*|$ , then  $\delta'$  dominates  $\delta_t^*$ . By Lemma 1,  $\delta'$  must dominate all other feasible paths that exist in  $G_t$  for  $\Psi_t$ . However,  $\delta_t^*$  dominates all other feasible paths that exist in  $G_t$  for  $\Psi_t \subseteq K$ . Thus,  $\delta'$  cannot exist because, for any set of obstacles,  $\delta_t^*$  is the lower bound on all potential ESP lengths and must be shortest. ■

Put differently, the convexpath-sf algorithm finds a path that is both feasible and optimal by satisfying Observation 2 in each iteration and terminates as soon as Observation 1 is satisfied. However, it should be noted that  $\delta_t^*$  does not satisfy Observation 2 for *all* obstacles in  $K$ , but instead satisfies it for all paths around  $\Psi_t$ , the obstacles *identified* up to that iteration. Each iteration's shortest path,  $\delta_t^*$ , provides a lower bound on the ESP, meaning Observation 2 is satisfied for  $\Psi_t$ . But, Observation 1 is not satisfied until the last iteration, when  $\delta_t^*$  becomes feasible for all obstacles  $k \in K$ . Because Observation 2 is always satisfied for path candidate  $\delta_t^*$  over obstacles  $\Psi_t$  and

$|\delta_t^*|$  is non-decreasing according to Theorem 3, convexpath-sf satisfies both Observation 1 and Observation 2 for all obstacles  $k \in K$  at termination.

Convexpath-sf can be contrasted with convexpath, illustrated in Figure 4.1. As shown in Figure 4.2, the graph constructed by convexpath-sf is significantly smaller than the unfiltered convexpath graph (Figure 4.1d). While the convexpath algorithm produces a graph with 124 edges, the convexpath-sf algorithm derives a graph with 18 edges, substantially smaller in size. As will be detailed below, this enhanced efficiency translates directly into faster computational processing. In this case, graph construction using convexpath-sf is over seven time faster. This substantial improvement is possible because the shortest path spatial filter explicitly considers only two obstacles, whereas convexpath evaluates all seven obstacles in ESP derivation.

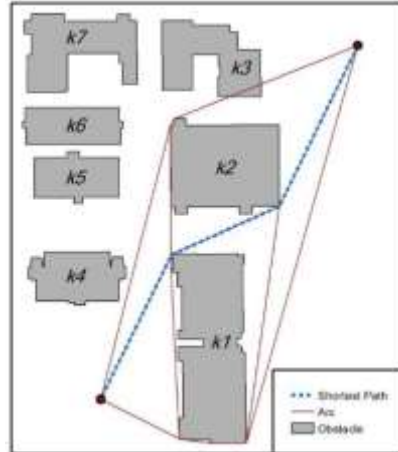


Figure 4.2. Resulting graph and ESP using the convexpath-sf algorithm

As suggested, the efficiency of the convexpath algorithm is significantly improved by the spatial filtering method developed in this chapter. By reducing the



number of obstacles evaluated, the resulting graph and required computing time are notably smaller than convexpath. Furthermore, the shortest path spatial filter does not require significant additional computing time, as it uses existing spatial information in intermediate graphs.

#### 4.5. Application results

To evaluate the efficiency of the convexpath-sf algorithm to support wayfinding, three different planning contexts are evaluated. To assess impacts of increased obstacles on performance, each planning context has a different number of obstacles. For the low obstacle density context, travel across Tampa, Florida considering six obstacles is utilized. The study area was also used in Hong and Murray (2013a). The medium-density case represents travel through a building interior, and was examined in Hong and Murray (2013b). This has 50 obstacles, including walls and tables. For the high density obstacle context, a portion of the Arizona State University campus is considered. There are 178 buildings that impede travel. The three planning problems are shown in Figure 4.3, with an illustrative origin-destination pair given. To avoid any potential collision with an obstacle, a small buffer is applied to each obstacle, one foot for the medium and three feet for high density contexts. A total of 30 origin-destination pairs are considered for each travel context, with all cases encountering at least one impeding obstacle.

The convexpath and convexpath-sf algorithms were implemented using Python. Derivation of the ESP for the origin-destination pairs in the various planning contexts on an Intel i5 personal computer (2.80 GHz) with 12 GB of RAM. In all cases the optimal ESP is found.

Comparative results for the two algorithms are summarized in Table 4.1, and average relative performance difference in percentage term is shown in Figure 4.4. The average number of obstacles, average number of arcs generated and average computing time is reported in Table 4.1 for the 30 different origin-destination paths to be found in the different obstacle density contexts. Table 4.1 indicates that convexpath evaluates 2.23 obstacles, on average, in the low-density case in deriving the graph needed to find the ESP. This increases to 38.03 and 74.67 in the medium and high cases for convexpath. In contrast, convexpath-sf explicitly evaluates only 1.47, 4.80 and 9.63 obstacles, on average, for the three different contexts. The significance of this, of course, is that a smaller graph results requiring less computing time. In the high density case, as an example, 5104.13 arcs on average are included in derived graphs requiring some 208.88 seconds using convexpath, whereas only 250.37 arcs on average are included in the graph requiring only 1.99 seconds using the spatial filtering algorithm (convexpath-sf). Comparative differences are shown in percentage terms in Figure 4.4 indicating that convexpath-sf explicitly assesses only 13.12% of the obstacles, generating 5.14% of the arcs needing 1% of the computing time. The performance trend is quite obvious in that as the number of obstacles increases, larger graphs and more computing time are required of the convexpath algorithm. The spatial filtering approach, however, is able to mitigate graph size growth and associated computational effort. This reduction is greatest in the higher density context, as suggested in Figure 4.4. Convexpath-sf requires 53.42% of the computing time in the low-density context, 10.25% in the medium-density context, and only 1% in the high density context.



Figure 4.3. Travel path planning contexts, each showing one origin-destination pair:

(a) Low obstacle density; (b) Medium obstacle density; (c) High obstacle density

Table 4.1. Average performance comparison of 30 origin-destination pairs

using the convexpath and convexpath-sf algorithms

Density	Convexpath			Convexpath-sf		
	Obstacles	Arcs	Time (sec)	Obstacles	Arcs	Time (sec)
Low	2.23	11.93	0.02	1.47	7.63	0.01
Medium	38.03	2366.27	27.60	4.80	153.53	0.57
High	74.67	5104.13	208.88	9.63	250.37	1.99

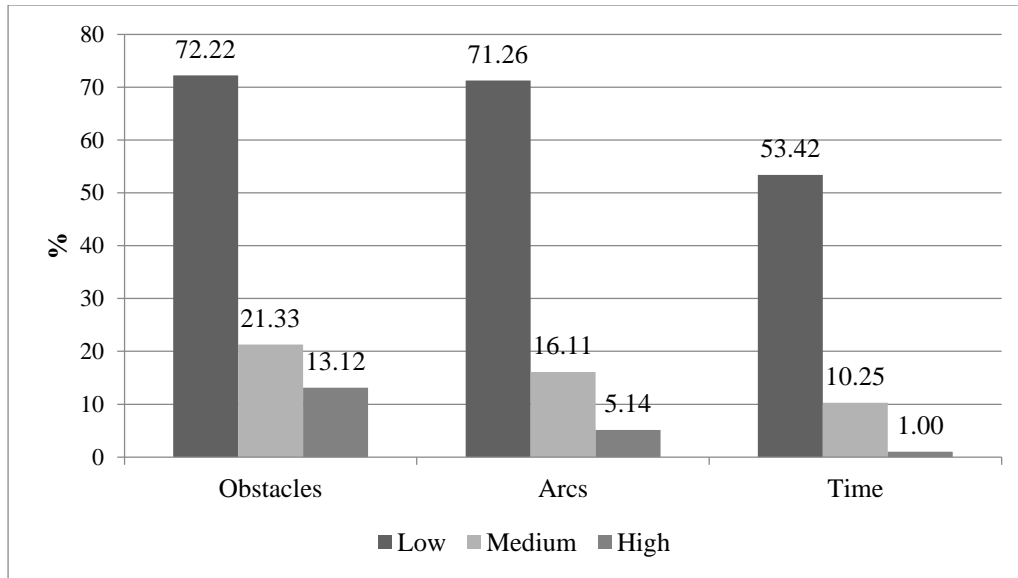


Figure 4.4. Average relative performance comparison of 30 origin-destination pairs in the different obstacle density contexts using the convexpath and convexpath-sf algorithms

#### 4.6. Discussion and conclusions

The application results show that the convexpath-sf algorithm enhances capabilities for deriving the ESP in real time, especially in cases where a high density of obstacles is encountered. Table 4.1 and Figure 4.4 clearly show that the effectiveness of the filtering technique increases with increasing obstacle density. Most certainly, the convexpath-sf algorithm can derive an ESP in real time, even in high obstacle density cases. For the high density obstacle context, convexpath-sf derives the ESP (graph and solution) in less than 2 seconds for 20 of the 30 cases. For the medium obstacle density context, 21 of the origin-destination pairs took less than 0.6 seconds using the convexpath-sf algorithm.

Another noteworthy point is that `convexpath-sf` is less influenced by oddly-shaped obstacles. For example, the medium density obstacle set shown in Figure 4.3b has several long and curved obstacles. This can lead to additional computing due to the inclusion of more obstacles that must be explicitly evaluated. However, `convexpath-sf` is insensitive to obstacle shape.

Another unique aspect of the `convexpath-sf` algorithm is its use of an identified shortest path in the resulting graph, regardless of feasibility, as a bound for the ESP. One might wonder why this approach focuses on deriving infeasible paths, which is unlike many other approaches. For example, operations research approaches commonly focus on bounding the problem by resolving the gap between a current feasible solution and the best possible infeasible solution. Branch and bound is such an approach, restricting the size of the problem search space by implicit consideration of unlikely branches. Through well-ordered evaluation, entire branches of a problem search tree can sometimes be pruned. At optimality, the upper and lower bound must converge. In contrast, the `convexpath-sf` algorithm is a different tact, focusing on the most promising path rather than worrying about finding an initial feasible solution.

The Euclidean shortest path is essential for spatial analysis and wayfinding. Numerous solution techniques have been developed for the ESP, but are not capable of supporting real-time path derivation. Although the `convexpath` algorithm is a major advance, a high number of obstacles can limit its performance. To address this, a spatial filtering method was developed to improve computational processing. By utilizing shortest paths as a bounding filter, the `convexpath-sf` algorithm derives the ESP very

efficiently. Empirical results clearly support capabilities for real-time derivation of the ESP even when there is a high density of obstacles.

## CHAPTER 5

### HIGH PERFORMANCE COMPUTING TO DERIVE AN OBSTACLE-AVOIDING SHORTEST PATH\*

In the previous chapters, the convexpath algorithm is developed and enhanced to provide solution for general ESP problem with improved efficiency. Still, the performance of convexpath can be limited in big data environment. This chapter develops a parallelized version of the convexpath algorithm to address performance and scale issues. Essential computationally intensive spatial operators are restructured to allow for parallelization.

#### 5.1. Introduction

The shortest path in continuous space around obstacles provides essential information for spatial analysis, location modeling and wayfinding tasks. A simple case involving one obstacle is shown in Figure 1 where one is seeking the shortest possible path that avoids the obstacle. This path has been referred to as the Euclidean shortest path (ESP), and has attracted the attention of many researchers. Techniques such as visibility graph (Lozano-Pérez and Wesley 1979), local visibility graph (Zhang *et al.* 2005), shortest path map (Mitchell 1999) and Voronoi diagrams (Papadopoulou and Lee 1998) have been applied to solve the ESP problem to support robotics, shipping, location modeling, and more (Lozano-Pérez and Wesley 1979, Fagerholt *et al.* 2000, Klamroth

---

\* This chapter represents a slightly revised version of paper submitted to *International Journal of Geographic Information Science*, co-authored with Dr. Alan T. Murray and Dr. Sergio J. Rey.

2001a). Methods to date for deriving the ESP utilize a graph in order to reduce combinatorial search space from infinite routing options to a finite, discretized set of line segments (Hong *et al.* 2014). However, the efficiency of existing methods is considerably limited by the need to evaluate most or all obstacle and region boundary vertices within a given area (Hong and Murray 2013a).

To address these limitations, new methods for deriving the ESP more efficiently have been developed by Hong and Murray (2013a, 2013b) and Hong *et al.* (2014). These approaches exploit spatial knowledge and geographic information system (GIS) functionality to efficiently generate a graph that guarantees inclusion of the ESP. By utilizing the notion of a convex hull, together with intersection, vector overlay and spatial filtering, the approaches of Hong and Murray (2013a, 2013b) and Hong *et al.* (2014) explicitly consider only relevant obstacles impeding a given origin-destination pair, thereby enabling efficient construction of a smaller-sized graph. In big data environments, however, even such highly efficient methods can require considerable computing resources. This is particularly true for solving the ESP as computationally intensive spatial operators need to be repeatedly utilized. As a result, large data applications supporting navigation and wayfinding remain a challenge to solve. Advanced computing techniques offer potential to enhance the performance of new algorithms for deriving an ESP, possibly overcoming scale limitations encountered in practice.



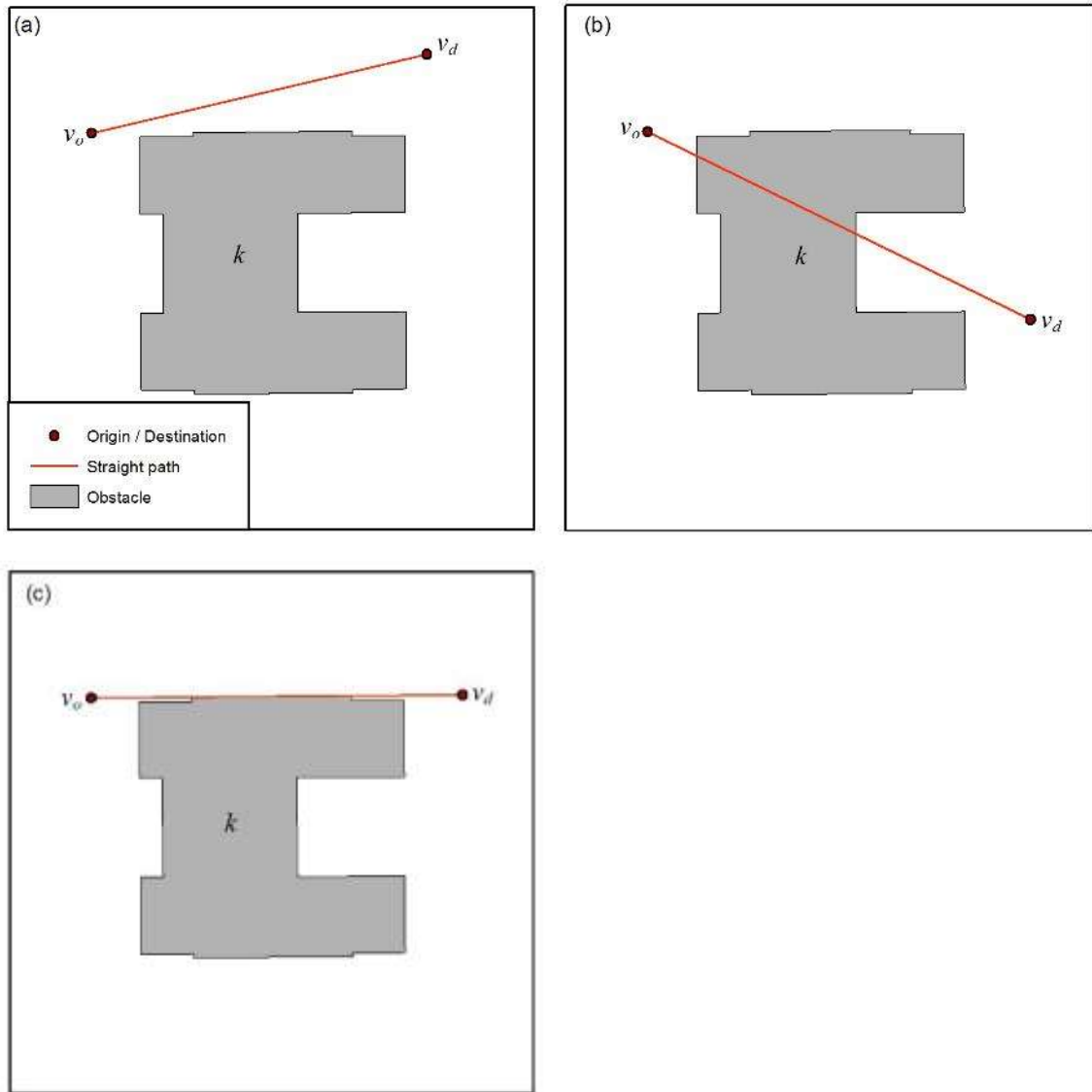


Figure 5.1. Three cases of intersection: (a) no intersection

(b) interior intersection (c) boundary intersection

Parallelization techniques have proven useful for addressing various performance and scale issues in spatial analysis (Lanthier *et al.* 2003, Zhang 2010, Anselin and Rey 2012, Rey *et al.* 2013). Computing architecture for parallelization has evolved from being

a feature of supercomputers and grid computation to relying on multicore CPUs, General Purpose Graphics Processing Unit (GPGPU) and virtual computing resources (Zhang 2010, Xia *et al.* 2011, Anselin and Rey 2012). These methods of parallelization exploit the computing power of multiple CPU/GPU cores in either single or multiple machines to boost performance.

