In-field Routing for Agricultural Vehicles through Convex Hull Generation

by

Sen Sun

submitted to the Faculty of Chemistry and Earth Sciences
in partial fulfillment of the requirements for the degree of
Master of Science (M. Sc.) in Geography
at
Heidelberg University
(Ruprecht-Karls-Universität Heidelberg)

Feb. 28, 2015

Supervisors:
Jun. Prof. Dr. Bernhard Höfle
Prof. Dr. Alexander Zipf

Statement of Originality

I hereby confirm that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgements. This applies also to all graphics, drawings, maps and images included in the thesis.

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig verfasst wurde und dass keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden. Diese Erklärung erstreckt sich auch auf in der Arbeit enthaltene Graphiken, Zeichnungen, Kartenskizzen und bildliche Darstellungen.

Place and Date Signature

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to Prof. Höfle and Prof. Zipf for taking their precious time to supervise this thesis, their useful comments, remarks and practical advices through the learning process. Also, I would like to thank Johannes Lauer for introducing me to the topic as well for the support on the way. Furthermore, I'd appreciate the previous work of all the members in the TeleAgro+ research team, without which this master thesis would not be possible.

Abstract

This master thesis aims to implement an approximation algorithm for finding the shortest path on a two dimensional surface with potential obstacles. The algorithm is especially geared at in-field routing for agricultural vehicles, where the obstacles and the boundary are inclined to exhibit complex geometries, whereas the total amount of obstacles is in general moderate.

The algorithm consists of two major stages. In the first stage, it generates a graph with non-colliding edges based on the recursive generation and transformation of convex hulls. In the second stage, the Dijkstra algorithm is applied to find the shortest path on this graph.

To evaluate the performance and the feasibility of the proposed algorithm, 100 test cases are conducted with real field data, derived from historical GPS trajectories, with both the proposed algorithm and the visibility graph algorithm. The results show that the proposed algorithm could reduce the running time significantly while delivering the same result. For a single-shot request, in which the graph should be generated on an ad hoc basis, the proposed algorithm needs in average only 5% of the time needed by the brute-force implementation of the visibility graph counterpart. Even for repeated requests, where the visibility graph approach could trade some space for time efficiency through caching, the proposed algorithm is still at a slight advantage.

Keywords: shortest path, convex hull, in-field routing

Kurzfassung

Das Ziel dieser Masterarbeit ist es, einen Algorithmus für die Suche nach dem kürzesten Weg auf einer zweidimensionalen Oberfläche mit Hindernissen zu implementieren. Der Algorithmus eignet sich speziell für die Anwendung im Infield-Routing landwirtschaftlicher Fahrzeuge, wo die Hindernisse und die Grenzen in der Regel komplexe Geometrien aufweisen, die Gesamtmenge von Hindernissen auf einem Feld allerdings gering ist.

Der Algorithmus besteht aus zwei Schritten: Als erster Schritt wird ein Graph mit kollisionsfreien Kanten durch die rekursive Erzeugung sowie Transformation der Konvexhülle erzeugt. Im zweiten Schritt wird der Dijkstra-Algorithmus verwendet, um den kürzesten Weg auf diesem Graphen abzuleiten.

Um die Leistung und die Eignung dieses Algorithmus zu bewerten, werden 100 Testfälle auf realen Felddaten, die von historischen GPS-Trajektorien abgeleitet sind, mit sowohl dem vorgeschlagenen Algorithmus als auch dem Sichtbarkeitsgraphen-Algorithmus durchgeführt. Die Ergebnisse zeigen, dass der vorgeschlagene Algorithmus die gleichen Routen mit einer erheblich reduzierten Laufzeit liefert. Bei einer einmaligen Anfrage, bei der der Graph auf einer Ad-hoc-Basis erzeugt werden muss, braucht der vorgeschlagene Algorithmus im Schnitt nur knapp 5% der Zeit, die der Sichtbarkeitsgraph-Algorithmus benötigt. Auch bei wiederholten Anfragen, bei denen der Sichtbarkeitsgraph-Ansatz die Laufzeit durch Caching des Graphen reduzieren kann, bleibt der vorgeschlagene Algorithmus in einem leichten Vorteil.

Stichwörter: kürzester Weg, Konvexe Hülle, Infield-Routing

Chapter 1 Introduction

A point to point routing problem can be defined as the derivation of the optimal path from one position to another. Such a problem usually consists of three elements: data, criteria and algorithms. Data specify the geometry, as well as the attributes of possible paths or passable areas. The optimal path could be defined by different criteria. It could be the shortest, the fastest, the most economical path and so on. The algorithms then start from the available data and calculate the optimal path, locally or globally, under the given criteria.

These three elements vary from domain to domain. In the agriculture domain, these three elements also have their particularities: The emerging application of drones and sensors are contributing more data than ever, enabling navigation with higher precision; When evaluating a path, both the parameters of the machine and its impact on the environment should be considered; Also, unlike urban traffic, agricultural traffic is not limited to a certain road network, it often occurs on a surface with boundary and obstacles, which means that algorithms for graph generation are prerequisites for routing.

However, the rationale of paying more attention to agricultural routing comes not only from its particularities, but also from its importance.

Ever since the industry revolution, great efforts have been made to improve the productivity in the primary sector. Considering the fact that the demand of affordable and safe food continues to increase as the result of growing world population, and that the percentage of population involved in agricultural production decreases steadily as the result of urbanization, it is reasonable to conclude that the need to increase the efficiency of agricultural production

will continue to exist in the future. According to the forecasts of World Bank (Figure 1.1[1]), until 2050 the total population on our planet will increase to approximately 9.5 billion (from 7.3 billion in 2015), while the total percentage of rural population will decrease to 33.8% (from 46.1% in 2015).

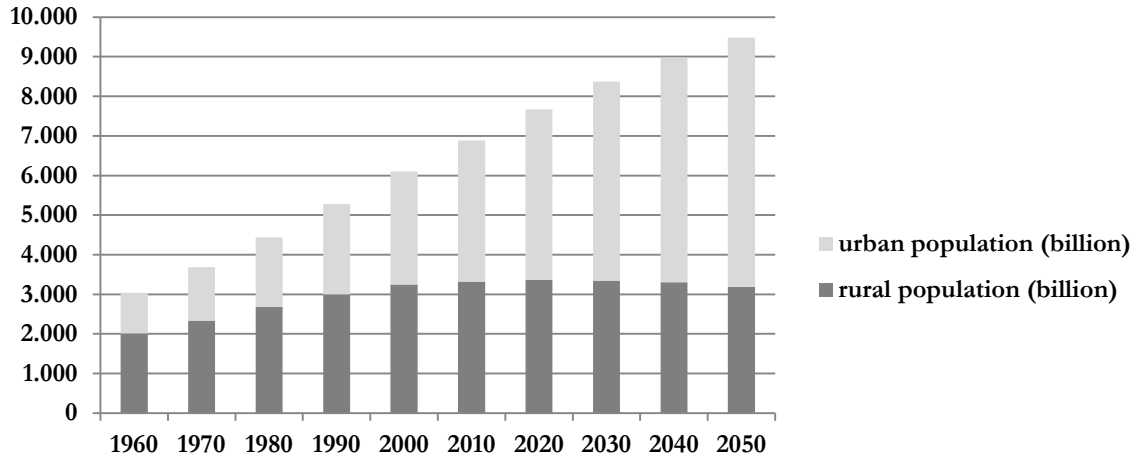


Figure 1. 1 Prediction for Urban and Rural Population

There are many different ways to improve productivity in the agriculture sector. Three important aspects highly related to path planning are mechanization, automation and optimization.

Machinery of different kinds enables an essentially higher work rate than human force and is at the same time the prerequisite for further automation. As the field machinery diversifies with rapid technological development, agricultural activities are usually achieved by a multi-machine system. The efficiency of the entire system depends on both the individual machine and the coordination among them.

Recently, the automation process in the agriculture sector is no longer limited to replacing human work with the operation of machines. Even the operation of the machinery is being automated through autonomous systems (Chapter 1.2.1). Compared to the manual operation, an autonomous system has several advantages. For starters, an autonomous system could achieve a higher operational accuracy. Another benefit of the autonomous systems is the extended work time. Unlike human beings, machines wouldn't get fatigued even through

[1] Data Source: Health Nutrition and Population Statistics: Population estimates and projections. Word Bank.

longer spells, thus operational breaks to maintain a high level of concentration are no longer necessary. For most autonomous systems, path planning is an important component.

On the other side, the impact of agricultural activities on the environment should not be neglected. In this regard, the application of field machinery could be seen as a two-edged sword. For example, the abundant use of herbicide has made some weed species resistant to it. A more sustainable solution that doesn't involve gene technology or harmful chemicals is high-speed mechanical weed control (Slaughter, Giles, & Downey, 2008). However, the traveling of large machines might cause soil compaction, which will reduce the yield (Chapter 1. 1. 3). Besides, the usually higher fuel consumption of field machinery will also cause more CO_2 emission. Therefore, it is both an environmental concern and an economical concern to optimize the in-field travel routes of agricultural machines.

Furthermore, various sensors mounted on the field machinery are providing tremendous data about the condition of the crops and the performance of the machine. The analysis of such data provides a huge potential for further optimization. Considering the emergence of various drones and field robotics, agricultural vehicles might not be the only beneficiary of a precise route planning system.

This thesis is organized as follows: The first chapter offers some background information related to field machinery and the automation of agricultural operations. The existing solutions to the described problem and the agricultural routing in general are introduced in Chapter 2. The concept and some implementation details of the proposed algorithm is the topic of Chapter 3. In Chapter 4, the performance of this algorithm is evaluated based on an in-depth comparison with the visibility graph algorithm.

1. 1 Field Machinery

The beginning of agricultural mechanization could be traced back to the late 19th and early 20th centuries, enabled by the development of the steam engine as a relatively portable power source. At that time, however, only wealthy agricultural producers could afford such costly machines. In the 1920s and 1930s, smaller tractors with internal combustion engines that ran on gasoline were massive produced and more farmers could afford them. Since then,

different attachments and machinery has been invented to automate tasks that were previously performed by hand (Kutz, 2013).

The introduction of field machinery like tractors and combine harvesters not only radically changed the nature of field operations towards more automation, but also constituted a new logistic chain that is different to the traditional one.

Depending on the direction of the material flow, the operations can be characterized as "input material flow" operations (e.g., seeding, spraying and fertilizing) or "output material flow" operations (e.g., harvesting). The field and the facility units, whether stationary (e.g. a farm depot or a silo) or mobile (e.g. public road transport trucks), constitute the two anti-diametric ending links of the material flow chain (D. D. Bochtis, Sørensen, & Vougioukas, 2010).

Using harvesting as an example, it is a common practice to use multiple harvesters, unloading vehicles and street transporters for large fields (Scheuren, Hertzberg, Stiene, Hartanto, & Center, 2013). The multi-machine coordination and the logistic chain could be depicted as follows: A harvester stores the crop in its tank until the maximum capacity is reached and waits for the unloading vehicle. During the transfer of crop from the harvester to the unloading vehicle, the two involved vehicles could both stay stationary or drive in parallel. Then the unloading vehicle approaches the next harvester that is in need of a service. This harvester might work on the same field or on another field nearby. When the unloading vehicle itself is fully loaded, it moves to the field border to unload the crop to a street transporter. The street transporter then uses the road system to transport the grain to the silo.

According to the terminology of D. D. Bochtis, Sørensen and Vougioukas (2010), the harvesters in this scenario that execute the main task are primary units (PU) and the unloading vehicles that support the primary units are service units (SU). In the following parts of this thesis, the notion of primary units and service units is adopted. In terms of trajectory patterns, these two kinds of machines differ: Primary units usually cover the whole field (see also Chapter 2.2.2) and service units usually move between the primary units and further servicing points. Thus, primary units and service units also require different methodology for path planning.

Besides the changes in logistic chain, the utilization of field machinery also brought in more challenges in terms of management (Sørensen & Bochtis, 2010), such as the choice of the optimal level of mechanization which involves the choice of machinery size and the constitution of the necessary fleet (Søgaard & Sørensen, 2004). This is also the topic of Chapter 1.1.2.

1. 1. 1 Guidance Systems for Field Machinery

The guidance system of a vehicle determines its moving course. The precision of the guidance system also determines the ability of a certain machine to follow a pre-defined path. Therefore, it is an important factor for the efficiency. According to the classification of Kutz (2013), there are four different strategies for guiding vehicles:

- manual – An operator steers the vehicle based on their observations of the surroundings.
- operator assisted – An operator steers the vehicle based on a signal from the guidance system .
- semi-autonomous – The guidance system steers the vehicle, but an operator is present to ensure the system is working properly and performs other vehicle functions that are not automated.
- fully autonomous – No operator required. Fully autonomous vehicles tend to use GPS as a primary source of location, and an array of secondary sensors for location relative to the crop and safety.

1. 1. 2 Fleet Management

The involvement of multiple machines in agricultural campaigns increases the need for better coordination and management, not only for the primary units, but also for the service units, since the performance of service units can also affect the productivity of the entire system significantly. For instance, a non-optimal path of a transport unit may cause a high-capacity combine to remain idle, while waiting to unload its fully loaded tank. Furthermore, assuming

the heavy loads carried by transport units, the traveled path may have a significant impact on soil compaction (Jensen, Bochtis, Sørensen, Blas, & Lykkegaard, 2012).

Agricultural fleet management mainly deals with farmers' or machine contractors' decision-making concerning resource allocation, scheduling, routing, and real-time monitoring of vehicles and materials. It involves the process of supervising the use and maintenance of machines and the associated administrative functions including the coordination and dissemination of tasks and related information for solving the heterogeneous scheduling and routing problems (Sørensen & Bochtis, 2010).

The emerging autonomous and intelligent agricultural machines will bring more challenge to fleet management since the food production, unlike other industrial activities, are more subject to the environmental factors such like weather conditions and crop yield. The traditional management system should be especially supplemented with planning features such as route planning and sequential task scheduling (Dionysis D. Bochtis, Sørensen, & Busato, 2014).

1. 1. 3 Impact of Large Machinery on the Soil

Large machinery benefits from the economy of scale, but also causes more negative environmental impacts: Unnecessary travels of agricultural machinery will not only cause soil compaction, but also more air pollution, as well as waste of fuel.

Soil compaction is one of the major problems facing modern agriculture. Besides the overuse of machinery, intensive cropping, short crop rotations, intensive grazing and inappropriate soil management could also lead to soil compaction. The result of soil compaction is additional fertilizer requirement and increasing production cost since the physical soil fertility decreases due to insufficient storage ability of water and nutrients (Hamza & Anderson, 2005). Besides, soil compaction also impedes infiltration of rainfall, so the increasing scale of mechanization might well be responsible for greater runoff, soil loss by water erosion and waterlogging (Tullberg, Yule, & McGarry, 2007).

In general, heavy machinery will cause two kinds of soil disturbance – vertical disturbance throughout the traversal and horizontal disturbance mainly through turning (D. D. Bochtis & Vougioukas, 2008). Especially the transport units will have more negative impact on the soil

because of the heavy load (D. D. Bochtis, Sørensen, & Vougioukas, 2010). Considering the current trend towards employing larger and larger machines (Dieter Kutzbach, 2000), the risk of soil compaction is increased even more (D. D. Bochtis, Sørensen, & Vougioukas, 2010).

To reduce the occurrence of soil compaction, several practical techniques could be considered, such as reducing pressure on soil either by decreasing axle load or increasing the contact area of wheels with the soil through improvement of tyre or track design (Hamza & Anderson, 2005).

Among the different possible solutions, two are highly relative to the path planning of field machinery. The first solution is to reduce the total travel distance and the number of turnings of field machinery, which could be achieved by route optimization. The other solution is to confine traffic to certain area on the field. This approach is also known as Controlled Traffic Farming, which will be introduced in detail in the next section.

1. 1. 4 Controlled Traffic Farming

The Controlled Traffic Farming concept was initiated in the USA around 1950 to increase crop yields by reducing soil compaction (Taylor, 1983). According to the Controlled Traffic Farming (CTF) principles, permanent parallel wheel tracks are created within the field area (D. D. Bochtis, Sørensen, Green, Moshou, & Olesen, 2010).

In such a system, the crop zone and the traffic lanes are distinctly and permanently separated. The traffic lanes are not deep tilled and are used for wheel paths year after year. The lanes become compacted, improving tractive efficiency and flotation, while the untrafficked crop zone, if initially well prepared, tends to stay that way without annual deep tillage (Taylor, 1983). In this way, the field traffic is limited to the traffic lanes and the headlands which is used to turning between lanes and the need for tillage, usually seen as the major cause for erosion and soil structural degradation (Tullberg et al., 2007), is also reduced.

In practice the introduction of Controlled Traffic Farming have three technical requirements:

- Traffic lanes with an equal span to each other should be established.
- The width between wheels of the vehicle should be in accordance with the width of the traffic lane.

- The vehicle should be able to follow the established traffic lanes through advanced guidance system, such as Real Time Kinematic Differential Global Positioning Systems (RTK-DGPS).

