Computers, Environment and Urban Systems 55 (2016) 1–10

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/ceus

# Obstacle-avoiding shortest path derivation in a multicore computing environment

Insu Hong [a,*], Alan T. Murray [b], Sergio Rey [c]

[a] Department of Geology and Geography, Eberly College of Arts & Sciences, West Virginia University, United States
[b] Center for Spatial Analytics and Geocomputation, College of Computing and Informatics, Drexel University, United States
[c] GeoDa Center for Geospatial Analysis and Computation, School of Geographical Sciences and Urban Planning, Arizona State University, United States

ARTICLE INFO

*Article history:*
Received 30 March 2015
Received in revised form 1 October 2015
Accepted 1 October 2015
Available online 22 October 2015

*Keywords:*
Euclidean shortest path
Convex hull
Vector overlay
High performance computing
Parallelization
GIS

ABSTRACT

The best obstacle avoiding path in continuous space, referred to as the Euclidean shortest path, is important for spatial analysis, location modeling and wayfinding tasks. This problem has received much attention in the literature given its practical application, and several solution techniques have been proposed. However, existing approaches are limited in their ability to support real time analysis in big data environments. In this research a multicore computing approach is developed that exploits spatial knowledge through the use of geographic information system functionality to efficiently construct an optimal shortest path. The approach utilizes the notion of a convex hull for iteratively evaluating obstacles and constructing pathways. Further, the approach is capable of incrementally improving bounds, made possible through parallel processing. Wayfinding routes that avoid buildings and other obstacles to travel are derived and discussed.



## 1. Introduction

The shortest path in continuous space around obstacles reflects behavior, movement and travel. Often such a path is not recorded or known a priori, yet is an essential and assumed input for spatial analytics (see Batta, Ghose, & Palekar, 1989; Bailey & Gatrell, 1995; Fotheringham, Brunsdon, & Charlton, 2000; Fagerholt, Heimdal, & Loktu, 2000; Klamroth, 2001; de Smith, Goodchild, & Longley, 2007; O'Sullivan & Unwin, 2010; Rogerson, 2010). Navigation and wayfinding also rely on efficient obstacle avoiding paths (Lozano-Pérez & Wesley, 1979; Hong & Murray, 2013b), requiring its derivation in realtime to support travel. A situation involving a single obstacle is shown in Fig. 1 where one is seeking the shortest possible path that avoids the obstacle.

### 1.1. Euclidean shortest path

The efficient obstacle avoiding route through space 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 & Wesley, 1979; Welzl, 1985; Ghosh & Mount, 1991; Pocchiola & Vegter, 1996), local visibility graph (Kim, Yu, Cho, Kim, & Yap, 2004; Zhang, Papadias, Mouratidis, & Manli, 2005; Gao, Yang, Chen, Zheng, & Chen, 2011), shortest path map (Mitchell, 1999), and Voronoi diagrams (Papadopoulou & Lee, 1995) have been applied to solve the ESP problems. The ESP supports wayfinding and navigation for robotics (Lozano-Pérez & Wesley, 1979; Habib & Asama, 1991; Schmickl & Crailsheim, 2007), trans-oceanic shipping (Fagerholt et al., 2000; Bekker & Schmid, 2006; Bhattacharya & Gavrilova, 2007), location modeling (Batta et al., 1989; Fagerholt et al., 2000; Klamroth, 2001), and more.

Methods to date for deriving the ESP construct a graph in order to reduce combinatorial search from infinite routing options through continuous space to a finite, discrete set of line segments that define a graph, guaranteeing inclusion of the ESP. However, the efficiency of existing methods is considerably limited by the need to evaluate most/all obstacle and region boundary vertices within a given area during graph construction (Hong & Murray, 2013a). Even approaches employing filtering techniques, such as the local visibility graph (Kim et al., 2004; Zhang et al., 2005), remain computationally intensive (Hong & Murray, 2013a). Therefore, these classic approaches have not been capable of supporting real-time, big data planning and analysis contexts.

Recent work by Hong and Murray (2013a, 2013b) has focused on new methods for deriving an ESP more efficiently. They exploit spatial knowledge and geographic information system (GIS) functionality in graph derivation, in a manner that guarantees inclusion of the optimal ESP. By utilizing the notion of a convex hull, together with spatial operators, they explicitly consider only relevant obstacles impeding a given origin–destination pair, thereby enabling construction of a more efficient graph.