In this chapter, a parallel computing technique is developed to enhance the performance of the convexpath algorithm of Hong *et al.* (2014) to solve the ESP. The parallel algorithm proves to significantly improve performance when compared to sequential implementation. The next section details the ESP and summarizes algorithms for solving it. A new parallelized solution approach is then introduced. Application results are presented, followed by discussion and concluding comments.

## 5.2. Background

The ability to derive a shortest path is fundamental for answering questions about movements, proximity and distance, and it is also essential for spatial analysis (see Bailey and Gatrell 1995, Fotheringham *et al.* 2000, O'Sullivan and Unwin 2010, Rogerson 2010, de Smith *et al.* 2012). Doing this accurately means that obstacles/barriers must be taken into account. Lozano-Pérez and Wesley (1979) presented a solution approach to derive a collision-free shortest path, and other research efforts have followed for robot path planning based on the ESP (Lozano-Perez 1987, Habib and Asama 1991, Szymanski *et al.* 2006, Schmickl and Crailsheim 2007). Similarly for trans-oceanic shipping, the ESP is utilized when considering obstacles like reefs, shoals, rocks, islands and other restrictions

that impede vessel travel (Fagerholt *et al.* 2000, Chang *et al.* 2003, Bekker and Schmid 2006, Bhattacharya and Gavrilova 2007, Zhang *et al.* 2011).

While attempts have been made to address obstacles in spatial analysis relying on a shortest path (Katz and Cooper 1981, Larson and Sadiq 1983, Batta *et al.* 1989, Aneja and Parlar 1994, Klamroth 2001a), major practical limitations exist. As a result, efficient problem solution remains a research challenge. Many of the widely-applied solution approaches depend on the visibility graph method. The visibility graph was suggested by Lozano-Pérez and Wesley (1979) and alternative graph generation techniques have been developed (Asano 1985, Welzl 1985, Asano *et al.* 1986, Rohnert 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996). Although the visibility graph is guaranteed to contain the optimal ESP, practical application is limited. The reason for this is that all vertices in a given area, including the origin, destination and points defining obstacle boundaries, must be considered and included in the resulting graph. Research has focused on reducing the size of the derived graph, such as local visibility graph approaches using spatial filtering techniques (Kim *et al.* 2004, Zhang *et al.* 2005, Gao *et al.* 2011, Li *et al.* 2011). However, local visibility graphs remain computationally intensive to identify. Further, they may produce a graph that omits the optimal shortest path (Hong and Murray 2013a).

Urban environments with many obstacles make derivation of an ESP a computationally intensive task, even when using a highly efficient method. Spatial analysis requiring substantial computation must contend with big data, requiring the use of parallelization approaches (Lanthier *et al.* 2003, Zhang 2010, Anselin and Rey 2012, Rey *et al.* 2013). Computing paradigms for parallelization have shifted from focusing on

supercomputers and grid computing to using multicore CPU and many-core GPGPU architectures on single or multiple machines (Zhang 2010, Xia *et al.* 2011, Rey *et al.* 2013). Such parallelization boosts performance by undertaking many tasks concurrently. Strategies for parallelization can be categorized as task and data oriented (Barry 2006, Gong *et al.* 2013). Task parallelization decomposes processes into individual functions that are then conducted both independently *and* simultaneously. In contrast, data parallelization uses identical functions but separates data into multiple computing threads.

Multicore CPU parallelization techniques utilize the power of multiple cores in a single CPU (Rey *et al.* 2013). Multicore CPU architecture has emerged to overcome several fundamental limitations of single core CPUs, utilizing a small number of coarsely-grained threads with shared memory (Zhang 2010, Gong *et al.* 2013). Multicore CPU parallelization has been used for spatial analysis, including agent-based simulation (Gong *et al.* 2013), LiDAR point cloud processing (Guan and Wu 2010), and thematic map classification (Rey *et al.* 2013).

GPGPU architecture exploits a GPU's numerical processing capabilities and applies them to general purpose problems, even though GPU architecture is intended to support 2D/3D visualization (Zhang 2010). Nvidia's CUDA (Compute Unified Device Architecture) and AMD's OpenCL (Open Computing Language) are the most widely used GPGPU frameworks. Of the two parallelization strategies, GPGPU is more suited to data parallelization (Xia *et al.* 2011, Rey *et al.* 2013). Spatial analysis applications such as interpolation and watershed analysis (Xia *et al.* 2011), large scale spatial regression

(Zhang 2010) and thematic map classification (Rey *et al.* 2013) all have benefited from GPGPU parallelization in terms of performance and scalability.

Parallelization techniques not only boost performance and help overcome scale problems (Guan and Wu 2010), but they also enable more realistic representation of real world processes compared to sequential processing (Openshaw and Turton 1999). However, complete reconstruction of existing algorithms is often required to achieve significant parallelization efficiencies.

### 5.3. Shortest path derivation

An efficient method for ESP derivation was recently proposed in Hong and Murray (2013a, 2013b) and Hong *et al.* (2014). The approach, referred to as the convexpath algorithm, utilizes spatial knowledge and GIS functionality to efficiently construct a graph through which the optimal path can be found. To reduce the size of the resulting graph, the convexpath algorithm explicitly accounts for only relevant obstacles for a given origin-destination pair by utilizing geometric properties of convex hulls along with spatial filtering. Convex hulls are constructed to simultaneously identify impeding obstacles and construct a graph through which the path will be routed. Spatial filtering is used to reduce the number of obstacles considered during graph construction. Similar to visibility graph approaches, it has been proven that the resulting graph of the convexpath algorithm is guaranteed to include the optimal ESP (Hong and Murray 2013a). However, the convexpath algorithm produces a graph that is significantly smaller in size compared to visibility graph (or other approaches) and requires substantially less computational effort to derive (Hong and Murray 2013a, 2013b, Hong *et al.* 2014).

The convexpath algorithm relies on four important spatial operators: interior intersection, convex hull, line-polygon overlay, and spatial filtering (Hong *et al.* 2014). Each operator is now defined and discussed.

**Definition (interior intersection):** Given a polygon  $k$ , the *interior* of  $k$ , denoted  $int(k)$ , is the open set bounded by the polygon edges but disjoint from the edges.

The interior intersection operator is utilized to evaluate whether a polygon obstructs straight line travel between two points. Assume a pair of origin and destination points,  $v_o$  and  $v_d$ , and an obstacle  $k$  are given, as shown in Figure 1. At issue is whether the line segment formed by the origin and destination intersects the obstacle, or formally if  $\overline{v_o v_d} \cap k \neq \emptyset$ . There are three potential cases of intersection between the straight line and the obstacle. The first case is depicted in Figure 1a, where  $\overline{v_o v_d} \cap k = \emptyset$ . That is, straight line travel is not impeded. The second case is shown in Figure 1b, where  $\overline{v_o v_d} \cap k \neq \emptyset$ . In this case straight line travel is not possible. The third case is shown in Figure 1c, where  $\overline{v_o v_d} \cap k \neq \emptyset$ , but intersection occurs on the boundary of the obstacle. Technically, straight line travel is permissible along this boundary as one could be close but not exactly on the boundary. For this reason, the interior intersection operator is necessary to account for the third situation. Thus, interior intersection evaluates whether the interior of the obstacle intersects a line segment, allowing movement along an obstacle boundary. As a result, the case shown in Figure 1c results in  $\overline{v_o v_d} \cap int(k) = \emptyset$ . Interior intersection therefore requires additional spatial assessment to separate boundary intersection from polygon interior intersection. The implication is that the

interior intersection operator requires additional computing time compared to the more generic intersection operator.

Another important spatial operator is convex hull. The following definition is offered in Hong *et al.* (2014):

**Definition (convex hull):** Given a set of objects  $S$ , the *convex hull* of  $S$ , denoted  $CH(S)$ , is the collection of all convex combinations, or the smallest convex set containing objects in  $S$ .

The convex hull is a key operator in the convexpath algorithm. It is utilized to construct the shortest detour avoiding an obstacle for a given origin-destination pair. A convex hull is illustrated in Figure 2 for three spatial objects,  $v_o$ ,  $v_d$  and  $k$ . Based on the minimum bounding property of a convex hull, Hong and Murray (2013a) proved that the ESP is on  $CH(v_o, v_d, k)$  when a single obstacle  $k$  impedes the straight travel between  $v_o$  and  $v_d$ . This necessarily means that  $\overline{v_o v_d} \cap int(k) \neq \emptyset$ . For multiple obstacles, relevant obstacles are identified by testing interior intersection with the convex hull. If an obstacle obstructs an edge on a convex hull, the edge is replaced by the sub-convex hull of an obstructed edge and corresponding obstacle. This involves adding vertices of the convex hull as well edges. A graph that contains the ESP is constructed by iteratively resolving obstructed arcs in the graph through the generation of sub-convex hulls. The convexpath algorithm therefore relies on the convex hull operator repeatedly in an application, potentially thousands or more times.

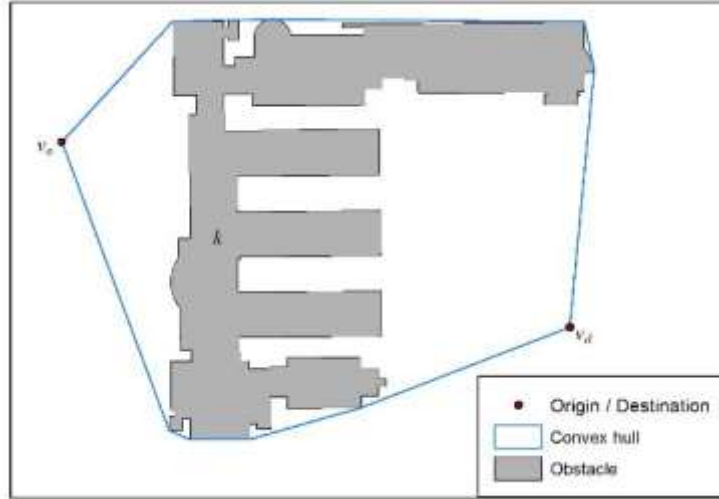


Figure 5.2. Convex hull for origin, destination and impeding obstacle

A third spatial operator utilized in deriving an efficient graph that contains the ESP is line-polygon overlay. This is a special case of the more general vector GIS overlay. Consider the following definition:

**Definition (line-polygon overlay):** Given the line defined by two points,  $\overline{v_1v_2}$ , that intersects polygon  $k$  ( $\overline{v_1v_2} \cap \text{int}(k) \neq \emptyset$ ), *line-polygon overlay* splits  $k$  into multiple disjoint faces  $f_j$  defined by the boundary of  $k$  and segments of  $\overline{v_1v_2}$ , such that  $\cup_j f_j = k$ .

The line-polygon overlay operator is used in the creation of feasible graph arcs. Suppose we have origin and destination vertices  $v_o$  and  $v_d$ , respectively, where  $\overline{v_o v_d} \cap \text{int}(k) \neq \emptyset$ . This situation is illustrated in Figure 5.3b, for the case shown in Figure 3a. The line segment is now used to divide obstacle  $k$  into sub-obstacles, or faces, as depicted in Figure 5.3c. The reason that this overlay is necessary is because  $\text{int}(CH(v_o, v_d, k)) \cap v_d \neq \emptyset$ , as shown in Figure 5.4a because  $v_d$  is not on the boundary



of  $CH(v_o, v_d, k)$ . This leads to a situation where  $v_d$  is not connected to the derived graph, making a path from  $v_o$  to  $v_d$  impossible. To address this, Hong and Murray (2013b) detailed an algorithm to make use of line-polygon overlay to construct a feasible and valid graph based upon subdividing polygon  $k$  and removing infeasible potential arcs. The resulting graph made possible because of line-polygon overlay is shown in Figure 5.4b. The obstacle is divided into two faces by line-polygon overlay operator using  $\overline{v_o v_d}$  (Figure 5.3c). The resulting graph using faces and post-processing is given in Figure 5.4b, making a path possible between  $v_o$  and  $v_d$  (and it is guaranteed to contain a shortest path).

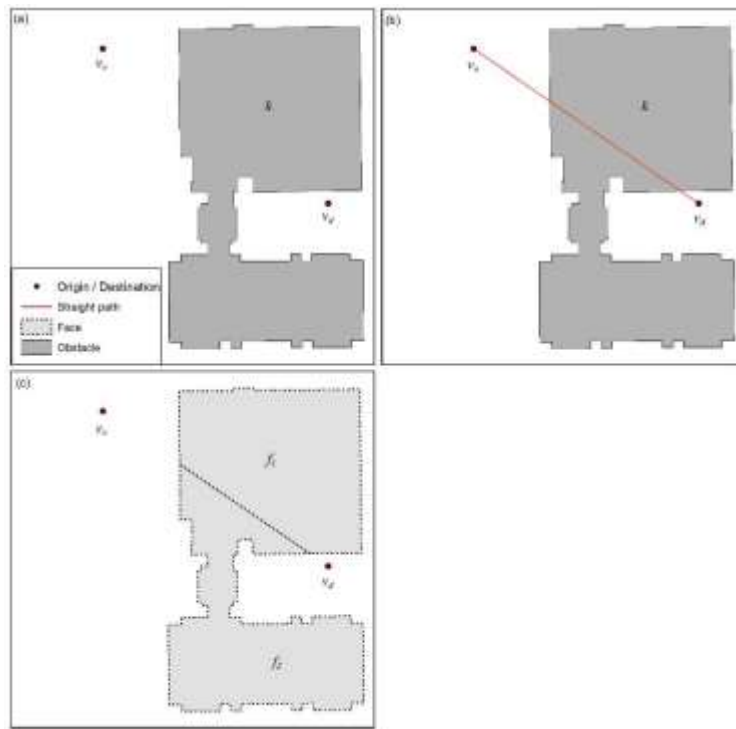


Figure 5.3. Line-polygon overlay: (a) Origin, destination and impeding obstacle  
 (b) Interior intersection test of straight line and obstacle  
 (c) Faces obtained by line-polygon overlay operator

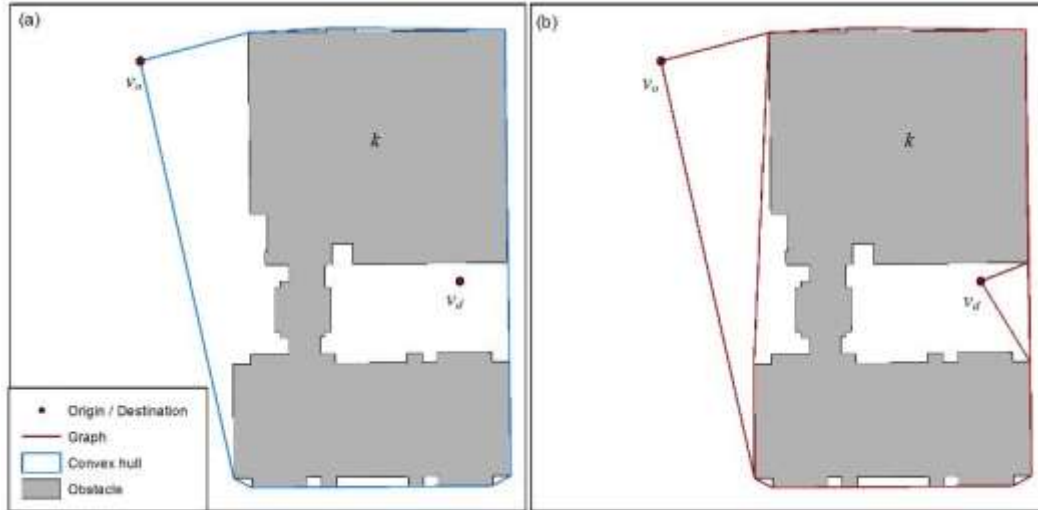


Figure 5.4. Graph construction using line-polygon overlay

(a) enclosed destination in convex hull

(b) derived graph that include ESP between origin and destination

The final operator to be noted is spatial filtering.