The first requirement can be achieved with specifically designed wide-span machines – gantries (Figure 1.2[2]), but most systems are presently based on modified conventional agricultural equipment (Vermeulen, Tullberg, & Chamen, 2010).

Figure 1. 2 A Picture for Gantry

The Controlled Traffic Farming concept can also be adopted partially. One example is the Seasonal Controlled Traffic Farming (SCTF) system where traffic lanes are not used for primary tillage or harvesting, but for all operations in between (Vermeulen et al., 2010).

The benefits of adopting Controlled Traffic Farming concept has been reported in drier regions such as Australia (Tullberg et al., 2007) and Northern China (Qingjie et al., 2009). In Western and Middle European some researches are underway to conclude if similar benefits could be achieved (Demmel, Brandhuber, Kirchmeier, Mueller, & Marx; Holpp et al., 2011).

It should also be mentioned that although Controlled Traffic Farming could reduce the impact of large machinery on the soil by limiting the damage to certain tracks, it increases the travel distance of the SU. The research of D. D. Bochtis, Sørensen, Green et al (2010) showed that the implementation of the CTF system rather than the UCTF system

[2] Dedousis, Athanasios P., and Thomas Bartzanas. Soil engineering. Vol. 20. Springer Science & Business Media, 2010. P102.

significantly increases the in-field transport distance travelled by the SU. The reduction in field efficiency (explained in Chapter 2.2) in terms of transport distance, ranged from 4.68% to 7.41%.

Therefore, the decision-making on the adoption of Controlled Traffic Farming system involves a trade-off between shorter travel distances and better soil structure conservation. Also, following the tracks precisely poses higher requirements to the guiding system. The optimal strategy might differ according to the characteristics of the involved machinery as well as the nature of the field operation.

1. 2 Automation of Field Operations

1. 2. 1 Autonomous Guiding System

Researches into autonomous vehicles in agriculture could be traced back to the early 1960s, mainly focused on developing automatic steering systems (Pedersen, Fountas, Have, & Blackmore, 2006). In the 1980s, the potential for combining computers with image sensors provided opportunities for machine vision based guidance systems. In 1997, agricultural automation had become a major issue along with the advocacy of precision agriculture (M. Li, Imou, Wakabayashi, & Yokoyama, 2009).

Such autonomous guiding system could further reduce the driver's work load and extend work time during twilight periods (Dieter Kutzbach, 2000).

An autonomous guidance system usually consists of sensors, computational methods, navigation planners and steering controllers (M. Li et al., 2009; John F. Reid, Zhang, Noguchi, & Dickson, 2000). Two major sensor components in an autonomous guidance system are GPS sensors, and machine vision sensors. GPS and machine vision fused together or one fused with another auxiliary technology is becoming the trend development for agricultural vehicle guidance systems (M. Li et al., 2009). Machine vision is a relative position and heading sensor with the image sensor mounted on the vehicle. Standard image sensors provide a color or monochrome response. Positioning of the sensor on the vehicle requires an understanding of the geometric relationship between the image sensor, the vehicle and the field-of-view that the sensor uses for guidance information. The processed images provide an

output signal that can be used to provide a steering signal for the vehicle (John F. Reid et al., 2000). But it should be mentioned that the machine vision must rely on some local references such as crop rows, tilled or untilled boundary and the like to work properly. Usually, a directrix needs to be extracted from these local references for the purpose of navigation.

Combining GPS and vision systems also has another advantage: The two systems have quite different failure modes. For example, GPS is subject to multi-path problems and occluded satellites, while the vision system tends to have trouble with poor lighting conditions and sparse crop. Combining the two systems will provide a significant increase in overall robustness of the harvesting operation (Pilarski et al., 2002).

To improve the navigation accuracy, other auxiliary sensors such as geomagnetic direction sensors and inertial sensors could also be used. A geomagnetic direction sensor (GDS) is a magnetometer which senses the earth's magnetic field. It can serve as heading sensor similar to an electronic compass. A limitation of GDS sensors is the influence of external electromagnetic interference from the vehicle and surrounding sources (John F. Reid et al., 2000). Inertial sensors take measurements of the moving state of the vehicle. The most common types of inertial sensors are accelerometers and gyroscopes.

Although path planner is an important component in an autonomous system, the research focus in this context is usually the extraction of directrix from local environment and the precise following of this directrix (Han, Zhang, Ni, & Reid, 2004; John F Reid & Searcy, 1991). The calculation of a globally optimized path, especially for service units whose driving paths are more related to the target machine rather than the crop structure, is rarely mentioned in the literature of this category.

1. 2. 2 Field Robotics

Parallel to the improvement of large machinery, much smaller field robotics that can handle the plants at an individual level is also turning up. Large machinery might be well suitable on open fields mainly for grain cultivation, but they are too cumbersome to be applied in greenhouse or horticulture applications (Suprem, Mahalik, & Kim, 2013). Thus, robots of smaller size but with more flexibility are required. Especially for high-value crops such as vegetables, fruits, ornaments and spices which still require high labor input, the introduction

of such field robots has a great potential to provide a leap forward in terms of cost reduction and efficiency (Bac, Henten, Hemming, & Edan, 2014).

However, the work environment of field robots is quite different to that of their industrial or laboratory counterparts. For example, the operating areas could be spacious; ground surfaces may be uneven, so that wheel slippage may be not negligible any more. Environmental conditions (rain, fog, dust, etc.) may affect sensor function (M. Li et al., 2009). Furthermore, the crops, which are the target objects of operations like fertilizing and harvesting, are subject to seasonal variations.

Different crops also show extreme variations in the physical form and technical requirements during different operations. As a result of these variations, the development of a general architecture that is suitable for every crop genre is difficult.

Although still not mature enough for a large-scale commercialization, some researchers have shown that for certain tasks, robots can outperform the conventional systems in terms of economic costs, for example, in cases of weeding in high value crops or organic farming, harvesting of high value crops, crop scouting in cereals and grass cutting on golf courses (Bac et al., 2014; Bakker, Asselt van, Bontsema, Müller, & Straten van, 2010; Grift, Zhang, Kondo, & Ting, 2008; Pedersen et al., 2006).

The field robots developed so far have different movement and navigation strategies. For the field robots that are not fixed to a certain position (Belforte, Deboli, Gay, Piccarolo, & Ricauda Aimonino, 2006) and those without a trail fixed on the ground, a path-planning algorithm that is able to avoid at least stationary obstacles is necessary. González, Rodríguez, Sánchez-Hermosilla and Donaire (2009) described such an algorithm based on the Voronoi Diagram and Depth First Search.

Another direction of robot navigation leads towards real-time obstacle detection based on machine vision technology. Since the real-time obstacle strategy usually ends up with a sub-optimal path, a system that could calculate a globally optimal path that avoids the stationary obstacles a priori, and then adjust the route to avoid dynamic obstacles will achieve a better performance.

Chapter 2 State of the Art

2. 1 The Euclidean Shortest Path Problem

Besides in-field navigation, the problem of finding shortest paths on surfaces with the presence of boundaries and obstacles also comes up in many other real-life scenarios, among these scenarios are pedestrian navigation, indoor navigation, motion planning of robots and avatar movements in computer games, to name a few. In computational geometry, this problem is usually referred to as the Euclidean Shortest Path (ESP) problem (Hong & Murray, 2013; F. Li & Klette, 2011). The open problem related to ESP is whether it can be solved within O(n + h * logh) time and O(n) space (n is the total number of vertices and h is the total number of obstacles) (Mitchell, 2000).

The difficulty of this problem is the absence of a graph, also known as roadmap in the literature. Once a graph is generated somehow, the problem can be solved easily by applying the well-known algorithms for finding the shortest path on a graph, such as Dijkstra algorithm, A* algorithm, Bellman-Ford algorithm etc. (Fagerholt, Heimdal, & Loktu, 2000).

Therefore, the general strategy to solve problems in this category is to construct a graph first. The following sections will compare four different approaches for the graph generation and discuss their suitability for the application in in-field routing.

Here we assume that some preprocessing has been done so that the moving object can be simplified as a point with zero size and its physical limitations (turning radius, width) are neglectable. One possible preprocessing is to expand the obstacles appropriately, which is widely used for robot motion planning (Berg, 2008; Latombe, 1991).

We also assume that the environment is static, which means that no other dynamic obstacles will intercept the path of the moving object.

2. 1. 1 Grid-based Approach

The first solution is based on the rasterization of the environment. The general idea of this approach is to divide the environment into equal-sized cells. This grid-based representation of the whole environment may be either binary (each grid cell is either traversable or not) or may associate each cell a cost reflecting the difficulty of traversing this cell (Ferguson & Stentz, 2006; Linker & Blass, 2008).

Each traversable grid corresponds to a node on the routing graph. Edges are added between each pair of nodes that are directly reachable. Two grid cells are directly reachable from each other if and only if they are adjacent. For each grid cell, there are either eight adjacent cells or four. The definition of neighborhood depends on the degree of freedom of the moving object.

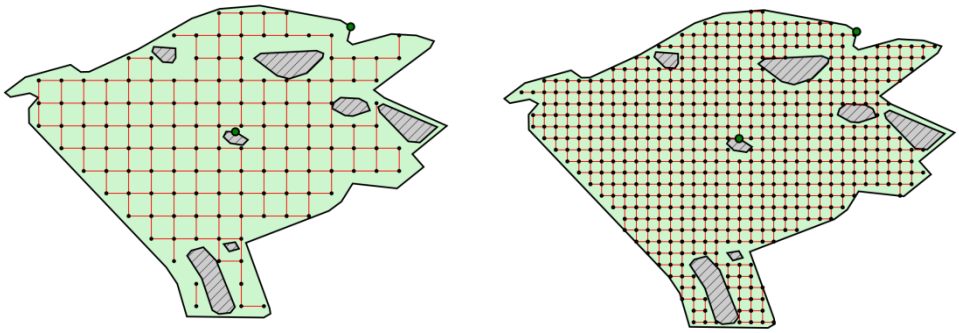


Figure 2. 1 Graph with Different Grid Sizes

With this approach, the final goal position accuracy as well as the path quality is dependent on the granularity of the grid division. Finding an optimal grid size usually involves a trade-off: Smaller grid sizes can improve the accuracy of both the goal position and the path, but only at the expense of larger routing graphs and more unnecessary turnings. Larger grid sizes can reduce the complexity of the graph, but may lead to disconnected graphs and eventually fail to find a path from the source to the destination. Figure 2.1 shows an example of the

relationship between grid size and the connectivity as well as the complexity of the generated graph. While the graph generated with larger grid size (left) has less nodes and edges than the one generated with smaller grid size (right), it is not connected, if the start and end positions are located on different parts, there would be no path found between them.

In general, this approach applies well to moving objects which can only move in four or eight directions, like avatars in old-time computer games, since it limits heading and heading changes to 45° or 90°. For omnidirectional moving objects, however, this approach is usually sub-optimal since it introduces unnecessary turns (Ferguson & Stentz, 2006).

Another advantage of this approach is that it can use Breadth First Search (BFS) to find the shortest path on the generated graph instead of Dijkstra algorithm, due to the identical length of each edge. Also, the running time of BFS is linear to the number of edges, which is faster than the Dijkstra algorithm.

2. 1. 2 Triangulation-based Approach

This category of algorithms involves the intial triangulation of the polygon. Its general idea is similar to the grid-based approach. Instead of grid cells, the passable area is represented by non-overlapping triangles. A triangulation of a polygon P is a decomposition of P into triangles by a maximal set of non-crossing diagonals (Devadoss & O'Rourke, 2011). The triangulation can be achieved by the well-accepted Delaunay triangulation algorithm (Berg, 2008), which adds the diagonals in such a way that the minimum interior angle of all triangles is maximized.

If finding a path fast is more important than finding the globally shortest path, the route graph could be identical to the dual graph of the planar graph that results from the triangulation step: Each triangle is represented by a node on the routing graph. Adjacent triangles are linked with an edge. Figure 2.2 shows the workflow of this algorithm with a concrete example.

Although the triangulation of a give polygon is not unique, the triangulation theorem guarantees that every triangulation of a simple polygon (without holes) with n vertices has n-2 triangles (Devadoss & O'Rourke, 2011), for a polygon with h holes and in total n vertices, the number of triangles is n+2h-2 (O'rourke, 1987). Therefore, the complexity of the generated

graph is determined by the number of vertices and holes of the given polygon. Similar to the grid-based approach, this variant may as well introduce many unnecessary turns.

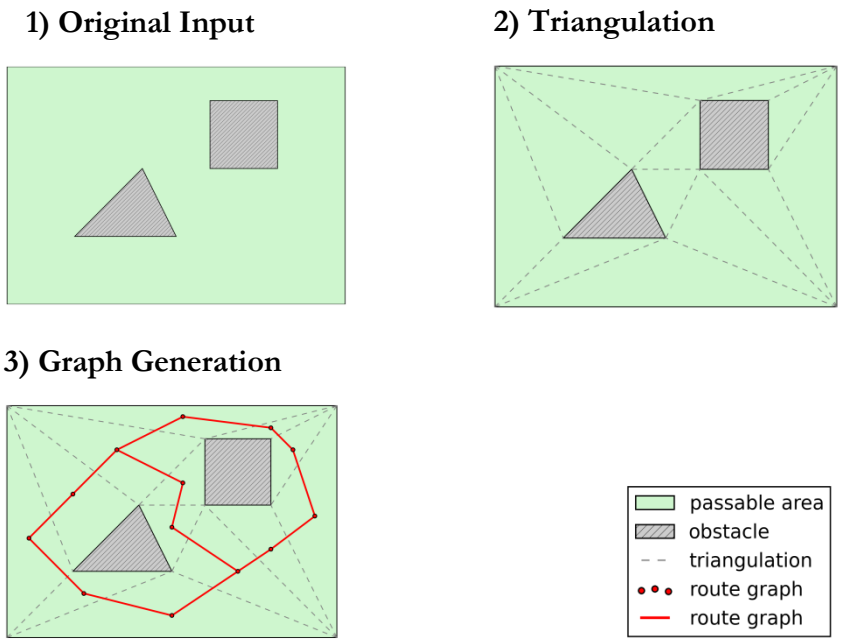


Figure 2. 2 The Triangulation-based Approach in Steps

If finding the shortest path is more important than finding a path quickly, the dual graph can not be used for routing directly. Kapoor, Maheshwari and Mitchell (1997) described an algorithm for finding the shortest path based on triangulation: Firstly, the start position and end position are incorporated into the triangulation through linking them to the three corners of the triangles that contain them respectively. Secondly, the triangles are divided into junction triangles (triangles that are formed with three diagonals) and corridors (triangles that serve as channels between the junction triangles), each corridor has two doors. If the shortest path passes through a corridor, it must intersect both two doors of this corridor. Then, the hourglass of each corridor is generated. The hourglass could be imagined as two non-crossing taut strings, originating from one door and ends at the other door. The hourglass can be either open or closed. Closed hourglasses usually have two funnels and a corridor path. The route graph is a subgraph of the visibility graph generated from the vertices on junction triangles, open hourglasses and funnels.

Either way, the time complexity of triangulation-based algorithms is usually asymptotically dominated by the triangulation process, which has a time complexity of O(n*logn), with n representing the number of vertices in total.

2. 1. 3 Visibility Graph

The visibility graph algorithm is a canonical solution to the shortest path problem in discussion. It was proposed by Lozano-Pérez and Wesley in 1979. The theoretical cornerstone of this algorithm is the following observation:

> In the case of motion in the plane with arbitrary polygonal objects, the shortest collision-free path connecting any two accessible points always has the property that it is composed of straight lines joining the origin to the destination via a possibly empty sequences of vertices of obstacles (Lozano-Pérez & Wesley, 1979).

Viegas and Hansen (1985) also proved that, when distances are measured by an lp–norm with $1 < p < \infty$, the shortest paths between given sets of points in the plane that do not cross any of a finite set of polygonal barriers are always formed by sequences of connected straight line segments whose intermediate (e.g. apart from origin and destination) end points are barrier vertices.

Since a visibility graph is constructed by connecting each pair of mutually visible vertices in the environment plus the source and destination position, the shortest path can only be a sequence of the edges on this graph. Figure 2.3 shows a concrete example of the visibility graph derived from a polygon with holes.

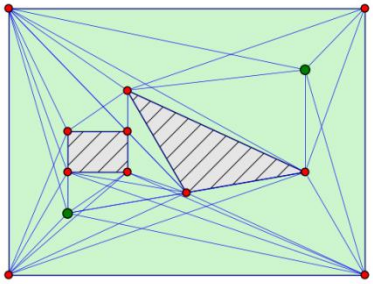


Figure 2. 3 An Example for Visibility Graph

The brute-force implementation of visibility graph generation usually takes $O(n^3)$ computation time (Huang & Chung, 2004), since it iterates through all pairs of nodes and checks if the line segment formed by this pair of vertices crosses with any obstacle. Although there are improved implementations which reduce the running time complexity to $O(n^2)$ (Gajentaan & Overmars, 1995; Welzl, 1985), a quadratic run-time is still very sensitive to the input size.