* Corresponding author.
*E-mail address:* iaminsu@gmail.com (I. Hong).

http://dx.doi.org/10.1016/j.compenvurbsys.2015.10.001
0198-9715/

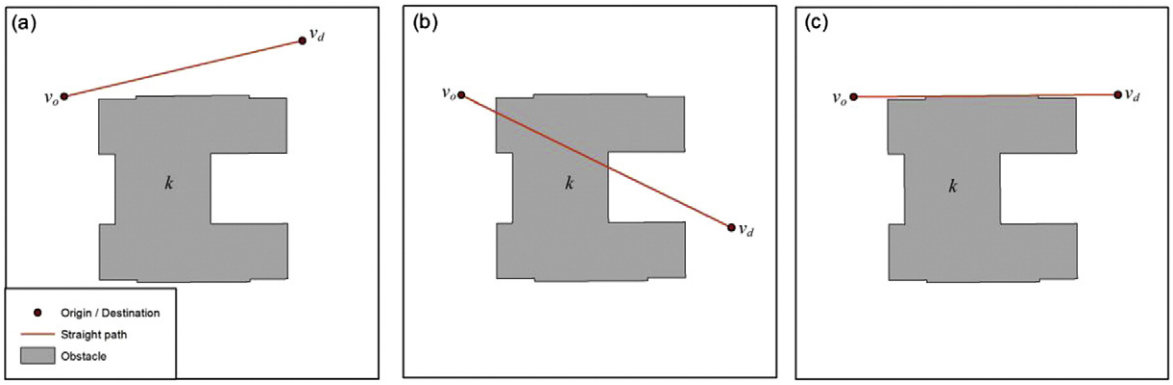


**Fig. 1.** Three instances of obstacle avoiding movement: (a) no intersection, (b) interior intersection, (c) boundary intersection.

Real-time derivation of paths, however, requires highly efficient methods, beyond current capabilities. When obstacle density is high, such as urban environments and building interiors, computational performance of existing approaches is significantly degraded. This is largely due to computationally intensive spatial operators that need to be repeatedly utilized. As a result, large data applications supporting navigation and wayfinding remain a challenge to support.

### *1.2. Parallelization approaches for spatial analysis*

Techniques taking advantage of parallel computing environments have proven useful for addressing various performance and scale issues in spatial analysis (Griffith, 1990; Armstrong, 2000; Lanthier, Nussbaum, & Sack, 2003; Zhang, 2010; Anselin & Rey, 2012; Rey, Anselin, Pahle, Kang, & Stephens, 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, Kuang, & Li, 2011; Anselin & Rey, 2012). These methods of parallelization exploit the computing power of multiple CPU/GPU cores in either single or multiple machines to boost performance, by undertaking many tasks concurrently. Strategies for parallelization can be categorized as task and data oriented (Barry, 2006; Gong, Tang, Bennett, & Thill, 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 & 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 viewshed 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 & Wu, 2010), but they also enable more realistic representation of real world processes compared to sequential processing (Openshaw & Turton, 1999). However, complete reconstruction and/or reconceptualization of existing algorithms is often required to achieve significant parallelization efficiencies (Anselin & Rey, 2012).

### *1.3. Purpose*

In this paper, an ESP derivation approach is introduced that utilizes spatial filtering and parallel computing. The spatial filtering technique dramatically reduces the number of obstacles that must be evaluated. The parallel algorithm significantly improves performance by distributing spatial queries and hull construction. The next section details the ESP and summarizes algorithms for solving it. A new parallel solution approach is then introduced. Application results are presented, followed by discussion and concluding comments.

## 2. Shortest path derivation

In this research, a solution approach is proposed to derive a path that avoids obstacles, solving the ESP. This algorithm, referred to as parallel convex graph (PCG), utilizes a novel spatial filtering technique combined with a parallel computing approach to efficiently identify the optimal path. This approach effectively extends preliminary work detailed in Hong and Murray (2013a, 2013b).

### *2.1. Convexpath algorithm*

The convexpath algorithm introduced in Hong and Murray (2013a, 2013b) operates based on the construction of a minimum size graph through which the optimal path can be found. It explicitly accounts for only relevant obstacles for a given origin–destination pair by utilizing geometric properties of convex hulls. Three spatial operators are extensively relied upon: interior intersection, convex hull, and line-polygon overlay.

*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.

*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 $S$.

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

The convexpath algorithm uses convex hull and interior intersection operators to identify obstacles and construct a valid graph. Let's assume a convex hull for three spatial objects, two points that represent the origin and destination, $v_o$, $v_d$, and an obstacle $k$. Based on the minimum