**Definition (spatial filtering):** A process for implicitly considering obstacles in graph construction, *spatial filtering* excludes those obstacles  $k \in K$  that are deemed not capable of impacting the ESP based on spatial proximity criteria.

The spatial filtering operator is used to reduce the number of obstacles whose vertices are explicitly incorporated in the graph through which a shortest path is derived. A number of approaches have been considered in the literature (Zhang *et al.* 2005, Gao *et al.* 2011, Li *et al.* 2011), often relying on a bounding rectangle or radius from an origin/destination. The spatial filtering approach utilized here is based on the work of Hong *et al.* (2014) who derived a process for identifying and evaluating potential graph

arcs and obstacles. Their process relies on a spatial filter based on shortest paths found at intermediate stages of the solution process. Hong *et al.* (2014) proved that among obstacles where  $int(k) \cap G \neq \emptyset$  for  $k \in K$ , only obstacles that impede the lower bound of the ESP, intermediate shortest paths, are necessary to evaluate at that stage of the algorithm. The reason is that these arcs constitute a graph containing the shortest possible path, but a given path could be infeasible if an arc intersects a yet to be considered obstacle. Therefore, a shortest path can be utilized as spatial filter to identify potential impeding obstacles. The spatial filter operator is illustrated in Figure 5.5. Assume that two obstacles,  $k_1$  and  $k_2$ , are already considered impeding obstacles for a given origin-destination pair, shown in Figure 5.5a. The graph for these two obstacles and the origin and destination are also shown, derived using interior intersection and convex hull operators. Given the graph, a shortest path can be found (Figure 5.5a). However, an obstacle,  $k_3$ , impedes an arc on this shortest path. What is known is that this path is a lower bound on the optimal ESP, so resolving arc obstructions on this path offers the promise for a feasible path (Figure 5.5b). In contrast, obstacle  $k_4$  does intersect an arc in the graph, but it is not on the shortest path. Because of this, at this stage, it does not appear promising. At some further iteration of the algorithm it may become promising and if so will then get explicitly evaluated. This technique enables valid optimality bounds to be established, and with this it is possible to preclude some obstacles if there is no chance that they would influence the optimal ESP. More discussion of this point will be provided after the convexpath algorithm is introduced.

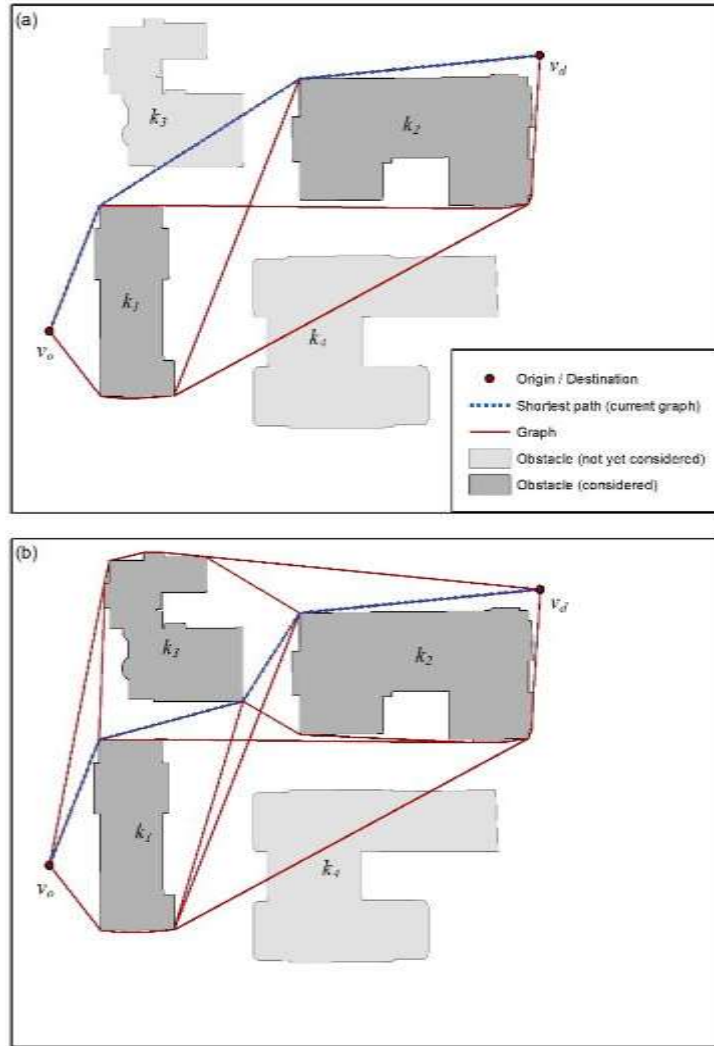


Figure 5.5. Spatial filter operator (a) Identifying potential impeding obstacle;  
 (b) Derived graph and ESP

The above spatial operators are utilized in an algorithm to derive a graph (and path) containing the ESP based Hong *et al.* (2014).

**Algorithm (Convexpath):**

- 1) Given an origin and destination,  $v_o$  and  $v_d$  respectively, and set of all obstacles  $k \in K$ . Initialize graph  $G = \{v_o, v_d, \overline{v_o v_d}\}$ , set of identified obstacles,  $\Psi = \emptyset$ , and set of impeded arcs and obstacle,  $\Theta = \emptyset$ .
- 2) Derive the shortest path between the origin and destination,  $\delta$ , in  $G$ . If  $\delta \cap \text{int}(k) \neq \emptyset$  for  $k \in K$ , then  $\Psi = \Psi + k$ . Otherwise,  $\delta$  is the optimal ESP.
- 3) If  $l \cap \text{int}(k) \neq \emptyset$  for  $l \in G$  and  $k \in \Psi$ , then add  $(l, k)$  to  $\Theta$ . If  $\Theta = \emptyset$ , then go to step 2.
- 4) For all  $(l, k) \in \Theta$ , replace  $l$  with  $CH(l, k)$ .  $G = G + l'$  for all  $l' \in CH(l, k)$ . If  $v_1 \cap \text{int}(CH(l, k)) \neq \emptyset$  and/or  $v_2 \cap \text{int}(CH(l, k)) \neq \emptyset$  when  $l = \overline{v_1 v_2}$ , then utilize line-polygon overlay to split  $k$  into  $f_j$  and construct convex hulls for each  $f_j$ .
- 5) Go to step 3.

The convexpath algorithm constructs a graph through which the optimal ESP can be found. Construction begins with the origin and destination as nodes of the graph and then proceeds to add arcs. The shortest path that links origin and destination is derived from the graph. Any obstacle that obstructs any arc in the shortest path spatial filter is added to the identified obstacle set for graph construction. Arcs obstructed by identified obstacles are resolved by iterating over the set and deriving convex hulls to replace obstructed arcs. This process continues until no arc in the graph is obstructed by identified obstacles. The convexpath algorithm stops if the spatial filter does not find any impeding obstacle.

The convexpath algorithm produces a valid lower bound, the intermediate shortest path, for the optimal ESP at every iteration and utilizes it as a spatial filter. As proved in Hong *et al.* (2014), using this shortest path spatial filter for obstacle identification prevents evaluating obstructed arcs and obstacles that will never impact the optimal ESP. The significance of this is that many obstacles need not be explicitly evaluated while optimality is still guaranteed. As a result, the convexpath algorithm is able to derive the optimal ESP with notably smaller-sized graphs and, more importantly, needs less computing time.

The convexpath algorithm has proven to be a superior approach for solving the ESP compared to approaches like the visibility and local visibility graph (Hong and Murray 2013a, 2013b, Hong *et al.* 2014). Empirical findings suggest results can be computed over 10,000 times faster while the number of arcs in derived graphs is less than 0.03% of alternative approaches (i.e., visibility graph). In complex environments characteristic of an urban area with a large number of obstacles, however, even a highly efficient method can require considerable computing time. Derivation of the ESP in this case requires a large-sized graph with repeated application of computationally intensive spatial operators. Advanced computing techniques can address the challenges of complex environments in order to support large-scale wayfinding and navigation.

#### 5.4. Algorithm parallelization

As mentioned in the previous section, the convexpath algorithm relies on four important spatial operators: interior intersection, convex hull, line-polygon overlay and spatial filtering. Advanced computing techniques show promise for overcoming scale

overhead due to repetitive utilization of certain operations. In particular, parallelization of spatial operators offers potential for algorithm improvement, especially in the context of navigation and wayfinding. In order for parallelization to be successful, steps and operations in the convexpath algorithm need to be analyzed in terms of computing requirements and supporting data structures. This may suggest useful restructuring strategies.

The computing resource consumption of each major step in the convexpath algorithm was analyzed for a number of planning applications detailed later in the paper. In a sequential environment, most processing is associated with convex hull construction, requiring roughly 58% of total computing time. The convex hull operator is repeatedly applied a significant number of times over subsequent iterations for impeded arcs and their corresponding obstacles. Such repetitive utilization of this computationally intensive spatial function makes it the most significant computational bottleneck in the convexpath algorithm. Interior intersection also consumes considerable resources, requiring about 27% of the total computing time. The interior intersection operator is applied for all arc segments in  $G$  and identified obstacles in  $\Psi$  at every iteration, and there can be several thousand arcs in  $G$ . Collectively, 85% of computing time is devoted to two spatial operators (convex hull and interior intersection). A prime target for parallelization is therefore those functions in the convexpath algorithm where there is repeated utilization of these operators. Based on this analysis, the theoretical expectation for maximum performance gain can be calculated using Amdahl's law (Amdahl 1967):

$$S(p) = \frac{1}{(1 - \alpha) + \frac{\alpha}{p}}$$

where  $\alpha$  is the parallelized portion of the process and  $p$  is the number of processors utilized for parallelization. By parallelizing convex hull and interior intersection operators,  $\alpha$  would be 0.85, and the total computing time will be a function of the number of cores in a CPU available for use.

While aspects of the convexpath algorithm are suitable for parallelization, significant performance increases are not likely to result from naive implementation of sequential steps in some simultaneous fashion. Parallel processing requires meaningful restructuring strategies. The major reasons for reconstruction are concurrent data modification, job distribution and overhead cost of parallelization. Concurrent editing (writing) of shared memory object is not allowed in parallel processing in order to prevent data corruption. Unlike the sequential case, results from multiple parallel processes cannot be stored in single data object. Rather, they need to be maintained separately. Successful parallelization will require the algorithm and data structure to efficiently recollect results after each iteration. Similarly, a strategy for efficiently distributing jobs to each subprocess also needs to be considered. Of course, initialization of subprocesses, distributing jobs and collecting results require additional computational effort. This represents computing overhead in this case. If overhead is significant, or the benefits of parallelization are minuscule, the effectiveness of parallel computing is degraded and may result in longer computing time than a sequential implementation.



Therefore, reducing overhead as much as possible, while maximizing efficiency gain, is essential.

Parallelization of the convexpath algorithm is done here using Python's official Multiprocessing library. The Multiprocessing library uses multiple cores in a single CPU, and generates subprocesses rather than threads to negotiate the limitations of the Global Interpreter Lock (GIL) (Rey *et al.* 2013). Jobs are divided up and then distributed to subprocesses. Once completed, subprocesses return the result. As mentioned earlier, to prevent data corruption, results from subprocesses need to be collected separately after parallel processing is completed.

Based on the analysis of complexity, processing requirements during application and algorithm insights for convexpath, a parallel algorithm is proposed based upon reorganization and restructuring relative to major computing processes.

**Algorithm (Convexpath-parallel):**

- 1) Given an origin and destination,  $v_o$  and  $v_d$  respectively, and set of all obstacles  $k \in K$ . Initialize graph  $G = \{v_o, v_d, \overline{v_o v_d}\}$  and set of identified obstacles,  $\Psi = \emptyset$ .
- 2) Derive the shortest path between the origin and destination,  $\delta$ , in  $G$ . If  $\delta \cap int(k) \neq \emptyset$  for  $k \in K$ , then  $\Psi = \Psi + k$ . Otherwise,  $\delta$  is the optimal ESP.
- 3) Initialize set of  $p$  queues for parallel processing,  $Q$ . Divide  $G$  into  $q \in Q$ .
- 4) Parallel processing: repeat following step in each subprocess
  - a. If  $l \cap int(k) \neq \emptyset$  for  $l \in q$  and  $k \in \Psi$ , then replace  $l$  with  $CH(l, k)$ .  $q = q$

+  $l'$  for all  $l' \in CH(l, k)$ .

- b. If  $v_1 \cap \text{int}(CH(l, k)) \neq \emptyset$  and/or  $v_2 \cap \text{int}(CH(l, k)) \neq \emptyset$  when  $l = \overline{v_1 v_2}$ , then utilize line-polygon overlay to split  $k$  into  $f_j$  and construct convex hulls for each  $f_j$ .

5) Collect results and generate new  $G$ ,  $G = \cup_p q_p$

6) If  $G \cap \text{int}(\Psi) \neq \emptyset$ , go to step 3. Otherwise, go to step 2.

The steps that utilize the two most computationally burdensome spatial operators in `convexpath` are restructured to take advantage of parallel processing. The primary components of parallelization for `convexpath` are illustrated in Figure 6. The inputs are the intermediate graph  $G$  and the explicitly evaluated obstacles  $\Psi$ . Recall that the set  $\Psi$  is a byproduct of the spatial filtering operator. Multiple arcs are processed simultaneously, which leads to faster derivation of a new graph. First, arcs in  $G$  are divided and distributed to queues for concurrent processing. The number of queues and subprocesses are determined by the number of cores in a CPU available for use, denoted  $p$ , and as mentioned above the number of subprocesses is a crucial factor that determines performance improvement of parallelization. The evaluated obstacle set  $\Psi$  is shared in memory to improve processing efficiency. In order to avoid unnecessary increases in overhead steps involving convex hull and interior intersection operators are integrated into a single parallel process. Each subprocess performs interior intersection tests for a given arc  $l \in q$  and  $k \in \Psi$ , then derives a convex hull,  $CH(l, k)$ , and replace  $l$  with  $CH(l, k)$  in cases where  $l \cap \text{int}(k) \neq \emptyset$ . Once the subprocesses are completed, an

updated graph  $G$  is generated using the collected results from parallel process. Data structures are modified accordingly for parallel efficiency. This can be contrasted to what occurs during this stage of the algorithm in the sequential implementation of convexpath. As the sequential case only makes use of one processor, the queue consists of all graph arcs that must be evaluated individually. In Figure 5.6 this would mean only one subprocess, or  $p=1$ .

This approach significantly reduces burdens associated with bottleneck operators, leading to notable decreases in total computing time for ESP derivation. Parallelization of the convexpath algorithm also facilitates analysis in complex urban environments. This efficiency gain is possible because of reconfiguration and data structure efficiency when applying spatial operators.