Another variant of the visibility graph is the local visibility graph, which is generated in a similar fashion as the visibility graph, but attempts to filter obstacles using different spatial knowledge. However, the local visibility graph usually fails to handle the intersection with regional boundaries and sometimes could not even find the optimal path (Hong & Murray, 2013).

2. 1. 4 Convex Hull Approach

The convex hull problem is a classical topic in computational geometry. It is defined as the smallest convex set that contains a set of given points (Berg, 2008).

But leveraging convex hull generation for finding shortest path it not yet widely practiced. Hong and Murray implemented a similar algorithm within a GIS system in 2013, but there are several minor flaws in their implementation. Firstly, it is implemented with GIS software, which poses limitations on the input and output format as well as the integration with other services. Secondly, the concavity of obstacles is not well discussed.

The underlying rationale of solving shortest path problems through recursive convex hull generation is the following observation:

> The optimal Euclidean shortest path between two points separated by a single convex contiguous obstacle will be on the convex hull boundary (Hong & Murray, 2013).

But the plot thickens if there are multiple obstacles. First of all, the beeline connecting the start position and the end position might cross with multiple obstacles instead of a single obstacle. In this case, we should generate the convex hull of each impeding obstacle plus the endpoints individually. Secondly, the edges on the generated convex hulls might further cross with other obstacles. According to the terminology coined by (Hong & Murray, 2013), these

obstacles are defined as indirect impeding obstacles, as opposed to the direct impeding obstacles. The convex hull routine should be applied recursively until all the edges on the graph are collision free.

The possible concavity of both the boundaries and the obstacles adds another layer of complexity to this algorithm. During the replacement of a colliding edge with a convex hull, an invariant should be maintained: The start position and end position should always be connected in the graph. The solutions to these edge cases will be discussed in more detail in Chapter 3.

Despite the intrinsic complexity of the edge cases, the convex hull approach shows several advantages compared to the other three approaches. Firstly, it takes only the direct and indirect obstacles into consideration which reduces the size of the generated graph. All the other three alternatives take the whole environment into consideration. Secondly, it will not introduce additional unnecessary turns as the grid-based approach and the triangulation-based approach usually do.

However, the reduction of graph size comes at an expense. Unlike the other three

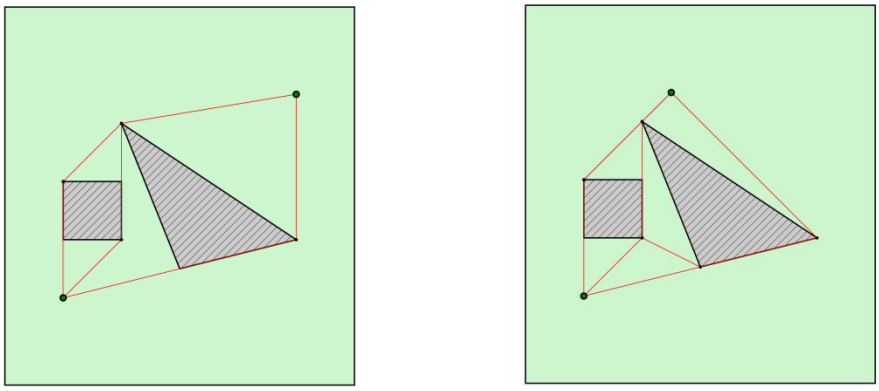


Figure 2. 4 Comparison of Different Start and End Positions

approaches, the graph generated using the convex hull routine is highly dependent on the position of the source and destination. As Figure 2.4 shows, even a slight shifting of the source or the destination might require the entire re-generation of the graph. In contrast, the graph generated by the other three approaches is global and independent of the start and end

position. Therefore, the graph can be cached for repetitive requests in the same environment configuration.

2. 1. 5 A Comparison Regarding In-field Navigation

To compare the four algorithms for graph generation in terms of their suitability for in-field navigation, the following aspects should be considered:

- running time (time complexity)

Since in-field navigation needs to cope with real-time requests that might come at a high frequency, during harvest champagnes, for example, the running time is critical. The total running time is added up by the running time of graph generation and the running time for finding the shorted path on the graph, including the assignment of weight to each edge on the graph.

- graph size

The graph size can be measured in terms of the number of nodes or edges. It matters for several reasons. Firstly, it is directly related to the running time for the finding the shortest path on the graph. Secondly, larger graphs require more memory usage. Furthermore, if the in-field navigation is to be integrated with other services such as inter-field navigation, the generated graph need to be passed on to other servers through the network. Transporting large amount of data through the network can easily become the bottleneck of the entire system.

- existence and length of the calculated shortest path

The connectivity of the generated graph is a prerequisite for the existence of the shortest path. The length of the shortest path is directly related to the travel time, energy consumption, as well as the impact on the soil by the field machinery.

- number of additional turns

Due to their usually over-average physical size, turning is more difficult for agricultural vehicles as it is for normal vehicles, especially when the agricultural vehicle has any

attachment at its front or back (e.g. combine harvester). Besides, as mentioned in Chapter 1, a turning vehicle will also incur more negative impact on the soil (additional horizontal disturbance of soil) as one going straight.

- possibility for caching the graph

If the generated graph is not dependent on the start and end position, it can be generated once and cached for repeated requests on the same field.

The table below evaluates the four algorithms according to the criteria discussed above:

Table 2. 1 Comparison of Graph Generation Algorithms

	Grid	**Triangulation**	**Visibility Graph**	**Convex Hull**
Running Time for Graph Generation	dependent on the grid size	O(n*logn) for triangulation(Berg, 2008) O(n) for graph generation on the triangulated polygon(Hershberger, 1989)	$O(n^2)$	O(n) for a single convex hull generation, but the total number of recursion is dependent on the input data
Existence of a Shortest Path	not guaranteed	guaranteed	guaranteed	guaranteed
Graph Size	dependent on the grid size	O(n) in general	$O(n^2)$	dependent on the input data
Additional Turns	dependent on the grid size (smaller grid size will introduce more additional turnings)	dependent on the number of vertices of the given polygon	no additional turns introduced	no additional turns introduced
Caching	possible	possible	possible	not possible

In terms of reducing additional turns, both the grid-based and the triangulation-based approach are suboptimal for in-field navigation. For the grid-based approach, the space between two obstacles on the field might be narrow. Correspondingly, to guarantee the existence of the shortest path, the grid size also needs to be small, which will increase the number of additional turns. For the triangulation-based approach, the number of nodes of the generated graph is identical to the number of triangles, which is linear to the number of vertices and holes. In some regions, such as Denmark or Mecklenburg-Vorpommern in Germany (Figure 2.5[3]), field geometries are sometimes complex and, as a consequence, involve many vertices.

Figure 2. 5 Complex Field Geometries in Mecklenburg-Vorpommern

A large amount of vertices also affects the performance of the visibility graph, which has a time complexity that grows at least quadratically with the input size. In such cases, the convex hull approach has an edge over the other three alternatives in terms of the graph size. The cost is that the graph needs to be generated in every new request. But as the test results in Chapter 4 show, even under the assumption of caching, the convex hull approach still outperforms the visibility graph algorithm.

Besides the algorithms that generate the graph first, there are also other approaches that find the shortest path without graph generation. One of them is the continuous Dijkstra which involves simulating the effect of a "wavefront" propagating out from the source point. An

[3] Screenshots from http://teleagroplus.geog.uni-heidelberg.de/taplus_webgui/

optimal implementation of this algorithm is described in the paper of Hershberger and Suri (1999).

2. 2 Route Planning in Agriculture Domain

The current research about route planning for agricultural vehicles generally falls into two categories. The first category is dedicated to improving the efficiency of field logistics. As mentioned in the first chapter, agricultural tasks are usually carried out by a fleet of machines, and the overall efficiency is dependent on both the efficiency of each single unit and their coordination. The second category is about field coverage planning which aims to find an optimal route, mainly for the PU, which traverses the entire area of a certain field.

The following two sections will give an overview of these two categories respectively.

2. 2. 1 Field Logistics Optimization

Logistics in general are defined as the provision of goods and services from a supply point to various demand points. Logistics within agriculture may be viewed as the material flow in the production process. This material flow usually has fields on one end and storage sites or the public transportation system on the other end, linked by multiple machines. Thus, a great part of field logistics optimization is about improving the inter-machinery coordination. The basic logistical notion is that all operations and actions must create an added value in the process chain through the process of having the right thing, at the right place, at the right time (Sørensen & Bochtis, 2010).

An important measure of agricultural machine performance during field operation is its field efficiency. In the literature it is defined differently. Hunt (2008) defined it as a percentage indicating the ratio of the time a machine is effectively operating to the total time that the machine is committed to the field operation, D. D. Bochtis and Vougioukas (2008) defined it as the ratio between the productivity of the machine under field conditions and the theoretical maximum productivity. The author prefers the first definition and is of the opinion that the second definition is probably a confusion with operational efficiency, which expresses the ratio between the actual in-field productivity and the maximum theoretical

productivity defined by the maximum speed and maximum working width (Witney, 1988). Either way, it reflects how well the machines are utilized for agricultural operations.

There are multiple factors that can impair field efficiency, such as the turning of the machine near field edges and obstacles as well as loading and offloading of agricultural goods (Spekken & de Bruin, 2013), which is often referred to as *servicing* in literature. Besides, field efficiency is also dependent on field size (Witney, 1988) and crop conditions like yield and moisture (D. D. Bochtis & Vougioukas, 2008).

Especially the first two culprits for field efficiency reduction are highly-related to path planning. Since most agricultural tasks are achieved by a multi-machinery system, the optimization of the overall system relies on a synchronous motion planning for both PUs and SUs (D. D. Bochtis, Sørensen, & Vougioukas, 2010). Scheuren et al. (2013) presented an approach for in-field path planning for autonomous unloading vehicles that considered the unharvested area as a dynamic obstacle and additional constraints like the harvester's kinematics, dynamics and its unloading direction.

Due to the intrinsic complexity of such overall optimization, most research in this area is still prototypical and conceptual. In most real life cases, crop harvesting operations are still planed based on the experience of the farmers operating the vehicles. Delays are common due to bad cooperation between the combines and the tractors, decreasing the overall duration (Ali, Verlinden, & Van Oudheusden, 2009).

2. 2. 2 Field Coverage Planning

The goal of coverage planning is to find a continuous non-overlapping route that covers a certain area with potential obstacles. It has many other application areas besides field operations, such as floor cleaning and underwater searching (Timo Oksanen & Visala, 2009).

The coverage path planning problem for agricultural field operations has been discussed in many publications: Zandonadi (2012) developed and evaluated a routing algorithm that took not only field efficiency but also off-target application areas into consideration, because these areas, especially in oddly shaped agricultural fields, might be as important as field efficiency when it comes down to the total operation cost. D. D. Bochtis and Vougioukas (2008) expressed the field coverage problem as the traversal of a weighted graph and related the

problem of finding an optimum traversing sequence with the problem of finding the shortest route in the graph . Oksanen and Visla (2007) presented two algorithms for coverage planning for agricultural fields (T Oksanen & Visala, 2007): one is off-line and uses a top-down approach to split complex-shaped fields into simple ones, another is on-line and uses a bottom-up approach to cover the field using prediction and exhaustive search methods. Since coverage planning is not the focus of this thesis, the details about these algorithms are not introduced here.

Chapter 3 Concepts and Implementation

3. 1 The Overarching Concept

The algorithm is consisted of two major steps. In the first step, a graph with non-colliding edges is generated based on the recursive application of the convex hull algorithm. In the second step, each edge in the graph is associated with a weight and the shortest path is calculated with the Dijkstra algorithm.

The following two sections in this chapter will describe the two steps respectively.

3. 2 Generation of the Routing Graph

3. 2. 1 Pseudocode

First of all, we give a mathematical formulation for the graph generation: Given a polygon P with potential holes, let V_P represent the set of all the vertices on P, either on the boundary or on the holes. Also, S and T are the start position and the end position respectively. Both S and T are either on the boundary of P or interior to P. The goal is to derive a connected graph $G = (V_G, E_G)$, so that $S, T \in V_G$, $\{V_G \backslash S, T\} \subseteq V_P$ and for $\forall e \in E_G$, e does not cross with P.

In simplicity, the algorithm for graph generation could be described with the following pseudocode:

Algorithm 1 GenerateGraph(Point* S, Point* T, Polygon* P):

```
1:  intialize a new graph G
2:  add edge (S, T) to G
3:  AdjustGraph(G, P)   // helper routine to adjust the graph
4:  return G
```

Algorithm 2 AdjustGraph(Graph* G, Polygon* P):

```
1:  for each edge in G:
2:      for each obstacle in P:
3:          if edge crosses with obstacle:
4:              generate the convex hull of {edge, obstacle}
5:              convert the convex hull to graph G'
6:              AdjustGraph(G', P) // adjust the graph recursively
7:              remove this edge from G
8:              add G' to G
9:          end if
10:     end for
11: end for
12: return
```

The pseudocode above applies well to the normal cases with only convex boundaries and convex obstacles. But we have to pay attention to the following edge cases:

If the boundary is concave, the beeline between source and destination might cross with the outer boundary. In this case, generating the convex hull will not work since both the source and destination are in the interior of the outer boundary (Figure 3.1 b). To cope with this kind of concavity, the outer boundary should be replaced by its convex hull, and the extra areas (pockets) introduced during the transformation should be added as new obstacles. These new obstacles should also be kept record of since they require a different treatment later. This will be explained in more detail in Chapter 3.2.3.

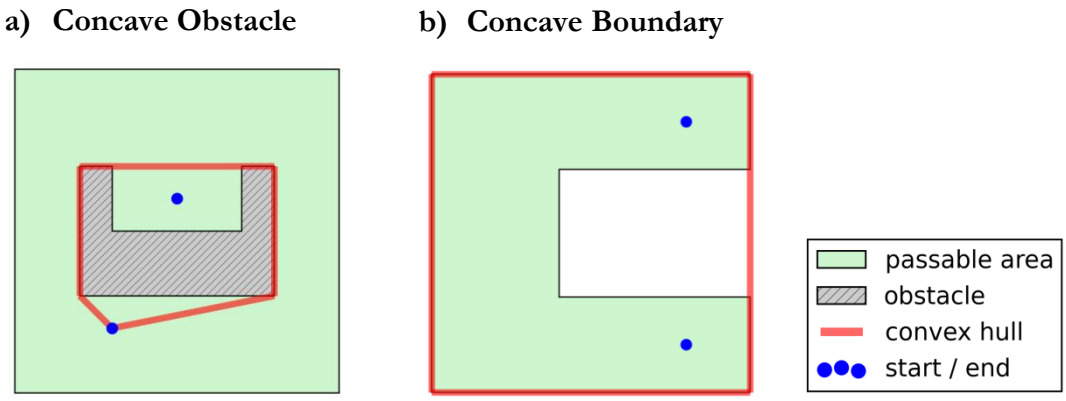


Figure 3. 1 Edge Cases with Concave Boundary or Concave Obstacle

For a certain edge that crosses with one or more obstacles, if the impeding obstacle is concave and at least one of the two extreme points of the edge is located within the convex hull of this obstacle, the generated convex hull would also fail to link the source and destination position (Figure 3.1 a). In this situation, the convex hull needs to be transformed to include both the endpoints. This is the topic of Chapter 3.2.4.

3. 2. 2 Normal Case with Convex Boundary and Convex Obstacles

If the outer boundary is convex, the edges of the graph will not cross with the outer boundary, since a convex polygon has the property that it covers all the line segments that link two of its interior points (Berg, 2008). If an edge crosses with multiple obstacles, the convex hull should be generated for each impeding obstacle individually.

Figure 3.2 is a concrete example of this category illustrated in steps:

a) initial input

b) generate the beeline and add it to the initialized graph

c) generate the convex hull for the impeding obstacle

d) adjust each edge on the graph recursively until it is collision-free.

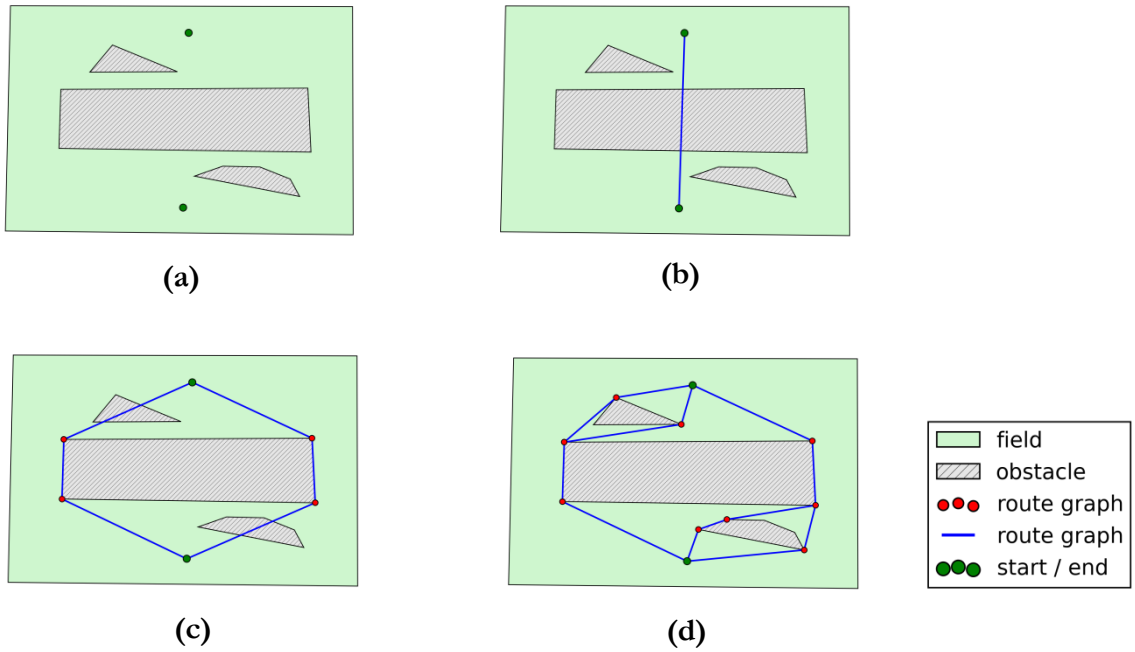


Figure 3. 2 An Example for Normal Case

3. 2. 3 The Edge Case with Concave Boundaries

To explain the edge cases better, some nomenclature should be introduced first:

auxiliary obstacle: one continuous area of the geometry difference between a concave polygon and its convex hull. In geometry, this area is also known as *pocket* (Toth, O'Rourke, & Goodman, 2004). One concave polygon could have several auxiliary obstacles.

auxiliary edge: the common edge that is shared by an auxiliary obstacle and the convex hull.

Figure 3.3 gives a visual explanation for the auxiliary obstacle and the auxiliary edge.

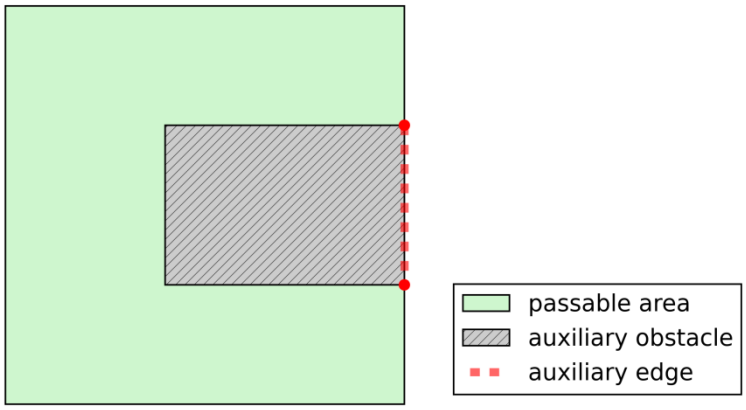


Figure 3. 3 Auxiliary Obstacle and Auxiliary Edge

As mentioned before, if the outer boundary is not convex, it could cross with the edges on the graph. In this case, the outer boundary should be transformed to a convex one in advance, and the auxiliary obstacles introduced through this process should be regarded as new obstacles. After this transformation, an edge case with concave boundary boils down to the normal case.

However, the newly introduced auxiliary obstacles require some additional logic when deciding whether they cross with a certain edge. Besides, the convex hull that stems from an auxiliary obstacle needs further processing as well to exclude the edges that are actually located outside the original outer boundary. Therefore, we should label the auxiliary obstacles differently so that we can tell them apart from the original ones.

For original obstacles, we say an edge crosses with an obstacle if and only if they have some internal points in common. For an edge, which is a line segment geometrically, the internal points are defined as all the points on it except for the two endpoints. For an obstacle, which is a polygon geometrically, the internal points are all the points inside the boundary, points on the boundary are not included. In other words, if an edge lies on the boundary of the

obstacle, they are not treated as crossing (Figure 3.4 left), as a result, the obstacle is not an impeding obstacle.

But for an auxiliary obstacle that is introduced through the transformation of concave boundaries, the auxiliary edge that it shares with the transformed boundary doesn't exist on the original boundary. If this auxiliary edge has some common points with the edge that we want to test, the obstacle and the edge are treated as crossing, although they don't have any internal point in common according to our previous criterion (Compare Figure 3.4 and 3.5).

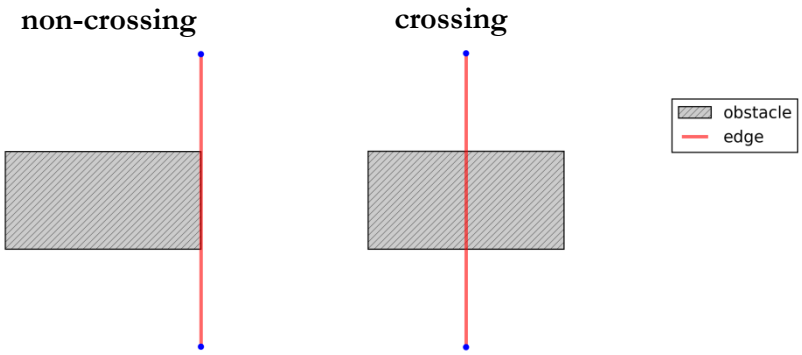


Figure 3. 4 Crossing for Original Obstacles

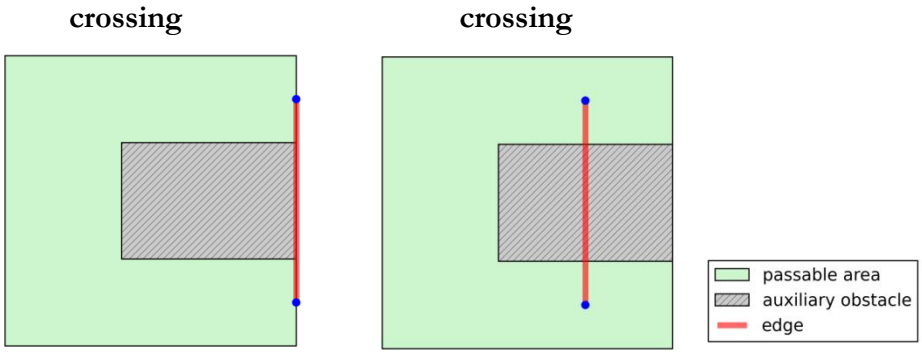


Figure 3. 5 Crossing for Auxiliary Obstacles

By the same token, the generated convex hull also requires further processing to exclude the path that is outside the original boundary. For an original obstacle, if it crosses with a certain edge, we generate the convex hull of the geometry collection which includes both the obstacle and the crossing edge. Regarded as a graph, the resulting convex hull is a cycle that

includes two valid paths from one endpoint to the other endpoint of the edge (Figure 3.7). For an auxiliary obstacle, however, we should exclude the path that passes through the auxiliary edge since part of it is outside the original boundary. For instance, in Figure 3.6, the blue path should be excluded because it is not valid on the original boundary, while both the red path and the blue path are valid for the original obstacle in Figure 3.7.

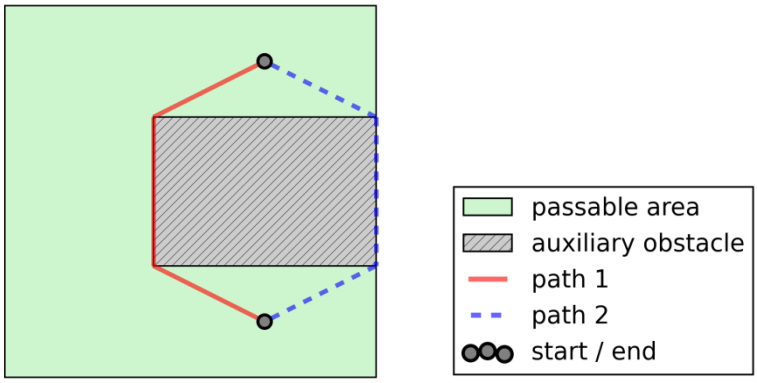


Figure 3. 6 Path Exclusion for Auxiliary Obstacle

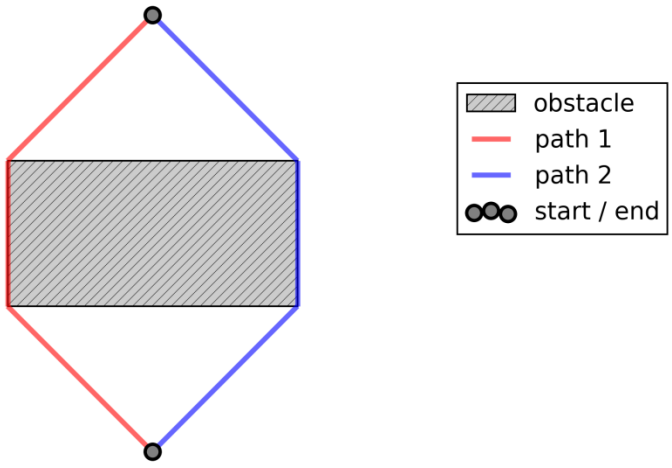


Figure 3. 7 Two Paths on the Convex Hull

Also, Figure 3.8 shows an example with concave boundary step-by-step:

a) initial input

b) transform the outer boundary and generate the beeline from the start position to the end position

c) generate the convex hull for an impeding auxiliary obstacle

d) exclude the path on the convex hull that is not traversable with the original boundary

e) adjust the edges on the graph recursively as in the normal case

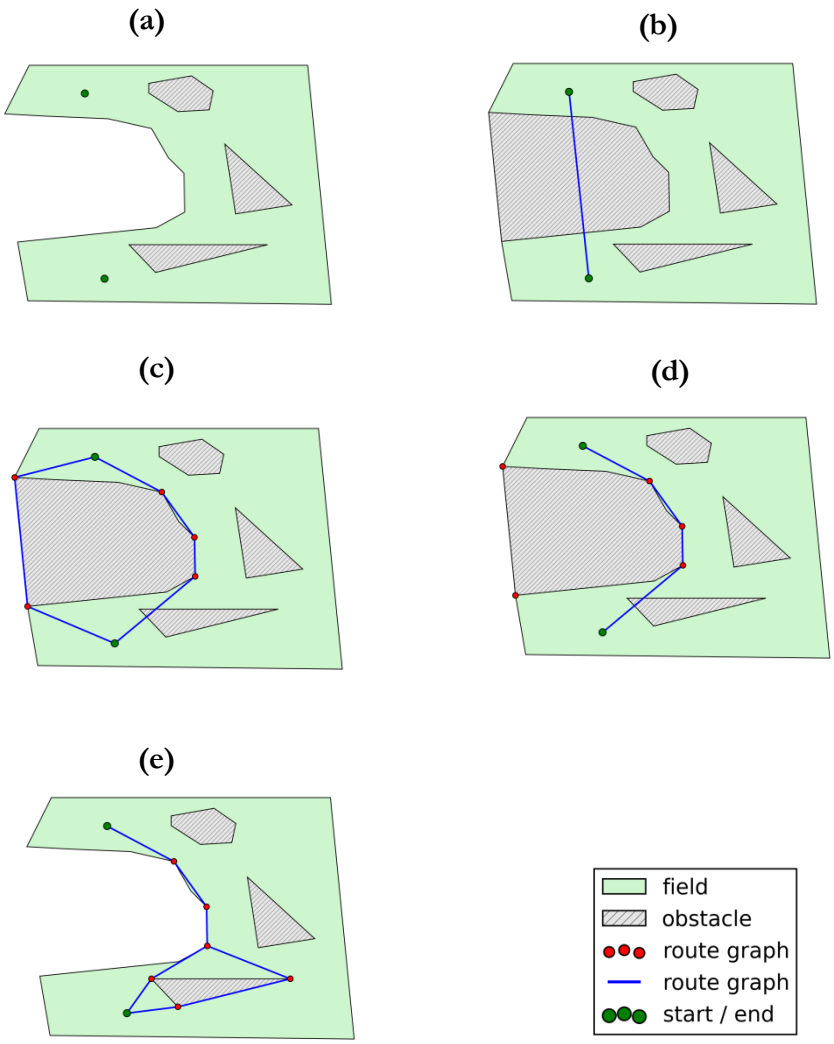


Figure 3. 8 An Example with Concave Boundary

3. 2. 4 The Edge Case with Concave Obstacles

If some of the obstacles are concave, it is possible that the start position or the end position, or even both are located in the "pockets" of the obstacles and thus interior to the convex hull. In such cases, the convex hull needs to be transformed to guarantee that it passes through both the start and end position, so that we could maintain the invariant that at each time during the transformation, the start position and end position stay connected, either by non-crossing edges or crossing edges.

According to the topological relationship between the start position, the end position and the concave obstacle, this edge case can be further divided into four different categories which would be discussed individually later.

The general idea to cope with the concavity of an obstacle is to transform the original problem to smaller sub problems: Firstly, we generate the convex hull of the obstacle and the crossing edge as usual. Secondly, we calculate the geometry difference between the convex hull and the original obstacle. The result could be either a simple polygon or a multi-polygon. For future reference, we call these polygons *indented regions*.

Although the indented region shares some similarities (both of them are derived from the geometry difference of a convex hull and a polygon) with the auxiliary obstacle defined in Chapter 3.2.3, they are in essence different. For an auxiliary obstacle, the convex hull is the convex hull of the outer boundary per se. For an indented region, however, the convex hull is the one of the obstacle together with the two endpoints, although it is identical to the convex hull of merely the obstacle if both the endpoints are located in the pockets of this obstacle.

The four categories described before could be formulated as follows and are also illustrated in Figure 3.9:

(1) One endpoint (either start or end position) is on the boundary of the convex hull, the other is not and they are not in the same indented region.

(2) One endpoint is on the boundary of the boundary of the convex hull, the other is not, but it is on the boundary or interior to the same indented region.

(3) The two endpoints are on the boundary or interior to two different indented regions. But neither of them is on the boundary of the convex hull.

(4) The two endpoints are on the boundary or interior to the same indented region. Neither of them is on the boundary of the convex hull.

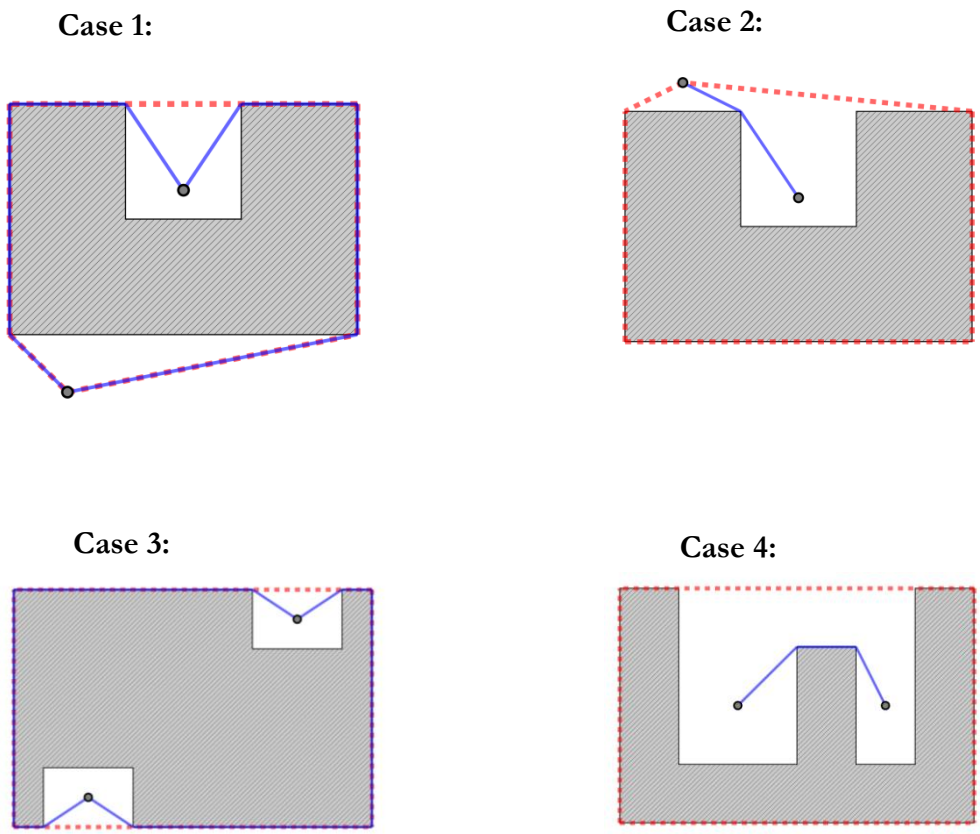


Figure 3. 9 Four Categories for Edge Cases with Concave Obstacles

For the second and fourth case, we need to solve the sub problem with the same endpoints and the boundary of the indented region as the new passable area.

For the first case, we have to link the endpoint that is not on the boundary of the convex hull to the convex hull. The key to achieve this linking is to solve two sub problems: this endpoint stays the start position, the end position is each of the two endpoints of the common edge that is shared by the indented region which includes the endpoint and the convex hull generated, the new passable area in the sub problem is the boundary of this indented region.

The forth case could be solved in a similar way as the first case. The only difference is that we have to link both of the two endpoints to the convex hull through solving four sub problems instead of two.