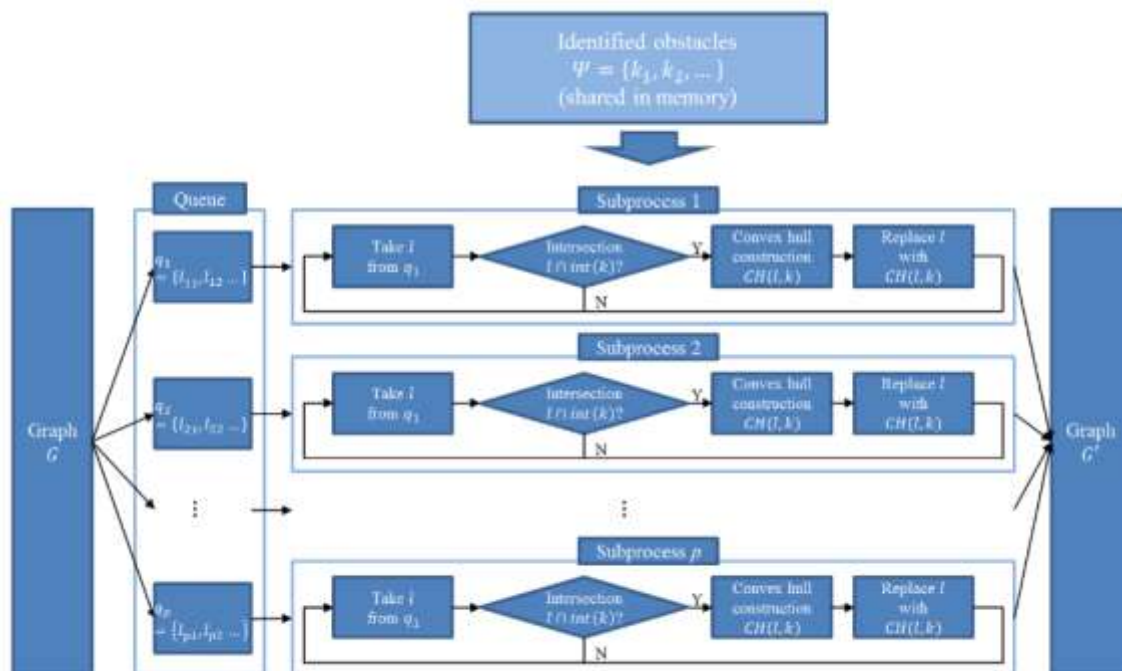


Figure 5.6. Parallel processing steps of the convexpath-parallel algorithm

## 5.5. Computational results

The effectiveness of `convexpath-parallel` is assessed for wayfinding in a high obstacle density urban environment. The initial area of focus is the Arizona State University campus in Tempe, Arizona. The campus has 179 buildings that represent obstacles to direct travel between an origin and destination (Figure 5.7). To avoid pedestrian contact with a building during travel, a 3-foot buffer is assumed for each building. In order to assess implications for computational processing, four additional applications are considered. These applications contain 500, 600, 700 and 800 obstacles, respectively, enabling evaluation as obstacle density increases. A total of 2,238 origin-destination pairs are used to derive the optimal ESP across the five application instances. Every origin-destination pair has at least one or more impeding obstacles obstructing a direct (i.e. straight) path. The computational evaluation is carried out on an Intel i7 CPU (4 physical cores) with 8 GB memory running a Mac OSX 10.7.5 operating system. Both the `convexpath` (sequential) and the `convexpath-parallel` algorithms are implemented in Python 2.7.

One origin-destination pair is shown in Figure 5.8, along with the derived graph. The optimal ESP is also included. The number of arcs in the graph is 723, and there were 19 obstacles (out of 179) that were necessary to explicitly consider in its derivation. The `convexpath` algorithm derives this in 6.58 seconds, while the `convexpath-parallel` algorithm requires only 4.13 seconds. This is nearly a 40% reduction in computing time.



Figure 5.7. Arizona State University campus buildings (buffered)

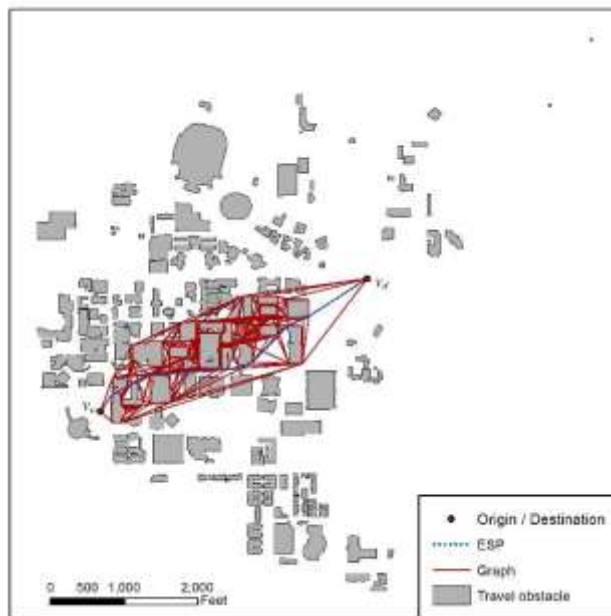


Figure 5.8. Generated graph using convexpath along with optimal ESP  
(ASU campus application)

A summary of the different origin-destination cases for convexpath and convexpath-parallel is given in Table 5.1. For the ASU campus, 239 different origin-destination pairs were evaluated. On average, convexpath-parallel is 44% faster in deriving the ESP. Other applications were considered as well, increasing in the number of obstacles in order to assess computational performance as obstacle density increases. These results are also summarized in Table 5.1. For the 500 obstacle application, 630 origin-destination pairs were evaluated. Average graph size increases and, on average, convexpath-parallel requires around 50% less computing time. Even better performance is seen for the 600, 700 and 800 obstacle applications, where computing time is 53.3%, 53.6% and 52.2% less for convexpath-parallel.

Table 5.1. Application summary

Application (number of obstacles)	Number of OD pairs	Number of arcs (avg)	Computing time (sec)	
			Convexpath	Convexpath-parallel
179	239	472.67	6.63	3.70
500	630	711.13	16.21	8.02
600	637	1038.07	32.52	15.18
700	635	1478.87	68.48	31.74
800	97	2179.82	151.44	72.45

While Table 5.1 reports average findings over the different applications, performance for each of the 2,238 ESP instances is summarized in Figure 5.9 and Table 5.2. To evaluate effectiveness of parallelization as a function of ESP problem size, the number of arcs in the resulting graph and sequential computing time are shown in Figure 5.9 to reflect the magnitude of obstacles encountered in each problem instance. In general, the convexpath-parallel algorithm reduces computing time by as much as 63.7%

compared to the sequential implementation of convexpath, making it some 2.75 times faster than sequential computation. The performance gain is higher for the cases that require identification of more arcs, therefore needing more computing time. Figure 5.9 shows this general tendency of performance improvement, where the ratio of convexpath/convexpath-parallel computing time is plotted against the number of arcs in graph. Table 5.2 provides statistics of three computing time ranges, 0 to 1 second, 1 to 10 seconds, and greater than 10 seconds for solution using the convexpath algorithm, and it shows improvement of efficiency gain over sequential computing time clearly. The optimal ESP is found within 1 second in 592 problem instances. The overhead costs are significant for 189 cases for this range, meaning that parallel implementation of convexpath takes more time to derive the ESP than sequential implementation. Excluding these cases, parallel computation requires at best 57.2% less computing time, and convexpath-parallel uses 20.9% less computing time on average. Convexpath takes 0.42 seconds to derive the ESP, and in case of convexpath-parallel it takes an average of 0.32 seconds.

Performance gains for parallel computing are significant in the 694 OD pairs that take 1 to 10 seconds using convexpath. Although the variance of computing time reduction in Table 5.2 appears similar to the first time range, results are more clustered around lower computing times, and the convexpath-parallel algorithm requires 40.1% less time than the convexpath algorithm on average. A mean of 4.37 seconds and 2.5 seconds are required for derivation of the ESP for convexpath and convexpath-parallel, respectively. As shown in Table 5.2, the 952 ESP instances that require more than 10 seconds to be solved benefit substantially from parallel processing. Computing time

reductions converge to between 36% and 64% and convexpath-parallel requires 51.2% less time than convexpath on average. It typically takes 42.95 seconds to derive results using convexpath-parallel, while convexpath uses 91.87 seconds on average to solve identical problems.

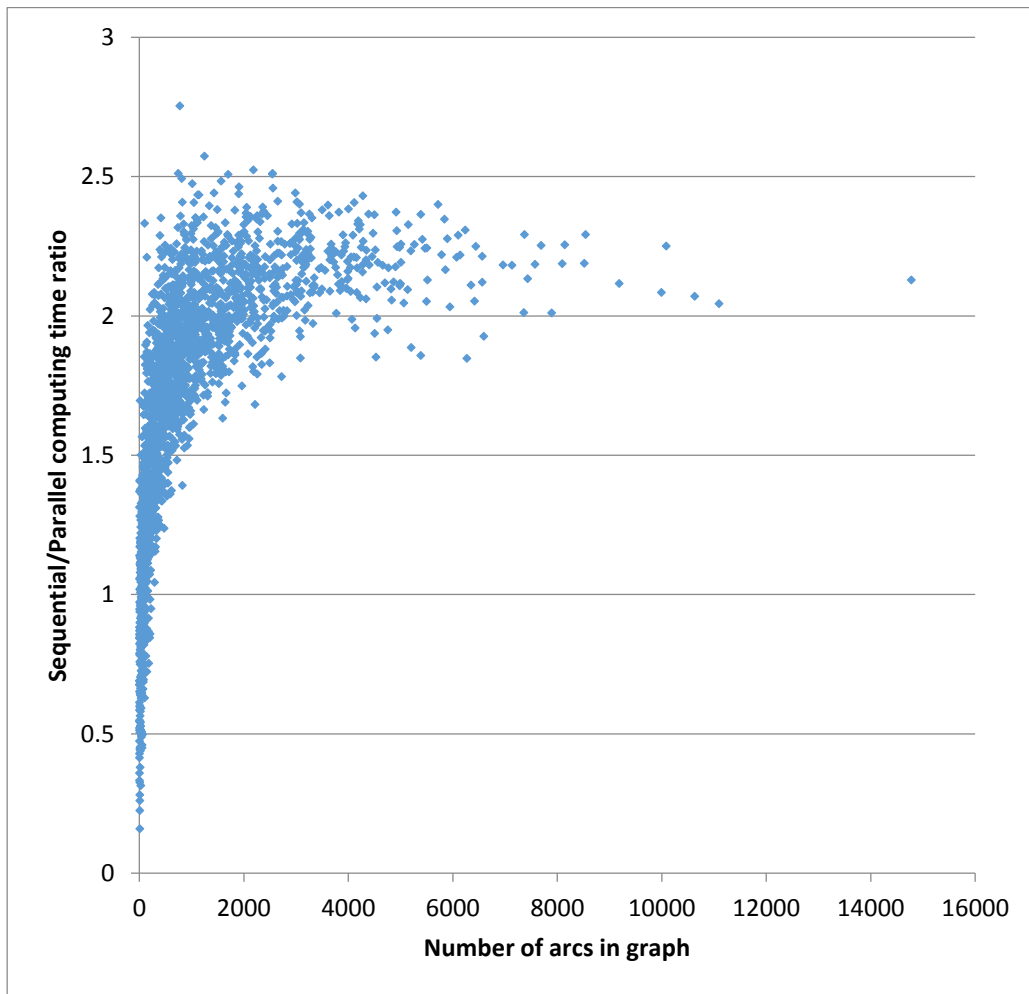


Figure 5.9. Scatterplot of sequential/parallel computing time ratio against the number of arcs in graph



Table 5.2. Summary of performance improvement

	Convexpath (sec)		
	$0 < t < 1$	$1 < t < 10$	$10 < t$
<b>Number of OD pairs</b>	592	694	952
<b>Number of cases that exceed overhead</b>	189	3	0
<b>Min reduction in computing time (%)</b>	0	4.3	35.8
<b>Max reduction in computing time (%)</b>	57.2	57.5	63.7
<b>Average reduction in computing time (%)</b>	20.9	40.1	51.2
<b>Average convexpath (sec)</b>	0.42	4.37	91.87
<b>Average convexpath-parallel (sec)</b>	0.32	2.5	42.95

## 5.6. Conclusions and discussion

Parallel processing significantly improves sequential computing for the convexpath algorithm, especially for ESP problems that need to address a large number of obstructions, thereby requiring longer computing time. Figure 5.9 and Table 5.2 show the effectiveness of parallelization as problem size increases. For example, parallelization reduces computing time more than 50%, or more than 2 times faster, in 62.2% of problem instances that require more than 10 seconds to solve, while only 5.2% of OD pairs that take less than 10 seconds show identical improvement. This tendency in efficiency gains is more clearly revealed in origin-destination pairs that need more than 100 seconds to derive the ESP, as 92.7% of these 220 cases can be solved less than 50% of convexpath computing time. When efficiency gain is viewed in terms of the number of arcs in a derived ESP graph, a similar tendency is revealed. The number of problem instances requiring more than 2,000 arcs to identify the optimal ESP is 352, and 85.5% of these cases take less 50% of convexpath computing time. Only 19% of the other 1,886 cases produce less than 2,000 arcs, but have a similar level of performance improvement.

Therefore, parallel computing can significantly improve performance for wayfinding and navigation tasks in complex urban area with large number of obstacles.

The findings suggest that excessive overhead associated with convexpath-parallel is minimal. The fact that almost all problems with unacceptably high overhead are ones obstructed by 5 obstacles or fewer indicates a simple fact about the application of convexpath-parallel: these problems can be solved very rapidly using sequential convexpath, 0.17 seconds on average, so overhead from parallelization consumes a relatively large portion of computing time  $f$ . However, the additional computing time required by convexpath-parallel over sequential convexpath in these cases is quite small, only 0.04 seconds on average. Therefore, the issue of excessive overhead in very small-sized ESP problems can be ignored and convexpath-parallel can be effectively applied to ESP problems of any size.

Another noteworthy point is that the performance gain from parallelization is strongly influenced by hardware configuration, namely the number of CPU cores in use. Amdahl's law suggests a theoretical expected maximum speed-up ratio for convexpath-parallel of 2.76, as  $\alpha = 0.85$  and  $p = 4$ . This means convexpath-parallel would compute the ESP 2.76 times faster than convexpath in the best case. The application results summarized in Table 5.1 show the average ratio varies from 1.79 to 2.15, and the achieved maximum ratio is 2.75 (see Figure 5.9). Considering the influence of overhead cost for parallelization, the results are consistent with the theoretical expectation. Furthermore, the maximum and average computing time reduction will increase somewhat if the application is run on a system with many CPU cores. Although an

evaluation of general scalability is beyond the scope of this paper, an interesting future research question is to test the broader impacts of computing system configuration and hardware.

Efficiently deriving the Euclidean shortest path is essential for spatial analysis, location modeling and path planning tasks. Several existing solution approaches have been developed, but lack capabilities for rapid real-time path derivation. The convexpath algorithm is proposed as a means to increasing the efficiency of ESP calculation, promising great potential for the development of practical real time path computation. In complex environments, however, a large number of obstacles continues to pose a significant computational burden for the convexpath algorithm when the computation is done sequentially, because of the use of computationally intensive spatial operators. In this research, parallelization is examined using a multi-core CPU in a single machine applied to the convexpath algorithm. Spatial operators involving tests of complex spatial relationships and resolving obstructions, currently the major bottlenecks in the derivation of the ESP in a sequential environment, benefit substantially from parallelization. The application results show that the parallelization approach is effective and scalable for wayfinding problems with a large number of densely-packed obstacles.

## CHAPTER 6

### ASSESSING RASTER GIS APPROXIMATION FOR EUCLIDEAN SHORTEST PATH ROUTING\*

In the previous chapters, the obstacle-avoiding shortest path is derived based on a vector representation of the spatial problem using the convexpath algorithm. In contrast, raster representations have been used to approximate the ESP using the least cost path approach. Although raster-based approaches have benefits, they have issues regarding to computational efficiency and the quality of estimated path is not guaranteed. This chapter assesses the issues of the ESP approximation in raster representation.

#### 6.1. Introduction

Planning routes to avoid obstacles in 2-dimensional space with no pre-defined network for movement is an essential operation for infrastructure planning, corridor alignment, robotic travel, simulation, and video gaming, among others (Lombard and Church 1993, Yap 2002, Ferguson and Stentz 2006, Daniel *et al.* 2010, Yap *et al.* 2011). Criteria for the desired path depends on the given problem context, but there are two prevailing solution approaches. One is that an efficient path is sought that avoids impenetrable obstacles, where efficiency is distance based. The second approach is that a path is sought that minimizes travel costs over/through a surface. Interestingly, these two approaches reflect vector and raster GIS views, respectively, as suggested in Figure 6.1.