Figure 3.10 shows a concrete example with a concave obstacle which falls into the third category:

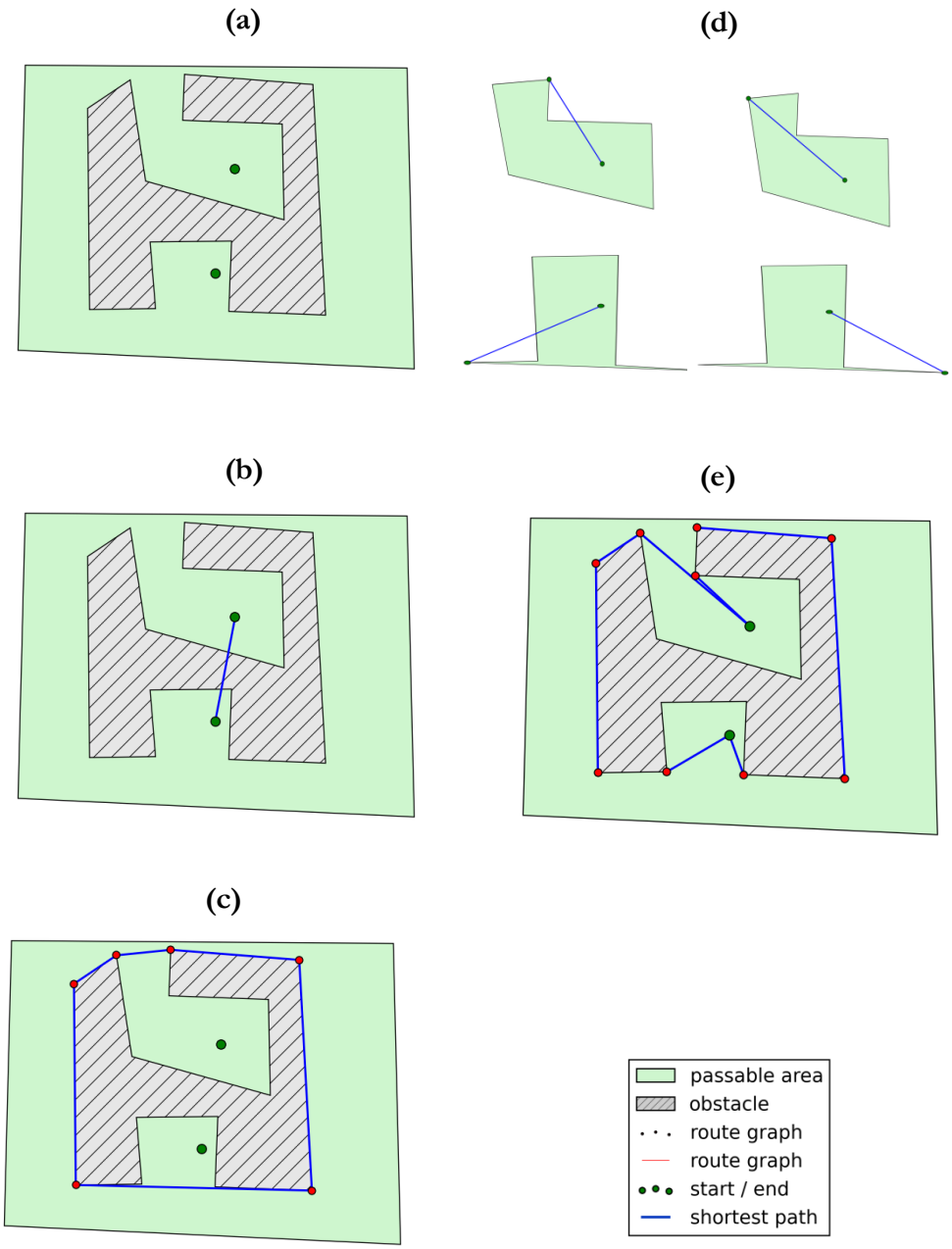


Figure 3. 10 An Example with Concave Obstacles

a) initial input

b) generate the beeline from the start position to end position

c) generate the convex hull as in the normal case

d) link the start position and end position to the convex hull through solving four sub problems

e) transform the convex hull to include the start position and the end position

3. 3 Derivation of the Shortest Path

Compared to the graph generation, the derivation of the shortest path is trivial. At first, each edge on the generated graph is assigned a weight according to a certain distance metric. For geographical coordinates, we use the great circle distance, which could be calculated with the haversine formula. For points defined in a Cartesian coordinate system, the Euclidean distance could be used as the distance metric.

After weight assignment, the shortest path is calculated with the Dijkstra Algorithm.

3. 4 Publishing the Algorithm as a Web Service

To enable further machine-to-machine communication, the algorithm is published as a web service, which processes POST-requests with the GeoJSON serialization of the geometry for the environment and the endpoints as parameters. The service also returns the GeoJSON serialization of the shortest path. The concrete API is provided in Appendix B.

To facilitate testing, an interactive front-end user interface with the possibility for manual geometry input with map reference is also provided. Figure 3.11 shows an example for the visualized polygon (boundary and obstacles) together with the calculated path.

Figure 3. 11 Interactive User Interface

Chapter 4 Results and Analysis

To test the correctness and the performance of the algorithm, 100 test cases are conducted with field geometries that were derived from historical GPS tracks of agricultural vehicles in previous work.

The following parameters are compared between the proposed algorithm and the brute-force implementation of the visibility graph algorithm:

- number of nodes and edges of the generated graph
- running time for the graph generation and for the routing phase with the Dijkstra algorithm
- length of the shortest path

The test dataset is generated as follows:

1) download the field geometries in GeoJSON format from the field service server
2) select 100 field geometries that have at least one obstacle
3) select a vertex on the outer boundary randomly as the start position
4) select an obstacle randomly and then select a vertex on its outer boundary randomly as the end position

To improve the significance of the comparison, if the beeline from the start position to the end position doesn't cross with the field geometry, step 4 is repeated until it does.

Parameters of the test machine are listed in Table 4.1.

Table 4. 1 Parameters of the Test Machine

Machine Type	13-inch MacBook Pro
Operating System	OS X Yosemite 10.10
Processor	2.4 GHz Intel Core i5

Both algorithms are implemented with Python 2.7. Three third-party packages are used: networkx for graph operation, django and its binding with the C library GEOS for geometric operation, and matplotlib for plotting.

The input data and results of the all the test cases are available in Appendix A.

4. 1 Length of the Shortest Path

Figure 4.1 shows that, when it comes to the length of the calculated shortest path, the

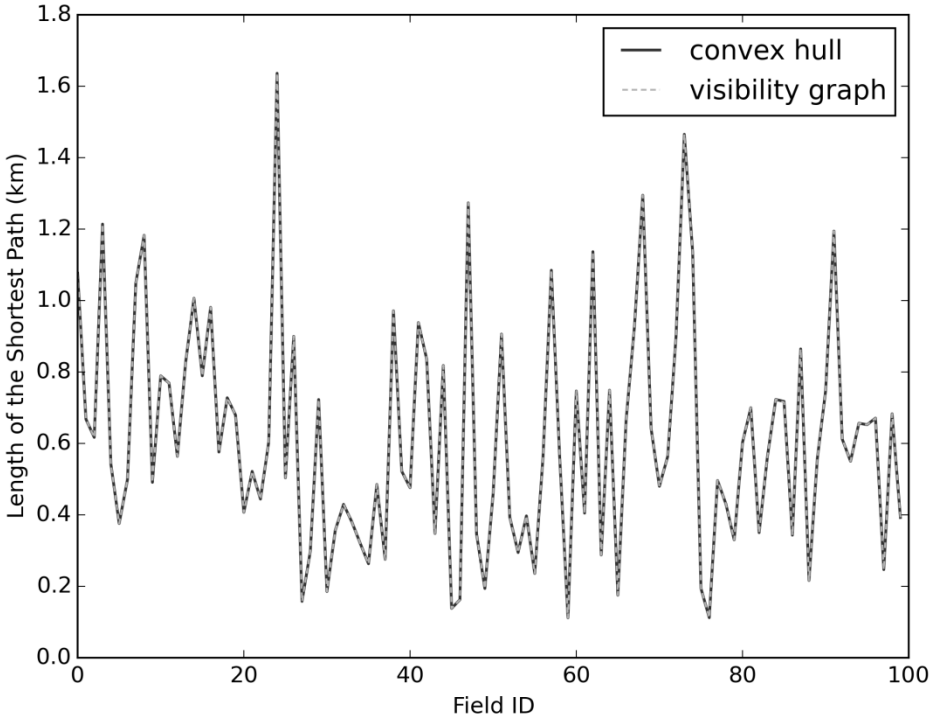


Figure 4. 1 Comparison – Length of the Shortest Path

proposed algorithm and the visibility graph alternative coincide in all the 100 test cases.

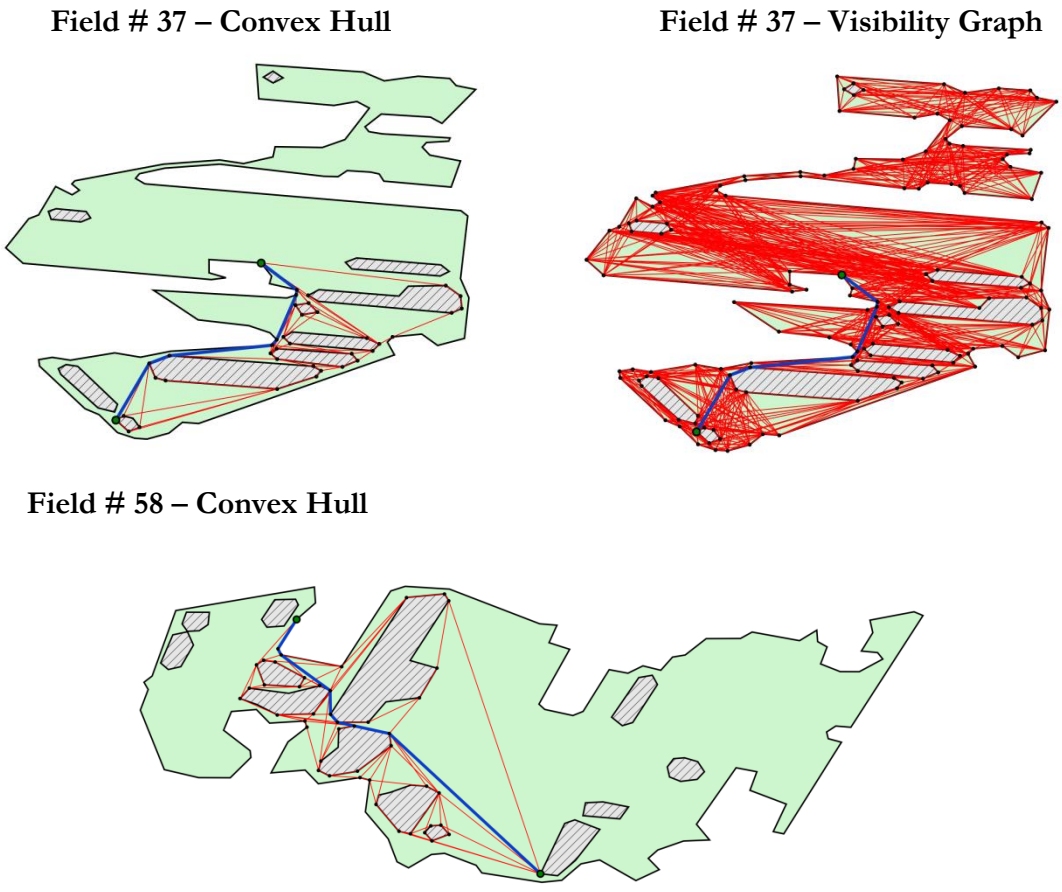


Figure 4. 2 Comparison with Field Examples

As the two concrete examples in Figure 4.2 demonstrate, although the two approaches come to the same final result, the graph generated by the convex hull approach is much more

lightweighted, which contributes to a better performance. The following sections will give a quantitative analysis on the graph size and running time.

4. 2 Graph Size

Since the convex hull approach only takes impeding obstacles into consideration, it normally results in a much smaller graph. As Figure 4.3 and Table 4.2 demonstrate, measured by number of nodes, the graph size in the convex hull approach is about 80% smaller. The reduction of graph size is even more obvious if it is measured by the number of nodes (nearly 96%).

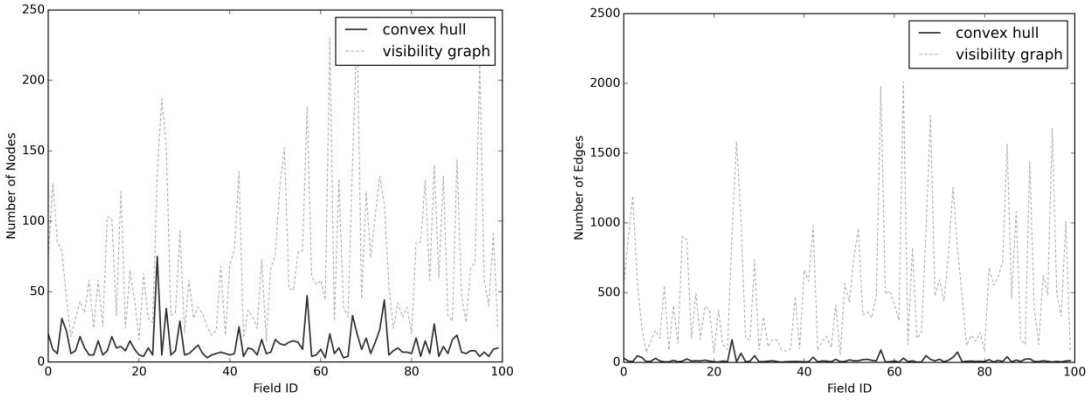


Figure 4. 3 Comparision – Graph Size

Table 4. 2 Number of Nodes and Edges

	min	max	average
Number of Nodes			
Convex Hull	3	75	12
Visibility Graph	15	240	70
Average ratio between the two methods: 0.19696 (convex hull / visibility graph)			
Number of Edges			
Convex Hull	2	162	16
Visibility Graph	55	2011	510
Average ratio between the two methods: 0.04187 (convex hull / visibility graph)			

4. 3 Running Time

As mentioned before, the running time for a one-off query is added up by two parts – graph generation and routing on the graph. These two parts will be analyzed separately in Chapter 4.3.1 and 4.3.2. But for repeating queries, the time complexity for each individual query depends on that whether the graph could be cached in advance and retrieved immediately when needed. Therefore, Chapter 4.3.3 also compares the time complexity of the two methods with caching techniques taken into consideration.

4. 3. 1 Running Time for Graph Generation

In the graph generation stage, the proposed algorithm outperforms the visibility graph algorithm significantly with respect to time complexity (Figure 4.4). On average, the visibility graph approach takes 3.54 seconds to generate the whole graph, while the convex hull option takes only 0.06 seconds (Table 4.3), approximately 20 times faster than the brute-force implementation of the visibility graph algorithm.

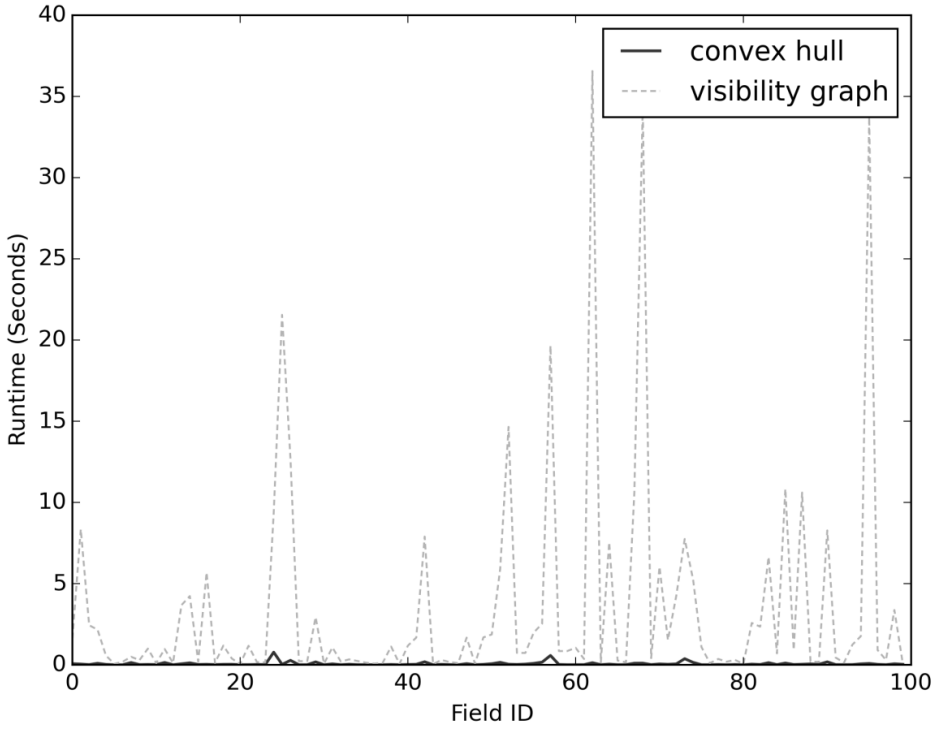


Figure 4. 4 Comparison - Time Complexity for Graph Generation

Table 4. 3 Running Time for Graph Generation

	min (s)	max (s)	average (s)
Convex Hull	0.00317	0.77588	0.05934
Visibility Graph	0.04182	36.56198	3.53935
Average ratio between the two methods: 0.04810 (convex hull / visibility graph)			

This result could be partially explained by the characteristics of the input data: As a result of its complex geometry, each field is represented by many vertices. The running time for visibility graph generation is sensitive to the total number of vertices since it iterates through all pairs of vertices to check if they cross with the given polygon.

To the contrary, the convex hull approach is insensitive to the total number of vertices, and thus has a more stable running time.

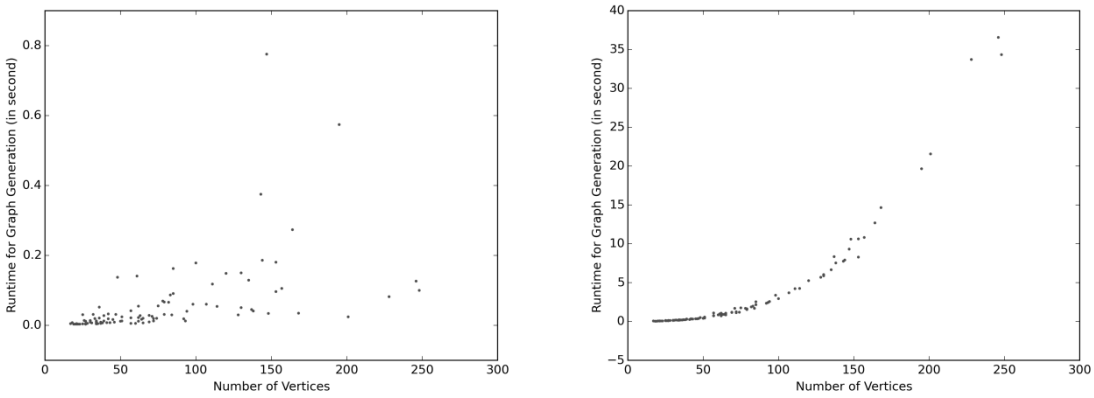


Figure 4. 5 Running Time and Input Size

Figure 4.5 shows how the running time of the two methods evolves as the size of the input data increases. For the visibility graph approach, the running time in the graph generation stage is polynomial (right), whereas the running time of the convex hull approach shows little correlation (left).

4. 3. 2 Running Time for Routing with Dijkstra Algorithm

Since the time complexity of the Dijkstra algorithm implemented with a priority queue is O(E*logV), where E is the number of edges and V the number of nodes, the running time of the routing phase is sensitive to the graph size.

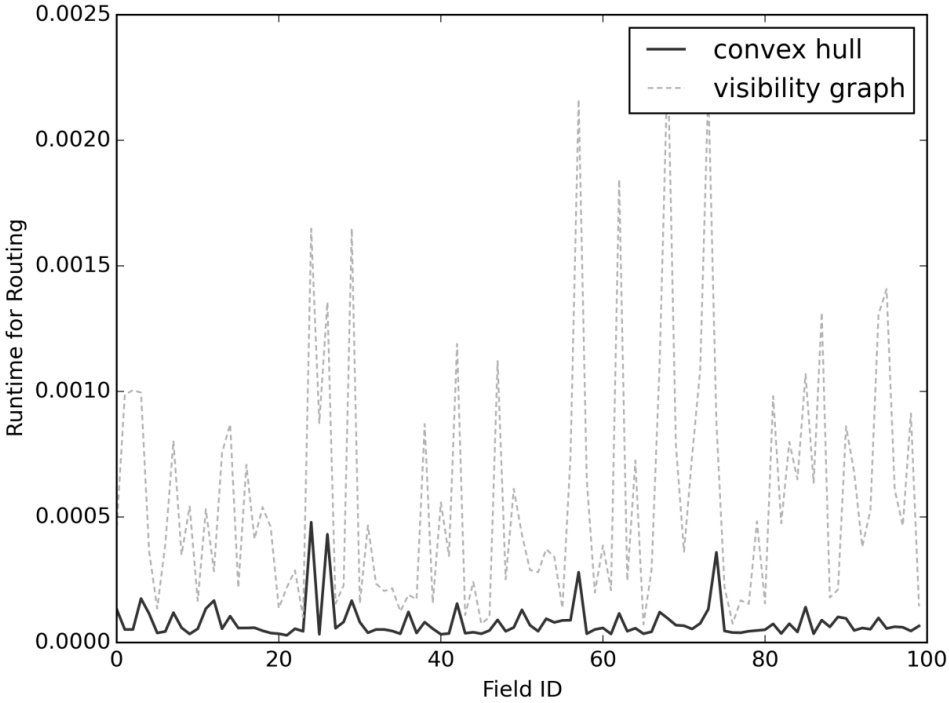


Figure 4. 6 Comparison - Time Complexity for Routing

Table 4. 4 Running Time for Routing

	Min (ms)	**Max (ms)**	**Avg (ms)**
Convex Hull	0.03	0.48	0.08
Visibility Graph	0.07	2.33	0.59
Average ratio between the two methods: 0.20248 (convex hull / visibility graph)			

Because of this reason, the convex hull approach also outperforms the visibility graph approach in the routing phase as a natural consequence of smaller graph size (Figure 4.6). However, as Table 4.4 indicates, the running time difference in the routing phase is far less

significant than it is in the graph generation phase, even the worst case among all the test data takes only less than 3 milliseconds. Compared to the graph generation stage, the running time difference in the routing stage is almost ignorable.

4. 3. 3 Total Running Time Considering Caching

As mentioned in Chapter 2.1, unlike the convex hull approach which requires the re-generation of the graph at every new request, the visibility graph could be generated only once and cached for further requests.

If we take this caching possibility into consideration, the total running time of the visibility graph approach could be reduced remarkably, since it's no longer necessary to iterate through every pair of vertices that are on the original polygon, but only the pairs with one endpoint (start position or end position) and one vertex on the original polygon, which reduces the quadratic running time to a linear one. For the same polygon, we could assume a constant time to determine whether the line segment crosses with a polygon.

Figure 4.7 compares the total running time of the two methods with the precondition that the visibility graphs of all the field geometries are cached and can be retrieved with no additional time cost (e.g. network delay) at each new request. In this case, the running time for generating the entire visibility graph is replaced with the running time for adding the start and end position to the previously cached graph.

Table 4. 5 Running Time Considering Caching

	min (ms)	**max (ms)**	**average (ms)**	**median (ms)**	**standard deviation (ms)**
Convex Hull	3.22	776.36	59.42	23.28	108.07
Visibility Graph	8.32	593.31	100.55	52.19	121.43
Average ration between the two methods: 0.61990 (convex hull / visibility graph)					

Although the two curves in the figure seem on par with each other, the statistical measurements in Table 4.5 show that the convex hull approach is still at a slight advantage in terms of average running time even if we assume that the visibility graphs are cached.

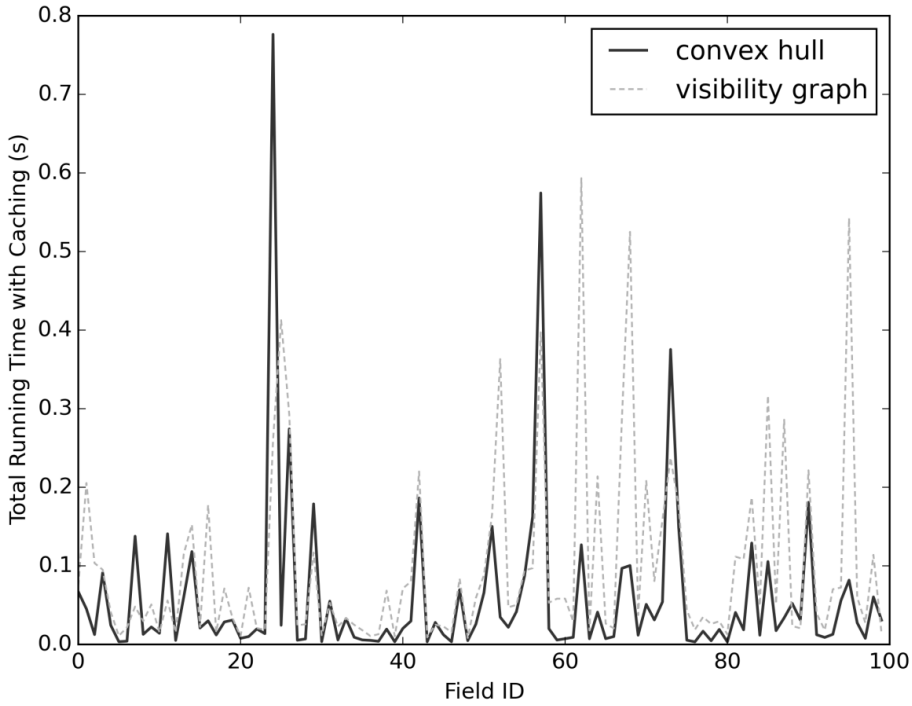


Figure 4. 7 Comparison – Time Complexity Considering Caching

4. 4 Thoughts on Further Optimization

4. 4. 1 Detecting Impeding Obstacles through Spatial Index

The current implementation iterates through all the obstacles to check if they cross with some certain edge on the graph. This doesn't incur serious performance problem on the test data, in which obstacles are in general sparse. In other scenarios with more obstacles, for example, pedestrian routing in urban area which involves many buildings or other constructions, iterating through all the obstacles is obviously inefficient.

In such cases, the query of impeding obstacles can be accelerated by storing the geometries of the obstacles in a spatial database (e.g. PostGIS). Utilizing spatial indices, the time complexity for finding impeding obstacles could be reduced to a logarithmic one.

4. 4. 2 Return the Shorter Path on the Convex Hull

As it is illustrated in Figure 3.7, each generated convex hull includes two paths from the start point to the end point, except for several edge cases where half of it is invalid and should be

excluded. The implementation in this thesis continues to adjust the two paths until both of them are collision-free. Since our goal is to find the shortest path, we could calculate the length of the two paths after each convex hull generation. If the shorter one happens to be collision-free, we could immediately return it and ignore the longer one, since the substitution of one edge with the convex hull would only increase the length of the path as long as the graph is positively-weighted.

However, whether this optimization possibility brings an effective performance gain is dependent on the spatial distribution of obstacles. In the worst case, where most convex hulls generated have two colliding paths, it could even increase the total running time due to the overhead of distance calculation.

Chapter 5 Conclusion

5. 1 Contributions

This thesis has three major contributions:

1) a robust Python implementation of the proposed algorithm, which also handles edge cases with concavity decently

2) an in-depth discussion about its suitability for in-field routing based on a thorough comparison with other possible approaches

3) the publishing of the algorithm as a web service

5. 2 Future Work

However, the description and implementation of the infield-routing algorithm is only the first step towards building an efficient real-time system for both in-field and inter-field navigation. Several avenues for future research are outlined below:

- integration with inter-field navigation

The current implementation of the algorithm presumes that both the start position and the end position are interior to the field geometry. In real world scenarios, the route of an agricultural vehicle usually begins from outside the field. The in-field route and inter-field route are usually connected through an entry point to the field. If the entry point is not unique, the in-field routing task is subject to multiple sources. A naïve integration involves

calculating the shortest path for each entry point, and adopting the entry point with the shortest in-field travel distance.

- derivation of the real-time passable area

During a harvest campaign, the passable area changes from time to time, since the new harvested area will become passable to other agricultural vehicles. Although including these real-time changes will increase the precision of the navigation system, it raises also greater challenges to the computational efficiency of the system.

- different graph generation approaches for CTF systems

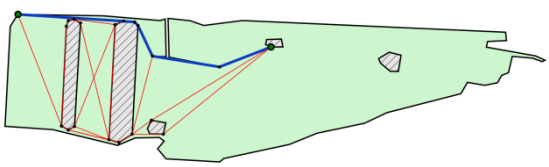
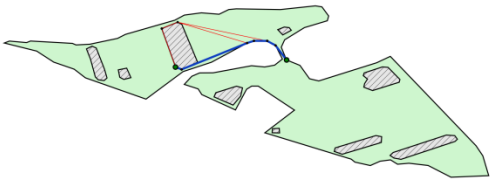
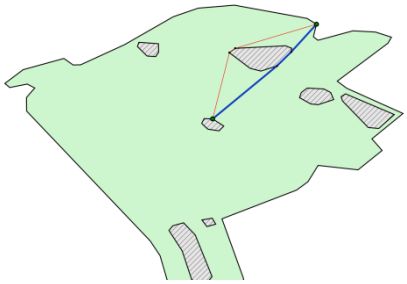
In a strictly Controlled Traffic Farming system, both the convex hull approach and the visibility graph option are not feasible, since none of them limits turning to headland. For such systems, the graph should be derived from the major operation direction and the headland geometry. For a partial CTF system, where the crossing with established parallel tracks is allowed but not desirable, we can reflect the conformity with the major operation direction of a path through weighting.

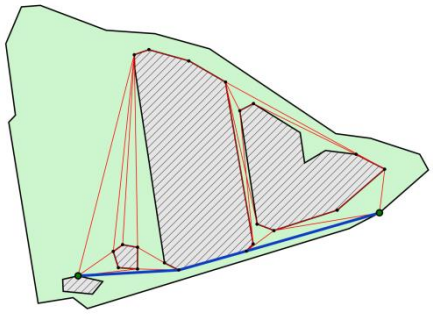
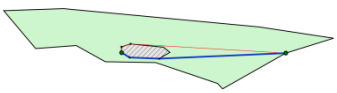
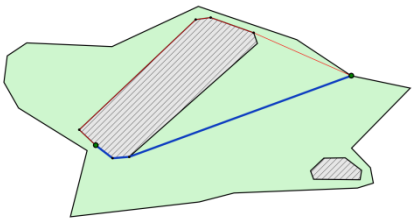
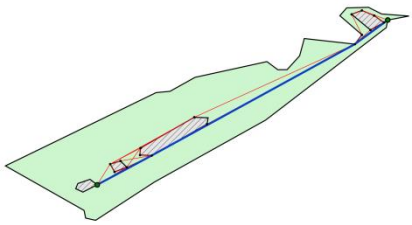
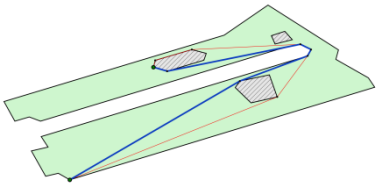
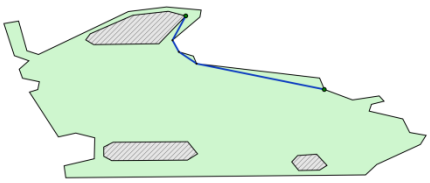
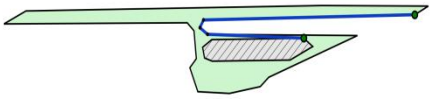
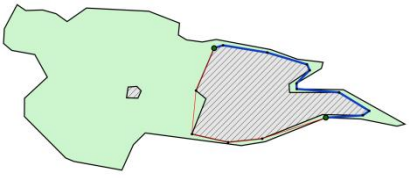
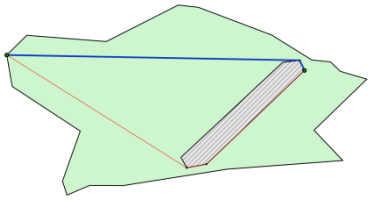
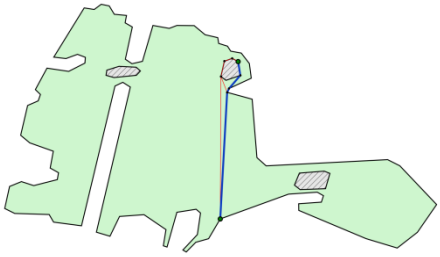
Chapter 6 Summary