---

\* This chapter represents a slightly revised version of paper submitted to *Transactions in GIS*, co-authored with Dr. Alan T. Murray

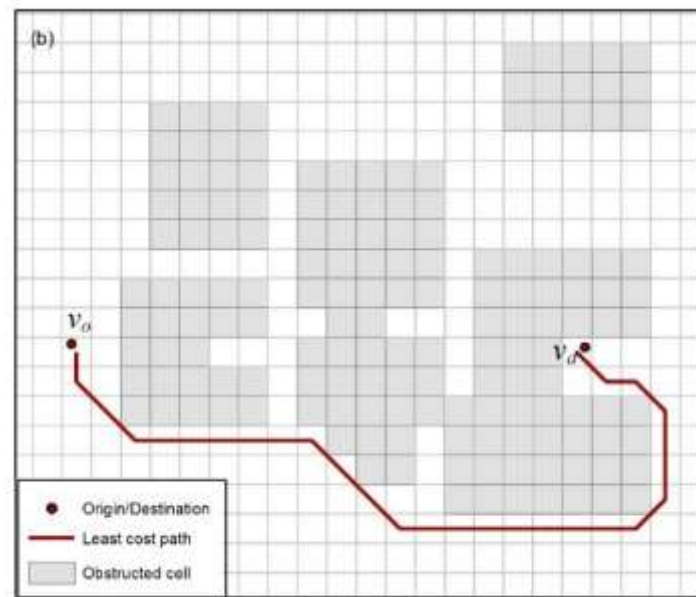
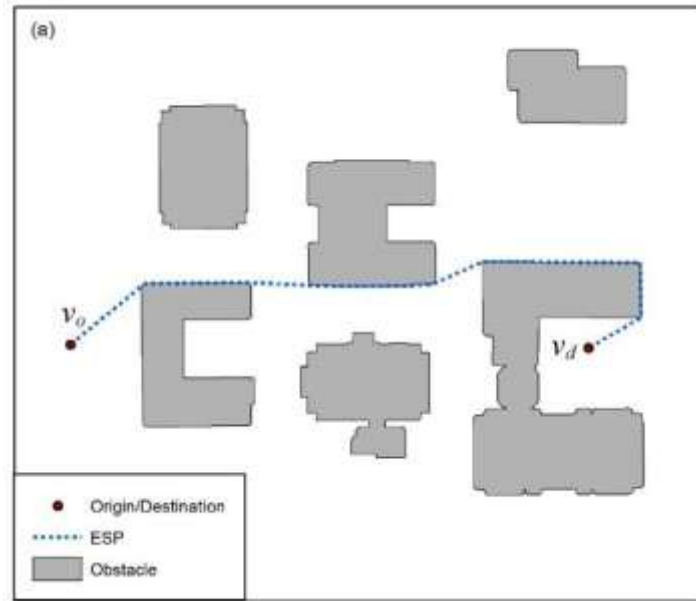


Figure 6.1. Shortest path derivation: (a) vector-based approach; (b) raster-based approach

Deriving a shortest path from a given origin to a given destination that avoids obstacles has attracted the attention of researchers in a range of disciplinary fields,

including computer science, computational geometry, planning, geography, and others (Lozano-Pérez and Wesley 1979, Batta *et al.* 1989, Zhang *et al.* 2005). There are many different naming conventions used to refer to this problem, but the most popular is the Euclidean shortest path (ESP) (Welzl 1985, Guibas and Hershberger 1989, Hershberger and Suri 1993). The ESP is typically viewed in terms of vector GIS, where vector objects (lines or polygons) reflect the obstacles to be avoided (Figure 6.1a). The goal then is to identify/construct a polyline based path of minimum length that does not intersect any obstacle.

Finding a minimum cost path from a given origin to a given destination through a surface has also received substantial attention across most disciplines, including civil engineering and robotics (Lombard and Church 1993, Yu *et al.* 2003, Ferguson and Stentz 2006, Yap *et al.* 2011). The least cost path is usually viewed in raster GIS terms, where the region is represented by a regular grid surface with each cell in the grid storing a travel cost based attribute (Figure 6.1b). The goal in this case is to find a least cost path beginning at the origin that connects neighbor cells until the destination is reached, ideally avoiding cells that correspond to obstacles.

While these two approaches, vector (ESP) and raster (least cost path), are related, they are fundamentally different in many ways. Most importantly, the ESP is not permitted to go through an obstacle that is to be avoided, whereas a least cost path may include any cell in a region. Nevertheless, the least cost path has been relied upon for solving ESP based problems, especially in robotics and video gaming (Mitchell 1988, Daniel *et al.* 2010, Yap *et al.* 2011). The major reason for this is the simplicity of

problem conception, but also the ease to attach high costs to obstacle cells as a way to discourage travel/routing through them (Jalbert and Dobson 1976). In contrast, the ESP is more mathematically complicated to express and requires specialized techniques for its solution.

This paper focuses on the ESP and issues that may arise through least cost path approximation. A review of shortest path derivation methods in vector and raster environments is given. Computational and representation issues are then detailed. Comparative results follow, highlighting differences between the two basic approaches. The paper ends with concluding comments and discussion.

## 6.2. Background

As mentioned in the previous section, continuous space path planning can be approached two ways: the ESP in vector GIS and as a least cost path in raster GIS. The ESP in a vector GIS environment takes into account an origin, a destination and obstacles that are stored as vector objects, namely points, lines and polygons. Movement can occur anywhere in space except through the interior of lines/polygons designated as obstacles. The objective is to minimize the length of a polyline based path that links the origin and destination while avoiding obstacles. The ESP has been utilized for robot path planning (Lozano-Pérez and Wesley 1979, Szymanski *et al.* 2006), trans-oceanic shipping (Fagerholt *et al.* 2000, Bekker and Schmid 2006), and location optimization (Larson and Sadiq 1983, Klamroth 2001b)

An effective solution approach for the ESP involves the construction of a graph to limit the search space, reducing the problem from an infinite number of route possibilities to a finite number of nodes and line segments (a graph). Most popular has been the visibility graph approach suggested by Lozano-Pérez and Wesley (1979), with subsequent development to improve efficiency in graph derivation (see Asano 1985, Welzl 1985, Rohnert 1986, Ghosh and Mount 1991, Pocchiola and Vegter 1996). It has been proven that the visibility graph contains the optimal ESP (Viegas and Hansen 1985, de Berg *et al.* 2008), and is constructed by connecting mutually visible vertices of obstacles and the origin and destination. Given the graph, a shortest path algorithm may be employed to identify the optimal route. An issue with the visibility graph approach is computational processing because all vertices must be evaluated in graph construction, regardless of proximal relevance to the origin and destination. Local visibility extensions have been developed to address efficiency issues (Kim *et al.* 2004, Zhang *et al.* 2005), often employing proximity-based spatial filtering techniques to reduce the number vertices evaluated. However, local visibility graphs also require considerable computing time and may omit the optimal path (Hong and Murray 2013a).

In a raster GIS environment, the least cost path has been utilized to identify a route through a continuous space from an origin to a destination (Goodchild 1977, Huber and Church 1985, Mitchell 1988, Lombard and Church 1993, Collischonn and Pilar 2000, Etherington and Holland 2013). Unique to raster GIS is the representation of space as a continuous surface, with a location corresponding to a cell that has a unique cost associated with traversing the cell. Thus, movement occurs as travel between two neighboring cells, beginning at the origin cell and ending at the destination cell. The goal



is then to identify a least cost path through the raster cost surface. This approach has been widely applied to infrastructure and corridor planning (Huber and Church 1985, Lombard and Church 1993, Lee and Stucky 1998, Yu *et al.* 2003), connectivity measurement (Douglas 1994, Etherington and Holland 2013), robotic travel (Mitchell 1988, Ferguson and Stentz 2006, Daniel *et al.* 2010), and virtual object path planning for video gaming and simulation (Yap 2002, Yap *et al.* 2011). Solution of a least cost path generally involves the construction of an implicit or explicit graph reflecting permissible movement between neighboring cells, with arc attributes representing a distance and travel cost through intervening cells (Huber and Church 1985, Collischonn and Pilar 2000, Church and Murray 2009). Once this is done, an optimal path can be identified using any shortest path solution approach.

Of particular relevance in this chapter is that the least cost path approach has been utilized for addressing what is ultimately an ESP problem, especially in robotics and video gaming (Mitchell 1988, Ferguson and Stentz 2006, Nash *et al.* 2007, Daniel *et al.* 2010, Yap *et al.* 2011). The reason for this is that a raster approach has many benefits (Nash *et al.* 2007): the raster grid itself is a simple and efficient structure; resolution of a raster may be modified; and, raster cells can be updated partially to reflect dynamic change. Additionally, availability of the least cost path approach in many commercial and open source GIS software packages expedite its utilization. At issue, however, is that travel cost in a raster is an approximation of the ESP, possibly good or possibly bad.

While ease and accessibility have facilitated the adoption of the least cost path approach for solving the ESP in some planning contexts, there are important issues that

arise through such an approximation. First, there can be inherent representational error caused by raster grid structure. Cell size very much impacts path cost and quality, as noted in Huber and Church (1985) and Nash *et al.* (2007). Second, cell neighbor definition can also impact path quality. Huber and Church (1985) demonstrate that there are many possible definitions of cell neighbors where connections via arc segments are possible. Addressing this requires enhanced cell resolution and accounting for more connections between cell neighbors (Huber and Church 1985, Yap *et al.* 2011). Third, the treatment of obstacle cells is typically handled through the assignment of high costs so as to discourage travel/utilization (Jalbert and Dobson 1976, Huber and Church 1985), but this does not guarantee that obstacles cells will be excluded from a least cost path. Finally, spatial analysis and processing in a raster environment generally requires major computational resources. The number of cells in a raster layer as well as cell size have an implications for computing capabilities. In a resource-restricted environment under real time operating conditions, a high resolution raster may not be viable. Ultimately, the fundamental issue is whether these issues are significant in using a least cost path approach to approximate the ESP.

### 6.3. Vector representation

As mentioned in the previous section, the visibility graph approach is widely used for deriving the ESP. Unfortunately, computational processing requirements limit the applicability of the visibility graph. To address this, Hong and Murray (2013a, 2013b) and Hong *et al.* (2014) developed approaches for solving the ESP problem. The algorithm, *convexpath*, utilizes GIS functionality and exploits spatial knowledge to

efficiently construct a minimum-sized graph that includes the optimal ESP. Central to convexpath is convex hull and spatial filtering operations, enabling intelligent evaluation of obstacles and vertices. The result is a graph that is significantly smaller in size compared to the visibility graph while still guaranteeing the inclusion of the optimal ESP. Because of this, real time derivation is possible.

Details of this solution approach are now provided, beginning with the convex hull in the context of the ESP.

**Definition (convex hull):** For a set of objects  $S$ , the *convex hull* of  $S$ ,  $CH(S)$ , is the collection of all possible convex combinations of  $S$ .

Given origin and destination points,  $v_o$  and  $v_d$ , and an obstacle  $k$  that impedes straight line travel between them, Hong and Murray (2013a) proved the ESP between two points,  $v_o$  and  $v_d$ , is on the a convex hull for the origin, destination and obstacle,  $CH(v_o, v_d, k)$ . This constitutes the graph, with vertices of the hull representing nodes in the graph and line segments representing arcs in the graph. It is proven that the ESP will be in this graph, or rather along the convex hull when considering a single obstacle. In the case of multiple obstacles, convex hulls can be iteratively constructed to identify necessary vertices and arcs, giving a graph. Hong and Murray (2013a) proved that the resulting graph guarantees inclusion of the ESP.

The convexpath algorithm derives the graph and ESP as follows (Hong *et al.* 2015):

- 1) An origin and destination,  $v_o$  and  $v_d$  respectively, and set of all obstacles  $k \in K$  a

- re given. Graph  $G = \{v_o, v_d, \overline{v_o v_d}\}$ , the set of identified obstacles,  $\Psi = \emptyset$ , and impeded arc/obstacle set,  $\theta = \emptyset$ , are all initialized.
- 2) The shortest path between the origin and destination,  $\delta$ , in  $G$  is derived (representing a spatial filter). If  $\delta \cap \text{int}(k) \neq \emptyset$  where  $k \in K$ , then  $\Psi = \Psi + k$ . Else,  $\delta$  is the optimal ESP.
  - 3) If  $l \cap \text{int}(k) \neq \emptyset$  where  $l \in G$  and  $k \in \Psi$ , then add  $(l, k)$  to  $\theta$ . If  $\theta = \emptyset$ , then go to step 2.
  - 4) For all  $(l, k) \in \theta$ , substitute  $l$  with  $CH(l, k)$ .  $G = G + l'$  for all  $l' \in CH(l, k)$ . If  $v_1 \cap \text{int}(CH(l, k)) \neq \emptyset$  and/or  $v_2 \cap \text{int}(CH(l, k)) \neq \emptyset$  where  $l = \overline{v_1 v_2}$ , then replace  $k$  with faces  $j$  in  $f_j \in LPO(l, k)$ .
  - 5) Go to step 3.

This algorithm relies on a number of spatial analytic operators available in GIS. The primary operators are  $\text{int}(\ )$  (interior of a polygon),  $LPO(\ )$  (line-polygon overlay) and spatial filtering.

**Definition (interior):** Given a polygon  $k$ , the interior of  $k$ , denoted  $\text{int}(k)$ , is the open set bounded by the polygon edges that are disjoint from the edges.

This means that the interior is everything inside the polygon, but excludes the defining edges of the polygon. The interior is important for the ESP because travel is permitted along the boundary/vertex of an obstacle, but not through the interior of the obstacle. The feasibility of a potential travel arc is dependent upon not intersecting the

interior of an obstacle. Thus, generic intersection is not capable of separating the boundary from the interior. To address this, interior intersection is necessary. In the algorithm, interior intersection evaluates whether a line segment intersects the interior of an obstacle  $k$ . In cases where a line segment intersects the interior of an obstacle, it is not feasible and must therefore be replaced by a corresponding convex hull.

**Definition (line-polygon overlay):** For a line segment defined by two points,  $\overline{v_1v_2}$ , such that  $\overline{v_1v_2} \cap \text{int}(k) \neq \emptyset$ , *line-polygon overlay*,  $LPO(\overline{v_1v_2}, k)$ , divides  $k$  into multiple disjoint faces  $f_j$ , defined by segments of  $\overline{v_1v_2}$  and the boundary of the polygon where  $\cup_j f_j = k$ .

Line-polygon overlay is a topological overlay operation in GIS. It is necessary in the algorithm because the convex hull properties assume that vertices  $v_1$  and  $v_2$  lie on the convex hull  $CH(\overline{v_1v_2}, k)$  when  $\overline{v_1v_2}$  intersects the interior of obstacle  $k$ . Thus, overlay enables the algorithm to effectively break up the polygon into faces so that the vertices will lie on the hull for each face. More details can be found in Hong and Murray (2013b).

A final point worth mentioning about the algorithm is the implicit spatial filtering possible, eliminating explicit evaluation of all vertices and/or obstacles. Hong *et al.* (2014) developed a spatial filtering technique that utilizes intermediate shortest paths as spatial filters for obstacle evaluation. Given a set of obstacles  $k \in K$  and some intermediate graph  $G$ , they proved that the intermediate shortest path is a lower bound on the optimal ESP. Given this, obstacles  $k \in K$  where  $\text{int}(k) \cap G \neq \emptyset$  are only considered if they obstruct the intermediate lower bound path at that stage in the algorithm. The spatial filtering technique effectively limits evaluation of obstacles that never impact the

ESP. What is important, however, is that this is done in a manner that preserves ESP optimality properties of the resulting graph.