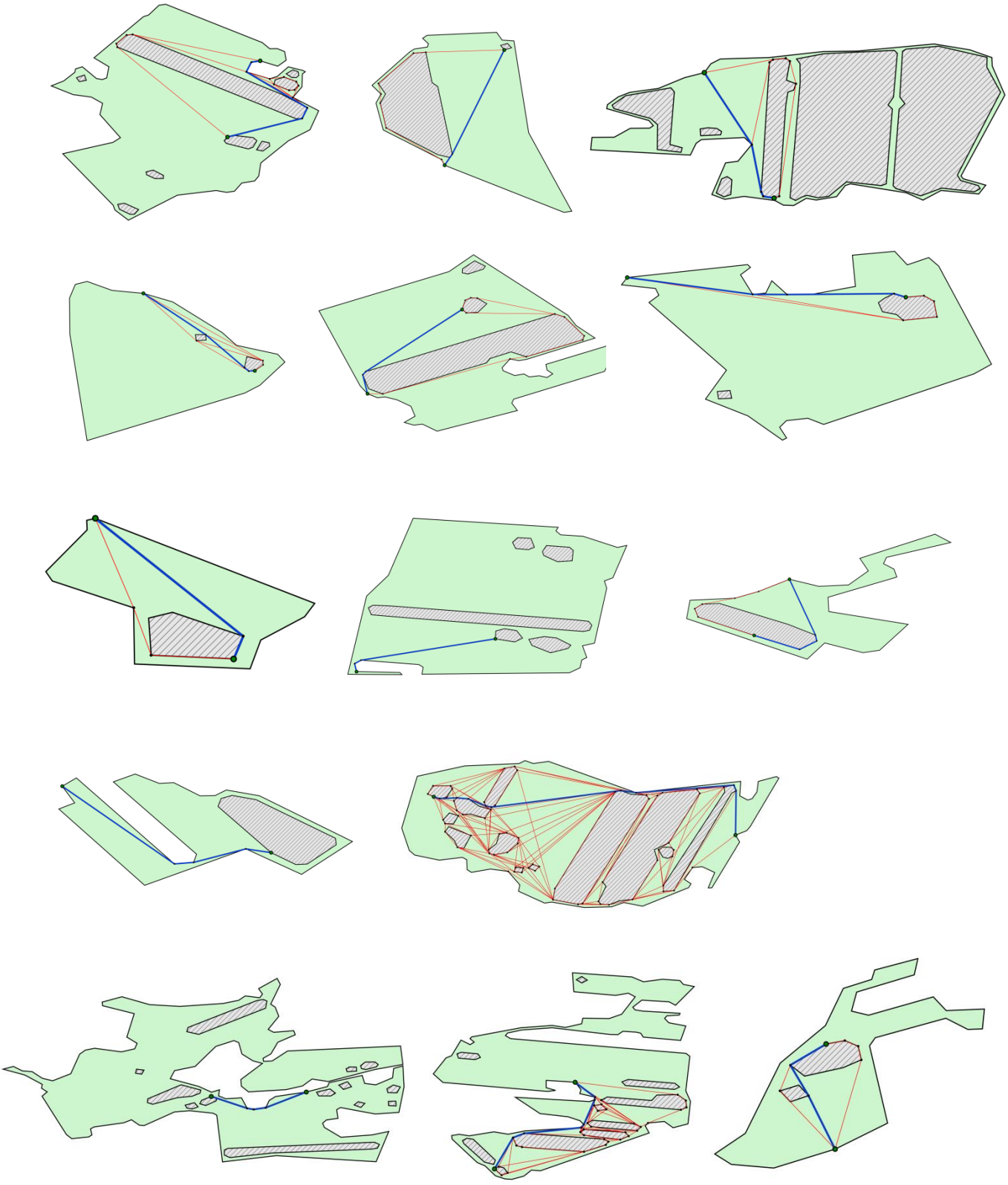
This thesis proposes and implements an approximation algorithm for finding the shortest path on two-dimensional surfaces with boundaries and obstacles. The algorithm is especially suitable for application scenarios with a large n/h value, with n representing the total number of vertices and h the total number of obstacles.

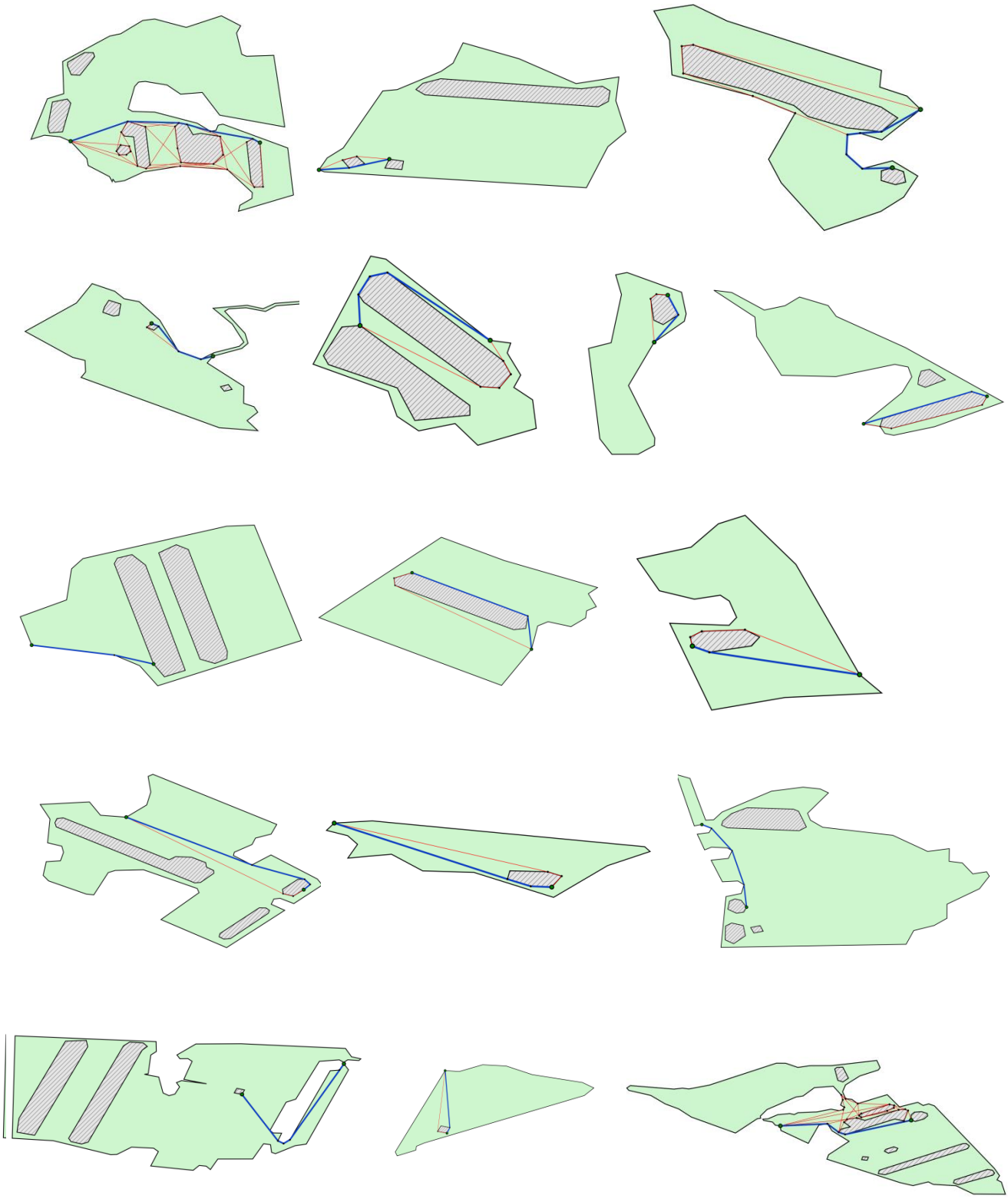
Test results from infield-routing show that, the algorithm could to a great possibility deliver the globally optimal result while reducing the running time significantly.