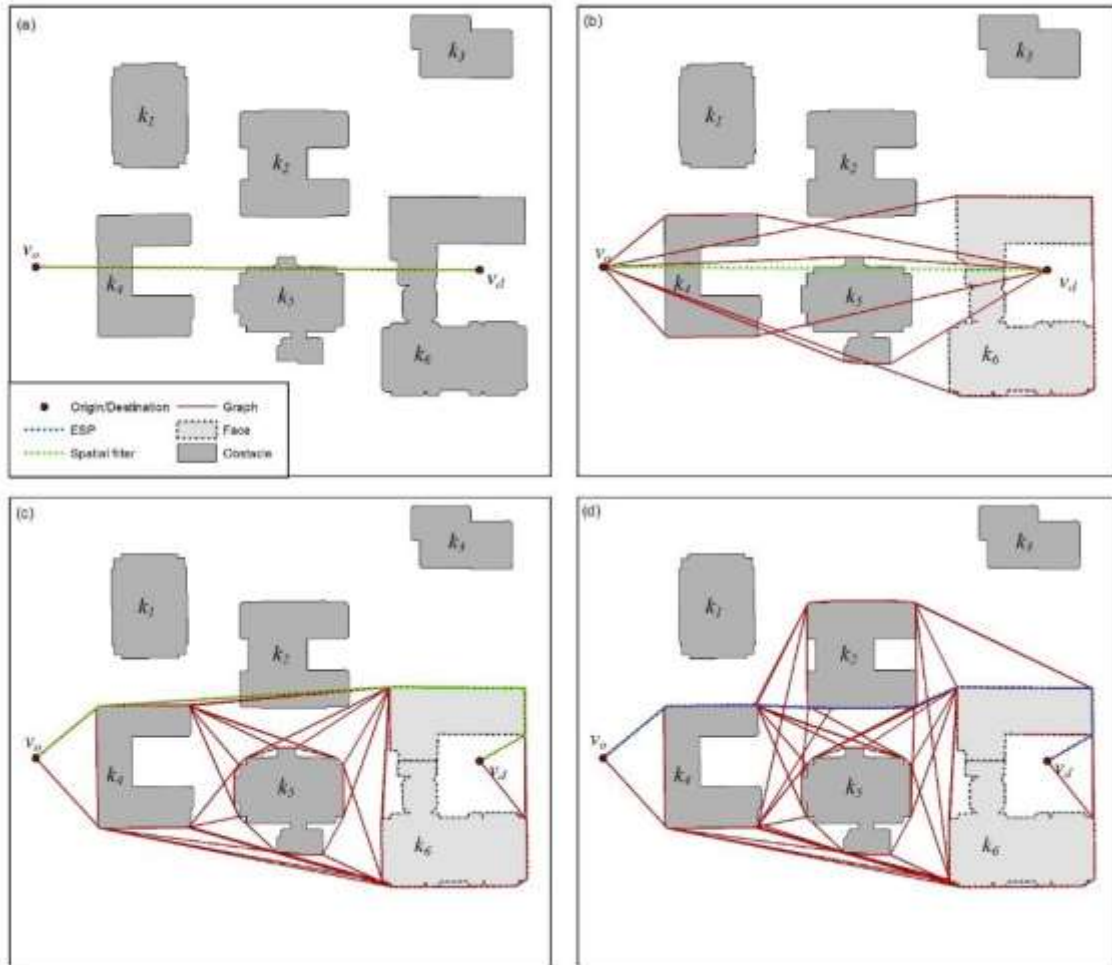


Figure 6.2. Convexpath algorithm graph and path generation process: (a) initialize graph; (b) generate subgraph using line-polygon overlay; (c) evaluate obstacles obstructing arcs on shortest path spatial filter; (d) terminate process as no obstructions to shortest path

The ESP graph and path derivation process is illustrated in Figure 6.2. The convexpath algorithm begins by initializing a graph containing origin and destination

points and a straight line linking them. Then, the shortest path between origin and destination points is derived given the intermediate graph (Figure 6.2a), in this case using Dijkstra's algorithm. Any obstacle that obstructs the shortest path is added to the identified obstacle set,  $\Psi$ . Every obstruction in  $G$  caused by obstacles in  $\Psi$  is resolved by construction of convex hulls,  $CH(\ )$ . If needed, line-polygon overlay,  $LPO(\ )$ , is applied (Figure 6.2b). Once all impediments are resolved for the current path, a new shortest path filter is identified from the graph (Figure 6.2c). In this case, obstacle  $k_2$  is evaluated as it impedes an arc on the spatial filter. These steps continue until no obstruction is found for the shortest path (Figure 6.2d). Obstacles that never impact the ESP, such as  $k_1$  and  $k_3$ , are never explicitly evaluated.

#### 6.4. Raster representation

The least cost path approach for solving the ESP relies on a raster representation of continuous space. Given the raster representation, deriving a least cost path requires a number of important details. First is assignment of an obstacle cost in cells containing all or part of an obstacle. Second is defining movement/travel between neighboring cells. Third is the derivation of travel costs associated with movement between neighboring cells. Finally, with the cost surface, a least cost path can be derived using any shortest path solution approach, such as Dijkstra's algorithm.

An algorithm for identifying a least cost path is as follows:

- 1) Given a raster surface, set of obstacles  $k \in K$ , and origin and destination points,  $v_o$  and  $v_d$ , assign costs for cells containing all or part of an obstacle.

- 2) Define/create a graph reflecting feasible movement between neighboring cells.
- 3) Derive costs on arcs for movement between neighboring cells.
- 4) Solve for the least cost path.

This algorithm is fairly straightforward. An issue, however, is the many operational/analytical details that can impact the quality of the least cost path as an approximation to the ESP. The remainder of this section will examine these details, and discuss their implications for the ESP.

#### *Raster cell resolution/orientation*

As noted, the least cost path approach requires as input a raster representation of a study region. It is not uncommon that this raster surface is given, perhaps a byproduct of a USGS DEM (digital elevation model). If so, the orientation and cell size is already established. If not, then a surface would need to be generated. Decisions about orientation and cell size would therefore be needed. Irrespective, orientation and cell size are very important, with much potential to impact the identified least cost path in terms of efficiency/optimality.

Figure 6.3 illustrates varying raster resolution in relation to the obstacles shown in Figure 6.1a. It is clear that accounting for obstacles in a raster surface can have different spatial impacts when the resolution of a cell is changed. Figure 6.3a, 6.3b, 6.3c, and 6.3d have a cell resolution of 60, 40, 30, and 20 feet, respectively. In general, the obstructed cells impede a larger area than the actual vector obstacles. The area of obstacle cells in Figure 6.3a is 234% larger than the polygonal obstacles (Figure 6.1a). However, the



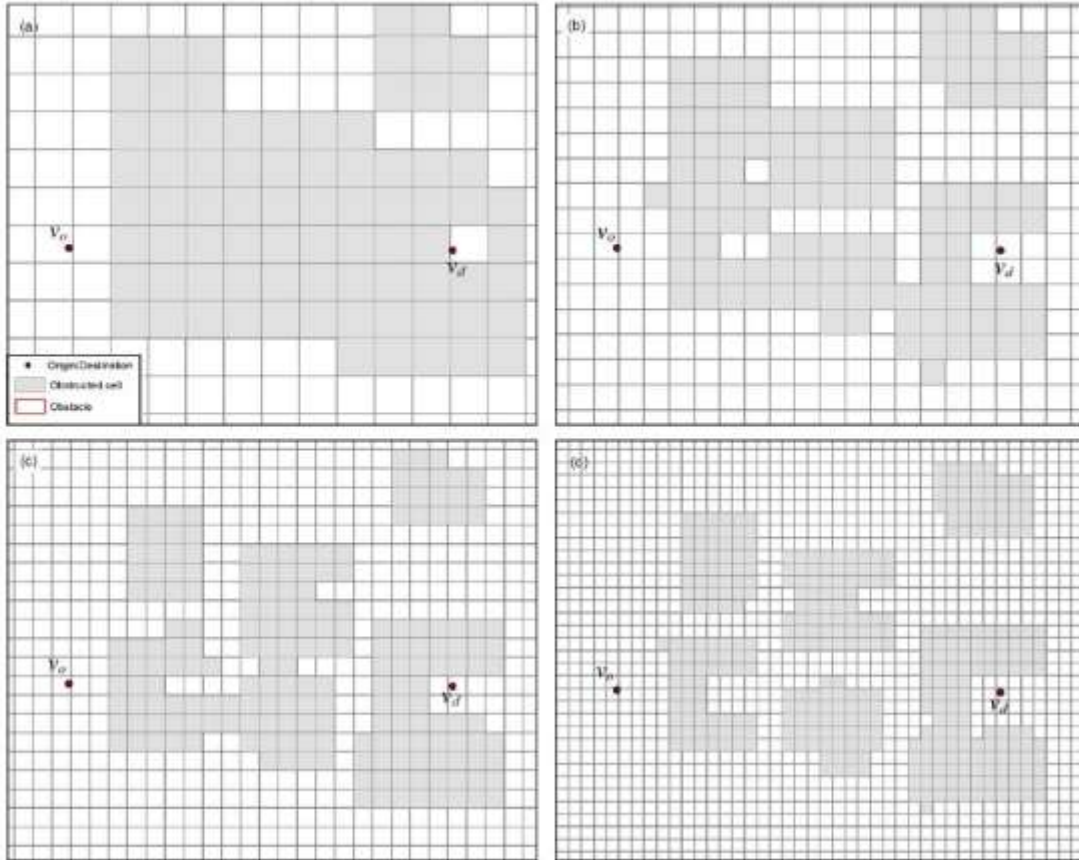


Figure 6.3. Impact of raster resolution on obstacle representation:

(a) 60 feet cell resolution; (b) 40 feet cell resolution;

(c) 30 feet cell resolution; (d) 20 feet cell resolution

spatial implication of this for path movement are pretty significant, altering feasible travel options, or even possibly producing infeasible path. In Figure 6.3a, low cell resolution results in enclosing the destination point. Although extreme travel cost is assigned to each obstacle cell to prevent moving through it, the least cost path approach will derive a path travel through some of the obstacle cells. However, such infeasible path may be considered as feasible if the path does not intersect any vector obstacles, and

Figure 6.3a is possibly an example of this *vector-feasible* path. The obstacle cell area in Figure 3b (40 feet cell resolution) is 164% larger, is 146% larger in Figure 6.3c (30 feet cell resolution), and is 123% larger in Figure 6.3d (20 feet cell resolution) than they actually are (Figure 6.1a). As cell size decreases, raster approximation of obstacles has less error and viable travel paths emerge. Clearly this has significant implications for path efficiency through the surface.

### *Neighbor movement*

Possible travel movement in a raster layer is limited by the definition of neighbor relationships between cells (Huber and Church 1985). When a graph is constructed for path derivation, connection from a node, typically a centroid of a cell, to other nodes are allowed for neighboring nodes. There are several possible neighbor definitions for a grid structure, and this impacts the shape and length of the resulting path (Huber and Church 1985). Figure 6.4 illustrates two commonly used neighbor definitions, namely the rook and queen's case (Goodchild 1977, Van Bemmelen *et al.* 1993). The rook relationship (Figure 6.4a) is defined by cells sharing a non-zero length boundary. The square grid structure of a regular raster representation means that the rook case allows arc connections from the centroid of a given cell to at most four adjacent cells. The queen's case (Figure 6.4b) allows both boundary and vertex sharing conditions in determining neighbors, resulting in a connection of up to eight adjacent cells. While continuous space movement is better approximated by the queen's case, there remains approximation error. This can be reduced by using an extended neighborhood definition (see Huber and Church 1985), but doing so requires considerably more computing effort.

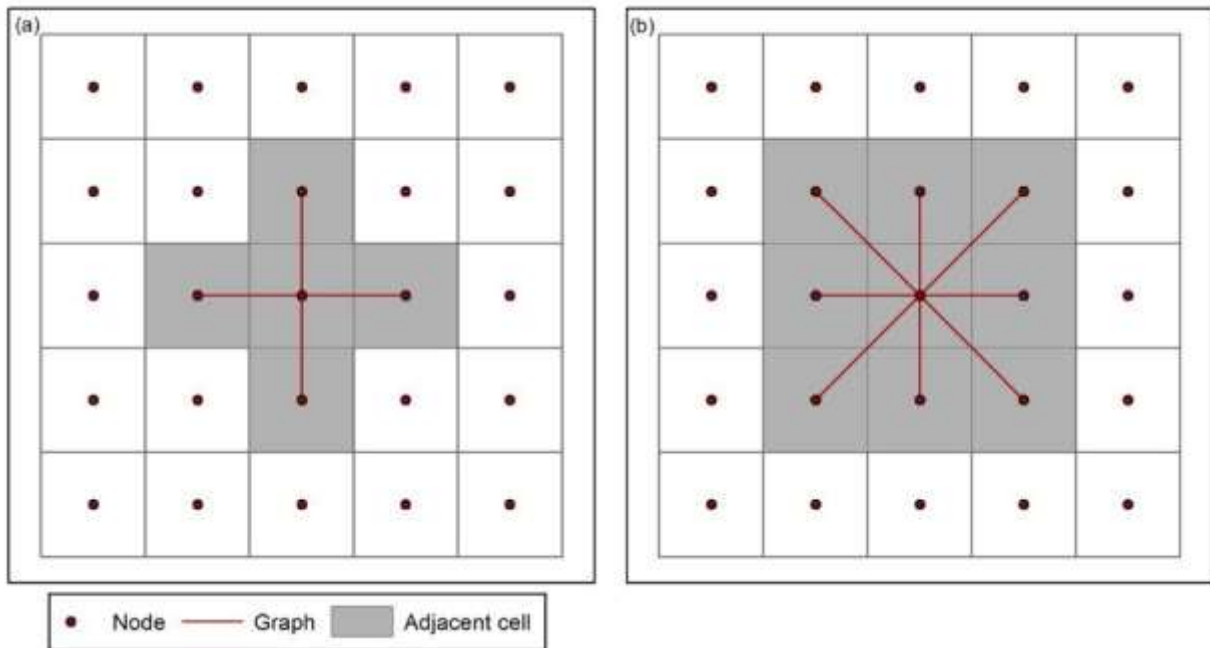


Figure 6.4. Different cell neighbor definitions: (a) rook case; (b) queen's case

### *Arc travel cost estimation*

The least cost path approach requires that arcs have an associated travel cost. This is a function of the length of the arc but also the cost attribute of the cells the arc traverses. Travel cost of an arc is typically derived as a weighted cost in proportion to the length and/or width of the arc through a given cell (Goodchild 1977, Huber and Church 1985, Church and Murray 2009). To discourage travel through cells representing obstacles, extremely high costs are utilized (Jalbert and Dobson 1976, Huber and Church 1985). How this is done is important, and impacts the travel cost of any arc between cells.

In summary, the least cost path approach relying on a raster representation is utilized for ESP approximation in several areas. An important question remains about whether such an approximation is good and reliable in practice.

## 6.5. Study and empirical findings

To assess the least cost path approximation for ESP routing, travel navigation across the Arizona State University (Tempe) campus (Figure 6.5) is considered. The Tempe campus has 178 buildings, each representing obstacles for travel between an origin and destination. The study examines 145 origin-destination pairs selected at random, and every pair has one or more impeding obstacles between them. The ESP (vector representation) is derived using the convexpath algorithm. The least cost path (raster representation) approximation is evaluated using ten different cell resolutions, ranging in size from 100 down to 10 feet, in intervals of 10 feet. The analysis is carried out on an Intel i5 CPU with 12 GB memory system. The convexpath algorithm is implemented in Python 2.7 using open source geospatial libraries. The least cost path approach is available in commercial GIS software, ArcGIS 10.1, and was utilized to identify all reported paths in raster representation.



Figure 6.5. Arizona State University campus buildings

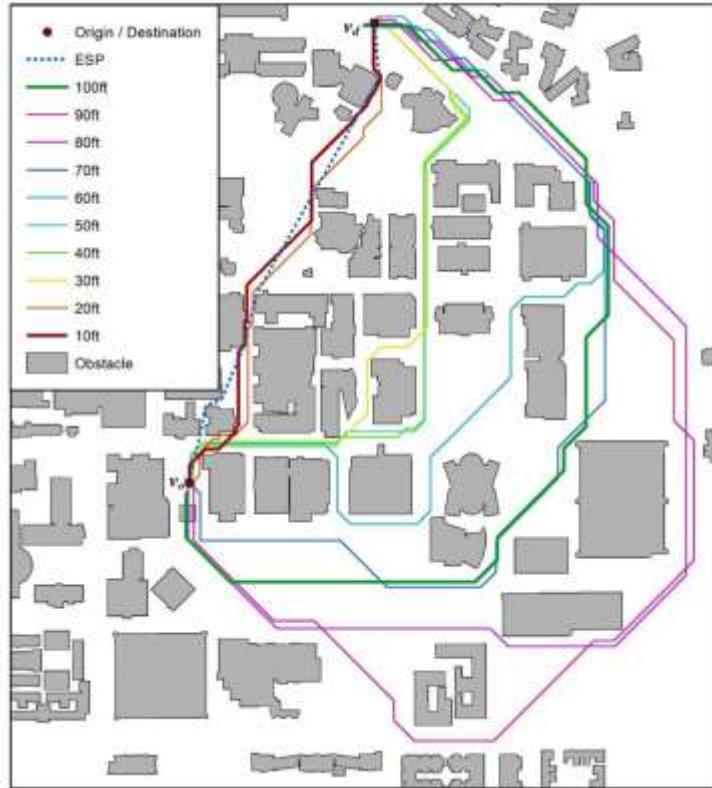


Figure 6.6. ESP and least cost path approximations

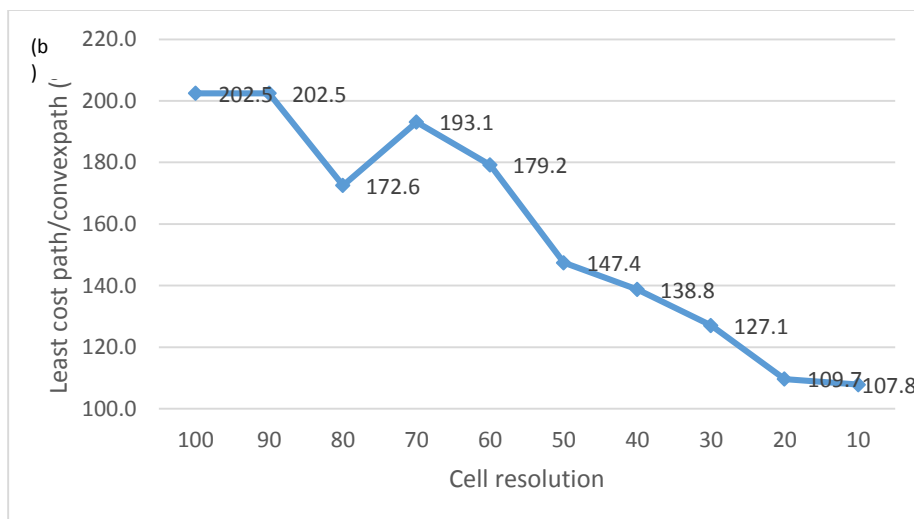
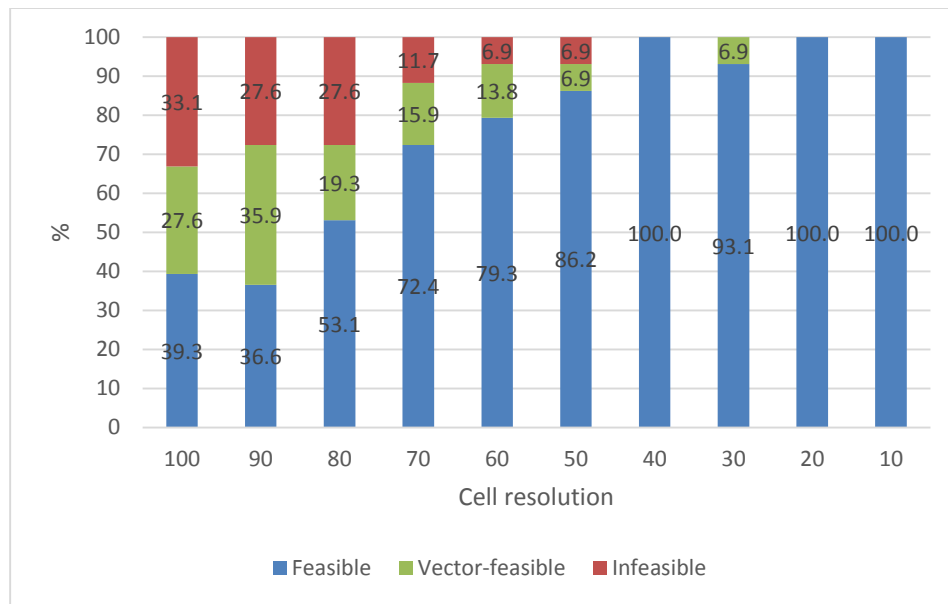
A comparison of the paths identified for one origin-destination pair is shown in Figure 6.6. The ESP derived using the convexpath algorithm was 2,299.4 feet in length, taking 0.51 seconds to compute. The least cost path approximations, obtained using ArcGIS, ranged from 5,006.9 feet in length to 2,499.7 feet. Computing time using ArcGIS ranged from 5.9 to 165.7 seconds. Worth noting in Figure 6 is that the coarse resolution raster representations, 100, 90 and 80 ft cases, actually are infeasible because they route through one or more buildings. Further, the 70 ft path is routed through one or more obstacle cells (not shown), making it technically infeasible as it is supposed to avoid high cost obstacle cells, but the path actually does not intersect the interior of any

building. In general, Figure 6 provides an accurate and consistent highlight of our findings across different raster representations, irrespective of the origin-destination pair. Specifically, coarse raster resolutions are typically infeasible as they intersect buildings (obstacles). Secondly, the spatial variability of the identified paths is significant. Thirdly, spatial variability is not a good thing, as this always translates to less efficiency (an increase in path length). Finally, the solution times associated with raster approximation are considerably more than solving the ESP explicitly as a vector representation.

A summary comparison across all 145 origin-destination pairs for the ten different raster representations is given in Figure 6.7. The three categories of feasibility are illustrated in Figure 6.7a: feasible path, vector-feasible path (infeasible in raster representation), and infeasible path (infeasible in both the raster and vector representations). In particular, Figure 6.7a shows the proportion of the three feasibility types for each raster cell resolution. For example, the 60 ft cell resolution paths were infeasible for 6.9% of the cases, vector-feasible for 13.8% of the cases (raster infeasible, but actually feasible with respect to the vector obstacles) and feasible for 79.4% of the 145 origin-destination path routing cases. The trend, of course, is that the proportion of infeasible cases decreases as the raster representation moves from coarse to finer resolution. Summarized in Figure 6.7b is the average length of the identified path over the optimal ESP length (including only feasible path cases). The general tendency is that raster representation approximations identify paths that are significantly longer compared to the optimal ESP distance. This is expected as the raster representation is an approximation, but the magnitude of the difference is a high of 202.5% of the optimal ESP for the 100 and 90 ft instances down to a low of 107.8% for the 10 ft instances.

Finally, Figure 6.7c indicates the average computing time differences as a ratio of the least cost path computing time over the convexpath algorithm computing time.

Computing time for the least cost path approach steadily increases as raster cell resolution increases. The least cost path approach takes 3,628% longer than the convexpath algorithm to derive a path in 100 feet cell resolution case, on average. This difference increases substantially as cell resolution increases.



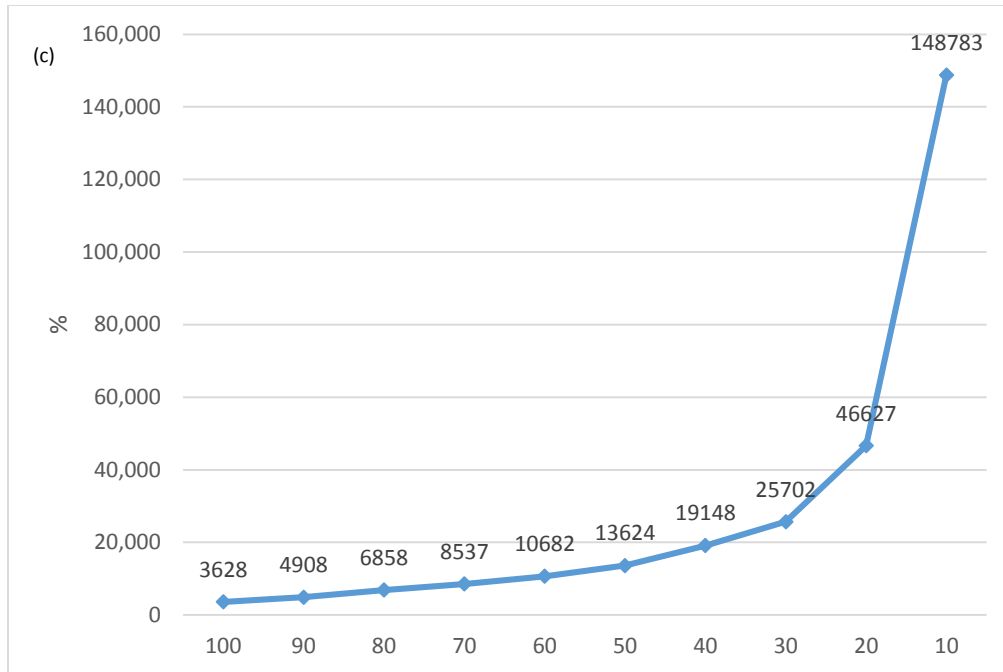


Figure 6.7. Path comparisons:

- (a) portion of feasible, vector-feasible, and infeasible paths;
- (b) proportion of estimated length over the ESP (average);
- (c) percentage of the least cost path computing time over the convexpath algorithm

## 6. Conclusions and discussion

There are number of observations and insights to be gained from the empirical findings presented in the previous section. Coarse raster resolutions typically lead to the identification of paths that are spatially infeasible due to intersection with buildings (obstacles). This is evident in Figure 6, but also in the summary presented in Figure 7a. The spatial variability of identified feasible paths is significant. Figure 6 is very representative of what is found for essentially all origin-destination pairs, that vary



considerably. Of course this explains how so many derived paths can be far longer than the optimal ESP, because they are largely not efficient. An interesting argument for the use of a raster based solution approach is computational efficiency associated with raster GIS. The results demonstrate that this is not the case. Raster based processing always requires more solution time compared to the vector GIS solution approach utilized here, convexpath.

One issue not addressed in this chapter is the impact of neighbor definition. On the one hand, extended neighborhood definitions would have likely been able to identify more efficient paths, at least for the fine resolution cases. However, there would have been a substantial price paid through the need for greater computational effort. Processing time could easily double or more, depending on the neighbor definition used.

While not the focus of this paper, it is worth mentioning that no attempt was made to evaluate other commercial GIS packages. It may be that other packages, like GRASS GIS or IDRISI, may employ more computationally efficient processing functions. Nevertheless, raster cell resolution examined explicitly in the paper would not change. That is, we can expect similar or the same findings.

Obstacle-avoiding shortest path planning is an essential operation for wayfinding, navigation, infrastructure planning and robotic travel. Numerous solution techniques have been developed based on vector and raster representation. On the vector GIS side, the convexpath algorithm derives the ESP efficiently using spatial knowledge and GIS functionality. On the raster GIS side, the least cost path approach is applied in order to approximate an ESP. A comparison of ESP and least cost paths was carried out, based on

examining a number of raster representations, each varying in spatial resolution.

Although the least cost path approach has several benefits for resource-constrained environments, issues regarding path approximation quality and computing efficiency arise. The least cost path is sensitive to representation, and produces low quality shortest paths for coarse raster cell resolutions. Beyond this, no evidence is found that raster based representations enable more efficient route identification. In fact, the empirical results indicate that raster based processing is far more computationally demanding, in contrast to vector GIS approaches that require less than one second.

## CHAPTER 7

### CONCLUSIONS

#### 7.1. Summary

The obstacle-avoiding shortest path between two points in Euclidean space, or the ESP, is an essential element for spatial analysis, location modeling, and wayfinding. However, existing methods that have been widely used to derive the ESP have limited computational capability. In this research, we developed a novel spatial approach, the convexpath algorithm, which is able to derive the ESP with significantly improved efficiency compared to existing methods. We formulated the ESP problem mathematically and provided proofs for essential theorems for the convexpath algorithm. Moreover, we improved the applicability of convexpath by relaxing one restricting assumption of the convexpath algorithm. Next, to improve the efficiency of convexpath in high obstacle density environments, a spatial filtering technique that utilizes the intermediate shortest path as a spatial filter was developed. Empirical wayfinding applications demonstrated significant performance improvement in the ESP derivation. Furthermore, a parallelized version of convexpath was developed to boost the performance of the convexpath algorithm in big data environment, which showed noticeable efficiency gain. Additionally, performance and quality issues of raster-based ESP approximation were analyzed.

Chapter 2 proposed the convexpath algorithm with important theorems and their proofs. By utilizing spatial knowledge and GIS functionality, convexpath was able to construct smaller size graph, as it evaluated fewer obstacles and vertices compared to the visibility graph and local visibility graph. The key spatial operator of convexpath is convex hull construction. Also, it was proved that the convexpath algorithm guarantees the inclusion of the ESP in its resulting graph. Convexpath can derive the ESP with significantly improved efficiency over visibility and local visibility graph.

Chapter 3 extended applicability of the convexpath algorithm to general ESP problems. The initial convexpath algorithm proposed in Chapter 2 had an assumption that limits location of origin, destination, and nodes of resulting graph at the boundary of resulting convex hulls. To address issues with enclosing non-convex obstacles, a line-polygon overlay spatial operator was added to ensure that subproblems will always generate subgraphs that can be used to derive the ESP globally. With this spatial operator, the convexpath algorithm is able to derive the ESP for any discrete obstacle shape, size, or configuration.

Chapter 4 improved the efficiency of the convexpath algorithm with the shortest path spatial filtering technique. Although the convex hull operator filters out irrelevant obstacles efficiently, it can include irrelevant obstacles in high obstacle density environments. The intermediate shortest path was utilized as spatial filter to eliminate unnecessary obstacles from evaluation. This enhanced convexpath with spatial filtering demonstrated significantly improved computational efficiency compared to convexpath without filtering technique.

Chapter 5 developed a parallelized version of the convexpath algorithm to address performance and scale issues. The convexpath algorithm was restructured for parallelization of computationally intensive spatial operators of convexpath. Harnessing the power of multicore CPUs in a single machine, parallelized convexpath showed significantly improved performance compared to the sequential convexpath.

Chapter 6 assessed the issues of computational efficiency and quality of estimation for raster-based ESP approximation technique, compared to vector-based ESP computation. Although the least cost path approach for the ESP approximation has several benefits, comparison results showed that the quality of raster ESP approximation is sensitive to resolution. Moreover, computational efficiency poses a severe challenge to the applicability of the least cost based ESP approximation.

## 7.2. Future work

While much progress has been made in this research to understand and advance the capabilities of deriving and ESP with spatial approaches, there remains a number of important research challenges, especially in terms of applications of the convexpath algorithm.

### 7.2.1. Developing GIS tool & library of the convexpath algorithm

While the convexpath algorithm can compute the ESP with high efficiency, it is not yet readily available for potential end users. To improve its usability, it needs to be developed as a tool for commercial GIS package such as ArcGIS. Furthermore, developing and distributing the convexpath algorithm as an open source geospatial tool would benefit the geospatial community.

### 7.2.2. Commercial delivery drone system

Recently, drones, which refer a range of small-sized unmanned aerial vehicles propelled by multiple rotors, have been utilized for various purpose. Commercial delivery of small products is one of potential economic use for drones. The convexpath algorithm can be applied to route planning of drones from a warehouse to customer's location, as the movement of a drone is not restricted to a road network, while obstacles may still impede its movement. Optimization of commercial delivery system of drones would be able to structured using the convexpath algorithm.

## BIBLIOGRAPHY

- Amdahl, G. M., 1967. Validity of the single processor approach to achieving large scale computing capabilities. In: *Proceedings of the spring joint computer conference*.
- Aneja, Y., and Parlar, M., 1994. Technical Note—Algorithms for weber facility location in the presence of forbidden regions and/or barriers to travel. *Transportation Science*, 28 (1), 70-76.
- Anselin, L., and Rey, S. J., 2010. Perspectives on spatial data analysis. In *Perspectives on Spatial Data Analysis*, 1-20.
- Anselin, L., and Rey, S. J., 2012. Spatial econometrics in an age of CyberGIScience. *International Journal of Geographical Information Science*, 26 (12), 2211-2226.
- Asano, T., 1985. An efficient algorithm for finding the visibility polygon for a polygonal region with holes. *IEICE TRANSACTIONS (1976-1990)*, 68 (9), 557-559.
- Asano, T., Guibas, L., Hershberger, J., and Imai, H., 1986. Visibility of disjoint polygons. *Algorithmica*, 1 (1), 49-63.
- Bailey, T. C., and Gatrell, A. C., 1995. *Interactive spatial data analysis*: Longman Scientific & Technical.
- Barry, W., 2006. *Parallel Programming: Techniques And Applications Using Networked Workstations And Parallel Computers*. 2nd ed: Pearson Education.
- Batta, R., Ghose, A., and Palekar, U. S., 1989. Locating facilities on the Manhattan metric with arbitrarily shaped barriers and convex forbidden regions. *Transportation Science*, 23 (1), 26-36.
- Bekker, J., and Schmid, J., 2006. Planning the safe transit of a ship through a mapped minefield. *ORiON: The Journal of ORSSA*, 22 (1), 1-18.
- Bhattacharya, P., and Gavrilova, M. L., 2007. Voronoi diagram in optimal path planning. In: *Voronoi Diagrams in Science and Engineering, 2007. ISVD '07. 4th International Symposium on*, 9-11 July 2007.
- Bischoff, M., and Klamroth, K., 2007. An efficient solution method for Weber problems with barriers based on genetic algorithms. *European Journal of Operational Research*, 177 (1), 22-41.
- Carling, K., Han, M., and Håkansson, J., 2012. Does Euclidean distance work well when the p-median model is applied in rural areas? *Annals of Operations Research*, 201 (1), 83-97.