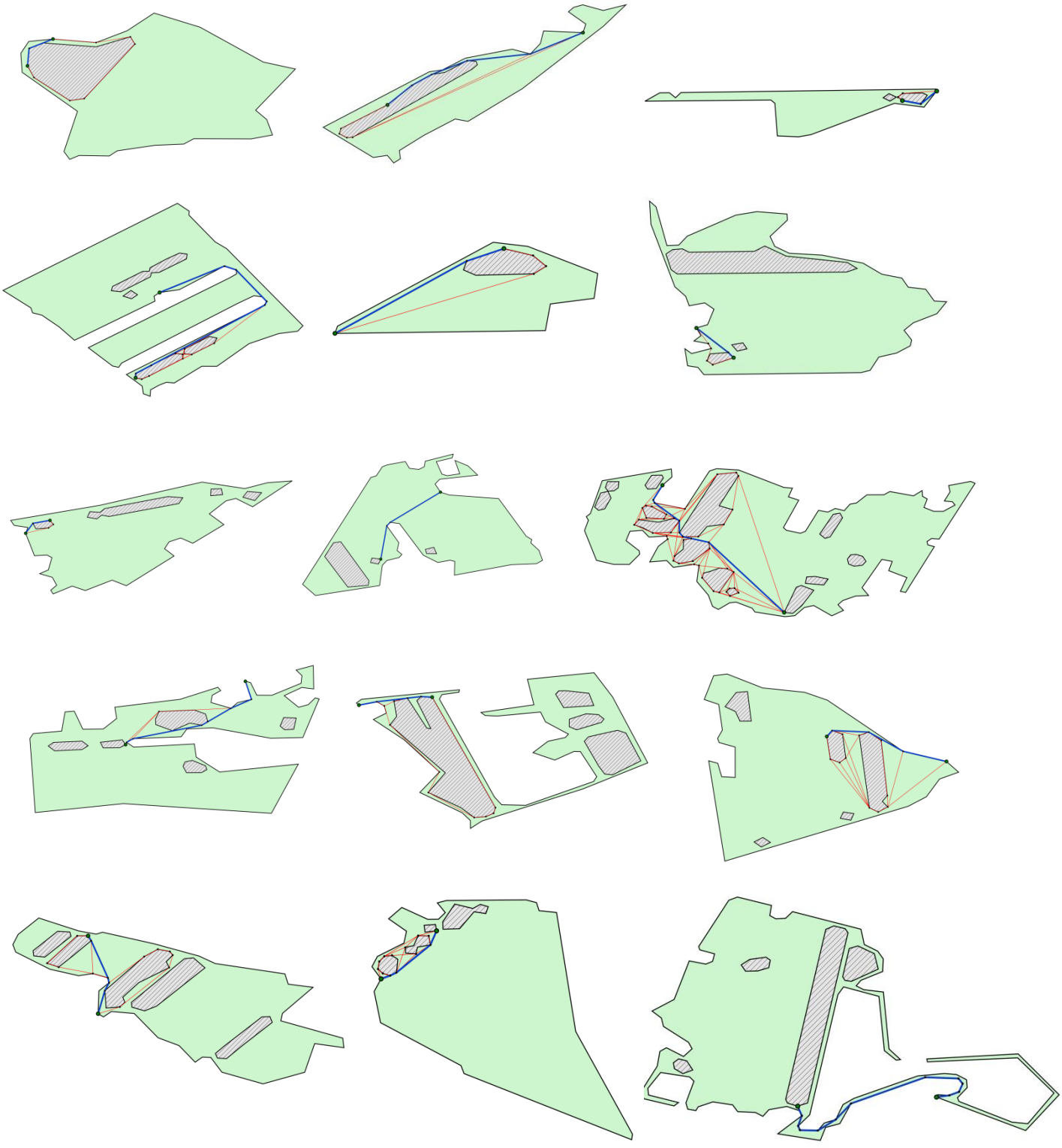
Appendix A Complete Test Results

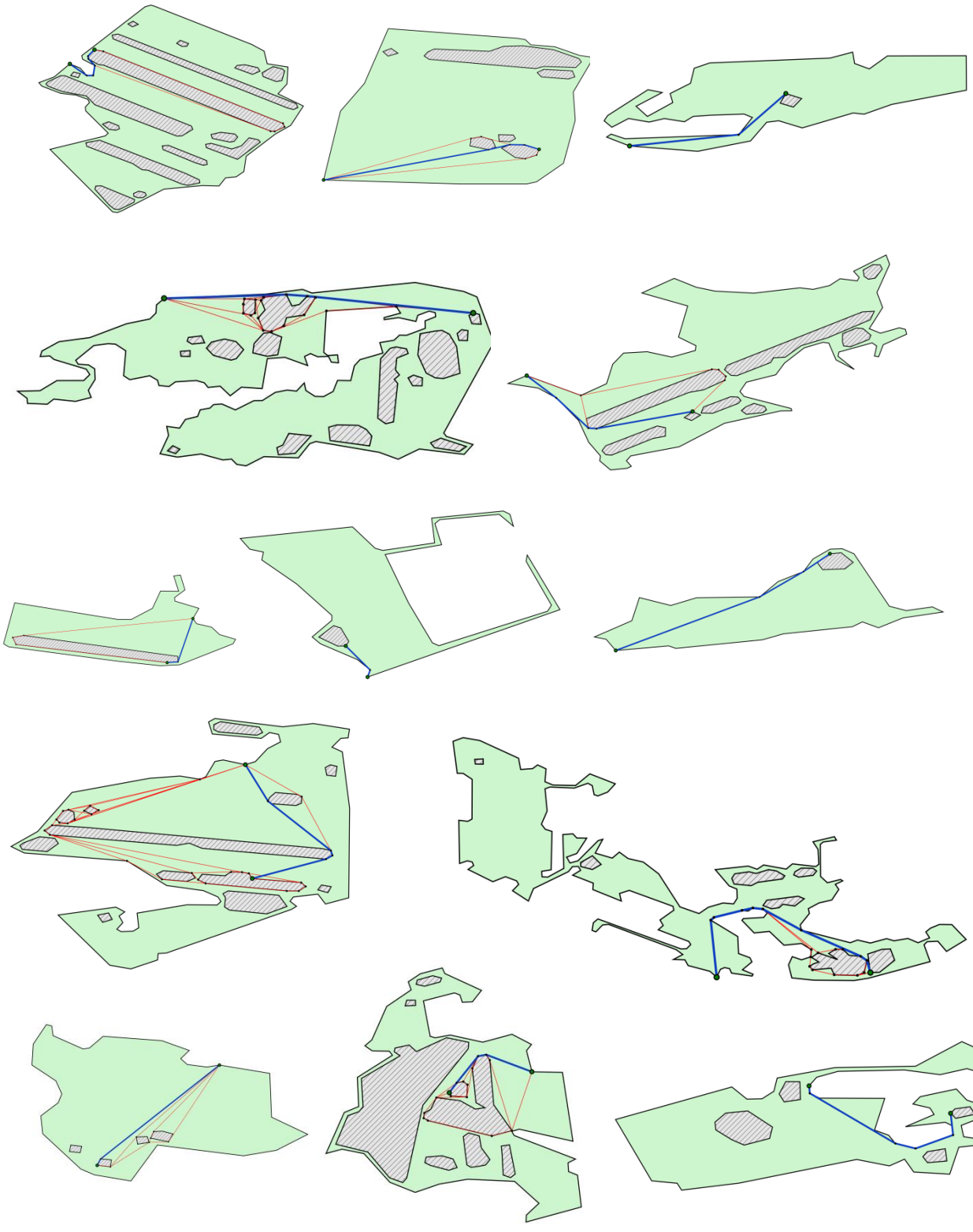




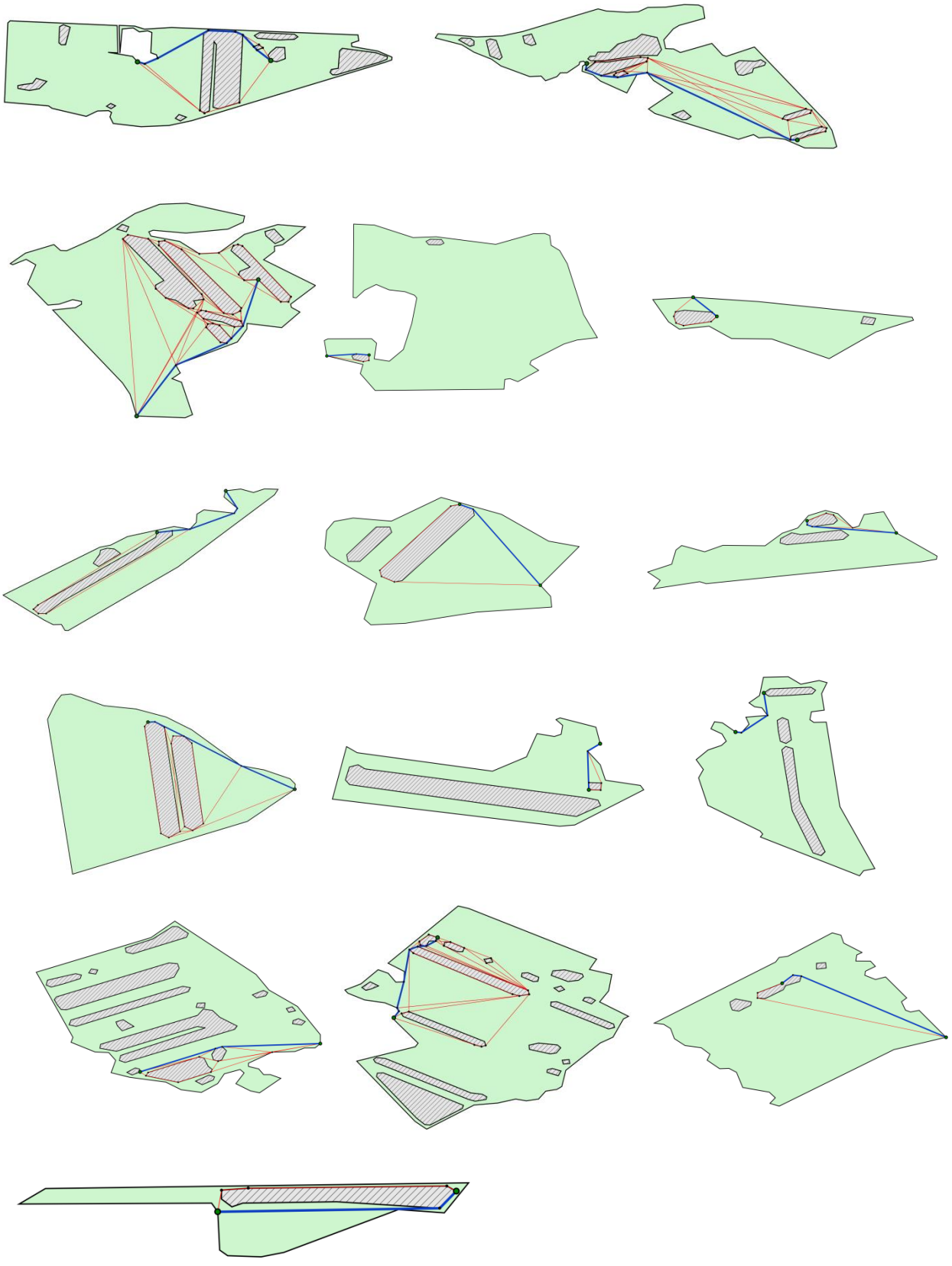











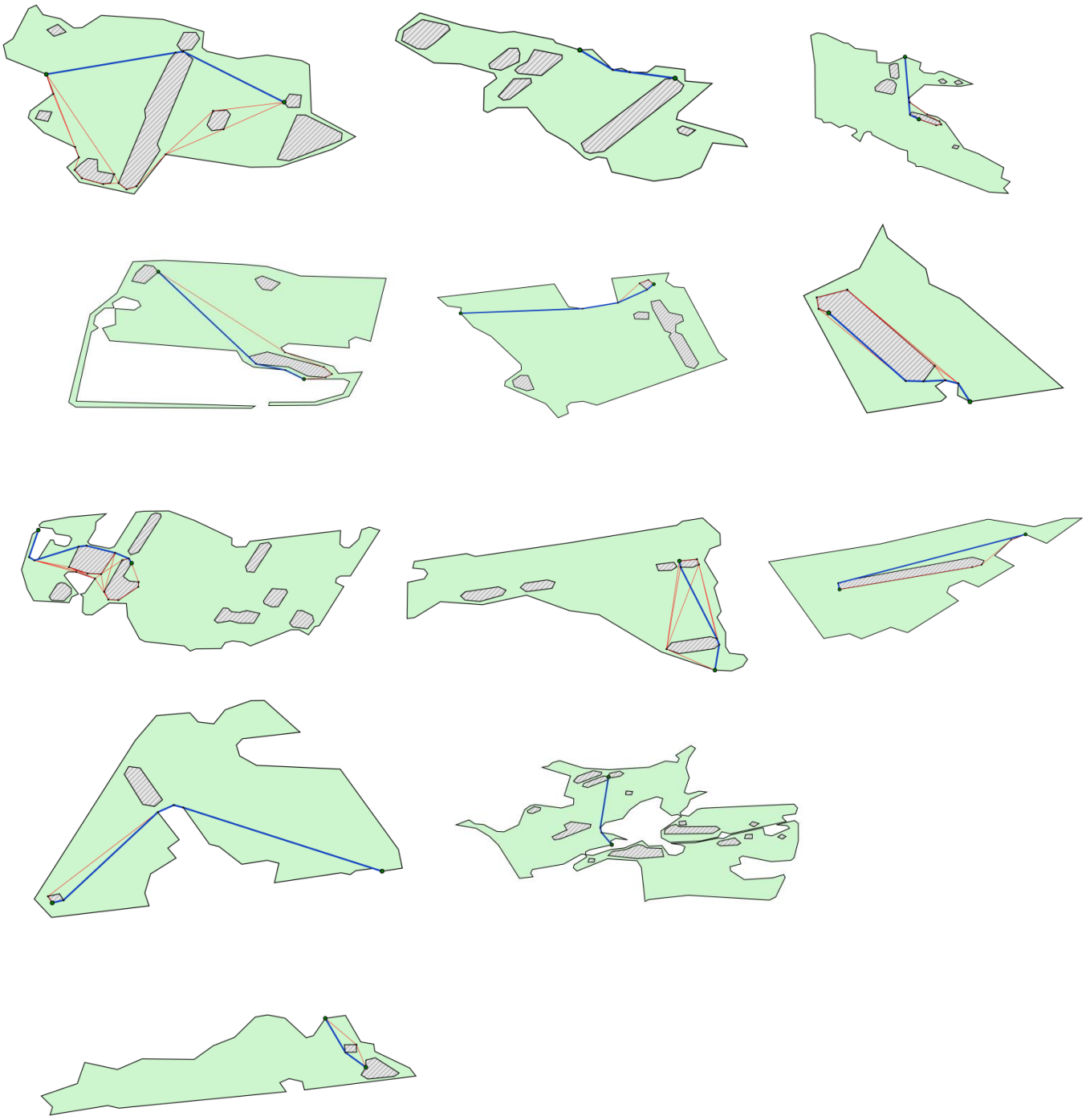












Appendix B API for the Published Web Service

Request:


```
{
  "polygonGeom":{
    "type":"Feature",
    "properties":{},
    "geometry":{
      "type":"Polygon",
      "coordinates":[[[8.67396354675293,49.4213032372524],
[8.675851821899412,49.414379560443805],[8.688039779663086,49.41516131480
479],[8.688039779663086,49.42091241054287],[8.67396354675293,49.42130323
72524]],[[8.680744171142576,49.41895823030466],[8.678169250488281,49.415
83138006123],[8.686323165893555,49.41812070066643],[8.680744171142576,49
.418958230304661]]]
    }
  },
  "endPoints":[
    {
      "type":"Feature",
      "properties":{},
      "geometry":{
        "type":"Point",
        "coordinates":[8.677911758422852,49.41739483008498]
      }
    },
    {
      "type":"Feature",
      "properties":{},
      "geometry":{
        "type":"Point",
        "coordinates":[8.683919906616211,49.415998894945986]
      }
    }
  ]
}
```


Response (Success):


```
{
  "status":"OK",
  "path":{
    "type": "LineString",
    "coordinates": [[8.677911758422852,49.41739483008498],
[8.678169250488281,49.415831380061228],[8.683919906616211,49.41599889494
5986]]
  }
}
```

Response (Error):

```
{
  "status":"ERROR",
  "message":”Error message from the server.”
}
```

Literature

[1] Ali, O., Verlinden, B., & Van Oudheusden, D. (2009). Infield logistics planning for crop-harvesting operations. *Engineering Optimization, 41*(2), 183-197.

[2] Bac, C. W., Henten, E. J., Hemming, J., & Edan, Y. (2014). Harvesting Robots for High ‐ value Crops: State ‐ of ‐ the ‐ art Review and Challenges Ahead. *Journal of Field Robotics*.

[3] Bakker, T., Asselt van, K., Bontsema, J., Müller, J., & Straten van, G. (2010). Systematic design of an autonomous platform for robotic weeding. *Journal of Terramechanics, 47*(2), 63-73. doi: http://dx.doi.org/10.1016/j.jterra.2009.06.002

[4] Belforte, G., Deboli, R., Gay, P., Piccarolo, P., & Ricauda Aimonino, D. (2006). Robot design and testing for greenhouse applications. *Biosystems Engineering, 95*(3), 309-321.

[5] Berg, M. d. (Ed.). (2008). *Computational geometry : algorithms and applications* (3. ed. ed.). Berlin ; Heidelberg: Springer.

[6] Bochtis, D. D., Sørensen, C. G., Green, O., Moshou, D., & Olesen, J. (2010). Effect of controlled traffic on field efficiency. *Biosystems Engineering, 106*(1), 14-25. doi: http://dx.doi.org/10.1016/j.biosystemseng.2009.10.009

[7] Bochtis, D. D., Sørensen, C. G., & Vougioukas, S. G. (2010). Path planning for in-field navigation-aiding of service units. *Computers and Electronics in Agriculture, 74*(1), 80-90. doi: http://dx.doi.org/10.1016/j.compag.2010.06.008

[8] Bochtis, D. D., Sørensen, C. G. C., & Busato, P. (2014). Advances in agricultural machinery management: A review. *Biosystems Engineering, 126*(0), 69-81. doi: http://dx.doi.org/10.1016/j.biosystemseng.2014.07.012

[9] Bochtis, D. D., & Vougioukas, S. G. (2008). Minimising the non-working distance travelled by machines operating in a headland field pattern. *Biosystems Engineering, 101*(1), 1-12. doi: http://dx.doi.org/10.1016/j.biosystemseng.2008.06.008

[10]Demmel, M., Brandhuber, R., Kirchmeier, H., Mueller, M., & Marx, M. *Controlled traffic farming in Germany–technical and organisational realisation and first results*.

[11]Devadoss, S. L., & O'Rourke, J. (2011). *Discrete and computational geometry*: Princeton University Press.

[12]Dieter Kutzbach, H. (2000). Trends in Power and Machinery. *Journal of Agricultural Engineering Research, 76*(3), 237-247. doi: http://dx.doi.org/10.1006/jaer.2000.0574

[13]Fagerholt, K., Heimdal, S. I., & Loktu, A. (2000). Shortest Path in the Presence of Obstacles: An Application to Ocean Shipping. *The Journal of the Operational Research Society, 51*(6), 683-688. doi: 10.2307/254011

[14]Ferguson, D., & Stentz, A. (2006). Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics, 23*(2), 79-101.

[15]Gajentaan, A., & Overmars, M. H. (1995). On a class of O (n 2) problems in computational geometry. *Computational geometry, 5*(3), 165-185.

[16]González, R., Rodríguez, F., Sánchez-Hermosilla, J., & Donaire, J. (2009). Navigation techniques for mobile robots in greenhouses. *Applied Engineering in Agriculture, 25*(2), 153.

[17]Grift, T., Zhang, Q., Kondo, N., & Ting, K. (2008). A review of automation and robotics for the bioindustry. *Journal of Biomechatronics Engineering, 1*(1), 37-54.

[18]Hamza, M. A., & Anderson, W. K. (2005). Soil compaction in cropping systems: A review of the nature, causes and possible solutions. *Soil and Tillage Research, 82*(2), 121-145. doi: http://dx.doi.org/10.1016/j.still.2004.08.009

[19]Han, S., Zhang, Q., Ni, B., & Reid, J. F. (2004). A guidance directrix approach to vision-based vehicle guidance systems. *Computers and Electronics in Agriculture, 43*(3), 179-195. doi: http://dx.doi.org/10.1016/j.compag.2004.01.007

[20]Hershberger, J. (1989). An optimal visibility graph algorithm for triangulated simple polygons. *Algorithmica, 4*(1-4), 141-155. doi: 10.1007/BF01553883

[21]Hershberger, J., & Suri, S. (1999). An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing, 28*(6), 2215-2256.

[22]Holpp, M., Anken, T., Sauter, M., Rek, J., Reiser, R., Zihlmann, U., . . . Hensel, O. (2011). Swiss Controlled Traffic Farming Trial–Preliminary Results 2008-2010. *Precision agriculture*.

[23]Hong, I., & Murray, A. T. (2013). Efficient measurement of continuous space shortest distance around barriers. *International Journal of Geographical Information Science, 27*(12), 2302-2318. doi: 10.1080/13658816.2013.788182

[24]Huang, H.-P., & Chung, S.-Y. (2004). *Dynamic visibility graph for path planning*. Paper presented at the Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on.

[25]Hunt, D. (2008). *Farm power and machinery management*: Waveland Press.

[26]Jensen, M. A. F., Bochtis, D., Sørensen, C. G., Blas, M. R., & Lykkegaard, K. L. (2012). In-field and inter-field path planning for agricultural transport units. *Computers & Industrial Engineering, 63*(4), 1054-1061. doi: http://dx.doi.org/10.1016/j.cie.2012.07.004

[27]Kapoor, S., Maheshwari, S., & Mitchell, J. S. (1997). An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete & Computational Geometry, 18*(4), 377-383.

[28]Kutz, M. (2013). *Handbook of farm, dairy and food machinery engineering*: Academic Press.

[29]Latombe, J.-C. (1991). Robot Motion Planning.

[30]Li, F., & Klette, R. (2011). *Euclidean Shortest Paths*: Springer.

[31]Li, M., Imou, K., Wakabayashi, K., & Yokoyama, S. (2009). Review of research on agricultural vehicle autonomous guidance. *International Journal of Agricultural and Biological Engineering, 2*(3), 1-16.

[32]Linker, R., & Blass, T. (2008). Path-planning algorithm for vehicles operating in orchards. *Biosystems Engineering, 101*(2), 152-160. doi: http://dx.doi.org/10.1016/j.biosystemseng.2008.06.002

[33]Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM, 22*(10), 560-570.

[34]Mitchell, J. S. (2000). Geometric shortest paths and network optimization. *Handbook of computational geometry, 334*, 633-702.

[35]O'rourke, J. (1987). *Art gallery theorems and algorithms* (Vol. 57): Oxford University Press Oxford.

[36]Oksanen, T., & Visala, A. (2007). Path planning algorithms for agricultural machines.

[37]Oksanen, T., & Visala, A. (2009). Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics, 26*(8), 651-668.

[38]Pedersen, S. M., Fountas, S., Have, H., & Blackmore, B. (2006). Agricultural robots—system analysis and economic feasibility. *Precision agriculture, 7*(4), 295-308.

[39]Pilarski, T., Happold, M., Pangels, H., Ollis, M., Fitzpatrick, K., & Stentz, A. (2002). The demeter system for automated harvesting. *Autonomous Robots, 13*(1), 9-20.

[40]Qingjie, W., Hao, C., Hongwen, L., Wenying, L., Xiaoyan, W., McHugh, A., . . . Huanwen, G. (2009). Controlled traffic farming with no tillage for improved fallow water storage and crop yield on the Chinese Loess Plateau. *Soil and Tillage Research, 104*(1), 192-197.

[41]Reid, J. F., & Searcy, S. W. (1991). An algorithm for computer vision sensing of a row crop guidance directrix: SAE Technical Paper.

[42]Reid, J. F., Zhang, Q., Noguchi, N., & Dickson, M. (2000). Agricultural automatic guidance research in North America. *Computers and Electronics in Agriculture, 25*(1–2), 155-167. doi: http://dx.doi.org/10.1016/S0168-1699(99)00061-7

[43]Scheuren, S., Hertzberg, J., Stiene, S., Hartanto, R., & Center, R. I. (2013). *Infield Path Planning for Autonomous Unloading Vehicles*. Paper presented at the GIL Jahrestagung.

[44]Slaughter, D. C., Giles, D. K., & Downey, D. (2008). Autonomous robotic weed control systems: A review. *Computers and Electronics in Agriculture, 61*(1), 63-78. doi: http://dx.doi.org/10.1016/j.compag.2007.05.008

[45]Søgaard, H. T., & Sørensen, C. G. (2004). A Model for Optimal Selection of Machinery Sizes within the Farm Machinery System. *Biosystems Engineering, 89*(1), 13-28. doi: http://dx.doi.org/10.1016/j.biosystemseng.2004.05.004

[46]Sørensen, C. G., & Bochtis, D. D. (2010). Conceptual model of fleet management in agriculture. *Biosystems Engineering, 105*(1), 41-50. doi: http://dx.doi.org/10.1016/j.biosystemseng.2009.09.009

[47]Spekken, M., & de Bruin, S. (2013). Optimized routing on agricultural fields by minimizing maneuvering and servicing time. *Precision agriculture, 14*(2), 224-244. doi: 10.1007/s11119-012-9290-5

[48]Suprem, A., Mahalik, N., & Kim, K. (2013). A review on application of technology systems, standards and interfaces for agriculture and food sector. *Computer Standards & Interfaces, 35*(4), 355-364. doi: http://dx.doi.org/10.1016/j.csi.2012.09.002

[49]Taylor, J. H. (1983). Benefits of permanent traffic lanes in a controlled traffic crop production system. *Soil and Tillage Research, 3*(4), 385-395. doi: http://dx.doi.org/10.1016/0167-1987(83)90040-5

[50]Toth, C. D., O'Rourke, J., & Goodman, J. E. (2004). *Handbook of discrete and computational geometry*: CRC press.

[51]Tullberg, J. N., Yule, D. F., & McGarry, D. (2007). Controlled traffic farming—From research to adoption in Australia. *Soil and Tillage Research, 97*(2), 272-281. doi: http://dx.doi.org/10.1016/j.still.2007.09.007

[52]Vermeulen, G., Tullberg, J., & Chamen, W. (2010). Controlled traffic farming *Soil Engineering* (pp. 101-120): Springer.

[53]Viegas, J., & Hansen, P. (1985). Finding shortest paths in the plane in the presence of barriers to travel (for any lp - norm). *European Journal of Operational Research, 20*(3), 373-381. doi: http://dx.doi.org/10.1016/0377-2217(85)90010-4

[54]Welzl, E. (1985). Constructing the visibility graph for n-line segments in O (n< sup> 2</sup>) time. *Information Processing Letters, 20*(4), 167-171.

[55]Witney, B. (1988). *Choosing and using farm machines*: Longman.

[56]Zandonadi, R. S. (2012). Computational Tools for Improving Route Planning in Agricultural Field Operations.