- Chang, K.-Y., Jan, G. E., and Parberry, I., 2003. A method for searching optimal routes with collision avoidance on raster charts. *The Journal of Navigation*, 56 (3), 371-384.
- Church, R. L., 2002. Geographical information systems and location science. *Computers & Operations Research*, 29 (6), 541-562.
- Church, R. L., and Murray, A. T., 2009. *Business site selection, location analysis, and GIS*: Wiley.
- Collischonn, W., and Pilar, J. V., 2000. A direction dependent least-cost-path algorithm for roads and canals. *International Journal of Geographical Information Science*, 14 (4), 397-406.
- Cudnik, M. T., Yao, J., Zive, D., Newgard, C., and Murray, A. T., 2012. Surrogate markers of transport distance for out-of-hospital cardiac arrest patients. *Prehospital Emergency Care*, 16 (2), 266-272.
- Daniel, K., Nash, A., Koenig, S., and Felner, A., 2010. Theta\*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 39 (1), 533-579.
- de Berg, M., Cheong, O., and Van Kreveld, M., 2008. *Computational geometry: algorithms and applications*. 3rd ed: Springer.
- de Smith, M. J., Goodchild, M. F., and Longley, P., 2012. *Geospatial analysis: a comprehensive guide to principles, techniques and software tools*: Troubador Publishing.
- Douglas, D. H., 1994. Least-cost path in GIS using an accumulated cost surface and slopelines. *Cartographica: the international journal for Geographic Information and Geovisualization*, 31 (3), 37-51.
- Etherington, T. R., and Holland, E. P., 2013. Least-cost path length versus accumulated-cost as connectivity measures. *Landscape ecology*, 28 (7), 1223-1229.
- Fagerholt, K., Heimdal, S., and Loktu, A., 2000. Shortest path in the presence of obstacles: An application to ocean shipping. *Journal of the operational research society*, 51 (6), 683-688.
- Ferguson, D., and Stentz, A., 2006. Using interpolation to improve path planning: The Field D\* algorithm. *Journal of Field Robotics*, 23 (2), 79-101.
- Fischer, M. M., and Getis, A., 1997. *Recent developments in spatial analysis: spatial statistics, behavioural modelling, and computational intelligence*: Springer.
- Fone, D. L., Christie, S., and Lester, N., 2006. Comparison of perceived and modelled geographical access to accident and emergency departments: a cross-sectional



- analysis from the Caerphilly Health and Social Needs Study. *International Journal of Health Geographics*, 5 (1), 16.
- Fotheringham, A. S., Brunsdon, C., and Charlton, M., 2000. *Quantitative geography: perspectives on spatial data analysis*: Sage.
- Gao, Y., Yang, J., Chen, G., Zheng, B., and Chen, C., 2011. On efficient obstructed reverse nearest neighbor query processing. In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.
- Ghosh, S. K., and Mount, D. M., 1991. An output sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20 (5), 888-910.
- Golledge, R. G., Klatzky, R. L., Loomis, J. M., Speigle, J., and Tietz, J., 1998. A geographical information system for a GPS based personal guidance system. *International Journal of Geographical Information Science*, 12 (7), 727-749.
- Gong, Z., Tang, W., Bennett, D. A., and Thill, J.-C., 2013. Parallel agent-based simulation of individual-level spatial interactions within a multicore computing environment. *International Journal of Geographical Information Science*, 27 (6), 1152-1170.
- Goodchild, M., 1977. An evaluation of lattice solutions to the problem of corridor location. *Environment and Planning a*, 9 (7), 727-738.
- Guan, X., and Wu, H., 2010. Leveraging the power of multi-core platforms for large-scale geospatial data processing: Exemplified by generating DEM from massive LiDAR point clouds. *Computers & Geosciences*, 36 (10), 1276-1282.
- Guibas, L. J., and Hershberger, J., 1989. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39 (2), 126-152.
- Güven, G., Ergen, E., Erberik, M., Kurc, O., and Birgönül, M., 2012. Providing guidance for evacuation during emergency based on a real-time damage and vulnerability assessment of facilities. In: *ASCE International Conference on Computing in Civil Engineering*.
- Habib, M. K., and Asama, H., 1991. Efficient method to generate collision Free paths for an autonomous mobile robot based on new Free space structuring approach. In: *IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91*.
- Haynes, R., Jones, A. P., Sauerzapf, V., and Zhao, H., 2006. Validation of travel times to hospital estimated by GIS. *International Journal of Health Geographics*, 5 (1), 40.

- Hershberger, J., and Suri, S., 1993. Efficient computation of Euclidean shortest paths in the plane. In: *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*.
- Hershberger, J., and Suri, S., 1999. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28 (6), 2215-2256.
- Higgs, G., 2009. The role of GIS for health utilization studies: literature review. *Health Services and Outcomes Research Methodology*, 9 (2), 84-99.
- Hong, I., and Murray, A. T., 2013a. Efficient measurement of continuous space shortest distance around barriers. *International Journal of Geographical Information Science*, 27 (12), 2302-2318.
- Hong, I., and Murray, A. T., 2013b. Efficient wayfinding in complex environments: derivation of a continuous space shortest path. In: *Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Computational Transportation Science*.
- Hong, I., Murray, A. T., and Rey, S. J., 2015. High Performance Computing to Derive an Obstacle-Avoiding Shortest Path.
- Hong, I., Murray, A. T., and Wolf, L. J., 2014. Spatial Filtering to Enhance Solution Efficiency in Euclidean Shortest Path Derivation.
- Huber, D. L., and Church, R. L., 1985. Transmission corridor location modeling. *Journal of Transportation Engineering*, 111 (2), 114-130.
- Jalbert, J. S., and Dobson, J. E. 1976. Cell-based land use screening procedure for regional siting analysis: Oak Ridge National Lab.
- Jones, S. G., Ashby, A. J., Momin, S. R., and Naidoo, A., 2010. Spatial Implications Associated with Using Euclidean Distance Measurements and Geographic Centroid Imputation. *Health Services Research*, 45 (1), 316-327.
- Jordan, H., Roderick, P., Martin, D., and Barnett, S., 2004. Distance, rurality and the need for care: access to health services in South West England. *International Journal of Health Geographics*, 3 (1), 21.
- Katz, I. N., and Cooper, L., 1981. Facility location in the presence of forbidden regions, I: Formulation and the case of Euclidean distance with one forbidden circle. *European Journal of Operational Research*, 6 (2), 166-173.
- Kim, D. S., Yu, K., Cho, Y., Kim, D., and Yap, C., 2004. Shortest paths for disc obstacles. *Computational Science and Its Applications-ICCSA 2004*, 62-70.
- Klamroth, K., 2001a. Planar Weber location problems with line barriers. *Optimization*, 49 (5-6), 517-527.

- Klamroth, K., 2001b. A reduction result for location problems with polyhedral barriers. *European Journal of Operational Research*, 130 (3), 486-497.
- Kulyukin, V., Gharpure, C., Nicholson, J., and Osborne, G., 2006. Robot-assisted wayfinding for the visually impaired in structured indoor environments. *Autonomous Robots*, 21 (1), 29-41.
- Kurata, T., Kouroggi, M., Ishikawa, T., Kameda, Y., Aoki, K., and Ishikawa, J., 2011. Indoor-outdoor navigation system for visually-impaired pedestrians: Preliminary evaluation of position measurement and obstacle display. In: *Wearable Computers (ISWC), 2011 15th Annual International Symposium on Wearable Computers*.
- Kwan, M.-P., and Lee, J., 2005. Emergency response after 9/11: the potential of real-time 3D GIS for quick emergency response in micro-spatial environments. *Computers, Environment and Urban Systems*, 29 (2), 93-113.
- Lanthier, M., Nussbaum, D., and Sack, J.-R., 2003. Parallel implementation of geometric shortest path algorithms. *Parallel Computing*, 29 (10), 1445-1479.
- Larson, R. C., and Sadiq, G., 1983. Facility locations with the Manhattan metric in the presence of barriers to travel. *Operations Research*, 652-669.
- Lee, J.-k., Eastman, C. M., Lee, J., Kannala, M., and Jeong, Y.-s., 2010. Computing walking distances within buildings using the universal circulation network. *Environment and planning. B, Planning & design*, 37 (4), 628.
- Lee, J., and Stucky, D., 1998. On applying viewshed analysis for determining least-cost paths on Digital Elevation Models. *International Journal of Geographical Information Science*, 12 (8), 891-905.
- Li, Z., Gao, Y., and Lu, Y., 2011. Continuous obstructed range queries in spatio-temporal databases. In: *System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2011 International Conference on*.
- Lombard, K., and Church, R., 1993. The gateway shortest path problem: generating alternative routes for a corridor location problem. *Geographical systems*, 1 (1), 25-45.
- Loomis, J. M., Klatzky, R. L., and Golledge, R. G., 2001. Navigating without vision: basic and applied research. *Optometry & Vision Science*, 78 (5), 282-289.
- Loomis, J. M., Marston, J. R., Golledge, R. G., and Klatzky, R. L., 2005. Personal guidance system for people with visual impairment: A comparison of spatial displays for route guidance. *Journal of Visual Impairment & Blindness*, 99 (4), 219.

- Lozano-Perez, T., 1987. A simple motion-planning algorithm for general robot manipulators. *Robotics and Automation, IEEE Journal of robotics and automation*, 3 (3), 224-238.
- Lozano-Pérez, T., and Wesley, M. A., 1979. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22 (10), 560-570.
- Martin, D., Roderick, P., Diamond, I., Clements, S., and Stone, N., 1998. Geographical aspects of the uptake of renal replacement therapy in England. *International Journal of Population Geography*, 4 (3), 227-242.
- Martin, D., Wrigley, H., Barnett, S., and Roderick, P., 2002. Increasing the sophistication of access measurement in a rural healthcare study. *Health & Place*, 8 (1), 3-13.
- Mitchell, J., 1989. An optimal algorithm for shortest rectilinear paths among obstacles. In: *Abstracts 1st Canadian Conference of Computational Geometry*.
- Mitchell, J. S., 1988. An algorithmic approach to some problems in terrain navigation. *Artificial Intelligence*, 37 (1), 171-201.
- Mitchell, J. S. B., 1996. Shortest paths among obstacles in the plane. *International Journal of Computational Geometry & Applications*, 6 (3), 309-332.
- Mitchell, J. S. B., 1999. Geometric Shortest Paths and Network Optimization. In *Handbook of Computational Geometry*, eds. J. R. Sack and J. Urrutia, 633-702. New York: Elsevier.
- Murray, A. T., 2010. Advances in location modeling: GIS linkages and contributions. *Journal of geographical systems*, 12 (3), 335-354.
- Nash, A., Daniel, K., Koenig, S., and Felner, A., 2007. Theta\*: Any-Angle Path Planning on Grids. In: *Proceedings of the National Conference on Artificial Intelligence*.
- O'Sullivan, D., and Unwin, D., 2010. *Geographic Information Analysis*: Wiley.
- Openshaw, S., and Turton, I., 1999. *High Performance Computing and the Art of Parallel Programming: An Introduction for Geographers, Social Scientists and Engineers*: Taylor & Francis.
- Opong, J. R., and Hodgson, M. J., 1994. Spatial accessibility to health care facilities in Suhum District, Ghana. *The Professional Geographer*, 46 (2), 199-209.
- Papadopoulou, E., and Lee, D., 1995. Efficient computation of the geodesic Voronoi diagram of points in a simple polygon. *Algorithms—ESA'95*, 238-251.

- Papadopoulou, E., and Lee, D., 1998. A new approach for the geodesic Voronoi diagram of points in a simple polygon and other restricted polygonal domains. *Algorithmica*, 20 (4), 319-352.
- Phibbs, C. S., and Luft, H. S., 1995. Correlation of travel time on roads versus straight line distance. *Medical Care Research and Review*, 52 (4), 532-542.
- Pocchiola, M., and Vegter, G., 1996. Minimal tangent visibility graphs. *Computational Geometry*, 6 (5), 303-314.
- Qin, Y.-Q., Sun, D.-B., Li, N., and Cen, Y.-G., 2004. Path planning for mobile robot using the particle swarm optimization with mutation operator. In: *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*.
- Ran, L., Helal, S., and Moore, S., 2004. Drishti: an integrated indoor/outdoor blind navigation system and service. In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications*.
- Rey, S. J., Anselin, L., Pahle, R., Kang, X., and Stephens, P., 2013. Parallel optimal choropleth map classification in PySAL. *International Journal of Geographical Information Science*, 27 (5), 1023-1039.
- Riehle, T., Lichter, P., and Giudice, N., 2008. An indoor navigation system to support the visually impaired. In: *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*.
- Rogerson, P. A., 2010. *Statistical Methods for Geography: A Student's Guide*. 2nd ed: Sage.
- Rohnert, H., 1986. Shortest paths in the plane with convex polygonal obstacles. *Information Processing Letters*, 23 (2), 71-76.
- Schmickl, T., and Crailsheim, K., 2007. A Navigation Algorithm for Swarm Robotics Inspired by Slime Mold Aggregation. In *Swarm Robotics*, eds. E. Şahin, W. Spears and A. T. Winfield, 1-13: Springer.
- Stock, R., 1983. Distance and the utilization of health facilities in rural Nigeria. *Social Science & Medicine*, 17 (9), 563-570.
- Szymanski, M., Breitling, T., Seyfried, J., and Wörn, H., 2006. Distributed Shortest-Path Finding by a Micro-robot Swarm. In *Ant Colony Optimization and Swarm Intelligence*, eds. M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli and T. Stützle, 404-411: Springer.

- Takahashi, O., and Schilling, R., 1989. Motion planning in a plane using generalized Voronoi diagrams. *Robotics and Automation, IEEE Transactions on*, 5 (2), 143-150.
- Tsou, M.-C., 2010. Integration of a geographic information system and evolutionary computation for automatic routing in coastal navigation. *Journal of Navigation*, 63 (2), 323.
- Van Bemmelen, J., Quak, W., Van Hekken, M., and Van Oosterom, P., 1993. Vector vs. raster-based algorithms for cross country movement planning. In: *Autocarto Conference*.
- Viegas, J., and Hansen, P., 1985. Finding shortest paths in the plane in the presence of barriers to travel (for any  $l_p$ -norm). *European Journal of Operational Research*, 20 (3), 373-381.
- Wangdahl, G. E., Pollock, S., and Woodward, J. B., 1974. Minimum-trajectory pipe routing. *Journal of Ship Research*, 18 (1)
- Welzl, E., 1985. Constructing the visibility graph for n-line segments in  $O(n^2)$  time. *Information Processing Letters*, 20 (4), 167-171.
- Wilson, J., Walker, B. N., Lindsay, J., Cambias, C., and Dellaert, F., 2007. Swan: System for wearable audio navigation. In: *11th IEEE International Symposium on Wearable Computers*.
- Xia, Y.-j., Kuang, L., and Li, X.-m., 2011. Accelerating geospatial analysis on GPUs using CUDA. *Journal of Zhejiang University SCIENCE C*, 12 (12), 990-999.
- Yao, J., Murray, A. T., Agadjanian, V., and Hayford, S. R., 2012. Geographic influences on sexual and reproductive health service utilization in rural Mozambique. *Applied Geography*, 32 (2), 601-607.
- Yap, P., 2002. Grid-based path-finding. In *Advances in Artificial Intelligence*, 44-55: Springer.
- Yap, P., Burch, N., Holte, R. C., and Schaeffer, J., 2011. Block A\*: Database-Driven Search with Applications in Any-Angle Path-Planning. In: *AAAI Conference on Artificial Intelligence*.
- Yu, C., Lee, J., and Munro-Stasiuk, M. J., 2003. Extensions to least-cost path algorithms for roadway planning. *International Journal of Geographical Information Science*, 17 (4), 361-376.
- Zhang, J., 2010. Towards personal high-performance geospatial computing (HPC-G): perspectives and a case study. In: *Proceedings of the ACM SIGSPATIAL*

*International Workshop on High Performance and Distributed Geographic Information Systems.*

- Zhang, J., Papadias, D., Mouratidis, K., and Manli, Z., 2005. Query processing in spatial databases containing obstacles. *International Journal of Geographical Information Science*, 19 (10), 1091-1111.
- Zhang, L., Zhang, L., Peng, R., Li, G., and Zou, W., 2011. Determination of the Shortest Time Route Based on the Composite Influence of Multidynamic Elements. *Marine Geodesy*, 34 (2), 108-118.